UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

“Desarrollo de sistemas domóticos bajo el paradigma del Internet de las Cosas incorporando una tarjeta de desarrollo con ESP32”

TESIS

Para obtener el título de:

Ingeniero Eléctrico Electrónico

Presenta:

Eduardo Olmos Castillo

Asesor de tesis:

Dr. Ismael Díaz Rangel

Ciudad Nezahualcóyotl, Estado de México, 2023

AGRADECIMIENTOS

En primer lugar, me gustaría expresar mi más profundo agradecimiento a mis padres. Quiero honrar la memoria de mi padre, quien, gracias a sus enseñanzas, aprendí la importancia de seguir mis objetivos y alcanzar mis metas con perseverancia y determinación, así también, me inculco valores como la bondad, la responsabilidad, el respeto, la tolerancia y la solidaridad, hacia los demás, cualidades que forman parte de mi personalidad y me hacen ser la persona que soy, espero que pueda ver con mucho orgullo lo que he logrado, gracias, papá.

A lo largo de esta emocionante travesía, mi madre ha sido mi mayor apoyo incondicional. Su amor inquebrantable, paciencia y aliento constante han sido pilares fundamentales en cada paso que he dado. Agradezco de todo corazón sus palabras de aliento y la confianza que siempre ha depositado en mí.

No puedo pasar por alto el apoyo de mi adorada hermana, así como mis tíos, primos y sobrinos. Su fe en mí y su constante estimulo han sido un verdadero regalo. Espero sinceramente ser un ejemplo a seguir para los más jóvenes de la familia, especialmente mis sobrinos, y que encuentren en mí la inspiración para perseguir sus propios sueños.

Quiero expresar mi gratitud a mi asesor el doctor, Ismael Díaz Rangel, cuya orientación, experiencia y dedicación fueron fundamentales para el éxito de este proyecto. Su guía y sus valiosos aportes fueron cruciales en el desarrollo de mi investigación.

También deseo agradecer a todos los buenos profesores que he tenido a lo largo de mi trayectoria académica. Su pasión por la enseñanza, su conocimiento y su compromiso con el aprendizaje han dejado una huella profunda en mi formación. Estoy agradecido por todas las oportunidades de aprendizaje que me brindaron y por su constante apoyo en mi camino hacia el crecimiento académico y personal.

Por último, pero no menos importante, quiero extender mi gratitud a todos mis buenos amigos, compañeros de escuela y trabajo. Su apoyo inmensurable, su confianza en mis capacidades y sus ánimos constantes han sido fundamentales para mantenerme motivado en momentos desafiantes. Agradezco de todo corazón su amistad y la creencia en todo lo que puedo lograr.

A todos ustedes, mi familia, asesor, profesores y amigos, les debo un profundo agradecimiento. Sin su apoyo, este logro no hubiera sido posible. Gracias por estar a mi lado en esta etapa tan importante de mi vida.

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN

Los microcontroladores son un componente esencial en la creación de proyectos con Internet de las Cosas (IoT, por sus siglas en inglés) y en la domótica. Estos dispositivos son pequeños sistemas de control integrados que contienen un microprocesador, memorias y periféricos, todo en un solo chip. Los microcontroladores están diseñados para ser eficientes en términos de energía y tamaño, lo que los hace ideales para una amplia gama de aplicaciones de automatización y control. Estos se pueden programar para realizar tareas específicas, lo que les permite interactuar con otros dispositivos, y así proporcionar información de utilidad.

Tarjetas de desarrollo como, por ejemplo, las que integran un microcontrolador ESP32 son ampliamente utilizadas en multitud de proyectos, debido a sus características técnicas y bajo costo. Estas tarjetas disponen de una integración rápida y fácil a conexiones inalámbricas, para ser controladas remotamente a través de la red, lo que permite a los usuarios monitorear y controlar dispositivos IoT y sistemas de automatización doméstica.

Las tarjetas con ESP32 permiten el procesamiento de datos y la ejecución de tareas de manera eficiente, admiten una amplia variedad de periféricos a través de la gran cantidad de pines GPIO que tienen disponibles, así como integrar conectividad Wifi y Bluetooth, ideal para aplicaciones de conectividad inalámbrica. Así también el ESP32 es compatible con una variedad entornos de programación como ESP-IDF, IDE Arduino, MicroPython, entre otros. Cuenta con una gran cantidad de bibliotecas y recursos en línea, que facilitan su programación e incorporación en proyectos.

En síntesis, los microcontroladores como el ESP32 son un elemento útil en la creación de proyectos del Internet de las Cosas y sistemas de automatización doméstica, debido a su tamaño compacto, eficiencia energética, alto rendimiento, costo de implementación bajo, amplia compatibilidad de periféricos y de programación, así como proveer de control remoto. Por lo que en este trabajo se explica cómo desarrollar soluciones IoT empleando tarjetas con un ESP32, comenzando desde lo más básico como lo es la descripción de estas, instalación y uso de un entorno de desarrollo, pasando por el desarrollo de circuitos de adquisición de datos y también de actuadores con sus respectivas etapas de potencia, y también se incluye la introducción y desarrollo de aplicaciones móviles IoT mediante una plataforma gratuita.

1.1 Objetivo general

Desarrollar sistemas domóticos bajo el paradigma del Internet de las Cosas, utilizando una tarjeta de desarrollo basada en un microcontrolador.

1.2 Objetivos particulares

- Realizar el monitoreo remoto de variables como luz, temperatura, humedad ambiental, humedad en sustratos, nivel de agua, presencia, presión atmosférica, altitud e índice de radiación ultravioleta.
- Incorporar un sistema que permita el control de accesos mediante RFID.
- Utilizar una plataforma IoT con servicios gratuitos.

- Realizar propuestas de control remoto de algunas variables.
- Permitir el funcionamiento sin acceso a Internet.
- Integrar el uso de asistentes virtuales para el control.

1.3 Actividades

- Investigar los conceptos de domótica e Internet de las Cosas.
- Conocer las características del microcontrolador ESP32 y las principales tarjetas de desarrollo que se basan en este microcontrolador.
- Integrar la tarjeta de desarrollo con microcontrolador ESP32 al entorno de programación de Arduino.
- Identificar y cargar bibliotecas para el uso de sensores y actuadores soportados en el entorno de programación de Arduino y compatibles con la tarjeta con ESP32.
- Investigar opciones y funcionamiento de sensores de las variables consideradas para esta propuesta.
- Estudiar y seleccionar actuadores para aplicaciones domóticas.
- Identificar etapas de potencia para los actuadores seleccionados.
- Investigar opciones de servicios web gratuitos orientados a el Internet de las Cosas.
- Conocer opciones de servicios web gratuitos para su empleo con la ESP32 y su integración a asistentes virtuales como asistente de Google o Amazon Alexa.
- Desarrollar experimentos para el monitoreo y control de variables a distancia.
- Crear experimentos para el control de variables usando asistentes virtuales.
- Implementar algoritmos que permitan trabajar sin depender de una conexión a Internet.

1.4 Justificación

Para el desarrollo de sistemas que cumplan con el paradigma de Internet de las Cosas, una de las opciones con la mejor relación costo-beneficio corresponde a las tarjetas de desarrollo basadas en el microcontrolador ESP32, que en términos generales es de muy bajo costo, así mismo cuenta con una cantidad amplia de recursos de desarrollo de software/programación para facilitar el trabajar con estas; sin embargo, también es verdad que para aquellos que se inician en el desarrollo de sistemas de esta naturaleza, la información aún no es tan amplia y hay un gran número de consideraciones implicadas y que provocan que los proyectos abordados no funcionen como se espera; por mencionar algunos inconvenientes, se tienen múltiples restricciones de uso de las terminales de la ESP32, como por ejemplo uso del ADC que no es funcional en algunos pines cuando se habilita la conectividad Wifi, o la incompatibilidad de los niveles de tensiones de algunos sensores, algunas bibliotecas pueden provocar problemas entre sí al momento de requerir hacer uso de sus funciones.

La información contenida en este trabajo puede ser de gran ayuda, ya que aborda un gran número de variantes y solución a diferentes problemáticas potenciales en el desarrollo de aplicaciones IoT con la ESP32 bajo el entorno de programación de Arduino.

1.5 Descripción del capitulado

El capítulo 1 destaca la importancia de los microcontroladores en la creación de proyectos IoT y de domótica, destacando las ventajas de su eficiencia energética, tamaño compacto, bajo costo y amplia compatibilidad. El objetivo general del trabajo es proponer el desarrollo de sistemas domóticos bajo el paradigma del IoT, utilizando una tarjeta de desarrollo basada en un microcontrolador, específicamente el ESP32. Se establecen objetivos particulares que incluyen la creación de algoritmos de adquisición de variables, procesamiento y comunicación, el monitoreo remoto de variables, el control de accesos mediante RFID, la utilización de una plataforma IoT con servicios gratuitos, la capacidad de funcionamiento sin acceso a Internet y la integración del uso de asistentes virtuales para el control.

En el capítulo 2 se presenta el marco teórico que sustenta el desarrollo de este trabajo. Se inicia con una explicación detallada de los conceptos de domótica e IoT y su relación con el proyecto. Se describe cómo la domótica y el IoT han revolucionado la forma en que interactuamos con nuestros hogares y otros entornos, y cómo estos conceptos se han expandido a nivel industrial y empresarial.

Posteriormente, se presenta una revisión del microcontrolador ESP32, que es el componente principal utilizado en los proyectos. Se describen las características principales del ESP32, como su velocidad de procesamiento, capacidad de conectividad, y opciones de programación, así como alguna de las diferentes tarjetas de desarrollo que lo integran, y su compatibilidad con el entorno de desarrollo integrado de Arduino (IDE de Arduino).

El capítulo continúa con una sección sobre los sensores y transductores utilizados en los proyectos. Se detallan los diferentes tipos de sensores, tales como los de luz, proximidad, ultrasonido, RFID, temperatura, humedad, presión atmosférica, altitud, radiación UV, y anemómetro. Seguidamente, se describen los actuadores utilizados en el proyecto, incluyendo los indicadores visuales, interruptores, bombas de agua, cerraduras tipo solenoide y electroimán, y etapas de potencia para control de corriente alterna y directa. Se explican los principios de funcionamiento de cada uno de estos componentes, así como su importancia para el monitoreo y control a través de la red.

La sección final del capítulo está dedicada a los servicios web orientados al IoT. Se presenta un esquema general para la implementación de proyectos IoT, y se describen las diferentes plataformas y servicios web disponibles para la implementación de proyectos IoT, incluyendo la plataforma Blynk 2.0.

El capítulo 3 detalla el desarrollo experimental de cada proyecto, así como las pruebas y resultados obtenidos. En primer lugar, se describe el proceso de habilitación de una tarjeta ESP32 mediante el IDE de Arduino en la plataforma Blynk. Esto incluye el registro en la plataforma, la configuración web y móvil, la creación de plantillas, la configuración de la biblioteca Blynk y la creación de canales para el flujo de datos. Además, se presenta un ejemplo de código para el funcionamiento básico, así como una interfaz web y móvil para la plataforma Blynk. También se explica el proceso de alambrado del circuito y la asociación de una tarjeta de desarrollo a la plataforma.

Así también, se describen soluciones desarrolladas en este capítulo que incluyen el control de luminarias, el control de potencia AC, control y monitoreo de nivel de agua, iluminación RGB, riego de sustrato, control de accesos y una estación meteorológica, cada una de ellas con su correspondiente diagrama de bloques, diagrama esquemático, diagrama de flujo, interfaces web y móvil desarrolladas para la plataforma Blynk, así como las pruebas y resultados obtenidos para cada una de ellas. Cabe resaltar que el control de iluminación con notificaciones de presencia y asistentes virtuales, se realizó mediante la integración del proyecto en tres plataformas: Blynk, Sinric Pro y ESP RainMaker, esto con el fin cumplir con el objetivo de integración de asistentes virtuales, ya que la plataforma Blynk perdió esta capacidad de asociación de manera gratuita, estos proyectos como los anteriores incluyen su diagrama de bloques, diagrama esquemático, y de flujo para la implementación de estas soluciones, así mismo se muestran las interfaces web y móvil (si están disponibles), así como la asociación a el asistente de Google y Amazon Alexa para cada plataforma, junto con las pruebas y resultados obtenidos.

CAPÍTULO 2. MARCO TEÓRICO

2.1 Conceptos de domótica e Internet de las Cosas

2.1.1 Domótica

La domótica consiste en la automatización de procesos de una vivienda, en las áreas de la seguridad, comunicación, gestión energética, accesibilidad y con ello otorgar comodidad y confort, todo esto a través de un conjunto de tecnologías aplicadas al control y la automatización inteligente (BANCO BILBAO VIZCAYA ARGENTARIA, S.A., 2021).

Ámbitos de aplicación de la domótica:

- Programación y consumo energético: ayuda a hacer una gestión eficiente del gasto del hogar, manejándose de forma inteligente los sistemas de climatización, toldos y persianas, además de contribuir a una mejor gestión eléctrica
- Seguridad: se trata de la configuración de una red de seguridad compuesta por alarmas, detectores de incendios, teleasistencia, cámaras...
- Comunicaciones: unifica los sistemas de comunicación del hogar, pudiendo controlarse de forma remota los sistemas de teleasistencia, mantenimiento, telefonía e internet, así como videoporteros, la trasmisión de alarmas, entre otros
- Accesibilidad: contempla las aplicaciones o instalaciones que favorecen la autonomía de las personas con algún tipo de limitación o discapacidad, por lo que la domótica es una herramienta para conseguir la inclusión social y la igualdad
- Confort. La domótica permite la automatización de la iluminación (encendido y apagado, regulación, etc.), el uso de cerraduras inteligentes, el control desde cualquier dispositivo conectado a Internet, la gestión del ocio... (Domínguez, 2022)

Al estar orientada hacia productos del hogar, como electrodomésticos, sistemas de climatización, iluminación, entre otros, se busca una interfaz más sencilla, con sensores muy poco invasivos y que no supongan una intrusión brusca en el ambiente del hogar. Teniendo como objetivo, generar el máximo confort en la casa facilitando la comunicación entre el sistema y las personas que la habitan (BANCO BILBAO VIZCAYA ARGENTARIA, S.A., 2021).

2.1.2 Internet de las Cosas

En el documento Internet de las Cosas redactado por Jordi Salazar y Santiago Silvestre definen a esta como “La arquitectura basada en internet que permite el intercambio de bienes y servicios entre los elementos, equipos y objetos conectados a la red…, se refiere a la interconexión en red de todos los objetos cotidianos…, que están equipados con sensores, software y entre otras tecnologías que los dota de algún tipo de inteligencia... en ese contexto, Internet puede ser también una plataforma que permite la comunicación, intercambio de información y datos específicos de los dispositivos con el mundo a su alrededor…, añadiendo así una interconectividad más extensa” (Salazar & Silvestre, 2006).

En conjunto el Internet de las Cosas y la domótica introducen un cambio radical en la calidad de vida de las personas, ofreciendo una gran cantidad de nuevas oportunidades de acceso a datos,

y nuevos servicios que pueden ser personalizados según las necesidades del usuario. El Internet de las Cosas trae como beneficios la mejora de la gestión, aumenta la cantidad de datos de información y permite la optimización de equipos y uso de los recursos que puede traducirse en ahorro de costes. Además, ofrece la oportunidad de crear nuevos dispositivos interconectados inteligentes (Salazar & Silvestre, 2006).

Por tanto, se puede considerar al Internet de las Cosas como una evolución en las comunicaciones del concepto de domótica, en el que pasamos de la programación a la recopilación y lectura de datos para la automatización.

2.2 Características del microcontrolador ESP32 y principales tarjetas de desarrollo que lo integran

Para lograr la interconexión de elementos y objetos dentro de una red en una vivienda se requiere de habilitar los mismos con apoyo de hardware especifico como por ejemplo las tarjetas de desarrollo que integran el microcontrolador ESP32.

2.2.1 Microcontrolador ESP32

El microcontrolador ESP32 es una serie de chips que integran todo un sistema de microcomputadora, de bajo coste y muy bajo consumo energético, creados por la empresa Espressif Systems. Esta es serie sucesora del microcontrolador ESP8266, desarrollado para aplicaciones del IoT, brinda más y mejores prestaciones con respecto a su antecesor, como principales características y mejoras a este ofrece:

- Una CPU Tensilica Xtensa LX6 a 32-bit y RISC (con conjunto de instrucciones reducido), de doble núcleo con una frecuencia de operación de 160MHz, alcanzando un máximo de 240MHz. También incluye seguridad criptográfica por hardware y un coprocesador para el modo bajo consumo (Deep sleep)
- 448KiB[1] de ROM, 520KiB de SRAM para instrucciones y datos respectivamente, 4MiB[2] Flash con soporte externo para Flash de hasta 16MiB y hasta 8MiB de memoria SRAM externa
- Wifi bajo norma 802.11 b/g/n/e/i (802.11n hasta 150Mbps)
- Bluetooth V4.2 a 2.4Ghz y Bluetooth Low Energy (BLE) V4.0 (comparte antena de radio con Wifi)
- 36 pines GPIO (no todos los pines están disponibles para uso)[3]
- 16 canales de ADC de 12 bits de resolución (2 ADC)
- 10 pines con sensor táctil (capacitivos)
- 2 x DAC de 8 bits
- 16 canales de PWM (por emulación con software)
- 2 x UART para comunicación serie

[1] 1 Kibibyte (KiB) ≈ 1.024 Kilobyte (kB) (2021)
[2] 1 Mebibyte (MiB) ≈ 1.049 Megabyte (MB) (2021)
[3] No todas las características son aprovechadas, ya que depende de su disponibilidad respecto a la tarjeta de desarrollo y fabricante.

- 2 canales I2C y 4 canales SPI
- Alimentación a 3.3V

Todas estas características se encuentran integradas en un módulo de circuito impreso ESP32-WROOM-32 (figura 2.1) igualmente creado por la empresa Espressif Systems, de dimensiones contenidas que como base porta el chip ESP32-D0WDQ6 con 4MiB de memoria flash. Este módulo cuenta con la electrónica necesaria para funcionar, además de blindaje que recubre los componentes y una antena impresa tipo MIFA para las interfaces de radio (Carmenate, programarfacil.com, 2021).

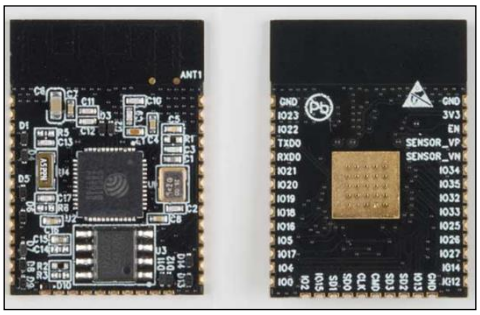


Figura 2.1. Módulo PCB ESP32-WROOM-32 distribución de GPIO, electrónica sin blindaje.

2.2.2 Ejemplos de tarjetas de desarrollo con el microcontrolador ESP32

Existen varias placas para el desarrollo de prototipos entre las que destacan:

ESP32 DEVKIT

Existen dos versiones de esta placa: una con 30 y otra con 38 pines (figura 2.2). Ambas se basan en el módulo PCB ESP32-WROOM-32 y son similares, siendo la distribución de pines su única diferencia (Carmenate, programarfacil.com, 2021).

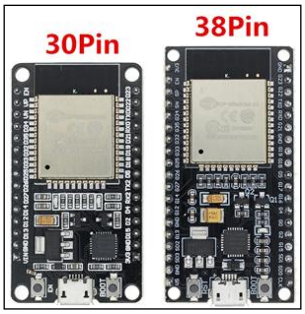


Figura 2.2. Tarjeta de desarrollo ESP32 DEVKIT genérica.

ESP32 con pantalla OLED

Esta placa de desarrollo igualmente basada en el módulo ESP-WROOM-32, posee una pantalla OLED de 0.96 pulgadas.

Contar con una pantalla integrada es una ventaja, ya que puede ser utilizada para depuración o incluso como interfaz visual de proyectos (Carmenate, programarfacil.com, 2021).

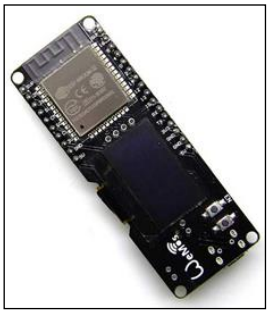


Figura 2.3. Tarjeta de desarrollo Wemos Lolin32 ESP32 OLED.

ESP32-CAM

Es una placa de desarrollo basada en el módulo ESP-WROOM-32 que cuenta con cámara y slot para tarjetas microSD (Carmenate, programarfacil.com, 2021).

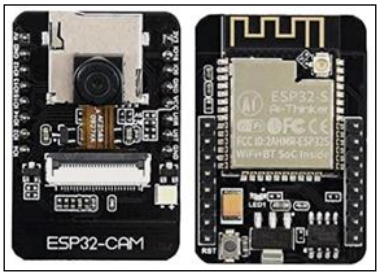


Figura 2.4. Tarjeta de desarrollo ESP32-CAM genérica.

2.2.3 Tarjeta NodeMCU-32S

La tarjeta NodeMCU-32S permite monitorear y/o controlar múltiples sensores, módulos y actuadores mediante los pines externos para la conexión de sus múltiples GPIO, así como a través de conexión Wifi y/o Bluetooth, para proyectos del IoT. Cuenta con las características generales del PCB ESP-WROOM-32, además de la electrónica adicional que facilita la creación de prototipos: puerto Micro-USB tipo B para alimentación y programación, integra un controlador CP2102 de USB a UART, así como regulador de voltaje de 5V a 3.3V (UNIT Electronics, 2020).

Los pines están etiquetados en la parte inferior de la placa, lo que facilita la conexión con otros dispositivos. Esta versión de 38 pines permite el acceso a los GPIO 6, 7, 8, 9, 10 y 11, pines utilizados para la memoria externa (se debe tomar en cuenta que el fabricante recomienda no utilizar estos pines) así como los pines correspondientes a GPIO 1 (TX0) y GPIO 3 (RX0) ya que podrían generar problemas de comunicación con el equipo de cómputo durante la carga de programa o visualización de información por medio del monitor serie, los pines correspondientes a las GPIO 34, 35, 36 y 39 solo pueden ser utilizados como entradas ya que no cuentan con resistor pull-up de forma interna (uPesy, 2022).

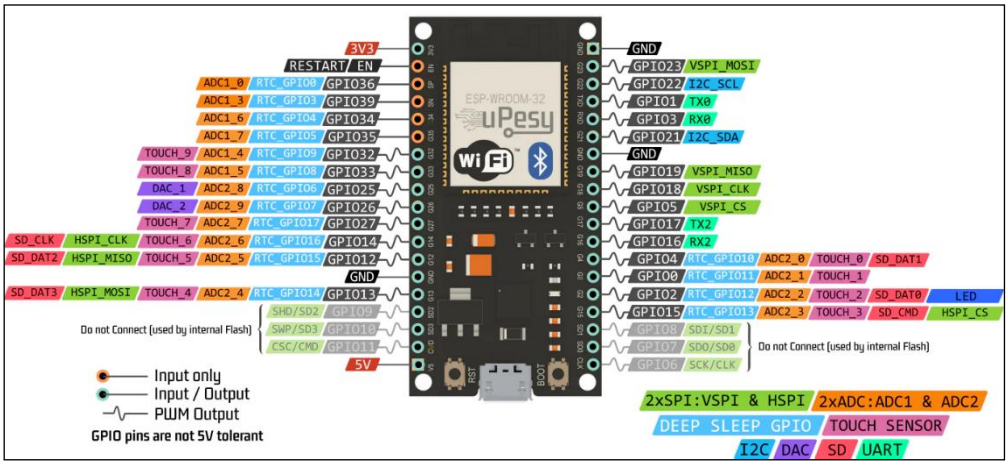


Figura 2.5. Distribución de pines en la tarjeta de desarrollo NodeMCU-32S.

2.2.4 Integrar las tarjetas basadas en el microcontrolador ESP32 a el entorno de desarrollo integrado de Arduino

Las tarjetas de desarrollo con ESP32 pueden ser programadas con una gran variedad de software, lenguajes de programación o framework, entre estos los que más destacan son: ESP-IDF (Espressif IoT Development Framework) desarrollado por el fabricante, IDE Arduino (en lenguaje C++), Python o MicroPython, por mencionar algunos (UNIT Electronics, 2020).

La solución más práctica en cuanto a la programación es trabajar con el Entorno de Desarrollo Integrado de Arduino (IDE Arduino), preparar dicho entorno para poder programar tarjetas que integren el microcontrolador ESP32 resulta sencillo gracias a que el fabricante desarrolla en código abierto un complemento para este entorno y permite que, una vez instalado, muestre a el ESP32 como un Arduino más, este núcleo para el entorno de desarrollo brinda soporte para las placas ESP32, ESP32-S2, ESP32-S3 y ESP32-C3, manteniendo la máxima compatibilidad posible en la estructura de instrucciones y la terminología de la programación en Arduino, facilitando el que se pueda empezar a desarrollar de forma inmediata con el microcontrolador ESP32 (Prometec www.prometec.net, 2020).

Para obtener el IDE Arduino, se realiza la descarga desde el sitio web oficial[1] en la pestaña de software se mostrará la versión más reciente del entorno, un poco más abajo en la página podemos encontrar también las versiones anteriores, versiones en desarrollo, así como sus variantes por plataforma: Windows, Linux y MacOS. Para este trabajo todas las instrucciones de instalación y configuración de los respectivos programas y aplicaciones se han realizado con las versiones más recientes disponibles a la fecha.

Al realizar la instalación, es importante permitir la instalación de los respectivos controladores ya que habilitan la comunicación entre las placas de desarrollo y el equipo de cómputo.

[1] https://www.arduino.cc/

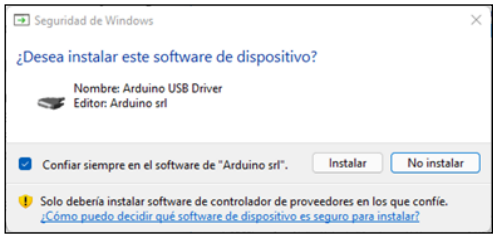


Figura 2.6. Confirmación de controladores para el sistema operativo Windows.

Una vez terminada la instalación iniciar el entorno de programación para configurar el complemento específico para trabajar con tarjetas basadas en el microcontrolador ESP32. En el menú Herramientas ► Placa, seleccionar el gestor de tarjetas que permite incluir microcontroladores y tarjetas de desarrollo no oficiales para el entorno de forma rápida.

Dentro de las preferencias, en el apartado: Gestor de URL Adicionales de Tarjetas, se escribe dentro del campo la url[1] para las compilaciones estables y más recientes de los núcleos de las tarjetas con arquitecturas Xtensa (ESP32 serie S) y RISC (ESP32 serie C), en caso de que ya existan otra u otras direcciones cargadas solo es necesario sepáralas por una coma y así el programa leerá todas aquellas direcciones que tenga en el campo, pulsar el botón “OK” para confirmar los cambios, ver figura 2.7.

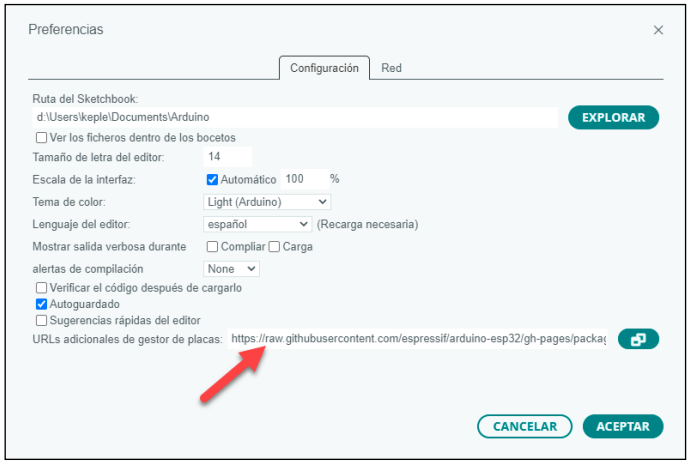


Figura 2.7. Ventana de preferencias en el IDE Arduino.

En el Gestor de tarjetas (Herramientas ► Placa: ► Gestor de tarjetas, o a través del botón a la izquierda de la interfaz, ver figura 2.8), buscar el complemento para tarjetas basadas en ESP32 de Espressif Systems, por defecto se instala la versión más reciente, una vez termine de descargar todos los datos del paquete, ver figura 2.8, al finalizar se deberá reiniciar el entorno de programación.

[1] https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

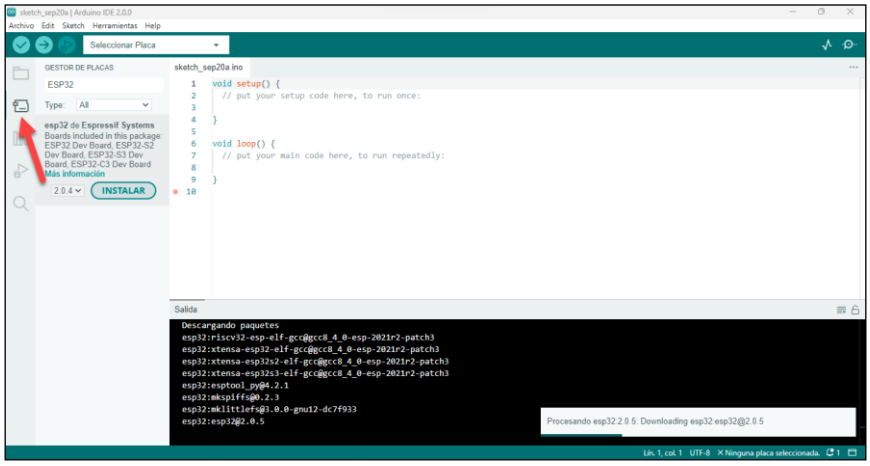


Figura 2.8. Instalación de complemento a través del gestor de tarjetas.

En el menú Herramientas ► Placa: ► ESP32 Arduino, ver figura 2.9, seleccionar el modelo correspondiente al tipo de placa y con esto el entorno queda listo para desarrollar proyectos en las tarjetas de desarrollo con microcontrolador ESP32.

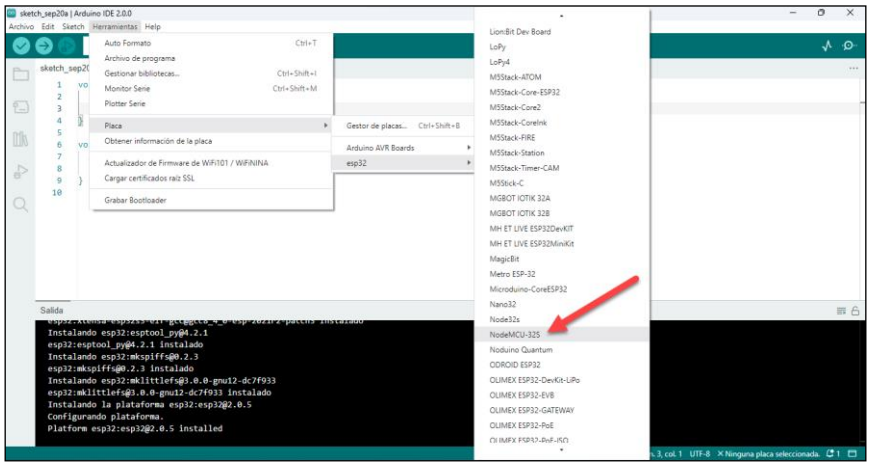


Figura 2.9. Selección de tarjeta ESP32 en Arduino IDE.

Observación, en este documento, para la tarjeta NodeMCU-32S seleccionada, si ocurre problemas de reconocimiento entre la tarjeta ESP32 y el equipo de cómputo se puede obtener los controladores más recientes del sitio web[1], en la pestaña “DOWNLOADS”, para el chip de programación CP21xx que integran la mayoría de las tarjetas de desarrollo genéricas.

2.2.5 Compatibilidad de tarjetas ESP32, bibliotecas para sensores y actuadores soportados por el IDE de Arduino

Para el presente trabajo de investigación se utiliza la tarjeta de desarrollo NodeMCU-32S V1.1, algunos de los casos de incompatibilidad e integración que aquí se detallan solo aplican a esta

[1] https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers

versión de tarjeta de desarrollo. Esta tarjeta cuenta con la electrónica necesaria para hacer funcionar al chip ESP32 que integra, así incluye un chip regulador de voltaje que se encarga de suministrar al ESP32 los 3.3V necesarios para su operación.

Las tarjetas basadas en ESP32 funcionan con niveles lógicos a 3.3V, si se aplicar un voltaje superior a uno de los GPIO con los que cuenta esta se puede dañar. El chip principal y parte del circuito de esta placa se encuentran protegidos por una cubierta metálica (blindaje) para reducir las interferencias.

Además, en la tarjeta se aprecian los siguientes componentes:

- Puerto micro USB. Listo para conectar, programar o alimentar la tarjeta igual que un Arduino UNO. Tener en cuenta que se trata un conector diferente USB en este caso Micro USB tipo B
- Push button RST. Permite reiniciar el ESP32. Equivalente al botón de reinicio de las tarjetas Arduino
- Push button BOOT. Utilizado para programar la placa. Este se encuentra conectado al GPIO 0 y puede ser utilizado como un pulsador normal en la programación.
- Antena tipo MIFA para Wifi y Bluetooth
- LED de alimentación. Este LED se enciende al alimentar correctamente la placa
- LED integrado. Conectado al GPIO2, y puede ser controlado desde la programación

Así también se puede apreciar el chip CP2102, que actúa como puente entre la computadora y el ESP32, es decir habilita la comunicación entre ellos utilizando el puerto micro USB de la tarjeta, permite programar el ESP32 sin necesidad de programadores u otras herramientas de hardware externas (Carmenate, programarfacil.com, 2021).

Esta tarjeta no realiza el volcado del programa de forma automática (modo flashing/uploading) por lo que es necesario utilizar el push button “BOOT” cuando se programa el ESP32 desde el IDE de Arduino, basta con pulsar y mantener unos segundos hasta que el IDE empiece a cargar el programa, cuando se intenta cargar un sketch sin realizar esta acción, el IDE enviara un mensaje de error. De manera general en la mayoría de las tarjetas de Arduino esta función es automática, esto no es así en todas las tarjetas que incluyen un ESP32, como solución, para la programación de forma automática se puede conectar un capacitor electrolítico entre los pines EN y GND, al colocar recordar que este tipo de capacitor tiene polaridad (RANDOM NERD TUTORIALS, 2019).

Los pines EN, 34, 35, 36 y 39 son pines de solo entrada como ya se mencionó, por tanto si se intenta utilizar la función “digitalWrite()” con uno de estos pines no se obtendrá el resultado esperado. Es recomendable reservar estos pines para dispositivos de entrada tales como: pulsadores, teclados, sensores analógicos, etcétera (Carmenate, programarfacil.com, 2021).

Para alimentar la tarjeta de desarrollo se puede hacer por medio del conector micro USB o directamente en los pines 5V o 3.3V, con su voltaje respectivo. Cuando la tarjeta es alimentada por el puerto Micro USB los pines 5V y 3.3V pueden ser utilizados para alimentar otros componentes que requieran una alimentación de 5 o 3.3V. Siempre teniendo en cuenta que un consumo excesivo puede dañar el puerto de la fuente de alimentación (atención al alimentar con puertos USB de equipos de cómputo). Teniendo en cuenta que los 3.3V son obtenidos a partir

del regulador integrado en la tarjeta, es necesario que el consumo no exceda los 500mA. De lo contrario, el regulador y/o la tarjeta pueden resultar dañados (Carmenate, programarfacil.com, 2021).

Es recomendable no alimentar la placa desde el puerto USB y el pin 5V al mismo tiempo. En este caso también es posible utilizar el pin 3.3V para alimentar otros dispositivos externos, pero siempre teniendo en cuenta no exceder el consumo sobre los 500mA (Carmenate, programarfacil.com, 2021).

Si no es necesario disponer de 5V en el proyecto, es posible utilizar una fuente de alimentación de 3.3V y conectarla directamente al pin correspondiente. De esta forma, el pin 5V no puede ser utilizado para alimentar dispositivos que requieran dicho voltaje. Además, es necesario garantizar que la fuente sea estable o de lo contrario el ESP32 podría resultar dañado (Carmenate, programarfacil.com, 2021).

Instalación de bibliotecas por medio del gestor y en formato ZIP

La instalación de bibliotecas al entorno de IDE Arduino, para la inclusión de estas en los códigos de los programas se puede realizar de tres formas distintas, para el sistema operativo de Microsoft Windows.

La primera consiste en realizar la búsqueda en los repositorios dentro del entorno, para esto en el entorno pulsar sobre el menú Sketch ► Incluir biblioteca ► Gestionar bibliotecas…, o bien por medio del botón Gestor de Bibliotecas a la izquierda de la interfaz ver figura 2.10.

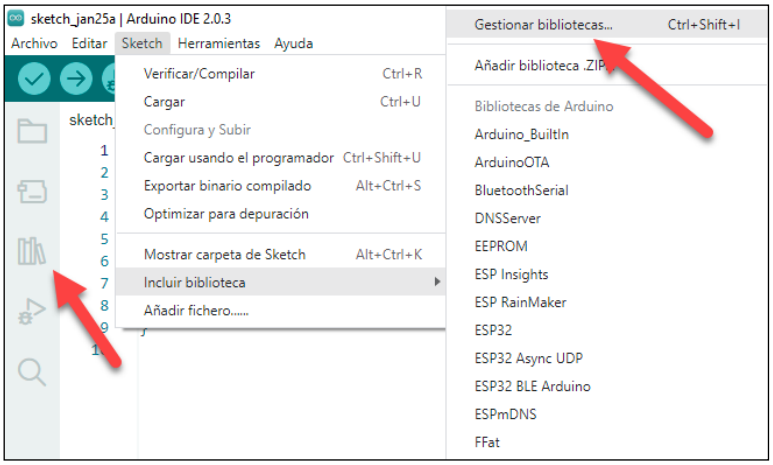


Figura 2.10. Gestor de bibliotecas, IDE Arduino

En el gestor se escribe en el campo de búsqueda (si es necesario esta se puede filtrar por tipo de biblioteca o tema, también es posible ver el nombre del desarrollador para verificar el origen) el nombre de la biblioteca que se va a instalar, ver figura 2.11, y pulsar sobre el botón instalar, esperar el mensaje de salida que confirme la instalación y así la biblioteca estará lista para ser implementada en los proyectos. Si es necesario se puede reiniciar el entorno de Arduino para evitar errores posteriores a la instalación.

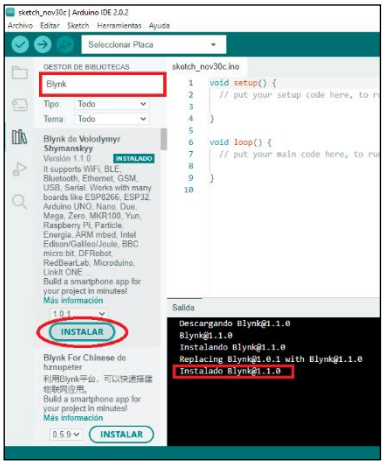


Figura 2.11. Instalación de biblioteca para plataforma Blynk.

De forma manual, se debe obtener el archivo ZIP de los repositorios de los desarrolladores, para este ejemplo se descarga la biblioteca para la plataforma de Blynk[1] del GitHub[2] oficial, en la página pulsar sobre el botón “Code” y después sobre “Download ZIP”.

Una vez descargado el archivo ZIP para cargarlo al entorno pulsar en el menú Sketch ► Incluir biblioteca ► Anadir biblioteca .ZIP…, ver figura 2.12. Se abrirá una ventana que permite seleccionar el archivo.

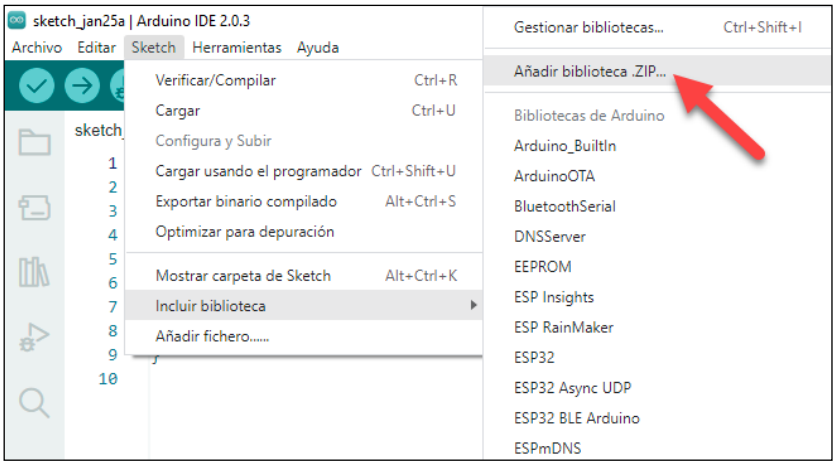


Figura 2.12. Instalación manual de biblioteca en IDE Arduino.

Esperar al procesamiento de la biblioteca y el mensaje de confirmación, con esto la biblioteca ha quedado instalada en el entorno.

Por último, otra forma de instalar las bibliotecas consiste en extraer el contenido del archivo ZIP con cualquier compresor de ficheros, es importante mantener la estructura del archivo ZIP (nombres de archivo y orden de capetas tal y como se encuentran dentro del archivo comprimido

[1] https://github.com/blynkkk/blynk-library/archive/refs/heads/master.zip
[2] https://github.com/blynkkk/blynk-library

ZIP). Extraído el contenido copiar a la carpeta en la ruta "X:\Users\<Nombre del usuario>\Documents\Arduino\libraries", donde "X" es la unida de almacenamiento donde se instaló el IDE Arduino y el nombre de usuario tal como se indica en el equipo de cómputo sin símbolos "< >". Esta instalación se realiza con el entorno de Arduino cerrado, y este se ejecutará una vez se hayan copiado las dependencias de la biblioteca.

Compatibilidad de bibliotecas con el entorno de Arduino y las tarjetas de desarrollo con ESP32

Las bibliotecas para la conexión Bluetooth y Wifi, se incluyen al instalar el núcleo de las tarjetas ESP32, las cabeceras para estas bibliotecas son "BluetoothSerial.h" y "Wifi.h", donde la primera engloba una serie de nuevos comandos que no son compatibles con las tarjetas Arduino ya que no solo funcionan con la conexión clásica de bluetooth, sino que también aprovechan el modo de bajo consumo (BLE) que integra el ESP32. En cuanto a funcionalidad la biblioteca Wifi trabaja en mayor parte con las mismas instrucciones que se utilizan para los programas de las tarjetas Arduino, en cada llamada del compilador a la cabecera Wifi.h, esta redirige a la cabecera esp_wifi.h con la misma serie de instrucciones de la primera, esta última biblioteca cuenta con alunas funcionalidades adicionales que no pueden ser aprovechadas por el IDE de Arduino y que solo están disponibles en el compilador Espressif IoT Development Framework (ESP-IDF) ((Alias) l. , 2022).

Al escribir programación para modulación por ancho de pulso (PWM del inglés Pulse Width Modulation) con el IDE Arduino en las tarjetas con ESP32, se puede generar señales de 8, 10 y hasta 12 bits de resolución a través de 16 canales virtuales, con salida por medio de los pines disponibles para PWM (circuits4you.com, 2018), ver figura 2.5. Para la generación de estas señales en las tarjetas con ESP32 es necesario especificar el canal, la frecuencia, la resolución con la función "ledcSetup()", así como la función "ledcAttachPin()" que acepta dos argumentos. El primero es el GPIO que emitirá la señal, y el segundo es el canal que generará la señal. Para escribir el valor en el GPIO de salida la función "analogWrite()" es remplazada por la función "ledcWrite()", esta función acepta como argumentos el canal que está generando la señal PWM y el ciclo de trabajo (RANDOM NERD TUTORIALS, 2018).

Otro cambio de programación considerable radica en el uso de la memoria no volátil (NVS) que dispone el microcontrolador ESP32.

En las tarjetas Arduino el microcontrolador dispone de una memoria para almacenar datos cuando la misma esta apagada, esta es también una memoria NVS llamada EEPROM, para hacer uso de esta memoria, se dispone de la biblioteca "EEPROM.h" que cuenta con funciones para leer y escribir datos sobre esta memoria (Giraldo, 2019). En las tarjetas con ESP32 se utiliza aproximadamente 4MB de memoria flash como almacenamiento NVS, la biblioteca EEPROM.h es compatible con la ESP32, no obstante, para el uso eficiente de esta memoria la documentación oficial recomienda el uso de la biblioteca "Preferences.h", incluida en la instalación del núcleo para tarjetas ESP32, que al igual que la biblioteca "EEPROM.h" permite guardar y recuperar datos almacenados en esta memoria flash (Espressif, 2022). A continuación, se presentan algunas de las funciones para guardar y obtener datos almacenados en el flash de las tarjetas con ESP32.

- Los datos guardados utilizando la biblioteca "Preferences.h" están estructurados como:

```
nombredelespacio {
palabra: valor
}
```

- Se pueden guardar diferentes palabras en el mismo espacio, por ejemplo:

```
nombredelespacio {
palabra1: valor1
palabra2: valor2
}
```

Como ejemplo práctico, esta configuración podría usarse para guardar credenciales de red:

```
conexionwifi {
ssid: "tu_ssid"
pass: "tu_contraseña"
}
```

- Se puede tener múltiples espacios con la misma palabra (pero cada palabra con su valor):

```
nombredelespacio1{
palabra: valor1
}
nombredelespacio2{
palabra: valor2
}
```

Cuando se utiliza la biblioteca se debe definir el tipo de datos que se va a almacenar, así mismo para recuperar los datos, se debe conocer el tipo de dato que se almaceno. En otras palabras, el tipo de datos de escritura y lectura debería ser el mismo. Se pueden guardar los siguientes tipos de datos: Char, Uchar, Short, Ushort, Int, Uint, Long, Ulong, Long64, Ulong64, Float, Double, Bool, String y Bytes.

Para almacenar datos, primero se debe incluir la biblioteca en el sketch:

```
#include <Preferences.h>
```

Definir un espacio de la biblioteca:

```
Preferences ejemplo;
```

Los siguientes métodos sirven para manejar datos usando la biblioteca.

Iniciar y abrir el "espacio de almacenamiento" con la función begin() con el nombre definido. El argumento “false” habilita el modo de lectura/escritura. Usar el argumento “true” Para el modo de solo lectura.

```
ejemplo.begin("mi-app", false);
```

El nombre del nombre del espacio es mi-app, y está limitado a 15 caracteres.

Usar clear() para borrar todas las palabras en el espacio abierto (no elimina el espacio):

```
ejemplo.clear();
```

Eliminar solo una palabra del espacio abierto:

```
1       ejemplo.remove(palabra);
```

Usar end() para cerrar el espacio abierto:

```
1       ejemplo.end();
```

Para almacenar (tabla 2.1) u obtener (tabla 2.2) datos se usan diferentes funciones que dependerán del tipo de información a gestionar (RAMDOM NERD TUTORIALS, 2021).

Tabla 2.1 Tipos de datos y funciones para guardar información.

Tipo de dato	Función
Char	putChar(const char* key, int8_t value)
Unsigned Char	putUChar(const char* key, int8_t value)
Short	putShort(const char* key, int16_t value)
Unsigned Short	putUShort(const char* key, uint16_t value)
Int	putInt(const char* key, int32_t value)
Unsigned Int	putUInt(const char* key, uint32_t value)
Long	putLong(const char* key, int32_t value)
Unsigned Long	putULong(const char* key, uint32_t value)
Long64	putLong64(const char* key, int64_t value)
Unsigned Long64	putULong64(const char* key, uint64_t value)
Float	putFloat(const char* key, const float_t value)
Double	putDouble(const char* key, const double_t value)
Bool	putBool(const char* key, const bool value)
String	putString(const char* key, const String value)
Bytes	putBytes(const char* key, const void* value, size_t len)

Tabla 2.2 Tipos de datos y funciones para guardar información.

Tipo de dato	Función
Char	getChar(const char* key, const int8_t defaultValue)

Unsigned Char	getUChar(const char* key, const uint8_t defaultValue)
Short	getShort(const char* key, const int16_t defaultValue
Unsigned Short	getUShort(const char* key, const uint16_t defaultValue)
Int	getInt(const char* key, const int32_t defaultValue)
Unsigned Int	getUInt(const char* key, const uint32_t defaultValue)
Long	getLong(const char* key, const int32_t defaultValue)
Unsigned Long	getULong(const char* key, const uint32_t defaultValue)
Long64	getLong64(const char* key, const int64_t defaultValue)
Unsigned Long64	gettULong64(const char* key, const uint64_t defaultValue)
Float	getFloat(const char* key, const float_t defaultValue)
Double	getDouble(const char* key, const double_t defaultValue)
Bool	getBool(const char* key, const bool defaultValue)
String	getString(const char* key, const String defaultValue)
String	getString(const char* key, char* value, const size_t maxLen)
Bytes	getBytes(const char* key, void * buf, size_t maxLen)

En general estos son los cambios de programación más significativos que se abordan en el presente trabajo de investigación; si es necesario, para aclaraciones adicionales se puede consultar los siguientes recursos: Foro general del uso de ESP32 usado el entorno de Arduino[1], compatibilidad de bibliotecas por versión de ESP32[2], y el foro oficial de Espressif para reporte y solución de problemas[3].

2.3 Sensores y transductores

2.3.1 Sensores de luz

Un sensor de luz es un dispositivo que transforma la energía de los fotones en señales eléctricas y permite conocer el grado de iluminación en un entorno concreto. Suelen estar conectados a otros dispositivos, que reciben esta información respecto a la luminosidad ambiental detectada y a partir de esta realizar ciertas acciones (simon, 2021).

Light dependent resistor (LDR)

El sensor LDR es un fotorresistor, es decir que el valor de su resistencia eléctrica varía en función de la luz que recibe. Es un sensor analógico, por tanto, su conexión deberá ser a una entrada analógica de algún microcontrolador (Manuel Hidalgo Díaz, 2016). No requiere el uso de bibliotecas, este fotorresistor también puede ser presentado en un módulo que integra regulador de voltaje y un comparador que junto a un potenciómetro genera una salida digital con solo dos estados HIGH/LOW y puede ser alimentado con 3.3V o 5V.

Figura 2.13. Fotorresistor (LDR).

Algunas de sus características generales son:

- Respuesta espectral: tipo 5mm ≈ 540nm, tipo 12 y 20mm ≈ 560nm
- Temperatura de trabajo: -30° a +70°C
- El voltaje máximo soportado en corriente directa es de 150 a 500V, (depende del modelo) (NTE Electronics, 2020)

Módulo BH1750

El módulo BH1750 a diferencia de un LDR posee un conversor interno de 16-bit, por lo que los datos a su salida son digitales y puede entregar directamente valores con una alta precisión de medición en un rango de 1 a 65535lx (Lux = lumen/m^2) unidad de medida estándar para niveles

[1] https://github.com/espressif/arduino-esp32
[2] https://docs.espressif.com/projects/arduino-esp32/en/latest/libraries.html
[3] https://github.com/espressif/arduino-esp32/issues

de iluminación. No es necesario realizar conversiones de voltaje para obtener datos legibles (© 2014 ROHM Co., 2013).

El módulo cuanta con regulador integrado y puede ser alimentado con 3.3 o 5V, la interfaz de comunicación es I2C y cuenta con un pin dedicado para la selección de dirección de comunicación (ADDR) que puede ser HIGH = 0x5C, LOW = 0x23, cuenta con resistor a GND ya que, si no se conecta la dirección, por defecto será 0x23 (©2021 Naylamp Mechatronics SAC).

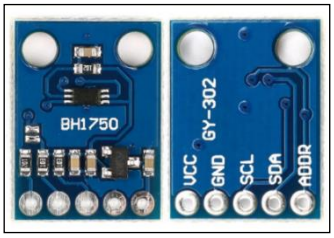


Figura 2.14. Módulo BH1750.

Para su uso se requiere la instalación de la biblioteca BH1750 escrita por Christopher Laws que se puede instalar desde el Gestor de Bibliotecas u obtener del sitio web[1] como paquete ZIP (©2021 Naylamp Mechatronics SAC) e integrar como se describe en la sección 2.2.5.

Funciones de la biblioteca:

- begin(mode): inicializa el módulo y la comunicación, así como configura la resolución del BH1750 con la configuración establecida con ***mode***.
- configure(mode): configura el módulo con la diferencia que esta función no inicializa la comunicación I2C, en ambos casos ***mode*** puede ser:
    - BH1750_CONTINUOUS_HIGH_RES_MODE
    - BH1750_CONTINUOUS_HIGH_RES_MODE_2
    - BH1750_CONTINUOUS_LOW_RES_MODE
    - BH1750_ONE_TIME_HIGH_RES_MODE
    - BH1750_ONE_TIME_HIGH_RES_MODE_2
    - BH1750_ONE_TIME_LOW_RES_MODE

  Si no se especifica el ***mode***, por defecto es:

    - BH1750_CONTINUOUS_HIGH_RES_MODE
- readLightLevel(), realiza la lectura del nivel de iluminación al que está expuesto el sensor, retorna un entero sin signo con el valor de la medición de luz en Lux (©2021 Naylamp Mechatronics SAC).

Módulo TEMP6000

[1] https://github.com/claws/BH1750

Construido a base de un sensor analógico y por lo tanto no requiere el uso de bibliotecas, opera con un fototransistor donde entre mayor es la luz que le incide mayor será la corriente en el pin de salida (VISHAY, 2004).

Características:

- Sensibilidad: 570nm
- Voltaje de saturación C-E: 0.1V
- Temperatura de operación: -40° a 85°C
- Alimentación a 3.3 o 5V
- Pines: S para salida analógica, G para GND, V para Vcc (Didácticas Electrónicas, 2016)

Figura 2.15. Módulo TEMP6000.

Módulos TSL2560, TSL2561 y TSL2591

Grupo de módulos que se integran por fotodiodos y convertidor analógico-digital, entregando un alto rango dinámico y mayor precisión para el cálculo de Lux, el módulo posee dos fotodiodos, uno dedicado al espectro infrarrojo y otro al de luz visible, esto permite hacer mediciones independientes del espectro de luz (Cetronic, S.L., 2018).

Características:

- Temperatura de operación: -30° a 80°C
- Voltaje de operación: 3.3 o 5V
- Interfaz de comunicaciones: I2C
- Rango de medición: TSL2560 y TSL2561 de 0.1 a 40000lx (Herramientas Tecnológicas Profesionales, S. de R.L. de C.V., 2018), TSL2591 de 188 a 88000lx (Cetronic, S.L., 2018)

Para su uso requiere de la instalación de bibliotecas en el IDE Arduino las cuales se pueden obtener del sitio web[1] para los sensores TSL2560 o TSL2561 (Herramientas Tecnológicas Profesionales, S. de R.L. de C.V., 2018) y del sitio[2] para TSL2591 (Cetronic, S.L., 2018), así como también a través del Gestor de Bibliotecas, ambas escritas por el fabricante Adafruit.

[1] https://github.com/adafruit/Adafruit_TSL2561
[2] https://github.com/adafruit/Adafruit_TSL2591_Library

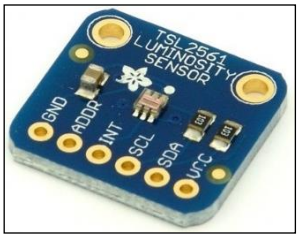


Figura 2.16. Módulo TSL2561.

2.3.2 Proximidad

El sensor de proximidad es un transductor que detecta la presencia de objetos u obstáculos sin la necesidad de contacto, existen diferentes tipos de sensores de proximidad las características de detección dependen básicamente de la tecnología del sensor y del medio de transmisión (Mecatrónica LATAM, 2021).

Módulo APDS-9960

El módulo APDS-9960 integra un sensor para uso de gestos, proximidad, colorímetro y de iluminación ambiental, así como filtros infrarrojos y ultravioleta, permite medir color, luz ambiental y detección de gestos sin contacto a una distancia de 5 a 20cm (LUIS LLAMAS Ingeniería, informática y diseño, 2018). La comunicación del módulo se realiza a través de I2C y la alimentación es a 3.3 o 5V ya que integra regulador de voltaje.

Se encuentra ensamblado con un emisor de infrarrojos, cuatro fotodiodos direccionales y cuatro más para la medición de color, así como la electrónica adicional para alimentación y salidas (Cooper, 2017).

Para realizar las mediciones a través de un microcontrolador se requiere de la integración de la biblioteca que se puede obtener del sitio web[1] o directamente del Gestor de Bibliotecas del IDE Arduino, escrita por el fabricante Adafruit.

Figura 2.17. Módulo APDS-9960.

Módulo IR FC-51

El módulo infrarrojo FC-51 es un dispositivo optoelectrónico activo que permite medir la proximidad por medio de infrarrojos, se compone de un transmisor que emite energía IR y un receptor que detecta la energía reflejada por cualquier obstáculo que se encuentre frente al

[1] https://github.com/adafruit/Adafruit_APDS9960

módulo, puede ser empleado en luz ambiental o en la obscuridad para proyectos de robótica, industria como el conteo de producción, sistemas de seguridad y/o detección de presencia.

Características:

- Voltaje de operación: 3.3 o 5V
- Salida digital
- Comparador LM393
- Distancia de detección: 20 a 300mm (ajustable por medio del potenciómetro)
- Angulo de detección: 35° (UNIT ELECTRONICS, 2020)

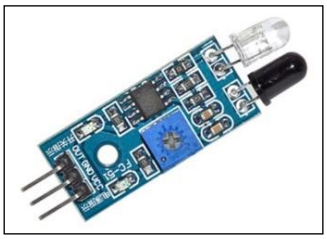


Figura 2.18. Módulo FC-51.

Módulos HC-SR501, HC-SR505, HY3612 y AM312

Estos módulos están construidos con base a un sensor piroeléctrico (PIR, Passive Infrared Sensor), el cual puede detectar cambios en la radiación infrarroja. Todo objeto o ser vivo emite radiación infrarroja en relación con la temperatura a la que se encuentran, a mayor temperatura la radiación aumenta. En base a esta característica se han creado sensores de infrarrojo pasivos, capaces de detectar el movimiento de objetos o seres vivos (punto flotante s.a., 2017).

Características:

- Voltaje de operación: HC-SR501, 4.5 a 12V, HC-SR505, 5 a 20V, AM312, 2.7 a 12V y HY3612, 12 a 24V
- Consumo de corriente en reposo: menos de 50μA en HY3612 hasta 3A depende de la fuente
- Voltaje de salida: 3.3V (HIGH), 0V (LOW)
- Rango de detección: de 3 a 7 metros, ajustable
- Angulo de detección: menos de 120º (en forma de cono)
- Temperatura de trabajo: -20ºC a 80ºC
- Todos con lente protector Frensel (UNIT ELECTRONICS, 2021)

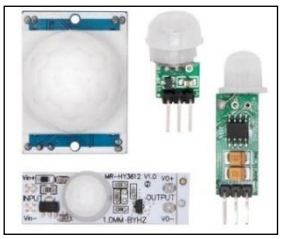


Figura 2.19. Módulos con sensor PIR.

2.3.3 Ultrasonido

Todos los sensores en este apartado permiten el cálculo de distancias por medio de ultrasonido (sonar) para determinar la misma. No requieren del uso de bibliotecas para su operación.

Estos sensores trabajan emitiendo un pulso de sonido (comúnmente denominado TRIGGER), este pulso viaja por el aire hasta rebotar con algún objeto dentro del rango de operación que depende del modelo de sensor, al regresar se mide la anchura del pulso (ECHO), con esto se puede calcular la distancia a partir de la diferencia de tiempo entre el pulso TRIGGER y ECHO (Naylamp Mechatronics, 2018).

Sensor JSN-SR04T

El diseño de este sensor destaca por ser resistente al agua lo que permite su uso en condiciones de alta humedad, agua o uso en intemperie, así mismo el sensor cuenta con módulo externo donde se aloja la electrónica necesaria para su funcionamiento.

Características:

- Voltaje de alimentación: 5V
- Corriente de trabajo: 30mA
- Rango de operación: 25 a 450cm
- Precisión: +/- 0.3mm
- Ángulo de detección: menor a 50º
- A prueba de agua (parte delantera)
- Temperatura de trabajo: -10° a 70ºC (Naylamp Mechatronics, 2018)

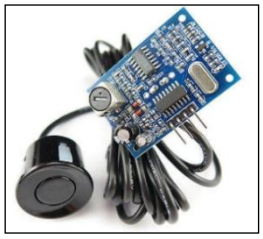


Figura 2.20. Sensor JSN-SR04T.

Módulo GY-US42

Módulo de alta precisión utilizado comúnmente para el control de: vuelo en drones, robots inteligentes, medición en equipos de laboratorio, pruebas de producción, laboratorios antropométricos, coches inteligentes entre otros. Para la comunicación con microcontroladores utiliza el protocolo I2C

Características:

- Voltaje de Operación: 5V
- Corriente de trabajo: 9mA
- Rango de operación: 20 a 720cm
- Precisión: +/- 3mm

- Ángulo de operación: 15°
- Temperatura de trabajo: -20° a 65ºC (VISTRONICA S.A.S, 2018)

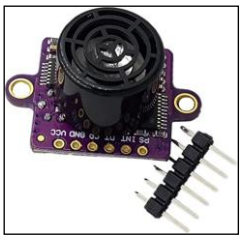


Figura 2.21. Módulo GY-US42.

Módulos HC-SR04, US-016 y US-100

Los módulos como el sensor HC-SR04 poseen dos transductores expuestos: un emisor y un receptor piezoeléctricos, además de la electrónica necesaria para su operación (NAYLAMP MECHATRONICS, 2018). Así el sensor US-016 es similar al HC-SR04, pero con salida de tipo analógico (NAYLAMP MECHATRONICS, 2018). También el sensor US-100 con la diferencia de que su salida es de tipo UART/Serial (NAYLAMP MECHATRONICS, 2018).

Características:

- Voltaje de Operación: 5V
- Corriente de trabajo: 15mA
- Rango de operación: 2 a 450cm
- Precisión: +/- 3mm
- Ángulo de operación: 15° (NAYLAMP MECHATRONICS, 2018)

Figura 2.22. Módulo HC-SR04.

2.3.4 Indicador electrónico de nivel de agua con ULN2803

Con la finalidad de tener una mejor precisión y control del microcontrolador sobre el nivel de agua se optó por la construcción de un indicador para medir el nivel de agua y depender de las lecturas de este para la activación del actuador correspondiente.

El montaje y funcionamiento, requiere de un circuito integrado ULN2803, se emplea el empaquetado tradicional DIP de 16 pines, en su interior cuenta con 8 puertas lógicas inversoras no convencionales ya que estas son implementadas con arreglo de transistores NPN tipo Darlington (dos transistores bipolares conectados formarán el par Darlington, ver figura 2.23 (Agustin, 2015)). Este arreglo permite que la corriente amplificada por el primer transistor

ingrese en la base del segundo transistor y se vuelva a amplificar nuevamente, el par Darlington se comporta como un solo transistor convencional, es decir, tras la combinación de los dos transistores sigue teniendo una sola base, un colector y un emisor, solo que la ganancia de corriente será la combinada. ((Alias) I. , 2020). Adicional también se requiere de 8 resistores de 2.7kΩ, como se indica en el datasheet para operar con circuitos TTL a 5V (Toshiba Semiconductor, 2006).

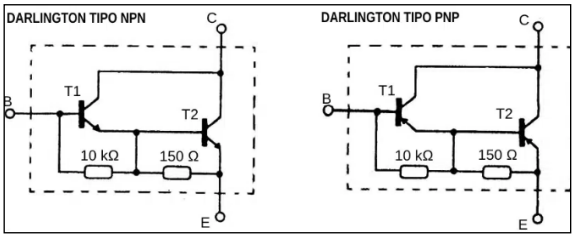


Figura 2.23. Arreglo Darlington NPN y PNP.

Diagrama de conexión:

Este circuito integrado admite voltajes de alimentación y de salida digital de hasta 50V, esto permite transformar señales digitales TTL de 5V a cualquier otro nivel de voltaje de hasta 50V. Este "Driver", aísla y hace de barrera eléctrica entre los circuitos lógicos digitales y los dispositivos de salida que requieren distintos voltajes y grandes corrientes. Este mismo circuito integrado cuenta con diodos en cada una de las salidas ver figura 2.24, para evitar flujos negativos transitorios de corriente que pudieran dañar sus transistores cuando se producen las conmutaciones de circuito cerrado a no circuito abierto (Agustin, 2015).

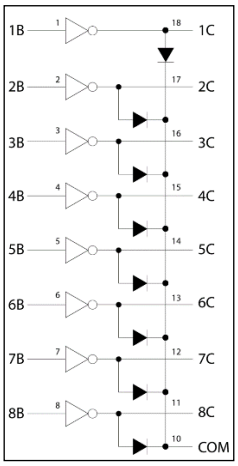


Figura 2.24. Diagrama lógico del circuito integrado ULN2803.

El modo de operación es el siguiente: se conectan las terminales, ver figura 2.25, Input de entrada a un voltaje positivo entre 3 y 5V, así, al estimular cada una de las entradas los transistores en el interior del circuito integrado conducen y quedan eléctricamente conectados los terminales Output y el negativo de la alimentación. Por lo tanto, se cierra el circuito enviando la señal indicando el estado HIGH correspondiente a las terminales de Output (Agustin, 2015).

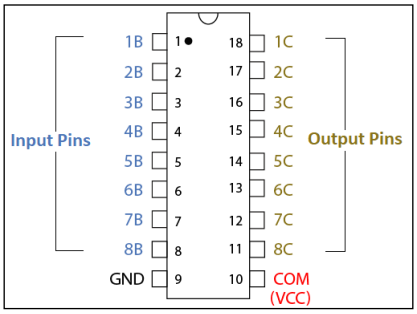


Figura 2.25. Pinout de circuito integrado ULN2803.

Para esta aplicación la demanda de corriente que requiere a su entrada (alta impedancia) para activar la conducción de los transistores es prácticamente nula, este nivel, funciona correctamente cuando el agua no es completamente pura, al contener ciertas sales en disolución conduce en mayor o menor medida la electricidad, de manera que esta, al alcanzar cada uno de los niveles en el depósito o contenedor, cierra el circuito eléctrico de las entradas del ULN2803, que detectan el nivel alto de voltaje y envía la señal indicadora correspondiente al microcontrolador (Agustin, 2015).

Para realizar el registro del nivel al cual se encuentra el agua, se emplean electrodos, para este documento se realizó la construcción de estos, ver figura 2.26, y conexión de los mismos al circuito integrado ULN2803, figura 2.28, también se puede hacer uso de la compra de electrodos diseñados con este fin, figura 2.27. Los electrodos generan electrólisis en el agua, por lo que en el programa se incluye un tiempo de espera donde se activa y desactiva el circuito integrado y así de esta manera reducir este efecto, también se recomienda el uso de contactos de acero inoxidable para prolongar la vida útil de estos electrodos ((Alias) C. , 2020).

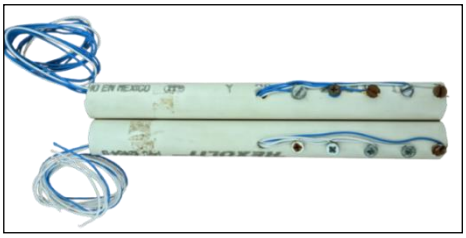


Figura 2.26. Electrodo construido con tubo PVC y tornillos.

Figura 2.27. Electrodos de bronce con cubierta plástica para reducir la corrosión galvánica.

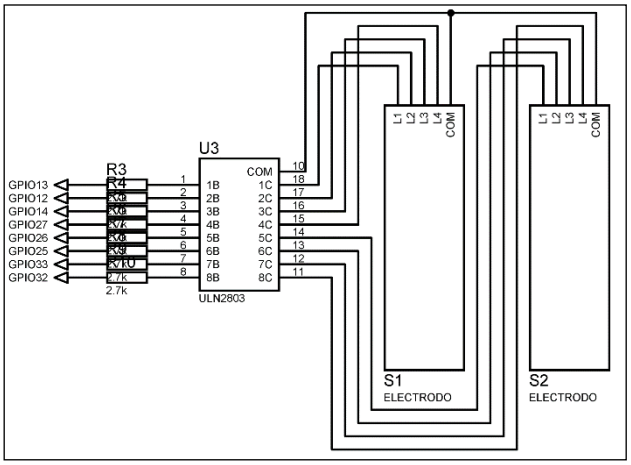


Figura 2.28. Alambrado de conexión electrodos y circuito integrado ULN2803.

2.3.5 Humedad sustrato

Existen sensores que determinan la cantidad de agua en el suelo y sirven de apoyo para definir el momento oportuno para el riego de plantas o cultivos.

Los sensores de humedad miden una señal eléctrica y con esta se puede calcular la cantidad de agua en el sustrato, así como determinar el volumen de agua almacenado en éste después de un riego o lluvia, calcular el consumo de agua por la planta o cultivo en un día o semana y determinar la eficiencia del riego (Cenicaña, 2018).

Sensor HD-38

El módulo consiste en una sonda con dos terminales metálicas resistivas separadas y tratadas para soportar la oxidación y alargar la vida útil y una placa de circuito impreso que realiza el tratamiento de datos entregando una salida tanto análoga como digital.

La salida analógica refleja un voltaje proporcional a la humedad registrada, la salida digital es a través de un comparador, su nivel lógico puede ser ajustado por medio del potenciómetro en el módulo y sus valores solo pueden ser HIGH o LOW.

Características:

- Voltaje de Operación: 3.3 a 12V
- Corriente de trabajo: 20 a 30mA
- Temperatura de operación: -25 a 85°C (ES ELECTROSTORE, 2020)

Figura 2.29. Sensor y módulo HD-38.

Sensor FC-28, módulos YL-38 o YL-69

Este sensor realiza la medición de la humedad del sustrato por medio de dos electrodos resistivos, el modo de sondeo se realiza midiendo la resistencia que existe entre los electrodos insertados en el sustrato, por lo tanto un sustrato muy húmedo tendrá una resistencia muy baja y por el contrario en sustrato seco tendrá una resistencia muy alta, los electrodos van conectados a un módulo que entrega tanto una salida analógica como una digital, en la salida digital por medio de un comparador, su nivel lógico puede ser ajustado a través del potenciómetro en el módulo y sus valores solo pueden ser HIGH o LOW, por su parte la salida analógica entrega valores de 0 a 5V dependiendo de la humedad del sustrato (NAYLAMP MECHATRONICS, 2016).

Recomendación, al introducir en el sustrato solo deberá ser la parte de los electrodos ya que los circuitos no cuentan con protección contra la humedad y oxidación, así como también para extender la vida útil de los mismos se pueden alimentar solo cuando se realicen lecturas (NAYLAMP MECHATRONICS, 2016).

Características:

- Voltaje de alimentación: 3.3 a 5V
- Corriente de operación: 35mA
- Voltaje de señal de salida analógico: 0 a 5V
- Voltaje de señal de salida digital: 3.3 o 5V TTL
- Amplificador operacional LM393 en modo comparador (NAYLAMP MECHATRONICS, 2016)

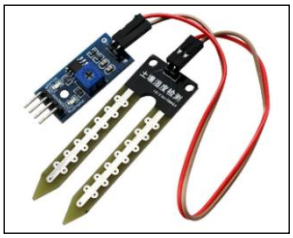


Figura 2.30. Sensor FC-28 con módulo.

Sensor SEN-HS-CAP

Este sensor de humedad de sustrato capacitivo permite medir la misma por medio del principio de capacitancia entre dos electrodos aumentando considerablemente su vida útil, adicional el electrodo posee una capa de protección anticorrosión (NAYLAMP MECHATRONICS, 2020).

La capacitancia entre los electrodos dependerá de la humedad del sustrato, por lo que para un sustrato muy húmedo tendremos una capacitancia muy baja y para un sustrato muy seco la capacitancia será muy alta. El electrodo cuenta con la electrónica de acondicionamiento que entrega solo una salida analógica, con voltaje de operación de 0V hasta 5V dependiendo de la humedad del sustrato (NAYLAMP MECHATRONICS, 2020).

Al igual que los electrodos resistivos al introducir en el sustrato solo deberá ser la parte del electrodo ya que los circuitos no cuentan con protección contra la humedad, si se desea se puede

añadir una capa protectora utilizando laca o barniz epóxico (NAYLAMP MECHATRONICS, 2020).

Características:

- Voltaje de alimentación: 3.3 a 5V
- Corriente operación: 5mA
- Voltaje de la señal de salida: 0 a 5V (analógico) (NAYLAMP MECHATRONICS, 2020)

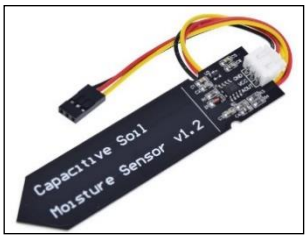


Figura 2.31. Sensor de humedad capacitivo.

2.3.6 Lector de tarjetas RFID

RFID (identificación por radiofrecuencia) es un sistema de almacenamiento y recuperación de datos remoto sin contacto basada en el uso de lectores mediante la radiación electromagnética de radiofrecuencia para identificar y registrar objetos automáticamente (© 2021 www.masterlogistica.es, 2018). Esta tecnología usa dispositivos denominados etiquetas, tarjetas o tags RFID. El propósito de esta tecnología es transmitir la identidad de un objeto (número de serie único) mediante ondas de radio (ESTUDIOS EUROPEOS DE POSTGRADO Y EMPRESA, S.L.).

Las etiquetas, tarjetas o tags RFID son unos dispositivos pequeños, estas pueden ser adheridas o incorporadas a un producto, un animal o una persona. Contienen antenas para permitirles recibir y responder peticiones por radiofrecuencia desde un emisor-receptor RFID. Las etiquetas pasivas no necesitan alimentación eléctrica interna, mientras que las activas sí lo requieren. Una de las ventajas del uso de radiofrecuencia es que no se requiere visión directa entre emisor y receptor (ESTUDIOS EUROPEOS DE POSTGRADO Y EMPRESA, S.L.).

Módulo PN532

Módulo de diseño compacto en este mismo incluye la antena impresa sobre el PCB, así como tres interfaces de comunicación: SPI, I2C y UART, trabaja con tarjetas y dispositivos NFC (soporta comunicación peer to peer) en la frecuencia de 13.56 Mhz (©2021 Naylamp Mechatronics SAC, 2018).

El chip que incluye el módulo permite leer y escribir tags, comunicación con teléfonos con soporte NFC y así también actuar como tag RFID/NFC por emulación. Requiere de la instalación de biblioteca para el IDE Arduino escrita por Adafruit y se puede integrar desde el

gestor de bibliotecas del mismo IDE o en el sitio web[1] (©2021 Naylamp Mechatronics SAC, 2018).

Características:

- Voltaje de operación: 3.3 a 5V
- Frecuencia de operación: 13.56 MHz
- Interfaces: SPI, I2C, High-speed UART (HSU)
- Transferencia de datos: Max. 10Mbit/s
- Compatible con el estándar RFID: ISO/IEC 14443-A
- Compatible con el estándar NFC: ISO/IEC 18092
- Tipos de tarjetas compatibles: Mifare 1K S50, S70 Mifare1, Mifare Ultralight, Mifare Pro, Mifare DESFire, Mifare Classic (©2021 Naylamp Mechatronics SAC, 2018).

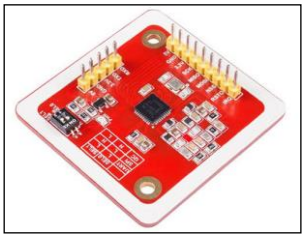


Figura 2.32. Módulo RFID/NFC PN532.

Módulo RC522

Módulo lector y grabador de RFID con una frecuencia de trabajo de 13.56 Mhz. Para la comunicación se realiza por SPI, utiliza un sistema de modulación y demodulación para dispositivos pasivos con el estándar ISO 14443-A así como soporte de encriptación Quick CRYPTO1 y MIFARE (©2021 Naylamp Mechatronics SAC, 2018).

Características:

- Voltaje de Operación: 3.3V
- Corriente de Operación: 13mA (Standby) a 26mA
- Frecuencia de operación: 13.56 Mhz
- Transferencia de datos: Max. 10Mbit/s
- Tipos de tarjetas compatibles: Mifare1 S50, S70 Mifare1, MIFARE Ultralight, Mifare Pro, Mifare DESFire.
- Temperatura de operación: -20 a 80°C (©2021 Naylamp Mechatronics SAC, 2018)

[1] https://github.com/adafruit/Adafruit-PN532/

Figura 2.33. Módulo RFID RC522.

2.3.7 Temperatura y humedad

Los sensores que se presentan en esta sección comparten gran parte de sus características, todos ellos son digitales, proporcionan la medición de temperatura y humedad relativa, integran un sensor capacitivo de humedad y un termistor para medir el aire circundante. Se alimentan con un voltaje de 3V a 5V, los datos recopilados los entregan por medio de salida digital, cada sensor es calibrado en fabrica asegurando alta estabilidad y fiabilidad, los coeficientes de la calibración son grabados en su memoria OTP integrada (©2021 Naylamp Mechatronics SAC, 2015). Requieren de la instalación de biblioteca para su uso en el entorno del IDE Arduino que se puede obtener del sitio[1], esta biblioteca es escrita por Adafruit y funciona para los tres tipos de sensores DHT 11, DHT 21 y DHT 22, la diferencia más notable entre estos es la resolución y precisión de las mediciones (©2021 Naylamp Mechatronics SAC, 2015).

Sensor DHT 11

Características:

- Voltaje de Operación: 3 a 5V
- Rango de medición de temperatura: 0 a 50 °C
- Precisión de medición de temperatura: +/- 2 °C
- Resolución temperatura: 0.1°C
- Rango de medición de humedad: 20 a 90% humedad relativa
- Precisión de medición de humedad: 5% humedad relativa
- Resolución humedad: 1% humedad relativa
- Interfaz digital: Un solo bus (bidireccional) (©2021 Naylamp Mechatronics SAC, 2015)

[1] https://github.com/adafruit/DHT-sensor-library

Figura 2.34. Módulo con sensor DHT 11.

Sensor DHT 21

Dada la construcción de este módulo se recomienda para uso en exteriores.

Características:

- Voltaje de Operación: 3.5 a 5.5V
- Consumo de corriente: 1mA a 1.5mA
- Rango de medición de temperatura: -40 a 80°C
- Precisión de temperatura: +/- 0.5°C
- Resolución de temperatura: 0.1°C
- Rango de humedad: 0 a 100% humedad relativa
- Precisión de humedad: +/- 3% humedad relativa
- Resolución de humedad: 0.1% humedad relativa
- Interfaz digital: Un solo bus (bidireccional)
- Modelo: AM2301
- Longitud cable: 50cm (©2021 Naylamp Mechatronics SAC, 2018)

Figura 2.35. Módulo con sensor DHT 21.

Sensor DHT 22

Características:

- Voltaje de Operación: 3 a 6V
- Rango de medición de temperatura: -40 a 80 °C
- Precisión de medición de temperatura: +/- 0.5 °C
- Resolución Temperatura: 0.1°C
- Rango de medición de humedad: 0 a 100% humedad relativa
- Precisión de medición de humedad: 2% humedad relativa
- Resolución Humedad: 0.1% humedad relativa

- Interfaz digital: Un solo bus (bidireccional)
- Modelo: AM2302 (©2021 Naylamp Mechatronics SAC, 2019)

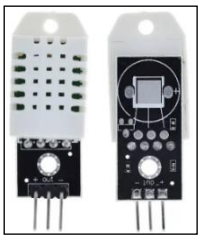


Figura 2.36. Módulo con sensor DHT 22.

2.3.8 Presión atmosférica y altitud

Para la medición de presión atmosférica, se emplea un sensor de presión barométrica que permite calcular la altura respecto al nivel del mar con gran precisión, esto lo consigue al medir la relación entre la presión del aire y la altitud (©2021 Naylamp Mechatronics SAC, 2018).

Todos los módulos en esta sección tienen como base un sensor de presión barométrica piezoresistivo fabricado por BOSCH en su división Sensortec, con variaciones solo en algunas de sus prestaciones por modelo.

Para su uso requiere el uso de la biblioteca escrita por Adafruit que se puede obtener desde el gestor de bibliotecas desde el buscador escribir BMP180 o para instalación manual del archivo .ZIP desde el sito[1] para los módulos BMP085 y BMP180, buscar BMP280 o bajar la biblioteca del sitio web[2] para este módulo.

Módulo GY-BME/BMP280

Características:

- Sensor BMP280
- Voltaje de operación: 1.8 a 3.3V
- Interfaz de comunicación: I2C o SPI
- Rango de presión: 300 a 1100 hecto pascales (0.3 a 1.1bar)
- Resolución: 0.16 pascal
- Precisión absoluta: 1 hecto pascal
- Medición de temperatura incluida
- Resolución de temperatura: 0.01°C
- Precisión temperatura: 1°C
- Frecuencia de muestreo: hasta 157 Hz
- Rango de altura medible: 0 a 9100 metros (©2021 Naylamp Mechatronics SAC, 2018)

[1] https://github.com/adafruit/Adafruit-BMP085-Library
[2] https://github.com/adafruit/Adafruit_BMP280_Library

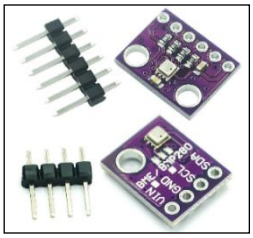


Figura 2.37. Módulos BME/BMP280.

Módulo BMP180, GY-68

Características:

- Sensor BMP180
- Voltaje de operación: 3.3 a 5V
- Interfaz de comunicación: I2C
- Rango de presión: 300 a 1100 hecto pascales (0.3 a 1.1bar)
- Resolución: 1 pascal
- Precisión absoluta: 1 hecto pascal
- Medición de temperatura incluida
- Resolución de temperatura: 0.1°C
- Precisión temperatura: 1°C
- Frecuencia de muestreo: hasta 120 Hz
- Rango de altura medible: 0 a 9100 metros (©2021 Naylamp Mechatronics SAC, 2015)

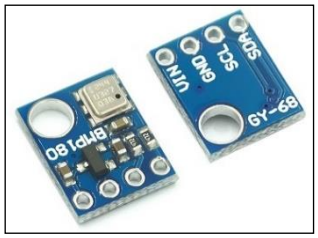


Figura 2.38. Módulo BMP180.

Módulo BMP085, GY-65

Características:

- Sensor BMP085
- Voltaje de operación: 3 a 5V
- Salida lógica: 3 a 5V
- Comunicación: I2C dirección de 7 bits (0x77)
- Rango de censo: 300 a 1100 hecto pascales (-500m a 9000m sobre el nivel del mar)
- resolución: 0.03hPa / 0.25m
- Rango de censo temperatura y operación: -40 a 85°C, +/- 2°C (Adafruit Industries, LLC, 2011)

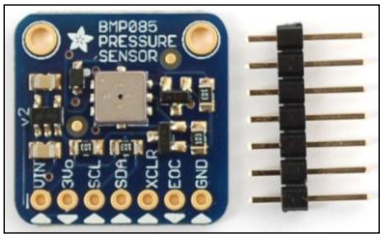


Figura 2.39. Módulo BMP085.

2.3.9 Radiación UV

Para proyectos de monitoreo de condiciones ambientales como en aplicaciones meteorológicas o medición de índice de luz ultravioleta (UV), por ejemplo, para el cuidado de la piel, la medición industrial de nivel UV, entre otros (©2021 Naylamp Mechatronics SAC, 2015). Se requiere realizar mediciones de la intensidad de luz UV, para ello existen en el mercado múltiples opciones de sensores cada uno con características y aplicaciones diferentes y su elección depende de las necesidades del proyecto.

Módulo SI1145 y VEML6075

El módulo SI1145 puede realizar el cálculo de índice de radiación ultravioleta (UV) por aproximación en función de luz visible y luz infrarroja ya que no cuenta con componente destinado a la detección de UV real, a pesar de esto es un sensor muy preciso. El módulo se comunica por I2C, y como se menciona cuenta con elementos para la detección de luz visible e infrarroja, se puede alimentar con 3 o 5V ya que cuenta con regulador de voltaje. Requiere la instalación de biblioteca escrita por el fabricante Adafruit que se puede obtener por medio del gestor de bibliotecas del IDE Arduino o instalando el archivo .ZIP descargado del sitio[1] (Adafruit Industries, LLC, 2014).

Características:

- Voltaje de operación: 3 a 5V
- Espectro del sensor infrarrojo: longitud de onda: 550nm a 1000nm
- Espectro del sensor de luz visible: longitud de onda: 400nm a 800nm (centrado en 530)
- Comunicación: I2C, dirección 0x60 (7 bits)
- Temperatura de trabajo: -40 a 85 °C (Adafruit Industries, LLC, 2014)

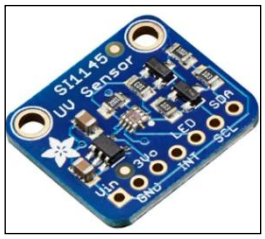


Figura 2.40. Módulo SI1145.

[1] https://github.com/adafruit/Adafruit_SI1145_Library

El módulo VEML6075 obtiene datos precisos de radiación UV ya que a diferencias del módulo SI1145 este si cuenta con elemento para la detección de radiación UV provee al módulo la capacidad de medición de luz UVA y UVB entregando los datos de forma digital por comunicación I2C (dirección 0x10), ese módulo puede ser alimentado con 3 o 5V (Adafruit Industries, LLC, 2018).

La biblioteca para su uso se puede obtener desde el gestor de bibliotecas del Arduino IDE o para instalación manual del archivo .ZIP desde el sitio web[1], escrita por el fabricante Adafruit (Adafruit Industries, LLC, 2018).

Figura 2.41. Módulo VEML6075.

Módulo GUVA-S12SD

El módulo incluye un sensor para detección de UV real en el rango de luz de 240 a 370nm por tanto cubre el espectro UVB y parte del UVA, ya que la señal de salida analógica es pequeña se recomienda el uso de un amplificador operacional para amplificar la señal, no requiere del uso de bibliotecas, el módulo se puede alimentar con 2.7 hasta 5.5V (Adafruit Industries, LLC, 2014).

Figura 2.42. Módulo GUVA-S12SD.

Módulo GY-ML8511

El módulo ML8511 incluye un elemento para la detección de luz UV, a su salida entrega una señal de voltaje analógica que depende de la cantidad de luz UV que detecta. El sensor es capaz de detectar luz de la longitud de onda entre 280 a 390nm, cubriendo el espectro de UVB y UVA, no requiere la instalación de bibliotecas para su uso (©2021 Naylamp Mechatronics SAC, 2015).

Características:

- Voltaje de alimentación: 3.3 o 5V

[1] https://github.com/adafruit/Adafruit_VEML6075

- Salida de voltaje analógica lineal de 1 a 3V
- Longitud de onda: 280 a 390nm (©2021 Naylamp Mechatronics SAC, 2015)

Figura 2.43. Módulo GY-ML8511.

2.3.10 Anemómetro

Un anemómetro es un dispositivo construido con la finalidad de medir la velocidad del viento, por tanto, su uso es siempre en exteriores, es un instrumento común de una estación meteorológica (Adafruit Industries, LLC, 2014).

Ninguno de los dispositivos aquí mencionados contiene componentes electrónicos activos, su funcionamiento es a base de interruptores de lámina magnéticos e imanes (Spark Fun Electronics, Inc, 2018). Por lo que ambos anemómetros tanto el de Adafruit y Sparkfun requieren de alimentación externa para realizar cualquier medición.

De manera que, para la construcción de un anemómetro se sugiere el uso de un motor de corriente continua, facilita la medición, con esto se prescinde del uso de electrónica adicional para la misma.

Anemómetro de Adafruit

Características:

- Voltaje de alimentación: 7 a 14V
- Salida de voltaje analógico: 0.4 a 2.0V
- Velocidad mínima de detección: 0,2 m/s
- Resolución: 0,1 m/s
- Precisión: la más baja es de 1 m/s
- Velocidad máxima de detección: 70 m/s (Adafruit Industries, LLC, 2014)

Figura 2.44. Anemómetro Adafruit.

Sparkfun SEN-15901

Este es un kit más completo para el censo de las condiciones meteorológicas, consta con tres componentes principales de medición, velocidad del viento, dirección y precipitación pluvial.

El pluviómetro es tipo cubeta, de vaciado automático y cuenta con un switch de activación momentáneo que se activa por cada 0.011 de pulgadas de agua. El anemómetro se alimenta con un voltaje de 9 a 24V produciendo una salida de 0 a 5V y comenzando el censo de la velocidad de 0,4 a 0,8 m/s con una resolución de 0,1 m/s y alcanzando una velocidad máxima de 30 m/s. Por último, el kit cuenta con una veleta que informa la dirección del viento por medio de un voltaje producido a la salida de la combinación de resistencias y puede indicar hasta 16 posiciones diferentes (Spark Fun Electronics, Inc, 2018).

Figura 2.45. Kit Sparkfun.

Construcción de anemómetro

Para la construcción del anemómetro se requiere de un motor de corriente continua pequeño y con la rotación más suave posible, en la confección de las aspas se deberá tomar en cuenta el tamaño del motor para este proyecto, para el que se emplea en las pruebas de los experimentos en el capítulo siguiente, se ha realizado la adaptación de unas cucharas desechables, la disposición de estas siempre deberá ser impar y puede ser de 3 o 5 aspas (Guazzaroni, 2017).

Con ayuda de un corcho, madera o plástico se deberán fijar las aspas al eje del motor, fijando la sujeción con pegamento, si el motor lo requiere se agrega una parte de cable a las terminales de este (Guazzaroni, 2017).

Figura 2.46. Anemómetro casero.

Este anemómetro es muy simple y funcional. Para su uso de manera más precisa requiere que este sea calibrado respecto a los valores que otorga a su salida, para ello se deberá tomar varias lecturas a diferente velocidad (puede ser a bordo de un vehículo, por la ventanilla lo más alejado de la estructura del vehículo para reducir el error) las lecturas se realizan midiendo el voltaje que produce la rotación del eje por efecto del viento con un multímetro y estos valores se

comparan con la velocidad conocida, así tomando varias medidas se puede referenciar el voltaje producido a una velocidad determinada, una vez conocemos dicho valor será referencia en las lecturas que tome el microcontrolador al que se conecte el anemómetro. La salida que produce el motor de corriente continua es un voltaje analógico por tanto no se requiere del uso de bibliotecas para la programación del microcontrolador (Guazzaroni, 2017).

2.4 Actuadores

2.4.1 Indicadores visuales

Un indicador visual tal y como dice su nombre proporcionan información inmediata y clara sobre el accionamiento de algún dispositivo empleado mostrando así el estado de este, existe una gran variedad de estos indicadores y pueden ser tan simples como, por ejemplo, visualizar una luz o más complejos como pantallas de alta resolución que pueden mostrar texto e imágenes.

Diodo emisor de luz

Los LED son componentes electrónicos semiconductores capaces de emitir radiación electromagnética para generar iluminación. El desglose de las siglas LED proveniente del inglés y su significado es: L “Light” E “Emitting” D “Diodo”. Es decir, diodo emisor de luz (Villar, 2020).

La estructura se compone por dos terminales de conexión (una más corta que otra, en LEDS de tecnología through-hole), denominadas como ánodo (de polaridad positiva) y cátodo (de polaridad negativa), por lo general un chip o LED contiene un semiconductor, que es el emisor de luz, dos terminales (yunque y poste) donde se fija el chip a través de los cuales le llega la corriente, todo es encapsulado por un material exterior denominado epoxy que sirve como protector, así como reflector. Si se conecta la polarización del diodo semiconductor como indica permite el paso de corriente entre el cátodo y ánodo hasta el semiconductor que finalmente emite luz (Villar, 2020).

Algunos tipos de LED son:

- LED DIP o THT: Con un tamaño entre 3mm a 5mm, tecnología through-hole (THT) (FactorLED, 2018).
- LED SMD: De tamaño aún más reducido respecto a la altura, entre 2.8mm a 5.7mm de tecnología de montaje de superficie (FactorLED, 2018).
- AMOLED, OLED, QLED y MicroLED: Tecnologías LED de tamaño micrométrico que pueden auto iluminarse, organizados en matriz de pixeles normalmente utilizados para la construcción de paneles de televisores (Alonso, 2022).
- LED HIGH POWER: Se utilizan para efectos de iluminación con lente (Villar, 2020).
- LED COB: Máxima resistor y durabilidad, diseñadas para iluminación en exteriores (Villar, 2020).

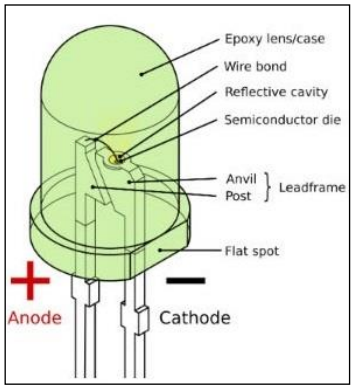


Figura 2.47. Esquema de LED THT y simbología.

Pantalla de cristal liquido

Son pantallas delgadas construidas con una capa de material de cristal líquido intercalada entre dos electrodos sobre sustratos de vidrio, con dos polarizadores en cada lado, formando una matriz de pixeles de color monocromo. Como la pantalla LCD no puede emitir luz por sí misma, normalmente se coloca una luz de fondo detrás de esta para poder verla en entornos obscuros. Si se desea tener una pantalla a color, se puede agregar una capa de filtro de color en el panel LCD. Este tipo de display se utiliza en infinidad de dispositivos debido a su bajo consumo y a que tienen una vida útil muy larga (Orient Display (USA) Corp., 2020).

Algunos tipos de pantallas LCD:

- LCD de matriz pasiva: Su uso es principalmente en aplicaciones sencillas que no requieran de visualización de gráficos complejos, algunas de sus limitaciones son: ángulo de visualización estrechos, velocidad de respuesta lenta, sin retroiluminación problemas de visualización en la oscuridad, como principal ventaja es que son de un consumo energético muy bajo (Diosdado, Zona Maker, 2016).

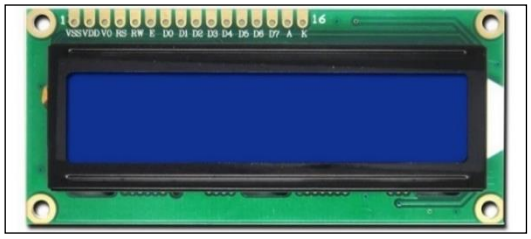


Figura 2.48. Pantalla LCD 2 líneas y 16 caracteres por línea.

- LCD TFT: Basados en transistores de película fina (del inglés Thin Film Transistor), con el que se consigue mejorar la calidad de la imagen (Fernández, 2021).

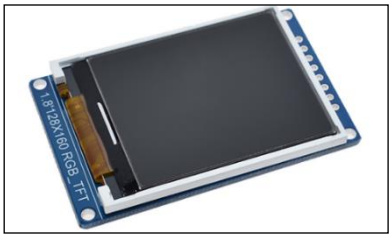


Figura 2.49. Pantalla LCD TFT, 128x160 pixeles, color RGB.

- LCD IPS: Las pantallas cambio de plano (del inglés In-Plane Switching) ofrecen ángulos de visión más amplios y una reproducción de color de mayor calidad gracias a que los cristales se excitan de una manera diferente, reducen su consumo frente a las pantallas TFT (Fernández, 2021).

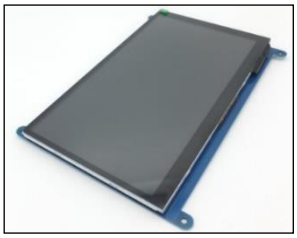


Figura 2.50. Pantalla LCD IPS 1024x600 pixeles, color RGB.

2.4.2 Interruptores

Un interruptor es un dispositivo que permite abrir y cerrar circuitos eléctricos en condiciones normales de carga. Su funcionamiento consiste en dejar pasar o no la corriente en un circuito eléctrico (SIMON, S.A., 2018), generalmente de forma manual.

Push button

Un push button o pulsador es un tipo de interruptor momentáneo. Los contactos dentro del interruptor se cierran al ejercer una fuerza sobre este (encendido - ON) y se abren al retirar dicha fuerza (apagado - OFF). Normalmente están diseñados para montarse en una placa de circuito impreso (PCB) (© Geek Factory 2013 - 2022, 2021). Existen de 2 y 4 pines por lo que se puede pensar que hay que conectar las 4 terminales, pero de manera interna dos terminales (de cada lado) están unidos, por tanto, este pulsador es también de solamente 2 pines. Algunas de las características de estos pulsadores son:

- Configuración de contacto: SPST (1 polo, 1 tiro)
- Capacidad de contacto máximo: 50mA a 24VDC
- Capacidad de contacto mínimo: 10μA a 1VDC
- Tipo de Contacto: Normalmente abierto
- Número de pines: 2 o 4 terminales
- Vida mecánico-eléctrico: 500,000 ciclos
- Resistor de contacto: máx. 100mΩ
- Resistor de aislamiento: min. 100MΩ

- Voltaje dieléctrico: 500VCA
- Fuerza de actuación: 1.6 a 2.6N (UNIT Electronics © 2016-2022, 2019)

Figura 2.51. Push Button.

Switch basculante

Es un tipo de interruptor que desconecta y reconecta el circuito eléctrico oscilando en una dirección para abrir el circuito y a la otra dirección para reconectarlo. Los interruptores basculantes son extremadamente comunes y pueden encontrarse en muchos tipos de dispositivos, desde consola de videojuegos hasta instrumentos médicos (Techlandia, 2017).

Se construyen en base a dos componentes, el primero de ellos es un actuador, montado en un eje que le permite rotar hacia adelante y atrás. Este actuador es la parte del interruptor que se mueve y aplica fuerza a los contactos para encenderlo o apagarlo, el segundo componente es el juego de contactos. El estado de los contactos determina si el interruptor está encendido - ON o apagado - OFF (Leaf Group Ltd., 2013). Este tipo de interruptores pueden ser de los tipos: SPST, SPDT, DPST y DPDT (Littelfuse, Inc., 2016), las características varían dependiendo del fabricante, tipo de interruptor y necesidades de aplicación.

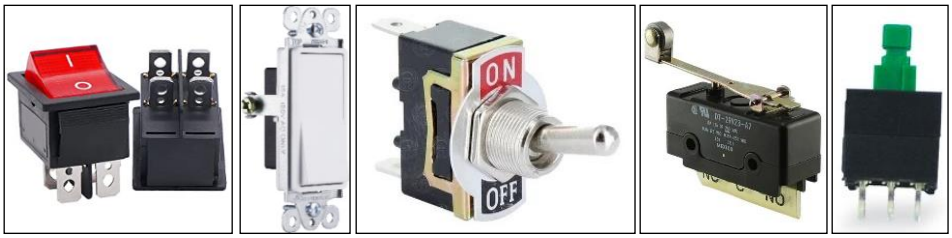


Figura 2.52. Ejemplos de Switch basculante.

Reed switch

Los reed switch respecto a otros interruptores es activado por la presencia de un campo magnético. Si es normal abierto los contactos se cierran en la presencia del campo, cuando es normal cerrado se abren en presencia de un campo magnético, en cualquier caso, sola una lámina presenta movimiento. Su uso y aplicación es más común de lo imaginado, se encuentran por ejemplo dentro de los “sensores magnéticos” que se utilizan en los sistemas de alarma para el hogar (Shoptronica S.L., 2015).

Un reed switch se compone de dos laminas ferromagnéticas compuestas de níquel y hierro, herméticamente selladas en una cápsula de vidrio. Las hojas se superponen internamente en la cápsula de vidrio dejando solo un pequeño espacio entre ellas, y se ponen en contacto ante la

presencia de un campo magnético adecuado. El área de contacto de ambas hojas se recubre con metales duros (rodio o rutenio) que aumentan la vida útil de los contactos (Shoptronica S.L., 2015).

Figura 2.53. Reed switch, de izquierda a derecha empaque comercial, componente electrónico.

Relevadores

Un relevador o relé, es un componente eléctrico que al igual que un interruptor permite abrir o cerrar circuitos eléctricos que pueden ser dependientes o independientes entre sí (REVISTA ESPAÑOLA DE ELECTRÓNICA S.L., 2022), en función de una bobina y un electroimán y son controlados de forma electrónica (R.).

- **Electromecánicos**

De los relevadores electromecánicos existen de tipo armadura, que contienen bobina, núcleo, armadura y contactos y que son las piezas principales de su funcionamiento. Al pasar corriente por la bobina, se magnetiza el núcleo y se transforma en un imán (REVISTA ESPAÑOLA DE ELECTRÓNICA S.L., 2022), así es atraída la armadura al núcleo y con la fuerza aplicada en esta se hace cambiar de posición los contactos. El que estaba abierto se cierra, y el que estaba cerrado se abre.

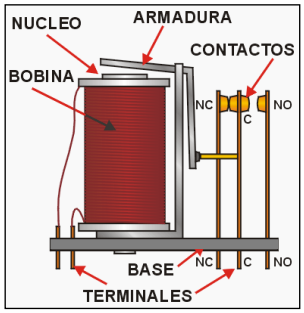


Figura 2.54. Esquema interno de un relé electromecánico.

Se enlistan las características de algunos de los relevadores fabricados por Songle, estas pueden variar dependiendo del modelo seleccionado:

- Modelos: SRD-5VDC-SL-C, SRD-12VDC-SL-C, SRD-24VDC-SL-C, SLA-05, 12, 24VDC-SL-A (este último en voltajes de 5, 12 o 24 V DC).
- Pines: 5
- Contacto: SPDT
- Voltaje nominal de la bobina: 5, 12, 24VDC

- Voltaje y Corriente Máx. Carga: (AC: 240V / 10A o 125V / 10A y DC: 30V / 10A o 28V / 10A) para SRD-5VDC-SL-C, (AC: AC: 240V / 7A o 125V / 10ª y DC: 28V / 10A) para SRD-12VDC-SL-C y SRD-24VDC-SL-C, (AC: 120 a 250V y DC: 30V / 30A MAX) para SLA-05, 12, 24VDC-SL-A
- Corriente nominal en bobina 75mA (5VDC-SL-C), 30mA (12VDC-SL-C), 15mA (24VDC-SL-C), 185, 75, 36mA (SLA-05, 12, 24VDC-SL-A)
- Expectativa de vida: 100,000 accionamientos
- Temperatura de operación -25℃ a 70 ℃ (UNIT Electronics © 2016-2022, 2020)

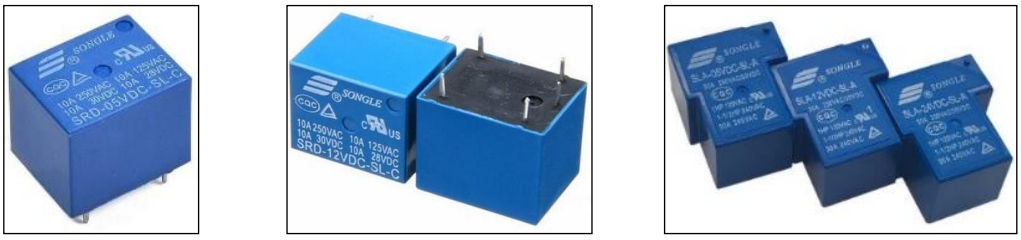


Figura 2.55. Relevadores de marca Songle.

- **Estado solido**

Funciona como switch eléctrico que conmuta entre los estados abierto y cerrado, es similar en tamaño al relé electromecánico, se emplean para operaciones que necesiten una utilización continua de los contactos del relevador, sin que se tenga que producir deterioro mecánico, además de que tienen el beneficio de efectuar el cambio de estado a grandes velocidades (REVISTA ESPAÑOLA DE ELECTRÓNICA S.L., 2022).

Características de relevador SSR-40 fabricado por FQFER:

- Serie: SSR-40
- Tipo: DA (control DC de 3 a 32V – carga AC de 24 a 380V), AA (control AC de 80 a 250V – carga AC de 24 a 380V), DD (control DC de 3 a 32V – carga DC de 5 a 200V)
- Corriente de conmutación: 40A
- Led indicador[1] (UNIT Electronics © 2016-2022, 2022)

Figura 2.56. Relevadores de estado sólido serie SSR-40, marca FQFER.

[1] Las características y valores aquí descritos son con fines de referencia, siempre se deberá consultar la hoja de datos proporcionada por el fabricante del componente seleccionado.

2.4.3 Bombeo de agua

Una bomba, es un dispositivo que se utiliza para mover un fluido mediante la acción de bombeo de un lugar a otro, sin importar el tipo de fluido. Una bomba de agua es aquella máquina que transforma energía, aplicándola para mover el agua, estas constan de un orificio de entrada (de aspiración) y otro de salida (de impulsión) (Euroinnova Formación S.L., 2022). Existen dos clases de bombas de agua, motobombas y electrobombas, las de la primera clase son accionadas por un motor con combustible. Se utilizan para grandes trabajos y siempre están alimentadas por gasolina u otro tipo de combustible, por otra parte, una electrobomba es una bomba de agua accionada eléctricamente, es decir, que está conectada a la corriente eléctrica (Electrobombas Jávea, S.L., 2020).

Los elementos que componen una bomba de agua y que permiten su funcionamiento son:

- Carcasa o Armazón: es el cuerpo en el que se encuentra recubierto su mecanismo. Generalmente debe ser anticorrosión, en acero inoxidable o hierro fundido.
- Entrada y Salida: hueco de entrada por donde pasa el fluido, y la respectiva salida donde se expulsa el mismo.
- Impulsor, Rotor o Rodete: elemento que se usa para poder impulsar el fluido contenido en la carcasa. Pueden ser de tipo aspas, álabes, entre otros.
- Sellos, Retenedores y Anillos: permiten que la bomba selle de manera correcta consiguiendo cierta compresión interna.
- Eje Impulsor: eje que sostiene el impulsor para que gire sobre este.
- Cojinetes o Rodamientos: sirven para sostener adecuadamente el Eje Impulsor
- Panel de Control: permite accionar la bomba de agua, puede contener switches o botones para realizar su encendido o parada, o bien ser controlada de manera remota o automática por medio de un microcontrolador.
- Motor: es el dispositivo que permite mover el eje y, a su vez, el impulsor para que el fluido pueda pasar de un lado a otro, dependiendo de la potencia de este, podrá movilizar mayor cantidad de agua en el menor tiempo posible. El motor puede contener otras piezas especiales, como ventilador, bobina, imanes, etc. (Electrobombas Jávea, S.L., 2020).

El funcionamiento de cualquier tipo de bomba es: el agua es aspirada por el tubo de entrada de la bomba, para luego ser impulsada por el motor, que por acción de un campo magnético generado con las bobinas e imanes logra que el impulsor gire de una manera constante. A medida que gira el rotor, se mueve el fluido que alimenta la bomba. El rodete es el encargado de impulsar los fluidos con fuerza, una vez que el agua entre al centro del rotor la fuerza centrífuga que se forma por la compresión y rotación del fluido genera una presión haciendo que este salga con rapidez y gran caudal (Electrobombas Jávea, S.L., 2020).

Existen gran cantidad de bombas para agua en el mercado aquí solo se muestran algunas de ellas como ejemplo, esta se deberá seleccionar en base a las necesidades del proyecto.



a) b) c)

Figura 2.57. Bombas de agua comercial para uso doméstico (a y b), bomba para Arduino (c).

Bomba de agua sumergible para Arduino

- Voltaje de funcionamiento: 2.5 a 6VDC
- Altura bombeo máx.: 40 a 110cm
- Caudal bombeo máx.: 80 a 120 l/h
- Diámetro salida Exterior: 7.5mm
- Diámetro salida Interior: 5mm
- Tiempo continuo de trabajo; 500 horas (UNIT ELECTRONIC SERVICES MX S.A. DE C.V., 2023)

La Bomba de agua sumergible (figura 2.57. (c)) es un dispositivo electromecánico totalmente sumergible que te permitirá tener un flujo de hasta 2L por minuto (de 70-120L/h) con un voltaje de funcionamiento de 2.5 a 6VDC. Incluye un motor interno de 800mA.

La Bomba de agua sumergible es útil para proyectos como filtros en acuarios (máximo de 100L), cascadas o fuentes, regadores de pasto, entre otros (UNIT ELECTRONIC SERVICES MX S.A. DE C.V., 2023).

2.4.4 Cerraduras tipo solenoide y electroimán

Se denomina solenoide a la bobina que, por su diseño, genera un campo magnético de gran intensidad. Esta bobina, de forma cilíndrica, cuenta con un hilo conductor que está enrollado de forma tal que la corriente provoca la formación de un campo magnético intenso. A través del hilo conductor del solenoide circula la corriente y se genera el campo: mientras más extensa sea la bobina, más uniforme resulta el campo en su interior. De acuerdo con el núcleo, el solenoide puede actuar como electroimán (Porto & Gardey, Definición.DE, 2014).

El accionamiento del solenoide se produce por pulsos eléctricos y pueden controlarse por medio de un programa (Porto & Gardey, Definición.DE, 2014).

Electrocerradura 12VDC

- Modelo de sobreponer de alta eficiencia.
- Cuerpo, cilindro, pestillo y picaporte en latón.
- Tirador recubierto para evitar choques y cortos.
- Instalación derecha/izquierda, abriendo hacia adentro y hacia afuera.
- Memoria mecánica, destraba al primer impulso eléctrico.

- Con o sin botón, para desbloqueo manual mecánico.
- Aplicación en puertas y portones.
- Peso: 300 Kg
- Voltaje nominal: 9 o 12 VDC/VAC dependiendo el modelo
- Corriente: 800mA a 1A (El Gran Tlapalero S.A. de C.V., 2023)

Figura 2.58. Electrocerradura a 12 V.

Cerradura eléctrica tipo solenoide 12VDC

- Voltaje: 12V
- Corriente: 0.8A
- Potencia: 9.6W
- 1~10 segundos el tiempo de activación
- Longitud del cable: 38cm
- Peso: 150g (UNIT Electronics © 2016-2022, 2020)

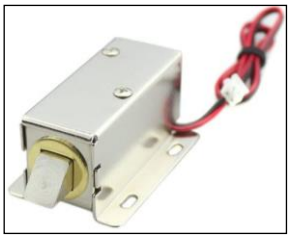


Figura 2.59. Cerradura electrónica a 12V.

Cerradura eléctrica XG-07 5V y 12V

- Modelo: XG-07E, XG-07C
- Corriente de Trabajo: modelo a 5V: 1.2/1.5A, 12V: 1.4/2A (UNIT Electronics © 2016-2022, 2022) (UNIT Electronics © 2016-2022, 2022)
- Voltaje de Funcionamiento: 5 o 12V
- Vida Útil: más de 300,000 veces
- Número de Hilos: 4
- Conectores JST 2P Macho y Hembra de 2.54mm
- Peso: 64g (UNIT Electronics © 2016-2022, 2022) (UNIT Electronics © 2016-2022, 2022)

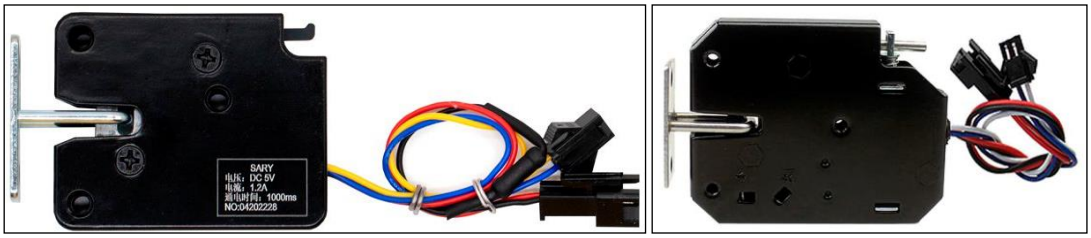


Figura 2.60. Cerraduras electrónicas XG-07E (izquierda) y XG-07C (derecha).

Una cerradura que como base utiliza un electroimán también puede ser empleada en los proyectos aquí descritos, estas como se menciona fundan su funcionamiento en un imán artificial que está construido con un núcleo de hierro dulce y una bobina a su alrededor. Es importante tener en cuenta que para mantener el campo magnético generado se demanda el uso constante de una fuente de alimentación (Porto & Gardey, Definición.DE, 2020).

Cerradura de electroimán Anviz AN-ML160

- Modelo: AN-ML160
- Resistente a corrosión
- Tipo de instalación: empotrable
- Voltaje de alimentación: 12 o 24VDC
- Tipo de alimentación: batería
- Fuerza de resistor máxima: 160Kg (2017 OCompra.com)

Figura 2.61. Cerradura de electroimán Anviz AN-ML160.

2.5 Etapas de potencia

Para el control de cargas que son alimentadas con voltajes e intensidades de corriente mayores a las que las tarjetas de desarrollo o microcontroladores pueden soportar es necesaria la implementación de circuitos de potencia que permitan este control con las pequeñas señales recibidas del microcontrolador. En este capítulo se describen algunos circuitos que pueden ser alambrados para aumentar la capacidad en el control de cargas a una tarjeta de desarrollo como las que integran el microcontrolador ESP32.

2.5.1 Switch por medio de un relevador para corriente alterna o directa

Un relevador o relé no puede ser conectado directamente a una tarjeta de desarrollo como la NodeMCU-32S, esto se debe a que la tarjeta no puede proporcionar ni mantener la corriente suficiente para alimentar la bobina interna con la que cuentan los relevadores, y, si se intenta

realizar la conexión existe la probabilidad de que la tarjeta de desarrollo resulte dañada, este problema no afecta solo a las tarjetas de desarrollo con un microcontrolador como el ESP32, el problema es más general, ya que los voltajes y corrientes con los que trabajan (enviar y recibir señales) la gran mayoría de las tarjetas de desarrollo que integren un microcontrolador son demasiado bajos. Esto está bien para aplicaciones pequeñas como proyectos escolares y/o que exijan muy poca potencia, pero cuando se habla de domótica es necesario la implementación de circuitos que nos permitan controlar las múltiples cargas que pueden existir en una vivienda.

Tal y como ya se mencionó un switch o interruptor es un dispositivo que permite abrir y cerrar circuitos eléctricos en condiciones normales de carga (SIMON, S.A., 2018). Conectar uno o varios relevadores a una tarjeta de desarrollo permite realizar múltiples tareas de automatización, no solo de circuitos de corriente directa, sino también de aquellos que emplean la corriente alterna como la que se recibe a través de la acometida por parte de la compañía suministradora. Permitiendo así controlar luminarias, enchufes, motores y todos esos dispositivos que emplean también corriente alterna como alimentación para su funcionamiento.

Para poder conectar un relevador a un microcontrolador como el ESP32 es necesario separa las señales de control y la alimentación de los dispositivos a controlar, pero como ya se hizo mención, el relevador requiere de una corriente constante para mantener los contactos internos cerrados y así permitir fluir la corriente del circuito de mayor potencia. Por tanto, la corriente que energiza al relevador también se encuentra aislada de la señal de control que se recibe del microcontrolador. El ensamble de las conexiones se puede ver en la figura 2.62, donde se emplean: un resistor (R1) de 1kΩ, R2 de 510Ω, un circuito integrado optoacoplador FL817C (U1), un transistor J3Y(Q1), y un diodo (D2) en configuración flyback, para protección.

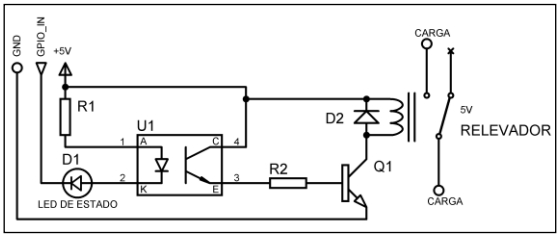


Figura 2.62. Arreglo electrónico en módulos relevadores.

Optoacoplador

Un optoacoplador es un dispositivo que emplea luz para conectar dos circuitos eléctricos manteniendo un aislamiento entre ellos, es decir, la carga eléctrica no tiene un camino para circular entre ambos circuitos (Lourdes Martin Lacasta, 2017).

La única comunicación entre ambos circuitos se realiza a través de la luz, sin existir contacto eléctrico. Frecuentemente se incorpora una barrera dieléctrica entre emisor y receptor para aumentar el aislamiento al orden de kilovolts (Lourdes Martin Lacasta, 2017).

Los optoacopladores normalmente se proporcionan en integrados de tipo DIP y SMD, de 4 a 16 pines. Existen diferentes tipos de optoacopladores, y cada uno requiere un número de pines diferentes. Además, existen integrados con uno o varios optoacopladores en su interior (single,

doble y quad). La combinación de estos determina el número de pines del integrado (Lourdes Martin Lacasta, 2017).

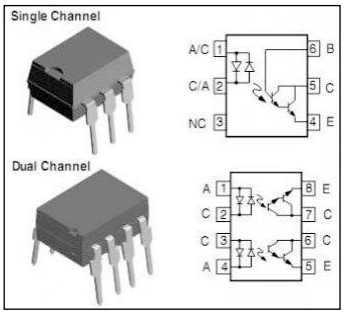


Figura 2.63. Empaquetado DIP de optoacopladores single y dual channel.

Existen muchas variantes de optoacopladores. Por ejemplo, los hay con un único emisor que activa varios receptores. Otros están diseñados para que la salida sea lineal a la entrada, por lo que pueden transmitir señales sin apenas distorsión. Algunos más incorporan funciones de amplificación u offset de la señal. También existen optoacopladores bidireccionales, formados por dos LED.

Los optoacopladores incorporan un emisor y un receptor en un único integrado compacto. En la mayoría de los casos el emisor es un diodo de arseniuro de galio (GaAs) IR LED, mientras que el fotorreceptor puede ser un fototransistor, un fotodiodo + transistor, un fototriac, entre otros. Por tanto, se recomienda revisar la hoja de datos del componente a utilizar antes de realizar cualquier montaje (Lourdes Martin Lacasta, 2017).

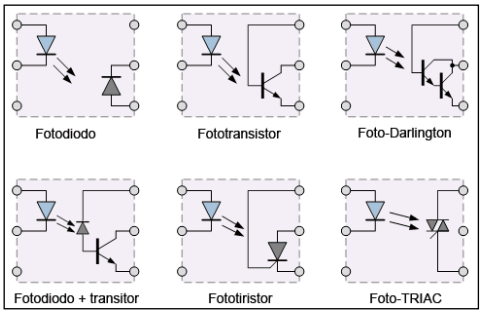


Figura 2.64. Arreglos en optoacopladores.

Relevador

Puede ser alguno de los modelos que se enlistan en el capítulo 2.4.2 fabricados por Songle, u otro fabricante, lo que se recomienda es elegir aquellos con un voltaje de bobina de 5V, en cuanto a las características y soporte para las cargas dependerá del modelo elegido.

Un diodo flyback es un diodo colocado en paralelo a una bobina en circuitos de conmutación. El diodo flyback reduce los altos voltajes y corrientes parasitas que se puedan presentar al conectar o desconectar la alimentación las bobinas (Ricardo, 2022).

Existen módulos de relé comerciales como el de la figura 2.65 (a), este posee dos canales. Existen otros modelos por ejemplo de, cuatro, ocho, dieciséis o más canales 2.65 (b). El módulo, ver figura 2.65 (a), debe ser alimentado con 5V, lo que lo convierte apropiado para usar con una tarjeta Arduino. También los hay que funcionan con 3.3V, ideales para ESP32, ESP8266 y otros microcontroladores, y también hay modelos de 12V o 24V (Eduardo Julio Carletti, 2020). Estos módulos simplifican los diseños y su implementación en proyectos que requieren de un ensamble rápido.

a) b)

Figura 2.65. Módulos de relevadores para Arduino 2 canales (izquierda), 32 canales (derecha).

2.5.2 Control de potencia de corriente alterna (AC)

Conforme al crecimiento de los proyectos que se pueden llevar a cabo a través de una tarjeta de desarrollo que incorpora el microcontrolador ESP32, es necesario poder actuar con dispositivos de otro nivel de potencia, algunos de estos pueden ser accionados mediante transistores BJT, MOSFET, o salidas por relé. Por otra parte, para realizar el control de por ejemplo un controlador de motores, o incluso la red eléctrica, es necesaria una forma de aislar y proteger la electrónica de la capa de control, ya que, si ocurriera un error de conexión o malfuncionamiento de un componente, e incluso el ruido de las señales o transitorios como por ejemplo al arranque de un motor, podría dañar toda la electrónica incluido el microcontrolador (Lourdes Martin Lacasta, 2017).

Existen múltiples formas de conseguir aislamiento entre circuitos, siendo una de las más habituales es el uso de uno o varios optoacopladores (Lourdes Martin Lacasta, 2017), que en arreglo junto a un triodo permite el control de cargas de corriente alterna, que además de requerir de una potencia mucho mayor, con respecto a la manera de energizar al microcontrolador ESP32, son de índole distinta.

2.5.2.1 Optoacoplador 4N25 y MOC3021

Para el circuito “dimmer” presente en este documento se han empleado dos optoacopladores 4N25 y el MOC3021 los cuales cuentan con las siguientes características:

4N25

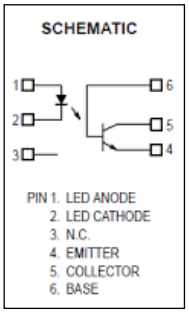


Figura 2.66. Distribución de pines de optoacoplador 4N25.

- Tipo salida del optoacoplador: fototransistor
- Número de canales: 1
- Voltaje de colector-emisor: 30V
- Voltaje de aislamiento: 5kV
- Corriente directa máxima: 60mA
- Encapsulado: Tipo DIP
- Número de pines: 6 (Carrod Electrónica Online S. de R.L. de C.V., 2015)

MOC3021

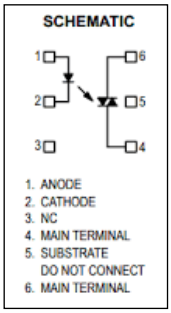


Figura 2.67. Distribución de pines MOC3021.

- Tipo de salida: fototriac
- Modo de funcionamiento: paso por no cero
- Número de canales: 1
- Voltaje máximo de apagado repetitivo: 400V
- Voltaje de aislamiento: 5.3kV
- Voltaje de reversa: 3V
- Corriente continua: 60mA
- Temperatura de operación mínima: -40°C
- Temperatura de operación máxima: 85°C
- Encapsulado: DIP
- Número de pines: 6 (Carrod Electrónica Online S. de R.L. de C.V., 2014)

2.5.2.2 Triodo de corriente alterna BTA16

El triodo de corriente alterna (TRIAC) es un dispositivo semiconductor de tres terminales que se usa para controlar el flujo de corriente a una carga, con la particularidad de que conduce en ambos sentidos entre las terminales A1 y A2 (ánodo 1 y 2), puede ser bloqueado por inversión del voltaje o al disminuir la corriente por debajo del valor de retención. El TRIAC puede ser disparado independientemente de la polarización de puerta (G), es decir, mediante una corriente de puerta positiva o negativa (Carrod Electrónica Online S. de R.L. de C.V., 2014).

BTA16-600B

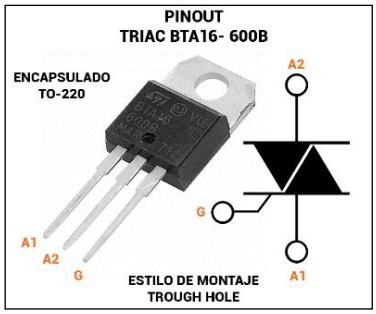


Figura 2.68. Distribución de pines BTA16-600B.

- Tipo: TRIAC
- Tensión máxima de apagado repetitiva: 600V
- Corriente RMS de encendido: 16A
- Corriente máxima de disparo de la compuerta máxima: 100mA
- Potencia de pico de la puerta: 1W
- Corriente máxima transitoria: 160A
- Corriente máxima de retención: 50mA
- Temperatura de operación máxima: 125°C
- Encapsulado: TO-220
- Número de pines: 3 (Carrod Electrónica Online S. de R.L. de C.V., 2014)

Un arreglo con los componentes MOC3021 y BTA16, permite activar o desactivar cargas de potencia mucho mayor, pero sin control alguno, solo podemos encender o apagar dicha carga.

Para poder variar la potencia de salida, el arreglo entre el optoacoplador 4N25 y el microcontrolador permite que este último pueda evaluar y realizar una regulación de la potencia y junto al MOC3021 y el BTA16, ver figura 2.69, por ejemplo, regular la intensidad de la luz de un foco incandescente.

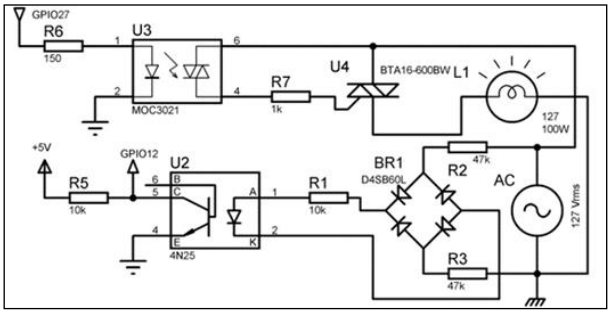


Figura 2.69. Diagrama esquemático de ensamble para control de potencia.

También es necesario realizar la sincronización de la señal de disparo del TRIAC con la frecuencia de la red eléctrica, para ello el optoacoplador 4N25 emite un pulso de voltaje entre 0V y 5V, proporcional a la señal de voltaje previamente rectificada que es recibida y emitida por el LED en el interior de este.

2.5.3 Control de potencia por medio de transistores MOSFET (corriente directa, DC)

Las salidas en las tarjetas de desarrollo con ESP32 están limitadas tanto en voltaje como en intensidad de corriente siendo estas de 3.6V y 40mA como máximo respectivamente, recomendando un valor de uso a 28mA (Guirado, 2020). Así para ejecutar acciones que requieren de un mayor consumo energético, y proveer de aislamiento entre circuitos, en el uso de microcontroladores como el ESP32, esta tarea es delegada a etapas de amplificación o controladores (drivers), que actuaran como adaptadores entre el nivel de potencia empleado en el controlador y el requerido por el actuador.

Los MOSFET son una subfamilia de la familia de transistores FET (Field Effect Transistor). De entre estos existen muchas otras subfamilias de FET, como los JFET (Join Field Effect Transistor), CMOS, TFT, por mencionar algunos. Existen multitud de modelos de transistores FET, cada uno con sus propias características, estos se presentan en una gran variedad de integrados y encapsulados, debiendo consultar su hoja de datos para conocer las características exactas de cada modelo en concreto. Los transistores FET son útiles en múltiples aplicaciones. Algunas de estas son actuar como amplificador o como interruptor controlado electrónicamente permitiendo manipular cargas con voltajes e intensidades de corriente mucho mayores (Lacasta, 2016).

IRF840

Características del transistor MOSFET:

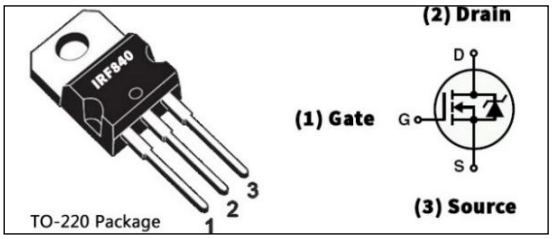


Figura 2.70. Transistor MOSFET IRF840.

- Disipación total del dispositivo (Pd): 125W
- Voltaje drain-source (Vds): 500V
- Voltaje gate-source (Vgs): 20V
- Corriente continua de drain (Id): 8A
- Temperatura operativa máxima (Tj): 150°C
- Voltaje umbral gate-source Vgs(th): 4V
- Carga de compuerta (Qg): 63nC
- Conductancia de drain-sustrato (Cd): 1500pF
- Resistividad drain-source RDS (on): 0.85Ω
- Empaquetado: TO220 - 3 pines
- Peso: 2g (UNIT ELECTRONIC SERVICES MX S.A. DE C.V., 2020).

Los transistores FET se presentan en dos variantes Canal N y Canal P. El funcionamiento de ambas variantes es análogo, pero determina la posición en el montaje. El transistor Canal P se coloca en la parte "Positiva" del circuito, por tanto, el transistor Canal N se pone en la parte "Negativa" del circuito (Lacasta, 2016) ver figura 2.71.

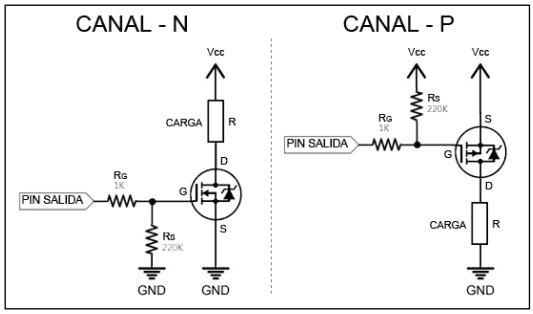


Figura 2.71. Variantes de transistores FET.

Al igual que los transistores de unión bipolar (BJT) un transistor FET dispone de tres terminales. gate (puerta), de trabajo similar a la base de los BJT, source (fuente), similar al emisor del BJT y drain (drenado), similar al colector del BJT. También de forma similar a los transistores BJT los transistores FET presentan tres modos de funcionamiento, aunque la zona activa de los BJT es sustituida por una zona lineal o resistiva, y en comparativo estos modos de funcionamiento son muy diferentes. Corte, el transistor se comporta como un circuito abierto entre source y drain, saturación se comporta como un cortocircuito entre source y drain, zona lineal se comporta como un resistor de valor variable. De esta forma, para encender una carga es necesario hacer funcionar el FET en modo corte o saturación, evitando la zona lineal (Lacasta, 2016).

Un aspecto a tomar en cuenta en el diseño de circuitos con MOSFET es la potencia soportada por el mismo, dado que el transistor deberá ser capaz de disipar esta energía sin dañarse. Un transistor MOSFET solo tiene que soportar una potencia realmente importante en la zona lineal, a medida que se trabaja acercándose al punto de saturación. Una vez saturado, el MOSFET es capaz de soportar una gran intensidad de corriente, con una pequeña disipación de energía (Lacasta, 2016).

Por tanto, para usar el MOSFET como interruptor se deberá evitar la zona lineal ya que en esta la energía disipada es muy elevada, esta energía se convierte en calor y en un incremento de temperatura que puede dañar el transistor. Si la potencia disipada por el MOSFTET es demasiado elevada, será necesario añadir un disipador (Lacasta, 2016).

El funcionamiento de los transistores FET no se basa en la unión de materiales semiconductores, estos crean un canal de conducción entre source y drain dentro de un único material semiconductor. El ancho de este canal es controlado por voltaje aplicado en el terminal gate, a diferencia de los BJT cuyo estado depende de la corriente que circula por la base (Lacasta, 2016).

Como ventaja, en modo de saturación los transistores MOSFET se comportan como un resistor de muy pequeño valor, esto permite manejar enormes cargas con poca disipación de energía. Además, disponen de una alta impedancia en la terminal gate (del orden de 100MΩ), lo que los hace excelentes a la hora de combinarlos para formar circuitos digitales, también los tiempos de conmutación son más rápidos que los BJT, generan menos ruido y son menos sensibles a la temperatura (Lacasta, 2016).

Como contraparte, el terminal gate del transistor se comporta como un condensador. Es decir, el transistor necesita absorber una cierta cantidad de carga eléctrica para cambiar de modo de trabajo. Para llegar a saturar el MOSFET se requiere superar un umbral de voltaje en la terminal gate y proporcionar la carga suficiente para que el MOSFET sature. Así en general, los transistores MOSFET no son apropiados para generar señales PWM simplemente conectándolos directamente a una salida de la tarjeta de desarrollo. Recordemos que el voltaje y la intensidad de corriente proporcionada por una GPIO de la tarjeta de desarrollo con ESP32 no es suficiente para saturar rápidamente al MOSFET (Lacasta, 2016).

Para superar esta limitación se puede emplear una etapa de preamplificación, entre la salida de la tarjeta de desarrollo y el MOSFET. Esta etapa recibe la salida GPIO, y proporciona al MOSFET el voltaje e intensidad de corriente que necesita para saturar completamente, y con una velocidad de conmutación rápida, este pequeño circuito, ver figura 2.72, puede ser alambrado junto con un transistor BJT como el 2N2222, BC337 o similar, y así aprovechar por completo el MOSFET, permitiendo proporcionar grandes cantidades de intensidad de corriente incluso en salidas PWM (Lacasta, 2016).

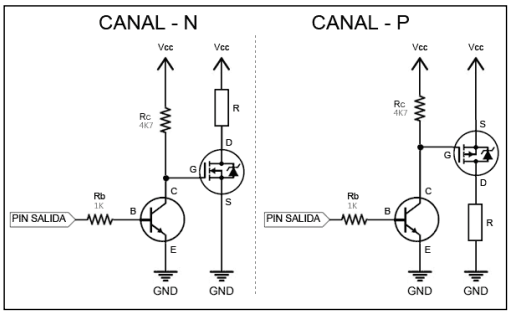


Figura 2.72. Etapa de preamplificación para canal N y canal P.

2.5.4 Fuentes de energía

Como fuentes de energía de corriente directa existen las de tipo electrónico y las baterías. Una fuente de corriente directa se trata de un arreglo electrónico cuya función principal es convertir la corriente alterna de la red doméstica, en corriente directa para alimentar los distintos circuitos de los aparatos electrónicos. Las fuentes de alimentación para uso en dispositivos electrónicos pueden clasificarse en dos: fuentes de alimentaciones lineales y conmutadas (Pérez, 2021).

Las fuentes de alimentación lineales, ver figura 2.73 (a), tienen un diseño relativamente sencillo, y suelen ser de uso en circuitos con baja demanda de corriente y cuyas necesidades de regulación energética o eficiencia no son importantes en el diseño. Las fuentes conmutadas, ver figura 2.73 (b), en contraposición con las lineales, son mucho más eficientes, pueden gestionar más potencia y su tamaño es más reducido, su diseño será más complejo y con más componentes (Diosdado, Zona Maker, 2014).

Características destacables en ambos tipos de fuentes:

- Las lineales tienen un transformador grande.
- Las conmutadas tienen un transformador muy pequeño, gracias a que trabajan a una frecuencia mayor.
- Las lineales no tienen componentes complejos, todos son pasivos o discretos.
- Las conmutadas necesitan un circuito de control activo, para ajustar el arranque, el voltaje de salida, y otras protecciones.
- En las lineales, la corriente va en un solo sentido a través de todas las etapas.
- Las lineales usan transformadores estándar fáciles de conseguir y sustituir, y desperdician mucha energía en el transformador.
- Las conmutadas suelen usar transformadores fabricados a medida para cada modelo. En caso de dañarse el transformador o algún otro componente, son muchísimo más difíciles de reparar.
- Las conmutadas generan mucho ruido, sobre todo en la etapa de conmutación.
- Las lineales apenas generan ruido en comparación (Nieto, 2018).

Las baterías, ver figura 2.73 (c), se presentan en muchas formas y tamaños, desde las miniaturas para reloj o en audífonos, o hasta los bancos de baterías de mayor tamaño que pueden proporcionar energía de reserva a centrales telefónicas o como respaldo de energía en un UPS (del inglés Uninterruptible Power Supply), entre otros. Entregan corriente directa, también son llamadas pilas, la batería es un dispositivo con una o más celdas electroquímicas que convierten la energía química almacenada en corriente eléctrica. Cada celda consta de un electrodo positivo, o ánodo, un electrodo negativo, o cátodo, y un medio llamado electrolito que permite que los iones se muevan entre los electrodos y así poder alimentar un circuito eléctrico (Pérez, 2021).

a) b) c)

Figura 2.73. Fuente lineal, conmutada y baterías.

2.6 Servicios WEB orientados al Internet de las Cosas

Las plataformas enfocadas al IoT ofrecen un conjunto de componentes que permiten configurar y administrar dispositivos conectados a internet. Así una persona puede recopilar datos de forma remota de un sistema, monitorear y administrar todos los dispositivos conectados a internet. Esta plataforma recibe los datos generados en sensores conectados a placas del tipo Arduino, ESP32, Raspberry Pi, entre otras, y pueden ser plataformas de terceros o de creación propia (Diéguez, 2020). Al recoger los datos, la plataforma les da tratamiento para realizar acciones y ofrecer servicios como envío de notificaciones por SMS, correo electrónico, aplicación móvil, supervisión de estos mismos datos en tiempo real o histórico conocido como Big Data (manejo de grandes cantidades de datos e información), monitoreo, gestión, predicción y activación de actuadores (Hernández, 2016).

2.6.1 Esquema general para implementar proyectos del Internet de las Cosas

Como idea general de lo que es un sistema basado en plataformas en la nube, a continuación, se describen los elementos que se deben tomar en cuenta en la creación de estos proyectos:

- El dispositivo conectado: existe una amplia gama de productos, algunos de estos son Arduino, ESP32, Raspberry Pi, BeagleBone entre otros (Hernández, 2016).
- La plataforma en la nube: algunas de estas plataformas orientadas exclusivamente a proyectos del Internet de las Cosas y otras que ofrecen múltiples servicios para todo un ecosistema. Los protocolos de comunicación más utilizados son HTTP, MQTT y CoAP. Además, para ampliar las comunicaciones, existen diferentes redes como LoRa o SigFox. Estas son redes WAN para el Internet de las Cosas y una alternativa a los sistemas tradicionales de comunicación (Hernández, 2016).
- Los dispositivos que consumen la información recabada en la plataforma: por medio de una API, esta es una capa de comunicación estándar para conectar a los datos. Existen diferentes protocolos y estándares. El acceso a dicha API dependerá del software desde donde se realiza la conexión. Si se trata de una aplicación web o framework, se simplifica esta tarea (Hernández, 2016).

Al desarrollar proyectos hay varios aspectos a tomar en cuenta, uno de los más importantes es la plataforma por usar para subir los datos recabados de los dispositivos (Diéguez, 2020). Existen múltiples plataformas en línea disponibles para la creación de soluciones del Internet de

las Cosas, un pequeño inconveniente que existe, son los conocimientos técnicos requeridos para el desarrollo de estas soluciones, esto dependerá de la plataforma que utilicemos y de la calidad del soporte que ofrece (Hernández, 2016). Algunos aspectos adicionales para tomar en cuenta en la selección de plataforma son:

- Si las plataformas pueden virtualizarse, supone una mejora en cuanto a la posibilidad de escalabilidad.
- Empleo de bases de datos.
- Tratamiento de Big Data (Diéguez, 2020).

Estas plataformas ofrecen varios servicios que simplifican el desarrollo de proyectos y un conjunto de herramientas para administrar dispositivos de forma remota (Diéguez, 2020).

En general, una plataforma de IoT es un conjunto de tecnologías de múltiples capas que permite a los usuarios administrar dispositivos conectados. Y que, además, se ocupan de todos los aspectos relacionados con la escalabilidad y la seguridad (Hernández, 2016).

La mayoría de estas plataformas proporcionan API (Application Programming Interface) para admitir placas de desarrollo como Arduino, ESP32, Raspberry Pi, BeagleBone... Este es un aspecto importante al crear proyectos. Tener una API nos permite consultar, modificar y borrar la información desde otros dispositivos de forma fácil y rápida (Hernández, 2016).

Dado el crecimiento del internet actualmente día con día surgen nuevas plataformas para el desarrollo de proyectos del Internet de las Cosas y es complicado hacer un análisis de todas. De manera general se puede realizar una clasificación de algunas de estas plataformas dependiendo del costo y sector al que está orientado (Hernández, 2016).

2.6.2 Plataformas para el Internet de las Cosas

En un primer conjunto, están las plataformas orientadas a startups y empresas pequeñas. Permiten su uso de una manera gratuita, pero con limitaciones en cuanto al número de mensajes enviados y de dispositivos conectados. Enfocadas exclusivamente a dispositivos u objetos conectados (Hernández, 2016).

- Cayenne myDevices: Una de las plataformas más sencillas de usar, en base a un gestor visual, es muy sencillo configurar un dispositivo para que se conecte con Cayenne, dispone de una biblioteca que se puede obtener a través del Gestor de Bibliotecas o descargarla del repositorio oficial, una vez instalada, podemos generar el programa o sketch. La plataforma nos facilita el código que necesitamos instalar, dentro de la misma vienen diferentes dispositivos (sensores y actuadores) preconfigurados. Una de las limitaciones es que sólo podemos enviar 10 valores por segundo (Hernández, 2016). Aunque no es de código abierto, los desarrolladores han puesto Cayenne a disposición de la comunidad de creadores de forma gratuita y lo han apoyado con tutoriales de vídeo, una biblioteca de proyectos y un foro de consulta.
- thinger.io: plataforma de código abierto, para uso alojada en su propio servidor, así como en GitHub para instalarla en una máquina propia. La programación es muy sencilla, dispone de una biblioteca en el repositorio oficial de Arduino e igualmente instalable por medio del Gestor de Bibliotecas o archivo ZIP. Además de poder utilizar esta

plataforma con Arduino, es compatible con ESP8266, ESP32, Raspberry Pi o Intel Edison. Dispone de una consola de administración donde podemos gestionar y geolocalizar nuestros objetos conectados. Además, la documentación es muy extensa y libre (Hernández, 2016), disponible con la creación de una cuenta gratuita, junto a estos beneficios obtenemos acceso a poder agregar hasta 2 dispositivos, y funciones básicas de la plataforma (Dashboard, data buckets, endpoints, access tokens, device management), los planes de pago van desde los €25 hasta los €519 al mes ofreciendo mejoras sustanciales para la implementación de proyectos (THINK BIG LABS S.L, 2019).

- ThingSpeak™: es la apuesta de MathWorks, los creadores de MathLabs, para el Internet de las Cosas. Está enfocado exclusivamente a la construcción de aplicaciones para IoT. Permite almacenar datos, visualizarlos y exponerlos a otras APIs. Tanto la aplicación como la API es de código abierto, podemos descargarla de GitHub y utilizarla en proyectos locales.
  La base de trabajo de esta plataforma son los canales, en ellos se almacenan los datos que le enviamos y se compone de 3 elementos:
  - 8 campos para almacenar datos de cualquier tipo.
  - 3 campos para almacenar la ubicación, latitud, longitud y elevación, requiere un componente que dote de esta información.
  - 1 campo para almacenar el estado.

  Cada uno de estos campos puede ser actualizado cada 15 segundos. Dentro del código hay bibliotecas para múltiples lenguajes de programación. Podemos encontrar esta biblioteca para utilizarla en proyectos con Arduino, dentro del entorno de desarrollo oficial. Tiene una documentación muy extensa con ejemplos y es totalmente gratuita (Hernández, 2016), también se puede hacer uso de la plataforma de forma comercial mejorando la capacidad de conexión al servidor por la cantidad de $675 dólares al año en la versión estándar de uso, ofreciendo también paquetes de uso para académicos, estudiantes y el hogar por $250, $75, $95 dólares al año respectivamente (The MathWorks, Inc., 2018).

Como segundo conjunto se engloba a plataformas que también ofrecen servicios gratuitos o versiones de prueba. Están más centradas en ofrecer servicios globales, ya no es solo recibir datos, estas plataformas nos permiten almacenar webs, API para móviles, bases de datos, entre otras funciones adicionales (Hernández, 2016).

- Ubidots: la plataforma es compatible con varios dispositivos, como Arduino, Raspberry, Particle, Espressif, Onion y muchos otros más. Ofrece diferentes tipos de servicios, que van desde la conectividad del dispositivo hasta la visualización de datos:
  - SDK / API: proporciona una SDK para diferentes dispositivos y así simplificar el proceso de integración entre los mismo y la plataforma, además, tiene varias API que se pueden invocar para interactuar con esta.
  - Soporta protocolos como MQTT y HTTP, entre otros.
  - Posibilidad de aplicar fórmulas matemáticas en los datos, almacenamiento y visualización de estos.

- Las interfaces gráficas que se pueden crear van desde posicionamiento por GPS sobre un mapa hasta el diseño personalizados con widgets específicos (Diéguez, 2020).

Ubidots es una plataforma de pago que cuenta con la opción de probarlo gratis durante un tiempo en la creación de una cuenta para estudiantes y académicos, limitando el uso para solo 3 dispositivos, más de 200 bibliotecas y tutoriales, dashboard en tiempo real con más de 30 widgets y herramientas, disparo de eventos y notificaciones por correo electrónico, mensaje de texto, telegram, llama de voz o webhooks, sus planes comerciales van desde los $49 hasta $499 dólares por mes y con precio negociable para empresas (Ubidots (IoT Services Inc.), 2022).

- Adafruit IO: a través de la plataforma podremos agregar, almacenar y visualizar datos en tiempo real en la nube desde dispositivos del Internet de las Cosas. Nos permitirá interactuar con estos dispositivos desde paneles de control (dashboards) basados en la web. Es compatible con dispositivos como Arduino, Raspberry Pi y ESP entre otros. Puede trabajar con APIs REST y MQTT. La versión gratuita permite el envío de 30 datos por minuto, almacenar los datos por 30 días, disparos de alarma (triggers) cada 15 minutos, mantener hasta 5 paneles con 10 campos por panel y conexión de 2 dispositivos, así como un plan de pago por $10 dólares al mes o $99 dólares de forma anual donde se pueden enviar hasta 60 datos por minuto, almacenar los datos por 60 días, disparos de alarma cada 5 segundos y las funcionalidades de forma ilimitada (Adafruit Industries, LLC, 2017).
- thethings.io: esta plataforma ofrece un abanico de servicios extenso, especializado en aplicaciones de agricultura, logística, industria 4.0, casas y ciudades inteligentes. Tiene un periodo de prueba de 15 días y después el costo del paquete más básico son €25 por mes (Hernández, 2016) pudiendo escalar este paquete hasta los €349 al mes, así como planes personalizados para empresas (DIPRO TECHNOLOGIES S.L., 2020).
- IFTTT: esta plataforma permite interconectar servicios en la nube. IFTTT son las siglas en inglés de If This Then That. Soporta múltiples servicios, casi todos los más conocidos y muchos más (Hernández, 2016).

  Uno de estos servicios son las Webhooks, con el que se pueden integrar nuestros proyectos para enviar y recibir datos o eventos. IFTTT no tiene un API, utiliza llamadas HTTP para comunicarse. Para la recepción de eventos se requiere implementar un servidor web sencillo o bien implementar las Webhooks en alguna plataforma adicional del Internet de las Cosas en nuestro dispositivo, así como configurar IFTTT para hacer la llamada que corresponda, adjuntando los datos del evento de origen. Para recibir los eventos de IFTTT nuestro dispositivo necesita estar conectado a internet.

  Algunos ejemplos de integración son: encender un LED en nuestro dispositivo conectado cuando nos mencionen en Twitter, enviar una notificación a nuestro celular si la temperatura sube demasiado o registrar los valores de nuestra estación meteorológica a una hoja de cálculo en una cuenta de Google Drive (Hernández, 2016). IFTTT en la versión de una cuenta gratuita nos permite realizar 5 asociaciones (Applets) solucionando esta limitación en la versión Pro con un costo de $50 pesos mexicanos por mes con 20 asociaciones, $100 por mes para la versión Pro+ con asociaciones ilimitadas, múltiples cuentas, y más herramientas de desarrollo (IFTTT Inc., 2017).

Otro conjunto son las plataformas que ofrecen las grandes empresas y corporaciones como Google, Amazon, Microsoft, IBM, etcétera… Están orientadas sobre todo al sector industrial y a grandes proyectos del Internet de las Cosas, donde se interconectan cientos o miles de dispositivos (Hernández, 2016).

- Amazon Web Services (AWS) IoT: una de las más grandes plataformas en la nube. Ofrece muchos servicios relacionados con la comunicación entre servicios y el análisis de datos. No es una plataforma gratuita, el precio depende de los servicios que se desean utilizar, en promedio el costo es de $5 dólares por millón de mensajes (Hernández, 2016).
  AWS ofrece un amplio conjunto de servicios:
  - Amazon FreeRTOS: sistema operativo para microcontroladores que proporciona algunos servicios, como conectividad, seguridad y programación fácil.
  - AWS Greengrass: software que permite ejecutar cómputos locales en dispositivos periféricos.
  - AWS Analytics: permite ejecutar análisis sofisticados en volúmenes de datos masivos, te permite automatizar cada uno de los pasos que se requieren para analizar esta cantidad de información proveniente de los dispositivos y sus sensores. Esta solución tiene también la funcionalidad de crear graficas para diseñar una interfaz visual con solo arrastrar y soltar para conectar y coordinar la interacción entre los dispositivos y los servicios web, para construir aplicaciones basadas en el Internet de las Cosas rápidamente.
  - Administración de dispositivos: simplifica el proceso de administración de dispositivos, especialmente cuando hay miles de dispositivos conectados.
  - AWS Core: es el núcleo de esta plataforma y permite que los dispositivos conectados interactúen con los servicios en la nube de forma fácil y segura (Diéguez, 2020).
- Google Cloud Platform IoT: este servicio como los anteriores se centra en los objetos conectados y permite escalar el sistema desde las primeras fases de desarrollo donde se ofrece un tráfico de datos bajo, hasta las fases donde el volumen de información es grande. A si también un kit de placa Seeed Studio BeagleBone Green Wireless (BBGW) adaptada específicamente para Google Cloud (Hernández, 2016). Permite un periodo gratuito y te regalan $300 dólares en crédito para gastar durante 90 días en sus servicios (Google LLC, 2022). Hay que introducir una tarjeta de crédito válida, garantizan que no cobrarán nada una vez concluido el periodo de prueba. La facturación depende de los servicios utilizados (Hernández, 2016).
- IBM Watson™ IoT: en la plataforma de IBM encontramos una amplia gama de servicios, específicamente para el Internet de las Cosas, tienen catalogados los servicios por áreas específicas como: automóviles, electrónica y seguridad (International Business Machines Corp., 2022). Permite un periodo de prueba de 30 días e incluye un crédito de $200 dólares para gastar en sus servicios, al finalizar este periodo se paga por los servicios consumidos (International Business Machines Corp., 2022).
- Azure IoT Suite: el Internet de las cosas de Azure es una colección de servicios en la nube administrados por Microsoft que conectan, supervisan y controlan miles de millones de dispositivos activos. Esta es una plataforma profesional con varios servicios.

Admite la comunicación bidireccional entre los dispositivos conectados y la plataforma utilizando protocolos estándar para estas soluciones en la nube, es compatible con la autenticación del dispositivo para abordar todos los aspectos de seguridad (Diéguez, 2020).

Permite crear una cuenta gratuita donde regalan hasta $200 dólares para utilizar con sus servicios por 30 días. Una vez se finaliza el periodo de prueba, se debe pagar por los servicios consumidos, ofrece tres niveles estándar, que van desde los $0.08 y 400 mensajes en nivel estándar 0, hasta $0.70 dólares al mes y 30000 mensajes en nivel estándar 2 respectivamente por dispositivo conectado permitiendo conectar 2 dispositivos de forma gratuita en cualquier nivel, así como costo adicional por uso encima del límite de mensajes de $0.07 hasta $0.015 de dólar por 1000 mensajes adicionales (Microsoft S. de R.L de C.V., 2017).

Como último conjunto se engloban plataformas de código abierto. Son todas aquellas que nos dan acceso al código sin restricciones. Pueden ser descargadas e instaladas en máquinas de forma local (Hernández, 2016).

- Node-RED: plataforma de código abierto basada en NodeJS. Permite crear proyectos del Internet de las Cosas de una forma visual (Hernández, 2016).
- openHAB: software de automatización de código abierto para el hogar basado en Java (Hernández, 2016).
- Kaa: solución completa basada en código abierto que te permite crear aplicaciones del Internet de las Cosas tanto en servidor como en cliente. Se trata de aplicación multiplataforma que está disponible para múltiples sistemas operativos y dispositivos (Hernández, 2016).
- DeviceHive: es una plataforma para proyectos que nos permite el análisis de datos en tiempo real. Se encuentra preparada para ser ejecutada en grandes plataformas en la nube como Azure, AWS o Google Cloud (Hernández, 2016).

2.6.3 Servicios WEB con integración del microcontrolador ESP32 y asistentes virtuales

- Blynk 2.0: la plataforma ofrece un conjunto completo de software para la creación de prototipos, implementación y administración de forma remota de dispositivos del Internet de las Cosas a cualquier escala (Blynk, Inc., 2021). Soporta múltiples dispositivos como: Arduino, ESP, Particle, Raspberry Pi, entre otras, habilitando la conexión de dicho hardware a la nube y permitiendo crear aplicaciones móviles sin código adicional (Blynk, Inc., 2021). La plataforma ofrece la oportunidad de realizar proyectos con una cuenta gratuita que permite la implementación de bibliotecas propias, como máximo 2 dispositivos y hasta 5 usuarios, para obtener características adicionales como por ejemplo la integración de asistentes virtuales, se puede contratar alguno de los 3 planes adicionales que van desde los $6.99 hasta los $599 dólares por mes (Blynk, Inc., 2016).
- Arduino Cloud: plataforma totalmente compatible con cualquier placa de Arduino que tenga conectividad, además de placas de desarrollo de otros fabricantes como la ESP de Espressif. Para configurar un dispositivo, solo se requiere de la creación de una cuenta, la plataforma nos guía y ofrece las herramientas necesarias para instalar en el equipo de

cómputo el Arduino Editor y así poder crear nuestro sketch desde la misma web y mandar la información a la infraestructura que tiene Arduino en la nube, el propio configurador da el código que se tiene que cargar en la placa, dependiendo de la que se allá elegido (Arduino S.r.l., 2022).

Utiliza el protocolo MQTT para transmitir los datos, con la infraestructura en la nube de Amazon. Cuanta con aplicación móvil para acceder en cualquier momento al dashboard y así controlar y monitorizar los proyectos, cuenta con la posibilidad de agregar una habilidad para el asistente de Amazon Alexa a través de algunos de sus planes de pago (Arduino S.r.l., 2022). La plataforma cuenta con limitación en cuanto a su uso ya que en el plan gratuito solo podremos conectar 2 objetos, dispondremos de solo 100 Mb para almacenamiento de los sketches que creemos, como máximo 1 día de histórico de datos y solo podremos compilar los códigos 25 veces por día. Todas estas limitaciones pueden ser solventadas al contratar alguno de los planes de pago que ofrece la plataforma que van desde los $1.99 hasta $19.99 dólares al mes (Arduino S.r.l., 2022).

- Sinric Pro: la plataforma permite conectar placas de desarrollo como: ESP8266, ESP32, RaspberryPi o Arduino y una infinidad más de dispositivos y sensores a Amazon Alexa o Google Home con un complemento confiable, sin emulaciones que puedan propiciar errores. Es compatible con todos los tipos de dispositivos Amazon, así como Google Home y ofrece bibliotecas para los lenguajes de programación como Python, C++, NodeJS, con ejemplos de ejecución. Para usuarios avanzados, permite crear dispositivos personalizados, configurar salas y rutinas, y monitorear los dispositivos a través de REST API (Amazon.com, Inc., 2019). Ofrece servicio de solución PaaS para aplicaciones móviles, permite la conexión de dispositivos a través de Bluetooth Low Energy (BLE), SmartAP y mediante Portal cautivo (WebUI), gestión de usuarios, dispositivos y actualizaciones por método inalámbrico (OTA). En cuestión de seguridad garantiza una comunicación encriptada SSL entre servidor, nube, aplicación y dispositivo (con hardware compatible) (© Sinric 2019-2022, 2022).

  Sinric Pro ofrece una prueba gratuita donde se pueden conectar hasta 3 dispositivos sin ninguna limitación y si se desea se puede agregar dispositivos adicionales pagando $3 dólares al año por dispositivo adicional, así como el servicio PaaS por $2,558 dólares por año permitiendo conectar de manera virtual hasta 10000 dispositivos al servidor (© Sinric 2019-2022, 2018).
- RainMaker: ESP RainMaker es una plataforma para el desarrollo de aplicaciones con el Internet de las Cosas creada por Espressif, que es totalmente compatible con las placas de desarrollo ESP32. Es muy fácil crear un proyecto en esta plataforma ya que no se requiere programación adicional para la aplicación móvil, esta es auto adaptativa, por tanto, solo es necesario escribir el código y la aplicación se generará automáticamente. Cuenta con integración gratuita de Alexa y el asistente de Google, uso compartido de miembros, aprovisionamiento de wifi a través del código QR y muchas funciones más (© 2013-2022 Microcontrollerslab.com, 2022). El uso de la plataforma es gratuito pero limitado, con opción de escalabilidad por medio de los servicios de Amazon (AWS) (Espressif Systems (Shanghai) Co., Ltd., 2021).

2.6.4 Plataforma Blynk 2.0

Esta nueva plataforma se presenta como evolución a su antecesora Blynk, como actualización ofrece un conjunto completo de software para la creación de prototipos, implementación y administración de forma remota de dispositivos del Internet de las Cosas permitiendo la escalabilidad desde proyectos personales hasta comerciales (Blynk, Inc., 2021). Soporta múltiples dispositivos como: Arduino (cualquier modelo), ESP8266, ESP32, Particle (cualquier modelo), Raspberry Pi (cualquier modelo), Lantronix Fox3, entre otros, habilitando la conexión de dicho hardware a la nube por medio de conectividad wifi, ethernet, datos móviles (GSM, 2G, 3G, 4G, LTE), serial USB vía PC o bluetooth (Beta, descontinuado a partir de octubre de 2022) (Blynk, Inc., 2021), permite crear aplicaciones para iOS, Android y Web sin código adicional, permite así simplificar el conectar casi cualquier hardware electrónico a internet, comenzar a recopilar datos de dispositivos, monitorearlos y controlarlos de forma remota desde cualquier parte del mundo. Los datos de los dispositivos se pueden almacenar, agregar y visualizar en aplicaciones móviles y web fáciles de crear (Blynk, Inc., 2017).

La plataforma ofrece la oportunidad de realizar proyectos con una cuenta gratuita que permite la implementación de las bibliotecas Blynk.Edgent, Blynk.Inject, Blynk.Air, 2 dispositivos y 5 usuarios, así como encriptación en la comunicación con la nube, para obtener características adicionales se puede contratar alguno de los 3 planes adicionales que van desde los $6.99 hasta los $599 dólares por mes (Blynk, Inc., 2016).

Características que ofrece la plataforma en su versión 2.0:

- Blynk.App: creador de aplicaciones móviles único y sin código, ver figura 2.74, para Internet de las Cosas (iOS versión 9 y superior, así como Android 4.2 y superior, incluido de forma gratuita solo con algunos Widgets e independiente de la interfaz web)

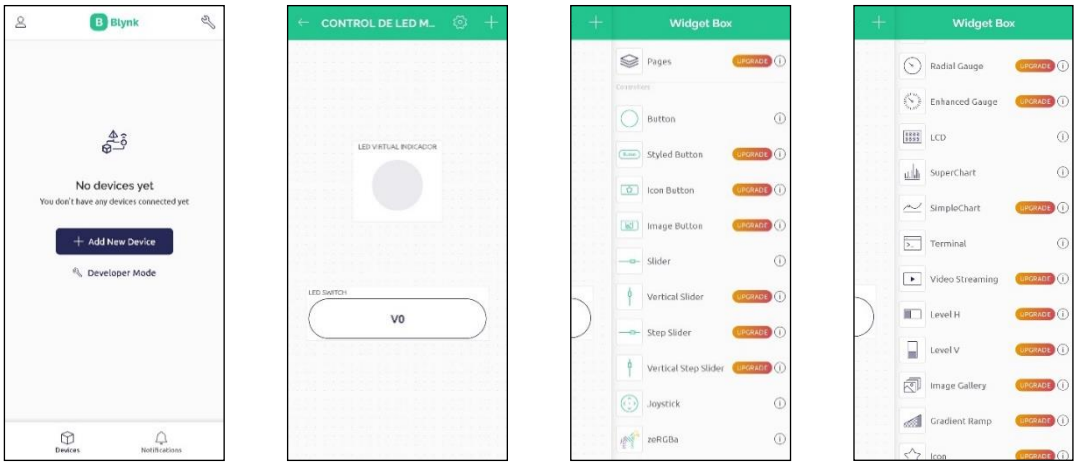


Figura 2.74. Blynk App.

- Blynk.360: interfaz web de usuario limpia y simple donde se puede agregar y administrar dispositivos, controlar el acceso a su servicio de Internet de las Cosas y monitorear su uso, ver figura 2.75. Se puede acceder a los datos del o los dispositivos con todos los principales navegadores web (excepto Internet Explorer descontinuado). Cada

dispositivo tiene un tablero para visualizar datos y controles de sensores por medio de históricos o en tiempo real. Para configurar el tablero (dashboard) es similar a la aplicación Blynk: arrastre y suelte de widget, ajustar tamaño, configurar entradas de datos y listo.
Adicional, hay una línea de tiempo donde puede registrar eventos provenientes de los dispositivos. Se puede configurar qué tipo de evento es, a quién se debe notificar y cómo. Cada dispositivo ahora lleva información adicional (metadatos), es información más o menos estática sobre el dispositivo (ubicación, número de serie, etc.), pueden ser convincentes para el análisis de datos. Esta interfaz es independiente de la aplicación móvil, y es lo suficientemente potente para soportar casos de uso comercial (incluido de forma gratuita)

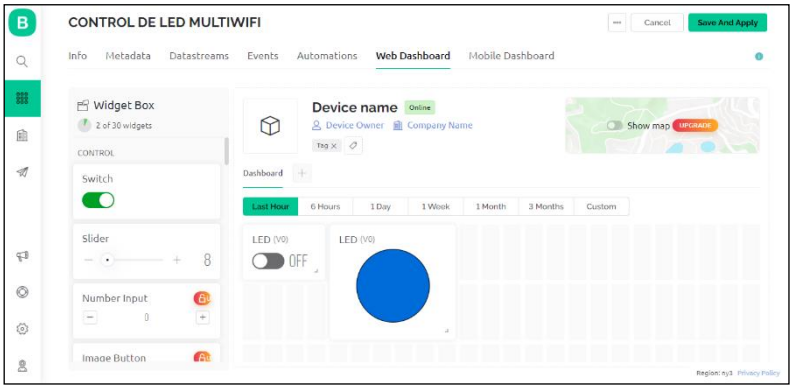


Figura 2.75. Blynk 360.

- Templates: nuevo proceso de creación de plantillas para la administración de dispositivos con la posibilidad para escalar (incluido de forma gratuita), que se puede configurar en las plantillas (template):
    - Metadatos: datos adicionales sobre el dispositivo, admite configuraciones adicionales
    - Flujos de datos: permite definir los datos que entran y salen del o los dispositivos
    - Eventos: una lista de alertas y notificaciones importantes
    - Panel web: un conjunto de widgets en la aplicación web
    - Panel móvil: un conjunto de widgets en la aplicación móvil (redirecciona a la descarga de la aplicación móvil en la tienda de aplicaciones correspondiente)

Figura 2.76. Templates Blynk.

- Blynk.Inject: asistente de conexión wifi que se encuentra integrado en la aplicación móvil Blynk de forma predeterminada. No hay necesidad de usar páginas web separadas o bibliotecas de terceros para ingresar credenciales wifi.
  Además del aprovisionamiento de dispositivos, permite la creación de perfiles para dispositivos, agregar metadatos al dispositivo después de que se haya conectado a Wifi. De manera comercial a sus clientes puede configurar qué campos se agregan, cuáles son obligatorios y cuáles deben incluirse en el perfil del dispositivo todo a través del teléfono móvil.
  Se puede configurar múltiples usuarios y como obtienen acceso a los datos, estableciendo roles y configurando permisos. (incluido de forma gratuita)
- Blynk.Edgent: bibliotecas de hardware de código abierto para conectar cualquier dispositivo a Blynk.Cloud mediante conexión Wifi, Ethernet o celular (2G, 3G, LTE, etcétera, incluido de forma gratuita).
- Blynk.Air: actualizaciones de firmware inalámbricas en tiempo real. Actualmente con soporte: ESP8266, ESP32, ESP8266, MKR1000, MKR Wifi 1010, TI CC3220SF (incluido de forma gratuita)
- Todo tipo de Automatizaciones (mejoradas con respecto a Blynk, solo se incluyen algunas de forma gratuita)
- Asistentes de voz como Amazon Alexa o asistente de Google y muchas funcionalidades más (funcionalidades adicionales son incluidas en planes con costo) (Blynk, Inc., 2021)

CAPÍTULO 3. DESARROLLO EXPERIMENTAL, PRUEBAS Y RESULTADOS

En este apartado se presenta en primera instancia, el proceso para trabajar con la plataforma Blynk y la tarjeta ESP32. Posteriormente se realizan ocho propuestas de sistemas domóticos, dando una breve introducción; donde, entre otras cosas, se menciona su propósito, diagrama de bloques, de flujo y esquemático, cada uno con su respectiva explicación. También se detalla el proceso de creación de las aplicaciones tanto web como móvil, se realizan las pruebas pertinentes y se describen sus resultados. Todo lo mencionado se aplica a cada propuesta.

3.1 Habilitar una tarjeta con ESP32 por medio del IDE Arduino en Blynk

Este primer experimento tiene como objetivo demostrar cómo lograr la interconexión de un sistema que cuente con un microcontrolador hacia una conexión a internet, por medio del denominado Internet de las Cosas. Para ello, se explica cómo conseguirla utilizando redes wifi, incluyendo un método para lograr la comunicación con cualquier acces point (punto de acceso) sin la necesidad de reprogramar al sistema al pasar de uno a otro.

Para todas las propuestas se utilizará una tarjeta de desarrollo que cuenta con un microcontrolador ESP32, la cual permite la conexión a internet mediante su módulo wifi integrado. La programación se realiza con el IDE Arduino, preparado como se ha descrito en el capítulo 2.2.4 y 2.2.5.

3.1.1 Registro en la plataforma Blynk 2.0

Para la creación de una cuenta se debe acceder al sitio de Blynk[1], y hacer clic en el botón “START FREE”, (si ya se tiene una cuenta, para acceder pulsar sobre el título “LOG IN” en la página principal) ver figura 3.1, una vez se ha hecho clic se redirecciona a la página de registro, donde se debe colocar una dirección de correo electrónico válida, aceptar los términos y condiciones y avanzar al pulsar “Sign Up”, posterior se enviará un correo electrónico que al consultar proporciona información para continuar con el registro.

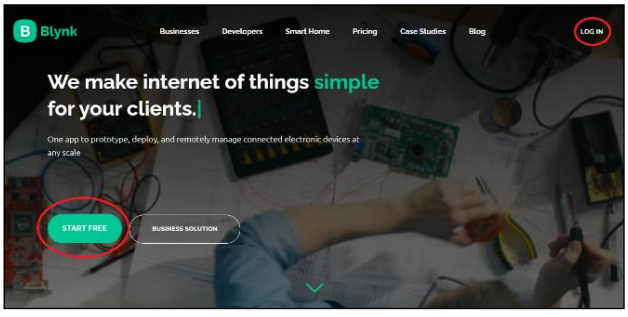


Figura 3.1. Sitio web Blynk.

[1] https://blynk.io/

El correo electrónico da la bienvenida a la plataforma y para habilitar la cuenta requiere la creación de una contraseña, al pulsar el botón redirige nuevamente a la plataforma, creada la contraseña solicitará solo como dato personal el nombre de quien administrará la cuenta, para finalizar el registro dar clic en el botón “Done”.

La plataforma introduce con un breve tour de las funciones más destacadas y noticias sobre nuevas funcionalidades agregadas, así como un tutorial para conectar por primera vez alguno de los dispositivos que son admitidos.

Tanto el tour como el tutorial se pueden omitir, en este documento se describe de forma detallada el proceso de habilitación de una tarjeta con el microcontrolador ESP32 a la plataforma. Omitida la introducción se muestra la página principal de la plataforma desde la cual se pude crear y administrar dispositivos, plantillas, ubicaciones, tableros de control, así como obtener soporte, ver figura 3.2.

El procedimiento para la creación de una cuenta en la plataforma es análogo a través de un dispositivo móvil, esta cuenta es única y puede usarse por igual en ambas aplicaciones web y móvil, la creación de esta cuenta no condiciona el uso exclusivo de esta en alguna plataforma, se hace hincapié porque se pueden diseñar de forma independiente tablero de control web y móvil y estos no dependen entre sí de ellos, en este documento algunas aplicaciones o funciones serán desarrolladas e implementadas en exclusivo para alguna plataforma en particular.

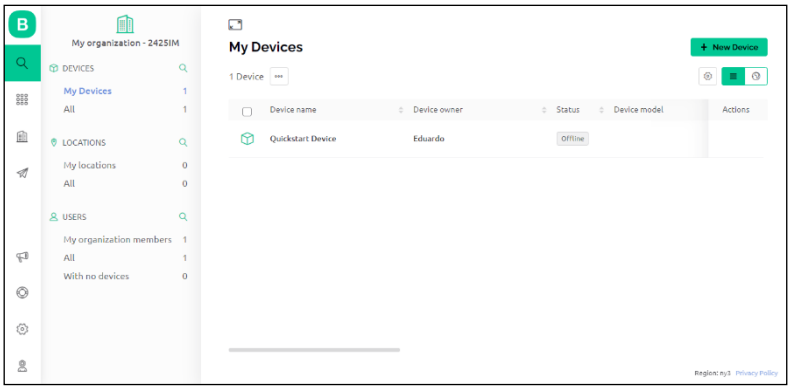


Figura 3.2. Página principal del sitio web Blynk.

3.1.2 Configuración web, móvil y biblioteca de plataforma Blynk en IDE Arduino

Ya instalado el IDE Arduino y se ha cargado la o las respectivas bibliotecas, en todos los casos en este documento se ha utilizado como base el código que provee la biblioteca para la plataforma Blynk para la asociación y conexión de la tarjeta de desarrollo con la misma. El ejemplo se obtiene por medio de la barra de menú en Archivo ► Ejemplos ► Blynk ► Blynk.Edgent ► Edgen_ESP32.

Al seleccionar se abre una nueva ventana que despliega dentro del entorno múltiples pestañas con las cabeceras (archivos “.h”) que conforman la estructura de la biblioteca, estas se encuentran programadas en un lenguaje propio del entorno basado en C y permiten modificar algunos aspectos del funcionamiento de la biblioteca en general.

El programa se escribe en la pestaña con la extensión ".ino", ver figura 3.3, el nombre de la pestaña puede cambiar una vez se guarda el sketch con la función Guardar como…, en el menú Archivo.

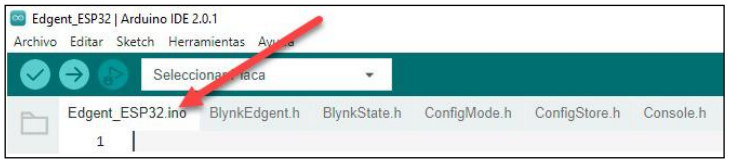


Figura 3.3. Pestañas en interfaz de IDE Arduino.

En la estructura que presenta el código se puede identificar varias funciones de uso común (si ya se posee algo de conocimiento en programación con el IDE Arduino) en el programa las funciones "define" que inician el código se declaran las palabras reservadas que almacenaran las credenciales para una conexión segura entre la tarjeta de desarrollo y el servidor en línea de Blynk.

Estas funciones permiten nombrar las constantes como ID de plantilla y nombre del dispositivo, respectivamente.

```
2       #define BLYNK_TEMPLATE_ID "TMPLxxxxxx"
3       #define BLYNK_DEVICE_NAME "CONTROL DE LED"
```

3.1.2.1 Creación de plantillas en Blynk

Para obtener las credenciales de identificación primero se deberá crear en la plataforma web de Blynk la plantilla que se asignara a la o las tarjetas y en la cual se indican las variables que serán monitoreadas por el servidor y los datos que este recibirá de la tarjeta de desarrollo. Esta plantilla se crea y administra en el apartado que en la plataforma recibe el nombre de "Templates" ver figura 3.4.

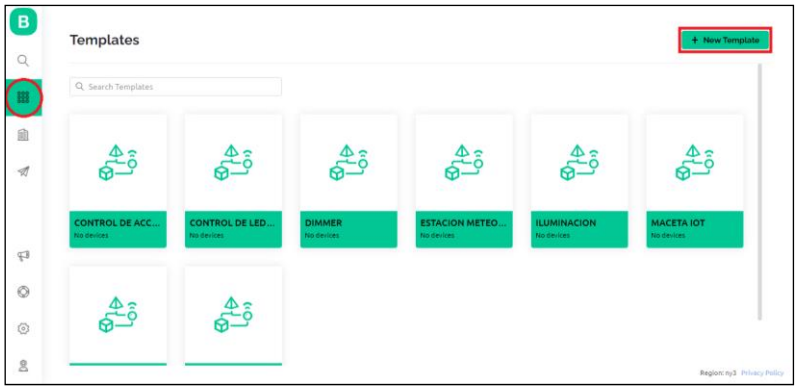


Figura 3.4. Portal para creación de plantillas.

Al hacer clic en el botón "+ New Template" se abre una ventana de configuración básica para la plantilla donde se puede asignar el nombre, el tipo de tarjeta de desarrollo que se va a asociar, tipo de conexión, y una descripción de esta para facilitar la identificación, ver figura 3.5.

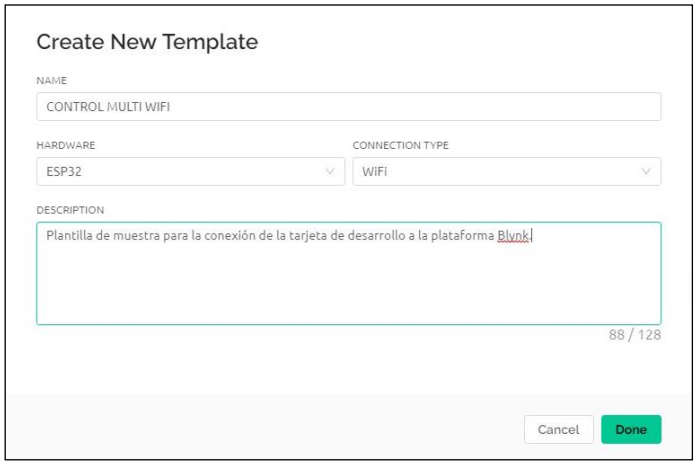


Figura 3.5. Creación de nueva plantilla.

Pulsando el botón "Done" se crea la plantilla y automáticamente se despliegan más configuraciones, así como herramientas adicionales, ver figura 3.6. En esta primer pestaña se muestra la información que deberá ser ingresada en las líneas (2 y 3 del programa principal) del código de programación de las tarjetas que deseemos conectar a la plantilla, estas líneas permiten copiado rápido desde la misma web, copiar y pegar remplazando las líneas anteriores del código del programa, (si se programan varias tarjetas con las mismas credenciales de la plantilla, en la versión gratuita de la plataforma se permite agregar un máximo de 2 dispositivos que pueden ser en la misma plantilla o en una diferente, para proyectos de mayor escala se recomienda revisar los planes actualizados de negocio que oferta la plataforma).

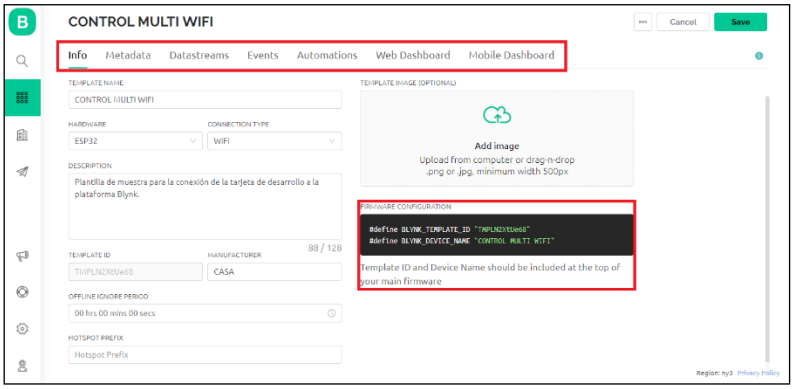


Figura 3.6. Configuraciones adicionales para la creación de plantilla.

```
5       #define BLYNK_FIRMWARE_VERSION "0.1.0"
6
7       #define BLYNK_PRINT Serial
8       //#define BLYNK_DEBUG
9
10      #define APP_DEBUG
11
12      // Uncomment your board, or configure a custom board in Settings.h
13      //#define USE_WROVER_BOARD
```

```
14      //#define USE_TTGO_T7
15      //#define USE_ESP32C3_DEV_MODULE
16      //#define USE_ESP32S2_DEV_KIT
```

En el programa también se define la palabra reservada que declara la versión de “firmware” con el que se identificará la tarjeta de desarrollo en la plataforma, así también si se habilita la comunicación de datos de la plataforma por medio de comunicación serial o la depuración en la aplicación móvil por medio del modo desarrollador.

Para los modelos de tarjetas desarrollo indicados en el código de programa, se puede remover el comentario de la línea de código que define el uso de dicho modelo, al declararlo se personalizará la cabecera “Settings.h” donde, en cada modelo se indica de forma correcta todas aquellas prestaciones que ofrece cada una de estas tarjetas. Si todos los modelos quedan comentados el programa al ejecutar la cabecera “Settings.h” cargará las configuraciones para tarjetas “personalizadas”. Estas configuraciones pueden ser modificadas dentro de la misma cabecera antes de la carga del programa a la tarjeta.

3.1.2.2 Configuración de biblioteca Blynk para uso de tarjetas de desarrollo “personalizadas”

```
45      #else
46
47        #warning "Custom board configuration is used"
48
49        #define BOARD_BUTTON_PIN            15
50        #define BOARD_BUTTON_ACTIVE_LOW     true
51
52        #define BOARD_LED_PIN               2
53        //#define BOARD_LED_PIN_R           15
54        //#define BOARD_LED_PIN_G           12
55        //#define BOARD_LED_PIN_B           13
56        //#define BOARD_LED_PIN_WS2812      4
57        #define BOARD_LED_INVERSE           false
58        #define BOARD_LED_BRIGHTNESS        255
59
60      #endif
```

Explorando la cabecera a través de las pestañas del entorno de programación, ver figura 3.7, para editar las configuraciones que definen las prestaciones de la o las tarjetas “personalizadas”, se realizaron las siguientes modificaciones con el fin de aprovechar indicadores visuales y diferentes pines para actuadores externos a los que integra la tarjeta NodeMCU-32S. Por medio de la figura 2.5, se observa que en el pin correspondiente al GPIO2 se encuentra uno de los LEDS que integra la tarjeta de desarrollo y que será aprovechado para visualizar el estado de la conexión wifi. En la línea del código `#define BOARD_LED_PIN` se modifica el parámetro por le GPIO y así utilizar el indicador. Si la tarjeta cuenta con LED RGB integrado, se puede quitar el comentario de las líneas de código y modificar los GPIO de salida para habilitar el uso de este tipo de LED. En la línea `#define BOARD_LED_INVERSE` se puede definir el modo de conexión a alimentación del LED integrado, y en la línea `#define BOARD_LED_BRIGHTNESS` la intensidad máxima de brillo que emitirá siendo 255 el valor máximo. En las líneas `#define`

`BOARD_BUTTON_PIN` y `#define BOARD_BUTTON_ACTIVE_LOW` permiten seleccionar el GPIO para agregar o utilizar un botón de reinicio, y el tipo de disparo que tendrá, que puede ser **`true`** o **`false`** para activar con un nivel lógico bajo o alto, por defecto en la NodeMCU-32S se encuentra predefinido el GPIO 0, este cuenta con la conexión al push button de programación integrado en la tarjeta, para los experimentos en este documento se ha remplazado este valor, adicionando un push button externo y evitar confusiones durante las pruebas de los proyectos.

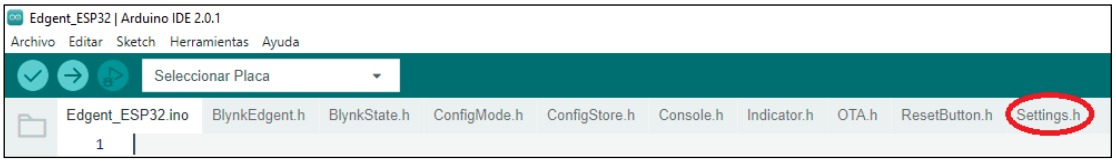


Figura 3.7. Pestañas correspondientes a las cabeceras del programa Edgent Blynk para ESP32.

3.1.2.3 Creación de canales para el flujo de datos “Datastreams”

En el programa principal, para la lectura de datos en la plataforma se deben crear los canales respectivos para el flujo de datos y su interpretación en esta. En la plataforma en las pestañas de la parte superior se pude modificar, administrar y agregar información adicional sobre los parámetros de las tarjetas conectadas, así como los canales para el flujo de datos, eventos, automatizaciones, creación de tablero de control web y móvil (el tablero móvil no es accesible desde la web, la configuración y administración se realiza desde la aplicación móvil a través del modo desarrollador).

En la pestaña “Datastreams” se administra el flujo de datos de sensores o actuadores a monitorizar o controlar por medio de la plataforma, estos datos serán recibidos o enviados por medio de un pin virtual el cual se crea y configura con el botón “+ New Datastream”, ver figura 3.8.

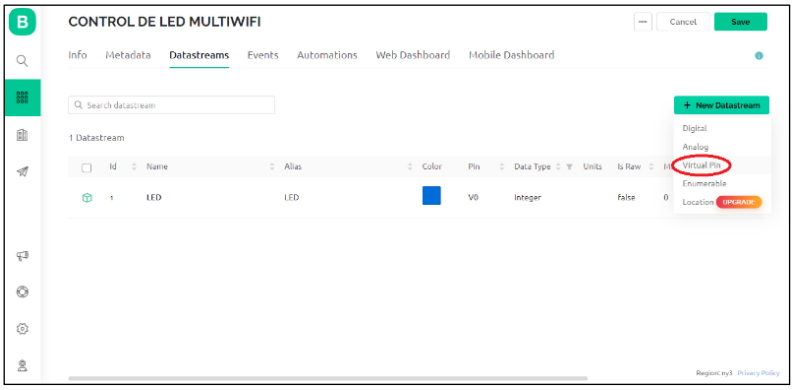


Figura 3.8. Creación de pin virtual en plataforma web Blynk

En la edición de pin virtual es posible agregar o editar un nombre, un alias, color que identifique el flujo de datos, el canal por donde se establecerá el dato (en el ejemplo, el flujo enviará o recibirá datos a través del pin V0, se dispone de hasta 255 pines), el tipo de dato que puede ser integer (entero), double (doble) o string (cadena de caracteres), el tipo de unidad que maneja el dato (funcional en la creación del tablero de control ya que agrega dicha unidad en los widgets

de la interfaz facilitando la lectura), límite de valores recibidos mínimo y máximo, así como el valor por defecto que deberá recibir o enviar el canal, ver figura 3.9.

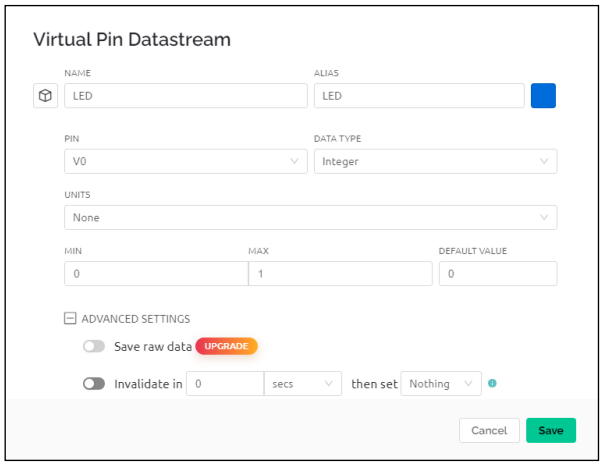


Figura 3.9. Configuración de pin virtual.

En los ajustes adicionales se puede condicionar acciones al flujo de datos como guardar los datos en crudo recibidos (función de paga), invalidarlos en cierto periodo de tiempo, sincronizar el ultimo valor recibido con el servidor, mostrar etiquetas y reportes, ver figura 3.10.

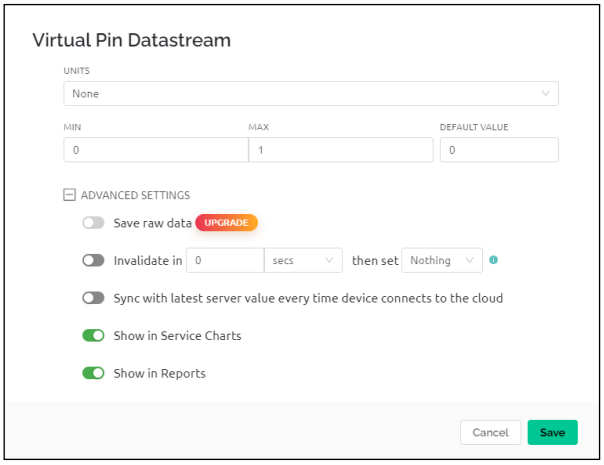


Figura 3.10. Ajustes adicionales para el pin virtual.

Pulsando el botón “Save” se crea el flujo de datos, en el código del programa que se cargará a la tarjeta se deberá crear e indicar el canal que administrara la conexión con el sensor o actuador como se muestra a continuación.

```
21      BLYNK_WRITE(V0) {
22        digitalWrite(LEDPIN, param.asInt());
23      }
```

Esta no es la única forma para interactuar con variables en un canal, también es posible a través de la función `Blynk.virtualWrite(vPin, valor);` donde "vPin" es el canal asignado y "valor" el dato a escribir en el pin virtual. De esta manera solo se admite enviar datos al servidor. Como diferencia, la primer función (`BLYNK_WRITE(vPin) {}`) permite enviar y recibir datos del servidor.

3.1.2.4 Código de ejemplo (funcionamiento básico de lenguaje IDE Arduino)

```
18      #include "BlynkEdgent.h"
19      #define LEDPIN 13
```

En el programa debe incluir la cabecera `"BlynkEdgent.h"` ya que en esta se invoca todas las demás cabeceras que forman parte del código y proporcionan configuraciones de la biblioteca, también se define la palabra reservada `LEDPIN` que indica el GPIO 13 como pin para un indicador LED externo.

```
25      void setup() {
26        Serial.begin(115200);
27        pinMode(LEDPIN, OUTPUT);
28        delay(100);
29        BlynkEdgent.begin();
30      }
```

En la función del programa "`setup()`" se declaran todas aquellas configuraciones en la tarjeta de desarrollo de manera local, es decir, se establece cuáles serán los pines de entrada o salida (GPIO 13 como salida), parámetros como velocidad (transferencia USART a 115200 baudios, puede ser cualquier otro valor de velocidad soportado por la tarjeta, la NodeMCU-32S puede enviar datos a esta velocidad sin inconvenientes) frecuencias, y demás acuerdos necesarios para establecer todas las compatibilidades con el hardware. Estas instrucciones se ejecutan una única vez, en el momento de encender (o reiniciar) la tarjeta. Es en este apartado donde se inicializa la biblioteca "BlynkEdgent".

```
32      void loop() {
33        BlynkEdgent.run();
34      }
```

Las instrucciones escritas dentro de la función "`loop()`" se ejecutan inmediatamente después de la sección "`void setup()`" de manera infinita, hasta que se interrumpe la alimentación de la tarjeta o se reinicia. Esto significa, que el código en la función "`loop()`" se ejecuta completo desde la primera línea de código hasta la última, y una vez alcanzada esta, empieza nuevamente desde la primera línea de código, y así una y otra vez. En el programa y subsecuentes de este documento se requiere que la biblioteca "BlynkEdgent" continue su ejecución para mantener sus servicios, entre estos la comunicación constante con los servidores de la plataforma Blynk.

Es conveniente recordar que lenguaje del IDE Arduino es "case-sensitive". Esto quiere decir que permite diferenciar entre mayúsculas y minúsculas, por ejemplo escribir "`serial.begin(115200);`" en vez de "`Serial.begin(115200);`". En el primer caso la

instrucción está escrita con “s” minúscula, esto arrojara un error de compilación, ya que para el IDE de Arduino “serial” con minúscula no tiene un significado en el lenguaje.

3.1.2.5 Interfaz web

Volviendo a la plataforma, creado el flujo de datos, con este se puede generar una interfaz de control tanto para la plataforma web como móvil. En la pestaña “Web Dashboard”, a la izquierda de la página se encuentra la “Widget Box” que es el conjunto de micro aplicaciones que permiten visualizar e interactuar con los sensores y actuadores conectados a la tarjeta de desarrollo, una vez asociada a la plataforma, para colocarlos basta con arrastrar manteniendo el clic principal del ratón y desplazar de izquierda a derecha el widget deseado para colocar en la interfaz, ver figura 3.11. Es posible agregar hasta treinta widgets a la interfaz en el plan gratuito, los widgets con la marca de candado o la palabra “UPGRADE” color anaranjado solo son utilizables en los planes de pago de la plataforma, en los widgets que lo permiten se puede habilitar la visualización en modo historial (1 hora, 1 día, 1 semana, 1 mes…, para almacenar los datos es necesario el pago de algún plan del servicio) o modo simple los datos recibidos de algún sensor y/o actuador.

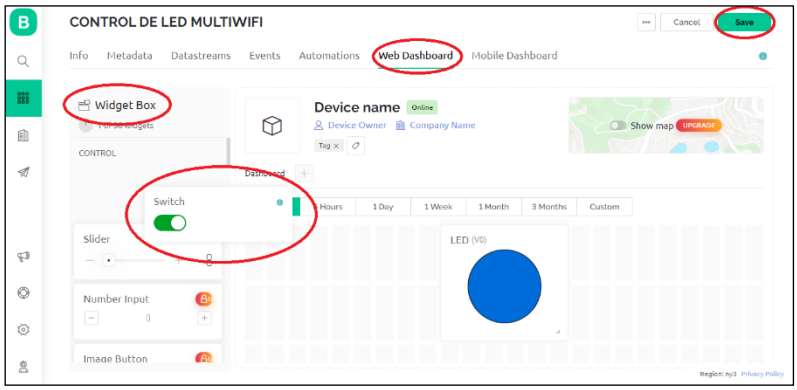


Figura 3.11. Creación de tablero de control web.

Colocados los widgets en la interfaz al pasar el puntero del ratón sobre estos se mostrarán tres pequeños iconos con los que se puede copiar, configurar y eliminar de la interfaz, también redimensionar el tamaño y forma de estos manteniendo el clic pulsado en la esquina derecha inferior del widget y arrastrando el cursor en la dirección deseada ver figura 3.12.

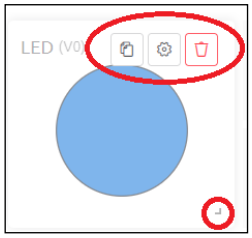


Figura 3.12. Configuración de widget.

En la configuración, cada widget tiene diferentes elementos que pueden ser establecidos, lo principal es colocar un nombre al widget para su identificación, seleccionar el flujo de datos que

estará monitorizando o por el cual se enviaran datos, así como los valores en los que realizaran las acciones, adicional se puede agregar etiquetas de la palabra o valor que mostrara al estar encendido/apagado, la posición del texto, entre otras configuraciones. Para guardar la configuración clic en el botón “Save”, ver figura 3.13.

Figura 3.13. Edición de propiedades widget.

Se guardan todos los cambios en la plantilla haciendo clic en el botón “Save” ver figura 3.11. Si es necesario realizar alguna modificación en cualquier apartado de la plantilla o crear una copiar de la misma para reutilizarla, al guardar los cambios de la plantilla actual o seleccionar alguna otra en “Templates”, en la parte superior estarán disponibles siempre un par de botones para realizar ambas acciones, ver figura 3.14.

Figura 3.14. Edición de propiedades en plantilla.

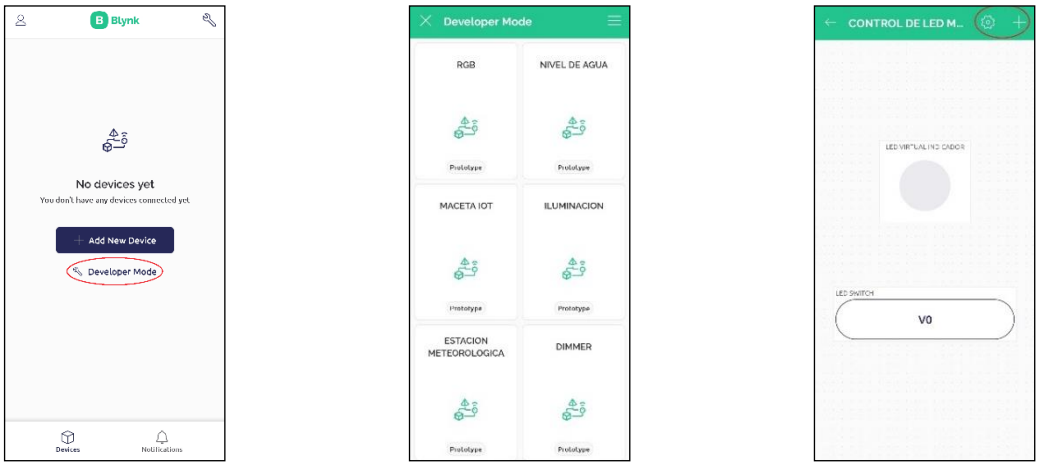
3.1.2.6 Interfaz móvil

El diseño de la interfaz de control móvil solo se puede realizar a través de la aplicación móvil por tanto es necesario instalarla, esta se encuentra disponible para Android[1] e iOS[2], también es posible la redirección por medio de la página web a la descarga de la aplicación móvil. En este documento se llevan a cabo todas las configuraciones en un dispositivo Android, para iOS la configuración es similar exceptuando algunos pequeños cambios de diseño de la aplicación.

Al abrir la aplicación lo primero que se solicita es la autentificación de nuestras credenciales de acceso o bien registrarnos por primera vez. Para iniciar sesión en una cuenta ya registrada. Se pulsa sobre el texto “Log In”, la aplicación despliega la pantalla de acceso, colocar el correo que se registró junto a la contraseña que se creó y pulsar nuevamente en el botón “Log In” (también se dispone de la opción “Forgot Password?” que permite recuperar la contraseña en caso de haberla olvidado).

[1] https://play.google.com/store/apps/details?id=cloud.blynk
[2] https://apps.apple.com/us/app/blynk-iot/id1559317868

En la pantalla principal de la aplicación, ver figura 3.15. (a), al pulsar en el texto junto a la figura de llave "🔧 Developer Mode", se mostrarán la o las plantillas que ya se han creado en la plataforma, seleccionamos la plantilla que se va a editar (si es la primera vez que se accede a esta sección, esta estará en blanco, se accede del mismo modo si se requiere realizar alguna modificación en un tablero móvil previamente creado figura 3.15. (b)), en la parte superior derecha se cuenta con dos botones que permiten, por un lado, la figura de engranaje, revisar las configuraciones de la plantilla como el nombre, cambiar el icono de identificación (el icono se muestra en la pantalla principal una vez se conecta un dispositivo a la plantilla), modificar el diseño del mismo icono, conocer los flujos de datos asociados a la plantilla y ver las credenciales para conectar un dispositivo a esta plantilla, ver figura 3.15. (c).



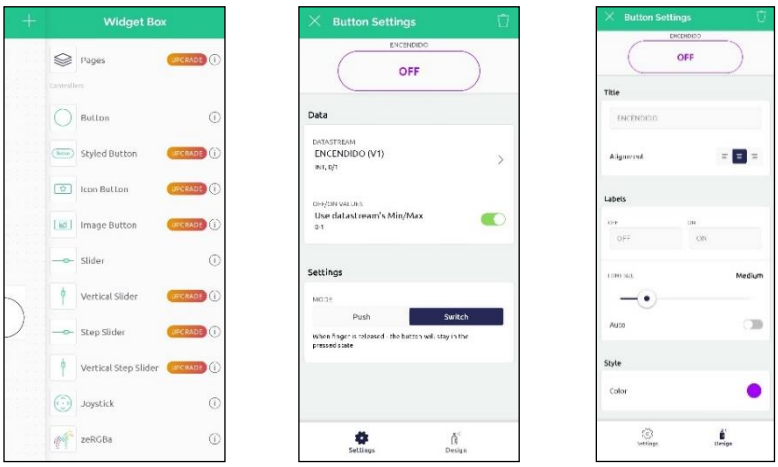
a) b) c)

Figura 3.15. Creación de tablero de control móvil.

El icono con el símbolo "+" permite abrir la "Widget Box" que es donde se encuentran las micro aplicaciones para la interfaz ver figura 3.16. Para agregar alguno solo pulsar sobre el deseado y se agregara automáticamente a la interfaz figura 3.16. (a), (como se indicó anteriormente los iconos con la marca "UPGRADE" solo pueden ser utilizados en planes de pago de la plataforma), mantener pulsado sobre el widget permite cambiar la posición de este en el tablero, en la parte superior aparecen dos iconos más, al arrastrar el widget uno permite hacer una copia de este (izquierda), el otro icono (derecha) permite eliminarlo del tablero, al soltarlo sobre la interfaz se habilita la redimensión para cambiar su tamaño.

Con una pulsación rápida sobre el widget se despliega la pantalla de configuración donde se puede elegir el canal de flujo de datos, valores a utilizar y mostrar en la interfaz, tipo de implementación, ver figura 3.16. (b), en el submenú "Design" podemos personalizar la información que muestra la miniaplicación, nombre, etiquetas, color, estos apartados pueden variar en función del widget empleado, ver figura 3.16. (c).

Para guardar los cambios en la edición de tablero móvil basta con retroceder de la interfaz en la aplicación móvil ya sea por medio de la flecha en parte superior izquierda o las teclas, botones virtuales o gestos de navegación en la capa de personalización del teléfono móvil.



a) b) c)

Figura 3.16. Edición de tablero móvil y configuración de widgets.

3.1.3 Alambrado de circuito

Con el programa listo se verifica que el entorno realice su compilación sin errores y se carga el programa a la tarjeta, ver figura 3.17. Previamente se explicó como instalar y seleccionar la tarjeta de desarrollo en el entorno de Arduino, así como programar con o sin la necesidad de pulsar el botón físico en la tarjeta para dicho fin, capítulos 2.2.4 y 2.2.5.

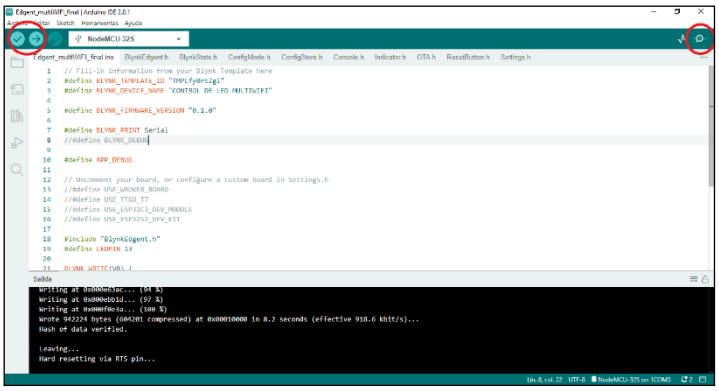


Figura 3.17. Carga del programa en tarjeta de desarrollo.

Para el alambrado del circuito se requiere de los siguientes componentes:

- 1 Protoboard (opcional)
- 1 Tarjeta NodeMCU-32S
- 1 Push button (opcional)
- 1 Capacitor 10µF a 50V (opcional)
- 1 Resistor de 220Ω a 1/2 watt
- 1 LED

- Cables jumper (opcional)

Las conexiones se realizan como se puede ver en la figura 3.18. Con fines de depuración se mantendrá la tarjeta conectada al puerto USB del equipo de cómputo para monitorear la información proveniente de la tarjeta y la interacción de esta con la aplicación móvil y así verificar su correcto funcionamiento a través del monitor serie al cual se puede acceder haciendo clic en el botón a la derecha indicado en la figura 3.17.

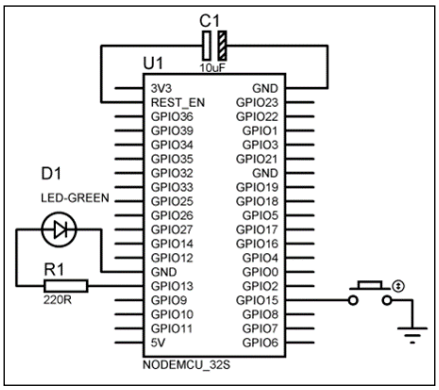
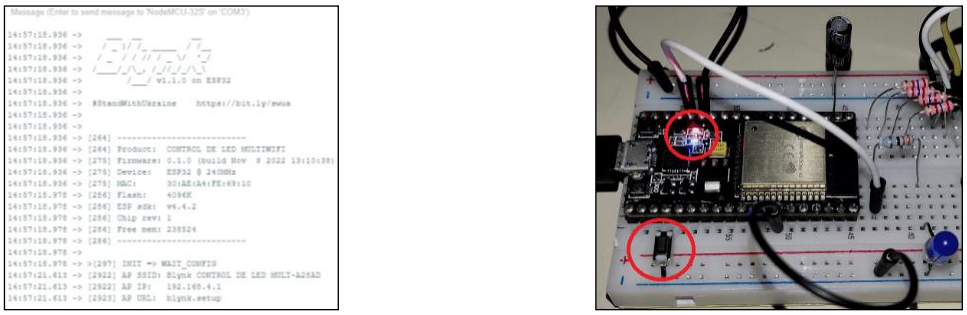


Figura 3.18. Diagrama esquemático (ejemplo control de LED a través de Blynk).

Este circuito permite verificar que las configuraciones realizadas en la biblioteca, la asociación y operación de la tarjeta de desarrollo por medio de la plataforma se realiza de forma correcta. Programada la tarjeta, a través del monitor serie en el IDE Arduino se obtiene el mensaje como se observa en la figura 3.19. (a), este indica que la tarjeta se encuentra en espera de la configuración inicial para asociarla a la plataforma por medio de la aplicación móvil. Si se ha configurado correctamente la cabecera "Settings.h", en este modo de espera el LED indicador, ver figura 3.19. (b), de la tarjeta (o el que se asignó por medio de algún otro GPIO) encenderá y apagará rápidamente.



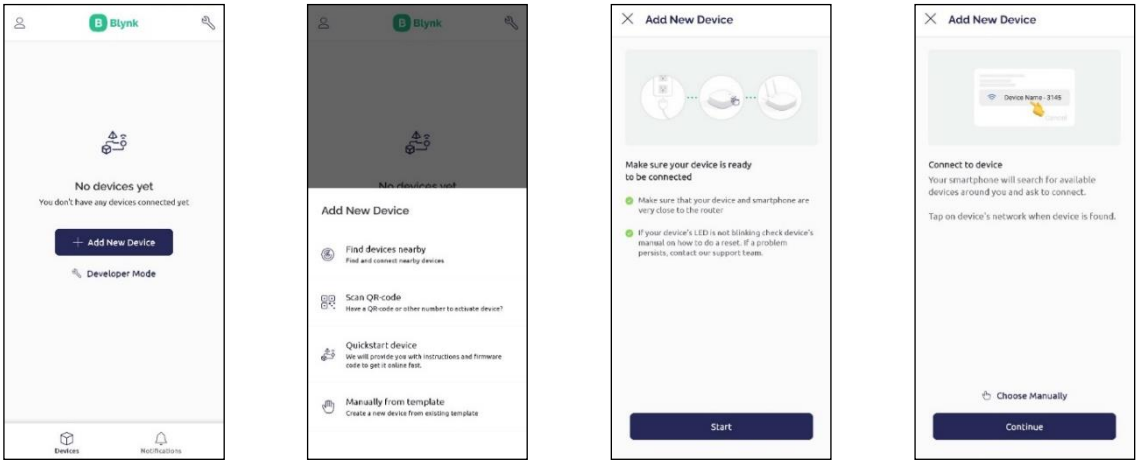
a) b)

Figura 3.19. Tarjeta de desarrollo en espera de configuración por medio de aplicación móvil.

3.1.4 Asociación de una tarjeta de desarrollo a la plataforma

En la aplicación móvil, creado o no el tablero de control móvil se pueden añadir dispositivos a la plataforma, para añadir uno compatible, programado y ya definido a una plantilla por medio

de “token” o “ID”, pulsar el botón “+ Add New Device”, ver figura 3.20. (a), en el menú que se despliega seleccionar la primera opción “Find devices nearby”, figura 3.20. (b), el dispositivo y el teléfono móvil deben estar cerca del router, pulsar el botón “Start”, figura 3.20. (c), se debe tener activa la ubicación en el teléfono móvil ya que es un requisito en la aplicación como en otras para encontrar dispositivos cercanos, para comenzar la búsqueda pulsar en “Continue”, figura 3.20. (d).

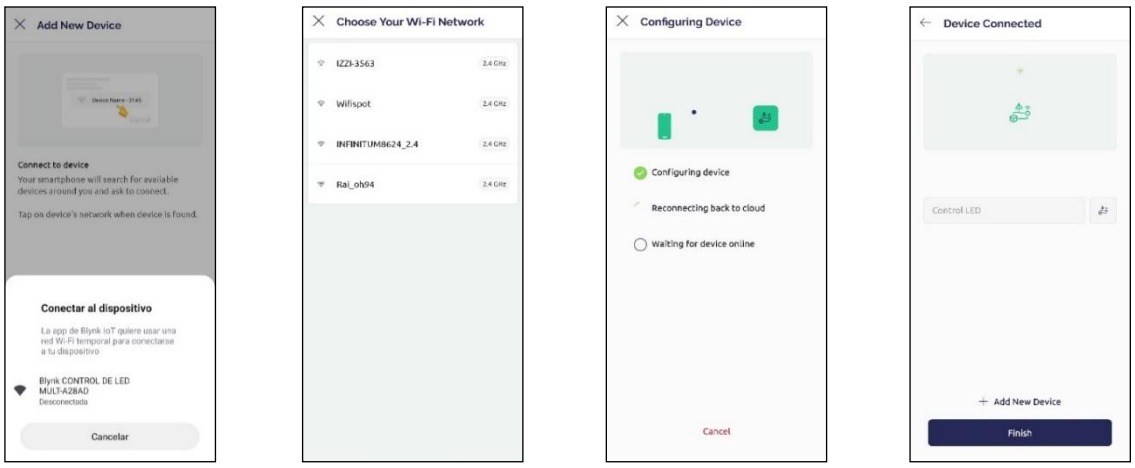


a) b) c) d)

Figura 3.20. Configuración de acceso a red por medio de aplicación móvil.

Durante este periodo de tiempo la tarjeta de desarrollo se encuentra en modo access point (AP). Al seleccionar la tarjeta en la aplicación móvil ver figura 3.21. (a), la aplicación permite conectar a este punto de acceso y así proveer las credenciales para la conexión con alguna de las redes Wifi que la tarjeta de desarrollo puede encontrar. De la lista de redes disponibles seleccionamos la red que se usara, después de añadir la clave de acceso figura 3.21. (b), pulsar en “Continue”, el proceso de asociación de la tarjeta al servido requiere de un breve periodo de espera para que la conexión se realice figura 3.21. (c). Conectado el dispositivo la aplicación permite agregar o modificar el nombre con el que identificarlo, para terminar el proceso pulsar en el botón “Finish” para finalizar la configuración, ver figura 3.21. (d).

Este proceso se repite al reiniciar las configuraciones de la tarjeta por medio del botón físico que asignamos para este fin en la configuración de la biblioteca.



a) b) c) d)

Figura 3.21. Conexión a red por medio de aplicación móvil.

Una vez conectada la tarjeta de desarrollo a la plataforma el indicador LED que se ha configurado en el capítulo 3.1.2.2 cambiara el modo de parpadeo y oscilara decreciendo y creciendo en intensidad la iluminación emitida por el mismo.

Por medio del monitor serie en el IDE Arduino se puede comprobar cómo se ha llevado a cabo el proceso de configuración, ver figura 3.22.

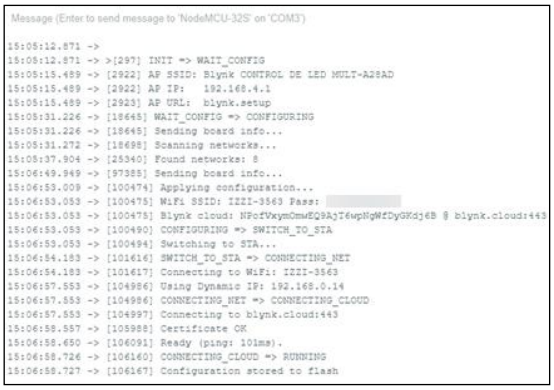


Figura 3.22. Salida de respuesta a configuración por medio de monitor serie en IDE Arduino.

El diagrama de flujo sobre la conexión y verificación con la plataforma se puede observar en la figura 3.23, el esquema incluye la modificación de este para cuando opera en modo fuera de línea (local). Cada experimento (si así está indicado) detalla cuando realiza operaciones del programa de forma local.

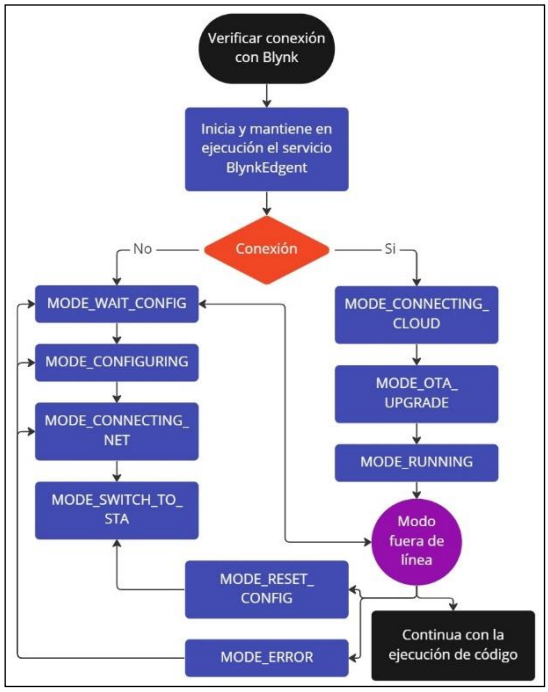
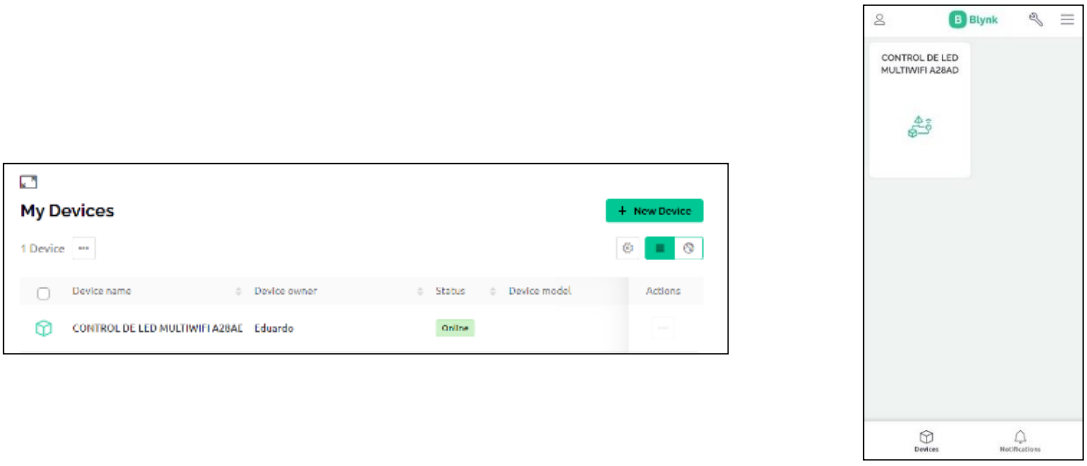


Figura 3.23. Diagrama de flujo, conexión con plataforma Blynk.

No se podrá ingresar información por medio del monitor, esta solo podrá ser agregada a través de la aplicación móvil, los datos ingresados por medio de esta última son almacenados en la memoria flash de la tarjeta de desarrollo.

De esta forma se ha conectado a una red wifi y asociado la tarjeta de desarrollo a una plantilla en la plataforma. Conectada la tarjeta a la plataforma, es posible acceder a los tableros de control web y móvil desde cualquier equipo de cómputo figura 3.24. (a), o teléfono móvil figura 3.24. (b), por medio de la cuenta registrada.



a) b)

Figura 3.24. Estado de la tarjeta de desarrollo en línea.

3.1.5 Restablecimiento de credenciales wifi

Para reiniciar las configuraciones de conexión almacenadas en la tarjeta de desarrollo basta con mantener pulsado el botón que se ha configurado como reinicio (capitulo 3.1.2.2) por 10 segundos para devolver la tarjeta al modo de configuración inicial y conectarla a una nueva red Wifi o distinta por medio de la aplicación móvil como ya se ha descrito, así no será necesario programar por medio de un equipo de cómputo en cada ocasión la tarjeta al ya no tener acceso a la red.

El indicador LED mostrara el estado de conexión en el que se encuentra la tarjeta de desarrollo con respecto a la plataforma, este al ser pulsado y mantener por 10 segundos oscilara de forma más rápida indicando que se realizara la restauración de las configuraciones, esta información puede ser confirmada por medio del monitor serie, ver figura 3.25. Pasados los 10 segundos el indicador LED dejará de mostrar la oscilación, esta es remplazada por un parpadeo rápido como indicativo de que se han reiniciado las configuraciones y se debe soltar el botón. Al soltar el push button el indicador LED parpadeara como cuando el estado inicial de la tarjeta al ser programada.

```
17:23:19.870 -> [21941] Resetting configuration!
17:23:19.870 -> [21942] Configuration stored to flash
17:23:19.870 -> [21943] RESET_CONFIG => WAIT_CONFIG
17:23:22.492 -> [24567] AP SSID: Blynk CONTROL DE LED MULT-A28AD
17:23:22.492 -> [24567] AP IP:   192.168.4.1
17:23:22.492 -> [24568] AP URL:  blynk.setup
```

Figura 3.25. Salida de respuesta por monitor serie de la restauración de configuración.

3.1. Pruebas y resultados

Como primer ejemplo se ha conectado un LED externo a la tarjeta de desarrollo, para el cual ya se ha descrito por completo la programación y configuración para asociar e interactuar mediante la plataforma IoT. De esta manera se puede comprobar que la tarjeta se ha conectado de manera exitosa y que existe comunicación bidireccional entre la plataforma y la tarjeta de desarrollo.

```
21      BLYNK_WRITE(V0) {
22        digitalWrite(LEDPIN, param.asInt());
23      }
```

A través del canal "V0" se habilita el control de un LED a Internet, por medio de este canal se puede realizar el monitoreo y cambio de estado de manera remota, como se puede observar en las figuras 3.26. y 3.27., desde cualquier interfaz de control y desde cualquier ubicación con acceso a Internet.

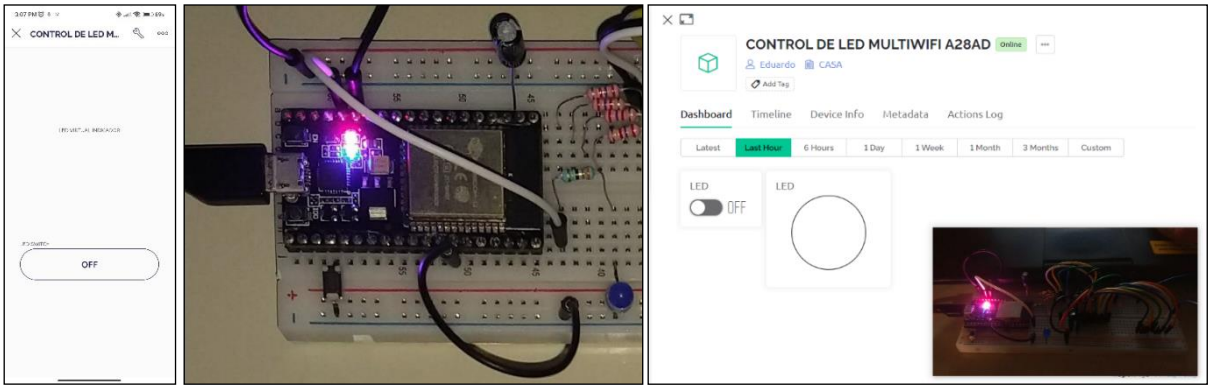


Figura 3.26. Diodo LED en estado apagado.

Al acceder a la interfaz web o móvil, se puede observar un indicador del estado de conexión entre la tarjeta de desarrollo y la plataforma, la interfaz creada permite apagar o encender el LED y conocer el estado actual de este.

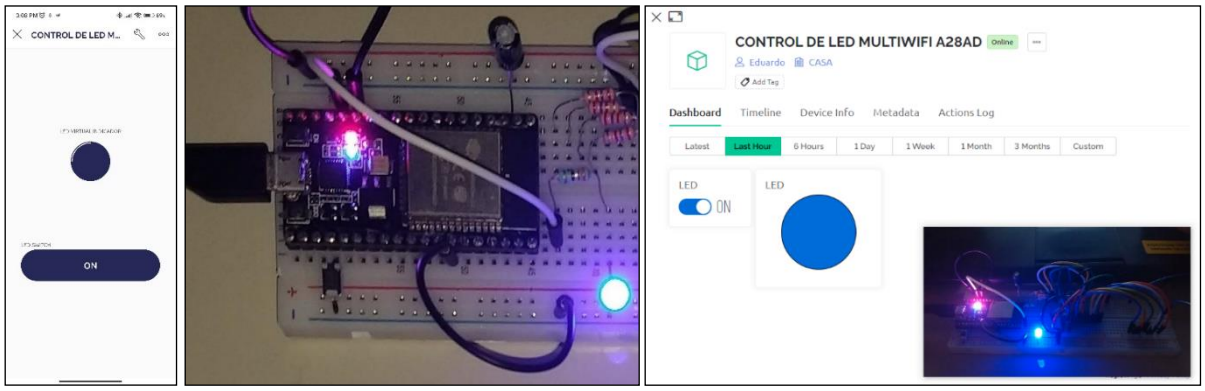


Figura 3.27. Diodo LED en estado encendido.

En este proyecto se detalla los pasos para la configurar de manera correcta una tarjeta con ESP32 para trabajar con la plataforma Blynk, el resultado del experimento es el esperado.

3.2 Control de una luminaria por medio de la plataforma Blynk (en línea)

Con lo descrito en el capítulo anterior, se toma como base el experimento realizado, remplazando el LED por una luminaria alimentada por corriente alterna, y añadiendo un circuito de potencia. Este circuito se compone de un par de resistores, un optoacoplador, un transistor y un diodo en configuración flyback, dispuestos como en el diagrama de la figura 2.62., ofreciendo aislamiento y protección de la corriente alterna al microcontrolador.

3.2.1 Diagrama de bloques

Ahora se presenta y describe (figura 3.28) un diagrama de bloques que representa a las etapas que conforman al módulo de este apartado.

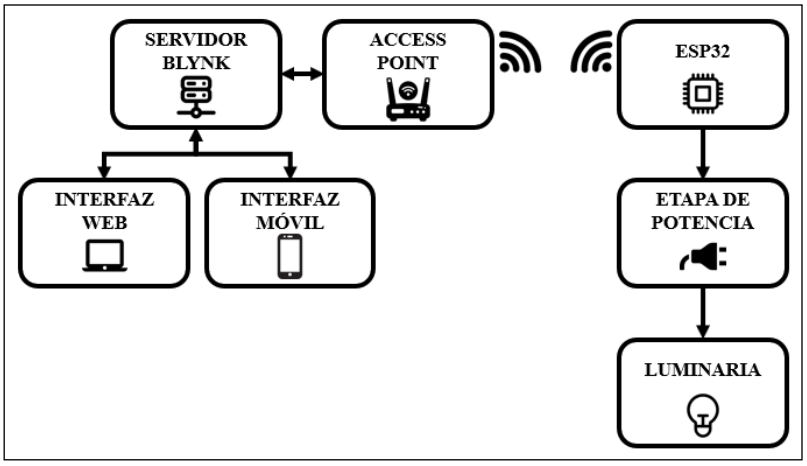


Figura 3.28. Diagrama de bloques, "Luminaria online".

- Interfaz web y móvil: a través de, es posible realizar el monitoreo y control en la plataforma Blynk, cabe recalcar que las interfaces son independientes en la creación de su diseño, ambas comparten como base canales de datos, lo que permite sincronizar y los cambios que se realicen en una interfaz sean reflejados en la otra.
- Servidor Blynk: permite administrar las configuraciones, así como visualizar la interfaz web de la plataforma, es en este dónde se crean los canales de datos que servirán como puente de comunicaciones entre las interfaces, el mismo servidor, el access point y la tarjeta ESP32.
- Access Point: es el dispositivo que permite establecer un canal de comunicación inalámbrica entre la tarjeta de desarrollo y la plataforma IoT por medio de internet. Este canal normalmente se encuentra cifrado y para obtener la conexión es necesario proporcionar al dispositivo que se conectara las credenciales de acceso (SSID y contraseña).
- ESP32: es la tarjeta de desarrollo que incluye el microcontrolador ESP32, este ya integra un módulo y antenas para realizar la conexión de manera inalámbrica con dispositivos que permiten el acceso a internet. Por medio de esta se realizan todas las operaciones lógicas de comunicación y control, lo cual incluye la gestión de la etapa de potencia y la plataforma de control IoT.
- Etapa de potencia: es el circuito electrónico que permite controlar la energía que se requiere en los actuadores de acuerdo con las funcionalidades incorporadas en el sistema.
- Luminaria: es el dispositivo que se conecta inmediatamente después de la etapa de potencia, y que en este caso produce la iluminación de un área. Puede utilizarse cualquier luminaria que trabaje a 127 VAC, y el relevador usado permite un consumo máximo de corriente de hasta 10A.

3.2.2 Diagrama esquemático

Para las conexiones entre la ESP32, la etapa de potencia y el elemento a controlar ver el diagrama esquemático de la figura 3.29.

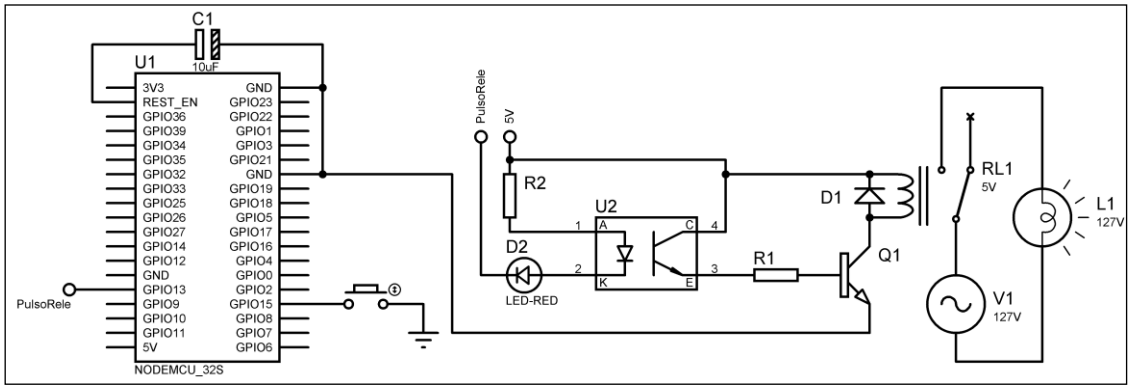


Figura 3.29. Diagrama esquemático, control de luminaria.

Para el ensamble son necesarios los siguientes dispositivos:

- 1 Tarjeta NodeMCU-32S.
- 1 Push button (opcional).
- 1 Capacitor 10µF a 50V (opcional).
- 1 Socket o portalámpara.
- 1 Módulo de relevadores SRD-5VDC-SL-C.
- 1 Conector o clavija.

En el diagrama esquemático presentado, se observa la disposición de los elementos para su conexión. Es importante tener en cuenta que el capacitor y el push button son componentes opcionales, puesto que en la tarjeta NodeMCU-32S se puede utilizar el push button integrado para ambas funciones (programar la tarjeta y, si se mantiene pulsado por diez segundos, reiniciar las configuraciones para la conexión Wifi de la tarjeta una vez cargada la biblioteca de Blynk). Para más detalles, consultar los capítulos 2.2.5 y 3.1.2.2.

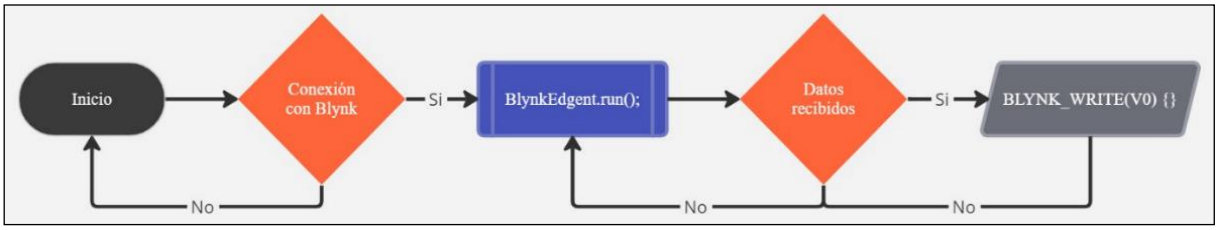
Para conectar una luminaria a un microcontrolador como el ESP32, es necesario separar las señales de control y alimentación. Esto se logra mediante un relé. En la figura 2.62 se muestra el ensamble de las conexiones, donde se requieren un resistor (R1) de 1kΩ, un R2 de 500Ω, un circuito integrado optoacoplador FL817C (U1), un transistor J3Y(Q1) y un diodo (D2) en configuración flyback, para protección. Para más información, consultar el capítulo 2.5.1.

La configuración mencionada para el circuito de potencia permite conectar una luminaria de corriente alterna al microcontrolador ESP32 de hasta 100 watts sin inconvenientes.

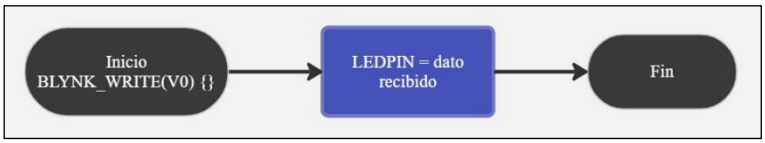
El circuito de potencia se controla mediante el GPIO13 de la tarjeta, que debe ser energizado con 5V a través de una fuente externa. Dicha fuente también puede alimentar a la tarjeta de desarrollo mediante el pin 5V. Para más detalles, consultar los capítulos 2.2.5 y 2.5.4.

3.2.3 Diagrama de flujo

En la figura 3.30 se presenta el diagrama de flujo, en este se puede observar la línea de programación que sigue la ejecución del programa, se omite la explicación de esta ya que se ha descrito en el capítulo 3.1 junto al proceso de registro de dispositivos en la plataforma Blynk.



a)



b)

Figura 3.30. a) Diagrama de flujo programa principal, b) Subrutina de función BLYNK_WRITE(V0)

3.2.4 Pruebas y resultados

El experimento que se presenta a continuación surge a partir del control de un LED mediante la plataforma Blynk. Para llevar a cabo este experimento, es necesario incorporar una etapa de potencia que permita controlar una luminaria alimentada con corriente eléctrica de la red. La programación utilizada en este experimento es la misma que en el experimento presentado en el capítulo 3.1, y se hace uso de la misma interfaz de control. En la figura 3.31 y 3.32, se puede observar tanto la interfaz web como la móvil, junto con el correspondiente estado de la luminaria.

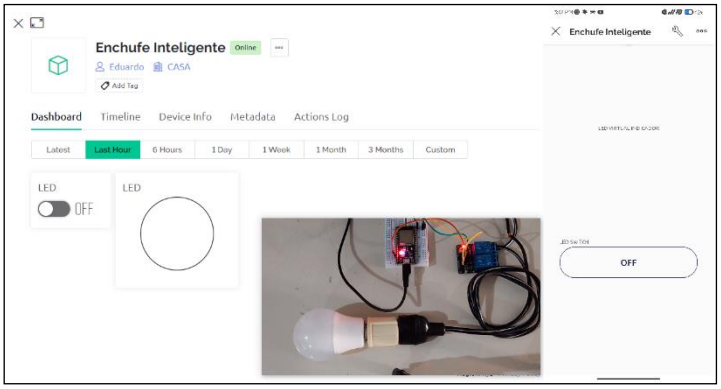


Figura 3.31. Luminaria en estado apagado, interfaz web (izquierda), móvil (derecha).

Es importante destacar que tanto el programa como la interfaz presentada en este experimento no se limitan exclusivamente al control de luminarias. Gracias a su implementación, es posible extender su uso a otros dispositivos inteligentes, como enchufes inteligentes, por ejemplo. Es necesario tener en cuenta el consumo de corriente que puede soportar el relé utilizado en el experimento y, en caso de ser necesario, sustituirlo por una etapa de potencia más adecuada. Esta consideración se presenta como un posible trabajo futuro.

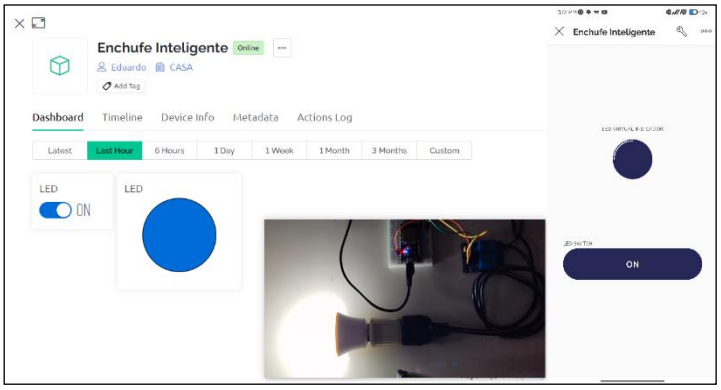


Figura 3.32. Luminaria en estado encendido, interfaz web (izquierda), móvil (derecha).

El presente proyecto tiene como objetivo introducir el uso de una tarjeta con ESP32 y la plataforma Blynk para el control de dispositivos que demandan el uso de corriente alterna, mediante el uso de una etapa de potencia correspondiente. Los resultados del experimento demuestran que se ha alcanzado el objetivo previsto.

3.3 Control de luminarias con notificaciones de presencia y asistentes de voz (en línea y local)

Los siguientes programas son un ejemplo de cómo utilizar las plataformas Blynk, SinricPro y ESP RainMaker junto con una tarjeta con ESP32 para controlar múltiples, actuadores y sensores, por medio de IoT.

En el entorno del IDE Arduino se toma como base cualquiera de los programas anteriores o bien abrir el programa de ejemplo que contiene la biblioteca para asociar una tarjeta de desarrollo con ESP32 a la plataforma Blynk (si se opta por iniciar con el programa de ejemplo como base se recomienda consultar el experimento 3.1 ya que en este se realiza la descripción a detalle de las modificaciones a este programa para su correcta implementación en el modelo de tarjeta NodeMCU-32S).

3.3.1 Diagrama esquemático

Para las conexiones entre la ESP32, la etapa de potencia y los elementos a controlar ver el diagrama esquemático de la figura 3.33.

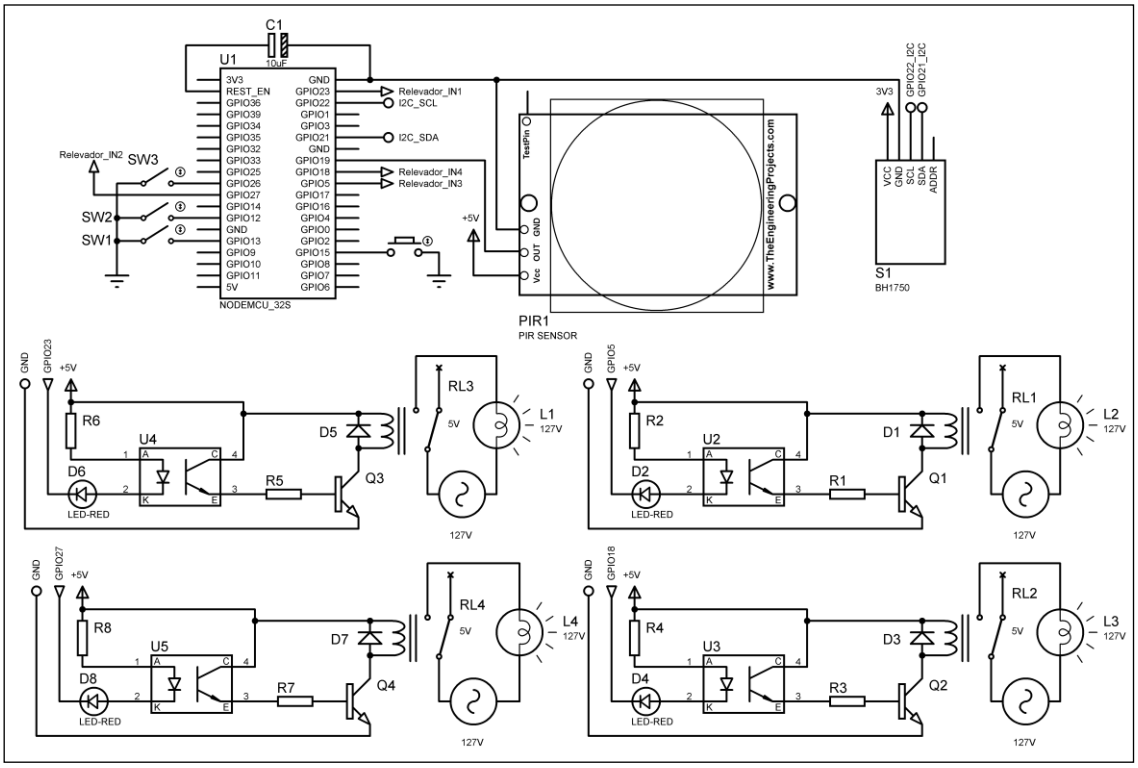


Figura 3.33. Diagrama esquemático, control de luminarias (múltiples plataformas).

Material necesario:

- 1 Tarjeta NodeMCU-32S.
- 1 Push button (opcional).
- 1 Capacitor 10µF a 50V (opcional).
- 3 o 4 Switchs basculantes.
- 1 Módulo HC-SR501 (solo para experimento en Blynk).
- 1 Módulo BH1750.
- 1 Módulo de relevadores SRD-5VDC-SL-C.

En el diagrama esquemático se puede apreciar la disposición de los elementos necesarios para su conexión. Es importante recordar que el capacitor y el push button son componentes opcionales, ya que la tarjeta NodeMCU-32S cuenta con un botón integrado que puede utilizarse para ambas funciones: programar la tarjeta y reiniciar las configuraciones para la conexión Wifi de la tarjeta si se mantiene pulsado por diez segundos. Para más detalles, consultar los capítulos 2.2.5 y 3.1.2.2.

Para conectar múltiples luminarias a un microcontrolador como el ESP32, es necesario separar las señales de control y la alimentación de cada una mediante el uso de relés, como se ha explicado en el capítulo anterior. La figura 2.62 muestra el ensamblaje detallado de las conexiones. Esta configuración para el circuito de potencia permite conectar sin inconvenientes una luminaria de corriente alterna al microcontrolador ESP32 de hasta 100 watts, por cada relé.

El circuito de potencia es controlado mediante los GPIO 5, 18, 23 y 27 de la tarjeta, y debe ser energizado con una fuente externa de 5V. Esta misma fuente también puede alimentar a la tarjeta de desarrollo a través del pin 5V. Para más detalles, consultar los capítulos 2.2.5 y 2.5.4.

En el diagrama también se puede observar la conexión entre los módulos HC-SR501 y BH1750, los cuales solo son utilizables en el proyecto desarrollado para la plataforma Blynk. El módulo HC-SR501 permite reconocer la presencia de fuentes de energía calorífica, como el cuerpo humano, por medio de radiación infrarroja. Este módulo requiere de un pin en la tarjeta (GPIO19) para enviar las señales que el sensor procesa y se energiza con 5V. Además, permite ajustes de sensibilidad, para más detalles consultar el capítulo 2.3.2.

El módulo BH1750 permite conocer la cantidad de iluminación de un lugar, y la información que proporciona es procesada por el microcontrolador ESP32, quien, en base a la programación realizada, puede realizar acciones de control. Para su conexión, se utiliza la comunicación I2C, ocupando los pines correspondientes en la tarjeta de desarrollo.

Para permitir el control manual de las luminarias sin necesidad de la interfaz web, móvil o asistente virtual, el proyecto cuenta con la conexión de tres interruptores mediante los GPIO 13, 12, y 26. Esta característica permite prescindir de la necesidad de mantener una conexión inalámbrica en la tarjeta de desarrollo.

3.3.2 Diagrama de bloques plataforma Blynk

Ahora se presenta y describe (figura 3.34) el diagrama de bloques en donde se representan las etapas que conforman a cada uno de los módulos de este apartado.

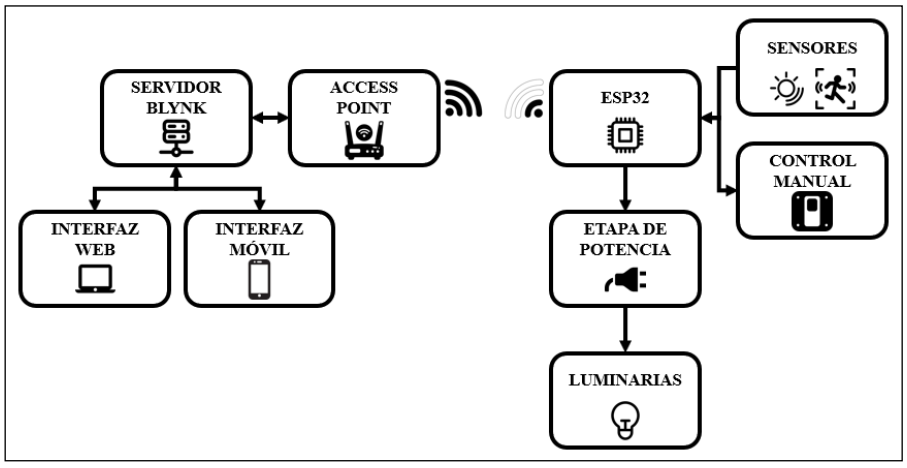


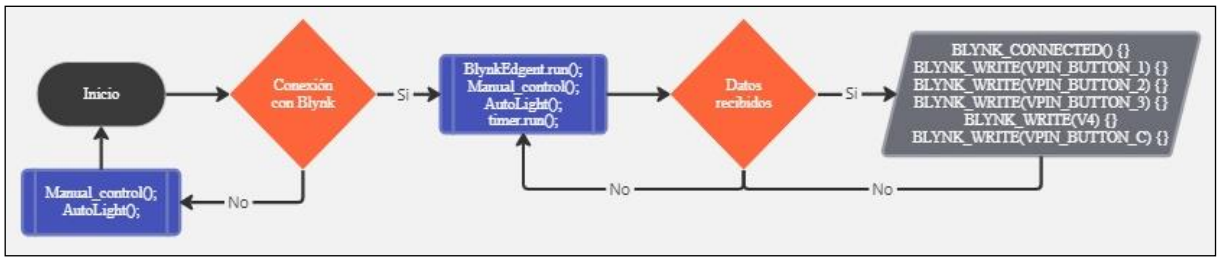
Figura 3.34. Diagrama de bloques, plataforma Blynk.

- Interfaz web, móvil: a través de, es posible realizar el monitoreo y control en la plataforma Blynk. Cabe recalcar que las interfaces son independientes en la creación de su diseño, ambas comparten como base canales de datos, lo que permite sincronizar y los cambios que se realicen en una interfaz sean reflejados en la otra.
- Servidor: cuenta con la infraestructura que permite administrar las configuraciones, así como visualizar la interfaz web o móvil de la plataforma. Es en este dónde se crean los

canales de datos que servirán como puente de comunicaciones entre las interfaces, servidor, access point y la tarjeta ESP32.

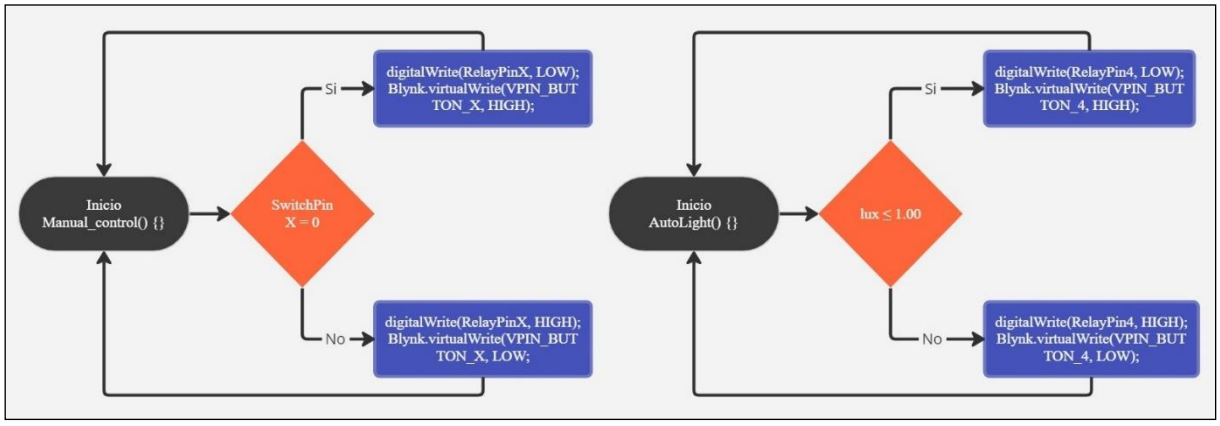
- Access Point: es el dispositivo que permite establecer un canal de comunicación inalámbrica entre la tarjeta de desarrollo y la plataforma IoT por medio de internet. Este canal normalmente se encuentra cifrado y para obtener la conexión es necesario proporcionar al dispositivo que se conectara las credenciales de acceso (SSID y contraseña).
- ESP32: es la tarjeta de desarrollo que incluye el microcontrolador ESP32, este ya integra un módulo y antenas para realizar la conexión de manera inalámbrica con dispositivos que permiten el acceso a internet[1]. Por medio de esta se realizan todas las operaciones lógicas de comunicación y control, lo cual incluye la gestión de la etapa de potencia y la plataforma de control IoT.
- Etapa de potencia: es el circuito electrónico que permite controlar la energía que se requiere en los actuadores de acuerdo con las funcionalidades incorporadas en el sistema.
- Sensores: en la plataforma Blynk es posible aprovechar algunas funcionalidades adicionales como las notificaciones de eventos, es por ello al trabajar con esta se integraron un par de sensores para medir el nivel de la iluminación (aprovechado como automático), así como si existe la presencia o movimiento en ciertos entornos.
- Control manual: conjunto de interruptores que permiten el control manual de las luminarias de forma local, sin depender de una conexión inalámbrica en la tarjeta de desarrollo. Al actualizar el estado de forma manual el cambio también es reflejado en las plataformas en línea.
- Luminarias: es el dispositivo que se conecta inmediatamente después de la etapa de potencia, y que en este caso produce la iluminación de un área. Puede utilizarse cualquier luminaria que trabaje a 127 VAC, por cada relevador usado, estos permiten un consumo máximo de corriente de hasta 10A.

3.3.2.1 Diagrama de flujo para plataforma Blynk

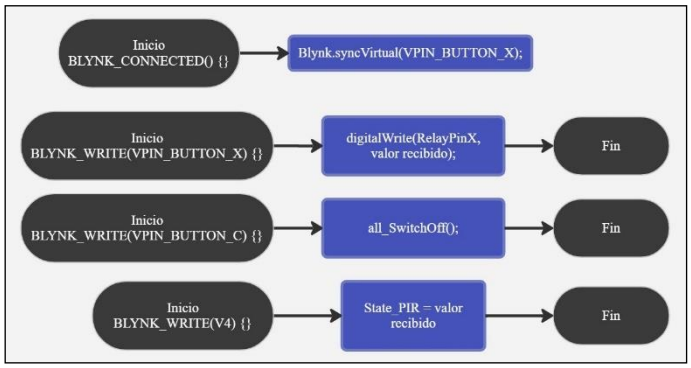


a)

[1] Nota: el icono indica la potencia de la señal a la conexión inalámbrica, al ser este como se muestra, denota que esta señal puede ser muy baja o nula, en este caso los programas que lo indiquen pueden continuar la ejecución del código sin la necesidad de una conexión inalámbrica.



b)



c)

Figura 3.35. a) Diagrama de flujo, programa principal, b) Subrutina de ejecución local y en línea, c) Subrutinas de ejecución en línea.

3.3.2.2 Implementación plataforma Blynk

Este programa está diseñado para controlar la iluminación de un hogar utilizando una tarjeta con ESP32 a través de la plataforma Blynk. Las funciones principales son el control local (mediante interruptores) y por medio de aplicación web o móvil de la iluminación, el control automático de la iluminación en función del nivel de luz ambiente y la detección de movimiento a través de un sensor PIR.

El programa comienza definiendo algunas variables de configuración, como las credenciales de la plantilla de Blynk, la versión del firmware y los pines utilizados para los relés, interruptores y sensores.

A continuación, se importan algunas librerías necesarias y se definen algunas variables booleanas que indican el estado de los interruptores y sensores. Algunas de las funciones principales se configuran utilizando la biblioteca de Blynk.

```
25      bool State_PIR = LOW;
26      bool toggleState_1 = LOW;
```

```
27      bool toggleState_2 = LOW;
28      bool toggleState_3 = LOW;
29      bool toggleState_4 = LOW;
30      bool SwitchState_1 = LOW;
31      bool SwitchState_2 = LOW;
32      bool SwitchState_3 = LOW;
33      bool SwitchState_4 = LOW;
34
35      #define APP_DEBUG
36
37      #include "BlynkEdgent.h"
38      #include <BH1750.h>
39      #include <Wire.h>
40      BlynkTimer timer;
41      BH1750 lightMeter;
```

Cabe destacar que el programa utiliza múltiples bibliotecas y cabeceras, como "`BlynkEdgent.h`", "`BH1750.h`" y "`Wire.h`". Además, el programa está diseñado para utilizar un temporizador incluido en la biblioteca Blynk, "`BlynkTimer`" para llamar algunas de las funciones periódicamente.

En la función "`setup()`", se inicializan los pines y se comienza la comunicación serial. También se inician los objetos necesarios para la medición de la luz y se configuran dos temporizadores que se encargarán de llamar a las funciones "`SensPIR()`" y "`AutoLight()`".
La función "`SensPIR()`" se encarga de leer el sensor PIR (Passive Infrared Sensor) y si se detecta movimiento (dependiendo del estado de la variable "`State_PIR`"), se envía un evento de registro a Blynk, (con fines de depuración, se escribe en la consola el mensaje "Hay Movimiento". Si no se detecta movimiento, se escribe en la consola el mensaje "Sin movimiento").

La función "`AutoLight()`" mide la intensidad de luz ambiental y, si ésta es baja (menor o igual a 1.00, este valor debe ser ajustado dependiendo del entorno y necesidades de aplicación), se enciende el cuarto relé y se envía un valor de "1" al botón virtual correspondiente en la aplicación de Blynk. Si la luz es suficiente, se apaga el relé y se envía un valor de "0" al botón virtual, sincronizando el estado en la plataforma.

La función "`Manual_control()`" se encarga de controlar los relés a través de los interruptores físicos conectados a los pines correspondientes, por medio de sentencias "`if`", que actualizan también el estado de los botones virtuales en la plataforma Blynk cuando existe conexión con esta.

```
155       if (digitalRead(SwitchPin1) == LOW && SwitchState_1 == LOW) {
156         digitalWrite(RelayPin1, LOW);
157         Blynk.virtualWrite(VPIN_BUTTON_1, HIGH);
158         toggleState_1 = HIGH;
159         SwitchState_1 = HIGH;
160       }
161       if (digitalRead(SwitchPin1) == HIGH && SwitchState_1 == HIGH) {
162         digitalWrite(RelayPin1, HIGH);
163         Blynk.virtualWrite(VPIN_BUTTON_1, LOW);
```

```
164         toggleState_1 = LOW;
165         SwitchState_1 = LOW;
```

La función "`all_SwitchOff()`" apaga todas las luces modificando las variables toggleState_X y el estado de los relevadores por medio de la función digitalWrite, al recibir el valor del botón virtual de la plataforma Blynk.

```
115     void app_loop() {
116         edgentTimer.run();
117         edgentConsole.run();
118         Manual_control();
119         AutoLight();
120     }
```

Finalmente, las funciones "`BLYNK_CONNECTED()`" y "`BLYNK_WRITE()`" están encargadas de forzar la sincronización de los valores en los botones virtuales de la aplicación de Blynk con los estados de los interruptores y relés físicos en la placa ESP32.

En resumen, este programa utiliza la plataforma Blynk para controlar varios relés y sensores en una placa ESP32 y permitir el monitoreo y control remoto de estos dispositivos desde una aplicación móvil.

```
25
26      void Manual_control();
27      void AutoLight();
28      inline...
```

Para la ejecución del programa de manera local se debe realizar las modificaciones de las cabeceras "ConfigMode.h" y "Settings.h", como se indica en anexo I. Así como también se debe agregar las funciones "`Manual_control()`" y "`AutoLight()`" entre las líneas 25 y 117 de la cabecera "BlynkEdgent.h", con la finalidad de garantizar la ejecución de estas funciones al no haber comunicación con la plataforma Blynk.

3.3.2.2.1 Interfaz web plataforma Blynk

La construcción de la interfaz de control web requiere la creación de una nueva plantilla que incluya seis canales de datos para el control de cuatro interruptores (encendido "1" y apagado "0"), tres canales para luminarias (dos disponibles en la interfaz web y uno en la interfaz móvil) y un canal para activar o desactivar el sensor de presencia (PIR) como medida de seguridad. Además, se debe incluir un widget indicador de estado para una luminaria automática denominada "ENTRADA PRINCIPAL".

La disposición y tamaño de los widgets utilizados en la interfaz de control son de uso libre y dependerán del criterio y preferencia del diseñador. Esto se puede apreciar en la figura 3.36. Para cada interruptor es necesario indicar el canal de datos correspondiente (Datastream), así como el valor que se enviará o recibirá en los estados encendido y apagado, respectivamente. También se debe proporcionar un nombre para el widget, así como etiquetas y posición adecuadas.

Por último, para configurar el indicador de estado, es necesario establecer el canal de datos correspondiente, el valor de estado que se recibirá y el nombre del widget.

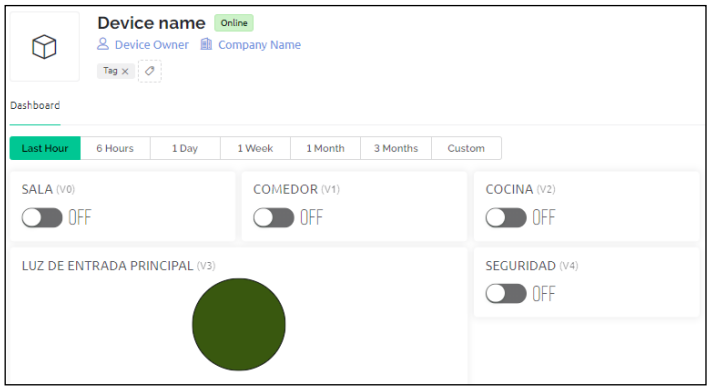


Figura 3.36. Iluminación, interfaz web, Blynk.

3.3.2.2.2 Interfaz móvil plataforma Blynk

En la interfaz móvil se ubican cinco interruptores, tres de ellos destinados al control de las luminarias, otro para la función de "seguridad" y un último para apagar todas las luminarias a la vez. Además, en la interfaz móvil se encuentra un indicador de estado para la luminaria de encendido automático. La disposición y tamaño de los widgets utilizados en esta interfaz son de uso libre y pueden variar en función de las preferencias del diseñador. Esto se puede apreciar en la figura 3.37.

Figura 3.37. Iluminación, interfaz móvil.

En cuanto a la configuración de los widgets, para los interruptores se debe especificar el nombre del widget, el canal de datos correspondiente, el tipo de interruptor (switch o push), y el color y la posición de las etiquetas, entre otras opciones importantes. En este caso, cuatro de estos interruptores serán del tipo switch, mientras que el que fuerza el apagado de todas las luminarias será del tipo push. Opcionalmente, se puede indicar el valor del interruptor o utilizar los valores

proporcionados por el canal (configuración automática), que se encuentra configurado por defecto.

Por último, para configurar el indicador de estado, es necesario establecer el nombre del widget, el canal de datos correspondiente, y opcionalmente el color del widget.

3.3.2.3 Pruebas y resultados plataforma Blynk

Se llevaron a cabo pruebas preliminares con el fin de asegurar el correcto funcionamiento de los actuadores (relés) en conjunto con los sensores BH1750 y PIR. Durante estas pruebas, se midieron diversos parámetros y se utilizó dicha información para calibrar los sensores. Cabe destacar que esta calibración se llevó a cabo únicamente con propósitos experimentales y que para una implementación real se deben tomar en cuenta un mayor número de muestras, así como los factores adicionales que puedan afectar la recopilación de estas.

Se llevaron a cabo pruebas para la implementación del proyecto en la plataforma Blynk, incluyendo la utilización de la función "Events", la cual permite programar alertas basadas en las acciones realizadas por el usuario o en las lecturas obtenidas de manera autónoma. En este proyecto, se utilizan para un sistema simple de seguridad que notifica al usuario a través de la aplicación móvil, como se muestra en la figura 3.38. Es importante destacar que la configuración de este evento no requiere programación adicional para la aplicación móvil y se ha explicado previamente el proceso de configuración.

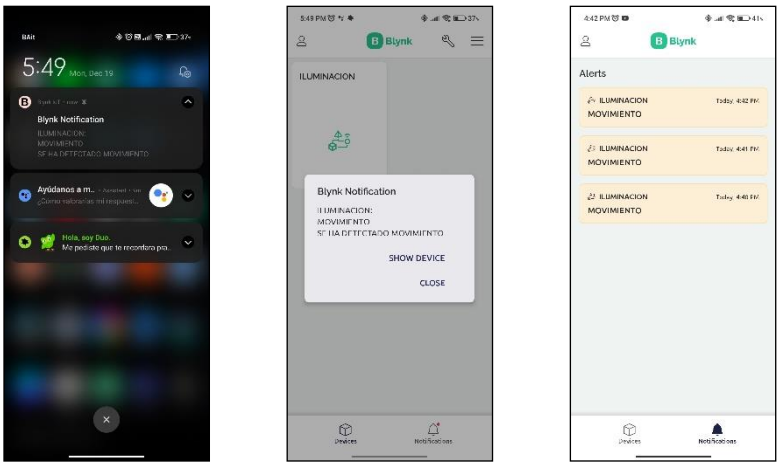


Figura 3.38. Notificaciones en la aplicación móvil.

A estas pruebas se adjunta la investigación y adaptación del programa con el fin de garantizar que el proyecto siga funcionando incluso en caso de que no haya conexión inalámbrica o de comunicación con el servidor de la plataforma. Esta parte del código depende completamente de los interruptores manuales, el estado de estos se refleja en la plataforma una vez que se restablece la comunicación.

En este proyecto, se intentó incluir, al igual que en los proyectos posteriores, el control mediante asistentes virtuales. Blynk ofrece acceso a esta API en sus planes de pago, por lo que como alternativa se realizaron pruebas a través de una plataforma externa: If This Then That (IFTTT).

Sin embargo, cuando se comenzó a experimentar con esta plataforma, IFTTT actualizó sus APIs para asistentes virtuales, lo que inhabilitó el control de acciones en múltiples plataformas, incluyendo Blynk, dejando así inutilizable su uso en los proyectos. Debido a esto, se decidió descartar esta implementación del proyecto final.

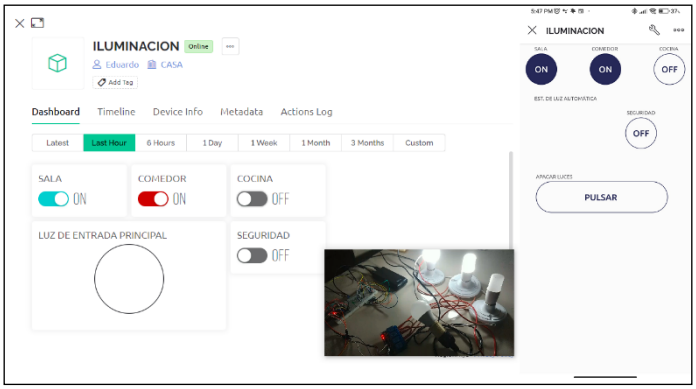


Figura 3.39. Control de iluminación por medio de la interfaz en Blynk.

Como resultado, se logró obtener un sistema de control de iluminación y, de manera complementaria, un sistema de seguridad que incluye alertas y notificaciones. En cuanto a la experimentación, el sistema demostró un comportamiento adecuado, permitiendo encender o apagar las luces de forma remota y cómoda, como se muestra en la figura 3.39. Además, también se puede controlar el sistema de manera física a través de interruptores manuales y sin depender de una conexión inalámbrica. Como pauta para trabajos futuros, se considera una posible mejora estética o embalaje para facilitar la instalación del usuario final.

3.3.3 Diagrama de bloques plataforma Sinric Pro

Ahora se presenta y describe (figura 3.40) el diagrama de bloques en donde se representan las etapas que conforman a cada uno de los módulos de este apartado[1].

[1] Debido a la similitud, se ha omitido la descripción de aquellos elementos expuestos previamente en diagrama de bloques correspondiente a la plataforma Blynk.

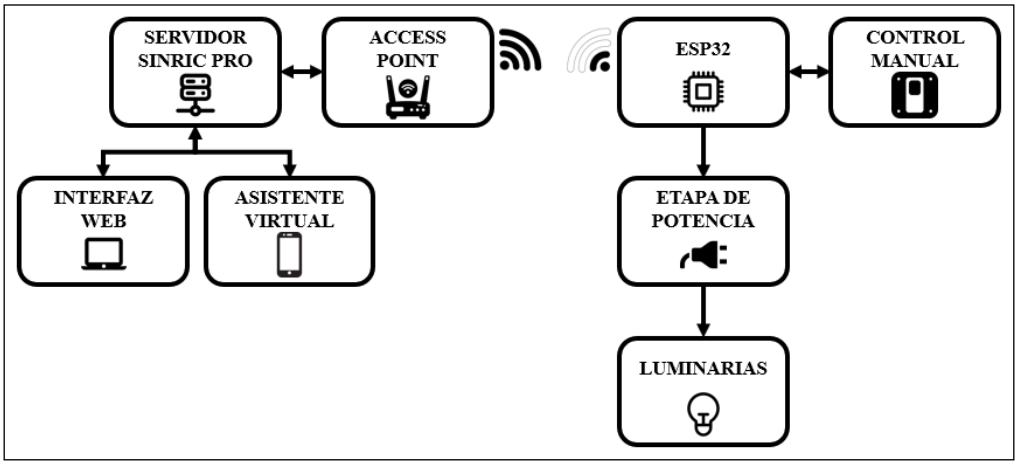
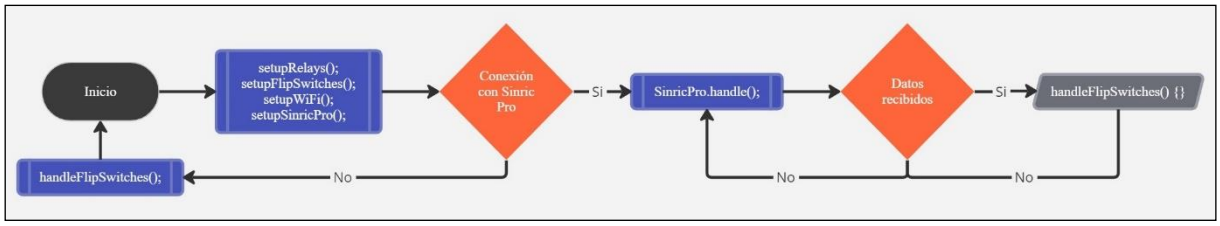


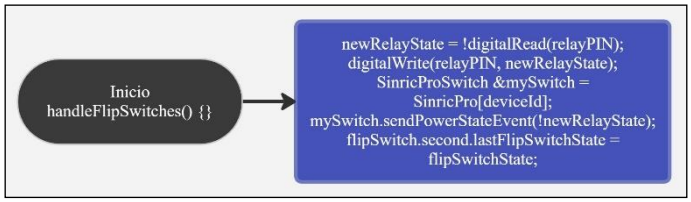
Figura 3.40. Diagrama de bloques, plataforma Sinric Pro.

- Interfaz web, asistente virtual: a través de, es posible realizar el monitoreo y control en la plataforma Sinric Pro. Permite generar una interfaz web bastante limitada en personalización y cuenta con complemento para asociar rápidamente a los asistentes virtuales.
- Servidor: cuenta con la infraestructura que permite administrar las configuraciones, así como visualizar la interfaz web de la plataforma. Es en este dónde se crean los canales de datos que servirán como puente de comunicaciones entre las interfaces, servidor, access point y la tarjeta ESP32.

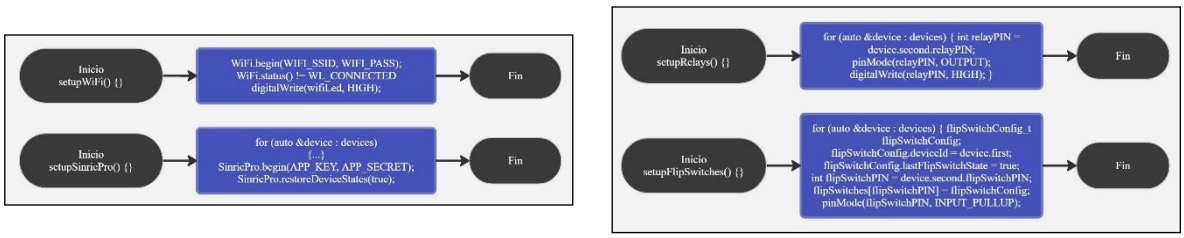
3.3.3.1 Diagrama de flujo Sinric Pro



a)



b)



c)

Figura 3.41. a) Diagrama de flujo, programa principal, b) Subrutina de ejecución local y en línea, c) Subrutinas de ejecución inicial y configuración.

3.3.3.2 Implementación plataforma Sinric Pro

El programa es una implementación de un interruptor inteligente utilizando la plataforma IoT SinricPro en una tarjeta ESP32. Permite controlar tres dispositivos (versión gratuita) a través de relés conectados a un ESP32. Los dispositivos se pueden controlar de manera remota a través de la nube de SinricPro o mediante interruptores manuales conectados a los pines asignados como entrada en el ESP32. El programa utiliza la biblioteca SinricPro para la comunicación con la plataforma de IoT y la biblioteca Wifi para la conexión a Internet. El programa prescinde del uso del sensor de presencia (PIR) y la luminaria automática.

En las primeras líneas de código, se definen las credenciales de conexión Wifi y las claves (`APP_KEY` y `APP_SECRET`) que proveen de acceso a la API de SinricPro y se obtienen en el apartado de "Credenciales" en la plataforma, ver figura 3.43. A continuación, se definen las ID de los dispositivos que se controlarán, esta ID se obtienen al crear el dispositivo en la plataforma, ver figura 3.42, así también se definen los pines de la placa ESP32 a los que se conectarán los relés y los interruptores de cada dispositivo.

```
17      #define WIFI_SSID         "IdWifi-xxxx"
18      #define WIFI_PASS         "contraseñaRedWifi"
19      #define APP_KEY           "fe0bxxxx-1x3x-4x3x-ax2x-5dabxxxxxxxx"
20      #define APP_SECRET        "c9a6xxxx-x3x7-4x3x-xexe-e86724a9xxxx-
4c4axxxx-3x3x-x5xe-x9x3-333d65xxxxxx"
21
22      //colocar el Id proporcionado en la plataforma
23      #define device_ID_1   "xxxxxxxxxxxxxxxxxxxxx18f"
24      #define device_ID_2   "xxxxxxxxxxxxxxxxxxxxx9d1"
25      #define device_ID_3   "xxxxxxxxxxxxxxxxxxxxx942"
26      #define device_ID_4   "dispositivo_adicional_con_costo"
```

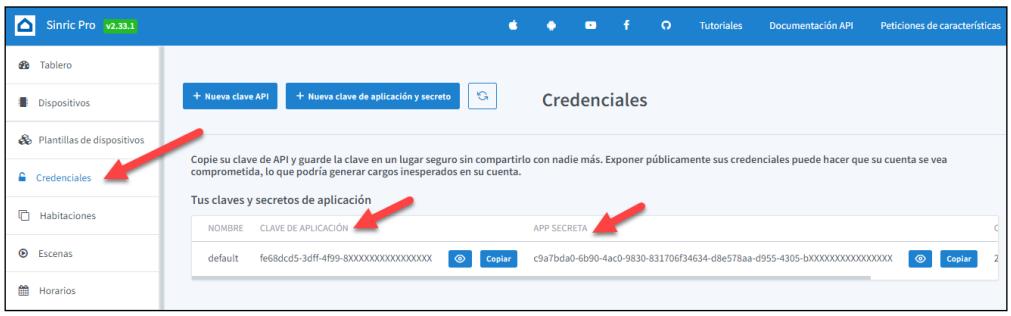


Figura 3.42. Claves de acceso para API de Sinric Pro.

El programa utiliza estructuras de datos tipo map que asigna los ID de dispositivo a los pines de relé, así como una adicional que asigna los pines de los switches físicos al correspondiente ID de dispositivo.

Dentro de las funciones principales destacan:

```
70      void setupRelays() {
71        for (auto &device : devices) {
72          int relayPIN = device.second.relayPIN;
73          pinMode(relayPIN, OUTPUT);
74          digitalWrite(relayPIN, HIGH);
75        }
76      }
77
78      void setupFlipSwitches() {
79        for (auto &device : devices)  {
80          flipSwitchConfig_t flipSwitchConfig;
81
82          flipSwitchConfig.deviceId = device.first;
83          flipSwitchConfig.lastFlipSwitchChange = 0;
84          flipSwitchConfig.lastFlipSwitchState = true;
85
86          int flipSwitchPIN = device.second.flipSwitchPIN;
87
88          flipSwitches[flipSwitchPIN] = flipSwitchConfig;
89          pinMode(flipSwitchPIN, INPUT_PULLUP);
90        }
91      }
```

La función "`setupRelays()`" inicializa los pines de los relés como salidas digitales. Utiliza un bucle "`for`" para activar todos los dispositivos en la map "`devices`" y configura el pin de relé correspondiente como salida y lo establece en un valor alto, para evitar que los relés se enciendan accidentalmente al inicio del programa. La función "`setupFlipSwitches()`" inicializa los pines de los interruptores como entradas digitales y los configura con las resistores internos en pull-up. Además, esta función crea una lista (flipSwitches) con las configuraciones de cada interruptor, que incluye la ID del dispositivo asociado, el estado anterior del interruptor y la última vez que se cambió de estado.

```
93      bool onPowerState(String deviceId, bool &state) {
```

```
94          Serial.printf("%s: %s\r\n", deviceId.c_str(), state ? "on" :
"off");
95          int relayPIN = devices[deviceId].relayPIN;
96          digitalWrite(relayPIN, !state);
97          return true;
98        }
```

La función "`onPowerState()`" se encarga de cambiar el estado del relé cuando se recibe una señal desde la plataforma SinricPro indicando que el usuario ha encendido o apagado el dispositivo. Esta función recibe la ID del dispositivo y el estado actual y devuelve como verdadero para indicar que el cambio de estado fue exitoso.

```
100     void handleFlipSwitches() {
101       unsigned long actualMillis = millis();
102       for (auto &flipSwitch : flipSwitches) {
103         unsigned long lastFlipSwitchChange =
flipSwitch.second.lastFlipSwitchChange;
104
105         if (actualMillis - lastFlipSwitchChange > DEBOUNCE_TIME) {
106
107           int flipSwitchPIN = flipSwitch.first;
108           bool lastFlipSwitchState =
flipSwitch.second.lastFlipSwitchState;
109           bool flipSwitchState = digitalRead(flipSwitchPIN);
110           if (flipSwitchState != lastFlipSwitchState) {
111     #ifdef TACTILE_BUTTON
112             if (flipSwitchState) {
113     #endif
114               flipSwitch.second.lastFlipSwitchChange = actualMillis;
115               String deviceId = flipSwitch.second.deviceId;
116               int relayPIN = devices[deviceId].relayPIN;
117               bool newRelayState = !digitalRead(relayPIN);
118               digitalWrite(relayPIN, newRelayState);
119
120               SinricProSwitch &mySwitch = SinricPro[deviceId];
121               mySwitch.sendPowerStateEvent(!newRelayState);
122     #ifdef TACTILE_BUTTON
123             }
124     #endif
125             flipSwitch.second.lastFlipSwitchState = flipSwitchState;
126           }
127         }
128       }
129     }
```

Por último, la función "`handleFlipSwitches()`" se encarga de manejar los interruptores de forma manual, no requiere confirmar de la conexión con la plataforma. Esta función verifica si ha pasado suficiente tiempo desde la última vez que se cambió el estado de un interruptor (para evitar cambios rápidos y no deseados), lee el estado actual del interruptor y, si ha cambiado, actualiza el estado del relé correspondiente. El programa incluye la programación necesaria para trabajar de manera local y en línea.

3.3.3.2.1 Interfaz web, Sinric Pro

Para la edición de la interfaz, una vez iniciada la sesión en la plataforma primero se crea una "Habitación", pulsar en "Habitaciones" y en "+ Agregar habitación", ver figura 3.43. La nueva habitación se configura con un nombre, una descripción y el lugar al que pertenece.

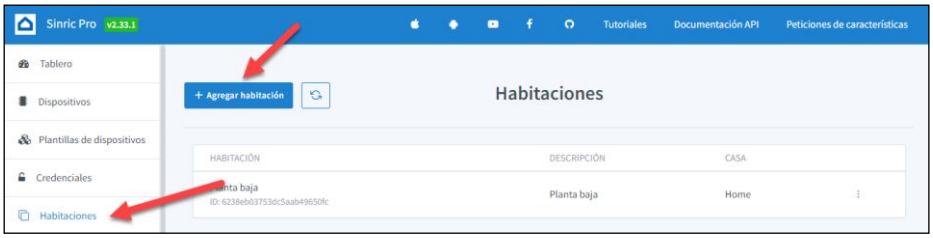


Figura 3.43. Agregar habitación, Sinric Pro.

Creada la habitación, en el apartado "Dispositivos", pulsar en el "+ Añadir dispositivos", para agregar un dispositivo, ver figura 3.44.

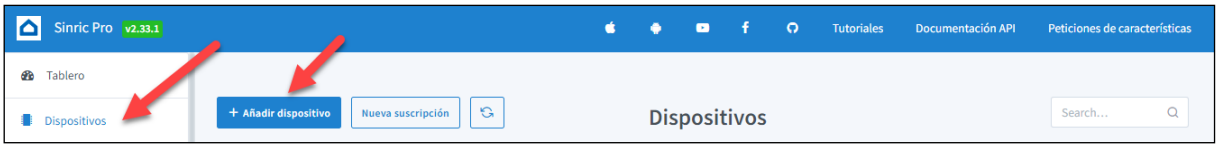


Figura 3.44. Crear dispositivos.

Este apartado permite configurar en nombre del dispositivo, descripción, tipo de dispositivo y en que habitación se encuentra, ver figura 3.45. Como configuración adicional permite añadir notificaciones para el cambio de estado, temporizadores y consumos eléctricos.

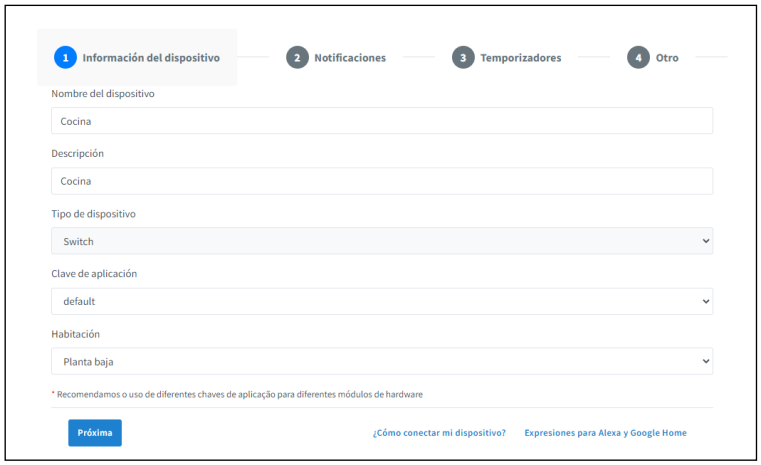


Figura 3.45. Configuración de dispositivo.

Por último, al pulsar en "Salvar" se crea el dispositivo, así como su ID que debe ser agregado en el programa para controlar este por medio de la plataforma IoT, ver figura 3.46.

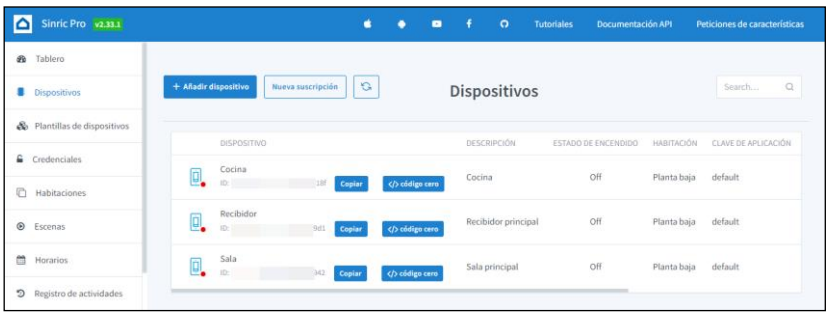


Figura 3.46. Dispositivos y sus respectivas ID.

Estos dispositivos pueden ser controlados desde la interfaz web por medio del apartado "Tablero" (ver figura 3.47.), o a través de un asistente virtual como Alexa o Google assistant, por medio de la integración de la API de Sinric Pro a estos.

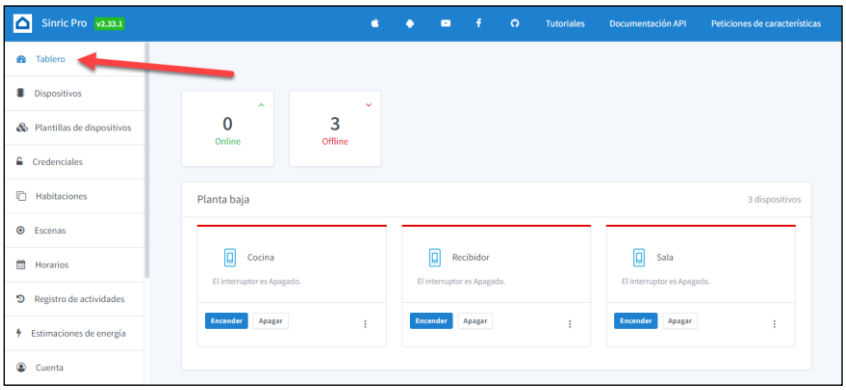


Figura 3.47. Iluminación, interfaz web, Sinric Pro.

3.3.3.2.2 Sinric Pro integración con Alexa y Google Assistant

Para integrar la función de control en los asistentes de Alexa[1] y Google[2], primero se descargan las aplicaciones respectivas de la tienda de aplicaciones, ver figura 3.48. (a) y (b), una vez instaladas, se realiza el registro de las cuentas respectivas, en la aplicación de Alexa si es la primera vez, se deberá habilitar la cuenta de correo electrónico para el uso en el ecosistema de Amazon, para el caso de Google Home el registro es mucho más rápido en dispositivos de Android ya que se usa la cuenta registrada en el dispositivo.

Actualmente la mayoría de los dispositivos Android ya cuentan con la aplicación de asistente[3] preinstalada por defecto ya que es un requisito integrado dentro de los servicios de Google, en caso de no estar instalada, se puede descargar para completar el entorno de Google.

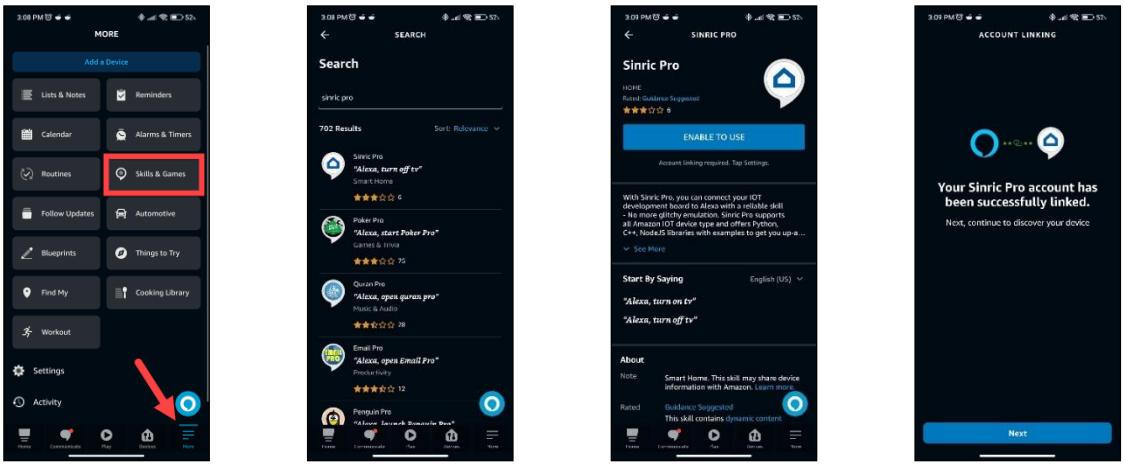
[1] https://play.google.com/store/apps/details?id=com.amazon.dee.app
[2] https://play.google.com/store/apps/details?id=com.google.android.apps.chromecast.app
[3] https://play.google.com/store/apps/details?id=com.google.android.apps.googleassistant

a) b)

Figura 3.48. App "Amazon Alexa" y "Google Home" en la tienda de aplicaciones.

Amazon Alexa

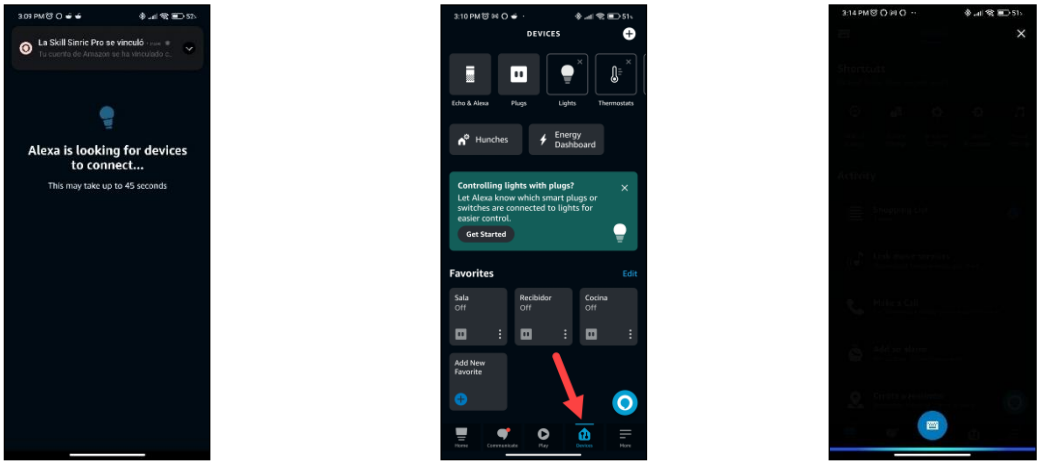
En la aplicación navegar hasta el apartado "Más" (menú inferior, apartado a la derecha, ver figura 3.49 (a)), pulsar en "Skills y juegos", con el icono de lupa buscar en las habilidades compatibles "Sinric Pro", ver figura 3.49 (b), pulsar en "HABILITAR", figura 3.49 (c), el asistente solicita el acceso a la cuenta de Sinric Pro, figura 3.49 (d).



a) b) c) d)

Figura 3.49. "Skill" Sinric Pro para Amazon Alexa.

Por último, al otorgar el acceso a la cuenta el mismo asistente busca los dispositivos que se encuentran en la plataforma y los agrega en el apartado de dispositivos, figuras 3.51. (a) y (b). Los dispositivos agregados pueden ser controlados desde los interruptores en la misma aplicación, por comandos de voz que reconoce el asistente, figura 3.50 (c), o dispositivos como altavoces "Echo DOT" registrados en la cuenta.

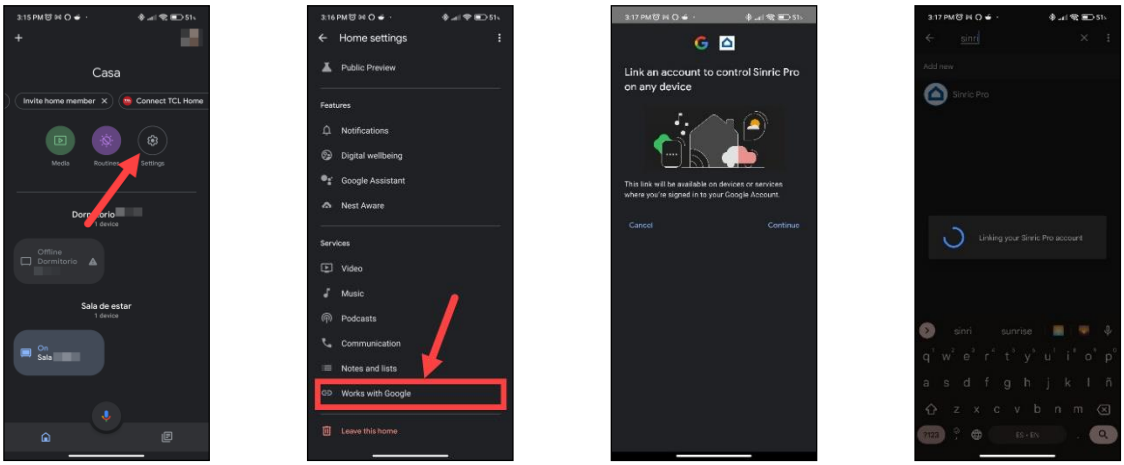


a) b) c)

Figura 3.50. Control de dispositivos Alexa.

Google Assistant

En la aplicación Google Home, una vez creado el perfil del hogar pulsar sobre el icono "Ajustes", ver figura 3.51 (a), navegar hasta "Funciona con Google" (apartado "Servicios", ver figura 3.51 (b)), con el icono de lupa buscar en los servicios compatibles "Sinric Pro", pulsar en "Continuar", figura 3.51 (c), el asistente solicita el acceso a la cuenta de Sinric Pro, figura 3.51 (d).

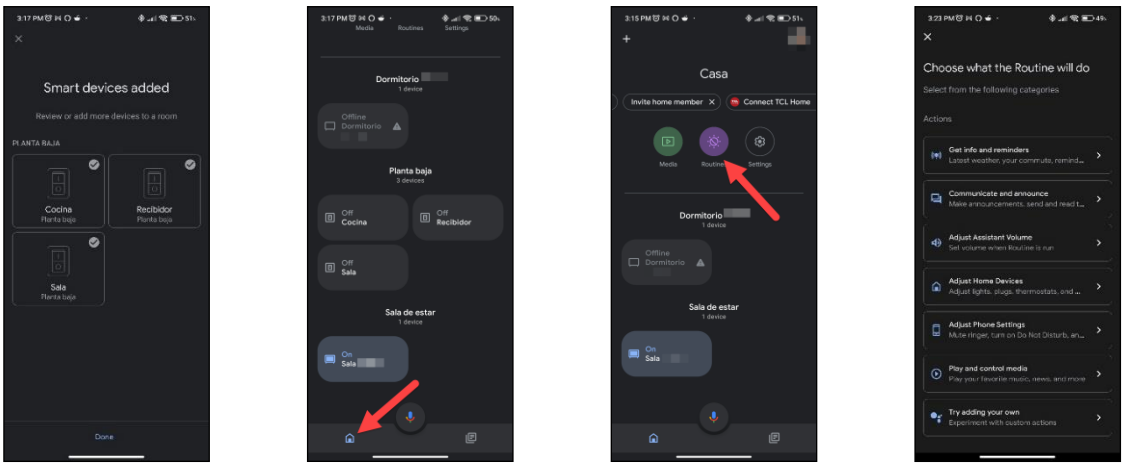


a) b) c) d)

Figura 3.51. Complemento para API Sinric Pro en Google Home.

Por último, al otorgar el acceso a la cuenta el mismo asistente busca los dispositivos que se encuentran en la plataforma y permite agregarlos individualmente en la pantalla principal donde se pueden ver todos los dispositivos que forman parte del perfil para ese hogar en particular, figuras 3.52 (a) y (b).

Una característica adicional con la que cuenta el asistente de Google es que, permite crear rutinas personalizadas para los dispositivos por medio de la aplicación de Google Home, estas pueden depender de otros dispositivos o acciones que realice el usuario, el menú de creación de rutinas está disponible en la pantalla principal, figura 3.52 (c), y permite configurar una gran cantidad de apartados, ver figura 3.52 (d).



a) b) c) d)

Figura 3.52. Dispositivos y configuración de rutinas.

Los dispositivos agregados pueden ser controlados desde los interruptores en la misma aplicación, por comandos de voz o rutinas que reconoce el asistente, ver figura 3.53, también a través de dispositivos como altavoces “Google Nest” registrados en el perfil del hogar.

Figura 3.53. Control de dispositivos por medio del asistente de Google.

3.3.3.3 Pruebas y resultados plataforma Sinric Pro

Para esta implementación, se decidió prescindir de los sensores BH1750 y PIR debido a que la plataforma solo permite el uso de alertas y notificaciones en planes de pago, y también se tiene el límite de tres dispositivos conectados.

Como se ha mencionado anteriormente, se realizaron pruebas para integrar dispositivos a los asistentes virtuales de Google y Amazon, así como para permitir que la tarjeta de desarrollo no dependa completamente de la información recibida del servidor de la plataforma y se pueda controlar de forma local. De esta manera, también se puede enviar el estado de la iluminación una vez restablecida la comunicación con la plataforma.

Como se puede observar en las figuras 3.54, este proyecto no cuenta con una aplicación móvil. En su lugar, la API de Sinric Pro ejecuta directamente las acciones en los asistentes donde ha sido asociada, lo que permite encender o apagar los dispositivos a través de comandos de voz (ver figura 3.53), interruptores virtuales en las respectivas aplicaciones de asistencia o mediante los interruptores físicos en el alambrado.

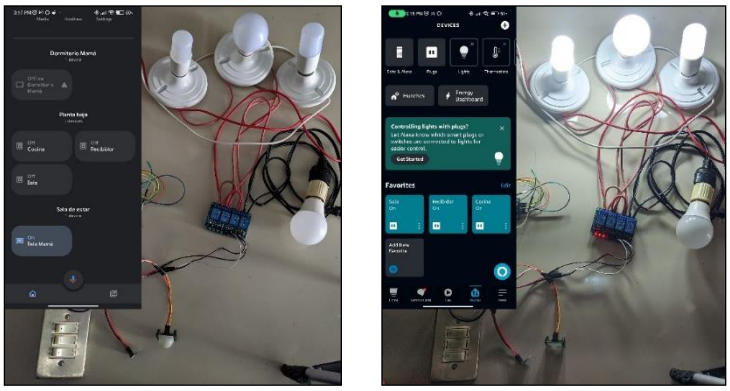


Figura 3.54. Control de iluminación en asistentes virtuales por API de Sinric Pro.

Sinric Pro cuenta con una interfaz web (ver figura 3.55) que no permite demasiada personalización, como la plataforma Blynk. No obstante, también permite el control de los dispositivos sin el uso de un smartphone.

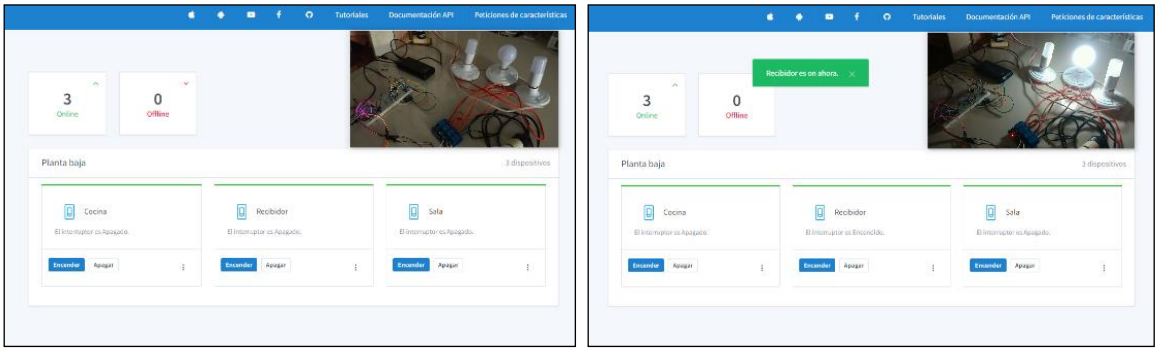


Figura 3.55. Control de iluminación con interfaz web Sinric Pro.

El resultado del experimento es satisfactorio. Es posible controlar las luminarias mediante comandos de voz o rutinas configuradas a través de los asistentes, lo que aumenta en gran medida el confort y la comodidad. Este control no se limita a la conexión a Internet, ya que los dispositivos también pueden ser manipulados manualmente de manera local.

3.3.4 Diagrama de bloques plataforma ESP RainMaker

Ahora se presenta y describe (figura 3.56) el diagrama de bloques en donde se representan las etapas que conforman a cada uno de los módulos de este apartado[1].

[1] Debido a la similitud, se ha omitido la descripción de aquellos elementos expuestos previamente en diagrama de bloques correspondiente a la plataforma Blynk.

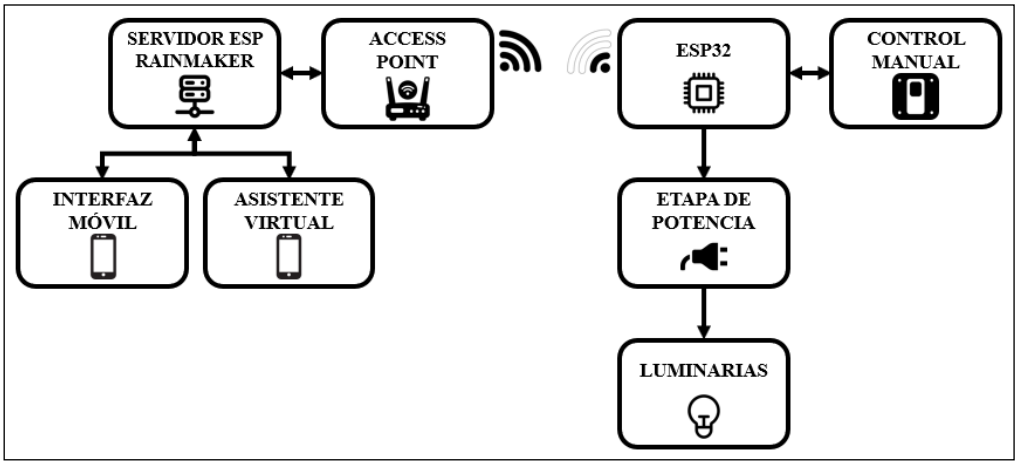


Figura 3.56. Diagrama de bloques, plataforma ESP RainMaker.

- Interfaz móvil, asistente virtual: a través de, es posible realizar el monitoreo y control en la plataforma ESP RainMaker. La plataforma genera su propia interfaz por medio de la aplicación móvil en base a la programación de la tarjeta de desarrollo, y cuenta con complemento para usar con asistentes virtuales.
- Servidor: cuenta con la infraestructura que permite administrar las configuraciones, así como visualizar la interfaz móvil de la plataforma, esta se encuentra potenciada por los servicios de AWS para IoT (Amazon Web Services para Internet de las Cosas), lo cual mejora las prestaciones en cuanto a escalabilidad.

3.3.4.1 Diagrama de flujo ESP RainMaker

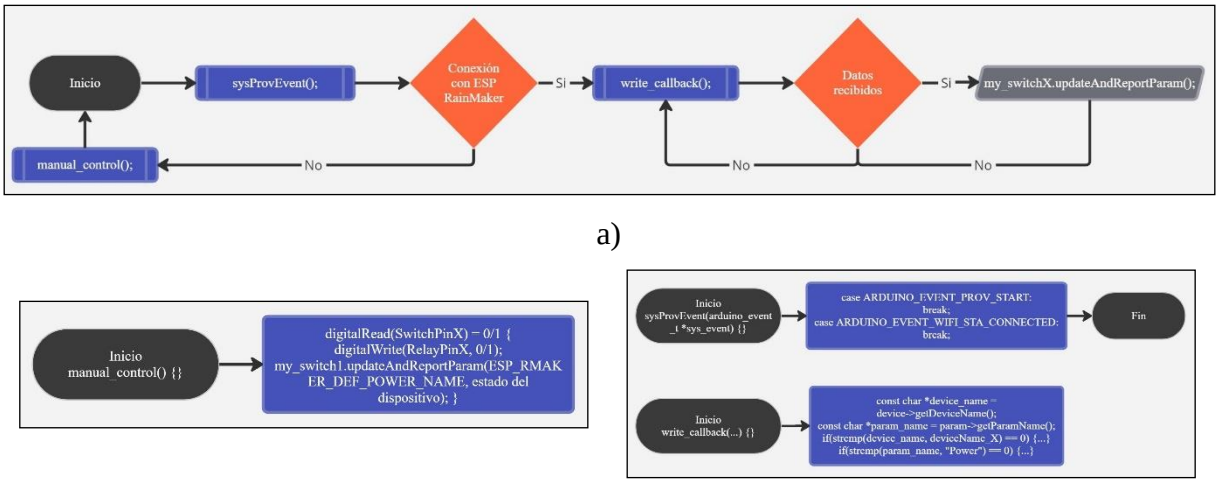


Figura 3.57. a) Diagrama de flujo, programa principal, b) Subrutinas de ejecución local y en línea, c) Subrutinas de ejecución inicial y configuración.

3.3.4.2 Implementación en el servicio ESP RainMaker

La aplicación ESP RainMaker utiliza el servicio de AWS para IoT (Amazon Web Services Internet of Things) para dotar, en conjunto con las tarjetas ESP, control y monitoreo de los dispositivos conectados, como interruptores, actuadores, sensores, entre otros, a través de una aplicación móvil. La función principal de este programa es controlar el estado de cuatro relés y cuatro interruptores conectados a una tarjeta ESP32, mediante conexión Wifi y el servicio AWS IoT. El programa también incluye el soporte para la configuración inicial de la tarjeta ESP32 y a través de la aplicación móvil ingresar las credenciales de conexión (provisioning). El programa prescinde del uso del sensor de presencia (PIR) y la luminaria automática remplazando esa por el uso de un interruptor virtual.

```
46      void sysProvEvent(arduino_event_t *sys_event) {
47          switch (sys_event->event_id) {
48              case ARDUINO_EVENT_PROV_START:
49      #if CONFIG_IDF_TARGET_ESP32
50              Serial.printf("\nProvisioning Started with name \"%s\" and
PoP \"%s\" on BLE\n", service_name, pop);
51              printQR(service_name, pop, "ble");
52      #else
53              Serial.printf("\nProvisioning Started with name \"%s\" and
PoP \"%s\" on SoftAP\n", service_name, pop);
54              printQR(service_name, pop, "softap");
55      #endif
56              break;
57              case ARDUINO_EVENT_WIFI_STA_CONNECTED:
58              Serial.printf("\nConnected to Wifi!\n");
59              digitalWrite(wifiLed, true);
60              break;
61          }
62      }
```

Función "`sysProvEvent()`": Esta función maneja los eventos del sistema que ocurren durante la conexión y configuración del dispositivo ESP32. Para realizar la conexión utiliza un switch para manejar dos eventos del sistema: "`ARDUINO_EVENT_PROV_START`", que se produce cuando se inicia el proceso de la biblioteca "`WifiProv.h`" y así realizar la configuración de la conexión Wifi por medio de los modos Bluetooth Low Energy (BLE) o Access Point (AP), habilitando uno u otro para sincronizar la tarjeta en la aplicación móvil. Imprime el código QR para realiza la configuración inicial y demás mensajes relativos a esta por el monitor serie del IDE Arduino. Realizada la configuración inicial o en conexiones subsecuentes se produce el evento "`ARDUINO_EVENT_WIFI_STA_CONNECTED`", cuando se conecta a la red Wifi, activa el LED indicador, una vez se ha validado las credenciales y se ha realizado la conexión.

```
64      void write_callback(Device *device, Param *param, const
param_val_t val, void *priv_data, write_ctx_t *ctx) {
65          const char *device_name = device->getDeviceName();
66          const char *param_name = param->getParamName();
67
68          if(strcmp(device_name, deviceName_1) == 0) {
69
```

```
70              Serial.printf("Lightbulb = %s\n", val.val.b? "true" :
"false");
71
72              if(strcmp(param_name, "Power") == 0) {
73                  Serial.printf("Received value = %s for %s - %s\n",
val.val.b? "true" : "false", device_name, param_name);
74                toggleState_1 = val.val.b;
75                (toggleState_1 == false) ? digitalWrite(RelayPin1, HIGH) :
digitalWrite(RelayPin1, LOW);
76                param->updateAndReport(val);
77              }...
```

La función "`write_callback()`", se encarga de actualizar el estado de los dispositivos e informar a la plataforma sobre los cambios. La función se llama cuando se recibe una solicitud de cambio de estado para alguno de los dispositivos. Primero, se verifica qué dispositivo y parámetro se están actualizando. Luego, se actualiza el estado del relé correspondiente, así como también el estado en la plataforma y con fines de depuración se imprime un mensaje en el monitor serie.

```
114     void Manual_control() {
115       if (digitalRead(SwitchPin1) == LOW && SwitchState_1 == LOW) {
116         digitalWrite(RelayPin1, LOW);
117         toggleState_1 = 1;
118         SwitchState_1 = HIGH;
119         my_switch1.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_1);
120         Serial.println("Switch-1 on");
121       }
122       if (digitalRead(SwitchPin1) == HIGH && SwitchState_1 == HIGH) {
123         digitalWrite(RelayPin1, HIGH);
124         toggleState_1 = 0;
125         SwitchState_1 = LOW;
126         my_switch1.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_1);
127         Serial.println("Switch-1 off");
128       }...
```

`Manual_control()`, esta función maneja los eventos de escritura de los dispositivos conectados de manera local. Lee las entradas digitales que se han definido para los switches y actualiza el estado del dispositivo correspondiente escribiendo el valor de salida en el pin de salida para el relé asignado. También envía una respuesta al servicio AWS IoT para notificar el cambio de estado del dispositivo, cuando este es manipulado de forma local.

En cuanto a las líneas de código sobre estas funciones, se puede detallar lo siguiente:

Las líneas 1 a 3 importan las librerías necesarias para el funcionamiento del programa: "`RMaker.h`", "`Wifi.h`" y "`WifiProv.h`".

Las líneas 5 y 6 definen dos variables que se usan en el proceso de aprovisionamiento (provisioning, adquisición de credenciales de conexión por medio de aplicación móvil).

Las líneas 9 a 12 definen los nombres de los dispositivos que se conectan a los relés.

Las líneas 15 a 23 definen las GPIO a las que se conectan los relés y los interruptores.

Las líneas 25 y 26 definen dos GPIO adicionales: LED para indicar la conexión Wifi y un GPIO para el reinicio de las configuraciones del dispositivo.

Las líneas 29 a 38 definen las variables de estado para los relés (toggleState_X) e interruptores (SwitchState_X), que se utilizan para mantener un registro del estado actual de cada dispositivo.

Las líneas 41 a 44 definen los objetos "Switch" como dispositivo estándar de control en el framework de la plataforma que representan a cada uno de los interruptores conectados a los relés.

El programa incluye la programación necesaria para trabajar de manera local y en línea.

3.3.4.2.1 Aplicación móvil ESP RainMaker, interfaz e integración con Alexa y Google Assistant

Integrar la función de control por medio de la plataforma ESP RainMaker en los asistentes de Alexa y Google, requiere la descarga, instalación y configuración de la aplicación[1] que se puede obtener de la respectiva tienda de aplicaciones, ver figura 3.58. Esta permite registro e inicio de sesión de forma rápida con cuentas GitHub, Google o via correo electrónico, así como también realizar el registro de forma tradicional por medio de correo electrónico. La aplicación posee una integración estrecha con el programa cargado en la tarjeta de desarrollo. En la programación, por medio del framework que dispone la API de la plataforma y donde se indica el tipo de dispositivo que será controlado. Los elementos de la interfaz como interruptores, slides y demás widgets se generan de forma automática en base a este framework y son colocados en la pantalla principal de la aplicación, ver figura 3.xx (d), y pueden ser controlados desde esta.

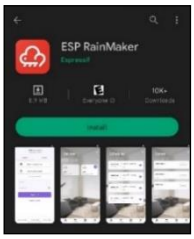
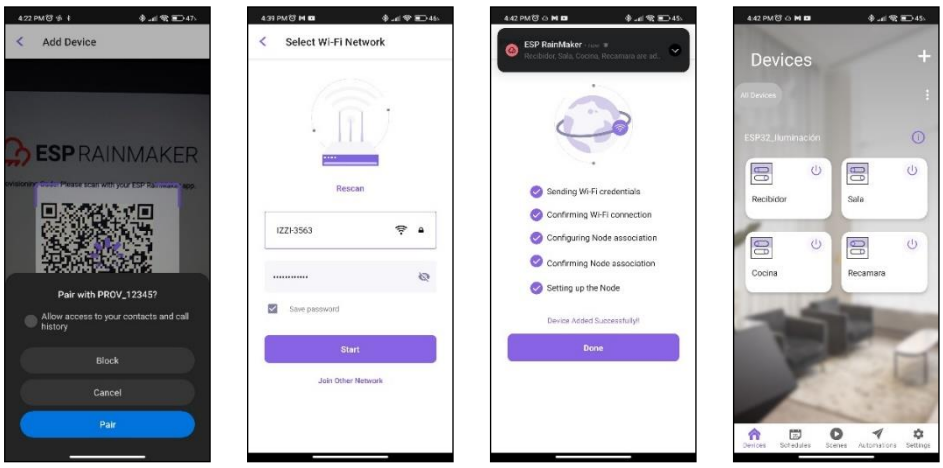


Figura 3.58. App "ESP RainMaker" en tienda de aplicaciones.

Una vez se ha iniciado sesión en la aplicación, se asocia a la tarjeta de desarrollo por medio del código QR generado por la misma en el monitor serie del IDE Arduino (nota: a través del motor serie comúnmente se genera de forma distorsionada el código QR, por ello el mismo programa genera un enlace que se puede abrir desde cualquier navegador para visualizar de forma correcta el código QR), ver figura 3.59. (a), se ingresan las credenciales de acceso para la conexión Wifi, figuras 3.59. (b), se conecta la tarjeta a la red y a la plataforma, figura 3.59. (c).

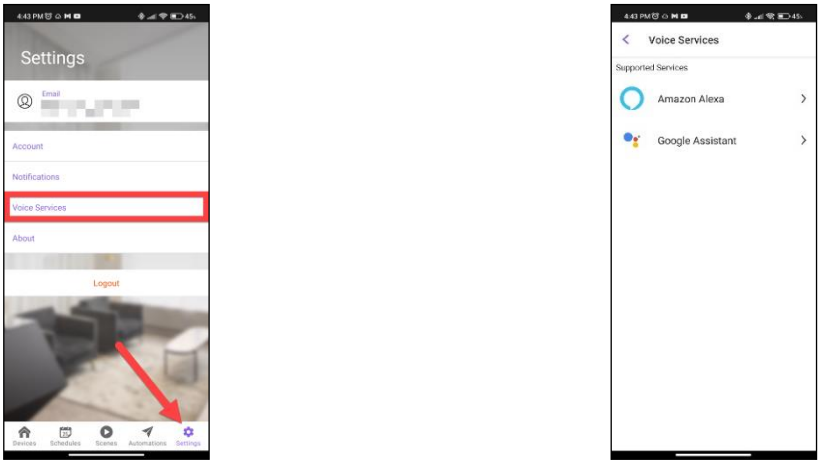
[1] https://play.google.com/store/apps/details?id=com.espressif.rainmaker



a) b) c) d)

Figura 3.59. Asociación de app con la tarjeta, programa y configuración de conexión Wifi.

Para integrar ESP RainMaker a los asistentes virtuales, navegar al apartado “Settings” (menú inferior, derecha, ver figura 3.60. (a)), seleccionar la opción “Voice Services”, la aplicación muestra los servicios de asistencia por voz que son compatibles, ver figura 3.60. (b), así como las instrucciones para integrar de forma rápida en cada servicio.



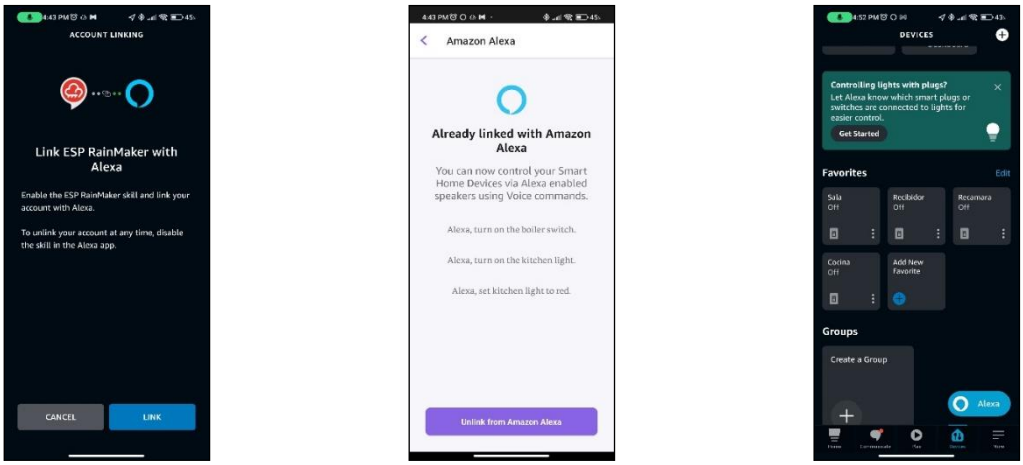
a) b)

Figura 3.60. Servicios de voz disponibles en ESP RainMaker.

Amazon Alexa

Para el asistente de Amazon Alexa, al pulsar sobre la opción en la aplicación ESP RainMaker se llama a la aplicación del asistente de Alexa y se solicita la asociación de la “Skill” para el asistente, ver figura 3.61. (a), una vez otorgado el acceso, ver figura 3.61. (b), el asistente busca los dispositivos que se encuentran en la plataforma y los agrega en el apartado de dispositivos, figuras 3.61. (c). Los dispositivos agregados pueden ser controlados desde los interruptores en

la aplicación del asistente, por comandos de voz, o dispositivos como altavoces “Echo DOT” registrados en la cuenta.



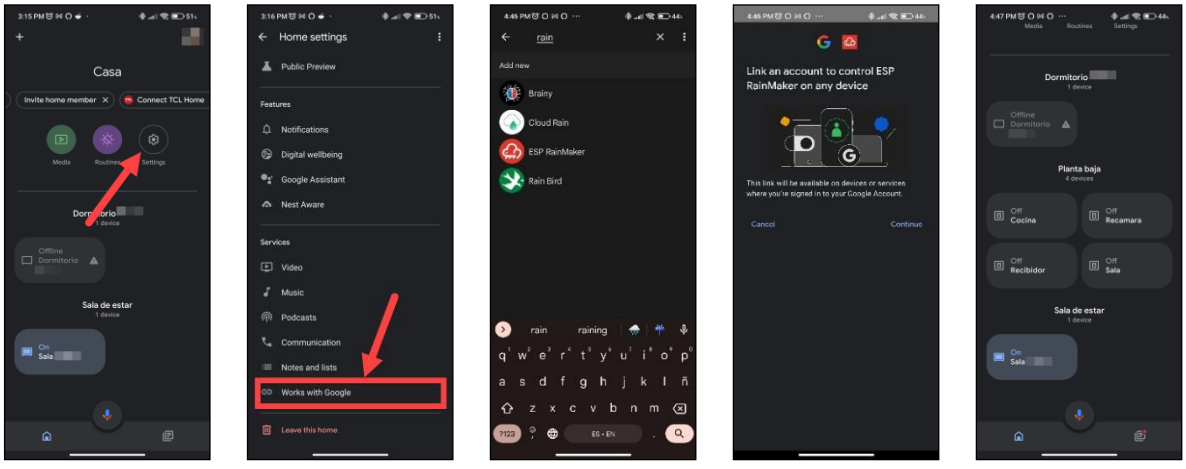
a) b) c)

Figura 3.61. “Skill” ESP RainMaker para Amazon Alexa.

Google Assistant

Para el asistente de Google, en el perfil del hogar pulsar sobre el icono “Ajustes”, ver figura 3.62. (a), navegar hasta “Funciona con Google” (apartado “Servicios”, ver figura 3.62. (b)), con el icono de lupa buscar en los servicios compatibles “ESP RainMaker”, figura 3.62. (c), pulsar en “Continuar”, figura 3.62. (d), el asistente solicita el acceso a la cuenta de ESP RainMaker y asocia ambas plataformas.

Por último, al otorgar el acceso a la cuenta el mismo asistente busca los dispositivos que se encuentran en la plataforma y permite agregarlos individualmente en la pantalla principal donde se pueden ver todos los dispositivos que forman parte del perfil para ese hogar en particular, figura 3.62. (e).



a) b) c) d) e)

Figura 3.62. Complemento ESP RainMaker para Google Home.

Como ya se ha hecho mención la aplicación de Google Home permite crear rutinas personalizadas para el asistente y los dispositivos asociados, estas pueden depender de otros dispositivos o acciones que realice el usuario. Los dispositivos agregados pueden ser controlados desde los interruptores en la misma aplicación, por comandos de voz o rutinas, así como a través de dispositivos como altavoces "Google Nest" registrados en el perfil del hogar.

3.3.4.3 Pruebas y resultados plataforma ESP RainMaker

Para la presente implementación se ha tomado la decisión de prescindir de los sensores BH1750 y PIR, ya que la plataforma aún se encuentra en desarrollo y, en el momento de las pruebas, no permite el uso de alertas y notificaciones. Además, actualmente la plataforma limita a cuatro dispositivos por usuario, ya que también ofrece planes de pago.

Se han realizado pruebas para integrar los dispositivos con los asistentes virtuales de Google y Amazon, así como para garantizar que la tarjeta de desarrollo no dependa completamente de la información recibida del servidor de la plataforma y se pueda controlar los dispositivos de manera local. De esta manera, se puede enviar el estado de la iluminación una vez restablecida la comunicación con la plataforma.

Cabe mencionar que, aunque este proyecto cuenta con una aplicación móvil, la API de ESP RainMaker se ejecuta directamente en base a las acciones en los asistentes donde ha sido asociada. Los dispositivos se pueden encender o apagar por medio de comandos de voz, interruptores virtuales en las respectivas aplicaciones de asistencia o en la propia aplicación de la plataforma, tal y como se puede observar en las figuras 3.59. y 3.63. También es posible controlar los dispositivos de manera manual a través de los interruptores físicos en el alambrado.

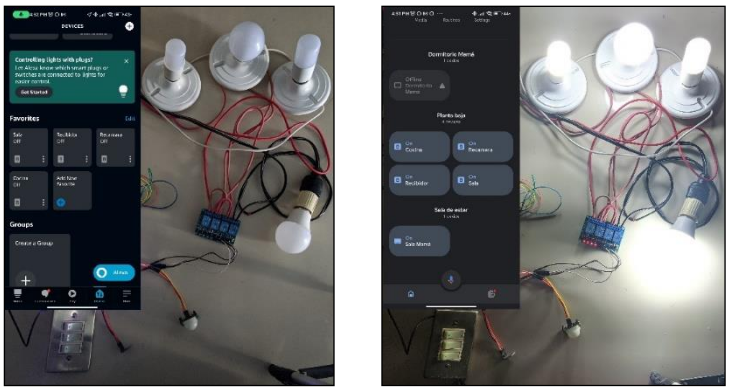


Figura 3.63. Control de iluminación asistentes virtuales por medio de API ESP RainMaker.

ESP RainMaker cuenta con una interfaz móvil, como se puede ver en la figura 3.59, que no permite personalización como la plataforma Blynk ya que se genera de forma automática en base al programa en la tarjeta de desarrollo. Esta aplicación admite la configuración de credenciales de conexión inalámbrica similar a como la aplicación de Blynk sin la necesidad de programar la tarjeta de desarrollo en cada ocasión, como ya se ha explicado previamente.

El resultado del experimento es el esperado, ya que se puede realizar el control de luminarias a través de comandos de voz o rutinas configuradas por medio de los asistentes, lo que proporciona confort y comodidad en gran medida. Además, este control extendido no requiere de la conexión de los dispositivos a internet, ya que también pueden ser manipulados de manera manual local. Un posible trabajo futuro sería una mejora estética o embalaje para facilitar la instalación del usuario final, así como experimentar con una mayor cantidad y variedad de dispositivos conectados por medio de la plataforma y conocer con más detalle las capacidades que esta ofrece, ya que ha sido desarrollada y diseñada para las tarjetas con ESP.

3.4 Control de potencia AC (en línea)

Para este experimento se ha ensamblado un circuito electrónico basado en un circuito cruce por cero o dimmer (atenuador), este permite regular la potencia entregada a una lámpara. Un dimmer, de forma general limita la potencia entregada a la o las cargas (lámpara) generalmente acortando la onda senoidal que proviene de la red eléctrica, para más información consultar el capítulo 2.5.2.

El programa es un controlador de dimmer para tarjetas ESP32 a través de la plataforma de IoT Blynk. La función principal del programa es permitir al usuario controlar el brillo de una luz conectada a la tarjeta ESP32 mediante la aplicación web o móvil de Blynk.

3.4.1 Diagrama de bloques

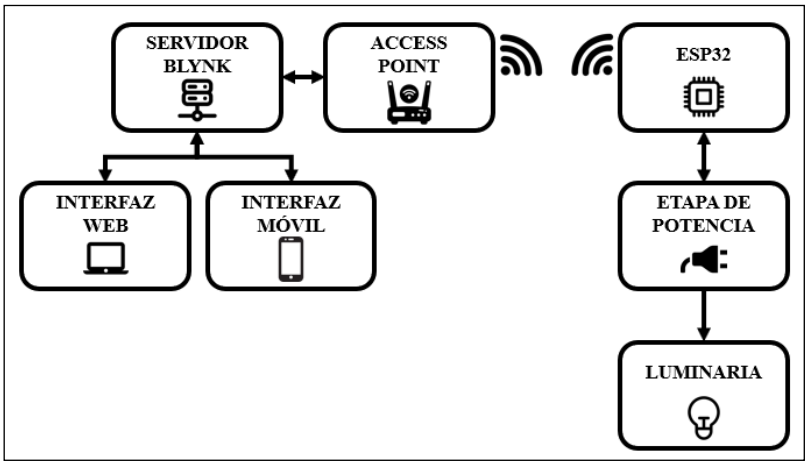


Figura 3.64. Diagrama de bloques, "Control de potencia AC".

- Interfaz web y móvil: a través de, es posible realizar el monitoreo y control en la plataforma Blynk, cabe recalcar que las interfaces son independientes en la creación de su diseño, ambas comparten como base canales de datos, lo que permite sincronizar y los cambios que se realicen en una interfaz sean reflejados en la otra.
- Servidor Blynk: permite administrar las configuraciones, así como visualizar la interfaz web de la plataforma, es en este dónde se crean los canales de datos que servirán como puente de comunicaciones entre las interfaces, el mismo servidor, el access point y la tarjeta ESP32.

- Access Point: es el dispositivo que permite establecer un canal de comunicación inalámbrica entre la tarjeta de desarrollo y la plataforma IoT por medio de internet. Este canal normalmente se encuentra cifrado y para obtener la conexión es necesario proporcionar al dispositivo que se conectara las credenciales de acceso (SSID y contraseña).
- ESP32: es la tarjeta de desarrollo que incluye el microcontrolador ESP32, este ya integra un módulo y antenas para realizar la conexión de manera inalámbrica con dispositivos que permiten el acceso a internet. Por medio de esta se realizan todas las operaciones lógicas de comunicación y control, lo cual incluye la gestión de la etapa de potencia y la plataforma de control IoT.
- Etapa de potencia: es el circuito electrónico que permite controlar la energía que se requiere en los actuadores de acuerdo con las funcionalidades incorporadas en el sistema.
- Luminaria: es el dispositivo que se conecta inmediatamente después de la etapa de potencia, y que en este caso produce la iluminación de un área. Puede utilizarse cualquier luminaria que trabaje a 127 VAC, y el circuito de potencia otorga una potencia para luminarias de hasta 100W.

3.4.2 Diagrama esquemático

A continuación, se presenta el diagrama esquemático (figura 3.65) donde se muestra los componentes electrónicos que se emplean en este proyecto.

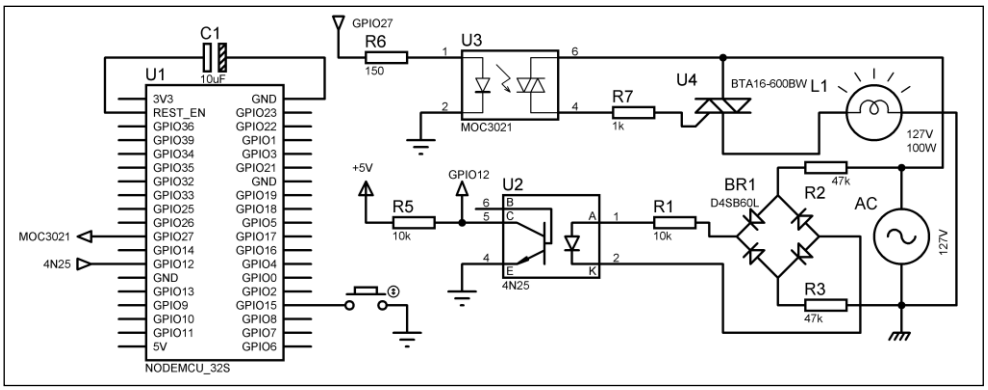


Figura 3.65. Diagrama esquemático de circuito cruce por cero.

Material necesario:

- 1 Tarjeta NodeMCU-32S.
- 1 Push button (opcional).
- 1 Capacitor 10µF a 50V (opcional).
- 1 Optoacoplador 4N25.
- 1 Optoacoplador MOC3021.
- 1 Triac BTA16.
- 1 Puente rectificador D4SB60L.
- 2 Resistores de 47kΩ a 1 watt.
- 2 Resistores de 10kΩ a 1/2 watt.

- 1 Resistor de 1kΩ a 1 watt.
- 1 Resistor de 150Ω a 1/2 watt.
- 2 Borneras.

En el diagrama esquemático presentado se puede observar la disposición de los elementos necesarios para conectar y controlar la potencia de una luminaria utilizando un microcontrolador como el ESP32. Es importante destacar que el capacitor y el push button son componentes opcionales, ya que la tarjeta NodeMCU-32S incluye un push button que puede ser utilizado para ambas funciones (programar la tarjeta y reiniciar las configuraciones para la conexión Wifi de la tarjeta si se mantiene pulsado por diez segundos luego de cargar la biblioteca de Blynk, consultar el capítulo 2.2.5 y 3.1.2.2).

Para poder controlar la potencia de una luminaria con un microcontrolador es necesario analizar la señal con la que ésta es energizada. El circuito descrito con detalle en el capítulo 2.5.2 permite al microcontrolador leer las variaciones de la frecuencia de la red eléctrica a través de un optoacoplador, y ajustar el ciclo de trabajo de ésta mediante otro optoacoplador y un TRIAC para así regular la potencia que se entrega a la luminaria. Esta configuración para el circuito de potencia permite conectar sin inconvenientes una luminaria de corriente alterna al microcontrolador ESP32 de hasta 100 watts.

Las señales provenientes de la red eléctrica a través del circuito de potencia son analizadas por medio del GPIO12, mientras que las señales de control tienen su salida por medio del GPIO27 de la tarjeta. Es importante destacar que este circuito debe ser energizado con 5V a través de una fuente externa, la cual también puede alimentar a la tarjeta de desarrollo a través del pin 5V (consultar los capítulos 2.2.5 y 2.5.4 para más detalles).

El ensamble del circuito, que se puede ver en la figura 3.66, se ha realizado en una placa PCB perforada para prototipos, asegurando de esta forma las conexiones y evitando cortos circuitos. Este circuito permite regular la potencia de corrientes y voltajes mayores a los que la tarjeta de desarrollo puede soportar.

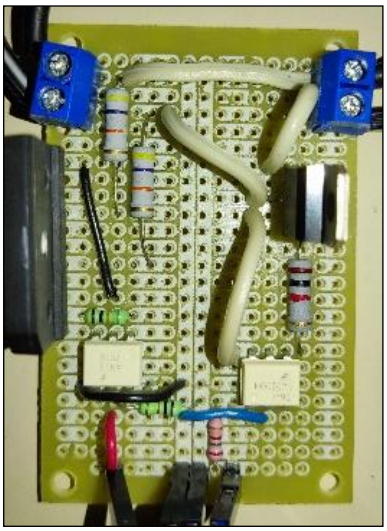
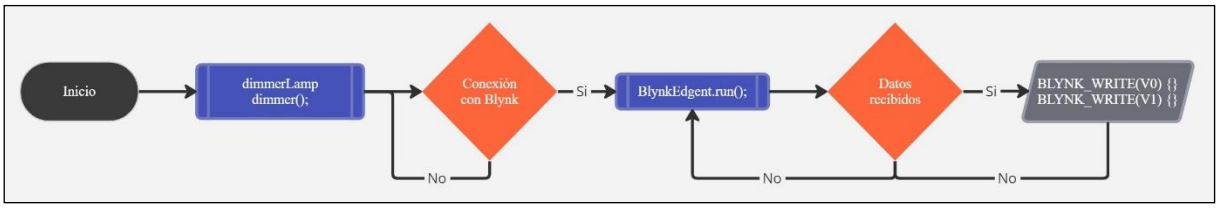
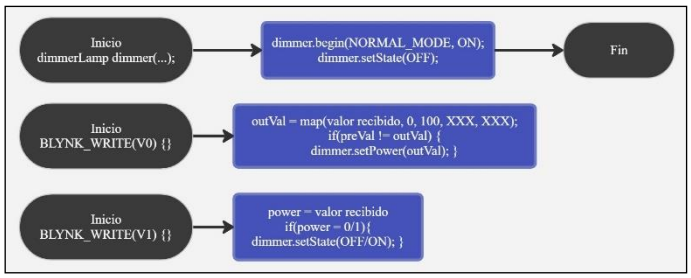


Figura 3.66. Montaje del circuito cruce por cero.

3.4.3 Diagrama de flujo



a)



b)

Figura 3.67. a) Diagrama de flujo, programa principal, b) Subrutinas de ejecución inicial, configuración y monitoreo.

Idealmente un dimmer está diseñado para un tipo de iluminación que está pensada para ser alimentada con una onda senoidal (corriente alterna), si se intenta regular con dimmers que alteran esa onda a cargas que no sean puramente resistivas o iluminación con electrónica adicional, pueden llegar a presentar irregularidades en la alimentación a esta y producir fallas no deseadas.

En este caso se ha seleccionado una lámpara incandescente. Este tipo de iluminación es completamente resistiva y por tanto "dimmable". En general podremos utilizar todas aquellas lámparas etiquetadas como dimmable.

Para el programa, es necesario la instalación de la biblioteca "RBDdimmer" la cual se puede obtener de su repositorio oficial en GitHub[1], en el repositorio del IDE Arduino no se encuentra disponible, descargar el archivo[2] en formato ZIP e instalar con alguno de los métodos del apartado 2.2.5 del capítulo anterior.

La forma en el ensamble del circuito permite el uso de esta biblioteca sin inconvenientes ya que el desarrollador de esta última comercializa un producto similar que cumple las mismas funciones, por tanto, la biblioteca es compatible con el ensamble a excepción del rango de potencia que puede ser controlado. Para conocer estos rangos es necesario un remplazo de componentes por aquellos en los que los valores se ajusten mejor o así también la calibración e

[1] https://github.com/RobotDynOfficial/RBDDimmer
[2] https://github.com/RobotDynOfficial/RBDDimmer/archive/refs/heads/master.zip

identificación de los límites de las señales de corte, esto puede hacerse por medio de un osciloscopio, y así fijar estos límites en la programación.

3.4.3.1 Inclusión de biblioteca RBDdimmer en Blynk Edgent

En el entorno de Arduino abrir el programa de ejemplo de la plataforma Blynk. Al comienzo del programa se deben incluir las credenciales que le permiten acceder y asociarse a la o las tarjetas de desarrollo con la plataforma, revisar tema 3.1.

En el programa destacan las líneas de código:

Las líneas "`#define BLYNK_TEMPLATE_ID, #define BLYNK_DEVICE_NAME`" definen el ID de la plantilla de Blynk y el nombre del dispositivo.

```
18      #include "BlynkEdgent.h"
19      #include <RBDdimmer.h>
20
21      #define outputPin 27
22      #define zerocross 12
23
24      dimmerLamp dimmer(outputPin, zerocross);
25
26      int outVal = 100;
27      int preVal = 0;
28      bool power = 0;
```

Las líneas "`#include`" incluyen el uso de las bibliotecas "`BlynkEdgent.h`" y "`RBDdimmer.h`".

Las líneas "`#define outputPin y #define zerocross`" definen los pines de salida del dimmer y el pin de interrupción de cruz cero respectivamente.

La línea "`dimmerLamp dimmer(outputPin, zerocross);`" instancia un objeto "dimmerLamp" y configura el controlador de dimmer con el pin de salida y el pin de interrupción de cruz cero.

La variable "`outVal`" servirá para almacenar el valor ajustado por la función `map` que se recibe del control deslizante en la plataforma, "`preVal`" esta variable no cambiará su valor que será siempre cero y servirá para comparar si el valor en "`outVal`" es diferente de cero y saber si existe cambio en el valor, por último "`power`" solo admite valores "0" y "1" y activa o desactiva el circuito.

```
30      void setup() {
31        Serial.begin(115200);
32        dimmer.begin(NORMAL_MODE, ON);
33        dimmer.setState(OFF);
34        delay(100);
35
36        BlynkEdgent.begin();
37      }
```

En `void setup()` iniciamos la comunicación serie, con `dimmer.begin(NORMAL_MODE, ON);` iniciamos el modo en el que operara el dimmer que puede ser NORMAL_MODE y

TOGGLE_MODE. En NORMAL_MODE el atenuador funciona en un valor definido de 0 a 100 (%) (entero) y el modo TOGGLE_MODE, realiza un cambio suave del valor de atenuación hacia arriba o hacia abajo en un rango definido. Este último implica el cambio de los valores de regulación mediante un temporizador de hardware, sin utilizar ciclos for(). El segundo parámetro puede ser ON/OFF y permite el uso o no de temporizadores para el atenuador. Con la función "`dimmer.setState()`" se establece el estado del dimmer, encendido/apagado de este. Se inicia la cabecera de la librería que facilita las configuraciones en la aplicación móvil.

```
39      BLYNK_WRITE(V0) {
40        outVal = map(param.asInt(), 0, 100, 13, 86);
41        if(preVal != outVal){
42        dimmer.setPower(outVal);
43        }
44      }
45
46      BLYNK_WRITE(V1) {
47        power = param.asInt();
48        if(power == 0){
49          delay(10);
50          dimmer.setState(OFF);
51        }
52
53        if(power == 1){
54          delay(10);
55          dimmer.setState(ON);
56        }
57      }
```

La función "`BLYNK_WRITE(V0)`" se llama cada vez que el usuario cambia el valor del widget deslizante en la aplicación Blynk. El valor del widget se mapea a un valor entre 13 y 86 (este valor debe ser calibrado dependiendo del circuito) y se almacena en la variable "`outVal`", que se utiliza para establecer la potencia de salida del dimmer por medio de la de la función "`dimmer.setPower()`".

La función "`BLYNK_WRITE(V1)`" se llama cada vez que el usuario cambia el estado del widget de interruptor en la aplicación Blynk. El estado del widget se utiliza para cambiar el estado de encendido/apagado del dimmer.

Por ultimo en la función "`void loop()`" ejecuta el bucle principal del programa y llama a la función "`BlynkEdgent.run()`" para manejar y mantener las comunicaciones con la aplicación Blynk.

3.4.4 Interfaz web

En la plataforma Blynk se ha creado una nueva plantilla que incluye dos flujos de datos en el apartado "Datastreams", tal como se muestra en la Figura 3.68. Estos flujos corresponden al ciclo de trabajo de cero al cien por ciento y al switch de encendido y apagado. Una vez creados los flujos de datos, se procede a implementar la interfaz web y guardar los cambios en la plantilla.

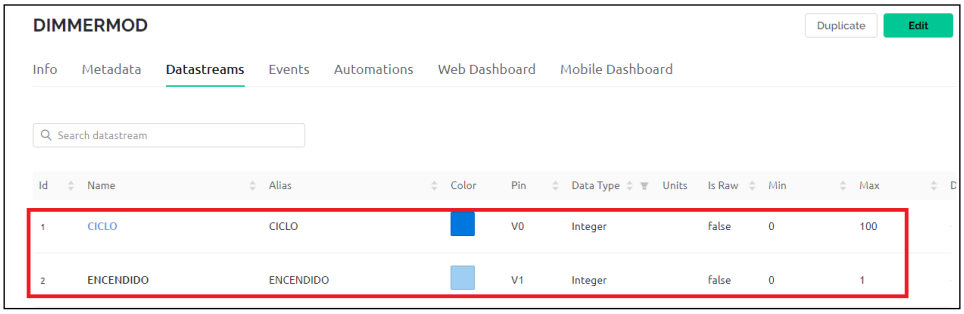


Figura 3.68. Datastreams para proyecto "Control de potencia AC".

Para la interfaz web, se ha seleccionado el control deslizante y el interruptor de los widgets disponibles de manera gratuita. La configuración del control deslizante requiere especificar el nombre del widget, el canal de datos (que va de 0 a 100), la velocidad de incremento (normalmente configurada en avance de uno en uno), el control de incremento fino (también en uno en uno) y la posición de visualización del valor.

Por último, en el interruptor se debe indicar el canal de datos (Datastream), el valor que se enviará en estado encendido y apagado, el nombre del widget, las etiquetas y la posición. La disposición y tamaño de los elementos de la interfaz son de elección libre. La Figura 3.69 muestra un ejemplo de cómo puede ser diseñada la interfaz.

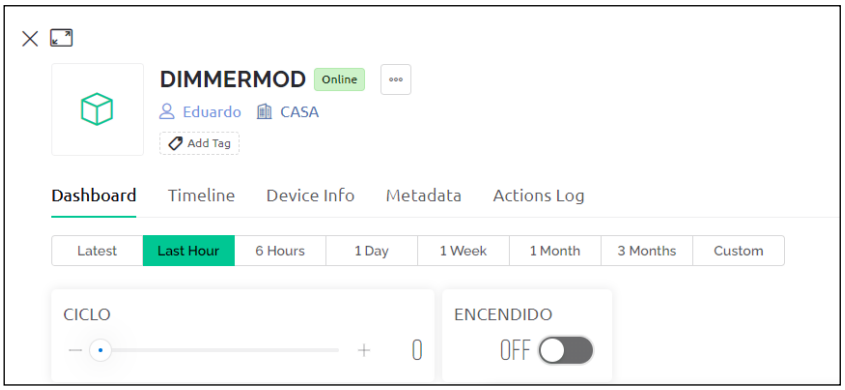


Figura 3.69. Interfaz web "Control de potencia" Blynk.

3.4.5 Interfaz móvil

En la interfaz móvil, como se puede ver en la Figura 3.70, se han colocado un interruptor y un control deslizante, cuya disposición y tamaño son de libre elección por parte del diseñador.

En cuanto a la configuración de los widgets, para el interruptor se debe especificar el nombre del widget, el canal de datos, el tipo de interruptor (switch o push) y color. En este proyecto en particular, se ha dado importancia a la selección del tipo de interruptor como switch. Opcionalmente, se puede indicar el valor del actuador o utilizar los valores proporcionados por el canal (automático), que viene configurado por defecto como tal.

Por último, el control deslizante se configura únicamente con la opción de canal de datos, y se permite personalizar el nombre del widget y el color de este elemento.

Figura 3.70. Interfaz móvil "Control de potencia" Blynk.

3.4.6 Pruebas y resultados

En el marco de este proyecto, se llevaron a cabo diversas pruebas con distintas bibliotecas, aunque no todas funcionaron de manera óptima en la plataforma originalmente elegida para el control de este proyecto. Debido a esto, se adaptaron los componentes disponibles para el ensamblaje, y se encontró que tanto el circuito como los propios componentes eran bastante similares al producto RBDDimmer de RobotDynOfficial para Arduino. Por lo tanto, al probar la biblioteca de este producto junto con el ensamblaje del proyecto, se comprobó que funcionaba correctamente en la plataforma Cayenne de myDevices.

Posteriormente, se decidió portar el programa a la plataforma Blynk para homogeneizar el presente trabajo, y así se desarrollaron los experimentos posteriores. Cabe señalar que este proyecto, así como los subsecuentes, carecen de integración con los asistentes virtuales de Google y Amazon. Esto no impide la asociación del programa con estos asistentes, pero como se mencionó previamente, era necesario el uso de la plataforma IFTTT para dotar a la plataforma Blynk de las capacidades de asistencia virtual. Desafortunadamente, al actualizarse ambas plataformas en sus términos de uso, la primera inhabilitó las funciones de asistentes virtuales para plataformas de terceros, mientras que la segunda ofrece esta API solo en planes de pago.

Una vez cargado el programa en la tarjeta y realizadas las configuraciones en la aplicación móvil, se obtiene un sistema que permite controlar la potencia en corriente alterna entregada a cargas resistivas. Como se muestra en la Figura 3.71, se puede observar el sistema cableado y en funcionamiento, el cual puede ser activado o desactivado mediante el botón/interruptor disponible en la interfaz móvil y web.

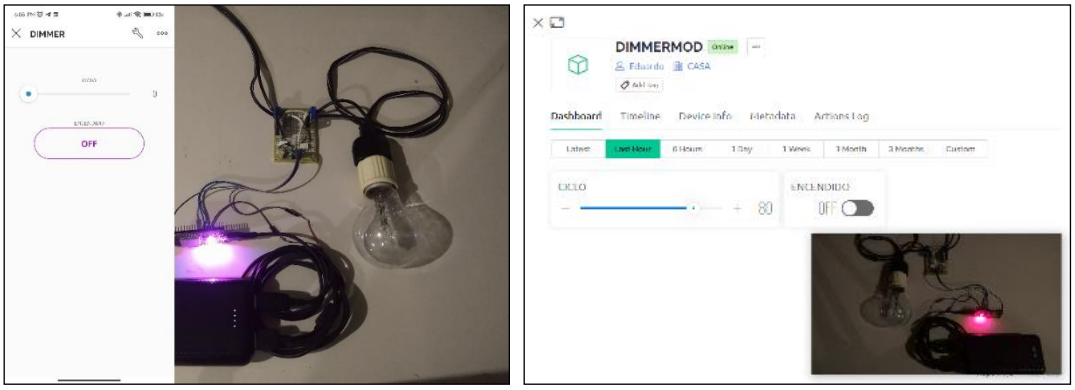
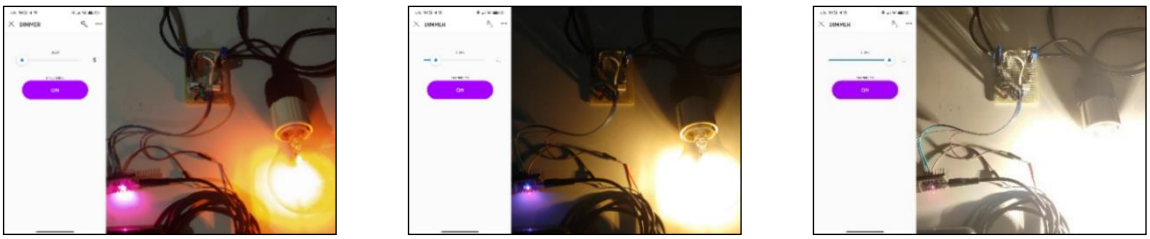


Figura 3.71. Dimmer inactivo, interfaz móvil (izquierda), web (derecha).

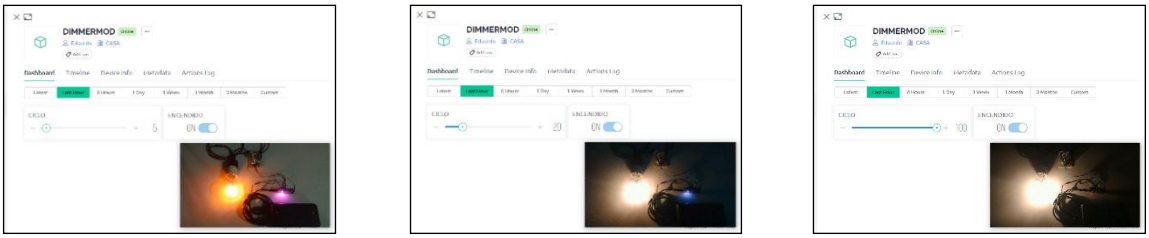
Circuito activo a una potencia del 5, 20 y 100 % figura 3.72. (a), (b), (c), respectivamente.



a) b) c)

Figura 3.72. Pruebas del circuito "Dimmer" interfaz móvil.

Así también se puede observar el funcionamiento por medio de la interfaz web. Ver figura 3.73. (a), (b), (c), donde el porcentaje es de 5, 20 y 100%, respectivamente.



a) b) c)

Figura 3.73. Pruebas del circuito "Dimmer" interfaz web.

El resultado del experimento es el esperado, se logró controlar la potencia de la iluminación por medio de detección de cruce por cero en implementación de un dimmer digital, el proyecto es extrapolable para su uso en circuitos de tipo resistivo como por ejemplo calentadores resistivos, incubadoras, motores de corriente alterna, entre otros. Esto a través de una conexión a internet que permite su operación remota, otorgando comodidad y confort al usuario. Como trabajo futuro, se podría considerar una mejora en la estética o embalaje del producto para facilitar su instalación por parte del usuario final.

3.5 Control y monitoreo de nivel de agua (en línea y local)

Este programa es un ejemplo de sketch para la tarjeta ESP32 para ser cargado por medio del IDE Arduino, que utiliza la biblioteca Blynk para conectarse al servicio IoT. El objetivo del programa es leer los niveles de agua en dos sensores y controlar el encendido y apagado de una bomba de agua para mantener un nivel de agua adecuado en un depósito-cisterna y en tinaco-contenedor. El programa puede operar de manera local y en línea, así como manual y de forma automática.

3.5.1 Diagrama de bloques

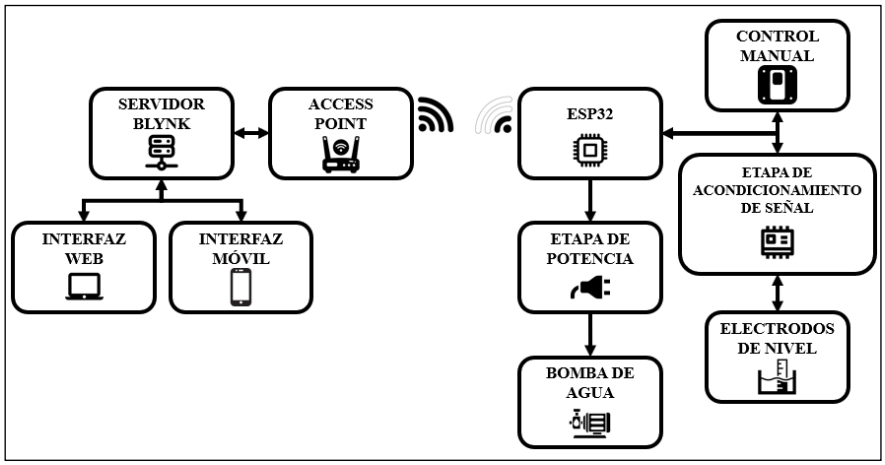


Figura 3.74. Diagrama de bloques, "Nivel de agua".

- Interfaz web y móvil: a través de, es posible realizar el monitoreo y control en la plataforma Blynk, cabe recalcar que las interfaces son independientes en la creación de su diseño, ambas comparten como base canales de datos, lo que permite sincronizar y los cambios que se realicen en una interfaz sean reflejados en la otra.
- Servidor Blynk: permite administrar las configuraciones, así como visualizar la interfaz web de la plataforma, es en este dónde se crean los canales de datos que servirán como puente de comunicaciones entre las interfaces, el mismo servidor, el access point y la tarjeta ESP32.
- Access Point: es el dispositivo que permite establecer un canal de comunicación inalámbrica entre la tarjeta de desarrollo y la plataforma IoT por medio de internet. Este canal normalmente se encuentra cifrado y para obtener la conexión es necesario proporcionar al dispositivo que se conectara las credenciales de acceso (SSID y contraseña).
- ESP32: es la tarjeta de desarrollo que incluye el microcontrolador ESP32, este ya integra un módulo y antenas para realizar la conexión de manera inalámbrica con dispositivos que permiten el acceso a internet. Por medio de esta se realizan todas las operaciones lógicas de comunicación y control, lo cual incluye la gestión de la etapa de potencia y la plataforma de control IoT.

- Control manual: en esta etapa se encuentra un interruptor que permite la activación de la etapa de potencia de manera local, junto a los electrodos de nivel se puede considerar una etapa semiautomática de llenado que no requiere de una conexión a internet para funcionar.
- Acondicionamiento de señal: etapa que contiene la electrónica para facilitar la interpretación de las lecturas provenientes de los electrodos de nivel, en el ESP32.
- Electrodos de nivel: sonda que permite calcular la cantidad de líquido contenido en un depósito.
- Etapa de potencia: es el circuito electrónico que permite controlar la energía que se requiere en los actuadores de acuerdo con las funcionalidades incorporadas en el sistema.
- Bomba de agua: es el dispositivo que se conecta inmediatamente después de la etapa de potencia, y que en este caso produce impulsa un líquido de un depósito a otro. En base a las pruebas del circuito de potencia, se determina que es posible conectar cualquier tipo de bomba doméstica en buenas condiciones.

3.5.2 Diagrama esquemático

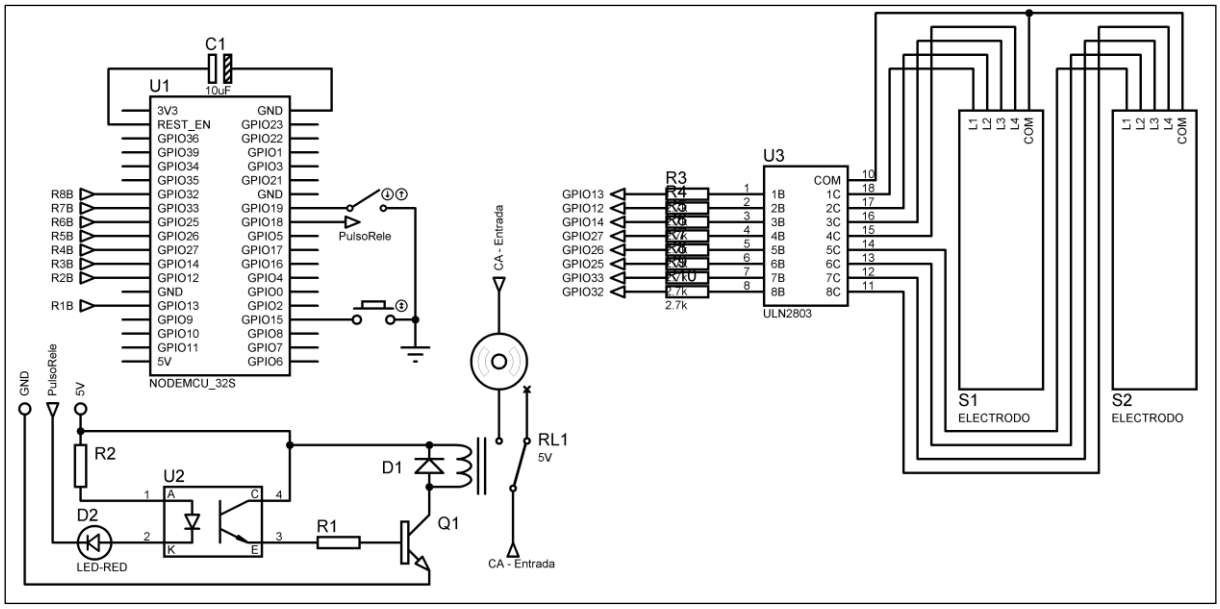


Figura 3.75. Diagrama esquemático de conexión "Nivel de agua".

Material necesario:

- 1 Tarjeta NodeMCU-32S.
- 1 Push button (opcional).
- 1 Capacitor 10µF a 50V (opcional).
- 1 Módulo de relevador SRD-5VDC-SL-C.
- 1 Circuito integrado ULN2803.
- 8 Resistores de 2.7kΩ a 1/2 watt.

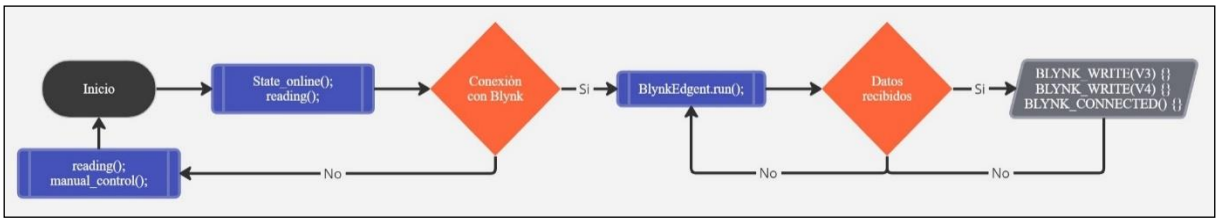
En el diagrama esquemático de la figura 3.75, en el cual se puede apreciar la disposición de los elementos para su conexión. Como ya se ha mencionado el capacitor y el push button son componentes opcionales, ya que la tarjeta NodeMCU-32S dispone de un push button integrado que puede utilizarse para ambas funciones: programar la tarjeta y reiniciar las configuraciones para la conexión Wifi de la tarjeta si se mantiene pulsado por diez segundos después de cargar la biblioteca de Blynk. Para más detalles, se puede consultar el capítulo 2.2.5 y 3.1.2.2.

Para conectar los electrodos de nivel a la tarjeta de desarrollo, es necesario un circuito integrado ULN2803 y una serie de resistores de acople que ayudan a la tarjeta de desarrollo a interpretar las señales de las sondas. Estas sondas pueden ser de fabricación propia, pero también se pueden utilizar sondas comerciales. El circuito integrado se activa y desactiva en ciertos periodos de tiempo mediante programación en la tarjeta de desarrollo, lo que evita la electrolisis de las sondas y amplía considerablemente su vida útil. Para más información sobre este tema, se puede consultar el capítulo 2.3.4.

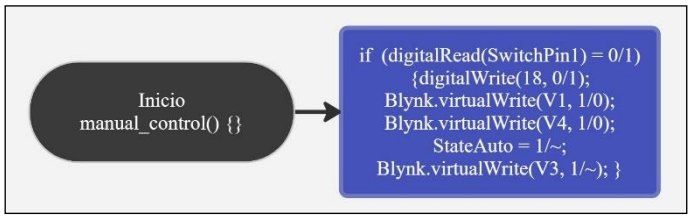
En la etapa de potencia para activar la bomba, se optó por utilizar un módulo relé, el cual se describe en capítulos anteriores. Basándose en las pruebas realizadas en el circuito de potencia, se determinó que es posible conectar cualquier tipo de bomba doméstica en buenas condiciones.

Para leer las señales de las sondas se utilizan los GPIO 13, 12, 14, 27, 26, 25, 33 y 32. La señal de control tiene su salida a través del GPIO18, y mediante el GPIO19 se conecta un interruptor que permite el control manual de la bomba y activa un modo semiautomático en caso de no haber conexión a internet o una falla en la comunicación de la tarjeta de desarrollo con la plataforma. Es importante tener en cuenta que este circuito debe ser energizado con 5V a través de una fuente externa, la cual también puede alimentar a la tarjeta de desarrollo mediante el pin 5V. Para obtener más detalles al respecto, se puede consultar los capítulos 2.2.5 y 2.5.4.

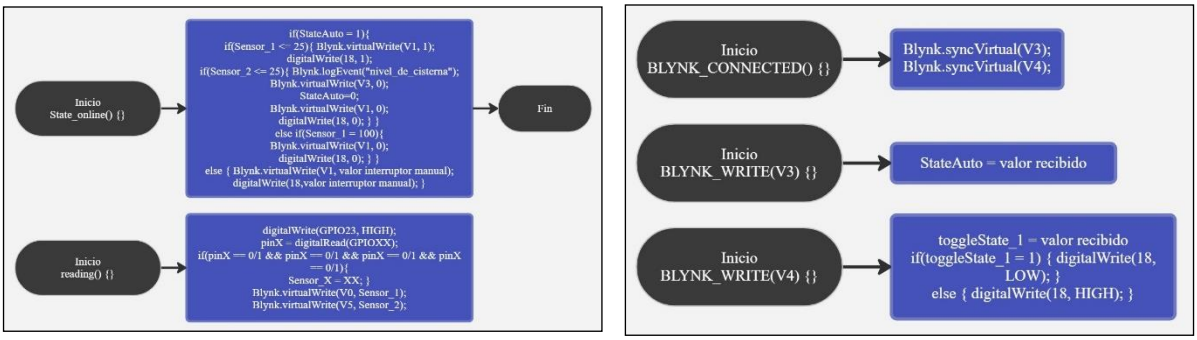
3.5.3 Diagrama de flujo



a)



b)



c)

Figura 3.76. a) Diagrama de flujo, programa principal, b) Subrutinas de ejecución local y en línea, c) Subrutinas de ejecución inicial y monitoreo.

El sketch tiene como objetivo controlar y monitorear el nivel de agua en una cisterna y en un tinaco mediante la lectura de dos sensores de nivel de agua, y controlar una bomba de agua a través de un botón físico (opción de control manual) o automática en la aplicación Blynk. El programa está diseñado para funcionar en una tarjeta ESP32 programada con el IDE Arduino y utiliza la biblioteca BlynkEdgent para configurar y conectar la tarjeta a la aplicación Blynk.

El programa comienza con la definición de varias constantes y variables globales utilizadas en el programa, incluyendo las identificaciones y nombres de dispositivo de la plantilla de Blynk, los pines GPIO utilizados por el programa, y las variables de nivel de agua y estado de control.

Las funciones principales del programa son:

```
64      void setup() {
65        Serial.begin(115200);
66        delay(100);
67        pinMode(GPIO13, INPUT_PULLUP);
68        pinMode(GPIO12, INPUT_PULLUP);
69        pinMode(GPIO14, INPUT_PULLUP);
70        pinMode(GPIO27, INPUT_PULLUP);
71        pinMode(GPIO26, INPUT_PULLUP);
72        pinMode(GPIO25, INPUT_PULLUP);
73        pinMode(GPIO33, INPUT_PULLUP);
74        pinMode(GPIO32, INPUT_PULLUP);
75        pinMode(GPIO23, OUTPUT);
76        pinMode(18, OUTPUT);
77        pinMode(SwitchPin1, INPUT_PULLUP);
78
79        BlynkEdgent.begin();
80      }
```

La función "`setup()`" contiene el código que se encarga de inicializar los pines de entrada y salida utilizados por el programa, configurar la conexión con la aplicación Blynk a través de BlynkEdgent, y establecer la velocidad de transmisión de datos en serie.

```
106     void reading() {
107       digitalWrite(GPIO23, HIGH);
108       delay(100);
109       pin15 = digitalRead(GPIO13);
110       pin2  = digitalRead(GPIO12);
111       pin0  = digitalRead(GPIO14);
112       pin4  = digitalRead(GPIO27);
113       pin16 = digitalRead(GPIO26);
114       pin17 = digitalRead(GPIO25);
115       pin5  = digitalRead(GPIO33);
116       pin18 = digitalRead(GPIO32);
117       delay(100);
118       digitalWrite(GPIO23, LOW);
119
120     if(pin15 == LOW && pin2 == LOW && pin0 == LOW && pin4 == LOW){
121         Sensor_1 =100;
122         }
123     if(pin15 == HIGH && pin2 == LOW && pin0 == LOW && pin4 == LOW){
124         Sensor_1 =75;
125         }
126     if(pin15 == HIGH && pin2 == HIGH && pin0 == LOW && pin4 == LOW){
127         Sensor_1 =50;
128         }
129     if(pin15 == HIGH && pin2 == HIGH && pin0 == HIGH && pin4 == LOW){
130         Sensor_1 =25;
131         }
132     if(pin15 == HIGH && pin2 == HIGH && pin0 == HIGH && pin4 == HIGH){
133         Sensor_1 =0;
134         }
135     if(pin16 == LOW && pin17 == LOW && pin5 == LOW && pin18 == LOW){
136         Sensor_2 =100;
137         }
138     if(pin16 == HIGH && pin17 == LOW && pin5 == LOW && pin18 == LOW){
139         Sensor_2 =75;
140         }
141     if(pin16 == HIGH && pin17 == HIGH && pin5 == LOW && pin18 == LOW){
142         Sensor_2 =50;
143         }
144     if(pin16 == HIGH && pin17 == HIGH && pin5 == HIGH && pin18 ==
LOW){
145         Sensor_2 =25;
146         }
147     if(pin16 == HIGH && pin17 == HIGH && pin5 == HIGH && pin18 ==
HIGH){
148         Sensor_2 =0;
149         }
150
151       Blynk.virtualWrite(V0, Sensor_1);
152       Blynk.virtualWrite(V5, Sensor_2);
153     }
```

La función "`reading()`" lee los valores de los sensores de nivel de agua y los almacena en las variables Sensor_1 y Sensor_2. Los valores leídos se utilizan para actualizar los valores en las

etiquetas virtuales V0 y V5 en la aplicación de Blynk. Dentro de esta función también se activa temporalmente el circuito integrado ULN2803 mientras se realizan las lecturas, con esto se trata de evitar en lo mayor posible electrolisis en los electrodos de lectura.

```
155     void manual_control() {
156       if (digitalRead(SwitchPin1) == LOW && SwitchState_1 == LOW) {
157         digitalWrite(18, LOW);
158         Blynk.virtualWrite(V1, HIGH);
159         Blynk.virtualWrite(V4, HIGH);
160         toggleState_1 = HIGH;
161         SwitchState_1 = HIGH;
162         StateAuto = HIGH;
163         Blynk.virtualWrite(V3, HIGH);
164       }
165       if (digitalRead(SwitchPin1) == HIGH && SwitchState_1 == HIGH) {
166         digitalWrite(18, HIGH);
167         Blynk.virtualWrite(V1, LOW);
168         Blynk.virtualWrite(V4, LOW);
169         toggleState_1 = LOW;
170         SwitchState_1 = LOW;
171       }
172     }
```

Esta función "`manual_control()`" se encarga de controlar la bomba de agua en función del estado del botón de control manual en la aplicación Blynk y el estado del botón físico en la tarjeta ESP32.

```
82      void State_online() {
83        if(StateAuto == 1){
84          if(Sensor_1 <= 25){
85          Blynk.virtualWrite(V1, HIGH);
86          digitalWrite(18, HIGH);
87          if(Sensor_2 <= 25){
88            Blynk.logEvent("nivel_de_cisterna");
89            Blynk.virtualWrite(V3, LOW);
90            StateAuto=0;
91            Blynk.virtualWrite(V1, LOW);
92            digitalWrite(18, LOW);
93          }
94        }
95          else if(Sensor_1 == 100){
96          Blynk.virtualWrite(V1, LOW);
97          digitalWrite(18, LOW);
98          }
99       }
100       else {
101         Blynk.virtualWrite(V1, toggleState_1);
102         digitalWrite(18, toggleState_1);
103       }
104     }
```

En "`State_online()`", esta función se encarga de controlar la bomba de agua en función del nivel de agua en la cisterna y el estado de control elegido (automático o manual) en la aplicación Blynk. Si el estado de control está en automático, la función verifica el nivel de agua y enciende o apaga la bomba de agua según sea necesario. Si el nivel de agua en el sensor 1 es inferior a 25, la bomba se enciende, y si el nivel de agua en el sensor 2 es inferior a 25, se escribe un evento en los registros de Blynk (Alerta), se cambia el estado del interruptor virtual V3 y se establece StateAuto en 0. Si el nivel de agua en el sensor 1 es igual a 100, la bomba se apaga. Si el estado de control está en manual (StateAuto es 0), la función simplemente activa la bomba de agua en función del estado del botón de control manual en la aplicación Blynk o el interruptor físico.

Algunas de las funciones complementarias incluyen:

En "`BLYNK_CONNECTED()`", donde: "`Blynk.syncVirtual(V3)`" y "`Blynk.syncVirtual(V4)`" sincronizan las variables virtuales con los valores actuales en la aplicación de Blynk.

En "`BLYNK_WRITE(V3)`", la línea StateAuto = param.asInt() actualiza el valor de StateAuto según el valor actual del interruptor virtual V3.

En "`BLYNK_WRITE(V4)`", la línea toggleState_1 = param.asInt() actualiza el valor de toggleState_1 (variable necesaria para el uso de interruptor manual físico) según el valor actual del interruptor virtual V4. La estructura if/else controla el encendido y apagado de la bomba según el valor de toggleState_1.

Las funciones "`State_online()`, `reading()` y `manual_control()`" se encargan de realizar las funciones principales del programa, y se ejecutan en el bucle principal del programa "`loop()`", así como "`BlynkEdgent.run()`" que mantiene la ejecución de las comunicaciones con la plataforma Blynk.

```
26
27      void State_online();
28      void reading();
29      void manual_control();
30
31      inline...
```

```
115
116     void app_loop() {
117         edgentTimer.run();
118         edgentConsole.run();
119         State_online();
120         reading();
121         manual_control();
122     }
```

Para la ejecución del programa de manera local, así como en línea se deben realizar las modificaciones de las cabeceras "ConfigMode.h" y "Settings.h", como se indica en anexo I. Así como adicional agregar las funciones "`State_online()`, `reading()` y `manual_control()`" en la cabecera "BlynkEdgent.h", entre las líneas de código 26, 30 y 115, 122.

3.5.4 Interfaz web

Para llevar a cabo la construcción de la interfaz de control web, como se muestra en la figura 3.77, se requiere de una nueva plantilla que permita la creación de cinco canales de datos en el apartado "Datastreams". Estos canales de datos serán utilizados para dos interruptores, dos indicadores de nivel "Gauge level" y un indicador adicional para conocer el estado de la bomba de agua.

La disposición y el tamaño de los widgets serán de uso libre y dependerán del gusto del diseñador. Para los interruptores, es necesario indicar el canal de datos (Datastream), el valor que se enviará o recibirá en los estados encendido y apagado, respectivamente, el nombre del widget, las etiquetas y la posición.

En cuanto a los indicadores de nivel, se deberá configurar el canal de datos, el nombre del widget y el color. De manera opcional, se podrá activar la sobreescritura de los valores si estos superan el valor máximo del widget, así como el cambio de color del indicador en función del valor recibido de los sensores. En cuanto al indicador de estado de la bomba de agua, se configurará con el nombre del widget, el canal de datos y el color (opcional).

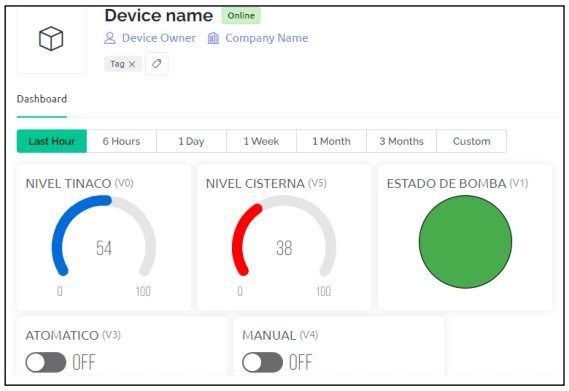


Figura 3.77. Interfaz web "Nivel de agua", Blynk.

3.5.5 Interfaz móvil

La interfaz móvil incluye dos interruptores para el control de la bomba de agua, así como tres indicadores de estado: dos para el nivel de agua y uno más para el estado de la bomba, tal como se muestra en la figura 3.78. La disposición y tamaño de los widgets es de uso libre.

Figura 3.78. Interfaz móvil, "Nivel de agua".

Para configurar los widgets, en los interruptores se deberá indicar el nombre del widget, el canal de datos, el tipo de interruptor (switch o push). Ambos interruptores deberán ser configurados como switch, indicando el color y la posición de la etiqueta (ON/OFF). Opcionalmente, se podrá indicar el valor del interruptor para cada estado o utilizar los valores proporcionados por el canal (automático), siendo este último valor el predeterminado.

Por último, en cuanto a los indicadores de nivel, se deberán configurar con el canal de datos, el nombre del widget, el tamaño del texto (opcional) y el color (opcional). El indicador de estado de la bomba admite la configuración del canal de datos y el color (opcional).

3.5.6 Pruebas y resultados

En la primera implementación del proyecto se empleó un sensor ultrasónico para realizar las mediciones de distancia y, así, permitir que la tarjeta de desarrollo determinara el nivel de agua en el contenedor. Sin embargo, durante el arranque automático de la bomba de agua, se presentaba un error que impedía el correcto funcionamiento del proyecto. A pesar de realizar algunas pruebas con el ensamble, no se logró solucionar el error en las lecturas, por lo que se optó por remplazar el sensor ultrasónico por un arreglo electrónico con un circuito integrado ULN2308 y la construcción de electrodos para tomar las lecturas del nivel.

Además, se agregaron nuevas funciones, tales como la lectura del nivel del depósito principal (cisterna), del cual se obtiene agua para abastecer el depósito secundario (tinaco). Se incluyó un control manual para que el usuario pueda activar la bomba, así como la desactivación del control automático y la notificación correspondiente en la aplicación móvil (figura 3.79) si el nivel del depósito principal no es suficiente para abastecer el depósito secundario. Una vez que se cuente con suficiente agua en el depósito principal, se podrá activar el control automático por medio de cualquiera de las interfaces o manualmente al encender la bomba con el interruptor manual físico, ver figura 3.80.

Figura 3.79. Notificaciones en aplicación móvil por nivel de agua insuficiente (izquierda), interfaz web y fotografía de pruebas (derecha).

Figura 3.80. Interfaz de monitoreo y control móvil (izquierda), web (derecha).

Durante el desarrollo del proyecto, se llevaron a cabo pruebas para permitir que el programa funcione de forma autónoma sin la necesidad de estar conectado al servidor de Blynk o depender de una conexión inalámbrica, como se puede observar en la figura 3.81. En este modo, el programa continúa funcionando normalmente en su modo automático a menos que el nivel del depósito principal no sea suficiente para abastecer el depósito secundario. En este caso, el programa sólo puede funcionar de forma local, si el nivel de agua es nuevamente el suficiente en el depósito principal, la función automática puede ser reactivada manualmente mediante el interruptor físico, este interruptor permite también la operación manual de la bomba.

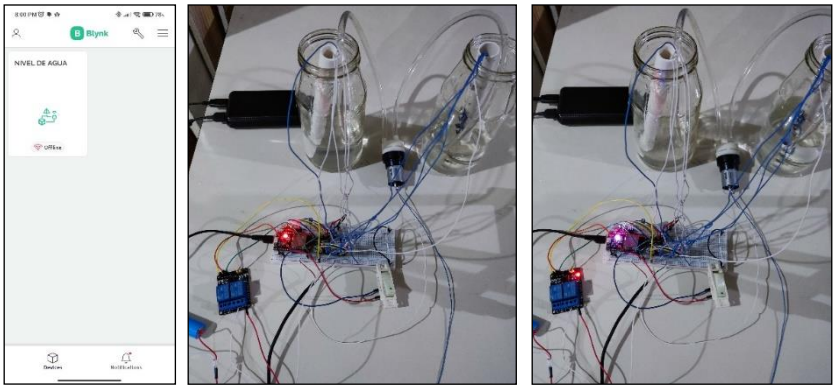


Figura 3.81. Programa funcional sin conexión a internet.

El resultado de este experimento ha sido satisfactorio y ha permitido el monitoreo y control remoto de la bomba de agua, ya sea a través de una conexión a Internet o sin ella. Esto ha dotado al proyecto de comodidad, practicidad y facilidad de uso para el usuario final. Como mejora futura, se puede considerar la estética y el embalaje del proyecto para facilitar su instalación y mejorar su presentación final.

3.6 Control de iluminación RGB (en línea)

El programa es un sketch de Arduino diseñado para tarjetas ESP32 y su propósito es controlar una tira LED RGB utilizando la plataforma Blynk. La función principal del programa es recibir señales desde la aplicación Blynk y utilizar esas señales para ajustar el brillo de los LED rojo, verde y azul en la tira LED RGB.

El circuito hace uso de transistores MOSFET, por medio de señales PWM permite regular la potencia que se entrega a cargas, este circuito puede funcionar para elementos de corriente continua y alterna.

3.6.1 Diagrama de bloques

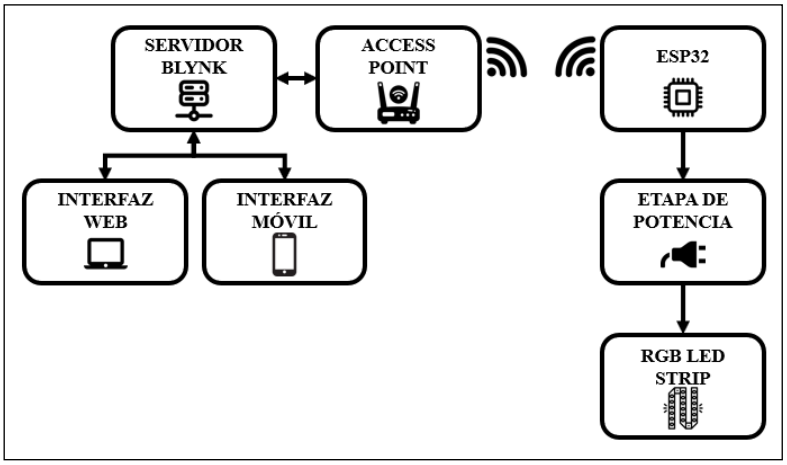


Figura 3.82. Diagrama de bloques, iluminación RGB.

- Interfaz web y móvil: a través de, es posible realizar el monitoreo y control en la plataforma Blynk, cabe recalcar que las interfaces son independientes en la creación de su diseño, ambas comparten como base canales de datos, lo que permite sincronizar y los cambios que se realicen en una interfaz sean reflejados en la otra.
- Servidor Blynk: permite administrar las configuraciones, así como visualizar la interfaz web de la plataforma, es en este dónde se crean los canales de datos que servirán como puente de comunicaciones entre las interfaces, el mismo servidor, el access point y la tarjeta ESP32.
- Access Point: es el dispositivo que permite establecer un canal de comunicación inalámbrica entre la tarjeta de desarrollo y la plataforma IoT por medio de internet. Este canal normalmente se encuentra cifrado y para obtener la conexión es necesario proporcionar al dispositivo que se conectara las credenciales de acceso (SSID y contraseña).
- ESP32: es la tarjeta de desarrollo que incluye el microcontrolador ESP32, este ya integra un módulo y antenas para realizar la conexión de manera inalámbrica con dispositivos que permiten el acceso a internet. Por medio de esta se realizan todas las operaciones lógicas de comunicación y control, lo cual incluye la gestión de la etapa de potencia y la plataforma de control IoT.
- Etapa de potencia: es el circuito electrónico que permite controlar la energía que se requiere en los actuadores de acuerdo con las funcionalidades incorporadas en el sistema.
- RGB LED Strip: es el dispositivo que se conecta inmediatamente después de la etapa de potencia, y que en este caso produce la iluminación en los colores primarios rojo, verde y azul en un área. Puede utilizarse tiras LED con conectores de cuatro pines, que trabaje a 12 VDC, con un consumo aproximado de 24 watts de potencia.

3.6.2 Diagrama esquemático

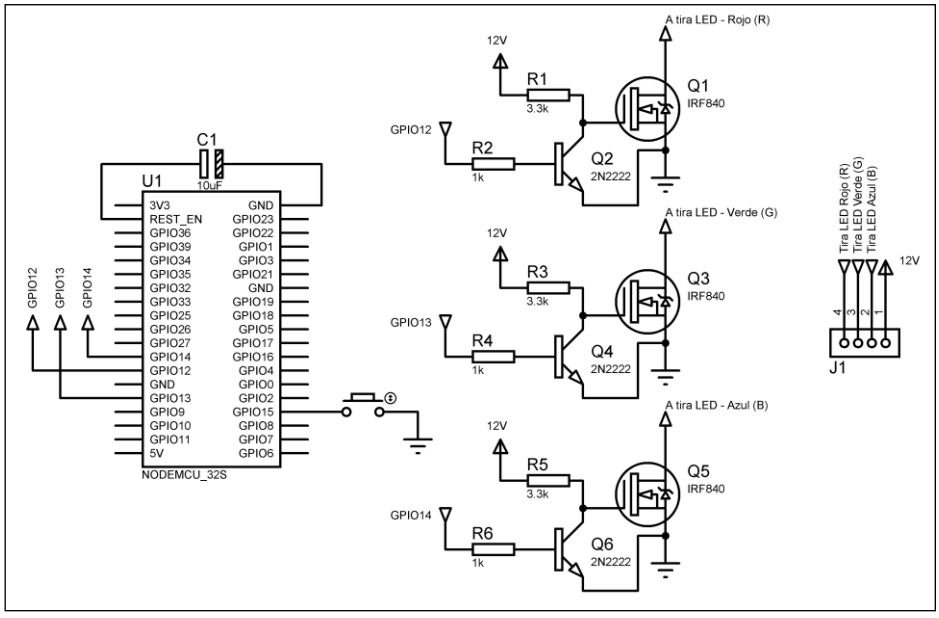


Figura 3.83. Diagrama esquemático control PWM.

Material necesario:

- 1 Tarjeta NodeMCU-32S.
- 1 Push button (opcional).
- 1 Capacitor 10µF a 50V (opcional).
- 3 Resistores de 1kΩ a 1/2 watt.
- 3 Resistores de 3.3kΩ a 1/2 watt.
- 3 Transistores BJT 2N2222.
- 3 Transistores MOSFET IRF840.
- 1 Tira LED RGB a 12V.
- 1 Fuente de alimentación a 12V.

En el diagrama esquemático, ver figura 3.83, se puede observar la disposición de los elementos para su conexión, recordando que el capacitor y push button son componentes opcionales, ya que en la tarjeta NodeMCU-32S se puede utilizar el push button que integra la misma para ambas funciones (programar la tarjeta y si se mantiene pulsado por diez segundos reiniciar las configuraciones para la conexión Wifi de la tarjeta una vez cargada la biblioteca de Blynk, consultar el capítulo 2.2.5 y 3.1.2.2).

Para poder conectar la tira LED RGB es necesario un arreglo electrónico que emplea el uso de transistores MOSFET. En el ensamble se utilizan los IRF840 que soportan un consumo de hasta 75 watts, lo cual es más que suficiente para el control de las tiras LED. Estos transistores pueden ser energizados con los 12VDC de la fuente que alimenta la tira LED. Este transistor se acompaña de una pequeña etapa de preamplificación para poder saturar de una forma más rápida el MOSFET (para más información consultar el capítulo 2.5.3). Se emplea un transistor BJT 2N2222 para la etapa de preamplificación. Las resistencias limitan la corriente que puede circular en el circuito y sirven de protección para los transistores del ensamble.

Esta etapa de potencia permite controlar la potencia individual de cada uno los colores que componen a la iluminación RGB, por medio de señales PWM que se generan en el microcontrolador. El diseño del arreglo electrónico permite el uso de tiras LED que cuentan solo con 4 pines de conexión.

Para emitir las señales PWM de control se emplean los GPIO 12, 13 y 14, a estos se conecta la base de los transistores 2N2222 respectivamente. La tarjeta de desarrollo se puede alimentar de forma independiente por medio del pin 5V, a través de fuente independiente o por medio de la fuente para la tira LED con el uso de un regulador de voltaje (consultar los capítulos 2.2.5 y 2.5.4, para más detalles), el alambrado del experimento se puede observar en la figura 3.84.

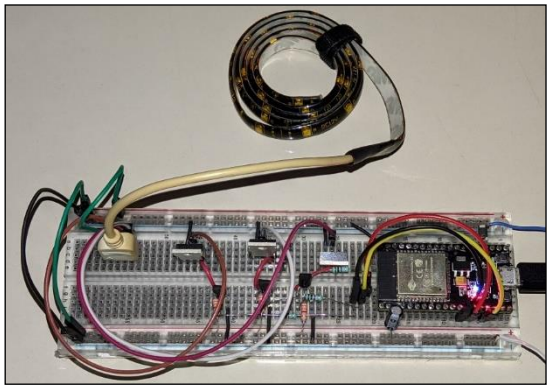
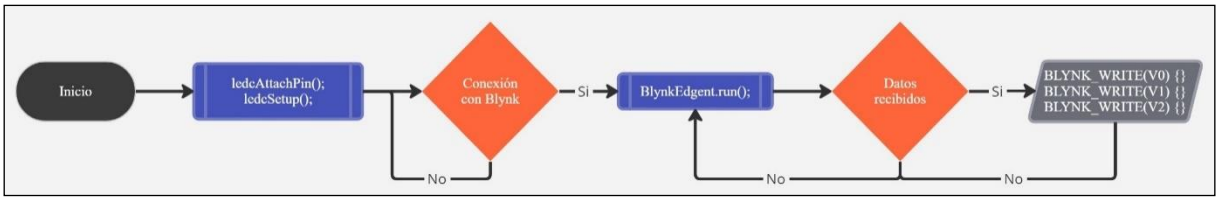
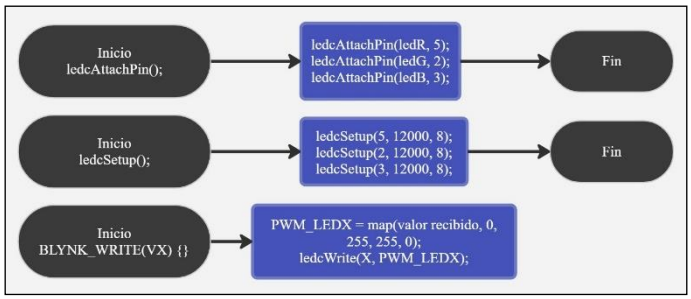


Figura 3.84. Alambrado del circuito "Iluminación RGB".

3.6.3 Diagrama de flujo



a)



b)

Figura 3.85. a) Diagrama de flujo, programa principal, b) Subrutinas de ejecución inicial y control.

En general, el programa lee los valores de los deslizadores o el selector de color en la aplicación móvil o web de Blynk y utiliza esos valores para controlar la intensidad de los colores LED rojo, verde y azul en la tira LED RGB. Las líneas de código principales incluyen la configuración de los pines de salida para la conexión de la tira LED, la configuración de la conexión Blynk, la configuración de los canales PWM en tarjetas ESP32 y las tres funciones "`BLYNK_WRITE()`" que ajustan la intensidad de brillo de los LED.

```
27      void setup() {
28        ledcAttachPin(ledR, 5);
29        ledcAttachPin(ledG, 2);
30        ledcAttachPin(ledB, 3);
31        ledcSetup(5, 12000, 8);
```

```
32          ledcSetup(2, 12000, 8);
33          ledcSetup(3, 12000, 8);
34          Serial.begin(115200);
35          ledcWrite(5, 255);
36          ledcWrite(2, 255);
37          ledcWrite(3, 255);
38          delay(100);
39
40          BlynkEdgent.begin();
41        }
```

En la función de configuración "`setup()`", se realiza la configuración de los pines LED de la tira RGB con los canales PWM del microcontrolador ESP32, especificando la frecuencia y la resolución de los pulsos PWM. También se establece la velocidad de transmisión serial a 115200 baudios, se establecen los pines de LED RGB en su estado inicial y se llama a la función "`BlynkEdgent.begin()`" para iniciar la conexión con la aplicación Blynk.

```
43        BLYNK_WRITE(V0) {
44          PWM_LEDR = map(param.asInt(), 0, 255, 255, 0);
45          ledcWrite(5, PWM_LEDR);
46        }
47        BLYNK_WRITE(V1) {
48          PWM_LEDG = map(param.asInt(), 0, 255, 255, 0);
49          ledcWrite(2, PWM_LEDG);
50        }
51        BLYNK_WRITE(V2) {
52          PWM_LEDB = map(param.asInt(), 0, 255, 255, 0);
53          ledcWrite(3, PWM_LEDB);
54        }
```

Las funciones "`BLYNK_WRITE(V0)`, `BLYNK_WRITE(V1)` y `BLYNK_WRITE(V2)`", se ejecutan cada vez que se cambia el valor del control deslizante en la aplicación Blynk para ajustar el brillo de los LED del color correspondiente en la tira RGB. Estas funciones utilizan la función "`map()`" para mapear el valor del control deslizante de 0-255 a 255-0, que corresponde a la intensidad de los LED. Finalmente, se llama a la función "`ledcWrite()`" para establecer la intensidad del LED correspondiente en función del valor del control deslizante.

La función "`loop()`" llama a la función "`BlynkEdgent.run()`" para mantener la conexión con la aplicación Blynk.

3.6.4 Interfaz web

Para la construcción de la interfaz de control web, tal como se muestra en la figura 3.86, se requiere la creación de tres canales de datos para su uso en controles deslizantes, así como de un selector de color que solo estará disponible en la interfaz móvil. La creación de estos canales se llevará a cabo en una nueva plantilla en la sección "Datastreams".

La disposición y tamaño de los widgets será de uso libre y estará sujeta a la discreción del diseñador. En el caso de los controles deslizantes, será necesario indicar el canal de datos correspondiente (Datastream). Es importante mencionar que dichos canales ya han sido

previamente configurados con un rango de valores de 0 a 255, los cuales se enviarán para controlar el ciclo de trabajo en los canales PWM designados en la tarjeta de desarrollo. Asimismo, se deberá añadir el nombre del widget, las etiquetas correspondientes, su posición y el color de la barra deslizable (opcional).

Figura 3.86. Interfaz web, controles deslizantes.

3.6.4 Interfaz móvil

En la interfaz móvil opcionalmente, se encuentran tres controles deslizantes que permiten modificar el color del canal de datos configurado y correspondiente al color, así como el nombre del widget, las etiquetas, la posición y el color del deslizador (opcional). Asimismo, se dispone de un selector de color denominado "ZeRGBa" en el apartado de widgets, que se configura en modo simple y permite elegir los canales de datos asociados a cada color (en la aplicación, "R DATASTREAM", "G DATASTREAM" y "B DATASTREAM"). Gracias a la mezcla de estos tres canales, es posible generar hasta 16,777,216 colores. En cuanto a la edición del widget, solo se permite modificar el nombre y la posición de la etiqueta.

Cabe destacar que la disposición y tamaño de los widgets son completamente libres y quedan a elección del diseñador, un posible diseño se muestra en la figura 3.87.

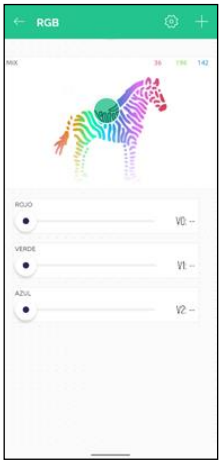


Figura 3.87. Interfaz móvil "Iluminación RGB".

3.6.5 Pruebas y resultados

Este proyecto se basa en el uso de señales PWM para variar el ciclo de trabajo en dispositivos, como, por ejemplo, iluminación LED. En una sección anterior se ha detallado la información del circuito y los componentes que forman parte del ensamble, como los transistores MOSFET y BJT que lo conforman por canal. Sin embargo, debido a algunas características en el uso de la biblioteca para la plataforma de Blynk, se presentaron algunos problemas que se resolvieron estudiando de forma más detallada su funcionamiento.

En el capítulo 3.1.2.2 se describe con mayor detalle cómo configurar una tarjeta con ESP32 "personalizada" por el fabricante para su correcto funcionamiento con la biblioteca Blynk y la plataforma. Es recomendable conocer las prestaciones que ofrecen los diferentes fabricantes en las tarjetas de desarrollo, ya que esto puede ahorrar tiempo en los proyectos a implementar.

Durante las pruebas de este experimento, no se logró hacer funcionar correctamente un canal PWM configurado en la tarjeta de desarrollo. En la figura 2.5, se puede observar que la tarjeta NodeMCU-32S cuenta con dos diodos LED en la PCB que pueden ser aprovechados como indicadores. Uno de los diodos LED está especificado en la biblioteca como indicador para mostrar el estado de la tarjeta de desarrollo en cuanto a las conexiones inalámbricas, y utiliza un canal PWM de los que disponemos en la NodeMCU-32S utilizada en este trabajo. Para conseguir que el proyecto funcionara de forma óptima, fue necesario cambiar la salida del canal PWM a una que no fuera utilizada por la biblioteca, y con esto se pudieron realizar las pruebas de forma correcta.

Por medio de la interfaz en la aplicación móvil, podemos seleccionar el color que deseemos, disponiendo de 16,777,216 combinaciones posibles, como se muestra en la figura 3.88.

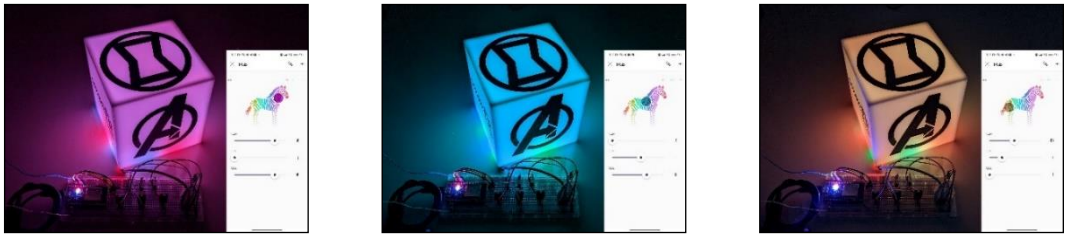


Figura 3.88. Selección de color en widget ZeRGBa, interfaz móvil.

La interfaz cuenta además con controles deslizantes que permiten ajustar la intensidad del brillo del color seleccionado, lo que permite elegir un color específico mediante su valor, que oscila entre 0 y 255. Cada color se define mediante un conjunto de valores correspondientes a las componentes "R", "G" y "B". En el experimento, el color negro se obtiene cuando todas las componentes son 0: (0, 0, 0), y se utiliza para apagar todas las luces LED. El color blanco, por otro lado, se obtiene al establecer los tres colores primarios al valor máximo de 255: (255, 255, 255), como se muestra en la figura 3.89.

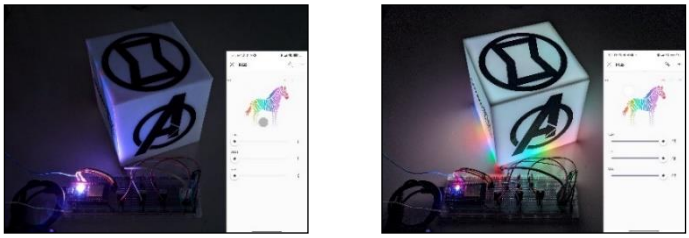


Figura 3.89. Combinación de los colores primarios para color negro y blanco.

De este modo, el rojo se obtiene con (255,0,0), el verde con (0,255,0) y el azul con (0,0,255), ver figura 3.90.

Figura 3.90. Colores monocromáticos.

Por otro lado, la interfaz web también permite seleccionar un color mediante controles deslizantes y su valor, tal como se muestra en la figura 3.91. Es importante destacar que esta interfaz puede resultar menos intuitiva y atractiva para el usuario final.

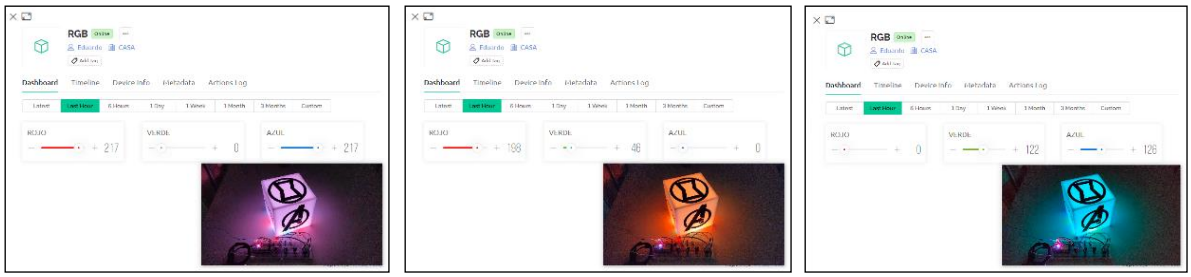


Figura 3.91. Control de color por medio de interfaz web.

El resultado del experimento es satisfactorio, ya que permite el control de la iluminación a través de una conexión a Internet, lo que permite su operación de forma remota, aportando confort y comodidad, así como la capacidad de crear ambientes agradables en el hogar. Como trabajo futuro, se podría considerar una mejora en la estética o en el embalaje para facilitar la instalación por parte del usuario final y mejorar la presentación general del proyecto.

3.7 Monitoreo y riego de sustrato (en línea y local)

El programa siguiente muestra las bases para la creación de una maceta inteligente, se encarga de medir el nivel de humedad del suelo, controlar la irrigación de una planta y mostrar el estado de humedad través de una pantalla LCD. Además, cuenta con botones para ajustar el nivel de

humedad deseado, y poder almacenar este valor en la memoria no volátil. El programa está escrito para ser implementado por medio del IDE Arduino y es compatible con tarjetas ESP32.

3.7.1 Diagrama de bloques

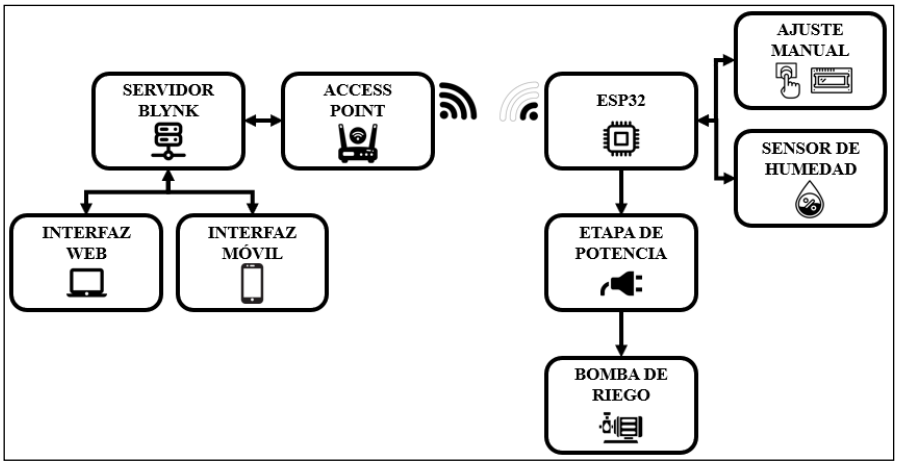


Figura 3.92. Diagrama de bloques, "Maceta IoT".

- Interfaz web y móvil: a través de, es posible realizar el monitoreo y control en la plataforma Blynk, cabe recalcar que las interfaces son independientes en la creación de su diseño, ambas comparten como base canales de datos, lo que permite sincronizar y los cambios que se realicen en una interfaz sean reflejados en la otra.
- Servidor Blynk: permite administrar las configuraciones, así como visualizar la interfaz web de la plataforma, es en este dónde se crean los canales de datos que servirán como puente de comunicaciones entre las interfaces, el mismo servidor, el access point y la tarjeta ESP32.
- Access Point: es el dispositivo que permite establecer un canal de comunicación inalámbrica entre la tarjeta de desarrollo y la plataforma IoT por medio de internet. Este canal normalmente se encuentra cifrado y para obtener la conexión es necesario proporcionar al dispositivo que se conectara las credenciales de acceso (SSID y contraseña).
- ESP32: es la tarjeta de desarrollo que incluye el microcontrolador ESP32, este ya integra un módulo y antenas para realizar la conexión de manera inalámbrica con dispositivos que permiten el acceso a internet. Por medio de esta se realizan todas las operaciones lógicas de comunicación y control, lo cual incluye la gestión de la etapa de potencia y la plataforma de control IoT.
- Ajuste manual: es el conjunto de interruptores e indicador que permiten el control manual y visualización de los valores de referencia del sistema de forma local, sin depender de una conexión inalámbrica en la tarjeta de desarrollo. Al actualizar el estado de forma manual el cambio también es reflejado en las plataformas en línea.
- Sensor de humedad: existen sensores que determinan la cantidad de agua en el suelo y sirven de apoyo para definir el momento oportuno para el riego de plantas o cultivos. A través de estos se puede medir una señal eléctrica y en base a esta se puede calcular la

cantidad de agua en el suelo, así como determinar el volumen de agua almacenado en éste después de un riego y así determinar la eficiencia del riego.

- Etapa de potencia: es el circuito electrónico que permite controlar la energía que se requiere en los actuadores de acuerdo con las funcionalidades incorporadas en el sistema.
- Bomba de agua: es el dispositivo que se conecta inmediatamente después de la etapa de potencia, y que en este caso produce la irrigación de una porción de sustrato. Se pueden emplear bombas pequeñas que trabajen de 2.5 a 6 V, el relevador empleado en la etapa de potencia permite un consumo máximo de corriente de hasta 10 A, suficiente para energizar bombas de estas características.

3.7.2 Diagrama esquemático

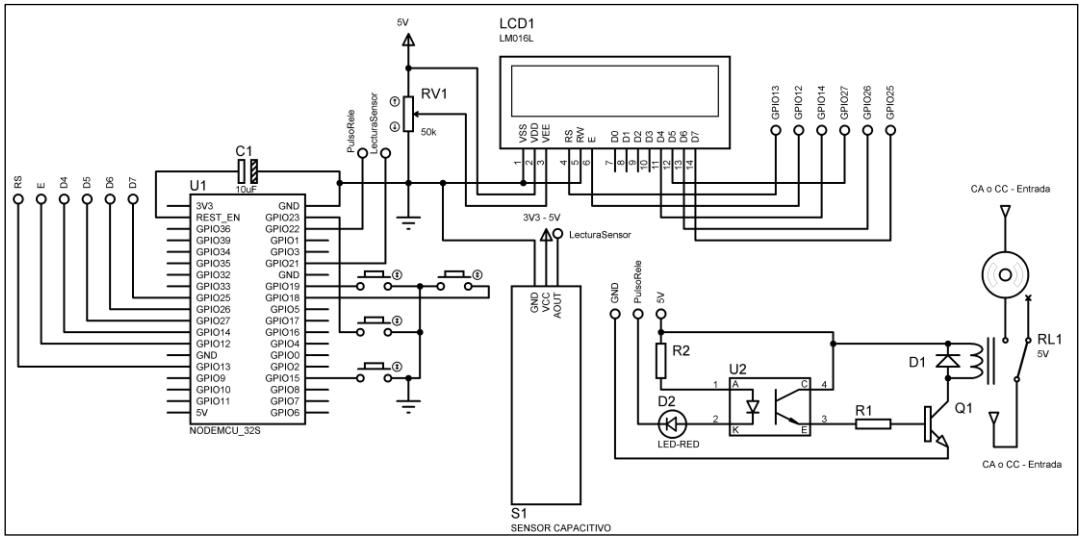


Figura 3.93. Diagrama esquemático para maceta IoT y de riego automático.

Material necesario:

- 1 Tarjeta NodeMCU-32S.
- 4 Push button.
- 1 Capacitor 10μF a 50V (opcional).
- 1 Potenciómetro a 50kΩ.
- 1 Display LCD 16x2.
- 1 Sensor capacitivo SEN-HS-CAP.
- 1 Módulo relevador SRD-5VDC-SL-C.

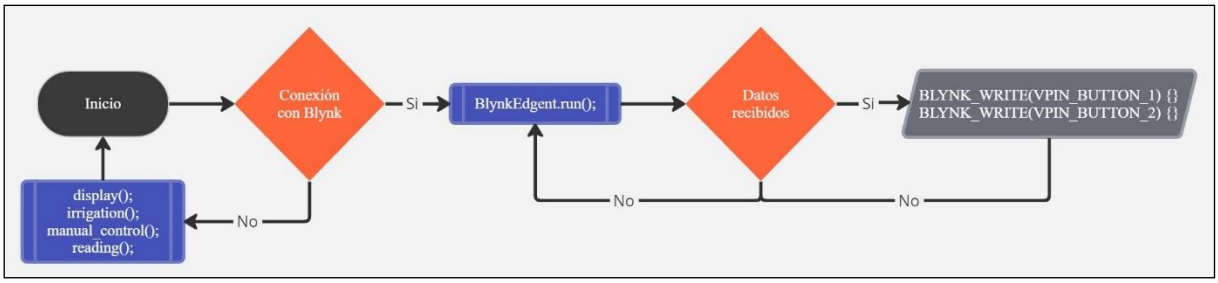
Tal y como se muestra en el diagrama esquemático de la figura 3.93. se puede apreciar la disposición de los componentes para su conexión. El capacitor como el push button son opcionales, ya que en la tarjeta NodeMCU-32S se puede utilizar el push button que se integra en la misma para ambas funciones (programar la tarjeta y, si se mantiene pulsado por diez segundos, reiniciar las configuraciones para la conexión Wifi de la tarjeta una vez cargada la biblioteca de Blynk). Para más detalles, se puede consultar el capítulo 2.2.5 y 3.1.2.2.

En cuanto al alambrado en el circuito de la pantalla LCD, se emplea la conexión con un bus de 4 datos, ya que se integra la misma biblioteca de programación para tarjetas Arduino, la cual también es compatible con la ESP32 (nota: también es posible realizar la conexión con un bus de 8 datos con solo 2 pines, pero se requiere de un módulo I2C para el panel LCD). Además, se utiliza un sensor capacitivo (para más detalles, consultar el capítulo 2.3.5), el cual puede ser alimentado con una fuente de 3.3 o 5 V y puede ser conectado a través de los pines de la tarjeta de desarrollo.

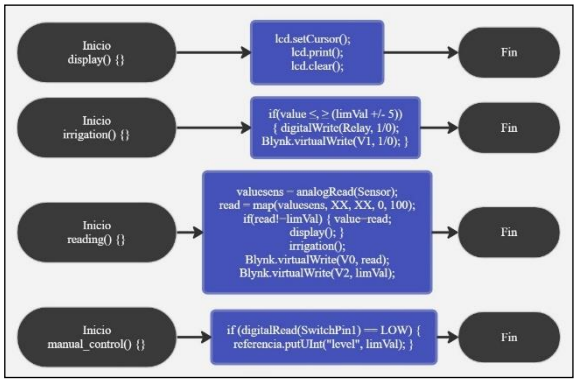
Para leer las señales del sensor, se emplea el GPIO21, mientras que la señal de control para la bomba tiene su salida a través del GPIO22. Por otro lado, los interruptores que permiten el ajuste manual y local del valor de referencia para el nivel de humedad deseado se conectan por medio de los GPIO 18, 19 y 23, en caso de que no haya conexión a internet o si hay alguna falla en la comunicación de la tarjeta de desarrollo con la plataforma.

En la etapa de potencia, se utiliza un módulo relé, tal como se describe en el capítulo 2.5.1, para activar la bomba. Cualquier tipo de bomba que cumpla con las características descritas en el capítulo 2.4.3 se puede conectar. Es importante destacar que este circuito debe ser energizado con 5V, a través de una fuente externa, la cual también puede alimentar a la tarjeta de desarrollo mediante el pin 5V. Para más información, se puede consultar los capítulos 2.2.5 y 2.5.4.

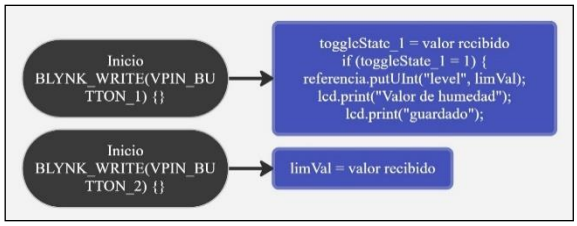
3.7.3 Diagrama de flujo



a)



b)



c)

Figura 3.94. a) Diagrama de flujo, programa principal, b) Subrutinas de ejecución local y en línea, c) Subrutinas de control en línea.

La función principal del programa es leer la humedad del suelo a través de un sensor, compararla con un valor de referencia establecido por el usuario y activar o desactivar un relé conectado a una bomba de agua según corresponda. Además, se muestra la humedad actual y el valor de referencia en una pantalla LCD y se permite al usuario ajustar este valor de referencia a través de dos botones físicos.

El programa utiliza varios pines de entrada y salida de la ESP32, incluyendo los GPIO 18, 19 y 23 para los botones físicos, el GPIO 32 para el relé, y el 33 para el sensor de humedad del suelo. También utiliza la biblioteca "`Preferences.h`" para almacenar el valor de referencia de humedad en la memoria no volátil de la ESP32, la biblioteca "`LiquidCrystal.h`" para controlar la pantalla LCD, y "`BlynkEdgent.h`" para conectarse y comunicarse con la plataforma de Blynk.

Las funciones principales del programa son:

```
79      void display() {
80          lcd.setCursor(0,1);
81          lcd.print("Referencia=  ");
82          lcd.setCursor(11,1);
83          lcd.print(limVal);
84          lcd.setCursor(0,0);
85          lcd.print("Humedad=    ");
86          if (value>9){
87             lcd.setCursor(8,0);
88             lcd.print(value);
89             delay(300);
90          }
91          else{
92             lcd.setCursor(9,0);
93             lcd.print(value);
94             delay(300);
95          }
96          lcd.setCursor(11,0);
97          lcd.print("%");
98          delay(20);
99      }
```

La función "`display()`" se encarga de mostrar la información de humedad en la pantalla LCD. La función primero lee el valor del sensor de humedad, lo convierte en un porcentaje y lo muestra en la pantalla junto con el valor de referencia de humedad.

```
101     void irrigation() {
102       if (value<=(limVal-5)){
103         digitalWrite(Relay, HIGH);
104         Blynk.virtualWrite(V1, 1);
105       }
106       else if (value>=(limVal+5)){
107         digitalWrite(Relay, LOW);
108         Blynk.virtualWrite(V1, 0);
109       }
110     }
```

La función "`irrigation()`" controla el riego de la planta en función de la humedad del suelo y el valor de referencia establecido por el usuario. Si el nivel de humedad es menor que el nivel de referencia, la función enciende el relé para activar el riego. Si el nivel de humedad es mayor que el nivel de referencia, la función apaga el relé para detener el riego.

```
112     void reading() {
113       int const valuesens = analogRead(Sensor);
114       int const read = map(valuesens, 3600, 1450, 0, 100);
115       if(read!=limVal){
116             value=read;
117             display();
118           }
119           irrigation();
120       if (button1.pressed) {
121         if (numberKeyPresses <=99 && numberKeyPresses >=0) {
122           limVal = constrain(numberKeyPresses, 0, 99);
123         }
124           button1.pressed = false;
125       }
126       if (button2.pressed2) {
127         if (numberKeyPresses <=99 && numberKeyPresses >=0) {
128           limVal = constrain(numberKeyPresses, 0, 99);
129         }
130           button2.pressed2 = false;
131       }
132       Blynk.virtualWrite(V0, read);
133       Blynk.virtualWrite(V2, limVal);
134       Blynk.virtualWrite(V4, limVal);
135     }
```

En la función "`reading()`" se lee el valor del sensor de humedad y llama a las funciones "`display()`" e "`irrigation()`" para mostrar la información en la pantalla y controlar el riego, también actualiza los widgets virtuales en las aplicaciones web o móvil de Blynk.

```
137     void manual_control() {
138       if (digitalRead(SwitchPin1) == LOW) {
139         referencia.putUInt("level", limVal);
140         lcd.clear();
141         lcd.setCursor(0,0);
142         lcd.print("Valor de humedad");
143         lcd.setCursor(4,1);
144         lcd.print("guardado");
145         delay(3000);
146         lcd.clear();
147       }
148     }
```

Por último la función "`manual_control()`" esta función se encarga de guardar el valor de referencia de humedad en la memoria no volátil de la ESP32 cuando se presiona el botón físico. Guarda el valor de referencia actual de humedad y muestra un mensaje en la pantalla LCD confirmando la acción.

La función "`loop()`" simplemente llama a las funciones "`BlynkEdgent.run()`" para mantener la conexión con la aplicación Blynk y "`reading()`, `display()`, `irrigation()` y `manual_control()`" para continuar la ejecución de cada función después de haber ejecutado las tareas correspondientes.

Funciones adicionales:

"`BLYNK_WRITE(VPIN_BUTTON_1)`", función que se ejecuta cuando se presiona el botón virtual asociado al pin virtual V3 en la aplicación de Blynk. Si el botón se activa, se guarda el valor de referencia actual de humedad en la memoria no volátil de la ESP32 y se muestra un mensaje en la pantalla LCD confirmando la acción.

"`BLYNK_WRITE(VPIN_BUTTON_2)`", función que se ejecuta cuando se cambia el valor del widget slider asociado al pin virtual V4 en la aplicación de Blynk. Actualiza el valor de referencia de humedad.

La estructura "`Button`" se compone de tres miembros de datos: "`PIN`", "`pressed`" y "`numberKeyPresses`". Donde "`PIN`" es una constante de tipo número entero que representa el número de GPIO utilizado para leer el estado del botón, "`pressed`" es una variable de tipo booleana que indica si el botón está actualmente presionado o no, y por último "`numberKeyPresses`" también es una variable de tipo entero que indica el número de veces que se ha presionado el botón.

Si el botón se presiona (es decir, si "`pressed`" es verdadero), se verifica que el valor de "`numberKeyPresses`" esté dentro del rango permitido de 0 a 99. Si es así, el valor se ajusta utilizando la función "`constrain()`" para asegurarse de que esté dentro de los límites definidos. El resultado se asigna a la variable con el valor de humedad deseado. Finalmente, "`pressed`" se establece en falso para indicar que el botón ya no está presionado.

Este fragmento de código está diseñado para permitir que el usuario ajuste un valor utilizando los botones físicos, y luego asegurarse de que el valor esté dentro de un rango permitido.

La función "`IRAM_ATTR isr()`" es una función que se ejecuta en respuesta a interrupciones. La función está definida con el atributo ""`IRAM_ATTR`", lo que significa que se almacena en la memoria RAM para una ejecución más rápida. Cuando se ha presionado un botón la función realiza un seguimiento del número de veces que se ha presionado el botón. La función "`isr()`" se ejecuta cuando existe una interrupción y realiza dos tareas: decrementar e incrementar la variable "`numberKeyPresses`" y establecer "`pressed`" en verdadero.

```
26
27      void reading();
28      void display();
29      void irrigation();
30      void manual_control();
31
32      inline...
```

```
117     void app_loop() {
118         edgentTimer.run();
119         edgentConsole.run();
```

```
120         manual_control();
121         reading();
122         display();
123         irrigation();
124     }
```

Para la ejecución del programa de manera local, así como en línea se deben realizar las modificaciones de las cabeceras "ConfigMode.h" y "Settings.h", como se indica en anexo I. Así como adicional agregar las funciones "`reading()`, `display()`, `irrigation()` y `manual_control()`" en la cabecera "BlynkEdgent.h", entre las líneas de código 26, 31 y 119, 124, para garantizar la ejecución del programa de forma correcta de manera local.

3.7.4 Interfaz web

Para la interfaz de control web, consulte la figura 3.95. Se debe crear una nueva plantilla, en "Datastreams", configurar cinco canales de datos para el uso de un control deslizante, un interruptor exclusivo para la interfaz móvil y tres indicadores (dos de nivel y uno adicional para conocer el estado de la bomba de agua). La disposición y el tamaño de los widgets son libres y dependerán del gusto del diseñador.

Para el control deslizante, se debe especificar el canal de datos (Datastream) este enviara ese valor a la tarjeta de desarrollo para ajustar el nivel de humedad de referencia. Además, es posible editar el nombre del widget, las etiquetas, la posición y el color del control deslizante (opcional).

En cuanto a los indicadores de nivel, se debe configurar el canal de datos, el nombre del widget y el color. Opcionalmente, se puede activar la sobreescritura de los valores si estos superan el valor máximo del widget, así como el cambio del color del indicador en base al valor recibido del sensor o referencia. Por último, el indicador de estado de la bomba de agua debe ser configurado con el nombre del widget, el canal de datos y el color del indicador (opcional).

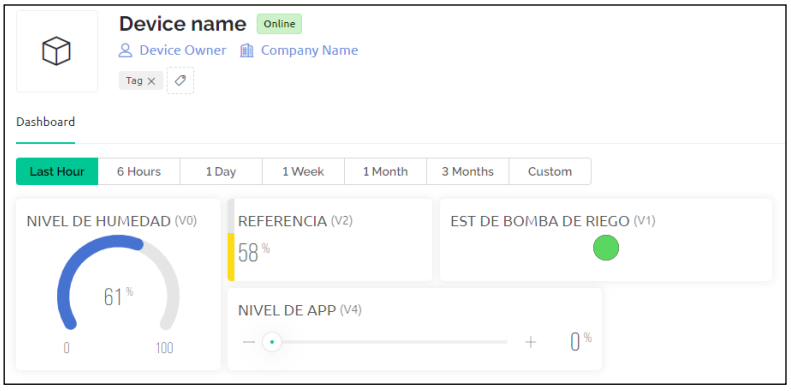


Figura 3.95. Interfaz web "Maceta IoT".

3.7.5 Interfaz móvil

En la interfaz móvil, es necesario colocar tres indicadores, un control deslizante y un interruptor. La disposición y tamaño de los widgets son libres, como se muestra en la figura 3.96.

Para configurar los widgets, el interruptor se debe ajustar con el nombre del widget, el canal de datos, el tipo de interruptor (switch o push) y debe configurarse como push, ya que solo se requiere una pequeña señal para activar la guarda del valor de referencia en la memoria no volátil. También se debe configurar el color y la posición de la etiqueta.

El control deslizante se configura especificando el canal de datos y permite personalizar el nombre del widget, el color, las etiquetas y la posición.

Por último, para los indicadores de nivel, se deben configurar con el canal de datos, el nombre del widget, el tamaño del texto (opcional) y el color (opcional). Se utilizan dos tipos de indicadores, "Gauge" y "Labeled Value". El indicador de estado de la bomba permite la configuración del canal de datos y el color del indicador (opcional).

Figura 3.96. Interfaz móvil "Maceta IoT".

3.7.6 Pruebas y resultados

Las primeras pruebas de desarrollo para el proyecto se centraron en la selección del sensor adecuado para realizar las mediciones de humedad en el sustrato. Como se detalla en el capítulo 2 de este trabajo, se evaluaron diversas alternativas para este fin, algunas de las cuales resultaban costosas. Por esta razón, se optó por el uso de un sensor capacitivo, que, aunque no mide directamente la humedad del sustrato, permite calcularla a través de la conductividad eléctrica del mismo. Este sensor seleccionado tiene un tiempo de vida superior al de otros sensores y es asequible en términos de costo, lo que lo hace ideal para el proyecto en cuestión. Se realizaron pruebas independientes al sensor capacitivo para delimitar las condiciones en las que se llevan a cabo las mediciones (calibración).

El programa base utilizado para el proyecto emplea la función de almacenamiento de datos en la memoria no volátil de las tarjetas Arduino. Sin embargo, al realizar la adaptación de dicho programa a la tarjeta ESP32, se encontró que el núcleo instalado en el IDE Arduino ha dejado en desuso la biblioteca "EEPROM". Aunque esta biblioteca aún es compatible con las tarjetas que integran ESP32, cuenta con limitaciones en su uso. Por ello, se implementó el uso de la biblioteca "Preferences", que optimiza el uso de la memoria no volátil disponible en las tarjetas con ESP32. El modo de empleo de la biblioteca se describe en el capítulo 2.2.5 y consiste en

almacenar los datos en espacios de palabras dentro de la memoria, a diferencia del uso de bytes en las tarjetas Arduino.

En adición al proyecto, se integraron tres interruptores de botón junto con una pantalla LCD para dotar al sistema de independencia en cuanto a la conectividad inalámbrica de la tarjeta de desarrollo y así no depender de una conexión o comunicación con los servidores de Blynk. El experimento cuenta con una bomba pequeña de riego que, en función del valor de referencia indicado por el usuario, mantiene la humedad en el sustrato de manera completamente automática.

Tanto en las interfaces como en el hardware físico, se cuenta con interruptores (control deslizante en la versión móvil) para ajustar el valor de referencia deseado y así tener un control del nivel de humedad en el sustrato para la planta o plantas deseada(s), tal como se muestra en la figura 3.97.

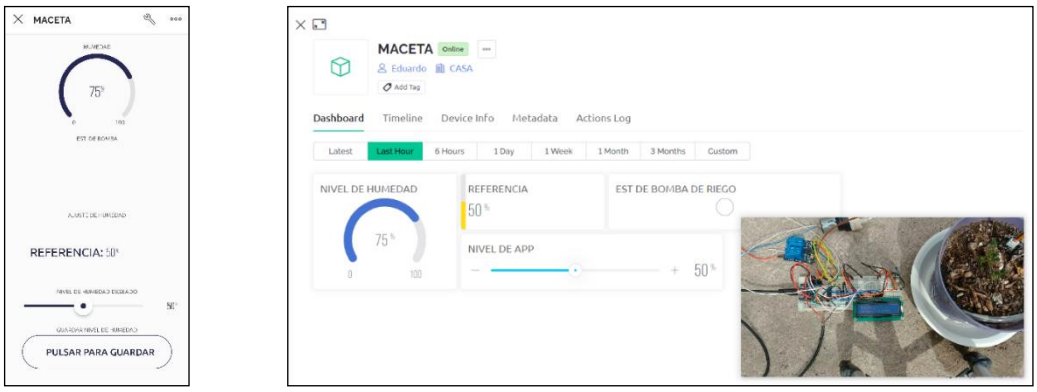


Figura 3.97. Nivel de humedad inicial en sustrato.

La figura 3.98 y 3.99 demuestran que es posible ajustar el valor de humedad de diversas formas, a través de la interfaz web o móvil, o mediante el uso de los interruptores físicos, sin depender de la comunicación con Blynk. Además, el valor de referencia fijado puede ser almacenado en la memoria no volátil de la tarjeta de desarrollo para asegurar que el nivel de humedad se mantenga dentro del rango de -5% a +5% del valor de referencia, incluso en situaciones de pérdida de energía o reinicio de la tarjeta. Esto permite que el usuario no tenga que ajustar el valor en cada ocasión, lo que contribuye a mantener las condiciones óptimas para la sustentabilidad de la planta en el sustrato.

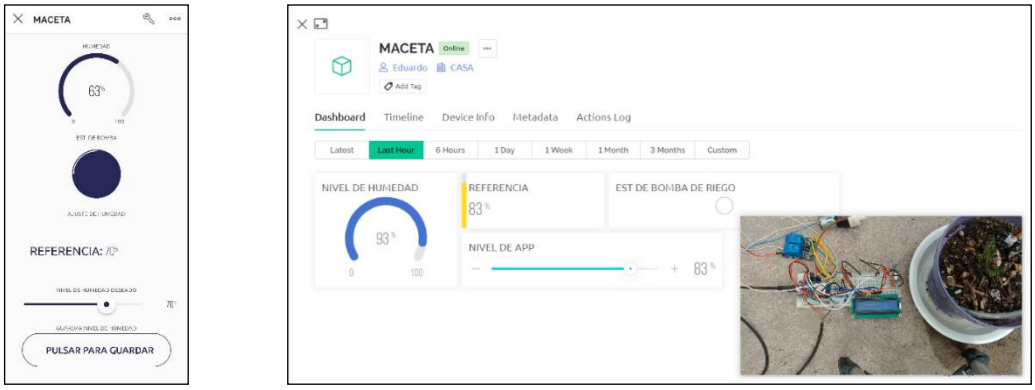


Figura 3.98. Incremento en el nivel de humedad deseado.

Finalmente, tal como se mencionó previamente, el programa es capaz de operar tanto en línea como de manera local, lo que brinda al usuario una experiencia cómoda y práctica en el cuidado de las plantas. Si bien existen posibles mejoras que se pueden implementar en el proyecto, como por ejemplo una carcasa protectora para resguardar la electrónica de este, estas se dejarán como trabajos potenciales a realizar en el futuro.

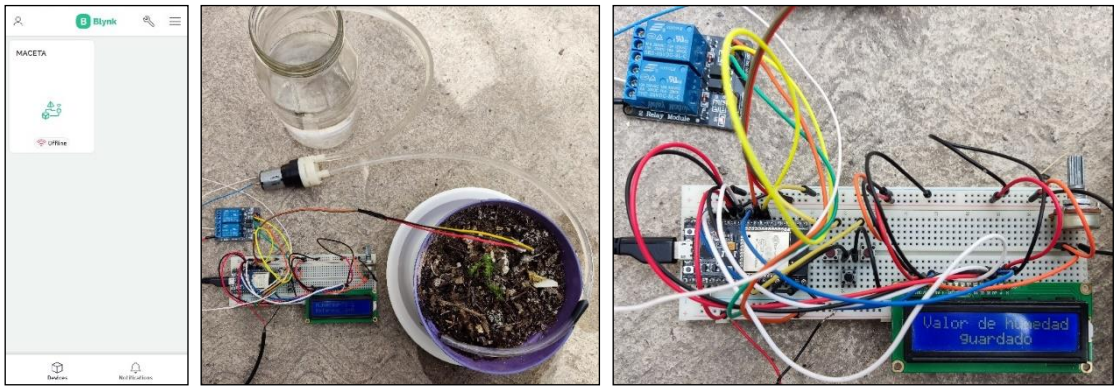


Figura 3.99. Control y almacenamiento del valor de referencia de manera local.

El resultado del experimento es el esperado, dado que ha permitido el monitoreo y control de los niveles de humedad en un sustrato mediante una conexión a internet o de manera local.

3.8 Control de accesos (en línea y local)

El siguiente sketch basa su programa en un control de acceso mediante tarjetas de identificación RFID, utilizando una tarjeta ESP32. El programa tiene como función principal el leer y verificar la identificación de las tarjetas de acceso para permitir el ingreso a una determinada zona, monitoreado mediante la plataforma Blynk.

3.8.1 Diagrama de bloques

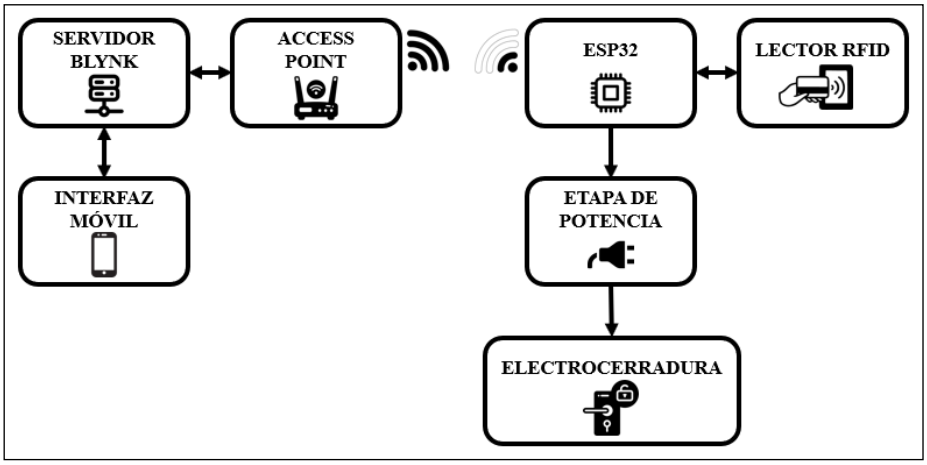


Figura 3.100. Diagrama de bloques, "Control de acceso".

- Interfaz móvil: a través de, es posible realizar el monitoreo y control en la plataforma Blynk, para la creación de la interfaz disponible se requiere de canales de datos como base, por lo tanto, primero se deberán crear estos en el servidor de Blynk lo que permite sincronizar la interfaz con este último.
- Servidor Blynk: permite administrar las configuraciones (como nombres de dispositivos, canales de flujo de datos, alertas o notificaciones, entre otras), que servirán como puente de comunicaciones entre la interfaz, servidor, access point y la tarjeta ESP32.
- Access Point: es el dispositivo que permite establecer un canal de comunicación inalámbrica entre la tarjeta de desarrollo y la plataforma IoT por medio de internet. Este canal normalmente se encuentra cifrado y para obtener la conexión es necesario proporcionar al dispositivo que se conectara las credenciales de acceso (SSID y contraseña).
- ESP32: es la tarjeta de desarrollo que incluye el microcontrolador ESP32, este ya integra un módulo y antenas para realizar la conexión de manera inalámbrica con dispositivos que permiten el acceso a internet. Por medio de esta se realizan todas las operaciones lógicas de comunicación y control, lo cual incluye la gestión de la etapa de potencia y la plataforma de control IoT.
- Lector RFID: sistema de almacenamiento y recuperación de datos remoto sin contacto basada en el uso de lectores mediante la radiación electromagnética de radiofrecuencia para identificar y registrar objetos automáticamente. Emplea dispositivos como etiquetas, tarjetas o tags RFID. El propósito es transmitir la identidad de un objeto (número de serie único) mediante ondas de radio.
- Etapa de potencia: es el circuito electrónico que permite controlar la energía que se requiere en los actuadores de acuerdo con las funcionalidades incorporadas en el sistema.
- Electrocerradura: es el dispositivo que se conecta inmediatamente después de la etapa de potencia, y que en este caso impide el acceso a un área. Para acceder se requiere de energizar o no un solenoide que desbloquea el pestillo u/o electroimán de la puerta donde se encuentra colocada la cerradura. Puede utilizarse una electrocerradura como las que

se muestran en el capítulo 2.4.4. tomado en cuenta el tipo de fuente que se requiere para su alimentación, así como las limitaciones del relevador empleado que permite un flujo máximo de corriente de hasta 10 A y hasta 240 VAC o 30 VDC.

3.8.2 Diagrama esquemático

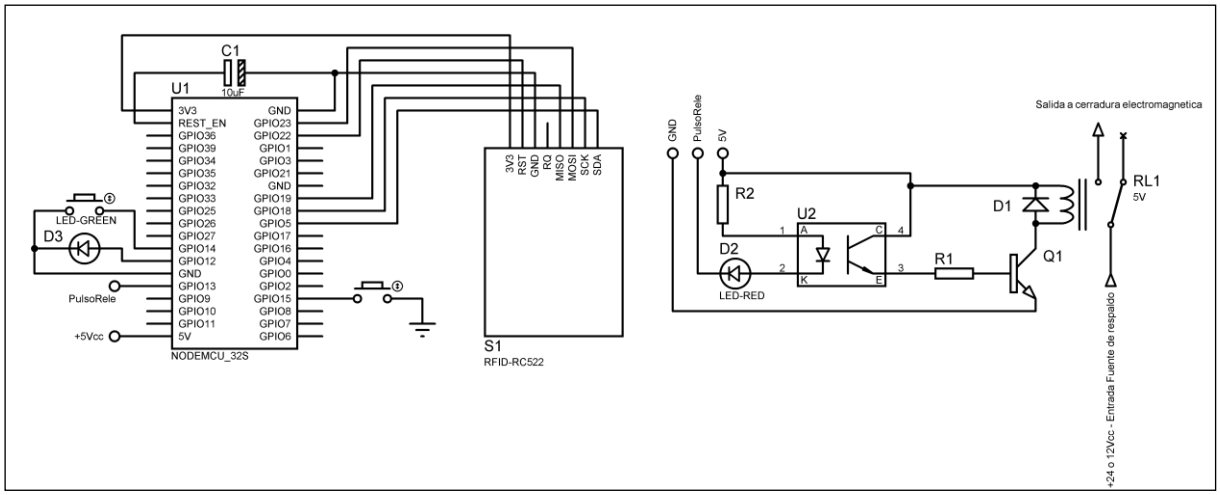


Figura 3.101. Diagrama esquemático para control de acceso.

Material necesario:

- 1 Tarjeta NodeMCU-32S.
- 2 Push button.
- 1 Capacitor 10µF a 50V (opcional).
- 1 Diodo LED.
- 1 Resistor de 220Ω.
- 1 Módulo relevador SRD-5VDC-SL-C.
- 1 Cerradura electromagnética.
- 1 Módulo RFID-RC522.
- Tarjetas o llaveros RFID.
- 1 Fuente de alimentación (consultar el capítulo 2.5.4).

En el diagrama esquemático presentado en la figura 3.101. se muestra la disposición de los elementos para su conexión. Es importante tener en cuenta que tanto el capacitor como el push button son componentes opcionales, puesto que la tarjeta NodeMCU-32S incorpora un push button que puede ser utilizado para ambas funciones, programar la tarjeta y reiniciar las configuraciones para la conexión Wifi de la tarjeta una vez cargada la biblioteca de Blynk, si se mantiene pulsado por diez segundos. Para más detalles, puede consultarse el capítulo 2.2.5 y 3.1.2.2.

En el esquemático se observan las conexiones necesarias para la colocación de un interruptor que permita desbloquear la cerradura de manera manual y local, así como la instalación de un LED indicador para conocer el estado de la cerradura cuando esta es desbloqueada. Para

conectar el lector de tarjetas RFID se utiliza la comunicación SPI, haciendo uso de los pines SCLK o SCK, MISO, MOSI y SS o CS, respectivamente en la tarjeta de desarrollo. El lector RFID se energiza con 3.3 V que pueden ser obtenidos del pin en la tarjeta de desarrollo o alguna otra fuente externa. Adicionalmente, se utiliza el GPIO22 para mandar señal de reinicio al lector y así poder realizar nuevas lecturas de tarjetas. Para la etapa de potencia se utiliza un módulo relé que ha sido descrito en el capítulo anterior 2.5.1, ver alambrado de referencia en figura 3.102.

En cuanto a las conexiones de la electrocerradura, estas se realizan en el módulo de relé como en proyectos anteriores. La señal de control tiene su salida a través del GPIO13. El módulo relé debe ser energizado con 5 V, al igual que la tarjeta de desarrollo, por medio del pin 5V o puerto USB. Para más detalles, puede consultarse los capítulos 2.2.5 y 2.5.4.

La electrocerradura debe ser alimentada con una fuente adecuada que pueda entregar a su salida las características de energía que requiere la cerradura. Esta misma fuente puede ser utilizada para alimentar todos los elementos del proyecto, siendo necesario el uso de reguladores de voltaje o etapas adicionales de rectificación o regulación en el alambrado.

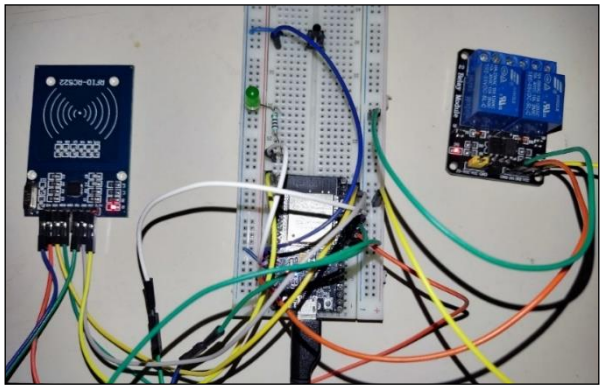
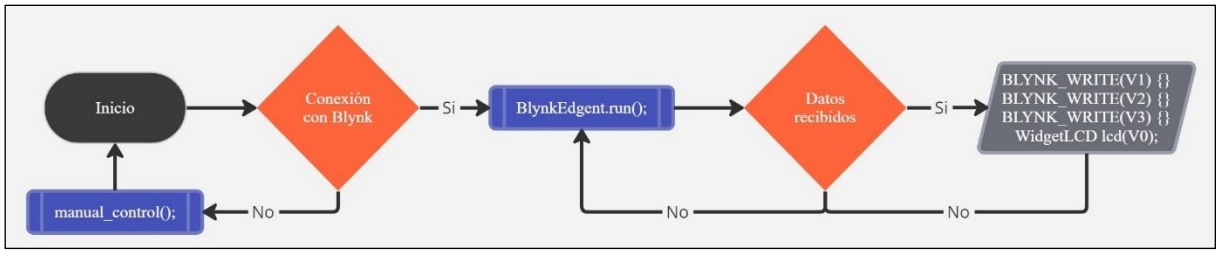
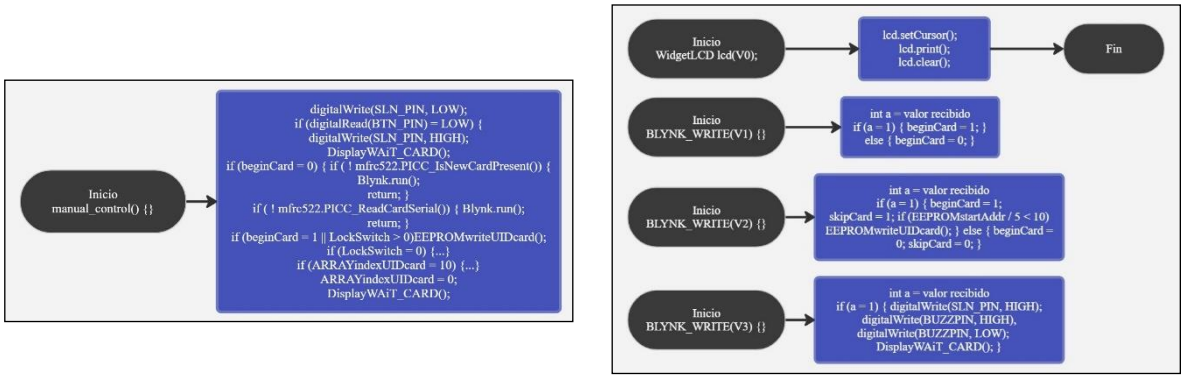


Figura 3.102. Alambrado del circuito para pruebas, control de acceso.

3.8.3 Diagrama de flujo



a)



b) c)

Figura 3.103. a) Diagrama de flujo, programa principal, b) Subrutinas de ejecución local y en línea, c) Subrutinas de control y monitoreo en línea.

El programa permite el control de acceso mediante tarjetas de RFID con el uso de una tarjeta ESP32, utiliza la plataforma Blynk para la configuración y monitoreo remoto. Incluye bibliotecas para la conexión SPI, para el controlador RC522, para el uso de memoria no volátil (se simula el uso de chip EEPROM), así como para las comunicaciones, configuraciones y temporizadores de la plataforma Blynk.

El código comienza con la definición de constantes y variables, incluyendo las identificaciones de plantilla y dispositivo de Blynk, el tamaño de la EEPROM, los GPIO para la conexión del módulo RFID, el GPIO para botón de apertura manual, el GPIO para activación de solenoide de la cerradura, GPIO para zumbador e indicadores de estado.

```
44      void setup() {
45        Serial.begin(115200);
46        delay(100);
47        pinMode(SLN_PIN, OUTPUT); digitalWrite(SLN_PIN, LOW);
48        pinMode(BTN_PIN, INPUT_PULLUP);
49        pinMode(BUZZPIN, OUTPUT);
50        SPI.begin();
51        mfrc522.PCD_Init();
52        lcd.clear();
53        EEPROM.begin(EEPROM_SIZE);
54        DisplayWAiT_CARD();
55        EEPROMreadUIDcard();
56        digitalWrite(BUZZPIN, HIGH), delay(100), digitalWrite(BUZZPIN,
LOW);
57
58        BlynkEdgent.begin();
59      }
```

En la función "`setup()`", se inicializan los GPIO, el módulo RFID, la pantalla LCD (virtual de la plataforma), la memoria EEPROM, se envía la señal al GPIO para mantener la cerradura bloqueada y se llama a la función "`EEPROMreadUIDcard()`" para cargar los datos de las tarjetas

previamente almacenadas. A continuación, se llama a la función "`BlynkEdgent.begin()`" para establecer la conexión con la plataforma Blynk.

El servidor Blynk se comunica con la aplicación móvil a través de los "widgets" virtuales que se asignaron en la aplicación. Cuando un usuario interactúa con los widgets en la aplicación móvil, la aplicación envía comandos al servidor Blynk, que a su vez los transmite al sketch por medio de la tarjeta. Las funciones "`BLYNK_WRITE(V1)`, `BLYNK_WRITE(V2)` y `BLYNK_WRITE(V3)`" son controladores de eventos que se activan cuando los botones virtuales en la aplicación Blynk se presionan y envían datos al dispositivo. Estas funciones cambian los valores de las variables `beginCard`, `skipCard` y `LockSwitch`, y desencadenan diferentes acciones en el programa. Permiten agregar, eliminar y aperturar el acceso por medio de la aplicación móvil, respectivamente.

```
91      void EEPROMwriteUIDcard() {
92
93        if (LockSwitch == 0) {
94          lcd.clear();
95          lcd.print(0, 0, " GUARDAR LLAVES ");
96          lcd.print(0, 1, "ESCANE UNA A UNA");
97          delay(1000);
98        }
99
100       if (LockSwitch > 0) {
101         if (skipCard == 1) {
102           lcd.clear();
103           lcd.print(0, 0, "REMOVER LLAVES-E");
104           lcd.print(0, 1, "                ");
105           lcd.print(0, 1, "   LABEL : ");
106           lcd.print(11, 1, EEPROMstartAddr / 5);
107           EEPROMstartAddr += 5;
108           skipCard = 0;
109         }
110         else {
111           Serial.println("writeCard");
112           EEPROM.write(EEPROMstartAddr, uidDec & 0xFF);
113           EEPROM.write(EEPROMstartAddr + 1, (uidDec & 0xFF00) >> 8);
114           EEPROM.write(EEPROMstartAddr + 2, (uidDec & 0xFF0000) >>
16);
115           EEPROM.write(EEPROMstartAddr + 3, (uidDec & 0xFF000000) >>
24);
116           EEPROM.commit();
117           delay(10);
118           lcd.clear();
119           lcd.print(0, 1, "                ");
120           lcd.print(0, 0, " GUARDADO OK EN ");
121           lcd.print(0, 1, "MEMORIA: ");
122           lcd.print(9, 1, EEPROMstartAddr / 5);
123           EEPROMstartAddr += 5;
124           delay(1000);
125         }
126       }
127
```

```
128         LockSwitch++;
129
130         if (EEPROMstartAddr / 5 == 10) {
131           lcd.clear();
132           lcd.print(0, 0, "SE HAN GUARDADO ");
133           delay(3000);
134           EEPROMstartAddr = 0;
135           uidDec = 0;
136           ARRAYindexUIDcard = 0;
137           EEPROMreadUIDcard();
138         }
139       }
```

La función "`EEPROMwriteUIDcard()`" se llama cuando se requiere agregar una nueva tarjeta al sistema. Esta función primero verifica si se está en modo de guardado o eliminación de tarjetas y muestra mensajes en la pantalla LCD. Luego, si se está en modo de guardado y la tarjeta no es la del administrador, la función escribe el número de identificación de la tarjeta en la memoria EEPROM. Finalmente, si se han guardado todas las tarjetas necesarias, la función muestra un mensaje en la pantalla LCD y llama a "`EEPROMreadUIDcard()`" para cargar los datos actualizados.

```
141       void EEPROMreadUIDcard() {
142         for (int i = 0; i <= 9; i++) {
143           byte val = EEPROM.read(EEPROMstartAddr + 3);
144           CardUIDeEPROMread[ARRAYindexUIDcard] =
(CardUIDeEPROMread[ARRAYindexUIDcard] << 8) | val;
145           val = EEPROM.read(EEPROMstartAddr + 2);
146           CardUIDeEPROMread[ARRAYindexUIDcard] =
(CardUIDeEPROMread[ARRAYindexUIDcard] << 8) | val;
147           val = EEPROM.read(EEPROMstartAddr + 1);
148           CardUIDeEPROMread[ARRAYindexUIDcard] =
(CardUIDeEPROMread[ARRAYindexUIDcard] << 8) | val;
149           val = EEPROM.read(EEPROMstartAddr);
150           CardUIDeEPROMread[ARRAYindexUIDcard] =
(CardUIDeEPROMread[ARRAYindexUIDcard] << 8) | val;
151
152           ARRAYindexUIDcard++;
153           EEPROMstartAddr += 5;
154         }
155
156         ARRAYindexUIDcard = 0;
157         EEPROMstartAddr = 0;
158         uidDec = 0;
159         LockSwitch = 0;
160         DisplayWAiT_CARD();
161       }
```

La función "`EEPROMreadUIDcard()`" carga los números de identificación de las tarjetas previamente almacenadas en la EEPROM y los guarda en el arreglo "`CardUIDeEPROMread[]`". Si no se han almacenado 10 tarjetas en la memoria EEPROM, la función muestra un mensaje en la pantalla LCD que indica cuántas tarjetas se han cargado y cómo agregar más. Si se han

cargado 10 tarjetas, la función muestra un mensaje en la pantalla LCD que indica que se han cargado todas las tarjetas y llama a la función "`DisplayWAiT_CARD()`" para esperar a que se presente una tarjeta.

```
163     void DisplayWAiT_CARD() {
164       lcd.clear();
165       lcd.print(0, 0, " ESCANEAR TARJ. ");
166       lcd.print(0, 1, "   O LLAVE-E    ");
167     }
```

La función "`DisplayWAiT_CARD()`" muestra un mensaje en la pantalla LCD que indica que el dispositivo está listo y en espera de una tarjeta RFID y espera a que se presente una tarjeta.

```
169     void manual_control() {
170     digitalWrite(SLN_PIN, LOW);
171
172       if (digitalRead(BTN_PIN) == LOW) {
173         digitalWrite(SLN_PIN, HIGH);
174         lcd.clear();
175         lcd.print(0, 0, " DBQ. POR BOTON ");
176         lcd.print(0, 1, " PUERTA ABIERTA ");
177         digitalWrite(BUZZPIN, HIGH), delay(200), digitalWrite(BUZZPIN,
LOW);
178         delay(5000);
179         DisplayWAiT_CARD();
180       }
181
182       if (beginCard == 0) {
183         if ( ! mfrc522.PICC_IsNewCardPresent()) {
184           Blynk.run();
185           return;
186         }
187
188         if ( ! mfrc522.PICC_ReadCardSerial()) {
189           Blynk.run();
190           return;
191         }
192       }
193       for (byte i = 0; i < mfrc522.uid.size; i++) {
194         uidDecTemp = mfrc522.uid.uidByte[i];
195         uidDec = uidDec * 256 + uidDecTemp;
196       }
197
198       if (beginCard == 1 || LockSwitch > 0)EEPROMwriteUIDcard();
199
200       if (LockSwitch == 0) {
201         for (ARRAYindexUIDcard = 0; ARRAYindexUIDcard <= 9;
ARRAYindexUIDcard++) {
202           if (CardUIDeEPROMread[ARRAYindexUIDcard] > 0) {
203             if (CardUIDeEPROMread[ARRAYindexUIDcard] == uidDec) {
204               lcd.clear();
205               lcd.print(0, 0, "LLAVE OK ABIERTO");
206               lcd.print(3, 1, uidDec);
```

```
207                 digitalWrite(SLN_PIN, HIGH);
208                 digitalWrite(BUZZPIN, HIGH), delay(200),
digitalWrite(BUZZPIN, LOW);
209                 delay(5000);
210                 break;
211               }
212             }
213           }
214
215           if (ARRAYindexUIDcard == 10) {
216             lcd.clear();
217             lcd.print(0, 0, " LLAVE SIN REG. ");
218             lcd.print(0, 1, "                ");
219             lcd.print(0, 1, "ID : ");
220             lcd.print(5, 1, uidDec);
221             for (int i = 0; i <= 2; i++)delay(100),
digitalWrite(BUZZPIN, HIGH), delay(100), digitalWrite(BUZZPIN, LOW);
222             digitalWrite(SLN_PIN, LOW);
223             delay(2000);
224           }
225
226           ARRAYindexUIDcard = 0;
227           DisplayWAiT_CARD();
228         }
229       }
```

Dentro de la función "`manual_control()`" se encuentran las instrucciones que permiten la espera hasta que se presente una tarjeta RFID, lee su número de identificación y lo compara con los números de identificación de las tarjetas previamente almacenadas. Si la tarjeta se encuentra en la memoria EEPROM, la función desbloquea la cerradura, muestra un mensaje en la pantalla LCD y activa el zumbador durante un breve periodo de tiempo. Si la tarjeta no se encuentra en la memoria EEPROM, la función muestra un mensaje en la pantalla LCD y activa el zumbador durante un periodo de tiempo más largo. Así también contiene la parte del código que permite el desbloqueo de la cerradura por medio del botón físico. Esta función garantiza la ejecución de este fragmento de código sin depender de la comunicación con la plataforma Blynk.

En la función "`loop()`" se llama constantemente a la función "`BlynkEdgent.run()` y `timer.run()`" para mantener la conexión con la aplicación Blynk y los temporizadores de la misma biblioteca, así como la función "`manual_control()`" para retomar la ejecución del código después de haber ejecutado las tareas correspondientes.

```
26
27      void manual_control();
28
29      inline...
```

```
114     void app_loop() {
115         edgentTimer.run();
116         edgentConsole.run();
117         manual_control();
118     }
```

Para la ejecución del programa de manera local, así como en línea se deben realizar las modificaciones de las cabeceras “ConfigMode.h” y “Settings.h”, como se indica en anexo I. Así como adicional agregar la funcion “`manual_control()`” en la cabecera “BlynkEdgent.h”, en la línea de código 27 y 117, para garantizar la ejecución del programa de forma correcta de manera local.

3.8.4 Interfaz móvil

La construcción de la interfaz de control móvil, como se puede observar en la figura 3.104, requiere la creación de una nueva plantilla. En el apartado "Datastreams" se deben crear cuatro canales de datos para tres interruptores y un display LCD virtual que permita conocer las accione que se llevan a cabo del proyecto en todo momento.

La disposición y tamaño de los widgets es completamente personalizable y dependerá del gusto del diseñador. Para los interruptores, es necesario indicar el nombre del widget, el canal de datos y el tipo de interruptor (switch o push). Es importante destacar que los interruptores deben ser configurados como push. Además, se pueden editar las etiquetas, posición, tamaño y color de los widgets.

En cuanto al display LCD, se configura en modo avanzado y se elige el canal de datos, así como el color de fondo y el color de los textos (opcional).

Es importante mencionar que, en este proyecto, no se hace uso de la interfaz web debido a que algunos widgets, como el display LCD, no están disponibles en ella. Asimismo, los interruptores solo pueden ser configurados como push en la interfaz móvil.

Figura 3.104. “Control de acceso”, interfaz móvil.

3.8.5 Pruebas y resultados

El programa se basa en el uso del módulo RC522 en tarjetas de desarrollo que contienen un ESP8266, lo que hace que su integración en la ESP32 sea prácticamente idéntica. Además, utiliza la biblioteca "EEPROM" que, aunque está en desuso, se conservó como ejemplo de implementación en este experimento.

El programa también emplea bibliotecas para la conexión SPI y el control y comunicación con el módulo RC522. Antes de integrar el proyecto con la biblioteca de la plataforma Blynk, se realizaron diversas pruebas con los módulos PN532 y RC522. El último presentó un mejor desempeño, mientras que el PN532 tuvo algunas fallas, ya que solo permitía una única lectura de la tarjeta colocada frente al lector. Se probaron los múltiples modos de comunicación (SPI, I2C, High-speed UART) que ofrece este módulo, sin obtener un resultado satisfactorio.

Una vez elegido el módulo RC522 para el proyecto, se integró la biblioteca de Blynk. Las pruebas incluyeron el uso de un interruptor físico para desbloquear la cerradura electrónica manualmente, así como un interruptor virtual en la interfaz móvil. En la figura 3.105. se puede observar el desbloqueo de la cerradura mediante el botón físico.

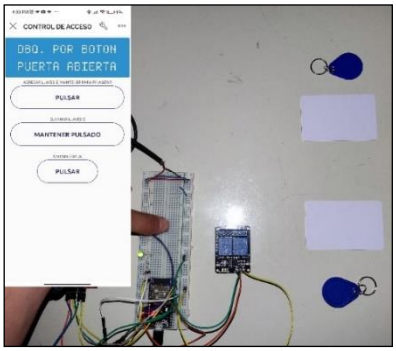
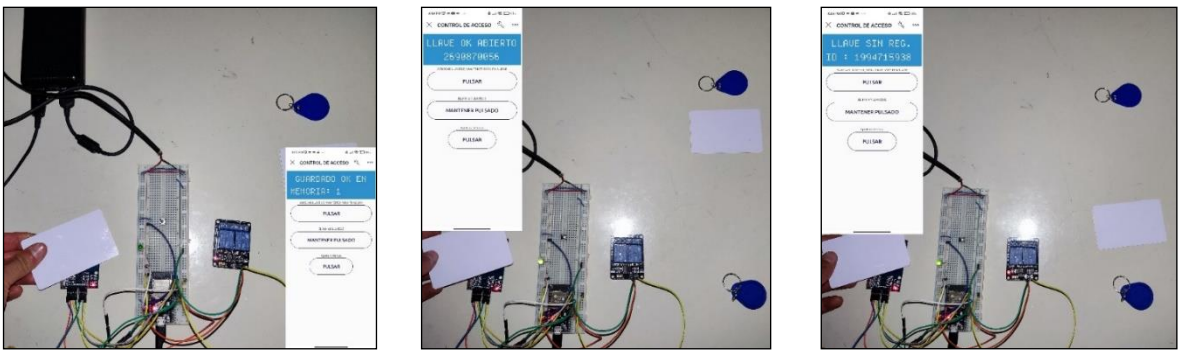


Figura 3.105. Desbloqueo de cerradura botón físico.

La programación del módulo junto con el almacenamiento de los datos obtenidos de las tarjetas RFID permiten la creación de una base de hasta 10 usuarios (siempre incluyendo al primer registro como administrador), los cuales serán identificados y tendrán acceso. En la figura 3.106 se muestra el proceso de registro y lectura de las tarjetas RFID para su identificación y acceso, figura 3.106 (b), así como también negar del acceso a aquellos usuarios que no cuenten con un registro previo, figura 3.106 (c).



a) b) c)

Figura 3.106. Registro y lectura de tarjetas RFID para usuarios.

Se destaca que el proyecto no incorpora una interfaz web debido a que widgets como el display virtual LCD solo están disponibles en la versión móvil y la configuración de los interruptores

como push es una característica solo configurable en la versión móvil. Para agregar, eliminar o modificar usuarios, es necesario que la tarjeta de desarrollo se comunique con la plataforma Blynk. Una vez que se ha registrado las tarjetas RFID, se puede prescindir de la comunicación con la plataforma. Estas modificaciones se han realizado en el programa para permitir su funcionamiento tanto local como en línea, como se ha visto en otros experimentos, ver figura 3.107.

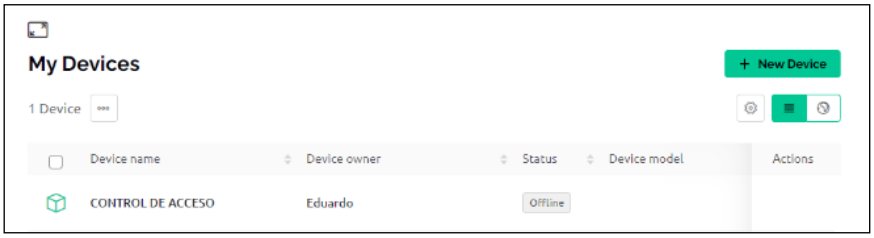
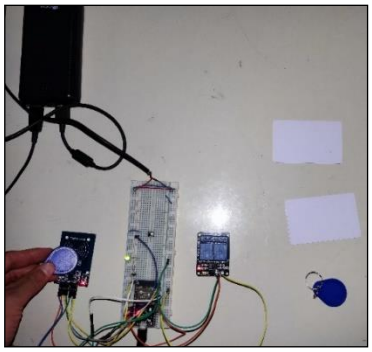


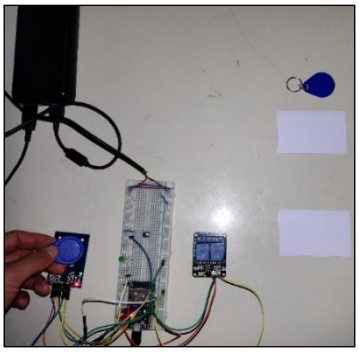
Figura 3.107. Comunicación (fuera de línea) con la plataforma Blynk.

La configuración del programa permite la lectura de tarjetas RFID y la decisión de permitir o no el acceso del usuario basándose en los registros almacenados permanentemente en la memoria no volátil de la ESP32. Es importante destacar que, si ocurre alguna falla que genere la desconexión de la tarjeta de desarrollo de la fuente de alimentación, algún reinicio o la falta de comunicación con el servidor de la plataforma, la tarjeta al ejecutar el programa nuevamente recuperará la información de los datos de acceso de los usuarios almacenados.

En la figura 3.108, se puede observar cómo después de un reinicio y sin comunicación con la plataforma, la tarjeta permite el acceso al tag registrado, figura 3.108 (a), a través del LED indicador verde, mientras que no se realiza la apertura de la cerradura en el tag no registrado, figura 3.108 (b).



a)



b)

Figura 3.108. Tarjeta de desarrollo operando fuera de línea y después de un reinicio.

El resultado del experimento ha cumplido con las expectativas, ya que se ha logrado implementar un sistema que permite el monitoreo y control de acceso de usuarios, tanto de manera local como a través de una conexión a internet.

3.9 Estación meteorológica (en línea)

Este programa está diseñado para el ensamble de una pequeña estación meteorológica basada en la tarjeta ESP32. Su función principal es medir datos meteorológicos como la temperatura, la humedad, la intensidad de luz, la presión barométrica, la velocidad del viento, la intensidad de los rayos UV y enviar los datos a la plataforma Blynk para su visualización en tiempo real.

3.9.1 Diagrama de bloques

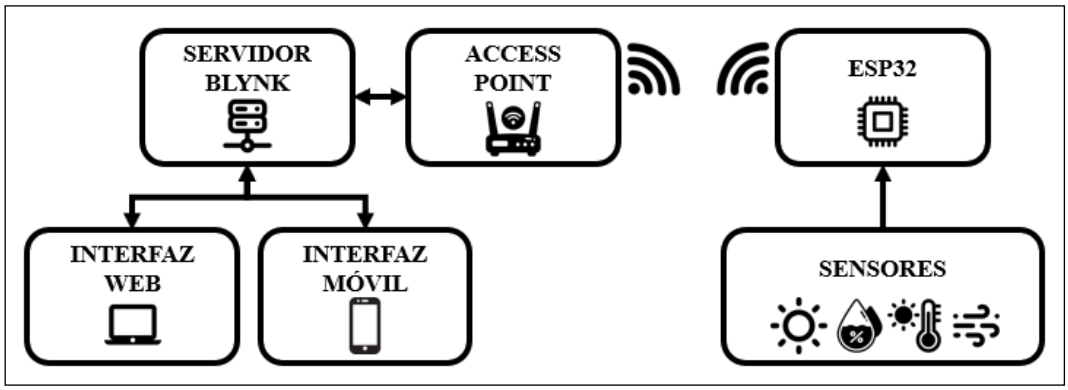


Figura 3.109. Diagrama de bloques, "Estación meteorológica".

- Interfaz web y móvil: a través de, es posible realizar el monitoreo y control en la plataforma Blynk, cabe recalcar que las interfaces son independientes en la creación de su diseño, ambas comparten como base canales de datos, lo que permite sincronizar y los cambios que se realicen en una interfaz sean reflejados en la otra.
- Servidor Blynk: permite administrar las configuraciones, así como visualizar la interfaz web de la plataforma, es en este dónde se crean los canales de datos que servirán como puente de comunicaciones entre las interfaces, el mismo servidor, el access point y la tarjeta ESP32.
- Access Point: es el dispositivo que permite establecer un canal de comunicación inalámbrica entre la tarjeta de desarrollo y la plataforma IoT por medio de internet. Este canal normalmente se encuentra cifrado y para obtener la conexión es necesario proporcionar al dispositivo que se conectara las credenciales de acceso (SSID y contraseña).
- ESP32: es la tarjeta de desarrollo que incluye el microcontrolador ESP32, este ya integra un módulo y antenas para realizar la conexión de manera inalámbrica con dispositivos que permiten el acceso a internet. Por medio de esta se realizan todas las operaciones lógicas de comunicación y control, lo cual incluye la gestión de la etapa de potencia y la plataforma de control IoT.
- Sensores: en este caso electrónicos, son dispositivos diseñados para captar un estímulo de su entorno y traducir esa información que recibe. Esa información recibida es convertida a un impulso eléctrico que posteriormente es procesado para generar una acción predeterminada en un actuador o sistema, esta información que transforma en otra se emplea para medir o controlar.

3.9.2 Diagrama esquemático

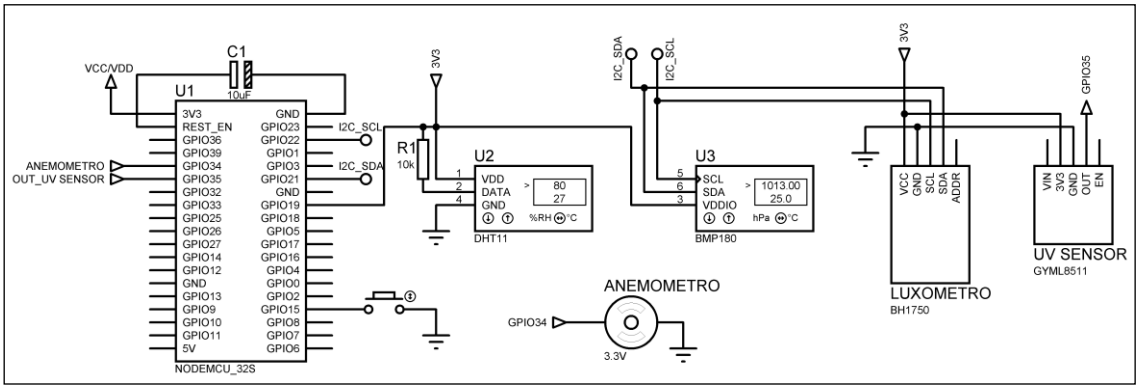


Figura 3.110. Diagrama esquemático para la conexión de los sensores en la estación meteorológica.

Material necesario:

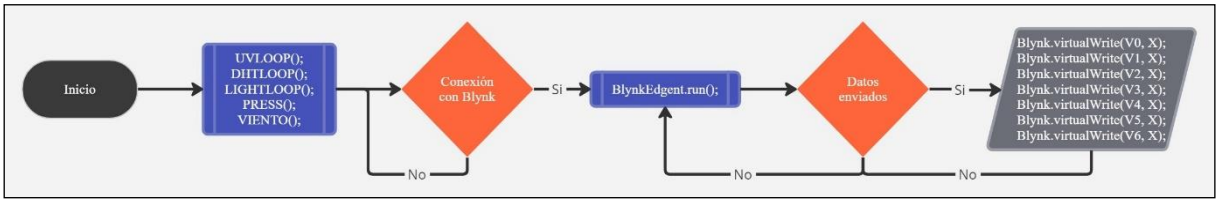
- 1 Tarjeta NodeMCU-32S.
- 1 Push button (opcional).
- 1 Capacitor 10µF a 50V (opcional).
- 1 Resistor de 10kΩ a 1/2 watt.
- 1 Sensor DHT 11.
- 1 Sensor BH1750.
- 1 Sensor GYML8511.
- 1 Sensor BMP180.
- 1 Anemómetro.

En la figura 3.110. del diagrama esquemático se puede apreciar la distribución de los elementos para su conexión, teniendo en cuenta que tanto el capacitor como el push button son componentes opcionales, ya que en la tarjeta NodeMCU-32S se puede emplear el botón integrado para ambas funciones (programar la tarjeta y reiniciar las configuraciones para la conexión Wifi de la tarjeta una vez cargada la biblioteca de Blynk si se mantiene pulsado por diez segundos, ver el capítulo 2.2.5 y 3.1.2.2).

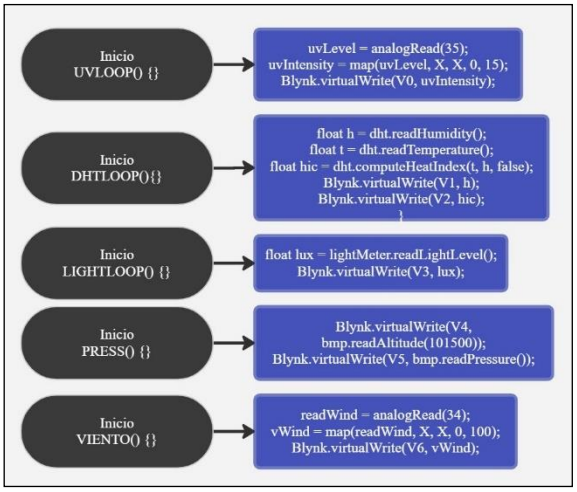
Para alimentar los sensores, estos se conectan directamente al pin 3.3V de la tarjeta de desarrollo, o bien, pueden ser alimentados a través de una fuente externa con un voltaje no mayor a 3.3 volts. Para leer las señales generadas por los sensores se utilizan los GPIO 19, 34 y 35, para el sensor de temperatura y humedad, así como el anemómetro y el sensor de UV, respectivamente, ya que estos proporcionan información de forma analógica. Estos pines se configuran para realizar lecturas analógicas a través de la programación.

Para los sensores de presión e iluminación se emplea la comunicación I2C. Ambos sensores se conectan a los pines SDA y SCL de la tarjeta de desarrollo, diferenciando las lecturas mediante la programación, donde se indica la dirección hexadecimal para cada dispositivo en el bus I2C.

3.9.3 Diagrama de flujo



a)



b)

Figura 3.111. a) Diagrama de flujo, programa principal, b) Subrutinas de monitoreo en línea.

Este programa para una estación meteorológica utiliza una tarjeta ESP32 programada con el IDE Arduino y el servicio Blynk para enviar los datos recopilados por diferentes sensores a la nube.

```
18      #include "BlynkEdgent.h"
19      #include "DHT.h"
20      #include <Wire.h>
21      #include <BH1750.h>
22      #include <Adafruit_BMP085.h>
23
24
25      #define DHTPIN 19
26      #define DHTTYPE DHT11
27
28      DHT dht(DHTPIN, DHTTYPE);
29      BH1750 lightMeter(0x23);
30      Adafruit_BMP085 bmp;
31      BlynkTimer timer;
```

El programa inicia incluyendo las bibliotecas necesarias para realizar las mediciones, tales como "`DHT.h`" para el sensor de temperatura y humedad, "`Wire.h`" para la comunicación por I2C, "`BH1750.h`" para el sensor de luz, "`Adafruit_BMP085.h`" para el sensor de presión barométrica.

Además, se incluye la librería "`BlynkEdgent.h`" para la conexión con la plataforma Blynk, y como en programas anteriores se deben definir también la plantilla y nombre del dispositivo.

A continuación, se definen algunas constantes y variables, como el GPIO utilizado para el sensor DHT, el tipo de sensor utilizado, un objeto "`DHT`", un objeto "`BH1750`" para el sensor de luz, un objeto "`Adafruit_BMP085`" para el sensor de presión barométrica y el uso de temporizadores "`BlynkTimer`".

```
33      void setup() {
34        Serial.begin(115200);
35        dht.begin();
36        Wire.begin();
37        lightMeter.begin();
38        if (!bmp.begin(0x77)){
39        Serial.println("Could not find a valid BMP085 sensor!");
40        while (1) {}
41        }
42        delay(100);
43
44        timer.setInterval(60000L, UVLOOP);
45        timer.setInterval(60000L, DHTLOOP);
46        timer.setInterval(120000L, LIGHTLOOP);
47        timer.setInterval(30000L, PRESS);
48
49        BlynkEdgent.begin();
50      }
```

En la función "`setup()`", se inicia la comunicación serial y se inicializan los sensores. Luego, se configuran los intervalos de lectura y envío de datos a la nube para cada sensor en el temporizador "`BlynkTimer`" y se inicia la conexión con la plataforma Blynk a través de "`BlynkEdgent.begin()`".

En la función "`loop()`", se llaman a las funciones "`UVLOOP()`, `DHTLOOP()`, `LIGHTLOOP()`, `PRESS()` y `VIENTO()`" para realizar las mediciones correspondientes y enviar los datos a la plataforma Blynk mediante las funciones "`Blynk.virtualWrite()`". También se llama a la función "`BlynkEdgent.run()`" para actualizar la conexión con la plataforma Blynk.

```
61      void UVLOOP() {
62        int uvLevel = analogRead(35);
63        int uvIntensity = map(uvLevel, 1160, 2150, 0, 15);
64        Blynk.virtualWrite(V0, uvIntensity);
65      }
```

En "`UVLOOP()`" se realiza la lectura del sensor de rayos UV y se mapea el valor a una escala de 0 a 15, este valor se envía a la nube mediante un pin virtual de Blynk.

```
67      void DHTLOOP() {
68        float h = dht.readHumidity();
69        float t = dht.readTemperature();
70        if (isnan(h) || isnan(t)){
71          Serial.println(F("Failed to read from DHT sensor!"));
72          return;
```

```
73          }
74          float hic = dht.computeHeatIndex(t, h, false);
75          Blynk.virtualWrite(V1, h);
76          Blynk.virtualWrite(V2, hic);
77        }
```

En "DHTLOOP()" se realiza la lectura del sensor de temperatura y humedad y se calcula el índice de calor en grados centígrados, utilizando la fórmula dht.computeHeatIndex. Ambos valores son enviados a la nube mediante dos pines virtuales de Blynk

```
79        void LIGHTLOOP() {
80          float lux = lightMeter.readLightLevel();
81          Blynk.virtualWrite(V3, lux);
82        }
```

En "LIGHTLOOP()" se realiza la lectura de la intensidad de la luz ambiental utilizando el sensor BH1750 y se envía el valor a Blynk en la unidad lux.

```
84        void PRESS() {
85            Blynk.virtualWrite(V4, bmp.readAltitude(101500));
86            Blynk.virtualWrite(V5, bmp.readPressure());
87        }
```

En "PRESS()" se realiza la lectura de presión atmosférica utilizando el sensor Adafruit_BMP085, la lectura del sensor recaba el valor de la presión y la altitud, estos valores son enviados a la plataforma Blynk.

```
89        void VIENTO() {
90          int readWind = analogRead(34);
91          int vWind = map(readWind, 0, 300, 0, 100);
92          Blynk.virtualWrite(V6, vWind);
93        }
```

En "VIENTO()" se realiza la lectura del sensor de velocidad del viento mediante un sensor analógico y se mapea el valor a una escala de 0 a 100.

En resumen, el programa recopila información meteorológica de diferentes variables y las envía a la nube a través del servicio Blynk, permitiendo ser monitoreadas en tiempo real desde cualquier lugar del mundo.

3.9.4 Interfaz web

Para implementar la interfaz web, como se muestra en la figura 3.112, se crean siete canales de datos para indicadores en una nueva plantilla en "Datastreams". Dado que este proyecto se enfoca en el monitoreo de las condiciones climáticas, solo se muestran los datos recopilados por los sensores conectados a la tarjeta de desarrollo en las interfaces. La disposición y tamaño de los widgets es personalizable según las necesidades del diseñador.

En cuanto a los indicadores, se configuran mediante el canal de datos, el nombre del widget y el color. Se puede activar la sobreescritura de los valores si estos superan el valor máximo del widget, así como el cambio del color del indicador en función del valor recibido de los sensores.

Si bien los indicadores varían en sus funciones de personalización de widgets, en la práctica se configuran de manera similar.

Las gráficas se configuran con el canal de datos correspondiente, el valor que se va a graficar (ya sea mínimo, máximo o promedio), el estilo de gráfico, el color, el número máximo de valores registrados por período de tiempo, la capacidad de auto escalar y la visualización del eje "Y" y las etiquetas del gráfico.

Por último, se debe activar la opción "Show time range selector" en la parte superior de la interfaz para generar gráficos con rango de tiempo y así poder consultar los gráficos de hasta una semana de antigüedad en la interfaz (plan gratuito de la plataforma).

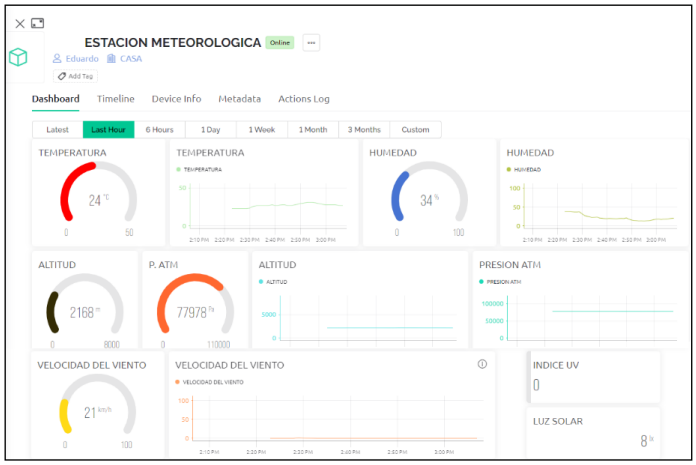


Figura 3.112. "Estación meteorológica", interfaz web

3.9.5 Interfaz móvil

La interfaz móvil, al igual que la interfaz web, ha sido diseñada para mostrar únicamente la información que se recibe de los sensores en el proyecto. Para configurar los indicadores, se establecen el canal de datos, el nombre del widget y el color. Aunque se eliminan algunas funciones de personalización del widget en algunos casos, en general todos los indicadores se configuran de la misma manera.

En cuanto a las gráficas, se configuran con el canal de datos correspondiente y el valor que se graficará con escalado automático (o también se dispone de las opciones: Min/Max, Height y Delta). La resolución de los datos obtenidos puede ser de hasta 1 año (a diferencia de la interfaz web, se requiere un plan de pago para almacenar los datos por un período de tiempo mayor a una semana). También se puede elegir el estilo de gráfico (con múltiples opciones de visualización dependiendo del estilo seleccionado), color, auto escalado del gráfico, tamaño, color y posición de etiquetas, así como visualizar el eje "X" (tiempo) y las leyendas del gráfico. Además, se puede habilitar el botón para visualizar el gráfico a pantalla completa. Es importante destacar que todas las interfaces móviles solo se pueden diseñar en formato vertical. En la figura 3.113, (la captura de la interfaz se ha dividido en dos imágenes para mostrarla de manera íntegra).

Figura 3.113. Interfaz móvil "Estación Meteorológica".

3.9.6 Pruebas y resultados

Durante la realización del proyecto se llevaron a cabo diversas pruebas con los diferentes sensores integrados. Se programó la tarjeta de desarrollo de manera individual para obtener lecturas los más precisas posibles y calibrar la información recabada. La plataforma Blynk es la única que permite la visualización de esta información en ambas versiones de la interfaz. Cabe mencionar que, debido a la falta de un anemómetro, se construyó uno basado en un motor de corriente continua que funciona como dinamo y genera un voltaje al ser impulsado por el viento. Sin embargo, no se cuenta con un sistema de referencia para su calibración exacta, como se explica en el capítulo 2.3.10.

Además, se llevaron a cabo pruebas con los sensores BH1750 para medir el nivel de iluminación y distinguir si es de día o de noche, el DHT11 para medir la humedad del ambiente y la temperatura con un margen de error del 1% al 2%, como se muestra en la figura 3.114.

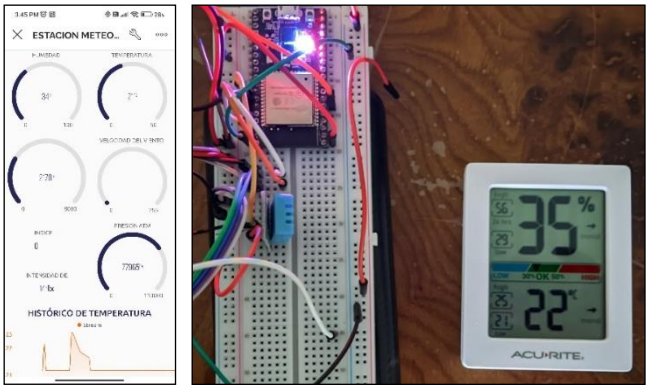


Figura 3.114. Pruebas de medición temperatura y humedad.

También se utilizó el sensor BMP180 para medir la presión atmosférica y calcular la altitud de la estación meteorológica, así como el sensor GY-ML8511 para conocer la cantidad de radiación UV que emite el sol durante el día, como se observa en la figura 3.115.

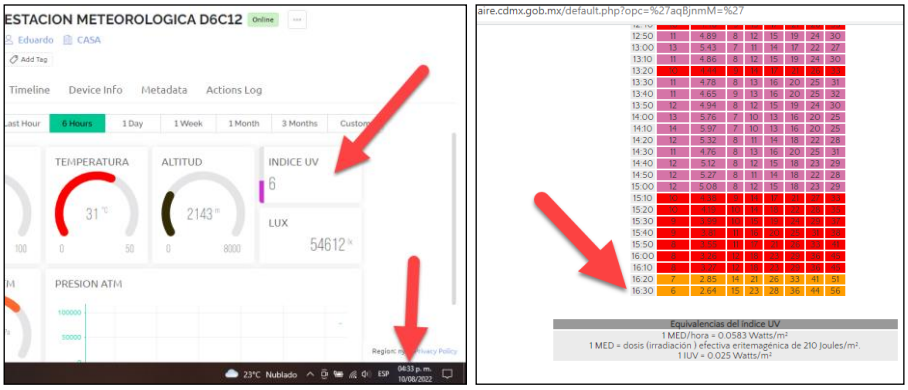


Figura 3.115. Radiación UV medición y reporte en portal del gobierno de la CDMX.

Las pruebas realizadas llevaron a la utilización de temporizadores disponibles en la biblioteca de la plataforma Blynk. El objetivo del proyecto es monitorear las condiciones medioambientales en tiempo real en la zona en que se encuentra la estación meteorológica, como se muestra en la figura 3.116. Sin embargo, se observó que las mediciones llegaban con demasiada rapidez al servidor y algunas de ellas no podían ser procesadas por el mismo, por lo que se optó por el uso de temporizadores que limitan el tiempo de envío de las lecturas al servidor. Además, se constató que la ejecución del programa seguía siendo muy rápida, pero la capacidad del microcontrolador ESP32 es suficiente para realizar las tareas sin inconvenientes.

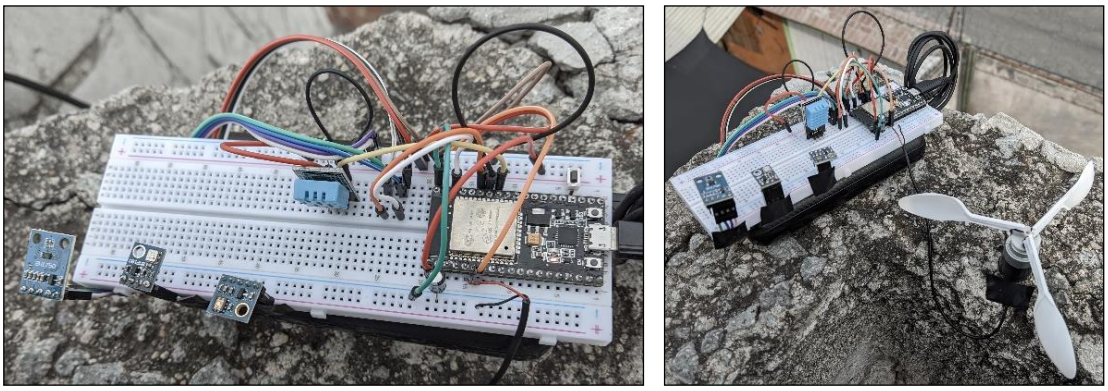


Figura 3.116. Estación monitoreando condiciones ambientales.

Los resultados obtenidos en el experimento son los esperados, ya que se ha comprobado que es posible monitorear las condiciones meteorológicas en tiempo real a través de una conexión a internet.

CONCLUSIONES

El uso de una tarjeta basada en el microcontrolador ESP32, se mostró apta para manejar todas las tareas de adquisición y procesamiento de la información, así como para la comunicación vía internet con las plataformas IoT, para la programación se decidió trabajar con el IDE de Arduino, lo cual permitió aprovechar la basta información y soporte disponible en la red, lo cual ayuda al desarrollo, ya que existen bibliotecas para todos los sensores incluidos en este trabajo.

Se realizaron experimentos que permitieron adquirir todas las variables propuestas (luz, temperatura, humedad ambiental, humedad en sustratos, nivel de agua, presencia, presión atmosférica, altitud e índice de radiación ultravioleta), cabe mencionar que, para el caso de la humedad en sustratos, los sensores que miden directamente dicha variable son costos, por lo que se utilizó un sensor de conductividad eléctrica del agua, que, aunque no mide directamente la humedad, si puede ser de utilidad. En términos generales, se utilizaron sensores y transductores de costo asequible, y tuvieron un desempeño a lo esperado de acuerdo con las pruebas realizadas.

Se realizó un sistema que mediante la lectura de tarjetas RFID permite la activación de una cerradura electrónica, la cual tiene capacidad de ser abierta con una llave convencional, y puede ser montada en puertas interiores o exteriores.

Para el desarrollo de los proyectos se realizaron pruebas con algunas de las plataformas que ofrecen servicios de forma gratuita, llegando a la conclusión de que la plataforma que brinda una cantidad mayor de servicios para implementación (siendo aún limitada en planes gratuitos), así como poseer de una programación e integración menos compleja, es la plataforma Blynk, esta permite crear interfaces (tanto en un servidor web, como en una aplicación móvil) para el monitoreo y control de variables, registro histórico de las mismas, alertas y notificaciones.

Aprovechando las capacidades de la plataforma Blynk, se desarrollaron tableros (interfaces) que permiten el monitoreo de nivel de agua, iluminación, humedad, temperatura, altitud, presión atmosférica, índice de radiación ultravioleta y presencia. Para el caso particular de la humedad en sustratos, se desarrolló un algoritmo que permite que sea mantenido de manera automática a un nivel de humedad determinado por el usuario mediante la interfaz de Blynk. Para el nivel de agua, además del monitoreo, se incluye la posibilidad de activar la bomba de agua de manera automática o manual, esto último se hace mediante la aplicación o directamente desde un interruptor colocado en el circuito. Para el caso de las luminarias, la interfaz incluye opciones para encender, apagar o establecer un nivel y color de iluminación a gusto del usuario. Respecto al control de accesos, además de tener una funcionalidad local, se incorporó la opción de apertura remota desde la aplicación de Blynk.

Para lograr la integración de la plataforma Blynk, fue necesario agregar su respectiva biblioteca; para la programación, se requiere que para cada lectura o escritura de variable al servidor de Blynk se obtengan credenciales, para ello se ocupa una función, y que de no lograrse el código no continua, por lo que representa un problema, particularmente en las aplicaciones de control, para solventar esta situación fue necesario modificar una de las funciones de la biblioteca, por lo que ahora, cuando no se logra comunicación con el servidor, solo se mantiene unos segundos en espera, lo cual permite continuar con los algoritmos de monitoreo y control del sistema.

Para los experimentos donde se realiza el control mediante asistentes virtuales se incluyó el uso de dos plataformas IoT gratuitas, que se integran a asistentes como el de Google y Alexa. Sinric Pro y ESP RainMaker ofrecen una integración con ambos asistentes. En estas plataformas encontramos que las capacidades de monitoreo y control son muy limitadas en sus versiones gratuitas, y la programación resulta ser más compleja.

Una forma viable de control de la plataforma Blynk por medio de asistentes virtuales se describe en capítulo 3.3.4.1.2, para el término del presente trabajo de investigación dejo de estar disponible debido a la actualización de las plataformas IFTTT y Blynk, esta última ofreciendo la interacción con asistentes similar a la de las plataformas Sinric Pro y ESP RainMaker pero solo en planes de pago.

Por lo tanto, con toda la información y desarrollo de sistemas presentados en este trabajo se considera que los objetivos planteados fueron plenamente satisfechos; lo cual no implica que no existan oportunidades de mejora, por lo que a continuación se mencionan los trabajos que podrían desarrollarse en un futuro a partir de lo aquí presentado.

Incluir más variables como sensores de gas, consumo eléctrico, por mencionar algunas, así como la incorporación de cámaras de video, También agregar más sistemas de control como cierre de llaves de gas, agua, etcétera.

Otra mejora importante corresponde al desarrolla de una plataforma IoT propia ya que depender de terceros siempre involucra inconvenientes como la modificación/limitación de servicios, costos y problemas de privacidad y seguridad.

También realizar tarjetas de circuitos para tener una mejor presentación, lo que también lleva a realizar diseños de embalaje de las tarjetas para su protección y mejora de aspecto.

REFERENCIAS

(Alias), C. (5 de febrero de 2020). *RogerBit "The Power Of The Bit"*. Recuperado el 22 de septiembre de 2022, de https://rogerbit.com/wprb/2020/02/medidor-de-nivel-de-agua-iot-con-uln2803-y-esp32-subiendo-datos-a-ubidots/

(Alias), I. (24 de abril de 2020). *Hardwarelibre*. (AB Internet Networks 2008 S.L.) Recuperado el 20 de septiembre de 2022, de https://www.hwlibre.com/uln2803/

(Alias), l. (27 de junio de 2022). *ESPRESSIF Discussion Forum*. Recuperado el 23 de septiembre de 2022, de https://esp32.com/viewtopic.php?t=28570

© 2013-2022 Microcontrollerslab.com. (17 de agosto de 2022). *MicrocontrollersLab.* Recuperado el 30 de septiembre de 2022, de https://microcontrollerslab.com/esp-rainmaker-tutorial-esp32-arduino-ide/

© 2014 ROHM Co. (6 de enero de 2013). *MOUSER ELECTRONICS.* Recuperado el 1 de agosto de 2022, de https://www.mouser.com/datasheet/2/348/bh1750fvi-e-186247.pdf

© 2021 www.masterlogistica.es. (27 de diciembre de 2018). *Máster Logística.* Recuperado el 10 de octubre de 2022, de http://www.masterlogistica.es/tecnologia-rfid-radio-frequency-identification-lectores-ondas-de-radio/

© Geek Factory 2013 - 2022. (4 de octubre de 2021). *Geek Factory*. Recuperado el 23 de septiembre de 2022, de https://www.geekfactory.mx/tienda/componentes/interruptores/push-button-estandar-4-pines-pulsador-boton/

© Sinric 2019-2022. (5 de enero de 2018). *Sinric Pro.* Recuperado el 30 de septiembre de 2022, de https://portal.sinric.pro/register

© Sinric 2019-2022. (11 de septiembre de 2022). *Sinric Pro documentation.* Recuperado el 30 de septiembre de 2022, de https://help.sinric.pro/pages/pass

©2021 Naylamp Mechatronics SAC. (28 de enero de 2015). *NAYLAMP MECHATRONICS.* Recuperado el 7 de septiembre de 2022, de https://naylampmechatronics.com/sensores-temperatura-y-humedad/57-sensor-de-temperatura-y-humedad-relativa-dht11.html

©2021 Naylamp Mechatronics SAC. (18 de julio de 2015). *NAYLAMP MECHATRONICS.* Recuperado el 8 de septiembre de 2022, de https://naylampmechatronics.com/sensores-posicion-inerciales-gps/75-sensor-de-presion-bmp180.html

©2021 Naylamp Mechatronics SAC. (11 de agosto de 2015). *NEYLAMP MECHATRONICS.* Recuperado el 8 de septiembre de 2022, de https://naylampmechatronics.com/sensores-luz-y-sonido/169-modulo-sensor-de-luz-ultravioleta-uv-ml8511.html

©2021 Naylamp Mechatronics SAC. (1 de mayo de 2018). *NAYLAMP MECHATRONICS.* Recuperado el 7 de septiembre de 2022, de https://naylampmechatronics.com/rfid-nfc/182-modulo-lector-rfid-nfc-1356mhz-pn532.html

©2021 Naylamp Mechatronics SAC. (30 de abril de 2018). *NAYLAMP MECHATRONICS.* Recuperado el 7 de septiembre de 2022, de https://naylampmechatronics.com/rfid-nfc/80-modulo-lector-rfid-1356mhz-rc522.html

©2021 Naylamp Mechatronics SAC. (1 de mayo de 2018). *NAYLAMP MECHATRONICS.* Recuperado el 7 de septiembre de 2022, de https://naylampmechatronics.com/sensores-temperatura-y-humedad/354-sensor-de-temperatura-y-humedad-relativa-dht21-am2301.html

©2021 Naylamp Mechatronics SAC. (1 de mayo de 2018). *NAYLAMP MECHATRONICS.* Recuperado el 8 de septiembre de 2022, de https://naylampmechatronics.com/sensores-posicion-inerciales-gps/358-sensor-de-presion-bmp280.html

©2021 Naylamp Mechatronics SAC. (2 de enero de 2019). *NAYLAMP MECHATRONICS.* Recuperado el 7 de septiembre de 2022, de https://naylampmechatronics.com/sensores-temperatura-y-humedad/58-sensor-de-temperatura-y-humedad-relativa-dht22-am2302.html

©2021 Naylamp Mechatronics SAC. (s.f.). *NAYLAMP MECHATRONICS.* Recuperado el 1 de agosto de 2022, de https://naylampmechatronics.com/blog/44_tutorial-modulo-sensor-de-luz-bh1750.html

2017 OCompra.com. (s.f.). *OCompra.* Recuperado el 27 de septiembre de 2022, de https://www.ocompra.com/mexico/item/cerradura-magnetica-160-kg-anviz-an-ml160-plantronics-vc-1400577792/

(3 de julio de 2021). (Fundación Wikimedia, Inc.) Recuperado el 5 de agosto de 2022, de https://es.wikipedia.org/wiki/Mebibyte

Adafruit Industries, LLC. (9 de julio de 2011). *adafruit.* Recuperado el 8 de septiembre de 2022, de https://www.adafruit.com/product/391

Adafruit Industries, LLC. (22 de marzo de 2014). *adafruit.* Recuperado el 8 de septiembre de 2022, de https://www.adafruit.com/product/1777

Adafruit Industries, LLC. (11 de junio de 2014). *adafruit.* Recuperado el 9 de septiembre de 2022, de https://www.adafruit.com/product/1918

Adafruit Industries, LLC. (9 de abril de 2014). *adafruit.* Recuperado el 12 de septiembre de 2022, de https://www.adafruit.com/product/1733

Adafruit Industries, LLC. (2 de octubre de 2017). *adafruit.* Recuperado el 30 de septiembre de 2022, de https://io.adafruit.com/

Adafruit Industries, LLC. (26 de septiembre de 2018). *adafruit.* Recuperado el 8 de septiembre de 2022, de https://www.adafruit.com/product/3964

Agustin, H. (10 de marzo de 2015). *TALLERELECTRONICA.COM / BLOG*. Recuperado el 20 de septiembre de 2022, de https://tallerelectronica.com/2015/03/10/indicador-electronico-de-nivel-de-agua/

Alonso, R. (2 de agosto de 2022). *HZ hardzone.* Recuperado el 20 de septiembre de 2022, de https://hardzone.es/tutoriales/compras/oled-vs-qled-cual-comprar/

Amazon.com, Inc. (4 de noviembre de 2019). *Amazon Skills de Alexa.* Recuperado el 30 de septiembre de 2022, de https://www.amazon.com.mx/HOME-Sinric-Pro/dp/B07ZT5VDT8

Arduino S.r.l. (1 de febrero de 2022). *ARDUINO CLOUD.* Recuperado el 30 de septiembre de 2022, de https://cloud.arduino.cc/how-it-works

Arduino S.r.l. (1 de febrero de 2022). *ARDUINO CLOUD.* Recuperado el 30 de septiembre de 2022, de https://cloud.arduino.cc/plans

BANCO BILBAO VIZCAYA ARGENTARIA, S.A. (3 de septiembre de 2021). *BBVA.* Recuperado el 9 de agosto de 2022, de https://www.bbva.com/es/sostenibilidad/domotica-e-inmotica-son-las-mismas-tecnologias-inteligentes/

Blynk, Inc. (8 de agosto de 2016). *Blynk.* Recuperado el 30 de septiembre de 2022, de https://blynk.io/pricing

Blynk, Inc. (8 de agosto de 2016). *Blynk.* Recuperado el 1 de octubre de 2022, de https://blynk.io/pricing

Blynk, Inc. (18 de abril de 2017). *Blynk.* Recuperado el 1 de octubre de 2022, de https://blynk.io/iot-platform-for-business

Blynk, Inc. (22 de mayo de 2021). *Blynk.* Recuperado el 1 de octubre de 2022, de https://blynk.io/blog/meet-the-new-blynk-platform

Blynk, Inc. (19 de marzo de 2021). *Blynk.* Recuperado el 1 de octubre de 2022, de https://blynk.io/blog/new-blynk-iot-platform-beta

Blynk, Inc. (10 de febrero de 2021). *Blynk Documentation.* Recuperado el 30 de septiembre de 2022, de https://docs.blynk.io/en/

Blynk, Inc. (14 de abril de 2021). *Blynk Documentation.* Recuperado el 30 de septiembre de 2022, de https://docs.blynk.io/en/concepts/device

Blynk, Inc. (5 de febrero de 2021). *Blynk Documentation.* Recuperado el 1 de octubre de 2022, de https://docs.blynk.io/en/blynk.edgent-firmware-api/supported-boards

Carmenate, J. G. (15 de febrero de 2021). *programarfacil.com*. (LUIS DEL VALLE HERNÁNDEZ) Recuperado el 5 de agosto de 2022, de https://programarfacil.com/esp8266/esp32/

Carmenate, J. G. (2 de marzo de 2021). *programarfacil.com*. (LUIS DEL VALLE HERNÁNDEZ) Recuperado el 23 de septiembre de 2022, de https://programarfacil.com/esp8266/programar-esp32-ide-arduino/

Carrod Electrónica Online S. de R.L. de C.V. (23 de diciembre de 2014). *carrod electrónica.* Recuperado el 3 de noviembre de 2022, de https://www.carrod.mx/products/triac-16-a-600-v-to220-bta16-600b

Carrod Electrónica Online S. de R.L. de C.V. (23 de diciembre de 2014). *carrod electrónica.* Recuperado el 3 de noviembre de 2022, de https://www.carrod.mx/products/optoacoplador-moc3021-salida-triac

Carrod Electrónica Online S. de R.L. de C.V. (17 de agosto de 2015). *carrod electrónica.* Recuperado el 3 de noviembre de 2022, de https://www.carrod.mx/products/optoacoplador-4n25-salida-transistor

Cenicaña. (3 de julio de 2018). (Centro de Investigación de la Caña de Azúcar, Cenicaña.) Recuperado el 12 de septiembre de 2022, de https://www.cenicana.org/abc-de-los-sensores-de-humedad-en-las-labores-de-riego/

Cetronic, S.L. (22 de agosto de 2018). *Cetronic Componentes Electrónicos.* Recuperado el 2 de agosto de 2022, de https://descargas.cetronic.es/SENSORLUZADAFRUIT.pdf

Cetronic, S.L. (22 de agosto de 2018). *Cetronic Componentes Electrónicos.* Recuperado el 2 de agosto de 2022, de https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=999334022&cPath=1343

circuits4you.com. (31 de diciembre de 2018). (circuits4you.com) Recuperado el 26 de septiembre de 2022, de https://circuits4you.com/2018/12/31/esp32-pwm-example/

Cooper, J. (6 de septiembre de 2017). *Adafruit*. (Adafruit Industries) Recuperado el 3 de agosto de 2022, de https://learn.adafruit.com/adafruit-apds9960-breakout

Didácticas Electrónicas. (23 de octubre de 2016). *I+D Electrónica.* Recuperado el 2 de agosto de 2022, de https://www.didacticaselectronicas.com/index.php/sensores/luz/sensor-de-luz-ambiente-temt6000-temt6000-sensores-de-luz-ambiental-ambiente-temt6000-detail

Diéguez, L. (4 de mayo de 2020). *Kolwidi.* Recuperado el 29 de septiembre de 2022, de https://kolwidi.com/blogs/blog-kolwidi/9-plataformas-iot-para-arduino-y-proyectos-de-iot

Diosdado, R. (14 de abril de 2014). *Zona Maker*. (© 2023 ZonaMaker) Recuperado el 8 de marzo de 2023, de https://www.zonamaker.com/electronica/intro-electronica/instrumentacion/fuente-de-alimentacion

Diosdado, R. (5 de mayo de 2016). *Zona Maker.* Recuperado el 21 de septiembre de 2022, de https://www.zonamaker.com/arduino/modulos-sensores-y-shields/tipos-de-lcd-para-arduino

DIPRO TECHNOLOGIES S.L. (25 de noviembre de 2020). *thethings.io.* Recuperado el 30 de septiembre de 2022, de https://thethings.io/new-platform-plans/

Domínguez, M. (3 de febrero de 2022). *caloryfrio.com el portal sectorial de las instalaciones.* Recuperado el 14 de octubre de 2022, de https://www.caloryfrio.com/calefaccion/herramientas-y-regulacion/que-es-la-domotica-y-como-funciona-una-casa-domotica.html

Eduardo Julio Carletti. (8 de julio de 2020). *https://robots-argentina.com.ar/*. Obtenido de robots-argentina.com: https://robots-argentina.com.ar/didactica/modulos-de-rele-y-arduino-domotica-1/

El Gran Tlapalero S.A. de C.V. (18 de abril de 2023). *EL GRAN TLAPALERO*. Obtenido de www.elgrantlapalero.com: https://www.elgrantlapalero.com/cerradura-electrica-12v-cilindro-clasico-sin-boton-acero-inoxidable-soprano.html

Electrobombas Jávea, S.L. (19 de septiembre de 2020). *ELECTROBOMBAS JÁVEA TECNOLOGÍAS DEL AGUA.* Recuperado el 26 de septiembre de 2022, de https://electrobombasjavea.com/blog/que-es-una-bomba-de-agua-y-como-funciona-una-electrobomba

ES ELECTROSTORE. (8 de diciembre de 2020). (© Electrostore 2019) Recuperado el 12 de septiembre de 2022, de https://grupoelectrostore.com/shop/sensores/temperatura/modulo-sensor-de-humedad-de-suelo-hd-38-higrometro-anticorrosivo/

Espressif. (24 de mayo de 2022). *ESP32 Arduino Core's documentation.* Recuperado el 26 de septiembre de 2022, de https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/preferences.html

Espressif Systems (Shanghai) Co., Ltd. (14 de mayo de 2021). *ESP RAINMAKER®.* Recuperado el 30 de septiembre de 2022, de https://rainmaker.espressif.com/

ESTUDIOS EUROPEOS DE POSTGRADO Y EMPRESA, S.L. (s.f.). *CEUPE magazine.* Recuperado el 7 de septiembre de 2022, de https://www.ceupe.com/blog/que-es-el-rfid.html?dt=1662935186005

Euroinnova Formación S.L. (mayo de 2022). *EUROINNOVA INTERNATIONAL ONLINE EDUCATION*. Recuperado el 27 de septiembre de 2022, de https://www.euroinnova.edu.es/blog/como-funcionan-las-bombas-de-agua

FactorLED. (23 de enero de 2018). *factorled.* Recuperado el 20 de septiembre de 2022, de https://www.factorled.com/blog/es/tipos-de-diodos-led-caracteristicas-tecnicas-y-formatos/

Fernández, S. (3 de octubre de 2021). *Xataka móvil.* Recuperado el 21 de septiembre de 2022, de https://www.xatakamovil.com/multimedia/lcd-oled-amoled-todos-tipos-pantalla-que-puede-tener-movil-sus-diferencias

Giraldo, S. A. (23 de diciembre de 2019). *Control Automático Educación*. (Control Automático Educación) Recuperado el 26 de septiembre de 2022, de https://controlautomaticoeducacion.com/arduino/eeprom-con-arduino/

Google LLC. (3 de octubre de 2022). *Google Cloud.* Recuperado el 30 de septiembre de 2022, de https://cloud.google.com/solutions/iot/

Guazzaroni, C. (25 de marzo de 2017). *Espacio de Cesar.* Recuperado el 12 de septiembre de 2022, de https://espaciodecesar.com/2017/03/25/como-medir-el-viento-muy-facil-anemometro-simple-y-con-arduino/

Guirado, J. F. (22 de julio de 2020). *Descubrearduino.com.* Recuperado el 5 de diciembre de 2022, de https://descubrearduino.com/esp32-modulo-esp32-wroom-gpio-pinout/

Hernández, L. d. (28 de noviembre de 2016). *Programarfacil.com.* Recuperado el 29 de septiembre de 2022, de https://programarfacil.com/podcast/proyectos-iot-con-arduino/

Herramientas Tecnológicas Profesionales, S. de R.L. de C.V. (28 de enero de 2018). *HETPRO.* Recuperado el 2 de agosto de 2022, de https://hetpro-store.com/TUTORIALES/sensor-de-luz-tsl2560/

IFTTT Inc. (16 de marzo de 2017). *IFTTT.* Recuperado el 30 de septiembre de 2022, de https://ifttt.com/plans

International Business Machines Corp. (2 de octubre de 2022). *IBM.* Recuperado el 2 de octubre de 2022, de https://www.ibm.com/cloud/internet-of-things

International Business Machines Corp. (2 de octubre de 2022). *IBM.* Recuperado el 2 de octubre de 2022, de https://cloud.ibm.com/registration?target=/catalog/%3Fcategory%3Diot

Lacasta, L. M. (18 de julio de 2016). *LUIS LLAMAS Ingeniería, informática y diseño.* Recuperado el 5 de diciembre de 2022, de https://www.luisllamas.es/arduino-transistor-mosfet/

Leaf Group Ltd. (3 de septiembre de 2013). *techlandia.* Recuperado el 23 de septiembre de 2022, de https://techlandia.com/reparar-cargador-taladro-como_101973/

Littelfuse, Inc. (23 de marzo de 2016). *Littelfuse Expertise Applied | Answers Delivered.* Recuperado el 23 de septiembre de 2022, de https://www.littelfuse.com/technical-resources/technical-centers/commercial-vehicle-technical-center/poles-and-throws.aspx

Lourdes Martin Lacasta. (8 de enero de 2017). *LUIS LLAMAS - Ingeniería, informática y diseño.* Recuperado el 03 de noviembre de 2022, de https://www.luisllamas.es/arduino-optoacoplador/

LUIS LLAMAS Ingeniería, informática y diseño. (23 de enero de 2018). (Lourdes Martin Lacasta) Recuperado el 3 de agosto de 2022, de https://www.luisllamas.es/detectar-gestos-con-arduino-y-sensor-apds-9960/

Manuel Hidalgo Díaz, J. M. (4 de febrero de 2016). *Practicas con Arduino*. (Ministerio de Educación, Cultura y Deporte, España) Recuperado el 1 de agosto de 2022, de http://www.practicasconarduino.com/manualrapido/detector_de_luz.html

Mecatrónica LATAM. (5 de mayo de 2021). *MecatrónicaLATAM*. (© Mecatrónica LATAM) Recuperado el 3 de agosto de 2022, de https://www.mecatronicalatam.com/es/tutoriales/sensores/sensor-de-proximidad/

Microsoft S. de R.L de C.V. (5 de diciembre de 2017). *Azure.* Recuperado el 30 de septiembre de 2022, de https://azure.microsoft.com/es-mx/pricing/details/iot-central/

NAYLAMP MECHATRONICS. (19 de enero de 2016). (©2021 Naylamp Mechatronics SAC) Recuperado el 6 de septiembre de 2022, de https://naylampmechatronics.com/sensores-temperatura-y-humedad/47-sensor-de-humedad-de-suelo-fc-28.html

NAYLAMP MECHATRONICS. (30 de abril de 2018). (©2021 Naylamp Mechatronics SAC) Recuperado el 3 de agosto de 2022, de https://naylampmechatronics.com/sensores-proximidad/10-sensor-ultrasonido-hc-sr04.html

NAYLAMP MECHATRONICS. (30 de abril de 2018). (©2021 Naylamp Mechatronics SAC) Recuperado el 3 de agosto de 2022, de https://naylampmechatronics.com/sensores-proximidad/224-sensor-ultrasonido-analogico-us-016.html

NAYLAMP MECHATRONICS. (30 de abril de 2018). (©2021 Naylamp Mechatronics SAC) Recuperado el 3 de agosto de 2022, de https://naylampmechatronics.com/sensores-proximidad/283-sensor-ultrasonico-us-100.html

Naylamp Mechatronics. (30 de abril de 2018). *NAYLAMP MECHATRONICS*. (©2021 Naylamp Mechatronics SAC) Recuperado el 3 de agosto de 2022, de https://naylampmechatronics.com/sensores-proximidad/326-sensor-ultrasonido-jsn-sr04t.html

NAYLAMP MECHATRONICS. (20 de abril de 2020). (©2021 Naylamp Mechatronics SAC) Recuperado el 6 de septiembre de 2022, de https://naylampmechatronics.com/sensores-temperatura-y-humedad/538-sensor-de-humedad-del-suelo-capacitivo.html

Nieto, E. (28 de diciembre de 2018). *fidestec aprende 1 vez & repara 1000*. (Fidestec Copyright © 2023) Recuperado el 8 de marzo de 2023, de https://fidestec.com/blog/podcast/electronicologia-8/

NTE Electronics, I. (4 de abril de 2020). *NTE ELECTRONICS, INC.* Recuperado el 1 de agosto de 2022, de https://www.nteinc.com/resistor_web/pdf/LDR-Series.pdf

Orient Display (USA) Corp. (18 de diciembre de 2020). *ORIENT DISPLAY Your Total LCD Solution Provider*. Recuperado el 21 de septiembre de 2022, de https://www.orientdisplay.com/es/knowledge-base/lcd-basics/what-is-lcd-liquid-crystal-display/

Pérez, N. A. (21 de marzo de 2021). *Portal Académico del CCH, UNAM.* (Escuela Nacional Colegio de Ciencias y Humanidades) Recuperado el 8 de marzo de 2023, de https://portalacademico.cch.unam.mx/cibernetica1/implementacion-de-circuitos-logicos/fuentes-de-poder

Porto, J. P., & Gardey, A. (2014). *Definición.DE.* Recuperado el 27 de septiembre de 2022, de https://definicion.de/solenoide/

Porto, J. P., & Gardey, A. (2020). *Definición.DE.* Recuperado el 27 de septiembre de 2022, de https://definicion.de/electroiman/

Prometec www.prometec.net. (19 de junio de 2020). (Prometec) Recuperado el 23 de mayo de 2022, de https://www.prometec.net/introduccion-a-iot-con-esp32/

punto flotante s.a. (2 de febrero de 2017). (Punto Flotante S.A.) Recuperado el 3 de agosto de 2022, de https://puntoflotante.net/MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501.pdf

R., J. L. (s.f.). *ComoFunciona.* Recuperado el 26 de septiembre de 2022, de ComoFunciona: https://como-funciona.co/un-rele-o-relevador/

RAMDOM NERD TUTORIALS. (2 de marzo de 2021). (RandomNerdTutorials.com © 2013-2022) Recuperado el 27 de septiembre de 2022, de https://randomnerdtutorials.com/esp32-save-data-permanently-preferences/

RANDOM NERD TUTORIALS. (30 de octubre de 2018). (RandomNerdTutorials.com © 2013-2022) Recuperado el 26 de septiembre de 2022, de https://randomnerdtutorials.com/esp32-pwm-arduino-ide/

RANDOM NERD TUTORIALS. (1 de febrero de 2019). (RandomNerdTutorials.com © 2013-2022) Recuperado el 2 de septiembre de 2022, de https://randomnerdtutorials.com/solved-failed-to-connect-to-esp32-timed-out-waiting-for-packet-header/

RANDOM NERD TUTORIALS. (10 de junio de 2021). (RandomNerdTutorials.com © 2013-2022) Recuperado el 23 de septiembre de 2022, de https://randomnerdtutorials.com/esp32-bluetooth-classic-arduino-ide/

REVISTA ESPAÑOLA DE ELECTRÓNICA S.L. (21 de enero de 2022). *REVISTA ESPAÑOLA DE electrónica.* Recuperado el 26 de septiembre de 2022, de https://www.redeweb.com/actualidad/que-es-un-relevador/

Ricardo, R. (17 de marzo de 2022). *Estudi/yando Siempre se puede ser mejor*. Recuperado el 8 de diciembre de 2022, de https://estudyando.com/que-es-un-diodo-flyback-proposito-y-calculos/

Salazar, J., & Silvestre, S. (2006). *UPC Universitat Politècnica de Catalunya.* Recuperado el 15 de mayo de 2022, de https://upcommons.upc.edu/bitstream/handle/2117/100921/LM08_R_ES.pdf

Shoptronica S.L. (19 de agosto de 2015). *Shoptronica.* Recuperado el 23 de septiembre de 2022, de https://www.shoptronica.com/curiosidades-tutoriales-y-gadgets/3981-que-son-los-interruptor-magnetico-reed-switch-0689593949974.html

simon. (14 de octubre de 2021). (SIMON, S.A.) Recuperado el 1 de agosto de 2022, de https://www.simonelectric.com/blog/que-es-un-sensor-de-luminosidad-y-para-que-se-utiliza

SIMON, S.A. (4 de abril de 2018). *simon.* Recuperado el 23 de septiembre de 2022, de https://bricoladores.simonelectric.com/bid/361789/qu-es-un-interruptor

Spark Fun Electronics, Inc. (3 de enero de 2018). *sparkfun START SOMETHING*. Recuperado el 12 de septiembre de 2022, de https://www.sparkfun.com/products/15901

Techlandia. (13 de septiembre de 2017). Obtenido de https://techlandia.com/reparar-cargador-taladro-como_101973/

The MathWorks, Inc. (10 de enero de 2018). *ThingSpeak™*. Recuperado el 30 de septiembre de 2022, de https://thingspeak.com/prices/thingspeak_standard

THINK BIG LABS S.L. (22 de octubre de 2019). *thinger.io.* Recuperado el 30 de septiembre de 2022, de https://pricing.thinger.io/#!/cloud

Toshiba Semiconductor. (15 de febrero de 2006). *ALLDATASHEET.COM*. Recuperado el 20 de septiembre de 2022, de https://pdf1.alldatasheet.com/datasheet-pdf/view/182615/TOSHIBA/ULN2803A.html

Ubidots (IoT Services Inc.). (1 de febrero de 2022). *ubidots.* Recuperado el 30 de septiembre de 2022, de https://ubidots.com/pricing

UNIT ELECTRONIC SERVICES MX S.A. DE C.V. (4 de diciembre de 2020). *UNIT ELECTRONICS*. Recuperado el 5 de diciembre de 2022, de https://uelectronics.com/producto/transistor-irf840/

UNIT ELECTRONIC SERVICES MX S.A. DE C.V. (18 de abril de 2023). *UNIT ELECTRONICS*. Obtenido de uelectronics.com: https://uelectronics.com/producto/bomba-de-agua-sumergible-70-120l-h/

UNIT Electronics © 2016-2022. (25 de abril de 2019). *UNIT ELECTRONICS.* Recuperado el 23 de septiembre de 2022, de https://uelectronics.com/producto/push-button-4-pines-microswitch/

UNIT Electronics © 2016-2022. (8 de diciembre de 2020). *UNIT ELECTRONICS.* Recuperado el 26 de septiembre de 2022, de https://uelectronics.com/categoria-producto/componentes/rele/

UNIT Electronics © 2016-2022. (4 de diciembre de 2020). *UNIT ELECTRONICS.* Recuperado el 27 de septiembre de 2022, de https://uelectronics.com/producto/solenoide-push-pull-zye1-0530z-carrera-10mm/

UNIT Electronics © 2016-2022. (3 de diciembre de 2020). *UNIT ELECTRONICS.* Recuperado el 27 de septiembre de 2022, de https://uelectronics.com/producto/cerradura-electrica-solenoide/

UNIT Electronics © 2016-2022. (21 de junio de 2022). *UNIT ELECTRONICS.* Recuperado el 26 de septiembre de 2022, de https://uelectronics.com/producto/ssr-40-relevador-estado-solido-40a/

UNIT Electronics © 2016-2022. (27 de julio de 2022). *UNIT ELECTRONICS.* Recuperado el 27 de septiembre de 2022, de https://uelectronics.com/producto/cerradura-electrica-xg-07-12v-1-4a-2a/

UNIT Electronics © 2016-2022. (27 de julio de 2022). *UNIT ELECTRONICS.* Recuperado el 27 de septiembre de 2022, de https://uelectronics.com/producto/cerradura-electrica-xg-07-5v-1-2a-1-5a/

UNIT Electronics. (17 de septiembre de 2020). (UNIT Electronics ©) Recuperado el 6 de agosto de 2022, de https://uelectronics.com/producto/esp32-38-pines-esp-wroom-32/

UNIT ELECTRONICS. (4 de diciembre de 2020). (UNIT ELECTRONIC SERVICES MX S.A. DE C.V.) Recuperado el 3 de agosto de 2022, de https://uelectronics.com/producto/fc-51-sensor-de-obstaculos-reflectivo-infrarojo/

UNIT ELECTRONICS. (13 de enero de 2021). (UNIT ELECTRONIC SERVICES MX S.A. DE C.V.) Recuperado el 3 de agosto de 2022, de https://uelectronics.com/producto/sensores-de-movimiento-pir-hc-sr501-hc-sr505-hy3612-am312

uPesy. (10 de febrero de 2022). (uPesy Electronics) Recuperado el 6 de agosto de 2022, de https://www.upesy.com/blogs/tutorials/esp32-pinout-reference-gpio-pins-ultimate-guide

Villar, S. (30 de enero de 2020). *EcoLuz LED ILUMINACIÓN Y DECORACIÓN LED BY TLUCE*. Recuperado el 20 de septiembre de 2022, de https://www.ecoluzled.com/blog/tipos-de-diodos-led-como-funcionan/

VISHAY. (8 de julio de 2004). *www.vishay.com.* Recuperado el 2 de agosto de 2022, de https://drive.google.com/file/d/0BzaKjvCRihgbSDdiaUYtakpxdzA/view?resourcekey=0-Ng2dRhHOrktIBvQeRUBpwA

VISTRONICA S.A.S. (30 de septiembre de 2018). *Vistrónica Tienda Virtual De Electrónica*. (VISTRONICA S.A.S) Recuperado el 3 de agosto de 2022, de https://www.vistronica.com/aeromodelismo/sensor-de-ultrasonido-gy-us42-para-controlador-apm--detail.html

FUENTES DE IMÁGENES

Figura 2.1. Modulo PCB ESP32-WROOM-32 distribución de GPIO, electrónica sin blindaje. Recuperado de: https://blog.bricogeek.com/noticias/electronica/comparativa-y-analisis-completo-de-los-modulos-wifi-esp8266-y-esp32/, el 22 de septiembre de 2022

Figura 2.2. Tarjeta de desarrollo ESP32 DEVKIT genérica. Recuperado de: https://es.aliexpress.com/i/1005002739635392.html, el 22 de septiembre de 2022

Figura 2.3. Tarjeta de desarrollo Wemos Lolin32 ESP32 OLED. Recuperado de: https://programarfacil.com/esp8266/esp32/, el 23 de septiembre de 2022

Figura 2.4. Tarjeta de desarrollo ESP32-CAM genérica. Recuperado de: https://www.amazon.com/-/es/tarjeta-desarrollo-Bluetooth-compatible-interfaz/dp/B07TZQ9NNM, el 23 de septiembre de 2022

Figura 2.5. Distribución de pines en la tarjeta de desarrollo NodeMCU-32S. Recuperado de: https://www.upesy.com/blogs/tutorials/esp32-pinout-reference-gpio-pins-ultimate-guide, el 23 de septiembre de 2022

Figura 2.6. Confirmación de controladores para el sistema operativo Windows.

Figura 2.7. Ventana de preferencias en el IDE Arduino.

Figura 2.8. Instalación de complemento a través del gestor de tarjetas.

Figura 2.9. Selección de tarjeta ESP32 en Arduino IDE.

Figura 2.10. Gestor de bibliotecas, IDE Arduino

Figura 2.11. Instalación de biblioteca para plataforma Blynk.

Figura 2.12. Instalación manual de biblioteca en IDE Arduino.

Figura 2.13. Fotorresistor (LDR). Recuperado de: https://articulo.mercadolibre.com.mx/MLM-763876878-fotoresistencias-sensor-de-luz-sensor-fotoresistivo-arduino-_JM, el 7 de septiembre de 2022

Figura 2.14. Módulo BH1750. Recuperado de: https://naylampmechatronics.com/blog/44_tutorial-modulo-sensor-de-luz-bh1750.html, el 7 de septiembre de 2022

Figura 2.15. Módulo TEMP6000. Recuperado de: https://www.amazon.com/-/es/unidades-simulaci%C3%B3n-intensidad-visible-temt6000/dp/B0711QBLSF, el 7 de septiembre de 2022

Figura 2.16. Módulo TSL2561. Recuperado de: https://www.mouser.mx/ProductDetail/Adafruit/2651?qs=GP1oqPAcsMG0HimHwVa7aA%3D%3D, el 7 de septiembre de 2022

Figura 2.17. Módulo APDS-9960. Recuperado de: https://www.digikey.com.mx/es/articles/quickly-design-your-own-low-cost-3d-gesture-controller, el 7 de septiembre de 2022

Figura 2.18. Módulo FC-51. Recuperado de: https://uelectronics.com/producto/fc-51-sensor-de-obstaculos-reflectivo-infrarojo/, el 7 de septiembre de 2022

Figura 2.19. Módulos con sensor PIR. Recuperado de: https://uelectronics.com/producto/sensores-de-movimiento-pir-hc-sr501-hc-sr505-hy3612-am312/, el 8 de septiembre de 2022

Figura 2.20. Sensor JSN-SR04T. Recuperado de: https://hetpro-store.com/jsn-sr04t/, el 8 de septiembre de 2022

Figura 2.21. Módulo GY-US42. Recuperado de: https://www.amazon.com.mx/UIOTEC-GY-US42-medici%C3%B3n-Distancia-ultrasonidos/dp/B07F2C7D6R, el 8 de septiembre de 2022

Figura 2.22. Módulo HC-SR04. Recuperado de: https://ar.mouser.com/new/sparkfun/sparkfun-hcsr04-distance-sensor/, el 8 de septiembre de 2022

Figura 2.23. Arreglo Darlington NPN y PNP. Recuperado de: https://www.hwlibre.com/uln2803/, el 1 de agosto de 2022

Figura 2.24. Diagrama lógico del circuito integrado ULN2803. Recuperado de: https://rogerbit.com/wprb/2020/01/como-hacer-un-medidor-de-nivel-de-agua-con-uln2803-primera-parte/, el 1 de agosto de 2022

Figura 2.25. Pinout de circuito integrado ULN2803. Recuperado de: https://rogerbit.com/wprb/2020/10/remote-water-level-meter-with-drf1276dm-drf1278dm-and-arduino-lora-modules/, el 1 de agosto de 2022

Figura 2.26. Electrodo construido con tubo PVC y tornillos.

Figura 2.27. Electrodos de bronce con cubierta plástica para reducir la corrosión galvánica. Recuperado de: https://purificadoragua.tododeagua.mx/filtro/tableros-de-control-juego-de-3-electrodos-con-cubiertas-de-plastico-e-3-nassar-electronics, el 9 de septiembre de 2022

Figura 2.28. Alambrado de conexión electrodos y circuito integrado ULN2803.

Figura 2.29. Sensor y módulo HD-38. Recuperado de: https://www.mactronica.com.co/sensor-de-humedad-del-suelo-hd-38, el 9 de septiembre de 2022

Figura 2.30. Sensor FC-28 con módulo. Recuperado de: https://es.aliexpress.com/item/32435196705.html, el 9 de septiembre de 2022

Figura 2.31. Sensor de humedad capacitivo. Recuperado de: https://yorobotics.co/producto/sensor-humedad-suelo-capacitivo-analogico-v1-2-3-3v-5v, el 9 de septiembre de 2022

Figura 2.32. Módulo RFID/NFC PN532. Recuperado de: https://naylampmechatronics.com/rfid-nfc/182-modulo-lector-rfid-nfc-1356mhz-pn532.html, el 12 de septiembre de 2022

Figura 2.33. Módulo RFID RC522. Recuperado de: https://naylampmechatronics.com/rfid-nfc/80-modulo-lector-rfid-1356mhz-rc522.html, el 12 de septiembre de 2022

Figura 2.34. Módulo con sensor DHT 11. Recuperado de: https://www.330ohms.com/products/modulo-sensor-de-humedad-y-temperatura-dht11-con-cables-jumper, el 12 de septiembre de 2022

Figura 2.35. Módulo con sensor DHT 21. Recuperado de: https://naylampmechatronics.com/sensores-temperatura-y-humedad/354-sensor-de-temperatura-y-humedad-relativa-dht21-am2301.html, el 12 de septiembre de 2022

Figura 2.36. Módulo con sensor DHT 22. Recuperado de: https://electronilab.co/tienda/modulo-sensor-dht22-am2302-temperatura-y-humedad/, el 12 de septiembre de 2022

Figura 2.37. Módulos BME/BMP280. Recuperado de: https://es.aliexpress.com/item/1005002687464954.html, el 27 de septiembre de 2022

Figura 2.38. Módulo BMP180. Recuperado de: https://es.aliexpress.com/item/1005003234120050.html, el 27 de septiembre de 2022

Figura 2.39. Módulo BMP085. Recuperado de: https://learn.adafruit.com/bmp085/wiring-the-bmp085, el 27 de septiembre de 2022

Figura 2.40. Módulo SI1145. Recuperado de: https://co-en.rs-online.com/product/adafruit-industries/1777/70460639/, el 27 de septiembre de 2022

Figura 2.41. Módulo VEML6075. Recuperado de: https://www.electronicaplugandplay.com/sensores-y-transductores/luz-y-color/product/987-uva-uvb-sensor-module-veml6075, el 27 de septiembre de 2022

Figura 2.42. Módulo GUVA-S12SD. Recuperado de: https://articulo.mercadolibre.com.mx/MLM-1443350732-adafruit-sensor-de-luz-uv-ultravioleta-analogico-guva-s12sd-_JM#position=8&search_layout=stack&type=item&tracking_id=791df73d-efa5-431f-988e-e61dfaf9e6e2, el 27 de septiembre de 2022

Figura 2.43. Módulo GY-ML8511. Recuperado de: https://articulo.mercadolibre.com.mx/MLM-754244229-modulo-sensor-de-luz-uv-ultravioleta-gy-8511-ml8511-arduino-_JM, el 27 de septiembre de 2022

Figura 2.44. Anemómetro Adafruit. Recuperado de: https://aelectronics.com.mx/metepec/sensores/1433-sensor-de-velocidad-de-viento.html, el 12 de septiembre de 2022

Figura 2.45. Kit Sparkfun. Recuperado de: https://www.330ohms.com/products/weather-meters, el 12 de septiembre de 2022

Figura 2.46. Anemómetro casero. Recuperado de: https://espaciodecesar.com/2017/03/25/como-medir-el-viento-muy-facil-anemometro-simple-y-con-arduino/, el 12 de septiembre de 2022

Figura 2.47. Esquema de LED THT y simbología. Recuperado de: https://www.panelesled.es/service/sobre-iluminacion-led/, el 30 de septiembre de 2022

Figura 2.48. Pantalla LCD 2 líneas y 16 caracteres por línea. Recuperado de: https://www.amazon.com/-/es/Caterpillar-CAT-320C-Excavadora-pantalla/dp/B08GG8F96W, el 30 de septiembre de 2022

Figura 2.49. Pantalla LCD TFT, 128x160 pixeles, color RGB. Recuperado de: https://www.circuitarte.com/producto/pantalla-lcd-tft-ips-1-8-65k-full-color-spi/, el 30 de septiembre de 2022

Figura 2.50. Pantalla LCD IPS 1024x600 pixeles, color RGB. Recuperado de: https://www.amazon.com/-/es/Labs-Pantalla-t%C3%A1ctil-pulgadas-Raspberry/dp/B0796SMRMG, el 30 de septiembre de 2022

Figura 2.51. Push Button. Recuperado de: https://articulo.mercadolibre.com.ar/MLA-1101695830-pulsador-boton-dip-tactil-switch-12x12x75mm-circular-hobb-_JM#position=5&search_layout=stack&type=item&tracking_id=aa82ada0-b332-4ffc-8ef3-0a53fb48504e, el 30 de septiembre de 2022

Figura 2.52. Ejemplos de Switch basculante. Recuperado de: https://www.amazon.com/-/es/interruptores-basculantes-posiciones-encendido-interruptor/dp/B0BDDRHVTV, https://www.amazon.com.mx/GE-18233-interruptor-paleta-blanco/dp/B002HEY3AU, https://articulo.mercadolibre.com.mx/MLM-1352102320-switch-interruptor-universal-palanca-cola-de-raton-auto-_JM, https://www.digikey.com.mx/es/products/detail/honeywell-sensing-and-productivity-solutions/DT-2RV23-A7/2748710, el 30 de septiembre de 2022

Figura 2.53. Reed switch, de izquierda a derecha empaque comercial, componente electrónico. Recuperado de: https://shopdelta.eu/reed-switch-interruptor-de-lengueta-lateral-kn-02_l6_p8427.html, https://es.aliexpress.com/item/4001027767596.html, el 30 de septiembre de 2022

Figura 2.54. Esquema interno de un relé electromecánico. Recuperado de: https://ditecnomakers.com/que-es-un-rele/, el 1 de agosto de 2022

Figura 2.55. Relevadores de marca Songle. Recuperado de: https://uelectronics.com/producto/relevadores-songle-30a-5v-12v-24v/, el 1 de agosto de 2022

Figura 2.56. Relevadores de estado sólido serie SSR-40, marca FQFER. Recuperado de: https://uelectronics.com/producto/ssr-40-relevador-estado-solido-40a/, el 1 de agosto de 2022

Figura 2.57. Bombas de agua comercial para uso doméstico (a y b), bomba para Arduino (c). Recuperado de: https://www.amazon.com.mx/Igoto-Pump-QB70-Bomba-Perif%C3%A9rica/dp/B07NLFKDP2, https://www.ferrepat.com/articulos-de-mantenimento/bomba-de-motor-electrico-1758, https://www.amazon.com/-/es/Bomba-agua-mini-sumergible-escobillas/dp/B07CRMMPYB, el 5 de agosto de 2022

Figura 2.58. Electrocerradura a 12 V. Recuperado de: https://www.instalmatic.com/tienda/accesorios-puerta-automatica/cerraduras-electricas-puerta-automatica/electrocerradura-con-pulsador-vds/, el 23 de enero de 2023

https://es.aliexpress.com/item/4000689836094.html, https://www.amazon.com.mx/descarga-recargable-superior-bater%C3%ADas-linterna/dp/B0BW11WDXX, el 30 de enero de 2023

[1] Todas las figuras sin fuente son de autoría propia.

ANEXO

Anexo A. Programa control de encendido de una luminaria[1]

```
1       //aquí se pega la información de las credenciales de la plantilla
creada en Blynk
2       #define BLYNK_TEMPLATE_ID "TMPLxxxxxx"
3       #define BLYNK_DEVICE_NAME "CONTROL DE LED MULTIWIFI"
4
5       #define BLYNK_FIRMWARE_VERSION "0.1.0"
6
7       #define BLYNK_PRINT Serial
8       //#define BLYNK_DEBUG
9
10      #define APP_DEBUG
11
12      // Uncomment your board, or configure a custom board in Settings.h
13      //#define USE_WROVER_BOARD
14      //#define USE_TTGO_T7
15      //#define USE_ESP32C3_DEV_MODULE
16      //#define USE_ESP32S2_DEV_KIT
17
18      #include "BlynkEdgent.h"
19      #define LEDPIN 13
20
21      BLYNK_WRITE(V0) {
22        digitalWrite(LEDPIN, param.asInt());
23      }
24
25      void setup() {
26        Serial.begin(115200);
27        pinMode(LEDPIN, OUTPUT);
28        delay(100);
29
30        BlynkEdgent.begin();
31      }
32
33      void loop() {
34        BlynkEdgent.run();
35      }
```

[1] El desarrollo de los programas contiene líneas de código provenientes del trabajo colectivo en los foros oficiales de consulta en los sitios web: https://community.blynk.cc/; https://forum.arduino.cc/; https://github.com/

Anexo B. Programas control de luminarias, adicional notificación de presencia y asistentes de voz[1]

B.1 Programa plataforma Blynk

```
1       //aquí se pega la información de las credenciales de la plantilla
creada en Blynk
2       #define BLYNK_TEMPLATE_ID "TMPLxxxxxx"
3       #define BLYNK_DEVICE_NAME "ILUMINACION"
4
5       #define BLYNK_FIRMWARE_VERSION        "0.1.0"
6
7       #define BLYNK_PRINT Serial
8       //#define BLYNK_DEBUG
9
10      #define PIR_Pin 19
11      #define RelayPin1 23
12      #define RelayPin2 27
13      #define RelayPin3 5
14      #define RelayPin4 18
15      #define SwitchPin1 13
16      #define SwitchPin2 12
17      #define SwitchPin3 26
18      #define SwitchPin4 14
19      #define VPIN_BUTTON_1    V0
20      #define VPIN_BUTTON_2    V1
21      #define VPIN_BUTTON_3    V2
22      #define VPIN_BUTTON_4    V3
23      #define VPIN_BUTTON_C    V5
24
25      bool State_PIR = LOW;
26      bool toggleState_1 = LOW;
27      bool toggleState_2 = LOW;
28      bool toggleState_3 = LOW;
29      bool toggleState_4 = LOW;
30      bool SwitchState_1 = LOW;
31      bool SwitchState_2 = LOW;
32      bool SwitchState_3 = LOW;
33      bool SwitchState_4 = LOW;
34
35      #define APP_DEBUG
36
37      #include "BlynkEdgent.h"
38      #include <BH1750.h>
39      #include <Wire.h>
40      BlynkTimer timer;
41      BH1750 lightMeter;
42
43      BLYNK_CONNECTED() {
44        Blynk.syncVirtual(VPIN_BUTTON_1);
```

[1] El desarrollo de los programas contiene líneas de código provenientes del trabajo colectivo en los foros oficiales de consulta en los sitios web: https://community.blynk.cc/; https://forum.arduino.cc/; https://github.com/

```
45        Blynk.syncVirtual(VPIN_BUTTON_2);
46        Blynk.syncVirtual(VPIN_BUTTON_3);
47        Blynk.syncVirtual(VPIN_BUTTON_4);
48      }
49
50      BLYNK_WRITE(VPIN_BUTTON_1) {
51        toggleState_1 = param.asInt();
52        if(toggleState_1 == 1){
53          digitalWrite(RelayPin1, LOW);
54        }
55        else {
56          digitalWrite(RelayPin1, HIGH);
57        }
58      }
59
60      BLYNK_WRITE(VPIN_BUTTON_2) {
61        toggleState_2 = param.asInt();
62        if(toggleState_2 == 1){
63          digitalWrite(RelayPin2, LOW);
64        }
65        else {
66          digitalWrite(RelayPin2, HIGH);
67        }
68      }
69
70      BLYNK_WRITE(VPIN_BUTTON_3) {
71        toggleState_3 = param.asInt();
72        if(toggleState_3 == 1){
73          digitalWrite(RelayPin3, LOW);
74        }
75        else {
76          digitalWrite(RelayPin3, HIGH);
77        }
78      }
79
80      BLYNK_WRITE(V4) {
81        State_PIR = param.asInt();
82      }
83
84      BLYNK_WRITE(VPIN_BUTTON_C) {
85        all_SwitchOff();
86      }
87
88      void all_SwitchOff() {
89        toggleState_1 = 0;
90        digitalWrite(RelayPin1, HIGH);
91        Blynk.virtualWrite(VPIN_BUTTON_1, toggleState_1);
92        delay(100);
93        toggleState_2 = 0;
94        digitalWrite(RelayPin2, HIGH);
95        Blynk.virtualWrite(VPIN_BUTTON_2, toggleState_2);
96        delay(100);
97        toggleState_3 = 0;
98        digitalWrite(RelayPin3, HIGH);
```

```
99        Blynk.virtualWrite(VPIN_BUTTON_3, toggleState_3);
100       delay(100);
101     }
102
103     void setup() {
104       Serial.begin(115200);
105       pinMode(RelayPin1, OUTPUT);
106       pinMode(RelayPin2, OUTPUT);
107       pinMode(RelayPin3, OUTPUT);
108       pinMode(RelayPin4, OUTPUT);
109       pinMode(SwitchPin1, INPUT_PULLUP);
110       pinMode(SwitchPin2, INPUT_PULLUP);
111       pinMode(SwitchPin3, INPUT_PULLUP);
112       pinMode(SwitchPin4, INPUT_PULLUP);
113       digitalWrite(RelayPin1, HIGH);
114       digitalWrite(RelayPin2, HIGH);
115       digitalWrite(RelayPin3, HIGH);
116       digitalWrite(RelayPin4, HIGH);
117       pinMode(PIR_Pin, INPUT);
118
119       Wire.begin();
120       lightMeter.begin();
121       delay(100);
122
123       timer.setInterval(1000L, SensPIR);
124       timer.setInterval(2000L, AutoLight);
125
126       BlynkEdgent.begin();
127     }
128
129     void SensPIR() {
130       bool state = digitalRead(PIR_Pin);
131       if(State_PIR == 1)  {
132         if(state == 1) {
133           Blynk.logEvent("movimiento");
134           Serial.println("Hay Movimiento");
135         }
136         else {
137           Serial.println("Sin movimiento");
138         }
139       }
140     }
141
142     void AutoLight() {
143       float lux = lightMeter.readLightLevel();
144       if (lux <= 1.00){
145         digitalWrite(RelayPin4, LOW);
146         Blynk.virtualWrite(VPIN_BUTTON_4, HIGH);
147       }
148       else {
149         digitalWrite(RelayPin4, HIGH);
150         Blynk.virtualWrite(VPIN_BUTTON_4, LOW);
151       }
152     }
```

```
153
154     void Manual_control() {
155       if (digitalRead(SwitchPin1) == LOW && SwitchState_1 == LOW) {
156         digitalWrite(RelayPin1, LOW);
157         Blynk.virtualWrite(VPIN_BUTTON_1, HIGH);
158         toggleState_1 = HIGH;
159         SwitchState_1 = HIGH;
160       }
161       if (digitalRead(SwitchPin1) == HIGH && SwitchState_1 == HIGH) {
162         digitalWrite(RelayPin1, HIGH);
163         Blynk.virtualWrite(VPIN_BUTTON_1, LOW);
164         toggleState_1 = LOW;
165         SwitchState_1 = LOW;
166       }
167       if (digitalRead(SwitchPin2) == LOW && SwitchState_2 == LOW) {
168         digitalWrite(RelayPin2, LOW);
169         Blynk.virtualWrite(VPIN_BUTTON_2, HIGH);
170         toggleState_2 = HIGH;
171         SwitchState_2 = HIGH;
172       }
173       if (digitalRead(SwitchPin2) == HIGH && SwitchState_2 == HIGH) {
174         digitalWrite(RelayPin2, HIGH);
175         Blynk.virtualWrite(VPIN_BUTTON_2, LOW);
176         toggleState_2 = LOW;
177         SwitchState_2 = LOW;
178       }
179       if (digitalRead(SwitchPin3) == LOW && SwitchState_3 == LOW) {
180         digitalWrite(RelayPin3, LOW);
181         Blynk.virtualWrite(VPIN_BUTTON_3, HIGH);
182         toggleState_3 = HIGH;
183         SwitchState_3 = HIGH;
184       }
185       if (digitalRead(SwitchPin3) == HIGH && SwitchState_3 == HIGH) {
186         digitalWrite(RelayPin3, HIGH);
187         Blynk.virtualWrite(VPIN_BUTTON_3, LOW);
188         toggleState_3 = LOW;
189         SwitchState_3 = LOW;
190       }
191       if (digitalRead(SwitchPin4) == LOW && SwitchState_4 == LOW) {
192         digitalWrite(RelayPin4, LOW);
193         Blynk.virtualWrite(VPIN_BUTTON_4, HIGH);
194         toggleState_4 = HIGH;
195         SwitchState_4 = HIGH;
196       }
197       if (digitalRead(SwitchPin4) == HIGH && SwitchState_4 == HIGH) {
198         digitalWrite(RelayPin4, HIGH);
199         Blynk.virtualWrite(VPIN_BUTTON_4, LOW);
200         toggleState_4 = LOW;
201         SwitchState_4 = LOW;
202       }
203     }
204
205     void loop() {
206       BlynkEdgent.run();
```

```
207       Manual_control();
208       AutoLight();
209       timer.run();
210     }
```

B.1.1 Configuración adicional para trabajar fuera de línea

Cabecera BlynkEdgent.h

```
1       extern "C" {
2         void app_loop();
3         void eraseMcuConfig();
4         void restartMCU();
5       }
6
7       #include "Settings.h"
8       #include <BlynkSimpleEsp32_SSL.h>
9
10      #ifndef BLYNK_NEW_LIBRARY
11      #error "Old version of Blynk library is in use. Please replace it
with the new one."
12      #endif
13
14      #if !defined(BLYNK_TEMPLATE_ID) || !defined(BLYNK_DEVICE_NAME)
15      #error "Please specify your BLYNK_TEMPLATE_ID and
BLYNK_DEVICE_NAME"
16      #endif
17
18      #include "BlynkState.h"
19      #include "ConfigStore.h"
20      #include "ResetButton.h"
21      #include "ConfigMode.h"
22      #include "Indicator.h"
23      #include "OTA.h"
24      #include "Console.h"
25
26      void Manual_control();
27      void AutoLight();
28      inline
29      void BlynkState::set(State m) {
30        if (state != m && m < MODE_MAX_VALUE) {
31          DEBUG_PRINT(String(StateStr[state]) + " => " + StateStr[m]);
32          state = m;
33
34          // You can put your state handling here,
35          // i.e. implement custom indication
36        }
37      }
38
39      void printDeviceBanner()
40      {
41        Blynk.printBanner();
42        DEBUG_PRINT("-------------------------");
43        DEBUG_PRINT(String("Product:  ") + BLYNK_DEVICE_NAME);
```

```
44          DEBUG_PRINT(String("Firmware: ") + BLYNK_FIRMWARE_VERSION "
(build " __DATE__ " " __TIME__ ")");
45          if (configStore.getFlag(CONFIG_FLAG_VALID)) {
46            DEBUG_PRINT(String("Token:    ...") +
(configStore.cloudToken+28));
47          }
48          DEBUG_PRINT(String("Device:   ") + BLYNK_INFO_DEVICE + " @ " +
ESP.getCpuFreqMHz() + "MHz");
49          DEBUG_PRINT(String("MAC:      ") + Wifi.macAddress());
50          DEBUG_PRINT(String("Flash:    ") + ESP.getFlashChipSize() / 1024
+ "K");
51          DEBUG_PRINT(String("ESP sdk:  ") + ESP.getSdkVersion());
52          DEBUG_PRINT(String("Chip rev: ") + ESP.getChipRevision());
53          DEBUG_PRINT(String("Free mem: ") + ESP.getFreeHeap());
54          DEBUG_PRINT("-------------------------");
55        }
56
57        void runBlynkWithChecks() {
58          Blynk.run();
59          if (BlynkState::get() == MODE_RUNNING) {
60            if (!Blynk.connected()) {
61              if (Wifi.status() == WL_CONNECTED) {
62                BlynkState::set(MODE_CONNECTING_CLOUD);
63              } else {
64                BlynkState::set(MODE_CONNECTING_NET);
65              }
66            }
67          }
68        }
69
70        class Edgent {
71
72        public:
73          void begin()
74          {
75            Wifi.persistent(false);
76            Wifi.enableSTA(true); // Needed to get MAC
77
78            indicator_init();
79            button_init();
80            config_init();
81            console_init();
82
83            printDeviceBanner();
84
85            if (configStore.getFlag(CONFIG_FLAG_VALID)) {
86              BlynkState::set(MODE_CONNECTING_NET);
87            } else if (config_load_blnkopt()) {
88              DEBUG_PRINT("Firmware is preprovisioned");
89              BlynkState::set(MODE_CONNECTING_NET);
90            } else {
91              BlynkState::set(MODE_WAIT_CONFIG);
92            }
93          }
```

```
94
95          void run() {
96            app_loop();
97            switch (BlynkState::get()) {
98            case MODE_WAIT_CONFIG:
99            case MODE_CONFIGURING:       enterConfigMode();    break;
100           case MODE_CONNECTING_NET:    enterConnectNet();    break;
101           case MODE_CONNECTING_CLOUD:  enterConnectCloud();  break;
102           case MODE_RUNNING:           runBlynkWithChecks(); break;
103           case MODE_OTA_UPGRADE:       enterOTA();           break;
104           case MODE_SWITCH_TO_STA:     enterSwitchToSTA();   break;
105           case MODE_RESET_CONFIG:      enterResetConfig();   break;
106           default:                     enterError();         break;
107           }
108         }
109
110     };
111
112     Edgent BlynkEdgent;
113     BlynkTimer edgentTimer;
114
115     void app_loop() {
116         edgentTimer.run();
117         edgentConsole.run();
118         Manual_control();
119         AutoLight();
120     }
```

B.2 Programa plataforma Sinric Pro

```
1       //quitar comentario para depurar por medio de comunicación serie
2       //#define ENABLE_DEBUG
3
4       #ifdef ENABLE_DEBUG
5              #define DEBUG_ESP_PORT Serial
6              #define NODEBUG_WEBSOCKETS
7              #define NDEBUG
8       #endif
9
10      #include <Arduino.h>
11      #include <Wifi.h>
12      #include "SinricPro.h"
13      #include "SinricProSwitch.h"
14
15      #include <map>
16
17      #define WIFI_SSID         "IdWifi-xxxx"
18      #define WIFI_PASS         "contraseñaRedWifi"
19      #define APP_KEY           "fe0bxxxx-1x3x-4x3x-ax2x-5dabxxxxxxxx"
20      #define APP_SECRET        "c9a6xxxx-x3x7-4x3x-xexe-e86724a9xxxx-
4c4axxxx-3x3x-x5xe-x9x3-333d65xxxxxx"
21
22      //colocar el Id proporcionado en la plataforma
23      #define device_ID_1   "xxxxxxxxxxxxxxxxxxxxxx18f"
```

```
24      #define device_ID_2   "xxxxxxxxxxxxxxxxxxxx9d1"
25      #define device_ID_3   "xxxxxxxxxxxxxxxxxxxx942"
26      #define device_ID_4   "dispositivo_adicional_con_costo"
27
28      //se definen GPIO de relevadores y switchs
29      #define RelayPin1 23
30      #define RelayPin2 27
31      #define RelayPin3 5
32      #define RelayPin4 18
33
34      #define SwitchPin1 13
35      #define SwitchPin2 12
36      #define SwitchPin3 26
37      #define SwitchPin4 14
38
39      #define wifiLed   2
40
41      //quitar comentario si se utilizan botones en lugar de switchs
42      //#define TACTILE_BUTTON 1
43
44      #define BAUD_RATE   115200
45
46      #define DEBOUNCE_TIME 250
47
48      typedef struct {
49        int relayPIN;
50        int flipSwitchPIN;
51      } deviceConfig_t;
52
53      //configuración principal de la estructura, se pueden agregar más
dispositivos
54      std::map<String, deviceConfig_t> devices = {
55          //{Id_dispositivo, {relayPIN, SwitchPIN}}
56          {device_ID_1, {  RelayPin1, SwitchPin1 }},
57          {device_ID_2, {  RelayPin2, SwitchPin2 }},
58          {device_ID_3, {  RelayPin3, SwitchPin3 }},
59          {device_ID_4, {  RelayPin4, SwitchPin4 }}
60      };
61
62      typedef struct {
63        String deviceId;
64        bool lastFlipSwitchState;
65        unsigned long lastFlipSwitchChange;
66      } flipSwitchConfig_t;
67
68      std::map<int, flipSwitchConfig_t> flipSwitches;
69
70      void setupRelays() {
71        for (auto &device : devices) {
72          int relayPIN = device.second.relayPIN;
73          pinMode(relayPIN, OUTPUT);
74          digitalWrite(relayPIN, HIGH);
75        }
76      }
```

```
77
78      void setupFlipSwitches() {
79        for (auto &device : devices)  {
80          flipSwitchConfig_t flipSwitchConfig;
81
82          flipSwitchConfig.deviceId = device.first;
83          flipSwitchConfig.lastFlipSwitchChange = 0;
84          flipSwitchConfig.lastFlipSwitchState = true;
85
86          int flipSwitchPIN = device.second.flipSwitchPIN;
87
88          flipSwitches[flipSwitchPIN] = flipSwitchConfig;
89          pinMode(flipSwitchPIN, INPUT_PULLUP);
90        }
91      }
92
93      bool onPowerState(String deviceId, bool &state) {
94        Serial.printf("%s: %s\r\n", deviceId.c_str(), state ? "on" :
"off");
95        int relayPIN = devices[deviceId].relayPIN;
96        digitalWrite(relayPIN, !state);
97        return true;
98      }
99
100     void handleFlipSwitches() {
101       unsigned long actualMillis = millis();
102       for (auto &flipSwitch : flipSwitches) {
103         unsigned long lastFlipSwitchChange =
flipSwitch.second.lastFlipSwitchChange;
104
105         if (actualMillis - lastFlipSwitchChange > DEBOUNCE_TIME) {
106
107           int flipSwitchPIN = flipSwitch.first;
108           bool lastFlipSwitchState =
flipSwitch.second.lastFlipSwitchState;
109           bool flipSwitchState = digitalRead(flipSwitchPIN);
110           if (flipSwitchState != lastFlipSwitchState) {
111     #ifdef TACTILE_BUTTON
112             if (flipSwitchState) {
113     #endif
114               flipSwitch.second.lastFlipSwitchChange = actualMillis;
115               String deviceId = flipSwitch.second.deviceId;
116               int relayPIN = devices[deviceId].relayPIN;
117               bool newRelayState = !digitalRead(relayPIN);
118               digitalWrite(relayPIN, newRelayState);
119
120               SinricProSwitch &mySwitch = SinricPro[deviceId];
121               mySwitch.sendPowerStateEvent(!newRelayState);
122     #ifdef TACTILE_BUTTON
123             }
124     #endif
125           flipSwitch.second.lastFlipSwitchState = flipSwitchState;
126         }
127       }
```

```
128       }
129     }
130
131     void setupWifi() {
132       Serial.printf("\r\n[Wifi]: Connecting");
133       Wifi.begin(WIFI_SSID, WIFI_PASS);
134
135       while (Wifi.status() != WL_CONNECTED)
136       {
137         Serial.printf(".");
138         delay(250);
139       }
140       digitalWrite(wifiLed, HIGH);
141       Serial.printf("connected!\r\n[Wifi]: IP-Address is %s\r\n",
Wifi.localIP().toString().c_str());
142     }
143
144     void setupSinricPro() {
145       for (auto &device : devices)
146       {
147         const char *deviceId = device.first.c_str();
148         SinricProSwitch &mySwitch = SinricPro[deviceId];
149         mySwitch.onPowerState(onPowerState);
150       }
151
152       SinricPro.begin(APP_KEY, APP_SECRET);
153       SinricPro.restoreDeviceStates(true);
154     }
155
156     void setup() {
157       Serial.begin(BAUD_RATE);
158
159       pinMode(wifiLed, OUTPUT);
160       digitalWrite(wifiLed, LOW);
161
162       setupRelays();
163       setupFlipSwitches();
164       setupWifi();
165       setupSinricPro();
166     }
167
168     void loop() {
169       SinricPro.handle();
170       handleFlipSwitches();
171     }
```

B.3 Programa plataforma ESP RainMaker

```
1       #include "RMaker.h"
2       #include "Wifi.h"
3       #include "WifiProv.h"
4
5       const char *service_name = "PROV_12345";
```

```
6       const char *pop = "1234567";
7
8       //define el nombre de los dispositivos
9       char deviceName_1[] = "Recibidor";
10      char deviceName_2[] = "Sala";
11      char deviceName_3[] = "Cocina";
12      char deviceName_4[] = "Recamara";
13
14      //define los GPIO que se van a utilizar
15      static uint8_t RelayPin1 = 23;
16      static uint8_t RelayPin2 = 27;
17      static uint8_t RelayPin3 = 5;
18      static uint8_t RelayPin4 = 18;
19
20      static uint8_t SwitchPin1 = 13;
21      static uint8_t SwitchPin2 = 12;
22      static uint8_t SwitchPin3 = 26;
23      static uint8_t SwitchPin4 = 14;
24
25      static uint8_t wifiLed    = 2;
26      static uint8_t gpio_reset = 15;
27
28      //define el valor de la variable para el estado de los relevadores
29      bool toggleState_1 = LOW;
30      bool toggleState_2 = LOW;
31      bool toggleState_3 = LOW;
32      bool toggleState_4 = LOW;
33
34      // estado de los switch
35      bool SwitchState_1 = LOW;
36      bool SwitchState_2 = LOW;
37      bool SwitchState_3 = LOW;
38      bool SwitchState_4 = LOW;
39
40      //el framework permite definir el comportamiento de algunos
dispositivos estándar como: switch, dimmers, ventiladores, sensores de
temperatura, etc
41      static Switch my_switch1(deviceName_1, &RelayPin1);
42      static Switch my_switch2(deviceName_2, &RelayPin2);
43      static Switch my_switch3(deviceName_3, &RelayPin3);
44      static Switch my_switch4(deviceName_4, &RelayPin4);
45
46      void sysProvEvent(arduino_event_t *sys_event) {
47          switch (sys_event->event_id) {
48              case ARDUINO_EVENT_PROV_START:
49      #if CONFIG_IDF_TARGET_ESP32
50              Serial.printf("\nProvisioning Started with name \"%s\" and
PoP \"%s\" on BLE\n", service_name, pop);
51              printQR(service_name, pop, "ble");
52      #else
53              Serial.printf("\nProvisioning Started with name \"%s\" and
PoP \"%s\" on SoftAP\n", service_name, pop);
54              printQR(service_name, pop, "softap");
55      #endif
```

```
56                break;
57                case ARDUINO_EVENT_WIFI_STA_CONNECTED:
58                Serial.printf("\nConnected to Wifi!\n");
59                digitalWrite(wifiLed, true);
60                break;
61            }
62        }
63
64        void write_callback(Device *device, Param *param, const
param_val_t val, void *priv_data, write_ctx_t *ctx) {
65            const char *device_name = device->getDeviceName();
66            const char *param_name = param->getParamName();
67
68            if(strcmp(device_name, deviceName_1) == 0) {
69
70              Serial.printf("Lightbulb = %s\n", val.val.b? "true" :
"false");
71
72              if(strcmp(param_name, "Power") == 0) {
73                  Serial.printf("Received value = %s for %s - %s\n",
val.val.b? "true" : "false", device_name, param_name);
74                toggleState_1 = val.val.b;
75                (toggleState_1 == false) ? digitalWrite(RelayPin1, HIGH) :
digitalWrite(RelayPin1, LOW);
76                param->updateAndReport(val);
77              }
78
79            } else if(strcmp(device_name, deviceName_2) == 0) {
80
81              Serial.printf("Switch value = %s\n", val.val.b? "true" :
"false");
82
83              if(strcmp(param_name, "Power") == 0) {
84                Serial.printf("Received value = %s for %s - %s\n",
val.val.b? "true" : "false", device_name, param_name);
85                toggleState_2 = val.val.b;
86                (toggleState_2 == false) ? digitalWrite(RelayPin2, HIGH) :
digitalWrite(RelayPin2, LOW);
87                param->updateAndReport(val);
88              }
89
90            } else if(strcmp(device_name, deviceName_3) == 0) {
91
92              Serial.printf("Switch value = %s\n", val.val.b? "true" :
"false");
93
94              if(strcmp(param_name, "Power") == 0) {
95                Serial.printf("Received value = %s for %s - %s\n",
val.val.b? "true" : "false", device_name, param_name);
96                toggleState_3 = val.val.b;
97                (toggleState_3 == false) ? digitalWrite(RelayPin3, HIGH) :
digitalWrite(RelayPin3, LOW);
98                param->updateAndReport(val);
99              }
```

```
100
101         } else if(strcmp(device_name, deviceName_4) == 0) {
102
103           Serial.printf("Switch value = %s\n", val.val.b? "true" :
"false");
104
105           if(strcmp(param_name, "Power") == 0) {
106             Serial.printf("Received value = %s for %s - %s\n",
val.val.b? "true" : "false", device_name, param_name);
107             toggleState_4 = val.val.b;
108             (toggleState_4 == false) ? digitalWrite(RelayPin4, HIGH) :
digitalWrite(RelayPin4, LOW);
109             param->updateAndReport(val);
110           }
111         }
112     }
113
114     void Manual_control() {
115       if (digitalRead(SwitchPin1) == LOW && SwitchState_1 == LOW) {
116         digitalWrite(RelayPin1, LOW);
117         toggleState_1 = 1;
118         SwitchState_1 = HIGH;
119         my_switch1.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_1);
120         Serial.println("Switch-1 on");
121       }
122       if (digitalRead(SwitchPin1) == HIGH && SwitchState_1 == HIGH) {
123         digitalWrite(RelayPin1, HIGH);
124         toggleState_1 = 0;
125         SwitchState_1 = LOW;
126         my_switch1.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_1);
127         Serial.println("Switch-1 off");
128       }
129       if (digitalRead(SwitchPin2) == LOW && SwitchState_2 == LOW) {
130         digitalWrite(RelayPin2, LOW);
131         toggleState_2 = 1;
132         SwitchState_2 = HIGH;
133         my_switch2.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_2);
134         Serial.println("Switch-2 on");
135       }
136       if (digitalRead(SwitchPin2) == HIGH && SwitchState_2 == HIGH) {
137         digitalWrite(RelayPin2, HIGH);
138         toggleState_2 = 0;
139         SwitchState_2 = LOW;
140         my_switch2.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_2);
141         Serial.println("Switch-2 off");
142       }
143       if (digitalRead(SwitchPin3) == LOW && SwitchState_3 == LOW) {
144         digitalWrite(RelayPin3, LOW);
145         toggleState_3 = 1;
146         SwitchState_3 = HIGH;
```

```
147         my_switch3.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_3);
148         Serial.println("Switch-3 on");
149       }
150       if (digitalRead(SwitchPin3) == HIGH && SwitchState_3 == HIGH) {
151         digitalWrite(RelayPin3, HIGH);
152         toggleState_3 = 0;
153         SwitchState_3 = LOW;
154         my_switch3.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_3);
155         Serial.println("Switch-3 off");
156       }
157       if (digitalRead(SwitchPin4) == LOW && SwitchState_4 == LOW) {
158         digitalWrite(RelayPin4, LOW);
159         toggleState_4 = 1;
160         SwitchState_4 = HIGH;
161         my_switch4.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_4);
162         Serial.println("Switch-4 on");
163       }
164       if (digitalRead(SwitchPin4) == HIGH && SwitchState_4 == HIGH) {
165         digitalWrite(RelayPin4, HIGH);
166         toggleState_4 = 0;
167         SwitchState_4 = LOW;
168         my_switch4.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_4);
169         Serial.println("Switch-4 off");
170       }
171     }
172
173     void setup() {
174         uint32_t chipId = 0;
175
176         Serial.begin(115200);
177
178         pinMode(RelayPin1, OUTPUT);
179         pinMode(RelayPin2, OUTPUT);
180         pinMode(RelayPin3, OUTPUT);
181         pinMode(RelayPin4, OUTPUT);
182         pinMode(wifiLed, OUTPUT);
183
184         pinMode(SwitchPin1, INPUT_PULLUP);
185         pinMode(SwitchPin2, INPUT_PULLUP);
186         pinMode(SwitchPin3, INPUT_PULLUP);
187         pinMode(SwitchPin4, INPUT_PULLUP);
188         pinMode(gpio_reset, INPUT);
189
190         //se asigna el valor de estado a las GPIO
191         digitalWrite(RelayPin1, !toggleState_1);
192         digitalWrite(RelayPin2, !toggleState_2);
193         digitalWrite(RelayPin3, !toggleState_3);
194         digitalWrite(RelayPin4, !toggleState_4);
195         digitalWrite(wifiLed, LOW);
196
```

```
197         Node my_node;
198         my_node = RMaker.initNode("ESP32_Iluminación");
199
200         //se indica el tipo de dispositivo estándar
201         my_switch1.addCb(write_callback);
202         my_switch2.addCb(write_callback);
203         my_switch3.addCb(write_callback);
204         my_switch4.addCb(write_callback);
205
206         //se añade cada dispositivo en el nodo
207         my_node.addDevice(my_switch1);
208         my_node.addDevice(my_switch2);
209         my_node.addDevice(my_switch3);
210         my_node.addDevice(my_switch4);
211
212         //este set de instrucciones es opcional, permite definir
región y zona horaria entre la tarjeta de desarrollo y el servidor
213         RMaker.enableOTA(OTA_USING_PARAMS);
214         RMaker.enableTZService();
215         RMaker.enableSchedule();
216
217         //configuraciones del servicio proporcionado por el servidor
218         for(int i=0; i<17; i=i+8) {
219           chipId |= ((ESP.getEfuseMac() >> (40 - i)) & 0xff) << i;
220         }
221
222         Serial.printf("\nChip ID:  %d Service Name: %s\n", chipId,
service_name);
223
224         Serial.printf("\nStarting ESP-RainMaker\n");
225         RMaker.start();
226
227         Wifi.onEvent(sysProvEvent);
228     #if CONFIG_IDF_TARGET_ESP32
229         WifiProv.beginProvision(WIFI_PROV_SCHEME_BLE,
WIFI_PROV_SCHEME_HANDLER_FREE_BTDM, WIFI_PROV_SECURITY_1, pop,
service_name);
230     #else
231         WifiProv.beginProvision(WIFI_PROV_SCHEME_SOFTAP,
WIFI_PROV_SCHEME_HANDLER_NONE, WIFI_PROV_SECURITY_1, pop, service_name);
232     #endif
233
234         my_switch1.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
false);
235         my_switch2.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
false);
236         my_switch3.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
false);
237         my_switch4.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
false);
238     }
239
240     void loop() {
241         if(digitalRead(gpio_reset) == LOW) {
```

```
242             Serial.printf("Reset Button Pressed!\n");
243             delay(100);
244             int startTime = millis();
245             while(digitalRead(gpio_reset) == LOW) delay(50);
246             int endTime = millis();
247
248             if ((endTime - startTime) > 10000) {
249               Serial.printf("Reset to factory.\n");
250               RMakerFactoryReset(2);
251             } else if ((endTime - startTime) > 3000) {
252               Serial.printf("Reset Wifi.\n");
253               RMakerWifiReset(2);
254             }
255         }
256         delay(100);
257
258         if (Wifi.status() != WL_CONNECTED)
259         {
260           //Serial.println("Wifi Not Connected");
261           digitalWrite(wifiLed, false);
262         }
263         else
264         {
265           //Serial.println("Wifi Connected");
266           digitalWrite(wifiLed, true);
267         }
268
269         Manual_control();
270     }
```

Anexo C. Programa control de potencia[1]

```
1       //aquí se pega la información de las credenciales de la plantilla
creada en Blynk
2       #define BLYNK_TEMPLATE_ID "TMPLxxxxxx"
3       #define BLYNK_DEVICE_NAME "DIMMER"
4
5       #define BLYNK_FIRMWARE_VERSION        "0.1.0"
6
7       #define BLYNK_PRINT Serial
8       //#define BLYNK_DEBUG
9
10      #define APP_DEBUG
11
12      // Uncomment your board, or configure a custom board in Settings.h
13      //#define USE_WROVER_BOARD
14      //#define USE_TTGO_T7
15      //#define USE_ESP32C3_DEV_MODULE
16      //#define USE_ESP32S2_DEV_KIT
17
```

[1] El desarrollo de los programas contiene líneas de código provenientes del trabajo colectivo en los foros oficiales de consulta en los sitios web: https://community.blynk.cc/; https://forum.arduino.cc/; https://github.com/

```
18      #include "BlynkEdgent.h"
19      #include <RBDdimmer.h>
20
21      #define outputPin 27
22      #define zerocross 12
23
24      dimmerLamp dimmer(outputPin, zerocross);
25
26      int outVal = 100;
27      int preVal = 0;
28      bool power = 0;
29
30      void setup() {
31        Serial.begin(115200);
32        dimmer.begin(NORMAL_MODE, ON);
33        dimmer.setState(OFF);
34        delay(100);
35
36        BlynkEdgent.begin();
37      }
38
39      BLYNK_WRITE(V0) {
40        outVal = map(param.asInt(), 0, 100, 13, 86);
41        if(preVal != outVal){
42        dimmer.setPower(outVal);
43        }
44      }
45
46      BLYNK_WRITE(V1) {
47        power = param.asInt();
48        if(power == 0){
49          delay(10);
50          dimmer.setState(OFF);
51        }
52
53        if(power == 1){
54          delay(10);
55          dimmer.setState(ON);
56        }
57      }
58
59      void loop() {
60        BlynkEdgent.run();
61      }
```

Anexo D. Programa control y monitoreo de nivel de agua[1]

```
1       //aquí se pega la información de las credenciales de la plantilla
creada en Blynk
2       #define BLYNK_TEMPLATE_ID "TMPLCGlFtzxn"
```

[1] El desarrollo de los programas contiene líneas de código provenientes del trabajo colectivo en los foros oficiales de consulta en los sitios web: https://community.blynk.cc/; https://forum.arduino.cc/; https://github.com/

```
#define BLYNK_DEVICE_NAME "NIVEL DE AGUA"

#define BLYNK_FIRMWARE_VERSION        "0.1.0"

#define BLYNK_PRINT Serial
//#define BLYNK_DEBUG

#define APP_DEBUG

#define GPIO13 13
#define GPIO12 12
#define GPIO14 14
#define GPIO27 27
#define GPIO26 26
#define GPIO25 25
#define GPIO33 33
#define GPIO32 32
#define GPIO23 23
#define SwitchPin1 19

int pin15 = 0;
int pin2  = 0;
int pin0  = 0;
int pin4  = 0;
int pin16 = 0;
int pin17 = 0;
int pin5  = 0;
int pin18 = 0;
float Sensor_1 = 0;
float Sensor_2 = 0;
bool SwitchState_1 = LOW;
bool toggleState_1 = LOW;
bool StateAuto = LOW;

// Uncomment your board, or configure a custom board in Settings.h
//#define USE_WROVER_BOARD
//#define USE_TTGO_T7
//#define USE_ESP32C3_DEV_MODULE
//#define USE_ESP32S2_DEV_KIT

#include "BlynkEdgent.h"

BLYNK_CONNECTED() {
  Blynk.syncVirtual(V3);
  Blynk.syncVirtual(V4);
}

BLYNK_WRITE(V3) {
  StateAuto = param.asInt();
}

BLYNK_WRITE(V4) {
  toggleState_1 = param.asInt();
  if(toggleState_1 == 1){
```

```
57          digitalWrite(18, LOW);
58        }
59        else {
60          digitalWrite(18, HIGH);
61        }
62      }
63
64      void setup() {
65        Serial.begin(115200);
66        delay(100);
67        pinMode(GPIO13, INPUT_PULLUP);
68        pinMode(GPIO12, INPUT_PULLUP);
69        pinMode(GPIO14, INPUT_PULLUP);
70        pinMode(GPIO27, INPUT_PULLUP);
71        pinMode(GPIO26, INPUT_PULLUP);
72        pinMode(GPIO25, INPUT_PULLUP);
73        pinMode(GPIO33, INPUT_PULLUP);
74        pinMode(GPIO32, INPUT_PULLUP);
75        pinMode(GPIO23, OUTPUT);
76        pinMode(18, OUTPUT);
77        pinMode(SwitchPin1, INPUT_PULLUP);
78
79        BlynkEdgent.begin();
80      }
81
82      void State_online() {
83        if(StateAuto == 1){
84          if(Sensor_1 <= 25){
85          Blynk.virtualWrite(V1, HIGH);
86          digitalWrite(18, HIGH);
87          if(Sensor_2 <= 25){
88            Blynk.logEvent("nivel_de_cisterna");
89            Blynk.virtualWrite(V3, LOW);
90            StateAuto=0;
91            Blynk.virtualWrite(V1, LOW);
92            digitalWrite(18, LOW);
93          }
94        }
95          else if(Sensor_1 == 100){
96          Blynk.virtualWrite(V1, LOW);
97          digitalWrite(18, LOW);
98          }
99       }
100       else {
101         Blynk.virtualWrite(V1, toggleState_1);
102         digitalWrite(18, toggleState_1);
103       }
104     }
105
106     void reading() {
107       digitalWrite(GPIO23, HIGH);
108       delay(100);
109       pin15 = digitalRead(GPIO13);
110       pin2  = digitalRead(GPIO12);
```

```
111       pin0  = digitalRead(GPIO14);
112       pin4  = digitalRead(GPIO27);
113       pin16 = digitalRead(GPIO26);
114       pin17 = digitalRead(GPIO25);
115       pin5  = digitalRead(GPIO33);
116       pin18 = digitalRead(GPIO32);
117       delay(100);
118       digitalWrite(GPIO23, LOW);
119
120     if(pin15 == LOW && pin2 == LOW && pin0 == LOW && pin4 == LOW){
121         Sensor_1 =100;
122         }
123     if(pin15 == HIGH && pin2 == LOW && pin0 == LOW && pin4 == LOW){
124         Sensor_1 =75;
125         }
126     if(pin15 == HIGH && pin2 == HIGH && pin0 == LOW && pin4 == LOW){
127         Sensor_1 =50;
128         }
129     if(pin15 == HIGH && pin2 == HIGH && pin0 == HIGH && pin4 == LOW){
130         Sensor_1 =25;
131         }
132     if(pin15 == HIGH && pin2 == HIGH && pin0 == HIGH && pin4 == HIGH){
133         Sensor_1 =0;
134         }
135     if(pin16 == LOW && pin17 == LOW && pin5 == LOW && pin18 == LOW){
136         Sensor_2 =100;
137         }
138     if(pin16 == HIGH && pin17 == LOW && pin5 == LOW && pin18 == LOW){
139         Sensor_2 =75;
140         }
141     if(pin16 == HIGH && pin17 == HIGH && pin5 == LOW && pin18 == LOW){
142         Sensor_2 =50;
143         }
144     if(pin16 == HIGH && pin17 == HIGH && pin5 == HIGH && pin18 ==
LOW){
145         Sensor_2 =25;
146         }
147     if(pin16 == HIGH && pin17 == HIGH && pin5 == HIGH && pin18 ==
HIGH){
148         Sensor_2 =0;
149         }
150
151       Blynk.virtualWrite(V0, Sensor_1);
152       Blynk.virtualWrite(V5, Sensor_2);
153     }
154
155     void manual_control() {
156       if (digitalRead(SwitchPin1) == LOW && SwitchState_1 == LOW) {
157         digitalWrite(18, LOW);
158         Blynk.virtualWrite(V1, HIGH);
159         Blynk.virtualWrite(V4, HIGH);
160         toggleState_1 = HIGH;
161         SwitchState_1 = HIGH;
162         StateAuto = HIGH;
```

```
163         Blynk.virtualWrite(V3, HIGH);
164       }
165       if (digitalRead(SwitchPin1) == HIGH && SwitchState_1 == HIGH) {
166         digitalWrite(18, HIGH);
167         Blynk.virtualWrite(V1, LOW);
168         Blynk.virtualWrite(V4, LOW);
169         toggleState_1 = LOW;
170         SwitchState_1 = LOW;
171       }
172     }
173
174     void loop() {
175       State_online();
176       reading();
177       manual_control();
178
179       BlynkEdgent.run();
180     }
```

D.1 Configuración adicional para trabajar fuera de línea

Cabecera BlynkEdgent.h

```
1       extern "C" {
2         void app_loop();
3         void restartMCU();
4       }
5
6       #include "Settings.h"
7       #include <BlynkSimpleEsp32_SSL.h>
8
9       #ifndef BLYNK_NEW_LIBRARY
10      #error "Old version of Blynk library is in use. Please replace it
with the new one."
11      #endif
12
13      #if !defined(BLYNK_TEMPLATE_ID) || !defined(BLYNK_DEVICE_NAME)
14      #error "Please specify your BLYNK_TEMPLATE_ID and
BLYNK_DEVICE_NAME"
15      #endif
16
17      BlynkTimer edgentTimer;
18
19      #include "BlynkState.h"
20      #include "ConfigStore.h"
21      #include "ResetButton.h"
22      #include "ConfigMode.h"
23      #include "Indicator.h"
24      #include "OTA.h"
25      #include "Console.h"
26
27      void State_online();
28      void reading();
29      void manual_control();
```

```
30
31      inline
32      void BlynkState::set(State m) {
33        if (state != m && m < MODE_MAX_VALUE) {
34          DEBUG_PRINT(String(StateStr[state]) + " => " + StateStr[m]);
35          state = m;
36
37          // You can put your state handling here,
38          // i.e. implement custom indication
39        }
40      }
41
42      void printDeviceBanner()
43      {
44        Blynk.printBanner();
45        DEBUG_PRINT("-------------------------");
46        DEBUG_PRINT(String("Product:  ") + BLYNK_DEVICE_NAME);
47        DEBUG_PRINT(String("Firmware: ") + BLYNK_FIRMWARE_VERSION "
(build " __DATE__ " " __TIME__ ")");
48        if (configStore.getFlag(CONFIG_FLAG_VALID)) {
49          DEBUG_PRINT(String("Token:    ") +
50                      String(configStore.cloudToken).substring(0,4) +
51                      " - •••• - •••• - ••••");
52        }
53        DEBUG_PRINT(String("Device:   ") + BLYNK_INFO_DEVICE + " @ " +
ESP.getCpuFreqMHz() + "MHz");
54        DEBUG_PRINT(String("MAC:      ") + Wifi.macAddress());
55        DEBUG_PRINT(String("Flash:    ") + ESP.getFlashChipSize() / 1024
+ "K");
56        DEBUG_PRINT(String("ESP sdk:  ") + ESP.getSdkVersion());
57        DEBUG_PRINT(String("Chip rev: ") + ESP.getChipRevision());
58        DEBUG_PRINT(String("Free mem: ") + ESP.getFreeHeap());
59        DEBUG_PRINT("-------------------------");
60      }
61
62      void runBlynkWithChecks() {
63        Blynk.run();
64        if (BlynkState::get() == MODE_RUNNING) {
65          if (!Blynk.connected()) {
66            if (Wifi.status() == WL_CONNECTED) {
67              BlynkState::set(MODE_CONNECTING_CLOUD);
68            } else {
69              BlynkState::set(MODE_CONNECTING_NET);
70            }
71          }
72        }
73      }
74
75      class Edgent {
76
77      public:
78        void begin()
79        {
80          Wifi.persistent(false);
```

```
81          Wifi.enableSTA(true); // Needed to get MAC
82
83          indicator_init();
84          button_init();
85          config_init();
86          printDeviceBanner();
87          console_init();
88
89          if (configStore.getFlag(CONFIG_FLAG_VALID)) {
90            BlynkState::set(MODE_CONNECTING_NET);
91          } else if (config_load_blnkopt()) {
92            DEBUG_PRINT("Firmware is preprovisioned");
93            BlynkState::set(MODE_CONNECTING_NET);
94          } else {
95            BlynkState::set(MODE_WAIT_CONFIG);
96          }
97        }
98
99        void run() {
100         app_loop();
101         switch (BlynkState::get()) {
102         case MODE_WAIT_CONFIG:
103         case MODE_CONFIGURING:       enterConfigMode();    break;
104         case MODE_CONNECTING_NET:    enterConnectNet();    break;
105         case MODE_CONNECTING_CLOUD:  enterConnectCloud();  break;
106         case MODE_RUNNING:           runBlynkWithChecks(); break;
107         case MODE_OTA_UPGRADE:       enterOTA();           break;
108         case MODE_SWITCH_TO_STA:     enterSwitchToSTA();   break;
109         case MODE_RESET_CONFIG:      enterResetConfig();   break;
110         default:                     enterError();         break;
111         }
112       }
113
114     } BlynkEdgent;
115
116     void app_loop() {
117         edgentTimer.run();
118         edgentConsole.run();
119         State_online();
120         reading();
121         manual_control();
122     }
```

Anexo E. Programa control de iluminación RGB[1]

```
1       //aquí se pega la información de las credenciales de la plantilla
creada en Blynk
2       #define BLYNK_TEMPLATE_ID "TMPLxxxxxx"
3       #define BLYNK_DEVICE_NAME "RGB"
4
```

[1] El desarrollo de los programas contiene líneas de código provenientes del trabajo colectivo en los foros oficiales de consulta en los sitios web: https://community.blynk.cc/; https://forum.arduino.cc/; https://github.com/

```
#define BLYNK_FIRMWARE_VERSION        "0.1.0"

#define BLYNK_PRINT Serial
//#define BLYNK_DEBUG

#define APP_DEBUG

// Uncomment your board, or configure a custom board in Settings.h
//#define USE_WROVER_BOARD
//#define USE_TTGO_T7
//#define USE_ESP32C3_DEV_MODULE
//#define USE_ESP32S2_DEV_KIT

#include "BlynkEdgent.h"

uint8_t ledR = 12;
uint8_t ledG = 13;
uint8_t ledB = 14;
int PWM_LEDR;
int PWM_LEDG;
int PWM_LEDB;

void setup() {
  ledcAttachPin(ledR, 5);
  ledcAttachPin(ledG, 2);
  ledcAttachPin(ledB, 3);
  ledcSetup(5, 12000, 8);
  ledcSetup(2, 12000, 8);
  ledcSetup(3, 12000, 8);
  Serial.begin(115200);
  ledcWrite(5, 255);
  ledcWrite(2, 255);
  ledcWrite(3, 255);
  delay(100);

  BlynkEdgent.begin();
}

BLYNK_WRITE(V0) {
 PWM_LEDR = map(param.asInt(), 0, 255, 255, 0);
 ledcWrite(5, PWM_LEDR);
}
BLYNK_WRITE(V1) {
 PWM_LEDG = map(param.asInt(), 0, 255, 255, 0);
 ledcWrite(2, PWM_LEDG);
}
BLYNK_WRITE(V2) {
 PWM_LEDB = map(param.asInt(), 0, 255, 255, 0);
 ledcWrite(3, PWM_LEDB);
}

void loop() {
  BlynkEdgent.run();
}
```

Anexo F. Programa control, monitoreo y riego de sustrato[1]

```
1       //aquí se pega la información de las credenciales de la plantilla
creada en Blynk
2       #define BLYNK_TEMPLATE_ID "TMPLxxxxxxx"
3       #define BLYNK_DEVICE_NAME "MACETA IOT"
4
5       #define BLYNK_FIRMWARE_VERSION        "0.1.0"
6
7       #define BLYNK_PRINT Serial
8       //#define BLYNK_DEBUG
9
10      #include <Preferences.h>
11      #include<LiquidCrystal.h>
12
13      LiquidCrystal lcd(13, 12, 14, 27, 26, 25);
14      Preferences referencia;
15
16      #define SwitchPin1 23
17      #define VPIN_BUTTON_1    V3
18      #define VPIN_BUTTON_2    V4
19
20      bool toggleState_1 = LOW;
21      uint8_t numberKeyPresses;
22      uint8_t limVal;
23      const int Relay = 32;
24      const int Sensor = 33;
25      int value = 0;
26
27      #define APP_DEBUG
28
29      // Uncomment your board, or configure a custom board in Settings.h
30      //#define USE_WROVER_BOARD
31      //#define USE_TTGO_T7
32      //#define USE_ESP32C3_DEV_MODULE
33      //#define USE_ESP32S2_DEV_KIT
34
35      struct Button {
36        const uint8_t PIN;
37        //uint32_t numberKeyPresses;
38        bool pressed;
39      };
40      struct Button2 {
41        const uint8_t PIN2;
42        //uint32_t numberKeyPresses;
43        bool pressed2;
44      };
45
46      Button button1 = {18, false};
47      Button2 button2 = {19, false};
48
```

[1] El desarrollo de los programas contiene líneas de código provenientes del trabajo colectivo en los foros oficiales de consulta en los sitios web: https://community.blynk.cc/; https://forum.arduino.cc/; https://github.com/

```
49      #include "BlynkEdgent.h"
50
51      BLYNK_WRITE(VPIN_BUTTON_1) {
52        toggleState_1 = param.asInt();
53        if (toggleState_1 == 1) {
54          referencia.putUInt("level", limVal);
55          lcd.clear();
56          lcd.setCursor(0,0);
57          lcd.print("Valor de humedad");
58          lcd.setCursor(4,1);
59          lcd.print("guardado");
60          delay(3000);
61          lcd.clear();
62        }
63      }
64
65      BLYNK_WRITE(VPIN_BUTTON_2) {
66        limVal=param.asInt();
67      }
68
69      void IRAM_ATTR isr() {
70        numberKeyPresses--;
71        button1.pressed = true;
72      }
73
74      void IRAM_ATTR isr2() {
75        numberKeyPresses++;
76        button2.pressed2 = true;
77      }
78
79      void display() {
80          lcd.setCursor(0,1);
81          lcd.print("Referencia=  ");
82          lcd.setCursor(11,1);
83          lcd.print(limVal);
84          lcd.setCursor(0,0);
85          lcd.print("Humedad=    ");
86          if (value>9){
87             lcd.setCursor(8,0);
88             lcd.print(value);
89             delay(300);
90          }
91          else{
92             lcd.setCursor(9,0);
93             lcd.print(value);
94             delay(300);
95          }
96          lcd.setCursor(11,0);
97          lcd.print("%");
98          delay(20);
99      }
100
101     void irrigation() {
102       if (value<=(limVal-5)){
```

```
        digitalWrite(Relay, HIGH);
        Blynk.virtualWrite(V1, 1);
      }
      else if (value>=(limVal+5)){
        digitalWrite(Relay, LOW);
        Blynk.virtualWrite(V1, 0);
      }
    }

    void reading() {
      int const valuesens = analogRead(Sensor);
      int const read = map(valuesens, 3600, 1450, 0, 100);
      if(read!=limVal){
            value=read;
            display();
          }
          irrigation();
      if (button1.pressed) {
        if (numberKeyPresses <=99 && numberKeyPresses >=0) {
          limVal = constrain(numberKeyPresses, 0, 99);
        }
          button1.pressed = false;
      }
      if (button2.pressed2) {
        if (numberKeyPresses <=99 && numberKeyPresses >=0) {
          limVal = constrain(numberKeyPresses, 0, 99);
        }
          button2.pressed2 = false;
      }
      Blynk.virtualWrite(V0, read);
      Blynk.virtualWrite(V2, limVal);
      Blynk.virtualWrite(V4, limVal);
    }

    void manual_control() {
      if (digitalRead(SwitchPin1) == LOW) {
        referencia.putUInt("level", limVal);
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Valor de humedad");
        lcd.setCursor(4,1);
        lcd.print("guardado");
        delay(3000);
        lcd.clear();
      }
    }

    void setup() {
      Serial.begin(115200);
      delay(100);
      lcd.begin(16,2);
      referencia.begin("humedad", false);
      pinMode(SwitchPin1, INPUT_PULLUP);
      pinMode(button1.PIN, INPUT_PULLUP);
```

```
157       attachInterrupt(button1.PIN, isr, FALLING);
158       pinMode(button2.PIN2, INPUT_PULLUP);
159       attachInterrupt(button2.PIN2, isr2, FALLING);
160       pinMode(Relay, OUTPUT);
161       pinMode(Sensor, INPUT);
162       limVal = referencia.getUInt("level", false);
163       numberKeyPresses = limVal;
164       lcd.clear();
165       BlynkEdgent.begin();
166     }
167
168     void loop() {
169       display();
170       irrigation();
171       manual_control();
172       reading();
173
174       BlynkEdgent.run();
175     }
```

F.1 Configuración adicional para trabajar fuera de línea

Cabecera BlynkEdgent.h

```
1       extern "C" {
2         void app_loop();
3         void restartMCU();
4       }
5
6       #include "Settings.h"
7       #include <BlynkSimpleEsp32_SSL.h>
8
9       #ifndef BLYNK_NEW_LIBRARY
10      #error "Old version of Blynk library is in use. Please replace it
with the new one."
11      #endif
12
13      #if !defined(BLYNK_TEMPLATE_ID) || !defined(BLYNK_DEVICE_NAME)
14      #error "Please specify your BLYNK_TEMPLATE_ID and
BLYNK_DEVICE_NAME"
15      #endif
16
17      BlynkTimer edgentTimer;
18
19      #include "BlynkState.h"
20      #include "ConfigStore.h"
21      #include "ResetButton.h"
22      #include "ConfigMode.h"
23      #include "Indicator.h"
24      #include "OTA.h"
25      #include "Console.h"
26
27      void reading();
28      void display();
```

```
29      void irrigation();
30      void manual_control();
31
32      inline
33      void BlynkState::set(State m) {
34        if (state != m && m < MODE_MAX_VALUE) {
35          DEBUG_PRINT(String(StateStr[state]) + " => " + StateStr[m]);
36          state = m;
37
38          // You can put your state handling here,
39          // i.e. implement custom indication
40        }
41      }
42
43      void printDeviceBanner()
44      {
45        Blynk.printBanner();
46        DEBUG_PRINT("-------------------------");
47        DEBUG_PRINT(String("Product:  ") + BLYNK_DEVICE_NAME);
48        DEBUG_PRINT(String("Firmware: ") + BLYNK_FIRMWARE_VERSION "
(build " __DATE__ " " __TIME__ ")");
49        if (configStore.getFlag(CONFIG_FLAG_VALID)) {
50          DEBUG_PRINT(String("Token:    ") +
51                      String(configStore.cloudToken).substring(0,4) +
52                      " - •••• - •••• - ••••");
53        }
54        DEBUG_PRINT(String("Device:   ") + BLYNK_INFO_DEVICE + " @ " +
ESP.getCpuFreqMHz() + "MHz");
55        DEBUG_PRINT(String("MAC:      ") + Wifi.macAddress());
56        DEBUG_PRINT(String("Flash:    ") + ESP.getFlashChipSize() / 1024
+ "K");
57        DEBUG_PRINT(String("ESP sdk:  ") + ESP.getSdkVersion());
58        DEBUG_PRINT(String("Chip rev: ") + ESP.getChipRevision());
59        DEBUG_PRINT(String("Free mem: ") + ESP.getFreeHeap());
60        DEBUG_PRINT("-------------------------");
61      }
62
63      void runBlynkWithChecks() {
64        Blynk.run();
65        if (BlynkState::get() == MODE_RUNNING) {
66          if (!Blynk.connected()) {
67            if (Wifi.status() == WL_CONNECTED) {
68              BlynkState::set(MODE_CONNECTING_CLOUD);
69            } else {
70              BlynkState::set(MODE_CONNECTING_NET);
71            }
72          }
73        }
74      }
75
76      class Edgent {
77
78      public:
79        void begin()
```

```
80          {
81            Wifi.persistent(false);
82            Wifi.enableSTA(true); // Needed to get MAC
83
84            indicator_init();
85            button_init();
86            config_init();
87            printDeviceBanner();
88            console_init();
89
90            if (configStore.getFlag(CONFIG_FLAG_VALID)) {
91              BlynkState::set(MODE_CONNECTING_NET);
92            } else if (config_load_blnkopt()) {
93              DEBUG_PRINT("Firmware is preprovisioned");
94              BlynkState::set(MODE_CONNECTING_NET);
95            } else {
96              BlynkState::set(MODE_WAIT_CONFIG);
97            }
98          }
99
100         void run() {
101           app_loop();
102           switch (BlynkState::get()) {
103           case MODE_WAIT_CONFIG:
104           case MODE_CONFIGURING:       enterConfigMode();    break;
105           case MODE_CONNECTING_NET:    enterConnectNet();    break;
106           case MODE_CONNECTING_CLOUD:  enterConnectCloud();  break;
107           case MODE_RUNNING:           runBlynkWithChecks(); break;
108           case MODE_OTA_UPGRADE:       enterOTA();           break;
109           case MODE_SWITCH_TO_STA:     enterSwitchToSTA();   break;
110           case MODE_RESET_CONFIG:      enterResetConfig();   break;
111           default:                     enterError();         break;
112           }
113         }
114
115     } BlynkEdgent;
116
117     void app_loop() {
118         edgentTimer.run();
119         edgentConsole.run();
120         manual_control();
121         reading();
122         display();
123         irrigation();
124     }
```

Anexo G. Programa control de accesos[1]

```
1       //aquí se pega la información de las credenciales de la plantilla
creada en Blynk
```

[1] El desarrollo de los programas contiene líneas de código provenientes del trabajo colectivo en los foros oficiales de consulta en los sitios web: https://community.blynk.cc/; https://forum.arduino.cc/; https://github.com/

```
2       #define BLYNK_TEMPLATE_ID "TMPLxxxxxx"
3       #define BLYNK_DEVICE_NAME "CONTROL DE ACCESO"
4
5       #define BLYNK_FIRMWARE_VERSION        "0.1.0"
6
7       #define BLYNK_PRINT Serial
8       //#define BLYNK_DEBUG
9
10      #define APP_DEBUG
11      #include <SPI.h>
12      #include <MFRC522.h>
13      #include <EEPROM.h>
14
15      #define EEPROM_SIZE 512
16      #define SS_PIN 5
17      #define RST_PIN 22
18      #define BTN_PIN 14
19      #define SLN_PIN 13
20
21      MFRC522 mfrc522(SS_PIN, RST_PIN);
22
23      unsigned long uidDec, uidDecTemp;
24      int ARRAYindexUIDcard;
25      int EEPROMstartAddr;
26      long adminID = 1122539531;
27      bool beginCard = 0;
28      bool addCard = 1;
29      bool skipCard = 0;
30      int LockSwitch;
31      unsigned long CardUIDeEPROMread[] = {0, 1, 2, 3, 4, 5, 6, 7, 8,
9};
32      int BUZZPIN = 12;
33
34      // Uncomment your board, or configure a custom board in Settings.h
35      //#define USE_WROVER_BOARD
36      //#define USE_TTGO_T7
37      //#define USE_ESP32C3_DEV_MODULE
38      //#define USE_ESP32S2_DEV_KIT
39
40      #include "BlynkEdgent.h"
41      WidgetLCD lcd(V0);
42      BlynkTimer timer;
43
44      void setup() {
45        Serial.begin(115200);
46        delay(100);
47        pinMode(SLN_PIN, OUTPUT); digitalWrite(SLN_PIN, LOW);
48        pinMode(BTN_PIN, INPUT_PULLUP);
49        pinMode(BUZZPIN, OUTPUT);
50        SPI.begin();
51        mfrc522.PCD_Init();
52        lcd.clear();
53        EEPROM.begin(EEPROM_SIZE);
54        DisplayWAiT_CARD();
```

```
55          EEPROMreadUIDcard();
56          digitalWrite(BUZZPIN, HIGH), delay(100), digitalWrite(BUZZPIN,
LOW);
57
58          BlynkEdgent.begin();
59        }
60
61        BLYNK_WRITE(V1) {
62          int a = param.asInt();
63          if (a == 1) beginCard = 1; else beginCard = 0;
64        }
65
66        BLYNK_WRITE(V2) {
67          int a = param.asInt();
68          if (a == 1) {
69            beginCard = 1;
70            skipCard = 1;
71            if (EEPROMstartAddr / 5 < 10) EEPROMwriteUIDcard();
72          } else {
73            beginCard = 0;
74            skipCard = 0;
75          }
76        }
77
78        BLYNK_WRITE(V3) {
79          int a = param.asInt();
80          if (a == 1) {
81            digitalWrite(SLN_PIN, HIGH);
82            lcd.clear();
83            lcd.print(0, 0, "DESBLOQ. POR APP");
84            lcd.print(0, 1, " PUERTA ABIERTA ");
85            digitalWrite(BUZZPIN, HIGH), delay(200), digitalWrite(BUZZPIN,
LOW);
86            delay(5000);
87            DisplayWAiT_CARD();
88          }
89        }
90
91        void EEPROMwriteUIDcard() {
92
93          if (LockSwitch == 0) {
94            lcd.clear();
95            lcd.print(0, 0, " GUARDAR LLAVES ");
96            lcd.print(0, 1, "ESCANE UNA A UNA");
97            delay(1000);
98          }
99
100         if (LockSwitch > 0) {
101           if (skipCard == 1) {
102             lcd.clear();
103             lcd.print(0, 0, "REMOVER LLAVES-E");
104             lcd.print(0, 1, "                ");
105             lcd.print(0, 1, "   LABEL : ");
106             lcd.print(11, 1, EEPROMstartAddr / 5);
```

```
107           EEPROMstartAddr += 5;
108           skipCard = 0;
109         }
110         else {
111           Serial.println("writeCard");
112           EEPROM.write(EEPROMstartAddr, uidDec & 0xFF);
113           EEPROM.write(EEPROMstartAddr + 1, (uidDec & 0xFF00) >> 8);
114           EEPROM.write(EEPROMstartAddr + 2, (uidDec & 0xFF0000) >>
16);
115           EEPROM.write(EEPROMstartAddr + 3, (uidDec & 0xFF000000) >>
24);
116           EEPROM.commit();
117           delay(10);
118           lcd.clear();
119           lcd.print(0, 1, "                ");
120           lcd.print(0, 0, " GUARDADO OK EN ");
121           lcd.print(0, 1, "MEMORIA: ");
122           lcd.print(9, 1, EEPROMstartAddr / 5);
123           EEPROMstartAddr += 5;
124           delay(1000);
125         }
126       }
127
128       LockSwitch++;
129
130       if (EEPROMstartAddr / 5 == 10) {
131         lcd.clear();
132         lcd.print(0, 0, "SE HAN GUARDADO ");
133         delay(3000);
134         EEPROMstartAddr = 0;
135         uidDec = 0;
136         ARRAYindexUIDcard = 0;
137         EEPROMreadUIDcard();
138       }
139     }
140
141     void EEPROMreadUIDcard() {
142       for (int i = 0; i <= 9; i++) {
143         byte val = EEPROM.read(EEPROMstartAddr + 3);
144         CardUIDeEPROMread[ARRAYindexUIDcard] =
(CardUIDeEPROMread[ARRAYindexUIDcard] << 8) | val;
145         val = EEPROM.read(EEPROMstartAddr + 2);
146         CardUIDeEPROMread[ARRAYindexUIDcard] =
(CardUIDeEPROMread[ARRAYindexUIDcard] << 8) | val;
147         val = EEPROM.read(EEPROMstartAddr + 1);
148         CardUIDeEPROMread[ARRAYindexUIDcard] =
(CardUIDeEPROMread[ARRAYindexUIDcard] << 8) | val;
149         val = EEPROM.read(EEPROMstartAddr);
150         CardUIDeEPROMread[ARRAYindexUIDcard] =
(CardUIDeEPROMread[ARRAYindexUIDcard] << 8) | val;
151
152         ARRAYindexUIDcard++;
153         EEPROMstartAddr += 5;
154       }
```

```
155
156       ARRAYindexUIDcard = 0;
157       EEPROMstartAddr = 0;
158       uidDec = 0;
159       LockSwitch = 0;
160       DisplayWAiT_CARD();
161     }
162
163     void DisplayWAiT_CARD() {
164       lcd.clear();
165       lcd.print(0, 0, " ESCANEAR TARJ. ");
166       lcd.print(0, 1, "   O LLAVE-E    ");
167     }
168
169     void manual_control() {
170     digitalWrite(SLN_PIN, LOW);
171
172       if (digitalRead(BTN_PIN) == LOW) {
173         digitalWrite(SLN_PIN, HIGH);
174         lcd.clear();
175         lcd.print(0, 0, " DBQ. POR BOTON ");
176         lcd.print(0, 1, " PUERTA ABIERTA ");
177         digitalWrite(BUZZPIN, HIGH), delay(200), digitalWrite(BUZZPIN,
LOW);
178         delay(5000);
179         DisplayWAiT_CARD();
180       }
181
182       if (beginCard == 0) {
183         if ( ! mfrc522.PICC_IsNewCardPresent()) {
184           Blynk.run();
185           return;
186         }
187
188         if ( ! mfrc522.PICC_ReadCardSerial()) {
189           Blynk.run();
190           return;
191         }
192       }
193       for (byte i = 0; i < mfrc522.uid.size; i++) {
194         uidDecTemp = mfrc522.uid.uidByte[i];
195         uidDec = uidDec * 256 + uidDecTemp;
196       }
197
198       if (beginCard == 1 || LockSwitch > 0)EEPROMwriteUIDcard();
199
200       if (LockSwitch == 0) {
201         for (ARRAYindexUIDcard = 0; ARRAYindexUIDcard <= 9;
ARRAYindexUIDcard++) {
202           if (CardUIDeEPROMread[ARRAYindexUIDcard] > 0) {
203             if (CardUIDeEPROMread[ARRAYindexUIDcard] == uidDec) {
204               lcd.clear();
205               lcd.print(0, 0, "LLAVE OK ABIERTO");
206               lcd.print(3, 1, uidDec);
```

```
207                digitalWrite(SLN_PIN, HIGH);
208                digitalWrite(BUZZPIN, HIGH), delay(200),
digitalWrite(BUZZPIN, LOW);
209                delay(5000);
210                break;
211              }
212            }
213          }
214
215          if (ARRAYindexUIDcard == 10) {
216            lcd.clear();
217            lcd.print(0, 0, " LLAVE SIN REG. ");
218            lcd.print(0, 1, "                ");
219            lcd.print(0, 1, "ID : ");
220            lcd.print(5, 1, uidDec);
221            for (int i = 0; i <= 2; i++)delay(100),
digitalWrite(BUZZPIN, HIGH), delay(100), digitalWrite(BUZZPIN, LOW);
222            digitalWrite(SLN_PIN, LOW);
223            delay(2000);
224          }
225
226          ARRAYindexUIDcard = 0;
227          DisplayWAiT_CARD();
228        }
229      }
230
231      void loop() {
232        manual_control();
233        BlynkEdgent.run();
234        timer.run();
235      }
```

G.1 Configuración adicional para trabajar fuera de línea

Cabecera BlynkEdgent.h

```
1        extern "C" {
2          void app_loop();
3          void restartMCU();
4        }
5
6        #include "Settings.h"
7        #include <BlynkSimpleEsp32_SSL.h>
8
9        #ifndef BLYNK_NEW_LIBRARY
10       #error "Old version of Blynk library is in use. Please replace it
with the new one."
11       #endif
12
13       #if !defined(BLYNK_TEMPLATE_ID) || !defined(BLYNK_DEVICE_NAME)
14       #error "Please specify your BLYNK_TEMPLATE_ID and
BLYNK_DEVICE_NAME"
15       #endif
16
```

```
17      BlynkTimer edgentTimer;
18
19      #include "BlynkState.h"
20      #include "ConfigStore.h"
21      #include "ResetButton.h"
22      #include "ConfigMode.h"
23      #include "Indicator.h"
24      #include "OTA.h"
25      #include "Console.h"
26
27      void manual_control();
28
29      inline
30      void BlynkState::set(State m) {
31        if (state != m && m < MODE_MAX_VALUE) {
32          DEBUG_PRINT(String(StateStr[state]) + " => " + StateStr[m]);
33          state = m;
34
35          // You can put your state handling here,
36          // i.e. implement custom indication
37        }
38      }
39
40      void printDeviceBanner()
41      {
42        Blynk.printBanner();
43        DEBUG_PRINT("--------------------------");
44        DEBUG_PRINT(String("Product:  ") + BLYNK_DEVICE_NAME);
45        DEBUG_PRINT(String("Firmware: ") + BLYNK_FIRMWARE_VERSION "
(build " __DATE__ " " __TIME__ ")");
46        if (configStore.getFlag(CONFIG_FLAG_VALID)) {
47          DEBUG_PRINT(String("Token:    ") +
48                      String(configStore.cloudToken).substring(0,4) +
49                      " - •••• - •••• - ••••");
50        }
51        DEBUG_PRINT(String("Device:   ") + BLYNK_INFO_DEVICE + " @ " +
ESP.getCpuFreqMHz() + "MHz");
52        DEBUG_PRINT(String("MAC:      ") + Wifi.macAddress());
53        DEBUG_PRINT(String("Flash:    ") + ESP.getFlashChipSize() / 1024
+ "K");
54        DEBUG_PRINT(String("ESP sdk:  ") + ESP.getSdkVersion());
55        DEBUG_PRINT(String("Chip rev: ") + ESP.getChipRevision());
56        DEBUG_PRINT(String("Free mem: ") + ESP.getFreeHeap());
57        DEBUG_PRINT("--------------------------");
58      }
59
60      void runBlynkWithChecks() {
61        Blynk.run();
62        if (BlynkState::get() == MODE_RUNNING) {
63          if (!Blynk.connected()) {
64            if (Wifi.status() == WL_CONNECTED) {
65              BlynkState::set(MODE_CONNECTING_CLOUD);
66            } else {
67              BlynkState::set(MODE_CONNECTING_NET);
```

```
68            }
69          }
70        }
71      }
72
73      class Edgent {
74
75      public:
76        void begin()
77        {
78          Wifi.persistent(false);
79          Wifi.enableSTA(true); // Needed to get MAC
80
81          indicator_init();
82          button_init();
83          config_init();
84          printDeviceBanner();
85          console_init();
86
87          if (configStore.getFlag(CONFIG_FLAG_VALID)) {
88            BlynkState::set(MODE_CONNECTING_NET);
89          } else if (config_load_blnkopt()) {
90            DEBUG_PRINT("Firmware is preprovisioned");
91            BlynkState::set(MODE_CONNECTING_NET);
92          } else {
93            BlynkState::set(MODE_WAIT_CONFIG);
94          }
95        }
96
97        void run() {
98          app_loop();
99          switch (BlynkState::get()) {
100         case MODE_WAIT_CONFIG:
101         case MODE_CONFIGURING:       enterConfigMode();     break;
102         case MODE_CONNECTING_NET:    enterConnectNet();     break;
103         case MODE_CONNECTING_CLOUD:  enterConnectCloud();   break;
104         case MODE_RUNNING:           runBlynkWithChecks();  break;
105         case MODE_OTA_UPGRADE:       enterOTA();            break;
106         case MODE_SWITCH_TO_STA:     enterSwitchToSTA();    break;
107         case MODE_RESET_CONFIG:      enterResetConfig();    break;
108         default:                     enterError();          break;
109         }
110       }
111
112     } BlynkEdgent;
113
114     void app_loop() {
115         edgentTimer.run();
116         edgentConsole.run();
117         manual_control();
118     }
```

Anexo H. Programa estación meteorológica[1]

```
1       //aquí se pega la información de las credenciales de la plantilla
creada en Blynk
2       #define BLYNK_TEMPLATE_ID "TMPLxxxxxx"
3       #define BLYNK_DEVICE_NAME "ESTACION METEOROLOGICA"
4
5       #define BLYNK_FIRMWARE_VERSION        "0.1.0"
6
7       #define BLYNK_PRINT Serial
8       //#define BLYNK_DEBUG
9
10      #define APP_DEBUG
11
12      // Uncomment your board, or configure a custom board in Settings.h
13      //#define USE_WROVER_BOARD
14      //#define USE_TTGO_T7
15      //#define USE_ESP32C3_DEV_MODULE
16      //#define USE_ESP32S2_DEV_KIT
17
18      #include "BlynkEdgent.h"
19      #include "DHT.h"
20      #include <Wire.h>
21      #include <BH1750.h>
22      #include <Adafruit_BMP085.h>
23
24
25      #define DHTPIN 19
26      #define DHTTYPE DHT11
27
28      DHT dht(DHTPIN, DHTTYPE);
29      BH1750 lightMeter(0x23);
30      Adafruit_BMP085 bmp;
31      BlynkTimer timer;
32
33      void setup() {
34        Serial.begin(115200);
35        dht.begin();
36        Wire.begin();
37        lightMeter.begin();
38        if (!bmp.begin(0x77)){
39        Serial.println("Could not find a valid BMP085 sensor!");
40        while (1) {}
41        }
42        delay(100);
43
44        timer.setInterval(60000L, UVLOOP);
45        timer.setInterval(60000L, DHTLOOP);
46        timer.setInterval(120000L, LIGHTLOOP);
47        timer.setInterval(30000L, PRESS);
48
```

[1] El desarrollo de los programas contiene líneas de código provenientes del trabajo colectivo en los foros oficiales de consulta en los sitios web: https://community.blynk.cc/; https://forum.arduino.cc/; https://github.com/

```
49        BlynkEdgent.begin();
50      }
51
52      void loop() {
53        UVLOOP();
54        DHTLOOP();
55        LIGHTLOOP();
56        PRESS();
57        VIENTO();
58        BlynkEdgent.run();
59      }
60
61      void UVLOOP() {
62        int uvLevel = analogRead(35);
63        int uvIntensity = map(uvLevel, 1160, 2150, 0, 15);
64        Blynk.virtualWrite(V0, uvIntensity);
65      }
66
67      void DHTLOOP() {
68        float h = dht.readHumidity();
69        float t = dht.readTemperature();
70        if (isnan(h) || isnan(t)){
71          Serial.println(F("Failed to read from DHT sensor!"));
72          return;
73        }
74        float hic = dht.computeHeatIndex(t, h, false);
75        Blynk.virtualWrite(V1, h);
76        Blynk.virtualWrite(V2, hic);
77      }
78
79      void LIGHTLOOP() {
80        float lux = lightMeter.readLightLevel();
81        Blynk.virtualWrite(V3, lux);
82      }
83
84      void PRESS() {
85          Blynk.virtualWrite(V4, bmp.readAltitude(101500));
86          Blynk.virtualWrite(V5, bmp.readPressure());
87      }
88
89      void VIENTO() {
90        int readWind = analogRead(34);
91        int vWind = map(readWind, 0, 300, 0, 100);
92        Blynk.virtualWrite(V6, vWind);
93      }
```

Anexo I. Personalización de biblioteca Blynk (Cabeceras ConfigMode.h y Settings.h)[1]

Se incluyen los cambios para que la tarjeta de desarrollo continue con la ejecución sin conexión a una red Wifi.

I.1 Modificación de la cabecera y continuar con la ejecución del programa sin conexión a red (ConfigMode.h)

```
#include <WifiClient.h>
#include <WebServer.h>
#include <DNSServer.h>
#include <Update.h>

#ifdef BLYNK_USE_SPIFFS
  #include "SPIFFS.h"
#else
  const char* config_form = R"html(
<!DOCTYPE HTML>
<html>
<head>
  <title>Wifi setup</title>
  <style>
  body {
    background-color: #fcfcfc;
    box-sizing: border-box;
  }
  body, input {
    font-family: Roboto, sans-serif;
    font-weight: 400;
    font-size: 16px;
  }
  .centered {
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);

    padding: 20px;
    background-color: #ccc;
    border-radius: 4px;
  }
  td { padding:0 0 0 5px; }
  label { white-space:nowrap; }
  input { width: 20em; }
  input[name="port"] { width: 5em; }
  input[type="submit"], img { margin: auto; display: block; width: 30%; }
  </style>
</head>
```

[1] El desarrollo de los programas contiene líneas de código provenientes del trabajo colectivo en los foros oficiales de consulta en los sitios web: https://community.blynk.cc/; https://forum.arduino.cc/; https://github.com/

```
41        <body>
42        <div class="centered">
43          <form method="get" action="config">
44            <table>
45            <tr><td><label for="ssid">Wifi SSID:</label></td>  <td><input
type="text" name="ssid" length=64 required="required"></td></tr>
46            <tr><td><label for="pass">Password:</label></td>   <td><input
type="text" name="pass" length=64></td></tr>
47            <tr><td><label for="blynk">Auth token:</label></td><td><input
type="text" name="blynk" placeholder="a0b1c2d..." pattern="[-_a-zA-Z0-
9]{32}" maxlength="32" required="required"></td></tr>
48            <tr><td><label for="host">Host:</label></td>       <td><input
type="text" name="host" value="blynk.cloud" length=64></td></tr>
49            <tr><td><label for="port_ssl">Port:</label></td>   <td><input
type="number" name="port_ssl" value="443" min="1" max="65535"></td></tr>
50            </table><br/>
51            <input type="submit" value="Apply">
52          </form>
53        </div>
54        </body>
55        </html>
56        )html";
57        #endif
58
59        WebServer server(80);
60        DNSServer dnsServer;
61        const byte DNS_PORT = 53;
62
63        static int connectNetRetries    = WIFI_CLOUD_MAX_RETRIES;
64        static int connectBlynkRetries  = WIFI_CLOUD_MAX_RETRIES;
65
66        static const char serverUpdateForm[] PROGMEM =
67          R"(<html><body>
68              <form method='POST' action='' enctype='multipart/form-data'>
69                <input type='file' name='update'>
70                <input type='submit' value='Update'>
71              </form>
72            </body></html>)";
73
74        void restartMCU() {
75          ESP.restart();
76          while(1) {};
77        }
78
79        void getWifiName(char* buff, size_t len, bool withPrefix = true) {
80          const uint64_t chipId = ESP.getEfuseMac();
81          uint32_t unique = 0;
82          for (int i=0; i<4; i++) {
83            unique = BlynkCRC32(&chipId, sizeof(chipId), unique);
84          }
85          unique &= 0xFFFFF;
86
87          String devPrefix = CONFIG_DEVICE_PREFIX;
```

```
88          String devName = String(BLYNK_DEVICE_NAME).substring(0, 31-7-
devPrefix.length());
89
90          if (withPrefix) {
91            snprintf(buff, len, "%s %s-%05X", devPrefix.c_str(),
devName.c_str(), unique);
92          } else {
93            snprintf(buff, len, "%s-%05X", devName.c_str(), unique);
94          }
95        }
96
97        void enterConfigMode()
98        {
99          char ssidBuff[64];
100         getWifiName(ssidBuff, sizeof(ssidBuff));
101
102         Wifi.mode(WIFI_OFF);
103         delay(100);
104         Wifi.mode(WIFI_AP);
105         delay(2000);
106         Wifi.softAPConfig(WIFI_AP_IP, WIFI_AP_IP, WIFI_AP_Subnet);
107         Wifi.softAP(ssidBuff);
108         delay(500);
109
110         IPAddress myIP = Wifi.softAPIP();
111         DEBUG_PRINT(String("AP SSID: ") + ssidBuff);
112         DEBUG_PRINT(String("AP IP:   ") + myIP[0] + "." + myIP[1] + "."
+ myIP[2] + "." + myIP[3]);
113
114         // Set up DNS Server
115         dnsServer.setTTL(300); // Time-to-live 300s
116         dnsServer.setErrorReplyCode(DNSReplyCode::ServerFailure); //
Return code for non-accessible domains
117       #ifdef WIFI_CAPTIVE_PORTAL_ENABLE
118         dnsServer.start(DNS_PORT, "*", Wifi.softAPIP()); // Point all to
our IP
119         server.onNotFound(handleRoot);
120       #else
121         dnsServer.start(DNS_PORT, CONFIG_AP_URL, Wifi.softAPIP());
122         DEBUG_PRINT(String("AP URL:  ") + CONFIG_AP_URL);
123       #endif
124
125         server.on("/update", HTTP_GET, []() {
126           server.sendHeader("Connection", "close");
127           server.send(200, "text/html", serverUpdateForm);
128         });
129         server.on("/update", HTTP_POST, []() {
130           server.sendHeader("Connection", "close");
131           if (!Update.hasError()) {
132             server.send(200, "text/plain", "OK");
133           } else {
134             server.send(500, "text/plain", "FAIL");
135           }
136           delay(1000);
```

```
137           restartMCU();
138         }, []() {
139           HTTPUpload& upload = server.upload();
140           if (upload.status == UPLOAD_FILE_START) {
141             DEBUG_PRINT(String("Update: ") + upload.filename);
142             //WifiUDP::stop();
143
144             if (!Update.begin(UPDATE_SIZE_UNKNOWN)) { //start with max
available size
145               Update.printError(BLYNK_PRINT);
146             }
147           } else if (upload.status == UPLOAD_FILE_WRITE) {
148             /* flashing firmware to ESP*/
149             if (Update.write(upload.buf, upload.currentSize) !=
upload.currentSize) {
150               Update.printError(BLYNK_PRINT);
151             }
152             BLYNK_PRINT.print(".");
153           } else if (upload.status == UPLOAD_FILE_END) {
154             BLYNK_PRINT.println();
155             DEBUG_PRINT("Finishing...");
156             if (Update.end(true)) { //true to set the size to the
current progress
157               DEBUG_PRINT("Update Success. Rebooting");
158             } else {
159               Update.printError(BLYNK_PRINT);
160             }
161           }
162         });
163
164         server.on("/config", []() {
165           DEBUG_PRINT("Applying configuration...");
166           String ssid = server.arg("ssid");
167           String ssidManual = server.arg("ssidManual");
168           String pass = server.arg("pass");
169           if (ssidManual != "") {
170             ssid = ssidManual;
171           }
172           String token = server.arg("blynk");
173           String host  = server.arg("host");
174           String port  = server.arg("port_ssl");
175
176           String ip   = server.arg("ip");
177           String mask = server.arg("mask");
178           String gw   = server.arg("gw");
179           String dns  = server.arg("dns");
180           String dns2 = server.arg("dns2");
181
182           bool forceSave  = server.arg("save").toInt();
183
184           String content;
185
186           DEBUG_PRINT(String("Wifi SSID: ") + ssid + " Pass: " + pass);
```

```
187         DEBUG_PRINT(String("Blynk cloud: ") + token + " @ " + host +
":" + port);
188
189         if (token.length() == 32 && ssid.length() > 0) {
190           configStore = configDefault;
191           CopyString(ssid, configStore.wifiSSID);
192           CopyString(pass, configStore.wifiPass);
193           CopyString(token, configStore.cloudToken);
194           if (host.length()) {
195             CopyString(host,  configStore.cloudHost);
196           }
197           if (port.length()) {
198             configStore.cloudPort = port.toInt();
199           }
200
201           IPAddress addr;
202
203           if (ip.length() && addr.fromString(ip)) {
204             configStore.staticIP = addr;
205             configStore.setFlag(CONFIG_FLAG_STATIC_IP, true);
206           } else {
207             configStore.setFlag(CONFIG_FLAG_STATIC_IP, false);
208           }
209           if (mask.length() && addr.fromString(mask)) {
210             configStore.staticMask = addr;
211           }
212           if (gw.length() && addr.fromString(gw)) {
213             configStore.staticGW = addr;
214           }
215           if (dns.length() && addr.fromString(dns)) {
216             configStore.staticDNS = addr;
217           }
218           if (dns2.length() && addr.fromString(dns2)) {
219             configStore.staticDNS2 = addr;
220           }
221
222           if (forceSave) {
223             configStore.setFlag(CONFIG_FLAG_VALID, true);
224             config_save();
225
226             content = R"json({"status":"ok","msg":"Configuration
saved"})json";
227           } else {
228             content = R"json({"status":"ok","msg":"Trying to
connect..."})json";
229           }
230           server.send(200, "application/json", content);
231
232           connectNetRetries = connectBlynkRetries = 1;
233           BlynkState::set(MODE_SWITCH_TO_STA);
234         } else {
235           DEBUG_PRINT("Configuration invalid");
236           content = R"json({"status":"error","msg":"Configuration
invalid"})json";
```

```
237           server.send(500, "application/json", content);
238         }
239       });
240       server.on("/board_info.json", []() {
241         // Configuring starts with board info request (may impact
indication)
242         BlynkState::set(MODE_CONFIGURING);
243 
244         DEBUG_PRINT("Sending board info...");
245         const char* tmpl = BLYNK_TEMPLATE_ID;
246         char ssidBuff[64];
247         getWifiName(ssidBuff, sizeof(ssidBuff));
248         char buff[512];
249         snprintf(buff, sizeof(buff),
250 
R"json({"board":"%s","tmpl_id":"%s","fw_type":"%s","fw_ver":"%s","ssid":"%
s","bssid":"%s","mac":"%s","last_error":%d,"wifi_scan":true,"static_ip":tr
ue})json",
251           BLYNK_DEVICE_NAME,
252           tmpl ? tmpl : "Unknown",
253           BLYNK_FIRMWARE_TYPE,
254           BLYNK_FIRMWARE_VERSION,
255           ssidBuff,
256           Wifi.softAPmacAddress().c_str(),
257           Wifi.macAddress().c_str(),
258           configStore.last_error
259         );
260         server.send(200, "application/json", buff);
261       });
262       server.on("/wifi_scan.json", []() {
263         DEBUG_PRINT("Scanning networks...");
264         int wifi_nets = Wifi.scanNetworks(true, true);
265         const uint32_t t = millis();
266         while (wifi_nets < 0 &&
267                millis() - t < 20000)
268         {
269           delay(20);
270           wifi_nets = Wifi.scanComplete();
271         }
272         DEBUG_PRINT(String("Found networks: ") + wifi_nets);
273 
274         if (wifi_nets > 0) {
275           // Sort networks
276           int indices[wifi_nets];
277           for (int i = 0; i < wifi_nets; i++) {
278             indices[i] = i;
279           }
280           for (int i = 0; i < wifi_nets; i++) {
281             for (int j = i + 1; j < wifi_nets; j++) {
282               if (Wifi.RSSI(indices[j]) > Wifi.RSSI(indices[i])) {
283                 std::swap(indices[i], indices[j]);
284               }
285             }
286           }
```

```
287
288           wifi_nets = BlynkMin(15, wifi_nets); // Show top 15 networks
289
290           // TODO: skip empty names
291           String result = "[\n";
292
293           char buff[256];
294           for (int i = 0; i < wifi_nets; i++){
295             int id = indices[i];
296
297             const char* sec;
298             switch (Wifi.encryptionType(id)) {
299             case WIFI_AUTH_WEP:           sec = "WEP"; break;
300             case WIFI_AUTH_WPA_PSK:       sec = "WPA/PSK"; break;
301             case WIFI_AUTH_WPA2_PSK:      sec = "WPA2/PSK"; break;
302             case WIFI_AUTH_WPA_WPA2_PSK:  sec = "WPA/WPA2/PSK"; break;
303             case WIFI_AUTH_OPEN:          sec = "OPEN"; break;
304             default:                      sec = "unknown"; break;
305             }
306
307             snprintf(buff, sizeof(buff),
308               R"json(
{"ssid":"%s","bssid":"%s","rssi":%i,"sec":"%s","ch":%i})json",
309               Wifi.SSID(id).c_str(),
310               Wifi.BSSIDstr(id).c_str(),
311               Wifi.RSSI(id),
312               sec,
313               Wifi.channel(id)
314             );
315
316             result += buff;
317             if (i != wifi_nets-1) result += ",\n";
318           }
319           server.send(200, "application/json", result + "\n]");
320         } else {
321           server.send(200, "application/json", "[]");
322         }
323       });
324       server.on("/reset", []() {
325         BlynkState::set(MODE_RESET_CONFIG);
326         server.send(200, "application/json",
R"json({"status":"ok","msg":"Configuration reset"})json");
327       });
328       server.on("/reboot", []() {
329         restartMCU();
330       });
331
332     #ifdef BLYNK_USE_SPIFFS
333       if (SPIFFS.begin()) {
334         server.serveStatic("/img/favicon.png", SPIFFS,
"/img/favicon.png");
335         server.serveStatic("/img/logo.png", SPIFFS, "/img/logo.png");
336         server.serveStatic("/", SPIFFS, "/index.html");
337       } else {
```

```
338           DEBUG_PRINT("Webpage: No SPIFFS");
339         }
340       #endif
341
342         server.begin();
343
344         while (BlynkState::is(MODE_WAIT_CONFIG) ||
BlynkState::is(MODE_CONFIGURING)) {
345           delay(10);
346           dnsServer.processNextRequest();
347           server.handleClient();
348           app_loop();
349           if (BlynkState::is(MODE_CONFIGURING) &&
Wifi.softAPgetStationNum() == 0) {
350             BlynkState::set(MODE_WAIT_CONFIG);
351           }
352         }
353
354         server.stop();
355
356       #ifdef BLYNK_USE_SPIFFS
357         SPIFFS.end();
358       #endif
359       }
360
361       void enterConnectNet() {
362         BlynkState::set(MODE_CONNECTING_NET);
363         DEBUG_PRINT(String("Connecting to Wifi: ") +
configStore.wifiSSID);
364
365         char ssidBuff[64];
366         getWifiName(ssidBuff, sizeof(ssidBuff));
367         String hostname(ssidBuff);
368         hostname.replace(" ", "-");
369         Wifi.setHostname(hostname.c_str());
370
371         if (configStore.getFlag(CONFIG_FLAG_STATIC_IP)) {
372           if (!Wifi.config(configStore.staticIP,
373                            configStore.staticGW,
374                            configStore.staticMask,
375                            configStore.staticDNS,
376                            configStore.staticDNS2)
377           ) {
378             DEBUG_PRINT("Failed to configure Static IP");
379             config_set_last_error(BLYNK_PROV_ERR_CONFIG);
380             BlynkState::set(MODE_ERROR);
381             return;
382           }
383         }
384
385         Wifi.begin(configStore.wifiSSID, configStore.wifiPass);
386
387         unsigned long timeoutMs = millis() + WIFI_NET_CONNECT_TIMEOUT;
```

```
388       while ((timeoutMs > millis()) && (Wifi.status() !=
WL_CONNECTED))
389       {
390         delay(10);
391         app_loop();
392
393         if (!BlynkState::is(MODE_CONNECTING_NET)) {
394           Wifi.disconnect();
395           return;
396         }
397       }
398
399       if (Wifi.status() == WL_CONNECTED) {
400         IPAddress localip = Wifi.localIP();
401         if (configStore.getFlag(CONFIG_FLAG_STATIC_IP)) {
402           BLYNK_LOG_IP("Using Static IP: ", localip);
403         } else {
404           BLYNK_LOG_IP("Using Dynamic IP: ", localip);
405         }
406
407         connectNetRetries = WIFI_CLOUD_MAX_RETRIES;
408         BlynkState::set(MODE_CONNECTING_CLOUD);
409       } else if (--connectNetRetries <= 0) {
410         config_set_last_error(BLYNK_PROV_ERR_NETWORK);
411         BlynkState::set(MODE_ERROR);
412       }
413     }
414
415     void enterConnectCloud() {
416       BlynkState::set(MODE_CONNECTING_CLOUD);
417
418       Blynk.config(configStore.cloudToken, configStore.cloudHost,
configStore.cloudPort);
419       Blynk.connect(0);
420
421       unsigned long timeoutMs = millis() + WIFI_CLOUD_CONNECT_TIMEOUT;
422       while ((timeoutMs > millis()) &&
423              (Wifi.status() == WL_CONNECTED) &&
424              (!Blynk.isTokenInvalid()) &&
425              (Blynk.connected() == false))
426       {
427         delay(10);
428         Blynk.run();
429         app_loop();
430         if (!BlynkState::is(MODE_CONNECTING_CLOUD)) {
431           Blynk.disconnect();
432           return;
433         }
434       }
435
436       if (millis() > timeoutMs) {
437         DEBUG_PRINT("Timeout");
438       }
439
```

```
440       if (Blynk.isTokenInvalid()) {
441         config_set_last_error(BLYNK_PROV_ERR_TOKEN);
442         BlynkState::set(MODE_WAIT_CONFIG); // TODO: retry after
timeout
443       } else if (Wifi.status() != WL_CONNECTED) {
444         BlynkState::set(MODE_CONNECTING_NET);
445       } else if (Blynk.connected()) {
446         BlynkState::set(MODE_RUNNING);
447         connectBlynkRetries = WIFI_CLOUD_MAX_RETRIES;
448
449         if (!configStore.getFlag(CONFIG_FLAG_VALID)) {
450           configStore.last_error = BLYNK_PROV_ERR_NONE;
451           configStore.setFlag(CONFIG_FLAG_VALID, true);
452           config_save();
453         }
454       } else if (--connectBlynkRetries <= 0) {
455         config_set_last_error(BLYNK_PROV_ERR_CLOUD);
456         BlynkState::set(MODE_ERROR);
457       }
458     }
459
460     void enterSwitchToSTA() {
461       BlynkState::set(MODE_SWITCH_TO_STA);
462
463       DEBUG_PRINT("Switching to STA...");
464
465       delay(1000);
466       Wifi.mode(WIFI_OFF);
467       delay(100);
468       Wifi.mode(WIFI_STA);
469
470       BlynkState::set(MODE_CONNECTING_NET);
471     }
472
473     void enterError() {
474       BlynkState::set(MODE_ERROR);
475
476       unsigned long timeoutMs = millis() + 10000;
477       while (timeoutMs > millis() || g_buttonPressed)
478       {
479         delay(10);
480         app_loop();
481         if (!BlynkState::is(MODE_ERROR)) {
482           return;
483         }
484       }
485       BlynkState::set(MODE_CONNECTING_NET);
486       /*DEBUG_PRINT("Restarting after error.");
487       delay(10);
488
489       restartMCU();*/
490     }
```

I.2 Modificación de la cabecera reducción del tiempo e intentos de conexión a una red y configuración de una tarjeta de desarrollo NodeMCU-32S (Settings.h)

```
1       /*
2        * Board configuration (see examples below).
3        */
4
5       #if defined(USE_WROVER_BOARD)
6
7         #define BOARD_BUTTON_PIN            15
8         #define BOARD_BUTTON_ACTIVE_LOW     true
9
10        #define BOARD_LED_PIN_R             0
11        #define BOARD_LED_PIN_G             2
12        #define BOARD_LED_PIN_B             4
13        #define BOARD_LED_INVERSE           false
14        #define BOARD_LED_BRIGHTNESS        128
15
16      #elif defined(USE_TTGO_T7)
17
18        // This board does not have a built-in button
19        // Connect a button to gpio0 <> GND
20        #define BOARD_BUTTON_PIN            0
21        #define BOARD_BUTTON_ACTIVE_LOW     true
22
23        #define BOARD_LED_PIN               19
24        #define BOARD_LED_INVERSE           false
25        #define BOARD_LED_BRIGHTNESS        64
26
27      #elif defined(USE_ESP32C3_DEV_MODULE)
28
29        #define BOARD_BUTTON_PIN            9
30        #define BOARD_BUTTON_ACTIVE_LOW     true
31
32        #define BOARD_LED_PIN_WS2812        8
33        #define BOARD_LED_INVERSE           false
34        #define BOARD_LED_BRIGHTNESS        32
35
36      #elif defined(USE_ESP32S2_DEV_KIT)
37
38        #define BOARD_BUTTON_PIN            0
39        #define BOARD_BUTTON_ACTIVE_LOW     true
40
41        #define BOARD_LED_PIN               19
42        #define BOARD_LED_INVERSE           false
43        #define BOARD_LED_BRIGHTNESS        128
44
45      #else
46
47        #warning "Custom board configuration is used"
48
49        #define BOARD_BUTTON_PIN            15
50        #define BOARD_BUTTON_ACTIVE_LOW     true
51
```

```
52          #define BOARD_LED_PIN               2
53          //#define BOARD_LED_PIN_R           15
54          //#define BOARD_LED_PIN_G           12
55          //#define BOARD_LED_PIN_B           13
56          //#define BOARD_LED_PIN_WS2812      4
57          #define BOARD_LED_INVERSE           false
58          #define BOARD_LED_BRIGHTNESS        255
59
60      #endif
61
62
63      /*
64       * Advanced options
65       */
66
67      #define BUTTON_HOLD_TIME_INDICATION   3000
68      #define BUTTON_HOLD_TIME_ACTION       10000
69      #define BUTTON_PRESS_TIME_ACTION      50
70
71      #define BOARD_PWM_MAX                 1023
72
73      #define BOARD_LEDC_CHANNEL_1          1
74      #define BOARD_LEDC_CHANNEL_2          2
75      #define BOARD_LEDC_CHANNEL_3          3
76      #define BOARD_LEDC_TIMER_BITS         10
77      #define BOARD_LEDC_BASE_FREQ          12000
78
79      #if !defined(CONFIG_DEVICE_PREFIX)
80      #define CONFIG_DEVICE_PREFIX          "Blynk"
81      #endif
82      #if !defined(CONFIG_AP_URL)
83      #define CONFIG_AP_URL                 "blynk.setup"
84      #endif
85      #if !defined(CONFIG_DEFAULT_SERVER)
86      #define CONFIG_DEFAULT_SERVER         "blynk.cloud"
87      #endif
88      #if !defined(CONFIG_DEFAULT_PORT)
89      #define CONFIG_DEFAULT_PORT           443
90      #endif
91
92      #define WIFI_CLOUD_MAX_RETRIES        10
93      #define WIFI_NET_CONNECT_TIMEOUT      3000
94      #define WIFI_CLOUD_CONNECT_TIMEOUT    3000
95      #define WIFI_AP_IP                    IPAddress(192, 168, 4, 1)
96      #define WIFI_AP_Subnet                IPAddress(255, 255, 255, 0)
97      //#define WIFI_CAPTIVE_PORTAL_ENABLE
98
99      //#define USE_TICKER
100     //#define USE_TIMER_ONE
101     //#define USE_TIMER_THREE
102     //#define USE_TIMER_FIVE
103     #define USE_PTHREAD
104
105     #define BLYNK_NO_DEFAULT_BANNER
```

```
106
107     #if defined(APP_DEBUG)
108       #define DEBUG_PRINT(...)  BLYNK_LOG1(__VA_ARGS__)
109       #define DEBUG_PRINTF(...) BLYNK_LOG(__VA_ARGS__)
110     #else
111       #define DEBUG_PRINT(...)
112       #define DEBUG_PRINTF(...)
113     #endif
```