



**Universidad Nacional Autónoma de
México**

Facultad de Estudios Superiores Aragón

“Desarrollo de soluciones domóticas bajo un enfoque sustentable”

T E S I S

Para obtener el título de:

INGENIERO ELÉCTRICO ELECTRÓNICO

Presenta:

Erick Ricardo Ramírez Cuevas

Director de Tesis:

Dr. Ismael Díaz Rangel

Nezahualcóyotl, Estado de México, 2023





Universidad Nacional
Autónoma de México



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Agradecimientos

Quiero aprovechar este espacio para expresar mi sincero agradecimiento a las personas que han sido fundamentales en el desarrollo y conclusión de esta etapa de mi vida. Sus contribuciones y apoyo han sido invaluable y merecen mi reconocimiento especial.

A mis queridos padres Ricardo Enrique Ramírez Herrera y Lorena Cuevas Ávila, quienes han sido un ejemplo que seguir y han brindado apoyo incondicional a lo largo de mi vida. Su amor, dedicación y aliento constante han sido la base sobre la cual he podido construir mis logros académicos. Gracias por creer en mí y por ser mi mayor fuente de inspiración.

A mi hermano Maximiliano Ramírez Cuevas, quien ha sido una luz en los momentos oscuros. Su presencia y ánimo han sido un constante recordatorio de la importancia de mantener la determinación y el optimismo en los desafíos que se presentan en el camino. Agradezco su compañía, consejos y el amor incondicional que siempre me ha brindado.

A la Universidad Nacional Autónoma de México, Facultad de Estudios Superiores Aragón, por brindarme los recursos necesarios y el conocimiento invaluable para llevar a cabo este trabajo. La excelencia académica y la calidad de la educación que he recibido han sido fundamentales en mi formación como estudiante y como persona.

Al Dr. Ismael Díaz Rangel, su experiencia, sabiduría y dedicación han sido fundamentales para orientarme en la dirección correcta, proporcionarme retroalimentación constructiva y desafiarme a alcanzar mis metas académicas. Agradezco su paciencia, disponibilidad y por compartir su conocimiento conmigo.

Una vez más, expreso mi profundo agradecimiento a mis padres, mi hermano, la UNAM y al Dr. Ismael, por su valioso apoyo, motivación y contribuciones a este trabajo. Sin su presencia y ayuda, no hubiera sido posible lograrlo. Estoy sumamente agradecido y honrado por contar con su respaldo en esta importante etapa de mi vida académica.

ÍNDICE

Agradecimientos.....	i
ÍNDICE.....	ii
Introducción.....	1
Objetivo general	2
Objetivos particulares	2
Actividades	2
Justificación	2
Descripción del capitulado	2
Capítulo 2. Marco teórico.....	4
2.1 Sustentabilidad.....	4
2.2 La domótica	5
2.2.1 Objetivos de la domótica	5
2.2.2 Elementos de un sistema domótico	6
2.3 Generación de energía con paneles solares	7
2.3.1 Célula solar	7
2.3.2 Panel Solar.....	9
2.3.3 Regulador.....	10
2.3.4 Baterías	10
2.3.5 Inversor	12
2.4 Arduino IDE	14
2.5 ESP32	14
2.6 Sensores	15
2.6.1 Sensor resistivo de humedad de sustrato	15
2.6.2 Sensor de humedad de sustrato capacitivo v1.2.	16
2.6.3 Sensor de gas MQ.....	16
2.6.4 Módulo <i>RFID</i>	19
2.7 IoT	20
2.7.1 Blynk	20
2.7.2 Cayenne	20
2.7.3 ThingSpeak.....	20
2.8 Medidas de tendencia central	20
2.8.1 Promedio o media.....	21

2.8.2 Mediana	21
2.8.3 Moda	21
2.9 Señales	21
2.9.1 Señal analógica	22
2.9.2 Señal digital	22
2.9.3 Digitalización de una señal analógica	22
2.10 Etapa de Potencia.....	23
2.10.1 Relevador.....	23
2.10.2 Optoacoplador	24
2.10.3 Capacitor.....	24
2.11 Radiación solar	25
2.11.1 Unidades de radiación solar.....	26
Capítulo 3. Desarrollo experimental.....	28
3.1 Diagrama de bloques.....	28
3.2 Desarrollo de circuitos de control	30
3.2.1 Módulo riego automático de sustrato	30
3.2.2 Módulo detección de gas licuado de petróleo (GLP) y cierre de válvula....	31
3.2.3 Módulo apertura de puerta mediante RFID	31
3.2.4 Etapa de potencia.....	32
3.3 Algoritmos, diagramas de flujo y codificación.....	33
3.3.1 Riego automático de sustrato.....	33
3.3.2 Detección de gas licuado de petróleo (GLP) y cierre de válvula	39
3.3.3 Apertura de puerta usando módulo RFID.....	44
3.4 Aplicación en <i>Blynk</i>	50
3.4.1 Riego automático de sustrato.....	50
3.4.2 Detección de gas licuado de petróleo (GLP) y cierre de válvula	51
3.4.3 Apertura de puerta mediante RFID	52
3.5 Fabricación de PCB.....	53
3.6 Diseño de gabinete.....	54
3.7 Sistema de captación de energía solar como fuente de alimentación.....	54
Capítulo 4. Pruebas y resultados.....	56
4.1 Pruebas fuera de línea.....	56
4.1.1 Riego automático de sustrato.....	56
4.1.2 Detección de gas licuado de petróleo y cierre de válvula.....	59

4.1.3 Apertura de puerta mediante <i>RFID</i>	62
4.2 Pruebas en línea	68
4.2.1 <i>Blynk</i>	68
4.2.2 Uso de <i>widgets</i> en la plataforma de <i>Blynk</i>	74
4.3 Elección de panel solar	75
Conclusiones.....	81
Trabajo a futuro	83
Referencias	84
Fuentes de imágenes.....	88
Anexo.....	90

Introducción

En la actualidad, la humanidad se enfrenta a desafíos significativos en términos de sustentabilidad y eficiencia energética. El agotamiento de los recursos naturales, la creciente demanda de energía y el impacto ambiental causado por las fuentes de energía convencionales obligan a replantear el enfoque hacia la generación y el consumo de energía [1]. Es en este contexto que las energías renovables y la automatización del hogar emergen como soluciones prometedoras.

El cambio hacia las energías renovables es fundamental debido a múltiples razones. En primer lugar, estas fuentes de energía, como la solar, eólica, hidroeléctrica y geotérmica, son inagotables, a diferencia de los combustibles fósiles que tienen una reserva limitada. Al aprovechar las energías renovables, se puede reducir la dependencia de los recursos no renovables y mitigar los efectos del cambio climático [2].

Además, las energías renovables presentan beneficios ambientales significativos. Al no generar emisiones de gases de efecto invernadero y otros contaminantes, contribuyen a la mejora de la calidad del aire y reducen el impacto negativo en la salud humana [3].

Desde el punto de vista económico, las energías renovables están experimentando una rápida reducción de costos y se están convirtiendo en una opción cada vez más asequible. Esto crea oportunidades para el desarrollo económico, la creación de empleo y la atracción de inversiones en la industria de energías limpias [4].

La automatización del hogar, también conocida como domótica, se refiere a la integración de la tecnología en la gestión de diversas funciones y sistemas en el entorno residencial. Esta tendencia está ganando impulso debido a los beneficios que ofrece tanto en términos de comodidad como de eficiencia energética, entre otros...

La domótica permite controlar y programar de forma centralizada diversos dispositivos y sistemas en el hogar, como la iluminación, la climatización, los electrodomésticos y la seguridad. Esto no solo brinda una mayor comodidad y conveniencia al permitirnos controlar estos elementos a distancia o mediante comandos de voz, sino que también puede ayudar a optimizar el consumo energético [5].

La automatización del hogar permite ajustar automáticamente la iluminación y la temperatura según las necesidades y la presencia de las personas en la casa, lo que puede resultar en un uso más eficiente de la energía. Además, mediante el monitoreo y análisis de datos, se pueden identificar patrones de consumo y hacer ajustes inteligentes para ahorrar energía y reducir costos.

Además de los beneficios energéticos, la domótica también puede mejorar la seguridad y la gestión del hogar. Con sistemas de seguridad integrados, detección de fugas de gas, y la capacidad de recibir notificaciones en tiempo real, se puede tener un mayor control y tranquilidad en cuanto a la protección del hogar y la seguridad de la familia [6].

La transición hacia las energías renovables y la automatización del hogar representan oportunidades valiosas para construir un futuro más sustentable, eficiente y cómodo. Al integrar estas tecnologías, se puede reducir la huella ecológica, ahorrar energía, proteger el medio ambiente y mejorar la calidad de vida en el hogar.

Objetivo general

Desarrollo de un conjunto de soluciones para monitoreo y automatización en el hogar que sean energizadas por un sistema de captación de radiación solar.

Objetivos particulares

- Implementar un sistema de alimentación de energía eléctrica utilizando paneles solares.
- Monitorear y regar automáticamente sustratos.
- Monitorear y controlar fugas de gas.
- Incluir métodos de control de acceso mediante RFID.
- Implementar una aplicación remota para el monitoreo y configuración de las implementaciones propuestas.

Actividades

- Investigar qué es la sustentabilidad.
- Revisar sistemas de alimentación eléctrica basados en paneles solares.
- Estudiar qué es la domótica.
- Identificar sensores de humedad en sustratos.
- Conocer sensores de gas.
- Examinar métodos de control de accesos.
- Revisar etapas de potencia.
- Revisar plataformas IoT.
- Crear tarjetas PCB.
- Estudiar la radiación solar.

Justificación

Ante la creciente adopción de la domótica en hogares, se decide desarrollar soluciones domóticas aprovechando el reciente auge del internet de las cosas, no sin olvidar nuestra obligación con el planeta mostrando respeto al medio ambiente, es importante apostar por métodos de obtención más limpios de energía eléctrica, por lo que es conveniente utilizar como fuente de alimentación un sistema de captación de energía solar capaz de alimentar a las soluciones domóticas, y tener así las ventajas de este tipo de sistemas como lo es la comodidad, seguridad, accesibilidad, gestión de consumos, y ello sin deteriorar al medio ambiente.

Descripción del capitulado

Capítulo 1 Introducción. Se detallan los objetivos generales del proyecto, así como los objetivos particulares, la justificación del trabajo donde se dice porque se realizó este trabajo y con qué fin.

Capítulo 2 Marco Teórico. Se muestra un panorama de conocimientos general para el entendimiento del trabajo, definición de sustentabilidad, sistema de captación de energía eléctrica mediante paneles solares, sensores, y plataformas *IoT*.

Capítulo 3 Desarrollo Experimental. En este capítulo se muestra la metodología que se siguió para desarrollar los módulos propuestos, detallando el procedimiento de creación como: diagrama de bloques, desarrollo de circuitos de control, algoritmos, diagramas de flujo y codificación; e implementación con la plataforma *IoT, Blynk*.

Capítulo 4 Pruebas y Resultados. En este capítulo se muestran los prototipos de los módulos propuestos, la transición del modo offline al modo online a través de la plataforma de *Blynk*, uso de *Blynk*, parámetros a tomar en cuenta sobre la elección del panel solar, ubicación y autonomía del sistema.

Se continúa proporcionando las conclusiones y las mejoras a realizar como trabajo a futuro; por último, se indican las referencias tomadas.

Capítulo 2. Marco teórico

2.1 Sustentabilidad

“Un hito fundamental en la historia del ambientalismo fue la Conferencia de las Naciones Unidas sobre Ambiente Humano que se dio lugar en Estocolmo, Suecia en el año 1972. Lo más significativo de esta conferencia fue el hecho de que se sembraron las semillas de aquello que más tarde se reconocería como **sustentabilidad**. Durante esa conferencia no sólo se habló de la protección del medioambiente sino de algo mucho más amplio: la búsqueda de relaciones comunes entre aspectos ambientales y temas económicos relacionados con el capital, el crecimiento y el empleo. Uno de los tantos resultados de esta conferencia fue el desarrollo del Programa Ambiental de las Naciones Unidas (UNEP, United Nations Environmental Programme) donde se estableció una misión que luego se convirtió en una definición: “proveer liderazgo y compromiso mutuo en el cuidado del medioambiente inspirando, informando y posibilitando a las naciones y las personas el mejoramiento de su calidad de vida sin comprometer las necesidades de las generaciones futuras.”

En el año 1983 las Naciones Unidas crean la Comisión Mundial de Ambiente y Desarrollo (WCED, World Comisión of Environment and Development) presidida por Gro Harlem Brundtland, primer ministro de Noruega en aquel momento. Uno de los resultados más significativos que salieron de los informes emitidos por esta comisión fue la de identificar por primera vez la importancia de evaluar cualquier acción o iniciativa desde tres enfoques:

1. Económico,
2. Ambiental,
3. Social.

Definiendo así “El desarrollo sustentable hace referencia a la capacidad que haya desarrollado el sistema humano para satisfacer las necesidades de las generaciones actuales sin comprometer los recursos y oportunidades para el crecimiento y desarrollo de las generaciones futuras.”

Además, Suecia, uno de los líderes actuales en sustentabilidad tiene una definición más holística y define una **sociedad sustentable** como:

“una sociedad en la cual el desarrollo económico, el bienestar social y la integración están unidos con un medioambiente de calidad. Esta sociedad tiene la capacidad de satisfacer sus necesidades actuales sin perjudicar la habilidad de que las generaciones futuras puedan satisfacer las suyas”.

Reinterpretando más a detalle esta definición, desde el punto de vista de la prosperidad económica, queda expresado de la siguiente manera:

“Sustentabilidad es la habilidad de lograr una prosperidad económica sostenida en el tiempo protegiendo al mismo tiempo los sistemas naturales del planeta y proveyendo una alta calidad de vida para las personas.”

Más tarde en el año 1992 se celebra en Río de Janeiro el Earth Summit donde se consolida la acción de las Naciones Unidas en relación con los conceptos relacionados con el

medioambiente y el desarrollo sustentable. De dicha conferencia se acuerdan **27 principios** relacionados con la Sustentabilidad que se materializan en un programa mundial conocido como **Agenda 21**. Luego de estas acciones concretas comenzó a explotar una conciencia global acerca de la importancia de esta temática y así se crearon decenas de consejos consultivos, organismos, asociaciones e investigaciones relacionadas con la sustentabilidad.

“Un proceso es sostenible cuando ha desarrollado la capacidad para producir indefinidamente a un ritmo en el cual no agota los recursos que utiliza y que necesita para funcionar y no produce más contaminantes de los que puede absorber su entorno.” [1]

2.2 La domótica

“El término “domótica” proviene del latín *domus* (casa) y del griego *αὐτόματoς* (automática). El vocablo “domótica” se definió en 1988 por parte de la enciclopedia Larousse, como “el concepto de vivienda que integra todos los automatismos en materia de seguridad, gestión de la energía, comunicaciones, etc.”

Hasta hoy se conocen múltiples definiciones de domótica, de las que cabe destacar las siguientes:

- La Asociación Española de Domótica (CEDOM-2004) define la domótica como “el conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía, además a de aportar seguridad, confort y comunicación entre el usuario y el sistema”.
- Según el diccionario de la Real Academia Española, define a la domótica como “conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda”
- Para Huidobro J.M. y Millán R. (2004), la vivienda domótica es “aquella que integra un conjunto de automatismos en materia de electricidad, electrónica, robótica, informática y telecomunicaciones, con el objetivo de asegurar al usuario un aumento del confort, la seguridad, el ahorro energético, las facilidades de comunicación y las posibilidades de entretenimiento.” [7]

2.2.1 Objetivos de la domótica

Suelen enmarcarse en las siguientes áreas funcionales:

Gestión de la energía: Las aplicaciones dentro de este grupo funcional se orientan básicamente a racionalizar los distintos consumos energéticos domésticos (electricidad, gas natural, agua, etc.) en función de diversos criterios (ocupación de la vivienda, tarifas energéticas existentes para el sector doméstico, nivel de potencia eléctrica contratada, etc.) [8].

Algunos ejemplos son:

- Programación de climatización.
- Programación de equipos domésticos.
- Racionalización de cargas eléctricas.
- Implementación de panel solar con seguidor.

Confort: Su finalidad es la simplificación de algunas tareas en el hogar o incrementar las posibilidades de control, creando nuevos hábitos o modelos de uso para el usuario, destinados siempre a mejorar el confort. Algunos ejemplos son:

- Apagado de todas las luces de la vivienda.
- Regulación de la iluminación según el nivel de luminosidad ambiente.
- Automatización de la iluminación.
- Control de luces por mando a distancia.
- Control de equipos e instalaciones por mando a distancia.

Seguridad: Las aplicaciones de seguridad más habituales de los sistemas domóticos contemplan tanto la protección de las personas (lo que se denomina *seguridad personal*) como la de los bienes (lo que se denomina *seguridad patrimonial*). Algunos ejemplos son:

- Detección de intrusión.
- Simulación de presencia.
- Detección de incendios.
- Detección de fugas de gas.
- Detección de escapes de agua.

Comunicaciones: Las aplicaciones de comunicaciones contemplan el intercambio de información tanto entre personas, como con equipos domésticos, ya sea dentro de la propia vivienda como desde ésta al exterior.

Algunos ejemplos que encontramos son:

- Control remoto de equipos e instalaciones.
- Transmisión de alarmas.

2.2.2 Elementos de un sistema domótico

Los elementos básicos e imprescindibles que conforman una instalación domótica son los siguientes y pueden ser apreciados en la Figura 2-1 [9].

- Central de gestión.
- Sensores.
- Actuadores.
- Soportes de comunicación.
- Aparatos terminales.

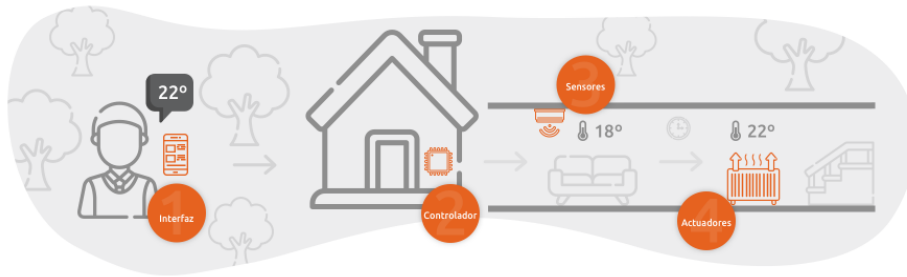


Figura 2-1. Elementos de un sistema domótico.

2.3 Generación de energía con paneles solares

La energía solar fotovoltaica es una tecnología que genera corriente continua (potencia medida en W o KW) por medio de semiconductores cuando éstos son iluminados por un haz de fotones. Mientras la luz incide sobre una célula solar, que es el nombre dado al elemento fotovoltaico individual, se genera potencia eléctrica; cuando la luz se extingue, la electricidad desaparece [10].

De manera general, una instalación solar fotovoltaica se ajusta a la Figura 2-2 [11].

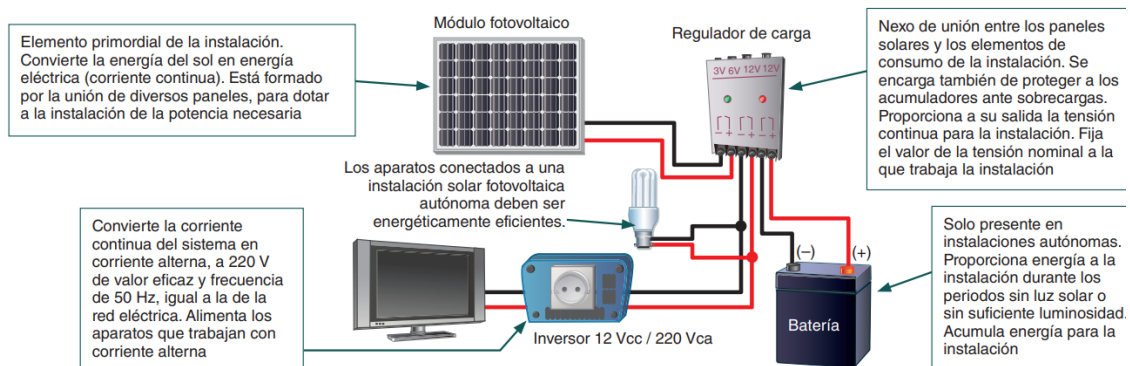


Figura 2-2. Elementos de una instalación solar fotovoltaica.

2.3.1 Célula solar

El elemento principal de cualquier instalación de energía solar es el generador, que recibe el nombre de **célula solar**. Se caracteriza por convertir directamente en electricidad los fotones provenientes de la luz del sol. Su funcionamiento se basa en el **efecto fotovoltaico**.

Una célula solar se comporta como un diodo: la parte expuesta a la radiación solar es la N, y la parte situada en la zona de oscuridad, la P. Las terminales de conexión de la célula se hallan sobre cada una de estas partes del diodo: la cara correspondiente a la zona P se encuentra metalizada por completo (no tiene que recibir luz), mientras que en la zona N el metalizado tiene forma de peine, a fin de que la radiación solar llegue al semiconductor [11].

En la Figura 2-3 puede apreciarse la estructura de una célula solar.

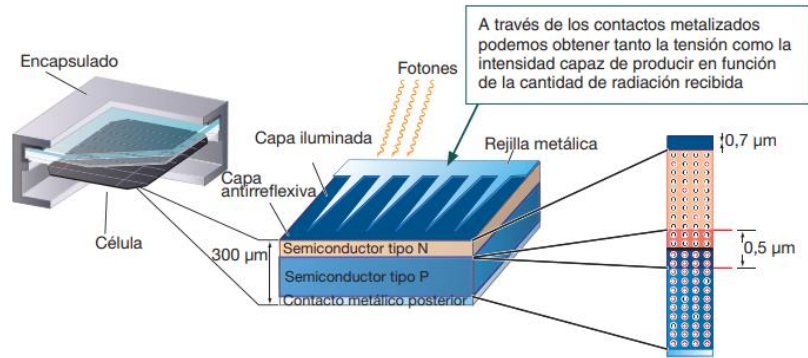


Figura 2-3. Estructura de la célula solar.

2.3.1.1 Parámetros fundamentales de la célula solar

- **Corriente de iluminación (I_L):** la corriente generada cuando incide la radiación solar sobre la célula.
- **Corriente de oscuridad:** es debida a la recombinación de los pares electrón-hueco que se produce en el interior del semiconductor.
- **Tensión de circuito abierto (V_{OC}):** la máxima tensión que se obtiene en los extremos de la célula solar, que se da cuando no está conectada a ninguna carga. Es una característica del material con el que está construida la célula.
- **Corriente de corto circuito (I_{SC}):** máximo valor de corriente que puede circular por la célula solar. Se da cuando sus terminales están cortocircuitadas.

Cuando la célula solar es conectada a una carga, los valores de tensión e intensidad varían. Existirán dos de ellos para los cuales la potencia entregada sea máxima: V_m (tensión máxima) e I_m (intensidad de corriente máxima), que siempre serán menores que V_{OC} e I_{SC} . En función de estos valores, la potencia máxima que puede entregar la célula solar es la indicada en la ecuación 2-1.

$$P_m = V_m I_m \quad \text{Ec. 2-1}$$

Esto permite definir un parámetro de la célula solar que recibe el nombre de *factor de forma (FF)* y que se calcula mediante la fórmula de la ecuación 2-2.

$$FF = \frac{V_m I_m}{V_{OC} I_{SC}} \quad \text{Ec. 2-2}$$

Así pues, el **factor de forma** es el cociente entre la máxima potencia que puede entregar la célula a la carga y el producto de la tensión de circuito abierto y la corriente de cortocircuito. En las células solares más habituales, los valores típicos de FF son 0.7 o 0.8 [11].

2.3.2 Panel Solar

Un **panel solar** o un **módulo fotovoltaico** está formado por un conjunto de células, conectadas eléctricamente, encapsuladas y montadas sobre una estructura de soporte o marco. Proporciona en su salida de conexión una tensión continua, y se diseña para valores concretos de tensión (6V, 12V, 24V, ...), que definirán la tensión a la que va a trabajar el sistema fotovoltaico.

En la Figura 2- puede ser apreciada la constitución de un panel solar.

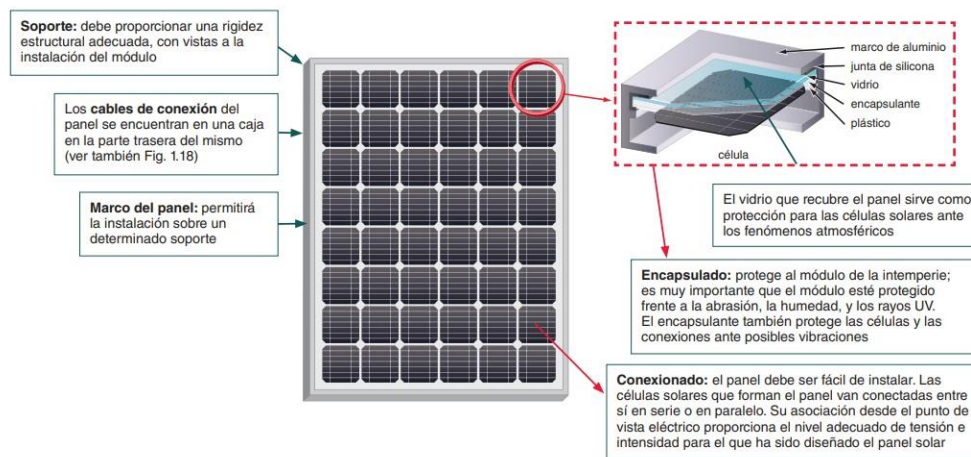


Figura 2-4 Constitución de un panel solar.

Además, los tipos de paneles solares vienen dados por la tecnología de fabricación de las células (tabla 2-1), y son fundamentalmente [11]:

- Silicio cristalino (monocristalino y multicristalino).
- Silicio amorfo.

Tabla 2-1. Tipos de células solares.

Silicio	Rendimiento laboratorio	Rendimiento directo	Fabricación
Monocristalino	24%	15 – 18%	Se obtiene de silicio puro fundido y dopado con boro.
Policristalino	19 – 20 %	12 -14 %	Igual que el del monocristalino, pero se disminuye el número de fases de cristalización.
Amorfo	16 %	< 10 %	Tiene la ventaja de depositarse en forma de lámina delgada y sobre un sustrato como vidrio o plástico

2.3.3 Regulador

Para un correcto funcionamiento de la instalación, hay que instalar un sistema de regulación de carga en la unión entre los paneles solares y las baterías. Este elemento recibe el nombre de regulador y tiene como misión evitar situaciones de carga y sobredescarga de la batería, con el fin de alargar su vida útil.

El regulador trabaja por tanto en las dos zonas. En la parte relacionada con la carga, su misión es garantizar una carga suficiente al acumulador y evitar las situaciones de sobrecarga, mientras que en la parte de descarga se ocupará de asegurar el suministro eléctrico diario suficiente y evitar la descarga excesiva de la batería [11].

De igual manera, en la Figura 2-5 se representan las conexiones del regulador en una instalación fotovoltaica.

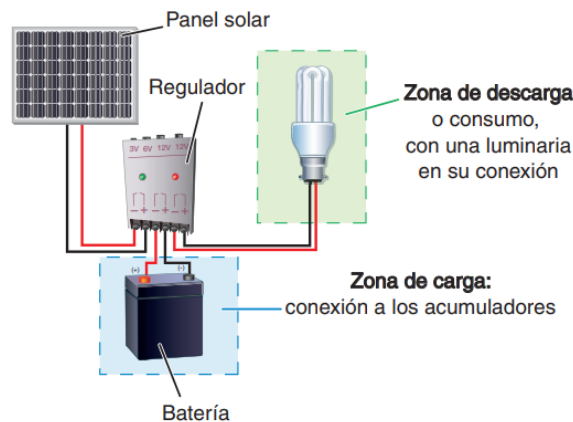


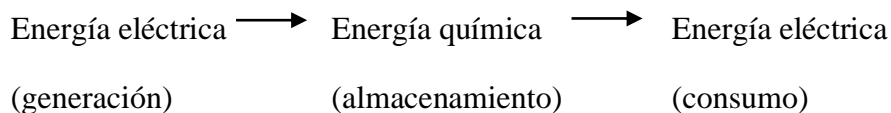
Figura 2-5. Conexiones del regulador en una instalación fotovoltaica.

2.3.4 Baterías

La llegada de la energía solar a los módulos fotovoltaicos no se produce de manera uniforme, sino que presenta variaciones por diferentes motivos. Algunas de estas variaciones son predecibles, como la duración de la noche o las estaciones del año, pero existen otras causas que pueden producir alteraciones de manera aleatoria en la energía recibida, como puede ocurrir con un aumento de nubosidad en un determinado instante.

Este hecho hace necesario utilizar algún sistema de almacenamiento de energía para aquellos momentos en que la radiación recibida sobre el generador fotovoltaico no sea capaz de hacer que la instalación funcione en los valores diseñados. Para ello se utilizan baterías.

Es de recordar que, las baterías son dispositivos capaces de transformar energía química en eléctrica. El funcionamiento en una instalación fotovoltaica es el siguiente:



Las baterías son recargadas desde la electricidad producida por los paneles solares, a través de un regulador de carga, y pueden entregar su energía a la salida de la instalación, donde será consumida.

En esencia, la misión de las baterías en una instalación fotovoltaica es:

- Almacenar energía durante un determinado periodo de tiempo.
- Proporcionar una potencia instantánea elevada.
- Fijar la tensión eléctrica de trabajo de la instalación.

Uno de los parámetros más importantes que tener en cuenta a la hora de elegir una batería es la capacidad. Se define como la cantidad de electricidad que puede lograrse en una descarga completa del acumulador partiendo de un estado de carga total del mismo. Se mide en amper-hora (Ah), y se calcula como el producto de la intensidad de descarga del acumulador durante el tiempo en el que está actuando: $C = t I$.

De igual manera, deben tomarse en cuenta los siguientes parámetros en instalaciones fotovoltaicas.

- **Eficiencia de carga:** relación entre la energía empleada para recargar una batería y la energía realmente almacenada. Interesa que sea un valor lo más alto posible (cercano al 100%, lo que indicaría que toda la energía utilizada para la recarga es factible de ser empleada en la salida de la instalación). Si la eficiencia es baja, será necesario aumentar el número de paneles solares para obtener los resultados deseados.
- **Autodescarga:** proceso mediante el acumulador, sin estar en uso, tiende a descargarse.
- **Profundidad de descarga:** cantidad de energía, en tanto por ciento, que se obtiene de la batería durante una determinada descarga, partiendo del acumulador totalmente cargado. Está relacionado con la duración o vida útil del acumulador [11].

2.3.4.1 Tipos de baterías

Las baterías se clasifican en función de la tecnología de fabricación y de los electrolitos utilizados, tal como se observa en la tabla 2-2.

Tabla 2-2 Clasificación de baterías.

Tipo de batería	Tensión por vaso [V]	Tiempo de recarga	Autodescarga por mes	No. de ciclos	Capacidad (por tamaño)	Precio
Plomo ácido	2	8 – 16 horas	< 5%	Medio	30-50 Wh/kg	Bajo
Ni-Cd (níquel-cadmio)	1.2	1 hora	20 %	Elevado	50 – 80 Wh/kg	Medio
Ni-Mh (níquel –	1.2	2 – 4 horas	20 %	Medio	60 – 120 Wh/kg	Medio

metal hydride)						
Li ion (ión litio)	3.6	2 – 4 horas	6 %	Medio – bajo	110 – 160 Wh/kg	Alto

Las baterías más utilizadas en las instalaciones solares son las de plomo-ácido, por las características que presentan. Dentro de este tipo de baterías nos podemos encontrar diferentes modelos, tal como se observa en la tabla 2-3.

Tabla 2-3 Tipos de baterías

Tipo	Ventajas	Inconvenientes
Tubular estacionaria	<ul style="list-style-type: none"> • Ciclado profundo • Tiempos de vida largos • Reserva de sedimentos 	<ul style="list-style-type: none"> • Precio elevado • Disponibilidad escasa en determinados lugares.
Arranque (SLI, automóvil)	<ul style="list-style-type: none"> • Precio • Disponibilidad 	<ul style="list-style-type: none"> • Mal funcionamiento ante ciclado profundo y bajas corrientes. • Tiempo de vida corto. • Escasa reserva de electrolito.
Solar	<ul style="list-style-type: none"> • Fabricación similar a SLI • Amplia reserva de electrolito • Buen funcionamiento en ciclados medios 	<ul style="list-style-type: none"> • Tiempos de vida medios. • No recomendada para ciclados profundos y prolongados.
Gel	<ul style="list-style-type: none"> • Escaso mantenimiento 	<ul style="list-style-type: none"> • Deterioro rápido en condiciones de funcionamiento extremas de V-I.

2.3.5 Inversor

El inversor se encarga de convertir la corriente continua de la instalación en corriente alterna, igual utilizada en la red eléctrica, es decir, 127 V a 60 Hz.

La Figura 2- muestra una instalación autónoma con inversor.

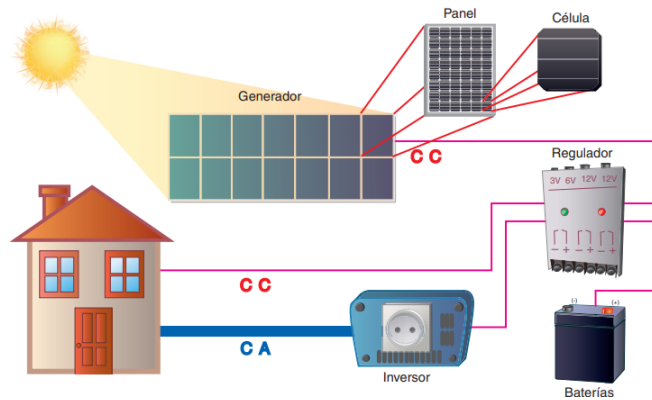


Figura 2-6 Esquema general de una instalación autónoma con inversor.

La misión del inversor en las instalaciones autónomas es proporcionar una corriente alterna como la de la red eléctrica, con el fin de que se puedan conectar a la misma electrodomésticos de los más utilizados habitualmente en las viviendas [11].

En el caso de las instalaciones conectadas a la red, el inversor debe proporcionar una corriente alterna que sea de las mismas características de la red eléctrica a la que está conectado, tanto en forma (senoidal) como en valor eficaz (127 V) y sobre todo en la frecuencia (60 Hz). No se permiten variaciones con el fin de evitar perturbaciones sobre la red eléctrica de distribución [11]. La Figura 2-7 clarifica una instalación fotovoltaica conectada a la red.

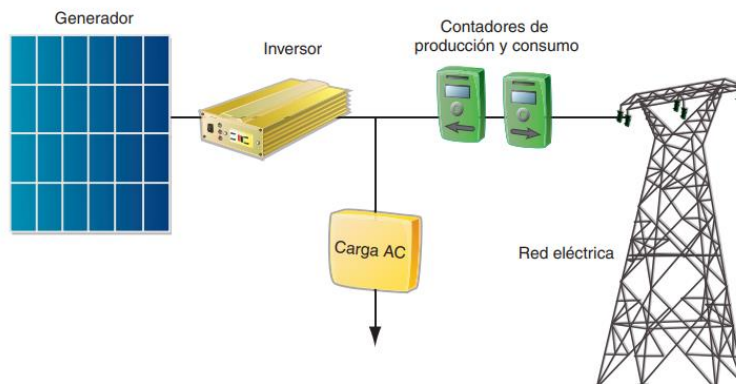


Figura 2-7. Instalación fotovoltaica conectada a la red.

Las características deseables para un inversor DC-AC pueden ser resumidas en:

- **Alta eficiencia:** debe funcionar para un amplio rango de potencias.
- **Bajo consumo en vacío:** es decir cuando no haya cargas conectadas.
- **Alta fiabilidad:** resistencia a los picos de arranque.
- **Protección contra cortocircuitos.**
- **Seguridad.**
- **Buena regulación de tensión eléctrica y de frecuencia de salida.**

2.4 Arduino IDE

El IDE (Integrated Development Environment) es un conjunto de herramientas de software que permiten a los programadores desarrollar y grabar todo el código necesario para que Arduino funcione como es deseado. El IDE de Arduino permite escribir, depurar, editar y grabar programas. La figura 2-8 muestra el Entorno Integrado de Desarrollo de Arduino [12].

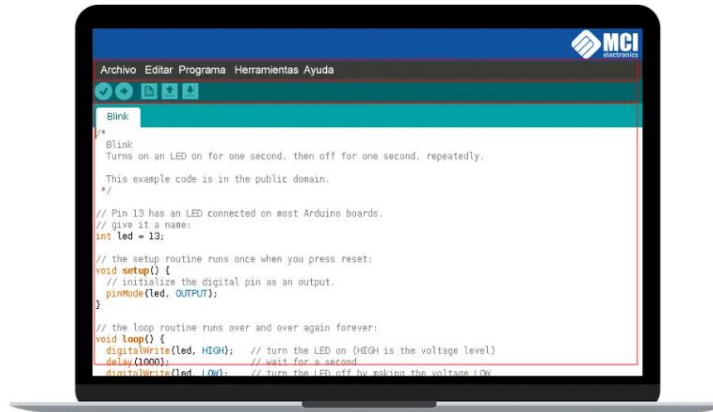


Figura 2-8. Arduino IDE.

2.5 ESP32

El módulo ESP32 es una solución de Wi-Fi/Bluetooth todo en uno, integrada y certificada que proporciona no solo la radio inalámbrica, sino también un procesador integrado con interfaces para conectarse con varios periféricos. El procesador en realidad tiene 2 núcleos de procesamiento cuyas frecuencias operativas pueden controlarse independientemente entre 80 MHz y 240 MHz. Los periféricos del procesador facilitan la conexión a una variedad de interfaces externas como [13]:

- Interfaz periférica serial (SPI),
- I2C,
- Transmisor receptor asíncrono universal (UART),
- Ethernet,
- Tarjetas SD,
- Interfaces táctiles y capacitivas.

Algunas de sus características principales son [14]:

- **Procesador principal:** Tensilica Xtensa LX6 de 32 bits.
- **Wi-Fi:** 802.11 b / g / n / e / i (802.11n @ 2.4GHz hasta 150 Mbps).
- **Bluetooth:** v4.2 BR / EDR y Bluetooth Low Energy (BLE).
- **Frecuencia de reloj:** Programable hasta 240 Mhz.
- **Rendimiento:** Hasta 600 DMIPS.
- **ROM:** 448 KB para arranque y funciones básicas.
- **SRAM:** 520 KiB, para datos e instrucciones.

La figura 2-9 muestra el pinout de la tarjeta ESP32.

ESP32 Wroom DevKit Full Pinout

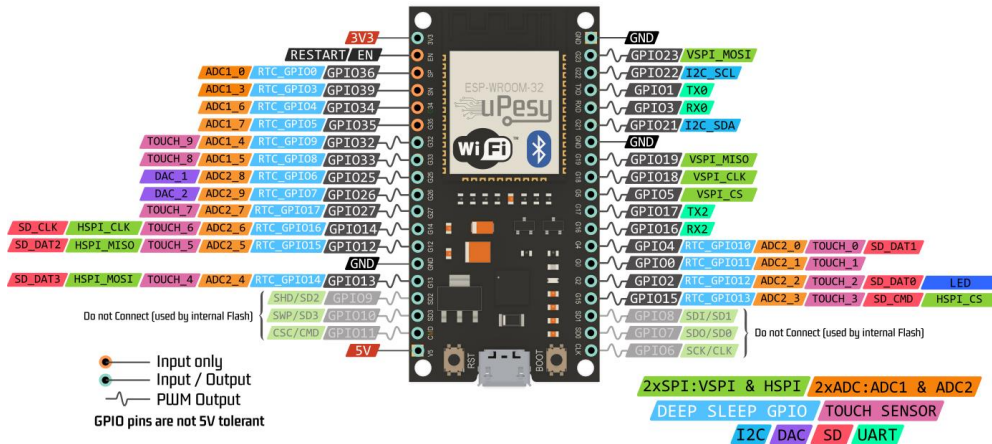


Figura 2-9. Pinout de la tarjeta ESP32

2.6 Sensores

2.6.1 Sensor resistivo de humedad de sustrato

El sensor de humedad de sustrato FC-28 permite medir de forma sencilla la humedad de sustrato por medio de 2 electrodos resistivos. Compatible con Arduino, PIC, NodeMCU.

El funcionamiento del sensor se basa en medir la resistencia entre 2 electrodos insertados dentro del sustrato, la resistencia entre los electrodos dependerá de la humedad del mismo, por lo que un sustrato muy húmedo tendrá una resistencia muy baja (cortocircuito) y para un sustrato muy seco una resistencia muy alta (circuito abierto). El electrodo va conectado a una tarjeta de acondicionamiento (YL-38) que entrega una salida digital y una salida analógica. La salida digital DO es la salida de un opamp en modo comparador, la salida digital se activa cuando el nivel de humedad es menor al deseado, este nivel (umbral o threshold) se puede regular con el potenciómetro de la tarjeta.

En cambio, la salida analógica A0 es la salida de un divisor de tensión entre una resistencia fija y la resistencia entre los electrodos, entrega un voltaje analógico desde 0V para un suelo muy húmedo hasta 5V para un suelo muy seco.

Se estima que la vida útil del electrodo sumergido es de 3 a 6 meses, por lo que se sugiere alimentar el sensor solo cuando va a ser leído y no permanentemente [15].

La figura 2-10 muestra el electrodo, así como el acondicionador de señal del sensor FC-28.

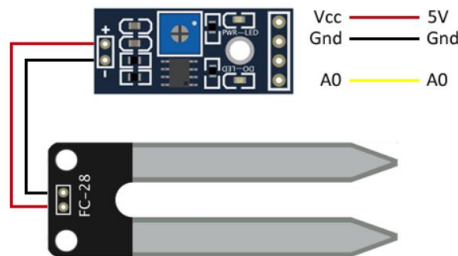


Figura 2-10. Esquema de montaje del sensor de humedad resistivo FC-28.

2.6.2 Sensor de humedad de sustrato capacitivo v1.2.

Un higrómetro capacitivo es un sensor que mide la humedad del suelo mediante la variación de capacitancia. Es un sensor habitual en sistemas de riego automático.

La principal ventaja que tienen los sensores capacitivos es que no tienen el problema de oxidación de los electrodos, que sí tienen los sensores resistivos.

Además, su precisión es ligeramente superior.

El sensor tiene una tensión de operación de 3.3V a 5v. Su salida es una señal analógica de 0 a 3V. El consumo típico es de 5mA.

Es importante indicar que el sensor dispone de una línea blanca que indica la profundidad máxima a la que se puede introducir el sensor al sustrato.

En este sensor, cuanto mayor es la humedad del suelo, mayor es la capacidad registrada por el sensor. Valores habituales son 2.3-2.5V para sensor totalmente seco (en el aire) y 1.2-1.3V para sensor totalmente húmedo (en el agua) [16].

La Figura 2-11 muestra el sensor de humedad capacitivo.



Figura 2-11. Pinout del sensor de humedad de sustrato capacitivo v1.2.

2.6.3 Sensor de gas MQ

Los sensores de gases MQ son una familia de dispositivos diseñados para detectar la presencia de distintos componentes químicos en el aire. Existe una gran variedad de sensores MQ. Cada modelo está diseñado para detectar una o más sustancias, pensadas para un uso específico, como, por ejemplo, detección de gases inflamables, calidad del aire o detección de alcohol en aire respirado.

Los sensores de gases MQ suelen proporcionarse con una placa de medición estándar con el comparador LMC662 o similar, que permite obtener la lectura tanto como un valor analógico como un valor digital cuando se supera cierto umbral regulado a través de un potenciómetro ubicado en la placa.

Los sensores de gases deben ser calibrados antes de obtener una medida precisa. Aún calibrados estos sensores no disponen de la garantía necesaria para formar parte de un sistema de seguridad [17].

La Figura 2-12 muestra algunos sensores de gas de la serie MQ.

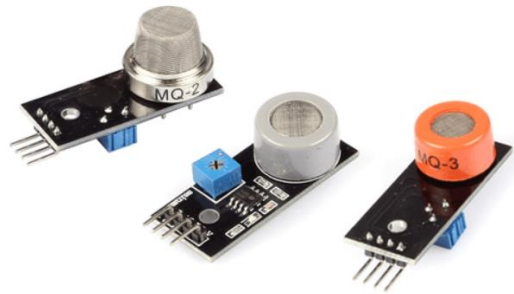


Figura 2-12. Algunos sensores de gas de la serie MQ.

Los sensores MQ están compuestos por un sensor electroquímico que varía su resistencia al estar en contacto con las sustancias.

Los sensores de gases son dispositivos de alta inercia, es decir, la respuesta necesita tiempos largos para estabilizarse tras un cambio de concentración de los gases medidos. Ello es debido a la necesidad física de que el gas abandone el material sensible, lo cual es un proceso lento.

Todos los modelos MQ disponen de un calentador necesario para elevar la temperatura del sensor y que sus materiales adquieran la sensibilidad. Mientras el calentador no alcance la temperatura de funcionamiento, la lectura del sensor no será fiable. El tiempo de calentamiento depende de cada modelo de sensor. En la mayoría de los modelos, es suficiente con unos minutos, pero otros requieren de 12 hasta 48 horas para obtener mediciones estables.

El consumo de los sensores MQ puede ser elevado debido al calor necesario para hacer funcionar el calentador, que puede llegar hasta los 800 mW. Esto puede ser superior a la potencia que puede suministrar el microcontrolador, por lo que será necesario una fuente de alimentación externa [17].

La tabla 2-4 muestra algunas de las sustancias que detectan los modelos de la familia de sensores MQ.

Tabla 2-4. Sensores de la familia MQ, sustancias que detectan y tensión del calentador

Modelo	Sustancias detectadas	Calentador
MQ-2	Metano, butano, GLP, humo	5 V
MQ-4	Metano, gas natural comprimido (GNP)	5 V
MQ-5	Gas natural, GLP	5V
MQ-6	Butano, GLP	5 V

El esquema de montaje es sencillo, basta con alimentar el sensor y conectar los pines correspondientes al microcontrolador [17]. La Figura 2-13 muestra el pinout general de la familia de sensores MQ.

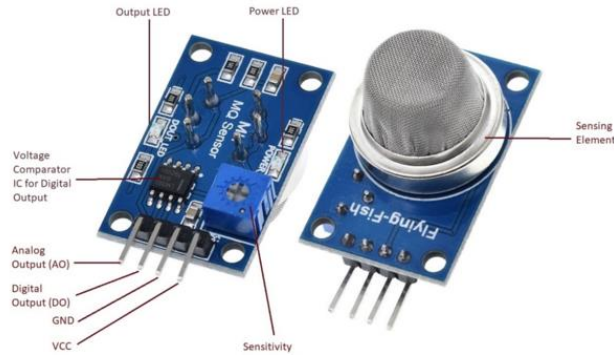


Figura 2-13. Pinout general de la familia de sensores de gas MQ.

2.6.3.1 MQ-4

El sensor MQ-4 es altamente sensible a gas metano y gas natural. En contraparte, es poco sensible a alcohol y humo. Es usado en la detección de fugas de los gases mencionados [18].

La figura 2-14 muestra el sensor MQ-4.



Figura 2-14 Sensor MQ-4.

2.6.3.2 MQ-5

El sensor MQ-5 es altamente sensible a gas licuado de petróleo (GLP) y gas natural. En contraparte, es poco sensible a gases como vapor de alcohol y humo. Tiene un rango de detección desde 200 ppm hasta 10000 ppm [19].

La figura 2-15 muestra el sensor MQ-5.



Figura 2-15. Sensor MQ-5.

2.6.3.3 MQ-6

El sensor MQ-6 es sensible a gas LP y metano. También es capaz de detectar iso butano, propano, humo y alcohol. Tiene un rango de detección desde 200 a 10,000 ppm [20].

La Figura 2-16 muestra el sensor MQ-6.



Figura 2-16. Sensor MQ-6.

2.6.4 Módulo *RFID*

Los lectores *RFID* (*Radio Frequency Identification*) abarcan usos en los sistemas de seguridad, acceso de personal, identificación y logística de productos, como llaves de puertas eléctricas, entre otras aplicaciones.

Su principio de funcionamiento consiste en pasar un *TAG* cerca de un lector *RFID*, el *TAG* tiene la capacidad de enviar información al lector. Dicha información puede ser desde un simple código o todo un paquete de información guardado en la memoria del *TAG*.

Los *TAG* vienen en diferentes modelos, los más comunes son tarjetas y llaveros, pero también vienen como etiquetas adhesivas e incluso ya vienen incrustados en algunos productos. Los *TAG* tienen internamente una antena y un microchip, encargado de realizar todo el proceso de comunicación, la energía la obtiene de la señal de radiofrecuencia, que, si bien la energía en la señal es pequeña, es suficiente para hacer trabajar el microchip, es por ello que generalmente es necesario acercarlos a una distancia de aproximadamente 10 cm [21].

El módulo *RFID* RC522 está diseñado para leer etiquetas (tags) a corta distancia de forma inalámbrica, posee comunicación *SPI* lo que facilita su uso con la mayoría de microcontroladores. Utiliza un sistema de modulación y demodulación para todo tipo de dispositivos pasivos *RFID* de 13.56MHz.

Los *RFID* RC522 son ampliamente empleados, por ejemplo, en sistemas de alarma, aplicaciones comerciales en sustitución de códigos de barras, cerraduras electrónicas, sistemas de pago, tarjetas personales, control de accesos recintos como gimnasios o piscinas, fichaje en empresas, entre otras muchas aplicaciones [22].

La Figura 2-17 muestra el módulo *RFID* RC522.

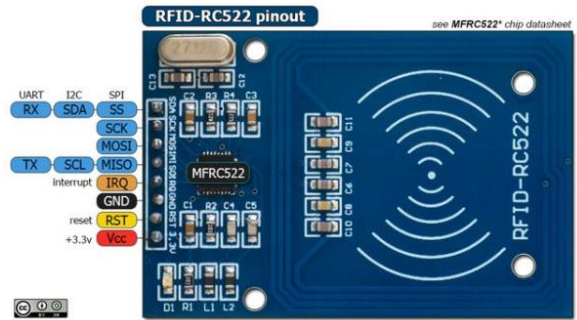


Figura 2-17. Módulo RFID RC522.

2.7 IoT

2.7.1 Blynk

Blynk ofrece una plataforma de Internet de las Cosas que ofrece software, firmware, soluciones web y aplicaciones móviles a miles de pequeñas, medianas y grandes empresas de todo el mundo. *Blynk* es la plataforma de *IoT* más popular para conectar dispositivos a la nube, diseñar aplicaciones para controlarlos y supervisarlos de forma remota, y administrar miles de productos implementados. El software *Blynk* ayuda a individuos y organizaciones a avanzar fluidamente desde un prototipo de un producto hasta su lanzamiento comercial [23].

2.7.2 Cayenne

Cayenne es de las primeras plataformas *IoT* del tipo arrastrar y soltar que permite a los desarrolladores, diseñadores e ingenieros crear rápidamente prototipos y compartir sus proyectos. Cayenne fue diseñado para ayudar a los usuarios a crear prototipos con el Internet de las cosas para después pasarlos a producción

El componente más valioso de Cayenne es su *dashboard* en línea, el cual usa *widgets* personalizables para visualizar datos, establecer reglas, horarios, eventos y más.

Es necesario contar con una Raspberry Pi o un Arduino conectado a internet. La lista de hardware que funciona con Cayenne continúa aumentando [24].

2.7.3 ThingSpeak

ThingSpeak es un servicio *IoT* que permite añadir, visualizar y analizar un conjunto de datos en vivo en la nube. Se puede enviar información a ThingSpeak desde dispositivos, crear visualización de datos en vivo, y enviar alertas.

ThingSpeak puede recolectar datos de manera privada en la nube, para después analizar y visualizarlos con ayuda de MATLAB, y así, ejecutar una acción [25].

2.8 Medidas de tendencia central

Las medidas de tendencia central son medidas estadísticas que pretenden resumir en un solo valor a un conjunto de valores. Representan un centro en torno al cual se encuentra

ubicado el conjunto de los datos. Las medidas de tendencia central más utilizadas son: **media, mediana y moda.**

Los procedimientos para obtener las medidas estadísticas difieren levemente dependiendo de la forma en que se encuentren los datos. Si los datos se encuentran de manera ordenada en una tabla estadística, entonces se encuentran “ordenados”; en cambio, si los datos no están en una tabla, se consideran “no agrupados” [26].

2.8.1 Promedio o media

La medida de tendencia central más conocida y utilizada es la media o promedio aritmético. Se representa por la letra griega μ cuando se trata del promedio del universo o población y por \bar{y} cuando se trata del promedio de la muestra. Se calcula con la expresión mostrada en la ecuación 2-3 [27].

$$\bar{y} = \frac{\sum_{i=1}^n Y_i}{n} \quad (\text{Ec. 2-3})$$

Donde:

n: cantidad de datos

Y: valor de la variable

i: i-ésima observación

2.8.2 Mediana

La mediana es el valor de la variable que ocupa la posición central, cuando los datos se disponen en orden de magnitud. Es decir, el 50% de las observaciones tiene valores iguales o inferiores a la mediana, y el otro 50% tiene valores iguales o superiores a la mediana [27].

2.8.3 Moda

La moda de una distribución se define como el valor de la variable que más se repite. En el caso de dos valores que presenten la misma frecuencia, se dice que existe un conjunto de datos bimodal. Para más de dos modas, se habla de un conjunto de datos multimodal.

Aunque tanto la media como la mediana son buenas medidas del centro de una distribución, la mediana es menos sensible a valores extremos o resultados atípicos. Si una distribución está fuertemente sesgada por uno o más valores extremos, el usuario debe emplear la mediana en lugar de la media como medida de centro [27].

2.9 Señales

Una señal se puede considerar como una variable o cantidad física que provee información sobre el estado o evolución de un sistema o fenómeno. La variable física puede ser función del tiempo, el espacio o cualquier otra variable o variables, es decir, que una señal puede considerarse como un fenómeno físico (cantidad física) que experimenta cambios en el tiempo, el espacio u otra variable independiente. Una señal es una descripción de cómo un parámetro varía con otro parámetro [28].

2.9.1 Señal analógica

Una señal analógica es una señal que varía de forma continua a lo largo del tiempo. La mayoría de las señales que representan una magnitud física (temperatura, luminosidad, humedad, etc.) son señales analógicas. Pueden tomar todos los valores posibles de un intervalo.

Las señales análogas se pueden percibir en todos los lugares, por ejemplo, la naturaleza posee un conjunto de estas señales, como es la luz, la energía, el sonido, etc., debido a que estas son señales que varían constantemente [29].

2.9.2 Señal digital

Es aquella que presenta una variación discontinua con el tiempo y que solo puede tomar ciertos valores discretos. Su forma característica es ampliamente conocida: la señal es una onda cuadrada (pulsos) y las representaciones se realizan en el dominio del tiempo. Sus parámetros son:

- Altura de pulso (nivel eléctrico).
- Duración (ancho de pulso).
- Frecuencia de repetición (velocidad de pulsos por segundo).

Las señales digitales no se producen en el mundo físico, sino que son creadas por el hombre y tiene una técnica particular de tratamiento [29].

2.9.3 Digitalización de una señal analógica

Es la transformación de señales analógicas en digitales, para simplificar su posterior proceso de codificación, compresión, etc., de esta forma obtener una señal más conveniente para el trabajo [30].

Una conversión analógico digital está formada por los siguientes procesos:

- **Muestreo:** se toman diferentes valores en el tiempo de la señal analógica. Dependiendo de la cantidad de valores tomados o muestras, la información obtenida será más o menos completa. La velocidad con la que se toman las muestras se denomina frecuencia o velocidad de muestreo. La figura 2-18 indica la comparación de una señal original contra una señal muestreada.

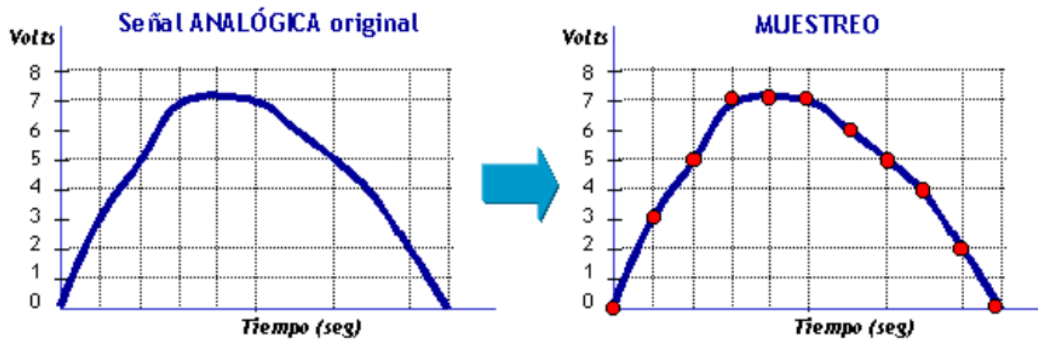


Figura 2-18. Señal original v.s. señal muestreada.

- **Cuantificación:** Consiste en asignar un margen de valor de una señal analizada a un único nivel de salida.
- **Codificación:** Como su propio nombre lo dice, es la traducción de los valores obtenidos en la cuantificación a un lenguaje o sistema, el cual pueda ser entendido por un sistema digital.

Normalmente la información se codifica a un lenguaje binario debido a que es el más utilizado [30].

La Figura 2-19 muestra el proceso de cuantificación y codificación.

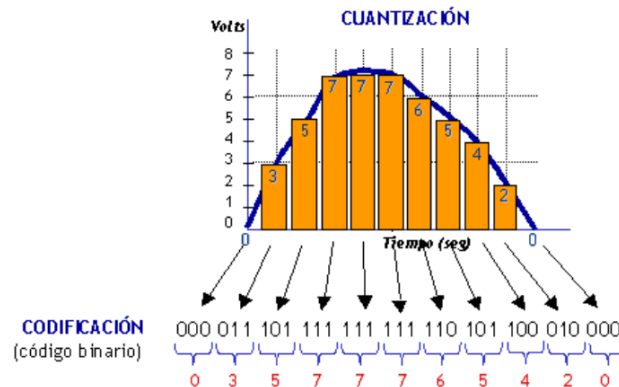


Figura 2-19. Cuantización y Codificación de una señal.

2.10 Etapa de Potencia

2.10.1 Relevador

El relevador es un interruptor eléctrico que permite el paso de la corriente eléctrica cuando está cerrado e interrumpla cuando está abierto, pero que es accionado eléctricamente, no manualmente.

El relé está compuesto de una bobina conectada a una corriente. Cuando la bobina se activa produce un campo magnético que hace que el contacto del relé que está normalmente abierto se cierre y permita el paso de la corriente. En contraparte, cuando se deja de

suministrar corriente a la bobina, el campo electromagnético desaparece y el contacto del relé se vuelve a abrir, interrumpiendo el paso de corriente [31].

Existen diferentes tipos de relés.

- Electromecánicos,
- Estado sólido,
- Temporizador y
- Térmicos.

2.10.2 Optoacoplador

Es un circuito electrónico que funciona como un interruptor aislado ópticamente. Es decir, que permite una conexión eléctricamente aislada entre dos circuitos que operan a distintos voltajes. Está construido por un led y un circuito de control activado por luz infrarroja. La única conexión entre ambos elementos es la luz del LED que activa al fototransistor.

La figura 2-20 muestra el diagrama de un optoacoplador.

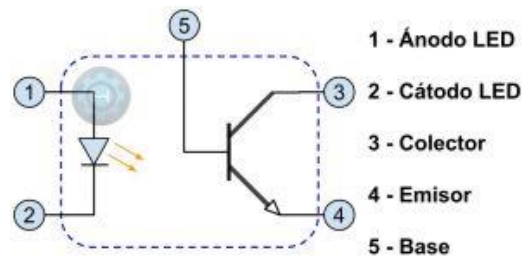


Figura 2-20 Diagrama de un optoacoplador.

Las aplicaciones de optoacopladores incluyen el de activar cargas que puedan inducir ruido eléctrico al sistema de control. Cuando una carga inductiva como un motor se activa y desactiva produce perturbaciones eléctricas. Por tanto, los optoacopladores se usan para aislar a estas perturbaciones electrónicas [32].

2.10.3 Capacitor

Es un dispositivo capaz de almacenar energía a través de campos eléctricos. Se clasifica dentro de los componentes pasivos eléctricos ya que no tiene la capacidad de amplificar o cortar el flujo eléctrico.

Los capacitores se utilizan principalmente como filtros de corriente continua ya que evitan cambios bruscos y ruidos en las señales debido a su funcionamiento [33].

Consta de 3 partes esenciales.

- **Placas metálicas:** Estas placas se encargan de almacenar las cargas eléctricas.
- **Material dieléctrico:** Sirve para evitar el contacto entre las dos placas.
- **Carcasa de plástico:** Cubre las partes internas del capacitor.

La figura 2-21 muestra los diferentes tipos de capacitores.

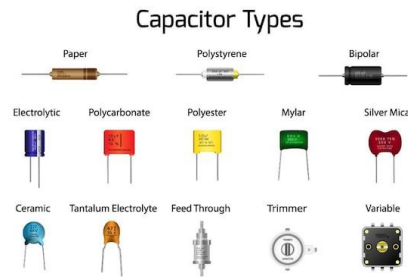


Figura 2-21. Tipos de capacitores.

2.11 Radiación solar

La energía proveniente del sol se denomina energía solar, también denominada radiación solar. La radiación solar que recibe la tierra es del orden de $1.5 \times 10^{18} \text{ kW}$ por hora. Ante estos datos, se podría decir que se dispone de una fuente de energía con un enorme potencial, además de ser renovable.

No obstante, esta fuente de energía presenta ciertos inconvenientes, tal como: la forma de captación, de almacenamiento; además de presentar variaciones debido a las condiciones: meteorológicas, ambientales y geográficas.

La radiación solar emitida por el sol llega a la atmósfera de la Tierra considerablemente debilitada, aproximadamente $1360 \frac{\text{W}}{\text{m}^2}$, debido a la distancia entre el Sol y la Tierra. Después esta radiación sufre una atenuación debido a la capa atmosférica, por lo que la radiación en la superficie terrestre es de aproximadamente $1000 \frac{\text{W}}{\text{m}^2}$ [34].

Se distinguen tres tipos de radiación solar en función de cómo inciden los rayos del Sol sobre la Tierra:

- **Directa:** Es la recibida desde el Sol sin que se desvíe en su paso por la atmósfera.
- **Difusa:** Es la que sufre cambios en su dirección principalmente debidos a la reflexión y difusión de la atmósfera.
- **Albedo o reflejada:** Es la radiación directa y difusa que se recibe por reflexión en el suelo u otras superficies próximas.

De entre los tres tipos, la radiación directa es la mayor y más importante en las aplicaciones fotovoltaicas y fototérmicas. Aunque en días nublados (por cuestiones meteorológicas) en las cuales no se recibe radiación directa (o debido a otro obstáculo), se continúa recibiendo radiación solar sobre la superficie debido a la radiación difusa. A dichos días se les denomina *días de poca radiación solar* [34]. La figura 2-22 ejemplifica la radiación.

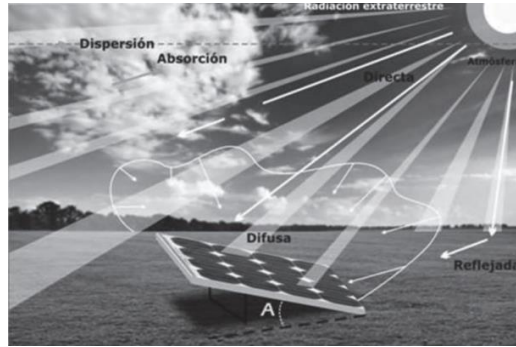


Figura 2-22. Radiación

2.11.1 Unidades de radiación solar

Para los cálculos de dimensionado se debe dar un valor de radiación solar y sus unidades son los $\frac{kW}{m^2}$, aunque en algunas documentaciones se utilizan los Joules.

Para obtener el valor de radiación solar de una determinada zona se puede tener en cuenta:

Irradiancia: Se define como el flujo de radiación solar que incide sobre una unidad de superficie en un tiempo dado. Se expresa normalmente en $\frac{W}{m^2}$.

Irradiación: Se define como la energía por unidad de superficie a lo largo de un periodo de tiempo. Se expresa en Joules (energía) por metro cuadrado [$\frac{J}{m^2}$], aunque también se expresa en [$\frac{Wh}{m^2}$] (potencia), mediante la conversión de unidades [34].

En México la irradiación diaria promedio anual es mostrada en la figura 2-23.

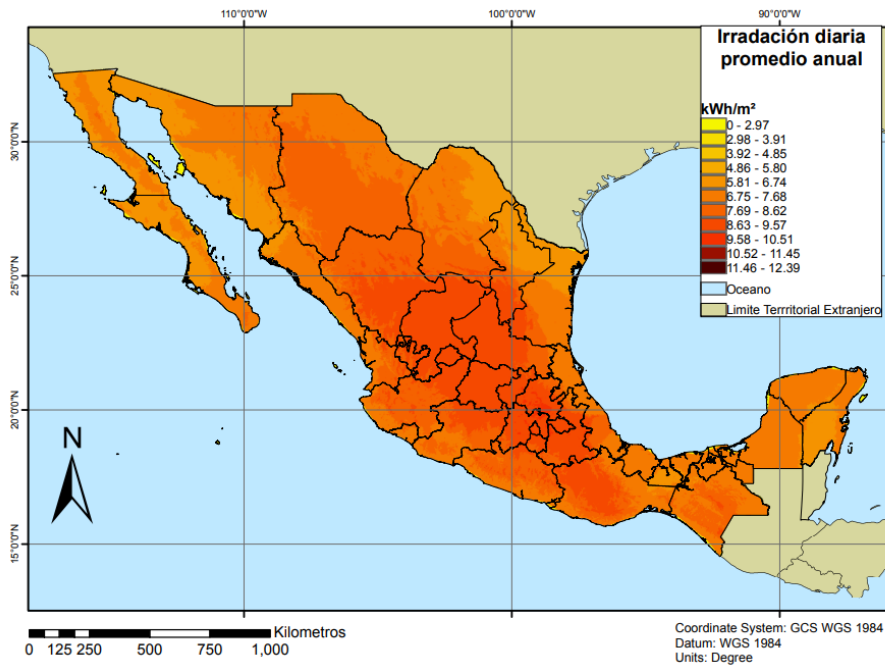


Figura 2-23. Irradiación diaria promedio anual en México.

Por otra parte, la figura 2-24 muestra el promedio diario mensual de horas pico de radiación solar en la Ciudad de México.



Figura 2-24. Promedio diario mensual de horas pico de radiación solar en la Ciudad de México.

Capítulo 3. Desarrollo experimental

En este capítulo se presenta el desarrollo de tres módulos para soluciones domóticas que utilizan el *IoT*.

El primer módulo consiste en una maceta inteligente que permite el riego automático de plantas, basado en la medición de la humedad de sustrato. El segundo módulo es un sistema para detección de gas licuado de petróleo y cierre de una electroválvula que permite el control del suministro de gas en una red de tuberías. Finalmente, el tercer módulo es un sistema de control de accesos mediante *RFID*, que permite la entrada de personas autorizadas al hogar. Todos estos módulos se alimentan con paneles solares, lo que los hace autónomos y sostenibles.

Se especifica cada uno de estos módulos de forma detallada, incluyendo la descripción de los componentes utilizados, el diseño del sistema, y la implementación de la solución.

En resumen, los tres módulos desarrollados en este capítulo son soluciones domóticas que utilizan el *IoT* y se alimentan con paneles solares. Cada uno de ellos presenta un enfoque semejante, permitiendo automatizar procesos en el hogar, con el afán de mejorar la calidad de vida de las personas y haciendo un uso más eficiente de los recursos disponibles.

3.1 Diagrama de bloques

A través de la lectura de variables el microcontrolador toma decisiones, a la vez que transmite la información al usuario para que conozca los valores obtenidos. En este caso, la plataforma *IoT* solo muestra información, pero no permite al usuario interactuar con la variable. La Figura 3-1 muestra el diagrama a bloques de los elementos que conforman al sistema (módulo) para la detección de gas licuado de petróleo y cierre de electroválvula.

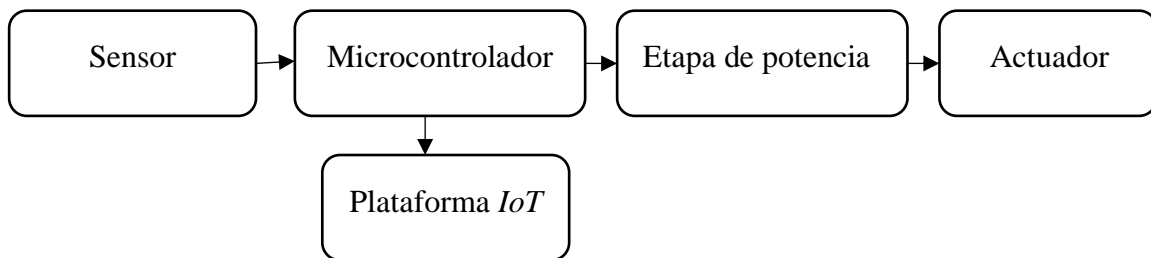


Figura 3-1 Diagrama de bloques del módulo detección de gas licuado de petróleo y cierre de electroválvula.

Siendo:

- Sensor: dispositivo que se encarga de la medición de una variable, para el módulo presentado se ha empleado un módulo que incorpora al MQ5, el cual es principalmente sensible a gas licuado de petróleo, por lo que es particularmente útil para detección de fugas de en un cuarto de cocina.
- Microcontrolador: dispositivo que recibe lecturas y toma decisiones, además de transmitir la información necesaria a la plataforma *IoT*, para este sistema se ocupa una tarjeta de desarrollo que incorpora una versión del microcontrolador ESP32, seleccionado debido a su capacidad de conectarse a redes *Wifi*.

- Plataforma *IoT*: Plataforma donde el usuario percibe los datos tomados del sistema, trabajando con *Blynk*.
- Etapa de potencia: acople de circuito de control con el actuador.
- Actuador: para el caso, electroválvula de diámetro de ½ pulgada alimentada a 12 Vdc.

El módulo para el riego de plantas y para el control de accesos, fueron diseñados para que el usuario pueda interactuar con las tomas de decisión. Para el caso del módulo de riego, se puede a través de la interfaz de usuario creada en *Blynk*, modificar el nivel de humedad en el sustrato que el sistema debe mantener de manera automática, y esto fue así debido a que el valor adecuado de esta variable depende del tipo de sustrato y planta a mantener. Para el caso del control de acceso, mediante la interfaz de usuario, se puede dar de alta/baja tarjetas para permitir accesos, se soportan hasta diez. El diagrama de bloques (figura 3-2) tiene un sutil cambio, y se encuentra en la bidireccionalidad entre la etapa de control y la plataforma *IoT*.

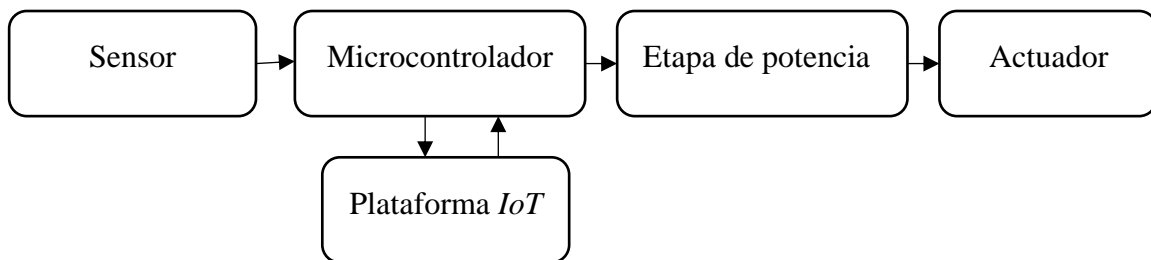


Figura 3-2 Diagrama de bloques del módulo riego automático de sustrato y apertura de puerta mediante *RFID*.

Siendo:

- Sensor: para el módulo de riego automático de sustrato un higrómetro capacitivo, mientras que para el módulo de apertura de puerta mediante *RFID*, un módulo lector de tarjetas *RFID* mfrc522.
- Microcontrolador: Para los módulos asignados al diagrama a bloques de la figura 3-2, es la tarjeta ESP32.
- Plataforma *IoT*: Se continúa utilizando la plataforma de *Blynk*, donde el usuario observa e interactúa con las variables del sistema. Para el módulo de riego automático de sustrato, es posible observar un valor orientativo de la humedad actual, además de establecer el nivel de humedad deseada. Mientras que en el módulo de apertura de puerta mediante *RFID*, se pueden modificar los usuarios admitidos además de brindar acceso.
- Etapa de potencia: acople de circuito de control con el actuador. Para el caso del módulo de riego automático de planta, consta de un optoacoplador, un transistor BC547, un relevador, un diodo en paralelo y en inversa a la bobina del relevador y capacitores electrolíticos en paralelo con la motobomba y en paralelo con la fuente. Mientras que el módulo de apertura de puerta está constituido por un transistor conmutador BC547 y un relevador.
- Actuador: motobomba sumergible de 12 Vdc con flujo de 80 l/h a 120 l/h y cerradura eléctrica marca Phillips de 12 Vdc y consumo nominal de 1.5 A.

3.2 Desarrollo de circuitos de control

En este trabajo se plantearon las siguientes soluciones domóticas: el riego automático de sustrato, detección de gas licuado de petróleo y cierre de válvula, así como el control de accesos mediante tarjetas *RFID*, todas ellas a través del internet de las cosas a través de la plataforma de *Blynk*; usando la tarjeta de desarrollo ESP32.

En esencia, los circuitos de control no varían mucho entre sí, pues constan de toma y análisis de lectura, y activación de un actuador, según corresponda la solución.

Para alimentar la tarjeta ESP32 es necesario suministrar 5 V por lo que se utilizó el regulador de voltaje LM7805, añadiendo un *LED* para verificar que, en efecto, está siendo energizado el módulo.

3.2.1 Módulo riego automático de sustrato

La figura 3-3 muestra el diagrama esquemático de la etapa de control del módulo de riego automático de sustrato.

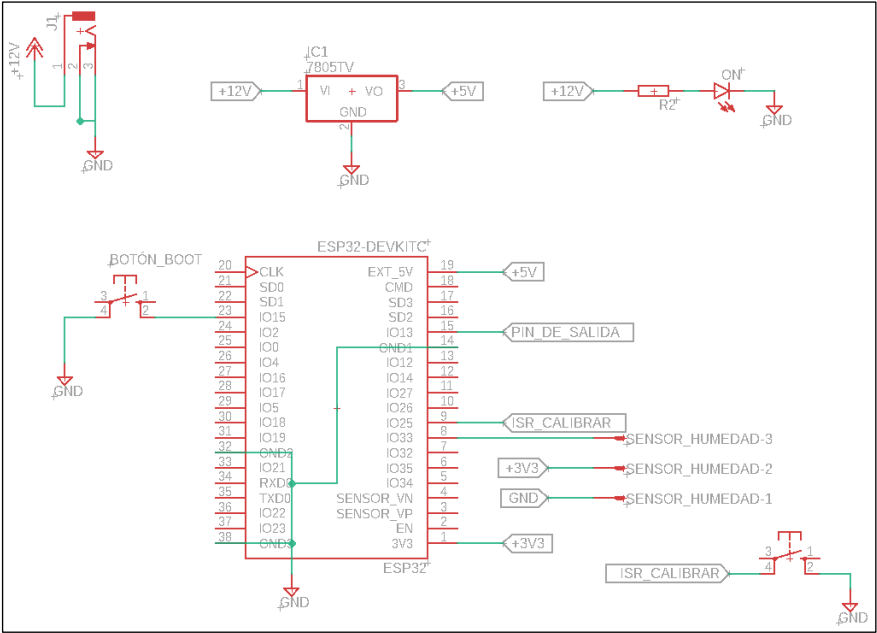


Figura 3-3. Diagrama esquemático riego automático de sustrato.

Como se puede apreciar en la Figura 3-3, el módulo de riego automático de sustrato tiene en el pin GPIO 15 el botón de *boot*, por el cual es posible establecer las configuraciones de red, el pin GPIO 24 se encuentra conectado a un botón llamado *ISR_CALIBRAR*, el pin GPIO 25 está conectado a la señal del higrómetro capacitivo v1.2 y las terminales restantes pertenecen a la alimentación de este; finalmente, el pin GPIO 13 corresponde a la señal de salida dirigida a la etapa de potencia.

3.2.2 Módulo detección de gas licuado de petróleo (GLP) y cierre de válvula

La figura 3-4 muestra el diagrama esquemático de la etapa de control del módulo de detección de gas licuado de petróleo y cierre de válvula.

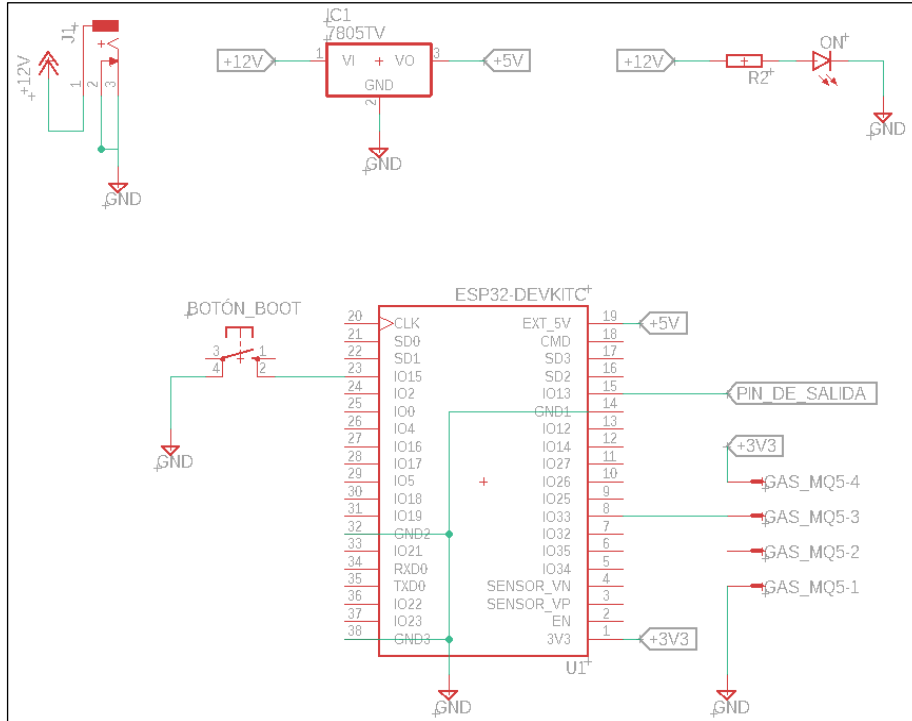


Figura 3-4. Diagrama esquemático detección de gas licuado de petróleo y cierre de válvula.

En la figura 3-4 se puede observar que en el pin GPIO 15 se encuentra conectado el botón de *boot*, mismo que es usado para establecer la configuración de red, en el pin GPIO 33 se encuentra la señal emitida por el sensor de gas MQ5, y los pines restantes corresponden a la alimentación de este; finalmente, en pin GPIO 13 se encuentra la señal de disparo para activar la etapa de potencia.

3.2.3 Módulo apertura de puerta mediante RFID

La figura 3-5 muestra el diagrama esquemático de la etapa de control del módulo de apertura de puerta mediante *RFID*.

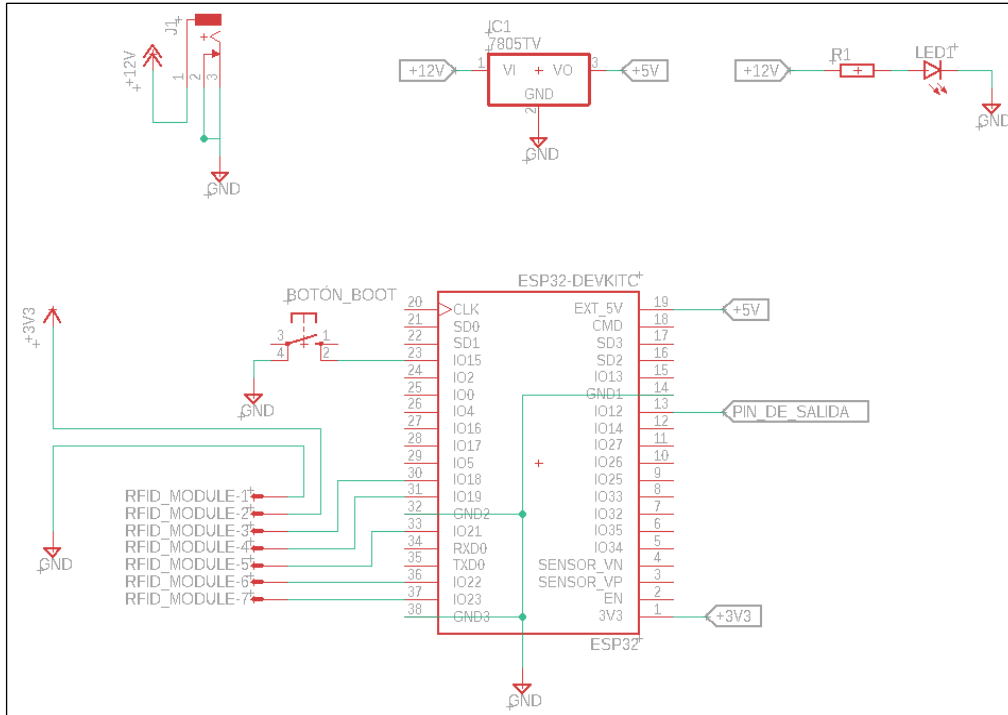


Figura 3-5. Diagrama esquemático apertura de puerta mediante RFID.

La figura 3-5 muestra que en el pin GPIO 15 es conectado el botón de *boot* para establecer la configuración de red, los pines GPIO 18, 19, 21, 22, 23 son conectados al módulo *RFID*, donde:

- GPIO 18: Pin SCK del módulo *RFID*.
- GPIO 19: Pin MISO del módulo *RFID*.
- GPIO 21: Pin SDA del módulo *RFID*.
- GPIO 22: Pin RST del módulo *RFID*.
- GPIO 23: Pin MOSI del módulo *RFID*.

Finalmente, el pin GPIO 12 es la señal de salida que activa la etapa de potencia.

3.2.4 Etapa de potencia

Los módulos riego automático de sustrato, y detección de gas y cierre de válvula comparten el circuito para la etapa de potencia, siendo este el presentado en la figura 3-6.

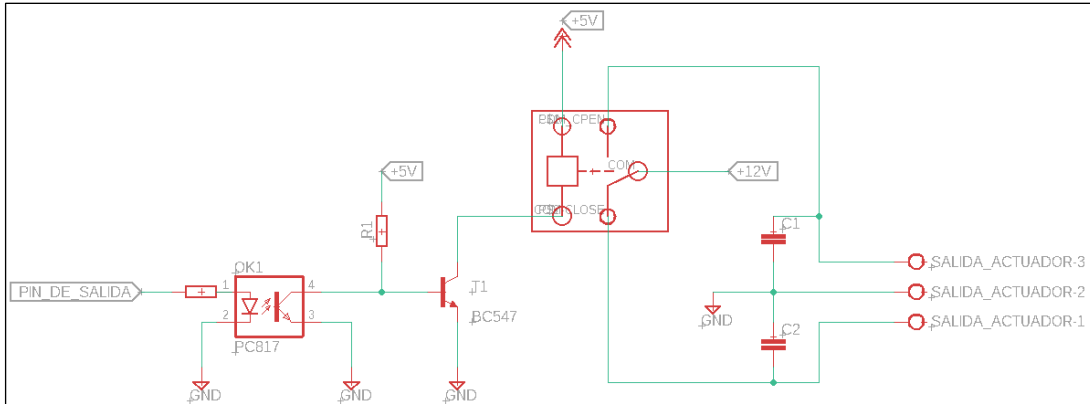


Figura 3-6. Etapa de potencia para módulo de riego automático de sustrato, y detección de gas licuado de petróleo y cierre de válvula.

Constando de una resistencia de $330\ \Omega$ que limita la corriente en el *LED* del optoacoplador PC817c, un transistor BC547 fungiendo como conmutador activando la bobina de un relevador. Finalmente, en las terminales normalmente abierto y normalmente cerrado es colocado un capacitor en paralelo como filtro para eliminar el ruido que las cargas inductivas tendían a introducir al circuito de control.

Por otra parte, la etapa de potencia del módulo apertura de puerta mediante *RFID* es presentado en la figura 3-7.

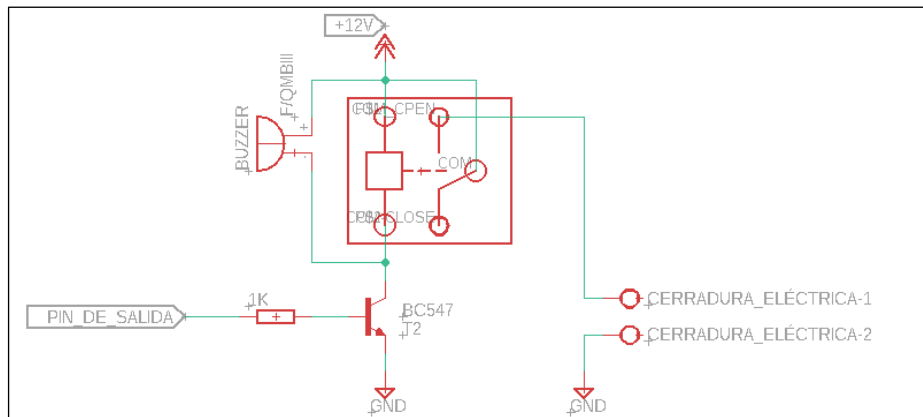


Figura 3-7. Etapa de potencia para módulo de apertura de puerta mediante *RFID*.

Ocurre que, el pin de salida va conectado a una resistencia hacia la base del transistor BC547 para poder activar la bobina del relevador cuando se señal es enviada. En paralelo a la bobina es conectado un *buzzer*. Finalmente, la carga es conectada en la terminal normalmente abierta del relevador.

3.3 Algoritmos, diagramas de flujo y codificación

3.3.1 Riego automático de sustrato

Se plantea el riego automático de sustrato con la tarjeta ESP32 y el higrómetro capacitivo v1.2.

El funcionamiento se basa en que el usuario seleccione una planta para automatizar su riego, humedezca el sustrato a un nivel deseado, posteriormente, insertar el higrómetro en el sustrato, cuidando que cubra hasta el nivel indicado del mismo sensor, en ese momento, a través de una aplicación diseñada en *Blynk*, pulsar el botón de calibrar – o bien, en el botón físico destinado al mismo fin- y entonces el sustrato tendrá siempre el valor deseado de humedad, censando cada segundo que esto ocurra.

La figura 3-8 muestra el diagrama de flujo de la maceta inteligente.

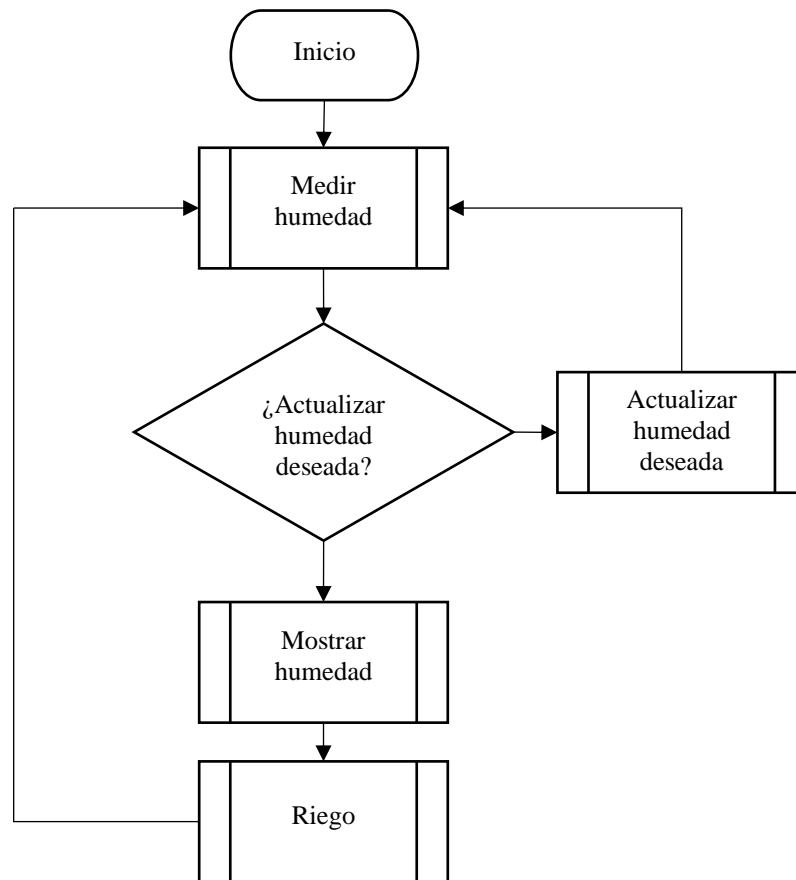


Figura 3-8. Diagrama de flujo riego automático de sustrato.

Se inicializa la configuración para ocupar la plantilla de *Blynk*, previamente diseñada.

```

1  #define BLYNK_TEMPLATE_ID "TMPLI1MfuCkZ"
2  #define BLYNK_DEVICE_NAME "PHumedadSustratoMed"
3
4  #define BLYNK_FIRMWARE_VERSION "0.1.0"
5
6  #define BLYNK_PRINT Serial
7  //#define BLYNK_DEBUG
8
9  #define APP_DEBUG
10
11 #define USE_ESP32S2_DEV_KIT
12
13 #include "BlynkEdgent.h"
  
```


Además, se incluye la biblioteca de Preferences para almacenar el valor de la humedad deseada en la memoria *flash* y se inicia con la palabra *referencia*.

```
14 #include <Preferences.h>
15 Preferences referencia;
```

De igual manera, se declaran y se establece el valor inicial de las variables globales a ocupar en el programa.

```
16 int humCali;
17 int humLectura;
18 int temporal = 0;
19 int auxMedir = 0;
20 int auxCalibrar = 0;
21 bool bCalibrar = 0;
22 bool bCalibrar2 = 0;
23 bool bandera = 0;
```

Una vez declaradas las variables globales, sigue la función *setup()* donde se indica el modo de operación de los pines, se inicializa la comunicación serial, se indica que una variable será guardada en la memoria *flash* y se inicializa la biblioteca de *Blynk*.

```
void setup()
26 {
27   pinMode(13, OUTPUT); //Salida a bomba
28   pinMode(25, INPUT_PULLUP); //Boton de calibrar, ISR
29
30   attachInterrupt(25, interrupcion, FALLING);
31   Serial.begin(115200);
32   delay(100);
33
34   referencia.begin("riegoAutomatico", false);
35   humCali = referencia.getInt("humCali", 0);
36
37   Blynk.virtualWrite(V4, humCali);
38
39   BlynkEdgent.begin();
40 }
```

Enseguida, inicia la función *loop()* donde se indica que la función de Blynk Edgent será ejecutada, posteriormente, es llamada la función *medirHumedad()* y verifica si alguno de los dos botones de calibrar ha sido pulsado, de ser así, le asigna valor de 1 a la bandera. Consecuentemente, pregunta que caso se ha cumplido, si la bandera tiene un valor de 0, entonces la función *mostrarHumedad()* es invocada, en el caso de que la bandera tenga valor de 1, manda a llamar a la función *actualizarHumedadDeseada()* y finalmente, ejecuta la función *riego()*.

```
42 void loop()
43 {
44   BlynkEdgent.run();
45
46   medirHumedad();
47
48   if (bCalibrar == 1 || bCalibrar2 == 1)
```

```

49  {
50    bandera = 1;
51  }
52
53  switch (bandera)
54  {
55    case 0:
56      mostrarHumedad();
57      break;
58    case 1:
59      actualizarHumedadDeseada();
60      break;
61  }
62  riego();
63 }

```

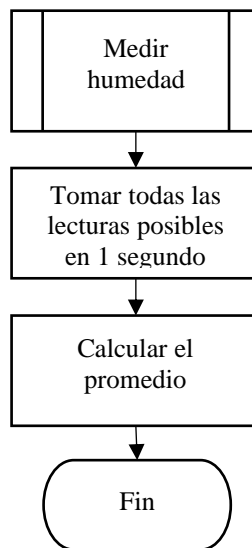


Figura 3-9 Diagrama de flujo de la función medir humedad.

La función *medirHumedad()*, cuyo diagrama de flujo es mostrado en la figura 3-9, se encarga de tomar todas las mediciones de humedad que pueda en 1 segundo, redimensionando la variable cuyo valor máximo es de 4000, y mínimo de 800, a valores de 100 a 0. Una vez tomadas todas las lecturas, obtiene el promedio y es guardado en una variable global.

```

65 void medirHumedad()
66 {
67   int contador = 0;
68
69   //Realiza todas las mediciones que pueda en 1 seg, y luego obtiene
el promedio
70   unsigned long tempRefCont = millis();
71   while ((millis() - tempRefCont) <= 1000)
72   {
73     temporal = temporal + map(analogRead(33), 800, 4000, 100, 0);
74     contador++;
75   }
76   Blynk.virtualWrite(V1, "EN LÍNEA");
77   temporal = temporal / contador;

```

78}

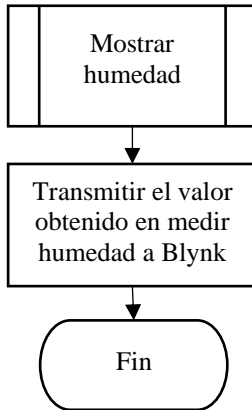


Figura 3-10 Diagrama de flujo de la función mostrar humedad.

La figura 3-10 muestra el diagrama de flujo de la función *mostrarHumedad()*, la cual se encarga de transmitir el valor de humedad actual a *Blynk* cada vez que este cambia.

```
81 void mostrarHumedad()  
82 {  
83   humLectura = temporal;  
84  
85   if (humLectura != auxMedir)  
86   {  
87     Blynk.virtualWrite(V3, humLectura);  
88     auxMedir = humLectura;  
89   }  
90 }
```

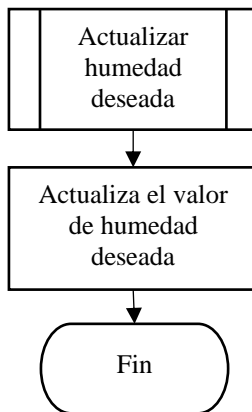


Figura 3-11 Diagrama de flujo de la función actualizar humedad deseada.

La función *actualizarHumedadDeseada()*(figura 3-11) guarda en la memoria flash el valor de la humedad actual en una variable distinta, además de transmitirlo a *Blynk* cada vez que este cambia, finalmente, reinicia el estado de la bandera y los botones de calibrar.

```
93 void actualizarHumedadDeseada()  
94 {  
95   humCali = temporal;  
96  
97   referencia.putInt("humCali", humCali);
```

```

98  referencia.end();
99
100 if (humCali != auxCalibrar)
101 {
102   Blynk.virtualWrite(V4, humCali);
103   auxCalibrar = humCali;
104 }
105
106 Blynk.virtualWrite(V0, 0);
107 bCalibrar = 0;
108 bCalibrar2 = 0;
109 bandera = 0;
110}

```

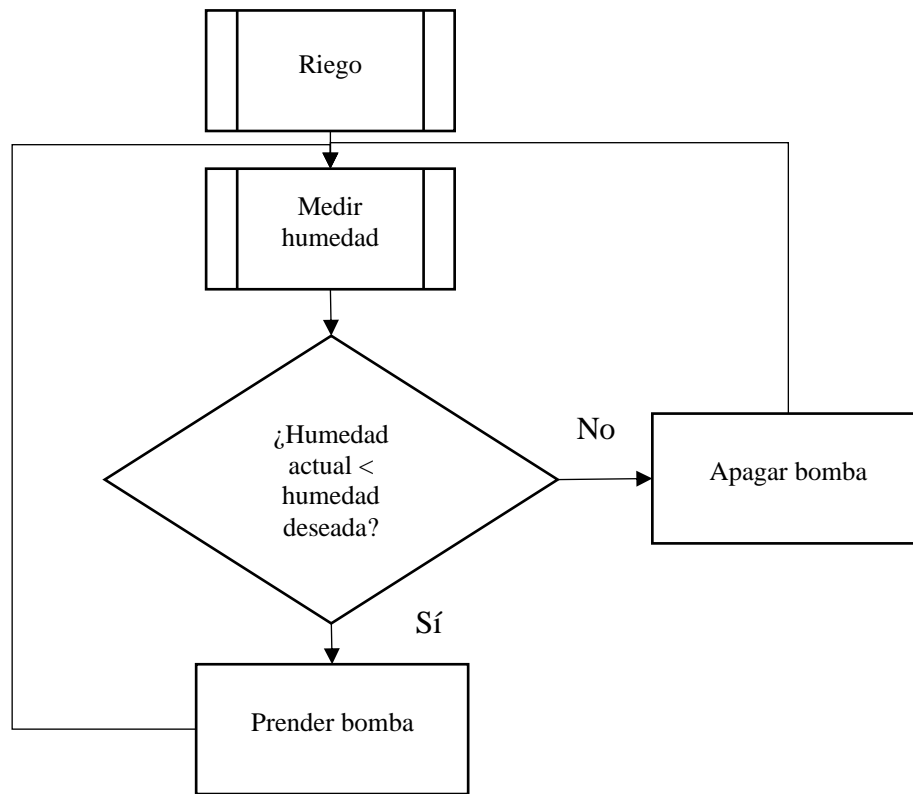


Figura 3-12 Diagrama de flujo de la función riego.

La figura 3-12 muestra la función *riego()*, la cual se encarga de evaluar la diferencia entre el valor de humedad actual contra el valor de humedad deseada. Si el valor de humedad actual es 10 unidades menor que la humedad deseada, entonces se activa la bomba y se transmite un mensaje de estado de riego a *Blynk*. En caso contrario, el valor de humedad actual es 10 unidades mayor que la humedad deseada, la bomba se apaga y se transmite un mensaje de estado de riego a *Blynk*.

```

111 void riego ()
112 {
113   if ( humLectura <= (humCali - 10))
114   {
115     digitalWrite(13, HIGH);

```

```

116   Blynk.virtualWrite(V2, "EN RIEGO");
117 }
118 else if ( humLectura >= (humCali + 10))
119 {
120   digitalWrite(13, LOW);
121   Blynk.virtualWrite(V2, "HUMEDAD ADECUADA");
122 }
123}

```

La función *interrupción()* cambia el estado del botón de calibrar

```

125void interrupcion ()
126{
127   bCalibrar2 = 1;
128}

```

Finalmente, la función *BLYNK_WRITE()* obtiene el valor del botón virtual de la plantilla creada y lo asigna a una variable.

```

130BLYNK_WRITE (V0)
131{
132   bCalibrar = param.asInt ();
133}
134

```

3.3.2 Detección de gas licuado de petróleo (GLP) y cierre de válvula

Se plantea la toma de lectura de gas licuado de petróleo en una sala de cocina, en caso de que se detecte una lectura por encima de valores permitidos, empezar un contador -en caso de que haya sido una lectura causada por una perturbación externa y no propia por detección de gas LP- y avisar al usuario de una probable fuga de gas. En caso de que hayan transcurrido 10 segundos y las lecturas sigan por encima del umbral, notificar al usuario con una alerta roja y cerrar válvula de gas. Cuando la lectura indique un valor menor al umbral establecido, volver a abrir válvula de gas. En caso de que ocurran 3 alertas rojas, entonces mantener cerrada la válvula de gas hasta que se restablezca el sistema de manera manual.

La figura 3-13 muestra el diagrama de flujo de detección de gas licuado de petróleo y cierre de válvula.

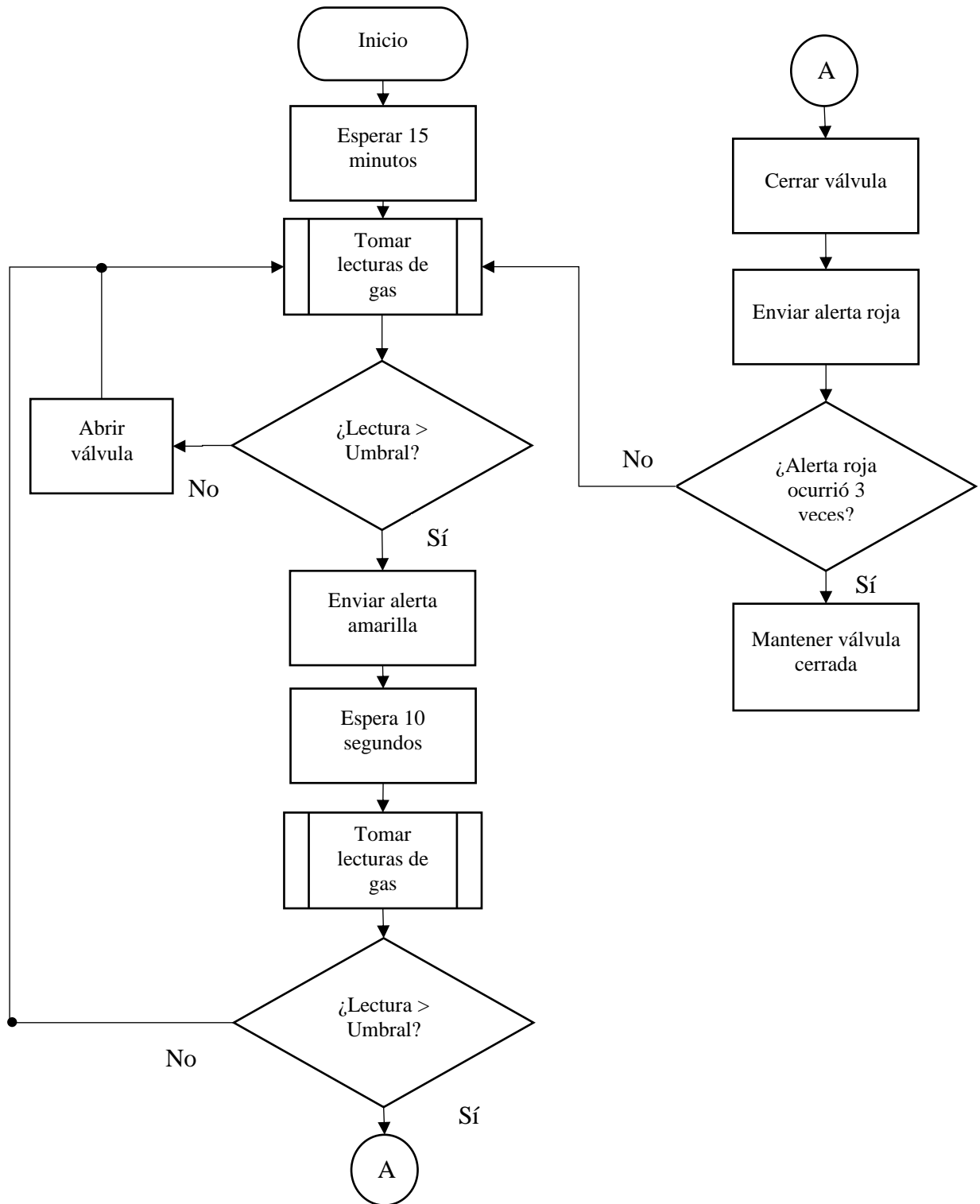


Figura 3-13 Diagrama de flujo de detección de gas licuado de petróleo y cierre de válvula.

En primera instancia, se colocan las credenciales de la plantilla previamente diseñada en la plataforma de *Blynk*, y seguido de ello se invoca la librería de *Blynk Edgent*.

```
1 #define BLYNK_TEMPLATE_ID "TMPLkHDP5J1"
```

```

2 #define BLYNK_DEVICE_NAME "PGas"
3 #define BLYNK_FIRMWARE_VERSION "0.1.0"
4 #define BLYNK_PRINT Serial
5 #define APP_DEBUG
6 #define USE_ESP32S2_DEV_KIT
7 #include "BlynkEdgent.h"

```

Después, se crean las variables globales de la línea 8 a la línea 10, iniciando con temporizadores, lecturas de gas, y terminando con banderas (para calentar el sensor, y para volver a un punto de partida)

```

8 unsigned long tempRefSUGas = 0;
9 unsigned long tempRefCalentar = 0;
10 unsigned long tempRef = 0;
11 int lecturaGas;
12 int contadorAlertas = 0;
13 int lecturaTx;
14 bool banderaCalentar = 0;
15 bool SUGas = 0;

```

Se continua con la función de *setup()*, donde se establece que el pin GPIO 13 es de salida, el programa se queda 15 minutos en un loop por única vez con el fin de calentar el sensor MQ5 y se indica que la función *Blynk Edget* es iniciada.

```

16 void setup()
17 {
18   Serial.begin(115200);
19   delay(100);
20   pinMode(13, OUTPUT);
21   calentarSensor();
22   BlynkEdgent.begin();
23 }

```

Seguido, se tiene la función *loop()* donde se indica que la función de *Blynk Edgent* es ejecutada, así como la función de medir gas, checar fuga, controlar fuga y cierre definitivo de válvula.

```

24 void loop()
25 {
26   BlynkEdgent.run();
27   medicionGas();
28   checarFuga();
29   controlarFuga();
30   cierreValvulaDefinitivo();
31 }

```

La función *calentarSensor()* se encarga de por única vez quedarse 15 minutos en un loop vacío, esto con el fin de brindar el tiempo necesario para que el sensor de gas MQ5 provea lecturas fiables. De acuerdo con el fabricante, un tiempo de precalentamiento adecuado es entre 10 y 15 minutos. Además, esto se logra con la función *millis()*, debido a que de acuerdo con la documentación de *Blynk*¹, *delay()* es incompatible con las funciones de la plataforma *IoT*.

```

33 void calentarSensor()
34 {

```

¹ <https://docs.blynk.io/en/legacy-platform/legacy-articles/keep-your-void-loop-clean>

```

35  if (banderaCalentar == 0)
36  {
37      tempRefCalentar = millis();
38      while ((millis() - tempRefCalentar) <= 900000) //Tiempo para
calentar el sensor. De acuerdo con el fabricante ...
39      {
40      }
41      banderaCalentar = 1;
42  }
43  }

```

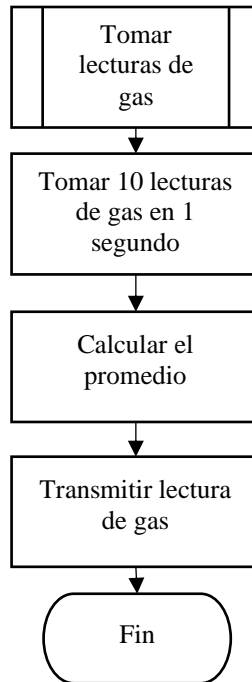


Figura 3-14 Diagrama de flujo de la función encargada de tomar lecturas de gas (medición gas).

La función medición gas (figura 3-14) se encarga calcular el promedio de la toma de 10 lecturas obtenidas en 1 segundo, además de transmitir este valor redimensionado de manera porcentual, haciendo hincapié en que es meramente orientativo.

```

45 void medicionGas ()
46 {
47     unsigned long tempRefCont = millis();
48     while ((millis() - tempRefCont) <= 1000)
49     {
50         for (int i = 0; i < 9; i++ )
51         {
52             lecturaGas = lecturaGas + analogRead(33);
53             tempRef = millis();
54         }
55         lecturaGas = lecturaGas / 10;
56     }
57     lecturaTx = map(lecturaGas, 50, 2000, 0, 100);
58     Blynk.virtualWrite(V2, lecturaTx);
59     Blynk.virtualWrite(V3, contadorAlertas);
60 }

```


La función *checharFuga()* se encarga de comparar las lecturas obtenidas previamente con el valor umbral establecido tras una serie de pruebas, siendo este valor de 500. En cuanto la lectura supere este valor, se envía una alerta al usuario indicando que hay una posible fuga de gas, y se inicia un contador que se detalla en la siguiente función.

Si la lectura es menor al umbral establecido, entonces se envía una alerta indicando que todo está en orden, manteniendo el pin de la electroválvula cerrado.

```
62 void checharFuga()
63 {
64   if ((lecturaGas > 500) && SUGas == 0)
65   {
66     tempRefsUGas = millis();
67     SUGas = 1;
68     Blynk.virtualWrite(V1, "Posible fuga de gas");
69     Blynk.setProperty(V0, "color", "#f1c232");
70   }
71   else if (lecturaGas < 500)
72   {
73     SUGas = 0;
74     digitalWrite(13, LOW);
75     Blynk.virtualWrite(V1, "Todo en orden");
76     Blynk.setProperty(V0, "color", "#8fce00");
77   }
78 }
```

La función *controlarFuga()* evalúa si desde que se registró una lectura por encima del umbral han pasado 10 segundos (debido a que podría tratarse de una perturbación externa momentánea), y de ser así, entonces refresca el temporizador, envía una alerta indicando que existe una fuga de gas, cierra la electroválvula, y el *LED* indicador de la interfaz gráfica se torna a rojo, además de incrementar el contador de alertas.

```
80 void controlarFuga()
81 {
82   if (((millis() - tempRefsUGas) > 10000) && SUGas == 1)
83   {
84     tempRefsUGas = millis();
85     Blynk.virtualWrite(V1, "FUGA DE GAS");
86     Blynk.setProperty(V0, "color", "#f44336");
87     digitalWrite(13, HIGH);
88     contadorAlertas++;
89     SUGas = 0;
90   }
91 }
```

Si el contador de alertas llega a 3, se trata de una fuga de gas que no es momentánea, por lo que se indica al usuario mediante una alerta que existe una fuga de gas y que debería checar tuberías, y el programa entra en un *loop* indeterminado. En este punto, la única manera de restablecer el sistema es manualmente mediante el botón de *reset* incorporado en la tarjeta ESP32.

```
93 void cierreValvulaDefinitivo()
94 {
95   if (contadorAlertas > 3)
96   {
97     Blynk.virtualWrite(V1, "FUGA DE GAS, CHECAR TUBERÍAS");
```

```
98     Blynk.setProperty(V0, "color", "#f44336");
99     while (1)
100    {
101    }
102 }
```

3.3.3 Apertura de puerta usando módulo RFID

Se plantea el control de accesos de una vivienda mediante la apertura de una puerta usando módulo *RFID*. El funcionamiento consta en la lectura de tarjetas *RFID*. Si el usuario está registrado en el sistema, entonces activa al actuador, en este caso, una chapa eléctrica, de otro modo, solo indica que el usuario no es conocido y no activa al actuador. Además, a través de la interfaz gráfica en *Blynk*, es posible añadir o quitar hasta 10 usuarios, siendo responsabilidad del usuario administrar los mismos; es decir, cuidar la sobreescritura de tarjetas. También, es posible la apertura de puerta con un botón virtual dispuesto en la *GUI* de *Blynk*. La figura 3-15 muestra el diagrama de flujo del módulo apertura de puerta usando módulo *RFID*.

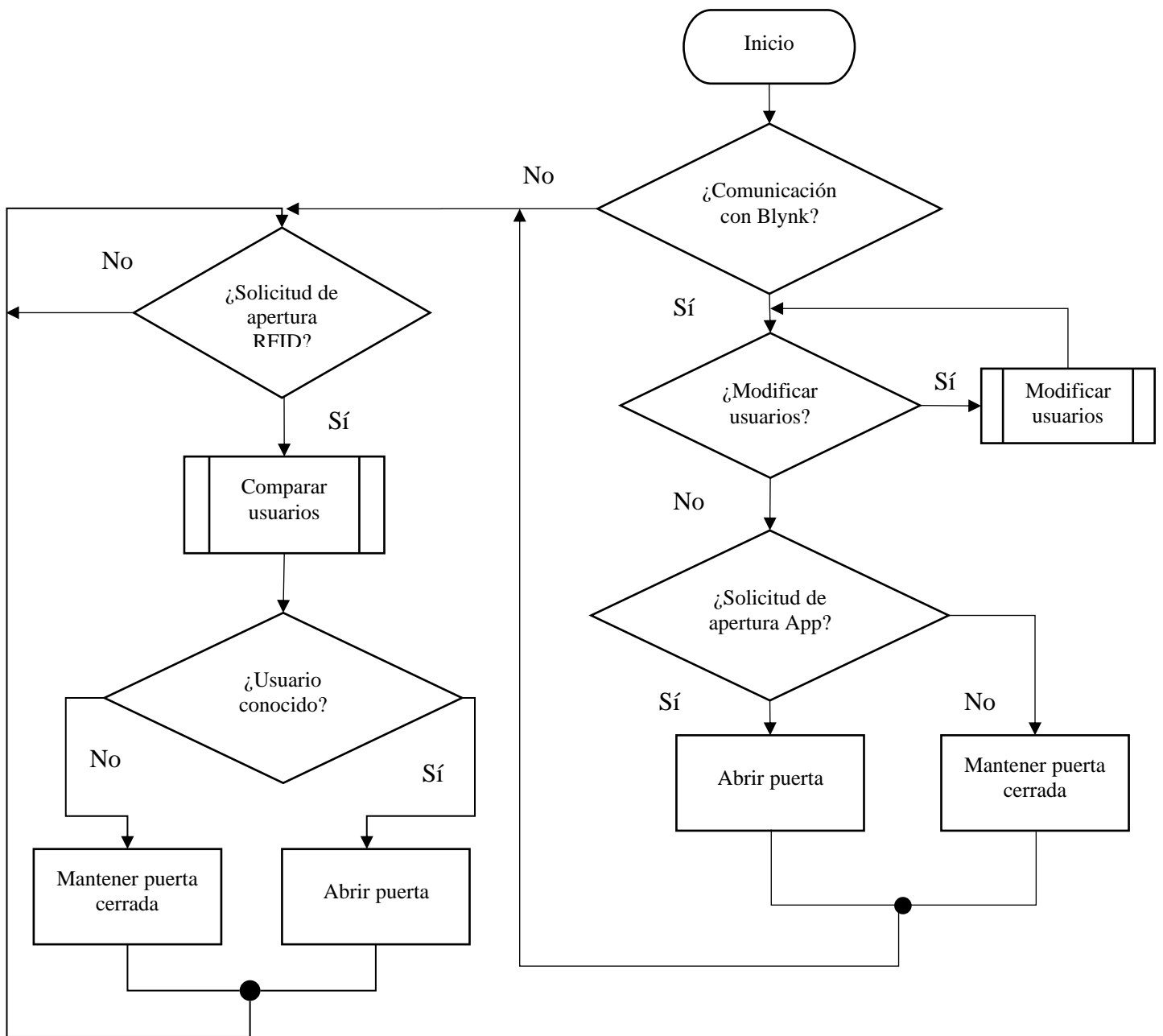


Figura 3-15. Diagrama de flujo del módulo apertura de puerta mediante RFID.

Por lo que, en primera instancia se inicializan las configuraciones de la plantilla en *Blynk*, y las bibliotecas del módulo *RFID*, así como de la memoria *EEPROM*.

```

1 #define BLYNK_TEMPLATE_ID "TMPLXKDu5xz2"
2 #define BLYNK_DEVICE_NAME "pControlAcceso"
3 #define BLYNK_FIRMWARE_VERSION "0.1.0"
4 #define BLYNK_PRINT Serial
5 #define APP_DEBUG
6 #include "BlynkEdgent.h"

```

```

7 #include <SPI.h>
8 #include <MFRC522.h>
9 #include <EEPROM.h>

```

Enseguida, se configuran los pines del módulo *RFID* y se indica que de *Blynk* se ocupa el *widget* del monitor *LCD*.

```

10 #define SS_PIN 21
11 #define RST_PIN 22
12 MFRC522 mfrc522(SS_PIN, RST_PIN);
13 WidgetLCD lcd(V0);

```

Y a continuación, se declaran las variables globales a ocupar en el programa.

```

14 byte LecturaID[4];
15 byte Usuario[10][4];
16 byte ranuraUsuario;
17 bool agregar;
18 bool quitar;
19 bool apertura;
20 bool bandera;

```

Ahora, en la función *setup()* se inician las funciones, tanto del módulo *RFID*, espacios de memoria a ocupar en la *EEPROM*, *Blynk Edgent*, y el pin *GPIO 12* como salida.

```

21 void setup()
22 {
23   Serial.begin(115200);
24   delay(100);
25   SPI.begin();
26   mfrc522.PCD_Init();
27   EEPROM.begin(40);
28   pinMode(12, OUTPUT);
29   BlynkEdgent.begin();
30 }

```

Después, en el *loop()*, se invocan las funciones de *Blynk Edgent* para contar con las funcionalidades de *Blynk*. Y se procede a conocer si hay alguna tarjeta presente, y de ser así, si es posible obtener sus datos. Si esta premisa se cumple, se obtiene el *ID* de la tarjeta presentada y se almacena en el vector *LecturaID[i]*.

```

31 void loop()
32 {
33   BlynkEdgent.run();
34   if (!mfrc522.PICC_IsNewCardPresent())
35   {
36     BlynkEdgent.run();
37     return;
38   }
39   if (!mfrc522.PICC_ReadCardSerial())
40   {
41     BlynkEdgent.run();
42     return;
43   }
44   for (byte i = 0; i < 4; i++)
45   {
46     LecturaID[i] = mfrc522.uid.uidByte[i];
47   }

```

Antes de realizar la comparación entre usuarios registrados contra usuario recién leído, se establece una bandera de estado con valor inicial de 0, cuyo valor lógico proporciona la apertura de puerta.

```
48  bandera = 0;
49  for (int j = 0; j < 10; j++)
50  {
51      if (comparar(LecturaID, Usuario[j]))
52      {
53          bandera = 1;
54          break;
55      }
56  }
57  if (bandera == 1)
58  {
59      lcd.clear();
60      lcd.print(0, 0, "  Bienvenido");
61      digitalWrite(12, HIGH);
62      delay(1000);
63  }
64  else
65  {
66      lcd.clear();
67      lcd.print(0, 0, " No te conozco");
68      delay(2000);
69  }
```

Finalmente, dentro del mismo *loop()* se establece que el valor del pin que permite la apertura de puerta este cerrado y finaliza la comunicación con el módulo *RFID*.

```
70  lcd.print(0, 0, "Presente tarjeta");
71  digitalWrite(12, LOW);
72  mfrc522.PICC_HaltA();
73 }
```

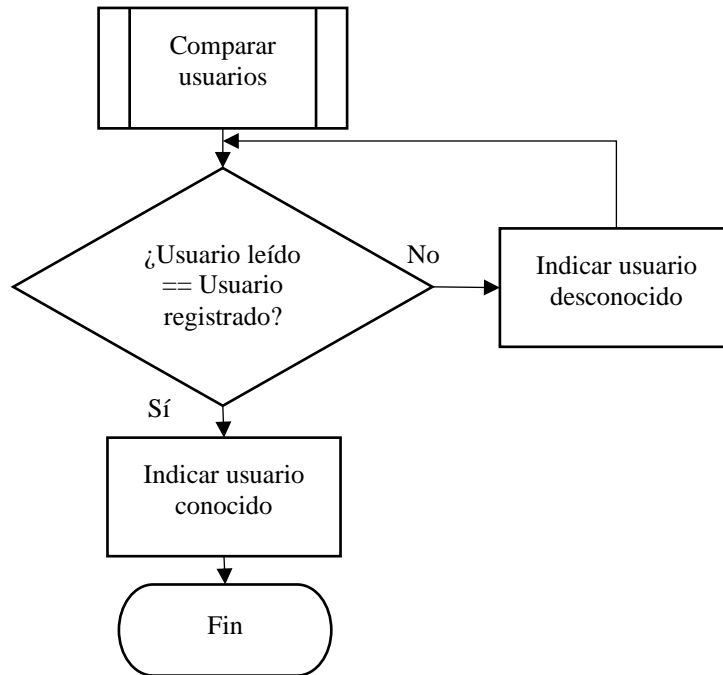


Figura 3-16. Diagrama de flujo de la función *comparar usuarios*.

Observando la línea 51 de código, existe una función llamada *comparar* (diagrama de flujo indicado en la figura 3-16), cuyos parámetros de entrada son *usuario recién leído* y *usuario registrado*. Esta función se encarga de cotejar los cuatro bytes de *ID* en el arreglo de la tarjeta recién leída contra la matriz de usuarios registrados. Si los cuatro *bytes* de la primera fila de la matriz de usuarios registrados no cotejan con los cuatro *bytes* del vector de usuario recién leído, entonces compara el usuario recién leído con la siguiente fila de la matriz de usuarios registrados. Devuelve verdadero cuando la comparación es exitosa.

```

74 bool comparar(byte laLectura[], byte elUsuario[])
75 {
76     for (byte j = 0; j < 4; j++)
77     {
78         if (laLectura[j] != elUsuario[j])
79         {
80             return (false);
81         }
82     }
83     return (true);
84 }
  
```

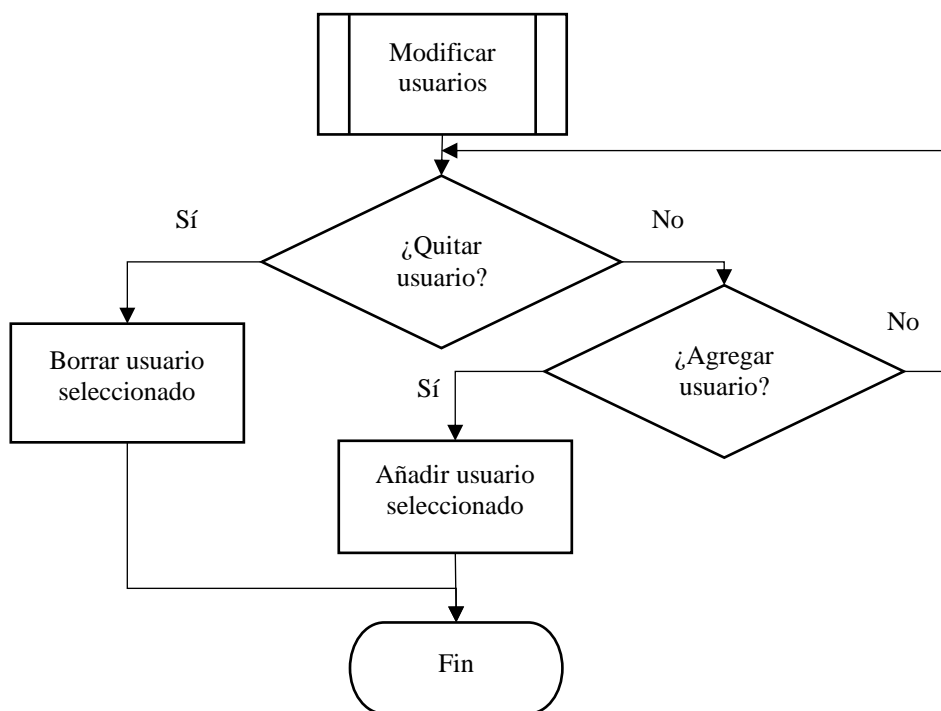


Figura 3-17. Diagrama de flujo de la función modificar usuarios.

Como se mencionó previamente, es posible modificar hasta 10 usuarios. Para ello, se implementó un botón virtual en la plataforma de *Blynk* denominado agregar, el cual permite almacenar en la ranura de usuario seleccionada la tarjeta presentada. La figura 3-17 enseña el diagrama de flujo de la función modificar usuarios.

```

86 BLYNK_WRITE(V1)
87 {
88   agregar = param.asInt();
89   if (agregar == 1)
90   {
91     for (int j = 0; j < 4; j++)
92     {
93       Usuario[ranuraUsuario - 1][j] = LecturaID[j];
94       EEPROM.write(j + (4 * (ranuraUsuario - 1)),
Usuario[ranuraUsuario - 1][j]);
95       EEPROM.commit();
96     }
97   }
98 }
  
```

Así, también es posible eliminar tarjetas. Basta con seleccionar la ranura de usuario a quitar y pulsar el botón virtual implementado en la plataforma de *Blynk* predispuesto para esta acción.

```

100 BLYNK_WRITE(V2)
101 {
102   quitar = param.asInt();
103   if (quitar == 1)
104   {
  
```

```

105     for (byte i = 0; i < 4; i++)
106     {
107         Usuario[ranuraUsuario - 1][i] = 0;
108         EEPROM.write(i + (4 * (ranuraUsuario - 1)),
Usuario[ranuraUsuario - 1][i]);
109         EEPROM.commit();
110     }
111 }
112 }

```

De igual manera se puede proporcionar acceso al presionar el botón virtual implementado en la plataforma de *Blynk* designado para esta tarea.

```

114 BLYNK_WRITE (V3)
115 {
116     apertura = param.asInt();
117     if (apertura)
118     {
119         lcd.clear();
120         lcd.print(0, 0, " Bienvenido");
121         digitalWrite(12, apertura);
122         delay(2000);
123         lcd.clear();
124         lcd.print(0, 0, "Presente tarjeta");
125         digitalWrite(12, LOW);
126     }
127 }

```

Finalmente, en el diseño de la interfaz gráfica se implementó un *widget* de *slider*, con el cual se puede seleccionar la ranura de usuario a modificar. Este se invoca por medio del pin virtual V4 de la plataforma de *Blynk*.

```

129 BLYNK_WRITE (V4)
130 {
131     ranuraUsuario = param.asInt();
132 }

```

3.4 Aplicación en *Blynk*

A través de la plataforma de *Blynk* es posible observar e interactuar (de ser el caso) con lo que ocurre en los módulos desarrollados.

3.4.1 Riego automático de sustrato

Como se puede observar en la Figura 3-, la *GUI* móvil para el módulo riego automático de sustrato se tienen cinco elementos o *widgets*. En la parte superior se encuentra el indicador de humedad actual en el sustrato, este valor es obtenido mediante el higrómetro capacitivo y se actualiza cada segundo. Es de mencionar que es un valor meramente orientativo, debido a que no representa de manera objetiva el valor real de humedad.

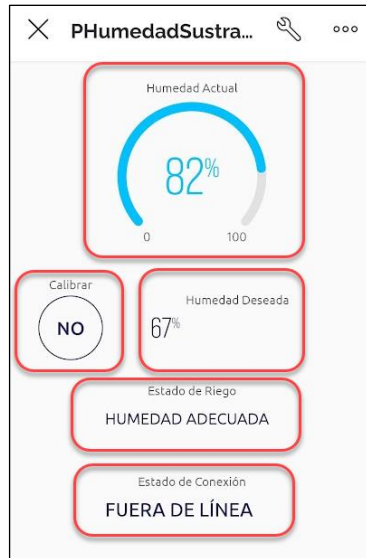


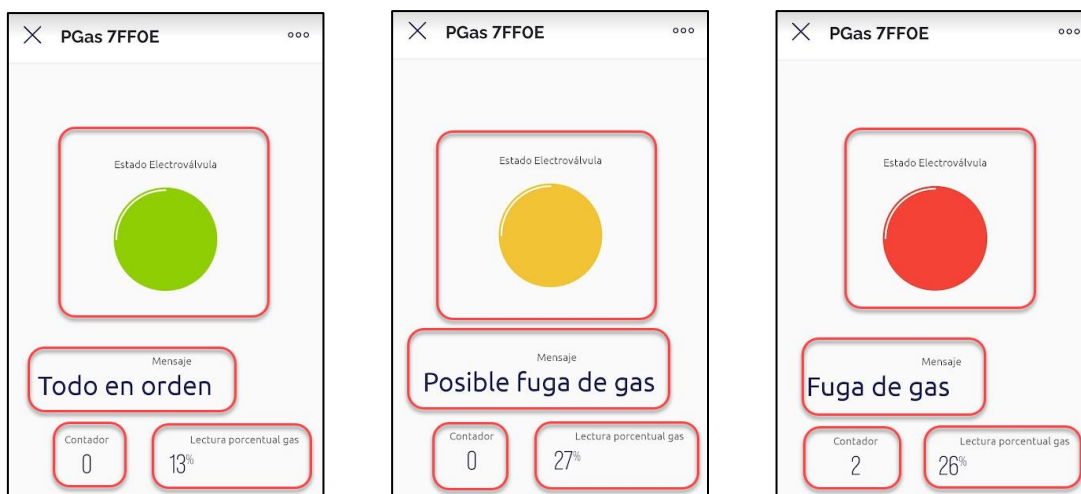
Figura 3-18 GUI del módulo riego automático de sustrato.

Después, se encuentra un botón virtual denominado calibrar, ya que, recordando el algoritmo del programa, el usuario humedece el sustrato a un nivel deseado, coloca el higrómetro capacitivo en el sustrato hasta donde indica la marca, y en ese momento pulsa el botón calibrar para que el módulo guarde dicho valor, y el sistema mantiene la humedad en un rango de $\pm 10\%$, respecto a ese valor.

También cuenta con un texto indicador que muestra el estado de riego, teniendo dos estados, humedad adecuada y en riego.

Finalmente, se añade un texto para conocer el estado de conexión respecto a la red del módulo.

3.4.2 Detección de gas licuado de petróleo (GLP) y cierre de válvula



a.

b.

c.

Figura 3-19. GUI del módulo detección de gas licuado de petróleo y cierre de válvula.

Como se puede observar en la Figura 3-19, para el caso del módulo de detección de gas licuado de petróleo y cierre de válvula se incorporó un *LED* indicador a manera de semáforo, donde:

- En la figura 3-19 a. el *LED* verde indica que la lectura registrada por el sensor de gas MQ5 se encuentra por debajo del umbral establecido,
- En la figura 3-19 b. el *LED* ámbar indica que el sensor de gas registró una lectura por encima del umbral establecido, además, si esta lectura permanece por encima del umbral por 10 segundos, entonces incrementa el contador en uno.
- Finalmente, en la figura 3-19 c. el *LED* rojo indica que el contador ha registrado que el evento en figura 3-19 b. ha ocurrido tres veces, por lo que señala que existe una fuga de gas.

Es de recalcar que el valor mostrado en el *widget* de lectura porcentual de gas es orientativo; es decir, no representa de manera objetiva la concentración de gas existente leída por el sensor.

3.4.3 Apertura de puerta mediante RFID



a.

b.

c.

Figura 3-20 GUI del módulo apertura de puerta mediante RFID.

Como se puede apreciar en la figura 3-20, la interfaz gráfica de la plataforma en *Blynk* está conformada por un monitor *LCD*, un *slider* con 10 posiciones (cada una para un usuario diferente), tres botones cuyas funciones son agregar y quitar usuarios, así como permitir el acceso mediante el botón abrir.

El monitor *LCD* enseña el mensaje mostrado en los tres posibles escenarios, siendo que:

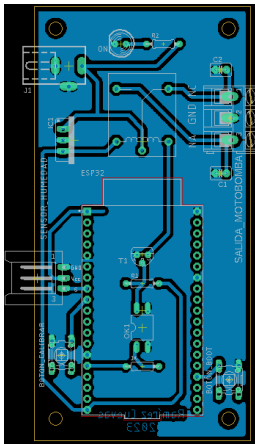
- En la figura 3-20 a. “Presente tarjeta” es el texto por defecto.
- En la figura 3-20 b. “Bienvenido” indica que la tarjeta presentada coincide con algún usuario previamente registrado, o bien, que el botón virtual “abrir” fue accionado.

- En la figura 3-20 c. “Acceso denegado” señala que la tarjeta presentada no está registrada en el módulo y, por ende, no permite el acceso.

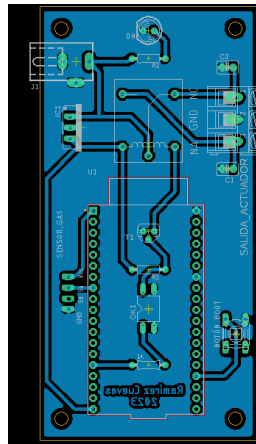
3.5 Fabricación de PCB

Con el fin de mantener los circuitos desarrollados libres de posibles falsos contactos, fueron diseñadas y fabricadas placas de circuito impreso con el software Eagle 9.6.2 con el método de planchado y revelado con cloruro férrico.

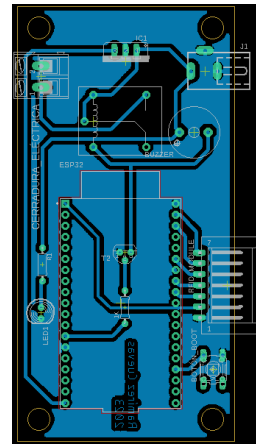
La figura 3-21 muestra los *layouts* (pistas) de las placas de circuito impreso de los módulos desarrollados, haciendo hincapié en que sus dimensiones son de 5 x 10 cm.



a. Diseño de PCB riego automático de sustrato.



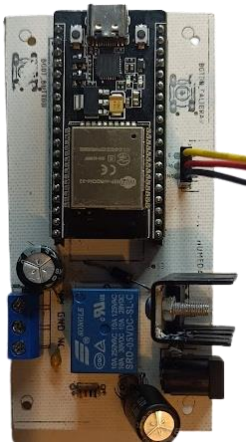
b. Diseño de PCB detección de gas licuado de petróleo y cierre de válvula.



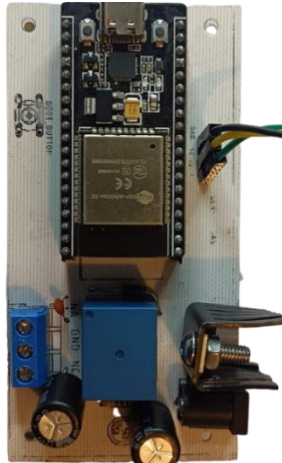
c. Diseño de PCB apertura de puerta mediante RFID.

Figura 3-21 Layouts de PCB de los módulos desarrollados.

Por otra parte, la figura 3-22 muestra la parte superior de las placas de circuito impreso de los módulos desarrollados.



a. PCB riego automático de sustrato.



b. PCB detección de gas licuado de petróleo y cierre de válvula.



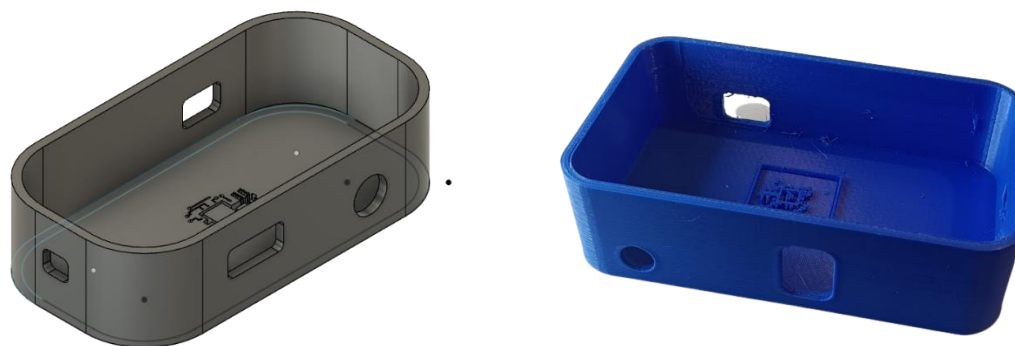
c. PCB apertura de puerta mediante RFID.

Figura 3-22 Parte superior de PCB de los módulos desarrollados.

3.6 Diseño de gabinete

Con el fin de mantener las placas de circuito impreso protegidas de factores externos como polvo, suciedad, humedad y golpes, se optó por diseñar en el software Fusion 360 y manufacturar con una impresora 3D unos gabinetes.

Por la disposición de los componentes en las placas de circuito impreso se tuvo la posibilidad de que los tres módulos compartieran el mismo diseño, el cual puede ser contemplado en la figura 3-23.



a. Gabinete visto desde Fusion 360.

b. Gabinete impreso.

Figura 3-23 Gabinete para los módulos desarrollados.

3.7 Sistema de captación de energía solar como fuente de alimentación

Finalmente, los tres módulos desarrollados fueron alimentados con un sistema de captación de energía solar de potencia nominal de 30 W y 12 Vdc cuyas dimensiones son 80 cm x 40 cm x 2.5 cm, eficiencia celular de 19.5% y regulador de carga que soporta corriente de hasta 10 A de la marca TOP SOLAR; y una batería ácido plomo de 12 V 4 Ah marca Steren.

El diagrama de bloques del sistema es mostrado en la figura 3-24.

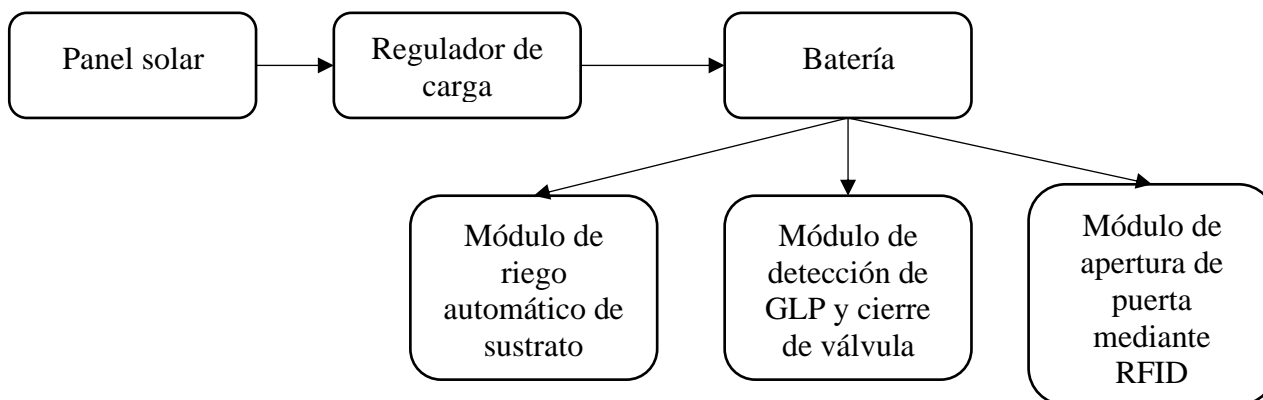


Figura 3-24. Diagrama de bloques de la alimentación del sistema.

Además, el montaje del sistema se muestra en la figura 3-25.

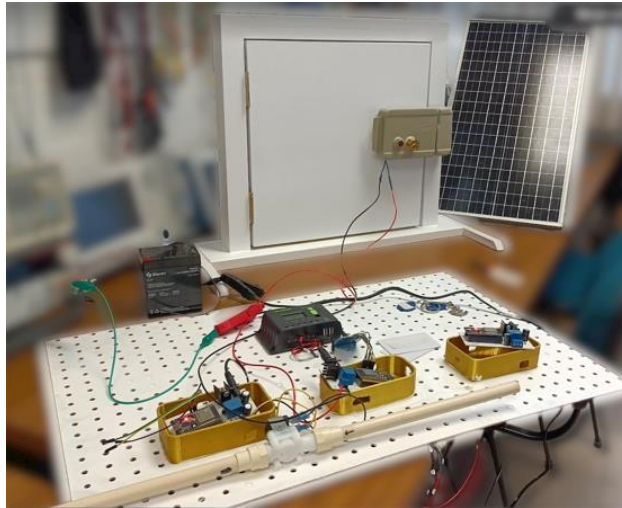


Figura 3-25. Sistema montado.

El sistema durante el mes de mayo de 2023 situado en Tecámec, Estado de México, presentó una autonomía de 8 horas alimentado únicamente con el panel solar, siempre y cuando las condiciones del tiempo fueran mayormente soleadas, de otra manera, el sistema de captación de energía solar no es capaz de proveer la potencia suficiente para el funcionamiento de los módulos desarrollados.

En adición, la autonomía del sistema en horas de poca radiación solar es de 2 horas, contemplando diez aperturas de puerta y diez riegos de sustrato, con la válvula eléctrica funcionando de manera constante.

Siendo así que, para la temporada de primavera en Tecámec, Estado de México, se estima que la autonomía de los módulos desarrollados alimentados con un sistema de captación de energía solar es cercana a las 10 horas.

Capítulo 4. Pruebas y resultados

En este capítulo se describen las pruebas realizadas y los resultados obtenidos. En particular, se detallan los experimentos modo *offline* y *online* a los módulos descritos.

4.1 Pruebas fuera de línea

4.1.1 Riego automático de sustrato

Para la implementación del módulo riego automático de sustrato fue necesario conocer los valores que toma el higrómetro insertado hasta donde la marca indica, en los siguientes escenarios:

- Sustrato seco,
- Sustrato recién humedecido,
- Sustrato sobre saturado de agua,
- Sensor al aire y
- Sensor en agua.

Esto con el fin de conocer el intervalo de valores que toma en cada escenario y así poder tomar decisiones sobre cuando regar el sustrato.

Además, es necesario colocar el higrómetro capacitivo hasta donde se cubra la marca en rojo indicada en la figura 4.1 a. La figura 4.1 b muestra como quedó el higrómetro insertado en el sustrato



Figura 4.1 Manera de colocar el higrómetro capacitivo dentro del sustrato.

La tabla 4.1 muestra el intervalo de valores obtenidos en 1 minuto en los escenarios propuestos.

Tabla 4.1 Valores mínimos y máximos registrados por el higrómetro en un intervalo de 1 minuto

	Sustrato seco	Sustrato recién humedecido	Sustrato sobre saturado de agua	Sensor al aire	Sensor en agua
Valor mínimo	1200	1500	1990	3400	1500
Valor máximo	1300	1520	2300	3700	1580

Además, con las lecturas obtenidas, se concluyó que el valor mínimo y máximo que puede tomar la variable de humedad en el ADC del microcontrolador es entre 1200 y 3700, decidiendo dar un margen a la lectura inferior reduciéndola a 800 y a la lectura mayor aumentándola a 4000, para no llegar a los valores límite. Entonces, se decidió redimensionar la lectura de humedad de 100 a 0, debido a que el funcionamiento del higrómetro capacitivo es con lógica inversa.

De igual manera, se observó que, tomando una lectura de humedad cada segundo, en el mismo escenario, los valores tendían a ser oscilantes, por lo que se optó por tomar todas las lecturas que el microcontrolador fuera capaz de tomar en 1 segundo y promediarlas. De esta manera se logra eliminar los valores mínimos y máximos ocasionados por perturbaciones externas, teniendo como resultado lecturas menos variantes en el tiempo.

El código que realiza lo descrito previamente se muestra a continuación.

```

1 void setup()
2 {
3   Serial.begin(115200);
4 }
5 void loop()
6 {
7   int lectura;
8   int contador;
9   unsigned long tempRefCont = millis();
10  while ((millis() - tempRefCont) <= 1000)
11  {
12    lectura = lectura + map(analogRead(33), 800, 4000, 100, 0);
13    contador ++;
14  }
15  lectura = lectura / contador;
16  Serial.println(lectura);
17}

```

Se optó por tomar como valor representativo la media debido a su facilidad de ser calculada.

Una vez estabilizadas las lecturas, se implementó un botón cuyo funcionamiento consta en que, cuando es pulsado, guarda en la memoria *EEPROM* del microcontrolador el valor leído del higrómetro en ese instante. Esto con el fin de brindarle un intervalo de humedad

deseada. Es decir, una vez pulsado el botón, la humedad leída por el higrómetro deberá estar en un rango de $h \pm 10\%$. Si el valor de humedad leída en el higrómetro disminuye 10% respecto al valor almacenado, entonces encender motobomba y apagar hasta que el valor de humedad sea el valor almacenado aumentado en 10%, logrando así una especie de histéresis.

Inicialmente un *LED* simulaba el actuador y funcionaba correctamente. Al momento de ser intercambiado por una motobomba y como etapa de potencia únicamente un transistor y un relevador, se observó la motobomba introducía ruido al sistema tras ser activada. Este hecho fue corregido al implementar a la etapa de potencia un optoacoplador, así como un diodo en inversa y en paralelo a la bobina del relevador y unos capacitores en paralelo al actuador y en paralelo a la fuente.

Debido a que los módulos desarrollados constan de activar y desactivar un actuador, se sigue la misma idea para los sistemas restantes, i.e., la implementación en la etapa de potencia de optoacoplador, diodo en paralelo e inversa al relevador, capacitores electrolíticos en paralelo a la fuente y al actuador.

La figura 4.2 muestra el módulo de riego automático de sustrato.



Figura 4.2 Módulo de riego automático de sustrato.

El siguiente experimento consiste en conocer el tiempo en que la humedad leída por el higrómetro en el sustrato llega al límite inferior, y con intervalos de observación de cada 12 horas y con la planta (lengua de suegra) situada en un ambiente de sombra dentro de casa y una humedad deseada de 75% se tuvieron los resultados plasmados en la tabla 4-2.

Tabla 4.2 Registros de humedad de sustrato para planta lengua de suegra.

Día	Intervalo	Humedad registrada
1	7:00 – 19:00	80%
2	19:00 - 7:00	76%
3	7:00 - 19:00	71%
4	19:00 – 7:00	65%

Es de mencionar que la maceta tiene una capacidad volumétrica de cerca de 4 litros, y que, para aumentar el nivel de humedad en el sustrato, la bomba sumergible bombeó alrededor de 450 ml en 43 segundos. Este hecho permite concluir que, para la planta elegida (lengua de suegra), con el volumen de la maceta (4 litros) y el caudal de la bomba elegida, el riego automático en el sustrato tiene lugar cada 4 días y tiene un tiempo de duración de aproximadamente 43 segundos. Resulta evidente mencionar que, al cambiar de planta, maceta, sustrato y bomba, se modifica la frecuencia y periodo del riego.

4.1.2 Detección de gas licuado de petróleo y cierre de válvula

Para la puesta en funcionamiento del módulo de detección de gas licuado de petróleo y cierre de válvula fue necesario conocer el tiempo de calentamiento del sensor MQ5 para dar lecturas fiables, por lo que, contactando con el fabricante, se sugiere un tiempo de precalentamiento de 15 minutos.

Además, de manera experimental en un cuarto de cocina de 4 x 4 m se hicieron pruebas de lectura de gas licuado de petróleo (GLP) para conocer que valores entregaba el sensor en los siguientes escenarios:

- Sin presencia de gas,
- Cocina cerrada sin gas,
- Cocina cerrada junto a estufa sin gas,
- Cocina con ventilación sin gas,
- Cocina con ventilación junto a estufa sin gas,
- Cocina cerrada cocinando,
- Cocina cerrada cocinando junto a estufa,
- Cocina cerrada con válvula de gas abierta,
- Cocina cerrada con válvula de gas abierta junto a estufa,
- Cocina con ventilación con gas,
- Cocina con ventilación con gas junto a estufa.

Para este caso, se tomó en cuenta el valor promedio de 10 lecturas de gas en cada segundo, con el fin de tener valores menos variantes en el tiempo.

La tabla 4.3 muestra los valores mínimos y máximos registrados para cada escenario mencionado en un tiempo de 1 minuto para cada uno.

Tabla 4.3 Valores mínimos y máximos registrados por el sensor de gas MQ5 en un intervalo de 1 minuto.

Escenario	Entorno	Valor mínimo	Valor máximo
Sin gas	Normal	170	180
	Cocina cerrada	200	205
	Cocina cerrada junto a estufa	230	240
	Cocina con ventilación	220	235

	Cocina con ventilación junto a estufa	225	240
Con gas	Cocina cerrada	420	500 en adelante.
	Cocina cerrada junto a estufa	435	500 en adelante.
	Cocina cerrada cocinando	230	240
	Cocina cerrada cocinando junto a estufa	250	260
	Cocina con ventilación	380	500 en adelante.
	Cocina con ventilación junto a estufa	380	500 en adelante

Por lo que se concluyó que, tras calentar el sensor MQ5 15 minutos -tiempo sugerido por el fabricante-, en presencia de gas licuado de petróleo, las lecturas tienden a aumentar, y un valor característico de presencia de gas es a partir de 420 en el ADC.

El código para conocer los valores que entrega el sensor en los diferentes escenarios descritos previamente se muestra a continuación.

```

1 void setup()
2 {
3   Serial.begin(115200);
4 }
5 void loop()
6 {
7   int lectura;
8   unsigned long tempRefCont = millis();
9   while ((millis() - tempRefCont) <= 1000)
10  {
11    for ( int i = 0; i < 9; i++)
12    {
13      lectura = lectura + analogRead(33);
14    }
15    lectura = lectura / 10;
16  }
17  Serial.println(lectura);
18}

```

Entonces, se propuso que el sistema este censando cada segundo, y si encuentra una lectura superior a 500 comenzar con un contador de tiempo, ya que podría tratarse de una perturbación externa. Si tras 10 segundos -tiempo razonable para confirmar que es una

posible fuga de gas- la lectura sigue siendo superior a 500, entonces se debe cerrar la válvula. En cuanto la lectura baje del umbral establecido, volver a abrir la válvula.

Al código previamente descrito se le añaden las líneas que se muestran a continuación para lograr el acometido.

```
21 bool SUGas = 0;
22 unsigned long tempRefSUGas = 0;
23 if (lectura > 500)
24 {
25
26     SUGas = 1;
27     tempRefSUGas = millis();
28     Serial.println("Posible fuga de gas");
29 }
30 else
31 {
32     SUGas = 0;
33     digitalWrite(13, LOW);
34     Serial.println("Todo en orden");
35 }
36
37 if (((millis() - tempRefSUGas) > 10000) && SUGas == 1)
38 {
39     tempRefSUGas = millis();
40     digitalWrite(13, HIGH);
41     Serial.println("FUGA DE GAS");
42 }
```

No obstante, el hecho de que la fuga de gas licuado de petróleo este presente y la válvula se encuentre en estado intermitente, presenta inseguridad al usuario, por lo que se decidió que, si este evento se repetía tres veces, entonces entrar en un *loop* indeterminado, de tal manera que el programa se reactivará solo cuando el usuario haya reparado la falla y reinicie el sistema.

Esto se logró con un contador, y añadiendo lo siguiente.

```
21 bool SUGas = 0;
22 int contadorAlertas;
23 unsigned long tempRefSUGas = 0;
24 if (lectura > 500)
25 {
26
27     SUGas = 1;
28     tempRefSUGas = millis();
29     Serial.println("Posible fuga de gas");
30 }
31 else
32 {
33     SUGas = 0;
34     digitalWrite(13, LOW);
35     Serial.println("Todo en orden");
36 }
37
38 if (((millis() - tempRefSUGas) > 10000) && SUGas == 1)
39 {
40     tempRefSUGas = millis();
```

```

41  digitalWrite(13, HIGH);
42  Serial.println("FUGA DE GAS");
43  contadorAlertas ++;
44  }
45
46  if (contadorAlertas > 3)
47  {
48  digitalWrite(13, HIGH);
49  Serial.println("FUGA DE GAS, REVISAR SISTEMA DE TUBERÍAS");
50  while (1) {
51  }
52  }
53}

```

En la línea 22 se crea una variable para conocer las veces que se emitieron alertas. Después, en la línea 43 se indica que cada vez que pase por ahí el programa, aumentar en uno el contador. Finalmente, la línea 46 evalúa si el contador ha ocurrido tres veces, y de ser así, muestra alerta al usuario y como se había descrito, entra en un *loop* inderterminado.

La figura 4.3 muestra el módulo de detección de gas licuado de petróleo y cierre de válvula.

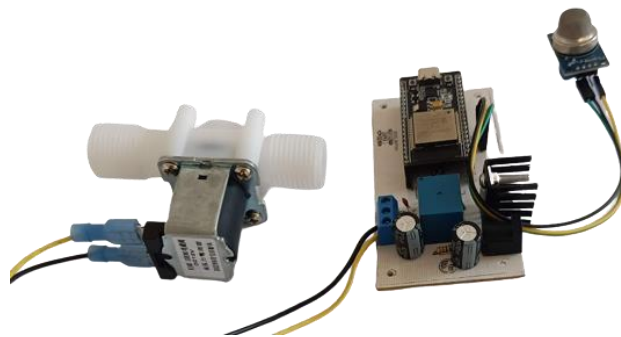


Figura 4.3 Módulo detección de gas licuado de petróleo y cierre de válvula.

4.1.3 Apertura de puerta mediante *RFID*

Para el desarrollo de este experimento fue propuesto almacenar en la memoria EEPROM 10 *ID* de tarjetas *RFID*, por lo que previamente se diseñó un programa que fuera capaz de leer y obtener los 4 *bytes* del *ID* de alguna tarjeta *RFID*. Una vez hecho esto, decidir si la tarjeta sería almacenada en la memoria. En la siguiente pasada, el programa lee y evalúa si el *ID* de la tarjeta presentada es igual a alguno de los *ID* almacenados. En primera instancia, el programa debía funcionar para tres usuarios, y una vez logrado, escalar el código para diez usuarios. La figura 4.4 muestra la propuesta de un mapa de memoria para estructurar los usuarios y después realizar una comparación.

	Byte 0	Byte 1	Byte 2	Byte 3
Usuario 1	0xA9	0x1F	0x0E	0xC1
Usuario 2	0x2F	0x9B	0x3D	0x98
Usuario 3	0x4E	0x6D	0xBA	0x78
...				
Usuario 10	0x8A	0x7F	0x1D	0x0E

	Byte 0	Byte 1	Byte 2	Byte 3
Usuario leído	0x4E	0x6D	0xBA	0x78

Figura 4-4. Mapa de memoria para almacenar y comparar usuarios.

Primero, se definen vectores para la lectura actual, y los 3 usuarios almacenados.

Después, se realizan las inicializaciones del módulo *RFID* y se establecen los espacios a ocupar en la tarjeta ESP32, tomando en cuenta que cada *ID* ocupa 4 bytes, entonces se inicializa la *EEPROM* con 12 espacios.

Seguido de ello, al acercarse alguna tarjeta al módulo *RFID* se obtiene su *ID*, y a manera de solución, si previamente fue presionado el botón de usuario 1, usuario 2 o usuario 3, el *ID* leído es guardado en este vector.

Cuando se acerque nuevamente alguna tarjeta, el programa lee y compara para saber si se trata de un *ID* conocido.

El meollo de este experimento consiste en el guardado y comparación de *ID*. Para el guardado de usuario, si se elige que la tarjeta presentada sea el usuario 1, entonces guardar los 4 bytes en el espacio 0, 1, 2, 3 de la memoria *EEPROM*. Si se elige el usuario 2, entonces guardar los 4 bytes en el espacio 4, 5, 6, 7 de la memoria *EEPROM*. Y así sucesivamente.

Al momento de comparar usuario leído con los usuarios registrados, el truco consiste en mantener estáticos los elementos del vector del usuario leído, y hacer un barrido con los usuarios registrados. De esta manera, cuando se hace la comparación usuario leído con usuario 1, si no coinciden sus respectivos *ID*, pasar a comparar con el siguiente usuario.

El código que lee el *ID* de la tarjeta y lo muestra en el monitor serial se muestra a continuación.

```

22 void loop()
23 {
24   bUser1 = digitalRead(5);
25   bUser2 = digitalRead(15);
26   bUser3 = digitalRead(4);
27   if ( ! mfrc522.PICC_IsNewCardPresent() )
28     return;
29
30   if ( ! mfrc522.PICC_ReadCardSerial() )
31     return;
32
33   Serial.println("UID actual: \t Usuario 1: \t Usuario 2: \t Usuario
34   3: \n"); // muestra texto UID:
35   for (byte i = 0; i < 4; i++)
36   {
37     LecturaUID[i] = mfrc522.uid.uidByte[i];
38     Serial.print(LecturaUID[i]);
39   }

```

```

39
40  agregarUsuario();
41
42  compararUsuarios();
43
44  mfrc522.PICC_HaltA();
45 }

```

Con la función *agregarUsuario()* es consultado si alguno de los botones para agregar usuario fue pulsado, de ser así, almacena los 4 bytes de su respectivo *ID* en la ranura de memoria determinada, recordando que el usuario 1 ocupa el espacio 0, 1, 2, 3 de la memoria *EEPROM*; el usuario 2 ocupa el espacio 4, 5, 6, 7 de la memoria *EEPROM*; y así sucesivamente.

```

51 void agregarUsuario()
52 {
53     if (bUser1 == 1)
54     {
55         for (byte i = 0; i < 4; i++)
56         {
57             Usuario1[i] = LecturaUID[i];
58             EEPROM.write(i, Usuario1[i]);
59             EEPROM.commit();
60         }
61     }
62
63     if (bUser2 == 1)
64     {
65         for (byte i = 4; i < 8; i++)
66         {
67             Usuario2[i] = LecturaUID[i - 4];
68             EEPROM.write(i, Usuario2[i]);
69             EEPROM.commit();
70         }
71     }
72
73     if (bUser3 == 1)
74     {
75         for (byte i = 8; i < 12; i++)
76         {
77             Usuario3[i] = LecturaUID[i - 8];
78             EEPROM.write(i, Usuario3[i]);
79             EEPROM.commit();
80         }
81     }
82 }

```

Como se puede apreciar, en cada ciclo for se aumenta el valor inicial de la variable de iteración, y al vector *LecturaUID* se le resta el mismo incremento del valor inicial.

Además, la instrucción `EEPROM.commit();` indica que se ha finalizado el proceso de escritura en el espacio de la memoria *EEPROM* seleccionado.

Finalmente, es necesario comparar usuario registrado contra usuario leído, lo cual se logra con el fragmento de código mostrado a continuación.

```

80 void compararUsuarios()

```

```

81 {
82   if ( (LecturaUID[0] == Usuario1[0]) && (LecturaUID[1] ==
Usuario1[1]) && (LecturaUID[2] == Usuario1[2]) && (LecturaUID[3] ==
Usuario1[3]) )
83   {
84     Serial.println("Bienvenido");
85     return;
86   }
87
88   if ( (LecturaUID[0] == Usuario2[0]) && (LecturaUID[1] ==
Usuario2[1]) && (LecturaUID[2] == Usuario2[2]) && (LecturaUID[3] ==
Usuario2[3]) )
89   {
90     Serial.println("Bienvenido");
91     return;
92   }
93
94   if ( (LecturaUID[0] == Usuario3[0]) && (LecturaUID[1] ==
Usuario3[1]) && (LecturaUID[2] == Usuario3[2]) && (LecturaUID[3] ==
Usuario3[3]) )
95   {
96     Serial.println("Bienvenido");
97     return;
98   }
99   else
100  {
101    Serial.println("Usuario no válido");
102    return;
103  }
104}

```

Que, en esencia, compara elemento a elemento los valores del usuario leído con usuario registrado. Si la primera condición no se cumple, compara al siguiente usuario, y de no coincidir con algún usuario registrado, mostrar mensaje indicando que el usuario no es válido.

El código mostrado previamente funcionó y es escalable a los 10 usuarios propuestos, no obstante, se encontró una manera más elegante de llegar al mismo resultado a través de ciclos *for* anidados y sustituir los vectores de usuario por una matriz de 4 x 10.

El siguiente fragmento de código muestra la inicialización del módulo *RFID*, la definición de la matriz usuarios de 10 x 4 y la declaración de tres botones para almacenar usuario según corresponda el botón pulsado.

```

1  #include <SPI.h>
2  #include <MFRC522.h>
3  #include <EEPROM.h>
4  #define SS_PIN 21
5  #define RST_PIN 22
6  MFRC522 mfrc522(SS_PIN, RST_PIN);
7
8  byte LecturaID[4];
9  byte Usuario[10][4];
10 bool agregar;
11 bool bandera;
12 byte ranuraUsuario;

```

```

13
14 void setup()
15 {
16   Serial.begin(115200);
17   SPI.begin();
18   mfrc522.PCD_Init();
19   EEPROM.begin(40);
20
21   bool bUser1 = digitalRead(5);
22   bool bUser2 = digitalRead(15);
23   bool bUser3 = digitalRead(4);
24   agregar = digitalRead(18);
25

```

El siguiente fragmento de código muestra que a través de una función llamada comparar cuyos parámetros LecturaID y Usuario[j], coteja el vector LecturaID con la matriz Usuario. Realiza este chequeo 10 veces -razón por la cual se creó una matriz de 10 x 4-. De cumplirse, entonces cambia el estado de la variable bandera y sale de ese fragmento.

```

49   agregar();
50   bandera = 0;
51   for (int j = 0; j < 10; j++)
52   {
53     if (comparar(LecturaID, Usuario[j]))
54     {
55       bandera = 1;
56       break;
57     }
58   }
59
60   if (bandera == 1)
61   {
62     Serial.println("Bienvenido");
63     delay(1000);
64   }
65
66   else
67   {
68     Serial.println("Usuario desconocido");
69     delay(1000);
70   }
71   mfrc522.PICC_HaltA();
72 }

```

El siguiente fragmento de código muestra la función comparar cuyos parámetros son laLectura y elUsuario.

```

74 bool comparar(byte laLectura[], byte elUsuario[])
75 {
76   for (byte j = 0; j < 4; j++)
77   {
78     if (laLectura[j] != elUsuario[j])
79     {
80       return (false);
81     }
82   }
83   return (true);

```


84 }

Con la función `comparar`, como lo indica su nombre, coteja los elementos del vector `laLectura` contra los elementos de la fila en que se encuentre el barrido de la matriz `elUsuario`, con lógica inversa, i.e., si no coinciden los elementos, entonces retorna falso, en caso contrario, devuelve valor verdadero.

Finalmente, el siguiente fragmento de código muestra la función `agregar()`.

```
86 void agregar()
87 {
88     if(bUser1){
89         ranuraUsuario = 1;
90     }
91     if (bUser2){
92         ranuraUsuario = 2;
93     }
94     if (bUser3){
95         ranuraUsuario = 3;
96     }
97
98     if (agregar == 1)
99     {
100         for (int j = 0; j < 4; j++)
101         {
102             Usuario[ranuraUsuario - 1][j] = LecturaID[j];
103             EEPROM.write(j + (4 * (ranuraUsuario - 1)),
Usuario[ranuraUsuario - 1][j]);
104             EEPROM.commit();
105         }
106     }
107 }
```

Donde, de acuerdo al botón seleccionado es guardado el usuario en los espacios de memoria que le corresponden, siendo que, para usuario 1 ocupa el espacio 0, 1, 2, 3; el usuario 2 ocupa el espacio 4,5,6,7. Y es lo que es logrado mediante la operación `Usuario[ranuraUsuario - 1][j]`. Es modificada primeramente la fila, indicando el usuario -usuario 1, 2, etc.- Mientras que en la línea 103 del código se realiza la operación para escribir en el espacio `(j + (4 * (ranuraUsuario - 1)))` el valor `Usuario[ranuraUsuario - 1][j]`.

Para el caso en que se seleccione la ranura de usuario 1 pulsando el botón `bUser1` además del botón `agregar`, la operación a realizar es:

```
102     Usuario[1 - 1][j] = LecturaID[j];
103     EEPROM.write(j + (4 * (1 - 1)), Usuario[1 - 1][j]);
```

Entonces

```
102     Usuario[0][j] = LecturaID[j];
103     EEPROM.write(j + (4 * (0)), Usuario[0][j]);
```

En consecuencia

```
102     Usuario[0][j] = LecturaID[j];
103     EEPROM.write(j + 0 ), Usuario[0][j]);
```

Finalmente

```
102     Usuario[0][j] = LecturaID[j];  
103     EEPROM.write(j, Usuario[0][j]);
```

Por lo que el microcontrolador realiza esta tarea por sí solo, de acomodar en los espacios predefinidos de memoria *EEPROM* los 4 *bytes* correspondientes a cada usuario.

El código es escalable a añadir más usuarios modificando el tamaño de la matriz y la inicialización de la memoria *EEPROM*.

La figura 4.5 muestra el módulo de apertura de puerta mediante *RFID*.

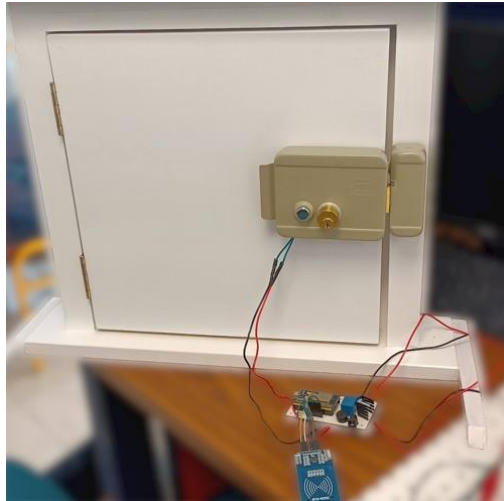


Figura 4.5 Módulo apertura de puerta mediante *RFID*.

4.2 Pruebas en línea

4.2.1 *Blynk*

Para el uso de la plataforma *Blynk* es necesario crear una cuenta en <https://blynk.io/> . Seguido de ello, debe crearse una nueva plantilla, que es donde se disponen los canales virtuales y los *widgets*. La figura 4-6 muestra la pantalla de inicio de la plataforma *Blynk* web.

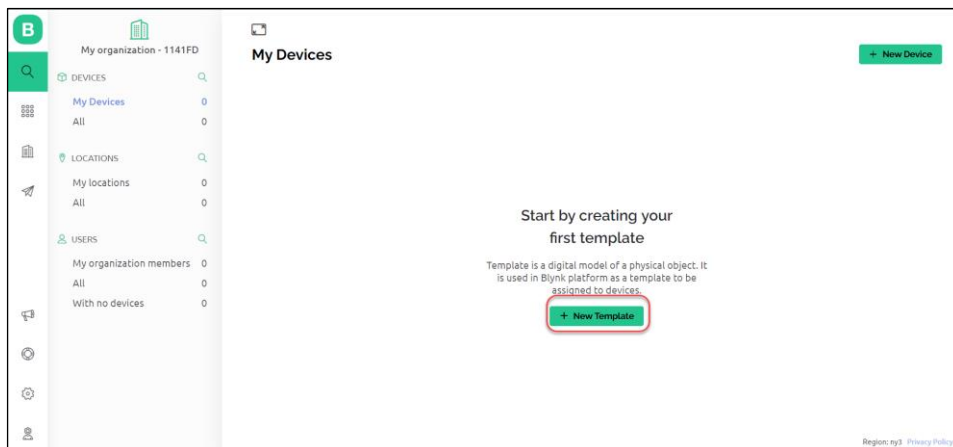


Figura 4-6. Pantalla de inicio de la plataforma *Blynk* web.

Entonces, tras dar clic en nueva plantilla, aparece una ventana para colocar nombre, microcontrolador y tipo de conexión, como lo indica la figura 4-7.

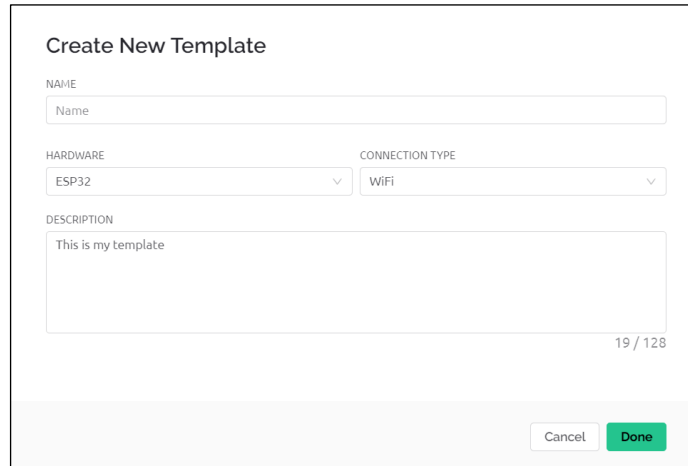


Figura 4-7. Ventana para crear una plantilla.

Al terminar de crear la plantilla, aparecen las ventanas mostradas en la figura 4-8, donde se puede visualizar información de credenciales, definir variables, crear automatizaciones y diseñar una interfaz gráfica modo web.

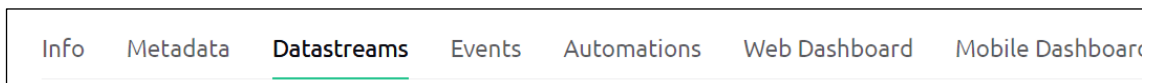


Figura 4-8. Ventanas para editar la plantilla de Blynk

La figura 4-9 muestra la ventana de información donde es posible editar el nombre de la plantilla, cambiar el microcontrolador y tipo de conexión, así como visualizar las credenciales.

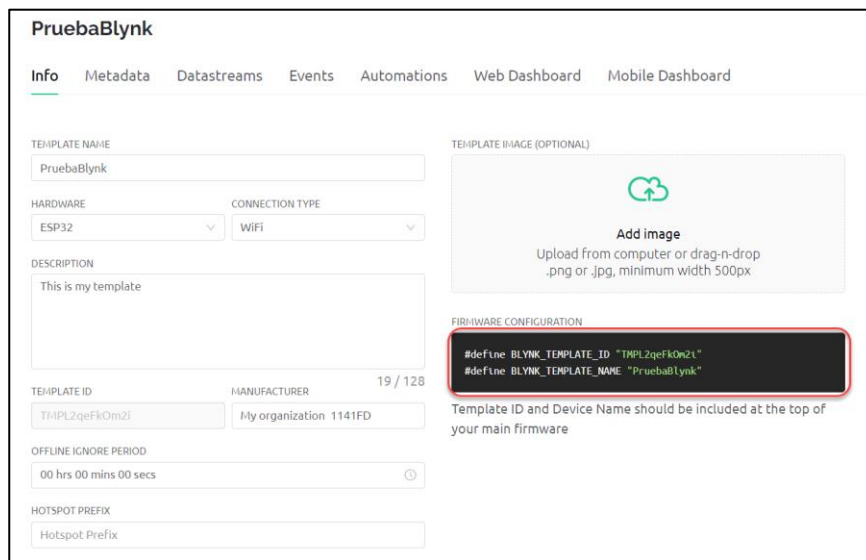


Figura 4-9. Ventana de info en la plataforma de Blynk.

Por otra parte, la figura 4-10 muestra la ventana de *datastream* la cual permite estructurar el flujo de datos de entrada y salida del microcontrolador al definir las variables a ocupar de acuerdo con la aplicación a desarrollar.

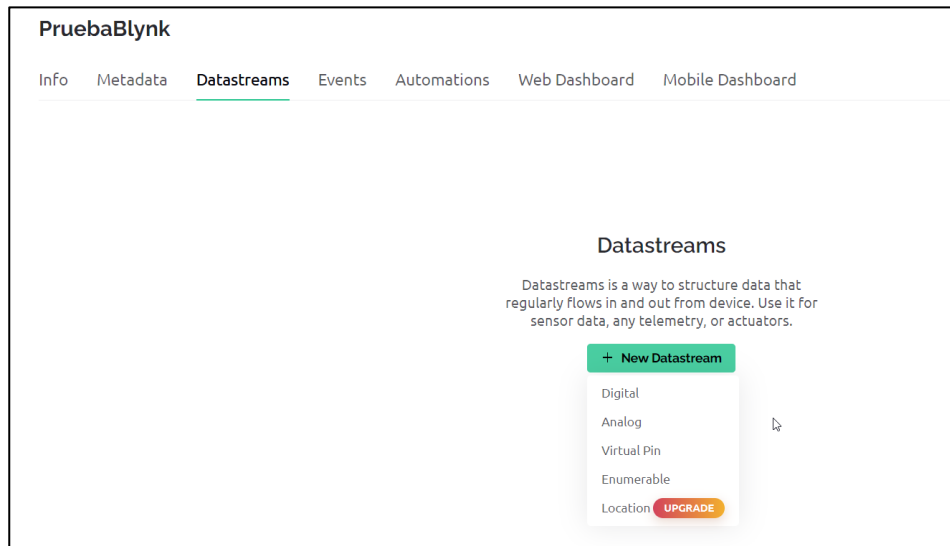


Figura 4-10. Ventana de Datastream en la plataforma de Blynk.

Así mismo, es posible diseñar una interfaz gráfica, tal como lo muestra la figura 4-11, desde la versión de escritorio, siendo que en la versión gratuita permite 10 *widjets*, algunos de ellos son: *switch*, *slider*, *LED*, etiqueta, medidor de aguja.

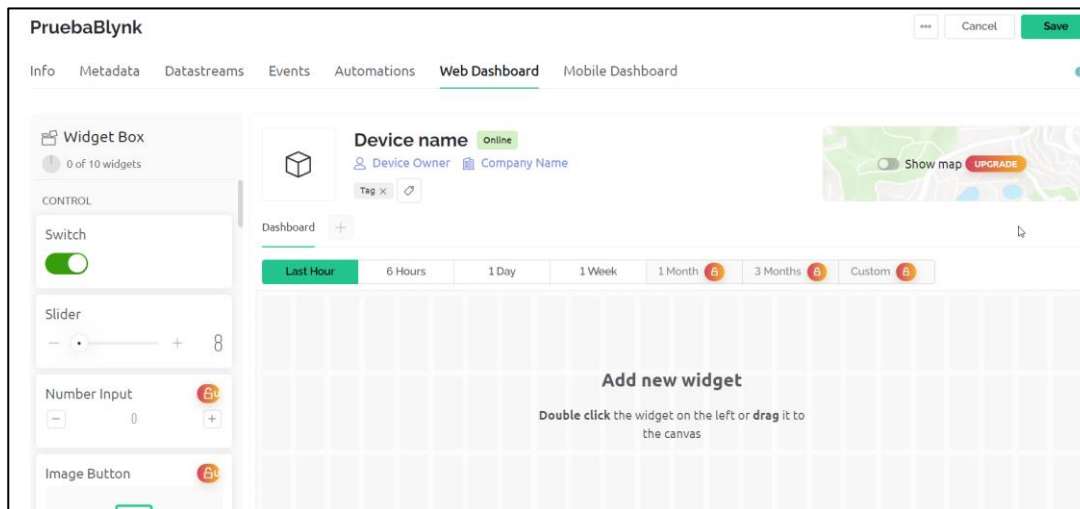


Figura 4-11. Ventana de Web Dashboard en la plataforma de Blynk.

Para migrar la aplicación desarrollada a *Blynk* basta con abrir el *Arduino IDE*, dar clic en *Herramientas*, administrar bibliotecas, buscar *Blynk* e instalar la que es desarrollada por *Volodymyr Shymanskyy*, tal como muestra la *Figura 4-12*.



Figura 4-12. Instalación de librería de Blynk en la IDE de Arduino.

Seguido de ello, dirigirse a la ventana de inicio, ejemplos, desplazar hacia abajo hasta encontrar Blynk, seleccionar *Blynk.Edgent* y finalmente seleccionar *Edgent_ESP32*, como muestra la figura 4-13.

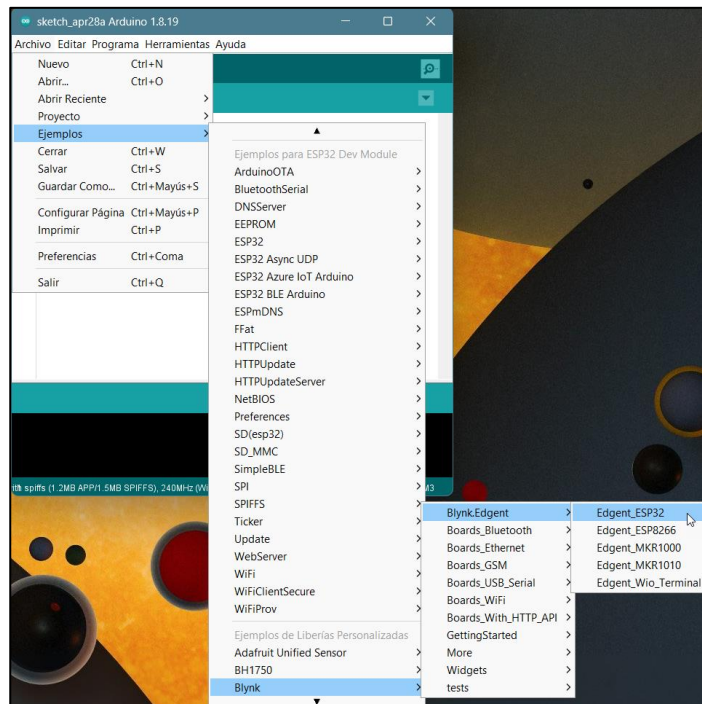


Figura 4-13. Ejemplo de Blynk Edgent para la ESP32.

Así, se abre la plantilla que muestra la figura 4-14.

```

Edgent_ESP32$ BlynkEdgent.h BlynkState.h ConfigMode.h ConfigStore.h Console.h Indicator.h OTA.h ResetButton.h Settings.h
1 // Fill-in information from your Blynk Template here
2 // #define BLYNK_TEMPLATE_ID "TMPLxxxxxx"
3 // #define BLYNK_DEVICE_NAME "Device"
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
20 void setup()
21 {
22     Serial.begin(115200);
23     delay(100);
24
25     BlynkEdgent.begin();
26 }
27
28 void loop() {
29     BlynkEdgent.run();
30 }
31

```

Figura 4-14. Ejemplo / Plantilla de Blynk Edgent.

Para el uso adecuado de esta plantilla es necesario adecuarla a la aplicación en cuestión, por lo que en primera instancia se modifican las credenciales ubicadas en la línea 2 y 3 de la figura 4-14. Recordando que estos identificadores son ubicados en la ventana de info de la página de *Blynk*, como se observa en la figura 4-9, debajo de *firmware configuration*.

Tras realizar las modificaciones indicadas, el *sketch* de Arduino queda como la figura 4-15.

```

Edgent_ESP32$ BlynkEdgent.h BlynkState.h ConfigMode.h ConfigStore.h Console.h Indicator.h OTA.h ResetButton.h Settings.h
1 // Fill-in information from your Blynk Template here
2 #define BLYNK_TEMPLATE_ID "TMPL2qeFkOm2i"
3 #define BLYNK_TEMPLATE_NAME "PruebaBlynk"
4 #define BLYNK_FIRMWARE_VERSION "0.1.0"
5
6 #define BLYNK_PRINT Serial
7 // #define BLYNK_DEBUG
8
9 #define APP_DEBUG
10 #define USE_ESP32S2_DEV_KIT
11 #include "BlynkEdgent.h"
12
13 void setup()
14 {
15     Serial.begin(115200);
16     delay(100);
17
18     BlynkEdgent.begin();
19 }
20
21 void loop()
22 {
23     BlynkEdgent.run();
24 }
25

```

Figura 4-15. Ejemplo / Plantilla de Blynk Edgent.

De igual manera, para que la aplicación desarrollada sea capaz de trabajar tanto *online* como *offline*, es necesario dirigirse a la cabecera llamada *settings.h* y ubicarse en la línea 93, 94 y 95 como lo indica la figura 4-16.

```

Edgent_ESP32.h  BlynkEdgent.h  BlynkState.h  ConfigMode.h  ConfigStore.h  Console.h  Indicator.h  OTA.h  ResetButton.h  Settings.h
83 #if !defined(CONFIG_AP_URL)
84 #define CONFIG_AP_URL "blynk.setup"
85 #endif
86 #if !defined(CONFIG_DEFAULT_SERVER)
87 #define CONFIG_DEFAULT_SERVER "blynk.cloud"
88 #endif
89 #if !defined(CONFIG_DEFAULT_PORT)
90 #define CONFIG_DEFAULT_PORT 443
91 #endif
92
93 #define WIFI_CLOUD_MAX_RETRIES 500
94 #define WIFI_NET_CONNECT_TIMEOUT 50000
95 #define WIFI_CLOUD_CONNECT_TIMEOUT 50000
96 #define WIFI_AP_IP IPAddress(192, 168, 4, 1)
97 #define WIFI_AP_Subnet IPAddress(255, 255, 255, 0)
98 // #define WIFI_CAPTIVE_PORTAL_ENABLE
99
100 // #define USE_TICKER
101 // #define USE_TIMER_ONE
102 // #define USE_TIMER_THREE
103 // #define USE_TIMER_FIVE
104 #define USE_PTHREAD
105
106 #define BLYNK_NO_DEFAULT_BANNER
107
108 #if defined(APP_DEBUG)
109 #define DEBUG_PRINT(...) BLYNK_LOG1(__VA_ARGS__)
110 #define DEBUG_PRINTF(...) BLYNK_LOG(__VA_ARGS__)
111 #else
112 #define DEBUG_PRINT(...)
113 #define DEBUG_PRINTF(...)
114 #endif
115

```

Figura 4-16. Cabecera de ajustes en la plantilla de Blynk Edgent.

La línea 93 de la figura 4-16 indica la cantidad de intentos para conectarse a la nube, la línea 94 y 95, el tiempo que tratará de reconectarse. Si tras este tiempo no es posible conectarse a la red, el algoritmo ejecuta las líneas de código que se ubiquen después de BlynkEdgent.run(); de la figura 4-15, e intentará conectarse de nuevo por el tiempo establecido en las líneas 94 y 95.

Tras algunos experimentos, se concluyó que el tiempo mínimo que puede ser colocado en las líneas 94 y 95 de la figura 4-16, es de 15 segundos. Al colocar tiempos menores, el microcontrolador no fue capaz de establecer conexión a la red.

De coloca un tiempo inferior a 15 segundos, existe la posibilidad de que la tarjeta logre comunicación con el servidor de la plataforma de Blynk, pero en la mayoría de los casos no es así. La figura 4-17 muestra un mensaje característico en el monitor serial de cuando no se logra conectar y el código de error.

```

COM3
13:49:54.126 -> [90958] CONNECTING_NET => RESET_CONFIG
13:49:54.126 -> [90959] Resetting configuration!
13:49:54.126 -> [90963] Configuration stored to flash
13:49:54.126 -> [90963] RESET_CONFIG => WAIT_CONFIG
13:49:56.728 -> [93574] AP SSID: Blynk pControlAcceso-608FD
13:49:56.728 -> [93574] AP IP:
13:49:56.728 -> [93575] AP URL: blynk.setup
13:50:23.370 -> [120219] WAIT_CONFIG => CONFIGURING
13:50:23.370 -> [120219] Sending board info...
13:50:24.878 -> [121749] Scanning networks...
13:50:30.040 -> [126870] Found networks: 14
13:50:43.621 -> [140487] Applying configuration...
13:50:43.621 -> [140487] Wifi SSID: IDEA Pass:
13:50:43.621 -> [140487] Blynk cloud: tZubPipkNNfPr6E544XG2fRkbtE0TxR @ blynk.cloud:
13:50:43.669 -> [140492] CONFIGURING => SWITCH_TO_STA
13:50:43.669 -> [140494] Switching to STA...
13:50:44.806 -> [141664] SWITCH_TO_STA => CONNECTING_NET
13:50:44.806 -> [141665] Connecting to WiFi: IDEA
13:50:49.799 -> [146666] Last error code: 701
13:50:49.799 -> [146671] Configuration stored to flash
13:50:49.799 -> [146671] CONNECTING_NET => ERROR

```

Figura 4-17. Mensaje de error al colocar un tiempo menor a 15 segundos para intentar conectarse al servidor de la plataforma de Blynk.

4.2.2 Uso de *widgets* en la plataforma de *Blynk*

Para el diseño de la interfaz gráfica y el funcionamiento deseado entre el microcontrolador y la plataforma *Blynk*, es necesario hacer un bosquejo de que variables son ocupadas, y de qué manera se ocupan.

Para este proceso, es necesario dirigirse al menú de *templates* (plantillas), situarse en la ventana de *datastream*, y dar clic en el botón de editar, como lo indica la figura 4-18.

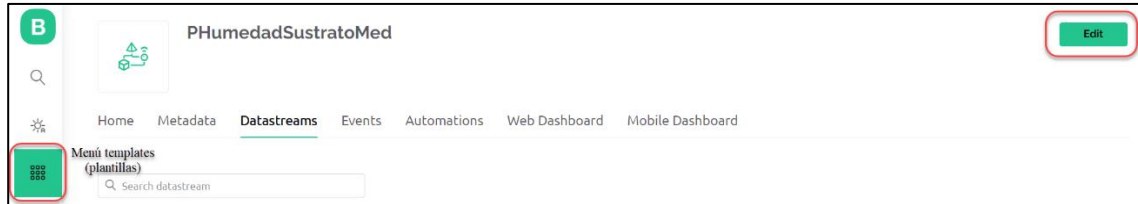


Figura 4-18. Menú *templates*, ventana de *datastream*.

Seguido de ello, permite crear variables, las que van a estar interactuando entre el microcontrolador y la plataforma de *Blynk*, como se muestra en la figura 4-19.

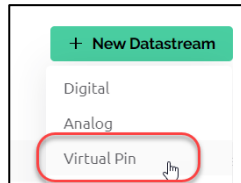


Figura 4-19. Crear y asignar variables en la plataforma de *Blynk*.

Posteriormente, despliega una ventana donde es posible asignar parámetros a la variable, como es indicado en la figura 4-20.

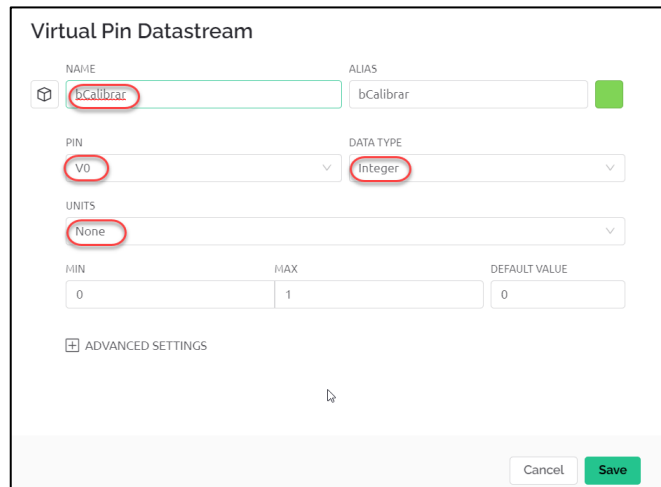


Figura 4-21. Edición de parámetros de la variable en la plataforma de *Blynk*.

Se observa en la figura 4-21 que se le debe asignar un nombre a la variable a trabajar, además de que se fija un pin virtual, para el caso, pin virtual V0, y se puede señalar que tipo de datos maneja, siendo *int*, *float* y *char*; en unidades colocar si es variable porcentual, de temperatura, de humedad, partes por millón, etc. Finalmente, se le indica a la plataforma los valores mínimos, máximos y por *default* que tiene la variable.

Concretado este paso, debe situarse en la ventana *web dashboard*, donde se colocan los *widgets* adecuados para las variables que van a representar, como se muestra en la figura 4-22.

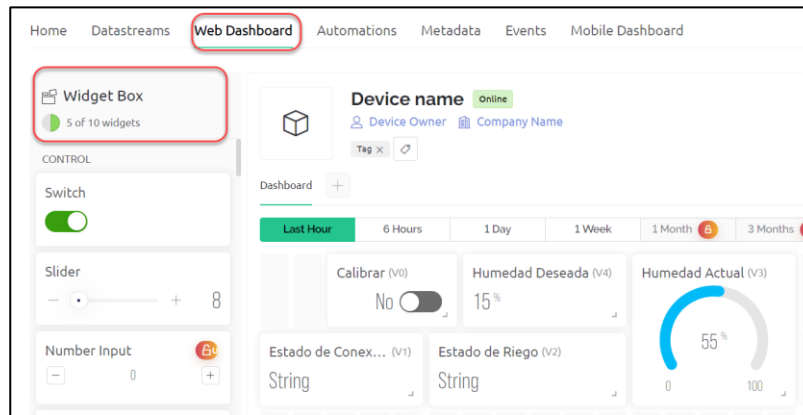


Figura 4-22. Asignación de variable a widgets en la plataforma de Blynk.

Una vez que se ha diseñado la interfaz gráfica modo web a través de la plataforma de *Blynk*, queda conocer las líneas de código que permiten interactuar con los *widgets* dispuestos. Una buena práctica es checar la documentación propia de *Blynk*², debido a que tienden a actualizar los comandos.

Para fines de comunicación entre el microcontrolador y la plataforma de *Blynk*, los comandos principales usados fueron los siguientes:

```

1 //Comunicación del microcontrolador hacia la plataforma de Blynk
2 Blynk.virtualWrite(V0, variableLeida); // Envía al pin virtual V0 el
  valor de la variable leída
3 Blynk.setProperty(V0, "color", "#f44336"); //Modifica el parámetro de
  color del pin virtual V0
4 Blynk.virtualWrite(V1, "Cadena de Texto"); //También se pueden mandar
  cadenas de texto
5
6 //Comunicación de la plataforma de Blynk hacia el microcontrolador
7 BLYNK_WRITE(V2)
8 {
9   agregar = param.asInt(); //Al pin virtual V2 se le asigna la
  variable "agregar".
10}

```

4.3 Elección de panel solar

Para la toma de decisión sobre cual panel solar adquirir para el desarrollo de este trabajo se tomó en cuenta la potencia máxima de trabajo que requieren en conjunto los módulos de manera simultánea. Para ello, una vez funcionales los módulos, se procedió a tomar lectura de corriente consumida de cada uno, arrojando los resultados de la tabla 4.4.

Tabla 4.4. Consumo de corriente de los módulos desarrollados

Módulo	Corriente [mA]	Potencia [W]
--------	----------------	--------------

² <https://docs.blynk.io/en/blynk.apps/widgets-app>.

	Mínima	Máxima	Mínima	Máxima
Riego automático de sustrato	160	380	1.92	4.56
Detección de gas licuado de petróleo y cierre de válvula	120	580	1.44	6.96
Apertura de puerta mediante RFID	88	1500	1.05	18
Total	368	2460	4.41	29.52

El siguiente paso fue valorar que, incluso en un determinado momento en que los tres módulos estén activando su actuador, el tiempo que esto ocurre no es prolongado, es decir, planteando los siguientes escenarios:

- Cuando el sustrato requiere riego, se observó que, en aproximadamente 1 minuto éste llega al nivel de humedad establecido (de acuerdo con la planta (lengua de suegra, la cantidad de sustrato en la maceta y el caudal que proporciona la motobomba seleccionados).
- Si en ese momento se requiere la apertura de puerta, el módulo activa la chapa eléctrica por 2 segundos.
- Debido a que la electroválvula elegida es normalmente cerrada, mientras no exista fuga de gas, dicho actuador permanece energizado.

Entonces, en condiciones normales de uso del sistema se tiene la tabla 4.5.

Tabla 4.5. Consumo de potencia por considerar

Módulo	Corriente por considerar [mA]	Potencia por considerar [W]
Riego automático de sustrato	160	1.92
Detección de gas licuado de petróleo y cierre de válvula	580	6.96
Apertura de puerta mediante RFID	88	1.05
Total	368	9.93

De acuerdo con los valores obtenidos en la tabla 4-4 y 4-5, se optó por un módulo de panel solar de 12V a 30 W debido a que solo cuando los tres módulos están en operación se alcanza un consumo de 29.52 W, pero en condiciones habituales el consumo es de 9.93 W.

Como se vio en el marco teórico en 2.3.4, por cuestión de costo, se eligió una batería de ácido plomo de 12 V 4 Ah.

Entonces, cuando el panel solar no sea capaz de suministrar la energía para el sistema, entra la batería, capaz de mantener energizado el sistema por:

$$t = \frac{C}{I} = \frac{4 \text{ Ah}}{0.368 \text{ A}}$$

$$t = 10.87 = 10 \text{ horas } 52 \text{ minutos}$$

En caso de que el sistema este ocupando su máximo, entonces la batería es capaz de mantener energizado al sistema por:

$$t = \frac{C}{I} = \frac{4 \text{ Ah}}{2.46 \text{ A}}$$

$$t = 1.62 = 1 \text{ hora } 37 \text{ minutos}$$

La figura 4-23 muestra el sistema antes de ser montado.



Figura 4-23. Sistema antes de ser montado.

El sistema se montó en Tecámac, Estado de México orientado mayormente hacia el sur geográfico, en la tercera semana de mayo de 2023, tratándose de la estación del año de primavera.

Se monitoreó el sistema en tres escenarios: alimentado con el panel solar, con la batería y combinados, arrojando los resultados plasmados en la tabla 4.6.

Tabla 4.6. Sistema alimentado por el panel solar

Hora	Tiempo	V_{Panel} [V]	$V_{Módulos}$ [V]	$I_{Módulos}$ [A]	Potencia [W]	Actuador que no se activó
7:00	Soleado	22	2.2	0.378	0.83	Sin activación.
8:00	Soleado	22	5	0.4	2	Sin activación.
9:00	Soleado	22	17	1.3	22.1	Activos.

10:00	Soleado	22.8	18	1.8	32.4	Activos.
11:00	Soleado	21	17	1.7	28.9	Activos
12:00	Soleado	22.9	18	1.7	30.6	Ninguna.
13:00	Nublado	21.56	7.7	0.44	3.38	Sin chapa eléctrica.
14:00	Nublado	21.26	6	0.15	0.9	Sin chapa eléctrica, válvula.
15:00	Soleado	23.17	19	1.5	28.5	Activos.
16:00	Nublado	18.7	1	0.1	0.1	Sin activación.
17:00	Soleado	22.29	7.73	0.28	2.16	Sin chapa eléctrica.

Con los valores obtenidos en la tabla 4.6, se concluye que, el panel solar solo es capaz de suministrar la energía suficiente para energizar los actuadores siempre y cuando el tiempo este soleado. Si empieza a nublarse, disminuye la potencia que puede suministrar.

También, se destaca el hecho de contar con 5 horas de potencia pico, comprendidas entre las 10:00 a 15:00, con sus altibajos debido a las condiciones de tiempo presentadas.

La figura 4-24 muestra el monitoreo de tensión y corriente eléctrica del sistema.



Figura 4-24. Monitoreo de tensión y corriente eléctrica del sistema.

Además, aunque en la temporada de observación comprendida en primavera de 2023, y el sol comienza a salir a las 6:00, el panel solar no es capaz de generar la potencia necesaria desde esa hora. Es hasta aproximadamente las 9:00 que esto ocurre. De igual manera,

aunque el sol se oculte alrededor de las 19:00, el panel solar deja de suministrar la energía suficiente para alimentar al sistema cerca de las 17:30.

Tabla 4.7. Sistema alimentado por panel solar y batería.

Riego automático	Detección de gas	Apertura de puerta	$V_{in\ sistema} [V]$	$I_{Sistema} [A]$	$V_{Panel} [V]$
x	x	x	13	0.5	22.68
x	x	✓	12.6	2.2	20.6
x	✓	x	14.5	0.95	17.1
x	✓	✓	12.4	3.1	18.4
✓	x	x	13.3	0.82	19.2
✓	x	✓	12.6	2.9	17.3
✓	✓	x	13.3	1.66	17.9
✓	✓	✓	12.9	3.74	19.6

Entonces, al colocar la batería al sistema, cuando el tiempo comienza a ser nublado la batería auxilia al panel permitiendo la constante energización del sistema. Es notable una caída de tensión en las terminales del panel solar al conectar una carga, pues como se observa en la tabla 4.7, en circuito abierto hay una tensión de hasta 22.68 V en el panel solar, y cuando se le conectan los módulos, se les hace llegar una tensión de entre 12 V y 13 V.

El siguiente experimento consistió en conocer la autonomía de la batería, realizando diez activaciones para cada actuador y con una tensión inicial en la batería de 13.14 V, dando lugar a los valores de la tabla 4.8.

Tabla 4.8. Sistema alimentado por batería.

Riego automático	Detección de gas	Apertura de puerta	$V_{Batería} [V]$	$I_{Sistema} [A]$
x	x	x	12.6	0.53
x	x	✓	11.2	3.42
x	✓	x	12.4	0.97

x	✓	✓	11.3	3.7
✓	x	x	12.4	0.7
✓	x	✓	11.7	3.4
✓	✓	x	11.7	1.54
✓	✓	✓	11.4	3.87

El funcionamiento habitual del sistema es mantener energizados los módulos desarrollados, y que la válvula eléctrica sea el único actuador activo. Dicho esto, para conocer la duración de la batería energizando el sistema, cada hora fueron activadas diez veces la bomba sumergible y la chapa eléctrica. Y se observó que, el sistema energizado únicamente con la batería es capaz de alimentar el sistema durante 2 horas (20 aperturas de puerta, 20 riegos automáticos de sustrato y continua detección de gas). En la tercera hora, el sistema respondió hasta siete iteraciones del experimento propuesto. En ese momento, la tensión mínima que la batería indicó fue de 11.7 V, reinició todos los módulos y dejó de ser capaz de alimentar al sistema.

Este experimento es orientativo debido a que, lo normal, es que la chapa eléctrica sea usada durante el día, además, como se mencionó en las pruebas para el riego automático de sustrato, tomando en consideración: la planta elegida (planta de suegra), sustrato y capacidad de maceta (aproximadamente 4 litros); la bomba trabaja un estimado de 1 minuto, es decir, en caso de que el sistema este ocupando los tres módulos al mismo tiempo, esto conlleva máximo 1 minuto.

De igual manera, se realizó el experimento en donde los tres módulos fueron energizados, pero la válvula eléctrica fue el único actuador activo. Con una tensión inicial de 13.1 V, transcurrieron 4 horas hasta que el sistema comenzó a reiniciarse dejando de ser efectivo.

Con los datos recabados, se concluye que el sistema de captación de energía solar es capaz de mantener energizados los módulos desarrollados entre 9:00 y 17:00, siempre y cuando el tiempo no se encuentre nublado. Además, cuando la radiación solar deja de ser suficiente, la batería elegida es capaz de mantener energizado al sistema durante 4 horas en condiciones normales, en condiciones extraordinarias como se planteó en el experimento descrito previamente, 3 horas, pero con limitantes de potencia.

Por lo tanto, este hecho suma una autonomía del sistema estimada de 10 horas para la temporada de primavera en la localidad de Tecámac.

Conclusiones

La implementación de paneles solares puede implicar un costo inicial considerable. Sin embargo, si está dentro de las posibilidades, es importante considerar la inversión en energías renovables. A medida que avanza la tecnología y aumenta la demanda, es probable que los costos de los paneles solares disminuyan gradualmente en el tiempo. Al invertir en paneles solares, se toma una posición activa en la transición hacia un sistema energético más sustentable y limpio. Además, al impulsar la demanda de energía solar, se fomenta la investigación y el desarrollo de tecnologías aún más eficientes y asequibles. A pesar de los desafíos económicos iniciales, apostar por las energías renovables es una inversión a largo plazo que puede generar beneficios al disminuir el impacto ambiental.

Para este trabajo se calculó y midió la potencia consumida por los módulos desarrollados en cada escenario, para poder elegir el panel solar que mejor se adecuó a las necesidades. No obstante, las empresas que hay en el mercado se enfocan a proyectos de mayor consumo energético, por lo que el panel adquirido es de menor escala, aun así, fue capaz de cumplir con su cometido.

En la locación donde se colocó el sistema considerando la temporada de primavera del año, se obtuvo una autonomía estimada de 10 horas comprendidas entre las 9:00 y las 19:00. Ésta puede variar de acuerdo con condiciones de tiempo, estación del año en que se implemente el sistema, radiación solar presente, potencia entregada por el panel solar y capacidad energética de la batería.

Se desarrollaron tres soluciones domóticas, siendo éstas el riego automático de sustrato, detección de gas licuado de petróleo y cierre de válvula, así como apertura de puerta mediante *RFID*; además de ser energizados por un sistema de captación de radiación solar.

Para el monitoreo y riego de plantas o jardines, se empleó un “sensor de humedad” capacitivo, que en realidad mide conductividad, y para que fuese útil, se incorporó en la aplicación (web y móvil) una opción para que cuando el usuario considere que su planta tiene un nivel de humedad en sustrato adecuado, indique que así quiere que permanezca; esto, aun que no es el nivel exacto de humedad, el sistema mantendrá ese estatus.

En adición, para el monitoreo y control de fugas de gas, se empleó un sensor de gas MQ5 capaz de detectar, principalmente gas LP (licuado de petróleo). Es importante mencionar que el módulo desarrollado no está calibrado para obtener la concentración exacta en partes por millón, por lo que, de momento, solo indica presencia de gas y al detectarlo por al menos 10 segundos, entonces realiza un cierre de válvula, impidiendo que siga circulando gas por las tuberías. Cuando la lectura tomada por el sensor de gas disminuye respecto a un umbral establecido, permite nuevamente la circulación de gas. Además, si el módulo desarrollado detecta que este evento ha ocurrido tres veces consecutivas, realiza un cierre definitivo de válvula, para evitar posibles accidentes. Para cancelar el bloqueo, el usuario debe reiniciar el sistema

Por otra parte, para el control de accesos mediante *RFID*, se diseñó un módulo capaz de permitir la entrada hasta a diez usuarios, que son guardados en la memoria del microcontrolador, siendo responsabilidad del usuario gestionar los usuarios registrados, los cual es posible realizar desde las aplicaciones, pudiendo dar de alta/baja tarjetas de acceso.

Gracias al uso de la plataforma *Blynk*, se logró una comunicación eficiente entre el usuario y el sistema, destacando que las soluciones domóticas desarrolladas no dependen exclusivamente de la conectividad a la red o del servidor de Blynk. Cada módulo puede funcionar de manera independiente en ausencia de comunicación con el servidor, llevando a cabo sus respectivas tareas además de intentar restablecer la conexión a la plataforma *IoT*, evitando así que el sistema quede inutilizable.

Trabajo a futuro

Este trabajo representa un acercamiento a la implementación de sistemas domóticos caseros alimentándolos con energías renovables, y por lo que las posibilidades para el desarrollo son inmensas, aun así, se enumeran algunas mejoras que podrían aplicarse a los módulos presentados, así como la incorporación de nuevas funcionalidades.

- Implementar un sistema de alimentación con mayor capacidad.
- Implementar un módulo de monitoreo de energía captada por el panel solar.
- Para el módulo de riego automático de sustrato, añadir al algoritmo frecuencia de riego, i.e., cada determinado número de días y si la humedad está por debajo del nivel deseado, activar el actuador.
- Para el módulo de detección de gas licuado de petróleo, implementar una válvula normalmente abierta para disminuir el consumo energético, pues con esto, el estado usual del sistema será solo alimentar los circuitos de control, y cuando sea necesario, activar el actuador correspondiente.
- Para el módulo de apertura de puerta, colocar el *buzzer* en un pin independiente al actuador, para así generar tono de bienvenida y tono de alerta debido a un usuario desconocido.
- Implementar algoritmo para que la tarjeta ESP32, a través de sus dos núcleos, pueda trabajar fuera de línea con un núcleo y con el otro núcleo comunicarse con la plataforma de Blynk.
- Usar una plataforma IoT que permita el uso de más de dos dispositivos simultáneos, o bien, ascender de categoría en la plataforma de *Blynk*.

Referencias

- [1] A. M. Calvente, «Universidad Abierta Interamericana,» Junio 2007. [En línea]. Available: <http://www.sustentabilidad.uai.edu.ar/pdf/sde/uais-sds-100-002%20-%20sustentabilidad.pdf>. [Último acceso: 21 Junio 2022].
- [2] J. Santamarta, «Las energías renovables son el futuro,» 2004. [En línea]. Available: <https://www.nodo50.org/worldwatch/ww/pdf/Renovables.pdf>. [Último acceso: 21 Mayo 2023].
- [3] Naciones Unidas, «Energías renovables: energías para un futuro más seguro,» Naciones Unidas, [En línea]. Available: <https://www.un.org/es/climatechange/raising-ambition/renewable-energy#:~:text=El%20cambio%20a%20fuentes%20de,en%20favor%20de%20la%20salud..> [Último acceso: 21 Mayo 2023].
- [4] United Nations Climate Change, «El costo de las renovables se reduce drásticamente y supera la opción más barata de combustibles fósiles,» 4 Junio 2020. [En línea]. Available: <https://unfccc.int/es/news/el-costo-de-las-renovables-se-reduce-drasticamente-y-supera-la-opcion-mas-barata-de-combustibles#:~:text=Los%20costos%20de%20la%20electricidad,la%20creciente%20experiencia%20de%20desarrolladores..> [Último acceso: 21 Mayo 2023].
- [5] A. Claverie, «Domótica en el hogar: Comodidad y eficiencia energética,» ABB, 23 Octubre 2017. [En línea]. Available: <https://www.linkedin.com/pulse/dom%C3%B3tica-en-el-hogar-comodidad-y-eficiencia-alejandra-claverie/?originalSubdomain=es>. [Último acceso: 21 Mayo 2023].
- [6] Ferrovial, «Domótica ¿Qué es la domótica?,» Ferrovial, [En línea]. Available: <https://www.ferrovial.com/es/recursos/domotica/>. [Último acceso: 21 Mayo 2023].
- [7] M. A. R. Loperana, «Modelo de Sistema Domótico Virtual Aplicado a Entornos Educativos,» Junio 2011. [En línea]. Available: <https://tesis.ipn.mx/jspui/bitstream/123456789/17816/1/Modelo%20de%20sistema%20domotico%20virtual%20aplicado%20a%20entornos%20educativos.pdf>. [Último acceso: 22 Junio 2022].
- [8] Fundación Privada Institut Ildefons Cerdà, La Vivienda Domótica, Barcelona, 2000.
- [9] USEIT, «Domótica: qué es, características principales y funcionamiento,» Use It Software SL, 26 Febrero 2020. [En línea]. Available: <https://www.useit.es/blog/domotica-que-es-caracteristicas-principales-y-funcionamiento>. [Último acceso: 4 Agosto 2022].

- [10] Grupo de Nuevas Actividades Profesionales, de Energía Solar Fotovoltaica, Madrid, Colegio Oficial de Ingenieros de Telecomunicación, 2002, p. 122.
- [11] «Componentes de una Instalación Fotovoltaica,» s.f.. [En línea]. Available: <https://www.mheducation.es/bcv/guide/capitulo/8448171691.pdf>. [Último acceso: 2022 Agosto 4].
- [12] Arduino, «Software de Arduino,» Arduino, [En línea]. Available: <https://arduino.cl/programacion/>. [Último acceso: 12 Octubre 2022].
- [13] J. Beningo, «Cómo seleccionar y usar el módulo ESP32 con Wi-Fi/Bluetooth adecuado para una aplicación IoT industrial,» Digi-Key, 21 Enero 2020. [En línea]. Available: <https://www.digikey.com.mx/es/articles/how-to-select-and-use-the-right-esp32-wi-fi-bluetooth-module>. [Último acceso: 13 Agosto 2022].
- [14] A. R. Bruno Saravia, «ESP32 NODE MCU,» 2019. [En línea]. Available: https://www.microelectronicash.com/downloads/ESP32_MANUAL.pdf. [Último acceso: 13 Agosto 2022].
- [15] NAYLAMP MECHATRONICS, «SENSOR DE HUMEDAD DE SUELO FC-28,» NAYLAMP MECHATRONICS, [En línea]. Available: <https://naylampmechatronics.com/sensores-temperatura-y-humedad/47-sensor-de-humedad-de-suelo-fc-28.html>. [Último acceso: 13 Agosto 2022].
- [16] L. Llamas, «Sensor de humedad del suelo capacitivo y arduino,» Luis Llamas, 28 Abril 2021. [En línea]. Available: <https://www.luisllamas.es/sensor-de-humedad-del-suelo-capacitivo-y-arduino/>. [Último acceso: 13 Agosto 2022].
- [17] L. Llamas, «Detector de gases con arduino y la familia de sensores MQ,» Luis Llamas, 21 Octubre 2016. [En línea]. Available: <https://www.teachmemicro.com/mq-4-methane-gas-sensor-arduino/>. [Último acceso: 14 Agosto 2022].
- [18] HANWEI ELECTRONICS, «Technical Data MQ-4 Gas Sensor,» [En línea]. Available: <https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-4.pdf>. [Último acceso: 14 Agosto 2022].
- [19] UNIT ELECTRONICS, «MQ-5 Detector de Gas Natural,» UNIT ELECTRONICS Tienda de Componentes Electrónicos, [En línea]. Available: <https://uelectronics.com/producto/mq-5-detector-de-gas-natural/>. [Último acceso: 14 Agosto 2022].
- [20] UNIT ELECTRONICS, «MQ-6 Detector de Gas LP,» UNIT ELECTRONICS Tienda de Componentes Electrónicos, [En línea]. Available: <https://uelectronics.com/producto/mq-6-detector-de-gas-lpg/>. [Último acceso: 14 Agosto 2022].
- [21] NAYLAMP MECHATRONICS, «TUTORIAL MÓDULO LECTOR RFID RC522,» NAYLAMP MECHATRONICS, [En línea]. Available:

- https://naylampmechatronics.com/blog/22_tutorial-modulo-lector-rfid-rc522.html. [Último acceso: 14 Agosto 2022].
- [22] UNIT ELECTRONICS, «Módulo Lector RFID RC522,» UNIT ELECTRONICS, [En línea]. Available: <https://uelectronics.com/producto/modulo-lector-rfid-rc522/>. [Último acceso: 14 Agosto 2022].
- [23] Capterra, «Blynk,» [En línea]. Available: <https://www.capterra.mx/software/179337/blynk#:~:text=Blynk%20es%20la%20plataforma%20de,administrar%20miles%20de%20productos%20implementados..> [Último acceso: 15 Noviembre 2022].
- [24] Cayenne, «Cayenne Docs. Introduction,» Cayenne, [En línea]. Available: <https://developers.mydevices.com/cayenne/docs/intro/>. [Último acceso: 28 Abril 2023].
- [25] ThingSpeak, «About ThingSpeak,» ThingSpeak, [En línea]. Available: <https://thingspeak.com/>. [Último acceso: 28 Abril 2023].
- [26] F. Quevedo, «Estadística Aplicada a la Investigación en Salud,» Marzo 2011. [En línea]. Available: <https://dsp.facmed.unam.mx/wp-content/uploads/2013/12/Quevedo-F.-Medidas-de-tendencia-central-y-dispersion.-Medwave-2011-Ma-113..pdf>. [Último acceso: 12 Octubre 2022].
- [27] W. Mendenhall, R. Beaver y B. Beaver, «Introducción a la probabilidad y estadística,» México, D.F., CENGAGE Learning, 2010, p. 56.
- [28] L. Escobar, «Conceptos Básicos de Procesamiento Digital de Señales,» México, D.F., Facultad de Ingeniería, UNAM, 2008, p. 6.
- [29] L. Monter y D. Rios, «1.3.3 Tipos de Señales (Análogica-Digital),» Universidad Autónoma del Estado de Hidalgo, [En línea]. Available: http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro27/133_tipos_de_seales_analgicadigital.html. [Último acceso: 12 Octubre 2022].
- [30] «Digitalización de señales analógicas,» [En línea]. [Último acceso: 12 Octubre 2022].
- [31] SEAS, «El Relé: para qué es, para qué sirve y qué tipos existen,» 22 Agosto 2019. [En línea]. Available: <https://www.seas.es/blog/automatizacion/el-rele-para-que-es-para-que-sirve-y-que-tipos-existen/>. [Último acceso: 15 Noviembre 2022].
- [32] HETPRO, «Optoacoplador, qué es y como funciona,» 2021. [En línea]. Available: <https://hetpro-store.com/TUTORIALES/optoacoplador/>. [Último acceso: 15 Noviembre 2022].
- [33] Ingeniería Mecafenix, «¿Qué es un capacitor? y sus tipos,» 20 Febrero 2019. [En línea]. Available: <https://www.ingmecafenix.com/electronica/el-capacitor/>. [Último acceso: 17 Noviembre 2022].

[34] M. PAREJA, Radiación solar y su aprovechamiento energético, Barcelona, España: Marcombo S.A., 2010.

Fuentes de imágenes

Figura 2-1. Elementos de un sistema domótico recuperada el 4 de agosto de 2022 de useit (2020) “Domótica: qué es, características principales y funcionamiento” del sitio <https://www.useit.es/blog/domotica-que-es-caracteristicas-principales-y-funcionamiento>

Figura 2-2. Elementos de una instalación solar fotovoltaica recuperada el 4 de agosto de 2022 de Anónimo (s.f.) “Componentes de una instalación fotovoltaica” del sitio <https://www.mheducation.es/bcv/guide/capitulo/8448171691.pdf>

Figura 2-3. Estructura de la célula solar recuperada el 4 de agosto de 2022 de Anónimo (s.f.) “Componentes de una instalación fotovoltaica” del sitio <https://www.mheducation.es/bcv/guide/capitulo/8448171691.pdf>

Figura 2-4 Constitución de un panel solar. recuperada el 4 de agosto de 2022 de Anónimo (s.f.) “Componentes de una instalación fotovoltaica” del sitio <https://www.mheducation.es/bcv/guide/capitulo/8448171691.pdf>

Figura 2-5. Conexiones del regulador en una instalación fotovoltaica recuperada el 4 de agosto de 2022 de Anónimo (s.f.) “Componentes de una instalación fotovoltaica” del sitio <https://www.mheducation.es/bcv/guide/capitulo/8448171691.pdf>

Figura 2-6 Esquema general de una instalación autónoma con inversor recuperada el 4 de agosto de 2022 de Anónimo (s.f.) “Componentes de una instalación fotovoltaica” del sitio <https://www.mheducation.es/bcv/guide/capitulo/8448171691.pdf>

Figura 2-7. Instalación fotovoltaica conectada a la red recuperada el 4 de agosto de 2022 de Anónimo (s.f.) “Componentes de una instalación fotovoltaica” del sitio <https://www.mheducation.es/bcv/guide/capitulo/8448171691.pdf>

Figura 2-8. Arduino IDE recuperada el 12 de octubre de 2022 de Arduino.cl (s.f.) “Software de Arduino” del sitio <https://arduino.cl/programacion/>

Figura 2-9. Pinout de la tarjeta ESP32 recuperada el 13 de agosto de 2022 de uPesy (s.f.) “Use the GPIO pins of the ESP32” del sitio <https://www.upesy.com/blogs/tutorials/esp32-pinout-reference-gpio-pins-ultimate-guide>

Figura 2-10. Esquema de montaje del sensor de humedad resistivo FC-28 recuperada el 13 de agosto de 2022 de LLAMAS, Luis. (2016) “Medir la humedad del suelo con Arduino e higrómetro FC-28” del sitio <https://www.luisllamas.es/arduino-humedad-suelo-fc-28/>

Figura 2-11. Pinout del sensor de humedad de sustrato capacitivo v1.2 recuperada el 13 de agosto de 2022 de LLAMAS, Luis (2021) “Sensor de humedad del suelo capacitivo y Arduino” del sitio <https://www.luisllamas.es/sensor-de-humedad-del-suelo-capacitivo-y-arduino/>

Figura 2-12. Algunos sensores de gas de la serie MQ recuperada el 13 de agosto de 2022 de LLAMAS, L. (2016) “Detector de gases con Arduino y la familia de sensores MQ” del sitio <https://www.luisllamas.es/arduino-detector-gas-mq/>

Figura 2-13. Pinout general de la familia de sensores de gas MQ recuperada el 14 de agosto de 2022 de PELAYO, Roland (s.f.) “How to use MQ-4 Methane Gas Sensor” del sitio <https://www.teachmemicro.com/mq-4-methane-gas-sensor-arduino/>

Figura 2-14 Sensor MQ-4 recuperada el 14 de agosto de 2022 de COMPONENTS 101 (2018) “MQ-4 Methane Gas Sensor” del sitio <https://components101.com/sensors/mq-4-methane-gas-sensor-pinout-datasheet>

Figura 2-15. Sensor MQ-5 recuperada el 14 de agosto de 2022 de UNIT ELECTRONICS (s.f.) “MQ-5 Detector de Gas Natural” del sitio <https://uelectronics.com/producto/mq-5-detector-de-gas-natural/>

Figura 2-16. Sensor MQ-6 recuperada el 14 de agosto de 2022 de UNIT ELECTRONICS (s.f.) “MQ-6 Detector de Gas LP” del sitio <https://uelectronics.com/producto/mq-6-detector-de-gas-lpg/>

Figura 2-17. Módulo RFID RC522 recuperada el 14 de agosto de 2022 de Ja-Bots (s.f.) “RFID - RC522” del sitio <https://ja-bots.com/producto/rfid-rc522/>

Figura 2-18. Señal original v.s. señal muestreada recuperada el 12 de octubre de 2022 de Anónimo (s.f.) “Digitalización de señales analógicas” del sitio <https://ikastaroak.birt.eus/edu/argitalpen/backupa/20200331/1920k/es/IEA/ELEC/ELEC01/es IEA ELEC01 Contenidos/webste 3 digitalizacin de seales analgicas.html>

Figura 2-19. Cuantización y Codificación de una señal recuperada el 12 de octubre de 2022 de Anónimo (s.f.) “Digitalización de señales analógicas” del sitio <https://ikastaroak.birt.eus/edu/argitalpen/backupa/20200331/1920k/es/IEA/ELEC/ELEC01/es IEA ELEC01 Contenidos/webste 3 digitalizacin de seales analgicas.html>

Figura 2-20 Diagrama de un optoacoplador recuperada el 15 de noviembre de 2022 de HETPRO (2021) “Optoacoplador, qué es y como funciona” del sitio <https://hetpro-store.com/TUTORIALES/optoacoplador/>

Figura 2-21. Tipos de capacitores recuperada el 17 de noviembre de 2022 de Anónimo (s.f.) “Capacitor Types” del sitio <https://images.app.goo.gl/UckjxWzJecBERuS58>

Figura 2-22. Radiación recuperada el 29 de mayo de 2023 de PAREJA, M. (2010) “Radiación solar y su aprovechamiento energético” del sitio <https://books.google.es/books?hl=es&lr=&id=YkxOEAAAQBAJ&oi=fnd&pg=PP1&dq=radiaci%C3%B3n+solar&ots=r8-b3GHjao&sig=TyRKFBfwT-qNDrlNggn-eEXsYt0#v=onepage&q=radiaci%C3%B3n%20solar&f=false>

Figura 2-23. Irradiación diaria promedio anual en México de Centro Mexicano de Innovación en Energía Solar (s.f.) “Irradiación solar diaria promedio mensual” del sitio <https://rayenari.geofisica.unam.mx/~mauro/mapas.php>

Figura 2-24. Promedio diario mensual de horas pico de radiación solar en la Ciudad de México recuperada el 29 de mayo de 2023 de Centro Regional de Medición de la Radiación Solar IV Región Meteorológica (AR-IV) Organización Meteorológica Mundial (s.f.) “Energía solar disponible en Ciudad Universitaria, D.F.” del sitio https://areas.geofisica.unam.mx/radiacion_solar/energia.php?grafica=hp

Anexo

A) Código de programación para el módulo de riego automático de sustrato

```
1 #define BLYNK_TEMPLATE_ID "TMPLI1MfuCkZ"
2 #define BLYNK_DEVICE_NAME "PHumedadSustratoMed"
3 #define BLYNK_FIRMWARE_VERSION "0.1.0"
4 #define BLYNK_PRINT Serial
5 #define APP_DEBUG
6 #include "BlynkEdgent.h"
7 #include <Preferences.h>
8 Preferences referencia;
9 int humCali;
10 int humLectura;
11 int temporal = 0;
12 int auxMedir = 0;
13 int auxCalibrar = 0;
14 int vecesRiego = 0;
15 bool bCalibrar = 0;
16 bool bCalibrar2 = 0;
17 bool bandera = 0;
18 void setup()
19 {
20   pinMode(13, OUTPUT); //Salida a bomba
21   pinMode(25, INPUT_PULLUP); //Boton de calibrar, ISR
22   attachInterrupt(25, interrupcion, FALLING);
23   Serial.begin(115200);
24   delay(100);
25   referencia.begin("riegoAutomatico", false);
26   humCali = referencia.getInt("humCali", 0);
27   Blynk.virtualWrite(V4, humCali);
28   BlynkEdgent.begin();
29 }
30 void loop()
31 {
32   BlynkEdgent.run();
33   calcularHumedad();
34   if (bCalibrar == 1 || bCalibrar2 == 1)
35   {
36     bandera = 1;
37   }
38   switch (bandera)
39   {
40     case 0:
41       mostrarHumedad();
42       break;
43     case 1:
44       calibrarHumedad();
45       break;
46   }
47   riego();
48 }
49 void calcularHumedad()
50 {
51   int contador = 0;
```



```

52 //Realiza todas las mediciones que pueda en 1 seg, y luego obtiene
el promedio
53 unsigned long tempRefCont = millis();
54 while ((millis() - tempRefCont) <= 1000)
55 {
56     temporal = temporal + map(analogRead(33), 800, 4000, 100, 0);
57     contador++;
58 }
59 Blynk.virtualWrite(V1, "EN LÍNEA");
60 temporal = temporal / contador;
61 bandera = 0;
62 }
63 void mostrarHumedad()
64 {
65     humLectura = temporal;
66
67     if (humLectura != auxMedir)
68     {
69         Blynk.virtualWrite(V3, humLectura);
70         Blynk.virtualWrite(V5, vecesRiego);
71         auxMedir = humLectura;
72     }
73 }
74 void calibrarHumedad()
75 {
76     humCali = temporal;
77     referencia.putInt("humCali", humCali);
78     referencia.end();
79     if (humCali != auxCalibrar)
80     {
81         Blynk.virtualWrite(V4, humCali);
82         auxCalibrar = humCali;
83     }
84     Blynk.virtualWrite(V0, 0);
85     bCalibrar = 0;
86     bCalibrar2 = 0;
87     bandera = 0;
88 }
89 void riego()
90 {
91     if ( humLectura <= (humCali - 10))
92     {
93         digitalWrite(13, HIGH);
94         Blynk.virtualWrite(V2, "EN RIEGO");
95         vecesRiego++;
96     }
97     else if ( humLectura >= (humCali + 10))
98     {
99         digitalWrite(13, LOW);
100         Blynk.virtualWrite(V2, "HUMEDAD ADECUADA");
101     }
102 }
103 void interrupcion()
104 {
105     bCalibrar2 = 1;
106 }
107 BLYNK_WRITE(V0)

```

```

108{
109  bCalibrar = param.asInt();
110}
111

```

B) Código de programación para el módulo de detección de gas licuado de petróleo y cierre de válvula

```

1  #define BLYNK_TEMPLATE_ID "TMPLkHdp5J1"
2  #define BLYNK_DEVICE_NAME "PGas"
3  #define BLYNK_FIRMWARE_VERSION "0.1.0"
4  #define BLYNK_PRINT Serial
5  #define APP_DEBUG
6  #include "BlynkEdgent.h"
7  unsigned long tempRefsUGas = 0;
8  unsigned long tempRefCalentar = 0;
9  unsigned long tempRef = 0;
10 int lecturaGas;
11 int contadorAlertas = 0;
12 int lecturaTx;
13 bool banderaCalentar = 0;
14 bool SUGas = 0;
15 void setup()
16 {
17   Serial.begin(115200);
18   delay(100);
19   pinMode(13, OUTPUT);
20   calentarSensor();
21   BlynkEdgent.begin();
22 }
23 void loop()
24 {
25   BlynkEdgent.run();
26   medicionGas();
27   checarFuga();
28   controlarFuga();
29   cierreValvulaDefinitivo();
30 }
31
32 void calentarSensor()
33 {
34   if (banderaCalentar == 0)
35   {
36     tempRefCalentar = millis();
37     while ((millis() - tempRefCalentar) <= 900000)
38     {
39     }
40     banderaCalentar = 1;
41   }
42 }
43
44 void medicionGas()
45 {
46   unsigned long tempRefCont = millis();
47   while ((millis() - tempRefCont) <= 1000)
48   {
49     for (int i = 0; i < 9; i++ )

```

```

50     {
51         lecturaGas = lecturaGas + analogRead(33);
52         tempRef = millis();
53     }
54     lecturaGas = lecturaGas / 10;
55 }
56 Serial.println(lecturaGas);
57 lecturaTx = map(lecturaGas, 50, 2000, 0, 100);
58 Blynk.virtualWrite(V2, lecturaTx);
59 Blynk.virtualWrite(V3, contadorAlertas);
60 }
61
62 void checarFuga()
63 {
64     if ((lecturaGas > 500) && SUGas == 0)
65     {
66         tempRefSUGas = millis();
67         SUGas = 1;
68         Blynk.virtualWrite(V1, "Posible fuga de gas");
69         Blynk.setProperty(V0, "color", "#f1c232");
70     }
71     else if (lecturaGas < 500)
72     {
73         SUGas = 0;
74         digitalWrite(13, LOW);
75         Blynk.virtualWrite(V1, "Todo en orden");
76         Blynk.setProperty(V0, "color", "#8fce00");
77     }
78 }
79
80 void controlarFuga()
81 {
82     if ((millis() - tempRefSUGas) > 10000) && SUGas == 1)
83     {
84         tempRefSUGas = millis();
85         Blynk.virtualWrite(V1, "FUGA DE GAS");
86         Blynk.setProperty(V0, "color", "#f44336");
87         digitalWrite(13, HIGH);
88         contadorAlertas++;
89         SUGas = 0;
90     }
91 }
92
93 void cierreValvulaDefinitivo()
94 {
95     if (contadorAlertas > 3)
96     {
97         Blynk.virtualWrite(V1, "FUGA DE GAS, CHECAR TUBERÍAS");
98         Blynk.setProperty(V0, "color", "#f44336");
99         while (1)
100         {
101         }
102     }
103 }
104

```

C) Código de programación para el módulo de apertura de puerta mediante *RFID*

```
1 #define BLYNK_TEMPLATE_ID "TMPLXKDu5xz2"
2 #define BLYNK_DEVICE_NAME "pControlAcceso"
3 #define BLYNK_FIRMWARE_VERSION "0.1.0"
4 #define BLYNK_PRINT Serial
5 #define APP_DEBUG
6 #include "BlynkEdgent.h"
7 #include <SPI.h>
8 #include <MFRC522.h>
9 #include <EEPROM.h>
10 #define SS_PIN 21
11 #define RST_PIN 22
12 MFRC522 mfrc522(SS_PIN, RST_PIN);
13 WidgetLCD lcd(V0);
14 byte LecturaID[4];
15 byte Usuario[10][4];
16 byte ranuraUsuario;
17 bool agregar;
18 bool quitar;
19 bool apertura;
20 bool bandera;
21 void setup()
22 {
23   Serial.begin(115200);
24   delay(100);
25   SPI.begin();
26   mfrc522.PCD_Init();
27   EEPROM.begin(512);
28   pinMode(12, OUTPUT);
29   BlynkEdgent.begin();
30 }
31 void loop()
32 {
33   BlynkEdgent.run();
34   if (!mfrc522.PICC_IsNewCardPresent())
35   {
36     BlynkEdgent.run();
37     return;
38   }
39   if (!mfrc522.PICC_ReadCardSerial())
40   {
41     BlynkEdgent.run();
42     return;
43   }
44   for (byte i = 0; i < 4; i++)
45   {
46     LecturaID[i] = mfrc522.uid.uidByte[i];
47   }
48   bandera = 0;
49   for (int j = 0; j < 10; j++)
50   {
51     if (comparar(LecturaID, Usuario[j]))
52     {
53       bandera = 1;
54       break;
55     }
56   }
57 }
```

```

56  }
57  if (bandera == 1)
58  {
59      lcd.clear();
60      lcd.print(0, 0, "  Bienvenido");
61      digitalWrite(12, HIGH);
62      delay(1000);
63  }
64  else
65  {
66      lcd.clear();
67      lcd.print(0, 0, " No te conozco");
68      delay(2000);
69  }
70  lcd.print(0, 0, "Presente tarjeta");
71  digitalWrite(12, LOW);
72  mfrc522.PICC_HaltA();
73  }
74  bool comparar(byte laLectura[], byte elUsuario[])
75  {
76      for (byte j = 0; j < 4; j++)
77      {
78          if (laLectura[j] != elUsuario[j])
79          {
80              return (false);
81          }
82      }
83      return (true);
84  }
85
86  BLYNK_WRITE(V1)
87  {
88      agregar = param.asInt();
89      if (agregar == 1)
90      {
91          for (int j = 0; j < 4; j++)
92          {
93              Usuario[ranuraUsuario - 1][j] = LecturaID[j];
94              EEPROM.write(j + (4 * (ranuraUsuario - 1)),
Usuario[ranuraUsuario - 1][j]);
95              EEPROM.commit();
96          }
97      }
98  }
99
100 BLYNK_WRITE(V2)
101 {
102     quitar = param.asInt();
103     if (quitar == 1)
104     {
105         for (byte i = 0; i < 4; i++)
106         {
107             Usuario[ranuraUsuario - 1][i] = 0;
108             EEPROM.write(i + (4 * (ranuraUsuario - 1)),
Usuario[ranuraUsuario - 1][i]);
109             EEPROM.commit();
110         }

```

```
111 }
112}
113
114BLYNK_WRITE(V3)
115{
116  apertura = param.asInt();
117  if (apertura)
118  {
119    lcd.clear();
120    lcd.print(0, 0, "  Bienvenido");
121    digitalWrite(12, apertura);
122    delay(2000);
123    lcd.clear();
124    lcd.print(0, 0, "Presente tarjeta");
125    digitalWrite(12, LOW);
126  }
127}
128
129BLYNK_WRITE(V4)
130{
131  ranuraUsuario = param.asInt();
132}
133
```