



**UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO**

**FACULTAD DE ESTUDIOS SUPERIORES
ARAGÓN**

**SISTEMA PARA CONFIGURACIÓN Y CONTROL DE
UNA CAFETERA POR MEDIO DE UNA APLICACIÓN
MÓVIL PARA SISTEMA OPERATIVO ANDROID**

T E S I S

Que para obtener el título de:

INGENIERO ELÉCTRICO ELECTRÓNICO

Presenta:

MIRANDA BARBA LEONARDO GABRIEL

Director de tesis:

DR. DÍAZ RANGEL ISMAEL



Ciudad Nezahualcóyotl Estado de México, octubre de 2020



Universidad Nacional
Autónoma de México

Dirección General de Bibliotecas de la UNAM

Biblioteca Central



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

A mis padres: por todo el apoyo, la paciencia y la comprensión que me han brindado en cada etapa de mi vida. Por el esfuerzo que han hecho por sacar adelante a mi hermana y a mí a pesar de las dificultades y los obstáculos que ha puesto la vida.

A mis amigos: por brindarme una mano cuando lo he necesitado, por inspirarme a ser cada vez una mejor persona, por recordarme el camino cuando he perdido el rumbo y por compartir todas las experiencias que nos han hecho lo que somos ahora.

A mis profesores: porque sus enseñanzas me han traído hasta donde estoy, porque me han brindado las herramientas para desarrollarme de manera profesional y personal.

A mi asesor: por el apoyo brindado en la realización de este trabajo, la constancia que tuvo al revisar mis avances y por motivarme a realizar este trabajo.

Índice general

Agradecimientos.....	i
Índice general	ii
Capítulo 1.- Introducción	1
1.1.- Objetivo General.....	2
1.2.- Objetivos Particulares	2
1.3.- Actividades	3
1.4.- Justificación	3
1.5.- Antecedentes	3
1.6.- Descripción del capitulado.....	4
Capítulo 2.- Marco teórico	5
2.1.- Tipos de cafeteras	5
2.1.1.- Cafeteras manuales	5
2.1.2.- Cafeteras automáticas	5
2.1.3.- Cafeteras súper automáticas.....	6
2.2.- Internet de las cosas (IoT).....	7
2.3.- Modulación por ancho de pulso.....	9
2.4.- Electrónica de potencia	10
2.4.1.- Detector de cruce por cero	10
2.4.2.- Circuito de acoplamiento.	14
2.4.3.- Disipación de calor	17
2.5.- Sistemas de control digital	19
2.5.1.- Microcontrolador	21
2.5.2.- Tarjeta de desarrollo	24
2.5.3.- ESP32.....	25
2.5.4.- IDE Arduino.....	26
2.6.- Plataforma Cayenne	27
3.- Desarrollo experimental.....	28
3.1.- Diagrama de bloques.....	28
3.2.- Diagrama esquemático del dispositivo	29
3.2.1.- Implementación del circuito de cruce por ceros.	32
3.2.2.- Diseño de la etapa de potencia.....	34

3.3.- Desarrollo de la aplicación en la plataforma Cayenne.....	35
3.3.1.- Uso de la aplicación.....	41
3.4.- Programación del código en el IDE Arduino.....	46
3.5.- Fabricación de la placa PCB.....	50
4.- Pruebas y resultados.....	53
4.1.- Pruebas al circuito de detección de cruce por cero.....	53
4.2.- Pruebas al circuito de potencia.....	53
4.3.- Pruebas del código de Arduino.....	56
4.4.- Pruebas de la aplicación.....	59
4.5.- Pruebas finales.....	61
Conclusiones y trabajo futuro.....	64
Referencias de imágenes.....	67
Referencias bibliográficas.....	72
Anexo.....	75
a) Código de Arduino.....	75
b) Lista de materiales.....	79

Capítulo 1.- Introducción

Al día de hoy es difícil saber hace cuánto tiempo se empezó a beber el café de forma habitual, hay autores que piensan que en el siglo IX, el médico árabe Rhazes, ya contemplaba las propiedades de esta bebida que él llamaba “*bunchun*”. Otros autores creen que fue Jeque (título árabe para líderes religiosos o políticos) Omar en el siglo XIII, durante su destierro en el desierto, quien descubrió las propiedades de esta bebida. [1]

En América existen muchas leyendas sobre la llegada del café durante la época colonial, unas hablan de un capitán francés que llegó a las Antillas, alrededor de 1720, con una de estas plantas, que pudo cultivarla y transportarla Centroamérica. Otros creen que fue gracias a los ingleses, llegados a la Isla de Jamaica con cafetos, que ahora tenemos café en nuestro continente. [2]

Lo cierto es que, según los registros, el café llegó a México en 1790 desde las Antillas por el puerto de Veracruz donde se empezó a cultivar. Más tarde llegó a Chiapas desde Guatemala. [2]

Tuvieron que pasar algunos años hasta que esta bebida ganara terreno en nuestro país ya que la cultura del chocolate estaba muy arraigada [2]. *El Café Manrique* fue la primera cafetería en México, que en sus inicios fue una nevería, en las calles de Tacuba y Monte de Piedad en el centro histórico de la Ciudad de México, donde se ubicó durante todo el siglo XIX. [3]

En México el consumo de café de grano aumentó de un 28% en 2005 a un 46% en 2016, mientras que el consumo de café soluble disminuyó de un 72% a un 54% en el mismo lapso de tiempo [4]. Lo que también representa un aumento en el uso de cafeteras por parte de los mexicanos que anteriormente bebían café soluble, las hay en diferentes modelos, marcas y, sobre todo, precios.

La mayoría de las personas optan por las cafeteras más sencillas, ya sea por el precio, por la facilidad de manejarlas o simplemente porque no les interesa nada muy sofisticado con muchas funciones ya que el ritmo de vida no les permite gastar mucho tiempo en algo tan cotidiano.

Con la creciente implementación del internet de las cosas (IoT por sus siglas en inglés) se han desarrollado diferentes sistemas que permiten que la vida cotidiana sea más simple, la implementación de esta corriente tecnológica podría ayudar a optimizar los tiempos que la gente invierte en algo tan común como preparar un café y así aprovechar su tiempo.

Actualmente ya existen máquinas de café que utilizan esta u otras tecnologías para poder facilitar la preparación del café, la mayoría se programan directamente desde una pequeña pantalla en la cafetera, pero esto no es muy práctico ya que tienes que estar ahí en todo momento para programarla, algunas otras lo hacen mediante una aplicación en un dispositivo móvil utilizando el bluetooth del dispositivo, la desventaja de este tipo de sistemas es que el alcance es relativamente corto (10 metros como máximo), y si existe algún obstáculo como una pared el alcance disminuye aún más. Las cafeteras más básicas carecen de todas estas facilidades, en muchos casos solo un botón de encendido y apagado.

Es por eso que se plantea diseñar un sistema que pueda implementarse a una cafetera de este tipo para poder ser controlada mediante un dispositivo móvil Android y tener las funciones de programar horarios de trabajo, así como regular la potencia con la que trabaja la resistencia que calienta el agua, esto no solo ahorrará el tiempo que la persona invierte en la preparación del café si no que disminuirá su consumo de energía eléctrica.

1.1.- Objetivo General

Desarrollar un dispositivo que se pueda integrar a una cafetera de calentamiento por resistencia, y mediante una aplicación de móvil, utilizando el internet de las cosas (IoT), pueda configurar su funcionamiento.

1.2.- Objetivos Particulares

- Desarrollar mediante la plataforma Cayenne una aplicación que permita controlar y programar el dispositivo.
- Enlazar la aplicación con el dispositivo a través de comunicación WiFi.
- Poder seleccionar fecha y hora de encendido para múltiples eventos.
- Programar el tiempo que permanecerá encendido el dispositivo.
- Controlar la potencia de calentamiento.
- Modo de operación programable y manual.
- Implementar una etapa de acoplamiento de potencia.
- Incorporar control mediante un sistema embebido.

- Cumplir con el paradigma del IoT.

1.3.- Actividades

- Estudiar el paradigma de internet de las cosas
- Conocer tarjetas de desarrollo basadas en microprocesadores
- Identificar dispositivos de comunicación inalámbrica.
- Familiarizarse con la plataforma Cayenne para el desarrollo de la aplicación móvil.
- Analizar los métodos de desarrollo de aplicaciones en Android.
- Elaborar un sistema de potencia de 127V desde un microcontrolador por PWM.
- Diseñar un algoritmo de control mediante un microcontrolador.
- Utilizar un sistema de comunicación para conectar el microcontrolador a una red Wi-Fi.
- Desarrollar una aplicación para que nos permita controlar y programar el microcontrolador desde un dispositivo Android.

1.4.- Justificación

Como se menciona en la introducción, ya existen cafeteras con sistemas programables, pero tienen desventajas en el alcance o la practicidad del manejo. El dispositivo que se describe en el presente trabajo intentará subsanar dichas fallas, así como reducir los tiempos que la gente necesita para preparar su café, ya que muchas veces, sobre todo por las mañanas en días laborales o escolares, la gente va a prisa para poder llegar a su trabajo o a sus clases y teniendo un sistema programado pueden enfocar ese tiempo en hacer otras cosas. Por otra parte, al reducir la potencia después de la preparación se reduce el consumo energético, lo cual ayuda económicamente al ahorro en el recibo de luz.

1.5.- Antecedentes

En 2018, las empresas Gemalto, empresa holandesa especializada en seguridad digital, y Bonaverde, empresa alemana de cafeteras súper automáticas, se aliaron para crear una cafetera usando el internet de las cosas, esta cafetera del tipo roast – grind - brew (tostar – moler – preparar) cuenta con conectividad a una plataforma en la nube que ayuda a todos los usuarios de esta línea a compartir las recetas programadas y compartir su experiencia con otros, también cuenta con un sistema de monitoreo que envía una solicitud al distribuidor cuando el café se empieza a agotar. Sin embargo esta cafetera no cuenta con

un sistema de programación de horarios y su tecnología está basada en la conectividad 3G. [5]

En noviembre de 2019 la empresa española Zemsania presento una cafetera inteligente, este modelo utiliza sensores para determinar cuándo las capsulas están por terminarse y hace el pedido al proveedor, fuera de eso no cuenta con mayor funcionalidad (a diferencia de la anterior), su conectividad es mediante 3G y WiFi y muestra los datos del stock en tiempo real. La única funcionalidad que posee esta cafetera es la del monitoreo del café utilizado y hacer el pedido cuando haga falta, por lo demás no se puede trabajar de manera automática o programable. [6]

El usuario Andrew_h del blog instructables.com publicó en 2018 un tutorial de como automatizar tu cafetera, utilizando unos pocos componentes electrónicos, mediante el internet de las cosas por medio de comandos de voz. En esta publicación describe de manera sencilla como armar tu cafetera, programarla e implementar una aplicación hecha en App Inventor. Este tutorial solo está diseñado para trabajar la cafetera en el momento, no cuenta con un sistema de programación de eventos igual que en los casos anteriores. [7]

1.6.- Descripción del capitulado

Capítulo 2. Se describe los diferentes tipos de cafeteras, los conocimientos teóricos necesarios para este trabajo y se detallan los componentes electrónicos utilizados y los entornos de programación utilizados.

Capítulo 3. Desarrollo experimental. En este capítulo se hace un planteamiento y descripción a detalle de la metodología a seguir para la elaboración de este dispositivo, cumpliendo los objetivos establecidos en esta sección.

Capítulo 4. Es en este capítulo se establecen las pruebas que se hicieron con los diferentes prototipos de cafeteras, aportando una presentación de las conclusiones y los resultados obtenidos a través de las pruebas aplicadas durante el proceso de elaboración.

Se finaliza este trabajo denotando las conclusiones e indicando los trabajos a realizar en un futuro.

Capítulo 2.- Marco teórico

2.1.- Tipos de cafeteras

Actualmente existen muchos tipos diferentes de cafeteras, cada una está enfocada a las diferentes necesidades de los usuarios y el tipo de café que se busque, existen desde cafeteras manuales hasta súper automáticas, pero no se puede decir que haya alguna mejor que otra. A continuación, se mencionarán algunos de los tipos más comunes que se pueden encontrar en el mercado.

2.1.1.- Cafeteras manuales

Estas cafeteras no requieren una alimentación eléctrica, el proceso suele ser más “artesanal”, su uso es bastante sencillo y solo se necesita agua caliente y café. Entre estas se encuentra la cafetera italiana o moka, la cual se puede apreciar en la ilustración 2.1, y la cafetera francesa o de embolo. En ambos casos suelen tener precios muy económicos, la preparación es más o menos rápida y tienen un sabor no tan fuerte [8].



Ilustración 2.1. Cafetera italiana o Moka.

2.1.2.- Cafeteras automáticas

Para estas cafeteras ya es necesaria una alimentación eléctrica, ya que la necesitan para realizar algunas funciones, en el caso anterior se necesitaba calentar el agua por separado o calentar la cafetera completa, estas ya cuentan con un sistema de calentamiento por resistencia o por inducción. También hay algunas que permiten la opción de programar tiempos de trabajo o la cantidad de tazas que se desean preparar [8].

Este es el tipo más fácil de encontrar en el mercado y se pueden encontrar muchos modelos y marcas diferentes, los tipos comunes con las cafeteras de goteo y las de capsula, ilustración 2.2. En este trabajo nos centraremos en este tipo ya que es la más común.



Ilustración 2.2. Cafetera de goteo.

2.1.3.- Cafeteras súper automáticas

Estas son similares a las cafeteras automáticas, ilustración 2.3., y tienen casi las mismas funciones, la diferencia principal está en que las cafeteras súper automáticas tuestan y muelen el grano, a diferencia de las otras donde es necesario comprar el grano ya tostado y molido, y con esto se consigue mayor diversidad en los tipos de café, la gran desventaja son sus altos costos y su gran tamaño. [8]



Ilustración 2.3. Cafetera súper automática.

2.2.- Internet de las cosas (IoT)

La *Internet of Things European Research Cluster* (IERC) define al internet de las cosas como “una infraestructura global para la sociedad de la información, que permite la ejecución de servicios avanzados mediante la conexión física y/o virtual de cosas mediante tecnologías de la información y comunicación interoperativas que existen actualmente, pero que evolucionan a lo largo del tiempo” [9].

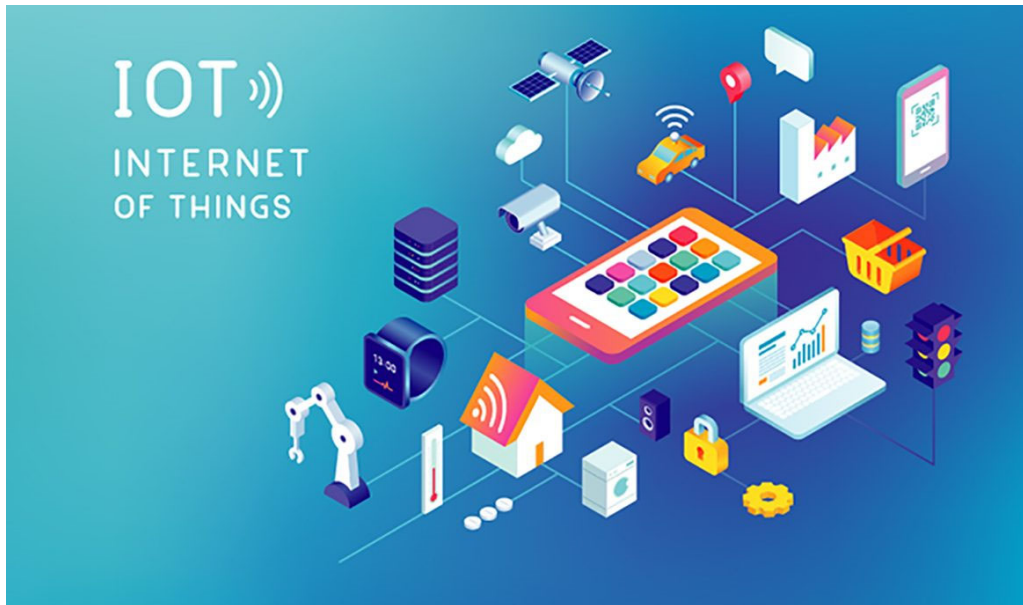


Ilustración 2.4. Imagen ilustrativa del internet de las cosas.

Existen diferentes características que definen a un sistema *IoT*, estas son [10]:

- Interconectividad: esta característica es fundamental ya que permite al dispositivo comunicarse con alguna infraestructura mundial.
- Heterogeneidad: Todos los dispositivos de diferentes plataformas hardware y redes pueden interactuar entre ellos.
- Cambios dinámicos: el estado del dispositivo (encendido, apagado, reposo, activo, etcétera), la posición del dispositivo (ubicación, velocidad, etcétera) o el número de dispositivos pueden variar constantemente.
- Escalabilidad: en los próximos años la cantidad de dispositivos IoT van a incrementar, lo que deberá hacer que la gestión y manipulación de los datos debe ser de forma eficiente.
- Conectividad basada en identificación: se deben poder conectar distintos objetos mediante la identificación.

- **Compatibilidad:** es necesario garantizar la compatibilidad para el uso de diferentes servicios.
- **Seguridad:** es necesario mantener los requisitos de seguridad cuando se integran varios dispositivos y redes para mantener la privacidad en la información.
- **Protección de la privacidad:** los dispositivos que recaben información personal del usuario es esencial que se protejan estos datos durante la transición y el almacenamiento de estos.
- **Autoconfiguración:** Los dispositivos IoT deben soportar una configuración automática.

Las anteriores son las características más importantes de los sistemas IoT tanto de bajo como de alto nivel, aunque para sistemas más básicos se permite usar las características relacionadas con la comunicación.

Los sistemas IoT más básicos tienen una arquitectura de tres capas, ilustración 2.5, estas son [10]:

- **Capa de percepción:** en este nivel se encuentran los sensores que captan las variables físicas que queremos medir como temperatura, distancia, humedad, etcétera. Estos sensores convierten las variables físicas en señales eléctricas que se procesan a través de un microcontrolador y después son enviadas.
- **Capa de red:** la función de esta capa es la de enviar las señales recogidas en la capa de percepción hacia su destino mediante WiFi, Bluetooth, 3G, 4G, etcétera.
- **Capa de aplicación:** en esta capa se desarrollan el objetivo de la aplicación del sistema y de los datos recogidos como activación de actuadores, relevadores, etcétera.



Ilustración 2.5. Arquitectura IoT de tres capas.

Aunque en la aplicación real, los sistemas de IoT deben considerar dos capas más, que son la de negocio y la de procesamiento, ilustración 2.6. Con esto ya podemos considerar una arquitectura de cinco capas. [10]



Ilustración 2.6. Arquitectura IoT de cinco capas.

2.3.- Modulación por ancho de pulso

La modulación por ancho de pulso o PWM (Pulse-Width Modulation), como su nombre lo indica, es una técnica de modulación que se basa en el ciclo de trabajo de una señal de corriente directa. Esta modulación utiliza un tren de pulsos que va desde una tensión máxima a cero y así sucesivamente utilizando el tiempo en el que se mantiene activo. Para ejemplificarlo de manera gráfica se puede ver la ilustración 2.7. [11]

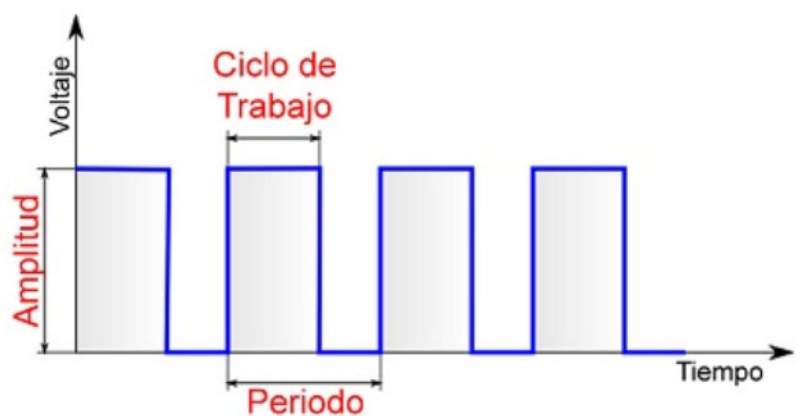


Ilustración 2.7. Modulación PWM

Para calcular el voltaje que genera esta modulación se utiliza la siguiente formula:

$$V_{media} = \frac{V_{max} d}{100}$$

Donde:

V_{media} – Voltaje obtenido por la modulación.

V_{max} – Amplitud máxima de la corriente directa.

d – Porcentaje del ciclo de trabajo

Este tipo de modulación sirve para diferentes propósitos ya que, actualmente, los microcontroladores ya permiten hacerlo de manera sencilla y eficiente. Algunas de sus aplicaciones pueden ser controlar los tonos de un LED RGB, variar la velocidad de un motor de CD, regular el brillo de un LED, controlar un servo motor analógico, etcétera.

2.4.- Electrónica de potencia

La electrónica de potencia es el área de la electrónica que trata de las aplicaciones de los dispositivos electrónicos de estado sólido para el control y conversión de sistemas eléctricos de gran potencia. Para esta aplicación se requiere trabajar en el régimen de conmutación de los dispositivos semiconductores, actualmente los circuitos integrados y otro tipo de componentes están siendo sustituidos por microcontroladores para controlar los sistemas, pero aun es necesario agregar algunos complementos, como transistores BJT, MOSFET, DIAC, TRIAC, etcétera, para poder manejar la gran cantidad de corriente ya que el microcontrolador por sí solo no lo soportaría. [12]

2.4.1.- Detector de cruce por cero

Al utilizar corriente alterna muchas veces es necesario detectar el punto donde la tensión es cero para medir la frecuencia, rectificar el desfase o conocer el momento en el que se tiene que conmutar la carga, por mencionar unos ejemplos. Para eso es necesario utilizar un detector de cruce por cero.

Este es un circuito utilizado en interfaces de potencia necesario para controlar el paso de la señal senoidal de la corriente alterna para hacer la conexión o desconexión de la carga en el punto donde la tensión es mínima para evitar armónicos y evitar reducir el tiempo de vida de los componentes. El circuito envía un estado alto cuando la señal pasa por cero al microcontrolador para poder sincronizarlo con onda de CA (ilustración 2.8.). [13]

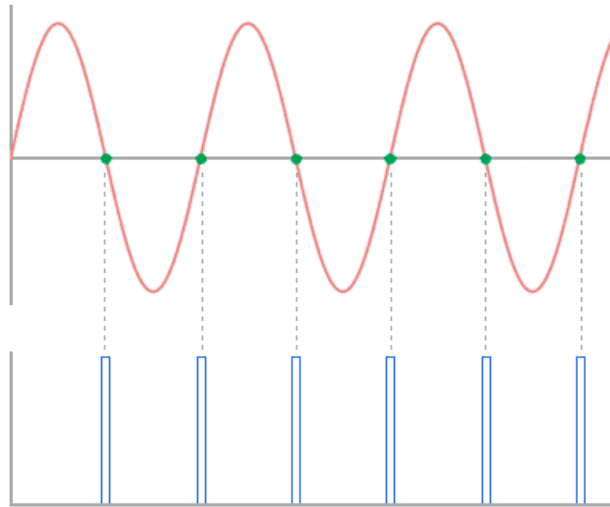


Ilustración 2.8. Detección de cruce por cero

Para esta labor se pueden utilizar diferentes sistemas y circuitos integrados, pero en este trabajo utilizamos el ejemplificado en la ilustración 2.9 ya que es sencillo de utilizar, fácil de conseguir y económico.

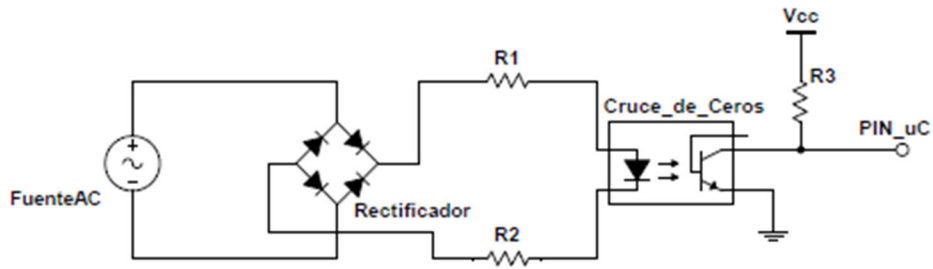


Ilustración 2.9. Circuito del detector de cruce por cero

2.4.1.1.- Puente rectificador

Los rectificadores son circuitos hechos con diodos que cambian la forma de onda de la señal que reciben. Pueden ser de dos tipos:

- Rectificador de media onda hecho por un solo diodo (ilustración 2.10) que reciben su nombre debido a que solo permiten pasar un semiciclo de la onda y bloquean el otro. En la ilustración 2.11 se ve la señal de entrada en azul y la señal rectificada en rojo y se aprecia que el semiciclo negativo se recorta.

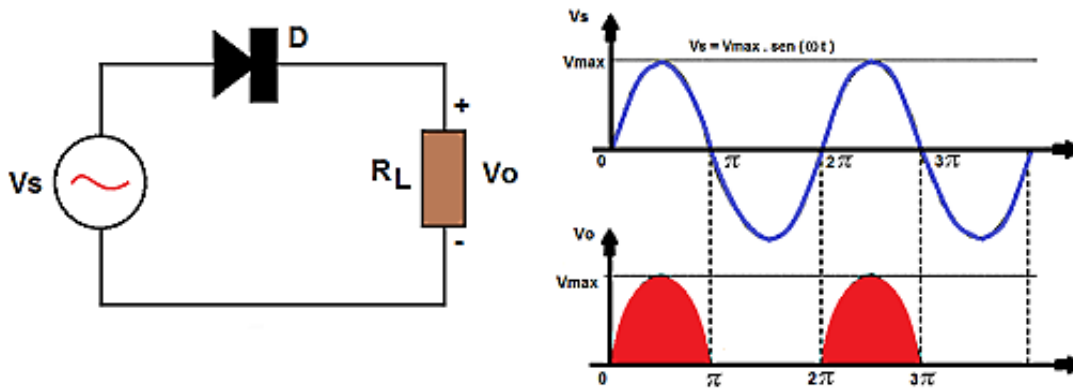


Ilustración 2.10 Rectificador de media onda

- Rectificador de onda completa, donde a su vez existen dos tipos, uno de ellos utiliza dos diodos y un transformador con tap central (ilustración 2.11) y donde cada parte del devanado secundario del transformador permite el paso de cada semiciclo. Un inconveniente de este rectificador es la necesidad de un transformador con tap central, cosa que en el siguiente se omite.

El último y más utilizado de los rectificadores es el puente rectificador, el cual se construye con cuatro diodos y no necesita un transformador con tap central para rectificar la señal. En la ilustración 2.12 se ve como rectifica el semiciclo negativo de la señal y después el positivo.

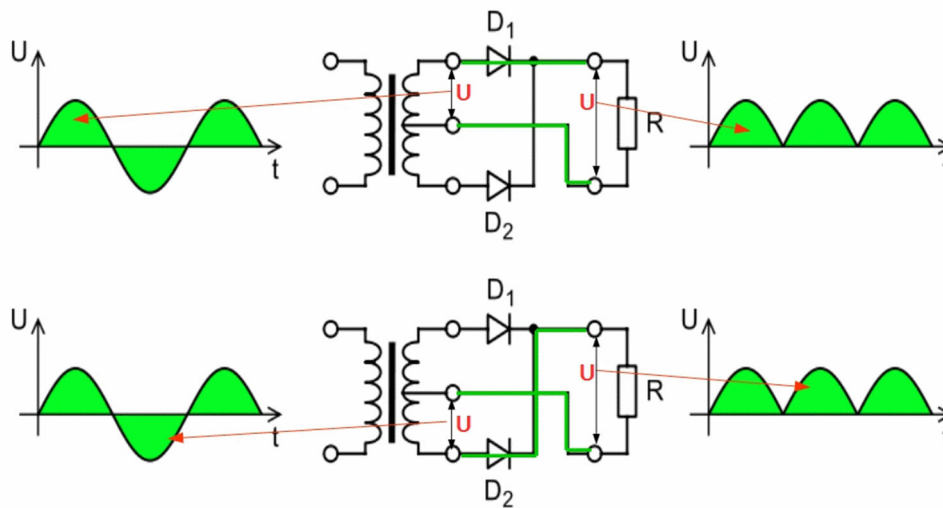


Ilustración 2.11 Rectificador de onda completa con tap central

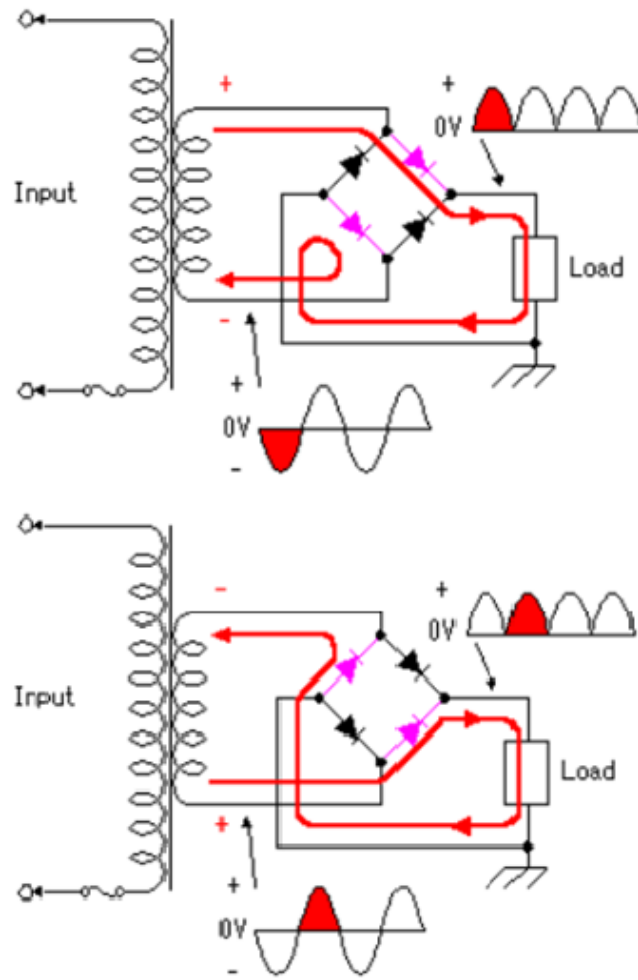


Ilustración 2.12 Puente rectificador de onda completa

Para calcular el voltaje de este rectificador tenemos la siguiente formula:

$$V_{CC} = 0.9 V_{ef} \quad (\text{ec. 2.1})$$

Donde:

V_{CC} – voltaje de corriente continua que sale del puente rectificador

V_{ef} – voltaje eficaz con el que se alimenta el puente

2.4.1.2.- Detector de cruce por ceros H11AA1

El H11AA1 es un circuito integrado (ilustración 2.13) conformado por dos LED infrarrojos conectados en anti paralelo, acoplados ópticamente con un fototransistor detector. Cuenta con una protección interna para evitar una polarización inversa.

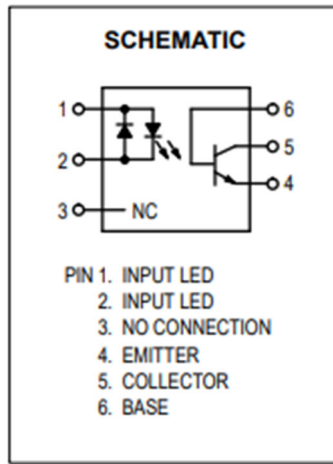


Ilustración 2.13 Esquema del H11AA1

Algunas de las aplicaciones que se le dan a este dispositivo es para detectar señales de CA, enviar señales a controladores programables, como interfaces y acoplamiento para sistemas de diferentes impedancias, etcétera. [14]

2.4.2.- Circuito de acoplamiento.

Los circuitos de acoplamiento eléctrico - electrónico se basan en recibir señales digitales de un circuito de control, procesarla y activar salidas del circuito de potencia. La manera más fácil de acoplar ambos circuitos es con un módulo relevador, que es un componente electromecánico, este módulo solo funciona en dos estados, encendido y apagado, pero para este proyecto no es útil ya que requiere el uso de un PWM y el dispositivo mecánico no soporta conmutaciones a alta frecuencia.

Para esta aplicación utilizaremos un acoplamiento electrónico mediante un optoacoplador y un SCR para controlar la carga con una señal PWM como se muestra en la ilustración 2.14.

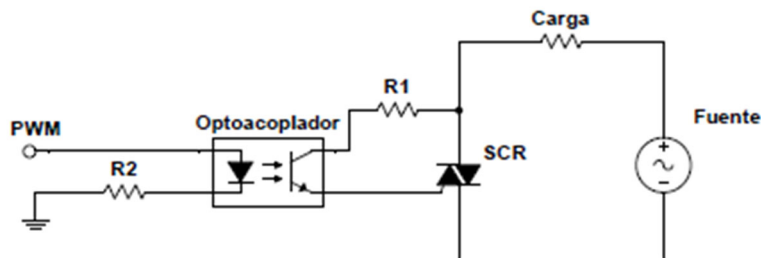


Ilustración 2.14 Circuito de acoplamiento

Donde el microcontrolador envía una señal al optoacoplador y este a su vez enciende el gate del SCR y este enciende hasta el siguiente paso por cero de la señal en la carga y esperará el siguiente pulso para hacerlo de nuevo (ilustración 2.15)

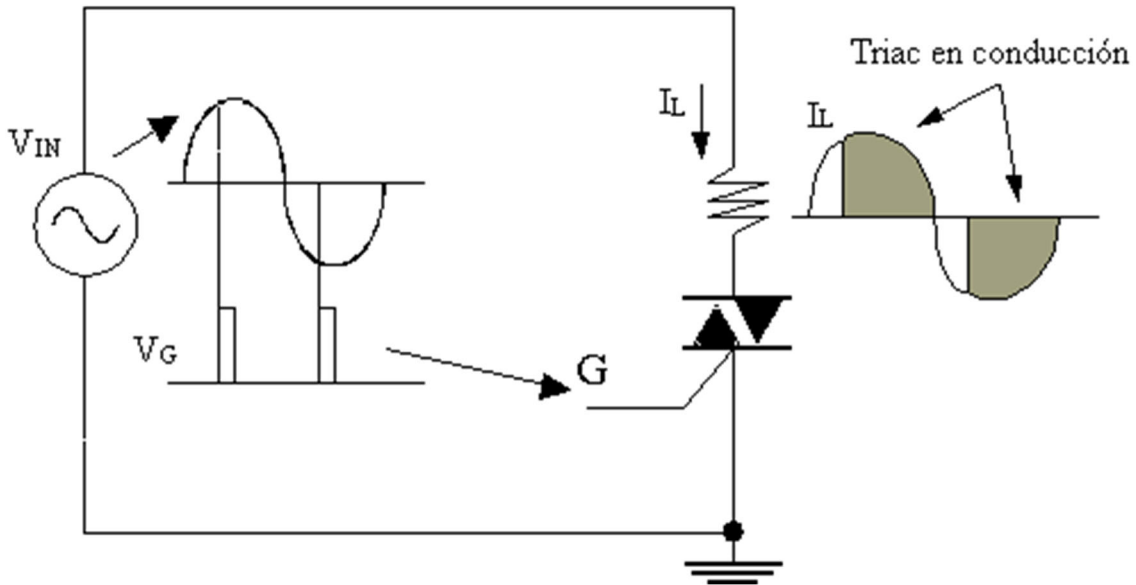


Ilustración 2.15 Esquema de funcionamiento del triac

Para calcular el voltaje de salida del triac se utiliza la siguiente ecuación:

$$V_m = \frac{V_p}{\pi} (1 + \cos \alpha) \quad (\text{ec. 2.2})$$

Donde:

V_m – es el voltaje medio que entrega el triac a la carga.

V_p – es el voltaje pico de la onda senoidal.

α – es el ángulo de disparo del triac.

2.4.2.1.- Optoacoplador MOC3010.

Un optoacoplador, también llamado optoaislador o aislador acoplado ópticamente, es un dispositivo electrónico que funciona como interruptor mediante la luz emitida por un LED que satura un componente opto electrónico que puede ser un fototransistor o un fototriac que se conectan con el LED de manera óptica. Estos componentes se encuentran encapsulados en un circuito integrado y se utilizan para aislar señales de dispositivos muy sensibles o para separar la etapa de control con la etapa de potencia para reducir el riesgo de daños en el microcontrolador. [15]

El MOC3010 (ilustración 2.16) es un optoacoplador con salida de triac de 250V, que contiene un LED infrarrojo GaAs. Este dispositivo está diseñado para ser la interfaz entre los controles electrónicos y la etapa de potencia para controlar cargas resistivas e inductivas. [15]

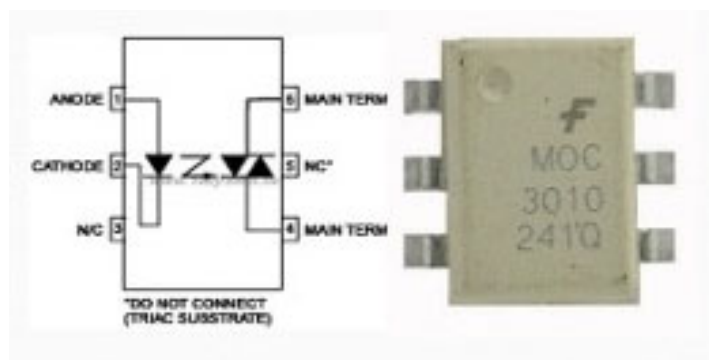


Ilustración 2.16 Esquema del MOC3010

2.4.2.2.- Triac BTA12.

El triac o SCR (por sus siglas en inglés Silicon Controlled Rectifier) es un dispositivo que hace la función de un interruptor como un transistor, pero a diferencia de este, el triac puede controlar la corriente alterna.

El triac es un componente electrónico de 6 capas, 2 de ellas tipo P y 4 de tipo N (ilustración 2.17). Este componente cuenta con tres terminales, dos ánodos y una puerta, los dos ánodos se conectan en serie con la carga que se quiere controlar y la puerta (o gate en inglés) al medio con el que se va a encender y apagar, ya que con la corriente se activa y permite el paso de la corriente alterna a través de él. [16]

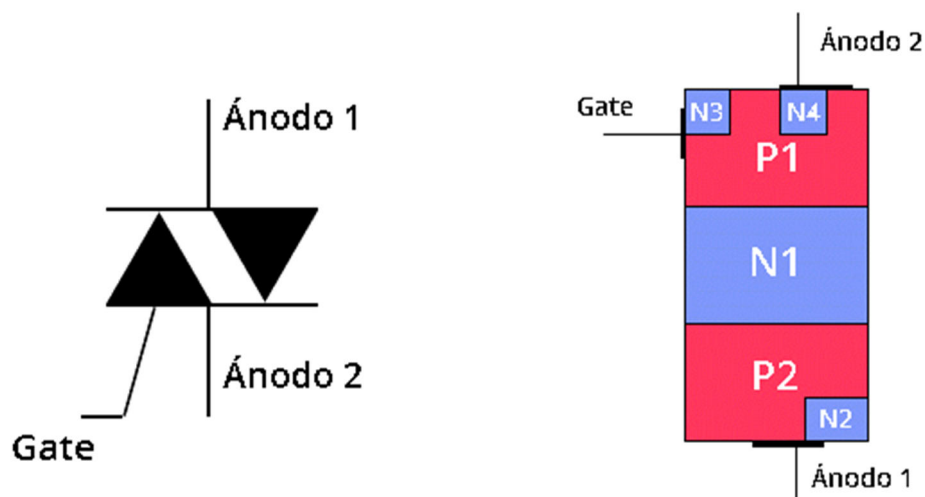


Ilustración 2.17 Composición de un SCR

Aunque la mayoría de sus aplicaciones es encender y apagar elementos de potencia, también se puede utilizar para controlar la velocidad de motores o regular la potencia de una resistencia encendiendo y apagando a una gran velocidad, a esto se le conoce como modulación por ancho de pulso o PWM

El SCR BTA12 (ilustración 2.18) es un dispositivo para aplicaciones de conmutación en C.A. de propósito general. Se puede usar como interruptor en relevadores de estado sólido, circuitos de arranque suave de motores, en control de fase como dimmers de luz, interfaz de dispositivos de baja potencia como microcontroladores, etcétera. [17]

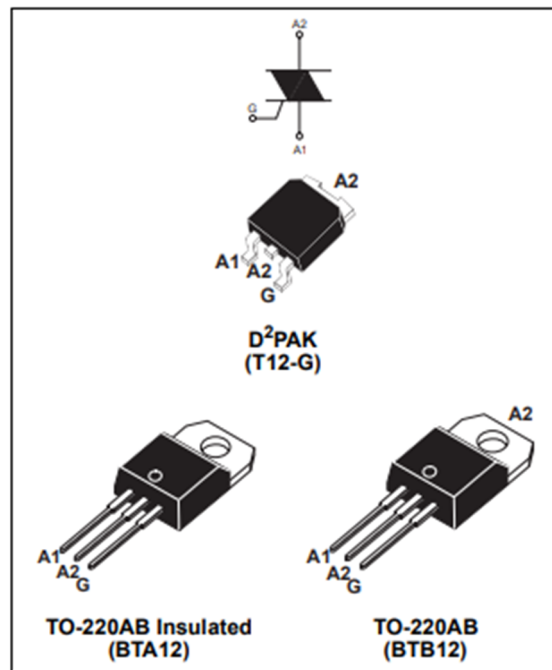


Ilustración 2.18 Triac BTA12 y sus tipos de encapsulado.

2.4.3.- Disipación de calor

Los elementos electrónicos de potencia como transistores, triacs, reguladores de voltaje, etcétera, suelen trabajar con potencias de magnitudes altas y su tamaño es pequeño en la mayoría de los casos. Debido al efecto Joule todos los elementos que conducen la electricidad sufren un calentamiento, en los semiconductores esto se da en las uniones PN y si la temperatura sobrepasa ciertos límites se provoca una fusión térmica que daña el componente.

Los disipadores de calor (ilustración 2.19) son elementos externos que ayudan a propagar la temperatura del componente electrónico hacia el ambiente por el método de convección, esto ayuda a reducir la temperatura interna de dicho elemento. [18] [19]

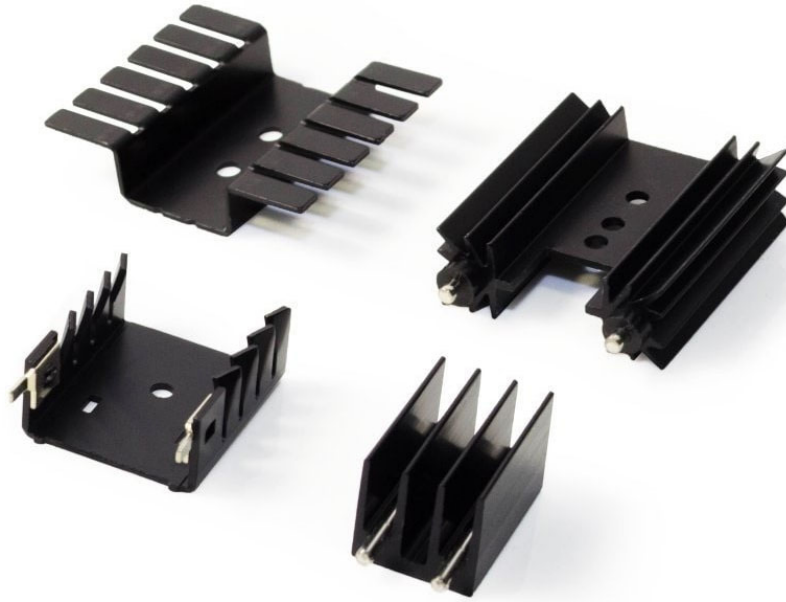


Ilustración 2.19 Algunos tipos de disipadores de calor.

Para hacer el cálculo de la disipación de temperatura del elemento electrónico de potencia tenemos la siguiente formula:

$$T_j - T_a = PR_{tht} \quad (\text{ec. 2.3})$$

Donde:

T_j – temperatura máxima de juntura del elemento semiconductor.

T_a – temperatura ambiente.

P – potencia consumida por el elemento.

R_{tht} – resistencia térmica total entre la unión y el ambiente.

Para saber qué tipo de disipador se necesita o si es necesario colocarlo primero se debe buscar en el datasheet del semiconductor los valores de T_j y $R_{thj-amb}$. Una vez teniendo esos datos se necesita conocer la potencia del circuito y la temperatura ambiente, con estos datos se calcula la T_j estimada despejando la formula anterior, de la siguiente forma:

$$T_{j \text{ estimada}} = PR_{thj-amb} + T_a \quad (\text{ec. 2.4})$$

Si $T_{j \text{ estimada}} \geq T_j$ o está muy cercana, debe colocarse un disipador para ayudar al componente electrónico a disipar el calor.

Una vez estimado que se necesita colocar un disipador necesitamos conocer la temperatura que es necesaria disipar al ambiente, para eso utilizamos la siguiente ecuación:

$$R_{th\ d-amb} = \frac{T_j - T_a}{P} - (R_{th\ j-c} + R_{th\ c-d}) \quad (\text{ec. 2.5})$$

Donde:

$R_{th\ d-amb}$ – resistencia térmica entre el disipador y el ambiente.

T_j – temperatura máxima de juntura del elemento semiconductor.

T_a – temperatura ambiente.

P – potencia consumida por el elemento.

$R_{th\ j-c}$ – resistencia térmica entre la juntura y el encapsulado del semiconductor.

$R_{th\ c-d}$ – resistencia térmica entre el encapsulado y el disipador.

2.5.- Sistemas de control digital

Un sistema de control es un conjunto de elementos que forman un sistema, de tal manera que es capaz de llevar a cabo un objetivo. A este sistema se le aplica una señal de entrada y se obtiene una señal de salida o respuesta y se puede representar por bloques como se ve en la ilustración 2.20. [20]

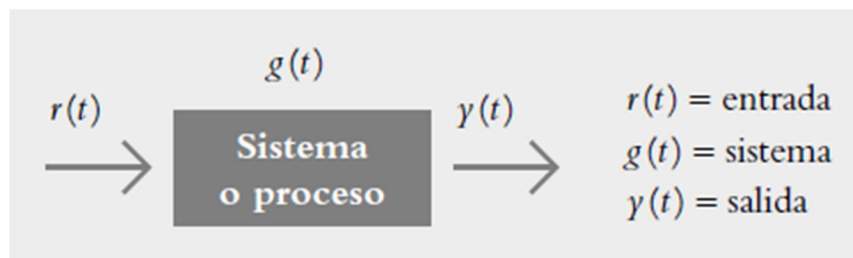


Ilustración 2.20 Sistema básico de control

Existen tres señales de entrada típicas para estos sistemas, ilustración 2.21, son las siguientes: [20]

- La entrada escalón es una señal de entrada constante al sistema.
- La entrada rampa se caracteriza por ser una señal que crece de manera constante con el tiempo.
- La entrada impulso es una señal de prueba de magnitud muy grande y tiempo muy corto.

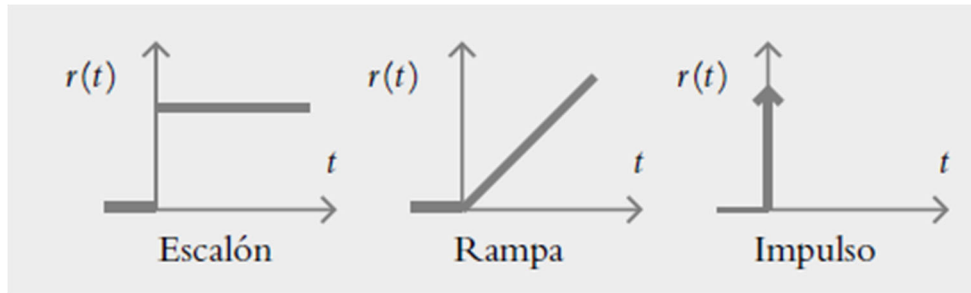


Ilustración 2.21 Tipos de entradas aplicadas a los sistemas de control

Los sistemas de control se clasifican en dos tipos, lazo abierto y lazo cerrado, también llamados sin retroalimentación y retroalimentado, respectivamente.

Los sistemas de lazo abierto (ilustración 2.22), son sistemas en los cuales el control es independiente de la salida. Utilizan un controlador o algún actuador para obtener la salida deseada y su precisión y exactitud dependen de la calibración de estos. [20]

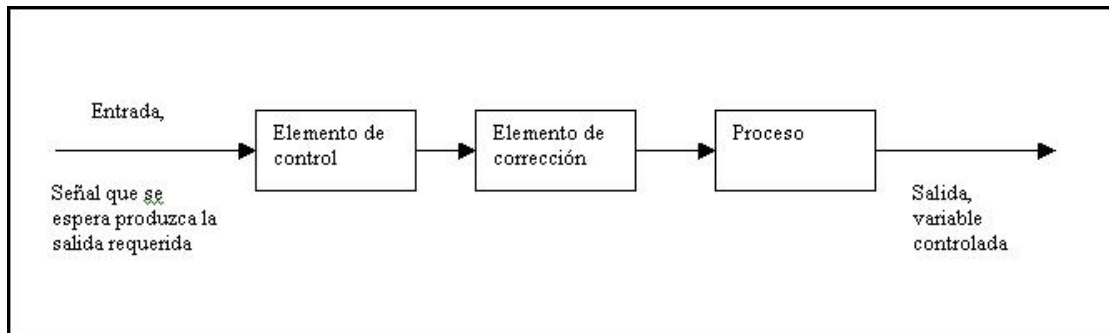


Ilustración 2.22 Sistema de control de lazo abierto

Los sistemas de lazo cerrado (ilustración 2.23), son aquellos cuyo control depende de la salida, es decir, toman una muestra de la salida mediante un sensor para compararla con una referencia y corrige el error, por esto son llamados retroalimentados. [20]

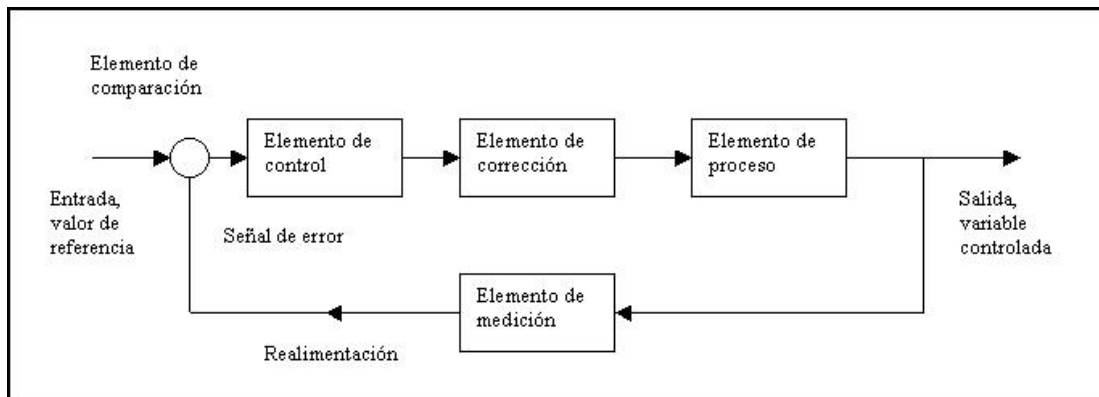


Ilustración 2.23 Sistema de control de lazo cerrado

2.5.1.- Microcontrolador

Un microcontrolador es un circuito integrado de propósito general que tiene los elementos más básicos de que se pueden encontrar en cualquier computadora (ilustración 2.24): [21]

- **CPU:** es el que realiza las acciones del microcontrolador, de él depende la velocidad de procesamiento y el ancho de banda del bus de datos, por lo general en esos circuitos integrados se utilizan de 8 a 16 bits.
- **Memoria:** es la que almacena los el programa de arranque y las variables temporales, generalmente son flash EEPROM.
- **Periféricos:**
 - **Entradas y salidas de propósito general:** permiten recibir y enviar datos, suelen utilizarse para comunicación entre microcontroladores o para controlar actuadores como LEDs o motores.
 - **Entradas analógicas:** son ADC que se encuentran dentro del microcontrolador que permiten recibir datos de variables continuas, se debe tener en cuenta su resolución cuando se trabaja con señales de alta frecuencia.
 - **Temporizadores:** son circuitos síncronos internos o externos que se utilizan para general señales de reloj.
 - **Memoria EEPROM:** la mayoría de los microcontroladores cuentan con una memoria interna que se encarga de guardar la información importante en aun cuando se desenergice el circuito integrado.
 - **Salidas PWM:** estos puertos permiten al usuario modificar el ciclo de trabajo para simular una salida analógica y así controlar la intensidad de un LED o la velocidad de un motor.

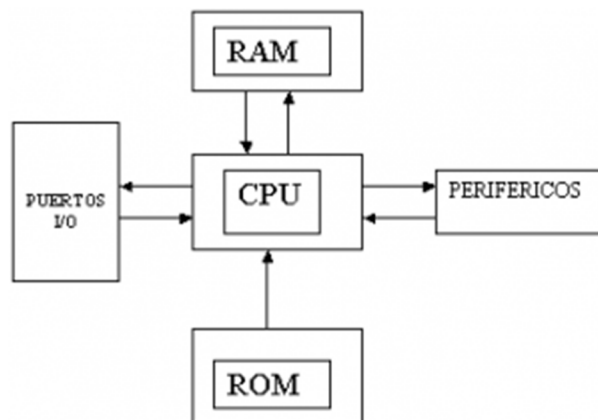


Ilustración 2.24 Arquitectura básica de un microcontrolador

Los microcontroladores son elegidos debido a que su uso es sencillo para aplicaciones de poca complejidad, además de que en internet existen muchos códigos compatibles, por su accesibilidad económica y por su bajo consumo de energía. [21]

Estos microcontroladores tienen una gran cantidad de aplicaciones, muchas de ellas aplicables a proyectos escolares como termómetros implementando un sensor de temperatura y una pantalla LCD, manejo de actuadores (motores, lumínicos, etcétera), aplicaciones de robótica, etcétera.

Los microcontroladores vienen en diferentes tipos de encapsulados y con diferentes cantidades de pines (ilustración 2.25), que van desde 8 hasta 40, que se eligen dependiendo su aplicación.

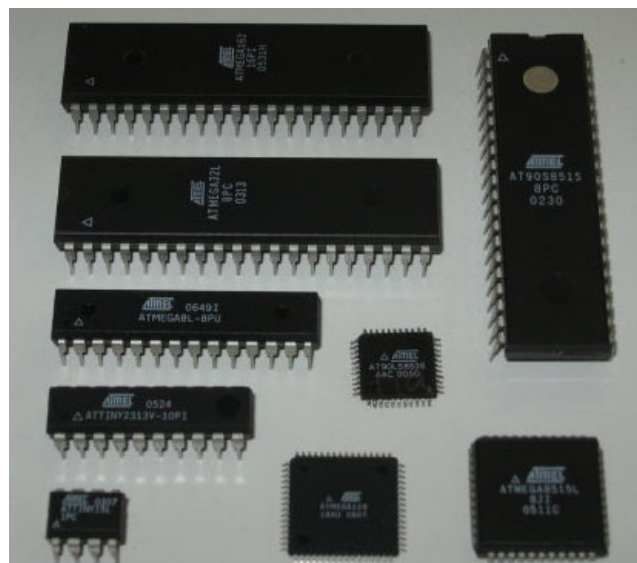


Ilustración 2.25. Tipos de encapsulados de un circuito integrado

Los microcontroladores deben ejecutar un programa previamente cargado en la memoria FLASH y es un conjunto de instrucciones de ceros y unos, este código se compone por palabras de hasta 16 bits, dependiendo la arquitectura. Pero en la realidad no se programan ingresando estos ceros y unos directo al microcontrolador, existen dos grupos de maneras de programación, los lenguajes de bajo nivel y los lenguajes de alto nivel. [22]

Los lenguajes de bajo nivel son lenguajes que están ligados de una manera muy directa a las instrucciones del hardware, estas instrucciones suelen ser muy complejas y específicas, por lo que se enfocan en trabajos que requieren una gran revisión de detalles. [23]

La ventaja de este tipo de programación es que las instrucciones son muy directas, es fácil de adaptar a diferentes modelos diferentes de microcontroladores, además de que trabaja a gran velocidad. Por otro lado, su mayor desventaja es que es una manera muy compleja de estructurar un programa y se necesita invertir mucho tiempo en diseñar y revisar cada programa. [23]

Existen tres tipos diferentes de códigos de bajo nivel, aunque todos tienen sus similitudes también son diferentes entre ellos: [23]

- ***Código binario***: es el lenguaje más básico de todos y el más famoso ya que de él surgen muchos conceptos de informática y de electrónica. Es el más simple de interpretar por los microcontroladores y por las computadoras ya que solo cuenta con dos datos diferentes, los unos que representan un estado alto y los ceros que representan un estado bajo.
- ***Lenguaje máquina***: es el más utilizado ya que, como lo dice su nombre, es la manera más directa de comunicarse con el hardware.
- ***Lenguaje ensamblador***: es el más complicado de los tres ya que el microcontrolador no lo descifra directamente y se tiene que traducir a lenguaje máquina para que se pueda llevar a cabo y para esto se necesitan programadores y compiladores, lo que lo hace más complejo.

Por otro lado, los lenguajes de alto nivel no se expresan considerando las capacidades del hardware, sino que toma en cuenta las capacidades de interpretación de las personas, por lo que es más simple desarrollar códigos complejos de una manera más simple y rápida. [24]

Este tipo de lenguaje tiene ventajas comparado con el anterior, algunas como que son más fáciles de aprender, por lo que el tiempo de capacitación es menor, las instrucciones son más similares a la forma de escribir de las personas, lo que lo hace más fácil de entender. Pero de igual manera presenta desventajas como que la compilación del programa es más lenta, ya que el compilador tiene que traducir el código a lenguaje máquina, no se explotan por completo los recursos del microcontrolador ya que las instrucciones no son tan precisas como en los lenguajes de bajo nivel y los programas se ejecutan de manera más lenta. [25]

Algunos de los lenguajes de programación más importantes son: [24]

- **Fortran**: fue creado por la empresa IBM en la década de los 50's, es considerado el primer lenguaje de alto nivel de la historia. Este lenguaje tiene una aplicación más enfocada a fines matemáticos y científicos ya que hace mucho énfasis en los aspectos numéricos, pero si el programa requiere otro tipo de comandos, Fortran no es muy útil.
- **Cobol**: es la abreviación de Common Business Oriented Language, un lenguaje creado a principios de los 60's y es especializado en aplicaciones de gestión. Su uso es muy simple ya que se basa en el uso del inglés de manera simple. Su gran desventaja es que requiere de gran detalle para representar el código.
- **BASIC**: creado a mitad de los años 60's, son las siglas de Beginner's All purpose, Symbolic Instruction Code, ya que buscaba convertirse en un estándar para la gente que se iniciaba en la programación. No es muy eficiente, pero con el tiempo ha ido evolucionando hasta convertirse en el actual Visual BASIC.
- **C**: nació en los 70's e intentó separarse de los lenguajes máquina para facilitar una nueva manera de ver los códigos de programación. Emplea todos los elementos de un lenguaje de alto nivel, pero a también tiene pequeños rasgos de los lenguajes de bajo nivel, lo que lo hace muy flexible y muy utilizado, pero a la vez lo coloca en discusiones sobre si realmente es un lenguaje de alto nivel, algunos lo consideran un lenguaje de nivel medio.
- **C++**: una evolución del lenguaje anterior, es creado en los 80's, agrega una mejor representación de los códigos de programación, pero con ello una mayor complejidad que su predecesor. Una desventaja de este es que es necesario aprender primero el lenguaje C, por lo que algunos prefieren quedarse con él.

2.5.2.- Tarjeta de desarrollo

Las tarjetas o placas de desarrollo son circuitos impresos que contienen un microprocesador o un microcontrolador. Están hechas para el desarrollo de proyectos de ingeniería por estudiantes o profesionales, aunque en los últimos tiempos se han simplificado tanto que ya son usadas con otros propósitos como la decoración, en los disfraces, etcétera. [26]

Hasta hace unos años elegir una tarjeta de desarrollo era bastante simple, para proyectos sencillos lo mejor era un Arduino, para proyectos que requieren más potencia de procesamiento o el uso de un sistema operativo Raspberry Pi era la mejor opción. Actualmente existen infinidad de empresas que crean tarjetas de desarrollo, por lo que cada vez la elección se hace más complicada (ilustración 2.26) [27]

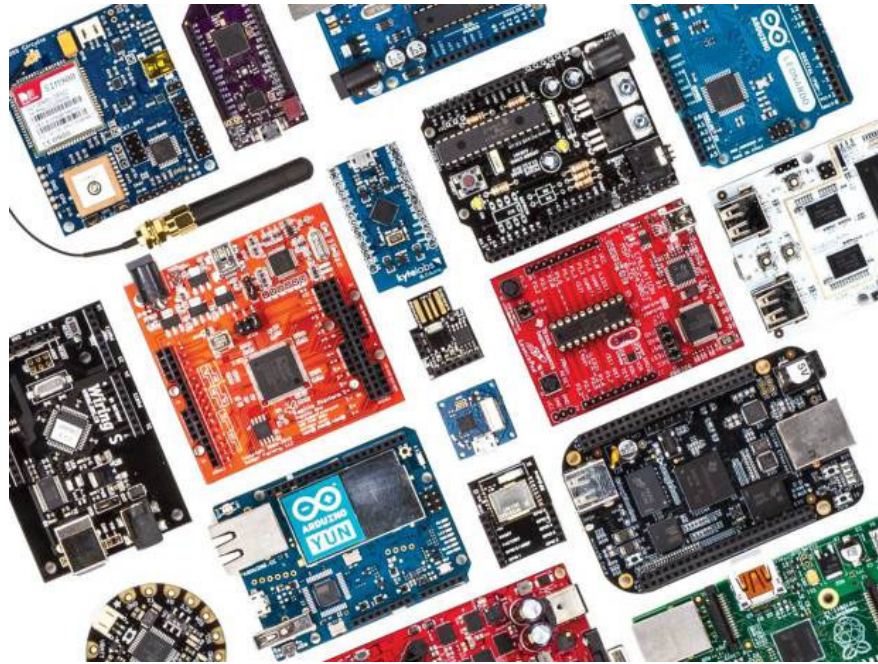


Ilustración 2.26 Imagen ilustrativa de tarjetas de desarrollo.

2.5.3.- ESP32

Hace unos años, el microcontrolador ESP8266 era el que se utilizaba en todas las aplicaciones que tenían que ver con el internet de las cosas, ya que contiene un módulo WiFi integrado, pero en 2016 la misma empresa, Espressif Systems, creó el microcontrolador ESP32 que es mucho mejor con respecto a su predecesor por lo que, actualmente, es una mejor opción para aplicaciones de IoT.

Este microcontrolador en su versión ESP – WROOM – 32 (ilustración 2.27) cuenta con las siguientes características: [28] [29] [30]

Tabla 1 Características del ESP32 vs ESP8266

	ESP32	ESP8266
Numero de núcleos	2	1
Arquitectura	32 bit	32 bit
Frecuencia del CPU	240 MHz	80 MHz
WiFi	Si	Si
Bluetooth	Si	No
RAM	512 KB	128 KB
FLASH	16 MB	4 MB
Pines de propósito general	36	16
Protocolos de comunicación	SPI, I2C, I2S, UART, CAN	SPI, I2C, I2S, UART
Resolución del ADC	12 bit	10
Resolución del DAC	8 bit	8

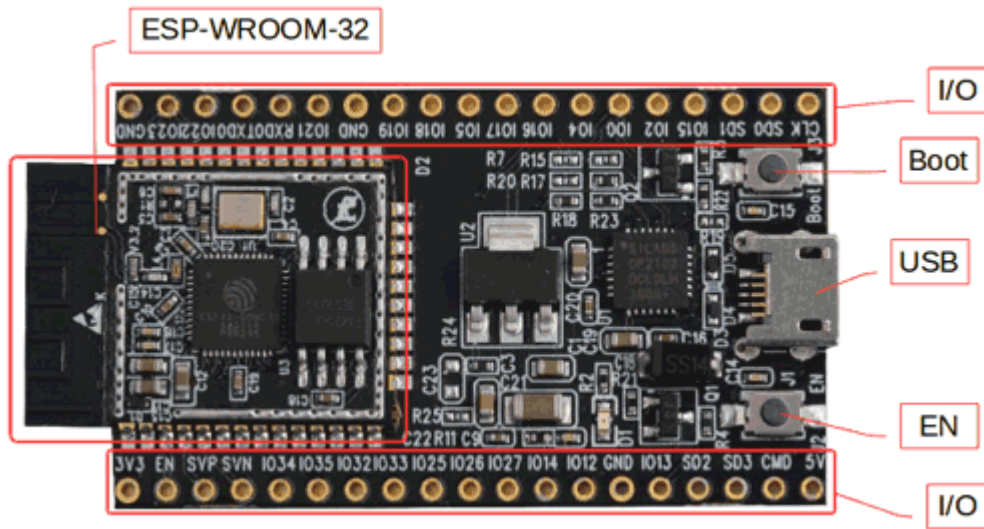


Ilustración 2.27 ESP-WROOM-32

2.5.4.- IDE Arduino

Un IDE es un “Entorno de Desarrollo Integrado” (del inglés Integrated Development Environment), es decir es un software mediante el cual se escribe, edita, compila y sube el código que se carga en las placas de arduino y compatibles conectados mediante un cable USB a una computadora. En la ilustración 2.28 se puede apreciar la pantalla principal de dicho entorno. [31]

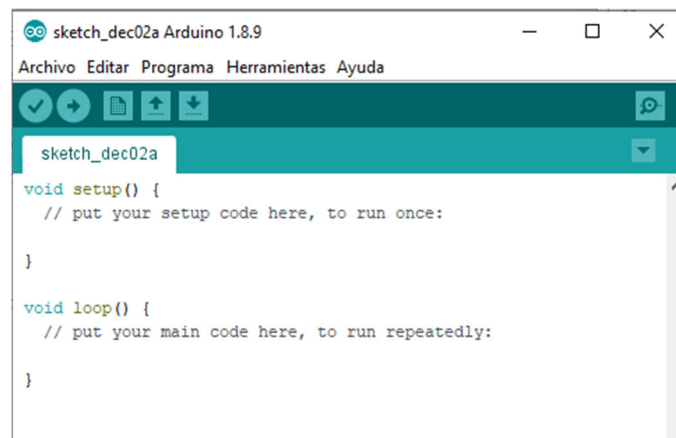


Ilustración 2.28 IDE Arduino

2.5.4.1.- Librerías de Arduino

En el contexto de programación, una librería, biblioteca o fichero son un conjunto de programas que hacen más fácil ejecutar un grupo de instrucciones relacionadas entre sí y que ayudan a que el código principal sea más sencillo. Algunas librerías requieren el uso de un hardware externo a la placa de desarrollo como una pantalla LCD, un motor a pasos, etcétera, otras sirven para comunicarse de una manera diferente al propio microcontrolador

de la tarjeta como la elección de la localidad de memoria donde se quiere escribir o leer datos.

En el proyecto mencionado en este documento se utiliza la siguiente librería:

- **CayenneMQTTESP32.h:** La librería es usada para conectar el microcontrolador con la nube de Cayenne. Después de escribir el código para conectar la tarjeta se puede enviar y recibir información, monitorear y controlar el dispositivo. Es librería soporta varios lenguajes y entornos de programación. [32]

2.6.- Plataforma Cayenne

Cayenne es la primera plataforma para desarrollar proyectos relacionados con internet de las cosas basada en arrastrar y soltar. Esta funcionalidad permite a creación de prototipos de forma rápida y fácil. Además de esto, la plataforma es completamente gratuita, lo que lo hace una opción ideal para estudiantes y desarrolladores de prototipos. [32] [33]

Para poder utilizar esta plataforma solo necesitamos una tarjeta de desarrollo con conexión a internet como Raspberry, Arduino, ESP8266, ESP32, etcétera, y un Smartphone o una computadora conectada a internet desde el cual crearemos nuestra aplicación (ilustración 2.29). [32]

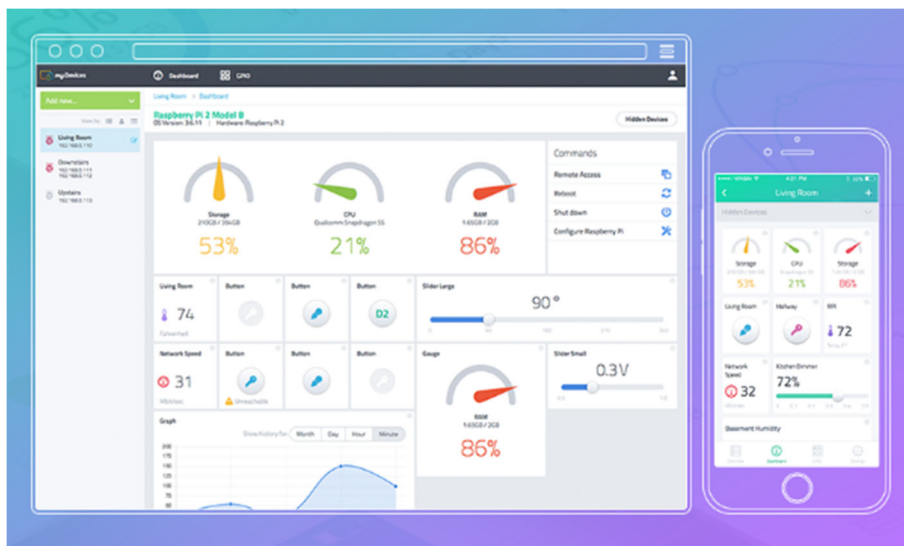


Ilustración 2.29 Pantalla ejemplo de la aplicación Cayenne

3.- Desarrollo experimental

Para el desarrollo del dispositivo se planeó poder controlar el encendido y apagado de una cafetera de calentamiento por resistencia a través de un dispositivo móvil con conexión WiFi. Para eso se diseñó un dispositivo basado en el microcontrolador ESP32 capaz de incorporarse a la cafetera y poder controlar su encendido y apagado, así como la regulación de la potencia para mantener la bebida caliente con el menor consumo de energía eléctrica posible. La aplicación de control será creada en la plataforma Cayenne mediante la cual se tiene una interfaz gráfica más didáctica e intuitiva para el usuario con un botón ON/OFF y un slider para configurar la potencia de trabajo en porcentaje. El microcontrolador recibe estos datos y los interpreta para activar la etapa de potencia. Para poder controlar la carga resistiva de 127V en corriente alterna se realizó un circuito con un optoacoplador que enviará la señal del ESP32 al gate de un triac para activarlo y desactivarlo de acuerdo a lo solicitado por la aplicación. Para hacer la sincronización entre la corriente alterna y la señal de activación del microcontrolador se implementó un detector de cruce de ceros, el cual envía una señal de interrupción al microcontrolador y este genera un delay de acuerdo a la potencia programada a modo de un PWM a 120Hz correspondiente a los dos semiciclos de la onda senoidal de la corriente alterna.

Para esto se dividió el prototipo en 4 etapas:

1. Implementación del circuito de cruce de ceros.
2. Diseño de la etapa de electrónica de potencia.
3. Desarrollo de la aplicación en la plataforma Cayenne.
4. Programación del código en el IDE Arduino.

3.1.- Diagrama de bloques

El siguiente diagrama (ilustración 3.1.) muestra el comportamiento básico del sistema integrado en el presente proyecto.

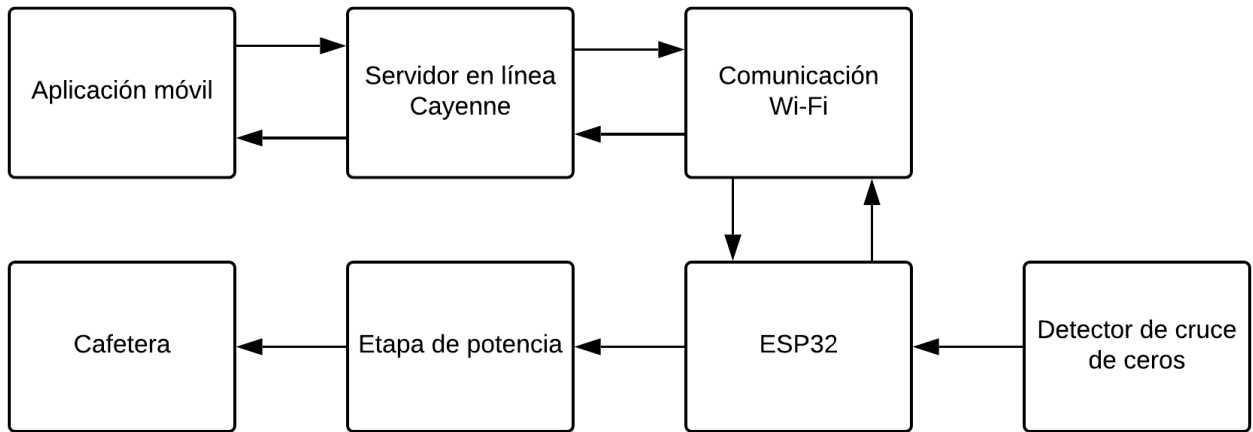


Ilustración 3.1 Diagrama de bloques del sistema

Donde:

- Cafetera: es la máquina que lleva acabo la preparación del café y que controlamos desde nuestro dispositivo.
- Etapa de potencia: es el circuito que acopla la señal del microcontrolador para poder controlar la cafetera.
- ESP32: es el microcontrolador con el que controlamos la señal de encendido y apagado del gate del triac de la etapa de potencia.
- Detector de cruce por cerros: circuito que envía una señal al microcontrolador para sincronizar con la frecuencia de la red de alimentación eléctrica.
- Comunicación Wi-Fi: protocolo de comunicación mediante el cual se comunican el microcontrolador, el servidor en línea y la aplicación móvil de Cayenne.
- Servidor en línea Cayenne: plataforma de desarrollo de aplicaciones para prototipos de internet de las cosas.
- Aplicación móvil: es desde donde se realiza el encendido, apagado y demás funciones del dispositivo.

3.2.- Diagrama esquemático del dispositivo

Como se mencionó antes, para el desarrollo de este dispositivo se consideraron cuatro etapas, dos de las cuales se implementaron de forma física, el circuito de potencia y el detector de cruce de cerros, ambos brevemente descritos en el Capítulo 2 y mostrados en las ilustraciones 2.9 y 2.11. A continuación se muestra el circuito completo del sistema (ilustración 3.2).

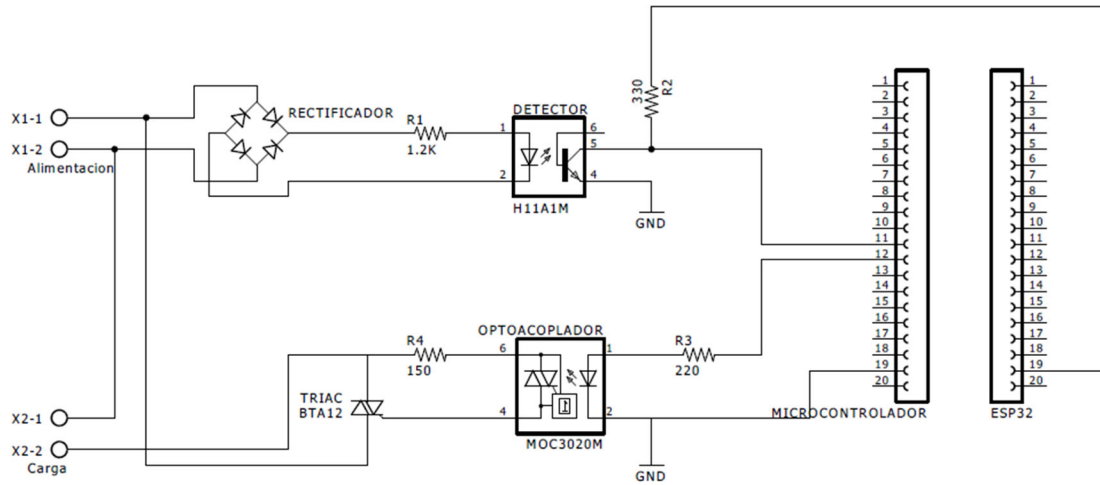


Ilustración 3.2 Diagrama esquemático

Donde:

- Alimentación: es la entrada de energía eléctrica de la red de 127V de corriente alterna.
- Carga: se refiere a la salida del sistema conectado a la resistencia de calentamiento de la cafetera.
- Rectificador: es una configuración de cuatro diodos para rectificar la corriente alterna y así tener una señal pulsante.
- Detector: es un circuito integrado que se activa con la señal pulsante del rectificador y envía un pulso al pin de interrupción de la ESP32.
- Microcontrolador: tarjeta de desarrollo basada en una ESP32 que lleva a cabo las funciones de control del sistema.
- Optoacoplador: este circuito integrado tiene la función de separar las señales electrónicas de las eléctricas para evitar daños en el dispositivo, cuando el microcontrolador envía un estado alto (o señal PWM) éste activa un foto triac interno que manda una señal de la alimentación al gate del triac.
- Triac: es una especie de interruptor que se enciende o apaga la resistencia de calentamiento de la cafetera de acuerdo a la señal que entra por al gate.

Con base en el diagrama esquemático anterior se diseñó la PCB mostrada en la ilustración 3.3, este diseño se hizo considerando en ancho de pistas necesario para la etapa de potencia, ya que según los datos de la cafetera, esta consume 700W, así que la corriente en onda completa se calcula de la siguiente manera:

$$I = \frac{P}{V} = \frac{700W}{127V} = 5.512A$$

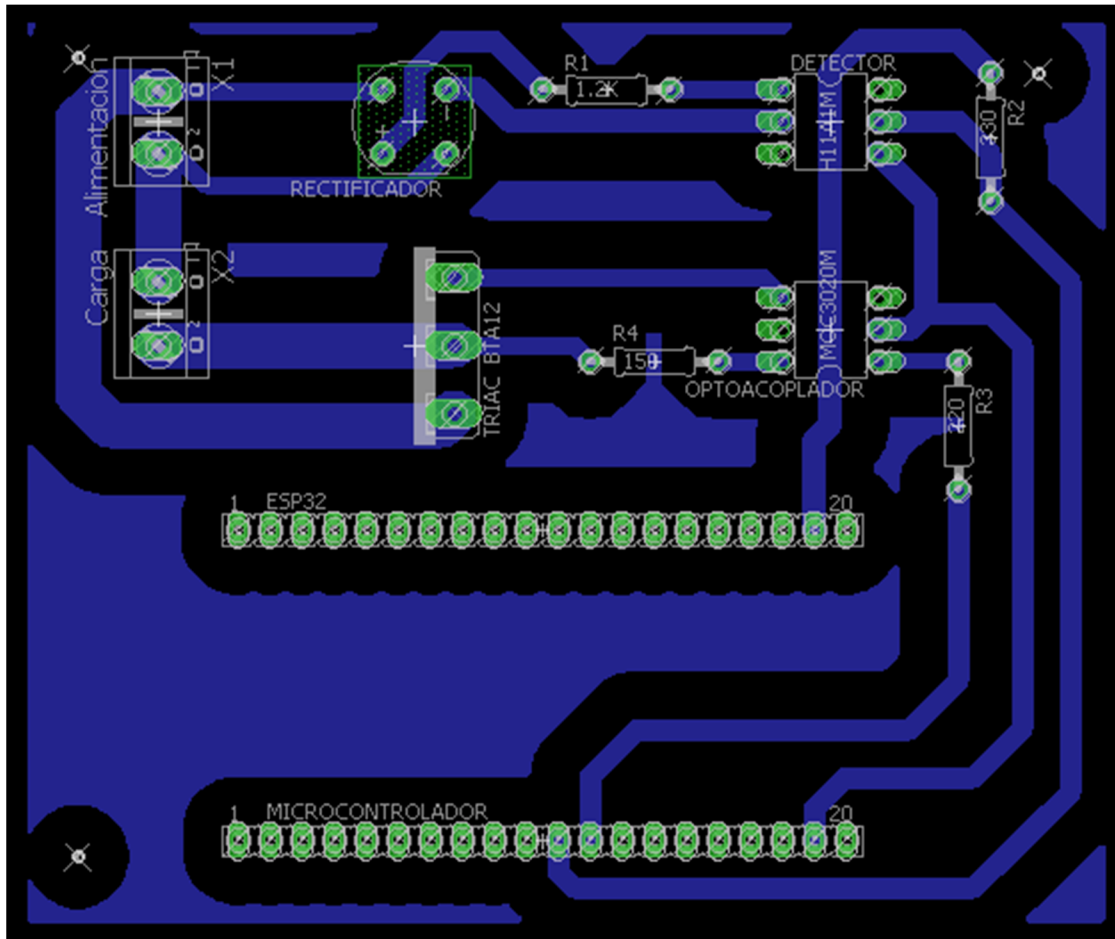


Ilustración 3.3 Capa inferior de la PCB del sistema.

Con la información anterior se calculó el ancho de las pistas mediante la aplicación que se puede encontrar en: <http://circuitcalculator.com/wordpress/2006/01/31/pcb-trace-width-calculator/> (ilustración 3.4) y, según el cálculo, se deben considerar de 3.16 mm pero se usó 3.5 mm para tener una mayor tolerancia.

Inputs:

Current	5.512	Amps
Thickness	1	oz/ft ² ▼

Optional Inputs:

Temperature Rise	10	Deg C ▼
Ambient Temperature	25	Deg C ▼
Trace Length	10	mm ▼

Results for Internal Layers:

Required Trace Width	8.23	mm ▼
Resistance	0.000613	Ohms
Voltage Drop	0.00338	Volts
Power Loss	0.0186	Watts

Results for External Layers in Air:

Required Trace Width	3.16	mm ▼
Resistance	0.00160	Ohms
Voltage Drop	0.00879	Volts
Power Loss	0.0485	Watts

Ilustración 3.4 Aplicación para cálculo de ancho de pistas

3.2.1.- Implementación del circuito de cruce por ceros.

Para la implementación de este sistema fue necesario implementar un puente rectificador ya que el circuito integrado H11AA1 contiene un foto diodo en su interior, el cual solo permite el paso de la corriente alterna en una dirección, por lo que no se hace la detección del cruce por cero en los semiciclos negativos. Con la rectificación el detector ya envía la señal de interrupción con ambos semiciclos de la onda (ilustración 3.5).

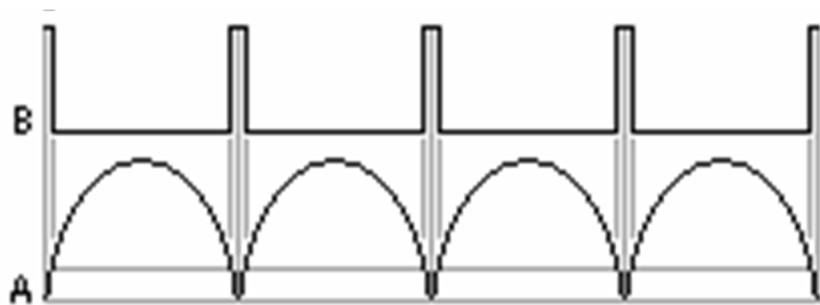


Ilustración 3.5 Muestra del cruce por cero de señal rectificada.

Para el cálculo de las resistencias de entrada del H11AA1 se utilizó un divisor de tensión considerando los datos de entrada de la ficha técnica del integrado que nos dice que los valores de entrada son 1.2V y 100mA. La tensión de la red de alimentación eléctrica es de 127V, por lo que primero es necesario calcular la tensión de corriente continua como se mencionó en el capítulo 2.4.1.1.

$$V_{CC} = 0.9 V_{ef} = 0.9 (127V) = 114.3V$$

Para el cálculo de la resistencia quedaría la siguiente formula:

$$R = \frac{V_{CC} - V_{LED}}{I} = \frac{114.3V - 1.2V}{100mA} = \frac{113.1V}{100mA} = 1131\Omega$$

Donde:

R – resistencia limitadora de corriente del LED infrarrojo del H11AA1.

V_{CC} – tensión de corriente continua que sale del rectificador.

V_{LED} – tensión de trabajo del LED infrarrojo.

I – corriente que debe circular por el circuito.

Dado que no existe una resistencia de 1131Ω, los valores comerciales de resistencias son 1KΩ y 1.2KΩ, se eligió esta última para no sobrepasar la corriente tolerada por el circuito integrado y evitar daños o la disminución del tiempo de vida del mismo.

A la salida del circuito integrado H11AA1 se colocó una resistencia en pull-up para enviar la señal de interrupción al microcontrolador, para el cálculo de esta resistencia también se utilizó la ley de Ohm considerando que el ESP32 tiene la limitación de cada pin de 12mA y el voltaje es de 3.3V, entonces:

$$R = \frac{V}{I} = \frac{3.3V}{12mA} = 275\Omega$$

Donde:

R – resistencia de pull-up.

V – voltaje de alimentación.

I – corriente máxima soportada por el microcontrolador.

En este caso las resistencias comerciales que existen son de 270Ω o 330Ω , una vez más se eligió la de valor más grande para evitar daños o disminuir la vida útil del microcontrolador.

3.2.2.- *Diseño de la etapa de potencia*

Para esta etapa se tuvieron menos consideraciones que en la anterior, pero de igual modo el cálculo de las resistencias para el LED infrarrojo del MOC y para el gate del triac. Para este primero se considera que la corriente máxima que tolera es de 60mA , según la ficha técnica, y la tensión máxima del microcontrolador de 3.3V , por lo que la resistencia es:

$$R_{MOC} = \frac{V_{ESP32}}{I_{MOC}} = \frac{3.3\text{V}}{60\text{mA}} = 55\Omega$$

Donde:

R_{MOC} – resistencia limitadora de corriente del LED infrarrojo del MOC.

V_{ESP32} – voltaje que suministra el microcontrolador.

I_{MOC} – corriente máxima soportada por el LED infrarrojo del MOC.

Para este caso se tomó la resistencia de 56Ω por ser el valor comercial más cercano.

En el caso de la salida del optoacoplador y entrada de la compuerta del SCR, las fichas técnicas nos dicen que tienen una tolerancia de 1A y 4A , respectivamente, por lo que tomaremos en cuenta la corriente del MOC para el cálculo de la resistencia ya que es menor.

$$R_G = \frac{V_{CA}}{I_{MOC}} = \frac{127\text{V}}{1\text{A}} = 127\Omega$$

Donde:

R_G – resistencia limitadora del gate del triac.

V_{CA} – voltaje de corriente alterna.

I_{MOC} – corriente máxima soportada por el optotriac del MOC.

Entonces la resistencia mínima que se puede utilizar es de 127Ω , en este caso se optó por utilizar la de 150Ω ya que es el valor comercial más pequeño sin sobrepasar el valor calculado.

3.3.- Desarrollo de la aplicación en la plataforma Cayenne

En el desarrollo de la aplicación se utilizó la plataforma Cayenne, ya que cumple con la función de servidor y desarrollador de aplicaciones, por lo que ahorra mucho trabajo en comparación a si se hiciera por separado. Se utilizó un botón para el encendido y apagado del sistema y dos sliders que se manipulan para ajustar el porcentaje de potencia en el que queremos que nuestra cafetera mantenga la temperatura después de preparar el café (de 0% a 100%) y el otro se ajusta para configurar el tiempo máximo de encendido, en caso del modelo RC-8016B de la empresa RCA (el que se utilizó), que fue el utilizado en este proyecto, el fabricante recomienda un tiempo de encendido máximo de una hora, por lo que el tiempo máximo del slider se configuró en 60 minutos.

Para crear la aplicación en la plataforma Cayenne (ilustración 3.6) se deben seguir los pasos descritos a continuación.

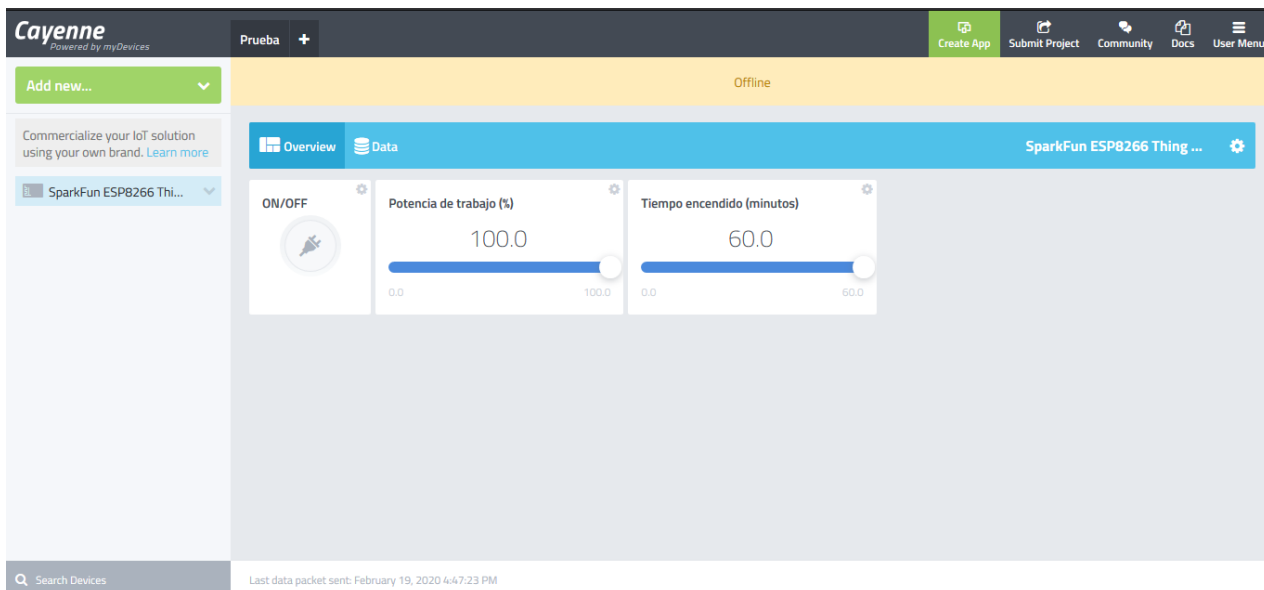


Ilustración 3.6 Aplicación de escritorio en la plataforma Cayenne.

1. Se debe ingresar al enlace cayenne.mydevices.com y aparecerá la pantalla mostrada en la ilustración 3.7. Si ya se tiene una cuenta solo será necesario iniciar sesión (pasar al paso 3), se no ser así seleccionar en la parte inferior “SIGN UP” para crear una.

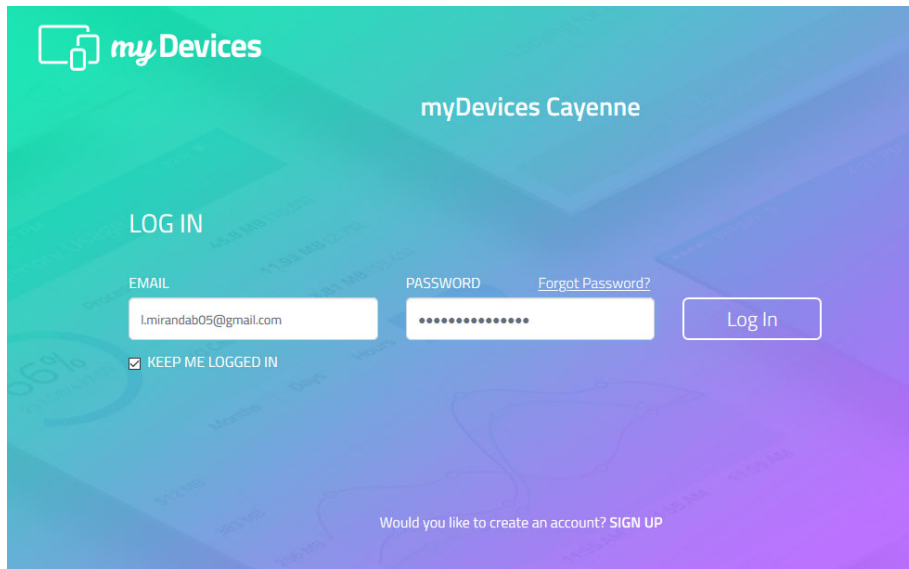


Ilustración 3.7 Pantalla principal de Cayenne.

2. Una vez que iniciamos el registro nos aparecerá la pantalla donde nos solicitará un nombre, correo y contraseña (ilustración 3.8), solo basta con llenar estos campos y presionar “NEXT” para empezar a crear la aplicación.

Ilustración 3.8 Pantalla de registro Cayenne.

3. Una vez creada la cuenta aparecerá la pantalla donde elegiremos el dispositivo con el que vamos a trabajar (ilustración 3.9), para este proyecto se utilizó un ESP32, el cual no aparece en la pantalla principal, por lo que seleccionaremos “All Devices” y nos enviará a otra pantalla (ilustración 3.10) donde aparecen más dispositivos, aquí elegiremos alguno de los modelos de la tarjeta ESP8266, aunque no son el mismo modelo, son del mismo fabricante y funcionan igual para esta aplicación.

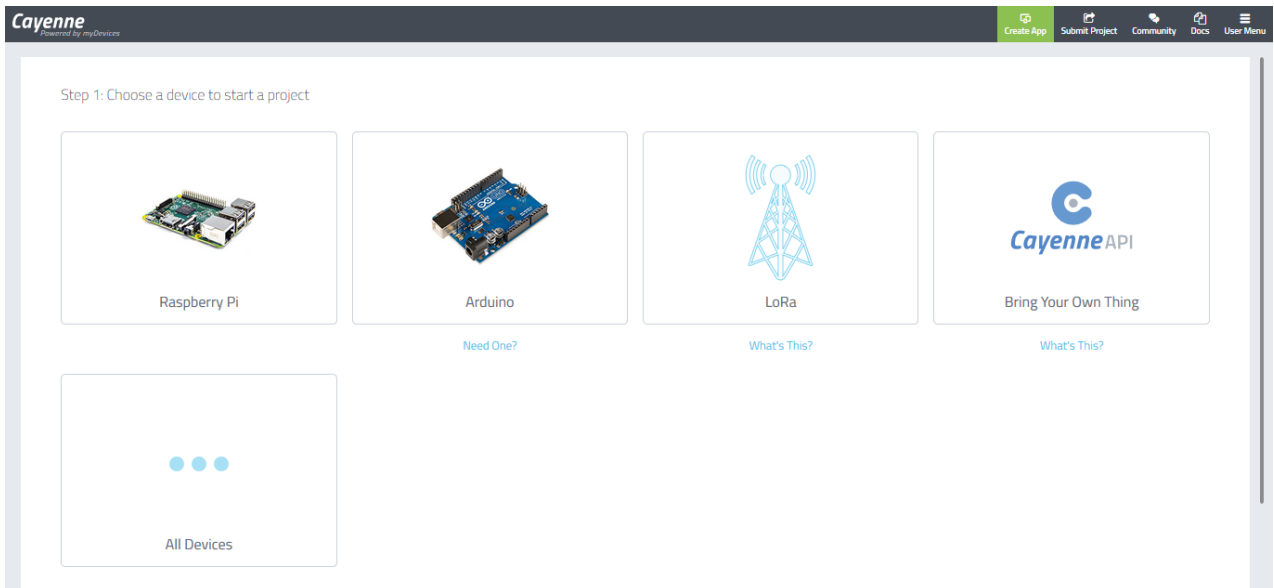


Ilustración 3.9 Pantalla de selección de dispositivo Cayenne

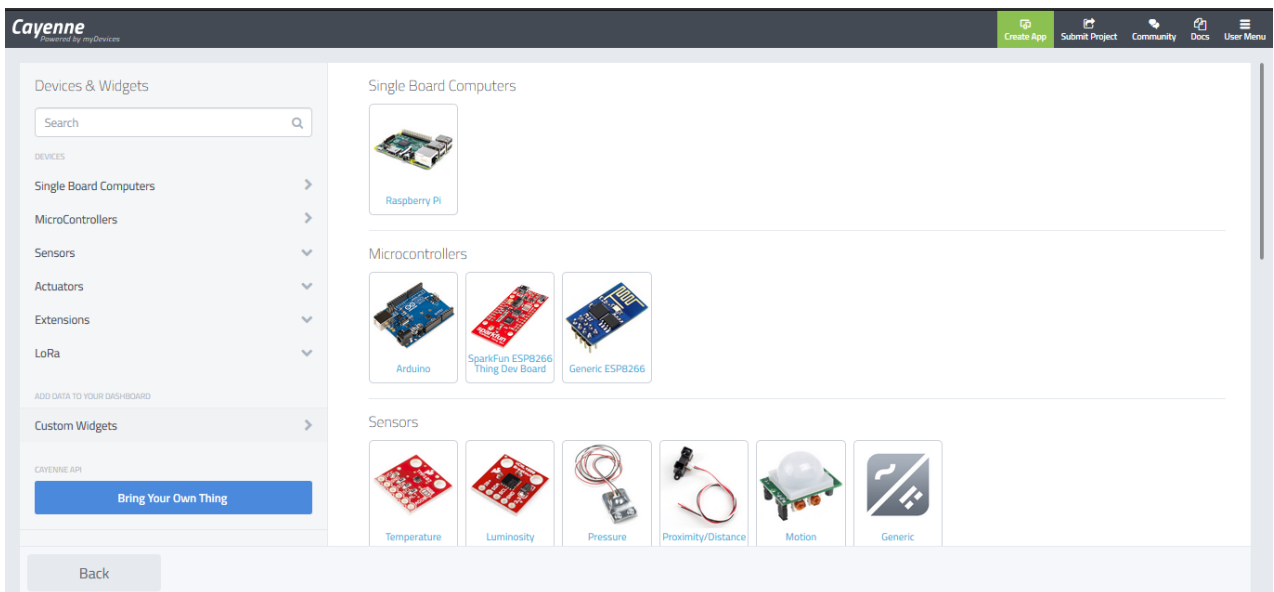


Ilustración 3.10 Pantalla con todos los dispositivos Cayenne

4. Una vez que seleccionamos el dispositivo nos enviara a una pantalla con la información para iniciar la comunicación entre la tarjeta y la plataforma (ilustración 3.11) donde se mantendrá hasta que se cargue el programa al microcontrolador. En este punto se le pone el nombre al proyecto en el recuadro de “NAME YOUR DEVICE”. La conexión con la tarjeta continuará en el apartado 3.4, ya que en este apartado solo se muestra el desarrollo de la aplicación.

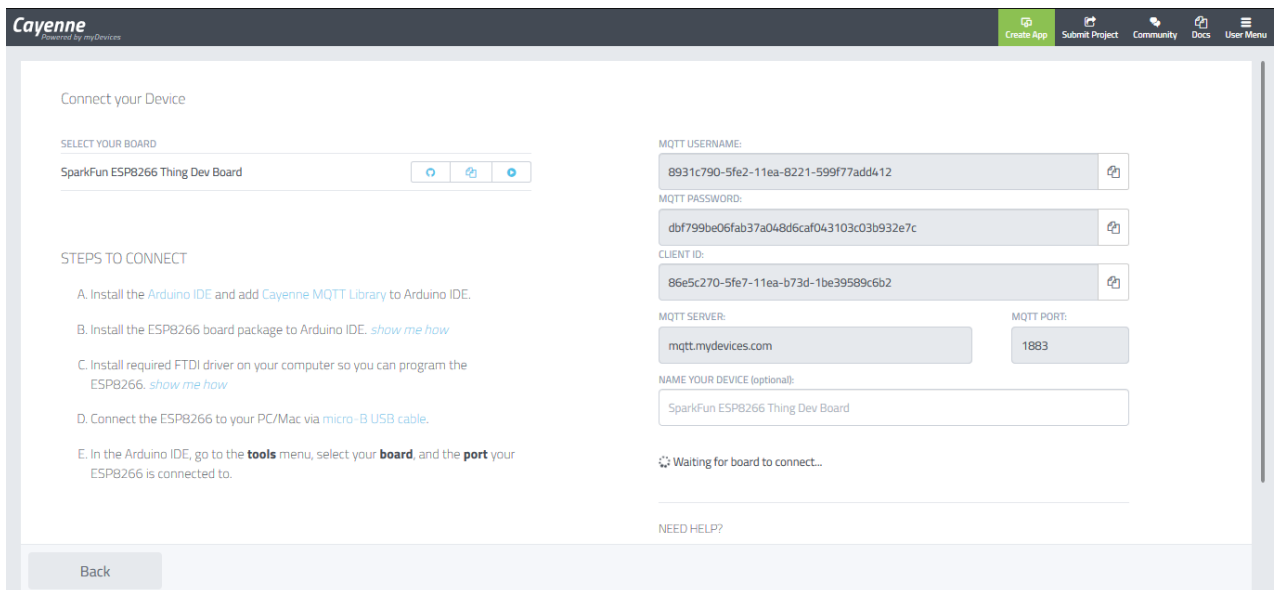


Ilustración 3.11 Pantalla de información Cayenne.

- Una vez conectada la tarjeta a la plataforma aparecerá una pantalla con video tutoriales de como continuar con la aplicación (ilustración 3.12), del lado izquierdo aparecerán los proyectos, seleccionaremos el nuestro de acuerdo al nombre que le pusimos en el paso anterior, vamos a seleccionarlo y arriba en el recuadro “Add new...” seleccionamos “Device/Widget” (ilustración 3.13).

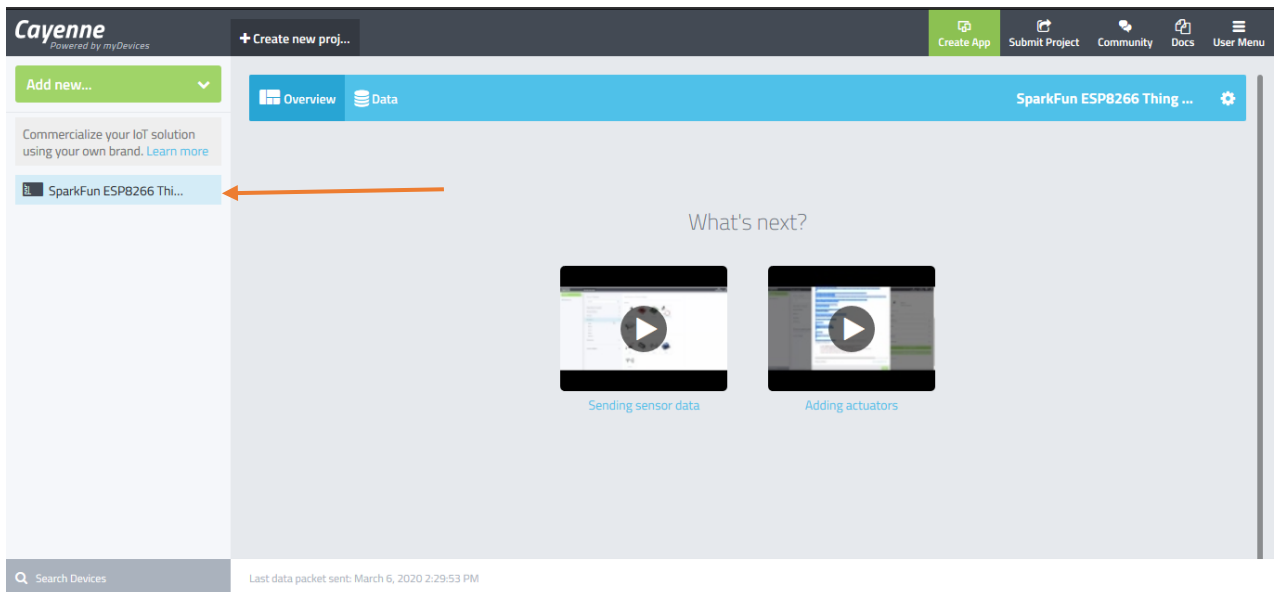


Ilustración 3.12 Pantalla de inicio del proyecto Cayenne.

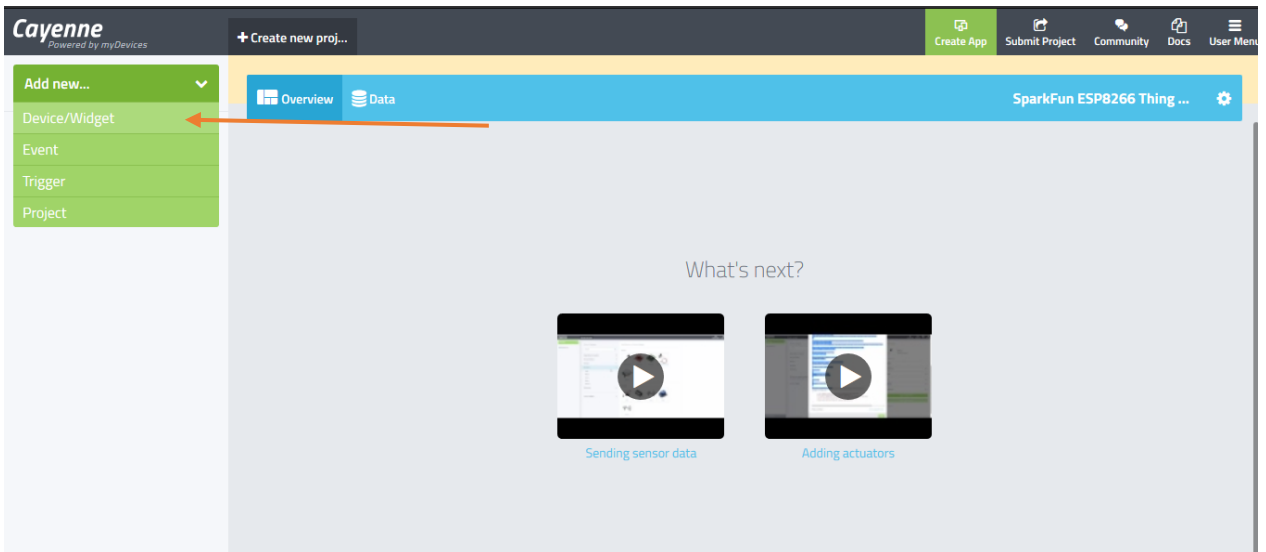


Ilustración 3.13 Insertar un nuevo widget en Cayenne.

- Nos aparecerá una pantalla similar a la de la ilustración 3.8, navegaremos dentro de ella hasta la sección de “Custom widgets” y elegiremos un botón (ilustración 3.14).

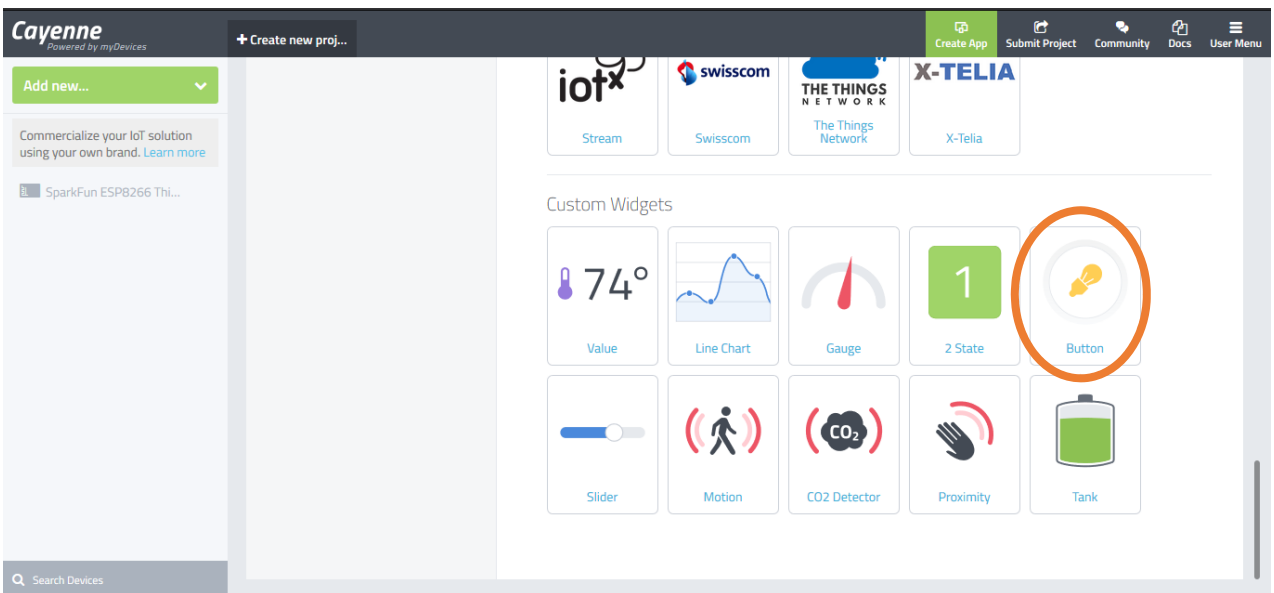
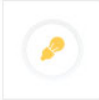


Ilustración 3.14 Menú de Widgets.

- Nos aparecerán las configuraciones del botón y se debe configurar como se ve en la ilustración 3.15 y al final “Add Widget”. Se puede elegir el nombre e icono que mejor les parezca.

Enter Settings



Button
Controller Widget

Name:

Device:

Sensor

Data:

Unit:

Channel:

Choose Icon:

[Step 1: Code](#)

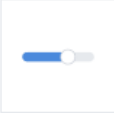
[Add Widget](#)

[Docs: Using MQTT with Cayenne](#)

Ilustración 3.15 Configuraciones del Widget.

- De igual modo se agregarán dos sliders para controlar la potencia de trabajo (en porcentaje) y el tiempo que se va a mantener encendido el dispositivo (en minutos) (ilustración 3.16).

Enter Settings



Slider
Controller Widget

Name:

Device:

Sensor

Data:

Unit:

Channel:

Slider Min Value (optional):


Slider Max Value (optional):

[Step 1: Code](#)

[Add Widget](#)

[Docs: Using MQTT with Cayenne](#)

Enter Settings



Slider
Controller Widget

Name:

Device:

Sensor

Data:

Unit:

Channel:

Slider Min Value (optional):

Slider Max Value (optional):

[Step 1: Code](#)

[Add Widget](#)

[Docs: Using MQTT with Cayenne](#)

Ilustración 3.16 Configuración de los sliders.

9. Una vez teniendo los tres Widgets agregados podemos cambiar su tamaño y posición como mejor nos parezca para verlos en nuestra pantalla principal como se ve en la ilustración 3.4

3.3.1.- Uso de la aplicación

Esta aplicación se encuentra disponible tanto en Android como en iOS, además de poder usar la aplicación de escritorio a través de una computadora o laptop, en este apartado se describe cómo usarla en las tres plataformas.

1. Primero se debe descargar la aplicación para los teléfonos móviles desde la App Store, en iOS, o Google Play, desde Android (ilustración 3.17).

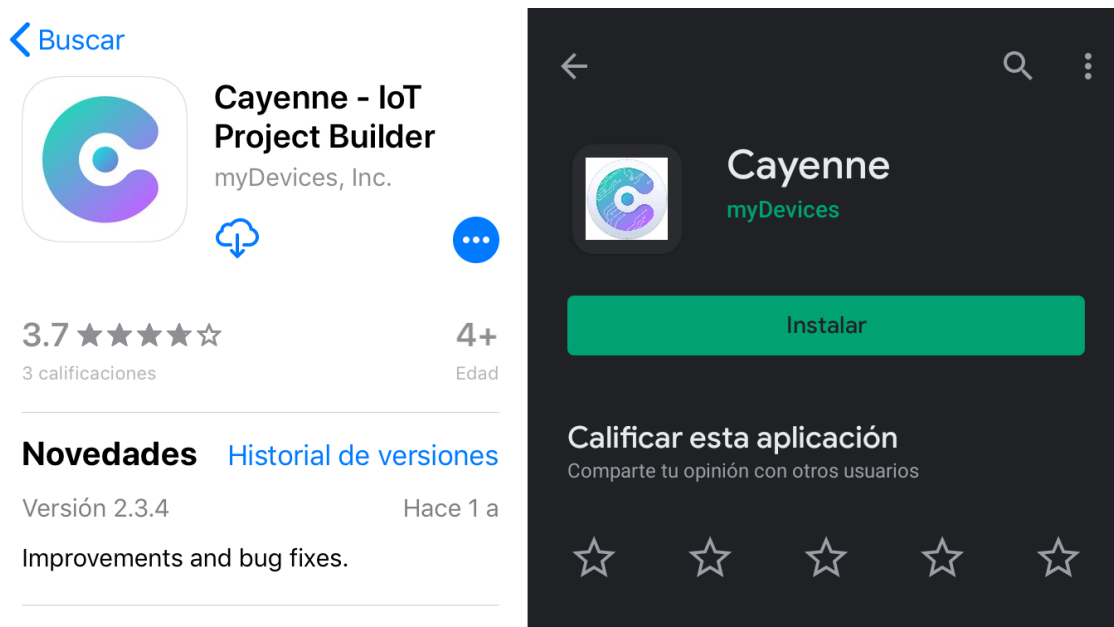


Ilustración 3.17 Tiendas de aplicación iOS (izquierda) y Android (derecha).

2. Una vez instalada la aplicación entraremos en ella o accederemos al enlace cayenne.mydevices.com, en caso de hacerlo desde la aplicación de escritorio. Para cualquier caso iniciaremos sesión con la cuenta creada anteriormente. (ilustración 3.18).

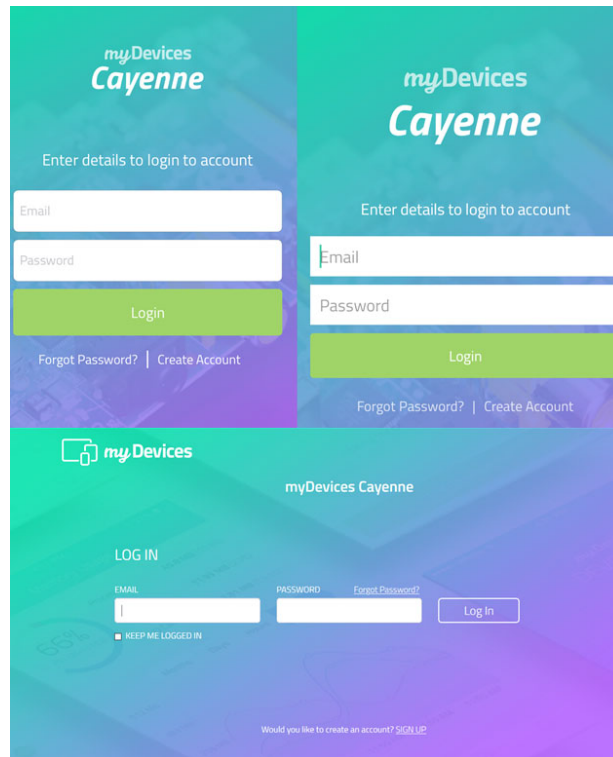


Ilustración 3.18 Pantalla de inicio de sesión Cayenne iOS (arriba izquierda), Android (arriba derecha) y escritorio (abajo).

3. Para la aplicación móvil, nos aparecerá el nombre de nuestro dispositivo en la pantalla de inicio (ilustración 3.19), debemos presionar sobre él para entrar a la aplicación. Para la versión de escritorio nos envira directo a la aplicación (ilustración 3.20).

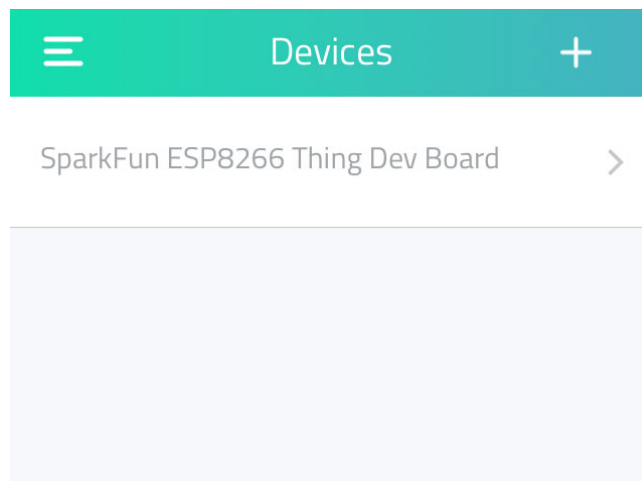


Ilustración 3.19 Pantalla principal de las aplicaciones móviles.

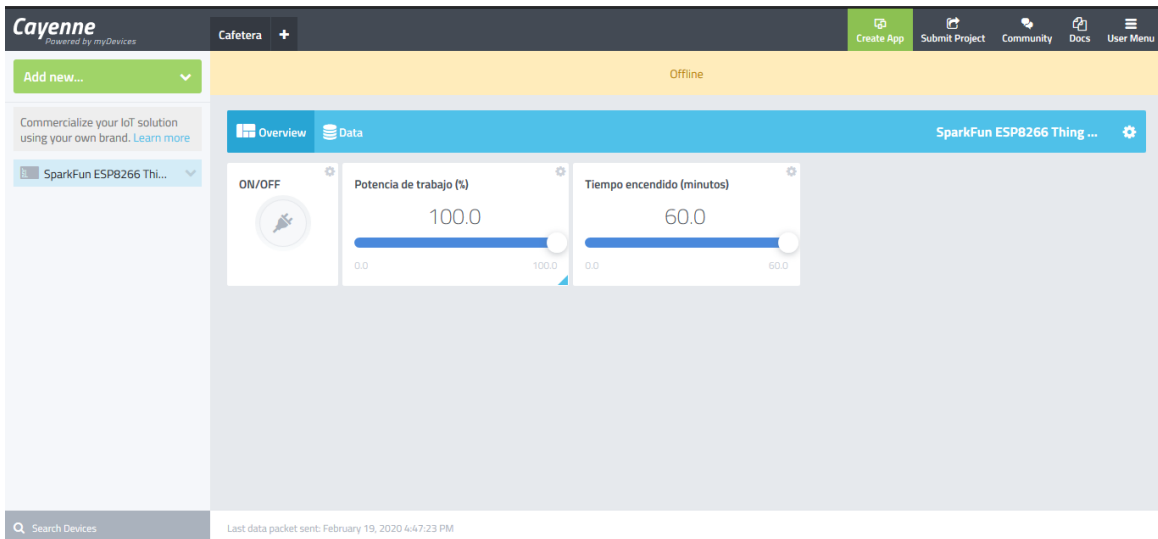


Ilustración 3.20 Pantalla principal de la aplicación de escritorio.

- Para iniciar el proceso solo es necesario ajustar el tiempo que va a estar encendida la cafetera y la potencia a la que trabajará una vez que termine la preparación para mantener el café caliente y por ultimo presionar el botón ON/OFF (ilustración 3.21).

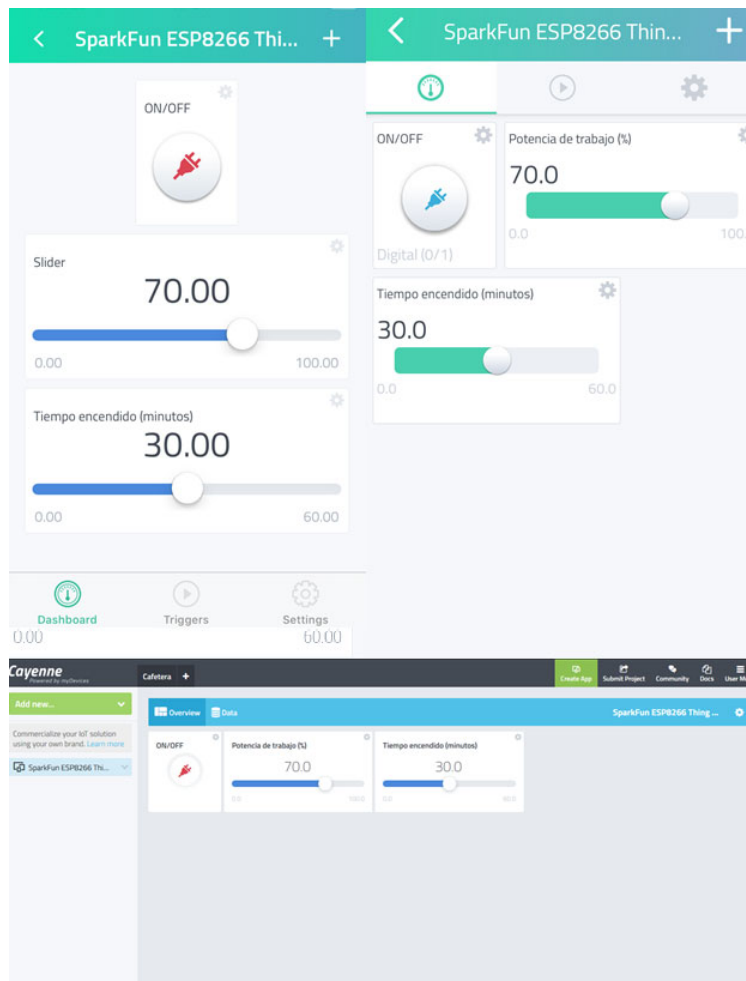


Ilustración 3.21 Pantalla de aplicación de Cayenne iOS (arriba izquierda), Android (arriba derecha) y escritorio (abajo).

De igual modo el dispositivo se puede apagar sin necesidad de que complete el tiempo, simplemente se presiona de nuevo el botón ON/OFF.

5. Para las plataformas móviles la aplicación presenta algunas limitaciones, mientras que en la versión de escritorio no, la que nos interesa es la programación horarios de encendido.

Para utilizar esta función basta con presionar en el menú de usuario que se encuentra en la parte superior derecha y dar click en Scheduling (ilustración 3.22).

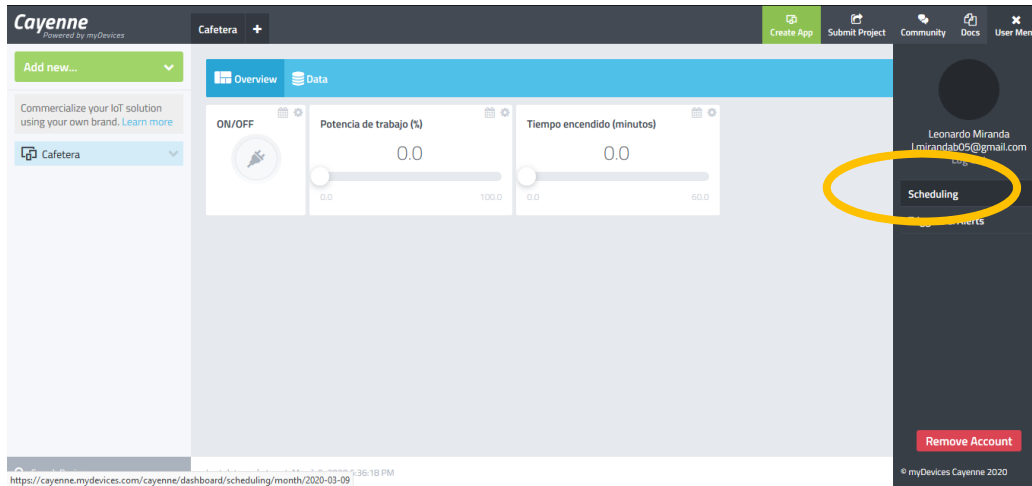


Ilustración 3.22 Menú de usuario.

Después nos aparecerá un calendario mensual (ilustración 3.23), se puede cambiar la vista por año, mes, semana o día, según se prefiera. Se debe presionar Add New Event para crear un nuevo programa.

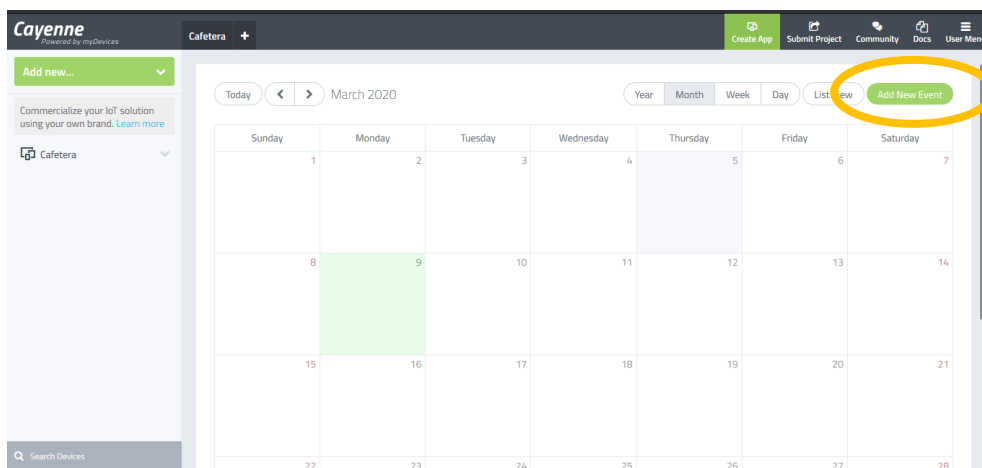


Ilustración 3.23 Vista de calendario Cayenne.

Aparecerá una ventana donde se debe configurar las acciones del programa (ilustración 3.24). Primero se debe poner un nombre, después programar el día y hora en que se quiere encender el dispositivo (la plataforma cuenta con las opciones de cambiar la zona horaria o enviar notificaciones vía MSN o mail, pero para el presente propósito no son necesarias), también se debe configurar si se quiere que se repita la acción y por ultimo configurar las acciones.

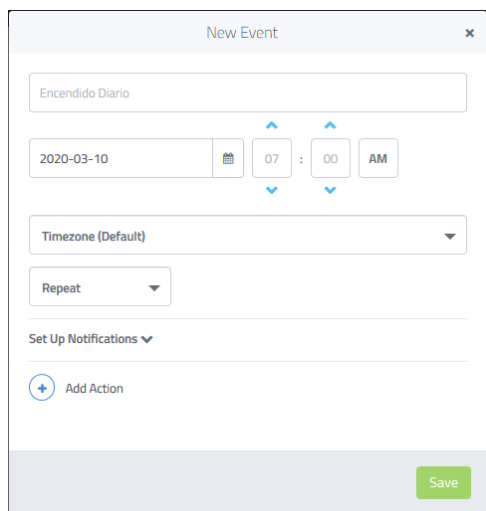
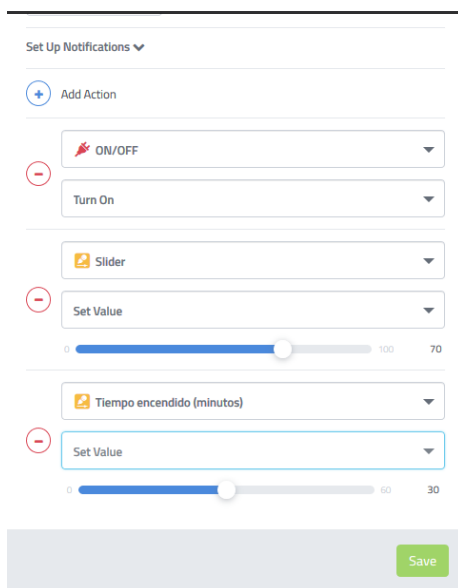


Ilustración 3.24 Ventana de configuración de evento.

Para configurar las acciones se debe presionar Add Actions. Se deben agregar tres acciones: uno para el encendido, en el segundo se debe seleccionar el slider correspondiente a la potencia de trabajo y ajustar el valor deseado y en el tercero el slider del tiempo de encendido y ajustarlo de igual manera (ilustración 3.25) y por ultimo presionar Save para guardar el evento.



Después aparecerá la vista con los eventos programados (ilustración 3.26).

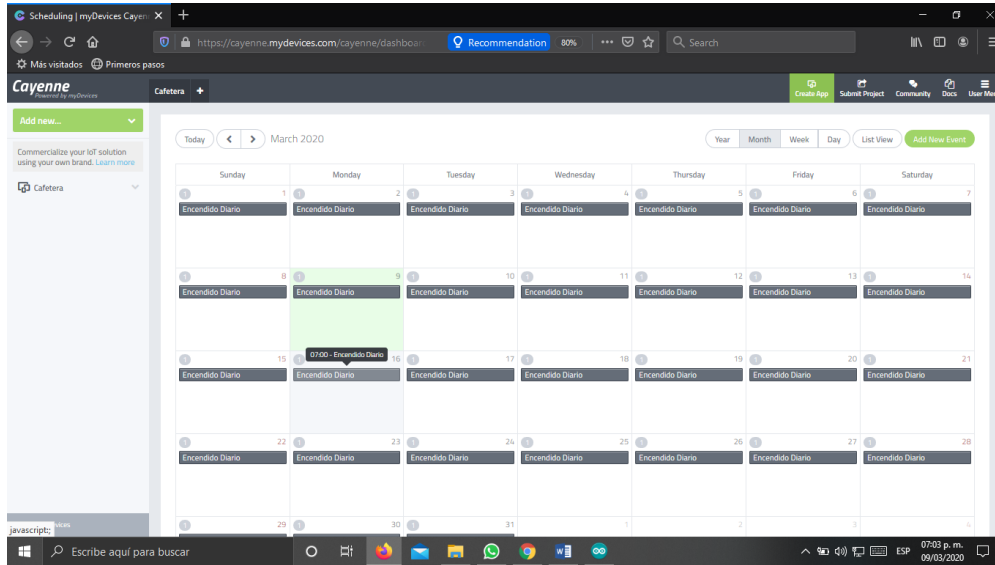


Ilustración 3.26 Evento diario programado.

3.4.- Programación del código en el IDE Arduino

Antes de empezar con la programación del código de control de nuestro ESP32 se configuró el IDE de Arduino para poder establecer comunicación con la tarjeta. Primero se descargaron los controladores, para eso se puso el enlace https://dl.espressif.com/dl/package_esp32_index.json en el apartado “Gestor de URLs Adicionales de Tarjetas” en las preferencias del IDE, como se muestra en la ilustración 3.27.

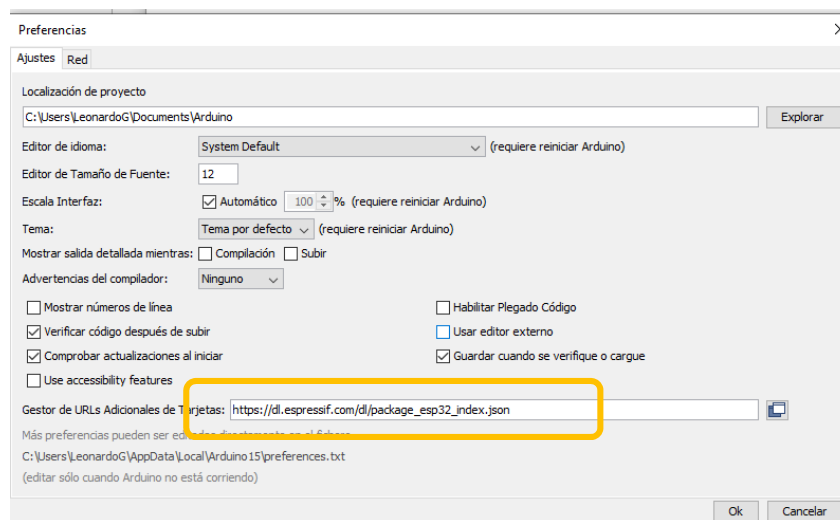


Ilustración 3.27 Preferencias del IDE de Arduino

Después de eso se descargó el controlador de las tarjetas ESP32, desde el gestor de tarjetas. Se escribió en la barra de búsqueda “ESP32” y se instaló la primera opción que aparece (ilustración 3.28).

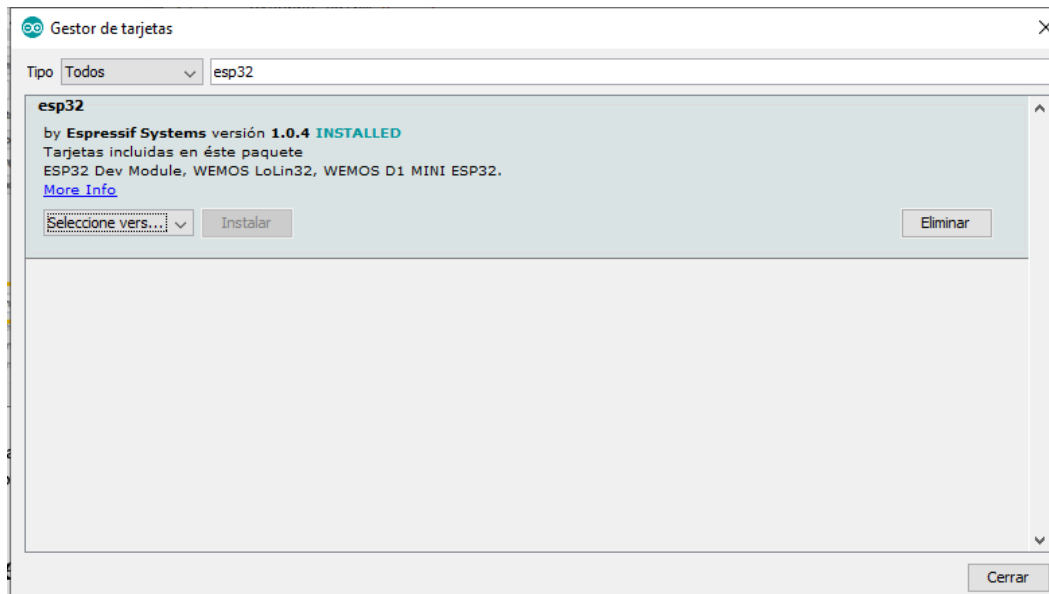


Ilustración 3.28 Gestor de tarjetas del IDE de Arduino

Se utilizó únicamente la librería CayenneMQTTESP32.h, para poder utilizarla se descargó desde el enlace que se muestra en la ilustración 3.9, en el inciso A (<https://github.com/myDevicesIoT/Cayenne-MQTT-ESP>), el cual nos envía a la página de GitHub que se muestra en la ilustración 3.29. Ahí se le dio click en “Clone or download” y después “Download ZIP”.

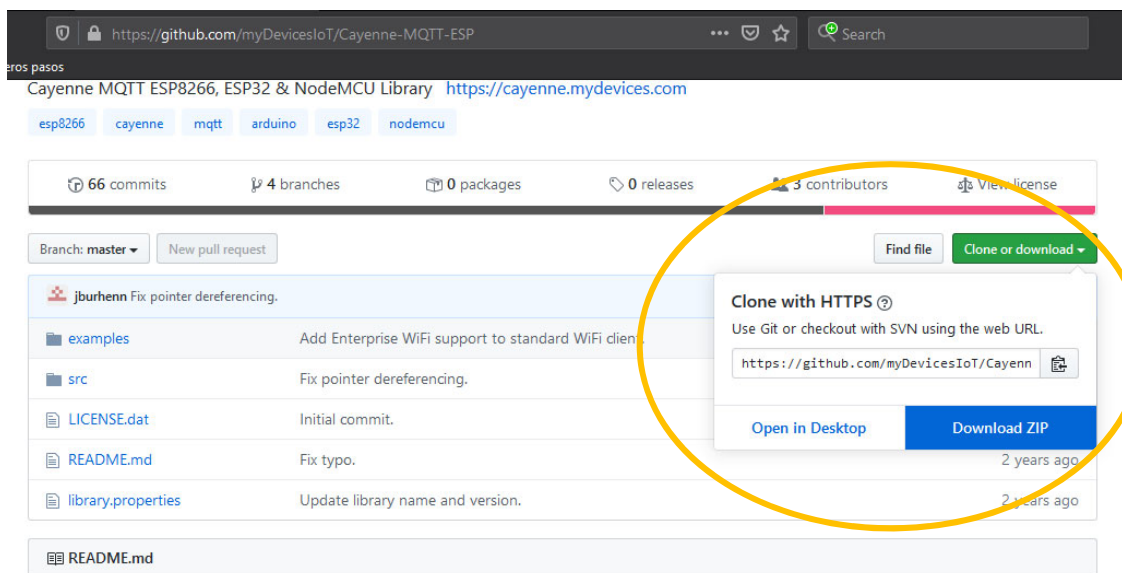


Ilustración 3.29 Pagina de descarga de la librería.

Para instalar la librería en el IDE de Arduino se hizo de la siguiente manera. En el menú “Programa” del IDE, después “Incluir librería” y “Añadir biblioteca .ZIP” (ilustración 3.30). Para finalizar se seleccionó la carpeta donde se descargó el archivo, se hizo click en el archivo y “Aceptar”. Y ya se pudo utilizar la biblioteca sin problemas.

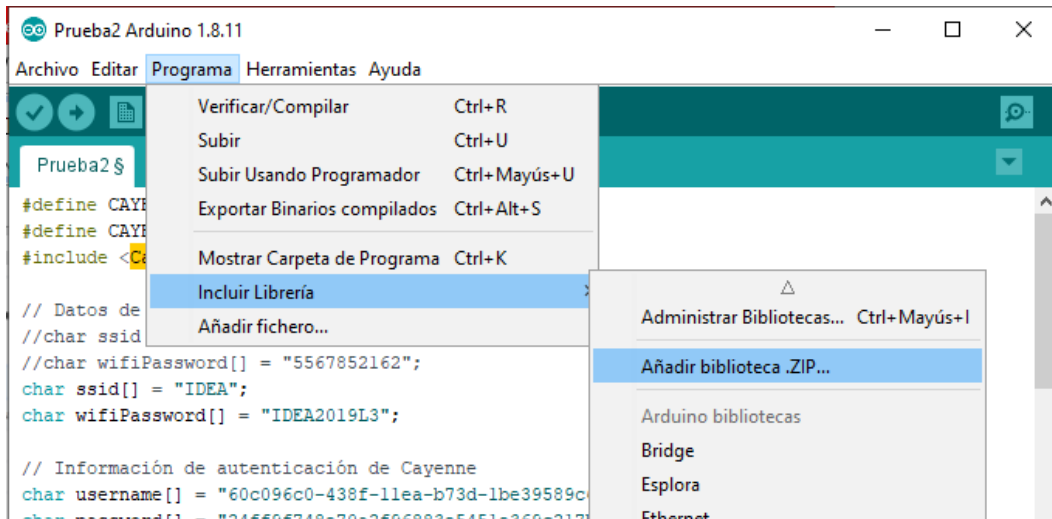


Ilustración 3.30 Menú del IDE de Arduino para añadir librería.

Una vez que se hicieron los ajustes anteriores se procedió a iniciar con la programación del código de control de acuerdo al diagrama de bloques de la ilustración 3.31.

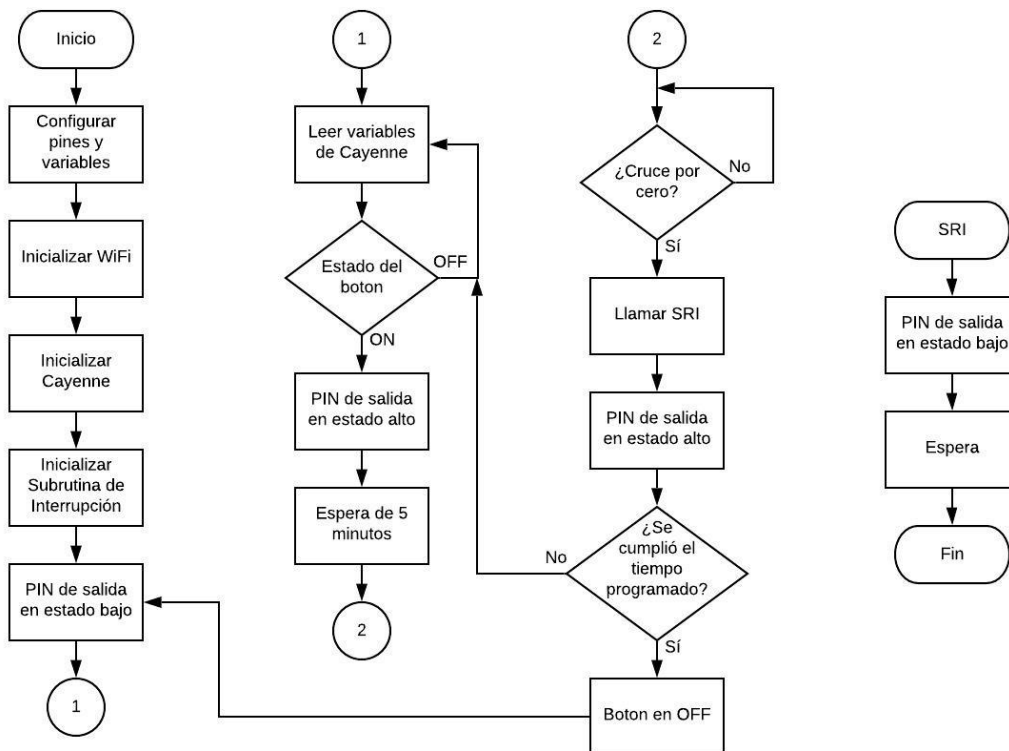


Ilustración 3.31 Diagrama de flujo del código

En el primer proceso se configuraron los pines de entrada y salida, así como las variables que se utilizaron en el código. Entre estas variables se agregaron unas de tipo char que sirvieron para la inicialización del WiFi y de Cayenne en los siguientes procesos, estas variables fueron:

- `char ssid[] = "*****";`
- `char wifiPassword[] = "*****";`
- `char username[] = "00000000-0000-0000-0000-000000000000";`
- `char password[] = "0000000000000000000000000000000000000000";`
- `char clientID[] = "00000000-0000-0000-0000-000000000000";`

Las variables `ssid` y `wifiPassword` son las correspondientes al nombre y la contraseña de la red, respectivamente.

Y las variables `username`, `password` y `clientID` son necesarias para establecer conexión con el proyecto de Cayenne, estas se encuentran en las configuraciones de la aplicación, de igual forma se vieron en la ilustración 3.9.

Una vez que se inicializan todos los procesos, se envía un estado bajo para asegurar que la salida se encuentre apagada al iniciar el proceso. Para iniciar el proceso se deben configurar los sliders del porcentaje de potencia al que va a trabajar después de que termine la preparación y el tiempo total de encendido de la cafetera y por ultimo presionar el botón de encendido.

Ya iniciado el proceso la cafetera trabajará a su máxima potencia durante 5 minutos, que es el tiempo que este modelo requiere para completar la preparación con el depósito de agua lleno, posteriormente va a reducir su potencia, esto mediante el recorte de onda que genera la subrutina de interrupción (SRI), que se activa cuando el detector de cruce por cero envía una señal al PIN de interrupción del microcontrolador, al apagar la salida que enciende el SCR y esperar un tiempo antes de activarla, se puede calcular el ángulo de disparo de acuerdo con la ecuación vista en el capítulo 2.4.2.

El dispositivo se mantendrá en este modo similar al PWM hasta que se cumpla con el tiempo total que se programó donde se cambiará el estado del botón y se mantendrá a la espera de una nueva señal.

3.5.- Fabricación de la placa PCB

La placa PCB se hizo por el método de planchado, ya que es la forma más fácil de hacer una tarjeta no profesional. Para esto es necesaria una impresora láser y un papel especial, se pueden usar algunos como transfer o acetatos, pero en esta ocasión se utilizó papel cuché, primero se imprime las capas de pistas en el papel y se plancha en la placa fenólica, previamente limpiada para evitar que la suciedad o grasa eviten que la tinta se pegue al cobre (ilustración 3.32). Si alguna pista no quedó bien planchada se puede rellenar con algún plumón permanente.

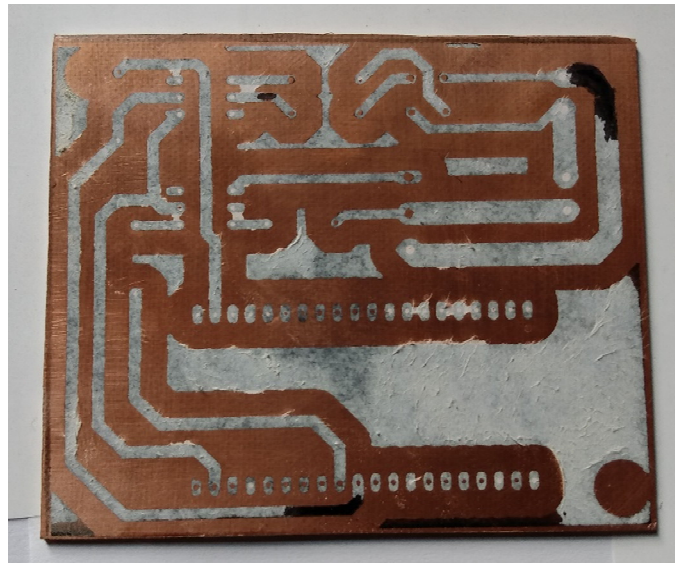


Ilustración 3.32 Placa PCB después del planchado.

Una vez teniendo la placa planchada se coloca el cloruro férrico en un recipiente de plástico suficientemente grande para la que quepa la placa (ilustración 3.33), se llena a que la cubra totalmente, después se menea el recipiente hasta que se remueva todo el cobre.

Nota: el cloruro férrico es un químico estable y en general no presenta grandes riesgos para la salud, pero puede generar gases corrosivos, por lo que se recomienda el uso de gafas de seguridad y mascarilla, así como su uso en lugares con buena circulación de aire. También el uso de bata y guates de látex, ya que en algunos casos puede causar desde irritaciones hasta quemaduras severas. Para mayor información consultar la hoja de riesgos. [34]

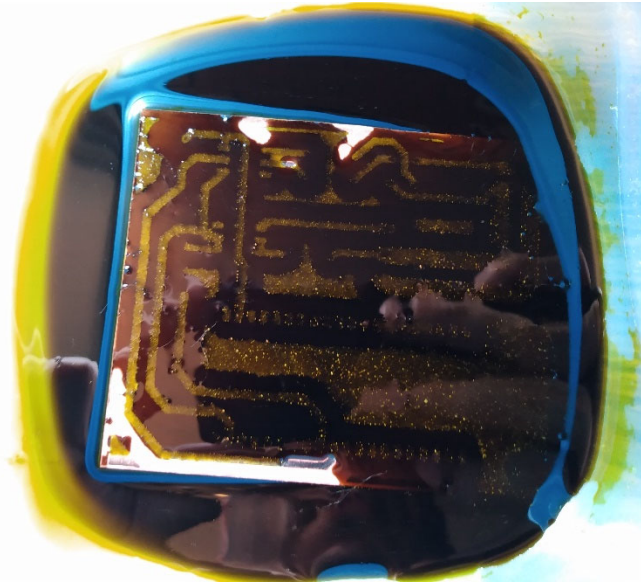


Ilustración 3.33 Placa PCB en cloruro férrico.

Una vez que se le elimina el cobre sobrante a la placa se limpia la tinta y se hacen los barrenos (ilustración 3.34).

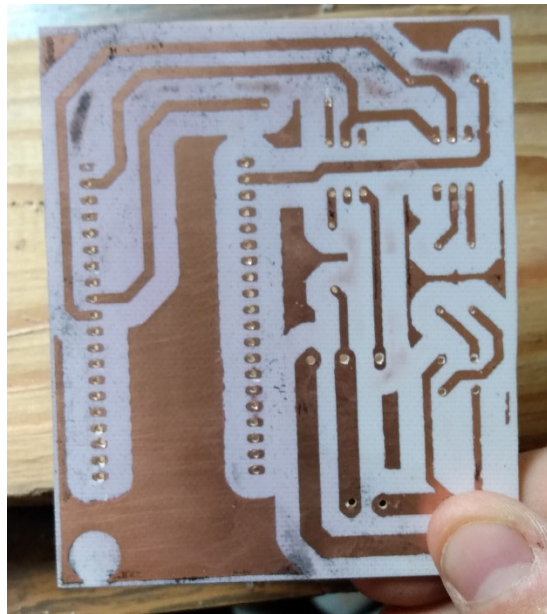


Ilustración 3.34 Placa PCB después de limpiar el cobre sobrante y barrenarla

De manera opcional, se pueden imprimir las capas de etiquetas de Eagle, de la misma manera que en el primer paso, y se planchan (ilustración 3.35) para tener una guía a la hora de colocar y soldar los componentes (ilustración 3.36).

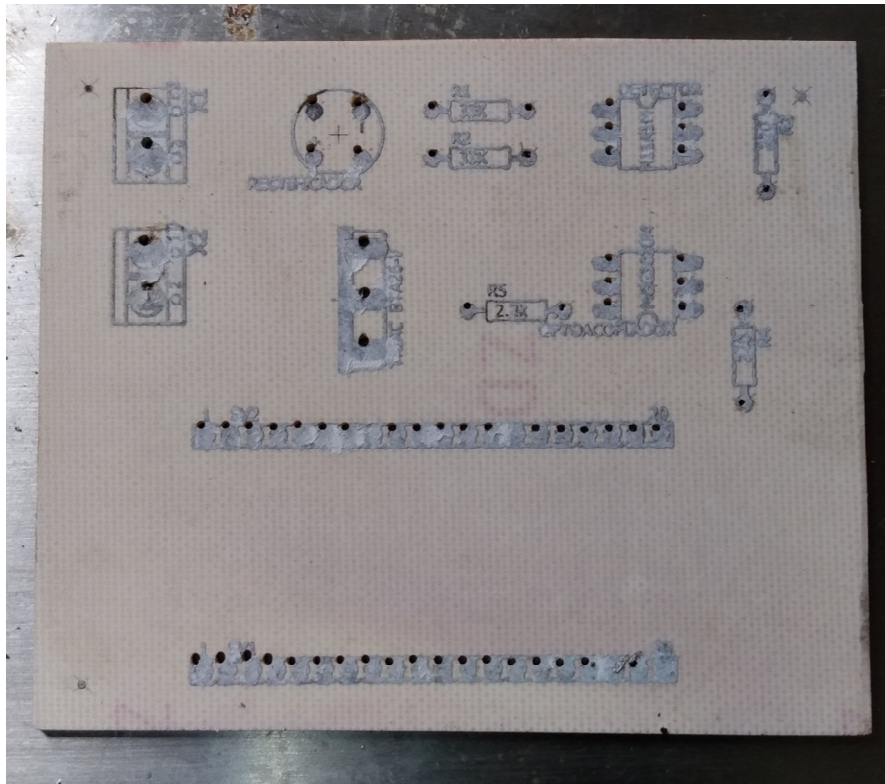


Ilustración 3.35 Placa PCB con las etiquetas impresas.

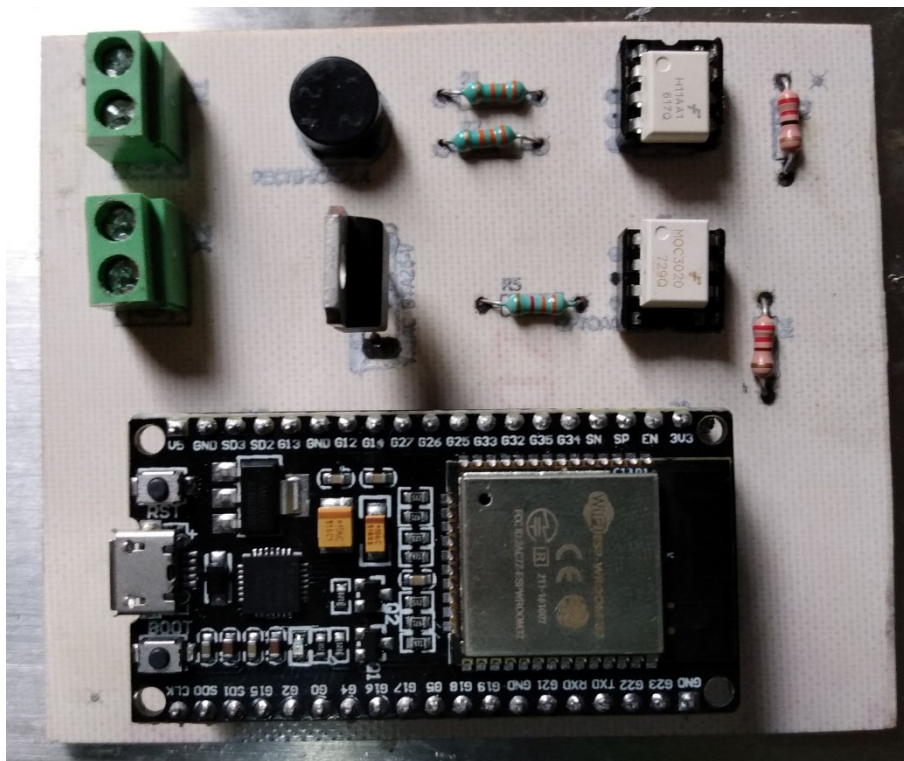


Ilustración 3.36 Placa PCB con los componentes montados.

4.- Pruebas y resultados

Para lograr un resultado satisfactorio de hicieron pruebas individuales previas a cada etapa, después se montó el circuito completo y se realizaron pruebas de nuevo, como se describe a continuación.

Nota importante: para las pruebas se utilizó un osciloscopio, pero ya que en este proyecto se trabajó corriente alterna es necesario utilizar puntas atenuadoras para evitar el daño del mismo y tener una correcta polarización de la alimentación eléctrica para evitar cortos circuitos, ya que este instrumento de medición cuenta con aterrizaje a tierra.

4.1.- Pruebas al circuito de detección de cruce por cero

Esta fue la etapa más sencilla de diseñar del sistema ya que es un circuito muy simple como se puede ver en la ilustración 2.9.

Inicialmente se planteó realizarlo sin el puente rectificador, pero surgía el problema de que solo detectaba un pulso por ciclo ya que no detectaba los semiciclos negativos debido a la composición interna del circuito integrado H11AA1; porque, aunque en el diagrama de la ficha técnica muestra que tiene dos LED en anti paralelo (como se puede ver en la ilustración 2.13), según las pruebas, esto no es cierto y con un solo led permite el paso de un semiciclo y bloquea el otro.

Al agregarle el puente rectificador el circuito integrado detectó correctamente ambos semiciclos y envía dos pulsos por ciclo al microcontrolador con lo que este pudo hacer la sincronización correctamente con la salida.

4.2.- Pruebas al circuito de potencia

En este circuito, originalmente, se consideró utilizar un potenciómetro digital para hacer variar al ángulo de disparo del SCR, este estaría conectado como se muestra en la ilustración 4.1.

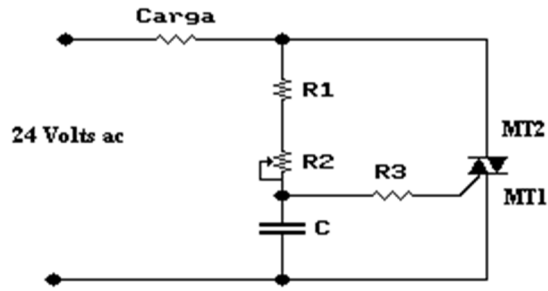


Ilustración 4.1 Diagrama de control de disparo por el 'tao' de carga.

Con esta configuración se controlaría el 'tao' de carga, esto debido a la carga y descarga del capacitor regulando la corriente mediante las resistencias, aquí es donde actuaría el potenciómetro digital, pero para aplicaciones de corriente alterna no es eficiente. Se realizaron tres pruebas de este sistema obteniendo los resultados mostrados en la ilustración 4.2.

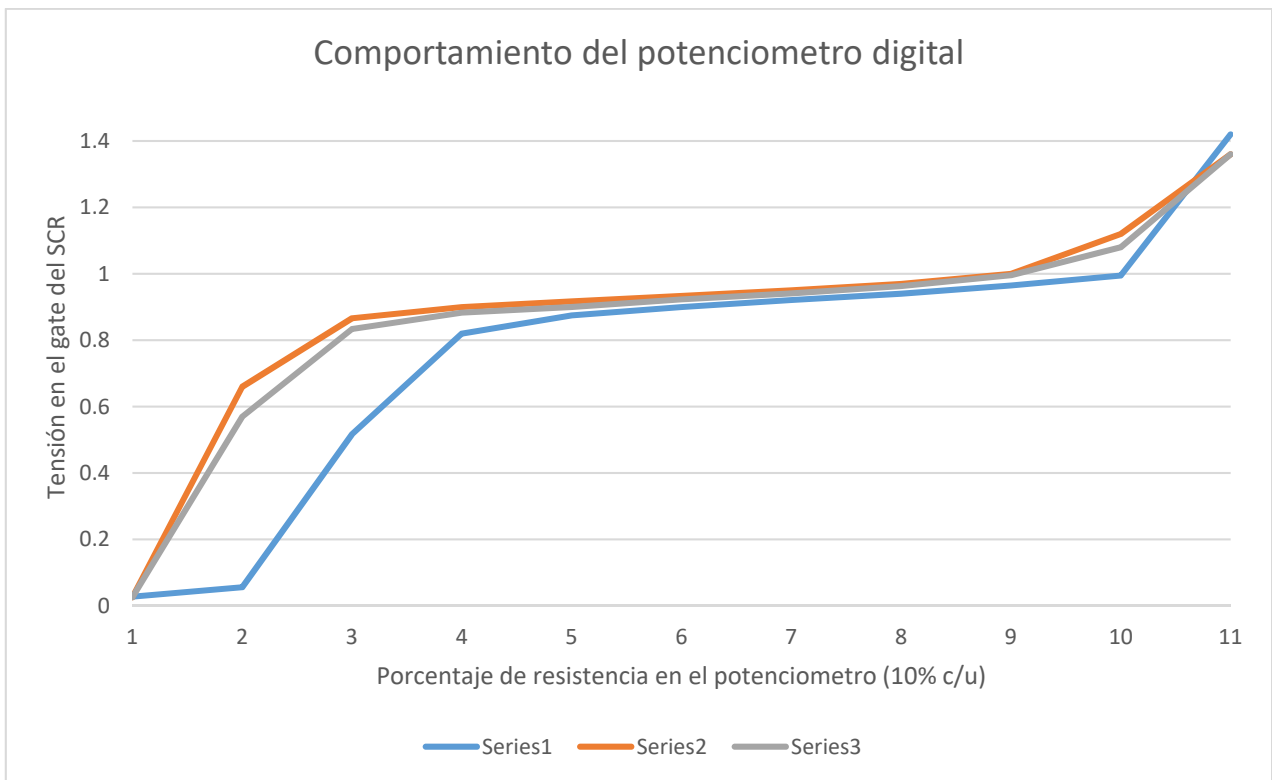


Ilustración 4.2 Grafica de comportamiento del potenciómetro digital.

Como se puede ver en la gráfica, el comportamiento del potenciómetro no es lineal, por lo que el cálculo del ángulo de disparo sería complicado, además de que, al ser electrónico, su tensión máxima de operación es de $5V_{CD}$ y con $127V_{CA}$ el dispositivo se dañaría y se dejó de lado esta opción para pasar a la versión utilizada en la versión final.

La configuración utilizada se puede ver en la ilustración 2.14. De esta forma controlamos el encendido y apagado del triac mediante un optoacoplador y enviando una señal desde el microcontrolador, esto fue posible a través la sincronización de la señal de entrada y salida mediante la detección de cruce por cero y en las pruebas realizadas con el osciloscopio.

En la ilustración 4.3 se puede ver en color amarillo la señal del microcontrolador y como se presenta un pequeño corte instantáneo cada que la señal alterna en azul cruza por cero volts.

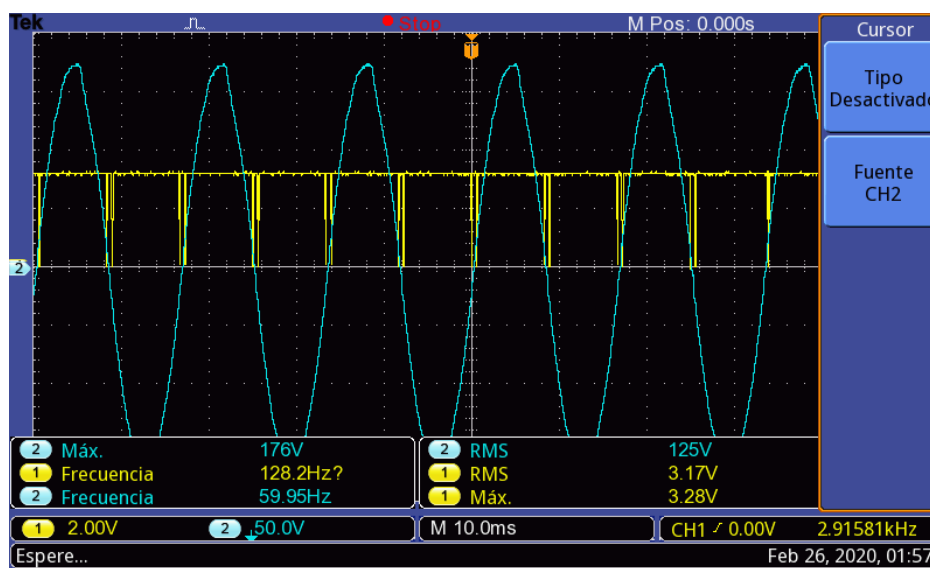


Ilustración 4.3 Oscilograma de señal en onda completa.

En la ilustración 4.4 podemos apreciar un recorte de onda aproximadamente del 75% de la señal en ambos sentidos.

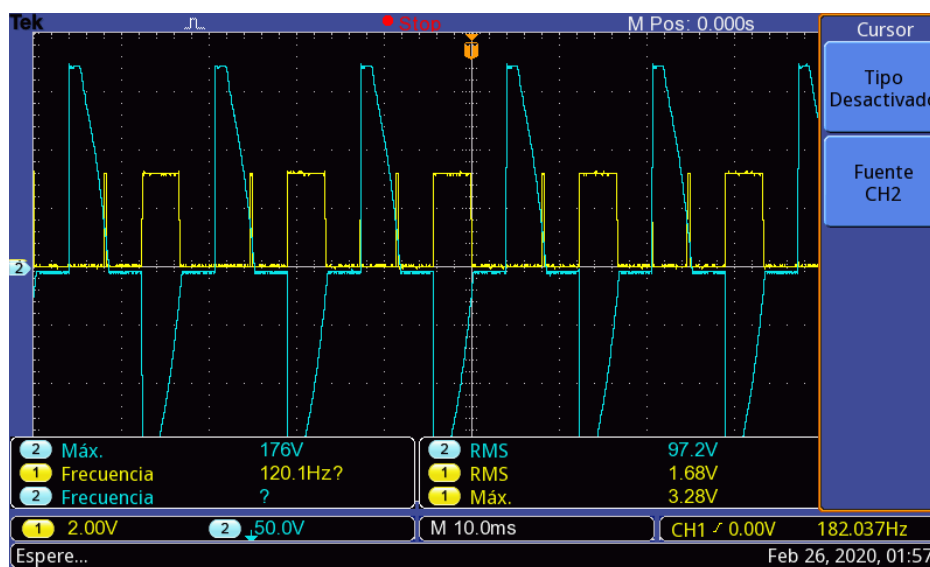


Ilustración 4.4 Oscilograma de señal al 75%

Estas pruebas se realizaron con un foco como carga para poder observar la atenuación (ilustración 4.5), una vez realizadas estas pruebas se procedió a hacer pruebas con un caudín para comprobar el comportamiento con una carga resistiva, para al final hacer pruebas con la cafetera, pero debido a la potencia consumida por esta, los jumpers no soportarían la carga, así que se hizo la PCB para continuar con las pruebas.

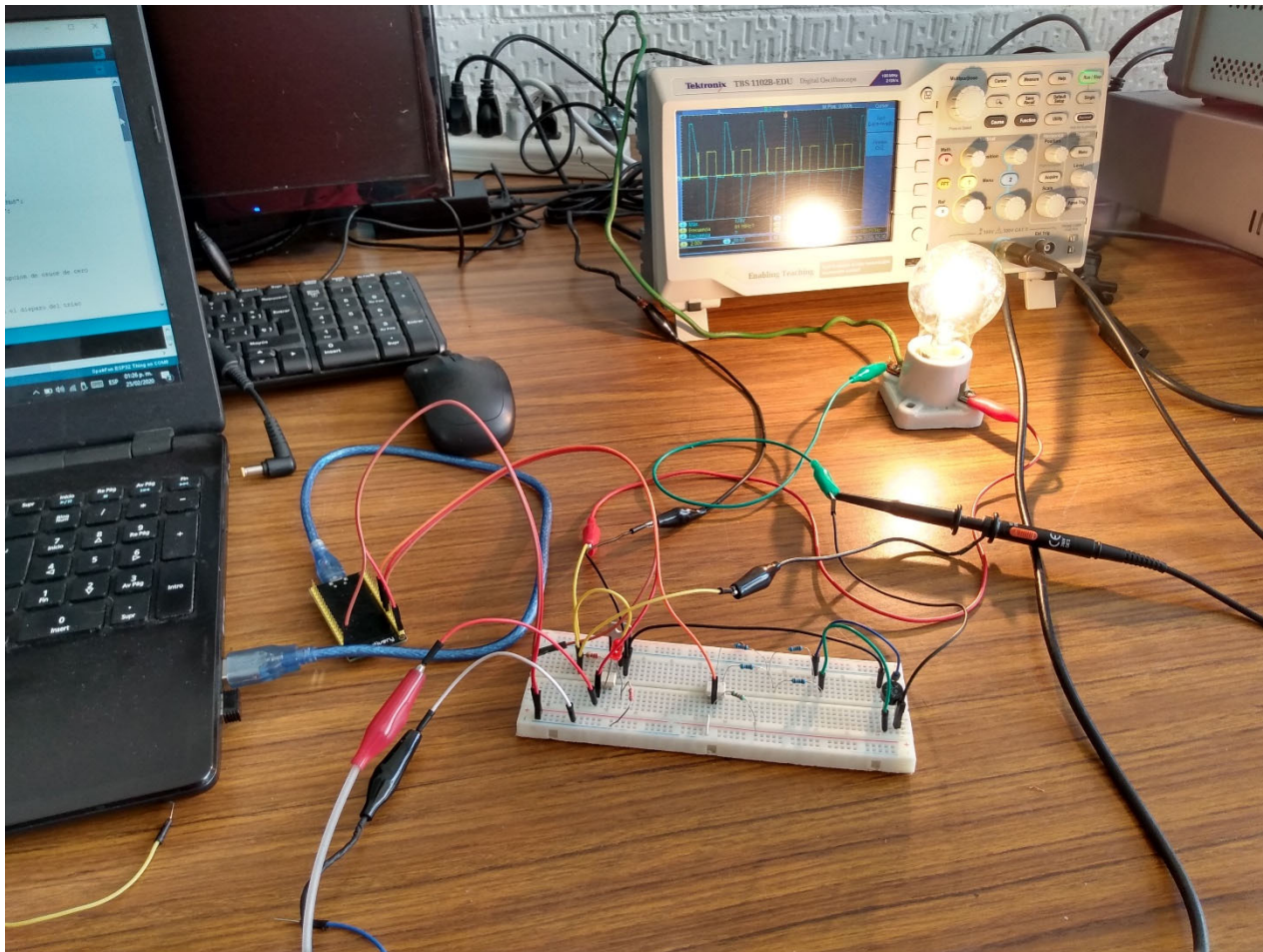


Ilustración 4.5 Pruebas con un foco como carga.

4.3.- Pruebas del código de Arduino

Durante las pruebas de la etapa de potencia se desarrolló el código del programa utilizando un switch case para el control de ángulo de disparo con tiempos fijos del delay en el que se mantendría apagado el SCR desde que se detecte el cruce por cero (ilustración 4.6).

```

switch (value)
{
case '0':
DT = 5000;
break;

case '1':
DT = 4000;
break;

case '2':
DT = 3000;
break;

case '3':
DT = 2000;
break;

case '4':
DT = 0;
break;
}

```

Ilustración 4.6 Código de prueba

Desde el monitor serial del IDE de Arduino se obtenía el valor de la variable ‘value’ y le asignaba un valor a la variable ‘DT’, este valor se utilizaba como delay de apagado después del cruce por cero. Esta estructura funcionó bien durante las pruebas, pero al agregar todos los elementos y enviar el dato desde la aplicación ya no se obtenía el resultado deseado por lo que se cambió a un comando ‘map’ como se puede ver en la ilustración 4.7.

```

if (tmrMillis - tmr1Millis > 300000)
{
DT = map (value, 0, 100, 5000, 1500);
}

```

Ilustración 4.7 Código modificado.

De esta forma, el microcontrolador toma el valor de la aplicación y le asigna el valor proporcional del rango de delay a la variable ‘DT’ después de 5 minutos (300,000 ms).

Por otra parte, para este sistema es necesario llevar a cabo diferentes acciones de manera “simultánea” por lo que no era posible utilizar el comando ‘delay’, ya que con este se detiene completamente el proceso.

Esto aparte de evitar que se lleven a cabo las demás acciones también imposibilita el poder apagar el sistema en el momento que se desee, por lo que se eligió utilizar el comando ‘millis’ para estas funciones, esto se puede apreciar en la ilustración 4.8, donde se comparan

dos variables determinadas por la instrucción 'millis' y se comparan, de ser mayor a lo asignado se lleva a cabo una acción. En el caso del primero, después de 5 minutos disminuye la potencia, en el segundo cuando se llega al tiempo total programado asigna un cero a la variable 'btn' lo que apaga el sistema y el tercero es un monitoreo de las variables de Cayenne, esto para comprobar algún cambio, se hace cada diez segundos ya que no se puede hacer continuamente porque, para establecer la comunicación apaga el sistema entre uno y dos segundos, por lo que durante las pruebas se consideró que este tiempo era adecuado para no afectar el funcionamiento en gran medida y que el usuario no pierda el control del sistema por demasiado tiempo.

```
while(btn==1)
{
    digitalWrite(MOC, HIGH);

    unsigned long currentMillis=millis();
    unsigned long tinicioMillis=millis();
    unsigned long tmrMillis=millis();

    if (tmrMillis - tmr1Millis > 300000)
    {
        DT = map (value, 0, 100, 5000, 1500);
    }

    if (tinicioMillis - tfinMillis > tfin)
    {
        btn=0;
    }

    if (currentMillis - previusMillis > interval)
    {
        Cayenne.loop();
        previusMillis = currentMillis;
    }
}
```

Ilustración 4.8 Uso de la instrucción 'millis'

El último detalle encontrado durante las pruebas fue que la plataforma Cayenne inicializa sus valores en cero, por lo que se consideró darle un valor inicial a las mismas por si el usuario olvida darle un valor al desconectar y conectar el sistema de nuevo. En el caso de la variable 'btn' no es necesario ya que nos conviene que se inicialice en cero. Los valores que se consideraron apropiados fueron los mostrados en la ilustración 4.9.

```

//Variables globales
int btn; //Señal ON/OFF
int value = 50; //Valor de potencia de trabajo Cayenne
int tfin = 15; //valor de tiempo total Cayenne

```

Ilustración 4.9 Valores iniciales de las variables de Cayenne.

4.4.- Pruebas de la aplicación

Originalmente de planteo diseñar la aplicación con App Inventor (ilustraciones 4.10 y 4.11) y hacer la comunicación mediante un servidos en línea gratuito como firebase (ilustración 4.12). La aplicación se quedó en fase de pruebas ya que presentaba muchas complicaciones.

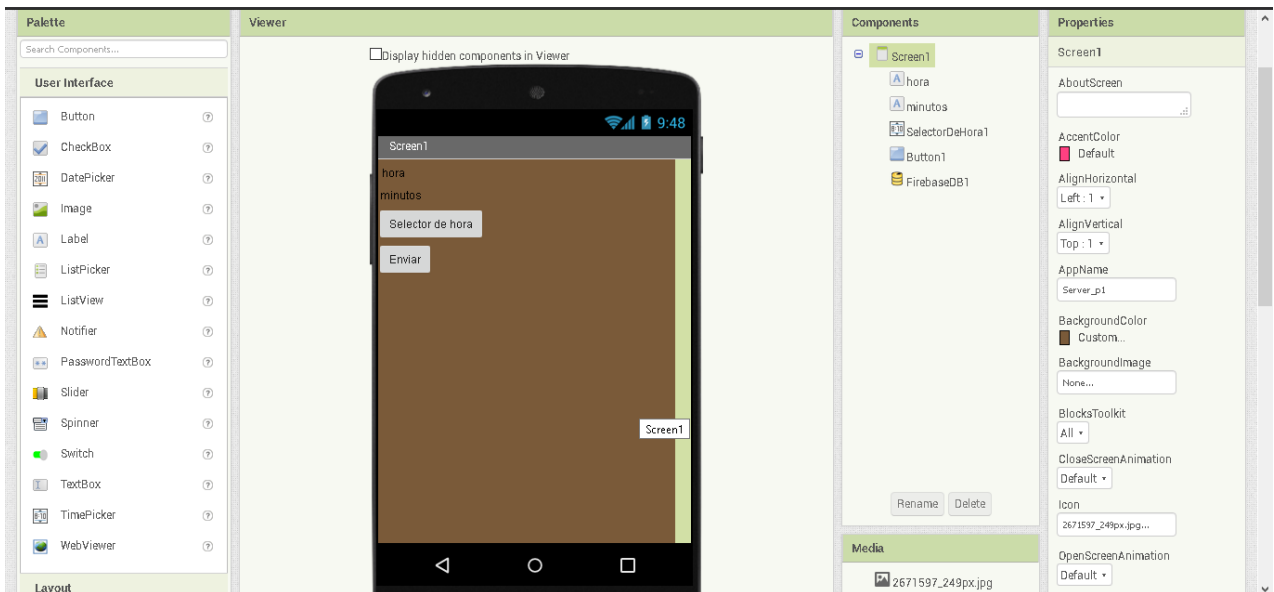


Ilustración 4.10 Vista de la aplicación de prueba en App Inventor

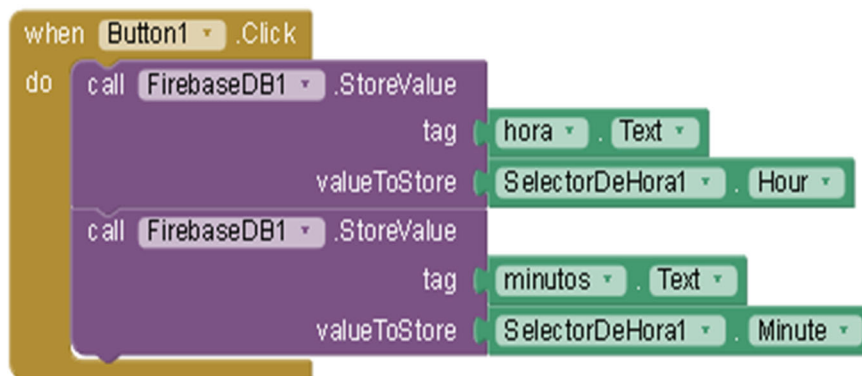


Ilustración 4.11 Diagrama de bloques de la aplicación de prueba en App Inventor



Ilustración 4.12 Vista general de la base de datos del servidor Firebase

Una de las complicaciones fue que se tenía que usar un reloj de tiempo real externo ya que ni la aplicación ni el servidor mantienen un horario fijo en el microcontrolador, este problema fue fácil de solucionar, aunque no demasiado, esto hacía el código más complejo.

El principal problema que se encontró en este sistema era la comunicación aplicación – servidor – microcontrolador ya que el servidor recibía los datos como variables de tipo ‘char’ y cuando las enviaba el microcontrolador este no las podía interpretar y la instrucción para convertir las variables no funcionaba.

Se consideró utilizar comandos como ‘switch case’ o ‘if’ para recibir la variable y convertirla, pero en ‘case’ solo puede tener un máximo de diez casos (de cero a nueve), así que el ‘if’ se consideraba como la opción más viable el problema es que se necesitaban veinticuatro casos para la hora y sesenta para los minutos, esto además de la regulación de potencia y el tiempo de encendido, por lo que el código se extendería demasiado.

Cuando se encontró la plataforma Cayenne se vio que el servidor proporcionaba la hora, por lo que no eran necesarias todas las complicaciones anteriores, así que se mudó la idea a esta plataforma, además de la facilidad que proporcionaba para crear la aplicación y que contaba con su propia app para Android o iOS.

De esta manera se pudo desarrollar la aplicación de una manera muy sencilla y se simplificó bastante el código, como se puede ver en el anexo ‘a’, ya que Cayenne también cuenta con una librería para el IDE de Arduino

4.5.- Pruebas finales

Como se mencionó anteriormente, las pruebas se desarrollaron en una protoboard con jumpers, lo que es común para pruebas de laboratorio, pero una vez el sistema funcionó correctamente y se le intentó conectar a carga de la cafetera la potencia fue muy alta para el calibre del cable, por lo que se quemó uno y se procedió a fabricar la PCB de acuerdo al procedimiento del apartado 3.5.

Con la PCB ya completada se hicieron las pruebas finales, una de ellas fue el monitoreo de corriente (ilustración 4.13). El cálculo fue diferente al real, ya que consume un amperre menos que el calculado a onda completa, eso ayuda a aumentar el tiempo de vida de los componentes de la etapa de potencia.



Ilustración 4.13 Medición de corriente real.

Después de eso se hicieron pruebas de operación (ilustración 4.14), lo cual funcionó de manera correcta y se tomaron los oscilogramas (ilustraciones 4.15 a 4.19), donde se observa el recorte de la onda y por ende la reducción de la potencia a la que trabaja la cafetera.

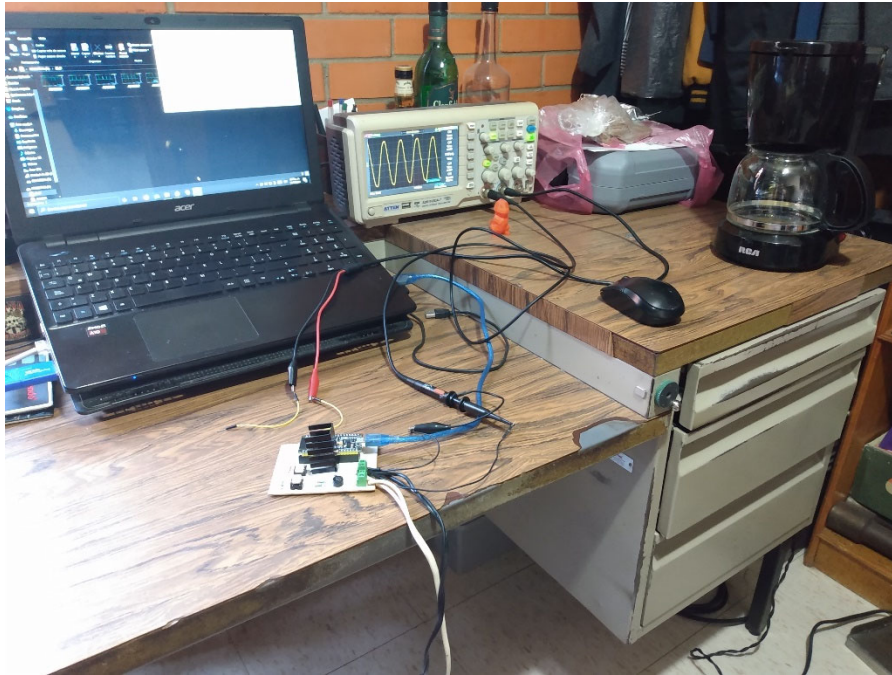


Ilustración 4.14 Pruebas de operación.

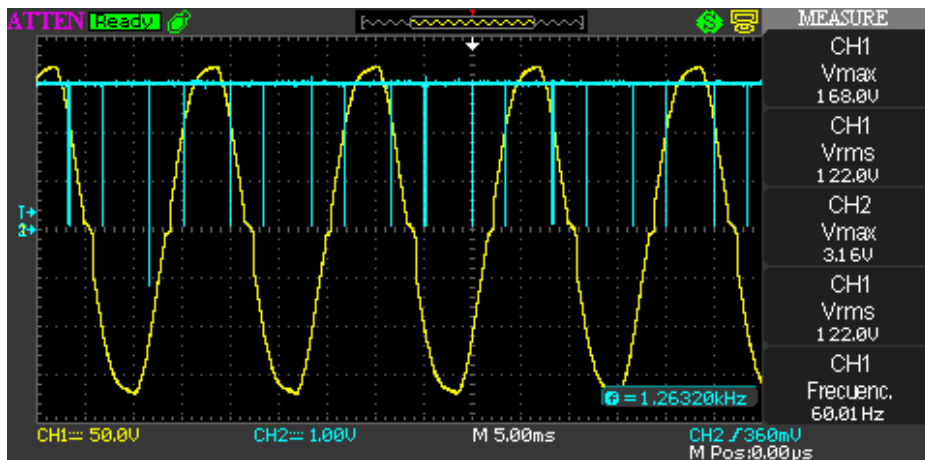


Ilustración 4.15 Oscilograma a 100% de la onda.

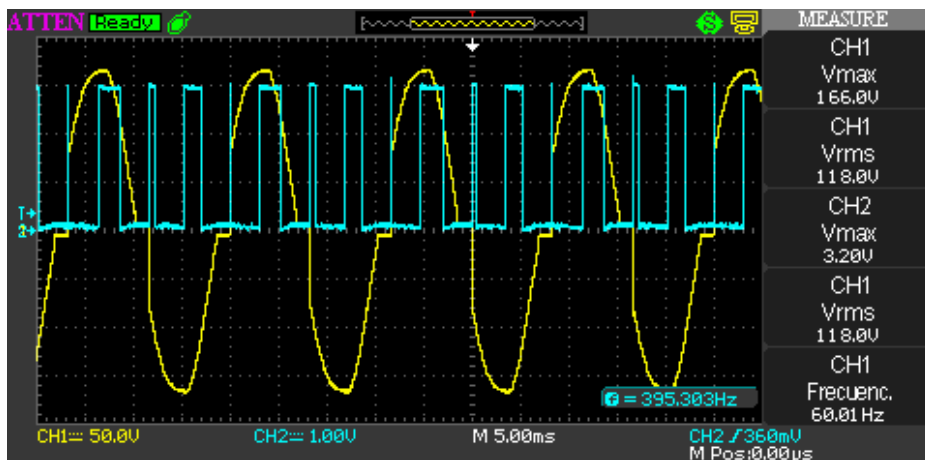


Ilustración 4.16 Oscilograma a 95% de la onda.

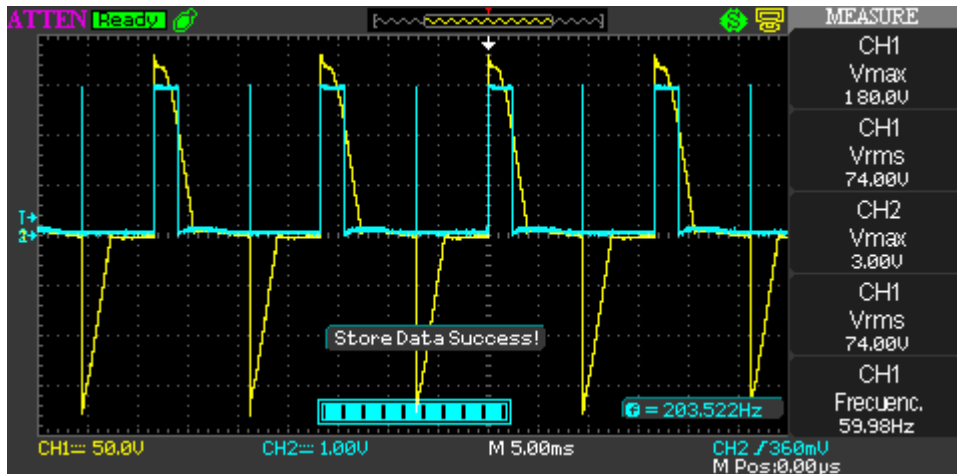


Ilustración 4.17 Oscilograma a 60% de la onda.

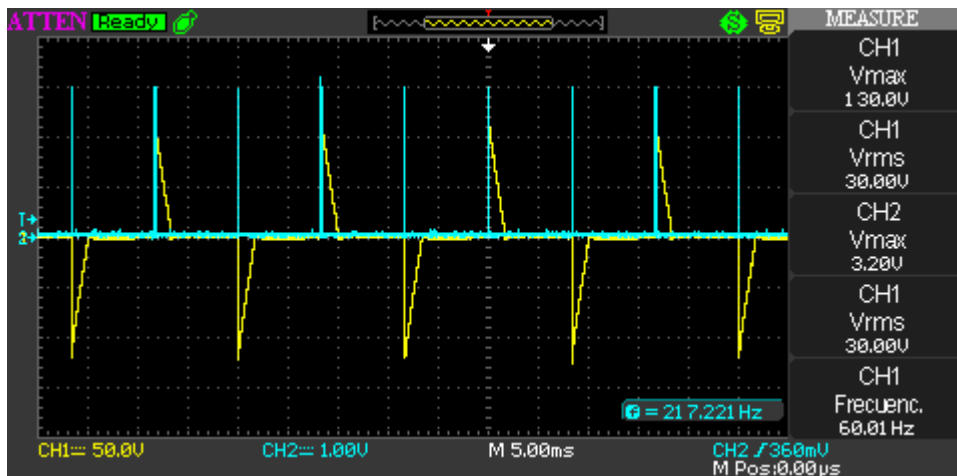


Ilustración 4.18 Oscilograma al 25% de la onda.

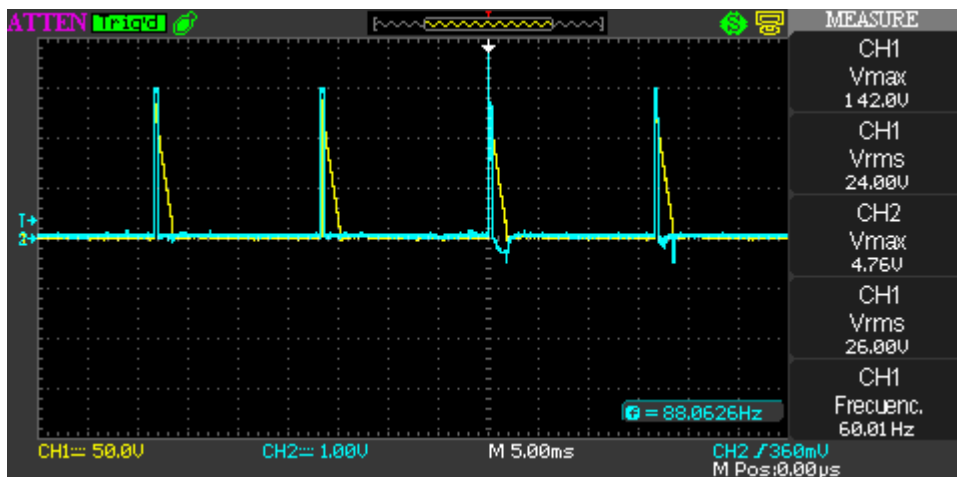


Ilustración 4.19 Oscilograma al 20% de la onda.

Conclusiones y trabajo futuro

A través de la plataforma Cayenne se desarrolló una aplicación para dispositivo móvil, laptop o computadora de escritorio, que permite al usuario el control y la programación del sistema de manera remota.

Para realizar la comunicación entre la aplicación y el dispositivo se aprovechó el módulo WiFi con el que cuenta el microcontrolador ESP-32 y se conecta de manera constante entre ambos para actualizar los datos de tiempo de encendido, potencia de trabajo y encendido / apagado.

A través de la aplicación de escritorio se pueden configurar las horas y fechas en las que el usuario quiera encender de la cafetera en diferentes eventos, así como programarlo de manera semanal, mensual o anual.

Además de la programación de horarios, la aplicación permite programar un tiempo de encendido total, así como la potencia a la que la cafetera va a trabajar después de que termine la preparación del café para reducir el consumo de energía eléctrica y que la cafetera se apague de manera automática una vez que termine el proceso.

Debido a que el proyecto cuenta con una etapa de control electrónico a $5V_{CD}$ y la carga trabaja a $127V_{CA}$ fue necesario implementar una etapa de acoplamiento de potencia, para esto se utilizaron un optoacoplador, para separar las señales y un triac para poder controlar el encendido y apagado de la carga, así como el ángulo de disparo en la regulación de potencia.

Para el control de este sistema de utilizo un microcontrolador ESP-32, este está encargado de procesar las instrucciones del código de acuerdo a las variables que obtiene de la aplicación. El código fue estructurado de acuerdo al diagrama de flujo mostrado en el apartado 3.4, en la ilustración 3.31 para poder realizarlo de manera ordenada.

Con lo mencionado anteriormente y lo que se estudió en el apartado 2.2, podemos decir que este sistema trabaja bajo la arquitectura básica del internet de las cosas y cumple con el paradigma del mismo, ya que cuenta con intercomunicación entre la aplicación móvil y el dispositivo diseñado, el dispositivo es sensible a cambios dinámicos como el encendido y apagado, el cambio en tiempo de encendido total y el cambio de potencia de trabajo, para

su conectividad es necesaria una identificación mediante el inicio de sesión en la aplicación móvil, entre otros aspectos que hacen que este proyecto se considere un dispositivo IoT.

Considerando todo lo anterior se puede decir que el sistema cumple con los objetivos establecidos, ya que el dispositivo final es un dispositivo que se conecta a una cafetera de calentamiento por resistencia y, con una aplicación móvil y, utilizando el internet de las cosas, se puede configurar el funcionamiento de esta.

El prototipo permite el control de la cafetera de manera remota sin la necesidad de que este y el dispositivo móvil se encuentren conectados a la misma red WiFi, por lo que, en teoría, se puede realizar el control de la cafetera desde cualquier parte del mundo, siempre y cuando se cuente con los datos de inicio de sesión de la aplicación de Cayenne.

Este proyecto puede tomarse como punto de partida para otros proyectos similares o mejoras para el mismo. Algunas de las mejoras que se pueden implementar es una aplicación auxiliar aprovechando el modulo Bluetooth con el que cuenta el microcontrolador en caso de que se pierda la conexión WiFi de alguno de los dos dispositivos, se considera una auxiliar debido a las limitaciones que presenta este tipo de conectividad como la distancia de alcance o las interferencias que se pueden presentar debido a muros u objetos que puedan estar en medio de ambos dispositivos. Además de esto se puede generar un pequeño sitio web a través de la IP del microcontrolador para que un usuario final pueda introducir la información de su red WiFi y no sea necesario cambiarlo directamente desde el código de Arduino. Por otra parte, se puede desarrollar una aplicación utilizando servidores diferentes a los planteados aquí para poder dar más funcionalidad desde la aplicación móvil, especialmente en la programación de horarios, ya que por ahora solo se puede por la aplicación de escritorio y en el proceso actual depende completamente de la plataforma Cayenne el alcance de las aplicaciones móviles.

Algunos de los proyectos que se pueden realizar a partir de un sistema similar al planteado en este trabajo pueden ser el de controlar hornos o parrillas eléctricas, así como algún otro aparato que utilice resistencias o motores universales como ventiladores o licuadoras, de los que se pueden controlar la velocidad.

El presente proyecto se pudo realizar gracias a los conocimientos adquiridos durante la carrera y se pueden aplicar en el mundo de la domótica, la cual es un área de la electrónica

que consta de diferentes sub áreas y en este proyecto nos enfocamos en la parte de confort de dicha disciplina tecnológica.

Referencias de imagines

Ilustración 2.1 Cafetera italiana o Moka. Recuperada el 04 de abril del 2019 de: <https://capuchinox.com/cuales-son-los-principales-tipos-de-cafeteras/>

Ilustración 2.2 Cafetera de goteo. Recuperada el 04 de abril del 2019 de: <https://capuchinox.com/cuales-son-los-principales-tipos-de-cafeteras/>

Ilustración 2.3 Cafetera superautomática. Recuperada el 04 de abril del 2019 de: <https://capuchinox.com/cuales-son-los-principales-tipos-de-cafeteras/>

Ilustración 2.4 Imagen ilustrativa del internet de las cosas. Recuperada el 05 de abril de 2019 de: <https://amarilo.com.co/blog/actualidad/el-auge-del-internet-de-las-cosas/>

Ilustración 2.5 Arquitectura IoT de tres capas. Recuperada el 27 octubre de 2019 de: A.Cobos, Diseño e implementacion de una arquitectura IoT basada en tecnologia open source, Seville, Universidad de Sevilla, 2016.

Ilustración 2.6 Arquitectura IoT de cinco capas. Recuperada el 27 octubre de 2019 de: A.Cobos, Diseño e implementacion de una arquitectura IoT basada en tecnologia open source, Seville, Universidad de Sevilla, 2016.

Ilustración 2.7 Modulacion PWM. Recuperada el 26 de febrero de 2020 de: <http://mecanicappweb.com/modulacion-de-ancho-de-pulso-pwm/>

Ilustración 2.8 Deteccion de cruce por cero. Recuperada el 26 de febrero de 2020 de: <https://www.luisllamas.es/arduino-cruce-por-cero-h11aa1/>

Ilustración 2.9 Circuito del detector de cruce por cero. Imagen propia.

Ilustración 2.10 Rectificador de media onda. Recuperado el 04 de marzo del 2020 de: <http://www.cifpn1.com/electronica/?p=2994>

Ilustración 2.11 Rectificador de onda completa con tap central. Recuperado el 04 de marzo del 2020 de: <http://www.cifpn1.com/electronica/?p=2994>

Ilustración 2.12 Rectificador de onda completa. Recuperado el 04 de marzo del 2020 de: <http://www.cifpn1.com/electronica/?p=2994>

Ilustración 2.13 Esquema del H11AA1. Recuperado el 26 de febrero de 2020 de: <https://pdf1.alldatasheet.com/datasheet-pdf/view/3037/MOTOROLA/H11AA1.html>

Ilustración 2.14 Circuito de acoplamiento. Imagen propia.

Ilustración 2.15 Esquema de funcionamiento del triac. Recuperado el 04 de marzo de 2020 de: http://profesormolina2.webcindario.com/tutoriales/enica_pot.htm

Ilustración 2.16 Esquema del MOC3010. Recuperado el 09 de diciembre de 2019 de: <https://articulo.mercadolibre.com.ve/MLV-462106198-moc3010-nte3047-optocoupler->

optoacoplador--_JM#position=2&type=item&tracking_id=490c45bc-adc3-4a33-afe6-3699671dc891

Ilustración 2.17 Composición de un SCR. Recuperado el 09 de diciembre de 2019 de: <https://www.ingmecafenix.com/electronica/triac/>

Ilustración 2.18 Triac BTA12 y sus tipos de encapsulado. Recuperado el 18 de diciembre de 2019 de: <https://www.st.com/resource/en/datasheet/bta08.pdf>

Ilustración 2.19 Algunos tipos de disipadores de calor. Recuperado el 13 de abril de 2020 de: <https://trends.directindustry.es/project-165549.html>

Ilustración 2.20 Sistema básico de control. Recuperado el 18 de noviembre de 2019 de: R Hernández, Introducción a los sistemas de control, Aguascalientes, Pearson Education, 2010.

Ilustración 2.21 Tipos de entradas aplicadas a los sistemas de control. Recuperado el 18 de noviembre de 2019 de: R Hernández, Introducción a los sistemas de control, Aguascalientes, Pearson Education, 2010.

Ilustración 2.22 Sistema básico de control de lazo abierto. Recuperado el 18 de noviembre de 2019 de: R Hernández, Introducción a los sistemas de control, Aguascalientes, Pearson Education, 2010.

Ilustración 2.23 Sistema básico de control de lazo cerrado. Recuperado el 18 de noviembre de 2019 de: R Hernández, Introducción a los sistemas de control, Aguascalientes, Pearson Education, 2010.

Ilustración 2.24 Arquitectura básica de un microcontrolador. Recuperado el 25 de noviembre de 2019 de: <https://www.electronicaestudio.com/que-es-un-microcontrolador/>

Ilustración 2.25 Tipos de encapsulados de un circuito integrado. Recuperado el 18 de noviembre de 2019 de: <http://avecomputointe.blogspot.com/2012/02/microcontroladores.html> 25/11/19

Ilustración 2.26 Imagen ilustrativa de tarjetas de desarrollo. Recuperada el 02 de diciembre de 2019 de: <https://hacedores.com/que-tarjeta-de-desarrollo-elegir-parte-2/>

Ilustración 2.27 ESP-WROOM-32. Recuperada el 27 de febrero de 2020 de: <https://circuitdigest.com/microcontroller-projects/getting-started-with-esp32-with-arduino-ide>

Ilustración 2.28 IDE Arduino. Imagen propia.

Ilustración 2.29 Pantalla ejemplo de la aplicación Cayenne. Recuperado el 28 de febrero de 2020 de: <https://cayenne.mydevices.com/cayenne/signup>

Ilustración 3.1 Diagrama de bloques del sistema. Imagen propia.

Ilustración 3.2 Diagrama esquemático. Imagen propia.

Ilustración 3.3 Capa inferior de la PCB del sistema. Imagen propia.

Ilustración 3.4 Aplicación para cálculo de ancho de pistas. Imagen propia.

Ilustración 3.5 Muestra del cruce por cero de señal rectificada. Recuperada el 03 de marzo de 2020 de: <https://sites.google.com/site/electronicaanalogicam/2--amplificadores-oper/2-3--comparadores/2-3-1--detectores-de-cruce-por-cero>

Ilustración 3.6 Aplicación de escritorio en la plataforma Cayenne. Imagen propia.

Ilustración 3.7 Pantalla principal de Cayenne. Imagen propia.

Ilustración 3.8 Pantalla de registro de Cayenne. Imagen propia.

Ilustración 3.9 Pantalla de selección de dispositivo Cayenne. Imagen propia.

Ilustración 3.10 Pantalla con todos los dispositivos Cayenne. Imagen propia.

Ilustración 3.11 Pantalla de información Cayenne. Imagen propia.

Ilustración 3.12 Pantalla de inicio del proyecto Cayenne. Imagen propia.

Ilustración 3.13 Inserta un nuevo widget en Cayenne. Imagen propia.

Ilustración 3.14 Menú de Widgets. Imagen Propia.

Ilustración 3.15 Configuraciones del Widget. Imagen propia.

Ilustración 3.16 Configuración de los sliders. Imagen propia.

Ilustración 3.17 Tiendas de aplicación iOS y Android. Imagen propia.

Ilustración 3.18 Pantalla de inicio de sesión Cayenne iOS, Android y escritorio. Imagen propia

Ilustración 3.19 Pantalla principal de las aplicaciones móviles. Imagen propia

Ilustración 3.20 Pantalla principal de la aplicación de escritorio. Imagen propia.

Ilustración 3.21 Pantalla de aplicación de Cayenne iOS, Android y escritorio. Imagen propia.

Ilustración 3.22 Menú de usuario. Imagen propia.

Ilustración 3.23 Vista de calendario Cayenne. Imagen propia.

Ilustración 3.24 Ventana de configuración de evento. Imagen propia.

Ilustración 3.25 Configuración de acciones. Imagen propia.

Ilustración 3.26 Evento diario programado. Imagen propia.

Ilustración 3.27 Preferencias del IDE de Arduino. Imagen propia.

Ilustración 3.28 Gestor de tarjetas del IDE de Arduino. Imagen propia.

Ilustración 3.29 Pagina de descarga de la librería. Imagen propia.

Ilustración 3.30 Menú del IDE de Arduino para añadir librería. Imagen propia.

Ilustración 3.31 Diagrama de flujo del código. Imagen propia.

Ilustración 3.32 Placa PCB después del planchado. Imagen propia.

Ilustración 3.33 Placa PCB en cloruro férrico. Imagen propia.

Ilustración 3.34 Placa PCB después de limpiar el cobre sobrante y barrenarla. Imagen propia.

Ilustración 3.35 Placa PCB con las etiquetas impresas. Imagen propia.

Ilustración 3.36 Placa PCB con los componentes montados. Imagen propia.

Ilustración 4.1 Diagrama de control de disparo por el ‘tao’ de carga. Recuperado el 13 de marzo de 2020 de: <http://cesarpfc.50webs.com/c2.htm>

Ilustración 4.2 Grafica de comportamiento del potenciómetro digital. Imagen propia.

Ilustración 4.3 Oscilograma de señal en onda completa. Imagen propia.

Ilustración 4.4 Oscilograma de señal al 75%. Imagen propia.

Ilustración 4.5 Pruebas con un foco como carga. Imagen propia.

Ilustración 4.6 Código de prueba. Imagen propia.

Ilustración 4.7 Código modificado. Imagen propia.

Ilustración 4.8 Uso de la instrucción 'millis'. Imagen propia.

Ilustración 4.9 Valores iniciales de las variables de Cayenne. Imagen propia.

Ilustración 4.10 Vista de la aplicación de prueba en App Inventor. Imagen propia.

Ilustración 4.11 Diagrama de bloques de la aplicación de prueba en App Inventor. Imagen propia.

Ilustración 4.12 Vista general de la base de datos del servidor Firebase. Imagen propia.

Ilustración 4.13 Medición de corriente real. Imagen propia.

Ilustración 4.14 Pruebas de operación. Imagen propia.

Ilustración 4.15 Oscilograma a 100% de la onda. Imagen propia.

Ilustración 4.16 Oscilograma a 95% de la onda. Imagen propia.

Ilustración 4.17 Oscilograma a 60% de la onda. Imagen propia.

Ilustración 4.18 Oscilograma a 25% de la onda. Imagen propia.

Ilustración 4.19 Oscilograma a 20% de la onda. Imagen propia.

Referencias bibliográficas

- [1] F. Serpa Flores, «La historia del café,» *Bolietín cultural y bibliográfico*, vol. 7, nº 9, pp. 1604-1607, 1964.
- [2] U. Martínez, «Organización mexicana de cafés y cafeterías de especialidad,» 5 Diciembre 2016. [En línea]. Available: <http://www.amcce.org.mx/letras-de-cafe/post/la-historia-del-cafe-en-mexico>. [Último acceso: 12 Marzo 2019].
- [3] D. Flores Magon, «Local MX,» 30 Mayo 2018. [En línea]. Available: <https://local.mx/ciudad-de-mexico/cronica-ciudad/estos-eran-los-cafes-del-siglo-xix-cultivo-del-ocio-y-la-bohemia-nacional/>. [Último acceso: 12 Marzo 2019].
- [4] N. Pére Quiroz, «Noticias 7 24,» 22 Agosto 2018. [En línea]. Available: <https://siete24.mx/mexico/el-consumo-del-cafe-de-especialidad-en-mexico-ha-crecido-de-28-a-46/>. [Último acceso: 14 Marzo 2019].
- [5] Gelmato, «Gelmato,» 3 octubre 2018. [En línea]. Available: https://www.gemalto.com/brochures-site/download-site/Documents/M2M_CS_Bonaverde-es.pdf. [Último acceso: 27 febrero 2020].
- [6] Zemsania, «Zemsania Global Goup,» 3 noviembre 2019. [En línea]. Available: <https://zemsaniaglobalgroup.com/zemsania-presenta-una-cafetera-inteligente-en-tiempo-real/>. [Último acceso: 27 febrero 2020].
- [7] Andrew_h, «Instructables,» 2018. [En línea]. Available: <https://www.instructables.com/id/IoT-Enabled-Coffee-Machine/>. [Último acceso: 27 febrero 2020].
- [8] Capuchinox, «Capuchinox,» 2 Julio 2019. [En línea]. Available: <https://capuchinox.com/cuales-son-los-principales-tipos-de-cafeteras/>. [Último acceso: 4 Octubre 2019].
- [9] B. Sánchez, *Internet de las cosas. Horizonte 2050*, Santiago de Compostela: Instituto Español de Estudios Estratégicos, 2018.
- [10] A. Cobos Domínguez, *Diseño e implementación de una arquitectura IoT basada en tecnología open source*, Sevilla: Universidad de Sevilla, 2016.
- [11] MecanicAPP, «MecanicAPP,» 23 enero 2017. [En línea]. Available: <http://mecanicappweb.com/modulacion-de-ancho-de-pulso-pwm/>. [Último acceso: 26 febrero 2020].
- [12] M. H. Rashid, *Power electronics circuits, devices and applications*, New Jersey.

- [13] L. Llamas, «Ingeniería Informática y Diseño,» 12 enero 2017. [En línea]. Available: <https://www.luisllamas.es/arduino-cruce-por-cero-h11aa1/>. [Último acceso: 26 febrero 2020].
- [14] Motorola, «All data sheet,» [En línea]. Available: <https://pdf1.alldatasheet.com/datasheet-pdf/view/3037/MOTOROLA/H11AA1.html>. [Último acceso: 26 febrero 2020].
- [15] «Carrod Electrónica,» [En línea]. Available: <https://www.carrod.mx/products/optoacoplador-moc3010-salida-triac>. [Último acceso: 09 diciembre 2019].
- [16] F. Mecafenix, «Ingeniería Mecafenix,» 17 octubre 2018. [En línea]. Available: <https://www.ingmecafenix.com/electronica/triac/>. [Último acceso: 9 diciembre 2019].
- [17] STMicroelectronics, «alldatasheets,» [En línea]. Available: <https://pdf1.alldatasheet.com/datasheet-pdf/view/158140/STMICROELECTRONICS/BTA12.html>. [Último acceso: 27 septiembre 2019].
- [18] A. Méndez, «Electronic an Biomedical Instrumentation Group,» enero 2016. [En línea]. Available: http://ieb-srv1.upc.es/gieb/tecnicas/pdf/disipadores_termicos.pdf. [Último acceso: 07 abril 2020].
- [19] J. Bertuccio, «Facultad de Ingenieria de la Universidad de Buenos Aires,» 05 marzo 2015. [En línea]. Available: <http://materias.fi.uba.ar/6610/Apuntes/Calculo%20de%20disipadores%20de%20calor.pdf>. [Último acceso: 07 abril 2020].
- [20] R. Hernández Gaviño, Introducción a los sistemas de control, Aguascalientes: Pearson Educación, 2010.
- [21] G. Ávila Alterach, «Electrónica, programación y otras cosas,» 2012. [En línea]. Available: <https://gzalo.com/intropic/>. [Último acceso: 18 Noviembre 2019].
- [22] M. Verle, de *PIC microcontrollers Programing in C with examples*, MikroElectronica.
- [23] E. d. Expertos, «Universidad Internacional de Valencia,» 21 Marzo 2018. [En línea]. Available: <https://www.universidadviu.com/lenguaje-nivel-caracteristicas-funciones/>. [Último acceso: 25 Noviembre 2019].
- [24] E. d. Expertos, «Universidad Internacional de Valencia,» 21 Marzo 2018. [En línea]. Available: <https://www.universidadviu.com/lenguaje-alto-nivel-los-mas-utilizados/>. [Último acceso: 25 Noviembre 2019].
- [25] Jessika, «Informática 2 & tercero!», 6 Octubre 2009. [En línea]. Available: <http://jessicajaime.blogspot.com/2009/10/ventajas-y-desventajas-de-programar.html>. [Último acceso: 25 Noviembre 2019].

- [26] K. Knights, «Roboperks,» 27 septiembre 2018. [En línea]. Available: <https://www.roboperks.com/language/es/placas-de-desarrollo/>. [Último acceso: 25 noviembre 2019].
- [27] Hacedores, «Hacedores,» 05 junio 2014. [En línea]. Available: <https://hacedores.com/que-tarjeta-de-desarrollo-elegir-parte-1/>. [Último acceso: 02 diciembre 2019].
- [28] R. Aswinth, «Circuit Digest,» 26 abril 2018. [En línea]. Available: <https://circuitdigest.com/microcontroller-projects/getting-started-with-esp32-with-arduino-ide>. [Último acceso: 27 febrero 2020].
- [29] Last Minute Engineers, «Last Minute Engineers,» [En línea]. Available: <https://lastminuteengineers.com/esp32-arduino-ide-tutorial/>. [Último acceso: 27 febrero 2020].
- [30] Last Minute Engineers, «Last Minute Engineers,» [En línea]. Available: <https://lastminuteengineers.com/esp8266-nodemcu-arduino-tutorial/>. [Último acceso: 27 febrero 2020].
- [31] F. Martínez, «OpenWebinars,» 03 febrero 2015. [En línea]. Available: <https://openwebinars.net/blog/tutorial-arduino-ide-arduino/>. [Último acceso: 02 diciembre 2019].
- [32] Cayenne, «Cayenne Docs,» [En línea]. Available: <https://developers.mydevices.com/cayenne/docs/downloads/>. [Último acceso: 28 febrero 2020].
- [33] Cayenne, «my Devices,» 13 octubre 2016. [En línea]. Available: <https://mydevices.com/article/cayenne-puts-iot-in-the-hands-of-students-and-businesses-at-world-maker-faire/>. [Último acceso: 28 febrero 2020].
- [34] Corporacion Química Omega , «Universidad Autónoma de Baja California,» [En línea]. Available: <http://iio.ens.uabc.mx/hojas-seguridad/cloruro-ferrico.pdf>. [Último acceso: 15 12 2019].
- [35] Prometec, «Prometec,» abril 2019. [En línea]. Available: <https://www.prometec.net/wemos-d1-esp8266-wifi/>. [Último acceso: 02 diciembre 2019].
- [36] L. Llamas, «Luis Llamas,» 18 octubre 2016. [En línea]. Available: <https://www.luisllamas.es/reloj-y-calendario-en-arduino-con-los-rtc-ds1307-y-ds3231/>. [Último acceso: 9 diciembre 2019].
- [37] «Geek Factory,» [En línea]. Available: <https://www.geekfactory.mx/tienda/modulos-para-desarrollo/ds1307-modulo-tiny-rtc-reloj-tiempo-real/>. [Último acceso: 9 diciembre 2019].

Anexo

a) Código de Arduino

```
so#define CAYENNE_DEBUG
#define CAYENNE_PRINT Serial
#include <CayenneMQTTESP32.h>

// Datos de la red WiFi
char ssid[] = "XXXXXXXXXX";

char wifiPassword[] = "XXXXXXXXX";

// Información de autenticación de Cayenne
char username[] = "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXXXXXXX";

char password[] = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";

char clientID[] = "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXXXXXXX";

//Canales de Cayenne
#define ON_CHANNEL 0
#define VALUE_CHANNEL 1
#define TIME_CHANNEL 2

//Puertos utilizados
const int MOC = 19; //Pin G5 activar el MOC
const int INT_PIN = 18; //Pin G18 activar la interrupción de cruce de cero

//Variables globales
int btn; //Señal ON/OFF
```



```

int value = 50; //Valor de potencia de trabajo Cayenne
int tfin = 15; //valor de tiempo total Cayenne
int tiempo = 0; //Valor del tiempo de espera para el disparo del triac
int DT; //Bandera de la interrupción
long interval=10000; //Intervalo en el dispositivo se conecta con Cayenne

void setup()
{
  Serial.begin(9600);
  Cayenne.begin(username, password, clientID, ssid, wifiPassword);

  pinMode(INT_PIN, INPUT);
  pinMode(MOC, OUTPUT);

  attachInterrupt(digitalPinToInterrupt(INT_PIN), espera, RISING);
}

void loop()
{
  Cayenne.loop();

  digitalWrite(MOC, LOW);

  unsigned long previusMillis=millis();
  unsigned long tmr1Millis=millis();
  unsigned long tfinMillis=millis();

  DT = 0;

  while(btn==1)

```

```

{
digitalWrite(MOC, HIGH);

unsigned long currentMillis=millis();
unsigned long tinicioMillis=millis();
unsigned long tmrMillis=millis();

if (tmrMillis - tmr1Millis > 300000)
{
DT = map (value, 0, 100, 5000, 1500);
}

if (tinicioMillis - tfinMillis > tfin)
{
btn=0;
}

if (currentMillis - preuiusMillis > interval)
{
Cayenne.loop();
preuiusMillis = currentMillis;
}
}

//Subrutina de inerrupcion
void espera()
{
digitalWrite(MOC, LOW);
delayMicroseconds(DT);
}

```

```
}
```

```
//Esta función se activa cuando recibe un dato de Cayenne
```

```
CAYENNE_IN(ON_CHANNEL)
```

```
{
```

```
  btn = getValue.asInt();
```

```
}
```

```
CAYENNE_IN(VALUE_CHANNEL)
```

```
{
```

```
  value = getValue.asInt(); // 0 a 100
```

```
  CAYENNE_LOG("Channel %d, pin %d, value %d", VALUE_CHANNEL, tiempo,  
value);
```

```
}
```

```
CAYENNE_IN(TIME_CHANNEL)
```

```
{
```

```
  tfin = 60000 * getValue.asInt();
```

```
}
```

b) Lista de materiales

Cantidad	Descripción	Precio unitario	Total
01	Placa fenólica 10x10cm	\$20.00	\$20.00
01	BTA12	\$25.00	\$25.00
01	MOC3020	\$25.00	\$25.00
01	H11AA1	\$30.00	\$30.00
02	Borneras	\$15.00	\$30.00
02	Base para circuito integrado de 6 pines	\$10.00	\$10.00
01	Puente rectificador W02M	\$18.00	\$18.00
01	Microcontrolador ESP-WROOP-32	\$50.00	\$50.00
01	Tira de header hembra	\$10.00	\$10.00
04	Resistencias varias	\$5.00	\$5.00
		Total	\$223.00