



Tarjeta Smart House para una habitación

**César Andres Tejada Torres
John Alejandro Barahona Pineda**

Universidad del Quindío
Facultad de Ingeniería, Ingeniería Electrónica
Armenia, Colombia
2018

Tarjeta Smart House para una habitación

**César Andres Tejada Torres
John Alejandro Barahona Pineda**

Tesis o trabajo de grado presentada(o) como requisito parcial para optar al título de:
Ingeniero Electrónico

Director:
Ing. César Augusto Álvarez Gaspar

Línea de Investigación:
Internet de las Cosas
Universidad del Quindío
Facultad de Ingeniería, Ingeniería Electrónica
Armenia, Colombia
2018

Agradecimientos

John Alejandro Barahona Pineda

César Andres Tejada Torres

Contenido

Agradecimientos	v
Lista de símbolos	xI
Resumen	xII
1 Introducción	1
2 Objetivos	3
2.1 Objetivo General	3
2.2 Objetivos Específicos	3
3 Marco Teórico	4
3.1 Internet de las Cosas	4
3.1.1 Plataforma Heroku	4
3.1.2 Framework Laravel	5
3.1.3 HTTP	5
3.1.4 JSON	6
3.1.5 Base de Datos	7
3.2 Smart House	7
3.3 Hardware	7
3.3.1 ESP-WROOM-32	7
3.3.2 Corriente Alterna (AC)	8
3.3.3 Corriente Directa (DC)	9
3.3.4 Control de potencia AC por ángulo de fase	9
3.3.5 Control de Cargas DC	10
3.3.6 I2C	11
3.3.7 Sensores	11
3.4 Software	15
3.4.1 RTOS	15
3.4.2 ESP-IDF	16
3.4.3 Proteus	17

4 Desarrollo	18
4.1 Hardware	18
4.1.1 Alimentacion:	19
4.1.2 Entradas:	20
4.1.3 Salidas:	22
4.2 Firmware	26
4.3 Software	28
4.4 Prueba Beta	30
5 Resultados y Análisis	31
5.1 Software	31
5.1.1 Parte Pública	32
5.1.2 Parte Privada	33
5.1.3 Base de Datos	36
5.2 Firmware	36
5.2.1 Conexión a Internet vía Wi-Fi	37
5.2.2 Escritura de Datos en la Aplicación Web	39
5.2.3 Lectura de Datos de Internet	39
5.3 Hardware	40
5.4 Prueba Beta Cerrada	40
6 Conclusiones	42
7 Trabajos Futuros	43
Glosario	44
Bibliografía	46
Anexos	49

Lista de Figuras

3-1	ESP WROOM 32[26]	8
3-2	Representación gráfica del ángulo de disparo y de conducción del TRIAC y de la carga [28].	10
3-3	Ciclo Útil PWM	10
3-4	Modulo GY-30 [31]	11
3-5	Sensor de temperatura y humedad DHT11 [32]	12
3-6	Estructura del sensor MQ [33]	12
3-7	Modulo sendor de calidad de Aire MQ-135 [33]	13
3-8	Sensor de Lluvia [35]	14
3-9	Modulo de sensor de lluviar [Imagen Propia]	14
3-10	Sensor HCSR501 [37]	15
4-1	ESP32 creado en proteus [Imagen Propia]	18
4-2	Modulo conversor DC-DC [42]	19
4-3	Entrada de sensores[Imagen Propia]	20
4-4	Entrada para sensor de calidad de aire[Imagen Propia]	21
4-5	Entrada para sensores con resistencia de pull down[Imagen Propia]	21
4-6	Control por angulo de fase [Imagen Propia]	22
4-7	Triac BTA26600 [43]	22
4-8	Interruptor para cargas AC [Imagen Propia]	23
4-9	Detector de cruce por cero [Imagen Propia]	23
4-10	Schmitt trigger para el detector de cruce por cero [Imagen Propia]	24
4-11	Integrado L293D [Imagen Propia]	24
4-12	Puente h para control de motores DC [Imagen Propia]	25
4-13	Control para cargas DC [Imagen Propia]	25
4-14	Circuito tipico para el LM386 [Imagen Propia]	26
4-15	ESP-IDF [39]	27
4-16	Modelo-Vista-Controlador	29
4-17	ORM	30
5-1	Esquema Solución SmartHouse	31
5-2	Página de Inicio	32
5-3	Vista Pública	32
5-4	Vistas de Usuarios	33

5-5	Página de intercambio de datos	35
5-6	Vista para añadir reglas	36
5-7	Base de datos SmartHouse	36
5-8	Esquema de Tareas [Imagen Propia]	37
5-9	Conexión a Internet vía Wi-Fi ESP32	38
5-10	Aplicación Conexión a Wi-Fi	38
5-11	URL de la petición HTTP	39
5-12	Respues del la APP Web a la Tarjeta	40
5-13	Descripción caja eléctrica tarjeta SmartHouse	41

Lista de símbolos

Abreviaturas

Abreviatura	Término
<i>IR</i>	Infrarrojo
<i>SBC</i>	Computadoras de Placa Simple
<i>IoT</i>	Internet de las Cosas
<i>AC</i>	Corriente alterna
<i>DC</i>	Corriente continua
<i>PaaS</i>	Plataforma como Servicio
<i>JSON</i>	Notación de Objetos de JavaScript
<i>GPIO</i>	Entradas/Salidas de propósito general
<i>ADC</i>	Conversión analógico a digital
<i>DAC</i>	Conversión digital a analógico
<i>PWM</i>	Modulación por ancho de pulsos
<i>LED</i>	Diodo emisor de luz

Resumen

Este documento detalla el desarrollo y la implementación de una sistema de monitoreo y control para una habitación en un entorno de Smart House basado en el chip ESP32 que esta diseñado para el uso en aplicaciones que usan el paradigma de Internet de las cosas (IoT). La tarjeta permite la conexión de diferentes sensores y dispositivos, por medio de el diseño de diferentes circuitos para su adecuación y correcto funcionamiento; para este caso se utilizan los sensores más comunes en dicho entorno, como lo son, sensor de movimiento, temperatura, luminosidad, entre otros. Para la parte de control se tienen salidas de AC y DC con la posibilidad de variar la energía entregada a esta o simplemente encender y apagar. En cuanto a la visualización de los datos obtenidos de los sensores y también la interacción del usuario con los diferentes elementos conectados a la tarjeta se desarrolla una aplicación web implementada en Laravel, framework de desarrollo de aplicaciones web que utiliza PHP, esta aplicación web funciona a través de Heroku, una plataforma como servicio (PaaS), que proporciona toda la infraestructura necesaria para su funcionamiento.

Palabras clave: Aplicación Web, ESP32, Internet de las Cosas, Monitoreo, Smart House.

Abstract

This document .

Keywords: palabras clave en inglés(máximo 10 palabras, preferiblemente seleccionadas de las listas internacionales que permitan el indizado cruzado)

1 Introducción

A lo largo del crecimiento de los entornos inteligentes, como Smart House, se han realizado investigaciones con múltiples orientaciones. Las cuales están centradas en factores sociales como la comodidad y la seguridad, sin dejar de lado factores ambientales como el ahorro energético. En cuanto a una parte más técnica, estos procesos inteligentes están compuestos por software, hardware y firmware.

Las investigaciones hacia el entorno de Smart House se enfocan en monitorear y/o controlar múltiples aspectos de una casa. Para realizar esta tarea físicamente se usa un hardware, en el cual se ven inmersos la unidad central de procesamiento, los sensores y los actuadores. La unidad central de procesamiento se encarga del monitoreo y control del entorno.

Autores como Behan [4] y Cheque [5] han usado mini computadoras o computadoras de placa simple (SBC), como lo es Raspberry Pi, siendo esta una unidad central o unidad de mando, permitiendo el control de la iluminación en la casa. Sin embargo, no solo se usan tarjetas de prototipado, también se construyen nuevas tarjetas con funciones más específicas, así como Kusriyanto [13], el cual usó otro microcontrolador con mas pines como lo es el ATmega16, por lo cual este es otro modo de hacer eficiente el uso del hardware.

En Smart House, se ha implementado variedad de software, usado para la comunicación entre dispositivos móviles y el dispositivo central, más aun, que sea posible controlar la casa o realizar la comunicación entre el dispositivo central y los dispositivos esclavos. Del mismo modo, para ejecutar diferentes tareas como enviar datos al servidor, entre otras.

Así, por ejemplo, Cheque [5] ha desarrollado una aplicación basada en PHP, usando servidores Web como Lighttpd, el cual, tiene como soporte PostgreSQL para las bases de datos; esta aplicación se conecta a la unidad central de procesamiento con el fin de monitorear y controlar cargas LED; teniendo esto en cuenta, realizar aplicaciones en PHP es muy usado para controlar la casa, sea localmente o desde internet como realizó Kasmi [6]. Otro servidor externo, como Heroku, el cual fue usado por Kaneko [8] para la visualización de datos desde cualquier lugar, sin necesidad de tener el servidor local compartido a internet.

Los dispositivos inteligentes aumentan a gran velocidad, por la necesidad de estar siempre conectados, dando paso a aplicar esta conexión a diferentes espacios como el hogar, una casa

inteligente o Smart House se compone de diferentes dispositivos que se encuentran conectados a la red con posibilidad de acceso desde cualquier parte del mundo,

Por lo tanto brindar diferentes soluciones para contribuir con el crecimiento de este paradigma es muy útil, ya que amplia la gama de dispositivos presentes en el mercado, c

En este trabajo se realiza la construcción...

Este trabajo está organizado en cinco capítulos. El lector en el Capítulo 2 encontrará la descripción del objetivo general y objetivos específicos de este trabajo. En el Capítulo 3 se encuentra recopilado el marco teórico y herramientas hardware y software. El Capítulo 4 presentan los resultados de este trabajo, en el capítulo 5 se muestran las conclusiones y en el capítulo 6 y 7 los trabajos futuros con la bibliografía, por último se presentan los Anexos.

2 Objetivos

2.1. Objetivo General

Desarrollar una solución IoT para una habitación en un entorno de Smart House.

2.2. Objetivos Específicos

- Desarrollar un prototipo de una tarjeta inalámbrica para una habitación en un entorno Smart House.
- Desarrollar una aplicación web encargada de permitir la interacción del usuario con su habitación.
- Evaluar el desempeño del sistema en una prueba beta.

3 Marco Teórico

3.1. Internet de las Cosas

La internet de las cosas es un sistema de dispositivos de computación interrelacionados, máquinas mecánicas y digitales, objetos, animales o personas que tienen identificadores únicos y la capacidad de transferir datos a través de una red, sin requerir de interacciones humano a humano o humano a computadora.

IoT ha evolucionado desde la convergencia de tecnologías inalámbricas, sistemas microelectromecánicos, microservicios e Internet. La convergencia ha ayudado a derribar las paredes de silos entre la tecnología operativa y la tecnología de la información, permitiendo que los datos no estructurados generados por máquinas sean analizados para obtener información que impulse mejoras. [17]

Kevin Ashton, cofundador y director ejecutivo del Auto-ID Center de MIT, mencionó por primera vez la internet de las cosas en una presentación que hizo a Procter & Gamble en 1999. He aquí cómo Ashton explica el potencial de la internet de las cosas:

“Las computadoras de hoy –y, por lo tanto, la internet– dependen casi totalmente de los seres humanos para obtener información. Casi todos los aproximadamente 50 petabytes (un petabyte son 1.024 terabytes) de datos disponibles en internet fueron capturados y creados por seres humanos escribiendo, presionando un botón de grabación, tomando una imagen digital o escaneando un código de barras.

El problema es que la gente tiene tiempo, atención y precisión limitados, lo que significa que no son muy buenos para capturar datos sobre cosas en el mundo real. Si tuviéramos computadoras que supieran todo lo que hay que saber acerca de las cosas –utilizando datos que recopilaron sin ninguna ayuda de nosotros– podríamos rastrear y contar todo, y reducir en gran medida los desechos, las pérdidas y el costo. Sabríamos cuándo necesitamos reemplazar, reparar o recordar cosas, y si eran frescas o ya pasadas”. [18]

3.1.1. Plataforma Heroku

“Heroku es una plataforma en la nube basada en un sistema gestionado por contenedores, con servicios de datos integrados y un potente ecosistema, para desarrollar y ejecutar aplica-

ciones modernas. La experiencia de los desarrolladores de Heroku es un enfoque centrado en aplicaciones para la entrega de software, integrado con las herramientas y flujos de trabajo de desarrollador más populares de la actualidad” [19].

3.1.2. Framework Laravel

“Laravel es un framework de aplicaciones web con una sintaxis expresiva y elegante. Laravel intenta aliviar el dolor del desarrollo al facilitar las tareas comunes que se utilizan en la mayoría de los proyectos web, como la autenticación, el enrutamiento, las sesiones y el almacenamiento en caché.

Laravel tiene como objetivo hacer que el proceso de desarrollo sea agradable para el desarrollador sin sacrificar la funcionalidad de la aplicación. Con este fin, se ha intentado combinar lo mejor de lo que hemos visto en otros marcos web, incluidos los marcos implementados en otros lenguajes, como Ruby on Rails, ASP.NET MVC y Sinatra.

Laravel es accesible, pero potente, y proporciona potentes herramientas necesarias para aplicaciones grandes y robustas. Una magnífica inversión de contenedores de control, un sistema de migración expresivo y un soporte de prueba de unidades estrechamente integrado le brindan las herramientas que necesita para construir cualquier aplicación” [20].

3.1.3. HTTP

El protocolo de transferencia de hipertexto (HTTP) es un protocolo de la capa de aplicación en el modelo OSI, se usa para transmitir documentos hipermedia, como HTML. Fue diseñado para la comunicación entre navegadores web y servidores web, pero también se puede usar para otros propósitos. HTTP sigue un modelo clásico de cliente-servidor, con un cliente que abre una conexión para realizar una petición, luego esperando la respuesta. HTTP es un protocolo sin estado, significa que el servidor no conserva ningún dato entre dos solicitudes. Para realizar las peticiones este protocolo cuenta con diferentes métodos [21].

Metodos de solicitud HTTP: HTTP define un conjunto de métodos de solicitudes para indicar la acción deseada que se realizará para un recurso determinado. Aunque pueden ser sustantivos, estos métodos algunas veces se denominan verbos HTTP. Cada uno de ellos implementa una semántica diferente, pero algunas características comunes son compartidas por un grupo de ellos [22].

- GET: este método solicita una representación del recurso especificado. Las peticiones que lo usan, solo deben regresar datos.

- HEAD: este método solicita una respuesta igual a una petición GET, pero sin el cuerpo de la respuesta.
- POST: se usa para enviar una entidad al recurso especificado, causando a menudo un cambio en el estado o efectos secundarios en el servidor.
- PUT: reemplaza todas las representaciones actuales del recurso destino con la petición de carga útil.
- DELETE: esta petición elimina el recurso especificado.
- CONNECT: establece un túnel para el servidor identificado por el recurso de destino.
- OPTIONS: se usa para describir las opciones de la comunicación para el recurso destino.
- TRACE: realiza una prueba de mensaje loop-back a lo largo de la ruta del recurso de destino.
- PATCH: se usa para realizar modificaciones parciales a un recurso.

3.1.4. JSON

“JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, al igual que para las máquinas resulta simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos” [23].

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras [23].

3.1.5. Base de Datos

“Se define una base de datos como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular.

Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más columnas y filas. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro” [24].

Para la gestión de datos es muy común encontrar las funciones básicas de crear, leer, actualizar y borrar (CRUD), para que el usuario final pueda acceder a estos sin necesidad de realizar peticiones directamente a la base de datos, sino en una interfaz gráfica (GUI).

3.2. Smart House

El concepto de Smart House implica tres características básicas. En primer lugar, el monitoreo a través de redes de sensores para obtener información sobre la casa y sus residentes. En segundo lugar, los mecanismos que controlan el uso de la comunicación entre dispositivos para permitir la automatización y el acceso remoto. Por último, las interfaces de usuario, como los teléfonos inteligentes y las computadoras que permiten a los usuarios especificar las preferencias, así como presentar información a las personas acerca de estas preferencias.

Smart House es un entorno que tiene sistemas sofisticados a través de los cuales se pueden controlar algunas de las cosas de la casa, como luces, puertas, ventanas, además puede racionalizar el consumo de energía, entre otras funciones mediante el uso de sensores. Básicamente, uno de los beneficios más importantes del uso de la tecnología en las casas, es la prestación de servicios a las personas.[10]

3.3. Hardware

3.3.1. ESP-WROOM-32

Es un potente módulo MCU Wi-Fi + BT + BLE que se dirige a una amplia variedad de aplicaciones, desde redes de sensores de baja potencia hasta las tareas más exigentes, como codificación de voz, transmisión de música y decodificación de MP3, además de su reducido tamaño, como se observa en la figura 3-1.

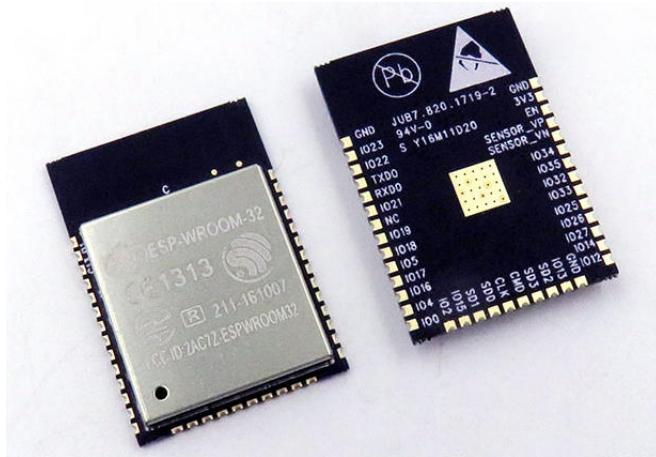
En el núcleo de este módulo está el chip ESP32-D0WDQ6. El chip integrado está diseñado para ser escalable y adaptable. Hay dos núcleos de CPU que se pueden controlar individual-

mente, y la frecuencia del reloj es ajustable de 80 MHz a 240 MHz. El usuario también puede apagar la CPU y utilizar el coprocesador de baja potencia para monitorear constantemente los periféricos en busca de cambios o cruces de umbrales. ESP32 integra un amplio conjunto de periféricos, que van desde sensores táctiles capacitivos, sensores Hall, interfaz de tarjeta SD, Ethernet, SPI de alta velocidad, UART, I2S e I2C.

La integración de Bluetooth, Bluetooth LE y Wi-Fi garantiza que se pueda orientar una amplia gama de aplicaciones, el uso de Wi-Fi permite un gran alcance físico y conexión directa a Internet a través de Wi-Fi, mientras usa Bluetooth, le permite al usuario conectarse convenientemente al teléfono o transmitir balizas de baja energía para su detección. La corriente de reposo del chip ESP32 es inferior a 5 uA, lo que lo hace adecuado para aplicaciones de electrónica con batería y portátiles. ESP32 admite una velocidad de datos de hasta 150 Mbps y una potencia de salida de 20.5 dBm en la antena para garantizar el rango físico más amplio.

El sistema operativo elegido para ESP32 es freeRTOS con LwIP; TLS 1.2 con aceleración de hardware está integrado también. También se admite la actualización segura (cifrada) a través del aire (OTA), de modo que los desarrolladores puedan actualizar continuamente sus productos incluso después de su lanzamiento.[25]

Figura 3-1: ESP WROOM 32[26]



3.3.2. Corriente Alterna (AC)

“Es un tipo de corriente eléctrica, en la que la dirección del flujo de electrones va y viene a intervalos regulares o en ciclos. La corriente que fluye por las líneas eléctricas y la electricidad disponible normalmente en las casas procedente de los enchufes de la pared es corriente alterna. La corriente estándar utilizada en los EE.UU. es de 60 ciclos por segundo (es decir, una frecuencia de 60 Hz); en Europa y en la mayor parte del mundo es de 50 ciclos por segundo (es decir, una frecuencia de 50 Hz.)”. [27]

3.3.3. Corriente Directa (DC)

“Es la corriente eléctrica que fluye de forma constante en una dirección, como la que fluye en una linterna o en cualquier otro aparato con baterías es corriente continua.

Una de las ventajas de la corriente alterna es su relativamente económico cambio de voltaje. Además, la pérdida inevitable de energía al transportar la corriente a largas distancias es mucho menor que con la corriente continua”. [27]

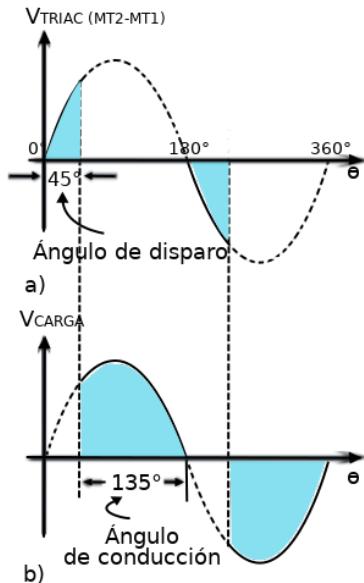
3.3.4. Control de potencia AC por ángulo de fase

“Los SCR y los TRIAC, permiten aplicar una técnica muy conveniente y eficaz para controlar el voltaje promedio y por lo tanto la potencia aplicada a una carga, cambiando el ángulo de fase con el cual la fuente de voltaje se aplica a ésta. Esta técnica de control de voltaje es muy usada en las aplicaciones de regulación de motores, iluminación y temperatura, por ser el voltaje la variable principal en estos tres procesos”.[28]

“Para entender como se controla el ángulo de fase, por medio de un TRIAC conectado en serie con la carga, se puede asumir que el TRIAC se comporta idealmente como un interruptor controlador por la corriente de compuerta Ig que se cierra o se abre ante su presencia o ausencia. Observando la figura 3-2 puede verse el control de una onda seno de tensión con un período con un período de 360 grados; en la parte (a) de la figura se muestra la tensión a través del TRIAC, mientras que en la parte (b) se ve la tensión sobre la carga; allí puede verse que el TRIAC se comporta como un circuito abierto durante los primeros 45 grados de cada semiciclo, y todo el voltaje cae en sus terminales eliminando el flujo de corriente por la carga. La porción del semiciclo durante la cual se presenta esta situación se conoce como ángulo de disparo”[28].

“Una vez el TRIAC es disparado a través de su terminal de compuerta (G), éste se engancha y se comporta como un interruptor cerrado, permitiendo que todo el voltaje se aplique a la carga durante los 135 grados restantes de cada semiciclo. La porción del semiciclo durante la cual el TRIAC conduce se denomina ángulo de conducción”[28].

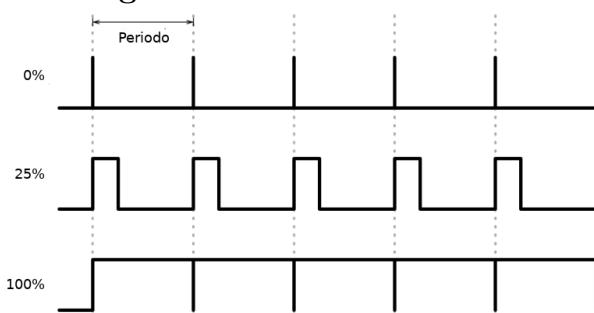
Figura 3-2: Representación gráfica del ángulo de disparo y de conducción del TRIAC y de la carga [28].



3.3.5. Control de Cargas DC

Los transistores como switch permiten controlar las cargas de corriente continua con ayuda de una señal PWM que los activa o desactiva. Las cargas de corriente continuas típicas como los motores y LED's, a parte de poder funcionar en dos estados, encendido y apagado, pueden ser controladas mediante la modulación por ancho de pulso (PWM), ya que al variar el ancho de pulso de la señal eléctrica se varía la cantidad de energía entregada a la carga, por ejemplo, si es un LED este cambio se refleja en su intensidad lumínica y si es un motor DC el cambio está en su velocidad de giro [29]. Este control se produce gracias a que en esta modulación se varía su ciclo útil, cambiando el tiempo en que la señal eléctrica está en alto durante un periodo, por lo tanto si el ciclo útil es del 10% la energía entregada es poca, en comparación, con un ciclo útil del 50% o 100% como se observa en la figura 3-3.

Figura 3-3: Ciclo Útil PWM



3.3.6. I2C

“I2C es un puerto y protocolo de comunicación serial, define la trama de datos y las conexiones físicas para transferir bits entre 2 dispositivos digitales. El puerto incluye dos cables de comunicación, SDA y SCL. Además el protocolo permite conectar hasta 127 dispositivos esclavos con esas dos líneas, con hasta velocidades de 100, 400 y 1000 kbits/s. También es conocido como IIC ó TWI – Two Wire Interface” [30].

3.3.7. Sensores

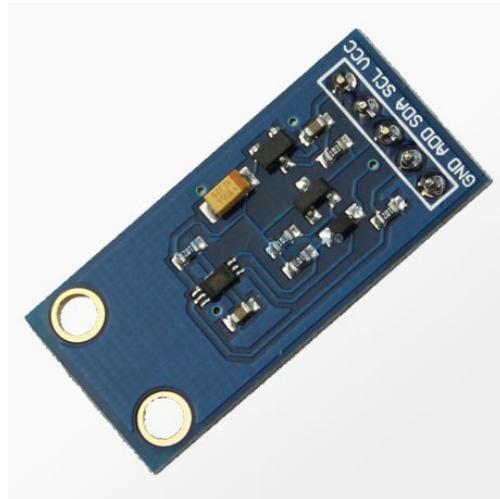
Módulo GY-30

“Sensor GY-30 BH1750FVI. Es un sensor digital de intensidad de luz ambiente, tiene un conversor ADC de 16bits interno y comunicación por I2C como se observa en la figura 3-4. Esta es una versión mejorada del típico sensor de luz a base de un LDR, el cual simplemente entrega un valor analógico. Compatible con Arduino, PIC, etc.

El módulo BH1750 es un sensor de luz, que a diferencia del LDR es digital y nos entrega valores de medición en Lux (lumen /m²) que es una unidad de medida estándar para el nivel de iluminación (iluminancia). Tiene alta precisión y un rango entre 1 – 65535 lx el cual es configurable.

La interfaz de comunicación es I2C pudiéndolo implementar en la mayoría de micro controladores, el módulo aparte de los pines de alimentación y pines I2C tiene un pin para establecer la dirección”.[31]

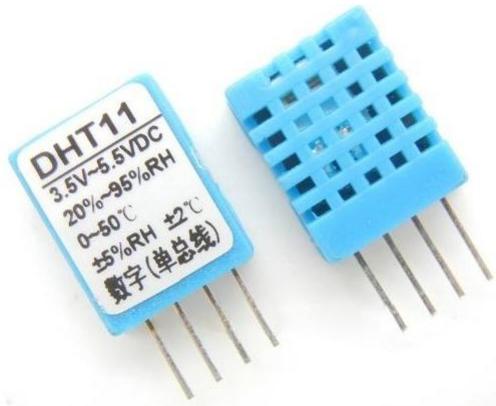
Figura 3-4: Modulo GY-30 [31]



Temperatura y Humedad DHT11

“El DHT11 es un sensor de temperatura y humedad digital de bajo costo. Utiliza un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no hay pines de entrada analógica). Es bastante simple de usar, pero requiere sincronización cuidadosa para tomar datos. El único inconveniente de este sensor es que sólo se puede obtener nuevos datos una vez cada 2 segundos, así que las lecturas que se pueden realizar serán mínimo cada 2 segundos”. [32]

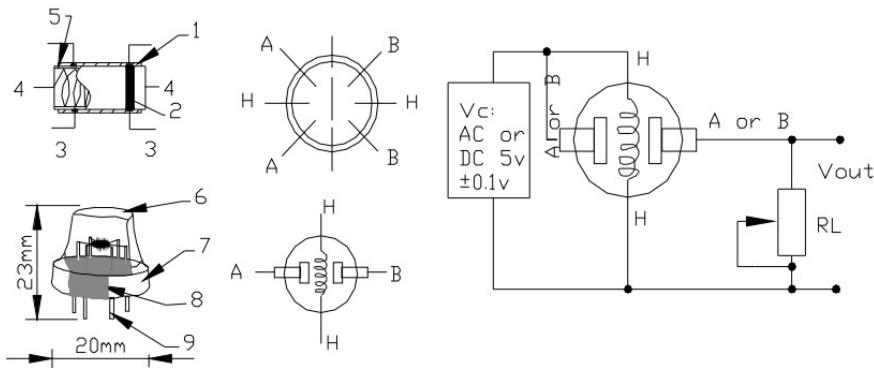
Figura 3-5: Sensor de temperatura y humedad DHT11 [32]



Módulo sensor de calidad de aire MQ-135

La serie MQ de sensores de gas son sensores analógicos por lo que son fáciles de implementar con cualquier microcontrolador que posea un conversor analógico digital (ADC) adecuado. Estos sensores son electroquímicos y varán su resistencia con la exposición a determinados gases, internamente poseen un calentador que se encarga de aumentar la temperatura interna para que el sensor pueda reaccionar con los gases provocando un cambio de valor en la resistencia, su estructura interna se puede observar en la figura 3-6.[33]

Figura 3-6: Estructura del sensor MQ [33]



El Sensor Calidad Aire MQ135 se utilizan en equipos de control de calidad del aire para edificios y oficinas, son adecuados para la detección de NH₃, NOx, alcohol, benceno, humo, CO₂, etc. Además de que estos sensores vienen en módulos como se observa en la figura 3-7, lo que facilita el uso de estos, simplemente se debe conectar al microcontrolador sin necesidad de hacer algún circuito de acople. [33]

Figura 3-7: Modulo sendor de calidad de Aire MQ-135 [33]



Este sensor es sensible en similar proporción a los gases mencionados, con lo que se puede determinar si el aire está limpio o si existe presencia de algún gas nocivo.

Los valores de salida de este sensor no son valores absolutos, simplemente proporciona una salida analógica que se debe monitorear y ser comparada con los valores típicos proporcionados en la hoja de datos como se menciona anteriormente.[34]

Sensores de Estado

Los sensores de estado describen si la variable esta en alto (1) o en bajo (0), para estos sensores se tienen diferentes variables típicamente, como el estado de una puerta o una ventana (abierta o cerrada), la lluvia, el movimiento.

Módulo detector de lluvia: Es un módulo relativamente simple que consiste en una serie de pistas conductoras organizadas de forma paralela e impresas sobre una placa de baquelita como se observa en la figura 3-8. La separación entre sus pistas es muy pequeña, con el fin de crear un corto circuito cada vez que las pistas se mojan, ya que es un circuito abierto y el agua hace que se cree un camino de baja resistencia entre las pistas que tienen diferente potencial (Vcc-GND). La corriente que fluye a través de estas pistas se ve limitada por resistencias de 10K en cada conductor, lo que impide que el corto circuito que se genera cuando se moja la placa vaya a estropear el micro controlador.[35]

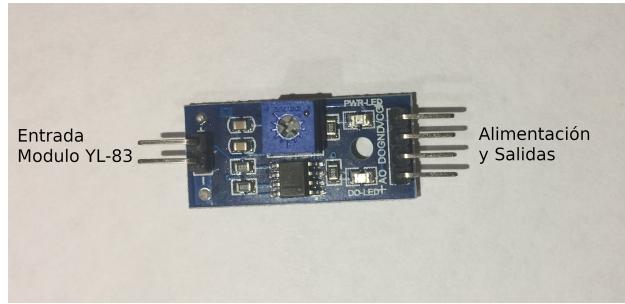
Figura 3-8: Sensor de Lluvia [35]



El circuito de control es el que posee las resistencias limitadoras de corriente y es el encargado de alimentar el módulo de la figura 3-8. Como se observa en la figura 3-9 posee un amplificador operacional, específicamente el circuito integrado LM392. Este es el encargado de amplificar el pequeño diferencial de voltaje que se genera cuando una gota de agua cae sobre las pistas del módulo. Aquí es donde se genera la señal de salida que puede ser del tipo analógica o digital. La señal digital oscilará entre los valores HIGH y LOW dependiendo de si hay agua o no sobre las pistas de la placa.

La salida analógica entregará un nivel de voltaje que variará dependiendo de la cantidad de agua que haya sobre el módulo.[35]

Figura 3-9: Modulo de sensor de lluvia [Imagen Propia]



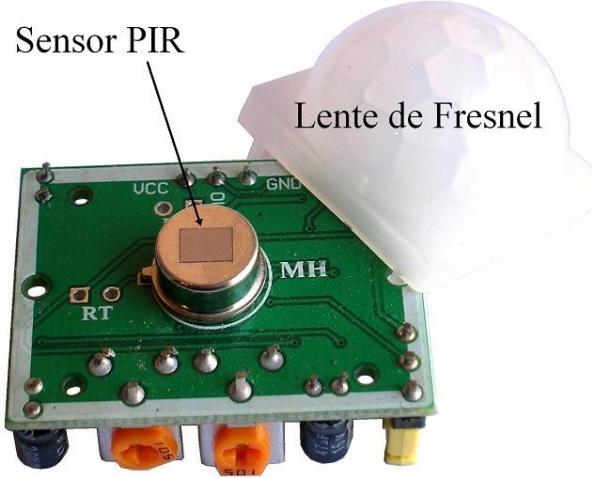
Módulo PIR HC-SR501: La función de los sensores PIR es detectar movimiento, normalmente se busca detectar el movimiento de una persona dentro del rango del sensor. Son baratos, pequeños, de bajo consumo y fáciles de utilizar, además no se desgastan. Normalmente se encuentran en electrodomésticos y gadgets para la oficina o el hogar. Son conocidos

como PIR, “Sensores Infrarrojos” o “Sensores de movimiento”.

Este módulo contiene un sensor Piroelectrico, el cual puede detectar niveles de radiación infrarroja como se observa en la figura **3-10**. Este sensor de movimiento está dividido en 2 mitades, la razón para esto es que se busca la diferencia en el movimiento y no el promedio. Las dos mitades están unidas de modo que se cancelan una a otra. Entonces, si una mitad recibe más o menos radiación IR, la salida cambiará a Alto o Bajo. [36]

El módulo PIR modelo HC-SR501, es pequeño y de bajo costo como se observa en la figura , incorpotando la tecnología más reciente en sensores infrarrojos pasivos para detectar movimiento. La emisión infrarroja se da en personas por su temperatura comporal y en animales (mamíferos), ya que emiten una radiación similar a los humanos. [37]

Figura 3-10: Sensor HCSR501 [37]



3.4. Software

3.4.1. RTOS

Los sistemas operativos en tiempo real, tienen como parámetro clave al tiempo, ya que en gran variedad de situaciones, como por ejemplo, un proceso industrial, se requiere recolectar múltiples datos, los cuales son usados para el control de múltiples procesos, los cuales deben ser ejecutados en determinados instantes, de no ser así, podría causar desde la mala ejecución de una tarea, hasta un accidente según la delicadeza del proceso.

Para procesos con nula tolerancia a fallos, se conoce como un sistema en tiempo real duro, muchos de estos sistemas se encuentran en el control de procesos industriales, en aeronáutica, en la milicia y en áreas de aplicación similares. el caso contrario, cuando se tiene cierta

tolerancia a que muy ocasionalmente existan fallos, se conoce como sistema en tiempo real suave, los sistemas de audio digital o de multimedia están en esta categoría. Los teléfonos digitales también son ejemplos de sistema en tiempo real suave. [38]

“Como en los sistemas en tiempo real es crucial cumplir con tiempos predeterminados para realizar una acción, algunas veces el sistema operativo es simplemente una biblioteca enlazada con los programas de aplicación, en donde todo está acoplado en forma estrecha y no hay protección entre cada una de las partes del sistema. Un ejemplo de este tipo de sistema en tiempo real es freeRTOS. Las categorías de sistemas para computadoras de bolsillo, sistemas integrados y sistemas en tiempo real se traslanan en forma considerable. Casi todos ellos tienen por lo menos ciertos aspectos de tiempo real suave. Los sistemas integrados y de tiempo real sólo ejecutan software que colocan los diseñadores del sistema; los usuarios no pueden agregar su propio software, lo cual facilita la protección.

Los sistemas de computadoras de bolsillo y los sistemas integrados están diseñados para los consumidores, mientras que los sistemas en tiempo real son más adecuados para el uso industrial. Sin embargo, tienen ciertas características en común”. [38]

3.4.2. ESP-IDF

ESP-IDF es el entorno de desarrollo oficial para el ESP32 desarrollado por Espressif System, el cual mediante una serie de comandos específicos escritos en la terminal (para el caso de linux), permite realizar una configuración del ESP32 en cuanto a su funcionamiento, es decir, permite encender o apagar características como el WiFi, el Bluetooth o realizar particiones de memoria, ademas de esto, se puede cargar el código por el puerto USB al ESP32, al igual que se puede visualizar la información generada por el ESP32 por este mismo puerto. Este entorno se encuentra construido con diferentes características y APIs, algunas de ellas se mencionan a continuación. [39]

FreeRTOS: este framework esta desarrollado sobre este sistema operativo de tiempo real, tomando de este diferentes funcionalidades como la tareas y las colas.

Wi-Fi: el modulo ESP-WROOM-32 posee la capacidad en el hardware para la conexión a una red Wi-Fi, y este software proporciona las librerías y las funcionalidades para realizar dicha conexión y enviar o recibir datos.

Particiones: el MCU contiene una memoria flash, la cual por medio de este framework se le pueden realizar particiones de memoria, dedicadas a guardar el programa, y también para guardar ciertos archivos. Estas se configuran por medio de un fichero csv.

Consola: el framework provee una librería para realizar la programación de una consola, para realizar diferentes tareas dentro del sistema, las cuales se pueden programar.

HTTP Request: para realizar diferentes peticiones HTTP como las descritas anteriormente, el entorno de desarrollo cuenta con la librería LWIP para realizarlas.

Timers: este framework provee funciones para disponer de los timers de 64 bits que posee el esp32 y además para configurar sea un timer periódico o de un solo disparo.

3.4.3. Proteus

Proteus combina facilidad de uso con características de gran alcance para ayudar a diseñar, probar y el diseño de PCB profesionales. Con casi 800 variantes de microcontroladores listos para la simulación directamente desde el esquemático, además de contar con uno de los paquetes de diseño de PCB profesionales más intuitivas en el mercado y un autoruteo de clase mundial incluido como estándar. [40]

Además es una herramienta muy completa y potente de simulación de circuitos y diseño de PCBs. Dentro de la simulación de circuitos admite componentes pasivos, digitales, analógicos y componentes más complejos como LCDs y motores. Por tanto, se puede hacer casi cualquier cosa, el programa está pensado para que una vez se tenga el circuito diseñado se pueda pasar a una PCB [41]. Adicionalmente los dispositivos que no se encuentren se pueden agregar, ya sea de la página oficial o construyéndolos dentro de la herramienta.

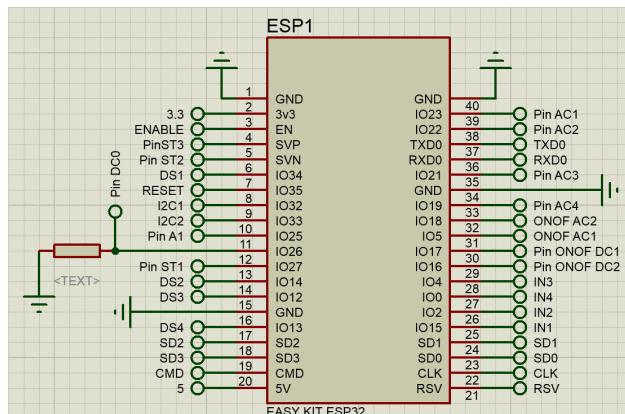
4 Desarrollo

4.1. Hardware

El prototipo para Smart House tiene como objetivo tanto monitorear el entorno de aplicación, como controlarlo por medio de mecanismos como motores o dispositivos de iluminación, para ello está equipada con etapas de potencia de corriente alterna y directa, etapa de adquisición de datos, entre otras características que permitan cumplir con los objetivos planteados.

El prototipo fue diseñado en el software Proteus, desde el esquemático hasta la placa de circuito impreso (PCB), en la figura 4-1 se observa el esquemático de la tarjeta ESP32 construido en Proteus junto con su distribución de pines, además de sus conexiones correspondientes dentro de este software. El prototipo está separado en dos secciones, la etapa de potencia AC y la etapa DC, en la última, se encuentra la mayor parte de circuitos que funcionan con corriente directa.

Figura 4-1: ESP32 creado en proteus [Imagen Propia]



En los siguientes ítems se resaltarán las características más importantes que lleva el circuito:

4.1.1. Alimentacion:

Corriente alterna (AC):

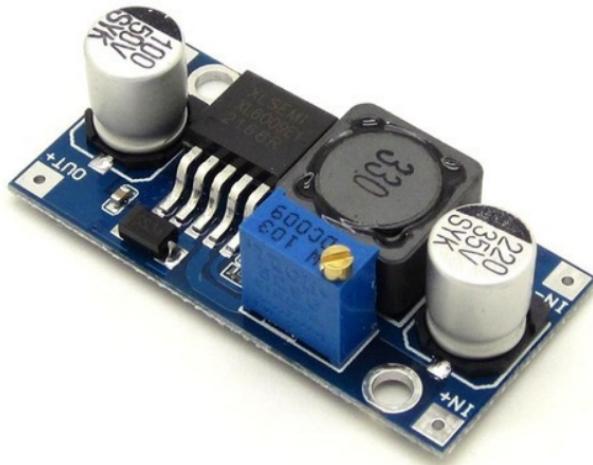
El prototipo recibe el voltaje directamente de la red eléctrica a la cual está conectado el entorno de aplicación, el cual está pensado para una habitación dentro de una Smart House; en el caso de Colombia, la red doméstica comúnmente otorga 110V AC, los cuales son regulados para el funcionamiento adecuado del prototipo, como la etapa de potencia AC y el detector de cruce por cero para sincronizar la tarjeta a dicha red eléctrica.

Corriente directa (DC):

Para la alimentación DC del circuito se hace uso de un conversor AC-DC que regula el voltaje de la red eléctrica a 12V DC, con los cuales se manejará la etapa de potencia DC, además de ser usados por dos modulos conversores DC-DC, mostrados en la figura 4-2, ambos con entradas de 12V DC y con salidas a los niveles lógicos comunes, tales como 5V y 3.3V, empleados para alimentar dispositivos como opto acopladores, transistores BJT o relevadores con activación de 5V, así como también la tarjeta de prototipo ESP32.

Cabe resaltar que la tarjeta permite una entrada externa de 12VDC a 5A si se desean controlar cargas a un máximo de 50W.

Figura 4-2: Modulo conversor DC-DC [42]



4.1.2. Entradas:

Sensores:

El prototipo viene equipado con una etapa de adquisición de datos con capacidad entre 7 a 134 sensores, pues está equipado con entrada I2C, ampliando la capacidad de dispositivos conectados, lo cual también permitiría adicionar tareas más específicas en escenarios que lo requieran.

Para realizar pruebas del prototipo, se hacen uso de 5 sensores para medir magnitudes y situaciones en el entorno tal como calidad del aire, temperatura y humedad, luz visible, movimiento y presencia de lluvia, debido a que estas magnitudes o estados se encuentran en casi cualquier entorno. Para esto, teniendo en cuenta que el ESP32 funciona en voltajes lógicos de 3.3V, se tienen 4 entradas de sensores directamente conectados a los pines de la tarjeta, con la capacidad de cambiar el voltaje de alimentación para 3 de ellos, tal como se muestra en la figura 4-3, pues en el mercado se encuentran sensores que manejan voltajes de alimentación ya sea de 3.3V o 5V, mientras que la cuarta entrada se encuentra alimentada con 5V, ya que tiene un uso específico en las pruebas para el sensor de calidad de aire, dicha entrada viene acondicionada con un diodo zener en contraposición, para evitar que la tarjeta ESP32 tenga un voltaje de entrada superior a 3.3V, como se observa en la figura 4-4.

Las tres entradas para sensores de estado (ST1, ST2, ST3), a diferencia de las demás, se encuentran conectadas a pines de la tarjeta que no presentan resistencia de pull down por software, por ello se agrega dicha resistencia al sistema, tal como se observa en la figura 4-5.

Figura 4-3: Entrada de sensores[Imagen Propia]

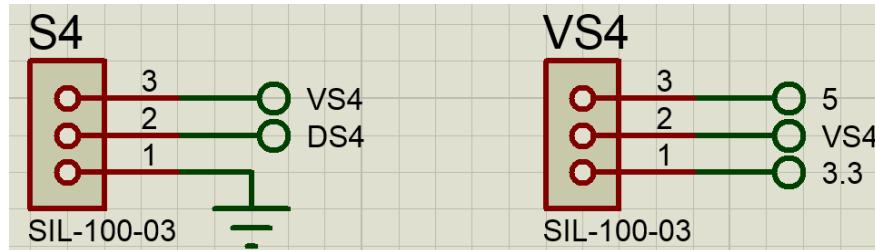


Figura 4-4: Entrada para sensor de calidad de aire[Imagen Propia]

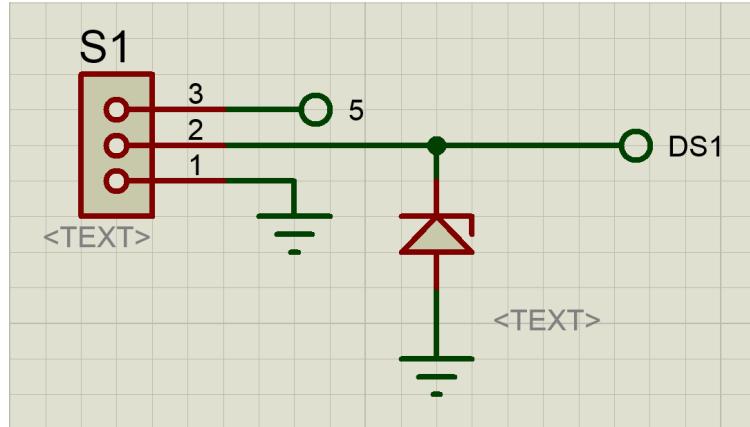
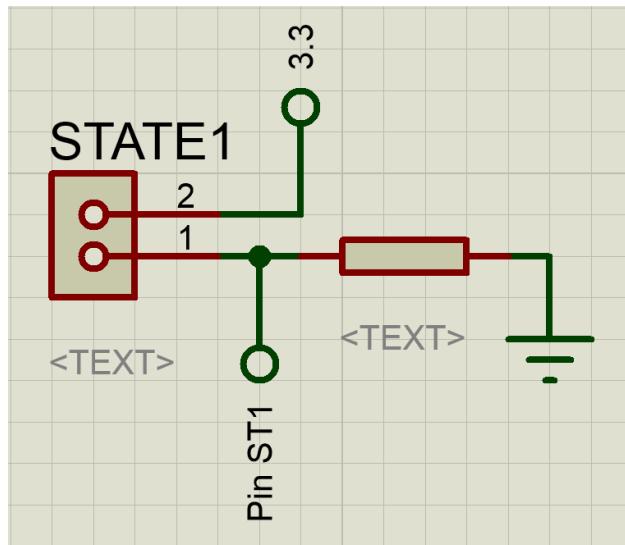


Figura 4-5: Entrada para sensores con resistencia de pull down[Imagen Propia]



Calibración de audio:

Para calibrar la salida audible se hace uso de una resistencia variable (Potenciómetro), el cual permite regular el voltaje de entrada al circuito de amplificación, mostrado en la figura 4-14 y será descrito en el presente capítulo en la sección de salidas del hardware.

Botón enable:

Presionando el botón enable se reinicia la tarjeta ESP32, junto con su firmware.

Botón reset:

Presionando el botón reset se reinician las credenciales ingresadas para la conexión correcta del ESP32 a la red wifi.

4.1.3. Salidas:

Etapa de potencia AC:

La etapa de potencia AC del prototipo, está diseñada para una potencia de 2000W en un total de seis cargas, cuatro de ellas cuentan con un circuito para el control por ángulo de fase, como se observa en la figura 4-6, con una capacidad individual de 500W, gracias a el TRIAC de potencia BTA26600, mostrado en la figura 4-7, el cual soporta una corriente máxima de 25A; para proteger el ESP32, se hizo uso de optoacopladores MOC3021, debido a su capacidad para aislar circuitos de forma óptica.

Figura 4-6: Control por angulo de fase [Imagen Propia]

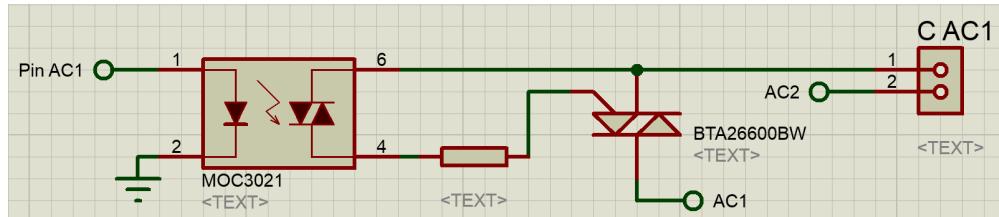
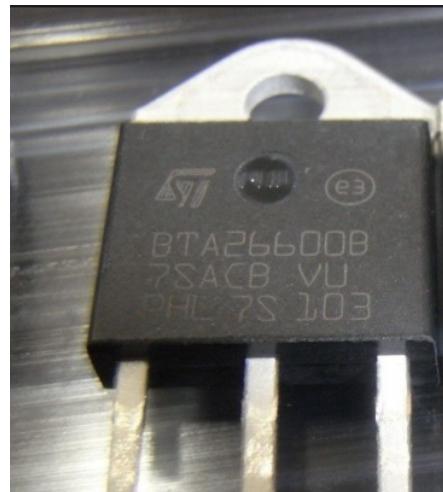
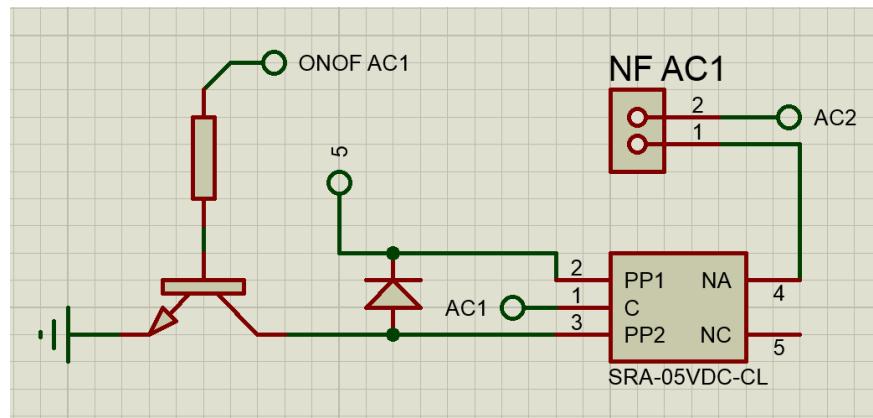


Figura 4-7: Triac BTA26600 [43]



Las dos cargas restantes corresponden a un sistema de encendido y apagado, cuyo funcionamiento se basa en un relevador SRA-05VDC-CL activado a 5V por medio de un transistor BJT como switch, gracias a este relevador, la salida tiene capacidad para cargas de 200W cada una, en la figura 4-8 se observa el circuito diseñado en proteus. Para proteger el ESP32 el prototipo se vale del relevador, puesto que presenta un aislamiento magnético por la naturaleza de su funcionamiento.

Figura 4-8: Interruptor para cargas AC [Imagen Propia]



Dentro de la etapa AC se encuentra el detector de cruce por cero, el cual se vale de un fototransistor 4N25, debido a su alta capacidad de aislamiento, soportando más de 2500VAC, tomando la onda rectificada completa y pasándola a un nivel lógico de 3.3V, esta parte del circuito se observa en la figura 4-9; para que la señal sea más confiable se hace uso de un Schmitt-Trigger CD40106 mostrado en la figura 4-10, valiéndose de la histéresis de voltaje para garantizar que su señal de salida sea poco susceptible al ruido [44].

Figura 4-9: Detector de cruce por cero [Imagen Propia]

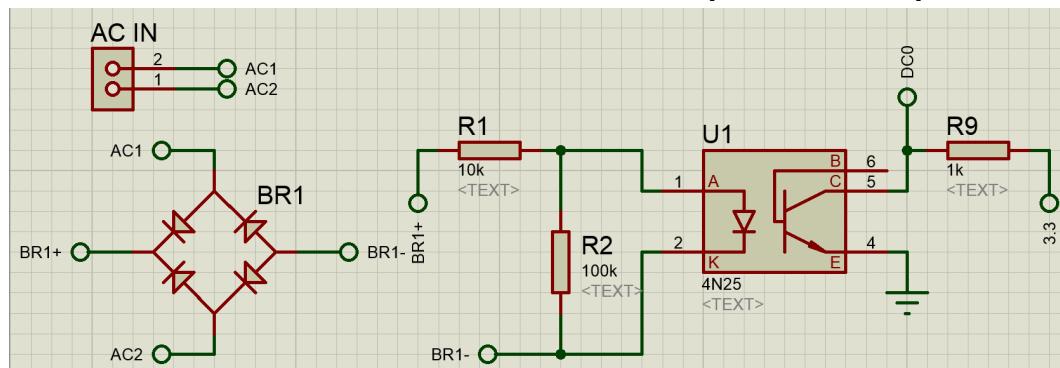
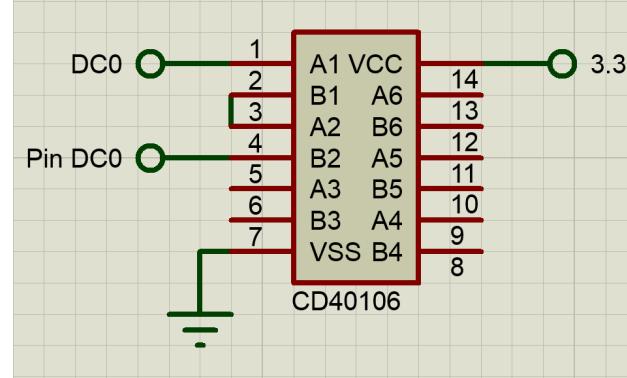


Figura 4-10: Schmitt trigger para el detector de cruce por cero [Imagen Propia]



Etapa DC:

La etapa DC cuenta con cuatro salidas de control diseñadas para cargas de 12V, de las cuales, dos de ellas están diseñadas con enfoque a motores, puesto que está equipada con control de velocidad a base de PWM e inversión de giro con un puente h usando transistores mosfet IRLZ44N; este puente h está controlado por un circuito integrado L293D, que garantiza un voltaje Vgs adecuado para la correcta activación del los transistores del puente h; este circuito se muestra en la figura 4-11 y 4-12.[45].

Las dos salidas restantes también cuentan con mosfet IRLZ44N, y su control también es a base de PWM, mas no permite hacer inversión de giro, por lo cual se enfoca a dispositivos como lámparas LEDs, el diseño en proteus se muestra en la figura 4-13.

Figura 4-11: Integrado L293D [Imagen Propia]

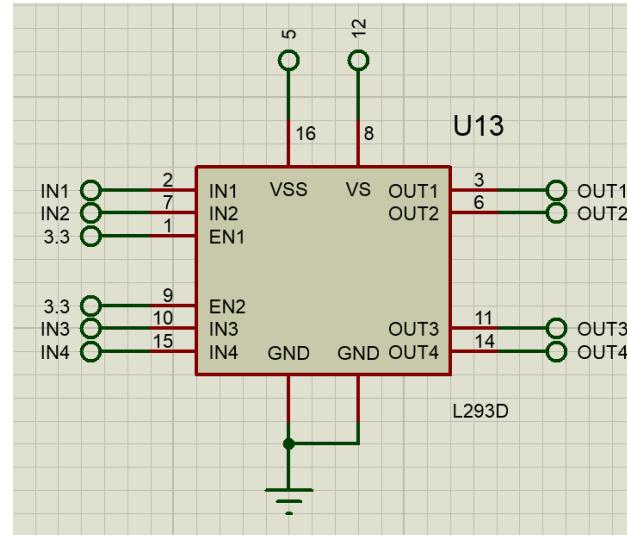
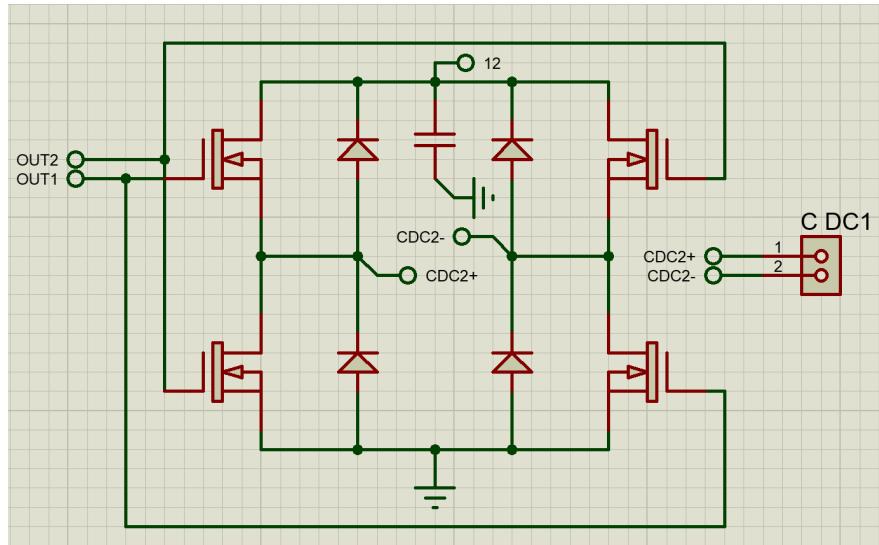
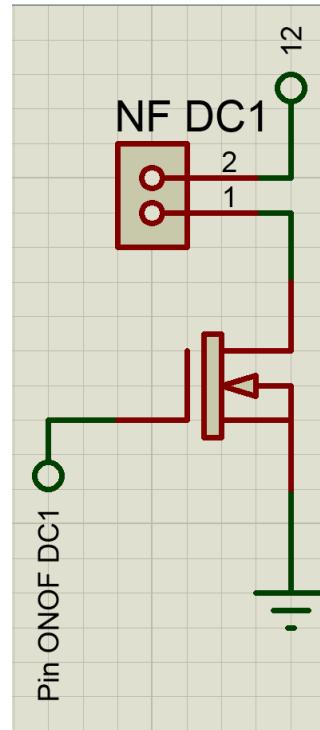


Figura 4-12: Puente h para control de motores DC [Imagen Propia]**Figura 4-13:** Control para cargas DC [Imagen Propria]**Salida audible:**

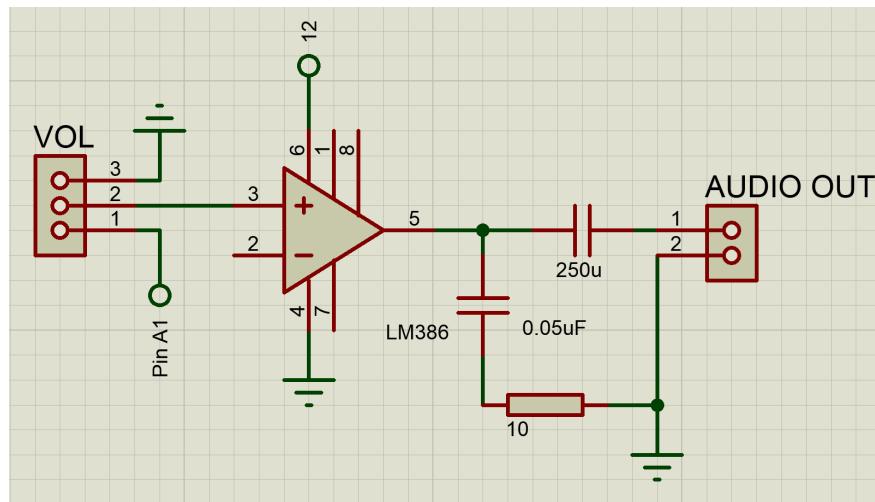
La salida audible está diseñada para emitir desde sonidos a una sola frecuencia, o sonido mono estéreo, caso dado cuando se activa una regla programada desde la aplicación web,

enfocada a las cargas de encendido y apagado, tanto de la etapa de potencia AC como la etapa DC; el sonido emitido por el prototipo corresponde a una voz con tonalidad femenina, pronunciando el estado en el cual se configura la carga según la regla (ya sea encendido, o apagado).

El circuito utilizado para la salida audible está basado en el amplificador de audio LM386, implementando el circuito típico de aplicación ilustrado en su datasheet [46], en la figura 4-14 se observa este circuito implementado en proteus.

Como se mencionó anteriormente, el circuito presenta un potenciómetro a la entrada para calibrar el voltaje de esta, con el fin de no saturar la entrada para que el sonido a la salida sea lo mas fiel posible.

Figura 4-14: Circuito tipico para el LM386 [Imagen Propia]

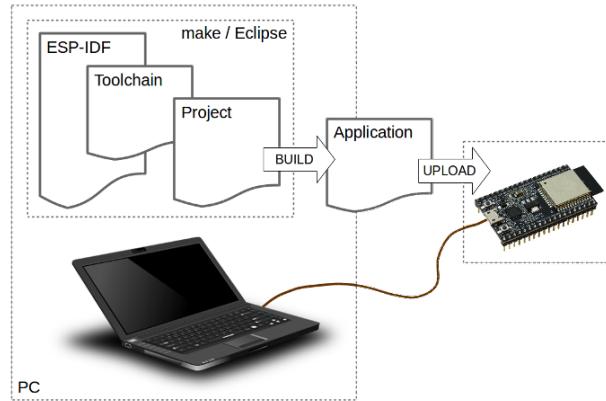


4.2. Firmware

El firmware se desarrolla sobre el framework o SDK oficial de Espressif Systems, ESP-IDF el cual posee una documentación [39] muy útil a la hora de utilizar las diferentes APIs que este posee; para el desarrollo de la aplicación es necesario contar con los diferentes requisitos tal como se observa en la figura 4-15. Este firmware incluye un kernel de tiempo real llamado FreeRTOS, el cual da soporte al manejo de los diferentes recursos del sistema; Al ser este un RTOS, las funciones se definen mediante las tareas, entonces para cada funcionalidad de la tarjeta o grupo de funcionalidades se desarrolla una o varias tareas para realicen las acciones adecuadas, por ejemplo, en el caso de los sensores, cada uno tiene una tarea para lectura y para gestión de datos, así como también ocurre de manera similar con las diferentes salidas de la tarjeta, pues cuentan con tareas para la gestión de encendido y apagado así

como también para el control de cargas, ya sea por ángulo de fase o PWM.

Figura 4-15: ESP-IDF [39]



Sobre el firmware se desarrollan los siguientes temas:

Tareas: se ejecutan constantemente en el sistema operativo, realizando diferentes funciones para lectura, escritura y control.

GPIO: el ESP-WROOM-32 posee diferentes GPIO, los cuales se usan para leer o escribir señales digitales, en cuanto a los sensores se pueden enviar señales para iniciar su lectura o simplemente tener el pin en modo entrada y leerlo cada cierto intervalo de tiempo para generar los datos de lectura, o en modo salida para el control de los diferentes dispositivos que se han desarrollado en el hardware.

ADC,DAC: los ADC se usan para leer los datos de algunos sensores que proporcionan datos analógicos, por este motivo se hace la conversión de la señal analógica a un valor digital dentro de la tarjeta, para luego identificar el valor de la lectura del sensor. Los DAC se usan para realizar la operación contraria, teniendo valores digitales convertirlos a un valor analógico por ejemplo para generar audios o diferentes señales a partir del software.

Consola: para realizar diferentes pruebas directamente desde la tarjeta, se usa la opción de la consola, la cual se comunica por medio del puerto serie, para esto se crean las funciones y los comandos que estaran disponibles; Algunos comandos disponibles son *http*, para realizar las peticiones http manualmente y observar su respuesta, *pin* para realizar la prueba de un pin digital como entrada o salida, *help* para observar la lista de comandos, entre otros.

HTTP Request: las peticiones HTTP son indispensables en estas aplicaciones del campo IOT, por este motivo en el desarrollo del firmware se usan las librerias pertinentes para

realizar peticiones y además leer las respuestas de estas desde el servidor, ya que este es el medio de comunicación tarjeta-servidor.

Hora de Red: se obtiene la hora mediante el protocolo simple de tiempo de red (SNTP), este resulta de gran utilidad para la sincronización de los relojes de los sistemas informáticos. Se mantiene actualizada para realizar diferentes acciones respecto a esta.

Timers: la tarjeta posee dos grupos de timers, que cuenta cada uno con dos timers, un timer lo usa el sistema operativo, otro es configurado para realizar el control de potencia AC por ángulo de fase, para tener la sincronía necesaria con la señal de la red eléctrica.

I2C: el protocolo I2C se activa por medio de la instalación del driver en algún par de pines GPIO disponibles en la tarjeta. Se configura e inicia y posteriormente se crea una tarea la cuál se encarga de solicitar y leer los datos de los diferentes sensores conectados a este.

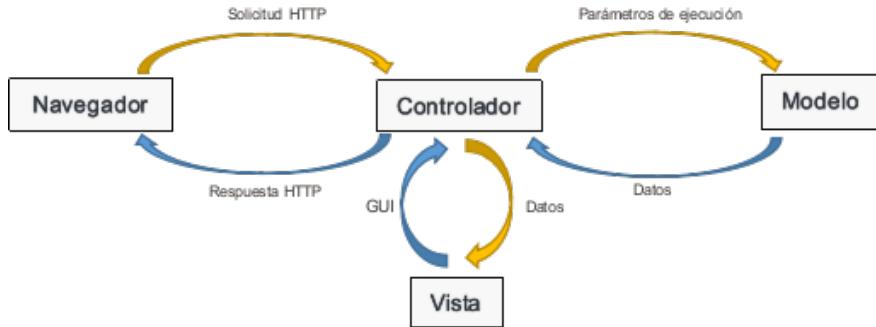
PWM: se ha mencionado anteriormente que para controlar las cargas DC se usa una salida PWM, el esp32 proporciona esta funcionalidad en algunos de sus pines, para su uso se configura y asignan los valores de funcionamiento.

Interrupciones: las interrupciones se usan para no gastar recursos en un monitoreo constante de las entradas, solo cuando existe un cambio de nivel en la entrada el dispositivo desencadena una serie de instrucciones relacionadas al tipo de interrupción y a diferentes funciones creadas para esta, la interrupción se usa por medio de los diferentes pines propuestos para esto en el hardware.

4.3. Software

En esta sección se desarrolla una aplicación web, la cual se encarga de hacer la gestión entre el usuario y la tarjeta. De este modo, se usa un patrón de arquitectura Modelo-Vista-Controlador (MVC). Este modelo es realmente útil ya que separa la lógica de negocio de la interfaz de usuario, incrementando la reutilización y flexibilidad, además la escalabilidad de ambos aspectos por separado, dicho esto la aplicación cuenta con diferentes modelos, controladores y vistas. La función de cada parte de esta arquitectura se puede observar en la figura 4-16.

Figura 4-16: Modelo-Vista-Controlador

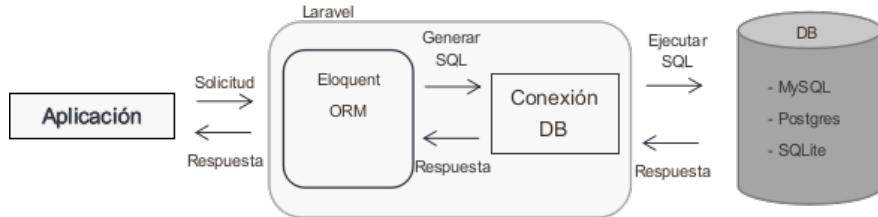


Su funcionamiento es el siguiente, primero el usuario realiza alguna acción en la interfaz (por ejemplo, presiona un botón, un enlace, etc), luego el controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega. Luego el controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza los datos del perfil del usuario) y después la interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Este patrón de diseño se usa en la programación orientada a objetos, por lo tanto se realiza la aplicación en el lenguaje de programación PHP, ya que es realmente útil para realizar la gestión de peticiones y envíos de formularios en dicha aplicación, además de que es importante también la gestión de las bases de datos de la aplicación, por este motivo se utiliza un framework basado en este lenguaje y esta arquitectura, el cual realiza diferentes trabajos en cuanto a la parte de la arquitectura. Para gestionar las diferentes partes de la aplicación, en este caso se usa el framework Laravel, el cual como se menciona anteriormente está orientado a facilitar las tareas comunes de la mayoría de proyectos web que utilizan HTML5 y PHP.

Además, con este framework se hace uso de un ORM (Mapeo Objeto-Relacional) llamado Eloquent. Esta es una forma de mapear los datos que se encuentran en la base de datos a objetos de PHP y viceversa, esto facilita el uso de diferentes gestores de bases de datos como MySQL, SQLite, entre otras, ya que todas las consultas están en PHP y el ORM ya se encarga del mapeo a los comandos SQL como se observa en la figura 4-17. Eloquent usa los modelos para enviar y recibir información de la base de datos.

Figura 4-17: ORM [Imagen Propia]



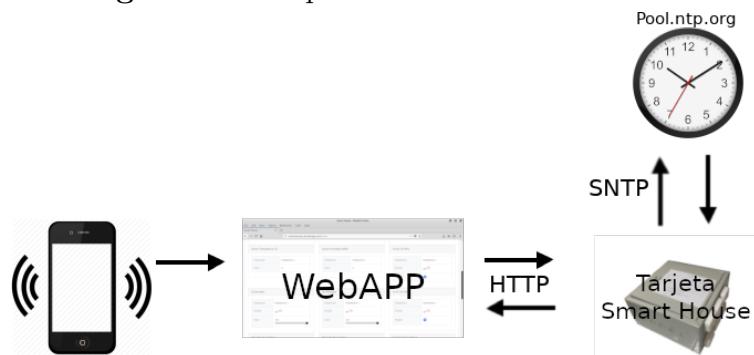
4.4. Prueba Beta

También llamada prueba de aceptación, la realiza el usuario final al prototipo listo para implementarse en el entorno

5 Resultados y Análisis

Para este capítulo se propone una habitación modelo, con la cual se describen ciertos pasos y funcionamientos de la solución en general. En la figura 5-1 esta el esquema de la solución IoT, para esto se supone una habitación con un sensor de temperatura, un ventilador y un bombillo led, las cuales el usuario va a visualizar y gestionar desde la aplicación web.

Figura 5-1: Esquema Solución SmartHouse



5.1. Software

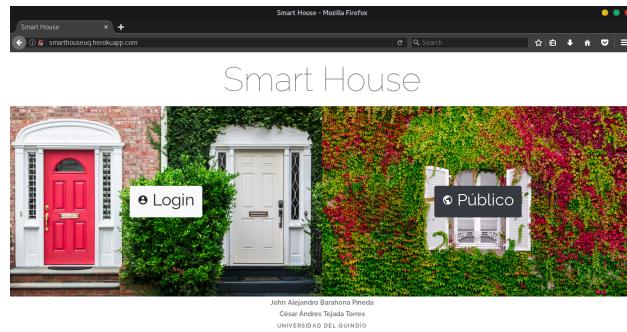
Se desarrolla la aplicación web de manera local y posteriormente se lanza a un servidor en Internet. Se encuentra compuesta por los siguientes sitios y las diferentes interacciones basadas en las funciones básicas, crear, leer, actualizar y borrar (CRUD).

- Parte Pública
- Parte Privada
 - API
 - Panel de Control
 - Crear
 - Ver
 - Editar
 - Eliminar

De acuerdo a la lista anterior, se toman en cuenta dos partes para esta, una pública y una privada, como se observa en la figura 5-2. En la parte pública se encuentra una vista con los datos de contacto, solicitudes de registro o productos y la cantidad de usuarios que actualmente están registrados en la aplicación. En la parte privada se encuentra la interacción de los usuarios sea administrador, dueño de una casa o de una habitación, para controlar y ver sus datos.

Las diferentes interacciones que tiene cada usuario en el panel de control se garantizan por medio del framework, creando diferentes roles para cada usuario que se está registrando y realizando la comprobación por parte de los controladores y el middleware que este provee.

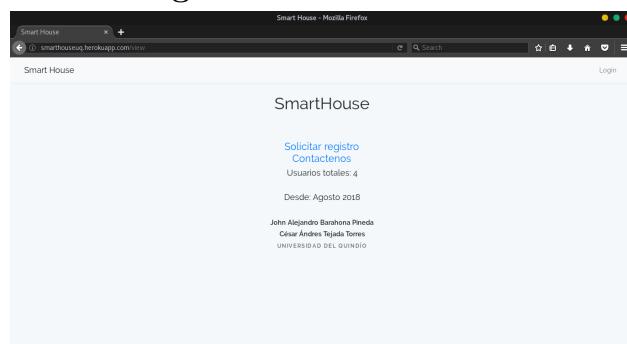
Figura 5-2: Página de Inicio



5.1.1. Parte Pública

En esta vista únicamente hay opciones para el contacto y solicitudes, como se menciona anteriormente, es una vista sencilla dada la poca información que contiene, como se observa en la figura 5-3.

Figura 5-3: Vista Pública



5.1.2. Parte Privada

En esta sección es donde se encuentra el Panel de Control para los diferentes usuarios de la aplicación. En primera instancia, para un usuario administrador, que es el encargado de gestionar la aplicación, este usuario tiene la posibilidad de crear, ver, editar y eliminar los diferentes registros de la aplicación, la vista de este usuario se puede observar en la figura 5.5(a). Por medio de este usuario es que se activan las cuentas de los demás, por esto en la parte pública estás las opciones de contacto y solicitud de registro.

Figura 5-4: Vistas de Usuarios

The screenshot shows the 'Smart House - Mozilla Firefox' browser window. The URL is smarthouseuq.herokuapp.com/home. The page title is 'Smart House - Panel de Control'. On the left, there is a sidebar titled 'Crear' with four options: '+ Usuario', '+ Casa', '+ Habitación', and '+ Dispositivo', each with a blue plus icon. The main content area is titled 'Dashboard' and has tabs for 'Usuarios', 'Casas', 'Habitaciones', and 'Dispositivos'. The 'Usuarios' tab is selected, showing a table with the following data:

Nombre	Apellido	Email	Rol	Acciones
Admin	Admin	admin@smarthouse.com	admin	
User	House	userhouse@smarthouse.com	userHouse	
User	Room2 H	userroom2h@smarthouse.com	userRoom	
User	Room	userroom@smarthouse.com	userRoom	
User	Room1 H	userroom1h@smarthouse.com	userRoom	

At the bottom of the page, there is a footer with the names 'John Alejandro Barahona Pineda' and 'César Andrés Tejada Torres'.

(a) Usuario Administrador

This screenshot shows the same application interface as above, but for a user with the role 'User'. The table in the 'Usuarios' section shows two entries:

Nombre	Apellido	Email	Rol	Acciones
User	Room2 H	userroom2h@smarthouse.com	userRoom	
User	Room1 H	userroom1h@smarthouse.com	userRoom	

Below the table, there is a footer with the names 'John Alejandro Barahona Pineda' and 'César Andrés Tejada Torres'.

(b) Usuario de Casa

This screenshot shows the same application interface for a user with the role 'User'. The table in the 'Usuarios' section shows two entries:

Nombre	Apellido	Email	Rol	Acciones
User	Room2 H	userroom2h@smarthouse.com	userRoom	
User	Room1 H	userroom1h@smarthouse.com	userRoom	

Below the table, there are several sections for managing rooms and devices, including 'Cargar Habitación', 'Cargar Dispositivo', 'Cargar Control', and 'AC Control'. Each section contains fields for 'Habitación', 'Estado', 'Valor', and 'Ajustes'.

(c) Usuario de Habitación

También existe el usuario dueño de la casa donde se encuentra el dispositivo, este usuario es opcional y es para gestionar los dispositivos presentes dentro de una misma casa, es un administrador de la casa, el cual puede ver y editar algunos campos de sus usuarios hijos

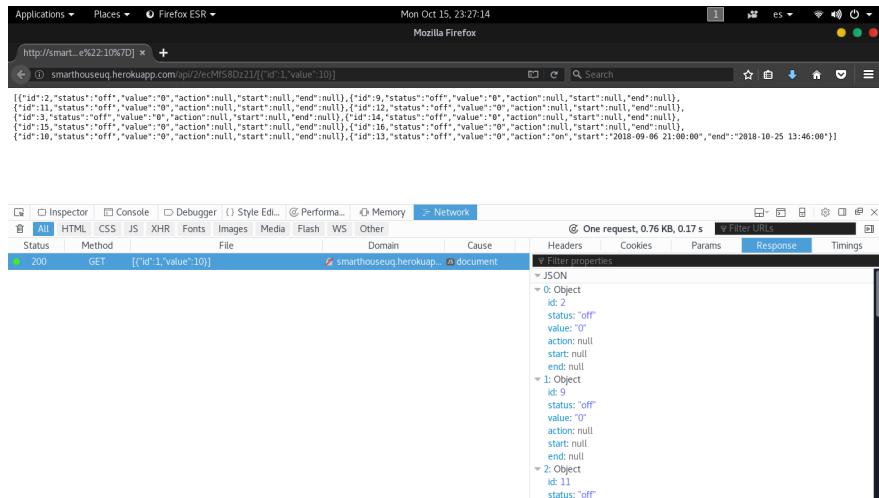
o usuarios habitación y sus diferentes casas y habitaciones, únicamente las que estén registradas a su nombre, como se ve en la figura 5.5(b), de este modo el rol de este usuario es administrar su casa y visualizar los datos de esta.

Por último, otro rol es el de usuario habitación, el cuál es un usuario que solo visualiza sus propios datos, como la habitación y los dispositivos presentes en esta, como se observa en la figura 5.5(c), a este solo le compete la información de lo que posee en su habitación, por tal motivo el panel de control muestra una vista general de los datos y el estado de sus dispositivos, además de tener la capacidad de editar partes básicas de su habitación y perfil. Este usuario puede o no estar sujeto a un usuario padre o usuario casa, ya que, solamente puede poseer una tarjeta para su habitación y ninguna otra en dicha casa.

Continuando con este usuario y la habitación modelo propuesta al inicio del capítulo, luego de que el usuario accede a la aplicación web e inicia sesión con los datos que ha registrado en el sistema, al ser un usuario de una habitación, este se encuentra con un panel de control como el de la figura 5.5(b), allí puede gestionar los dispositivos presentes en su habitación, así pues, puede visualizar la temperatura que se ha sensado y también encender o apagar los dispositivos conectados a la tarjeta.

Además de este panel de control, desde el cuál se realizan las operaciones sobre la aplicación, en la parte privada se encuentra la ruta encargada de la actualización Servidor-Tarjeta, es decir, en esta ruta es donde se da la comunicación. En esta ruta se realiza una petición HTTP tipo GET por parte de la tarjeta, esta contiene el id de la habitación en la cuál está instalada la tarjeta y también el token correspondiente a esta, y además en la URL se añade un texto tipo JSON en el cual se encuentra toda la información de la lectura actual de los sensores, esta petición la responde el servidor con un texto, también tipo JSON que contiene la información pertinente de las cargas o actuadores, como se observa la figura 5-5, para dar seguridad a esta transacción, se utiliza el token mencionado anteriormente, el cual se verifica mediante el id de la habitación y que este coincida con los datos almacenados, de este modo se garantiza que lo que se envía este sometido a verificación.

Figura 5-5: Página de intercambio de datos

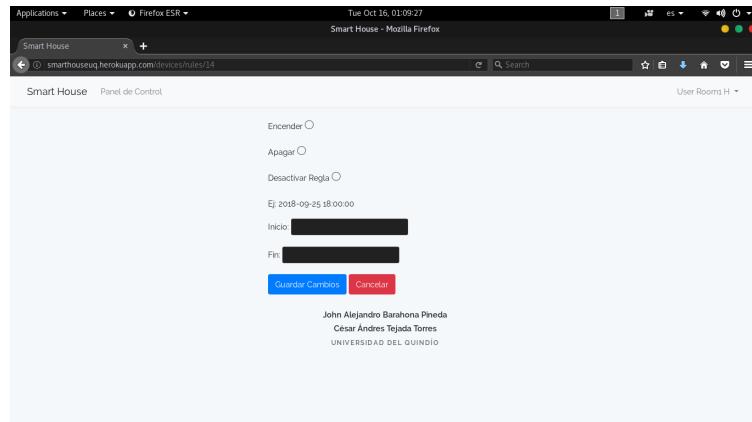


De este modo, el usuario interactuando con la aplicación genera modificaciones en el texto con que responde la aplicación web a la petición de la tarjeta. Si el usuario desea encender el ventilador, como se observa en la figura 5.5(b) esta presente un botón en la información del dispositivo, el cual con presionarlo lo enciende o apaga, esto es valido para cualquiera de los dos dispositivos, sea el ventilador o el bombillo led.

Si el ventilador esta conectado a una salida de AC controlada es posible que por medio del deslizador se le asigne un valor para que cambie su funcionamiento, del mismo modo para el bombillo led donde se refleja en su cambio de intensidad, pero este conectado a una salida DC controlada. Al generar estas interacciones el texto en formato JSON cambia de acuerdo a lo pedido por el usuario.

También si el usuario desea añadir, modificar o eliminar una regla, por ejemplo, desea encender el bombillo led a una hora deseada, esto lo puede lograr mediante el botón de reglas en el panel de control, el cual lo redirige a la vista que se observa en la figura 5-6, en esta se indica una hora de inicio y finalización en la cual el bombillo led enciende a la hora de inicio y se apaga a la hora de fin, si es la regla de apagado solo necesita una hora de inicio para apagar el bombillo.

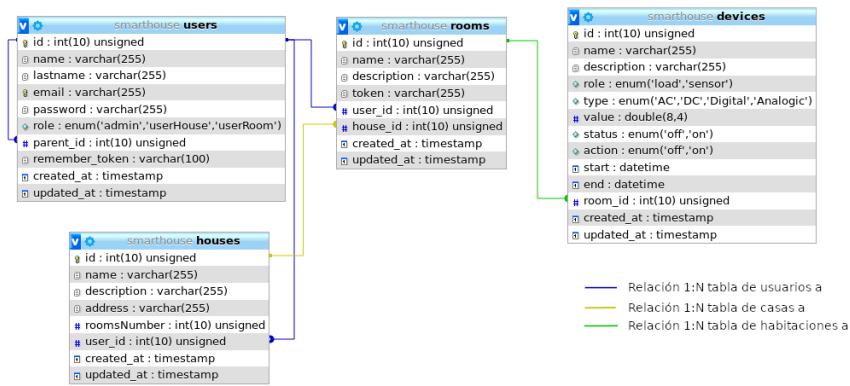
Figura 5-6: Vista para añadir reglas



5.1.3. Base de Datos

La estructura de la base de datos se puede observar en la figura 5-7, aquí se observan los diferentes campos que posee cada tabla, además de las llaves y sus relaciones, las relaciones presentes en esta estructura son de tipo 1:N, es decir, por ejemplo un usuario puede tener relacionadas N casas.

Figura 5-7: Base de datos SmartHouse



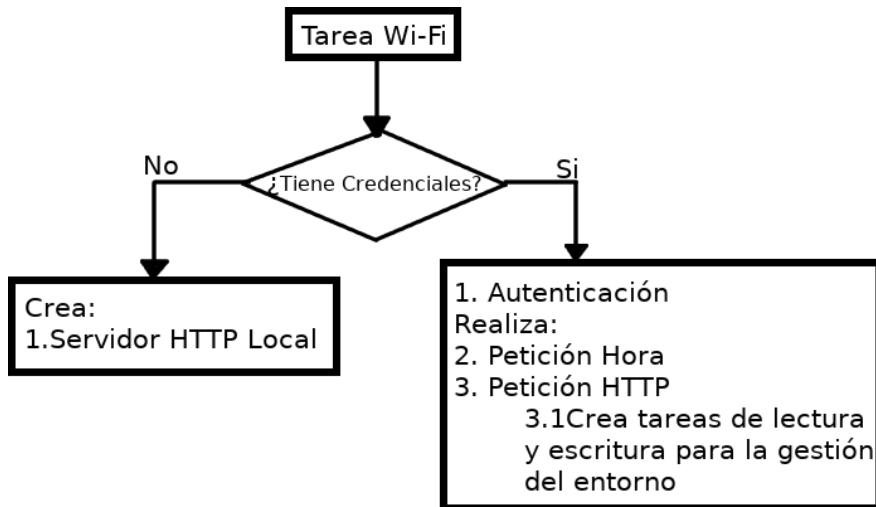
5.2. Firmware

El firmware se encuentra compuesto, como se ha mencionado anteriormente, de tareas, en la figura 5-8 se observa un bosquejo de como funcionan las diferentes tareas de las que se compone este, tomando como función principal la encargada de gestionar la conexión a Wi-Fi y almacenar sus credenciales, dependiendo del estado de si encuentra o no estas, el sistema se comporta de una u otra forma. Si existen credenciales almacenadas en la tarjeta, el sistema se trata de conectar, si la conexión es exitosa comienza el proceso de actualización

de la hora del sistema y también de las diferentes ordenes de la aplicación web. En cambio si no existen credenciales el dispositivo inicia un servidor web local para que el usuario pueda proporcionarle estas credenciales como se menciona en la sección 5.2.1.

En cada caso se crean tareas diferentes, para el caso de no existir credenciales únicamente se configura la tarea del servidor http local, y para el otro caso se lanzan todas la demás tareas encargadas de la escritura y lectura de datos, como se menciona en las siguientes secciones. Cabe resaltar que la tarjeta tiene disponible en promedio 160KB de memoria heap para realizar otras operaciones o implementar más funcionalidades en esta.

Figura 5-8: Esquema de Tareas [Imagen Propia]



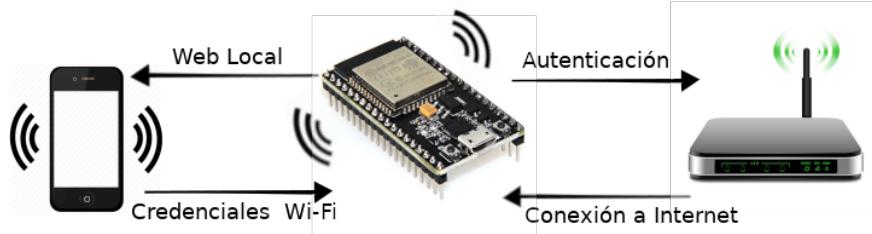
Además de esto, se mide el promedio del tiempo en que se demora la ejecución de la tarea, en dos casos, cuando realiza la primera petición después de conectado a la red, es decir, cuánto se demora realizando las peticiones necesarias, como obtener el tiempo de la red y realizar la petición a la aplicación web, obteniendo un tiempo promedio de aproximadamente de 2.7s, este tiempo depende de la disponibilidad de los diferentes servidores en la web además de la velocidad y el tráfico de la red, ya que la tarjeta espera hasta que obtiene la hora y luego continua con la petición HTTP también esperando que el servidor responda. El otro caso es el tiempo que se demora la tarea en leer los datos de los sensores, enviar la petición HTTP a la aplicación web, recibirlos y enviarlos a los diferentes actuadores, este tiempo es de 1 s en promedio con un total de 3056 muestras.

5.2.1. Conexión a Internet vía Wi-Fi

Los sistemas IoT deben estar conectados siempre a Internet, por este motivo se debe brindar una forma para conectar al sistema a este, por lo tanto, se desarrolla un servidor local en la tarjeta que se encarga de esto, como se ha mencionado el módulo del esp32 funciona como

Punto de Acceso (AP) y como Cliente o Estación (STA) al mismo tiempo, aprovechando esta capacidad se usa el servidor local y esta encargado de gestionar la conexión de la tarjeta vía Wi-Fi como se observa en la figura 5-9.

Figura 5-9: Conexión a Internet vía Wi-Fi ESP32



De este modo, en la figura 5-10 están algunas páginas del servidor local de la tarjeta, en la figura 5.11(a) se puede ver la lista de las diferentes redes al alcance de la tarjeta, basta con seleccionar una red e ingresar sus credenciales para conectarse a esta, en la figura 5.11(b) se observan los detalles de la conexión actual y también la opción de desconectarse de esta. Su funcionamiento es muy intuitivo, se selecciona la red a la que se desea conectar la tarjeta, se ingresan sus credenciales y posteriormente el dispositivo verifica si la conexión fue exitosa o no, si la conexión es exitosa ya la tarjeta está lista para su funcionamiento, se debe reiniciar para que solo quede funcionando como STA y no en el modo dual, además de esto si las credenciales de la red cambian también se incluye un botón para el borrado de estas, para que se pueda configurar de nuevo la conexión a la red Wi-Fi.

Figura 5-10: Aplicación Conexión a Wi-Fi

(a) Lista de Redes

(b) Datos de Conexión

5.2.2. Escritura de Datos en la Aplicación Web

Los datos que esta leyendo la tarjeta provienen de los diferentes sensores que tiene conectados como se ha mencionado, se usan diferentes tipos, como de estado para sensar la presencia, de calidad de aire entre otros presentes en esta. Para la escritura de los datos, en el firmware, se desarrollan diferentes tareas encargadas de leer y enviar estos a una tarea central. Los datos que están enviando contienen el id del dispositivo y la medida que lee en ese momento, estos se envían en forma de texto en formato JSON, de este modo, la tarea central los gestiona y envía a la aplicación con el mismo formato, organizandolos en la petición HTTP tipo GET que realiza, así la url que la tarjeta solicita, incluyendo el JSON de cada sensor, se observa en la figura 5-11, de este modo, en la url se encuentra el dominio del servidor, y la dirección que contiene el id y token de la habitación, por último esta la información del sensor en formato JSON, que se compone por el id y el valor de este. La aplicación ya se encarga de almacenarlos y mostrarlos al usuario como se menciona anteriormente.

Figura 5-11: URL de la petición HTTP

smarthouseuq.herokuapp.com/api/2/ecMfS8Dz21/["id":1,"value":10}]

5.2.3. Lectura de Datos de Internet

Para la configuración y comparación de las reglas que suministra el usuario a el dispositivo que dese controlar es necesario contar con la hora actual y que se siga actualizando localmente gracias al RTC que posee internamente el esp32, así, al inicio de la aplicación, se sincroniza y almacena la hora actual de la red, por medio del protocolo SNTP. Estas reglas actúan en cuanto a que encienda o apague un dispositivo a una hora dada, después de obtener este dato la aplicación continua con normalidad para realizar las diferentes peticiones a la aplicación web.

La interacción del usuario se da con la aplicación web mencionada anteriormente, de este modo la tarjeta siempre se debe actualizar, para esto, cuando la tarjeta envía los datos de los sensores la aplicación responde con datos de cabecera HTTP y además la información de los dispositivos que controla la tarjeta, esta los recibe en una cadena texto en formato JSON como se observa en la figura 5-12, los procesa y envía a las tareas pertinentes ya sea para encender o apagar algún dispositivo conectado a la tarjeta, además envía las reglas que el usuario ha definido.

Figura 5-12: Respues del la APP Web a la Tarjeta

HTTP/1.1 200 OK	Correcciones generales	3 hours ago
Connection: close		
Date: Tue, 30 Oct 2018 23:33:24 GMT		
Server: Apache		
Cache-Control: no-cache, private=*.tex	agrego Imagenes, corrección de redacción y nuevo texto	14 days ago
Set-Cookie: XSRF-TOKEN=eYjpd161n1HViCXC5UK0xbldyVfwPwamIyXgBPFt1lCJ2yWx1Z516lKj60lJ; MENEWPj1Uu29yMDY20zFY0W93bE1ES35JMTHnb1ldhLbg04dEpLVjRClwvbVwvzbChpEe53zC0nuVL0GpDaDByWEfRnfY0zUWp1Ntaigj30WEwN2RmNm10MjdlNzM2D0A0Mw11MwE30YV2NDQ4YmM3Zdk2YzBkyWjMwY3ZfLZGFLMu0YzI3NT5NUt103%3D3D; expires=Wed, 31-10-2018 01:33:24 GMT; Max-Age=7200; path=/		
Set-Cookie: session=eyJpdiI6InRwZEo0ldx1pUrR0RTY3hsjO9EFP5IsInZhbwLlj0ui1FkM09zylwag0D23pUkJwaErZYyLa3FGT0x6TmDzLz1xJnsLwzGwxfJ1YJUw0UR0SzuNsFwWmmXtG0Z3Iem53Z2h6R3fTxYezTmUk0R0T91lwBf1oiJmzAzYTMyW0YbYzJtkzJxU2NDNmYZM2U0NmVlNjY4ZTawXyMjd1nlDtC2yM3YzU0M0DzKznkjNC9; expires=Wed, 31-Oct-2018 01:33:24 GMT; Max-Age=7200; path=/; httponly	new repository	9 months ago
Content-Type: application/json; charset=utf-8	Via: 1.1 vegur	
	EstadoAnte.tex	new repository
		9 months ago

5.3. Hardware

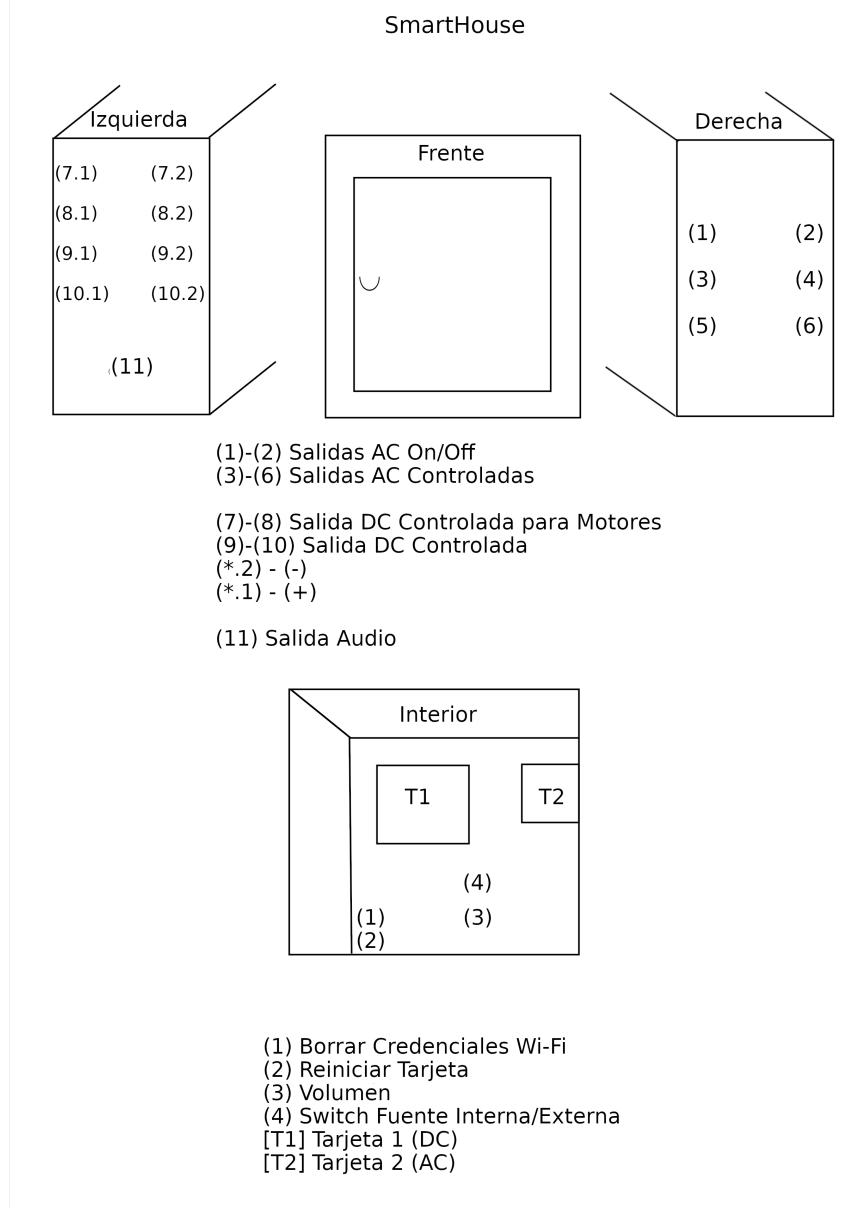
De acuerdo a los circuitos diseñados en la sección 4.1 donde se propone el desarrollo de hardware de la solución IoT, en la figura XXXX se observan las diferentes tarjetas ya ensambladas en una caja eléctrica para probar el funcionamiento del prototipo. Las salidas y entradas están distribuidas por la caja eléctrica como se observa en la figura XXXX de acuerdo a lo propuesto, para las salidas AC se usan toma corrientes para conectar allí los diferentes dispositivos, para las salidas DC se utilizan conectores hembra tipo banana para facilitar la conexión de estos dispositivos.

La distribución de las diferentes salidas que posee la tarjeta se organiza en pegatinas y se colocaron sobre la caja eléctrica, la figura 5-13 muestra esta información, tanto para la parte interna como externa de la caja.

De acuerdo a lo mencionado anteriormente, el sistema se ha probado con cargas AC como bombillos LED entre 7W y 20W, de filamento de 100W, probando funcionalidades como el control de potencia AC por ángulo de fase, obteniendo los resultados esperados, como se observa en las figuras XXX, el voltaje de alimentación viene dado por la red eléctrica, la tarjeta simplemente conmuta el estado de la alimentación o controla la potencia entregada. Para las salidas DC se realizan pruebas con un motor DC de X W, además de una tira LED de X W, la cual se le varia la energía entrega, como se observa en las figuras XXX, estas cargas se alimentan con 12VDC directamente desde la fuente o convertidor AC-DC, los circuitos que se implementan simplemente conmutan el estado de encendido/apagado o por medio de PWM variar la energía entregada.

5.4. Prueba Beta Cerrada

Para la prueba beta se escoge un grupo de personas, las cuales interactúan directamente con la aplicación web y el prototipo de la tarjeta SmartHouse, se detallan diferentes ítems para evaluar, como el ingreso a la aplicación y visualización de los datos almacenados en esta.

Figura 5-13: Descripción caja eléctrica tarjeta SmartHouse

6 Conclusiones

■

7 Trabajos Futuros

Se propone añadir más funcionalidades, en cuanto al ahorro de energía, manejo de otros dispositivos por medio de infrarrojo, actualizaciones remotas, una aplicación web local y agregar diferentes funcionalidades de administración y control parental a los usuarios dueños de la casa que poseen mas de un dispositivo en cada habitación. Además se pueden implementar algunos circuitos usando componentes de montaje superficial para reducir su tamaño.

- El ahorro de energía permite mejorar el consumo del sistema en cuanto a la gestión de los datos, generar un modo “stand by” que lo activa, desactiva o programa el usuario.
- Implementar la funcionalidad por firmware y hardware para la memorización de mandos infrarrojos y su emisión para controlar diferentes dispositivos por este medio.
- Activar las actualizaciones remotas o sobre el aire, las cuales permiten al fabricante tener actualizado o realizar diferentes modificaciones al firmware sin la necesidad de estar conectado directamente en la tarjeta.
- La aplicación web local actúa como espejo de la que se encuentra en Internet. Sus funcionalidades son en caso de mantenimiento de la aplicación principal o de cualquier problema relativo a la conexión a Internet, esta daría soporte local hasta que entre de nuevo en funcionamiento la que se encuentra en la nube.

Glosario

AC (Corriente alterna): corriente eléctrica variable en la que las cargas eléctricas (electrones) cambian el sentido del movimiento a través de un conductor de manera periódica.

DC (Corriente continua): corriente de intensidad constante en la que el movimiento de las cargas eléctricas (electrones) siempre es en el mismo sentido.

Internet del todo (IoE): es un concepto que extiende el énfasis de la internet de las cosas (IoT) en las comunicaciones de máquina a máquina para describir un sistema más complejo que también abarca personas y procesos.[47]

Internet de las Cosas (IoT): parte fundamental del internet del todo (IdT), el cual se refiere principalmente a la interacción máquina-máquina, en incluso interacción máquina-persona.

Software: conjunto de programas y rutinas que permiten a un sistema realizar determinadas tareas.

Hardware: partes físicas que componen un sistema electrónico, como por ejemplo los componentes de un circuito electrónico.

Firmware: programa informático que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo, es decir, software que maneja físicamente al hardware.

Infrarrojo: se refiere a la radiación electromagnética con longitud de onda mayor (menor energía) a la de la luz visible por el ser humano.

HTML5: siendo la última versión de HTML, contiene elementos, atributos y comportamientos nuevos, ademas de un conjunto más amplio de tecnologías que proporciona mayor diversidad y alcance a los sitios Web.

PHP (Preprocesador de hipertexto): es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

SQL (lenguaje de consulta estructurado): es un lenguaje de programación estándar e interactivo para la obtención de información desde una base de datos y para actualizarla.

Framework: Tarjeta de Adquisición: **API:** Tarea: **GPIO:**

Bibliografía

- [1] Mark D. Gross. *Smart House And Home Automation Technologies*. 1998.
- [2] Basil Hamed. Design & implementation of smart house control using labview. 2012.
- [3] N. Datta, T. Masud, R. Arefm, A. A. Rimon, M. S. Rahman, and B. B. Pathik. Designing and implementation of an application based electrical circuit for smart home application. 2014.
- [4] Miroslav Behan and Ondrej Krejcar. Vision of smart home point solution as sustainable intelligent house concept. 2013.
- [5] César Cheque, Felipe Baeza, Gastón Márquez, and Juan Calderón. Towards to responsive web services for smart home led control with raspberry pi. a first approach. 2015.
- [6] Mahdi Kasmi, Faouzi Bahloul, and Haykel Tkitek. Smart home based on internet of things and cloud computing. 2016.
- [7] Samuel Tang, Vineetha Kalavally, Kok Yew Ng, and Jussi Parkkinen. Development of a prototype smart home intelligent lighting control architecture using sensors onboard a mobile computing system. 2017.
- [8] Masayuki Kaneko, Kazuki Arima, Takashi Murakami, Masao Isshiki, and Hiroshi Sugimura. Design and implementation of interactive control system for smart houses. 2017.
- [9] Ashwini Deshmukh and K.B.Khanchandani. Designing and implementation of an application based electrical circuit for smart home application. 2016.
- [10] Aadel Howedi and Ali Jwaid. Design and implementation prototype of a smart house system at low cost and multi-functional. 2016.
- [11] Himanshu Verma, Madhu Jain, KhushhaliGoel, Aditya Vikram, and Gaurav Verma. Smart home system based on internet if things. 2016.
- [12] Medilla Kusriyanto and Bambang Dwi Putra. Smart home using local area network (lan) based arduino mega 2560. 2016.

- [13] Medilla Kusriyanto and Beny Setiawan. Android smart home system based on atmega16. 2015.
- [14] Tomas Sysala, Martin Pospichal, and Petr Neumann. Monitoring and control system for a smart family house controlled via programmable controller. 2016.
- [15] Shigeru Owada and Fumiaki Tokuhisa. Kadecot: Html5-based visual novels development system for smart homes. 2012.
- [16] Shih-Pang Tseng, Bo-Rong Li, and Jun-Long Pan amd Chia-Ju Lin. An application of internet of things with motion sensing on smart house. 2014.
- [17] TechTarget. Internet de las cosas (iot).
- [18] Kevin Ashton. That 'internet of things' thing. 2009.
- [19] Heroku Inc. The heroku platform, 2018.
- [20] Laravel. Laravel philosophy, 2013.
- [21] MDN web docs Mozilla. Http, 2018.
- [22] MDN web docs mozilla. Http request methods.
- [23] JSON.org. Introducción a json.
- [24] Damián Pérez Valdés. ¿qué son las bases de datos?, 2007.
- [25] Espressif Systems. Esp-wroom-32 datasheet, 2018.
- [26] Electrodragon. Esp-wroom-32.
- [27] Greenfacts. Corriente alterna y corriente continua.
- [28] CEKIT. *Curso práctico de Electrónica Industrial y Automatización*. 2001.
- [29] Enrique Gómez. Qué es pwm y para qué sirve, 2017.
- [30] HETPRO. I2c – puerto, introducción, trama y protocolo.
- [31] AMGkits. Sensor de intensidad óptica (sensor de luz) gy-30 bh1750fvi.
- [32] ElectroniLab. Sensor de temperatura y humedad dht11.
- [33] Naylamp Mechatronics. Tutorial sensores de gas mq2, mq3, mq7 y mq135.
- [34] Geekbot Electronics. Mq-135 módulo sensor de calidad del aire.
- [35] Antony Garcia Gonzales. Módulo yl-83: Un detector de lluvia, 2014.

- [36] Naylamp Mechatronics. Módulo pir hc-sr501.
- [37] Punto Flotante S.A. Pruebas al módulo sensor pasivo infrarrojo pir hc-sr501. sensor de movimiento (body motion).
- [38] Andrew S. Tanenbaum. *Sistemas Operativos Modernos*. 3ra edition, 2009.
- [39] Espressif Systems. Esp-idf programming guide.
- [40] Labcenter Electronics Ltd. Proteus design suite.
- [41] Geeky Theory. ¿qué es proteus?
- [42] Indiamart. Sunrobotics dc dc step up boost power module.
- [43] Picclick. Bta26-600b triac 600v 25a.
- [44] platea pntic. Disparadores de schmitt.
- [45] International Rectifier. Irlz44n.
- [46] Texas Instrument. Lm386 low voltage audio power amplifier.
- [47] Margaret Rouse. Internet de todo (ioe), 2017.

Anexos