ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO DE CONTROL INTELIGENTE DE ILUMINACIÓN RESIDENCIAL UTILIZANDO ESP32

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

DANIEL SEBASTIAN CELA TOAQUIZA

daniel.cela@epn.edu.ec

DIRECTOR: ING. CARLOS ANDRÉS YUNGA SÁNCHEZ

carlos.yunga@epn.edu.ec

DMQ, Febrero 2024

CERTIFICACIONES

Yo, Daniel Sebastian Cela Toaquiza declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Daniel Sebastian Cela Toaquiza

daniel.cela@epn.edu.ec

Certifico que el presente trabajo de integración curricular fue desarrollado por Daniel Sebastian Cela Toaquiza, bajo mi supervisión.

Carlos Andrés Yunga Sánchez
DIRECTOR

carlos.yunga@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

DANIEL SEBASTIAN CELA TOAQUIZA

DEDICATORIA

El presente proyecto de integración curricular es dedicado para mi familia, debido a que han aportado tanto su apoyo incondicional y ánimos como sus conocimientos, además de ello, quiero dedicarle este pequeño logro al ciclismo ya que fue mi manera de afrontar diversos momentos difíciles que se han presentado dentro como fuera de la realización del documento.

DANIEL SEBASTIAN CELA TOAQUIZA

AGRADECIMIENTO

Agradezco a las personas que me apoyaron y dieron ese aliento para lograr realizar este paso inmenso dentro de mi vida, por ello quiero agradecer a mis padres, hermanos, e ingenieros, a las personas mencionadas anteriormente solo puedo decirles gracias por todo.

A la institución que me forjó, la Escuela Politécnica Nacional, a cada ingeniero involucrado en mi proceso educativo dentro de la solemne institución mencionada, que me ayudaron a obtener el título en la honorable ESFOT, a la que pertenezco y mantengo un cariño especial.

Para finalizar, quiero agradecer al ciclismo ya que fue mi manera de afrontar diversos momentos difíciles que se han presentado dentro como fuera de la realización de este documento.

Daniel Sebastian Cela Toaquiza

ÍNDICE DE CONTENIDOS

RESUMEN

El desarrollo de la tecnología ha sido marcado por el uso de artefactos inteligentes dentro de actividades cotidianas e industriales, tales como: encender luces, cerrar cortinas, cerrar puertas, abrir ventanas, encendido remoto, entre otros. Es allí donde el presente proyecto es aplicable, su objetivo principal es la ejecución de un control remoto inteligente de luces, implementando una menor inversión para obtener un resultado óptimo para su despliegue.

El presente proyecto contará con la libertad tanto de la modificación individual del estado de las luces como también la ejecución del encendido automático. El proyecto por exponerse hará uso de un ESP32, el cual tiene la capacidad de elaborar proyectos complejos formados por comunicaciones tanto *Wi-Fi* como *Bluetooth*, sus diversas acciones, se implementan mediante líneas de código que hacen uso de librerías, ofreciendo una estructura al proyecto. Obteniendo así, una página web o una aplicación accesible mediante la red que permitirá la modificación del estado de las luces.

El trabajo de titulación se dividirá en varias secciones, tales como: en el desarrollo del documento se encontrará el enfoque, las restricciones y las definiciones necesarias para su realización. Posteriormente, se especificará la metodología que guiará la documentación del prototipo, al igual que la elaboración del prototipo, la selección de las librerías imprescindibles, las aplicaciones adicionales para el acceso a la red, entre otras. Finalmente, se llevará a cabo un análisis exhaustivo del prototipo, con las conclusiones y recomendaciones respectivas generadas por el presente trabajo.

PALABRAS CLAVE: Residencias, Automatización, prototipo, control inteligente de luces, programar, acceso a la red.

ABSTRACT

The development of technology has been marked by the use of intelligent devices within daily and industrial activities, such as: turning on lights, closing curtains, closing doors, opening windows, remote ignition, among others. This is where this project is applicable, its main objective is the execution of an intelligent remote control of lights, implementing a lower investment to obtain an optimal result for its deployment.

This project will have the freedom of both individual modification of the state of the lights and also the execution of automatic ignition. The project to be exposed will use an ESP32, which has the capacity to develop complex projects made up of both Wi-Fi and Bluetooth communications, its various actions are implemented through lines of code that make use of a storage of functions known as libraries, offering a structure to the project. Thus, obtaining a web page or an application accessible through the red that will allow the modification of the state of the lights.

The degree work will be divided into several sections, such as: in the development of the document, you will find the approach, restrictions and definitions necessary for its completion. Subsequently, the methodology that will guide the documentation of the prototype will be specified, as well as the development of the prototype, the selection of essential libraries, additional applications for the red access, among others. Finally, an exhaustive analysis of the prototype will be carried out, with the respective conclusions and recommendations generated by this work.

KEYWORDS: *Residences, Automation, prototype, intelligent light control, programming,* the red *access.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El presente trabajo de titulación se enfocará en la elaboración de un prototipo de control inteligente de iluminación residencial haciendo uso de un microcontrolador ESP32. Tiene como objetivo manipular el estado de la luminaria dentro de una vivienda (que en este caso estará representada por una maqueta), de forma remota, mediante la utilización de una aplicación o una página web.

1.1 Objetivo general

Implementar un prototipo de control inteligente de iluminación residencial utilizando la tecnología ESP32.

1.2 Objetivos específicos

- Investigar los requerimientos para el diseño del prototipo.
- Seleccionar el *hardware* y *software* acorde a los requerimientos establecidos.
- Diseñar el prototipo de control inteligente de iluminación.
- Implementar el prototipo en una maqueta.
- Realizar pruebas de funcionamiento del prototipo

1.3 Alcance

El presente proyecto tiene como objetivo desarrollar un prototipo funcional que tendrá la capacidad de controlar las luminarias residenciales haciendo uso de un microcontrolador ESP32. Algunas de estas capacidades especificadas son las siguientes:

- Controlar el encendido y apagado de las iluminarias de la residencia de forma remota.
- Programar la hora de encendido y apagado de las iluminarias de la residencia.
- Automatizar el encendido o apagado de las iluminarias de la residencia.

1.4 Marco Teórico

IOT

Con la evolución constante de la tecnología ha permitido que objetos inteligentes sin restricción de distancia puedan interactuar entre sí, a la tecnología anterior se le conoce como IOT, la cual busca unir artefactos capaces de monitorear ambientes y cambiar su estado si lo requiere conectándose a una red. Para posteriormente, procesarlos, almacenarlos y analizarlos. Algunos de los ejemplos de su aplicación son: manipulación

de la navegación de automóviles inteligentes, obtención de datos de sensores, manipulación de maquinaria y luminaria remota, entre otros [1].

IOT brinda una amplia disponibilidad hacía varias plataformas que facilitan el procesamiento y visualización de los valores obtenidos de los dispositivos inteligentes mediante la red, algunas de las aplicaciones que logran dar una interfaz gráfica son: *Blynk*, *Arduino Cloud*, servidores web, entre otras [1].

Ngrok

La realización de servidores de forma local para uso propio se ha popularizado en la actualidad. Sin embargo, cuentan con la imposibilidad de acceder hacia una red externa. Por lo tanto, se requieren de plataformas que logren solventar el problema de conexión. Ngrok permite obtener conexiones seguras a proyectos que se ejecuten de forma local utilizando el concepto inverso de lo que se conoce como proxy, el proceso realizado no afecta la estructura previamente instalada, lo cual es un gran beneficio, además, es compatible con varios tipos de sistemas operativos, algunas de las estructuras compatibles son: AWS, servidores locales, Raspberry Pi, Azure, entre otras [2].

Además, Ngrok permite realizar varios tipos de acciones diferentes, tales como: el empaquetamiento de información de diferentes tipos, para ser enviados mediante la red del usuario sin generar modificaciones, ejecución de aplicaciones web de prueba dentro de un ambiente controlado, realización de una puerta trasera para cualquier aplicación móvil, gestión de servicios desde la nube, impulsar módulos HTTP, control remoto utilizando protocolos seguros como son SSH Y RDP [2].

Microcontrolador

Un microcontrolador es un dispositivo electrónico indispensable dentro de diversas tecnologías para la automatización de acciones complejas de alta prioridad, su utilización se ha visto incrementa debido a sus capacidades y características, una de esas es el uso de memoria RAM y ROM, como se puede ver en la **Figura 1.1.** Estructura de un microcontrolador, los cuales son los encargados de almacenar la información tanto temporalmente como por un tiempo prolongado para posteriormente ser ejecutadas [3] [4].

Su principal objetivo es ejecutar los códigos configurados deseados por el usuario teniendo la posibilidad de aceptar cambios a su estructura indefinidamente sin la utilización de tecnología de alto coste [3] [4].

Las órdenes que se insertan dentro de los microcontroladores utilizan lenguajes de programación comunes como es el lenguaje C, no obstante, aquel lenguaje no puede ser comprendido y procesado por el microcontrolador. Por consiguiente, se hace uso del compilador que la mayoría de los programas especializados contienen, el compilador es el encargado de modificar el lenguaje original en un lenguaje apto para su procesamiento conocido como lenguaje de máquina [3] [4].

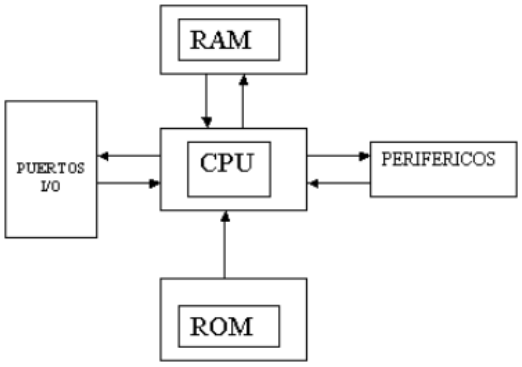

Figura 1.1. *Estructura de un microcontrolador*

La evolución de los microcontroladores ha generado que varias empresas se especialicen en su ejecución generando así un nicho de mercado variado, algunas de las empresas que desarrollan los microcontroladores son: Motorola, Toshiba, *Texas Instruments*, entre otros. Sin embargo, no varían en su propósito, el cual se enfoca en el cumplimiento de órdenes del usuario. Dentro de la variedad de microcontroladores especializados para la realización de proyectos, se pueden encontrar: Arduino Mega, ESP32, Rasberry, entre otros [4].

Esp32

El interés de las empresas hacia el desarrollo de la interconexión entre artefactos inteligentes ha impulsado la creación de módulos especializados accesibles orientados hacia IOT, el *Wi-Fi* y el *Bluetooth* han sido un pilar fundamental para que estos artefactos inteligentes obtengan una comunicación de fácil acceso entre sí, estos medios de comunicación desencadenan un amplio rango de artefactos, tales como el ESP32 [5].

El módulo ESP32 introducido dentro del mercado por empresas especializadas como es *espressif systems* se ven implementados dentro de varios sistemas de control, como son: transmisión de música, redes de sensores, controles inteligentes, entre otros. Su variedad de aplicaciones genera varios grupos de microcontroladores con características desiguales, por ejemplo: *Socs*, H2 *Series*, C6 *Series*, C3 *Series*, C2 *Series*, S3 *Series*, S2 *Series*, ESP32 *Modules* y ESP32 *DevKits* [6] [7].

Los microcontroladores dentro del proyecto han sido seleccionados no solo por sus características, sino también por la facilidad de su programación y su compatibilidad, es por ello, que se debe tener presente el concepto de los IDEs ya que pertenecen a plataformas que trabajan a la par con el microcontrolador [6] [7].

Modulo relé

Los módulos relé se implementan en prototipos, dado que, facilitan la realización de proyectos que usen artefactos electrónicos que requieran voltajes superiores a los manejados por microprocesadores. Un módulo relé se puede considerar como un interruptor capaz de manejar 2 tipos de voltajes para su accionamiento, obteniendo así, un aislamiento entre distintos circuitos, como se muestra en la **Figura 1.2**. Estructura interna del relé [8] [9].

Debido a su modo de funcionamiento, se lo puede encontrar en diversas aplicaciones como, por ejemplo: control de sistemas complejos como motores o luces, ejecución de procedimientos orientados hacia la seguridad, automatización de procesos y maquinaria [8] [9].

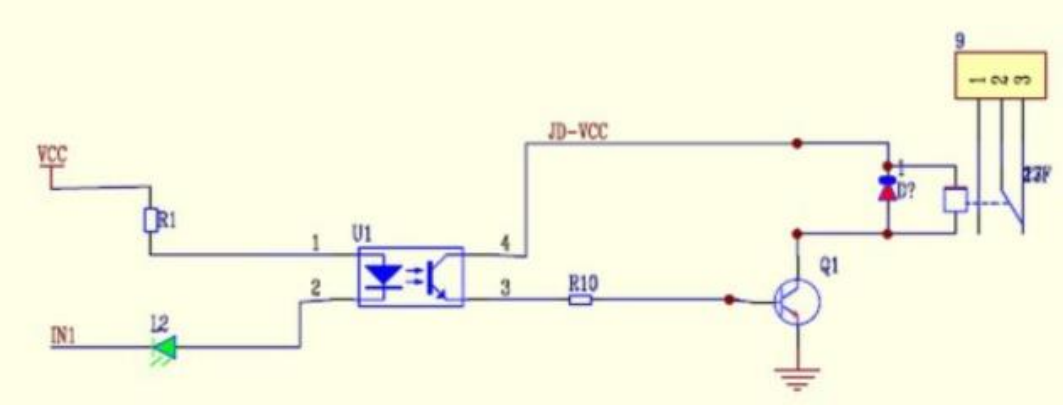

***Figura 1.2**. Estructura interna del relé*

Arduino IDE

Arduino es una plataforma popular enfocada hacia el desarrollo de proyectos que utilicen microcontroladores, la plataforma cuenta con una variada compatibilidad hacia distintos tipos de microcontroladores, además de contar con un gran repertorio de un conjunto de funciones que facilitan su utilización conocida como “librerías”. Las cantidades exorbitantes de conocimiento que se ha generado dentro de Arduino ha sido gracias a su comunidad, por consiguiente, se considera como la plataforma más amigable hacia el usuario [10].

El lenguaje que se implementa en la plataforma es C++, que facilita el aprendizaje e incentiva un desarrollo profundo. Los IDEs permiten modificar el código, verificar su correcta aplicación y el descargo de la información hacia la memoria flash del microprocesador. Por lo tanto, los documentos desarrollados dentro de la plataforma son identificados por su extensión “ino” [10] [11].

Las librerías son parte fundamental en la realización de proyectos, se encargan de agregar varias acciones adicionales para ser ejecutadas dentro del código, estas pueden ser desarrolladas tanto por la empresa como por terceros, algunas de las librerías más utilizadas en conjunto con los microcontroladores ESP32 son, *Wi-Fi*, ESPAsyncwebServer, wire, NTPCclient, entre otras [10] [11].

***Wi-Fi*:**

Es la encargada de efectuar el enlace entre el microcontrolador y la conexión hacia la red de forma estática como dinámica, además tiene la capacidad de crear servidores y comportarse tanto host como cliente [12].

***ESPAsyncWebServer*:**

Desarrollada por terceros, facilita la aplicación de servidores HTTP asincrónicos, dentro de estos servidores se pueden albergar varias peticiones generadas por diversos clientes [13].

***Wire*:**

Agiliza él envió de paquetes de diferente estructura hacia diversos artefactos, lo logra implementando una comunicación I2C la cual utiliza 2 vías de transmisión tales como: SDA y SCL [14].

***NTPClient*:**

Este conjunto de funciones permite realizar y mantener la conexión hacia servidores NTP, estos servidores son los encargados de ofrecer la hora de distintas zonas horarias [15].

***Time*:**

Este conjunto de funciones desarrollados por terceros da la posibilidad de ejecutar varias acciones tales como: generar cronómetros, realizar conexión entre servidores GPS y realizar conexiones hacia servidores NTP [16].

***Wi-Fi*Udp:**

Admite él envió y la recepción de paquetes orientados hacia la utilización de UDP. Además, logra crear una conexión remota mediante la retribución de una IP mediante la inserción de sus subcódigos, cabe recalcar que su investigación y configuración se ha desarrollado por personas externas a Arduino [17].

***Preferences*:**

Permite la utilización de sus diversos espacios de almacenamiento dentro del microcontrolador, es decir, genera un soporte sólido para que el usuario pueda tanto almacenar como utilizar los datos guardados dentro del microcontrolador, es por ello que se presenta como posible sustitución de la librería EEPROM [18].

2 METODOLOGÍA

Como punto de partida, se enfocó en los objetivos y el alcance planteados para comprender las exigencias que se necesita y definir los distintos dispositivos electrónicos para llevar a cabo el desarrollo del proyecto.

Una vez detectados todos los componentes necesarios para el prototipo se procedió con la investigación del *software* que me permita obtener una interfaz gráfica entre *hardware* y el usuario, con el fin de realizar las configuraciones para que el prototipo pueda funcionar tanto en una red interna como una red externa.

El siguiente paso, se deberá seleccionar elementos electrónicos con las características capaces para la aplicación del prototipo, el microcontrolador a seleccionar debe cumplir con las exigencias previstas del proyecto.

Todas las investigaciones mencionadas anteriormente se usarán para el desarrollo del prototipo que permitiría realizar un control inteligente de luminaria residencial, el prototipo logrará encender y apagar las luces individualmente, además podrá encender las luces dentro de un horario específico.

El proceso de selección de los elementos como *software*, *hardware* y el perfeccionamiento del presente proyecto ha sido guiado por las referencias de documentos, trabajos de investigación y foros digitales con temas similares a nuestro tema de investigación han sido de suma importancia, facilitando así la decisión de los requerimientos necesarios que posteriormente serán visualizados.

Como siguiente punto, se analizó las posibles soluciones al problema de conexión a la red tiene el proyecto, es por ello que se realizó una investigación hacia diferentes tipos de plataformas o configuraciones de algún dispositivo adicional para lograrlo, algunos de las soluciones para solventar aquel problema son los siguientes: *Arduino Cloud*, *Adafruit*, configuración interna del router que proporciona el acceso hacia la red de la residencia, *Blynk cloud*, Ngrok, entre otras. Admitirán el acceso hacia la red, además de contar con la generación de una interfaz gráfica dirigida hacia el usuario final.

El siguiente paso del proyecto se encuentra el desarrollo del código que permitirá integrar todos los artefactos seleccionados, se realizó una búsqueda de librerías para solventar cada requerimiento que la interconexión exija, para ello se hiso uso de varios documentos, foros y comunidades de la red especializados en la implementación de microcontroladores, algunos de estos lugares donde se extrajeron información para la ejecución del código del presente proyecto son: *Arduino Fórum*, *ALSWnet*, *Github,* entre otros.

Así pues, teniendo los dispositivos configurados, se desarrollará el esquema de la maqueta en la cual se implementará los componentes del prototipo, además de la realización de un bosquejo de las conexiones que se realizarán dentro de esta, para posteriormente realizar las uniones y las últimas modificaciones, todo aquello teniendo en cuenta las metas a cumplirse expresadas en el planteamiento del proyecto.

Para ejemplificar un entorno real donde se pueda utilizar el prototipo, se investigó la funcionalidad del módulo relé el cual permite la utilización de 5 (v) para accionar diferentes equipos que funciones a voltajes altos, como por ejemplo una luminaria de 110 (v).

Para finalmente, ejecutar las pruebas necesarias para obtener un resultado esperado del prototipo, las pruebas se realizaron mediante la conexión de varios usuarios para corroborar la Inter operatividad, así como la eliminación de cualquier error generado dentro del proceso de implementación, garantizando la funcionalidad del prototipo acorde a los lineamientos propuestos.

3 RESULTADOS

En la siguiente sección del trabajo de integración curricular se procederá a explicar el procedimiento para obtener el prototipo de control inteligente de luces haciendo uso de un microcontrolador ESP32 mediante un desglose de los objetivos propuestos dentro del plan de trabajo del presente documento.

Se podrá encontrar la exploración de las disposiciones que podrían ser útiles dentro del desarrollo del prototipo, seguido de la clasificación de toda la estructura necesaria para su implementación, además de la indagación de las configuraciones de los elementos del proyecto, como también la programación y las plataformas, continuando con la aplicación de los puntos anteriores dentro de una maqueta que permitirá generar una perspectiva de su implementación dentro de una residencia, para concluir con la ejecución de pruebas del prototipo para corroborar el cumplimiento de las expectativas y si es el caso corregir errores existentes.

3.1 Investigar los requerimientos para el diseño del prototipo

Teniendo en cuenta el alcance predefinido dentro de la documentación de este trabajo de titulación, se reconocieron las necesidades que presenta la ejecución del prototipo de control inteligente de luminarias mediante un ESP32.

Energización de la placa

La alimentación constante de los elementos eléctricos es un punto crítico dentro del desarrollo del prototipo. No obstante, se debe tener en cuenta que la energización de las diferentes partes eléctricas del presente proyecto no debe ser sobredimensionada, es decir, el consumo de energía eléctrica utilizada no debe generar un costo adicional que desemboque en el incremento del precio para la implementación del prototipo. Por ello, la gestión del consumo de cada elemento del circuito es parte fundamental para garantizar un desempeño formidable, evitando así la inserción de voltajes erróneos y la falta de potencia.

Además, se debe seleccionar los diversos elementos electrónicos que permita obtener el voltaje requerido para el microcontrolador, evitando así la exposición al daño del voltaje.

Registro horario

El uso de un rango de horas para el control del encendido y apagado de las luminarias es un requerimiento necesario para cumplir con las directrices planteadas en el proyecto. La configuración debe realizarse dentro del código mediante la utilización de software que tenga la capacidad de crearlo, la configuración a ejecutarse tiene como objetivo ofrecer al usuario el encendido y apagado dentro de un periodo deseado.

Para lograr realizar aquel registro horario se requieren bibliotecas o servidores que permitan tener presente la hora, de forma que se puede utilizar sus valores para aplicarlo dentro del código.

Módulo de control de alto voltaje

Al manejar voltajes de alto valor para el correcto funcionamiento de la luminaria, se debe tener presente la diferencia de voltaje presente entre el microprocesador y la iluminación residencial. Por ello, se debe ejecutar una búsqueda de elementos electrónicos que permitan controlar dispositivos de alto voltaje, tales como 110 (V) mediante la utilización de voltajes bajos alrededor de los 3 (V), como es el módulo relé.

Librerías

Para el desarrollo de proyectos que involucran microcontroladores es indispensable la utilización de funciones necesarias para lograr realizar las acciones requeridas. Por ello, las librerías desempeñan una parte critica dentro de cualquier código. Por consiguiente, es fundamental la búsqueda de librerías que generen un soporte al código del presente proyecto.

Las librerías requeridas por el proyecto deben ofrecer un manejo dentro de las siguientes acciones: conexión *Wi-Fi*, creación de página web, manejo de tiempo, compatibilidad hacia él envió de paquetes y almacenamiento de variables.

Navegación por la red

Hoy en día, la navegación en la red es parte fundamental para los prototipos orientados al control remoto, como es el caso de IOT, la conexión se podrá dar mediante la utilización de plataformas que ofrezcan solo el acceso hacia la red sin configuración adicional, o mediante plataformas que cuenten con el requerimiento mencionado anteriormente agregando una interfaz gráfica como es el caso de *Arduino cloud o Ngrok*. Esta conexión hacia la red debe permitir el control del estado de las luminarias simplemente ingresando hacia la página web.

Plataformas IDE

El desarrollo de códigos para proyectos que implican el accionamiento de elementos eléctricos debe realizarse mediante plataformas especializadas para su desarrollo y ejecución compatibles con los diversos microcontroladores que se pueden encontrar dentro del mercado. Sin embargo, existe la necesidad de generar una interfaz gráfica minimalista para permitir la manipulación por parte del usuario. Por ello, se debe buscar

plataformas que favorezcan tanto el desarrollo como la generación de una interfaz gráfica para el presente proyecto, tales como Arduino, Blynk, entre otros.

Desarrollo de PCB

El proyecto cuenta con una extensa variedad de elementos que deben unirse entre sí para dar como resultado el control de luminarias cumpliendo con los objetivos planteados. En consecuencia, se debe generar un circuito fijo también conocido como PCB para facilitar su manipulación, la PCB hará uso de *jumpers,* tanto machos como hembras para abrir la posibilidad de la sustitución o reensamble del microcontrolador.

3.2 Selección del *hardware* y *software*

Teniendo en cuenta los requerimientos anteriormente desglosados para la realización del proyecto de integración curricular, se procedió a escoger los elementos que harán parte de la estructura del prototipo. Los elementos dentro de este proyecto se pueden dividir en 2 grupos, el *hardware* y el *software*.

Selección de *hardware*

El microcontrolador utilizado en el prototipo será seleccionado tomando en cuenta los objetivos que deberá cumplir. Por esta razón, se comparó los diversos desarrolladores de microcontroladores aptos para ejecutar las conexiones tanto entre plataformas para el control del usuario como hacia la red, los proveedores de los microcontroladores orientados hacia IOT que se tomaron en cuenta son: *espressif*, Texas *Instruments* y *Microchip*, se seleccionó los dispositivos elaborados por *espressif* debido a su extenso tiempo dentro del mercado y sus variados modelos son aptos para implementarse en el proyecto.

El microcontrolador destinado para la implementación en el prototipo ha sido seleccionado previamente dentro del plan de desarrollo del proyecto de integración curricular. Por consiguiente, la opción de selección del microcontrolador se centrará en obtener un modelo de ESP32 que concuerde con las capacidades que se requieren para el proyecto en el mercado.

Los modelos que proporciona la empresa espressif para la realización de proyectos se dividen entre 4 grupos. Por lo tanto, se comparará las características de cada categoría dentro de la

Tabla 3.1. Comparativa entre grupos del ESP32 para definir el grupo ideal para implementarlo dentro del proyecto.

Tabla 3.1. Comparativa entre grupos del ESP32 *[6]*

	ESP32-S-*SERIES*	**ESP32-C-*SERIES***	**ESP32-H-*SERIES***	**ESP32 *SERIES***
Microprocesador	LX7 de 1 núcleo	RISC-V de 1 núcleo	RISC-V de 1 núcleo	LX6 de 2 o un núcleo
Frecuencia de trabajo	Hasta 240 MHz	Hasta 120 MHz	Hasta 96 MHz	Desde 80 MHz y 240 MHz
Conectividad	*Wi-Fi*	*Wi-Fi* y *bluethooth*	*Wi-Fi* y *bluethooth*	*Wi-Fi* y *bluethooth*
Memoria RAM	No contiene	576 KB	320 KB	4 MiB
Memoria ROM	No contiene	272 KB	128 KB	8 MiB
Funciones adicionales	Opciones de encriptación	No contiene	Utilizado para dispositivo final y enrutador de borde	Conexión hacia *smartphones* y consumo bajo de energía

Al analizar la

Tabla 3.1. Comparativa entre grupos del ESP32, el grupo de microcontroladores ESP32 Series cuenta con la capacidad de conexión hacia otros dispositivos inteligentes mediante el uso del Wi-Fi y Bluetooth, sin mencionar el espacio de almacenamiento considerablemente extenso a comparación de los otros grupos de microcontroladores. Por consiguiente, se prefirió el grupo ESP32-Series para la selección del microcontrolador que se usará en el proyecto de titulación.

***Tabla 3.2**. Confrontación de placas de desarrollo de ESP32-Series [6]*

	ESP32-WROOM-32E	**ESP32-WROOM-DA**	**ESP32-WROOM-32D**	**ESP32-WROVER-KIT**
Memoria *Flash*	4 MB	4 MB	4 MB	4 MB
Wi-Fi	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n
Versión del *Bluetooth*	4.2	4.2	5.0 LE	4.2

Temperatura de funcionamiento en centígrados	-40 a +85	-40 a +85	-40 a +125	-40 a +85
Precio	$25	$20	$15	$40

Como se observó en la

Tabla 3.2. Confrontación de placas de desarrollo de ESP32-Series, el modelo ESP32-WROOM-32D cuenta con una memoria encargada del almacenamiento de las variables que el código requiera, una SRAM de 8 MiB (mebibyte, 2^{20} bytes según la ISO 80000-13) [19], además de contar con una compatibilidad entre las 2 memorias, manejo de normativas orientadas hacia el Wi-Fi, puertos para el manejo de entradas y salidas del microchip (GPIOs), manejo de energía para lograr un consumo controlado, sistema de prevención para corrientes levemente altas para su alimentación

Puesto que, se requiere almacenar el estado de los focos, la configuración de horarios de encendido y la unión entre plataformas, se seleccionó el modelo ESP32-WROOM-32D para insertarlo en el desarrollo del prototipo debido a que cumplirá todos los requerimientos que el prototipo exige.

Debido a que el prototipo se utilizará dentro de la luminaria de una residencia, la cual maneja voltajes altos a comparación de los GPIOs del microcontrolador, se requiere un artefacto que permita controlar contactos aptos para voltajes de valores altos mediante pulsos de voltaje relativamente bajos. Es allí donde los módulos relé se hacen presentes, el cual permiten satisfacer las necesidades anteriormente mencionadas, además, cuenta con elementos de protección, tales como: diodos de protección, resistencias, leds de prevención, entre otros, estos elementos evitarán cortocircuitos entre ambas partes.

El funcionamiento del módulo relé da inicio con la detección de la señal de control enviado por el microcontrolador. La señal acciona el electroimán dentro del relé, el cual obliga a cambiar de posición los contactos, dando paso a la energización de los artefactos conectados en el circuito de alto voltaje. Los módulos relé cuentan con estados definidos antes de su activación, tales como: NO (*Normally Open*) y NC (*Normally Close*).

El estado NO permite la interrupción de la alimentación del circuito si este no es activado o si el módulo detecta un “0” lógico. Por otro lado, el estado NC permite alimentar el

circuito para su adecuado funcionamiento si cumple con la detección anteriormente mencionada. Si el módulo relé es activado o se observa un “1” lógico los estados anteriores ejecutan la acción contraria.

Debido a que se manejan voltajes peligrosos para dispositivos de bajo voltaje, cuenta con elementos de protección, tales como: un diodo ubicado en paralelo del electroimán; este diodo hace que la corriente circule hacia una dirección evitando así que esta cambie de sentido.

Los módulos relé dentro del mercado cuentan con modelos de diversas prestaciones, los modelos que se pueden encontrar dentro del mercado varían en la cantidad de relés y en el ingreso de señal de control que manejen. Los módulos relé que existen dentro del mercado se dividen en clasificaciones, como son:

Cantidad de canales:

Se puede encontrar módulos relé de 1, 2, 4 y 8 canales, el número de canales del módulo relé dependerá del enfoque del proyecto en donde se aplicará.

Voltaje de alimentación:

La alimentación dentro de un proyecto puede variar según su utilización. Por consiguiente, existe una variedad de módulos relé que satisfacen lo mencionado anteriormente, los módulos que existen dentro del mercado son módulos relé que utilizan 5 (v), 12 (v) y 24 (v).

Selección de *software*.

Otro aspecto para tocar son las plataformas utilizadas, para la selección de la plataforma de desarrollo del código que controlará el prototipo, se buscó la compatibilidad hacia el microcontrolador elegido anteriormente. Por ende, la plataforma Arduino IDE para la realización del código presenta una opción optima, no solo por su amplia compatibilidad que tiene para la utilización de varias librerías sino también por la cantidad exorbitante de información elaborada por la comunidad de Arduino.

Ya definida la plataforma donde se desarrollará el código, se procedió a seleccionar una plataforma que tenga la capacidad de ofrecer al usuario una interfaz gráfica amigable para el control de las luminarias. Como se puede ver en la **Tabla 3.3.** Confrontación de plataformas que ofertan una interfaz gráfica, se analizará las prestaciones que ofrecen las distintas plataformas, considerando que el delimitador del proyecto se genera por el ingreso de variables por parte del usuario y la utilización de horas para el control inteligente de luminarias residenciales.

Tabla 3.3. *Confrontación de plataformas que ofertan una interfaz gráfica [3] [10]*

	Blynk	**Arduino IOT *Cloud***	**Página web local**	***Cadio Home automation***
Capacidad	5 usuarios	20 usuarios	Ilimitado	5 usuarios
Conectividad	Con acceso hacia la red	Con acceso hacia la red	Requiere una salida hacia la red	Con acceso hacia la red
Disponibilidad de proyectos	1	1	Ilimitado	1
Almacenamiento	20 MB en un plan gratuito	2 MB en un plan gratuito	Limitado por el *hardware* aplicado	Ilimitado
Widgets	Limitado por el almacenamiento	20	Ilimitado	3

Debido a las ventajas de realizar un servidor local como se puede divisar en la **Tabla 3.3.** Confrontación de plataformas que ofertan una interfaz gráfica, Se procedió a elegir aquella opción, el microcontrolador ESP32 seleccionado mediante el uso de configuraciones especificas tiene la capacidad de generar una página web con una interfaz gráfica que permita la manipulación del usuario.

La página web generada por el microcontrolador al ser de tipo local tendrá problemas para obtener una conexión a la red. Por esta razón, se investigó formas de dar una salida hacia la red de fácil aplicación para el prototipo. Por ello, se eligió la plataforma *Ngrok*, Ngrok cuenta con la capacidad de generar un túnel para crear un acceso a la red sin generar conflictos en el rendimiento del prototipo, otras causas de su elección fueron la fácil manipulación que se realiza para habilitarlo y la compatibilidad hacia sistemas operativos popularmente usados.

Los proveedores de servicios por temas de seguridad no permiten que el usuario ingrese a sus terminales limitando así las posibles modificaciones que podrían beneficiar al usuario. Es por ello que, las aplicaciones como es Ngrok y sus derivaciones son una vía mucho más factible para dar acceso a la red a servidores locales, como es el caso del presente proyecto.

3.3 Diseñar el prototipo de control inteligente de iluminación

Estructura de la comunicación

La comprensión de las vías que utiliza tanto el prototipo como el usuario para establecer la conexión y realizar los cambios de estado de los leds mediante las diversas posibilidades dentro de la página web, es fundamental para generar una guía del funcionamiento que el conjunto del código logrará realizar. Para lo cual, el presento proyecto contará con la unión entre la página web dentro del ESP32 y la plataforma *Ngrok.*

Al divisar la **Figura 3.1**. Diagrama de conexión entre el ESP32 y el usuario, el control inteligente de luminarias necesita conexión desde el microcontrolador hacia las luminarias de forma que exista la posibilidad de su manipulación, así como la conexión existente entre el módulo relé y el microcontrolador; añadido a lo anteriormente mencionado, se contará con la unión entre la página web y la red haciendo uso de las plataformas especializadas mencionadas anteriormente y la coordinación del envío de datos esenciales tales como la asignación de horarios de encendido, apagado de las luminarias y la variación de la potencia hacia el ESP32.

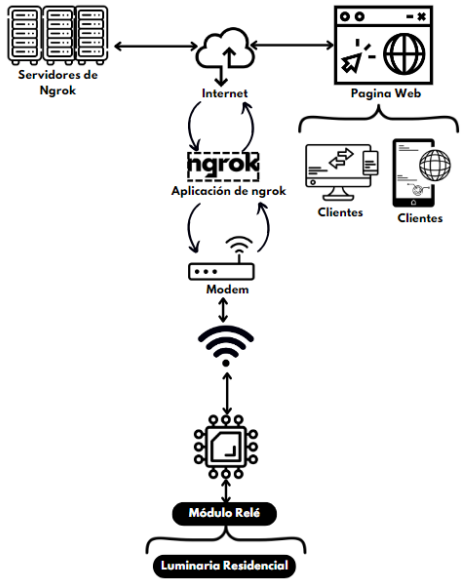

***Figura 3.1**. Diagrama de conexión entre el ESP32 y el usuario*

Como se distingue en la **Figura 3.1**. Diagrama de conexión entre el ESP32 y el usuario, se visualiza que, el encargado de atender las peticiones realizadas por el cliente es el microcontrolador *ESP32-WROOM-32D*, estas peticiones viajan mediante la red hacia

los servidores de *Ngrok.* Para posteriormente, redireccionarse hacia el microcontrolador que generará y enviará la página web para el control de luminarias por parte del usuario haciendo el mismo recorrido que se realizó anteriormente.

Desarrollo de código

Las librerías utilizadas dentro de la programación es el paso inicial para el desarrollo del proyecto, es por ello que se requiere llamar a cada una de estas para hacer uso de sus acciones, en este caso se utilizaran 5 librerías, como se ve en la **Figura 3.2:** Librerías utilizadas en el proyecto.

```
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
```

Figura 3.2: *Librerías utilizadas en el proyecto*

En el siguiente paso de la programación podemos encontrar la declaración de las variables globales, estas serán utilizadas para almacenar el estado de las luminarias, además, se podrá encontrar las credenciales del acceso a la red de la vivienda, la asignación del puerto a utilizar y la asignación de una IP fija. Como se puede ver en la **Figura 3.3**. Variables globales y credenciales para el acceso a la red, podemos divisar algunas de las variables para el encendido y apagado, para regular el brillo las luminarias y la asignación de los leds a los GPIOs del microcontrolador.

```
// credenciales para acceso al internet
const char* ssid = "router01";
const char* password = "LuisFlores18";

//declaración de variables necesarias
const int led = 4;
const int led1 = 16;
const int led2 = 17;
const int led3 = 18;

const char* PARAM_INPUT_1 = "input1";
const char* PARAM_INPUT_2 = "input2";
const char* PARAM_INPUT_3 = "input3";
const char* PARAM_INPUT_4 = "input4";

String sliderValueprueba = "0";
String sliderValueprueba1 = "0";
String sliderValueprueba2 = "0";
String sliderValueprueba3 = "0";
```

Figura 3.3*. Variables globales y credenciales para el acceso a la red*

Una dirección fija permitirá que la conexión entre los usuarios y la página web no sea alterada por un corte de alimentación eléctrica, para este caso se eligió la IP 192.168.1.4, la cual se mantiene dentro del rango que oferta el router del domicilio,

además de ello se definió el puerto 80 que se utilizará para la conexión de los usuarios hacia el microcontrolador.

Para que el ESP32 pueda comprender y utilizar el archivo HTML, se realizó funciones tales como: *UPdatessliderPWM,* las cuales son las encargadas de asignar una respuesta a la interacción de los elementos de la página web y su enviado hacia el ESP32 como cadena de texto plano, de forma que el microcontrolador detecte una interacción, tal como se ve en la **Figura 3.4:** Funciones para el envío de una respuesta a la interacción en la página web. Debido a la utilización de varios elementos dentro de la página web, esta será replicada para cada elemento que requiera una interacción del usuario.

```
function updateSliderPWMprueba(element) {
  var sliderValueprueba = document.getElementById("pwmSliderprueba").value;
  document.getElementById("textSliderValueprueba").innerHTML = sliderValueprueba;
  console.log(sliderValueprueba);
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/sliderprueba?value="+sliderValueprueba, true);
  xhr.send();
}

function updateSliderPWMprueba1(element) {
  var sliderValueprueba1 = document.getElementById("pwmSliderprueba1").value;
  document.getElementById("textSliderValueprueba1").innerHTML = sliderValueprueba1;
  console.log(sliderValueprueba1);
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/sliderprueba1?value="+sliderValueprueba1, true);
  xhr.send();
}
```

Figura 3.4: *Funciones para el envío de una respuesta a la interacción en la página web*

Posteriormente dentro del *void setup*, se procede a generar la página web si existe una petición generada por un usuario tomando en cuenta el archivo HTML que contiene su estructura, para ello se hace uso del comando *server*.on especificando dentro de este la utilización del archivo index_html si existe una petición de un usuario, como se ve en la **Figura 3.5**: Utilización del archivo HTML por petición.

```
server.on("/", HTTP_GET, [](AsyncWebServerRequest* request) {
  request->send_P(200, "text/html", index_html, processor);
});
```

Figura 3.5*: Utilización del archivo HTML por petición*

La página web manda cadenas de texto hacia el microcontrolador, por ello se debe especificar qué acción se debe realizar para cada respuesta enviada hacia el ESP32, por consiguiente, se realizó la configuración de las acciones teniendo en cuenta la respuesta como se ve en la **Figura 3.6:** Asignación de acciones a la interacción de la página web, para lograrlo se utilizó el comando server.on y la respuesta específica a leer, en el cuerpo del comando se detalla el almacenamiento del estado del elemento y

su acción correspondiente. Debido a la presencia de varias acciones configuradas en la página web el proceso se repetirá hasta definir una acción para cada respuesta.

```
server.on("/slider", HTTP_GET, [](AsyncWebServerRequest* request) {
  String inputMessage;
  // GET input1 value on <ESP_IP>/slider?value=<inputMessage>
  if (request->hasParam(PARAM_INPUT)) {
    inputMessage = request->getParam(PARAM_INPUT)->value();
    sliderValue = inputMessage;
    ledcWrite(ledChannel, sliderValue.toInt());
  } else {
    inputMessage = "No message sent";
  }
  Serial.println(inputMessage);
  request->send(200, "text/plain", "OK");
});
```

Figura 3.6: *Asignación de acciones a la interacción de la página web*

Finalmente, se programó dentro del *void loop* el encendido y apagado automático de las luminarias utilizando los valores horarios ingresados por la página web, tal como se ve en la **Figura 3.7:** Programación del encendido horario de las luminarias, primero se asignó variables comprensibles para el microcontrolador para guardar la hora deseada por el usuario evitando así el problema de incompatibilidad entre lenguajes diferentes, para finalmente definir mediante la utilización del comando *if* la hora de encendido o apagado comparando la hora actual y el valor almacenado en la variable hora y minuto para los 2 casos.

```
timeClient.update();
int currentHour = timeClient.getHours();
int currentMinute = timeClient.getMinutes();
int currentSecond = timeClient.getSeconds();

Serial.print("Hora: ");
Serial.println(currentHour);

Serial.print("minutos: ");
Serial.println(currentMinute);

delay(1000);

if (hora1 == currentHour && minutos1 == currentMinute) {
  ledcWrite(ledChannel, 255);
  ledcWrite(ledChannel1, 255);
  ledcWrite(ledChannel2, 255);
  ledcWrite(ledChannel3, 255);
}
if (hora2 == currentHour && minutos2 == currentMinute) {
  ledcWrite(ledChannel, 0);
  ledcWrite(ledChannel1, 0);
  ledcWrite(ledChannel2, 0);
  ledcWrite(ledChannel3, 0);
}
```

Figura 3.7: *Programación del encendido horario de las luminarias*

Creación de la página web

Para la conexión entre el usuario y el microcontrolador se requiere una interfaz que permita al usuario manipular a su preferencia el estado de las luminarias. Por lo tanto, se debe realizar una interfaz mediante una página web, teniendo en cuenta la comodidad del usuario al momento de su uso. La página web al desarrollarse dentro del

microcontrolador, se requiere ejecutar un lenguaje específico para esa clase de acciones conocido como HTML, de forma que las funciones programadas no generen problemas.

Se generó y guardó la cadena de texto que tendrá la estructura de la página web dentro de la memoria denominada PROGMEM en la variable llamada index_html[], luego se especificó el estilo de la página web insertando los elementos que conformarán la página web tal como se ve en la **Figura 3.8:** Definición del estilo de la página web, los elementos a utilizar serán: los botones de apagado, las casillas de ingreso de la información, el botón de almacenamiento de la información insertada, la barra de intensidad de las luminarias, el título, los diferentes colores y la visualización de la hora.

```
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>ESP Web Server</title>
  <style>
 html {font-family: Arial;
  display: inline-block;
  text-align: center;  background-color: #bfc0c2;;}
  .topnav {
  font-family: 'Courier New', Courier, monospace;
  font-weight: bold;
  overflow: hidden;
  background-color: #4e6b89;
  color: #161616;
  font-size: 1rem;
```

Figura 3.8: *Definición del estilo de la página web*

Aquellos elementos son relacionados dentro de la variable haciendo uso de sinónimos dentro del lenguaje HTML, tales como: *text-aling, slider, card, card title*, *topnav, container clock e input.* Dentro de cada una de estas se insertan las dimensiones, colores y todos los requerimientos que tendrán en la página web.

Siguiendo la configuración de la página web, se procedió a realizar el cuerpo de la página web, para ello se utilizó las especificaciones anteriores para posteriormente ubicarlas dentro de la página web y establecer una variable para cada elemento si este lo necesita de forma que el valor almacenado se pueda utilizar en el código, como se puede ver en la **Figura 3.9:** Cuerpo de la página web, además se agregó texto sobre los elementos para mejorar el funcionamiento del usuario. Lo realizado allí se puede ver completo en el anexo

```
<div class="container-clock">
        <h1 id="time">00:00:00</h1>
    </div>

<div class="cards">
<div class="card">
<p class="card-title">HORA DE ENCENDIDO</p>
<form action="/get" target="Hidden-form"><br>
   <input type="number"  placeholder ="Ingrese la hora" name="input1">
    <input type="submit" value="Guardar">
  </form><br>
  </form>
  <form action="/get1" target="Hidden-form"><br>
   <input type="number"  placeholder ="Ingrese los minutos"name="input2">
    <input type="submit" value="Guardar">
  </form><br>
  </form>
```

Figura 3.9: *Cuerpo de la página web*

En el cuerpo de la configuración de la página HTML se definió los estados que tendrán los elementos, los estados permiten generar un cambio de posición que desatará una acción específica dentro del código realizado como es el encendido y apagado de las luminarias. Posteriormente se utilizará estas configuraciones de manera que el ESP32 genere la página web en base a lo realizado obteniendo así la **Figura 3.10.** Interfaz gráfica del proyecto.

Figura 3.10. *Interfaz gráfica del proyecto*

Configuración de Ngrok

Se hará uso de la plataforma *Ngrok* que facilitará la conexión a la red, de forma que, el usuario pueda conectarse a la página web y utilizar los *widgets* para el control de la luminaria.

En primer lugar, se registró dentro de la plataforma para hacer uso del plan gratuito. Luego, se descargó el archivo comprimido de la plataforma *Ngrok* desde la página oficial, cabe recalcar que la implementación de *Ngrok* debe realizarse en un dispositivo

activo conectado dentro de la red en la que se encuentra el prototipo. Posteriormente, dentro del terminal activo se descomprimirá el archivo y se instalará en el artefacto, cabe recalcar que, el artefacto donde se instalará el programa servirá como intermediario, al instalar la aplicación *Ngrok* se deberá admitir permisos de administrador para evitar errores a futuro. Finalmente, se aceptan los términos que la plataforma obliga.

Finalizada la instalación de la aplicación, se procedió a verificar la conexión entre el terminal y el microcontrolador mediante el acceso de la página web de acceso local. A continuación, se estableció el comando “Ngrok http 192.168.1.4“, en donde la IP insertada será la asignada estáticamente a la página web creada por el microcontrolador

El resultado arrojado por la aplicación visualizado en la **Figura 3.11.** Asignación del link para el acceso desde Internet, proporciona información adicional de la conexión creada. Finalmente, para permitir la conexión de usuarios que se encuentren fuera de la red local se copiará y compartirá el link ubicado en el apartado *forwarding*, de manera que, cualquier usuario que contenga el link tendrá la posibilidad de conectarse remotamente al prototipo.

Figura 3.11. *Asignación del link para el acceso desde Internet*

Para corroborar la asignación de la dirección estática configurada dentro del código, se utilizó la aplicación *Fing*, la aplicación *Fing* realiza un escaneo exhaustivo de los dispositivos conectados a un punto de red en específico, permitiendo la visualización tanto de la dirección asignada por el router como de algunos datos técnicos del *hardware* del dispositivo. Como se puede ver en la **Figura 3.12.** Escaneo de la red, dentro de la red local se encuentran varios dispositivos haciendo uso del servicio de la red, sin embargo, se puede identificar la IP asignada al prototipo mencionado como dispositivo inteligente.

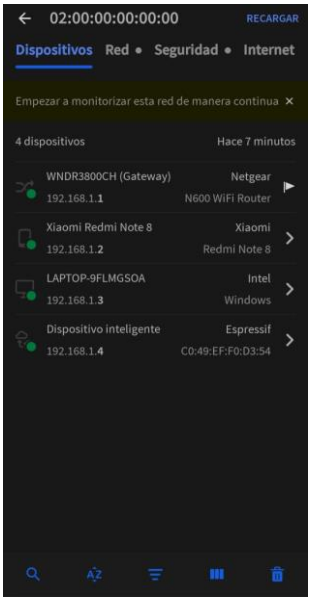

Figura 3.12. *Escaneo de la red*

Diagramas del circuito

Anterior a la ejecución del circuito impreso, se desarrolló el diagrama de uniones que hará funcionar el prototipo de manera óptima haciendo uso de un *software* de simulación conocido como Proteus, Como se muestra en la **Figura 3.13.** Diagrama de uniones eléctricas, se utilizó 2 hileras de *headers* hembra separadas para la introducción del microcontrolador, debido a la falta de librerías del microcontrolador ESP32 de 38 pines compatibles con el *software* Proteus, además de facilitar la extracción del ESP32 si fuese necesario.

Seguidamente, se utilizó *headers* macho en las señales de salida proporcionados por el microcontrolador, de forma que sea de fácil manipulación. Por otro lado, para la energización del microcontrolador se requirió el uso de 2 *headers* y un regulador de voltaje como se mencionó anteriormente.

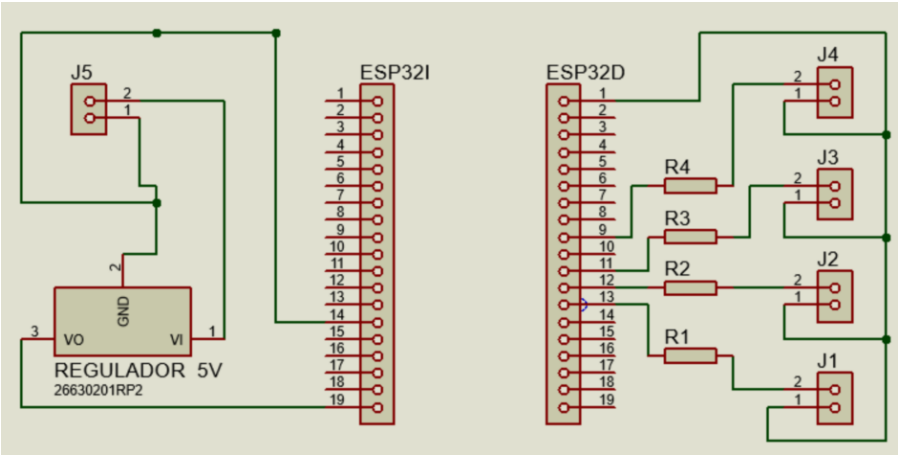

Figura 3.13. *Diagrama de uniones eléctricas*

3.4 Implementación del prototipo

Desarrollo de PCB

Al obtener un resultado satisfactorio dentro del *software* de simulación Proteus y en la elaboración del código agregado en los anexos, se procedió a realizar la PCB. Por esta razón, se obtuvo una baquelita virgen con las dimensiones 10 (cm) de largo y 7 (cm) de Ancho. Proteus además de proporcionar la simulación de circuitos eléctricos oferta un apartado que permite a los usuarios realizar diseños PCB basados en la distribución de pines previamente realizada, siempre y cuando los elementos utilizados en el circuito sean compatibles con aquel apartado.

En consecuencia, a lo mencionado anteriormente, se elaboró el diagrama de conexiones haciendo uso del auto enrutamiento de pistas y la configuración para generar el circuito en una cara, para posteriormente imprimirlo sobre el papel transfer con un escalado del 100% y el efecto espejo, como se ve en la **Figura 3.14**. Esquema de conexiones para aplicar en la PCB.

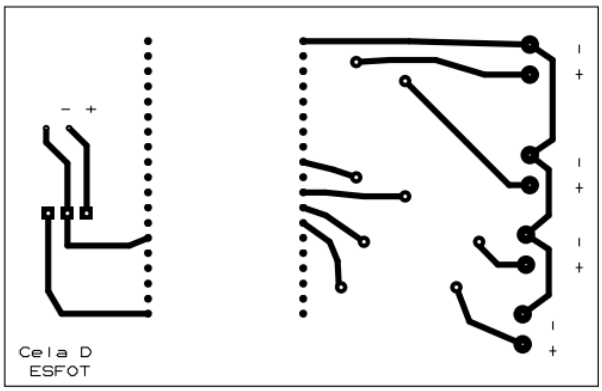

***Figura 3.14**. Esquema de conexiones para aplicar en la PCB*

A continuación, Se traspasó el esquema de conexiones elaborado en Proteus hacia la baquelita mediante la técnica de planchado. La técnica de planchado no es más que aplicar calor al papel transfer ubicado en la cara de cobre de la baquelita. Para posteriormente, retirar el sobrante mediante la utilización del agua, cabe recalcar que, la técnica mencionada depende mucho del tiempo y la uniformidad del planchado para obtener un acabado impecable. Finalmente, se divisará la transferencia del circuito electrónico en la baquelita, como en la **Figura 3.15.** Esquema de conexión reflejado en la baquelita.

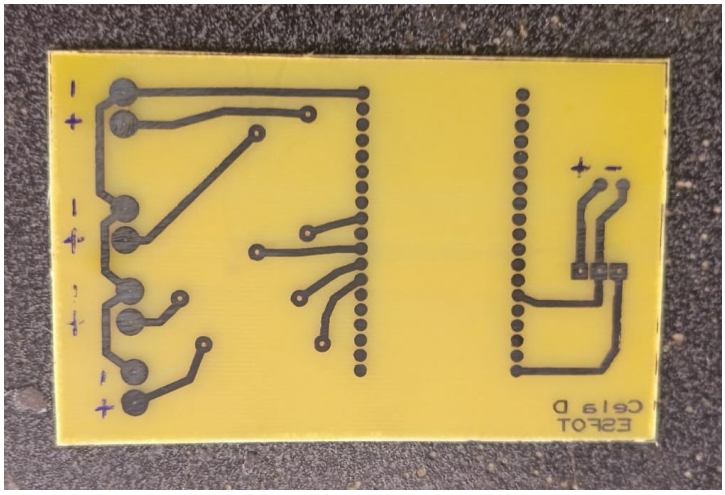

Figura 3.15. *Esquema de conexión reflejado en la baquelita*

Finalizado el traspaso del esquema del circuito electrónico hacia la baquelita, se procedió a quemar los caminos del prototipo. Para lograrlo, se adquirió ácido férrico, para posteriormente, insertarlo dentro de un recipiente de plástico lo suficientemente extenso para que la baquelita sea cubierta por el ácido, de esta forma obtener una PCB de buena calidad y sin inconvenientes. Luego, se introdujo la baquelita hacia el ácido férrico como se visualiza en la **Figura 3.16**. Quemado del diagrama de conexiones.

Figura 3.16*. Quemado del diagrama de conexiones*

Seguidamente, se realizó una limpieza de los restos de ácido y tinta presentados en la placa quemada, además, se comprobó que los caminos realizados cuenten con una separación que les impida ocasionar un corto circuito. Finalizada la limpieza de la placa da como resultado la placa visualizada en la **Figura 3.17.** PCB finalizada. Posteriormente, se realizó los agujeros haciendo uso de brocas del tamaño adecuado que permitirán el ensamble de los elementos electrónicos presentes en el circuito.

Figura 3.17. *PCB finalizada*

Por último, se ejecutó el ensamblaje de los elementos electrónicos. Para luego, soldar los elementos hacia la PCB de manera que queden lo suficientemente separados para evitar cortocircuitos o el daño de los elementos, el elemento para soldar dichos elementos no deberá contener un manejo de energía muy alto debido al riesgo de quemar los elementos por exceso de calor. Para corroborar el estado de los elementos soldados, se procedió a revisar la continuidad entre las soldaduras, obteniendo así la placa base observada en la **Figura 3.18.** Placa base finalizada.

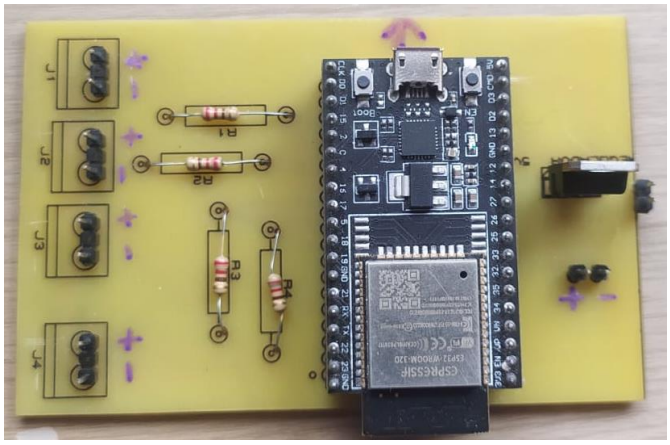

Figura 3.18. *Placa base finalizada*

Al utilizar voltajes con valores superiores a los manejados por el microcontrolador, se hiso uso del módulo mencionado anteriormente, el módulo relé lee el pulso emitido por el microcontrolador. Para posteriormente, accionar el relé que cambiará el estado del foco, cabe recalcar que, el módulo relé debe ser energizado con un voltaje de 5 (v). Por consiguiente, existen 2 formas de realizarlo.

La primera opción, es conectar tanto la alimentación como la señal emitida por el microcontrolador hacia el módulo relé, la desventaja de ejecutar este tipo de conexión

es la baja protección hacia el microcontrolador, por otro lado, la conexión con una mayor seguridad alimenta el módulo mediante una batería externa limitando las entradas y salidas que se conectan directo al ESP32.

Se agregó un módulo regulador de voltaje con alimentación externa para energizarlo, como se puede divisar en la ***Figura 3.19.*** *Conexión hacia el módulo relé*. El módulo de alimentación externa contiene un regulador de voltaje para la batería de 9 (v), de modo que se energiza al módulo relé con 5 (v) y 10 (mA), evitando así dañarlo, el voltaje resultante del 7805 se insertará dentro de los pines de alimentación GND y JD-VCC. Luego, del microcontrolador se utilizará la alimentación de 5 (v) para insertarlo dentro del terminal VCC como la señal de control generada por el programa la cual se conecta hacia el relé que se haga uso.

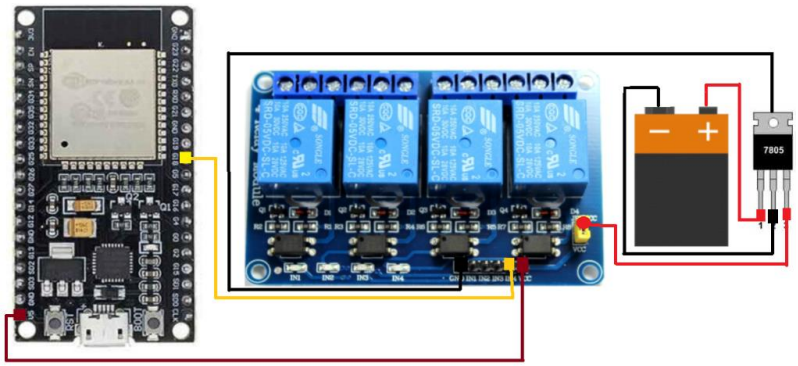

Figura 3.19. *Conexión hacia el módulo relé*

Como se puede divisar en la ***Figura 3.19.*** *Conexión hacia el módulo relé*, la configuración presentada para la conexión del relé como se mencionó sobrevoltajes, además de minimizar el consumo de energía que requerirá energizar aquel módulo. Por lo tanto, se eligió aquella conexión para minimizar los posibles errores que se puedan presentar.

Diseño de la Maqueta

Para ofrecer una mejor apreciación del funcionamiento del prototipo por parte del usuario, se elaboró una maqueta de una residencia pequeña que contenga una estructura popularmente utilizada, de forma que sea posible la implementación del prototipo.

Se diseño la distribución de la maqueta acorde a la investigación previamente realizada sobre residencias de dimensiones pequeñas. La distribución de la maqueta contendrá: 1 cocina, 1 dormitorio, 1 comedor y un baño, tal como se divisar en la **Figura 3.20.** Maqueta final.

La implementación del proyecto genera un altercado visual, ya que se requiere unir los diferentes focos hacia la placa base. Para solventar el problema, se utilizó pajillas de color blanco que permitió ocultar los cables necesarios para el funcionamiento del prototipo y dar así una perspectiva más organizada.

Figura 3.20. *Maqueta final*

Para salvaguardar la integridad de todos los elementos electrónicos que conforman el prototipo realizado anteriormente, se realizó un contenedor lo suficientemente extenso para que cupiesen, obteniendo así, una mejor visualización de la maqueta para un análisis previo.

3.5 Realizar pruebas de funcionamiento del prototipo

Finiquitado el proceso de ejecución del control inteligente de luces en la maqueta, se desarrolló una serie de pruebas al prototipo para corroborar el cumplimiento de los objetivos planteados anteriormente, así como el correcto desempeño del prototipo realizado.

Conexión entre el microcontrolador y la red local

Primero, se corrobora que el microcontrolador ESP32 mediante la utilización del código fuente sea capaz de conectarse a la red local, la conexión hacia la red local deberá realizarse siempre con la IP fija asignada dentro del código, la cual es: 192.168.1.4, de forma que el usuario inserte la dirección estática en el navegado de su preferencia . Por esta razón, se procede a conectarse desde cualquier dispositivo dentro de la red local hacia la página web mediante la IP asignada al a página web, tal cómo se divisa en la **Figura 3.21.** Validación de la conexión del microcontrolador hacia la red local.

Figura 3.21. *Validación de la conexión del microcontrolador hacia la red local*

Como se puede ver en la **Figura 3.21.** Validación de la conexión del microcontrolador hacia la red local. El *router* llamado "*router*01" será utilizado para permitir la conexión entre el microcontrolador y la red local, es decir, es el encargado de manejar localmente las peticiones de los usuarios al microcontrolador redirigiéndoles hacia la página web, además se divisa una conexión sin problemas a la página web configurada.

Salida hacia la red

Una vez verificado el correcto funcionamiento del prototipo de forma local, se comprobó la correcta labor de la plataforma Ngrok para obtener una salida hacia la red, de forma que el usuario tenga la capacidad de conectarse a la página web desde cualquier localidad, siempre y cuando este tenga acceso a la red. Por lo tanto, se procede a copiar el link proporcionado por la aplicación Ngrok en un dispositivo que se encuentre fuera de la red local, como se puede ver en la **Figura 3.22.** Conexión del usuario desde una red distinta.

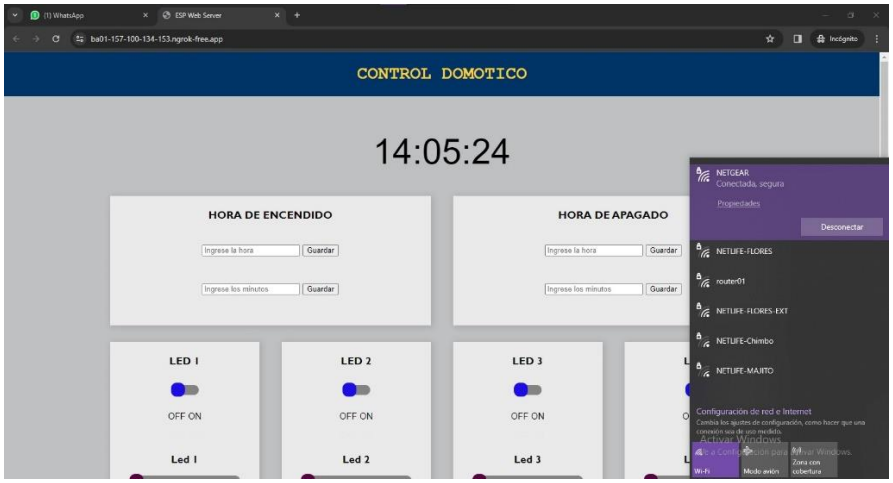

Figura 3.22. *Conexión del usuario desde una red distinta*

En la **Figura 3.22.** Conexión del usuario desde una red distinta, podemos ver que el acceso hacia la página web creada dentro del microcontrolador es diferente a la utilizada en la **Figura 3.21.** Validación de la conexión del microcontrolador hacia la red local. Dado que, se usa la aplicación Ngrok enlaza la IP anteriormente asignada hacia un link reconocible dentro de la red, cabe recalcar que supongamos que la máquina que aloja el programa sufre un percance, la salida hacia la red del prototipo sufrirá una caída repentina.

Uso de las funciones del controlador inteligente de luminarias

Finalizado las pruebas de conectividad entre el usuario y la página web, se procede a realizar pruebas a las diversas funciones dentro del prototipo para validar el buen funcionamiento del control inteligente. Una de las opciones que proporciona el prototipo realizado es el encendido de las luminarias mediante un rango de horas.

En el estado inicial, los leds permanecen apagados, debido a la falta de inserción del rango de horas que defina el encendido y apagado de las luminarias. Sin embargo, si el usuario agrega un rango de horas a la página web, el estado de los leds cambiarán siempre y cuando los valores insertados sean los deseados. Como se divisa en la **Figura 3.23**. Estado final de la configuración mediante el uso de un rango de horas.

***Figura 3.23**. Estado final de la configuración mediante el uso de un rango de horas*

El prototipo adicionalmente a lo mencionado anteriormente contiene la posibilidad de modificar el estado individualmente de las luminarias. Por lo tanto, se procede a realizar varios ensayos del funcionamiento del control individual de las luces para corroborar su buen funcionamiento, tal como se visualiza de **Figura 3.24.** Pruebas de funcionamiento del encendido y apagado individual de las luminarias.

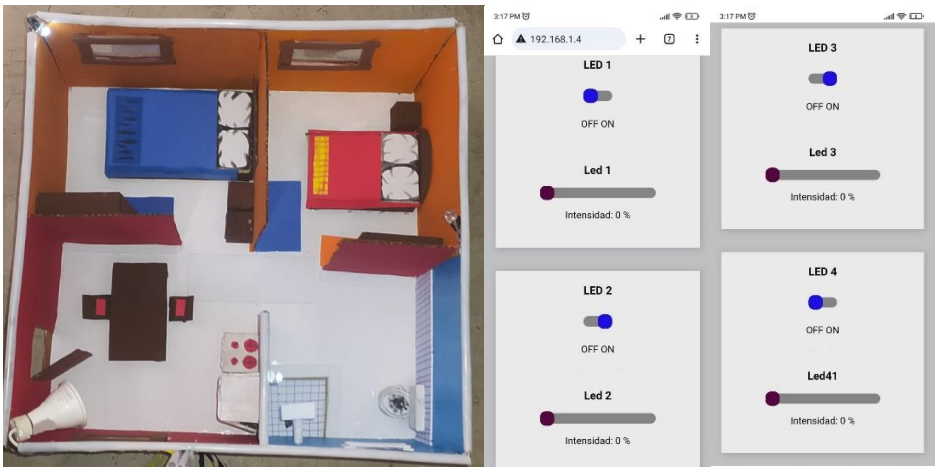

***Figura 3.24.** Pruebas de funcionamiento del encendido y apagado individual de las luminarias*

Finalmente, la última función ofertada por el microcontrolador es el control de potencia mediante porcentajes. Para corroborar el correcto funcionamiento de la función mencionada, se establecen pruebas de la función de la potencia de los focos, aquello se puede ver dentro de la **Figura 3.25.** Pruebas de funcionamiento del control de potencia de las luminarias.

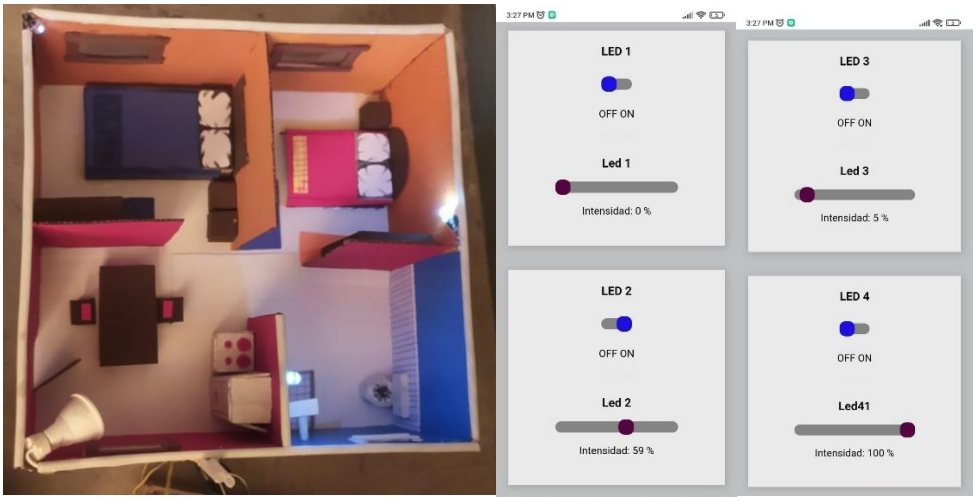

Figura 3.25. *Pruebas de funcionamiento del control de potencia de las luminarias*

4 CONCLUSIONES

- A través de las investigaciones realizadas en el transcurso de este proyecto, se llega a concluir que la implementación de un sistema de control inteligente de luminarias ha resultado en una optimización significativa de su funcionamiento, además, ha generado un mayor nivel de control sobre la iluminación residencial, sin descuidar su integridad y sobre todo el bienestar de los usuarios.
- Para la integración de todos los componentes electrónicos tales como el microcontrolador y las funciones que se pueden programar dentro del mismo, Arduino IDE ha destacado, ya que cumplió un trabajo satisfactorio debido a una gran compatibilidad con la mayoría de los microcontroladores comerciales, en especial con la tarjeta de ESP32.
- La implementación del prototipo en una maqueta contribuyó con el desarrollo de este proyecto, debido a que emula a una implementación en una residencia, además de revelarnos posibles acontecimientos al momento de su instalación que posteriormente serán solventados obteniendo así una versión más optimizada del proyecto.
- Las pruebas realizadas demuestran que el funcionamiento del prototipo es funcional y eficaz, se permitió comprobarlo mediante la modificación del estado de las luminarias mediante la página web tanto dentro de la red local como en una localidad lejana haciendo uso de una conexión a la red.
- El módulo relé cumple un rol fundamental en la presentación del proyecto ya que al ser un dispositivo electrónico que trabaja con voltajes mínimos y logra activar equipos a un voltaje mayor, se puede comprender que el prototipo puede trabajar sin problemas en un entorno real, ya sea para encender luminarias que trabajan con 110 (v), encender motores o activar alarmas.
- El *software* Ngrok, ha sido parte fundamental para el desarrollo del prototipo, ya que crea un puente seguro para la utilización de aplicaciones de red interna permitiendo la interconexión entre usuarios de redes externas.

5 RECOMENDACIONES

- Las variables utilizadas para guardar el estado de los elementos de una página web generan conflictos al utilizarlos dentro del código, por consiguiente, se recomienda utilizar variables generadas con el lenguaje de programación original donde se guardará en contenido de la variable con problemas evitando errores de compatibilidad.
- Si bien la utilización del módulo relé ayuda en el manejo de dispositivos alimentados por voltajes altos, contiene una limitación en varios de sus modelos, al limitarse a comprender solo un encendido y un apagado en la señal de control crea conflictos con los reguladores de intensidad de las luces, por lo tanto, se recomienda utilizar módulos relés inteligentes o dispositivos similares adicionales.
- Algunos proveedores de servicio restringen el acceso al router del cliente para precautelar la seguridad de la empresa, por ello, se recomienda hacer uso de aplicaciones gratuitas que facilitan la conexión hacia la red externa sin alterar la estructura del terminal que proporciona conexión a la red.
- Se recomienda utilizar modelos de ESP32 que sean compatibles para todos los apartados del *software* de diseño electrónico como, por ejemplo, el microcontrolador ESP32-DEVKIT el cual cuenta con librerías capaces de garantizar su uso dentro de programas, tales como Proteus.
- Se recomienda que, para futuros trabajos relacionados al tema, se inserte una fotocelda dentro del prototipo de forma que las luminarias tengan otro método de automatizar las luminarias empleadas dentro de este proyecto.
- Se recomienda que, para futuros proyectos, su implementación sea 100% dentro de una edificación o vivienda real, evitando así la realización de una maqueta que puede presentar problemas no relacionados a su lugar ideal de aplicación.

REFERENCIA BIBLIOGRÁFICA

[1] IBM, «What is the internet of things (IoT)?,» IBM in Deutschland, Österreich und der Schweiz, [En línea]. Available: https://www.ibm.com/topics/internet-of-things. [Último acceso: 19 02 2024].

[2] R. Kolavo, «ngrok | Unified Application Delivery Platform for Developers,» Ngrok, 16 02 2024. [En línea]. Available: https://ngrok.com/docs/what-is-ngrok/. [Último acceso: 19 02 2024].

[3] Arduino, «arduino.cl,» [En línea]. Available: https://arduino.cl/que-es-arduino/. [Último acceso: 19 02 2024].

[4] E. Systems, «Espressif,» [En línea]. Available: https://www.espressif.com/en/products/socs/esp32. [Último acceso: 19 02 2024].

[5] S. GLOBAL, «LinkedIn,» 21 07 2021. [En línea]. Available: https://www.linkedin.com/pulse/microcontroladores-qu%C3%A9-son-y-su-importancia-en-la-industria-/?originalSubdomain=es. [Último acceso: 19 02 2024].

[6] E. Estudio, «Electronica Estudio,» [En línea]. Available: https://www.estudioelectronica.com/que-es-un-microcontrolador/. [Último acceso: 19 02 2024].

[7] J. Beningo, «digikey,» 21 01 2020. [En línea]. Available: https://www.digikey.com/es/articles/how-to-select-and-use-the-right-esp32-wi-fi-bluetooth-module. [Último acceso: 19 02 2024].

[8] NovatronicEc, «NovatronicEc,» [En línea]. Available: https://novatronicec.com/index.php/product/esp32-placa-de-desarrollo-v1/. [Último acceso: 19 02 2024].

[9] William, «geya,» 14 11 2022. [En línea]. Available: https://www.geya.net/es/what-is-a-relay-module-and-what-does-it-do/. [Último acceso: 19 11 2024].

[10] A. Electronics, «avelectronics,» [En línea]. Available: https://avelectronics.cc/producto/modulo-rele-4-canales/. [Último acceso: 19 02 2024].

[11] Arduino, «docs.arduino,» 16 02 2024. [En línea]. Available: https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics/. [Último acceso: 19 02 2024].

[12] Arduino, «arduino.cc,» [En línea]. Available: https://www.arduino.cc/reference/en/libraries/wifi/. [Último acceso: 19 02 2024].

[13] C. Pascual, «programarfacil,» 2022. [En línea]. Available: https://programarfacil.com/esp32/servidor-web-con-esp32/#comment-5833012770. [Último acceso: 19 02 2024].

[14] Arduino, «arduino,» 05 12 2023. [En línea]. Available: https://www.arduino.cc/reference/en/language/functions/communication/wire/. [Último acceso: 19 02 2024].

[15] Arduino, «arduino,» 25 04 2015. [En línea]. Available: https://www.arduino.cc/reference/en/libraries/ntpclient/. [Último acceso: 19 02 2024].

[16] M. Margolis, «Arduino,» 31 12 2018. [En línea]. Available: https://playground.arduino.cc/Code/Time/. [Último acceso: 19 02 2024].

[17] L. Martorella, «github,» 15 08 2023. [En línea]. Available: https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/WiFiUdp.h. [Último acceso: 19 02 2024].

[18] L. LLamas, «luisllamas,» 05 03 2023. [En línea]. Available: https://www.luisllamas.es/esp32-preferences/#:~:text=La%20biblioteca%20preferences.,la%20memoria%20flash%20del%20dispositivo. [Último acceso: 19 02 2024].

[19] O. Moreno, *M.I.B IEEE 802.11 como información para producir registtros y evidencia digital,* Bogotá, 2007, p. 7.

6 ANEXOS

La lista de los Anexos se muestra a continuación:

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces

ANEXO III. Conjunto de datos extensos

ANEXO I: Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 15 de Marzo de 2022

De mi consideración:

Yo, CARLOS ANDRÉS YUNGA SÁNCHEZ, en calidad de Director del Trabajo de Integración Curricular titulado IMPLEMENATACIÓN DE UN PROTOTIPO DE CONTROL INTELIGENTE DE ILUMINACIÓN RESIDENCIAL UTILIZANDO ESP32 elaborado por el estudiante DANIEL SEBASTIAN CELA TOAQUIZA de la carrera en Tecnología Superior en Redes y Telecomunicaciones, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 8%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

LINK

3. Reporte de Turnitin_Daniel_Cela_Tic_2023B.pdf

Atentamente,

CARLOS ANDRÉS YUNGA SÁNCHEZ

Docente

Escuela de Formación de Tecnólogos

ANEXO II: Enlaces

A continuación, se insertará el link donde se podrá visualizar el video demostrativo del control inteligente de luminarias, así como el código QR del mismo:

Video_demostrativo_Cela Daniel.mp4

Anexo II.I Código QR de la implementación y pruebas de funcionamiento

ANEXO III: Códigos Fuente

LIBRERÍAS

```
// LIBRERIAS A UTILIZAR
#INCLUDE <WIFI.H>
#INCLUDE <ASYNCTCP.H>
#INCLUDE <ESPASYNCWEBSERVER.H>
#INCLUDE <NTPCLIENT.H>
#INCLUDE <WIFIUDP.H>
```

VARIABLES GENERALES

```
// CREDENCIALES PARA ACCESO AL INTERNET
CONST CHAR* SSID = "******";
CONST CHAR* PASSWORD = "********";

//DECLARACIÓN DE VARIABLES NECESARIAS
CONST INT LED = 4;
CONST INT LED1 = 16;
CONST INT LED2 = 17;
CONST INT LED3 = 18; //LUZ RELE

CONST CHAR* PARAM_INPUT_1 = "INPUT1";
CONST CHAR* PARAM_INPUT_2 = "INPUT2";
CONST CHAR* PARAM_INPUT_3 = "INPUT3";
CONST CHAR* PARAM_INPUT_4 = "INPUT4"; //

STRING SLIDERVALUEPRUEBA = "0";
STRING SLIDERVALUEPRUEBA1 = "0";
STRING SLIDERVALUEPRUEBA2 = "0";
STRING SLIDERVALUEPRUEBA3 = "0";


STRING SLIDERVALUE = "0";
STRING SLIDERVALUE1 = "0";
STRING SLIDERVALUE2 = "0";
STRING SLIDERVALUE3 = "0";
// VARIABLES PARA CONTROL DE BRILLO
CONST INT FREQ = 5000;
CONST INT LEDCHANNELPRUEBA = 0;
CONST INT LEDCHANNELPRUEBA1 = 1;
CONST INT LEDCHANNELPRUEBA2 = 2;
CONST INT LEDCHANNELPRUEBA3 = 3;///


CONST INT LEDCHANNEL = 0;
CONST INT LEDCHANNEL1 = 1;
CONST INT LEDCHANNEL2 = 2;
CONST INT LEDCHANNEL3 = 3;
CONST INT RESOLUTION = 8;

CONST CHAR* PARAM_INPUTPRUEBA = "VALUE";
CONST CHAR* PARAM_INPUTPRUEBA1 = "VALUE";
CONST CHAR* PARAM_INPUTPRUEBA2 = "VALUE";
CONST CHAR* PARAM_INPUTPRUEBA3 = "VALUE";

CONST CHAR* PARAM_INPUT = "VALUE";
CONST CHAR* PARAM_INPUT1 = "VALUE";
CONST CHAR* PARAM_INPUT2 = "VALUE";
CONST CHAR* PARAM_INPUT3 = "VALUE";


INT HORA1;
INT MINUTOS1;
INT HORA2;
INT MINUTOS2;
```

```
// CONFIGURA EL SERVIDOR NTP
CONST CHAR* NTPSERVER = "POOL.NTP.ORG";
CONST LONG GMTOFFSET_SEC = -5 * 3600;
CONST INT DAYLIGHTOFFSET_SEC = 0;


WIFIUDP NTPUDP;
NTPCLIENT TIMECLIENT(NTPUDP, NTPSERVER, GMTOFFSET_SEC, DAYLIGHTOFFSET_SEC);

// CREA EL SERVIDOR EN PUERTO 80
ASYNCWEBSERVER SERVER(80);


//IP FIJA
IPADDRESS IP_LOCAL(192, 168, 1, 4);
IPADDRESS GATEWAY(192, 168, 1, 1);
IPADDRESS SUBNET(255, 255, 255, 0);
```

ESTILO DEL ARCHIVO HTML

```
//ESTRUCTURA DE LA PÁGINA WEB

CONST CHAR INDEX_HTML[] PROGMEM = R"RAWLITERAL(
<!DOCTYPE HTML><HTML>
<HEAD>
  <META NAME="VIEWPORT" CONTENT="WIDTH=DEVICE-WIDTH, INITIAL-SCALE=1">
  <TITLE>ESP WEB SERVER</TITLE>
  <STYLE>
 HTML {FONT-FAMILY: ARIAL;
  DISPLAY: INLINE-BLOCK;
  TEXT-ALIGN: CENTER;  BACKGROUND-COLOR: #BFC0C2;;}
  .TOPNAV {
  FONT-FAMILY: 'COURIER NEW', COURIER, MONOSPACE;
  FONT-WEIGHT: BOLD;
  OVERFLOW: HIDDEN;
  BACKGROUND-COLOR: #4E6B89;
  COLOR: #161616;
  FONT-SIZE: 1REM;
}
}
    H1 {FONT-SIZE: 1.5REM;}
    P {FONT-SIZE: 1REM;}
      P2 {FONT-SIZE: 1REM;OPACITY:0.01;}
    .CONTENT { PADDING: 20PX;}
    BODY {MARGIN: 0;}
    .SLIDER { -WEBKIT-APPEARANCE: NONE; MARGIN:0 AUTO; WIDTH: 200PX; HEIGHT: 18PX;
BACKGROUND: #848484;
      OUTLINE: NONE; -WEBKIT-TRANSITION: .2S; TRANSITION: OPACITY .2S;BORDER-RADIUS:
10PX;}
    .SLIDER::-WEBKIT-SLIDER-THUMB {-WEBKIT-APPEARANCE: NONE; APPEARANCE: NONE;
WIDTH: 25PX; HEIGHT: 25PX; BACKGROUND: #55053F; CURSOR: POINTER;BORDER-RADIUS:
10PX;}
    .SLIDER::-MOZ-RANGE-THUMB { WIDTH: 35PX; HEIGHT: 35PX; BACKGROUND: #55053F;
CURSOR: POINTER; }

       .SLIDER1 { -WEBKIT-APPEARANCE: NONE; MARGIN: 14PX; WIDTH: 50PX; HEIGHT: 18PX;
BACKGROUND: #848484;
      OUTLINE: NONE; -WEBKIT-TRANSITION: .2S; TRANSITION: OPACITY .2S;BORDER-RADIUS:
10PX;}
    .SLIDER1::-WEBKIT-SLIDER-THUMB {-WEBKIT-APPEARANCE: NONE; APPEARANCE: NONE;
WIDTH: 25PX; HEIGHT: 25PX; BACKGROUND: #1E10E4; CURSOR: POINTER;BORDER-RADIUS:
10PX;}
    .SLIDER1::-MOZ-RANGE-THUMB { WIDTH: 35PX; HEIGHT: 35PX; BACKGROUND: #1E10E4;
CURSOR: POINTER; }
   .CARD {
  BACKGROUND-COLOR: #EAEAEB;
  PADDING-BOTTOM: 0.8CM;
```

```
  ALIGN-CONTENT: CENTER;
  ALIGN-TRACKS: CENTER;
  ALIGN-ITEMS: CENTER;
  BOX-SHADOW: 2PX 12PX 1PX RGBA(140,140,140,.5);
}
.CARD-TITLE {

  COLOR:#0D0D0D;
FONT-SIZE: LARGER;
  FONT-WEIGHT: BOLD;
FONT-FAMILY: 'GILL SANS', 'GILL SANS MT', CALIBRI, 'TREBUCHET MS', SANS-SERIF;
PADDING-TOP: 0.1CM;

}
.TOPNAV {
  OVERFLOW: HIDDEN;
  BACKGROUND-COLOR: #003366;
  COLOR: #FFD43B;
  FONT-SIZE: 1REM;
}
.CARDS {
  PADDING-BOTTOM: 0.8CM;
  ALIGN-CONTENT: CENTER;
  ALIGN-TRACKS: CENTER;
  ALIGN-ITEMS: CENTER;
  MAX-WIDTH: 1200PX;
  MARGIN: 0 AUTO;
  DISPLAY: GRID;
  GRID-GAP: 2.5REM;
  GRID-TEMPLATE-COLUMNS: REPEAT(AUTO-FIT, MINMAX(250PX,1FR));
}

.CONTAINER-CLOCK {
    TEXT-ALIGN: CENTER;
    ;
}

.CONTAINER-CLOCK H1 {
    FONT-SIZE: 4REM;
    FONT-WEIGHT: 50;
}

.CONTAINER-CLOCK P {
    FONT-SIZE: 2.5REM;
    FONT-FAMILY: VERDANA, GENEVA, TAHOMA, SANS-SERIF;
}
INPUT {
  MARGIN: 0.4REM 0;
}
  </STYLE>
</HEAD>
```

CUERPO DEL ARCHIVO HTML

```
<BODY>
 <DIV CLASS="TOPNAV">
        <H1>CONTROL DOMOTICO</H1>

    </DIV>
    <DIV CLASS="CONTENT">
<DIV CLASS="CONTAINER-CLOCK">
        <H1 ID="TIME">00:00:00</H1>
    </DIV>

<DIV CLASS="CARDS">
<DIV CLASS="CARD">
<P CLASS="CARD-TITLE">HORA DE ENCENDIDO</P>
<FORM ACTION="/GET" TARGET="HIDDEN-FORM"><BR>
   <INPUT TYPE="NUMBER"  PLACEHOLDER ="INGRESE LA HORA" NAME="INPUT1">
```

```
   <INPUT TYPE="SUBMIT" VALUE="GUARDAR">
  </FORM><BR>
  </FORM>
  <FORM ACTION="/GET1" TARGET="HIDDEN-FORM"><BR>
   <INPUT TYPE="NUMBER"  PLACEHOLDER ="INGRESE LOS MINUTOS"NAME="INPUT2">
    <INPUT TYPE="SUBMIT" VALUE="GUARDAR">
  </FORM><BR>
  </FORM>
</DIV>
<DIV CLASS = "CARD">
<P CLASS="CARD-TITLE">HORA DE APAGADO</P>
    <FORM ACTION="/GET2" TARGET="HIDDEN-FORM"><BR>
   <INPUT TYPE="NUMBER"  PLACEHOLDER ="INGRESE LA HORA" NAME="INPUT3">
    <INPUT TYPE="SUBMIT" VALUE="GUARDAR">
  </FORM><BR>
  </FORM>
    <FORM ACTION="/GET3" TARGET="HIDDEN-FORM"><BR>
  <INPUT TYPE="NUMBER" PLACEHOLDER ="INGRESE LOS MINUTOS" NAME="INPUT4" SIZE =
"0.5PX">
    <INPUT TYPE="SUBMIT" VALUE="GUARDAR">
  </FORM><BR>
  </FORM>
</DIV>
</DIV>

<DIV CLASS="CARDS">
<DIV CLASS="CARD">
            <P CLASS="CARD-TITLE">LED 1</P>
            <P CLASS="SWITCH">
               <INPUT TYPE="RANGE" ONCHANGE="UPDATESLIDERPWMPRUEBA(THIS)"
ID="PWMSLIDERPRUEBA" MIN="0" MAX="100" STEP="100" VALUE ="%SLIDERVALUEPRUEBA%"
CLASS="SLIDER1">
            </P>
             <P CLASS="STATE">OFF  ON </SPAN></P>
            <P2 CLASS="STATE">OFF  ON <SPAN
ID="TEXTSLIDERVALUEPRUEBA"></SPAN></P2>

            <P CLASS="CARD-TITLE">LED 1</P>
            <P CLASS="SWITCH">
               <INPUT TYPE="RANGE" ONCHANGE="UPDATESLIDERPWM(THIS)" ID="PWMSLIDER"
MIN="0" MAX="100" STEP="1" VALUE ="%SLIDERVALUE%" CLASS="SLIDER">
            </P>
            <P CLASS="STATE">INTENSIDAD: <SPAN
ID="TEXTSLIDERVALUE">%SLIDERVALUE%</SPAN> &PERCNT;</P>
         </DIV>
<DIV CLASS="CARD">


  <P CLASS="CARD-TITLE">LED 2</P>
            <P CLASS="SWITCH">
               <INPUT TYPE="RANGE" ONCHANGE="UPDATESLIDERPWMPRUEBA1(THIS)"
ID="PWMSLIDERPRUEBA1" MIN="0" MAX="100" STEP="100" VALUE
="%SLIDERVALUEPRUEBA1%" CLASS="SLIDER1">
            </P>
             <P CLASS="STATE">OFF  ON </SPAN></P>
            <P2 CLASS="STATE">OFF  ON <SPAN
ID="TEXTSLIDERVALUEPRUEBA1"></SPAN></P2>


            <P CLASS="CARD-TITLE">LED 2</P>
            <P CLASS="SWITCH">

               <INPUT TYPE="RANGE" ONCHANGE="UPDATESLIDERPWM1(THIS)"
ID="PWMSLIDER1" MIN="0" MAX="100" STEP="1" VALUE ="%SLIDERVALUE1%"
CLASS="SLIDER">
            </P>
            <P CLASS="STATE">INTENSIDAD: <SPAN
ID="TEXTSLIDERVALUE1">%SLIDERVALUE1%</SPAN> &PERCNT;</P>
         </DIV>
```

```
<DIV CLASS="CARD">


  <P CLASS="CARD-TITLE">LED 3</P>
            <P CLASS="SWITCH">
               <INPUT TYPE="RANGE" ONCHANGE="UPDATESLIDERPWMPRUEBA2(THIS)"
ID="PWMSLIDERPRUEBA2" MIN="0" MAX="100" STEP="100" VALUE
="%SLIDERVALUEPRUEBA2%" CLASS="SLIDER1">
            </P>
             <P CLASS="STATE">OFF  ON </SPAN></P>
            <P2 CLASS="STATE">OFF  ON <SPAN
ID="TEXTSLIDERVALUEPRUEBA2"></SPAN></P2>




            <P CLASS="CARD-TITLE">LED 3</P>
            <P CLASS="SWITCH">
               <INPUT TYPE="RANGE" ONCHANGE="UPDATESLIDERPWM2(THIS)"
ID="PWMSLIDER2" MIN="0" MAX="100" STEP="1" VALUE ="%SLIDERVALUE2%"
CLASS="SLIDER">
            </P>
            <P CLASS="STATE">INTENSIDAD: <SPAN
ID="TEXTSLIDERVALUE2">%SLIDERVALUE2%</SPAN> &PERCNT;</P>
          </DIV>
<DIV CLASS="CARD">


  <P CLASS="CARD-TITLE">LED 4</P>
            <P CLASS="SWITCH">
               <INPUT TYPE="RANGE" ONCHANGE="UPDATESLIDERPWMPRUEBA3(THIS)"
ID="PWMSLIDERPRUEBA3" MIN="0" MAX="100" STEP="100" VALUE
="%SLIDERVALUEPRUEBA3%" CLASS="SLIDER1">
            </P>
             <P CLASS="STATE">OFF  ON </SPAN></P>
            <P2 CLASS="STATE">OFF  ON <SPAN
ID="TEXTSLIDERVALUEPRUEBA3"></SPAN></P2>




            <P CLASS="CARD-TITLE">LED41</P>
            <P CLASS="SWITCH">
               <INPUT TYPE="RANGE" ONCHANGE="UPDATESLIDERPWM3(THIS)"
ID="PWMSLIDER3" MIN="0" MAX="100" STEP="1" VALUE ="%SLIDERVALUE3%"
CLASS="SLIDER">
            </P>
            <P CLASS="STATE">INTENSIDAD: <SPAN
ID="TEXTSLIDERVALUE3">%SLIDERVALUE3%</SPAN> &PERCNT;</P>
          </DIV>
          </DIV>
  </DIV>
<SCRIPT>
```

ASIGNACIÓN DE UNA CADENA DE TEXTO

```
FUNCTION UPDATESLIDERPWM(ELEMENT) {
  VAR SLIDERVALUE = DOCUMENT.GETELEMENTBYID("PWMSLIDER").VALUE;
  DOCUMENT.GETELEMENTBYID("TEXTSLIDERVALUE").INNERHTML = SLIDERVALUE;
  CONSOLE.LOG(SLIDERVALUE);
  VAR XHR = NEW XMLHTTPREQUEST();
  XHR.OPEN("GET", "/SLIDER?VALUE="+SLIDERVALUE, TRUE);
  XHR.SEND();
}

FUNCTION UPDATESLIDERPWM1(ELEMENT) {
  VAR SLIDERVALUE1 = DOCUMENT.GETELEMENTBYID("PWMSLIDER1").VALUE;
  DOCUMENT.GETELEMENTBYID("TEXTSLIDERVALUE1").INNERHTML = SLIDERVALUE1;
```

```
  CONSOLE.LOG(SLIDERVALUE1);
  VAR XHR = NEW XMLHTTPREQUEST();
  XHR.OPEN("GET", "/SLIDER1?VALUE="+SLIDERVALUE1, TRUE);
  XHR.SEND();
}

FUNCTION UPDATESLIDERPWM2(ELEMENT) {
  VAR SLIDERVALUE2 = DOCUMENT.GETELEMENTBYID("PWMSLIDER2").VALUE;
  DOCUMENT.GETELEMENTBYID("TEXTSLIDERVALUE2").INNERHTML = SLIDERVALUE2;
  CONSOLE.LOG(SLIDERVALUE2);
  VAR XHR = NEW XMLHTTPREQUEST();
  XHR.OPEN("GET", "/SLIDER2?VALUE="+SLIDERVALUE2, TRUE);
  XHR.SEND();
}

FUNCTION UPDATESLIDERPWM3(ELEMENT) {
  VAR SLIDERVALUE3 = DOCUMENT.GETELEMENTBYID("PWMSLIDER3").VALUE;
  DOCUMENT.GETELEMENTBYID("TEXTSLIDERVALUE3").INNERHTML = SLIDERVALUE3;
  CONSOLE.LOG(SLIDERVALUE3);
  VAR XHR = NEW XMLHTTPREQUEST();
  XHR.OPEN("GET", "/SLIDER3?VALUE="+SLIDERVALUE3, TRUE);
  XHR.SEND();
}

FUNCTION UPDATESLIDERPWMPRUEBA(ELEMENT) {
  VAR SLIDERVALUEPRUEBA = DOCUMENT.GETELEMENTBYID("PWMSLIDERPRUEBA").VALUE;
  DOCUMENT.GETELEMENTBYID("TEXTSLIDERVALUEPRUEBA").INNERHTML =
SLIDERVALUEPRUEBA;
  CONSOLE.LOG(SLIDERVALUEPRUEBA);
  VAR XHR = NEW XMLHTTPREQUEST();
  XHR.OPEN("GET", "/SLIDERPRUEBA?VALUE="+SLIDERVALUEPRUEBA, TRUE);
  XHR.SEND();
}

FUNCTION UPDATESLIDERPWMPRUEBA1(ELEMENT) {
  VAR SLIDERVALUEPRUEBA1 =
DOCUMENT.GETELEMENTBYID("PWMSLIDERPRUEBA1").VALUE;
  DOCUMENT.GETELEMENTBYID("TEXTSLIDERVALUEPRUEBA1").INNERHTML =
SLIDERVALUEPRUEBA1;
  CONSOLE.LOG(SLIDERVALUEPRUEBA1);
  VAR XHR = NEW XMLHTTPREQUEST();
  XHR.OPEN("GET", "/SLIDERPRUEBA1?VALUE="+SLIDERVALUEPRUEBA1, TRUE);
  XHR.SEND();
}

FUNCTION UPDATESLIDERPWMPRUEBA2(ELEMENT) {
  VAR SLIDERVALUEPRUEBA2 =
DOCUMENT.GETELEMENTBYID("PWMSLIDERPRUEBA2").VALUE;
  DOCUMENT.GETELEMENTBYID("TEXTSLIDERVALUEPRUEBA2").INNERHTML =
SLIDERVALUEPRUEBA2;
  CONSOLE.LOG(SLIDERVALUEPRUEBA2);
  VAR XHR = NEW XMLHTTPREQUEST();
  XHR.OPEN("GET", "/SLIDERPRUEBA2?VALUE="+SLIDERVALUEPRUEBA2, TRUE);
  XHR.SEND();
}

FUNCTION UPDATESLIDERPWMPRUEBA3(ELEMENT) {
  VAR SLIDERVALUEPRUEBA3 =
DOCUMENT.GETELEMENTBYID("PWMSLIDERPRUEBA3").VALUE;
  DOCUMENT.GETELEMENTBYID("TEXTSLIDERVALUEPRUEBA3").INNERHTML =
SLIDERVALUEPRUEBA3;
  CONSOLE.LOG(SLIDERVALUEPRUEBA3);
  VAR XHR = NEW XMLHTTPREQUEST();
  XHR.OPEN("GET", "/SLIDERPRUEBA3?VALUE="+SLIDERVALUEPRUEBA3, TRUE);
  XHR.SEND();
}

CONST TIME = DOCUMENT.GETELEMENTBYID('TIME');
CONST INTERVAL = SETINTERVAL(() => {
```

```
    CONST LOCAL = NEW DATE();
    TIME.INNERHTML = LOCAL.TOLOCALETIMESTRING();
}, 1000);
</SCRIPT>
</BODY>
</HTML>
)RAWLITERAL";


//ALERTA CUANDO NO SE CONECTE
VOID NOTFOUND(ASYNCWEBSERVERREQUEST* REQUEST) {
  REQUEST->SEND(404, "TEXT/PLAIN", "NOT FOUND");
}

// DEFINE EL ESTADO DE LOS BOTONES
STRING PROCESSOR(CONST STRING& VAR) {
  //SERIAL.PRINTLN(VAR);
  IF (VAR == "SLIDERVALUE") {
    RETURN SLIDERVALUE;
  } ELSE IF (VAR == "SLIDERVALUE1") {
    RETURN SLIDERVALUE1;
  } ELSE IF (VAR == "SLIDERVALUE2") {
    RETURN SLIDERVALUE2;
  } ELSE IF (VAR == "SLIDERVALUE3") {
    RETURN SLIDERVALUE3;
  } ELSE IF (VAR == "SLIDERVALUEPRUEBA") {
    RETURN SLIDERVALUEPRUEBA;
  } ELSE IF (VAR == "SLIDERVALUEPRUEBA1") {
    RETURN SLIDERVALUEPRUEBA1;
  } ELSE IF (VAR == "SLIDERVALUEPRUEBA2") {
    RETURN SLIDERVALUEPRUEBA2;
  } ELSE IF (VAR == "SLIDERVALUEPRUEBA3") {
    RETURN SLIDERVALUEPRUEBA3;
  }
  RETURN STRING();
}


VOID SETUP() {

  SERIAL.BEGIN(115200);


  // DECLARACIÓN DE SALIDAS DE LEDS CON REGULADOR DE POTENCIA
  LEDCSETUP(LEDCHANNEL, FREQ, RESOLUTION);
  LEDCSETUP(LEDCHANNEL1, FREQ, RESOLUTION);
  LEDCSETUP(LEDCHANNEL2, FREQ, RESOLUTION);
  LEDCSETUP(LEDCHANNEL3, FREQ, RESOLUTION);




  LEDCATTACHPIN(LED, LEDCHANNELPRUEBA);
  LEDCATTACHPIN(LED1, LEDCHANNELPRUEBA1);
  LEDCATTACHPIN(LED2, LEDCHANNELPRUEBA2);
  LEDCATTACHPIN(LED3, LEDCHANNELPRUEBA3);


  LEDCATTACHPIN(LED, LEDCHANNEL);
  LEDCATTACHPIN(LED1, LEDCHANNEL1);
  LEDCATTACHPIN(LED2, LEDCHANNEL2);
  LEDCATTACHPIN(LED3, LEDCHANNEL3);

  LEDCWRITE(LEDCHANNELPRUEBA, SLIDERVALUEPRUEBA.TOINT());
  LEDCWRITE(LEDCHANNELPRUEBA1, SLIDERVALUEPRUEBA1.TOINT());
  LEDCWRITE(LEDCHANNELPRUEBA2, SLIDERVALUEPRUEBA2.TOINT());
  LEDCWRITE(LEDCHANNELPRUEBA3, SLIDERVALUEPRUEBA3.TOINT());

  LEDCWRITE(LEDCHANNEL, SLIDERVALUE.TOINT());
```

```
LEDCWRITE(LEDCHANNEL1, SLIDERVALUE1.TOINT());
LEDCWRITE(LEDCHANNEL2, SLIDERVALUE2.TOINT());
LEDCWRITE(LEDCHANNEL3, SLIDERVALUE3.TOINT());


// CONEXIÓN AL WI-FI
WIFI.BEGIN(SSID, PASSWORD);
WHILE (WIFI.STATUS() != WL_CONNECTED) {
  DELAY(1000);
  SERIAL.PRINTLN("CONECTADO A LA RED WIFI");
}



SERIAL.PRINTLN(WIFI.LOCALIP());

// CREA LA PÁGINA WEB
SERVER.ON("/", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
  REQUEST->SEND_P(200, "TEXT/HTML", INDEX_HTML, PROCESSOR);
});
```

ASIGNACIÓN DE ACCIONES A CADA CADENA DE TEXTO

```
SERVER.ON("/", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
  REQUEST->SEND_P(200, "TEXT/HTML", INDEX_HTML);
});
// ENVIA UNA RESPUESTA
SERVER.ON("/SLIDER", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
  STRING INPUTMESSAGE;
  // GET INPUT1 VALUE ON <ESP_IP>/SLIDER?VALUE=<INPUTMESSAGE>
  IF (REQUEST->HASPARAM(PARAM_INPUT)) {
    INPUTMESSAGE = REQUEST->GETPARAM(PARAM_INPUT)->VALUE();
    SLIDERVALUE = INPUTMESSAGE;
    LEDCWRITE(LEDCHANNEL, SLIDERVALUE.TOINT());
  } ELSE {
    INPUTMESSAGE = "NO MESSAGE SENT";
  }
  SERIAL.PRINTLN(INPUTMESSAGE);
  REQUEST->SEND(200, "TEXT/PLAIN", "OK");
});

SERVER.ON("/SLIDER1", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
  STRING INPUTMESSAGE1;

  IF (REQUEST->HASPARAM(PARAM_INPUT1)) {
    INPUTMESSAGE1 = REQUEST->GETPARAM(PARAM_INPUT1)->VALUE();
    SLIDERVALUE1 = INPUTMESSAGE1;
    PINMODE(4, OUTPUT);
    LEDCWRITE(LEDCHANNEL1, SLIDERVALUE1.TOINT());
  } ELSE {
    INPUTMESSAGE1 = "NO MESSAGE SENT";
  }
  SERIAL.PRINTLN(INPUTMESSAGE1);
  REQUEST->SEND(200, "TEXT/PLAIN", "OK");
});


SERVER.ON("/SLIDER2", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
  STRING INPUTMESSAGE2;
  // GET INPUT1 VALUE ON <ESP_IP>/SLIDER?VALUE=<INPUTMESSAGE>
  IF (REQUEST->HASPARAM(PARAM_INPUT2)) {
    INPUTMESSAGE2 = REQUEST->GETPARAM(PARAM_INPUT2)->VALUE();
    SLIDERVALUE2 = INPUTMESSAGE2;
    LEDCWRITE(LEDCHANNEL2, SLIDERVALUE2.TOINT());
  } ELSE {
    INPUTMESSAGE2 = "NO MESSAGE SENT";
  }
  SERIAL.PRINTLN(INPUTMESSAGE2);
  REQUEST->SEND(200, "TEXT/PLAIN", "OK");
});
```

```
  SERVER.ON("/SLIDER3", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
    STRING INPUTMESSAGE3;

    IF (REQUEST->HASPARAM(PARAM_INPUT3)) {
      INPUTMESSAGE3 = REQUEST->GETPARAM(PARAM_INPUT3)->VALUE();
      SLIDERVALUE3 = INPUTMESSAGE3;
      LEDCWRITE(LEDCHANNEL3, 0);
     SLIDERVALUE3.TOINT()
    } ELSE {
      INPUTMESSAGE3 = "NO MESSAGE SENT";
    }
    SERIAL.PRINTLN(INPUTMESSAGE3);
    REQUEST->SEND(200, "TEXT/PLAIN", "OK");
  });

  SERVER.ON("/SLIDERPRUEBA", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
    STRING INPUTMESSAGEPRUEBA;

 // ENTRADA 1

    IF (REQUEST->HASPARAM(PARAM_INPUTPRUEBA)) {
      INPUTMESSAGEPRUEBA = REQUEST->GETPARAM(PARAM_INPUTPRUEBA)->VALUE();
      SLIDERVALUEPRUEBA = INPUTMESSAGEPRUEBA;
      LEDCWRITE(LEDCHANNELPRUEBA, SLIDERVALUEPRUEBA.TOINT());
    } ELSE {
      INPUTMESSAGEPRUEBA = "NO MESSAGE SENT";
    }
    SERIAL.PRINTLN(INPUTMESSAGEPRUEBA);
    REQUEST->SEND(200, "TEXT/PLAIN", "OK");
  });

  SERVER.ON("/SLIDERPRUEBA1", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
    STRING INPUTMESSAGEPRUEBA1;

    IF (REQUEST->HASPARAM(PARAM_INPUTPRUEBA1)) {
      INPUTMESSAGEPRUEBA1 = REQUEST->GETPARAM(PARAM_INPUTPRUEBA1)->VALUE();
      SLIDERVALUEPRUEBA1 = INPUTMESSAGEPRUEBA1;
      LEDCWRITE(LEDCHANNELPRUEBA1, SLIDERVALUEPRUEBA1.TOINT());
    } ELSE {
      INPUTMESSAGEPRUEBA1 = "NO MESSAGE SENT";
    }
    SERIAL.PRINTLN(INPUTMESSAGEPRUEBA1);
    REQUEST->SEND(200, "TEXT/PLAIN", "OK");
  });


  SERVER.ON("/SLIDERPRUEBA2", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
    STRING INPUTMESSAGEPRUEBA2;
    // ENTRADA 2

    IF (REQUEST->HASPARAM(PARAM_INPUTPRUEBA2)) {
      INPUTMESSAGEPRUEBA2 = REQUEST->GETPARAM(PARAM_INPUTPRUEBA2)->VALUE();
      SLIDERVALUEPRUEBA2 = INPUTMESSAGEPRUEBA2;
      LEDCWRITE(LEDCHANNELPRUEBA2, SLIDERVALUEPRUEBA2.TOINT());
    } ELSE {
      INPUTMESSAGEPRUEBA2 = "NO MESSAGE SENT";
    }
    SERIAL.PRINTLN(INPUTMESSAGEPRUEBA2);
    REQUEST->SEND(200, "TEXT/PLAIN", "OK");
  });

  SERVER.ON("/SLIDERPRUEBA3", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
    STRING INPUTMESSAGEPRUEBA3;
    // ENTREADA 3
    IF (REQUEST->HASPARAM(PARAM_INPUTPRUEBA3)) {
      INPUTMESSAGEPRUEBA3 = REQUEST->GETPARAM(PARAM_INPUTPRUEBA3)->VALUE();
      SLIDERVALUEPRUEBA3 = INPUTMESSAGEPRUEBA3;
```

```
      LEDCWRITE(LEDCHANNELPRUEBA3, 255);
    } ELSE {
      INPUTMESSAGEPRUEBA3 = "NO MESSAGE SENT";
      //LEDCWRITE(LEDCHANNELPRUEBA3, 0);
    }
    SERIAL.PRINTLN(INPUTMESSAGEPRUEBA3);
    REQUEST->SEND(200, "TEXT/PLAIN", "OK");
  });

  // ENTRADA 4
  SERVER.ON("/GET", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
    STRING INPUTMESSAGE4;
    STRING INPUTPARAM4;

    IF (REQUEST->HASPARAM(PARAM_INPUT_1)) {
      INPUTMESSAGE4 = REQUEST->GETPARAM(PARAM_INPUT_1)->VALUE();
      INPUTPARAM4 = PARAM_INPUT_1;
    } ELSE {
      INPUTMESSAGE4 = "NO MESSAGE SENT";
      INPUTPARAM4 = "NONE";
    }
    SERIAL.PRINTLN(INPUTMESSAGE4);
    REQUEST->SEND(200, "TEXT/HTML", "<BR><CENTER><H1>HORA GUARDADA
CORRECTAMENTE (" + INPUTPARAM4 + ") HORA DE INICIO : " + INPUTMESSAGE4 +
"<BR><BR><A HREF=\"/\">PAGINA PRINCIPAL</A></H1>");

    HORA1 = INPUTMESSAGE4.TOINT();
    SERIAL.PRINT("HORA: ");
    SERIAL.PRINTLN(HORA1);
  });


  SERVER.ON("/GET1", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
    STRING INPUTMESSAGE5;
    STRING INPUTPARAM5;

    IF (REQUEST->HASPARAM(PARAM_INPUT_2)) {
      INPUTMESSAGE5 = REQUEST->GETPARAM(PARAM_INPUT_2)->VALUE();
      INPUTPARAM5 = PARAM_INPUT_2;
    } ELSE {
      INPUTMESSAGE5 = "NO MESSAGE SENT";
      INPUTPARAM5 = "NONE";
    }

    SERIAL.PRINTLN(INPUTMESSAGE5);
    REQUEST->SEND(200, "TEXT/HTML", "<BR><CENTER><H1>HORA GUARDADA
CORRECTAMENTE (" + INPUTPARAM5 + ") HORA DE INICIO : " + INPUTMESSAGE5 +
"<BR><BR><A HREF=\"/\">PAGINA PRINCIPAL</A></H1>");
    MINUTOS1 = INPUTMESSAGE5.TOINT();
    SERIAL.PRINT("MINUTIS 1 : ");
    SERIAL.PRINTLN(MINUTOS1);
  });

  SERVER.ON("/GET2", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
    STRING INPUTMESSAGE6;
    STRING INPUTPARAM6;

    IF (REQUEST->HASPARAM(PARAM_INPUT_3)) {
      INPUTMESSAGE6 = REQUEST->GETPARAM(PARAM_INPUT_3)->VALUE();
      INPUTPARAM6 = PARAM_INPUT_3;
    } ELSE {
      INPUTMESSAGE6 = "NO MESSAGE SENT";
      INPUTPARAM6 = "NONE";
    }

    SERIAL.PRINTLN(INPUTMESSAGE6);
    REQUEST->SEND(200, "TEXT/HTML", "<BR><CENTER><H1>HORA GUARDADA
CORRECTAMENTE (" + INPUTPARAM6 + ") HORA DE INICIO : " + INPUTMESSAGE6 +
"<BR><BR><A HREF=\"/\">PAGINA PRINCIPAL</A></H1>");
```

```
    HORA2 = INPUTMESSAGE6.TOINT();
    SERIAL.PRINT("HORA: ");
    SERIAL.PRINTLN(HORA2);
  });

  SERVER.ON("/GET3", HTTP_GET, [](ASYNCWEBSERVERREQUEST* REQUEST) {
    STRING INPUTMESSAGE7;
    STRING INPUTPARAM7;

    IF (REQUEST->HASPARAM(PARAM_INPUT_4)) {
      INPUTMESSAGE7 = REQUEST->GETPARAM(PARAM_INPUT_4)->VALUE();
      INPUTPARAM7 = PARAM_INPUT_4;
    } ELSE {
      INPUTMESSAGE7 = "NO MESSAGE SENT";
      INPUTPARAM7 = "NONE";
    }

    SERIAL.PRINTLN(INPUTMESSAGE7);
    REQUEST->SEND(200, "TEXT/HTML", "<BR><CENTER><H1>HORA GUARDADA
CORRECTAMENTE (" + INPUTPARAM7 + ") HORA DE INICIO : " + INPUTMESSAGE7 +
"<BR><BR><A HREF=\"/\">PAGINA PRINCIPAL</A></H1>");
    MINUTOS2 = INPUTMESSAGE7.TOINT();
    SERIAL.PRINT("HORA: ");
    SERIAL.PRINTLN(MINUTOS2);
  });

  SERVER.ONNOTFOUND(NOTFOUND);


  SERVER.BEGIN();
}
```

PROGRAMCIÓN DEL ENCENDIDO UTILIZANDO UNA VARIABLE HORARIA

```
VOID LOOP() {

  // OBTIENE LA HORA ACTUAL
  TIMECLIENT.UPDATE();
  INT CURRENTHOUR = TIMECLIENT.GETHOURS();
  INT CURRENTMINUTE = TIMECLIENT.GETMINUTES();
  INT CURRENTSECOND = TIMECLIENT.GETSECONDS();

  SERIAL.PRINT("HORA: ");
  SERIAL.PRINTLN(CURRENTHOUR);

  SERIAL.PRINT("MINUTOS: ");
  SERIAL.PRINTLN(CURRENTMINUTE);

  DELAY(1000);

// CASOS DE ENCENDIDO POR HORAS
  IF (HORA1 == CURRENTHOUR && MINUTOS1 == CURRENTMINUTE) {
    LEDCWRITE(LEDCHANNEL, 255);
    LEDCWRITE(LEDCHANNEL1, 255);
    LEDCWRITE(LEDCHANNEL2, 255);
    LEDCWRITE(LEDCHANNEL3, 255);
  }
  IF (HORA2 == CURRENTHOUR && MINUTOS2 == CURRENTMINUTE) {
    LEDCWRITE(LEDCHANNEL, 0);
    LEDCWRITE(LEDCHANNEL1, 0);
    LEDCWRITE(LEDCHANNEL2, 0);
    LEDCWRITE(LEDCHANNEL3, 0);
  }
}
```