ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA DETECCIÓN DE RADIACIÓN UV A TRAVÉS DEL PROTOCOLO ESP-NOW EN UN DASHBOARD WEB

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

KEVIN BRYAN COELLO NUÑEZ

kevin.coello@epn.edu.ec

DIRECTOR: ING. CARLOS ANDRÉS YUNGA SÁNCHEZ

carlos.yunga@epn.edu.ec

DMQ, 25 JULIO DEL 2024

CERTIFICACIONES

Yo, KEVIN BRYAN COELLO NUÑEZ 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.

Kevin Bryan Coello Nuñez
kevin.coello@epn.edu.ec

coellokevin908@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por KEVIN BRYAN COELLO NUÑEZ, bajo mi supervisión.

Ing. 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.

KEVIN BRYAN COELLO NUÑEZ

C.I: 1751505882

DEDICATORIA

Deseo dedicar mi trabajo de tesis a algunas personas que han estado presente durante toda mi vida, quienes han sido mi inspiración para poder llegar a realizar este proyecto.

La primera persona a la quiero agradecer es a mi mamá, quien con su profundo amor, ayuda incondicional y sacrificio constante ha sido mi mayor orgullo de inspiración. Le doy las gracias a mi mamá por siempre creer en mi potencial, por enseñarme valores y ser la persona que está siempre detrás de mí como soporte para no rendirme antes las adversidades.

También quiero dedicar mi tesis a mi abuelita Clara Núñez, cuyo afecto, apoyo, risas, lecciones y momentos han hecho que mi corazón siempre esté con ella. Le agradezco, por siempre consentirme, llenarme de sonrisas, complacerme con mis comidas favoritas. Sin duda alguna, tu influencia ha sido una guía constante en mi vida, y siempre te recordaré como la persona más especial que existe en mi vida.

En otro punto, pero no menos importante, son a mis profesores y tutores que han estado inculcándome conocimiento y sabiduría con la mejor paciencia que se puede tener a lo largo de mi estancia en la Universidad. De una manera especial, le doy a gracias a mi tutor de trabajo de integración curricular el Ing. Carlos Andrés Yunga Sánchez quien con su conocimiento y consejos ha sabido guiarme para terminar este último trabajo.

Finalmente, dedico este trabajo a mi enamorada Sandra Mora quien han contribuido de alguna u otra manera en mi desarrollo personal como profesional. Cada afecto, consejo y muestra de apoyo han servido para la construcción de mi proyecto.

ÍNDICE DE CONTENIDOS

RESUMEN

Este trabajo de integración curricular aborda el desarrollo y la implementación de un prototipo para la detección de radiación ultravioleta (UV), utilizando el protocolo ESP-NOW y una interfaz web basada en Arduino IoT Cloud. El objetivo central del proyecto es crear un sistema para la medición y visualización en tiempo real de los niveles de radiación UV. El prototipo se apoya en un sensor de radiación UV para captar las lecturas de medición, que luego se transmiten al microcontrolador ESP32 a través del protocolo ESP-NOW. Este microcontrolador procesa los datos recibidos y los envía a un servidor en la nube para su visualización.

El sistema se compone de dos elementos principales: el sensor de radiación UV y la comunicación de las placas ESP32 mediante el protocolo SP. La construcción y el diseño del prototipo se realizan sobre una placa de baquelita, mientras que la programación del microcontrolador se efectúa en el entorno de Arduino IDE. y la información se visualiza en el dashboard de Arduino IoT Cloud. Este dashboard presenta datos en tiempo real y utiliza un código de colores para mostrar los niveles de radiación UV: verde para niveles bajos, naranja para niveles moderados y rojo para niveles altos.

Los resultados obtenidos demuestran la medición y visualización de la radiación UV, cumpliendo con los objetivos del proyecto y validando la viabilidad del uso de tecnologías de comunicación inalámbrica y plataformas en la nube para el monitoreo ambiental.

PALABRAS CLAVE: Radiación UV, sensor, ESP-NOW, ESP32, Arduino IoT Cloud.

ABSTRACT

This curricular integration work addresses the development and implementation of a prototype for the detection of ultraviolet (UV) radiation, using the ESP-NOW protocol and a web interface based on Arduino IoT Cloud. The central objective of the project is to create a system for real-time measurement and visualization of UV radiation levels. The prototype relies on a UV radiation sensor to capture measurement readings, which are then transmitted to the ESP32 microcontroller via the ESP-NOW protocol. This microcontroller processes the received data and sends it to a server in the cloud for visualization.

The system consists of two main elements: the UV radiation sensor and the communication of the ESP32 boards through the SP protocol. The construction and design of the prototype is done on a Bakelite board, while the programming of the microcontroller is done in the Arduino IDE environment. and the information is visualized on the Arduino IoT Cloud dashboard. This dashboard presents real-time data and uses color coding to display UV radiation levels: green for low levels, orange for moderate levels, and red for high levels.

The results obtained demonstrate the measurement and visualization of UV radiation, fulfilling the project objectives and validating the feasibility of using wireless communication technologies and cloud platforms for environmental monitoring.

KEYWORDS: *UV radiation, sensor, ESP-NOW, ESP32, Arduino IoT Cloud*

1. DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En las últimas décadas, se ha registrado un notable incremento en la incidencia de cáncer de piel, una tendencia que está estrechamente relacionada con una mayor exposición a la radiación ultravioleta (UV). Esta exposición proviene tanto del sol durante actividades al aire libre como de fuentes artificiales, como lámparas y camas solares. La exposición excesiva a la radiación UV no solo es un factor clave en el desarrollo del cáncer de piel, sino que también está asociada con efectos adversos en los ojos y el sistema inmunológico.

En respuesta a esta problemática, el presente proyecto tiene como objetivo desarrollar un prototipo para medir la radiación ultravioleta utilizando el protocolo ESP-NOW con dos placas ESP32. La información obtenida se transmitirá a un dashboard web, donde se visualizará a través de una pantalla con código de colores que indica los niveles de radiación UV.

El modelo se fundamenta en el desarrollo de un dispositivo para la medición y visualización de la radiación UV, utilizando el protocolo ESP-NOW para la comunicación entre los módulos. El sensor GUVA-S12SD se integra con una unidad de adquisición de datos que lee los niveles de radiación y transmite la información a un módulo receptor. Este módulo receptor se conecta a un servidor web que despliega los datos en una interfaz gráfica, ofreciendo una visualización en tiempo real del monitoreo de la radiación UV.

El prototipo consta de dos procesos principales: la medición de radiación UV y su visualización. La parte de medición está compuesta por dos placas, una emisora y otra receptora, junto con un sensor UV. La placa emisora recibe los datos medidos del sensor y los transmite a la placa receptora, que los recoge y procesa. La placa receptora se conecta a la red externa, procesa la información recibida y la almacena para su posterior visualización en una pantalla.

El protocolo ESP-NOW es fundamental para garantizar una comunicación robusta y eficiente entre la placa emisora y la placa receptora, facilitando la transmisión de datos al servidor web.

1.1 Objetivo general

Implementar un prototipo para la detección de radiación UV a través del protocolo ESP-NOW en un dashboard web.

1.2 Objetivos específicos

- Identificar los requerimientos para el diseño del prototipo.
- Seleccionar el hardware y software acorde a los requerimientos establecidos.
- Diseñar el prototipo que detecte los niveles de radiación UV.
- Implementar el prototipo que detecte los niveles de radiación UV.
- Realizar pruebas de funcionamiento del prototipo.

1.3 Alcance

El alcance del proyecto que abarca la implementación de un prototipo que detecte los niveles de radiación UV contiene las siguientes funcionalidades:

- Detección de niveles de radiación UV.
- Establecer la comunicación inalámbrica entre 2 tarjetas de desarrollo con el ESP-NOW.
- Desarrollo de aplicativo web para que el usuario pueda leer los datos con mayor facilidad.

1.4 Marco Teórico

Radiación Ultravioleta

La radiación ultravioleta o conocida por sus siglas en inglés (UV) es una forma de energía que se encuentra caracterizado por la radiación dentro del espectro electromagnético. Su principal forma de propagación o fuente de iniciación es el sol, sin embargo, existe otras maneras de producirla mediante la creación de dispositivos como lámparas UV, de bronceado y aparatos de medicina [1].

La radiación ultravioleta al encontrarse en longitudes de ondra entre 100 y 290nm, pueden alterar moléculas en nuestros organismos de manera no beneficiosa. Por tanto, la protección de la piel contra la radiación ultravioleta es esencial para no presentar resultados negativos como arrugas, caída de cabello, ampollas o el crecimiento de un tumor maligno [1].

Radiación UV-A

Este tipo de radiación es la más cercana al espectro visible, ya que tiene una longitud de onda mayor que otros tipos de radiación. Su longitud de onda varía entre 320 nm y 400 nm, y es responsable de broncear la piel. Aunque generalmente se considera inofensiva y no representa un peligro inmediato para las personas, puede causar envejecimiento prematuro de la piel e incluso aumentar el riesgo de desarrollar cáncer de piel [2].

Radiación UV-B

La radiación UV-B es aquella que no alcanza a llegar de manera directa hasta la naturaleza debido a que se tiene a la capa de ozono con un filtro de protección. Aunque se encuentre parcialmente filtrada, aquellos rayos que si llegan hasta la tierra puede hacer que las personas expuestas lleguen a desarrollar daños en la piel, mientras que en las plantas provoca una falta de desarrollo tanto de tallo como de hoja [3].

Radiación UV-C

La radiación UV-C se considera que los rayos de esta radiación son las que más realizan daño a la piel por su alta energía. No obstante, los rayos UV-C no llegan a tierra debido a que son absorbidos por el oxígeno y el ozono en la estratosfera [4].

Escala de la radiación UV

Se presenta una estandarización del índice UV diario para la radiación solar, basado en un código de cinco colores determinado por la Organización Mundial de la Salud (OMS). El primer rango de índice UV es de 0 a 3, donde no existe riesgo de quemaduras solares. El siguiente índice, de 4 a 5, sugiere tomar precauciones como usar ropa protectora, gorro y gafas solares. El índice de 6 a 7 indica una radiación UV relativamente alta, por lo que se recomienda, además de las gafas de sol y el sombrero, aplicar protector solar en el cuerpo [5].

El índice muy alto, que va de 8 a 10, generalmente se presenta entre las 11 a.m. y las 5 p.m. En este caso, es crucial usar todos los elementos mencionados anteriormente, incluyendo ropa de manga larga, y aplicar protector solar con un factor de protección solar (FPS) de 50+. Se debe evitar la exposición prolongada al sol [5].

Finalmente, un índice UV superior a 11 señala una radiación UV extremadamente alta, por lo que se debe evitar salir. Si es necesario hacerlo, se recomienda cubrir la piel lo más posible, usar gafas de sol, sombrilla, protector solar con FPS 50+ y un sombrero. La Figura 1.1 ilustra de manera gráfica los colores correspondientes al índice UV y los elementos de protección recomendados según el nivel de radiación [5].

Color	Riesgo	IUV	SUGERENCIA
Verde	**Muy Bajo**	0-3	Puede permanecer en el exterior sin riesgo
Verde	**Bajo**	4-5	
Amarillo	**Moderado**	6-7	
Naranja	**Alto**	8-10	
Rojo	**Muy Alto**	11-15	

Figura 1.1 Solmáforo [5].

ESP32 devKit V1

La placa de desarrollo ESP32 DevKit V1 es posiblemente la más conocida y utilizada dentro de la categoría de SoC Wi-Fi ESP32 desarrollados por la empresa Espressif, como se muestra en la figura Figura 1.2. Su popularidad entre la comunidad de desarrolladores ha superado a cualquier otra placa de desarrollo similar [6].

Es un sistema de un solo chip (SoC) que se almacena por su bajo consumo de energía y costo accesible. Entre sus principales características destaca la conectividad inalámbrica mediante tecnología Wi-Fi y Bluetooth. Este chip en desarrollo integra varios componentes avanzados, como un balun RF; un tipo de transformador que convierte señales de línea balanceadas en desequilibradas y viceversa, interruptores de antena, amplificadores de potencia, amplificadores de recepción de bajo ruido, filtros y módulos de gestión de energía. Su principal aplicación es para dispositivos móviles, aplicaciones de IoT y electrónica portátil [6].

El bajo consumo de energía del ESP32 se logra gracias a diversas funciones de ahorro de energía, como el escalado dinámico de potencia, múltiples modos de energía y la sincronización de reloj de resolución final [7].

Además de estas características, el ESP32 ofrece una gran flexibilidad y versatilidad, lo que lo hace ideal para el desarrollo en la comunidad de electrónica. Su capacidad para integrarse en cualquier proyecto de electrónica ha generado una gran comunidad que proporciona soporte técnico y soluciones a problemas. Esta combinación de

características ha consolidado al ESP32 como una herramienta esencial en el desarrollo de soluciones innovadoras en el campo de la tecnología [7].

Figura 1.2 Placa ESP32 DevKit V1 [7]

ESP-NOW

Actualmente, hay diversos protocolos de comunicación inalámbrica para conectar dispositivos y sistemas en redes inteligentes, incluyendo Wi-Fi, Li-Fi, Bluetooth y Zigbee, cada uno adecuado para diferentes entornos. Espressif ha desarrollado el protocolo ESP-NOW para sus microcontroladores ESP32 y ESP8266. Este protocolo, diseñado por la misma empresa que creó estos chips, opera en la banda de 2.4 GHz y se destaca por su bajo consumo de energía. ESP-NOW ofrece comunicación eficiente y de baja latencia, siendo ideal para aplicaciones IoT que requieren rápida transmisión de datos y eficiencia energética. Su robustez y economía lo han hecho popular en proyectos de IoT [8].

Arduino IoT Cloud

En el contexto de la integración entre placas Arduino y el Internet de las Cosas (IoT), Arduino ha desarrollado una solución avanzada denominada Arduino IoT Cloud. Esta plataforma no solo proporciona hardware y firmware, sino también servicios basados en la nube y soporte técnico integral. Arduino IoT Cloud permite crear dashboards personalizados que muestran información en tiempo real desde los dispositivos conectados, facilitando así el monitoreo y la gestión de proyectos IoT [9].

Arduino IoT Cloud ofrece diversos métodos de interacción, como API REST HTTP, MQTT, herramientas de línea de comandos, Javascript y Websockets, entre otros. Esto permite una flexibilidad considerable en la forma en que los dispositivos IoT se comunican y gestionan los datos. La combinación de dashboards gráficos y la versatilidad de Arduino IoT Cloud proporciona una poderosa herramienta para visualizar y controlar el rendimiento de dispositivos y sistemas IoT, optimizando así las operaciones y mejorando la toma de decisiones basadas en datos en tiempo real [9].

En resumen, se trata un aplicativo que brinda pantalla que visualiza de manera gráfica los datos más relevantes, proporcionando una visión general en tiempo real sobre la efectividad de las acciones implementadas [9].

Sensor GUVA-S12SD

El módulo GUVA-S12SD que se muestra en la Figura 1.3 es un sensor de radiación UV que utiliza un fotodiodo tipo Schottky para realizar sus mediciones. Este módulo trabaja mediante un sistema analógico proporcional a la radiación UV detectada, lo que lo hace útil en el campo de monitoreo del índice UV para aplicaciones de cuidado de la piel y aplicaciones meteorológicas [10].

Se caracteriza por su bajo consumo energético, con un voltaje de alimentación entre 2.5 y 5V, y una corriente de funcionamiento en microamperios. Detecta luz con una longitud de onda entre 240 y 370 nm, cubriendo tanto el espectro UV-B como el UV-A. La señal analógica que genera está linealmente relacionada con la intensidad UV (mW/cm^2), y puede ser conectada a un microcontrolador para su conversión mediante un ADC y posterior procesamiento [10].

El sensor destaca por su buena linealidad, alta sensibilidad y estabilidad, ofreciendo un amplio rango de detección con un ángulo de 130 grados. La salida del sensor está en microamperios, y el módulo incorpora un amplificador cuya salida se calcula como Vo=4.3V*(Intensidad en µA), permitiendo determinar con precisión la intensidad UV detectada. Este sensor es aplicable en probadores UV, relojes UV, equipos deportivos para exteriores, teléfonos móviles y más [10].

Figura 1.3 Sensor GUVA-S12SD [10].

2 METODOLOGÍA

El proceso para el desarrollo del presente trabajo de integración curricular consta de cinco fases esenciales, basadas en los objetivos planteados inicialmente para implementar un prototipo de detección de radiación UV a través del protocolo ESP-NOW en un dashboard web. Estas fases se muestran en la Figura 2.1.

Figura 2.1 Procedimiento del prototipo.

Como se estableció al inicio del proyecto, se definió su alcance, lo que permitió establecer las características esenciales a tener en cuenta para su diseño y desarrollo. Esta fase fue crucial para comenzar a trabajar con los objetivos planteados. Para abordar los diferentes requerimientos del desarrollo, se realizó una búsqueda de información en fuentes confiables para identificar los requerimientos mínimos para el desarrollo del prototipo.

Posteriormente, con las características del prototipo definidas, se analizaron los elementos indispensables para cumplir eficientemente con los requisitos de funcionamiento. Los principales componentes del prototipo incluyen dos placas ESP32, el sensor de radiación UV GUVA-S12SD y LEDs de visualización. Asimismo, se realizó

un análisis de la radiación UV recibida por el sensor para establecer un código de colores que permita visualizar el nivel de radiación mediante dicho código.

Una vez analizados los elementos que participan en el prototipo, se inició la creación del código que se cargará en las dos placas ESP32. Una placa tendrá la función de emisor y la otra de receptor para establecer las funciones de comunicación ESP-NOW entre ambas. El código consta de una fase inicial en la que se establecen las órdenes para la radiación UV recibida por el sensor GUVA-S12SD. Los datos se cargan en la placa emisora y se envían mediante ESP-NOW a la placa receptora, que procesa la información recibida para su calibración, determinando el LED que se encenderá según el nivel de radiación, y visualizando los datos mediante el dashboard web.

Analizando los equipos necesarios y el código de programación para el prototipo, se procede a la construcción del prototipo de radiación UV usando los componentes mencionados anteriormente. Se realiza una fase inicial de experimentación con una protoboard para validar el funcionamiento del código junto con los componentes seleccionados. Si se encuentran problemas, se realizan las correcciones necesarias en el código o en los componentes.

Finalmente, se ensambla la placa PCB, realizando las conexiones adecuadas y dejando espacio para la ubicación de los elementos. Es importante destacar que, al tener como objetivo una comunicación inalámbrica mediante ESP-NOW, se realizaron dos placas PCB separadas. La primera placa consta del sensor GUVA-S12SD, una placa ESP32 y una fuente de alimentación mediante USB. La segunda placa incluye la otra ESP32, LEDs, resistencias y una batería como suministro de energía. Una vez realizadas las placas PCB, se llevan a cabo pruebas de funcionamiento en lugares con altos y bajos niveles de radiación para verificar su operatividad.

3 RESULTADOS

En esta parte del proyecto se presentarás los resultados obtenidos por cada objetivo planteado, en conjunto con una reseña de comparación de las características que debe poseer los elementos tanto de hardware y software para efectivizar el funcionamiento del prototipo. Se muestran los resultados del diseño, desarrollo e implementación del prototipo, y al finalizar se realizan las pruebas de funcionamiento para analizar los niveles de radiación UV obtenidos.

3.1 Identificación de los requerimientos para el diseño del prototipo

Selección del Sistema de un solo Chip (SoC)

Para seleccionar el chip SoC adecuado que cumpla con los requisitos del prototipo para la detección de radiación UV, es crucial asegurarse de que sea compatible con el protocolo de comunicación ESP-NOW. Este SoC también debe ofrecer capacidades de comunicación inalámbrica, como la conexión Wi-Fi, para transmitir la información al dashboard web. Además, debe contar con soporte para interfaces analógicas y digitales, facilitar la programación y el desarrollo del prototipo, y proporcionar una documentación completa para resolver problemas en caso de que surjan.

Selección del sensor de detección de radiación UV

Para seleccionar un sensor de radiación UV adecuado para el prototipo en desarrollo, es esencial considerar varios factores clave. Primero, el sensor debe ser compatible con el protocolo de comunicación ESP-NOW, que es fundamental para el funcionamiento del proyecto. Además, debe ser capaz de medir niveles de radiación UV de manera precisa y acorde con las variaciones diarias. Es importante elegir un sensor que cuente con una documentación exhaustiva, facilitando así la resolución de problemas potenciales. Finalmente, se debe buscar un sensor que ofrezca una buena relación calidad-precio, equilibrando el costo con las prestaciones y la fiabilidad del dispositivo.

Selección del dashboard web

Para la selección de un dashboard web eficiente para el prototipo en desarrollo, es necesario considerar varios criterios que permitan al usuario final una experiencia intuitiva en la visualización de la información. Es importante que el dashboard sea compatible con los elementos de hardware seleccionados para su desarrollo. Además, el dashboard debe ofrecer una gran facilidad de uso, mostrar información en tiempo real y proporcionar un buen soporte técnico para la resolución de problemas.

3.2 Selección del software y hardware necesario para el desarrollo del prototipo en base a los requerimientos establecidos

Selección del hardware

Para el desarrollo del prototipo, se utilizaron diversos componentes electrónicos que fueron esenciales para implementar con éxito el software. Entre estos elementos se incluyeron dos placas ESP32, un sensor de radiación UV, una protoboard para pruebas, una fuente de alimentación externa, luces LED, resistores y cables jumpers. A continuación, se detalla cada componente y su justificación técnica:

Selección del SoC para implementación del prototipo

Para la selección del chip de desarrollo, se eligió el microcontrolador ESP32 en su versión DevKit V1 que se muestra en la Figura 3.1. Esta opción resulta ser la más segura y confiable para el desarrollo del prototipo, ya que proporciona una integración robusta para la comunicación ESP-NOW. El ESP32 incluye un chip Wi-Fi integrado, que facilita la comunicación inalámbrica con el dashboard web. Además, cuenta con interfaces de pines tanto analógicos como digitales, lo que permite una conversión eficiente de información. Su compatibilidad con el entorno de programación Arduino IDE facilita el desarrollo, especialmente al integrarse con el dashboard proporcionado por la plataforma Arduino IoT Cloud [11].

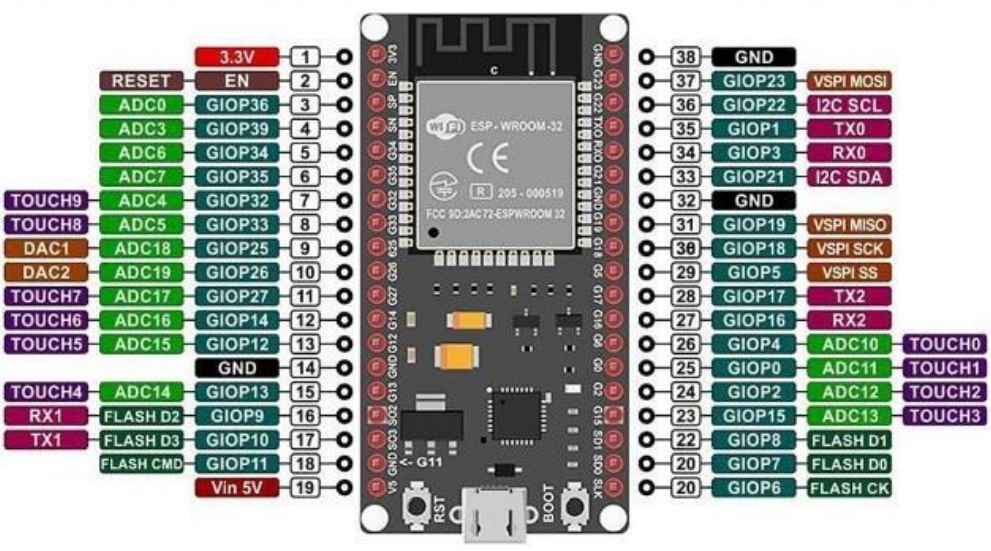

Figura 3.1 Pines de la Placa ESP32 DevKit V1

En la selección del chip SoC, inicialmente se consideró la posibilidad de utilizar la placa ESP8266, que también soporta la comunicación inalámbrica mediante el protocolo ESP-NOW. Sin embargo, se decidió no optar por este chip debido a varias razones. Aunque el ESP8266 maneja la comunicación inalámbrica y es compatible con el protocolo ESP-NOW, presenta un procesador de menor potencia en comparación con el ESP32. Además, el ESP32 ofrece mayores recursos en términos de memoria RAM, interfaces adicionales para conversiones DAC necesarias para el sensor de detección UV, y una antena RF que mejora la recepción de señal. Aunque el ESP8266 tiene un costo menor, las ventajas del ESP32 en cuanto a capacidad de procesamiento, recursos y funcionalidad justifican su elección [12].

A continuación, la Tabla 3.1 presenta una comparativa entre ambos chips para facilitar una mejor comprensión de las diferencias.

Tabla 3.1 Tabla comparativa entre SoC ESP32 y ESP8266.

Características	**ESP32**	**ESP8266**
Memoria flash	Dual-core, hasta 240 MHz	Single-core, hasta 160 MHz
Procesador	16 MB	16 MB
RAM	520 KB	160 KB
Conectividad	Wi-Fi, Bluetooth (Classic y BLE)	Solo Wi-Fi
Soporte de interfaces	Más interfaces (SPI, I2C, UART, ADC, DAC, PWM)	Interfaces limitadas
Antena RF	Amplificador de potencia, balun y amplificadores de bajo ruido integrados	Sin amplificador de potencia integrado
Uso de energía	Modo de bajo consumo más avanzado	Menos opciones de bajo consumo
Compatibilidad	Compatible con Arduino IDE, Espressif IDF, Arduino IoT Cloud.	Compatible con Arduino IDE, pero menos soporte general

Por todos los motivos mencionados anteriormente, se eligió el SoC ESP32 en específico la versión DevKit V1 que posee todas las características necesarias haciendo una integración más robusta para el desarrollo del presente proyecto [12].

Selección del sensor de detección de radiación UV

Para la selección del sensor adecuado para la detección de radiación UV, se optó por el GUVA-S12SD, como se muestra en la Figura 3.2. Este sensor fue elegido por sus características destacadas en la medición de radiación UV. El GUVA-S12SD cuenta con interfaces analógicas, lo que facilita una comunicación eficaz con el ESP32, permitiendo la conversión de datos en valores digitales para su visualización [13].

La elección del GUVA-S12SD se realizó también considerando la disponibilidad de otros sensores, como el LTR-390UV y el ML8511 UVB, los cuales no estaban disponibles para su adquisición en el país en el momento de la selección. Por lo tanto, el GUVA-S12SD fue la opción más viable y adecuada para el desarrollo del prototipo [13].

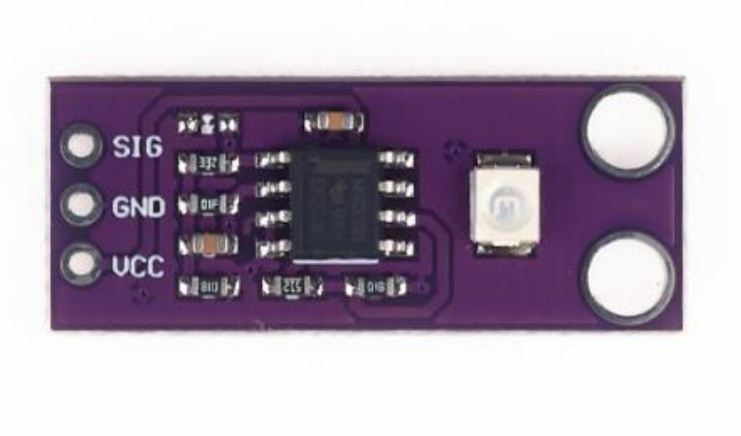

Figura 3.2 Módulo sensor GUVA-S12SD

Adicional, se realiza una comparación de distintos tipos de sensores como se observa en la Tabla 3.2 de los 3 sensores de radiación UV que se tenían como opciones para usar desde su inicio, pero se optó por el sensor GUVA-S12SD.

Tabla 3.2 Tabla comparativa entre sensores UV GUVA-S12SD vs LTR-390UV vs ML8511

Características	GUVA-S12SD	LTR-390UV	ML8511
Tipo de sensor	Sensor de radiación UV analógico	Sensor de radiación UV digital	Sensor de radiación UV analógico
Rango de medición	200-400 nm	200-400 nm	280-400 nm
Salida			
Sensibilidad	Alta (proporciona lecturas precisas)	Moderada (dependiente del procesamiento digital)	Alta (dependiente de la conversión analógica)
Interfaz	Salida analógica	Interfaz I2C	Salida analógica
Voltaje de operación	5V	3.3V/5V	3.3V
Disponibilidad	Alta	Baja	Baja
Costo:	Bajo	Alto	Alto

En resumen, la elección del sensor GUVA-S12SD se basó en sus características técnicas y en su disponibilidad. Este sensor destacó por su compatibilidad con el SoC ESP32 y Arduino Cloud, lo que facilitó su integración en el prototipo. Comparado con el sensor digital LTR-390UV y el sensor analógico ML8511 UVB, el GUVA-S12SD ofreció una mayor facilidad de integración en proyectos, un costo más bajo y una mejor disponibilidad en el mercado. Estas ventajas hicieron que el GUVA-S12SD fuera la opción preferida para el desarrollo del prototipo [14].

Selección del software de desarrollo

Plataforma de desarrollo

El software del prototipo de radiación UV es la base interna de funcionamiento del proyecto, debido a que este realiza de manera lógica y sistemática todos los pasos que va a realiza el prototipo. También, es importante destacar que para que funciones este proyecto se tuvo que trabajar con el lenguaje de programación C++ de Arduino que es compatible con las placas esp32, el lenguaje programación C++ es un lenguaje de

mediano nivel, que es usado en gran parte por la comunidad de clientes que se dedican a la electrónica por su gran portabilidad, y a su fácil creación de construcción de proyectos [15].

El Arduino IDE es una herramienta fundamental en el desarrollo del prototipo. Este software proporciona la plataforma necesaria para la programación de los microcontroladores y el sensor de radiación UV. Ofrece acceso a bibliotecas especializadas que facilitan la integración de placas ESP32, el protocolo ESP-NOW, sensores y otros componentes [15]

Una de las ventajas del Arduino IDE es su capacidad para cargar programas y realizar validaciones rápidas, lo que permite un desarrollo ágil con etapas de prueba y corrección de errores. Su flexibilidad también permite la integración de diversos componentes, como el sensor de radiación UV, y es compatible con la mayoría de los equipos utilizados en este proyecto [15].

Además, el Arduino IDE es crucial para la carga del firmware en las placas ESP32, asegurando una configuración adecuada para el funcionamiento del protocolo ESP-NOW. Una característica destacada del Arduino IDE es su capacidad de depuración y corrección de errores a través de su ventana de visualización. Durante el desarrollo del código, se presentaron problemas comunes que fueron solucionados con las herramientas de depuración del IDE. El uso de foros y blogs de comunidades especializadas en esta plataforma también resultó muy útil para implementar bibliotecas y resolver errores que surgieron durante el proceso [15].

Dashboard Web

En la elección de una plataforma para la demostración de resultados en la nube, se optó por la plataforma Arduino IoT Cloud debido a que es una aplicación en la nube que ayuda al usuario a visualizar datos en tiempo real con una interfaz intuitiva y una gran cantidad de widgets modificables. Se centra en su facilidad de uso y permite conectar una gran cantidad de dispositivos electrónicos sin requerir un conocimiento avanzado en programación. Aunque se consideraron otras opciones como Adafruit IO y ThingSpeak**,** la elección de Arduino IoT Cloud se basó en las características que se presentan en la Tabla 3.3 [16]**.**

Tabla 3.3 Comparativa entre plataformas de la nube para dashboard web

Características	Arduino IoT Cloud	Adafruit IO	ThinkSpeak
Integración	Nativa con dispositivos Arduino y ESP32	Amplia gama de dispositivos IoT	Compatible con dispositivos IoT y MATLAB
Facilidad de uso	Interfaz intuitiva	Interfaz amigable con curva de aprendizaje moderada	Interfaz sencilla, requiere conocimientos básicos de programación
Visualización en tiempo real	Dashboards personalizables con gráficos y widgets en tiempo rea	Dashboards personalizables con widgets dinámicos	Gráficos en tiempo real y análisis con MATLAB
Compatibilidad	Múltiples tipos de datos y sensores	Soporta múltiples protocolos y tipos de datos	Soporta múltiples tipos de datos y fácil integración con
Documentación y soporte	Amplia documentación y comunidad activa	Extensa documentación y comunidad activa	Amplia documentación y soporte técnico de MATLAB
Costo	Modelo de suscripción con opciones gratuitas y de pago	Modelo de suscripción con opciones gratuitas y de pago	Modelo gratuito con opciones de pago para características avanzadas

La plataforma Arduino IoT Cloud se encargó de gestionar el dashboard web del proyecto, ya que permite la visualización en tiempo real de los datos de radiación UV, los cuales se suben a la nube y están disponibles en cualquier momento. Esta plataforma no solo facilita la visualización continua de los datos, sino que también

proporciona un sistema de seguridad eficiente con encriptación para proteger la información recibida [16].

Una vez que los datos llegan al dashboard web, se almacenan de manera permanente en la nube, garantizando su accesibilidad para el usuario en todo momento. El dashboard web incluye un diagrama de barras que muestra todos los eventos registrados por las placas ESP32, permitiendo una línea de tiempo detallada de la radiación UV a lo largo del día. Además, se implementó un sistema de semaforización de colores para alertar a los usuarios sobre los niveles de radiación: rojo para alta radiación, naranja para niveles normales y verde para baja radiación o ausencia de radiación [16].

Arduino Cloud se eligió como dashboard web debido a su compatibilidad con elementos de IoT, lo que optimiza el funcionamiento de las placas ESP32 al trabajar bajo los mismos protocolos de desarrollo. Esta integración facilita una experiencia eficiente en la gestión y visualización de los datos de radiación UV [16].

3.2 Diseño del prototipo de detección de radiación UV

Principio de funcionamiento del sensor

La Figura 3.3 muestra los pasos que se van a seguir para los resultados apropiados del proyecto en el objetivo del diseño del prototipo de detección, se muestra de manera minuciosa como se va a recibir desde el primer paso la información hasta su último paso que será la visualización de la radiación UV.

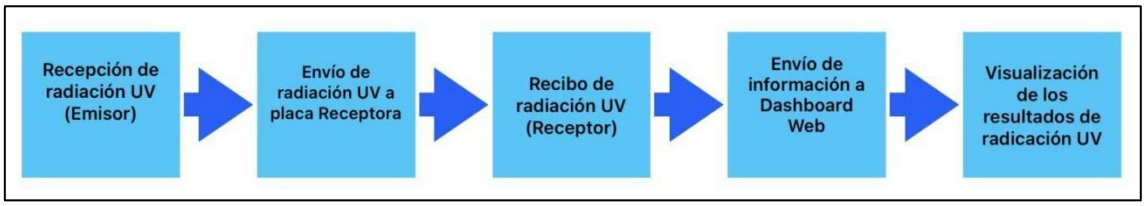

Figura 3.3 Diagrama de bloques del desarrollo del proyecto

La Figura 3.4 muestra el proceso que se va a realizar de manera gráfica en un entorno natural desde su primer punto que son los niveles de radiación UV hasta su último paso que es la visualización de datos en un dashboard web.

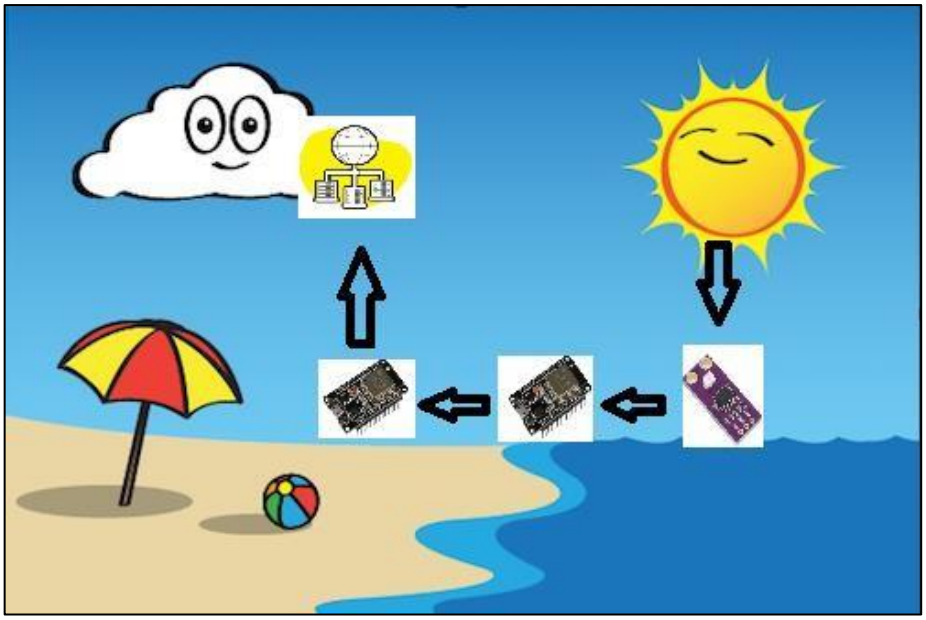

Figura 3.4 Esquema del sensor de radiación UV

Creación de código de funcionamiento

A continuación, se presentan los pasos que se siguen para el desarrollo del proyecto en base a la programación. Este proyecto se divide en varias etapas para asegurar su correcto funcionamiento. La primera etapa consiste en la medición de la radiación UV utilizando el sensor GUVA-S12SD, desde el cual se envía la información al emisor. Este emisor transmite los datos a la placa receptora, que, a su vez, envía automáticamente la información al servidor web de Arduino Cloud.

Programación del ESP32 emisor

El diagrama de flujo para el ESP32 del emisor se presenta en la Figura 3.5 en donde se detalla la base de cómo funciona la transmisión de datos hasta la placa receptora. Empieza con la comunicación serial y del pin analógico para leer los datos del sensor GUVA-S12SD. Luego, se establece el enlace por comunicación inalámbrica Wi-Fi con sus debidas credenciales. Si la configuración Wi-Fi es correcta se procede a activar el protocolo ESP-NOW caso contrario espera 200ms para repetir la acción. Una vez que se obtiene el enlace con ESP-NOW, el emisor envía la información a la placa receptora y es recibido con un mensaje de éxito.

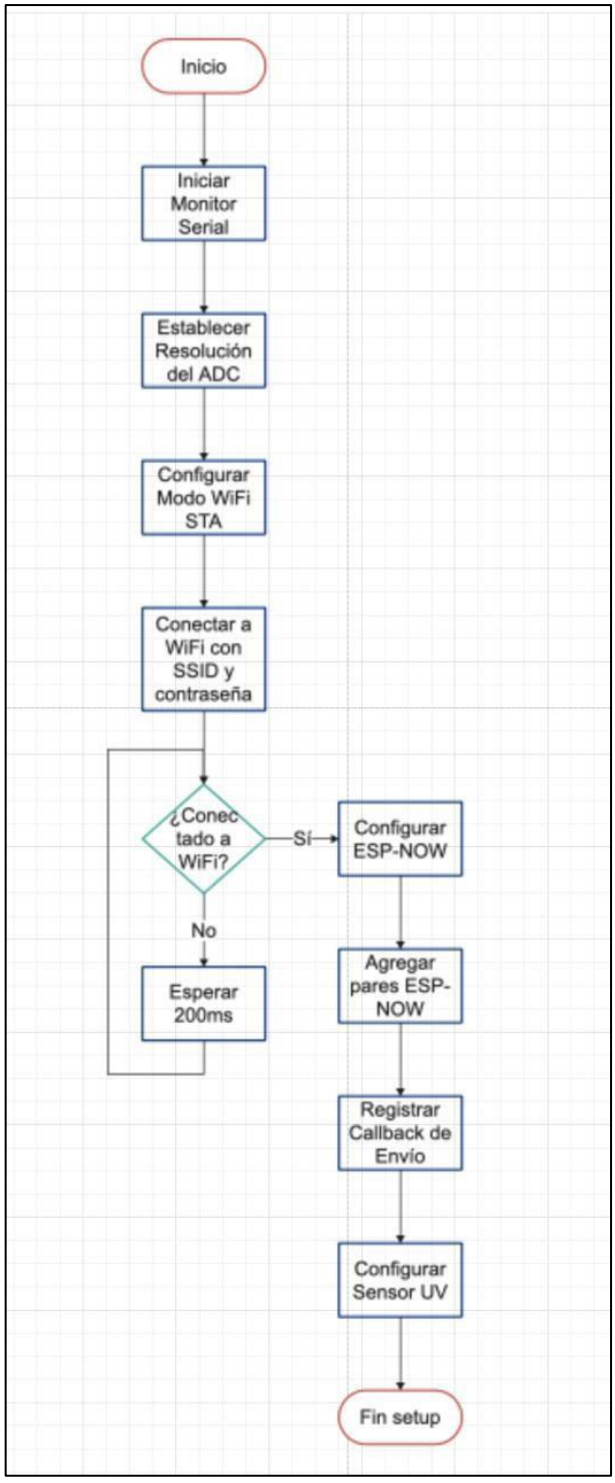

Figura 3.5 Diagrama de flujo ESP32 Emisor

El primer paso en el desarrollo del prototipo es la implementación de la programación en la placa emisora. Para comenzar, se añaden las librerías necesarias, que en este caso son <esp_now.h> y <WiFi.h>, como se muestra en la Figura 3.6 .La librería <esp_now.h> se utiliza para permitir la comunicación inalámbrica entre las dos placas ESP32, que es el objetivo principal del proyecto. Por otro lado, la librería <WiFi.h> se emplea en varias etapas del proceso, y su función principal es facilitar la conexión con el servidor de Arduino Cloud.

```
1   #include <esp_now.h>
2   #include <WiFi.h>
3
```

Figura 3.6 Declaración de librerías Emisor

A continuación, se procede a declarar las constantes y variables del programa. SSID y PASSWORD están relacionadas con el acceso a la red Wi-Fi local. UvSensorPin corresponde al pin analógico conectado al sensor GUVA-S12SD. Además, RECEIVER_MAC representa la dirección MAC única utilizada para identificar el dispositivo receptor. La Figura 3.7muestra todas las variables y constantes utilizadas.

```
const char* ssid = "CELERITY_WLAN802";
const char* password = "LAURA1304413345";

const int uvSensorPin = 34;  // Pin analógico al que está conectado el sensor UV

// Reemplazo con la mac de la placa receptor
uint8_t mac_receptor[] = { 0xE4, 0x65, 0xB8, 0x7A, 0xAF, 0x08 };
```

Figura 3.7 SSID y Password conexión Wi-Fi Emisor

En la Figura 3.8 se define una estructura llamada struct_message, que incluye un campo denominado lectura para almacenar el valor obtenido del sensor UV. Esta estructura se instancia con el nombre mensaje.

```
// Estructura de envio de datos
typedef struct struct_message {
  int lectura;
} struct_message;

struct_message mensaje;
```

Figura 3.8 Estructura de radiación UV Emisor

En la Figura 3.9, se observa que se ha creado una variable llamada peerinfo para recolectar los datos del dispositivo receptor. Además, la función OnDataSent actúa como una función de retorno que se ejecuta cada vez que se envía información, mostrando en la pantalla serial un mensaje de error o éxito según el resultado de la transmisión.

```
esp_now_peer_info_t peerInfo;

void OnDataSent(const uint8_t* mac_addr, esp_now_send_status_t status) {
  Serial.print("Estado de envío: ");
  if (status == ESP_NOW_SEND_SUCCESS) {
    Serial.println("Enviado");
  } else {
    Serial.println("Fallo");
  }
}
```

Figura 3.9 Recolección de datos del receptor y envio con ESP-NOW Emisor

En esta sección del código, como se muestra en la Figura 3.10, se inicializa el monitor serial, se configura la resolución para la lectura analógica y se conecta el ESP32 a una red WiFi. Tras establecer la conexión, se inicializa ESP-NOW y se registra la función de retorno OnDataSent. A continuación, se configura la información del dispositivo receptor al copiar la dirección MAC en peerinfo y se añade el dispositivo receptor a la red.

```
void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);
  analogReadResolution(9);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  // Esperar a que nos conectemos
  while (WiFi.status() != WL_CONNECTED) {
    delay(200);
    Serial.print('.');
  }

  Serial.println();
  Serial.print("Conectado a:\t");
  Serial.println(WiFi.SSID());
  Serial.print("IP address:\t");
  Serial.println(WiFi.localIP());

  delay(1500);

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
```

Figura 3.10 Inicialización de ESP-NOW Emisor

A continuación, la Figura 3.11 ilustra la función loop, en la que se lee el valor del sensor UV conectado al pin uvSensorPin. Este valor se almacena en el campo lectura de la estructura mensaje. Luego, el mensaje se envía al dispositivo receptor utilizando ESP-NOW. Finalmente, se muestra un mensaje de éxito o error en la pantalla, según el resultado del envío, y este proceso se repite cada 2 segundos.

```
void loop() {

  // Read UV sensor value
  int uvValue = analogRead(uvSensorPin);

  mensaje.lectura = uvValue;
  esp_err_t result = esp_now_send(mac_receptor, (uint8_t*)&mensaje, sizeof(mensaje));

  if (result == ESP_OK) {
    Serial.println("Sent with success");
  } else {
    Serial.println("Error sending the data");
  }
  delay(2000);
}
```

Figura 3.11 Lectura del sensor UV

Programación del ESP32 Receptor

En la Figura 3.12 se describe el procedimiento de la placa de recepción ESP32. Se inicia, el sistema configurando mensajes de enlazamiento hacia la red Wi-Fi y el éxito con la inicialización del protocolo ESP-NOW, posterior de estos pasos se empieza la comunicación para los datos de recepción. Finalizado el enlace de recepción de los datos, el diagrama de flujo muestra los datos que se almacena en una estructura específica. El microcontrolador ESP32 lleva a cabo un proceso de calibración y procesamiento de la información garantizando una correcta lectura de radiación UV. Luego de este procesamiento, imprime los valores de radiación UV en el monitor serial y envía la información con conectividad Wi-Fi hacia el Arduino Cloud.

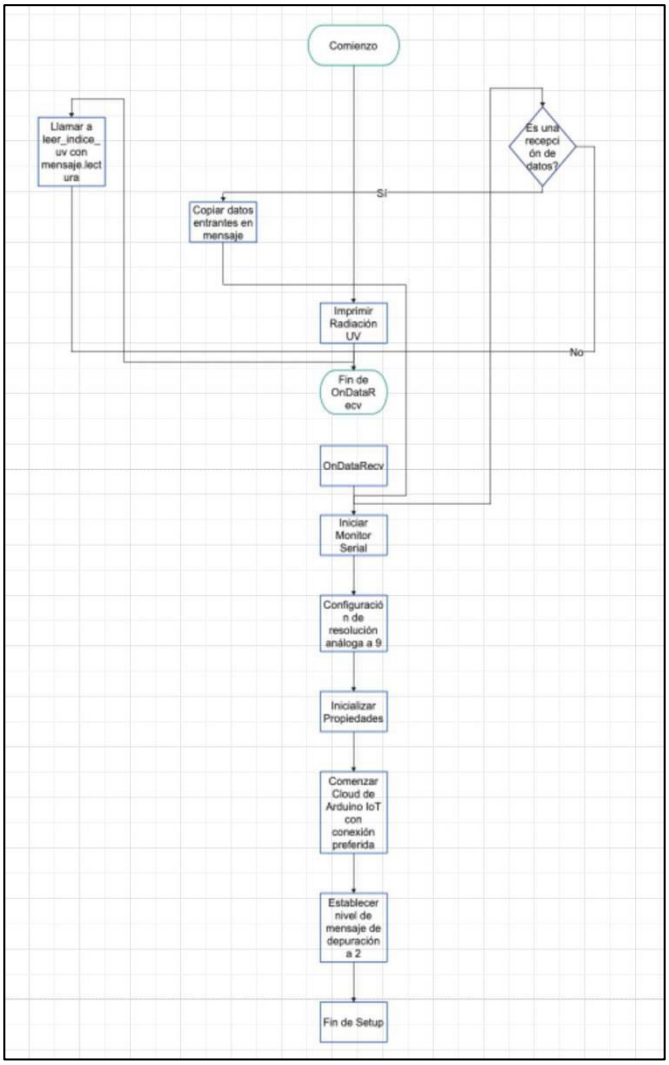

Figura 3.12 Diagrama de flujo microcontrolador ESP32 Receptor

En la imagen del código del receptor la Figura 3.13 muestra la incorporación de la librería de ESP-NOW, la cual facilita la comunicación entre dispositivos ESP32 sin requerir una red WiFi. La inclusión de otras librerías está comentada, ya que no se utilizan directamente en este código.

```
#include <esp_now.h>
//#include <WiFi.h>
//#include <WiFiClientSecure.h>
#include "thingProperties.h"
//#include <ArduinoJson.h>
```

Figura 3.13 Inclusión de librerias Receptor

La Figura 3.14 muestra la utilización de tres pines: uno verde, uno amarillo y uno rojo, para controlar los LEDs que indican los niveles de radiación UV mediante un sistema de semaforización. También se declararon variables para almacenar el índice UV y una bandera auxiliar. La estructura struct_message se empleó para registrar el valor capturado por el sensor UV.

```
const int verde = 5;
const int amarillo = 18;
const int rojo = 19;

// const char* ssid =
// const char* password = "095017068@[_Da";

float uvIndex = 0;
bool aux = false;

typedef struct struct_message {
  int lectura;
} struct_message;

struct_message mensaje;
```

Figura 3.14 Declaración de Variables Receptor

La Figura 3.15 muestra el uso de una función que se activa cada vez que se recibe información mediante el protocolo ESP-NOW. Los datos obtenidos se almacenan directamente en la estructura mensaje, se imprimen en el monitor serial y se llama a la función leer_indice_uv para procesar la lectura.

```
void OnDataRecv(const uint8_t* mac, const uint8_t* incomingData, int len) {
  memcpy(&mensaje, incomingData, sizeof(mensaje));
  Serial.print("Radiación UV recibida por ESP-NOW: ");
  Serial.println(mensaje.lectura);
  leer_indice_uv(mensaje.lectura);
}
```

Figura 3.15 Almacenamiento de datos recibidos Receptor

En la sección que muestra la Figura 3.16 , se inicia la comunicación serial para la visualización del ESP32, se configura la lectura analógica y se establecen los ajustes necesarios para la integración con Arduino Cloud. Además, en la Figura 3.17 se observa que los LEDs se configuran como salidas, pero no se encienden al inicio del proceso.

```
void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);
  analogReadResolution(9);

  //WiFi.mode(WIFI_STA);
  initProperties();

  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  Serial.println();
  Serial.print("Conectado a:\t");
  Serial.println(WiFi.SSID());
  Serial.print("IP address:\t");
  Serial.println(WiFi.localIP());

  delay(1500);
```

Figura 3.16 Configuración de lectura analógica Receptor

```
  pinMode(verde, OUTPUT);
  pinMode(amarillo, OUTPUT);
  pinMode(rojo, OUTPUT);

  digitalWrite(verde, LOW);
  digitalWrite(amarillo, LOW);
  digitalWrite(rojo, LOW);
}
```

Figura 3.17 Configuración de LEDs como salidas del receptor

En la Figura 3.18 se muestra el ciclo principal, el código verifica la conexión del ESP32 con Arduino Cloud. Si ESP-NOW aún no ha sido inicializado, procede a configurarlo. Luego, se actualiza el estado de Arduino Cloud y se registra la función de retorno OnDataRecv para recibir datos. Este proceso se repite continuamente, con breves pausas entre cada iteración.

```
void loop() {
  //esp_now_register_recv_cb(esp_now_recv_cb_t(OnDataRecv));
  if (ArduinoCloud.connected() && !aux) {
    if (esp_now_init() != ESP_OK) {
      Serial.println("Error initializing ESP-NOW");
      //return;
    }
    aux = true;
  }
  ArduinoCloud.update();

  delay(1500);
  esp_now_register_recv_cb(esp_now_recv_cb_t(OnDataRecv));
  delay(1000);
}
```

Figura 3.18 Ciclo principal para la conexión del ESP-NOW con arduino Cloud

Por último, en la Figura 3.19 se puede como la función “leer_indice_uv” realiza varias tareas críticas para la interpretación y visualización de los niveles de radiación UV en tiempo real. Primero, convierte un valor analógico del sensor UV en voltaje usando la fórmula estándar (valor_analogico * (3.3 / 1023.0)), esencial para la precisión de la lectura. A partir de este voltaje calculado, determina el índice UV dividiéndolo por 0.1, proporcionando una estimación del nivel de radiación UV detectado. Este índice UV se muestra en el monitor serial y se asigna a la variable global indiceUV, permitiendo su

uso continuo. Luego, actualiza el estado de los LEDs según el índice UV para indicar diferentes niveles de riesgo: enciende el LED verde para riesgo bajo (uvIndex entre 0 y 3), el LED amarillo para riesgo moderado (entre 3 y 5), y el LED rojo para niveles más altos de riesgo.

Después de cada actualización de LEDs, la función espera 1 segundo antes de repetirse, asegurando mediciones periódicas sin sobrecargar el sistema. Este proceso integrado de conversión, cálculo, visualización y control de LEDs es crucial para aplicaciones de monitoreo ambiental y protección contra los efectos nocivos de la radiación UV.

```
void leer_indice_uv(int analogico) {
  float voltaje = analogico * (3.3 / 1023.0);  //valor analogico a voltaje
  uvIndex = voltaje / 0.1;                     // Ejemplo de fórmula genérica para el sensor
  // Imprimir el valor del índice UV en el monitor serial
  Serial.print("UV Index: ");
  Serial.println(uvIndex);
  indiceUV = uvIndex;

  //niveles de radiacion: https://es.wikipedia.org/wiki/%C3%8Dndice_UV
  if (uvIndex >= 0 && uvIndex < 3) {
    Serial.println("Riesgo: Bajo");
    riesgoBajo = true;
    digitalWrite(verde, HIGH);
    digitalWrite(amarillo, LOW);
    digitalWrite(rojo, LOW);
  } else if (uvIndex >= 3 && uvIndex < 5) {
    Serial.println("Riesgo: Moderado");
    riesgoBajo = true;
    digitalWrite(verde, LOW);
    digitalWrite(amarillo, HIGH);
    digitalWrite(rojo, LOW);
  } else if (uvIndex >= 5 && uvIndex < 7) {
    Serial.println("Riesgo: Alto");
    riesgoBajo = false;
    digitalWrite(verde, LOW);
    digitalWrite(amarillo, LOW);
    digitalWrite(rojo, HIGH);
```

Figura 3.19 Interpretación de Radiación UV con semaforización de LEDs Receptor

Programación del ESP32 Receptor – Integración con Arduino IoT Cloud

Finalmente, en la plataforma IoT Cloud, la visualización de la nube se llevará a cabo durante varios períodos sucesivos. Primero, se configura el entorno de IoT para integrar las bibliotecas necesarias para manejar los datos. Luego, se asocia el dispositivo con un identificador único e ingrese las credenciales de acceso a las redes Wi-Fi. A continuación, se declara el índice UV y las variables variables para representar los niveles de radiación UV y las variables asociadas asociadas. Posteriormente, se

configura el identificador del dispositivo con una clave secreta para establecer una conexión Wi-Fi segura y transmitir los datos. Este proceso asegura que la información se muestre en tiempo real, con gráficos intuitivos para el usuario, como se ilustra en la Figura 3.20.

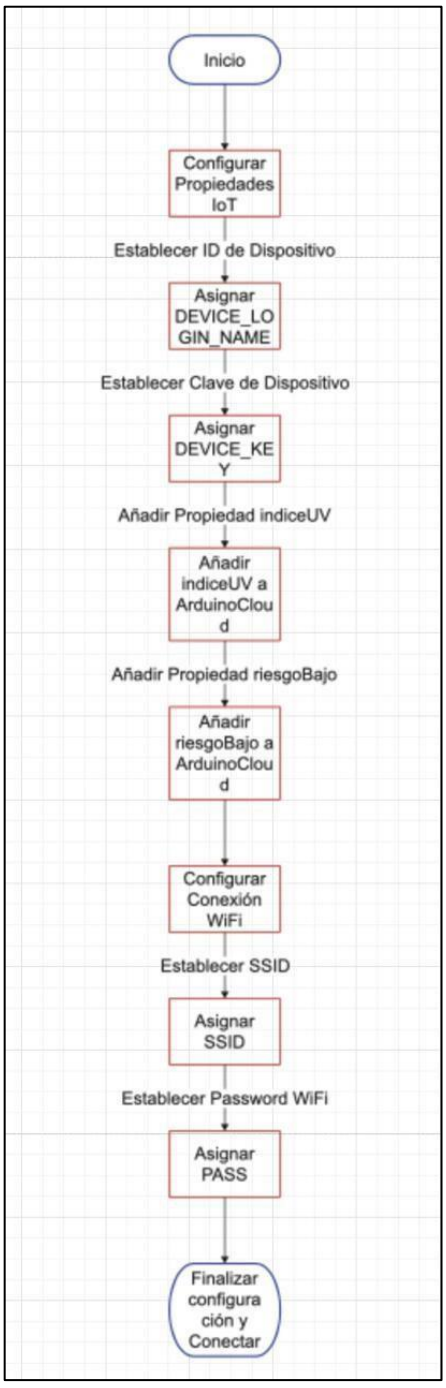

Figura 3.20 Diagrama de flujo de Placa ESP32 hacia Arduino Cloud

La siguiente sección del código del receptor está diseñada para integrar el prototipo con Arduino IoT Cloud, facilitando la gestión del sensor desde la nube. A continuación, se describe la programación utilizada para configurar la conexión con Arduino Cloud.

El primer paso en el proceso de programación consiste en incluir las librerías necesarias para vincular el prototipo con la nube a través de Arduino IoT Cloud. Las librerías requeridas se muestran en la Figura 3.21.

```
#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>
```

Figura 3.21 Inclusión de librerías

A continuación, se realiza la declaración de variables y constantes que se utilizarán. La Figura 3.22 muestra estas constantes. La variable DEVICE_LOGIN_NAME representa una dirección única asignada al dispositivo, mientras que SSID y PASS corresponden a las credenciales necesarias para conectar el prototipo a la red Wi-Fi y, posteriormente, a la nube de Arduino IoT Cloud. Por otro lado, DEVICE_KEY es el código secreto del dispositivo utilizado para autenticar la conexión.

```
const char DEVICE_LOGIN_NAME[] = "3730afba-89de-41fd-b425-aec8da58474b";

const char SSID[] = "CELERITY_WLAN802";     // Nombre de la Red Wifi
const char PASS[] = "LAURA1304413345";  // Clave de la red Wifi

const char DEVICE_KEY[] = "ZkeD4MbnOv0F@Zby8efm#aeTd";  // Clave secreta
```

Figura 3.22 Declaración de variables

Además, el código posee unas variables globales llamadas índiceUV y el estado de riesgoBajo se muestran en la Figura 3.23 que se usa para la medición de radiación UV, estos datos se actualizarán periódicamente y se subirán para su respectiva visualización en Arduino IoT Cloud.

```
float indiceUV;
bool riesgoBajo;
```

Figura 3.23 Variables globales

La Figura 3.24 ilustra la función utilizada para enlazar las propiedades de la placa de desarrollo ESP32 con Arduino IoT Cloud. Esta función se encarga de asignar la identificación (ID) y la clave secreta del dispositivo. Además, configura las variables correspondientes a las propiedades radiaciónUV y riesgoBajo, permitiendo que estas se actualicen en tiempo real siempre que sus estados cambien.

```
void initProperties() {

  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
  ArduinoCloud.addProperty(indiceUV, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(riesgoBajo, READ, ON_CHANGE, NULL);
}
```

Figura 3.24 Enlazamiento de ESP32 con Arduino Cloud

Finalmente, el código que se muestra en la Figura 3.25 administrará la conectividad inalámbrica a través de Wi-Fi, utilizando los datos especificados en las primera líneas.

```
WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```

Figura 3.25 Credenciales de conexión vía Wi-Fi

3.3 Implementación del prototipo

Para la construcción del prototipo, se inicia con la realización de pruebas de funcionamiento de los componentes mencionados anteriormente, que incluyen: dos placas ESP32, un sensor GUVA-S12SD, LEDs, resistencias, cables USB y jumpers. La Figura *3.26* ilustra la fase de prueba, en la que se utilizó un protoboard para ensamblar el circuito y llevar a cabo las pruebas correspondientes.

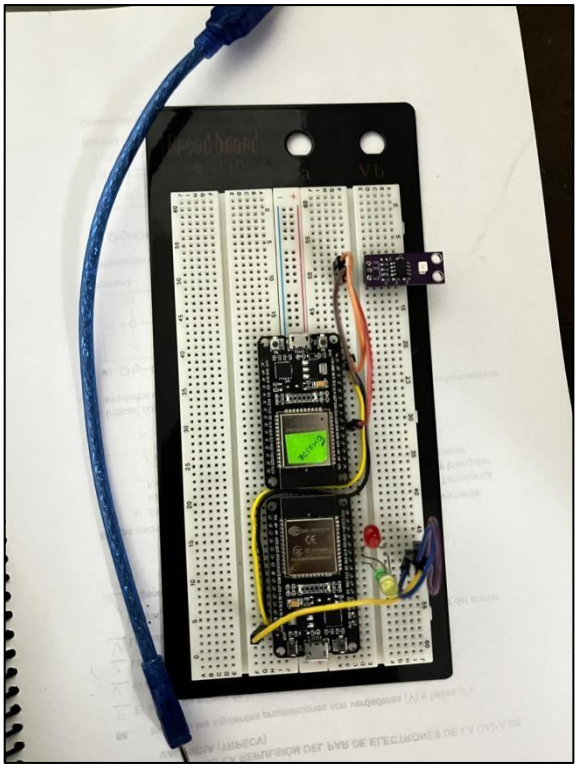

Figura 3.26 Pruebas de funcionamiento en Protoboard

Una vez completadas las etapas de prueba en el Protoboard, se procede al ensamblaje del prototipo. El primer paso es el diseño del circuito de PCB, utilizando el software Proteus se realiza todas las conexiones como se puede observar en la Figura 3.27.Tras finalizar el diseño, se exporta a un archivo PDF, que también se presenta en la Figura 3.28, para su posterior impresión. Es crucial verificar que todas las conexiones en el diseño sean correctas y que el diseño se ajuste adecuadamente al espacio destinado para cada componente.

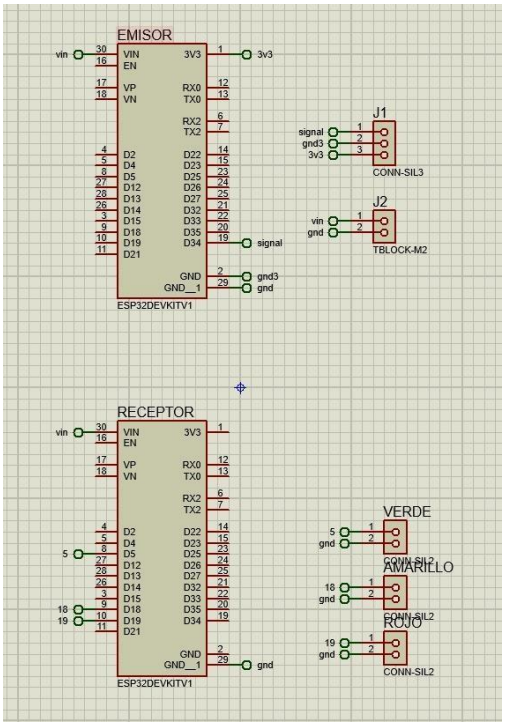

Figura 3.27 Diseño del prototipo emisor y receptor en Proteus

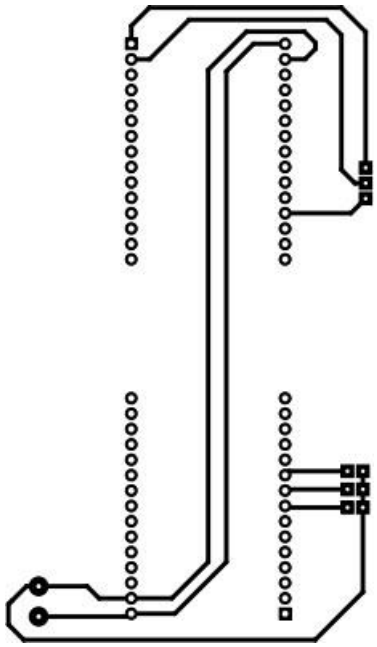

Figura 3.28 Diseño de prototipo del emisor y receptor con Proteus transformado a PDF

Después de realizar el diseño, se procede con la impresión del diseño en la baquelita. Para imprimir el diseño de PCB, se comienza exportando el diseño final en formato PDF desde el software de diseño. Luego, se imprime en una impresora láser sobre papel fotográfico o transparencias, ajustando la calidad a alta y sin escalado. El papel impreso se coloca sobre una placa de baquelita limpia y se transfiere el diseño utilizando una plancha, aplicando calor y presión uniformemente. Finalmente, se revisa la baquelita para asegurar que el patrón del circuito se haya transferido correctamente y sin defectos como se muestra en la Figura 3.29.

Figura 3.29 Impresión del diseño en PCB

Una vez que el diseño se ha impreso en la baquelita, se procede a la etapa de grabado utilizando una solución de percloruro férrico en una bandeja de plástico, como se muestra en la Figura 3.30 . La baquelita se sumerge completamente en la solución y se espera hasta que el cobre expuesto haya sido disuelto, dejando solo el cobre protegido por el tóner impreso. Tras retirar la baquelita de la solución, se limpia con alcohol isopropílico para eliminar cualquier resto de cobre no deseado, dejando únicamente el cobre en las pistas del prototipo.

Figura 3.30 Etapa de grabado de diagrama en PCB

Para perforar los agujeros en el diseño y permitir el ajuste de los componentes, se utiliza un taladro con una broca de 0.8 mm. Esto permite obtener la baquelita con los agujeros necesarios, como se muestra en la Figura 3.31 **.**

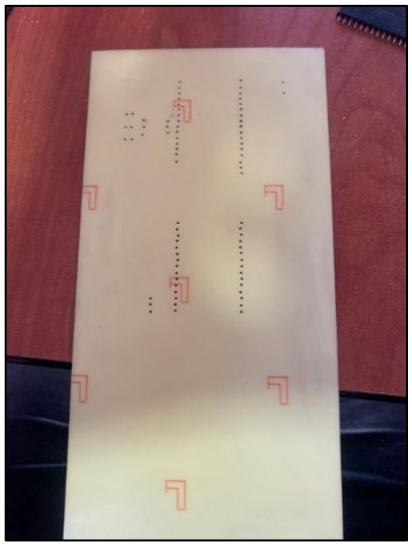

Figura 3.31 Perforación de agujeros para los elementos

Una vez realizado los respectivos agujeros, es tiempo de empezar con el armado del prototipo. En este caso para comprobación del protocolo ESP-NOW se procede a usar 2 baquelitas, la primera baquelita que para uso de este proyecto se llamará baquelita emisor, esta baquelita se encarga de realizar la medición de radiación de UV con el

sensor GUVA-S12-SD tal como se muestra en la baquelita 1 de la Figura 3.32. La segunda baquelita que se va a conectar con el protocolo ESP-NOW se muestra en la Figura 3.33 . Esta baquelita se la nombra la baquelita receptora debido a que recibe la información, se implementó una fuente externa de energía para usos de portabilidad en el prototipo.

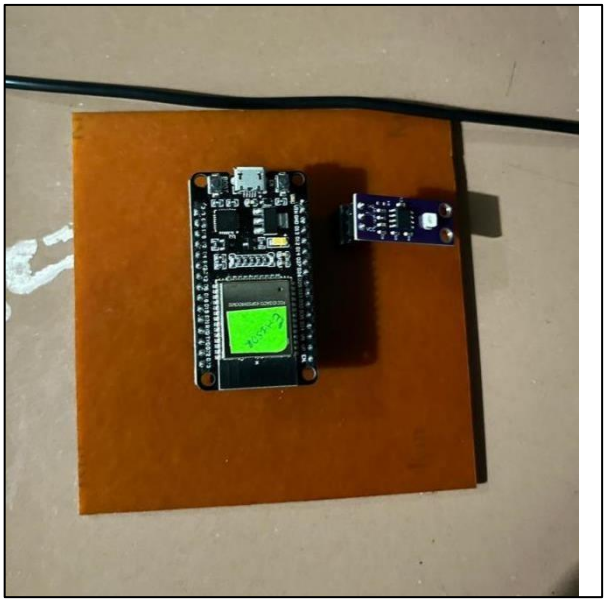

Figura 3.32 Placa con sensor GUVA-S12SD con ESP32 Emisor

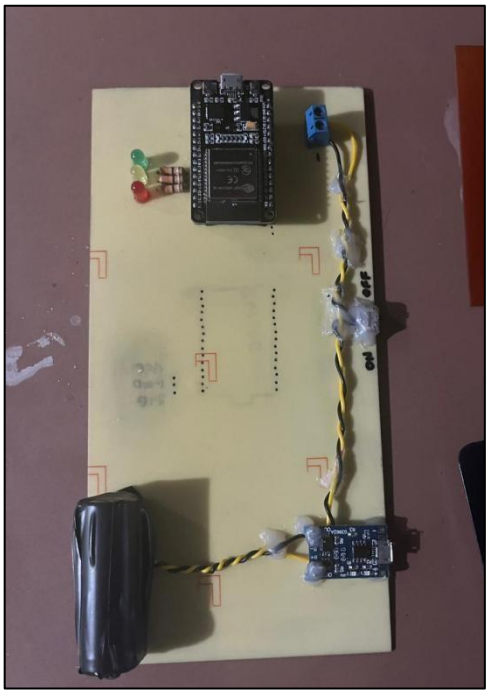

Figura 3.33 Prototipo armado de Placa Receptora

3.4 Pruebas de funcionamiento del prototipo

Por último, se llega al último objetivo planteado que es realizar pruebas de funcionamiento de prototipo de detección de radiación UV. Para este último paso, es importante señalar que para asegurar que se encuentra correctamente funcionando el prototipo se realizaron pruebas de funcionamiento en una terraza del autor que sirve como caso de estudio que tenían tanto sombra como sol en el ambiente.

Para la comprobación de funcionamiento del protocolo ESP-NOW como protocolo inalámbrico se separaron las placas a una distancia de 1 (m), y también se muestra en la ventana serial de la plataforma Arduino IDE. Por tanto, en la siguiente sección se empezará con la inicialización del prototipo.

Inicialización del prototipo

Para observar la conexión de las placas tanto a la red Wi-Fi como el enlace del protocolo ESP-NOW, se utilizó el monitor serial de la placa receptora ESP32, la cual se conectó mediante cable USB al PC para esta prueba de inicialización. Como se puede observar en la Figura 3.34, hubo mensajes de fallo en la conexión Wi-Fi hasta que se estableció la conexión con el mensaje “Connected to CELERITY_WLAN802”, indicando que se conectó a la red Wi-Fi. La comunicación mediante el protocolo ESP-NOW se estableció una vez que comenzó la impresión de “UV Index: 00”. Antes de este mensaje, se visualizaba "ESP-NOW not init", que era el mensaje programado en caso de que no existiera conexión mediante el protocolo. En la última línea de la figura, también se visualiza la conexión a Arduino IoT Cloud con el mensaje “Connected to Arduino IoT Cloud”.

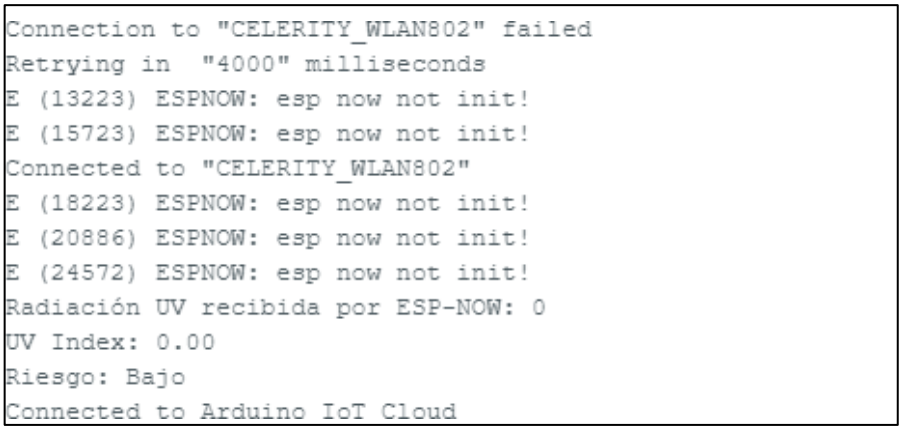

Figura 3.34 Conexión a ESP-NOW, Wi-Fi y Arduino Cloud

Ubicación de los dispositivos

Como parte de los objetivos de las pruebas de funcionamiento, era necesario comprobar que el protocolo inalámbrico ESP-NOW estuviera operando correctamente. En la Figura 3.35 se puede observar la ubicación de la placa ESP32 emisora, mientras que en otra Figura 3.36 se visualiza la ubicación de la placa ESP32 receptora. Ambas placas se encuentran en la parte alta de la terraza de uno de los autores del prototipo, funcionando esta ubicación servirá como un caso de estudio. En la Figura 3.37 se puede ver que las placas están situadas a una distancia de 1 (m).

Figura 3.35 Placa ESP32 Emisor

Figura 3.36 Placa ESP32 Receptor

Figura 3.37 Vista General del Prototipo

Lecturas de funcionamiento

Niveles normales de radiación UV

Una vez iniciada la comunicación y definida la ubicación para las mediciones respectivas, se comienzan las pruebas de medición. Durante las pruebas se verificaron dos condiciones climáticas. La primera se llevó a cabo a la sombra, sin exposición directa al sol, se puede el señor GUVA S-12SD al descubierto para su medición y al LED verde encendido por sus niveles normales de radiación UV como se muestra en la Figura 3.38. En esta condición, se esperaban mediciones normales. Tal como se puede observar en la Figura 3.39, las lecturas fueron menores a 3 (UV), específicamente un valor de 2,71 (UV). Este valor se refleja en la aplicación Arduino IoT Cloud, donde en la pantalla RADIACIÓN UV se muestra un color verde, indicando niveles bajos de radiación UV.

Figura 3.38 Medición de sensor UV (Niveles Normales)

Figura 3.39 Lectura de medición del sensor UV (Niveles Normales)

Niveles bajos de radiación UV

Posterior a esta primera medición, se realizó una medición con el sensor completamente cubierto en el cual se debe mostrar un valor aproximadamente a cero. La Figura 3.40 muestra cómo se cubrió el sensor UV y la Figura 3.41 muestra su valor de 0 (UV) al no tener ninguna exposición al sol.

Figura 3.40 Medición de sensor UV (Niveles Bajos)

Figura 3.41 Lecturas de medición del sensor UV (Niveles Bajos)

Niveles altos de radiación UV

Mientras que, para finalizar con las pruebas de medición, se expuso el sensor de radiación UV a plena luz solar a las 12:30 PM. La Figura 3.42 muestra su exposición a altos niveles de sol y los resultados indican una lectura de radiación UV de 11.968 que se muestran en la Figura 3.43, lo cual advierte que se debe usar protección ante la radiación.

Figura 3.42 Medición de radiación UV (Niveles Altos)

Figura 3.43 Lecturas de medición del sensor UV (Niveles Altos)

4 CONCLUSIONES

- En la fase de identificación de requerimiento, se evidenció que existen elementos en la parte de hardware y software que pueden servir para el desarrollo del presente proyecto, pero su adquisición se volvió dificultosa debido a que algunos elementos como los sensores de detección de radiación UV no se encuentran disponibles en el mercado para su fácil acceso.
- Para registrar un rendimiento óptimo en radiación UV, es fundamental contar con el hardware adecuado, como un sensor GUVA-S12SD y placas ESP32. El sensor de detección UV fue acorde con las características técnicas que posee la placa ESP32 para la lectura de niveles analógico y su calibración a niveles digitales, así mismo el ESP32 tuvo una eficiente conexión WI-Fi a la red de 2.4 (Ghz) que se estableció para la comunicación hacia el dashboard web. Además, se requiere de un software como Arduino Cloud, que ofrece una interfaz gráfica amigable para el usuario que muestre resultados claros.
- Para el diseño del presente prototipo se verificó que la programación de las placas ESP32 de desarrollo se tuvo que realizar una calibración idónea del sensor GUVA-S12SD debido a que lee únicamente datos analógicos. Por tanto, se realizó una conversión de valores analógicos a digitales que fueron eficientes para poder presentar en base a la escala de índice UV los niveles de radiación y posterior su impresión en la plataforma Arduino IoT Cloud.
- En la etapa de implementación del proyecto, se realizaron 2 baquelitas por motivo de demostración del protocolo ESP-NOW. Se pudo comprobar que, aunque las 2 baquelitas se encuentren separados una cierta distancia el funcionamiento del protocolo ESP-NOW seguían siendo eficiente en el envío de datos entre placas.
- Las pruebas funcionales realizadas son fundamentales para verificar la precisión y confianza del prototipo. Este proceso le permite identificar posibles errores y áreas de mejora, facilitando ajustes antes de su presentación. Se demostró que los niveles de radiación detectados se censaron correctamente, adaptando sus valores satisfactoriamente según los diferentes escenarios a los que fue expuesto el sensor. Los valores varían dependiendo de la exposición al sol por parte del sensor.
- El sensor de radiación UV revela que los niveles de radiación durante el medio día pueden alcanzar valores muy altos. Estos valores los presenta el prototipo mediante un LED de color rojo que se va a encender siempre que se encuentren

estos niveles de radiación, mientras que cuando la radiación se encuentre en niveles bajos se enciende un LED de color verde. Cuando el sensor está ubicado en la sombra, incluso si los niveles de radiación UV son altos en las áreas expuestas al sol, puede detectar pequeños niveles de radiación. En este caso, el sensor mostró un nivel de 2,71 UV.

5 RECOMENDACIONES

- En el desarrollo del prototipo de detección UV es importante tener en claro todos los elementos en la parte del hardware que son necesarios para su creación. En específico del sensor de detección UV debido a que dependerá de este elemento los valores que se obtenga, debido a que en caso de problemas este sensor que se elige debe tener un gran respaldo técnico para la solución de problemas.
- Realizar una evaluación exhaustiva de las opciones de hardware y software disponibles en el mercado, tomando en cuenta factores como costo y disponibilidad, para seleccionar las mejores opciones que cumplan con los requisitos.
- En el desarrollo del presente proyecto el uso de algunos elementos no previstos para la elaboración del prototipo como el uso de LEDs, te brinda una ayuda para verificar si el prototipo está trabajando de manera correcta para proceder a construir el dashboard web.
- Se recomienda que al momento de realizar las pruebas de funcionamiento se lo haga en momentos que se tenga sol en el clima, debido a que cuando se desarrolló este prototipo, se finalizó en horas de la noche y al no existir niveles de radiación UV se creyó que el prototipo estaba mal diseñado, pero cuando se probó en ambientes con sol se pudo comprobar su debido funcionamiento.
- La ejecución de pruebas rigurosas que incluyen diversas condiciones ambientales y escenarios de uso garantiza que el prototipo funcione correctamente en todas las situaciones esperadas.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] M. d. T. y. E. S. d. España, «Estudio de la exposición a radiación ultravioleta solar en buques pesqueros - Año 2022,» Instituto Nacional de Seguridad y Salud en el trabajo insst, 2022. [En línea]. Available: https://www.insst.es/documentacion/material-tecnico/documentos-tecnicos/estudio-exposicion-a-radiacion-uv-solar-en-buques-pesqueros-ano-2022. [Último acceso: 04 Julio 2024].

[2] M. Navarra, «Radiación ultravioleta,» navarra.es, [En línea]. Available: http://meteo.navarra.es/definiciones/radiacion_ultravioleta.cfm. [Último acceso: 04 Julio 2024].

[3] M. d. A. y. D. S. d. Colombia, «GENERALIDADES DE LA RADIACIÓN ULTRAVIOLETA,» IDEAM, [En línea]. Available: http://www.ideam.gov.co/web/tiempo-y-clima/generalidades-de-la-radiacion-ultravioleta. [Último acceso: 04 Julio 2024].

[4] L. E. S.A.U., «Rayos UV: ¿Qué son y cómo afectan a la piel?,» Laroche-posay, [En línea]. Available: https://www.laroche-posay.es/article/rayos-uv-que-son-y-como-afectan-a-la-piel. [Último acceso: 12 Julio 2024].

[5] NIVEA, «CALCULADORA UV NIVEA: CALCULA EL ÍNDICE DE RAYOS UV EN TU LOCALIDAD,» NIVEA, [En línea]. Available: https://www.nivea.com.ec/consejos/uv-calculator. [Último acceso: 07 Julio 2024].

[6] P. Flotanet, «Starter Kit, NodeMCU ESP32, versión DevKit V1, de 30 pines. Incluye 1 cable micro USB y 2 protoboard de 170 puntos. Desarrollo en Arduino IDE.,» puntoflotanet, [En línea]. Available: https://www.puntoflotante.net/NODEMCU-ESP32-DEVKIT-V1-STARTER-KIT.htm. [Último acceso: 16 Julio 2024].

[7] EINSTRONIC, «ESP32 DevKit V1,» EINSTRONIC, 2021. [En línea]. Available: https://einstronic.com/product/esp32-devkit-v1/. [Último acceso: 20 Julio 2024].

[8] D. Carrasco, «ESP-Now conecta dos o más ESP32/ESP8266,» ElectroSoftCloud, 14 Mayo 2021. [En línea]. Available: https://www.electrosoftcloud.com/esp-now-conecta-dos-o-mas-esp32-esp8266/. [Último acceso: 17 Julio 2024].

[9] O. Barajas, «Arduino IoT Cloud: La Nueva plataforma,» TdE, 27 Febrero 2019. [En línea]. Available: https://tiendadeelectronica.mx/blog/arduino-iot-cloud-la-nueva-plataforma/. [Último acceso: 18 Julio 2024].

[10] Electronilab, «GUVA-S12SD – Sensor de luz UV (luz ultravioleta) analógico,» Electronilab, [En línea]. Available: https://electronilab.co/tienda/guva-s12sd-sensor-de-luz-uv-luz-ultravioleta-analogico/. [Último acceso: 18 Julio 2024].

[11] OEM, «ESP32 Development Board - DEVKIT V1,» GROBOTRONICS, [En línea]. Available: https://grobotronics.com/esp32-development-board-devkit-v1.html?sl=en. [Último acceso: 19 Julio 2024].

[12] J. G. Carmenate, «ESP32 Wifi y Bluetooth en un solo chip,» programarfacil, 07 Enero 2024. [En línea]. Available: https://programarfacil.com/esp8266/esp32/. [Último acceso: 20 Julio 2024].

[13] N. MECHATRONICS, «MÓDULO SENSOR DE LUZ ULTRAVIOLETA (UV) GUVA-S12SD,» NAYLAM MECHATRONICS, [En línea]. Available: https://naylampmechatronics.com/sensores-luz-y-sonido/1221-modulo-sensor-de-luz-ultravioleta-uv-guva-s12sd.html. [Último acceso: 20 Julio 2024].

[14] ElectroStore, «SENSOR DETECTOR GY-ML8511 UV ULTRAVIOLETA,» GrupoElectrostore, [En línea]. Available: https://grupoelectrostore.com/shop/sensores/luzuv/sensor-detector-gy-ml8511-uv-ultravioleta/. [Último acceso: 25 Julio 2024].

[15] BeJob, «Qué es la programación con arduino y para qué sirve,» BeJob, 14 Febrero 2017. [En línea]. Available: https://bejob.com/que-es-la-programacion-con-arduino-y-para-que-sirve/. [Último acceso: 21 Julio 2024].

[16] J. C. J. F. Ángel Rodrigues, «Análisis comparativo de plataformas IoT para el desarrollo de,» Working Papers - ECBTI, [En línea]. Available: https://hemeroteca.unad.edu.co/index.php/wpecbti/article/view/6779. [Último acceso: 21 Julio 2024].

7 ANEXOS

F_AA_236

CERTIFICADO DE ORIGINALIDAD

TRABAJO DE INTEGRACIÓN CURRICULAR

Quito, D.M. 01 de agosto de 2024

De mi consideración:

Yo, **CARLOS ANDRÉS YUNGA SÁNCHEZ**, en calidad de Director del Trabajo de Integración Curricular titulado **IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA DETECCIÓN DE RADIACIÓN UV A TRAVÉS DEL PROTOCOLO ESP-NOW EN UN DASHBOARD WEB** asociado al **IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA DETECCIÓN DE RADIACIÓN UV A TRAVÉS DEL PROTOCOLO ESP-NOW EN UN DASHBOARD WEB** elaborado por el **KEVIN BRYAN COELLO NUÑEZ** de la carrera en **TECNOLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES**, certifico que he empleado la herramienta antiplagio Turnitin para la revisión de originalidad del documento escrito producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 10%.

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 informe generado por la herramienta Turnitin.
3. TIC_KCoello_Reporte_turnitin.pdf

Atentamente,

CARLOS ANDRÉS YUNGA SÁNCHEZ

Técnico Docente

Escuela de Formación de Tecnólogos

ANEXO II: Enlaces

Se debe colocar los códigos QR o enlaces de videos demostrativos, etc

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

ANEXO III: Códigos Fuente

ANEXO I. Código de programación

Código del Emisor

```
#include <esp_now.h>

#include <WiFi.h>


const char* ssid = "CELERITY_WLAN802";

const char* password = "LAURA1304413345";


const int uvSensorPin = 34;  // Pin analógico al que está conectado el sensor UV


// Reemplazo con la mac de la placa receptor

uint8_t mac_receptor[] = { 0xE4, 0x65, 0xB8, 0x7A, 0xAF, 0x08 };


// Estructura de envio de datos

typedef struct struct_message {

int lectura;

} struct_message;


struct_message mensaje;


esp_now_peer_info_t peerInfo;


// Función de callback que se ejecuta cuando se envían datos

void OnDataSent(const uint8_t* mac_addr, esp_now_send_status_t status) {
```

```
  Serial.print("Estado de envío: ");
  if (status == ESP_NOW_SEND_SUCCESS) {
    Serial.println("Enviado");
  } else {
    Serial.println("Fallo");
  }
}

void setup() {
  // Inicializar monitor serial
  Serial.begin(115200);
  analogReadResolution(9);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  // Esperar a que nos conectemos
  while (WiFi.status() != WL_CONNECTED) {
    delay(200);
    Serial.print('.');
  }

  Serial.println();
  Serial.print("Conectado a:\t");
  Serial.println(WiFi.SSID());
  Serial.print("Dirección IP:\t");
  Serial.println(WiFi.localIP());
```

```
  delay(1500);

  // Inicializar ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error al inicializar ESP-NOW");
    return;
  }

  esp_now_register_send_cb(OnDataSent);

  memcpy(peerInfo.peer_addr, mac_receptor, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // Añadir cliente ESP-NOW
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Falla al añadir cliente");
    return;
  }
}

void loop() {
  // Leer el valor del sensor UV
  int uvValue = analogRead(uvSensorPin);

  mensaje.lectura = uvValue;
```

```
  esp_err_t result = esp_now_send(mac_receptor, (uint8_t*)&mensaje, sizeof(mensaje));


  if (result == ESP_OK) {
    Serial.println("Enviado con éxito");
  } else {
    Serial.println("Error al enviar los datos");
  }
  delay(2000);
}
```

Código del Receptor

```
#include <esp_now.h>
//#include <WiFi.h>
//#include <WiFiClientSecure.h>
#include "thingProperties.h"
//#include <ArduinoJson.h>

const int verde = 5;
const int amarillo = 18;
const int rojo = 19;

// const char* ssid =
// const char* password = "095017068@[_Da";

float uvIndex = 0;
bool aux = false;
```

```
typedef struct struct_message {
  int lectura;
} struct_message;

struct_message mensaje;

// Función de callback que se ejecuta cuando se reciben datos
void OnDataRecv(const uint8_t* mac, const uint8_t* incomingData, int len) {
  memcpy(&mensaje, incomingData, sizeof(mensaje));
  Serial.print("Radiación UV recibida por ESP-NOW: ");
  Serial.println(mensaje.lectura);
  leer_indice_uv(mensaje.lectura);
}

void setup() {
  // Inicializar monitor serial
  Serial.begin(115200);
  analogReadResolution(9);

  //WiFi.mode(WIFI_STA);
  initProperties();

  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  Serial.println();
```

```
  Serial.print("Conectado a:\t");
  Serial.println(WiFi.SSID());
  Serial.print("Dirección IP:\t");
  Serial.println(WiFi.localIP());

  delay(1500);
  // Inicializar ESP-NOW
  // if (esp_now_init() != ESP_OK) {
  //   Serial.println("Error al inicializar ESP-NOW");
  //   return;
  // }

  // Una vez que ESPNow se inicializa correctamente, registraremos la función de callback para recibir datos
  //esp_now_register_recv_cb(esp_now_recv_cb_t(OnDataRecv));
  pinMode(verde, OUTPUT);
  pinMode(amarillo, OUTPUT);
  pinMode(rojo, OUTPUT);

  digitalWrite(verde, LOW);
  digitalWrite(amarillo, LOW);
  digitalWrite(rojo, LOW);
}

void loop() {
  //esp_now_register_recv_cb(esp_now_recv_cb_t(OnDataRecv));
  if (ArduinoCloud.connected() && !aux) {
```

```
    if (esp_now_init() != ESP_OK) {
      Serial.println("Error al inicializar ESP-NOW");
      //return;
    }
    aux = true;
  }
  ArduinoCloud.update();

  delay(1500);
  esp_now_register_recv_cb(esp_now_recv_cb_t(OnDataRecv));
  delay(1000);
}

void leer_indice_uv(int analogico) {
  float voltaje = analogico * (3.3 / 1023.0);  // Convertir valor analógico a voltaje
  uvIndex = voltaje / 0.1;                     // Ejemplo de fórmula genérica para el sensor
  // Imprimir el valor del índice UV en el monitor serial
  Serial.print("Índice UV: ");
  Serial.println(uvIndex);
  indiceUV = uvIndex;

  // Niveles de radiación: https://es.wikipedia.org/wiki/%C3%8Dndice_UV
  if (uvIndex >= 0 && uvIndex < 3) {
    Serial.println("Riesgo: Bajo");
    riesgoBajo = true;
    digitalWrite(verde, HIGH);
    digitalWrite(amarillo, LOW);
```

```
  digitalWrite(rojo, LOW);
} else if (uvIndex >= 3 && uvIndex < 5) {
  Serial.println("Riesgo: Moderado");
  riesgoBajo = true;
  digitalWrite(verde, LOW);
  digitalWrite(amarillo, HIGH);
  digitalWrite(rojo, LOW);
} else if (uvIndex >= 5 && uvIndex < 7) {
  Serial.println("Riesgo: Alto");
  riesgoBajo = false;
  digitalWrite(verde, LOW);
  digitalWrite(amarillo, LOW);
  digitalWrite(rojo, HIGH);
} else if (uvIndex >= 7 && uvIndex < 10) {
  Serial.println("Riesgo: Muy alto");
  riesgoBajo = false;
  digitalWrite(verde, LOW);
  digitalWrite(amarillo, LOW);
  digitalWrite(rojo, HIGH);
} else if (uvIndex >= 10) {
  Serial.println("Riesgo: Extremadamente alto");
  riesgoBajo = false;
  digitalWrite(verde, LOW);
  digitalWrite(amarillo, LOW);
  digitalWrite(rojo, HIGH);
}
delay(1000);
```

```
}
```

Código del Receptor a Arduino IoT Cloud

```
#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>

const char DEVICE_LOGIN_NAME[] = "3730afba-89de-41fd-b425-aec8da58474b";

const char SSID[] = "CELERITY_WLAN802";    // Nombre de la Red Wifi
const char PASS[] = "LAURA1304413345";  // Clave de la red Wifi

const char DEVICE_KEY[] = "ZkeD4MbnOv0F@Zby8efm#aeTd";  // Clave secreta

float indiceUV;
bool riesgoBajo;

void initProperties() {

  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
  ArduinoCloud.addProperty(indiceUV, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(riesgoBajo, READ, ON_CHANGE, NULL);
}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```