ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO DE ALERTA DE DETERIORO DE LA CALIDAD DEL AIRE POR MEDIO DE *TELEGRAM* Y DESPLIEGUE DE INFORMACIÓN EN UN *DASHBOARD* WEB

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

HENRY ANDRES CALVA ABAD

henry.calva@epn.edu.ec

DIRECTOR: LEANDRO ANTONIO PAZMIÑO ORTIZ

leandro.pazmino@epn.edu.ec

DMQ, agosto 2023

CERTIFICACIONES

Yo, Henry Andres Calva Abad 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.

HENRY CALVA

henry.calva@epn.edu.ec

henryandrescalva@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Henry Andres Calva Abad bajo mi supervisión.

Leandro Antonio Pazmiño Ortiz
DIRECTOR

leandro.pazmino@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.

Henry Andres Calva Abad

CI: 172868572-6

DEDICATORIA

Este proyecto está dedicado a Dios porque a pesar de los múltiples obstáculos que se presentan en mi camino, me demostró que existe una salida que quizás no era la menos complicada pero tampoco imposible de superar.

A mis padres Roberto y Margarita, cuyo apoyo incondicional y sacrificio me demostraron ser guías de mi vida, siendo ellos los pilares fundamentales de mis logros. Su constante amor y aliento me han inspirado a superar obstáculos y alcanzar nuevas alturas.

Así mismo a mis hermanas Yuri y Camila, quienes han sido fuente de alegría y compañía en todo momento, compartiendo cada paso de nuestro camino. De la misma manera a mi hermano Vinicio quien con su comprensión y paciencia ha logrado instruirme buenos valores basándose en el respeto, gratitud, responsabilidad y honestidad. El apoyo de mi familia es invaluable siendo de esta dedicación un reconocimiento de la importancia que tienen en mi vida.

Finalmente, mas no por ello menos importante a mi enamorada Ivette y mis amigos Anthony y Joan quienes demuestran cada día y ante cualquier obstáculo que el ser humano puede superar con valor, sacrificio y dedicación.

Henry Andres Calva Abad

AGRADECIMIENTO

Agradezco a Dios por ser mi guía y demostrarme que todo el esfuerzo después de todo al final siempre dará frutos.

Agradezco infinitamente a mis Padres Roberto C. y Margarita A. porque con su esfuerzo me brindaron sus enseñanzas, la oportunidad de estudiar y ser mis guías de vida.

A mis hermanos Yuri, Camila y Vinicio porque nunca me falto su apoyo en los momentos más complicados, los quiero mucho.

A mi enamorada Ivette P. porque su apoyo ha sido muy importante, a pesar de los días más difíciles siempre me motivo a seguir adelante y perseverar hasta alcanzar los objetivos.

A mis amigos, quienes han compartido risas y recuerdos invaluables les extiendo mi gratitud. Puesto que su amistad ha sido el mejor regalo que ha enriquecido mi vida de formas únicas e inexplicables.

Henry Andres Calva Abad

ÍNDICE DE CONTENIDOS

RESUMEN

El presente trabajo de titulación expone de manera detallada un prototipo para alertar con respecto a la calidad del aire, emitida través de la plataforma de mensajería *Telegram y un Dashboard* web. Esta alerta tiene la finalidad de informar a las personas que realizan actividades deportivas en las primeras horas del día, en circunstancias en las cuales resulta complicado discernir si las condiciones del aire son adecuadas con el fin de pretender que las personas tomen medidas preventivas antes de llevar a cabo la rutina.

Dicho sistema está compuesto por un módulo NodeMCU ESP32, la cual incorpora un chip que posibilita la conectividad inalámbrica orientada a *IoT*. También se integra un sensor GP2Y1051, junto con un módulo relé, todos ellos alojados en el interior de una carcasa de plástico.

El documento se organiza en cuatro secciones principales. En la primera, se aborda la problemática generada por la falta de conocimiento sobre partículas PM 2.5 presentes en el aire responsables de la generación de enfermedades respiratorias. Además de detallar el modo de trabajo del prototipo, se expone los objetivos, alcance y se describe los datos técnicos.

En la segunda sección se describe la metodología empleada para elaborar y desarrollar de forma adecuada el prototipo según los objetivos planteados para este proyecto con el fin de cumplir cada uno de ellos. En la tercera sección está dirigido a los resultados obtenidos durante la fase de pruebas del prototipo final.

Finalmente, en la cuarta sección, se menciona las conclusiones y se proponen recomendaciones para futuras mejoras del prototipo.

PALABRAS CLAVE: ESP32, sensor GP2Y1051, PM 2.5, *Iot*, *Telegram*, *Dashboard*.

ABSTRACT

This degree work presents in detail a prototype for an alert regarding air quality, issued through the Telegram messaging platform and a web Dashboard. This alert is intended to inform people who perform sports activities in the early hours of the day, in circumstances in which it is complicated to discern whether the air conditions are adequate in order to pretend that people take preventive measures before carrying out the routine.

The system consists of an ESP32 *NodeMCU module, which incorporates a chip that enables IoT-oriented wireless connectivity. A GP2Y1051 sensor is also integrated, along with a relay module, all housed inside a plastic casing.*

The paper is organized into four main sections. In the first one, it addresses the problem generated by the lack of knowledge about PM 2.5 *particles present in the air responsible for the generation of respiratory diseases. In addition to detailing the working mode of the prototype, the objectives, scope and technical data are described.*

The second section describes the methodology used for the development of the prototype according to the objectives set for this project in order to meet each of them. The third section presents the results obtained, together with the design, implementation and performance tests of the final prototype.

Finally, in the fourth section, the conclusions derived from the results are discussed and recommendations for future improvements are proposed.

KEYWORDS: ESP32*, GP2Y1051 sensor,* PM 2.5*, Iot, Telegram, Dashboard.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La implementación de este componente implica desarrollar un sistema capaz de evaluar y emitir alertas sobre la calidad de aire en un entorno que rodea al ser humano. Esto se logra utilizando sensores especializados capaces de detectar partículas en suspensión como: aerosoles microscópicos, partículas de polvo y gotas de agua [1]. Una vez que estas partículas son detectadas, se activa una alerta la misma que será notificada mediante una plataforma de mensajería en línea y a la vez registrar la información en un *Dashboard* web.

El objetivo principal es proporcionar a los usuarios información en tiempo real sobre la calidad del aire y las concentraciones de partículas diminutas que afectan a la salud, manteniendo una relación directa con enfermedades respiratorias y trastornos cardiorrespiratorios, con el fin de tomar medidas para el respectivo cuidado. Para lograr esto, se utilizarán sensores especializados que puedan detectar la presencia de partículas PM 2.5 [1]

Las alertas se envían a través de *Telegram*, una plataforma de mensajería instantánea ampliamente disponible y confiable. *Telegram* se selecciona como canal de comunicación debido a su capacidad para enviar mensajes de manera rápida y eficiente. Las alertas contienen información relevante sobre la clasificación de la calidad del aire en categorías como: buena, moderada, mala, peligrosa y muy peligrosa, lo que ayuda a las personas a tomar precauciones.

Estos sensores estarán conectados a un sistema central que procesará los datos recopilados y generará alertas cuando se superen ciertos niveles predefinidos. Una vez activada la alerta, se enviará automáticamente un mensaje a través de *Telegram,* notificando a los usuarios sobre la situación.

Además de las notificaciones en *Telegram,* este componente implica la presentación de información a través de un *Dashboard* web. Un *Dashboard* es una interfaz visual que ofrece datos en tiempo real y resúmenes pertinentes de estado. En este contexto, el *Dashboard* web ofrece detalles acerca de la cantidad de partículas asociadas a la calidad de aire. Esta funcionalidad permite a los usuarios acceder y analizar con facilidad los datos recopilados, brindándoles la capacidad de tomar decisiones fundamentadas.

El prototipo presenta dos puntos de control completo. Por un lado, podrán interactuar con él mediante la aplicación de mensajería *Telegram*. Por otro, tendrán acceso y control a través de un panel de control en un *Dashboard* web. Este enfoque dual garantiza que

los usuarios puedan gestionar y monitorear el prototipo de manera cómoda y efectiva, adaptándose a sus preferencias y necesidades.

1.1 Objetivo general

Implementar un prototipo de alerta de deterioro de la calidad del aire por medio de *Telegram* y despliegue de información 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 del sistema de alerta.
- Implementar el prototipo en una maqueta.
- Realizar pruebas de funcionamiento del prototipo.

1.3 Alcance

A través de este proyecto se quiere llevar a cabo un sistema de alarma que sirva para alertar el deterioro de la calidad del aire a personas que realizan deporte en horas de la mañana, de tal manera que puedan tomar acciones preventivas antes de realizar la práctica deportiva. Adicionalmente, tendrá la capacidad de llevar a cabo las siguientes acciones:

- Sensar la presencia de partículas peligrosas en el aire.
- Alertar sobre el deterioro de la calidad de aire por medio de *Telegram*.
- Desplegar la información en un tablero (*Dashboard*) Web.

1.4 Marco Teórico

ESP32

El *Espressif System Platform 32-bit* (ESP32*)* es un módulo que contiene un grupo de microcontroladores los cuales poseen bajo costo y consumo, su principal ventaja es la integración de *Wi-Fi* 802.11 b/g/n y *Bluetooth v4.2,* su popularidad incrementó debido a las implementaciones en trabajos relacionados con el Internet de las cosas *(IoT),* también cuenta con múltiples entornos para trabajar en el desarrollo con el código, así como una variedad de librerías que permite expandir las posibilidades de aplicaciones [2].

Sensores

Los sensores son los encargados de responder en función a la información que se encuentre en el entorno, además de ser usados para herramientas de automatización, estos pueden ser clasificados en función a las características que estos son capaces de brindar. De acuerdo con su aplicación existen sensores capaces de detectar fluidos, corriente eléctrica, partículas, entre otros [3].

Botfather

Dentro de la aplicación de *Telegram* se encuentra la herramienta *Botfather* esta facilita la creación de *bots* los cuales son programas usados cuando existen tareas que son repetitivas o que no se pueden iniciar de manera manual dentro de la aplicación *Telegram*, estos *bots* son capaces de brindar varias herramientas útiles entre ellas las alertas mediante el envío de notificaciones, monitoreo web, entre otras [4].

Los *bots* de *Telegram* posibilitan la comunicación con cada uno de los usuarios para llevar a cabo tareas programables que fueron preconfiguradas en un inicio y posteriormente utilizarlas de forma adecuada a través del chat. Cada acción y respuesta echa por el *bot* se basa en el propósito para el que fue creado. Además, no necesitan ser instalados adicionalmente, ya que solo necesitan ser inicializados mediante el chat de *Telegram* [5].

Dashboard

Esta herramienta permite monitorear, analizar y gestionar los indicadores de desempeño en los cuales se encuentran datos fundamentales y son útiles para realizar el seguimiento de cualquier actividad en específico, dicha información se mostrará en forma de resumen y en un solo lugar, lo que facilita su comprensión y análisis, es importante tener en cuenta que la información mostrada también puede ser en tiempo real [6].

Arduino Cloud

Es una plataforma usada para la aplicación IoT, la cual cuenta con un tablero o *Dashboard* facilitando la implementación y control de aplicaciones en IoT, además con esta herramienta facilita la creación de interfaces web y móviles para poder interactuar de una manera más sencilla con los proyectos, entre las características principales se encuentran: el uso de los servicios en la nube lo que permite un acceso más rápido y acceso remoto mediante las interfaces antes mencionadas [7].

Modulo Relé

Un relé es denominado un dispositivo electro-mecatrónico, fabricado para actuar como interruptor. Un módulo relé se compone de varios transistores y resistencias además de un solo relé. Esto indica si el módulo está alimentado, además de mostrar si está activo o no, lo que permite facilitar la conexión con otros componentes electrónicos y controlar una corriente mucho mayor de manera segura [8].

2 METODOLOGÍA

En primer lugar, se analizaron y consideraron detenidamente los requerimientos tanto de *hardware* como de *software.* En lo que respecta al *hardware* se consideraron los componentes físicos que forman parte del prototipo principalmente basándose tanto en el tipo de sensor como también el tipo de microcontrolador.

Para el *software se* plantea la aplicación de mensajería instantánea destinada a controlar las lecturas del sensor y emitir las alertas pertinentes, además de una interfaz web con la función de habilitar o deshabilitar las lecturas del sensor, además de presentar de manera comprensible al usuario los datos recopilados y relevantes.

En base al análisis inicial sobre cómo debe trabajar el prototipo para emitir las alertas, se determinó según el Indicie de Calidad de Aire (ICA) los niveles sobre la calidad de aire y en que rangos se debe enviar dichas alertas. Esto según estudios establecidos y aprobados por la Organización Mundial de la Salud.

Para el diseño del prototipo se encuentra el esquema de conexiones entre los elementos como son el módulo, la conexión *Wi-Fi*, la aplicación de mensajería que recibirá la alerta y la interfaz web que también mostrará la información, se encuentran además las configuraciones que son necesarias para el correcto uso del *bot* de *Telegram*, de esta forma el prototipo logra cumplir con todos los objetivos planteados.

Dentro de la implementación del prototipo se integraron los módulos y sensores junto con la aplicación de mensajería e interfaz web en este caso *Arduino Cloud* y se verificó el funcionamiento del mismo con el fin de comprobar que ningún elemento produzca errores en sus resultados y las tablas de registro de datos.

Finalmente, las pruebas de funcionamiento se basaron en los objetivos planteados para este proyecto y con el propósito de comprobar si ofrece un desempeño correcto, dichas pruebas son esenciales para corregir errores en caso de existir y de esa forma garantizar que los requerimientos del proyecto se encuentren cumplidos.

3 RESULTADOS

En este apartado se expone el procedimiento para el correcto funcionamiento del prototipo, además de la elección de los diversos componentes empleados para su implementación. Además, se describe el proceso de diseño sin dejar de lado su enfoque y presentar paso a paso el proceso de la implementación con el fin de obtener el producto final cumpliendo adecuadamente los puntos establecidos para este proyecto.

Esto toma en cuenta la fabricación de la placa electrónica, instalación de algunos componentes electrónicos, programación del módulo ESP32 en el entorno de desarrollo de *Arduino Cloud*. Explicando detalladamente el código desarrollado. Finalmente se presentan las pruebas realizadas según su funcionamiento.

El prototipo trabaja mediante la detección de partículas PM 2.5 suspendidas en el aire y capta la información a través del sensor GP2Y1051, al detectar partículas mediante los fotodetectores ópticos transmite señales a la placa del ESP32 con los datos obtenidos en tiempo real de tal manera que este los traduce en indicadores sobre la calidad de aire. Después de procesar la información, se emite una alerta mediante el servicio de mensajería de *Telegram* y también en su *Dashboard* web, registrando la información obtenida.

3.1 Identificación de los requerimientos para la implementación del prototipo

Respecto a la investigación previa realizada, se identificó los requerimientos esenciales para la ejecución del diseño del prototipo. Esto fue posible mediante la utilización de sensores de partículas y microcontroladores de tarjetas inalámbricas. Estos componentes a su vez se encuentran interconectadas mediante la red *Wi-Fi* del hogar.

El propósito principal de esta configuración es notificar al usuario acerca de la calidad de aire lo cual se lleva a cabo a través de la aplicación de mensajería instantánea y un *Dashboard* web, Además de habilitar y deshabilitar las lecturas del sensor desde ambas partes.

Selección de *Hardware* del controlador

Para la respectiva selección del controlador electrónico se determinó el uso indispensable de una plataforma de código abierto el cual permite realizar combinaciones de *hardware*. Reconocida principalmente por su flexibilidad y facilidad de uso, lo que la hace mayormente conocida entre los creadores y desarrolladores [9]. Es posible construir microcontroladores de placa única y aprovechar una comunidad activa de creadores para darles diversos usos y aplicaciones.

Implementación de *Software* libre en la nube

Para este requerimiento es necesario un *software* de código abierto, el mismo que pueda mantenerse en línea y ofrezca al usuario las facilidades de su libre uso de forma ilimitada, además de ser gratuito. Así también, de la misma manera que permita conectar dispositivos para lograr la interoperabilidad, facilitando la comunicación efectiva y coordinada, mejorando de forma óptima su funcionamiento [10].

Programación de placas automáticas para controlar y gestionar dispositivos

La automatización y sus respectivas rutinas son necesarias en el servidor *IoT,* de tal manera que el dispositivo inteligente seleccionado de forma adecuada tenga un centro de control independientemente de la marca o tipo, con el fin de disminuir el manejo de forma manual. Con ello de forma analítica se crea la función de los dispositivos, generalmente es el control de apagado y encendido del dispositivo.

Selección de dispositivos compatibles con el servidor IoT

El prototipo demanda un componente electrónico inteligentes que se controlan de forma inalámbrica mediante la conexión de una red *Wi-Fi*. Con la finalidad de recoger datos y alertar al usuario de tal manera que lo mantenga informado. también su precio debe ser cómodo y asequible. Por esta razón se investiga las marcas disponibles y su disponibilidad en el mercado.

Definición de parámetros del *bot*

El *bot* de *Telegram* al ser considerado un centro de control de lecturas del sensor, se encuentra vinculado a un único usuario a través de las credenciales del chat id del mencionado usuario. Aunque cumple funciones de notificación e información en general, es imprescindible que solamente un usuario pueda mantener el control total sobre el *bot*.

Por lo tanto, es necesario que este usuario se autentifique utilizando las credenciales correspondientes para activar o desactivar las lecturas del sensor según sea necesario. Esta medida garantiza que solo el usuario autorizado tenga el poder de controlar las funcionalidades del *bot* en relación con el sensor.

3.2 Selección de *hardware* y *software* acorde a los requerimientos del prototipo

Selección del *hardware*

El modelo de desarrollo requiere dos dispositivos electrónicos inteligentes que se manejan de forma inalámbrica a través de una red *Wi-Fi* del hogar, a fin de obtener y recolectar los datos, posteriormente deben ser notificado al usuario para mantenerlo informado.

No obstante, la selección de modulo del sensor se encuentra basado en la capacidad de detección de partículas pequeñas y no susceptibles al ojo humano. Para ello se ha encontrado varios modelos de un especifico sensor, como por ejemplo el GPY1010AU, GP2Y1051. Estos dos tipos de sensores pueden llegar a ser similares, pero no en todos los aspectos.

El sensor GPY1051 logra destacar la reducción de cables en sus conectores, de 6 a 3 ya que la comunicación con 3 de sus componentes electrónicos ya vienen incorporados. Por ende, la conexión a la placa principal es mucho más sencilla. Otro ejemplo es su salida de Transmisión (Tx), ya que cuenta con una salida analógica en sus primeras versiones, pero en la segunda versión su entrada es digital, es por esta razón que la sensibilidad para la detección de partículas PM 2.5 del aire es más elevada [11], [12].

Los sensores pueden llegar a trabajar en temperaturas que van de -10 (°C) hasta los 65 (°C). Cuando se llega a mejorar los sensores, versiones anteriores ya no se encuentran disponibles, pero las nuevas versiones presentan mejorías tanto que pueden llegar a simplificarse y trabajar de forma más eficiente [12].

Otro tipo de sensor es el DSM501 el cual se caracteriza por no permitir trabajar a detalle, un claro ejemplo es la detección de partículas, únicamente puede lograr detectar hasta 1 microgramo, cabe recalcar que para detectar las partículas diminutas estos deben lograr detectar partículas menos de 2,5 micrómetros de diámetro, esto según la Oficina de Evaluación de Peligros para la Salud Ambiental de California o mayormente conocida por sus siglas en inglés (*OEHHA*) [13]. Sin embargo, su precio redondea los $50.00 y

solo se puede obtenerlo bajo pedido en páginas extranjeras, por el momento se encuentra agotado y descontinuado por sus fabricantes [14].

En la Tabla 3.1 se presenta la comparación las características importantes de los sensores.

Tabla 3.1 Comparación entre detectores de partículas [12], [14]

	Sensor DSM501	**Sensor GP2Y1051**
Dimensiones	5,9 (cm) largo x 4,5 (cm) ancho	4,8 (cm) largo x 43,2 (cm) ancho
Consumo de corriente	90 (mA)	20 (mA)
Temperatura de funcionamiento	-10 (ºC) a 65 (ºC)	-10 (ºC) a 65 (ºC)
Límite mínimo de detención	3.33 (ug/m^3)	3.33 (ug/m^3)

Para elegir el módulo de *Wi-Fi*, se ha determinado contar con el chip ESP32. El cual cuenta con más de un único modelo los cuales son ESP01, ESP12 y ESP32. Siendo el ESP32 el indicado ya que presenta una alta velocidad de transmisión de 150 (Mbps), lo necesario para trabajar con la tecnología enfocada al *IoT*, además de constar con microcontroladores que facilitan su programación [15].

En cuanto al uso de voltaje para su funcionamiento, puede trabajar con 5 (V) por su entrada USB, al igual que ocupar pines con entradas de 3.3 (V) y 5(V). Sus entradas de alimentación cuentan con resistencias las cuales permiten controlar de cierta forma el voltaje que ingresa, a continuación, en la Tabla 3.2 se logra contemplar la comparación entre los dos tipos de microcontroladores, haciendo énfasis en la velocidad de transmisión [16].

Tabla 3.2 Comparación de microcontroladores [16] [17]

	ESP8266	**ESP32**
Velocidad de transmisión	72,2 (Mbps)	150 (Mbps)
Pines (A/D)	22	32
Voltaje de Operación	3,3 – 5 (V)	3,3 – 5 (V)
Velocidad de reloj	240 (MHz)	160 (MHz)

3.3 Diseño del prototipo *IoT*

Esquema general del proyecto

El grafico de la Figura representa de forma general la manera en la que el sensor se logra comunicar con el tablero web, mediante la placa ESP32 compatible con *Arduino Cloud*, permitiendo supervisar y gestionar de forma centralizada las lecturas del sensor, siendo programado en lenguaje C y manteniendo una comunicación con el *Dashboard.* El prototipo creado garantiza el cumplimento de los objetivos establecidos.

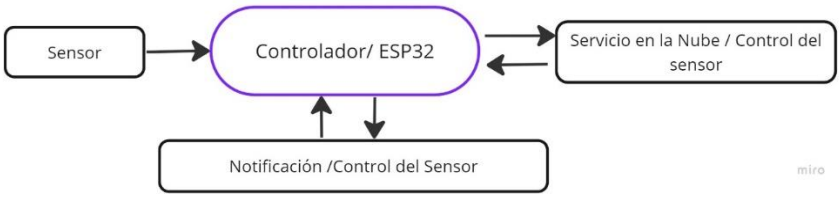

Figura 3.1 Esquema centralizado del prototipo

A partir del esquema proporcionado, la Figura 3.1 representa el método por el cual el prototipo se organiza. El ESP32 actúa como sistema de conexión y gestión de las lecturas del sensor. Se conecta de forma inalámbrica a la red *Wi-Fi* del hogar, lo que permite la intercomunicación con el servidor web de Arduino Cloud y la actualización del panel de control (*dashboard*).

Además, se consideran las notificaciones en la programación en lenguaje C de *Arduino Cloud*, lo que permite alertar al usuario sobre la calidad del aire. El funcionamiento es automático y, según las condiciones del aire en el entorno, se envía una alerta con información relevante.

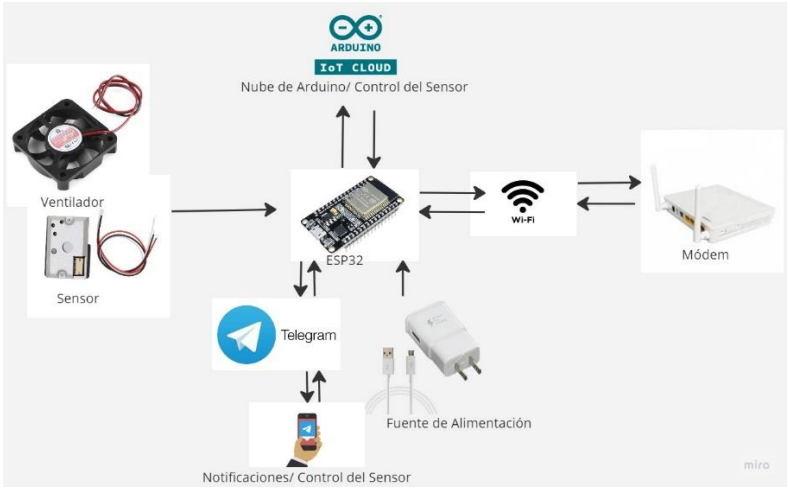

Figura 3.1 Esquema general del prototipo

Creación del *Bot* en *Telegram*

Para desarrollar el *bot* en *Telegram* y permitir la asistencia de notificaciones e interacción entre la placa ESP32, el sensor y el usuario, se utiliza la herramienta *Botfather*. El proceso comienza asignándole un nombre al *bot,* el cual debe finalizar con la palabra "*Bot*". En este caso específico, el *bot* se denomina "Estado_del_Aire-ESP32_BOT", tal como se indica en la Figura 3.2. Esta figura proporciona la guía visual para llevar a cabo la configuración y asignación del nombre al *bot* en *Telegram*

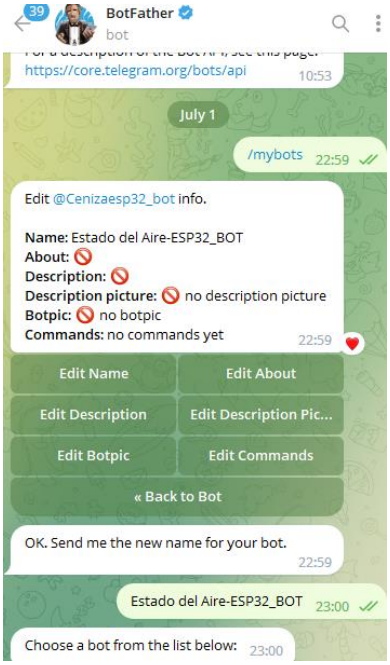

Figura 3.2 Creación del bot en *Telegram*

En la Figura 3.3 se presenta el uso de otro *bot* que permite obtener el identificador único, mayormente conocido por sus siglas en inglés (*ID*) del chat de usuario. Este *bot* proporciona una funcionalidad específica para obtener el *ID* del chat, asignado a cada conversación con un usuario en *Telegram.* También se muestra los pasos necesarios para utilizar este *bot* y obtener el *ID* del chat correspondiente.

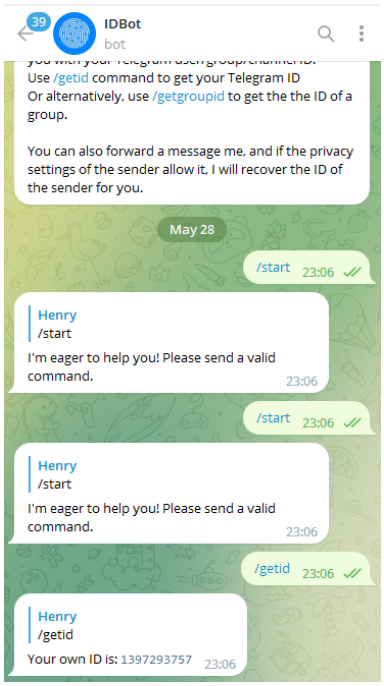

Figura *3.3* Obtener el *ID* de usuario en *Telegram*

En *Arduino Cloud* se realiza la configuración correspondiente en lenguaje C para permitir la interacción con el *chat* de usuario, con el objetivo de notificar las alertas respectivas sobre la calidad del aire. Se pueden observar datos importantes en la Figura 3.4 y Figura 3.5, donde se muestra cómo se almacenan los datos obtenidos, como el *Token* y el *ID*, para su configuración adecuada.

```
// Token del Bot y ID de Telegram
const String BOT_TOKEN = "5718887550:AAFyiRl3Dx-Qz-npstp2C-t5cd6ObUeX_iw";
const String CHAT_ID = "1397293757";
```

Figura 3.4 *ID* del chat de usuario y Token en el código de programación Arduino

```
//Configuracion para conectarse a Telegram mediante al red Wi-Fi
  bot_didactico.wifiConnect(SSID, PASS);
  bot_didactico.setTelegramToken(BOT_TOKEN);
  if (bot_didactico.testConnection()) {
    Serial.println("Conectado a Telegram");
  } else {
    Serial.println("Error de conexion");
  }
```

Figura 3.5 Configuración de conexión de *Telegram* y envío de notificaciones

Conexión microcontrolador ESP32 con *Arduino Cloud*

Para conectar la placa ESP32 a la plataforma de *Arduino Cloud* y realizar su programación correspondiente en lenguaje C, es necesario seguir ciertos pasos. En

primer lugar, se debe seleccionar el tipo de dispositivo y, en este caso específico, se muestra el modelo de la placa en la Figura 3.6.

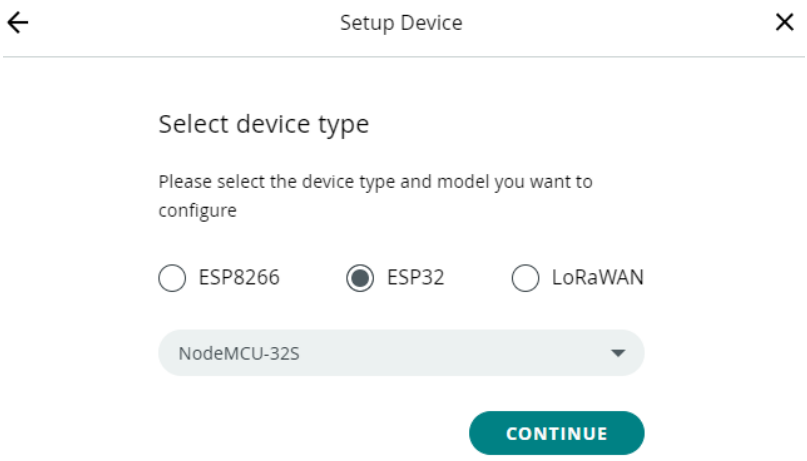

Figura 3.6 Selección del tipo de dispositivo en *Arduino Cloud*

A través de la Figura 3.7 se puede observar que después de completar los pasos necesarios, los datos del dispositivo aparecen junto con el nombre asignado.

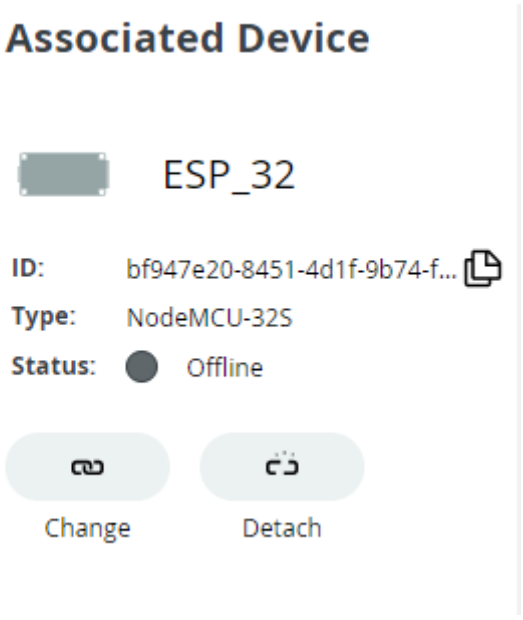

Figura 3.7 Datos del dispositivo asociado a *Arduino Cloud*

Creación del *Dashboard* web

Arduino Cloud presenta una gran cantidad de *widgets* disponibles. En la Figura 3.8 se logra visualizar algunos de ellos. Para la creación del *Dashboard* se utiliza *widgets* como *Swicth*, *Status*, *Stick Note*, *Gauge*, *Slider* y tablas de registro de datos.

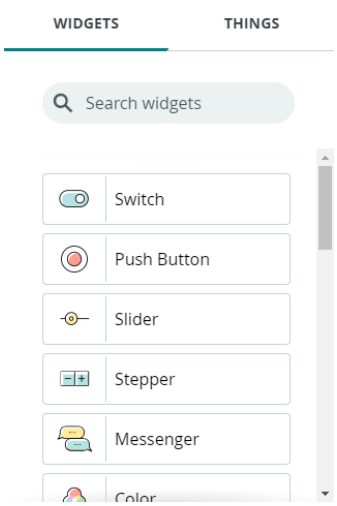

Figura 3.8 Widgets disponibles en *Arduino Cloud*

Arduino Cloud al ser una interfaz amigable con el usuario, permite crear un *Dashboard* simplemente asignándole a que variable desea asignar cada *widget*, y ubicarlos en la tabla únicamente ubicándolos a gusto del creador, tal como se indica en la Figura 3.9 y Figura 3.10.

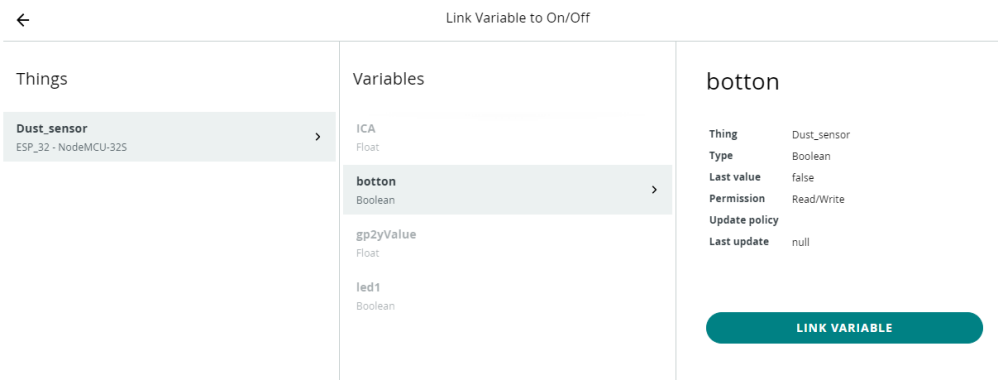

Figura 3.9 Asignación de Variables a widgets del tablero

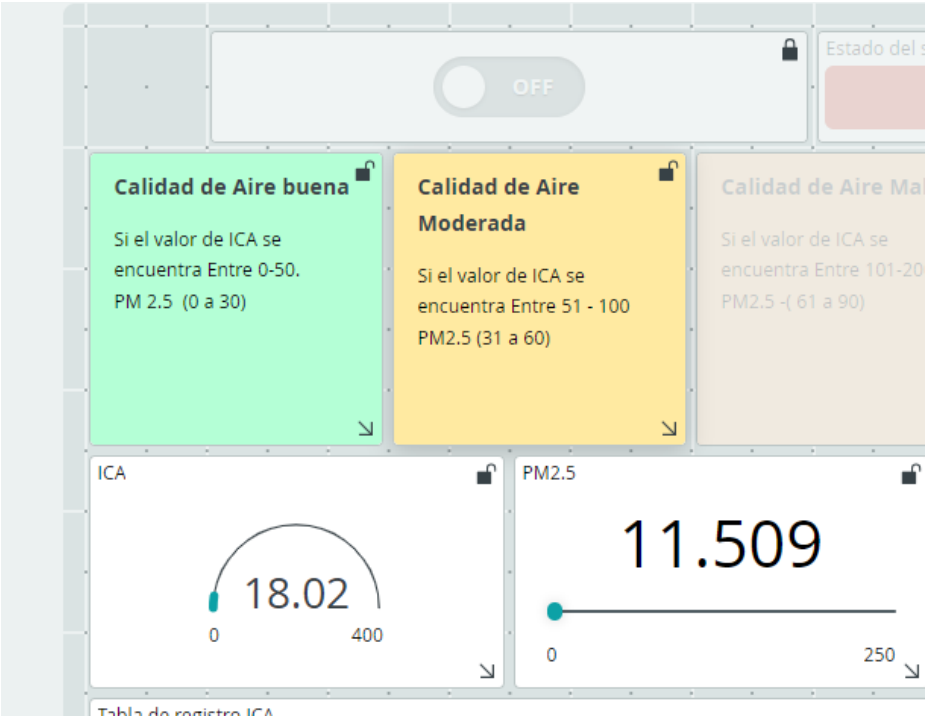

Figura 3.10 Ubicación de los Widgets en el Dashboard

En resumen, en la Figura 3.11 se muestra el resultado final de la creación completa del *Dashboard*, mediante la cual se controla y monitorea las lecturas del sensor, así como también presenta las alertas correspondientes.

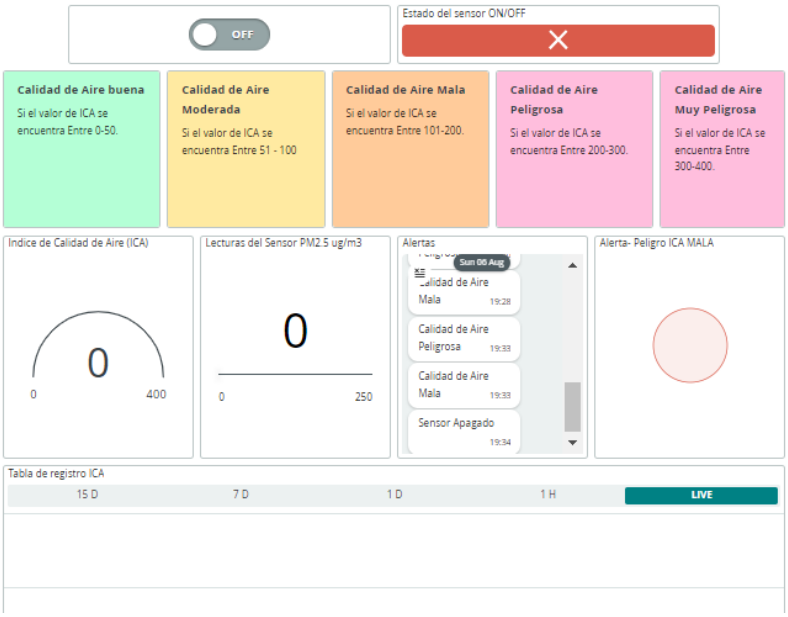

Figura 3.11 Resumen del Dashboard

Elaboración de la placa

Se utilizó Proteus para crear el circuito, que incluye sensores y módulos fácilmente disponibles para simulación y corrección de errores previos antes de trabaja en la placa de circuito. Cada módulo ofrece espacio suficiente para su mantenimiento, y el diseño evita que las pistas se toquen como se muestra en la Figura 3.12.

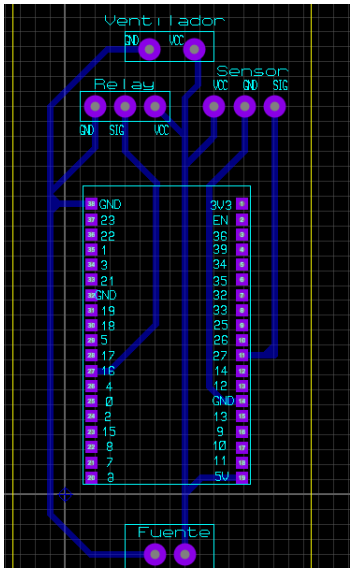

Figura 3.12 Prototipo ESP32 del Ares realizado en Proteus

Justificación de intervalos operativos establecidos para el prototipo

Al tratarse de un valor que cambia continuamente en función de partículas suspendidas en el aire, el módulo GP2Y1051 tiene la capacidad de reconocerlas mediante su módulo infrarrojo y fotodiodo detector, al detectar obstrucción de la luz por las partículas en el aire, envía una serie de señales eléctricas al microcontrolador para que estas sean procesadas.

En específico el sensor envía siete paquetes, cada uno de los cuales contiene un valor de control y un valor variable o flotante. La norma de trabajo estándar funciona entre 30 y 1500 (ug/m3) [12]. El rango de funcionamiento, que se determinó utilizando el Índice de Calidad del Aire (ICA) [18], se muestra en la Tabla 3.3.

Tabla 3.3 Tabla de comparación ICA [18]

Valor Indice	PM 2.5 (ug/m^3)	Calidad del Aire
0 - 50	0 - 30	Buena
51 - 100	31 - 60	Moderada
101 - 200	61 - 90	Mala
201 -300	91 - 120	Peligrosa
301 - 400	121 -150	Muy Peligrosa

Justificación de consumo de energía

Para realizar el cálculo sobre el consumo de energía se tuvieron en cuenta cada uno de los componentes . En la Tabla 3.4 se muestra el rango de voltaje y corriente de cada componente principal, que oscila entre 3,3 y 5 (V).

Tabla 3.4 Voltaje y corriente de trabajo

Componentes Electrónicos	Voltaje	Corriente
ESP32	3,3 - 5 (V)	80 (mA)
Ventilador	5 (V)	250 (mA)
Sensor	5 (V)	20 (mA)
Relé	5 (V)	70 (mA)

Para obtener la corriente de operación del prototipo se utiliza la Ecuación 3.1.

$$I_{Sistema} = I_{ESP32} + I_{Ventilador} + I_{Sensor} + I_{Relé}$$

Ecuación 3.1 Corriente total del prototipo

Donde:

I_{ESP32} : 80 (mA) corriente módulo ESP32

$I_{Ventilador}$: 250 (mA) corriente ventilador

I_{Sensor} : 20 (mA) corriente sensor

$I_{Relé}$: 70 (mA) corriente módulo relé

Por lo tanto:

$$I_{Sistema} = 80\ (mA) + 250\ (mA) + 20\ (mA) + 70\ (mA)$$

$$I_{Sistemas} = 420\ (mA)$$

Al obtener el consumo de corriente del prototipo, se debe seleccionar adecuadamente el tipo de fuente que alimentará todo el sistema. Para este caso en particular la fuente seleccionada establece un voltaje de 5 (V) debido a que todos los componentes funcionan a 5 (V) DC. Además, debe abastecer una corriente alrededor de 420 (mA).

Diagrama de flujo

El diagrama de proceso de la

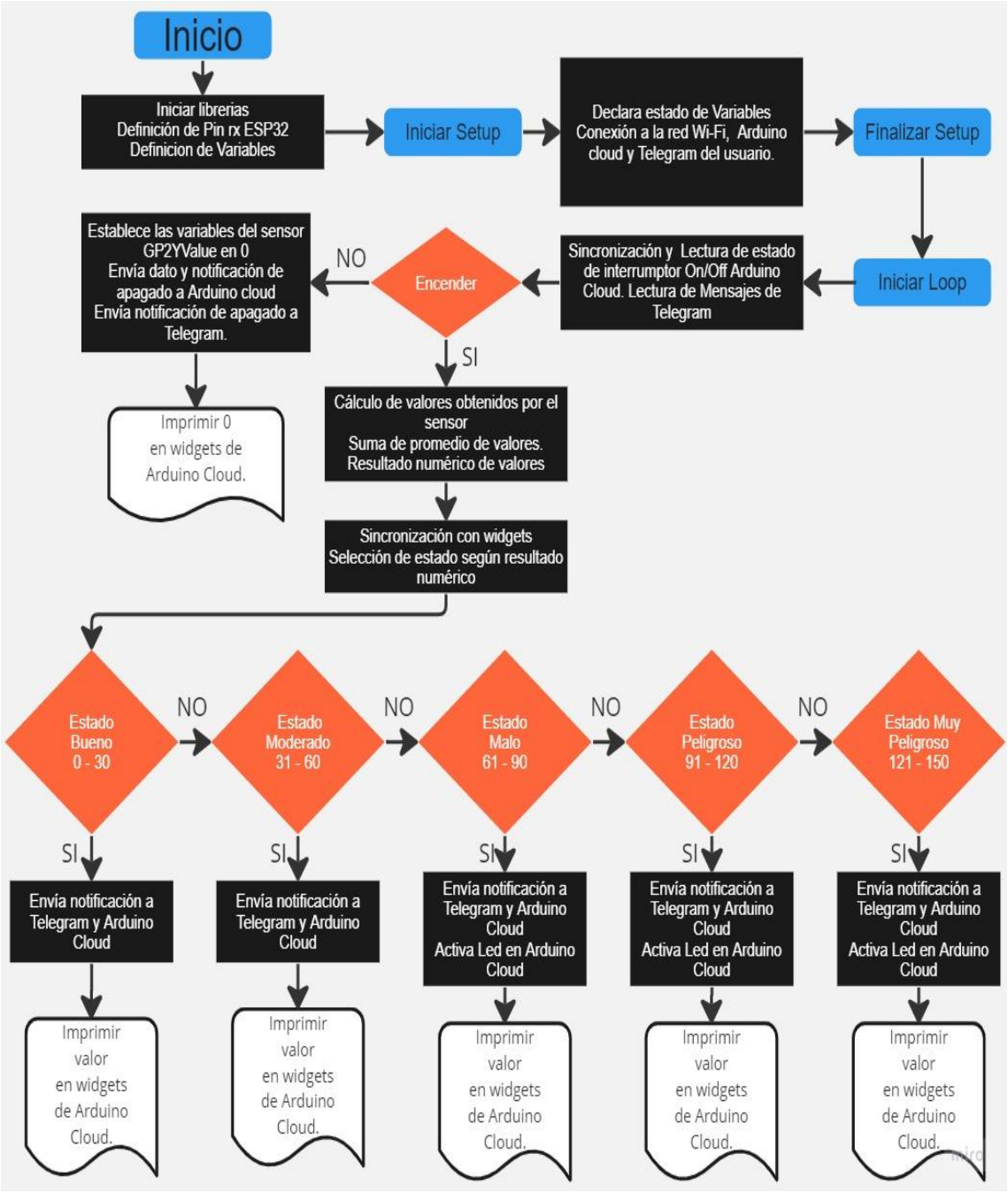

Figura 3.13 sirvió de base para el código de programación. La conversión del valor flotante suministrado por el sensor se establece una vez especificadas Primero las variables, asignando a cada sensor y módulo se establece la conexión mediante los pines digitales del ESP32.

Luego, realiza las conexiones a internet mediante la red *Wi-Fi*. Si todo es correcto inicia con el proceso de obtención y cálculo de valores. Los mismo que serán enviados al

Dashboard. Por último, inicia con las decisiones y envió de notificaciones a *Telegram* si los valores se encuentran en cada uno de los rangos correspondientes.

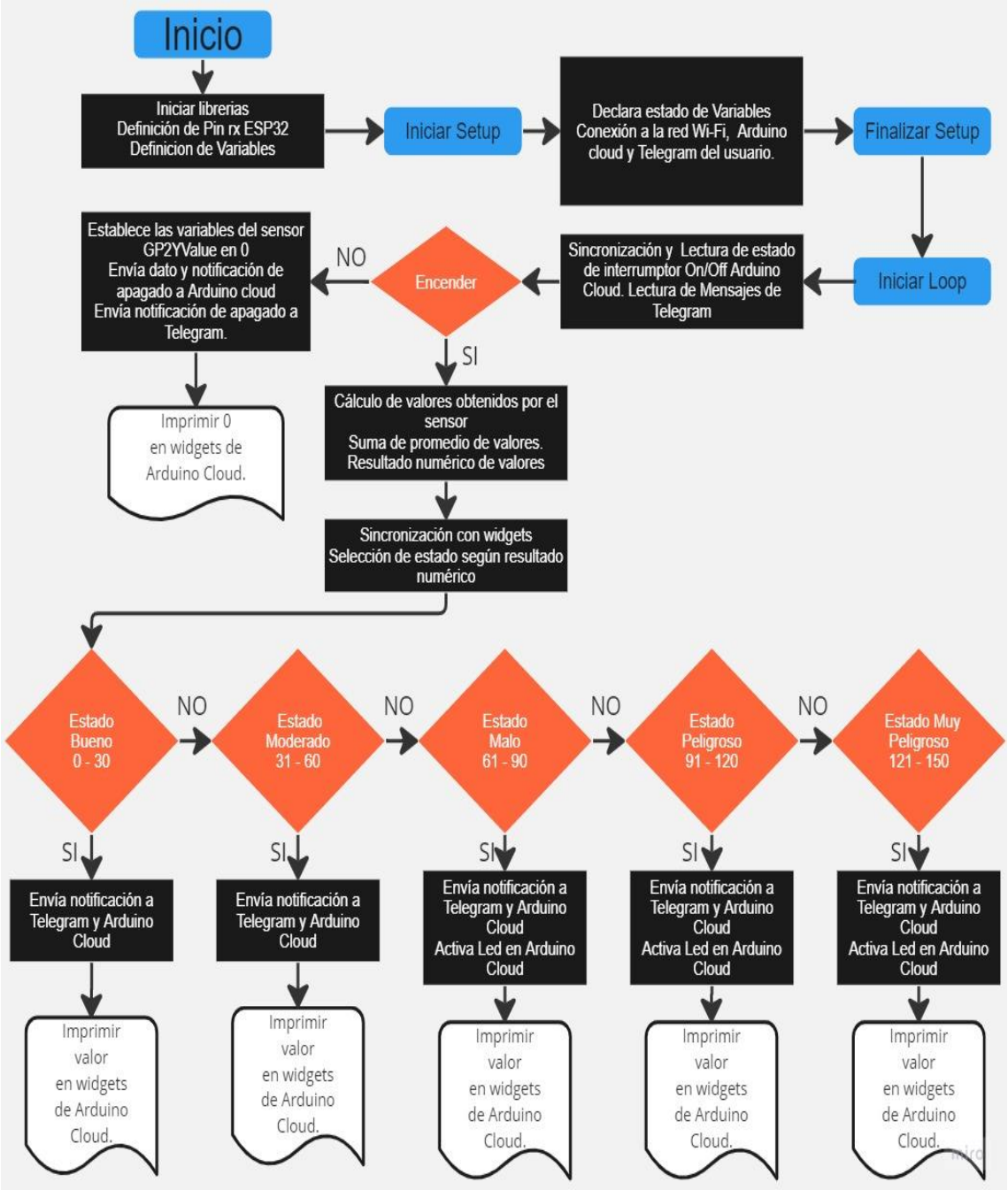

Figura 3.13 Diagrama de flujo

3.4 Implementación del Prototipo

Fabricación de la placa PCB

Para la fabricación de la placa se debe hacer uso del software Proteus, el cual destaca por presentar un entorno de diseño y posterior permite su exportación en formato PDF tal como se muestra en la Figura 3.14; elaborando con medidas exactas del ESP32 y ubicando las conexiones para los componentes y módulos electrónicos.

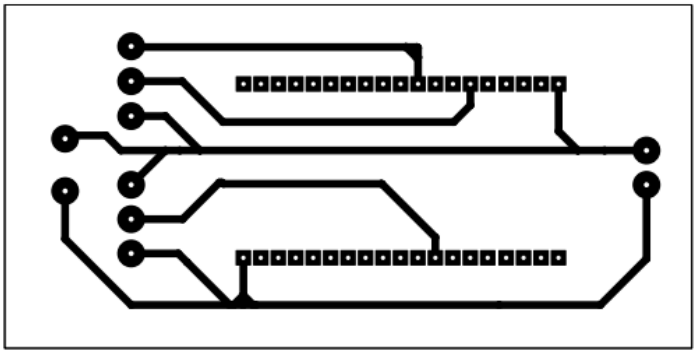

Figura 3.14 Diagrama de la PCB para el método del planchado

Los materiales que se adquieren para la fabricación de la placa PCB son: ácido férrico, una baquelita, estaño. El resultado será la placa diseñada previamente mediante la implementación del método del planchado para PCB [19]. El resultado final se evidencia en la Figura 3.15

Figura 3.15 Elaboración de la PCB final

Montaje de Componentes Electrónicos

En la Figura 3.16 se aprecia el prototipo ya montado con todos los componentes electrónicos soldados a la placa PCB en una caja plástica, en donde se divide en dos partes.

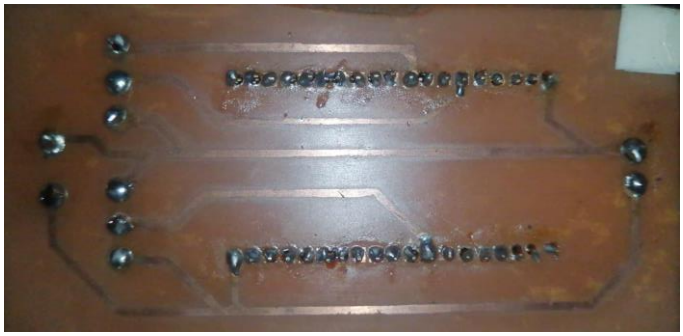

Figura 3.16 Soldadura de la placa PCB con los componentes electrónicos montados

En la parte superior se ubicó el ventilador y el sensor, esta área únicamente realizará la toma de muestras de partículas suspendidas en el aire gracias al ventilador implementado en el costado derecho el cual extraerá el aire del exterior empujándolo hacia el sensor ubicado al lado derecho para que este realice su trabajo, mediante un orificio será expulsado todo el aire ya evaluado por el sensor.

En la parte inferior se encuentra el módulo con toda su circuitería correspondiente y fijada a la caja plástica, esta zona al encontrarse separada del conducto por donde se realizará el censado, se mantendrá protegida de partículas de polvo y otros contaminantes evitando así algún problema de estática y otros problemas derivados.

Figura 3.17 Montaje y ubicación de los componentes y módulos electrónicos en la caja plástica

Después de fijar todo se cierra con su respectiva tapa atornillada. En la Figura 3.18 se observa terminado su montaje.

Figura 3.18 Prototipo final terminado

Programación del Módulo ESP32

Las librerías del módulo *Wi-Fi* utilizadas para la conexión con la red inalámbrica se muestran en la Figura 3.19. Para que el módulo *Wi-Fi* se conecte a la red y pueda cargar los datos recibidos en su plataforma y visualizarlos. Así también las respectivas librerías para conectarse con la aplicación mensajería *Telegram*.

Además, se debe configurar el *Token* del *bot* e *ID* del usuario de *Telegram* para que este logre emitir las alertas al dispositivo móvil.

```
// Librerias necesarias para la conexion Wi-Fi, comunicacion del ESP32 y Telegram
#include <CTBot.h>
#include <CTBotDataStructures.h>
#include <CTBotDefines.h>
#include <CTBotInlineKeyboard.h>
#include <CTBotReplyKeyboard.h>
#include <CTBotSecureConnection.h>
#include <CTBotStatusPin.h>
#include <CTBotWifiSetup.h>
#include <Utilities.h>
#include "thingProperties.h"
#include <SoftwareSerial.h>

// Token del Bot e ID de chat de usuario de  Telegram
const String BOT_TOKEN = "5718887550:AAFyiRl3Dx-Qz-npstp2C-t5cd6ObUeX_iw";
const String CHAT_ID = "1397293757";
```

Figura 3.19 Librerías implementadas, Token e *ID* de usuario

Toda la información de la Red se encuentra en la librería denominada como "*thingProperties.h*" tal como se observa en la Figura 3.20.

```
// Code generado para conectarse a la red Wi-Fi

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

const char DEVICE_LOGIN_NAME[]  = "bf947e20-8451-4d1f-9b74-fa619f709eef";

const char SSID[]               = SECRET_SSID;    // Network SSID (name)
const char PASS[]               = SECRET_OPTIONAL_PASS;    // Network password
const char DEVICE_KEY[]  = SECRET_DEVICE_KEY;    // Secret device password
```

Figura 3.20 Librería “thingProperties.h” con variables de la red Wi-Fi del hogar

Las variables son configuradas previamente en la pestaña denominada como *secret* tal como se visualiza en la Figura 3.21. La misma que ofrece *Arduino Cloud* para facilitar el ingreso de datos de la red *Wi-Fi* del usuario.

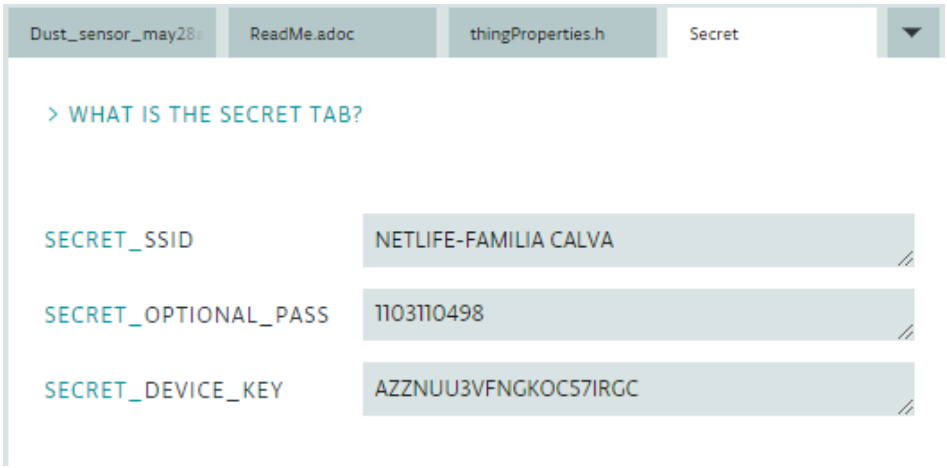

Figura 3.21 Datos de la red Wi-Fi

En la Figura 3.22 se logra observar las definiciones de las variables para el sensor, los respectivos pines para activar el relé que servirá para activar el ventilador. De la misma manera se establece las variables que verifiquen si la notificación ha sido enviada a *Telegram* y se establece la cantidad de muestras recogidas y otros parámetros para realizar un promedio y este permita de cierta forma emitir valores que no cambien drásticamente.

```
//Definicion de pines del ESP32 para leer datos
#define rxPin 16
#define txPin 17
#define relay_gpio 27
// Variables de notificaciones de Telegram
bool lecturaActiva = false;
bool notifBuenaEnviada = false;
bool notifModeradaEnviada = false;
bool notifMalaEnviada = false;
bool notifPeligrosaEnviada = false;
bool notifextrePeligrosaEnviada = false;
 // Número de muestras para calcular el promedio móvil
const int NUM_MUESTRAS = 20;
 // Almacenar las últimas muestras
static float muestras[NUM_MUESTRAS];
// Índice de la muestra actual
static int indiceMuestra = 0;
```

Figura 3.22 Variables del código.

En la Figura 3.23 se observan las funciones que se han creado para guardar los valores del sensor, así como también el valor correspondiente al ICA y el estado del led de alarma representado en el *Dashboard* web de *Arduino Cloud*.

```
void setGp2yValue(float value) {
  // Esta función se encarga de establecer el valor de gp2yValue en el Arduino IoT Cloud
  gp2yValue = value;
  ArduinoCloud.update();
}
void setIca(int value) {
  // Esta función se encarga de establecer el valor de ICA en el Arduino IoT Cloud
  ica = value;
  ArduinoCloud.update();
}
void setLed1(float value) {
  //Esta función se encarga de establecer el estado del led en el Arduino IoT Cloud
  led1 = value;
  ArduinoCloud.update();
}
```

Figura 3.23 Funciones de valores y envío al Dashboard**.**

El sensor de partículas suspendidas en el aire contiene una serie de siete bytes con los valores 0xAA denominado como InicioData y 0xFF denominado como FinData. La Figura 3.24 muestra valores que componen la trama tal y como se indica en la hoja de datos (*datasheet*). Los dos primeros bytes, Valto y Vbajo, se utilizan para calcular el voltaje proveniente del fotodetector y estos se asigne un valor a cada parte de la trama.

```
//configuración del sensor según el datasheet
  if (getSerial() != 0xff) {
    return;
  }
  for (int i = 0; i < 7; i++) {
    cuadro[i] = getSerial();
  }
  InicioData = cuadro[0];
  Valto = cuadro[1];
  Vbajo = cuadro[2];
  Valtor = cuadro[3];
  Vbajor = cuadro[4];
  verificacion = cuadro[5];
  FinData = cuadro[6];

  if (InicioData != 0xaa || FinData != 0xff) {
    return;
  }
  int testSum = Valto + Vbajo + Valtor + Vbajor;

  if (testSum != verificacion) {
    return;
```

Figura 3.24 Comunicación del sensor con módulo ESP32

Las circunstancias de funcionamiento se muestran en la Figura 3.25 y Figura 3.26. En función del número de partículas detectadas, emitirá las notificaciones de alarma en *Telegram* y *Dashboard* web, según la cantidad de partículas detectadas dentro del intervalo establecido para identificar la calidad de aire.

```
//Configuracion de parametros para las alertas y relacion de valores obtenidos con el ICA
    if ((gp2yValue >= 31) && (gp2yValue <= 60)) {
      ica = ((gp2yValue - 31) * (100 - 51) / (60 - 31)) + 50;;
      setIca(ica);
      messenger_Arduino = "Calidad de Aire Moderada";
      setLed1(false);
      if (!notifModeradaEnviada) {
        bot_didactico.sendMessage(chatId, "La calidad de aire es moderada");
// Marcar la notificación como enviada
        notifModeradaEnviada = true;
      }
    } else {
      notifModeradaEnviada = false;
    }
    if ((gp2yValue >= 61) && (gp2yValue <= 90)) {
      ica = ((gp2yValue - 61) * (200 - 101) / (90 - 61)) + 101;
      setIca(ica);
      setLed1(true); // Encender el LED
      messenger_Arduino = "Calidad de Aire Mala";

      if (!notifMalaEnviada) {
        bot_didactico.sendMessage(chatId, "La calidad de aire es mala");
// Marcar la notificación como enviada
        notifMalaEnviada = true;
```

Figura 3.25 Configuración de Alertas y rangos de estado del aire

```
    }
  } else {
    notifMalaEnviada = false;
  }

  if ((gp2yValue >= 91) && (gp2yValue <= 120)) {
    ica = ((gp2yValue - 91) * (300 - 201) / (120 - 91)) + 201;
    setIca(ica);
    setLed1(true);
    messenger_Arduino = "Calidad de Aire Peligrosa";
    if (!notifPeligrosaEnviada) {
      int chatId = CHAT_ID.toInt();
      bot_didactico.sendMessage(chatId, "La calidad de aire es peligrosa");
// Marcar la notificación como enviada
      notifPeligrosaEnviada = true;

    }
  } else {
    notifPeligrosaEnviada = false;

  }
  if ((gp2yValue >= 121) && (gp2yValue <= 150)) {
    ica = ((gp2yValue - 121) * (400 - 301) / (150 - 121)) + 301;
    setIca(ica);
    setLed1(true);
    messenger_Arduino = "Calidad de Aire es Muy peligrosa";
    if (!notifextrePeligrosaEnviada) {
      int chatId = CHAT_ID.toInt();
      bot_didactico.sendMessage(chatId, "La calidad de aire es  muy peligrosa");
// Marcar la notificación como enviada
      notifextrePeligrosaEnviada = true;
```

Figura 3.26 Configuración de Alertas y rangos de estado del aire

3.5 Pruebas de Funcionamiento

Se llevaron a cabo las pruebas siguientes para validar el funcionamiento adecuado del prototipo.

Encender lecturas del sensor desde *Arduino Cloud*

Para activar las lecturas del sensor primero se debe manipular el interrumpir de estado ON/OFF que se encuentra en el Dashboard web. Esta es una de las formas que se puede activar las lecturas del sensor tal como se muestra un antes y después de haber manipulado el interruptor. En la Figura 3.27 y Figura 3.28 respectivamente se logra evidenciar el cambio de estado.

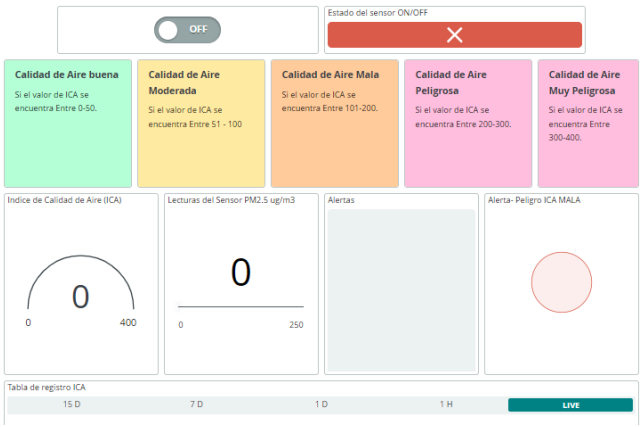

Figura 3.27 Estado del Sensor OFF

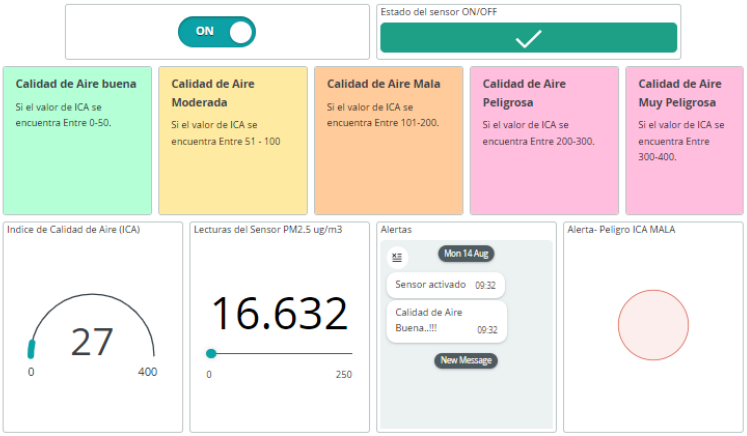

Figura 3.28 Estado del Sensor ON

Encender lecturas del sensor desde *Telegram*

En la Figura 3.29, se logra observar el proceso de inicio del *bot* de *Telegram* al enviar la palabra "*/start*". Con esta acción, el bot responderá con indicaciones para controlar el sensor.

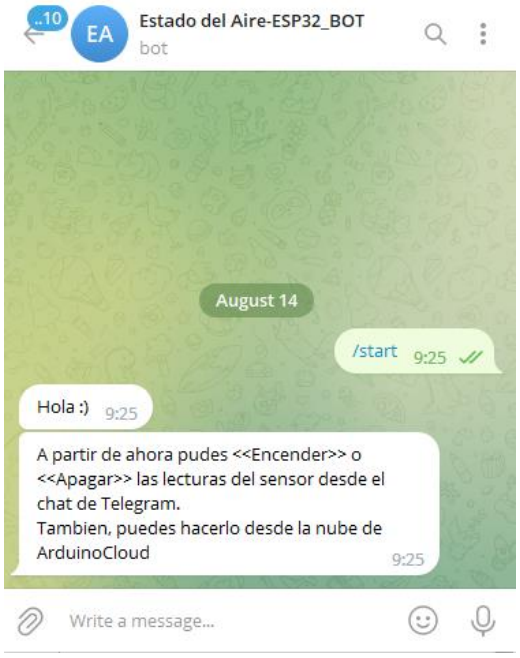

Figura 3.29 Inicio del bot en *Telegram*

Como el *bot* indica, se debe enviar la palabra Encender o Apagar según lo que se necesite. En la Figura 3.30 se logra visualizar la interacción con el *bot* de *Telegram* al enviar la palabra "Encender", de tal manera que el bot ofrezca información de acuerdo con el estado del sensor y mensajes predefinidos en la programación indicando la calidad del aire.

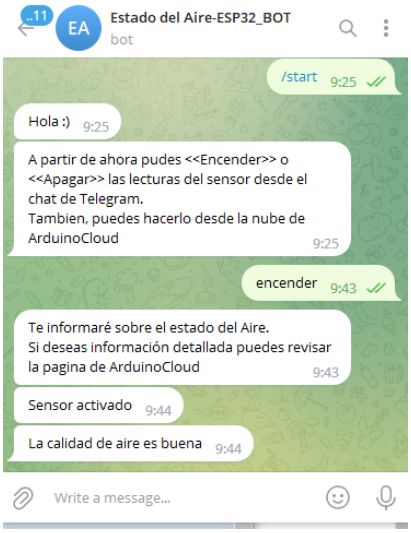

Figura 3.30 Envió de la palabra “encender”

De forma simultánea, también se actualizan los valores y las alertas en el *dashboard web*, tal como se logra visualizar en la Figura 3.31 comparando el horario de registro de las notificaciones.

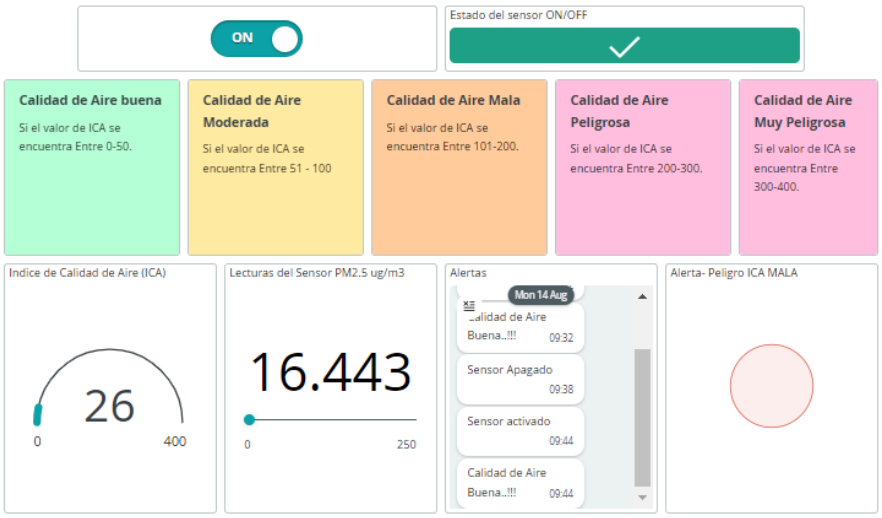

Figura 3.31 Actualización del Dashboard web

Interpretación de los valores en el *Dashboard* web.

El *Dashboard* web fue diseñado de tal manera que cualquier usuario logre interpretar las alertas de manera sencilla, lo que les permita tomar medidas previas para evitar daños en la salud. Con esta idea, en la Figura 3.32 se muestran los datos que el sensor recoge sobre el estado del aire. En la parte superior del tablero, se encuentran algunas notas, las mismas que contienen información sobre los rangos en los que se considera la calidad de aire: buena, moderada, mala, peligrosa y muy peligrosa según el “Índice de Calidad de Aire (ICA)”

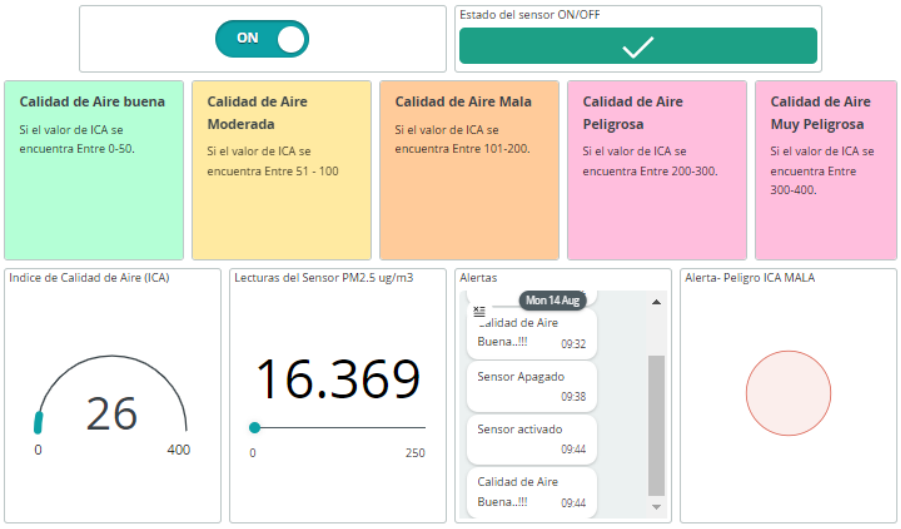

Figura 3.32 Interpretación de valores en el Dashboard web

Los valores corresponden a un entorno en estado normal sin agregar algún agente contaminante del aire. Indicando el primer rango “La calidad de Aire es Buena.!!!” Por lo tanto, no hay necesidad que active las alertas.

También los valores obtenidos se almacenan y son presentados en “La tabla de registro ICA” tal como se evidencia en la Figura 3.33.

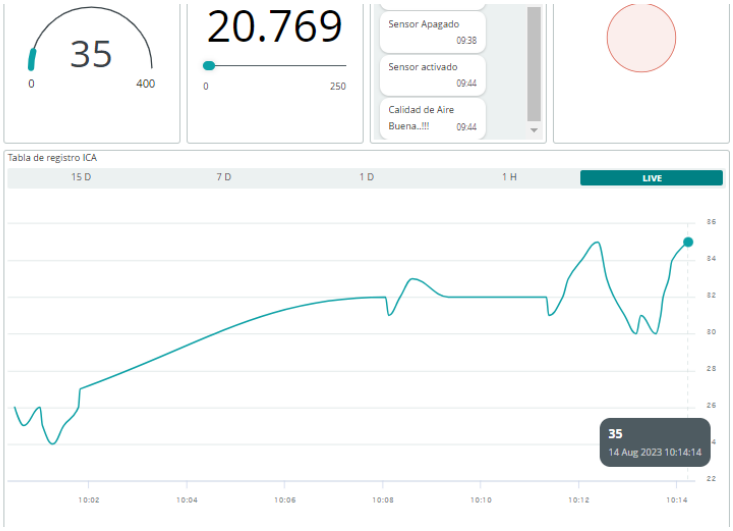

Figura 3.33 Tabla de registro del ICA

Visualización de Alertas en *Arduino Cloud* y *Telegram*

Cuando el sensor detecta algún tipo de partícula que en su interior disperse la luz de los fotodetectores, este procederá a elevar los valores en “Indice de Calidad de Aire (ICA)” así como también los valores “Lecturas del Sensor PM 2.5 (ug/m^3)” por lo cual las alertas empezaran a activarse según los rangos establecidos.

En el *dashboard* web se visualiza los valores del sensor tan pronto los obtiene, emitiendo las alertas correspondientes tal como se observa en la Figura 3.34. Las condiciones muestran un ICA alto, respectivamente la alerta emitida se considera como “Calidad de Aire Muy Peligrosa”.

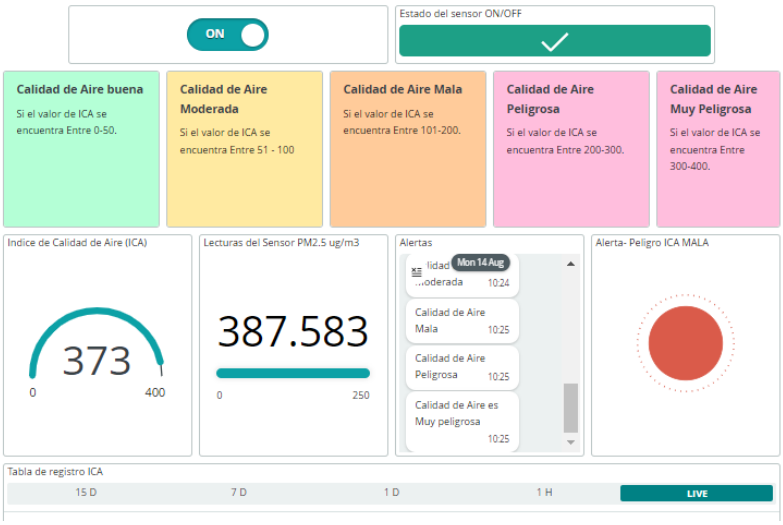

Figura 3.34 Visualización de Alertas en el Dashboard web

Tan pronto la información es procesada, las alertas correspondientes también iniciaran a llegar en la aplicación de mensajería de *Telegram* como se observa en la Figura 3.35. En donde se observa que nuevamente coincide el horario en el cual se emitió las alertas tanto en el *Dashboard* como en la aplicación de *Telegram*

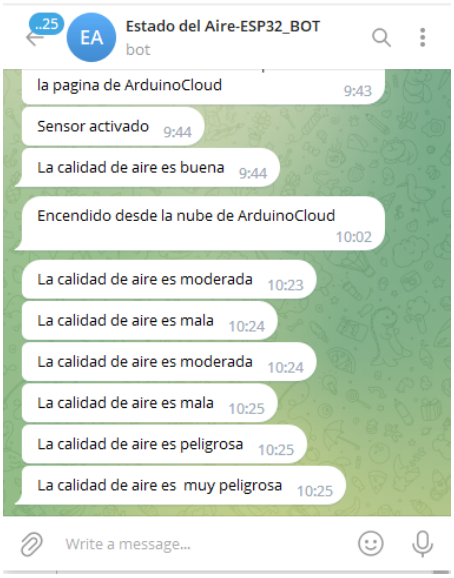

Figura 3.35 Visualización de alertas en *Telegram*

Cabe mencionar que las pruebas de funcionamiento se llevaron a cabo en el contexto de una fase experimental, dado que no se dispone de valores exactos de comparación con los cuales sea posible cuantificar con precisión la concentración de partículas PM 2.5.

Costos de Fabricación e Implementación del prototipo

En la Tabla 3.5 se observa los valores correspondientes a los costos de los componentes electrónicos utilizados para la fabricación e implementación del prototipo.

Tabla 3.5 Costo del prototipo

Material	Cantidad	Precio Unitario	Precio Total
EPS32	1 (u)	$ 11,00	$ 11,00
Sensor GP2Y1051	1 (u)	$ 17,50	$ 17,50
Modulo Relé	1 (u)	$ 2,50	$ 2,50
Ventilador	1 (u)	$ 3,00	$ 3,00
Caja plástica	1 (u)	$ 3,50	$ 3,50
Cinta Doble Faz	1 (u)	$ 2,00	$ 2,00
Fuente de alimentación	1 (u)	$ 17,08	$ 17,08
Baquelita 10 x 5 (cm)	1 (u)	$ 0,75	$ 0,75
Mano de Obra	15 (h)	$ 20,00	$ 300.00
Total			$ 357.33

Video de funcionamiento del prototipo

En la Figura 3.36 se presenta el código QR que redirecciona al video de funcionamiento del prototipo.

Figura 3.36 Código QR del video de funcionamiento del prototipo

4 CONCLUSIONES

- El ICA, que estima la cantidad de partículas aceptable para los seres humanos, es el principal estudio en el que se basa el rango operativo del prototipo, ya que indica cuántas partículas de impurezas por metro cúbico suponen una amenaza para la salud. Por lo tanto, es importante tener en cuenta que la información sobre las partículas PM 2.5 suspendidas en el aire, emplea pronósticos que se basan en datos actuales y modelos de predicción. Sin embargo, factores como la dirección del viento, la altura y la cantidad de partículas diminutas pueden cambiar rápidamente, lo que afecta la precisión de los pronósticos y las alertas. Estas variaciones dificultan la predicción precisa de los efectos en una región específica, este inconveniente se solventa con el prototipo ya que permite determinar del aire en el área en donde se lo coloque.
- Este prototipo se realiza con el fin de dar un seguimiento de la cantidad de partículas en suspensión en cada zona residencial urbana ya que es uno de los objetivos clave del desarrollo de este indicador de calidad de aire. Una mejor calidad del aire es la consecuencia de una vigilancia continua impidiendo que los componentes de estas partículas repercutan en la salud de los seres humanos.
- El correcto funcionamiento del sensor requiere de un flujo continuo de aire que circule por el área de censado. En este sentido, se aconseja la instalación de un ventilador con la fuerza de empuje de aire adecuada, dispuesto a favorecer las mediciones. Esto es esencial, ya que, de no seguir esta indicación, el ventilador podría interrumpir el flujo de aire necesario y por consecuente las lecturas serían erróneas. Esta es una de las consideraciones clave tomadas en cuenta para la creación del prototipo.
- La selección del sensor fue crucial para el despliegue del proyecto, debido a que existe más de un solo tipo de sensor para la detección de partículas PM 2.5. Sin embargo, en algunos casos, estos no se ofrecen en el mercado local o cuestan más de lo esperado y en casos más específicos no es compatible con el microcontrolador empleado. Sin considerar, las temperaturas de operación, debido a que el prototipo estará sometido a la intemperie. Tomando en cuenta todas estos aspectos se seleccionó el sensor GP2Y1051 como el más adecuado a emplearse en la implementación del presente proyecto.
- El ESP32 es la solución más viable para la elaboración del prototipo, ya que cuenta en especial con un módulo *Wi-Fi* integrado facilitando la implementación de proyectos dirigidos al *Iot*, esto evita tener que incluir conexiones y fuentes de

alimentación para agregar módulos externos adicionales para proporcionar la conectividad inalámbrica.

- Durante el desarrollo de la programación es importante establecer una comunicación adecuada entre el sensor y el ESP32. Es necesario conocer la asignación de pines Tx y Rx implementados para establecer una comunicación de serie con los sensores. La mayor cantidad de sensores trabaja con pines de serie por lo cual se debe revisar el *datasheet* del sensor y del ESP32 detalladamente.
- El uso de la plataforma *Arduino Cloud* proporciona una gran ventaja para los usuarios en el ámbito de la programación, ya que facilita el proceso de conexión a Internet y la creación de la aplicación para su control, considerando que a pesar de que presenta limitaciones por ser una plataforma de paga, con la versión gratuita permitió desarrollar de forma adecuada este proyecto de titulación.
- Para fabricar el prototipo final, se debe considerar tanto los materiales utilizados como las medidas de protección necesarias para prevenir posibles daños, en este caso al ser un prototipo diseñado al uso de exteriores, se debe tomar gran consideración las bajas y altas temperaturas de funcionamiento, la presencia de polvo o la exposición al agua, los cuales podrían afectar negativamente a los componentes electrónicos.

5 RECOMENDACIONES

- Se recomienda llevar a cabo la configuración del sensor una vez que el prototipo esté instalado, puesto que hay elementos ambientales como la temperatura y el flujo del aire que pueden influir en las mediciones. Esto implica trabajar en la programación del código para adaptarlo a estas variaciones y lograr una precisión óptima.
- Dado que hay distintas variantes disponibles del sensor de partículas, es aconsejable consultar las fichas técnicas para comprender su requerimiento de voltaje y corriente de operación. Esta precaución ayudará a evitar posibles contratiempos en las lecturas del sensor, ya que de no ser alimentarlo de forma correcta con sus voltajes y corrientes que la ficha técnica menciona puede darse lecturas erróneas.
- Se sugiere la inclusión de un interruptor físico de encendido y apagado, acompañado por luces LED de indicación de estado, en el prototipo. Esta mejora

permite que los usuarios tengan un control directo sobre el funcionamiento del dispositivo, brindando información acerca del estado de encendido y apagado.

- Para futuros trabajos del presente proyecto, se recomienda crear una interfaz de acceso en donde el usuario pueda ingresar datos como nombre y contraseña de la red *Wi-Fi* y datos de la aplicación de mensajería *Telegram*, de manera más eficiente. Esto evitará la necesidad de modificar el código fuente directamente y tener que volver a programarlo en cada instancia.

6 BIBLIOGRAFÍA

[1] Manasvi Kumar, «Prana Air,» Manasvi Kumar, 21 noviembre 2021. [En línea]. Available: https://www.pranaair.com/es/blog/measuring-pm2-5-and-techniques/. [Último acceso: 15 agosto 2023].

[2] Jacob Beningo, «DigiKey,» DigiKey, 21 enero 2020. [En línea]. Available: https://www.digikey.com/es/articles/how-to-select-and-use-the-right-esp32-wi-fi-bluetooth-module. [Último acceso: 14 agosto 2023].

[3] SDI, «SDI,» SDI, 22 mayo 2022. [En línea]. Available: https://sdindustrial.com.mx/blog/sensores/. [Último acceso: 14 agosto 2023].

[4] José Lópz, «ThinkBig,» Think big, 24 Julio 2019. [En línea]. Available: https://blogthinkbig.com/crear-bot-de-telegram-botfather#:~:text=Un%20bot%20que%20controla%20todos,de%20crear%20tu%20propio%20bot.. [Último acceso: 14 agosto 2023].

[5] GCFGLOBAL, «GCFGLOBAL,» GCFGLOBAL, 18 febrero 2021. [En línea]. Available: https://edu.gcfglobal.org/es/curso-de-telegram/que-son-los-bots-de-telegram/1/. [Último acceso: 14 agosto 2023].

[6] D. Ortiz, «Cyberclik,» Cyberclik, 27 septiembre 2022. [En línea]. Available: https://www.cyberclick.es/numerical-blog/que-es-un-dashboard. [Último acceso: 14 agosto 2023].

[7] Arduino, «aprendiendoarduino,» Arduino Cloud, 15 junio 2019. [En línea]. Available: https://aprendiendoarduino.wordpress.com/category/arduino-

cloud/#:~:text=Arduino%20IoT%20Cloud%20permite%20conectar,ser%20interesante%20en%20el%20futuro.. [Último acceso: 14 agosto 2023].

[8] Components101, «Components101,» Components101, 21 diciembre 2020. [En línea]. Available: https://components101.com/switches/5v-single-channel-relay-module-pinout-features-applications-working-datasheet. [Último acceso: 14 agosto 2023].

[9] XATXACA, «XATAXA BASICS,» XATACA, SEPTIEMBRE 23 2022. [En línea]. Available: https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno. [Último acceso: 14 AGOSTO 2023].

[10] IBM, «IBM,» IBM, 30 Julio 2020. [En línea]. Available: https://www.ibm.com/es-es/topics/open-source. [Último acceso: 14 agosto 2023].

[11] Amazon, «Amazon,» Cientific, 29 agosto 2019. [En línea]. Available: https://www.amazon.com/-/es/GP2Y1051AU0F-GP2Y1051-GP2Y10-compacto-acondicionado/dp/B089F4PZ4V. [Último acceso: 14 agosto 2023].

[12] Telktonica, «PocketMagic,» Thechnology, 22 agosto 2022. [En línea]. Available: https://www.pocketmagic.net/sharp-gp2y1051-dust-sensor/. [Último acceso: 14 agosto 2023].

[13] OEHHA, «OEHHA,» OEHHA, 3 abril 2020. [En línea]. Available: https://oehha.ca.gov/calenviroscreen/indicator/pm25#:~:text=La%20materia%20particulada%20o%20PM,grosor%20de%20un%20cabello%20humano.. [Último acceso: 15 agosto 2023].

[14] Dielect, «ssdielect.com,» dielect, 12 abril 2020. [En línea]. Available: https://ssdielect.com/sensores-opticos/659-senor-dsm501a.html. [Último acceso: 14 agosto 2023].

[15] Espressif, «Espressif,» Espressif, 24 julio 2016. [En línea]. Available: https://www.espressif.com/en/products/socs/esp32. [Último acceso: 14 agosto 2023].

[16] Programarfacil.com, «Programarfacil.com,» Programarfacil.com, 22 septiembre 2021. [En línea]. Available: Programarfacil.com. [Último acceso: 14 agosto 2023].

[17] naylampmechatronics, «naylampmechatronics,» naylampmechatronics, 2 abril 2020. [En línea]. Available: https://naylampmechatronics.com/blog/56_usando-esp8266-con-el-ide-de-arduino.html. [Último acceso: 23 agosto 2023].

[18] Smart Air, «Smart Air,» Smart Air, 29 abril 2019. [En línea]. Available: https://smartairfilters.com/en/blog/difference-pm2-5-aqi-measurements/. [Último acceso: 14 agosto 2023].

[19] Ivan Espinoza, «Electronica,» Ivan Espinoza, 13 septiembre 2017. [En línea]. Available: http://www.electronicaivanespinoza.com/2017/09/como-hacer-circuitos-impresos-con-el.html. [Último acceso: 15 agosto 2023].

7 ANEXOS

La lista de los **Anexos** se muestra a continuación:

ANEXO I. Certificado de originalidad

ANEXO II. Código fuente

ANEXO I: Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 28 de Agosto de 2023

De mi consideración:

Yo, **LEANDRO ANTONIO PAZMIÑO ORTIZ**, en calidad de Director del Trabajo de Integración Curricular titulado **IMPLEMENTACIÓN DE UN PROTOTIPO DE ALERTA DE DETERIORO DE LA CALIDAD DE AIRE POR MEDIO DE TELEGRAM Y DESPLIEGUE DE INFORMACIÓN EN UN DASHBOARD WEB**, con un único componente con el mismo tema, elaborado por el estudiante **HENRY ANDRES CALVA ABAD** de la carrera en **TECNÓLOGÍA 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 11%.

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 resumen del reporte generado por Turnitin

Resumen completo generado por Turnitin

Atentamente

Leandro Pazmiño Ortiz

Docente

Escuela de Formación de Tecnólogos

ANEXO II: Código Fuente

```
// Librerias necesarias para la conexion Wi-Fi, comunicacion del ESP32 y Telegram
#include <CTBot.h>
#include <CTBotDataStructures.h>
#include <CTBotDefines.h>
#include <CTBotInlineKeyboard.h>
#include <CTBotReplyKeyboard.h>
#include <CTBotSecureConnection.h>
#include <CTBotStatusPin.h>
#include <CTBotWifiSetup.h>
#include <Utilities.h>
#include "thingProperties.h"
#include <SoftwareSerial.h>

// Token del Bot e ID de chat de usuario de  Telegram
const String BOT_TOKEN = "5718887550:AAFyiRl3Dx-Qz-npstp2C-t5cd6ObUeX_iw";
const String CHAT_ID = "1397293757";

//Definicion de pines del ESP32 para leer datos
#define rxPin 16
#define txPin 17
#define relay_gpio 27
// Variables de notificaciones de Telegram
bool lecturaActiva = false;
bool notifBuenaEnviada = false;
bool notifModeradaEnviada = false;
```

```
bool notifMalaEnviada = false;
bool notifPeligrosaEnviada = false;
bool notifextrePeligrosaEnviada = false;
 // Número de muestras para calcular el promedio móvil
const int NUM_MUESTRAS = 20;
 // Almacenar las últimas muestras
static float muestras[NUM_MUESTRAS];
// Índice de la muestra actual
static int indiceMuestra = 0;



CTBot bot_didactico;
TBMessage mensaje;
int chatId = CHAT_ID.toInt();
SoftwareSerial mySerial(rxPin, txPin);
int getSerial() {
  while (!mySerial.available()) {}
  return mySerial.read();
}


void setGp2yValue(float value) {
  // Esta función se encarga de establecer el valor de gp2yValue en el Arduino IoT Cloud
  gp2yValue = value;
  ArduinoCloud.update();
}
void setIca(int value) {
  // Esta función se encarga de establecer el valor de ICA en el Arduino IoT Cloud
```

```
  ica = value;
  ArduinoCloud.update();
}
void setLed1(float value) {
  //Esta función se encarga de establecer el estado del led en el Arduino IoT Cloud
  led1 = value;
  ArduinoCloud.update();
}

void setup() {
  // Establecimiento de velocidad de lectura en baudios
  Serial.begin(9600);
  mySerial.begin(2400);
  delay(100);
  pinMode(relay_gpio, OUTPUT);
  // Definir el uso de la libreria
  initProperties();
  // Conección a la nube de arduino
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
//Configuracion para conectarse a Telegram mediante al red Wi-Fi
  bot_didactico.wifiConnect(SSID, PASS);
  bot_didactico.setTelegramToken(BOT_TOKEN);
  if (bot_didactico.testConnection()) {
    Serial.println("Conectado a Telegram");
  } else {
```

```
    Serial.println("Error de conexion");
  }

}
//Configuracion de Parametros de control en Telegram
void processTelegramMessages() {
  //Mnesjae de presentacion
  while (bot_didactico.getNewMessage(mensaje)) {
    if (mensaje.text.equalsIgnoreCase("/start")) {
     bot_didactico.sendMessage(chatId, "Hola :)");
     bot_didactico.sendMessage(chatId, "A partir de ahora pudes <<Encender>> o <<Apagar>> las lecturas del sensor desde el chat de Telegram.\nTambien, puedes hacerlo desde la nube de ArduinoCloud");
    }

    // Verificar si el mensaje contiene "encender"
    if (mensaje.text.equalsIgnoreCase("encender")) {
      lecturaActiva = true;
      botton = true ;
       messenger_Arduino = "Sensor activado";
      //int chatId = CHAT_ID.toInt();
      bot_didactico.sendMessage(chatId, "Te informaré sobre el estado del Aire.\nSi deseas información detallada puedes revisar la pagina de ArduinoCloud");
      bot_didactico.sendMessage(chatId, "Sensor activado");

    }
    // Verificar si el mensaje contiene "apagar"
    else if  (mensaje.text.equalsIgnoreCase("apagar")) {
```

```
      lecturaActiva = false;
      botton = false;
      messenger_Arduino = "Sensor Apagado";
      notifBuenaEnviada = false;
      notifModeradaEnviada = false;
      notifMalaEnviada = false;
      notifPeligrosaEnviada = false;
      notifextrePeligrosaEnviada = false;
      bot_didactico.sendMessage(chatId, "Sensor desactivado");
    }
  }
}

void loop() {
// Procesar los mensajes entrantes de Telegram
  processTelegramMessages();
  if (lecturaActiva) {
//Encender Relay del ventilador.
    digitalWrite(relay_gpio,LOW);
// Leer los datos del sensor
    static int cuadro[7];
    static int InicioData, Valto, Vbajo, Valtor, Vbajor, verificacion, FinData;
//configuración del sensor según el datasheet
    if (getSerial() != 0xff) {
      return;
    }
```

```
    for (int i = 0; i < 7; i++) {
      cuadro[i] = getSerial();
    }

    InicioData = cuadro[0];
    Valto = cuadro[1];
    Vbajo = cuadro[2];
    Valtor = cuadro[3];
    Vbajor = cuadro[4];
    verificacion = cuadro[5];
    FinData = cuadro[6];

    if (InicioData != 0xaa || FinData != 0xff) {
      return;
    }

    int testSum = Valto + Vbajo + Valtor + Vbajor;

    if (testSum != verificacion) {
      return;
    }
  // Almacenar la muestras actuales y calibracion del sensor
    float Vout = (Valto * 256 + Vbajo) / 1024.0 * 5.0;
      muestras[indiceMuestra] = ((Vout / 3.5) * 1000);
      indiceMuestra = (indiceMuestra + 1) % NUM_MUESTRAS;

  // Calcular el promedio de datos obtenidos
```

```
      for (int i = 0; i < NUM_MUESTRAS; i++) {
        gp2yValue += muestras[i];
      }
      gp2yValue /= NUM_MUESTRAS;

//Imprime valores en la consola
    Serial.print("Densidad del aire: ");

    Serial.print(gp2yValue);
    Serial.println(" ug/m3");
    Serial.print("\n");
    Serial.print("ICA: ");
    Serial.print(ica);
// Envia a la nube
    setGp2yValue(gp2yValue);

    if ((gp2yValue >= 0) && (gp2yValue <= 30)) {
      ica = ((gp2yValue - 0) * (50 - 0) / (30 - 0)) + 0;
      setIca(ica);
      messenger_Arduino = "Calidad de Aire Buena..!!!";
      setLed1(false);

      if (!notifBuenaEnviada) {
        bot_didactico.sendMessage(chatId, "La calidad de aire es buena");
// Marcar la notificación como enviada
        notifBuenaEnviada = true;
```

```
      Serial.print("ICA: ");
      Serial.print("\n");
      Serial.print(ica);
    }
  } else {
    notifBuenaEnviada = false;

  }
//Configuración de parámetros para las alertas y relacion de valores obtenidos con el ICA
  if ((gp2yValue >= 31) && (gp2yValue <= 60)) {
    ica = ((gp2yValue - 31) * (100 - 51) / (60 - 31)) + 50;;
    setIca(ica);
    messenger_Arduino = "Calidad de Aire Moderada";
    setLed1(false);
    if (!notifModeradaEnviada) {
      bot_didactico.sendMessage(chatId, "La calidad de aire es moderada");
// Marcar la notificación como enviada
      notifModeradaEnviada = true;
    }
  } else {
    notifModeradaEnviada = false;
  }
  if ((gp2yValue >= 61) && (gp2yValue <= 90)) {
    ica = ((gp2yValue - 61) * (200 - 101) / (90 - 61)) + 101;
    setIca(ica);
    setLed1(true); // Encender el LED
```

```
      messenger_Arduino = "Calidad de Aire Mala";


      if (!notifMalaEnviada) {
        bot_didactico.sendMessage(chatId, "La calidad de aire es mala");
// Marcar la notificación como enviada
        notifMalaEnviada = true;
      }
    } else {
      notifMalaEnviada = false;
    }


    if ((gp2yValue >= 91) && (gp2yValue <= 120)) {
      ica = ((gp2yValue - 91) * (300 - 201) / (120 - 91)) + 201;
      setIca(ica);
      setLed1(true);
      messenger_Arduino = "Calidad de Aire Peligrosa";
      if (!notifPeligrosaEnviada) {
        int chatId = CHAT_ID.toInt();
        bot_didactico.sendMessage(chatId, "La calidad de aire es peligrosa");
// Marcar la notificación como enviada
        notifPeligrosaEnviada = true;


      }
    } else {
      notifPeligrosaEnviada = false;


    }
```

```
    if ((gp2yValue >= 121) && (gp2yValue <= 150)) {
      ica = ((gp2yValue - 121) * (400 - 301) / (150 - 121)) + 301;
      setIca(ica);
      setLed1(true);
      messenger_Arduino = "Calidad de Aire es Muy peligrosa";
      if (!notifextrePeligrosaEnviada) {
        int chatId = CHAT_ID.toInt();
        bot_didactico.sendMessage(chatId, "La calidad de aire es  muy peligrosa");
// Marcar la notificación como enviada
        notifextrePeligrosaEnviada = true;

    } else {
      notifextrePeligrosaEnviada = false;

    }
    if ((gp2yValue >= 151) && (gp2yValue <= 240)) {
        setLed1(true);
      }

    }
  delay(2500);
  } else {
    //Establecer valores en 0 cuando se apaga las lecturas del  sensor
    float gp2yValue = 0;
    float ica = 0;
    //Presentar los valores apra corroborar en la interfaz de monitor
    digitalWrite(relay_gpio, HIGH);
```

```
    Serial.print("Densidad del aire: ");
    Serial.print(gp2yValue);
    Serial.println(" ug/m3");
    Serial.print("\n");

    setGp2yValue(gp2yValue);
    Serial.print("Ica:");
    Serial.print(ica);
    Serial.print("\n");
    Serial.print("Sensor apagado");
    setLed1(false);
    setIca(ica);
  }
  ArduinoCloud.update();
// Pausa de 100 milisegundos
  delay(100);
}
//Configuracion del Boton ON/OFF de Arduino Cloud
void onBottonChange()  {
  if (botton == true) {
    lecturaActiva = true;
    Serial.println("Detector Encendido");
    messenger_Arduino = "Sensor activado";
    bot_didactico.sendMessage(chatId, "Encendido desde la nube de ArduinoCloud");
  } else {
    lecturaActiva = false;
    Serial.println("Detector Apagado");
```

```
      botton = false;
      notifBuenaEnviada = false;
      notifModeradaEnviada = false;
      notifMalaEnviada = false;
      notifPeligrosaEnviada = false;
      notifextrePeligrosaEnviada = false;
    messenger_Arduino = "Sensor Apagado";
    bot_didactico.sendMessage(chatId, "Apagado desde la nube de ArduinoCloud");
  }
  // Actualizar el estado en Arduino IoT Cloud
  ArduinoCloud.update();
}
```