ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE SENSORES DE BAJO COSTO PARA EL MONITOREO DE GASES CONTAMINANTES Y MATERIAL PARTICULADO EN EL AIRE

IMPLEMENTACIÓN DE UN MÓDULO SENSOR DE BAJO COSTO PARA LA DETERMINACIÓN DE LOS NIVELES DE CONCENTRACIÓN DE DIÓXIDO DE NITRÓGENO Y OZONO EN EL AIRE

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

SISA CANDO KEVIN DAVID

kevin.sisa@epn.edu.ec

DIRECTOR: Ing. FABIO MATÍAS GONZÁLEZ GONZÁLEZ

fabio.gonzalez@epn.edu.ec

DMQ, agosto 2023

CERTIFICACIONES

Yo, Kevin David Sisa Cando 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 DAVID SISA CANDO

Certifico que el presente trabajo de integración curricular fue desarrollado por Kevin David Sisa Cando, bajo mi supervisión.

_____

FABIO MATÍAS GONZÁLEZ GONZÁLEZ

DIRECTOR

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 DAVID SISA CANDO

FABIO MATÍAS GONZÁLEZ GONZÁLEZ

DEDICATORIA

A mis padres Luis y Fanny por el apoyo, comprensión y todo el esfuerzo realizado para que pueda completar esta etapa educativa. Esfuerzo, dedicación y perseverancia fueron los pilares inculcados por mis padres, quienes merecen el honor y crédito de este logro.

A mi abuela Aurora, por sus consejos y amor incondicional, quien desde mi infancia ha estado presente en los mayores logros de mi vida, como a su vez en las peores situaciones. Gracias abuela.

A mis amigos, Daniel, Erika y Joel, por demostrarme que las mejores amistades no nacen por la cantidad de tiempo en conocerlas, gracias por el apoyo brindado durante esta grandiosa etapa, siempre los llevaré presente.

Por las viejas amistades que perduran con los años y su grandioso significado, a mis amigos Jorge, Karina, Mariela y Ángel, quienes han estado presentes en mi vida desde la adolescencia, gracias por hacer que nuestra amistad se fortalezca con los años.

A mis familiares, por los consejos y apoyo, en no rendirme y ser perseverante hasta lograr mis objetivos.

Kevin David Sisa Cando

AGRADECIMIENTO

Agradecimiento y mención al MSc. Fabio González, director del presente trabajo de integración curricular, por el apoyo y las directrices para el desarrollo de este trabajo.

A los docentes de la Escuela Politécnica Nacional quienes formaron parte de mi desarrollo profesional, por el soporte y consejos, para tener una visión del profesional a futuro.

Kevin David Sisa Cando

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ÍNDICE DE CÓDIGOS

RESUMEN

El presente Trabajo de Integración Curricular plantea el diseño y la implementación de un dispositivo que permita la toma de mediciones de concentración de dióxido de nitrógeno (NO_2) y ozono (O_3) haciendo uso de sensores de bajo costo, posterior a ello, las mediciones respectivas son alojadas en una aplicación web para una adecuada visualización de los datos. Los resultados del dispositivo de bajo costo permitirán realizar las respectivas comparaciones con las mediciones de la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito ubicado en el Centro Histórico para determinar la correcta medición de los sensores.

En el primer capítulo se establece el estado y objetivo del Proyecto PVI DICA 2023-01 de la Escuela Politécnica Nacional, se investiga sobre la Norma de Calidad de Aire en Ecuador. Adicionalmente, la tecnología y elementos electrónicos necesarios para las mediciones de NO_2 y O_3.

Respecto al segundo capítulo, se define la metodología Kanban para llevar a cabo las actividades del presente trabajo, se establecen los requerimientos y parámetros para el diseño del sistema, así como las funcionalidades para el módulo sensor y la aplicación web. Para finalmente, llegar a la implementación del módulo sensor y de la aplicación web, en ambos casos realizando la explicación debida del procedimiento de implementación hasta el resultado final.

En el tercer capítulo se detallan y analizan las pruebas de funcionamiento del módulo sensor y de la aplicación web, hasta llegar a un resultado óptimo cumpliendo con los objetivos establecidos para el presente proyecto.

PALABRAS CLAVE: Calidad de aire, dióxido de nitrógeno, ozono, sensores, metodología Kanban, Firebase, Cloud Firestore, Firebase Hosting, ESP32.

ABSTRACT

This curricular integration project presents the design and implementation of a device that allows the measurement of nitrogen dioxide (NO_2) and ozone (O_3) concentration using low-cost sensors. Subsequently, the respective measurements are hosted on a web application for proper data visualization. The results from the low-cost device will enable comparisons with measurements from the Quito Metropolitan Atmospheric Monitoring Network station located in the Historic Center to determine the correct sensor measurements.

In the first chapter, the state and objective of the 2023-01 PVI DICA Project from the Escuela Politécnica Nacional are established. In relation to this, research is conducted about the Air Quality Standard in Ecuador, the NO_2 and O_3 measurement standards for the respective study. Additionally, the technology, sensors, and electronic devices necessary to achieve the set objectives are investigated.

Regarding the second chapter, the Kanban methodology is initially defined to carry out the activities of this work. Then, the requirements and parameters for the System design are established, as well as the respective functionalities for the sensor module and the web application. Finally, the implementation of the sensor module and the web application is reached, in both cases explaining the implemented technology and the implementation procedure ups to the result.

In the last chapter, the calibration of the sensors is detailed, along with the respective tests of the sensor device and the web application, ultimately achieving an optimal result that fulfills the established objectives for this project.

KEYWORDS: Air quality, nitrogen dioxide, ozone, sensors, Kanban methodology, Firebase, Cloud Firestore, Firebase Hosting, ESP32.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La contaminación del aire ha sido uno de los problemas más complicados que atraviesan las grandes ciudades en varios países, esto sucede debido a la falta de control y marcos regulatorios que establezcan límites permisibles para los contaminantes. Tal es el caso de Ecuador, país en el cual se tiene la reforma a la Norma de Calidad del Aire Ambiente o Nivel de Inmisión la cual fue publicada el martes 7 de junio de 2011.

La Norma de Calidad del Aire trata de preservar la salud de personas, de tener un control sobre la calidad de aire y el bienestar de ecosistemas [1]. Así resulta importante mencionar que en sus artículos 4.1.1.3 y 4.1.1.4 se encuentra descrita la responsabilidad del manejo conjunto entre las autoridades ambientales y los municipios de las ciudades respectivas para los programas de monitoreo y hacer cumplir la norma presente. Sin embargo, se conoce que solamente las ciudades de Quito y Cuenca cuentan con dichos medidores de calidad de aire [2].

Por ello surge la interrogante del ¿por qué solamente estas ciudades tienen medidores de calidad del aire? Y es que, por palabras del ingeniero Carlos Páez, profesor de la Escuela Politécnica Nacional, se ha podido comprender que existe una complejidad tecnológica y altos costos para la implementación de un sistema de monitoreo. Como resultado en búsqueda de otros mecanismos para el monitoreo se tiene la iniciativa de implementar dispositivos con sensores de bajo costo que permitan obtener los mismos datos que los medidores de calidad de aire oficiales, la presente iniciativa forma parte del proyecto PVI DICA 2023-01 en el cual se pretende la implementación de dispositivos, los cuales al estar ligados con tecnologías IoT (Internet de las cosas) podrán incorporar sensores y software para la recolección y el intercambio de datos, permitiendo una mejor accesibilidad a los mismos.

El presente trabajo de integración curricular presenta la implementación de un dispositivo que permita obtener datos de niveles de concentración en el aire de dióxido de nitrógeno (NO_2) y ozono (O_3) mediante el uso de los módulos sensores MICS-6814 y MQ-131, respectivamente. Ambos sensores son de bajo costo, de forma complementaria al aplicar tecnologías basadas en IoT los datos han podido ser almacenados y mostrados en una aplicación web desarrollada con React.js para la toma de decisiones en relación con el monitoreo de la calidad del aire.

1.1 Objetivo general

Implementar un dispositivo con sensores de bajo costo para el monitoreo de calidad del aire en la ciudad de Quito.

1.2 Objetivos específicos

1. Medir la concentración de dióxido de nitrógeno (NO_2) y ozono (O_3) en el aire, mediante el uso de módulos sensores MQ-131 y MICS-6814, para el monitoreo de calidad de aire.
2. Probar el funcionamiento del dispositivo final, mediante pruebas de campo en una localidad específica, para determinar la viabilidad de la implementación de sensores de bajo costo.
3. Almacenar las mediciones de concentración obtenidas por los sensores en Cloud Firestore haciendo uso de la placa de desarrollo ESP32 de 38 pines con conexión hacia Internet.
4. Implementar una aplicación web hacia el usuario final mediante la biblioteca JavaScript React para la visualización y exportación de los datos almacenados.

1.3 Alcance

Para cumplir con los objetivos planteados en el presente trabajo se realizarán las siguientes actividades:

Diseño del dispositivo haciendo uso de los sensores MQ-131 y MICS-6814, para medición de dióxido de nitrógeno (NO_2) y ozono (O_3), respectivamente.

Configuración de la placa de desarrollo ESP32 38 Pines ESP WROOM 32 para el envío y almacenamiento de los datos obtenidos por los sensores en Cloud Firestore, el cual es un servicio otorgado por Firebase como base de datos en la nube.

Configuración del servicio de host en Firebase Hosting, de tal manera que con el uso de la biblioteca JavaScript React, se programe y despliegue una aplicación web para mostrar los datos de las mediciones almacenadas por el dispositivo de sensado.

1.4 Marco teórico

1.4.1 Situación actual del Proyecto PVI DICA 2023-01

El proyecto de Monitoreo de calidad del aire con sensores de bajo costo y desarrollo de prototipos de fabricación local que se está desarrollando, se encuentra vinculado con las Carreras de Ingeniería Ambiental (FICA) e Ingeniería en Tecnologías de la Información (FIEE). El objetivo de este es utilizar sensores que sean de bajo costo para medir partículas de elementos contaminantes para el estudio de la calidad del aire en la ciudad de Quito, por ello se desea fabricar un prototipo local, en este caso para medir dióxido de nitrógeno (NO_2) y ozono (O_3), y que los resultados obtenidos de la medición sean validados con los resultados de una estación de la Red Metropolitana de Monitoreo Atmosférico de Quito.

Entre los resultados del proyecto, se espera que se valide la viabilidad de la implementación a mayor escala de los dispositivos sensores para el uso en otras ciudades, de tal manera que se tenga una reducción en costos y mediciones similares que con los dispositivos de alto costo.

1.4.2 Norma Ecuatoriana de Calidad del Aire

La Norma Ecuatoriana de Calidad del Aire está “dictada bajo el amparo de la Ley de Gestión Ambiental y del Reglamento a la Ley de Gestión Ambiental para la Prevención y Control de la Contaminación Ambiental” [1], la cual define los objetivos de calidad de aire, además de establecer los límites máximos permitidos para los contaminantes.

Fue publicada en el año 2011 formando parte del conjunto de normas técnicas ambientales, las cuales sirven como prevención y para realizar un control de la contaminación, teniendo entre sus propósitos cuidar la salud de las personas, por lo cual brinda procesos y métodos para determinar la concentración de los contaminantes. Es importante definir ciertos términos en base a lo mencionado por la norma:

Aire: Se denomina a una porción de la atmósfera, siendo una mezcla gaseosa de composición 20% de oxígeno, 79% nitrógeno y 1% de dióxido de carbono.

Concentración de una sustancia en el aire: Consiste en la relación entre el volumen o peso de cierta sustancia con la unidad de volumen de aire en la que se encuentra contenida.

Contaminante del aire: Se refiere a sustancias o materiales que son emitidos hacia la atmósfera, ya sea por actividad humana o por un proceso natural.

Contaminantes criterio del aire: Se refiere a cualquier contaminante de aire, estableciéndose un máximo de concentración que será permitido.

Contaminante peligroso del aire (no convencional): Se denominan a los contaminantes del aire que llegan a ser amenaza con efectos adversos para el ambiente o la salud humana.

Es importante mencionar que en los requisitos 4.1.1.3 y 4.1.1.4 de los contaminantes del aire ambiente se determina que se deberán implementar los programas de monitoreo a nivel nacional cumpliendo con la Norma de Calidad de Aire, además de que se realizarán verificaciones, con programas de monitoreo, que no se llegue a superar el límite permitido de los contaminantes no convencionales y criterio, y haciendo que se cumpla con la Norma de Calidad de Aire.

La Tabla 1.1 presenta los niveles de concentraciones de contaminantes criterio.

Tabla 1.1 Concentración de contaminantes criterio [1]

Contaminante	Periodo de tiempo	Alerta	Alarma	Emergencia
Monóxido de carbono (CO)	8 horas (ug/m^3)	15000	30000	40000
Ozono (O_3)	8 horas (ug/m^3)	200	400	600
Dióxido de nitrógeno (NO_2)	1 hora (ug/m^3)	1000	2000	3000
Dióxido de azufre (SO_2)	24 horas (ug/m^3)	200	1000	1800
Material particulado PM10	24 horas (ug/m^3)	250	400	500
Material particulado PM2.5	24 horas (ug/m^3)	150	250	350

Para el presente Trabajo de Integración Curricular se va a trabajar con los contaminantes: dióxido de nitrógeno (NO_2) y ozono (O_3).

1.4.3 Módulo sensor MICS-6814

El sensor MICS-6814 es un “sensor compacto con la capacidad de detectar con precisión la concentración simultánea de CO, NO_2 y NH_3, proporcionando canales independientes para cada gas” [3]. Este sensor también permite medir otro tipo de gases: metano (CH_4), propano (C_3H_8), etanol (C_2H_5OH), isobutano (C_4H_{10}) e hidrógeno (H_2).

De tal manera que este tipo de sensor permite detectar la contaminación, ya sea automovilística o en otros ambientes como la agricultura o industria, con el objetivo de monitorear la calidad de aire, por lo cual es de vital importancia para el presente proyecto. Adicional a ello, presenta 3 chips de sensores los cuales tienen calentadores independientes de sus capas sensibles.

Por la referencia [4] se conoce que el voltaje de operación del sensor es de 5 [V] DC, con dimensiones de aproximadamente 17mm x 14mm x 3mm y un peso de 1,5 gramos. Adicional a ello, la Figura 1.1 muestra el rango de detección del sensor MICS-6814 para el caso de partículas de dióxido de nitrógeno (NO_2).

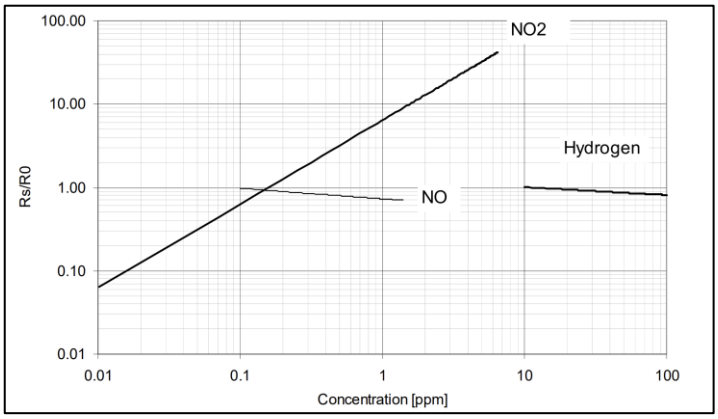


Figura 1.1 Rango de detección de NO_2 [5]

La Figura 1.2 presenta la distribución de pines del sensor, siendo: CO, NH_3, NO_2, GND (tierra) y +5V (alimentación de 5 [V]).

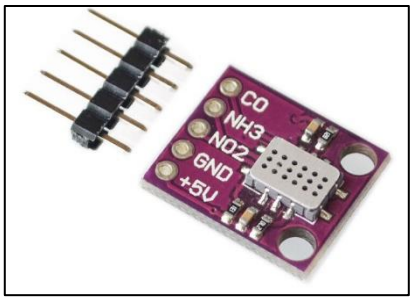


Figura 1.2 Sensor MICS-6814 [4]

1.4.4 Módulo sensor MQ-131

El sensor MQ-131 presenta un material sensible de SnO_2, el que permite conductividad baja en aire limpio, pero esta conductividad es más alta al momento en que aumenta la concentración de gas, en este caso de Ozono, por lo cual se infiere que el sensor es sensible a Ozono, además de Cl_2, NO_2, entre otros [6]. En rangos amplios presenta buena sensibilidad y aún más sensible a presencia de ozono siendo de larga vida a un bajo costo, con aplicaciones domésticas, industriales y portátiles.

Se considera el sensor MQ-131 de baja concentración como un elemento óptimo para el desarrollo del presente trabajo de integración curricular, en la referencia [7] se presentan ciertas especificaciones técnicas de este sensor:

- Voltaje de funcionamiento: 5 [V] en CC
- Resistencia de carga: 1 [KΩ]
- Rango de detección: 10 – 1000 [ppb]
- Dimensiones: 35mm x 40mm x 28mm

Dada la Figura 1.3 se tiene la gráfica de concentración de Ozono (O_3):

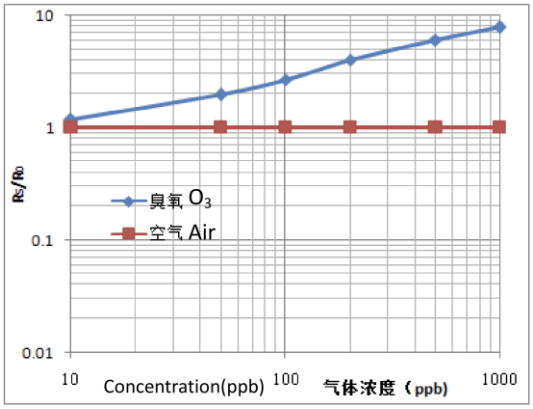


Figura 1.3 Rango de detección de Ozono [6]

Adicional, en la Figura 1.4 se presenta la distribución de pines del módulo sensor MQ-131 de baja concentración: GND (tierra), Analog Out (pin de salida analógica) y Digital Out (pin de salida digital) y VCC (alimentación de 5 [V]):

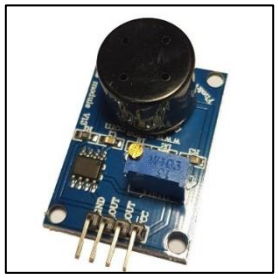


Figura 1.4 Distribución de pines del sensor MQ-131 [7]

1.4.5 Sensor BMP280

El módulo sensor BMP280 consiste en un sensor para mediciones de temperatura con precisión de ±1 °C y presión barométrica con precisión de ±1 hPa. De la referencia [8] se tienen ciertas especificaciones técnicas del sensor:

- Voltaje de alimentación: 3 [V] / 3.3 [V]
- Rango de presión de aire: de 300 [hPa] a 1100 [hPa]

- Precisión de temperatura: -40 °C a +85 °C
- Bajo consumo de energía

Dadas las especificaciones, se ha considerado al sensor BMP280 apto para utilizarlo en el presente trabajo, para la toma de temperatura y presión barométrica; adicionalmente, en la Figura 1.5 se presentan los pines de conexión del sensor: VCC (alimentación de 3.3[V]), GND (tierra), SDA (pin de datos serie en I2C), SCL (pin de reloj serie en I2C) y por último los pines SCB SDO los cuales no serán utilizados para este caso.

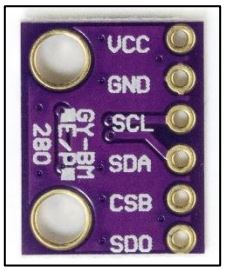


Figura 1.5 Distribución de pines sensor BPM180 [8]

1.4.6 ESP32 38 Pines ESP WROOM 32

El dispositivo ESP32 consiste en una placa de desarrollo la cual posee un microcontrolador ESP32-WROOM-32 proveniente de Espressif, siendo una placa importante para realizar trabajos relacionados con IoT ya que posee WiFi y Bluetooth. A continuación, de la referencia [9] se mencionan ciertas especificaciones técnicas de este tipo de ESP32:

- Modelo: ESP32 38 Pines
- Voltaje de alimentación: 5 [V] en DC
- Procesador: Tensilica Xtensa 32-bit LX6
- Frecuencia de reloj: hasta 240 MHz
- Wi-Fi: 802.11 b / g / n / e / i (802.11n 2.4 GHz)
- Bluetooth: 4.2 BR / EDR y BLE (Bluetooth Low Energy)
- ROM: 448KB
- SRAM: 520KB
- Pines Digitales GPIO: 24
- Conversor Analógico Digital: 2 ADC de 12 bits
- Seguridad: IEEE 802.11 con WFA, WAPI y WPA/WPA2

Considerando las características de este modelo de placa de desarrollo se ha elegido para ser usada en el desarrollo del presente proyecto. La Figura 1.6 presenta la distribución de pines de la ESP32 de 38 pines.

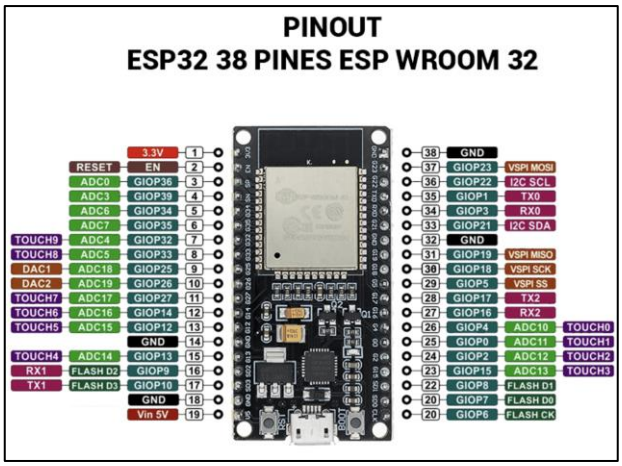


Figura 1.6. ESP32 de 38 Pines ESP WROOM 32 [9]

1.4.7 Cloud Firestore

Cloud de Firebase consiste en una base de datos NoSQL, la cual es flexible y escalable para almacenar datos y sincronizarlos para desarrollar el lado del servidor y del cliente desde Firebase y Google Cloud. Ofrece soporte sin conexión, además de integrarse a otros productos ofrecidos por Firebase como Google Cloud y Cloud Functions [10].

Entre las funciones presentes se tiene que admite estructuras de datos jerárquicas y flexible, permitiendo que los datos se guarden en documentos, además permite realizar consultas basadas en documentos individuales y recuperar una colección de acuerdo con ciertos parámetros. Es importante destacar que usa sincronización para actualizar los datos en cualquier dispositivo, en tiempo real y con capacidad de almacenar en su caché de datos de tal manera que cualquier aplicación podrá leer, escribir, escuchar y consultar los datos, aunque el dispositivo no posea conexión.

1.4.8 Firebase Hosting

Es un servicio para hosting para los desarrolladores, de tal forma que se implementan aplicaciones web con contenido dinámico y estático, ofreciendo la entrega de contenido con conexión segura, ya que incluye SSL sin que se deba configurar, además de ello su alojamiento puede ser dinámico y estático permitiendo que se tenga archivos CSS o HTML para hosting. Es importante mencionar que su función permite que se emule los cambios en una URL, la cual está alojada de manera local, lo que permite ver los cambios antes de ser publicados [11].

Mediante el CLI de Firebase se llega a implementar los archivos de los directorios del ordenador hacia el servidor de hosting, además de tener beneficios relacionados al *backend* permitiendo que sea más fácil realizar las pruebas y mejoramiento del sistema.

1.4.9 Node.js

Consiste en “un entorno de tiempo de ejecución JavaScript multiplataforma y de código abierto utilizado para ejecutar código JavaScript fuera de un navegador web” [12]. Presenta el administrador de paquetes de nodos llamado “npm” con más de 350.000 paquetes en su biblioteca los cuales sirven para proyectos y desplegar aplicaciones de manera eficiente, además de ser un modelo de entrada y salida con un solo subproceso, el cual es controlado por eventos de tal manera que se creen aplicaciones desde el servidor, las cuales lleguen a ser escalables haciendo uso de JavaScript.

La referencia [13] menciona que se debe considerar 3 componentes principales para comprender el funcionamiento de Node.js:

- **Motor V8:** Es el motor de JavaScript el cual llega a interpretar y ejecutar coódigo JavaScript en Chrome, además de entender que otros navegadores usan diferentes motores, y este motor sirve para que el ordenador comprenda JavaScript.
- **Libuv:** Consiste en una biblioteca C, del tipo multiplataforma, con la cual se realizan las operaciones E/S (entrada/salida), de tal manera que se envían solicitudes y se reciben respuestas del equipo, estas operaciones son realizadas más de una a la vez.
- **Event Loop:** El bucle de eventos utiliza el motor JavaScript V8 para ejecutar las devoluciones de las llamadas de las instrucciones asíncronas.

1.4.10 React.js

React.js o también conocido como React, consiste en una biblioteca JavaScript de interfaz de usuario con la cual se crean componentes para interfaces de usuario, de tal manera que se llegue a combinar estos componentes en pantallas o páginas de manera rápida y eficiente [14]. El creador de React.js fue Jordan Walke con el objetivo de simplificar el proceso de desarrollo con una forma estructurada para construir las interfaces, las cuales tengan sus componentes reutilizables.

Con React.js se desarrollan aplicaciones las cuales crean componentes que se pueden reutilizar, siendo piezas individuales que al final se acoplan a una interfaz. Se considera que su función principal es manejar la capa de vista de las aplicaciones, algo similar al

MVC, con eficiencias en renderizado [15]. React.js permite que los desarrolladores separen estas interfaces en bloques siendo más eficiente para manejar el DOM y representar las páginas web de manera más veloz, siendo dinámicas y receptivas.

1.4.11 Arduino IDE

Entorno de desarrollo integrado (IDE), consiste en un programa el que presenta varias herramientas para programación, ya sea para un lenguaje de programación o varios, además el IDE ha sido empaquetado por lo cual llega a consistir en un editor de código, compilador, depurador y GUI, siendo especifico el caso de Arduino, porque también tiene herramientas para que un programa que ya se encuentra compilado se cargue en la memoria flash del dispositivo [16]. El fichero de un programa en Arduino presenta la extensión “ino”, el cual se encuentra en un directorio con el mismo nombre del fichero.

La descarga del IDE de Arduino es gratuita y se lo hace desde la página oficial, es necesario que al instalar y empezar a usar se tenga en cuenta que se debe indicar el tipo de placa que se está utilizando y a la cual se va a cargar el programa, además elegir el puerto de comunicación adecuado al cual se ha conectado la placa. Como ventaja, el IDE permitirá validar el código, para después compilarlo y que éste sea cargado en el microcontrolador [17].

1.4.12 Visual Studio Code

Es un editor de código que brinda herramientas que resultan eficaces para los desarrolladores, siendo compatible con macOS, Linux y Windows, por lo cual es independiente de la plataforma. Su editor de código fuente es rápido y soporta varios lenguajes siendo productivo en resaltar la sintaxis del código, además de la sangría automática, entre otros [18]. La referencia [19] menciona ciertas características de VS Code:

- **Multi-proyecto:** En este aspecto se puede tener abiertos de manera simultánea varios proyectos los cuales pueden contener varias carpetas o archivos.
- **Soporte de Git:** Como ventaja los recursos pueden ser extraídos desde GitHub Repo en línea o viceversa.
- **Estructura jerárquica:** Los archivos de los códigos realizados se ubican en carpetas y archivos.
- **Soporte de múltiples lenguajes:** Soporta varios lenguajes de programación, además detecta fallas o referencias en otros lenguajes.

- **Intelli-Sense:** Esto se refiere a que tiene la capacidad de detectar si se ha dejado incompleto algún fragmento de código.
- Se conoce que VS Code “se compila con el Shell Electron, Node.js, TypeScript y el protocolo Language Server, y se actualiza mensualmente. Las numerosas extensiones se actualizan tantas veces como sea necesario” [20].

2 METODOLOGÍA

En este capítulo se presenta el proceso de diseño e implementación del dispositivo con sensores de bajo costo, además el desarrollo de la aplicación web, juntamente con las consideraciones establecidas para su ejecución y correcto funcionamiento.

En el **Apartado 2.1** se plantea la metodología para el desarrollo de las actividades, los requerimientos del sistema, además de las funcionalidades del dispositivo sensor y de la aplicación web, en el **Apartado 2.2** se establece el proceso de implementación del dispositivo sensor, mientras que en **Apartado 2.3** se detalla el proceso de implementación de la aplicación web.

2.1 Diseño del sistema

Se ha establecido la metodología Kanban, la cual permite gestionar las tareas de manera visual, así se podrán ir monitoreando los estados de las actividades definidas en este trabajo, se llega a establecer un tablero con columnas de estados de las actividades de tal manera que se clasifiquen en: pendientes, en progreso y terminadas [21].

2.1.1 Planteamiento del tablero Kanban

Para establecer el tablero Kanban se hará uso de Microsoft Planner, mediante el cual se plantearán las actividades a realizar tal como presenta la Figura 2.1, las cuales están clasificadas en: pendiente, progreso y terminadas; junto con las etiquetas de: Investigación, Diseño, Implementación de dispositivo y aplicación web, y Pruebas.

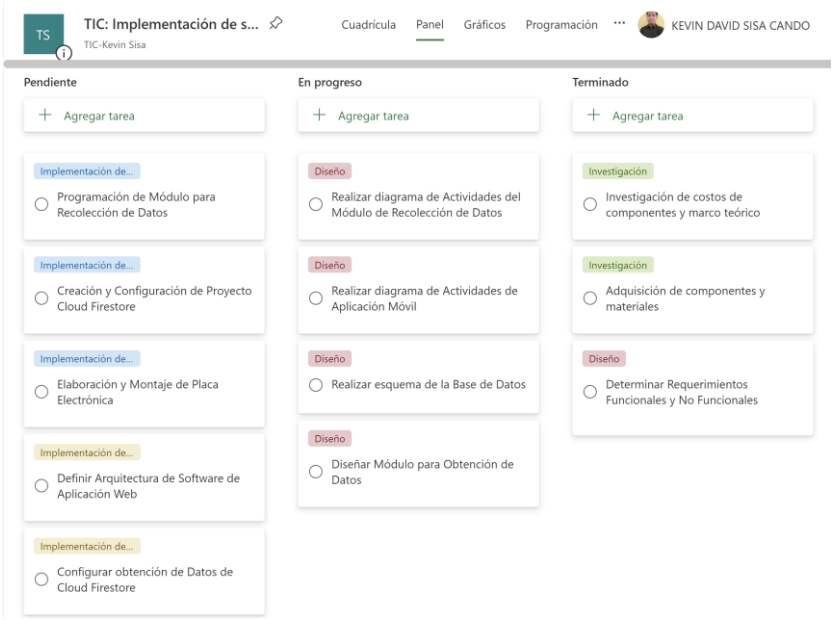


Figura 2.1 Planteamiento de tablero Kanban

2.1.2 Definición de requerimientos

Es importante definir los requerimientos funcionales y no funcionales para desarrollar la aplicación web, dado que se puede tener un sustento de los objetivos y aspectos que debe tener el aplicativo desarrollado, estos requerimientos se obtienen con base en historias de usuario con información otorgada por los responsables del Proyecto PVI DICA 2023-01.

2.1.3 Requerimientos funcionales

Los requerimientos funcionales corresponden a las descripciones sobre cómo debe funcionar el sistema, además de las particularidades de comportamiento en ciertas situaciones. En el presente caso, los requerimientos funcionales presentan las funcionalidades del sistema frente a la visualización de las mediciones de los sensores dadas por el dispositivo de bajo costo.

Los requerimientos funcionales obtenidos se los mostrará mediante las historias de usuario:

Tabla 2.1 Historia de usuario para visualización de medición de sensores

RF-01	**Visualización de las mediciones de los sensores**
Usuario:	Estándar
Prioridad:	Alta
Descripción:	
Se requiere visualizar las mediciones de NO_2 y O_3.	
Criterios de aceptación:	
• El sistema deberá presentar en su pantalla principal las mediciones obtenidas por el módulo sensor.	

Tabla 2.2 Historia de usuario para visualización de mediciones en tablas

RF-02	**Visualización de las mediciones de NO_2 y O_3 en tablas dinámicas**
Usuario:	Estándar
Prioridad:	Alta
Descripción:	
Se requiere visualizar las mediciones de NO_2 y O_3 en tablas dinámicas.	
Criterios de aceptación:	
• El sistema deberá mostrar las mediciones del módulo sensor en una tabla dinámica. • El sistema permitirá realizar la búsqueda de datos en base a la fecha de medición.	

Tabla 2.3 Historia de usuario para exportar datos en Excel

RF-03	**Exportar en Excel las mediciones de NO_2 y O_3**
Usuario:	Estándar
Prioridad:	Alto

Descripción:
Se requiere exportar en Excel las mediciones de NO_2 y O_3.
Criterios de aceptación:
• El sistema permitirá exportar en un archivo de extensión .xlsx las mediciones de NO_2 y O_3. • La exportación del archivo .xlsx será en base a un intervalo de fechas establecido por el usuario.

Tabla 2.4 Historia de usuario para presentación de mediciones en gráficas

RF-04	**Presentación de gráficas de las mediciones de NO_2 y O_3**
Usuario:	Estándar
Prioridad:	Media
Descripción:	
Se requiere mostrar gráficas de las mediciones de NO_2 y O_3 de forma independiente.	
Criterios de aceptación:	
• El sistema deberá mostrar gráficas de áreas para las mediciones de NO_2 y O_3 desde la primera medición realizada hasta la última. • El sistema deberá mostrar la gráfica con hora, fecha y la cantidad medida de NO_2 y O_3.	

2.1.4 Requerimientos no funcionales

Los Requerimientos No Funcionales abarcan otros ámbitos distintos a los funcionales, pero aún siguen ligados al sistema como tal, entre ellos se tiene la disponibilidad, rendimiento o seguridad. De tal forma que son un complemento a los requerimientos funcionales.

Los requerimientos no funcionales se presentan mediante las historias de usuario siguientes:

Tabla 2.5 Historia de usuario para usabilidad de aplicación web

RNF-01	**Usabilidad de la aplicación web**
Usuario:	Estándar
Prioridad:	Alta
Descripción:	
Se requiere que la aplicación web sea amigable con el usuario y fácil de usar por parte del usuario estándar.	
Criterios de aceptación:	
• El sistema presentará una interfaz gráfica de facilidad de uso. • Se deberá brindar la capacitación respectiva sobre el uso de la aplicación con duración de 15 minutos.	

Tabla 2.6 Historia de usuario para registro y validación de usuarios

RNF-02	**Registro y validación de usuarios**
Usuario:	Estándar
Prioridad:	Alta

Descripción:
Se requiere que la aplicación web solicite un registro e inicio de sesión de usuarios para visualizar los datos.
Criterios de aceptación:
• El sistema presentará un formulario de registro del usuario que desee visualizar las mediciones respectivas. • El sistema presentará un formulario para inicio de sesión del usuario validando las credenciales ingresadas en el registro.

Es importante destacar que en la aplicación web a desarrollar no se ha considerado el manejo de roles de usuarios para ingresar al sistema, debido a que los resultados de mediciones de NO_2 y O_3 son de carácter público e informativo para la sociedad, además de que la única funcionalidad será la de visualización y exportación de datos.

2.1.5 Funcionalidades del módulo para obtención de datos

Se describe el funcionamiento del módulo que permite recolectar los datos mediante la realización de diagramas de actividades en el software Draw.io. Las funciones de la recolección de datos en el presente trabajo se realizan mediante el uso de la placa de desarrollo ESP32 38 Pines ESP WROOM 32. En la función Setup() se definirán ciertas funciones que se ejecutarán en el programa, además se establecen ciertos criterios de ejecución, tal como se observa en la Figura 2.2, se inicia la comunicación serial, con ello intentará conectarse a Internet, tras conectarse procederá a iniciar sesión con Firebase para enviar los datos a Firestore Database. A continuación, iniciará la comunicación I2C y la del sensor BMP280, por último, la comunicación con el servidor NTP para obtener fecha y hora.

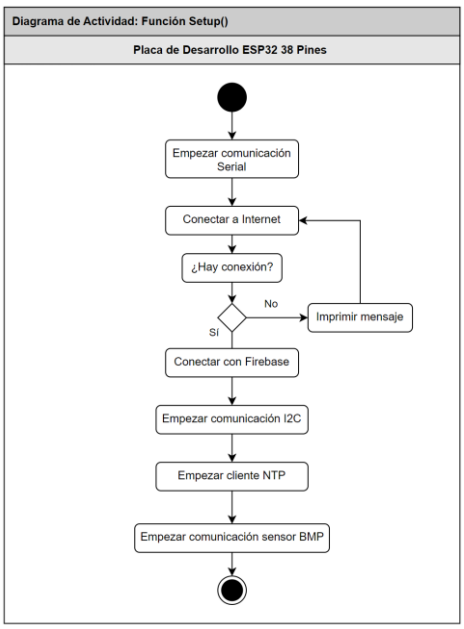


Figura 2.2 Diagrama de actividad de función Setup()

Por otra parte, la función Loop() es la que se ejecutará de manera continua hasta desconectarse o reiniciar el ESP32, siendo la que se ejecuta después del Setup(). En la Figura 2.3 se tiene en primera instancia la obtención de la fecha y hora mediante el servicio NTP, las cuales servirán para registrar la fecha y hora del almacenamiento de la medición realizada. Cada 15 minutos los sensores MQ-131, MICS-6814 y BMP280 harán sus respectivas mediciones, se valida si los datos medidos son correctos, es decir, si el sensor ha realizado alguna medición y luego los valores obtenidos se envían a Firebase.

Finalizado el envío a Firebase se actualiza la variable de tiempo para validar las siguientes mediciones a intervalos de 15 minutos.

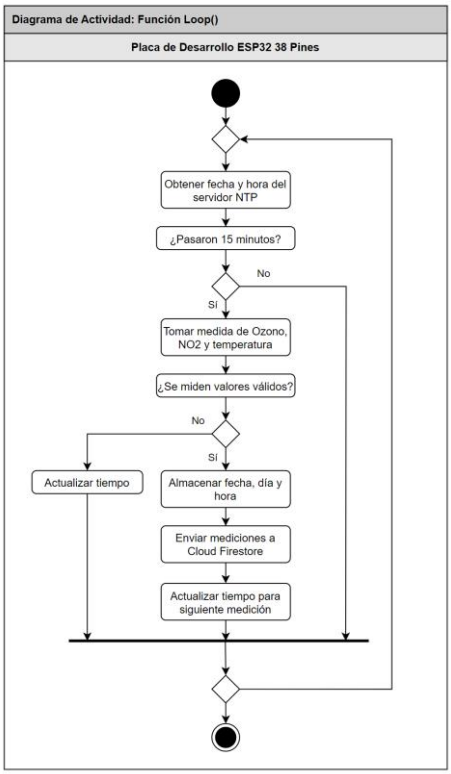


Figura 2.3 Diagrama de actividad de función Loop()

2.1.6 Funcionalidades del sistema

Para presentar y analizar las funcionalidades que presenta el sistema a implementar se desarrollan los diagramas de actividades respectivos mediante el uso de Draw.io, de tal forma que se tenga una mejor apreciación sobre el flujo de desarrollo que tendrá la aplicación web relacionado con las distintas actividades dentro del sistema.

Proceso de autenticación y registro

La Figura 2.4 presenta el proceso de autenticación y registro. Además, para el diagrama de flujo se mostrarán las entidades participantes: Usuario, Aplicación web y Firebase.

Al ingresar a la dirección URL de la aplicación web se presentará la interfaz para Inicio de sesión, en la cual el usuario previamente registrado deberá ingresar sus credenciales de correo electrónico y contraseña, de tal forma el servicio de Firebase Authentication validará las credenciales para el ingreso. En caso de ingresar datos correctos, el usuario será redirigido a la pantalla principal, por otra parte, si las credenciales son incorrectas, se presentarán los respectivos mensajes de error relacionados al correo electrónico o a la contraseña. Adicional a ello, se tiene la opción de recuperar contraseña en la cual se debe ingresar el correo y Firebase se encarga de enviar un enlace al correo para realizar el cambio de contraseña.

Si el usuario no posee credenciales en la aplicación web, se mostrará la opción de Registro, en la cual se deberá registrar un correo electrónico y una contraseña mayor o igual a 8 caracteres. Una vez registrado se redireccionará al usuario a la pantalla principal.

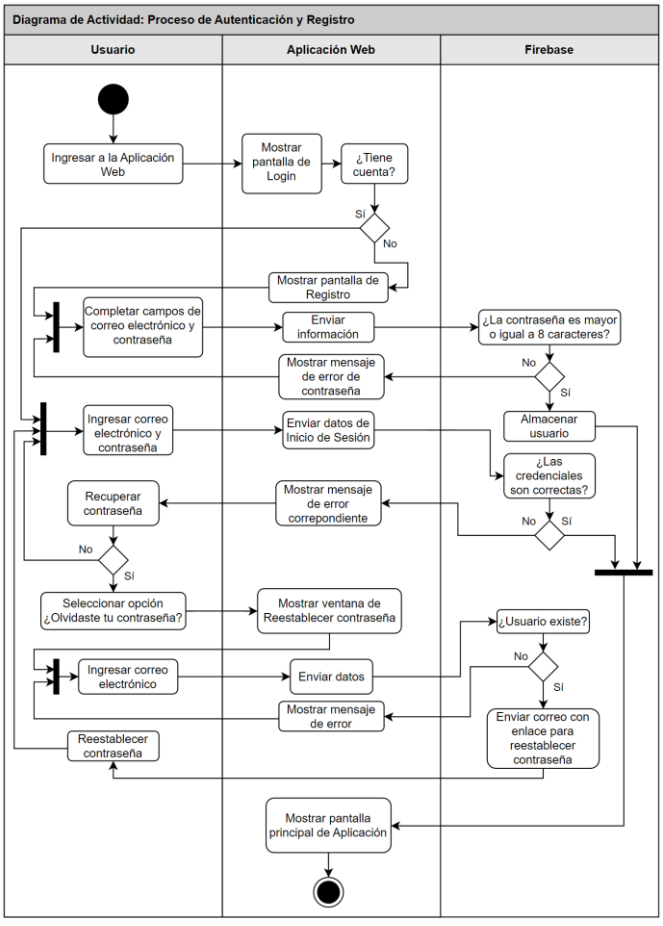


Figura 2.4 Diagrama de actividad de proceso de autenticación y registro

Proceso de visualización y exportación de datos de O_3 y NO_2

Para el proceso de visualización de los datos obtenidos con los sensores se presenta el diagrama de actividades en la Figura 2.5, en la cual el usuario al iniciar sesión podrá

observar la pantalla principal donde se muestra una tabla con las mediciones de ozono (O_3), dióxido de nitrógeno (NO_2) y temperatura, junto con los demás datos obtenidos desde el dispositivo, correspondientes a la fecha, hora y día de la medición.

Se presentan dos botones, el primero de “Visualizar gráfica”, el cual permite que el usuario observe una gráfica de áreas de las mediciones almacenadas de ozono (O_3), dióxido de nitrógeno (NO_2) y temperatura. Mientras que el otro botón de “Exportar datos” presenta una ventana para escoger una fecha de inicio y fecha de fin en un calendario, luego de ello, al seleccionar “Exportar”, se descargará un archivo XLSX con las mediciones realizadas en las fechas escogidas.

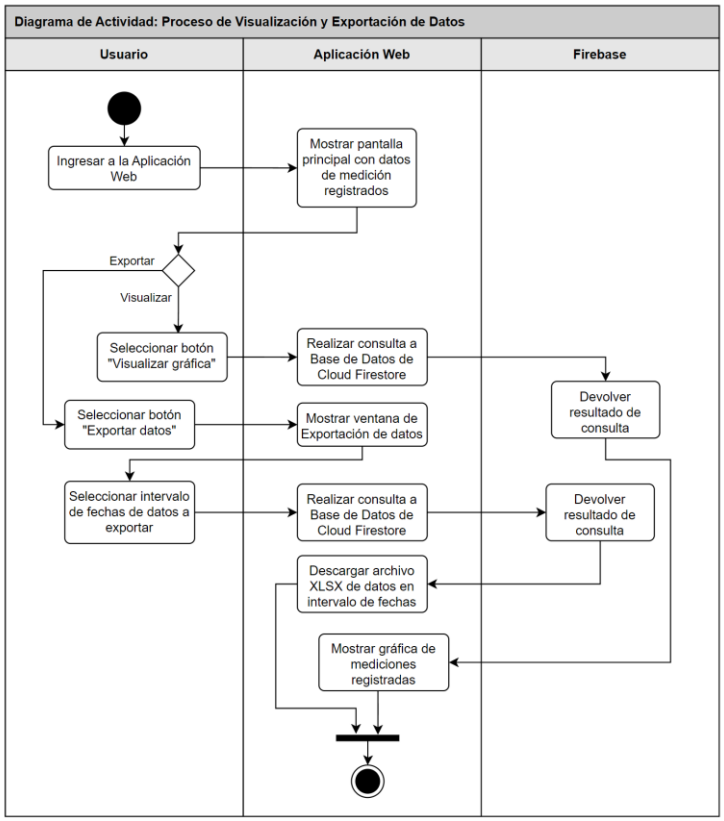


Figura 2.5 Diagrama de proceso de visualización y exportación de datos

2.1.7 Esquema de base de datos

Entre los servicios que Firebase presenta hacia el público en general, se tiene el de Firestore Database, a la cual se conoce como una base de datos tipo NoSQL con la característica que permite guardar información en documentos los cuales forman parte de alguna colección establecida.

Al crear el documento se tendrá una estructura con los datos medidos por los sensores. En el presente caso, se realizarán las tomas de niveles de ozono (O_3) y de dióxido de

nitrógeno (NO_2), las cuales serán obtenidas mediante los sensores MQ-131 y MICS-6814, respectivamente. Estas medidas serán tomadas cada 15 minutos y luego enviadas a Cloud Firestore, adicional a ello, se registrará el día, fecha y hora de la medición.

Todos los datos mencionados se guardarán dentro de la colección **"Sensores"** en Firestore Database, siendo la estructura de cada documento la siguiente:

- **Día:** Corresponde al día en que se ha medido la cantidad de ozono (O_3) y de dióxido de nitrógeno (NO_2).
- **Fecha:** Se almacenará la fecha en que se ha realizado la medición.
- **Hora:** Se almacenará la hora en que se ha realizado la medición.
- **Ozono:** Se almacenará la medición de cada 15 minutos correspondiente a la presencia de ozono (O_3). La medición se registrará en partes por billón (ppb).
- **NO2:** Se almacenará la medición promedio de cada hora correspondiente a la presencia de dióxido de nitrógeno (NO_2). La medición se registrará en partes por billón (ppb).

La Figura 2.6 representa el esquema de la colección llamada "Sensores" en la cual se almacenarán los datos obtenidos desde el módulo sensor, es importante destacar que se ha decidido no crear dos colecciones por separado para cada sensor, ya que la toma de datos de ambos se hará al mismo tiempo, por lo cual las variables Fecha, Día y Hora serán las mismas y es adecuado guardar la toma de datos de ambos sensores dentro de la misma colección.

Esquema Colección Sensores

```
{
   "documents": [
      {
         "name": "Sensores",
         "fields": {
            "Dia": string,
            "Fecha": string,
            "Hora": string,
            "Ozono": string,
            "NO2": string
         },
      }
   ]
}
```

Figura 2.6 Esquema de colección Sensores

2.1.8 Diseño de módulo para obtención de datos

Para el funcionamiento del módulo sensor se utilizarán los sensores y elementos electrónicos mencionados en el subcapítulo 1.4, a continuación, se detallan los pines de conexión de cada sensor hacia el dispositivo ESP32 38 Pines ESP WROOM 32. Para la conexión del sensor MQ-131, con sus pines: GND, AOUT, DOUT y VCC, se presenta la Tabla 2.7:

Tabla 2.7 Pines de conexión del sensor MQ-131

Pines MQ-131	Pines ESP32 38 Pines
GND	GND
AOUT	GPIO35
DOUT	No se utiliza
VCC	5 [V]

Para el caso del sensor MICS-6814 con sus pines: CO, NH3, NO2, GND y +5V, se tiene la conexión de la Tabla 2.8:

Tabla 2.8 Pines de conexión del sensor MICS-6814

Pines MICS-6814	Pines ESP32 NodeMCU
CO	No se utiliza
NH3	No se utiliza
NO2	GPIO32
GND	GND
+5V	5 [V]

Los pines CO y NH3 del sensor MICS-6814 no se usan, dado que las mediciones de dichas partículas no forman parte de los objetivos de este trabajo. Por último, se tiene el sensor BMP280, del cual solamente se usarán los pines: VCC, GND, SCL y SDA, con la conexión de la Tabla 2.9:

Tabla 2.9 Pines de conexión sensor BMP280

Pines BMP280	Pines ESP32 NodeMCU
VCC	3.3 [V]
GND	GND
SCL	GPIO22
SDA	GPIO21

En la Figura 2.7 se muestra el diseño de la conexión de los sensores y elementos electrónicos para formar el módulo sensor. Para diseñar la placa electrónica se utilizó el software EasyEDA, muy útil para este propósito.

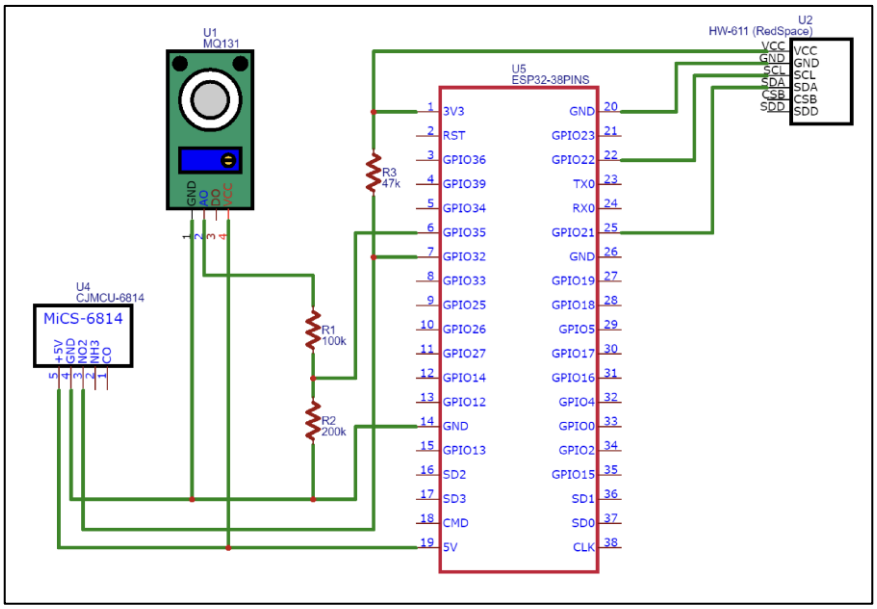


Figura 2.7 Esquema de conexión del módulo sensor

Luego de realizar el esquema de conexión se procede a convertir el esquemático a PCB, para continuar con las rutas de conexión de todos los sensores hacia la ESP32 de 38 pines, tal como se evidencia en la Figura 2.8.

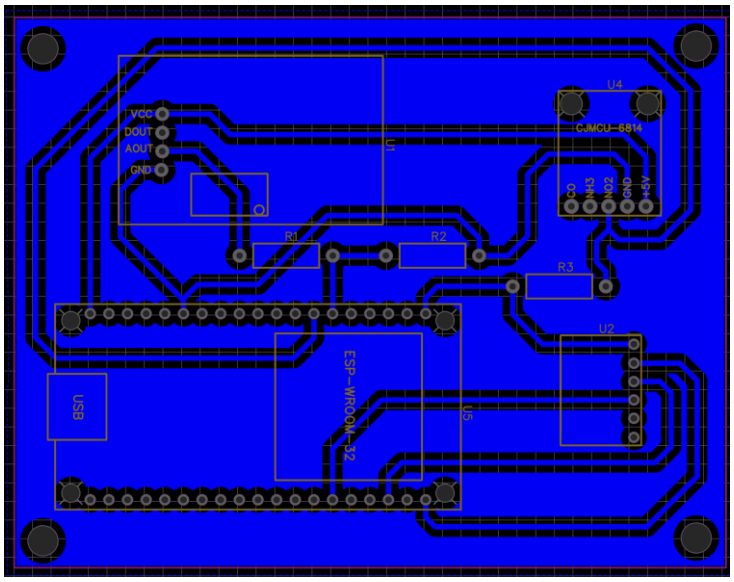


Figura 2.8 Generación de esquemático a PCB

2.1.9 Calibración de sensores

Para un funcionamiento correcto del dispositivo se debe realizar un calentamiento de los sensores MQ-131 y MICS-6814 durante 48 horas.

Los sensores deben ser calibrados para obtener un resultado adecuado, para ello primero se calcula la ecuación de la curva a partir de las gráficas obtenidas de la hoja de datos de los sensores, las cuales se encuentran en la Figura 1.1 y Figura 1.3.

En la Figura 2.9 se presenta la ecuación de la línea de tendencia de medición de ozono del sensor MQ-131.

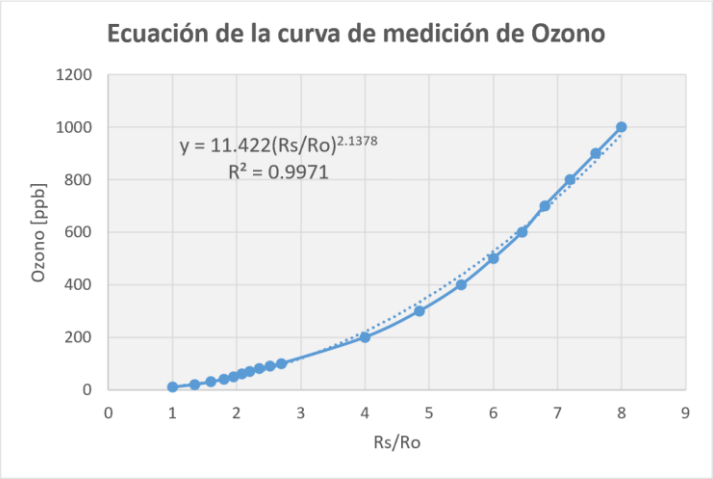


Figura 2.9 Gráfica de ecuación de la curva de Ozono

De esta forma la Ecuación 2.1 representa la medición de ozono en [ppb].

$$ozono\ [ppb] = 11.422\left(\frac{Rs}{Ro}\right)^{2.1378}$$

Ecuación 2.1 Curva de medición de ozono

Siendo el valor de *Rs* representado por la Ecuación 2.2:

$$Rs = Rl * \left[\frac{V - \left(va * \frac{V}{ADC}\right)}{va * \frac{V}{ADC}}\right]$$

Ecuación 2.2 Cálculo de Rs

Para el sensor MQ-131 corresponden los siguientes valores:

Rl: Resistencia de carga de 1 [kΩ]

V: voltaje del sensor de 5 [V]

Va: Lectura de valor analógico

ADC: Convertidor analógico a digital con resolución de 12 bits para ESP32, es decir el valor máximo de lectura analógica da un valor digital de 4095.

De manera similar se calcula la ecuación de la línea de tendencia de medición de dióxido de nitrógeno del sensor MICS-6814 la cual se representa en la Figura 2.10.

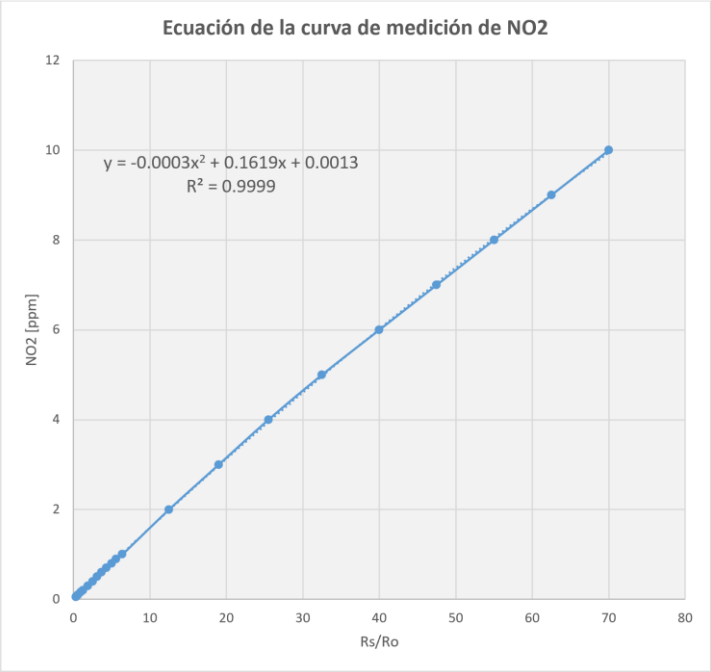


Figura 2.10 Gráfica de ecuación de la curva de NO_2

La medición de dióxido de nitrógeno se encuentra representada en la Ecuación 2.3.

$$NO2\ [ppm] = -0.0003\left(\frac{Rs}{Ro}\right)^2 + 0.1619\left(\frac{Rs}{Ro}\right) + 0.0013$$

Ecuación 2.3 Curva de medición de NO_2

El valor de *Rs* corresponde al mismo representado en la Ecuación 2.2, la única diferencia es que en este caso el valor de *Rl es* de 121 [Ω].

Según la hoja de datos del sensor MQ-131, Ro es la resistencia del sensor en un ambiente de medición con las siguientes condiciones: concentración de ozono 50 ppm, temperatura 20℃, humedad relativa 65%. En este proyecto se determina el valor de Ro al momento de comparar los valores medidos con los valores proporcionados por la Red Metropolitana de Monitoreo Atmosférico de Quito, de manera de lograr una adecuada correspondencia entre los dos conjuntos de valores.

2.2 Implementación del módulo sensor

En esta sección se presenta el proceso para implementar el prototipo del módulo sensor propuesto en el apartado 2.1. El proceso comienza con la instalación del software para programar la ESP32 de 38 Pines, luego de ello se implementa la placa electrónica dado

que previamente se ha realizado su diseño, culminando con la creación y configuración para la conexión con el servicio Cloud Firestore de Firebase.

2.2.1 Creación y configuración de proyecto Cloud Firestore

Se debe tener una cuenta de Google con la cual se acceda a los servicios de Firebase. Se ingresa en la URL *Firebase.google.com*, se escoge *Comienza ahora,* y se agrega un nuevo proyecto, en este caso el proyecto será “Sensores” como se evidencia en la Figura 2.11:

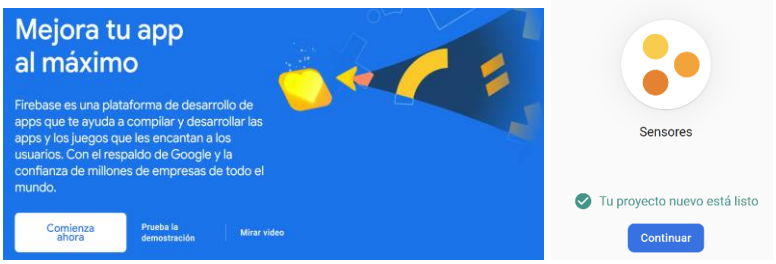


Figura 2.11 Acceso a servicios de Firebase

Se debe seleccionar el servicio a usar, en este caso Firestore Database, luego de ello se ingresa en *Crear base de datos.* Primero se debe seleccionar el *Modo de Producción,* posteriormente colocar la ubicación de southamerica-east1, así se tendrá creada la base de datos como se observa en la Figura 2.12:

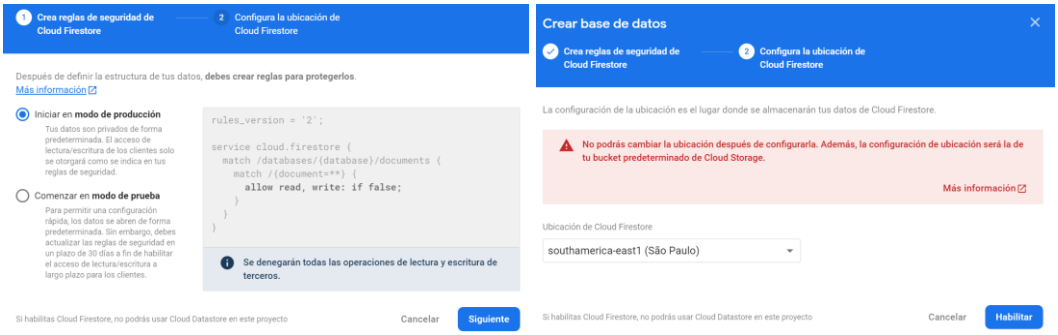


Figura 2.12 Configuración de Cloud Firestore

Es importante actualizar las reglas dentro del proyecto de Cloud Firestore, para evitar que otras personas que no estén autenticadas tengan acceso de lectura/escritura de los datos. Para esto, se debe ir a la opción de Reglas e ingresar el comando de la línea #5 la cual hace referencia a que la lectura será permitida a los usuarios que estén registrados, mientras que en la línea #6 se permite la escritura solamente si el usuario corresponde al UID establecido, tal como se indica en la Figura 2.13:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read: if request.auth !=null;
      allow write: if request.auth.uid == 'd9fk0hv80xTadFL3hUo5Zg3Ez0C2';
    }
  }
}
```

Figura 2.13 Configuración de Reglas

El UID del usuario se obtiene al ir a la opción de *Authentication*, allí se agrega un usuario y al ser agregado se muestran sus credenciales como en la Figura 2.14. Las credenciales son necesarias para ingresar las constantes USER_EMAIL y USER_PASSWORD.

Figura 2.14 Agregación de usuario para escritura de datos

Se procede a ingresar en *Configuración del Proyecto*, para obtener los parámetros necesarios para la comunicación del módulo sensor con Cloud Firestore dentro del código de Arduino IDE, dado que “Clave de API web” e “ID del proyecto” irán asignados a las variables API_KEY y FIREBASE_PROJECT_ID, respectivamente.

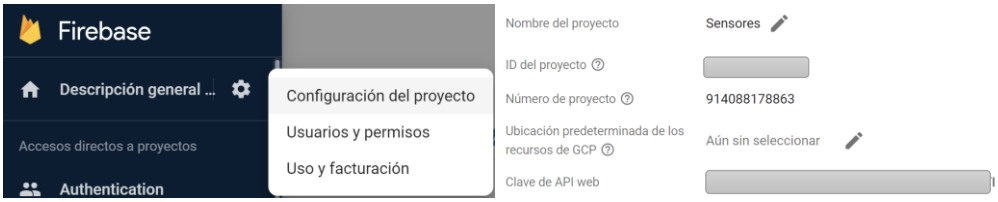


Figura 2.15 Parámetros del proyecto de Firebase

2.2.2 Elaboración y montaje de placa electrónica

En función del diseño realizado en la subsección 2.1.8, se procede a elaborar la placa electrónica para la implementación del dispositivo de sensado, tal como se muestra en la Figura 2.16:

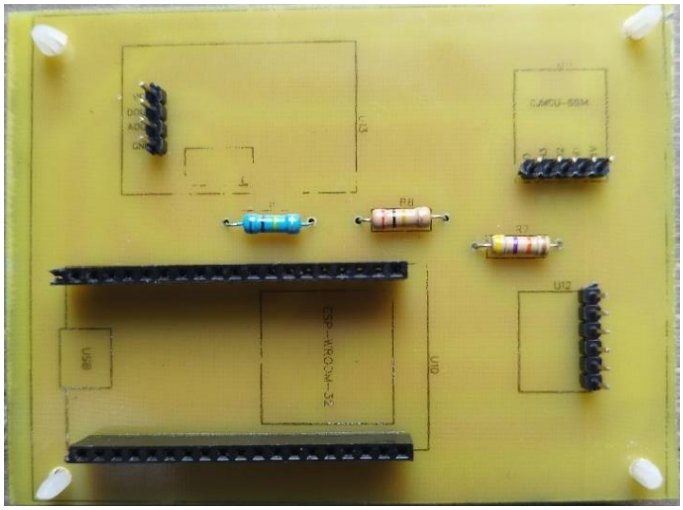


Figura 2.16 Implementación de la placa electrónica

Ahora se procede a realizar el montaje de la placa electrónica dentro del contenedor plástico adaptado para el uso del módulo sensor, como se observa en la Figura 2.17; una vez implementado el dispositivo se continúa con la siguiente subsección.

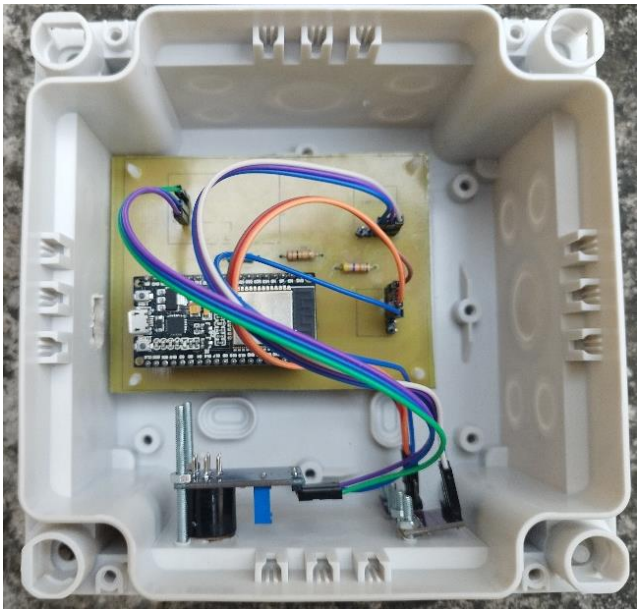


Figura 2.17 Montaje de placa electrónica

2.2.3 Programación del módulo para recolección de datos

En esta subsección se hace mención del proceso para llevar a cabo la programación de la placa de desarrollo ESP32 de 38 pines, lo cual se realiza mediante el uso del software Arduino IDE, por lo cual se detallan las librerías a utilizar, además de las variables que se han definido y las funciones implementadas. El programa desarrollado en Arduino IDE se encuentra en ANEXOS.

Librerías

El uso de librerías permite un beneficio para la programación del microcontrolador, de tal forma que en la Tabla 2.10 se detallan las librerías usadas para el presente caso:

Tabla 2.10 Librerías utilizadas

Librería	Descripción	Versión
WiFi.h	Hace uso del módulo WiFi para realizar una comunicación de forma inalámbrica, es decir que se envíe o reciba datos conectado a una red WiFi.	1.2.7
WiFiManager.h	Permite configurar la red WiFi a la cual se conectará el ESP32 mediante un portal web cautivo.	2.0.16
WiFiUdp.h	Sirve para crear un objeto del tipo UDP para enviar o recibir datos desde WiFi, además de servir para la comunicación mediante UDP hacia el servidor NTP.	4.3.10
NTPClient.h	Permite realizar una conexión con un servidor NTP para ajustar la zona horaria y luego obtener la hora actual.	3.2.1
Firebase_ESP_Client.h	Utilizada para la comunicación con Firebase, de tal forma que se usen los servicios proporcionados por la plataforma.	4.3.10
Time.h	Permite realizar conversiones de tiempo para formatos más comprensibles o formatos personalizados.	1.6.1
Adafruit_BMP280.h	Permite interactuar con mediciones de temperatura y presión atmosférica del sensor BMP280.	2.6.8

Funciones

En la Tabla 2.11 se presentan las funciones usadas en la programación del ESP32 de 38 pines, de tal forma de explicar la funcionalidad de cada una:

Tabla 2.11 Funciones implementadas

Función	Descripción
resultado_ppb_mqq131	Función del tipo float, para hacer lectura del pin del sensor MQ-131, luego de su debido procesamiento se retorna el valor en partes por billón (ppb).
resultado_ppb_mics6814	Función del tipo float, para hacer lectura del pin del sensor MICS-6814, luego de su debido procesamiento se retorna el valor en partes por billón (ppb).
obtenerLecturaSensores	Función en la cual se hace lectura de las funciones respectivas de cada sensor.
obtenerDiaFechaHora	Función que usa la librería Time.h para obtener la fecha, día y hora.
enviarCloudFirestore	Función que envía los datos medidos a Cloud Firestore.

Variables

En la Tabla 2.12 se presentan las variables definidas para la programación del ESP32 de 38 pines, las cuales se explican a detalle:

Tabla 2.12 Variables implementadas

Nombre	Descripción	Tipo
Pin_MQ131	Variable para almacenar el pin del sensor MQ-131.	Int
Pin_MICS6814	Variable para almacenar el pin del sensor MICS-6814.	Int
Temperatura	Variable para almacenar la lectura de temperatura del sensor BMP280.	Float
Ozono	Variable para almacenar la lectura del sensor MQ-131.	Float
NO2	Variable para almacenar la lectura del sensor MICS-6814.	Float
tiempoAnterior	Variable que registra el tiempo anterior en cada intervalo de 15 minutos para la toma de datos.	unsigned long
datoEnviado	Variable para validar el envío a Cloud Firestore.	Bool
Dia	Variable que registra el día de medición.	String
Hora	Variable que registra la hora de medición.	String
stringFecha	Variable que registra la fecha de medición de Ozono y NO_2 en formato año/mes/día.	String
Days	Array que almacena nombres de los días de la semana.	String[7]
bmp	Variable declarada para interacciones con los métodos del sensor BMP280.	Adafruit_BMP280
ntpUDP	Variable declarada para tener comunicación UDP.	WiFiUDP
timeClient	Variable para obtener la hora actual mediante la comunicación con un servidor NTP.	NTPClient
fbdo	Variable para el almacenamiento e interacción de datos dentro de la plataforma Firebase.	Firebase Data
auth	Variable para validaciones en el proceso de autenticación de Firebase.	Firebase Auth
config	Variable para realizar configuraciones de la conexión hacia Firebase.	Firebase Config
wifiManager	Crea una instancia de la clase WiFiManager para configurar WiFi en la ESP32.	WiFiManager
documentPath	Variable del nombre de la colección en Cloud Firestore.	String
content	Variable para cargar objetos JSON para envío de datos.	Firebase Json

Constantes

En la Tabla 2.13 se presentan las constantes definidas para la programación del ESP32:

Tabla 2.13 Constantes implementadas

Nombre	Descripción
API_KEY	Código de llave del proyecto Firebase.
USER_EMAIL	Email del usuario para autenticarse en Cloud Firestore.
USER_PASSWORD	Clave del usuario que se autentica en Cloud Firestore.
FIREBASE_PROJECT_ID	Identificador del proyecto de Firebase llamado “Sensores”.
Utc0OffsetInSeconds	Presenta el desplazamiento horario para ajuste de zona horaria local de Ecuador.

2.3 Implementación de aplicación web

En la sección 2.3 se presenta el proceso para el diseño de la aplicación web, siendo importante la mención de las herramientas y tecnologías aplicadas.

2.3.1 Actualización de tablero Kanban

Dada la metodología implementada para este nuevo apartado relacionado a la Implementación de la aplicación web, se procede a actualizar las actividades registradas y cambiar el estado de ellas, tal como se observa en la Figura 2.18, existen actividades que se han terminado y algunas que actualmente se encuentran en progreso.

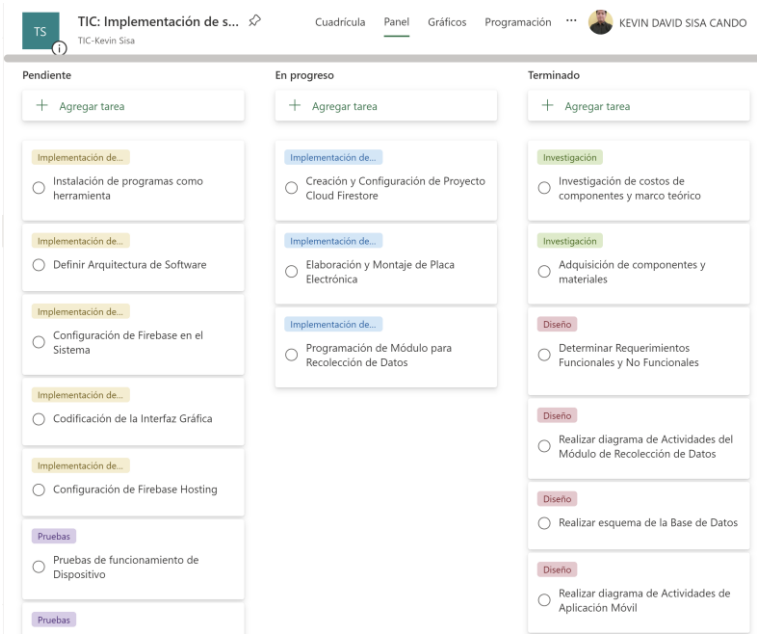


Figura 2.18 Actualización del tablero Kanban

2.3.2 Instalación de software como herramienta de desarrollo

Las herramientas que se deben instalar para la programación de la aplicación web son: Visual Studio Code, desde su página oficial se procede a descargar el instalador para Windows 8, 10 u 11; y también la instalación de Node.js mediante la cual se crean aplicaciones web, tal como se observa en la Figura 2.19.

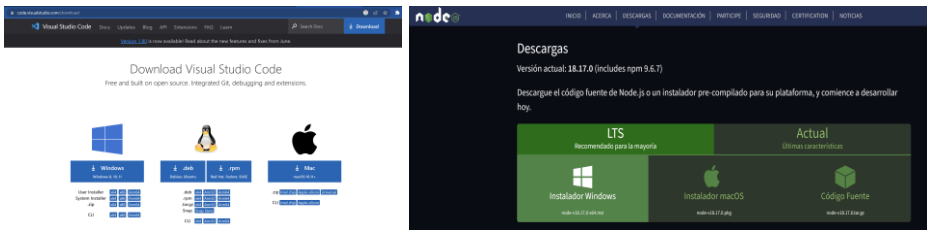


Figura 2.19 Instalación de Visual Studio Code y Node.js

Una vez instalados ambos programas, se procede a crear el proyecto de React.js el cual presenta el nombre “app-sensores”, para lo cual se ejecutará el comando “*npx create-react-app app-sensores*”, de esa forma se creará un directorio llamado *app-sensores* el cual contendrá los archivos necesarios para la configuración del proyecto y de la aplicación.

Dentro del directorio “src” del proyecto “app-sensores” se crearán varios directorios ordenados de la siguiente forma:

- **Components:** Contendrán los archivos JSX de componentes definidos.
- **Context:** Contendrá el archivo JSX que validará la autenticación del usuario en Firebase, en caso de tratarse de registro, inicio o cierre de sesión.
- **Firebase:** Contendrá el archivo JS con los parámetros de conexión a Firebase.
- **Images:** Contendrá las imágenes usadas para la aplicación web.
- **Route:** Contendrá el archivo JSX de ruta protegida de las distintas URL.
- **Styles:** Contendrá los archivos CSS definidos para los estilos de cada vista.
- **Views:** Contendrá los archivos JSX de las vistas definidas para la aplicación.

La Figura 2.20 presenta el orden de los directorios dentro del proyecto creado.

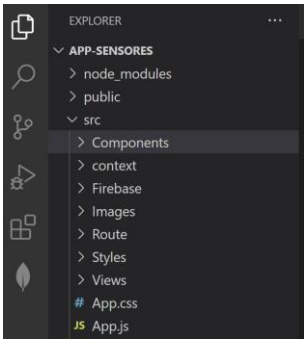


Figura 2.20 Directorios dentro de src de app-sensores

2.3.3 Arquitectura del software

La estructura de la aplicación web a desarrollar tendrá el patrón MVC (Modelo-Vista-Controlador). Teniendo el modelo que describe la forma para acceder a los datos registrados en Cloud Firestore, el controlador que hace uso del servicio de Firebase (Autenticación y conexión a la base de datos), además de los componentes definidos en React para integrarse con la vista, y finalmente la vista, en la cual se tiene definidos los componentes para gestión de la funcionalidad y diseño de cada interfaz de la aplicación web.

2.3.4 Bibliotecas

La Tabla 2.14 presenta la lista de comandos ejecutados desde un terminal dentro de la carpeta del proyecto para instalar las bibliotecas a usarse para el desarrollo y configuración de la aplicación web.

Tabla 2.14 Comandos de instalación de bibliotecas

Comando	Descripción
npm install react-router-dom@6.0.0	Permite instalar la biblioteca de enrutamiento para la aplicación web.
npm install firebase	Permite el uso del servicio de Firebase y sus respectivas dependencias.
npm install bootstrap reactstrap	Permite el uso de componentes que integran apariencia y estilo de Bootstrap con React.
npm i bootstrap-icons	Permite realizar la instalación de íconos de Bootstrap.
npm install recharts	Esta biblioteca brinda el uso de gráficos para React, de tal forma que se desarrollen distintos tipos de gráficos.
npm i react-data-table-component	La biblioteca permite generar una tabla de datos la cual puede ser personalizada y usada para aplicaciones React.
npm install react-datepicker	Esta biblioteca permite el uso de calendarios para selección de fechas.

2.3.5 Configuración de Firebase en el sistema

En el Código 2.1 se muestra la creación del archivo `conexion.js` dentro del directorio Firebase, el cual tendrá los parámetros de configuración de Firebase obtenidos desde la opción de Configuración de Proyecto. En las líneas #18 y #19 se exportan las constantes `auth`, la cual obtiene una instancia del servicio de Authentication de Firebase, y `db`, la cual permite acceder al servicio de Firestore, ambas se usarán posteriormente.

src > Firebase > JS conexion.js > ...

```
1   //Código obtenido desde la consola de Firebase
2   //Uso de parámetros para el funcionamiento de Firebase en el proyecto
3   import { getAuth } from "firebase/auth";
4   import firebase from "firebase/compat/app";
5   import "firebase/compat/firestore";
6   import "firebase/compat/auth";
7
8   export const app = firebase.initializeApp({
9     //parámetros de configuración de Firebase
10    "projectId": "sensores-99f30",
11    "appId": "1:914088178863:web:e62210021def48ed3a905a",
12    "storageBucket": "sensores-99f30.appspot.com",
13    "apiKey": "AIzaSyCFEvMx4ALc2VkhQgzJ1M-S19pKuCxUGTI",
14    "authDomain": "sensores-99f30.firebaseapp.com",
15    "messagingSenderId": "914088178863"
16  });
17
18  export const auth = getAuth(app);
19  export const db = app.firestore();
```

Código 2.1 Parámetros de configuración de Firebase

- **Configuración del servicio de autenticación**

Para configurar el método de autenticación en Firebase habrá que dirigirse al apartado de *Authentication*, en la Figura 2.21 se muestra la selección *Agregar proveedor nuevo,* allí se escoge *correo electrónico/contraseña* y se agrega un usuario.

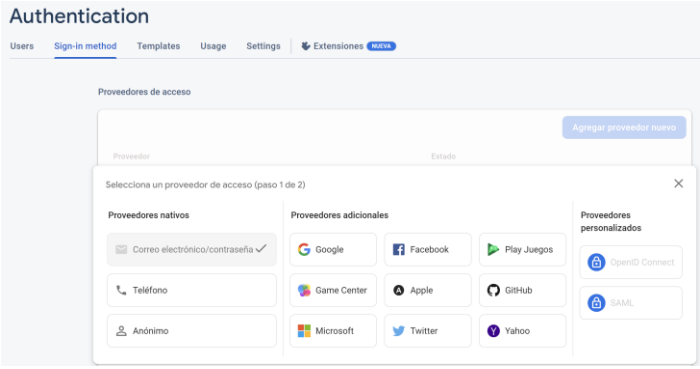


Figura 2.21 Configuración de *Authentication* en Firebase

En el archivo `authContext.jsx` se establece un componente llamado `AuthProvider()` el cual sirve de proveedor del contexto definido en la línea #7, de tal manera que dentro del componente se hace uso de las funciones: `createUserWithEmailAndPassword()` para crear un usuario en Firebase, `signInWithEmailAndPassword()` para iniciar sesión de usuario y `signOut()` para cerrar sesión. Estas funciones son obtenidas del servicio de Firebase.

```
src > context > authContext.jsx > ...
6   //Creación del contexto de autenticación
7   export const authContext = createContext();
8
9   //Hook personalizado que usa el contexto de autenticación
10  export const useAuth = () => {
11      const context = useContext(authContext);
12      if (!context) throw new Error("No hay un auth provider")
13      return context;
14  };
15
16  //Componente AuthProvider que sirve como proveedor del contexto
17  export function AuthProvider({children}){
18      const [loading, setLoading] = useState(true);
19      const [user, setUser] = useState(null);
20      //Función para registrar un usuario mediante correo y contraseña
21      const signup = async(email, password) => {
22          await createUserWithEmailAndPassword(auth, email, password);
23      }
24      //Función para iniciar sesión
25      const login = async(email, password) => {
26          await signInWithEmailAndPassword(auth, email, password);
27      }
28      //Función para cerrar sesión
29      const logout = () => signOut(auth)
30
31      //Mediante useEffect se toma los cambios de estado de autenticación
32      useEffect(() => {
33          //onAuthStateChanged escucha los cambios de estado de la autenticación
34          const unSubscribe = onAuthStateChanged(auth, (currentUser) =>{
35              setUser(currentUser);
36              setLoading(false);
37          });
38          return () => unSubscribe();
39      },[])
40
41      return (
42          <authContext.Provider value={{signup, login, user, logout, loading}}>{children}</authContext.Provider>
43      )
44
45  }
```

Código 2.2 Componente de proveedor de contexto AuthProvider

- **Configuración del servicio de acceso a Cloud Firestore**

Cloud Firestore brinda el servicio de una base de datos NoSQL, por lo cual se crean documentos dentro de la colección “Sensores” al enviarse los datos medidos desde el módulo sensor a Cloud Firestore cada 15 minutos. Para acceder a los datos de los documentos se usan consultas simples o compuestas, en este caso las consultas serán del tipo compuestas, por lo cual primero deberán ser creadas, tal como se muestra en la Figura 2.22, se crea la consulta con los campos elegidos.

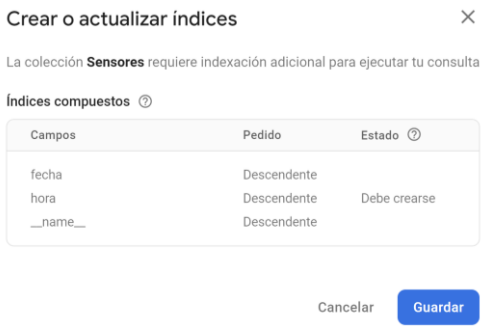


Figura 2.22 Creación de consulta compuesta a Cloud Firestore

En el Código 2.3 se observa la función `getDataForTable()` la cual en la línea #34 hace uso de la consulta compuesta creada para obtener los documentos de la colección “Sensores”, para ello usa el método `db.collection('Sensores')`, para acceder a la colección, luego `orderBy('fecha','desc').orderBy('hora','desc')` para ordenar los documentos por fecha y hora de forma descendente. En la línea #43 se actualiza el estado del objeto `data` con los documentos almacenados en el arreglo `docs`.

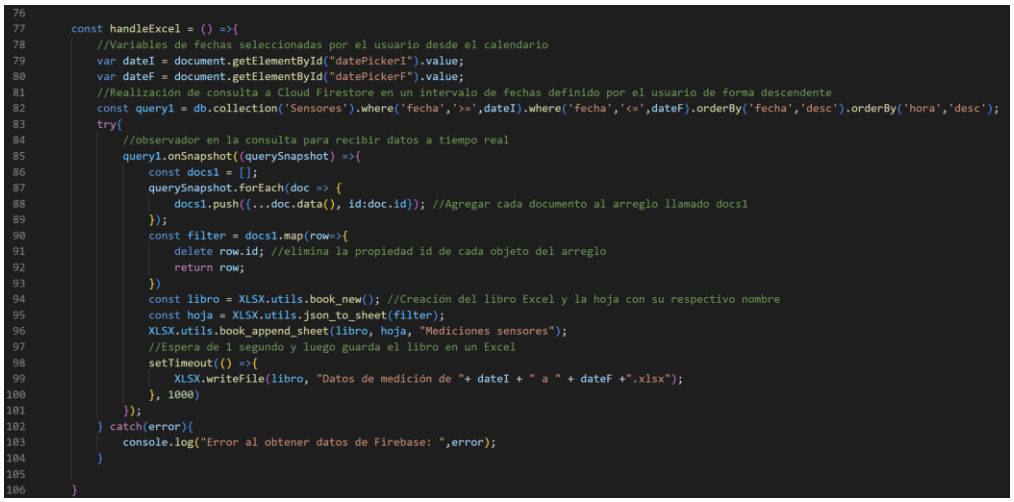
```
31  //Función para obtener datos de Cloud Firestore
32  const getDataForTable = () =>{
33      //Realización de consulta a Cloud Firestore de fechas y horas de registro en orden descendente
34      const query = db.collection('Sensores').orderBy('fecha','desc').orderBy('hora','desc');
35      try{
36          //observador en la consulta para recibir datos a tiempo real
37          query.onSnapshot((querySnapshot) =>{
38              const docs = [];
39              querySnapshot.forEach(doc => {
40                  //Agregar cada documento al arreglo llamado docs
41                  docs.push({...doc.data(), id:doc.id});
42              });
43              setData(docs); //Actualiza el estado de data con el arreglo docs
44          });
45      } catch(error){
46          console.log("Error al obtener los datos: ", error);
47      }
48
49  };
```

Código 2.3 Consulta compuesta para datos de registro en Tabla

En el Código 2.4 se observa la función `handleExcel()`, la cual realiza una consulta a Cloud Firestore, primero usando el método `db.collection('Sensores')` para acceder a la colección “Sensores”, luego se hace uso del método `where('fecha','>=',dateI)`

`.where('fecha','<=',dateF)` para establecer la fecha de consulta en un intervalo de inicio y fin, finalmente `orderBy('fecha','desc').orderBy('hora','desc')` para ordenar los documentos de la consulta por fecha y hora de forma descendente.

Desde la línea #93 a la línea #99 se hace uso de los métodos de la librería para exportar en Excel, siendo que `XLSX.utils.book_new()` permite crear el libro Excel, luego mediante `XLSX.utils.json_to_sheet()` se define la hoja y con `XLSX.utils.book_append_sheet()` se anexa la hoja con el libro y un nombre establecido para la hoja. Finalmente, con `XLSX.writeFile()` se guarda el archivo Excel.



```
76
77      const handleExcel = () =>{
78          //Variables de fechas seleccionadas por el usuario desde el calendario
79          var dateI = document.getElementById("datePickerI").value;
80          var dateF = document.getElementById("datePickerF").value;
81          //Realización de consulta a Cloud Firestore en un intervalo de fechas definido por el usuario de forma descendente
82          const query1 = db.collection('Sensores').where('fecha','>=',dateI).where('fecha','<=',dateF).orderBy('fecha','desc').orderBy('hora','desc');
83          try{
84              //observador en la consulta para recibir datos a tiempo real
85              query1.onSnapshot((querySnapshot) =>{
86                  const docs1 = [];
87                  querySnapshot.forEach(doc => {
88                      docs1.push({...doc.data(), id:doc.id}); //Agregar cada documento al arreglo llamado docs1
89                  });
90                  const filter = docs1.map(row=>{
91                      delete row.id; //elimina la propiedad id de cada objeto del arreglo
92                      return row;
93                  })
94                  const libro = XLSX.utils.book_new(); //Creación del libro Excel y la hoja con su respectivo nombre
95                  const hoja = XLSX.utils.json_to_sheet(filter);
96                  XLSX.utils.book_append_sheet(libro, hoja, "Mediciones sensores");
97                  //Espera de 1 segundo y luego guarda el libro en un Excel
98                  setTimeout(() =>{
99                      XLSX.writeFile(libro, "Datos de medición de "+ dateI + " a " + dateF +".xlsx");
100                 }, 1000)
101             });
102         } catch(error){
103             console.log("Error al obtener datos de Firebase: ",error);
104         }
105
106     }
```

Código 2.4 Consulta compuesta para exportar datos en Excel

Para realizar gráficas se debe acceder a la colección “Sensores” de manera similar a lo explicado anteriormente, por lo cual en el Código 2.5 se define la función `getDataForGraphic()`, la cual realiza la consulta a la colección “Sensores” mediante `db.collection('Sensores')`, luego ordena los documentos de la consulta por fecha y hora de forma descendente mediante el uso del método `orderBy()`.

```
7   function Graphics(){
8       const [data1, setData1] = useState([]);
9
10      //Función para obtener datos en consulta de Cloud Firestore
11      const getDataForGraphic = () =>{
12          //Realiza consulta a Cloud Firestore de Fecha y Hora de registro descendente
13          const query = db.collection('Sensores').orderBy('fecha','asc').orderBy('hora','asc');
14          try {
15              //Observador en la consulta para recibir datos
16              query.onSnapshot((querySnapshot) =>{
17                  const docs = [];
18                  querySnapshot.forEach(doc => {
19                      //Agregar cada documento al arreglo
20                      docs.push({...doc.data(), id:doc.id});
21                  });
22                  setData1(docs); //Actualizar el estado de data con datos del arreglo
23              });
24          } catch (error){
25              console.log("Error al obtener los datos:", error);
26          }
27      };
28
```

Código 2.5 Consulta realizada a Cloud Firestore para generación de gráficas

2.3.6 Codificación de la interfaz gráfica

En la presente subsección se realiza la codificación de los componentes de la interfaz gráfica, en primera instancia se crea el archivo `Table.jsx` en el que se define el componente de Tabla para mostrar los datos almacenados en Cloud Firestore haciendo uso de la función de consulta personalizada explicada en el Código 2.3. Después, con el uso de la biblioteca `DataTable` de `react` se define en el Código 2.6 la creación de la tabla de datos dentro de la etiqueta `<DataTable></DataTable>`, se establecen las columnas de la tabla definidas en el Código 2.7 mediante `columns={columns}`, luego se consumen los datos que se obtienen de la consulta a la base de datos de Cloud Firestore mediante `data={searchFiltered}`, por último se usa `fixedHeader` para tener el encabezado de la tabla fijo en caso de desplazar la pantalla y `pagination` para tener los datos en paginación dentro de la tabla.

```
160                 <div className="table-responsive-lg">
161                     <DataTable
162                     columns={columns}
163                     data={searchFiltered}
164                     fixedHeader
165                     pagination>
166                     </DataTable>
167
168                 </div>
169
```

Código 2.6 Implementación de Tabla con ReactDataTable

```
23      const columns = [
24          { name: 'NO2 [ppb]', selector: row=>row.no2},
25          { name: 'Ozono [ppb]', selector: row=>row.ozono},
26          { name: 'Temperatura [°C]', selector: row=>row.temperatura},
27          { name: 'Fecha', selector: row=>row.fecha},
28          { name: 'Hora', selector: row=>row.hora},
29          { name: 'Día', selector: row=>row.dia },
30      ];
31
```

Código 2.7 Establecimiento de columnas de DataTable

Dentro del mismo componente del archivo `Table.jsx` se maneja la creación de una ventana flotante mediante el uso de Bootstrap, en el Código 2.8 se muestra la etiqueta `<div></div>` en la cual se desarrolla la ventana modal, además del uso de `ReactDatePicker` para elección de fechas desde un calendario en `React`, luego de ello ambas elecciones servirán para establecer una consulta personalizada a Cloud Firestore.

```
145 <div className="modal-body">
146     <p>Seleccione la fecha de inicio:</p>
147     <ReactDatePicker selected={dateIn} onChange={handleSelectedDateI} locale={es} dateFormat={"yyyy/MM/dd"}
        className="datePickerI" value={dateIn} id="datePickerI" autoComplete="off"/>
148     <p>Seleccione la fecha de fin:</p>
149     <ReactDatePicker selected={dateFin} onChange={handleSelectedDateF} locale={es} dateFormat={"yyyy/MM/dd"}
        className="datePickerI" value={dateFin} id="datePickerF" autoComplete="off" minDate={dateIn}  disabled={!dateIn}/>
150 </div>
```

Código 2.8 Implementación de calendario con ReactDatePicker

La generación de gráficas de los datos registrados se la implementa haciendo uso de `ReCharts` de `React`, como se muestra en el Código 2.9, se usa la etiqueta `<ResponsiveContainer/>` estableciendo el tipo de gráfica con `<AreaChart/>`, posteriormente se configuran ciertos parámetros de la gráfica como `CartesianGrid`, `XAxis`, `YAxis`, `ToolTip`, `Legend` y `Area`.

```
return(
    <div className="graphics">
        <div className="graphicNO2">
            <div className="title">
                <h3>GRÁFICA DIÓXIDO DE NITRÓGENO (NO2)</h3>
            </div>
            <ResponsiveContainer width="90%" aspect={4}>
                <AreaChart data={data1} width="500" height="300" margin={{top:10, right:60, left:0, bottom: 0}}>
                    <CartesianGrid strokeDasharray="3 8 3" stroke="grey"/>
                    <XAxis dataKey="fecha"/>
                    <YAxis/>
                    <Tooltip/>
                    <Area type={"monotone"} dataKey="no2" name="NO2 [ppb]" stackId="1" stroke="#565656" fill="#72C8F1"/>
                    <Area type={"monotone"} dataKey="hora" name ="Hora" stackId="1" stroke="blue"/>
                </AreaChart>
            </ResponsiveContainer>
        </div>

        <div className="graphicOzono">
            <div className="title">
                <h3>GRÁFICA OZONO (O3)</h3>
            </div>
            <ResponsiveContainer width="90%" aspect={4}>
                <AreaChart data={data1} width="500" height="300" margin={{top:10, right:60, left:0, bottom: 0}}>
                    <CartesianGrid strokeDasharray="3 8 3" stroke="grey"/>
                    <XAxis dataKey="fecha"/>
                    <YAxis/>
                    <Tooltip/>
                    <Area type={"monotone"} dataKey="ozono" name="Ozono [ppb]" stackId="1" stroke="#565656" fill="#11FB40"/>
                    <Area type={"monotone"} dataKey="hora" name="Hora" stackId="1" stroke="blue"/>
                </AreaChart>
            </ResponsiveContainer>

        </div>
```

Código 2.9 Implementación de gráficas con ReCharts

Como resultado de la implementación, se tiene la interfaz gráfica del inicio de sesión de la Figura 2.23, en caso de no tener una cuenta podrá registrarse desde *Crear nueva cuenta.*

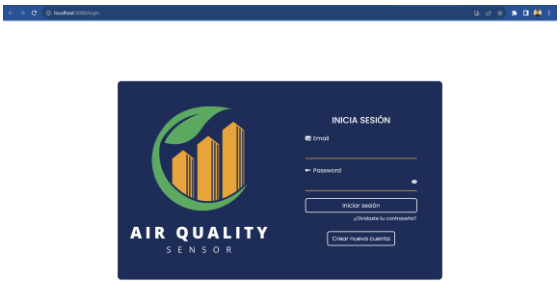


Figura 2.23 Interfaz gráfica de Inicio de sesión

2.3.7 Configuración de Firebase Hosting

Para alojar la aplicación web en el servicio de Firebase Hosting, se ingresa a la sección de *Hosting* del proyecto de Firebase y se debe seleccionar *Comenzar.* Aparecerán los pasos que se deben seguir para el despliegue de la aplicación web, para ello se debe ingresar al directorio del proyecto de la aplicación web desde un Terminal y ejecutar el comando `npm install -g firebase-tools` para instalar Firebase CLI.

Luego, se debe ejecutar `firebase login`, lo cual redireccionará al navegador para seleccionar la cuenta con la cual va a autenticarse. Posteriormente, se ejecuta el comando `firebase init`, allí se elegirá el servicio de Hosting a utilizar de Firebase como se presenta en la Figura 2.24.

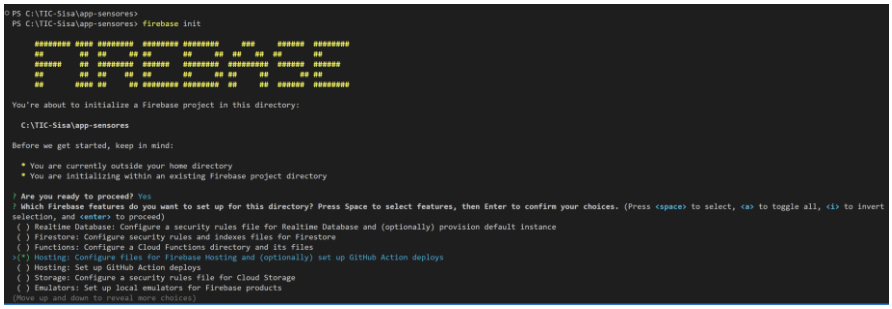


Figura 2.24 Inicialización de proyecto de Firebase

Se deben elegir ciertas configuraciones, tal como se muestra en la Figura 2.25, luego se deberá ejecutar el comando `npm run build`, dado que ese será el directorio que contendrá al proyecto de React.js a desplegar.

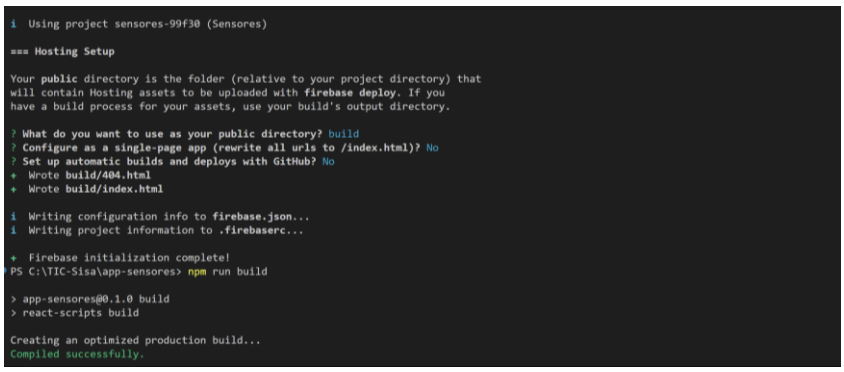


Figura 2.25 Elección de proyecto Firebase y directorio

Por último, se debe ejecutar el comando `firebase deploy` para desplegar la aplicación web y se tendrá el enlace URL de la aplicación web, como se observa en la Figura 2.26.

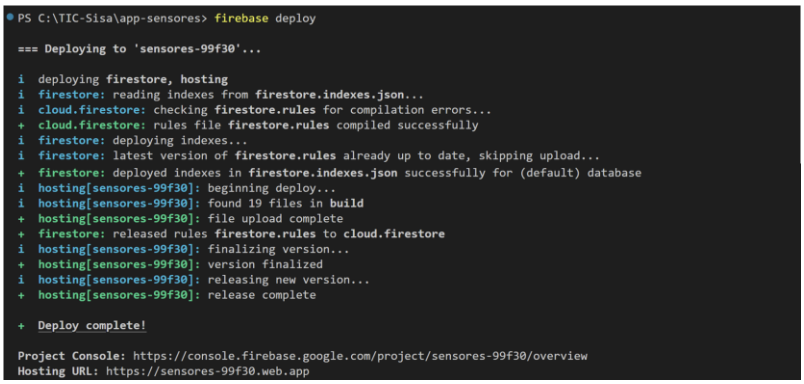


Figura 2.26 Despliegue de aplicación web

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Actualización del tablero Kanban

Según el desarrollo de las actividades, se presenta el avance del tablero Kanban en la Figura 3.1, donde se observan las actividades en progreso y las pendientes que corresponden a la etiqueta de pruebas.

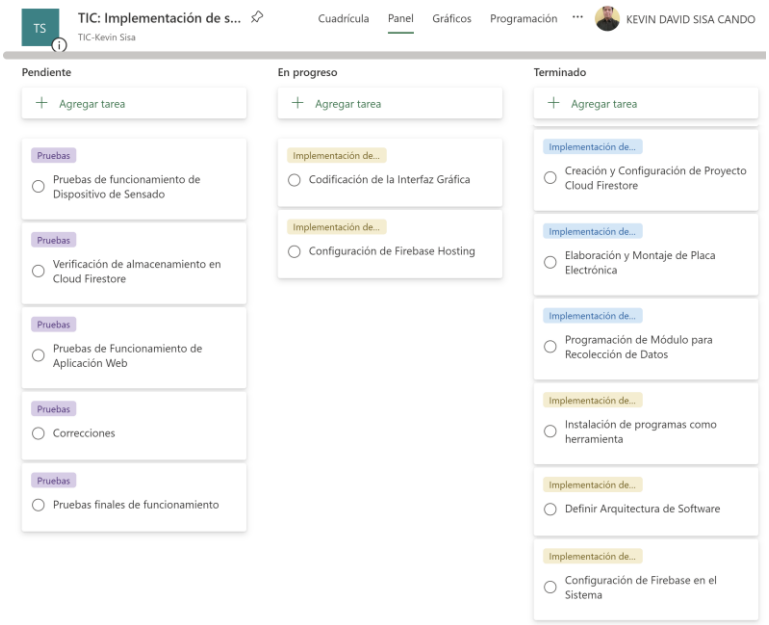


Figura 3.1 Actualización de tablero Kanban en Capítulo 3

3.2 Pruebas de funcionamiento del módulo sensor

En la presente sección se presentan los resultados del funcionamiento del módulo sensor, detallándose el proceso de instalación del dispositivo, para lo cual se adjunta un manual de usuario ubicado en el ANEXO IV.

Al iniciar se ha cargado el sketch “moduloSensor” el cual presenta todo el proceso de conexión a una red WiFi, hasta la obtención de las mediciones de los sensores para ser cargadas en Cloud Firestore en intervalos de 15 minutos. El código del sketch “moduloSensor” se encuentra en el ANEXO I.

Se conectó el dispositivo a una fuente de alimentación y posterior a ello la placa de desarrollo ESP32 crea un punto de acceso WiFi temporal llamado *ESP32-Config,* al cual se debe conectar desde un terminal móvil u ordenador y se redirige a la IP 192.168.4.1, tal como se muestra en la Figura 3.2. Luego de ello se elige *Configure WiFi*.

Figura 3.2 Configuración de WiFi desde la IP 192.168.4.1

Al ingresar se selecciona la red WiFi deseada, posterior a ello se tiene un mensaje de resultado de conexión de la ESP32 a la red WiFi escogida, como se observa en la Figura 3.3.

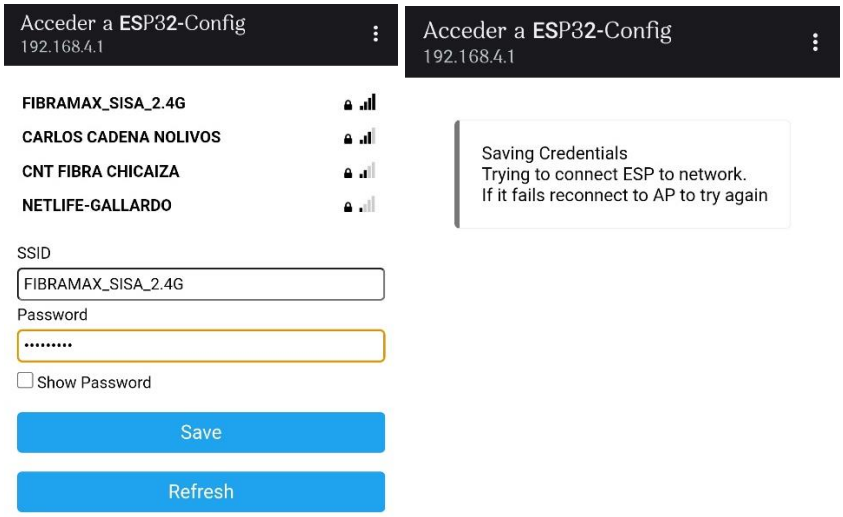


Figura 3.3 Resultado de conexión a red WiFi desde ESP32

De forma complementaria en la Figura 3.4 se tiene el resultado desde la consola de Arduino IDE, en la cual se observa que se ha creado el portal web con la IP 192.168.4.1 y posterior a ello, la ESP32 se conectó a la red WiFi seleccionada teniendo como resultado un aviso de conexión establecida.

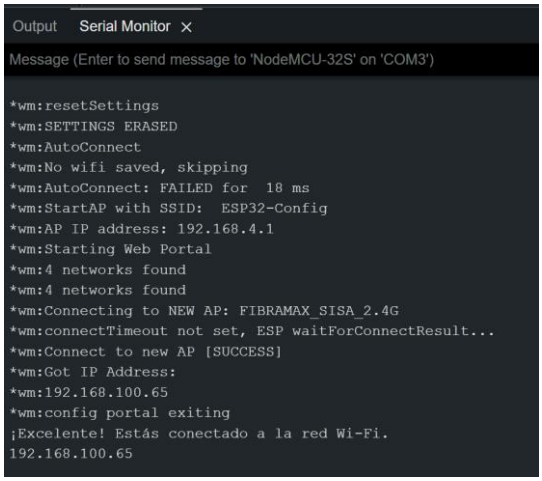


Figura 3.4 Resultado de conexión a red WiFi desde Arduino IDE

Posterior a la conexión se tuvo que esperar el intervalo de tiempo de 15 minutos, establecido para que el dispositivo realice la primera medición de dióxido de nitrógeno (NO_2), ozono (O_3) y temperatura.

3.2.1 Verificación de almacenamiento en Cloud Firestore

Se ingresó al proyecto creado en Firebase desde el navegador para verificar el registro de los datos en la colección “Sensores”, tal como se observa en la Figura 3.5, se tienen los documentos creados con los respectivos campos explicados en la subsección 2.1.7.

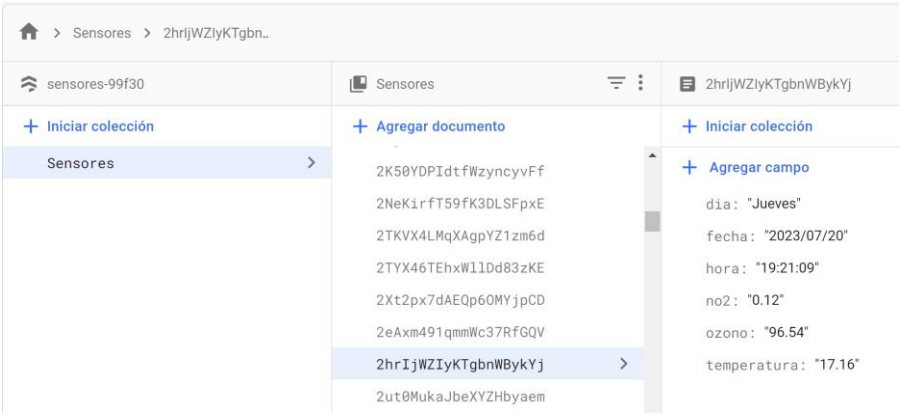


Figura 3.5 Verificación de registro de documentos en colección Sensores

Las pruebas de funcionamiento del módulo sensor se han realizado a partir del 2023/07/20, por lo cual el registro se tiene desde esa fecha.

3.3 Análisis de resultados

3.3.1 Mediciones de ozono

A partir del 20/07/2023 se realizó la instalación del dispositivo en la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito ubicada en el Centro Histórico de Quito. Se ha decidido agrupar por días los resultados de mediciones del sensor MQ-131 para tener un mejor análisis del progreso de las pruebas de funcionamiento. Considerando que las pruebas empezaron a partir del 20/07/2023 y la Figura 3.6 presenta mediciones hasta el 23/07/2023, se observa que la medición de ozono [ppb] del módulo sensor presenta valores demasiados altos en el intervalo de 80 – 120 [ppb], con relación a las mediciones de la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito, la cual entrega valores menores a 30 [ppb] por lo cual se dedujo que la calibración está incorrecta. En esta primera etapa de medición, se realizó la toma de datos cada 30 minutos.

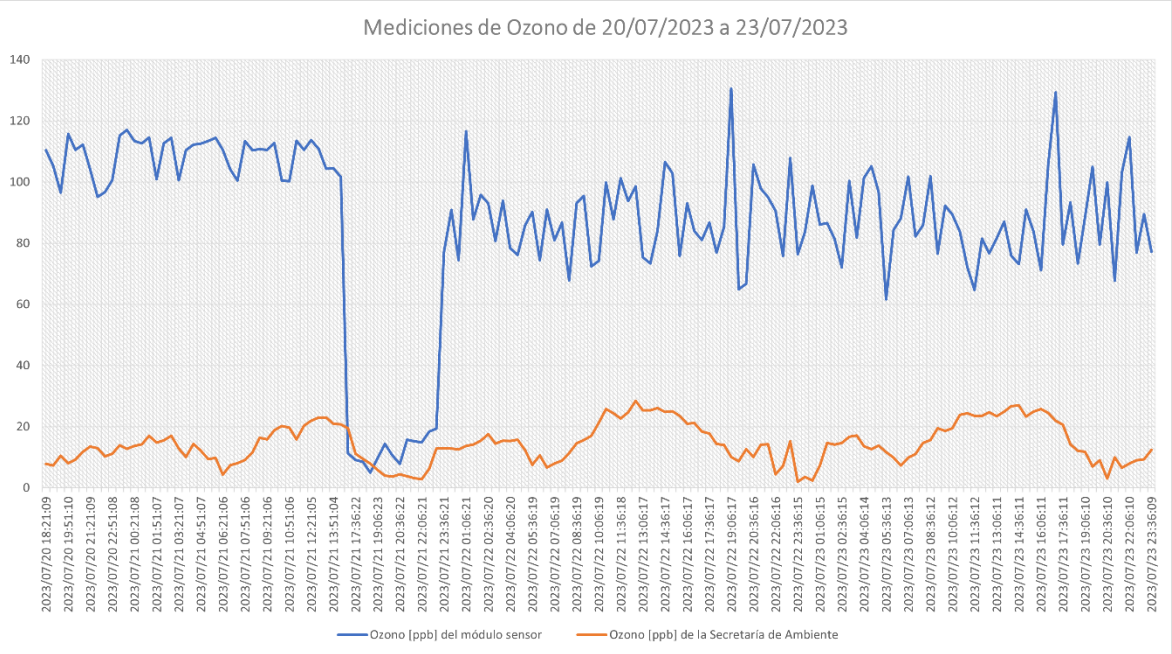


Figura 3.6 Primer análisis de mediciones de ozono

A partir del 24/07/2023 hasta el 29/07/2023, se tiene una diferente calibración del sensor MQ-131 con lo cual se evidencia, en la Figura 3.7, que la medición de ozono se ha estabilizado en ciertas horas del día, por ejemplo, desde las 14:46 p.m. del 24/07/2023 hasta las 10:16 a.m. del 26/07/2023 se tienen valores semejantes a los de la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito, de forma similar sucede desde las 7:37 a.m. del 29/07/2023 hasta las 23:37 p.m. del mismo día, en este intervalo las curvas de medición son idénticas. Sin embargo, se observan picos altos en ciertos intervalos de tiempo.

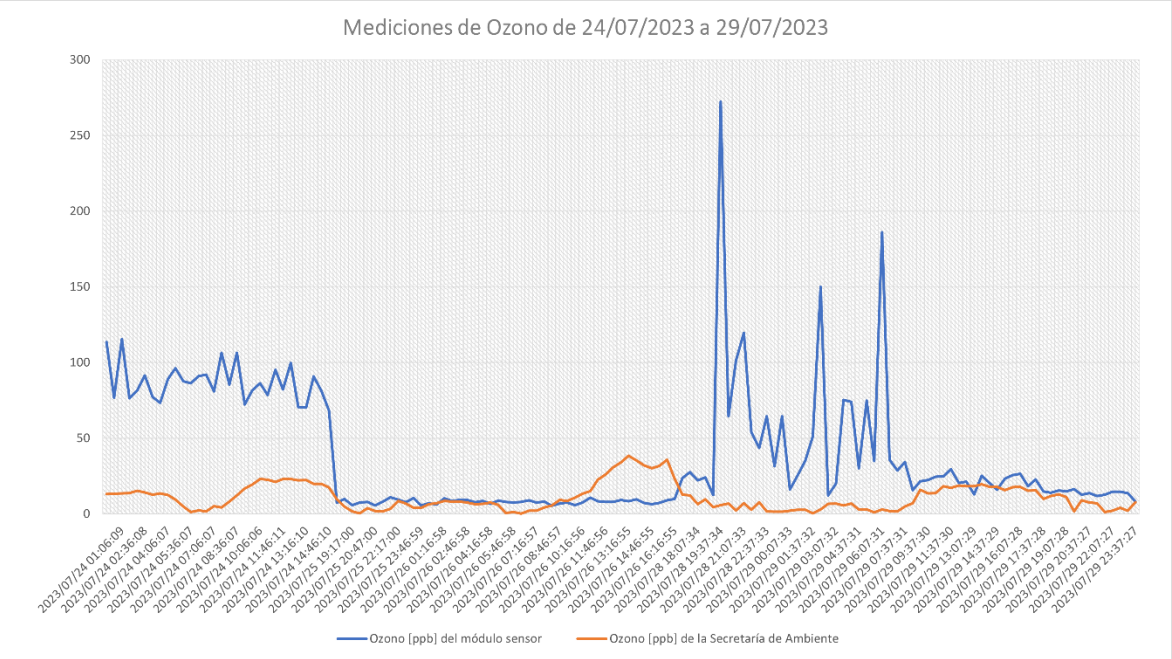


Figura 3.7 Segundo análisis de mediciones de ozono

A partir del 30/07/2023, como se observa en la Figura 3.8, se tienen resultados de medición más equilibrados por parte del sensor MQ-131 en el cual se observa que hasta las 16:23 p.m. del 02/08/2023 la curva de medición sigue una tendencia similar a lo medido por la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito, pero con valores mayores. Luego de ese intervalo de tiempo se observa una descalibración del sensor por lo cual muestra valores de picos altos de medición, llegando hasta 160 [ppb]. A partir de esta etapa se ha considerado que el intervalo de medición sea de 15 minutos.

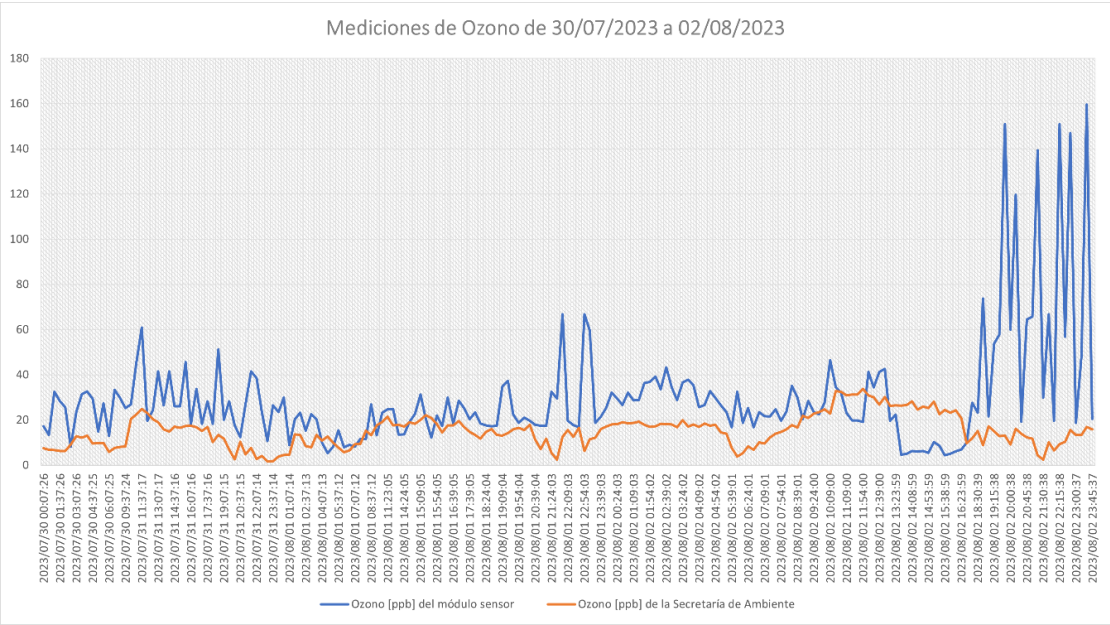


Figura 3.8 Tercer análisis de mediciones de ozono

En la Figura 3.9 se observa que a partir de las 9:45 a.m. del 03/08/2023 se tienen mediciones del sensor MQ-131 que tratan de seguir la tendencia del sensor de la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito, siendo valores más semejantes que los mostrados en la Figura 3.8. Sin embargo, existen ciertas mediciones de valores bajos, las cuales no pueden ser realizadas correctamente por el sensor MQ-131, dado su rango de detección, por tal razón, el sensor tendrá fallas en ciertos puntos de medición mínimos. De forma general se infiere que el valor de calibración adecuado para este sensor, de forma específica para las pruebas realizadas en la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito ubicada en el Centro Histórico, corresponde a ***Ro* =** 15000.

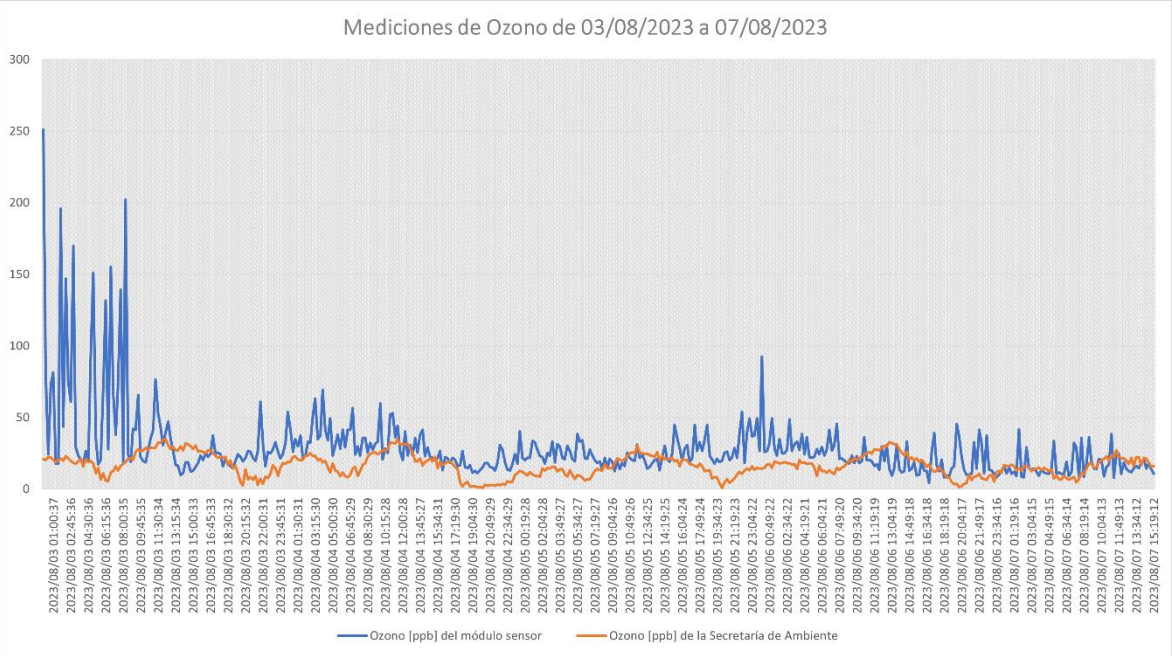


Figura 3.9 Cuarto análisis de mediciones de Ozono

3.3.2 Mediciones de dióxido de nitrógeno

Los resultados de mediciones del sensor MICS-6814 han sido agrupados por días, al igual que en el caso del sensor MQ-131. Considerando que las pruebas empezaron a partir del 20/07/2023 se observa en la Figura 3.10 que al inicio el sensor medía valores cercanos a cero, por lo cual se tuvo que cambiar la calibración, resultando que a partir de las 13:51 p.m. del 21/07/2023 el sensor ha podido realizar mediciones con tendencia a la curva de lo obtenido por la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito, en los valores de pico alto. En esta primera etapa de medición se ha realizado la toma de datos cada 30 minutos.

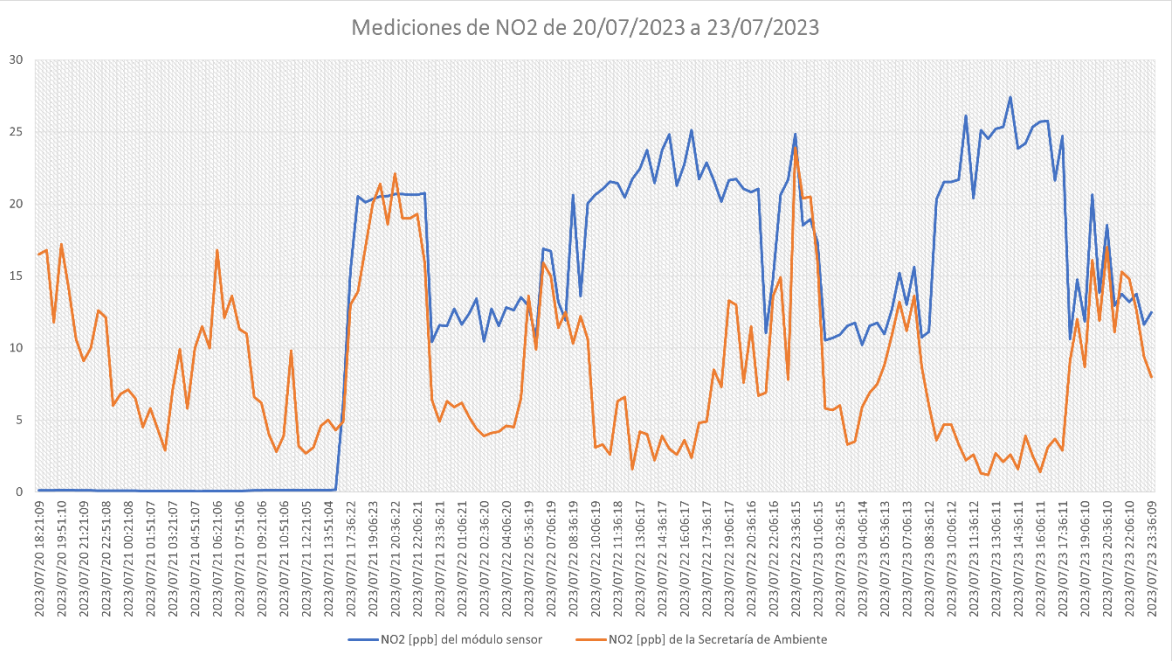


Figura 3.10 Primer análisis de mediciones de dióxido de nitrógeno

Continuando con el registro de mediciones, en la Figura 3.11 se observan los resultados del 24/07/2023 al 29/07/2023, pudiéndose notar que en ciertas zonas las mediciones del sensor MICS-6814 siguen la tendencia de curva de la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito, sin embargo, se tienen otros intervalos en los cuales no existe semejanza.

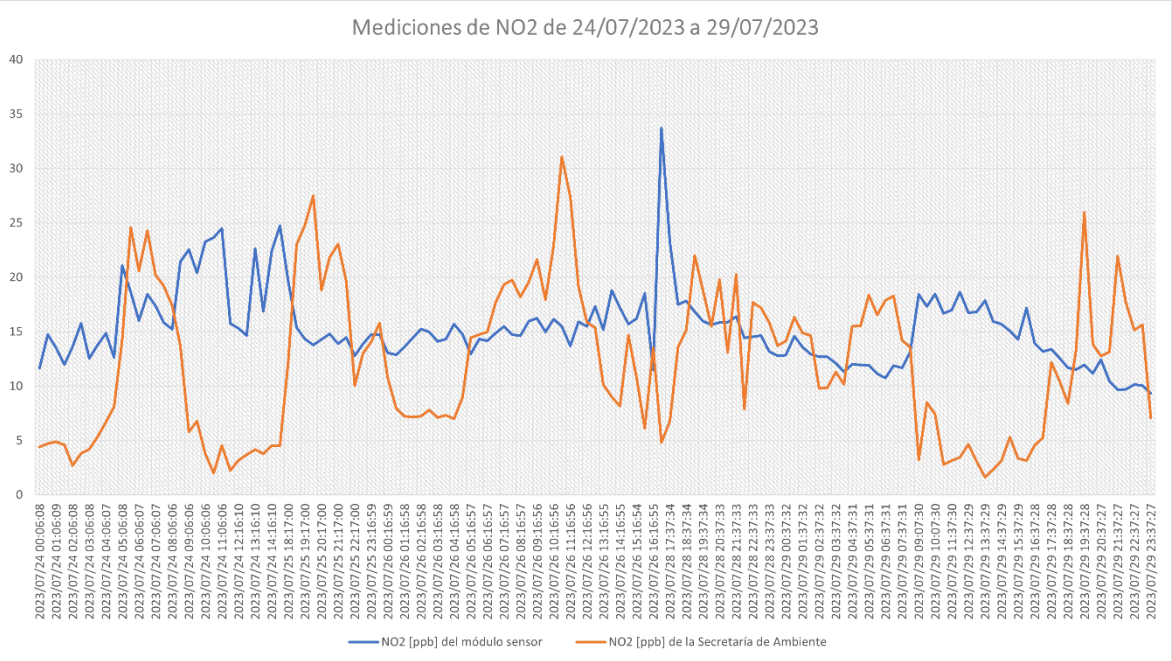


Figura 3.11 Segundo análisis de mediciones de dióxido de nitrógeno

En la Figura 3.12 se observa que desde la media noche del 30/07/2023 hasta las 6:07 am del 31/07/2023 se tienen valores semejantes a la curva de medición de la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito, luego de ello las mediciones presentan curvas distintas; aunque la tendencia de la curva es parecida, los valores medidos por el sensor MICS-6814 son bajos comparados con los de la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito, pudiéndose observar que el sensor no reacciona de manera óptima ante los cambios repentinos de medición. A partir de esta etapa se ha considerado que el intervalo de medición sea de 15 minutos.

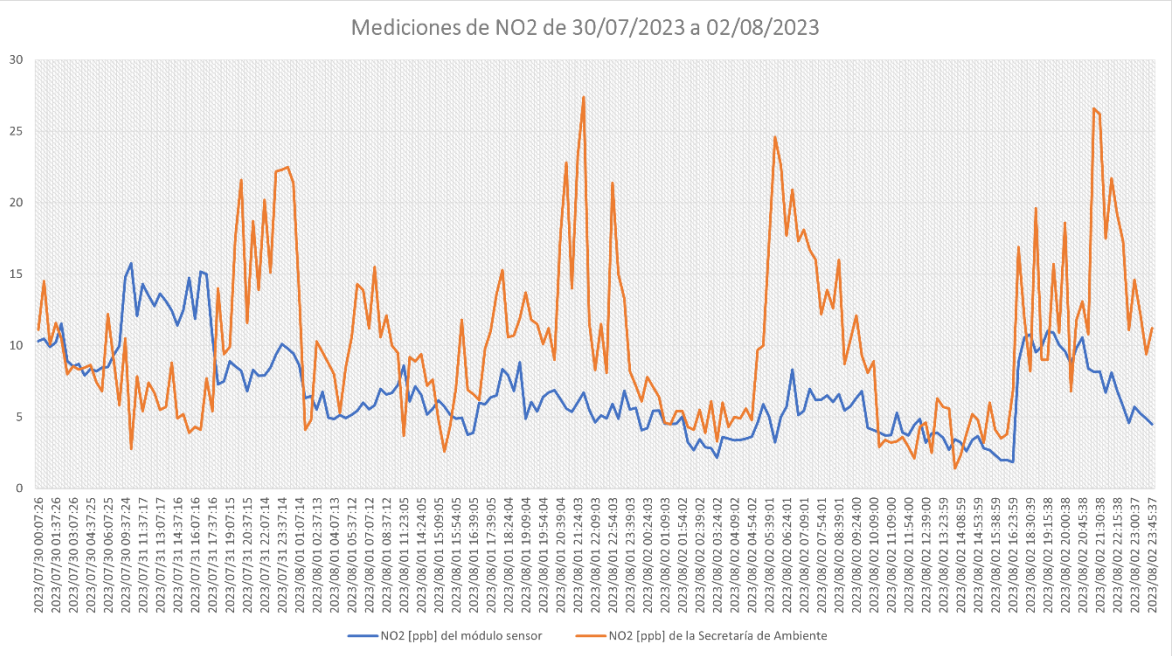


Figura 3.12 Tercer análisis de mediciones de dióxido de nitrógeno

Para las fechas entre el 03/08/2023 y el 07/08/2023 se observan las curvas con picos altos en la Figura 3.13, que en el caso de la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito, los picos llegan hasta 37 [ppb], mientras que las mediciones del sensor MICS-6814 no sobrepasan los valores de 13 [ppb], las mediciones se obtuvieron con el valor de ***Ro* =** 1800 asignado. Al utilizar valores de Ro mayores o menores se tuvieron datos con una diferencia notoria, en referencia a los de la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito, por lo cual se considera al valor de 1800 como el más adecuado para las respectivas mediciones.

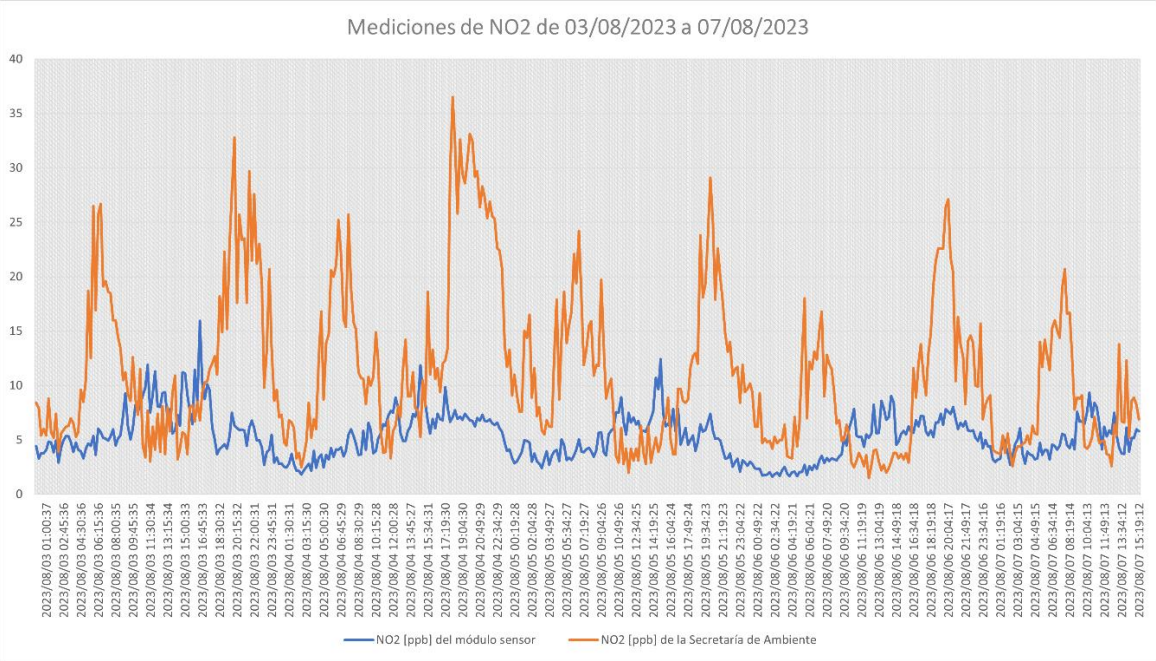


Figura 3.13 Cuarto análisis de mediciones de dióxido de nitrógeno

Como conclusión del proceso desarrollado para la toma de mediciones de dióxido de nitrógeno con el sensor MICS-6814, se menciona que el sensor resulta adecuado para ciertos niveles de referencia de concentración de NO_2, sin embargo, no permite medir adecuadamente los cambios abruptos de concentración, tal como lo hacen los sensores de la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito, adicionalmente a ello, el sensor resulta ser inconsistente para mediciones bajas de NO_2, cercanas al valor cero.

3.3.3 Error relativo de resultados de medición

Para comprobar la viabilidad de las mediciones de los sensores MQ-131 y MICS-6814 se ha realizado el cálculo del error relativo de cada medición realizada, posterior a ello se calculó un promedio del error relativo por día para tener una mejor visualización de los resultados, dado que la cantidad de mediciones es extensa, tal como se observa en la Tabla 3.1.

Tabla 3.1 Error relativo promedio de resultados por día

Prueba	Fecha	Error relativo promedio de Ozono [%]	Error relativo promedio de NO2 [%]
1	20/07/2023	936,25%	98,96%
2	21/07/2023	558,62%	75,11%
3	22/07/2023	613,40%	262,89%
4	23/07/2023	701,72%	377,57%
5	24/07/2023	1091,26%	253,94%
6	25/07/2023	256,44%	29,57%

7	26/07/2023	245,73%	52,24%
9	28/07/2023	1769,92%	80,19%
10	29/07/2023	898,74%	147,70%
11	30/07/2023	179,96%	43,19%
12	31/07/2023	269,93%	94,17%
13	01/08/2023	114,67%	45,88%
14	02/08/2023	213,85%	42,65%
15	03/08/2023	209,70%	59,10%
16	04/08/2023	199,21%	58,51%
17	05/08/2023	132,95%	70,71%
18	06/08/2023	37,96%	72,86%
19	07/08/2023	51,36%	43,54 %

Con base en los resultados presentados en la Tabla 3.1, se concluye que las primeras pruebas de medición de ozono del sensor MQ-131 sobrepasan el 100% de error relativo, lo cual significa que las mediciones difieren notoriamente de las mediciones de la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito. Es importante destacar que, al ser mediciones de cantidades pequeñas, si bien los valores de las diferencias no son muy grandes, se tiene un porcentaje de error alto. Por el contrario, en las últimas mediciones realizadas se ha podido obtener valores de error relativo cercanos y menores al 50%, tal como se observa en la Tabla 3.1, la última medición realizada el 07/08/2023 presenta un error relativo promedio de 51,36%.

Para el caso de las mediciones de dióxido de nitrógeno del sensor MICS-6814, se evidencia que el porcentaje de error relativo promedio es menor al del sensor MQ-131, presentando únicamente en sus pruebas #3, 4, 5 y 10 un promedio mayor al 100%. Mientras que en las demás pruebas realizadas se presentan errores promedio alrededor del 50%, teniendo en su última prueba realizada el 07/08/2023 un error relativo promedio de 43,54%, de este resultado se infiere que el funcionamiento y calibración del sensor MICS-6814 ha sido más eficiente que el del sensor MQ-131.

3.4 Pruebas de funcionamiento de aplicación web

Para verificar el correcto funcionamiento de la aplicación web se debe ingresar al enlace URL de la aplicación web que se encuentra en el ANEXO II, en el cual se deberá registrar una cuenta en caso de no tenerla, caso contrario se deberá iniciar sesión. Adicionalmente a ello, en el ANEXO III se tiene el enlace del proyecto de la aplicación cargado en GitHub.

En la subsección 2.2.1 ya se había registrado una cuenta para almacenar datos desde el dispositivo sensor, por lo cual se iniciará sesión mediante la respectiva cuenta, como se observa en la Figura 3.14.

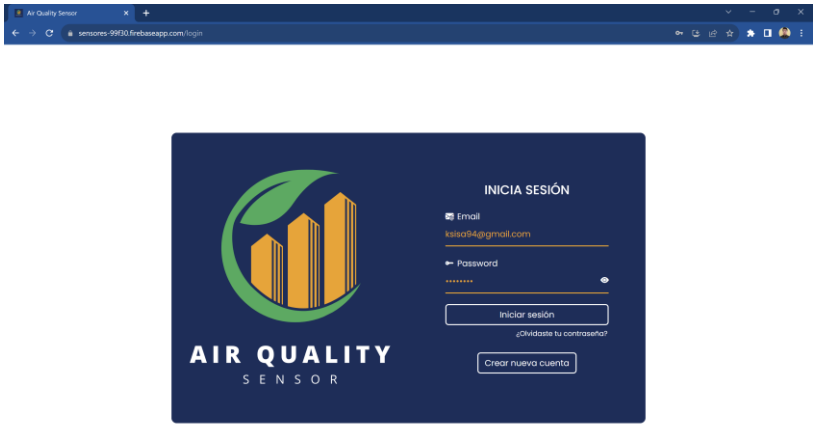


Figura 3.14 Prueba de inicio de sesión en aplicación web

Al iniciar sesión se observa la pantalla principal con la tabla dinámica de mediciones de dióxido de nitrógeno (NO_2), ozono (O_3) y temperatura. Esto demuestra el correcto funcionamiento del Código 2.3, implementado para obtener los documentos registrados en Cloud Firestore, como se evidencia en la Figura 3.15. Además de ello, los datos mostrados en la tabla se van actualizando en tiempo real mientras se almacenan en Cloud Firestore.

Figura 3.15 Prueba de obtención de documentos desde Cloud Firestore

Para observar las gráficas de áreas implementadas, se ingresa en el botón *Visualizar gráficas,* de esa manera se evidencia en la Figura 3.16 el resultado de visualización de las gráficas implementadas por cada medición de dióxido de nitrógeno (NO_2), ozono (O_3) y temperatura.

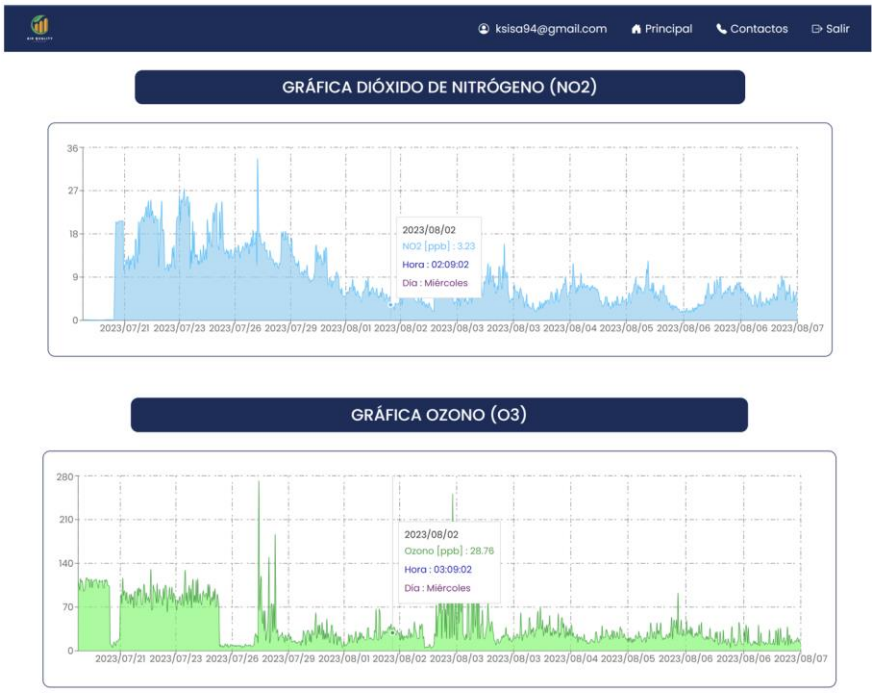


Figura 3.16 Prueba de visualización de gráficas de mediciones

Además, es posible exportar los datos de las respectivas mediciones mediante el botón *Exportar datos,* luego de lo cual aparecerá una ventana flotante donde se debe elegir el intervalo de fechas de las mediciones que se desean exportar, como se muestra en la Figura 3.17.

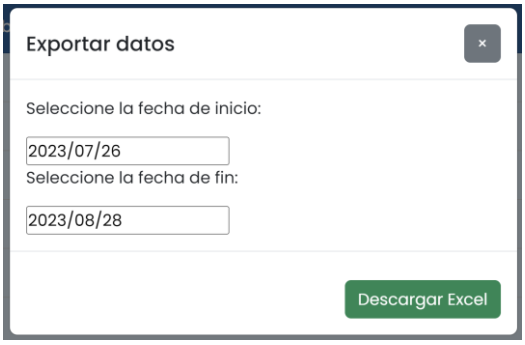
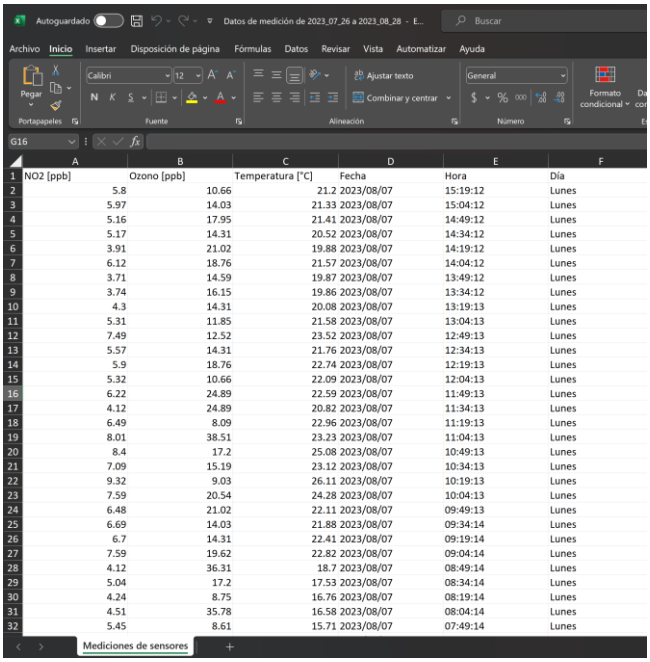


Figura 3.17 Prueba de exportación de datos de mediciones

Como resultado se presenta en la Figura 3.18 el archivo de extensión XLSX, descargado desde la aplicación web, con las mediciones de dióxido de nitrógeno (NO_2), ozono (O_3) y temperatura con sus respectivas fecha, día y hora de medición.



	A	B	C	D	E	F
1	NO2 [ppb]	Ozono [ppb]	Temperatura [°C]	Fecha	Hora	Día
2	5.8	10.66	21.2	2023/08/07	15:19:12	Lunes
3	5.97	14.03	21.33	2023/08/07	15:04:12	Lunes
4	5.16	17.95	21.41	2023/08/07	14:49:12	Lunes
5	5.17	14.31	20.52	2023/08/07	14:34:12	Lunes
6	3.91	21.02	19.88	2023/08/07	14:19:12	Lunes
7	6.12	18.76	21.57	2023/08/07	14:04:12	Lunes
8	3.71	14.59	19.87	2023/08/07	13:49:12	Lunes
9	3.74	16.15	19.86	2023/08/07	13:34:12	Lunes
10	4.3	14.31	20.08	2023/08/07	13:19:13	Lunes
11	5.31	11.85	21.58	2023/08/07	13:04:13	Lunes
12	7.49	12.52	23.52	2023/08/07	12:49:13	Lunes
13	5.57	14.31	21.76	2023/08/07	12:34:13	Lunes
14	5.9	18.76	22.74	2023/08/07	12:19:13	Lunes
15	5.32	10.66	22.09	2023/08/07	12:04:13	Lunes
16	6.22	24.89	22.59	2023/08/07	11:49:13	Lunes
17	4.12	24.89	20.82	2023/08/07	11:34:13	Lunes
18	6.49	8.09	22.96	2023/08/07	11:19:13	Lunes
19	8.01	38.51	23.23	2023/08/07	11:04:13	Lunes
20	8.4	17.2	25.08	2023/08/07	10:49:13	Lunes
21	7.09	15.19	23.12	2023/08/07	10:34:13	Lunes
22	9.32	9.03	26.11	2023/08/07	10:19:13	Lunes
23	7.59	20.54	24.28	2023/08/07	10:04:13	Lunes
24	6.48	21.02	22.11	2023/08/07	09:49:13	Lunes
25	6.69	14.03	21.88	2023/08/07	09:34:14	Lunes
26	6.7	14.31	22.41	2023/08/07	09:19:14	Lunes
27	7.59	19.62	22.82	2023/08/07	09:04:14	Lunes
28	4.12	36.31	18.7	2023/08/07	08:49:14	Lunes
29	5.04	17.2	17.53	2023/08/07	08:34:14	Lunes
30	4.24	8.75	16.76	2023/08/07	08:19:14	Lunes
31	4.51	35.78	16.58	2023/08/07	08:04:14	Lunes
32	5.45	8.61	15.71	2023/08/07	07:49:14	Lunes

Figura 3.18 Resultado de archivo XLSX exportado

Esto evidencia el correcto funcionamiento de la aplicación web, de acuerdo con los objetivos planteados y los requerimientos del sistema implementado.

3.5 Actualización del tablero Kanban

Como verificación final del registro de actividades del presente trabajo de integración curricular se realiza la última actualización de actividades en el tablero Kanban, tal como se observa en la Figura 3.19, en la cual se ve que todas las actividades han sido terminadas.

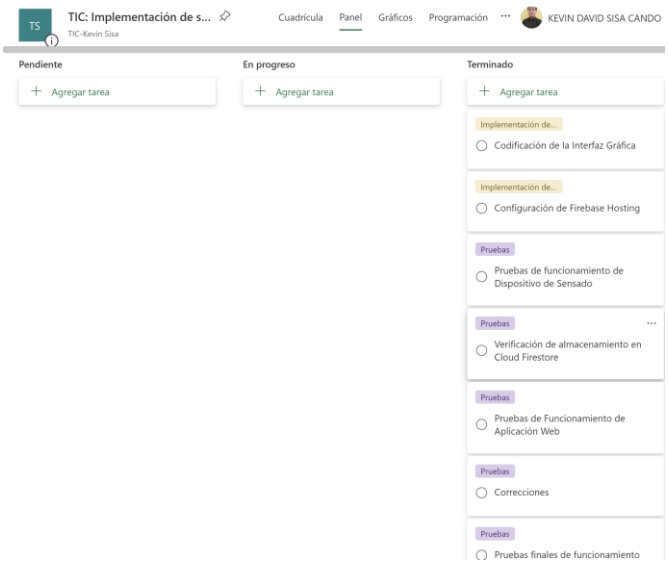


Figura 3.19 Actualización final del tablero Kanban

3.6 Conclusiones

- El presente trabajo de integración curricular ha permitido el desarrollo de un prototipo de dispositivo sensor, el cual permitió obtener las concentraciones de ozono (O_3) y dióxido de nitrógeno (NO_2) mediante la utilización de sensores de bajo costo, conjuntamente se ha desarrollado el prototipo de aplicación web para la adecuada visualización y exportación de los datos almacenados.

- Los sensores basados en electrquímica MQ-131 y MICS-6814 han permitido la obtención de concentración de ozono (O_3) y dióxido de nitrógeno (NO_2) en la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito ubicada en el Centro Histórico de Quito. De lo cual se concluye, con base en los resultados, que usar los sensores de bajo costo sirven como referencia de la concentración de gases contaminantes; sin embargo, su funcionamiento no resulta ser óptimo, ya que en ciertos intervalos de tiempo ambos sensores no lograban seguir la curva de medición del equipo usado por la estación de la Red Metropolitana de Monitoreo Atmosférico de Quito.

- De acuerdo con las pruebas de funcionamiento y los errores relativos promedio registrados por día se concluye que la calibración de los sensores es primordial para tener resultados más confiables, en el último día de medición de ozono se registró un error relativo promedio de 51,36% y un error relativo promedio de dióxido de nitrógeno de 43,54%.

- Se ha podido comprobar el registro en Cloud Firestore de las mediciones de concentraciones de ozono (O_3) y dióxido de nitrógeno (NO_2) realizadas con el dispositivo sensor, mediante el uso de la placa de desarrollo ESP32 y su capacidad de comunicación WiFi, estos datos también han podido ser visualizados mediante la aplicación web, usando consultas personalizadas hacia la base de datos.

- La utilización de la placa de desarrollo ESP32 de 38 pines ha permitido ejecutar tareas complejas, teniendo la capacidad de procesar datos en tiempo real, gracias a que está basada en un procesador de 32 bits.

- Para un funcionamiento óptimo del dispositivo sensor se ha implementado la librería WiFiManager, la cual ha permitido crear una red WiFi temporal para que el usuario configure la red a la cual el dispositivo sensor debe conectarse, de esa manera se mejora la eficiencia y se evita que el usuario tenga que cargar un nuevo sketch con configuraciones de otra red WiFi, cuando el prototipo sea trasladado a otro lugar.

- Para el funcionamiento de la aplicación web se ha utilizado la biblioteca React.js la cual ha permitido construir las interfaces de la aplicación de manera dinámica y que los componentes sean reutilizados, además de sus complementos que han sido beneficiosos para el desarrollo y ejecución de la aplicación web.
- El uso de Firebase como plataforma de servicio de base de datos y de host ha permitido que el dispositivo sensor almacene los datos en Cloud Firestore y adicional a ello, el despliegue de la aplicación web haciendo uso de Firebase Hosting. Así esta plataforma presenta beneficios adecuados y orientados para desarrollar aplicaciones web.
- El servicio de Firebase Hosting de la plataforma Firebase ha permitido el proceso de alojamiento de la aplicación web, otorgando beneficios de simplificación en su implementación, además de ofrecer certificados SLL de forma gratuita y habilitando HTTPS para el dominio creado.

3.7 Recomendaciones

- En relación con el trabajo de integración curricular desarrollado se recomienda investigar otras opciones de sensores de bajo costo para comparar el desempeño de otros proveedores, dado que los implementados brindaron una curva de medición que, a ciertos intervalos de tiempo, presentaba resultados no tan exactos, en comparación con los valores medidos por la Red Metropolitana de Monitoreo Atmosférico de Quito.
- Con base en los resultados obtenidos se recomienda que las pruebas de medición en una localidad específica se las realice durante más días, dado que en el presente trabajo se las realizaron a partir del 20/07/2023 hasta el 07/08/2023 por problemas en la disponibilidad de los sensores, de esa forma se podrá tener más resultados de medición para comparar la variabilidad de las mediciones.
- Se recomienda añadir un sensor de geolocalización al dispositivo para mostrar la ubicación del dispositivo desde la aplicación web, así se tendrá un mejor control y conocimiento del lugar en donde está ubicado el dispositivo.
- Se recomienda añadir una pantalla OLED al dispositivo, para visualizar las mediciones en tiempo real en el propio dispositivo, sin necesidad de esperar el tiempo de 15 minutos para la visualización de cada medición en la aplicación web.

- Se recomienda que, a partir de los datos exportados en Excel, se incluya el uso de Power BI para crear *dashboards* con las mediciones de los sensores, y de esa manera se obtengan gráficas más útiles para el análisis y toma de decisiones.
- Se recomienda que al iniciar las pruebas de medición se las haga a través de la consola de Arduino IDE para que se valide el correcto funcionamiento de los sensores, posterior a ello se codifique para el envío a Cloud Firestore.
- Para un mejor acceso a la información de las mediciones de concentración de ozono (O_3) y dióxido de nitrógeno (NO_2) resulta recomendable desarrollar un prototipo de aplicación móvil la cual esté disponible para que los usuarios la instalen en sus dispositivos móviles.
- Como complemento, se recomienda implementar mensajes de alerta en la aplicación web de acuerdo con el nivel de concentración de ozono (O_3) y dióxido de nitrógeno (NO_2), en caso de que estos niveles se incrementen a un valor crítico.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] Secretaría de Ambiente del Municipio del Distrito Metropolitano de Quito, “NORMA ECUATORIANA DE CALIDAD DEL AIRE,” Quito, Jun. 2011.

[2] Teleamazonas, “¿Cuáles son las ciudades de Ecuador con más contaminación del aire?,” 2022. https://www.teleamazonas.com/ciudades-ecuador-contaminacion-aire-oms/ (accessed Mar. 22, 2023).

[3] H. P. L. De Medeiros and G. Girao, “An IoT-based Air Quality Monitoring Platform,” in *2020 IEEE International Smart Cities Conference, ISC2 2020*, Institute of Electrical and Electronics Engineers Inc., Sep. 2020. doi: 10.1109/ISC251055.2020.9239070.

[4] Techonicsltd, “MICS-6814 (Air Quality Co, NH3, Nitrogen Oxide Gas Sensor),” 2018. https://www.techonicsltd.com/product/mics-6814-air-quality-co-nh3-nitrogen-oxide-gas-sensor/ (accessed Apr. 08, 2023).

[5] SGX SensorTech, “Data Sheet MiCS-6814 1143 rev 8.” [Online]. Available: www.sgxsensortech.com

[6] L. Zhengzhou Winsen Electronics Technology Co., “Ozone Gas Sensor （Model： MQ131 Low Concentration） Manual,” May 2014. [Online]. Available: www.winsen-sensor.com

[7] TresD Print Tech, “MQ-131 Sensor De Gas Ozono Baja Concentracion MQ XL V3.0.” https://tresdprinttech.com/en/sensores/613-sensor-de-ozono-mq131-mq-xl-v30-7503040290385.html (accessed Jul. 21, 2023).

[8] Components101, “GY-BMP280 Module,” Sep. 12, 2018. https://components101.com/sensors/gy-bmp280-module (accessed Jul. 21, 2023).

[9] UNIT Electronics, “ESP32 38 Pines ESP WROOM 32.” https://uelectronics.com/producto/esp32-38-pines-esp-wroom-32/ (accessed Jul. 22, 2023).

[10] Google Developers, “Cloud Firestore,” Nov. 14, 2022. https://firebase.google.com/docs/firestore?hl=es-419 (accessed Apr. 09, 2023).

[11] Google Developers, “Firebase Hosting,” Aug. 28, 2022. https://firebase.google.com/docs/hosting?hl=es-419 (accessed Apr. 09, 2023).

[12] Christina Kopecky, “What is Node.js? A beginner’s introduction to JavaScript runtime,” Sep. 30, 2020. https://www.educative.io/blog/what-is-nodejs (accessed Apr. 09, 2023).

[13] Quincy Larson, “What Exactly is Node.js? A Guide for Beginners,” May 25, 2023. https://www.freecodecamp.org/news/what-exactly-is-node-guide-for-beginners/ (accessed Jun. 03, 2023).

[14] W3Schools, “What is React?” https://www.w3schools.com/whatis/whatis_react.asp (accessed Jun. 03, 2023).

[15] David Herbert, “What is React.js? (Uses, Examples, & More),” Jun. 27, 2023. https://blog.hubspot.com/website/react-js (accessed Jun. 03, 2023).

[16] Aprendiendo Arduino, “Etiqueta: IDE,” Jan. 27, 2018. https://www.aprendiendoarduino.com/tag/ide/ (accessed Jun. 03, 2023).

[17] CEAC, “Qué es el IDE de Arduino en robótica,” Dec. 10, 2018. https://www.ceac.es/blog/que-es-el-ide-de-arduino-en-robotica (accessed Jun. 03, 2023).

[18] Microsoft, “Why did we build Visual Studio Code?,” Mar. 30, 2023. https://code.visualstudio.com/Docs/editor/whyvscode (accessed Apr. 09, 2023).

[19] Priya Pedamkar, “What is Visual Studio Code?” https://www.educba.com/what-is-visual-studio-code/ (accessed Jun. 03, 2023).

[20] Martin Heller, “What is Visual Studio Code? Microsoft’s extensible code editor,” Jul. 08, 2022. https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html (accessed Jun. 03, 2023).

[21] Laia Gilibets, “What is the Kanban methodology and how to use it,” Jan. 12, 2023. https://www.iebschool.com/blog/metodologia-kanban-agile-scrum/ (accessed Jun. 23, 2023).

5 ANEXOS

ANEXO I. Enlace a sketch de Arduino IDE

https://n9.cl/ls3yw

ANEXO II. Enlace de la aplicación web

El siguiente enlace corresponde a la aplicación web implementada:

https://sensores-99f30.firebaseapp.com/login

ANEXO III. Enlace de proyecto en GitHub

El siguiente enlace corresponde al proyecto de React.js con la aplicación web implementada:

https://github.com/kevindavid17/TIC-Kevin-Sisa.git

ANEXO IV. Enlace de manual de usuario

El siguiente enlace corresponde al manual de usuario:

https://n9.cl/0d1o9