UNIVERSIDAD PRIVADA DE TACNA
FACULTAD DE ÍNGENIERIA

ESCUELA PROFESIONAL DE ÍNGENIERIA ELECTRÓNICA

TESIS

“DISEÑO DE UN SISTEMA IOT PARA EL MONITOREO REMOTO DE LA CALIDAD DEL AIRE EN AMBIENTES INTERIORES DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD PRIVADA DE TACNA”

PARA OPTAR:

TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

PRESENTADO POR:

Bach. RUDY PABLO MENDOZA MERMA

Bach. VÍCTOR PABLO ALEJANDRO CORDERO OCHOA

TACNA- PERÚ

2022

UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

"DISEÑO DE UN SISTEMA IoT PARA EL MONITOREO REMOTO DE LA CALIDAD DEL AIRE EN AMBIENTES INTERIORES DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD PRIVADA DE TACNA"

Tesis sustentada y aprobada el 09 de diciembre de 2022; estando el jurado calificador integrado por:

PRESIDENTE	**:**	**Mag. JOSÉ MARCIAL SUMARRIVA BUSTINZA**
SECRETARIO	**:**	**Mtro. MARKO JESÚS POLO CAMACHO**
VOCAL	**:**	**Mtra. MARÍA ELENA VILDOZO ZAMBRANO**
ASESOR	**:**	**Mag. HUGO JAVIER RIVERA HERRERA**

DECLARACIÓN JURADA DE ORIGINALIDAD

Nosotros Bach. Rudy Pablo Mendoza Merma y Bach. Víctor Pablo Alejandro Cordero Ochoa, en calidad de: bachilleres de la Escuela Profesional de Ingeniería Electrónica de la Facultad de Ingeniería de la Universidad Privada de Tacna, identificado con DNI 76150208 y 43071900 respectivamente. Declaramos bajo juramento que:

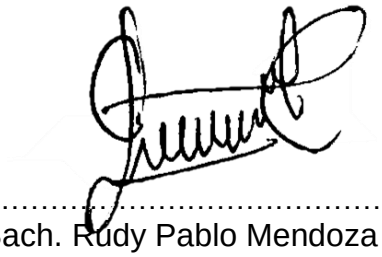
1. Somos autores de la tesis titulada: "*Diseño de un sistema IoT para el monitoreo remoto de la calidad del aire en ambientes interiores de la Facultad de Ingeniería de la Universidad Privada de Tacna"* la misma que presentamos para optar el Título Profesional de *Ingeniero Electrónico*.
2. La tesis no ha sido plagiada ni total ni parcialmente, habiéndose respetado las normas internacionales de citas y referencias para las fuentes consultadas.
3. La tesis presentada no atenta contra derechos de terceros.
4. La tesis no ha sido publicada ni presentada anteriormente para obtener algún grado académico previo o título profesional.
5. Los datos presentados en los resultados son reales, no han sido falsificados, ni duplicados, ni copiados.

Por lo expuesto, mediante la presente asumimos frente a *La Universidad* cualquier responsabilidad que pudiera derivarse por la autoría, originalidad y veracidad del contenido de la tesis, así como por los derechos sobre la obra.

En consecuencia, nos hacemos responsables frente a La Universidad y a terceros, de cualquier daño que pudiera ocasionar, por el incumplimiento de lo declarado o que pudiera encontrar como causa del trabajo presentado, asumiendo todas las cargas pecuniarias que pudieran derivarse de ello en favor de terceros con motivo de acciones, reclamaciones o conflictos derivados del incumplimiento de lo declarado o las que encontrasen causa en el contenido de la tesis.

De identificarse fraude, piratería, plagio, falsificación o que la obra haya sido publicada anteriormente; asumimos las consecuencias y sanciones que de nuestras acciones se deriven, sometiéndonos a la normatividad vigente de la Universidad Privada de Tacna.

Tacna 25 de octubre del 2022



……
Bach. Rudy Pablo Mendoza Merma

DNI: 76150208

……
Bach. Víctor Pablo Alejandro Cordero Ochoa

DNI: 43071900

DEDICATORIA

A mis padres Pablo y Benilda por su amor, apoyo, sacrificio y las ganas de no rendirse jamás.

A mi hijo Pablito por ser el motor y motivo de querer salir adelante sin importar los obstáculos.

Rudy Pablo Mendoza Merma

A Dios y a mi familia que estuvieron a mi lado en cada momento importante de mi vida.

Víctor Pablo Alejandro Cordero Ochoa

AGRADECIMIENTO

A nuestro asesor Mag. Hugo Javier Rivera Herrera por su apoyo y colaboración durante la realización de esta tesis.

A los docentes y demás integrantes de la Escuela Profesional de Ingeniería Electrónica de la Facultad de Ingeniería de la Universidad Privada de Tacna, quienes fueron participes importantes durante nuestra etapa de formación profesional a lo largo de nuestra estadía en la universidad.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

Este proyecto tiene como objetivo diseñar un sistema IoT para el monitoreo remoto de la calidad del aire en ambientes interiores de la Facultad de Ingeniería de la Universidad Privada de Tacna. El proyecto incluye el análisis de la topología tipo Malla y su respectiva implementación, tanto en software como hardware y captura de los valores de parámetros a través de sensores de calidad de aire. Además de esto, se hace una integración de la red Malla con una red WLAN para la transmisión hacia una base de datos. El proyecto incluye el diseño y modelado de una interfaz gráfica para poder mostrar la data al usuario final de una forma gráfica e intuitiva usando el framework .NET (C#) y WinForms. Como resultado se tuvo un buen rendimiento en la red Malla con valores óptimos de RSSI (> -60 dBm) y de velocidad de conexión. También se observó que la estructura de base de datos era estable y liviana capturando cerca de 54000 registros, ocupando tan solo 80 MB de espacio de memoria. Por ultimo los registros se mostraron en la interfaz gráfica dando a conocer que los ambientes probados están dentro de los estándares de calidad de ambiente (ECA) del aire máximos permitidos. También se capturo parámetros meteorológicos como temperatura, humedad y presión atmosférica las cuales fueron contrastados con los datos locales en IQAir.

Palabras claves: IoT, Topología Malla, desarrollo de software, sensores, conexiones inalámbricas.

ABSTRACT

This project aims to design an IoT system for remote monitoring of air quality in indoor environments of the Faculty of Engineering of the Universidad Privada de Tacna. The project includes the analysis of the Mesh type topology and its respective implementation, both in software and hardware and capture of parameter values through air quality sensors. In addition to this, an integration of the mesh network with a WLAN network for transmission to a database is made. The project includes the design and modeling of a graphical interface to display the data to the end user in a graphical and intuitive way using the .NET framework (C#) and WinForms. As a result there was a good performance in the Mesh network with optimal values of RSSI (> -60 dBm) and connection speed. It was also observed that the database structure was stable and lightweight, capturing about 54000 records, occupying only 80 MB of memory space. Finally, the records were displayed on the graphical interface showing that the tested environments are within the maximum allowable ambient air quality (ECA) standards. Meteorological parameters such as temperature, humidity and atmospheric pressure were also captured and compared with the local data in IQAir.

Keywords: IoT, Mesh Topology, software development, sensors, wireless connections.

INTRODUCCIÓN

El modelo de calidad la población mundial del aire de la Organización Mundial de la Salud (OMS) confirma que el 92% de vive en lugares donde los niveles de calidad del aire exceden los límites fijados en las directrices de la OMS sobre la calidad del aire ambiente para una media anual de partículas con un diámetro inferior a 2,5 micrómetros (PM2,5). Los límites establecidos en las directrices de la OMS respecto de la media anual de PM2,5 son 10 $\mu g/m^3$.

Es además uno de los factores de salud ambiental que tiene una mayor contribución a la carga de enfermedad según el informe “Medio ambiente y salud” de la Agencia Europea de Medio ambiente, las infecciones agudas del tracto respiratorio inferior atribuibles a la contaminación del aire interior explican el 4,6% de todas las muertes y el 3,1% de AVAD (años de vida ajustados por discapacidad). Como muestra de la creciente preocupación acerca de los efectos que sobre la salud tiene estos factores, los organismos internacionales como la Organización Mundial de la Salud, la Comisión Europea, y los nacionales, tienen ya legislación, informes y estudios relacionados con la contaminación del aire exterior, y en algunos casos incluyen también apartados específicos y menciones al aire interior a falta de un mayor desarrollo legislativo específico de la materia.

De acuerdo con las referencias entregadas por la OMS y la Agencia europea de Medio Ambiente, da por relevancia evaluar la calidad del aire, a través de Niveles de CO2, temperatura y humedad al interior de la sala de clases de la Escuela de Ingeniería Electrónica de la Universidad Privada de Tacna, puesto que esto es un factor que interfiere directamente en la salud, concentración y atención de los estudiantes en sus aulas.

CAPÍTULO I: EL PROBLEMA DE INVESTIGACIÓN

1.1. Descripción Del Problema

Estudios realizados por la organización Mundial de la Salud, demuestran que las personas pasamos entre un 75 a 90 porciento de nuestras vidas en lugares interiores, lo cual da real importancia, del porque es importante el tener una buena calidad del aire interior, además que nos dice que este aire se encuentra entre 3 y 7 veces más viciado que el aire exterior. En los últimos años ha cobrado especial relevancia al asociarse al término “síndrome del edificio enfermo” que comprende un amplio rango de síntomas o enfermedades que las personas que trabajan o habitan en dicho edificio atribuyen al edificio en sí. Es por ello por lo que, cuidando la calidad del aire o ambiente interior, se cuida de la salud de las personas que viven o trabajan, en definitiva, que pasan un tiempo considerable en el interior de dicho edificio.

Es además uno de los factores de salud ambiental que tiene una mayor contribución a la carga de enfermedad según el informe “Medio ambiente y salud” de la Agencia Europea de Medio ambiente, las infecciones agudas del tracto respiratorio inferior atribuibles a la contaminación del aire interior explican el 4,6% de todas las muertes y el 3,1% de AVAD (años de vida ajustados por discapacidad). Como muestra de la creciente preocupación acerca de los efectos que sobre la salud tiene estos factores, los organismos internacionales como la Organización Mundial de la Salud, la Comisión Europea, y los nacionales, tienen ya legislación, informes y estudios relacionados con la contaminación del aire exterior, y en algunos casos incluyen también apartados específicos y menciones al aire interior a falta de un mayor desarrollo legislativo específico de la materia.

Uno de los aspectos más importantes es salvaguardar la salud del personal dentro de las oficinas, centros de trabajo y de estudio, por lo que la calidad del aire dentro de ellas llega a ser algo de gran importancia (Ingeniería TV, 2021).

Según Guzmán, el aire es fundamental para vivir, es muy importante ya que nutre de oxígeno a los pulmones, a la sangre y en consecuencia a todo el cuerpo. La respiración es un acto mecánico que la mayoría de las veces pasa inadvertida, pero es fundamental para realizar todas nuestras funciones vitales (Guzmán, 2021).

Según el MINAM una de las principales fuentes de gases es por el uso de vehículos motorizados ocupando, en el 2016, el tercer lugar a nivel nacional en tener más vehículos cada 1000 habitantes con un indicador de 142.72 (MINAM, 2016).

Es por eso por lo que en este trabajo presentamos una forma de monitorear la calidad del aire de ambientes de interiores de una forma segura, remota y en tiempo real usando la tecnología IOT en un ambiente multiplataforma.

1.2. Formulación del problema

¿Con un sistema IoT se logrará monitorear remotamente la calidad del aire en ambientes interiores de la Facultad de Ingeniería de la Universidad Privada de Tacna?

1.3. Justificación e importancia de la investigación

Las principales fuentes de emisión de gas contaminante, según el MINAM, en la ciudad de Tacna son las ladrilleras y el parque automotor (vehículos) (MINAM, 2017). Existen leyes que normalizan la cantidad de gases contaminantes emitidos por dichas fuentes o también normas técnicas como la EM.030 la cual tiene por objeto establecer los lineamientos técnicos mínimos que se deben considerar para el diseño, construcción, instalación y operación de los sistemas de ventilación mecánica en una edificación, con la finalidad de obtener niveles adecuados de calidad y cantidad del aire en las edificaciones, a fin de garantizar la seguridad así como, la salubridad e higiene de las personas (El Peruano, 2020). A pesar de las normas existentes ya nombradas, el monitoreo de la calidad del aire en ambientes cerrados es una práctica muy poco común o que simplemente no se hace, pero la cual tiene una gran importancia.

Son innumerables los beneficios que tiene respirar aire puro entre los cuales está, por ejemplo, el ayudar a disminuir la tensión y el estrés lo cual genera una sensación de bienestar total. También se ha demostrado que respirar aire puro por lo menos 30 minutos al día ayuda a reducir los trastornos cardiacos y los cuadros depresivos, también se ha demostrado un beneficio en la estimulación del sistema inmunológico que disminuye trastornos alérgicos y afecciones respiratorias. Otro gran beneficio es que reequilibra el funcionamiento óptimo de todos los órganos de nuestro cuerpo gracias a la oxigenación celular.

Debido a la coyuntura que estamos viviendo se necesita una solución de bajo costo y accesible para cualquier tipo de locación, que a la vez sea eficiente y de confianza. Es por eso que se plantea la solución de un sistema de monitoreo de control de aire para medir diferentes tipos de variables como la cantidad de gases contaminantes (CO, CO2, NH3, NOx), Temperatura y humedad; la cual se puedan ser

almacenadas en una base de datos y observadas en distintas plataformas como son Web, Android y Aplicación de Escritorio (Windows).

1.4. Objetivos

1.4.1. Objetivo General

Diseñar un sistema IoT para el monitoreo remoto de la calidad del aire en ambientes interiores de la Facultad de Ingeniería de la Universidad Privada de Tacna.

1.4.2. Objetivos Específicos

a. Seleccionar los dispositivos electrónicos requeridos para el diseño.
b. Identificar los parámetros para el control de calidad del aire en ambientes interiores.
c. Implementar un prototipo para comprobar el funcionamiento del diseño propuesto.

1.5. Hipótesis

El diseño de un sistema IoT permite monitorear remotamente la calidad del aire en ambientes interiores de la Facultad de Ingeniería de la Universidad Privada de Tacna.

CAPÍTULO II: MARCO TEÓRICO

2.1. Antecedentes Del Estudio

2.1.1. Antecedentes internacionales

Gahona y Gavilema (2020), en su tesis *"Diseño de la red internet de las cosas (IoT) para el edificio de la empresa Consel"* diseñaron la red IoT en el edificio de la empresa Consel para que satisfaga las necesidades tecnológicas y de seguridad de los residentes luego de identificar los requerimientos de la red IoT de los residentes en el edificio de la empresa Consel para la determinación de los aspectos técnicos en cuanto a cobertura. El autor concluyo que la calidad de servicio basada en la espera equitativa ponderada (WFQ), cuenta con las etiquetas para priorizar el tráfico en la red, partiendo de datos de red IoT, servicios ftp, streaming y dejando de último a datos de terminales no IoT, permitiendo que la pérdida de paquetes sea mejorada en 57,01 %.

Castro (2017), en su tesis *"Cifrado simétrico de datos en la comunicación de sistemas embebidos para su uso en el internet de las cosas"* logró Implementar un modelo de cifrado simétrico de datos eficiente y simple mejorando así la comunicación de los sistemas embebidos aplicado en el internet de las cosas (IoT). El autor concluyo que procedimiento del diseño del modelo de cifrado simétrico propuesta evidencia que ocupa 7664 bytes de almacenamiento del programa siendo el 23% de un máximo de 32256 bytes, consume unos 826 bytes siendo el 40% de 2048 bytes de memoria RAM en el Arduino, en relación con el algoritmo de cifrado simétrico estándar AES. Por lo tanto, demuestro que el modelo criptográfico simétrico propuesto es eficiente en cuanto a utilizar menor espacio de memoria en los sistemas embebidos ya que cumple uno de los parámetros importantes para su medición

Morales (2018), en su tesis *"Evaluación de la calidad del aire interior en salas de clases en la UTFSM, sede concepción"* evaluó la Calidad del aire interior de una sala de clases en la UTFSM, describió y los potenciales efectos en los usuarios de esta, Al seleccionar y caracterizar un salón de clases para evaluando la calidad de aire interior, también al determinar la renovación de aire por hora en el salón y Determino los niveles de dióxido de carbono (CO_2), humedad relativa, temperatura, nivel de saturación de oxígeno y comparando sus valores con estándares nacionales e internacionales. El autor concluyo que el rango de saturación está bajo los parámetros normales establecidos, esto se refleja en los gráficos de saturación, lo cual tiene una directa relación con el grafico de CO_2, donde puede ver tres periodos de diferentes medidas

sin y con ocupantes, para la cual solo analizo y comparo las mediciones con ocupantes, nuevamente recomendó que los 3 extractores, deben estar en funcionamiento, lo cual permitirá que los niveles de CO2 disminuyan y con esto el aire este menos viciado y la oxigenación de la sangre en los ocupantes este dentro de los parámetros normales.

2.1.2. Antecedentes Nacionales

Corpus y Portugal (2021) en su tesis *"Diseño de una red inalámbrica para el servicio de pesaje en el sector agroindustrial utilizando Balanzas Super SS mediante protocolo SPI basado en microcontrolador ESP32"* diseñó una red inalámbrica para el servicio de pesaje para el sector agroindustrial utilizando balanzas SUPER SS mediante protocolo SPI basado en microcontrolador ESP32, para visualizar una mejora y optimización en el proceso de productividad; en los productos alimenticios. Como resultado de la investigación, este sistema logró disminuir los tiempos excesivos en el proceso de conteo para los pesajes que se realizaban de manera manual, reduce los errores humanos al momento del ingreso de los datos obtenidos en el pesaje y facilita tener la información en tiempo real enviando los datos obtenidos al SQL server.

Murata (2020), en su tesis *"Diseño de un sistema de monitoreo de contaminación acústica urbana bajo una plataforma IoT"* expone el diseño de un sistema ciber físico de monitoreo de contaminación acústica urbana, el mismo que, combinando radiofrecuencia y conexión a internet a través de wifi, y tiene como principal objetivo el informar y advertir a los ciudadanos y entidades gubernamentales cuando los niveles de ruido presentes pueden afectar la salud. Finalmente se presentó una solución novedosa de bajo consumo energético y computacional capaz de cubrir grandes zonas de la ciudad y enviar la data a una plataforma Cloud sin necesidad de depender de redes WIFI, la cual podrá ser implementada con el objetivo de monitorear y controlar este tipo de contaminación de forma más eficaz.

(Cahuantico, 2019), en su tesis *"Evaluación de contaminantes atmosféricos CO, SO2, PM10, PM2.5 de la zona urbana cusco 2017"* encontró que con respecto al PM2.5 en Huanchaq se obtuvo el valor de 73,401 µg/m^3, San Sebastián 51,928 µg /m^3 y San Jerónimo 137,134 µg/m^3, sobrepasaron el ECA MINAM (50 µg/m^3 en 24 horas) y ECA OMS (25 µg/m^3); El PM10 de Huanchaq se obtuvo el valor de 57,32 µg/m^3, San Sebastián 54,443 µg/m^3, San Jerónimo 95,027 µg/m^3, los tres no sobrepasaron el ECA MINAM (100 µg/m^3), pero si el ECA OMS (50 µg/m^3); el gas de CO en Huanchaq es 6201,555 µg/m^3, San Sebastián 4379,806 µg/m^3 y San Jerónimo 5095,975 µg/m^3, los tres no sobrepasaron el ECA MINAM (10000 µg/m^3) ni ECA OMS (10285 µg/m^3); el

SO2, en Huanchaq es 2,705 µg/m^3, San Sebastián 3.848 µg/m^3, San Jerónimo 2,612 µg/m^3, están por debajo del ECA MINAM (20 µg/m^3) y ECA OMS (18,753 µg/m^3).

2.2. Bases Teóricas

2.2.1. Internet de las Cosas (IoT)

El Internet de las Cosas es una tendencia de la tecnología el cual representa el mañana de la informática y las telecomunicaciones, su desarrollo está sujeto a la evolución e innovación de distintos campos importantes, desde sensores inalámbricos hasta la nanotecnología e Inteligencia Artificial. Es un componente tecnológico fundamental sobre el que sienta sus bases muchas áreas del mundo incluido el paradigma de la Industria 4.0 (Cruz, y otros, 2015, pág. 11).

En definición es una red de objetos físicos conectados a través de internet, que logran interactuar entre ellos a través de sistemas embebidos, redes de comunicación, mecanismos de computación de respaldo y aplicaciones típicamente en la nube. Permite a los objetos comunicarse entre si, acceder a la información de internet; capturar, almacenar y recuperar datos e interactuar con usuarios humanos, así como con otros sistemas y aplicaciones, creando ambientes cada vez más conectados e inteligentes (Quiñones, 2019).

Para que un sistema IoT funcione necesitamos de dispositivos configurados para que sean compatible con dicha tecnología. Estos dispositivos denominados “Dispositivos IoT” serán nuestros ojos y oídos cuando físicamente estemos ausentes los cuales tendrán la función de capturar cualquier dato que estén programados para almacenar. También se requiere un sistema Backend que se encargue de almacenar dichos datos, ya sea en un servidor local o en la nube. En la figura uno se muestran las 4 etapas claves de los dispositivos IoT:

a. **Capture los datos:** A través de sensores, los dispositivos de IoT capturan datos de sus entornos. Esto podría ser algo tan simple como la temperatura o tan complejo como una retroalimentación de video en tiempo real.
b. **Comparta los datos:** Mediante conexiones de red disponibles, los dispositivos IoT envían estos datos a un sistema en la nube público o

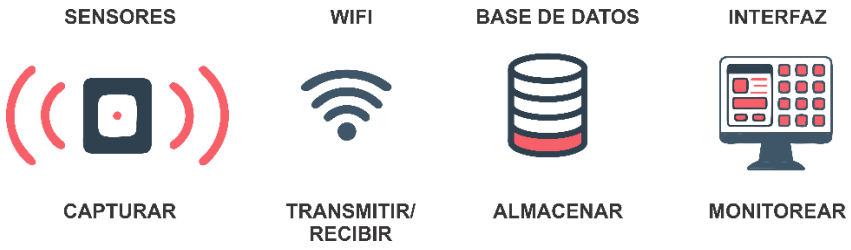
privado (dispositivo-sistema-dispositivo) o a otro dispositivo (dispositivo-dispositivo).

c. **Almacene y procese los datos**: En este punto, el software se programa para almacenar estos datos en base de datos, para luego ser procesadas.

d. **Actúe a partir de los datos:** Este procesamiento de datos se muestra en una interfaz gráfica de usuario, para que con los datos obtenidos se procesa a realizar alguna acción ya sea correctiva, preventiva o de simple monitoreo.

Figura 1

Etapas claves en el proceso de un sistema IoT



2.2.1.1. Captación de Datos por Sensores

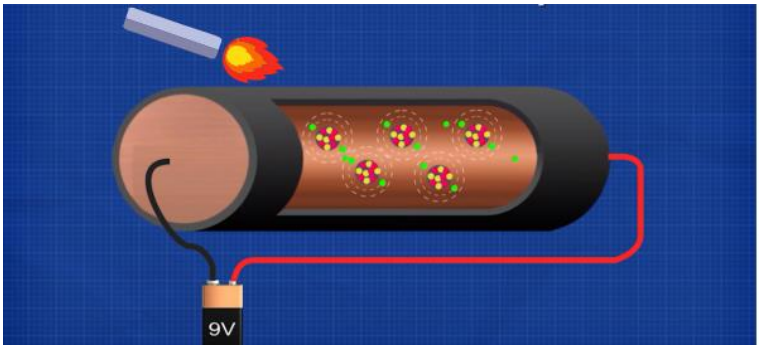
La representación de un parámetro de algún fenómeno físico, mediante el cual, a su vez, describa su comportamiento, se conoce como variable física. Por otra parte, el resultado de cuantificar un atributo físico, asignando valores numéricos a estos a través de una variable o constante física, se conoce como magnitud física (Corona, Abaca, & Mares, 2014).

Los sensores o transductores nos permiten transformar un parámetro de algún fenómeno físico en una señal eléctrica que puede ser entendida y procesada por algún microprocesador. Para captar diferentes fenómenos físicos y transformarlas en una señal, los transductores, necesitan diferentes formas de leerlas; incluso existen diferentes métodos para leer el mismo fenómeno físico.

Un ejemplo es el sensor de temperatura RTD el cual es un sensor de tipo resistivo que consta de un metal que varía su resistencia dependiendo de la temperatura que experimente. Esto ocurre porque a medidas que los átomos y moléculas se excitan, se van a mover mucho, lo que hace más difícil que los electrones libres pasen sin colisionar entre ellos como se muestra en la figura 2.

Figura 2

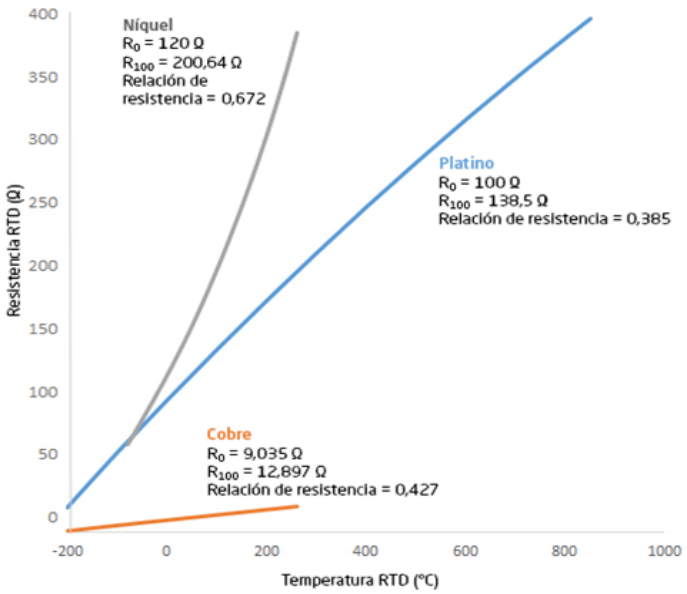
Átomos y electrones excitados debido al calor



Este efecto del calor sobre el metal hace que sea más difícil el paso de electrones aumentando su resistencia eléctrica. Por la Ley de Ohm podemos deducir que, si mantenemos constante la corriente, tendremos un voltaje variable dependiente a la temperatura. El metal más usado es el platino ya que tiene una variación de resistencia casi lineal frente a la variación de temperatura, en comparación con otros metales como se muestra en la figura 3.

Figura 3

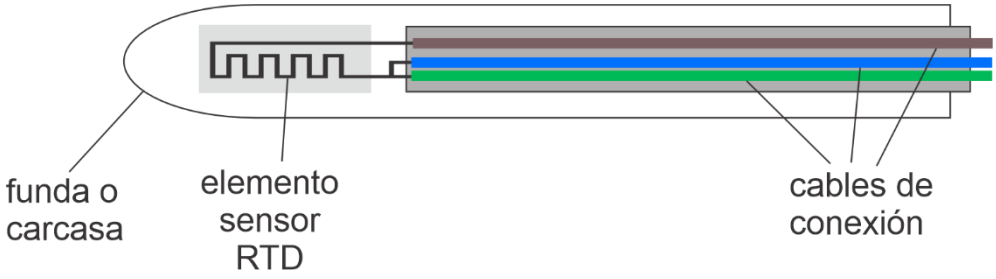
Variación de resistencia de distintos metales respecto a la temperatura.



El RTD se encapsula en un recubrimiento resistente a altas temperaturas como es el cerámico y puede tener 2 o 3 cables de conexión como se muestra en la figura 4.

Figura 4

Recubrimiento y cables de conexión en un RTD de 3 hilos



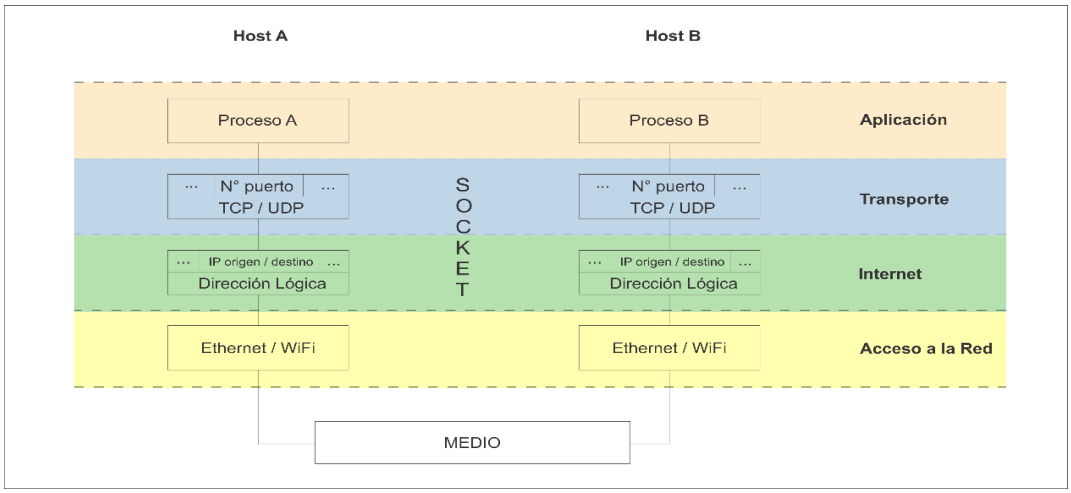
Así como el RTD, existen múltiples transductores y sensores que realizan distintos procedimientos para convertir un parámetro físico en una señal eléctrica.

2.2.1.2. Socket de Internet

Roffé (2009) argumenta que la interfaz Socket es una API para redes TCP/IP que se compone de funciones o rutinas. Las aplicaciones desarrolladas en Sockets están basadas en la arquitectura de Cliente-Servidor que permiten comunicaciones orientadas a la conexión o sin conexión. Para que se dé la comunicación es una red, el programa requiere un socket en cada extremo del proceso el cual consta con una dirección IP y un puerto TCP el cual puede ser programado en diversos tipos de lenguaje de programación. En la figura 5 se muestra el socket de internet y su ubicación en el modelo TCP/IP.

Figura 5

Diagrama de un Socket dentro del modelo TCP/IP.



2.2.1.3. Topología de red IoT

Recabando opiniones aportadas por expertos en redes IoT se observa, en la figura 6, que existen principalmente 3 modelos topológicos básicos para IoT: el modelo Estrella, Peer-To-Peer y Mesh o Malla (Sillero, y otros, 2018).

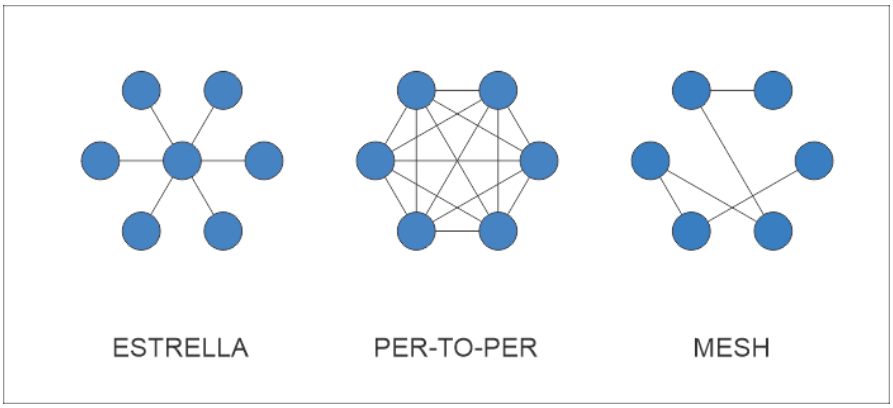
La topología estrella está basada en la centralización y el control. Un dispositivo final está conectado solo a un único nodo en la red, cumpliendo este "único nodo" el rol de nodo intermedio o central. Este nodo central tiene la función de interconectar todos los dispositivos de la red para que puedan comunicarse intercambiando mensajes usando ciertos protocolos.

Una topología Per-To-Per, a diferencia de la topología estrella, no está orientada a la centralización, más bien, cada dispositivo puede comunicarse con cualquier otro dispositivo dentro de su radio de cobertura directamente, eliminando la necesidad de un nodo central. La gran desventaja que se tiene con este tipo de topologías es que la cobertura de la red se limita al radio de cobertura de cada dispositivo, por lo que, si existe algún dispositivo fuera de ese radio, no podrá comunicarse con dicho dispositivo.

Por último, tenemos a la topología Mesh en donde cada dispositivo puede tener la función de nodo final, así como nodo intermedio, pudiendo retransmitir la data de otros dispositivos. Se crea una topología con la ausencia de un nodo central, pero pudiendo extender la red más allá que la cobertura del dispositivo.

Figura 6

Tipos de Topología de Redes



Hablando en términos topológicos, se le denominara como "nodo" a cualquier dispositivo de una red que conforme la misma. Existes distintos tipos de nodos dependiendo al nivel que pertenezca:

a. **Nodo Final:** Es todo dispositivo encargado de capturar la información para el cual ha sido programado.

b. **Nodo Central:** También conocido como nodo intermedio es todo dispositivo encargado de recibir y transmitir toda la información de los nodos finales y transmitirlos hacia un nivel superior.

c. **Nodo Superior:** Son los dispositivos que se encargan del almacenamiento, procesamiento y visualización de la data.

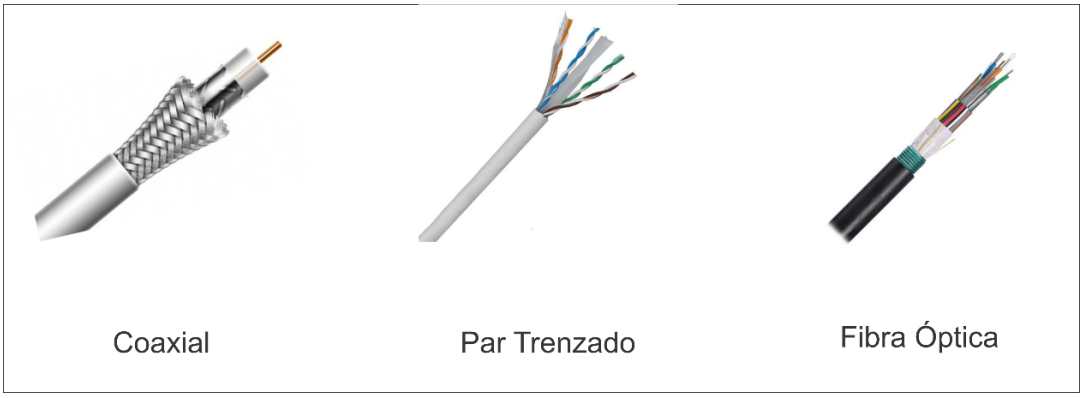
2.2.1.4. Medios de transmisión

El medio de transmisión es el camino físico que existe entre dos dispositivos. Cualquier medio que pueda transportar información en forma de señales electromagnéticas se puede usar como medio de transmisión en redes (Fernández, s.f.). Existen 2 tipos de medios principales, los medios guiados y no guiados y forman parte de la capa de Acceso a la Red del modelo TCP/IP.

Los medios guiados son aquellos medios que transportan la señal de información mediante un camino físico. Estos “caminos” son el cable coaxial, la fibra óptica y el par trenzado. Los parámetros más importantes en este tipo de medios es la velocidad de transmisión, Ancho de Banda y la distancia máxima permitida. En la figura 7 se muestran los medios guiados.

Figura 7

Tipos de Medios Guiados.



El medio guiado más común es el par trenzado, que se encuentra en todos los hogares y cableado estructurado en áreas empresariales. El protocolo más utilizado de medios guiados en redes LAN es el Ethernet, este permite comunicar distintos dispositivos como ordenadores, impresoras, servidores, etc. El estándar en el cual se

basa Ethernet es el IEEE 802.3 y sus variaciones, los cuales definen distintas velocidades, tipo de medio guiado, entre otros. Mas detalles se muestran tabla 1.

Tabla 1

Estándar IEEE802.3 y sus variantes

Año	IEEE	Nombre	Velocidad	Tipo de cable
1990	802.3i	10BASE-T	10 Mb/s	Cat-3
1995	802.3u	100BASE-T	100 Mb/s	Cat-5
1998	802.3z	1000BASE-SX	1 Gb/s	Fibra multimodo
		1000BASE-LX/EX	1 Gb/s	Fibra monomodo
1999	802.3ab	1000BASE-T	1 Gb/s	Cat-5e o superior
2003	802.3ae	10GBASE-SR	10 Gb/s	Laser-Opt. MMF
		10GBASE-LR/ER	10 Gb/s	Fibra monomodo
2006	802.3an	10GBASE-T	10 Gb/s	Cat-6A
2010	802.3ba	40GBASE-SR4/LR4	40 Gb/s	Laser MMF o SMF
		100GBASE-SR10/LR4/ER4	100 Gb/s	Laser MMF o SMF
2015	802.3bq	40GBASE-T	40 Gb/s	Cat-8 (Clase I y II)

El segundo tipo de medio es el no guiado, es decir, que no usa un medio físico como el cable para propagarse, más bien, está compuesto de antenas de transmisión y recepción. Durante la transmisión la antena emite energía en forma de ondas electromagnéticas, y en la recepción la antena capta dichas ondas. Cuando los electrones se mueven, crean ondas electromagnéticas que pueden propagarse incluso en el vacío. Una tecnología común en las redes WLAN es Wifi, que permite la interconexión inalámbrica de dispositivos electrónicos. Esta tecnología está basada en los estándares de la IEEE802.11 relacionado con redes inalámbricas. Al igual que es estándar de Ethernet, Wifi también tiene sus variantes y versiones mejoradas como se muestra en la siguiente tabla. Mas detalles se muestran en la tabla 2.

Para medir la calidad de la señal inalámbrica existen dos factores fundamentales los cuales son el RSSI y la relación señal a ruido SNR.

El RSSI es el indicador de fuerza de señal recibida la cual es medida en dBm. Este valor nos indica el nivel de potencia de la señal recibida por un dispositivo final en redes inalámbricas (Wifi o móvil). Los valores y la calidad de esta se detallan en la tabla 3.

Tabla 2

Estándar IEEE802.11 y sus variantes

Año	IEEE	Generación	Velocidad	Frecuencia
1997	802.11	-	2 Mb/s	2.4 GHz
1999	802.11b	-	11 Mb/s	2.4 GHz
1999	802.11a	-	54 Mb/s	5 GHz
2003	802.11g	-	54 Mb/s	2.4 GHz
2008	802.11n	Wi-Fi 4	600 Mb/s	2.4 / 5 GHz
2014	802.11ac	Wi-Fi 5	6933 Mb/s	5 GHz
2019	802.11ax	Wi-Fi 6	9608 Mb/s	2.4 / 5 GHz
2020	802.11ax	Wi-Fi 6E	9608 Mb/s	2.4 / 5 / 6 GHz
2022	802.11be	Wi-Fi 7	40 Gb/s	2.4 / 5 / 6 GHz

Tabla 3

Valores de RSSI

RSSI (dBm)	Intensidad de señal
> -50	Excelente intensidad de señal
-50 a -67	Buena intensidad de señal para navegación web, llamadas de voz/video.
-67 a -70	Buena intensidad para la navegación web, entrega de paquetes confiable.
-70 a -80	Mala conectividad, la entrega de paquetes puede no ser confiable.
-90 a -100	Peor intensidad o sin señal.

Nota. Hay que entender sobre dBm es que estamos trabajando en negativos. -30 es una señal más alta que -80, porque -80 es un número mucho más bajo (Singh, s.f.).

Luego de esto tenemos la relación señal a ruido. Se define como la relación que existente entre la potencia de la señal que se transmite con respecto a la potencia del ruido que la perturba. Estos valores se miden en decibelios y mientras más alto sea este

valor, mejor será el rendimiento de la red. Los valores de SNR se describen a continuación en la tabla 4.

Tabla 4

Valores de SNR y calidad de señal.

SNR (dB)	Calidad de Señal
>40	Excelente calidad, las velocidades de datos más altas.
25 a 40	Muy buena calidad, altas tasas de datos.
15 a 25	Buena calidad, buenas tasas de datos.
10 a 15	Buena calidad, velocidades de datos más bajas.
0 a 10	Señal baja o nula, velocidad de datos muy baja o nada.

2.2.2. Monitoreo de la Calidad del Aire

Martínez y Romieu (1997), nos da un concepto concreto sobre lo que es monitoreo de la calidad del aire o monitoreo atmosférico el cual lo definen como “Todas las metodologías diseñadas para hacer un muestreo, analizar y procesar en forma continua las concentraciones de sustancias o de contaminantes presentes en el aire en un lugar establecido y durante un tiempo determinado”.

2.2.2.1. Niveles de gases máximos permitidos

Hablando en términos de gases contaminantes tenemos como principal actor al CO_2 que, en ambientes interiores, se está tomando como indicador de la calidad de aire la concentración del dióxido de carbono producido en la respiración de los ocupantes del edificio y así, si no existe una reducción de su concentración por otro medio distinto de la ventilación, se considera que la ventilación es inadecuada cuando se superan las 1000 ppm de dióxido de carbono (INSST, 2000).

Este y otros gases contaminantes son detallados en la tabla 5 la cual nos muestra los efectos negativos en la salud si una persona está expuesta (INSST, 2001).

Tabla 5

Valores guía según la OMS basados en efectos conocidos para la salud.

Compuesto	**Efectos sobre la Salud**	**Valor Guía ($\mu g/m^3$)**	**Tiempo de Exposición**
Dióxido de azufre	Cambios en la función pulmonar en asmáticos. Aumento de los síntomas respiratorios en individuos sensibles.	500	10 minutos
		125	24 horas
		50	1 año
Dióxido de nitrógeno	Ligeros cambios de la función pulmonar en asmáticos	200	1 hora
		40	1 año
Monóxido de carbono	Nivel crítico de Carboxihemoglobina < 2,5%	100000	15 minutos
		60000	30 minutos
		30000	1 hora
		10000	8 horas
Ozono	Respuestas de la función respiratoria	120	8 horas

Department of Occupational Safety and Health, Malasia (2010) nos da una lista de rangos aceptables, en sus unidades respectivas, de las sustancias o contaminantes en el aire más general y completa, la cual se muestra en la tabla 6.

Tabla 6

Rango aceptable para parámetros ambientales.

Parámetros	**Rango aceptable**
Monóxido de carbono	< 10 ppm
Dióxido de carbono	< = 1000 ppm
Temperatura	23 °C – 26°C
Humedad Relativa	40% - 70%
Movimiento del aire	0,15 – 0,5 m/s
Microbiológico	-
Recuento total de bacterias	500 cfu/m^3
Recuento total de hongos	1000 cfu/m^3
Partículas respirables (PM10)	0,150 mg/ m^3
Compuestos orgánicos volátiles totales (COV)	3 ppm
Formaldehído	0,1 ppm
Ozono	0,05 ppm

2.2.2.2. Índice de calidad de Aire

El índice de calidad de aire (IAQ o INCA) es un numero o valor adimensional, es decir que no tiene una unidad de medición. Este valor nos indica, de forma cualitativa, la calidad de aire en cierto sector y su efecto en la salud humana. Para una mejor comprensión de este valor, se divide en 4 categorías (tabla 7). La banda de color verde comprende valores del INCA de 0 a 50 y significa que la calidad del aire es buena; la banda de color amarillo comprende valores de 51 a 100 e indica una calidad moderada del aire; la banda de color anaranjado se encuentra comprendida entre los valores 101 y el valor umbral del estado de cuidado (VUEC) de cada contaminante, lo que nos indica que la calidad del aire es mala; finalmente el color rojo de la cuarta banda nos indica que la calidad del aire es mayor al valor umbral del estado de cuidado del contaminante, a partir de este valor corresponde la aplicación de los Niveles de Estados de Alerta Nacionales por parte de la autoridad de Salud (MINAM, 2016).

Tabla 7

Valores de Índice de Calidad de Aire

Calificación	Valores de INCA	Colores
Buena	0 - 50	Verde
Moderada	51 - 100	Amarillo
Mala	101 - VUEC*	Naranja
VUEC*	> VUEC*	Rojo

Nota. *Valor Umbral del Estado de Ciudadano (MINAM, 2016).

Para calcular los índices de calidad de aire se toma como referencia los Estándares de Calidad de Aire (ECA) de aire y como rango final, el valor umbral de la aplicación de los Niveles de Estado de Alerta. El cálculo matemático de INCA para cada contaminante está basado en una relación entre la concentración del contaminante y su valor en el ECA. La información se muestra en la tabla 8.

Tabla 8

Cálculo del Índice de Calidad del Aire por contaminante.

Intervalo de ICA	Intervalo de concentración (µg/m³)	Ecuación
Material Particulado (PM2.5) promedio 24 horas		
0 - 50	0 – 12,5	
51 – 100	12,6 – 25	$I(PM2.5) = [PM2.5] * 100/25$
101 – 500	25,1 – 125	
> 500	> 125	
Material Particulado (PM10) promedio 24 horas		
0 - 50	0 - 75	
51 – 100	76 - 150	$I(PM10) = [PM10] * 100/150$
101 – 167	151 - 250	
> 167	> 250	
Monóxido de Carbono (CO) promedio 8 horas		
0 - 50	0 - 5049	$I(CO) = [CO] * 100/10000$
51 – 100	5050 - 10049	
101 – 150	10050 - 15049	
> 150	> 15050	

Dióxido de Azufre (SO_2) promedio 24 horas		
0 - 50	0 - 10	
51 – 100	11 - 20	$I(\mathrm{SO2}) = [\mathrm{SO2}] * 100/20$
101 – 625	21 - 500	
> 625	> 500	
Dióxido de Nitrógeno (NO_2) promedio 1 hora		
0 - 50	0 - 100	$I(\mathrm{NO2}) = [\mathrm{NO2}] * 100/200$
51 – 100	101 - 200	
101 – 150	201 - 300	
> 150	> 300	
Sulfuro de Hidrogeno (H_2S) promedio 24 horas		
0 - 50	0 - 75	
51 – 100	76 – 150	$I(\mathrm{H2S}) = [\mathrm{H2S}] * 100/150$
101 – 1000	151 – 1500	
> 1000	> 1500	
Ozono (O_3) promedio 8 horas		
0 - 50	0 - 75	$I(\mathrm{H2S}) = [\mathrm{H2S}] * 100/120$
51 – 100	76 – 150	
101 – 1000	151 – 1500	
> 1000	> 1500	

Nota. [ ]= Concentración del contaminante en µg/m^3 (MINAM, 2016).

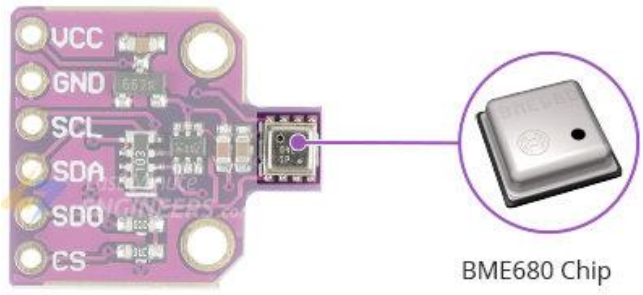
2.2.2.3. Captación de parámetros de calidad de aire

Para poder captar los parámetros necesarios para mantener la calidad del aire en un rango de valores aceptables se necesita el uso de sensores diseñados especialmente para detectar las concentraciones de gases en el aire. Los sensores son dispositivos electrónicos que nos permiten interactuar con el entorno, de forma que nos proporcionan información de ciertas variables que nos rodean para poder procesarlas y así generar ordenes o activar procesos (Serna, Ros, y Rico, 2010)

Existen distintos tipos de sensores de calidad de aire, cada uno de estos varia en cuanto a su precisión, incertidumbre y principio de funcionamiento. Uno de estos sensores es el sensor BME680 que es un sensor digital 4 en 1 con medición de gas, humedad, presión y temperatura basada en principios de detección probados (BOSCH, 2017). En la figura 8 se muestra el modelo usado para este proyecto.

Figura 8

Módulo del sensor BME680

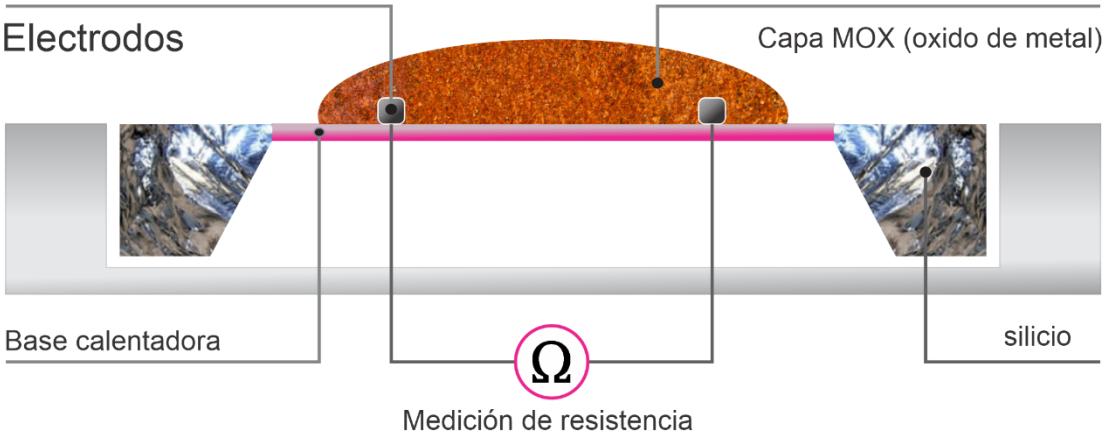


Nota. Nótese que el chip del sensor ocupa la parte mínima del módulo en general. Tomada de (Last Minute Engineers, s.f.).

Estos tipos de sensores son también conocidos como sensores de óxido de metal (MOX), los cuales cuentan con una base calentadora, una capa MOX, un semiconductor y un par de electrodos. Las partes principales se muestran en la figura 9.

Figura 9

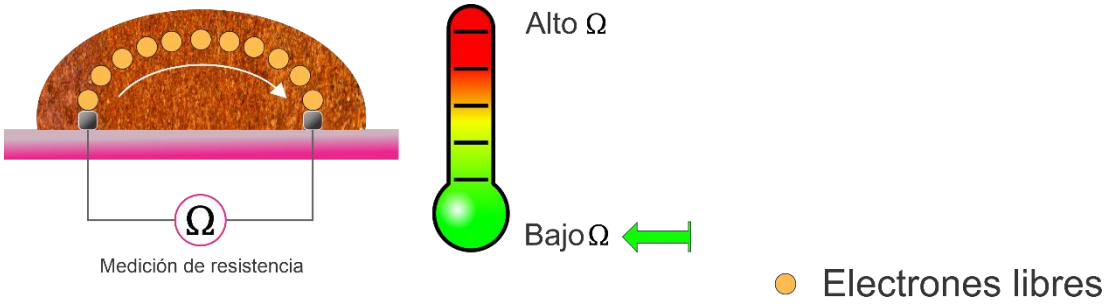
Diagrama interno de un sensor MOX



Los electrodos alimentan con una pequeña corriente la capa MOX creando un flujo de electrones libres. Esto se traduce en una resistencia total baja como se muestra en la figura 10.

Figura 10

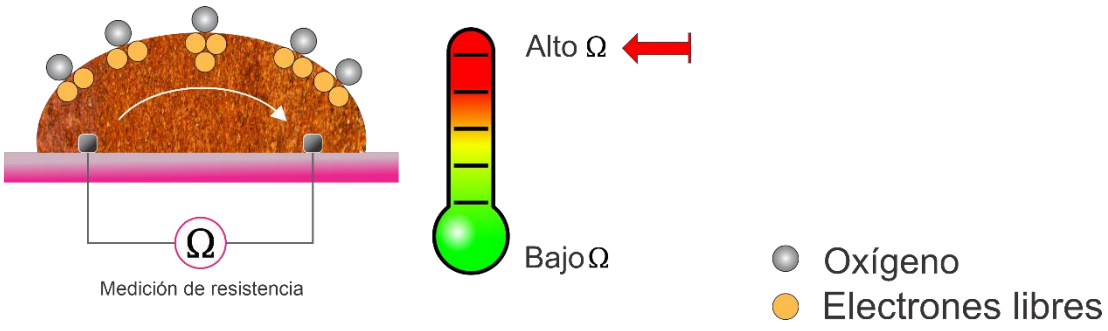
Corriente interna creada por los electrodos



Al mismo tiempo la base calentadora calienta la capa MOX (normalmente entre 200 °C y 400 °C) haciendo que, por adsorción, las moléculas de oxígeno libres se adhieran a este. Este efecto causa que los electrones libres que se encontraban en la capa MOX se peguen a las moléculas de oxígeno, evitando que haya un flujo de corriente. Esto se traduce en un aumento de la resistencia. Hasta ahora podemos deducir que el valor de la resistencia en el sensor es inversamente proporcional a la cantidad de gas que detecta. Todo este efecto se grafica en la figura 11.

Figura 11

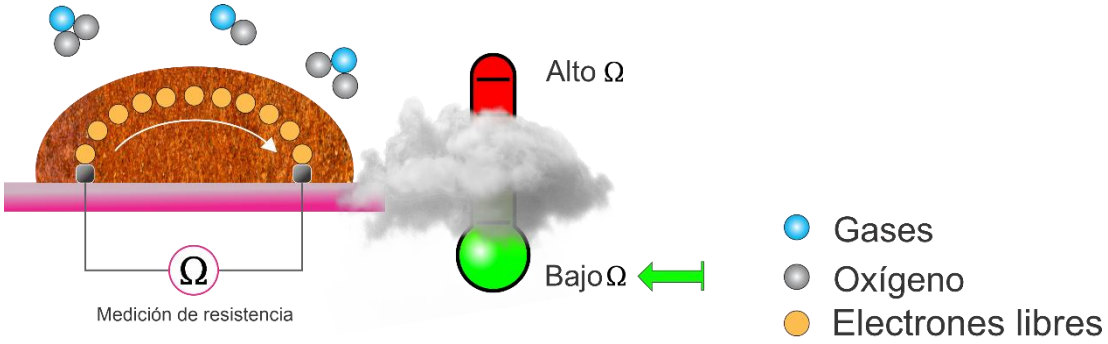
Adsorción de oxígeno debido al calentamiento de la capa MOX



Cuando el sensor está en presencia de moléculas de gases como CO2 y demás gases, este reacciona con las moléculas de oxígeno adheridas a la capa MOX a través de un proceso llamado oxidación-reducción por lo que, las moléculas de oxígeno se desprenden de la capa MOX liberando los electrones y, por lo tanto, reduciendo la resistencia total del circuito (figura 12). Es de esta manera en que este tipo de sensores detecta la concentración de gases.

Figura 12

Desprendimiento del oxígeno debido al gases contaminantes.



El sensor BME680 soporta dos tipos de protocolos de comunicación serial los cuales son I2C y SPI. Dependiendo del diseño del sensor con el microcontrolador se dispondrá de uno u otro.

Otro parámetro que es importante en la calidad de aire es la cantidad de material particulado, más conocido como PM. Existes múltiples sensores que nos permite detectar la concentración de PM, en partículas PM2.5 (material particulado con un diámetro menor a 2.5 micrómetros). Uno de estos sensores es el DSM501a.

El DSM501a, mostrado en la figura 13, es un sensor de material particulado que puede detectar humo, polen de tabaco, polvo doméstico, etc. Como se muestra en la figura 14, este sensor está compuesto por un led infrarrojo que incide su luz sobre un lente detector. Cuando las partículas de aire pasan entre el led y el detector, las partículas, colisionan con el haz de luz provocando la dispersión de esta; el detector mide la dispersión de las luces reflejadas por las partículas. Esta dispersión se refleja en dos señales PWM las cuales representar las partículas con un diámetro mayor a 1um y 2.5um. También cuenta con una resistencia la cual empieza a calentarse y a calentar el aire a su alrededor, este aire caliente crea una corriente ascendente debido a que el aire caliente es menos denso y por ende tiende a subir.

Figura 13

Sensor de polvo DSM501a

Figura 14

Diagrama de bloque interno del sensor DSM501a

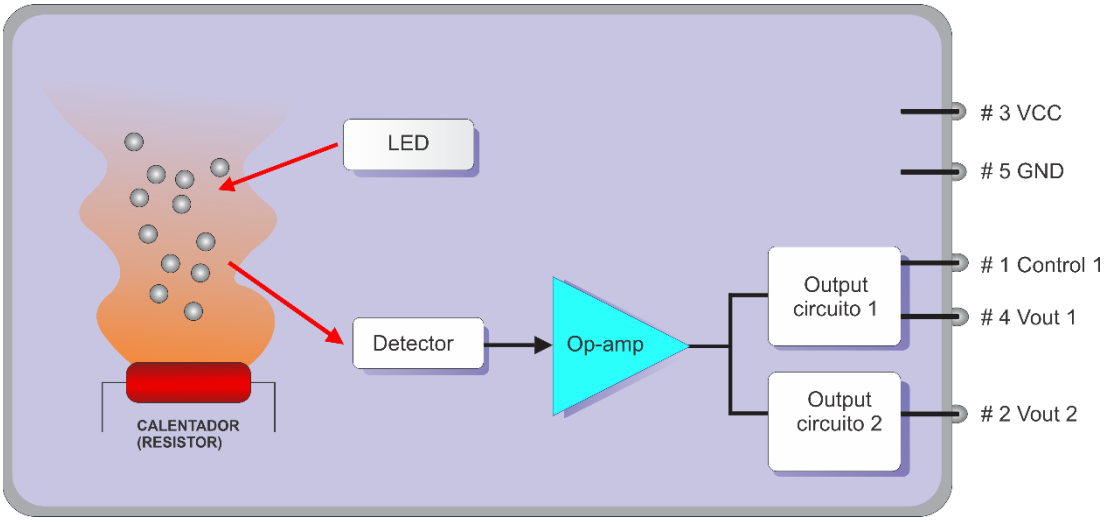
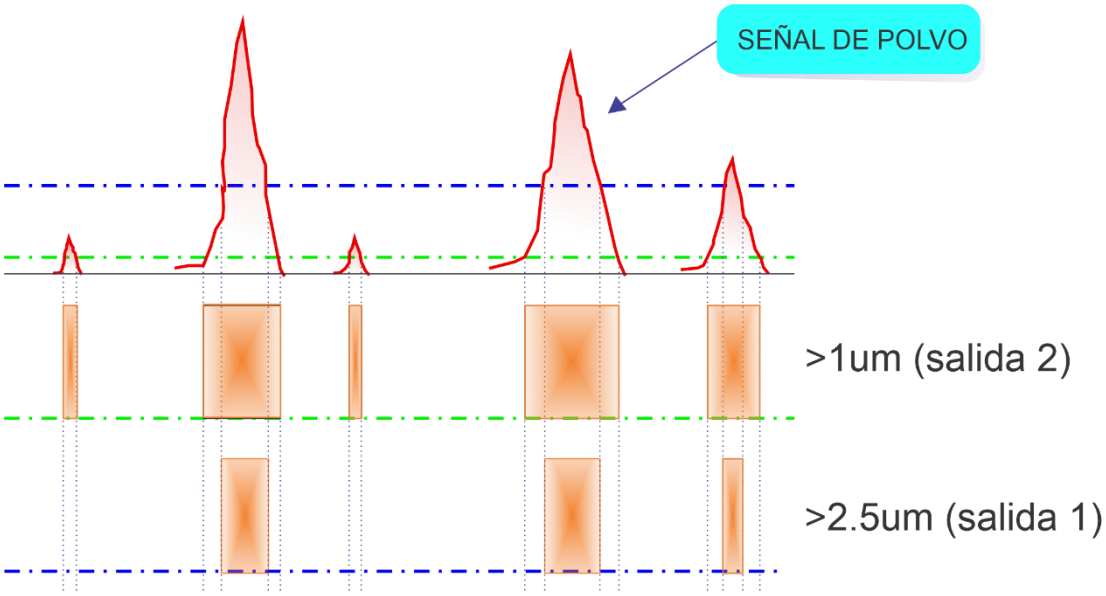


Figura 15

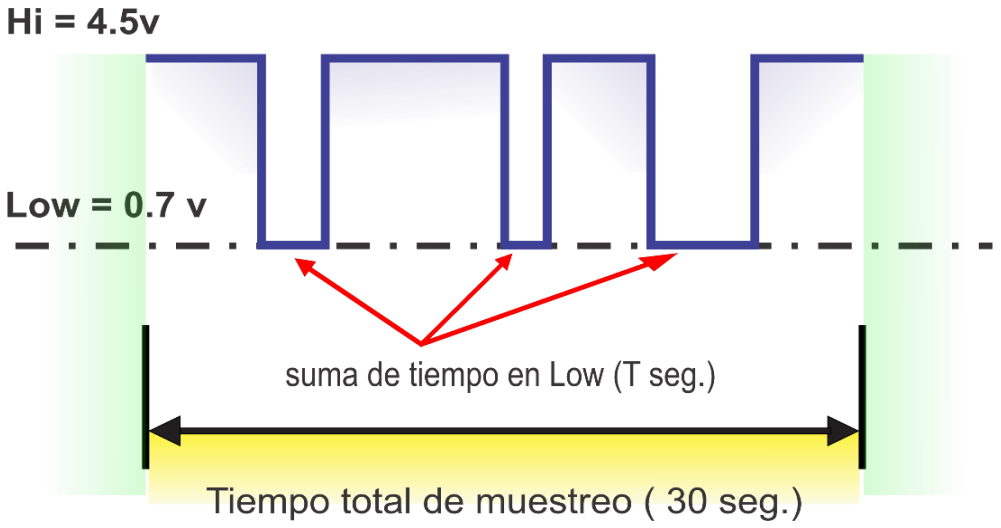
Señales de salida del DSM501A



El tiempo que demora en tomar la muestra y arrojar un valor confiable es de 30 segundos. El material particulado detectado es representado por todo el tiempo en que la señal PWM está en LOW, por lo que, para calcular la concentración de PM2.5 se debe de hallar el tiempo de LOW en % con respecto al tiempo total de la muestra que es de 30s. Esta relación se conoce como Lowratio y se puede entender de manera grafica en la figura 16.

Figura 16

Periodo de muestra de la señal de salida PWM del DSM501a



Una vez obtenido el valor de Lowratio, se procede a calcular la concentración utilizando la gráfica de la figura 17 dada por el fabricante. Con la gráfica se puede deducir la ecuación que define el comportamiento del sensor, el cual es:

$$Con = 0{,}5831(r)^3 - 15{,}924(r)^2 + 729.37(r) - 82{,}523; \qquad (1)$$

Donde:

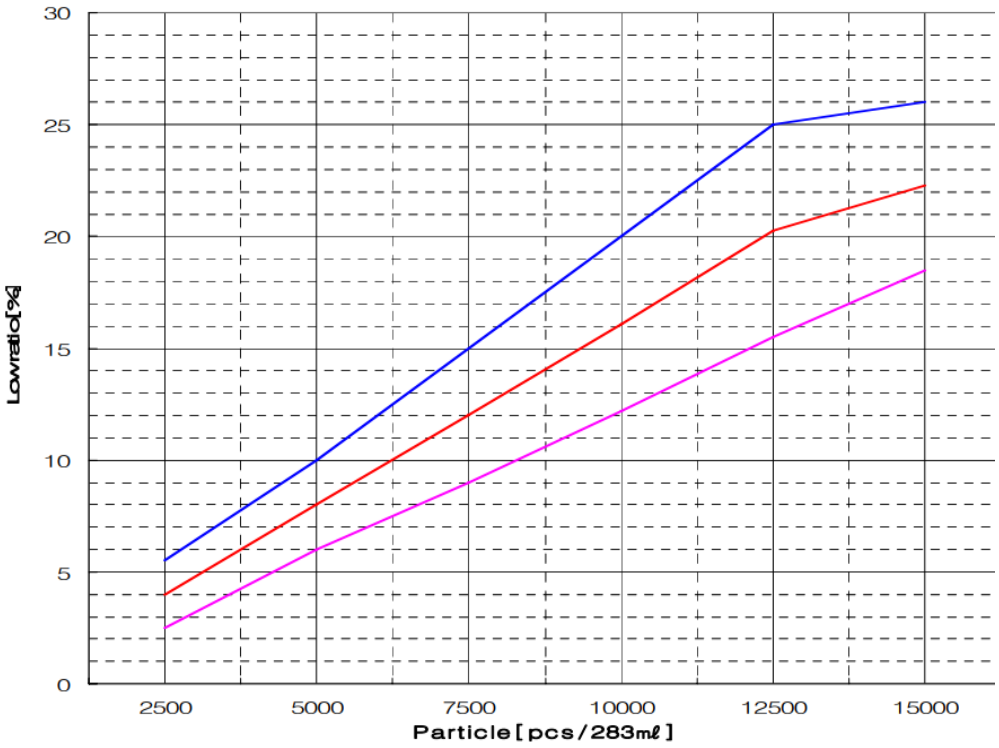
Con = Es la concentración de polvo en parts/283mL.

r = Es el Lowratio.

Donde r es la relación entre el tiempo en LOW y el tiempo total en porcentaje.

Figura 17

Grafica de Lowratio sobre la concentración de partículas.



2.3. Definición de Términos

2.3.1. Sistema IoT

Son sistemas electrónicos que transmiten datos de proceso desde áreas seguras o peligrosas conectando sensores y actuadores binarios y analógicos al sistema de

control a través de una interfaz de bus de datos con el objetivo de realizar algún tipo de proceso.

2.3.2. Transmisión de datos

Transmisión de datos, transmisión o comunicaciones digitales es la transferencia física de datos por un canal de comunicación punto a punto o punto a multipunto.

2.3.3. Monitoreo

La teoría de la planificación del desarrollo define el seguimiento o monitoreo como un ejercicio destinado a identificar de manera sistemática la calidad del desempeño de un sistema, subsistema o proceso a efecto de introducir los ajustes o cambios pertinentes y oportunos para el logro de sus resultados y efectos en el entorno (Valle y Rivera, 2008).

2.3.4. Calidad del Aire

Se define la inmisión o calidad del aire como la concentración de contaminante que llega a un receptor, más o menos lejano de la fuente de emisión, una vez transportado y difundido por la atmósfera (Troposfera, s.f.)

CAPÍTULO III: MARCO METODOLÓGICO

La formulación del marco metodológico en una investigación es permitir, descubrir los supuestos del estudio para reconstruir datos, a partir de conceptos teóricos habitualmente operacionalizados. Significa detallar cada aspecto seleccionado para desarrollar dentro del proyecto de investigación que deben ser justificado por el investigador. Respaldado por el criterio de expertos en la temática, sirviendo para responder al “como” de la investigación (Azuero, 2018).

3.1. Diseño de la Investigación

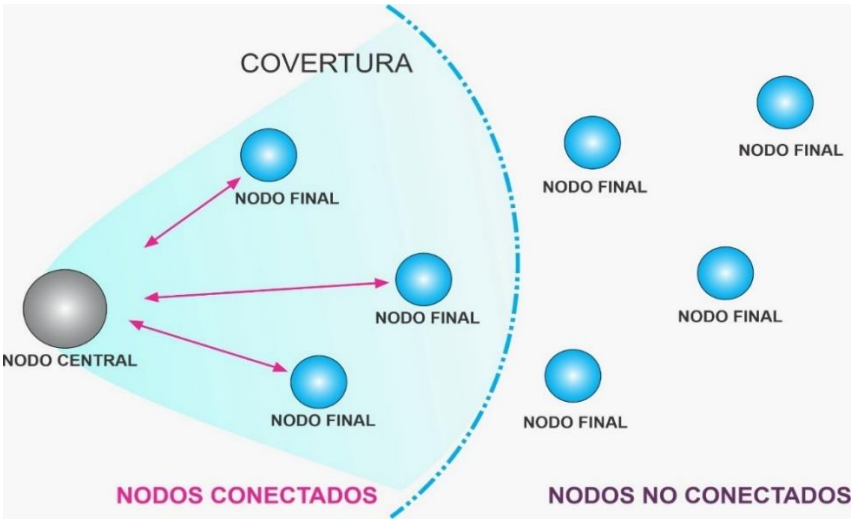
3.1.1. Criterios iniciales

La idea principal y que se mantuvo durante todo el proyecto fue hacer una red de sensores independiente, es decir, que no dependa de un despliegue de Puntos de Acceso para cubrir todo el espacio físico que ocupa toda la Escuela Profesional de Ingeniería Electrónica y además que sea inalámbrica. La topología que más se adecuó a nuestras necesidades fue la topología tipo Malla (figura 19) en la que cada dispositivo puede autoconfigurarse tanto como Punto de acceso, o como Nodo final, eliminando así la restricción, en cuanto a cobertura, que existe al usar una topología tipo Estrella (figura 18) que es la más común en redes LAN o WLAN.

La red Mesh solo estaba conformada por lo sensores, por lo que, se necesitaba de una interfaz que nos permita comunicarnos con la red WLAN de la institución para poder mandar toda la data capturada por los sensores a un servidor de base de datos. Es por eso que se implementó un nodo central el cual se encargaba de unir la red Mesh con la red WLAN a través de una comunicación serial.

Figura 18

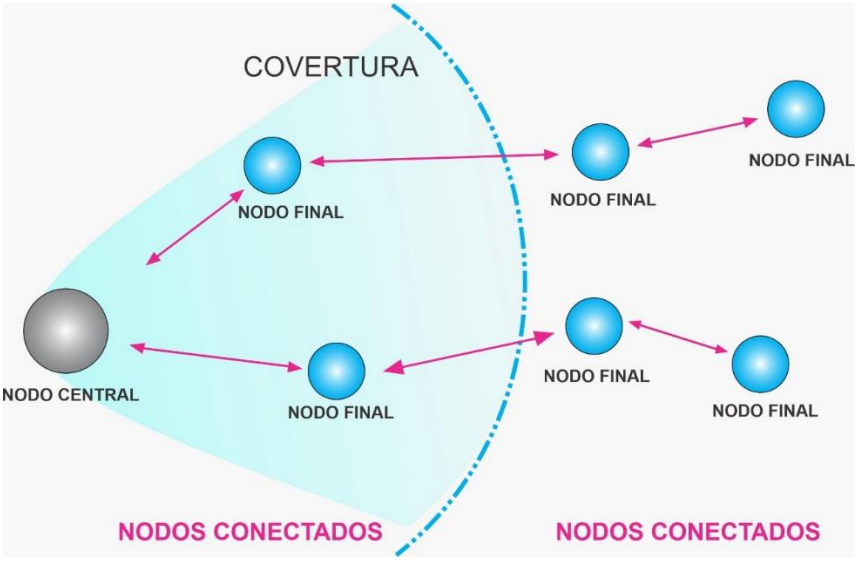
Topología tipo Estrella



Nota. Nótese que, en este tipo de topología, la distancia máxima entre los dispositivos finales y el nodo central está limitada por la cobertura de este último.

Figura 19

Topología tipo Malla



Nota. Este tipo de topología permite extender el alcance de la red sin la necesidad de hacer un despliegue de Puntos de Acceso. Cada dispositivo se autoconfigura como AP o nodo final.

3.1.2. Criterio de selección de los dispositivos

3.1.2.1. Sensores

Por la naturaleza de la propuesta, el criterio principal fue buscar sensores de calidad de aire compactos que cumplan con medir parámetros como CO2, Temperatura, Humedad, Índice de calidad de Aire (INCA) y material particulado (PM2.5 o PM10). Se optó por usar el sensor BME680 de Bosch el cual es un sensor que mide Temperatura, Humedad, INCA y CO_2 equivalente. Además, que tiene una precisión superior al promedio de sensores, es compacto y muy ligero. También nos ofrece interfaces seriales I2C y SPI compatible con la mayoría de los microcontroladores. En la tabla 9 se muestran las principales características técnicas del sensor BME680.

Tabla 9

Características técnicas del sensor BME680

Característica	**Min.**	**Type**	**Max.**	**Unidad**
Voltaje	1,71	1,8	3,6	V
Voltaje I/O	1,2	1,6	3,6	V
Corriente (Sleep)	-	0,15	1	µA
Corriente (Al medir Humedad)	-	340	450	µA
Corriente (Al medir Presión)	-	714	849	µA
Corriente (Al medir Temperatura)	-	350		µA
Temperatura	-40	-	85	°C
Humedad	0	-	100	%
Presión	300	-	1100	hPa
CO_2	400	-	-	ppm
INCA	0	-	500	-

También se optó por utilizar el sensor DSM501a que mide el material particulado PM2.5 por tener una mejor precisión en comparación con otros sensores del mismo tipo. Este sensor nos da una señal PWM por lo que se tiene que hacer el cálculo para hallar el valor en $\mu g/m^3$. En la tabla 10 se muestran las principales características técnicas del sensor DSM501a.

Tabla 10

Características técnicas del sensor DSM501a

Característica	Min.	Type	Max.	Unidad
Voltaje	4,5	5,0	5,5	V
Corriente	-	-	90	mA
Rango de temperatura	-10	-	65	°C
Concentración	0	-	15000	pcs/283mL

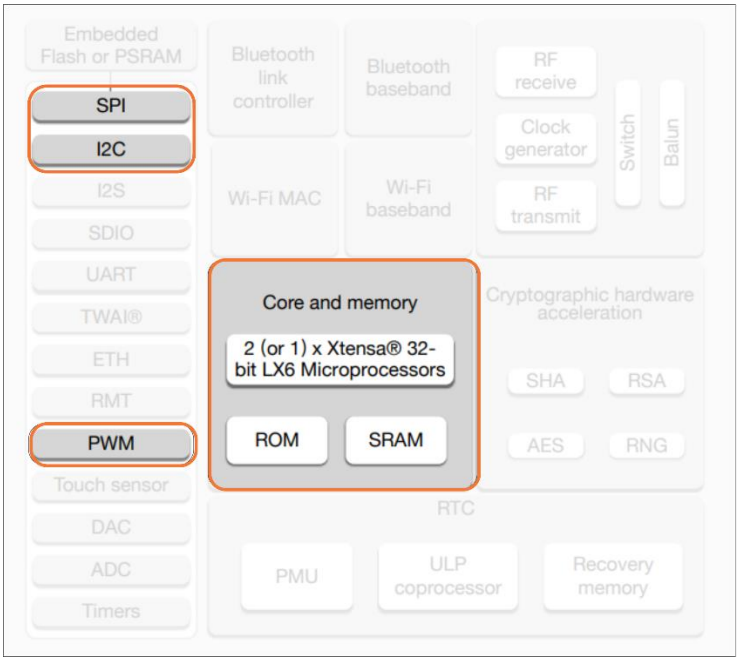
3.1.2.2. Microcontrolador

El sensor BME680 tiene una librería desarrollada por Bosch que es compatible con C++, el cual necesita un microcontrolador con una memoria RAM mayor a 2KBits, además debía de contar con un protocolo de comunicación serial I2C o SPI. Para el DSM501a se necesitó que el microcontrolador tenga entradas digitales con interrupciones para la lectura PWM, además que, debido a que la señal de PWM en el que el valor está en LOW son muy cortas (en el orden de los microsegundos) se necesita un ciclo de reloj considerable. Para la lectura, procesamiento de los sensores se consideró los procesadores Xtensa Dual-Core LX6 (que vienen dentro de los módulos de desarrollo ESP32) ya que son chips de alto rendimiento y bajo costo- consumo gracias a su tecnología RTO. El diagrama de dicho microcontrolador se muestra en la figura 20 y sus principales características técnicas en la tabla 11.

Para el desarrollo de la topología tipo Malla se tuvo en cuenta, como principal parámetro, la potencia de transmisión. En el mercado existen múltiples tipos de chips wifi en placas de desarrollo. El inconveniente de estas placas es que tenían una antena PCB impresa con una potencia de transmisión limitada. Debido a esto se consideró utilizar un chip con un conector para una antena externa. El chip que más se ajustaba a estas necesidades fue el ESP-07 (el cual se basa en el ESP8266) el cual cuenta, como se ve en la figura 21, con un conector UFL (el cual sirve para agregar una antena externa) además de una antena cerámica interna. Las principales especificaciones técnicas se señalan en la tabla 12.

Figura 20

Diagrama de Bloques interno del ESP32



Nota. En la figura se muestra que este chip cuenta con los requisitos necesarios para la lectura y procesamiento de datos.

Tabla 11

Características técnicas de Xtensa Dual-Core LX6 (ESP32).

Característica	**Valor**	**Unidad**
Voltaje de operación	3,3 ~ 3,6	V
Voltaje de alimentación	3,3 ó 5	V
Corriente de operación	80	mA
Corriente mínima enviada por la fuente	500	mA
Temperatura de operación	-40 ~ 85	°C
Cristal interno	40	MHz
SRAM (datos e instrucciones)	520	KB

Figura 21

Módulo ESP-07.

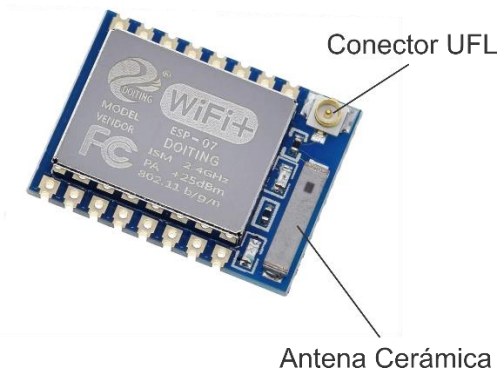


Tabla 12

Características técnicas del ESP-07

Característica	**Valor**	**Unidad**
Voltaje de operación / alimentación	3,3 ~ 3,6	V
Corriente de operación	80	mA
Temperatura de operación	-40 ~ 85	°C
Cristal interno	24	MHz
SRAM (datos e instrucciones)	36	KB
Potencia de transmisión	+25	dBm

Para aumentar el alcance en la señal de este chip, se agregó una antena externa de 2.4 GHz con una ganancia de 2 dBi (figura 22) y una longitud de 11cm.

Figura 22

Antena Wifi con su conector SMA

Para comprobar la efectividad de esta antena, se realizó pruebas de cobertura evaluando el valor de RSSI que es un indicador de fuerza de la señal recibida en dBm. La distancia máxima la cual se midió fue de 50,85m de longitud atravesando una pared y un parque con árboles frondosos como se ve en la figura 23. El valor de RSSI medido llego a ser de -72 dBm, siendo este un valor mínimo aceptable, y teniendo un rendimiento mucho mayor al de la antena integrada. La distancia máxima experimental que se llegó tener con la antena integrada atravesando una pared fue de 9,84m perdiendo la conexión al sobrepasar dicha distancia.

Figura 23

Prueba de cobertura usando un ESP-07 con una antena externa.

Nota. La ganancia de la antena es de 2dBi.

3.1.2.3. Interfaz gráfica

Para el desarrollo del software de la interfaz gráfica decidimos realizarlo a través de una aplicación de escritorio. Una aplicación consta de dos partes importantes: el Frontend, que es la parte visual y gráfica del software; y el Backend que es la lógica detrás del programa (incluido la base de datos). Debido a que el equipo en el cual trabajamos tenía el sistema operativo de Windows, la aplicación debe de ser desarrollada para esta plataforma. Por excelencia el entorno para desarrollo de aplicaciones de escritorio de Windows es Visual Studio, el cual nos permite trabajar sobre el lenguaje de programación C#. A su vez, también nos ofrece la capacidad de gestionar una base de datos SQL Server a través de un framework llamado EntityFramework.

3.1.3. Desarrollo del prototipo

3.1.3.1. Implementación de Hardware

Para el desarrollo del prototipo empezamos con diseñar la fuente de alimentación. Es importante calcular la corriente de consumo máxima del circuito. Los detalles se muestran en la tabla 13.

Tabla 13

Consumo de corriente total del circuito (Nodo Final).

Dispositivo	**Consumo**
Xtensa	80 mA
ESP-07	290 mA
BME680	849 µA
DSM501a	90 mA
Subtotal	**460,85 mA**
Tolerancia (20%)	92.17 mA
TOTAL	**553,02 mA**

Teníamos dispositivos que se alimentaban con 5v y 3,3v, eso significó que nuestra fuente de alimentación debía de tener dos voltajes de salida. Para esto usamos un transformador de 220 Vac a 12 Vdc 1A. También se usó dos reguladores de voltaje, el regulador L7805cv (Vi_{max} = 35 Vdc, V_{out} = 5 Vdc) con su respectivo circuito de amplificación y el módulo regulador AMS1117 (V_{in} = 5 Vdc, V_{out} = 3,3 Vdc). Los valores de los condensadores y el tipo de transistor se detallan en la figura 24.

Una vez terminada el circuito de alimentación, procedimos a realizar el circuito de mando entre los controladores y sensores. Para un solo nodo final utilizamos el ESP-07 como adaptador de red inalámbrico en la topología tipo Malla, y el procesador Xtensa para conectar y procesar los datos enviados por los sensores. Para la comunicación con el BME680 se utilizó la interfaz serial I2C y para la comunicación con el DSM501A, un pin digital que permite interrupciones externas. Debido a que este procesador viene en un módulo de desarrollo, no necesitó conexiones extras para su funcionamiento. El diagrama de conexión se muestra en la figura 25.

Figura 24

Esquemático del circuito de alimentación.

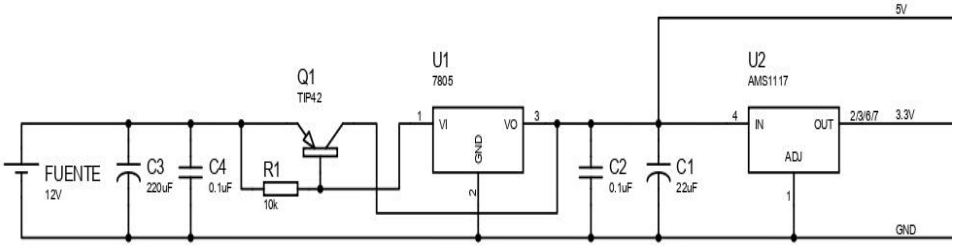
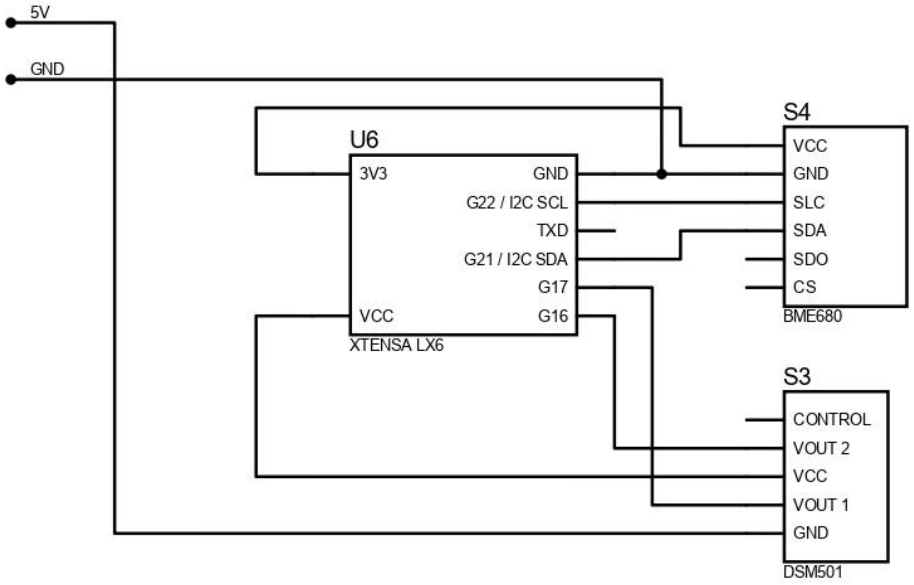


Figura 25

Conexión entre procesador Xtensa y los sensores.



Una vez que tenemos los datos captados y procesados, se tuvo que buscar una forma de trasmitirlo por la red Malla. Para esto se utilizó el chip ESP-07 que, a través de la comunicación serial convencional, se conectó al procesador Xtensa para la recepción de datos. Ya que el ESP-07 no es un módulo de desarrollo, se necesita hacer el circuito para adaptarlo y poder programarlo (ya que este no cuenta con un conector USB tipo C). El esquemático completo del nodo final se muestra en la figura 26 y la imagen física en la figura 28. Aparte del nodo final tenemos el nodo central, el cual es la interfaz que se encarga de comunicar la red Malla y la red WLAN. Este nodo no lleva sensores y la comunicación va del esp-07 al ESP8266. La comunicación entre estos dos es a través de la comunicación serial convencional. El esquemático completo del nodo central se muestra en la figura 27 y la imagen física en la figura 29. En general se creó 1 nodo central y 3 nodos finales.

Figura 26

Esquemático del circuito total del nodo final.

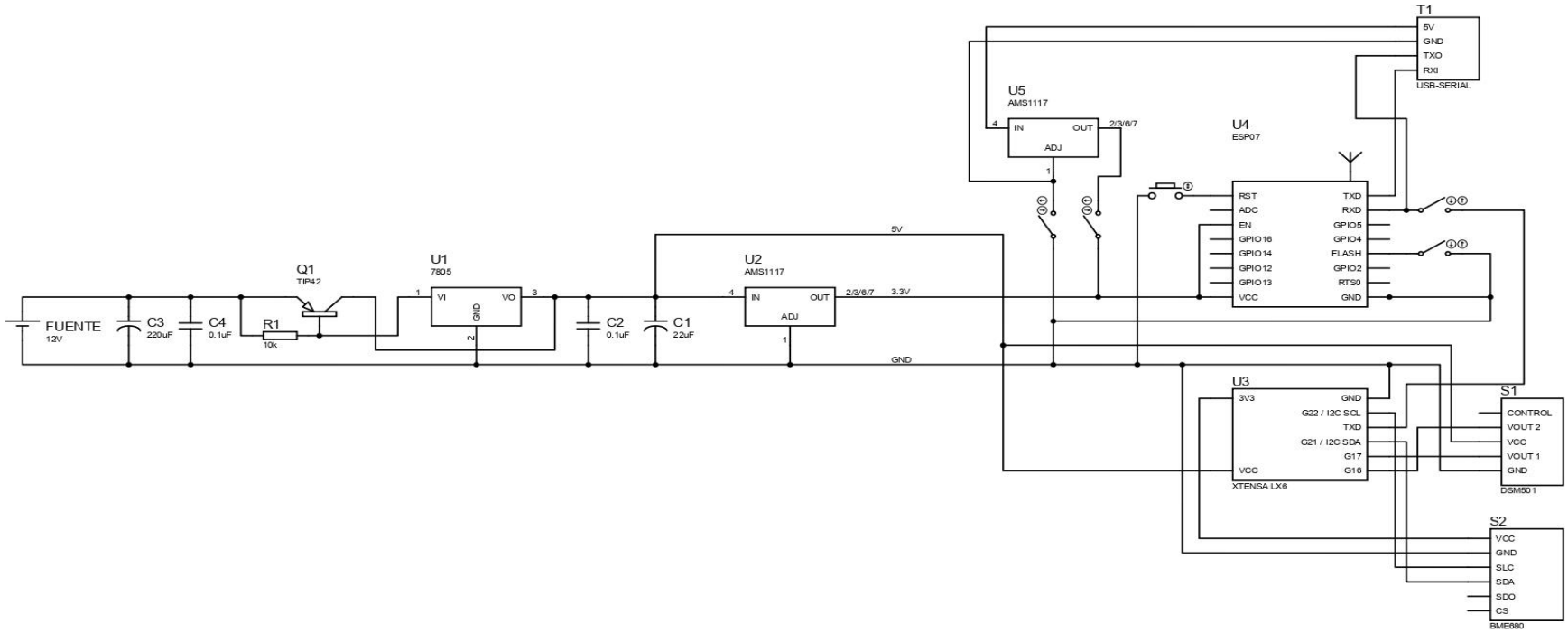


Figura 27

Esquemático del circuito total del nodo central.

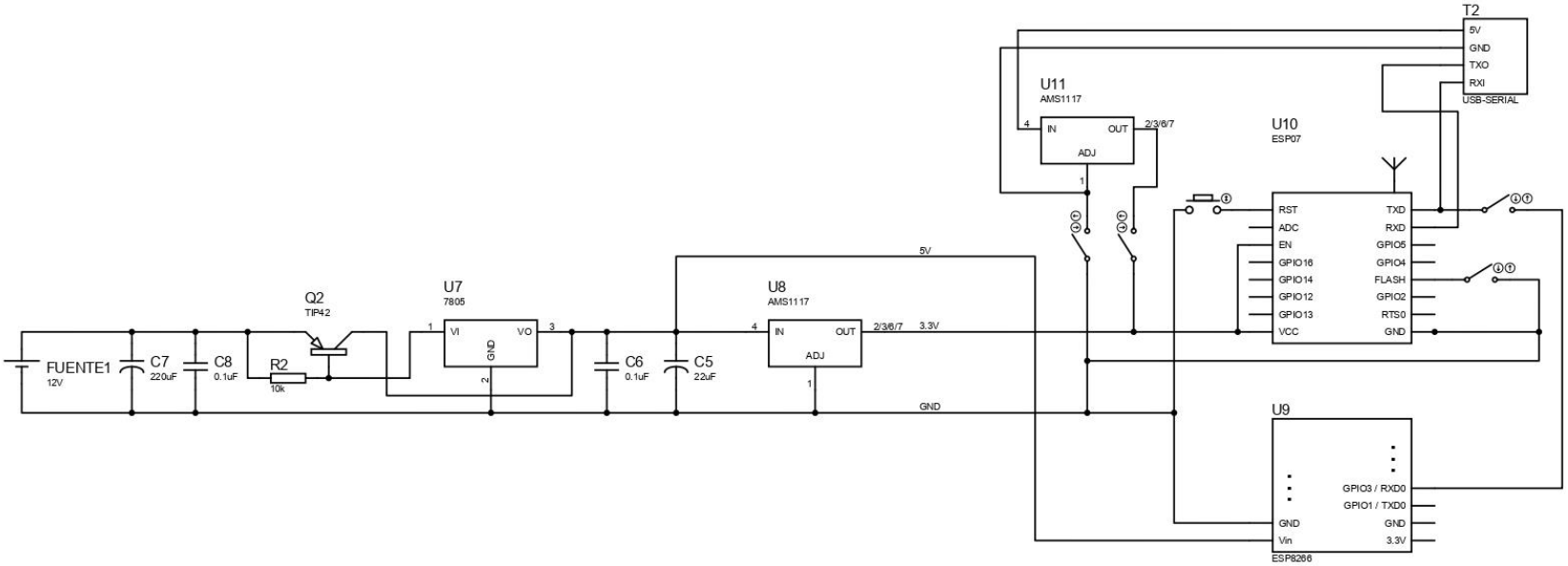


Figura 28

Ubicación de componentes dentro del Nodo Final.

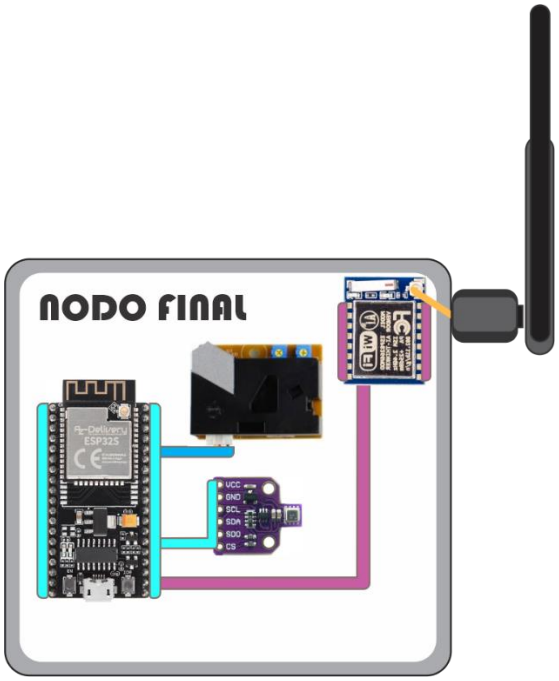
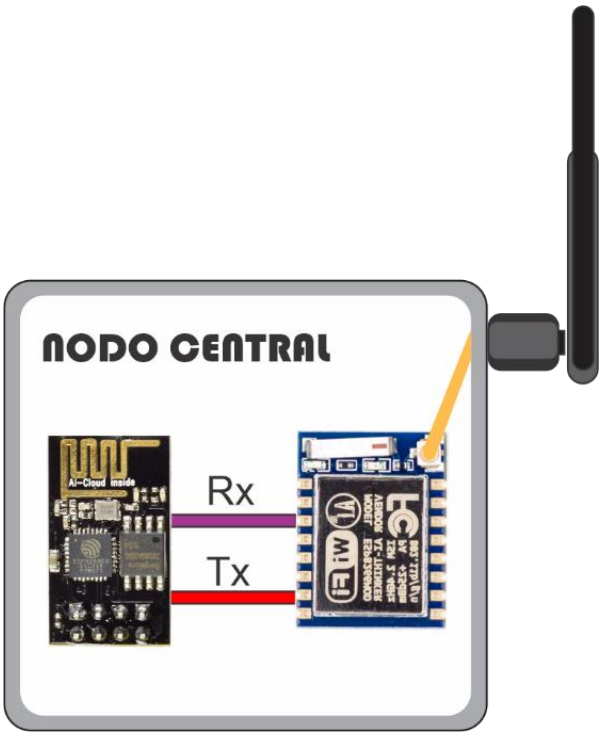


Figura 29

Ubicación de componentes dentro del nodo Central

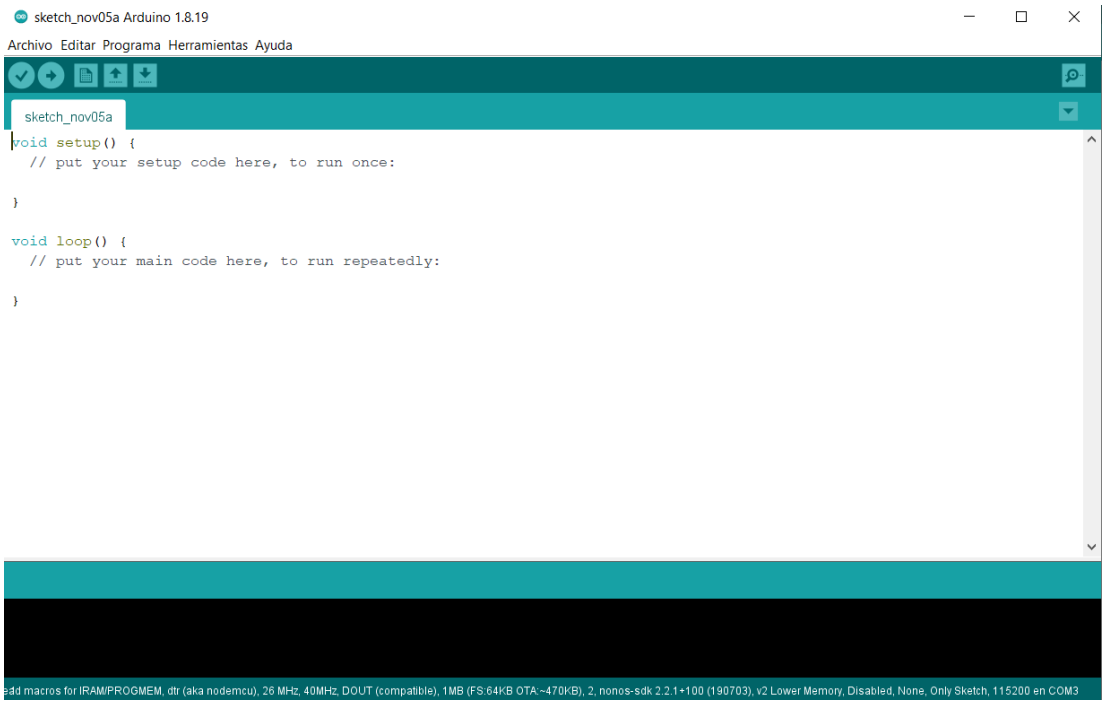


3.1.3.2. Implementación de Software

En el desarrollo de software se empezó programando los microcontroladores. Para esto se puede usar cualquier tipo de editor de texto. En este caso se usó el IDE de Arduino pues nos ofrece un terminal para leer el puerto serial, además de enviar el código fuente (sketch) hacia el controlador. Este IDE usa el lenguaje C++ para la programación de microcontroladores. La figura 30 muestra la interfaz gráfica del IDE mencionado.

Figura 30

Vista del entorno de IDE de Arduino



El sensor BME680 tiene una librería especial, dada por el fabricante BOSCH, para la lectura e interpretación correcta de los valores medidos por este. Esta librería se llama BSEC el cual está disponible para C++. Se decidió utilizar la comunicación I2C porque ocupa menos pines para su configuración. Cuando se usa I2C los dispositivos usan una dirección hexadecimal, en este caso, el BME680 viene con la dirección 0x76 o 0x77. Es necesario elegir la dirección correcta para que exista comunicación. En la figura 31 se muestra la configuración del I2C, que en nuestro caso fue 0x77. En código, la librería lo reconoce como *"BME680_I2C_ADDR_SECONDARY".* El BME680 ofrece múltiples parámetros que puede leer, pero se configuró para que lea solo los que necesitamos los cuales son la temperatura, humedad, presión, CO_2, IAQ e índice de precisión.

Figura 31

Configuración inicial del BME680 en el IDE de Arduino.

```
Wire.begin();             //Inicializacion de I2C
iaqSensor.begin(BME680_I2C_ADDR_SECONDARY, Wire);   //Configuracion de conmunicacion  de sensor BME 680
bsec_virtual_sensor_t sensorList[6] = {
  BSEC_OUTPUT_RAW_PRESSURE,
  BSEC_OUTPUT_IAQ,
  BSEC_OUTPUT_STATIC_IAQ,
  BSEC_OUTPUT_CO2_EQUIVALENT,
  BSEC_OUTPUT_SENSOR_HEAT_COMPENSATED_TEMPERATURE,
  BSEC_OUTPUT_SENSOR_HEAT_COMPENSATED_HUMIDITY
};
iaqSensor.updateSubscription(sensorList, 6, BSEC_SAMPLE_RATE_LP);
```

El sensor DMS501A da dos señales de salida PWM, exactamente no nos daba el valor de PM2.5 (material particulado con un diámetro de 2,5 µm o menos) si no que nos da dos señales, una que detecta la concentración de material particulado con un diámetro > 1 µm y la otra detecta > 2,5 µm. Las señales PWM vienen sin procesar, por lo que debimos crear una función que nos permita obtener el tiempo total en que la señal se encuentra en LOW. Para esto se usó funciones de interrupción (figura 32) que nos permite detectar cuando una señal PWM tiene un flanco de bajada o de subida.

Figura 32

Función de interrupción de la señal PWM (> 1.0 µm)

```
IRAM_ATTR void interrupcion_10()
{
  if (digitalRead(pin10) == LOW)
  {
    tiempo_start_10 = micros();
  }
  else
  {
    tiempo_stop_10 = micros();
    duracion_10 += (tiempo_stop_10 - tiempo_start_10);
  }
}
```

Una vez obtenido el tiempo, se sacó el porcentaje que representa este con respecto al tiempo de muestreo total, este valor fue nombrado “Lowratio, que en este procedimiento fue de 30s. Se utilizó la siguiente fórmula:

$$Lowratio(\%) = \frac{t_{LOW}}{t_m * 10} \qquad (2)$$

Donde:

$Lowratio$ = Relación entre el tiempo en LOW y tiempo total de muestreo en %.

t_{LOW} = Tiempo en que la señal está en LOW en microsegundos.

t_m = Tiempo total de muestreo en milisegundos.

Una vez obtenido el Lowratio tuvimos que convertir dicho valor en unidades de concentración (parts/283mL) que nos proporciona el fabricante mediante una gráfica mostrada en la Figura 17**.** Por lo que aplicamos la formula (1). Todo este procedimiento se realizó para cada una de las dos señalas PWM dada por el sensor DSM501A.

Para hallar el valor real de PM2.5 tuvimos que, mediante programación, restar estos dos valores, así obtuvimos la concentración de material particulado con un diámetro mayor a 1µm y menor a 2,5µm que, en síntesis, es el PM2.5. Se utilizo la siguiente formula:

$$PM2.5(parts/283mL) = Con1 - Con2 \quad (3)$$

Donde:

$PM2.5$ = Concentración de PM2.5 en parts/283mL.

$Con1$ = Concentración de material particulado con diámetro >1um.

$Con2$ = Concentración de material particulado con diámetro >2.5um.

El MINAM nos da los valores máximos permitidos de PM2.5 en unidades de µg/m^3, por lo que tuvimos que buscar la forma de convertirlo.

Consideramos las pequeñas partículas como esferas diminutas y ya que el rango de diámetro de las partículas es de entre 1 y 2,5 µm, tomamos un valor promedio de 1,75 µm de diámetro o 0,875 µm de radio. Con este valor podemos hallar el volumen medio de PM2.5 que se muestra en la siguiente ecuación.

$$V = \frac{4}{3}\pi r^3 \quad (4)$$

Donde:

V = Es el volumen de la partícula PM2.5.

r = Es el radio de la esfera.

Aplicando la formula (4) obtuvimos que el volumen de dichas partículas es de 2,80616 x 10^{-18} m^3. El valor de la densidad de estas pequeñas partículas tiene un valor

promedio de 2 x 10^{12} µg/m^3. Con estos datos hallamos la masa de cada partícula con la siguiente ecuación:

$$m = d * v \quad (5)$$

Donde:

m = Es la masa en µg.

d = Es la densidad de las partículas en µg/m^3.

v = Es el volumen en m^3.

Aplicando la formula (5) obtuvimos que la masa de cada partícula individual es de 5,70908 x 10^{-6} µg. Solo quedaría convertir los 283mL a m^3. La conversión de un metro cubico es de 10^6 mL. Agregando esta operación a lo antes hallado tendríamos la formula final siguiente:

$$PM2.5(\mu g/m^3) = PM2.5(parts/283mL) * 0{,}02017 \quad (6)$$

Donde:

$PM2.5(ug/m^3)$ = Concentración de PM2.5 en µg/m^3.

$PM2.5(parts/283mL)$= Concentración de PM2.5 en parts/283mL.

$0{,}02017$ = Constante de conversión de parts/283mL a µg/m^3.

Todo el cálculo para el sensor DSM501A se programó en el código fuente que está en el Anexo 9.

Con los datos ya procesados se procedió a programar los ESP-07 con la configuración tipo Malla. Para esta configuración se utilizó la librería PainLessMesh que nos permite realizar una configuración tipo Malla compatible con los dispositivos ESP-07, además de distintas funciones de transmisión, recepción y detección de nuevos nodos. En cada nodo de la Malla se debe de configurar un mismo nombre de la red, una misma contraseña y un mismo puerto. Los valores de los parámetros mencionados se pueden ver en la figura 33.

Figura 33

Configuración de parámetros para la red Malla

```
//####--- MALLA ---####-------------------------
#define   NOMBRE_MALLA     "RedSensor"
#define   CONTRASEÑA_MALLA   "2015052397"
#define   PUERTO_MALLA       5555
```

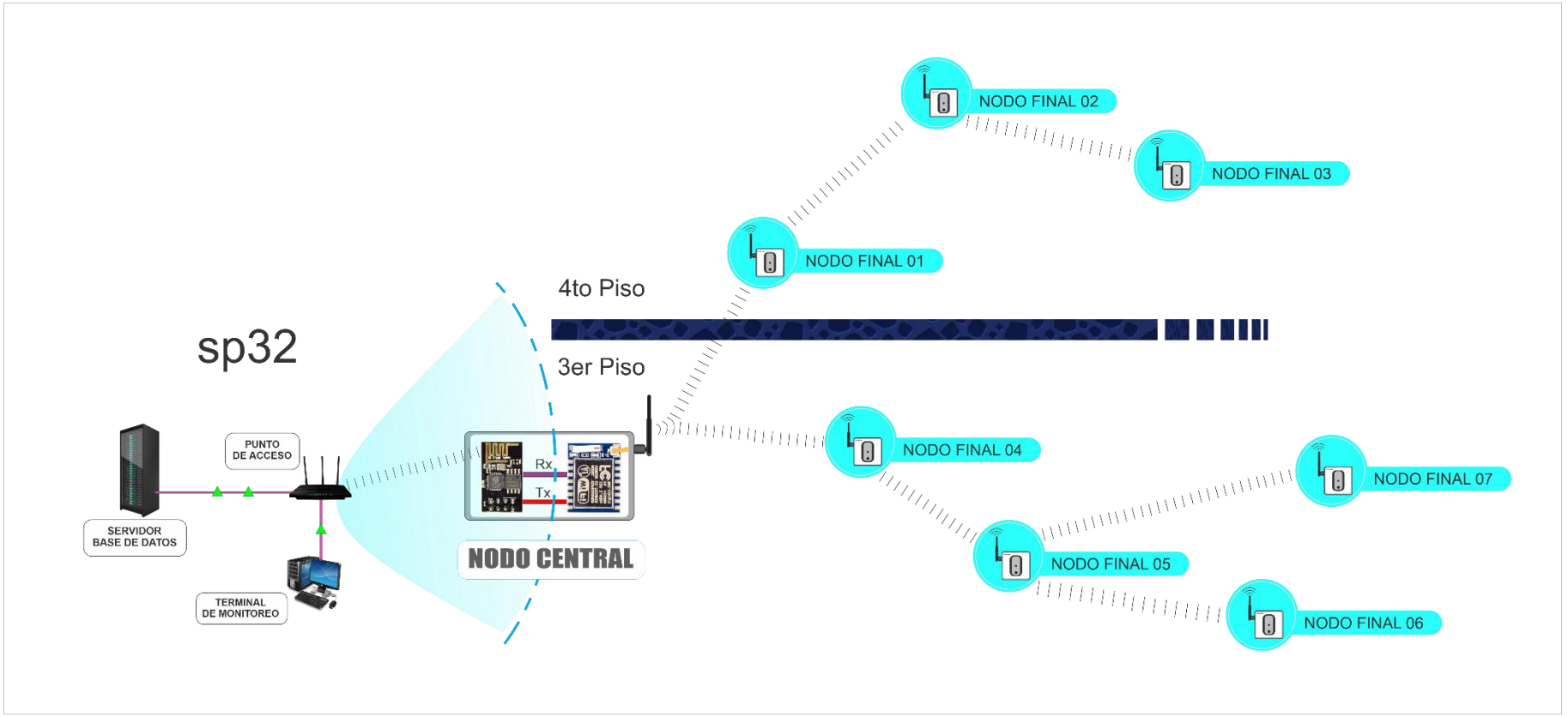
Nota. El código fuente completo está en el Anexo 10 y 11.

Cuando un nuevo dispositivo se une a la malla, toda la topología se actualiza para configurarse a su mayor rendimiento posible. Esta red Malla debía de comunicarse con la red WLAN por lo que se programó un nodo Central el cual conto con 2 adaptadores de red (uno para cada red), y la comunicación entre estos a través de la interfaz serial. La topología final se muestra en la figura 34.

El ESP8266 (que conecta con la red WLAN) en el nodo central se configuró con las credenciales SSID y contraseña de la red WLAN. Es posible guardar más de una credencial por si el dispositivo tiene que conectarse a distintas redes. Las credenciales de las redes de prueba pueden verse en la figura 35.

Figura 34

Topología final del proyecto



Nota. Nótese como se integra las dos redes a través del nodo central.

Figura 35

Configuración de distintas redes WLAN

```
ESP8266WiFiMulti wifi_multi;

const char* ssid1 = "SALMOS 2.4GHZ";
const char* pass1 = "0799376060";

const char* ssid2 = "tesis";
const char* pass2 = "12345678";

const char* ssid3 = "BOB TORONJA";
const char* pass3 = "2015052397";
```

Nota. El código fuente completo está en el Anexo 12.

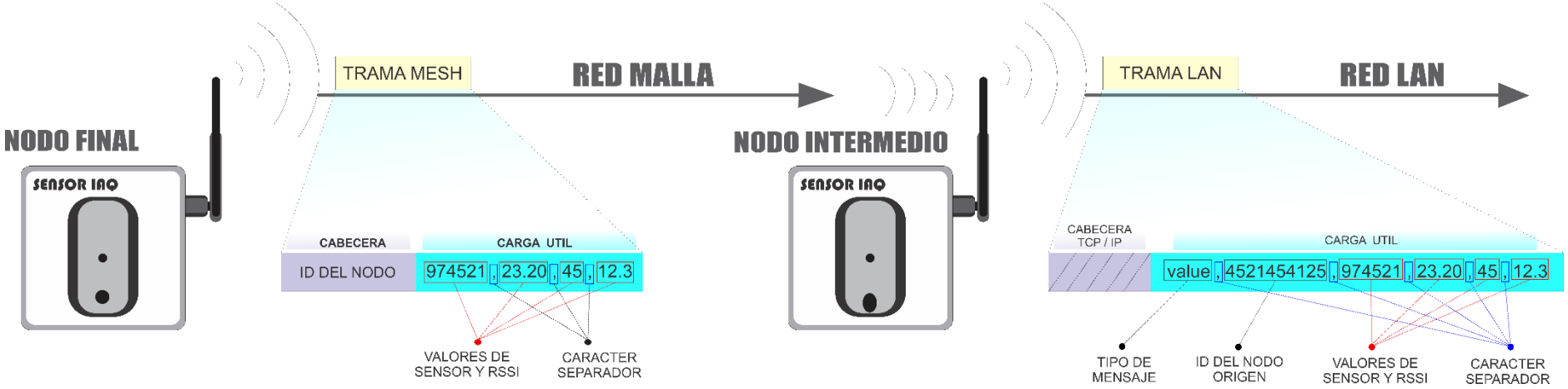
Se configuro 2 tipos de datos enviados a través de la red Malla, los mensajes que contengan valores de parámetros de los sensores (como por ej. Temperatura, presión, etc.) y también potencia de señal (RSSI), al cual se le puso el prefijo “value” además de un carácter separador “,” (ya que para separar decimales se usó “.”); y el segundo es los mensajes que contengan avisos de cambios en la red, es decir que contengan información de cuando un nodo se ha integrado o quitado de la red Malla enviando como mensaje los nodos conectados luego del cambio. El prefijo para este último mensaje fue “network” y se usó el mismo carácter separador.

Los valores de los sensores son enviados por el nodo final y al llegar al nodo central, este les agrega el prefijo del tipo de dato (“value” o “network”) y también el id del nodo para reconocer el origen.

Los dispositivos de la red Malla, si bien usan el chip Wifi del ESP-07, no usa direcciones IP para la identificación y comunicación, por lo tanto, no usa la pila de TCP/IP; usa en su lugar un propio protocolo con identificadores de nodos creados a partir de su dirección MAC. El flujo de la data desde el sensor hacia la red WLAN se muestra en la figura 36.

Figura 36

Estructura del paquete de datos en las dos redes

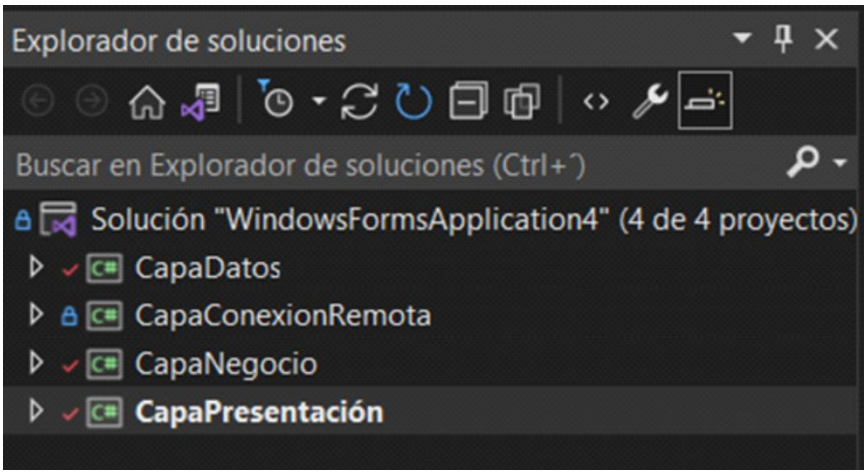


3.1.3.3. Desarrollo del software de monitoreo

El software de monitoreo se realizó en Visual Studio, utilizando WinForms y EntityFramework. Como se muestra en la figura 37, se utilizó una arquitectura de programación de 4 capas (parecido al modelo OSI).

Figura 37

Estructura del desarrollo de software en capas.

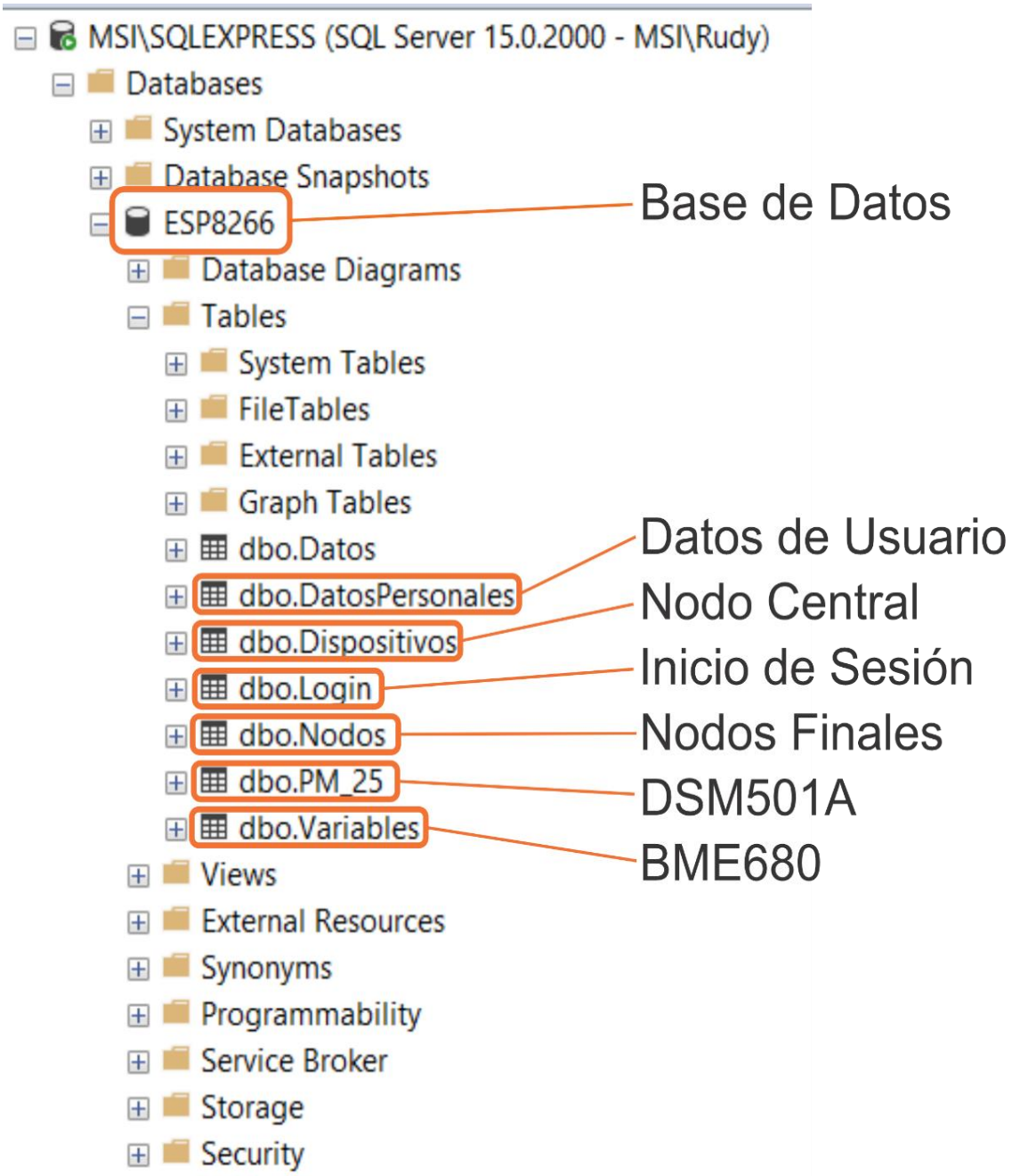


Como base de datos se decidió usar SQL Server que es la más común y compatible con lo anterior mencionado. Para el diseño de la base de datos se usó Microsoft SQL Server Management Studio 18. La base de datos se llamó ESP8266 y dentro de esta se creó 6 tablas para almacenar los distintos datos. En la figura 38 se muestra la estructura de la Base de Datos, así como las tablas de datos.

Para vincular la base de datos con nuestro software usaremos la primera capa, la Capa de Datos. Esta capa crea una entidad del mismo formato que la base de datos original, haciendo que no haya problemas de incompatibilidad con los tipos de datos existentes. Nótese en la figura 39 que la capa de datos tiene la misma estructura que la base de datos original de la figura 38.

Figura 38

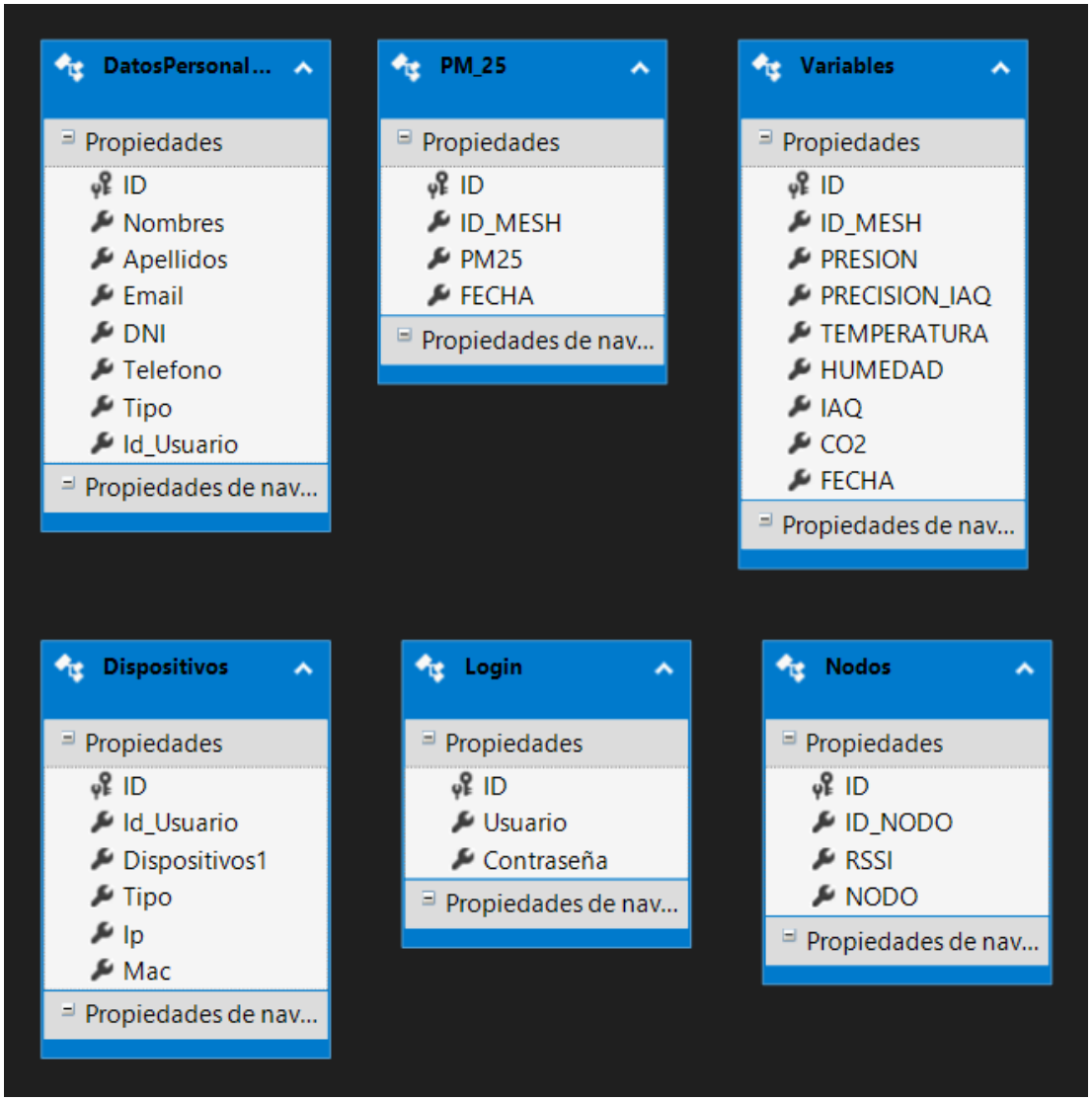
Estructura general de la Base de Datos



La segunda capa, la capa de conexión Remota. Esta capa es puramente código fuente y se encargó de recibir la data del nodo Intermedio, separarla y almacenarla en la tabla correspondiente en la base de datos. El código fuente completo está en el Anexo 8.

Figura 39

Capa de Datos en Visual Studio.



La tercera capa, la capa de Negocio es también puramente código y se encargó de comunicar la Capa de Datos y la Capa de Presentación, esto a través de consultas usando EntityFramework. El código fuente completo está en el Anexo 7.

La última capa, la Capa de Presentación, se usó para crear cada ventana de la aplicación. Empezamos por la ventana de Inicio de Sesión (figura 40), se decidió poner un sistema de Login para tener una capa extra de seguridad y que solo un usuario autorizado pueda ver la data.

Figura 40

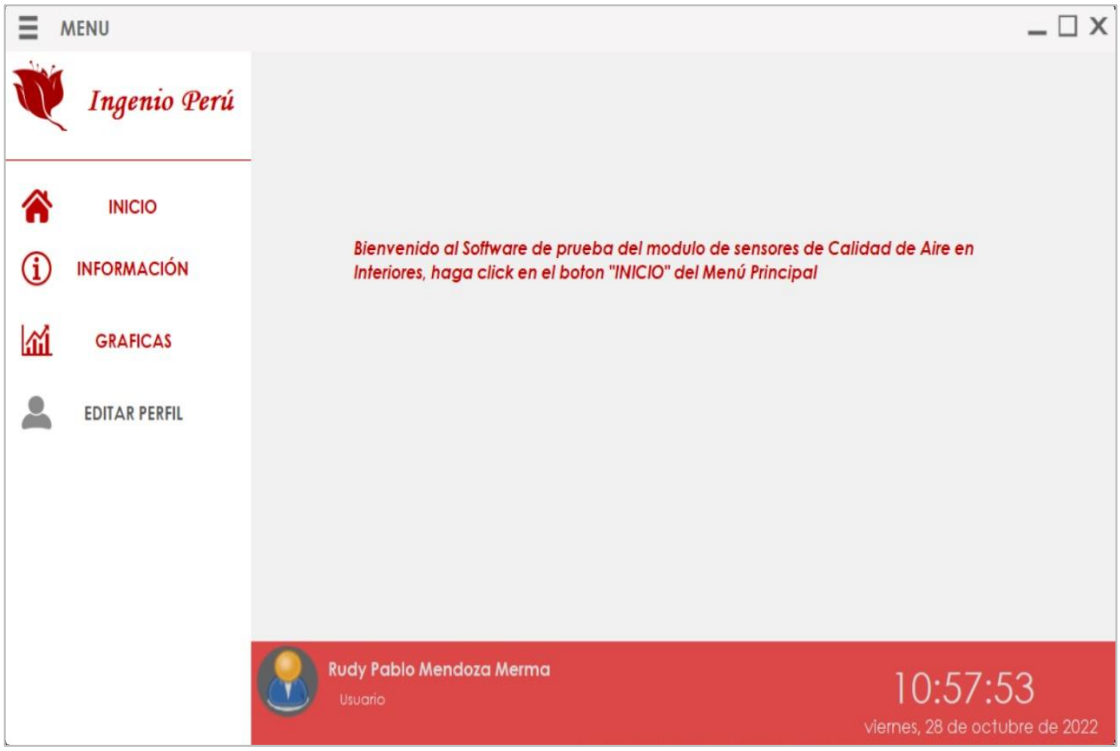
Ventana de Inicio de Sesión.

Nota. El código fuente completo está en el Anexo 1.

Luego de del inicio de sesión tenemos la ventana principal del software (figura 41) con un menú lateral con todos los ítems principales de este y un panel inferior con los datos del usuario.

Figura 41

Ventana Principal

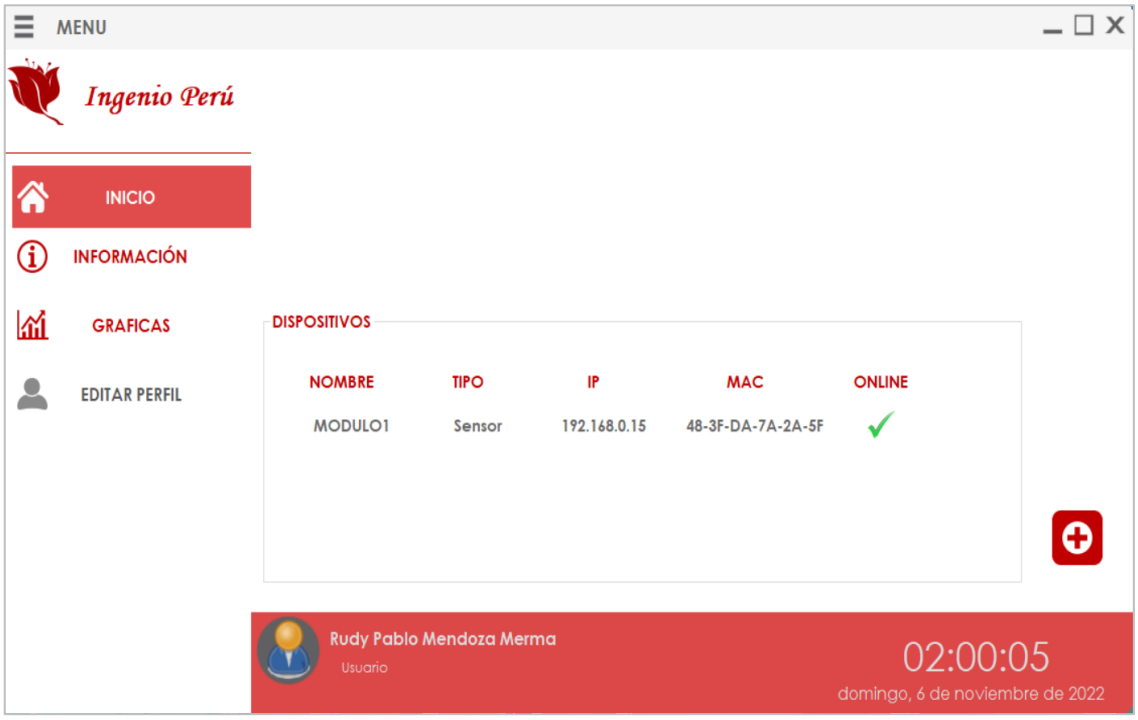


Nota. El código fuente completo está en el Anexo 2.

Luego tenemos la ventana de INICIO (figura 42) que contiene los dispositivos vinculados o disponibles dentro de la red WLAN, es decir, el Nodo Intermedio de la red Malla. Además, contiene información de si está conectada a la Red y un botón para escanear la red (en el caso de no estar vinculado).

Figura 42

Ventana de Inicio



Nota. El código fuente completo está en el Anexo 3.

Si no se encuentra el dispositivo aun vinculado, podemos hacer un escaneo para buscarlo dentro de nuestra red WLAN. Para eso tenemos el botón +, el cual nos abre una nueva ventana (figura 43). Una vez encontrado nos da la opción de vincularlo.

Figura 43

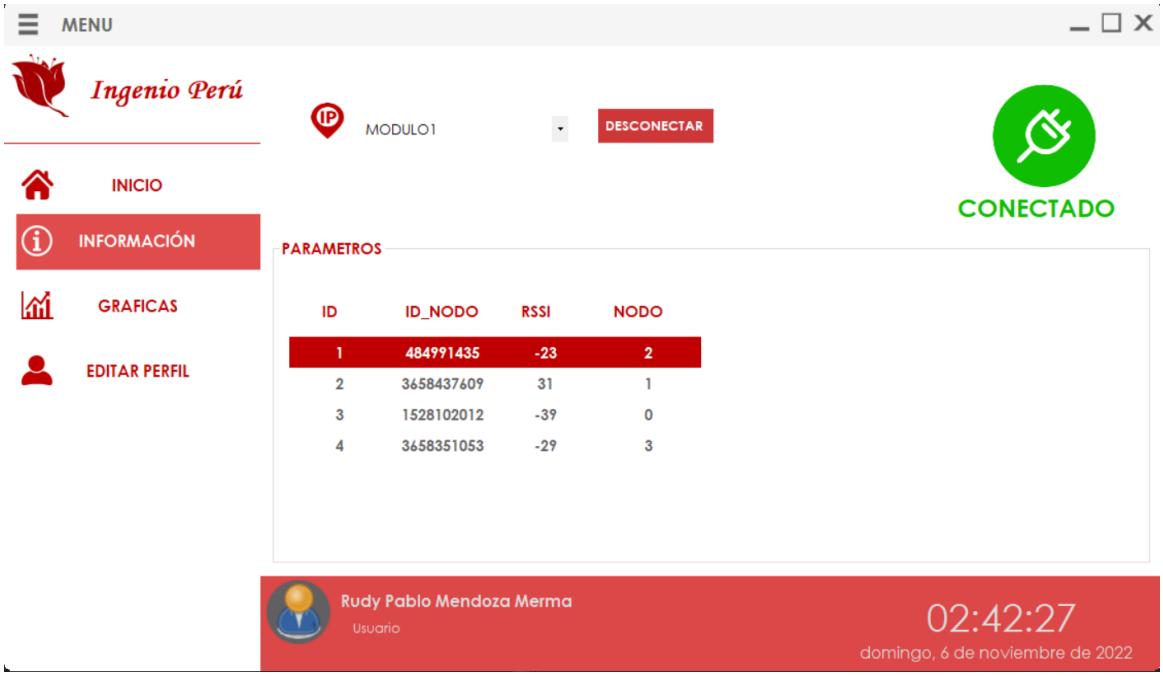
Ventana de Agregar Dispositivo.

Nota. El código fuente completo está en el Anexo 4.

Una vez se vinculó el dispositivo, tenemos que conectar y verificar los Nodos Finales asociados a ese dispositivo (que es el nodo central). Además de ver la calidad de la señal de estos (RSSI). Es por eso por lo que se creó una ventana de información (figura 44) que nos ofrece información de red de los nodos finales y central.

Figura 44

Ventana de Información de Nodos

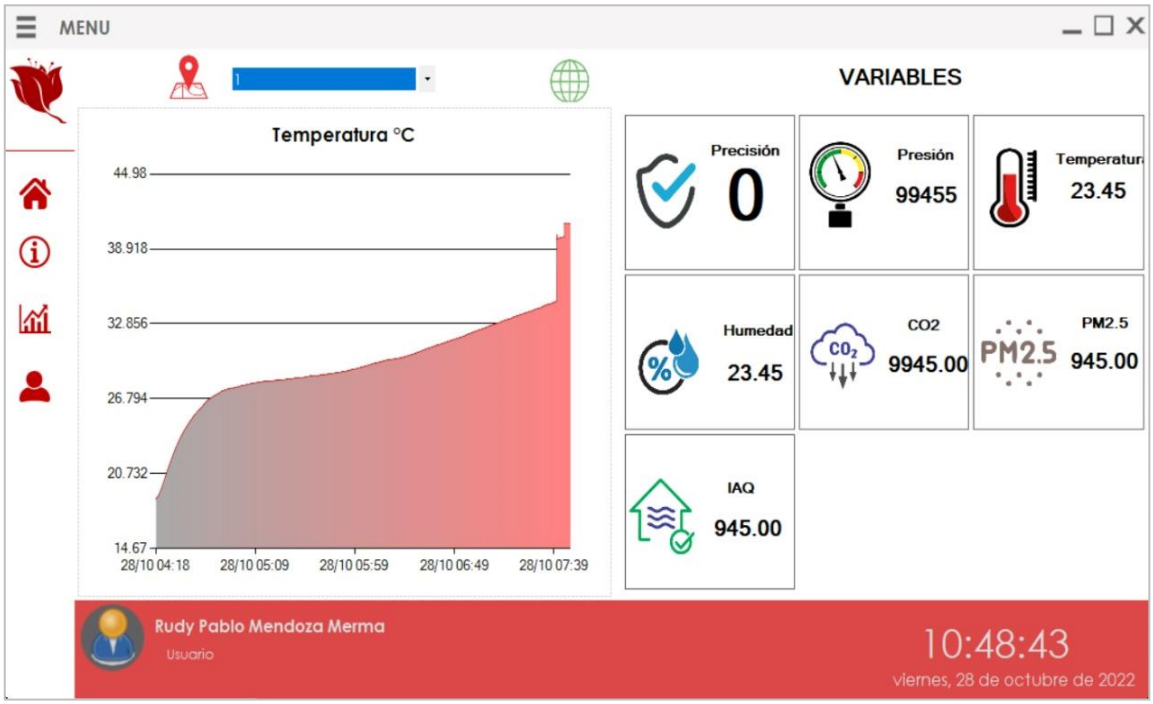


Nota. El código fuente completo está en el Anexo 5.

Una vez que tenemos monitoreado los parámetros de red, procedemos a graficar y mostrar los parámetros medidos de Calidad de Aire. La visualización de datos se hace a través de una gráfica y cuadros de texto debidamente ordenado. La interfaz gráfica para la visualización se muestra en la figura 45.

Figura 45

Ventana de Gráficas de parámetros de Calidad de Aire



Nota. El código fuente completo está en el Anexo 6.

3.1.3.4. Despliegue de la red

El montaje y despliegue de la red Malla se realizó en las instalaciones de la Escuela Profesional de Ingeniería Electrónica de la UPT. LA dirección es en el Campus Capanique, Av. Jorge Basadre Grohmann s/n –Pocollay, Tacna – Tacna – Pocollay. Geográficamente se ubica a 18°00’23’’ latitud Sur y 70°13’37’’ Latitud Oeste. La figura 46 muestra una imagen satelital de las instalaciones de la Universidad Privada de Tacna.

Figura 46

Ubicación de la zona de Despliegue de la red Malla

Nota. Fuente: Google Earth.

Como se conectó anteriormente, se crearon 3 nodos finales y un nodo central; los cuales fueron ubicaciones en:

a. Nodo Central: Sala de profesores

b. Nodo Final 01: Laboratorio de máquinas eléctricas

c. Nodo Final 02: Aula EPIS

d. Nodo Final 03: Aula APIE

Además de eso se tiene un servidor de base de datos y un terminal de monitoreo. Por temas de presupuesto, ambos se configuraron en el terminal de monitoreo, es decir que el terminal de monitoreo a su vez tiene montado una instancia de base de datos para el almacenamiento.

Para la ubicación exacta de los nodos finales (sensores) dentro de los ambientes antes mencionados se consideró, como principal condición, la corriente de aire y la naturaleza de los gases. Los gases contaminantes habituales (como el CO_2) tienen una densidad más baja que el aire por lo que tienden a subir. El ingreso de la corriente del aire en su mayor porcentaje es por la puerta de ingreso creando en los vértices de las esquinas mayor fluido de aire y mayor registro de los productos y gases contaminantes. Esto nos da una idea del posicionamiento de los nodos. Se pensó conveniente ubicar

los nodos a 2 metros del suelo en la esquina superior del aula. La ubicación de los nodos se puede apreciar gráficamente en la figura 47 y 48.

Figura 47

Ubicación de los nodos en el tercer piso bloque dos

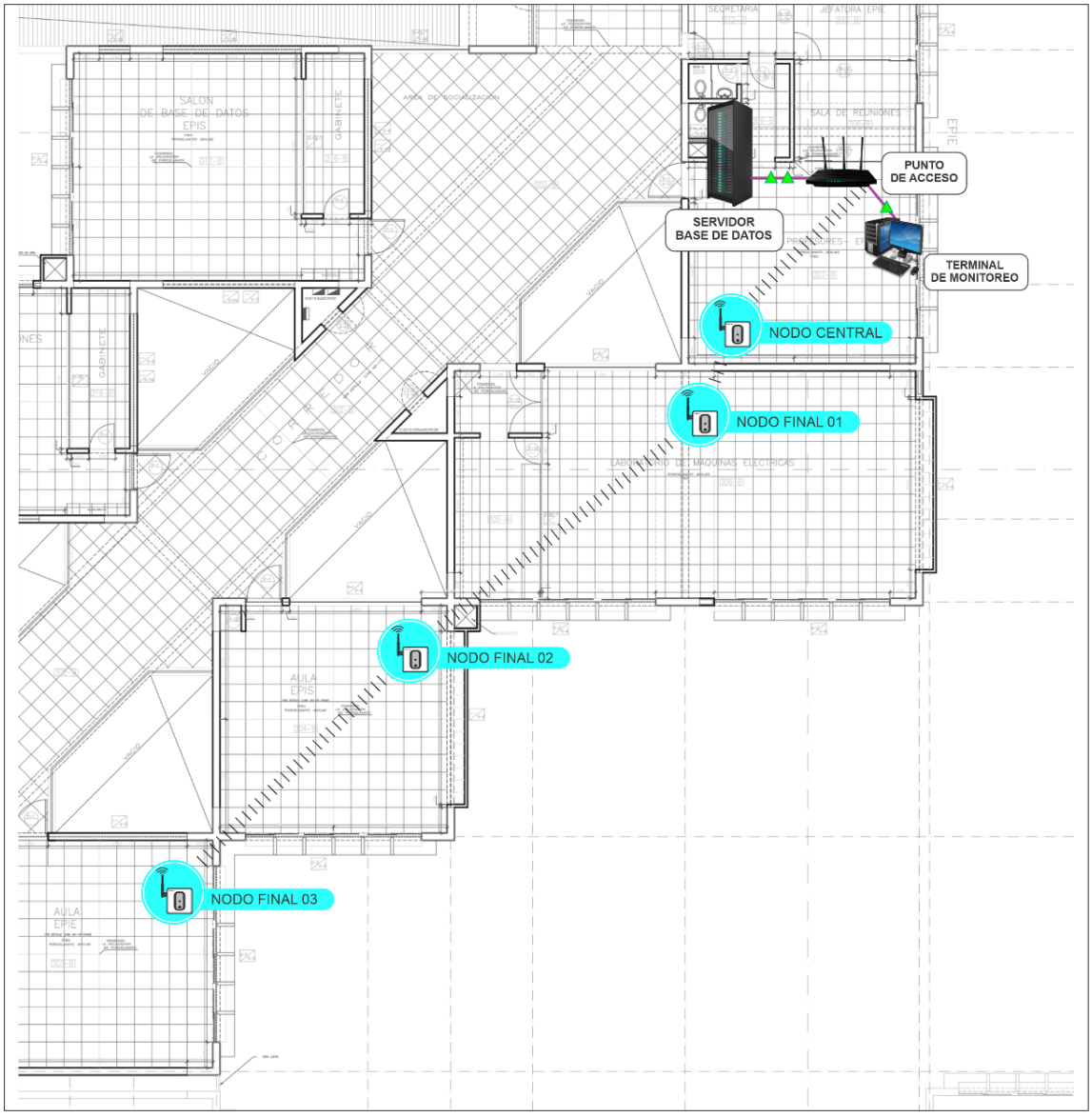
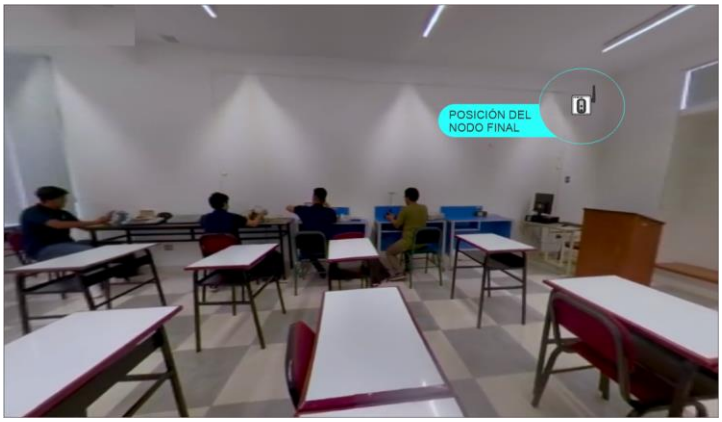


Figura 48

Vista de la ubicación del nodo final dentro del aula



Nota. Fuente: Plataforma virtual de la Universidad Privada De Tacna

3.2. Acciones y Actividades

Tabla 14

Cronograma de Actividades

ACTIVIDADES	SEMANAS POR MESES															
	SEPTIEMBRE				OCTUBRE				NOVIEMBRE				DICIEMBRE			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
INICIO																
Redacción de titulo	x															
Elementos del proyecto	x	x														
Objetivos de la Investigación	x	x														
Justificación e Importancia			x													
DESARROLLO																
Revisión de la Bibliografía			x													

Elaboración del Marco Teórico	x	x												
Diseño del sistema			x	x	x	x								
Implementación del prototipo						x	x	x						
Pruebas								x	x	x				
Corrección de posibles errores										x				
CIERRE														
Redacción del Borrador Trabajo Final											x			
Revisión y Corrección del Trabajo Final												x		
Entrega del Trabajo Final													x	
Sustentación del Trabajo Final														x

3.3. Materiales y/o instrumentos

3.3.1. Materiales

a. Estaño:

La soldadura o estaño es el complemento del cautín para soldar los equipos electrónicos se aplica estaño sin plomo ya que este tiende a ser nocivo para el medio ambiente y nuestra salud.

b. Alicate de corte fino:

Usado para cortar sobrantes en las uniones de la placa.

c. Cajas de paso cuadradas:

Caja de paso de plástico de PVC usado como contenedor de los sensores y microcontroladores.

d. Placas de trupán:

   Trupán MDF de 10 cm x 7 cm de 3mm de espesor, usado en el encapado en la caja de paso por motivos estéticos.

e. Terokal:

   Se usó un Terokal en chisguete 50ml para la unión de la caja de PVC y el MDF.

f. Adhesivo de 10 x 10 cm:

   Usado en la parte frontal de la caja de paso de plástico de PVC para poder identificar donde están ubicados los orificios y el nombre de “SENSOR IAQ”

g. Interruptor On-Off:

   Usado en la parte externa para cortar la energía principal si es necesario.

h. Cableado:

   Cable calibre 22 AWG para conexiones entre puntos de transmisión de toma de energía y de la placa al interruptor On-Off.

3.3.2. Instrumentos

a. Multímetro:

   Usado para la verificación de la continuidad del cableado y la buena continuidad entre los sensores y los microcontroladores.

b. Pistola de temperatura

   Usado para medir la temperatura generada por los microcontroladores y los sensores y poder discriminar ese dato en la medición de la temperatura tomada del ambiente.

c. Cautín 60wtts.

   Usado para la unión con estaño entre los puntos de transmisión y la placa impresa.

d. Laptop

   Usado para implementación del software y programación, la laptop usada es de marca MSI con un RAM de 16gb, SSD 1tera, Procesador COREI7 11800h.

e. Cable-USB 3.0

Usado para la transmisión de datos en la programación de los módulos y pruebas de finales de una implementación correcta, el tipo de cableado entre el pc y los módulos es cable xtech USB 3.0 macho a micro USB macho b, 90cm.

f. Taladro inalámbrico

Usado para la perforación de la estructura del módulo y algún punto en la impresión de la placa que se desea definir

g. Brocas #12:

Usado en las perforaciones de las Caja de paso de plástico de PVC y trupan, así logramos los orificios para la toma de datos en los sensores y el punto de salida del cable de poder.

h. Placa de circuito impreso

Usada como soquete en la unión de los microcontroladores, sensores y tomas de poder, la placa es de baquelita diseñada he impresa

3.4. Población y/o muestra de estudio

3.4.1. Población

No aplica.

3.4.2. Muestra

No aplica.

3.5. Operacionalización de las Variables

En la Tabla 15 se detallan las variables de la investigación, así como sus dimensiones e indicadores.

Tabla 15

Cuadro de Operacionalización de las Variables

Variable	Definición Conceptual	Dimensiones	Indicadores
Independiente Sistema IoT	Son sistemas electrónicos que nos permiten monitorizar la calidad del aire a través de un sistema inalámbrico de bus de datos controlado a distancia.	• Red Mesh	• Procesamiento y transmisión de datos
		• Almacenamiento	• Base de Datos.
		• Visualización	• Interfaz Gráfica
Dependiente Monitoreo de Calidad del Aire	Es medir la cantidad de gases contaminantes presentes en un ambiente cerrado.	• Monitoreo	• Cantidad de Gases Contaminantes • Temperatura • Humedad • PM2.5

3.6. Procesamiento y análisis datos

3.6.1. Base de datos

Para almacenar correctamente los datos ofrecidos por lo sensores, se creó una serie de tablas en una base de datos. Estas tablas estructuran los datos de los parámetros por columnas y por filas. Otra característica importante es que cada columna se le debió configurar un correcto tipo de dato. Debido a que los variables no son números enteros sino decimales, se decidió dar el tipo de variable “decimal” para cada columna de tipo de parámetro. También de eso también se tiene una columna denominada FECHA con el tipo de dato “datetime”, esto nos permitió almacenar cada variable junto a la hora y fecha al que fue capturado. Una columna adicional es la de ID_MESH el cual es el ID del dispositivo final del cual proviene el dato, eso nos permitió diferenciar los valores de

parámetros de cada nodo para sus respectivas graficas. La figura 49 y 50 nos muestra la estructura mencionada.

Figura 49

Estructura de la tabla de variables BME680

Column Name	Data Type	Allow Nulls
ID	int	☐
ID_MESH	nvarchar(50)	☑
PRESION	decimal(8, 2)	☑
PRECISION_IAQ	int	☑
TEMPERATURA	decimal(5, 2)	☑
HUMEDAD	decimal(6, 2)	☑
IAQ	decimal(5, 2)	☑
CO2	decimal(6, 2)	☑
FECHA	datetime	☑
		☐

Figura 50

Estructura de la tabla de variables DSM501a

Column Name	Data Type	Allow Nulls
ID	int	☐
ID_MESH	nvarchar(50)	☑
PM25	decimal(5, 2)	☑
FECHA	datetime	☑
		☐

También se configuro para que, si dos datos seguidos son iguales, no se guarde en la base de datos. La cantidad de datos que se tomó en la tabla de variables del BME680 fue de 16061 registros de cada variable y de la tabla de DSM501a se registró 1566 registros, todo esto en el transcurso de un día. La diferencia tan grande entre la cantidad de registros de ambas tablas se debe a que el sensor DSM501a toma una muestra cada 30 segundos, en cambio el BME680 toma una muestra cada 2 segundos aproximadamente. Una pequeña muestra de los registros se ve en la figura 51 y 52.

La versión del servidor de base de datos SQL Server es la versión “Express”, esta nos da un almacenamiento limitado de 10 GB que es más que suficiente para el propósito de este proyecto.

Figura 51

Resultado de los primeros 10 registros de la tabla de BME680

Results Messages

	ID	ID_MESH	PRESION	PRECISION_IAQ	TEMPERATURA	HUMEDAD	IAQ	CO2	FECHA
1	1	3658437609	95727.00	0	18.67	52.36	25.00	500.00	2022-10-28 04:18:59.023
2	2	3658437609	95727.00	0	18.68	52.32	25.00	500.00	2022-10-28 04:19:01.983
3	3	3658437609	95727.00	0	18.69	52.30	25.00	500.00	2022-10-28 04:19:04.990
4	4	3658437609	95727.00	0	18.70	52.26	25.00	500.00	2022-10-28 04:19:08.047
5	5	3658437609	95727.00	0	18.71	52.29	25.00	500.00	2022-10-28 04:19:11.000
6	6	3658437609	95727.00	0	18.71	52.34	25.00	500.00	2022-10-28 04:19:14.007
7	7	3658437609	95729.00	0	18.72	52.34	25.00	500.00	2022-10-28 04:19:17.017
8	8	3658437609	95727.00	0	18.73	52.30	25.00	500.00	2022-10-28 04:19:20.023
9	9	3658437609	95729.00	0	18.75	52.23	25.00	500.00	2022-10-28 04:19:22.977
10	10	3658437609	95727.00	0	18.75	52.19	25.00	500.00	2022-10-28 04:19:25.980

Figura 52

Resultado de los primeros 10 registros de la tabla de DMS501a

Results Messages

	ID	ID_MESH	PM25	FECHA
1	1	3658437609	291.93	2022-10-28 04:19:26.133
2	2	3658437609	306.76	2022-10-28 04:19:56.047
3	3	3658437609	270.46	2022-10-28 04:20:26.107
4	4	3658437609	267.44	2022-10-28 04:20:56.023
5	5	3658437609	280.51	2022-10-28 04:21:26.087
6	6	3658437609	228.24	2022-10-28 04:21:56.200
7	7	3658437609	364.81	2022-10-28 04:22:26.120
8	8	3658437609	229.31	2022-10-28 04:22:56.137
9	9	484991435	6.54	2022-10-28 04:23:23.213
10	10	3658437609	300.39	2022-10-28 04:23:26.120

3.6.2. Rendimiento de la red

Los datos de rendimiento de la red se basan a partir del RSSI de cada dispositivo, además del tamaño de las tramas para verificar el ancho de banda. Los datos del RSSI no se almacenaban, solo se mostró en tiempo real el valor instantáneo para cada nodo; eso nos permitió evaluar y monitorear el estado de las conexiones. En la figura 53 se muestra una captura de los valores de RSSI.

Figura 53

Valores de RSSI para cada nodo en tiempo real

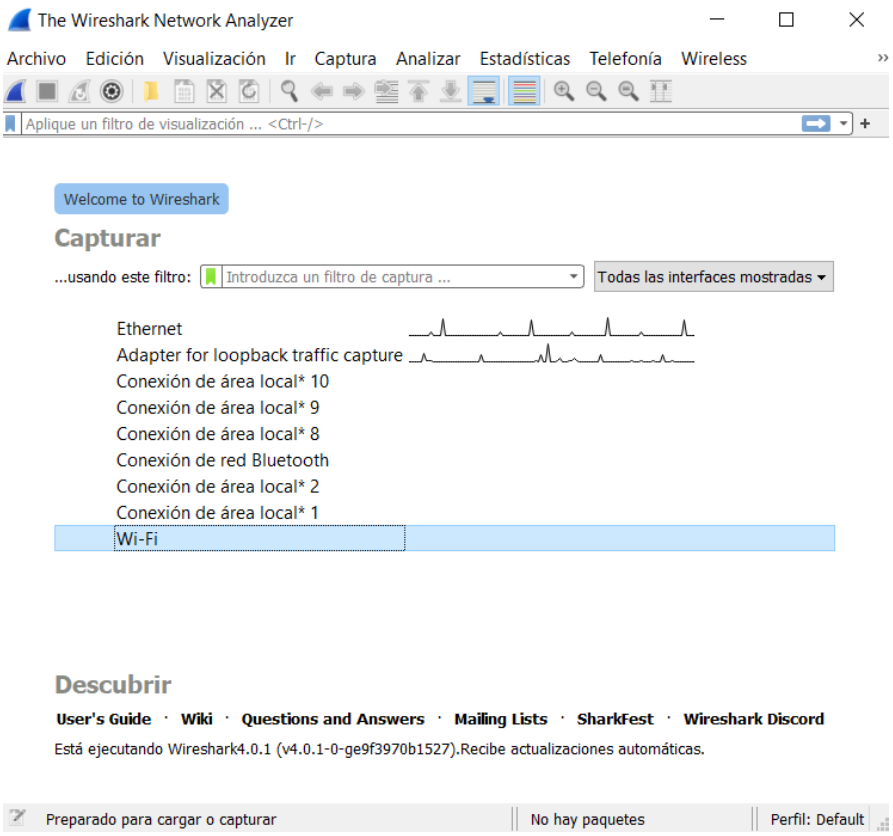
ID	ID_NODO	RSSI	NODO
1	484991435	-23	2
2	3658437609	31	1
3	1528102012	-39	0
4	3658351053	-29	3

Nota. Los valores positivos están configurados como AP.

El análisis de la trama de datos se hizo a los paquetes de la red WLAN a través de Wireshark. Esto nos permite capturar todos los paquetes de datos entrantes y/o salientes de cada adaptador de red existente en el pc de monitoreo.

Figura 54

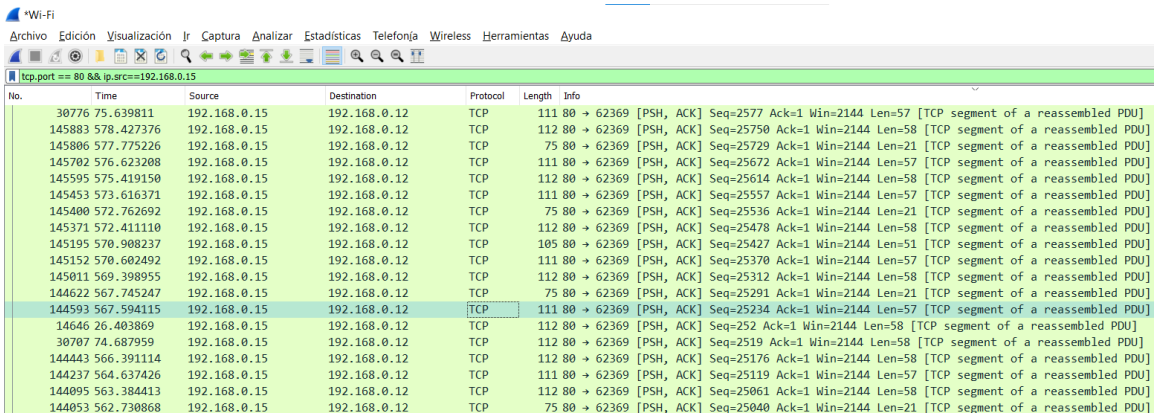
Pantalla inicial de Wireshark



Lo primero fue seleccionar el adaptador de red que se usó en ese instante, en este caso Wi-fi. Una vez seleccionado se procede a realizar la captura. La captura realiza el registro de todo el tráfico, eso incluye datos de entrada y salida que cualquier dispositivo en cualquier puerto. Se pudo filtrar los datos seleccionando el protocolo (en nuestro caso TCP) y la dirección IP del nodo central, que es el que envía los datos. Así filtramos para poder analizar cada paquete de datos. Todo esta se puede comprobar en la figura 54 y 55.

Figura 55

Captura de paquetes de datos del adaptador Wi-fi en Wireshark



Nota. En el cuadro de texto de la parte superior esta puesto el filtro para paquetes del puerto 80 con una dirección de origen que corresponde al nodo central.

CAPÍTULO IV: RESULTADOS

4.1. Calidad de la conexión

La calidad de las conexiones se midió tomando el RSSI de cada uno de los nodos y mostrando sus variaciones en tiempo real. En la tabla 16 se muestra el RSSI de cada nodo, la unidad de medida es en dBm.

Tabla 16

Captura del valor de RSSI en cada nodo de la Malla

N° de Nodo	Nodo ID	RSSI (dBm)
0	1528102012	-56
1	3658437609	31
2	484991435	-48
3	3658351053	-40

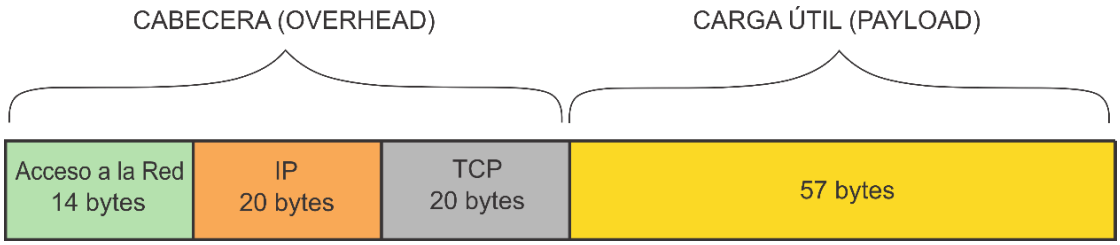
Los niveles de RSSI están en el rango de excelente y muy bueno según la tabla 3. Se comprobó que los dispositivos que se encuentren con un RSSI positivo están configurados como AP dentro de la Malla. Además de eso el ancho de banda disponible en la conexión es de 26 Mbits/s o 3.25 MB/s.

4.2. Tamaño de la trama

La captura de las tramas entrantes del nodo intermedio hacia la base de datos se realizó con el software Wireshark. Se tuvo tramas con un tamaño de 75; 111 y 112 bytes. Esta variación se debió a que los valores de los sensores no siempre son lo mismo y que no siempre se envía todos los parámetros en una misma trama. Para el análisis de la trama se tomó el de 111 bytes ya que es el más común. La cabecera ocupa un total de 54 bytes y la carga útil es de 57 bytes (figura 56). El tamaño de la carga útil de nuestra trama tuvo menos tamaño que el tamaño máximo de segmento TCP (que por defecto es 536 bytes de carga útil), esto significa que no tuvo que ser necesario fragmentar la data. Esto hace que el ancho de banda sea más que suficiente para tener un rendimiento optimo en la velocidad de conexión de la malla y el nodo central.

Figura 56

Trama recibida con tamaño de 111 bytes

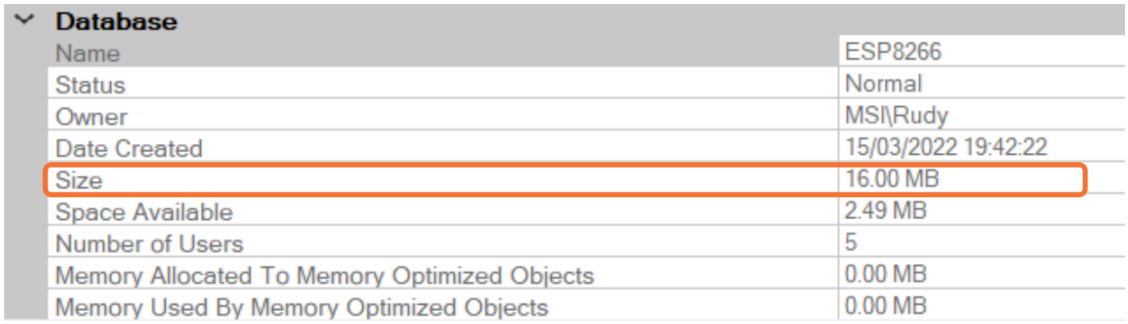


4.3. Base de Datos

Al utilizar la versión Express (gratis) de SQL Server, esta nos ofrece un almacenamiento total de solo 10 GB. El cual es un valor limitado en comparación con base de datos reales. Poniéndonos en el contexto de nuestro proyecto, se almacenó un total de 54696 registros teniendo dos tablas separadas (uno para cada sensor) en el transcurso de un día. Esta cantidad de datos ocupó un tamaño de 80,00 MB en el disco duro de la laptop. Teniendo en cuenta que contamos con 10GB de uso libre, tendríamos la capacidad de almacenar aproximadamente 21 meses de corrido. La información general se puede observar en la figura 57.

Figura 57

Información general de la Base de Datos



4.4. Parámetros de calidad de aire

Para una mejor visualización de la variación de los datos a través del tiempo, se realizó una gráfica de líneas por cada parámetro y para cada nodo (3 en total).

4.4.1. Presión

La presión se midió en hectopascales (hPa). Los valores de presión deberían ser constantes, pero varían ligeramente debido a la incertidumbre del sensor. A

continuación, se muestran el comportamiento de la presión durante un día entero en los 3 distintos nodos (figura 58, 59 y 60 respectivamente).

Figura 58

Gráfica de presión en el nodo final 1

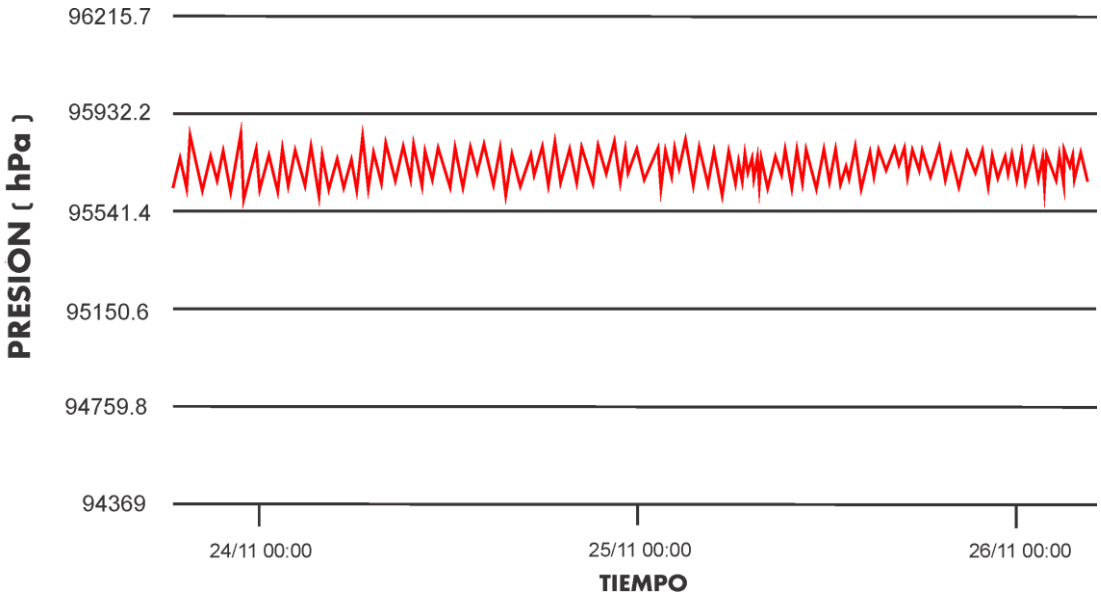


Figura 59

Gráfica de presión en el nodo final 2

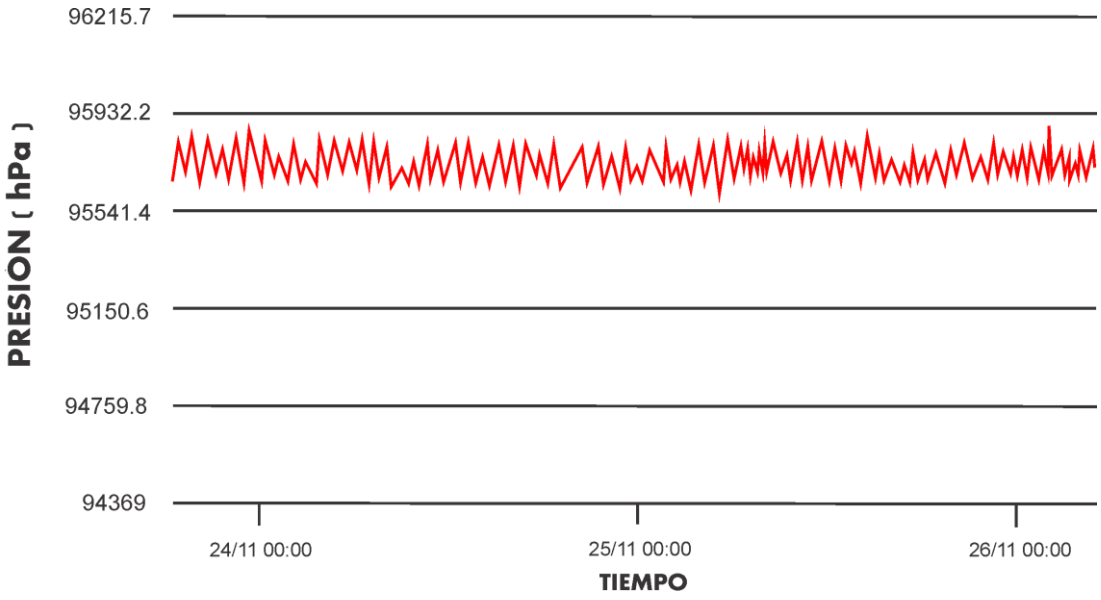
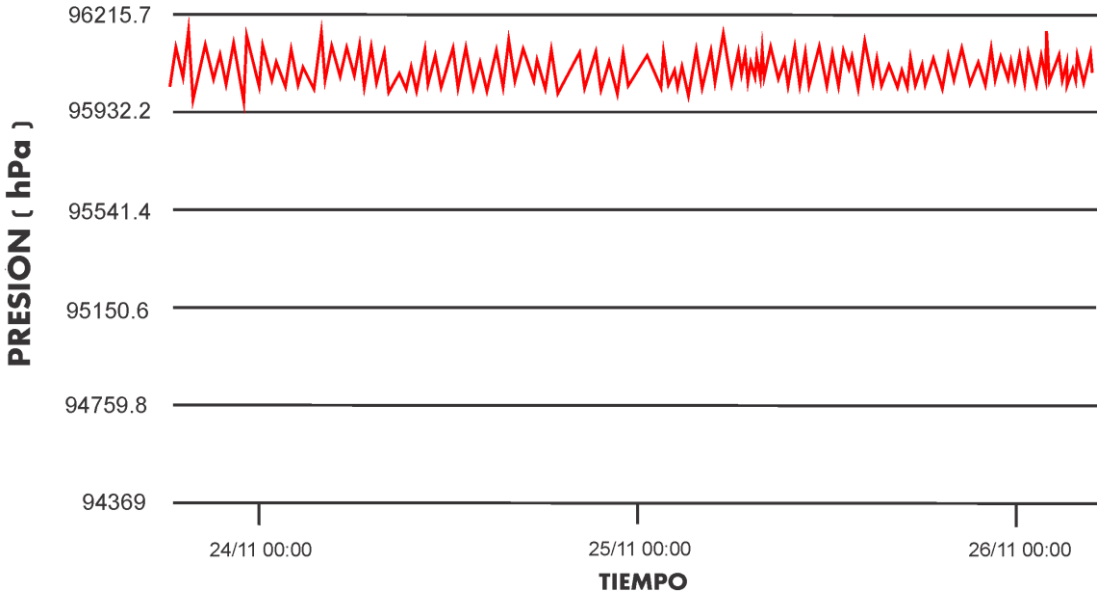


Figura 60

Gráfica de presión en el nodo final 3



4.4.2. Temperatura

La temperatura se midió en grados Celsius (°C). Los valores tuvieron un comportamiento de campana por la variación de temperatura debido al sol. Esto hace que en al inicio y final del día tenga una temperatura más baja en comparación con el resto del día que sube gradualmente teniendo el pico más alto en el medio día aproximadamente. A continuación, se muestra el comportamiento de la temperatura en cada nodo (figura 61, 62 y 63 respectivamente).

Figura 61

Gráfica de temperatura en el nodo fina 1

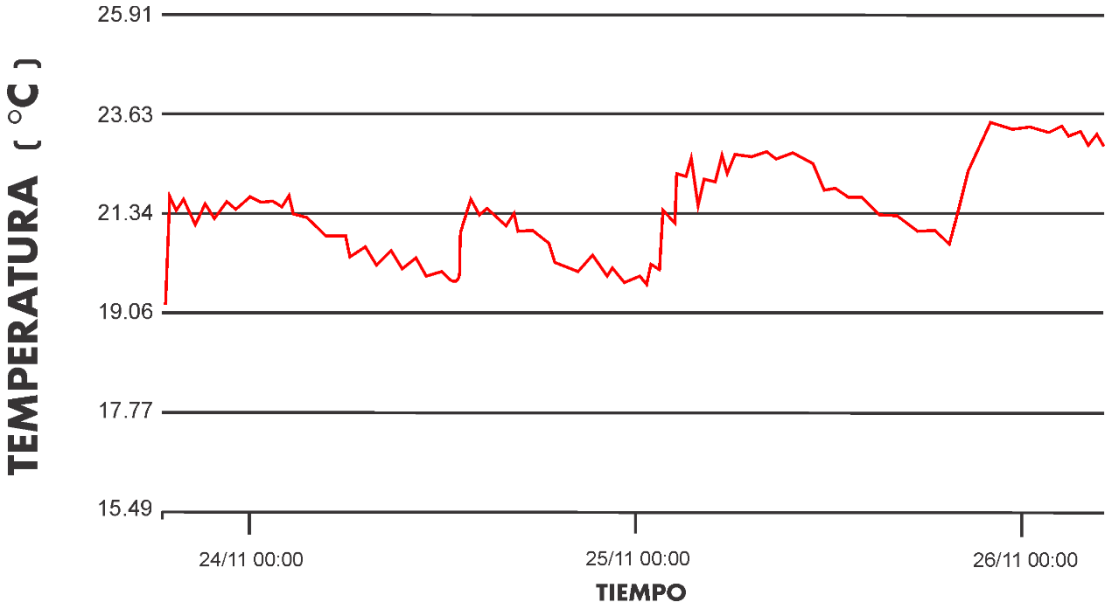


Figura 62

Gráfica de temperatura en el nodo final 2

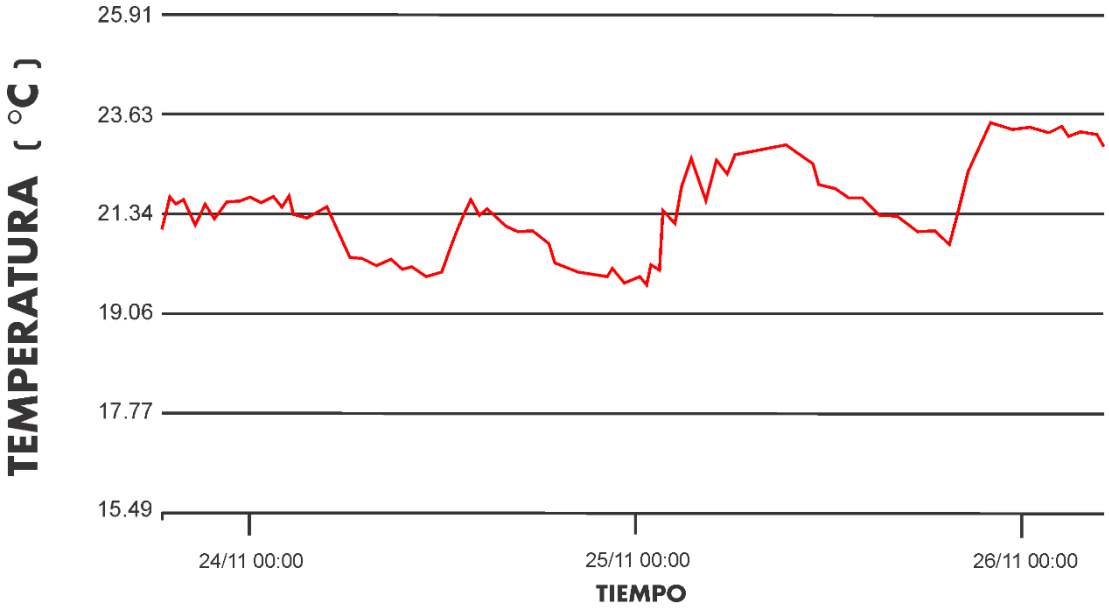
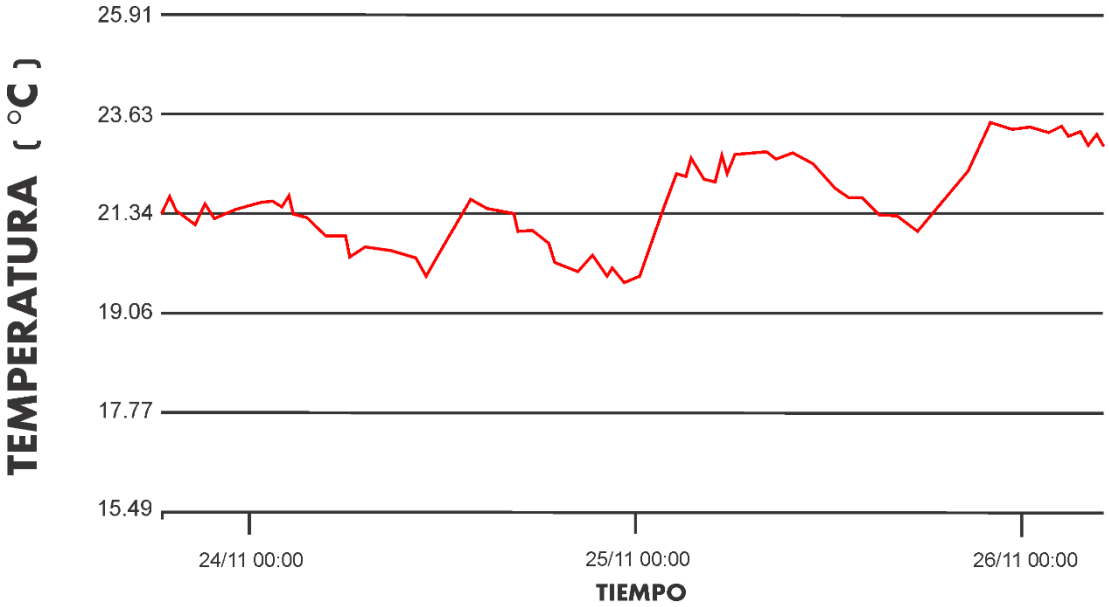


Figura 63

Gráfica de temperatura en el nodo final 3



4.4.3. Humedad

La humedad se midió en porcentaje (%). Esta variable tiene un comportamiento opuesto a la temperatura (en U) ya que la humedad relativa es inversamente proporcional a la temperatura. A medida que sube la temperatura, el aire se vuelve más seco y al disminuir, el aire se vuelve más húmedo. A continuación, se muestra el comportamiento de la humedad en cada nodo (figura 64, 65 y 66 respectivamente).

Figura 64

Gráfica de humedad en el nodo final 1

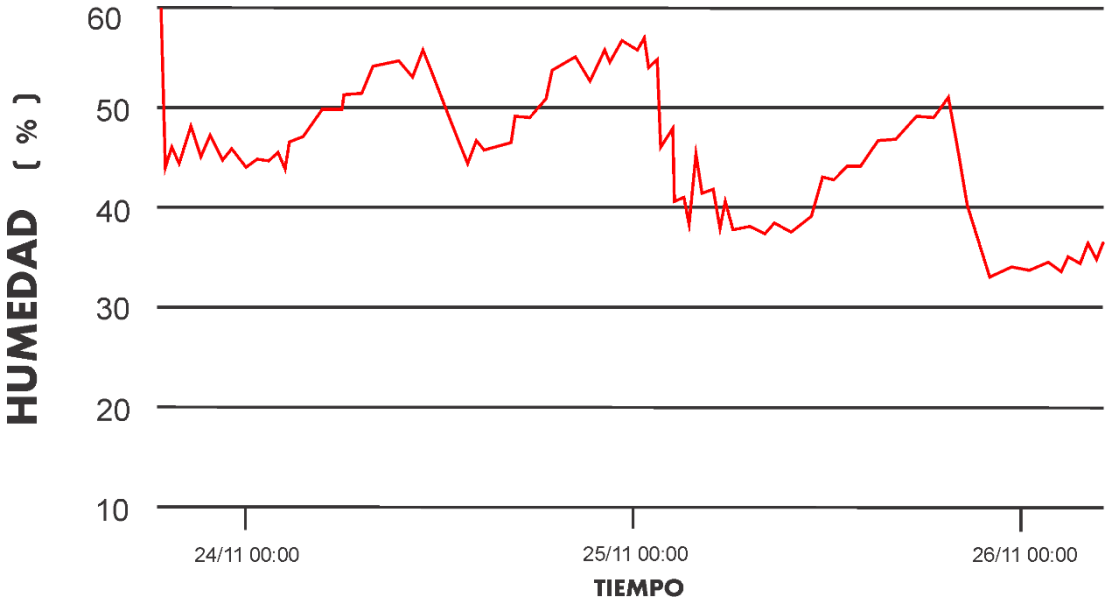


Figura 65

Gráfica de humedad en el nodo final 2

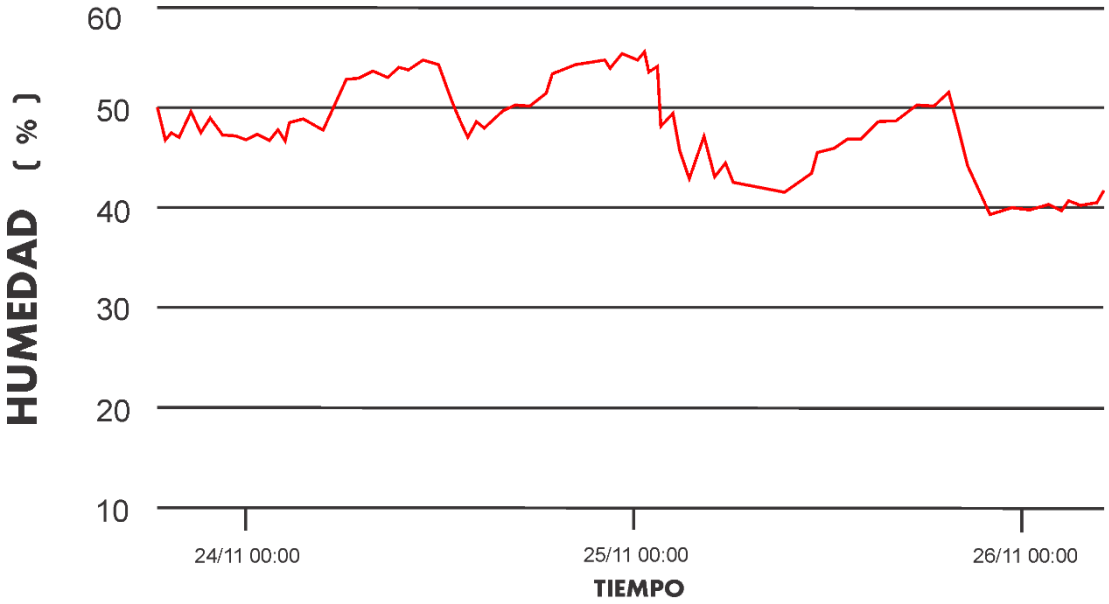
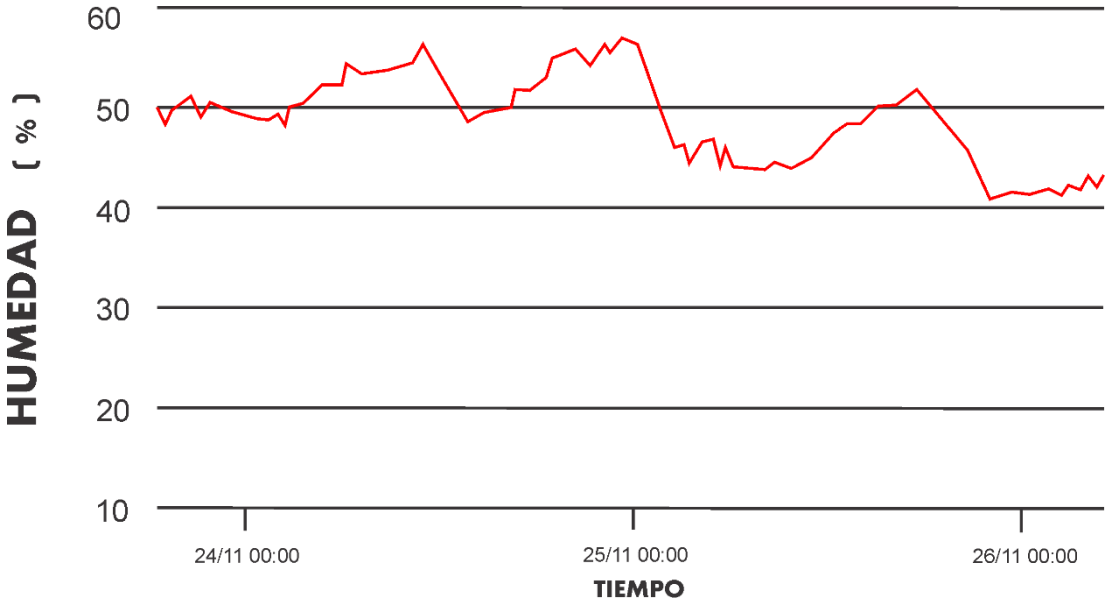


Figura 66

Gráfica de humedad en el nodo final 3



4.4.4. CO2

El CO_2 se midió en partículas por millón (ppm). El comportamiento de este tiene un valor casi constante durante la mañana, teniendo un pequeño incremento durante la tarde-noche debido al aumento de personas en las áreas medidas. Las personas generan pequeñas cantidades de CO_2 al exhalar. El comportamiento de la cantidad de CO_2 en cada nodo se muestra en las siguientes imágenes (figura 67, 68 y 69 respectivamente).

Figura 67

Gráfica de CO2 en el nodo final 1

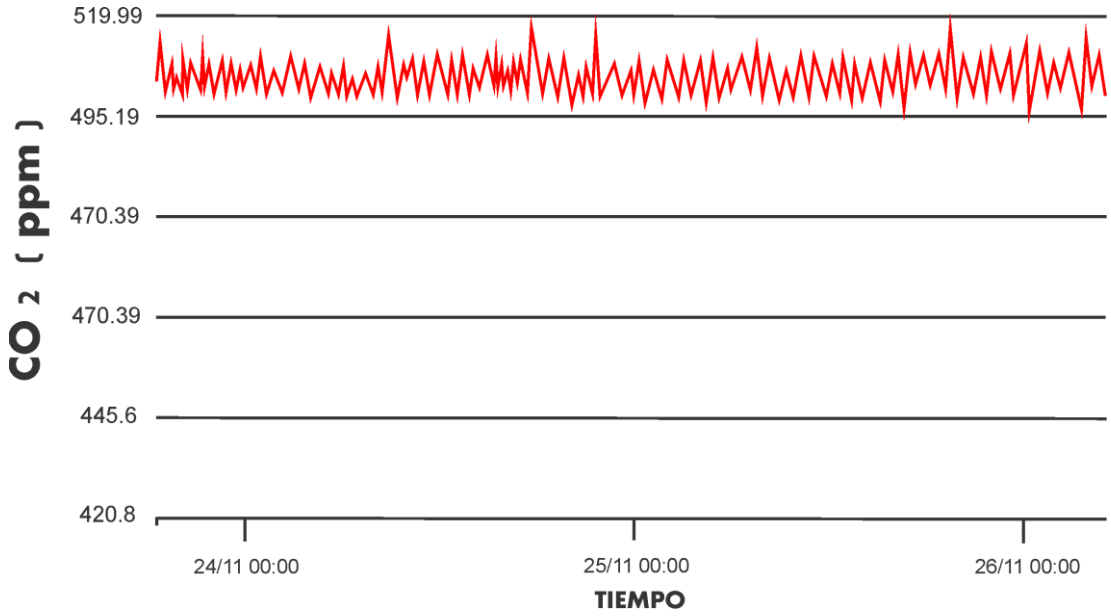


Figura 68

Gráfica de CO2 en el nodo final 2

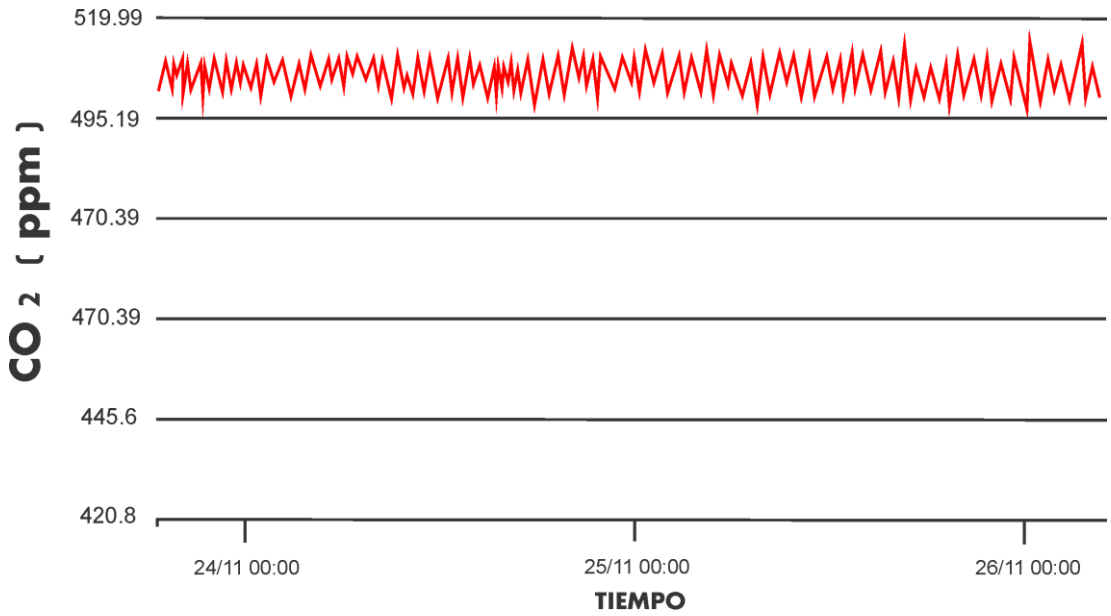
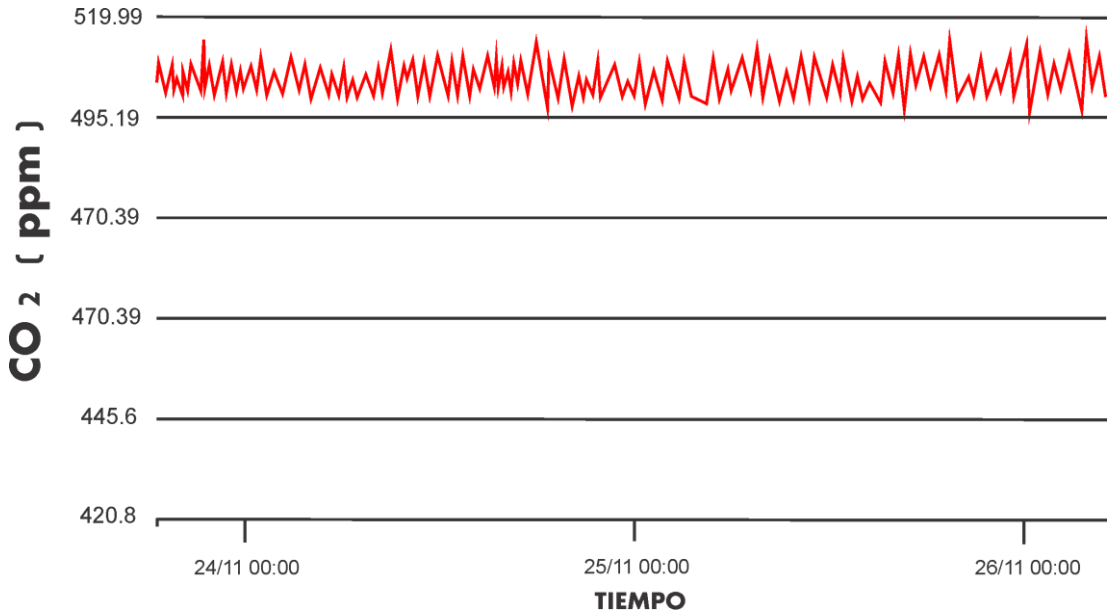


Figura 69

Gráfica de CO2 en el nodo final 3



4.4.5. PM2.5

El PM2.5 se midió en microgramos por metro cubico (µg/m^3). Esto hace referencia al polvo en general. El comportamiento de este parámetro es el que tuvo mayor variación, esto debido a que la concentración de polvo es afectada por muchos factores como corriente de aire, labores de limpieza que levanta polvo, el paso de personas, etc. Esta variación siempre está en un rango fijo por lo que podemos deducir que es un comportamiento normal. Las gráficas de cada nodo de PM2.5 se muestran a continuación (figura 70, 71 y 72 respectivamente).

Figura 70

Gráfica de PM2.5 en el nodo final 1

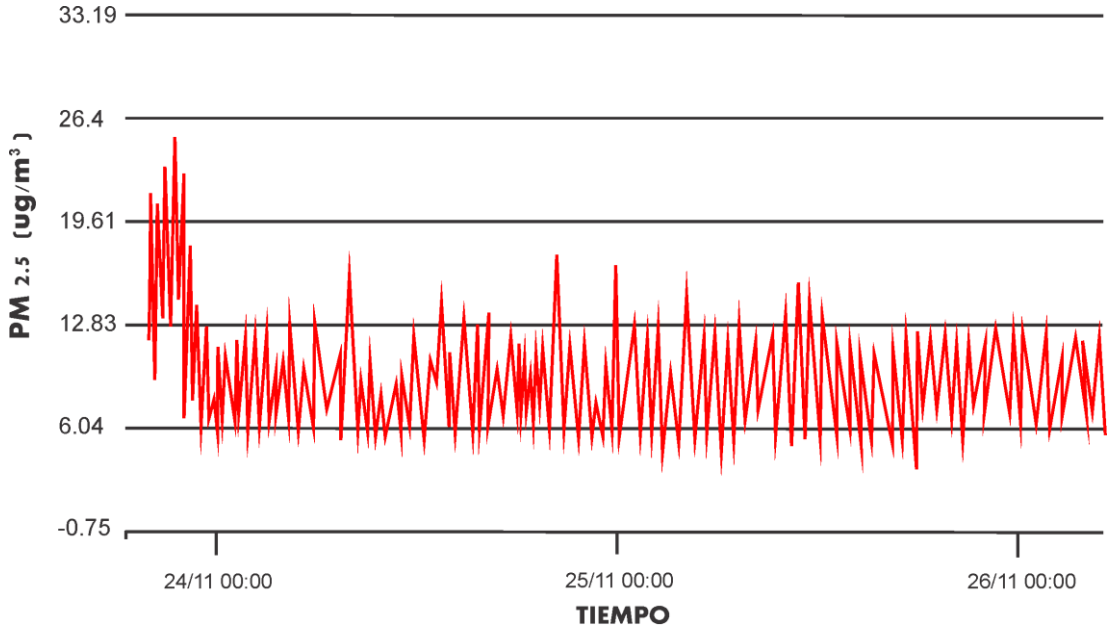


Figura 71

Gráfica de PM2.5 en el nodo final 2

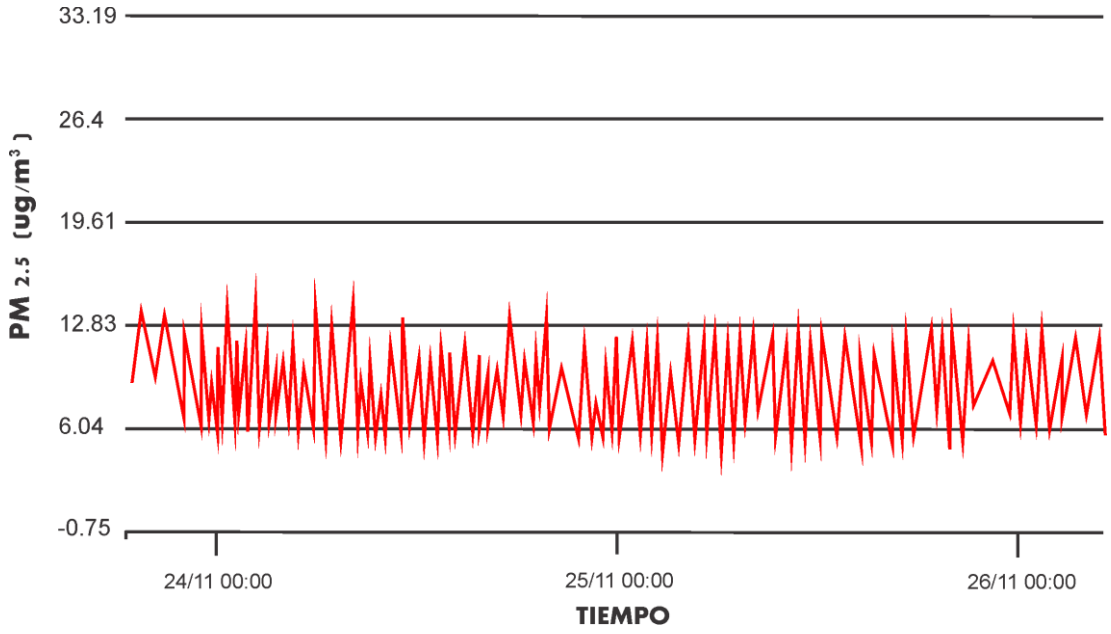
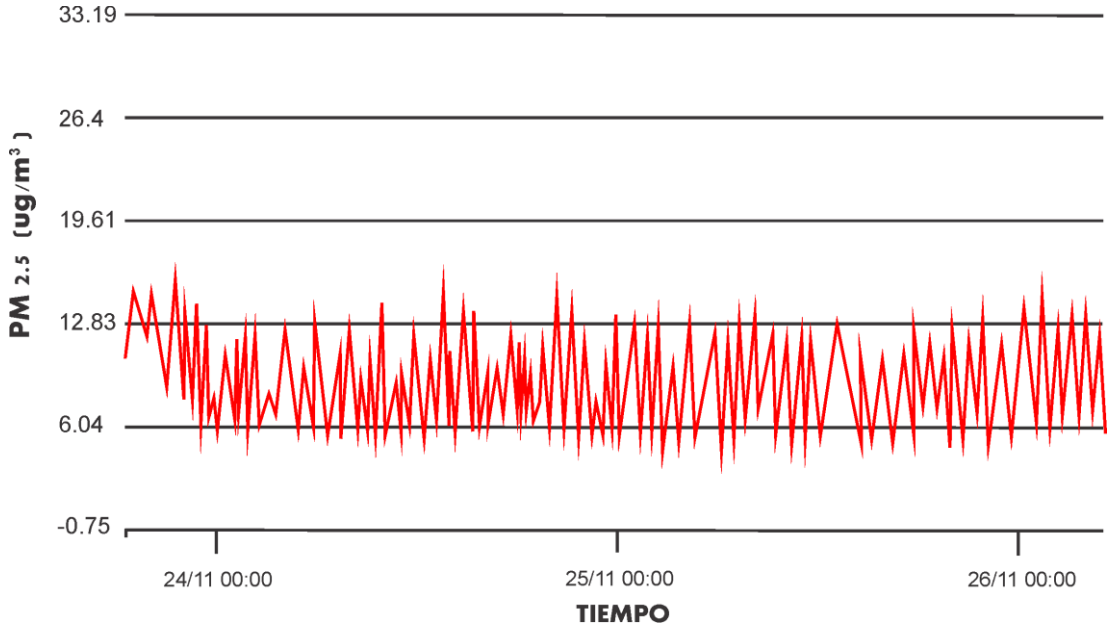


Figura 72

Grafica de PM2.5 en el nodo final 3



4.4.6. Vista general

Con los datos obtenidos se tabuló los valores máximos y mínimos para cada tipo de parámetro, además de la media (tabla 17).

Tabla 17

Valores generales de los parámetros medidos

Parámetros	Min.	Max.	Media
Presión (hPa)	955,60	957,50	956,02
Temperatura (°C)	17,3	24,9	22,6
Humedad (%)	30,6	44,9	36,2
CO_2 (ppm)	400	802,35	530,32
PM2.5(µg/m^3)	8,2	17,53	9,35

CAPÍTULO V: DISCUSIÓN

A partir de los resultados hallados, aceptamos la hipótesis general que plantea que: “El diseño de un sistema IoT permite monitorear remotamente la calidad del aire en ambientes interiores de la Facultad de Ingeniería de la Universidad Privada de Tacna”.

Se comprobó que se puede expandir la cobertura de una red sin la necesidad de realizar un despliegue de puntos de acceso, todo esto gracias a la topología de red tipo Malla; teniendo como resultado conexiones con un RSSI mayores a -60 dBm, las cuales se consideran dentro del rango de “muy buena” según Singh (s.f.). Tambien se comprobo que la velocidad de conexión entre los nodos es de 26 Mbits/s o 3,25 MB/s. Esta velocidad es más que suficiente teniendo en cuenta que cada trama de información tiene un tamaño de 111 bytes promedio. Todos estos datos se comprobaron en ambiente de interiores como lo es la Facultad de Ingeniería de la Universidad Privada de Tacna, teniendo como obstáculos, paredes de concreto.

Otro punto importante que se comprobó fue el rendimiento de la base de datos con respecto a la cantidad de datos almacenados y su tamaño que ocupan. Teniendo como resultado un total de 54696 registros almacenados en un día, se obtiene un tamaño final ocupado de 80,00 MB en el disco duro. Por la versión Express de la base de datos, que nos ofrece 10 GB de almacenamiento gratis, se estimó que tiene la capacidad de almacenar un tiempo máximo de 21 meses de corrido. Esto comprueba el rendimiento, la calidad de la organización y estructuración de los datos óptimos para este tipo sistemas basados en IoT.

También se planteó verificar la calidad del aire y otros factores en ambientes interiores. El resultado arrojo que, en cuanto a la concentración de CO_2 y PM2.5, los valores medidos cumplen con los Estándares de Calidad Ambiental (ECA) para aire. En cuanto a los parámetros meteorológicos (como temperatura, humedad relativa y presión atmosférica) se vio que varían ligeramente con los valores dados por IQAir (2022) en la ciudad de Tacna.

CONCLUSIÓN

Según los resultados encontrados se concluye que es posible ampliar la cobertura de una red sin la necesidad de instalar APs utilizando una topología de red tipo Malla con la capacidad de ser vinculada a una red WLAN para transmitir la información al exterior teniendo un buen rendimiento tanto en potencia de transmisión, como en velocidad de transmisión.

Se comprobó que es posible crear una red de sensores independientes que se comunican entre ellos usando chips de bajo costo como es el ESP-07 y estructurar la formación de datos para tener un mejor rendimiento en el ancho de banda de la red.

Se llegó a la conclusión de que el almacenamiento de datos de acuerdo con la magnitud de este proyecto no es un problema, ya que los datos ocupan una cantidad mínima con respecto al tamaño total del disco duro de almacenamiento.

Se comprobó que el desarrollo de aplicaciones de software son una herramienta importante para el monitoreo de datos, ya que permite, de forma intuitiva y ordenada, la visualización correcta de estos de tal manera que el usuario final sea capaz de tomar decisiones correctas frente a algún comportamiento anormal.

Se comprobó que los sensores utilizados para este proyecto (BME680 Y DSM501A) fueron los correctos, ya que nos ofrecen parámetros para determinar la concentración de gases y material particulado que influyen en la calidad del aire en interiores.

Se concluyó también que los parámetros de calidad de aire medidos están dentro de los Estándares de Calidad Ambiental (ECA) del aire. Y los parámetros meteorológicos como temperatura, presión y humedad relativa varían un poco con respecto a los datos dados por IQAir en la ciudad de Tacna.

RECOMENDACIONES

Se recomienda, para el diseño de la fuente de alimentación, calcular correctamente la corriente de consumo total de todos los dispositivos usados, ya que de tener una fuente con una corriente deficiente provoca que el sensor de lecturas erróneas, o que el nodo en la malla se desconecte constantemente por falta de potencia.

Se recomienda a los futuros investigadores que, para obtener resultados más contundentes en cuanto al diseño de la red, comprobar el tamaño del buffer en cada nodo (final o intermedio, ESP-07) y su método de reenvío de tramas para poder calcular el número máximo de nodos antes de saturar la red Malla.

Se recomienda investigar el uso de otras tecnologías para el desarrollo de software (como desarrollo web o móvil) ya que la plataforma creada en este proyecto solo está disponible para el SO de Windows.

Las muestras tomadas en este proyecto corresponden a sensores de bajo costo que, si bien han sido calibrados conforme a lo establecido por el fabricante, en una investigación posterior se recomiendan ser comparados con dispositivos profesionales a fin de encontrar una correlación entre ellos.

REFERENCIAS BIBLIOGRÁFICAS

Azuero, Á. (12 de Noviembre de 2018). Significatividad del marco metodológico en el desarrollo de proyectos de investigación. Cuenca, Ecuador.

BOSCH. (Julio de 2017). *BME680.* Obtenido de https://cdn-shop.adafruit.com/product-files/3660/BME680.pdf

Cahuantico, R. (2019). *Evaluación de contaminantes atmosféricos CO, SO2, PM10, PM2.5 de la zona urbana cusco 2017.* Obtenido de http://repositorio.unsa.edu.pe/handle/UNSA/10999

Castro, G. (2017). *Cifrado simétrico de datos en la comunicación de sistemas embebidos para su uso en el internet de las cosas.* Obtenido de http://repositorio.umsa.bo/xmlui/handle/123456789/12902

Corona, L., Abaca, G., & Mares, J. (2014). *Sensores y actuadores. Aplicaciones con Arduino.* México: Grupo Editorial Patria.

Corpus, B., & Portugal, A. (2021). *Diseño de una red inalámbrica para el servicio de pesaje en el sector agroindustrial utilizando Balanzas Super SS mediante protocolo SPI basado en microcontrolador ESP32.* Obtenido de https://hdl.handle.net/20.500.14138/4851

Cruz, M., Oliete, P., Morales, C., Gonzáles, C., Cendon, B., & Hernandez, A. (2015). *Las tecnologías IoT dentor de la industria conectada 4.0.* Madrid: Fundación EOI.

Department of Occupational Safety and Health, Malasia. (2010). *Industry Code of Practice on Indoor Air Quality 2010 (ICOP IAQ 2010).* Obtenido de https://www.dosh.gov.my/index.php/chemical-management-v/indoor-air-quality

El Peruano. (22 de Setiembre de 2020). Modifican la Norma Técnica EM.030 Instalaciones de Ventilación del Reglamento Nacional de Edificaciones y dictan otras disposiciones. *El Peruano*. Obtenido de https://busquedas.elperuano.pe/normaslegales/modifican-la-norma-tecnica-em030-instalaciones-de-ventilaci-resolucion-ministerial-no-232-2020-vivienda-1887042-2/

Fernández, M. (s.f.). *Redes de datos. Medios de transmisión*. Obtenido de Universidad de Cádiz: https://rodin.uca.es/bitstream/handle/10498/16867/tema05_medios.pdf

Gahona, R., & Gavilema, A. (2020). *Diseño de la red internet de las cosas (IOT) para el edificio de la empresa CONSEL.* Obtenido de http://dspace.ups.edu.ec/handle/123456789/18755

Ingeniería TV. (28 de setiembre de 2021). *Tendencias de salud: La calidad del aire en ambientes interiores [video]*. Facebook. Obtenido de https://www.facebook.com/ciptvpe/videos/354134109791649

INSST. (2000). *NTP 549: El dióxido de carbono en la evaluación de la calidad del aire.* Obtenido de https://www.insst.es/documents/94886/327064/ntp_549.pdf/e9364a82-6f1b-4590-90e0-1d08b22e1074

INSST. (2001). *NTP 607: Guías de calidad de aire interior: contaminantes químicos.* Obtenido de https://www.insst.es/documents/94886/326775/ntp_607.pdf/0c6960b6-b461-4d21-9757-e4ea03004327

IQAir. (2022). Calidad del aire en Tacna. Tacna, Tacna, Peru. Obtenido de https://www.iqair.com/es/peru/tacna

Last Minute Engineers. (s.f.). Obtenido de Interface BME680 Environmental Sensor with Arduino: https://lastminuteengineers.com/bme680-gas-pressure-humidity-temperature-sensor-arduino-tutorial/

Martinez, A., & Romieu, I. (1997). *Introducción al monitoreo atmosférico.* Obtenido de https://pesquisa.bvsalud.org/portal/resource/pt/pah-24717

MINAM. (2016). *Informe Nacional de la Calidad del Aire 2013-2014.* Obtenido de https://www.minam.gob.pe/wp-content/uploads/2016/07/Informe-Nacional-de-Calidad-del-Aire-2013-2014.pdf

MINAM. (14 de Julio de 2016). *MINAM.* Obtenido de Ministerio del Ambiente: https://www.minam.gob.pe/wp-content/uploads/2016/07/RM-N%C2%B0-181-2016-MINAM.pdf

MINAM. (2017). *SINIA*. Obtenido de Sistema Nacional de Información Ambiental: https://sinia.minam.gob.pe/indicador/966

Morales, A. (2018). *Evaluación de la calidad del aire interior en salas de clases en la UTFSM.* Obtenido de https://repositorio.usm.cl/handle/11673/46872

Murata, R. (2020). *Diseño de un sistema de monitoreo de contaminación acústica urbana bajo una plataforma IoT.* Obtenido de http://hdl.handle.net/20.500.12404/17601

Quiñones, O. (2019). *Internet de las Cosas.* Ibukku.

Serna, A., Ros, F., & Rico, J. (2010). *Guía Practica de Sensores.* Creaciones Copyright. Obtenido de https://books.google.es/books?id=CuoXCd6ZZqwC

Sillero, J., Rodríguez, N., Montiveros, M., Murazzo, M., Píccoli, F., & Méndez, M. (2018). Análisis de las Topologias IoT en Entornos Fog Computing mediante simulación. *VI Jornadas de Cloud Computing & Big Data (JCC&BD 2018)*, 94.

Singh, R. (s.f.). *Techmusa*. Obtenido de Technology to the Point: https://techmusa.com/wireless-dbm-table/

Troposfera. (s.f.). *troposfera.org*. Obtenido de troposfera.org: https://www.troposfera.org/conceptos/calidad-aire/

Valle, O., & Rivera, O. (2008). Monitoreo e Indicadores. Obtenido de http://aularedim.net/wp-content/uploads/monitoreo_indocadores.pdf

ANEXO

Anexo 1. Código fuente de ventana de inicio de sesión

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication4
{
    public partial class Login : Form
    {
        private int IdUsuario;
        public int _IdUsuario
        {
            get { return IdUsuario; }
            //set { IdUsuario = value; }
        }
        public Login()
        {
            InitializeComponent();
            pic_logo.Select();
        }
        #region BotonesBarraTitulo
        private void btn_cerrar_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
        private void btn_min_Click(object sender, EventArgs e)
        {
            WindowState = FormWindowState.Minimized;
        }
        #endregion
        private void btn_entrar_Click(object sender, EventArgs e)
        {
           int aut= CapaNegocio.Autenticacion.verificar(txt_usuario.Text,txt_contraseña.Text);
            if (aut!=0)
            {
                IdUsuario = aut;
                this.Hide();
                Form_principal fp = new Form_principal();
                AddOwnedForm(fp);
                fp.Show();
            }
            else
            {
                MessageBox.Show("Datos Incorrectos");
            }
        }
    }
```

```
53 }
54
```

Anexo 2. Código fuente de ventana principal

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.ComponentModel;
 4  using System.Data;
 5  using System.Drawing;
 6  using System.Linq;
 7  using System.Text;
 8  using System.Threading.Tasks;
 9  using System.Windows.Forms;
10  using System.Data.SqlClient;
11  using System.Configuration;
12  using System.Net.Sockets;
13
14  namespace WindowsFormsApplication4
15  {
16      public partial class Form_principal : Form
17      {
18          private int IdUsuario;
19          public int _IdUsuario { get { return IdUsuario; } }
20
21          public Form_principal()
22          {
23              InitializeComponent();
24              btn_rest.Visible = false;
25
26
27          }
28          #region BotonesBarraTitulo
29          private void btn_min_Click(object sender, EventArgs e)
30          {
31              WindowState = FormWindowState.Minimized;
32          }
33
34          private void btn_max_Click(object sender, EventArgs e)
35          {
36              this.Size = Screen.PrimaryScreen.WorkingArea.Size;
37              this.Location = new Point(0, 0);
38              btn_rest.Visible = true;
39          }
40
41          private void btn_cerrar_Click(object sender, EventArgs e)
42          {
43              Application.Exit();
44          }
45
46          private void btn_rest_Click(object sender, EventArgs e)
47          {
48              this.Size = new Size(1100, 650);
49              centerForm();
50              btn_rest.Visible = false;
51          }
52
```

```
53      private void timer1_Tick(object sender, EventArgs e)
54      {
55          lbl_Hora.Text = DateTime.Now.ToLongTimeString();
56          lbl_Fecha.Text = DateTime.Now.ToLongDateString();
57      }
58
59      private void pictureBox1_Click(object sender, EventArgs e)
60      {
61          if (pnl_menu.Width == 249)
62          {
63              pnl_menu.Visible = false;
64              pnl_menu.Width = 71;
65              bunifuTransition2.Show(pnl_menu);
66          }
67          else
68          {
69              pnl_menu.Visible = false;
70              pnl_menu.Width = 249;
71              bunifuTransition1.Show(pnl_menu);
72          }
73      }
74      #endregion
75      #region BotonesEnterLeave
76      private void btn_graficas_MouseEnter(object sender, EventArgs e)
77      {
78          btn_graficas.ForeColor = Color.White;
79          btn_graficas.Image = WindowsFormsApplication4.Properties.Resources.graficasB;
80      }
81
82      private void btn_graficas_MouseLeave(object sender, EventArgs e)
83      {
84          btn_graficas.ForeColor = ColorTranslator.FromHtml("192; 0; 0");
85          btn_graficas.Image = WindowsFormsApplication4.Properties.Resources.graficas;
86      }
87
88      private void btn_inicio_MouseEnter(object sender, EventArgs e)
89      {
90          btn_inicio.ForeColor = Color.White;
91          btn_inicio.Image = WindowsFormsApplication4.Properties.Resources.inicioB;
92      }
93
94      private void btn_inicio_MouseLeave(object sender, EventArgs e)
95      {
96          btn_inicio.ForeColor = ColorTranslator.FromHtml("192; 0; 0");
97          btn_inicio.Image = WindowsFormsApplication4.Properties.Resources.INICIO;
98      }
99
100     private void btn_edit_MouseEnter(object sender, EventArgs e)
101     {
102         btn_edit.ForeColor = Color.White;
103         btn_edit.Image = WindowsFormsApplication4.Properties.Resources.userB;
104     }
105
106     private void btn_edit_MouseLeave(object sender, EventArgs e)
107     {
108         btn_edit.ForeColor = ColorTranslator.FromHtml("192; 0; 0");
```

```
109             btn_edit.Image = WindowsFormsApplication4.Properties.Resources.USUARIO;
110         }
111
112         private void btn_info_MouseEnter(object sender, EventArgs e)
113         {
114             btn_info.ForeColor = Color.White;
115             btn_info.Image = WindowsFormsApplication4.Properties.Resources.infoB;
116         }
117
118         private void btn_info_MouseLeave(object sender, EventArgs e)
119         {
120             btn_info.ForeColor = ColorTranslator.FromHtml("192; 0; 0");
121             btn_info.Image = WindowsFormsApplication4.Properties.Resources.INFO;
122         }
123         #endregion
124         private void abrirFormHijo(object form_hijo)
125         {
126
127             Form fh = form_hijo as Form;
128             AddOwnedForm(fh);
129             fh.TopLevel = false;
130             fh.Dock = DockStyle.Fill;
131             this.pnl_fondo.Controls.Add(fh);
132             this.pnl_fondo.Tag = fh;
133             label1.Visible = false;
134             fh.Show();
135             fh.BringToFront();
136
137         }
138
139         private void btn_inicio_Click(object sender, EventArgs e)
140         {
141             //if (pnl_fondo.Controls[0].Name != "Dispositivos")
142             //{
143             //    abrirFormHijo(new Dispositivos_v());
144             //}
145             int b = pnl_fondo.Controls.Find("Dispositivos_v", true).Count();
146             if (b == 0)
147             {
148                 abrirFormHijo(new Dispositivos_v());
149             }
150             else
151                 ((Form)pnl_fondo.Controls.Find("Dispositivos_v",
152 true).FirstOrDefault()).BringToFront();
153         }
154         private void Idusuario()
155         {
156             Login log = Owner as Login;
157             this.IdUsuario = log._IdUsuario;
158             CapaDatos.DatosPersonales datosUsuarios =
159              CapaNegocio.Autenticacion.InformacionUsuario(this.IdUsuario);
160             lblNombre.Text = datosUsuarios.Nombres + " " + datosUsuarios.Apellidos;
161             lblCargo.Text = datosUsuarios.Tipo;
162
163         }
164
```

```
165         private void Form_principal_Load(object sender, EventArgs e)
166         {
167             Idusuario();
168         }
169
170         private void btn_info_Click(object sender, EventArgs e)
171         {
172             //if (pnl_fondo.Controls[0].Name != "Informacion")
173             //{
174             //    abrirFormHijo(new Informacion());
175             //}
176             int b = pnl_fondo.Controls.Find("Informacion", true).Count();
177             if (b == 0)
178             {
179                 abrirFormHijo(new Informacion());
180             }
181             else
182                 ((Form)pnl_fondo.Controls.Find("Informacion",
183 true).FirstOrDefault()).BringToFront();
184         }
185
186         private void btn_graficas_Click(object sender, EventArgs e)
187         {
188             int b = pnl_fondo.Controls.Find("Graficas", true).Count();
189             if (b == 0)
190             {
191                 abrirFormHijo(new Graficas());
192             }
193             else
194                 ((Form)pnl_fondo.Controls.Find("Graficas",
195 true).FirstOrDefault()).BringToFront();
196             pnl_menu.Visible = false;
197             pnl_menu.Width = 71;
198             bunifuTransition2.Show(pnl_menu);
199         }
200         private void Form_principal_Shown(object sender, EventArgs e)
201         {
202             //this.Size = Screen.PrimaryScreen.WorkingArea.Size;
203             //this.Location = new Point(0, 0);
204         }
205         private void centerForm()
206         {
207             Rectangle area = Screen.PrimaryScreen.WorkingArea;
208
209             this.Top = (area.Height - this.Height) / 2;
210             this.Left = (area.Width - this.Width) / 2;
            }
        }
    }
```

Anexo 3. Código fuente de la ventana de inicio

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.IO;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10 using System.Windows.Forms;
11 using CapaConexionRemota;
12 using CapaNegocio;
13
14 namespace WindowsFormsApplication4
15 {
16     public partial class Dispositivos_v : Form
17     {
18         ConexionESP8266 c = new ConexionESP8266();
19
20         public Dispositivos_v()
21         {
22             InitializeComponent();
23             dgv_Vinculados.Visible = false;
24         }
25         private void Dispositivos_Load(object sender, EventArgs e)
26         {
27             mostrarDispositivos();
28         }
29         private void mostrarDispositivos()
30         {
31             if (Autenticacion.verDispositivos(1) != null)
32             {
33                 dgv_Vinculados.DataSource = Autenticacion.verDispositivos(1);
34                 dgv_Vinculados.Columns[0].Width = 150;
35                 dgv_Vinculados.Columns[1].Width = 100;
36                 dgv_Vinculados.Columns[2].Width = 150;
37                 dgv_Vinculados.Columns[3].Width = 150;
38                 dgv_Vinculados.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
39                 dgv_Vinculados.Visible = true;
40             }
41             else
42             {
43                 dgv_Vinculados.DataSource = null;
44                 dgv_Vinculados.Visible = false;
45             }
46         }
47         private void Dispositivos_Shown(object sender, EventArgs e)
48         {
49             dgv_Vinculados.ClearSelection();
50             dgv_Vinculados.CurrentCell = null;
51         }
52
53         private void btnAgregar_Click(object sender, EventArgs e)
54         {
55             AgregarDispositivos formulario = new AgregarDispositivos(this);
56             formulario.ShowDialog();
```

```
57      }
58      public DataGridView _dgv_vinculados
59      {
60          get { return dgv_Vinculados; }
61          set { dgv_Vinculados = value; }
62      }
63      private void dgv_Vinculados_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)
64      {
65          if (e.Button == MouseButtons.Right && e.RowIndex != -1)
66          {
67              cms_dispositivos.Items.Clear();
68              cms_dispositivos.Items.Add("Eliminar").Name = "ELIMINAR";
69              dgv_Vinculados.CurrentCell = dgv_Vinculados.Rows[e.RowIndex].Cells[1];
70              cms_dispositivos.Show(dgv_Vinculados, e.Location);
71              cms_dispositivos.Show(Cursor.Position);
72          }
73      }
74
75      private void cms_dispositivos_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
76      {
77          cms_dispositivos.Close();
78          if (e.ClickedItem.Name == "ELIMINAR")
79          {
80              DialogResult mensaje = MessageBox.Show("¿Esta seguro de ELIMINAR esta actividad?. Esta
81                actividad se borrara permanentemente", "Confirmacion!", MessageBoxButtons.OKCancel,
82                MessageBoxIcon.Question);
83              if (mensaje == DialogResult.OK)
84              {
85                  Autenticacion.eliminarDispositivos();
86                  mostrarDispositivos();
87                  MessageBox.Show("El dispositivo se ha borrado correctamente", "Exito!", MessageBoxButtons.OK,
88  MessageBoxIcon.Information);
89              }
90
91          }
92      }
93      private void deviceTimer_Tick(object sender, EventArgs e)
94      {
95          if (dgv_Vinculados.Rows.Count > 0)
96          {
97              string ip = Autenticacion.verIP(dgv_Vinculados.Rows[0].Cells["NOMBRE"].Value.ToString());
98              if (c.conPing(ip) == 3)
99              {
100                 dgv_Vinculados.Rows[0].Cells["ONLINE"].Value =
101 File.ReadAllBytes("C:/Users/Rudy/Desktop/imagenes/varios/desconectado.png");
102             }
103             else
104             {
105                 dgv_Vinculados.Rows[0].Cells["ONLINE"].Value =
106 File.ReadAllBytes("C:/Users/Rudy/Desktop/imagenes/varios/conectado.png");
107              }
             }
          }
        }
      }
```

Anexo 4. Código fuente de ventana de agregar dispositivo

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using CapaConexionRemota;

namespace WindowsFormsApplication4
{
    public partial class AgregarDispositivos : Form
    {
        int row = 0;
        int column = 0;

        //public AgregarDispositivos()
        //{
        //    InitializeComponent();
        //}
        private Dispositivos_v mainForm = null;
        public AgregarDispositivos(Form callingForm)
        {
            mainForm = callingForm as Dispositivos_v;
            InitializeComponent();
        }

        private void btn_cerrar_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private async void AgregarDispositivos_Load(object sender, EventArgs e)
        {
```

```
36          if(CapaNegocio.Autenticacion.comprobarVinculados())
37          {
38              lbl_Vinculados.Visible = false;
39              dgv_vinculados.Visible = true;
40              dgv_vinculados.DataSource = null;
41              dgv_vinculados.DataSource = CapaNegocio.Autenticacion.verVinculados();
42          }
43          else
44          {
45              lbl_Vinculados.Visible = true;
46              dgv_vinculados.Visible = false;
47          }
48          ConexionESP8266.lista.CollectionChanged += (sendere, evento) =>
49          {
50              if (evento.NewItems != null)
51              {
52                  if (InvokeRequired)
53                  {
54                      Invoke(new Action(() => dgv_disponibles.DataSource = null));
55                  }
56                  if (InvokeRequired)
57                  {
58                      Invoke(new Action(() => dgv_disponibles.DataSource = ConexionESP8266.lista));
59                  }
60
61              }
62          };
63          Task nota = new Task(() => {
64              ConexionESP8266.encontrarDispositivos();
65          });
66          nota.Start();
67          await nota;
68          pictureBox1.Visible = false;
69      }
70
71      private async void button1_Click(object sender, EventArgs e)
```

```
72      {
73          ConexionESP8266.lista.CollectionChanged += (sendere, evento) =>
74              {
75                  if (evento.NewItems != null)
76                  {
77                      foreach (var item in evento.NewItems)
78                      {
79                          dgv_disponibles.DataSource = null;
80                          dgv_disponibles.DataSource = ConexionESP8266.lista;
81                      }
82                  }
83              };
84          Task nota = new Task(() => {
85              ConexionESP8266.contar();
86          });
87          await nota;
88      }
89
90      private void dgv_disponibles_MouseClick(object sender, MouseEventArgs e)
91      {
92          if (e.Button == MouseButtons.Right)
93          {
94              var ubicacion = dgv_disponibles.HitTest(e.X, e.Y);
95              row = ubicacion.RowIndex;
96              column = ubicacion.ColumnIndex;
97              dgv_disponibles.ClearSelection();
98
99              if (ubicacion.RowIndex != -1)
100             {
101                 dgv_disponibles.Rows[ubicacion.RowIndex].Selected = true;
102                 menu.Items.Clear();
103                 menu.Items.Add("Vincular").Name="VINCULAR";
104                 menu.Show(dgv_disponibles, new Point(e.X, e.Y));
105             }
106
107         }
```

```
108             }
109
110         private void menu_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
111         {
112             switch (e.ClickedItem.Name)
113             {
114                 case "VINCULAR":
115                     try
116                     {
117                         CapaNegocio.Autenticacion.agregarDispositivos(dgv_disponibles.Rows[row].Cells["HOSTNAME"].Value.ToString(),
118                         dgv_disponibles.Rows[row].Cells["IP"].Value.ToString(), dgv_disponibles.Rows[row].Cells["MAC"].Value.ToString());
119                         DialogResult m = MessageBox.Show("Dispositivo Vinculado Correctamente", "Exito", MessageBoxButtons.OK);
120                         if (m == DialogResult.OK)
121                         {
122                             dgv_vinculados.Visible= true;
123                             dgv_vinculados.DataSource= CapaNegocio.Autenticacion.verVinculados();
124                             if (CapaNegocio.Autenticacion.verDispositivos(1) != null)
125                             {
126                                 this.mainForm._dgv_vinculados.DataSource = CapaNegocio.Autenticacion.verDispositivos(1);
127                                 this.mainForm._dgv_vinculados.Columns[0].Width = 150;
128                                 this.mainForm._dgv_vinculados.Columns[1].Width = 100;
129                                 this.mainForm._dgv_vinculados.Columns[2].Width = 140;
130                                 this.mainForm._dgv_vinculados.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
131                                 this.mainForm._dgv_vinculados.Visible = true;
132                             }
133                         }
134                         break;
135                     }
136                     catch (Exception ex)
137                     {
138                         MessageBox.Show(ex.Message);
139                         break;
140                     }
141             }
142         }
143
```

```
144         private void dgv_vinculados_DataSourceChanged(object sender, EventArgs e)
145         {
146             dgv_vinculados.Columns[0].Width = 140;
147             dgv_vinculados.Columns[1].Width = 120;
148             dgv_vinculados.Columns[2].Width = 150;
149         }
150
151         private void dgv_disponibles_DataSourceChanged(object sender, EventArgs e)
152         {
153             try
154             {
155                 dgv_disponibles.Columns[0].Width = 140;
156                 dgv_disponibles.Columns[1].Width = 120;
157                 dgv_disponibles.Columns[2].Width = 150;
158             }
159             catch(Exception ex) { }
160
161         }
162
163         private void menu_Closed(object sender, ToolStripDropDownClosedEventArgs e)
164         {
165             menu.Items.Clear();
166         }
167     }
168 }
```

Anexo 5. Código fuente de la ventana de información de nodos

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Threading.Tasks;
9   using System.Windows.Forms;
10  using CapaConexionRemota;
11  using CapaNegocio;
12  using CapaDatos;
13  using TableDependency.SqlClient;
14
15  namespace WindowsFormsApplication4
16  {
17      public partial class Informacion : Form
18      {
19          ConexionESP8266 c = new ConexionESP8266();
20          public Informacion()
21          {
22              InitializeComponent();
23              using (ESP8266Entities db0 = new ESP8266Entities())
24              {
25                  SqlTableDependency<Nodos> dependecyNodes = new
    SqlTableDependency<Nodos>(db0.Database.Connection.ConnectionString);
26
27                  dependecyNodes.OnChanged += DependecyNodes_OnChanged;
28                  dependecyNodes.Start();
29
30              }
31          }
32          private void btn_conectar_Click(object sender, EventArgs e)
33          {
34              if (cbx_Dispositivos.SelectedIndex != 0)
35              {
36                  if (c.conectarSocket(Autenticacion.verIP(cbx_Dispositivos.Text)))
37                  {
38                      pic_estado.Image = WindowsFormsApplication4.Properties.Resources.conectado;
39                      lbl_estado.Text = "CONECTADO";
40                      lbl_estado.ForeColor = ColorTranslator.FromHtml("15; 189; 3");
41                      Form_principal fp = Owner as Form_principal;
42                      btn_conectar.Visible = false;
43                      btn_desconect.Visible = true;
44                      fp.btn_edit.Enabled = true;
45                      fp.btn_graficas.Enabled = true;
46                      fp.btn_info.Enabled = true;
47                      timerConectado.Enabled = true;
48                      Autenticacion.offline();
49                      dgv_Nodos.DataSource = Autenticacion.verNodos();
50                  }
51              }
52          }
```

```
53
54          private void btn_conectar_MouseEnter(object sender, EventArgs e)
55          {
56              btn_conectar.Font = new Font("Century Gothic", 11.0f);
57              btn_conectar.Font = new Font(btn_conectar.Font, FontStyle.Bold);
58          }
59
60          private void btn_conectar_MouseLeave(object sender, EventArgs e)
61          {
62              btn_conectar.Font = new Font("Century Gothic", 9.7f);
63
64          }
65
66          private void btn_desconect_Click(object sender, EventArgs e)
67          {
68              desconect();
69          }
70          private void timerConectado_Tick(object sender, EventArgs e)
71          {
72              if (c.conPing(Autenticacion.verIP(cbx_Dispositivos.Text)) == 3)
73              {
74                  timerConectado.Enabled = false;
75                  desconect();
76              }
77          }
78          private void desconect()
79          {
80              c.desconectar();
81              pic_estado.Image = Properties.Resources.desconectado;
82              lbl_estado.Text = "DESCONECTADO";
83              lbl_estado.ForeColor = ColorTranslator.FromHtml("192;0;0");
84              btn_conectar.Visible = true;
85              btn_desconect.Visible = false;
86              timerConectado.Enabled = false;
87          }
88
89          private void Informacion_Load(object sender, EventArgs e)
90          {
91              cbx_Dispositivos.DataSource = Autenticacion.nombres();
92              Task leer = new Task(c.escuchar);
93              leer.Start();
94          }
95            private void DependecyNodes_OnChanged(object sender,
96      TableDependency.SqlClient.Base.EventArgs.RecordChangedEventArgs<Nodos> e)
97          {
98              dgv_Nodos.Invoke(new Action(() => { dgv_Nodos.DataSource = Autenticacion.verNodos(); }));
99          }
100         private void cbx_Dispositivos_DropDownClosed(object sender, EventArgs e)
101         {
102             lbl_estado.Focus();
103         }
104     }
105 }
```

Anexo 6. Código fuente de ventana de visor de parametros

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using CapaNegocio;
using System.Reflection;
using System.Windows.Forms.DataVisualization.Charting;
using System.Threading;

namespace WindowsFormsApplication4
{
    public partial class Graficas : Form
    {
        string id = "";
        bool bucle = true;
        public delegate void OnSerialDataReceived(Data datos);

        public Graficas()
        {
            InitializeComponent();
        }
        private void Graficas_Load(object sender, EventArgs e)
        {
            cbx_nodos.DataSource = Autenticacion.cbxGraficas();
        }
        private void AddPointToChart(Data data)
        {
            chartTemp.Series[0].Points.AddXY(data.value.Value, data.date.Value);
        }
        private void graficar()
        {
            chartTemp.Series[0].Points.Clear();
            chartTemp.Series[0].XValueType = ChartValueType.DateTime;
            chartTemp.ChartAreas[0].AxisY.LabelStyle.Format = "0.##";
            chartTemp.ChartAreas[0].AxisX.LabelStyle.Format = "dd/MM HH:mm";
            Parametros result = new Parametros();
            List<decimal?> val = new List<decimal?>();
            List<DateTime> fecha = new List<DateTime>();
            result = Autenticacion.graficaTempList(id);
            result.values.ForEach(item =>
            {
                val.Add(item.value.Value);
                fecha.Add(item.date.Value);
            });
            chartTemp.Series[0].Points.DataBindXY(fecha, val);
        chartTemp.ChartAreas[0].Axes[1].Maximum = Decimal.ToDouble(result.max) + 4;
        chartTemp.ChartAreas[0].Axes[1].Minimum = Decimal.ToDouble(result.min) - 4;
            var task = new Task(async () =>
            {
                while (bucle)
                {
```

```
57                  All all = new All();
58                  all = Autenticacion.graficaAll(id);
59                  lbl_precision.Invoke(new Action(() => { lbl_precision.Text =
60                   all.val.PRECISION_IAQ.ToString(); }));
61                  lbl_presion.Invoke(new Action(() => { lbl_presion.Text =
62                 all.val.PRESION.ToString(); }));
63                  lbl_temp.Invoke(new Action(() => { lbl_temp.Text =
64                 all.val.TEMPERATURA.ToString(); }));
65                  lbl_humedad.Invoke(new Action(() => { lbl_humedad.Text =
66                 all.val.HUMEDAD.ToString(); }));
67                  lbl_co2.Invoke(new Action(() => { lbl_co2.Text = all.val.CO2.ToString(); }));
68                  lbl_pm25.Invoke(new Action(() => { lbl_pm25.Text =
69                 all.pm25.Value.ToString(); }));
70                  lbl_iaq.Invoke(new Action(() => { lbl_iaq.Text = all.val.IAQ.ToString(); }));
71                  await Task.Delay(1000);
72              }
73          });
74          task.Start();
75      }
76
77      private void chart1_Click(object sender, EventArgs e)
78      {
79      }
80      private void cbx_nodos_SelectedIndexChanged(object sender, EventArgs e)
81      {
82          if (cbx_nodos.SelectedIndex != 0)
83          {
84              List<dynamic> result = Autenticacion.nodoById(cbx_nodos.Text);
85              id = result[0].GetType().GetProperty("NODO").GetValue(result[0]);
86              if (result[0].GetType().GetProperty("RSSI").GetValue(result[0]) == "Offline")
87              {
88                  pb_online.Image = Properties.Resources.offline;
89              }
90              else
91              {
92                  pb_online.Image = Properties.Resources.online;
93              }
94              if (id != "")
95              {
96                  bucle = true;
97                  graficar();
98              }
99          }
100         else
101             bucle = false;
102     }
103   }
104 }
```

Anexo 7. Código de comunicación entre ventanas y base de datos

```
1  using System;
2  using System.Collections.Generic;
```

```
3  using System.Globalization;
4  using System.IO;
5  using System.Linq;
6  using System.Net.NetworkInformation;
7  using System.Text;
8  using System.Threading.Tasks;
9  using CapaDatos;
10
11 namespace CapaNegocio
12 {
13    public class Autenticacion
14    {
15       public static int verificar(string user, string pass)
16       {
17          using (ESP8266Entities db0 = new ESP8266Entities())
18          {
19             var lst = from d in db0.Login
20                       where d.Usuario == user
21                       && d.Contraseña == pass
22                       select d.ID;
23             if (lst.Count() > 0)
24             { return lst.First(); }
25             else
26             { return 0; }
27          }
28       }
29       public static void agregarDispositivos(string dispo, string ip, string mac)
30       {
31          using (ESP8266Entities db = new ESP8266Entities())
32          {
33             CapaDatos.Dispositivos oDispositivos = new CapaDatos.Dispositivos();
34             oDispositivos.Dispositivos1 = dispo;
35             oDispositivos.Id_Usuario = 1;
36             oDispositivos.Tipo = "Sensor";
37             oDispositivos.Ip = ip;
38             oDispositivos.Mac = mac;
39             db.Dispositivos.Add(oDispositivos);
40             db.SaveChanges();
41          }
42       }
43       public static DatosPersonales InformacionUsuario(int Id)
44       {
45          using (ESP8266Entities db0 = new ESP8266Entities())
46          {
47             DatosPersonales lst = db0.DatosPersonales.Find(Id);
48             return lst;
49          }
50       }
51       public class miDisp
52       {
53          public string NOMBRE { get; set; }
54          public string TIPO { get; set; }
55          public string IP { get; set; }
56          public string MAC { get; set; }
57          public byte[] ONLINE { get; set; }
58       }
```

```
59      private static bool hacerPing(string ip)
60      {
61          int exito = 0;
62          for (int i = 0; i < 1; i++)
63          {
64              Ping p1 = new Ping();
65              PingReply respuesta = p1.Send(ip, 150);
66              if (respuesta.Status == IPStatus.Success)
67                  exito++;
68          }
69          if (exito > 0)
70              return true;
71          else return false;
72      }
73      public static List<miDisp> verDispositivos(int id)
74      {
75          using (ESP8266Entities db0 = new ESP8266Entities())
76          {
77              var lst = (from d in db0.Dispositivos
78                         where d.Id_Usuario == id
79                         select new miDisp { NOMBRE = d.Dispositivos1, TIPO = d.Tipo, IP =
80              d.Ip, MAC = d.Mac }).ToList();
81              if (lst.Count() > 0)
82              {
83                  if (hacerPing(lst[0].IP))
84                  {
85                      lst.ElementAt(0).ONLINE =
86              File.ReadAllBytes("C:/Users/Rudy/Desktop/imagenes/varios/conectado.png");
87                  }
88                  else lst.ElementAt(0).ONLINE =
89              File.ReadAllBytes("C:/Users/Rudy/Desktop/imagenes/varios/desconectado.png");
90                  return lst.ToList();
91              }
92              return null;
93          }
94      }
95      public static void eliminarDispositivos()
96      {
97          using (ESP8266Entities db0 = new ESP8266Entities())
98          {
99              db0.Database.ExecuteSqlCommand("TRUNCATE TABLE Dispositivos");
100         }
101     }
102     public static List<Vinculados> verVinculados()
103     {
104         using (ESP8266Entities db0 = new ESP8266Entities())
105         {
106             var lst = from d in db0.Dispositivos
107                       select new Vinculados { HOSTNAME = d.Dispositivos1, IP = d.Ip, MAC =
108             d.Mac };
109             return lst.ToList();
110         }
111     }
112     public static List<string> nombres()
113     {
114         using (ESP8266Entities db0 = new ESP8266Entities())
```

```
115         {
116             List<string> lst = new List<string>();
117             lst = (from d in db0.Dispositivos
118                    select d.Dispositivos1).ToList();
119             lst.Insert(0, "--Seleccione Dispositivo--");
120             return lst;
121         }
122     }
123     public static string verIP(string hostname)
124     {
125         using (ESP8266Entities db0 = new ESP8266Entities())
126         {
127             var ip = (from d in db0.Dispositivos
128                       where d.Dispositivos1 == hostname
129                       select d.Ip).FirstOrDefault();
130             return ip.ToString();
131         }
132     }
133     public static void offline()
134     {
135         using (ESP8266Entities db0 = new ESP8266Entities())
136         {
137             var lst = (from d in db0.Nodos
138                        select d).ToList();
139             foreach (var item in lst)
140             {
141                 item.RSSI = "Offline";
142             }
143             db0.SaveChanges();
144         }
145     }
146     public static bool verificarVinculados(string mac)
147     {
148         using (ESP8266Entities db0 = new ESP8266Entities())
149         {
150             var lst = from d in db0.Dispositivos
151                       where d.Mac == mac
152                       select d;
153             if (lst.Count() > 0)
154                 return true;
155             else return false;
156         }
157     }
158     public static bool comprobarVinculados()
159     {
160         using (ESP8266Entities db0 = new ESP8266Entities())
161         {
162             var lst = from d in db0.Dispositivos
163                       select d;
164             if (lst.Count() > 0)
165                 return true;
166             else return false;
167         }
168     }
169     public static All graficaAll(string idMesh)
170     {
```

```
171         using (ESP8266Entities db0 = new ESP8266Entities())
172         {
173             var lst = (from d in db0.Variables
174                        where d.ID_MESH == idMesh
175                        orderby d.ID descending
176                        select new Var
177                        {
178                            PRESION = d.PRESION,
179                            CO2 = d.CO2,
180                            HUMEDAD = d.HUMEDAD,
181                            IAQ = d.IAQ,
182                            PRECISION_IAQ = d.PRECISION_IAQ,
183                            TEMPERATURA = d.TEMPERATURA,
184                        }).FirstOrDefault();
185             var lst2 = (from e in db0.PM_25
186                         where e.ID_MESH== idMesh
187                         orderby e.ID descending
188                         select e.PM25).FirstOrDefault();
189 
190             All result = new All()
191             {
192                 val= lst,
193                 pm25=lst2
194             };
195             return result;
196         }
197     }
198     public static Data graficaTemp(string idMesh)
199     {
200         using (ESP8266Entities db0 = new ESP8266Entities())
201         {
202             var lst = (from d in db0.Variables
203                        where d.ID_MESH == idMesh
204                        orderby d.ID descending
205                        select new Data
206                        {
207                            value = d.TEMPERATURA,
208                            date = d.FECHA
209                        }
210                        ).FirstOrDefault();
211             return lst;
212             //return float.Parse(lst, CultureInfo.InvariantCulture.NumberFormat);
213         }
214     }
215     public static Parametros graficaTempList(string idMesh)
216     {
217         using (ESP8266Entities db0 = new ESP8266Entities())
218         {
219             var lst = (from d in db0.Variables
220                        where d.ID_MESH == idMesh
221                        orderby d.ID ascending
222                        select new Data
223                        {
224                            value = d.TEMPERATURA,
225                            date = d.FECHA
226                        }).ToList<Data>();
```

```
227
228             decimal? max = (from e in db0.Variables
229                             where e.ID_MESH == idMesh
230                             select e.TEMPERATURA).Max();
231
232             decimal? min = (from f in db0.Variables
233                             where f.ID_MESH == idMesh
234                             select f.TEMPERATURA).Min();
235
236             Parametros result = new Parametros()
237             {
238                 values = lst,
239                 max = max ?? 50,
240                 min = min ?? 0
241             };
242             return result;
243         }
244     }
245     public static decimal? graficaHum()
246     {
247         using (ESP8266Entities db0 = new ESP8266Entities())
248         {
249             var lst = (from d in db0.Variables
250                        where d.ID_MESH == "1528102012"
251                        orderby d.ID descending
252                        select d.HUMEDAD).FirstOrDefault();
253             return lst;
254             //return float.Parse(lst, CultureInfo.InvariantCulture.NumberFormat);
255         }
256     }
257     public static decimal? graficaPres()
258     {
259         using (ESP8266Entities db0 = new ESP8266Entities())
260         {
261             var lst = (from d in db0.Variables
262                        where d.ID_MESH == "1528102012"
263                        orderby d.ID descending
264                        select d.PRESION).FirstOrDefault();
265             return lst;
266             //return float.Parse(lst, CultureInfo.InvariantCulture.NumberFormat);
267         }
268     }
269     public static decimal? graficaCo2()
270     {
271         using (ESP8266Entities db0 = new ESP8266Entities())
272         {
273             var lst = (from d in db0.Variables
274                        where d.ID_MESH == "1528102012"
275                        orderby d.ID descending
276                        select d.CO2).FirstOrDefault();
277             return lst;
278             //return float.Parse(lst, CultureInfo.InvariantCulture.NumberFormat);
279         }
280     }
281     public static void addNode(string id_nodo)
282     {
```

```
283         using (ESP8266Entities db0 = new ESP8266Entities())
284         {
285             var lst = (from d in db0.Nodos
286                        where d.ID_NODO == id_nodo
287                        select d).FirstOrDefault();
288             if (lst == null)
289             {
290                 Nodos e = new Nodos();
291                 e.ID_NODO = id_nodo;
292                 db0.Nodos.Add(e);
293                 db0.SaveChanges();
294             }
295         }
296     }
297     public static void verifyNodes(string[] nodes)
298     {
299         using (ESP8266Entities db0 = new ESP8266Entities())
300         {
301             var lst = (from d in db0.Nodos
302                        select d).ToList();
303             foreach (var item in lst)
304             {
305                 if (!nodes.Contains(item.RSSI))
306                 {
307                     item.RSSI = "Offline";
308                     db0.SaveChanges();
309 
310                 }
311             }
312         }
313     }
314     public static void addNodeNetwork(string id_nodo, string nodo, string rssi)
315     {
316         using (ESP8266Entities db = new ESP8266Entities())
317         {
318             var lst = (from d in db.Nodos
319                        where d.ID_NODO == id_nodo
320                        select d).FirstOrDefault();
321             if (lst != null)
322             {
323                 lst.NODO = nodo;
324                 lst.RSSI = rssi;
325                 db.SaveChanges();
326             }
327             else
328             {
329                 Nodos node = new Nodos();
330                 node.ID_NODO = id_nodo;
331                 node.NODO = nodo;
332                 node.RSSI = rssi;
333                 db.Nodos.Add(node);
334                 db.SaveChanges();
335             }
336         }
337     }
338     public static void addNodeParameters_9(string id_mesh, decimal pm25)
```

```
339     {
340         using (ESP8266Entities db = new ESP8266Entities())
341         {
342             PM_25 parametros = new PM_25();
343             parametros.ID_MESH = id_mesh;
344             parametros.PM25 = pm25;
345             parametros.FECHA = DateTime.Now;
346             db.PM_25.Add(parametros);
347             db.SaveChanges();
348 
349         }
350     }
351     public static void addNodeParameters_10(string id_mesh, decimal presion, int
352       precision, decimal temperatura, decimal humedad, decimal iaq, decimal co2)
353     {
354         using (ESP8266Entities db = new ESP8266Entities())
355         {
356             Variables parametros = new Variables();
357             parametros.ID_MESH = id_mesh;
358             parametros.PRESION = presion;
359             parametros.PRECISION_IAQ = precision;
360             parametros.TEMPERATURA = temperatura;
361             parametros.HUMEDAD = humedad;
362             parametros.IAQ = iaq;
363             parametros.CO2 = co2;
364             parametros.FECHA = DateTime.Now;
365             db.Variables.Add(parametros);
366             db.SaveChanges();
367         }
368     }
369     public static void addNodeParameters_11(string id_mesh, decimal presion, int
370     precision, decimal temperatura, decimal humedad, decimal iaq, decimal co2, decimal
371     pm25)
372     {
373         using (ESP8266Entities db = new ESP8266Entities())
374         {
375             Variables parametros = new Variables();
376             PM_25 parametros_2 = new PM_25();
377             parametros.ID_MESH = id_mesh;
378             parametros.PRESION = presion;
379             parametros.PRECISION_IAQ = precision;
380             parametros.TEMPERATURA = temperatura;
381             parametros.HUMEDAD = humedad;
382             parametros.IAQ = iaq;
383             parametros.CO2 = co2;
384             parametros.FECHA = DateTime.Now;
385             parametros_2.ID_MESH = id_mesh;
386             parametros_2.PM25 = pm25;
387             parametros_2.FECHA = DateTime.Now;
388             db.Variables.Add(parametros);
389             db.PM_25.Add(parametros_2);
390             db.SaveChanges();
391         }
392     }
393     public static List<dynamic> verNodos()
394     {
```

```
395         using (ESP8266Entities db0 = new ESP8266Entities())
396         {
397             var lst = from d in db0.Nodos
398                       select d;
399             return lst.ToList<dynamic>();
400         }
401     }
402     public static List<string> cbxGraficas()
403     {
404         using (ESP8266Entities db0 = new ESP8266Entities())
405         {
406             List<string> lst = new List<string>();
407             lst = (from d in db0.Nodos
408                    orderby d.NODO ascending
409                    select d.NODO).ToList();
410             lst.Insert(0, "--Seleccione Ubicacion--");
411             return lst;
412         }
413     }
414     public static List<dynamic> nodoById(string nodo)
415     {
416         using (ESP8266Entities db0 = new ESP8266Entities())
417         {
418             var lst = (from d in db0.Nodos
419                        where d.NODO == nodo
420                        select new
421                        {
422                            NODO = d.ID_NODO,
423                            RSSI = d.RSSI
424                        }).ToList<dynamic>();
425             return
426         }
427     }
428 }
429 public class Vinculados
430 {
431     private string hostname;
432 
433     public string HOSTNAME
434     {
435         get { return hostname; }
436         set { hostname = value; }
437     }
438     private string ip;
439 
440     public string
441     {
442         get { return ip; }
443         set { ip = value
444     }
445     private string mac;
446 
447     public string
448     {
449         get { return mac; }
450         set { mac = value
```

```
451         }
452     }
453     public class Parametros
454     {
455         public List<Data> values;
456         public decimal max;
457         public decimal min;
458     }
459     public class Data
460     {
461         public decimal? value;
462         public DateTime? date;
463     }
464
465     public class All
466     {
467         public Var val;
468         public decimal? pm25;
469     }
470     public class Var
471     {
472         public decimal? PRESION;
473         public int? PRECISION_IAQ;
474         public decimal? TEMPERATURA;
475         public decimal? HUMEDAD;
476         public decimal? IAQ;
477         public decimal? CO2;
478       }
479     }
```

Anexo 8. Interface entre nodo intermedio y la base de datos

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Net;
5  using System.Net.NetworkInformation;
6  using System.Net.Sockets;
7  using System.Text;
8  using System.Threading;
9  using System.Threading.Tasks;
10 using System.Windows.Forms;
11 using System.Collections.ObjectModel;
12 using System.Runtime.InteropServices;
13
14 namespace CapaConexionRemota
15 {
16    public class ConexionESP8266
17    {
18       Socket s = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
19       ProtocolType.Tcp);
20       IPEndPoint servi;
21       public static ObservableCollection<dispositivos> lista = new
22       ObservableCollection<dispositivos>();
23       public static bool conectado;
24
25       [DllImport("iphlpapi.dll", ExactSpelling = true)]
26       public static extern int SendARP(int dstIP, int srcIP, [Out] byte[] macAddr, ref int
27       hwAddrLength);
28
29       public bool conectarSocket(string direccion)
30       {
31          try
32          {
33             s = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
34       ProtocolType.Tcp);
35             servi = new IPEndPoint(IPAddress.Parse(direccion), 80);
36             s.Connect(servi);
37             return true;
38          }
39          catch (Exception e)
40          {
41             MessageBox.Show(e.ToString());
42             return false;
43          }
44       }
45       public void responder(string texto)
46       {
47          if (s.Connected)
48          {
49             //Task t = new Task(escuchar);
50             byte[] enviar = new byte[100];
51             string data;
52             data = texto;
53             enviar = Encoding.UTF8.GetBytes(data);
54             s.Send(enviar);
55             //t.Start();
```

```
56          }
57      }
58      public void escuchar()
59      {
60          while (true)
61          {
62              byte[] recibir;
63              while (s.Connected)
64              {
65                  try
66                  {
67                      recibir = new byte[100];
68                      string dato = "";
69                      int arreglo = 0;
70                      arreglo = s.Receive(recibir, 0, recibir.Length, 0);
71                      Array.Resize(ref recibir, arreglo);
72                      dato = Encoding.UTF8.GetString(recibir);
73                      string[] fdata = dato.Split(',');
74                      if (fdata[0] == "network")
75                      {
76                          foreach (var item in fdata)
77                          {
78                              if (item != "network" && item != "")
79                              {
80                                  CapaNegocio.Autenticacion.addNode(item);
81                              }
82                          }
83                          CapaNegocio.Autenticacion.verifyNodes(fdata.Skip(1).ToArray());
84                      }
85                      else if (fdata[0] == "value")
86                      {
87                          if (fdata.Length == 9)
88                          {
89                              CapaNegocio.Autenticacion.addNodeNetwork(fdata[1], fdata[8],
90                               fdata[7]);
91                              CapaNegocio.Autenticacion.addNodeParameters_9(fdata[1],
92                             Convert.ToDecimal(fdata[6]));
93                          }
94                          else if (fdata.Length == 10)
95                          {
96                              CapaNegocio.Autenticacion.addNodeNetwork(fdata[1], fdata[9],
97                               fdata[8]);
98                              CapaNegocio.Autenticacion.addNodeParameters_10(fdata[1],
99              Convert.ToDecimal(fdata[2]), Convert.ToInt32(fdata[3]),
100             Convert.ToDecimal(fdata[4]), Convert.ToDecimal(fdata[5]),
101             Convert.ToDecimal(fdata[6]), Convert.ToDecimal(fdata[7]));
102                         }
103                         else if (fdata.Length == 11)
104                         {
105                             CapaNegocio.Autenticacion.addNodeNetwork(fdata[1], fdata[10],
106                           fdata[9]);
107                             CapaNegocio.Autenticacion.addNodeParameters_11(fdata[1],
108             Convert.ToDecimal(fdata[2]), Convert.ToInt32(fdata[3]),
109             Convert.ToDecimal(fdata[4]), Convert.ToDecimal(fdata[5]),
110             Convert.ToDecimal(fdata[6]), Convert.ToDecimal(fdata[7]),
111             Convert.ToDecimal(fdata[8]));
```

```
                    }
                    else if (fdata.Length == 4)
                    {
                        CapaNegocio.Autenticacion.addNodeNetwork(fdata[1], fdata[3],
                         fdata[2]);
                    }
                }
            }
            catch (Exception e) { }
        }
    }

}
public int conPing(string ip)
{
    int promePing = 0;
    Ping p = new Ping();
    for (int i = 0; i < 3; i++)
    {
        PingReply respuesta = p.Send(ip, 120);
        if (respuesta.Status != IPStatus.Success)
        {
            promePing++;
        }
    }
    return promePing;
}

public void desconectar()
{
    s.Close(50);
}
public static void encontrarDispositivos()
{
    var addresses = Dns.GetHostEntry(Dns.GetHostName());
    string result = "";
    foreach (var ip in addresses.AddressList)
    {
        if (ip.AddressFamily == AddressFamily.InterNetwork)
        {
            result = ip.ToString();
        }
    }
    string address = result.Substring(0, result.LastIndexOf(".")) + ".";
    lista.Clear();
    for (int i = 7; i < 150; i++)
    {
        Ping p1 = new Ping();
        PingReply respuesta = p1.Send(address + i.ToString(), 150);
        if (respuesta.Status == IPStatus.Success)
        {
            try
            {
                IPHostEntry disp = Dns.GetHostEntry(address + i.ToString());
                if (disp != null)
                {
```

```
168                     string nombre = disp.HostName;
169
170                     IPAddress addr = IPAddress.Parse(address + i.ToString());
171                     byte[] mac = new byte[6];
172                     int hwLength = 6;
173                     SendARP(BitConverter.ToInt32(addr.GetAddressBytes(), 0), 0, mac, ref
174                     hwLength);
175                     String macAddress = BitConverter.ToString(mac, 0, hwLength);
176                     if (!CapaNegocio.Autenticacion.verificarVinculados(macAddress))
177                         lista.Add(new dispositivos { HOSTNAME = nombre, IP = address +
178                         i.ToString(), MAC = macAddress });
179                 }
180             }
181             catch (Exception e)
182             {
183             }
184         }
185     }
186 }
187 public static async void contar()
188 {
189     lista.Clear();
190     Ping p1 = new Ping();
191     p1.PingCompleted += (sendere, evento) =>
192     {
193         if (evento.Reply != null && evento.Reply.Status == IPStatus.Success)
194         {
195             try
196             {
197                 IPHostEntry disp = Dns.GetHostEntry(evento.Reply.Address);
198             }
199             catch (Exception e)
200             {
201             }
202         }
203     };
204     for (int i = 7; i < 30; i++)
205     {
206         try
207         {
208             p1.Send("192.168.0." + i.ToString(), 100);
209             await Task.Delay(100);
210         }
211         catch (Exception e) { }
212     }
213 }
214 }
215 public class dispositivos
216 {
217     private string hostname;
218     public string HOSTNAME
219     {
220         get { return hostname; }
221         set { hostname = value; }
222     }
223     private string ip;
```

```
224         public string IP
225         {
226             get { return ip; }
227             set { ip = value; }
            }
            private string mac;
            public string MAC
            {
                get { return mac; }
                set { mac = value; }
              }

          }
        }
```

Anexo 9. Captura de los datos mediante los sensores – Nodo Final

```
#include "bsec.h"

//------------------------------- FUNCIONES Y VARIABLES-----------------------------------------------


String msg;

//####------------------------- BME680 ---####--------------------

Bsec iaqSensor;            // Crea un objeto de la clase BSEC
String output;            //Salida del sensor BME680


//####------------------------- DSM501A ---####--------------------

#define pin10  16
#define pin25  17


volatile unsigned long duracion_10 = 0;
volatile unsigned long tiempo_start_10 = micros();
volatile unsigned long tiempo_stop_10 = micros();

volatile unsigned long duracion_25 = 0;
volatile unsigned long tiempo_start_25 = micros();
volatile unsigned long tiempo_stop_25 = micros();

unsigned long starttime;
unsigned long t_muestreo = 30000;

float tasa_10 = 0;
float concentracion_10 = 0;

float tasa_25 = 0;
float concentracion_25 = 0;

float con_pm25 = 0;
float _con_pm25 = 0;
float pm25 = 0;

IRAM_ATTR void interrupcion_10()
{
  if (digitalRead(pin10) == LOW)
  {
    tiempo_start_10 = micros();
  }
  else
  {
    tiempo_stop_10 = micros();
    duracion_10 += (tiempo_stop_10 - tiempo_start_10);
  }
}

IRAM_ATTR void interrupcion_25()
{
```

```
  if (digitalRead(pin25) == LOW)
  {
    tiempo_start_25 = micros();
  }
  else
  {
    tiempo_stop_25 = micros();
    duracion_25 += (tiempo_stop_25 - tiempo_start_25);
  }
}


//------------------------------ CONFIGURACION INICIAL ----------------------------------------------
void setup(void)
{
  Serial.begin(115200);     //Inicializacion de serial

  //####--- BME680 ---####-----------------------------------------

  Wire.begin();             //Inicializacion de I2C
  iaqSensor.begin(BME680_I2C_ADDR_SECONDARY, Wire);   //Configuracion de conmunicacion  de sensor
BME 680
  bsec_virtual_sensor_t sensorList[6] = {
    BSEC_OUTPUT_RAW_PRESSURE,
    BSEC_OUTPUT_IAQ,
    BSEC_OUTPUT_STATIC_IAQ,
    BSEC_OUTPUT_CO2_EQUIVALENT,
    BSEC_OUTPUT_SENSOR_HEAT_COMPENSATED_TEMPERATURE,
    BSEC_OUTPUT_SENSOR_HEAT_COMPENSATED_HUMIDITY
  };
  iaqSensor.updateSubscription(sensorList, 6, BSEC_SAMPLE_RATE_LP);


  //####--- DSM501A ---####-----------------------------------------

  pinMode(pin10, INPUT);
  pinMode(pin25, INPUT);

  starttime = millis();

  attachInterrupt(digitalPinToInterrupt(pin10), interrupcion_10, CHANGE);
  attachInterrupt(digitalPinToInterrupt(pin25), interrupcion_25, CHANGE);

}
//------------------------------- FUNCION PRINCIPAL ---------------------------------------------------
void loop(void)
{

  //####--- BME680 ---####-----------------------------------------

  //unsigned long time_trigger = millis();
  if (iaqSensor.run()) { // If new data is available
    output = String(iaqSensor.pressure);
    output += "," + String(iaqSensor.iaqAccuracy);
    output += "," + String(iaqSensor.temperature);
    output += "," + String(iaqSensor.humidity);
```

```
112     output += "," + String(iaqSensor.staticIaq);
113     output += "," + String(iaqSensor.co2Equivalent);
114   }
115
116
117   //####--- DSM501A ---####----------------------------------------
118
119   if ((millis() - starttime) > t_muestreo)
120   {
121     tasa_10 = (duracion_10) / (t_muestreo * 10.0);
122     concentracion_10 = 0.5831 * pow(tasa_10, 3) - 15.924 * pow(tasa_10, 2) + 729.37 * tasa_10 - 82.523;
123
124     tasa_25 = (duracion_25) / (t_muestreo * 10.0);
125     concentracion_25 = 0.5831 * pow(tasa_25, 3) - 15.924 * pow(tasa_25, 2) + 729.37 * tasa_25 - 82.523;
126
127     con_pm25 = concentracion_10 - concentracion_25;
128     _con_pm25 = con_pm25 < 0.00 ? 0.00 : con_pm25;
129     pm25 = _con_pm25 *  0.002179;
130
131     duracion_10 = 0;
132     duracion_25 = 0;
133     starttime = millis();
134   }
135
136   if (output == "" && pm25 != 0)
137   {
138     msg =
139 String(tasa_10)+","+String(tasa_25)+","+String(concentracion_10)+","+String(concentracion_25)+","+String(pm25);
140     Serial.println(msg);
141   }
142   else if (output != "" && pm25 == 0)
143   {
144     msg = output;
145     Serial.println(msg);
146   }
147   else if (output != "" && pm25 != 0)
148   {
149     msg = output + "," + String(pm25);
150     Serial.println(msg);
151   }
152   output = "";
153   pm25 = 0;
154 }
```

Anexo 10. Transmisión de la data desde el Nodo Final hacia la Red Mesh

```
1   //Scheduler userScheduler;
2   painlessMesh mesh;
3   void recibirMensaje( uint32_t from, String &msg )
4   {
5     Serial.printf("Respuesta: Recibido de %u msg=%s\n", from, msg.c_str());
6   }
7   String msg;
8
9
10  //------------------------------- CONFIGURACION INICIAL ----------------------------------------
11  -----
12  void setup(void)
13  {
14    Serial.begin(115200);     //Inicializacion de serial
15  Serial.println("Serial Inicializada");
16    mesh.setDebugMsgTypes( ERROR | STARTUP);
17    mesh.init(NOMBRE_MALLA, CONTRASEÑA_MALLA, PUERTO_MALLA);
18    mesh.onReceive(&recibirMensaje);
19  }
20  //-------------------------------- FUNCION PRINCIPAL ------------------------------------------
21  ---
22  void loop()
23  {
24    mesh.update();
25    int rssi = WiFi.RSSI();
26    if(Serial.available() > 0)
27    {
28      String data = Serial.readStringUntil('\n');
29      data = data.substring(0, data.length() - 1);
30      Serial.println(data);
31      mesh.sendBroadcast(data+","+String(rssi)+",3");
32    }
33  }
```

Anexo 11. Resección de la data desde la Red Mesh al Nodo Central

```
#include <painlessMesh.h>


#define   NOMBRE_MALLA     "RedSensor"
#define   CONTRASEÑA_MALLA   "2015052397"
#define   PUERTO_MALLA       5555

unsigned long tiempo;
int rssi;
int _rssi = 0;
painlessMesh mesh;

void recibirMensaje( uint32_t from, String &msg ) {
  Serial.println("value," + String(from) + "," + String(msg.c_str()));

}
void changedConnectionCallback() {
  auto nodes = mesh.getNodeList(true);
  String str = "network,";
  for (auto && id : nodes)
    str += String(id) + String(",");

  Serial.println(str);
}
void setup() {
  Serial.begin(115200);
  Serial.setTimeout(100);
  mesh.setDebugMsgTypes( ERROR | STARTUP);
  mesh.init(NOMBRE_MALLA, CONTRASEÑA_MALLA, PUERTO_MALLA);
  mesh.onReceive(&recibirMensaje);
  mesh.onChangedConnections(&changedConnectionCallback);

  tiempo = millis();

}

void loop() {
  mesh.update();
  if ((millis() - tiempo) > 5000)
  {
    rssi = WiFi.RSSI();
    uint32_t id = mesh.getNodeId();
    if (rssi != _rssi || rssi >= 0) {
      Serial.println("value," + String(id) + "," + String(rssi) + ",0");
    }
    _rssi = rssi;
    tiempo = millis();
  }


}
```

Anexo 12. Transmisión de la data desde la Red WLAN al Nodo Central

```
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <ESP8266NetBIOS.h>

ESP8266WiFiMulti wifi_multi;

const char* ssid1 = "SALMOS 2.4GHZ";
const char* pass1 = "0799376060";

const char* ssid2 = "tesis";
const char* pass2 = "12345678";

const char* ssid3 = "BOB TORONJA";
const char* pass3 = "2015052397";

uint16_t connectTimeOutPerAP=5000;

WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  Serial.setTimeout(100);
  Serial.println("Serial inicializada");
  wifi_multi.addAP(ssid1, pass1);
  wifi_multi.addAP(ssid2, pass2);
  wifi_multi.addAP(ssid3, pass3);
  Serial.print("Conectando a Wi-Fi...");
  while (wifi_multi.run(connectTimeOutPerAP) != WL_CONNECTED)
  {
    Serial.print(".");
    delay(1000);
  }
  Serial.println();
  Serial.print("Conectado a: ");
  Serial.println(WiFi.SSID());
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
  server.begin();
  Serial.println("Servidor inicializado");
  NBNS.begin("Modulo1");
}

void loop() {
  WiFiClient cliente = server.available();
  if (cliente)
  {
    while (cliente.connected())
    {
      //    while (cliente.available()>0)
      //    {
      //    char d_in_wifi = cliente.read();
      //    Serial.write(d_in_wifi );
      //    }
      while (Serial.available() > 0)
      {
```

```
56            String data = Serial.readStringUntil('\n');
57            data = data.substring(0, data.length() - 1);
58            cliente.print(data);
59          }
60          delay(50);
61        }
62      }
63
64      if (Serial.available() > 0)
65      {
66        Serial.print(char(Serial.read()));
67      }
68    }
```

Anexo 13. Matriz de Consistencia

PROBLEMA	OBJETIVO	HIPOTESIS	VARIABLES	INDICADOR
¿Con un sistema IoT se logrará monitorear remotamente la calidad del aire en ambientes interiores de la Facultad de Ingeniería de la Universidad Privada de Tacna?	Diseñar un sistema IoT para el monitoreo remoto de la calidad del aire en ambientes interiores de la Facultad de Ingeniería de la Universidad Privada de Tacna.	El diseño de un sistema IoT permite monitorear remotamente la calidad del aire en ambientes interiores de la Facultad de Ingeniería de la Universidad Privada de Tacna.	**V. INDEPENDIENTE** Sistema IoT	• Procesamiento y transmisión de datos. • Base de Datos. • Interfaz Gráfica.
			V. DEPENDIENTE Monitoreo de Calidad del Aire	• Cantidad de Gases Contaminantes • Temperatura • Humedad • PM2.5

Anexo 14.