UNIVERSIDAD UTE

FACULTAD DE CIENCIAS DE LA INGENIERÍA E INDUSTRIAS

CARRERA DE INGENIERÍA INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

DISEÑO DE UN PROTOTIPO PARA CONTROL, MONITOREO Y ANÁLISIS DE LOS NIVELES DE CONTAMINACIÓN EN UN SECTOR DE LA CIUDAD DE QUITO MEDIANTE SENSORES IoT

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO

DE INGENIERO (A) EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

QUISHPE TOPÓN ANDRÉS FRANCISCO

DIRECTOR: ING. GÁLVEZ HUGO

Quito, Septiembre 2021

FORMULARIO DE REGISTRO BIBLIOGRÁFICO

TRABAJO DE TITULACIÓN

DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD:	**1725264996**
APELLIDO Y NOMBRES:	**QUISHPE TOPÓN ANDRÉS FRANCISCO**
DIRECCIÓN:	**SANGOLQUÍ, INÉS GANGOTENA Y ATAHUALPA #3-61**
EMAIL:	**andresf13@hotmail.es**
TELÉFONO FIJO:	**022330-610**
TELÉFONO MOVIL:	**0999709284**

DATOS DE LA OBRA	
TÍTULO:	**DISEÑO DE UN PROTOTIPO PARA CONTROL, MONITOREO Y ANÁLISIS DE LOS NIVELES DE CONTAMINACIÓN EN UN SECTOR DE LA CIUDAD DE QUITO MEDIANTE SENSORES IoT**
AUTOR O AUTORES:	**QUISHPE TOPÓN ANDRÉS FRANCISCO**
FECHA DE ENTREGA DEL PROYECTO DE TITULACIÓN:	**Septiembre del 2021**
DIRECTOR DEL PROYECTO DE TITULACIÓN:	**ING. GÁLVEZ HUGO**
PROGRAMA	**PREGRADO** [x] **POSGRADO** [ ]
TÍTULO POR EL QUE OPTA:	**INGENIERO EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN**
RESUMEN: Mínimo 250 palabras	En el presente trabajo de titulación se diseñó un prototipo para el control, monitoreo y análisis de los niveles de

	contaminación que existen dentro de un sector de Quito, el mismo que se elaboró mediante el uso de sensores IoT los cuales, generaron datos llegando a detectar contaminantes del aire, estos fueron recolectados utilizando un Arduino ESP32s, permitiendo la interpretación de los datos mediante el uso de librerías especialmente creadas para los sensores de la serie MQ (basados en un termo reactivo) y el uso de fórmulas para el sensor de partículas, el resultado de esos datos se envió mediante protocolo MQTT. El prototipo físico cuenta con otros componentes para una óptima experiencia del usuario, siendo una pantalla LCD, para la visualización de datos al instante, junto con un sensor de temperatura y humedad relativa del ambiente. El Aplicativo web IoT cuenta con un servicio en la nube, el cual se trata de un servidor VPS (Virtual Private Server) y un dominio exclusivo del sitio web (airmonitor-ute.live), donde se encuentra y accede al aplicativo web. Dentro de este aplicativo se usa MongoDB para el almacenamiento con datos no relacionales, el uso de un Broker MQTT que permitió el intercambio de tópicos y payloads entre los diferentes participantes (Dispositivo, usuario y servidor VPS) y el uso de node.js para la ejecución del aplicativo, todos estos servicios se

	levantaron dentro de Docker con la creación de contenedores de Linux, los cuales almacenan los servicios mencionados. Se decidió utilizar la metodología SCRUM tanto para la investigación como para el desarrollo, otorgándonos herramientas como bitácoras y registros de actividades. El uso de la IoT para la calidad del aire es de una gran ayuda y nos permitió la exploración de manera automática y en tiempo real de sus datos.
PALABRAS CLAVES:	Arduino, Prototipo, Contaminación, Calidad de Aire, Aplicativo Web, Internet de las Cosas, IoT, MQTT, TCP.
ABSTRACT:	In this degree work, a prototype was designed for the control, monitoring and analysis of pollution levels that exist within a sector of Quito, which was developed through the use of IoT sensors which generated data to detect air pollutants, These were collected using an Arduino ESP32s, allowing the interpretation of the data through the use of libraries specially created for the MQ series sensors (based on a thermo reactive) and the use of formulas for the particle sensor, the result of these data was sent via MQTT protocol. The physical prototype has other components for an optimal user experience, being an LCD screen, for instant data visualization, together with an ambient temperature and relative humidity

	sensor. The IoT web application has a cloud service, which is a VPS server (Virtual Private Server) and an exclusive website domain (airmonitor-ute.live), where the web application is located and accessed. Within this application MongoDB is used for storage with non-relational data, the use of a MQTT Broker that allowed the exchange of topics and payloads between the different participants (Device, user and VPS server) and the use of node.js for the execution of the application, all these services were raised within Docker with the creation of Linux containers, which store the aforementioned services. It was decided to use the SCRUM methodology for both research and development, giving us tools such as logs and activity logs. The use of IoT for air quality is a great help and allowed us to explore automatically and in real time its data.
KEYWORDS	Arduino, Prototype, Pollution, Air Quality, Web Application, Internet of Things, IoT, MQTT, TCP.

Se autoriza la publicación de este Proyecto de Titulación en el Repositorio Digital de la Institución.

f:_____

QUISHPE TOPÓN ANDRÉS FRANCISCO

172526499-6

DECLARACIÓN Y AUTORIZACIÓN

Yo, **QUISHPE TOPÓN ANDRÉS FRANCISCO,** CI 172526499-6 autor/a del trabajo de titulación: **Diseño de un prototipo para control, monitoreo y análisis de los niveles de contaminación en un sector de la ciudad de Quito mediante sensores IoT** previo a la obtención del título de **GRADO ACADÉMICO COMO APARECE EN EL CERTIFICADO DE APROBACIÓN ACADÉMICA** en la Universidad UTE.

1. Declaro tener pleno conocimiento de la obligación que tienen las Instituciones de Educación Superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación de grado para que sea integrado al Sistema Nacional de información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2. Autorizo a la BIBLIOTECA de la Universidad UTE a tener una copia del referido trabajo de titulación de grado con el propósito de generar un Repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Quito, septiembre del 2021.

f:_____

QUISHPE TOPÓN ANDRÉS FRANCISCO

172526499-6

DECLARACIÓN JURAMENTADA DEL AUTOR

Yo, **QUISHPE TOPÓN ANDRÉS FRANCISCO**, portador de la cédula de identidad **N° 1725264996**, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad UTE puede hacer uso de los derechos correspondientes a este trabajo, según los establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

f:_____

QUISHPE TOPÓN ANDRÉS FRANCISCO

172526499-6

CERTIFICACIÓN DEL TUTOR

En mi calidad de tutor, certifico que el presente trabajo de titulación que lleva por título **Diseño de un prototipo para control, monitoreo y análisis de los niveles de contaminación en un sector de la ciudad de quito mediante sensores IoT** para aspirar al título de **GRADO ACADÉMICO COMO APARECE EN EL CERTIFICADO DE APROBACIÓN ACADÉMICA** fue desarrollado por **QUISHPE TOPÓN ANDRÉS FRANCISCO,** bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería e Industrias; y que dicho trabajo cumple con las condiciones requeridas para ser sometido a las evaluación respectiva de acuerdo a la normativa interna de la Universidad UTE.

ING. GÁLVEZ HUGO

DIRECTOR DEL TRABAJO

C.I. 1707796536

DEDICATORIA

Dedico esta tesis a mis padres **Francisco Quishpe** y **Olga Topón** quienes fueron un gran apoyo emocional toda mi vida y quienes me alentaron para continuar, cuando parecía que me iba a rendir.

A mi hermana **Andrea** quien fue mi inspiración como profesional y guía necesaria.

A mi sobrina **Ana Paula** quien me dio su amor, ternura y fuerzas, sin decir una sola palabra.

A todos mis amigos y familiares quienes apoyaron para escribir y concluir esta tesis.

A mis maestros quienes nunca desistieron al enseñarme, aun sin importar que muchas veces no ponía atención en clase.

Para ellos es esta dedicatoria de tesis, pues es a ellos a quienes se las debo por su apoyo incondicional. Sin ustedes nada esto hubiese sido posible.

"No está mal, para un loco del barrio"

Andrés.

ÍNDICE DE CONTENIDOS

Contenido **Página**

ÍNDICE DE TABLAS

Contenido **Página**

ÍNDICE DE FIGURAS

Contenido **Página**

ÍNDICE DE ANEXOS

Contenido **Página**

RESUMEN

En el presente trabajo de titulación se diseñó un prototipo para el control, monitoreo y análisis de los niveles de contaminación que existen dentro de un sector de Quito, el mismo que se elaboró mediante el uso de sensores IoT los cuales, generaron datos llegando a detectar contaminantes del aire, estos fueron recolectados utilizando un Arduino ESP32s, permitiendo la interpretación de los datos mediante el uso de librerías especialmente creadas para los sensores de la serie MQ (basados en un termo reactivo) y el uso de fórmulas para el sensor de partículas, el resultado de esos datos se envió mediante protocolo MQTT. El prototipo físico cuenta con otros componentes para una óptima experiencia del usuario, siendo una pantalla LCD, para la visualización de datos al instante, junto con un sensor de temperatura y humedad relativa del ambiente. El Aplicativo web IoT cuenta con un servicio en la nube, el cual se trata de un servidor VPS (Virtual Private Server) y un dominio exclusivo del sitio web (airmonitor-ute.live), donde se encuentra y accede al aplicativo web. Dentro de este aplicativo se usa MongoDB para el almacenamiento con datos no relacionales, el uso de un Broker MQTT que permitió el intercambio de tópicos y payloads entre los diferentes participantes (Dispositivo, usuario y servidor VPS) y el uso de node.js para la ejecución del aplicativo, todos estos servicios se levantaron dentro de Docker con la creación de contenedores de Linux, los cuales almacenan los servicios mencionados. Se decidió utilizar la metodología SCRUM tanto para la investigación como para el desarrollo, otorgándonos herramientas como bitácoras y registros de actividades. El uso de la IoT para la calidad del aire es de una gran ayuda y nos permitió la exploración de manera automática y en tiempo real de sus datos.

Palabras Clave: Arduino, Prototipo, Contaminación, Calidad de Aire, Aplicativo Web, Internet de las Cosas, IoT, MQTT, TCP.

ABSTRACT

In this degree work, a prototype was designed for the control, monitoring and analysis of pollution levels that exist within a sector of Quito, which was developed through the use of IoT sensors which generated data to detect air pollutants, These were collected using an Arduino ESP32s, allowing the interpretation of the data through the use of libraries specially created for the MQ series sensors (based on a thermo reactive) and the use of formulas for the particle sensor, the result of these data was sent via MQTT protocol. The physical prototype has other components for an optimal user experience, being an LCD screen, for instant data visualization, together with an ambient temperature and relative humidity sensor. The IoT web application has a cloud service, which is a VPS server (Virtual Private Server) and an exclusive website domain (airmonitor-ute.live), where the web application is located and accessed. Within this application MongoDB is used for storage with non-relational data, the use of a MQTT Broker that allowed the exchange of topics and payloads between the different participants (Device, user and VPS server) and the use of node.js for the execution of the application, all these services were raised within Docker with the creation of Linux containers, which store the aforementioned services. It was decided to use the SCRUM methodology for both research and development, giving us tools such as logs and activity logs. The use of IoT for air quality is a great help and allowed us to explore automatically and in real time its data.

Keywords: Arduino, Prototype, Pollution, Air Quality, Web Application, Internet of Things, IoT, MQTT, TCP.

1. INTRODUCCIÓN

1. INTRODUCCIÓN

Al hablar de contaminación nos referimos a un entorno en el cual existen elementos o sustancias que normalmente no deberían estar allí, los cuales afectan al equilibrio del ecosistema, “El ser humano existe gracias a que la naturaleza proporciona beneficios para la supervivencia, pero el uso indiscriminado de los recursos naturales han ido ocasionando problemas ambientales como el perjuicio de la biodiversidad, contaminación ambiental, cambio climático, adelgazamiento de la capa de ozono, entre otros problemas” (Casa et al., 2019), la calidad del aire se ve afectada y en consecuencia la salud humana, al respirar sustancias que no deberíamos o que simplemente por descuido del ser humano.

La presente investigación abordará uno de los factores más importantes a tomar en cuenta para la realización del problema de estudio del plan de titulación, el cual es la importancia de la calidad del aire con respecto a los niveles de contaminación que existen hoy en día, dando como lugar el uso de herramientas y sensores del prototipo construido para la implementación de tecnología IoT, el cual nos dará como resultado los valores estadísticos, los cuales serán filtrados, visualizados e interpretados fácilmente con un interfaz web y recursiva de uso amigable para el usuario final.

CALIDAD DEL AIRE

Según el último plan nacional de calidad de aire de Ecuador, en el cual se investigaba los efectos de la contaminación del Aire para la salud y se destaca la situación que en el país, “se cuenta con pocas investigaciones sobre los efectos de la contaminación del aire en la salud de las personas, estos temas no han sido incluidos en los programas de desarrollo urbano y no se han llevado a cabo estudios epidemiológicos relacionados con la contaminación del aire” (Hernández et al., 2010), lo que da como consecuencias la desinformación sobre el nivel de calidad de aire, dando a entender que la contaminación ambiental es uno de los factores determinantes para la salud de un país.

En el Ecuador se realizó un estudio sobre enfermedades respiratorias en el cual se determina que, a partir de julio de 200, hasta la actualidad aumento el nivel de mortalidad, “Entre las afecciones más relevantes generadas por esta problemática se destacan entre otros, la aparición de enfermedades a los ojos, problemas a las vías respiratorias, granos, dolores a la garganta, tos, hongos en los pies, náusea y dolores de cabeza” (Hernández et al., 2010), en la tabla 1 se muestra el número de casos registrados de los infecciones respiratorias en el país:

Tabla 1 Casos registrados de Infecciones respiratorias agudas en el Ecuador.

Provincia	Casos registrados de Infecciones Respiratorias Agudas
Carchi	18 360
Imbabura	30 302
Pichincha	140 005
Cotopaxi	23 092
Tungurahua	25 200
Bolívar	2 642
Chimborazo	31 764
Cañar	12 749
Azuay	41 339
Loja	36 660
Esmeraldas	51 334
Manabí	156 152
Los Ríos	60 708
Guayas	232 540
El Oro	68 190
Sucumbíos	12 734
Napo	1 390
Orellana	5 314
Pastaza	6 871
Morona Santiago	21 148
Zamora Chinchipe	11 533
Galápagos	1 717
TOTAL	991 744

Fuente: (Hernández et al., 2010)**.**

Dentro de Ecuador también se muestra los resultados de un estudio realizado en la ciudad de Cuenca, en el cual muestra que, "con 505.585 habitantes, los promedios de partículas PM10 superan la guía de la OMS, ubicándose entre los objetivos provisionales IT3 e IT2 de la guía; el incremento de la mortalidad por enfermedades cardiopulmonares y cáncer de pulmón estaría entre el 3 y el 9%" (Palacios Espinoza & Espinoza Molina, 2014), por consecuencia, se puede percibir la importancia de la calidad del aire, el cual perjudica ayuda al incremento de mortalidad relacionado con enfermedades respiratorias.

Según la OMS y sus guías para la calidad del aire con respecto al PM (Material Particulado) se muestra los niveles correspondientes dentro de la tabla 2:

Tabla 2 Guías para Material Particulado.

Guías de calidad del aire para Material Particulado				
Contaminante	**Media Anual**	**Media de 24 Horas**	**Nombre Unidad**	**Símbolo**
PM 2.5	10	25	Microgramos / metro cúbico	$\mu g/m^3$
PM 10	20	50		

Fuente: (OMS, 2005)

Los datos del material particulado de este trabajo de investigación son particulado fino (PM2.5) y material particulado grueso (PM10), hay que tener en cuenta que, “dentro de las ciudades hay grupos más afectados, sea por su actividad, lugar de trabajo, lugar de residencia, o cantidad de horas en ambientes externos. Los grupos que sufren más la contaminación pueden coincidir con los grupos que acceden a menos áreas verdes y/o están más expuestos ante amenazas naturales, entre otras variables.” (Rodríguez-Guerra & Cuvi, 2019) , por lo tanto, dentro de la ciudad es el lugar donde se encuentra la mayor contaminación del aire, en el cual existe vías de alto tránsito o zonas industriales que afectan a la calidad del aire.

CONTAMINANTES

La contaminación del aire también se compone por gases que serán nombrados a continuación y los mismos serán monitoreados dentro del proyecto de titulación:

- El **Monóxido de carbono (CO)** “Es un gas incoloro no irritante sin olor o sabor. Se le encuentra tanto en el aire puertas adentro como al aire libre” (Agencia para Sustancias Tóxicas y Registro de Enfermedades, 2012), sin embargo, el monóxido de carbono es también un contaminante del aire, cuando el mismo tiene concentraciones por encima de lo normal.
  Este gas puede llegar a tener varias fuentes de producción y con respecto a estas, “en la atmósfera proviene de fuentes naturales como incendios forestales, la degradación de clorofila, animales marinos y descomposición de materia vegetal, pero mayormente de fuentes antropogénicas incluyendo la oxidación de metano, procesos tecnológicos de combustión, procesos industriales, incendios urbanos, incineración de materia orgánica, uso de sistemas de calefacción, quema de biomasa, oxidación de hidrocarburos diferentes al metano, combustión de combustibles fósiles como gasolina y diésel y, en general, de la combustión incompleta de materiales que contienen carbono” (CeMCAQ, 2017), en la figura 1 se puede observar las principales fuentes de CO.

Figura 1 Origen del CO

Fuente: (CeMCAQ, 2017).

- El **dióxido de carbono (CO_2),** “Es un gas de efecto invernadero, es el contaminante que está causando en mayor medida el calentamiento de la Tierra. Si bien todos los seres vivos emiten dióxido de carbono al respirar, éste se considera por lo general contaminante cuando se asocia con coches, aviones, centrales eléctricas y otras actividades humanas que requieren el uso de combustibles fósiles como la gasolina y el gas natural” (NatGeo, 2010), en la figura 2 se puede ver una referencia a la contaminación del aire por la industria.

Figura 2 Contaminación del Aire por industria.

Fuente: (NatGeo, 2010).

- El **Ozono (O_3),** “Es un gas altamente reactivo compuesto por tres átomos de oxígeno. Aparece fundamentalmente en dos áreas de la atmósfera, la estratosfera y la troposfera. La presencia de 0_3, en la troposfera es indeseable, ya que se considera un contaminante con capacidad irritante sobre los tejidos vivos y sobre muchos materiales.”(Aránguez et al., 1999).

  El valor actual para proteger a los seres humanos de este contaminante según la OMS se puede observar en la tabla 3:

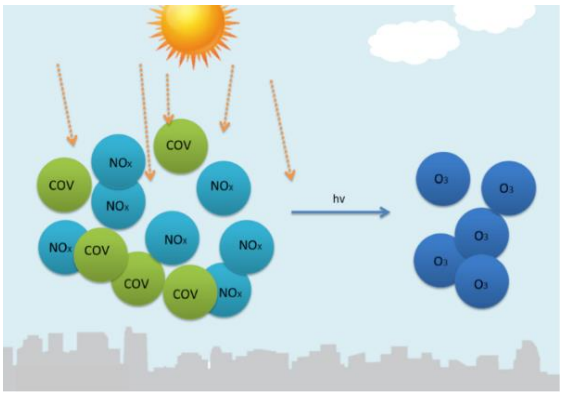
Tabla 3 Guía OMS para el Ozono

Guías de calidad del aire para Ozono			
Contaminante	**Media de 8 Horas**	**Nombre Unidad**	**Símbolo**
O_3 (Ozono)	100	Microgramos / metro cúbico	$\mu g/m^3$

Fuente: (OMS, 2005).

Este gas es considerado un contaminante secundario debido a que, “su formación depende de la presencia de otros compuestos como los óxidos de nitrógeno (NOX) cuyas fuentes son: vehículos automotores y emisiones industriales entre otras, y los compuestos orgánicos volátiles (COV´s), cuya fuente de emisión proviene de vehículos automotores, y actividades emisoras como talleres de laminado y pintura, alfarería, estaciones de servicio (gasolineras), y la industria de diversos ramos ”(SEMADET, 2017), en la figura 3 se puede ver de manera gráfica la creación del ozono.

Figura 3 Formación del ozono contaminante



Fuente: (SEMADET, 2017).

- El **Dióxido de Nitrógeno (NO_2)** “Es un gas sin olor, color ni sabor, que constituye el 78% del aire. Aunque en condiciones normales no es perjudicial para la salud, se puede combinar con oxígeno para formar diversos óxidos de nitrógeno. La importancia biológica del óxido nítrico (NO) y del dióxido de nitrógeno (NO,) es la mejor estudiada; ambos se consideran contaminantes ambientales, y son los más abundante óxidos de nitrógeno producidos por el hombre en áreas urbanas” (Aránguez et al., 1999).

  El valor actual para proteger a los seres humanos de este contaminante según la OMS se representa en la tabla 4:

Tabla 4 Guías OMS para el Dióxido de Nitrógeno

Guías de calidad del aire para Dióxido de Nitrógeno				
Contaminante	**Media Anual**	**Media de una Hora**	**Nombre Unidad**	**Símbolo**
NO_2 (Dióxido de nitrógeno)	40	200	Microgramos / metro cúbico	$\mu g/m^3$

Fuente: (OMS, 2005).

Tomando en cuenta los niveles de contaminación de un determinado lugar “La calidad del aire se ve alterada notablemente por la presencia en el mismo de una serie de sustancias de diversa naturaleza que resultan tóxicas para la salud y que conocemos como contaminantes” (Sánchez Bayle et al., 2019), junto con la realización e implementación del prototipo podemos lograr establecer que tan contaminado se encuentra el entorno de dicho lugar, llegando a mejorar la calidad de vida de manera significativa al dar una información efectiva, sobre la calidad de aire de manera estadística y amigable para el cliente final.

Otro aspecto para resaltar es ayudar a diferenciar entre los diferentes tipos de niveles de contaminación donde, “La contaminación medioambiental supone, por lo tanto, una concentración excesiva en la atmósfera de dióxido de nitrógeno (NO_2), monóxido de carbono (CO), dióxido de azufre (SO_2), hidrocarburos como el benceno, y las partículas en suspensión de tamaño inferior a 2,5 μ y 10 μ, entre otros” (Sánchez Bayle et al., 2019), que puede llegar a existir dentro del perímetro del presente estudio, para poder proyectarlo dentro de paneles informativos.

EL INTERNET DE LAS COSAS (IoT) Y EL MEDIO AMBIENTE

La IoT dentro de los últimos años se encuentra en un constante crecimiento, fomentando al desarrollo de aplicativos que pretenden ayudar en el control del medioambiente. El uso de estas tecnologías puede inclusive disminuir la presencia de los contaminantes generando una información eficaz en su momento para evitar filtraciones masivas de contaminantes provenientes de alguna fabrica o similar, al darnos alerta de tal presencia se actuaría de manera oportuna.

Según un trabajo de titulación de la Universidad Nacional Tecnológica de Lima Sur dice que uno de los beneficios de la IoT en cuanto al cuidado del medio ambiente se refiere a la optimización de la gestión de residuos sólidos, realizada mediante contenedores inteligentes con tecnología IoT para poder reducir la contaminación donde se concluye que “La propuesta de diseño de contenedor ecológico inteligente es una alternativa de solución para mejorar la gestión de residuos sólidos en temas económicos, sociales y ambientales” (Huaman Chiroque, 2019) ya que la adquisición de estos componentes son muy económicos y de fácil implementación, se puede observar la propuesta en la figura 4.

Figura 4 Propuesta de Diseño de Contenedores Inteligentes.

Fuente: (Huaman Chiroque, 2019)

Un estudio de grado realizado por la Universidad de Zaragoza, en España, utilizó tecnología Arduino para realizar un sistema de medida de gases contaminantes, utilizando Arduino como un administrador de sensores IoT ambientales que ayuden a determinar la calidad del aire, donde la posibilidad que nos otorga Arduino para hacer este tipo de sistemas es muy grande ya que son elementos muy económicos pero capaces de registrar datos con sensores accesibles, tanto para interiores como para exteriores.

También existen complicaciones en el mundo IoT las cuales tienen que ver con el tipo de conectividad que se tendrá al momento de la recepción y representación de los datos, para un proyecto de grado la Escuela Superior Politécnica de Chimborazo investigó la posibilidad de implementar un sistema de monitoreo de polución de aire, la cual contaba con diferentes estaciones denominadas “Nodo” con redes inalámbricas independientes llegaron a la conclusión que “la interoperabilidad de plataformas IoT no es la opción más conveniente para la implantación de una red de monitoreo pues necesitan programación personalizada por cada nodo, lo que conlleva gran cantidad de tiempo y dificulta el proceso de implementación de la red” (Coronel & Tenelanda, 2016) , este tipo de conectividades independientes lleva a tener consigo una configuración en específico y personalizada, llevando a traer conflictos si es que se configura de manera errónea, es por eso que la mejor opción en la conectividad es el uso de terceros en cuanto a redes, es decir utilizar protocolos de comunicación dentro de una red en común (privada o pública) que se puedan conectar a internet para poder acceder a servidores en la nube capaces de almacenar la información generada por los sistemas IoT.

La contaminación puede venir de varios factores, tomando en cuenta que hoy en día existen varios tipos, el cual por mencionar alguno trata de una contaminación acústica dentro del Medio Ambiente es decir el excesivo ruido provocado por diferentes factores dentro de un entorno, con este tipo de contaminación, se llevó a cabo un estudio realizado por la Pontificia

Universidad Católica Del Perú en el cual se realizó un diseño de un sistema de monitoreo de contaminación acústica urbana bajo una plataforma IoT la cual utilizaría sensores de ruido siendo estos dispositivos IoT con una alta precisión y medición de manera continua, los cuales fueron conectados a un sistema Arduino siendo este para el nodo o estación y el sistema Raspberry PI como Gateway el cual recibía los datos de cada nodo. Este tipo de sistemas, nos ayudan a identificar sectores en donde la contaminación acústica urbana es excesiva y dañina para la salud siendo más preciso para nuestro oído, se puede observar el diseño de esté, en la figura 5.

Figura 5 Diseño 3D del Sistema de Monitoreo Acustico IoT, nodo (izquierda) y gateway (derecha).

Fuente: (Del et al., 2020)

El uso de tecnologías IoT dentro de lo que es el medio ambiente y su preservación nos dice que “La mayoría de las aplicaciones del IoT al medioambiente se llevan a cabo mediante sensores. Existen dispositivos de muy reducido tamaño que generan datos útiles para tomar medidas en pro del medioambiente” (ZemsaniaGlobalGroup, 2019), dentro del mercado de IoT existen varias opciones para generar información a base de sensores de todo tipo, ayudando de manera significativa a la toma de decisiones en base a datos generados de manera oportuna, los cuales se pueden administrar en beneficio del cuidado ambiental, siendo estos los siguientes representados en la figura 6:

Figura 6 Beneficios del IoT en el Medioambiente.

Fuente: (ZemsaniaGlobalGroup, 2019)

Por consecuente, se puede notar que la utilización de tecnologías IoT en el ámbito del Medio Ambiente es de gran ayuda ya que nos proporciona información vital mediante sensores los cuales se pueden utilizar para diversos tipos de contaminación donde estos pueden llegar a ser de un bajo costo pero que generan información útil en la que se puede medir ciertos parámetros en lo que se refiere a la calidad del aire, del agua o permitiendo detectar sustancias químicas dañinas en el ambiente, es indispensable que al momento de utilizar estas tecnologías, tomar en cuenta los aspectos de conectividad, representación de datos, almacenamiento de la información, lenguajes de programación necesarios y materiales utilizados, sean en lo posible componentes que se comportaran de manera óptima, para obtener resultados favorables.

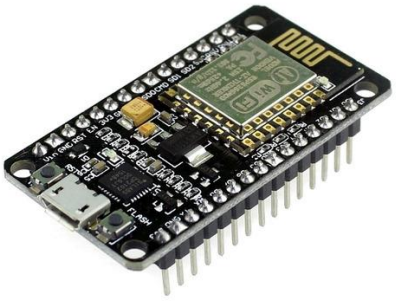
DISPOSITIVOS Y SENSORES IoT

Dentro del mercado IoT existen varios modelos en cuanto costo y beneficio se refiere, siendo de los más populares el Arduino como pilar fundamental de funcionamiento de estos sensores siendo brevemente explicado a continuación junto con los sensores utilizados.

ARDUINO: Es un dispositivo de creación de prototipos electrónicos que trata de una placa para sistemas IoT con código abierto, el Arduino utilizado para el proyecto en cuestión se lo puede observar en la figura 7 y se trata del Arduino ESP32S NodeMCU siendo este modelo el que “Incluye el firmware que se ejecuta en el SoC (Sistema en chip) Wifi ESP32S de Espressif Systems y el hardware que se basa en el módulo ESP-12” (Godoy et al., 2020) permitiendo a los usuarios crear dispositivos electrónicos interactivos.

.

Figura 7 Arduino ESP32S NodeMCU



(Godoy et al., 2020)

Fue utilizado para establecer la conexión entre la nube y los diferentes sensores utilizados dentro del proyecto de una manera muy sencilla.

SENSORES IoT

Dentro del mundo de los sensores, estos vienen a ser la parte más fundamental dentro de un sistema basado en el Internet de las Cosas, ya que son los designados e indispensables en cualquier sistema IoT que se base en la creación de información y datos efectivos dependiendo del uso que se le vaya a dar al prototipo, en el caso del presente proyecto se detallará a continuación los sensores para la calidad del aire.

- **Sensor MQ-135:** Es un sensor utilizado para la adquisición de datos relacionados con la calidad de aire en un determinado lugar y tiene como finalidad la detección adecuada de contaminantes tales como: NH_4, NOx, Alcohol, Benceno, CO_2 y CO así determinar si el aire se encuentra en sus óptimas condiciones (Ver Figura 8).

Figura 8 Sensor MQ-135

Fuente: (Pinzón et al., 2018)

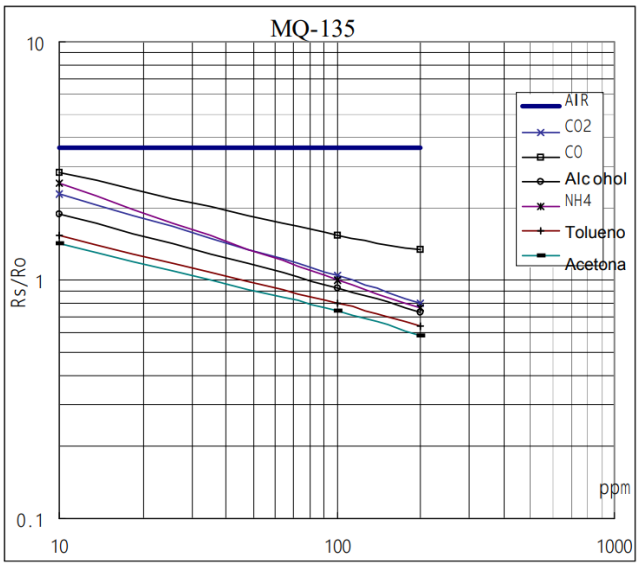
Las características del sensor son presentadas en la tabla 5:

Tabla 5 Características del Sensor MQ-135

Modelo	MQ-135
Voltaje de Alimentación	5v
Señal de Salida	Analógica y digital
Tipo de Sensor	Semiconductor
Unidad	ppm (partes por millón)
Librería Arduino	MQUnifiedsensor
Dimensiones	3.2 cm x 1.9 cm x 2.2 cm

En la figura 9, se muestra la curva de la sensibilidad del sensor MQ-135:

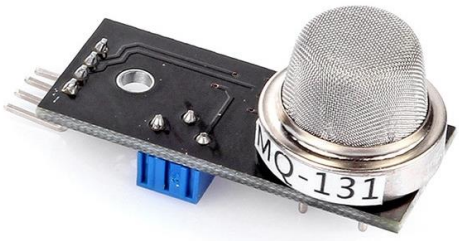
Figura 9 Características de sensibilidad del MQ-135



Fuente: (Olimex, 2012)

- **Sensor MQ-131:** Es un sensor que se utilizó para la lectura de ozono, dentro de un determinado lugar en base a una salida analógica la cual aumenta según la concentración de ozono, cuando mayor es la tensión de la salida, es mayor su concentración (Ver figura 10).

Figura 10 Sensor MQ-131



Fuente: (Zhengzhou Winsen Electronics Technology Co. Ltd, 2014)

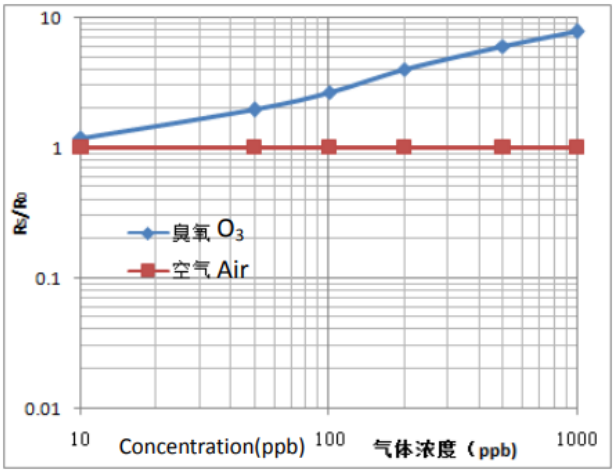
Las características del sensor son presentadas en la tabla 6:

Tabla 6 Características Técnicas del Sensor MQ-131

Modelo	MQ-131
Voltaje de Alimentación	5v
Señal de Salida	Analógica y digital
Tipo de Sensor	Semiconductor
Unidad	ppm (partes por millón)
Librería Arduino	MQUnifiedsensor
Dimensiones	4 cm x 1.7 cm x 2.2 cm

En la figura 11, se muestra la curva de la sensibilidad del sensor MQ-131:

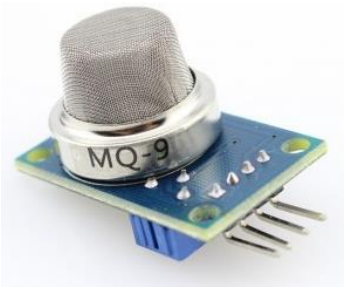
Figura 11 Características de sensibilidad del MQ-131



Fuente: (Zhengzhou Winsen Electronics Technology Co. Ltd, 2014)

- **Sensor MQ-09:** Es un módulo detector de gas que permite la medición de concentración en monóxido de carbono y gases inflamables; en un rango de 10 a 1000 ppm y 100-10000 ppm (partes por millón), estando en presencia de algún gas de los antes mencionados, el sensor emitirá una señal analógica dependiendo de la concentración y la misma nos ayuda a detectar su presencia (Ver Figura 12).

Figura 12 Sensor MQ-09



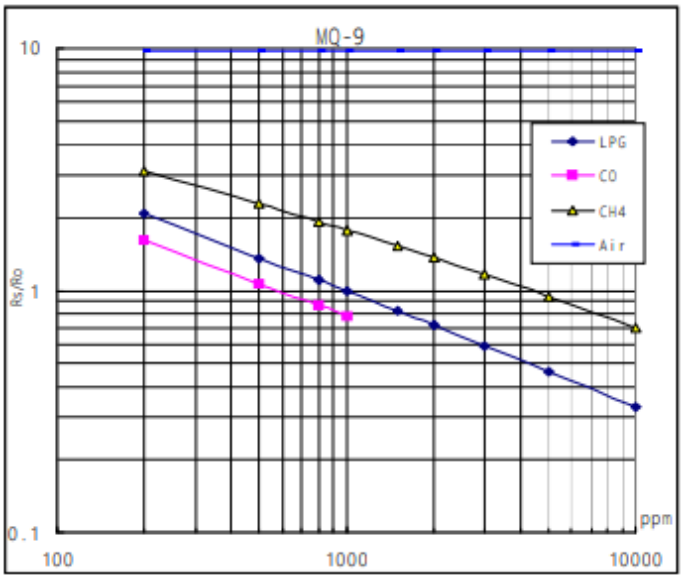
Las características del sensor son presentadas en la tabla 7:

Tabla 7 Características Técnicas del Sensor MQ-09

Modelo	MQ-09
Voltaje de Alimentación	5v
Señal de Salida	Analógica y digital
Tipo de Sensor	Semiconductor
Unidad	ppm (partes por millón)
Librería Arduino	MQUnifiedsensor
Dimensiones	3.2 cm x 2 cm x 2.2 cm

En la figura 13, se muestra la curva de la sensibilidad del sensor MQ-9:

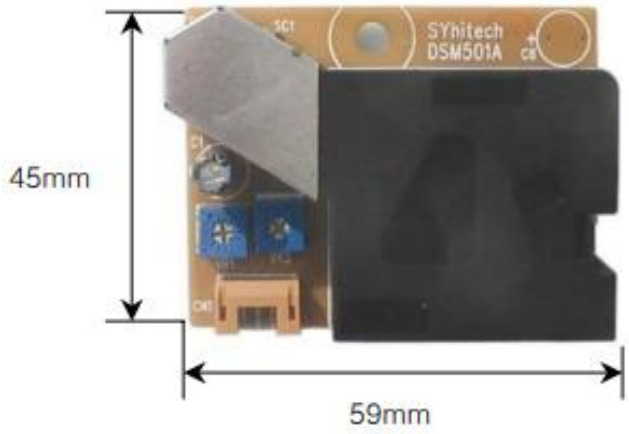
Figura 13 Curva de sensibilidad MQ-9



Fuente: (Hanwei Electronics, 2018)

- **Sensor DSM501A:** Se trata de un sensor de bajo costo y tamaño compacto utilizado para la medición de datos correspondientes al material particulado de tipo 2.5 PM y 1.0 PM en cuanto a su medición cuantitativa en lo que respecta a las partículas, llegando a tener una alta sensibilidad en cuanto a sus diámetros y siendo estos mayores de una micra (Ver Figura 14).

Figura 14 Sensor DSM501A



Fuente: (SYhitech, 2004)

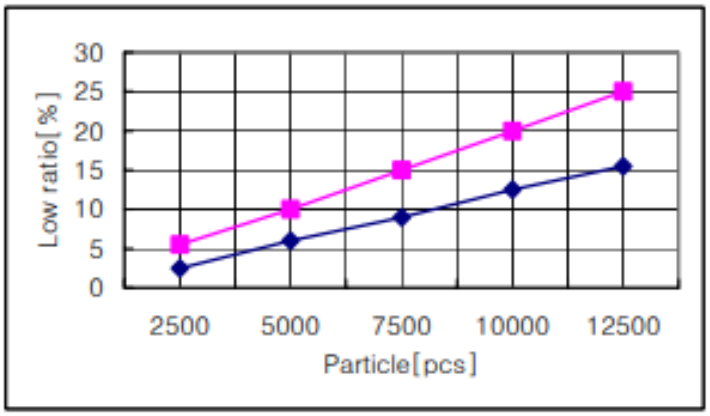
En la tabla 8, se muestra las características del sensor DSM501A:

Tabla 8 Características del Sensor DSM501A

Modelo	DSM501A
Voltaje de Alimentación	5v
Señal de Salida	Analógica
Tipo de Sensor	Laser
Unidad	µg/m^3
Dimensiones	4.5 cm x 5.9 cm x 1.7 cm

En la figura 15, se muestra la curva de la sensibilidad con respecto a las partículas y su relación de bajo voltaje en el sensor DSM501A.

Figura 15 Curva de Sensibilidad Sensor DSM501A



Fuente: (SYhitech, 2004)

- **Módulo DHT22:** Es un sensor que mide la Temperatura y humedad dentro del ambiente, cuenta con una salida digital calibrada siendo esta fiable y con un mayor tiempo de respuesta, en tanto a temperatura y humedad se trate (Ver Figura 16).

Figura 16 Sensor Módulo DHT22

En la tabla 9, se muestra las características otorgadas del sensor DHT22:

Tabla 9 Características del Sensor DHT22

Modelo	DHT22
Voltaje de Alimentación	3.3 a 5v
Corriente de Consumo	Max 2.5 mA
Señal de Salida	Digital
Lectura de Humedad	100% a 2-5%
Lectura de Temperatura	-40 a 80 °C ±0.5 °C
Unidades	°C y %
Librería Arduino	DHTesp
Dimensiones	3.6 cm x 1.8 cm x 1 cm

CONECTIVIDAD Y PROTOCOLOS IoT

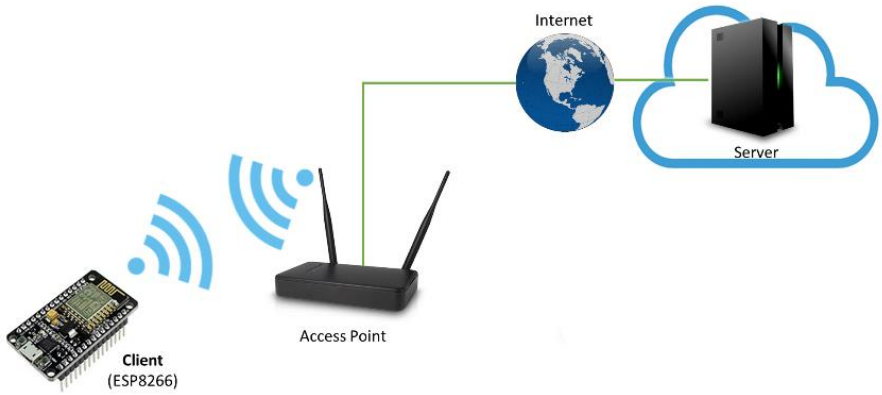
La **conectividad** dentro del proyecto son todos los recursos que nos hace referencia al uso de tecnologías ya industrializadas desplegadas en una zona determinada los cuales serán utilizados para la transportación de información entre dispositivos siendo en este caso el prototipo del monitor de calidad de aire y la nube.

Hoy en día el internet está basado y depende de la conectividad por lo que existen diferentes tipos de recursos y servicios de conectividad probados y que han demostrado su eficacia dentro del mundo de la tecnología IoT, los mismos que son:

- LAN
- Wifi
- GSM, GPRS, LTE
- Bluetooth
- LPWAN (LoRa, SigFox)

Tomando en cuenta la portabilidad del prototipo y la excesiva existencia de redes inalámbricas se tomó la decisión de utilizar el tipo de conectividad Wifi para el transporte de información por internet al servidor en la nube, se puede observar la ilustración de la conectividad en la figura 17.

Figura 17 Ilustración gráfica de la conectividad del prototipo.

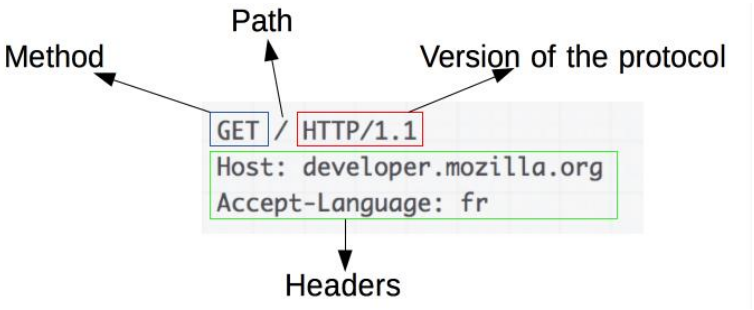


El uso de **protocolos IoT** en cuanto a comunicación se refiere son básicamente un acuerdo en el cual un mensaje o dato se lo va a enviar entre dispositivos y servicios, dando a entender que con respecto a los protocolos utilizados en IoT es la manera fundamental en la cual dos o más partes de un determinado sistema puedan entenderse entre sí, en este caso la misión principal del prototipo IoT determinado como monitor de calidad de aire pueda enviar y recibir mensajes con éxito sin tener interferencia y de manera constante con la nube, usuarios y plataforma IoT, por lo cual se definió y utilizó los siguientes protocolos:

- **HTTP:** Viene de las siglas en ingles “Hypertext Transfer Protocol” o “Protocolo de transferencia de Hipertexto”, el cual es un protocolo sincronizado utilizado diariamente por millones de personas y se lo ve reflejado al utilizar una página web, siendo de manera imperceptible lo que estamos haciendo es generar una conexión o una petición HTTP hacia el servidor para así poder visualizar el contenido que está dentro de determinada página web.

  Las peticiones de HTTP están compuestas por los elementos que se muestran en la figura 18:

Figura 18 Ejemplo de Petición HTTP.



Fuente: (Mozilla.org, n.d.)

Existen varios métodos para diferentes funciones los cuales están detallados en la tabla 10:

Tabla 10 Métodos de petición HTTP

MÉTODOS DE PETICIÓN HTTP	
GET	El método GET solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.
HEAD	El método HEAD pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.
POST	El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.
PUT	El modo PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.
DELETE	El método DELETE borra un recurso en específico.
CONNECT	El método CONNECT establece un túnel hacia el servidor identificado por el recurso.
OPTIONS	El método OPTIONS es utilizado para describir las opciones de comunicación para el recurso de destino.
TRACE	El método TRACE realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso de destino.
PATCH	El método PATCH es utilizado para aplicar modificaciones parciales a un recurso.

(Mozilla.org, n.d.)

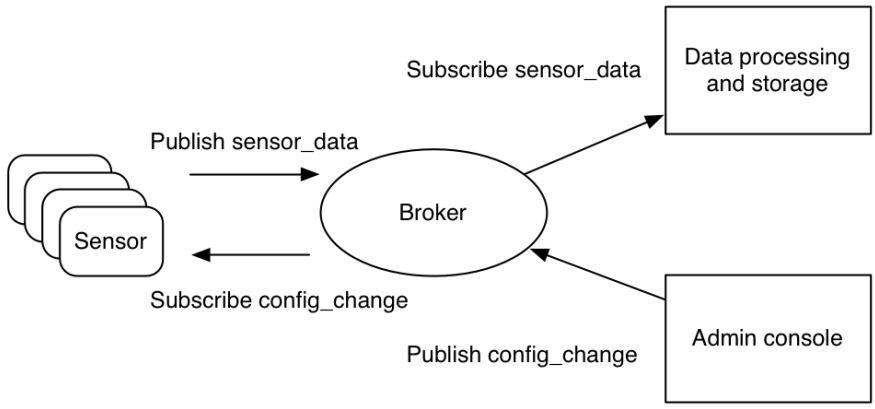
El uso de HTML dentro del prototipo nos sirvió de manera tal que pudimos enviar una petición y recibir una respuesta casi inmediata de lo que se solicita, dando órdenes al servidor de la nube para que pueda hacer un determinado proceso.

- **MQTT:** Es un protocolo estándar (Message Queue Telemetry Transport) ensamblado sobre la pila de TCP/IP es mayormente utilizado para las comunicaciones dentro del mundo de la tecnología IoT, denominado como un protocolo de código abierto para que cualquiera pueda utilizarlo.

Originariamente, “MQTT fue inventado y desarrollado por IBM a finales de los 90. Su aplicación original era conectar sensores de los oleoductos con satélites. Tal como sugiere su nombre, es un protocolo de mensajería que soporta la comunicación asíncrona entre las partes. Un protocolo de mensajería asíncrona disocia al emisor y al receptor de los mensajes tanto en espacio como en tiempo, y, por lo tanto, es escalable en entornos de red no confiables” (IBM Developer, n.d.)**.**

El funcionamiento de este protocolo es el de tomar la lectura de los sensores utilizados para poder publicarlos, de tal manera que también se habla de una suscripción a estos datos cuando el sensor los envié, por lo cual las aplicaciones de procesamiento de los datos estarán al pendiente de manera backend de esta suscripción y así poder guardarlos en la base de datos utilizada. Dentro del mismo existe una consola de administración en la cual se puede recibir comandos del sistema para así ajustar las configuraciones de los sensores, esto esta expresado en la figura 19.

Figura 19 Modelo de publicación y suscripción de MQTT para los sensores de IoT



(IBM Developer, n.d.)

MQTT consta de dos entidades básicas que participan dentro de este protocolo las cuales son:

- El **Bróker MQTT** viene a ser un tipo de servidor local o cloud que trabaja como intermediario de mensajería, el cual va a recordar el tópico al que se suscribió algún usuario y el mismo se encargara de despachar el mensaje generado por el sensor, con el tópico requerido cuando hubo la suscripción a esté.

- El **Cliente** el mismo que puede ser un sensor (dispositivo) o un usuario, que a través de una aplicación web quiere ver o enviar

datos, esto quiere decir que tanto usuarios como dispositivos pueden ser clientes para un Bróker MQTT.

ACCIONES DEL BROKER MQTT

- **CONNECT:** El Bróker como tal va a soportar la conexión de clientes entre el mismo, y el dispositivo el cual generara los tópicos al que los clientes estarán suscritos.

- **SUBSCRIBE:** Los clientes se pueden suscribir y van a ser los clientes quienes decidan el tópico que quieran recibir.

- **PUBLISH:** Esta acción está relacionada con el dispositivo que transmite los tópicos que puede publicar hacia el broker a través de la conexión que se realizó de manera previa, este mensaje va a estar relacionado a un tópico el cual tenga suscripciones de los demás usuarios.

- **UNSUSCRIBE:** Es todo lo contrario con la acción de suscribir, esta cancela la acción de que ya no lleguen más tópicos o mensajes a los cuales estaba suscrito, siendo esta aplicada sobre un tópico donde previamente haya sido suscrito.

- **DISCONNECT:** Esta acción de manera obvia, nos permite desconectarnos del Broker MQTT.

ESTRUCTURA DE UN MENSAJE MQTT

La estructura de los mensajes MQTT están divididos en:

- Un **Header** o encabezado fijo que tiene un tamaño de 2 bytes, el cual se encuentra siempre presente dentro de la mensajería, el mismo consta de un campo de control y longitud del paquete que contiene la variable.

- El **Topic** o temas de un mensaje MQTT que un cliente puede crear o suscribirse a él siendo una manera de comunicación de uno a uno o a muchos.

- Y la otra parte del mensaje el cual se trata de la carga o **Payload**, siendo estos el contenido de dicho mensaje.

- **TCP (Protocolo de Control de Transmisión) :** El protocolo TCP se representa a través de reglas a seguir en un nivel informático para establecer una conexión entre computadoras o dispositivos siendo estos similares o diferentes el mismo trabaja con una arquitectura que es formada por cinco niveles, los cuales son:

    1. **Aplicación:** Es el nivel en el cual se encuentra el protocolo HTTP conocido como “Hypertext Transfer Protocol” el cual aborda la página web con la aplicación de monitoreo de aire y sus widgets en cuestión.

    2. **Transporte:** Es el nivel en el cual se incluye a los protocolos TCP y UDP, dando paso al uso de un Bróker MQTT el cual se ocupa del manejo sincronizado y el transporte de los datos con el usuario a través del anterior nivel el cual lo hace a través de la aplicación (Ver Figura 20) .

    Figura 20 Transporte TCP dentro de un Broker MQTT

    3. **Internet:** Es el nivel que se encarga sobre la conexión, el cual identifica si está conectado o no a la red, esto lo hace mediante mensajes MQTT que comprueban la conexión con el Bróker y por ende al internet.

    4. **Físico:** El mismo está relacionado con el uso físico de dispositivos que proporcionen la información, en este caso del tema de titulación, trataría de la información proporcionada por los sensores del prototipo.

    5. **Red:** Es el nivel correspondiente a la interfaz de la red de usuarios los cuales están suscritos a ciertos tópicos que tendrán como resultado determinados datos.

En conjunto con el protocolo MQTT se creó una conexión que se va a mantener abierta dando así un beneficio de extremo a extremo en el

cual se va a poder mandar todos los mensajes que los clientes deseen, y si por algún motivo el cliente o el bróker MQTT decide cerrar la conexión, esta puede ser cerrada.

- **WebSockets:** Son básicamente el protocolo de comunicación para la conexión entre el MQTT Bróker, el aplicativo y el dispositivo. La información sobre el acceso al bróker son las siguientes:

  - Puerto TCP: **1883**
  - Puerto de Websocket: **8083**

- **JWT (JSON Web Tokens):** En cuanto a seguridad se trate, en la aplicación IoT consta de un estándar de código abierto el cual permite la transmisión de forma segura y eficiente de la información de un objeto JSON.

  La aplicación dada para JWT es la autorización y autentificación del rol dentro del aplicativo web, siendo este el escenario más utilizado y común para el JWT y funciona de tal manera que “Una vez que el usuario haya iniciado sesión, cada solicitud posterior incluirá el JWT, lo que le permitirá acceder a rutas, servicios y recursos que están permitidos con ese token” (jwt.io, n.d.).
  En la Figura 21, se puede ver el código encargado para crear un token JWT.

Figura 21 JavaScript encargado de crear el Token.

```
const jwt = require('jsonwebtoken');

let checkAuth = (req, res, next) => {

    let token = req.get('token');

    jwt.verify(token, "securePasswordHere", (err, decoded) => {

        if (err){
            return res.status(401).json({
              status: "error",
              error: err
            });
        }

        req.userData = decoded.userData;

        next();

    });

}

module.exports = {checkAuth}
```

SERVICIOS EN LA NUBE

De manera general, un servicio en la nube es considerado como un ordenador (Servidor) considerándose una de las partes principales del sistema IoT, el mismo está a cargo de recibir conexiones y datos de los dispositivos IoT que se pretenda conectar al mismo para ponerlos en funcionamiento.

Este servidor también puede estar encargado para tener una base de datos para alojar cualquier tipo de información en ella, pudiendo ser informaciones de usuarios, o simplemente grabarlos datos que serán transmitidos por los dispositivos IoT.

Otra función fundamental de este servidor es el alojamiento del aplicativo web en el cual los usuarios se conectan y ven en tiempo real o históricos de los datos transmitidos por los dispositivos IoT, los mismos que serán acumulados en la base de datos.

- **AMAZON WEB SERVICES (AWS)**

  El uso elegido de Amazon como nuestro proveedor de Servicios Web da la confianza de que siempre estará disponible, ya sea por problemas de conexión o simplemente un apagón en el sistema, dándonos la seguridad de que tendrá todo bajo control, siempre.

  AMAZON EC2

  Es un servicio de Amazon Web Services que nos permite crear un VPS o mejor conocida en español como “Servidor Virtual Privado”. El cual ofrece todo lo indispensable para poder desarrollar de manera fácil una aplicación web proporcionando de manera infalible la capacidad informática en la nube siendo está segura y de un tamaño escalable.

  SECURITY GROUPS

  Es básicamente un Firewall externo a la instancia (VPS) utilizado para la configuración de los puertos, haciendo que estos abran o cierren determinado puerto, tomando el control de la entrada de usuarios que utilicen diferentes puertos para mantener una conexión con nuestro VPS el cual contiene el aplicativo web IoT.

  El motivo de esta configuración es limitar los puertos de acceso, ya que podría existir algún conflicto entre los puertos usados con los sensores del dispositivo IoT, siendo los siguientes presentados en la tabla 11.

Tabla 11 Tipos de puertos

PUERTOS		
#	**Tipo**	**Estado**
80	HTTP	No Disponible
443	HTTPS	Disponible
22	SSH	Disponible
1883	MQTT	Disponible

COMPARATIVA DE TECNOLOGÍAS DE BASES DE DATOS NoSQL

En el mercado de las bases de datos, existe gran variedad, para el presente proyecto de titulación se enfocó para el almacenamiento en tecnologías NoSQL basadas en la nube por lo cual el análisis de estas se determinó con las siguientes opciones:

Amazon SimpleDB: Una de las primeras alternativas a tomar en cuenta fue la propia del servicio en la nube que se eligió, ya que al formar parte de Amazon Web Services sería la mejor opción en cuanto conectividad se refiere, siendo un administrador de bases de datos NoSQL de alta disponibilidad.

CouchDB: Se trata de una base de datos NoSQL escrita en JSON y distribuida mediante código abierto desarrollado por parte de Apache Software Foundation para el almacenamiento masivo de datos, muy similar a MongoDB, en el cual los dos utilizan JavaScript como lenguaje de consulta.

MongoDB: Por último, MongoDB en el cual desde un punto de vista práctico fue el sistema de base de datos más utilizado dentro de la carrera universitaria, siendo el mismo de código abierto y no relacional, lo cual lo determino como una herramienta fácil de usar bajo consultas en JavaScript y manejo de objetos como lo demandaba el sistema, almacenando la información en estructuras de datos BSON convirtiéndolo en más dinámico e integrador.

A continuación, en la tabla 12, se muestra un cuadro comparativo entre las Bases de datos NoSQL definidas con anterioridad.

Tabla 12 Cuadro comparativo de Bases de datos NoSQL.

Cuadro comparativo de Bases de Datos NoSQL			
	Amazon SimpleDB	**CouchDB**	**MongoDB**
Rendimiento	Optimizada para pocos y grandes recursos de manera pagada.	Optimizada para conjuntos pequeños de datos.	Optimizada para pocos y grandes recursos.
Disponibilidad	Alta	Alta	Alta
Almacenamiento	Escalable de paga.	Depende de la capacidad de almacenamiento del Server.	Depende de la capacidad de almacenamiento del Server.
Data Streaming	Apta	Apta	Apta
Precio	281$ / mes	Gratis	Gratis

Una vez determinadas las comparaciones entre bases de datos NoSQL para el proceso de Data Streaming, se llegó a la decisión de utilizar MongoDB como Base de Datos NoSQL para el proyecto de titulación, ya que tiene los requerimientos necesarios en cuanto optimización, disponibilidad, almacenamiento, Data Streaming y precio.

LENGUAJES DE PROGRAMACIÓN

Dentro de la programación se debe tomar en cuenta que “el software es importante porque afecta a casi todos los aspectos de nuestras vidas y ha invadido nuestro comercio, cultura y actividades cotidianas. La ingeniería de software es importante porque nos permite construir sistemas complejos en un tiempo razonable y con alta calidad” (Tobergte & Curtis, 2013) siendo estos los facilitadores de varios procesos que se toman en cuenta al momento de programar.

- **JAVASCRIPT:** Se trata de un lenguaje basado en secuencias de comandos que permite al programador crear contenido de comportamiento y actualización dinámica que la página web necesita, el cual permite un mayor flujo de trabajo, siendo este utilizado para la corrección de errores, implementar nuevos requisitos o incluso acomodar cambios inesperados dentro del sistema que se está creando.

El uso de JavaScript nos permite el manejo y almacenamiento de variables que pueden ser útiles para el sistema, el mismo también permite ejecutar código u operaciones que tienen como representación las respuestas de eventos o requerimientos que sucede dentro de una página web.

Lo más importante de JavaScript dentro del proyecto de titulación, es el uso de las denominadas interfaces de programación de aplicaciones (API), las cuales vienen a ser un conjunto de bloques de programación los cuales permiten al desarrollador implementar pequeños programas con diversas funcionalidades las cuales son de gran ayuda y proporcionan herramientas adicionales dentro del código fuente original.

- **JSON:** Se utilizó como lenguaje capturador de datos a JSON ya que el mismo, lleva cadenas de caracteres como objetos, los cuales almacenan diferentes variables con respecto a lo que se envía mediante protocolo MQTT, se podría decir que tanto JSON como MQTT van de la mano, ya que el tópico generado por este último, corresponde a un documento JSON, otra de las características de estos servidores web, los bróker y bridges MQTT es que estos están soportados por una autorización mediante JWT (JSON Web Token).

- **ARDUINO:** Es el principal lenguaje para el uso del dispositivo Arduino, el cual ejecutara todo lo referido a la adquisición de los datos de la calidad del aire y los procesara mediante protocolos para poder enviarlos al bróker MQTT y por ende al aplicativo IoT.

TECNOLOGIA DE IMPRESIÓN 3D

El uso de la tecnología 3D es muy amplio estos días ya que nos permite el crear piezas físicas a través de nuestras ideas, dándonos un sinfín de oportunidades, “También llamado manufactura por adición, consiste en producir objetos a través de la adición de material en capas que corresponden a las sucesivas secciones transversales de un modelo 3D. Los plásticos y las aleaciones de metal son los materiales más usados para impresión 3D, pero se puede utilizar casi cualquier cosa, desde hormigón hasta tejido vivo” (AutoDesk, 2020), y es por ello que se eligió la impresión 3D como medio de construcción en el cual se hizo el case que contendrá el prototipo como tal.

El software utilizado para el diseño del case fue **TinkerCad**, “Es una colección online que incluye herramientas de software de Autodesk que permite a los principiantes crear modelos 3D. Este software CAD se basa en una geometría

sólida constructiva (CSG), que permite a los usuarios crear modelos complejos mediante la combinación de objetos más simples" (Alicia M., 2020)

Después de la investigación explorativa sobre los temas pertinentes al presente plan de titulación se definió la zona de ejecución del proyecto siendo esta de manera específica en la Universidad UTE Occidental – Sede QUITO siendo su dirección Avenida Mariscal Sucre y Mariana de Jesús .

El proyecto constó de la construcción de un prototipo tecnológico que nos permita la adquisición de datos sobre la contaminación del aire dentro del sector con la utilización de un sistema Arduino, mediante la utilización de sensores IoT y siendo almacenados en una BDD de datos no relacional en la nube, los cuales generaran datos en cuanto se refiere a la calidad sobre los niveles de contaminación del aire los mismos que serán visualizados dentro de múltiples dashboard informativos en una página web responsiva para un fácil manejo por parte del usuario final.

El objetivo general de este trabajo de investigación fue desarrollar un estudio y prototipo que permita determinar las tecnologías IoT utilizables para medir, almacenar y procesar parámetros de contaminación del aire en el sector de la Universidad UTE – SEDE QUITO, y se plantearon como objetivos específicos:

- Realizar un estudio del Estado del Arte en IoT y sus aplicaciones en sistemas ambientales.
- Desarrollar un Prototipo con Raspberry Pi y/o Arduino que permita mediante sensores capturar la información de la calidad de aire.
- Analizar las tecnologías de bases de datos NoSQL en la nube capaces de almacenar la información de Data Streaming.
- Implementar una aplicación móvil/Web para presentar al usuario las condiciones actuales correspondientes a la calidad del aire.

2. METODOLOGÍA

2. METODOLOGÍA

Dentro del trabajo de titulación se utilizó la metodología Scrum, la cual tendrá como principales puntos a tomar en cuenta la construcción del prototipo, conectividad IoT y construcción de la aplicación.

2.1 METODOLOGÍA SCRUM

La etapa de desarrollo se utilizará la metodología SCRUM la cual nos ofrece un proceso de desarrollo con procesos agiles para el desarrollo e implementación. “Los principios Scrum son congruentes con el manifiesto ágil y se utilizan para guiar actividades de desarrollo dentro de un proceso de análisis que incorpora las siguientes actividades estructurales: requerimientos, análisis, diseño, evolución y entrega” (Tobergte & Curtis, 2013), los cuales ayudaron a establecer métodos de investigación junto con un proceso de desarrollo que utiliza el razonamiento para obtener conclusiones que parten de hechos aceptados como válidos para llegar a los resultados deseados.

El proyecto de titulación fue dividido en diferentes sprint con diferentes propósitos, los mismos que se enlistan a continuación:

- SPRINT 1: Fase de Análisis
- SPRINT 2: Fase de Diseño
- SPRINT 3: Fase de Desarrollo
- SPRINT 4: Fase de Pruebas
- SPRINT 5: Fase de Implementación

Cada Sprint mencionado, se ejecutó en ciclos cortos temporales y de duración fija donde cada uno tuvo una duración media de 3 semanas. Y donde por lo general cada iteración o sprint proporcionó un resultado completo ayudando en el incremento al producto final para así ser entregado en el mínimo tiempo.

2.1.1 SPRINT 1: FASE DE ANÁLISIS

Dentro del análisis para la creación del prototipado se llegó a tomar varias consideraciones con respecto al uso de sus materiales a continuación se detallará estos aspectos:

- En cuanto a la adquisición de los sensores necesitados para este proyecto fue uno de los principales puntos de preocupación, ya que los mismos no podían ser encontrados con facilidad dentro de la

Ciudad de Quito, por lo cual se procedió a la adquisición de estos por Amazon.

- Los sensores adquiridos necesitaron de un proceso de curado de 48 horas para los sensores MQ-131 y MQ-09, por otro lado, hubo un proceso de 24 horas para el sensor MQ-135, este proceso trata de dejar encendido el dispositivo dentro de su tiempo de curado respectivo, siendo este proceso el más importante y realizado por única vez para la óptima calibración de los sensores.

- Dentro del proceso de investigación se determinó el uso de plataformas IoT para la adquisición de los datos de manera automática y así crear los dashboard a partir de widgets, los cuales utilizan un código embebido para ser insertados en la página web principal del proyecto.

En la tabla 13 se describió todas las tareas realizadas en esta fase de análisis con sus respectivos tiempos de cumplimiento y el encargado del sprint.

Tabla 13 Fase de Análisis.

Sprint 1, Tiempo: 3 Semanas	**Fechas:** 01/05/2021 – 21/05/2021	
TAREA	**TIEMPO**	**ENCARGADO**
Recolección de información con respecto al tema ambiental de la calidad del aire y sus antecedentes.	7 Días	Andrés Quishpe
Recolección de requisitos fundamentales para la excelente calidad de aire otorgado mediante guías por parte de la OMS.	1 Día	Andrés Quishpe
Recolección de información y definición con respecto a los diferentes tipos de materiales que se utilizaron dentro del proyecto.	7 Días	Andrés Quishpe
Definición de las diferentes plataformas y software que se van a utilizar para el proyecto y sus fases de recolección de información.	6 Días	Andrés Quishpe

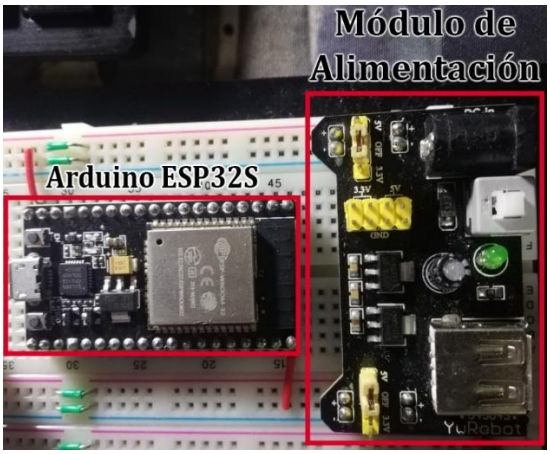
2.1.2 SPRINT 2: FASE DE DISEÑO

Al terminar el sprint 1, se empezó a concretar y realizar los respectivos diseños a tomar en cuenta que serían el prototipado junto con sus sensores, modelado de base de datos, esbozo del diseño de la página web y la creación del circuito de la placa PBC.

ELABORACIÓN DE LA PLACA DE CIRCUITO IMPRESO (PCB)

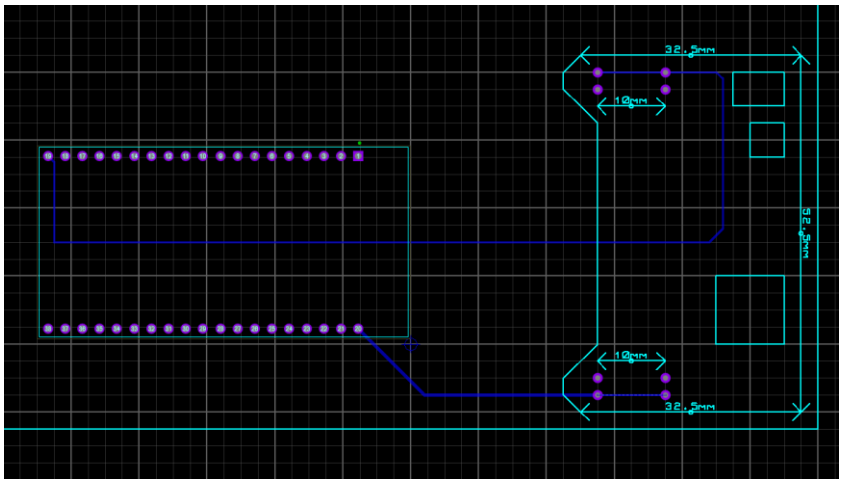
- **Diseño e impresión del PCB:** Para el diseño de la placa de circuito impreso se empezó por la pieza más importante tomando en cuenta el componente Arduino NODEMCU ESP32S el cual consta de una alimentación de 5V constantes por parte del módulo de alimentación MB102, como se puede apreciar en la siguiente imagen (Ver Figura 22).

Figura 22 Arduino ESP32S y Módulo de alimentación en Protoboard



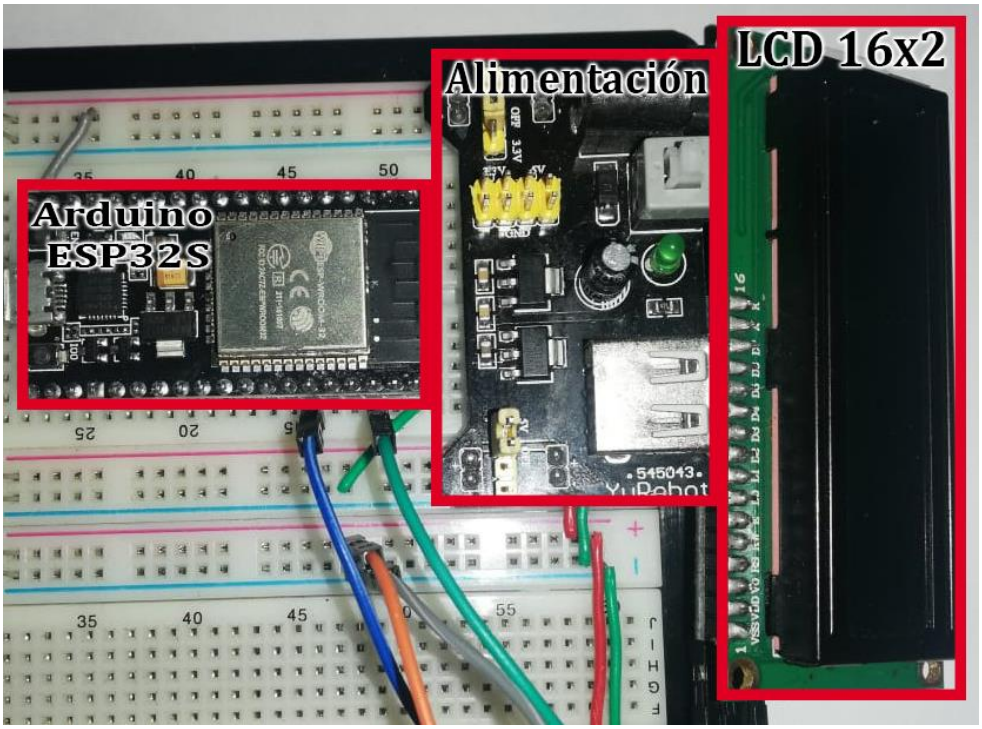
Una vez identificado los componentes que empezaran con el diseño del circuito PCB del prototipo los pasamos al programa Proteus, como se puede ver en la figura 23.

Figura 23 Diseño ESP32S y Módulo de Alimentación en Proteus



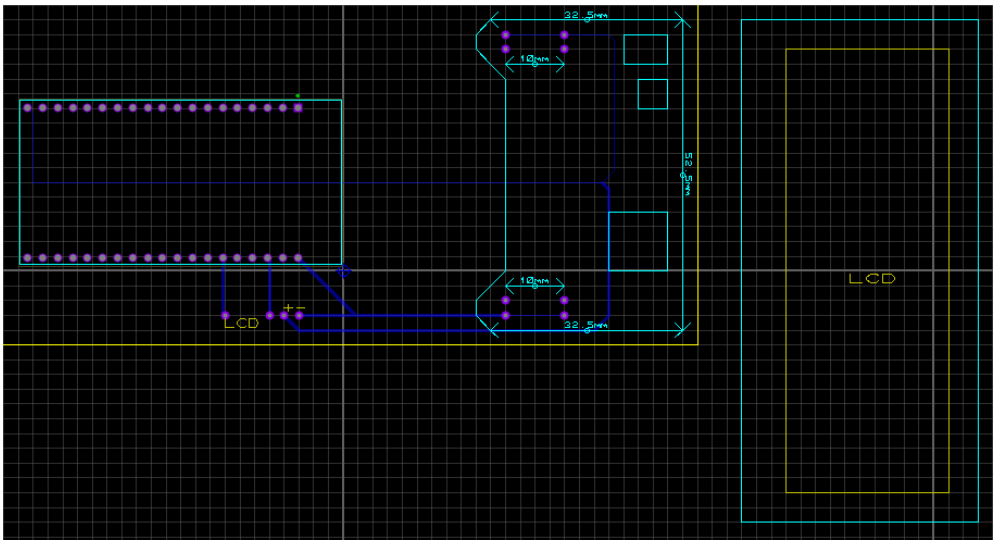
El siguiente dispositivo fue el LCD 16x2 con un adaptador I2C el cual nos da solo 2 pines que irán al Arduino ESP32S y 2 pines para la alimentación de 5V y negativo del Protoboard (Ver Figura 24).

Figura 24 Dispositivos en el Protoboard.



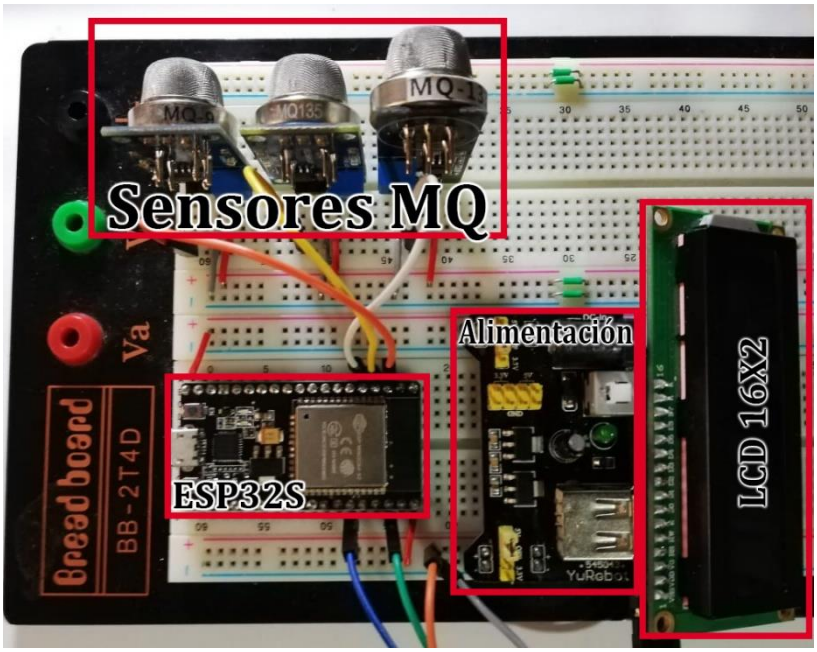
A continuación, en la figura 25 se muestran los componentes antes mencionados en el diseño de la PCB.

Figura 25 Diseño componentes en Proteus



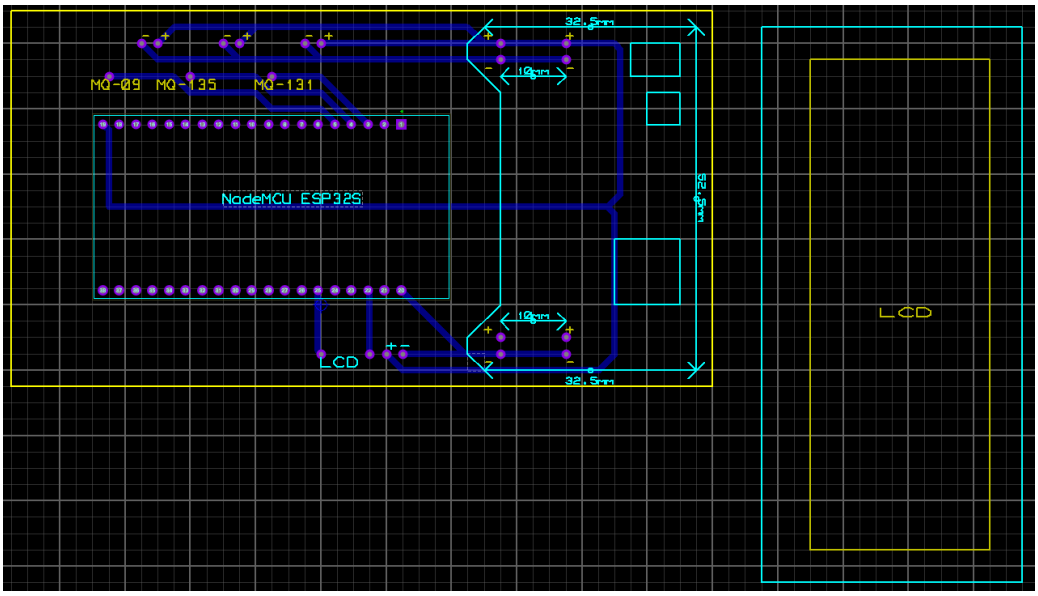
Los siguientes módulos Arduino agregados son los sensores de la serie MQ a las entradas analógicas del ESP32S como se muestra en la figura 26.

Figura 26 Varios módulos conectado al Arduino ESP32S



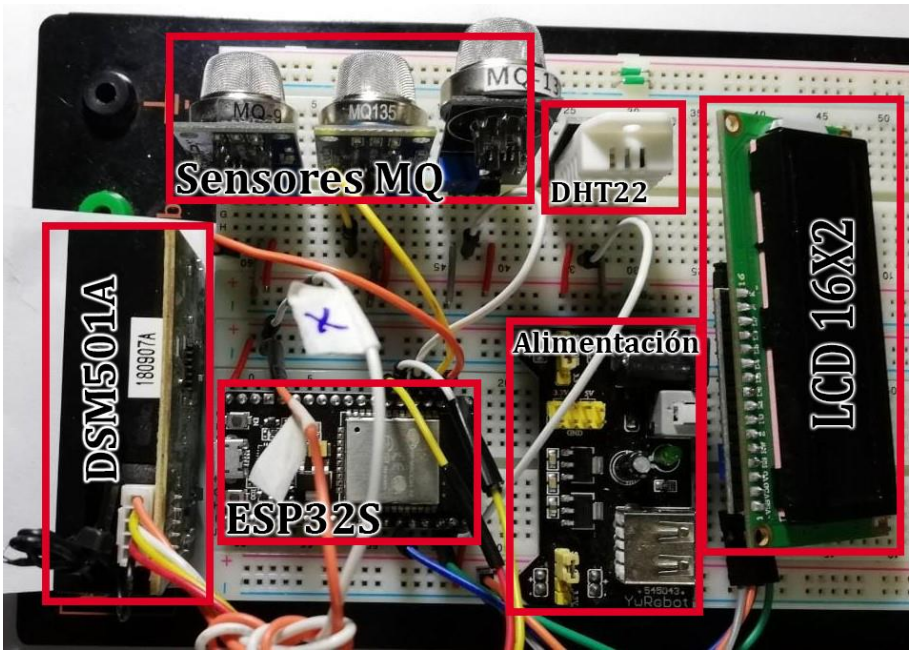
Se puede ver que ya está implementado los espacios para los sensores MQ, en este apartado se utilizó agujeros genéricos del Proteus, ya que no existen librerías diseñadas para este tipo de sensores (Ver Figura 27).

Figura 27 Diseño en Proteus del Prototipo



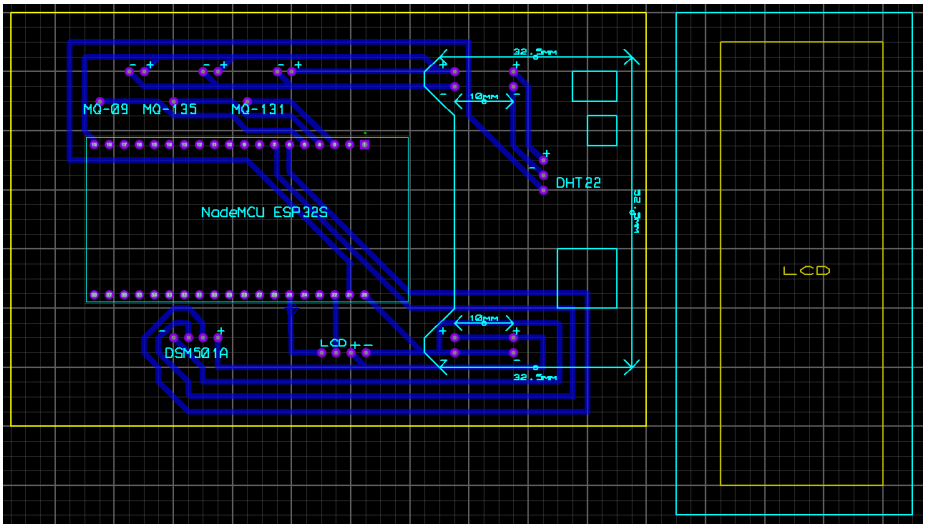
Habiendo añadido los sensores para la calidad del aire, quedan restantes Sensor de partículas DSM501A, sensor de temperatura y humedad DHT22, la conexión y distribución de los dispositivos queda de la siguiente manera dentro del protoboard, como se puede observar en la figura 28.

Figura 28 Diseño de sensores de Calidad del Aire en Protoboard



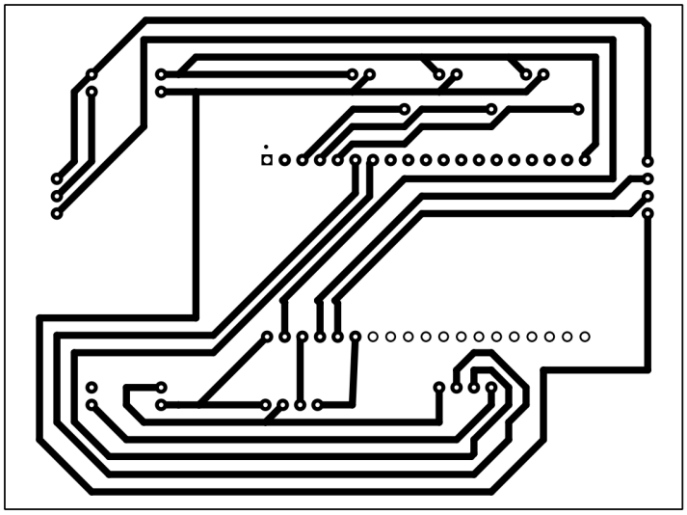
Con respecto al diseño del PCB realizado en proteus y los dispositivos agregados, se puede observar en la figura 29.

Figura 29 Diseño en Proteus del Prototipo con módulos para la calidad del aire.



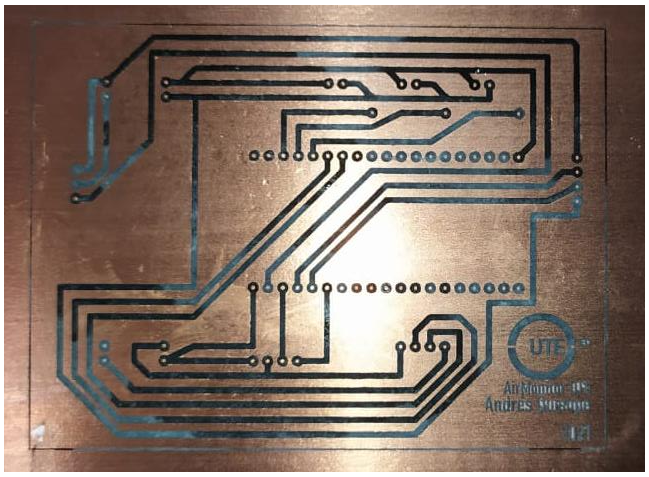
- **Construcción de la placa:** La construcción del PCB empieza con la impresión a laser (Blanco y Negro) en modo espejo del circuito que se va a realizar el PCB, siendo el de este la figura 30.

Figura 30 Diseño final para impresión PCB



Una vez impreso, se transfirió la tinta a la baquelita de cobre la cual se lo hizo por el método de planchado utilizando una plancha y ejerciendo un calor indirecto en la placa por cinco minutos, una vez cumplido con este paso, se lo enfría de manera directa en un recipiente lleno de agua, con el cual se quita el papel, frotando con las yemas de los dedos y dejando solo la tinta en la placa, como se puede observar en la figura 31.

Figura 31 Baquelita con el diseño del circuito en tinta transferida.



Como siguiente paso, se realizó la aplicación del ácido denominado Cloruro Férrico directamente en la baquelita dentro de un recipiente de plástico, dejando reposar por 30 min y realizando movimientos constantes, el ácido retiró de manera homogénea el cobre que no fue cubierto por la tinta, dejando a la vista solo las pistas del circuito, como se puede observar en las figuras 32 y 33.

Figura 32 Baquelita sumergida en Cloruro Férrico.

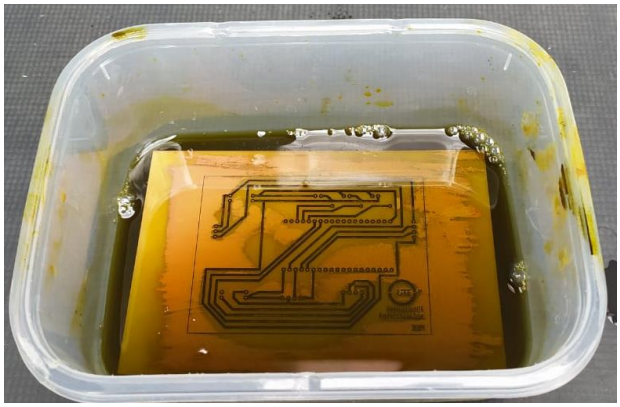
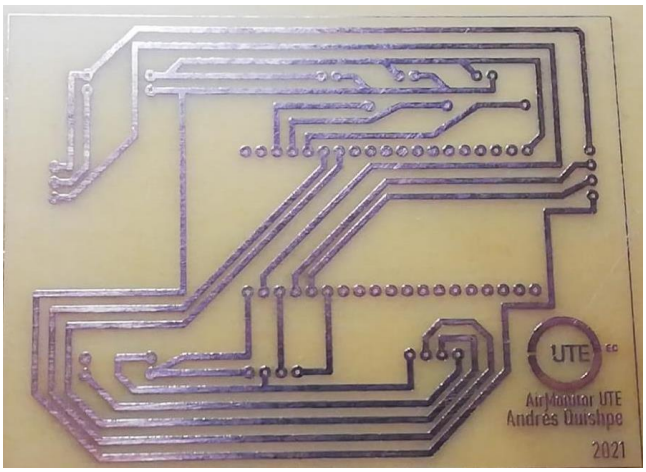


Figura 33 Baquelita sumergida en Cloruro Férrico con una superficie casi libre de cobre a excepción de las pistas cubiertas con tinta.

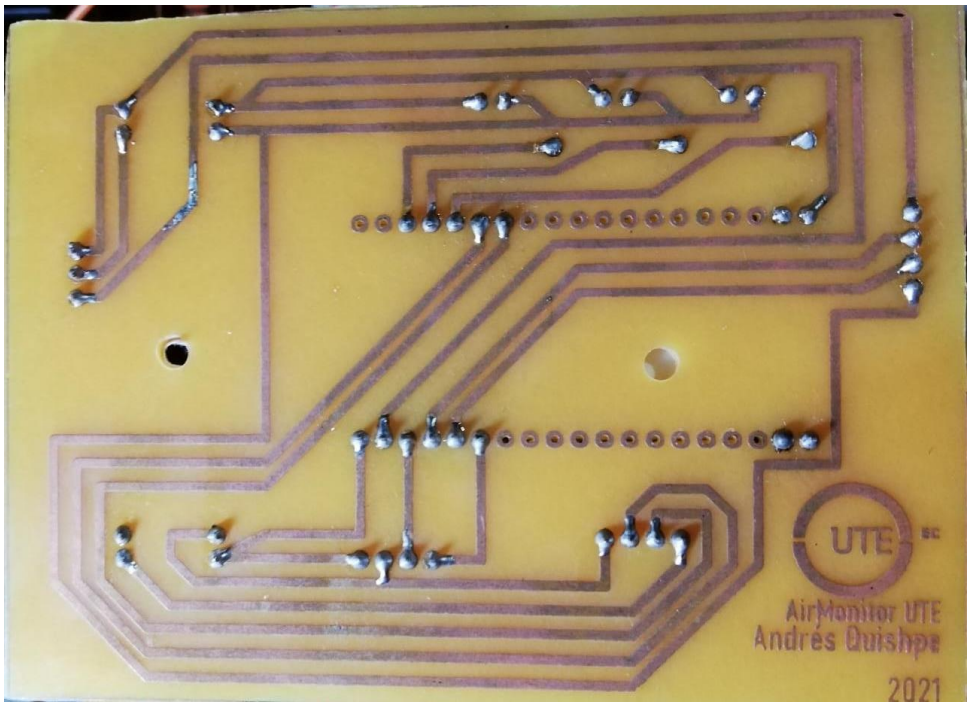
Una vez retirado todo el cobre en exceso de la baquelita, se procedió a la pulida de la placa, la cual retira la tinta que sirvió para proteger el cobre y deja a la vista las pistas de la placa (Ver Figura 34).

Figura 34 Placa PCB lista para perforar y soldar.



Como siguiente paso, se realizó la perforación de los huecos correspondientes con una broca de 1/32” y a soldar cada uno de ellos con sus componentes, dejándola lista para las pruebas de funcionamiento (Ver Figura 35).

Figura 35 Placa PCB perforada y soldada.

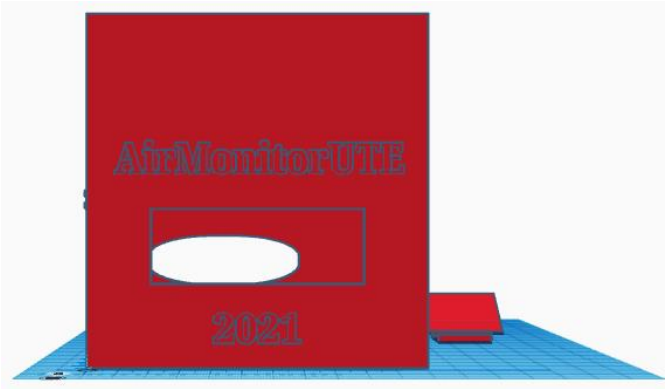


IMPRESIÓN Y MODELADO 3D

Las figuras 36, 37, 38, 39 y 40 muestran los diferentes tipos de vista que tiene el modelado 3D del case para complementar la protección y cuidado del prototipo ya terminado, realizado a partir de tomar medidas de los componentes que forman parte del prototipado.

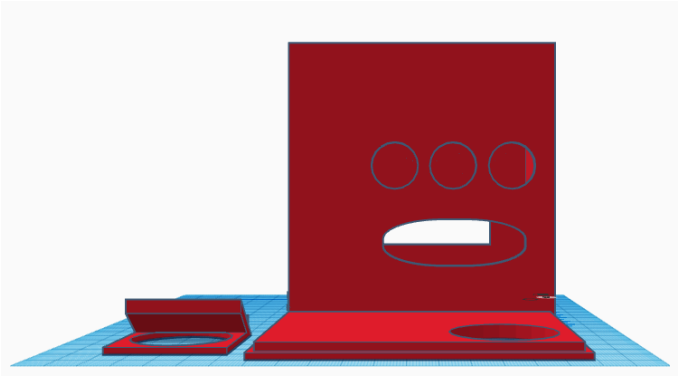
FRONTAL

Figura 36 Vista frontal modelado 3D del prototipo.



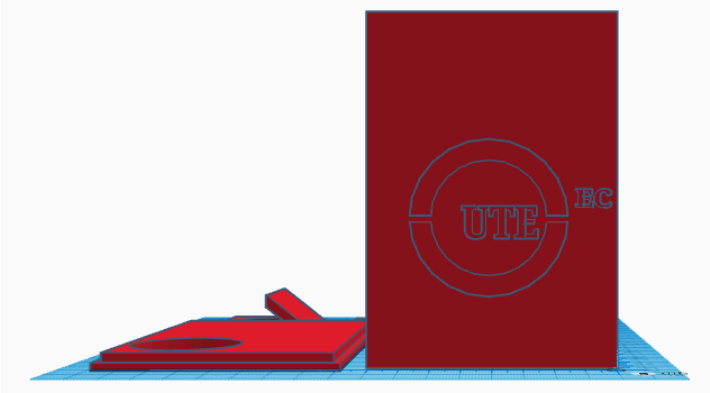
POSTERIOR

Figura 37 Vista posterior del modelado 3D



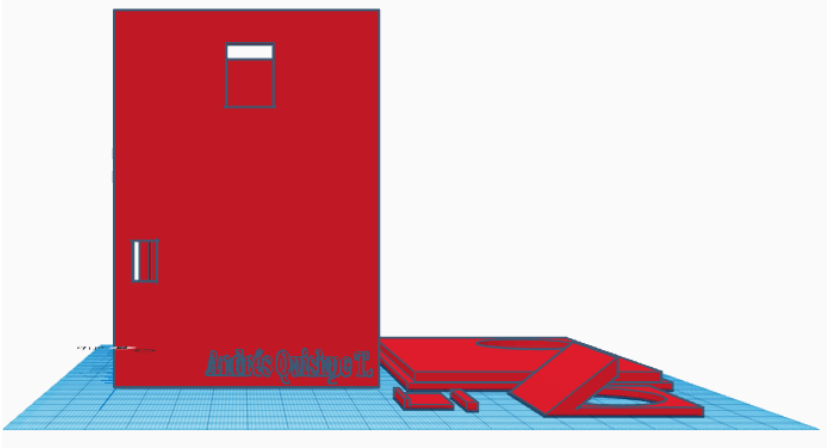
IZQUIERDA

Figura 38 Vista izquierda del modelado 3D



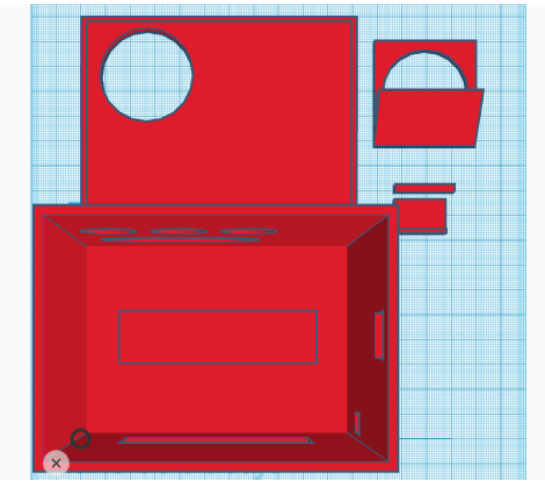
DERECHA

Figura 39 Vista derecha del modelado 3D



SUPERIOR

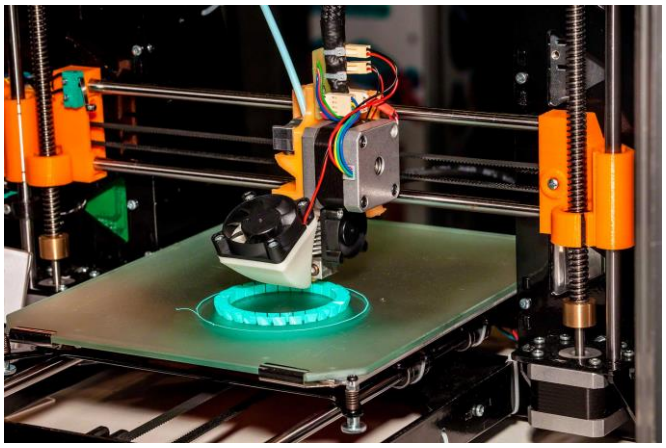
Figura 40 Vista superior del modelado 3D



- **Proceso de impresión:** Dentro del proceso de impresión se lo realizo en una empresa especializada en área de diseño y fabricación con Impresión 3D, denominada también manufactura aditiva (Ver Figura 41).

  El material utilizado para la construcción de este case fue a base de un filamento llamado PLA (ácido poliláctico) y este se denomina como uno de los filamentos más utilizados, seguros y de mejor rendimiento dentro del mercado.

  Figura 41 Imagen referencial sobre la impresión 3D



(Mecalux, 2020)

Siguiendo con la fase de diseño, en la tabla 14 se muestra las tareas correspondientes a este sprint, con sus tiempos definidos y encargado de cada tarea.

Tabla 14 Fase de Diseño.

Sprint 2, Tiempo: 4 Semanas	**Fechas:** 21/05/2021 – 18/06/2021	
TAREA	**TIEMPO**	**ENCARGADO**
Diseño del prototipado de los circuitos necesarios junto con cables y módulos Arduino (sensores) para el uso correcto de los mismos dentro de un protoboard o placa de pruebas.	7 Días	Andrés Quishpe
Implementación de una base datos NoSQL.	7 Días	Andrés Quishpe
Diseño que tendrá la página web en la cual se presentará el proyecto de titulación.	7 Días	Andrés Quishpe
Diseño del modelado 3D que será impreso con material de filamento PLA (ácido poliláctico) y donde se ensamblará el prototipo	7 Días	Andrés Quishpe

2.1.3 SPRINT 3: FASE DE DESARROLLO

El sprint 3 tiene como objetivo el desarrollo de las tareas establecidas en los Sprint anteriores de Análisis y Diseño, cumpliendo así con los parámetros establecidos dentro de cada uno junto con el objetivo del proyecto.

En la tabla 15, se muestra las tareas a realizar en este sprint, junto con sus tiempo, fechas, encargado e integrantes de la revisión de cada tarea.

Tabla 15 Fase de Desarrollo

Sprint 3, Tiempo: 4 Semanas		**Fechas:** 18/06/2021 – 16/07/2021		
TAREA	**TIEMPO**	**ENCARGADO**	**Fecha de revisión**	**Integrantes de revisión**
Creación del circuito impreso a partir del diseño del prototipado en el protoboard donde esté se tomó como referencia para realizarlo en una placa PCB.	7 Días	Andrés Quishpe	25/06/2021	Andrés Quishpe

Ensamblaje y soldado de todos los componentes que conforman el prototipo.	7 Días	Andrés Quishpe	02/07/2021	Andrés Quishpe
Implementación de programas escritos a utilizarse dentro del proyecto.	7 Días	Andrés Quishpe	09/07/2021	Andrés Quishpe
Realización de la página web final, estableciendo conectividad con la nube, base de datos y el prototipo.	7 Días	Andrés Quishpe	16/07/2021	Andrés Quishpe

2.1.4 SPRINT 4: FASE DE PRUEBAS

Con respecto al sprint 4, en la tabla 16 se muestra lo que se realizó en tanto a las pruebas funcionales de todo lo relacionado al prototipo y sistema final, en la cual se realizará una retroalimentación de la funcionalidad de todo el proyecto en su totalidad.

Tabla 16 Fase de Pruebas

Sprint 4, Tiempo: 2 Semanas	**Fechas:** 17/07/2021 – 01/08/2021	
TAREA	**TIEMPO**	**ENCARGADO**
Pruebas Funcionales del prototipo y sistema de monitoreo	15 Días	Andrés Quishpe

2.1.5 SPRINT 5: FASE DE IMPLEMENTACIÓN

Con respecto a esta etapa de la metodología, no es necesaria una implementación dentro de la organización (Universidad UTE), ya que no se cuenta con un acuerdo previo con la organización como tal. Haber pasado la fase de pruebas con satisfacción nos muestra el claro funcionamiento del sistema con respecto a lo que se pidió dentro de los objetivos del proyecto.

3. RESULTADOS Y DISCUSIÓN

3. RESULTADOS Y DISCUSIÓN

En esta fase del proyecto ya se obtiene resultados acerca de la funcionalidad del proyecto desarrollado en cuanto se refiere al aplicativo web, diseño y programación a nivel de componentes (Arduino) de este prototipo, el cual se obtuvo los siguientes resultados basados en una metodología SCRUM.

3.1 ANÁLISIS DE RESULTADOS

3.1.1 SPRINT 1: FASE DE ANÁLISIS

Dentro de esta fase nos dio como resultado los tipos de contaminantes con respecto a la calidad de aire tomando en cuenta sus niveles aceptables de exposición para considerarse óptimos, dentro de los contaminantes el que se tomó en cuenta para medir la calidad de aire es el Material Particulado en donde los niveles fueron obtenidas de las guías escritas por la OMS para un buen estado en la calidad del aire (Ver Tabla 17).

Tabla 17 Niveles de Calidad de Aire

Contaminante	Índice de Calidad de Aire (AQI)			
	Bueno	Aceptable	Dañino	Peligroso
Material Particulado (PM)	0 a 50	51 a 150	151 a 300	301 a 500

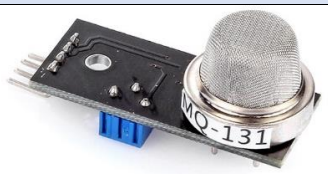
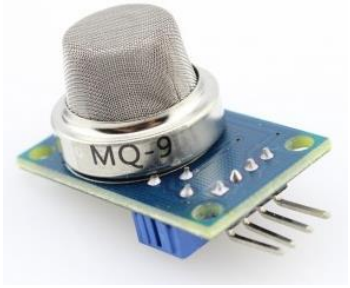
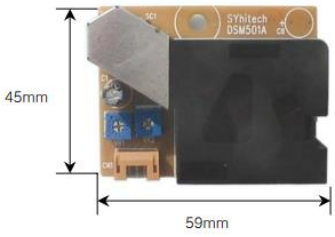
Estos contaminantes se pueden apreciar dentro de la tabla 18, los mismos que fueron pensados para la adquisición de los sensores:

Tabla 18 Contaminantes más comunes en el Aire.

Contaminantes	Unidad
PM 2.5	µg/m^3
PM 10	µg/m^3
O_3	ppm (partículas por millón)
NO_2	ppm (partículas por millón)
CO	ppm (partículas por millón)
CO_2	ppm (partículas por millón)

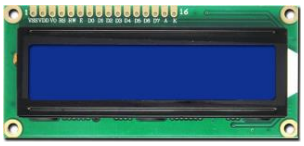
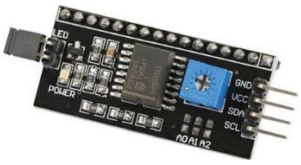
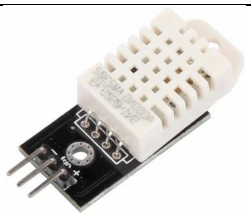
En base a la tabla anterior se realizó la adquisición de los sensores más indicados siendo estos determinados por cada contaminante y el porcentaje de concentración que ofrece cada uno, los cuales se puede apreciar en la tabla 19:

Tabla 19 Sensores con sus contaminantes.

Sensor	Contaminante que mide	Representación gráfica
MQ-131	NO_2 & O_3	
MQ-09	CO	
MQ-135	CO_2	
DSM501A	PM 10 & PM 2.5	

En la Tabla 20, se observa que el prototipo fue complementado con la adquisición de los siguientes componentes y sus funciones.

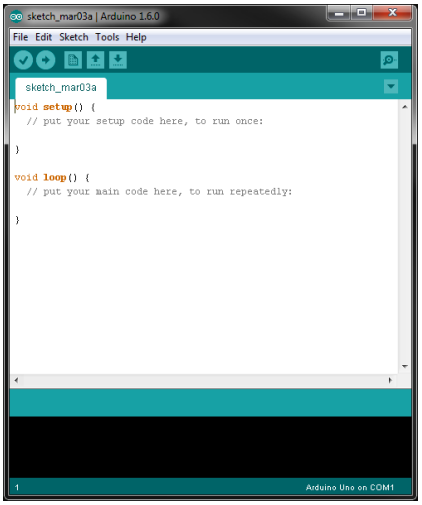
Tabla 20 Componentes complementarios.

Componente	Función	Representación Gráfica
Arduino ESP32S NodeMCU	Componente programable principal para el control del prototipo.	
LCD 16x2	Visualización de los datos e información alfanumérica de una manera más dinámica.	
Módulo Adaptador I2C	Módulo controlador I2C del LCD 16x2, utiliza una comunicación de 2 pines al Arduino (SDA y SCL) y 2 pines para la alimentación.	
Módulo de Alimentación Arduino (MB102)	Es una de las partes más importantes del prototipo ya que se encarga de la alimentación eléctrica y de manera homogénea entrega 5v a la placa y sus componentes	
Sensor DHT22	Genera como dato la temperatura y porcentaje de humedad relativa al ambiente.	

Para finalizar el Sprint 1, se llegó a la definición del software a utilizarse dentro del proyecto siendo estos los siguientes:

- **Arduino IDE:** Donde se escribió el programa de este proyecto, el cual se carga dentro de la placa de Arduino ESP32S NodeMCU (Ver Figura 42).

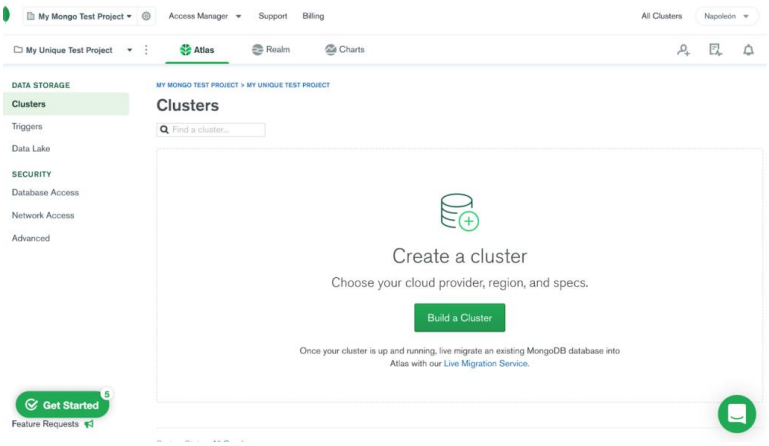
Figura 42 Interfaz de usuario de Arduino



Fuente: (Arduino, n.d.)

- **MongoDB:** Es un sistema para administrar y almacenar base de datos tipo NoSQL (no relacional) de código abierto en la cual se guarda la información en una estructura de datos denominada BSON (Ver Figura 43).

Figura 43 Interfaz de usuario MongoDB



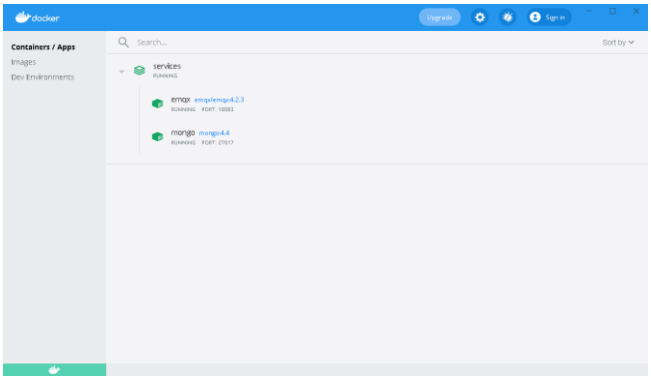
Fuente: (MongoDB, n.d.)

- **Vue.js:** Framework que fue utilizado para la construcción de las interfaces dinámicas y aplicaciones para el usuario de nuestro aplicativo web.

- **Nuxt.js:** Framework modular que está basado en Vue.js en el cual se añade librerías o paquetes necesarios para cualquier requerimiento del proyecto realizado en Vue.js.

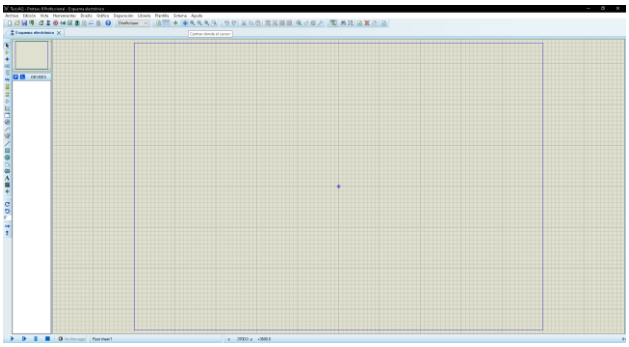
- **Docker:** Se encarga de crear contenedores con servicios específicos (EMQX, MONGO DB y Node) ligeros y portables donde se los ejecuta para el uso del proyecto (Ver Figura 44).

Figura 44 Interfaz Docker con servicios Emqx y Mongo DB



- **PROTEUS 8.11:** Es un programa que crea proyectos de construcción de la rama de la electrónica, nos permitió realizar el diseño del circuito base en el cual harán parte algunos componentes (Ver Figura 45).

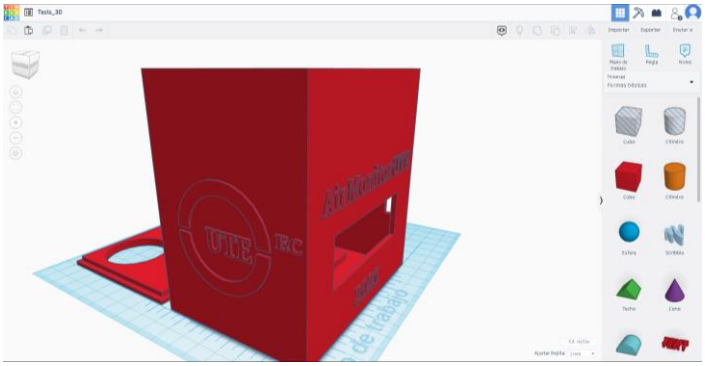
Figura 45 Interfaz de Proteus



Fuente: (Proteus, n.d.)

- **TINKERCAD:** Se utilizó para la creación del modelado de la caja para su posterior impresión en 3D (Ver Figura 46).

Figura 46 Interfaz web de TinkerCAD

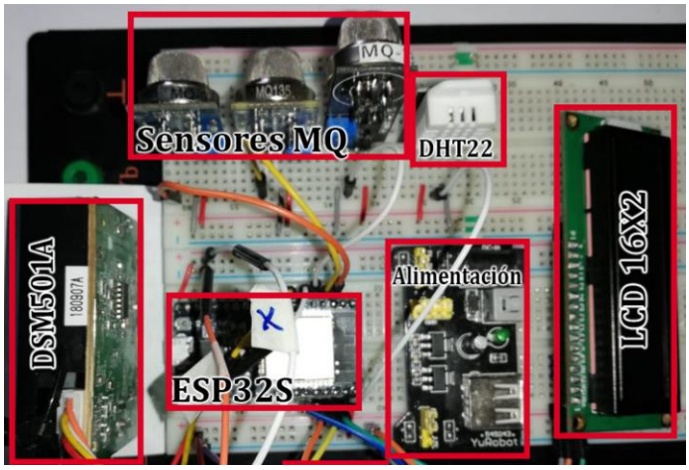


3.1.2 SPRINT 2: FASE DE DISEÑO

Dentro de esta fase se estableció todo lo estético y superficial del proyecto enlistándolos de la siguiente manera:

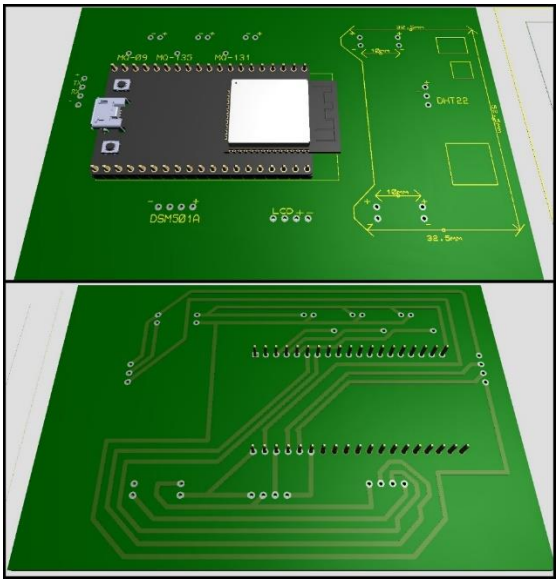
- **Diseño del circuito a partir del protoboard:** Como tal fue realizar la creación del circuito eléctrico a partir de la conexión del Arduino ESP 32 para el correcto funcionamiento del Arduino junto con todos los sensores y la distribución de estos (Ver Figura 47).

Figura 47 Diseño circuito en protoboard.



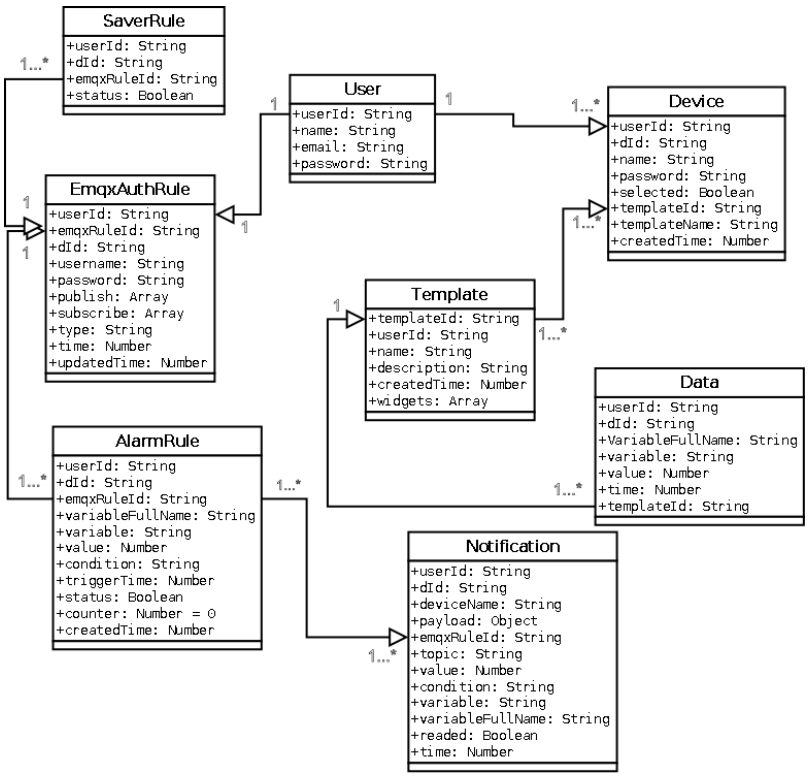
A continuación, en la figura 48 se muestra el diseño final con todos los módulos utilizados según la vista 3D del Proteus.

Figura 48 Vista 3D del Diseño de PCB.



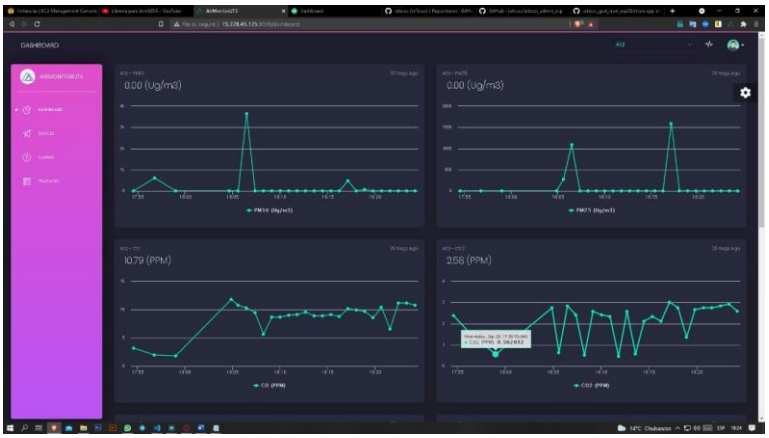
- **Implementación de una base datos NoSQL:** Se determinó que el modelado de una Base de Datos No relacional no es tan importante a la hora de llevar el armado del proyecto, es por lo que se tomó la decisión y gracias a la facilidad que tiene la implementación de una base de datos de este tipo, que el uso de las variables se vaya añadiendo según la necesidad de la cual el proyecto iba necesitando (Ver Figura 49).

Figura 49 Meta-modelo Lógico de la base de datos NoSQL.



- **Diseño de la Página Web:** Dentro de lo que es el diseño de la Página Web como tal, se tomó en cuenta la implementación de varios indicadores IoT, los mismos que se distribuyen dentro del entorno como se los puede ver en la figura 50:

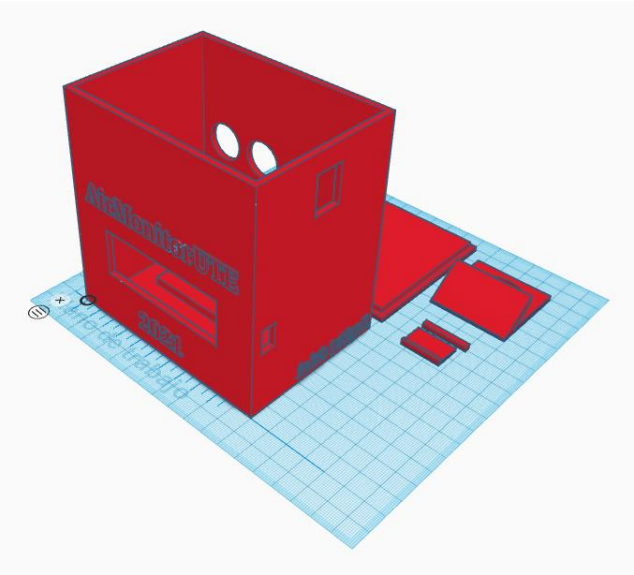
Figura 50 Diseño principal de la Página Web.



- **Modelado de la Impresión 3D:** Dentro de este apartado se realizó el diseño 3D de lo que es la carcasa de todo el prototipo (Ver Figura 51) y en el cual fue ensamblado, este fue impreso en un filamento PLA, se muestra el resultado final en la figura 52.

Modelado 3D

Figura 51 Modelado 3D en TinkerCAD



Impresión 3D

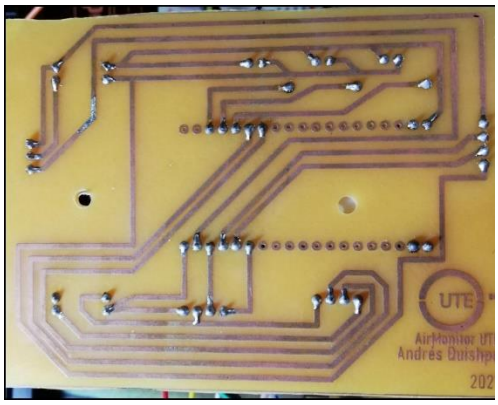
Figura 52 Impresión de PLA

3.1.3 SPRINT 3: FASE DE DESARROLLO

Dentro de esta fase se realizó todo lo relacionado con la funcionalidad del proyecto (Backend y Frontend) y la creación final del prototipo tomando en cuenta los siguientes pasos:

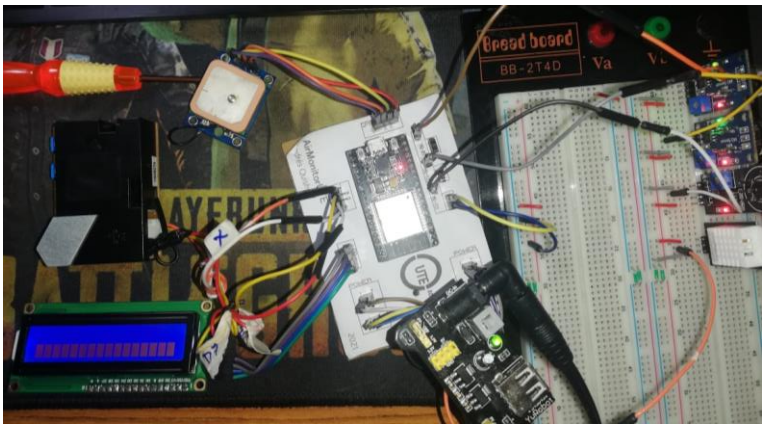
- **Creación de los circuitos PCB:** En este procedimiento nos dio como resultado, un circuito casero el cual lleva consigo entradas de tipo hembra para cables conectores tipo macho distribuidos para todos y cada uno de los componentes que corresponden al prototipo, este proceso se puede observar en la figura 53.

Figura 53 Placa PCB soldada.



Al ya tener lista la placa PCB, se procedió a probar los componentes ya conectados directamente a la misma la cual nos daría como resultado el correcto funcionamiento de los componentes con la placa PCB como se puede observar en la figura 54.

Figura 54 Placa PCB funcionando de manera correcta.



- **Ensamblaje del prototipo:** Consta de la colocación de los respectivos sensores y componentes dentro de una impresión en 3D, el cual se trata de una carcasa a medida para que encaje de manera óptima todos los componentes.

  Para el montaje se necesitó realizar más perforaciones de las ya planteadas en el diseño 3D las cuales pertenecen a la sujeción de los componentes a las paredes de la impresión, colocando también pernos y tuercas para el mejor agarre de estos, en las siguientes figuras 55, 56, 57, 58 y 59 se puede observar el prototipo ya montado y listo para funcionar.

VISTA SUPERIOR

Figura 55 Vista Superior del prototipo montado sin tapa.

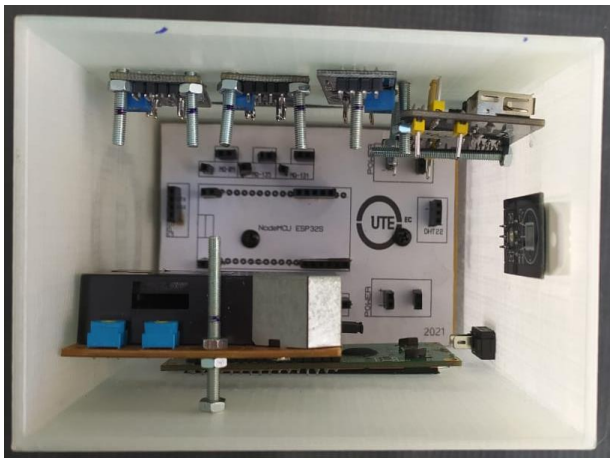


Figura 56 Vista Superior del prototipo montado con tapa.

VISTA FRONTAL

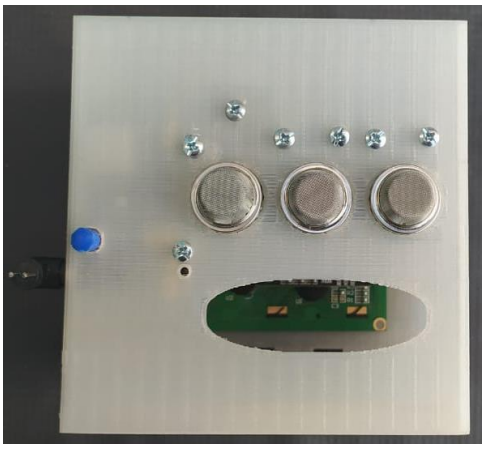
Figura 57 Vista frontal del prototipo montado.

VISTA LATERAL DERECHA

Figura 58 Vista Lateral derecha del prototipo montado.

VISTA POSTERIOR

Figura 59 Vista posterior del prototipo montado.



- **Implementación de programas escritos necesarios dentro del prototipo:** Los programas a utilizarse en este apartado son los considerados en lo que es Arduino, el mismo que se usa para la utilización de los sensores mediante funciones independientes, lo que nos da como resultado el hacer que se utilice de manera que trabajen todos los sensores a la vez, este programa se sube a la memoria del Arduino ESP-32s (ANEXO 1).

- **Página web, conectividad nube, base de datos y prototipo:** Dentro de esta sección se realizó lo que es la página web y su conectividad entre el servidor en la nube y cualquier usuario que intente acceder a la página, la misma que gracias a la nube realiza las conexiones necesarias en la base de datos alojadas en la misma nube.

  En el Anexo 2, se muestra el código necesario de la conectividad de la Pagina web a la Base de datos NoSQL.

  La conectividad del prototipo con la página web, es la más importante ya que conlleva a lo que se refiere a la distribución del contenido es decir las mediciones de cada uno de los sensores, siendo estos procesados por el backend del prototipo Arduino, teniendo como objetivo principal el adquirir correctamente las credenciales que le permitirán publicar el contenido en el bróker MQTT y por ende verse reflejado en la base de datos y página web, esta conexión se la realiza primero con la adquisición de las credenciales MQTT (Ver Anexo 3).

  La conectividad entre la página web y el bróker MQTT se lo hace mediante credenciales de superusuario y el código se lo puede visualizar en el Anexo 4.

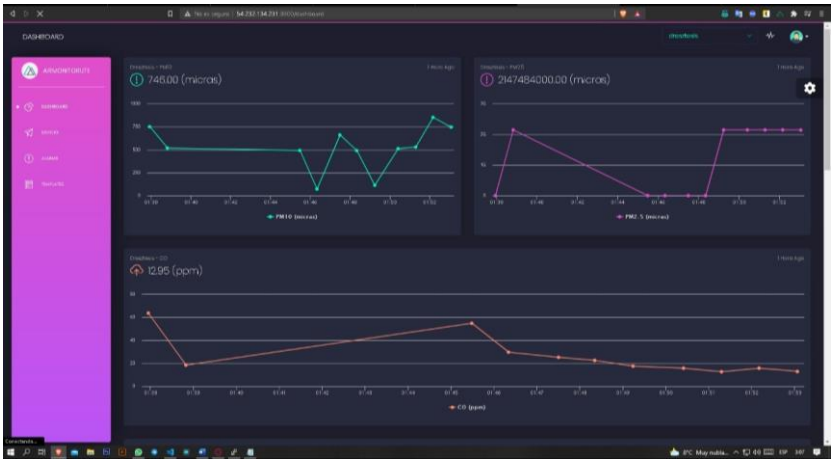
  En la parte de Backend de la página web se cuenta con variables de entorno para que se pueda lograr una conexión exitosa, estas variables de pueden ver en el Anexo 5.

3.1.4 SPRINT 4: FASE DE PRUEBAS

Dentro de esta fase se hizo las pruebas de funcionamiento del prototipo con la página web, las cuales dieron como resultado la representación gráfica de los datos enviados por los sensores al broker MQTT para poder almacenarlo en la base de datos de MongoDB en la nube, en la

figura 60 se puede observar reflejado la representación de los datos de manera gráfica:

Figura 60 Página web con datos emitidos por el prototipo.

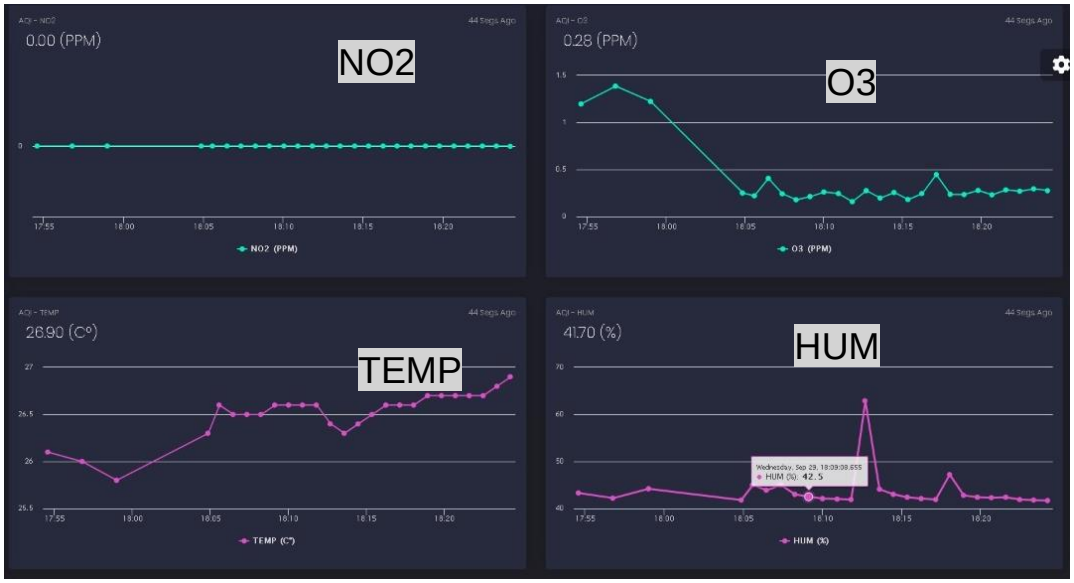


Dentro de las pruebas de funcionamiento se tomó en cuenta 2 escenarios los cuales fueron de interior y exterior, siendo sus datos los siguientes:

Interior: Como se puede observar en las Figuras 61 y 62, los datos en su mayoría son 0 o un poco mayores, siendo por excepción en la variable PM1.0 y PM2.5 en donde el último que se prendió una cerilla para que se pudiera ver una diferencia en el resultado.

Figura 61 Datos emitidos en interior.

Figura 62 Datos emitidos en el interior (parte 2).



Exterior: A diferencia de los datos emitidos en interior, los del exterior no se encuentran en 0 por mucho tiempo, por lo cual, si existe una variación en cuanto a la captura de los datos permitiéndonos entender que el sistema en conjunto funciona perfecta mente en cuanto lecturas, cabe recalcar también que el lugar donde se realizó la medición existe una baja movilización de autos (Ver Figura 63 y 64).

Figura 63 Datos emitidos en exterior.

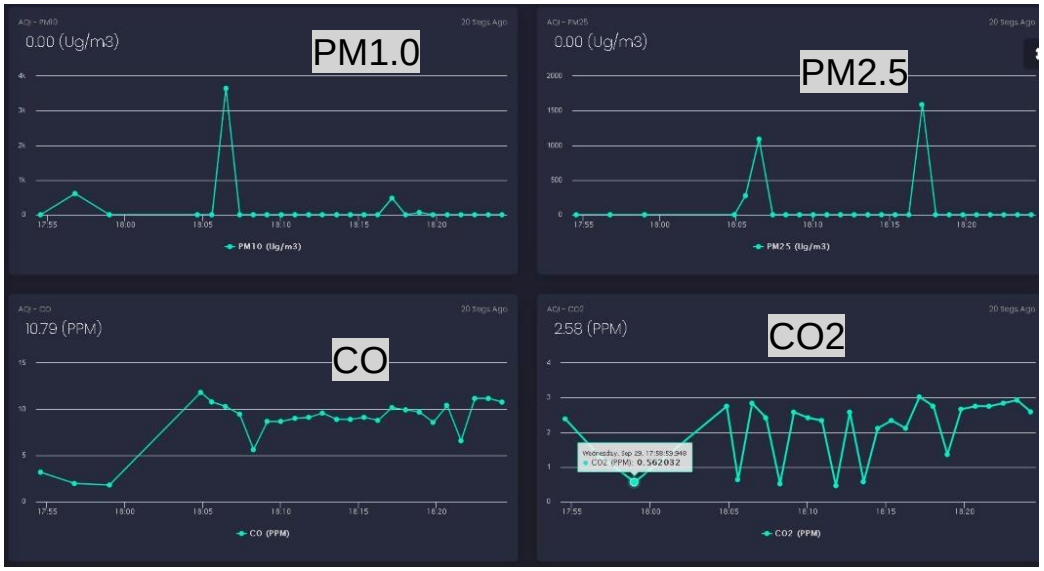


Figura 64 Datos emitidos en exterior (Parte 2).

Se puede observar en los resultados tanto en interior como exterior que el contaminante NO2 se encuentra en 0, el motivo de esto es porque se necesita una alta concentración de dicho contaminante para poder detectarlo el mismo que no se encuentra en los lugares que fue implementado.

3.1.5 SPRINT 5: FASE DE IMPLEMENTACIÓN

En la fase final de la metodología, se efectuó la implementación del sistema y prototipo de manera óptima en una ubicación exterior, dando paso a la revisión de lo que se refiere a la obtención de datos, procesamiento de estos y conectividad a todos los servicios en la nube siendo estos el broker MQTT, alojamiento página web y base de datos NoSQL (Ver Figura 65 y 66).

Figura 65 Levantamiento del servidor en la nube y los servicios requeridos.

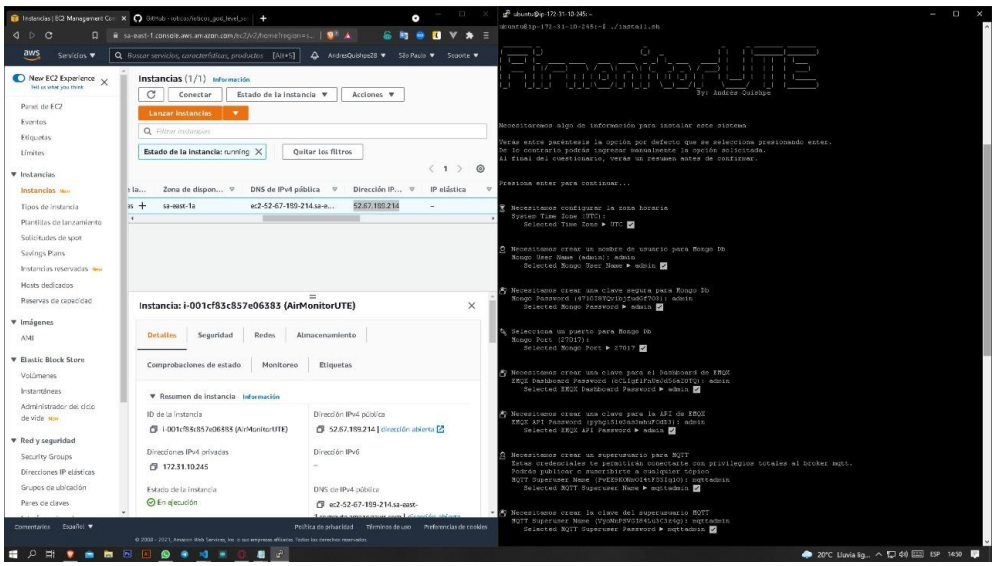
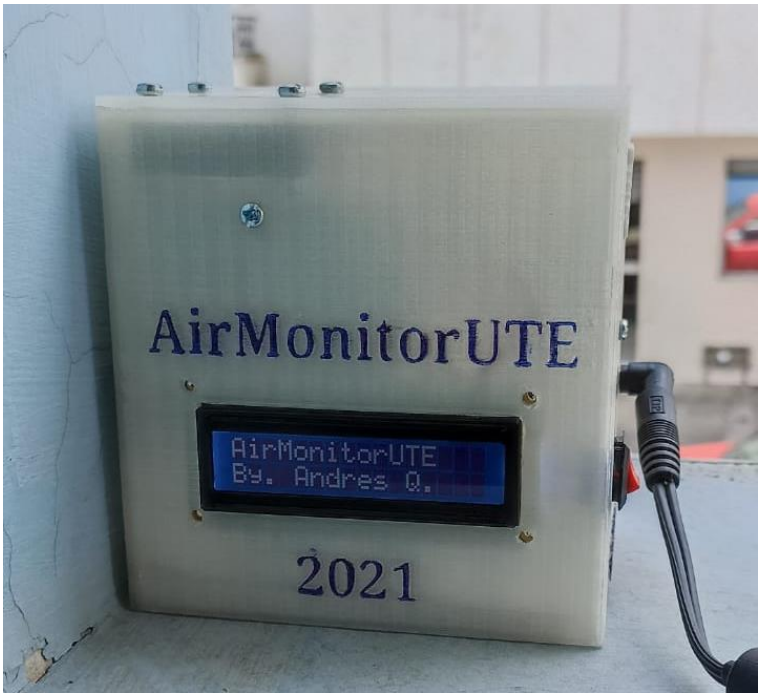


Figura 66 Prototipo probado e implementado en el exterior.



4. CONCLUSIONES Y RECOMENDACIONES

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

El prototipo fue realizado con la finalidad de contar con un dispositivo de bajo costo, que cuenta con una fácil operación. A continuación, se muestra ciertos puntos que determina las conclusiones según los objetivos planteados:

- El estudio realizado sobre la tecnología IoT y el medio ambiente, nos da como importancia la adquisición de conocimiento en los diferentes tipos de componentes especializados para el ambiente y contaminantes que existen en el ambiente, en donde para la realización del análisis, monitoreo y procesamiento de datos, una tecnología de este tipo es de gran ayuda ya que los mismos dispositivos se administran solos y con ello nos brindan información determinando los diferentes contaminantes del aire que existen dentro de nuestra atmosfera, la investigación nos ayudó a identificar qué tipo de sensores utilizar para una mejor manipulación de los datos, tomando en cuenta que existen diferentes sensores dentro del mercado siendo estos económicos o costosos, dentro de la creación del prototipo se utilizó sensores económicos y se llegó a la adquisición de datos bastante confiables y acordes a sus límites donde cada sensor puede determinar un tipo diferentes de contaminante del aire y junto con el trabajo con librerías de programación que ayudan en el procesamiento de la información de manera analógica a un tipo de información más comprensible.

- En cuanto al diseño del prototipo se desarrolló con componentes que se puede adquirir fácilmente en el mercado, llegando a tener una placa PCB principal para la distribución de todos los componentes, siendo estos los sensores IoT y estando todos conectados a un dispositivo Arduino meramente utilizado para la tecnología IoT denominado ESP-32S, todos estos componentes fueron ensamblados dentro de una carcasa diseñada en 3D para que exista una distribución más homogénea dentro del prototipo, teniendo todo listo para la adquisición de los datos y el procesado de los mismos.

- Con respecto a la base de datos, se eligió una de tipo NoSQL, la misma nos permite flexibilidad para almacenar los datos, sin tener que llevar necesariamente una estructura de estos. La

base de datos la cual almaceno las respectivas lecturas de los sensores en tiempo real (Streaming) se denomina MongoDB, ya que tiene mucha compatibilidad al usarlo con JavaScript que facilita su almacenamiento, el mismo que se lo hace gracias al protocolo MQTT, el cual es enviado por el dispositivo hacia un broker MQTT y siendo el que redirecciono a la base de datos NoSQL, en donde se captó, proceso y guardo la información.

- Se obtuvo una página web funcional basada en Vue.js que se encuentra alojada en la capa gratuita de la Nube de Amazon Web Services con su propio dominio siendo este *airmonitor-ute.live* la cual consta con Docker para crear los contenedores en el cual se levantó servicios tales como Node.js para la creación de los módulos de Vue, Emqx el cual es un broker que utiliza protocolo MQTT y Mondo DB que es usado como la base de datos NoSQL, la página web está conformada con paneles informativos que muestran las capturas de lectura de los sensores conectados al prototipo y dándonos como resultado diferentes concentraciones tomadas en cuenta para la calidad del aire.

4.2 RECOMENDACIONES

Una vez finalizada la tesis, se llegó a tener varias consideraciones a tomarse en cuenta con respecto a una investigación sobre los aspectos relacionados a la calidad de aire para futuros estudios y se propone lo siguiente:

- Desarrollar una investigación más amplia y abarcar más tipos de contaminantes que existen y afectan con lo que respecta la calidad del aire y no solo los principales contaminantes.

- Investigar sobre nuevos sensores con mayor precisión y rango de concentración a lo que respecta en la detección de contaminantes del aire.

- Promover el uso de tecnologías con protocolo MQTT no solo para la contaminación del aire sino de manera más general y con diferentes tipos de sensores ambientales y no ambientales.

- Investigar sobre nuevas tecnologías de conectividad a la nube que no limiten a la portabilidad de cualquier proyecto enfocado a la recolección de datos mediante sensores.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

Agencia para Sustancias Tóxicas y Registro de Enfermedades. (2012). Resumen de Salud Pública - Monóxido de carbono. *National Technical Information Service*, *5*, 6.

Alicia M. (2020, April 20). *TinkerCAD - 3Dnatives*. TinkerCAD: ¡Te Contamos Todo Lo Que Necesitas Saber! https://www.3dnatives.com/es/tinkercad-software-200420202/#!

Aránguez, E., Ordóñez, J. M., Serrano, J., Aragonés, N., Fernández-Patier, R., Gandarillas, A., & Galán, I. (1999). Contaminantes atmosfericos y su vigilancia. *Revista Espanola de Salud Publica*, *73*(2), 123–132. https://doi.org/10.1590/s1135-57271999000200003

Arduino. (n.d.). *Software | Arduino*. Retrieved April 15, 2021, from https://www.arduino.cc/en/software

AutoDesk. (2020). *¿Qué es la impresión 3D? | Tecnología de impresión 3D | Autodesk*. IMPRESIÓN 3D. https://latinoamerica.autodesk.com/solutions/3d-printing

Casa, M., Cusi, L., & Vilca, L. (2019). Percepciones sobre contaminación ambiental y actitudes en estudiantes universitarios. *Revista Innova Educación*, *1*(1), 140–146. https://doi.org/10.35622/j.rie.2019.01.012

CeMCAQ. (2017). *Monóxido de carbono (CO)*. Monóxido de Carbono. http://www.cemcaq.mx/contaminacion/bioxido-de-carbono-co2

Coronel, V., & Tenelanda, D. (2016). *Análisis de interoperabilidad de plataformas IoT aplicado al desarrollo de un sistema de monitoreo de polución de aire para la ESPOCH*. 1. http://dspace.espoch.edu.ec/bitstream/123456789/5440/1/98T00093.pdf%0Ahttp://dspace.espoch.edu.ec/bitstream/123456789/5440/1/98T00093.pdf%0Ahttp://dspace.espoch.edu.ec/handle/123456789/5440

Del, M., Al, A., Bancario, F., Mypes, E., Herramientas, U., Mining, D. E. D., Ing, P., Edward, J., & Polo, R. (2020). *Pontificia Universidad Católica del Perú Facultad de Ciencias e Ingeniería*. 1–6.

Godoy, D. A., Bareiro, H., Favret, F., Belloni, E., & Colloti, G. (2020). *Análisis de componente de hardware para la utilización con Frameworks de IoT*. 120–124.

Hanwei Electronics. (2018). *Technical Mq-9 Gas Sensor*. *1*, 3–6.

Hernández, M., Encalada, M., & Molina, S. (2010). Plan Nacional de Calidad del Aire. *Ministerio Del Medio Ambiente*, *1*(Reintegración Comunitaria), 5–90.

Huaman Chiroque, L. M. (2019). Universidad Nacional Tecnológica De Lima

Sur. *Universidad Nacional Tecnològica de Lima Sur*, *1*, 1–74.

IBM Developer. (n.d.). *Conozca MQTT – IBM Developer*. Retrieved May 13, 2021, from https://developer.ibm.com/es/articles/iot-mqtt-why-good-for-iot/

jwt.io. (n.d.). *Introducción a JSON Web Token - jwt.io*. Retrieved July 6, 2021, from https://jwt.io/introduction

Mecalux. (2020, May 26). *La impresión 3D y su dimensión logística - Mecalux.es*. https://www.mecalux.es/blog/impresion-3d-logistica

MongoDB. (n.d.). *Managed MongoDB Hosting | Database-as-a-Service | MongoDB*. Retrieved April 16, 2021, from https://www.mongodb.com/cloud/atlas

Mozilla.org. (n.d.). *Métodos de petición HTTP - HTTP | MDN*. Retrieved May 13, 2021, from https://developer.mozilla.org/es/docs/Web/HTTP/Methods

NatGeo. (2010). *La contaminación del aire | National Geographic*. https://www.nationalgeographic.es/medio-ambiente/la-contaminacion-del-aire

Olimex. (2012). Technical Data Mq135 Gas Sensor. *Hanwei Electronics Co.,Ltd*, *1*, 2. http://www.hwsensor.com

OMS. (2005). OMS. (2005). Guías de calidad del aire de la OMS relativas al material particulado, el ozono, el dióxido de nitrógeno y el dióxido de azufre - Actualización mundial 2005. Guías de Calidad Del Aire de La OMS Relativas Al Material Particulado, El Ozono, El . *Guías de Calidad Del Aire de La OMS Relativas Al Material Particulado, El Ozono, El Dióxido de Nitrógeno y El Dióxido de Azufre Actualización*, *5*(1), 1–21. https://apps.who.int/iris/bitstream/handle/10665/69478/WHO_SDE_PHE_OEH_06.02_spa.pdf%0Ajsessionid=970454FA25DFB60943EBC3409FF7E87B?sequence=1

Palacios Espinoza, E., & Espinoza Molina, C. (2014). Contaminación Del Aire Exterior. Cuenca - Ecuador, 2009- 2013.Posibles Efectos En La Salud. *Revista de La Facultad de Ciencias Médicas*, *32*(2), 6–17.

Pinzón, A., Castillo, M., González, E., Araúz, J., & Villarreal, V. (2018). Sistema de detección de sustancias y partículas contaminantes para el ambiente a través de sensores arduino. *Revista de Iniciación Científica*, *4*(1), 55–59. https://doi.org/10.33412/rev-ric.v4.1.1868

Proteus. (n.d.). *PCB Design and Circuit Simulator Software - Proteus*. Retrieved April 20, 2021, from https://www.labcenter.com/

Rodríguez-Guerra, A., & Cuvi, N. (2019). Air pollution and environmental justice in Quito, Ecuador. *Fronteiras*, *8*(3), 13–46. https://doi.org/10.21664/2238-8869.2019v8i3.p13-46

Sánchez Bayle, M., Martín Martín, R., & Villalobos Pinto, E. (2019). Impacto de la contaminación ambiental en los ingresos hospitalarios pediátricos:

estudio ecológico. *Pediatría Atención Primaria*, *21*(81), 21–29. https://doi.org/10.4321/s1139-76322019000100003

SEMADET. (2017). *Los contaminantes Atmosféricos | Calidad del Aire*. https://aire.jalisco.gob.mx/como-me-afecta/contaminantes-efectos

SYhitech. (2004). *DSM501A Dust sensor module*. 1–8.

Tobergte, D. R., & Curtis, S. (2013). Ingeniería de Software un enfoque práctico. In *Journal of Chemical Information and Modeling* (Vol. 53, Issue 9).

Vue.js. (n.d.). *Introducción — Vue.js*. Retrieved June 15, 2021, from https://es.vuejs.org/v2/guide/index.html

ZemsaniaGlobalGroup. (2019). *Qué es el Environmental IoT y cómo puede beneficiarnos*. https://zemsaniaglobalgroup.com/environmental-iot-mejorar-entorno/

Zhengzhou Winsen Electronics Technology Co. Ltd. (2014). *Ozone Gas Sensor （Model : 131 Low Concentration)*. 1–6. www.winsen-sensor.com

ANEXOS

Anexo 1

CÓDIGO FUENTE ARDUINO

```
#include <Arduino.h>
#include "Colors.h"
#include "IoTicosSplitter.h"
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <PubSubClient.h>
#include "LiquidCrystal_I2C.h"
#include <MQUnifiedsensor.h>
#include "DHTesp.h"
#include "DSM501.h"
/****************************************
 * CONFIG LCD
 ****************************************/
LiquidCrystal_I2C lcd(0x27, 16, 2);
/*CONFIGURACION MQTT*/
/****************************************/
String dId = "2204";
String webhook_pass = "mbxz3hSnCw";
String webhook_endpoint =
"http://15.228.159.207:3001/api/getdevicecredentials";
const char *mqtt_server = "15.228.159.207";
unsigned long timer;
/****************************************
 * Variables Auxiliares
 ****************************************/
/*Sensor DHT22*/
DHTesp dht; // Creacion de objeto DHTesp
```

```
/*Sensor PM2.5 y PM10 */
#define DSM501_PM10 32
#define DSM501_PM25 35

DSM501 dsm501(DSM501_PM10, DSM501_PM25);

/**************************************
 * Definicion de Sensores MQ
 **************************************/
#define Board ("ESP-32")
#define Voltage_Resolution (5)
#define ADC_Bit_Resolution (10)

/*************MQ131*************/
#define Pin2 (A0)
#define Type2 ("MQ-131")
#define RatioMQ131CleanAir (15)
/*************MQ135*************/
#define Pin1 (A3)
#define Type1 ("MQ-135")
#define RatioMQ135CleanAir (3.6)
/*************MQ9*************/
#define Pin0 (A6)
#define Type0 ("MQ-9")
#define RatioMQ9CleanAir (9.6)

/*DECLARACION DE SENSORES MQ*/
MQUnifiedsensor MQ9(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin0, Type0);  //MQ-9
```

```
MQUnifiedsensor MQ135(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin1, Type1); //MQ-135
MQUnifiedsensor MQ131(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin2, Type2); //MQ-131
//PINS
#define led 2

//WiFi
const char *wifi_ssid = "CLARO_QUISPHE";
const char *wifi_password = "1704342899";

//Functions definitions
bool get_mqtt_credentials();
void check_mqtt_connection();
bool reconnect();
void process_sensors();
void process_actuators();
void send_data_to_broker();
void callback(char *topic, byte *payload, unsigned int length);
void process_incoming_msg(String topic, String incoming);
void clear();

//Global Vars
WiFiClient espclient;
PubSubClient client(espclient);
IoTicosSplitter splitter;
long lastReconnectAttemp = 0;
long varsLastSend[20];

DynamicJsonDocument mqtt_data_doc(2048);
```

```
void setup()
{
  Serial.begin(921600);
  lcd.init();
  lcd.backlight();
  // Initialize DSM501
  dsm501.begin(MIN_WIN_SPAN);

  /*Regresion Lineal para Sensores MQ*/
   MQ9.setRegressionMethod(1);   //Establecer el modelo matemático para calcular la concentración de PPM
                                 //y el valor de las constantes
   MQ135.setRegressionMethod(1); //Establecer el modelo matemático para calcular la concentración de PPM
                                 //y el valor de las constantes
   MQ131.setRegressionMethod(1); //Establecer el modelo matemático para calcular la concentración de PPM
                                 //y el valor de las constantes
                                 /*Iniciar Sensores MQ*/
  MQ9.init();
  MQ135.init();
  MQ131.init();

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("AirMonitorUTE");
  lcd.setCursor(0, 1);
  lcd.print("By. Andres Q.");
  delay(2000);
```

```
  /*Calibrar Sensores*/
  Serial.print("Calibrando...");
  lcd.setCursor(0, 0);
  lcd.print("...Calibrando...");
  float calcR0MQ9 = 0;
  float calcR0MQ135 = 0;
  float calcR0MQ131 = 0;
  for (int i = 1; i <= 10; i++)
  {
    MQ9.update();
    calcR0MQ9 += MQ9.calibrate(RatioMQ9CleanAir);
    MQ135.update();
    calcR0MQ135 += MQ135.calibrate(RatioMQ135CleanAir);
    MQ131.update();
    calcR0MQ131 += MQ131.calibrate(RatioMQ131CleanAir);
    Serial.print(".");
  }
  MQ9.setR0(calcR0MQ9 / 10);
  MQ135.setR0(calcR0MQ135 / 10);
  MQ131.setR0(calcR0MQ131 / 10);
  //Error de Sensores

  if (isinf(calcR0MQ9))
  {
    Serial.println("Warning1: Conection issue founded, R0 is infite (Open circuit
detected) please check your wiring and supply");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ERROR MQ9 Inf.");
    while (1)
```

```
    ;
  }
  if (calcR0MQ9 == 0)
  {
    Serial.println("Warning1: Conection issue founded, R0 is zero (Analog pin with short circuit to ground) please check your wiring and supply");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ERROR MQ9 Zero");
    while (1)
      ;
  }
  if (isinf(calcR0MQ135))
  {
    Serial.println("Warning2: Conection issue founded, R0 is infite (Open circuit detected) please check your wiring and supply");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ERROR MQ135 Inf");
    while (1)
      ;
  }
  if (calcR0MQ135 == 0)
  {
    Serial.println("Warning2: Conection issue founded, R0 is zero (Analog pin with short circuit to ground) please check your wiring and supply");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ERROR MQ135 Zero");
    while (1)
```

```
    ;
  }
  if (isinf(calcR0MQ131))
  {
    Serial.println("Warning3: Conection issue founded, R0 is infite (Open circuit detected) please check your wiring and supply");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ERROR MQ131 Inf");
    while (1)
      ;
  }
  if (calcR0MQ131 == 0)
  {
     Serial.println("Warning3: Conection issue founded, R0 is zero (Analog pin with short circuit to ground) please check your wiring and supply");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ERROR MQ131 Zero");
    while (1)
      ;
  }


  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Calibrado!");
  delay(2000);
  Serial.println("Calibrado!.");
  lcd.clear();
  lcd.setCursor(0, 0);
```

```
  lcd.print("Pre-calentado...");
  lcd.setCursor(0, 1);
  lcd.print("  ...Espere...");
  delay(60000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Buscando senal..");
  lcd.setCursor(0, 1);
  lcd.print("......WiFi......");
  Serial.begin(921600);
  pinMode(led, OUTPUT);
    Serial.print(underlinePurple + "\n\n\nWiFi Connection in Progress" + fontReset + Purple);

  WiFi.begin(wifi_ssid, wifi_password);

  int counter = 0;

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
    counter++;

    if (counter > 10)
    {
      Serial.print("  ↴" + fontReset);
      Serial.print(Red + "\n\n         Ups WiFi Connection Failed :( ");
```

```
      Serial.println(" -> Restarting..." + fontReset);
      delay(2000);
      ESP.restart();
    }
  }

  Serial.print("  ↴" + fontReset);

  //Printing local ip
   Serial.println(boldGreen + "\n\n          WiFi Connection -> SUCCESS :)" +
fontReset);
  Serial.print("\n        Local IP -> ");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("WiFi Connectado ");
  lcd.setCursor(0, 1);
  lcd.print(WiFi.localIP());
  delay(2000);
  Serial.print(boldBlue);
  Serial.print(WiFi.localIP());
  Serial.println(fontReset);
  client.setCallback(callback);
  timer = millis();
  dht.setup(23, DHTesp::DHT22); // Conecta el sensor DHT22 al GPIO 16 (pin
D0)
}
float sensor_pm10()
{
  dsm501.update();
  float pm10 = dsm501.getParticleWeight(0);
```

```
  return pm10;
}
float sensor_pm25()
{
  dsm501.update();
  float pm25 = dsm501.getParticleWeight(1);
  return pm25;
}
float sensor_co()
{
  /*CO*/
  MQ9.update();
  MQ9.setA(599.65);
  MQ9.setB(-2.244); // Valores configurados de la ecuación para obtener la concentración de CO
  float CO = MQ9.readSensor(); // El sensor leerá la concentración de PPM usando el modelo y los valores a y b establecidos antes o en la configuración
  return CO;
}
float sensor_co2()
{
  /*CO2*/
  MQ135.update();
  MQ135.setA(110.47);
  MQ135.setB(-2.862); // Configurate the ecuation values to get CO2 concentration
  float CO2 = MQ135.readSensor(); // Sensor will read PPM concentration using the model and a and b values setted before or in the setup
  return CO2;
}
```

```
float sensor_no2()
{
  /*NO2*/
  MQ131.update();
  MQ131.setA(-462.43);
   MQ131.setB(-2.204); // Configurate the ecuation values to get CO2 concentration
  float NO2 = MQ131.readSensor();
  return NO2;
}
float sensor_o3()
{
  /*O3*/
  MQ131.update();
  MQ131.setA(23.943);
   MQ131.setB(-1.11); // Configurate the ecuation values to get CO2 concentration
  float O3 = MQ131.readSensor();
  return O3;
}
void loop()
{
  if (WiFi.status() == WL_CONNECTED)
  {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(" AirMonitorUTE");
    lcd.setCursor(0, 1);
    lcd.print(".....");
    float pm10;
```

```
float pm25;
float NO2;
float CO;
float CO2;
float O3;
delay(500);

pm10 = sensor_pm10();
pm25 = sensor_pm25();

/*PM10*/
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("PM1.0:");
lcd.setCursor(7, 0);
lcd.print(pm10);

/*PM2.5*/
lcd.setCursor(0, 1);
lcd.print("PM2.5:");
lcd.setCursor(7, 1);
lcd.print(pm25);
delay(1000);

NO2 = sensor_no2();
CO = sensor_co();

/*NO2*/
lcd.clear();
```

```
lcd.setCursor(0, 0);
lcd.print("NO2: ");
lcd.setCursor(6, 0);
lcd.print(NO2);

/*CO*/
lcd.setCursor(0, 1);
lcd.print("CO: ");
lcd.setCursor(5, 1);
lcd.print(CO);
delay(1000);
CO2 = sensor_co2();
O3 = sensor_o3();
/*CO2*/
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("CO2: ");
lcd.setCursor(6, 0);
lcd.print(CO2);
/*O3*/
lcd.setCursor(0, 1);
lcd.print("O3: ");
lcd.setCursor(5, 1);
lcd.print(O3);
delay(1000);
/*AQI*/
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("CALIDAD DE AIRE");
```

```
if (pm10 >= 0 && pm10 < 1000)
{
  lcd.setCursor(4, 1);
  lcd.print("BUENO!");
}
if (pm10 > 1001 && pm10 < 2000)
{
  lcd.setCursor(3, 1);
  lcd.print("ACEPTABLE");
}
if (pm10 > 2001 && pm10 < 3000)
{
  lcd.setCursor(4, 1);
  lcd.print("DAÑINO!");
}
if (pm10 > 3001)
{
  lcd.setCursor(2, 1);
  lcd.print("PELIGROSO!");
}
/*AQI*/
dsm501.update();
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("AQI:");
lcd.setCursor(5, 1);
lcd.print(dsm501.getAQI());
delay(1000);
```

```
    MQ9.update();
    MQ135.update();
    MQ131.update();
    check_mqtt_connection();
  }
  else
  {
    lcd.clear();
    lcd.setCursor(6, 0); // ubica cursor en columna 2 y linea 0
    lcd.print("SIN");
    lcd.setCursor(4, 1); // ubica cursor en columna 2 y linea 0
    lcd.print("CONEXION");
  }
}
int prev_temp = 0;
int prev_hum = 0;
void process_sensors()
{
  //Declarar Variables
  //int AQI = dsm501.getAQI(); //indice de calidad del aire
  float PM10 = sensor_pm10();
  float PM25 = sensor_pm25();
  float CO = sensor_co();
  float NO2 = sensor_no2();
  float CO2 = sensor_co2();
  float O3 = sensor_o3();
  float Temp = dht.getTemperature();
  float Humedad = dht.getHumidity();
```

```
if (isnan(Humedad) || isnan(Temp))
{ // verifica si alguna lectura ha fallado
  Serial.println(¡”Existe un error en la lectura del sensor DHT22!”);
  return;
}
boolean Act1, Act2, Act3, Act4;
if (PM10 >= 0 && PM10 < 1000)
{
  Act1 = true;
  Act2 = false;
  Act3 = false;
  Act4 = false;
}
if (PM10 > 1001 && PM10 < 2000)
{
  Act1 = false;
  Act2 = true;
  Act3 = false;
  Act4 = false;
}
if (PM10 > 2001 && PM10 < 3000)
{
  Act1 = false;
  Act2 = false;
  Act3 = true;
  Act4 = false;
}
if (PM10 > 3001)
{
```

```
    Act1 = false;
    Act2 = false;
    Act3 = false;
    Act4 = true;
  }
  //get variables
  /*AQI*/
  mqtt_data_doc["variables"][0]["last"]["value"] = Act1;
  mqtt_data_doc["variables"][1]["last"]["value"] = Act2;
  mqtt_data_doc["variables"][2]["last"]["value"] = Act3;
  mqtt_data_doc["variables"][3]["last"]["value"] = Act4;
  mqtt_data_doc["variables"][4]["last"]["value"] = PM10;
  mqtt_data_doc["variables"][5]["last"]["value"] = PM25;
  mqtt_data_doc["variables"][6]["last"]["value"] = CO;
  mqtt_data_doc["variables"][7]["last"]["value"] = CO2;
  mqtt_data_doc["variables"][8]["last"]["value"] = NO2;
  mqtt_data_doc["variables"][9]["last"]["value"] = O3;
  mqtt_data_doc["variables"][10]["last"]["value"] = Temp;
  mqtt_data_doc["variables"][11]["last"]["value"] = Humedad;
  //guardar variables
  mqtt_data_doc["variables"][4]["last"]["save"] = 1;
  mqtt_data_doc["variables"][5]["last"]["save"] = 1;
  mqtt_data_doc["variables"][6]["last"]["save"] = 1;
  mqtt_data_doc["variables"][7]["last"]["save"] = 1;
  mqtt_data_doc["variables"][8]["last"]["save"] = 1;
  mqtt_data_doc["variables"][9]["last"]["save"] = 1;
  mqtt_data_doc["variables"][10]["last"]["save"] = 1;
  mqtt_data_doc["variables"][11]["last"]["save"] = 1;
}
```

```
void process_actuators()
{
  if (mqtt_data_doc["variables"][2]["last"]["value"] == "true")
  {
    //digitalWrite(led, HIGH);
    mqtt_data_doc["variables"][2]["last"]["value"] = "";
    varsLastSend[4] = 0;
  }
  else if (mqtt_data_doc["variables"][3]["last"]["value"] == "false")
  {
    //digitalWrite(led, LOW);
    mqtt_data_doc["variables"][3]["last"]["value"] = "";
    varsLastSend[4] = 0;
  }
}

String last_received_msg = "";
String last_received_topic = "";

// sdfgsdfgsdfg/121212/3CTkjlSaxa/actdata
//TEMPLATE ⤵
void process_incoming_msg(String topic, String incoming)
{
  last_received_topic = topic;
  last_received_msg = incoming;
  String variable = splitter.split(topic, '/', 2);
  for (int i = 0; i < mqtt_data_doc["variables"].size(); i++)
  {
```

```
    if (mqtt_data_doc["variables"][i]["variable"] == variable)
    {
      DynamicJsonDocument doc(256);
      deserializeJson(doc, incoming);
      mqtt_data_doc["variables"][i]["last"] = doc;
    }
  }
  process_actuators();
}
void callback(char *topic, byte *payload, unsigned int length)
{
  String incoming = "";
  for (int i = 0; i < length; i++)
  {
    incoming += (char)payload[i];
  }
  incoming.trim();
  process_incoming_msg(String(topic), incoming);
}
void send_data_to_broker()
{
  long now = millis();
  for (int i = 0; i < mqtt_data_doc["variables"].size(); i++)
  {
    if (mqtt_data_doc["variables"][i]["variableType"] == "output")
    {
      continue;
    }
    int freq = mqtt_data_doc["variables"][i]["variableSendFreq"];
```

```
    if (now - varsLastSend[i] > freq * 1000)
    {
      varsLastSend[i] = millis();
      //armar el topico raiz+variable+tipo topico
      String str_root_topic = mqtt_data_doc["topic"];
      String str_variable = mqtt_data_doc["variables"][i]["variable"];
      String topic = str_root_topic + str_variable + "/sdata";
      String toSend = "";
      //transformar de string (toSend) a json
      serializeJson(mqtt_data_doc["variables"][i]["last"], toSend);
      //char array
      client.publish(topic.c_str(), toSend.c_str());
      Serial.println(topic);
      Serial.println(toSend);
    }
  }
}
bool reconnect()
{
  if (!get_mqtt_credentials())
  {
     Serial.println(boldRed + "\n\n        Error getting mqtt credentials :( \n\n
RESTARTING IN 10 SECONDS");
    Serial.println(fontReset);
    delay(10000);
    ESP.restart();
  }
  //Setting up Mqtt Server
  client.setServer(mqtt_server, 1883);
```

```
  Serial.print(underlinePurple + "\n\n\nTrying MQTT Connection" + fontReset +
Purple + "  ↴");
  String str_client_id = "device_" + dId + "_" + random(1, 9999);
  const char *username = mqtt_data_doc["username"];
  const char *password = mqtt_data_doc["password"];
  String str_topic = mqtt_data_doc["topic"];
  if (client.connect(str_client_id.c_str(), username, password))
  {
    Serial.print(boldGreen + "\n\n        Mqtt Client Connected :) " + fontReset);
    delay(2000);
    client.subscribe((str_topic + "+/actdata").c_str());
  }
  else
  {
     Serial.print(boldRed + "\n\n           Mqtt Client Connection Failed :( " +
fontReset);
  }
}
void check_mqtt_connection()
{
  if (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(Red + "\n\n        Ups WiFi Connection Failed :( ");
    Serial.println(" -> Restarting..." + fontReset);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("WiFi Lost!");
    lcd.setCursor(0, 1);
    lcd.print("Reiniciando...");
    delay(15000);
```

```
    if (!client.connected())
      ESP.restart();
  }
  if (!client.connected())
  {
    long now = millis();
    if (now - lastReconnectAttemp > 5000)
    {
      lastReconnectAttemp = millis();
      if (reconnect())
      {
        lastReconnectAttemp = 0;
      }
    }
  }
  else
  {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Conexión Ok!");
    delay(500);
    client.loop();
    process_sensors();
    process_actuators();
    send_data_to_broker();
  }
}
bool get_mqtt_credentials()
{
```

```
  Serial.print(underlinePurple + "\n\n\nGetting MQTT Credentials from WebHook" + fontReset + Purple + " ⤵");
  delay(1000);
  String toSend = "dId=" + dId + "&password=" + webhook_pass;
  HTTPClient http;
  http.begin(webhook_endpoint);
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
  int response_code = http.POST(toSend);
  if (response_code < 0)
  {
    Serial.print(boldRed + "\n\n          Error Sending Post Request :( " + fontReset);
    http.end();
    return false;
  }
  if (response_code != 200)
  {
    Serial.print(boldRed + "\n\n        Error in response :(   e-> " + fontReset + " " + response_code);
    http.end();
    return false;
  }
  if (response_code == 200)
  {
    String responseBody = http.getString();
    Serial.print(boldGreen + "\n\n        Mqtt Credentials Obtained Successfully :) " + fontReset);
    deserializeJson(mqtt_data_doc, responseBody);
    http.end();
    delay(1000);
```

```
  }
  return true;
}
void clear()
{
  Serial.write(27);    // ESC command
  Serial.print("[2J"); // clear screen command
  Serial.write(27);
  Serial.print("[H"); // cursor to home command
}
```

Anexo 2

CONEXIÓN PÁGINA WEB A BASE DE DATOS

```
//Mongo Connection
const mongoose = require("mongoose");
const mongoUserName = process.env.MONGO_USERNAME;
const mongoPassword = process.env.MONGO_PASSWORD;
const mongoHost = process.env.MONGO_HOST;
const mongoPort = process.env.MONGO_PORT;
const mongoDatabase = process.env.MONGO_DATABASE;

var uri =
  "mongodb://" +
  mongoUserName +
  ":" +
  mongoPassword +
  "@" +
  mongoHost +
  ":" +
  mongoPort +
  "/" +
  mongoDatabase;

const options = {
  useNewUrlParser: true,
  useCreateIndex: true,
  useUnifiedTopology: true,
  useNewUrlParser: true,
  authSource: "admin"
};
```

```
mongoose.connect(uri, options).then(
  () => {
    console.log("\n");
    console.log("*******************************".green);
    console.log("✔ Mongo Successfully Connected!".green);
    console.log("*******************************".green);
    console.log("\n");
    global.check_mqtt_superuser();
    console.log("url: "+uri);
  },
  err => {
    console.log("\n");
    console.log("*******************************".red);
    console.log("    Mongo Connection Failed    ".red);
    console.log("*******************************".red);
    console.log("\n");
    console.log(err);
  }
);
```

Anexo 3

CÓDIGO PARA ADQUIRIR CREDENCIALES MQTT

```
bool get_mqtt_credentials()
{
   Serial.print(underlinePurple + "\n\n\nGetting MQTT Credentials from WebHook" + fontReset + Purple + " ⤵");
  delay(1000);
  String toSend = "dId=" + dId + "&password=" + webhook_pass;
  HTTPClient http;
  http.begin(webhook_endpoint);
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
  int response_code = http.POST(toSend);
  if (response_code < 0)
  {
     Serial.print(boldRed + "\n\n          Error Sending Post Request :( " + fontReset);
    http.end();
    return false;
  }
  if (response_code != 200)
  {
    Serial.print(boldRed + "\n\n        Error in response :(   e-> " + fontReset + " " + response_code);
    http.end();
    return false;
  }
  if (response_code == 200)
  {
    String responseBody = http.getString();
```

```
    Serial.print(boldGreen + "\n\n          Mqtt Credentials Obtained Successfully
:) " + fontReset);
    deserializeJson(mqtt_data_doc, responseBody);
    http.end();
    delay(1000);
  }

  return true;
}
```

Anexo 4

CODIGO DE CONEXIÓN PÁGINA WEB Y BROKER MQTT

```
function startMqttClient() {
  const options = {
    port: 1883,    host: process.env.EMQX_API_HOST,     clientId:
      "webhook_superuser" + Math.round(Math.random() * (0 - 10000) * -1),
    username: process.env.EMQX_NODE_SUPERUSER_USER,
    password: process.env.EMQX_NODE_SUPERUSER_PASSWORD,
    keepalive: 60,
    reconnectPeriod: 5000,
    protocolId: "MQIsdp",
    protocolVersion: 3,
    clean: true,
    encoding: "utf8"
  };
  client = mqtt.connect("mqtt://" + process.env.EMQX_API_HOST, options);
  client.on("connect", function() {
    console.log("MQTT CONNECTION -> SUCCESS;".green);
    console.log("\n");
  });
  client.on("reconnect", error => {
    console.log("RECONNECTING MQTT");
    console.log(error);
  });
  client.on("error", error => {
    console.log("MQTT CONNECTION FAIL -> ");
    console.log(error);
  });
}
```

Anexo 5

VARIABLES DE ENTORNO

environment=dev

TIMEZONE

TZ=UTC

MONGO CREDENTIALS

MONGO_USERNAME=user

MONGO_PASSWORD=password

MONGO_EXT_PORT=27017

MONGO_HOST=mongo

MONGO_PORT=27017

MONGO_DATABASE=AirMonitorUTE

EMQX

EMQX_DEFAULT_USER_PASSWORD=UTE2021

EMQX_DEFAULT_APPLICATION_SECRET=emqxsecret

EMQX_NODE_SUPERUSER_USER=user

EMQX_NODE_SUPERUSER_PASSWORD=pass

EMQX_NODE_HOST=18.230.178.19

EMQX_API_TOKEN=121212

EMQX_RESOURCES_DELAY=30000

API

API_PORT=3001

FRONT

APP_PORT=3000

AXIOS_BASE_URL=http://18.230.178.19:3001/api

#MQTT

MQTT_PREFIX=ws://

MQTT_HOST=18.230.178.19

MQTT_PORT=8083