ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DISEÑO E IMPLEMENTACIÓN DE SENSORES DE MEDICIÓN DE CONTAMINACIÓN DEL AIRE BASADO EN UNA RED LORAWAN EN LA ESCUELA POLITÉCNICA NACIONAL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

JORGE FERNANDO MIÑO AYALA
jorge.mino@epn.edu.ec

DIRECTOR: SANG GUUN YOO. PhD.
sang.yoo@epn.edu.ec

CODIRECTOR: ING. BARRIGA ANDRADE JHONATTAN JAVIER, MSc.
jhonattan.barriga@epn.edu.ec

Quito, Enero 2021

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Jorge Fernando Miño Ayala, bajo mi supervisión.

Sang Guun Yoo. PhD.
DIRECTOR DEL PROYECTO

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Jorge Fernando Miño Ayala, bajo mi supervisión.

Firmado electrónicamente por:
JHONATTAN JAVIER
BARRIGA ANDRADE

ING. BARRIGA ANDRADE JHONATTAN JAVIER, MSc.
CODIRECTOR DEL PROYECTO

DECLARACIÓN DE AUTORÍA

Yo, Jorge Fernando Miño Ayala, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Jorge Fernando Miño Ayala

AGRADECIMIENTO

A mis padres, por saberme guiar cada día de mi vida, apoyarme en las decisiones que tome y estar presente en los momentos más difíciles de mi vida. Por siempre tener un consejo y un abrazo de aliento le agradeceré toda la vida por todo su amor.
A mi hermana, por quererme y estar ahí cuando lo necesite, le agradezco a Dios por tener alguien como tu hermanita querida.
A mi abuelito Jorgito, por darme siempre su amor y cariño hasta el último de su vida.
A Alexandra Orozco por cuidarme, quererme, aconsejarme siempre y empujarme cada día para ser mejor, eres parte de los pilares de mi vida.
A el MSc. Jhonattan Barriga y PhD. Sang Guun Yoo, muchas gracias por su paciencia, profesionalismo, guía y consejos en el desarrollo de mi tesis.

DEDICATORIA

Dedico con todo mi alma y corazón mi tesis a mis padres, que con su bendición, valores, trabajo duro, paciencia, protección, amor, guía, ejemplo, alegrías, motivación, consejos, apoyo incondicional y demostrándome que las segundas oportunidades que te da la vida son las que Dios tiene preparado para ti.

Dedico a mi ángel que está en el cielo mi abuelito Jorgito, que siempre me cuida y ora por mí, que con su ternura y alegrías lo recordare por siempre.

A mi hermana, que la vida me dio la oportunidad de verle crecer, compartir y ver como se convierte en un persona con un corazón único y especial.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE ANEXOS

RESUMEN

La contaminación atmosférica genera efectos negativos en la salud de la población, por lo que se creó la Red Metropolitana de Monitoreo Atmosférico de Quito (REMMAQ), encargada de medir la calidad del aire dentro de la ciudad. La trasformación a ciudades inteligentes permite la interacción entre dispositivos sin la intervención humana, teniendo sistemas que gestionen los servicios de tránsito, agua, etc. Actualmente existen redes de largo alcance y de bajo consumo de energía, que mediante sensores captan cambios en el entorno.

El presente proyecto propone un sistema de medición de contaminación del aire de bajo costo basado en una red LoRaWAN en la Escuela Politécnica Nacional, utilizando dispositivos con sensores que captan las concentraciones de monóxido de carbono (CO) y el uso de software libre para el desarrollo, a través de metodologías ágiles. Los datos son procesados en un servidor escalable en la nube, categorizando las mediciones y por medio de un aplicativo móvil permite visualizar en tiempo real la información.

Palabras Claves.- Contaminación atmosférica, LoRaWAN, Ciudades Inteligentes, monóxido de carbono

ABSTRACT

Air pollution generates negative effects on the health of the population, which is why Red Metropolitana de Monitoreo Atmosférico de Quito (REMMAQ) was created, that in charge of measuring air quality within the city. The transformation to smart cities allows the interaction between devices without human intervention, having systems that manage traffic services, water, etc. Currently there are long-range networks with low energy consumption, which through sensors capture changes in the environment.

This project proposes a low-cost air pollution measurement system based on a LoRaWAN network at Escuela Politécnica Nacional, using devices with sensors that capture the concentrations of carbon monoxide (CO) and use free software for development, through agile methodologies. The data is processed in a scalable server in the cloud, categorizing the measurements and through a mobile application, it allows that the information to be viewed in real time.

Keywords. - Air Pollution, LoRaWAN, smart cities, carbon monoxide

CAPÍTULO I

1. INTRODUCCIÓN

Los efectos que produce la contaminación del aire, sus impactos que ocasiona en la salud de las personas y cómo éstos se correlaciona en la economía de la sociedad son factores que preocupan a los gobiernos para proporcionar un ambiente de buena calidad de vida para sus habitantes. Alrededor de 3000 millones de personas, no tienen acceso a combustibles y tecnologías que generen menos contaminación. La contaminación del aire produce la muerte de millones de personas en hogares: 34% por accidentes cerebrovasculares, 26% por cardiopatías isquémicas, 22% por neumopatías obstructivas crónicas, 12% por neumonía infantil y 6% por cáncer de pulmón (Organización Mundial de la Salud , 2016).

El tener un control sobre los niveles de contaminación puede ayudar a reducir los accidentes cerebrovasculares, neuropatías, cáncer de pulmón y asma. Según la Organización Mundial de la Salud (OMS) en 2016, el 91% de la población no cumplía con las directrices planteadas sobre la calidad del aire, provocando que la contaminación atmosférica en zonas rurales y urbanas alrededor del mundo generen más de 4.2 millones de defunciones prematuras. Los países más afectados por defunciones prematuras son países de ingresos bajos a medianos, los más representativos son: Asia Sudoriental y el Pacífico Occidental.

Actualmente los países están implementando políticas para invertir en transporte que emitan menos contaminantes, utilizando medios energéticos con un impacto ambiental menor, generación de electricidad amigable por parte de las compañías y una gestión de los residuos industriales permitirá tener una reducción de desechos contaminantes de las ciudades (Organización Mundial de la Salud , 2016).

Por otro lado, el Ministerio de Ambiente ha creado el Plan Nacional de la Calidad del Aire con el fin de cumplir con la Constitución de la República del Ecuador, fundamentándose en el derecho de la población a vivir en un medio ambiente sano y ecológicamente equilibrado, que garantice un desarrollo sustentable, dicho plan se basa en tres programas (Ministerio de Ambiente, 2010) :

1.- Programa 1: Mejoramiento de la calidad del aire y prevención de su deterioro.
2.- Programa 2: Medidas a ser aplicadas durante los estados de alerta.
3.- Programa 3: Control y vigilancia de calidad del aire.

En 1999, los valores de partículas en suspensión y de monóxido de carbono registrados en Quito fueron superiores a los habituales. En el año 2000 un estudio expuso, que en el Distrito Metropolitano de Quito hubo incremento en los problemas respiratorios en escolares, debido a la contaminación vehicular; dando valores de carboxihemoglobina para el sector urbano de 5%, en el urbano periférico 2.5% y rural 0.7%.

Con estos antecedentes y problemas ambientales en la ciudad de Quito, en el año 2004 se creó la Corporación para el Mejoramiento del Aire de Quito (CORPAIRE), una iniciativa

del Municipio del Distrito Metropolitano de Quito. Las funciones de la organización es el control de la Red Metropolitana de Monitoreo Atmosférico (REMMAQ) y el estudio del Índice Quiteño de Calidad del Aire (IQCA), que consideró cinco parámetros para determinar la calidad del aire en la zona metropolitana:

1.- Material particulado fino (PM2.5).
2.- Óxido de nitrógeno (NOx).
3.- Monóxido de carbono (CO)
4.- Dióxido de azufre (SO2).
5.- Oxidantes fotoquímicos expresados como ozono (O3).

El Índice Calidad del Aire de Quito establece una escala de categorización entre 0 a 500, con 5 rangos de valores, que expresan la calidad del aire (The World Air, 2019). Para el monóxido de carbono no existe valores determinados en la Norma Ecuatoriana de la Calidad del Aire (Secretaria Ambiente, 2020). Según REMMAQ, en el año 2007, estableció que los valores promedio CO deben ser 0.5 mg/m3 en los promedios medidos en el lapso de 1 a 8 horas y que no sobrepasen los valores límites máximos de 10 mg/m^3 diarios.

Los principales problemas de contaminación en Quito son:

- Automotores a Diesel que no cumplen la norma de opacidad.
- Tecnología inapropiada de motores para la altura de Quito.
- Tráfico.
- Baja calidad de combustibles.

El Ministerio de Ambiente en el 2010 para mejorar la calidad del aire en Quito, planteó las siguientes soluciones y hacer frente a estas problemáticas (Ministerio de Ambiente, 2010):

- Implementación de trasporte masivo como: eco vía, trolebús y articulados.
- Revisión vehicular por la CORPAIRE.
- Restauración de áreas verdes.

La Red Metropolitana de Monitoreo Atmosférico con estaciones en: Cotocollao, Belisario, Centro, El Camal, Guamaní, Los Chillos, Tumbaco y San Antonio de Pichincha, estas proporcionan información cada dos horas. Además Quito cuenta con otro sistema que analiza la concentración de gases (CO. NO2, O3, SO2) en el Laboratorio Químico de la Secretaría del Ambiente cada 30 días (Secretaría de Ambiente, 2016).

En la actualidad varios países como Japón, Estados Unidos y México utilizan sistemas de medición de la calidad del aire de calidad con el fin de recolectar y analizar toda la información a través modelos matemáticos y técnicas de muestreo, esto es complementado con datos socioeconómicos, geográficos, meteorológicos y culturales de cada localidad (Instituto Nacional de Ecología, 2020). Un ejemplo de proyecto de monitoreo de la calidad del aire es el desarrollado en la ciudad de La Plata, implementado por a la cantidad de habitantes e industrias, como empresas petroquímicas, destilerías y la Central Térmica de la región (Represa, y otros, 2018).

La necesidad de tener dispositivos de bajo costo para infraestructuras de redes de largo alcance, que se adapten a las prestaciones de hardware en el desarrollo de redes inalámbricas de sensores es inminente. Es importante masificar el uso de sensores para medir y visualizar la contaminación en el aire en tiempo real en la Escuela Politécnica Nacional.

1.1. Objetivos

1.1.1. Objetivo general

Diseñar e implementar sensores para medir la contaminación del aire sobre una red de tipo LoRaWAN en la Escuela Politécnica Nacional.

1.1.2. Objetivos específicos

- Configurar e integrar los sensores en la arquitectura LoRaWAN.
- Desarrollar un backend que permita recoger la información enviada por los sensores.
- Desarrollar un aplicativo móvil que permita visualizar y monitorear los niveles de gas captados por los sensores desplegados.
- Evaluar la calidad del aire en la Escuela Politécnica Nacional.
- Validar el funcionamiento de los sensores de medición de gas.

1.2 Marco Teórico

En el presente capitulo, se enmarcará en discutir los problemas de salud provocados por la calidad del aire. Además, buscar una alternativa a sistemas de medición de la calidad del aire con tecnologías inalámbricas de bajo consumo energético y largo alcance, y como se aplica el Internet de las Cosas (IoT - Internet of the Things) para mejorar la calidad de vida de las personas.

1.2.1 Calidad del Aire

Evaluar los efectos que provoca en la salud en las personas los contaminantes comunes del aire, teniendo en la mayoría de las personas efectos nocivos en la salud, por lo que la OMS ha publicado directrices específicas sobre los diferentes efectos de la contaminación ambiental en la población de las ciudades (Organización Mundial de la Salud , 2016).

1.2.1.1 Calidad del Aire en Ecuador

Ecuador posee el Plan Nacional de Calidad del Aire, que genera acciones para gestionar la calidad del aire que están en un marco de políticas y estrategias de calidad del aire que el Ministerio de Ambiente ha creado para sustentabilidad ambiental del desarrollo del país. Este plan permite una regulación, control, seguimiento de todas los componentes que están involucrados en la gestión ambiental de la calidad del aire. Sin embargo, Ecuador tiene pocas investigaciones sobre los efectos de la contaminación del aire, por lo que estos tópicos no están en el programa de desarrollo urbano. No existen estudios epidemiológicos relacionados con la contaminación del aire y existe un estudio de la Fundación Natura en el proyecto Calidad del Aire (Ministerio de Ambiente, 2010), determinando que el impacto económico de contaminación atmosférica en la salud en los años 1991 – 2000 ascendió a 34 millones de dólares, estos valores son de ingresos hospitalarios, costos de tratamientos y estimaciones de años perdidos por problemas de salud.

La termoeléctrica y refinería de Esmeraldas son una de las mayores fuentes de producción de combustible y derivados a nivel Ecuador, pero son una fuente de contaminación ambiental, produciendo consecuencias nocivas para la salud de la ciudad. Las fumigaciones son otros casos que afectan a la población sobre todo en zonas fronterizas con Colombia para reducir el crecimiento de plantaciones de coca, teniendo consecuencias mortales en la flora fauna y poblaciones que están asentadas en estas zonas (Ministerio de Ambiente, 2010).

El Ministerio de Salud Pública muestra los casos registrados de las infecciones respiratorias registradas en el Ecuador (Figura 1.1).

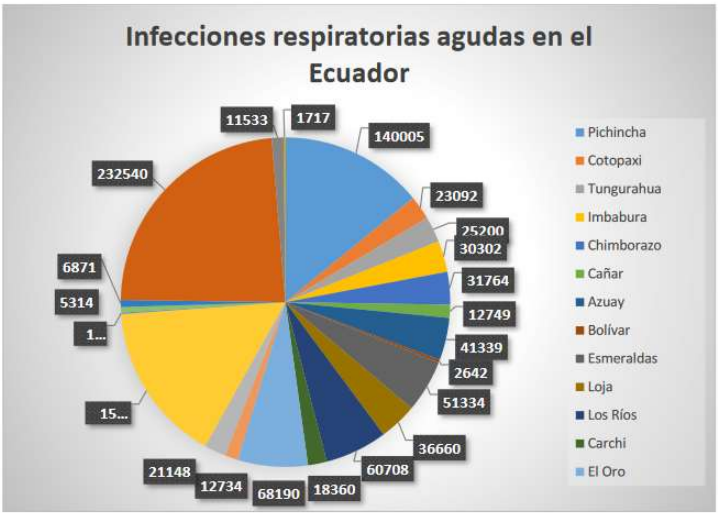


Figura 1.1.- Infecciones respiratorias agudas en el Ecuador. Imagen tomada (Ministerio de Ambiente, 2010)

Para mitigar los efectos de la contaminación ambiental el Ministerio de Ambiente planteo los siguiente en la (Tabla 1.1):

Tabla 1.1.- Programas y proyectos del Plan Nacional de Calidad del Aire del Ecuador. Tabla basada (Ministerio de Ambiente, 2010)

PROGRAMAS	PROYECTOS
PROGRAMA 1: CONTROL Y VIGILANCIA DE LA CALIDAD DEL AIRE.	Proyecto 1. Desarrollo del inventario nacional de emisiones Proyecto 2. Sistema Nacional De Monitoreo Y Vigilancia De La Calidad De Aire Proyecto 3. Elaboración de un SISTEMA DE INFORMACIÓN DE LA CALIDAD DEL AIRE (SICA). Proyecto 4. Implantación del Sistema Nacional De Vigilancia Epidemiologica De Las Enfermedades Respiratorias Generadas Por Contaminacion Del Aire Proyecto 5. Modelos Predictivos De Calidad Del Aire
PROGRAMA 2: MEJORAMIENTO DE LA CALIDAD DEL AIRE Y PREVENCIÓN DE SU DETERIORO	Proyecto 6. Competencias sobre la gestión del recurso aire Proyecto 7. Formación, capacitación e investigación en gestión de la calidad del aire integrados a las políticas nacionales de ciencia y tecnología Proyecto 8. Programa Nacional De Reducción De Emisiones. Proyecto 9. Producción de combustibles de mejor calidad para fuentes móviles y fijas.

	Proyecto 10. Sistema Nacional De Revisión Técnica Vehicular. Proyecto 11. Sistema De Participación Ciudadana En Tema Calidad Del Aire
PROGRAMA 3: MEDIDAS A SER APLICADAS DURANTE LOS ESTADOS DE ALERTA	Proyecto 12. Implementación De Planes De Contingencia Ante Episodios Críticos De Contaminación Del Aire

1.2.1.2 Índice de la Calidad del Aire

Son valores que establecen una categorización de los valores permisibles de concentraciones de componentes en el aire en un intervalo de tiempo determinado, con el propósito de proteger la salud y ambiente.

1.2.1.3 Norma Ecuatoriana de la Calidad del Aire

Se ha establecido una Norma Ecuatoriana de la Calidad del Aire (NECA) con el objetivo de preservar la salud de las personas, el bienestar del ecosistema, la calidad del aire, determinando métodos y procedimientos para medir las concentraciones de contaminantes en el aire.

La norma plantea la siguiente clasificación: contaminantes del aire ambiente, normas generales para concentraciones de contaminantes comunes en el aire ambiente, planes de alerta, alarma y emergencia de la calidad del aire, métodos de medición de concentración de contaminantes comunes del aire ambiente peligros inducidos por otros contaminantes del aire (Secretaría de Ambiente, 2016).

Los principales contaminantes del aire son:

- Partículas Sedimentarias
- Material Particulado con un diámetro de 10 micrones, abreviación PM10
- Material Particulado menos a 2.5 micrones, abreviación PM2.5
- Óxido de Nitrógeno: NO y NO2.
- Dióxido de Azufre SO2.
- Monóxido de Carbono.
- Oxidantes Fotoquímicos.

La Secretaria de Ambiente determinó rangos de componentes del aire para determinar la calidad de aire y preservar la salud humana (Tabla 1.2).

Tabla 1.2.- Rangos calidad del Aire (Secretaría de Ambiente, 2016)

Nivel	Bueno (ug/m^3)	Alerta (ug/m^3)	Alarma (ug/m^3)	Emergencia (ug/m^3)
CO_2	0-14999	15000	30000	40000
CO	0-4.5	4.5-9.5	9.5-15.5	>15.5
O_3	0-199	200	400	600
NO_2	0-999	1000	2000	3000
SO_2	0-199	200	1000	1800
PM_{10}	0-249	250	400	500
$PM_{2.5}$	0-149	150	250	350

Y estableciendo la siguiente información según la tabla (Tabla 1.2) para obtener el Índice de Calidad del Aire (Tabla 1.3):

Tabla 1.3.- Estándar Índice calidad del Aire US-EPA 2016 tomada de (The World Air, 2019)

CA	**Color de referencia**	**Calidad del Aire**	**Advertencia**	**Efectos**
0-50		Bueno	La calidad del aire se considera satisfactoria y la contaminación del aire representa poco o ningún riesgo.	Ninguna
51-100		Moderado	La calidad del aire es aceptable; sin embargo, para algunos contaminantes puede haber un problema de salud moderado para un número muy pequeño de personas que son inusualmente sensibles a la contaminación del aire.	Los niños y adultos activos y las personas con enfermedades respiratorias, como el asma, deben limitar la exposición prolongada al aire libre.
101-150		No es saludable	Los miembros de grupos sensibles pueden	Los niños y adultos activos, y las

		para grupos que son sensibles.	experimentar efectos en la salud. El público en general no es probable que se vea afectado.	personas con enfermedades respiratorias, como el asma, deben limitar el esfuerzo prolongado al aire libre.
151-200		Insalubre	Todos pueden comenzar a experimentar efectos en la salud; los miembros de grupos sensibles pueden experimentar efectos de salud más graves.	Los niños y adultos activos, y las personas con enfermedades respiratorias, como el asma, deben evitar el esfuerzo prolongado al aire libre; todos los demás, especialmente los niños, deben limitar el esfuerzo prolongado al aire libre
201-300		Muy poco saludable	Advertencias sanitarias de condiciones de emergencia. Existe la posibilidad que la población entera se vea afectada.	Los niños y adultos activos, y las personas con enfermedades respiratorias, como el asma, deben evitar todo esfuerzo al aire libre; todos los demás, especialmente los

				niños, deben limitar el esfuerzo al aire libre.
300+		Peligroso	Alerta de salud: todos pueden experimentar efectos de salud más graves.	Todos deberían evitar todo esfuerzo al aire libre.

1.2.1.4 Red Metropolitana de Monitoreo Atmosférico de Quito (REMMAQ)

La Red Metropolitana de Monitoreo Atmosférico de Quito (REMMAQ) provee datos captados y analizados basados en políticas y normativas con el objetivo de aplicar acciones para el mejoramiento del aire y mostrando la información comprensible para el público (Secretaria Ambiente, 2020).

La infraestructura mostrada indica los equipos que posee la REMMAQ, la ubicación y los métodos que utiliza para medir los diferentes contaminantes (Tabla 1.4).

Tabla 1.1.- Equipos de medición REMMAQ (Secretaria Ambiente, 2020)

Contaminante	Número Equipos	Ubicación	Método de medida o principio de operación	Marca y modelo
Material particulado PM10	4	Tumbaco, Guamaní, Carapungo, Sap	Atenuación de rayos beta	Thermo Scientific/FH62C14
Material particulado PM2.5	6	Belisario, Camal, Centro, Cotocollao, Carapungo, Sap	Atenuación de rayos beta técnica semejante para PM10 EPA No. EQPM-1102-150	Thermo Andersen / FH62C14
Dióxido de azufre (SO2)	7	Belisario, Camal, Centro, Tumbaco, Cotocollao, Carapungo, Chimbacalle	Fluorescencia por pulsos de luz ultravioleta técnica semejante EPA No. EQSA-0486-060	THERMO 43C / 43i
Dióxido de azufre (SO2)	2	Laboratorio Estándares, E. móvil	Fluorescencia ultravioleta técnica semejante EPA No. EQSA-0495-0100	TELEDYNE API / T100
Ozono (O3)	10	Bel, Cam, Cen, Tum, Chi, Cot, Car, Gua,	Absorción de luz técnica semejante EPA No. EQOA-0880-04	THERMO 49C / 49i

		Jip*,Lab. Electrónico		
Ozono (O3)	2	Laboratorio Estándares, E. móvil	Absorción de luz ultravioleta técnica semejante EPA No. EQOA-0992-08	TELEDYNE API / T400
Óxidos de nitrógeno (NOX)	8	Bel, Cam, Cen,Cot, Car, Gua, Jip, Chi	Quimioluminiscencia técnica semejante a EPA No. RFNA-1289-074	THERMO 42C / 42i
Óxidos de nitrógeno (NOX)	2	Laboratorio Estándares, E. móvil	Quimioluminiscencia técnica semejante EPA No. RFNA-1194-099	TELEDYNE API / T200
Monóxido de carbono (CO)	8	Bel, Cam, Cen, Cot, Car, Gua, Jip, Lab. Electrónico	Absorción infrarroja no dispersiva técnica semejante EPA No. RFCA-0981-054	THERMO / 48C / 48i
Monóxido de carbono (CO)	2	Laboratorio Estándares, E. móvil	Absorción infrarroja no dispersiva técnica semejante EPA No. RFCA-1093-093	TELEDYNE API/ T300
Multicalibrador (SO2, NOX, CO, O3)	12	Bel, Cam, Cen, Tum, Chi, Cot, Car, Gua, Jip, Lab. Electrónico, E. móvil	Principio de operación: Dilución de gases, aire cero con un material de referencia certificado	THERMO/ 146C / 146i
Multicalibrador (SO2, NOX, CO, O3)	1	Laboratorio Estándares	Principio de operación: Dilución de gases, aire cero con un material de referencia certificado	TELEDYNE API/ 700E
Generador Aire Cero	12	Bel, Jip, Cam, Cen, Tum, Chi, Cot, Car, Gua, Lab. electrónico, E. móvil	Principio de operación: Filtración de aire comprimido por medio de carbón activado y purafill, y calentamiento para oxidación.	THERMO / 111
Generador Aire Cero	1	Lab. Estándares J ipijapa	Principio de operación: Filtración de aire comprimido por medio de carbón activado y purafill, y calentamiento para oxidación.	ECOTECH / HTO-1000HC
Estación portátil para monitoreo de CO, SO2, NO2, 03,	1	Lab. Electrónico	Principio de operación: Gas Sensitive Semiconductor (GSS) Gas Sensitive Electrochemical (GSE)	AQM60

PM2.5, humedad relativa y temperatura del aire				

1.2.1.5 Centro de datos

El centro de datos Red Metropolitana de Monitoreo Atmosférico de Quito (REMMAQ) tiene la capacidad de recibir, almacenar y procesar la información que monitorizan las estaciones en los diferentes puntos de Quito (Secretaria Ambiente, 2020) las conexiones de red son mediante el estándar ethernet.

Tabla 1.2.- Centro de datos REMMAQ (Secretaria Ambiente, 2020)

Sistema Blade, con 5 servidores físicos	1	Centro de Datos	Virtualizado de los servidores de: comunicaciones, bases de datos, web, índice quiteño calidad del aire, proxy, respaldos, correo, envío de alarmas.	>HP C-3000
Sistema Almacenamiento, con capacidad de 9.6 TB.	1	Centro de Datos	Almacenamiento de información de Analizadores de Gases, Meteorología, bases de datos y sistemas que utiliza la REMMAQ y la Secretaría de Ambiente.	HP P2000
Librería Cintas para 24 cintas de (1.5 TB) con tecnología LTO-4 y LTO-5	1	Centro de Datos	Respaldos de información y sistemas	HP M5L2 024

El IoT permite que se pueda implementar soluciones de monitoreo de aire rentables y completas midiendo cambios de la calidad del aire, utilizando cambios en la calidad del aire, a través de sensores implementados en ubicaciones donde podría haber un mayor impacto de contaminación como residencias o en zonas vulnerables como hospitales o escuelas, para tomar medidas y mejorar la calidad de vida de personas sensibles a la contaminación (IoTforall, 2020).

1.2.2 IoT en el mundo

La necesidad por implementar IoT para dar soluciones adaptables a diferentes problemas cotidianos está masificando la implementación de dispositivos en varios campos al rededor del mundo, siendo los más relevante es con un 23% los proyectos implementados para Smart City que es la aplicación de tecnología inteligente, con el IoT y los sistemas de información geográfica (Universidad Internacional de Valencia, 2019), teniendo un mayor impacto en Europa (IoT-Analytics, 2018), el segundo campo es la Industria permitiendo a los sensores y dispositivos captar datos que provienen de las máquinas en una industria y

obtener información de cómo se mejoraría la productividad, reducir costes en caso de detectar fallos tempranos en maquinarias con todos estos factores obtener mayor margen de ganancia (Deloitte, 2020), en el siguiente gráficos se detalla todos los segmentos de IoT que se están implementando a nivel mundial ,detallados a continuación (Figura 1.2):

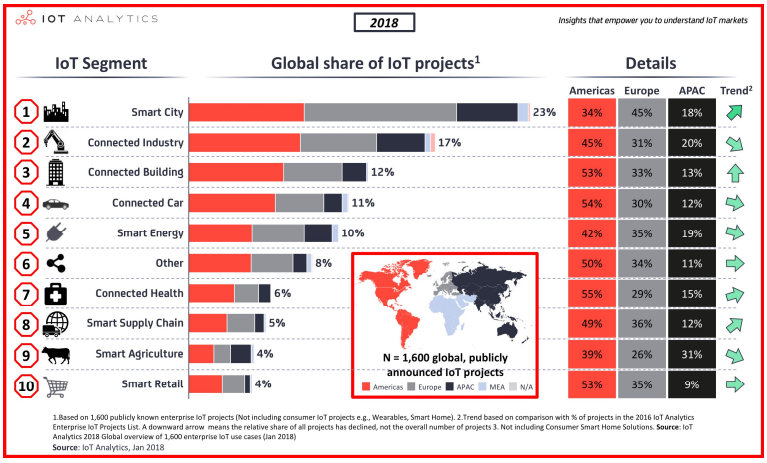


Figura 1.1**.- Proyectos IoT distribución por segmento (IoT-Analytics, 2018)**
Incremento y proyección de dispositivos IoT hasta el año 2025 Figura 1.3:

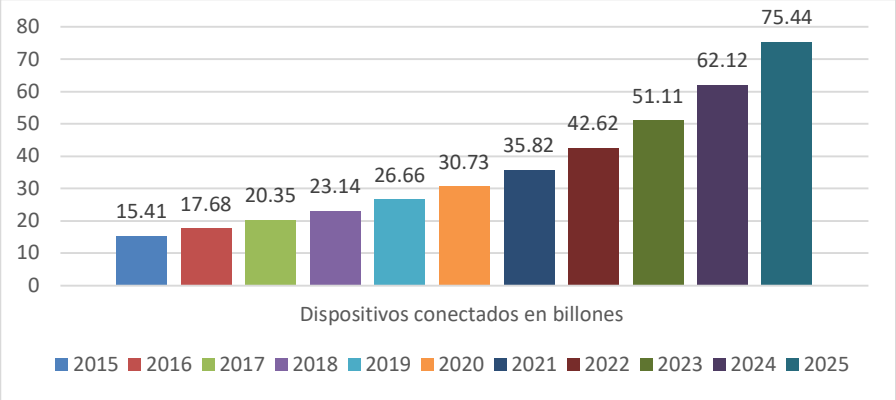


Figura 1.2.- Tendencia implementación dispositivos IoT (en billones). Imagen Tomada (IoT World Online, 2017)

1.2.3 Internet de las Cosas

El término Internet de las Cosas o Internet of the Things (IoT) fue utilizado por Kevin Ashton en 1999 para dar a entender la obtención de información computacional sin intervención humana y autónoma, obteniendo información del entorno físico e interactuando con el mismo.

Existen varias definiciones de IoT entre ellas contamos:

- "IoT es considerado como la conexión en red de objetos físicos o dispositivos en una red abierta y llena de objetos inteligentes que tiene la capacidad de auto

gestionarse, compartir información, datos y recursos, reaccionar y actuar frente a situaciones y cambios en el medio ambiente" (Hegde, 2010).

- Se define el Internet de las Cosas como la conexión de objetos o cosas a Internet permitiendo obtener información en función con su interacción con su entorno (Ordóñez Monfort, 2017).

Hay diversos sistemas implementados en ambientes urbanos que utilizan Wireless Sensor Networks (WSN), Vehicular Sensor Networks (VSN) u otras tecnologías basadas en IoT (Filip, y otros, 2020). Una red de sensores (WSN) comprende al conjunto de sensores que se comunican inalámbricamente con un nodo central y tienen la capacidad de captar información de su entorno como la temperatura, humedad, aire, etc. La estructura de este tipo de un nodo con un sensor, el Gateway o nodo central, la estación base es un ordenador o conjunto de ordenadores que almacenan los datos para poder ser procesador o visualizados y la red inalámbrica la encargada de comunicar los diferentes dispositivos que utilizan banda ISM es una banda no licenciada, utilizando una diversidad de topologías como son estrella, malla e híbrida (Capella Hernández, 2011).

1.2.4 Tecnologías de amplia cobertura y bajo consumo energético

IoT está en pleno desarrollo debió al aumento de dispositivos conectados por año según Gartner (IDC, 2018), esto permite el desarrollo de nuevas tecnologías inalámbricas con la capacidad de interconectar una gran cantidad de dispositivos, debido a que las redes inalámbricas no se ajustaban a los requerimientos de IoT como ZigBee o bluetooth, o de largo alcance como 3G, 4G y 5G que presentan un mayor consumo de energía y coste de infraestructura (ALFAIOT, 2019).

LPWAN, siglas de Low Power Wide Area Network, se refiere a conexiones de largo alcance, con la posibilidad de conectar un número elevado de dispositivos en un área de cobertura amplia con un bajo consumo de energía, los dispositivos son sensores o dispositivos conectados que tienen un volumen bajo de datos.

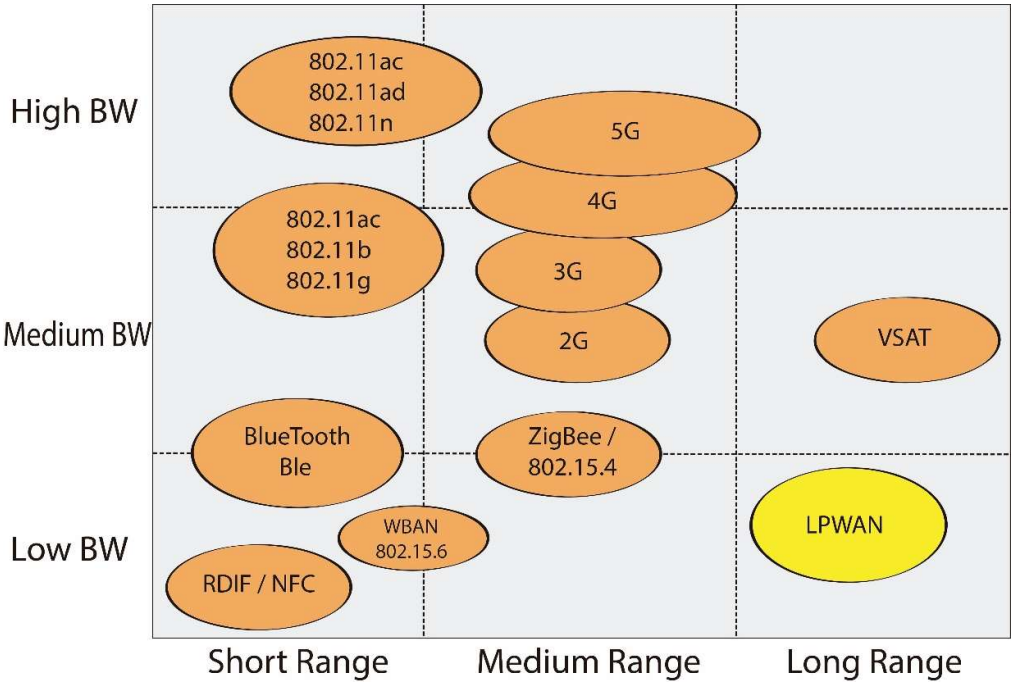


Figura 1.3.- Comparación de tecnologías inalámbrica. (Modificado de Egli, 2015)

En la Figura 1.4 muestra que no existe una tecnología que abarque todos los requerimientos IoT, pero LPWAN se adapta a los requerimientos de cada red y son pequeño volumen de datos, envió de datos concurrentemente y a través de largas distancias.

1.2.5 Redes LPWAN

Las redes LPWAN, permiten la transmisión de pequeños volúmenes de datos a grandes distancias, con la característica de bajo consumo de energía en comparación de las otras redes empleadas en la actualidad, como la de redes telefónicas con la posibilidad de cubrir grandes superficies.

Las redes LPWAN al enviar poco volumen de datos a grandes distancias, no pueden ser implementadas para el uso de texto, voz o video, pero sirve para conectar una gran cantidad de dispositivos con un ancho de banda menor, la aplicación de este tipo de redes es:

- Medicina
- Ciudades Inteligentes
- Dispositivos personales
- Industria
- Agricultura
- Ganadería

Actualmente no está firmemente establecido las características que debe cumplir los dispositivos y las redes IoT, pero entre ellos podemos contar con (López Vicario & Vilajosana Guillen, 2020):

- Dispositivos de reducido tamaño para ser implementados o adaptados y ser utilizados en cualquier lugar tomando en cuenta las limitaciones físicas de fabricación de cada componente.
- Servicio de localización, debe poseer el dispositivo un sistema de localización permite aplicarse en la industria, trabajo u hogar esta prestación.
- Movilidad, la posibilidad de que los dispositivos de conexión y la red deben ser diseñados para tener una movilidad total.
- Comunicación Segura, asegurar la integridad y confidencialidad de los datos, ejemplo información de procesos industriales o información de salud de un paciente.
- Redes que permitan una conexión de amplio alcance y permita en dichas redes la conexión de cientos de dispositivos.
- Bajo Coste de implementación de los dispositivos finales y montado de la infraestructura de red.

Las tecnologías que provee estas prestaciones son las siguientes:

1.2.5.1 Sigfox

Empresa fundada por Ludovic Le Moan y Christophe Fourtet en 2010 teniendo la visión de conectar todos los objetos del mundo físico en el mundo digital desarrolla la tecnología LPWAN (SIGFOX, 2020), permitiendo conectar a dispositivos usando tecnología de banda ultra ancha (UNB).

El protocolo Sigfox se enfoca en:

Autonomía: Bajo consumo de energía.

Simplicidad: Sin configuración, dispositivos funcionando y conectados en poco tiempo.

Eficiencia en coste: Hardware de los dispositivos optimizados para la implementación en la red.

Mensajes pequeños: Datos hasta 12 bytes.

Complementariedad: Bajo coste permite implementar como una solución de otro tipo de red como Wi-Fi, Bluetooth, GPRS, etc...

Características

Sigfox utiliza radio sin licencia ISM en el espectro 868 MHZ para Europa y 902 MHz para Norte América, a bajas velocidades de trasferencias de 10 a 1000 bits por segundo. Consumo de 50 micro vatios con la vida útil de las baterías de 20 años. Alcance de 10 kilómetros en perímetros urbanos y 40 kilómetros en perímetros rurales y con un límite de mensajes diarios por dispositivo a 140.

1.2.5.2 Lora

LoRa (Long Range) es una tecnología inalámbrica de modulación CCS o Chirp Spread Spectrum, aplicada en comunicaciones espaciales y militares, con comunicaciones a larga distancia. Desarrollado por Semtech, pero su administración actualmente la realiza "LoRa Alliance". Utiliza el protocolo de capa MAC (LoRaWAN) con arquitectura de red de acceso específica. LoRa es la capa física o de modulación inalámbrica que crea el enlace de comunicación de largo alcance.

Características

- Desarrollado en modulación "chirp"
- Largo alcance hasta 20 km
- Baja transferencia de datos con límite de 255 bytes.
- Conexión Punto a punto
- Frecuencia utilizada en América 915 MHz
- Potencia de transmisión hasta -168dB
- Factor de dispersión (Spreading Factor SF) de SF7 hasta SF12, SF corresponde la relación entre el ancho de banda y la velocidad de trasmisión, mayor SF implica más alcance y menor tasa de transferencia y viceversa.
- Alcance de 2 kilómetros para SF7 y 14 kilómetros para SF12.
- Se define como la relación que existe del ancho de banda y la velocidad de trasmisión de datos.

1.2.5.3 Ingenu

La empresa Ingenu desarrolla la tecnología RPMA o Randon Phase Multiple Access se implementa para comunicaciones inalámbrica M2M e IoT, opera en la banda ISM con una frecuencia de 2.4 GHz, esta banda tiene una longitud menor por lo que la cobertura es menor en comparación a otras tecnologías con bandas que

utilizan frecuencias menores gigahercios, al verse limitado a implementar menor cantidad de dispositivos por estación base (Ingenu, 2020).

Características

- Comunicación bidireccional.
- Límite de carga de datos hasta 48 bytes.
- Velocidad de datos miles de bytes por segundo.
- Mayor consumo de energía en comparación de tecnologías semejantes.
- Limite envió de paquetes entre 6 bytes hasta 10 Kbyte

1.2.5.4 Weigthless

El estándar Weightless de comunicación inalámbrica de bajo consumo energético, con la opción de implementarse en red públicas o privadas con dispositivos finales del Internet de las Cosas IoT, que operan en cualquier frecuencia, pero está definido actualmente para operar en sub-frecuencias que no requieren licencia que son 138 MHz, 433 MHz, 780 MHz, 915 MHz, 923 MHz (Weightless, 2020).

Características

- Comunicación bidireccional.
- Límite de carga de datos hasta 48 bytes.
- Velocidad de datos de 0.625 kbps hasta 100 kbps.
- Topología estrella.
- Potencia de transmisión estación base y nodo hasta 30 dBm.

1.2.5.5 NB-IoT

NB-FI (NarrowBand-IoT) es un protocolo de comunicación desarrollada por 3GPP para aplicaciones del Internet de las Cosas (IoT), máquina a máquina (M2M) e Industria 4.0. NI-IoT es una tecnología relacionada con LTE que maneja la modulación QPSK y encriptación LTE (Rashmi , Yiqiao , & Seung-Hoon , 2017).

Características

- Comunicación half-duplex.
- Límite de carga de datos hasta 1600 bytes.
- Velocidad de datos de hasta 200 kbps.
- Topología estrella.
- Eficiencia energética media.

1.2.6 Comparativa Tecnologías LPWAN

En la actualidad existe varias tecnologías LPWAN como: SigFox (SIGFOX, 2020), LoRa (LoRa-Alliance, 2020), Weightless (Weightless, 2020), Ingenu (Ingenu, 2020) y NB-IoT (Rashmi , Yiqiao , & Seung-Hoon , 2017).

Tabla 1.3.- Tabla Tecnologías LPWAN

Estándar	Sigfox	LoRa	Ingenu	Weightless	NB-IoT
Banda de frecuencias	868 MHz / 915 MHz	433/868/780/915 MHz	2.4 GHz	138, 433, 780, 915, 923 MHz	Ancho de banda LTE con licencia
Alcance	10 km urbano/ 50 km rural	5 km urbano/ 20 km rural	> 500 km	5km	1 km urbano/ 10 km rural
Tamaño paquete datos	12 bytes	243 bytes	6 bytes - 10 Kbyte	48 bytes	1600 bytes
Máximo mensajes diarios	100 Uplink, 4 Downlink	Ilimitado	384000 por sector	Ilimitado	Ilimitado
Topología	Estrella	Estrella sobre estrella	Estrella	Estrella	Estrella

Las LPWAN permiten implementar dispositivos y cubrir las necesidades para el usuario, la Tabla 1.6 muestra que el LoRa no restringe el número máximo de mensajes, con una cobertura ideal para implementar proyectos en zonas urbanas y lo profundizaremos en la siguiente sección.

1.2.7 Protocolo LoRaWAN

LoRaWAN es un protocolo de red de baja consumo de red y área amplia, clasificando los dispositivos en tres clases, con ancho de banda y frecuencia determinado como se visualiza en la Figura 1.5.

LoRaWAN tiene variaciones de velocidad de trasmisión desde 0.3 kbps a 50 kbps, las cuales gestionan la velocidad de trasmisión individualmente para cada nodo, permitiendo

que cada nodo trasmita datos en cualquier velocidad y canal mediante un esquema de Velocidad de Datos Adaptativa (ADR) (LoRa-Alliance, 2020).

Figura 1.5.- Solución LoRaWAN. Imagen tomada (VARGAS REINOSO, 2018)

1.2.7.1 Arquitectura de red

La arquitectura de la red es del tipo estrella, los dispositivos finales se comunican con varios Gateways y estos a su vez con el servidor. Los elementos que intervienen en una red LoRaWAN ver Figura 1.6 son:

Nodos o Dispositivos finales: Dispositivos que captan datos y los trasmiten, dependiendo de la configuración pueden recibir información por parte del servidor de aplicaciones.

Servidor de red: El Gateway envía la información por medio de Ethernet, Wi-Fi o celular, es el dispositivo que tiene las funciones de administrar la red, administra los mecanismos de activación, gestión de tráfico, enrutamiento de tramas y administración (LoRa-Alliance, 2020).

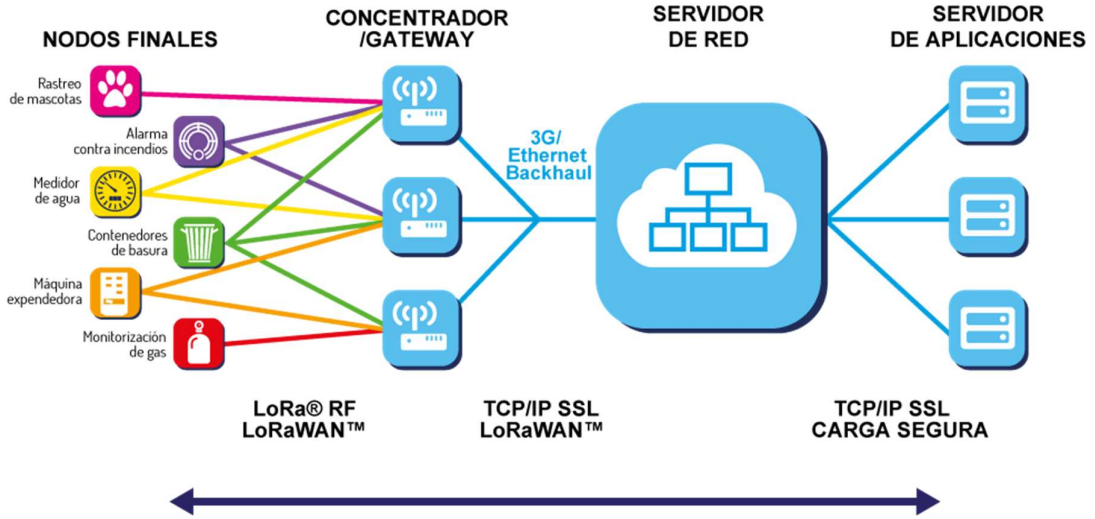


Figura 1.6.- Arquitectura de red LoRaWAN. Imagen Tomada (DEEPDATA CONSULTING, 2020)

1.2.7.2 Especificaciones LoRaWAN

Frecuencia de Canal

En el mundo existe el espectro de banda de radio ISM (Industrial, Científica, Médica) que es libre y basando en el Plan del Canal que es determinado en la zona donde este implementado el Gateway, se presenta los siguientes parámetros ver Tabla 1.7:

Tabla 1.4.- Plan de Canal LoRaWAN. Referenciada (LoRa-Alliance, 2020)

Plan del Canal	Nombre común del plan	Frecuencia (MHz)
EU863-870	EU868	863 - 873
US902-928	US915	902 - 928
CN779-787	CN779	779 - 787
EU433	EU433	433 – 434.79
AU915-928	AU915	915 - 928
CN470-510	CN470	470 - 510
AS923	AS923	902 - 928
KR920-923	KR920	917 - 923.5
IN865-867	IN865	865 - 867
RU864-870	RU864	866 - 868 864 - 865 868.7 - 869.2 433.075 - 434.75 916 - 921

En Ecuador la frecuencia utilizada es de 915-928 MHz con el Plan del Canal US902-928 con las siguientes características:

- Upstream: 64 canales numerados de 0 a 63 utilizando LoRa 125 kHz, variando de DR0 a DR3 utilizando la tasa de codificación 4/5, comenzando em 902.3 MHz, incrementándose linealmente por 200 kHz a 914.9 MHz. Se manejan 8 canales del 64 al 71 utilizando LoRa 500 kHz con DR4 iniciando en 903.0 MHz e incrementándose linealmente por 1.6 MHz a 914.2 MHz
- Downstream: 8 canales numerados de 0 a 7 utilizando LoRa 500 kHz variando de DR8 a DR13 iniciando en 923.3 MHz e incrementándose linealmente por 600 kHz hasta 927.5 MHz (SEMTECH, 2020).

Factor de dispersión

Se define como la relación que existe del ancho de banda y la velocidad de trasmisión de datos. LoRa utiliza diferentes tipos de factores de propagación ortogonales lo que permite que son invisibles entre sí, esto permite que si dos paquetes llegan al mismo tiempo en el mismo canal de recepción no colisionen si tiene diferentes factores de dispersión y llegaran con éxito al gateway, existe otro caso que, si dos paquetes llegan al mismo tiempo en el

mismo canal y con el mismo canal de dispersión podría colisionar, sin embargo si un paquete tiene más dB será el paquete recibido. Un factor de dispersión bajo tendrá un tiempo en el aire menor, esto permite tener en dispositivos un tiempo de vida útil más prolongado de la batería, por lo que a mayor factor de dispersión permitirá una conexión a mayor distancia del gateway (SEMTECH, 2020).

Velocidad de Datos (Data Rate)

La velocidad de datos (Data Rate) con su respectivo factor de dispersión (SF) y la tasa de bits físico definido para la banda US902-928, ver Table 1.8:

Tabla 1.5.- Data Rate para US902-928 (LoRa-Alliance, 2020)

Data Rate	Configuración	Tasa de bits físico [bit/sec]
0	SF10 / 125 kHz	980
1	SF9 / 125 kHz	1760
2	SF8 / 125 kHz	3125
3	SF7 / 125 kHz	5470
4	SF8 / 500 kHz	12500
5:7	RFU	
8	SF12 / 500 kHz	980
9	SF11 / 500 kHz	1760
10	SF10 / 500 kHz	3900
11	SF9 / 500 kHz	7000
12	SF8 / 500 kHz	12500
13	SF7 / 500 kHz	21900
14	RFU	
15	Definido en LoRaWAN	

Tamaño máximo de payload

El tamaño máximo de la longitud de payload que es trasmitido en el tiempo por medio de la cabecera física, Data Rate es la velocidad de datos, MACPayload es el tamaño máximo de un paquete de datos que se puede mandar dependiendo de la velocidad de datos, FOpt es el tamaño máximo de paquete de datos de aplicación, a continuación, se muestra los parámetros en la Tabla 1.9:

Tabla 1.6.- Tamaño de Payload Máximo para US902-928 (LoRa Alliance, 2017)

DataRate	MACPayload (bytes)	FOpt (bytes)
0	19	11
1	61	53
2	133	125
3	230	222
4	230	222
5:7	No definido	
8	41	33
9	117	109
10	230	222
11	230	222
12	230	222
13	230	222
14:15	No definido	

1.2.7.3 Clases de Dispositivos LoRaWAN

LoRaWAN posee tres clases de dispositivos en función de las necesidades de cada aplicación.

1.2.7.3.1 Dispositivos Clase A

Son dispositivos finales que permiten comunicaciones bidireccionales, son dispositivos que pasan la mayor parte del tiempo en un estado inactivo. El dispositivo cuando monitorea un cambio en el entorno se activa e inicia un uplink, enviando datos sobre el cambio de estado. El dispositivo si no reciben un downlink durante Rx1 se pone en suspensión como se muestra en la gráfica 1.7.

Figura 1.7.- Estado suspensión RX1

Cuando el dispositivo pasa al estado de escucha para Rx2, en caso de que no reciba respuestas se pone en suspensión hasta que existan un nuevo cambio en el entorno como se visualiza en la Figura 1.8. Estos dispositivos no se pueden activar desde la aplicación

de administración de dispositivos, esto es un limitante para aplicaciones con dispositivos actuadores.

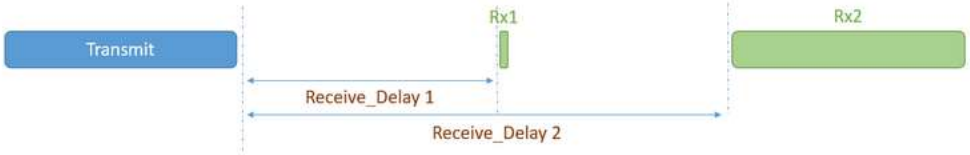


Figura 1.8.- Estado escucha RX2

Los dispositivos no enviaran otro mensaje en downlink hasta que:

1. Se reciba un mensaje en downlink durante Rx1.
2. En Rx2 después de la última transmisión se complete.

1.2.7.3.2 Dispositivo Clase B

Es una mejora de los dispositivos de Clase A, permite programar downlink de la red, esto permite que los dispositivos funcionen para el monitoreo de sensores o actuadores. El funcionamiento de los dispositivos de Clase B se lo realiza mediante el proceso de balizamiento, la red debe emitir una baliza sincronizada a través del tiempo al gateway, el dispositivo final debe recibir periódicamente una de estas balizas que son mensajes programados en el tiempo en downlink al servidor (SEMTECH, 2020).

1.2.7.3.3 Dispositivo Clase C

Son dispositivos que siempre están activos y dependen de la energía externa para funcionar, este tipo de dispositivos son utilizados para medidores eléctricos, etc. El funcionamiento de estos dispositivos tiene el principio de dispositivos de Clase A y B, pero con la diferencia que siempre están a escucha de mensajes donwlink, permitiendo una latencia más baja entre el dispositivo final y el Gateway. Las ventanas de recepción Rx2 no se cierra hasta la siguiente trasmisión de regreso para el servidor (SEMTECH, 2020).

1.2.7.4 Seguridad en LoRaWAN

LoRaWAN especifica llaves de seguridad: NwkSkey, AppSkey y AppKey, estas llaves tienen una longitud de 128 bits utilizando un algoritmo AES-128 (LoRa-Alliance, 2020).

Llaves de Sesión

Cuando un dispositivo realiza un join o procedimiento de unión, la llave de sesión de aplicación AppSKey y la llave de sesión de red NwkSKey son generadas, la llave NwkSKey es compartida en la red mientras la AppSkey es privada, estas llaves son usadas únicamente durante la sesión de conexión.

Llave de sesión de red NwkSKey: se utiliza entre el dispositivo final y el servidor de red, validando la integridad de cada mensaje mediante el código de integridad del mensaje (MIC) utilizado LoRaWAN AES-CMAC.
Llave de aplicación de Sesión AppSkey: se emplea en la encriptación y desencriptación del payload, que es completamente cifrado entre el dispositivo final y el servidor de aplicación, permitiendo leer los mensajes enviados y recibidos únicamente al que posea las llaves.
Las llaves NwkSKey y AppSKey son únicas en cada sesión y por cada dispositivo, si es activado el dispositivo mediante OTAA (The Things Network, 2020).

1.2.7.5 Mecanismos de Autenticación de red

Existen dos tipos de activación en una red LoRaWAN APB y OTAA, que requiere parámetros para acceder correctamente a la red.

1.2.7.5.1 APB (Activation By Personalisation)

Es un mecanismo de autenticación donde el dispositivo necesita tres parámetros para unirse a la red descritos en la Tabla 1.10 y son: llave de sesión de red (NetworkSessionKey), llave de sesión de la aplicación (ApplicationSessionKey) y dirección del dispositivo (DevAddress).

Tabla 1.7.- Parámetros para acceso de red ABP. Tomada de (MEDIUM, 2017)

Parámetros	**Descripción**
DevAddress	Dirección lógica es lo equivalente a una dirección IP que se utiliza para la comunicación con la red.
NetworkSessionKey	Clave de cifrado entre el dispositivo y el Gateway para utilizar en la trasmisión y para validar los mensajes.
ApplicationSessionKey	Clave de cifrado entre el dispositivo y el Gateway a través de la aplicación para trasmitir y validar la integridad de los mensajes.

Con los parámetros mencionados en la Tabla 1.10 la conexión se realiza de la siguiente manera:

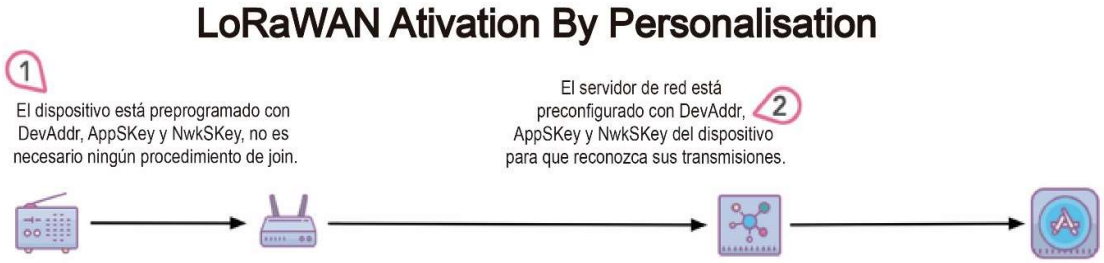


Figura 1.9.- Activation By Personalisation. Imagen extraída (MEDIUM, 2017)

La Figura 1.9 muestra los pasos para realizar la unión a la red y son:

1. El dispositivo final envía los parámetros de unión al Gateway.
2. El Gateway realiza una validación de parámetros de la sesión actual.
3. Si los parámetros de sesión son válidos son procesados caso contrario son rechazados.

El dispositivo no necesita realizar un join a la red para enviar los datos, debido a que no necesita una confirmación par parte del servidor, debido a que la sesión esta manualmente asignada, la desventaja de este tipo de activación es que si las llaves son extraídas se puede obtener los mensajes o clonar el dispositivo.

1.2.7.5.2 OTAA (Over the Air Activation)

OTAA es un mecanismo de autenticación, la sesión “se crea en el aire” es una ventaja al ser más seguro debido a que las llaves de sesión se renuevan cada vez que el dispositivo pierde conexión, es desconcertado o reiniciado, dificultando que las llaves de sesión puedan se robadas o que el dispositivo sea clonado. Es un mecanismo de autenticación más seguro que ABP, los parámetros de configuración son:

Tabla 1.8.- Parámetros para acceso de red OTAA, referenciada de (LoRa-Alliance, 2020)

Parámetros	**Descripción**
DeVEUI	Identificador único de fábrica, este parámetro en ciertos dispositivos es ajustable.
AppEUI	Identificador de aplicación, permite agrupar grupo de dispositivos finales, con un tamaño de 64 bits permite clasificar los dispositivos por aplicación, este parámetro puede configurarse.
AppKey	Llave secreta AES de 128 bits que es compartida entre los dispositivos finales y servidor de red, genera las llaves de sesión.

Con los parámetros mencionados en la Tabla 1.11 la conexión se realiza de la siguiente manera ver Figura 1.10:

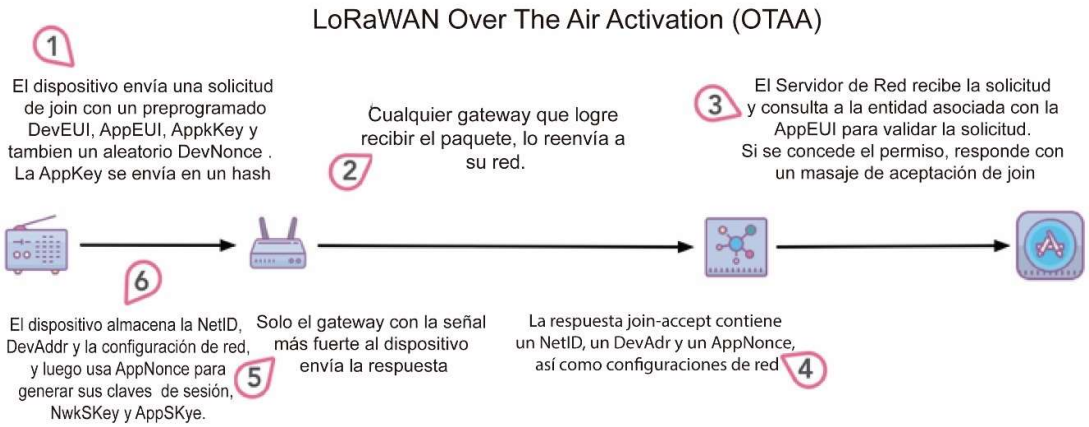


Figura 1.10.- Over the Air Activation (OTAA). (MEDIUM, 2017)

La Figura 1.10 muestra los pasos para realizar la unión a la red:

- El dispositivo realiza un join a la red con la información de configuración.
- El Gateway recibe una solicitud del dispositivo final y envía al servidor.
- El servidor de red comprueba que el dispositivo este dado de alta y que las llaves previamente creadas sean correctas.
- Al validar las llaves crea una sesión temporal y envía los datos de sesión mediante el Gateway al dispositivo final, si los datos no son válidos se rechaza el join.
- El dispositivo recibe la información de sesión temporal y es permitido para enviar información en la red.

CAPÍTULO II

2 METODOLOGÍA

El desarrollo de este proyecto se realizó mediante la metodología PPDIOO, complementándolo para el desarrollo de software con el Marco de Trabajo SCRUM, fragmentando las historias de usuario en las correspondientes fases de la metodología PPDIOO, las evidencias técnicas del desarrollo de cada Sprint se encuentran en el Anexo I.

2.1 Metodología PPDIOO

La metodología PPDIOO define al ciclo de vida de una red en seis fases: Preparación, Planificación, Diseño, Implementación, Operación, y Optimización. Las fases tienen una función definida y se relacionan con su antecesora y predecesora.

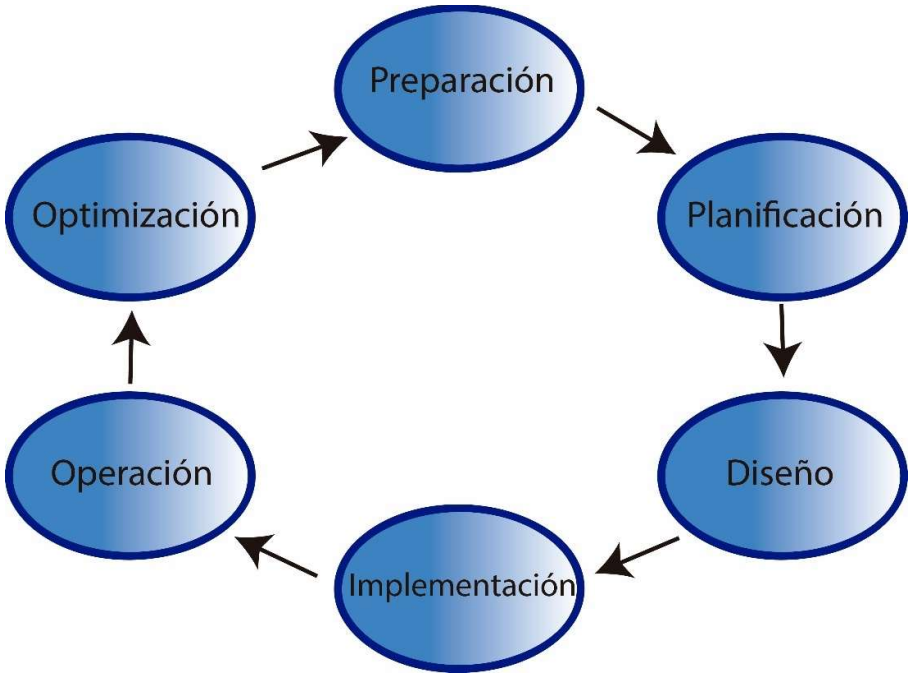


Figura 2.1.- Fases del Ciclo de Vida de la Metodología PPDIOO (CISCO, 2014)

2.1.1 Fases de la Metodología PPDIOO

Preparación: Esta fase permitió analizar las características técnicas de los equipos del Smart Lab de la Escuela Politécnica Nacional, laboratorio del cual se tuvo el apoyo técnico para el desarrollo del presente proyecto. Se analizaron las características de los equipos, software, medios de transmisión y herramienta de desarrollo para el desarrollo del backend y frontend.

Planificación: En esta fase se analizó los requerimientos técnicos necesarios para conectar los dispositivos finales y las configuraciones pertinentes del Gateway y servidor del laboratorio Smart-Lab para el correcto funcionamiento del proyecto.

Diseño: En esta fase se diseñó la programación de los dispositivos Arduino en base a los requerimientos obtenidos en la fase de planificación.

Implementación: Se ha desarrollado el sistema de monitoreo del aire basado en el diseño realizado en la fase anterior.

Operación: En esta fase se puso en funcionamiento, el sistema desarrollado. También se probó el desempeño de los dispositivos.

Optimización: En esta fase y se realizó la corrección fallos detectados, antes de realizar la implementación real.

2.2 Metodologías ágiles de desarrollo de software

Las metodologías ágiles garantizan el desarrollo de proyectos a corto plazo, que se adapten a varios tipos de requerimientos. Las metodologías ágiles que permiten aumentar la calidad y aseguramiento de que un producto de software de un proyecto tenga éxito siguiendo una secuencia de pasos para estructurar, planear y controlar procesos de desarrollo hasta el final del proyecto. Scrum es un marco de trabajo usado a nivel mundial por una gran cantidad de organizaciones a nivel mundial debió a su simplicidad adaptabilidad y flexibilidad que permite entregables continuos, esto permite determinar que este marco de trabajo es ideal para implementar el desarrollo de software para el proyecto (Xavier Villamil, 2019).

2.2.1 Marco de Trabajo Scrum

Scrum presenta un proceso adaptativo, rápido y organizado para desarrollar un producto. Se centra en la gestión de los proyectos de difícil planificación, con bucles de control que permiten la retroalimentación para el mejoramiento del producto. El desarrollo es en equipo organizado en incrementos denominado sprint, la coordinación de equipos, las características que se implementaran en el backlog y el Product Owner determina los elementos a ser implementados en el siguiente sprint. El Scrum Master resuelve los problemas para que el trabajo en equipo funcione eficazmente.

Al momento de desarrollo de proyectos de dispositivos móviles los requisitos son diversos e implementados con tecnologías diferentes, por lo que es recomendable que el sprint inicial tenga la funcionalidad con la tecnología (Balaguera, 2015).

2.2.1.1 Proceso Scrum

El proceso scrum para el desarrollo de proyectos se realizan en ciclos cortos o iteraciones de 2 semanas hasta un límite de 4 semanas, con el objetivo de tener resultados completos de cada iteración, dando el resultado final el producto entregado con un mínimo esfuerzo al cliente en función de los requerimientos, ver Figura 2.2.

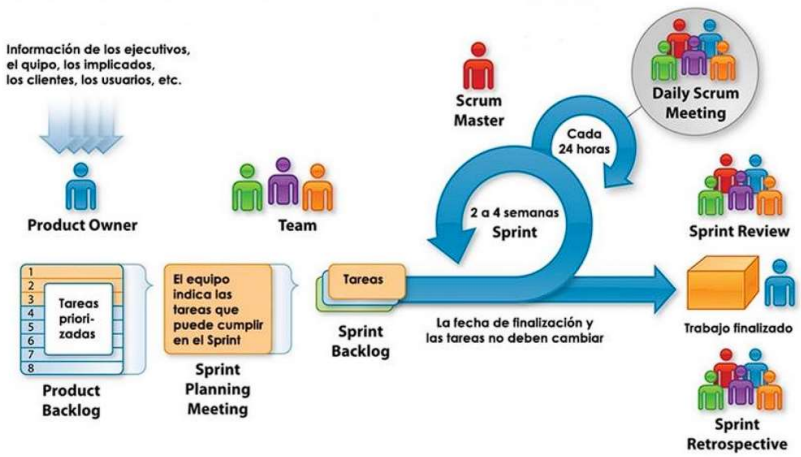


Figura 2.2.- Marco de trabajo de SRUM, (Schwaber & Sutherland, 2013)

2.2.1.2 Eventos Scrum

Los eventos Scrum permiten reducir reuniones no definidas en Scrum, permitiendo al equipo de trabajo tener una mejor comunicación y colaboración minimizando el tiempo en reuniones. El tiempo de cada sprint es fijo por lo que la duración no puede aumentar o acortar al ya establecido. Los eventos de Scrum son:

2.2.1.2.1 Sprint

Es un lapso o time-box con una duración de un mes o menos, periodo en que el producto crea un incremento al producto final. Es recomendable que la duración de los Sprints sea definida en función del esfuerzo que va a ser tomada a cada uno. El Sprint siguiente empieza inmediatamente que finaliza su antecesor.

El Sprint es el resultado de la panificación del Sprint (Sprint Planning), los Scrums Diarios (Daily Scrum), el desarrollo del producto, la Revisión del Sprint (Sprint Review), Restrospectiva del Sprint (Sprint Retrospective).

En el desarrollo del Sprint se debe tomar en cuenta los siguientes puntos:

- La calidad de objetivo no cambia o disminuye.
- Los objetivos del Sprint no son cambiados.

- El alcance puede cambiar entre el Product Owner y el equipo que se encarga en el desarrollo del proyecto.

2.2.1.2.2 Sprint Planning

El Sprint Planning es el trabajo que se va a realizar durante el Sprint, el plan se crea con el Equipo Scrum.

El tiempo de Planificación de Sprint tiene un límite máximo de 8 horas por cada Sprint que dure un mes, si los Sprint son menores se reducirá el tiempo de planificación. El Scrum Master es la persona encargada de que este evento se cumpla satisfactoriamente y que cada actor entienda el objetivo que debe desempeñar dentro del tiempo establecido, tomando en cuenta del producto resultante de cada Sprint y los mecanismos para lograr los objetivos dentro del bloque de tiempo.

2.2.1.2.3 Daily Scrum

Daily Scrum es una reunión diaria de duración de 15 minutos con el equipo de desarrollo durante cada Sprint. El equipo de desarrollo se plantea la meta de trabajo de las siguientes 24 horas, permitiendo optimizar el desempeño y la colaboración. Daily Scrum se realiza en un horario preestablecido reduciendo la complejidad de coordinación con el equipo.

2.2.1.2.4 Sprint Review

Es la inspección del desarrollo del Sprint y la lista de productos a ser adaptados si es necesario. La reunión se desarrolla para que cada actor del desarrollo y planificación del proyecto aporte ideas de cómo mejorar los siguientes Sprint y como obtener el un mayor valor del producto final.

2.2.1.2.5 Sprint Retrospective

Sprint Retrospective permite la retroalimentación del Equipo Scrum para autoevaluarse y crear un plan para mejorar y optimizar el trabajo en los siguientes Scrum.

Se analiza ultimo Sprint de como fue el desenvolvimiento de las personas, el conocimiento del proceso y el uso de las herramientas. Los elementos prioritarios que fueron satisfactorios y como mejorar el proceso en general de desarrollo.

2.2.1.3 Artefactos Scrum

Los Artefactos Scrum representa el valor que permite dar trasparencia y oportunidad de tener adaptación e inspección. Facilita el entendimiento de todo el grupo de Scrum de cada uno de los procesos a realizarse durante el proyecto.

2.2.1.3.1 Product Backlog

Es la lista ordenada de producto de lo que es necesario en el producto, es la fuente de requisitos para realizar cambios en el producto. La lista de Productos muestra la funcionalidad, requisito, posibles mejoras y correcciones para entregas futuras. Esta lista es dinámica se adapta a los nuevos sucesos durante el desarrollo para obtener un producto útil, adecuado y competitivo.

2.2.1.3.2 Sprint Backlog

Sprint Backlog es la lista de pendientes del Sprint, es la predicción del equipo de desarrollo que se realizara en los próximo Sprint como funcionalidad y tener el estado de Sprint Terminado. Esta lista permite ver que hace falta para lograr los siguientes objetivos de los Sprint, esto permite el mejoramiento continuo.

2.2.1.3.3 Sprint

El Incremento es la somatización los elementos de Product Backlog completos de un Sprint y el valor de los incrementos de anteriores Sprints.

2.2.1.4 Equipo Scrum

2.2.1.4.1 Product Owner

Product Owner rol que se encarga de optimizar al máximo el valor del producto final realizado por el Equipo de Desarrollo. Es el encargado de administrar el Product Backlog y es el que canaliza los requerimientos del cliente.

2.2.1.4.2 Equipo de desarrollo

Es un equipo de profesionales que tiene por objetivo cumplir cada Sprint, es un grupo autoorganizado, multifuncional, con habilidades especializadas en cada área de trabajo, pero compartiendo responsabilidad como Equipo de Desarrollo.

2.2.1.4.3 Scrum Master

Scrum Master persona responsable en cumplir el entendimiento y la aplicación de la teoría, práctica, reglas y valores de Scrum a las personas externas e internas el Equipo Scrum para hacer entender cada iteración durante el desarrollo de un proyecto (Schwaber & Sutherland, 2017).

2.3 Desarrollo

La propuesta es desarrollar un aplicativo móvil y backend para proyecto, con interfaces de fácil visualización de datos para ver la contaminación del aire, con herramientas de software intuitivas que faciliten el desarrollo del backend y el frontend, tomando en cuenta

que son datos de los dispositivos diseñados obtenidos por el Gateway LoRaWAN y filtrados por Node-RED para su posterior almacenamiento en una base de datos.

2.3.1 Fase Preparación

El laboratorio Smart Lab cuenta con un Gateway MultiTech MTCPTIP-LEU1-266A diseñado para LoRa y puede ser implementado en redes públicas y privadas, las especificaciones técnicas son (Tabla 2.1):

Tabla 2.1.- Descripción componentes Gateway MultiTech MTCPTIP-LEU1-266, referenciada de (Multitech, 2019)

Especificación	Descripción
Procesador y memoria	ARM9 procesador con 32-Bit ARM & 16-Bit Thumb instruction sets 400 MHz, 16K Data Cache, 16K Instruction Cache, 128X16 MB DDR RAM, 256 MB Flash Memory
Modo de Red	Publica / Privada
Wi-Fi/Bluetooth (-267 models)	Wi-Fi: 802.11abng (2.4 & 5 GHz) / Bluetooth: Classic 4.1 and BLE
GPS/GNSS	GNSS for LoRa Packet Time Stamping Concurrent GNSS connections: 3 GNSS Systems Supported: (default: concurrent GPS/QZSS/SBAS and GLONASS)
LoRa Especificaciones	
Frecuencia de Banda	915 MHz
Plan del Canal	US915
Capacidad del Canal	8 canales half-duplex
Power Output	27 dBm
Conexiones	
E-NET	RJ45 Ethernet jack (10/100 port)
USB HOST	USB 2.0 Conector Tipo A
Antenas	Cellular, GPS, LoRa
Certificaciones	
Calidad	MIL-STD-810G: funcionamiento en ambientes de humedad, polvo, viento, lluvia, nieves y calor extremo

Perfiles configurados en el Gateway Tabla 2.2:

Tabla 2.2.- Perfil Gateway LoRaWAN

Perfil	Detalle
Perfil de dispositivo	LW102-OTA-US915: LoRaWAN v1.0.2, OTAA, Plan del Canal US915
Perfil de red	CLASS A/B/C

Se realizó un análisis de la configuración del Gateway que admite dispositivos de diferentes clases, pero para aplicar en el proyecto se utilizó la clase A debido a las ventajas de dispositivos que tienen mayor eficacia de energía, mayor soporte a dispositivos y Downlink disponible solo después de que el sensor trasfiera información.

La Red LoRaWAN que posee Smart Lab ver Figura 2.3, permite a los sensores conectarse a un Gateway, el Network Server desencripta los mensajes enviados por los sensores y encriptarlos para enviarlos a los sensores, el Aplication Server procesa los datos mediante Node-RED.

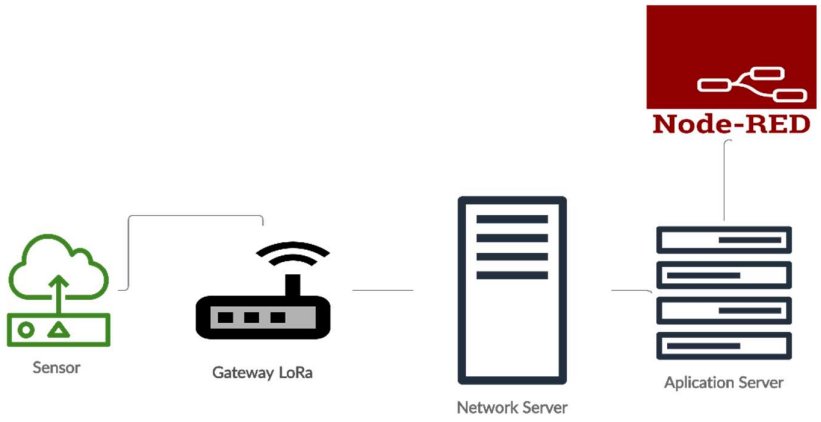


Figura 2.3.- Diagrama de la Red LoRaWAN Smart Lab

2.3.2 Fase Planificación

Al poseer un Gateway que permite el uso de LoRaWAN se analizó que componentes se necesita y que tecnología es compatible para ser implementada. Con la generación del Product Backlog permitió ajustar con cada fase de la metodología PPDDIO su respectivo Sprint.

2.3.2.1 Software

En la Tabla 2.3 detalla la definición de las herramientas de software aplicadas en el proyecto.

Tabla 2.3.- Tabla herramientas de software utilizadas

Herramienta	Definición
Arduino IDE	Software de código abierto permite la escritura del código para ser implementado en la placa, entorno basado en Java, Processing y otros softwares open source. El estándar para crear código para Arduino es un lenguaje C++ adaptado de la avr-libc una librería C, para utilizar con GCC en microcontroladores ARV Atmel. avr-gcc procesa el código del sketch en C++ y lo transfiere a un fichero binario (.hex) que se carga en la memoria del Arduino para ser ejecutado (Arduino, 2020).
Librería LMIC	La librería IBM LMIC (LoraMAC-in-C) permite el uso de transceptores SX1272, SX1276 con compatibilidad de módulos RFM95. La librería permite una implementación completa con LoRaWAN clase A y B, admite bandas EU-868 y US-915. Características: • Envió de paquetes Uplink, respetando el ciclo de trabajo. • Comprobación de integridad del mensaje y el cifrado. • Recepción de paquetes en Downlink en ventana RX2. • Activación por aire (OTAA).
Node-RED	Node-RED es una herramienta de programación para aplicaciones impulsada por eventos que conecta hardware, API, y servicios en línea. El editor de flujo permite una conexión entre flujos con una amplia gama de funciones en la paleta, es desarrollado con Node.js por lo que permite la ejecución de forma rápida.

2.3.2.2 Hardware

Arduino

Arduino es una plataforma para el desarrollo de proyectos electrónicos en una placa de hardware libre con microcontroladores programables y pines que permiten la conexión con diversos sensores y dispositivos compatibles con el hardware (Arduino, 2020). Consta de una "Printed Circuit Board" (PCB) o Placa de Circuito impreso que son configuradas cada placa con diferentes prestaciones, que se adaptan a la necesidad de cada proyecto con su core de microcontroladores AVR marca Atmel, compartiendo software, arquitectura, documentación y librerías para implementar proyectos.

Modelos de Arduino

En la Tabla 2.4 se muestra tres modelos de Arduino con sus características:

Tabla 2.4.- Características modelos Arduino Nano, Uno, Mega, datos tomados (Arduino, 2020)

Modelo	Arduino Nano	Arduino Uno	Arduino Mega
Microcontrolador	ATmega328	ATmega328p	ATmega2556
Voltaje operativo	5V	5V	5V
Pines Digitales E/S	22(6 de salida de energía)	14(6 de salida de energía)	54(15 de salida de energía)
Pines Analógicos de entrada	8	6	16
Corriente DC por pin de E/S	40 mA	20 mA	20 mA
Corriente DC pin 3.3V	-	50 mA	50 mA
Flash Memory	32 KB	32 KB	256 KB
SRAM	2 KB	2 KB	8 KB
Velocidad del Reloj	16 MHz	16 MHz	16 MHz
Longitud	18 mm	68.6 mm	101.52 mm
Ancho	45 mm	53.4 mm	53.3 mm
Peso	7 g	25 g	37 g

Arduino Uno

Para el desarrollo del proyecto será de utilidad el modelo de Arduino UNO (Figura 2.4), que posee un microprocesador Atmega328p de bajo consumo de energía, ejecuta instrucciones

en un solo ciclo de reloj esto permite optimizar el poder de consumo versus la velocidad de procesamiento (Arduino, 2020).

Hardware

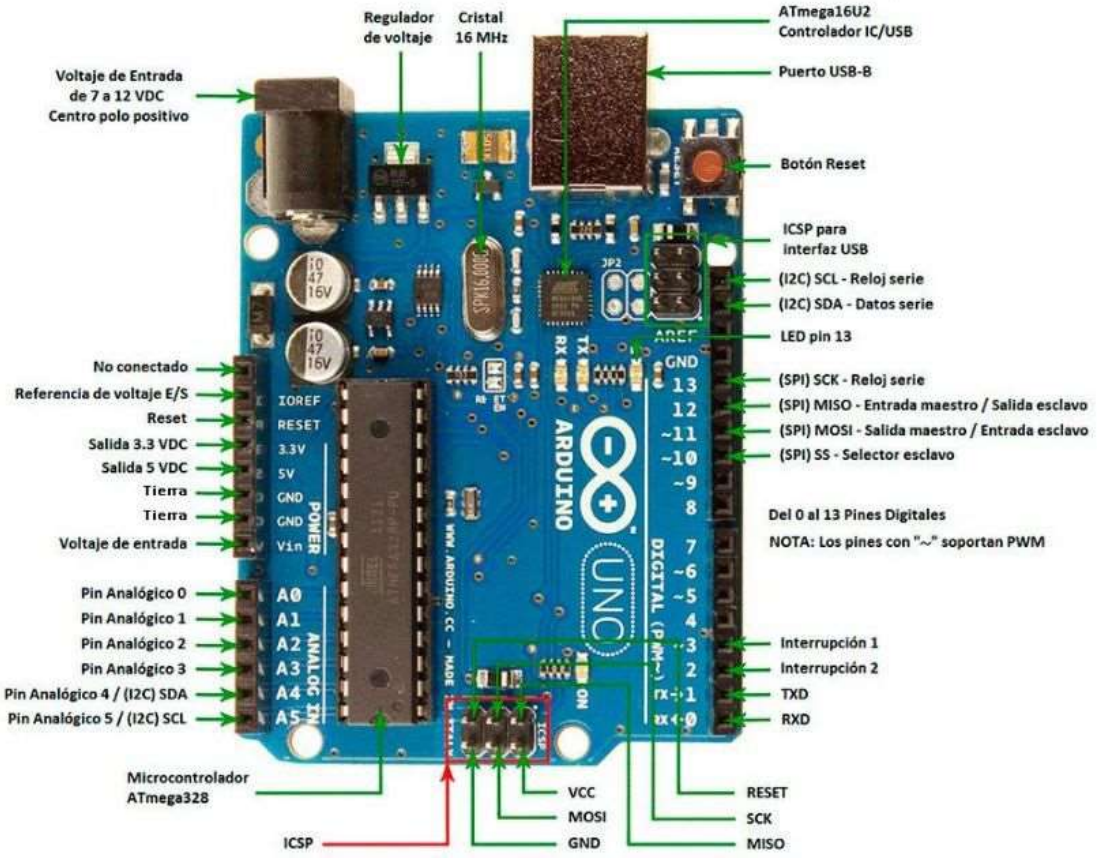


Figura 2.4.- Arduino Uno. Imagen tomada de (Arduino, 2020)

Descripción componentes Arduino Uno, ver Tabla 2.5:

Tabla 2.5.- componentes principal Arduino UNO referenciada (Arduino, 2020).

Componente	Descripción
Botón Reset	Inicializa el programa cargado al microcontrolador, permite restablecer el Arduino en caso de que no responda.
Puerto USB	Funciona para trasferir los programas al microcontrolador y recepción de datos y proveer energía a la placa.
Pin Analógico de entrada o salida	Pines que funcionan para conectar sensores o actuadores de señal analógica.

Regulador de Voltaje	Controla la cantidad de electricidad a la placa.
Pin Digital de entrada o salida	Pines que funcionan para conectar sensores o actuadores de señal digital.
ICPS	Siglas In-Circuit Serial Programming: permite comunicarse con dispositivos externos en una conexión tipo bus denominando al microcontrolador de la placa de Arduino como maestro y los demás dispositivos conectados al bus como esclavos.

Microcontrolador ATmega328p

Es un microcontrolador con una arquitectura RISC de AVR de Atmel, optimizado para compiladores C, ver Figura 2.6.

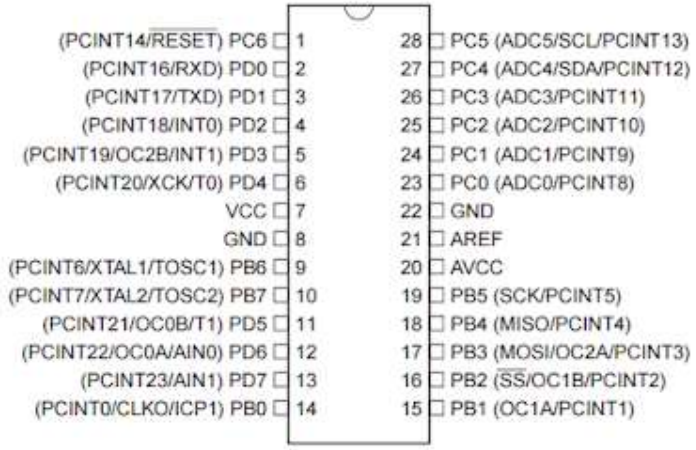


Figura 2.6.- Microcontrolador ATmega328p. Imagen tomada de (Microchip, 2018)

Descripción de pines (Figura 2.6)

VCC: Fuente de alimentación digital

GND: Tierra

Puerto B (PB#): Puertos de entrada y salida bidireccional de 8 bits, resistencias pull-up internas.

Funciones de los Puertos B son:

- XTAL: Refiere a oscilador de cristal

- SPI: Serial Peripheral Interface estándar de comunicación para trasferir información entre circuitos.
- Comparador de salida para Timers.

Puertos C (PC#): Puertos de entrada y salida bidireccional de 7 bits, sus funciones son entradas analógicas (ADC) que permiten la conversión analógica-digital de las señales.
Puerto PC6/Reset: Puerto utilizado como reset de entrada del microcontrolador.
Puertos D (PD#): Puerto de entrada y salida bidireccional de 8 bits.
Funciones de los Puertos D son:

- Puerto serial USART (Universal Synchronous/Asynchronous Receiver Transmitter) Transmisor-Receptor Síncrono/Asíncrono Universal): Utilizado para trasmitir datos entre microcontroladores (MrElberni, 2020).
- Comparador de salida para Timers.

AVcc: Pin de voltaje para la conversión analógica-digital (ADC)
AREF: Pin de referencia analógica para ADC (Microchip, 2018).

Arduino LoRa Shield

Son transceptores de largo alcance implementados en un Arduino Shield, que se fundamentan en bibliotecas de código abierto, estos dispositivos permiten el envío de datos a rangos largos con un bajo volumen de datos, con baja interferencia de frecuencia.
El sx127x Shield se basa en un chip Semtech sx1276/sx1278 se aplica en redes de sensores inalámbricos para aplicaciones de sistemas para la agricultura, medición con sensores inteligentes, ciudades inteligentes, la industria, etc. (DRAGINO, 2020)

Características

- Bajo consumo de energía
- Compatibilidad con Arduino Mega, Uno, Leonardo y DUE con voltajes de entrada y salida de 3.3 o 5 voltios.
- Antena externa compatible con el conector I-Pex
- Frecuencias compatibles dependiendo de las configuraciones de fábrica 915 MHz para América, 433MHz y 868 MHz para Europa.
- Velocidad de bits hasta 300 kbps.
- Bajo consumo de RX de 10.3 mA.

Hardware

Dragino Lora Shield v1.4 permite la conexión de varios dispositivos con compatibilidad SPI (Figura 2.7).

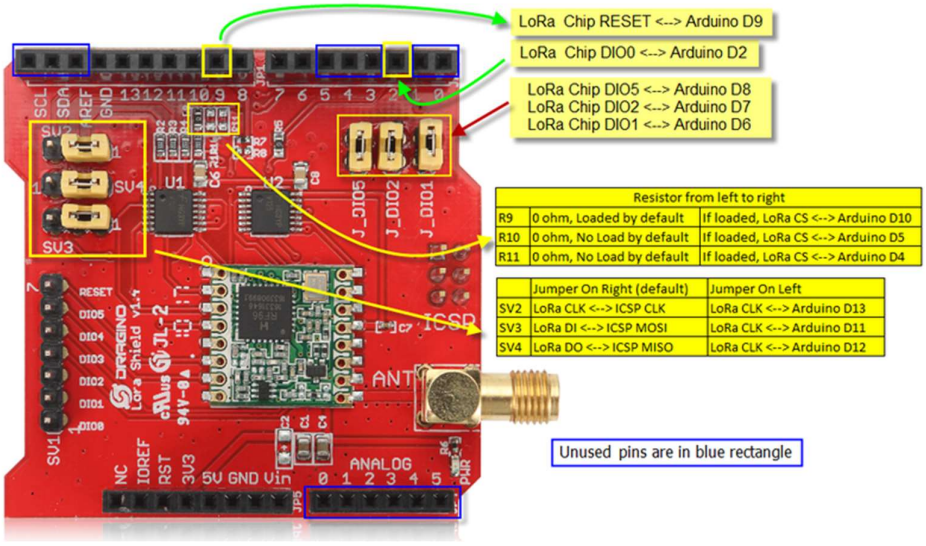


Figura 2.7.- Dragino LoRa Shield. Imagen tomada de (Dragino, 2017)

Funciones pines Lora Shield v1.4 son:

Reset: Permite reinicia el Shield

DIO0: Trasmisión (Tx) y recepción (Rx) hecha

DIO1: Tiempo de espera recepción

DIO2: Tiempo de espera (Kooijman, 2020)

Sensor Mq-7

El sensor Mq-7 es empleado para la detección de Monóxido de Carbono (CO), permite ver las concentraciones de monóxido de carbono en el aire y depende de la calibración que se realice al censor, ver Tabla 2.6 (Hanwei Electronics CO .LTD, 2019).

Tabla 2.6.- Características MQ-7 tomada (Hanwei Electronics CO .LTD, 2019)

Descripción	**Detalle**
Voltaje de Calentamiento	1.4 voltios bajo y alto de 5 voltios
Resistencia de carga	Regulable con el potenciómetro permite aumentar o disminuir la sensibilidad digital al CO.
Consumo de Resistencia	350mW
Temperatura de uso	-20 ℃ hasta 50 ℃

2.3.2.3 Configuración de conexión

Cambios Librería LMIC

Se hizo cambios en los archivos de config.h, main.cpp y lmic.c para poder tener una conexión correcta con el Gateway indicados a continuación:

config.h

Se cambió la línea #define CFG_eu868 1 por #define CFG_us915 1

Se realizó el cambio en lmic.c:

```
        LMIC.channelMap[channel/4]  &= ~(1<<(channel&0xF));
```

Por:

```
        LMIC.channelMap[channel/16]  &= ~(1<<(channel&0xF));
```

Cambio en main.cpp:

```
uint32_t freq = 868100000;
```

Por

```
uint32_t freq = 915000000;
```

Creación de llaves

Se debe crear llaves para los dispositivos para ser conectados mediante la conexión OTAA el cual se necesita crear llaves parar DevEUI, AppEUI y AppKey que será implementado con clase A y canal US915.

Las credenciales asignadas se las crea en lenguaje hexadecimal (Tabla 2.7). AppEUI puede ser igual entre los LoRa, pero por motivos de seguridad es crear AppEUI únicos.

Tabla 2.7.- Configuración de llaves del dispositivo final

Parámetro	**Detalle**
DevEUI	8 bytes
AppEUI	8 bytes
Appkey	16 bytes
Clase	A
Perfil de dispositivo	LW102-OTA-US915
Perfil de Network	DEFAULT-CLASS-A

Estas llaves son únicas para cada uno de los dispositivos con un total de dos LoRa para ser implementados en el proyecto.

2.3.2.3 Product Backlog

El desarrollo del proyecto inicio el lunes 6 de julio del 2020 y finalizo el martes 15 de septiembre del 2020, con una duración de dos meses y medio aproximadamente y un recurso humano asignado.

El aplicativo monitorizará los valores obtenidos por los sensores esto implica que donde este desplegado el backend en la nube siempre deberá estar disponible para que el usuario pueda observar los valores.

En esta fase del Marco de Trabajo Scrum se especifica el Product Backlog en función de los requerimientos de usuario para implementar tareas en cada Sprint. Las historias de usuario fueron implementadas en dos tipos las que contribuyen con funcionalidad y valor al usuario y los de requerimientos no funcionales que aportan características como eficiencia, tolerancia a fallos, etc.., ver Tabla 2.8.

Tabla 2.8.- Historias de usuario

ID	Nombre de la Historia	Descripción	Prioridad
A0	Plataforma Tecnológica	Como sistema, necesito analizar los requerimientos técnicos que se utiliza para implementar en el proyecto.	Alta
A1	Gestión de conexiones del backend	Como sistema, necesito un backend que me permita gestionar las conexiones con el MQTT y el frontend	Alta
A2	Gestión de datos del backend	Como sistema, necesito un backend que me permita gestionar los datos categorizándolos, guardarlos en la base de datos en formato correcto y generar rutas de consulta para el frontend.	Alta
B1	Desarrollo interfaz principal de la aplicación móvil	Como usuario, necesito una interfaz principal que me permita visualizar los datos de forma gráfica y registros de la contaminación del aire.	Alta
C1	Desarrollo interfaz datos diarios de la aplicación móvil	Como usuario, necesito una interfaz que me permita visualizar los datos de forma gráfica y picos de medición de un día determinado.	Media
D1	Corrección bugs backend y frontend	Como aplicación, necesito corregir los bugs de manejo de datos en el backend y frontend, para la publicación del backend en la nube y el compilado de la aplicación para Android., para la publicación del backend en la nube y el compilado de la aplicación para Android.	Media

2.3.2.4 Sprint Planning

Para completar el Sprint Planning, se tiene que verificar que el Product Backlog este bien definido, esto permitirá estructurar los Sprint requeridos priorizando su importancia para el desarrollo del proyecto, y obtener los mejores resultados para tener entregables en las fechas establecidas.

Sprint 0

Tabla 2.9.- Product Backlog Sprint 0

ID	Historia de Usuario	Descripción	Puntos Estimados	Inicio	Fin	Como probarlo	Release
A0	Plataforma Tecnológica	Como sistema, necesito analizar los requerimientos técnicos que se utiliza para implementar en el proyecto.	5	06/07/2020	10/07/2020	Verificar que Flutter y JavaScript este correctamente instalado en Visual Studio Code. Las librerías NodeJs, ExprerssJs, MQTT.js, Mongoose, MongoDB sean compatibles con la versión de JavaScript instalada	V1.0

Sprint 1

Tabla 2.10.- Product Backlog Sprint 1

ID	Historia de Usuario	Descripción	Puntos Estimados	Inicio	Fin	Como probarlo	Release
A1	Configuración Arduino con Dragino LoRa Shield y Node-RED	Como sistema, necesito configurar el Arduino que me permita enviar los datos captados por el sensor MQ-7, a	8	13/07/2020	17/07/2020	Comprobar que los dispositivos capten datos por medio del MQ-7 y los envíen a CloudMQTT	V1.0

		través del Dragino LoRa Shield al Gateway que sean filtrados por Node-RED y la información sea enviada a CloudMQTT.					
A2	Gestión de conexiones del backend	Como sistema, necesito un backend que me permita gestionar las conexiones con CloudMQTT y la base de datos.	8	20/07/2020	28/07/2020	Comprobar por consola que el backend estableció conexión con el CloudMQTT y está en escucha y se estableció conexión con la base de datos con las credenciales previamente definidas.	V1.0
A3	Gestión de datos del backend	Como sistema, necesito un backend que me permita gestionar los datos categorizándolos y guardándolos según el modelo de la base de datos y generar rutas de consulta para el frontend.	8	29/07/2020	07/08/2020	Verificar que el backend guardo los datos según el modelo de base creado ver Tabla AI4. Comprobar realizando consultas al backend y que retorne los valores guardados en la base de datos y calculados por el backend.	V1.0

Sprint 2

Tabla 2.11.- Product Backlog Sprint 2

ID	Historia de Usuario	Descripción	Puntos Estimados	Inicio	Fin	Como probarlo	Release
B1	Desarrollo interfaz principal de la aplicación móvil	Como usuario, necesito una interfaz principal que me permita visualizar los datos de los sensores del proyecto representados en forma de gráfico de puntos y mostrar registros de la contaminación del aire en forma de lista y ultimo registro.	8	10/08/2020	26/08/2020	Acceder mediante un dispositivo Android e interactuar con la gráfica de puntos para visualizar los datos de un punto determinado y ver los datos de los últimos 40 minutos desplegados en una lista.	V2.0

Sprint 3

Tabla 2.12.- Product Backlog Sprint 3

ID	Historia de Usuario	Descripción	Puntos Estimados	Inicio	Fin	Como probarlo	Release
C1	Desarrollo interfaz datos diarios de la aplicación móvil	Como usuario, necesito una interfaz que me permita visualizar los datos de los sensores del proyecto en forma de gráfica de puntos y mostrar los registros de picos de medición de un día determinado.	5	27/08/2020	07/09/2020	Acceder al Widget day, interactuar con la gráfica y visualizar los datos de un punto determinado y ver los datos de pico diario.	V2.0

Sprint 4

Tabla 2.13.- Product Backlog Sprint 4

ID	Historia de Usuario	Descripción	Puntos Estimados	Inicio	Fin	Como probarlo	Release
D1	Corrección bug backend y frontend	Como aplicación, necesito corregir los bugs de manejo de datos en el backend y frontend, para la publicación del backend en la nube y el compilado de la aplicación para Android., para la publicación del backend en la nube y el compilado de la aplicación para Android.	5	08/09/2020	15/09/2020	Comprobar que los datos se guarden en formato correcto y las gráficas se visualicen bien. Se pueda ejecutar la aplicación en Android y el backend se ejecute correctamente en la nube.	V2.0

2.3.3 Fase Diseño

En esta fase se procedió a diseñar los dispositivos y diagramar la arquitectura de red del proyecto.

2.3.3.1 Sprint 0

Sprint 0 Historia de Usuario A0
El Sprint 0 se tiene la siguiente historia de usuario:

Tabla 2.14.- Sprint 0 Historia de Usuario A0

Historia de Usuario	
Número: A0	**Usuario:** Sistema
Nombre historia: Plataforma Tecnológica	
Prioridad en negocio: Alta	
Puntos estimados: 5	**Sprint: 0**
Asignado a: Jorge Miño	
Descripción: Como sistema, necesito analizar los requerimientos técnicos que se utiliza para implementar en el proyecto.	
Criterio de aceptación: Se debe verificar que los servicios que se va a implementar en el proyecto estén instalados y funcionando correctamente son: Visual Studio Code, NodeJs, ExprerssJs, MQTT.js, Mongoose, MongoDB, Flutter.	

Arquitectura del Proyecto

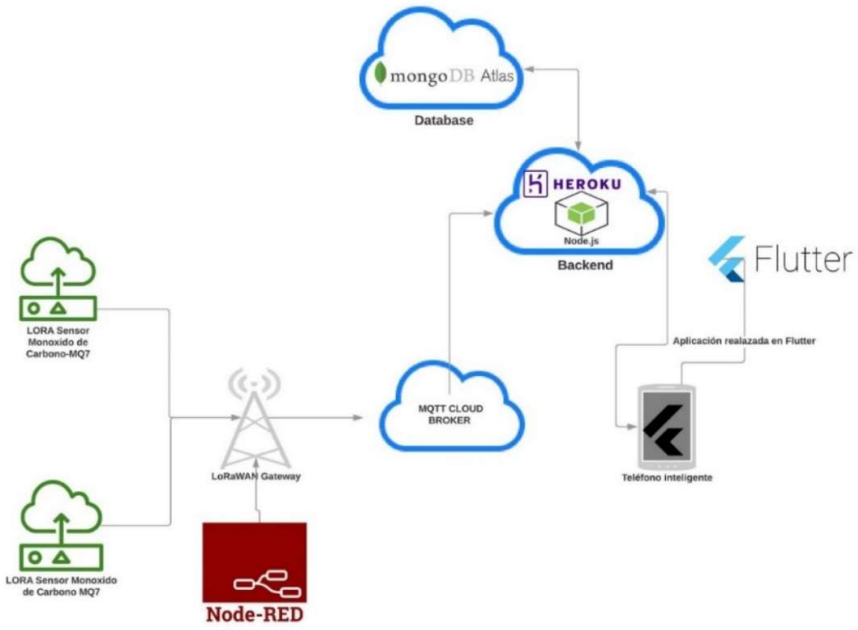


Figura 2.8.- Arquitectura del Proyecto

La Figura 2.8 describe la arquitectura del proyecto:

1) El sensor capta datos del entorno.
2) El Gateway recibe los datos, Node-RED los procesa y envía a CloudMQTT.
3) El backend recibe los datos de CloudMQTT y los guarda en la base de datos.
4) El usuario ejecuta el aplicativo móvil desde el celular.
5) El usuario realiza selecciona que sensor desea visualizar.
6) El backend procesa la petición y envía los datos al aplicativo móvil.
7) El aplicativo móvil recepta la información y la muestra en los diferentes Widgets.

2.3.3.1 Diseño de los dispositivos de medición contaminación

Se esquematizo que el Dragino LoRa Shield con el Arduino UNO y el sensor MQ-7 se lo conecte al pin Analógico y en el pin de voltaje de 5V, debido a que los datos que tomara los sensores serán del tipo analógico.

Inicialmente se tomó en cuenta implementar una power bank con panel solar con capacidad de 5000 mAh, pero debido a la frecuencia de envió y que la librería no está optimizada para el funcionamiento de OTAA se optó que el diseño se conecte a la una fuente de energía permanente con el voltaje de 5V y 1 Amperio.

Debido a que el diseño será implementado en exteriores, se realizó el siguiente diseño, que consta en cubrir el Arduino y los otros componentes electrónicos con un envase de plástico que los protegerá la lluvia y del polvo preservando la integridad y funcionamiento de los dispositivos, ver Figura 2.9. El sol es un factor que daña a los dispositivos y afecta el funcionamiento de los circuitos, el recubrimiento de los componentes se realizó con material aislante.

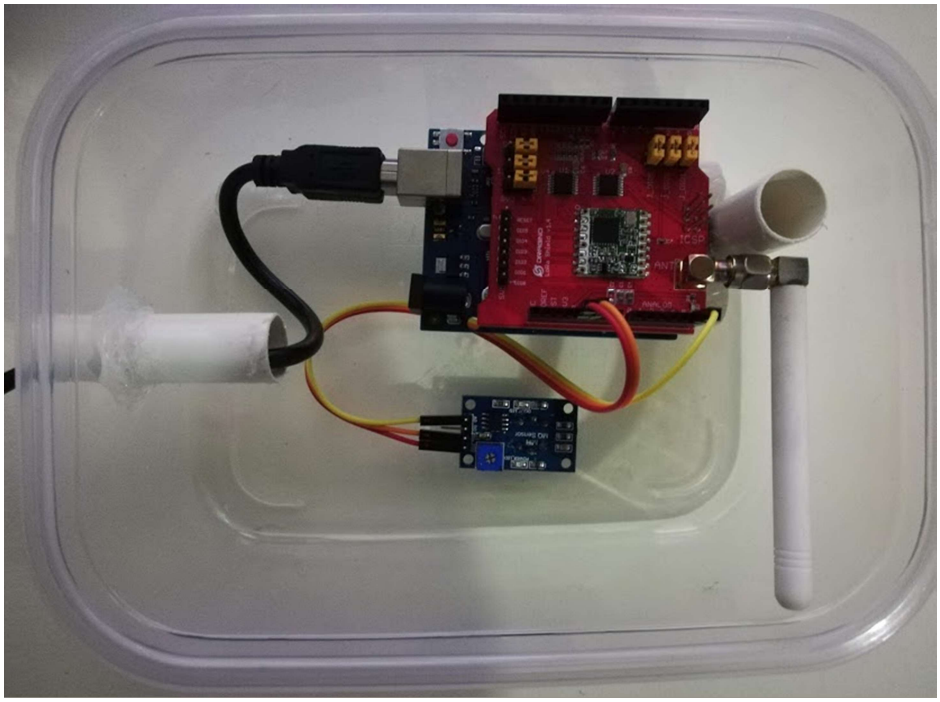


Figura 2.9.- Vista superior del sensor, LoRa Shield, Arduino, interior del envase plástico

Para que los sensores capten los datos correctamente se dejó un orifico en parte inferior del envase de plástico y se le protegió para poder mantener el correcto funcionamiento del sensor mq-7, ver Figura 2.10.

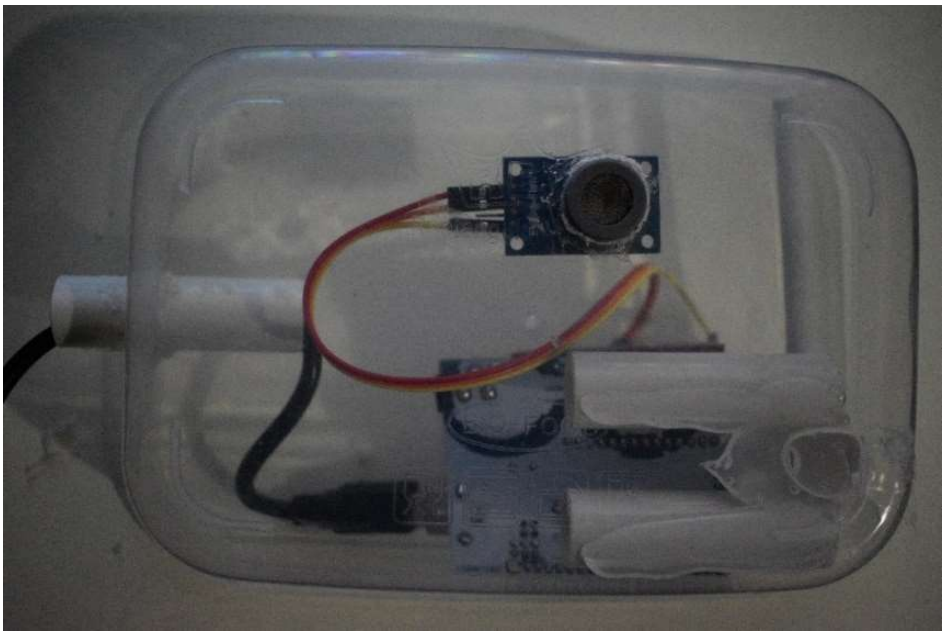


Figura 2.10.- Vista inferior dispositivo, LoRa Shield, Arduino exterior envase plástico

2.3.4 Fase de Implementación

Se Integro los dispositivos a la red, en base a lo desarrollado en la fase de diseño, con la configuración de los sensores, en caso de falla se volvería al paso de diseño.

2.3.4.1 Sprint 1

Sprint 1 Historia de Usuario A1

El Sprint 1 se tiene la siguiente historia de usuario:

Tabla 2.15.- Sprint 1 Historia de Usuario A1

Historia de Usuario	
Número: A1	**Usuario:** Sistema
Nombre historia: Configuración Arduino con Dragino LoRa Shield y Node-RED	
Prioridad en negocio: Alta	
Puntos estimados: 8	**Sprint:** 1
Asignado a: Jorge Miño	
Descripción: Como sistema, necesito configurar el Arduino que me permita enviar los datos captados por el sensor MQ-7, a través del Dragino LoRa Shield al Gateway que sean filtrados por Node-RED y la información sea enviada a CloudMQTT.	
Criterio de aceptación: El Arduino debe conectarse al Gateway y enviar los datos del sensor MQ-7 cada minuto. El pin del sensor MQ-7 debe estar calibrado en función de (Ananto, Ahmed, & Ferdousi). Node-RED debe filtrar los dispositivos únicamente del proyecto. Node-RED debe reconstruir el payload enviado por los dispositivos del proyecto. Node-RED debe conectarse al servicio CloudMQTT y enviar los datos captados por los dispositivos.	

Sprint 1 Historia de Usuario A2

El Sprint 1 se tiene la siguiente historia de usuario:

Tabla 2.16.- Sprint 1 Historia de Usuario A2

Historia de Usuario	
Número: A2	**Usuario:** Sistema
Nombre historia: Gestión de conexiones del backend	
Prioridad en negocio: Alta	
Puntos estimados: 8	**Sprint:** 1
Asignado a: Jorge Miño	
Descripción: Como sistema, necesito un backend que me permita gestionar las conexiones con CloudMQTT y la base de datos.	
Criterio de aceptación: El backend debe conectarse con CloudMQTT y recibir los datos cada vez que son enviados por los sensores. El backend debe trasformar los datos recibidos de acuerdo con el modelo de base de datos MongoDB. El backend debe validar y transformar al uso horario admisible para el frontend.	

Sprint 1 Historia de Usuario A3

El Sprint 1 se tiene la siguiente historia de usuario:

Tabla 2.17.- Sprint 1 Historia de Usuario A3

Historia de Usuario	
Número: A3	**Usuario:** Sistema
Nombre historia: Gestión de datos del backend	
Prioridad en negocio: Alta	
Puntos estimados: 8	**Sprint:** 1
Asignado a: Jorge Miño	
Descripción: Como sistema, necesito un backend que me permita gestionar los datos categorizándolos y guardándolos según el modelo de la base de datos, y generar rutas de consulta para el frontend.	
Criterio de aceptación: El backend debe categorizar los datos según el estándar de la calidad del Aire. El backend debe guardar los datos en la base de datos. El backend debe responder las consultas que haga el frontend.	

2.3.4.2 Sprint 2

Sprint 2 Historia de Usuario B1

El Sprint 2 se tiene la siguiente historia de usuario:

Tabla 2.18.- Sprint 2 Historia de Usuario B1

Historia de Usuario	
Número: B1	**Usuario:** Usuario
Nombre historia: Desarrollo interfaz principal de la aplicación móvil	
Prioridad en negocio: Alta	
Puntos estimados: 8	**Sprint: 2**
Asignado a: Jorge Miño	
Descripción: Como usuario, necesito una interfaz principal que me permita visualizar los datos de los sensores del proyecto representados en forma de gráfico de puntos y mostrar registros de la contaminación del aire en forma de lista y ultimo registro.	
Criterio de aceptación: La aplicación debe graficar los valores de los últimos 40 minutos La aplicación debe mostrar valores del ultimo registro en formato ppm y mg/m^3 La aplicación debe mostrar el valor y la categoría de la calidad en un Widget que cambia según en la categoría de la calidad del aire. La aplicación debe mostrar una lista de valores con el formato ppm, mg/m^3, AQI y Hora de los últimos 40 minutos.	

Interfaz desarrollada en el Sprint

Figura 2.11.- Interfaz Principal Medición Contaminación

2.3.4.3 Sprint 3

Sprint 3 Historia de Usuario C1

El Sprint 3 se tiene la siguiente historia de usuario:

Tabla 2.19.- Sprint 3 Historia de Usuario C1

Historia de Usuario	
Número: C1	**Usuario:** Usuario
Nombre historia: Desarrollo interfaz datos diarios de la aplicación móvil	
Prioridad en negocio: Media	
Puntos estimados: 5	**Sprint:** 3
Asignado a: Jorge Miño	
Descripción: Como usuario, necesito una interfaz que me permita visualizar los datos de los sensores del proyecto en forma de gráfica de puntos y mostrar los registros de picos de medición de un día determinado.	
Criterio de aceptación: La aplicación debe graficar los valores de un día especifico. La aplicación debe mostrar la hora de los valores mínimo y máximo expresados en partes por millón. La aplicación debe mostrar la fecha y el día que se está observando.	

Interfaz desarrollada en el Sprint

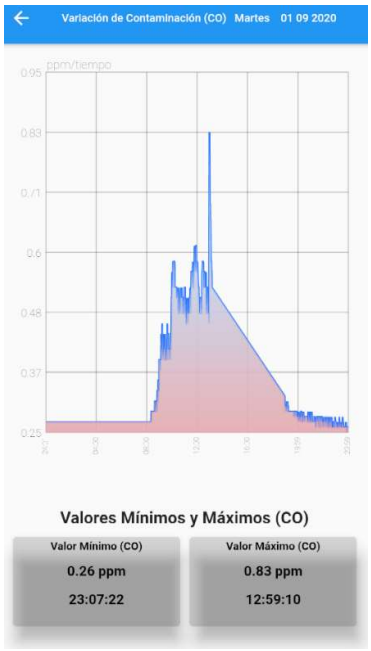


Figura 2.12.- Widget Variación de Contaminación de un día determinado

2.3.4.4 Sprint 4

Sprint 4 Historia de Usuario D1

El Sprint 4 se tiene la siguiente historia de usuario:

Tabla 2.20.- Sprint 4 Historia de Usuario D1

Historia de Usuario	
Número: D1	**Usuario:** Sistema
Nombre historia: Corrección bugs backend, frontend	
Prioridad en negocio: Media	
Puntos estimados: 5	**Sprint: 4**
Asignado a: Jorge Miño	
Descripción: Como aplicación, necesito corregir los bugs de manejo de datos en el backend y frontend, para la publicación del backend en la nube y compilado de la aplicación para Android.	
Criterio de aceptación: Los datos estén formateados con dos cifras significativas a que sean guardados en la base de datos. Las rutas en el backend para la consulta en el backend este correctas. La aplicación tenga acceso a internet.	

2.3.4.5 Release Burndown

Durante el desarrollo del proyecto podemos evidenciar en la Figura 2.13 el cumplimiento de las historias de usuarios en cada Sprint, los Sprint 0 hasta Sprint 4 denota el cumplimiento de satisfactorio de las historias de usuario en los tiempos establecidos.

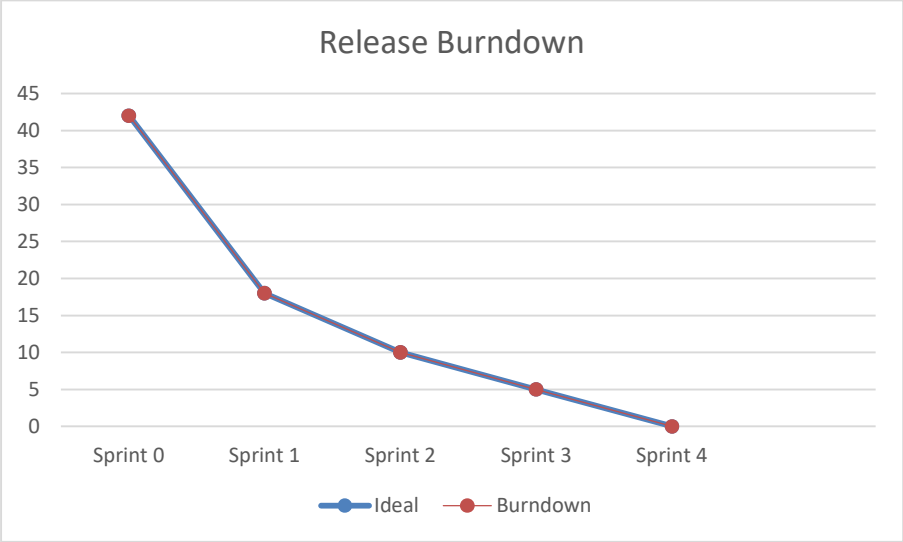


Figura 2.13.- Release Burndown

2.3.5 Fase de Operación

Los cambios establecidos en esta fase permitieron tener una conexión estable de más de cinco meses durante las pruebas de estabilidad de los dispositivos, con envío de datos captados por los sensores en intervalos de 1 minuto y en caso de que exista problemas de conexión el Arduino pueda conectarse nuevamente.

Se hizo pruebas y existió fallos de acceso de los dispositivos dentro de la red LoRaWAN se modificó el cambio de pines con una conexión exitosa (DRAGINO, 2018):

```
lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2, 6, 7},
};
```

En las especificaciones de github LMIC informa que existe problemas con Downlink en OTAA por lo que se debe establecer el error de precisión del reloj en diez por ciento (Kooijman, 2020).

```
LMIC_setClockError(MAX_CLOCK_ERROR * 10 / 100);
```

En caso de que exista errores de conexión de cualquier tipo se optó por hacer un join nuevo a la red con la función LMIC_startJoining() establecido en la librería LMIC.

2.3.5.1 Log de conexión

Se verifico con los cambios realizado en la programación del Arduino que existe un envió periódico de datos por parte de los sensores en intervalos de un minuto ver Figura 2.15 de los sensores LabFIS y FIS respectivamente ver Figura 2.14.

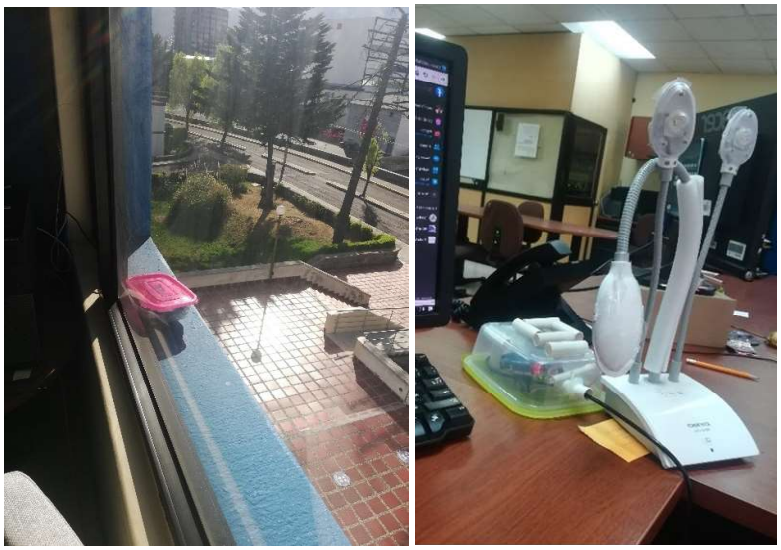


Figura 2.14.- Sensores FIS y LabFIS

Device EUI	Freq	Datarate	SNR	RSSI	Size	FCnt	Type	Tx/Rx Time
89-ad-02-c8	923.300	SF7BW500	-	-	69	00000020	DnCnf	5 minutes ago
89-ad-02-c8	903.000	SF8BW500	10	-85	18	00000109	UpUnc	5 minutes ago
56-76-0a-8c	923.300	SF7BW500	-	-	144	0000005C	DnCnf	6 minutes ago
56-76-0a-8c	903.000	SF8BW500	10	-79	18	0000030E	UpUnc	6 minutes ago
89-ad-02-c8	923.300	SF7BW500	-	-	69	0000001F	DnCnf	6 minutes ago
89-ad-02-c8	903.000	SF8BW500	10	-84	18	00000101	UpUnc	6 minutes ago
56-76-0a-8c	923.300	SF7BW500	-	-	144	0000005B	DnCnf	6 minutes ago
56-76-0a-8c	903.000	SF8BW500	9	-79	18	00000306	UpUnc	6 minutes ago
89-ad-02-c8	923.300	SF7BW500	-	-	64	0000001E	DnCnf	7 minutes ago
89-ad-02-c8	903.000	SF8BW500	9	-83	18	000000F9	UpUnc	7 minutes ago
56-76-0a-8c	923.300	SF7BW500	-	-	139	0000005A	DnCnf	7 minutes ago
56-76-0a-8c	903.000	SF8BW500	10	-78	18	000002FE	UpUnc	7 minutes ago
89-ad-02-c8	923.300	SF7BW500	-	-	64	0000001D	DnCnf	8 minutes ago
89-ad-02-c8	903.000	SF8BW500	9	-82	18	000000F1	UpUnc	8 minutes ago
56-76-0a-8c	923.300	SF7BW500	-	-	139	00000059	DnCnf	8 minutes ago
56-76-0a-8c	903.000	SF8BW500	10	-77	18	000002F6	UpUnc	8 minutes ago
89-ad-02-c8	923.300	SF7BW500	-	-	64	0000001C	DnCnf	9 minutes ago
89-ad-02-c8	903.000	SF8BW500	9	-86	18	000000E9	UpUnc	9 minutes ago
56-76-0a-8c	923.300	SF7BW500	-	-	139	00000058	DnCnf	9 minutes ago
56-76-0a-8c	903.000	SF8BW500	9	-79	18	000002EE	UpUnc	9 minutes ago
89-ad-02-c8	923.300	SF7BW500	-	-	64	0000001B	DnCnf	10 minutes ago
89-ad-02-c8	903.000	SF8BW500	9	-84	18	000000E1	UpUnc	10 minutes ago

Figura 2.15.- Log del Gateway Smart Lab

2.3.6 Fase Optimización

Se realizó un depurado de variables eliminando los mensajes de impresión por pantalla del proyecto original de LMIC. El tamaño de payload se lo optimizó a 17 bytes cambiando el tipo de variable de String a byte resolviendo el problema de manejo de datos. Los cambios realizados permitieron reducir el tamaño de espacio de almacenamiento del programa en la ROM del 72% al 62% y una disminución de uso de la memoria dinámica al 50%.

CAPITULO III

3 RESULTADOS Y DISCUSIÓN

Durante el desarrollo del proyecto se buscó encontrar la mejor solución de bajo costo en hardware como en software que cumpliera con el objetivo del proyecto y a la vez sea una solución real a las alternativas implementadas en algunos estudios enfocados en medir la calidad del aire como: El trabajo de (Devarakonda, y otros, 2013) implementa un prototipo con tecnología 3G/GPRS con un costo por nodo de $700 dólares y con un flujo de datos de 1600 bytes/minuto, la solución del presente proyecto implementa un flujo de datos de 17 bytes/minuto por lo que el costo computacional es radicalmente menor y el costo por unidad es de $51.1 dólares.

El trabajo (Liu, Xia, & Zhao, 2016) propuso una solución para redes LoRa para visualizar los datos captados por los sensores utilizando el software LabVIEW el usuario no tiene automatizado el proceso de visualización de datos, el presente proyecto posee una interfaz móvil desarrollada en Flutter con la posibilidad de ver en tiempo real la información procesada por el backend que fue desarrollada con JavaScript, que son tecnologías muy utilizadas actualmente.

En el artículo (Fuertes , y otros, 2015) presenta una solución implementada en Xbee con frecuencia de 2.4 GHz esta tecnología posee problemas de interferencia y con un alcance en zona urbana de 90 metros, el presente proyecto fue desarrollado con tecnología LoRa solucionando los problemas de interferencia y con un alcance obtenido en pruebas en el perímetro urbano de 1.1816 kilómetros.

Respecto a los costos para medir la calidad del aire en (Fuertes , y otros, 2015) fue de $200 dólares, en comparación al proyecto actual realizado fue de $51.1 dólares por nodo implementado ver Tabla 3.1.

Tabla 3.1.- Detalle de costo componentes dispositivos finales proyecto

Cantidad	Descripción	Unidad (dólares)	Total(dólares)
2	Sensor MQ-7	5	10
2	Arduino UNO	12	24
2	Dragino LoRa Shield	26	52
2	Recipiente plástico con tapa	3	6
2	Cable USB	5	10
2	Tubo plástico	0.10	0.20
Total		**51.1**	**102.2**

Los sensores implementados en el presente proyecto fueron configurados para que registren mediciones cada minuto intervalo que permitió tener una mayor tasa de datos a los recolectados por REMMAQ que son mediciones en intervalos de 20 minutos.

La normativa de calidad del aire establecida en el 2011 por la Secretaria de Ambiente (Secretaría de Ambiente, 2016) estableció estándares y lineamientos sobre el nivel de emisión de gases en el ambiente, pero no determinó los valores planteados por las Instituciones internacionales (The World Air, 2019) y (Enviromental Protetion Agency, 2020) sobre los rangos de la calidad del aire. Por lo que para ser implementados los rangos de los valores en el backend se estableció la normativa internacional de (The World Air, 2019).

3.1 Análisis de la Arquitectura de Red

Luego de analizar los trabajos de (Fuertes , y otros, 2015) y (Devarakonda, y otros, 2013) se puede determinar que el presente proyecto soluciona los problemas de alcance de red en zona urbana obteniendo los siguientes resultados:

Se realizó pruebas de alcance de conexión entre el Gateway y el diseño de sensor del proyecto, estas pruebas se las realizo implementando una Powerbank para proveer energía al diseño del sensor y monitorizando la conexión desde interfaz del Gateway.

Prueba 1 la Figura 3.1 indica la posición donde se realizó la prueba de alcance de conexión, del dispositivo de prueba ubicado intersección de la 12 de Octubre y Coruña en la Plaza Artigas y el Gateway del Smart Lab en la Facultad de Ingeniería de Sistemas en la Escuela Politécnica Nacional, teniendo como resultado 0.9747 km y un RSSI promedio de -99.1 dBm.

Figura 3.1.- Prueba 1, referencia de la distancia entre el Gateway Smart-Lab y el dispositivo tesis de prueba

Prueba 2 la Figura 3.2 indica en el mapa la posición donde se realizó la prueba de alcance de conexión, con el dispositivo de prueba ubicado intersección de Av. 6 de Diciembre y San Ignacio y el Gateway del Smart-Lab en la Facultad de Ingeniería de Sistemas en la

Escuela Politécnica Nacional, teniendo como resultado 1.1816 km y RSSI promedio de 100.2 dBm.

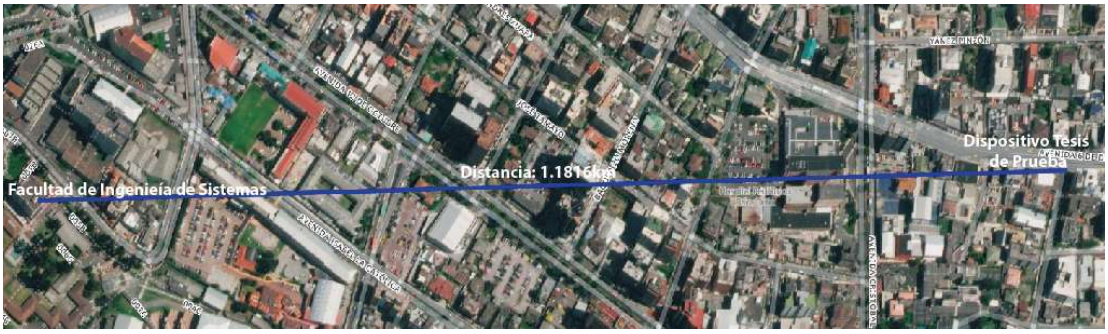


Figura 3.2.- Prueba 2, referencia de la distancia entre el Gateway Smart-Lab y el dispositivo tesis de prueba

Distancia de cobertura

En la figura 3.3 se pudo determinar la cobertura real del Gateway con el sensor de prueba y se determinó un área de posible cobertura del Gateway del laboratorio Smart-Lab aplicando la siguiente formula:

$$A = \pi r^2$$

Donde:

A: Área de cobertura de red (Km^2)

r: radio de prueba de enlace (Km)

Se obtiene los siguientes valores:

$$A = \pi(1.1816)^2 = 4.38\ Km^2$$

Por lo tanto el área de cobertura teórica en función de la pruebas de conexión entre el Gateway y los dispositivos finales es 4.38 Km^2.

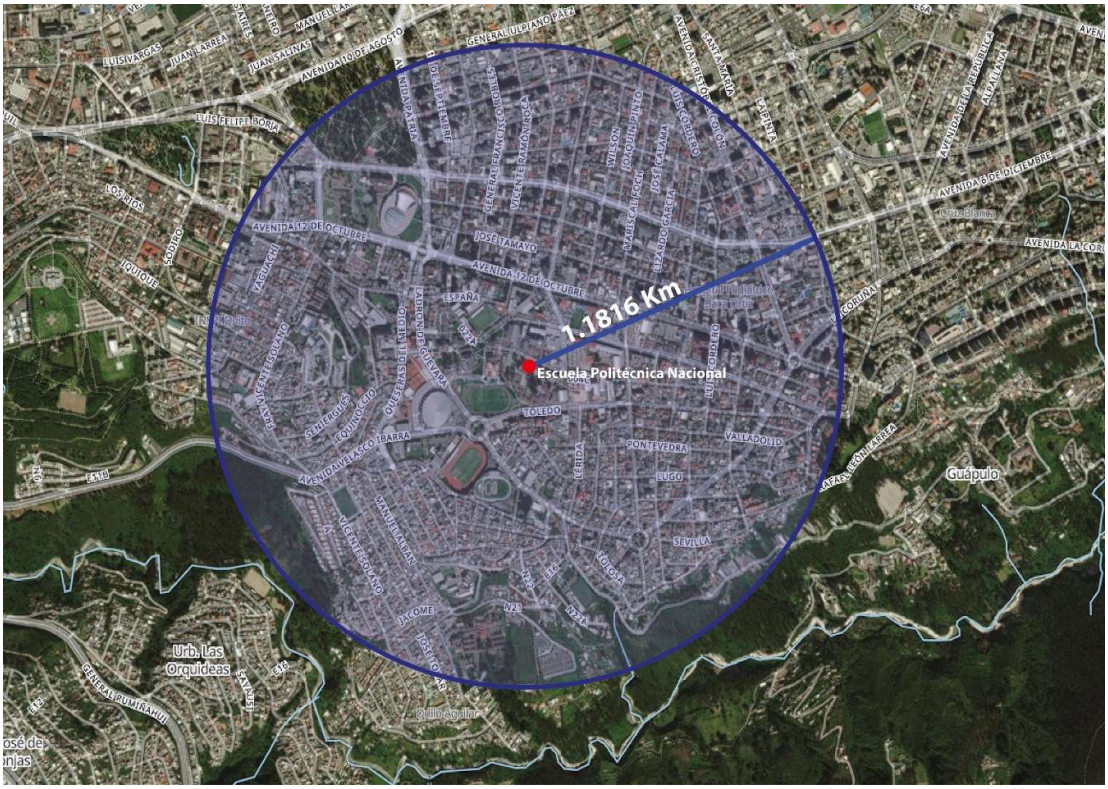


Figura 3.3.- Área de cobertura en función prueba de alcance realizada

La Arquitectura de red en el que está implementado el proyecto es de tipo privada en contraste al trabajo de (Devarakonda, y otros, 2013) que opera en una red celular 3G que es de pago, el uso de una red privada en el presente proyecto permite que una sola entidad en este caso de Smart-Lab de la Escuela Politécnica Nacional, permita administrar que los mensajes sean enviados directamente desde el nodo final a la aplicación que gestiona los datos.

3.2 Análisis de Hardware

LoRa con los módulos Lora Shield posibilita aumentar la cantidad de N dispositivos según se requiera y la infraestructura soporte, para ser utilizada en futuros proyectos, en comparación de las tecnologías 3G/GPRS (Devarakonda, y otros, 2013) y Xbee(Fuertes , y otros, 2015) que demandan un gasto económico de más de cuádruple al momento de implementar un nuevo nodo final.

Para la implementación del proyecto se diseñó dos dispositivos para monitorear la contaminación en el exterior e interior de las instalaciones de la Escuela Politécnica Nacional y un dispositivo de prueba de alcance de conexión. En base a los valores de calibración (Ananto, Ahmed, & Ferdousi) se realizó pruebas en instalaciones externas al

Campus Politécnico, obteniendo resultados satisfactorios en función a lo indicado en el estudio del sensor MQ-7.

El diseño en los dispositivos para medir la contaminación del aire permite ser implementados en exteriores, evitando tener datos erróneos de mediciones por desperfectos en caso de que exista lluvia, el calor que es uno de los factores que deteriora la vida útil de los componentes electrónicos aplicados en el proyecto y el viento trasporta partículas, por lo que se aisló los componentes electrónicos. Es una solución eficiente y económica en comparación a las planteadas por (Devarakonda, y otros, 2013) el problema de protección de los componentes electrónicos externos a lo planteado en el diseño de (Fuertes , y otros, 2015).

3.3 Análisis de Sistema de Software

Los datos fueron filtrados desde su recepción en el Gateway con Node-RED, utilizando una función con un condicional que analice el payload y en caso de que pertenezca a los sensores del proyecto construya un JSON en función al modelo implementado en la base de datos y envié a CloudMQTT, esto permitió tener datos únicamente de los sensores del proyecto y evitando tener que filtrarlos en el backend ahorrando un costo computacional innecesario.

Los backend desarrollados en los trabajos de (Devarakonda, y otros, 2013), (Fuertes , y otros, 2015) y (Liu, Xia, & Zhao, 2016) presentan limitantes en la diversidad de plataformas para ser ejecutados, este trabajo lo resuelve utilizando NodeJS, posibilitando la implementación en cualquier plataforma compatible, además de ser escalable y con la posibilidad de ser migrado a otra plataforma que soporte JavaScript.

3.4 Resultados

Durante el desarrollo del proyecto se tomó un total de doscientos treinta y siete mil seiscientos treinta (237730) datos captados por los sensores en aproximadamente 5 meses desde el 7 de junio del 2020 hasta 27 de octubre del 2020. El peso de la base de datos es aproximadamente de 52.1 megabytes, por lo que la elección de la utilización de MongoDB para el proyecto fue eficiente por el reducido consumo de espacio en la nube.

El consumo de memoria RAM de la aplicación desarrollada para dispositivos Android fue de 200 kilobytes aproximadamente (Figura 3.4), teniendo una aplicación fluida y con poco uso de recursos.

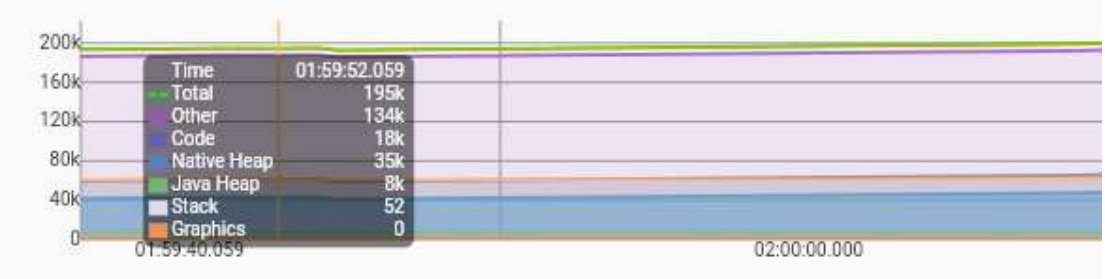


Figura 3.4.- Memoria ocupada por la aplicación

El tiempo promedio de respuesta del backend a la aplicación móvil fue menor a 300 ms, en las diferentes rutas de consulta, ver Tabla 3.2:

Tabla 3.2.- Tabla tiempo de respuesta rutas backend en Heroku

Ruta de consulta	Descripción	Tiempo (ms)
/sensor/datosmin/	Mínimo Absoluto	17
/sensor/datosmax/	Máximo Absoluto	12
/sensor/dashboard/	Registros últimos 40 minutos	89
/sensor/dataday	Valores diarios	295
/sensor/datadayquality/	Calidad del aire diarios	212

3.4.1 Datos Sensores FIS

La distribución de dispositivos se realizó de la siguiente manera, sensor FIS ubicado en los exteriores de la Facultad de Ingeniería de Sistemas y el sensor LabFIS ubicado en el laboratorio Alfa del tercer piso. Por lo que se hizo una comparativa entre la distancia al gateway y RSSI de los sensores del proyecto y el de prueba tomando en cuenta que el factor de dispersión fue de sf8/500 kHz para los sensores dentro del perímetro de la universidad ver Tabla 3.3. Los valores de RSSI de los sensores indican que a mayor proximidad al Gateway el valor se aproxima a cero y la velocidad de envió de información aumenta.

Tabla 3.3.- Alcance de red del Gateway Smart-Lab

Sensor	Distancia (m)	RSSI (dBm)
FIS	50	-80
LabFIS	52	-80
Sensor prueba	1181,6	-100.2
	974,7	-99.1

El COE Nacional informó que el estado de excepción finalizó el 13 de septiembre del 2020 representado en la Figura 3.5, con color naranja, dejando a disposición de los municipios,

la decisión de circulación vehicular. El Distrito Metropolitano de Quito para el mes de septiembre determinó restricción vehicular en función de las último digito de las placas.

3.4.1.1 Sensor FIS

Los valores de la Figura 3.5 muestra un incremento en los valores de monóxido de carbono en el intervalo del 27 de agosto al 27 de octubre del 2020, periodo de la pandemia donde se realizó el presente estudio. Los valores representados en la Figura 3.5, muestran un aumento en el intervalo de septiembre en los valores máximos, mínimos y promedio diarios de monóxido de carbono; en contraste para el mes de octubre estos valores aumentan sustancialmente. El mínimo absoluto obtenido de la muestra se registró el 28 de agosto y el máximo absoluto el 26 de octubre. Esto denota que a mayor carga vehicular la calidad en aire de Quito disminuye, teniendo como consecuencia la afección de salud en personas más vulnerables por problemas respiratorios.

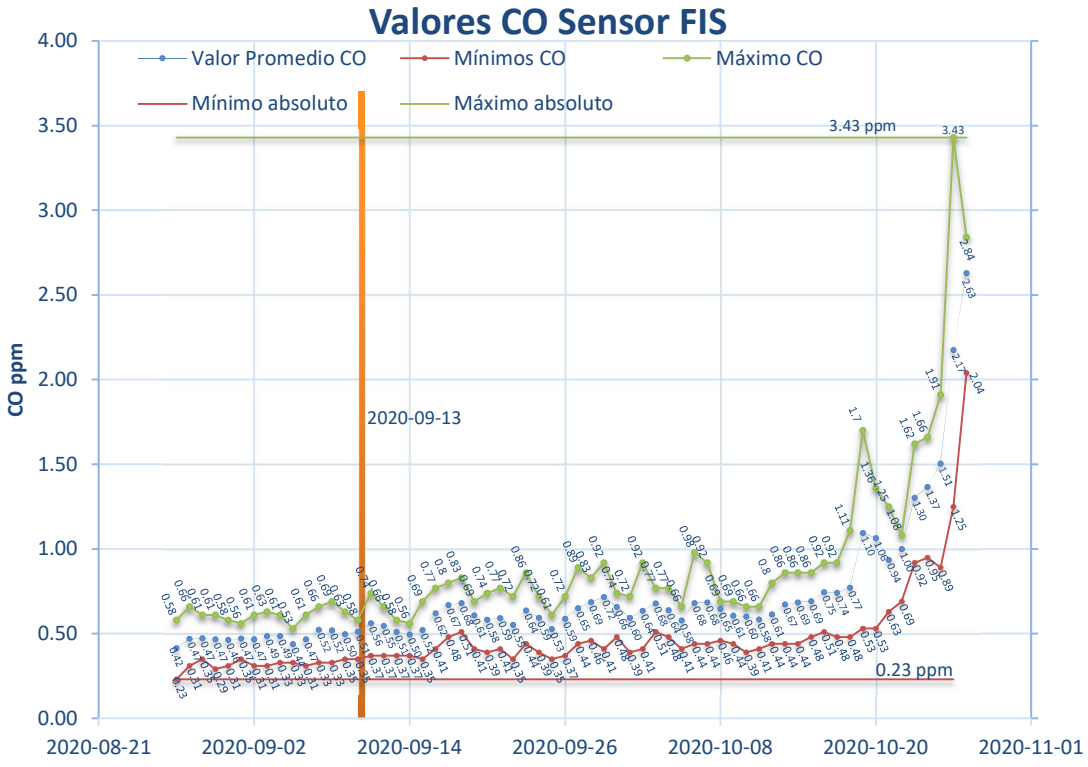


Figura 3.5.- Datos captados sensor FIS del 27-08-2020 hasta 27-10-2020

3.4.1.2 Sensor LabFIS

El sensor LabFIS implementado en laboratorio Alfa del tercer piso de la Facultad de Ingeniería de Sistemas, muestra (ver Figura 3.6) un incremento en los valores promedios captados aproximado de 0.5 ppm durante el estudio realizado, los picos altos reflejan

mayor actividad en el laboratorio, ya sea de presencia de personas, aumento de emisión de gases de los equipos de computación o el sistema de aire acondicionado que es el único medio externo que permite el ingreso de aire.

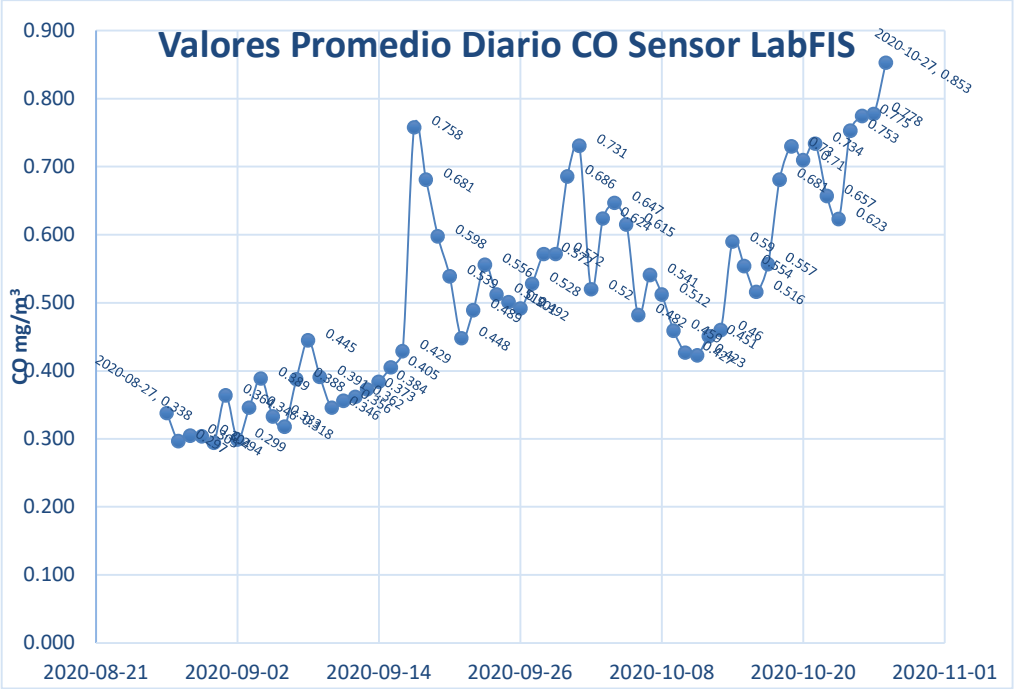


Figura 3.6.- Datos captados sensor LabFIS del 27-08-2020 hasta 27-10-2020

3.4.2 Comparativa Sensor FIS vs LabFIS

Los valores observados en las mediciones externas sensor FIS versus internas sensor LabFIS (Tablas 3.4) reflejan que la calidad de aire en el laboratorio Alfa es mejor a pesar que tiene elementos como aire acondicionado, computadores, access point, servidores funcionando las 24 horas y la presencia de una persona del personal de LabFIS acudiendo en promedio al laboratorio de cinco horas por día y un día por semana desde el mes de agosto hasta fines de octubre, las mediciones entre sensores muestran picos de mínimos absolutos con pequeñas diferencias entre sensores, existiendo un diferencial mayor en los valores máximos absolutos captados, debido a que el laboratorio Alfa permanece en condiciones constantes en comparación del sensor FIS que varía por la contaminación vehicular.

Tabla 3.4.- Tabla obtenida Anexo VIII, sección Tabla Datos Sensor FIS y Tabla Datos Sensor LabFIS

Detalle	**Mínimo Absoluto** (mg/m^3)	**Máximo Absoluto** (mg/m^3)	**Media** (mg/m^3)
FIS	0,42	2,63	0,71

LabFIS	0,29	0,85	0.51

3.4.3 Comparativa Sensor FIS vs. REMMAQ

Se realizó un análisis de los datos captados por REMMAQ en las zonas: Belisario, Carapungo, Centro, Cotocollao y el Camal, obteniendo los valores promedios comprendidos desde el 1 de febrero al 13 de marzo, lapso previo a la pandemia, y comparándole con los valores promedios captados del Sensor FIS desde el 27 de agosto al 27 de Octubre ver Tabla 3.5:

Tabla 3.5.- Tabla obtenida Anexo VIII, sección Tabla REMMAQ y Tabla Datos Sensor FIS

	BELISARIO	**CARAPUNGO**	**CENTRO**	**COTOCOLLAO**	**El CAMAL**	**Sensor FIS**
Detalle	(mg/m^3)	(mg/m^3)	(mg/m^3)	(mg/m^3)	(mg/m^3)	(mg/m^3)
Promedio	0,69	0,54	0,75	0,67	0,88	0,71
Máximos Absoluto	2,6	2,42	3,05	3,419	3,43	2,63
Mínimos Absoluto	0,17	0,04	0,11	0.078	0,23	0,42

Los valores de la Tabla 3.4 describen, que los valores promedios previos a la pandemia en varios puntos de Quito son semejantes a los captados por el sensor, eso demuestra que la contaminación del aire ha vuelto a los parámetros habituales de Quito.

CAPITULO IV

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La selección de LoRa y el protocolo LoRaWAN para la implementación del sistema propuesto cumplió las expectativas de cobertura y tamaño del paquete.
- En cuanto a cobertura de conexión, se logró cubrir el perímetro de la Escuela Politécnica Nacional para los dispositivos finales, dado que el alcance en pruebas realizadas del gateway fue de 1,1816 km, superando el área de estudio.
- El gateway LoRa funciona en una banda no comercial de frecuencia libre ISM, empleada en el área industrial, científica y médica, sin la necesidad de requerir una licencia para su uso, permitiendo así el uso de los sensores sin la interferencia teórica de otras frecuencias.
- El protocolo LoRaWAN permitió enviar paquetes de tamaños menores a 53 bytes permitiendo un mínimo consumo de energía.
- Poseer una red privada para el desarrollo del proyecto, permitió administrar los datos y los dispositivos conectados, sin tener la limitante de conectarse a un servicio de la nube o servicio de cobertura celular.
- Al utilizar el Dragino LoRa Shield basado en el chip SX1276 con la librería LMIC (Kooijman, 2020) , inicialmente dificulto la continuidad de envío de datos al Gateway, ya que alcanzaba el límite de tamaño de payload de 53 bytes permitido por la librería. La solución fue implementar un método para reestablecer el tamaño del payload, teniendo estabilidad de recopilación de datos por parte de los dispositivos finales al Gateway por más de 5 meses.
- Los valores captados entre el 27 de agosto y el 27 de octubre por el sensor FIS, ubicado en el exterior de la Facultad de Ingeniería de Sistemas en contraste a los valores captados por los sensores de la REMMAQ previos a la pandemia del 02 de febrero al 13 de marzo son semejantes (ver Tabla 3.5). Los valores promedios, máximos y mínimos absolutos analizados son aproximados entre el sensor Fis y REMMAQ, esto implica que los índices de contaminación del aire en Quito se categorizan como “Bueno” en función de (The World Air, 2019). Se observa a través de esta medición que el diseño del dispositivo del proyecto cumple su objetivo.
- El masificar, diseñar e implementar prototipos desarrollados en el proyecto para medir la contaminación del aire, será una alternativa a los sistemas tradicionales como REMMAQ. Sin embargo aún existe trabajo e investigación por realizar para

integrar un sistema con tecnología LoRa de sensores de medición de la contaminación del aire, que abarque la ciudad de Quito, debido a que aún no existe infraestructura de red pública y privada compatible para LoRaWAN.

- El uso de tecnologías libres facilitó la implementación del sistema de IoT propuesto. También permite tener sistemas escalables con conexión a la nube para procesamiento de datos y generar datos estadísticos.
- El presente estudio sienta las bases para la generación de otros proyectos de IoT basado en LPWAN, específicamente de LoRaWAN.

4.2 Recomendaciones

- Una de las principales limitantes al momento de implementar los dispositivos, fue la fuente de alimentación propia con una batería. Para la implementación real, se recomienda realizar un estudio de la capacidad de las baterías de los dispositivos basado en la cantidad de datos enviados, en caso de no poder tener una fuente de alimentación de energía continua.
- Se recomienda implementar los sensores en dispositivos que consumen menor cantidad de energía que un Arduino Uno, como son Arduino Nano o dispositivos ESP32 que tiene implementado el módulo de LoRa.
- Para futuros prototipos, se recomienda el uso de una impresora 3D para obtener un diseño más compacto y resistente a los efectos ambientales.
- Se recomienda implementar un mayor número de dispositivos de medición dentro del Campus Politécnico para conocer el estado de contaminación del aire de una manera más detallada.
- Se recomienda aumentar el número de sensores que capten otros tipos de gases en cada dispositivo que permita diversificar la lectura de otros contaminantes.

REFERENCIAS BIBLIOGRÁFICAS

- ALFAIOT. (2019). *ALFAIOT*. Obtenido de https://alfaiot.com/blog/ultimas-noticias-2/post/tecnologia-lpwan-vs-5g-para-internet-de-las-cosas-11
- Ananto, S. E., Ahmed, M. N., & Ferdousi, J. A. (s.f.). *Development of Carbon Monoxide detecting device using MQ-7 sensor along with its statistical analysis.* Obtenido de http://dspace.bracu.ac.bd/xmlui/handle/10361/3590
- *Arduino*. (2020). Obtenido de https://www.arduino.cc/
- Arduino. (2 de Mayo de 2020). *Arduino*. Obtenido de https://store.arduino.cc/usa/arduino-uno-rev3
- Arduino. (2020). *Store Arduino*. Obtenido de https://store.arduino.cc/usa/arduino-nano
- Balaguera, Y. D. (2015). Guía metodológica ágil, para el desarrollo de aplicaciones móviles “AEGIS-MD”. *Revista de investigaciones UNAD*, 101-103.
- Capella Hernández, J. V. (2011). Redes inalámbricas de sensores: Una nueva arquitectura eficiente y robusta basada en jerarquía dinámica de grupos. *Editorial Universitat Politècnica de València* .
- Chirpstack. (2020). *Chirpstack.* Obtenido de https://www.chirpstack.io/application-server/use/device-profiles/
- CISCO. (2014). *CISCO.* Obtenido de https://www.cisco.com/en/US/services/ps2961/ps5868/CE_Services_Overview_EXTERNAL_121208.pdf
- DEEPDATA CONSULTING. (14 de Febrero de 2020). *¿QUÉ ES LA RED LORA?* Obtenido de https://deepdata.es/red-lora/
- Deloitte. (2020). *¿Qué es la Industria 4.0?* Obtenido de https://www2.deloitte.com/es/es/pages/manufacturing/articles/que-es-la-industria-4.0.html
- Devarakonda, S., Parveen , S., Liu, H., Liu, R., Iftode, L., & Nath, B. (2013). Real-time air quality monitoring through mobile sensing in metropolitan areas. *the 2nd ACM SIGKDD International Workshop* (pág. 1). Chicago, Illinois: ACM Press.
- Dragino. (19 de Octubre de 2017). *Dragino*. Obtenido de http://wiki.dragino.com/index.php?title=File:LoRa_Shield_Pin_Mapping.png
- DRAGINO. (2018). *GiTHUB Dragino*. Obtenido de https://github.com/dragino
- DRAGINO. (01 de Agosto de 2020). *DRAGINO.* Obtenido de https://www.dragino.com/products/lora/item/102-lora-shield.html

- DSET ENERGY. (Julio de 2019). Obtenido de Estructura de red Sigfox: http://www.dset-energy.com/wp-content/uploads/2019/06/Estructura-red-Sigfox-768x526.png
- Enviromental Protetion Agency. (27 de October de 2020). *U.S. Environmental Protection Agency*. Obtenido de https://www.epa.gov/
- Filip, B., Flaviu M., F.-I., Attila, S., Sorin , M., Doru, V., & Petru, A. (2020). LoRaWAN Based Real-Time Air Quality Monitoring System . *IEEE 18th World Symposium on Applied Machine Intelligence and Informatics*, 69-72.
- Flutter. (2020). *Flutter*. Obtenido de https://esflutter.dev/
- Fuertes , W., Carrera, D., Villacís, C., Toulkeridis, Galárraga, Torres, E., & Aules. (2015). Distributed System as Internet of Things for a New Low-Cost, Air Pollution Wireless Monitoring on Real Time. *2015 IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, (págs. 58-67). Quito.
- Fundación OpenJS. (2020). *Express*. Obtenido de https://expressjs.com/es/
- Gasca Mantilla, M. C., Camargo Ariza, L. L., & Medina Delgado, B. (2014). Metodología para el desarrollo de aplicaciones móviles. *Tecnura*, 20-35.
- Hanwei Electronics CO .LTD. (2019). *HWSENSOR.* Obtenido de http://www.hwsensor.com
- Hofacker, A. (2008). *Rapid lean construction - quality rating model.* Manchester: s.n.
- IDC. (2018). *IDC*. Obtenido de BusinessWire: https://www.businesswire.com/news/home/20180618005142/en/IDC-Forecasts-Worldwide-Technology-Spending-Internet-Things
- Ingenu. (2020). *Ingenu Inc.* Obtenido de https://www.ingenu.com
- Instituto Nacional de Ecología. (2020). Obtenido de https://sinaica.inecc.gob.mx/archivo/guias/1-%20Principios%20de%20Medici%C3%B3n%20de%20la%20Calidad%20del%20Aire.pdf
- IoT World Online. (29 de Noviembre de 2017). Obtenido de Las grandes estadísticas del Internet de las Cosas (IoT): https://www.iotworldonline.es/las-grandes-estadisticas-del-internet-de-las-cosas-iot/
- IoT-Analytics. (25 de Enero de 2018). *New Research on 1,600 Enterprise IoT Projects: Upsurge in Smart City and Connected Building Related IoT Projects*. Obtenido de https://iot-analytics.com/global-overview-1600-enterprise-iot-projects/
- IoTforall. (3 de Agosto de 2020). *IoTforall*. Obtenido de Monitoreo de la contaminación del aire: https://www.iotforall.com/use-case/air-pollution-

monitoring#:~:text=Air%20pollution%20monitoring%20uses%20stationary,and%20make%20more%20informed%20decisions.

- Kooijman, M. (2020). *GITHUB.* Obtenido de https://github.com/matthijskooijman/arduino-lmic
- Koskela, L. (1992). *Application of the new production philosophy to construction.* Finland: VTT Building Technology.
- Liu, S., Xia, C., & Zhao, Z. (2016). A low-power real-time air quality monitoring system using LPWAN based on LoRa. *2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, (págs. 379-381).
- LOGITEK. (12 de Junio de 2020). Obtenido de Glosario de la tecnología LoRaWAN: https://www.m2mlogitek.com/glosario-de-la-tecnologia-lorawan/#:~:text=AppSKey%3A%20siglas%20de%20Application%20Session,traducir%20el%20Payload%20del%20mensaje.
- López Vicario, J., & Vilajosana Guillen, X. (1 de Enero de 2020). *OpenAccess*. Obtenido de http://openaccess.uoc.edu/webapps/o2/bitstream/10609/106369/7/shernandezc6TFM0120memoria.pdf
- LoRa Alliance. (2017). LoRaWAN™ 1.0.2 Regional Parameters. *LoRaWAN™ 1.0.2 Regional Parameters*, 13-17. Obtenido de https://lora-alliance.org/sites/default/files/2019-11/rp_2-1.0.0_final_release.pdf
- LoRa-Alliance. (2020). *LoRa-Alliance*. Obtenido de https://www.lora-alliance.org/
- MEDIUM. (02 de Octubre de 2017). Obtenido de MEDIUM: https://medium.com/beelan/haciendo-iot-con-lora-capitulo-2-tipos-y-clases-de-nodos-3856aba0e5be
- Microchip. (2018). *Microchip Technology Inc.* Obtenido de http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf
- Microsoft. (2020). *Visual Studio*. Obtenido de https://code.visualstudio.com/docs
- Ministerio de Ambiente. (2010). *Plan Nacional de la Calidad de Aire.* Obtenido de https://www.ambiente.gob.ec/wp-content/uploads/downloads/2012/10/libro-calidad-aire-1-final.pdf
- Modificado de Egli, P. R. (2015). *Indigoo*. Obtenido de http://indigoo.com/dox/itdp/12_MobileWireless/LPWAN.pdf?_ga=2.211
- Mongoose. (2019). *Mongoose*. Obtenido de https://mongoosejs.com

- MrElberni. (2020). *microcontroladores-mrelberni*. Obtenido de http://microcontroladores-mrelberni.com/usart-pic-comunicacion-serial/
- Multitech. (Septiembre de 2019). *Multi-Tech System, Inc*. Obtenido de https://www.multitech.com/documents/publications/data-sheets/86002221.pdf
- OpenJS Foundation. (2020). *OpenJS*. Obtenido de https://nodejs.org/es/about/
- Ordóñez Monfort, I. (2017). Estuddio de la arquitectura y el nivel ded desarrollo de la red LoRaWAN y de los dispositivos LoRa. 1-11.
- Organización Mundial de la Salud . (02 de Mayo de 2016). *Calidad del aire y salud.* Obtenido de https://www.who.int/es/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health
- Rashmi , S., Yiqiao , W., & Seung-Hoon , H. (2017). A survey on LPWA technology: LoRa and NB-IoT. *ICT Express*, 14-21.
- Represa, S. N., Giuliani, D., Candia, A., Luengo, M. Á., Porta, A. A., & Marrone, L. A. (2018). Soluciones para SmartCities: propuesta de un sistema de monitoreo de la calidad del aire basado en una red LoRaWAN con sensores portátiles. *IEEE*, 1-4.
- Rudd, A., Collina, M., Agor, M., & Buntsevich, S. (2020). *MQTT.js*. Obtenido de https://github.com/mqttjs/MQTT.js
- Schwaber, K., & Sutherland, J. (2013). La guía de Scrum. *La guía de Scrum*, 21.
- Schwaber, K., & Sutherland, J. (2017). La Guía de Scrum. *La Guía Definitiva de Scrum: Las Reglas del Juego*, 3-21.
- Secretaria Ambiente. (Septiembre de 2020). *Red de Metropolitana de Monitoreo Atmosferico de Quito* . Obtenido de http://www.quitoambiente.gob.ec/ambiente/index.php/politicas-y-planeacion-ambiental/red-de-monitoreo
- Secretaría de Ambiente. (2016). *Norma Ambiental de Calidad del Aire.* Obtenido de http://extwprlegs1.fao.org/docs/pdf/dom60781.pdf
- SEMTECH. (2020). *LoRa*. Obtenido de https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/
- SIGFOX. (2020). *SIGFOX España*. Obtenido de https://www.sigfox.es
- Statista. (15 de Diciembre de 2020). *React Native is the most popular cross-platform mobile framwork used by global developers, according to a 2020 developer survey.* Obtenido de Cross-platform mobile frameworks used by global developers 2020
- The Things Network. (2020). Obtenido de The Things Network: https://www.thethingsnetwork.org/docs/lorawan/security.html

- The World Air. (2019). *World Air Quality Index* . Obtenido de https://aqicn.org/scale/
- The World Air Quality Proyect. (2019). *Calidad del Aire*. Obtenido de https://aqicn.org/
- Universidad Internacional de Valencia. (23 de Abril de 2019). Obtenido de Qué es una Smart city: tecnología, ventajas y seguridad: https://www.universidadviu.com/que-es-una-smart-city-tecnologia-ventajas-y-seguridad/
- VARGAS REINOSO, L. P. (2018). ANÁLISIS COMPARATIVO ENTRE LORAWAN Y LTE EN UN ESCENARIO DE RED PARA LA INTERNET DE LAS COSAS USANDO EL SOFTWARE DE SIMULACIÓN. 7. Obtenido de http://repositorio.uchile.cl/bitstream/handle/2250/169977/An%C3%A1lisis-comparativo-entre-LORAWAN-y-LTE-en-un-escenario-de-red-para-la-Internet-de-las-cosas.pdf?sequence=1&isAllowed=y
- Weightless. (2020). *Weightless. Simply better*. Obtenido de http://www.weightless.org
- Xavier Villamil, T. G. (2019). App Móvil Desarrollada con Metodología Ágil para IoT Controlada desde una Red LAN/WAN con Placa de Desarrollo de Hardware Libre (Arduino). *Revista Ibérica de Sistemas e Tecnologias de Informação*, 379–392.

ANEXOS

ANEXO I: DESARROLLO PARTE TÉCNICA SPRINTS

Sprint 0

Sprint 0 Historia de Usuario A0

Análisis

Mediante la Tabla I.1 tabla defino los diferentes servicios y componentes que se utilizaran en la arquitectura de proyecto.

Tabla AI.1.- Servicios y Componentes para el Proyecto

Descripción	Componente
Información	Datos sensores, Índice calidad del aire, Protocolo Comunicación.
Servicios	Heroku, MongoDB Cloud
Negocio	Registro base de datos, Información en tiempo real
Aplicación	Aplicación Móvil

Diseño

Se procede a diseñar la arquitectura del proyecto para dar soporte a los requisitos funcionales y no funcionales, como se muestra en la Figura 2.8.

Implementación

En este apartado se procede a describir los elementos requeridos para el desarrollo del proyecto ver Tabla AI.2:

Tabla AI.2.- Herramientas de Software para el Desarrollo

Descripción	Definición
Backend	
Visual Studio Code	Es un editor de código fuente ligero que se ejecuta en múltiples plataformas con soporte para JavaScript, Node.js y TypeScript y un ecosistema de extensiones para otros lenguajes (Microsoft, 2020).
NodeJS	Es una tecnología para desarrollar aplicaciones multipropósito, creado en un entorno JavaScript orientado a eventos con la versatilidad de atender muchas conexiones en simultaneo, debido a que no utiliza hilos ni procesos para realizar sus funciones, por lo que es una alternativa viable a desarrollar sistemas escalabres en este framework. Utiliza HTTP que permite

	operaciones streaming de baja latencia (OpenJS Foundation, 2020).
ExpressJS	Infraestructura que se aplica en Node.js es flexible y mínima con características para aplicaciones móviles y web. Su API proporciona métodos HTTP y middleware complementando las funcionalidades con Node.js (OpenJS Foundation, 2020).
MQTT.js	Es una librería de código abierto para el protocolo MQTT, escrita en JavaScript para node.js (Rudd, Collina, Agor, & Buntsevich, 2020).
Mongoose	Es una librería escrita en Node.js para utilizar MongoDB (Mongoose, 2019).
MongoDB	Es una base de datos distribuida, se basa en documentos, creada para aplicaciones modernas con alta productividad. Almacena datos en documentos de tipo JSON una forma mejorada del modelo tradicional de filas y columnas. Los Documentos JSON permite almacenar objetos anidados y matrices como valores.

Se creó y configuro un nuevo proyecto e instaló las librerías que se utilizaran en el proyecto que ese implementara, ver Anexo II Package.json.

Pruebas

Tabla AI.3.- Caso de prueba 1

Caso de Prueba	
Número de Caso de Prueba: 1	**Número de Historia Usuario:** A0
Nombre de Historia de Usuario: Plataforma Tecnológica	
Descripción: Como sistema, necesito analizar los requerimientos técnicos que se utiliza para implementar en el proyecto.	
Condiciones de Ejecución: Se debe instalar las librerías indicadas en el Anexo II**.**	
Entradas: Subsistemas y servicios	
Resultado Esperado: Que Flutter y JavaScript este correctamente instalado en Visual Studio Code. Las librerías NodeJs, ExprerssJs, MQTT.js, Mongoose, MongoDB sean compatibles con la versión de JavaScript instalada.	
Evaluación: Satisfactoria Las metas de este sprint se cumplen en base a lo establecido y acordado.	

Sprint 1

Sprint 1 Historia de Usuario A1

Análisis

Para cumplir esta historia, es tener un dispositivo que mande datos al Gateway y esos datos sean enviados a CloudMQTT

Implementación

Desarrollo del Código

El código fue implementado con la librería LMIC para OTAA modificado para los dispositivos en la fase de planificación, es necesario que en el código los parámetros AppEUI, DevEUI tienen que ser ingresados en formato little endian: El formato little endian representa la secuencia de procesamiento en un orden donde el dispositivo procesa desde el bit menos significativo, LoRaWAN establece que los bytes en OTAA se deben enviar en little endian.

```
static const ul_t PROGMEM APPEUI[8] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
void os_getArtEui (ul_t* buf) {
  memcpy_P(buf, APPEUI, 8);
}

static const ul_t PROGMEM DEVEUI[8] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
void os_getDevEui (ul_t* buf) {
  memcpy_P(buf, DEVEUI, 8);
}
```

Appkey será escrito en formato Big endian esta codificación procesa el dispositivo en orden desde el bit más significativo, permitiendo con Appkey encriptar o desencriptar mediante AES la clave entre el nodo y la red (LOGITEK, 2020).

```
static const ul_t PROGMEM APPKEY[16] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
void os_getDevKey (ul_t* buf) {
  memcpy_P(buf, APPKEY, 16);
}
```

Según las especificaciones el intervalo de Trasferencia fue configurado en 5 segundo que en función del ciclo de transferencia final se traduce que se envié datos cada minuto.

```
#define TX_INTERVAL 5
```

Según las especificaciones de github que indica en la librería el mapa de pines, debe configurarse las conexiones de entrada y salida para el correcto funcionamiento del LoraShield, fue el siguiente (Kooijman, 2020):

```
lmic_pinmap lmic_pins = {
    .nss = 6,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 5,
    .dio = {2, 3, 4},
};
```

Configuración Pines Sensor Arduino

Los datos son recibidos en el pin A0, los datos son almacenados en una variable de tipo double.

```
Lecturasensor = analogRead(MQ_PIN);
```

Existe una transformación de datos según el voltaje máximo que puede percibir

```
voltaje = lecturasensor * (5 / 1023.0);
```

El valor del voltaje captado por el sensor es transformado a valores de partes por millón. Los valores captados por el sensor van en el intervalo de 0 a 5 y debido a que la salida analógica es un valor en escala logarítmica y como antecedente la Universidad de Brac en Estados Unidos realizó un estudio del sensor MQ-7 el cálculo de partes por millón con la siguiente formula (Ananto, Ahmed, & Ferdousi):

$$Valor\ CO\ ppm = \frac{95{,}501}{\left(\frac{Vc-Vo}{Vout}\right)^{1.543}}$$

Donde:

Vc es el voltaje que es alimentado el sensor (5v)

Vout es el voltaje de la salida analógica (0 a 5v)

El tipo de datos que es enviado en el payload es del tipo byte

```
static byte mydata[5];
```

Para identificar cual sensor es el que envía datos y tener facilidad en identificar cual es dentro del mensaje del payload se lo realizo de la siguiente manera:

`mydata[0] = 'a' y mydata[0] ='b';` para el segundo dispositivo.

Debido a que el tipo de dato byte no soporta decimales, se separó en partes el número la parte entera y la parte decimal de la siguiente manera:

```
byte d1 = int(monoxidodeCaborno);
byte d2 = int((monoxidodeCaborno - d1) * 100 + 1);
```

Los datos son almacenados y enviados de la siguiente manera:

Indicando el puerto de envió, los datos a enviar, el tamaño de datos a enviar y por último si se desea que los datos enviados sean confirmados (Chirpstack, 2020).

```
LMIC_setTxData2(1, mydata, sizeof(mydata), 0);
```

Finalmente se programa el siguiente envío de datos según el intervalo de tiempo establecido anteriormente:

```
os_setTimedCallback(j, os_getTime() + sec2osticks(TX_INTERVAL),
do_send);
```

Diagrama Node-RED

En la Figura AI.1, muestra el diagrama los mensajes son recibidos por el nodo mediante la tarjeta de acceso MTAC-LORA

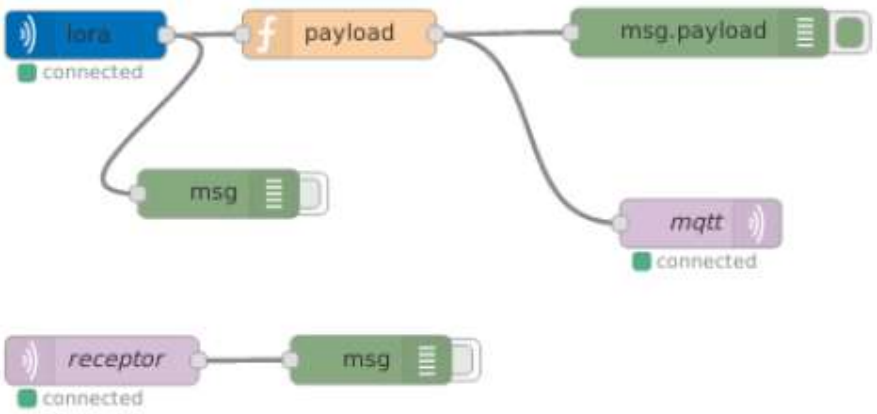


Figura AI.1.- Diagrama Node-RED Proyecto

La función “payload” de la Figura AI.2 filtra los datos recibidos para poder procesar los datos de los sensores implementados en el proyecto.

```
var sen1=msg.payload[0];

    if (sen1==97) {
         sen1= "primero"
         var mqttPayload = {

    device_id: msg.deveui,
    sensor: sen1,
    monoxidoppm:msg.payload[1]+msg.payload[2]/100,
    monoxidomg:msg.payload[3]+msg.payload[4]/100,
    tiempo:msg.time

         }
msg.payload = mqttPayload;

```

Figura AI.2.- Función payload Node-RED

La función “mqtt” de la Figura AI.3 publica a Cloud MQTT los datos en formato JSON

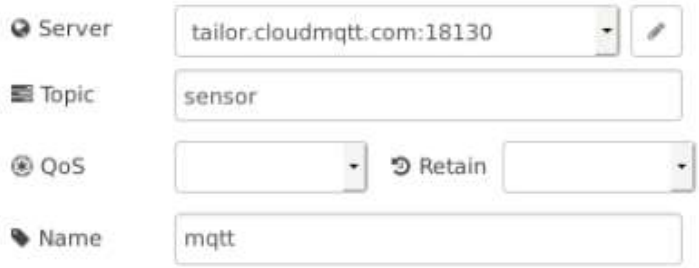


Figura AI.3.- Función mqtt Node-RED

Sprint 1 Historia de Usuario A2

Análisis

Para cumplir con este requerimiento se analizó el tipo de datos recibidos por el CloudMQTT y cómo transformarlos.

Diseño

Para cumplir los requerimientos de esta historia de usuario se procedió a crear una clase que permita estar en estado de suscripción con CloudMQTT ver Anexo III Clase Mqtttodb.js, a su vez se tuvo que transformar los datos para que se adapten y validen con el modelo de base de datos ver Tabla AI4, hay atributos extra que se pueden utilizar para trabajos futuros e implementar una función que guarde en la base de datos Ver Anexo III Función ifdatos.

Tabla AI.4.- Atributos modelo base datos

Atributo	Tipo	Descripción
deveui	String	Identificador único del sensor
namesensor	String	El nombre del sensor asignado
latitud	Number	Posición latitud sensor
longitud	Number	Posición longitud sensor
sensormq	Number	Valor partes por millón
mqco	Number	Valores en mg/m^3
batery	Number	Valor referencia de batería
hora	String	Hora de medición valor
fecha	String	Fecha de medición valor
airq	Number	Número valor calidad del aire
airqc	String	Descripción calidad del aire

Implementación

Para implementar esta historia se la debe instanciar la clase Mqtttodb.js ver Anexo III en la clase principal del backend, la que permitirá tener activa la funcionalidad que requiere el proyecto.

Pruebas

Tabla AI.5.- Caso de prueba 2

Caso de Prueba	
Número de Caso de Prueba: 2	**Número de Historia Usuario:** A2
Nombre de Historia de Usuario: Gestión de conexiones del backend	
Descripción: Como sistema, necesito un backend que me permita gestionar las conexiones con CloudMQTT y la base de datos.	
Condiciones de Ejecución: Se debe tener la clase Mqtttodb.js instanciada en la clase principal del backend.	
Entradas: Datos provenientes CloudMQTT	
Resultado Esperado: Los datos sean leídos por el backend para su posterior procesamiento.	
Evaluación: Satisfactoria	

Tabla AI.6.- Caso de prueba 3

Caso de Prueba	
Número de Caso de Prueba: 3	**Número de Historia Usuario:** A2
Nombre de Historia de Usuario: Gestión de conexiones del backend	
Descripción: Como sistema, necesito un backend que me permita gestionar las conexiones con CloudMQTT y la base de datos.	
Condiciones de Ejecución: Se debe tener la clase Mqtttodb.js instanciada en la clase principal del backend.	
Entradas: Datos provenientes CloudMQTT	
Resultado Esperado: Los datos serán formateados.	
Evaluación: Satisfactoria	

Sprint 1 Historia de Usuario A3

Análisis

Para cumplir con este requerimiento se analizó el modelo de la base de datos Tabla AI4, para poder procesar los datos y consultas en formato correcto para posterior ser tomados por el frontend.

Diseño

Para cumplir los requerimientos de esta historia de usuario se procedió a crear la clase monóxido.js ver Anexo IV que contiene la función AQICO para el cálculo de la calidad aire, tomando los parámetros de referencia para el monóxido de carbono (The World Air Quality Proyect, 2019), esta función permite determinar el rango de calidad del aire del valor medido por el sensor, valor con función AQICategory es categorizada ver Anexo IV y guardar los datos en la base de datos.

Implementación

Para implementar esta historia se la debe instanciar la clase la RutaSensor.js ver Anexo IV en la clase principal del backend, el backend responderá peticiones del frontend ver Anexo IV Clase Rutasensor.js.

Pruebas

Tabla AI.7.- Caso de prueba 4

Caso de Prueba	
Número de Caso de Prueba: 4	**Número de Historia Usuario:** A3
Nombre de Historia de Usuario: Gestión de conexiones del backend	
Descripción: Como sistema, necesito un backend que me permita gestionar los datos categorizándolos y guardándolos según el modelo de la base de datos, y generar rutas de consulta para el frontend.	
Condiciones de Ejecución: Se debe tener la clase Mqtttodb.js instanciada en la clase principal del backend.	
Entradas: Datos sensor partes por millón	
Resultado Esperado: Los datos son guardados con los datos recibidos por CloudMQTT y son guardados en la base de datos	
Evaluación: Satisfactoria	

```
2020-11-16T19:35:22.973851+00:00 app[web.1]: {
2020-11-16T19:35:22.973867+00:00 app[web.1]:   device_id: '34-63-a6-b0-56-76-0a-8c',
2020-11-16T19:35:22.973868+00:00 app[web.1]:   sensor: 'segundo',
2020-11-16T19:35:22.973869+00:00 app[web.1]:   hora: '14:35:22',
2020-11-16T19:35:22.973869+00:00 app[web.1]:   fecha: '2020-11-16',
2020-11-16T19:35:22.973869+00:00 app[web.1]:   lecsen: 0.73,
2020-11-16T19:35:22.973870+00:00 app[web.1]:   monoxido: 0.82,
2020-11-16T19:35:22.973870+00:00 app[web.1]:   airq: 9,
2020-11-16T19:35:22.973870+00:00 app[web.1]:   airqc: 'Bueno'
2020-11-16T19:35:22.973871+00:00 app[web.1]: }
2020-11-16T19:35:37.722635+00:00 app[web.1]: {
2020-11-16T19:35:37.722681+00:00 app[web.1]:   device_id: '82-a7-72-bf-89-ad-02-c8',
2020-11-16T19:35:37.722682+00:00 app[web.1]:   sensor: 'primero',
2020-11-16T19:35:37.722682+00:00 app[web.1]:   hora: '14:35:37',
2020-11-16T19:35:37.722682+00:00 app[web.1]:   fecha: '2020-11-16',
2020-11-16T19:35:37.722683+00:00 app[web.1]:   lecsen: 2,
2020-11-16T19:35:37.722683+00:00 app[web.1]:   monoxido: 2.28,
2020-11-16T19:35:37.722684+00:00 app[web.1]:   airq: 25,
2020-11-16T19:35:37.722684+00:00 app[web.1]:   airqc: 'Bueno'
2020-11-16T19:35:37.722684+00:00 app[web.1]: }
```

Figura AI.4.- Datos formateados que se guardan en la base de datos

Tabla AI.8.- Caso de prueba 5

Caso de Prueba	
Número de Caso de Prueba: 5	**Número de Historia Usuario:** A3
Nombre de Historia de Usuario: Gestión de datos del backend	
Descripción: Como sistema, necesito un backend que me permita gestionar los datos categorizándolos y guardándolos según el modelo de la base de datos, y generar rutas de consulta para el frontend.	
Condiciones de Ejecución: Se debe tener la clase Rutasensor.js instanciada en la clase principal del backend.	
Entradas: Datos provenientes de la base de datos MongoDB	

Resultado Esperado: Que el frontend retorne valores para que retorne los valores de los valores diarios, valores promedio diario, valor calidad del aire, valores máximos, mínimos y valores recientes.
Evaluación: Satisfactoria

Figura AI.5.- Consulta datos promedio diario, ruta api/sensor/datadayquality/

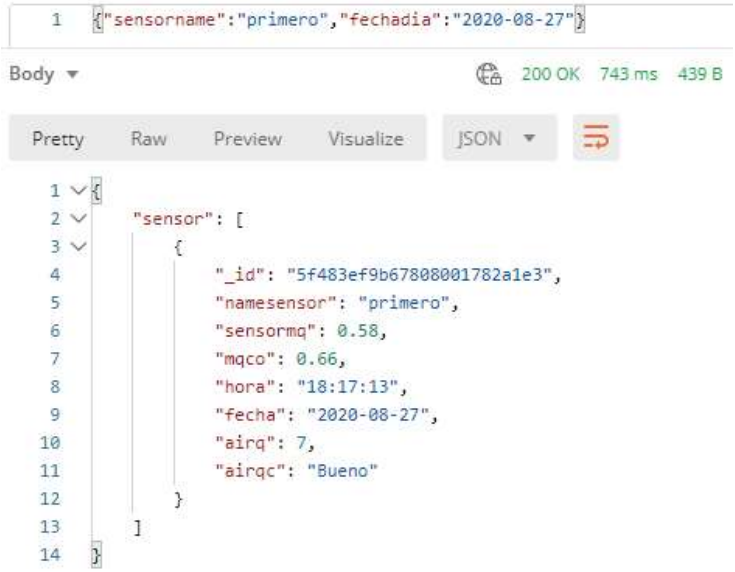


Figura AI.6.- Consulta dato máximo diario, ruta api/sensor/datosmax/

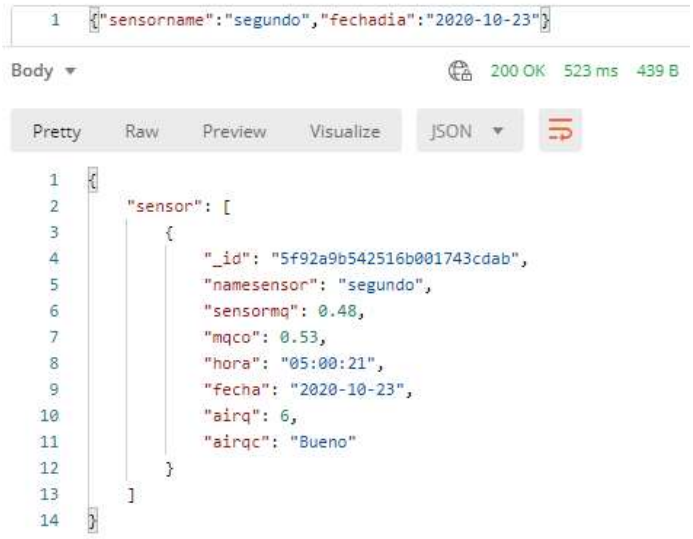


Figura AI.7.- Consulta datos promedio diario, ruta api/sensor/datosmin/

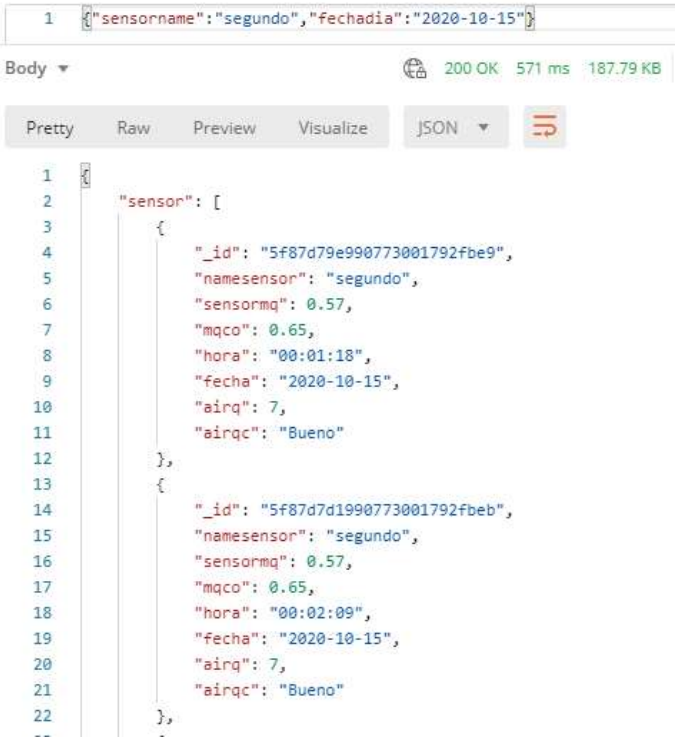


Figura AI.8.- Consulta datos diario, ruta api/sensor/dataday/

Sprint 2

Sprint 2 Historia de Usuario B1

Análisis

Para cumplir con este requerimiento, es tener una interfaz de una aplicación móvil para ver la cantidad de contaminación que capta los sensores.

Diseño

La aplicación enviara peticiones http al backend retornando información extraída de la base de datos MongoDB en formato JSON. El framework de desarrollo móvil que se utilizara es Flutter ver Tabla AI9.

Tabla AI.9.- Descripción Flutter

Frontend	
Flutter	Flutter es un framework de código abierto desarrollado por Google, para realizar aplicaciones nativas para móvil para Android y IOS, web y escritorio mediante el lenguaje Dart (Flutter, 2020).

Diseño de Interfaces

El esqueleto principal de la aplicación consta de un menú principal, que consta de tres módulos hecho de widgets, esto permite tener un desarrollo más estructurado en la programación y de cada componente de la interfaz.

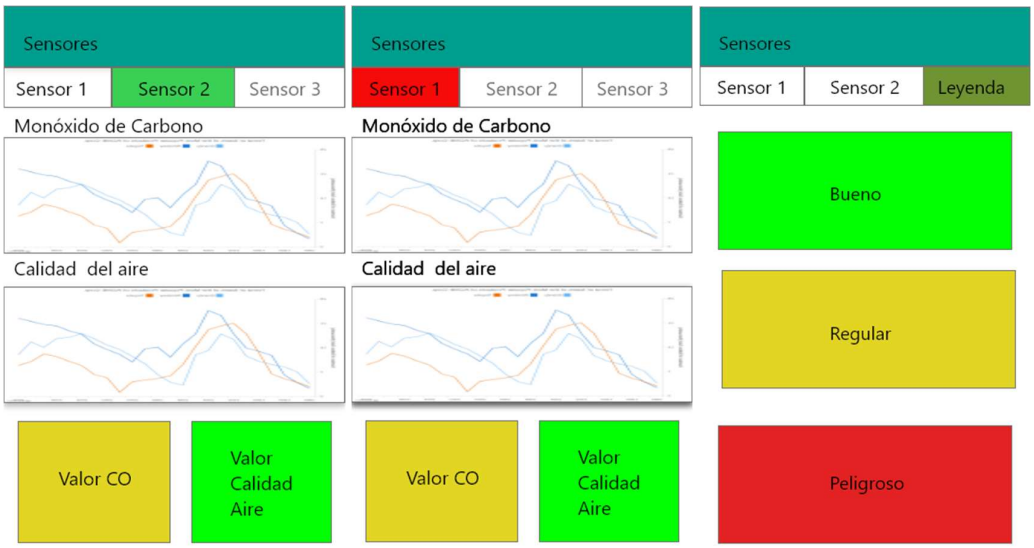


Figura AI.9.- Diseño interfaz aplicación Calidad del Aire

La Aplicación esta diagramado de la siguiente forma:

- FIS
- LabFIS
- Leyenda

Árbol de estructura de la interfaz gráfica:

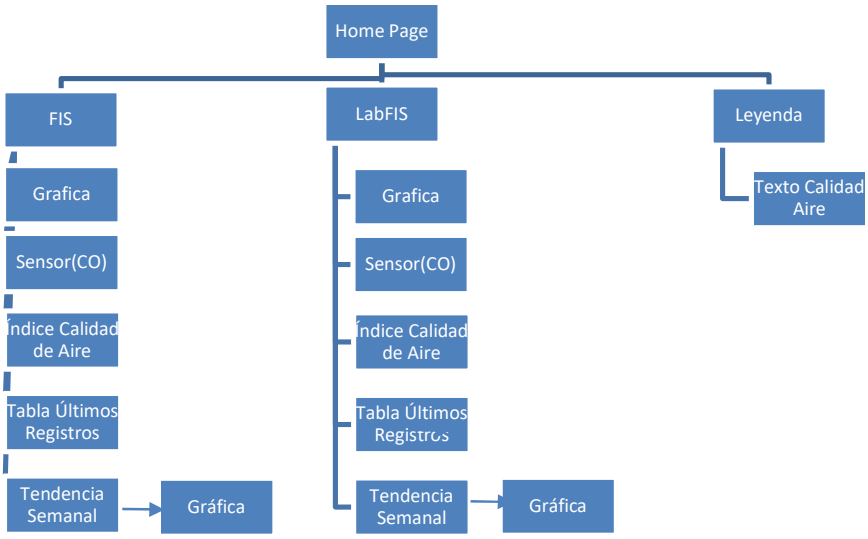


Figura AI.10.- Árbol de estructura interfaz gráfica aplicación

Implementación

Debido a que ya está implementado el backend y generado la ruta de consulta. La interfaz que se va a crear se basa en la Figura AI.10, la clase Home Main ver Anexo V clase main.dart, tiene tres módulos correspondientes para FIS, LabFIS y Leyenda, esto permite al usuario navegar entre las interfaces de la aplicación ver Figura AI.11.

Figura AI.11.- Widget Menú Tab Principal

El widget interfaz.dart ver Anexo V se configura la dimensión que tendrán los Widgets de la interfaz principal Figura AI.12 y que se seguirán implementado en el desarrollo del Sprint.

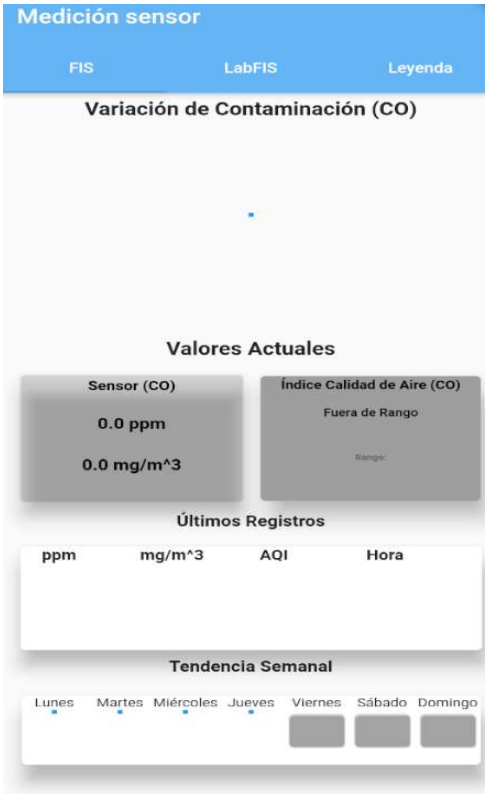


Figura AI.12.- Widget Menú Principal

Los widgets que se implementaran en la interfaz principal ver Figura AI.12 son:
El widget datalist.dart ver Anexo V permite ver los registros en forma lista de los últimos 40 minutos deslizándose por el listado, Figura AI.13.

ppm	mg/m^3	AQI	Hora
0.25	0.28	Bueno	08:08:15
0.25	0.28	Bueno	08:10:01
0.25	0.28	Bueno	08:10:55

Figura AI.13.- Widget Lista de Registro

El widget graficaprincipal.dart ver Anexo V permite graficar con la librería Animated Line Chart los valores de contaminación actual, la clase datagrafico.dart ver Anexo V formatea los datos para la correcta visualización de datos. La ventaja de esta librería permite al usuario seleccionar un punto dentro de la gráfica y obtener más información del punto seleccionado Figura AI.14.

Los puntos son graficados para el eje de las x el tiempo y para el eje y los valores en partes por millón de los sensores. En caso de que no tenga internet el dispositivo o no exista datos de los sensores se mostrara un círculo de progreso hasta que el estado cambie Figura AI.15.

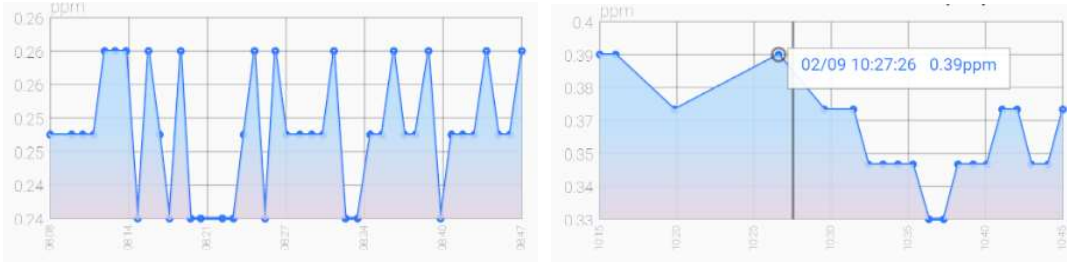


Figura AI.14.- Widget grafica últimos registros

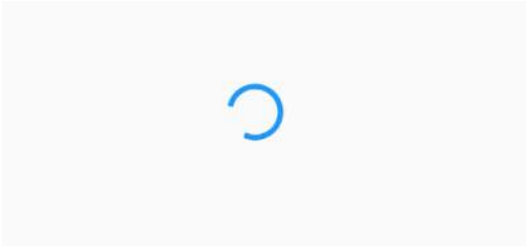


Figura AI.15.- Widget Circulo Progreso

Widget Valor Actual ver Anexo V, muestra el ultimo registro de los valores registrados en formato ppm y mg/m^3 ver Figura AI.16.

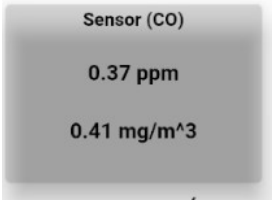


Figura AI.16.- Widget Sensor (CO)

Widget Índice Calidad del Aire ver Anexo V, muestra la información procesada para ser visualizada por el usuario de una forma más fácil de entender, este Widget cambia de color en función de la calidad del aire captado Figura AI17 por el sensor con los datos son procesados por la funcion Air_Quality y Textaqi ver Anexo V.

Figura AI.17.- Widget Índice Calidad del Aire

Widget Tendencia Semanal ver Anexo V, crear una lista de los días disponibles de la semana a la fecha presente se tuvo que realizar un barrido de cada día y solo habilitar los días de la semana correspondientes ver Figura AI.18.

Figura AI.18.- Widget Tendencia Semanal

Pruebas

Tabla AI.10.- Caso de prueba 6

Caso de Prueba	
Número de Caso de Prueba: 6	**Número de Historia Usuario:** B1
Nombre de Historia de Usuario: Desarrollo interfaz principal de la aplicación móvil	
Descripción: Como usuario, necesito una interfaz principal que me permita visualizar los datos de los sensores del proyecto representados en forma de gráfico de puntos y mostrar registros de la contaminación del aire en forma de lista y último registro.	
Condiciones de Ejecución: Se debe tener el servidor ejecutándose que permita recibir peticiones http y tenga conexión a la base de datos.	
Entradas: Interface principal Aplicación	
Resultado Esperado: Que los datos sean graficados en el Widget correspondientes, se despliegue los datos de los últimos registros, se pueda observar los Widgets de registro actual y calidad del aire y se muestre bien el Widget tendencia semanal	
Evaluación: Satisfactoria	

Sprint 3

Sprint 3 Historia de Usuario C1

Análisis

El requerimiento para cumplir este Sprint es necesario tener la interfaz principal de la aplicación móvil para que el backend funcione correctamente y despliegue los datos gráficamente, los máximos y mínimos registrados ese día.

Diseño

La aplicación enviara peticiones http al backend retornando información extraída de la base de datos MongoDB en formato JSON. A continuación, se describe el diseño de pantalla de la gráfica, valores mínimo y máximo de un día especifico ver Figura AI.19.

Figura AI.19.- Interfaz Día Medición Contaminación

Implementación

Esta historia de usuario se debe crear la interfaz que contengan Widgets de gráfica día y los iconos de máximo y mínimo.

La grafica de un día determinado es la replicación de la función de la gráfica principal de la aplicación con el rango ampliado de toma de datos, ver Figura AI.20.

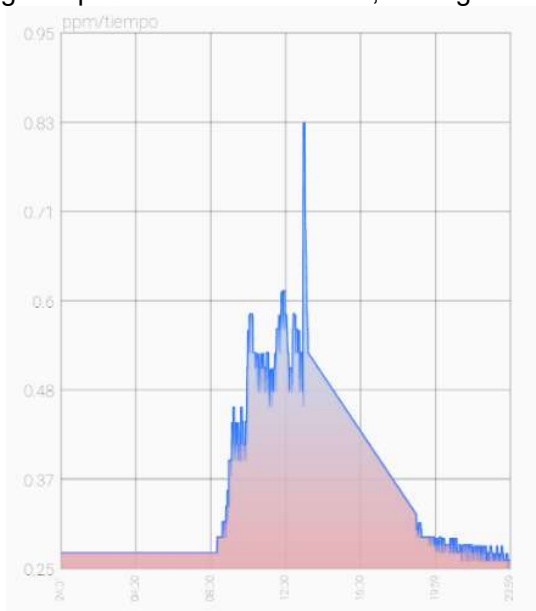


Figura AI.20.- Widget Día Medición Contaminación

Para crear los Widgets para visualizar los valores máximos y mínimos Figura AI.21, se realizaron dos funciones que hace una petición post al backend que calcula el valor máximo

y mínimo, las horas de las mediciones de un día determinado y lo retorna a la aplicación, ver Anexo VI Widget Máximo y Mínimo.

Figura AI.21.- Widget Máximo y Mínimo

Para el desarrollo de la interfaz Variación de la Contaminación ver Figura AI19, donde se visualiza los datos de un día determinado, en la cabecera superior se tiene el día que se está visualizando, con la fecha correspondiente, la gráfica de todos los datos medidos, y los valores máximos y mínimos captados, ver Anexo VI Interfaz Día Medición Contaminación.

Pruebas

Tabla AI.11.- Caso de prueba 7

Caso de Prueba	
Número de Caso de Prueba: 7	**Número de Historia Usuario:** A2
Nombre de Historia de Usuario: Desarrollo interfaz datos diarios de la aplicación móvil	
Descripción: Como usuario, necesito una interfaz que me permita visualizar los datos de los sensores del proyecto en forma de gráfica de puntos y mostrar los registros de picos de medición de un día determinado.	
Condiciones de Ejecución: Se debe tener el servidor ejecutándose que permita recibir peticiones http y tenga conexión a la base de datos.	
Entradas: Datos provenientes de la interfaz principal.	
Resultado Esperado: Los datos sean leídos por el backend y sean visualizado correctamente la gráfica y los valores de los mínimos y máximos	
Evaluación: Satisfactoria	

Sprint 4

Sprint 4 Historia de Usuario D1

Implementación

Para la corrección del error en la visualización de los datos se implementó:
Este fragmento de código permite tener un número con dos cifras significativas

```
Number(parsedMessage.monoxidoppm.toFixed(2))
```

Se corrigió las rutas del backend debido a que existía un error de compilado en Heroku adicionalmente a ciertas limitantes en la conexión en el tiempo de conexión en Heroku se implementó el siguiente código en el backend para el nuestro servidor este en escucha todo el tiempo.

```
var reqTimer = setTimeout(function wakeUp() {
    request("https://thesisairquality.herokuapp.com", function() {
       console.log("WAKE UP DYNO");
    });
```

```
    return reqTimer = setTimeout(wakeUp, 1200000);
}, 1200000);
```

Para que la aplicación tenga permisos para acceder a internet, determino el nombre de la app y el icono de ja app debemos adicionar la configuración en AndroidManifest.xml.

```
<uses-permission android:name="android.permission.INTERNET" />
```

Cambio de nombre del proyecto:

```
android:label="Tesis Calidad del Aire"
```

El backend cuando es ejecutado verifica si existe o no la base de datos a la que está tratando de conectar, esto facilita la creación e inicialización de la base de datos.

En el backend se configuro las variables globales para que exista conexión entre la base de datos y exista disponible para recibir peticiones de la aplicación móvil.

Pruebas

Tabla AI.12.- Caso de prueba 8

Caso de Prueba	
Número de Caso de Prueba: 8	**Número de Historia Usuario:** D1
Nombre de Historia de Usuario: Corrección bugs backend, frontend	
Descripción: Como aplicación, necesito corregir los bugs de manejo de datos en el backend y frontend, para la publicación del backend en la nube y compilado de la aplicación para Android.	
Condiciones de Ejecución: Tener una cuenta en Heroku	
Entradas: Servicios	
Resultado Esperado: Verificar que el backend este ejecutándose en Heroku	
Evaluación: Satisfactoria	

Figura AI.22.- Reporte backend en Heroku

Tabla AI.13.- Caso de prueba 9

Caso de Prueba	
Número de Caso de Prueba: 9	**Número de Historia Usuario:** D1
Nombre de Historia de Usuario: Corrección bugs backend, frontend	
Descripción: Como aplicación, necesito corregir los bugs de manejo de datos en el backend y frontend, para la publicación del backend en la nube y compilado de la aplicación para Android.	
Condiciones de Ejecución: Aplicación desarrolla en Flutter	
Entradas: Api Flutter	

Resultado Esperado: Correcto compilado de la aplicación
Evaluación: Satisfactoria

Figura AI.23.- Aplicación Proyecto Instalada

ANEXO II: CONFIGURACIONES BACKEND Y FRONTEND

Package.json

Se instalo NodeJS en Visual Studio Code creamos el nuevo proyecto con “npm init” la información del proyecto es la siguiente

```
"name": "backend_quality_air",
 "version": "1.0.0",
 "description": "proyecto de air quality ",
 "main": "./index.js",
```

Se instanciaron dentro del package.json del backend las librerías doten que permite crear variables de entorno globales, express API que permite implementar métodos HTTP, Morgan nos permitirá visualizar las solicitudes HTTP del backend, mqtt nos permitió implementar métodos para conectarnos a CloudMQTT y para la parte de desarrollo se utilizó nodemon que permite reiniciar el backend cuando se registren cambios en el código.

```
 "dependencies": {
   "dotenv": "^8.2.0",
   "express": "^4.17.1",
   "mongoose": "^5.9.20",
   "morgan": "^1.10.0",
   "mqtt": "^4.1.0",
   "request": "^2.88.2"
 }
```

ANEXO III: CONEXIÓN BASE DE DATOS Y CLOUDMQTT

Clase database.js

Se creo una función asincrónica para realizar la conexión entre nuestro backend y la base de datos MongoDB, se definió inicialmente variables locales para el desarrollo, pero se definió variables globales para la futura implementación en Heroku.

```
const mongoose=require('mongoose');
require('dotenv').config({path:'variables.env'});

async function connect(){
    await mongoose.connect(process.env.DB_URL
    ,{useNewUrlParser:true}
    );
        console.log('Data Base: Connected')
};
module.exports={connect};
```

Clase Mqtttodb.js

La función Mqtttodb permite estar en estado de suscriptor de los datos que provienen de CloudMQTT y la configuración es la siguiente:

```
const options = {
    port: 14433,
    host: 'mqtt://.cloudmqtt.com',
    clientId: 'mqttjs_' + Math.random().toString(16).substr(2, 8),
    username:
    password:
    keepalive: 60,
    reconnectPeriod: 1000,
    protocolId: 'MQIsdp',
    protocolVersion: 3,
    clean: true,
    encoding: 'utf8'
};
```

La función permite conectarse a CloudMQTT y transforma los datos recibidos en un json, los datos de la hora se transforman a zona horaria perteneciente a Ecuador y llama a la función de la calidad del aire, además implementa un try y catch para evitar problemas que existían en la función cuando se realizaba una reconexión con CloudMQTT.

```
const client = mqtt.connect('mqtt://soldier.cloudmqtt.com', options);
try {
    client.on('connect', function () {
        console.log('conectado mqtt');
        client.subscribe('sensor', function () {
            client.on('message', function (topic, message, packet) {
            var ecTime = new Date().toLocaleString("en-
            US", {timeZone: "America/Bogota"});
                var date = new Date(ecTime);
```

```
                var d = date.getDate();
                var m = date.getMonth() + 1;
                var y = date.getFullYear();
                    var dateString = y + '-' + (m <= 9 ? '0' + m : m) + '-
            ' + (d <= 9 ? '0' + d : d);
                datos = JSON.stringify({
                    device_id: parsedMessage.device_id,
                    sensor: parsedMessage.sensor,
                    hora: new Date(ecTime).toLocaleTimeString('en-US', {
                        hour12: false,
                        hour: "numeric",
                        minute: "numeric",
                        second: "numeric"
                    }),
                    fecha: dateString,
                    lecsen: parsedMessage.monoxidoppm,
                    monoxido: parsedMessage.monoxidomg,
                    airq: monoxido.AQICO(parsedMessage.monoxidomg),
                   airqc:monoxido.AQICategory(monoxido.AQICO(parsedMessage.
                  monoxidomg))});
                var objetodato = JSON.parse(datos);
                ifdatos(objetodato);
            });});});});
} catch (error) {
    client.end
}
```

Función ifdatos

La función ifdatos guarda los datos procesados en función del modelo de la base de datos, el modelo implementado permite validar que los datos ingresados.

```
function ifdatos(objetodato) {
       Sensor.create({
            deveui: objetodato.device_id,
            namesensor: objetodato.sensor,
            latitud: 0,
            longitud: 0,
            sensormq: objetodato.lecsen,
            mqco: objetodato.monoxido,
            batery: 100,
            hora: objetodato.hora,
            fecha: objetodato.fecha,
            airq:objetodato.airq ,
            airqc:objetodato.airqc
        });}
```

ANEXO IV: CLASIFICACIÓN DE DATOS Y CREACIÓN DE RUTAS DE CONSULTA BACKEND

Clase monóxido.js

Función AQICO

La función AQICO permite determinar en qué rango de calidad del aire se encuentra el valor medido por el sensor:

```
function AQICO(Concentration)
{
var Conc=parseFloat(Concentration);
var c;
var AQI;
c=(Math.floor(10*Conc))/10;
if (c>=0 && c<4.5)
{
    //good
    AQI=Linear(50,0,4.4,0,c);
}
else if (c>=4.5 && c<9.5)
{
    AQI=Linear(100,51,9.4,4.5,c);
    //Moderate
}
else if (c>=9.5 && c<12.5)
{
    AQI=Linear(150,101,12.4,9.5,c);

    //Unhealthy for Sensitive Groups
    /*
}
else if (c>=12.5 && c<15.5)
{
    AQI=Linear(200,151,15.4,12.5,c);
    //Unhealthy
}
else if (c>=15.5 && c<30.5)
{
    AQI=Linear(300,201,30.4,15.5,c);
    //Very Unhealthy
}
else if (c>=30.5 && c<40.5)
{
    AQI=Linear(400,301,40.4,30.5,c);
    //Hazardous
}
else if (c>=40.5 && c<50.5)
{
    AQI=Linear(500,401,50.4,40.5,c);
     //Hazardous
}
else
{
    AQI="Fuera de Rango";
}
return AQI;
}
```

Función AQICategory

La función AQICategory permite categorizar en que rango está dentro de los parámetros definidos por (The World Air Quality Proyect, 2019). Esta función permite tener una forma fácil de identificar para el usuario la calidad del aire que capta cada sensor.

```
function AQICategory(AQIndex)
{
var AQI=parseFloat(AQIndex)
var AQICategory;
if (AQI<=50)
{
    AQICategory="Bueno";
}
else if (AQI>50 && AQI<=100)
{
    AQICategory="Moderado";
}
else if (AQI>100 && AQI<=150)
{
    AQICategory="Muy poco saludable para grupos sensibles";
}
else if (AQI>150 && AQI<=200)
{
    AQICategory="Poco saludable";
}
else if (AQI>200 && AQI<=300)
{
    AQICategory="Muy poco saludable";
}
else if (AQI>300 && AQI<=400)
{
    AQICategory="Peligroso";
}
else if (AQI>400 && AQI<=500)
{
    AQICategory="Peligroso";
}
else
{
    AQICategory="Fuera de Rango";
}
return AQICategory;
}
module.exports={AQICategory,AQICO};
```

Clase Rutasensor.js

Creamos con Expressjs rutas para poder obtener información de la base de datos y poder mostrarlos en la aplicación móvil.

La ruta /api/sensor/datos/ muestra los datos de cada sensor almacenados.

```
router.post('/api/sensor/datos/', async (req, res) => {
    //consulta desde el navegador y mandar datos
    const { sensorname } = req.body;
    console.log(sensorname);
    const sensor = await Sensor.find({ 'namesensor': sensorname }
        , { sensormq: 1, mqco: 1, hora: 1, fecha: 1, namesensor: 1 }
    );
    res.json(sensor);
});
```

La ruta post /api/sensor/dashboard/ realiza la consulta en la base de datos con los parámetros con fechas y horas específicas.

```
router.post('/api/sensor/dashboard/', async (req, res) => {
    const { sensorname, fecha1, fecha2, hora1, hora2 } = req.body;
    const sensor = await Sensor.find({
        'namesensor': sensorname,
        hora: { $gte: hora1, $lte: hora2 },
        fecha: { $gte: fecha1, $lte: fecha2 }
    }
        , { sensormq: 1, mqco: 1, hora: 1, fecha: 1, namesensor: 1, airq: 1, ai
rqc: 1 }
    );
    console.log(sensorname);
    res.json({ sensor });
});
```

La ruta post /api/sensor/dataday/ consulta en la base de datos con los parámetros del nombre del sensor y la fecha correspondiente para retornar los datos registrados en un día especifico.

```
router.post('/api/sensor/dataday/', async (req, res) => {
    const { sensorname, fechadia } = req.body;
    const sensor = await Sensor.find({
        'namesensor': sensorname,
        hora: {$gte: '00:01:00', $lte: '23:59:59'},
        fecha: fechadia}
      , {sensormq: 1, mqco: 1, hora: 1, fecha: 1, namesensor: 1, airq: 1,
      airqc: 1});
    console.log(sensorname);
    res.json({ sensor });
});
```

La ruta post /api/sensor/datadayquality/ retorna el valor de la calidad del aire del valor promedio de un día.

```
router.post('/api/sensor/datadayquality/', async (req, res) => {
    const { sensorname, fechadia } = req.body;
    const sensor = await Sensor.find({
        'namesensor': sensorname,
        hora: { $gte: '00:01:00', $lte: '23:59:59' },
        fecha: fechadia} , { mqco: 1 }
    );
    var valmed = 0;
    var proday = 0.0;
    var aqip = 0,categoria='';
    for (var i = 0; i < sensor.length; i++) {
        valmed += sensor[i]['mqco']
    }
```

```
    if (sensor.length != 0) {
        proday = valmed / sensor.length;
        aqip = AQICO(proday);
        categoria = AQICategory(aqip);
    }
    res.json({ 'valorm':parseFloat(proday.toFixed(3)) , 'aqi': aqip,'categoria'
:categoria });
});
```

La ruta post /api/sensor/datosmin/ retorna el valor mínimo y la hora de registro por el sensor en un día determinado.

```
router.post('/api/sensor/datosmin/', async (req, res) => {
    const { sensorname, fechadia } = req.body;
    const sensor = await Sensor.find(
        {
        'namesensor': sensorname,
        hora: { $gte: '00:01:00', $lte: '23:59:59' },
        fecha: fechadia,
    }
, { sensormq: 1, mqco: 1, hora: 1, fecha: 1, namesensor: 1, airq: 1, airqc: 1 }
        ).sort({mqco:1}).limit(1);
        res.json({ sensor });
});
```

La ruta post /api/sensor/datosmax/ retorna el valor máximo y la hora de registro por el sensor en un día determinado.

```
router.post('/api/sensor/datosmax/', async (req, res) => {
    const { sensorname, fechadia } = req.body;
    const sensor = await Sensor.find(
        {
        'namesensor': sensorname,
        hora: { $gte: '00:01:00', $lte: '23:59:59' },
        fecha: fechadia,
    }
, { sensormq: 1, mqco: 1, hora: 1, fecha: 1, namesensor: 1, airq: 1, airqc: 1 }
        ).sort({mqco:-1}).limit(1);
        res.json({ sensor });
});
```

ANEXO V: WIDGETS INTERFAZ PRINCIPAL

Widget main.dart

Se utilizo un DefaultTabController para poder crear el texto principal de cabecera “Medición sensor” y los tabs de FIS, LabFIS y Leyenda

```
DefaultTabController(
        length: 3,
        child: Scaffold(
          appBar: AppBar(
              title: Text('Medición sensor',),
              backgroundColor: Colors.blue[300],
              bottom: TabBar(
                tabs: <Widget>[
                  Tab(text: 'FIS',),
                  Tab(text: 'LabFIS',),
                  Tab(text: 'Leyenda',)
                ],
                unselectedLabelColor: Colors.white,
                indicatorColor: Colors.black12,
              )),
          body: TabBarView(children: containers),
        ));
```

El Widget DefaultTabController llama a sus hijos Widgets que son las interfaces de visualización de datos

```
  List<Widget> containers = [
    Container(child: Interfaz('primero')height: 120, ),
    Container(child: Interfaz('segundo'), height: 150, ),
    Container(child: Text('Indicación aqi'), height: 150,),
  ];
}
```

Widget interfaz.dart

Para la construcción principal de la interfaz se utilizó un StaggeredGridView que permite agregar, cambiar de tamaño a los Widget para obtener una mejor visualización de datos, aquí se realizó el llamado a los Widgets que se seguían implementando durante el desarrollo de los Sprint.

```
Widget build(BuildContext context) {
    return StaggeredGridView.count(
      scrollDirection: Axis.vertical,
      shrinkWrap: true,
      physics: ScrollPhysics(),
      crossAxisCount: 2,
      crossAxisSpacing: 5,
      mainAxisSpacing: 3,
```

```
        padding: EdgeInsets.symmetric(horizontal: 15, vertical: 5),
        children: <Widget>[
          Padding(
            padding: const EdgeInsets.all(0.0),
            child: Text(
              'Variación de Contaminación (CO)',
              textAlign: TextAlign.center,
              style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,),),),),
          Padding(
            padding: const EdgeInsets.all(0.0),
            child: Graficafl(sensor),
          ),
          Padding(
            padding: const EdgeInsets.all(0.0),
            child: Text(
              'Valores Actuales',
              textAlign: TextAlign.center,
              style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,), ),),),
          Padding(
            padding: const EdgeInsets.all(0.0),
            child: IconoSensor(sensor),),
          Padding(padding: const EdgeInsets.all(1), child: IconoCalidad(sensor
)),
          Padding(
            padding: const EdgeInsets.all(3.0),
            child: Text(
              'Últimos Registros',
              textAlign: TextAlign.center,
              style: TextStyle(
                fontSize: 15,
                fontWeight: FontWeight.bold,
              ),),),),
          Padding(
            padding: const EdgeInsets.all(0.0),
            child: Tabladatos(sensor),
          ),
          Padding(
            padding: const EdgeInsets.all(0.0),
            child: Text(
              'Tendencia Semanal',
              textAlign: TextAlign.center,
              style: TextStyle(
                fontSize: 15,
```

```
            fontWeight: FontWeight.bold,),),),
        Padding(
          padding: const EdgeInsets.all(1.0), child: Weekten(sensor),),
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Text('grafico'),),],
      staggeredTiles: [
        StaggeredTile.extent(2, 30),
        StaggeredTile.extent(2, 200),
        StaggeredTile.extent(2, 30),
        StaggeredTile.extent(1, 133),
        StaggeredTile.extent(1, 133),
        StaggeredTile.extent(2, 30),
        StaggeredTile.extent(2, 110),
        StaggeredTile.extent(2, 30),
        StaggeredTile.extent(2, 78)
      ],);}}
```

Widget datalist.dart

```
var hourbase=hournowdate.format(now.subtract(new Duration(minutes:minact)));
```

Este segmento de código permite realizar una consulta a la base de datos por medio del backend y guardar los datos en una variable tipo List.

```
String url = urlh + '/api/sensor/dashboard/';
  Map<String, String> headers = {"Content-type": "application/json"};
  String jsondata = "{" +'"sensorname":"' +sensor+'","fecha1":"'+datenow+
      '","fecha2":"'+datenow+'","hora1":"'+ hourbase +'","hora2":"' +
      hournow +'"' +"}";
http.Response response=await http.post(url,headers: headers, body: jsondata);
  datasensores = json.decode(response.body);
  sensorData = datasensores['sensor'];
```

Debido a que los datos se actualizan periódicamente y como Flutter posee una estructura para manejar el estado de los Widgets y que cuando el usuario cambie de interfaz el Widget no presente errores de estado se implementó lo siguiente:

```
@override
  void initState() {
    _countdownTimer = Timer.periodic(Duration(seconds: tiempoact), (timer) {
      if (mounted) {
        setState(() {});
      }
    });
    super.initState();
  }
```

```
  void dispose() {
    _countdownTimer?.cancel();
    _countdownTimer = null;
    super.dispose();
  }
```

Los datos son guardados en una variable de tipo List en Flutter para el correcto funcionamiento del Widget se necesita utilizar un Widget de tipo FutureBuilder que permite analizar los datos y guardarlos en los Widgets hijos.

```
Card(
        child: FutureBuilder(
            future: listadogenerado(sensor),
            builder: (context, snapshot) {
              return Column(children: [
                Row(children: [
                  Expanded(
                      child: Text("     ppm",
                          style: TextStyle(fontWeight: FontWeight.bold))),
                  Expanded(
      child: Text(" mg/m^3",style: TextStyle(fontWeight: FontWeight.bold))),
                  Expanded(
      child: Text("AQI",style: TextStyle(fontWeight: FontWeight.bold))),
                  Expanded(
      child: Text("Hora",style: TextStyle(fontWeight: FontWeight.bold))),
                ]),
                Expanded(
                  child: Container(
                    child: ListView.builder(
                      itemExtent: 20.0,
                      scrollDirection: Axis.vertical,
                      shrinkWrap: true,
                      physics: ScrollPhysics(),
                      itemCount: sensorData == null ? 0 : sensorData.length,
                      itemBuilder: (BuildContext context, int index) {
                        return ListTile(
                          onTap: null,
                          title: Row(children: <Widget>[
                  Expanded(child:Text("${sensorData[index]["sensormq"]}",)),
                     Expanded(child: Text("${sensorData[index]["mqco"]}")),
                     Expanded(child: Text("${sensorData[index]["airqc"]}")),
                     Expanded(child: Text("${sensorData[index]["hora"]}")),
                          ]),);},),
                    padding: const EdgeInsets.all(.0),
                  ),),]);}),
        elevation: 12);}
```

Clase datagrafico.dart

La clase datagrafico permite formatear los datos en valores permisibles para que sean graficados y obtener el ultimo valor registrado por los sensores.

```
Map data, datasensores;
List sensorData;
List<double> valorsensor = [];
List<double> valoraq = [];
List<int> valoraqc = [];
double sensoractual = 0;
double aqactual = 0;
var airactual = 0;
Map<DateTime, double> datos = Map();
Future<void> datagrafico(String sensor) async {
  DateTime now = DateTime.now();
  var datenow = DateFormat('yyyy-MM-dd').format(now);
  var hournowdate = DateFormat().add_Hms();
  var hournow = hournowdate.format(now);
  debugPrint(hournowdate.format(now));
  var hourbase =
      hournowdate.format(now.subtract(new Duration(minutes: minact)));
  String url = urlh + '/api/sensor/dashboard/';
  Map<String, String> headers = {"Content-type": "application/json"};
  String jsondata = "{" + '"sensorname":"' +
sensor +'","fecha1":"' +datenow +'","fecha2":"' +datenow +'","hora1":"' +hourba
se +'","hora2":"' +hournow +'"' +"}";
  http.Response response;
  try {
    print(jsondata);
    response = await http.post(url,
        headers: headers,
        body: jsondata
    datasensores = json.decode(response.body);
  } catch (e) {
    throw Exception('Request Error:  $e');
  }
  sensorData = datasensores['sensor'];
  valorsensor = [];
  valoraq = [];
  //hora = [];
  valoraqc = [];
  datos = Map();
  print(sensorData.length);
  for (var i = 0; i < sensorData.length; i++) {
    valorsensor.add(sensorData[i]['mqco']);
```

```
    valoraq.add(sensorData[i]['sensormq']);
    valoraqc.add(sensorData[i]['airq']);
    var horap = DateFormat.Hms().parse(sensorData[i]['hora']);
    var fechap = DateFormat('yyyyy-MM-dd').parse(sensorData[i]['fecha']);
    var time = DateTime(fechap.year, fechap.month, fechap.day, horap.hour,
        horap.minute, horap.second);
    datos.update(time, (value) => sensorData[i]['sensormq'],
        ifAbsent: () => sensorData[i]['sensormq']);  }
  aqactual = valorsensor.elementAt(sensorData.length - 1);
  sensoractual = valoraq.elementAt(sensorData.length - 1);
  airactual = valoraqc.elementAt(sensorData.length - 1);}
```

Widget graficaprincipal.dart

```
AreaLineChart.fromDateTimeMaps([datos],[Colors.blueAccent[400]],
['ppm/tiempo'], tapTextFontWeight: FontWeight.bold,
 gradients: [Pair(Colors.red.shade200, Colors.blue.shade100)])
```

Widget Valor Actual

```
      Card(
                child: Column(
                  children: <Widget>[
                    Padding(
                      padding: const EdgeInsets.all(3.0),
                      child: Text(
                        'Sensor (CO)',
                        style: TextStyle(
                            color: Colors.black,
                            fontSize: 13,
                            fontWeight: FontWeight.bold),
                        textAlign: TextAlign.center,
                      ),),
                    Column(
                      children: <Widget>[
                        Padding(
                          padding: const EdgeInsets.all(18.0),
                          child: Text(
                            '$sensoractual ppm',
                            style: TextStyle(
                                color: Colors.black,
                                fontSize: 15,
                                fontWeight: FontWeight.bold),
                            textAlign: TextAlign.center,
                          ),),
                        Padding(
                          padding: const EdgeInsets.all(5.0),
                          child: Text(
```

```
                    '$aqactual mg/m^3',
                     style: TextStyle(color: Colors.black,
                         fontSize: 15,
                         fontWeight: FontWeight.bold),
                     textAlign: TextAlign.center,
                   ),),],),],),
          elevation: 13,
          borderOnForeground: true,
          color: Colors.black12,)
```

Función Air_Quality

Esta clase recibe los valores calculados de la calidad del aire desde el backend. La función AQICategory categoriza los valores calculados en color basados en (The World Air Quality Proyect, 2019).

```
AQICategory(var AQIndex) {
  var AQI = AQIndex;
  Color AQICategory;
  if (AQI == null) {
    AQICategory = Colors.black12;
  }
  if (AQI > 0 && AQI <= 50) {
    //AQICategory = "Bueno";
    AQICategory = Colors.green;
  } else if (AQI > 50 && AQI <= 100) {
    AQICategory = Colors.amberAccent[100];
    // AQICategory = "Moderado" as Colors;
  } else if (AQI > 100 && AQI <= 150) {
    //AQICategory = "Muy poco saludable para grupos sensibles";
    AQICategory = Colors.amberAccent[200];
  } else if (AQI > 150 && AQI <= 200) {
    // AQICategory = "Poco saludable";
    AQICategory = Colors.deepOrange;
  } else if (AQI > 200 && AQI <= 300) {
    //AQICategory = "Muy poco saludable";
    AQICategory = Colors.red[400];
  } else if (AQI > 300 && AQI <= 400) {
    //AQICategory = "Peligroso";
    AQICategory = Colors.red;
  } else if (AQI > 400 && AQI <= 500) {
    // AQICategory = "Peligroso";
    AQICategory = Colors.red;
  } else {
    // AQICategory = "Fuera de Rango";
    AQICategory = Colors.black12;
  }
  return AQICategory;
```

```
}
```

Función Textaqi

Categoriza el estado de la calidad del aire.

```
Textaqi(var AQIndex) {
  var AQI = (AQIndex);
  var AQICategory;
  if (AQI == null) {
    AQICategory = "";
  }
  if (AQI > 0 && AQI <= 50) {
    AQICategory = "Bueno";
  } else if (AQI > 50 && AQI <= 100) {
    AQICategory = "Moderado";
  } else if (AQI > 100 && AQI <= 150) {
    AQICategory = "Muy poco saludable para grupos sensibles";
  } else if (AQI > 150 && AQI <= 200) {
    AQICategory = "Poco saludable";
  } else if (AQI > 200 && AQI <= 300) {
    AQICategory = "Muy poco saludable";
  } else if (AQI > 300 && AQI <= 400) {
    AQICategory = "Peligroso";
  } else if (AQI > 400 && AQI <= 500) {
    AQICategory = "Peligroso";
  } else {
    AQICategory = "Fuera de Rango";
  }
  return AQICategory;
}
```

La función RangoAQI permite establecer los rangos de calidad del aire.

```
Rangoaqi(var AQIndex) {
  var AQI = (AQIndex);
  var AQICategory;
  if (AQI == null) {
    AQICategory = "";
  }
  if (AQI > 0 && AQI <= 50) {
    AQICategory = " 0 - 50";
  } else if (AQI > 50 && AQI <= 100) {
    AQICategory = " >50 - 100";
  } else if (AQI > 100 && AQI <= 150) {
    AQICategory = " >100 - 150";
  } else if (AQI > 150 && AQI <= 200) {
    AQICategory = " >150 - 200";
```

```
  } else if (AQI > 200 && AQI <= 300) {
    AQICategory = " >200 - 300";
  } else if (AQI > 300 && AQI <= 400) {
    AQICategory = " >300 - 400";
  } else if (AQI > 400 && AQI <= 500) {
    AQICategory = " >400 - 500";
  } else {
    AQICategory = "> 500";
  }
  return AQICategory;
}
```

Widget Índice Calidad del Aire

Muestra la información procesada para ser visualizada por el usuario de una forma más fácil de entender, este Widget cambia de color en función de la calidad del aire captado por el sensor.

```
Card(
      child: Column(
        children: <Widget>[
          Padding(
            padding: const EdgeInsets.all(2.0),
            child: Text(
              'Índice Calidad de Aire (CO)',
              style: TextStyle(
                  color: Colors.black,
                  fontSize: 13,
                  fontWeight: FontWeight.bold),
              textAlign: TextAlign.center,),),
          Column(
            children: <Widget>[
              Padding(padding: const EdgeInsets.all(18.0),
                child: Text(
                  //funcion retornara el tipo de calidad del aire como texto
                  Textaqi(airactual),
                  style: TextStyle(
                   color: Colors.black,
                      fontSize: 14,
                      fontWeight: FontWeight.w500),
                  textAlign: TextAlign.center,),),
              Padding(padding: const EdgeInsets.all(1.0),
                child: Text(
                  //funcion retornara el tipo de calidad del aire como texto
                  "$airactual",
                  style: TextStyle(
                      color: Colors.black,
                      fontSize: 17,
                      fontWeight: FontWeight.w500),
                  textAlign: TextAlign.center,),),
```

```
            Padding(
              padding: const EdgeInsets.all(1.0),
              child: Text(
               //aqui se mostrara los rangos de valores en funcion de retorno
                'Rango:' + Rangoaqi(airactual),
                style: TextStyle(
                    color: Colors.black38,
                    fontSize: 11,
                    fontWeight: FontWeight.bold),
                textAlign: TextAlign.center,),),],),],),
      elevation: 15,
      borderOnForeground: true,
      //según la calidad del aire
      color: AQICategory(airactual),);
```

Función listweek

Las variables se crearon de tipo Map para tener un mejor manejo de datos en los Widgets.

```
Map listweek(String day) {
  Map<String, String> daycontrol = Map();
  Map<String, String> dataday = Map();
  String daynow = DateFormat('EEEE').format(now);
  var monday = 1;
  dataday = Map();
  dataday.update(
      nameday(now.weekday), (value) => DateFormat('yyyy-MM-dd').format(now),
      ifAbsent: () => DateFormat('yyyy-MM-dd').format(now));
  while (now.weekday != monday) {
    now = now.subtract(new Duration(days: 1));
    dataday.update(
    nameday(now.weekday), (value) => DateFormat('yyyy-MM-dd').format(now),
        ifAbsent: () => DateFormat('yyyy-MM-dd').format(now));
  if (dataday.containsKey(day)) {
    daycontrol.update(day, (value) => dataday[day],
        ifAbsent: () => dataday[day]);
  } else {
    daycontrol = {};
  }
  return daycontrol;
}
```

Debido a que la comunicación es asincrónica con el servidor se implementó en el Widget individual de día que muestra la calidad del aire promedio un FuturerBuilder que da tiempo a la aplicación esperar los datos a ser cargados. Esta Widget permite valores de entrada del nombre del sensor y de día tomados de la función ListWeek.

```
Widget _card(String sensor, String dia, BuildContext context) {
    var diacontrol = '';
    var fechacontrol = '';
```

```
var valmedio = 0.0;
var valaqip = 0;
var valstate = '';
Color colcard = Colors.grey;
var daycontrol = listweek(dia);
if (daycontrol.containsKey(dia)) {
  diacontrol = dia;
  fechacontrol = daycontrol[dia];
  return card(daycontrol.containsKey(dia), fechacontrol);
} else {
  return card(daycontrol.containsKey(dia), fechacontrol);
} }
```

Widget Tendencia Semanal

El Widget card recibe el estado de tipo bool, permitiendo habilitar ese Widget para visualizar datos, con estos datos llama a la función AQICategory para colorear y categorizar el Widget dependiendo de los valores medidos por los sensores, en caso de que el día de la semana este fuera del rango pondrá el color gris. Si los valores están dentro del rango el usuario podrá seleccionar para acceder a un nuevo menú.

```
Widget card(bool estado, String fecha) {
    if (estado) {
      return Container(
        height: 40,
        child: FutureBuilder(
            future: datalent.dateten(sensor, fecha),
            builder: (context, snapshot) {
              if (snapshot.connectionState == ConnectionState.done) {
                debugPrint(datalent.listado['valordiap'].toString());
                return Card(
                  elevation: 12,
                  color: AQICategory(datalent.listado['valordiap']),
                  child: Align(
                    alignment: Alignment.center,
                    child: new GestureDetector(
                        onTap: () {
                          Navigator.push(context,
                              MaterialPageRoute(builder: (context) {
                            return Interfazgraficaday(sensor, fecha, dia)
                          }));}},
               child: _texto(Textaqi(datalent.listado['valoraqip'])))),),); }
              else {
                return Center(child: CircularProgressIndicator());
              }}),); } else {
      return Container(
        height: 40,
        child: Card(
            //poner funcion dias de la semana color deshabilitar
```

```
            color: Colors.black12,
            elevation: 1,
            child: new GestureDetector(
                onTap: () {
                  },
                child: _texto('     '))),); }}
```

Para la interfaz final de tendencia semanal se utilizó un Container que contiene a los Widget de cada día parar forma un solo Widget semanal.

```
Widget build(BuildContext context) {
   return Card(
     child: Column(
       children: <Widget>[
         Container(
           child: Row(
             children: <Widget>[
               Expanded(
                 child: Text(
                   'Lunes',
                   style: TextStyle(fontSize: 12),
                   textAlign: TextAlign.center,
                 ),),
               Expanded(
                 child: Text(
                   'Martes',
                   style: TextStyle(fontSize: 12),
                   textAlign: TextAlign.center,
                 ),),
               Expanded(
                 child: Text(
                   'Miércoles',
                   style: TextStyle(fontSize: 12),
                   textAlign: TextAlign.center,
                 ),),
               Expanded(
                 child: Text(
                   'Jueves',
                   style: TextStyle(fontSize: 12),
                   textAlign: TextAlign.center,
                 ), ),
               Expanded(
                 child: Text(
                   'Viernes',
                   style: TextStyle(fontSize: 12),
                   textAlign: TextAlign.center,
                 ),),
               Expanded(
```

```
                child: Text(
                  'Sábado',
                  style: TextStyle(fontSize: 12),
                  textAlign: TextAlign.center,
                ),),
              Expanded(
                child: Text(
                  'Domingo',
                  style: TextStyle(fontSize: 12),
                  textAlign: TextAlign.center,
                ),),],),),),
        Container(
          child: Row(
            children: <Widget>[
              Expanded(child: Carted(sensor, 'Lunes')),
              Expanded(child: Carted(sensor, 'Martes')),
              Expanded(child: Carted(sensor, 'Miércoles')),
              Expanded(child: Carted(sensor, 'Jueves')),
              Expanded(child: Carted(sensor, 'Viernes')),
              Expanded(child: Carted(sensor, 'Sabado')),
              Expanded(child: Carted(sensor, 'Domingo')),
            ],),),],),elevation: 12,); }}
```

Función listweek

Las variables se crearon de tipo Map para tener un mejor manejo de datos en los Widgets.

```
Map listweek(String day) {
  Map<String, String> daycontrol = Map();
  Map<String, String> dataday = Map();
  String daynow = DateFormat('EEEE').format(now);
  var monday = 1;
  dataday = Map();
  dataday.update(
      nameday(now.weekday), (value) => DateFormat('yyyy-MM-dd').format(now),
      ifAbsent: () => DateFormat('yyyy-MM-dd').format(now));
  while (now.weekday != monday) {
    now = now.subtract(new Duration(days: 1));
    dataday.update(
    nameday(now.weekday), (value) => DateFormat('yyyy-MM-dd').format(now),
        ifAbsent: () => DateFormat('yyyy-MM-dd').format(now));
  if (dataday.containsKey(day)) {
    daycontrol.update(day, (value) => dataday[day],
        ifAbsent: () => dataday[day]);
  } else {
    daycontrol = {};
  }
  return daycontrol;
```

```
}
```

Debido a que la comunicación es asincrónica con el servidor se implementó en el Widget individual de día que muestra la calidad del aire promedio un FuturerBuilder que da tiempo a la aplicación esperar los datos a ser cargados.

Esta Widget permite valores de entrada del nombre del sensor y de día tomados de la función ListWeek.

```
Widget _card(String sensor, String dia, BuildContext context) {
    var diacontrol = '';
    var fechacontrol = '';
    var valmedio = 0.0;
    var valaqip = 0;
    var valstate = '';
    Color colcard = Colors.grey;
    var daycontrol = listweek(dia);
    if (daycontrol.containsKey(dia)) {
      diacontrol = dia;
      fechacontrol = daycontrol[dia];
      return card(daycontrol.containsKey(dia), fechacontrol);
    } else {
      return card(daycontrol.containsKey(dia), fechacontrol);
    } }
```

Widget Tendecia Semanal

El Widget card recibe el estado de tipo bool, permitiendo habilitar ese Widget para visualizar datos, con estos datos llama a la función AQICategory para colorear y categorizar el Widget dependiendo de los valores medidos por los sensores, en caso de que el día de la semana este fuera del rango pondrá el color gris. Si los valores están dentro del rango el usuario podrá seleccionar para acceder a un nuevo menú.

```
Widget card(bool estado, String fecha) {
    if (estado) {
      return Container(
        height: 40,
        child: FutureBuilder(
            future: datalent.dateten(sensor, fecha),
            builder: (context, snapshot) {
              if (snapshot.connectionState == ConnectionState.done) {
                debugPrint(datalent.listado['valordiap'].toString());
                return Card(
                  elevation: 12,
                  color: AQICategory(datalent.listado['valordiap']),
                  child: Align(
                    alignment: Alignment.center,
                    child: new GestureDetector(
                        onTap: () {
                          Navigator.push(context,
```

```
                            MaterialPageRoute(builder: (context) {
                           return Interfazgraficaday(sensor, fecha, dia)
                      }));}},
              child: _texto(Textaqi(datalent.listado['valoraqip']))),),); }
             else {
                return Center(child: CircularProgressIndicator());
             }),); } else {
      return Container(
        height: 40,
        child: Card(
            //poner funcion dias de la semana color deshabilitar
            color: Colors.black12,
            elevation: 1,
            child: new GestureDetector(
                onTap: () {
                  },
                child: _texto('     '))),); }}
```

Para la interfaz final de tendencia semanal se utilizó un Container que contiene a los Widget de cada día parar forma un solo Widget semanal.

```
Widget build(BuildContext context) {
   return Card(
     child: Column(
       children: <Widget>[
         Container(
           child: Row(
             children: <Widget>[
               Expanded(
                 child: Text(
                   'Lunes',
                   style: TextStyle(fontSize: 12),
                   textAlign: TextAlign.center,
                 ),),
               Expanded(
                 child: Text(
                   'Martes',
                   style: TextStyle(fontSize: 12),
                   textAlign: TextAlign.center,
                 ),),
               Expanded(
                 child: Text(
                   'Miércoles',
                   style: TextStyle(fontSize: 12),
                   textAlign: TextAlign.center,
                 ),),
               Expanded(
                 child: Text(
```

```
            'Jueves',
            style: TextStyle(fontSize: 12),
            textAlign: TextAlign.center,
          ), ),
        Expanded(
          child: Text(
            'Viernes',
            style: TextStyle(fontSize: 12),
            textAlign: TextAlign.center,
          ),),
        Expanded(
          child: Text(
            'Sábado',
            style: TextStyle(fontSize: 12),
            textAlign: TextAlign.center,
          ),),
        Expanded(
          child: Text(
            'Domingo',
            style: TextStyle(fontSize: 12),
            textAlign: TextAlign.center,
          ),),],),),),
Container(
  child: Row(
    children: <Widget>[
      Expanded(child: Carted(sensor, 'Lunes')),
      Expanded(child: Carted(sensor, 'Martes')),
      Expanded(child: Carted(sensor, 'Miércoles')),
      Expanded(child: Carted(sensor, 'Jueves')),
      Expanded(child: Carted(sensor, 'Viernes')),
      Expanded(child: Carted(sensor, 'Sabado')),
      Expanded(child: Carted(sensor, 'Domingo')),
    ],),),],),elevation: 12,); }}
```

ANEXO VI: WIDGETS INTERFAZ DÍA ESPECIFICO

Widget Máximo y Mínimo

```
Widget build(BuildContext context) {
    return Container(
      child: FutureBuilder(
          future: data_max(sensor, fecha),
          builder: (context, snapshot) {
            return Container(
              child: Card(
                child: Column(
                  children: <Widget>[
                    Padding(
                      padding: const EdgeInsets.all(3.0),
                      child: Text(
                        'Valor Máximo (CO)',
                        style: TextStyle(
                            color: Colors.black,
                            fontSize: 12,
                            fontWeight: FontWeight.bold),
                        textAlign: TextAlign.center,
                      ),),
                    Column(
                      children: <Widget>[
                        Padding(
                          padding: const EdgeInsets.all(8.0),
                          child: Text(
                            '$sensormq ppm',
                            style: TextStyle(
                                color: Colors.black,
                                fontSize: 15,
                                fontWeight: FontWeight.bold),
                            textAlign: TextAlign.center,
                          ),),
                        Padding(
                          padding: const EdgeInsets.all(5.0),
                          child: Text(
                             '$horap',
                            style: TextStyle(
                                color: Colors.black,
                                fontSize: 15,
                                fontWeight: FontWeight.bold),
                            textAlign: TextAlign.center,
                          ),),],),],),
                elevation: 13,
                borderOnForeground: true,
                color: Colors.black12,
```

```
),); }),); }}.
```

Interfaz Día Medición Contaminación

Se utilizó StaggerdeGridView para tener una interfaz que ese pueda agregar más Widgets verticalmente además permite adaptar el tamaño de los Widget para ser visualizados de la mejor manera y con la opción de una flecha para volver a la interfaz principal.

```
@override
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(
        title: Text(
     'Variación de Contaminación (CO)' + '    ' + dia + '     ' + fechaf,
         //fecha.substring(8, 9),
         style: TextStyle(fontSize: 12),),),),
      body: StaggeredGridView.count(
        scrollDirection: Axis.vertical,
        shrinkWrap: true,
        physics: ScrollPhysics(),
        crossAxisCount: 2,
        crossAxisSpacing: 2,
        mainAxisSpacing: 3,
        padding: EdgeInsets.symmetric(horizontal: 15, vertical: 2),
        children: <Widget>[
          Padding(
            padding: const EdgeInsets.all(1.0),
            child: Graficadia(sensor, fecha, dia),),
          Padding(
            padding: const EdgeInsets.all(0.0),
            child: Text(
              'Valores Mínimos y Máximos (CO)',
              textAlign: TextAlign.center,
              style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
              ), ), ),
          Padding(
            padding: const EdgeInsets.all(0.0),
            child: Cardmin(sensor, fecha)),
          Padding(
              padding: const EdgeInsets.all(0),
              child: Cardmax(sensor, fecha)),
        ],
        staggeredTiles: [
          StaggeredTile.extent(2, 500),
          StaggeredTile.extent(2, 25),
          StaggeredTile.extent(1, 105),
```

```
  StaggeredTile.extent(1, 105),
],)); }}
```

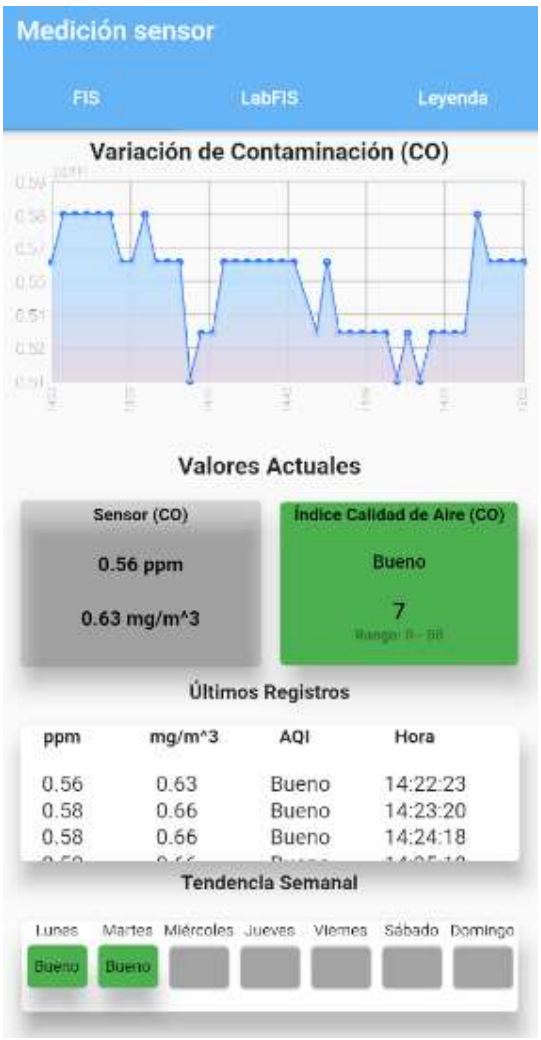
ANEXO VII: MANUAL APLICACIÓN MOVIL

Aplicación (Usuario)

Este apartado muestra las interfaces y funcionamiento de la aplicación.

Acceso a la aplicación

La aplicación debe ser descargada y posteriormente instalada en los dispositivos, al acceder a la aplicación se despliega la interfaz principal.



Menú de Navegación

La aplicación permite navegar entre los diferentes sensores que están conectados y visualizar los valores emitidos por los sensores, además en el apartado Leyenda categoriza los valores de calidad del aire.

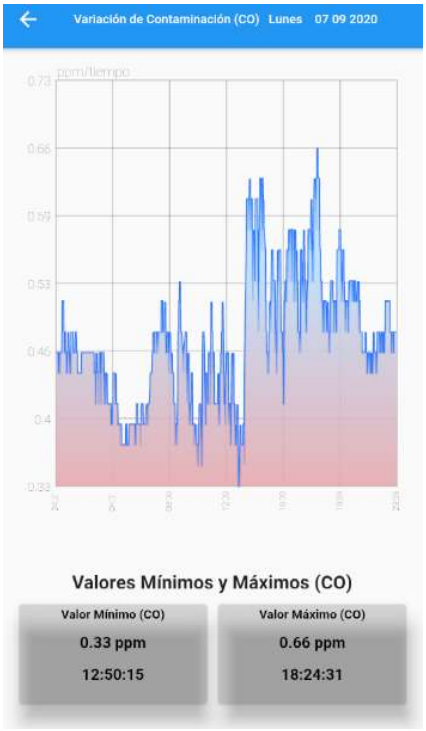
En la interfaz principal nos muestra la gráfica de los datos medidos por un sensor de los últimos cuarenta minutos, el usuario podrá seleccionar un punto y poder ver más detallado los valores de dicho punto.

Este segmento permite al usuario visualizar el último dato medido por el sensor y permite observa la calidad del aire actual.

El usuario podrá visualizar los datos de los últimos 40 minutos y desliarse entre los datos que están en forma de lista.

La aplicación permitirá visualizar la calidad del aire de todos los días hábiles de la semana y el usuario podrá acceder al día de la semana para obtener mayor detalle del día.

Esta interfaz permite visualizar en la cabecera a detalle el día de la semana seleccionado, la gráfica correspondiente a todos los valores captados en el día, y los valores máximos y mínimos del día.



Dentro del menú de navegación en Leyenda se muestra los valores categorizados de la medición de la calidad del aire.

ANEXO VIII: TABLA DE DATOS SENSORES PROYECTO (27/08/2020-27/10/2020) Y REMMAQ (01/02/2020-13/03/2020)

Enlace al documento en formato digital: https://bit.ly/2XplpqG

ANEXO IX: Repositorio Backend y Frontend

Enlace al repositorio de github: https://bit.ly/38qchrQ

ANEXO X: CONFIGURACIÓN GATEWAY SMART-LAB