ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE SISTEMAS

UNIDAD DE TITULACIÓN

IMPLEMENTACIÓN DE UN PROTOTIPO IOT PARA LA MONITORIZACIÓN DE LA CALIDAD DEL AIRE EN AMBIENTES INTERNOS

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

ALEX RICARDO AGUACONDO TAPIA

alex.aguacondo@epn.edu.ec

DOMÉNICA CAMILA JIMÉNEZ SUPE

domenica.jimenez@epn.edu.ec

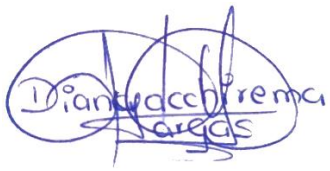
DIRECTORA: DIANA CECILIA YACCHIREMA VARGAS. PhD.

diana.yacchirema@epn.edu.ec

2023

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Alex Ricardo Aguacondo Tapia y Doménica Camila Jiménez Supe, bajo mi supervisión.



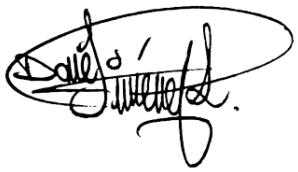
Diana Cecilia Yacchirema Vargas, PhD.

DIRECTOR

DECLARACIÓN DE AUTORÍA

Nosotros, **ALEX RICARDO AGUACONDO TAPIA** y **DOMÉNICA CAMILA JIMÉNEZ SUPE**, Declaro solemnemente que el trabajo presentado aquí es resultado de nuestro propio esfuerzo y que no ha sido utilizado previamente para obtener ningún título o calificación profesional. Además, confirmamos que hemos consultado las referencias bibliográficas que se incluyen en este documento.

Asimismo, mediante esta declaración, cedemos los derechos de propiedad intelectual de este trabajo a la Escuela Politécnica Nacional, de acuerdo con lo estipulado por la Ley de Propiedad Intelectual, su Reglamento y las normas institucionales vigentes.



Alex Ricardo Aguacondo Tapia

Doménica Camila Jiménez Supe

DEDICATORIA

Dedico este trabajo de tesis a mi madre, quien ha sido mi apoyo absoluto a lo largo de mi vida y me han brindado las herramientas necesarias para alcanzar mis metas. A mi padre, su bendición a lo largo de mi vida me protege y me lleva por el camino del bien. El amor, paciencia y constante motivación de ambos han sido fundamentales en este camino académico. También quiero dedicar este logro a mi hermano, quien ha estado a mi lado en cada etapa de mi formación. Este trabajo es un testimonio de gratitud hacia todos aquellos que han sido parte de mi vida y han contribuido a mi desarrollo personal y académico.

Alex

DEDICATORIA

Está tesis está dedicada a mis padres, quienes me han brindado su amor incondicional y su constante apoyo en cada paso que he dado. Sin su aliento y confianza, este logro no sería posible. También quiero expresar mi profundo agradecimiento a mis hermanas, que con su sabiduría, conocimiento y dedicación han sido una fuente constante de inspiración y motivación a lo largo de mi vida.

Doménica

AGRADECIMIENTOS

Quisiera expresar mi más sincero agradecimiento a todas las personas que me han apoyado y ayudado en la realización de esta tesis. Ha sido toda una empresa, pero gracias al apoyo y aliento de muchas personas, he logrado completar este proyecto junto a Doménica.

En primer lugar, quiero agradecer a mi familia por su amor incondicional y su constante apoyo. Han sido mi mayor fuente de motivación y me han brindado el respaldo emocional necesario para enfrentar los desafíos que se presentaron durante este proceso. Agradezco a mi madre, Silvia, por su dedicación y sacrificio para brindarme una educación de calidad. Quiero expresar mi profundo agradecimiento a mi padre difunto, Carlos, cuyo amor y apoyo incondicional me han guiado y motivado. Y a mi hermano, Carlos, por su invalorable apoyo y dedicación a lo largo del desarrollo de esta tesis.

También quiero expresar mi gratitud a Doménica que me ha acompañado en este viaje académico. Sus palabras de aliento, discusiones estimulantes y colaboraciones han sido invaluables para el desarrollo de este proyecto.

No puedo dejar de mencionar a nuestra tutora de tesis, quien nos brindó orientación, conocimientos y consejos expertos a lo largo de todo el proyecto. Su guía nos permitió enfocar y mejorar nuestro trabajo.

Por último, pero no menos importante, deseo agradecer a Dios por su amor, sabiduría y guía en todo momento. Sin Su ayuda divina, fortaleza y protección, no hubiera sido capaz de superar los desafíos y obstáculos que encontré en el camino. Le agradezco por las bendiciones recibidas y por estar a mi lado durante todo este proceso.

Alex

AGRADECIMIENTOS

Me gustaría expresar mi profundo agradecimiento a todas las personas que han contribuido de alguna manera en la realización de este proyecto de tesis.

En primer lugar, quiero agradecer a mi familia por su amor, comprensión y constante apoyo durante toda mi carrera universitaria. A mi madre, Yolanda, por su sacrificio y dedicación que han sido fuente de inspiración a lo largo de mi vida. A mi padre, Jaime, quien ha sido un faro de luz en mi vida y que me ha guiado en todo momento, impulsándome a alcanzar mis metas y crecer como persona. A mis hermanas, Gabriela y Verónica, gracias por su paciencia infinita, por escucharme en mis momentos de duda y por brindarme su sabiduría en cada decisión importante que he tomado. La confianza que han tenido en mí ha sido una fuente de motivación para alcanzar mis metas.

Agradezco a mi compañero de tesis, Alex, con quien he compartido momentos de aprendizaje y desafíos a lo largo de la carrera y en el desarrollo de esta tesis. También quiero agradecer a nuestra tutora, por su guía y sus valiosos consejos, su orientación ha sido fundamental para el presente proyecto.

Por último, quiero agradecer a mis amigos, Alexander y Jeferson, su amistad y apoyo constante han hecho de esta etapa académica una experiencia inolvidable. Sin el apoyo y contribución de todas estas personas, este logro no sería posible.

Doménica

ÍNDICE DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

Existe un alto porcentaje de la población que pasa la mayor parte de su tiempo en ambientes internos realizando actividades de ocio, trabajo o estudio. Estas personas están expuestas a los altos niveles de contaminación que se generan en ambientes internos que no poseen una correcta ventilación. Con el objetivo de solucionar este problema se propone la implementación de un prototipo IoT que se encargue de la monitorización de la calidad del aire dentro de los ambientes internos que envía alertas cuando el aire esté viciado.

El presente proyecto se enmarca en el diseño, implementación, aplicación y validación de un prototipo IoT para la medición de la calidad del aire en ambientes internos mediante la utilización de tecnologías emergentes como IoT y protocolos de comunicación M2M (Machine to Machine) de bajo consumo de recursos. La solución propuesta permite la monitorización de los parámetros de CO2, PM2.5, temperatura y humedad, así como el desarrollo de una aplicación web. Para el desarrollo del proyecto se utilizaron dos metodologías que son la Investigación constructivista de Oyegoke junto con algunas prácticas de Scrum.

El prototipo IoT implementado dentro de un ambiente familiar, adicionalmente se validó su funcionabilidad y su usabilidad. La funcionabilidad se validó al mostrar los parámetros monitorizados dentro de una aplicación web. Y la usabilidad, fue validada mediante encuestas de conformidad realizadas a los usuarios que habitaban el entorno familiar donde se implementó el prototipo IoT. Gracias a esto, se pudo demostrar que el prototipo IoT es una alternativa económica y funcional para medir y alertar a los usuarios acerca de la calidad del aire dentro de los ambientes internos en los que se encuentren.

Palabras clave: Prototipo IoT, Calidad del Aire, Internet de las Cosas, Ambientes Internos.

ABSTRACT

There is a high percentage of the population that spends most of its time in indoor environments performing leisure, work, or study activities. These people are exposed to high levels of pollution generated in indoor environments that do not have proper ventilation. To solve this problem, we propose the implementation of an IoT prototype that is responsible for monitoring the air quality inside indoor environments and sends alerts when the air is foul.

The present project is framed in the design, implementation, application, and validation of an IoT prototype for the measurement of air quality in indoor environments using emerging technologies such as IoT and M2M (Machine to Machine) communication protocols with low resource consumption. The proposed solution allows the monitoring of CO2, PM2.5, temperature, and humidity parameters, as well as the development of a web application. For the development of the project two methodologies were used which are Oyegoke Constructivist Research together with some Scrum practices.

The IoT prototype implemented within a familiar environment, additionally its functionality and usability were validated. Functionality was validated by displaying the monitored parameters within a web application. And usability was validated by means of compliance surveys conducted with the users living in the family environment where the IoT prototype was implemented. Thanks to this, it was possible to demonstrate that the IoT prototype is an economical and functional alternative to measure and alert users about air quality within the indoor environments where they are.

Keywords: IoT Prototype, Air Quality, Internet of Things, Indoor Environments.

1. INTRODUCCIÓN

1.1. Planteamiento del problema

De acuerdo con la Organización Mundial de la Salud (OMS), la polución atmosférica representa una de las principales preocupaciones medioambientales para la salud pública a nivel global. Muy al contrario de lo que suele pensarse, la contaminación del aire no solo se encuentra en ambientes externos. La realidad es que existe contaminación de dos a cinco veces mayor en ambientes internos que en externos, ya que no existe un flujo correcto del mismo y puede contener algunos contaminantes nocivos, por ejemplo, monóxido de carbono (CO), dióxido de azufre (SO2) y plomo (Pb). El dióxido de nitrógeno (NO2), el ozono (O3), la emisión de partículas (PM) y algunas otras mezclas naturales también se consideran un riesgo genuino para el bienestar [1].

Además, las personas pasan la mayor parte del tiempo en su hogar y trabajo, es decir, pasan aproximadamente un 80% del tiempo en ambientes internos [2], expuestos a altos niveles de contaminación que pueden causar enfermedades tales como: asma, enfermedad pulmonar obstructiva crónica, enfermedad coronaria, derrame cerebral o cáncer de pulmón [3].

A raíz de la pandemia de COVID-19, una considerable cantidad de individuos se vio obligada a adoptar la modalidad de trabajo a distancia [4], por lo que están más propensas a permanecer en ambientes internos llegando a una exposición de contaminantes nocivos del 100% en algunos casos. Para mantener unos estándares de calidad del aire en ambientes interiores es preciso monitorizar los diferentes contaminantes presentes en estos ambientes. Considerando que, en la actualidad, la modalidad de teletrabajo se ha mantenido permanente para varios trabajadores [5], monitorizar la calidad del aire en ambientes internos cobra más relevancia.

Para abordar esta cuestión en particular en el marco de este proyecto, se plantea la ejecución un prototipo de internet de las cosas (IoT, por sus siglas en inglés *Internet of Things*) que permita medir la calidad del aire en ambientes internos, mediante la utilización de diferentes sensores, cuyos datos recopilados serán transmitidos utilizando protocolos de comunicación como Cola de Mensajes Transporte de Telemetría (MQTT, por sus siglas en inglés Message Queing Telemetry Transport), que puede ser utilizado para máquina a máquina (M2M, por sus siglas en inglés machine to machine) a dispositivos Sistemas En un Chip (SoC, por sus siglas en inglés *System On a Chip*) que son dispositivos de baja potencia y costo.

1.2. Objetivos

1.2.1. Objetivo General

Implementar un prototipo IoT para la monitorización de la calidad del aire en ambientes internos.

1.2.2. Objetivos Específicos

- Diseñar el prototipo IoT utilizando dispositivos *constrained devices* y dispositivos SoC para la monitorización de la calidad del aire.
- Programar los *scripts* que permitan la obtención de la información proveniente de los sensores y el envío de mensajes correo electrónico por protocolos IoT M2M livianos.
- Programar una aplicación web que permita mostrar los datos monitorizados en tiempo real.
- Validar el prototipo IoT implementado mediante la usabilidad.

1.3. Marco Teórico

1.3.1. Calidad del aire en ambientes internos

Se estima que las personas pasan más del 90% del tiempo dentro de edificios [6] como viviendas, oficinas, escuelas, guarderías, edificios públicos, centros sanitarios u otros edificios públicos y privados, por lo que la calidad del aire en ambientes internos (IAQ, por sus siglas en inglés *Indoor Air Quality)*, es un determinante esencial de la vida sana y del bienestar de las personas [7].

Durante más de 150 años se han realizado estudios y se ha determinado que la IAQ es fundamental para la salud del ser humano, es por esto, que mantener una IAQ adecuada es esencial para minimizar los impactos negativos en la salud. La IAQ puede verse afectada por diferentes factores como: calidad del aire exterior, actividades en el interior, emisiones intrínsecas en el interior, ventilación, etc. En particular, se ha establecido que la causa de la contaminación del aire interior se debe a la concentración de contaminantes como las partículas (PM), los compuestos orgánicos volátiles (COV), el ozono (O3) y el monóxido de carbono (CO) que pueden llegar a estar mucho más concentradas que en el exterior, causando un gran impacto a la salud.

Por ejemplo, los hospitales funcionan las 24 horas del día, por lo que no tienen tiempo de inactividad para reducir sus niveles de contaminación es por esto por lo que se han realizado estudios en Estados Unidos y se ha determinado que el malestar presentado por el personal de la salud como dolores de cabeza, fatiga, sequedad e irritación de ojos y piel se debe a una IAQ deficiente. Debido a esto la cantidad de estudios realizados sobre la IAQ en hospitales se ha duplicado del 2015 al 2020 según un estudio realizado por la UFP Energy, FP-ENAS y Universidad Fernando Pessoa, como se muestra en la Figura 1; demostrando la creciente relevancia de investigar la calidad del aire en espacios interiores, influyendo de manera significativa en el ámbito de la salud [8].

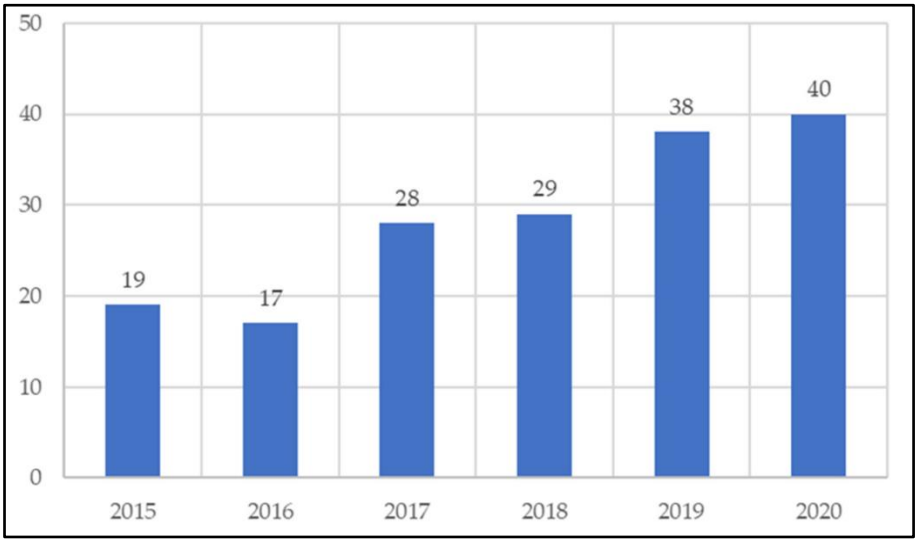


Figura 1. Incremento de estudios relacionados con la IAQ (2015-2020) *[8]*.

1.3.2. Índices de la Calidad del Aire

Se han desarrollado diversos métodos en el pasado por investigadores y agencias que determinan los AQI por siglas en inglés *Air Quality Index*, sin embargo, no existe un método universalmente aceptado. Cada uno de los métodos que calculan el AQI funcionan de distinta manera, algunos consideran diferentes tipos de contaminantes y las funciones con las que los calculan difieren de un índice a otro, pero, sus usos previstos convergen, siendo la identificación de las zonas de mala calidad del aire y la notificación pública de la gravedad de la exposición a la mala calidad del aire [9]. Dentro de la investigación realizada para esta tesis se estudiaron varios índices de la calidad del aire. A continuación, se explicarán los índices más utilizados:

1.3.2.1. *Air Quality Index System of U.S. Environmental Protection Agency* (EPAQI)

Este índice fue desarrollado por *Environmental Protection Agency* (EPA) y se emplea para determina la calidad del aire a corto y largo plazo. Utiliza los cinco contaminantes más comunes que son: ozono (O3), dióxido de nitrógeno (NO2), material particulado (PM2.5 y PM10), monóxido de carbono (CO) y dióxido de sulfuro (SO2). Los limites o puntos de quiebre para la concentración de estos gases son definidos por la EPA.

1.3.2.2. *Common Air Quality Index* (CAQI)

Fue desarrollado por el Citeair project con el objetivo de presentar de manera comprensible la situación de la calidad del aire en las ciudades de Europa. Tiene la capacidad de mostrar la calidad del aire casi en tiempo real. El índice final es el de mayor valor de los subíndices de cada componente. Se maneja con una escala de cinco niveles.

1.3.2.3. *New Air Quality Index* (NAQI)

Fue propuesto por Bishoi *et al*. Está basado en EPAQI, por lo que usa un factor de análisis de los principales contaminantes: CO, NO2, O3, PM y SO2. Su diferencia con el índice propuesto por EPA es que el primer factor causará la variación más alta de AQI. El segundo contribuirá con menos varianza que el primero pero más que el tercer factor y así sucesivamente.

1.3.2.4. *Pollution Index* (PI)

Fue desarrollado por Cannistraro, *et al*. Para comunicar a los ciudadanos de manera simple el estado de la calidad del aire en el área urbana de la ciudad de Nápoles. Este índice utiliza el promedio aritmético de los subíndices pertenecientes a los dos contaminantes más críticos. Se expresa mediante un rango del 1 al 7, donde el valora más alto representa una contaminación ambiental más alta [9].

1.3.3. Índice de la Calidad del Aire en Ambientes internos (IAQI)

Para hablar de un IAQI por sus siglas en inglés *Indoor Air Quality Index*, es necesario entender que todos los contaminantes que se evalúan con los IAQs no se encontrarán en la misma medida en un ambiente interno. Y que tampoco existe un índice especializado en cada ambiente interno ya que pueden encontrarse diferentes contaminantes en mayor o menor medida. Debido a esto, existen varias aproximaciones a un IAQI que pueda ser utilizado en

diferentes ambientes internos y que solo considere los contaminantes y/o las condiciones que más afecten a la salud o conformidad de una persona dentro de diferentes ambientes internos.

Al realizar la investigación de un IAQI para esta tesis se encontró uno llamado Índice Compuesto, por su nombre en inglés *Composite Index*. Este considera dos índices, el AQI y el *Comfort Index*. A este índice Compuesto de aquí en adelante se lo denominará como IAQCI por sus siglas en inglés *Indoor Air Quality & Comfort Index*, y se explicará con mayor detalle cada uno de sus componentes en los siguientes apartados.

1.3.4. Indoor Air Quality & Comfort Index (IAQCI)

Este índice, propuesto por Kim et al. [10] fue creado debido a la necesidad de obtener una medida que corrobore la mejora de la calidad del aire gracias a la instalación de jardines inteligentes. Este índice refleja los parámetros de temperatura y humedad, y un AQI en el ambiente interno. Adicionalmente, cuenta con una división de cinco grados que será explicado junto a su fórmula dentro de las siguientes secciones.

1.3.4.1. EPAQI

Es el AQI más utilizado en diferentes partes del mundo por ser un índice que presenta la cantidad de contaminación de una forma fácil de entender. Adicionalmente, este índice ha sido modificado varias veces para adaptarse a los lugares donde será empleado [11]. Debido a su fácil comprensión y adaptación, en esta tesis se ha seleccionado este índice para calcular la calidad del aire en ambientes internos.

Este índice utiliza los principales contaminantes más comunes en ambientes externos que son: material particulado (PM10 y PM2.5), monóxido de carbono (CO), ozono (O3), dióxido de azufre (SO2) y dióxido de nitrógeno (NO2). El valor de contaminación se obtiene considerando el resultado más alto entre todos los contaminantes monitorizados. EPAQI utiliza tablas de concentración límite basadas en los Estándares Nacionales de Calidad del Aire Ambiental del inglés *National Ambient Air Quality Standards* (NAAQS) [9] [12].

Este estudio se realizará en ambientes internos y se debe tener en cuenta que las principales fuentes de contaminación para ambientes internos son el polvo y el dióxido de carbono (CO2) emitido por las personas que ocupan el espacio [13]. Debido a esto, los contaminantes escogidos para calcular el índice de calidad del aire son: PM2.5 y CO2. A continuación, se explicará porque estos contaminantes son dañinos para la salud.

CO2 es el principal gas de efecto invernadero generado por los humanos, cuando se está expuesto a altas concentraciones, este gas puede afectar al cuerpo humano causando trastornos metabólicos como la del síndrome del edificio enfermo.

PM2.5 son las partículas en el ambiente que tienen un diámetro de 2.5 µm. Cuando se tiene una exposición continua este contaminante puede causar enfermedades respiratorias, demenciales e incluso varios estudios han determinado que aumenta la probabilidad de mortalidad temprana.

Para determinar el EPAQI, se emplea la siguiente ecuación de interpolación lineal (1) para cada contaminante, específicamente para PM2.5 y CO2. Posteriormente, se utiliza la ecuación (2) para obtener el valor final del índice de calidad del aire. Las concentraciones límite empleadas en este estudio se pueden consultar en la Tabla 1 y fueron propuestas en el artículo "*Evaluation of IAQ Management Using an IoT-Based Indoor Garden*" basándose en el estándar de la EPA y los resultados de estudios epidemiológicos que se refieren al efecto de contaminantes individuales en la salud humana. Adicionalmente la Tabla 2 muestra los colores de clasificación [9] [10].

$$I_p = \frac{(I_{HI} - I_{LO})}{BP_{HI} - BP_{LO}} (C_P - BP_{LO}) + I_{LO} \quad (1)$$

Donde:

I_P = Índice del contaminante P
C_P = Concentración redondeada del contaminante P
BP_{HI} = Punto de quiebre mayor o igual que C_P
BP_{LO} = Punto de ruptura menor o igual que C_P
I_{HI} = Valor AQI correspondiente a BP_{HI}
I_{LO} = Valor AQI correspondiente a BP_{LO}

$$EPAQI = Max(I_p) \quad (2)$$

Donde:

EPAQI = Índice de la calidad del aire de EPA
I_P = Índices de los contaminantes

La Tabla 1 muestra el cálculo integrado de los índices de los componentes PM2.5 y CO2, esta información se utilizará para medir la calidad en ambientes internos.

Tabla 1. Cálculo integrado del índice de calidad del aire por cada componente en ambientes internos de EPA *[10]*

Componente	Bueno	Moderado	Poco saludable	Malo	Peligroso
	A (0-20)	B (21-40)	C (41-60)	D (61-80)	E (81-100)
PM2.5 (µg/m3)	0-15	16-35	36-50	51-80	>81
CO2 (ppm)	0-450	451-700	701-1000	1001-3000	>3000

El índice de la calidad del aire de EPA se dividió en cinco niveles. En una escala del 0 al 100 donde una puntuación más alta refleja una calidad de aire inferior.

Tabla 2. Clasificación del índice de la calidad del aire de EPA *[10]*

Clasificación	Índice	Puntuación	Color
A	Bueno	0-20	Azul
B	Moderado	21-40	Verde
C	Poco saludable	41-60	Amarillo
D	Malo	61-80	Anaranjado
E	Peligroso	81-100	Rojo

Para calcular el IAQCI es necesario, a más del cálculo de EPAQI, calcular el índice de confort para él cual se utilizarán los datos de temperatura y humedad, se explica a continuación:

1.3.4.2. Índice de confort (Temperatura y humedad)

El índice de confort, también llamado temperatura aparente o índice de calor, se refiere a la percepción que experimenta el cuerpo humano en relación con la temperatura cuando se combinan la humedad relativa y la temperatura del aire. Esta es una consideración crucial para el bienestar del cuerpo humano, ya que cuando el cuerpo se calienta demasiado, comienza a sudar para regular su temperatura. La transpiración permite que el sudor se evapore, lo que contribuye al proceso de enfriamiento del cuerpo. Sin embargo, si el sudor no puede evaporarse adecuadamente, el cuerpo tiene dificultades para regular su temperatura. A medida que el sudor se evapora del cuerpo, contribuye a reducir la temperatura corporal de manera efectiva. Sin embargo, cuando el contenido de humedad en el ambiente (es decir, la humedad relativa) es alto, la tasa de evaporación del sudor disminuye. En otras palabras, en un entorno húmedo, el cuerpo humano experimenta una sensación de mayor calor. Por otro lado, en condiciones de baja humedad relativa, el aumento en la tasa de transpiración puede

generar una sensación de frescura. Existe una relación directa entre la temperatura del aire, la humedad relativa y el índice de calor, lo que significa que a medida que la temperatura del aire y la humedad relativa aumentan (o disminuyen), el índice de calor también aumenta (o disminuye) en consecuencia [14].

Existen tablas que determinan el índice de confort dependiendo de la ubicación geográfica. Para esta investigación, se ha utilizado como referencia la tabla publicada por ITIEFFE [15]. Esta tabla está dividida dentro de seis categorías que determinan el nivel de conformidad con una temperatura y humedad relativa. Sin embargo, para la aplicación de la fórmula del IAQCI se necesita consistencia de acuerdo con las clasificaciones entre los índices del EPAQI y el índice de calor.

Para lo cual se ha adaptado la tabla del ITIEFFE, estableciendo otros márgenes mediante parametrización, como se muestran en la Tabla 3. En función de los valores de humedad relativa y temperatura, se ubican los valores dentro de la Tabla 3 y se obtiene el índice de confort respectivo.

Tabla 3. Índice de confort

Humedad Relativa (%)																	
Temperatura (°C)	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**
15	0,0	0,0	0,0	0,0	0,0	1,6	1,6	3,3	4,9	4,9	6,6	8,2	8,2	9,8	11,5	11,5	13,1
16	1,6	1,6	1,6	1,6	1,6	3,3	4,9	6,6	6,6	8,2	9,8	9,8	11,5	13,1	13,1	14,8	16,4
17	3,3	3,3	3,3	3,3	4,9	4,9	6,6	8,2	9,8	11,5	11,5	13,1	14,8	16,4	16,4	18,0	19,7
18	4,9	4,9	4,9	4,9	6,6	8,2	9,8	11,5	11,5	13,1	14,8	16,4	18,0	19,7	19,7	21,3	23,0
19	6,6	6,6	6,6	6,6	8,2	9,8	11,5	13,1	14,8	16,4	18,0	19,7	21,3	23,0	23,0	24,6	26,2
20	8,2	8,2	8,2	9,8	11,5	13,1	14,8	16,4	18,0	19,7	21,3	23,0	24,6	24,6	26,2	27,9	29,5
21	9,8	9,8	9,8	11,5	13,1	14,8	16,4	18,0	19,7	21,3	23,0	24,6	26,2	27,9	29,5	31,1	32,8
22	11,5	11,5	11,5	14,8	16,4	18,0	19,7	21,3	23,0	24,6	26,2	27,9	29,5	31,1	34,4	36,1	37,7
23	13,1	13,1	14,8	16,4	18,0	19,7	23,0	24,6	26,2	27,9	29,5	31,1	32,8	34,4	37,7	39,3	41,0
24	14,8	14,8	16,4	18,0	21,3	23,0	24,6	26,2	29,5	31,1	32,8	34,4	36,1	39,3	41,0	42,6	44,3
25	16,4	16,4	19,7	21,3	23,0	26,2	27,9	29,5	31,1	34,4	36,1	37,7	41,0	42,6	44,3	45,9	47,5
26	18,0	19,7	21,3	23,0	26,2	27,9	31,1	32,8	34,4	37,7	39,3	41,0	44,3	45,9	47,5	49,2	52,5
27	19,7	21,3	24,6	26,2	27,9	31,1	32,8	36,1	37,7	41,0	42,6	44,3	47,5	49,2	50,8	54,1	55,7
28	21,3	23,0	26,2	29,5	31,1	34,4	36,1	39,3	41,0	42,6	45,9	47,5	50,8	52,5	55,7	57,4	60,7
29	23,0	26,2	29,5	31,1	34,4	36,1	39,3	41,0	44,3	45,9	49,2	52,5	54,1	57,4	59,0	60,7	63,9
30	26,2	27,9	31,1	34,4	36,1	39,3	42,6	44,3	47,5	49,2	52,5	55,7	57,4	60,7	62,3	67,2	68,9
31	27,9	31,1	34,4	36,1	39,3	42,6	45,9	47,5	50,8	54,1	55,7	59,0	60,7	65,6	68,9	70,5	73,8
32	31,1	34,4	36,1	39,3	42,6	45,9	47,5	50,8	54,1	57,4	59,0	62,3	67,2	68,9	72,1	75,4	78,7
33	32,8	36,1	39,3	42,6	45,9	49,2	50,8	54,1	57,4	60,7	62,3	67,2	70,5	73,8	77,0	80,3	83,6
34	36,1	39,3	42,6	45,9	49,2	50,8	54,1	57,4	60,7	65,6	68,9	72,1	75,4	78,7	82,0	85,2	88,5
35	39,3	42,6	44,3	47,5	50,8	54,1	57,4	60,7	65,6	68,9	72,1	77,0	80,3	83,6	86,9	90,2	93,4
36	41,0	44,3	47,5	50,8	54,1	57,4	60,7	65,6	70,5	73,8	77,0	80,3	85,2	88,5	91,8	95,1	100,0

El índice de confort se dividió en cinco grados teniendo en cuenta la elaboración del índice de confort y seguridad, que se puede observar en la Tabla 4.

Tabla 4. Clasificación del índice de confort del aire interior *[10]*

Clasificación	Índice	Puntuación	Color
A	Muy confortable	0-20	Azul
B	Confortable	20.1-40	Verde
C	Incómodo para un grupo sensible	40.1-60	Amarillo
D	Incómodo	60.1-80	Anaranjado
E	Muy incómodo	80.1-100	Rojo

A continuación, se detalla como calcular el IAQCI utilizando los valores de EPAQI e índice de confort calculados previamente.

1.3.5. IAQCI

En base al estudio propuesto por Kim et al, los índices se clasifican en cinco grados, esta clasificación es utilizada tanto para el índice de confort (Tabla 4) y el índice EPAQI (Tabla 3). La clasificación para IAQCI se visualiza en la Tabla 5. Para el cálculo del índice, se utiliza la fórmula para el cálculo del IAQCI (índice de IAQ + índice de confort) [10]. Utilizando la escala de 1 a 100 puntos, se determina que las puntuaciones más altas representan una menor calidad de la IAQ y de los índices de confort, según lo indicado por la temperatura, la humedad y la seguridad.

Tabla 5. Clasificación del IAQCI *[10]*

Clasificación	Índice	Puntuación	Color
A	Bueno	0-20	Azul
B	Moderado	21-40	Verde
C	Poco saludable	41-60	Amarillo
D	Malo	61-80	Anaranjado
E	Peligroso	81-100	Rojo

Para el cálculo de este índice se utilizan los índices ya mencionados anteriormente, el EPAQI y el índice de confort:

$$IAQCI = \omega_1 \times EPAQI + \omega_2 \times índice\ de\ confort \quad (4)$$

El EPAQI, compuesto por dos componentes, PM2,5 y CO2, y el índice de confort que se compone de la temperatura y la humedad, se calculan en una escala de 100 puntos. Los componentes ω1 y ω2 pueden expresarse según las siguientes ecuaciones cuando las

ponderaciones del índice IAQ y del índice de confort se calculan como 1:1 en la determinación del IAQCI:

$$IAQCI = 0.5 \times EPAQI + 0.5 \times índice\ de\ confort \quad (5)$$

1.3.6. Internet de las Cosas

Es una red que posibilita la conexión y comunicación entre múltiples dispositivos de forma interconectada. Dentro de los dispositivos hay sensores y actuadores [16]. Estos dispositivos desempeñarán un papel fundamental al capturar información en tiempo real en entornos internos, lo cual será de gran ayuda para la toma de decisiones [17].

El IoT facilita la implementación de soluciones de monitoreo del aire rentables al medir los cambios en la calidad del aire, a través de sensores que pueden ser ubicados en distintos espacios como oficinas, departamentos u hogares en general, con el objetivo de tomar medidas que ayuden a mejorar la calidad de vida de las personas más sensibles a la contaminación [18].

1.3.7. Protocolos de aplicación utilizados en IoT

Los protocolos son indispensables para establecer una comunicación entre los dispositivos que conforman el sistema IoT, en la siguiente sección se analizarán los protocolos más comúnmente utilizados con el fin de seleccionar el más adecuado para el proyecto en el Capítulo 2.

1.3.7.1. *Constrained Application Protocol* (CoAP)

El protocolo CoAP fue creado en el año 2010 por el Grupo de Trabajo de Ingeniería de Internet (IETF, por sus siglas en inglés de *Internet Engineering Task Force*). Es un protocolo web de transferencia basado en la arquitectura de Transferencia de Estado Representacional (REST, por sus siglas en inglés *Representational State Transfer*) que posee dos capas. La primera integra el Protocolo de Datagrama de Usuarios (UDP, por sus siglas en inglés *User Datagram Protocol*), para el intercambio de mensajes y la segunda capa está diseñada para transportar mensajes *request/response*.

Las principales características de CoAP son que es un protocolo ligero y de baja velocidad ya que tiene menor sobrecarga y complejidad de encabezados y métodos que otros protocolos de aplicación. Los mensajes se codifican en formato binario simple y tienen soporte opcional

de solicitudes *uni-cast* y *multicast* con UDP, a pesar de esto el uso de UDP reduce la confiabilidad del protocolo [19].

1.3.7.2. Advanced Protocol of Message Queue Server (AMQP)

AMQP es un protocolo de aplicación que fue creado en el año 2003 por John O'Hara. Este protocolo tiene dos implementaciones AMQP versión 0.9.1 y versión 1.0, siendo la última versión la estandarizada por el consorcio OASIS, por sus siglas en ingles *Organization for the Advancement of Structured Information Standards.* La arquitectura de AMQP soporta el paso de mensajes tanto de *request/response* como de *publish/subscribe* y la comunicación para el intercambio de mensajes se realiza a través de TCP.

AMQP ofrece fiabilidad e interoperabilidad por su amplia gama de funciones como colas confiables, publicación y suscripción de mensajes, transacciones flexibles, enrutamiento y seguridad. Es un protocolo flexible ya que puede soportar diferentes esquemas de comunicación e integra seguridad con el uso de la Capa de Seguridad y Autenticación Simple (SASL, por sus siglas en inglés *Simple Authentication and Security Layer*) y la Seguridad de la Capa de Transporte (TLS, por sus siglas en inglés *Transport Layer Security*), pero es un protocolo pesado en cuanto a recursos de red y requiere mucha potencia computacional por parte de uno o varios nodos IoT [20].

1.3.7.3. *Message Queuing Telemetry Transport* (MQTT)

MQTT es un protocolo que fue creado en 1999 por Andy Stanford-Clark de IBM y Arlen Ninpper de Arcom. La designación "MQ" en MQTT se deriva de MQ Series, un producto desarrollado por IBM que brinda soporte al transporte de telemetría MQ. OASIS la estandarizó en su versión 3.1. Fue diseñado para la comunicación "máquina a máquina", su arquitectura es *publish/subscribe* y es independiente de los datos a transmitir así que su formato puede ser binario, texto, XML o JSON. Utiliza el protocolo TCP para transporte.

Sus características por destacar son que es un protocolo adecuado para dispositivos con recursos limitados y condiciones de red no ideales ya que tiene un tamaño de paquete de datos con sobrecarga baja, esto lo convierte en una opción muy conveniente para IoT. Al transmitir mensajes mediante *publish/subscribe* el remitente y el receptor de los datos no se conocen ya que existe un intermediario [21], facilitando el intercambio de información, ya que, ambos clientes no necesitan conocer sus Protocolos de Internet (IP, por sus siglas en inglés *Internet Protocol*) y solo es necesario conocer la dirección del intermediario.

1.3.8. Arquitecturas utilizadas en IoT

Existen diferentes arquitecturas propuestas por varios investigadores. Sin embargo, el modelo básico dentro de la literatura es la arquitectura de tres capas que fue cambiando con el tiempo hasta llegar al nivel de detalle de siete capas [22]. En este apartado se analizarán las arquitecturas más utilizadas en IoT y, posteriormente, La elección de la arquitectura se realizará en el Capítulo 2 y se basará en la evaluación de diversos factores, con el objetivo de seleccionar aquella que se adapte de manera óptima para cumplir los objetivos específicos de este proyecto.

1.3.8.1. Arquitecturas de tres capas

Es la arquitectura más básica y suele ser utilizado como modelo de referencia para otras arquitecturas. A continuación, se resume su funcionamiento [23]:

- Capa de percepción: Aquí se encuentran diferentes dispositivos electrónicos (sensores, placas, etc.) e incluso código fuente que permiten recopilar datos para posteriormente ser transmitidos.
- Capa de red: Enruta y transmite los datos a través de la red. Las tecnologías normalmente utilizadas por esta capa son Wi-Fi, *Bluetooth*, 3G/LTE, *Zigbee*, *Lora*, etc.
- Capa de aplicación: Es el puente entre las aplicaciones y los clientes, permitiendo visualizar los datos antes recopilados. Aquí se muestra la información monitorizada por los sensores.

1.3.8.2. Arquitecturas de cinco capas

Con el objetivo de ampliar el nivel de detalle de la arquitectura de tres capas se descompuso las responsabilidades y funcionalidades de la capa de aplicación, añadiendo dos capas adicionales que son servicio y operación, además de las tres capas que se mantienen de la arquitectura de tres capas. La funcionalidad de cada una de las capas se explica a continuación [23]:

- Capa de servicio: Permite el manejo de datos, facilitando el uso de herramientas o plataformas que permitan el procesamiento, computación y análisis de los datos obtenidos en capas anteriores.

- Capa de operación: Garantiza la Calidad del Servicio o QoS, por sus siglas en inglés *Quality of Service*. Permitiendo la supervisión, control y evaluación en tiempo real de varios parámetros de IoT.

1.3.8.3. Arquitecturas de siete capas

Esta arquitectura es el incremento de los modelos más tradicionales de capas, y proporciona mayor confiabilidad debido a la modularización que implementa. Esto se debe a que integra dos nuevas capas, que son la capa de abstracción y de cómputo, además de las cinco capas que se mantienen de la arquitectura de cinco capas. Gracias a esto, se distribuye de mejor manera las diferentes funcionalidades que puede integrar una solución IoT. La funcionalidad de cada una de las siete capas se detalla a continuación [23]:

- Capa de abstracción: Estandariza el lenguaje y los protocolos que se utilizarán para evitar problemas de comunicación.
- Capa de cómputo: Realiza el análisis y procesamiento de datos utilizando en su mayoría tecnologías emergentes como *machine learning, big data, cloud computing*, *deep learning*, etc.

1.3.9. *Scrum*

Scrum es una metodología de desarrollo de software que garantiza el desarrollo de proyectos en períodos de tiempo ajustados [24]. Debido a su adaptabilidad, rapidez, flexibilidad y efectividad es la metodología ágil que permite entregar un valor significativo de forma rápida. Además, implementa diferentes diagramas y modelos para la implementación de la solución para el presente proyecto [25]; por lo que está alineado al modelo constructivista de Oyegoke.

El desarrollo en equipo dentro de este marco de trabajo se organiza en incrementos llamados *sprints*. La coordinación entre los equipos se realiza a través del *backlog*, donde se apilan los requerimientos. El *Product Owner* es responsable de determinar los elementos que serán implementados en los próximos *sprints*. Mientras el *Scrum Master* brinda soporte al equipo para que trabaje de forma eficaz.

Scrum resulta muy conveniente al momento de trabajar con tecnologías diferentes como lo son IoT y las aplicaciones web, ya que los requisitos son diversos y cada miembro del equipo se encontrará resolviendo distintos problemas en paralelo [24].

1.3.9.1. Proceso *Scrum*

El proceso Scrum, representado en la Figura 2, para el desarrollo de proyectos se lleva a cabo en ciclos de trabajo breves y enfocados, generalmente de una a cuatro semanas de duración. El objetivo es entregar resultados completos en cada iteración. El resultado final es un producto entregado al cliente de manera eficiente, cumpliendo con los requisitos establecidos.

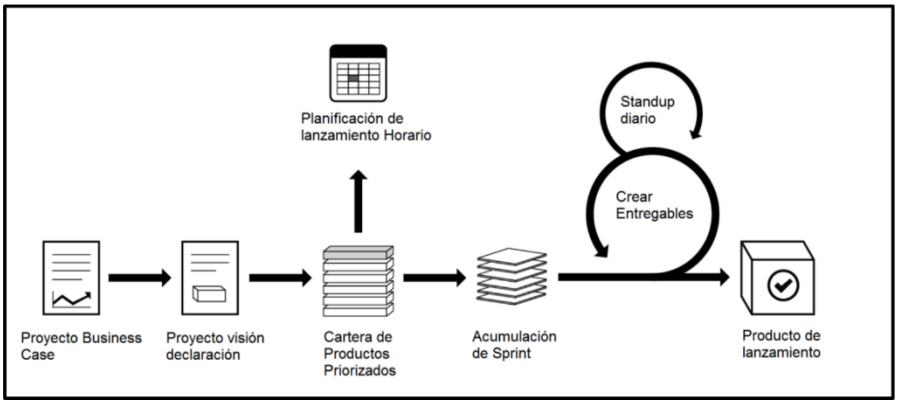


Figura 2. Flujo de *scrum* para un sprint [24]

1.3.9.2. Eventos *Scrum*

El *Sprint* sirve como un marco que abarca todos los demás eventos dentro de *Scrum*. Los eventos son actividades de tiempo limitado que proporcionan oportunidades para la colaboración, la inspección, y la adaptación a lo largo del proyecto. Cada evento proporciona una ocasión estructurada para examinar y ajustar los artefactos *Scrum*, garantizando la transparencia necesaria. El propósito de estos eventos en *Scrum* es establecer un ritmo constante y reducir la necesidad de reuniones adicionales fuera del marco definido de *Scrum*.

1.3.9.2.1. *Sprint*

Un *Sprint* en *Scrum* es una iteración de tiempo limitado durante el cual el Equipo *Scrum* trabaja para entregar un incremento de producto potencialmente entregable. Es un período definido, por lo general oscila entre una y cuatro semanas, en el que el equipo en colaboración completa un conjunto de elementos de trabajo priorizados del *Product Backlog*. El *Sprint* proporciona un marco estructurado para planificar, ejecutar, revisar y adaptar el trabajo del equipo, permitiendo la entrega incremental de valor y promoviendo la retroalimentación y los ajustes regulares. Al centrarse en la consecución de un objetivo específico del *Sprint*, el equipo pretende ofrecer resultados tangibles y mejorar continuamente sus procesos.

1.3.9.2.2. *Sprint Planning*

Es un evento de colaboración en *Scrum* que marca el comienzo de un *Sprint*. Durante este evento, el Equipo *Scrum*, formado por el *Product Owner* y el Equipo de Desarrollo, se reúne para determinar qué trabajo se llevará a cabo en el próximo *Sprint*.

El objetivo de la Planificación del *Sprint* es definir el objetivo del *Sprint* y crear un plan para alcanzarlo. El evento suele constar de dos partes:

1. Primera parte: El *Product Owner* presenta al Equipo de Desarrollo los componentes más prioritarios incluidos en el *Product Backlog*. Se discuten los elementos, se aclaran las dudas y se proporciona la información adicional necesaria.
2. Segunda parte: El Equipo de Desarrollo, con la orientación del *Product Owner*, selecciona los componentes del *Product Backlog* que pueden comprometerse a entregar dentro del *Sprint*. Desglosan los componentes escogidos en tareas más pequeñas y manejables y calculan el esfuerzo necesario para cada tarea.

1.3.9.2.3. *Daily Scrum*

Es una reunión corta, de tiempo limitado que se celebra todos los días durante un *Sprint* en *Scrum*. Es un evento clave donde el Equipo de Desarrollo sincroniza su trabajo y colabora para asegurar el progreso hacia el objetivo del *Sprint*. El propósito del *Daily Scrum* es facilitar la transparencia, la investigación y el acomodo dentro del equipo.

Además del *Daily Scrum*, los desarrolladores tienen la flexibilidad de reunirse a lo largo del día para tener discusiones más detalladas sobre la adaptación o replanificación del resto del trabajo del *Sprint*.

1.3.9.2.4. *Sprint Review*

Es un evento de colaboración en *Scrum* que tiene lugar al final de cada *Sprint*. Es una oportunidad para que el Equipo *Scrum*, las partes interesadas y los clientes inspeccionen el trabajo realizado durante el Sprint y proporcionen retroalimentación. El *Sprint Review* se centra en demostrar el incremento del producto potencialmente entregable y la recopilación de información valiosa.

1.3.9.2.5. *Sprint Retrospective*

Es un evento dedicado en el marco de *Scrum* que tiene lugar al final de cada *Sprint*. Proporciona una oportunidad para que el Equipo *Scrum* reflexione sobre su colaboración, los procesos y el rendimiento durante el *Sprint* recién terminado. El *Sprint Retrospective* se centra en la identificación de fortalezas, debilidades y mejoras potenciales para el próximo *Sprint*.

1.3.9.3. Artefactos *Scrum*

Los artefactos Scrum simbolizan compromiso o valía. Están diseñados para maximizar la transparencia de la información relevante. Facilitan la comprensión de todos los procesos que se llevarán a cabo en el proyecto para todo el Equipo Scrum.

1.3.9.3.1. *Product Backlog*

En *Scrum*, el *Product Backlog* es un artefacto esencial que representa una lista dinámica y ordenada de todos los requisitos, características, mejoras y correcciones que definen el producto. Sirve como la única fuente de información veraz para el Equipo *Scrum* y orienta los esfuerzos de desarrollo a lo largo de todo el proyecto. Dentro de esta lista se muestra la funcionalidad, requisito, mejoras y correcciones que puedan aplicarse en futuras entregas.

1.3.9.3.2. *Sprint Backlog*

Se denominan juntos *Sprint Backlog*, al objetivo del Sprint (por qué), los elementos del *Product Backlog* seleccionados para el *Sprint* (qué), más el plan para entregarlos (cómo).

1.3.9.3.3. Incremento

En *Scrum*, un Incremento es un artefacto clave que representa la totalidad de todos los elementos incluidos en el *Product Backlog* terminados y potencialmente entregables al final de un *Sprint*. Es el trabajo completado por el Equipo de Desarrollo durante el *Sprint* y sirve como una demostración tangible de los progresos realizados hacia el objetivo general del proyecto.

1.3.9.4. Equipo *Scrum*

La unidad básica en *Scrum* es un equipo pequeño de personas conocido como Equipo Scrum. El Equipo *Scrum* está compuesto por un *Scrum Master*, un *Product Owner* y los desarrolladores. Los equipos *Scrum* se caracterizan por ser interfuncionales y autogestionados, cada uno de los miembros posee las destrezas necesarias para definir el valor y deciden el rol que cada quién desempeñará a lo largo del proyecto.

1.3.9.4.1. Desarrolladores

Son las personas del Equipo *Scrum* que poseen habilidades específicas acordes con el dominio de trabajo, también están comprometidas a crear cualquier aspecto de un Incremento utilizable cada *Sprint*. Tienen la responsabilidad de crear un plan para el *Sprint*, conocido como el *Sprint Backlog*, y de adaptar dicho plan diariamente para avanzar hacia el objetivo del *Sprint*.

1.3.9.4.2. *Product Owner*

Es el individuo encargado de optimizar el valor del producto que se obtiene del esfuerzo del Equipo *Scrum*. Su función principal es representar al cliente y garantizar la entrega de un producto de valor al final de cada iteración.

1.3.9.4.3. *Scrum Master*

El *Scrum Master* es responsable de facilitar la aplicación efectiva del marco *Scrum*. El *Scrum Master* sirve como un líder-servidor y ayuda al equipo a abrazar y adoptar los principios y prácticas de *Scrum*. Ayuda a todos a concebir la teoría y la ejecución de *Scrum* [26].

2. METODOLOGÍA

El desarrollo de este proyecto se llevó a cabo mediante la aplicación de la metodología de investigación constructivista propuesta por Oyegoke (2011), complementada con el Marco de Trabajo *Scrum* para el desarrollo de software. Las historias de usuario se fragmentaron en las distintas etapas correspondientes a la investigación constructivista de Oyegoke (2011). Las pruebas técnicas de cada Sprint se encuentran detalladas en el ANEXO I.

2.1. Investigación constructivista de Oyegoke (2011)

La investigación constructiva se utiliza para definir y resolver problemas, del mismo modo se utiliza para mejorar un sistema o desempeño existente [27]. El constructivismo tiene como objetivo organizar la experiencia que un individuo tiene respecto a las cosas, formalizando las interpretaciones de los fenómenos en reglas generales. Y la forma en que un investigador aborda un problema es con una preconcepción de este, así como una posible solución, teniendo en cuenta que la realidad es una construcción interpretada por el observador [25]. Además, el modelo constructivista ha sido aplicado exitosamente, facilitando obtener resultados sobre la efectividad de medición sobre proyectos de TI (Tecnologías de la Información) en Colombia, como lo menciona Ariza D. [25].

Según Oyegoke, la investigación constructivista se inicia a partir de un problema específico, cuyas posibles soluciones son examinadas en la literatura existente, de modo que el investigador cuente con una referencia documentada de las propuestas planteadas hasta el momento [25]. Dentro del presente proyecto, se construyeron los diagramas respectivos utilizando esta metodología para resolver el problema planteado en el Capítulo 1.

Este modelo se compone de seis fases de las cuales solo se tomarán las cuatro primeras fases, esto debido a que están relacionadas con el alcance del proyecto. A continuación, se detallan las cuatro primeras fases: Planteamiento del problema, Diseño del prototipo, Implementación del prototipo y Validación del prototipo. Las fases tienen una funcionalidad definida y están relacionadas con sus fases principal y predecesora.

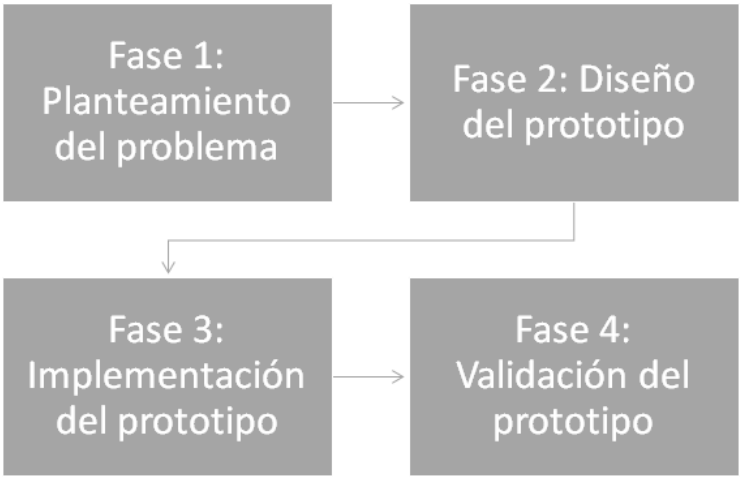


Figura 3. Fases de la metodología para el proyecto

2.1.1. Fases del Modelo Constructivista de Oyegoke

Fase 1: Consistió en realizar un análisis de la situación actual tomando como base el problema a resolver, que es la calidad del aire en ambientes cerrados. Este planteamiento ha sido analizado en el capítulo previo (Capítulo 1). En consecuencia, se obtuvo el IAQCI, encargado de medir la calidad del aire en ambientes cerrados. Mismo que será calculado dentro del prototipo *IoT* implementada a partir de la arquitectura de tres capas.

Fase 2: Se definió qué protocolo, nodos IoT y dispositivos SoC fueron adecuados para la implementación del prototipo IoT. Después, se diseñó el prototipo con sus respectivas conexiones y nodos. Tras el análisis del problema planteado, utilizando la metodología *Scrum* se estableció una lista de requerimientos tanto para aplicación web como para el nodo IoT.

Fase 3: En la Fase 2 se muestra cómo implementar el nodo IoT y su integración al dispositivo SoC, por lo que se utilizó el diseño del prototipo IoT definido para la presente fase. Para esto, se programaron los scripts que permiten la obtención de información proveniente de los sensores y el envío de mensajes a los actuadores. Adicionalmente, se despliega un aplicativo web que permite la visualización de los datos en tiempo real, así como un histórico de los datos almacenados. También se integra un *plugin* existente que enviará notificaciones (correos electrónicos) en tiempo real cuando los parámetros de contaminación monitorizados sobrepasen los umbrales establecidos. A continuación, siguiendo con la metodología *Scrum*, se dividen las actividades de desarrollo y se asignan los sprints. Finalmente, se realiza la conexión entre el nodo IoT, el dispositivo SoC y la aplicación web.

Fase 4: Mediante pruebas de funcionamiento del prototipo IoT se valida que el prototipo IoT cumple con los objetivos planteados. Las pruebas se evidencian en el Capítulo 3 de esta tesis.

2.3. Prototipo IoT

Dentro de este apartado se especifica el diseño e implementación del prototipo.

2.3.1. Diseño del prototipo

Para el diseño de prototipo se evaluaron las distintas arquitecturas IoT mencionadas en el Capítulo 1 y se seleccionó la que mejor se ajustó a los objetivos del proyecto. Para ello se consideraron los siguientes criterios que se muestran en la Tabla 6.

Tabla 6. Comparativa entre las arquitecturas de tres, cinco y siete capas

	Arquitecturas IoT		
Criterios	**3 capas**	**5 capas**	**7 capas**
Simplicidad en el diseño	X	X	
Simplicidad en la implementación	X		
Seguridad mejorada		X	X
Latencia reducida	X		
Escalabilidad	X	X	X
Relación coste-eficacia	X		
Adaptabilidad con tecnologías emergentes	X		

Adaptabilidad con requerimientos cambiantes	X		

Como se puede apreciar en la Tabla 6, la arquitectura de tres capas se destaca por su simplicidad, eficiencia y adaptabilidad. Para este proyecto, se eligió la arquitectura de tres capas debido a que se adapta mejor a las tecnologías y al alcance propuesto. Su estructura sencilla permite simplificar varios procesos, lo que facilita su implementación y comprensión.

La Figura 4 muestra cómo se ha adaptado la arquitectura de tres capas a la solución del problema planteado en esta tesis y, cómo cada una de las capas interactúa entre sí.

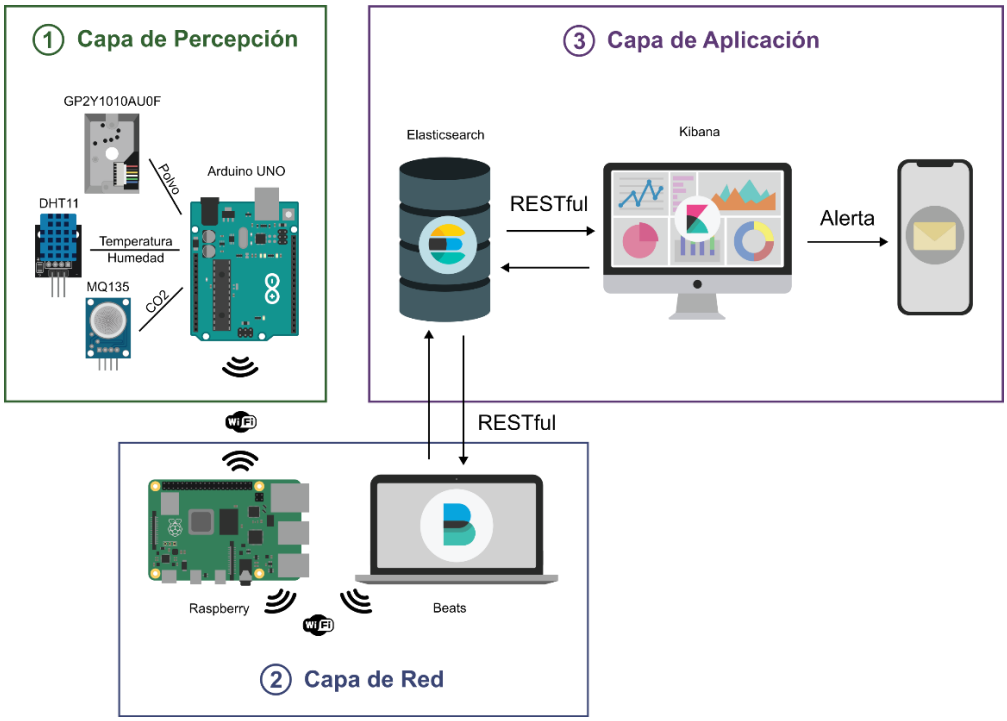


Figura 4. Arquitectura del prototipo IoT

A continuación, se explicará brevemente el funcionamiento de cada capa:

1. **La capa de percepción:** Comprende el nodo IoT, constituido por la placa Arduino UNO y los tres sensores, colocados estratégicamente en el ambiente interno que miden los parámetros monitorizados, que son: el polvo (GP2Y1010AU0F), el CO2 (MQ135), temperatura y humedad (DHT11). Adicionalmente, la capa de percepción es la encargada de enviar los datos monitorizados a la capa de red, para ellos utiliza una conexión WiFi.

2. **La capa de red:** Representada por la Raspberry Pi, que es un dispositivo SoC, actúa como intermediario para facilitar la comunicación de todos los componentes del prototipo. Para ello, la Raspberry Pi actúa como bróker que ha sido configurando, utilizando MQTT. Adicionalmente, la capa de red se encarga de enviar la información de los sensores a la capa de aplicación, para ello normaliza los datos utilizando un cliente Filebeat utilizando un ordenador.
3. **La capa de aplicación:** En esta capa se realiza todo el almacenamiento y enriquecimiento de los datos para poder ser posteriormente visualizados a través de un aplicativo web. Adicionalmente, esta capa se encarga del envío de notificaciones y alertas en caso de que alguno de los parámetros monitorizados exceda algún umbral establecido.

2.3.1.1. Capa de percepción

Para obtener los diferentes parámetros utilizados por el IAQCI, se utilizaron los siguientes sensores:

- Humedad y Temperatura: Dentro de un ambiente interno si estos parámetros no están dentro de umbrales de conformidad, pueden causar problemas como problemas respiratorios, de circulación, enfermedades relacionadas con el calor, etc. Para obtener estos parámetros, se utilizó el sensor DHT11.
- Partículas finas: El polvo dentro de ambientes internos, puede causar irritaciones respiratorias, reacciones alérgicas y asmáticas, etc. Para obtener este parámetro se utilizó el sensor GP2Y1010AU0F.
- CO2: Este parámetro es un indicador de la eficacia de la ventilación dentro de un ambiente interno, ya que, unos niveles elevados de CO2 sugieren que el intercambio y la circulación de aire fresco pueden ser insuficientes, lo que puede provocar la acumulación de otros contaminantes en el interior. Este parámetro fue obtenido mediante el sensor MQ135.

Estos sensores fueron integrados y conectados a una placa Arduino UNO, que es la parte central de un nodo IoT, que es la encargada de gestionar todos los datos obtenidos por los diferentes sensores. El diagrama de pines se detalla en la Figura 5.

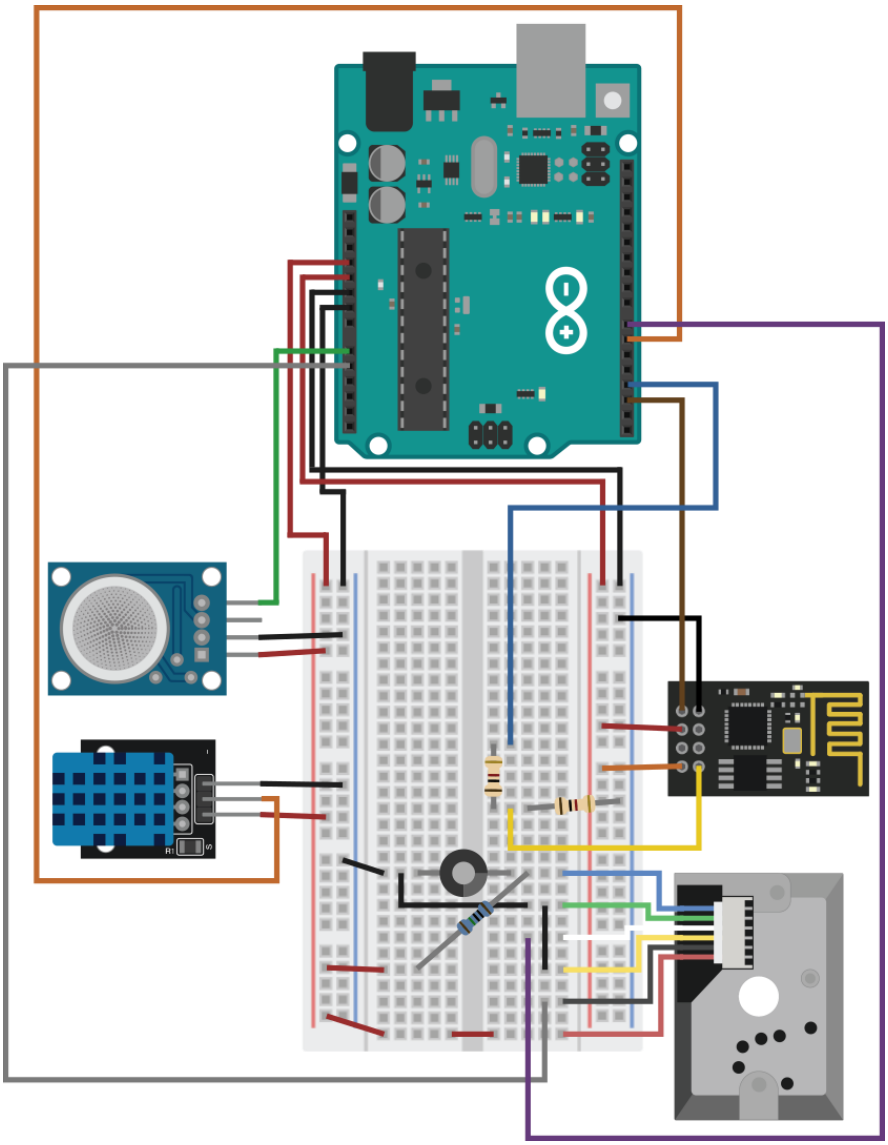


Figura 5. Diagrama de conexión del nodo IoT

2.3.1.1.1. Nodo IoT

Un dispositivo IoT es una pieza de hardware que incluye sensores inalámbricos, software, actuadores y dispositivos informáticos. Su función es recopilar, procesar y transmitir datos. Esta información ayudará en la toma de decisiones como, por ejemplo, reducir costos, aumentar eficiencia, etc. Para el presente proyecto, el nodo IoT es un dispositivo diseñado específicamente para monitorizar la calidad del aire en espacios internos. Este nodo está compuesto por la placa central, en este caso el Arduino UNO, y los sensores MQ135, DHT11 y GP2Y1010AU0F, los cuales están conectados a la placa central de forma respectiva.

2.3.1.1.1.1. Arduino UNO

Ver Figura 6. Es una placa que incorpora el microcontrolador ATmega328P, es la placa más recomendada para iniciarse en cualquier proyecto de electrónica al ser la más robusta y tener la mayor cantidad de documentación. Tiene 14 pines digitales y 6 entradas analógicas que permiten tener una mayor cantidad de módulos conectados en simultaneo [28].

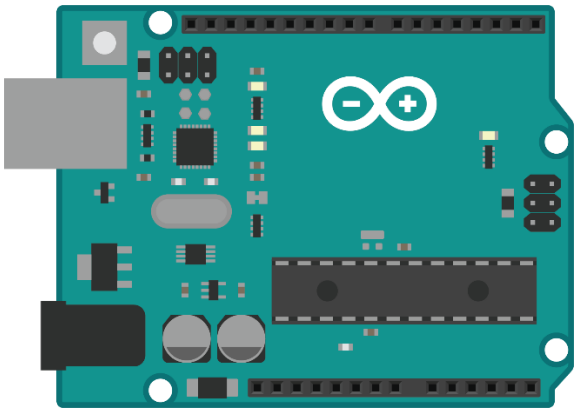


Figura 6. Placa Arduino UNO

2.3.1.1.1.2. MQ135

Ver Figura 7. Este sensor está fabricado con una capa sensible al dióxido de estaño, un electrodo de medición y un calentador que permite obtener las condiciones óptimas. Tiene 6 pines, 4 para la señal y 2 para la corriente. Este sensor permite detectar gases como NH3, NOx, alcohol, benceno, humo, CO2, etc. [29]

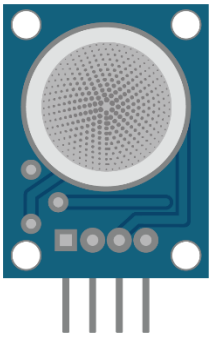


Figura 7. Sensor MQ135

2.3.1.1.1.3. DHT11

Ver Figura 8. Este sensor posee una memoria OTP interna donde se almacenarán los datos de calibración para que las medidas de temperatura y humedad sea más precisas. Tiene 3 pines, 2 para la corriente y 1 para la señal de salida digital. Es un sensor muy ligero de solo 0.1 oz lo que lo hace ideal para robots y sistemas de monitoreo de ambiente [30].

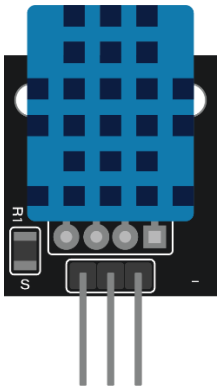


Figura 8. Sensor DHT11

2.3.1.1.1.4. GP2Y1010AU0F

Ver Figura 9. Es un sensor mide la densidad del polvo mediante un diodo emisor de infrarrojos y un fototransistor. Es muy eficiente al medir partículas muy finas como humo de cigarrillo. Tiene 6 pines, 4 son para alimentar la luz del infrarrojo y el sensor; los 2 restantes son para las señales de salida [31].

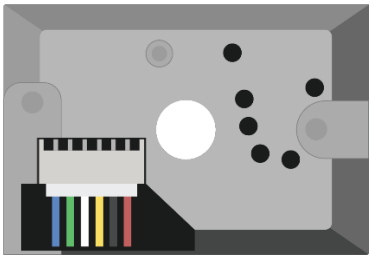


Figura 9. Sensor GP2Y1010AU0F

2.3.1.2. Capa de Red

Para transportar los datos desde la capa de percepción a la capa de red se seleccionará un protocolo IoT que se adapta a las necesidades del prototipo.

2.3.1.2.1 Selección de los protocolos IoT

Por sus características, IoT no puede basarse en un solo protocolo de mensajería como es el caso de la web que utiliza como estándar el protocolo HTTP. Existen diversos protocolos para los diferentes sistemas IoT, los cuatro protocolos emergentes y ampliamente aceptados son MQTT, CoAP, AMQP y HTTP [32]. Tomando en cuenta que uno de los parámetros más importantes para valorar el rendimiento de una red IoT es la latencia en la transferencia de mensajes y que el protocolo HTTP posee desventajas como un encabezado con un tamaño que agrega sobrecarga de procesamiento y una alta latencia de red, no será incluido en la comparación para este trabajo [33]. La comparación se realizará en la Tabla 7.

Tabla 7. Comparativa entre los protocolos MQTT, CoAP y AMQP *[34]*.

	Arquitecturas IoT		
Criterios	**MQTT**	**CoAP**	**AMQP**
Protocolo de transporte	TCP	UDP	TCP
Arquitectura Publish/Subscribe	X	X	X
Seguridad	TLS/SSL	DTLS	TLS/SSL
Huella de codificación (Cantidad de recursos y memoria utilizados)	Pequeña	Grande	Grande
Eficiencia en el uso del ancho de banda	Formato de mensaje compacto, compresión de cabecera.	Formatos de mensaje más detallados o complejos.	Formatos de mensaje más detallados o complejos.
Integración e interoperabilidad	Diferentes plataformas, lenguajes de programación y sistemas operativos.	Opciones de integración multiplataforma limitadas o con mayor esfuerzo.	Opciones de integración multiplataforma limitadas o con mayor esfuerzo.
Calidad de Servicio	Sí, con tres niveles.	Sí, con dos niveles.	Sí, con tres niveles.

Velocidad de desarrollo e implementación	Rápido desarrollo e implementación.	Con su complejidad adicional, puede requerir más tiempo y esfuerzo de desarrollo.	Con su complejidad adicional, puede requerir más tiempo y esfuerzo de desarrollo.

Después de analizar los diferentes protocolos, se concluye que MQTT tiene características de ligereza y eficacia, por lo que lo hacen idóneo para entornos con recursos limitados, dispositivos de baja potencia y redes con ancho de banda limitado, además de su fácil implementación. También, se determinó que utilizando Raspberry Pi sobre una LAN el protocolo MQTT tiene un mejor tiempo de transferencia de mensajes entre editor y suscriptor que otros protocolos [35]. Por estas razones en el presente proyecto se utilizará MQTT como protocolo de aplicación. En la Figura 10 se muestra un ejemplo de MQTT adaptado al presente proyecto.

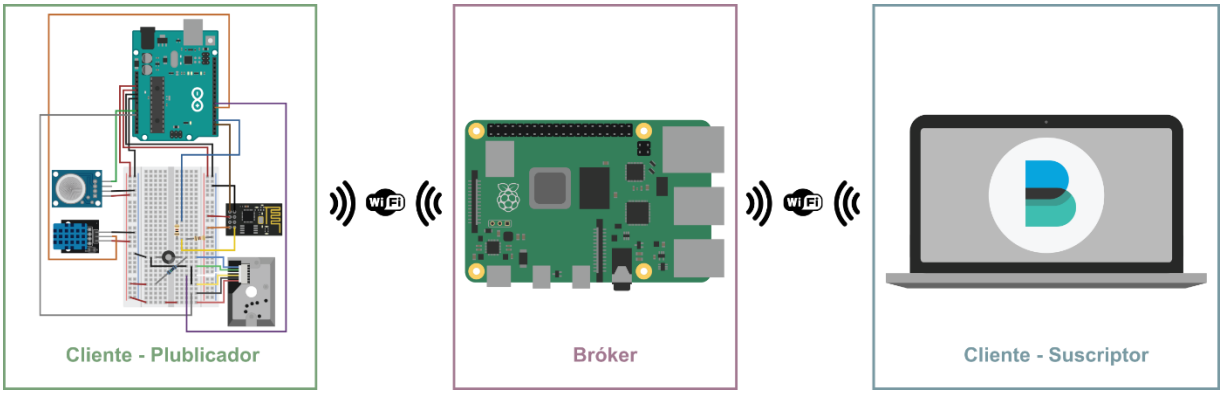


Figura 10. Diagrama de la capa de red

En los próximos apartados se explicará más a detalle los componentes de este protocolo, que serán adaptados junto a las capas de aplicación y percepción respectivamente. Cabe recalcar que los clientes serán divididos entre publicador y suscriptor debido a que tienen distintos propósitos dentro de la solución planteada; como tal serán solo etiquetas para identificarlos.

2.3.1.2.1.1. Bróker

El bróker es una entidad central de software en la arquitectura MQTT y juega un papel fundamental en el protocolo de publicación/suscripción. Su principal función es garantizar que los mensajes sean entregados a los suscriptores correspondientes. Dependiendo de la

implementación, un único bróker puede manejar simultáneamente a millones de clientes MQTT conectados.

La función principal del bróker es recibir todos los mensajes enviados a través del protocolo MQTT, filtrarlos y determinar a qué clientes suscritos deben ser entregados dichos mensajes. El bróker actúa como intermediario, asegurándose de que cada mensaje se envíe únicamente a los clientes que están suscritos a ellos. Adicionalmente, los brókers facilitan las transacciones entre clientes MQTT permitiendo que los dispositivos realicen una solicitud de conexión, autenticándolos, asegurando la conexión mediante el cifrado *Transport Layer Security* (TLS) solo como opción y, por último, el bróker también puede almacenar los mensajes en el servidor para que puedan ser reenviados en caso de pérdida de conexión no deseada, reconexión del cliente u otros eventos similares. Esta capacidad de almacenamiento asegura que los mensajes no se pierdan y puedan ser entregados correctamente a los clientes, incluso en situaciones de desconexión temporal [36] [37].

Al ser el bróker MQTT una entidad central, este se encarga de todo el trabajo pesado (procesamiento), los dispositivos cliente solo tienen que realizar un procesamiento mínimo con un ancho de banda mínimo. Por este motivo, MQTT funciona dentro de entornos que no puedan ser controlados todo el tiempo.

El bróker que se utiliza para el presente proyecto es Mosquitto, debido a que es un bróker que funciona bien en proyectos pequeños que no deben manejar múltiples clientes o un procesamiento tan amplio [38], esto no quiere decir que este bróker no puede manejar múltiples clientes. Asimismo, Mosquitto funciona bien con una placa Raspberry, ya que no necesita instalaciones adicionales, como en el caso de HiveMQ que requiere instalar recursos adicionales para su funcionamiento [39].

2.3.1.2.1.2. Publicador

Este cliente MQTT tendrá la función de enviar el IAQCI hacia el bróker al que se suscribirá el otro cliente (suscriptor). Las publicaciones se envían de los publicadores (nodo IoT) al intermediario de publicación/suscripción (bróker) un publicador no necesita saber quién utiliza la información (publicación) que proporciona [40].

Los clientes MQTT puede ser cualquier dispositivo que ejecute alguna librería MQTT, las librerías MQTT pueden ser escritas en una gran variedad de lenguajes de programación [36].

2.3.1.2.1.3. Suscriptor

Este cliente es el encargado de recibir la información que se almacena temporalmente en el bróker o intermediario. El suscriptor no necesita saber quién proporciona la información que recibe como resultado de una suscripción a un tema [40]. De forma similar al publicador, este cliente puede ser escrito en cualquier lenguaje de programación que soporte MQTT [36].

2.3.1.3. Capa de Aplicación

Esta capa es la encargada de almacenar y graficar los datos para su futuro análisis, además de proporcionar herramientas que permiten alertar al usuario cuando los datos no son los esperados. Con este objetivo se utilizará Elastic Stack que está conformado por Elasticsearch, que permitirá almacenar los datos; Kibana, que ayudará a crear las gráficas, así como las alertas y Beats, que ayudará a transportar los datos.

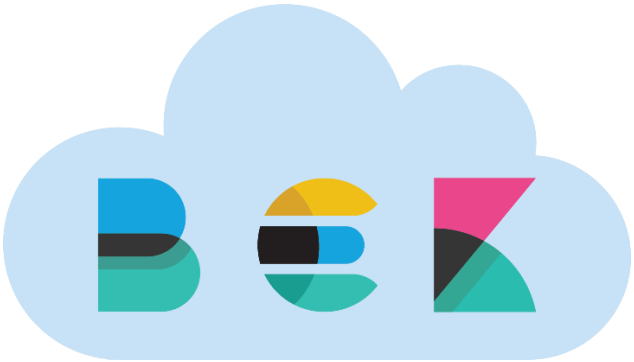


Figura 11. Diagrama de la capa de aplicación

El Elastic Stack, también conocido como ELK Stack, está compuesto por Elasticsearch, Kibana, Beats y Logstash. Esta suite de herramientas permite de forma confiable y segura tomar datos de diversas fuentes, en cualquier formato imaginable, permitiendo buscarlos, analizarlos y visualizarlos [41]. Elasticsearch es un motor de búsqueda y análisis de datos. Logstash, por otro lado, es un *pipeline* de procesamiento de datos en el lado del servidor que permite la ingestión de datos desde múltiples fuentes simultáneamente, los transforma y luego los envía a Elasticsearch. Por último, Kibana es una herramienta que permite a los usuarios visualizar los datos almacenados en Elasticsearch mediante la creación de cuadros y gráficos interactivos. Beats permite realizar el seguimiento a un solo archivo [41], por lo que se lo utilizará en el presente proyecto en vez de Logstash [42].

2.3.1.3.1. Servicio de almacenamiento y búsqueda

Elasticsearch es un motor de búsqueda y análisis distribuido basado en una arquitectura RESTful, lo que permite que sus sistemas se comuniquen de manera segura a través de internet [43]. Puede ser instalado dentro de un ordenador local, como ser ejecutado de inmediato dentro de un proveedor de nube como AWS, Google Cloud o Azure; esto permite almacenar y explorar datos para que ser estructurados, procesados y organizados según las necesidades del proyecto. Elasticsearch es el componente central del Elastic Stack, encargado de almacenar los datos de manera centralizada para permitir búsquedas rápidas y eficientes, relevante y con analíticas escalables; permite combinar búsquedas estructuradas y no estructuradas, además de que permite encontrar patrones fáciles de interpretar dentro de los datos almacenados [44].

Figura 12. Logotipo de Elasticsearch

Una de las ventajas que posee el Elastic Stack son los desencadenantes o sistema de alertas que integra. Dependiendo de los datos con los que se trabaje, Elastic permite enviar notificaciones gracias a la integración llamada Alerting. Esta integración incluso permite monitorear las alertas que se han enviado. Las alertas pueden estar vinculadas a acciones que utilizan integraciones incorporadas con herramientas como correo electrónico, webhooks, IBM Resilient, Jira, Microsoft Teams, PagerDuty, ServiceNow y Slack. Esto permite una respuesta rápida y eficiente a las alertas generadas por los datos monitoreados en el Elastic Stack [45]. Para el presente proyecto se optará por el envío de correos electrónicos.

2.3.1.3.2 Servicio de interfaz de usuario

Dentro del Elastic Stack, Kibana es la herramienta que proporciona una interfaz de usuario gratuita y de código abierto para visualizar los datos almacenados en Elasticsearch y explorar el entorno del Elastic Stack. Kibana ofrece una amplia gama de tipos de gráficos, como histogramas, gráficos de líneas, gráficos circulares, proyecciones solares, entre otros, para mostrar la información de manera visualmente atractiva. De igual forma se puede realizar

análisis avanzado de series temporales, entrenamiento y encuentro de anomalías con Machine Learning Asimismo, Kibana también brinda la capacidad de realizar análisis avanzados de series temporales, así como entrenamiento y detección de anomalías utilizando técnicas de aprendizaje automático (del inglés, *Machine Learning*) y exploración de grafos [46]. Esto permite aprovechar al máximo los datos almacenados en Elasticsearch y obtener información valiosa a partir de ellos. Dentro del proyecto, se utilizará un tablero o *dashboard* personalizado que muestre la información relevante relacionada al IAQCI.

Figura 13. Logotipo de Kibana

2.3.1.3.3. Servicio de recolección de datos

Para la parte de recolección de datos, se explicó en el apartado de la Capa de Red que se utilizaría Filebeat. Filebeat es un agente de la familia de Beats que permite trabajar con MQTT. Beats es una plataforma de código abierto y gratuita que se utiliza como agentes de datos con un objetivo específico. Estos agentes recopilan y envían datos de cientos o miles de máquinas y sistemas a Logstash o Elasticsearch. Beats permite recolectar datos que pueden ser almacenados en servidores, contenedores, o pueden ser desplegados como funciones para ser centralizados después en Elasticsearch [47].

Figura 14. Logotipo de Beats

2.3.2. Implementación del prototipo

El proceso de implementación del proyecto se dividió el en cuatro etapas que serán explicadas dentro del presente capítulo:

1. Creación de las historias de usuario con Scrum para la codificación necesaria dentro del prototipo.
2. La conexión física del nodo IoT.
3. Las configuraciones de red respectivas para la conectividad del prototipo con la nube.
4. La configuración de la aplicación en la nube.

2.3.2.1. *Scrum*

Dentro de este apartado se indican el equipo, los eventos y artefactos definidos para el proyecto, siguiendo lo dictado por la metodología *Scrum* y como se lo ha definido previamente. De igual forma se definirá Jira como software de gestión de proyectos que acopla *Scrum* como la metodología de trabajo para el presente proyecto. Adicionalmente, para la implementación de *Scrum*, se utilizó Jira.

Jira es una herramienta de software diseñada para la gestión de proyectos, tareas y errores, con un enfoque particular en el desarrollo de software. Permite la colaboración en equipo al integrar diferentes herramientas para la toma de decisiones, planificación y seguimiento del proyecto en tiempo real. Se seleccionó esta herramienta debido a su compatibilidad con *Scrum* y su opción gratuita para proyectos y equipos pequeños, a diferencia de Monday o YouTrack de Jetbrains que solo ofrecen un período de prueba [48]. Además, Jira se puede integrar con otras herramientas como GitHub o GitLab mediante *plugins*.

2.3.2.1.1. Equipo *Scrum*

Como primer paso para la implementación de la metodología, se definió los roles de los involucrados en el proyecto como se muestra en la Tabla 8.

Tabla 8. Equipo *Scrum*

Equipo *Scrum*	
Scrum Master	Alex Aguacondo
Product Owner	Diana Yacchirema
Desarrolladores	Doménica Jiménez, Alex Aguacondo

2.3.2.1.2. Épicas

Dentro de Jira, se crearon las épicas acordes al proyecto.

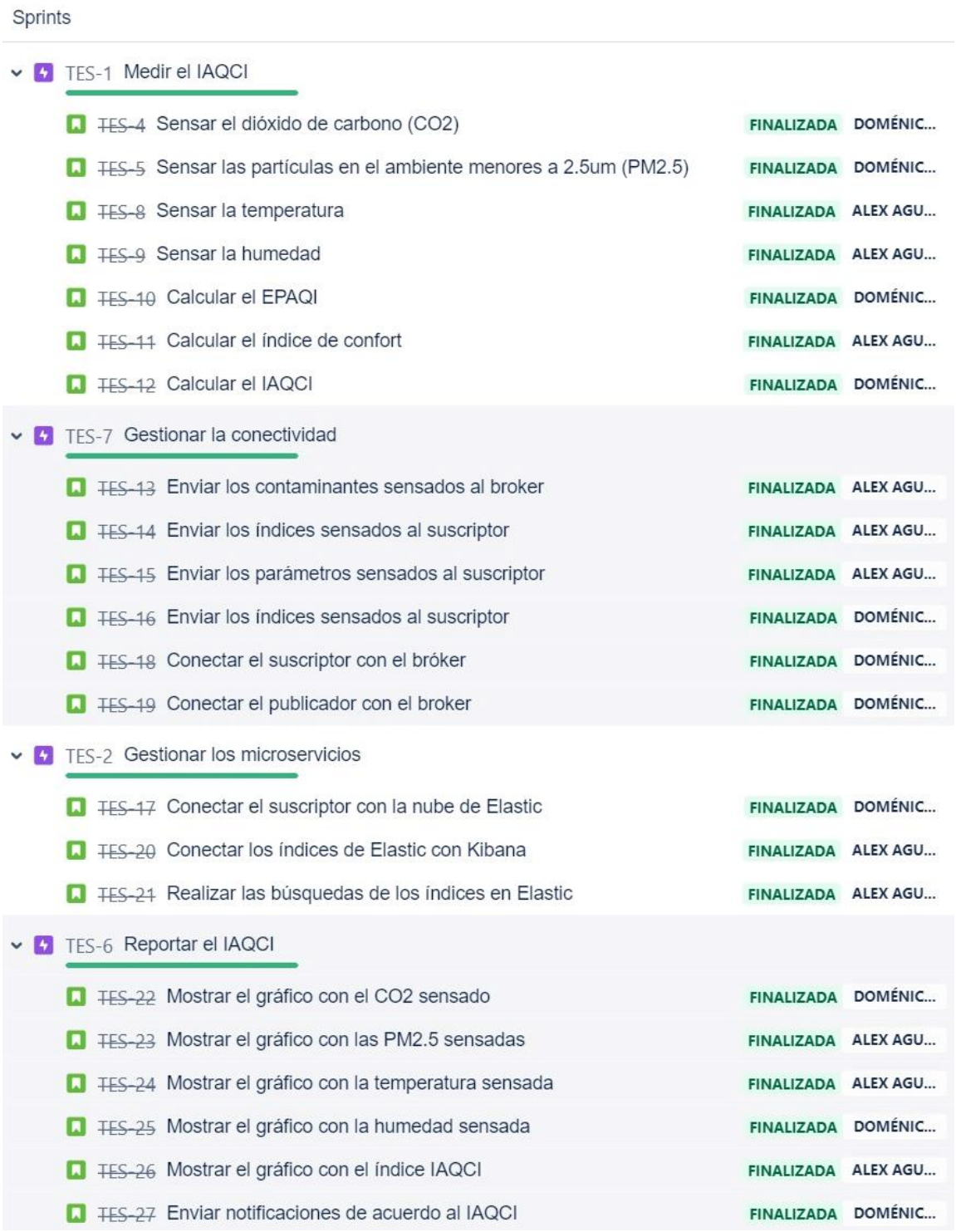


Figura 15. *Product backlog*

2.3.2.2. Configuración del nodo IoT

En esta sección se explicarán las configuraciones a nivel de hardware previamente realizadas para poder programar y utilizar correctamente los sensores y la placa del prototipo diagramado en la Figura 5. Adicionalmente, se explicará la codificación realizada.

2.3.2.2.1. Conexión del nodo IoT

Para que la placa Arduino UNO pueda conectarse a una red WiFi fue necesario integrar el módulo ESP-01 representado en la Figura 16. Para ello se conectó el módulo al Arduino utilizando las entradas de transmisión (Tx) y recepción (Rx) y se configuró la terminal en modo NL & CR (de las siglas en ingles *Newline* y *Carriage Return*) para poder interactuar con el módulo. Después, Se utilizaron los siguientes comandos que permiten configurar la placa: AT para verificar que se reconoce la placa. AT+UART_DEF para escoger los baudios con los que se debe trabajar (Para el presente proyecto se utilizó una velocidad de 9600 baudios). AT+CWJAP para conectarnos a la red mediante el nombre y la contraseña de la red. La Figura 17 muestra la conexión del nodo con la red.

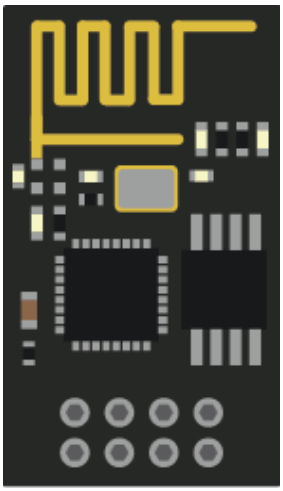


Figura 16. Modulo ESP-01

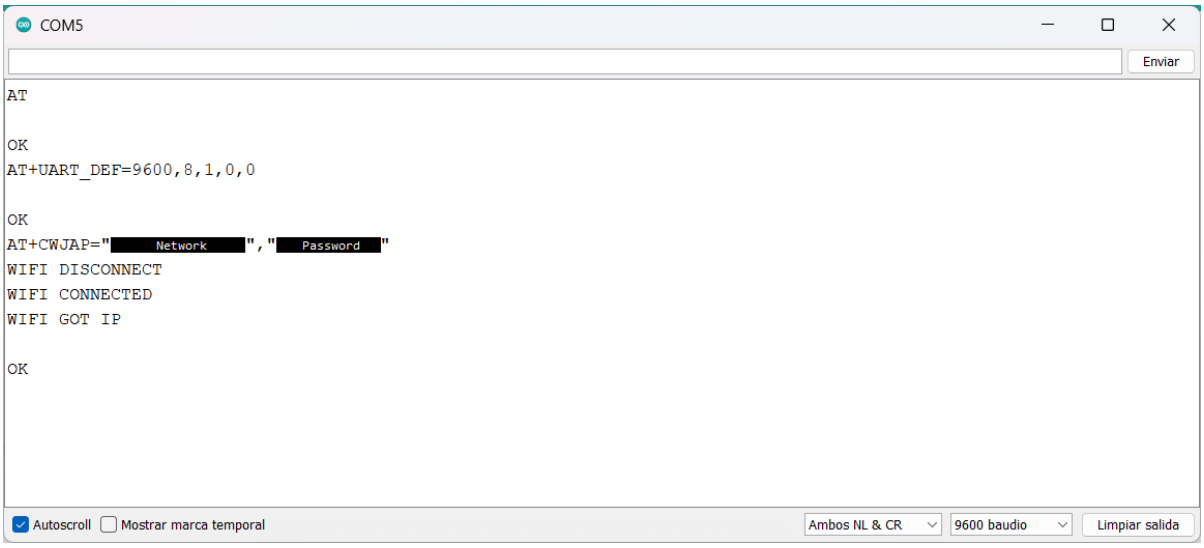
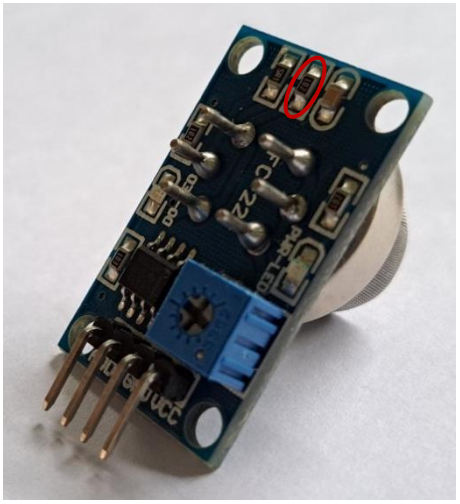
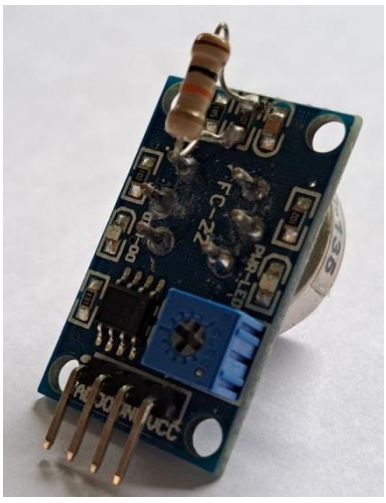


Figura 17. Conexión del nodo a la red mediante módulo ESP-01

Adicionalmente, a continuación, se detalla una configuración extra realizada para la obtención más precisa del CO2. El sensor MQ135 viene integrada con una resistencia de 1K como se visualiza en la Figura 18.a. Sin embargo, en las especificaciones técnicas de dicho sensor se recomienda utilizar una resistencia de 10K a 47K [29] con el fin de obtener medidas más precisas de los datos. Por esto, se adaptó el sensor con una resistencia de 10K para mejorar la toma de datos como se visualiza en la Figura 18.b.



a. Resistencia de 1K



b. Resistencia de 10K

Figura 18. Mejora del sensor MQ135

Adicionalmente el sensor MQ135 necesitaba ser calibrado antes de poder extraer la información necesaria, para ello fue necesario obtener el valor más bajo que pueda obtener el sensor para lo cual el sensor fue conectado a una placa Arduino, misma que fue desplegada en un ambiente cerrado con ventilación a fin de que recopile datos durante 24 horas y se obtenga el dato requerido. Para esto se configuró la placa Arduino con el código que se muestra en la Figura 19.

```
#define sensor A0
int gas, minval = 1000;

void setup() {
  pinMode(sensor, INPUT);
  Serial.begin(9600);
}

void loop() {
  gas = analogRead(sensor);
  if(gas <= minval){
    minval = gas;
  }
  Serial.print(gas);
  Serial.print(" ");
  Serial.print(minval);
  Serial.println();
  delay(200);
}
```

Figura 19. Código para obtener el menor valor del sensor MQ135

El menor dato que el sensor recopiló después de 24 horas es 115 ppm como muestra la Figura 20.

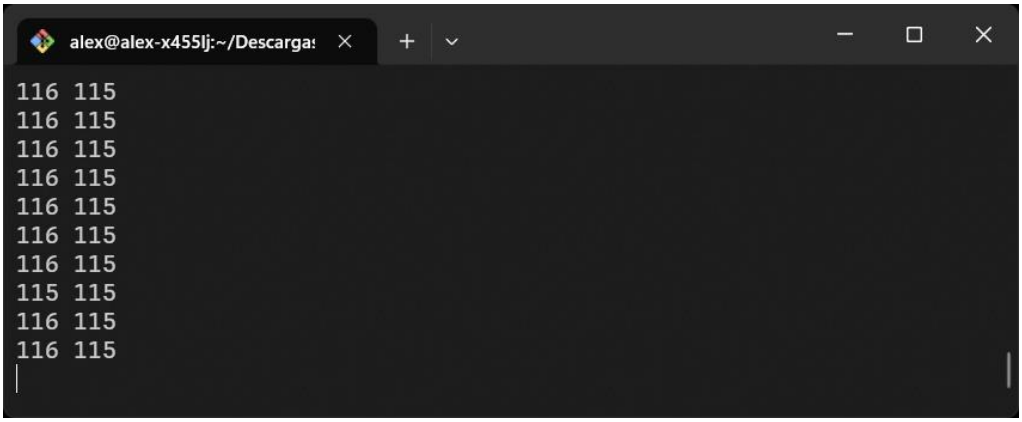


Figura 20. Últimos datos recopilados para obtener el menor valor del sensor MQ135

Para asegurar que el valor analógico que se recibe del sensor está en el rango de 0 a 1024 fue necesario restar al valor monitorizado por el sensor el valor menor que se obtuvo. Finalmente, para normalizar los datos fue necesario mapear esta información con los valores máximos y mínimos establecidos en los estándares de la concentración de CO2 en partes por millón. El código para realizar este proceso se muestra a continuación en la Figura 21:

```
//MQ135
float co2Input = analogRead(PIN_MQ135);
co2lvl = co2Input - 115;
co2lvl = map(co2lvl, 0, 1024, 400, 5000);
Serial.print(" co2lvl: ");
Serial.print(co2lvl);
Serial.println(" ppm ");
```

Figura 21. Código para normalizar el valor obtenido por el sensor MQ135

2.3.2.2.3. Programación del Nodo IoT

La Figura 22 muestra el nodo final que se utilizará para monitorizar los datos del entorno. En esta sección se explicarán los bloques de código más importantes y las configuraciones realizadas a nivel de red para el envío de datos desde el nodo. El código completo del nodo se encuentra en el ANEXO II.

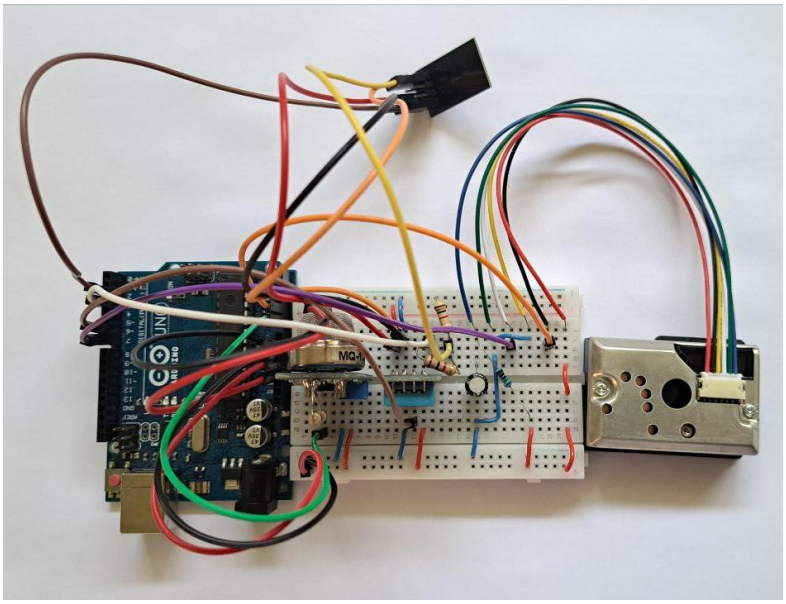


Figura 22. Conexiones finales del nodo

La Tabla 9 muestra las librerías utilizadas para la programación del nodo IoT incluyendo una breve explicación de su funcionalidad

Tabla 9. Librerías utilizadas en el nodo IoT

	Librerías	Versión	Descripción
Arduino	`#include < WiFiEsp.h>`	1.8.0	Esta librería de Arduino permite conectar a la placa al internet [49].
MQ135	`#include "MQ135.h"`	1.1.0	Esta biblioteca permite a cualquier placa compatible con Arduino interactuar fácilmente con el sensor de calidad del aire MQ135 y calcular concentraciones de CO2 corregidas por temperatura y humedad [50]. Esta librería es opcional, suele estar instalada por defecto en la IDE de Arduino.
DHT11	`#include "DHT.h"`	1.4.4	Librería Arduino para sensores de temperatura y humedad DHT11, DHT22 [51].
GP2Y1010AU0F	`N/A`	N/A	Para utilizar el sensor de polvo no se necesita ninguna librería adicional.
MQTT	`#include <PubSubClient.h>`	2.8.0	Librería que proporciona un cliente que simplifica el publicar/suscribir mensajería con un servidor MQTT [52] [53].

Lo primero que se realizó fue conectar el nodo IoT a la red WiFi, para esto se utilizó la librería `WiFiEsp.h` que se mencionó en la Tabla 9 y el código presentado en la Figura 23 donde primero se verificará el estado de la conexión y si aún no se encuentra activa se inicializará la conexión

para lo que es necesario proporcionar la información de la red a la que queremos conectarnos y su contraseña, en este caso esa información está guardada en constantes.

```
Serial.println("Iniciar conección a la red WIFI");
while(status != WL_CONNECTED) {
    Serial.print("Intentando conectarse a WPA SSID: ");
    Serial.println(WIFI_AP);
    //Conectar a red WPA/WPA2
    status = WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    delay(500);
}
Serial.println("Conectado a la red WIFI");
```

Figura 23. Código para conexión del nodo IoT a la red WiFi

Para enviar los datos a la capa de red fue necesario configurar el nodo IoT para que funcione como cliente MQTT y con la ayuda del protocolo MQTT realizar una conexión hacia la Raspberry que funciona como bróker. Para esto utilizamos el código mostrado en la Figura 24 que mediante la IP del bróker permite entablar la conexión y enviar los datos.

```
//Creamos un loop en donde intentamos hacer la conexión
while(!client.connected()) {
    Serial.print("Conectando a: ");
    Serial.println(server);
    //Creamos una nueva cadena de conexión para el servidor
    //e intentamos realizar la conexión nueva
    //si requiere usuario y contraseña la enviamos connect(clientId, username, password)
    String clientId = "ESP8266Client-" + String(random(0xffff), HEX);
    if(client.connect(clientId.c_str())) {
        Serial.println("[DONE]");
    } else {
        Serial.print( "[FAILED] [ rc = " );
        Serial.print( client.state() );
        Serial.println( " : retrying in 5 seconds]" );
        delay( 5000 );
    }
}
```

Figura 24. Código para conexión del nodo IoT a la Raspberry

Para obtener los datos de las partículas de polvo del sensor GP2Y1010AU0F se utilizó el código presentado en la Figura 25. Primero, se necesita que la luz led del sensor se encuentre parpadeando para captar el polvo, para ello se utilizó la función *digitalWrite* que permite encender y apagar la luz. Posteriormente fue necesario normalizar los datos en el rango de 0

a 1024 de acuerdo con las especificaciones del sensor, con este propósito utilizamos el valor del voltaje y el valor analógico recibido en las fórmulas presentadas en la Figura 25.

```
// GP2Y
digitalWrite(LED,LOW); // power on the LED
delayMicroseconds(samplingTime);
float gp2YInput = analogRead(GP2Y); // read the dust value
delayMicroseconds(deltaTime);
digitalWrite(LED,HIGH); // turn the LED off
delayMicroseconds(sleepTime);
// 0 - 5V mapped to 0 - 1023 integer values
calcVoltage = gp2YInput * ( 5.0 / 1024.0 );
dustDensity = 17 * calcVoltage - 0.1;
Serial.print("Dust Sensot ");
Serial.print(dustDensity); // unit: ug/m3
Serial.print(F(" ug/m3 "));
delay(1000);
```

Figura 25. Código para normalizar el valor obtenido por el sensor GP2Y1010AU0F

El siguiente sensor que se programó es el DHT11, que permite extraer los datos de temperatura y humedad. Para esto se utilizaron las funciones *readHumidity* y *readTemperature* de la librería `DHT.h` detallada en la Tabla 8 que facilita recopilar los datos de dicho sensor como muestra la Figura 26.

```
//DHT11
humidity = dht.readHumidity();
temperature = dht.readTemperature();
Serial.print("Humedad: ");
Serial.print(humidity);
Serial.print(" % - ");
Serial.print("Temperatura: ");
Serial.print(temperature);
Serial.print(" C ");
delay(1000);
```

Figura 26. Código para obtener los valores obtenido por el sensor DHT11

Finalmente se almacenaron los datos captados por los sensores y se enviaron a la Raspberry mediante el protocolo MQTT, esto se realizó cada 5 segundos. Para esto se creó una variable de tipo cadena de caracteres similar a un JSON en la que se almacenaron los datos y se

publicaron por el cliente MQTT al *topic* escogido, en este caso fue “casa/piso-1/sala/nodo-iot” como muestra la Figura 27.

```
// Prepare a JSON payload string
getData();
String json = buildJson();
char jsonStr[200];
json.toCharArray (jsonStr, 200);
Serial.print(jsonStr);
delay(5000);

client.publish( "casa/piso-1/sala/nodo-iot", jsonStr );
```

Figura 27. Código para publicar datos a la Raspberry

La Figura 28 muestra de manera gráfica el envío de datos desde el nodo IoT hacia el bróker por medio de una conexión inalámbrica WiFi.

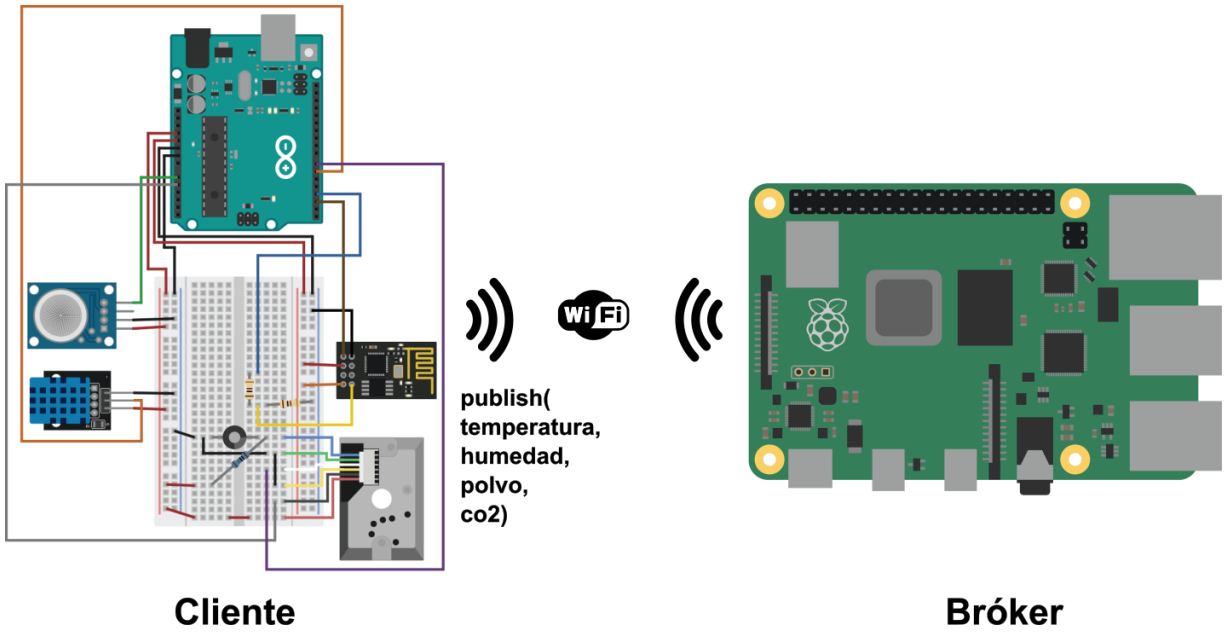


Figura 28. Envío de datos desde el Cliente hacia el Bróker

2.3.2.3. Configuración de red

El dispositivo que permite la interconexión de todos los componentes del prototipo es la Raspberry PI, permitiendo obtener los datos enviados del nodo Iot utilizando el prototipo MQTT y enviando estos datos a la instancia en la nube de Elastic utilizando una comunicación REST En esta sección se detallaran las configuraciones realizadas a nivel de red.

2.3.2.3.1. Configuración del cliente MQTT (publicador)

El cliente MQTT que funciona como publicador, es el nodo IoT. Dentro de esta sección no se ahondará más que cómo se definió el tema que se implementa en su sección de código mencionado en el apartado anterior.

Para este proyecto se utilizó el filtrado por tema de MQTT, ya se conoce que el tema a publicar es la información sensada por el nodo IoT. Y, para esto es necesario publicarlo dentro de un árbol jerárquico que es construido a partir de la implantación de la localidad donde se realizará la captación de datos que se muestra en la Figura 29. Dentro de la localidad, el nodo IoT se ubicará en la sala del primer piso representado en la Figura 29.a, debido a que se encuentra cerca del punto de acceso a internet inalámbrico.

a. Piso 1 **b.** Piso 2 **c.** Piso 3

Figura 29. Implantación de la localidad dende se captarán los datos

El árbol jerárquico se muestra en la Figura 30 y permite ubicar al nodo IoT dentro de la localidad de prueba, así como cada piso y sus respectivas habitaciones en las que se puede agregar diferentes dispositivos IoT ya que cada dispositivo actuaría como un tema diferente y no afectaría al funcionamiento de los demás.

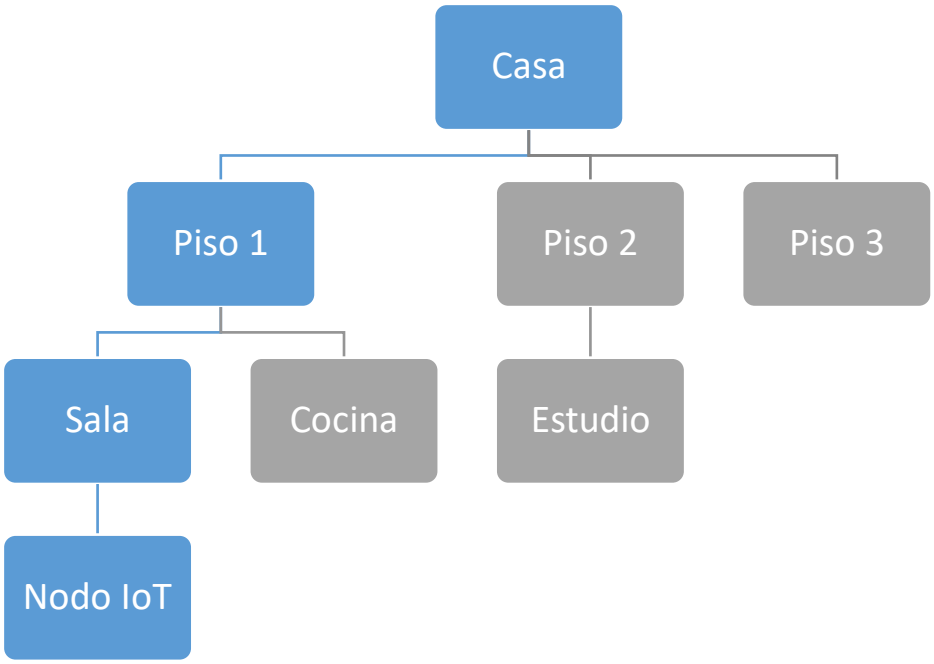


Figura 30. Árbol jerárquico de temas MQTT

Se toma en cuenta la ubicación del nodo IoT dentro de la Figura 30 y se obtiene el tema para enviar los datos mediante un publicador que es casa/piso-1/sala/nodo-. Finalmente, el envío de datos en la capa de red mediante MQTT para este proyecto se muestrea en la Figura 31.

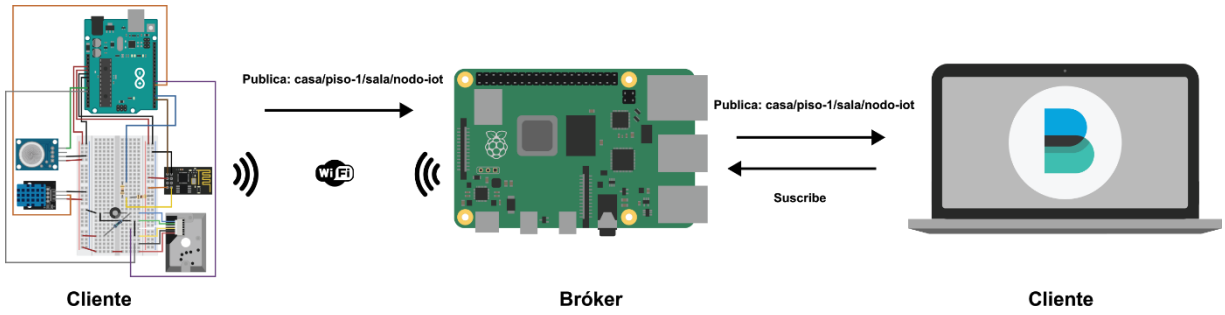


Figura 31. Arquitectura MQTT

2.3.2.3.2. Configuración del bróker MQTT

Para configurar la Raspberry como publicador MQTT se utilizó el bróker Mosquitto en su versión 3.1. Este bróker es ligero y puede utilizarse en todos los dispositivos, desde ordenadores monoplaca de bajo consumo hasta servidores completos [54].

Figura 32. Logotipo de Mosquitto

Las funciones que se utilizaron dentro del proyecto son las explicadas en la Tabla 10, son las encargadas de publicar mensajes o suscribirse en algún tema.

Tabla 10. Comandos de Mosquitto

Comando	Opciones	Descripción
`mosquitto_pub`	• `-h / --host:` Especifica el host al que conectarse. Por defecto es localhost [55]. • `-m / --message:` Envía un único mensaje desde la línea de comandos [55]. • `-t / --topic:` El tema MQTT en el que publicar el mensaje.	Utilidad de línea de comandos para publicar mensajes en un bróker [56].
`mosquitto_sub`	• `-h / --host:` Especifica el host al que conectarse. Por defecto es localhost [57]. • `-t / --topic:` El tema MQTT en el que publicar el mensaje [57].	Utilidad de línea de comandos para suscribirse a temas en un bróker [56].

Para la parte del servidor se utilizó una Raspberry Pi. Raspberry, es una computadora de placa única y bajo coste. Se puede emplear para varios proyectos y con varios objetivos, para este proyecto se utilizó a modo de Gateway para captar los datos y procesarlos. La versión que se manejó es Raspberry Pi 4 [58].

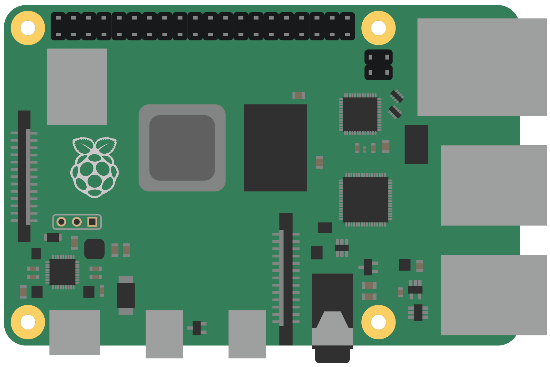


Figura 33. Placa Raspberry

Para utilizar Raspberry Pi, es necesario contar con una tarjeta de memoria y SO (Sistema Operativo). El sistema operativo predeterminado para este dispositivo es Raspberry Pi OS, que es una variante de Unix basada en la distribución Debian Linux y está diseñado específicamente para los ordenadores de placa única de la familia Raspberry Pi [59].

Dado que Raspberry Pi OS se basa en una distribución Debian Linux, podemos utilizar los comandos que se enumeran en la Tabla 11 para establecer el bróker de Mosquitto como un servicio dentro de la Raspberry Pi. Esto quiere decir que el bróker de Mosquitto iniciará automáticamente cuando la Raspberry Pi se encienda.

Tabla 11. Comandos de Raspberry Pi para iniciar el bróker de Mosquitto

Comando	Descripción
`systemctl enable mosquitto.service`	Utilidad de línea de comandos para iniciar el servicio de Mosquitto cuando se reinicie el sistema operativo [60].
`systemctl start mosquitto.service`	Utilidad de línea de comandos para iniciar el servicio de Mosquitto con el sistema operativo encendido [60].

Primero, se realizó la conexión inalámbrica al punto de red disponible, posteriormente se estableció una dirección IP estática a la cual apuntará el nodo cuando quiera publicar información. Para esto se debe editar el archivo **`/etc/dhcpcd.conf`** con las siguientes configuraciones:

1. Con el archivo, se establece una interfaz de red en el campo **`interface`**, **`wlan0`** para la conexión inalámbrica, **`eth0`** para una conexión mediante cable de red; la dirección IP del rúter al que se conectará la Raspberry especificándolo en el campo **`static routers`**; la dirección IP estática que se establecerá para la Raspberry se especifica en el campo **`static ip_address`**; y, por último, se establece una dirección IP que le permite comunicarse hacia el exterior estableciendo este valor en el campo **`domain_name_servers`**.

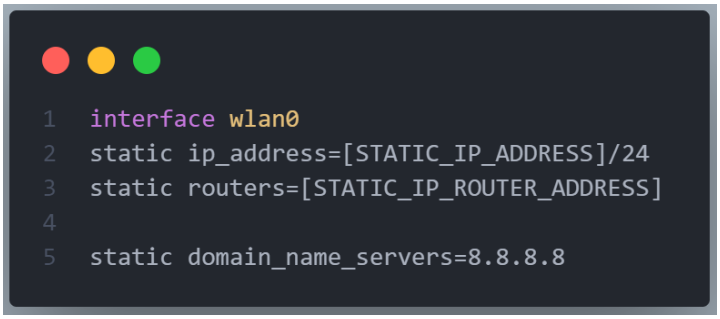


Figura 34. Configuraciones estáticas para la Raspberry

A continuación, se instaló el bróker de Mosquitto dentro de la Raspberry. Posterior a esto, se comprobó la versión que se instaló con el comando **`mosquitto -v`** como se ve en la Figura 35.

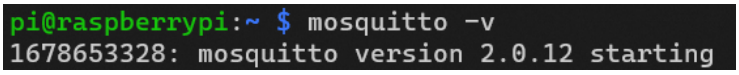


Figura 35. Versión de Mosquitto en la Raspberry Pi 4

Adicionalmente, se detallan los pasos para que el bróker funcione como un servicio dentro del sistema operativo y al momento de encender la Raspberry también se inicie este servicio. Esto se logró ejecutando el comando **`sudo systemctl enable mosquitto.service`**.

Todos los pasos detallados anteriormente permiten la conexión con el bróker de Mosquitto de forma local, es decir, se puede utilizar el bróker solamente desde la Raspberry. Para que el nodo IoT pueda utilizarlo, se tiene también que modificar el archivo **`/etc/mosquitto/mosquitto.conf`**. Dentro del archivo se modificaron los campos

`listener`, para configurar el puerto y las direcciones IP que van a ser aceptadas por el bróker; y, el campo **`allow_anonymous`** para no solicitar al cliente MQTT credenciales de acceso.

```
1  listener 1883 0.0.0.0
2  allow_anonymous true
```

Figura 36. Configuraciones adicionales para la Raspberry Pi 4

2.3.2.3.3. Configuración del cliente MQTT (suscriptor)

Filebeat es una herramienta liviana diseñada para transportar y consolidar datos de registros o logs. Por lo general, se instala como un agente en servidores, y su función principal es supervisar los registros o ubicaciones específicas. Recopila eventos de registro y los envía posteriormente a Elasticsearch para su posterior indexación [61].

En la Figura 37 se muestra el funcionamiento de Filebeat. Los recolectores de datos o *harvester*, apuntan a las ubicaciones de los datos de registros, estos se encargan de leer nuevo contenido, para luego enviarlo a libbeat (libbeat es una librería escrita en Go que contiene los paquetes comunes para todos los Beats, como Filebeat [62]), que luego son enviados a una salida como en este caso es Elasticsearch.

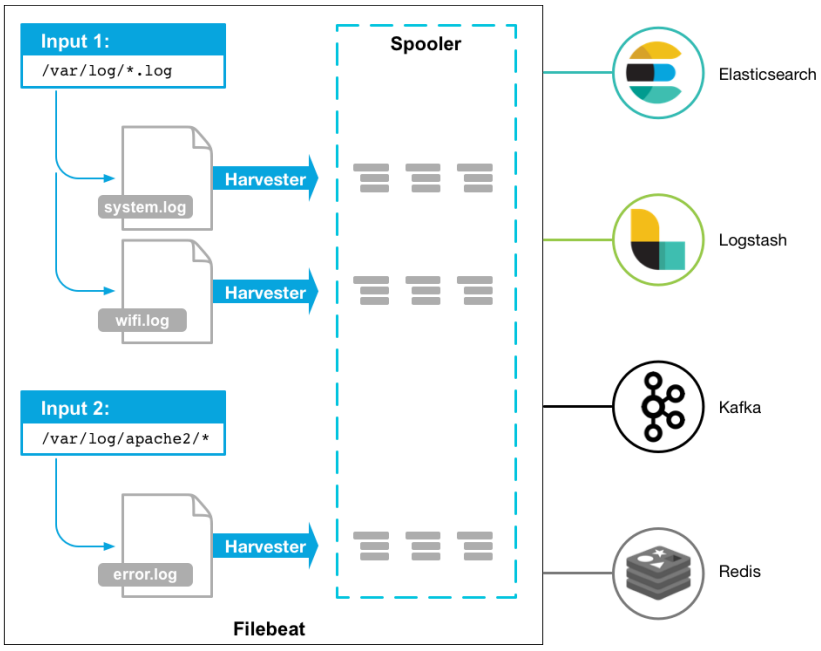


Figura 37. Representación del funcionamiento de Filebeat

Se utilizó un ordenador donde se instaló el servicio de Filebeat, que es el encargado de gestionar la comunicación con la aplicación de Elastic, además de suscribirse al tema con los datos del IAQCI. La instalación y ejecución de este servicio, fue realizada siguiendo las indicaciones de la guía oficial de Elastic. Por lo que dentro de este apartado se abordará principalmente la configuración del archivo **`filebeat.yml`**.

Las principales modificaciones que se realizaron dentro de **`filebeat.yml`** fueron añadirle configuraciones de *inputs*, *cloud* y *processors*. Los *processors* utilizados fueron *dissect*, *convert* y *script*.

Dentro del presente proyecto, el *input* que se utilizó es el propio *input* para MQTT de Filebeat. Se conecta al bróker MQTT, se suscribe a los temas seleccionados y analiza los datos en líneas de mensajes comunes [63]. En la Figura 38 se muestra el ejemplo de configuración del input utilizado:

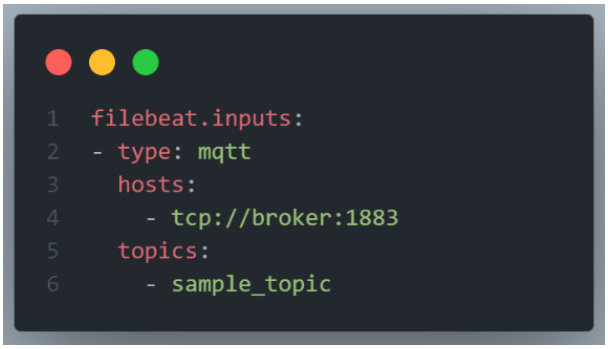


Figura 38. Configuración de *inputs* en el archivo filebeat.yml

Y como salida se configuró la instancia de Elasticsearch, para esto se utilizaron las variables *cloud.id* y *cloud.auth* cuyos valores pertenecen a la instancia de Elasticsearch a la que serán enviados los datos [64]. Esta configuración se muestra en la Figura 39.

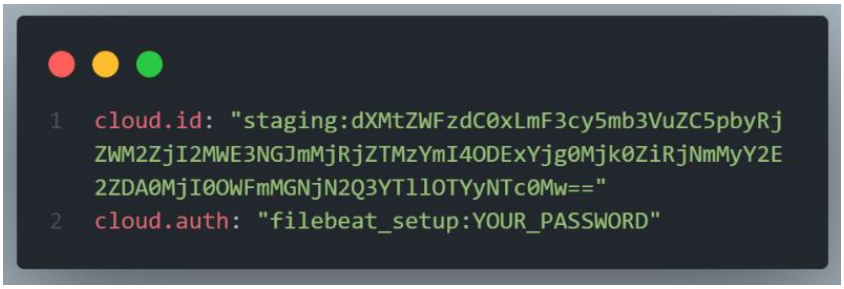


Figura 39. Configuración de *cloud* en el archivo filebeat.yml

El procesador *dissect* es utilizado para dividir el mensaje que llega del nodo en campos que son considerados texto y se muestra en la Figura 40.

```
- dissect:
    tokenizer: "%{temperatura},%{humedad},%{co2},%{polvo}"
    field: "message"
    target_prefix: ""
```

Figura 40. Configuración de dissect en el archivo filebeat.yml

Para transformar los campos de texto en valores numéricos, se utiliza el procesador *convert* como se muestra en la Figura 41.

```
- convert:
    fields:
      - {from: "temperatura", to: "temperatura_1", type: "float"}
      - {from: "humedad", to: "humedad_1", type: "float"}
      - {from: "co2", to: "co2_1", type: "float"}
      - {from: "polvo", to: "polvo_1", type: "float"}
    ignore_missing: true
    fail_on_error: false
```

Figura 41. Configuración de covert en el archivo filebeat.yml

Una vez convertidos los campos de texto a valores numéricos se puede realizar cálculos con ellos utilizando el modificador *script*. Este modificador permite integrar código JavaScript al servicio, se utilizó el archivo JavaScript adjunto en el Anexo IV para el cálculo del IAQCI mediante la fórmula (5). Adicionalmente, el archivo de configuración completo de Filebeat se encuentra dentro del Anexo III.

2.3.2.4 Configuración de aplicación web

Para mostrar de manera gráfica los datos monitorizados por los sensores a los usuarios finales fue necesario crear una aplicación web. Se creo la aplicación web utilizando un *dashboard* de Kibana dentro de una instancia de Elasticsearch. La instancia utilizada para este proyecto se puede visualizar en la Figura 42.

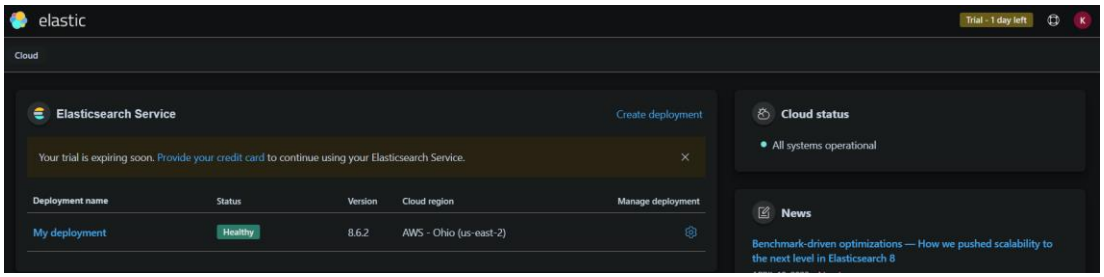


Figura 42. Venta de instancias de Elasticsearch

Para conectar esta instancia con Filebeat, solo se necesita añadir las credenciales que fueron generadas durante el proceso de creación. Además, también se requiere el Cloud ID, que se obtiene al ver las propiedades de la instancia de Elastic creada. Una vez obtenida esta información, se lo adjunta al archivo de configuración de Filebeat, **`filebeat.yml`**. Para este punto solo resta cargar los índices con el comando **`sudo ./filebeat -e`** y reiniciar el servicio de Filebeat para el envío normal de datos.

Para generar las visualizaciones de los datos se utilizará Kibana y se tendrán 3 maneras de visualizar los datos: Lineal, Último valor y Medidor. Se debe crear un *Dashboard* para poder generar varias visualizaciones de los diferentes datos de interés, esta opción se encuentra en el panel de opciones de Elastic, en la sección *Analytics* como muestra la Figura 43.

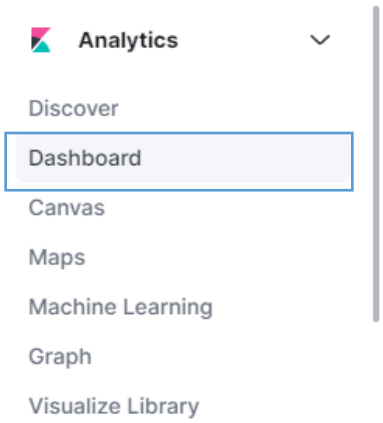


Figura 43. Opción *Dashboard* en Elastic

Para crear la visualización Lineal, dentro del *Dashboard* se debe seleccionar la opción *Create visualization* que permitirá acceder a una nueva ventana con las opciones de visualización. En este nuevo panel se debe escoger la instancia de Filebeat que contiene todos los datos, el tipo de gráfica y los campos a mostrar, en la Figura 44 se puede observar la configuración de la visualización del campo IAQCI y en la Figura 45 se muestra el resultado de la gráfica final del campo IAQCI.

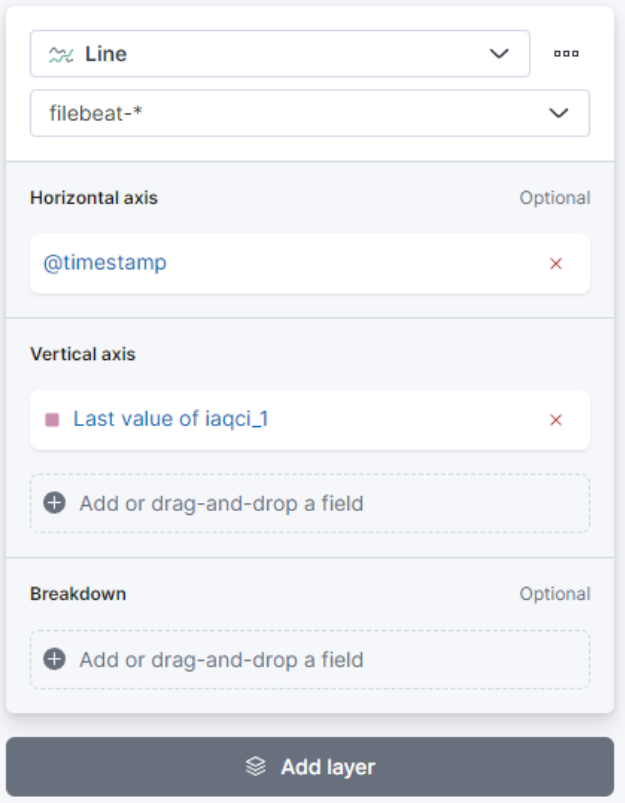


Figura 44. Configuración de la visualización para el histórico del campo IAQCI

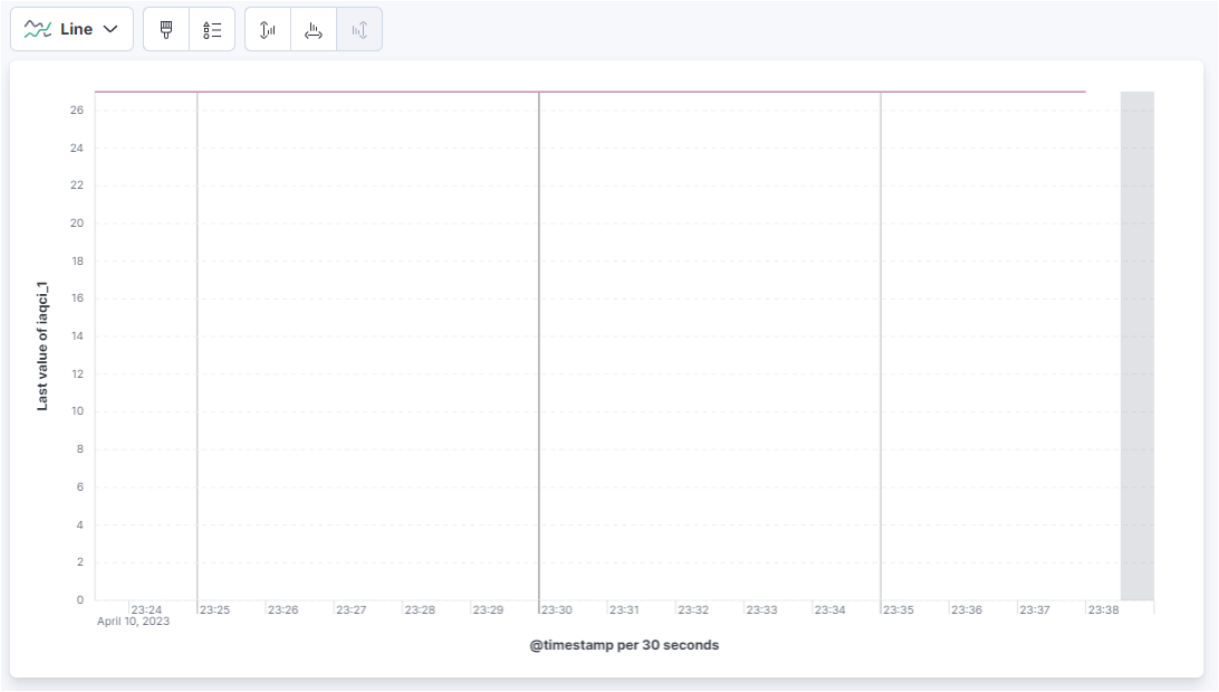


Figura 45. Visualización del histórico del campo IAQCI

Para la visualización del Último valor, en la aplicación existe una opción llamada Last value (último valor) que permite obtener los datos en tiempo real, se debe escoger esta opción como muestra la Figura 46. Adicionalmente, esta visualización se ha configurado con una gama de colores basada en los umbrales de color presentados en el Capítulo 1 como muestra en la Figura 47 lo que la hace más intuitiva para el usuario. Finalmente, un ejemplo de esta visualización se puede observar en la Figura 48.

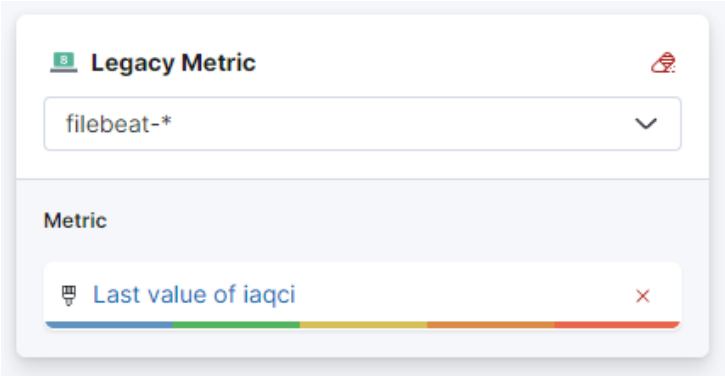


Figura 46. Configuración de la visualización para el valor actual del campo IAQCI

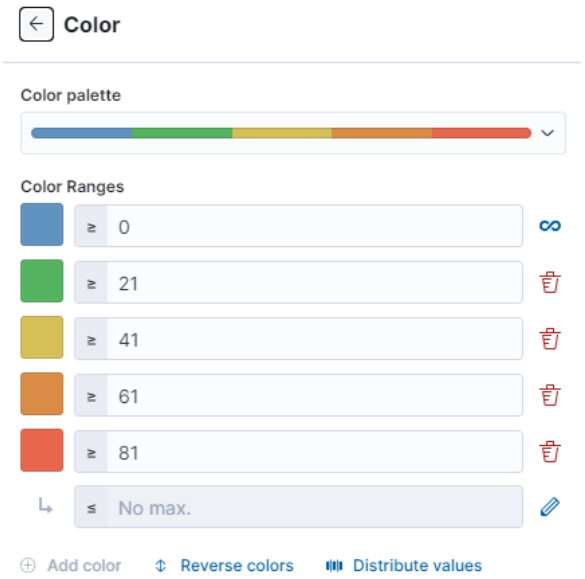


Figura 47. Configuración de la paleta de colores para el valor actual del campo IAQCI

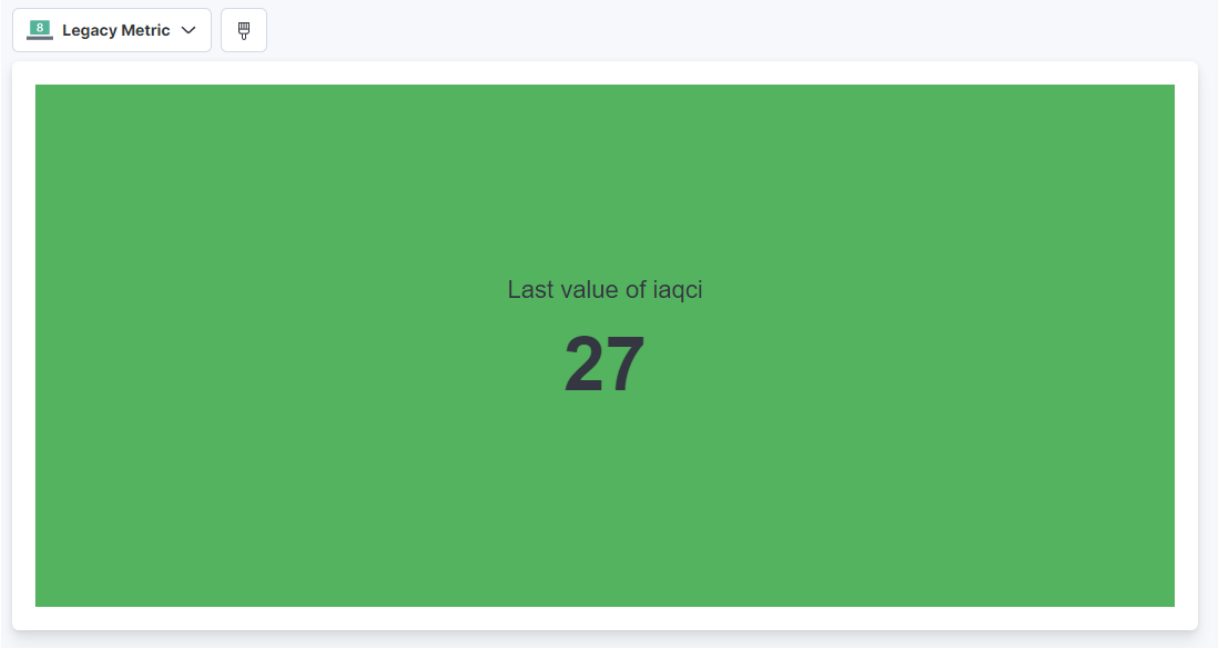


Figura 48. Visualización del valor actual del campo IAQCI

Para la visualización Medidor, se debe acceder a la opción *Visualize Library* en la sección *Analytics* dentro del panel de opciones de Elastic como se muestra en la Figura 49.

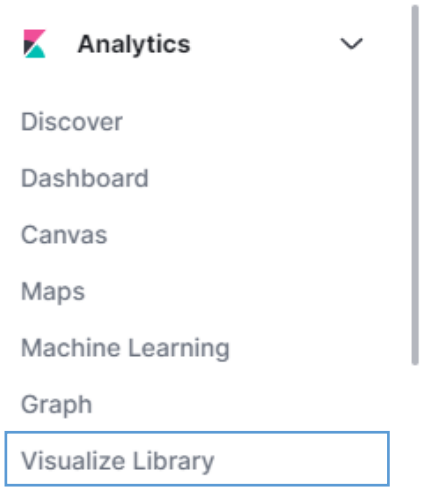


Figura 49. Opción *Visualize Library* en Elastic

Se debe crear una nueva visualización con la opción *Create visualization*, Figura 50. Dentro de este nuevo panel se debe escoger la opción *Explore options* que se encuentra dentro de la sección *Aggregation based* como se ve en la Figura 51. Posteriormente se debe escoger la opción Gauge, Figura 52. Y finalmente se debe escoger el origen de los datos para esta visualización que debe ser Filebeat como muestra la Figura 53.

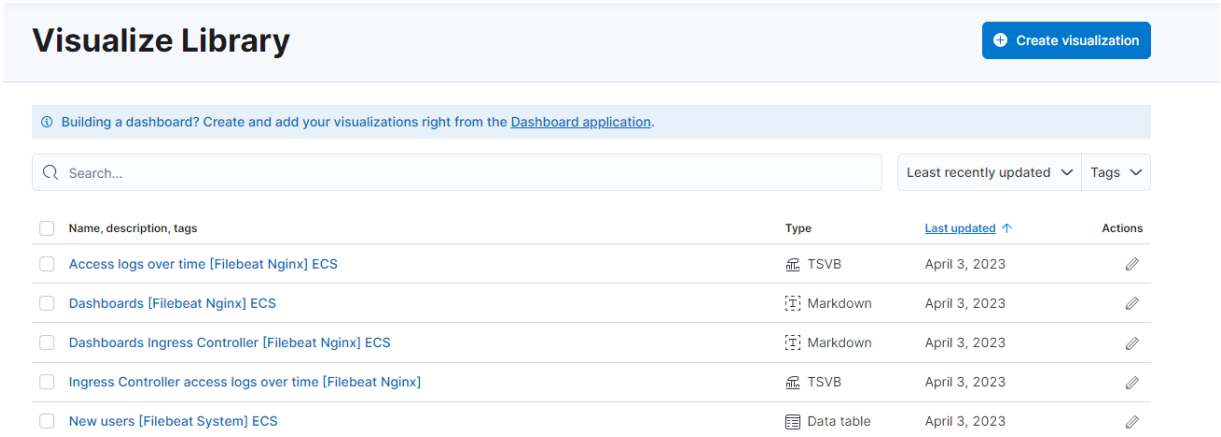


Figura 50. Ventana *Visualize Library* en Elastic

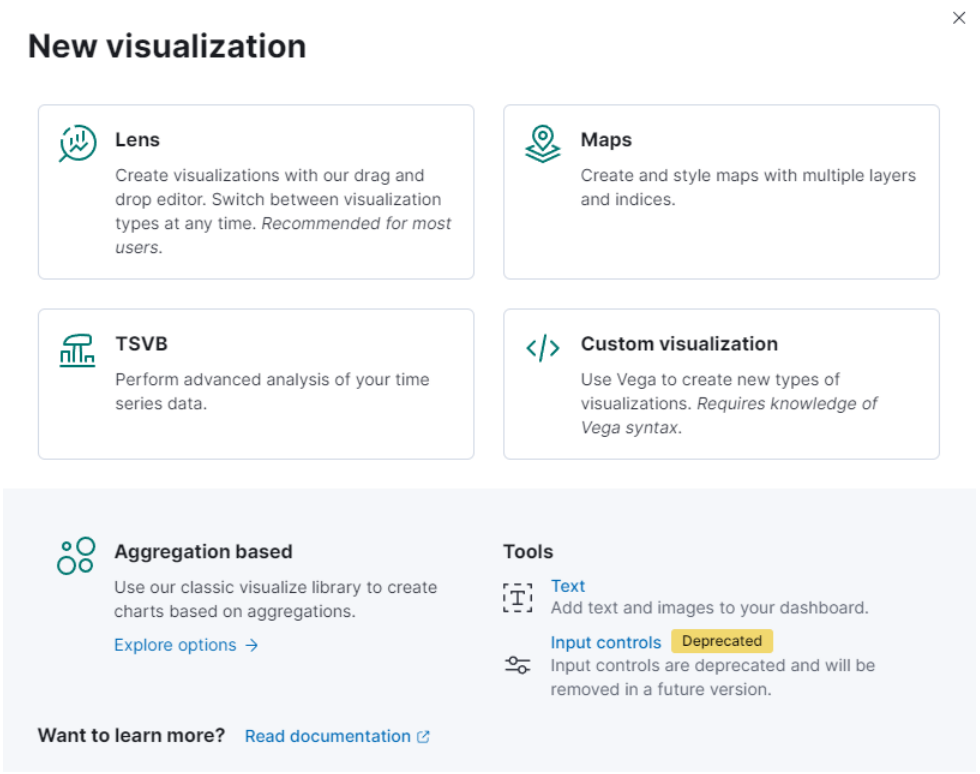


Figura 51. Ventana *New visualization* en Elastic

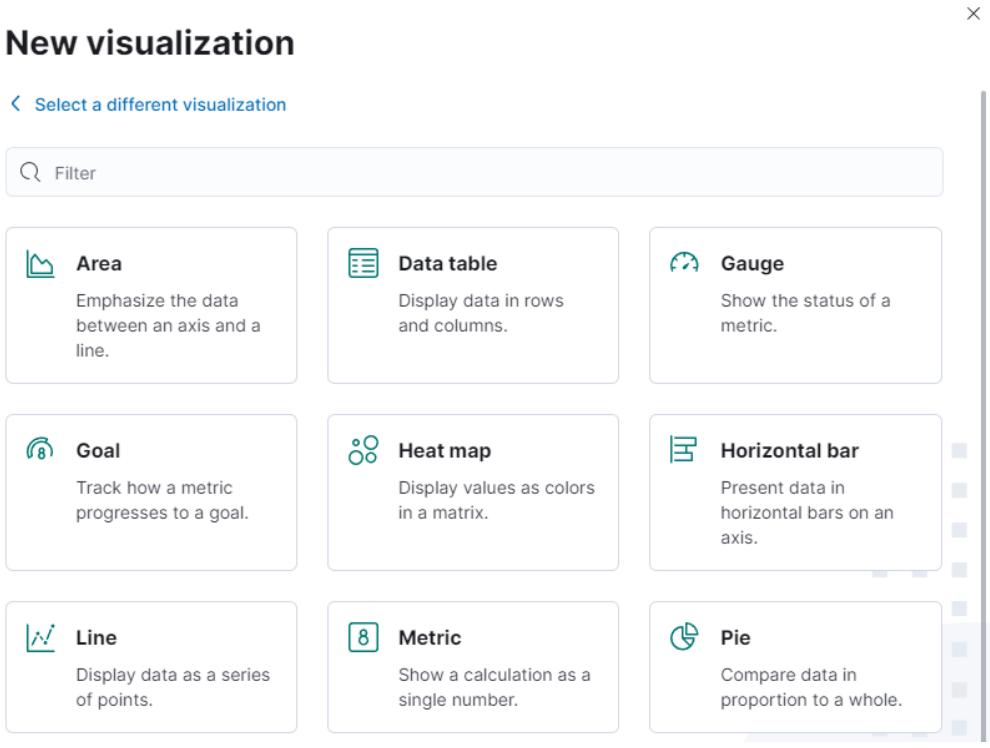


Figura 52. Ventana para seleccionar una visualización diferente en Elastic

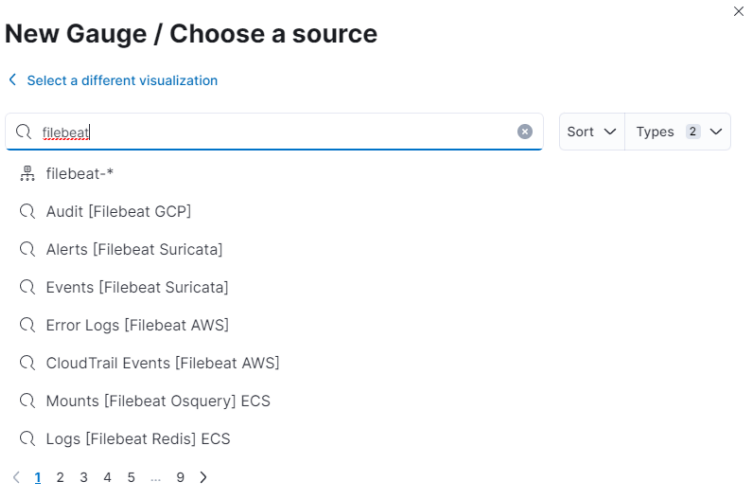


Figura 53. Ventana para seleccionar el origen de los datos para la visualización

Las configuraciones mostradas en la Figura 54 permiten crear un medidor del último valor obtenido de IAQCI en Elastic. En la Figura 55 se muestra la configuración de los rangos, los valores referenciales son los presentados en la Tabla 5. La gráfica final de estas configuraciones se puede observar en la Figura 56.

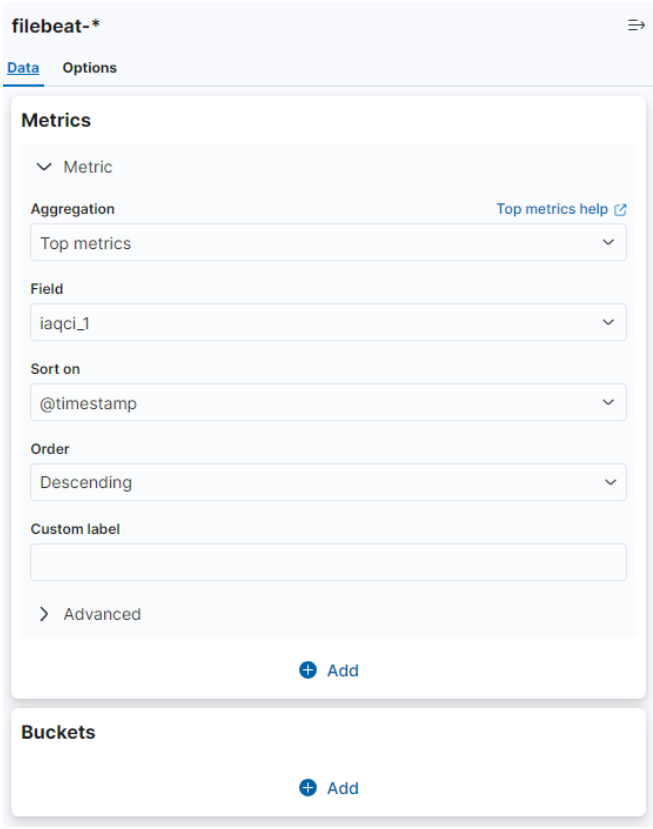


Figura 54. Configuración de la visualización de medidor del campo IAQCI

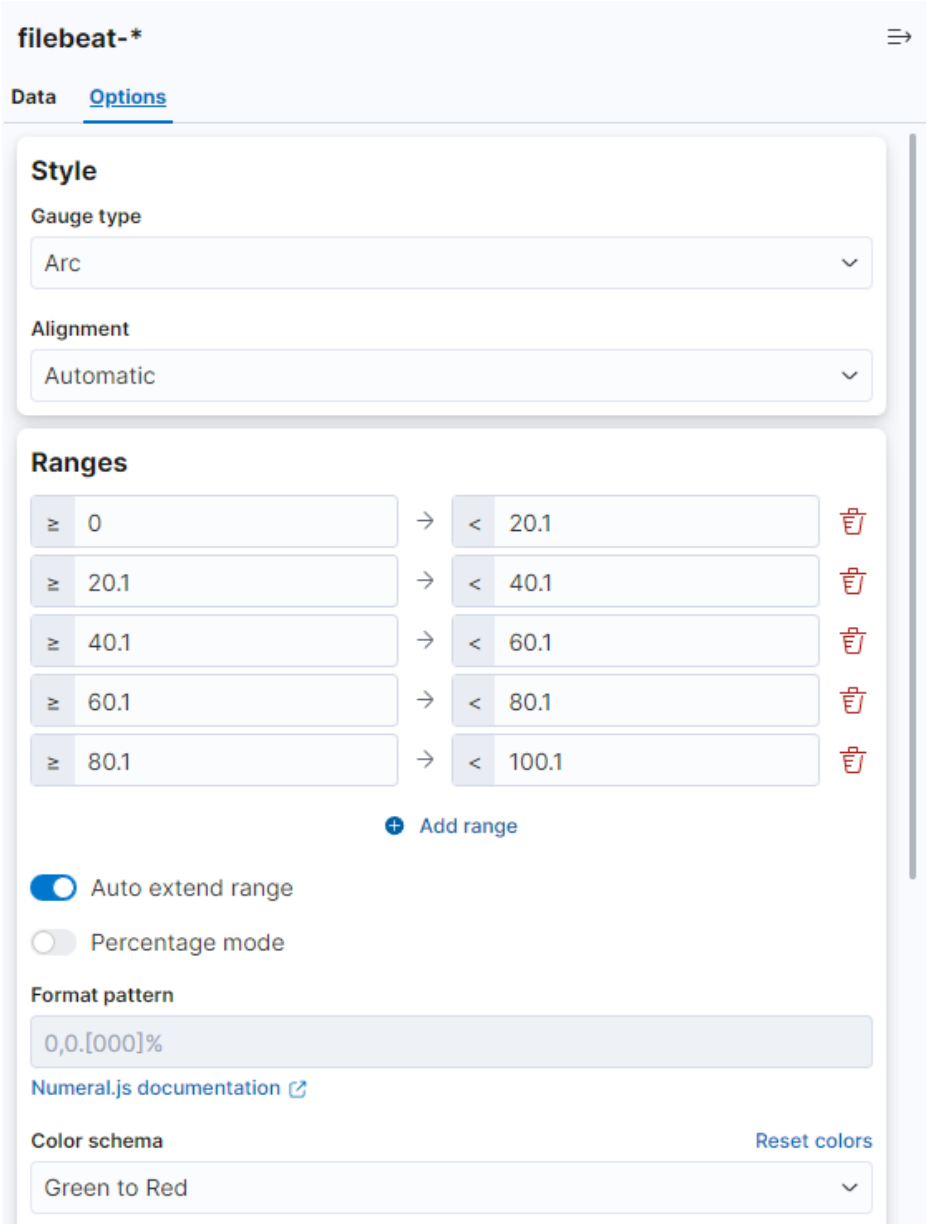


Figura 55. Configuración de los rangos para el medidor del campo IAQCI

Figura 56. Visualización del medidor del campo IAQCI

Por último, se debe guardar la visualización y añadirla al Dashboard creado con anterioridad.

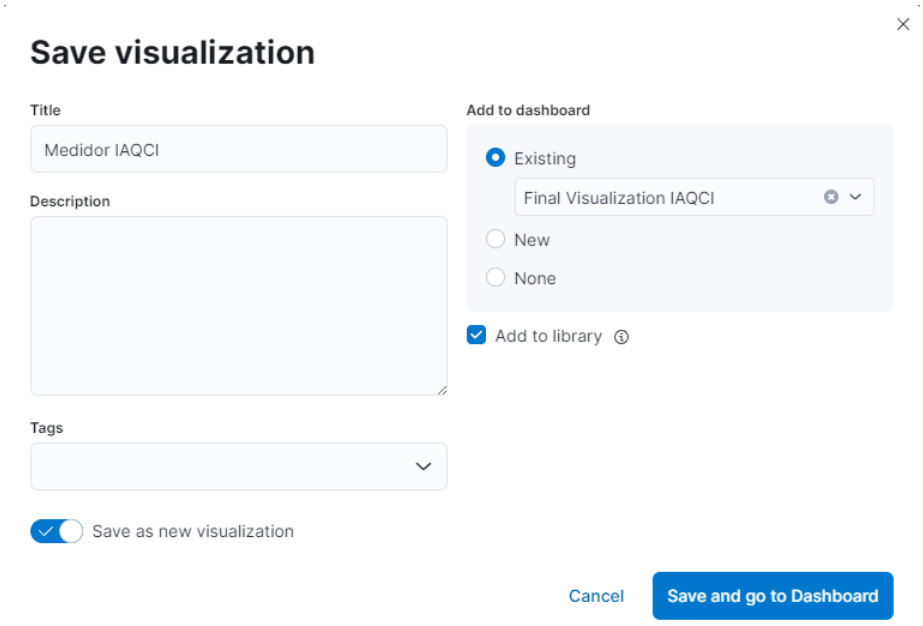


Figura 57. Ventana para guardar y añadir a un *Dashboard* una visualización en Elastic

Adicionalmente, se debe generar visualizaciones para los datos polvo, co2, humedad y temperatura. Dentro del *Dashboard* se debe organizar estas visualizaciones para que sean fáciles de comprender para el usuario.

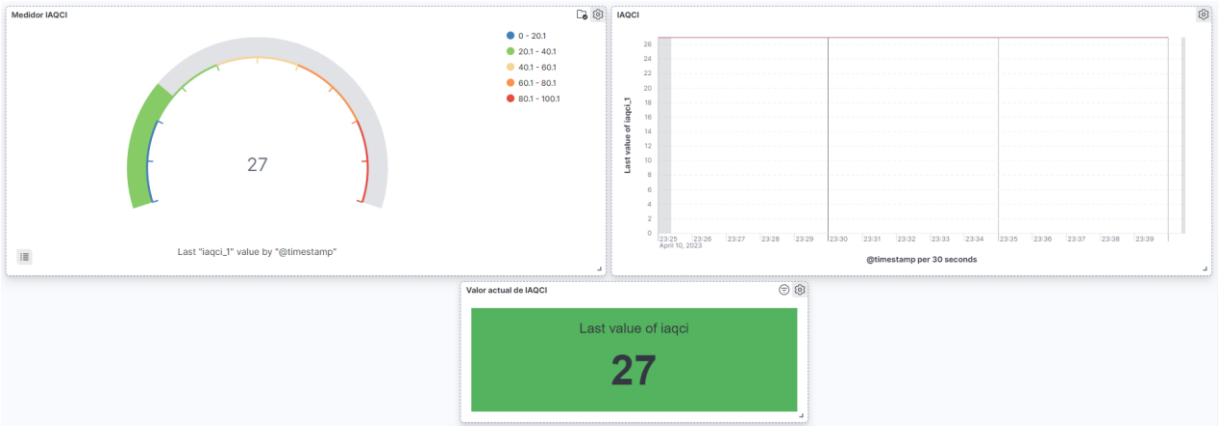


Figura 58. Visualización final del campo IAQCI en el *Dashboard*

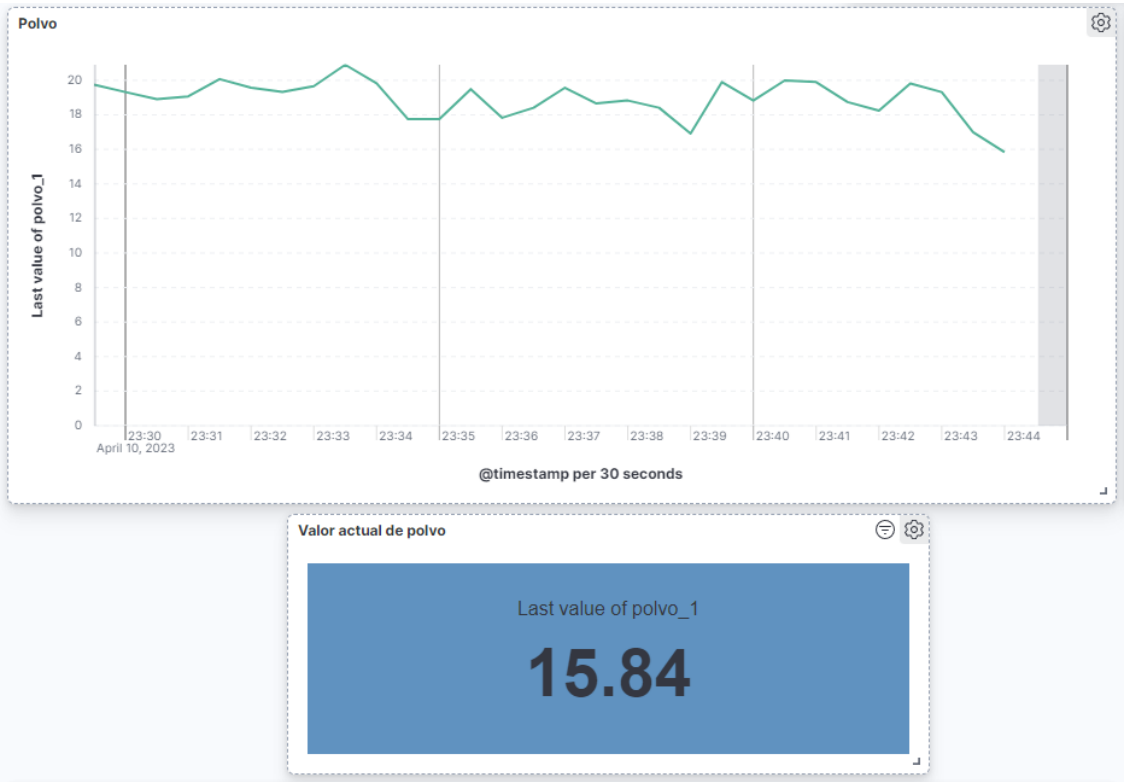


Figura 59. Visualización final del campo polvo en el *Dashboard*

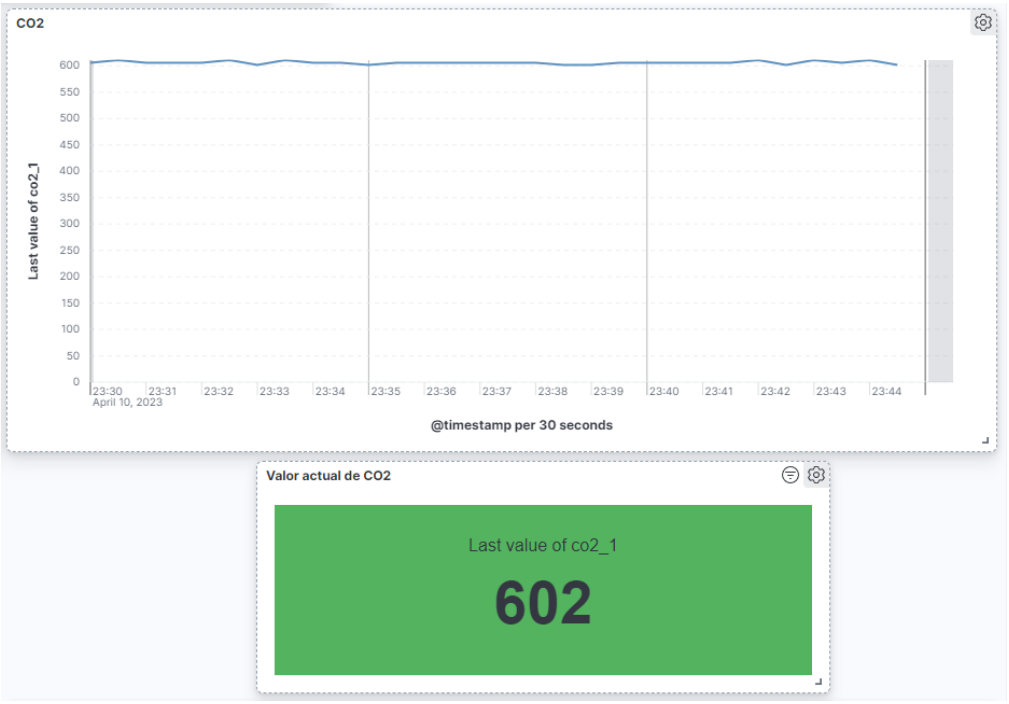


Figura 60. Visualización final del campo co2 en el *Dashboard*

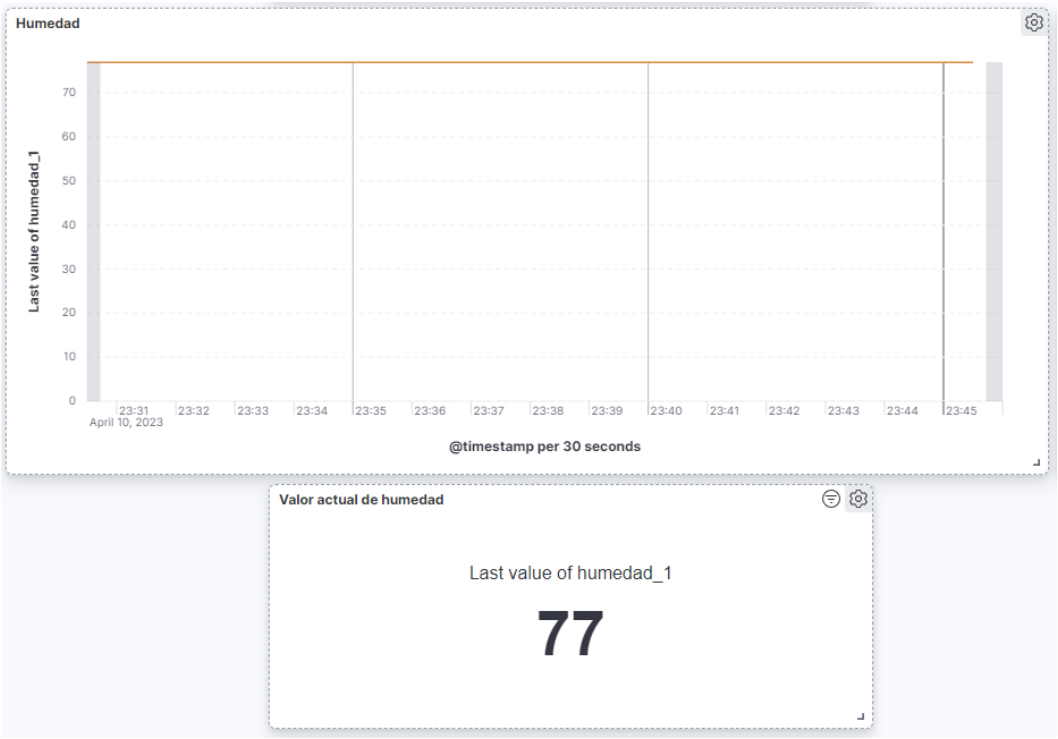


Figura 61. Visualización final del campo humedad en el *Dashboard*

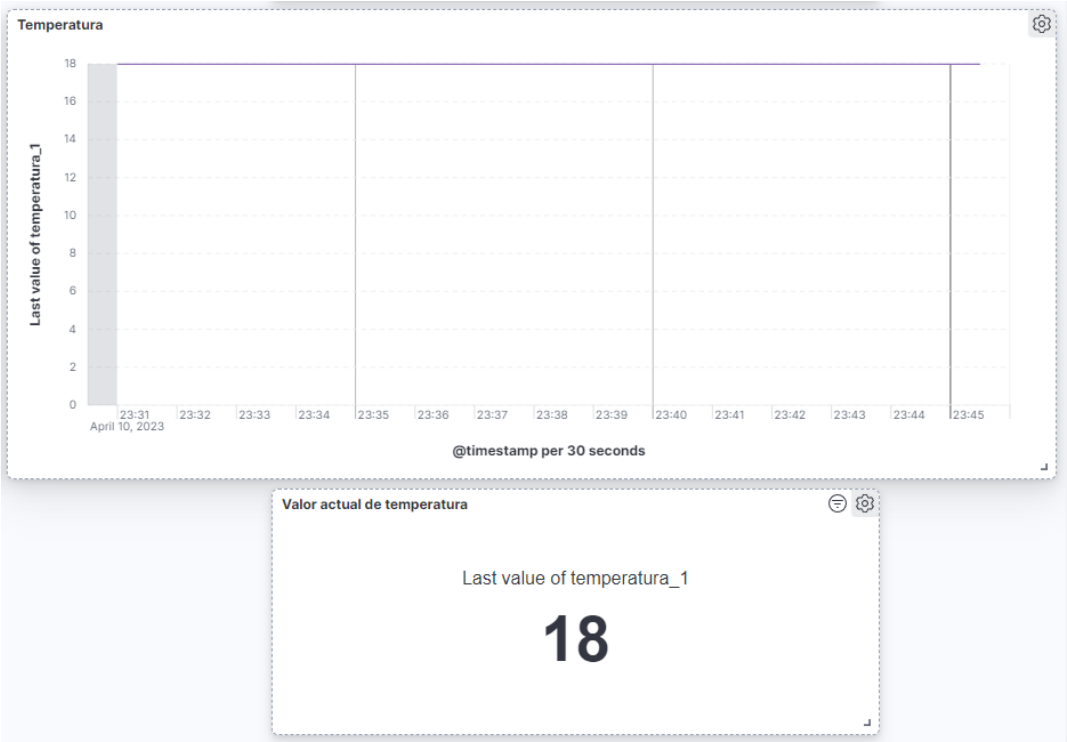


Figura 62. Visualización final del campo temperatura en el *Dashboard*

Para que el usuario sea alertado mediante correo electrónico de que hay un nivel bajo de calidad de aire dentro del ambiente se debe crear una alerta. Para esto, dentro de Elastic se debe acceder a *Discover* dentro de *Analytics* y seleccionar la opción *Alert*. A modo de prueba se configuró la condición para que la alerta sea enviada cuando el nivel de IAQCI sea igual o mayor a 30 como muestra la Figura 63, para la configuración final se toman en cuenta los umbrales establecidos en la Tablas 5 del Capítulo 1. La configuración para el envío del correo electrónico se muestra en la Figura 64.

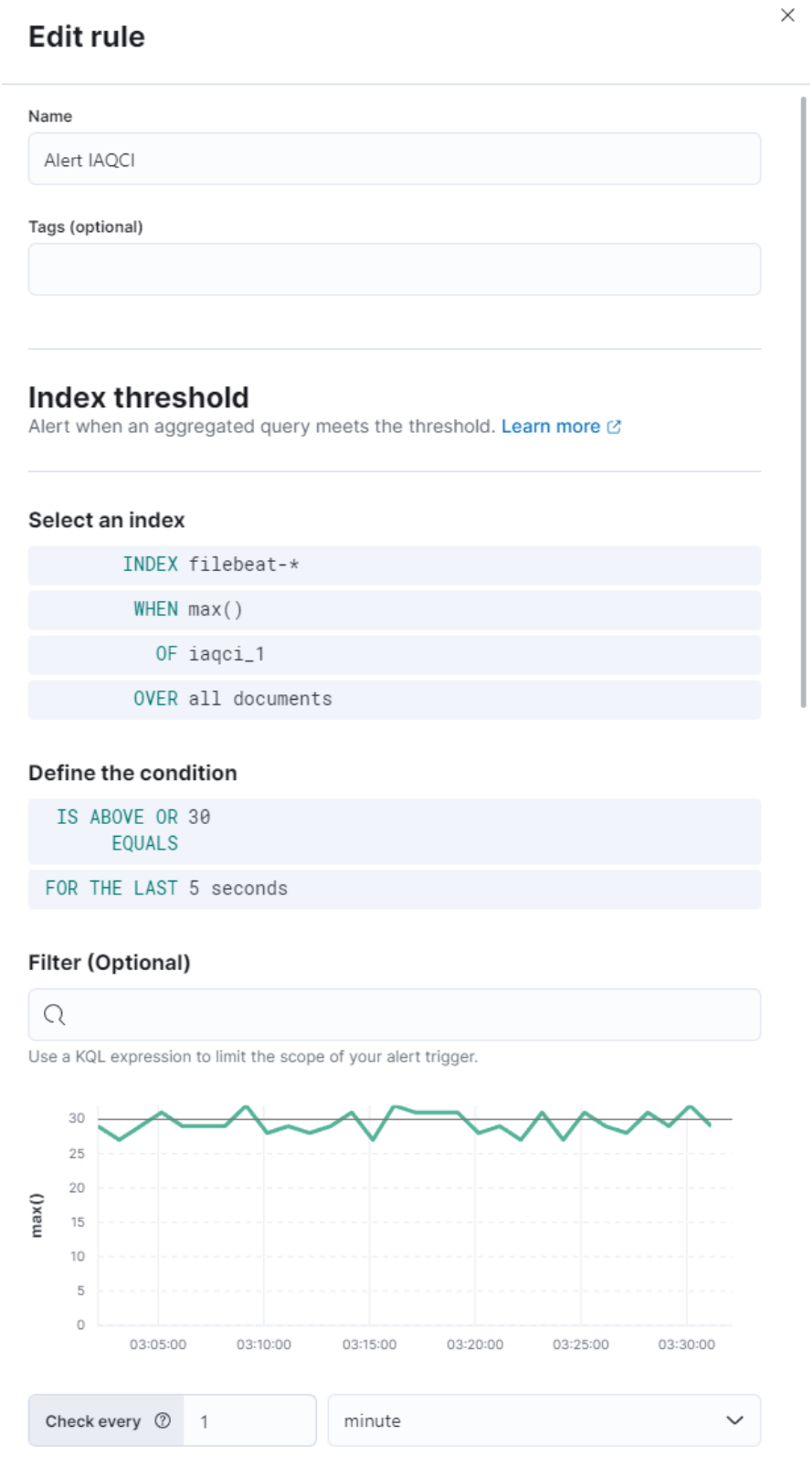


Figura 63. Configuración para crear una alerta de prueba en base al campo IAQCI

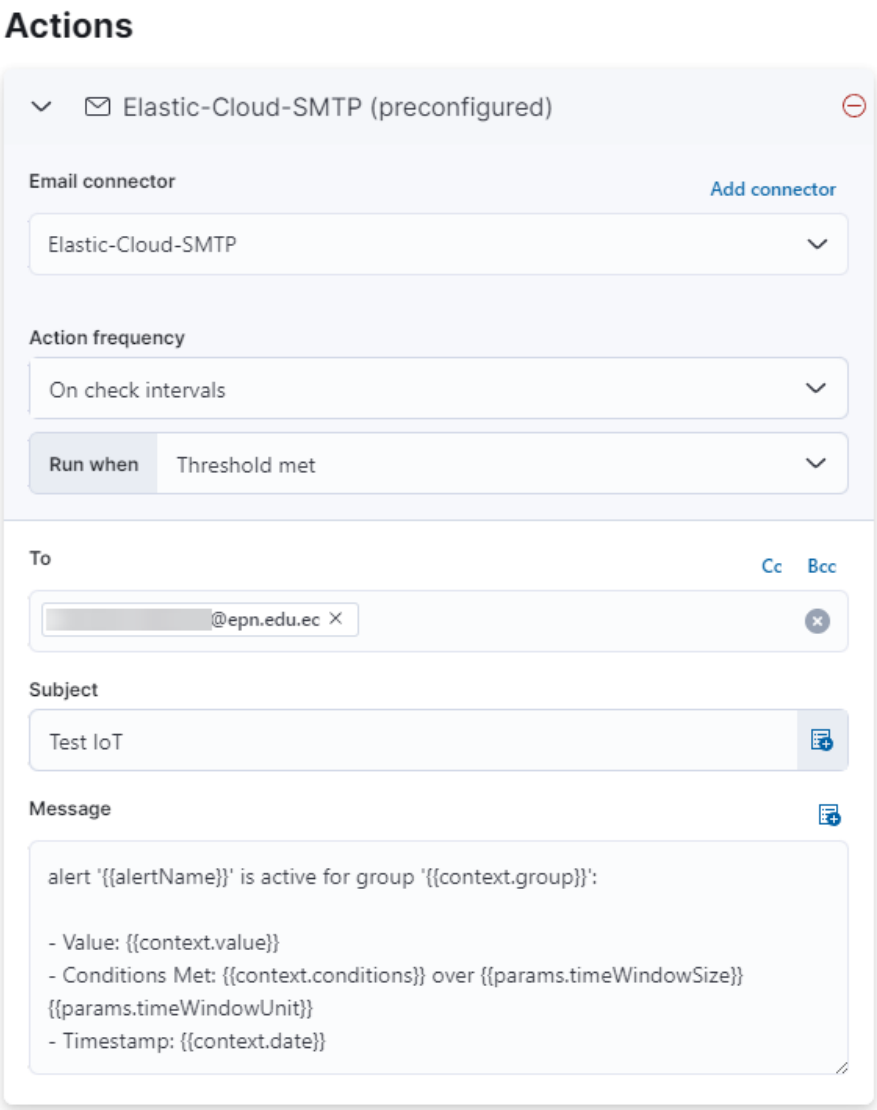


Figura 64. Configuración para notificar una alerta mediante un correo electrónico

3. VALIDACIÓN DEL PROTOTIPO

La validación del prototipo consiste en dos fases, la primera, que consiste en el despliegue del prototipo en ambientes internos, donde se valida la funcionalidad del mismo. Y la segunda, que consiste en una encuesta a los usuarios que utilizaron el prototipo a lo largo de su despliegue, que valida la usabilidad del mismo.

La Figura 65 muestra el lugar en el que fue desplegado el prototipo para las pruebas de funcionalidad, siendo este la sala de la casa de uno de los autores de este proyecto. El prototipo estuvo encendido recopilando datos todos los días a lo largo de dos semanas.

Figura 65. Ambiente interno donde se realizó el despliegue del prototipo

A continuación, se valida el despliegue del prototipo. En la Figura 66 se puede observar el nodo IoT, el cual utiliza un Arduino UNO como placa principal al que se integran los sensores GP2Y1010AU0F, DHT11 y MQ135 que permiten monitorizar los parámetros de calidad del aire en el ambiente interno.

Figura 66. Nodo IoT con los sensores GP2Y1010AU0F, DHT11 y MQ135 activos

En la Figura 67 se puede observar el módulo ESP-01 que se integra al Arduino UNO y permite el envío de los datos mediante una conexión WiFi, necesarios para calcular el valor del IAQCI.

Figura 67. Nodo IoT con el módulo ESP-01 activo

La Figura 68 muestra la Raspberry Pi 4, que actúa como bróker permitiendo recopilar la información que envía el nodo IoT.

Figura 68. Raspberry activa actuando como bróker

Se tiene un ordenador con el servicio de Filebeat activo (ver Figura 69), este actúa como subscriptor del bróker y recibe los datos. Además, calcula el valor de calidad del aire IAQCI y envía los datos a la nube de Elastic.

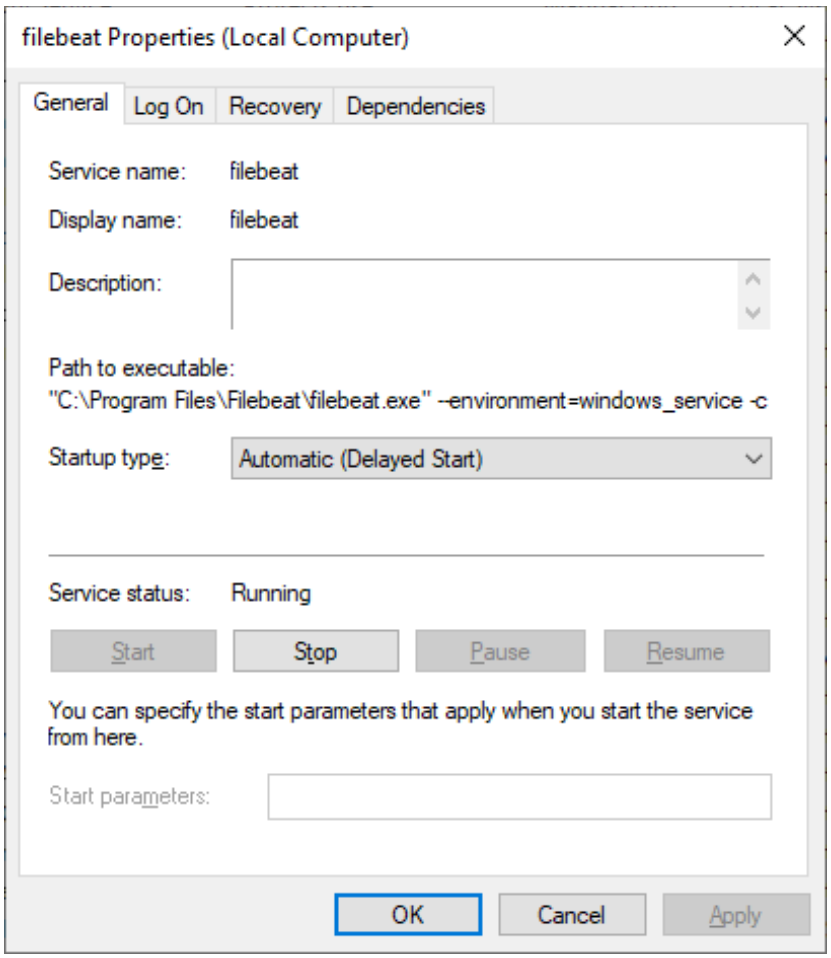


Figura 69. Servicio de Filebeat activo

ElasticSearch almacena los registros en tiempo real como se observa en la Figura 70.

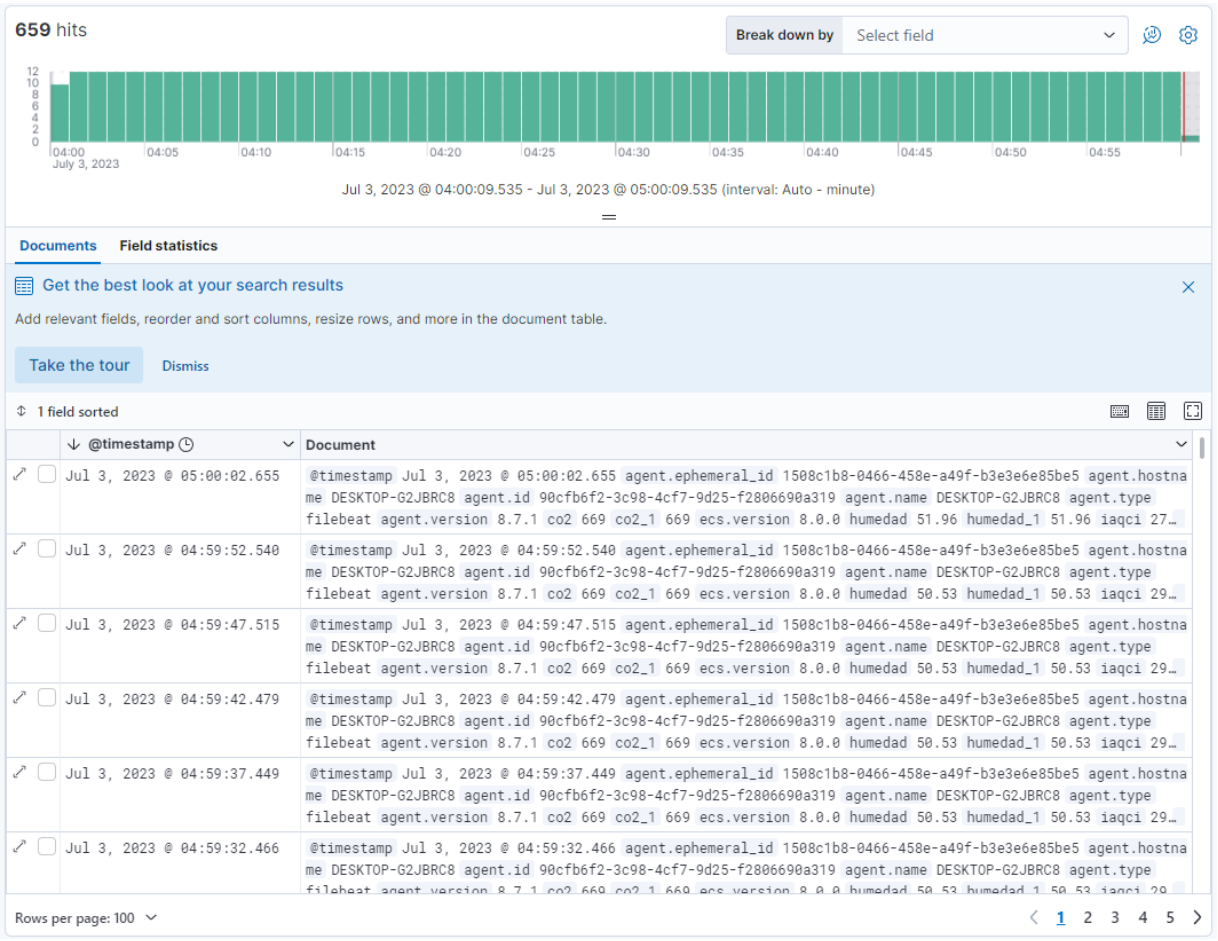


Figura 70. Datos guardados en la base de datos de Elastic

Cada registro almacena la información monitorizada por los sensores y el valor del cálculo del IAQCI como se detalla en la Figura 71.

Field	Value
k _id	69ExG4kBdmUzHpi75yeD
k _index	.ds-filebeat-8.7.1-2023.06.27-000001
# _score	-
@timestamp	Jul 3, 2023 @ 05:00:02.655
k agent.ephemeral_id	1508c1b8-0466-458e-a49f-b3e3e6e85be5
k agent.hostname	DESKTOP-G2JBRC8
k agent.id	90cfb6f2-3c98-4cf7-9d25-f2806690a319
k agent.name	DESKTOP-G2JBRC8
k agent.type	filebeat
k agent.version	8.7.1
k co2	669
# co2_1	669
k ecs.version	8.0.0
k humedad	51.96
# humedad_1	51.96
# iaqci	27
# iaqci_1	27
k input.type	mqtt
t message	21.27,51.96,669,10.1
mqtt.duplicate	false
# mqtt.message_id	0
# mqtt.qos	0
mqtt.retained	false
k mqtt.topic	casa/piso-1/sala/nodo-iot
k polvo	10.1
# polvo_1	10.1
k temperatura	21.27
# temperatura_1	21.27

Figura 71. Atributos de un dato guardado en la base de datos de Elastic

Se realiza la monitorización de los datos en tiempo real mediante Kibana como se muestra en la Figura 72.

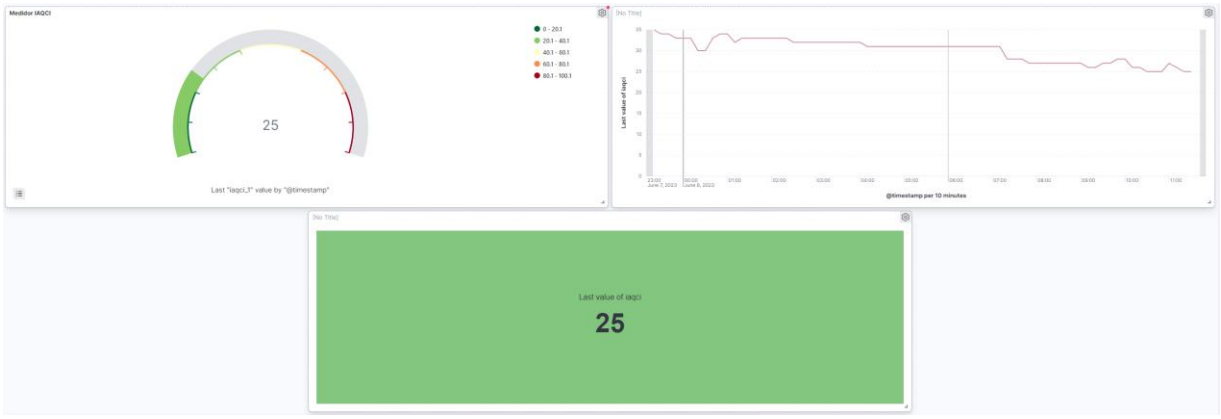


Figura 72. Visualización de los datos en tiempo real

Para analizar las gráficas de los valores obtenidos por el prototipo desplegado se tomaron los datos de un día en el horario de 23:00 a 11:00, a fin de corroborar que los niveles de contaminación al interior disminuyen cuando la actividad de personas en el ambiente interno es menor.

La Figura 73 muestra la visualización obtenida del IAQCI para el horario mencionado, como se puede observar conforme avanza el tiempo el parámetro del IAQCI disminuye, esto se debe a que los parámetros monitorizados no varían su medida durante este horario.

Figura 73. Visualización de los datos de IAQCI

La Figura 74 muestra la visualización de los datos del parámetro polvo, en este caso, podemos apreciar que la información obtenida varía mucho a pesar del horario en el que fueron capturados los datos, esto se debe a que siempre existe una cantidad de polvo en el ambiente y al ser partículas flotantes se mantienen en movimiento.

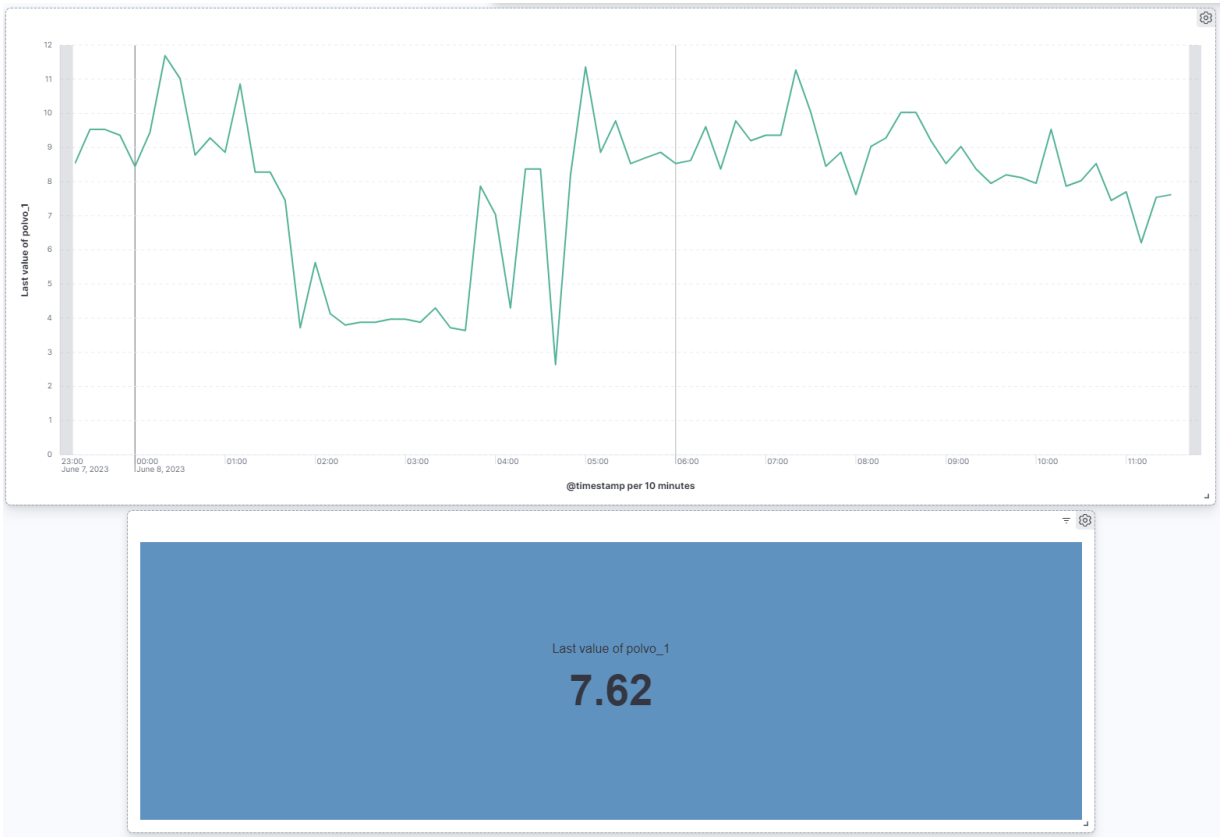


Figura 74. Visualización de los datos de polvo

La Figura 75 muestra la visualización de los datos de CO2, se puede apreciar que la cantidad de CO2 en el ambiente disminuye conforme avanza el tiempo, ya que en el horario mencionado no hay personas en esta ubicación.

Figura 75. Visualización de los datos de CO2

Las Figura 76 y 77 muestran la visualización de los datos de humedad y temperatura respectivamente, durante todo el periodo de monitorización se puede apreciar una tendencia plana de los mismos, debido a que estos datos no son fuertemente afectados por la presencia de personas en el ambiente interno si no por factores de ventilación.

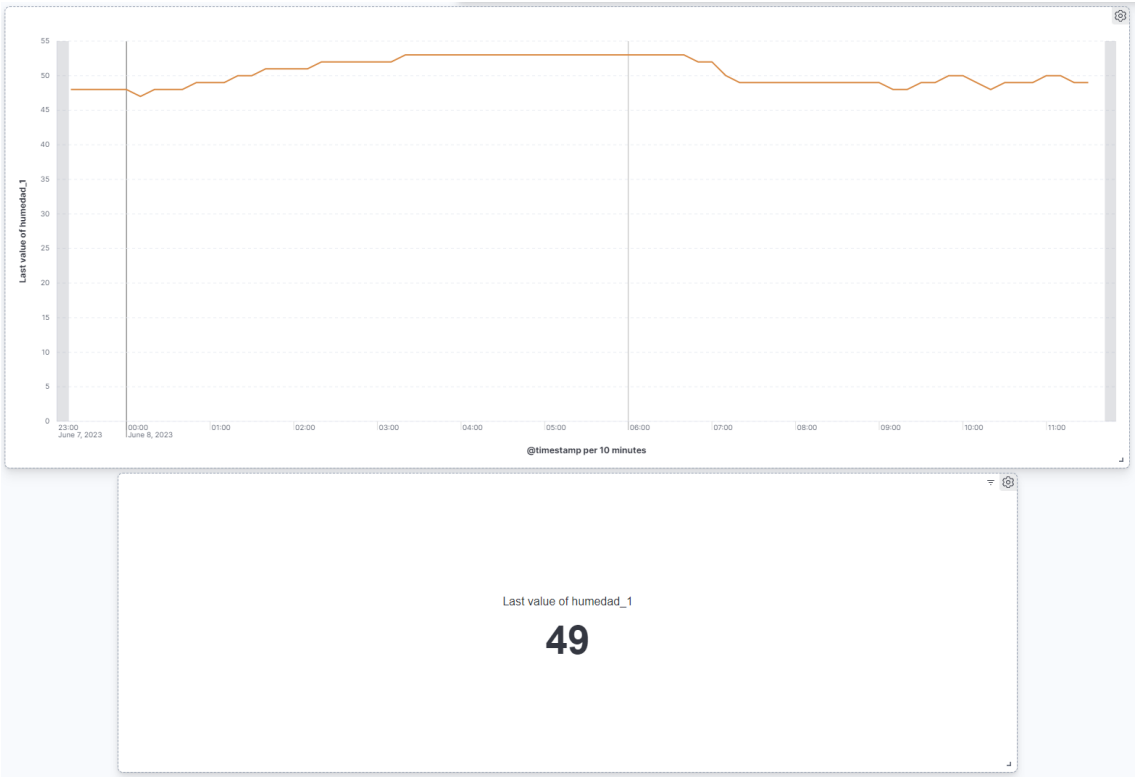


Figura 76. Visualización de los datos de humedad

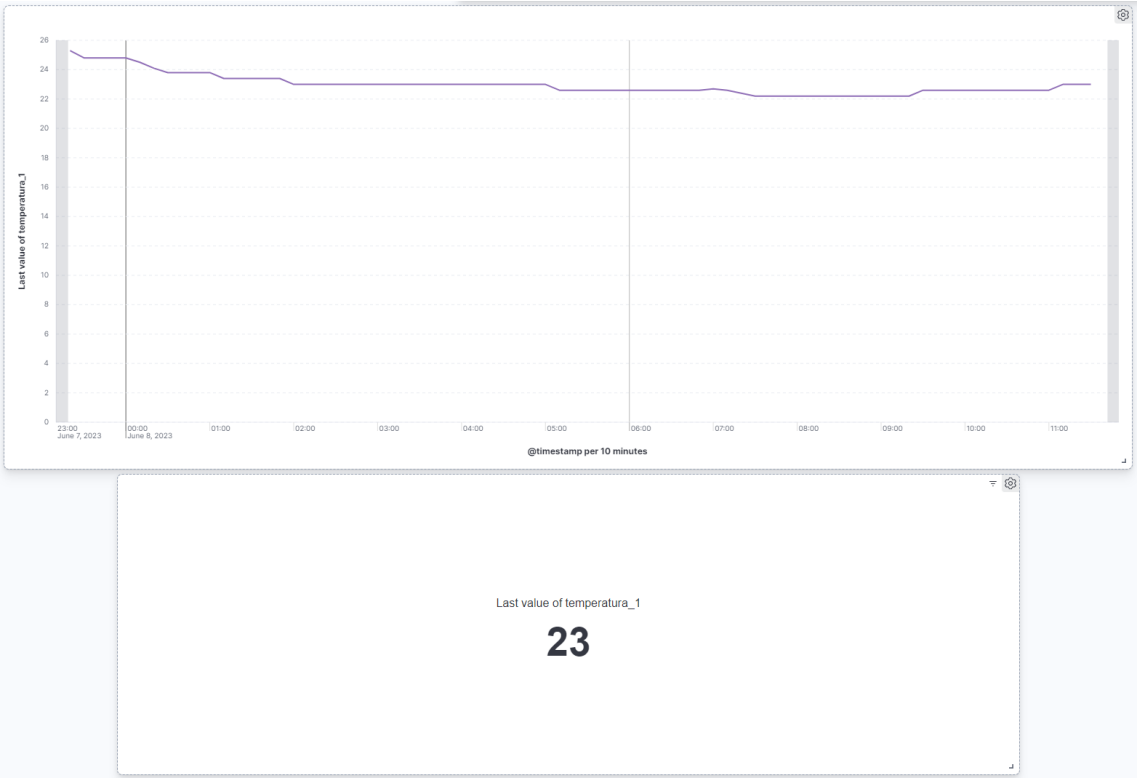


Figura 77. Visualización de los datos de temperatura

La alerta es enviada al usuario a través de correo electrónico cuando el valor de IAQCI sobrepasa el umbral configurado, esto se puede aprecia en la Figura 78.

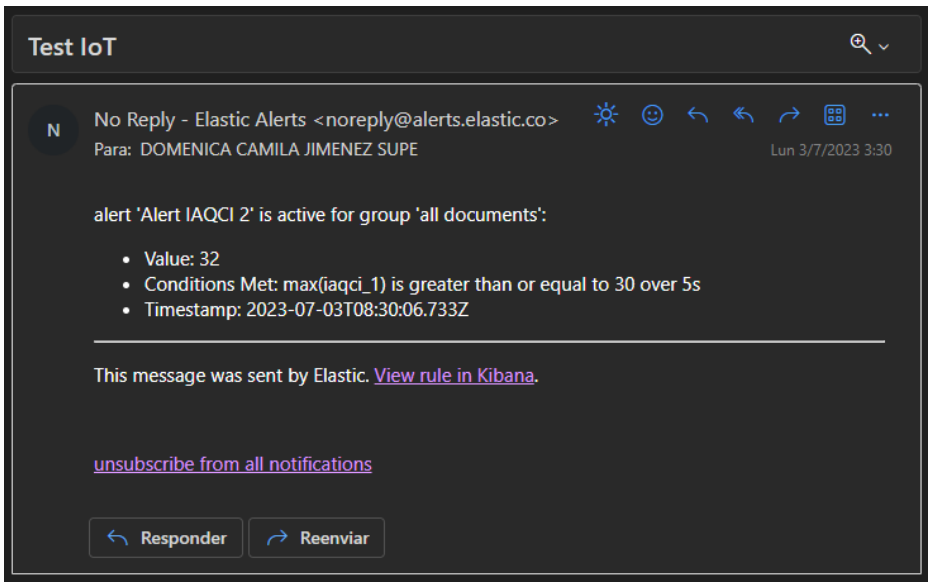


Figura 78. Correo de activación de alertas de Elastic

Adicionalmente, una vez implementado y desplegado el prototipo se procedió a verificar su usabilidad mediante el uso de dos metodologías, la Metodología de Evaluación de Prototipo Innovador [65], que realiza preguntas evaluando la satisfacción del usuario al utilizar el prototipo. Las preguntas se realizan tomando como referencia una escala de Likert que ofrece diferentes niveles de medición. También, se utiliza la Escala de Usabilidad del Sistema (SUS, por sus siglas en inglés *System Usability Scale*), es una herramienta que permite medir la usabilidad; de esta metodología, solo se emplea el significado de la puntuación obtenida. Las respuestas para las preguntas que se realizaron son las siguientes:

- De acuerdo
- Parcialmente de acuerdo
- En desacuerdo

Adicionalmente, cabe recalcar que este cuestionario fue aplicado a los miembros familiares de cada uno de los ambientes donde se desplegó el prototipo. En total de los dos ambientes donde se desplegó el prototipo se obtuvo el resultado de ocho encuestados voluntarios. Las preguntas que resolvió cada uno de los encuestados son las siguientes:

Pregunta 1: ¿Considera que el despliegue del prototipo en su entorno puede beneficiar a su salud?

Como se puede ver en la Figura 79, el 75% de los encuestados considera que la implementación del prototipo puede beneficiar su salud ya que les ha permitido tomar decisiones cuando una alerta relacionada con la calidad del aire les ha sido enviada.

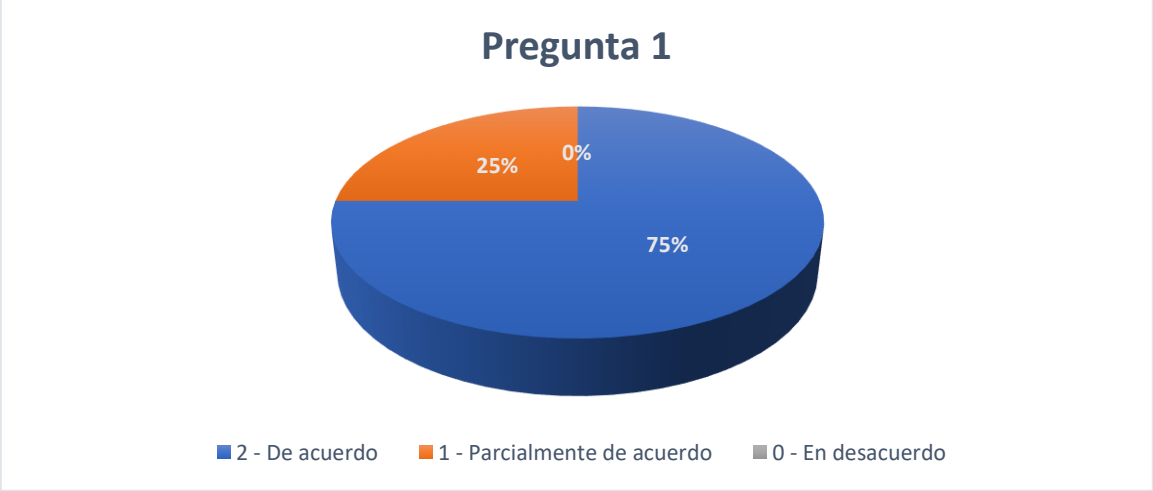


Figura 79. Resultados de la pregunta 1 para la validación del prototipo IoT

Pregunta 2: ¿Utilizaría el prototipo en su vida diaria?

La figura 80, muestra que el 75% de los encuestados utilizarían el prototipo en su vida diaria debido a que pasan mucho tiempo dentro del hogar y esto les ayudará a prevenir que los contaminantes en el aire atenten en contra de su salud.

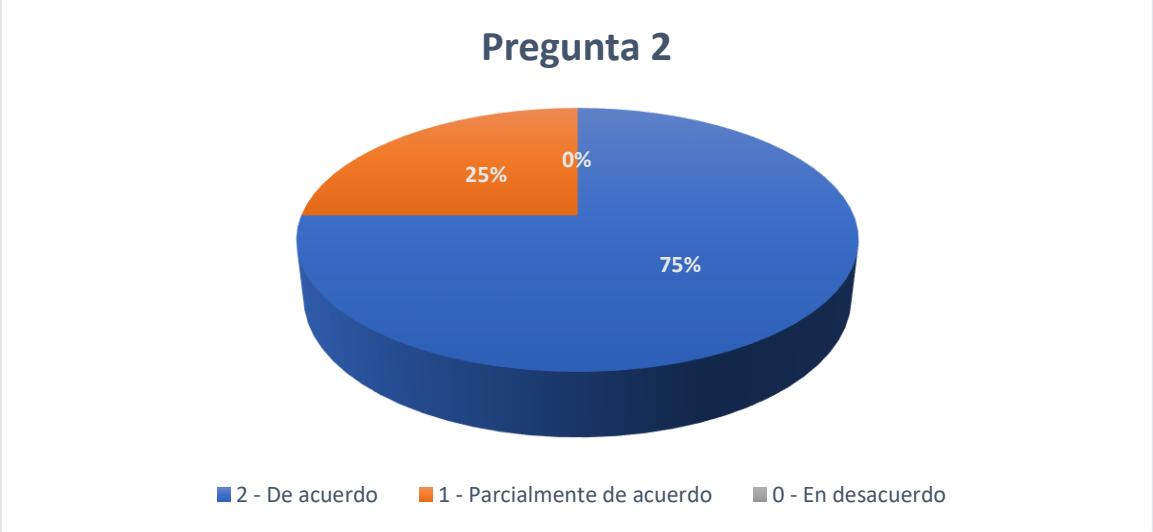


Figura 80. Resultados de la pregunta 2 para la validación del prototipo IoT

Pregunta 3: ¿Las gráficas presentadas en el prototipo son fáciles de interpretar?

En la Figura 81, se aprecia que el 100% de los encuestados entiende las gráficas presentadas en el aplicativo web. Esto debido a que las gráficas donde se despliega el IAQCI son intuitivas.

Figura 81. Resultados de la pregunta 3 para la validación del prototipo IoT

Pregunta 4: ¿Se perciben los beneficios de ser alertados en caso de que la calidad del aire disminuya?

Dentro de la Figura 82, se observa que el 87% de los encuestados ven beneficios de obtener una alerta cuando la calidad del aire disminuye, mencionaron que de este modo no tienen que revisar el aplicativo web seguidamente, tan solo revisarlo si se obtiene una alerta.

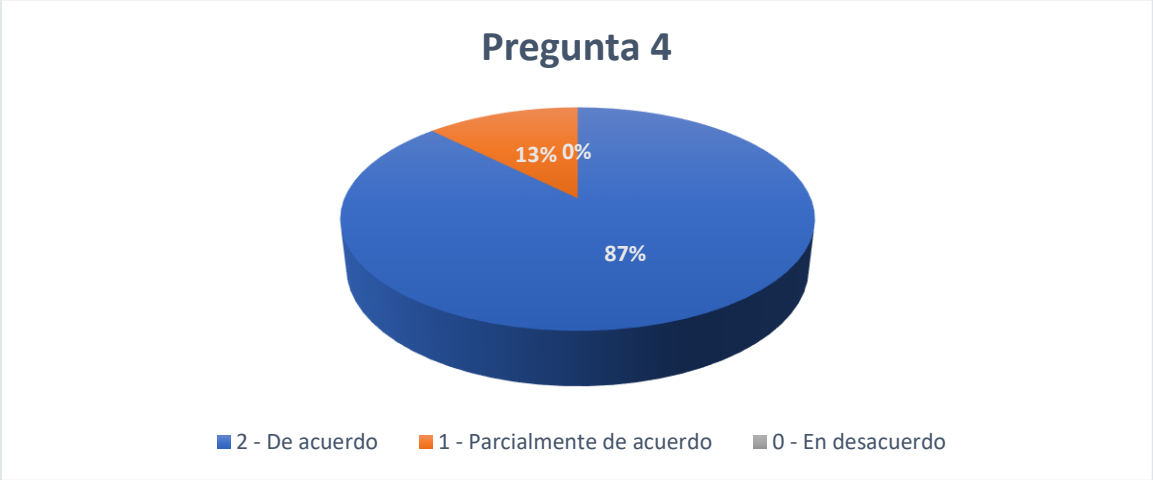


Figura 82. Resultados de la pregunta 4 para la validación del prototipo IoT

Pregunta 5: ¿Recomendaría el uso de este prototipo a otros usuarios?

Como se observa en la Figura 83, el 100% de usuarios está de acuerdo con la recomendación a otros usuarios para la implementación del prototipo IoT debido a lo fácil que es de usar y lo útil que les resultó dentro de la realidad donde fue implementado.

Figura 83. Resultados de la pregunta 5 para la validación del prototipo IoT

En base a las respuestas obtenidas de las 5 preguntas realizadas, se realizó un gráfico de barras (ver Figura 84) donde se puede apreciar un alto porcentaje de aprobación respecto al prototipo IoT. Este porcentaje es calculado al sumar todas las respuestas que posean un “De acuerdo” o “Parcialmente de acuerdo”, con puntajes de 2 y 1 respectivamente. Luego el valor obtenido se divide para 80 y se multiplica por 100, de este modo se obtuvo un porcentaje de aprobación del 94%. Al tener un puntaje mayor al 80% significa que tiene luz verde dentro de la usabilidad[66], esto quiere decir que posteriormente el prototipo IoT puede ser mejorado dentro de otra iteración para posteriormente ser implementado como un sistema.

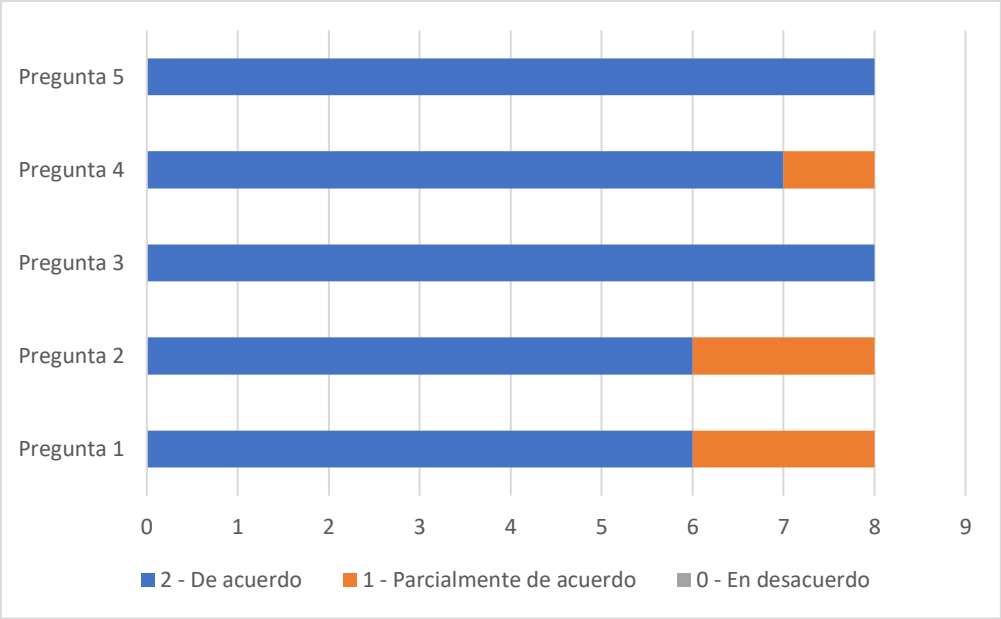


Figura 84. Diagrama de barras con los resultados de las preguntas realizadas a los encuestados

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

1. La solución implementada del prototipo IoT permite la monitorización constante de parámetros que ayudan a medir si existen niveles peligrosos de contaminación del aire en ambientes internos y de este modo alertar a los usuarios si es necesario tomar acciones pertinentes como ventilar el espacio.
2. El haber realizado la investigación de los diferentes índices de calidad del aire, permitió concluir qué índice se adaptaba al alcance del presente proyecto. Esto debido a que existen varios índices que consideran contaminantes que no se encuentran dentro de ambientes internos.
3. Para el presente proyecto utiliza el IAQCI, este índice se obtuvo mediante el cálculo del EPAQI y el índice de confort, que a su vez se obtuvieron en base a los parámetros monitorizados.
4. El diseño e implementación del prototipo IoT se realizó utilizando *constrained devices* y sistemas SoC. Este prototipo permite monitorizar parámetros de la calidad del aire tal como CO2, el polvo, la humedad y la temperatura. Estos datos son transferidos a un bróker MQTT que en este proyecto se ha considerado dentro de una Raspberry PI, la misma que a su vez envía a una aplicación web en Kibana donde se muestran los datos monitorizados.
5. La integración de los dispositivos *constrained devices* con el dispositivo SoC resulta en una solución económica para la monitorización de la calidad del aire, debido al precio que tienen cada uno de estos dispositivos.
6. Para que los datos sean sensados desde el nodo IoT se codificó un script en Arduino que agrupa los datos monitorizados por los *constrained devices* y los publica mediante el protocolo MQTT.
7. El servicio Filebeat, permitió ejecutar un script que se encarga de calcular el IAQCI con la información de los parámetros monitorizados, también permitió normalizar la información para enviarla al servicio de Elastic.
8. Los datos obtenidos se almacenaron en ElasticSearch y son utilizados para generar las visualizaciones con Kibana. Permitiendo a los usuarios acceder a la información mediante un aplicativo web de manera fácil e intuitiva.

9. Las alertas enviadas a los usuarios finales permiten que estos tomen decisiones para mantener un confort y nivel de salubridad interna, como, por ejemplo, ventilar el espacio en el que se encuentran, esto con el fin de mejorar la calidad del aire en el ambiente interno y alcanzar un IAQCI que esté dentro de los umbrales establecidos.

4.2. Recomendaciones

1. Durante las pruebas realizadas para el presente proyecto, la manipulación del nodo IoT provocó constantes desconexiones de los cables conectados a la *protoboard*, misma que se utilizó para conectar los *constrained devices* con la placa Arduino UNO. Por lo tanto, para prevenir posibles desconexiones, se recomienda incluir una placa de conexión y soldar todos los componentes a la misma.
2. A fin de que los datos que se obtengan sean más precisos, es necesario revisar la documentación técnica de los sensores para saber qué resistencia utilizar en los mismos.
3. Se recomienda desde el inicio del prototipado del nodo IoT analizar en función del número de sensores que formarán parte del nodo, cuántos pines digitales y analógicos serán necesarios, esto con el fin de elegir la placa que se adapte a este número de pines. En nuestro prototipo fue necesario reemplazar el módulo ESP8266 por la placa Arduino UNO, debido a que solo contaba con un pin analógico y se requerían dos.
4. Para instalar el servicio Filebeat, que en este caso facilita el procesamiento y la normalización de datos, se recomienda utilizar un dispositivo SoC que permita la virtualización de este servicio. En este proyecto no se pudo realizar el procesamiento en la Raspberry Pi debido a su arquitectura, por lo que se optó por añadir un ordenador.
5. A fin de reducir el costo en la solución del proyecto, se recomienda implementar de manera local el servicio Elastic debido a que los costos del uso de la nube son muy altos.
6. Para tener una validación adicional del funcionamiento del prototipo, se recomienda implementarlo en espacios internos donde exista una mayor aglomeración de personas como por ejemplo restaurantes, oficinas o aulas.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] J. Jose y T. Sasipraba, «Indoor air quality monitors using IOT sensors and LPWAN,» de 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2019.

[2] L. Zhao, W. Wu y S. Li, «Design and Implementation of an IoT-Based Indoor Air Quality Detector With Multiple Communication Interfaces,» IEEE Internet of Things Journal, vol. 6, p. 9621–9632, December 2019.

[3] S. Esfahani, P. Rollins, J. P. Specht, M. Cole y J. W. Gardner, «Smart City Battery Operated IoT Based Indoor Air Quality Monitoring System,» de 2020 IEEE SENSORS, 2020.

[4] D. L. Margall, «Post Covid: la nueva era del Teletrabajo en el sector asegurador,» 2020. [En línea]. Available: http://diposit.ub.edu/dspace/bitstream/2445/172017/1/TFM-DEAF-267_Lafer.pdf. [Último acceso: 2 Julio 2021].

[5] H. Valenzuela-Garcia, «TELETRABAJO Y AMPLIFICACIÓN DE LA DESIGUALDAD EN LA SOCIEDAD POST-PANDEMIA ESPAÑOLA.,» Revista Andaluza de Antropología, p. 14–36, 2020.

[6] G. Marques, J. Saini, M. Dutta, P. K. Singh y W.-C. Hong, «Indoor Air Quality Monitoring Systems for Enhanced Living Environments: A Review toward Sustainable Smart Cities,» Sustainability, vol. 12, p. 4024, May 2020.

[7] W. H. Organization, «WHO guidelines for indoor air quality: selected pollutants,» 2010. [En línea]. Available: https://www.euro.who.int/__data/assets/pdf_file/0009/128169/e94535.pdf. [Último acceso: 13 April 2022].

[8] A. Fonseca, I. Abreu, M. J. Guerreiro y N. Barros, «Indoor Air Quality in Healthcare Units—A Systematic Literature Review Focusing Recent Research,» Sustainability, vol. 14, p. 967, January 2022.

[9] Kanchan, A. K. Gorai y P. Goyal, «A Review on Air Quality Indexing System,» Asian Journal of Atmospheric Environment, vol. 9, p. 101–113, June 2015.

[10] H.-H. Kim, M.-J. Kwak, K.-J. Kim, Y.-K. Gwak, J.-H. Lee y H.-H. Yang, «Evaluation of IAQ Management Using an IoT-Based Indoor Garden,» International Journal of Environmental Research and Public Health, vol. 17, p. 1867, March 2020.

[11] J. Kang y K.-I. Hwang, «A Comprehensive Real-Time Indoor Air-Quality Level Indicator,» Sustainability, vol. 8, p. 881, September 2016.

[12] B. Bishoi, A. Prakash y V. K. Jain, «A Comparative Study of Air Quality Index Based on Factor Analysis and US-EPA Methods for an Urban Environment,» Aerosol and Air Quality Research, vol. 9, p. 1–17, 2009.

[13] S. Parey, «SIMULATION OF INDOOR AIR QUALITY IN AN OFFICE BUILDINGFLOOR : A FIRST CASE STUDY,» 1999.

[14] N. W. Service, «What is the heat index?,» [En línea]. Available: https://www.weather.gov/ama/heatindex. [Último acceso: 7 September 2022].

[15] ITIEFFE, «Heat index,» [En línea]. Available: https://www.itieffe.com/en/air-conditioning/air-conditioning-tables/heat-index/. [Último acceso: 7 September 2022].

[16] AIR QUALITY PROSESCAN, «SENSORES DE CALIDAD DEL AIRE BASADOS EN IOT - 4 CRITERIOS,» 30 Septiembre 2022. [En línea]. Available: https://www.airqualityprosescan.com/blog/sensoresinteligentes2. [Último acceso: 24 Mayo 2023].

[17] K. K. Patel y S. M. Patel, «Internet of Things-IOT : Definition , Characteristics , Architecture , Enabling Technologies , Application & Future Challenges,» 2016.

[18] IoT For All, «Air Quality Index (AQI) Basics,» 2020. [En línea]. Available: https://www.iotforall.com/air-pollution-monitoring-using-iot-can-help-us-breathe-easier. [Último acceso: 8 June 2022].

[19] R. A. Rahman y B. Shah, «Security analysis of IoT protocols: A focus in CoAP,» de 2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC), 2016.

[20] G. Caiza, E. S. Llamuca, C. A. Garcia, F. Gallardo-Cardenas, D. Lanas y M. V. Garcia, «Industrial Shop-Floor Integration Based on AMQP protocol in an IoT Environment,» de 2019 IEEE Fourth Ecuador Technical Chapters Meeting (ETCM), 2019.

[21] R. A. Atmoko, R. Riantini y M. K. Hasin, «IoT real time data acquisition using MQTT protocol,» Journal of Physics: Conference Series, vol. 853, p. 012003, May 2017.

[22] P. Pieroni, R. Concetti, A. Belli y L. Palma, «Amazon, Google and Microsoft Solutions for IoT: Architectures and a Performance Comparison,» IEEE Access , vol. 1, nº 1, 2019.

[23] R. Krishna, A. Priyadarshini, A. Jha, B. Appasani, A. Srinivasulu y N. Bizon, «State-of-the-Art Review on IoT Threats and Attacks: Taxonomy, Challenges and Solutions,» Sustainability, vol. 13, nº 16, 2021.

[24] F. Hays, A guide to the SCRUM BODYOF KNOWLEDGE (SBOKTM Guide): A comprehensive Guide to Deliver Project using Scrum, SCRUMstudy, 2016.

[25] D. Ariza, «Efectividad de la gestión de los proyectos: una perspectiva constructivista,» 4 Agosto 2017. [En línea]. Available: https://www.scielo.cl/pdf/oyp/n22/0718-2805-oyp-22-0075.pdf. [Último acceso: 4 Marzo 2023].

[26] J. S. Ken Schwaber, «The 2020 Scrum Guide,» 2020. [En línea]. Available: https://scrumguides.org/scrum-guide.html. [Último acceso: 9 June 2022].

[27] A. Oyegoke, «The constructive research approach in project management research,» International Journal of Managing Projects in Business, vol. 4, p. 573–595, September 2011.

[28] Arduino, [En línea]. Available: https://arduino.cl/arduino-uno/. [Último acceso: 5 Marzo 2023].

[29] Olimex, «TECHNICAL DATA MQ-135 GAS SENSOR,» [En línea]. Available: https://www.olimex.com/Products/Components/Sensors/Gas/SNS-MQ135/resources/SNS-MQ135.pdf. [Último acceso: 29 Septiembre 2022].

[30] Mouser, «DHT11 Humidity & Temperature Sensor,» [En línea]. Available: https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf. [Último acceso: 29 Septiembre 2022].

[31] Sharp, «GP2Y1010AU0F,» [En línea]. Available: https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y1010au_e.pdf. [Último acceso: 29 Septiembre 2022].

[32] B. Mishra y A. Kertesz, «The Use of MQTT in M2M and IoT Systems: A Survey,» IEEE Access, vol. 8, p. 201071–201086, 2020.

[33] E. Al-Masri, K. R. Kalyanam, J. Batts, J. Kim, S. Singh, T. Vo y C. Yan, «Investigating Messaging Protocols for the Internet of Things (IoT),» IEEE Access, vol. 8, p. 94880–94911, 2020.

[34] S. Elhadi, A. Marzak, N. Sael y S. Merzouk, «Comparative Study of IoT Protocols,» SSRN Electronic Journal, 2018.

[35] D. Glaroudis, A. Iossifides y P. Chatzimisios, «Survey, comparison and research challenges of IoT application protocols for smart farming,» Computer Networks, vol. 168, p. 107037, February 2020.

[36] HiveMQ, «MQTT & MQTT 5 Essentials,» [En línea]. Available: https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf. [Último acceso: 7 Noviembre 2022].

[37] Catchpoint Systems, «MQTT Broker,» [En línea]. Available: https://www.catchpoint.com/network-admin-guide/mqtt-broker. [Último acceso: 22 Enero 2023].

[38] A. R. R. Xavier, «ANÁLISIS COMPARATIVO A NIVEL TRANSACCIONAL DE BROKERS MQTT (MOSQUITTO, MOSCA Y EMQ) CON RESPECTO A LA DISPONIBILIDAD EN INFRAESTRUCTURAS IoT ANTE ATAQUES DDoS,» 2020. [En línea]. Available: https://bibdigital.epn.edu.ec/bitstream/15000/21374/1/CD%2010446.pdf. [Último acceso: 22 Enero 2023].

[39] F. Delporte, «MQTT on Raspberry Pi: Send Sensor Data to HiveMQ Cloud with Java and Pi4J,» HiveMQ, 10 Diciembre 2021. [En línea]. Available: https://www.hivemq.com/blog/mqtt-raspberrypi-part01-sensor-data-hivemqcloud-java-pi4j/. [Último acceso: 22 Enero 2023].

[40] IBM, «Resumen de Publicación/suscripción,» 24 Agosto 2022. [En línea]. Available: https://www.ibm.com/docs/es/integration-bus/10.0?topic=applications-publishsubscribe-overview. [Último acceso: 04 Diciembre 2022].

[41] Elastic, «Elastic Stack,» [En línea]. Available: https://www.elastic.co/es/elastic-stack/. [Último acceso: 22 Febrero 2023].

[42] Elastic, «¿Qué es el ELK Stack?,» [En línea]. Available: https://www.elastic.co/es/what-is/elk-stack. [Último acceso: 22 Febrero 2023].

[43] Amazon, «¿Qué es la API RESTful?,» [En línea]. Available: https://aws.amazon.com/es/what-is/restful-api/#seo-faq-pairs#what-is-restful-api. [Último acceso: 22 Febrero 2023].

[44] Elastic, «Elasticsearch,» [En línea]. Available: https://www.elastic.co/es/elasticsearch/. [Último acceso: 22 Febrero 2023].

[45] Elastic, «Alerting,» [En línea]. Available: https://www.elastic.co/es/what-is/kibana-alerting. [Último acceso: 23 Febrero 2023].

[46] Elastic, «Kibana,» [En línea]. Available: https://www.elastic.co/es/kibana/. [Último acceso: 23 Febrero 2023].

[47] Elastic, «Beats,» [En línea]. Available: https://www.elastic.co/es/beats/. [Último acceso: 23 Febrero 2023].

[48] Ionos, «¿Qué es Jira? Todo lo que necesitas saber sobre el software de gestión de proyectos,» 11 Abril 2022. [En línea]. Available: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-jira/. [Último acceso: 5 Marzo 2023].

[49] Arduino, «WiFiEsp,» 10 Diciembre 2022. [En línea]. Available: http://www.arduinolibraries.info/libraries/wi-fi-esp. [Último acceso: 12 Diciembre 2022].

[50] Arduino, «MQ135,» 10 Diciembre 2022. [En línea]. Available: https://www.arduinolibraries.info/libraries/mq135. [Último acceso: 10 Diciembre 2022].

[51] Arduino, «DHT sensor library,» 10 Diciembre 2022. [En línea]. Available: https://www.arduinolibraries.info/libraries/dht-sensor-library. [Último acceso: 10 Diciembre 2022].

[52] N. O'Leary, «Arduino Client for MQTT,» [En línea]. Available: https://pubsubclient.knolleary.net/. [Último acceso: 15 Enero 2023].

[53] Arduino, «PubSubClient,» 15 Enero 2023. [En línea]. Available: https://www.arduinolibraries.info/libraries/pub-sub-client. [Último acceso: 15 Enero 2023].

[54] Eclipse Mosquitto, «Eclipse Mosquitto™ An open source MQTT broker,» [En línea]. Available: https://www.mosquitto.org/. [Último acceso: 22 Enero 2023].

[55] Eclipse, «mosquitto_pub man page,» [En línea]. Available: https://mosquitto.org/man/mosquitto_pub-1.html. [Último acceso: 22 Enero 2023].

[56] Eclipse, «Documentation,» [En línea]. Available: https://mosquitto.org/documentation/. [Último acceso: 22 Enero 2023].

[57] Eclipse, «mosquitto_sub man page,» [En línea]. Available: https://mosquitto.org/man/mosquitto_sub-1.html. [Último acceso: 22 Enero 2023].

[58] Particle, «RASPBERRY PI DATASHEET,» [En línea]. Available: https://docs.particle.io/assets/pdfs/raspberry-pi/datasheet.pdf. [Último acceso: 09 Septiembre 2022].

[59] The Raspberry Pi Foundation, «About us,» [En línea]. Available: https://www.raspberrypi.org/about/. [Último acceso: 26 Enero 2023].

[60] D. Garn, «How to use systemctl to manage Linux services,» 17 Mayo 2022. [En línea]. Available: https://www.redhat.com/sysadmin/linux-systemctl-manage-services. [Último acceso: 26 Enero 2023].

[61] Elastic, «Filebeat overview,» [En línea]. Available: https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html. [Último acceso: 15 Febrero 2023].

[62] Elastic, «Beats,» GitHub, 2019. [En línea]. Available: https://github.com/elastic/beats/tree/main/libbeat. [Último acceso: 21 Febrero 2023].

[63] Elastic, «MQTT input,» [En línea]. Available: https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-input-mqtt.html. [Último acceso: 21 Febrero 2023].

[64] Elastic, «Filebeat quick start: installation and configuration,» [En línea]. Available: https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-installation-configuration.html#collect-log-data. [Último acceso: 21 Febrero 2023].

[65] J. A. A. J. H. Sarraipa, «Metodología De Evaluación De Prototipo Innovador,» 2019. [En línea]. Available: https://acacia.red/wp-content/uploads/2019/07/Gu%C3%ADa-Metodologi%CC%81a-de-evaluaci%C3%B3n-de-prototipo-innovador.pdf. [Último acceso: 6 Julio 2023].

[66] N. Thomas, «How To Use The System Usability Scale (SUS) To Evaluate The Usability Of Your Website,» [En línea]. Available: https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/. [Último acceso: 6 Julio 2023].

[67] G. W. H. Organization, WHO global air quality guidelines. Particulate matter (PM2.5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide, G. W. E. C. f. E. Bonn y Health, Edits., WHO Press, 2021.

[68] A. Now, «Air Quality Index (AQI) Basics,» s.f.. [En línea]. Available: https://www.airnow.gov/aqi/aqi-basics/. [Último acceso: 18 May 2022].

[69] H. Wang, D. Xiong, P. Wang y Y. Liu, «A Lightweight XMPP Publish/Subscribe Scheme for Resource-Constrained IoT Devices,» IEEE Access, vol. 5, p. 16393–16405, 2017.

[70] G. M. B. Oliveira, D. C. M. Costa, R. J. B. V. M. Cavalcanti, J. P. P. Oliveira, D. R. C. Silva, M. B. Nogueira y M. C. Rodrigues, «Comparison Between MQTT and WebSocket Protocols for IoT Applications Using ESP8266,» de 2018 Workshop on Metrology for Industry 4.0 and IoT, 2018.

[71] R. F. et al., «Hypertext Transfer Protocol – HTTP/1.1,» 1999. [En línea]. Available: https://www.ietf.org/rfc/rfc2616.txt. [Último acceso: 6 April 2022].

[72] Geeks for Geeks, «3 layer IoT architecture,» 13 Abril 2021. [En línea]. Available: https://www.geeksforgeeks.org/3-layer-iot-architecture/. [Último acceso: 13 Septiembre 2022].

[73] P. Sethi y S. R. Sarangi, «Internet of Things: Architectures, Protocols, and Applications,» Journal of Electrical and Computer Engineering, vol. 2017, p. 1–25, 2017.

[74] Amathulhadishakara y M. Hasan, «Solutions of common challenges in IoT,» November 2017.

[75] MongoDB, «What is IoT Architecture?,» [En línea]. Available: https://www.mongodb.com/cloud-explained/iot-architecture#what-is-iot-architecture. [Último acceso: 13 Septiembre 2022].

[76] Espressif, «ESP8266 Technical Reference,» [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf. [Último acceso: 29 Septiembre 2022].

[77] MQTT, «MQTT: The Standard for IoT Messaging,» [En línea]. Available: https://mqtt.org/. [Último acceso: 26 10 2022].

[78] Random Nerd Tutorials, «Installing ESP8266 Board in Arduino IDE (Windows, Mac OS X, Linux),» [En línea]. Available: https://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/. [Último acceso: 10 Diciembre 2022].

[79] Bejob, «QUÉ ES LA PROGRAMACIÓN CON ARDUINO Y PARA QUÉ SIRVE,» 14 Febrero 2017. [En línea]. Available: https://www.bejob.com/que-es-la-programacion-con-arduino-y-para-que-sirve/. [Último acceso: 13 Noviembre 2022].

[80] E. Brown, «Linux and Open Source Hardware for IoT,» Linux, 27 Septiembre 2016. [En línea]. Available: https://www.linux.com/news/linux-and-open-source-hardware-iot/. [Último acceso: 13 Noviembre 2022].

ANEXO I

Historias de usuario: Tesis - HU

ANEXO II

Código fuente para el nodo IoT: Final.ino · main · TesisEPN / IAQCI · GitLab

ANEXO III

Código fuente para Filebeat.yml: filebeat.yml · main · TesisEPN / IAQCI · GitLab

ANEXO IV

Código fuente para el archivo JavaScript utilizado en Filebeat: iaqci.js · main · TesisEPN / IAQCI · GitLab