UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de Ingeniero de Sistemas.

PROYECTO TÉCNICO:

"Telemetría como servicio (TEaaS) basada en el protocolo ZigBee y Fog Computing para monitoreo de la calidad del aire."

AUTORES:

Ivan Fernando Yuquilima Albarado

Christian Hernán Argudo Vásconez

TUTOR:

Ing. Pablo Leonidas Gallegos Segovia

CUENCA – ECUADOR
OCTUBRE, 2016

CESIÓN DE DERECHOS DE AUTOR

Cuenca, Octubre del 2016

Nosotros, **Ivan Fernando Yuquilima Albarado**, con documento de identificación N° **0105719843** y **Christian Hernán Argudo Vásconez**, con documento de identificación N° **0301974499**, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos los autores del trabajo de grado intitulado: "**TELEMETRÍA COMO SERVICIO (TEaaS) BASADA EN EL PROTOCOLO ZIGBEE Y FOG COMPUTING PARA MONITOREO DE LA CALIDAD DEL AIRE**", mismo que ha sido desarrollado para optar por el título de Ingeniero de Sistemas, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

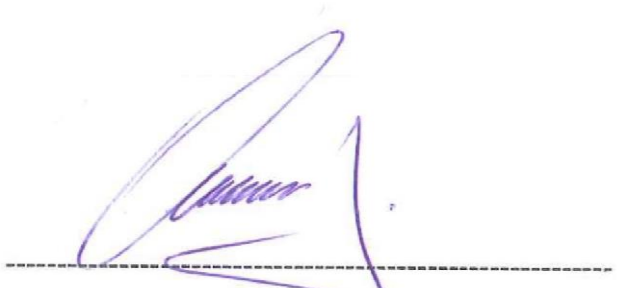
Ivan Fernando Yuquilima Albarado
0105719843

Christian Hernán Argudo Vásconez
0301974499

CERTIFICACIÓN

Cuenca, Octubre del 2016

Yo declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: "**TELEMETRÍA COMO SERVICIO (TEaaS) BASADA EN EL PROTOCOLO ZIGBEE Y FOG COMPUTING PARA MONITOREO DE LA CALIDAD DEL AIRE**", realizado por: Ivan Fernando Yuquilima Albarado y Christian Hernán Argudo Vásconez, obteniendo este Proyecto Técnico que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana para ser considerado como Trabajo de Titulación.



Pablo Leonidas Gallegos Segovia
C.I 0102593589

DECLARATORIA DE RESPONSABILIDAD

Cuenca, Octubre del 2016

Nosotros, **Ivan Fernando Yuquilima Albarado**, con documento de identificación N° **0105719843** y **Christian Hernán Argudo Vásconez**, con documento de identificación N° **0301974499**, del tema "**TELEMETRÍA COMO SERVICIO (TEaaS) BASADA EN EL PROTOCOLO ZIGBEE Y FOG COMPUTING PARA MONITOREO DE LA CALIDAD DEL AIRE**", certificamos que el total contenido de este Proyecto técnico es de nuestra exclusiva responsabilidad y autoría.

Ivan Fernando Yuquilima Albarado
0105719843

Christian Hernán Argudo Vásconez
0301974499

AGRADECIMIENTOS

Queremos hacer un extensivo agradecimiento a la Universidad Politécnica Salesiana, a los docentes de la carrera de Ingeniería de Sistemas quienes nos han guidado a lo largo de nuestra formación humana y universitaria.

A la Directora de carrera, Ing. Bertha Tacuri por el apoyo brindado en el desarrollo del presente proyecto técnico.

A los integrantes del Grupo de investigación de Cloud Computing, Smart Cities & High Performance Computing, quienes han sido un pilar fundamental para la culminación del proyecto, especialmente al Ing. Gabriel León, Ing. Jairo Sacoto e Ing. Jennifer Yépez.

A nuestro compañero Paúl Montesdeoca quien nos ayudó durante las diferentes fases del desarrollo del proyecto.

Finalmente queremos expresar nuestro más sincero agradecimiento a nuestro tutor el Ing. Pablo Gallegos por que más que un docente es un amigo.

Ivan Yuquilima – Christian Argudo

DEDICATORIA

A Dios por haberme dado la vida, salud y sabiduría necesaria para poder llegar a superar todas las metas que me he propuesto. A mis padres Daniel y Narcisa y a mi hermana Patricia que con sus consejos, valores, apoyo y amor incondicional han sido el pilar fundamental de toda mi vida y mi mayor fuente de motivación. A mi madrina Gladis a mis tíos Juan y Olga que a pesar de la distancia me han brindado todo su apoyo. A mi abuelita Rosa que hoy se encuentra en el cielo pero siempre supo aconsejarme y alentarme a seguir adelante. A todos mis amigos y compañeros con los cuales compartí gratos momentos y que constantemente mostraron su confianza en mí. A mi novia Diana por estar conmigo en los buenos y malos momentos, apoyándome de manera incondicional. A todos mis profesores que con sus valores y conocimientos me ayudaron a crecer profesionalmente, de manera muy especial al Ingeniero Pablo Gallegos. Finalmente a mis compañeros y amigos Christian Argudo y Paul Montesdeoca sin los cuales no hubiera sido posible el desarrollo del presente proyecto.

Ivan Fernando Yuquilima Albarado.

Quiero dedicar este trabajo primeramente a Dios por darme la vida, fortaleza y sabiduría para poder culminar con éxito este proyecto técnico. A mis padres Hernán y Rina junto con mis hermanos Andrea y Jonathan quienes han sido un pilar fundamental en mi vida estudiantil, con su apoyo incondicional y estar ahí cuando más los necesite. A todos mis profesores tanto de primaria, secundaria como universitarios por brindarme su conocimiento y permitirme crecer como profesional, especialmente al Ing. Pablo Gallegos tutor de este proyecto. A mis amigos y compañeros de clases quienes siempre han estado ahí para ayudarme de manera desinteresada, especialmente a mis compañeros Ivan Yuquilima y Paul Montesdeoca.

Christian Hernán Argudo Vásconez.

ÍNDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1. Resumen

En el presente trabajo realizamos un laboratorio de pruebas en el campus de la Universidad Politécnica Salesiana que permite el monitoreo de la calidad del aire, contaminación y temperatura ambiental mediante el uso de tecnologías actuales como redes de sensores, Fog Computing y Cloud Computing.

La red de sensores permite recolectar información sobre la concentración de Dióxido de carbono en partes por millón (ppm), la cantidad de partículas contaminantes en el aire (µg/m³), temperatura ambiental (°Celsius, °Fahrenheit), porcentaje de humedad (%), cantidad de luz (lux) y ruido ambiental (dB), los cuales se encuentran implementados en placas Arduino, formando una cantidad de siete nodos distribuidos en diferentes sitios del campus, a su vez, estos se comunican inalámbricamente con un nodo de Fog Computing o Computación Niebla, utilizando el protocolo ZigBee que permite el envío de datos recolectados por los sensores.

El nodo Fog Computing además de almacenar la información en una base de datos no relacional llamada MongoDB permite extender la capacidad de procesamiento al extremo de la red, permitiendo generar gráficos estadísticos en tiempo real de los últimos datos recolectados.

También se desplegó un Cloud privado basada en Openstack que permite ofrecer "Telemetría como servicio", la cual está dividida en dos partes: Una instancia de base de datos no relacional MongoDB, en la que también se integra un bróker MQTT el cual es un protocolo de mensajería encargado de recibir la información replicada desde el nodo Fog hacia el Cloud, en la segunda instancia se encuentra una plataforma web que permite presentar gráficos estadísticos, en función de los datos almacenados en la primera instancia, con el fin de que cualquier persona pueda acceder a la información de manera libre y hacer uso de estos para poder realizar análisis de factores contaminantes medioambientales.

2. Abstract

In the present work we made a test lab at the campus of the Salesian Polytechnic University that allows the monitoring of the quality of the air, pollution and environmental temperature through the use of current technologies such as sensor networks, Fog Computing and Cloud Computing.

The network sensors allows the collection of information about the concentration of carbon dioxide in parts per million (ppm), the amount of particulate pollutants in the air (µg/m³), ambient temperature (°Celsius, °Fahrenheit), humidity rate (%), amount of light (lux) and environmental noise (dB). Which are implemented in Arduino boards, forming a quantity of seven nodes distributed in different sites of the campus, these communicate wirelessly with a node of Fog Computing, using the ZigBee protocol that allows the sending of data collected by the sensors.

The Fog Computing node, in addition to storing the information in a non-relational database MongoDB, allows to extend the capacity of its processing to the edge of the network, allowing generate statistical graphs in real time of the last data collected.

Also we deployed a private cloud based on Openstack that allows to offer "Telemetry as a service", which is divided into two parts: An instance of non-relational database MongoDB, that integrates a broker MQTT which is a messaging protocol responsible for receiving the information that has been replicated from the Fog node to Cloud. The second instance is a web platform for presenting statistical graphics, depending on the data stored in the first instance, so anyone can access the information freely and use of these in order to perform an analysis of factors environmental contaminants.

3. Introducción

La contaminación del aire representa un importante riesgo medioambiental para la salud, provocando altos índices de mortalidad debido a la insalubridad del medio ambiente [1]. La Organización Mundial de la Salud (OMS) a través de la segunda edición de su informe "Ambientes saludables y prevención de enfermedades: Hacia una estimación de la carga de morbilidad atribuible al medio ambiente" ha definido un conjunto de estrategias para mejorar el medio ambiente y prevenir enfermedades, mediante el trabajo en conjunto desde el plano tecnológico, político y ambiental, como la creación de plataformas inteligentes que detecten factores contaminantes en una zona determinada mejorando la calidad de vida de las personas.

Nuestra propuesta consta del diseño e implementación de una plataforma tecnológica, que permita el monitoreo de la calidad del aire, detección de contaminación y temperatura ambiental dentro del campus de la Universidad Politécnica Salesiana sede Cuenca, en la que convergen varias tecnologías de la información como redes de sensores, Fog Computing y Cloud Computing en una arquitectura de cuatro capas que se detallan a continuación:

En la primera capa se encuentra la red de sensores que son los encargados de recolectar información sobre la concentración de Dióxido de carbono en partes por millón (ppm), la cantidad de partículas contaminantes en el aire ($\mu g/m^3$), temperatura ambiental (°Celsius, °Fahrenheit), porcentaje de humedad (%), cantidad de luz (lux) y ruido ambiental (dB). Estos sensores se encuentran implementados en siete nodos que están desplegados en las instalaciones de la Universidad Politécnica Salesiana sede Cuenca, y se comunican mediante el estándar inalámbrico ZigBee, que define un conjunto de protocolos de alto nivel de comunicación basándose en el estándar IEEE 802.15.4 [2] para redes de área personal, ideal para aplicaciones de baja tasa de envío de datos. Todos los nodos cuentan con un módulo Xbee Pro S1 que cumple con las especificaciones de ZigBee y permite la comunicación de los mismos a una distancia de un kilómetro con línea de vista. Dentro de una red ZigBee se puede diferenciar tres tipos de dispositivos como son: coordinador, router y dispositivo final, siendo el coordinador el nodo más importante de la red, ya que permite iniciar, mantener, gestionar y finalizar la comunicación con los demás nodos que pertenecen a la red.

En la segunda capa denominada Fog Computing o Computación Niebla es un modelo de computación distribuida, en la cual los datos son previamente procesados y almacenados en dispositivos que se encuentran al borde de una infraestructura de red, permitiendo a los usuarios finales el acceso rápido a la información con menor latencia, ideal para aplicaciones en tiempo real. En esta capa se encuentra el nodo

Fog que esta implementado en un Raspberry PI, el mismo es encargado del pre procesamiento y almacenamiento previo de los datos que son enviados de forma cifrada cada cinco minutos desde los sensores, y permite la visualización grafica de estos datos en tiempo real.

En la tercera capa se encuentra el Cloud Computing o Computación en la nube que permite el aprovisionamiento de servicios a través del internet. En el presente proyecto se ha desplegado una nube privada bajo una plataforma de código libre llamada Openstack, dentro de esta se generaron dos instancias. La primera instancia es una base de datos no relacional que usa MongoDB para el almacenamiento de los datos censados que son reenvidados desde el nodo Fog mediante un protocolo de mensajería llamado MQTT, mientras que en la segunda instancia se encuentra la plataforma web.

En la cuarta capa denominada presentación, se encuentra la aplicación web en la que se evidencia los resultados cuantificados y cualificados de los datos recolectados por la red de sensores y tratados por la plataforma permitiendo la visualización gráfica e interactiva de la información.

La presente memoria técnica incluye diferentes secciones que se desarrollan secuencialmente para obtener la plataforma de medición de calidad del aire. La estructura del documento se define a continuación: En el apartado cuatro se presentan la problemática del proyecto así como los antecedentes, importancia y delimitación, en el apartado cinco se presenta el objetivo general y los objetivos específicos que se han cumplido durante el desarrollo de este proyecto, mientras que en el apartado seis se realiza un estudio sobre el estado del arte de las tecnologías utilizadas como: Las redes de sensores, Cloud Computing, Fog Computing, el estándar de comunicación inalámbrica ZigBee y el protocolo MQTT para el envío de mensajes desde el nodo Fog al Cloud.

En el apartado siete se puede observar la topología implementada y la solución propuesta frente a la problemática planteada anteriormente. En el apartado ocho se detallan los dispositivos, protocolos y tecnologías utilizadas en la implementación de la plataforma como la identificación de requerimientos que muestra la ubicación y estructura de los nodos. En este apartado también se puede conocer las características principales de los sensores utilizados como el sensor de contaminación CO2, temperatura DHT11, partículas en el aire, luz y ruido.

En el apartado nueve se presenta información sobre: El diseño, desarrollo e implementación de la red de sensores, el nodo Fog que es el coordinador de la red encargado de centralizar los datos para un posterior envió al Cloud, despliegue del Cloud bajo una infraestructura de código libre Openstack, desarrollo y

funcionamiento de la plataforma web, en la cual se presentan reportes estadísticos en horas, diarias y mensuales de los datos medidos. Finalmente se presentan los resultados, conclusiones y recomendaciones obtenidas durante el desarrollo de este trabajo seguido de la bibliografía y anexos.

4. Problema

4.1. Antecedentes

Cuenca es la tercera ciudad más importante en el Ecuador, sin embargo al igual que otras ciudades metropolitanas del mundo, tiene una diversa variedad de problemas inherentes a la contaminación ambiental, contaminación visual y contaminación por ruido [3]

La Organización Mundial de la Salud (OMS) clasifica a la ciudad de Cuenca como una de las ciudades en el Ecuador que se encuentra en el límite más alto de la contaminación del aire [3], por lo tanto, es necesario tomar medidas que retarden y corrijan dichos niveles de contaminación.

Nuestra ciudad actualmente cuenta con un parque automotor de aproximadamente 89.000 vehículos matriculados [4], según datos de la Agencia Nacional de Tránsito (ANT), por lo que son una de las causas principales de la emisión de gases contaminantes que afectan al medio ambiente.

4.2. Importancia y Alcances

Generar prototipos y plataformas en las que se pueden desplegar varios escenarios y pruebas para la posterior selección de tecnologías adecuadas para mediciones de contaminación ambiental como son el ruido y CO2 en un campus universitario, pueden ser la base para implementar soluciones de mayores alcances. Finalmente la evaluación de los resultados nos dará la pauta para determinar si nuestro proyecto aporta a soluciones que aún no están disponibles en nuestro medio.

Se pretende desarrollar una plataforma alojada en el Cloud que permita consultar datos históricos de los niveles de contaminación existentes en el campus de la Universidad Politécnica Salesiana, y en base a esta información a futuro se puedan tomar las medidas correctivas, para mejorar la calidad de vida de la comunidad universitaria, además se genera un modelo de “Telemetría como Servicio” el cual puede ser tomado como referencia para investigadores en el campo de Cloud Computing y redes de sensores.

4.3. Delimitación

El presente proyecto se implementará en el Campus de la Universidad Politécnica Salesiana sede Cuenca, siendo un plan piloto al cual a futuro se pueden seguir incrementando más sensores en otros lugares. Inicialmente se beneficiará a la comunidad universitaria y a todas las personas que hagan investigación sobre plataformas WSN, Cloud Computing, M2M, Fog Computing y analítica acerca de factores ambientales como temperatura, ruido, luz y CO2.

5. Objetivos

5.1. General

Desarrollar una plataforma que combine Cloud Computing, bases de datos no relacionales y Fog Computing, la cual se integrará a una red de sensores basados en el protocolo ZigBee, creando un nuevo paradigma que se conoce como TEaaS o "Telemetría como servicio".

5.2. Específicos

- Implementar una plataforma de Cloud Computing basado en el software open source Openstack y crear instancias de aplicación web, base de datos y telemetría.

- Generar un prototipo de red de sensores basados en el protocolo ZigBee con open hardware y Smart sensors.

- Desarrollar un prototipo de nodo FOG para que cumpla la función de coordinador de la red de sensores, almacenamiento previo y procesamiento de la información.

- Desplegar una interfaz de visualización en un ambiente WEB, de la información recolectada por la red de sensores.

- Implementar la red de sensores en las instalaciones de la Universidad Politécnica Salesiana sede Cuenca.

6. Estado del Arte

6.1. Cloud Computing

El termino Cloud Computing [5] hace referencia tanto a las aplicaciones que pueden ser provistas como servicio a través de Internet, así como también a todo el hardware y software que está detrás de estos servicios y que posibilita su funcionamiento. A su vez también se puede definir al Cloud Computing [6] como un modelo el cual provee un conjunto de recursos computacionales configurables lo cuales pueden ser accedidos a través de red y de manera ubicua, estos recursos pueden ser desplegados rápidamente cuando se los requiera.

Características

Las características principales del Cloud Computing son las siguientes [6] :

Auto-Servicio Bajo demanda: Los usuarios pueden configurar recursos computacionales de acuerdo a sus necesidades de manera automática (por ejemplo la cantidad de almacenamiento, cantidad de memoria, entre otros).

Amplio acceso a la red: se puede acceder mediante mecanismos estándar de red, con lo que se permite su uso mediante plataformas heterogéneas de clientes ligeros y pesados.

Agrupación de recursos: combina los recursos computacionales del proveedor del servicio con el fin de que estos se puedan asignar y reasignar de manera dinámica a diferentes usuarios en base a su demanda.

Rápida elasticidad: alta escalabilidad debido a que los recursos pueden ser asignados y liberados rápidamente de acuerdo a las necesidades de los usuarios.

Servicio medido: se puede controlar, monitorizar y generar reportes en cuanto al uso de los recursos.

Tipos de Clouds

Públicos: se caracterizan por estar disponible para el público en general. [7] A su vez el acceso a los recursos de este tipo de Cloud se lo realiza a través de Internet, mediante aplicaciones Web o Servicios Web. [8] Cabe mencionar que sus usuarios generalmente necesitan pagar por el uso de sus servicios. [9]

Privados: se caracterizan por ser propios de una compañía u organización. [7] Al pertenecer a una organización y ser construidos para su propio uso con lo que

permiten tener un mayor control de la información, su seguridad y el nivel de la calidad de servicio. [8]

Híbridos: son un intermedio entre los Clouds privados y los públicos. Permiten que un Cloud privado pueda utilizar recursos de un Cloud público con el fin de incrementar sus capacidades. [9]

Modelos de servicio que provee el Cloud Computing

Inicialmente el Cloud Computing ofreció tres modelos principales entre ellos: "Software como Servicio" (SAAS), "Plataforma como Servicio" (PAAS) e "Infraestructura como Servicio" (IASS); sin embargo debido su éxito en el mercado, surgió la necesidad de proveer nuevos servicios, dando origen al término "Todo como Servicio" comúnmente conocido como XAAS.

- **SAAS**

Software como Servicio [10] hace referencia a aplicaciones que se encuentran alojadas en una plataforma y están disponibles para los usuarios a través de Internet. Este tipo de aplicaciones se encuentran listas para su utilización, sin embargo posiblemente sea necesario realizar ciertas configuraciones, como por ejemplo la creación de cuentas de usuarios en el servicio de correo electrónico ofrecido por Microsoft Office 365.

- **PAAS**

Plataforma como servicio [11] provee todos los recursos necesarios para permitir desarrollar, probar y poner en producción aplicaciones web, a su vez también puede proveer más servicios como colaboración en línea, almacenamiento, entre otros.

- **IASS**

Infraestructura como servicio [10] este modelo provee hardware como servicio y también el software necesario para su funcionamiento. Generalmente puede ser considerado como la evolución del hosting tradicional, esto debido a que los usuarios pueden acceder a los recursos de hardware bajo demanda mientras que al mismo tiempo pueden configurar, gestionar e instalar los servicios que crean necesarios sobre los mismos.

- **XAAS**

Todo como servicio [12] hace referencia al cualquier tipo de servicio que puede ser aprovisionado a través del Cloud, dando origen a un sin número de posibilidades.

6.2. Fog Computing

Fog Computing [13] es un modelo de computación distribuida propuesta por Cisco, para extender el concepto de Cloud Computing al borde de una infraestructura de red [14], proporcionando funcionalidades de procesamiento, almacenamiento y aprovisionamiento de datos a los usuarios finales, logrando que la información sea previamente procesada y almacenada en los dispositivos que se encuentran en el extremo de la red, en lugar de hacerlo completamente en la nube [15].

Si bien el Cloud Computing ofrece este tipo de servicios, hoy en día, estas arquitecturas no están diseñadas para soportar el volumen, variedad y velocidad de datos que el internet de las cosas (IOT) exige, según [16] se estima que en el año 2020 existirá un total de cincuenta billones de dispositivos conectados al internet, generando un total de dos exabytes de datos por día, tomando en cuenta esta información, es necesario la implementación de nuevas arquitecturas o modelos que ayuden a gestionar y administrar de manera flexible la transaccionalidad de los datos, debido a la gran cantidad de ancho de banda que se necesitará si estos datos van a ser almacenados solamente en el Cloud, por tal motivo, nace el concepto de Fog Computing que permite que cierta información pueda ser accedida por los usuarios de manera rápida debido a la capacidad de almacenamiento que cuentan los nodos Fog, sin que todo el procesamiento sea realizado totalmente en la nube, además proporciona ciertas ventajas para la implementación de aplicaciones [15] donde la velocidad y la baja latencia de acceso a los datos sean el principal factor de su funcionamiento, siendo un modelo adecuado para el despliegue de proyectos en tiempo real, recibiendo una gran cantidad de datos que pueden ser transmitidos por multiples redes de comunicación como por ejemplo ZigBee, WiFi, 2G/3G/4G, WiMax, 6Lowpan entre otras [17] logrando una homogeneidad entre los usuarios finales y los nodos Fog.

6.3. Message Queue Telemetry Transport (MQTT)

MQTT es un protocolo de mensajería (M2M) el cual se puede utilizar para enviar datos al estilo de telemetría como mensajes, a través de redes de alta latencia o con restricciones [18]. Está diseñado para ser una mensajería Publish-Subscribe muy ligera para conexiones donde es importante no consumir gran cantidad de ancho de banda, con el fin de optimizar el consumo energético de los dispositivos. Dentro de este protocolo se pueden distinguir tres tipos de dispositivos: Broker, Publicador y Suscriptor.

6.4. Estándar inalámbrico ZigBee

6.4.1. ZigBee Alliance

La Alianza ZigBee [2] es una asociación sin fines de lucro, formada por más de 400 compañías, que se dedican a la creación de normas para la interconexión de dispositivos a nivel mundial, desarrollando soluciones inalámbricas rentables y flexibles.

Figura 1: Logo de ZigBee Alliance [2]

6.4.2. Estándar ZigBee

ZigBee define un conjunto de protocolos de alto nivel de comunicación para la conexión de redes de área personal inalámbrica o WPAN [2], en donde su objetivo es el desarrollo de aplicaciones que están diseñadas para comunicaciones con baja tasa de envío de datos y se requiera un limitado consumo energético.

Este tipo de comunicación se basa en el estándar IEEE 802.15.4 en el cual se definen las capas PHY (Physical Layer) y MAC (Medium Access Control), sin embargo las capas NWK (Network Layer) y APL (Application Layer) han sido establecidas por la Alianza ZigBee [2], por lo que ZigBee se diferencia de los demás estándares que pertenecen a comunicaciones inalámbricas WPAN, a continuación se puede observar las caracteristicas principales de este estándar.

Tabla 1: Caracteristicas Principales del Estándar ZigBee.

CARACTERISTICA	RANGO
Bandas de Frecuencia.	868 – 868.8 MHz (Europa) 902 – 928 MHz (Norte América) 2400 – 2483.5 MHz (Todo el Mundo)
Alcance	10 – 75 m
Retardo (Latencia)	Menor a 15 ms
Número de canales	868 – 868.8 / 902 – 928 MHz: 11 canales 2400 – 2483.5 MHz: 16 canales
Rango de transmisión de datos	868 – 868.8 MHz: 20 kbps 902 – 928 MHz: 40 kbps

	2400 – 2483.5 MHz: 250 kbps
Direccionamiento	64 bits Adicional: Corto de 16 bits
Nodos por red	Hasta 65536 Hasta 255 por subred
Canal de acceso	CSMA-CA CSMA-CA Rasurado
Temperatura	Rango de temperatura industrial: -40 a 85 °C

En una red inalámbrica que utiliza el estándar ZigBee para la comunicación, se pueden encontrar tres tipos de dispositivos:

- **COORDINADOR:**

Este dispositivo es el más importante dentro de una red ZigBee, ya que permite la administración, gestión y monitoreo de la red así como la asignación del direccionamiento lógico a sus nodos hijos. En aplicaciones reales este tipo de dispositivo requiere de memoria y capacidad de computación para realizar las tareas mencionadas anteriormente, además puede cumplir funciones de enrutamiento con el fin de interconectarse con redes externas.

- **ROUTER:**

Dentro de una red ZigBee el dispositivo configurado en modo router es encargado de comunicar los dispositivos finales (End Devices) con el coordinador o con otros dispositivos que no estén dentro del mismo segmento de red, por lo que requiere tener su tabla de enrutamiento actualizada.

- **END DEVICE (DISPOSITIVO FINAL):**

Es el dispositivo más básico dentro de una red ZigBee ya que solo se encarga del envió de datos a su nodo padre, ya sea el coordinador o el router, la mayoría del tiempo puede estar en modo dormido (sleep) aumentando de esa forma la vida útil de las baterías de los equipos. Tiene requerimientos mínimos de memoria y es por tanto significativamente más económico.

7. Topología y Solución Propuesta

Este proyecto cuenta con 7 nodos, cada nodo está compuesto por un Arduino, un módulo Zigbee y diferentes tipos de sensores de acuerdo al nodo, revisar tabla 2. Estos sensores tienen el objetivo de determinar la calidad del aire dentro de las instalaciones de la Universidad Politécnica Salesiana. Actualmente contamos con

sensores de calidad del aire, temperatura, humedad, luxómetro, ruido y cantidad de partículas en el aire. Los datos recolectados de los diferentes tipos de sensores son almacenados en la nube, permitiendo realizar gráficos estadísticos de sus valores.

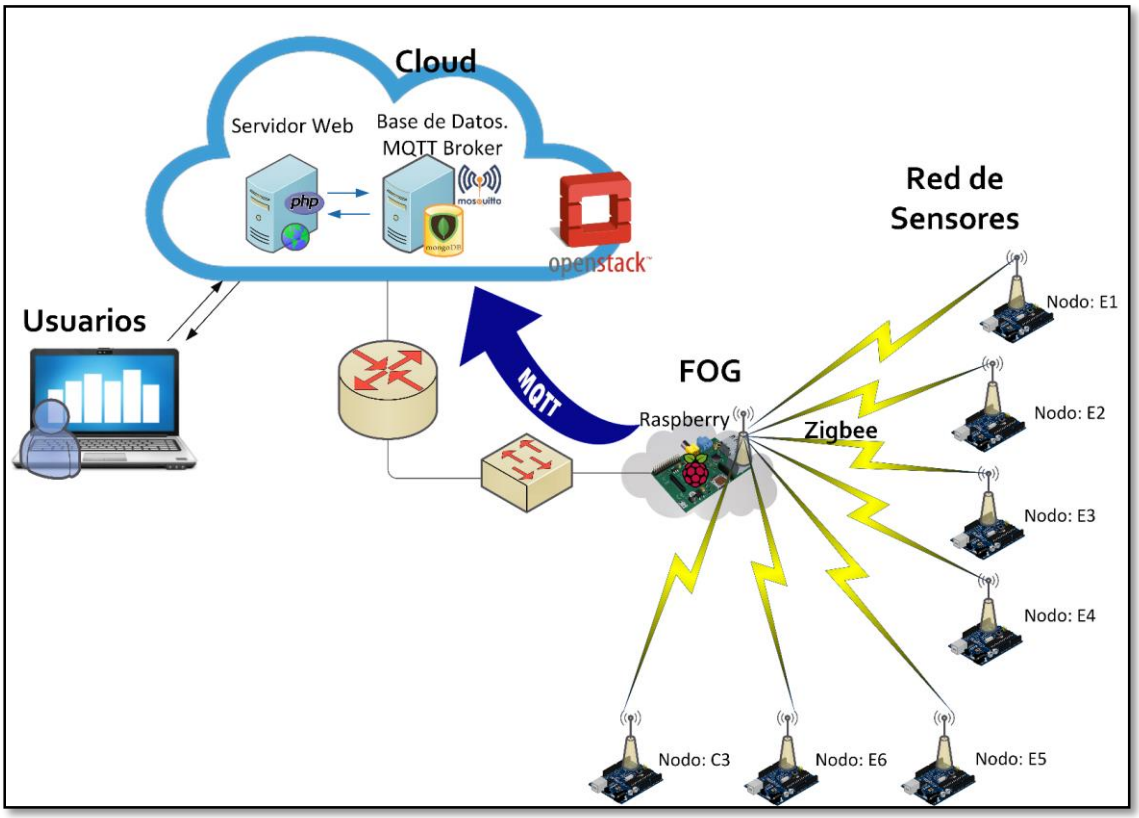


Figura 2: Arquitectura Propuesta

8. Especificaciones Técnicas

En este apartado se detallan los dispositivos, protocolos y tecnologías utilizadas en la implementación de la plataforma, en donde se combina el Cloud Computing y Fog Computing integrándose a una red de sensores que se comunican entre sí mediante el protocolo ZigBee.

8.1. Red de sensores

8.1.1. Identificación De requerimientos

En la siguiente tabla se puede observar la distribución geográfica de los nodos, y los diferentes tipos de sensores que poseen, para mayor información revisar el anexo 14.4 en el cual se encuentra el despliegue de los nodos en un mapa.

Tabla 2: Distribución de los nodos de la plataforma

NODO	UBICACIÓN	DESCRIPCION
COORDINADOR 2 (FOG)	Segundo piso del edificio Cornelio Merchán. Entrada a los laboratorios de eléctrica y electrónica	-Encargado de la recolección de los datos, para enviarlos al Cloud.
COORDINADOR 3	Segundo piso del Edificio Cornelio Merchán en parte superior del antiguo laboratorio de Idiomas	Posee sensores de: -CO2 -Temperatura -Luz -Ruido
E1	Entrada secundaria a la Universidad Politécnica Salesiana, junto al cajero del Banco del Pacifico.	Posee sensores de: -Calidad del Aire -CO2 -Ruido
E2	Entrada principal de la Universidad.	Posee sensores de: -Calidad del Aire -CO2 -Ruido
E3	Entrada principal de la Iglesia UPS.	Posee sensores de: -CO2 -Temperatura -Ruido
E4	Segundo piso del Edificio Mario Rizzini frente al departamento de Sistemas	Posee sensores de: -CO2 -Temperatura -Luz -Ruido
E5	Segundo piso del Edificio Mario Rizzini junto al Edifico Guillermo Mensi (Coworking)	Posee sensores de: -CO2 -Temperatura -Luz -Ruido

E6	Segundo piso del Edificio Cornelio Merchán junto a Ciencias Administrativas y Economía	Posee sensores de: -CO2 -Temperatura -Ruido

8.1.2. Especificación de los sensores

En esta sección se describen los sensores utilizados en la infraestructura de red, así como sus características técnicas principales que provee su fabricante.

8.1.2.1. Sensor MQ135 (CO2)

El sensor MQ-135 se utiliza para la detección de concentración de gas en el medio ambiente, ideal para aplicaciones de monitoreo y calidad del aire. Además permite detectar niveles de dióxido de carbono (CO2), alcohol y factores contaminantes producidos por el humo del tabaco. Su funcionalidad según el fabricante [19] radica en la variación de una resistencia interna cuando se expone a determinados gases, ya que posee un calentador interno que se encarga de aumentar la temperatura del circuito, para que el sensor tenga la capacidad de reaccionar frente a la exposición de gases, provocando un cambio en su resistencia.

En la implementación de los nodos se utilizó este sensor para detectar la concentración de CO2 en el medio ambiente en partes por millón (ppm), ya que al tener salidas analógicas y digitales facilita su lectura a través de una placa Arduino. Este sensor requiere de una alimentación de 5V, una conexión a tierra (Ground), y las conexiones de lectura a los pines analógicos o digitales dependiendo del tipo de aplicación que se desea realizar. Revisar la imagen de este sensor en la figura 3.

Figura 3: Sensor de contaminación MQ-135

8.1.2.2. Sensor de Temperatura DHT11

El sensor DHT11 permite medir la temperatura y humedad del medio ambiente, utilizando un sensor capacitivo de humedad y un termistor para medir el aire que circula según datos técnicos su fabricante [20]. Además se caracteriza por tener una sola salida digital que es calibrada por su fabricante por lo que se garantiza su fiabilidad al momento de censar medidas de temperatura del medio ambiente. Puede medir la humedad entre el rango 20% – 95% y la temperatura entre el rango 0ºC – 50ºC, por lo que se ha visto conveniente la utilización de este sensor para la implementación de los nodos en la infraestructura de sensores, obteniendo medidas en Centígrados, Fahrenheit y porcentaje de humedad.

Este sensor requiere de una alimentación de 5V o 3,3V, una conexión a tierra (Ground), y la conexión de lectura al pin digital. Revisar la imagen de este sensor en la figura 4.

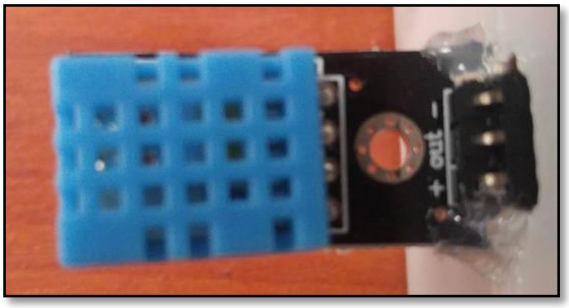


Figura 4: Sensor de temperatura DHT11

8.1.2.3. Sensor de Partículas

El sensor de partículas en el aire o conocido en inglés como (dust density) permite la cuantificación de partículas contaminantes en el ambiente de hasta 0.5 micras, en la actualidad es utilizado en proyectos para el monitoreo de la calidad del aire o en sistemas de aire acondicionado ideal para empresas u organizaciones. Este sensor según su fabricante [21] está compuesto internamente por un diodo emisor de infrarrojos (IRED) y por un fototransistor, que se encuentran ubicados diagonalmente de tal forma que pueden detectar las reflexiones de luz sobre las partículas de polvo en el aire. Cuando pasan las partículas por el sensor, la respuesta del fototransistor genera una medida de la cantidad de partículas en la muestra. Una de las caracteristicas principales de este sensor es su capacidad para generar una salida equilibrada a la cantidad de partículas en µg/m³, esto ayuda a simplificar el diseño de sistemas de control y monitoreo de la calidad del aire. En la figura 5 se puede observar de forma gráfica este sensor.

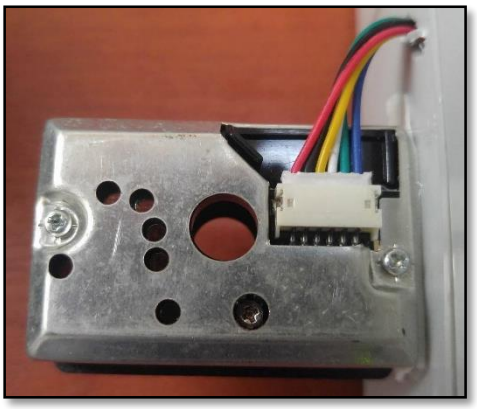


Figura 5: Sensor de partículas del aire

8.1.2.4. Sensor de Luz – LDR

Para medir la cantidad de luz en el ambiente se utilizó un LDR o conocido como fotorresistencia, donde su funcionamiento radica en la disminución de una resistencia interna con el crecimiento de la intensidad de luz reflejada en el sensor.

Este sensor requiere de una alimentación de 5V o 3.3V, una conexión a tierra (Ground), y la conexión de lectura al pin analógico junto con una resistencia de 5KOhm, obteniendo valores en lux. Revisar la imagen de la fotorresistencia en la figura 6.

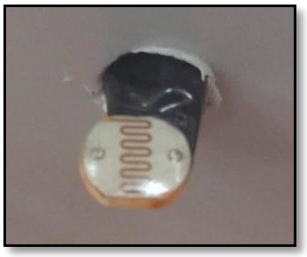


Figura 6: Fotorresistencia

8.1.2.5. Sensor De Ruido

En el presente proyecto técnico para medir la cantidad de ruido ambiental en decibelios se diseñó e implementó un circuito electrónico compuesto por un micrófono electrect, un amplificador LM358, capacitores de 1µf - 100µf y resistencias de 10KΩ - 1 KΩ.

Al no contar con un sensor de ruido analógico, se procedió a la implementación de una solución práctica, con la ayuda de un micrófono el cual es encargado de captar las señales de ruido ambiental posteriormente estas señales son amplificadas con la ayuda de un circuito integrado LM358 conjuntamente con capacitores y resistencias,

permitiendo así tener una lectura en un pin analógico. Revisar la imagen del micrófono Electrect en la figura 7.

Figura 7: Micrófono Electrect

8.1.3. Comunicación Inalámbrica

En esta sección se presentan los componentes técnicos para la comunicación de los End Device con el nodo Fog, que forman parte del diseño de la solución, con la correspondiente descripción de sus caracteristicas principales.

8.1.3.1. Módulos Xbee Pro S1

En el presente proyecto técnico se utilizan los módulos Xbee Pro S1 [22] que son soluciones integradas que permiten la interconexión y comunicación de dispositivos en redes punto a punto o redes punto a multipunto, utilizando el estándar IEEE 802.15.4, estos módulos son capaces de transmitir datos hasta una milla (1,6 km) con línea de vista, según las especificaciones técnicas de su fabricante. Revisar la tabla 3.

Tabla 3: Especificaciones técnicas del módulo Xbee Pro S1 [23]

CARACTERISTICA	RANGO
Bandas de Frecuencia.	2400 – 2483.5 MHz
Alcance	Indoor: 90 metros Outdoor: 1600m
Retardo (Latencia)	Menor a 15 ms
Rango de transmisión de datos	2,4 GHz
Direccionamiento	64 bits Adicional: Corto de 16 bits
Velocidad	250 Kbps
Encriptación	128-bit AES
Temperatura	Rango de temperatura industrial: -40 a 85 °C

Potencia de transmisión	60 mW (+18 dBm)*
Sensibilidad de Recepción (1%)	-100 dBm

El módulo Xbee Pro S1 requiere una alimentación de 3,3 V, una conexión a tierra (Ground), y las conexiones para la transmisión de datos (TX y RX) que pueden ser conectados a un microcontrolador o una placa Arduino.

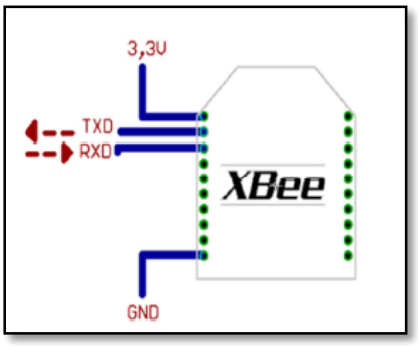


Figura 8: Conexión Requerida para un módulo Xbee.

8.1.3.2. SOFTWARE XCTU

XCTU [22] es una aplicación multiplataforma de acceso libre, diseñado para permitir a los desarrolladores interactuar con los módulos de radio frecuencia Digi, además posee herramientas que hacen que sea fácil de instalar, configurar y probar los módulos RF Xbee, asimismo se puede utilizar este software para obtener de forma gráfica la topología de red, de los dispositivos ya configurados.

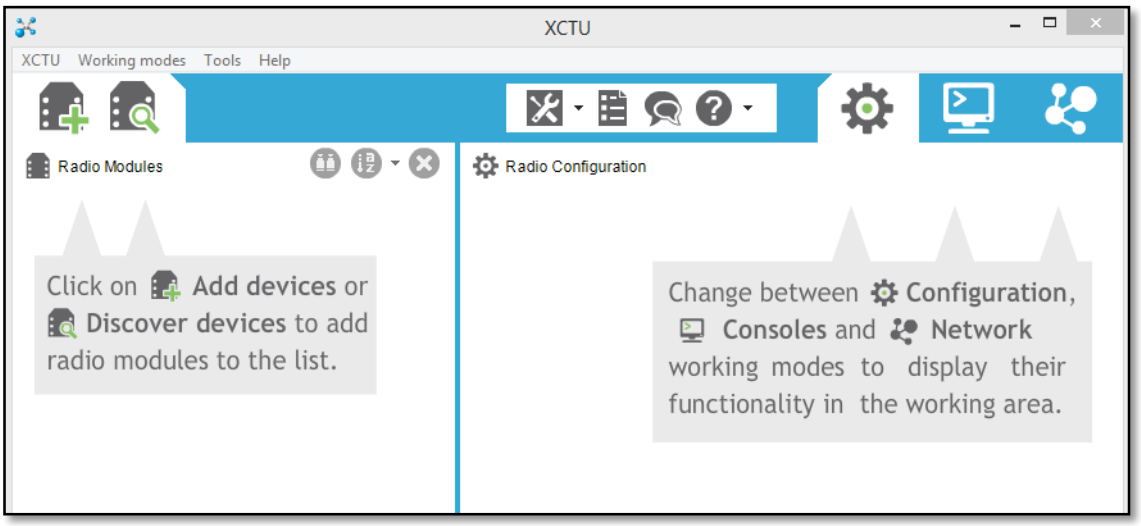


Figura 9: Pantalla principal del software XCTU

9. Marco Metodológico

En esta sección se especifica cómo se llevaron a cabo las tareas o etapas (figura 10) para el desarrollo de la plataforma de sensores.

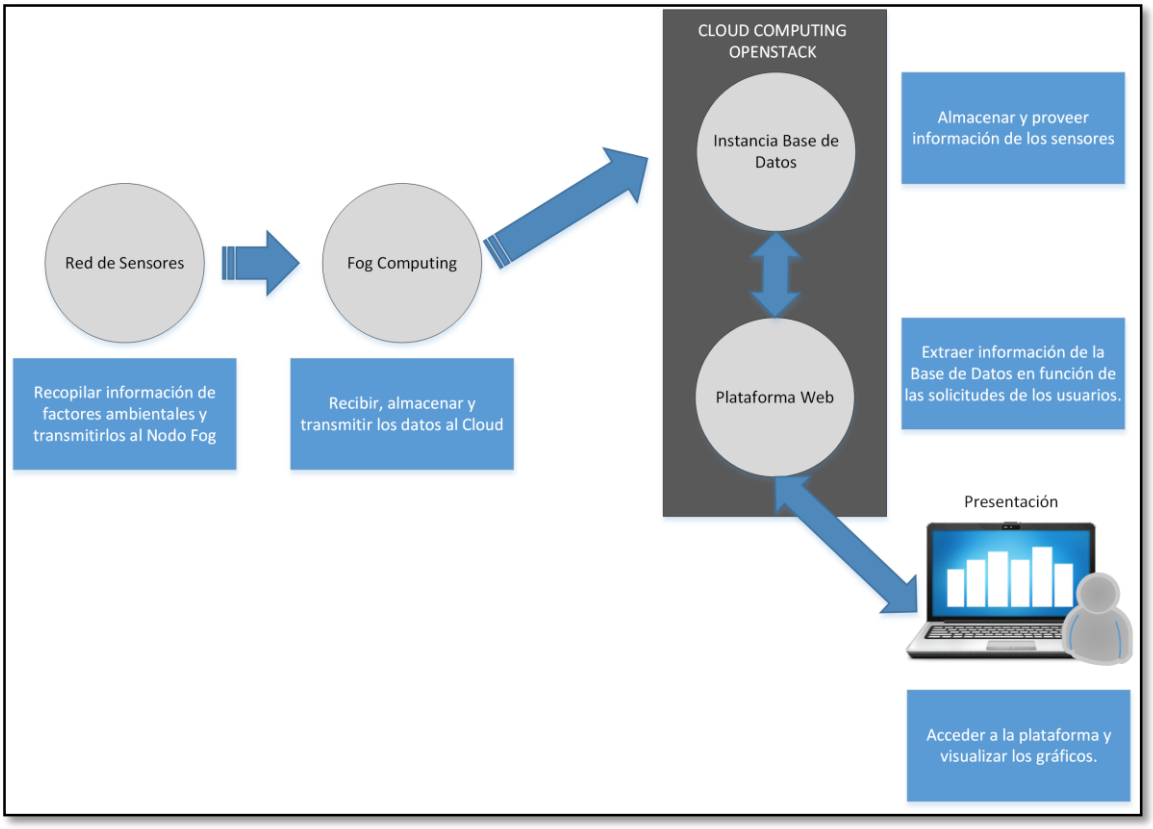


Figura 10: Etapas del proyecto técnico

9.1. Red De Sensores

9.1.1. Diseño De la Infraestructura

Este proyecto cuenta con 7 nodos, que están compuestos por un Arduino, un módulo Xbee Pro S1 y diferentes tipos de sensores de acuerdo al tipo de nodo. Estos sensores tienen el objetivo de determinar la calidad del aire dentro de las instalaciones de la Universidad Politécnica Salesiana sede Cuenca. Actualmente contamos con sensores de temperatura, humedad, ruido, luz, Dióxido de Carbono (CO2) y cantidad de partículas en el aire. Estos sensores son los encargados de censar las medidas ambientales para ser enviados bajo el protocolo de comunicación inalámbrica ZigBee al nodo coordinador (Fog) cada cinco minutos.

9.1.1.1. CONFIGURACIÓN DE LOS MÓDULOS XBEE

La comunicación inalámbrica entre los sensores se basa en el protocolo ZigBee por lo que es necesario configurar los módulos Xbee Pro S1, para ello es necesario conectar la placa Arduino con el módulo Xbee como se muestra en la figura 11, además es necesario la instalación del software XCTU en el computador, el mismo permitirá configurar los parámetros de la red, revisar la tabla 4, también mediante la ayuda de este software se puede configurar para que la información se transmita de forma cifrada y además establecer una contraseña común para todos los dispositivos que pertenecen a la red, para así evitar ataques informáticos.

Tabla 4: Arquitectura de red de los módulos Xbee Pro S1

NODO	PAN ID	CANAL	DIRECCION DE 16 BITS (MY)	DIRECCION DE DESTINO 16 BITS (DL)
COORDINADOR 2 (FOG)	1701	C	CCC2	-
COORDINADOR 3	1701	C	CCC3	CCC2
E1	1701	C	AAAA	CCC2
E2	1701	C	BBBB	CCC2
E3	1701	C	DDDD	CCC2
E4	1701	C	AAA1	CCC2
E5	1701	C	BBB1	CCC2
E6	1701	C	DDD1	CCC2

- **PAN ID:** Este parámetro identifica la red del área personal (PAN), cabe mencionar que todos los Xbee que pertenecen a la misma red, deberán especificar en su configuración el mismo valor de PAN ID.

- **MY:** Es la dirección corta de 16 bits que identifica a cada módulo, el mismo debe ser único dentro de un segmento de red.

- **DL:** Es la dirección de destino de 16 bits, con la que el módulo pretende comunicarse.

- **BD:** Tasa de baudios, es la velocidad a la que se comunicarán los módulos, en el presente proyecto se utilizó el valor de 9600 baudios, que es el valor por defecto.

Una vez instalado este software, se procede al reconocimiento de los módulos Xbee y la configuración de los mismos, cabe mencionar que debería estar extraído el chip **ATMEGA328P-PU** si estamos trabajando con Arduino UNO, en el caso de trabajar con un Arduino MEGA no existirá problemas de comunicación.

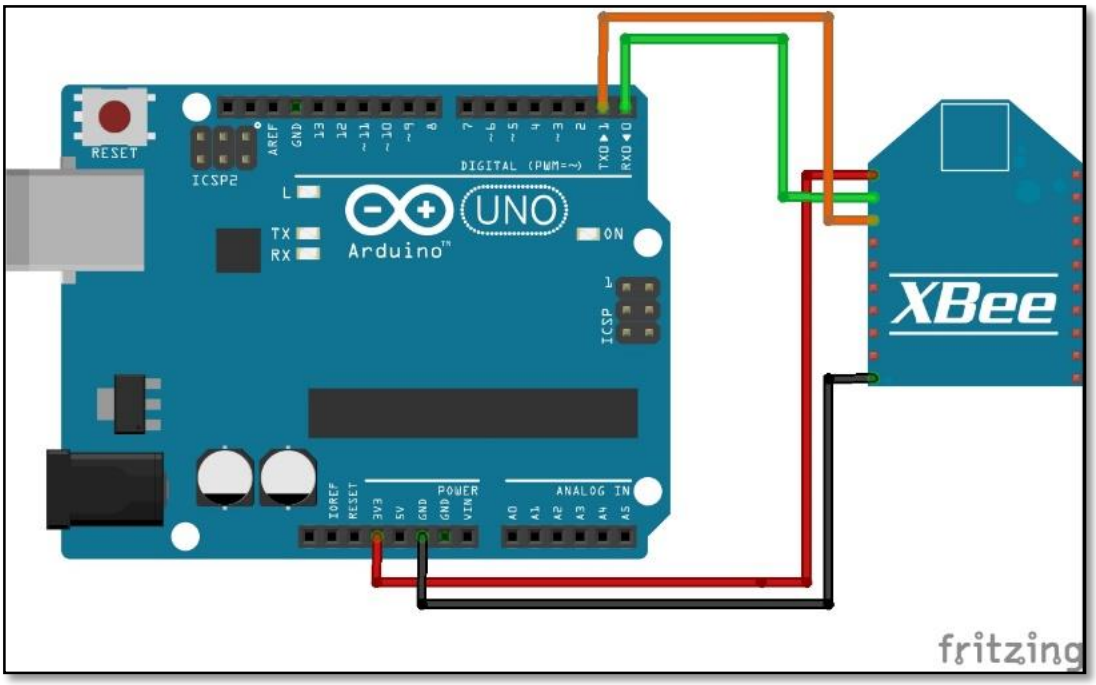


Figura 11: Conexión básica para la configuración del módulo Xbee Pro S1

9.1.1.2. Configuración de los Módulos Xbee en modo Coordinador

Al ser el coordinador el nodo más importante dentro de una arquitectura de red bajo el protocolo ZigBee, debido a su capacidad de iniciar, mantener, gestionar y finalizar la comunicación con los demas nodos, se ha visto conveniente la implementación de dos coordinadores, revisar la tabla 4, dentro de la estructura de la red de sensores.

Para configurar los módulos Xbee Pro S1 en modo Coordinador es necesario utilizar el software XCTU, el cual permitirá el reconocimiento del módulo Xbee para proceder a especificar los requerimientos planteados como se puede observar en la figura 12.

Cabe recalcar que siendo el coordinador 2 el nodo Fog, se debe especificar en el campo Destination Address Low **DL** el valor de cero, ya que este es el encargado de centralizar los datos provenientes de los demás nodos que pertenecen a la red.

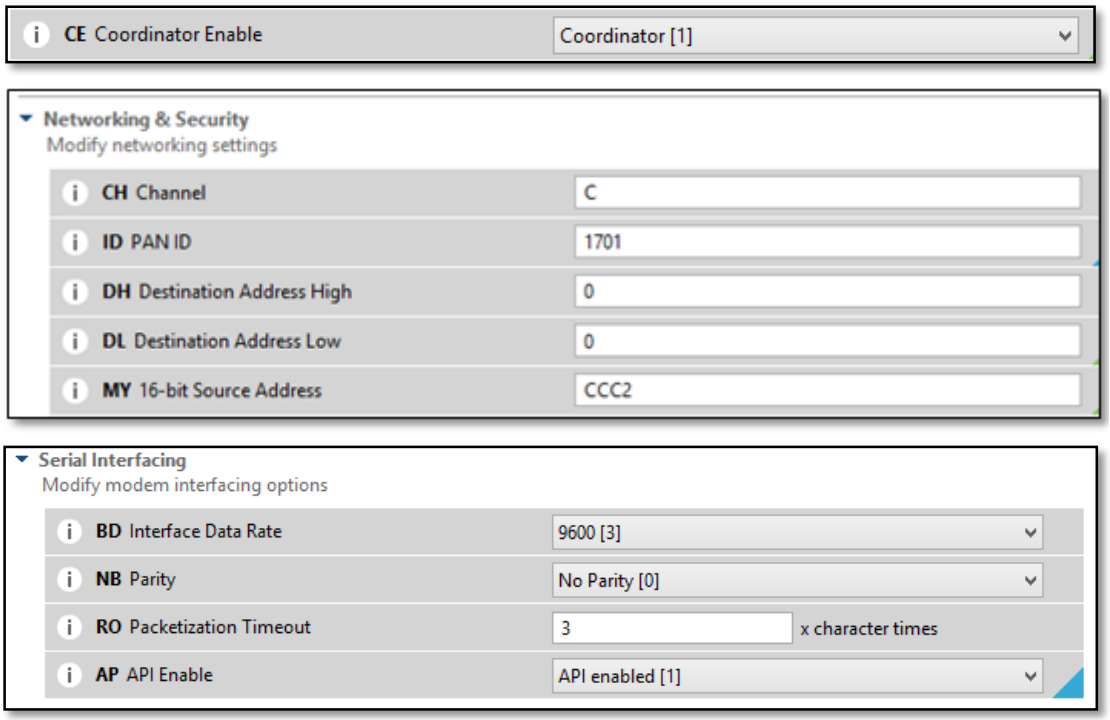


Figura 12: Configuración del Módulo Xbee Pro S1, en modo Coordinador

9.1.1.3. Configuración de los Módulos Xbee en modo End Device

Además del nodo coordinador es necesario la configuración de los dispositivos finales o conocidos como End Devices, en la plataforma implementada se cuenta con seis nodos configurados de esta forma, los mismos pueden enviar datos censados al coordinador 2 (Fog) utilizando su dirección corta de 16 bits como destino **DL**.

De igual manera que la configuración en modo coordinador, para establecer los módulos Xbee Pro S1 en modo End Device, es necesario la utilización del software XCTU, sin embargo la diferencia entre las dos configuraciones varía en el campo Coordinator Enable **CE** y en el Destination Address Low **DL**, revisar la figura 13.

La configuración de los End Devices es la misma para todos los seis nodos, excepto en el parámetro 16 - bit Source Address **MY** ya que se debe colocar la dirección asignada en la especificación de requerimientos, revisar la tabla 4, este apartado es muy importante, ya que si existen dos nodos con la misma dirección lógica se provocará un bucle al momento de la trasmisión y recepción de los datos.

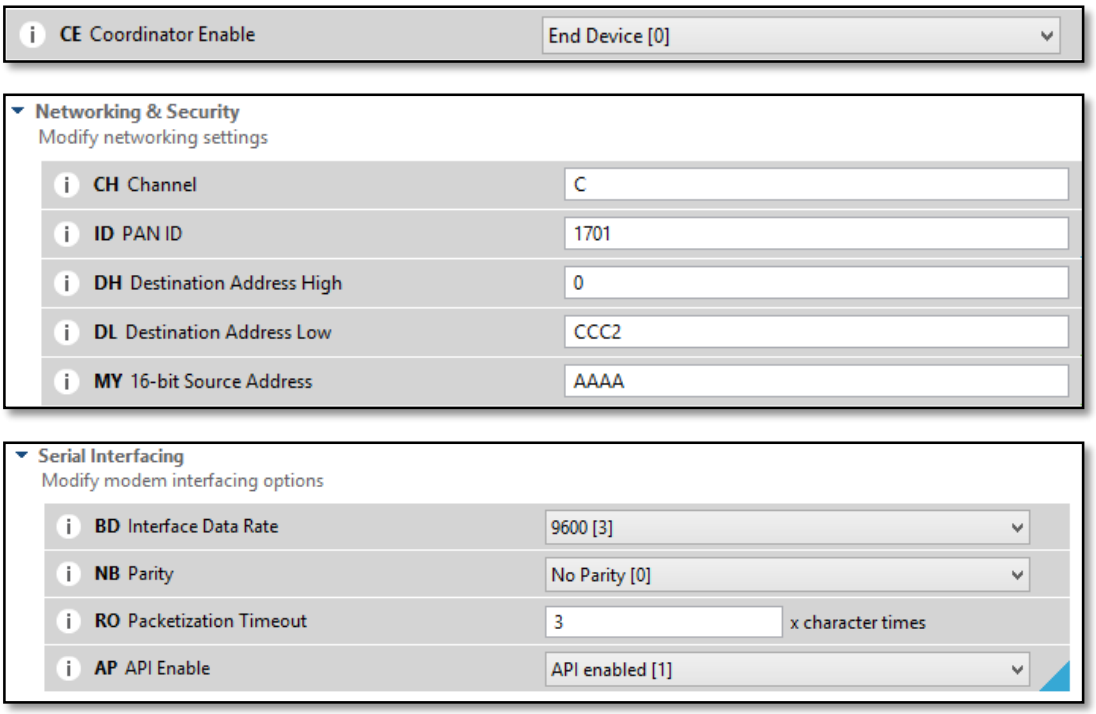


Figura 13: Configuración del Módulo Xbee Pro S1, en modo End Device 1

9.1.1.4. Diseño e Implementación de los nodos

En el presente proyecto técnico se implementaron siete nodos (End Devices) que forman la red de sensores, los mismos tienen la capacidad de censar o medir factores ambientales como temperatura, luz, ruido, Dióxido de Carbono (CO2), y la cantidad de partículas en el aire generados por el humo del cigarrillo u otro tipo de factores contaminantes, revisar tabla 2, para enviarlos cada **cinco minutos** al nodo coordinador (Fog) ya que este se encarga de centralizar la información, para un posterior envió al Cloud que se encuentra en una infraestructura de código libre Openstack. Estos medidores se encuentran distribuidos en el campus de la Universidad Politécnica Salesiana sede Cuenca, para mayor información de la ubicación revisar la tabla 2.

Cada nodo de la red cuenta con una placa Arduino, un Led, módulo Xbee Pro S1, sensores de acuerdo al tipo de nodo y las conexiones internas necesarias de los mismos en el protoboard. Un factor importante dentro de esta infraestructura de sensores es la parte energética ya que cada nodo cuenta con una conexión eléctrica a un toma corriente (110V) y una conexión a un cargador solar (5V) para tener un respaldo energético en el caso de que se fuera la luz, con el fin de evitar la pérdida de información que son de vital importancia al momento del monitoreo de la calidad del aire, cabe mencionar que el cargador solar permite el funcionamiento de los nodos durante 11 horas seguidas, tiempo necesario para el restablecimiento de la luz,

en el caso de no ser así, hay que tener en cuenta que la Universidad Politécnica Salesiana sede Cuenca posee un generador de luz.

9.1.1.4.1. Nodos E1 – E2

Los Nodos E1 y E2 están formados por sensores de ruido, contaminación y partículas en el aire, por lo que se pueden obtener medidas sobre el ruido (dB), dióxido de carbono (ppm) y la cantidad de partículas en el aire generados por factores contaminantes (mg/m³), además de una placa Arduino, un módulo Xbee configurado en modo End Device y un Led que va a parpadear al momento del envío de los datos al nodo coordinador (Fog). A continuación se puede observar el esquema de conexión de estos nodos.

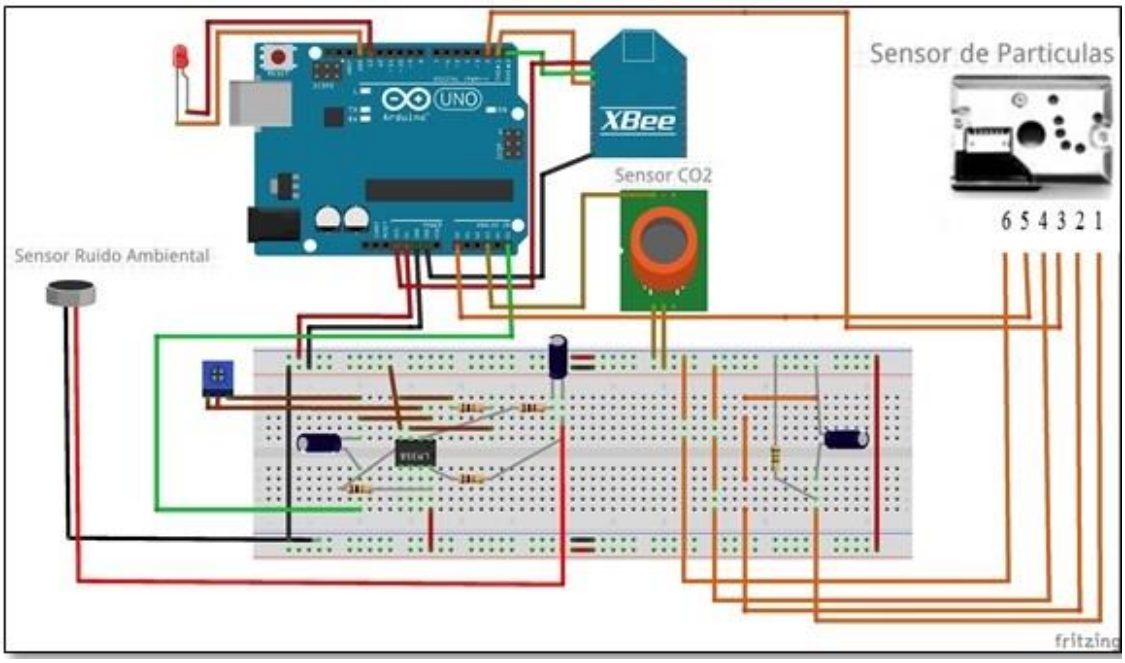


Figura 14: Esquema de conexión Nodos E1 – E2

9.1.1.4.2. Nodos E3 – E6

Los Nodos E3 y E6 están formados por sensores de contaminación, ruido y temperatura, por lo que se pueden obtener medidas sobre dióxido de carbono (ppm), ruido (dB) y temperatura en Celsius (°C), Fahrenheit (°F), Punto de rocío (°C), y porcentaje de humedad (%) , además de una placa Arduino, un módulo Xbee configurado en modo End Device y un Led que va a parpadear al momento del envío de los datos al nodo coordinador (Fog). A continuación se puede observar el diagrama de conexión de estos nodos.

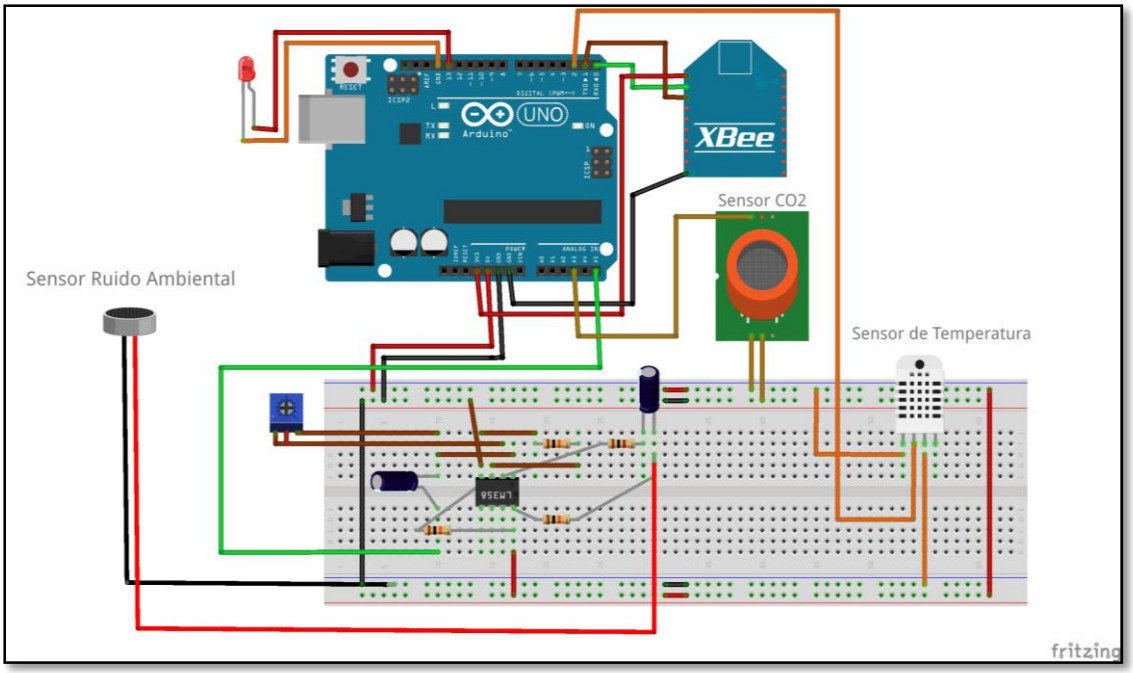


Figura 15: Esquema de conexión Nodos E3 – E6

9.1.1.4.3. Nodos E4 – E5 – C3

Los Nodos E4, E5 y C3 están formados por sensores de contaminación, ruido, temperatura y luz, por lo que se pueden obtener medidas sobre dióxido de carbono (ppm), iluminancia (lx), ruido (dB) y temperatura en Celsius (°C), Fahrenheit (°F), Punto de rocío (°C), y porcentaje de humedad (%) , además de una placa Arduino, un módulo Xbee configurado en modo End Device y un Led que va a parpadear al momento del envío de los datos al nodo coordinador (Fog). A continuación se puede observar el diagrama de conexión de estos nodos.

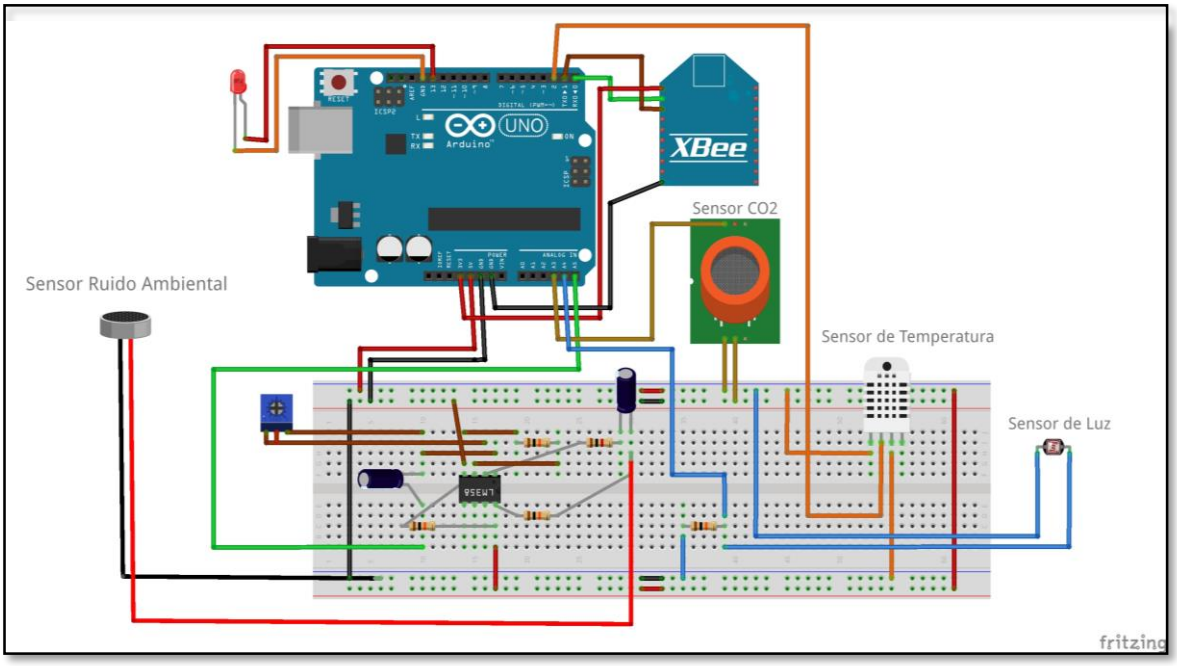


Figura 16: Esquema de conexión Nodos E4 – E5 – C3

9.1.2. Recolección y Envío de datos (sensores)

Para programar el microcontrolador Arduino se utilizó Arduino Software IDE que permite la conexión del computador con la placa para compilar programas y comunicarse con ella, mediante el uso de puertos USB. Los programas llamados comúnmente sketches cuentan con dos funciones principales:

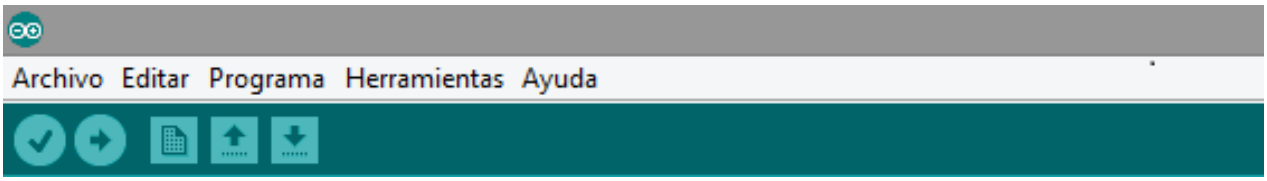
- ***Setup( ) :*** En esta función se puede especificar tareas de configuración, como la definición de pines o la inicialización de librerías. Cabe recalcar que esta función se ejecuta solamente una vez cuando el programa se ejecuta.

- ***Loop( ) :*** Como su nombre lo indica, esta función se ejecuta de forma cíclica, por lo que es un buen lugar para definir acciones que se repitan varias veces.

Al momento de desarrollar los programas de los diferentes tipos de nodos para la recolección y envío de datos se tomaron en cuenta algunas consideraciones:

- Comunicación inalámbrica mediante el protocolo ZigBee.
- Esperar diez segundos para la formación de la red, en el cual los nodos End Devices se comunicarán con el nodo Coordinador (Fog).
- El tiempo de medición y envío de datos se realizará cada cinco minutos.
- Debe existir un Led que parpadee cuando se envíen los datos.
- El mensaje que se envía desde los dispositivos finales (End Devices) hacia el coordinador debe incluir: Dispositivo, sensor, medida, valor.
- La lectura de los pines de los diferentes sensores:
    - **Sensor de partículas:** A0
    - **Sensor de luz:** A4
    - **Sensor de temperatura:** P2 (DIGITAL)
    - **Sensor de contaminación:** A3
    - **Sensor de ruido:** A5

A continuación se puede observar la estructura de los programas desarrollados para el envío de datos censados al coordinador. Básicamente la estructura de los programas varía según el tipo de sensores correspondientes a cada nodo, sin embargo al final de la figura 17 existe un método llamado xbeeTX que es común para todos los nodos ya que permite el envío de los mensajes utilizando la dirección de 16 bits del coordinador2 como destino. Cabe recalcar que la comunicación se realiza bajo el protocolo ZigBee.




```
// Importamos librerías y variables para transmisión ZigBee.
  #include <XBee.h>
   XBee = XBee();
   Tx16Request tx;
   TxStatusResponse txStatus = TxStatusResponse();
```

```
// Importamos librerías y variables para los sensores
```

```
// Definición de variables de identificación de los nodos
```

```
void setup ()
```

```
// Inicializamos variables.
```

```
// Definimos el PIN de lectura de los sensores.
```

```
void loop()
```

```
// Métodos: leerParticulas, leerCO2, leerRuido, leerSensorDHT11, leerSensorLuz
```

```
/**
   Metodo utilizado para enviar los datos de los
   sensores a travez de Zigbee
*/
void xbeeTX( String sensor, String m, String val) {
  //Serial.print("Enviando: ");
  String datos = "'disp': '" + dispositivo + "', 'sen': '" + sensor + "' , 'm': '" + m + "' , 'val': " + val;
  //Serial.println(datos);
  char output[128];
  datos.toCharArray(output, 128);
  char Buffer[128];
  strcpy(Buffer, output);
  tx = Tx16Request(0xCCC2, (uint8_t *)Buffer, strlen(Buffer)); //Indicar la direccion corta del xbee receptor.
  xbee.send(tx);
}
```

Figura 17: Estructura del Programa en Arduino IDE

9.2. Fog Computing

9.2.1. Diseño Del Nodo Fog

El coordinador2 llamado nodo Fog dentro de la infraestructura de sensores, se encuentra implementado en una plataforma de hardware libre llamada Raspberry PI, que permite el procesamiento, almacenamiento y aprovisionamiento de datos. Además cuanta con un módulo Xbee Pro S1 configurado en modo coordinador, el cual cumple la función de centralizar la información que los dispositivos finales (End Devices) han enviado, mediante la ejecución de un script desarrollado en Python, que se encuentra en segundo plano (background), y se inicia automáticamente al encenderse el Raspberry. De igual manera este nodo cuenta con una conexión a un cargador solar y a una toma corriente, para evitar pérdidas de información.

Al encontrarse el coordinador al extremo o borde de la red, la característica principal de este nodo Fog es proporcionar a los usuarios finales acceso a los datos de manera ágil sin tanta latencia en comparación con el Cloud ya que se debería pasar por mas redes aumentando así la latencia durante el acceso a los datos. Tomando en cuenta esta consideración se ha desplegado una aplicación web en el Raspberry que permite visualizar en minutos de forma gráfica las estadísticas generadas por los sensores.

A continuación se puede observar el esquema de conexión del Raspberry PI con el módulo Xbee Pro S1.

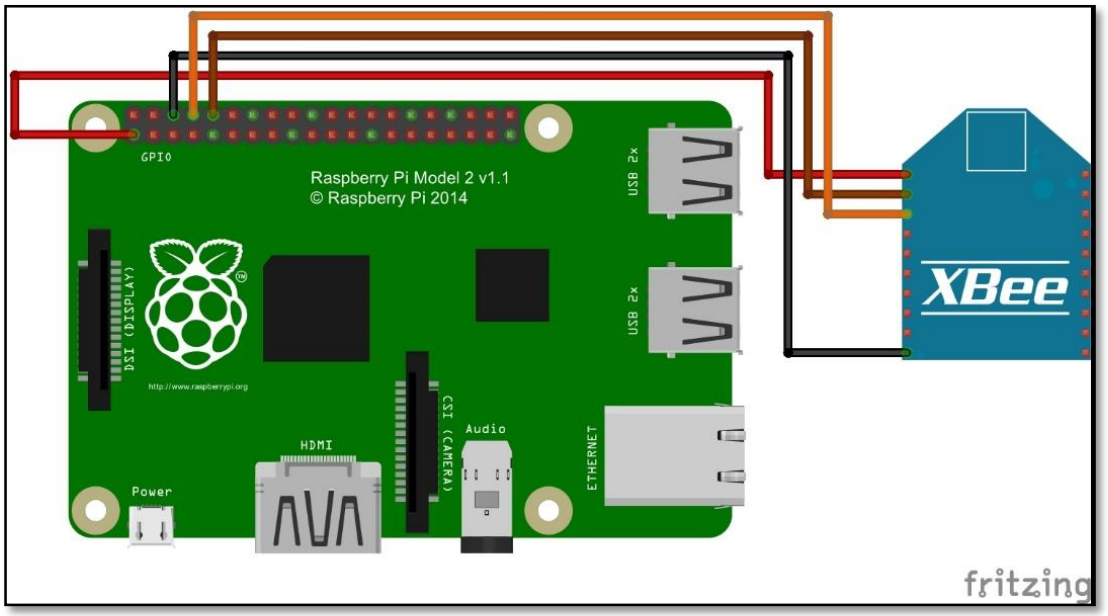


Figura 18: Esquema de conexión Raspberry Pi con el Módulo Xbee S1

9.2.2. Recolección De Datos

Así como se desarrollaron programas para él envío de datos utilizando la placa Arduino, en el Raspberry PI ocurre algo similar, ya que el script desarrollado permite la recolección de datos, instanciando la librería Xbee y Serial para Python. Este script se encuentra ejecutándose de forma indefinida y posee una función para la recepción e impresión de mensajes mediante ZigBee. En la siguiente imagen se puede observar una parte del script en donde se indica la interfaz de comunicación entre el Raspberry y el módulo Xbee Pro S1.

```
##############################################################################

"""
Interaz serial en donde se encuntra nuestro dispositivo
XBEE en nuestro caso, en el Raspberry, se encuentra
en /dev/ttyAMA0
"""
ser = serial.Serial('/dev/ttyAMA0', 9600)
xbee = XBee(ser, callback=mensajeRecibido)

##############################################################################
```

Figura 19: Interfaz de comunicación Raspberry con módulo Xbee

9.2.3. Almacenamiento De Datos

Para almacenar los datos que llegan al coordinador se ha decidido la utilización de una base de datos no relacional llamada MongoDB que a comparación de las bases de datos relacionales permite la administración y gestión de los datos de forma sencilla y dinámica, mediante el uso de estructuras de datos en formato JSON.

La base de datos creada en el Raspberry se llama **sensoresdb** y solamente puede ser accedida mediante autenticación ya que se crearon usuarios específicos de lectura y escritura, garantizando así la seguridad de la base de datos, por lo que es necesario conocer el usuario y contraseña para acceder a los datos. A continuación se puede observar la metodología de creación de las colecciones que posee la base de datos, cabe recalcar que en la colección **'sen'** es donde se almacenarán los mensajes que han llegado al coordinador.

Tabla 5: Creación de colecciones en MongoDB

---------- Creación de la colección 'sensores' ----------   db.createCollection('sensores')

```
db.sensores.insert({'codigo':'CO2', 'nombre':"Concentracion CO2",'descripcion':'Permite medir la concentración de dióxido de carbono en ppm', 'valores':[{'acronimo':"ppm", 'valor':"CO2 (ppm)"}]})

db.sensores.insert({'codigo':'part', 'nombre':"Calidad del aire",'descripcion':'Permite medir la calidad del aire', 'valores':[ {'acronimo':"dd", 'valor':"Dust Density (mg/m³)"},{'acronimo':"sng", 'valor':"Voltage Medido (v)"}]})

db.sensores.insert({'codigo':'dht11', 'nombre':"Temperatura y humedad",'descripcion':'Permite medir la temperatura y humedad del ambiente', 'valores':[{'acronimo':"c", 'valor':"Celsius (°C)"}, {'acronimo':"f", 'valor':"Fahrenheit (°F)"}, {'acronimo':"h", 'valor':"Humedad (%)"}, {'acronimo':"dp", 'valor':"Dew Point (°C)"}]})

db.sensores.insert({'codigo':'lux', 'nombre':"Luxómetro",'descripcion':'Permite medir la iluminancia del ambiente', 'valores':[{'acronimo':"lx", 'valor':"Iluminancia (lx)"}]})

db.sensores.insert({'codigo':'dbmeter', 'nombre':"Ruido Ambiental",'descripcion':'Permite medir la cantidad de ruido en el ambiente', 'valores':[{'acronimo':"db", 'valor':"Decibelios (dB)"}]})

---------- Creación de la colección 'dispositivos' ----------
db.createCollection('dispositivos')

db.dispositivos.insert({'nombre':"E1",'sensores':['CO2','part','dbmeter']})
db.dispositivos.insert({'nombre':"E2",'sensores':['CO2','part','dbmeter']})
db.dispositivos.insert({'nombre':"E3",'sensores':['CO2','dht11','dbmeter']})
db.dispositivos.insert({'nombre':"E4",'sensores':['CO2','dht11','lux','dbmeter']})
db.dispositivos.insert({'nombre':"E5",'sensores':['CO2','dht11','lux','dbmeter']})
db.dispositivos.insert({'nombre':"E6",'sensores':['CO2','dht11','dbmeter']})
db.dispositivos.insert({'nombre':"C3",'sensores':['CO2','dht11','lux','dbmeter']})

---------- Creación de la colección 'sen' ----------
db.createCollection('sen')
```

Además de la función para recolectar e imprimir los mensajes que llegan al coordinador, el script también cuenta con funciones de verificar los mensajes e insertar los datos en la base de datos, por lo tanto, para que los datos se almacenen correctamente primero tienen que pasar por la función verificar mensaje, caso contrario se mostrará un mensaje diciendo “Error el mensaje no paso la validación.”, cabe recalcar que el formato de validación es: texto=mensaje['disp'] texto = mensaje['sen'] texto=mensaje['m'] double=mensaje['val']

Para que los datos se almacenen adecuadamente, primero es necesario hacer una conexión a la base de datos sensoresdb mediante Python, e indicar algunos parámetros como: Dirección IP de la base de datos, puerto, usuario y contraseña, posteriormente indicar la colección en donde se va insertar los datos en este caso la colección es db.sen y finalmente cerrar la conexión. También se realiza un cálculo de la fecha actual que va ser insertado junto con los demás datos. A continuación se puede visualizar la función insertarDatos (mensaje).

Tabla 6: Función insertar datos en Python, en el nodo Fog

```
def insertarDatos(mensaje):
    try:

        x=datetime.datetime.now()
        fecha=('%s-%s-%s'%(x.year, x.month, x.day))
        hora=('%s:%s:%s'%(x.hour, x.minute, x.second))
        completo="ISODate("+   fecha+"T"+hora+".000Z)""
        dateN=datetime.datetime(x.year,  x.month,x.day,x.hour,  x.minute, x.second, 0)
        print "Insertando datos en la  BD. Fecha: " + completo
       dato = {'dp':str(mensaje['disp']),'sen':str(mensaje['sen']),'m':str(mensaje['m']),'val':(mensaje['val']),'fecha':dateN}
        client = MongoClient('127.0.0.1',27017)
        client.sensoresdb.authenticate('**************','***************')
        db = client.sensoresdb
        collection=db.sen
        collection.insert(dato)
        client.close()
        print "Datos Gurdados correctamente, en Sensoresdb."

    except Exception as e:
        print "Ocurrio el siguiente error: "+str(e)
        print "Tipo de Error: ", sys.exc_info()[0]
    except:
        print "Error Desconocido", sys.exc_info()[0]
```

9.3. Interacción Entre El Fog Computing y Cloud Computing

Para la replicación de los datos desde el nodo Fog Computing hacia la instancia de base de datos en el Cloud Computing se utilizó el protocolo MQTT, que permite el envío de mensajes de sensores, consumiendo un ancho de limitado ideal para transmisiones de baja tasa de datos.

9.3.1. MOSQUITTO

Mosquitto es un Broker de codigo libre que implementa el protocolo MQTT, y provee un método de comunicación de mensajería usando un modelo publicación /suscripción, se encarga de la centralización, gestión y administración de los tópicos o temas que los clientes finales han publicado o suscrito.

Mosquitto se encuentra ejecutándose en modo Background en la instancia base de datos, el cual se encuentra escuchando en el puerto 8883 que está definido para usar SSL. Dentro de esta instancia también se encuentra ejecutándose un script desarrollado en Python el cual permite suscribirse al Broker para ello es necesario indicar la dirección del certificado de la autoridad certificadora, así como los certificados propios de la instancia. Al suscribirse al Broker con el mismo tópico o tema que los clientes finales han publicado la instancia tiene la capacidad de recibir los mensajes publicados, con el fin de persistir los mensajes enviados por el nodo coordinador (Fog Computing) el mismo tiene ejecutando un script realizado en Python en el cual se encuentra una función que permite publicar las medidas recolectadas al Broker. Cabe resaltar que en el script del nodo Fog se deberá especificar la dirección del certificado de la autoridad certificadora caso contrario no se podrá publicar los mensajes al Broker, ya que toda la comunicación está configurada para usar SSL.

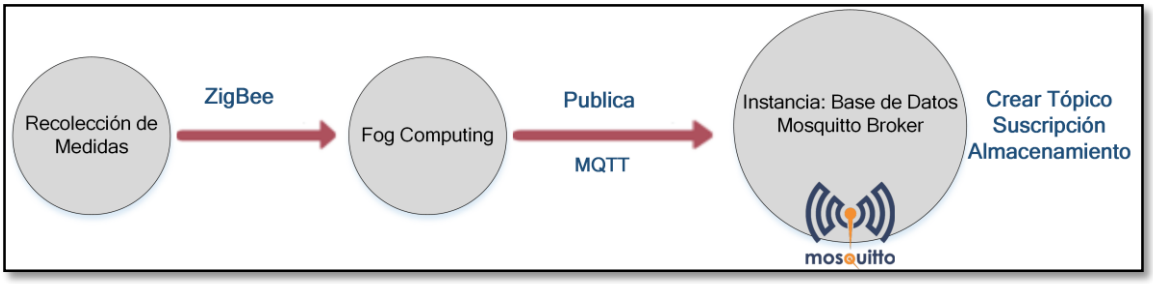


Figura 20: Comunicación entre Fog Computing y Cloud Computing

9.4. Cloud Computing

En el presente apartado se detalla la instalación, de OpenStack que es una plataforma de código libre para el despliegue de servicios virtualizados en el Cloud.

9.4.1. Requisitos Previos

En nuestro caso previamente se realizó la instalación Ubuntu 14.04 desktop el cual puede ser encontrado en la siguiente dirección:
http://releases.ubuntu.com/14.04/ubuntu-14.04.4-desktop-amd64.iso;
se utilizara la versión de 64 bits.

9.4.1.1. Actualizar Ubuntu

Después de tener una instalación limpia de Ubuntu 14.04 Desktop, es necesario actualizar el sistema operativo por tanto se ejecutaran en la terminal los siguientes comandos:

Para actualizar los repositorios: *sudo apt-get update*

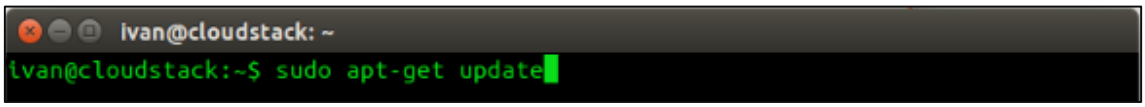


Figura 21: Comando para actualizar los repositorios

Para actualizar el sistema: *sudo apt-get upgrade*

Figura 22: Comando para actualizar el sistema

9.4.1.2. Instalar GIT

Git es un programa utilizado para el control de versiones, el cual nos permitirá descargar una copia de los repositorios existentes de OpenStack.

Ejecutar en terminal el siguiente comando: *sudo apt-get install git*

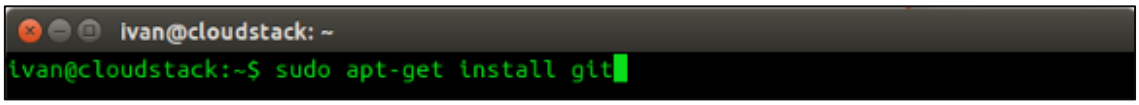


Figura 23: Comando para la instalación de GIT

9.4.2. Instalación de Openstack Devstack Kilo

9.4.2.1. Descargar Openstack Devstack Kilo desde sus repositorios

Se realizara la descarga desde los repositorios utilizando git en nuestro caso se utilizara la versión estable de kilo, existen otras versiones como liberty y mitaka disponibles para descargar.

Ejecutar en terminal el siguiente comando:
git clone https://www.github.com/openstack-dev/devstack.git -b stable/kilo

Figura 24: Descarga de Openstack devstack Kilo

9.4.2.2. Entrar en la carpeta devstack

La carpeta en donde se descargan todos los archivos de Openstack es "devstack", por tanto luego de haber finalizado la descarga es necesario entrar dicha carpeta.

Ejecutar en terminal el siguiente comando: *cd devstack*

Figura 25: Comando para entrar en la carpeta Devstack

9.4.2.3. Crear el archivo Local.conf

La creación de este archivo es opcional, sin embargo este archivo será utilizado para especificar ciertos parámetros de instalación de Openstack, como por ejemplo las direcciones de red para las instancias, contraseñas por defecto entre otros.

Ejecutar en terminal el siguiente comando: *sudo gedit local.conf*

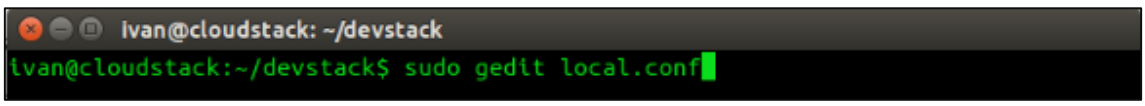


Figura 26: Creación y edición del archivo de configuración local.conf

A continuación se presenta el contenido del archivo utilizado para la presente instalación:

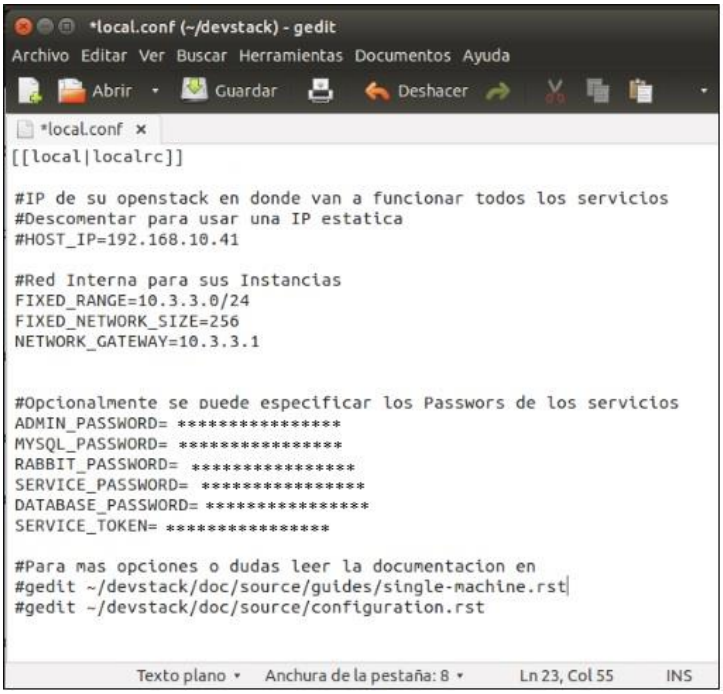


Figura 27: Contenido del archivo local.conf

9.4.2.4. MODIFICAR LA BASE DE REPOSITORIOS

Debido a problemas de compatibilidad entre versiones de programas necesarios en Openstack Kilo es necesario modificar la base de los repositorios de donde se descargara openstack. En este caso es necesario editar el archivo ***stackrc.***

Ejecutar en terminal el siguiente comando: *sudo gedit stackrc*

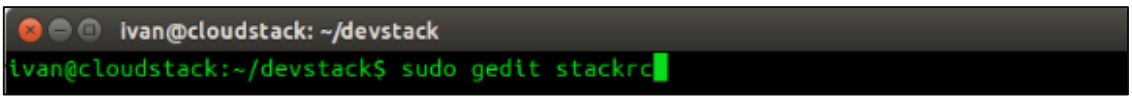


Figura 28: Comando para editar el archivo stackrc

Localizar la siguiente línea: *GIT_BASE=${GIT_BASE:-git://git.openstack.org}*

Figura 29: Línea a reemplazar el archivo stackrc

Reemplazar por: *GIT_BASE=${GIT_BASE:-https://www.github.com}*

Figura 30: Modificación del archivo stackrc

9.4.2.5. INSTALAR OPENSTACK

Luego de haber realizado todos los pasos anteriores se procederá con la instalación de openstack. Este proceso tomara entre una hora a dos horas dependiendo de la velocidad del internet.

Ejecutar en terminal el siguiente comando: *./stack.sh*

Figura 31: Comando para la instalación de Openstack

9.4.2.6. ACCEDER AL DASHBOARD WEB

Luego de haber finalizado la instalación de Openstack se podrá acceder a su dashboard web ingresando a la siguiente URL: **http://localhost/**

Se presentara la siguiente página de login cuyo User Name es: ***admin*** y el Password es la que se especificó en el archivo local.conf

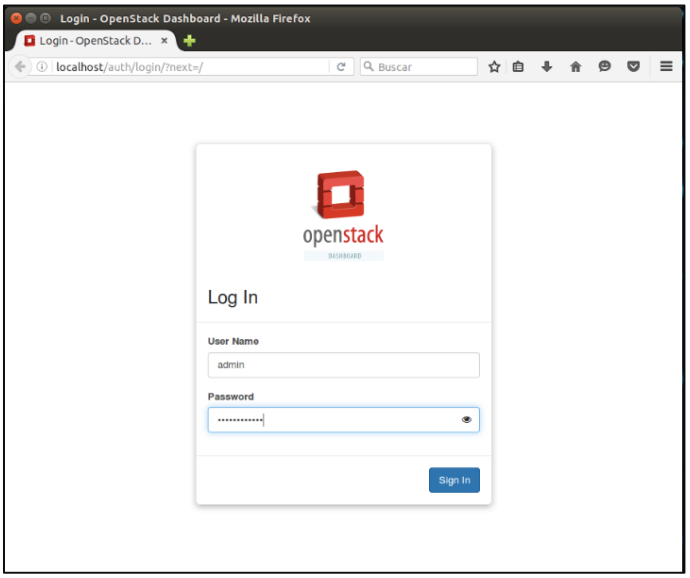


Figura 32: Página de login de Openstack

9.4.2.7. DASHBOARD DE ADMINISTRACIÓN

Luego de haber ingresado al dashboard de administración se podrá observar la siguiente página:

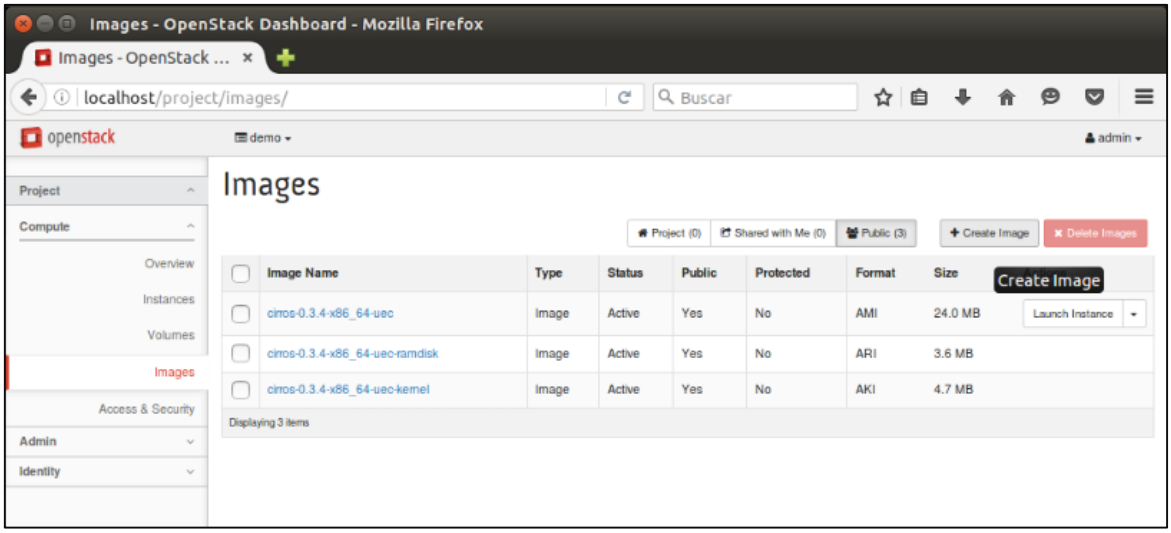


Figura 33: Dashboard de Openstack

9.4.3. CREACIÓN DE UNA INSTANCIA EN OPENSTACK

9.4.3.1. DESCARGAR UNA IMAGEN UBUNTU PARA CLOUD

Es necesario descargar una imagen para Cloud de algún Sistema Operativo en nuestro caso descargaremos Ubuntu Server 14.04 de la siguiente URL: *http://Cloud-images.ubuntu.com/trusty/current/trusty-server-Cloudimg-amd64-disk1.img*

Figura 34: Descarga de Ubuntu 14.04 LTS Cloud

Además se pueden encontrar diferentes imágenes en la siguiente URL:

http://docs.openstack.org/image-guide/obtain-images.html

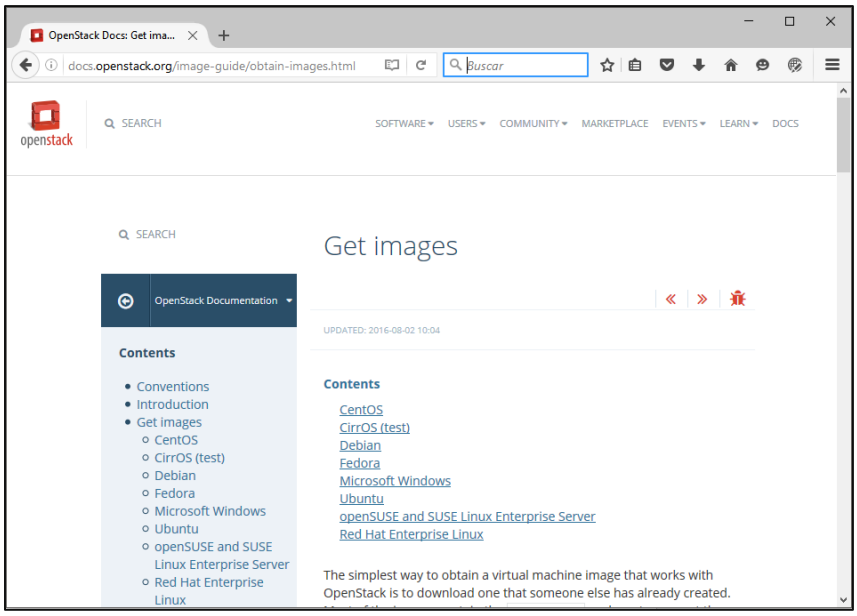


Figura 35: Página en donde se pueden encontrar imágenes para Openstack

9.4.3.2. Despliegue De las Instancias Web y Base de datos.

Para el funcionamiento de la plataforma de sensores es necesario desplegar instancias en Openstack, para lo cual es necesario la instalación de una imagen Ubuntu, posteriormente se deberá instalar en la primera instancia apache y PHP, mientras que en la segunda instancia se instaló la base de datos MongoDB en este caso no se entrara en detalle de la instalación de los mismos.

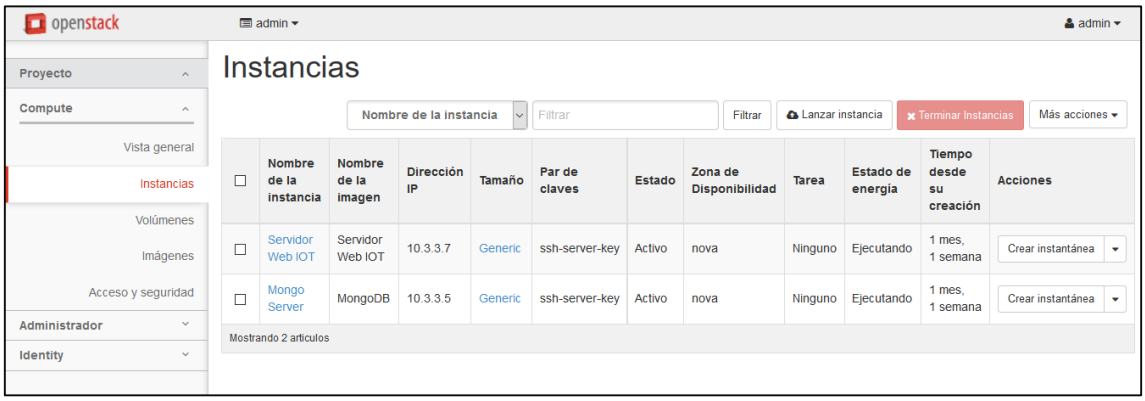


Figura 36: Instancias instaladas en Openstack

9.5. Desarrollo de plataforma web para visualización de los datos recolectados por la red de sensores

9.5.1. Funcionamiento General de la plataforma

El usuario accede a la página web de la plataforma, internamente se envían peticiones al servidor web solicitando los datos recolectados por los sensores, este obtiene los mismos de la instancia de base de datos MongoDB, posteriormente los procesa y los envía de vuelta al navegador del usuario para que finalmente se puedan generar los gráficos estadísticos en base a estos valores. A continuación se presenta el esquema de funcionamiento:

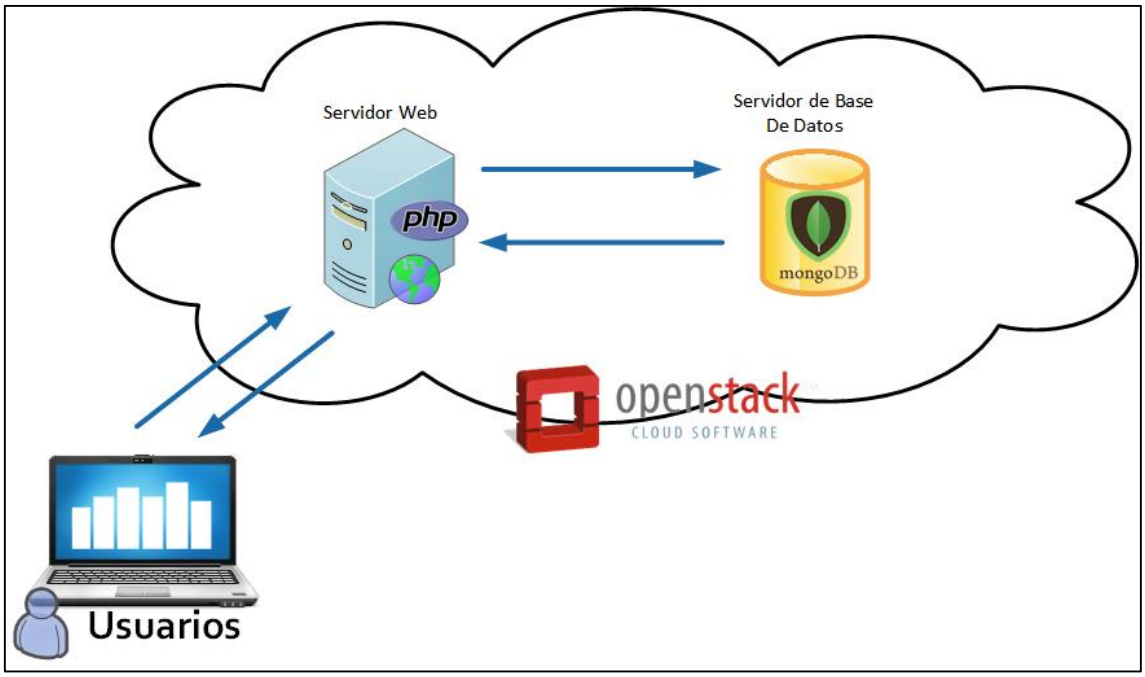


Figura 37: Esquema de funcionamiento de la Plataforma Web

9.5.2. Tecnologías Utilizadas

A continuación se presentaran las diferentes tecnologías que fueron utilizadas dentro del desarrollo de la plataforma:

9.5.2.1. PHP

PHP fue utilizado para recibir peticiones web originadas por acciones de los usuarios en la aplicación web, realizar las consultas necesarias en la base de datos y devolver los datos obtenidos en formato JSON para que los gráficos estadísticos

puedan ser generados. A su vez con PHP se generó una consola de administración que permite insertar, actualizar y eliminar dispositivos y sensores a la plataforma.

9.5.2.2. JavaScript

JavaScript fue utilizado para ejecutar acciones en base a la interacción de los usuarios con la aplicación web, estas acciones incluyen obtener los valores de los diferentes campos de los formularios para que se puedan generar las solicitudes a las paginas PHP. A su vez se encarga de procesar los datos de respuesta de las paginas PHP y pasarlos a la librería Amcharts. Finalmente permite generar diferentes objetos HTML de manera dinámica.

9.5.2.3. Amcharts

Amcharts es una librería JavaScript que permite la generación de gráficos estadísticos, y como su descripción lo indica fue utilizada para este mismo propósito. Los datos que utiliza esta librería son el resultado del procesamiento de las respuestas PHP con JavaScript.

9.5.2.4. Jquery

Jquery fue utilizado fundamentalmente para enviar solicitudes Web tipo AJAX a páginas PHP con el fin de obtener los datos que se utilizaran para generar las gráficas estadísticas.

9.5.2.5. Bootstrap

Bootstrap fue utilizado en el diseño web de la plataforma ya que posee diferentes componentes y permite que la aplicación web se adapte al tipo de pantalla.

9.5.2.6. Otras tecnologías utilizadas

Se utilizaron diferentes librerías para diferentes propósitos como animaciones, formatos de fechas, generación de componentes, cuadros de diálogos entre otras. A continuación se presenta dichas librerías: Moment.js, Datepicker.js Alertify.js, Jquery.matchHeight.js y Animate.css.

Estas tecnologías fueron escogidas ya que son Open Source y que juntas permiten la visualización de los datos recolectados por los sensores en forma de gráficos estadísticos de manera eficiente.

9.5.3. INTERFAZ GRÁFICA DE LA PLATAFORMA WEB

A continuación se presentará las interfaces graficas que componen la plataforma web.

9.5.3.1. PÁGINA DE INICIO

La página de Inicio contiene un menú superior para la navegación dentro de la plataforma, además en su contenido se encuentran enlaces para acceder a las páginas web de Dispositivos y Sensores, visualización de Grafico estadísticos e información del proyecto, finalmente en la parte inferior se presentan a los patrocinadores del proyecto. Revisar figura 38.

Figura 38: Página de inicio de la plataforma.

9.5.3.2. PÁGINA DE DISPOSITIVOS Y SENSORES

En esta página se presentan cada uno de los dispositivos y sus sensores, los cuales forman parte de la plataforma, además se presenta una breve descripción por sensor y sus parámetros de medición. Revisar figuras 39 y 40.

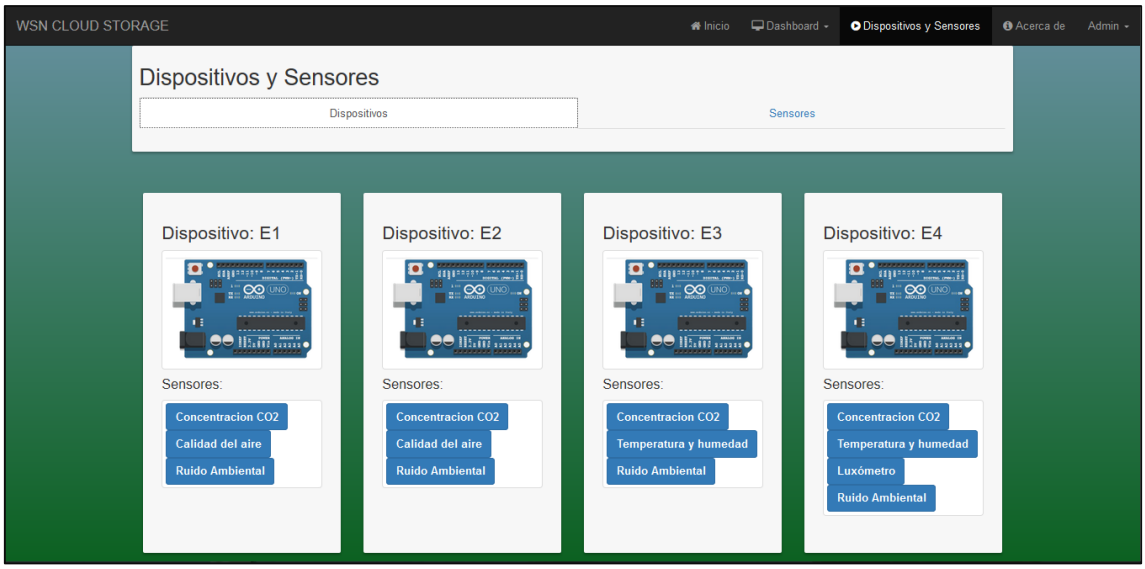


Figura 39: Página de Dispositivos y Sensores: Dispositivos.

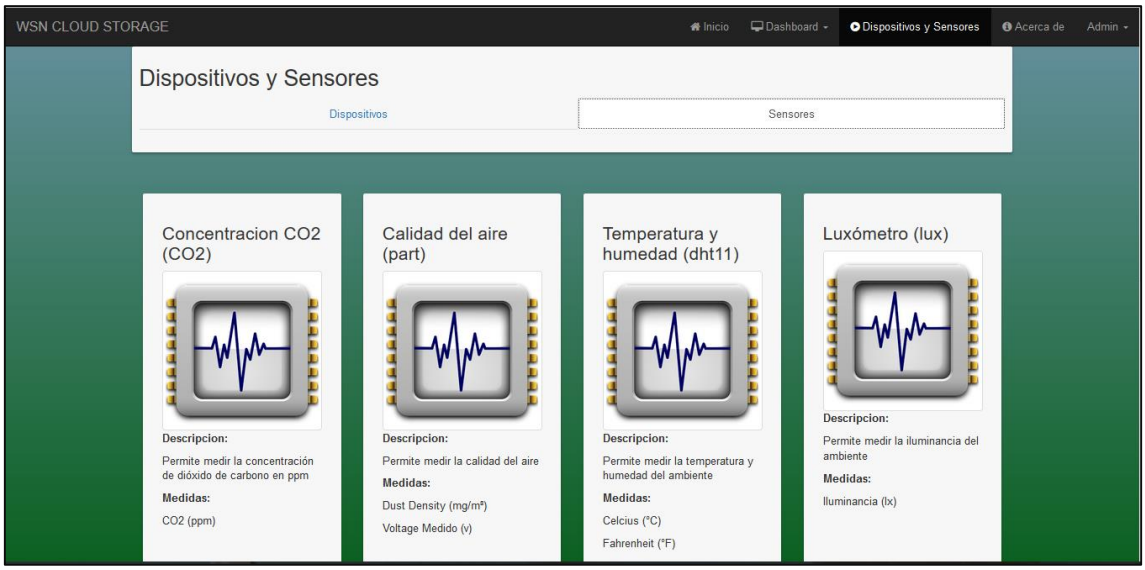


Figura 40: Página de Dispositivos y Sensores: Sensores.

9.5.4. VISUALIZACIÓN DE GRÁFICOS ESTADÍSTICOS

Los datos recolectados por los sensores de la plataforma pueden ser presentados mediante gráficos estadísticos con el objetivo de que el usuario pueda analizarlos y tener una visión más clara acerca de lo que está sucediendo. Los gráficos son creados de acuerdo a cada dispositivo, sus sensores y los valores que proporciona cada sensor, por tanto el usuario tendrá que seleccionar en base a un parámetro de hora, fecha o mes. Además de generar los gráficos estadísticos, la plataforma permite obtener los datos que generaron el grafico en formato JSON, esto con el objetivo de generar Open Data, es decir que los datos de los sensores pueden ser accedidos libremente.

9.5.4.1. REPORTES

Los reportes permiten visualizar los datos por intervalos específicos de tiempo en el presente proyecto se definieron tres tipos de intervalos: Hora, Diarios y Mensuales.

9.5.4.1.1. HORA

Permite graficar los datos recolectados por los sensores durante una hora específica del día, por defecto se visualizan los datos del día y hora actual de la computadora del usuario, sin embargo para visualizar los datos de otras fechas el usuario deberá seleccionar el día y la hora. Revisar figura 41.

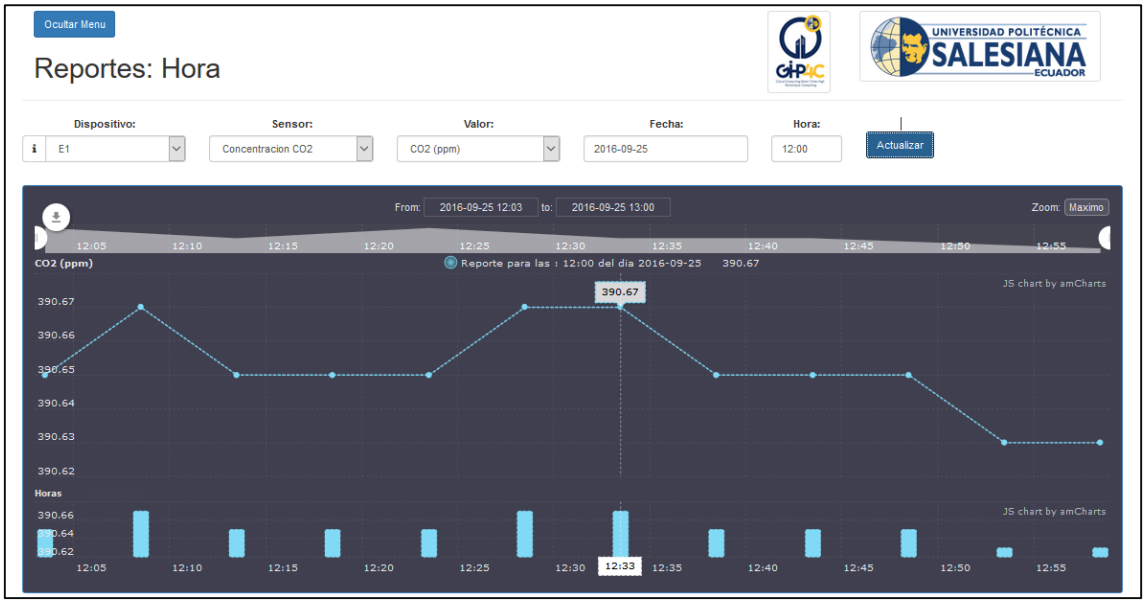


Figura 41: Reportes Hora

9.5.4.1.2. DIARIOS

Permite graficar los datos recolectados por los sensores durante un día específico por horas, es decir el grafico presenta el promedio del valor del sensor por cada hora del día. Es necesario que el usuario seleccione el día que desea visualizar. Revisar figura 42.

Figura 42: Reportes Diarios

9.5.4.1.3. MENSUAL

Permite graficar los datos recolectados por los sensores sacando el promedio del valor del sensor por horas durante un mes. Es necesario que el usuario seleccione el mes que desea visualizar. Revisar figura 43.

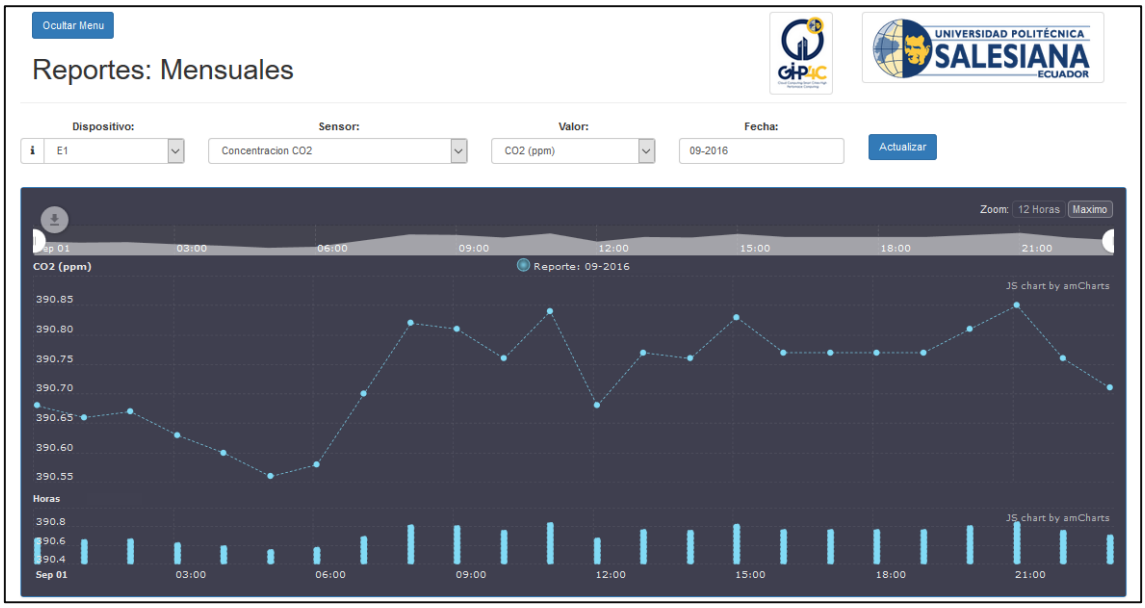


Figura 43: Reportes Mensuales

9.5.4.2. COMPARATIVAS

Como su nombre lo indica, las comparativas permiten comparar los datos recolectados por los sensores por horas, meses y días.

9.5.4.2.1. HORA ENTRE FECHAS

Permite graficar los datos recolectados por los sensores en una hora específica entre dos fechas. Es necesario que el usuario seleccione la fecha de inicio y fin y la hora. Revisar figura 44.

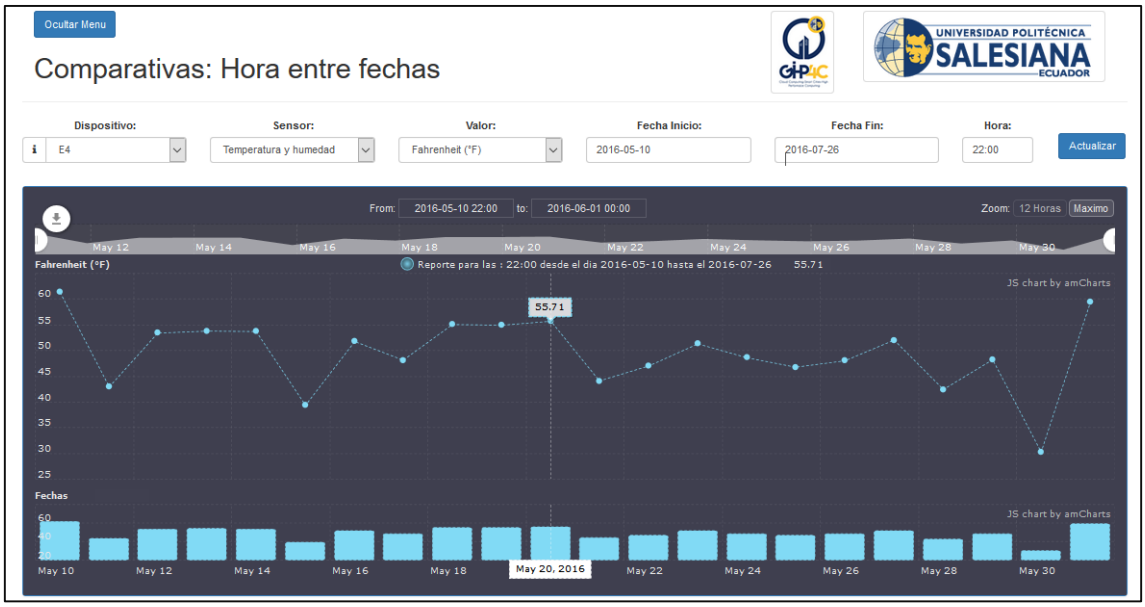


Figura 44: Comparativas Hora entre fechas

9.5.4.2.2. DIARIOS

Permite graficar los promedios por hora de los datos recolectados por los sensores de dos días diferentes. Es necesario que el usuario seleccione los días que desea comparar. Revisar figura 45.

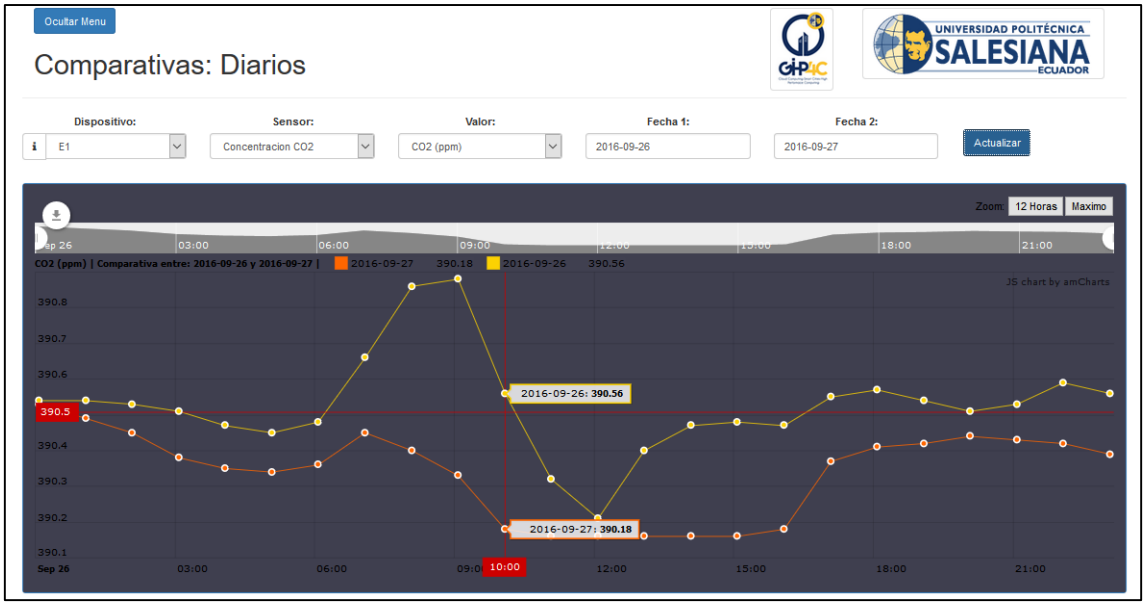


Figura 45: Comparativas Diarios

9.5.4.2.3. MENSUALES

Finalmente permite graficar los promedios por hora de los datos recolectados por los sensores de dos meses. Es necesario que el usuario seleccione los meses que desea comparar. Revisar figura 46.

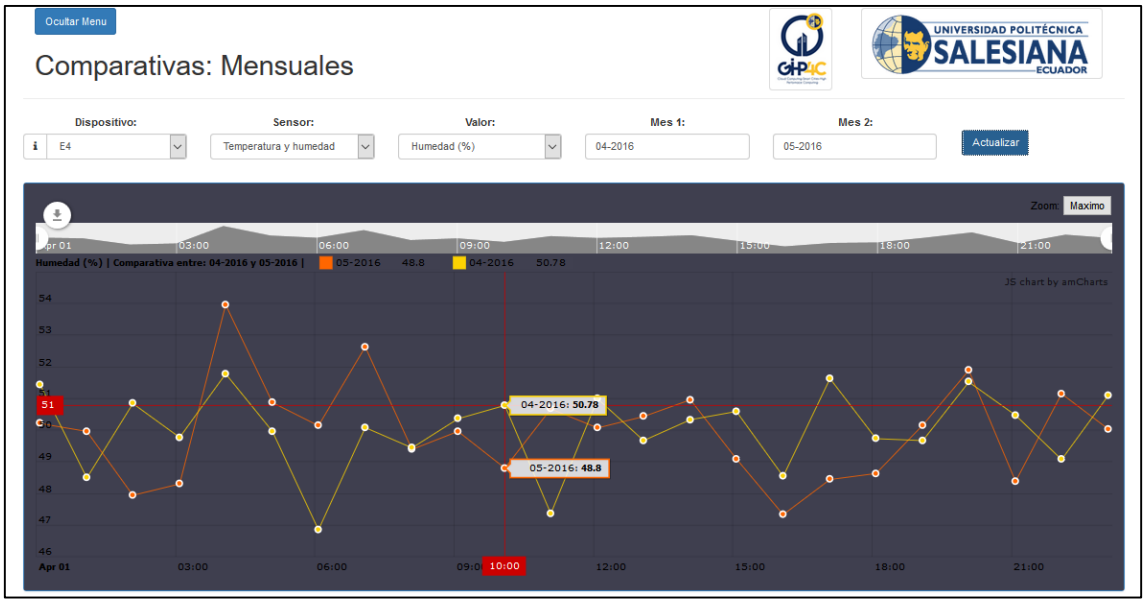


Figura 46: Comparativas Mensuales

9.5.4.3. DESCARGA DE INFORMACIÓN

Los gráficos generados pueden ser fácilmente descargados por el usuario, únicamente debe dar click en el botón de descarga, ubicado en la esquina superior derecha. Revisar figura 47.

A su vez existe la posibilidad de visualizar los datos que generaron el grafico en formato JSON el cual es estándar ampliamente utilizado para el intercambio de información. Debajo de cada grafico existe un botón el cual abre una ventana con dicha información. Revisar figura 48.

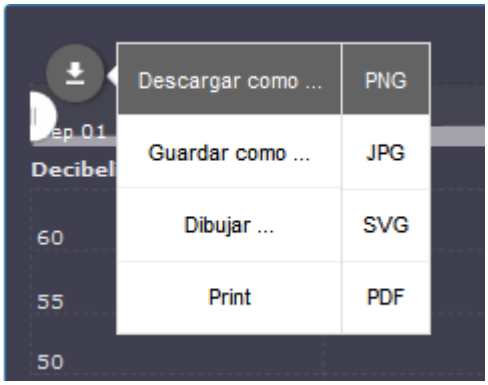


Figura 47: Descarga de grafico estadístico.

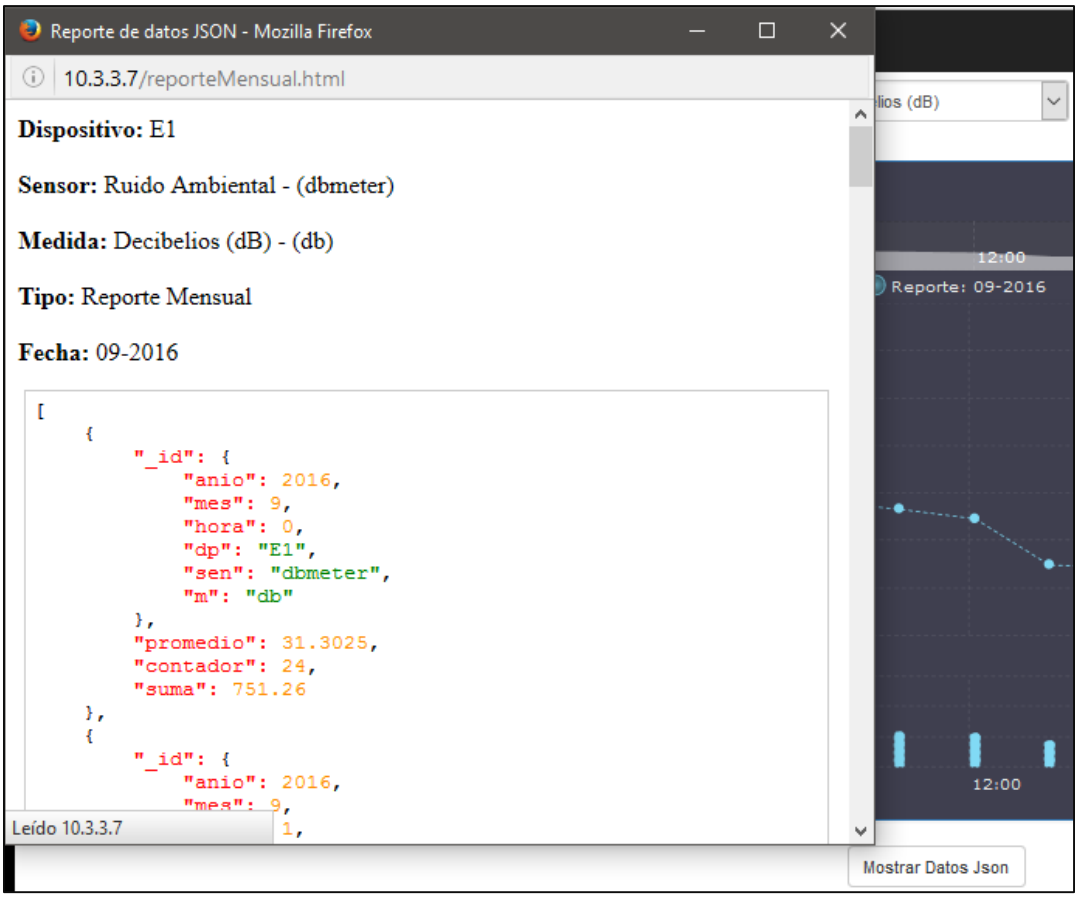


Figura 48: Visualización de datos JSON que generaron el grafico.

9.5.4.4. Administración de Dispositivos y Sensores

Con el objetivo de poder agregar, modificar y eliminar los dispositivos y sus sensores se desarrolló una sección de administración. A continuación se detallaran las partes que componen dicha sección.

9.5.4.4.1. Login de Usuario

Para poder acceder a la sección de administración es necesario estar autenticado, por tanto la plataforma solicita las credenciales del usuario al acceder a esta sección. Revisar figura 49.

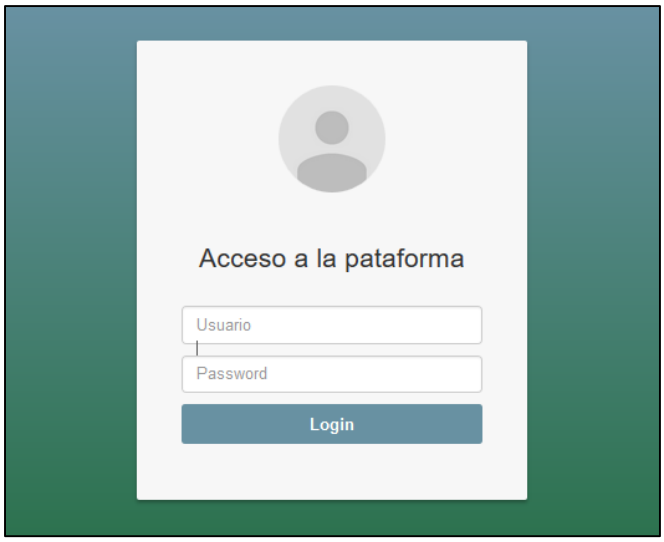


Figura 49: Página de Login para acceder a la administración de la plataforma

9.5.4.4.2. VISUALIZACIÓN DE DISPOSITIVOS Y SENSORES

Luego de iniciar sesión en la plataforma el usuario podrá visualizar los dispositivos de la plataforma, su número de registros, descripción y los sensores que posee y un botón que permite editar el dispositivo. Además se pueden visualizar los sensores, su descripción, sus parámetros de medición así como un botón para su edición. Revisar figuras 50 y 51.

Figura 50: Dispositivo visualizado en la sección de Administración.

Figura 51: Sensor visualizado en la sección de Administración.

9.5.4.4.3. AGREGAR NUEVO DISPOSITIVO

Inicialmente la plataforma posee ocho dispositivos, sin embargo se pueden agregar más dispositivos a medida que esta vaya creciendo. Por tanto para agregar un nuevo sensor, es necesario dar click en el botón nuevo dispositivo y se redireccionará al usuario a una nueva página, en la cual deberá especificar un nombre de dispositivo, una descripción y seleccionar los sensores que posee el dispositivo. El nombre del dispositivo a su vez debe ser especificado dentro del código del Arduino para que la plataforma lo pueda identificar. Revisar figuras 52 y 53.

Figura 52: Botón para agregar un nuevo dispositivo.

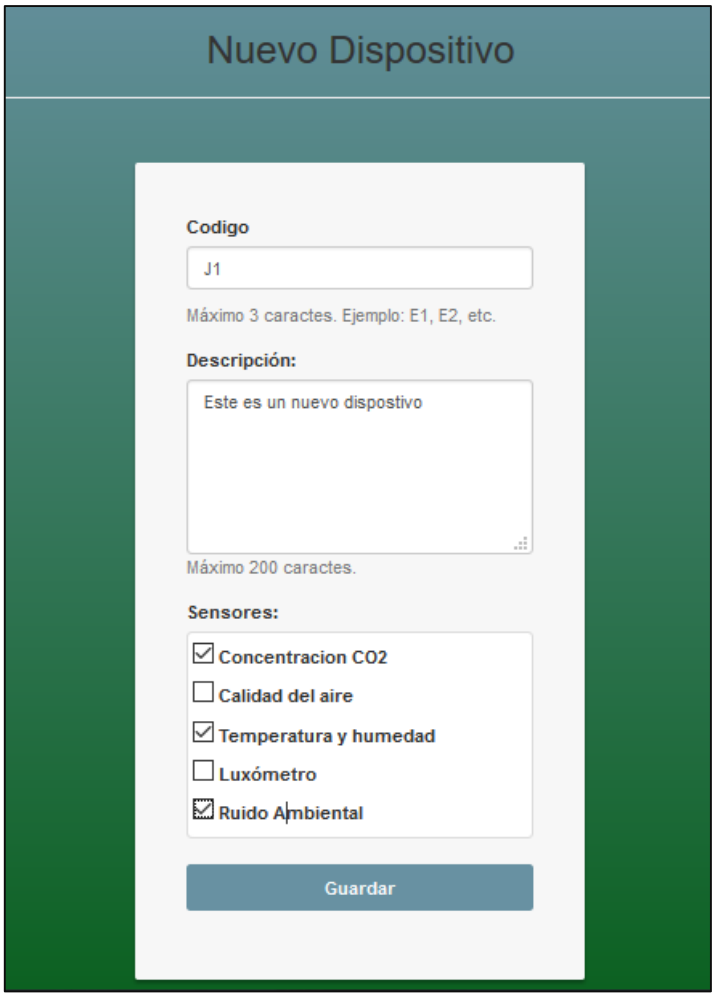


Figura 53: Creación de un nuevo dispositivo.

9.5.4.4.4. EDITAR O ELIMINAR UN DISPOSITIVO

Para poder modificar o eliminar un dispositivo es necesario dar click en el botón "Editar" que se encuentra en la parte inferior de cada dispositivo. Posteriormente se redirigirá al usuario a una página en la que podrá editar el dispositivo o a su vez eliminarlo. Revisar figuras 50 y 54.

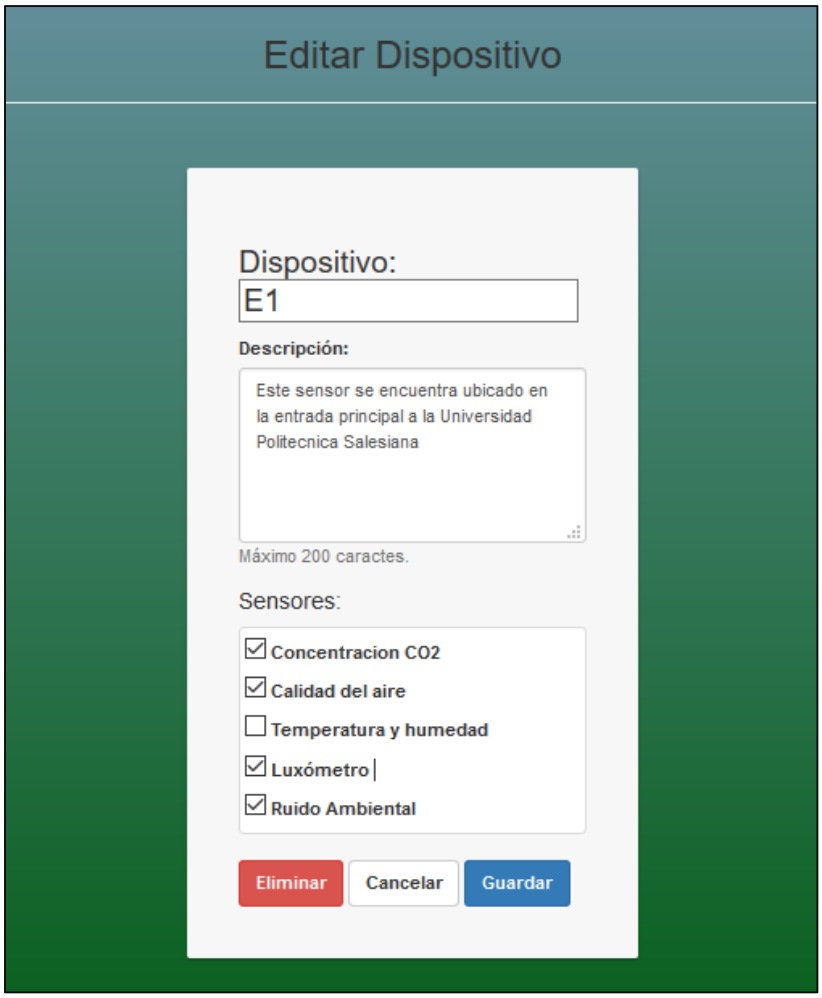


Figura 54: Edición de un dispositivo

9.5.4.4.5. AGREGAR UN NUEVO SENSOR

Para poder agregar un nuevo sensor, es necesario dar click en el botón nuevo sensor y se redireccionará al usuario a una nueva página, en la cual deberá especificar un código identificador de sensor, un nombre, los valores de medición con su acrónimo. Revisar figuras 55 y 56.

Figura 55: Botón para agregar un nuevo sensor

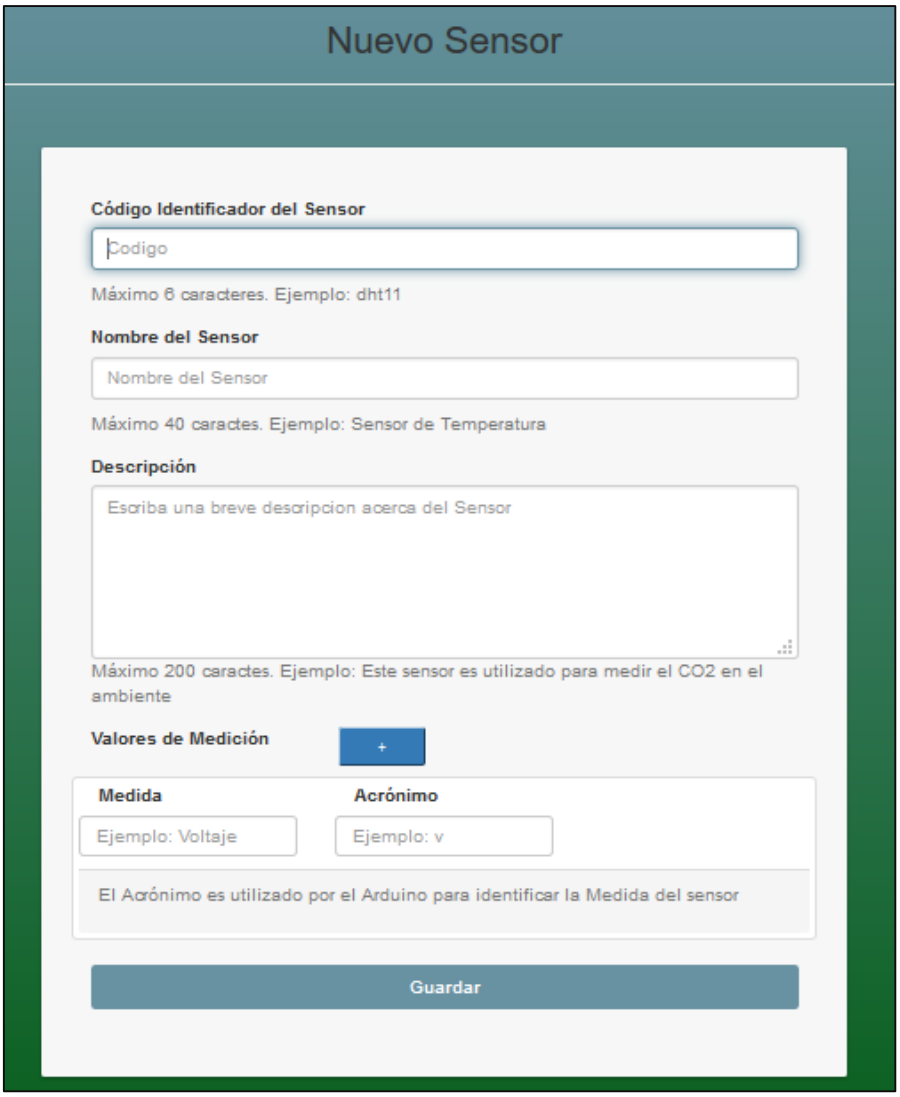


Figura 56: Creación de un nuevo sensor

9.5.4.4.6. EDITAR O ELIMINAR UN SENSOR

Similar al proceso con un dispositivo poder modificar o eliminar un sensor es necesario dar click en el botón “Editar” que se encuentra en la parte inferior de cada sensor el cual redirigirá al usuario a una página en la que podrá editar el sensor o a su vez eliminarlo. Revisar figuras 51 y 57.

Figura 57: Edición de un sensor.

10. Resultados

Los resultados obtenidos en el presente proyecto técnico se detallaran en base a las 4 capas que componen al mismo:

10.1. Red de Sensores

10.1.1. Sensores

Los sensores recolectan información durante intervalos de tiempo de 5 minutos, que en el presente proyecto se pudo constatar que es una buena métrica, para detectar las variaciones de factores contaminantes que existen en el ambiente y a su vez procesar y generar gráficos estadísticos en función de los mismos. En contraste reducir el intervalo de tiempo de recolección puede generar una gran cantidad de

información que requiere mayor cantidad de almacenamiento, procesamiento y que en la mayoría de las veces puede estar duplicada.

10.1.2. ZigBee como protocolo de comunicación IOT

Dentro del laboratorio de pruebas el uso del protocolo ZigBee fue eficiente, ya que permite transportar los datos recolectados por los sensores de manera cifrada con un retardo menor a 1 segundo y a una distancia probada de 300 metros con línea de vista (LOS). Cabe mencionar que el payload de un mensaje que se transporta mediante ZigBee es menor a 73 bytes (con cifrado habilitado), lo que es suficiente para enviar en cada mensaje el nombre del dispositivo, el código del sensor, el código de la variable de medición y finalmente el valor medido.

10.2. Fog Computing

Al usar Fog Computing como capa siguiente a la de red de sensores, en la plataforma se logró procesar de manera eficiente la información, en tiempo real, con un retardo menor a décimas de segundo, ya que aprovecha sus capacidades de computo mejorando el tiempo de respuesta en el tratamiento de datos, y por ende presentando resultados actualizados todo el tiempo.

10.2.1. Uso de Raspberry como nodo Fog

Usar un Raspberry como un dispositivo de Fog Computing nos permitió obtener una solución de bajo costo, confiable y con una capacidad de procesamiento suficiente para la plataforma desarrollada. Se debe mencionar que se hicieron pruebas con el envío de 39 datos simultáneos desde la red de sensores al Raspberry respondiendo de manera eficiente.

10.2.2. Recolección y procesamiento de Información

Los datos al ser recibidos por el Raspberry son sometidos a una validación con el fin de comprobar que estos se encuentren correctos, y son almacenados en la base de datos local del Raspberry (todo esto en un tiempo menor a un segundo) para posteriormente enviarlas al Cloud.

10.2.1. Interacción de Fog Computing con Cloud Computing

Los datos recolectados por el nodo Fog son enviados mediante el protocolo MQTT y recibidos en el Cloud con un retardo aproximado de 4 segundos producidos por la red de transporte.

Además se pudo comprobar las ventajas de MQTT como son: poco uso ancho de banda, integración de SSL y calidad de servicio.

10.2.2. Visualización de Información en el nodo FOG

Se pudo comprobar que los datos recibidos por la red de sensores en el nodo Fog pueden ser visualizados en tiempo en tiempo real, lo que logra una mejora significativa en cuanto a la obtención de datos en entornos en los que los sensores recolectan información crítica.

10.3. Cloud Computing

10.3.1. Uso de MongoDB

Durante el desarrollo de la plataforma web, se trabajaron con 4099680 registros de prueba, lo que equivale a un año de información recolectada por la red de sensores, con esta cantidad de registros se pudo comprobar que el uso de MongoDB como base de datos no relacional permite recuperar la información de los sensores en un periodo corto de tiempo sin afectar el rendimiento del sistema.

10.3.2. Telemetría como Servicio

En nuestro laboratorio de pruebas desplegado en el campus de la Universidad Politécnica Salesiana sede Cuenca se puedo evidenciar que este modelo planteado puede ser utilizado en cualquier otro tipo de proyecto en el que sea necesario realizar cualquier tipo de medición basada en sensores. Además dado que la plataforma desarrollada se encuentra en el Cloud esta puede ser replicada y de esta manera ofrecer Telemetría como servicio con la diferencia de que el solicitante del servicio debe implementar su propio nodo de Fog Computing, crear y administrar su propia red de sensores.

10.4. PRESENTACIÓN

Los resultados obtenidos son las gráficas estadísticas y comparativas de los valores medidos por los sensores, como se puede observar en la Figura 58 estas pueden ser fácilmente interpretadas por los usuarios, además estos gráficos pueden ser generados por horas, días o meses. Mientras que al mismo tiempo se integra la posibilidad de poder descargar estos gráficos y acceder a los datos que generaron el grafico estadístico en formato JSON.

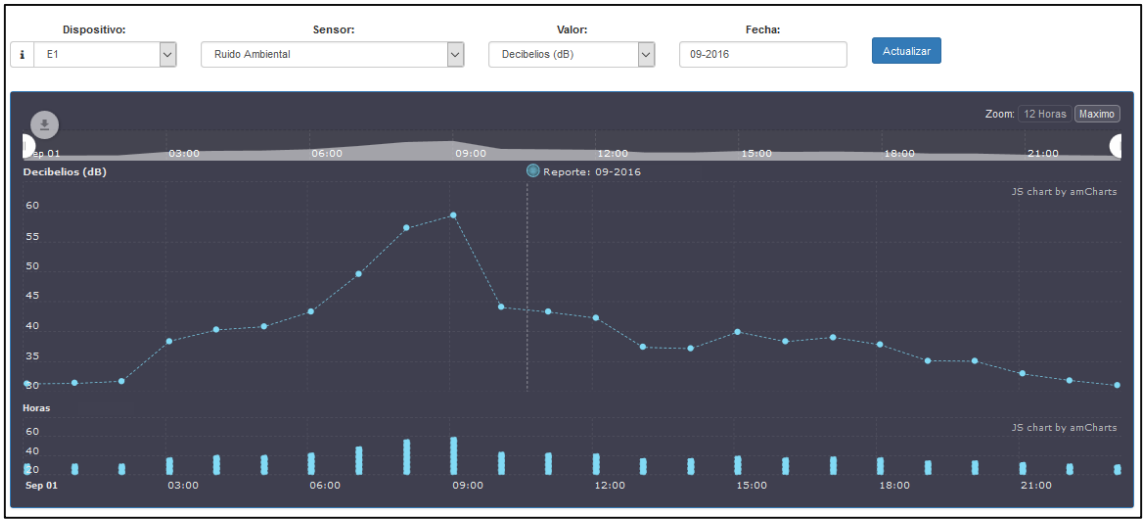


Figura 58: Ejemplo de grafico estadístico generado.

11. Conclusiones

Implementar una plataforma tecnológica que permite el monitoreo de la calidad del aire, detección de contaminación y temperatura ambiental, facilita la obtención de datos históricos sobre las variaciones de factores contaminantes con los cuales se pueden hacer proyecciones o estimaciones y tomar decisiones a futuro.

El uso del protocolo inalámbrico ZigBee posee cierta complejidad para su configuración, pero en contraste es ideal para entornos en donde se requiera comunicar dispositivos cuyas distancias no superen los trecientos metros, y la cantidad de información cifrada que se desea transportar sea menor a los setenta y tres bytes.

En una red de sensores es necesario contar con mecanismos que garanticen la seguridad de la información con la finalidad de que esta no sea manipulada, alterada e interceptada por terceros, es por eso que habilitamos en los módulos Xbee el protocolo AES de 128 bits para cifrar la información.

Al definir un intervalo de cinco minutos para la recolección de datos de la red de sensores, logramos evitar la duplicidad de la información y generar una métrica confiable la cual posteriormente puede ser presentada en la plataforma web.

Implementar un nodo Fog permite acercar la capacidad de cómputo al borde de una red de sensores, con lo cual se obtienen resultados en tiempo real, y de esta manera apoyar a la toma de decisiones en entornos donde la información medida sea critica, además almacenar un respaldo de la información generada en caso de pérdida de conexión con el Cloud.

MQTT es un protocolo de fácil implementación, bajo consumo de recursos y adecuado para el envío de datos hacia el Cloud, prestando integridad en el transporte de la información mediante el uso de SSL.

Dentro de la plataforma web al permitir que los usuarios puedan acceder a la información sobre la generación de los gráficos estadísticos, se garantiza que estos puedan realizar sus propios análisis e investigaciones en función de los mismos.

12. Recomendaciones

En función de los datos recolectados por la plataforma se pueden realizar una gran diversidad de análisis, además se pueden ampliar las funcionalidades de la misma, generando gráficos que permitan combinar diferentes tipos de medidas recolectadas por un conjunto de sensores, con el objetivo de buscar correlaciones o patrones entre los diferentes efectos contaminantes por ejemplo: entre el nivel de CO2, el ruido y temperatura de una determinada área.

Cuando se pretenda crear una plataforma con una cantidad mayor de sensores y que estos se encuentren desplegados por zonas extensas, se recomienda realizar geo posicionamiento de cada dispositivo, con el objetivo de poder agruparlos y permitir hacer estimaciones por sectores.

13. Referencias Bibliográficas

[1] Organización Mundial de la salud, «Cada año mueren 12,6 millones de personas a causa de la insalubridad del medio ambiente,» [En línea]. Available: http://www.who.int/mediacentre/news/releases/2016/deaths-attributable-to-unhealthy-environments/es/.

[2] ZigBee Alliance, «What is ZigBee?,» [En línea]. Available: http://www.zigbee.org/what-is-zigbee/.

[3] D. Caceres, *El aire de Cuenca en el límite de lo permisible,* 10 06 2016.

[4] F. Minchala, *Matriculación vehicular se incrementa en Cuenca,* 30 12 2015.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz y A. Konwinski, «Above the Clouds: A Berkeley View of Cloud,» 2009.

[6] P. Mell y T. Grance, «The NIST definition of cloud computing.,» 2011.

[7] K. A. Jamsa, Cloud Computing, Jones & Bartlett Publishers, 2013.

[8] B. Furht y A. Escalante, Handbook of Cloud Computing, Springer Science & Business Media, 2010.

[9] I. P. Rodriguez, J. E. Pettoruti, F. Chichizola y A. E. De Giusti, «Despliegue de un Cloud Privado para entornos de cómputo científico,» de *XI Workshop Procesamiento Distribuido y Paralelo*, 2011.

[10] B. Sushil, J. Leena y J. Sandeep, «CLOUD COMPUTING: A STUDY OF INFRASTRUCTURE AS A SERVICE (IAAS),» *International Journal of Engineering and Information Technology*, vol. 2, nº 1.

[11] M. Abuelhassan y M. Abdallah, «Cloud Computing Overview».

[12] Y. Duan, «Value Modeling and Calculation for Everything as a Service (XaaS) based on Reuse,» de *13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed*, 2012.

[13] S. C. S. M. Subhadeep Sarkar, «Assessment of the Suitability of Fog Computing in the Context of Internet of Things,» *IEEE Transactions on Cloud Computing*, 2015.

[14] S. M. Subhadeep Sarkar, «Theoretical modelling of fog computing: a green computing paradigm to support IoT applications,» *The Institution of Engineering and Technology 2016*, 2015.

[15] S. W. Ivan Stojmenovic, «The Fog Computing Paradigm: Scenarios and Security Issues,» de *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*.

[16] Cisco, «Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are,» 2015. [En línea]. Available: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf.

[17] H. W. S. L. G. D. H. YingJuan Shi, «The Fog Computing Service for Healthcare».

[18] MQTT, «What is MQTT?,» [En línea]. Available: http://mqtt.org/faq.

[19] Olimex , «Technical Data MQ-135 Gas Sensor,» [En línea]. Available: https://www.olimex.com/Products/Components/Sensors/SNS-MQ135/resources/SNS-MQ135.pdf.

[20] Micropik, «DHT11 Humidity & Temperature Sensor,» [En línea]. Available: http://www.micropik.com/PDF/dht11.pdf.

[21] Sharpsma, «Compact Optical Dust Sensor,» [En línea]. Available: www.sharpsma.com/webfm_send/1488.

[22] Digi International , «XCTU, Next Generation Configuration Platform for XBee/RF Solutions,» [En línea]. Available: http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu.

[23] Digi, «XBEE S1 802.15.4 RF Modules,» [En línea]. Available: http://www.digi.com/pdf/ds_xbeemultipointmodules.pdf.

14. Anexos

14.1. Nodos E1 – E2

14.2. Nodos E3 – E6

14.3. Nodos E5 –E6 – C3

14.4. Distribución de los nodos