ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UNA APLICACIÓN PROTOTIPO PARA LA OBTENCIÓN DE LA ZONA DE COBERTURA DE LA RED SIGFOX EN EL INTERIOR DE EDIFICACIONES

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

EDUARDO PAOLO REVELO VIZCAINO

DIRECTOR: MSc. CARLOS EGAS

Quito, diciembre del 2021

AVAL

Certifico que el presente trabajo fue desarrollado por Eduardo Paolo Revelo Vizcaíno, bajo mi supervisión.

MSc. Carlos Egas
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Eduardo Paolo Revelo Vizcaíno, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

EDUARDO PAOLO REVELO VIZCAÍNO

DEDICATORIA

A mi madre y abuelos, quienes han sido el pilar más importante, cuyo amor y cariño me motivan a ser cada día mejor.

A Alejandra por ser mi compañera en esta travesía y estar conmigo en los buenos y malos momentos.

AGRADECIMIENTO

Quiero agradecer primeramente a Dios que siempre ha estado conmigo y nunca me ha abandonado.

A mi madre y abuelos, por estar a mi lado y que siempre me han brindado su apoyo y confianza.

A Alejandra por ser la persona que ha sido mi compañera en todo este camino, por nunca dejarme solo, esta fue la mejor etapa de mi vida porque tú me has hecho muy feliz.

Mi gratitud a mi tutor, Carlos Egas por brindarme su confianza y haberme guiado en el desarrollo de este trabajo de titulación.

A mis queridos primos que son como hermanos para mi agradecerles por ser un apoyo y que todas las vivencias que hemos tenido juntos las llevare conmigo siempre.

A mis amigos, que todas las risas y momentos compartidos han hecho que mi vida universitaria sea inolvidable.

ÍNDICE DE CONTENIDO

1. Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

RESUMEN

El presente trabajo de titulación describe el diseño e implementación de una aplicación prototipo para la obtención de la zona de cobertura de la red SigFox en el interior de edificaciones. El prototipo consta de un nodo SigFox Pycom que permite el envío de mensajes a través de la red SigFox, una aplicación que permite su configuración para el envío de mensajes a los servidores propios de la plataforma SigFox conocido como BackEnd SigFox, una aplicación servidor que permite obtener los mensajes del BackEnd SigFox y una aplicación de usuario que facilita la interacción gráfica con el prototipo.

El presente documento consta de 4 cuatro capítulos:

El primer capítulo presenta los conceptos acerca de la red SigFox, características del nodo SigFox Pycom, el lenguaje de programación para su configuración MicroPython, conceptos de aplicaciones web de la arquitectura REST y el lenguaje de programación JavaScript.

El segundo capítulo presenta la metodología empleada para la realización del prototipo, los requerimientos funcionales y no funcionales de la aplicación que fueron obtenidas a través de entrevistas, el diseño y la implementación de todos los subsistemas en donde se hace una explicación de las partes de código más importantes que conforman al prototipo.

El tercer capítulo contiene las pruebas del prototipo, las cuales se llevaron a cabo dentro del edificio de Química – Eléctrica en el campus de la Escuela Politécnica Nacional y en el Gimnasio “Ignite” ubicados en la ciudad de Quito, en ambos se comprueba que cumpla los requerimientos que se encuentran en el segundo capítulo.

Por último, en el cuarto capítulo se encuentran las conclusiones y recomendaciones del presente trabajo de titulación.

PALABRAS CLAVE: SigFox, Pycom, MicroPython, REST, JavaScript

ABSTRACT

This project describes the design and implementation of a prototype application to obtain the coverage area of the SigFox network inside buildings.

The prototype consists of a SigFox Pycom node that allows the sending of messages through the SigFox network, an application that allows its configuration to send messages to the servers of the SigFox platform known as BackEnd SigFox, a server application that allows to get the messages from the BackEnd and a user application that allows a graphic interaction with the prototype.

This document is made up of 4 chapters:

The first chapter presents the concepts about the SigFox network, characteristics of the SigFox Pycom node, the MicroPython programming language for its configuration, concepts of REST architecture web applications and the JavaScript programming language.

The second chapter presents the methodology used for the realization of the prototype, the functional and non-functional requirements of the application that were obtained through interviews, the design, and the implementation of all the subsystems where an explanation of the parts of most important code that make up the prototype.

The third chapter contains the tests of the prototype, which were carried out inside the "Química - Eléctrica" building on the campus of the Escuela Politécnica Nacional and in the "Ignite" Gymnasium located in the city of Quito, in both it is verified that meets the requirements found in the second chapter.

Finally, in the fourth chapter are the conclusions and recommendations of the present project.

KEYWORDS: SigFox, Pycom, MicroPython, REST, JavaScript

1. INTRODUCCIÓN

La creciente importancia de Internet de las cosas (IoT) crea una necesidad cada vez mayor de estándares de comunicación de área extendida que garanticen una conectividad confiable entre una multitud de dispositivos de IoT. SigFox es una red IoT de área extendida, de bajo consumo de potencia, que admite comunicaciones de largo alcance y alta escalabilidad de dispositivos finales.

La red SigFox proporciona un mapa de cobertura de su red en su página oficial, sin embargo, no proporciona información detallada de la cobertura de red en los interiores de áreas de campus, edificaciones, casas, escuelas, etc. Lugares en donde se colocarán los dispositivos de monitoreo asociados con el IoT que utilizarán esta red. Por lo tanto, no se dispone de información detallada del nivel de la señal de SigFox en ubicaciones dentro de las edificaciones. El desconocimiento de los sitios donde no existe señal trae problemas en la conectividad de la red. El no conocer el área donde la cobertura de la red SigFox es mínima o nula imposibilita la colocación de nodos sensores en esas zonas y por lo tanto imposibilita el satisfacer los requerimientos de los usuarios.

Por otra parte, es necesario contar con una herramienta que facilite el diseño de la red, para asegurar que todos los objetos a los cuales se conectan los nodos sensores puedan ser monitoreados dentro de una edificación además de facilitar la inspección del sitio, actividad muy común en el diseño y la implementación de comunicaciones inalámbricas.

1.1. OBJETIVOS

El objetivo general de este Proyecto Técnico es: Desarrollar una aplicación prototipo para la obtención de la zona de cobertura de la red SigFox en el interior de edificaciones.

Los objetivos específicos del Proyecto Técnico son:

- Analizar la información teórica requerida para el desarrollo del prototipo.
- Diseñar la aplicación de usuario, aplicación servidor y la aplicación para configuración de los nodos sensores.
- Implementar las aplicaciones diseñadas.
- Analizar los resultados obtenidos a partir de las pruebas realizadas con el prototipo desarrollado.

1.2. ALCANCE

En este proyecto se implementa una aplicación prototipo que se compone de: nodos sensores Pycom, una aplicación servidor, una aplicación de usuario, una aplicación para configuración de los nodos y la base de datos que forma parte de la aplicación de usuario. Para el funcionamiento de la aplicación prototipo se requiere del uso de la red internet y de la red SigFox que está compuesta por la nube SigFox (BackEnd) y de las estaciones de radio SigFox, que funcionan como gateways de los nodos SigFox, tal como se aprecia en el esquema de la red presentado en la Figura 1.1.

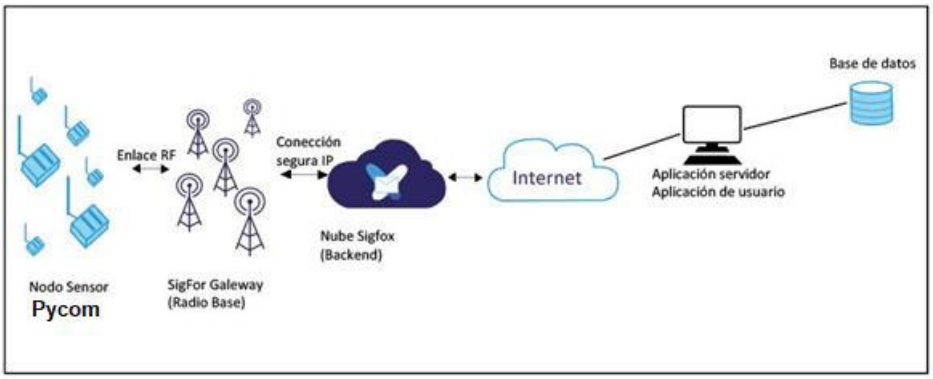

Figura 1.1 Esquema del sistema de monitoreo.

La implementación del prototipo implica el desarrollo de un software para la operación de los nodos sensores Pycom que permita enviar información a las radios bases SigFox y un software para los aplicativos que se ejecutan en el computador personal que permita el desarrollo de la aplicación servidor, la aplicación de usuario y la base de datos.

El software para los nodos sensores Pycom permite la interacción entre el usuario y el nodo sensor Pycom con el propósito de establecer los parámetros de la aplicación de usuario y poder enviar los datos al BackEnd de SigFox. Los nodos sensores Pycom envían al BackEnd de Sigfox información relacionada con:

- La ubicación dentro de la edificación.
- El Indicador de calidad de la señal (LQI).
- Un código que permita la identificación del diagrama.
- Un código que permita indicar la posición del nodo dentro de la edificación.

La información enviada por los nodos sensores Pycom es procesada en la aplicación de usuario con el propósito de presentar, de manera visual, los niveles de intensidad de señal en la edificación en la que se está realizando la inspección.

Las aplicaciones por implementar son las siguientes:

- Aplicación servidor que permite que la computadora se conecte con el BackEnd SigFox, para recuperar los datos obtenidos del nodo sensor, esta conexión al BackEnd de SigFox se la realiza mediante el uso de la API (Interfaz de programación de aplicaciones) REST (*Representational State Transfer*) proporcionadas por el BackEnd de SigFox [7]. La API REST permite obtener los datos que envían los nodos sensores al BackEnd y serán almacenados en una base de datos específicamente creada para el efecto, de igual manera este servidor contará con un servicio API REST que permita a la aplicación de usuario acceder a la información guardada en la base de datos que será implementada para tal efecto.

- Aplicación de usuario que se conecta a la aplicación servidor utilizando la API REST. Estas API REST son implementadas en la aplicación servidor y de esta manera poder acceder a la información de la base de datos, la aplicación de usuario hará uso de un plano de la edificación en donde se obtendrá los valores del LQI en zonas puntuales, el cual será previamente ingresado en formato digital para su utilización en la aplicación de usuario. La aplicación presenta los niveles de intensidad de la señal y el diagrama de cobertura de la señal de SigFox como se aprecia en la Figura 1.2. La aplicación de usuario presentará la información relacionada con el nodo sensor seleccionado, el cual será presentado con un color dependiendo del LQI recibido. La aplicación de usuario almacenará la información relacionada con las ubicaciones que tendrán los nodos sensores durante el proceso de la obtención de la zona de cobertura.

Figura 1.2 Ejemplo del plano de la edificación.

- Aplicación para la configuración de los nodos sensores Pycom permite actualizar la configuración de los nodos en el sitio en el cual se está realizando la obtención de la zona de cobertura, mediante una conexión directa, en el caso de ser necesario. Si bien, al

terminar la fase de obtención de los requerimientos para la aplicación se obtendrán los parámetros que deben ser configurados. En principio los parámetros que van a ser configurados en el nodo para su operación óptima son: la ubicación dentro del edificio, intervalos de envíos de mensajes e identificación del diagrama.

La aplicación servidor y de usuario serán desarrolladas usando la arquitectura REST que permite la comunicación entre el Servidor (aplicación servidor) y Cliente (aplicación de usuario) como se muestra en la Figura 1.3.

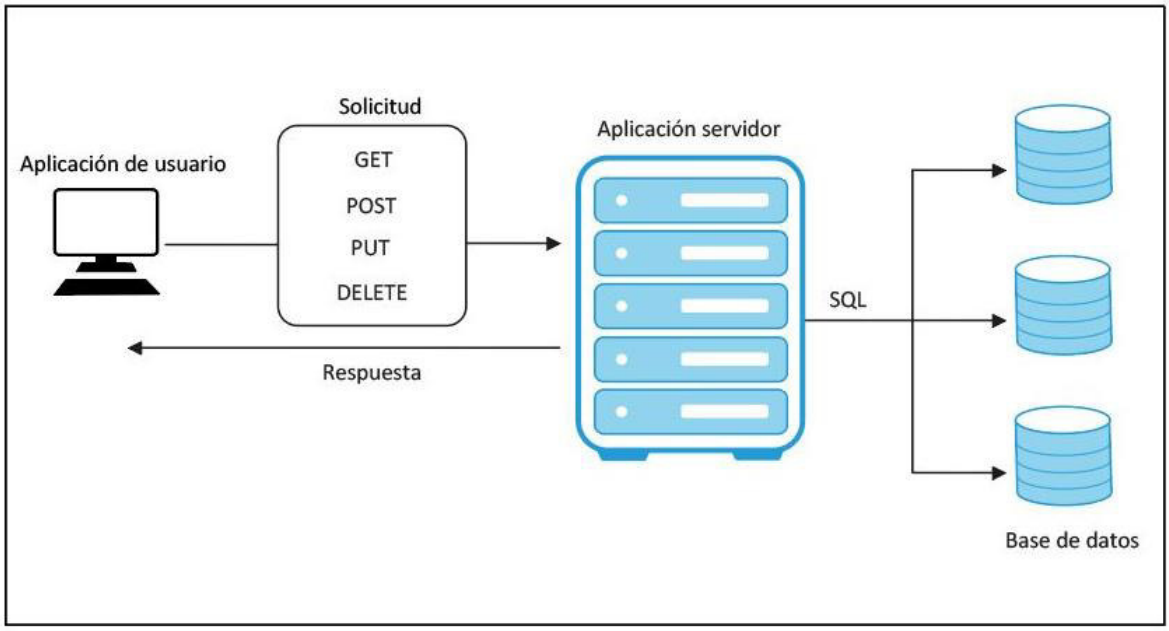

Figura 1.3 Arquitectura REST.

Previa a la realización de la inspección de sito para obtener la zona de cobertura de la red SigFox, es necesario definir con anticipación los lugares donde se ubicarán los nodos sensores dentro de la edificación y el tiempo que el nodo estará operando en cada ubicación para obtener el nivel de intensidad de la señal.

El proceso de inspección del sitio para obtener el diagrama de cobertura de la red SigFox requiere de la utilización de varios nodos sensores a ser ubicados dentro de la edificación para evaluar los niveles de señal de una manera más rápida, esto debido a que la red SigFox tiene una limitante en el tiempo mínimo entre dos transmisiones consecutivas de un nodo, cuyo valor es de 10 minutos aproximadamente. Las pruebas del prototipo se realizarán en por lo menos dos edificaciones en las cuales se obtendrán los diagramas de cobertura, una de las edificaciones será el Edificio de Química-Eléctrica, y la otra edificación será el Gimnasio Ignite Ubicado en la ciudad de Quito.

1.3. MARCO TEÓRICO

1.3.1. Red SigFox

SigFox es una red de bajo costo, bajo consumo de energía y confiable que permite conectar dispositivos utilizados en el IoT (Internet de las cosas) [1]. Es una tecnología de red de área extendida de bajo consumo de energía LPWAN (*Low Power Wide Area Networks*) que permite la transmisión de datos entre un dispositivo a una radio base lejana ubicada a kilómetros o cientos de metros [2].

La red SigFox está implementada en Ecuador y es operada por la empresa WND (Wireless Network Development), actualmente tiene cobertura dentro de las principales ciudades del Ecuador [3], la Figura 1.4 muestra la cobertura en el año 2021 de la red SigFox en el Ecuador.

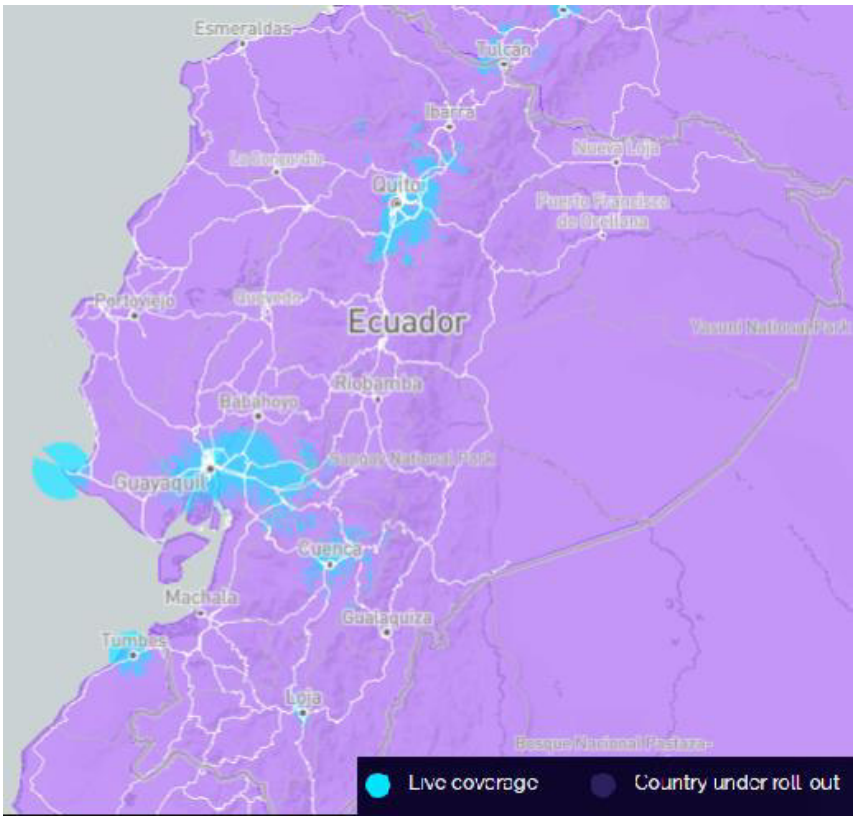

Figura 1.4 Mapa de cobertura de la red SigFox en Ecuador.

1.3.1.1 Características

La red SigFox se encuentra diseñada de forma eficiente que permite a los dispositivos que usen esta red tengan una larga duración de sus baterías. SigFox hace uso del espectro no licenciado y opera a diferentes frecuencias dependiendo de la zona geográfica en la que

se encuentre el dispositivo, como se muestra en la Figura 1.5, a Ecuador le corresponde la zona denominada RC4 (Radio Configuration 4).

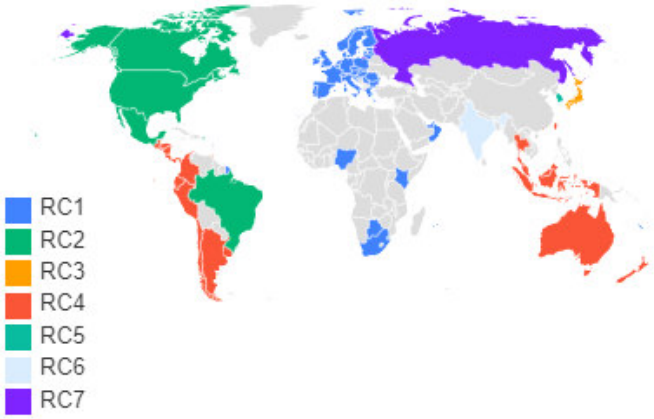

Figura 1.5 Zonas geográficas SigFox [5].

La zona geográfica además define los parámetros con los que el dispositivo operará, como se muestra en la Tabla 1.1.

Tabla 1.1 Parámetros técnicos SigFox por zona geográfica [5].

	RC1	RC2	RC3	RC4	RC5	RC6	RC7
Frecuencia central del enlace de subida (MHZ)	868.130	902.200	923.200	920.800	923.300	865.200	868.800
Frecuencia central del enlace de bajada (MHZ)	869.525	905.200	922.200	922.300	922.300	866.300	869.100
Velocidad de transmisión del enlace de subida (bit/s)	100	600	100	600	100	100	100
Velocidad de transmisión del enlace de bajada (bit/s)	600	600	600	600	600	600	600

La zona geográfica también define la técnica que se usa para evitar la pérdida de mensajes, la zona RC4 que le corresponde a Ecuador hace uso de la técnica de salto de frecuencia (*Frecuency Hopping*) de la siguiente manera: Cada mensaje que envía el dispositivo es enviado 3 veces a 3 diferentes frecuencias. Tiene un tiempo máximo de transmisión por canal de 400 ms y no existen transmisiones hasta después de 20 segundos [5].

1.3.1.2 Arquitectura de red SigFox

La arquitectura de la red SigFox se compone de dispositivos finales como sensores o actuadores, radio bases y una red central como se muestra en la Figura 1.6.

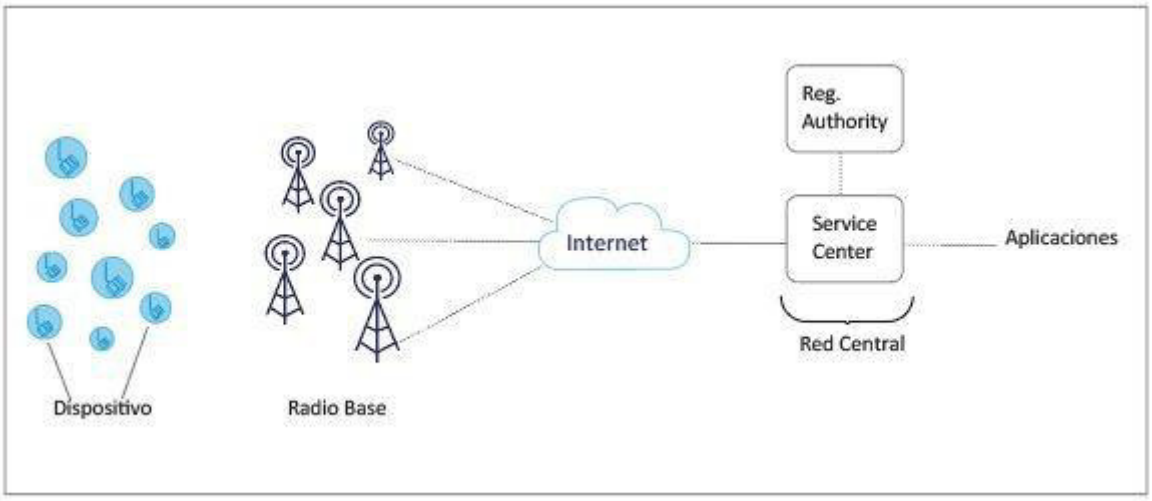

Figura 1.6 Arquitectura de red SigFox [6].

- Dispositivos: Son dispositivos finales que tienen la capacidad de enviar y recibir mensajes a las radios bases SigFox. Estos dispositivos contienen un identificativo único de 4 bytes.
- Radio Bases: Son estaciones de radio que tienen la capacidad de enviar y recibir mensajes de los Dispositivos por medio de la configuración de radio establecida por cada país. Estas estaciones de radio mantienen una comunicación cifrada con la Red Central por medio del internet.
- Red Central: Se encuentra compuesto por el centro de servicios (*Service Center*) que se encarga de la conexión de datos entre las radio bases con el internet, además de la administración de las radio bases y los dispositivos, y la autoridad de registro (*Registration Authority)* encargada de dar acceso a la red a los dispositivos finales [7].

Los dispositivos se conectan a la red SigFox de manera inalámbrica a través de la radio base más cercana. Las radios bases se conectan a internet a través de la red central. Las aplicaciones interactúan con los datos que se recolectan por los dispositivos, a través de una interfaz web y una serie de interfaces de programa de aplicación (API) [6].

1.3.1.3 Comparación de la red SigFox con otras redes LPWAN.

Para esta comparación se escogió a las principales alternativas LPWAN, presentadas en la Tabla 1.2.

Tabla 1.2 Tabla comparativa entre tecnologías LPWAN [6]

	SigFox	LoRaWAN	NB-IoT
Modulación	BPSK (Modulación por desplazamiento de fase binaria)	CSS (Chirp de espectro extendido)	QPSK (Modulación por desplazamiento de fase en cuadratura)
Frecuencia	Banda ISM sin licencia RC1 868 - 878.6 MHz RC2 902.13 - 904.66 MHz RC3 922.3 - 923.5 MHz RC4 920.13 - 922.66 MHz RC5 922 - 923.4 MHz RC6 865 - 867 MHz	Banda ISM sin licencia Europa 867 – 869 MHz América del Norte 902 – 908 MHz China 470 – 510 MHz Corea y Japón 920 – 925 MHz India 865 – 867 MHz	Frecuencia licenciada LTE B2 (Ul – Dl) 1850 MHz – 1910 MHz 1930 MHz – 1990 MHz B3 (Ul – Dl) 1710 MHz – 1785 MHz 1805 MHz – 1880 MHz B5 (Ul – Dl) 824 MHz – 849 MHz 869 MHz – 894 MHz B28 (Ul – Dl) 703 MHz – 748 MHz 758 MHz – 803 MHz
Ancho de banda	100 Hz	250 kHz y 125 kHz	200 kHz
Velocidad de transmisión máxima	100 bps o 600 bps dependiendo de la región	50 kbps	200 kbps
Máximo mensajes por día	140 enlace de subida y 4 enlace de bajada	Ilimitado	Ilimitado
Tamaño máximo de carga útil	12 bytes enlace de subida y 8 bytes enlace de bajada	243 bytes	1600 bytes
Rango de cobertura de una radio base	10 km (Urbano) y 40 km (Rural)	5 km (Urbano) y 20 km (Rural)	1 km (Urbano) y 10 km (Rural)
Inmunidad a interferencia	Muy alta	Muy alta	Baja
Autenticación y encriptación	No	Si	Si
Localización	Si	Si	No

Estandarización	SigFox y ETSI (Instituto Europeo de Normas de Telecomunicaciones)	Alianza LoRa	3GPP (3rd Generation Partnership Project)

1.3.1.4 BackEnd de SigFox

El BackEnd de SigFox es la plataforma en donde se almacena toda la información obtenida por los dispositivos finales o sensores, la misma que puede ser accedida mediante aplicaciones que se conecten al BackEnd de SigFox por dos métodos que permiten integrar las aplicaciones de usuarios con el BackEnd: la API de devolución de llamada (*callbacks*) y la API REST [8]. De manera general, las API de devolución de llamadas se utilizan para recuperar los mensajes de los dispositivos, y la API REST es utilizada para la administración y gestión de todos los dispositivos que están conectados [9].

Para una mejor gestión de los dispositivos finales dentro del BackEnd de SigFox, cada empresa o usuario que tiene asociado dispositivos finales al BackEnd está representada por un Grupo (*Group*), cada grupo contiene al menos un tipo de dispositivo (*Device type)*, el mismo que asocia a todos los dispositivos finales del mismo producto o proyecto, de esta manera les permite tener un comportamiento igual cuando la red SigFox recibe un mensajes de ellos [9]. En la Figura 1.7 se puede apreciar que la API de devolución de llamada (callbacks) se las configura para cada grupo de dispositivos y que la API REST se configura para todo el conjunto de dispositivos que tenga registrado la empresa.

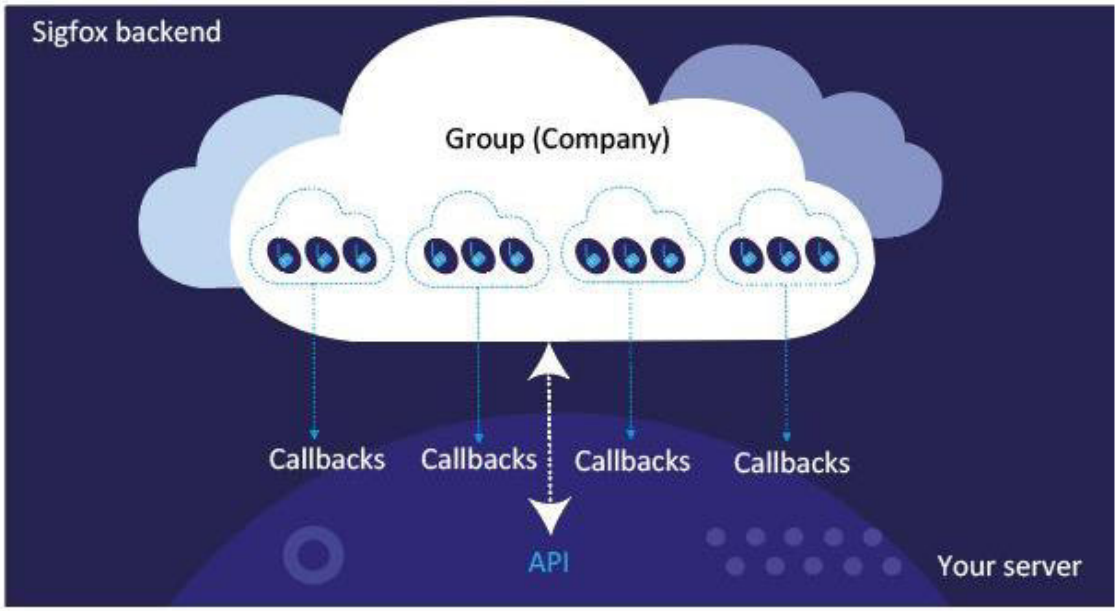

Figura 1.7 Diagrama BackEnd SigFox *[9]*.

1.3.1.5 Formas de acceso al BackEnd de SigFox

Para acceder a la información de nuestros dispositivos y a los mensajes enviados por los mismos existen dos métodos:

- API de devolución de llamada (*callbacks*):

Las *callbacks* requieren de solicitudes del protocolo HTTP (Protocolo de Transferencia de Hipertexto) que son mensajes unidireccionales que están asociadas a un tipo de dispositivo (*device type*), es decir las *callbacks* solo permiten obtener datos, no existe la posibilidad de administrar los dispositivos a través del BackEnd [9].

Cuando el BackEnd recibe un mensaje con información de un dispositivo final, se genera instantáneamente una *callback* que se envía a través del internet hacia un servidor configurado y de propiedad del dueño del dispositivo. De este modo no es necesario hacer una petición al BackEnd para recibir los mensajes de un dispositivo, sino que el BackEnd los envía automáticamente.

- API REST:

Las solicitudes de las consultas de la API REST de igual forma usan el protocolo HTTP, pero la diferencia es que son bidireccionales y funcionan a nivel de grupo (*Group*). El servidor de propiedad del dueño de los dispositivo solicita los datos al BackEnd a través de la API [9]. La API REST permite la gestión automática de todos los dispositivos asociados a una cuenta en el BackEnd.

Los *callbacks* son más sencillas de usar; pero el usar la API REST se tiene más funcionalidades y las siguientes ventajas:

- Gestionar una gran lista de dispositivos.
- Registro de nuevos dispositivos.
- Saber si hay cobertura de la red SigFox en una ubicación designada.
- Automatización de la gestión de *callbacks.*
- Recuperación de *callbacks* fallidos.
- Administrar a través de una plataforma propia todos los aspectos de negocio relacionado con SigFox.

1.3.1.6 Indicador de la calidad el enlace proporcionado por el BackEnd de SigFox.

El indicador de calidad de enlace (LQI) es una métrica que se usa para medir la calidad de la señal del enlace, SigFox ofrece un indicador de la calidad del enlace que se calcula usando lo siguiente [10]:

- Indicador de intensidad de la señal recibida (Received Signal Strength Indicator) que es la potencia recibida por las estaciones de radio SigFox y se mide en dBm.
- Número de estaciones que recibieron el mensaje del dispositivo final (Redundancia en la recepción).
- La zona RC (Radio Configuration) en donde se encuentra el dispositivo final.

En la Tabla 1.3 se muestra el indicador LQI en función del número de radio bases Sigfox y el indicado de intensidad de señal recibida (RSSI) para la zona RC4.

Tabla 1.3 Calculo de RSSI en la zona RC4 [10].

RSSI	Número de radio bases	LQI
-114dBm < RSSI	3	Excelente
-127dBm < RSSI < -114dBm	3	Bueno
-114dBm < RSSI	1 o 2	Bueno
-127dBm < RSSI < -114dBm	1 o 2	Promedio
RSSI < -127dBm	ninguna	Limitado

1.3.2. Nodo SigFox Pycom

SiPy es el nodo SigFox de la marca Pycom, es un triple portador integrando con Bluetooth, Wifi y SigFox. Incluye 12 meses de suscripción a SigFox para desarrollo.

Figura 1.8 Nodo Pycom SiPy [11].

Características principales [11]:

- Dimensión: 55mm x 20mm x 3.5mm
- Peso: 7g
- Voltaje de entrada: 3.3V y 5.5V
- Voltaje de salida: 3.3V
- Consumo para SigFox: 24mA en la recepción, 257 mA en transmisión y 0.5uA en reposo.
- Potencia de transmisión: 22dBm con un alcance superior a 50 km.

1.3.1.7 MicroPhyton

El lenguaje de programación usado para la programación del nodo es Python, específicamente MicroPython que es una implementación sencilla y eficiente del lenguaje de programación Python 3 [12], MicroPython está pensado para usarse en entornos restringidos como microcontroladores, incluye un conjunto pequeño de bibliotecas de Python. MicroPython es muy compatible con Python normal por lo que permite transferir código desde una PC a un microcontrolador[13].

MicroPython tiene los siguientes beneficios de Python:

- Es orientado a objetos.
- Tipo de datos y estructura de datos.
- La naturaleza altamente dinámica de los objetos de Python.
- Manejo de excepciones.
- Gran cantidad de funciones integradas de Python.

1.3.1.8 Configuración del nodo SiPy

Para la configuración del nodo es necesario conectar la placa de desarrollo a la computadora a través de USB, una vez hecho esto es necesario instalar un software para poder interactuar con el dispositivo, se debe instalar Pymakr que es un complemento para los editores de código para Visual Studio Code y Atom IDE [14]. A través de cualquiera de estos entornos es posible conectarnos con la placa y cargar los programas escritos en Python. El siguiente paso para la configuración del nodo es escribir un código en Python para ejecutar en nuestro nodo en la Figura 1.9 se muestra un código de ejemplo que prende el LED RGB de la placa.

```
>>> import pycom
>>> pycom.heartbeat(False)
>>> pycom.rgbled(0x330033)
```

Figura 1.9 Código ejemplo Python.

1.3.3. REST

Transferencia del estado representacional (Representacional State Transfer, REST) no es un estándar especifico, es un estilo arquitectónico que es utilizado en tecnologías y protocolos que implementen servicios web basados en REST [15]. El estilo arquitectónico REST está compuesto de solicitudes y respuestas, es una arquitectura cliente servidor en donde cada recurso u objeto está representado por una URI (Identificador Único del Recurso) [16].

REST permite el ahorro de recursos dentro del servidor ya que no guarda ningún estado del cliente cuando este hace una petición, REST es un protocolo sin estado.

Para representar la información se usa hipermedios como XML (Lenguaje de Marcado Extensible) o JSON (Notación de Objetos de JavaScript), que son usados para recibir información desde el servidor web como un arreglo (*array*)[17].

Para realizar peticiones al servidor el cliente realiza peticiones HTTP, usando especialmente los siguientes métodos:

- GET: Solicita la representación de un recurso.
- POST: Envía una entidad a un recurso en específico.
- PUT: Remplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.
- DELETE: Borra un recurso.

1.3.1.9 API REST

Las aplicaciones de usuario usan las API para poder comunicarse con los servidores WEB. Una API es un conjunto de funciones y datos que facilitan las interacciones entre aplicaciones para intercambiar información entre ellas [18]. En un servicio WEB una API es la cara del servicio, responde y escucha solicitudes de los clientes WEB como se muestra en la Figura 1.10.

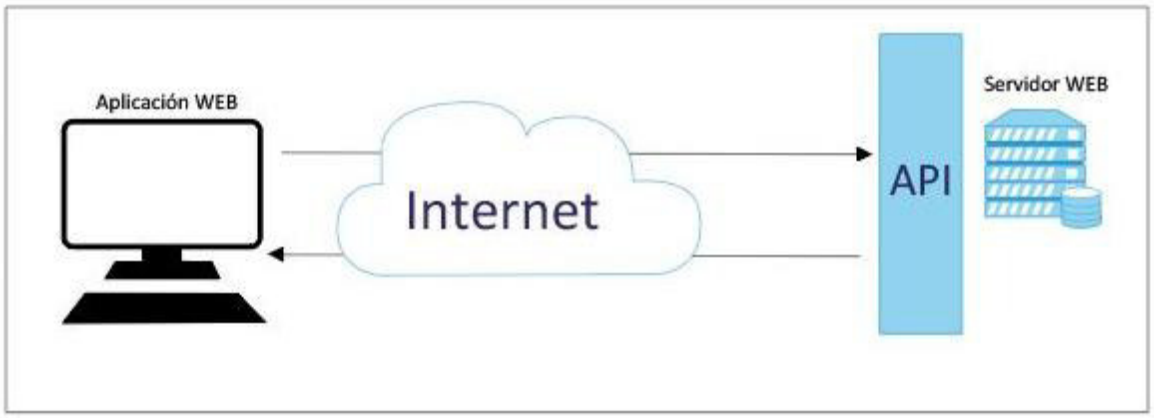

Figura 1.10 WEB API.

La API REST hace que un servicio web sea "RESTful". Una API REST consta de un conjunto de recursos interconectados. Este conjunto de recursos se conoce como modelo de recursos de la API REST. Las operaciones de una API REST se pueden llamar desde un cliente HTTP, incluido código del lenguaje JavaScript que se ejecuta en el navegador WEB de un cliente [19]. Todos los recursos de una API REST tiene una ruta base, la ruta base proporciona aislamiento entre diferentes API REST, así como aislamiento entre diferentes versiones de la misma API REST. Un ejemplo se puede exponer un proyecto a través de HTTP. La ruta base de la API REST para nuestro proyecto podría ser /proyecto/v1, y para la versión 2 de la misma API REST puede ser /proyecto/v2.

1.3.4. JavaScript

JavaScript es un lenguaje de programación ligero e interpretado [20], es uno de los lenguaje más famosos del mundo al ser lenguaje de los navegadores WEB, permite trabajar con él sin saber mucho sobre el lenguaje. JavaScript se encuentra diseñado en un paradigma basado en objetos, los objetos se pueden crear simplemente enlistando sus atributos [21]. JavaScript posee un sistema de objetos sin clases, en el que los objetos heredan propiedades directamente de otros objetos.

JavaScript corre en el lado del cliente de un sitio WEB, es usado para diseñar y programar el comportamiento de la página web cuando ocurre un evento [22].

ECMAScript es el estándar en el que JavaScript está basado, ECMAScript tiene ciclos de lanzamiento anuales de su estándar a partir de su sexta versión oficial ECMAScript 6, desde el 2012 todos los navegadores modernos son compatibles con ECMAScript 5.1 [20].

1.3.1.10 React.js

React es una librería de JavaScript para crear interfaces de usuario web, fue diseñada y mantenida por Facebook. React se basa en estos 3 aspectos:

- Declarativa: React facilita la creación de interfaces de usuario, permite diseñar vistas simples para cada estado de la aplicación, React se encarga de actualizar y renderizar de manera eficiente los componentes cuando se haga un cambio de estado en la aplicación [23].
- Basada en componentes: Permite crear componentes encapsulados que gestionan su propio estado y luego juntarlas para crear interfaces de usuario complejas [23].
- Aprenda una vez, escriba en cualquier lugar: Permite desarrollar nuevas funciones en React sin reescribir el código existente. React también puede renderizar en el servidor usando Node y potenciar aplicaciones móviles usando React Native [23].

React tiene dos características claves:

- JSX (short for JavaScript eXtension)

Es una extensión de React que hace más fácil modificar el DOM (Modelo de Objetos del Documento) mediante el uso de código muy parecido a HTML. El DOM se crea cuando los navegadores leen el código HTML para mostrar lo que está en el navegador, el DOM es un árbol representativo de cómo está organizada una página WEB[24].

JSX es compatible con cualquier plataforma de navegador, debido a que React.js se extiende a todos los navegadores modernos. El uso de JSX para actualizar el DOM genera mejoras significativas en el rendimiento del sitio WEB y eficiencia en el desarrollo.

- Virtual DOM

Si no se usa React JS (y JSX), un sitio web usará HTML para actualizar su DOM (el proceso que hace que las cosas "cambien" en la pantalla sin que el usuario tenga que actualizar manualmente una página). Esto funciona bien para sitios web simples y estáticos, pero para sitios web dinámicos que implican una gran interacción del usuario puede convertirse en un problema (ya que todo el DOM debe recargarse cada vez que el usuario hace clic en una función que solicita una actualización de página).

Se usa JSX para manipular y actualizar su DOM, React JS crea algo llamado DOM virtual. El DOM virtual (como su nombre lo indica) es una copia del DOM del sitio, y React JS usa esta copia para ver qué partes del DOM real deben cambiar cuando ocurre un evento (como cuando un usuario hace clic en un botón) [24].

1.3.1.11 Heatmap.js

Heatmap.js es una biblioteca de JavaScript liviana y fácil de usar para visualizar datos tridimensionales. Heatmap.js no es un script de seguimiento del comportamiento de un usuario, Heatmap.js crea mapas de calor dinámicos basados en los datos que recibe[25]. Es la biblioteca de visualización de mapas de calor más avanzada para aplicaciones WEB. Heatmap.js ofrece lo siguiente:

- Tiene un módulo de renderizado muy rápido.
- Maneja más de 40 mil puntos de datos.
- Es fácil de usar.
- Fácil de ampliar para funcionalidades personalizadas

2. METODOLOGÍA

En este capítulo se encuentra la información relacionada al diseño e implementación del prototipo para la obtención de la zona de cobertura de la red SigFox en el interior de edificaciones.

Para diseñar el prototipo es necesario obtener los requerimientos de la aplicación a ser desarrollada, la obtención de requerimientos se llevará a cabo mediante entrevistas, que serán analizadas para establecer los requerimientos funcionales y no funcionales del prototipo. El diseño de la aplicación servidor y la aplicación de usuario, utilizará una metodología orientada a objetos y como resultado del diseño se obtendrán: un diagrama de máquinas de estado, un diagrama de clases y de casos de uso; la base de datos será diseñada con un diagrama entidad relación. El diseño de la aplicación para la configuración del nodo sensor Pycom tendrá como resultado un diagrama de flujo.

La fase de implementación se compone de codificar la aplicación de usuario, la aplicación servidor, la aplicación de configuración de los nodos y la base de datos.

2.1. Diseño del prototipo.

Aquí se definirán los requerimientos para la realización del prototipo y se mostrará el diseño de las partes que lo componen. Se presentarán los diferentes diagramas necesarios para definir y abstraer los elementos del prototipo.

2.1.1. Entrevistas

Para obtener los requerimientos del prototipo se realizaron entrevistas mediante encuestas a profesores, estudiantes y profesionales que tengan algún conocimiento de la red SigFox. Debido a la pandemia del Covid-19, las entrevistas fueron realizadas mediante preguntas cerradas enviadas a los correos de los entrevistados. Los detalles de las entrevistas se encuentran en el Anexo A y en el Anexo B.

Las entrevistas tienen como objetivo obtener información referente al proceso de realizar mediciones de la zona de cobertura dentro de una edificación.

En base a la información entregada en las entrevistas se definieron los requerimientos que se muestran en la sección 2.1.2

2.1.2. Requerimientos del prototipo.

Para determinar los requerimientos que va a tener el prototipo se consideraron las encuestas realizadas. Los datos obtenidos permiten establecer los requerimientos funcionales que son las funcionalidades que brinda el prototipo y los requerimientos no funcionales que describen las características de funcionamiento del prototipo:

2.1.2.1 Requerimientos funcionales.

Los requerimientos funcionales se detallan a continuación:

- El usuario tiene la posibilidad de cargar un plano de una edificación antes de realizar las mediciones, ingresando un nombre para plano y la imagen del plano en la aplicación.
- El prototipo permite la configuración del nodo antes del envió de cada mensaje, el usuario debe ingresar la ubicación del nodo dentro de la edificación con coordenadas en pixeles de acuerdo con el plano ingresado y un identificador del diagrama.
- El usuario puede ver los planos de las edificaciones cargados en la base de datos editarlos y eliminarlos.
- El usuario puede observar los diagramas de cobertura creados, en base al plano cargado, el LQI y la posición enviada en cada mansaje SigFox.
- El prototipo permite observar los mensajes que se encuentran en el BackEnd de SigFox de cada plano en una tabla.

2.1.2.2 Requerimientos no funcionales.

Los requerimientos no funcionales se detallan a continuación:

- La aplicación prototipo debe estar conectado a internet para su funcionamiento.
- Los planos subidos a la aplicación deben estar en formato JPG o JPEG.
- Los mensajes enviados por los nodos se realizan aproximadamente cada 10 minutos, por lo que es necesario tener en cuenta esto al momento de realizar las mediciones.
- La información necesaria para realizar los diagramas de cobertura es almacenada en una base de datos relacional realizada en SQL.
- La aplicación servidor utiliza API REST desarrollada en C#.

- La aplicación de usuario utiliza las API REST del servidor desarrollado en C# y están realizada en JavaScript con las librerías REACT.JS y HEATMAP.JS.
- El nodo solo puede ser configurado con el lenguaje MicroPython.

2.1.3. Diseño del sistema prototipo

El esquema general del prototipo se muestra en la Figura 2.1, en el que se detallan las partes que lo conforman. Para un mejor esclarecimiento del diseño del prototipo se divide en diseño de hardware y diseño de software.

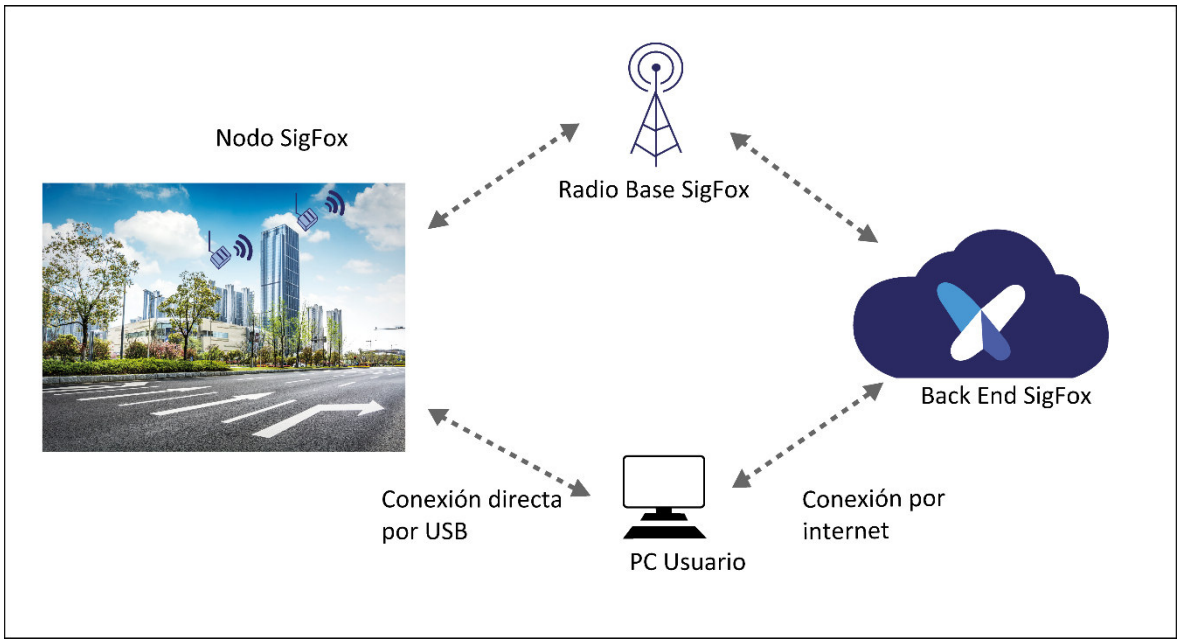

Figura 2.1 Esquema general del sistema prototipo.

2.1.3.1 Diseño del componente de Hardware.

La figura 2.2 muestra las partes que tiene el hardware que son usados para la implementación del prototipo.

Detalle de cada una de las partes:

- Nodo SigFox: Este componente se encarga de enviar los mensajes SigFox con el LQI correspondiente a la red SigFox.
- Radio bases SigFox: Este componente se encarga de recibir los mensajes SigFox del nodo SigFox por medio del espectro de radio frecuencia y envía el mensaje a la nube SigFox.

- Nube de SigFox: Conocido como BackEnd SigFox se encarga del almacenamiento, gestión y administración de los mensajes y nodos de la red, se encuentra dentro del internet. También se encarga de enviar mensajes al servidor.
- Servidor: Este componente se encarga de la conexión con la nube de SigFox para obtener los mensajes enviados por los nodos, además se encarga de la conexión con la base de datos y con la aplicación de usuario. Físicamente se encontrará en una computadora.
- Base de Datos: Es el componente encargado de almacenar toda la información necesaria de los mensajes del servidor de manera ordenada. Físicamente se encontrará en una computadora.
- Computador de usuario: Es el componente que contiene físicamente al software del prototipo, la base de datos y el servidor.

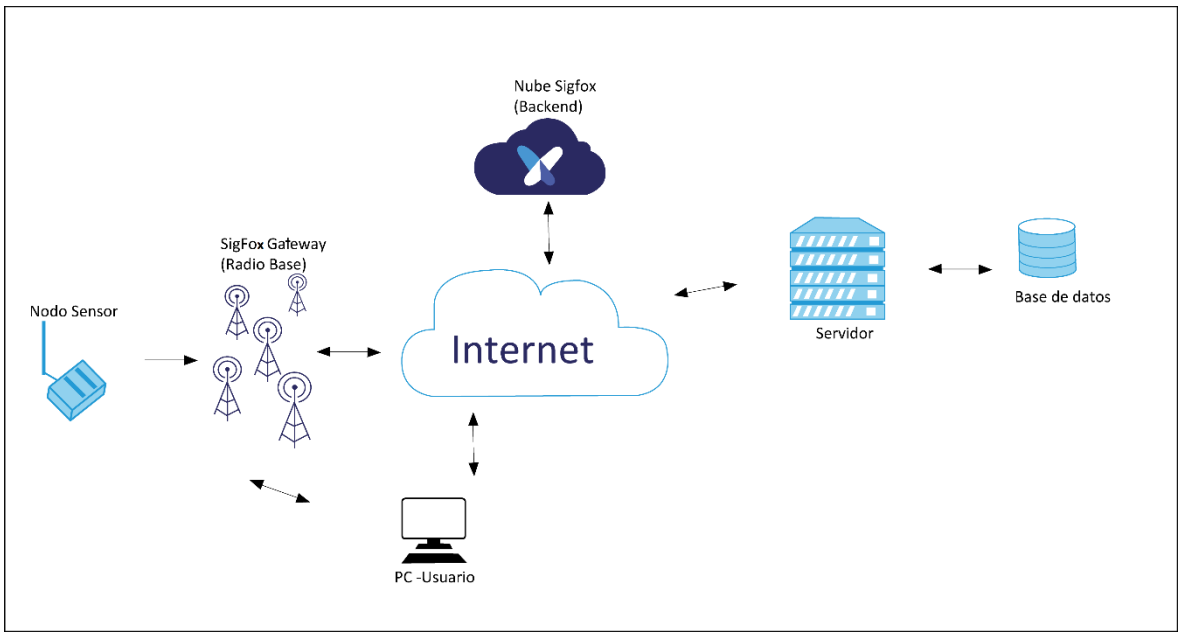

Figura 2.2 Componentes del prototipo de Hardware.

Dentro de los componentes del prototipo, las radio bases y la nube SigFox forman parte de la red SigFox que ya se encuentran implementadas por el operador y se hace uso de ellas.

Los demás componentes del prototipo serán implementados dentro de este trabajo de titulación. El servidor, la base de datos, la aplicación de usuario y la aplicación de configuración del nodo estarán dentro de una misma computadora. El nodo SigFox escogido para la realización de este trabajo de titulación es de la marca Pycom, en la sección 2.2.1 se muestra su implementación.

2.1.3.2 Diseño del componente de Software.

Para el diseño de software de los 4 subsistemas del prototipo es importante tomar en consideración el diagrama de la Figura 2.3, el cual contiene los elementos que actúan en el prototipo. El sistema general consta de 4 subsistemas que son:

- Subsistema Aplicación para configuración de los nodos: Este componente se encarga de crear un código que permita que los nodos sensores Pycom envíen mensajes a la red Sigfox. Los mensajes SigFox tendrán un formato específico.
- Subsistema Aplicación Servidor: Este componente se encarga de que el computador se conecte con el BackEnd SigFox, para recuperar los datos obtenidos del nodo sensor, esta conexión al BackEnd de SigFox se la realiza mediante el uso de la API REST.
- Subsistema Base de Datos: Este componente se encarga del almacenamiento de los mensajes correspondientes al subsistema aplicación de usuario.
- Subsistema Aplicación de usuario: Este componente se conecta a la aplicación servidor utilizando la API REST. Estas API REST son implementadas en la aplicación servidor y de esta manera poder acceder a la información de la base de datos, la aplicación hace uso de un plano de la edificación en donde se obtendrá los valores del LQI en zonas específicas.

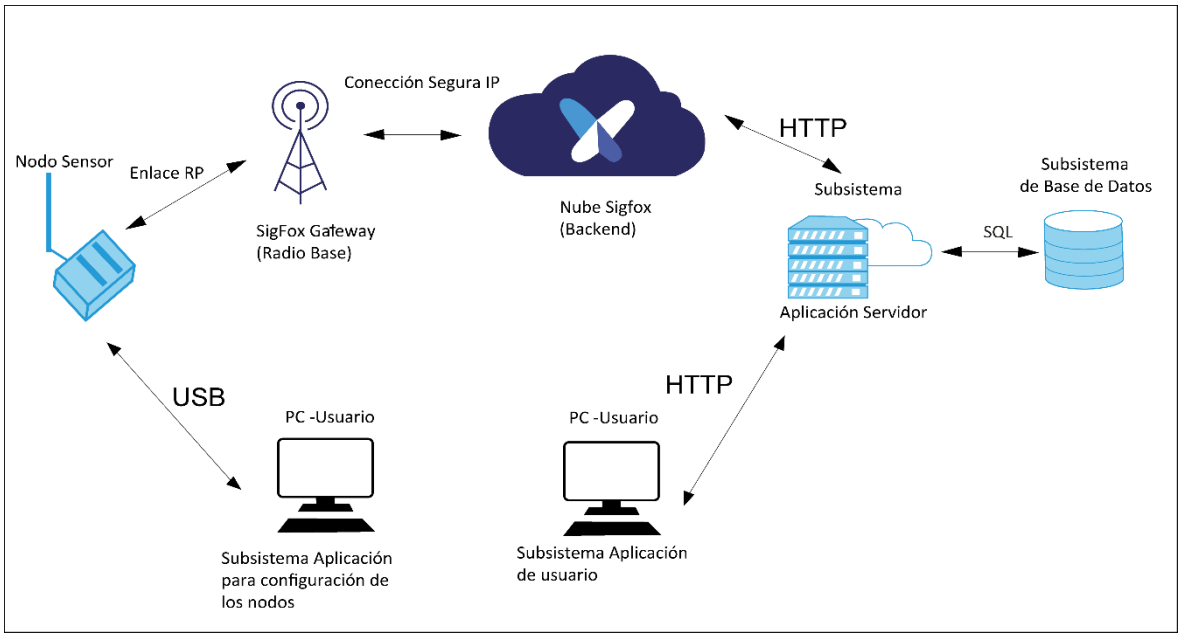

Figura 2.3 Subsistemas del software del prototipo.

Para el diseño de software de los 4 subsistemas y de los casos de uso del prototipo se hace uso de diferentes diagramas, en la Tabla 2.1 podemos apreciar los distintos diagramas para cada subsistema y el programa que se va a utilizar para su implementación.

Tabla 2.1. Subsistemas y diagramas.

Subsistema	Diagrama	Programa
Casos de uso del prototipo.	Diagrama de casos de uso.	-
Subsistema Base de datos.	Diagrama entidad – relación.	Microsoft SQL Server Management Studio 18.
Subsistema aplicación servidor.	Diagrama de clases y diseño del servicio API REST.	Entorno de Desarrollo Integrado (IDE) de Visual Studio
Subsistema aplicación de usuario.	Diagrama de máquinas de estado y diseño de la interfaz gráfica.	Visual Studio Code.
Subsistema aplicación para configuración de nodos.	Diagrama de flujo.	Atom y Pymakr.

2.1.3.2.1 Casos de uso del prototipo.

El sistema general está compuesto por todas las partes que se encuentran en la figura 2.3. y para el diseño de manera general se hace uso de un diagrama de casos de uso en base a los requerimientos funcionales del prototipo obtenidos de las encuestas realizadas.

- Diagrama de casos de uso:

En función de los requerimientos funcionales se realiza un diagrama de caso de uso del prototipo con el propósito de indicar las diferentes interacciones del usuario con el prototipo, en la Figura 2.4 se muestra el diagrama de uso del prototipo.

El diagrama de casos de uso nos muestra 5 casos en el que el usuario interactúa con el prototipo que son:

- Cargar plano de una edificación: Para cumplir con este caso de uso es necesario que el usuario realice las tareas de nombrar el plano e ingresar una imagen del plano.
- Configurar nodo antes de enviar mensaje: Para cumplir con este caso de uso es necesario que el usuario ingrese en el mensaje a enviar los datos relacionados a la ubicación dentro de la edificación y el identificador del plano de la edificación.

- Ver planos de las edificaciones cargados: Este es un caso de uso que se realiza en el subsistema de aplicación de usuario con la información de la base de datos que tiene como propósito la verificación de la carga del plano.
- Editar planos de las edificaciones cargados: Para cumplir con este caso de uso es necesario que usuario realice las tareas de actualizar el nombre del plano y la imagen del plano.
- Ver diagramas: Este es un caso de uso que se realiza en el subsistema de aplicación de usuario con la información de la base de datos que tiene como propósito mostrar el mapa de calor de la cobertura SigFox en el plano de la edificación por medio de los mensajes SigFox enviado por el nodo sensor Pycom. Tiene una función adicional de proporcionar información de los mensajes relacionados al plano de la edificación.

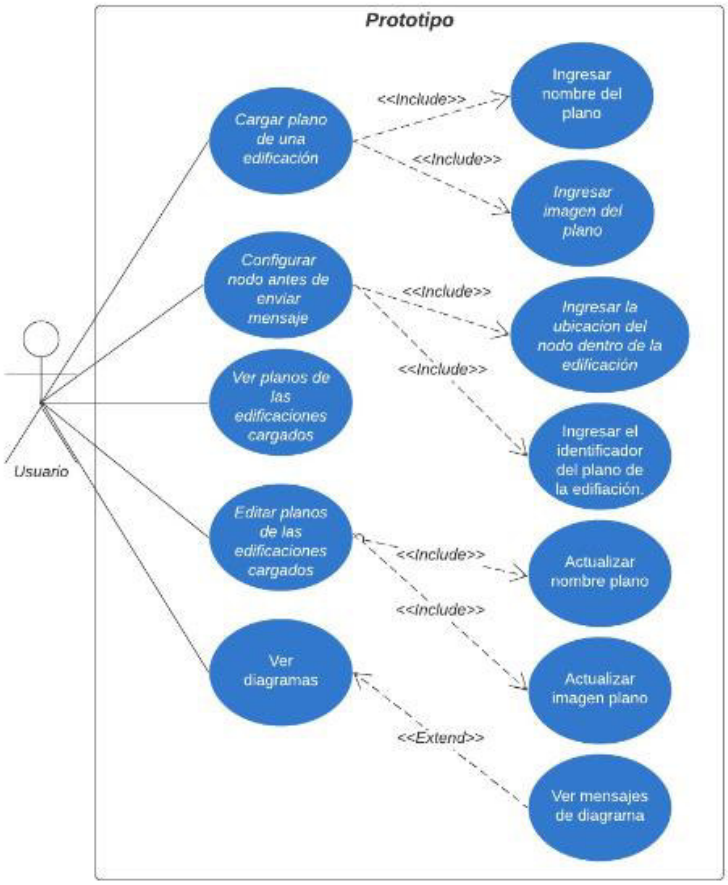

Figura 2.4 Diagrama de casos de uso del prototipo.

2.1.3.2.2 Subsistema Base de datos

- Diagrama de Entidad-Relación:

En base a los requerimientos establecidos para el prototipo se elabora el diagrama Entidad-Relación que se muestra en la Figura 2.5. Este diagrama es de gran utilidad para la creación de la base de datos que se utiliza para el almacenamiento de información de los nodos, mensajes enviados por los nodos y los planos de las edificaciones para los diagramas de cobertura.

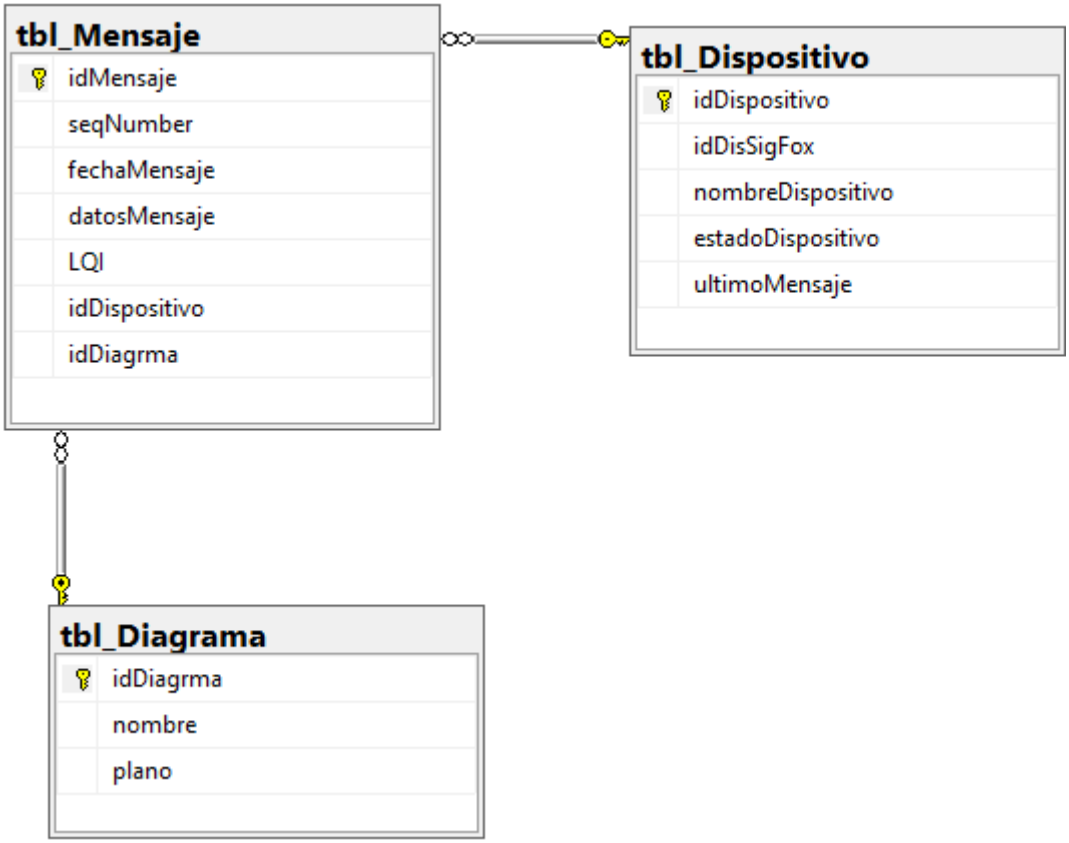

Figura 2.5 Diagrama Entidad-Relación.

Para poder cumplir con los requerimientos del prototipo es necesario contar con 3 tablas:

- tbl_Mensaje: La tabla Mensaje consta de datos importantes recibidos por el BackEnd de SigFox como el número de secuencia del mensaje SigFox enviado por el nodo sensor Pycom, el tiempo en que se recibió el mensaje en el BackEnd SigFox, el payload del mensaje SigFox enviado por el nodo sensor Pycom, el LQI asociado al mensaje SigFox enviado por el nodo sensor Pycom, el identificador del dispositivo SigFox que envía el mensaje propio de la base de datos y el identificador de diagrama obtenido del payload del mensaje SigFox.

- tbl_Dispositivo: La tabla Dispositivo consta de datos importantes del dispositivo que envía los mensajes a la red SigFox como el identificador único de dispositivo SigFox que consta de 32 bits, el nombre de dispositivo, el estado de dispositivo y el último mensaje enviado por el dispositivo.
- tbl_Diagrama: La tabla de Diagrama se encarga de almacenar los datos de los diagramas que es cargado por el usuario como el nombre del plano y el plano.

2.1.3.2.3 Subsistema aplicación Servidor:

- Diagrama de clases:

El diagrama de clases presentado en la Figura 2.6 representa las clases que sirven para la implementación de la aplicación servidor, la cual permitirá la obtención de los mensajes del BackEnd de SigFox para almacenarlos en la base de datos y posteriormente puedan ser manipulados por la aplicación de usuario a través de la API REST creada en la aplicación servidor. En el diagrama se visualizan los métodos y atributos de las clases utilizadas.

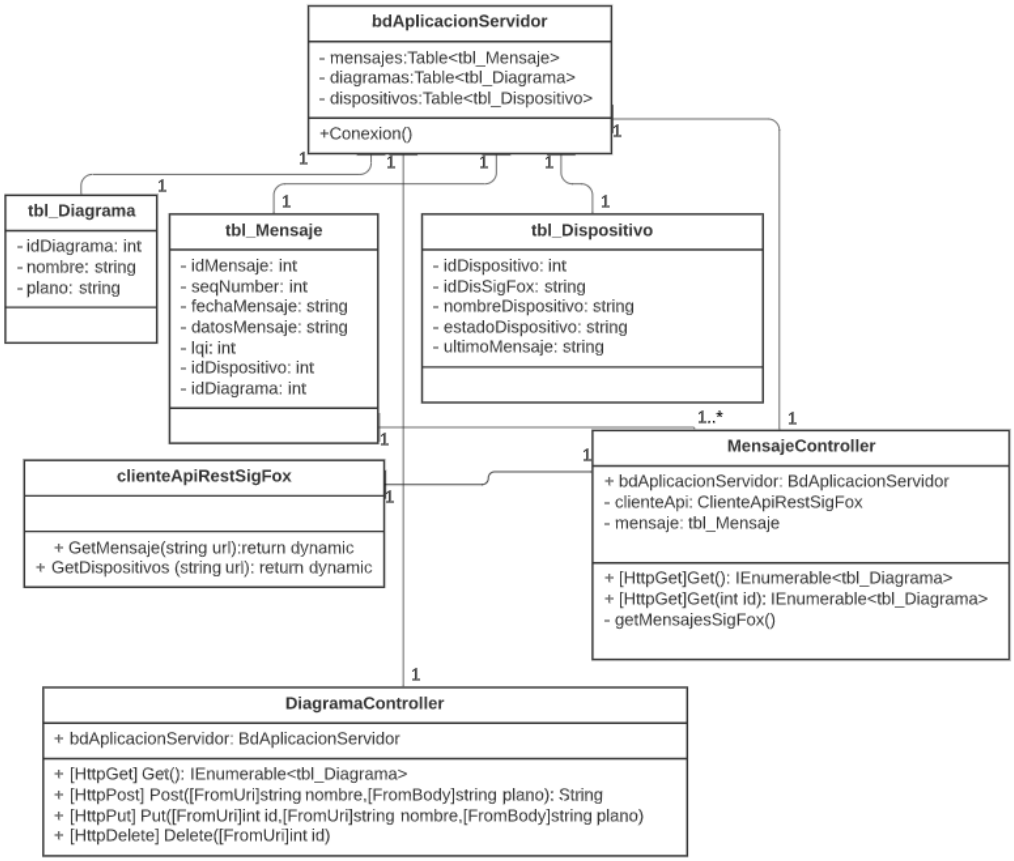

Figura 2.6 Diagrama de clases.

- Diseño del servicio API REST

En la Tabla 2.2 y la Tabla 2.3 se muestran el método HTTP (GET, PUT, POST y DELETE) y la URI (Identificador Uniforme de Recursos) de cada controlador de la API, las clases controladores que se diseñaron en el diagrama de clases son MensajeCotroller y DiagramaController. Estos permiten la manipulación y gestión de los mensajes SigFox de cada diagrama que están almacenados en la base de datos, además de los diagramas creados y existentes de igual manera en la base de datos.

Tabla 2.2 URI y métodos de Mensajes.

MensajeCotroller	Gestiona Mensajes SigFox	
Uso	Método	URI
Obtiene todos los mensajes SigFox	GET	api/Mensaje
Obtiene los mensajes de acuerdo con un ID de diagrama	GET	api/Mensaje/{id}

Tabla 2.3 URI y métodos de Diagrama.

DiagramaController	Gestiona Diagramas	
Uso	Método	URI
Obtiene todos los diagramas	GET	api/Diagrama
Crea un diagrama nuevo	POST	api/Diagrama?nombre={nombre}
Actualizar datos de un diagrama	PUT	api/Diagrama/{id}?nombre={nombre}
Elimina un diagrama seleccionado	DELETE	api/Diagrama/{id}

2.1.3.2.4 Subsistema aplicación de usuario.

- Diagrama de máquinas de estado:

El diagrama de máquinas de estado muestra los principales estados que tiene el subsistema aplicación de usuario, las actividades principales de la aplicación y son:

- CRUD plano: Compuesta de funciones para poder crear, ver, actualizar y eliminar planos de las edificaciones.
- Ver Diagramas: Esta compuesta de funciones que permiten la creación y visualización de los diagramas de cobertura, que se crean con el plano de la

edificación y con el LQI de los mensajes de la red SigFox correspondientes al plano de la edificación.

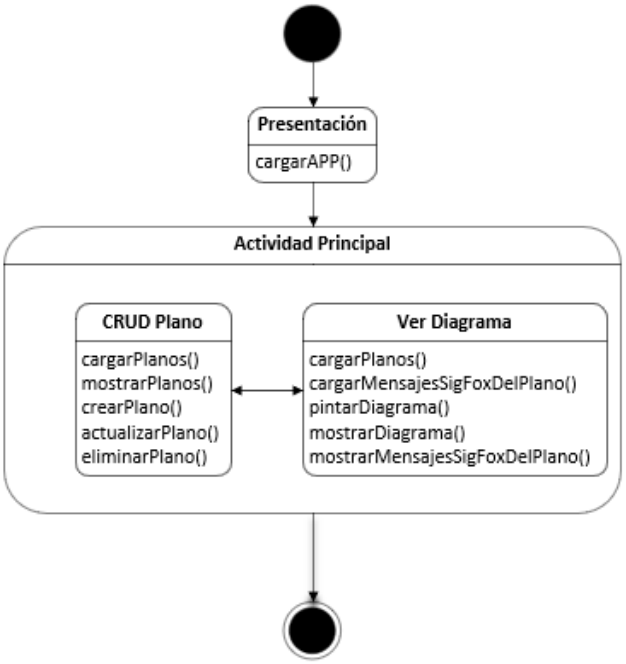

Figura 2.7 Diagrama de Máquina de Estados.

- Diseño de la Interfaz de Usuario:

A continuación, se muestra el bosquejo de la interfaz que servirán como base para el diseño de la interfaz gráfica de la aplicación de usuario que se desarrollara en React.js.

Para la página de inicio el bosquejo que se muestra permite mostrar un mensaje de bienvenida a la aplicación y tener un menú ubicado en la parte superior de la pantalla que facilitara navegar en todas las páginas de la aplicación de usuario, como se puede observar en la Figura 2.8.

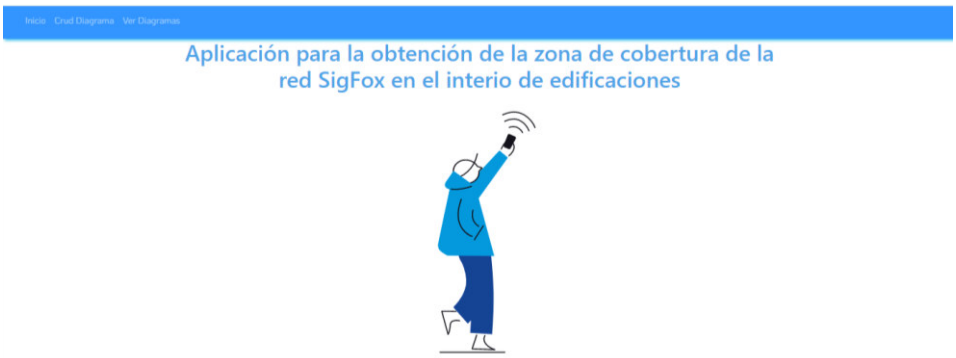

Figura 2.8 Bosquejo página de inicio.

El bosquejo de la interfaz correspondiente a la página que permite ver, crear, editar y eliminar los planos de los diagramas en la aplicación, se muestra en la Figura 2.9. En esta los usuarios que deseen agregar un plano deberán ingresar en un formulario el nombre del diagrama y el archivo correspondiente al plano de la edificación, también tiene la opción de visualizar los planos existentes con su nombre, editarlos o eliminarlos.

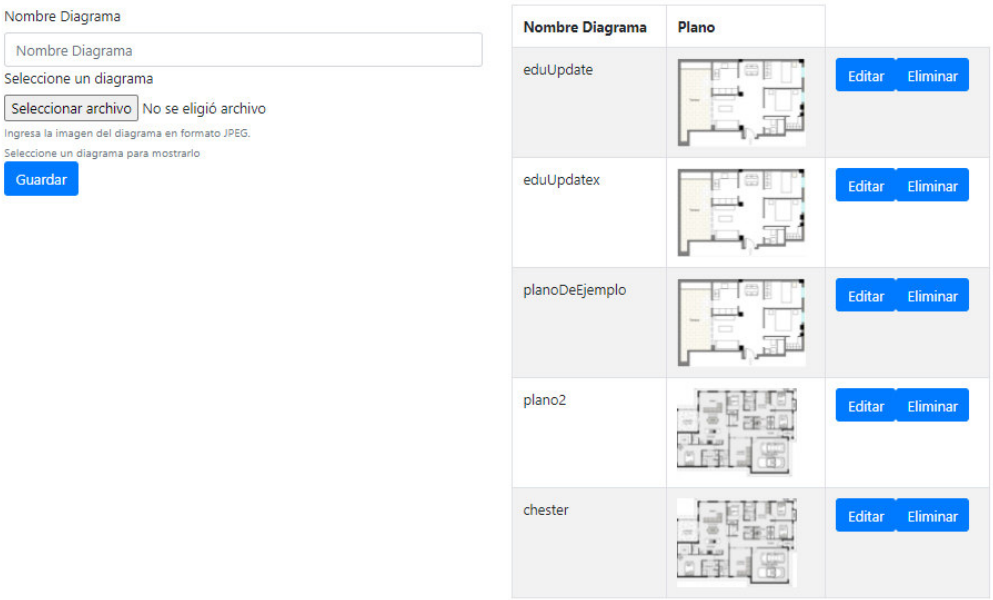

Figura 2.9 Bosquejo para ver, crear, editar y eliminar diagramas.

Para facilitar la observación de los diagramas de cobertura obtenidos por la aplicación primero se debe seleccionar un diagrama, como se puede apreciar en la Figura 2.10.

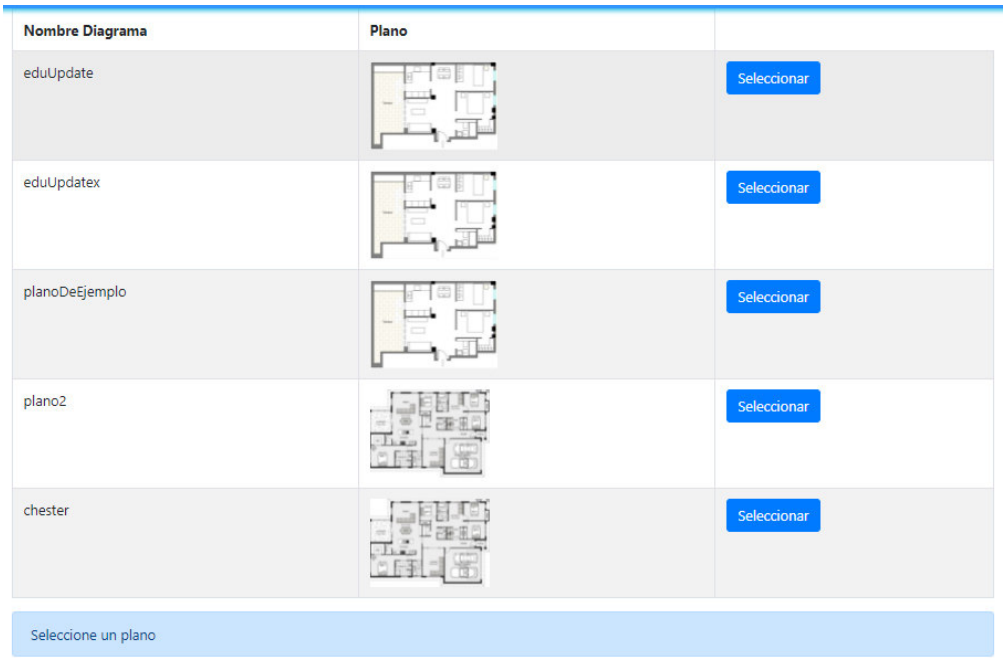

Figura 2.10 Bosquejo selección de diagrama.

Cuando el usuario seleccione un diagrama se muestra el mismo graficado, sus puntos y a continuación los mensajes de SigFox correspondientes a dicho diagrama, como se puede ver en la Figura 2.11.

Numero de secuencia	Fecha	Datos	LQI
100	8/4/2021 16:4:2	115/300/150	1
100	8/4/2021 16:4:2	115/450/150	2
100	8/4/2021 16:4:2	115/450/350	3

Figura 2.11 Bosquejo para visualización del diagrama con la zona de cobertura y los mensajes de SigFox del diagrama.

2.1.3.2.5 Subsistema para configuración del nodo.

- Diagrama de flujo:

A continuación, en base a los requerimientos funcionales de la sección 2.1.2.1 se realiza el diagrama de flujo de la configuración del nodo sensor Pycom, que se utiliza para el desarrollo de la aplicación en el lenguaje de programación MicroPython. El diagrama de flujo del subsistema para configuración del nodo se lo puede apreciar en la Figura 2.12.

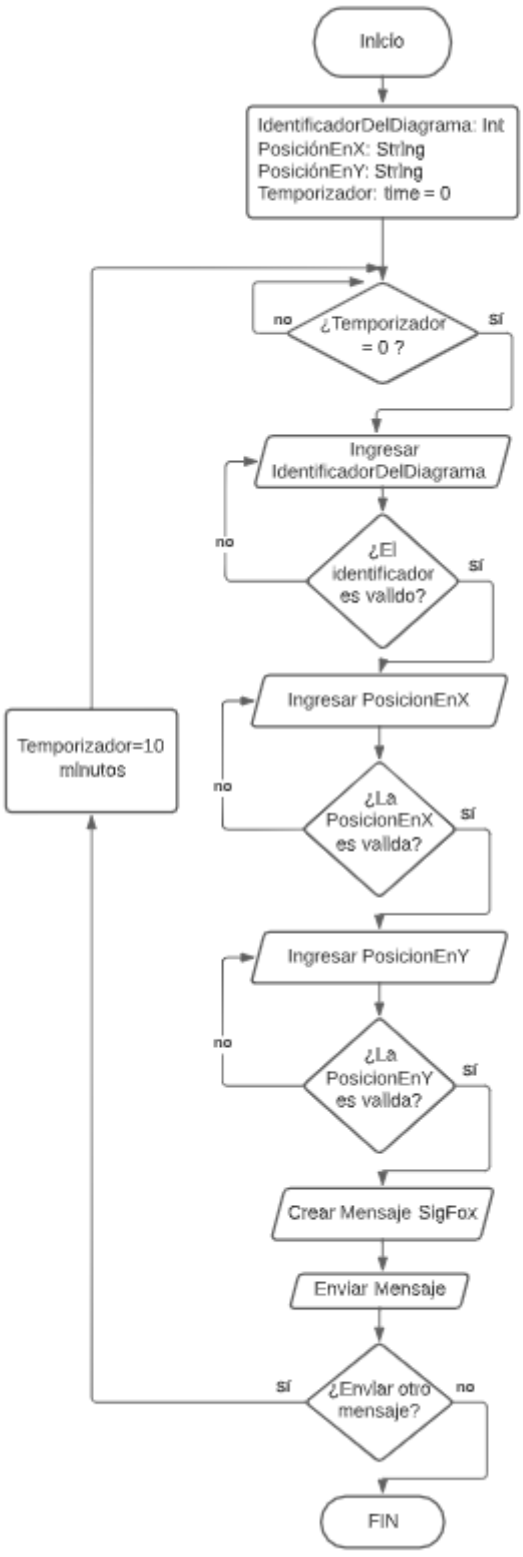

Figura 2.12 Diagrama de flujo de la aplicación de configuración de los nodos Pycom.

2.2. Implementación

2.2.1. Implementación del Hardware

2.2.1.1 Nodo SigFox

Para la implementación del prototipo se escogieron los componentes idóneos para la comunicación de los dispositivos con la red SigFox. Los elementos usados en la implementación son:

1. SiPy (de la marca Pycom para la red SigFox): El SiPy es una placa de desarrollo habilitada para SigFox y se programa con MicroPython.
2. Antena SigFox: Adecuada para la conexión a la red en la banda de 922 Mhz.
3. Batería de litio recargable: La batería de litio es usada para la alimentación del SiPy, la cual es de 3.3 V y 500mAh para su correcto funcionamiento.

En la Figura 2.13 se aprecian los elementos del Nodo SigFox.

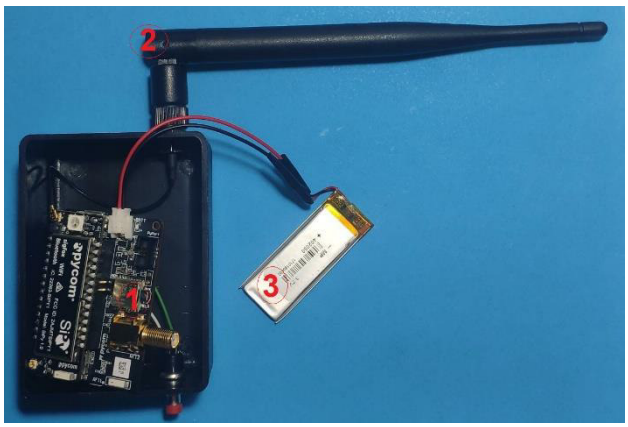

Figura 2.13 Componentes del dispositivo.

Para hacer uso del nodo Pycom es necesario registrar el dispositivo en el BackEnd de SigFox. Para eso se debe crear una cuenta en el siguiente enlace https://build.sigfox.com/sign-up. Posteriormente con la cuenta creada se accede a la página del BackEnd https://backedn.sigfox.com como se observar en la figura 2.14.

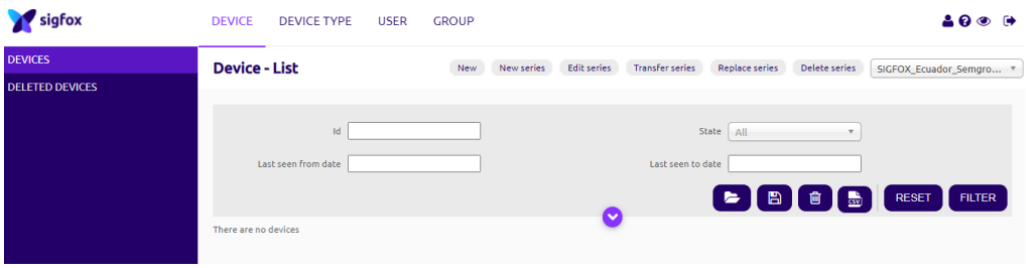

Figura 2.14. Portal del BackEnd SigFox.

En el portal se accede a la pestaña "*DEVICE*" y en new se completa la información del dispositivo que se ve en la Figura 2.15. Toda la información que se pide en el formulario es incluida con el nodo.

Figura 2.15 Formulario para agregar dispositivo al BackEnd SigFox.

2.2.1.2 Radio Bases SigFox

Las radio bases ya están implementadas por el operador de la red SigFox en el país y el nodo SigFox ya viene configurado para uso inmediato.

2.2.1.3 Nube SigFox

La nube ya está implementada por SigFox, se encuentra en el internet y para hacer uso de esta se debe tener una cuenta activa de prueba para desarrollo, que es gratuita por un año y se puede acceder a todas las herramientas dentro de la nube SigFox.

2.2.1.4 Servidor, Base de datos, aplicación de usuario y aplicación de configuración de los nodos

Los 4 subsistemas correspondientes al aplicativo se los implementa en un solo computador, razón por la cual el computador debe contener las siguientes características mínimas:

- Memoria RAM de 8GB.
- Procesador Intel 10ª generación i3 o Ryzen 3.
- Disco duro de 500 GB.
- Entrada USB 2.0 o 3.0.
- Sistema Operativo Windows 10.

2.2.2. Implementación de Software

2.3.2.1 Implementación subsistema base de datos.

Para la implementación de la base de datos se utilizó SQL de Microsoft SQL Server Management Studio 18 y se implementó en base al diseño realizado en la sección 2.1.3.2. En el código 2.1. se muestra la creación de la base de datos llamada BD_AplicacionServidor y el código para el uso de la base de datos.

```
create database BD_AplicacionServidor
go
use BD_AplicacionServidor
```

Código 2.1 Creación base de datos.

En el código 2.2 se muestra la creación de las tablas: mensaje, dispositivo, diagrama y diagramaMensaje, que están definidas en el diagrama entidad relación.

```
create table tbl_Dispositivo(
    idDispositivo int identity primary key,
    idDisSigFox varchar(20),
    nombreDispositivo varchar(50),
    estadoDispositivo varchar(50),
    ultimoMensaje varchar(50),
);
go
create table tbl_Mensaje(
    idMensaje int identity primary key,
    seqNumber int,
    fechaMensaje varchar(50),
    datosMensaje varchar(50),
    LQI int,
    idDispositivo int foreign key references tbl_Dispositivo(idDispositivo)
);
go
create table tbl_Diagrama(
    idDiagrma int identity primary key,
    nombre varchar(20),
    plano varchar(max)
);
go
create table tbl_MensajeDiagrama(
    idMensajeDiagrama int identity primary key,
    idMensaje int foreign key references tbl_Mensaje(idMensaje),
    idDiagrma int foreign key references tbl_Diagrama(idDiagrma)
)
```

Código 2.2 Creación tablas base de datos.

2.3.2.2 Implementación de la aplicación servidor

La aplicación servidor fue implementada en el Entorno de Desarrollo Integrado (IDE) de Visual Studio ajustado con el diseño de la sección 2.1.3.2.3. La tecnología utilizada para

esto fue ASP.NET del lenguaje de programación C#. La aplicación sirve para la conexión con el BackEnd de SigFox y de la creación del servicio API REST para que la aplicación de usuario pueda consumir la información almacenada en la base de datos. Las consultas de la aplicación servidor con la base de datos utilizan LINQ (*Language Integrated Query*) nombre que corresponde a un conjunto de tecnologías basadas en la integración de capacidades de consulta a base de datos directamente en el lenguaje C#.

Para la conexión con el BackEnd SigFox que permite la obtención de los mensajes de los nodos y los dispositivos (nodos sensores) se creó la clase ClienteApiRESTSigFox, la que mediante el método GET del protocolo HTTP permite obtener los mensajes y la información de los dispositivos, en el código 2.3 se puede observar el método para obtener los mensajes SigFox.

```
1 referencia
public dynamic GetMensajes(string url)
{
    HttpWebRequest myWebRequest = (HttpWebRequest)WebRequest.Create(url);
    myWebRequest.UserAgent = "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:23.0) Gecko/20100101 Firefox/23.0";
    myWebRequest.Credentials = new NetworkCredential("5f357d4725643258bc5e86e4", "abae22e78428f22a9b094e68e0c29f82");
    myWebRequest.Proxy = null;
    HttpWebResponse myHttpWebResponse = (HttpWebResponse)myWebRequest.GetResponse();
    Stream myStream = myHttpWebResponse.GetResponseStream();
    StreamReader myStreamReader = new StreamReader(myStream);
    string Datos = HttpUtility.HtmlDecode(myStreamReader.ReadToEnd());
    var data = JsonConvert.DeserializeObject<ResultadoMensaje>(Datos);
    return data;
}
```

Código 2.3 Método para obtención de los mensajes SigFox.

La conexión con la base de datos que permite guardar los mensajes y dispositivos obtenidos del BackEnd de SigFox usa LINQ con un *string* de conexión que posibilita conectarse a la base de datos, para su uso se creó la clase bdAplicacionServidor como se observar en el código 2.4.

```
5 referencias
public class BdAplicacionServidor : DataContext
{
    2 referencias
    public BdAplicacionServidor() : base(@"Data Source=DESKTOP-V6SBFOG\SQLEXPRESS;Initial Catalog=BD_AplicacionServidor;Integrated Security=True") { }
    public Table<tbl_Mensaje> mensajes;
    public Table<tbl_Diagrama> diagramas;
    public Table<tblDispositivo> dispositivos;
}
```

Código 2.4 Clase BdAplicacionServidor.

Además, se necesita referenciar las tablas creadas en la base de datos con SQL, por lo que es necesario la creación de las clases: tbl_Mensaje, tbl_Diagrama, y tbl_Dispositivo con los mismos atributos que tienen las tablas en SQL, en el código 2.5 se puede ver un ejemplo de la clase tbl_Mensaje.

```
public class tbl_Mensaje
{
    [Column(IsPrimaryKey = true, AutoSync = AutoSync.OnInsert, IsDbGenerated = true)]
    public int idMensaje;
    [Column]
    public int seqNumber;
    [Column]
    public string fechaMensaje;
    [Column]
    public string datosMensaje;
    [Column]
    public int lqi;
    [Column]
    public int idDispositivo;
    [Column]
    public int idDiagrma;
    0 referencias
    public tbl_Mensaje()
    {
    }

    1 referencia
    public tbl_Mensaje(int seqNumber, string fechaMensaje, string datosMensaje, int LQI, int idDispositivo, int idDiagrama)
    {
        this.seqNumber = seqNumber;
        this.fechaMensaje = fechaMensaje;
        this.datosMensaje = datosMensaje;
        this.lqi = LQI;
        this.idDispositivo = idDispositivo;
        this.idDiagrma = idDiagrama;
    }
}
```

Código 2.5 Clase tbl_Mensaje.

La implementación del servicio API REST para que la aplicación de usuario pueda obtener mediante métodos HTTP la información necesaria que permite la realización de los diagramas de cobertura, utilizo las clases controladoras: DiagramaController y MensajeController. Estas clases posibilitan controlar el comportamiento que tendrá cada método del protocolo HTTP, en el código 2.6 se puede observar el comportamiento que tendrá el método GET de DiagramaController. Este método permite obtener los mensajes SigFox de un determinado diagrama y para ello se envía un atributo que contenga el id del diagrama a consultar, primeramente el método llama a la función getMensajesSigFox() que obtiene los mensajes del BackEnd SigFox y actualiza la base de datos en caso de existir un mensaje nuevo, posteriormente hace una consulta a la base de datos para obtener los mensajes que le corresponda al id de diagrama ingresado y finalmente devuelve la respuesta que tendrá esta consulta.

```
[HttpGet]
0 referencias
public IEnumerable<tbl_Mensaje> Get(int id)
{
    getMensajesSigFox();

    var mensajesDiagrama = bdAplicacionServidor.mensajes.ToList().Where(x => x.idDiagrma == id);

    return mensajesDiagrama;
}
```

Código 2.6 Método GET de la clase MensajeController.

En el código 2.7 se muestra el método HTTP Post de la clase DiagramaController que permite la creación de una tabla diagrama y guardarla en la base de datos, para ello recibe como datos el nombre del diagrama a crear que viene dentro de la URL y un *string* que

guarda el valor de la imagen del plano del lugar donde se va a realizar las mediciones en formato base64. El método crea una nueva instancia de la clase tbl_Diagrama y a continuación a través de consultas LINQ a la base de datos permite guardar la instancia creada.

```
[HttpPost()]
0 referencias
public String Post( [FromUri]string nombre, [FromBody]  string plano)
{
    tbl_Diagrama diagrama = new tbl_Diagrama(nombre, plano);


    bdAplicacionServidor.diagramas.InsertOnSubmit(diagrama);
    bdAplicacionServidor.SubmitChanges();
    return plano;
}
```

Código 2.7 Método POST de la clase DiagramaController.

El método HTTP Put de la clase DiagramaController de la aplicación servidor que usa la aplicación de usuario para actualizar los valores de los diagramas se muestra en el código 2.8. El método recibe como datos el id del diagrama a actualizar y el nombre de diagrama que va a actualizar dentro de la URL, además recibe un string de la imagen del plano a actualizar en formato base64, el string es enviado dentro del cuerpo de la petición. El método hace una consulta a la base de datos con LINQ usando el id de diagrama ingresado y actualiza los valores.

```
[HttpPut()]
0 referencias
public void Put([FromUri] int id, [FromUri] string nombre,[FromBody]string plano)
{
    var diagrama = (from i in bdAplicacionServidor.diagramas
                    where i.idDiagrma == id
                    select i).FirstOrDefault();

    diagrama.nombre = nombre;
    diagrama.plano = plano;
    bdAplicacionServidor.SubmitChanges();
}
```

Código 2.8 Método PUT de la clase DiagramaController.

2.3.2.3 Implementación aplicación de usuario

La aplicación de usuario fue implementada en el editor de código Visual Studio Code de acuerdo con el diseño de la sección 2.1.3.2.4. En el desarrollo se utilizó el lenguaje de programación JavaScript con las librerías Heatmap.js y React.js. La aplicación tiene la

función mediante consultas HTTP conectarse a la aplicación servidor y de esta manera poder maquetar una interfaz gráfica para el usuario que sea intuitiva y fácil de usar.

Para poder realizar las peticiones HTTP se utilizó la API Fetch que proporciona una interfaz a JavaScript que permite acceder y manipular peticiones HTTP, en el código 2.9 se muestra la implementación de la función que posibilita su uso.

```
return fetch(endpoint, options)
  .then((res) =>
    res.ok
      ? res.json()
      : Promise.reject({
          err: true,
          status: res.status || "00",
          statusText: res.statusText || "Ocurrió un error",
        })
  )
  .catch((err) => err);
```

Código 2.9 Implementación de la API Fetch.

Para la interfaz del CRUD diagrama se creó un componente de React llamado crudDiagrama, el que permite ver, crear, editar y eliminar los diagramas cargados en la base de datos. Para esto primeramente se hace una petición GET a la aplicación servidor y se obtienen los diagramas como se muestra en el código 2.10, la petición se hace al arrancar el programa y permite visualizar en la interfaz gráfica los diagramas. Para hacer la petición se envía la URL de la aplicación servidor, si no existe error en la petición se actualiza la variable diagramas a la respuesta de la petición y si existe error se muestra un mensaje de error.

```
helpHttp().get(url).then((res) => {
    if(!res.err){
        setDiagramas(res);
        setError(null);
    }else{
        setDiagramas(null);
        setError(res);
    }
    setLoading(false);
});
```

Código 2.10 Petición GET para obtener los diagramas.

El componente crudDiagrama además tiene los métodos que permiten crear un nuevo diagrama, editarlo o eliminarlo. En el código 2.11 se muestra como ejemplo el código de la función createData que permite la creación de un nuevo diagrama. La función recibe como atributo los valores del formulario. Primeramente, los datos a enviar se transforman a formato JSON, después se declara la URL de la aplicación servidor que crea un nuevo

diagrama, posteriormente se configura las opciones de la petición Fetch y si no hay errores se realiza la petición POST a la aplicación servidor con los valores para crear un nuevo diagrama.

```
const createData = (form) =>{
    let data = JSON.stringify(form.plano);
    console.log(data);
    let endpoint = `${url}?nombre= ${form.nombre}`;
    let options ={
        body: data.replace(/['"]+/g, ''),
        headers: {'content-type': 'text/json'}
    };
    console.log(options);
    if(options.body === null){
    }else {
        api.post(endpoint,options).then(res=>{
            console.log(res);
            if(!res.err){
                console.log(res);
            }else{
                setError(res);
            }
        });
    }
};
```

Código 2.11 Función createData.

Para mostrar los mensajes de un diagrama se creó el componente DiagramaMensaje que permite mostrar en la interfaz gráfica los mensajes de cada diagrama, esto se hace con una petición Get a la aplicación servidor que posibilita obtener los mensajes de un diagrama previamente seleccionado como se muestra en el código 2.12. El método implementado permite obtener los mensajes de la aplicación servidor. Primeramente, declara la URL para hacer la consulta GET la cual debe contener el id del diagrama seleccionado, posteriormente se realiza la consulta, si la petición no tiene errores se actualiza el valor de la variable mensajes para que sean mostrados en la interfaz de usuario.

```
useEffect(() => {
    if(diagramaSelected){
        let endpoint = `${urlM}/${diagramaSelected.idDiagrma}`;
            helpHttp().get(endpoint).then((res) => {
            if(!res.err){
                setError(null);
                setMensajes(res);
            }else{
                setError(res);
                setMensajes(null);
            }
        });
    }else{
        console.log("No se ha selecinado ningun diagrama");
    }
    console.log(mensajes);
}, [diagramaSelected]);
```

Código 2.12 Método Get de mensajes.

Después de seleccionar el plano de la edificación y obtener los mensajes de la aplicación servidor, los mensajes obtenidos se utilizan para pintar los diagramas de acuerdo con la información que contenga cada mensaje, el código 2.13 muestra el método que permite crear los puntos y se grafiquen de acuerdo con el LQI obtenido de cada mensaje.

```
mensajes.map((el) => {
var cadena = el.datosMensaje.split("/");
let x = parseInt(cadena[1]);
let y = parseInt(cadena[2]);
console.log(cadena[1]);
  if(el.lqi === 1){
    for( let i=0; i < 1; i++ ){
      for( let j=0 ; j < 1; j++){
        let data = {
          coordinates: [
            x + i,
            y+j,
          ],
        }
        dataPoints.push(data);
      }
    }
  } else if(el.lqi === 2){
      for( let i=0; i < 7; i++ ){
        for( let j=0 ; j < 7; j++){
          let data = {
            coordinates: [
              x - i,
              y - j,
            ],
          }
          dataPoints.push(data);
        }
      }
  }else{
    for( let i=0; i < 20; i++ ){
      for( let j=0 ; j < 20; j++){
        let data = {
          coordinates: [
            x + i,
            y+j,
          ],
        }
        dataPoints.push(data);
      }
    }
  }
```

Código 2.13 Método para la creación de los puntos.

Posteriormente de la creación de los puntos la librería HEATMAP.js procede a graficar el diagrama como se muestra en el código 2.14, para esto hace uso del plano seleccionado y de los mensajes que le corresponden.

```
<Map
  className=""
  crs={L.CRS.Simple}
  bounds={bounds}
  maxZoom={0}
  // scrollWheelZoom={false}
  onzoomstart={(e) => console.log(e)}
  attributionControl={false}
  scrollWheelZoom = {false}
>
  <HeatmapLayer
    points={dataPoints}
    longitudeExtractor={(m) => m.coordinates[0]}
    latitudeExtractor={(m) => m.coordinates[1]}
    intensityExtractor={(m) => 1}
    radius={20}
    weight={1}

  />
  <ImageOverlay
    url={diagramaSelected.plano.replace(/['"]+/g, '')}
    bounds={[
      [400, 0],
      [0, 800],
    ]}
  />
</Map>
```

Código 2.14 Maquetación del diagrama de cobertura.

2.3.2.4 Implementación del subsistema para la configuración de los nodos.

La aplicación para la configuración de los nodos fue implementada en el editor de código Visual Studio Code de acuerdo al diseño realizado en la sección 2.1.3.2.5. En el desarrollo se utilizó el lenguaje de programación Python. La aplicación permite la configuración del nodo para el envío de mensajes a través de la red SigFox, para ello pide al usuario el ingreso de los valores establecidos en el diseño, en el código 2.15 se observa la petición al usuario del ingreso de los valores, los mismo que son almacenados en una variable antes de ser enviados por el nodo.

```
print('Ingrese el ID del diagrama')
IdDiagrama=str(lee_entero())
print('Ingrese la posición en X')
PosX=str(lee_entero())
print('Ingrese la posición en Y')
PosY=str(lee_entero())
temp=0
mensaje=IdDiagrama+'/'+PosX+'/'+PosY
```

Código 2.15 Petición de valores usuario subsistema configuración de los nodos.

Una vez que la variable con los datos es creada para que el nodo envié el mensaje a través de la red SigFox se realiza lo siguiente: (como se puede observar en el código 2.16)

- Se importa la librería de SigFox y socket para poder realizar el envío de los mensajes.
- Se crea un *socket* de SigFox.
- Se bloquea el *socket*.
- Se configura el socket para el envío de mensajes por la red SigFox en sentido de subida (Up Link).
- Se envía el mensaje SigFox, se procede a cambiar el color del led del nodo a blanco y se activa un temporizador de 10 minutos para poder enviar un nuevo mensaje.

```
from network import Sigfox
import socket
# init Sigfox for RCZ4 (ECUADOR)
sigfox = Sigfox(mode=Sigfox.SIGFOX, rcz=Sigfox.RCZ4)
# create a Sigfox socket
s = socket.socket(socket.AF_SIGFOX, socket.SOCK_RAW)
# make the socket blocking
s.setblocking(True)
# configure it as uplink only
s.setsockopt(socket.SOL_SIGFOX, socket.SO_RX, False)
# send some bytes
s.send(mensaje)
pycom.rgbled(0x101010)  # White
print('Mensaje enviado correctamente')
while temp<60:
    temp=temp+1
    time.sleep(1)
    print(temp)
```

Código 2.16 Configuración del nodo para el envío de mensajes SigFox.

3. Resultados y Discusión

En este capítulo se describen las pruebas de funcionamiento del prototipo. Las pruebas se van a realizar en 2 lugares diferentes en los que la red SigFox tenga cobertura.

Como objetivos de las pruebas se tiene:

- Verificar el cumplimiento de los requerimientos funcionales propuestos en el capítulo 2 sección 2.1.2.
- Verificar el correcto funcionamiento de la aplicación de usuario por medio de las siguientes pruebas:
    - Cargar plano de la edificación.
    - Ver, editar y eliminar planos cargados.
    - Ver diagramas de cobertura creados.
- Verificar el correcto funcionamiento de la aplicación para la configuración de los nodos por medio de la verificación de la recepción de los mensajes en el BackEnd de SigFox y su posterior creación de diagramas de cobertura creados.
- Comprobar el correcto funcionamiento del prototipo dentro de dos edificaciones puntuales.

Los 2 lugares para realizar las pruebas de funcionamiento se detallan a continuación:

1. El primer lugar de pruebas es en el edificio de Química/Eléctrica de la Escuela Politécnica, se consideró realizar las pruebas en el subsuelo y planta baja de este edificio ya que son lugares críticos en donde se espera tener poca cobertura, debido a como se encuentra construido el edificio. En lo posible se trató de realizar mediciones en todas las aulas, oficinas y laboratorios del edificio, pero por las restricciones existentes debidas a la pandemia del COVID 19 a la fecha en que se realizaron estas pruebas solo se tuvo acceso a las aulas de eléctrica del mismo. En la Figura 3.1 se aprecia el edificio de Química/Eléctrica en la que se realizaron las pruebas.

Figura 3.1 Edificio Química Eléctrica.

2. El segundo lugar de pruebas es el Gimnasio *Ignite Athletics*, el mismo que debido a la pandemia se vio obligado a dividir toda el área destinada para hacer ejercicio en estaciones, las mismas que solo pueden ser ocupadas por una persona y para ser ocupadas deben ser reservadas, el dueño ha pensado en automatizar este proceso con el internet de las cosas, por lo que las mediciones del LQI dentro de la edificación se realizará en cada estación. En la Figura 3.2 se aprecia el interior del gimnasio Ignite Athletics.

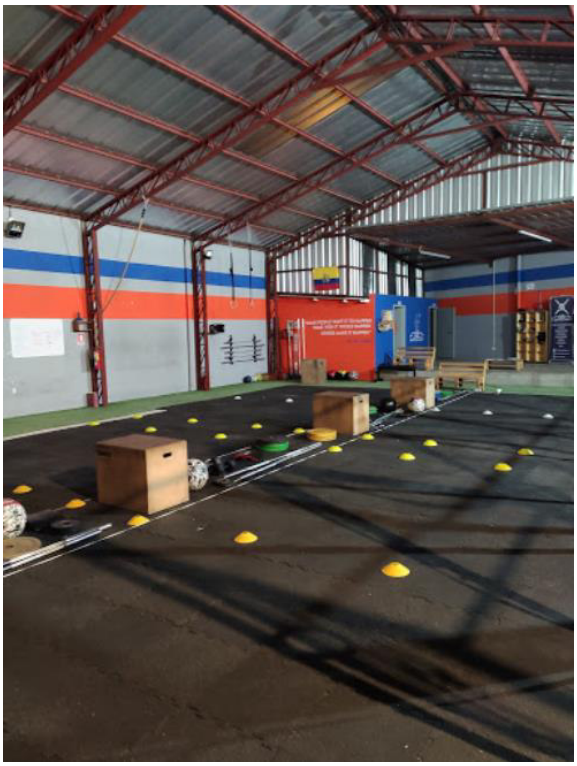

Figura 3.2 Gimnasio “Ignite Athletics”.

3.1. Pruebas de cumplimiento de los requerimientos funcionales del prototipo.

Las pruebas están basadas en los requerimientos presentados en la sección 2.1.2.1, las mismas que servirán para las pruebas que se realizan en el escenario de prueba 1 en la sección 3.2 y en el escenario de prueba 2 en la sección 3.3

3.1.1. Cargar plano de la edificación.

Como requerimiento funcional se tiene que el usuario debe cargar un plano de la edificación en la aplicación antes de realizar las mediciones, para esto se ingresa el nombre del plano, una imagen del plano en formato JPEG y cada imagen del plano debe ser de solo un piso de la edificación, para las pruebas se escogerá la imagen de un plano como ejemplo y se verificará que el mismo sea cargado y guardado en la base de datos.

Para cargar un plano se tiene los siguientes pasos:

1. Ingresar el nombre del plano en la aplicación de usuario como en la Figura 3.3 en este caso planoDeEjemplo.

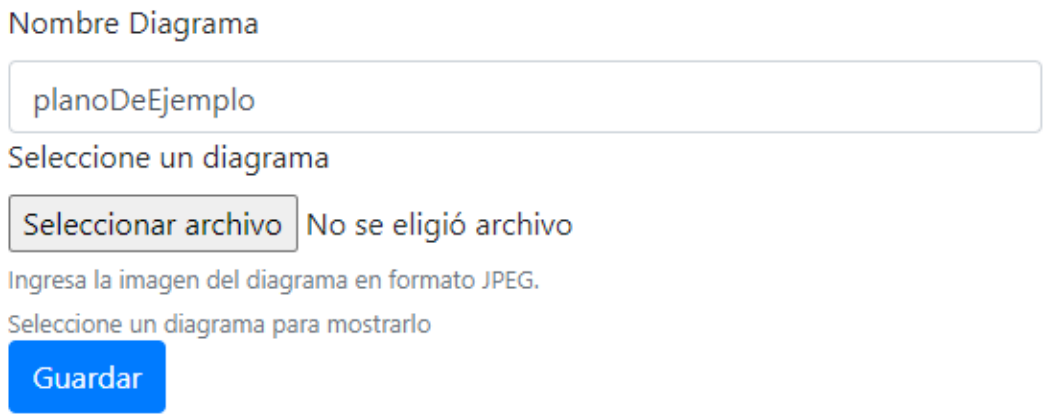

Figura 3.3 Ingreso del nombre del diagrama.

2. Como siguiente paso se da clic en seleccionar un archivo, se selecciona un archivo y se procede a dar guardar en el botón guardar como se muestra en la Figura 3.4, después de esto la aplicación guarda la información en la base de datos.

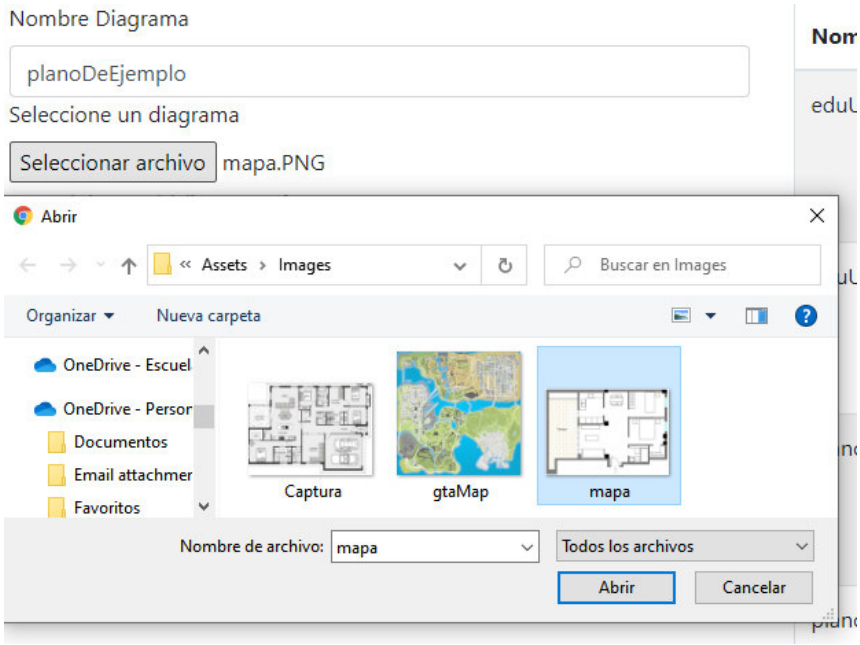

Figura 3.4 Ingreso del plano de ejemplo.

Para comprobar el correcto funcionamiento y que el plano realmente se encuentre cargado se verifica que se muestre en los diagramas ya cargados como se muestra en la Figura 3.5 y posteriormente se comprueba que en la base de datos exista el plano como se muestra en la Figura 3.6.

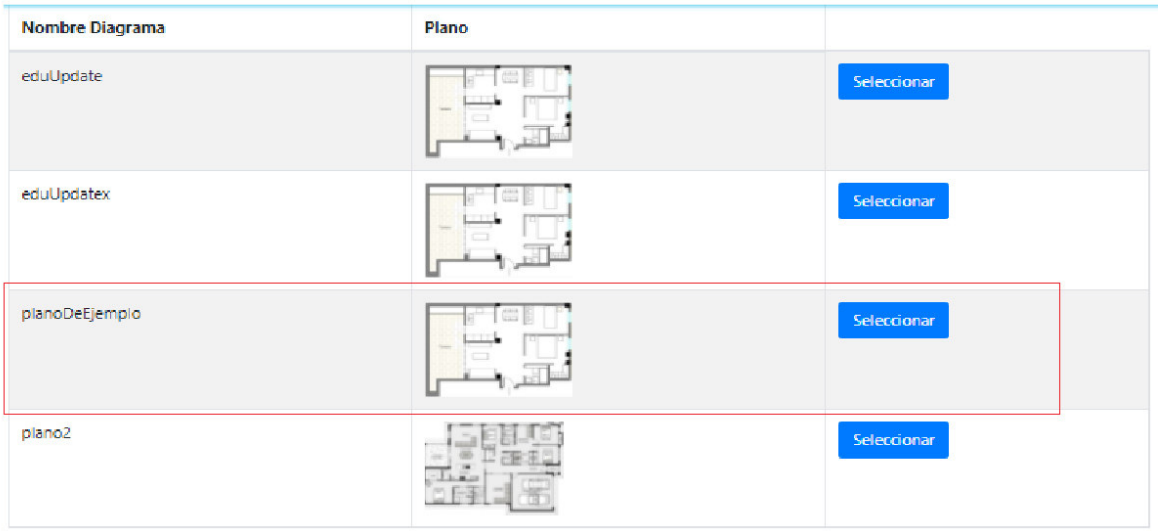

Figura 3.5 Verificación del plano cargado.

	idDiagrma	nombre	plano
1	44	eduUpdate	"...
2	112	eduUpdatex	...
3	115	planoDeEjemplo	"...
4	116	plano2	"...

Figura 3.6 Verificación del plano cargado en la base de datos.

Como se puede observar el requerimiento fue cumplido correctamente.

3.1.2. Ver, Editar y Eliminar plano cargados.

El prototipo como requerimiento debe permitir ver, editar y eliminar los planos cargados en la base de datos. En la Figura 3.7 se puede observar que el prototipo permite ver los planos existentes en la base de datos.

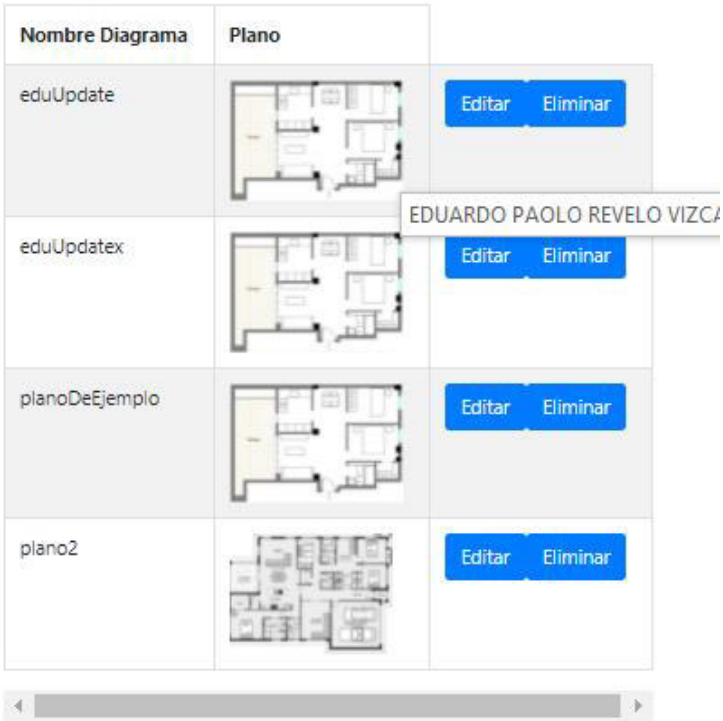

Figura 3.7 Verificación de los planos cargados en la base de datos.

La aplicación a un lado de la imagen del plano tiene el botón Editar como se observa en la Figura 3.7. Al dar clic en el botón Editar nos muestra un formulario en el que se puede renombrar el plano y volver a cargar el plano. Ver Figura 3.8.

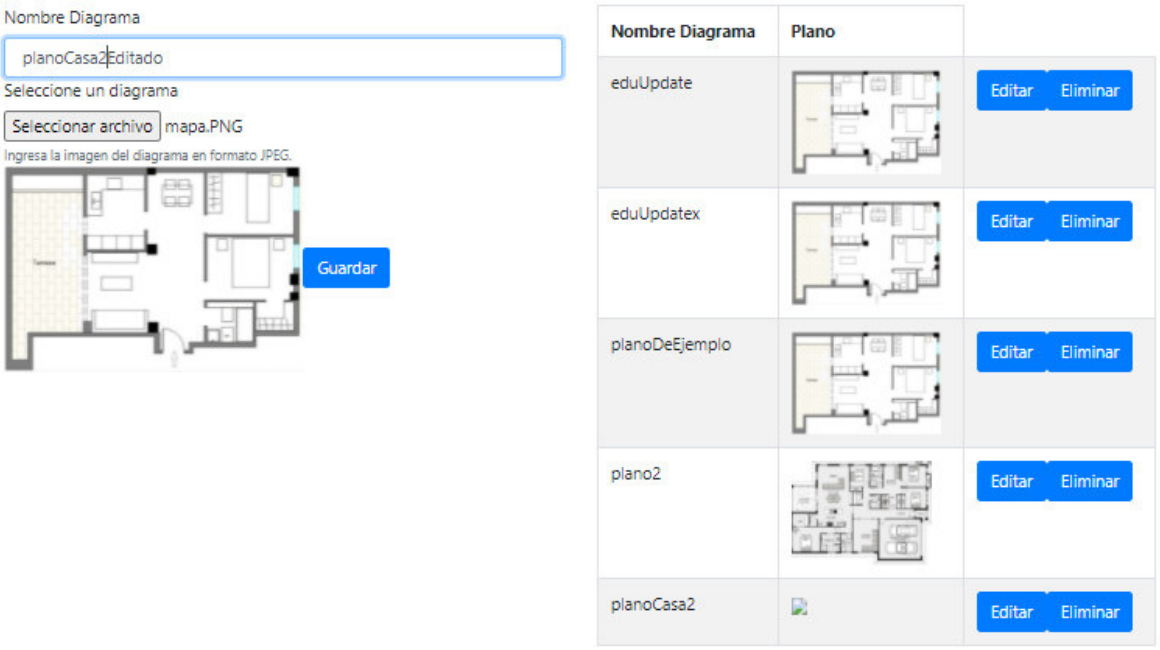

Figura 3.8 Ejemplo de edición de un plano.

Para finalizar de editar se da clic en guardar y los nuevos datos son actualizados en la base de datos como se ve en la Figura 3.9 como comprobación de que realmente fueron actualizados.

	idDiagrma	nombre	plano
1	44	eduUpdate	"...
2	112	eduUpdatex	...
3	115	planoDeEjemplo	"...
4	116	plano2	"...
5	171	planoCasa2Editado	[]

Figura 3.9 Verificación en la base de datos.

Como se puede observar en la Figura 3.8 el nombre planoCasa2 es editado al nombre planoCasa2Editado. Figura 3.9.

Para eliminar un plano previamente guardado se debe dar clic en el botón eliminar que se encuentra a lado de la imagen del plano a eliminar, un ves se haya dado clic el plano será eliminado de la base de datos, en la Figura 3.10 se da clic en el botón eliminar del plano llamado planoDeEjemplo1, en la Figura 3.11 se puede observar que el plano llamado planoDeEjemplo1 existe en la base de datos antes de dar clic en eliminar y en la Figura 3.12 se comprueba que el plano ya no existe en la base de datos.

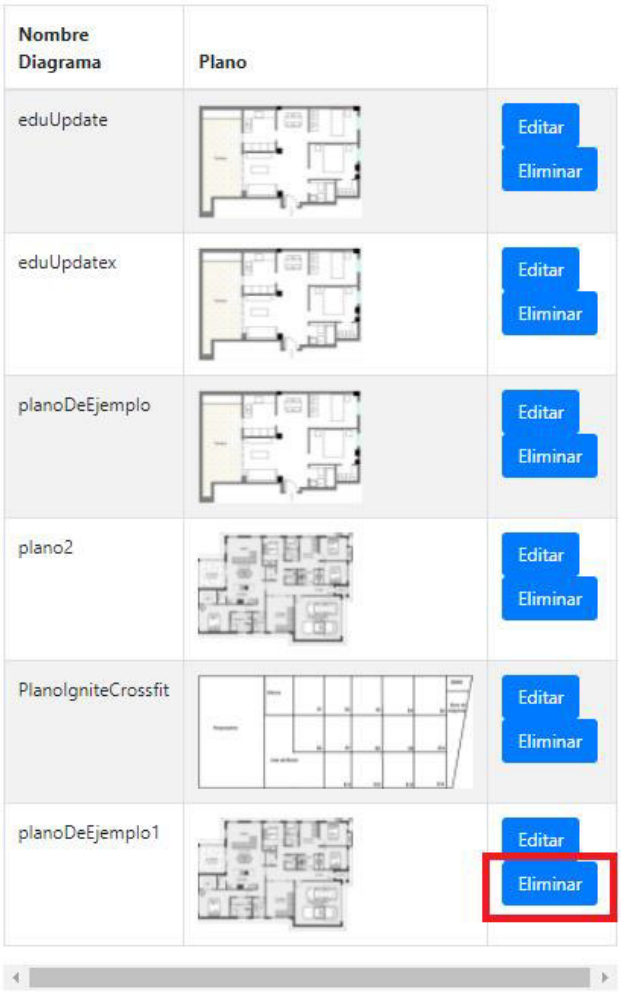

Figura 3.10 Ejemplo de eliminar plano.

	idDiagrma	nombre	plano
1	44	eduUpdate	"...
2	112	eduUpdatex	...
3	115	planoDeEjemplo	"...
4	116	plano2	"...
5	172	PlanoIgniteCrossfit	...
6	174	planoDeEjemplo1	...

Figura 3.11 Verificación del plano antes de eliminar.

	idDiagrma	nombre	plano
1	44	eduUpdate	"...
2	112	eduUpdatex	...
3	115	planoDeEjemplo	"...
4	116	plano2	"...
5	172	PlanoIgniteCrossfit	...

Figura 3.12 Verificación del plano eliminado.

3.1.3. Configuración del nodo

El requerimiento del sistema a implementar, indica que se permita la configuración del nodo antes del envió de cada mensaje con la información de la calidad del enlace de la red SigFox. El nodo necesita ser configurado antes del envío de cada mensaje para que con la información ingresada cambie la carga útil del mensaje a enviar por la red SigFox.

El usuario para configurar el nodo debe ingresa la ubicación del nodo dentro de la edificación en coordenadas X y Y, y un identificador del diagrama.

```
Ingrese el ID del diagrama
Ingrese un número entero: 172
Ingrese la posición en X
Ingrese un número entero: 325
Ingrese la posición en Y
Ingrese un número entero: 350
<class 'str'>
172/325/350
Mensaje enviado correctamente
1
```

Figura 3.13 Configuración del nodo SigFox.

En la Figura 3.13 se encuentra un ejemplo de la aplicación de configuración del nodo que permite el ingreso de los datos requeridos antes del envió del LQI por la red SigFox, si el mensaje es enviado correctamente se puede observar un mensaje que dice “Mensaje Enviado Correctamente”. Como verificación en la Figura 3.14 se puede observar dentro del BackEnd de SigFox que el mensaje llego de manera correcta.

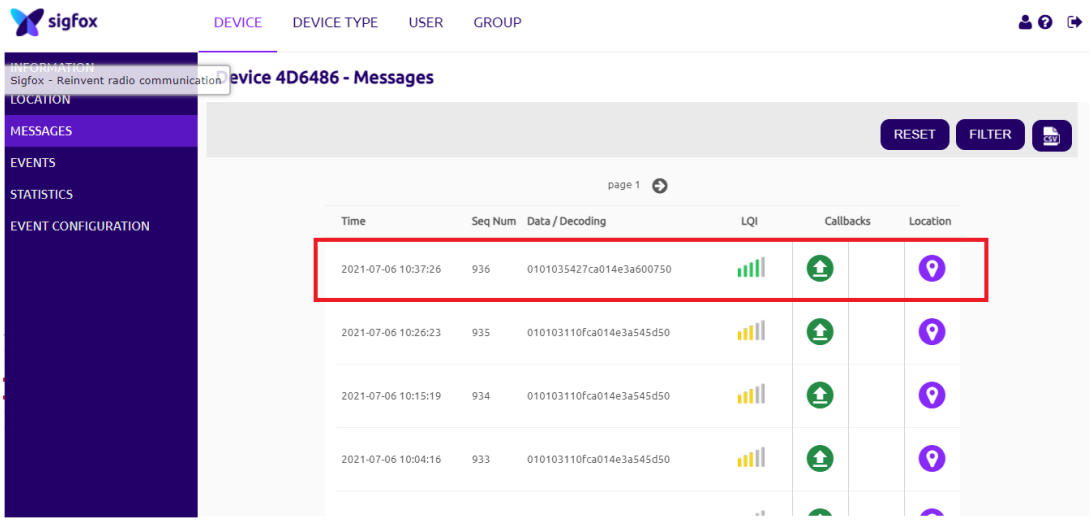

Figura 3.14 BackEnd SigFox.

3.1.4. Ver Diagramas de cobertura creados.

El prototipo dentro de la aplicación de usuario permite observar los niveles de la señal de recepción. Para su creación la aplicación usa: el plano cargado, los mensajes que le corresponden a cada plano con su LQI y su posición en X e Y. Cada nivel de LQI tiene un color diferente de acuerdo con el contenido de cada mensaje enviado por el nodo sensor como se observa en la Tabla 3.1.

Tabla 3.1 Nivele de LQI con su correspondiente color.

Nivel de LQI	Color
Excelente/Bueno	
Bueno/Promedio	
Limitado	

Para ver un diagrama de cobertura dentro de la aplicación, se selecciona un plano dentro de la lista de planos existentes dando clic en el botón seleccionar del plano que queramos observar cómo se ve en la Figura 3.15 y después se observa el plano seleccionado como se ve en la Figura 3.16, en el cual a manera de ejemplo existen 9 mediciones, cada una de acuerdo con su LQI medido y según la Tabla 3.1 es dibujada en el diagrama.

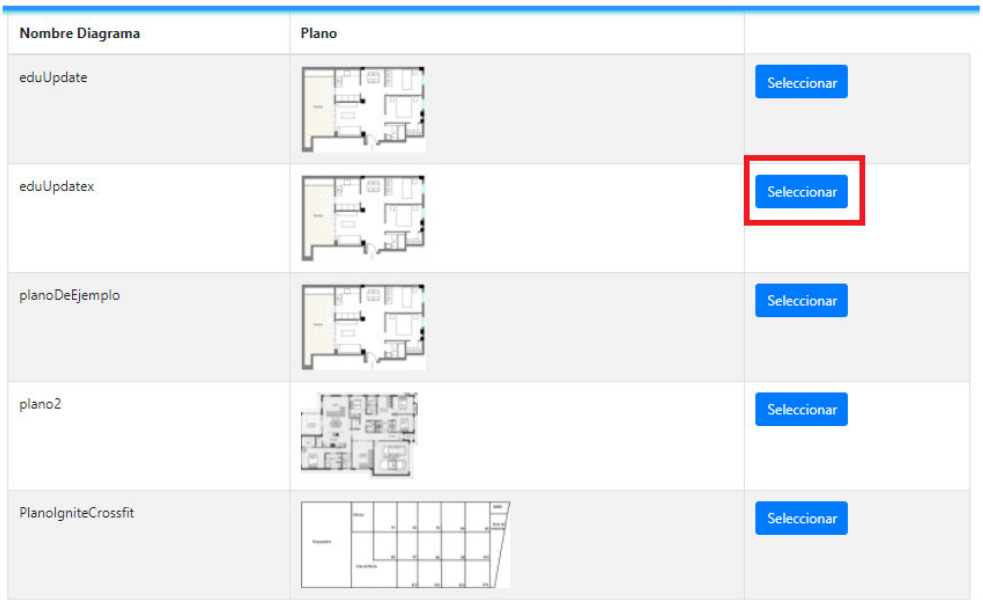

Figura 3.15 Lista de diagramas de cobertura aplicación de usuario.

Figura 3.16 Ejemplo del diagrama de cobertura seleccionado.

Para la creación de los diagramas de cobertura el prototipo lo hace en función de los niveles de intensidad recibidos que se observan en la Tabla 3.1, por lo que no exactamente son diagramas de cobertura, sino más bien son niveles de intensidad LQI recibidos en puntos específicos dentro de una edificación.

3.1.5. Visualización de los mensajes enviados por el nodo por cada plano.

Dentro de la aplicación de usuario el prototipo permite el observar en una tabla los mensajes con el LQI medido y sus coordenadas, los que permiten la creación de los diagramas de cobertura. Para ver los mensajes se selecciona un diagrama como se ve en la Figura 3.15 y aparecerá debajo del diagrama la tabla. En la Figura 3.17 está la tabla con los mensajes que crean el diagrama de la Figura 3.16.

Numero de secuencia	Fecha	Datos	LQI
100	8/4/2021 16:4:2	115/300/150	1
100	8/4/2021 16:4:2	115/450/150	2
100	8/4/2021 16:4:2	115/450/350	3
100	8/4/2021 16:4:2	115/650/300	1
100	8/4/2021 16:4:2	115/600/200	3
100	8/4/2021 16:4:2	115/600/50	2
100	8/4/2021 16:4:2	115/150/300	3
100	8/4/2021 16:4:2	115/150/100	3
100	8/4/2021 16:4:2	115/300/300	2

Figura 3.17 Tabla de mensajes enviados por diagrama de cobertura.

3.2. Escenario de pruebas 1.

El miércoles 14 julio del 2021 se procedió a realizar las pruebas correspondientes del prototipo en el edificio de Química/Eléctrica de la Escuela Politécnica Nacional.

Por motivos de las restricciones existentes, debido a la pandemia que atraviesa el mundo, a la fecha de realizar las pruebas solo se pudo tener acceso a las aulas ubicadas en el piso 3, 4 y 5 de la parte de la facultad de eléctrica.

Un inconveniente para la realización de las mediciones fue no tener acceso al plano del edificio, por lo que se tuvo que realizar uno por cuenta propia. En la Figura 3.18 se observa el plano realizado del tercer piso del edificio lo más cercano a la realidad y que sirva para el caso.

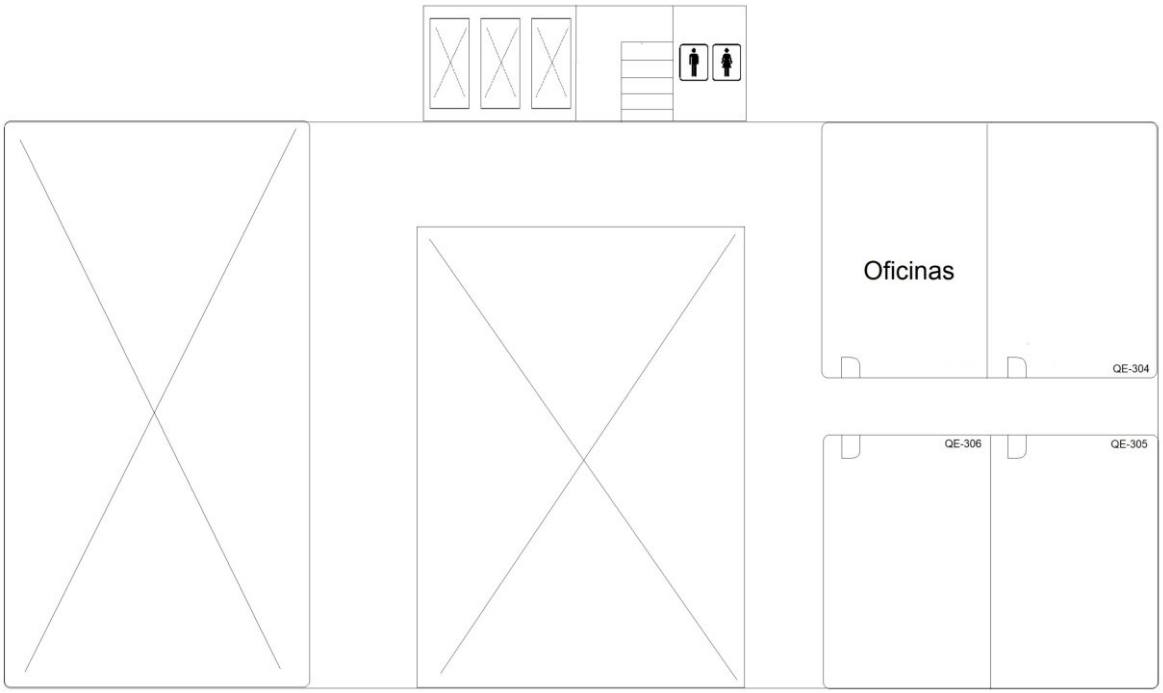

Figura 3.18 Plano de ejemplo Piso 3 Edificio Química/Eléctrica.

Como ejemplo se presenta las mediciones realizadas en el aula QE-304 Figura 3.19.

Figura 3.19 Aula QE-304.

Se tomaron medidas del LQI en cada esquina ya que se consideraron estos puntos críticos al estar rodeados de paredes, en la Figura 3.20 se observan los puntos tomados numerados.

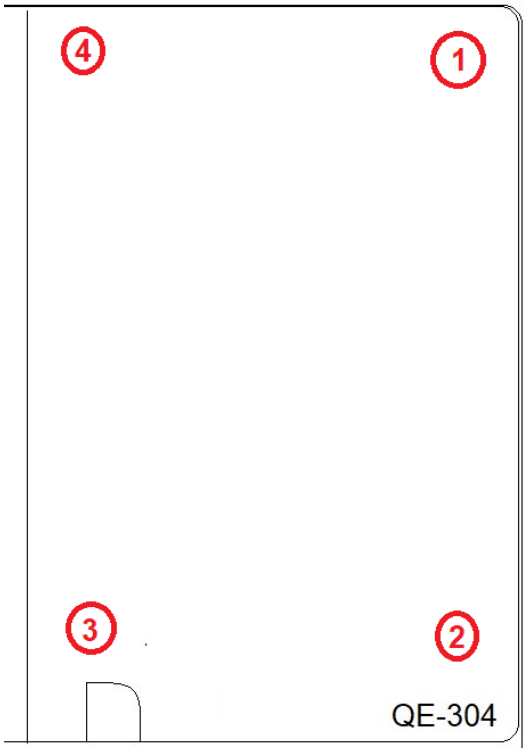

Figura 3.20 Plano aula QE-304.

Para esto se configuró el nodo antes del envío de cada mensaje como se observa en las siguientes figuras que se presentan como evidencia de las pruebas realizadas:

1. Esquina 1

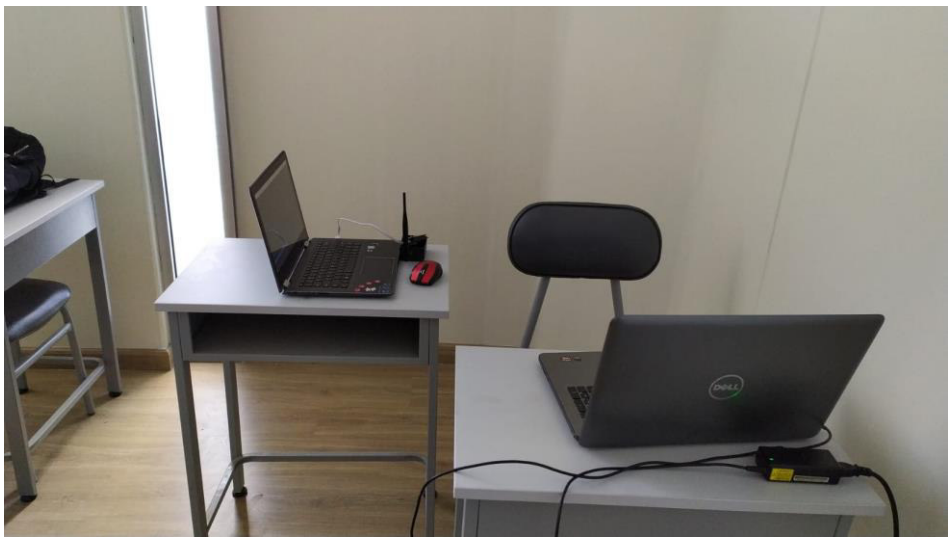

Figura 3.21 Esquina 1 aula QE-304.

Como primer paso se configuró el nodo con los parámetros necesarios para el envío del mensaje como se observa en la Figura 3.22.

```
Ingrese el ID del diagrama
Ingrese un número entero: 178
Ingrese la posición en X
Ingrese un número entero: 733
Ingrese la posición en Y
Ingrese un número entero: 310
<class 'str'>
178/733/310
Mensaje enviado correctamente
```

Figura 3.22 Configuración del nodo mensaje 1 Aula QE-304.

Se comprueba que el mensaje haya sido recibido en el BackEnd SigFox como se observa en la Figura 3.23.

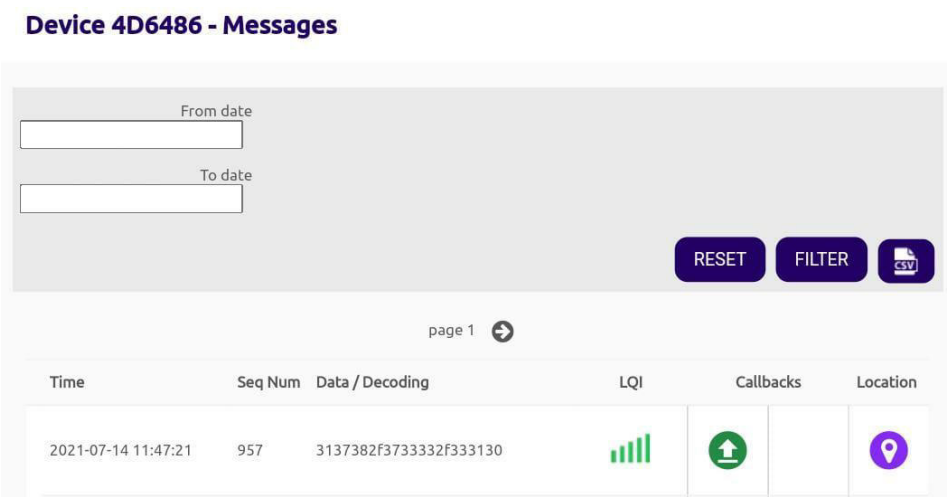

Figura 3.23 Mensaje 1 aula QE-404 en el BackEnd SigFox.

2. Esquina 2

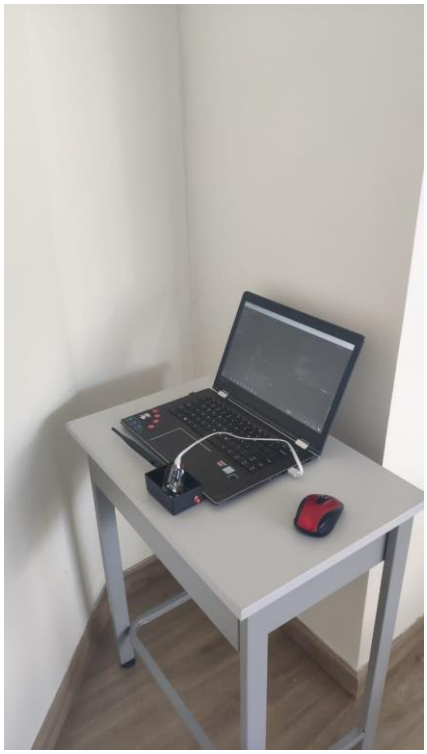

Figura 3.24 Esquina 2 aula QE-404.

Se configuró el nodo con los parámetros necesarios para el envío del mensaje como se observa en la Figura 3.25.

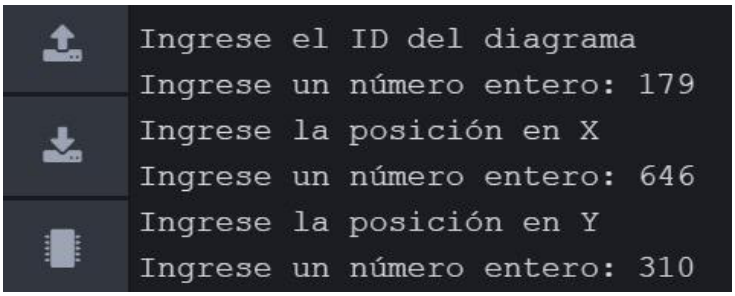

Figura 3.25 Configuración del nodo mensaje 2 Aula QE-304.

Se comprueba que el mensaje haya sido recibido en el BackEnd SigFox como se observa en la Figura 3.26.

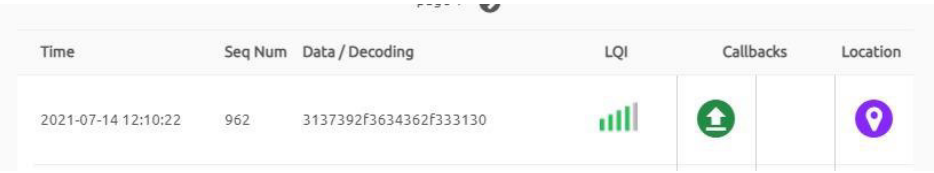

Figura 3.26 Mensaje 2 aula QE-404 en el BackEnd SigFox.

3. Esquina 3

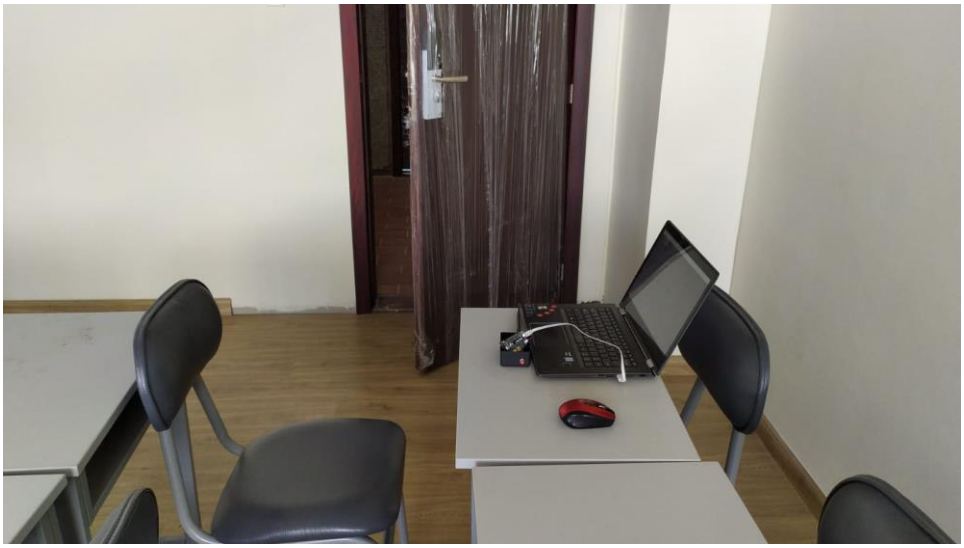

Figura 3.27 Esquina 3 aula QE-404

Se configuró el nodo con los parámetros necesarios para el envío del mensaje como se observa en la Figura 3.28.

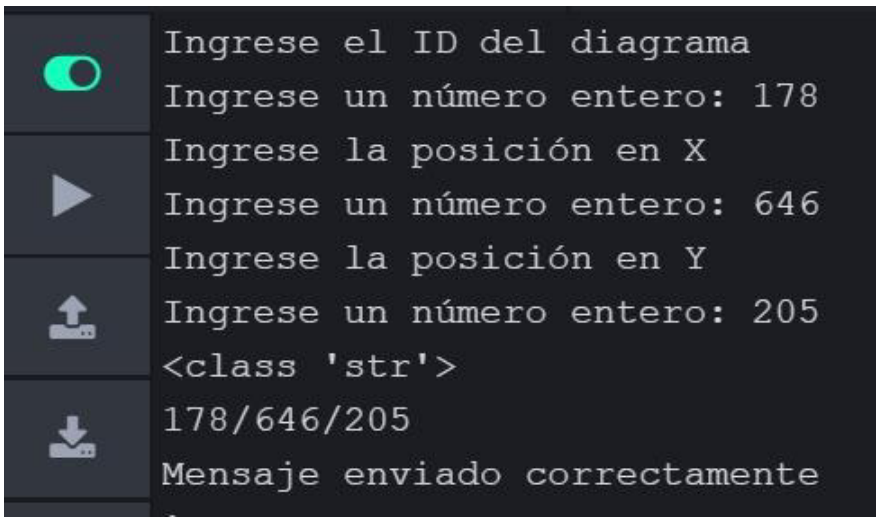

Figura 3.28 Configuración del nodo mensaje 3 Aula QE-404.

Se comprueba que el mensaje haya sido recibido en el BackEnd SigFox como se observa en la Figura 3.29.

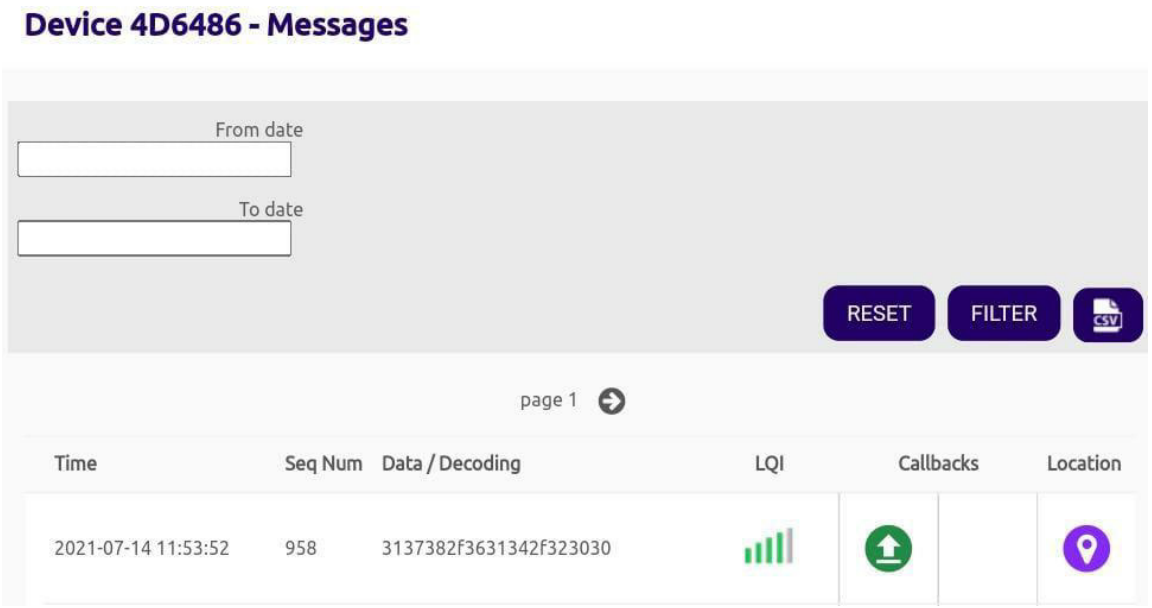

Figura 3.29 Mensaje 3 aula QE-404 en el BackEnd SigFox.

4. Esquina 4

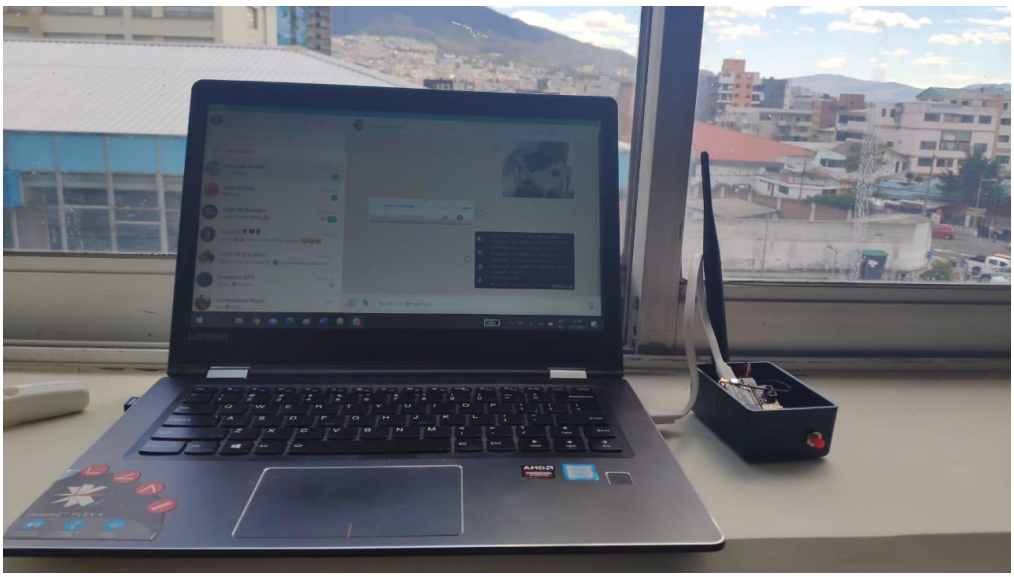

Figura 3.30 Esquina 4 aula QE-404.

Se configuró el nodo con los parámetros necesarios para el envío del mensaje como se observa en la Figura 3.31.

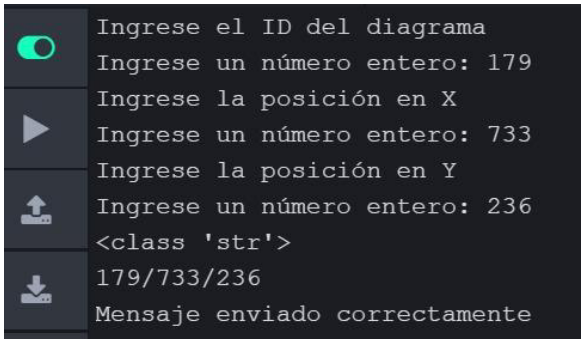

Figura 3.31 Configuración del nodo mensaje 4 Aula QE-404

Se comprueba que el mensaje haya sido recibido en el BackEnd SigFox como se observa en la Figura 3.32.

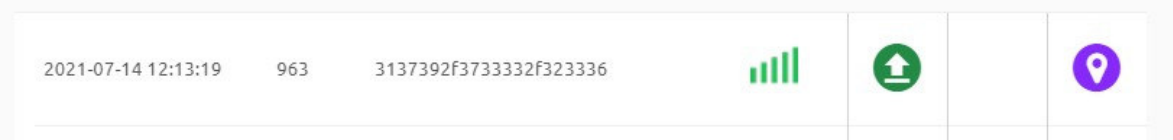

Figura 3.32 Mensaje 4 aula QE-404 en el BackEnd SigFox.

A continuación, se presenta los resultados obtenidos de las aulas de los pisos en los que se realizó mediciones dentro del edificio de Química-Eléctrica y sus respectivas conclusiones:

- Piso 3:

En la Figura 3.33 y la Figura 3.34 se presentan los resultados obtenidos de las mediciones realizas en el piso 3 del edificio de Química – Eléctrica.

Numero de secuencia	Fecha	Datos	LQI
960	14/7/2021 10:5:14	179/733/310	2
961	14/7/2021 10:7:4	179/646/236	2
962	14/7/2021 10:10:22	179/646/310	2
963	14/7/2021 10:13:19	179/733/236	3
965	14/7/2021 10:22:27	179/733/90	3
964	14/7/2021 10:21:6	179/733/155	2
967	14/7/2021 10:25:31	179/646/155	2
966	14/7/2021 10:24:3	179/646/90	2
969	14/7/2021 10:29:19	179/550/90	2
968	14/7/2021 10:28:4	179/550/155	2

Figura 3.33 Resultados obtenidos piso 3 edificio de Química-Eléctrica.

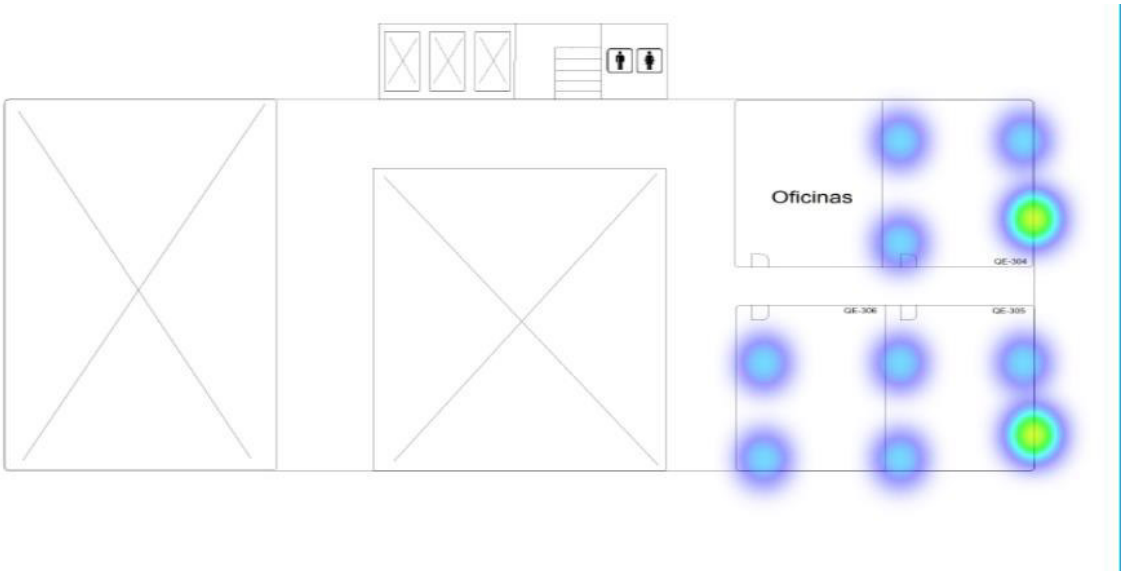

Figura 3.34 Resultados del piso 3 edificio de Química-Eléctrica.

- Piso 4:

En la Figura 3.35 y la Figura 3.36 se presentan los resultados obtenidos de las mediciones realizas en el piso 4 del edificio de Química – Eléctrica.

Numero de secuencia	Fecha	Datos	LQI
970	14/7/2021 10:39:30	180/733/310	2
975	14/7/2021 10:46:39	180/733/90	2
974	14/7/2021 10:45:25	180/733/155	2
973	14/7/2021 10:43:28	180/646/310	2
972	14/7/2021 10:42:7	180/646/236	2
971	14/7/2021 10:40:47	180/733/236	2
981	14/7/2021 10:54:18	180/550/310	2
980	14/7/2021 10:53:10	180/550/236	2
979	14/7/2021 10:51:37	180/550/90	2
978	14/7/2021 10:50:13	180/550/155	2
977	14/7/2021 10:49:0	180/646/155	2
976	14/7/2021 10:47:47	180/646/90	2

Figura 3.35 Resultados obtenidos piso 4 edificio de Química-Eléctrica.

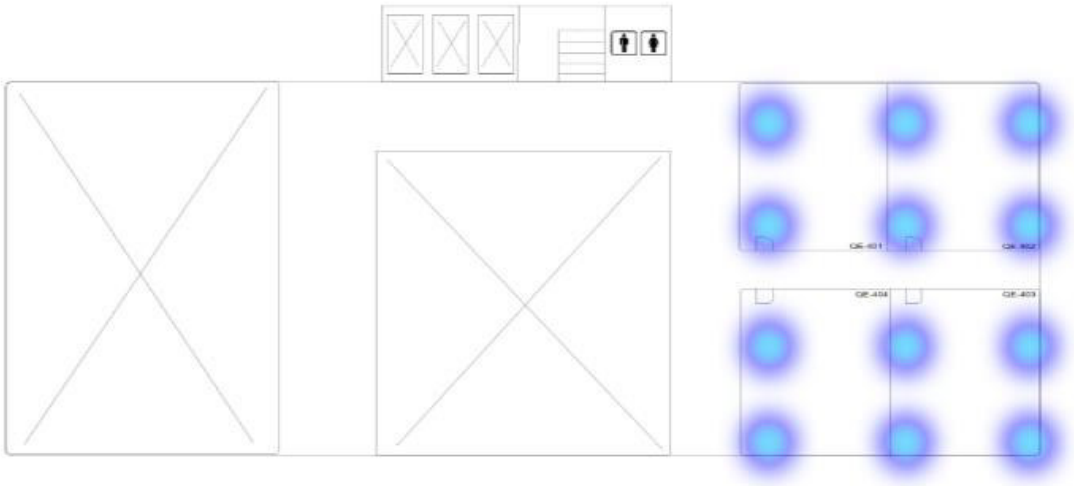

Figura 3.36 Resultados del piso 4 edificio de Química-Eléctrica.

- Piso 5:

En la Figura 3.37 y la Figura 3.38 se presentan los resultados obtenidos de las mediciones realizas en el piso 5 del edificio de Química – Eléctrica.

Numero de secuencia	Fecha	Datos	LQI
982	14/7/2021 10:55:57	181/733/310	2
984	14/7/2021 10:58:18	181/646/236	2
983	14/7/2021 10:57:8	181/733/236	2
987	14/7/2021 11:1:58	181/733/90	2
986	14/7/2021 11:0:51	181/733/155	2
985	14/7/2021 10:59:27	181/646/310	2
989	14/7/2021 11:4:34	181/646/155	2
988	14/7/2021 11:3:23	181/646/90	2
991	14/7/2021 11:8:3	181/550/155	2
994	14/7/2021 11:12:5	181/550/310	3
993	14/7/2021 11:10:54	181/550/236	3
992	14/7/2021 11:9:9	181/550/90	2

Figura 3.37 Resultados obtenidos piso 5 edificio Química-Eléctrica.

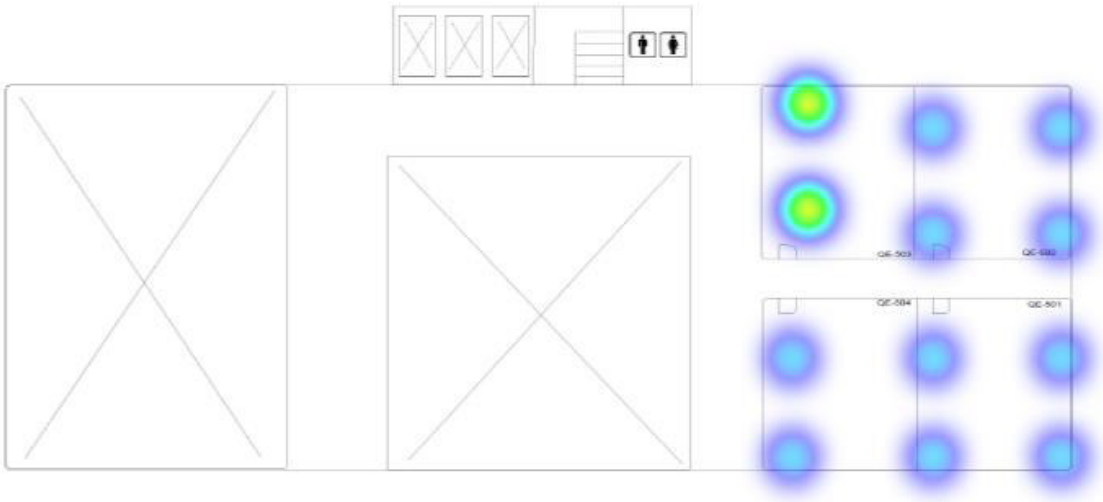

Figura 3.38 Resultados del piso 5 edificio de Química-Eléctrica.

De las pruebas realizadas se puede concluir que en la mayor parte del edificio se tiene un indicador del enlace de valor promedio como se puede comparar en la Tabla 3.1, con un indicador promedio similar en los 3 pisos.

Estos resultados sin un patrón bien definido en los valores del indicador de la calidad del enlace se debe a que dentro de la ciudad de Quito en donde se realizaron las pruebas existen varias radio bases por lo que el nodo se puede enlazar a una diferente cada que envía un mensaje con la información de la calidad del enlace cada 10 minutos y confirman la importancia de este trabajo de titulación ya que aunque se tenga cobertura dentro de la ciudad existirá lugares puntuales dentro de edificaciones que tendrán una pobre señal lo que podría afectar el buen funcionamiento de la aplicación que se fuera a realizar. Este prototipo ayuda a realizar una inspección de sitio del lugar en donde se vaya a implementar los nodos sensores.

3.3. Escenario de Pruebas 2

El viernes 9 de Julio del 2021 se procedió a realizar mediciones dentro del gimnasio "Ignite" ubicado al norte de la ciudad de Quito en el barrio de Ponceano, en la Figura 3.39 se puede observar su ubicación exacta.

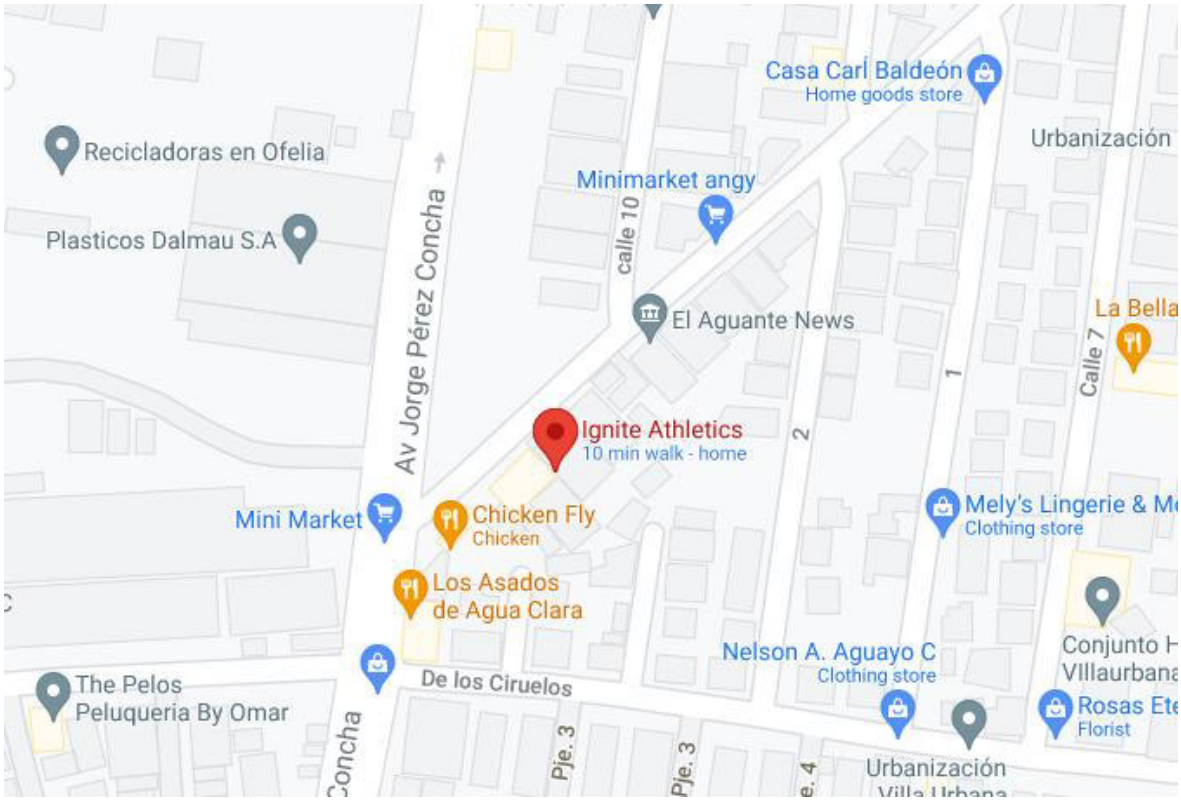

Figura 3.39 Ubicación del Gimnasio “Ignite”.

En la Figura 3.40 se observa el plano del gimnasio, el mismo que por motivos de la pandemia se encuentra dividido en 14 estaciones de trabajo, las mismas que son solo ocupadas por un usuario, en la Figura 3.41 se puede ver como ejemplo una foto de una estación de trabajo.

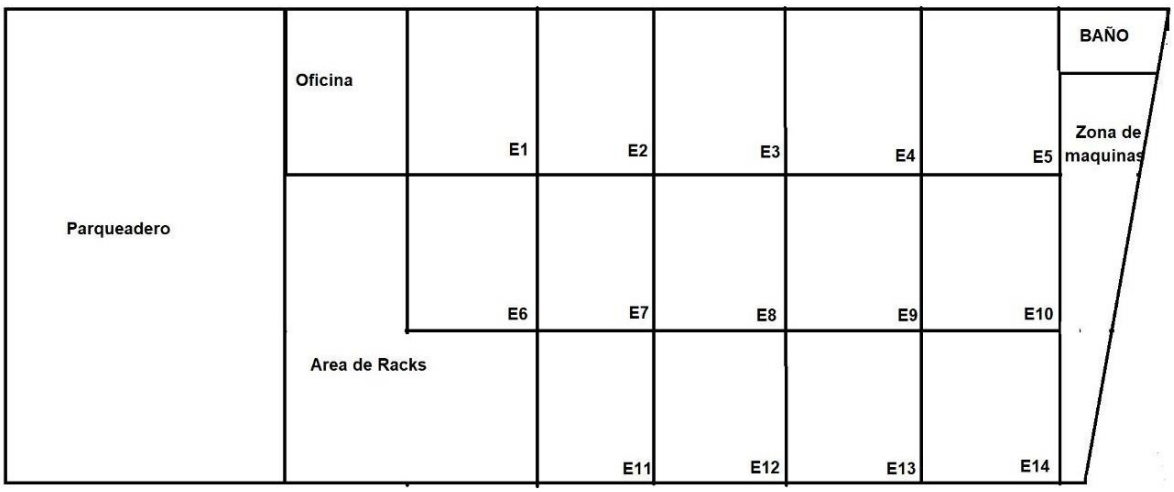

Figura 3.40 Plano del Gimnasio “Ignite”.

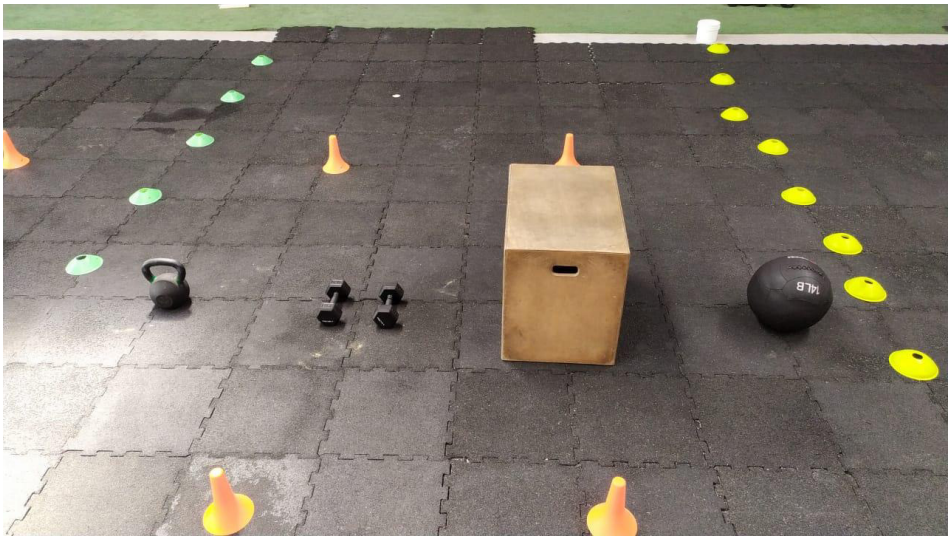

Figura 3.41 Estación de trabajo del Gimnasio “Ignite”.

Las zonas de interés en este caso son las estaciones de trabajo, por lo que se realizó mediciones en las catorce estaciones, como se observa en la Figura 3.42.

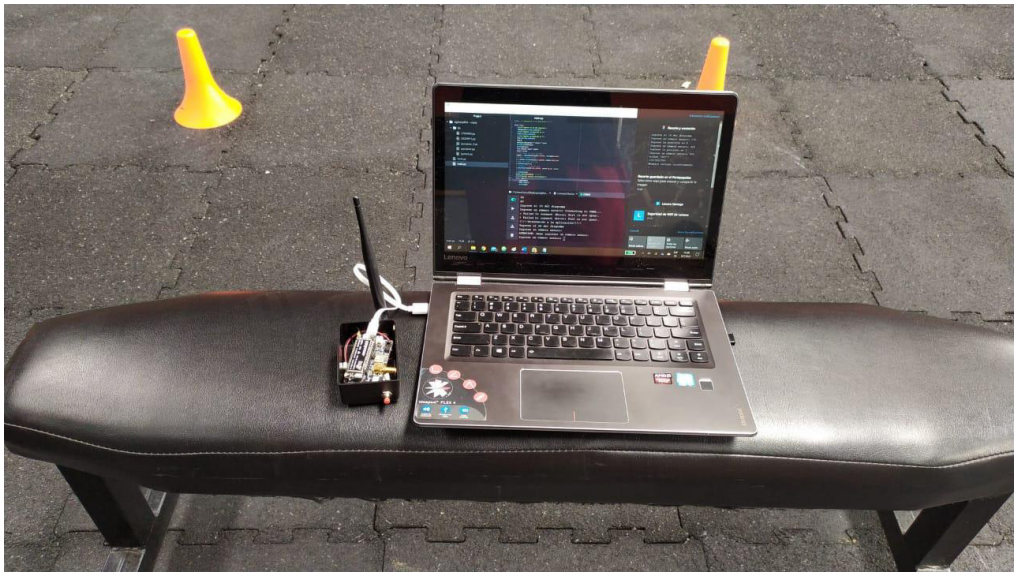

Figura 3.42 Medición en una estación Gimnasio “Ignite”.

Para realizar la medición se siguió lo siguientes pasos:

1. Cargar el plano del gimnasio “Ignite” dentro de la aplicación de usuario del prototipo como en la Figura 3.43.

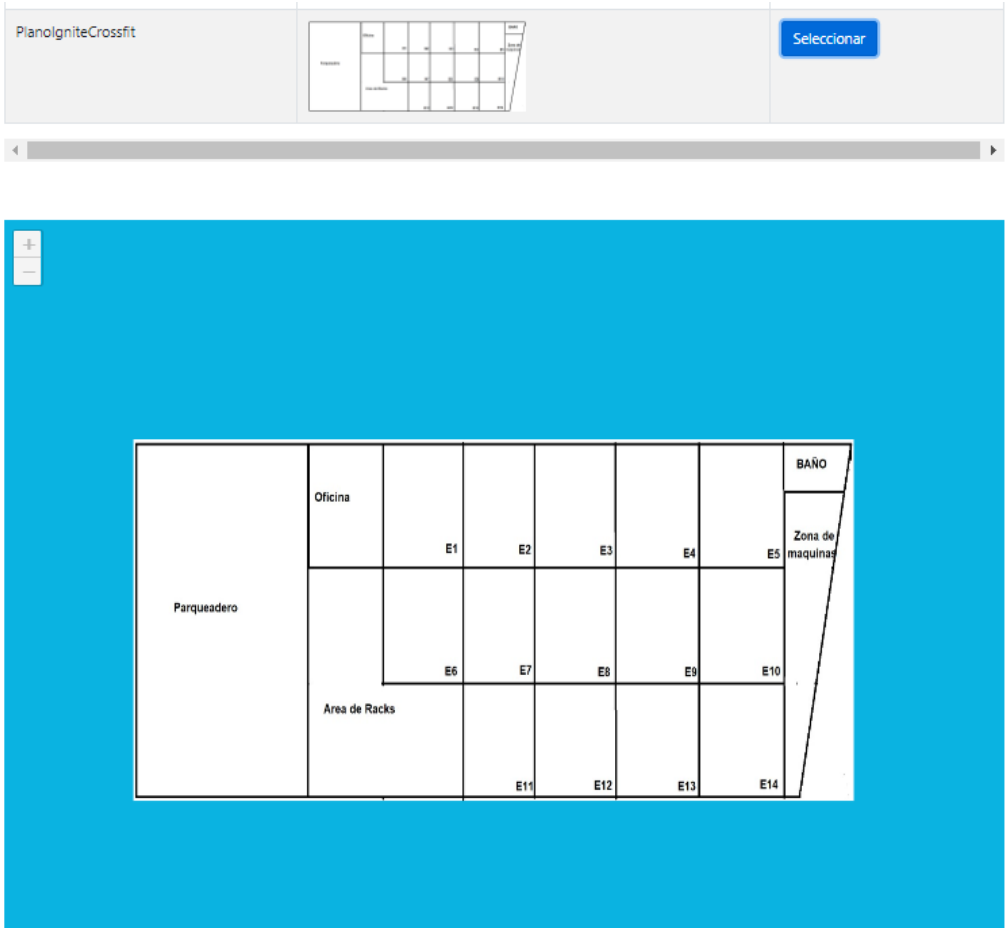

Figura 3.43 Plano del Gimnasio “Ignite” cargado en la aplicación de usuario.

2. Configurar el nodo para enviar el mensaje por la red SigFox con la medición del LQI para cada una de las 14 estaciones, teniendo en cuenta que entre cada mensaje corre un temporizador de 10 minutos para poder a volver a configurar el nodo. En cada estación se configura el nodo con el id de diagrama correspondiente al plano del gimnasio y la ubicación de la estación dentro del plano, en la Figura 3.44 se observa un ejemplo.

Figura 3.44 Configuración del nodo.

Como resultado de las mediciones realizadas se obtuvo los siguientes resultados:

En la Figura 3.45 y la Figura 3.46 se puede observar el resultado de las mediciones realizadas en cada estación dentro del gimnasio, y se comprobó que en las estaciones que están cerca del área de racks tienen un LQI con valor de 3 lo que indica que tienen una buena cobertura debido a que se encuentran junto a una puerta, por otra parte, las estaciones que se encuentran cerca de las zona de máquinas tiene un LQI de 1 esto se debe a que la pared del gimnasio esta adosada a una pared de un edificio lo que complica la conectividad de la red, las demás mediciones realizadas muestra un nivel medio de conexión lo que es suficiente para no tener problemas al envío de mensajes de la red SigFox.

Numero de secuencia	Fecha	Datos	LQI
962	9/7/2021 4:42:58	172/325/350	2
963	22/7/2021 4:44:38	172/400/350	2
962	22/7/2021 4:48:38	172/575/350	2
965	22/7/2021 4:50:38	172/660/350	1
966	22/7/2021 4:52:38	172/660/200	1
962	9/7/2021 4:42:58	172/660/50	1
968	22/7/2021 4:55:38	172/325/200	3
969	22/7/2021 4:58:38	172/400/200	3
970	22/7/2021 4:59:38	172/400/50	3
971	22/7/2021 5:2:38	172/475/200	2
972	22/7/2021 5:4:38	172/575/200	2
973	22/7/2021 5:5:38	172/575/50	1
974	22/7/2021 5:7:38	172/475/50	2
964	22/7/2021 4:46:38	172/475/350	3

Figura 3.45 Resultados obtenidos Gimnasio “Ignite”.

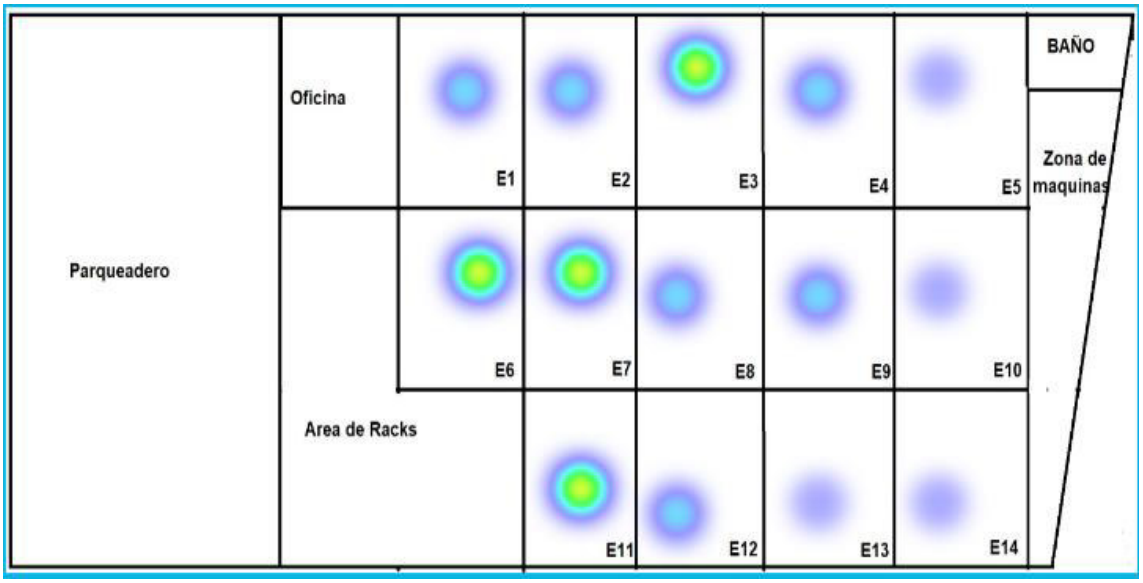

Figura 3.46 Resultados gimnasio Ignite.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

Al finalizar el proyecto de titulación se logró desarrollar un prototipo que permite la obtención de la calidad del enlace de un lugar puntual dentro de una edificación. Este prototipo ayuda a expertos que estén desarrollando aplicaciones usando la red SigFox a saber la calidad del enlace que tendrán dentro del lugar en donde vaya a desplegar la aplicación.

Las entrevistas realizadas a personas con conocimientos o experiencia con la red SigFox, sirvieron para obtener los requerimientos funcionales y no funcionales del prototipo. Estos requerimientos fueron la base para el diseño de los diferentes subsistemas que componen al prototipo.

La implementación de los subsistemas que conforman al prototipo se lo realizó siguiendo el diseño propuesto en la sección 2.1 de este proyecto. En el subsistema de la base de datos se creó una base de datos SQL con todas las tablas necesarios para la manipulación de la información, en el subsistema de aplicación servidor se creó todas las clases y sus respectivos métodos para la conexión entre el BackEnd SigFox, la base de datos y el subsistema de la aplicación de usuario. En el subsistema de la aplicación de usuario se creó un método para la conexión con el subsistema de la aplicación servidor además de la interfaz gráfica que permite la interacción del usuario con el prototipo. Finalmente, en el subsistema para configuración del nodo sensor se creó un programa que permite la conexión de una computadora con el nodo para su configuración.

Las pruebas del cumplimiento de los requerimientos funcionales realizadas permitieron comprobar el correcto funcionamiento y con lo establecido dentro del diseño del prototipo. Por otra parte, las pruebas realizadas físicamente en el Gimnasio “Ignite” y en las aulas del edificio de Química – Eléctrica en la Escuela Politécnica Nacional, permitieron comprobar la utilidad y la importancia del prototipo ya que existen puntos específicos dentro de las edificaciones que por varios motivos se puede tener baja calidad del enlace lo que provocaría errores y el no adecuado funcionamiento de la aplicación que se fuera a realizar.

El resultado de estas pruebas nos permite concluir que a pesar de la existencia de una zona de cobertura dada por SigFox hay lugares en donde las señales recibidas son diferentes debido al entorno físico de donde se envía la señal, las señales puede disminuir con la posibilidad de que el nodo sensor no pueda recibir ni transmitir con una señal adecuada.

El prototipo para la obtención de la zona de cobertura de la red SigFox en el interior de edificaciones nos permite obtener el indicador de la calidad de la red en un punto fijo dentro de edificaciones.

El subsistema Servidor del prototipo puede ser de gran utilidad para la implementación de futuros proyectos de IoT con tecnología SigFox.

El subsistema de Aplicación de Usuario puede adaptarse a futuros proyectos de IoT en que se utilice la tecnología SigFox.

La librería de JavaScript Heatmap.js es de gran utilidad al momento de graficar la cobertura SigFox en un punto específico.

4.2. RECOMENDACIONES

El prototipo desarrollado cumple adecuadamente con los requerimientos funcionales y no funcionales, sin embargo, el mismo puede mejorarse o adecuarse de diferentes maneras para un mejor funcionamiento, por lo cual se presentan recomendaciones para futuros trabajos y recomendaciones técnicas.

Recomendaciones para futuros trabajos:

- Se recomienda la expansión del prototipo para que la aplicación de usuario pueda de manera remota configurar los nodos sensores sin la necesidad de una conexión directa con el mismo.
- Debido a que la aplicación de usuario es web y está separada de la aplicación servidor, se recomienda escalar la aplicación de usuario a una aplicación móvil haciendo uso de la misma aplicación servidor.
- Para un futuro trabajo se recomienda la obtención de la ubicación dentro de la edificación a través de un módulo GPS, lo cual permitiría automatizar y agilizar el proceso.

Recomendaciones técnicas:

- Es recomendable que para sacar a producción al prototipo se lo realiza a través del uso de los servicios que ofrece la nube, ya que estos facilitan el proceso.
- Para el correcto funcionamiento del prototipo:
    1. Se debe verificar la existencia de una conexión a internet en el lugar que se quiere realizar las mediciones.

2. Comprobar que existe carga en la batería del nodo sensor.
3. Verificar que la antena este conectada adecuadamente al nodo sensor.
4. Mirar en la página oficial de SigFox que exista cobertura en el lugar que se vaya a realizar las mediciones.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] Laura Joris, Francois Dupont, Philippe Laurent, Pierre Bellier, Serguei Stoukatch, y Jean-Michel Redouté, «An Autonomous Sigfox Wireless Sensor Node for Environmental Monitoring», *IEEE Sens. Lett.*, vol. 3, n.º 7, pp. 01-04, jul. 2019, doi: 10.1109/LSENS.2019.2924058.

[2] Carles Gomez, Juan Carlos Veras, Rafael Vidal, Lluís Casals, y Josep Paradells, «A Sigfox Energy Consumption Model», *Sensors*, vol. 19, n.º 3, Art. n.º 3, ene. 2019, doi: 10.3390/s19030681.

[3] «WND Ecuador», *WND Group*. https://www.wndgroup.io/ecuador/ (accedido feb. 10, 2021).

[4] «SIGFOX.COM». https://www.sigfox.com/en (accedido feb. 10, 2021).

[5] «Radio Configurations | Sigfox build». https://build.sigfox.com (accedido feb. 20, 2021).

[6] Carles Gomez, Juan Carlos Veras, Rafael Vidal, Lluís Casals y Josep Paradells, «A Sigfox Energy Consumption Model», *Sensors*, vol. 19, p. 681, feb. 2019, doi: 10.3390/s19030681.

[7] Stephen Farrell <stephen.farrell@cs.tcd.ie>, «Low-Power Wide Area Network (LPWAN) Overview». https://tools.ietf.org/html/rfc8376#page-16 (accedido feb. 11, 2021).

[8] Yeonjoon Chung, Jae Young Ahn, y Jae Du Huh, «Experiments of A LPWAN Tracking(TR) Platform Based on Sigfox Test Network», en *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, oct. 2018, pp. 1373-1376. doi: 10.1109/ICTC.2018.8539697.

[9] «Sigfox Cloud Integration | Sigfox build». https://build.sigfox.com (accedido feb. 22, 2021).

[10] «Link Quality: general knowledge | Sigfox Resources». https://support.sigfox.com/docs/link-quality:-general-knowledge (accedido feb. 22, 2021).

[11] «SiPy - Pycom MicroPython Programable Triple-Bearer», *Pycom*. https://pycom.io/product/sipy/ (accedido feb. 10, 2021).

[12] Nicholas Tollervey, *Programming with MicroPython: Embedded Programming with Microcontrollers and Python*. O'Reilly Media, Inc., 2017.

[13] «MicroPython - Python for microcontrollers». http://micropython.org/ (accedido feb. 23, 2021).

[14] «Getting Started». https://docs.pycom.io/gettingstarted/ (accedido feb. 23, 2021).

[15] Robert Richards, «Representational State Transfer (REST)», en *Pro PHP XML and Web Services*, Robert Richards, Ed. Berkeley, CA: Apress, 2006, pp. 633-672. doi: 10.1007/978-1-4302-0139-7_17.

[16]«Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST)». https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (accedido feb. 23, 2021).
[17]Michael Jakl, *REST Representational State Transfer*.
[18]Mark Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O'Reilly Media, Inc., 2011.
[19]Vijay Surwase, «REST API Modeling Languages - A Developer's Perspective», vol. 2, n.º 10, p. 4.
[20]«JavaScript | MDN». https://developer.mozilla.org/en-US/docs/Web/JavaScript (accedido feb. 24, 2021).
[21]Douglas Crockford, *JavaScript: The Good Parts: The Good Parts*. O'Reilly Media, Inc., 2008.
[22]«About JavaScript - JavaScript | MDN». https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript (accedido feb. 24, 2021).
[23]«React – A JavaScript library for building user interfaces». https://reactjs.org/ (accedido feb. 24, 2021).
[24]Cory Gackenheimer, *Introduction to React*. Apress, 2015.
[25]«Dynamic Heatmaps for the Web». https://www.patrick-wied.at/static/heatmapjs/ (accedido feb. 25, 2021).

ANEXOS

ANEXO A. Entrevistas para la definición de requerimientos del prototipo.

ANEXO B. Resultado de las entrevistas realizadas.

ANEXO C. Script de la base de datos.

ANEXO D. Códigos del prototipo.

ANEXO F. Manual de usuario

ORDEN DE EMPASTADO