ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO DE APLICACIÓN PARA DETECTAR EL NIVEL DE OCUPACIÓN DE CONTENEDORES DE BASURA DE GRAN CAPACIDAD UTILIZANDO TECNOLOGÍA LORAWAN

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

BYRON RAFAEL VALLE PAILLACHO

DIRECTOR: ING. CARLOS ROBERTO EGAS ACOSTA M.Sc.

Quito, 29 de noviembre de 2022

AVAL

Certifico que el presente trabajo fue desarrollado por Byron Rafael Valle Paillacho, bajo mi supervisión.

Ing. Carlos Roberto Egas Acosta M.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Byron Rafael Valle Paillacho, 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.

BYRON RAFAEL VALLE PAILLACHO

DEDICATORIA

A mis padres y mi hermana por su apoyo y soporte incondicional a lo largo de la carrera. Este trabajo es el resultado haberme impulsado a ser una mejor versión de mí todos los días.

AGRADECIMIENTO

A mi madre Marcia por permitir que todo esto sea posible, por guiarme a lo largo de la carrera y hacer que las cosas sean mucho más sencillas. A Byron, mi padre, por su apoyo y esfuerzo en varios sentidos que me permitieron llegar a este punto. A Tatiana, mi hermana, por su cariño y disposición a ayudarme en todo lo que le sea posible.

A todos los profesores de la Escuela Politécnica Nacional que de alguna forma u otra aportaron a mi formación profesional y personal, a mi tutor el Ing. Víctor Reyes Cifuentes por compartir varios consejos que me ayudaron durante mis años en la FIEE.

A varios de mis familiares que están y ya no están conmigo, las lecciones aprendidas de ellos y su soporte en momentos críticos fueron fundamentales para llegar a este punto.

A mis amigos por su lealtad y apoyo, por estar en los buenos y malos momentos. A las personas con quienes he compartido durante estos años, de quienes he aprendido varias lecciones en diferentes aspectos.

Al Ing. Carlos Egas, por facilitar el desarrollo de este trabajo de titulación y por permitirme trabajar en una tecnología que únicamente conocía teóricamente, además de sus consejos y colaboración.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Varias urbes en el Ecuador han optado por un sistema de recolección de basura mecanizado, el cual consiste en ubicar contenedores de gran capacidad en zonas determinadas, con el fin de que estos sean desocupados periódicamente por camiones adaptados específicamente para esta tarea.

Sin embargo; debido al constante aumento de la población en las ciudades, este método resulta cada vez menos eficiente, ya que los contenedores no dan abasto en comparación al ritmo de generación de desperdicios. Esto provoca su colapso y por ende contaminación ambiental, visual e incluso pueden convertirse en focos de transmisión de enfermedades.

Este trabajo de titulación plantea un prototipo de aplicación que permita detectar el nivel de ocupación de estos contenedores, utilizando sensores ultrasónicos de bajo consumo de potencia basados en la tecnología LoRaWAN, además de herramientas de software que faciliten el desarrollo de una aplicación de escritorio que permita visualizar los datos obtenidos por los nodos de la red.

PALABRAS CLAVE: recolección de basura, contenedores de gran capacidad, LoRaWAN, sensores ultrasónicos, bajo consumo de potencia, aplicación de escritorio.

ABSTRACT

Several cities in Ecuador have opted for a mechanized garbage collection system, which consists of placing large-capacity containers in certain areas, so that they are periodically emptied by trucks specifically adapted for this task.

But nevertheless; Due to the constant increase in the population in the cities, this method is becoming less efficient, since the containers are not enough compared to the rate of waste generation. This causes their collapse and therefore environmental and visual contamination and can even become sources of disease transmission.

This degree work proposes an application prototype that allows detecting the occupancy level of these containers, using ultrasonic sensors with low power consumption based on LoRaWAN technology, as well as software tools that facilitate the development of a desktop application that allows visualize the data obtained by the nodes of the network.

KEY WORDS: garbage collection, large capacity containers, LoRaWAN, ultrasonic sensores, low power consumption, desktop application.

1. INTRODUCCIÓN

El Internet de las Cosas (IoT, siglas en inglés) es un concepto utilizado para describir la integración de objetos cotidianos con elementos de hardware para que estos puedan recibir y enviar datos desde y hacia el Internet, puede abarcar desde artefactos del hogar como bombillas eléctricas hasta dispositivos de salud, pasando por sistemas para la agricultura, industria y ciudades inteligentes[1].

A pesar de que el mundo de las telecomunicaciones crece rápidamente, la conectividad con el Internet o el acceso a puntos de energía eléctrica no está totalmente asegurado dentro de zonas urbanas o rurales. La tecnología LoRa da respuesta a estos problemas mediante la utilización de dispositivos de bajo consumo de energía y largo rango de comunicación. Su protocolo denominado LoRaWAN ya ha sido utilizado dentro de ciudades en asuntos como control de calidad del aire, monitoreo de bosques, control de inundaciones, entre otros[2], [3].

El presente proyecto busca aprovechar esta tecnología, y mediante el uso de sensores ultrasónicos obtener datos concernientes al nivel de ocupación de los contenedores de gran capacidad, para que esta información sea transportada a través de una red hasta un entorno de gestión donde se contará con un prototipo de aplicación de escritorio que recuperará dichos datos, los mostrará de forma clara al usuario y los almacenará en una base de datos.

La finalidad de esta aplicación es informar al usuario que tan lleno se encuentra un contenedor de gran capacidad, y así sea vaciado antes de que este colapse. Por otra parte, con la implementación de una base de datos se busca en un futuro obtener un registro histórico del ritmo al que se van ocupando los depósitos, para que las rutas de recolección puedan planificarse de acuerdo con el comportamiento de los habitantes de las zonas donde se encuentren según cada época o ubicación, en caso de que el prototipo sea implementado a mayor escala.

1.1 OBJETIVOS

Este proyecto tiene como objetivo general desarrollar un prototipo de aplicación para detectar el nivel de ocupación de contenedores de basura de gran capacidad utilizando la tecnología LoRaWAN.

Mientras que los objetivos específicos son:

- Analizar el funcionamiento de la tecnología LoRaWAN, de manera que facilite el desarrollo del proyecto.
- Implementar los servidores de The Things Network, para que los nodos sensores puedan comunicarse con Internet.
- Diseñar la aplicación de escritorio, base de datos e interfaz de usuario en base a encuestas y bosquejos previamente realizados.
- Implementar la aplicación de escritorio diseñada, mediante los entornos de desarrollo correspondientes.
- Evaluar el funcionamiento adecuado del sistema.

1.2 PLANTEAMIENTO DEL PROBLEMA

Las grandes ciudades del Ecuador como Quito, Guayaquil y Cuenca crecen a un ritmo acelerado debido a la migración interna que existe en el país [4]. Esto provoca una mayor demanda en la utilización de servicios públicos, como lo es el del sistema de recolección de basura. Solo en Quito diariamente se generan 2200 toneladas de basura [5], las cuales se recogen a través de distintos sistemas como el denominado Pie de Vereda o el de Recolección Mecanizada. Este trabajo de titulación se enfoca en el segundo método, el cual emplea contenedores de 2400 y 3200 litros de capacidad[6] distribuidos en diferentes sectores, donde los ciudadanos depositan desperdicios sólidos que son acopiados periódicamente por camiones con características específicas para esta tarea.

Este sistema fue implementado en la capital ecuatoriana en septiembre del 2015[7], sin embargo, tan solo un par de años después en el 2017, el 80% de los contenedores colapsó[8], representando un riesgo para la salud de los habitantes de la ciudad, además de afectar al ornato de esta. En un intento por mitigar este problema en 2019 EMASEO EP incrementó la cantidad de contenedores en distintas zonas, la ciudad llegó a tener 5300 depósitos de gran capacidad al finalizar dicho año[9], de todas maneras, los desbordamientos persisten hoy en día, e incluso han empeorado bajo el contexto de la pandemia por la COVID-19, ya que esta urbe generó 600 toneladas adicionales de basura durante esa época[10].

1.3 ALCANCE

El presente trabajo de titulación implementará un prototipo de aplicación de escritorio en un computador funcionando bajo el sistema operativo Windows. La finalidad de este aplicativo será recuperar los datos captados por dos sensores ultrasónicos funcionando en una red con la tecnología LoRaWAN, estos nodos serán ubicados en dos contenedores tipo, con dimensiones que se acerquen a los utilizados por la EMASEO EP, estas medidas son: 158cm de altura, 160cm de largo, 130cm de ancho y 2400 litros de capacidad.

En la Figura 1.1 se presenta el esquema del sistema, el cual incluye la red LoRaWAN, la conexión con los servidores TTN y un entorno de gestión.

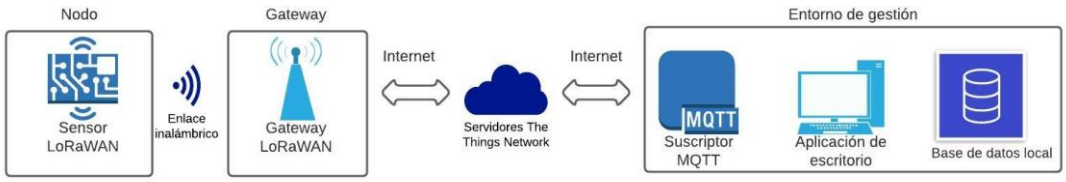


Figura 1.1 Esquema del sistema.

Se utilizarán dos nodos ultrasónicos LoRaWAN modelo DF703, como los que se muestran en la Figura 1.2, estos son elaborados por la empresa Dingtek, que cuentan con varios años de experiencia en el diseño y elaboración de sensores que funcionan bajo distintos protocolos. Una de las características principales de estos sensores es su consumo de potencia ultra bajo, el ciclo útil de un cambio de baterías puede durar más de 3 años con intervalos de medición de 4 horas según el fabricante del dispositivo. El protocolo en cuestión contempla una comunicación inalámbrica en distintas bandas ISM como 915MHz, 865MHz, 868MHz entre otras, en el presente trabajo de titulación se utilizará la banda definida por el ente regulador correspondiente estudiado en la Fase de Diseño. Estos nodos operan normalmente en temperaturas entre los -20ºC y 70ºC, lo cual es ideal considerando las condiciones ambientales de Quito, cuentan con una precisión de +/- 3cm respecto a la medición de altura y +/-2ºC como precisión de temperatura[11]. Estos sensores se implementarán en un ambiente que permita realizar pruebas y configuraciones que se ajusten a las necesidades del proyecto; para esto se utilizarán los contenedores tipo mencionados. Otra característica importante de estos sensores es que cuentan con certificación IP68[11], lo cual los hace resistentes ante cualquier partícula sólida y a la inmersión en líquidos que pueda afectar su funcionamiento [12].

Figura 1.2 Sensor ultrasónico LoRaWAN modelo DF703. [11]

Para el gateway originalmente se tenía planificado utilizar el LoRaWAN Gateway Core de la marca Microchip, pero finalmente se optó por utilizar el modelo Sentrius RG191 del fabricante Laird, el cual se muestra en la Figura 1.3. Este dispositivo se encarga de establecer una conexión segura con los nodos y reenviar los paquetes generados por estos hacia los servidores en la nube. En la sección Marco Teórico se realizará una descripción más profunda de estos elementos y se explicará las razones por las cuales se decidió utilizar un gateway diferente.

Figura 1.3 Gateway Laird Sentrius RG191. [13]

El estándar LoRaWAN contempla la utilización de un Servidor de Red (Network Server) y de un Servidor de Aplicación (Application Server) dentro de su arquitectura. Este trabajo de titulación utilizará la plataforma de The Things Network (TTN, por sus siglas en inglés) para este apartado del sistema. TTN provee de una serie de herramientas a través de su consola web para configurar estos servidores bajo parámetros de la tecnología LoRaWAN de manera gratuita a nivel de prueba o académico. Permite además la integración externa con plataformas como Amazon Web Services (AWS), Microsoft Azure

o Google Cloud y protocolos como MQTT (Message Queuing Telemetry Transport) o HTTP (Hypertext Transfer Protocol) mediante sus servidores de aplicación[14].

El presente trabajo de titulación consideró en un inicio la utilización de esta plataforma en su versión 2 (V2), sin embargo, TTN se encuentra en proceso de migración hacia la versión 3 (V3), la cual será implementada totalmente en diciembre de 2021[15], dejando obsoleta cualquier versión anterior. Por este motivo, y para evitar inconvenientes a futuro, se utilizará la última versión disponible, lo cual no implica una gran diferencia en comparación con el plan original.

Se desarrollará una aplicación con interfaz gráfica mediante el IDE Visual Studio 2019 basado en el lenguaje de programación C#, la cual permitirá obtener los datos desde el servidor de aplicación en la nube, esto se realizará mediante un suscriptor al bróker MQTT. De esta forma se accederá a distintos parámetros enviados por el nodo tales como su identificador, fecha y hora del envío de datos, Gateway desde donde se envían los datos, frecuencia de transmisión, canal utilizado, RSSI (Received Signal Strength Indicator), SNR (Signal Noise Ratio) y finalmente payload con los datos de interés.

El presente proyecto tendrá un producto final demostrable.

1.4 MARCO TEÓRICO

1.4.1 EL INTERNET DE LAS COSAS

El Internet de las Cosas (IoT) busca la integración de objetos comunes y dispositivos a redes privadas o al Internet para intercambiar información captada a través de sensores hacia otros dispositivos con nula o muy baja intervención humana [16]. Esto es más común de lo que se podría pensar, un teléfono celular concentra varios sensores, difiere dependiendo del modelo y marca, pero en su mayoría se puede encontrar lector de huellas digitales, sensor de luz, sensor de proximidad, acelerómetro[1], giroscopio[2], GPS (Sistema de Posicionamiento Global) y en casos especiales lector de iris o sensor de ritmo cardiaco[17]. Uno o varios de estos sensores pueden conectarse al Internet mediante aplicaciones específicas sin la necesidad de que el usuario intervenga [18].

Sin embargo, la idea de IoT es mucho más amplia que la de utilizar sensores en dispositivos móviles, podemos encontrar nodos especiales en el campo de la medicina, industria, agricultura y más. Esto ha llevado a la aparición de conceptos como ciudades

[1] Sensor utilizado para conocer la orientación del teléfono celular[17].
[2] Sensor utilizado para medir la rotación del teléfono celular[17].

inteligentes, industria inteligente, transporte inteligente, energía inteligente, monitoreo del medio ambiente, entre otros[19].

La arquitectura de los sistemas IoT está compuesta por 4 etapas, la primera comprende la adquisición de datos mediante sensores adaptados a objetos, la etapa siguiente es la red de comunicaciones, que reenvía los paquetes captados por los nodos hacia los servidores en el Internet, donde son procesados según la aplicación que tengan, y finalmente el último nivel es la de recuperación de datos a través de programas o apps[19]. La Figura 1.4 muestra dicha arquitectura.

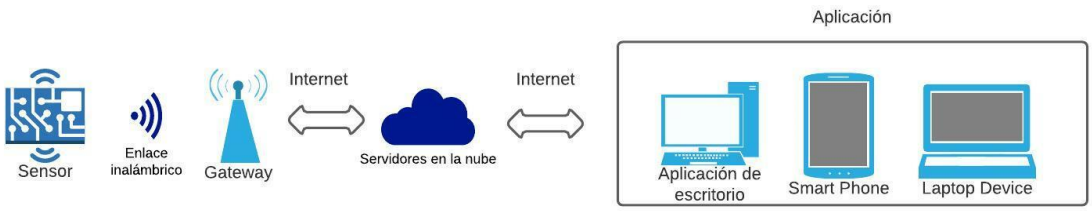


Figura 1.4 Arquitectura IoT.

Distintas tecnologías inalámbricas se ajustan al concepto del IoT, lo cual permite encontrarlas en escenarios como [20], [21] :

- Dispositivos usables o wearables: se refiere a relojes inteligentes, gafas de realidad virtual u objetos de rastreo que se pueden adaptar en zapatos o ropa. Es decir, dispositivos que las personas pueden utilizar como prendas de vestir o accesorios.
- Medicina: dispositivos relativos a este campo permiten monitorear la salud de los pacientes fuera de los hospitales, así se puede realizar un seguimiento del estado de las personas y asistirlos en caso de riesgo.
- Ciudades inteligentes: permite monitorear y gestionar problemas propios de ciudades con densidad poblacional alta.
- Agricultura: a través de sistemas IoT se puede medir la calidad y estado del suelo además de los cultivos.
- Industria: se puede adaptar sensores a distinta maquinaria industrial con el fin de obtener mediciones respecto a la calidad del producto que se fabrica, condiciones bajo las que funcionan las líneas de ensamblaje, entre otras.

En lo que respecta a las ciudades inteligentes el Internet of Things puede ayudar a gestionar, monitorear y solventar problemas en áreas como el transporte público, congestión vehicular, zonas verdes, medio ambiente y mantenimiento de espacios públicos. Dentro de las ventajas que estos sistemas representan están: sostenibilidad ambiental, mejor condición de vida para sectores vulnerables, ciudades ordenadas y atractivas para el sector turístico, ciudades eficientes en cuanto a movilidad y transporte público [22]. A continuación, se exponen casos donde IoT se ha aplicado dentro del concepto de las ciudades inteligentes [23]:

- En Copenhagen-Dinamarca se implementó una red de semáforos inteligentes, los cuales cambian su luz priorizando el transporte público y a los ciclistas. En un principio se utilizó únicamente 10 dispositivos, sin embargo, el impacto fue tal que decidieron adoptarlo a lo largo de la ciudad.
- En Ciudad del Cabo-Sudáfrica se implementó una red de cámaras de vigilancia dedicadas a reconocer las placas de los automóviles que circulan por suburbios con el fin de reducir la criminalidad en el sector, se reportó tiempo después que la tasa de delitos se redujo hasta en un 65%.
- En Buenos Aires-Argentina se implementó un sistema de detección de nivel de agua en los ríos de La Emilia, ya que en el 2017 se vio severamente afectada por inundaciones. Así, las autoridades pueden constantemente monitorearlos y prevenir desastres subsecuentes.

IoT es un concepto muy amplio, y como tal, está compuesto por distintos protocolos y estándares que se adaptan a cada necesidad o campo específico. Dentro de estas tecnologías se encuentran: LPWAN (Low Power Wide Area Networks), Redes Celulares (3G/4G/5G), Zigbee[3], Bluetooth, BLE (Blueetooth Low Energy), Wi-Fi, RFID (Radio Frequency Identification)[4][24].

Este trabajo de titulación utilizará la tecnología LoRa para el apartado de red de comunicaciones, la cual corresponde a la categoría de LPWANs, por lo que se pondrá especial énfasis en este tipo de redes. La Figura 1.5 muestra una comparativa de tasa de datos y consumo de energía vs alcance de comunicaciones de las tecnologías mencionadas, además de infografía de costos.

[3] Protocolo de comunicaciones LPWAN utilizado en dispositivos como tomacorrientes, bombillas, cerraduras entre otros[84].
[4] Tecnología de comunicaciones utilizada en etiquetas, cumplen una función similar a los códigos de barras[85].

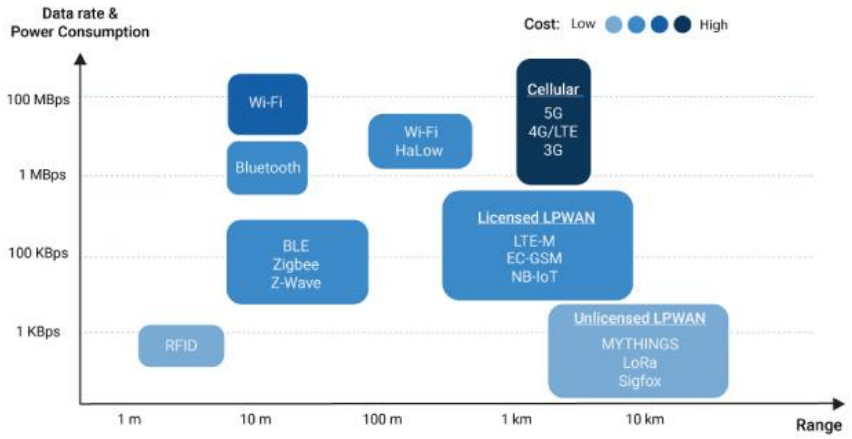


Figura 1.5 Tasa de datos vs alcance de comunicaciones tecnologías IoT[24].

Las Low Power Wide Area Networks (Redes de Área Extendida de Baja Potencia) o LPWAN, por sus siglas en inglés, son redes de telecomunicaciones inalámbricas diseñadas para transmitir tasas de datos reducidas en rangos amplios, con un consumo bajo de energía [25].

Dentro de las ventajas que presenta están: permite implementar redes con centenares de sensores distribuidos en zonas amplias, las baterías de los nodos pueden durar años, y dependiendo del fabricante, pueden ser recargables o fácilmente reemplazables, no requieren de infraestructura complicada o sistemas de cableado avanzados, lo cual las hace sencillas de llevar a la práctica, trabajan en bandas ISM(Industrial, Scientific and Medical)[5], por lo cual no es necesario pagar una licencia de uso de espectro radioeléctrico. Como toda tecnología, las LPWAN también presentan desventajas, dentro de estas se puede mencionar que no son útiles en caso de que se requiera transmitir grandes tasas de datos, como si pasa con las Redes Celulares o Wi-Fi, se puede mencionar también que dependiendo del lugar donde se ubiquen podrían presentar problemas de interferencia por la cantidad de dispositivos transmitiendo en la misma frecuencia al utilizar bandas no comerciales[25], [26].

[5] Bandas de frecuencia que se pueden utilizar sin licencia, varían según cada región[86].

Las tecnologías LPWAN que más se han desarrollado en los últimos años son:

- Sigfox: es propietaria, creada por la empresa que lleva el mismo nombre, la cual cuenta con una red propia que incluye estaciones base y dispositivos en diferentes países. Es necesario pagar una suscripción para utilizar su infraestructura [25].
- LoRaWAN: estándar creado por la LoRa Alliance, que está compuesta por empresas como Alibaba Group, Amazon, Cisco, Microsoft entre otras. A diferencia de Sigfox, la infraestructura debe ser desplegada y configurada de manera independiente, es decir, se debe gestionar a todo nivel [25], [27].
- NB-IoT (Narrow Band IoT): esta tecnología depende de las teleoperadoras de cada país, ya que su implementación está sujeta a la infraestructura de estas, tiene menor difusión que las anteriores ya que los proyectos bajo este estándar deben ser previamente aprobados por las empresas de telecomunicaciones de cada país. Tienen una complejidad elevada, debido a que se debe prevenir interferencia con las Redes Celulares GSM o LTE [25], [28].

Tabla 1.1 Comparación entre LoRaWAN, Sigfox y NB-IoT[29].

Tecnología / Parámetros	LoRaWAN	Sigfox	NB-IoT
Banda de frecuencia	433/868/915 MHz (ISM)	433/868/915 MHz (ISM)	Frecuencias licenciadas LTE
Ancho de banda	250 kHz y 125 kHz	100 Hz	200 kHz
Modulación	CSS	BPSK	QPSK
Tasa de datos	50 kbps	100 bps	200 kbps
Cantidad de mensajes por día	Ilimitado	140 (UL[6]), 4 (DL[7])	Ilimitado
Carga útil[8]	250 bytes	12bytes(UL), 8bytes (DL)	1600 bytes
Rango (zonas urbanas)	5km	10km	1km
Rango (zonas rurales)	20km	40km	10km
Resistencia a interferencias	Muy alta	Muy alta	Baja
Autenticación y encriptación	AES128	No	Encriptación LTE

[6] Enlace de subida, en IoT va desde el nodo hasta la estación base[87].
[7] Enlace de bajada, en IoT va desde la estación base hasta el nodo[87].
[8] En telecomunicaciones se refiere a la parte de la trama que contiene el mensaje real, se obtiene al excluir cabeceras, datos de control y otros[88].

1.4.2 LORAWAN

LoRaWAN es una tecnología de comunicaciones inalámbricas LPWAN que se viene desarrollando desde el 2009, sus creadores buscaban crear un protocolo que permita transmitir a grandes distancias sin utilizar demasiada potencia. Desde esa época el enfoque era proveer soluciones que obtengan métricas en industrias como la del gas, agua y electricidad de manera inalámbrica. Con este fin adoptaron como técnica de modulación a CSS (Chirp Spread Spectrum), que ya era utilizada en tareas militares, específicamente en sonares marítimos y de aviación. Para el 2012, Semtech adquirió Cycleo, la empresa a cargo de desarrollar LoRaWAN en sus primeros 3 años. Esto permitió que se elaboren microprocesadores para dispositivos y gateways, entre una serie de protocolos de capa de red, tramas y seguridad. En el 2015 se formalizó la LoRa Alliance, que reúne varios miembros y socios alrededor del mundo interesados en esta tecnología, en ese año se bautizó el protocolo como LoRaWAN [30].

IoT es muy amplio, por lo tanto, agrupa varias tecnologías y estándares de comunicaciones inalámbricas bajo su idea principal, que es la de conectar cosas, sin embargo, no establece un modelo de referencia[9] único en el cual todas puedan basarse, debido a esto cada protocolo difiere uno del otro[31]. Como muestra la tabla 1.1, a pesar de que LoRaWAN, IoT y Sigfox son tecnologías LPWAN, no son del todo similares en cuanto a especificaciones, existiendo en ocasiones amplias diferencias, como por ejemplo en el tamaño de la carga útil o velocidad de transmisión. La diferencia es aún mayor cuando son comparadas con otras tecnologías como Wi-Fi, Bluetooth, RFID, que según se apliquen, pueden ser consideradas IoT.

1.4.2.1 Modelo de Referencia LoRaWAN

El modelo de referencia elaborado por la Lora Alliance para LoRaWAN está compuesto desde su capa inferior hasta su capa superior por[31]:

- Capa de Modulación Lora: definida por Semtech.
- Capa LoRa MAC: definida por Semtech y la Lora Alliance.
- Capa LoRaWAN: definida por la LoRa Alliance.
- Capa de Aplicación: definida por el usuario final.

La Figura 1.6 muestra este modelo de referencia.

[9] Los modelos de referencia en el mundo de las redes de información son ideas abstractas que se utilizan para estandarizar dispositivos de distintos fabricantes en una misma tecnología[89].

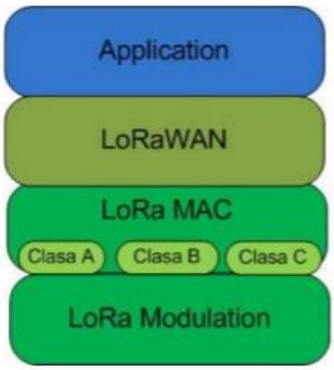


Figura 1.6 Modelo de referencia LoRaWAN [31].

1.4.2.2 Topología y arquitectura LoRaWAN

La topología de red LoRaWAN se considera una estrella de estrellas, ya que los dispositivos finales o nodos se pueden conectar a uno o varios gateways al mismo tiempo, y estos se conectan al mismo servidor de red, donde aunque reciban varias veces el mismo paquete, se procesa tan solo uno de ellos. Los gateways utilizan protocolos TCP/IP[10] y actúan como un puente, el cuál adapta los paquetes de los dispositivos finales en estos protocolos de Internet en los dos sentidos (enlace de subida y bajada). Las características de esta arquitectura permiten que exista un solo salto entre nodos y gateways, define especificaciones para garantizar la conexión entre dispositivos de distintos fabricantes, fijando características técnicas, pero dejando libertad para que las marcas desarrollen sus productos de manera pública, privada o a nivel empresarial [32].

Los Join Servers se encuentran integrados en el servidor de red y proveen seguridad a la infraestructura LoRaWAN, para esto almacenan las claves de acceso de los dispositivos y generan otras para acceder de manera externa al servidor de aplicación. Estos Join Servers reciben las solicitudes desde el resto de los servidores y las acepta si verifica que estás son válidas y funcionales[33].

Esta arquitectura se muestra en la Figura 1.7.

[10] Familia de protocolos utilizados en redes de información[90].

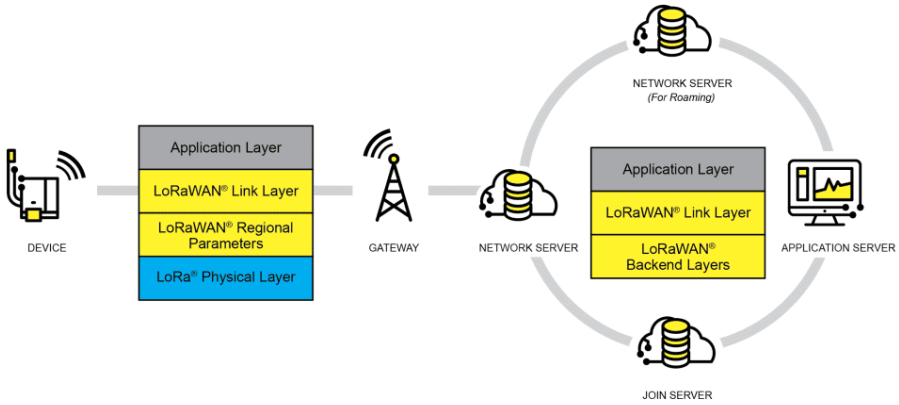


Figura 1.7 Arquitectura LoRaWAN[32].

1.4.2.3 Chirp Spread Spectrum (CSS)

LoRaWAN utiliza una técnica de espectro ensanchado[11] (SS) como técnica de modulación, la cual se denomina Chirp Spread Spectrum (CSS), Chirp es un acrónimo en inglés que significa Compressed High Intensity Radar Pulse (Pulso de Radar de Alta Intensidad). Cada chirp representa una portadora. A lo largo de la comunicación la amplitud de la señal se mantiene constante y se utiliza el ancho de banda completo de extremo a extremo durante un periodo de tiempo. La frecuencia puede variar, e ir de un valor menor a uno mayor, denominado up-chirp, o de un valor mayor hasta uno menor, conocido como down-chirp[34]. La Figura 1.8 muestra un ejemplo de up-chirp en el cual la amplitud de la señal se mantiene constante.

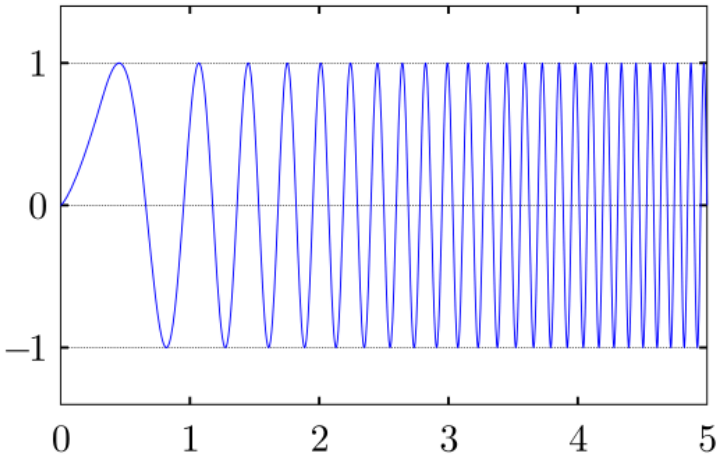


Figura 1.8 Proceso de up-chirp en una señal[34].

[11] Técnicas de modulación utilizadas en comunicaciones digitales.

1.4.2.4 CSS aplicado en LoRaWAN

CSS es resistente al efecto Doppler[12], se utiliza para transmitir pequeñas porciones de datos en dispositivos que utilizan poca potencia en distancias largas, lo cual lo hace ideal para esta tecnología. Los anchos de banda que LoRaWAN utiliza son 125kHz, 250kHz y 500kHz, la cantidad de portadoras y la velocidad de transmisión es determinada por SFs (Spreading Factors)[34].

LoRa utiliza 6 Spreading Factors numerado desde SF7 hasta SF12. La influencia de estos factores de propagación afecta directamente la velocidad de transmisión, consumo de batería y sensibilidad del receptor. A menor SF se tendrá portadoras más rápidas, tasas de transmisión mayores pero un menor rango de cobertura ya que se reduce los tiempos de procesamiento, y cada vez que el SF aumenta la velocidad se reduce a la mitad, al igual que la cantidad de datos transmitidos. Estos factores de propagación son ortogonales, lo cual impide que cuando las señales transmitidas interfieran entre sí [35]. La tabla 1.2 muestra como el ancho de banda afecta a la velocidad de transmisión utilizando un mismo factor de propagación[35].

Tabla 1.2 Tasas de transmisión para SF7 y distinto AB[13][35].

Spreading Factor (SF)	Ancho de banda (kHz)	Velocidad de transmisión (kbps)
7	125	5.5
7	250	10.9
7	500	21.9

Por lo general, LoRaWAN utiliza factores de propagación mayores cuando las condiciones de transmisión son adversas y la señal es débil. Un mayor SF también implica mayor sensibilidad en los receptores. En cuanto al tiempo de vida de las baterías del dispositivo, estas sufrirán mayor desgaste y descarga si se usa SFs altos, ya que la conexión se mantendrá activa durante más tiempo[35].

[12] Desplazamiento de las señales causado por la variación en frecuencia.
[13] Acrónimo en español para ancho de banda.

1.4.2.5 Dispositivos finales y clases

Los dispositivos finales LoRaWAN también se conocen como motas o nodos, y son sensores, actuadores o ambos, cuya fuente de energía es una batería de uso único o recargable. Se conectan a los concentradores de dispositivos o gateways de manera inalámbrica, en general, deben ser capaces de transmitir a grandes distancias sin consumir demasiada energía. Su objetivo es recolectar información para el usuario como calidad del aire, mediciones a través señales ultrasónicas, temperatura, luminosidad, proximidad, movimiento, calidad del suelo, nivel de agua o sustancias líquidas en un contenedor entre otros. Existen distintas opciones según la necesidad de la red. De todas maneras, estos dispositivos finales también pueden comportarse como actuadores para realizar acciones o intervenir en un sistema mecánico. En la tecnología LoRa se utilizan para puertas eléctricas, cerraduras eléctricas, motores eléctricos y accionamientos de peine, lo cual ha permitido que sea aplicada en la industria y sus procesos[36].

LoRaWAN define 3 clases de dispositivos finales, cada una está elaborada para satisfacer distintas necesidades de los campos de aplicación del estándar. Estas clases son[32],[37]:

- Clase A: todos los dispositivos finales deben implementar esta clase, es la que menos potencia utiliza, de comunicación bidireccional, mayor latencia[14] y es completamente asíncrona[15]. Los nodos Clase A poseen un modo de ultra bajo consumo de energía, al cual entran durante periodos configurados, una vez que este lapso culmina el dispositivo inicia la conexión inalámbrica estableciendo un enlace de subida seguido por dos ventanas de transmisión para el enlace de bajada con un retraso entre estas, el servidor de red puede utilizar uno de estos dos periodos para responder, pero no los dos. Los sensores Clase A generalmente se aplican en monitoreo ambiental, localización, detección de fuego, detección de fugas de agua, detección temprana de terremotos. La Figura 1.9 muestra las ventanas de comunicación de esta clase.

[14] Tiempo que tarda en viajar un paquete de datos de un extremo a otro.
[15] Comunicación digital en la cual no interviene un reloj entre los extremos.

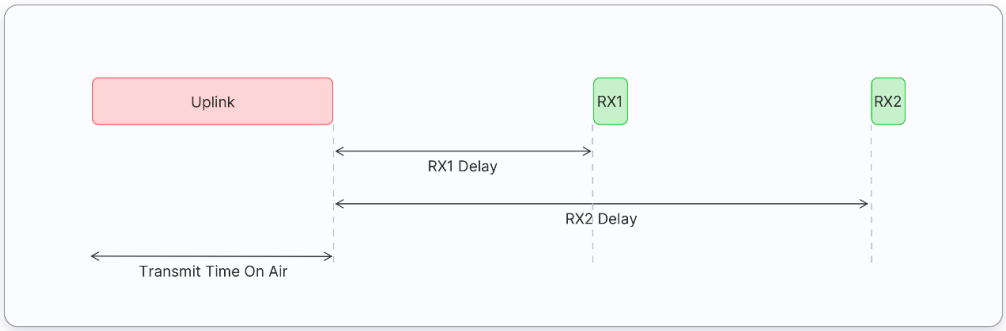


Figura 1.9 Ventanas de comunicación Clase A [37].

- Clase B: esta clase de dispositivos finales añade periodos programados a la comunicación para que el servidor utilice el enlace de bajada. La conexión es sincrónica y utiliza beacons al inicio y al final para controlarla, el tiempo entre estos se conoce como beacon period, y los tiempos donde los nodos se habilitan para recibir mensajes en el enlace de bajada se denomina ping slots. Una de las características de esta clase es su baja latencia en comparación con la Clase A, ya que los espacios para transmitir están preestablecidos y no es necesario enviar un mensaje en el enlace de subida para recibir uno en el de bajada, sin embargo, el consumo de potencia es mayor ya que pasan más tiempo activos, por ende, la batería de los dispositivos sufrirá un mayor desgaste. Generalmente esta clase se utiliza en motas de medición inteligente o temperatura. La Figura 1.10 muestra las ventanas de comunicación correspondientes.

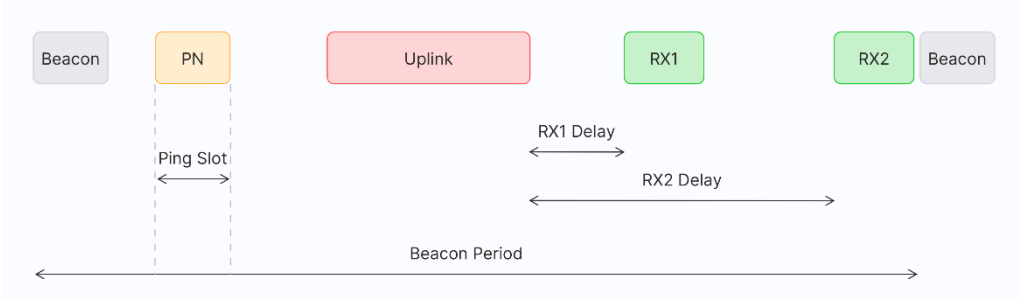


Figura 1.10 Ventanas de comunicación Clase B [37].

- Clase C: en esta clase de dispositivos las ventanas de recepción se mantienen abiertas únicamente si el dispositivo no se encuentra transmitiendo, de esta manera se asegura una latencia mínima, pero con un alto consumo de energía en comparación con los nodos Clase A. Estas motas necesitan una fuente de alimentación continua, y se utilizan en semáforos o aplicaciones que requieren constantemente medir. La Figura 1.11 muestra las ventanas de comunicación de esta clase.

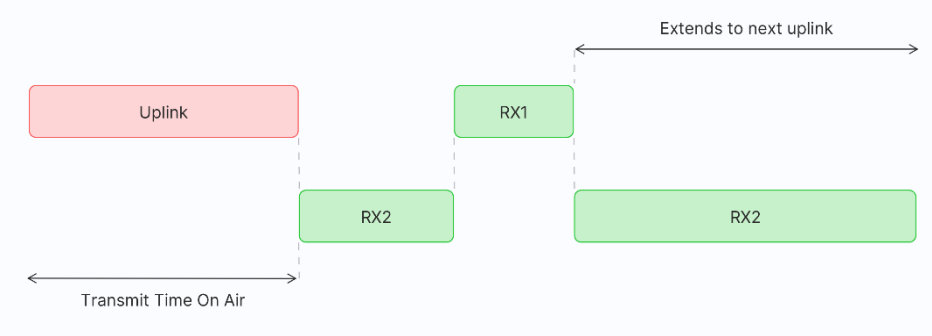


Figura 1.11 Ventanas de comunicación Clase C[37].

1.4.2.6 Modos de activación de los dispositivos finales.

Como muestra la Figura 1.6, el servidor de red incorpora un Join Server en la arquitectura LoRaWAN, y una de sus funciones es la de intercambiar claves con los dispositivos finales para autorizar su conexión con la red. Para esto, la tecnología LoRa especifica dos modos de activación, dependiendo de cual de estos se elija para el sistema se utilizará más o menos claves, y, por ende, la seguridad será robusta o débil. Estos modos son OTAA (Over-The-Air Activation) y ABP (Activation By Personalization).

En la versión 1.1 del estándar, para el modo de activación OTAA, se intercambian dos tipos de mensajes MAC. El mensaje Join-request va desde el dispositivo final al Join Server, mientras que el Join-accept va en el camino contrario. Los pasos de este modo se describen a continuación[38]:

1. Join Request: el proceso es iniciado por el nodo, este envía un mensaje de solicitud para incorporarse a la red. Los campos que se incluyen en este mensaje son DevEUI, JoinEUI y DevNonce. El DevEUI [8 bytes] es un identificador único del dispositivo dentro de la red, usualmente el fabricante asigna esta dirección, sin embargo, el usuario puede establecer su valor siempre y cuando se encuentre en el formato de espacio de direcciones IEEE EUI64. El JoinEUI [8 bytes] sigue el mismo formato que el DevEUI, y es un identificador de aplicación global utilizado para conectarse con el autenticador en el servidor de red. El DevNonce [2 bytes] es un valor auto incrementable que inicia en 0 y aumenta de uno en uno cada vez que se realiza un Join Request. Este mensaje se envía sin encriptación, y se utiliza cualquier canal disponible de la frecuencia configurada en la red.
2. Reenvío de mensaje: el servidor de red envía el mensaje Join Request hasta el Join Server correspondiente.
3. Proceso de mensaje: el Join Server analiza el mensaje recibido en el paso 2, y si se comprueba que es correcto, genera 3 claves de sesión.

4. Join Accept: una vez que las claves son generadas el servidor de red elabora un mensaje Join Accept, este consta de campos como: JoinNonce, NetID, DevAddr, DLSettings, RxDelay, CFList. El JoinNonce [1 byte] es un campo que contiene las claves generadas por el Join Server en el paso anterior, las cuales deben ser informadas al dispositivo final. El NetID [3 bytes] es el identificador de la red. El DevAddr [4 bytes] es un valor generado por el servidor de red y se utiliza para identificar al dispositivo dentro de la red. El DLSettings [1 byte] campo que guarda la información respecto a las configuraciones que debe usar el dispositivo en el enlace de bajada. El RxDelay [1 bytes] guarda la información del retraso que debe existir entre las ventanas de transmisión y recepción. El CFList [16 bytes] es un campo opcional que contiene una lista de los canales que el dispositivo final puede usar según la configuración regional. Este mensaje se encripta mediante la AppKey.
5. Reenvío de claves: el Join Server envía las claves de sesión generadas en el paso 2 hasta el servidor de red.
6. Desencriptación del mensaje: finalmente el dispositivo final desencripta el mensaje Join Accept del paso 4 utilizando una operación AES mediante los valores de la AppKey, NwkKey y JoinNonce. Si el proceso se completó correctamente el dispositivo se encontrará conectado a la red al finalizar este paso.

Este proceso se muestra en la Figura 1.12.

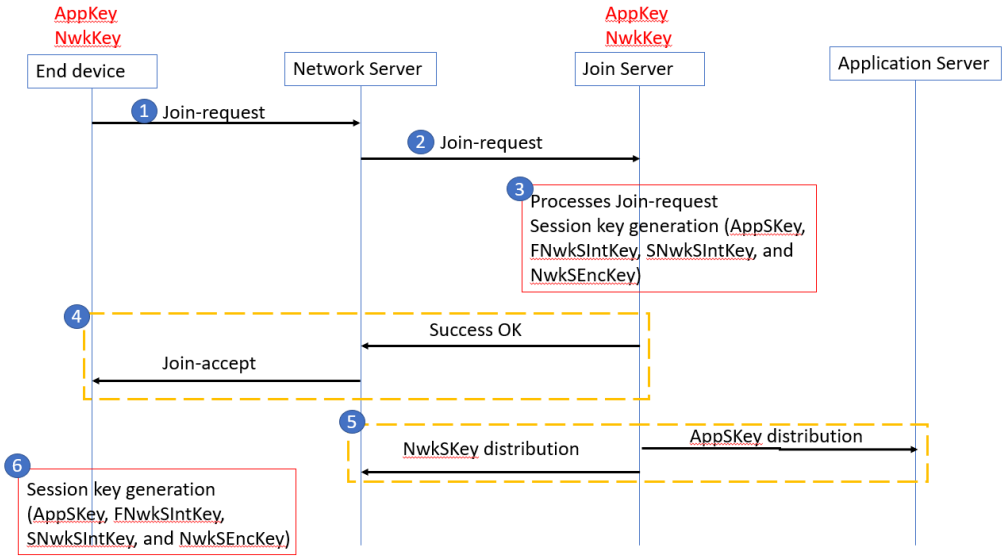


Figura 1.12 Proceso del modo de activación OTAA para LoRa 1.1 [38].

El modo de activación ABP es más sencillo, ya que involucra menos claves de autenticación. Generalmente este modo de activación se utiliza únicamente para pruebas. Para este modo los dispositivos finales no utilizan las direcciones DevEUI, JoinEUI y AppKey. Las claves que se generaban por el Join Server en el paso 3 del modo OTAA se almacenan directamente en el dispositivo final, mientras que el servidor de red y servidor de aplicación almacenan la AppSKey entre otras. La intervención del Join Server es inexistente en este modo[38], [39]. El procedimiento se muestra en la Figura 1.13.

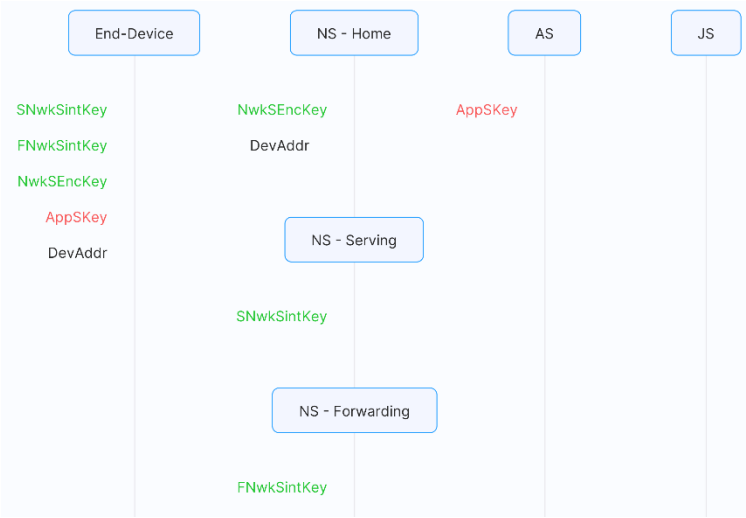


Figura 1.13 Proceso del modo de activación ABP [38].

1.4.2.6 Gateways LoRaWAN

Los gateways o concentradores dispositivos se encargan de conectar los nodos al servidor de red en el Internet, para esto utilizan protocolos TCP/IP mediante conexiones Wi-Fi, Ethernet o incluso redes celulares. Un mismo dispositivo final puede conectarse a varios gateways a la vez, pero una vez que el servidor de red reciba estos mensajes procesará solo uno de ellos. Se puede considerar estos dispositivos como enrutadores de la red LoRa. Pueden implementarse físicamente mediante componentes de hardware o virtualmente mediante sistemas operativos, de cualquier manera, los dos cumplen la misma función dentro de la red, sin embargo, los que se basan en software brindan mayor libertad de configuración al usuario[40]. La implementación de uno o varios gateways en una red depende del nivel de interferencia en la zona donde se instalen, uno de estos puede concentrar varios dispositivos a la vez, pero se conectarán únicamente a un solo servidor de red, por esto es por lo que se considera a la arquitectura LoRaWAN como una estrella de estrellas. La Figura 1.3 muestra un Gateway de la marca Laird.

1.4.2.6 Trama física LoRaWAN

Esta trama está compuesta por un preámbulo que se utiliza para la sincronización y contiene información referente a la modulación, de esta forma se puede diferenciar cada paquete y su forma de modularse de manera separada para cada trama. También facilita el tratamiento de datos para el gateway o concentrador de dispositivos, ya que le permite adaptar el consumo de energía y rendimiento para nodos que se encuentren en movimiento. El último byte de este preámbulo contiene evita que otras redes LoRaWAN que se encuentren transmitiendo en la misma frecuencia interfieran entre sí, de esta manera los dispositivos finales solo se comunicarán con otros elementos de la red que tengan su mismo byte de sincronización [41].

El payload de la trama física puede tener un tamaño máximo de 255 bytes, lo cual es suficiente para informar los datos capturados por los sensores así como otros de posicionamiento global, estado y configuración del dispositivo y otros definidos por los fabricantes [41].

1.4.2.7 Limitaciones del protocolo

LoRaWAN es un estándar amplio, sin embargo, no siempre cubre las necesidades de toda aplicación. Una de las principales limitaciones es la cantidad de datos que se puede enviar mediante este tipo de redes, sería impensable utilizarlas para enviar audio, imágenes o video. Tampoco es ideal para tareas que requieran monitoreo en tiempo real o muy baja latencia. A pesar de que es un protocolo IoT no se puede utilizar para aplicaciones como control de luces, interruptores o tomacorrientes como lo hacen ZigBee o Bluetooth[42], [43].

1.4.3 THE THINGS NETWORK (TTN)

The Things Network es una plataforma de código abierto[16] construida por la comunidad interesada en el Internet de las Cosas, específicamente en el estándar LoRaWAN. Permite a través de su infraestructura configurar servidores de red y aplicación que se ajusten a las necesidades de la red, ya que proveen cobertura a nivel mundial. TTN funciona sobre la plataforma de The Things Stack Community (TTS), la cual tiene como objetivo ser una plataforma abierta y descentralizada de servidores LoRa. Permite la integración con varias aplicaciones y protocolos como MQTT y HTTP. Cuenta con

[16] Software que no requiere una licencia para utilizarse y cuyo código está disponible [91].

clusters[17] en Oceanía, Europa y Norte América, permitiendo reducir tiempos de comunicación dependiendo de la región donde se encuentre implementada la red LoRaWAN[44].

1.4.3.1 Consola TTN

The Things Network Console es una interfaz en línea que permite controlar y monitorear los dispositivos de la red LoRaWAN, además de facilitar el manejo de conexión con aplicaciones externas mediante protocolos o plataformas IoT[45]. A través de esta interfaz además se puede configurar los distintos parámetros de los servidores de red y aplicación del sistema, y hacer un seguimiento de los paquetes que se envían desde el o los gateways.

1.4.3.2 Arquitectura TTN

La Figura 1.6 muestra la arquitectura LoRaWAN, la cual en uno de sus extremos implementa 4 servidores con distintos propósitos. La arquitectura de The Things Network en su V3 es la misma, pero además implementa un servidor de identidad y la consola descrita anteriormente para mayor seguridad y administración de este segmento del sistema [46]. Esta arquitectura se muestra en la Figura 1.14.

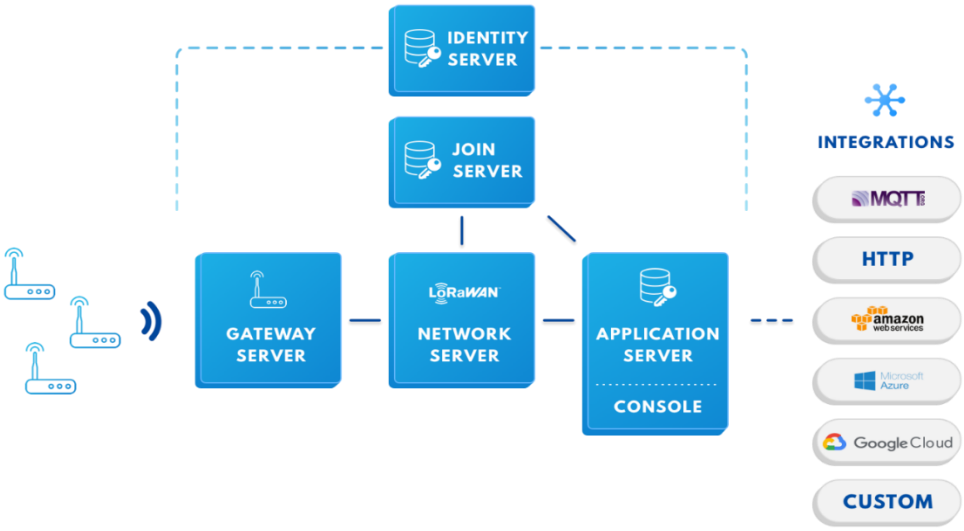


Figura 1.14 Arquitectura de TTN[46].

Las funciones del servidor de aplicación son manejar esta capa de la arquitectura LoRaWAN, aquí se desencripta y decodifica el payload que se recibe en los mensajes del enlace de subida, además de codificar y encriptar los mensajes del enlace de bajada que

[17] Conjunto de servidores que se comportan como si fuera uno solo [92].

pudiese generar el usuario. En este servidor se gestiona la integración con plataformas externas como Google Cloud, Amazon Web Services o Microsoft Azure, además de la conexión a aplicaciones personalizadas que el usuario pudiese desarrollar mediante protocolos como MQTT y HTTP[47].

El Gateway Server que se ilustra en la Figura 1.14 se encarga de mantener conectados los gateways a la red y reenvía los paquetes hacia el servidor de red, además manejar los flujos de tráfico en los enlaces de subida y bajada, para evitar colapsos, mal uso de las frecuencias y controla que estas se encuentren dentro de las regulaciones regionales configuradas [48].

El Identity Server o Servidor de Identidades almacena registros de los dispositivos finales, gateways, usuarios, autenticación y aplicaciones involucradas en la red. Estos registros incluyen información como nombres, descripción y otros atributos[49].

El Join Server, como se discutió anteriormente en este documento, autoriza o rechaza la conexión de dispositivos a la red mediante distintas claves generadas por el usuario o por los equipos involucrados en la red LoRaWAN.

Por otra parte, el Servidor de Red, como también se ha discutido en este trabajo de titulación, es un elemento del sistema LoRaWAN que maneja los dispositivos en sus enlaces de subida y bajada además de reenviar mensajes a los servidores de aplicación y Join servers.

1.4.3.3 Integración con aplicaciones y protocolo MQTT

A lo largo de esta sección se ha indicado distintas integraciones que ofrece The Things Network mediante sus servidores de aplicación. A continuación, se realizará un estudio más extenso del protocolo MQTT, ya que este será utilizado para conectarse con la plataforma mediante la aplicación de escritorio.

Una de las principales características de los servidores de aplicación de TTN es que integran un bróker MQTT el cuál es accesible mediante distintos clientes de diferentes sistemas operativos o plataformas. Este bróker publica los mensajes del enlace de subida que envían los dispositivos finales de la red.

MQTT es un protocolo utilizado en redes del Internet de las Cosas ya que es simple y se ajusta a las necesidades de estas tecnologías en cuanto a consumo de potencia y ancho de banda. MQTT son las siglas para Message Queing Telemtry Transport, se considera un protocolo M2M (machine-to-machine) donde la conexión se mantiene todo el tiempo hasta que el cliente la de por terminada, a diferencia de por ejemplo HTTP, que debe

realizar un proceso de conexión cada vez que se desee enviar datos. El intercambio de mensajes en MQTT se da a través de un sistema publicador-suscriptor, donde cada paquete se organiza dentro de un tópico o tema. Sin embargo, esta conexión no se da de manera directa, sino que atraviesa un bróker el cual filtra los temas del publicador y se los envía al suscriptor correspondiente de cada categoría, los puertos que ocupa MQTT son 1883 y 8883 si funciona con TLS.[50]. En las redes LoRaWAN los dispositivos finales toman el rol de publicador de mensajes, el bróker del sistema, en el caso de TTN, se encuentra en el servidor de aplicación, mientras que el suscriptor puede ser configurado por el usuario de distintas maneras, como en el presente proyecto, a través de una API para el lenguaje C#. Esta conexión se muestra en la Figura 1.15.

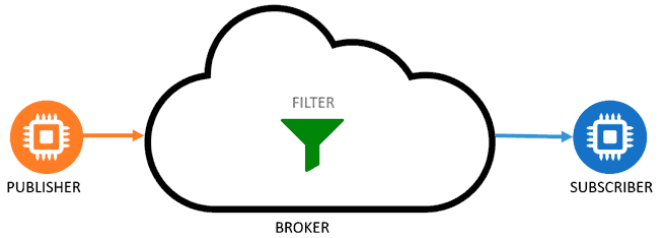


Figura 1.15 Proceso de comunicación en MQTT[50].

Al establecer la conexión entre publicador y bróker el cliente envía una solicitud CONNECT, la cual almacena información como usuario, clave, identificación, tópico, entre otros. El bróker responderá a este mensaje aceptándolo o rechazándolo mediante un CONNACK. Los paquetes del publicador se difunden mediante mensajes PUBLISH, que únicamente contienen el tópico y la carga útil o payload. Mientras que para la suscripción el cliente emplea mensajes SUBSCRIBE o UNSUSCRIBE, según sea el caso, a lo que el bróker responde con SUBACK o UNSUBACK respectivamente. Ya que la conexión se encuentra activa todo el tiempo el cliente constantemente se envía mensajes de PINGREQ al servidor, quién responde con mensajes PINGRESP [50].

Para ofrecer robustez MQTT maneja tres niveles de QOS (Calidad de Servicio), sin embargo, TTN únicamente soporta el primer nivel[50], [51]:

- QoS 0 unacknowledged: en este nivel el mensaje se envía máximo una vez, si la comunicación falla lo más probable es que este no se entregue.

1.4.3.4 TTN frente a otros servidores LoRaWAN

Como es de esperarse, The Things Network no es la única plataforma que ofrece las herramientas necesarias para incorporar servidores LoRa a la red. En esta sección se hará una breve revisión de las alternativas a TTN, además de una comparación.

Everynet al igual que TTN ofrece una plataforma para configurar elementos de red LoRaWAN como servidores de red, aplicación entre otros, sin embargo, se enfoca en ofrecer sus servicios y soporte técnico a las empresas de telecomunicaciones que estén interesadas en incorporar este tipo de redes a sus sistemas. Se considera un modelo de red pública[52].

ThingPark pertenece a la empresa Actility, compañía que se convirtió en uno de los fundadores de la LoRa Alliance. Al igual que Everynet, ThingPark ofrece servidores de red a empresas de telecomunicaciones, especialmente a aquellas que utilizan gateways de los fabricantes Cisco, Kerlink o Multitech [52].

ChirpStack, o anteriormente conocido como LoRa Server, es un software de código abierto que se asemeja bastante a TTN, ya que cuenta con servidores como el de red, aplicación o Join Server. Su principal diferencia reside en que requiere mayor atención en lo que refiere a configuración e instalación, ya que se implementa mediante un terminal en sistemas operativos basados en Linux gracias a repositorios ubicados en el Internet. En cuanto a integración con aplicaciones externas, también soporta protocolos como MQTT o HTTP, además de sistemas como Amazon Web Services, Google Cloud entre otros. Se considera una alternativa para personas con conocimiento previo en el área de LoRaWAN o de las Tecnologías de la Información[52].

1.4.4 ELEMENTOS DE RED LORAWAN

En el alcance de este trabajo de titulación se describió brevemente los dispositivos de red que se utilizarán para implementar la red LoRaWAN, en esta sección se hará un estudio más profundo de ellos, se los comparará con otro dispositivo y además se explicará el cambio de Gateway utilizado en relación con el plan de titulación originalmente elaborado.

1.4.4.1 Gateway Sentrius Laird RG191

El Laird RG191 pertenece a la familia de dispositivos RG1xx, los cuales son fabricados específicamente para trabajar dentro de redes LoRaWAN. Incorpora conectividad Wi-Fi y Ethernet para conectarse al Internet, aunque también cuenta con la opción de añadir la capacidad de comunicarse mediante redes LTE [53]. Las especificaciones de este producto se describen en la tabla 1.3.

Tabla 1.3 Características de Sentrius Laird RG191[54].

Especificación	Descripción
Chipset LoRa	Semtech SX1301
Arreglo de antenas Wi-Fi	2x2 MIMO[18]
Frecuencias Wi-Fi	2.4 y 5 GHz
Protocolos Wi-Fi	802.11 a/b/g/n (2.4 y 5 GHz)
Frecuencias LoRa por región	US: 902-928 MHz (UL/DL)
	EU: 863-870 MHz (UL/DL)
Red cableada	Ethernet con conector RJ-45
Seguridad	Estándares: WEP, WPA, WPA2
	Encriptación: WEP, TKIP, AES
Sistema Operativo	Linux embebido Kernel 4.x
LoRa Forwarders (protocolos de reenvío)	TTN con forwarder Semtech, forwarder UDP, Chirpstack con forwarder UDP, Senet legacy
Temperaturas de operación	-30º a 70º C
Rango de cobertura	Más de 16km
Potencia de transmisión	27 dBm
Certificación IP	IP[19]67 (versión para exteriores)

Para el presente trabajo de titulación se utilizará la versión del RG191 para interiores sin conectividad LTE, el cual se muestra en la parte izquierda de la Figura 1.16.

Figura 1.16 Laird RG191 para interiores (izq.) y para exteriores (der.) [54].

[18] Acrónicmo para Multiple Input Multiple Output, y es un arreglo de antenas que sirve para mejorar el rendimiento de un sistema inalámbrico [93].
[19] Acrónimo para Ingress Protection, es una certificación que garantiza que el dispositivo es resistente a partículas o líquidos, dicha protección se especifica a través de los números que prosiguen a las siglas IP [94].

1.4.4.2 Dispositivos finales Dingtek DF703

Este nodo se desarrolló específicamente para obtener información sobre la cantidad de ocupación de los depósitos de basura de distintos tamaños, tanto para interiores como para exteriores, también incorpora sensores de temperatura y ángulo de inclinación [11].

Para este proyecto de titulación se utilizará la versión del dispositivo que contiene un módulo de comunicaciones LoRaWAN, sin embargo, está disponibles para otros como NB-IoT o Sigfox[11]. La tabla 1.4 muestra las especificaciones más relevantes respecto a este dispositivo.

Tabla 1.4 Especificaciones del dispositivo final Dingtek DF703[11].

Dimensiones	Altura: 115mm
	Largo: 115mm
	Profundidad: 40mm
Peso	150g
Material de la carcasa	ABS (Acrilonitrilo Butadieno Estireno)
Tipo de sensor	Ultrasónico a 40kHz
Rango de detección	4 metros
Precisión de medición	+/- 3cm
Precisión de temperatura	+/- 2ºC
Chipset LoRa	Semtech 1276
Frecuencias	US: 902-928 MHz
	EU: 863-870 MHz
Modo de activación	Clase A (OTTA por defecto)
Rango de cobertura	3km en línea de vista
Batería	De litio a 3.6V 8500mAh no recargable
Tiempo útil de la batería	Mas de 3 años en intervalos de medición de 4 horas
Temperatura de operación	-20º a 70º C
Certificación IP	IP68

La Figura 1.3 muestra las dos caras de este sensor.

1.4.4.3 Comparación de gateway y nodos con otros dispositivos

En una investigación previa al desarrollo de este trabajo de titulación se encontró varios dispositivos de red LoRaWAN que pudieron ser útiles para el desarrollo del proyecto, sin embargo, se eligió los anteriormente descritos. En esta sección se realizará una comparación de estos elementos y se explicará porque se realizó el cambio de equipo en el gateway.

Para el gateway se puso a consideración dos equipos que se encontraban al alcance, el primero de estos fue el LoRaWAN Gateway Radio + Core Board, del fabricante Microchip, que es parte del LoRa Technology Evaluation Kit – 800. El autor del presente trabajo de titulación ya había trabajado previamente con este concentrador de dispositivos para un proyecto distinto bajo la tecnología LoRa durante julio y diciembre del 2020, por lo cual fue la primera opción que se manejó y propuso en el plan original.

Sin embargo, en el proceso de investigación y desarrollo del actual proyecto, se encontró que este gateway funciona exclusivamente con dispositivos finales que incorporen módulos RN, que son microprocesadores LoRa fabricados por la misma empresa. Esto resultaba un gran inconveniente, ya que a pesar de que se puede adquirir uno de estos módulos y adaptarlo a un circuito que obtenga la información de los contenedores esto implicaba un mayor enfoque en la parte de red, y considerando que este trabajo no se concentra totalmente en aquel aspecto hubiese representado un retraso en el desarrollo de este. Otro inconveniente encontrado fue que el fabricante no provee información suficiente sobre las características técnicas del gateway, por ejemplo, en comparación con el Laird Sentrius RG191, no se encontró protocolos de seguridad, protocolos de reenvío, rangos de cobertura, temperatura de operación entre otros. Además, al ser un gateway que se divide en la parte de Core, que maneja los paquetes LoRa, y la parte de Radio, que maneja las comunicaciones inalámbricas, era necesario adquirir un módulo que trabaje dentro de los 902 – 9028 MHz, correspondientes a las bandas ISM de nuestra región, ya que únicamente se contaba con uno que trabaja en los 863 – 870 MHz.

Parte de la información detallada en el párrafo anterior se puede encontrar en el manual de usuario del fabricante [55]. En las Figuras 1.17 y 1.18 se muestra el Evaluation Kit – 800 y el chip RN2903, parte de la familia RN de Microchip.

Figura 1.17 Gateway Microchip LoRa Technology Evaluation Kit – 800. Nodos RN2483 (izq.) y LoRaWAN Gateway Core + Radio Board (der.)[56].

Figura 1.18 Chipset RN2903 de Microchip [57].

Por lo anterior expuesto se decidió utilizar el Laird Sentrius RG191.

Para los nodos se consideraron dos opciones: Dingtek DF703 y NETOP Waste Bin Sensor. En la tabla 1.5 se comparan las características técnicas de estos dispositivos [11], [58], [59]

Tabla 1.4 Comparación de dispositivos finales.

Nodo / Especificación	Dingtek DF703	NETOP Waste Bin Sensor
Dimensiones	Altura: 115mm	90mm
	Largo: 115mm	55mm
	Profundidad: 40mm	No especificado
Peso	150g	No especificado
Material de la carcasa	ABS (Acrilonitrilo Butadieno Estireno)	ABS (Acrilonitrilo Butadieno Estireno)
Tipo de sensor	Ultrasónico a 40kHz	Ultrasónico
Rango de detección	4 metros	Sobre los 2 metros
Precisión de medición	+/- 3cm	+/- 10mm

Precisión de temperatura	+/- 2ºC	No aplica
Chipset LoRa	Semtech 1276	No especifica
Frecuencias	US: 902-928 MHz	902-928 MHz
	EU: 863-870 MHz	863-870 MHz
Modo de activación	Clase A (OTTA por defecto)	No especifica
Rango de cobertura	3km en línea de vista	15km
Batería	De litio a 3.6V 8500mAh no recargable	Pilas AA 3.6V x3 o batería recargable de litio 3.6V
Tiempo útil de la batería	Mas de 3 años en intervalos de medición de 4 horas	Mas de 2 años en intervalos de medición de 4 horas
Temperatura de operación	-20º a 70º C	-40º a 85ºC
Certificación IP	IP68	No especifica
Precio	$80.00	$123.42 - $219.91

El nodo de NETOP posee las características necesarias para encajar en el presente proyecto, sin embargo, no tiene ningún adicional como si lo hace el de la empresa Dingtek, ya que el segundo tiene un detector de riesgo de incendio además de detección de ángulo de inclinación. Ciertas características relevantes no se especifican por parte de NETOP, como, por ejemplo, el modo de activación del dispositivo, microprocesador que ocupa y certificación IP (importante para proteger al dispositivo en la intemperie). Previamente se tomó contacto con los dos fabricantes para conocer más a profundidad acerca de los dispositivos finales, en asuntos como compatibilidad con el gateway, disponibilidad, tiempos de envío entre otras cosas, lastimosamente el soporte de parte NETOP no fue satisfactorio y no se pudo obtener más información de la que se podía encontrar en sus páginas web, por lo que se decidió optar por el dipositivo de Dingtek, cuya atención fue oportuna y puntual. El precio fue otra motivación para seleccionar el modelo DF703. El NETOP Waste Bin Sensor se muestra en la Figura 1.19.

Figura 1.19 NETOP Waste Bin Sensor [59].

1.4.5 HERRAMIENTAS DE DESARROLLO DE SOFTWARE

El presente proyecto no se centra únicamente en la implementación de la red LoRaWAN dentro del sistema, sino que también cuenta con una parte de software, este apartado, como se muestra en la Figura 1.1, se denomina entorno de gestión. En la actual sección se estudiará los lenguajes de programación envueltos en el desarrollo de la aplicación de escritorio y la base de datos, así como los entornos de desarrollo y APIs utilizadas.

1.4.5.1 Lenguaje C#

C# o C Sharp es un lenguaje de programación desarrollado por Microsoft utilizando recursos de otros lenguajes como C++, C y Java, incorporando distintas características propias. Es orientado a objetos y funciona para .NET Framework y .NET Core, con el tiempo aprovechó la facilidad que ofrecía Visual Basic, otro de sus lenguajes, para que la curva de aprendizaje sea sencilla sin sacrificar las mejores características de C. Al ser un lenguaje desarrollado para NET, cumple las mismas características: sencillo, moderno, seguro, compatible con otros lenguajes y eficiente[60].

1.4.5.2 Visual Studio 2019

Visual Studio es un IDE (Integrated Development Enviroment) desarrollado por Microsoft que permite crear aplicaciones de consola, aplicaciones web, aplicaciones para dispositivos móviles, aplicaciones en la nube, servicios web, entre otros. Integra herramientas externas propias de Microsoft como Silverlight y Windows API. Una de las mayores ventajas de este IDE es que facilita la utilización de distintos lenguajes como C#, Visual Basic, Phyton, JavaScript, Visual Basic y otros 31 diferentes[61]. Su primera versión fue lanzada en 1997, mientras que su versión más actual es la 2022, sin embargo, por familiaridad, este proyecto se desarrollará sobre la versión 2019.

Cuenta con distintas versiones, las cuales son descritas a continuación[61]:

- Community: es la única versión gratuita del IDE, sin embargo, es bastante similar a las versiones pagadas. Está orientada a desarrolladoras independientes y organizaciones no muy grandes.
- Professional: es una de las versiones comerciales del programa, una de las principales diferencias con la versión Community es que incorpora soporte para XML y XSLT, además de integrar herramientas de desarrollo directo con Microsoft SQL Server, Microsoft Azure y otras.
- Enterprise: esta versión ofrece un periodo de prueba de 90 días, y está elaborada para grandes equipos de desarrollo que requieren escalabilidad y calidad en sus programas.

Este proyecto ocupará la versión Community, ya que ofrece las herramientas suficientes para desarrollar la aplicación de escritorio.

1.4.5.3 SQL

SQL (Structured Query Language) es un lenguaje de programación orientado al desarrollo y administración de bases de datos relacionales, además de realizar operaciones algebraicas con sus datos. La sintaxis de este lenguaje se basa en sentencias para crear, borrar, actualizar tablas, sus filas y contenido, las más utilizadas suelen ser: select, add, insert, update, delete y create[62].

Este lenguaje se encuentra estandarizado desde 1986 a través de la ANSI (American National Standards Institute) y en 1987 por la ISO (International Organization for Standardization), los dos institutos desde la fecha han actualizado el estándar en 6 ocasiones, siendo la última en el 2011. A partir de estos estándares varias compañías y organizaciones desarrollan su propio entorno de gestión y motores de bases de datos, entre estas se puede encontrar: Microsoft SQL Server, Oracle Data Base, IBM DB2, SAP HANA, SAP Adaptive Server, MySQL y PostgreSQL, entre las más conocidas. A pesar de que las anteriormente mencionadas se basan en un mismo estándar, es probable que no sean totalmente compatibles entre sí, ya que han ido añadiendo extensiones para implementar características propias y funciones que las hacen únicas, como por ejemplo, Transact-SQL de Microsoft y PL/SQL de Oracle, sin embargo, su base lógica es muy similar [62], [63].

1.4.5.4 Microsoft SQL Server

Este trabajo de titulación utilizará Microsoft SQL Server como motor y sistema de gestión para el apartado de la base de datos de la aplicación de escritorio, por lo cual se realizará una breve revisión de este.

Este sistema es uno de los más utilizados en el mundo para gestionar bases de datos relacionales, utiliza Transact-SQL, que es una modificación de SQL para implementar extensiones propias de Microsoft. Provee de una interfaz gráfica, Microsoft SQL Server Management, para crear, borrar o administrar distintas bases de datos locales mediante sentencias en un script. Está compuesto por un motor relacional que se encarga de las operaciones, consultas, tablas, procesos, entre otras tareas relacionadas con la memoria y almacenamiento del computador o servidor donde se ejecute[64].

Algunas de las características principales de este sistema de gestión son: permite establecer clústeres de Big Data, funciona en distintos sistemas operativos como Windows o distribuciones Linux, optimiza los recursos de memoria, posee fuertes mecanismos de cifrado y encriptación, permite integración con sistemas BI y fácil escalabilidad con Azure. Al igual que con Microsoft Visual Studio 2019, SQL Server presenta distintas versiones de pago y gratuitas, para este proyecto se utilizará la llamada Developer Edition, que es de acceso libre e integra todas las funcionalidades de las versiones de pago, sin embargo, se considera como entorno de pruebas, más no de producción masiva [64].

1.4.5.5 APIs

Las APIs (Application Programming Interfaces) en C# se consideran como familias de métodos y protocolos que se integran en los proyectos de programación para implementar soluciones entre aplicaciones para cumplir determinadas tareas [65].

Este trabajo de titulación, en la parte de la aplicación de escritorio, utilizará dos APIs: la primera servirá para implementar el cliente MQTT en el código del programa, así se podrá recuperar los datos que generan los nodos desde el bróker MQTT en la plataforma de TTN y la segunda será usada para graficar la ubicación de los contenedores en un mapa.

2. METODOLOGÍA

Este capítulo discutirá la fase de diseño e implementación de este proyecto de titulación. Para esto, de manera rápida se revisará teóricamente el modelo en cascada y el paradigma de la programación orientada a objetos, que se utilizarán en la fase de diseño.

2.1 FASE DE DISEÑO

2.1.1 PARADIGMA DE LA PROGRAMACIÓN ORIENTADA A OBJETOS

C# es un lenguaje de programación que se basa en la Programación Orientada a Objetos (POO), la cual describe 4 características principales [66]:

- Abstracción: se debe modelar el sistema mediante clases utilizando los atributos más relevantes e interacciones entre entidades.
- Encapsulación: el acceso a objetos debe ser únicamente a través de métodos, sus atributos deben estar protegidos de otras clases del mismo sistema.
- Herencia: propiedad que permite crear clases a través de otras abstractas.
- Polimorfismo: propiedad que permite implementar métodos desde clases abstractas de diferentes formas.

2.1.2 MODELO EN CASCADA

Es un modelo de desarrollo de software en el cual sus fases se dan de forma progresiva, es decir, para avanzar al siguiente paso se debe culminar el anterior, es una de las metodologías más antiguas de desarrollo de software [67]; sin embargo, se acopla bien al presente proyecto ya que la intervención del usuario en el programa es mínima y se orienta únicamente a mostrar los datos que generan los dispositivos finales.

A continuación se describen brevemente las fases de este modelo [67], recalcando que no todas las fases se ejecutan en este capítulo, sino también en el tercer capítulo correspondiente a la Fase de Pruebas de Funcionamiento.

- Análisis de requisitos: en esta fase se define las tareas que realizará el software, es decir, lo que debería ofrecer al usuario al finalizar el desarrollo de este. No se enfoca en un análisis técnico sino más bien de forma y de necesidades.
- Diseño del sistema: en esta fase se define como se estructurará el sistema en cada una de sus partes y se especificarán sus tareas.
- Diseño del programa: en esta fase se elaborará los algoritmos que se utilizará en la parte de codificación.
- Codificación: en esta fase se escribe o codifica el programa en base a los algoritmos realizados en la etapa anterior.

- Ejecución de pruebas: en esta fase el desarrollador pondrá en prueba el programa codificado con la intención de buscar errores que pudiesen afectar el rendimiento de este. En caso de encontrar alguna falla será corregida de inmediato.
- Verificación: en esta fase el programa será puesto a prueba por el usuario, con el fin de encontrar nuevos posibles errores no detectados en la fase anterior. Al igual que antes, si se detecta un problema este será corregido.
- Mantenimiento: en esta fase se realizan actualizaciones al programa o corrección de nuevos errores no detectados en fases anteriores. Al ser este proyecto un prototipo no se tomará en cuenta este nivel.

El presente trabajo de titulación se basa en el Modelo en Casada para desarrollar el sistema, sin embargo, se modificaron la segunda y tercera fase para que se acople a las necesidades de la aplicación. Es importante recalcar que este modelo únicamente se aplicará en la parte de software, ya que para la sección de hardware se seguirá los protocolos LoRa, regulaciones emitidas por los entes estatales y las recomendaciones emitidas por los fabricantes.

2.1.3 ANÁLISIS DE REQUISITOS

Esta sección del capítulo 2 pondrá en ejecución la primera fase del modelo en cascada. Para definir las características y tareas que debía cumplir el programa se utilizaron encuestas aplicadas a profesionales y estudiantes que se relacionen con la tecnología LoRaWAN o con el desarrollo de aplicaciones en cualquier lenguaje. Otra alternativa que se puso a consideración fue la utilización de historias de usuario; sin embargo, al no contar con un usuario o usuarios específicos a quienes va dirigido el programa se descartó esta idea. Para llevar esto a cabo se utilizó formularios online, los cuales se difundieron mediante correo electrónico, debido a las restricciones al margen de la pandemia por el Covid-19 que impedían que se realicen de manera presencial. Se puso como meta obtener al menos 30 encuestas completas con el fin de diversificar la cantidad de opiniones y tener un mejor acercamiento a un producto final óptimo, las preguntas y resultados se pueden encontrar en la sección de Anexos como Anexo A.

La tabla 2.1 muestra los requisitos obtenidos para el programa con base en los resultados de las encuestas.

Tabla 2.1 Análisis de requisitos de la aplicación utilizando los resultados de las encuestas.

Requisitos de la aplicación	
N.º	**Información**
1	La aplicación a través de la interfaz gráfica mostrará únicamente los datos de interés respecto al estado de los contenedores.
2	De todos los datos generados por la red LoRaWAN, la aplicación a través de la interfaz gráfica mostrará sólo los siguientes: ID del dispositivo, medición de altura (ocupación), fecha y hora de la medición, otros datos medidos (batería, riesgo de incendio, contenedor caído).
3	La aplicación a través de la interfaz gráfica mostrará el nivel de ocupación de los contenedores de basura en porcentaje.
4	La aplicación a través de la interfaz gráfica mostrará la ubicación de los contenedores en un mapa al cual se accederá mediante la ventana principal.
5	La aplicación permitirá diferenciar los datos entre cada sensor, es decir, permitirá mostrar los datos de cada nodo por separado o en conjunto.
6	La interfaz gráfica y el cliente MQTT funcionarán de manera independiente.
7	La base de datos del programa almacenará todos los datos generados por los nodos de la red LoRaWAN.
8	La interfaz gráfica de la aplicación mostrará el nivel de ocupación utilizando colores para los casilleros con dicha información.

2.1.4 DISEÑO DEL SISTEMA

Esta fase es una de las alteradas con respecto al Modelo en Cascada original, y se utilizará para establecer la estructura del programa, así como los esquemas en los cuales se basa su desarrollo. El resultado de esta fase permitirá obtener el o los diagramas de clases, diagramas de casos de uso de todo el sistema y diagrama relacional para la base de datos. Además, se definirá las APIs C# que se utilizará para implementar el suscriptor MQTT.

2.1.4.1 Diagramas de clases y estructura del programa

Tal como se obtuvo en los Requisitos de la aplicación de la tabla 2.1, el programa de escritorio estará dividido en una aplicación de consola que monitoreará los mensajes publicados por el bróker MQTT de TTN, para procesarlos, mostrar lo más relevante y almacenar los datos en la base de datos, y la segunda parte que contará con la interfaz gráfica para recuperar la información y mostrarla. En esta sección se hablará sobre los diagramas de clases obtenidos para la primera parte del programa.

El bróker MQTT publica los datos en dos cadenas de formato JSON serializado, por lo que es necesario adaptarlos en clases y utilizar una API que permita deserializar esta información para convertirla en objetos C#. Las clases utilizadas para la aplicación de consola se describen a continuación:

- Nombre: ConexionMQTT_TTN.cs. Funcionalidad: esta clase contiene información correspondiente al cliente MQTT, tal como usuario, ID de los dispositivos, dirección web del bróker y clave de acceso. También contiene el método y los delegados para escuchar constantemente al publicador.
- Nombre: DeserializarConDatos.cs. Funcionalidad: esta clase representa la cadena JSON relevante para el proyecto, contiene los atributos necesarios para deserializar la información, procesarla y mostrarla en la consola.
- Nombre: DeserializarSinDatos.cs. Funcionalidad: esta clase representa la cadena JSON menos relevante para este proyecto, ya que no contiene información de interés para el usuario, de todas maneras, se la implementó para procesar ciertos datos y mostrarlos en la consola.
- Nombre: ClasesDB.cs. Funcionalidad: esta clase es similar a DeserializarConDatos.cs, pero contiene alteraciones como un ID único para cada clase y ciertos atributos fueron alterados en su tipo de dato, con el fin de coincidir con el diagrama relacional de la base de datos y que esta sea consistente.

La Figura 2.1 muestra uno de los diagramas de clase utilizados para tomar los datos JSON, transformarlos cada vez que un mensaje llegue y mostrarlos, mientras que la Figura 2.2 muestra el diagrama de clases utilizado para escribir la información obtenida desde el bróker MQTT en la base de datos. El resto de los diagramas de clase se pueden encontrar en la sección de anexos.

Como se puede observar, la diferencia entre estos diagramas reside en añadir un identificador único para cada clase de la Figura 2.2, de esta forma se pudo establecer relaciones entre entidades de la base de datos, ya que por defecto los objetos JSON generados por el bróker no incorporan un ID propio por clase que se pudiera utilizar como clave primaria o secundaria.

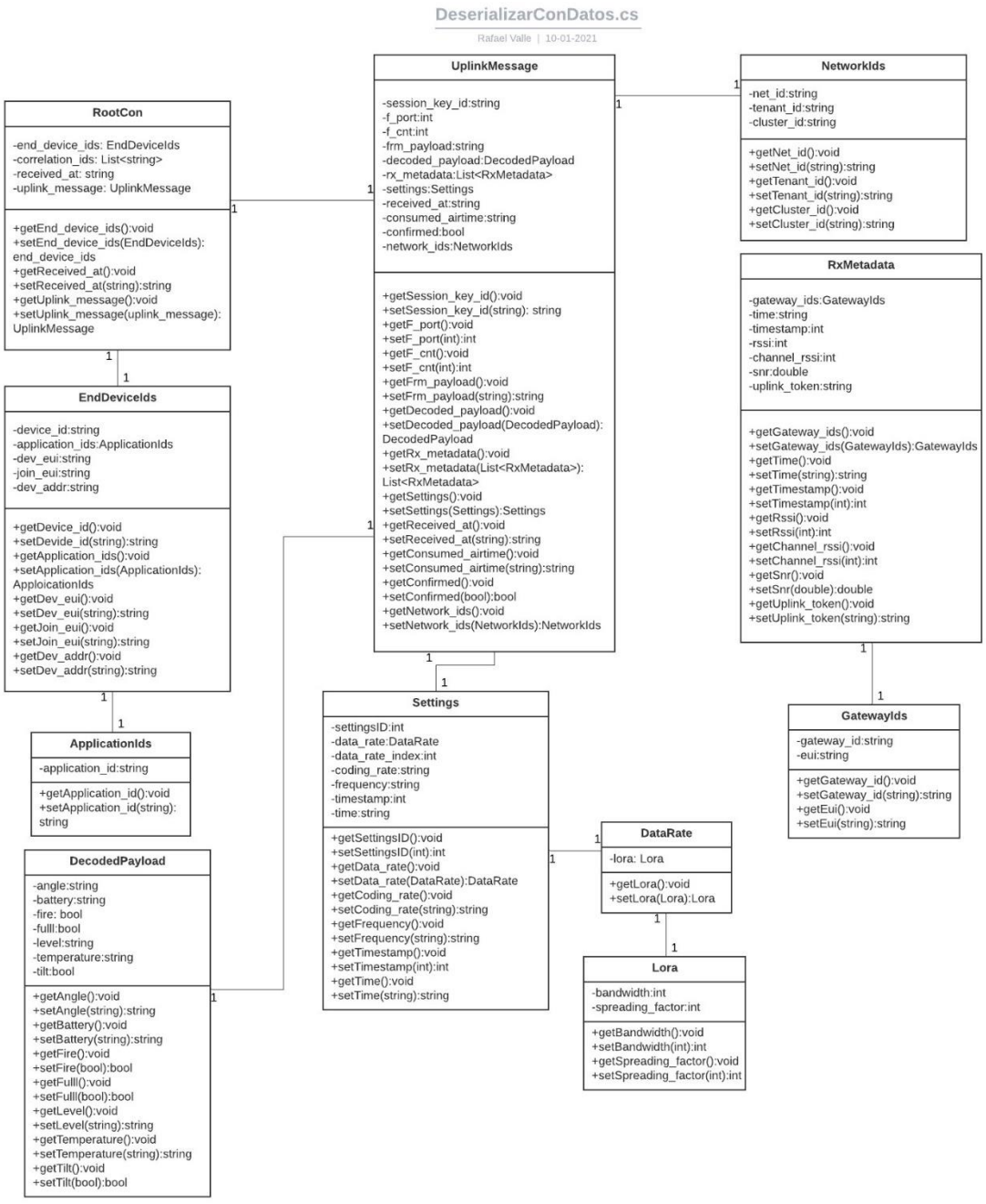


Figura 2.1 Diagramas de clases para convertir datos JSON en objetos C#.

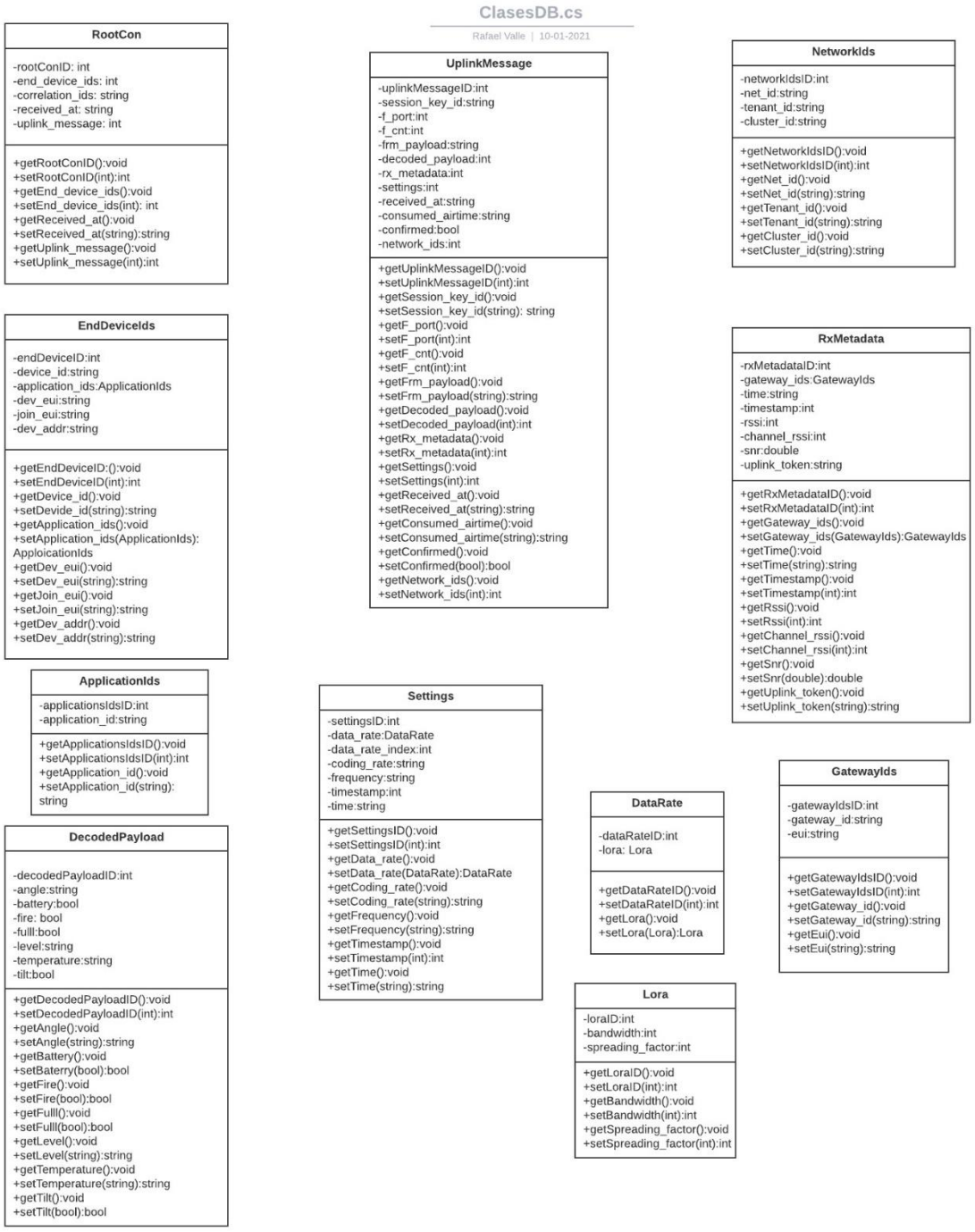


Figura 2.2 Diagrama de clases para conectar Visual Studio 2019 a Microsoft SLQ mediante LINQ.

2.1.4.2 Diagrama de casos de uso

El diagrama de casos de uso de la Figura 2.3 muestra como el desarrollador/administrador de red y usuarios interactúan con el sistema. La función principal de la aplicación es mostrar al cliente el estado de los contenedores mediante las mediciones que realicen los nodos cada cierto tiempo, adicional podrá acceder a la ubicación de estos mediante un mapa y al registro histórico de determinado contenedor

seleccionando un intervalo de fechas. Por otra parte, el desarrollador/administrador de red podrá ejecutar varias tareas relacionadas con la aplicación, base de datos y código del programa, así como gestión de los dispositivos y conexión con aplicaciones externas.

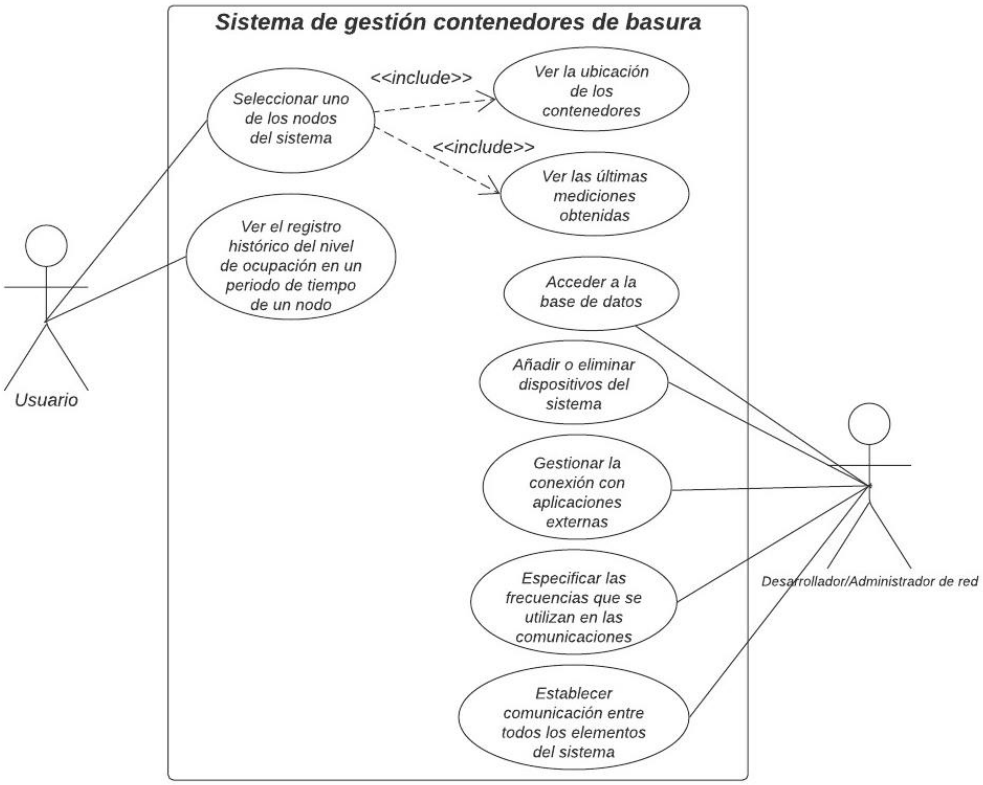


Figura 2.3 Diagrama de casos de uso del sistema.

2.1.4.3 Diagrama relacional

El diagrama relacional o modelo entidad relación en el campo del software se utiliza para diseñar bases de datos a través de figuras geométricas y símbolos que representan como las entidades se comunican entre sí, así como atributos, claves primarias y claves secundarias [68]. La Figura 2.2 muestra el diagrama relacional para la base de datos aplicados en este prototipo de aplicación, el cual guarda estrecha relación con el diagrama de clases de la Figura 2.3.

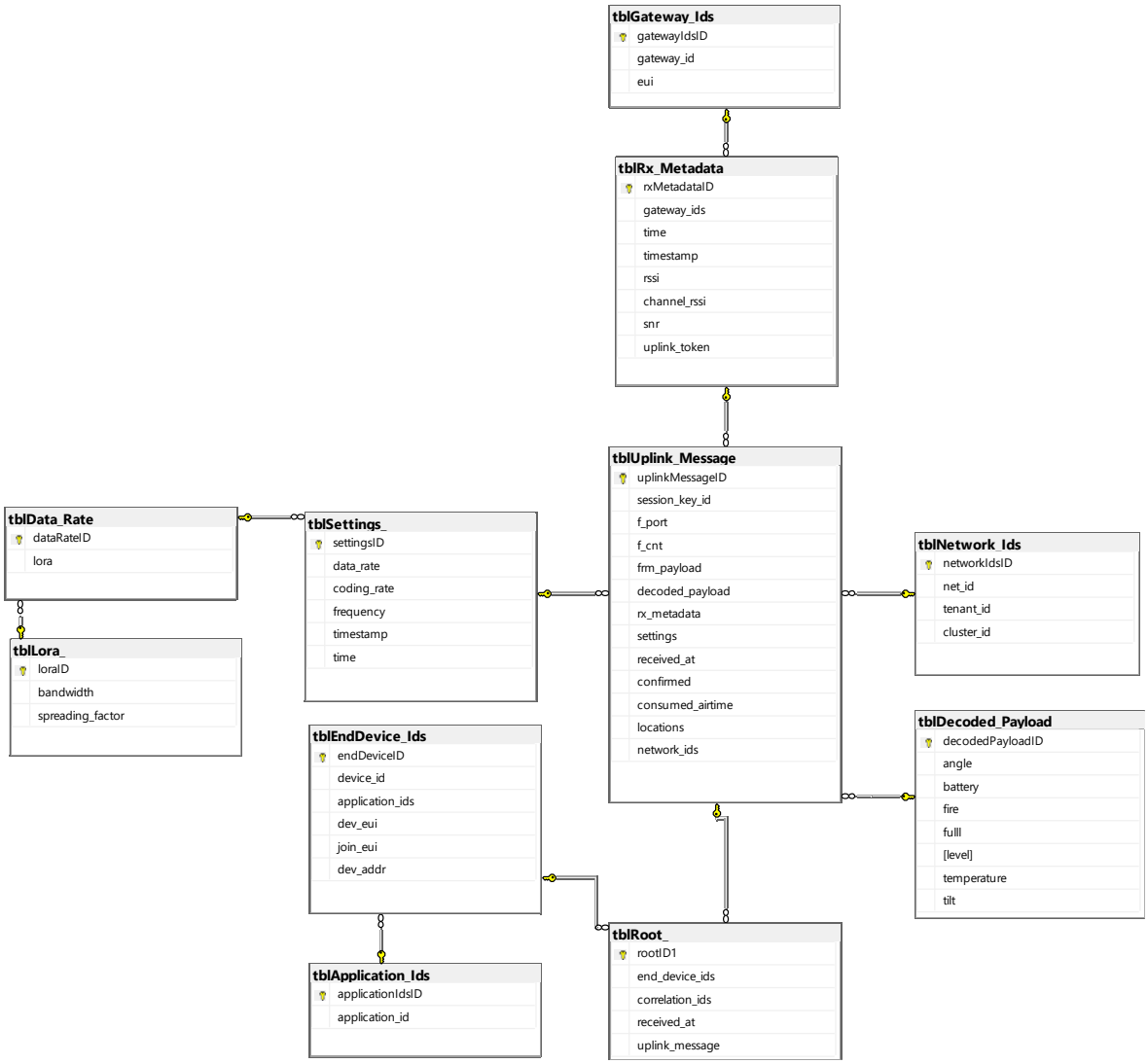


Figura 2.4 Diagrama relacional de la base de datos.

2.1.4.4 APIs para el desarrollo de la aplicación

API para el cliente MQTT:

- M2Mqtt permite implementar clientes MQTT en aplicaciones C# para plataformas .NET y WinRT, se enfoca especialmente en comunicaciones M2M para IoT. Es una API muy versátil, ya que se puede programar clientes para brókers MQTT locales o en el Internet. Además de esto provee de métodos para controlar excepciones, sesiones entre otras utilidades [69].

API para la implementación de mapas:

- GMap.Net es una API que contiene librerías para implementar mapas basados en Google Maps, Bing Maps, Yahoo Maps entre otros a una aplicación de Windows Forms en Visual Studio sobre el lenguaje C#. Cuenta con diferentes herramientas

gráficas que permiten personalizar la forma en la que los mapas son mostrados en los formularios, además de un marcador de rutas. Una de sus limitaciones es que se guía por latitud y longitud, no por direcciones exactas, ya que dependiendo de la plataforma en la cual se basen los mapas, algunas diferencias estarán restringidas. Por ejemplo, para Google Maps, las consultas a través de coordenadas son ilimitadas, sin embargo, si se quiere hacer lo mismo con direcciones exactas la plataforma cobrará una cantidad después de cierta cantidad de búsquedas. Este proyecto utilizará esta API y Google Maps como fuente de los mapas, ya que no requiere una elevada cantidad de consultas [70].

2.1.5 DISEÑO DE INTERFAZ GRÁFICA

Esta fase también fue cambiada con respecto a la original del Modelo en Cascada, ya que es importante que la aplicación implemente una interfaz gráfica agradable y simple de manejar, pero que se acople a las necesidades del usuario. Con este fin se utilizará el resultado de las encuestas aplicadas, cuyas conclusiones se puede observar en la tabla 2.1, además de bosquejos previamente realizados en la planificación de este proyecto, estos se muestran en las figuras 2.5 y 2.6.

Figura 2.5 Bosquejo de interfaz gráfica correspondiente a la representación de datos obtenidos desde los nodos.

Figura 2.6 Bosquejo de interfaz gráfica correspondiente a la representación de la ubicación de los contenedores a través de mapas.

El diseño final de la interfaz gráfica se realizará con base en los bosquejos de las dos figuras anteriores, pero tomando en cuenta los resultados de las encuestas realizadas.

2.1.6 DISEÑO DE LA RED (HARDWARE)

Por la pandemia provocada a raíz del Covid-19 se optó por instalar la red de comunicaciones en un ambiente controlado en lugar de en uno real. Como se explicó en el Alcance de este documento, se utilizarán contenedores tipo de madera o cartón con capacidad similar a los que EMASEO EP ha colocado a lo largo de la ciudad de Quito. Los nodos serán colocados en la parte superior de estos depósitos, al interior de la estructura. Este escenario evita que se deba realizar un estudio de campo previo a la implementación de los equipos de red, ya que se encontrarán en un ambiente sin perturbaciones que afecten a la señal o factores climáticos y ambientales. Eventualmente los contenedores se irán llenando con el fin de obtener diferentes mediciones y comprobar que en efecto los sensores funcionan correctamente, esta parte se revisa a profundidad más adelante en el segmento de Discusión de Resultados.

Los dispositivos que intervendrán en la parte de red fueron revisados y comparados en la sección de Marco Teórico, subíndice 1.4.4.

Una de las ventajas de LoRa es que opera dentro de las bandas ISM para sus comunicaciones inalámbricas LoRaWAN; sin embargo, estas varían según la región o la legislación de cada país, por lo tanto, este proyecto se regirá a la descrita por los entes correspondientes en el Ecuador.

El Arcotel en el 2020 emitió una resolución respecto a las UDBL (Uso Determinado en Bandas Libres), el cual describe un espectro de frecuencias que pueden ser utilizadas sin

la necesidad de adquirir licencia alguna en el rango de los 902MHz hasta los 928MHz con una frecuencia central de 915MHz. De todas maneras, no hace referencia a tecnologías en específico como LoRa, Zigbee y otras que utilizan bandas ISM. Es importante aclarar que para la prestación de servicios de telecomunicaciones o redes privadas que funcionen dentro de este rango de frecuencias sí se requiere la habilitación por parte del organismo estatal [71]. Ya que este proyecto únicamente implementará un prototipo no es necesario adquirir documento habilitante alguno, y utilizará uno de los planes de frecuencias que son oficialmente soportados por la plataforma de TTN.

Otro de los aspectos de red a considerar para el diseño del apartado de hardware es el modo de activación que se utilizará en los dispositivos finales. Como ya se estudió previamente el más seguro es el modo OTAA, por lo cual este será configurado en los nodos.

Una de las configuraciones que se debe realizar en el gateway Sentrius Laird RG191 tiene que ver con el reenviador de paquetes, a pesar de que este modelo cuenta con un modo preestablecido para conectarse con TTN se optará por utilizar el denominado Semtech Basic Fordwarder, ya que implementa certificados LNS (LoRaWAN Netowrk Server) y CUPS (Configuration and Update Server), el primero se utiliza para establecer una comunicación entre gateway y servidor mientras que el segundo permite que el servidor de red configure remotamente al concentrador de dispositivos [72], el segundo tipo de certificado no será utilizado en este proyecto ya que no se contempla una opción de configuración remota. Finalmente, la configuración de los dispositivos finales irá de la mano con las opciones que The Things Network brinda a través de su plataforma, apegándose estrictamente a los parámetros que especifica OTAA, además de las herramientas que brinda el fabricante de los nodos para el establecimiento de parámetros.

2.1.7 TOPOLOGÍA DE RED

Como ya se señaló en este documento las redes LoRaWAN pueden adoptar una topología tipo estrella de estrellas, sin embargo, por la limitada cantidad de dispositivos utilizados en este proyecto simplemente se considerará como una estrella. Los nodos se conectarán únicamente a un gateway, y a la vez este se conectará a un servidor de red. Esta topología se muestra en la Figura 2.7.

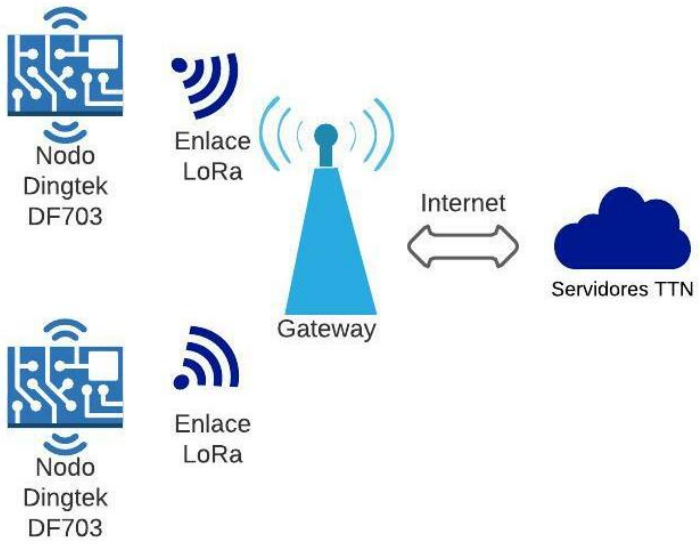


Figura 2.7 Topología de red a implementar.

2.2 FASE DE IMPLEMENTACIÓN

Esta sección del documento explicará cómo se configuró cada uno de los elementos de la red LoRaWAN para formar una red que se conecte a los servidores de TTN, además de la codificación de la aplicación de escritorio, incluyendo el cliente MQTT, base de datos e interfaz gráfica, utilizando los diagramas obtenidos en la Fase de Diseño. Para el entorno de gestión de la Figura 1.1 se utilizó un computador de escritorio con las siguientes características (de las más importantes): procesador Intel i5 4440 a 3.1GHz, 8GB de RAM DDR3, disco sólido 256GB, disco duro 1TB, Windows 10 versión 21H1. En el siguiente capítulo se analizará los recursos que requiere la aplicación para funcionar con fluidez y así establecer los requisitos mínimos de sistema.

2.2.1 CONFIGURACIÓN DEL SERVIDOR DE RED Y ADICIÓN DE GATEWAY A TTN

Para la parte de red LoRaWAN será necesario empezar por configurar el servidor de aplicación en la plataforma de The Things Network, ya que desde aquí se crearán ciertas claves que se usarán más adelante para validar certificados que se subirán en el gateway RG191. Fue necesario crear una cuenta en TTN, sin embargo, esto no será revisado en este documento. Se accedió a la consola de configuración de TTN mediante la dirección https://console.cloud.thethings.network/, aquí se desplegó una lista de los clústers con los que esta plataforma cuenta alrededor del mundo, se seleccionó el denominado nam1, que se encuentra en Norte América.

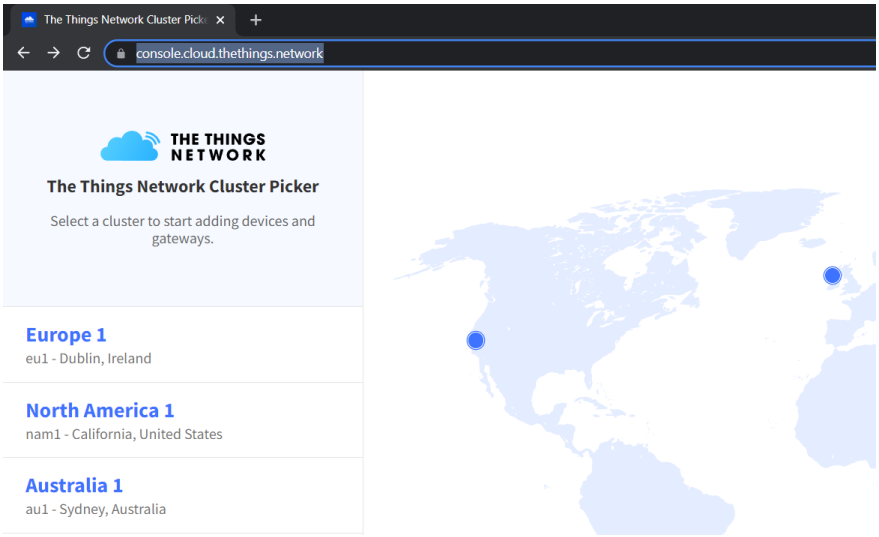


Figura 2.8 Selección de clúster TTN.

A continuación, tras ingresar con las credenciales correspondientes, en la parte superior de la ventana se mostró distintas opciones correspondientes a la consola de configuración. Dentro de estas se encuentran un resumen del perfil (Overview), Aplicaciones, Gateways y Organizaciones.

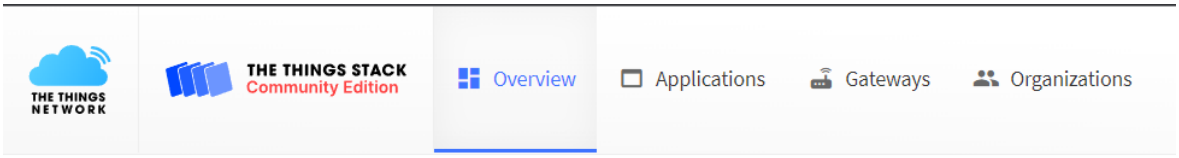


Figura 2.9 Opciones en la consola de configuración.

Para empezar se añadió el gateway, con este objetivo se dirigió hacia la opción Gateways, que se muestra en la Figura 2.11. Allí se encontraron opciones como buscar un concentrador de dispositivos por ID y reclamar o añadir un gateway, de las cuales se seleccionó la última.

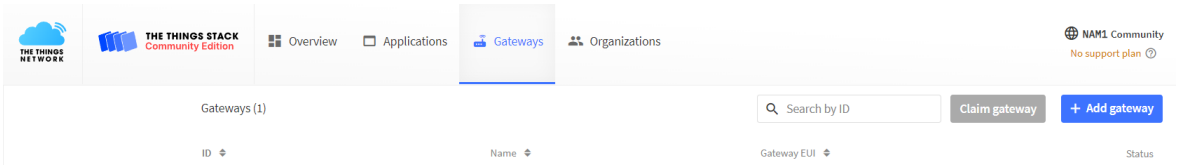


Figura 2.10 Pestaña de configuración Gateways.

Al hacer click sobre este botón se desplegaron distintos campos a completar con la opción correspondiente, algunos de estos parámetros ya se definieron en la Fase de Diseño, mientras que los que no fueron definidos se completaron según era conveniente. A continuación se describen estos campos y con qué dato o datos fueron completados.

- Owner: en esta opción se debe elegir el perfil de quien administrará el gateway, una misma cuenta puede tener distintos perfiles u organizaciones, por lo cual se debe elegir tan solo uno. En este caso se seleccionó el usuario brafavp creado al momento de registrarse en la plataforma TTN.
- Gateway ID: este campo debe ser único en comparación al del resto de gateways registrado bajo el correo electrónico de la cuenta, puede estar conformado por letras minúsculas, números y guiones (-). Para este proyecto se utilizó el ID rg1xx-rvp.
- Gateway EUI: es un identificador único del dispositivo que deseamos conectar, en el caso del Laird Sentrius RG191 se encuentra en una etiqueta colocada en la parte inferior del dispositivo junto a otras direcciones que se utilizarán más adelante. Por motivos de seguridad lo ideal es mantener esta información oculta a personal no autorizado.
- Gateway name: es un campo opcional, sin embargo es recomendable completarlo con un nombre que ayude a identificarlo del resto. Para este proyecto se utilizó el nombre Gateway Laird.
- Gateway description: es un campo opcional que sirve para describir el gateway y sus funciones, entre otras cosas. En este proyecto no se completó este campo.
- Gateway Server address: es la dirección del servidor de gateway en la plataforma TTN, dependerá de la ubicación de la red y se recomienda colocar el clúster que se encuentre más cerca a esta. Para este proyecto se utilizó la dirección nam1.cloud.thethings.network.
- Require authenticated connection: esta opción se deberá activar si se desea trabajar con el protocolo TLS, este proyecto no contempla la utilización de esto por lo cual permanecerá desactivada.
- Gateway status: si se activa esta opción el gateway será visible para otros usuarios en la red. Para este proyecto se activó esta opción ya que no causaría conflicto al tener únicamente un usuario.
- Gateway location: similar a la opción anterior, pero relativo a la ubicación del gateway. Esta opción también se activó para este proyecto.
- Frequency plan: en este menú desplegable se mostrarán todas las frecuencias admitidas por TTN para las distintas regiones del mundo. Tal como ya se definió en la Fase de Diseño, se utilizará el subgrupo de frecuencias FSB1, en la plataforma se encuentran con el nombre United States 902-298 MHz, FSB 1.
- Schedule downlink late: esta opción permite almacenar mensajes del enlace de bajada en los servidores de TTN, se recomienda activar para gateways que no

cuentan con un buffer para estos mensajes. Ya que este proyecto no requiere enviar mensajes en este enlace, la opción se desactivó.

- Enforce duty cycle: no es obligatorio activar esta opción, sin embargo, es recomendable ya que a través de esta TTN monitoreará que las regulaciones del espectro del plan de frecuencia seleccionado se cumplan. Para este proyecto la opción se mantendrá activa.
- Schedule any time delay: este campo se utiliza para dispositivos Clase C, y ya que este proyecto no utilizará ese tipo de nodos no adquiere relevancia.

Las opciones restantes referentes a actualizaciones automáticas fueron desactivadas.

Al completar los campos descritos anteriormente con la información detallada se hizo click sobre el botón Create gateway, al final de esta página, y así el dispositivo se añadió a la plataforma de TTN, aunque aún no se encuentran conectados.

Tras finalizar esta tarea la plataforma se redireccionó hacia la ventana de la Figura 2.11 y mostró el gateway creado.

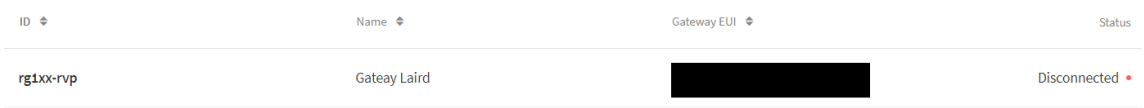


Figura 2.11 Resumen del gateway añadido en la plataforma TTN.

Esta sección explica únicamente una parte de la configuración del servidor de red, de todas maneras, más adelante durante la configuración de gateway y nodos, se volverá a utilizar esta consola con distintos propósitos que serán explicados en su momento.

2.2.2 CONFIGURACIÓN DE GATEWAY

La configuración y establecimiento de parámetros del Laird Sentrius RG191 se realizó a través del navegador Google Chrome en el entorno de gestión. Para acceder a la interfaz gráfica del concentrador de dispositivos es necesario que este se conecte al router de la red local donde se encuentra el computador, tanto PC como gateway se pueden conectar mediante enlace inalámbrico o guiado, el presente proyecto utilizó la segunda opción. Es importante recalcar que esta configuración se puede realizar en sistemas operativos Windows, Linux o Mac que cuenten con un navegador actualizado. El primer paso fue conectar el dispositivo a una fuente de energía alterna y posteriormente al enrutador de la red, como lo muestra la Figura 2.12.

Figura 2.12 Energización de gateway y conexión a la red local.

Existen dos formas de comprobar que la conexión se estableció correctamente y los dispositivos se encuentran en la misma red. La primera es enviar un mensaje de ping hacia el gateway mediante un terminal del sistema operativo y la segunda es acceder directamente a la interfaz gráfica de configuración mediante un navegador web. La Figura 2.13 muestra el primer método, mientras que el segundo se detalla en seguida.

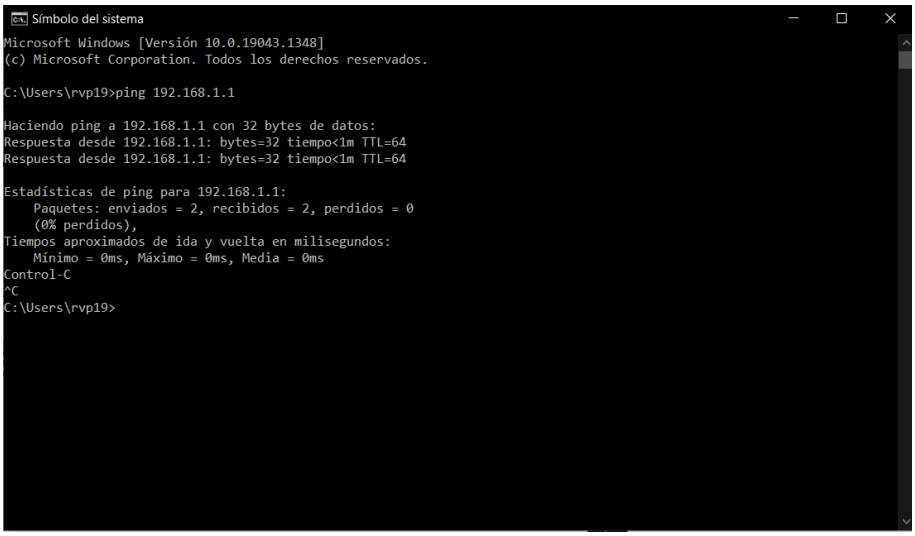


Figura 2.13 Ping hacia la IP del gateway.

Para llevar a cabo el segundo método: en una pestaña del navegador Google Chrome se introdujo la dirección https://rg1xx29836f.local, donde 29836f corresponde a los últimos 6 dígitos de la dirección MAC del Gateway, como se encuentra resaltado en la siguiente figura. Esta etiqueta informativa se ubica en la parte inferior del dispositivo.

Figura 2.14 Etiqueta incorporada en el Laird RG191.

Este trabajo de titulación utilizó los dos métodos para comprobar la conexión del gateway a la red local, el resultado del primero de estos se muestra en la Figura 2.13, mientras que para el segundo se obtuvo la respuesta de la Figura 2.15.

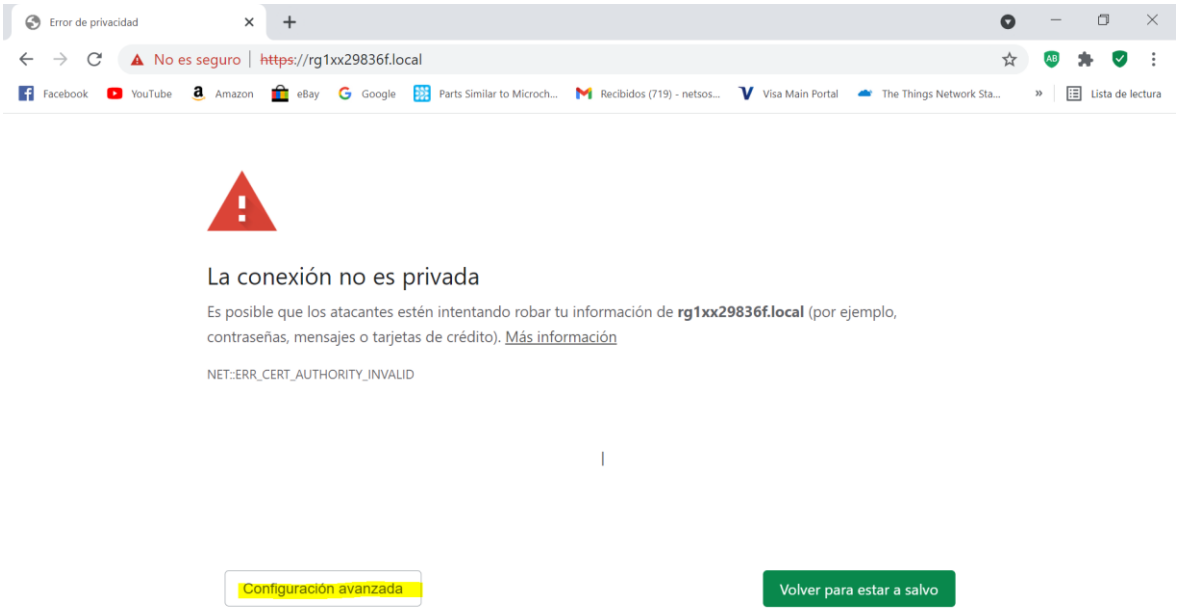


Figura 2.15 Respuesta desde el Laird RG191 mediante Google Chrome.

La primera vez que se ejecutó el segundo método se obtuvo un mensaje de conexión no privada, al ser un dispositivo de confianza se ignoró la advertencia y se hizo click sobre el botón de Configuración avanzada, resaltado en la figura anterior, lo cual derivó en la información mostrada en la Figura 2.18, donde se seleccionó el enlace de “Acceder a rg1xx29836f.local (sitio no seguro)”, que se encuentra remarcado en amarillo.

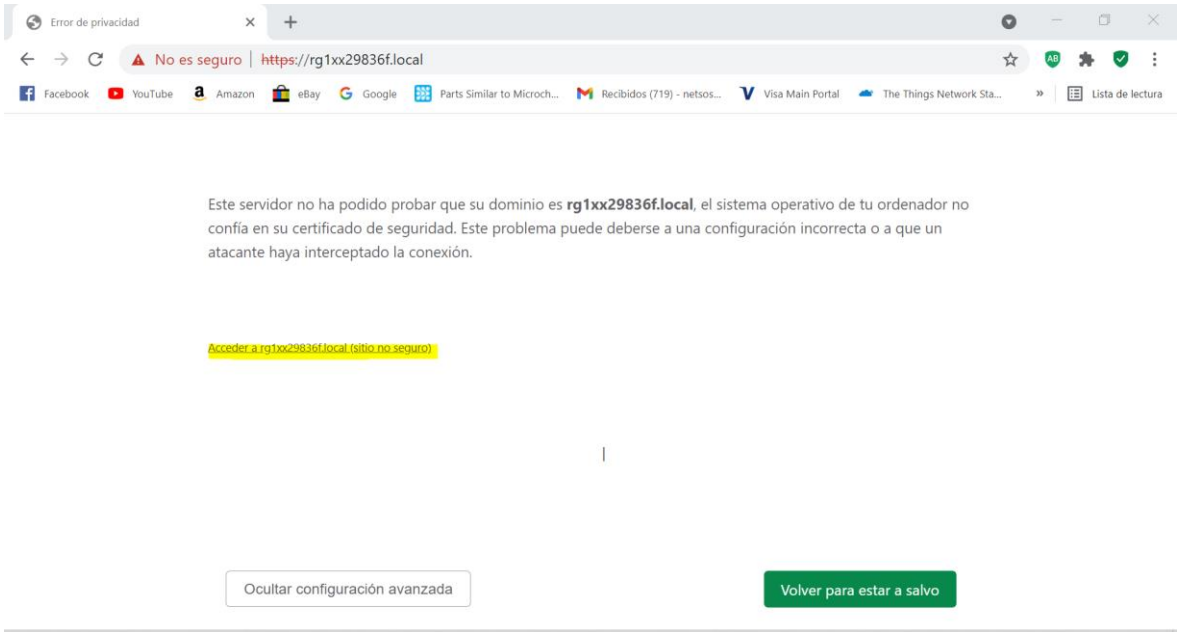


Figura 2.16 Acceso a la interfaz gráfica del gateway.

Esta advertencia se mostró únicamente la primera ocasión que se utilizó esta forma de acceso, las siguientes veces el navegador ingresó directamente a la UI.

A continuación, el equipo solicitó las credenciales de acceso, en su configuración por defecto el gateway Laird RG191 utiliza sentrius y RG1xx como nombre de usuario y contraseña respectivamente. Estos parámetros fueron cambiados más adelante por motivos de seguridad. Una vez adentro lo primero que se obtuvo fue un panel que mostraba un resumen del dispositivo respecto al sistema, red LoRa configurada, red Wi-Fi y red LAN, esto se puede observar en la siguiente figura.

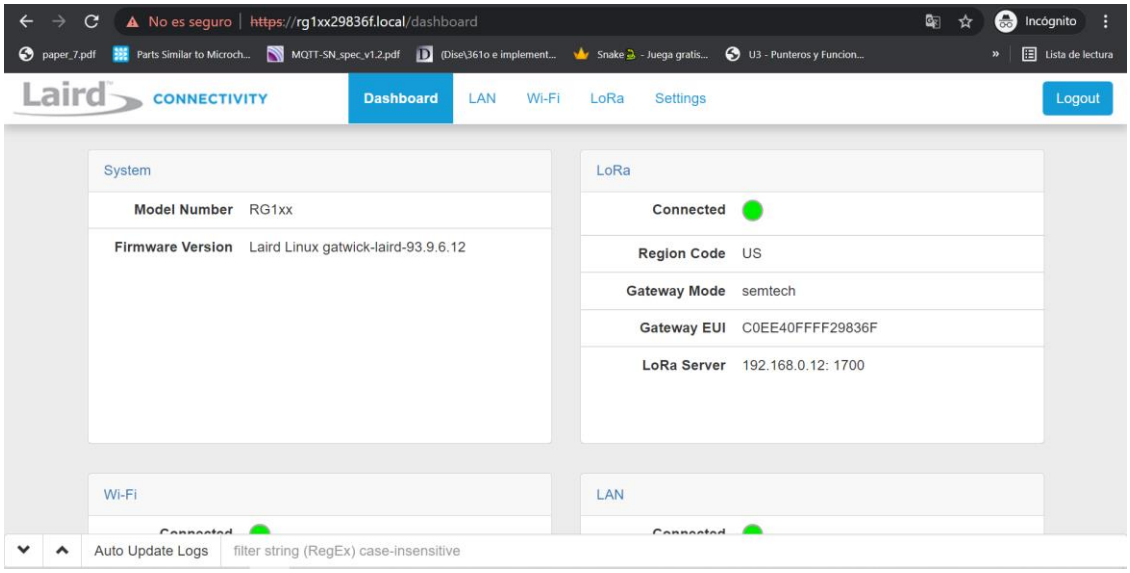


Figura 2.17 Consola principal del Laird RG191.

Es en este punto donde se retomó la configuración del servidor de red mediante la consola de TTN, ya que como se comentó en la Fase de Diseño es necesario crear una clave para el protocolo LNS.

La clave para LNS requirió un solo permiso individual, tal como se detalla en [73] y [74]:

- Enlazar en modo de gateway a servidor de gateway para intercambio tráfico, por ejemplo, enviar datos en el enlace de subida y leer los del enlace de bajada, en inglés: Link as Gateway to a Gateway Server for traffic exchange, i.e. write uplink and read downlink.

Con el fin de crear esta clave, en la consola de TTN se dirigió al panel de control de Gateways, y de la lista de dispositivos se escogió el creado antes con nombre rg1xx-rvp, y de inmediato se mostró un resumen de configuración relacionado con el dispositivo, además de un panel de control ubicado en la parte izquierda el cual incluye un segmento de datos en vivo, localización, colaboradores, la opción de interés para este paso llamada API Keys, entre otros.

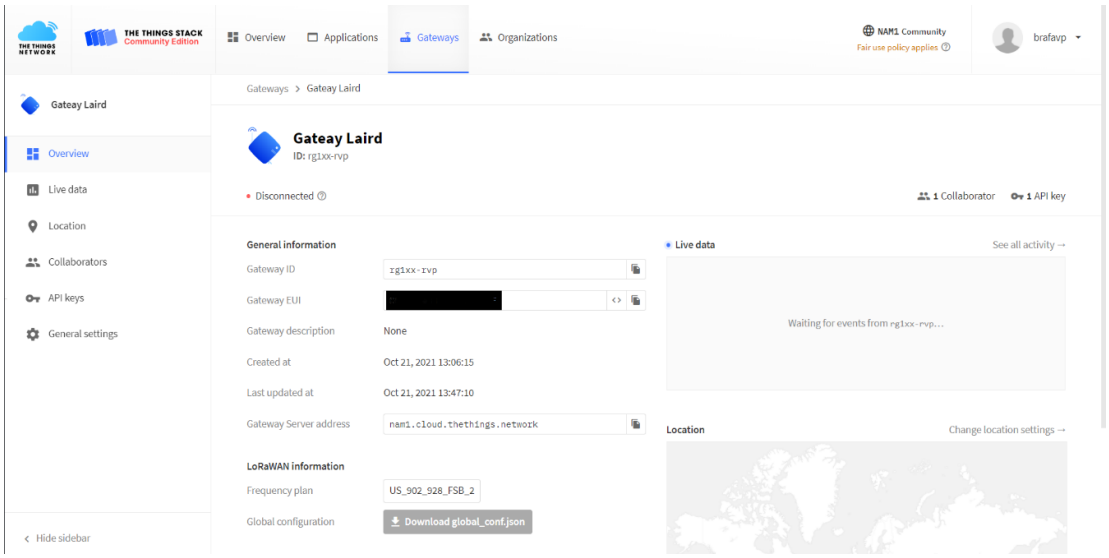


Figura 2.18 Panel de control gateway rg1xx-rvp.

De las opciones mencionadas anteriormente se hizo click sobre API keys, que despliega un menú de opciones similar al de la Figura 2.10 pero relativo a las claves para protocolos. Al hacer click sobre el botón “+ Add API key” se mostraron dos campos por completar, el primero es el nombre de la clave, que en este proyecto se denominó LNS key, mientras que la segunda opción corresponde al otorgamiento de permisos, el cual fue explicado anteriormente. La siguiente figura ilustra lo explicado en este párrafo.

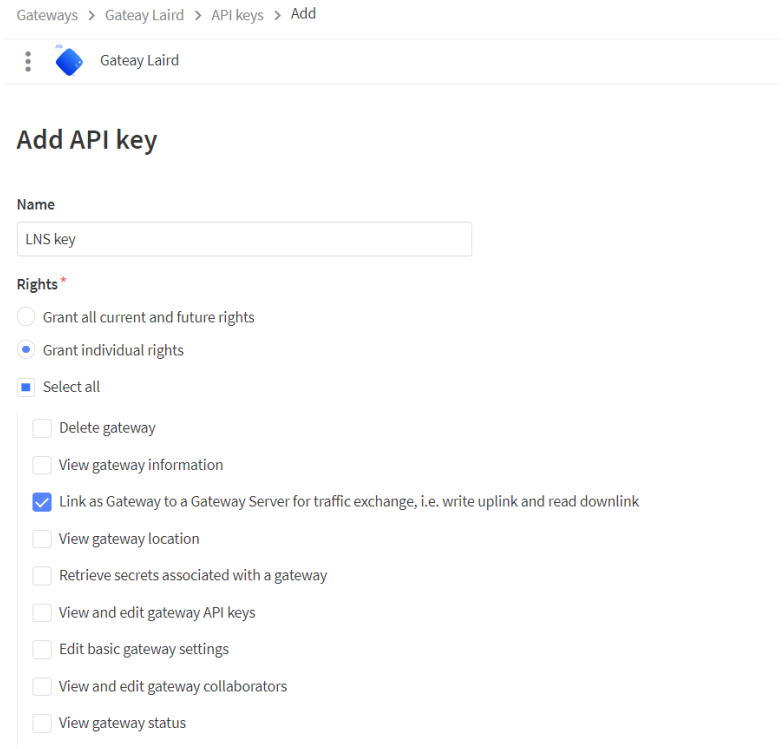


Figura 2.19 Creación de clave para LNS.

Para culminar este proceso se dirigió a la parte final de esta página y se hizo click sobre el botón “Create API key”.

Es importante aclarar que en este apartado el usuario únicamente da la directiva de crear la clave, sin embargo es el servidor de The Things Network quién se encarga de generarla, además del identificador único de esta.

La siguiente información que se mostrará es la clave generada, la cual debe ser copiada y almacenada en un lugar seguro ya que no se podrá volver a ver en cualquier momento en el futuro a través de la consola de TTN. La Figura 2.20 muestra lo explicado en este párrafo.

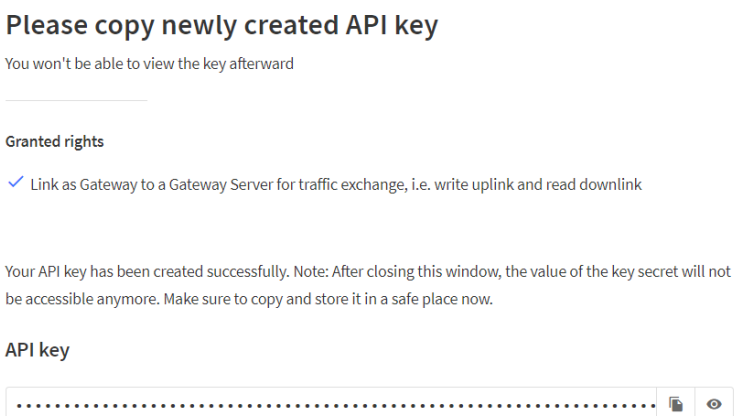


Figura 2.20 API key que podrá ser vista únicamente esa ocasión.

Tras completar estos pasos se dirigió a la opción de “General settings” mostrada en la Figura 2.18, y se incluyó la contraseña de la denominada LNS key en los campos de configuración del gateway dentro de la plataforma.

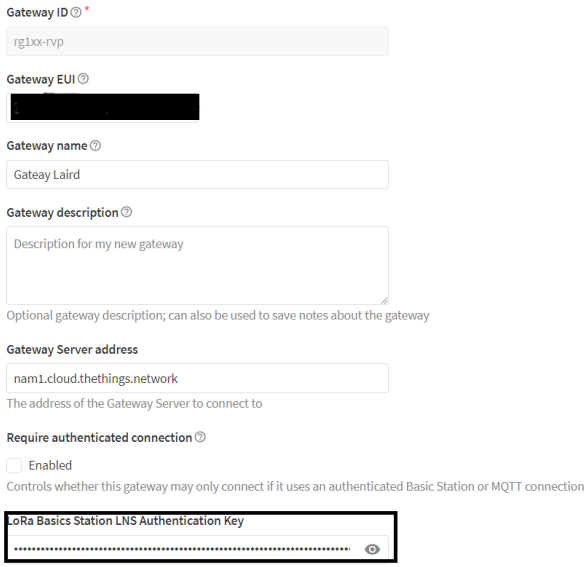


Figura 2.21 Inclusión de la LNS Authentication Key en la configuración del gateway.

Mediante la interfaz gráfica de configuración del gateway se cargó un certificado generado por una Autoridad de Certificación (CA) en el dispositivo, en este caso por la organización Let’s Encrypt, quién los provee de manera gratuita además de hacerlos abiertos al público en general [75]. TTN acepta distintos certificados para este apartado, de todas maneras se utilizó el denominado ISRG Root X1 elaborado por la organización mencionada, y se descargó directamente desde su página web. Los CAs son entidades que verifican la autenticidad de certificados digitales para tareas como autenticación y firmas electrónicas [76]. Bajo el contexto de este proyecto, este certificado se utilizó para verificar la autenticidad de los servidores de TTN cada vez que se conecten con el gateway RG191.

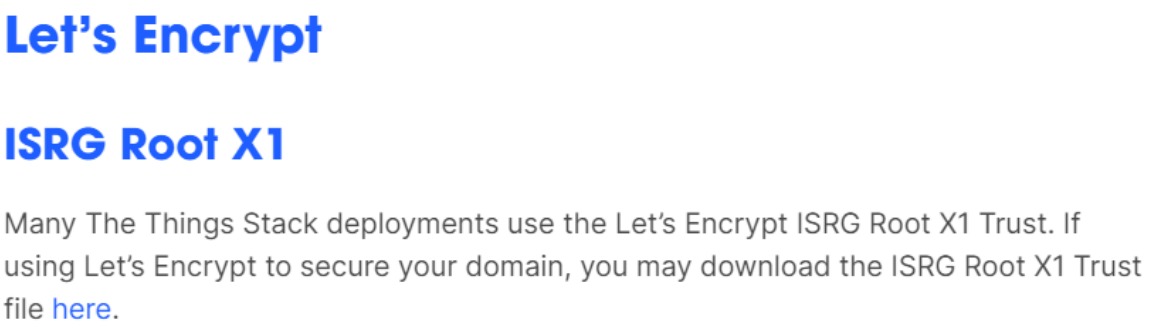


Figura 2.22 Enlace de descarga del certificado ISRG Root X1 desde TTN [75].

En la interfaz de configuración del gateway mostrada en la Figura 2.19 se seleccionó la pestaña LoRa, y en el menú ubicado en la parte izquierda se hizo click sobre la opción Forwarder. En las opciones desplegadas en la parte derecha se mostró un deslizable llamado Mode, donde se seleccionó la opción Semtech Basic Station. En la sección de Server Configuration únicamente se completó el campo LNS Server con la dirección wss://nam1.cloud.thethings.network:8887, correspondiente al servidor de gateway en TTN. La segunda sección se denomina LNS Certificates, y aquí se cargó el certificado ISRG Root X1 y la clave en archivo con extensión .key. El primero se descargó directamente desde The Thing Networks, como se explicó en la Figura anterior, mientras que el segundo se obtuvo desde un terminal del sistema operativo, utilizando la clave copiada desde la ventana de la Figura 2.20. La siguiente figura muestra el comando utilizado para este paso.

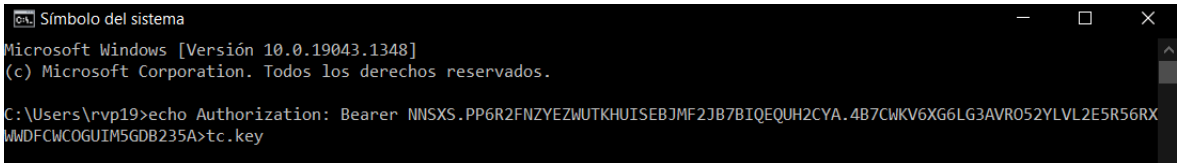


Figura 2.23 Comandos para obtener el archivo .key desde la LNS key.

El archivo en este caso se nombró tc.key se ubicó en el directorio sobre el cual se ejecutó esta línea de comando, que para el caso anterior fue C:\Users\rvp19.

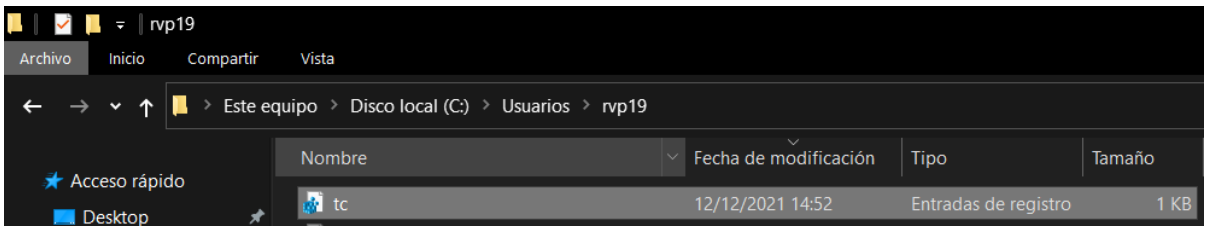


Figura 2.24. Archivo tc.key y directorio de ubicación.

En la parte de LNS Certificates se encontraron 3 botones blancos con la leyenda Choose File. De estos se utilizaron el primero y tercero, en el número 1 se subió el certificado ISRG Root X1, que lleva como nombre isrgrootx1.pem, y en el número 3 se subió el archivo tc.key de la Figura 2.24. Para finalizar este proceso se dio click sobre el botón azul con la leyenda Upload Certificates.

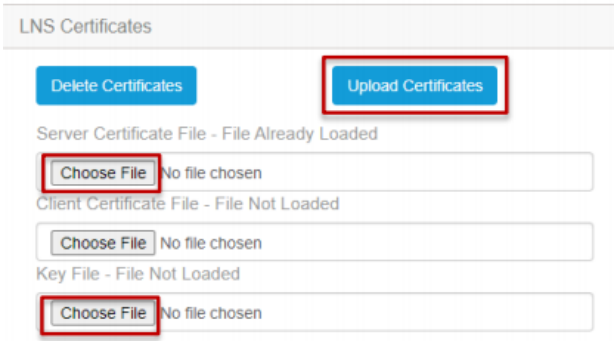


Figura 2.25 Espacio LNS Certificates [77].

A lo largo de este subíndice se utilizó las guías mostradas en [72], [73], [77]–[79].

Tras culminar todas las tareas anteriores se esperó durante unos segundos y finalmente el gateway estableció conexión con los servidores de TTN. Esto se verificó mediante la interfaz gráfica del Sentrius RG191 en Google Chrome, que mostró como estado Conectado, junto a un ícono circular de color verde.

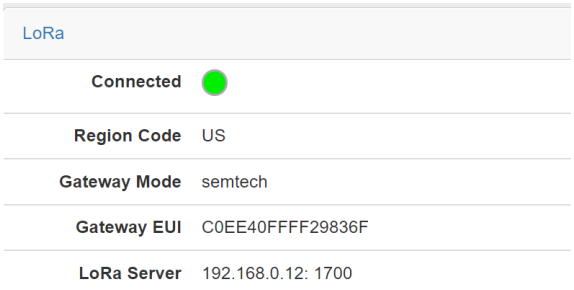


Figura 2.26 Estatus de conexión LoRa en el gateway.

2.2.3 CONFIGURACIÓN DE DISPOSITIVOS FINALES

Al igual que con el Sentrius RG191 la configuración de los dispositivos finales se realizó en paralelo con algunas implementaciones en la plataforma de TTN. Uno de los requisitos previos a la incorporación de los nodos a la red fue configurar un servidor de aplicación en la plataforma de TTN. Para esto, a través de sus opciones mostradas en la Figura 2.11, se seleccionó la denominada Applications. Esta desplegó la lista de servidores disponibles y un botón para agregar uno nuevo, donde se hizo click. La configuración de

este fue mucho más rápida que la del gateway en la sección anterior, los campos que se completaron fueron:

- Owner: en este menú desplegable se debe asignar el usuario de la cuenta al cual se desea que se vincule el servidor de aplicación. En este proyecto se utilizó el usuario brafavp.
- Application ID: este campo sirve para identificar al nuevo servidor, debe contener únicamente letras minúsculas, números y guiones (-). En este proyecto se utilizó el ID binsensors-appserver01 y binsensors-appserver02.
- Application name: es un campo opcional, y se completa según el usuario crea conveniente para identificar al nuevo servidor. En este proyecto se utilizó el nombre Servidor sensores basura.
- Description: es un campo opcional y sirve para describir al servidor. En este proyecto no se completó este campo.

Para finalizar la creación se hizo click sobre el botón Create application.

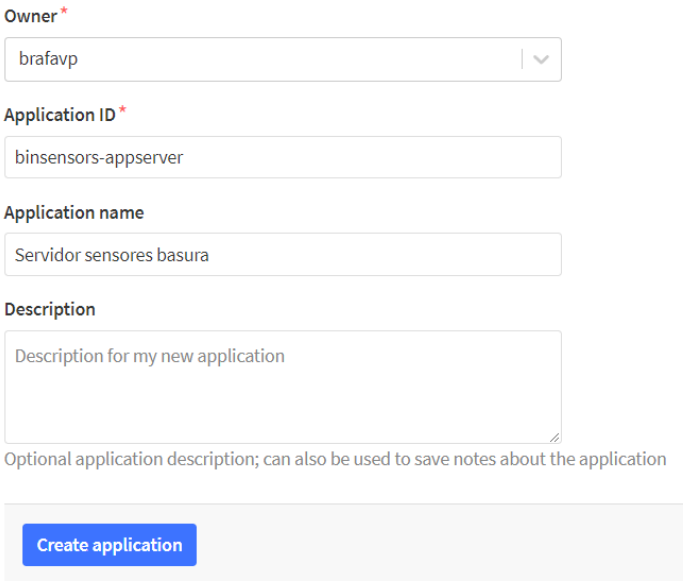


Figura 2.27 Creación del nuevo servidor de aplicación.

A continuación, la consola de TTN se redirigió al panel de control de este servidor. En la parte izquierda se encontró un menú de acceso a distintas configuraciones y opciones de visualización, estas se describen a continuación:

- Overview: muestra un resumen del servidor que incluye nombre, fecha de creación, última actualización y datos en vivo.
- End devices: desde aquí se agregan, eliminan, modifican y configuran los dispositivos finales, en esta sección esta será una de las opciones más utilizadas.

- Live data: permiten observar los datos que se generen por los nodos, es importante aclarar que la plataforma únicamente muestra la información a través de la caché del navegador, por lo que si se actualiza la página la mayoría de esta se perderá si no se almacena en una base de datos persistente.
- Payload formatters: este campo debe ser configurado de acuerdo con lo establecido por el fabricante de cada dispositivo, ya que permite decodificar los datos generados por los sensores y enviados a través del gateway. Para este proyecto se utilizó un código en Java Script provisto por Dingtek.
- Integrations: esta sección permite configurar la integración con protocolos y aplicaciones externas. Será utilizada más adelante para configurar el bróker MQTT.
- Collaborators: mediante esta opción se pueden añadir colaboradores externos que cuenten con una cuenta de TTN.
- API keys: aquí se podrán crear claves para la integración con protocolos o aplicaciones externas, dependiendo de la opción utilizada la plataforma las creará de manera automática o tendrán que añadirse de forma manual.
- General settings: esta sección permite modificar la configuración del servidor de aplicación en cuestión.

El siguiente paso correspondió a la integración de los dispositivos finales a la red mediante las utilidades del servidor de aplicación recién configurado. El fabricante de los sensores envió los nodos preconfigurados con las especificaciones definidas en la Fase de Diseño, de todas maneras, una vez que arribaron se revisó los parámetros de estos mediante un módulo serial TTL a USB, incluido en el paquete, el computador utilizado en el entorno de gestión y un programa provisto por el fabricante para estas tareas de configuración y comprobación de estado.

Figura 2.28 Sensor en el empaque y conversor TTL a USB.

Ya que los dispositivos están diseñados para colocarse a la intemperie no poseen ningún puerto de entrada o salida en su carcasa, por lo que fue necesario abrirlos para acceder a su puerto UART.

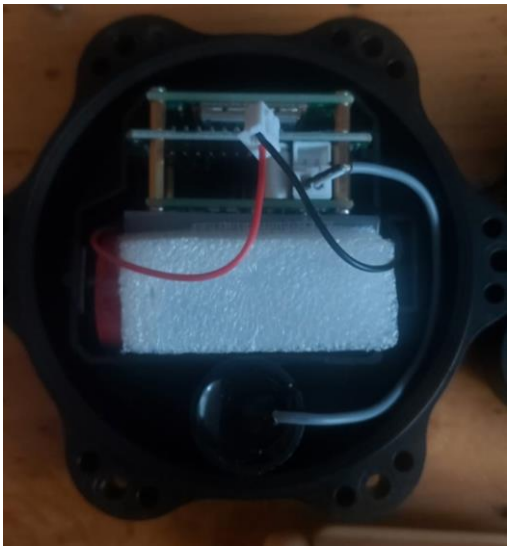


Figura 2.29 DF703 abierto.

Una vez abiertos se verificó que cuentan con una batería no recargable de litio la cual se encontraba desconectada, a diferencia del sensor ultrasónico que ya venía acoplado al circuito. Se procedió a alimentar los dispositivos con la batería y a conectarlo al computador mediante un puerto USB.

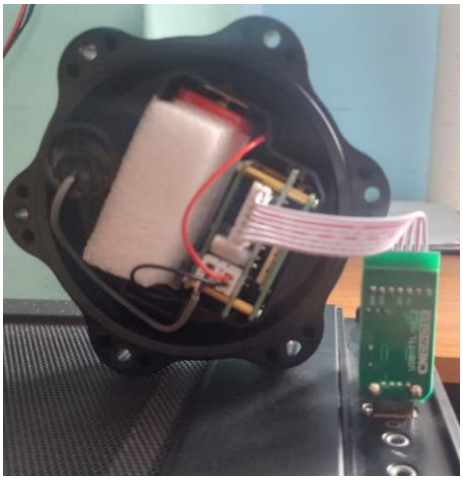


Figura 2.30 Conexión del dispositivo mediante un puerto USB.

Se utilizó el programa CommUart Assistant (V3.8), provisto por el fabricante, el cual permite establecer comunicación con dispositivos conectados a un computador mediante USB. En la parte izquierda superior de este se encontraron las configuraciones respecto al número de puerto, velocidad de transmisión, bits de paridad y bits de parada, se utilizó los parámetros recomendados por el fabricante. En la misma ubicación, pero al medio y la parte baja, se encontraron opciones respecto al formato de envío y recepción de mensajes, aquí únicamente se deseleccionó la opción enviar y recibir en hexadecimal, activada por defecto. La parte derecha muestra las respuestas que se recibe desde el dispositivo y bajo esta se encuentra el cuadro que se utilizó para enviar mensajes a los nodos.

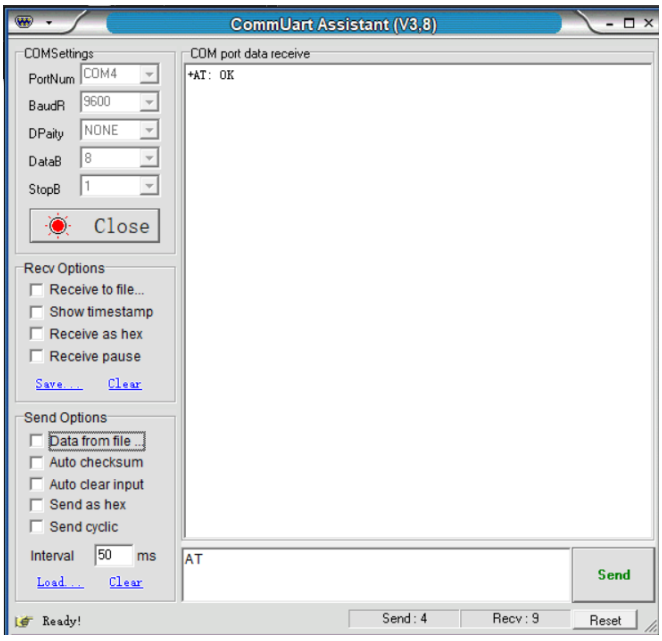


Figura 2.31 CommUart Assistant en ejecución.

La documentación del fabricante especifica que los nodos tienen 3 modos de funcionamiento, en el primero los dispositivos finales realizan mediciones en periodos de entre 1 minuto a 60 minutos, pero únicamente reportarán los datos a la red LoRaWAN en caso de que exista algún cambio respecto al estado anterior, el segundo modo elimina el ciclo de detección, por lo que los nodos reportarán los datos con su gateway en periodos de entre 1 hora a 24 horas, mientras que el último modo reportará los datos en el ciclo de detección, es decir, no comparará los datos actuales con los anteriores, sino que se comunicará con la red directamente. Al finalizar cualquiera de estos modos los sensores entran en modo de ultra ahorro de energía, para garantizar la larga duración de su batería. Para este proyecto se utilizó el último modo de funcionamiento.

Posteriormente se verificó el estado del dispositivo respecto a parámetros relacionados con la red LoRaWAN, además de campos relativos a altura y periodos de medición.

Los comandos AT se encuentran especificados por la LoRa Alliance, y se utilizan para manejar el módulo de este protocolo en los circuitos de los dispositivos finales [80], en todos los nodos estos mensajes se envían a través de un puerto UART. La cantidad de ATs que puede reconocer un dispositivo varía de modelo a modelo, en el caso del DF703 identifica una cantidad limitada de estos, y utiliza otro tipo de mensajes para modificar los parámetros de configuración de los nodos. Para que estos dispositivos pudiesen reconocer los comandos AT fue necesario programarlos en modo debug, para lo que se siguió la guía provista por el fabricante [81]. Una vez ahí se envió el comando AT, al que todo dispositivo debería responder OK, tal como se muestra en la Figura 2.24. Otros comandos ejecutados fueron AT+VER, AT+DR y AT+CLASS, que devuelven la versión del dispositivo, las tasas de transmisión configuradas y la clase (A, B o C), respectivamente. Sin embargo, como se explicó anteriormente los parámetros de la red LoRa no pudieron ser alterados mediante estos comandos, como suele ser normalmente. Para esto se utilizó otro de los manuales de usuario elaborados por el fabricante [82].

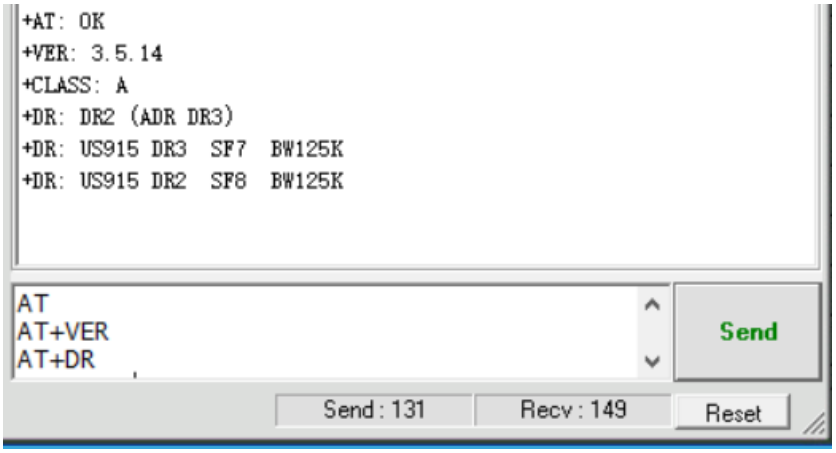


Figura 2.32 Respuesta a distintos comandos AT.

Para los comandos de configuración no fue necesario que los dispositivos se encuentren modo debug, así que para regresarlos a su modo normal fueron reiniciados, y mediante el mismo programa se los ejecutó. Una de las complicaciones encontradas en este paso fue que para que los comandos sean aceptados por el dispositivo este tenía encontrarse activo, por lo que se tuvo que esperar ciertos periodos de tiempo para ejecutarlos, por configuración de fábrica el dispositivo entra en modo de ahorro de batería (o Sleep) cada 10 minutos. La siguiente figura muestra las respuestas que envían los dispositivos finales cuando salen del estado mencionado.

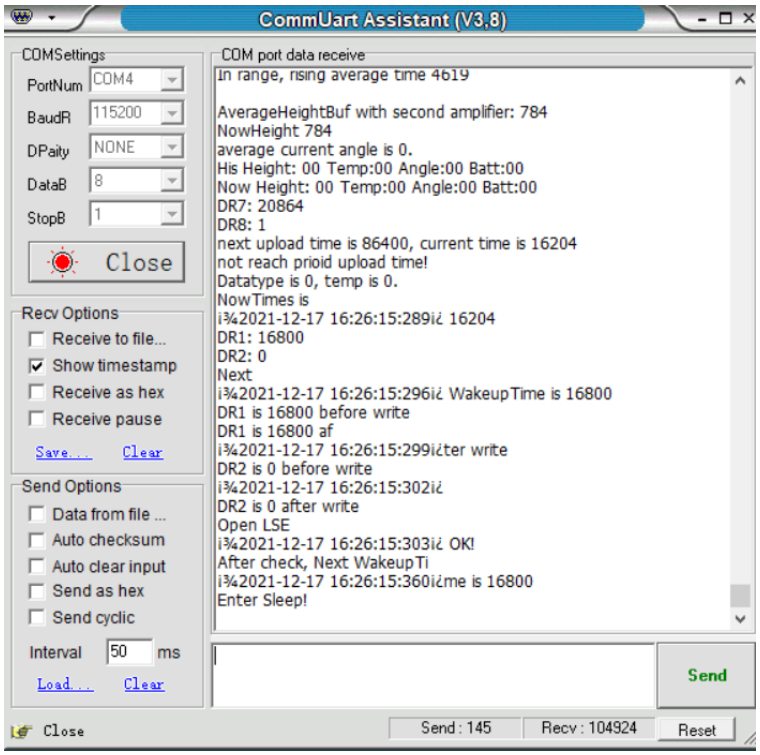


Figura 2.33 Respuesta de uno de los dispositivos al activarse.

Para configurar los periodos de medición se utilizó el comando en decimal 80029999080481, donde 04 representa el intervalo de activación. Una vez ejecutado, los dispositivos empezaron a realizar mediciones cada 4 minutos, como se puede ver en la siguiente figura la medición se notificó a las 18:16, y la próxima se programó para las 18:20.

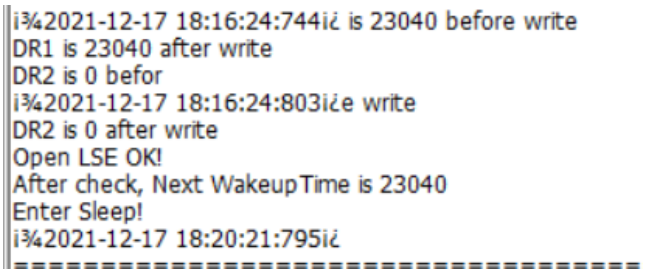


Figura 2.34 Respuesta desde el dispositivo.

Se estableció una alarma para la altura de medición de 30cm en los dispositivos finales. Para esto se utilizó el comando 80029999021E81, donde 1E representa este parámetro en decimal. Esto en la práctica permite que los dispositivos finales envíen un dato dentro de la carga útil del tipo boolean donde se especifica si la altura es igual o menor a la establecida, por esta razón se le considera una alarma.

Figura 2.35 Establecimiento de alarma de altura a 30cm.

Una vez que se comprobó y configuró los dispositivos finales se los registró en la plataforma de TTN, para que formen parte de la red LoRa. Para esto, en la consola web, apartado Applications, se seleccionó el servidor de red creado, y posteriormente End devices.

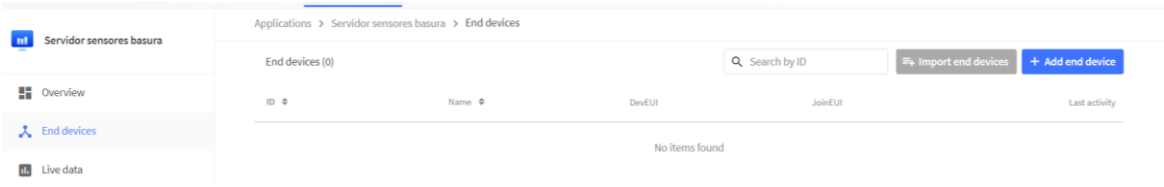


Figura 2.36 Opción End devices en servidor de aplicación.

Allí se hizo click sobre la opción Add end device, en la parte superior derecha. La plataforma de TTN tiene dos opciones para añadir dispositivos finales, la primera es a través de un repositorio de nodos ordenados por fabricantes, donde tan solo es necesario buscar el sensor en cuestión, de todas formas, aunque se encontró modelos del fabricante Dingtek, no estaba registrado el modelo DF703, por lo que se tuvo que utilizar el segundo modo, que es manual.

En esta opción se debe completar varios campos como plan de frecuencia, versión de LoRaWAN, parámetros regionales, modo de activación, clase, keys y contraseñas, entre otras, como lo muestra la siguiente Figura.

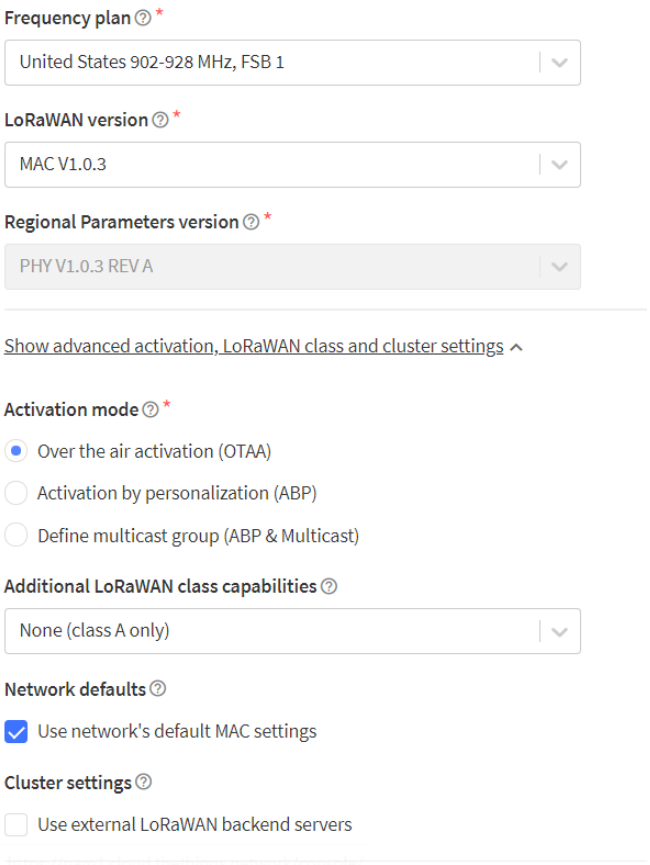


Figura 2.37 Configuración de parámetros de red para los dispositivos.

Para el segundo dispositivo se realizó el mismo procedimiento, pero cambiando únicamente su nombre.

Una vez registrados los dispositivos ejecutaron un proceso de join la próxima vez que salieron del modo sleep, y ya estaban conectados a la red y comunicándose con el resto de los componentes.

2.2.4 CONFIGURACIÓN DEL BRÓKER MQTT EN EL SERVIDOR DE APLICACIÓN

En este punto la red LoRaWAN ya se encontraba funcionando y sus componentes intercambiaban información entre sí. De todas maneras, antes de pasar a la etapa de codificación de la aplicación de escritorio fue necesario configurar el bróker MQTT en el servidor de aplicación, con el fin de implementar un suscriptor en el cliente.

Para esto se tuvo que dirigir a la opción de Integrations, en el panel de control del servidor de aplicación con nombre Servidor sensores basura.

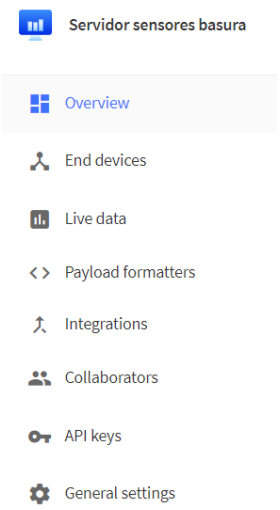


Figura 2.38 Opciones del panel de control del servidor de aplicación.

Una vez ahí se desplegaron las distintas opciones que ofrece TTN para integrar este servidor con aplicaciones externas, entre estas se encuentran: MQTT, Webhooks, Amazon Web Services IoT, Azure IoT Hub entre otras. De las de la lista anterior se hizo click sobre MQTT, ya que es el protocolo de interés para este proyecto de titulación

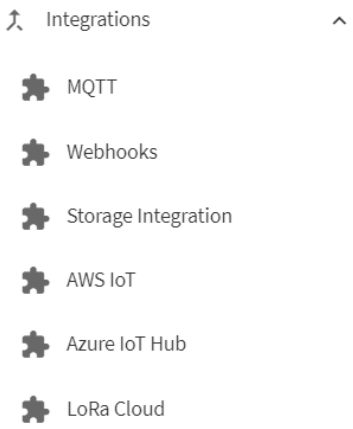


Figura 2.39 Plataformas de integración externa de TTN.

En seguida la plataforma mostró información relativa a este bróker, además de especificar que se debía crear una contraseña para que cualquier cliente externo se pudiera conectar a través de este protocolo. Al igual que con ciertas API keys anteriores, el usuario no se encarga de crear esta clave, sino que simplemente da la instrucción de que la plataforma la genere. En esta ventana además se mostró la dirección pública del servidor MQTT, dirección pública con TLS, nombre de usuario y el botón para generar la contraseña ya mencionada, donde se hizo click.

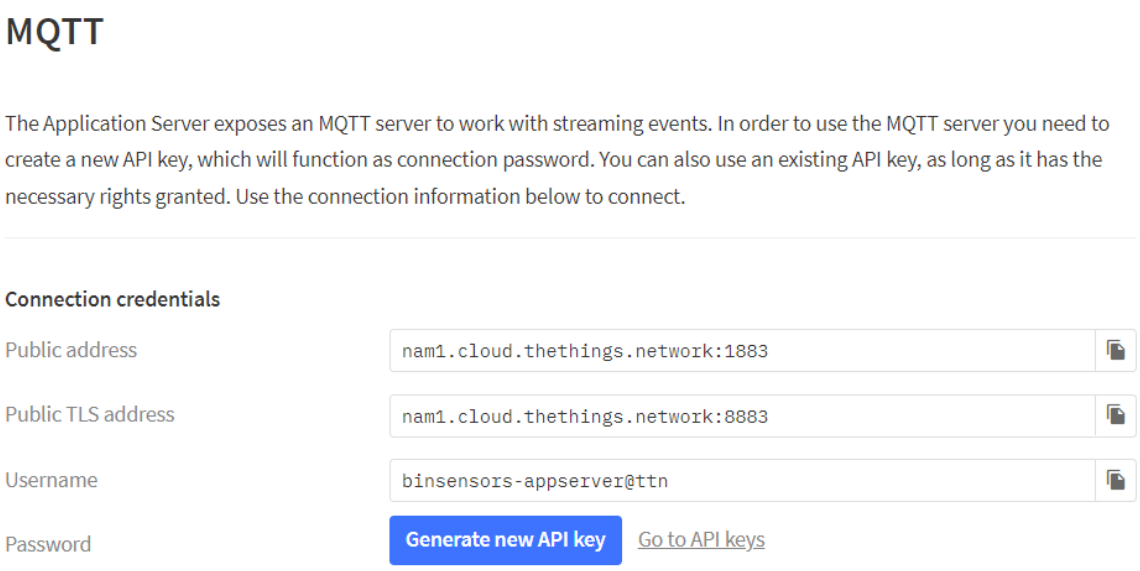


Figura 2.40 Información y configuración del servidor MQTT en TTN.

La clave generada por The Things Network se mostró de inmediato tras pulsar el botón azul de la figura anterior, y a diferencia de otras contraseñas manejadas por la plataforma esta se podía ver o copiar en cualquier momento en el futuro.

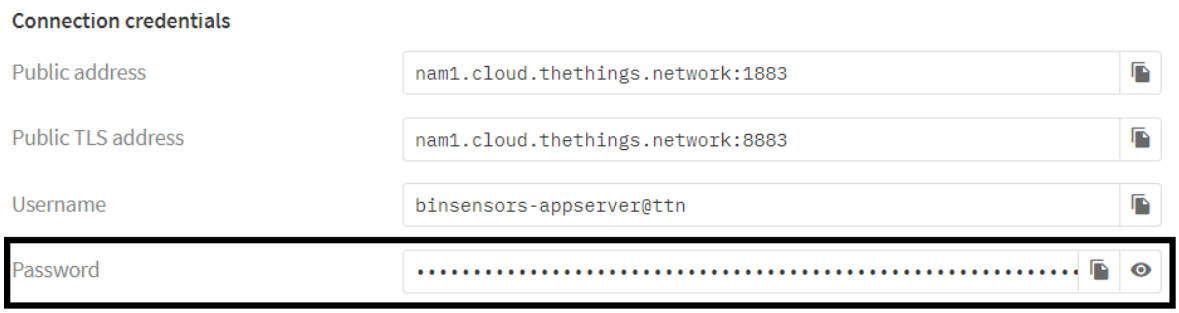


Figura 2.41 API key para conectarse desde un cliente externo mediante MQTT.

2.2.5 CODIFICACIÓN DEL CLIENTE MQTT MEDIANTE VISUAL STUDIO 2019

Uno de los requisitos definidos a través de las encuestas y que se muestra en la tabla 2.1 es que la interfaz gráfica y el cliente MQTT funcionen de manera independiente. De esta manera el servidor local, en este caso el computador de escritorio, podrá monitorear constantemente los datos de la red LoRaWAN y almacenarlos en la base de datos sin necesidad de ejecutar un programa que utilice más recursos del sistema, como es el caso de la mayoría de interfaces gráfica de usuario.

Para esto en Visual Studio 2019 se eligió la plantilla de proyecto Aplicación de consola (.NET Framework) en C#, como lo muestra el ícono resaltado con azul de la siguiente figura.

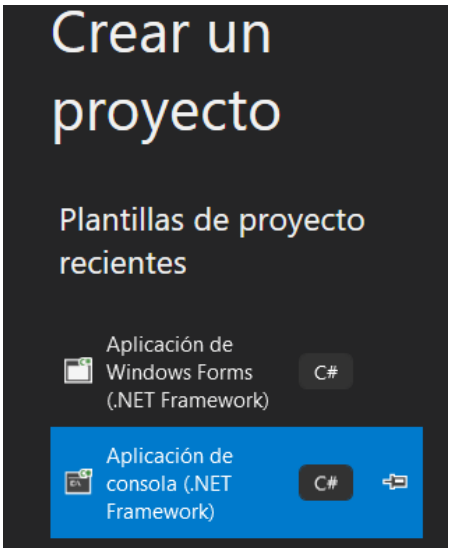


Figura 2.42 Selección del tipo de proyecto en VS2019.

La siguiente figura muestra las clases que se utilizaron para que esta parte de la aplicación funcione, y se explicarán detalladamente a continuación.

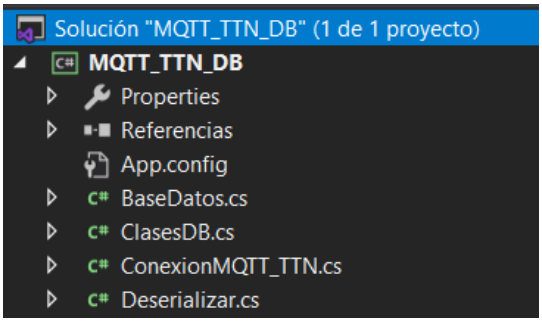


Figura 2.43 Clases para cliente MQTT y conexión a base de datos.

La clase con nombre ConexionMQTT_TTN.cs implementa los atributos y métodos necesarios para que la aplicación de consola se conecte con el bróker de The Things Network, para esto se utilizó la API M2Mqtt, tal como se definió en la Fase de Diseño.

Los métodos que incorpora esta API son suficientes para conectarse al bróker y mostrar los datos generados por los nodos y gateway de la red LoRaWAN, sin embargo, ya que el proyecto incorpora una base de datos es necesario deserializar esta información que se publica por defecto en formato JSON. Es por eso por lo que se utilizó Deserializar.cs, que es una representación de esta cadena de datos en forma de clases y atributos con sus respectivos métodos get y set.

La Figura 2.44 muestra una de estas cadenas JSON anidadas y sus partes generadas por el bróker MQTT a partir de la información que recibió el gateway desde los nodos, y tal como se puede observar, esta se encuentra compuesta por signos de puntuación y signos ortográficos entre los que se encuentran llaves, corchetes, comillas, dos puntos, entre otros donde se concentra la información de interés para este proyecto. Al ser una cadena demasiado larga resultó complejo transformarla en clases y atributos propios de C# de forma manual, por lo que se optó por utilizar un conversor online llamado JSON2CSharp (disponible en https://json2csharp.com), esta herramienta retornó el resultado mostrado en la Figura 2.45. Los dispositivos finales DF703 generan dos cadenas JSON las cuales son transmitidas mediante el bróker MQTT. La primera contiene información respecto a los nodos, gateways, conexión, radioenlace, cadena encriptada, cadena desencriptada según el código configurado en el servidor de aplicación que contiene datos de interés, entre otros tipos de mensajes. La segunda es bastante similar, excepto que no contiene la información de interés para este proyecto en su carga útil, sino datos respecto a la configuración del nodo, es decir, tiempos de medición, parámetros de las alarmas, a pesar de que esta cadena también es tratada los datos generados por esta no serán almacenados en la base de datos.

Figura 2.44 Cadena JSON publicada por el bróker MQTT de TTN.

```
public class ApplicationIds
  {
     public string application_id { get; set; }
  }

  public class EndDeviceIds
  {
     public string device_id { get; set; }
     public ApplicationIds application_ids { get; set; }
     public string dev_eui { get; set; }
     public string join_eui { get; set; }
     public string dev_addr { get; set; }
  }

  public class DecodedPayload
  {
     public string angle { get; set; }
     public bool battery { get; set; }
     public bool fire { get; set; }
     public bool full { get; set; }
     public string level { get; set; }
     public string temperature { get; set; }
     public bool tilt { get; set; }
  }

  public class GatewayIds
  {
     public string gateway_id { get; set; }
     public string eui { get; set; }
  }

  public class RxMetadata
  {
     public GatewayIds gateway_ids { get; set; }
     public string time { get; set; }
     public int timestamp { get; set; }
     public int rssi { get; set; }
     public int channel_rssi { get; set; }
     public double snr { get; set; }
     public string uplink_token { get; set; }
  }

  public class Lora
  {
     public int bandwidth { get; set; }
     public int spreading_factor { get; set; }
  }

  public class DataRate
  {
     public Lora lora { get; set; }
  }

  public class Settings
  {
     public DataRate data_rate { get; set; }
     public string coding_rate { get; set; }
     public string frequency { get; set; }
     public int timestamp { get; set; }
     public string time { get; set; }
  }

  public class NetworkIds
  {
     public string net_id { get; set; }
     public string tenant_id { get; set; }
     public string cluster_id { get; set; }
  }

  public class UplinkMessage
  {
     public string session_key_id { get; set; }
     public int f_port { get; set; }
     public int f_cnt { get; set; }
     public string frm_payload { get; set; }
     public DecodedPayload decoded_payload { get; set; }
     public List<RxMetadata> rx_metadata { get; set; }
     public Settings settings { get; set; }
     public string received_at { get; set; }
     public bool confirmed { get; set; }
     public string consumed_airtime { get; set; }
     public NetworkIds network_ids { get; set; }
  }

  public class Root
  {
     public EndDeviceIds end_device_ids { get; set; }
     public List<string> correlation_ids { get; set; }
     public string received_at { get; set; }
     public UplinkMessage uplink_message { get; set; }
  }
```

Figura 2.45 Cadena JSON transformada en lenguaje C#.

Se hizo una comparación entre la cadena original y el resultado en C# y se pudo notar que el transformador discriminó los datos y tomó únicamente los atributos de cada objeto JSON para hacerlos parte de distintas clases C#. Otra observación que se puede hacer al respecto de esta conversión es que la herramienta online creó una clase que a simple vista no formaba parte de la cadena original y la llamó Root. En el diagrama de clases de la Figura 2.1 se puede observar que de esta clase parten la mayoría de las demás, y es porque muchos de estos objetos JSON se encontraban anidados, es decir, formaban parte de otro objeto, y para representarlos en el lenguaje C# se los incluyó como atributos de esas clases padre, resultando Root como una especie de clase superior desde la cual se puede acceder al resto y por ende a sus atributos, que es lo que finalmente interesaba en el proyecto actual. De todas maneras, esta representación originada por la JSON2Csharp no fue adoptada a fidelidad, ya que como se puede observar en la figura anterior, no cumple con uno de los paradigmas de la POO, que es el encapsulamiento, por lo que se editó el código obtenido para establecer los atributos de las clases en privado, además de añadir los métodos correspondientes de lectura (get) y modificación (set) y constructores por defecto.

```
5 referencias
public class RootCon
{
    private EndDeviceIds end_device_ids;
    private List<string> correlation_ids;
    private string received_at;
    private UplinkMessage uplink_message;

    1 referencia
    public RootCon()
    {
    }

    7 referencias
    public EndDeviceIds End_device_ids { get => end_device_ids; set => end_device_ids = value; }
    2 referencias
    public List<string> Correlation_ids { get => correlation_ids; set => correlation_ids = value; }
    3 referencias
    public string Received_at { get => received_at; set => received_at = value; }
    26 referencias
    public UplinkMessage Uplink_message { get => uplink_message; set => uplink_message = value; }
}
```

Figura 2.46 Clase Root modificada y complementada en el código de la aplicación.

Otra de las motivaciones que impulsó a implementar esta clase fue que a pesar de que la aplicación de consola ya mostraba los mensajes con los métodos provistos por la API M2Mqtt lo hacía tal y como se muestra en la Figura 2.36, es decir, mostraba toda la cadena JSON, incluyendo datos de menor o nula relevancia para el usuario, por lo que se decidió adaptar una forma de extraer únicamente los datos de mayor relevancia para así indicarlos al usuario.

La clase ConexionMQTT_TTN.cs se puede considerar como principal ya que contiene el método Main, y aquí se encuentran los métodos implementados desde la API para el cliente MQTT, uno de estos se denomina Client_MqttMsgPublishReceived, el cual es del tipo void ya que no devuelve ningún valor, es asíncrono, estático y tiene como parámetros de entrada un objeto del tipo object y una clase del tipo MqttMsgPublishEventArgs, propia de M2Mqtt. El object, denominado sender en este proyecto, se utiliza para que un delegado en Main haga referencia al método Client_MqttMsgPublishReceived, mientras que MqttMsgPublishEventArgs es una clase que hereda de EventArgs e implementa atributos propios del cliente MQTT como tópico de suscripción, mensaje, bandera de duplicación de mensaje, nivel de QoS y retención de datos.

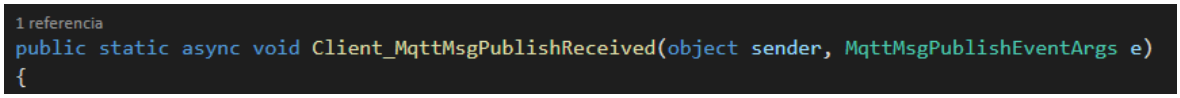


Figura 2.47 Método Client_MqttMsgPublishReceived.

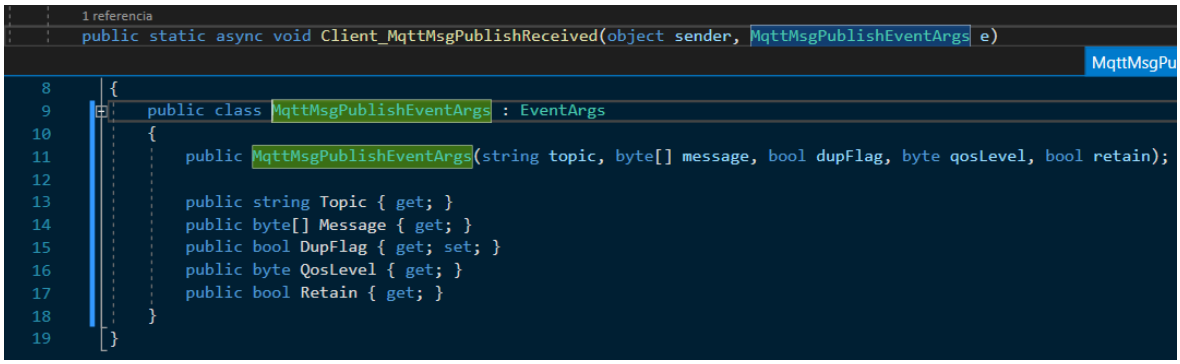


Figura 2.48 Clase MqttMsgPublishEventArgs.

El método anteriormente descrito se encarga de recibir y publicar los mensajes que capta el suscriptor, pero para esto hace falta definir un cliente MQTT por dispositivo final de red, los cuales se implementaron con la misma API en el método Main. Para esto se empezó por definir dos atributos de clase denominados clienteMqtt1 y clienteMqtt2, ambos del tipo MqttClient, además de 3 atributos del tipo string especificados a continuación:

- usuario: hace referencia al ID del servidor de aplicación creado en TTN, en este caso, binsensors-appserver.
- iDdispositivo1: hace referencia al ID del primer sensor añadido a la red LoRaWAN mediante el servidor de aplicación en TTN, en este caso, sensor1.
- iDdispositivo2: hace referencia al ID del segundo sensor añadido a la red LoRaWAN mediante el servidor de aplicación en TTN, en este caso, sensor2.

Más adelante se hará referencia a estos atributos en métodos y construcciones de objetos.

```
//creación de los clientes MQTT
private static MqttClient clienteMqtt1;
private static MqttClient clienteMqtt2;
//usuario correspondiente a la sección de integración MQTT
private static string usuario = "binsensors-appserver";
//ID de los dispositivos final en TTN
private static string idDispositivo1 = "sensor1";
private static string idDispositivo2 = "sensor2";
0 referencias
```

Figura 2.49 Atributos de clase implementados.

En el método Main se implementó métodos desde la clase MqttClient así como los objetos construidos a partir de los atributos declarados en la figura anterior. En la línea 44 del código de la Figura xx se le asigna a clienteMqtt1 como atributo la dirección del host de la Figura xx, además de inicializarlo como dato del tipo MqttClient. La línea 46 hace referencia al método Subscribe, el cual tiene como parámetros de entrada un arreglo del tipo string que hace referencia al tópico de suscripción al bróker y un arreglo de bytes correspondiente al nivel de QoS.

```
35      private static void Main(string[] args)
36      {
37          try
38          {
39              Console.WriteLine("Prueba de conexión a TTN mediante MQTT");
40
41              Console.WriteLine("Esperando mensajes");
42
43              //cliente con direccion del broker como atributo
44              clienteMqtt1 = new MqttClient("nam1.cloud.thethings.network");
45              //método para suscribirse a un tópico del bróker MQTT
46              clienteMqtt1.Subscribe(
47                  new[] { $"v3/{usuario}/devices/{idDispositivo1}/up"},
48                  new[] { MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE });
49
50              //Creación de delegados para eventos de suscripcion y publicacion de mensajes recibidos
51              clienteMqtt1.MqttMsgPublishReceived += Client_MqttMsgPublishReceived;
52
53              //Conexión a la API mediante la key correspondiente
54              //El método Guid.NewGuid() genera un ID para este suscriptor MQTT, necesario para conectarse
55              var response = clienteMqtt1.Connect(Guid.NewGuid().ToString(), usuario, "NNSXS.FJVGEVH6NAV4TF
56
57          }
```

Figura 2.50 Construcción de clienteMqtt1 del tipo MqttClient.

```
public ushort Subscribe(string[] topics, byte[] qosLevels);
```

Figura 2.51 Método Subscribe.

Como se puede observar en la línea 47 de la Figura xx el string contiene referencias a los atributos usuario e idDispositivo1, declarados anteriormente. En la línea 51 de la misma figura se declaró un delegado para que el método Client_MqttMsgPublishReceived procese los mensajes desde el publicador y los muestre a través de la consola. En la línea 55 de la misma figura se utilizó el método Connect, el cual contiene información relacionada al cliente: identificador, usuario y la clave creada previamente en la plataforma de TTN. Como se puede notar además se incluyó un segmento de control de excepciones, en caso de que exista algún error de conexión con el servidor.

La deserialización de la cadena JSON se implementó en el método Client_MqttMsgPublishReceived, aprovechando su funcionalidad, para esto se utilizó las librerías de Newtonsoft.Json, que a través de las clases convertidas en la Figura xx permitió asignar los datos de los objetos JSON a atributos C#.

```
//Librería para el tratamiento de los datos JSON
using Newtonsoft.Json;
using Newtonsoft.Json;
using Newtonsoft.Json.Converters;
```

Figura 2.52 Directivas para el uso de las librerías Newtonsoft.Json.

```
//Permite deserializar la cadena JSON recibida desde el bróker en el objeto C# Root llamado claseDeserializada
Root claseDeserializada = JsonConvert.DeserializeObject<Root>(datosJson);
```

Figura 2.53 Línea de código para deserializar objetos JSON.

La figura anterior muestra un objeto Root con nombre claseDeserilizada, datosJson es el nombre que se le dio al string que se recupera desde la clase MqttMsgPublishEventArgs que es argumento de entrada del método y DeserializeObject es el método que deserializa la cadena JSON, invocado desde las librerías mencionadas anteriormente.

Una de las ventajas encontradas en este método de deserialización es que cada uno de los datos de los objetos JSON se asignan automáticamente a su clase y atributo C# correspondiente, por lo que en este punto ya se podía navegar a través del objeto claseDeserializada para buscar el dato de interés que se desee mostrar al usuario, en las siguientes figuras se ilustra la comparación entre imprimir en consola la cadena JSON sin tratamiento o deserilización frente a la impresión selectiva de datos relevantes.

Figura 2.54 Cadena JSON sin tratamiento.

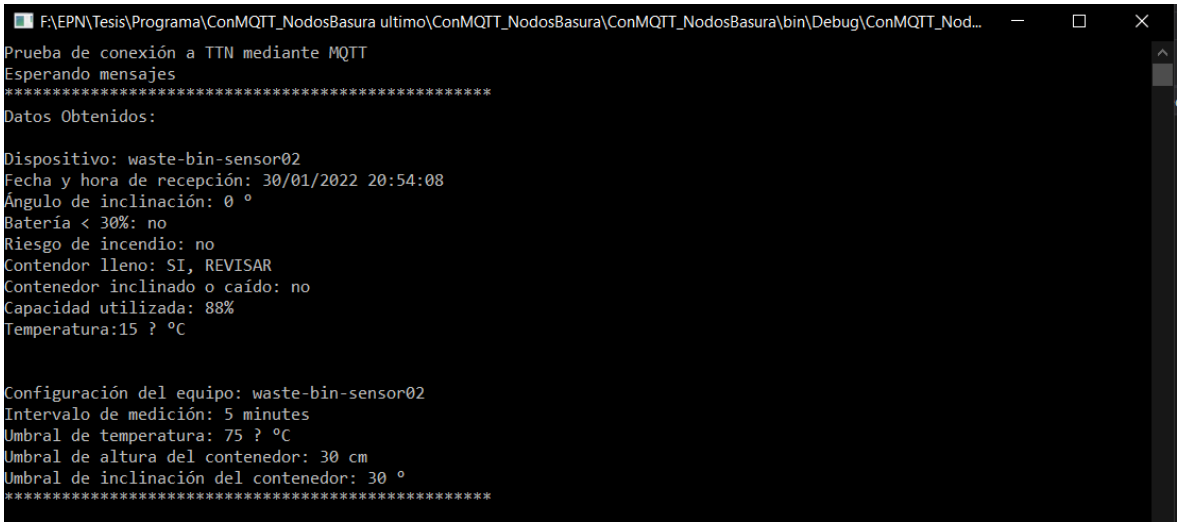


Figura 2.55 Impresión selectiva de datos tras deserializar.

2.2.6 IMPLEMENTACIÓN DE LA BASE DE DATOS EN MICROSOFT SQL SERVER

Tal como se decidió en fases anteriores el motor de base de datos que se utilizó en este proyecto fue Microsoft SQL Server a través del programa Microsoft SQL Server Management Studio. En esta sección se revisará parte del código que se utilizó para crear la base de datos, tablas, relaciones y ciertas consultas que se utilizaron para comprobar la correcta escritura de los datos. Además de esto se explicará el código implementado en la aplicación de consola de Visual Studio 2019 para la escritura de información en esta DB.

Siguiendo el diagrama relacional en la Fase de Diseño se crearon las tablas mediante la instrucción create table, para las claves principales se utilizó como tipo de dato int, la declaración identity (1,1) para todas las tablas, ya que cada uno de estos identificadores aumentaría de uno en uno por cada registro, empezando desde el número 1, además en la misma línea se codificó la instrucción not null para especificar que este registro no puede ser vacío y finalmente primary key, para indicar que es la clave primaria de la tabla. El resto de los atributos se escribieron según sea el tipo de dato correspondiente al

declarado en las clases C#, por ejemplo, para los tipo string en la base de datos se asignó varchar, para los tipo double se asignó float en la base de datos, entre otros casos. La siguiente figura muestra el código para la creación de la tabla Root, correspondiente a la clase con el mismo nombre en el proyecto de Visual Studio 2019.

```
create table tblRoot_(
rootID1 int identity(1,1) not null primary key,
end_device_ids int,
correlation_ids varchar(1000),
received_at DateTime,
uplink_message int
)
go
```

Figura 2.56 Creación de la tabla Root en la base de datos.

Para la creación de relaciones entre las tablas se utilizó las instrucciones alter table, para modificar la tabla, add constraint, para especificar que la modificación es del tipo foreign key y references para apuntar a la tabla que hace referencia la clave secundaria. Para el resto de tablas que requerían esta modificación se siguió el mismo procedimiento.

```
alter table tblRoot_
add constraint fk_Root_EndDeviceIds
foreign key (end_device_ids) references tblEndDevice_Ids(endDeviceId)
go
```

Figura 2.57 Inclusión de clave secundaria en la tabla Root.

La siguiente figura muestra una de las consultas codificadas para obtener información escrita en la base de datos que se consiguió desde los nodos.

```
SELECT tblRoot_.rootID1 as ID,
tblEndDevice_Ids.device_id as ID_Dispositivo,
tblDecoded_Payload.level as Nivel,
tblDecoded_Payload.battery as Bateria,
tblDecoded_Payload.fulll as Lleno,
tblDecoded_Payload.fire as Fuego,
tblDecoded_Payload.tilt as Caído,
tblRoot_.received_at as fecha
FROM tblDecoded_Payload
    INNER JOIN tblUplink_Message
ON tblDecoded_Payload.decodedPayloadID = tblUplink_Message.decoded_payload
    INNER JOIN tblRoot_
ON tblUplink_Message.uplinkMessageID = tblRoot_.uplink_message
    INNER JOIN tblEndDevice_Ids
ON tblRoot_.end_device_ids = tblEndDevice_Ids.endDeviceID
```

Figura 2.58 Consulta para obtener datos en la base de datos.

	ID	ID_Dispositivo	Nivel	Bateria	Lleno	Fuego	Caído	fecha
1	1	waste-bin-sensor01	647 mm	0	0	0	0	2022-01-14 12:43:07.000
2	2	waste-bin-sensor02	651 mm	0	0	0	0	2022-01-14 12:44:57.000
3	3	waste-bin-sensor01	647 mm	0	0	0	0	2022-01-14 12:48:08.000
4	4	waste-bin-sensor02	651 mm	0	0	0	0	2022-01-14 12:49:58.000
5	5	waste-bin-sensor01	227 mm	0	0	0	0	2022-01-14 12:53:09.000
6	6	waste-bin-sensor02	651 mm	0	0	0	0	2022-01-14 12:54:58.000
7	7	waste-bin-sensor01	227 mm	0	0	0	0	2022-01-14 12:58:09.000
8	8	waste-bin-sensor02	651 mm	0	0	0	0	2022-01-14 12:59:59.000
9	9	waste-bin-sensor01	648 mm	0	0	0	0	2022-01-14 13:03:10.000
10	10	waste-bin-sensor02	236 mm	0	0	0	0	2022-01-14 13:04:59.000

Figura 2.59 Resultados obtenidos desde la consulta de la figura anterior.

En el lado de la aplicación de consola, en Visual Studio 2019, fue necesario crear una clase denominada ClasesDB.cs, la cual contiene las mismas clases que Deserializar.cs, pero a cada una de estas se le añadió un identificador único, ya que por defecto los nodos no crean alguno para los objetos JSON que generan. Además en esta clase se implementó las instrucciones LINQ para especificar que cada una de estas representa una tabla en la base de datos anteriormente descrita, así como columnas, claves primarias y el código que especifica que estas son generadas por la DB. La siguiente figura muestra estas instrucciones para la clase Root, para el resto el procedimiento fue similar.

```
[Table(Name = "tblRoot_")]
2 referencias
public class RootCon
{
    [Column(IsPrimaryKey = true, IsDbGenerated = true)]
    private int rootID1;
    [Column]
    private int end_device_ids;
    [Column]
    private string correlation_ids;
    [Column]
    private DateTime received_at;
    [Column]
    private int uplink_message;



    0 referencias
    public RootCon()
    {
    }

    9 referencias
    public int End_device_ids { get => end_device_ids; set => end_device_ids = value; }
    0 referencias
    public string Correlation_ids { get => correlation_ids; set => correlation_ids = value; }
    8 referencias
    public DateTime Received_at { get => received_at; set => received_at = value; }
    9 referencias
    public int Uplink_message { get => uplink_message; set => uplink_message = value; }
    8 referencias
    public int RootID1 { get => rootID1; set => rootID1 = value; }

}
```

Figura 2.60 Implementación de instrucciones LINQ para las clases de ClasesDB.cs.

La clase BaseDatos.cs hereda de DataContext, que es una clase de LINQ que representa un punto de entrada para la transformación de datos desde este nombre de espacios a SQL. La línea 14 de la siguiente figura es la cadena de conexión con la base de datos, así el programa conoce las tablas y atributos a los cuales debe hacer referencia cuando se escriba algún dato, además se encuentran las tablas y la clase C# a la cual apuntan en el programa. Por ejemplo, Table<Root> tblRoot_ hace referencia a la clase Root en el actual espacio de nombres y tblRoot_ apunta a la tabla con el mismo nombre en la DB.

```
public class BaseDatos : DataContext
{
    2 referencias
    public BaseDatos() : base(@"Data Source=DESKTOP-H32
    public Table<RootCon> tblRoot_;
    public Table<UplinkMessage> tblUplink_Message;
    public Table<NetworkIds> tblNetwork_Ids;
    public Table<Settings> tblSettings_;
    public Table<DataRate> tblData_Rate;
    public Table<Lora> tblLora_;
    public Table<RxMetadata> tblRx_Metadata;
    public Table<GatewayIds> tblGateway_Ids;
    public Table<DecodedPayload> tblDecoded_Payload;
    public Table<EndDeviceIds> tblEndDevice_Ids;
    public Table<ApplicationIds> tblApplication_Ids;

}
```

Figura 2.61 String de conexión y referencia a tablas SQL.

En la clase ConexionMQTT_TTN.cs, se implementó una clase llamada MetodosDB, la cual describe una función para ejecutar procedimientos de escritura sobre la base de datos. Tiene como parámetros de entrada un objeto tipo Root llamado claseDeserilizada, que se obtiene desde el método Client_MqttMsgPublishReceived, además de un tipo de dato DateTime denominado fechaR, que se obtiene del mismo método anterior.

A diferencia del caso anterior, donde a partir de la deserialización se asignaban los atributos a cada clase de manera automática, en esta ocasión se tuvo que construir el objeto de clase a partir del objeto Root principal, ya que por defecto la base de datos no podría reconocer a que tabla pertenece cada dato y para mantener consistencia entre los registros. Como se muestra en la figura xx.

La siguiente figura ilustra lo descrito para dos tablas, tal como se mencionó, cada atributo se construye a través de un objeto vacío correspondiente a la clase en C# y se asigna mediante el objeto claseDeserializada y su clase correspondiente. Una vez que el objeto se terminó de construir se utilizó funciones de LINQ para insertar los datos en la tabla correspondientes, y finalmente se utilizó el método SubmitChanges() para que estos cambios sean ejecutados. El mismo proceso se repite para cada una de las tablas y se hace en orden ascendente, es decir, desde la clase última hasta la clase raíz.

```
ClasesDB.UplinkMessage upAux = new UplinkMessage();
upAux.Session_key_id = claseDeserializada.Uplink_message.Session_key_id;
upAux.F_port = claseDeserializada.Uplink_message.F_port;
upAux.F_cnt = claseDeserializada.Uplink_message.F_cnt;
upAux.Frm_payload = claseDeserializada.Uplink_message.Frm_payload;
upAux.Decoded_payload = decPayAux.DecodedPayloadID;
upAux.Rx_metadata = rxAux.RxMetadataID;
upAux.Settings = settAux.SettingsID;
upAux.Received_at = claseDeserializada.Uplink_message.Received_at;
upAux.Consumed_airtime = claseDeserializada.Uplink_message.Consumed_airtime;
upAux.Network_ids = netAux.NetworkIdsID;
miDB.tblUplink_Message.InsertOnSubmit(upAux);
miDB.SubmitChanges();
//Console.WriteLine(upAux.ToString());
ClasesDB.RootCon rootAux = new RootCon();
rootAux.End_device_ids = endDevAux.EndDeviceID;
rootAux.Uplink_message = upAux.UplinkMessageID;
rootAux.Received_at = fechaR;
rootAux.Correlation_ids = String.Join(",", claseDeserializada.Correlation_ids.ToArray());
miDB.tblRoot_.InsertOnSubmit(rootAux);
miDB.SubmitChanges();
```

Figura 2.62 Proceso de escritura de datos para las tablas tblUplink_Message y tblRoot_.

2.2.7 IMPLEMENTACIÓN DE LA INTERFAZ GRÁFICA

La implementación de la interfaz gráfica se basó en los bosquejos de las figuras 2.5 y 2.6, además de los requerimientos de la aplicación en la tabla 2.1. El objetivo de la interfaz, aparte de lo definido anteriormente, es que sea fácil de entender y manejar para el cliente final, y por último que a nivel visual sea agradable.

Para esto se utilizó un proyecto del tipo Windows Forms en Visual Studio 2019. Esta clase de proyecto ofrece herramientas del tipo GUI (Graphical User Interfaces) bajo .NET Framework, que permite implementar controles visuales y no visuales como TextBox, ComboBox, Label, ProgressBar, Button, DataGridView, entre otros.

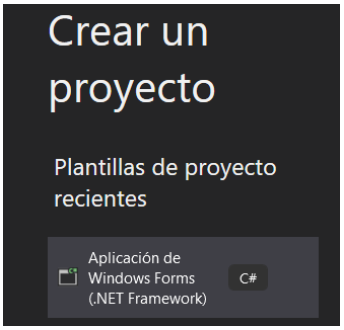
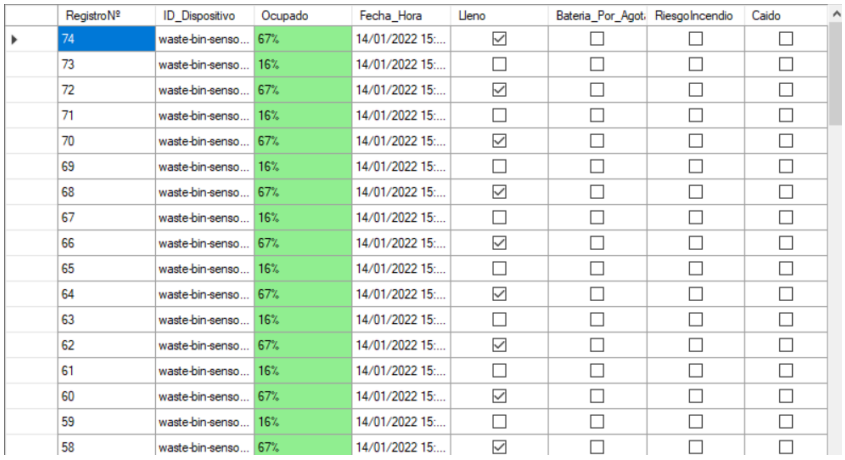


Figura 2.63 Creación de proyecto en VS2019.

La interfaz consta de dos ventanas, la principal muestra la información recuperada desde la base de datos en una tabla ordenada en filas y columnas. Como se estableció en la fase requerimientos se mostrará los siguientes datos: ID del dispositivo, nivel de ocupación en porcentaje, fecha y hora de la muestra, estado del contenedor (lleno o no), estado de la batería, riesgo de incendio y si se encuentra caído o no, además de número de registro. Estos datos se muestran mediante el control visual denominado DataGridView, ideal para información obtenida desde una base de datos.



RegistroNº	ID_Dispositivo	Ocupado	Fecha_Hora	Lleno	Bateria_Por_Agot	RiesgoIncendio	Caido
74	waste-bin-senso...	67%	14/01/2022 15:...	☑	☐	☐	☐
73	waste-bin-senso...	16%	14/01/2022 15:...	☐	☐	☐	☐
72	waste-bin-senso...	67%	14/01/2022 15:...	☑	☐	☐	☐
71	waste-bin-senso...	16%	14/01/2022 15:...	☐	☐	☐	☐
70	waste-bin-senso...	67%	14/01/2022 15:...	☑	☐	☐	☐
69	waste-bin-senso...	16%	14/01/2022 15:...	☐	☐	☐	☐
68	waste-bin-senso...	67%	14/01/2022 15:...	☑	☐	☐	☐
67	waste-bin-senso...	16%	14/01/2022 15:...	☐	☐	☐	☐
66	waste-bin-senso...	67%	14/01/2022 15:...	☑	☐	☐	☐
65	waste-bin-senso...	16%	14/01/2022 15:...	☐	☐	☐	☐
64	waste-bin-senso...	67%	14/01/2022 15:...	☑	☐	☐	☐
63	waste-bin-senso...	16%	14/01/2022 15:...	☐	☐	☐	☐
62	waste-bin-senso...	67%	14/01/2022 15:...	☑	☐	☐	☐
61	waste-bin-senso...	16%	14/01/2022 15:...	☐	☐	☐	☐
60	waste-bin-senso...	67%	14/01/2022 15:...	☑	☐	☐	☐
59	waste-bin-senso...	16%	14/01/2022 15:...	☐	☐	☐	☐
58	waste-bin-senso...	67%	14/01/2022 15:...	☑	☐	☐	☐

Figura 2.64 DataGridView con datos recuperados.

Otro control visual utilizado fue ComboBox, que en este caso despliega una lista de los nodos en el sistema, que al seleccionarlos muestra únicamente los datos generados por ese sensor, de todas maneras también incluye una opción para mostrar los datos de todos los nodos. Otro control visual utilizado en esta parte fue label, para indicar al usuario la existencia de este ComboBox y lo pueda utilizar.

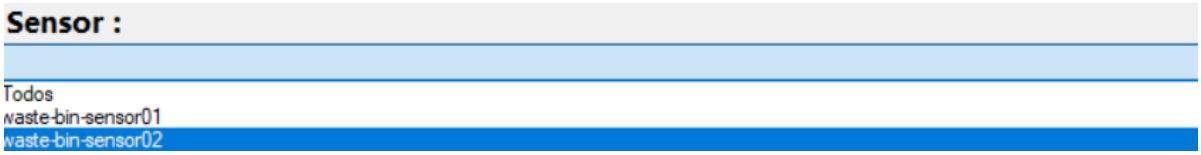


Figura 2.65 ComboBox y Label utilizado.

En la parte izquierda de esta ventana se utilizó controles visuales de imágenes, labels, timers, y un control especial de notificaciones. Con el control denominado pictureBox se colocó una imagen del logo de la Escuela Politécnica Nacional, como añadido visual. También se incluyó un label como título del programa, bajo estos se utilizó otros pictureBox para añadir imágenes que servirían como botones: el primero de arriba hacia abajo se utiliza para actualizar los datos en el DataGridView mientras que segundo notifica al usuario sobre la implementación del servicio de mapas en el programa, esto se puede ver en la Figura 2.66.

Figura 2.66 Controles visuales en la parte izquierda de la ventana principal.

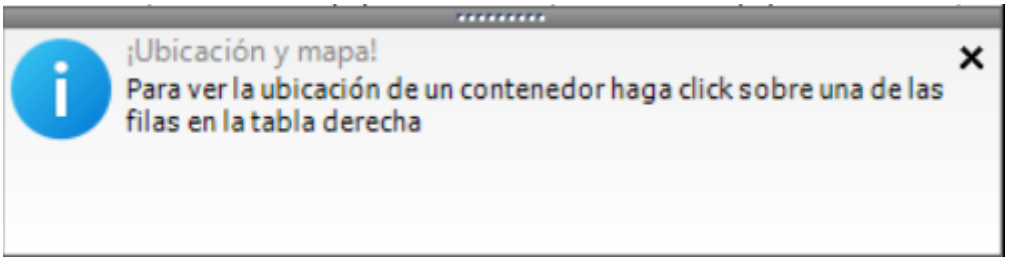


Figura 2.67 Aviso de implementación del servicio de mapas.

Otro elemento visual utilizado fue el control denominado timer, que aunque no se muestra directamente en la ventana se utiliza junto con otros segmentos de código para hacer funcionar un reloj, además de un label para obtener la fecha actual.

La segunda ventana de esta interfaz gráfica muestra la ubicación de los contenedores mediante un mapa, para este proyecto se utilizó ubicaciones referenciales, sin embargo, otras versiones de los nodos implementan un módulo GPS para obtener la localización exacta de estos.

Para acceder a esta ventana se ideó varias opciones, como ingresar mediante un botón, a través del mismo ComboBox anteriormente explicado, pero al final se decidió utilizar un evento generado por el DataGridView donde se muestran los datos de los nodos.

El evento tiene como nombre SelectionChanged, y ejecutará determinada tarea cada vez que se haga click sobre una fila en el cuadro de datos, en este caso ejecuta el formulario de mapas, enviando una longitud y latitud la cual graficará mediante la API Gmap.NET. El segmento de código correspondiente se muestra en la siguiente figura.

```
private void dgvDatos_SelectionChanged(object sender, EventArgs e)
{
    foreach (DataGridViewRow row in dgvDatos.SelectedRows)
    {

        string nodo = row.Cells[1].Value.ToString();

        if (nodo == "waste-bin-sensor01")
        {
            double newLat = Convert.ToDouble(-0.2197791);
            double longit = Convert.ToDouble(-78.5070088);

            FrmUbicacion frmGraf = new FrmUbicacion(newLat, longit, nodo);
            frmGraf.Show();
        }
        else if (nodo == "waste-bin-sensor02")
        {
            double newLat = Convert.ToDouble(-0.2067698);
            double longit = Convert.ToDouble(-78.4876112);

            FrmUbicacion frmGraf = new FrmUbicacion(newLat, longit, nodo);
            frmGraf.Show();
        }
    }
}
```

Figura 2.68 Código para abrir una nueva ventana el mapa de la ubicación del contenedor seleccionado.

Esta ventana en su parte izquierda muestra, además del logo y el título del proyecto, el identificador único del sensor correspondiente a algún contenedor y la capacidad actualmente ocupada. Aunque al inicio se podía acceder a este mapa de localización mediante cualquier fila del DataGridView, esto se cambió debido a errores con el método utilizado, lo cual será explicado en el siguiente capítulo. Para el mapa se utilizó el control llamado GMapControl, incorporado en la API GMap, este grafica determinada ubicación basándose en una latitud y longitud dada. Se añadió un marcador para indicar la localización exacta de las coordenadas, el cual adoptará un color dependiendo de el nivel de ocupación: verde-bajo, amarillo-medio, rojo-ocupado. El control permite implementar este mapa mediante distintas fuentes como Google, Bing, Yahoo, Yandex, entre otros, por familiaridad se eligió el primero. La siguiente figura muestra esta parte del proyecto implementada y mostrando la ubicación del sensor waste-bin-sensor02.

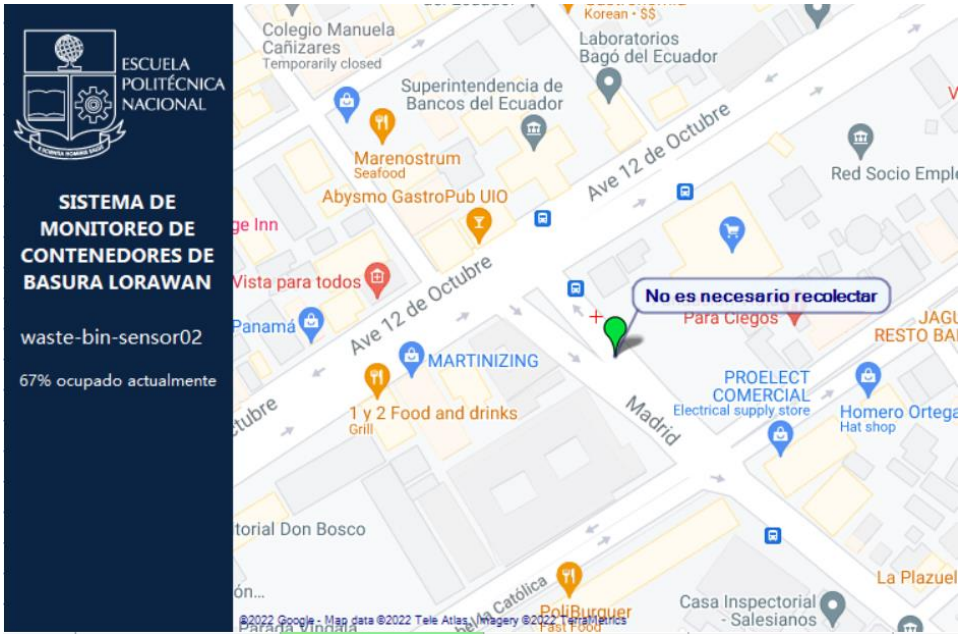


Figura 2.69 Ubicación referencial del sensor waste-bin-sensor02.

```
//Inicialización de servicios
mapActividad.MapProvider = GMapProviders.GoogleMap;
mapActividad.DragButton = MouseButtons.Left;
mapActividad.CanDragMap = true;
mapActividad.Position = new PointLatLng(latitud, longitud);
mapActividad.MinZoom = 0;
mapActividad.MaxZoom = 24;
mapActividad.Zoom = 18;
mapActividad.AutoScroll = true;

//Marcador de posición
//con esto se coloca el marcador azul
overlayMarcador = new GMapOverlay("Marcador");
```

Figura 2.70 Extracto de código para inicializar el mapa, ubicación y controles de zoom

3. RESULTADOS Y DISCUSIÓN

3.1 PRUEBAS DE RED

3.1.1 COMPROBACIÓN DE ENLACE ENTRE NODOS, GATEWAY Y SERVIDORES TTN

El propósito de esta etapa fue verificar que dispositivos finales, gateway y servidores se comuniquen entre sí e intercambien mensajes, sin embargo, no se utilizó para comprobar los datos obtenidos desde los nodos sensores. Para esto se utilizó la herramienta de computador personal Uartassist, la interfaz gráfica del Laird RG191 y la de The Things Network.

Para empezar se comprobó el enlace entre los nodos y el concentrador de dispositivos, debido a que solo se contó con un conversor USB a TTL se realizó la verificación un dispositivo a la vez. Con los elementos de la red LoRaWAN energizados y uno de los sensores conectados al entorno de gestión se ejecutó el programa Uartassist y mediante un navegador web se accedió a la interfaz gráfica del gateway. Los resultados correspondientes al nodo llamado waste-bin-sensor01 se explican a continuación.

```
SendData Ok
After sending data!
NowTimes is 22
DR1: 300
DR2: 0
Next WakeupTime is 300
DR1 is 300 before write
DR1 is 300 after write
DR2 is 0 before write
DR2 is 0 after write
Open LSE OK!
After check, Next WakeupTime is 300
Enter Sleep!
```

Figura 3.1 Resultados obtenidos desde el programa Uartassist para el sensor waste-bin-sensor01.

La Figura 3.1 muestra una parte de los datos generados por el primer nodo testeado en el programa, la información proporcionada por esta aplicación es extensa, ya que traduce todo lo generado por el sensor como parámetros de configuración, parámetros de red, mediciones obtenidas, modos de funcionamiento y otros. De todas maneras, para esta sección del documento, la información más relevante se encuentra al final del reporte, ya que aquí se informa si los datos fueron enviados a través del gateway. Y tal como se puede apreciar en la primera línea de la figura anterior, esto se confirma mediante el texto SendData Ok.

En el lado del gateway, a través de una consola incorporada en la interfaz gráfica, se puede apreciar ciertos datos cada vez que el nodo estableció comunicación con el nodo, esta información corresponde puramente a aspectos de la comunicación como frecuencia utilizada, potencia de transmisión, tiempo en el aire de la señal, sin embargo para verificar esta comunicación se hizo énfasis en un campo, el cual especifica si la transmisión fue autorizada o no, esto se puede verificar en la cuarta línea de la siguiente figura.

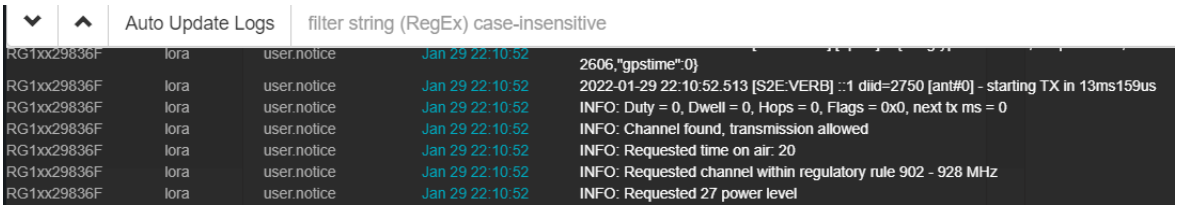


Figura 3.2 Mensajes mostrados en la consola del gateway Laird RG191 para el nodo waste-bin-sensor.01.

Para el dispositivo nombrado waste-bin-sensor02 se repitió el mismo proceso y se evaluó los resultados de manera similar, es importante mencionar que los dispositivos finales previamente fueron testeados sin la presencia del gateway en la red, obteniendo la línea SendData Failed en lugar de SendData Ok mostrada en la Figura 3.1. Las siguientes figuras muestran los resultados obtenidos para este segundo nodo.

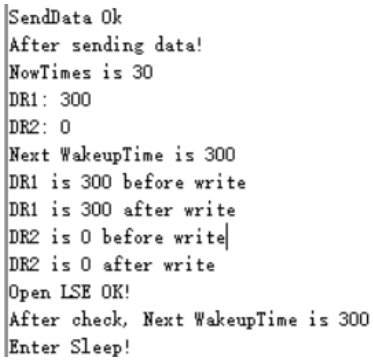


Figura 3.3 Resultados obtenidos del programa Uartassist para el nodo waste-bin-sensor02.

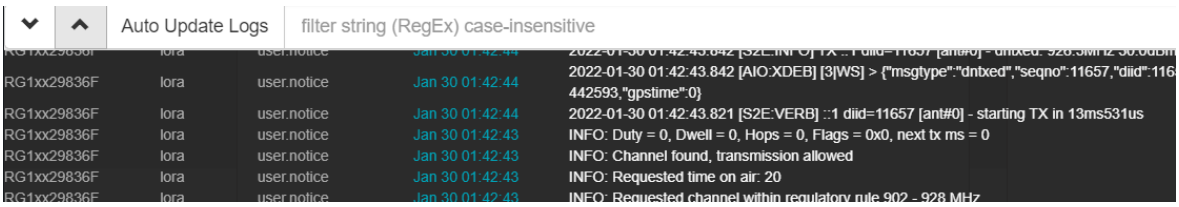


Figura 3.4 Mensajes mostrados en la consola del gateway Laird RG191 para el nodo waste-bin-sensor02.

Para comprobar que el gateway retransmitía los datos generados por los nodos hasta los servidores de The Things Network se utilizó la consola provista por la plataforma, en la opción Live data, y se energizó los dos dispositivos a la vez además de el concentrador de dispositivos. Para esta etapa de pruebas se estableció periodos de 5 minutos entre medición por nodo. En las siguientes figuras se muestran los resultados obtenidos para cada nodo.

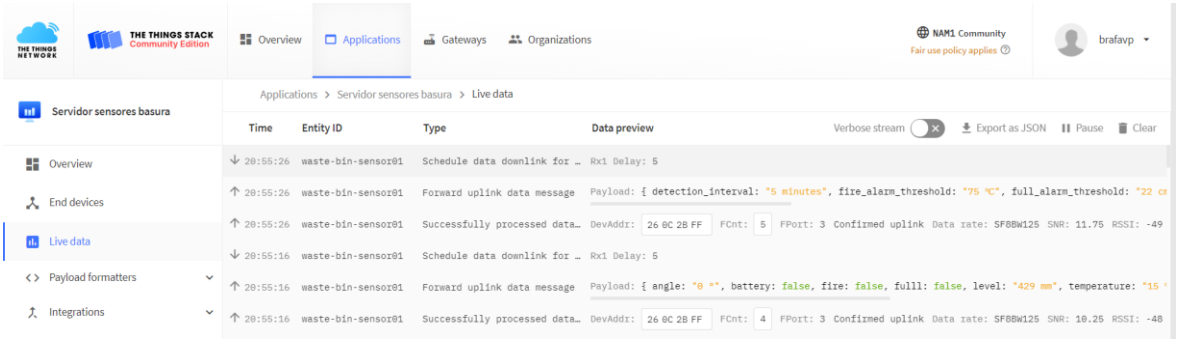


Figura 3.5 Datos del nodo waste-bin-sensor01 en la consola de TTN.

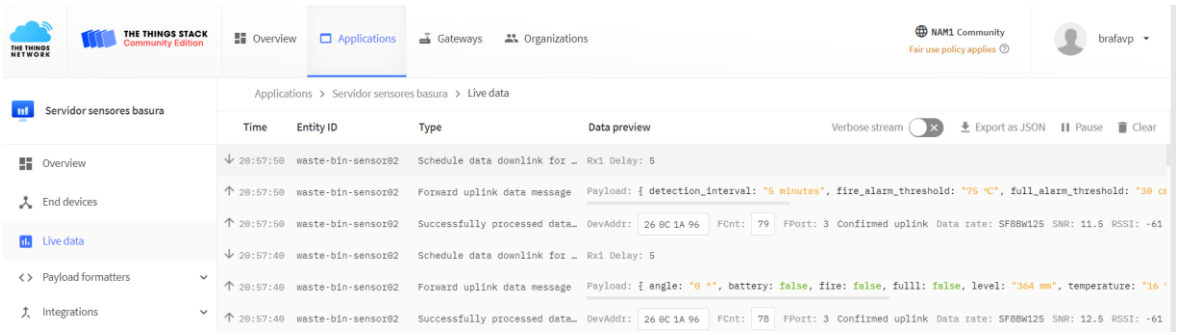


Figura 3.6 Datos del nodo waste-bin-sensor02 en la consola de TTN.

Las dos figuras anteriores muestran como la plataforma de TTN presenta los datos generados por los sensores y reenviados a través del gateway mediante el Internet. Cada vez que los dispositivos finales salen del modo ultra ahorro de energía generan dos segmentos o cadenas de datos, la primera contiene información respecto a la configuración de estos, mientras que la segunda muestra los datos de interés, es decir, las mediciones obtenidas mediante sus sensores. Por ahora no se analizará estos datos, pero mediante esta prueba se pudo afirmar que la red se encontraba activa y funcionando con las configuraciones realizadas en la fase de implementación.

3.1.2 CAPTURA DE TRAMAS GENERADAS POR LOS DISPOSITIVOS DE RED EN UN EXTREMO DEL SISTEMA

En esta fase se comprobó que los datos sean capaces de llegar hasta el entorno de gestión, para esto se utilizó un cliente MQTT en dicho extremo del sistema. Con todos los elementos de la red LoRaWAN funcionando se ejecutó este cliente mediante un terminal del sistema Windows 10, previamente se instaló el bróker MQTT Eclipse Mosquitto, pero únicamente se utilizó los comandos correspondientes a la parte del cliente.

El primer comando que se ejecutó fue *cd C:\Program Files\mosquitto*, para dirigirse al directorio donde se instaló Eclipse Mosquitto, posteriormente se ejecutó el comando *mosquitto_sub -h nam1.cloud.thethings.network -t "#" -u "binsensors-appserver@ttn" -P "XXX…" -d*, donde “XXX…” fue la API key correspondiente al bróker MQTT en la plataforma de TTN. De inmediato el cliente empezó el proceso de autenticación y conexión, y una vez que el proceso fue aprobado se pudo ver los mensajes que generaban los nodos a través de este terminal.

```
Client mosq-1XmQ794ut3dKJqyqfN sending CONNECT
Client mosq-1XmQ794ut3dKJqyqfN received CONNACK (0)
Client mosq-1XmQ794ut3dKJqyqfN sending SUBSCRIBE (Mid: 1, Topic: #, QoS: 0, Options: 0x00)
Client mosq-1XmQ794ut3dKJqyqfN received SUBACK
Subscribed (mid: 1): 0
Client mosq-1XmQ794ut3dKJqyqfN sending PINGREQ
Client mosq-1XmQ794ut3dKJqyqfN received PINGRESP
Client mosq-1XmQ794ut3dKJqyqfN received PUBLISH (d0, q0, r0, m0, 'v3/binsensors-appserver@ttn/devices/waste-bin-sensor01/up',
... (1489 bytes))
```

Figura 3.7 Proceso de autenticación entre cliente y bróker MQTT.


```
{"end_device_ids":{"device_id":"waste-bin-sensor01","application_ids":{"application_id":"binsensors-appserver"},"dev_eui":"8CF
95720000586E5","join_eui":"8CF9572000000000","dev_addr":"260CD1D3"},"correlation_ids":["as:up:01FTPAWJR2ASVVH82HKKNNRTAC","gs:
conn:01FTNWQN1KY21JX6EK6H26PJR1","gs:up:host:01FTNWQN654WE27ZKXA3VPNYHW","gs:uplink:01FTPAWJHJFWW6Q36JW46NJSQT","ns:uplink:01F
TPAWJHMPPC0E4TFY4TB106V","rpc:/ttn.lorawan.v3.GsNs/HandleUplink:01FTPAWJHMC7Y01M7FW36Y8561","rpc:/ttn.lorawan.v3.NsAs/HandleUp
link:01FTPAWJR1A0M7GKS91YMQN3V0"],"received_at":"2022-01-30T20:30:10.692608686Z","uplink_message":{"session_key_id":"AX6sgn0xI
PBQMjUVxBBZgQ==","f_port":3,"f_cnt":21,"frm_payload":"gAABAxkGCRgFFkseAAICAAAAAAAAAAAAgQ==","decoded_payload":{"detection_inte
rval":"5 minutes","fire_alarm_threshold":"75 Ôãâ","full_alarm_threshold":"22 cm","periodic_interval":"24 hours","tilt_alarm_th
reshold":"30 T","tilt_enable":false,"ultrasonic_range":"5000 mm"},"rx_metadata":[{"gateway_ids":{"gateway_id":"rg1xx-rvp","eu
i":"C0EE40FFFF29836F"},"time":"2022-01-30T20:30:10.313622951Z","timestamp":1955366916,"rssi":-54,"channel_rssi":-54,"snr":8.75
,"uplink_token":"ChcKFQoJcmcxeHgtcnZwEgjA7kD//ymDbxCEkLKkBxoMCNLq248GEPfKkeYBIKCfzaf0rwMqDAjS6tuPBhCng8aVAQ=="}],"settings":{"
data_rate":{"lora":{"bandwidth":125000,"spreading_factor":8}},"coding_rate":"4/5","frequency":"904300000","timestamp":19553669
16,"time":"2022-01-30T20:30:10.313622951Z"},"received_at":"2022-01-30T20:30:10.484318930Z","confirmed":true,"consumed_airtime"
:"0.143872s","network_ids":{"net_id":"000013","tenant_id":"ttn","cluster_id":"ttn-nam1"}}}
```


Figura 3.8 Mensaje generado por el nodo waste-bin-sensor01 mostrado mediante el terminal de Windows 10.

El propósito de este segmento de pruebas fue verificar que mediante la configuración de un cliente MQTT en el sistema operativo Windows 10 se podía suscribir a los tópicos publicados por el bróker MQTT en la plataforma de The Things Network. Esto garantiza que el programa de escritorio puede ser ejecutado en cualquier dispositivo bajo este sistema operativo, o incluso, que se puede acceder a estos datos mediante un cliente MQTT en distintos sistemas operativos o plataformas.

3.2 PRUEBAS DE SOFTWARE

Para la fase de pruebas de software se retomó la metodología en cascada, utilizada en la fase de diseño e implementación.

3.2.1 PRUEBAS INICIALES

En esta etapa la aplicación de escritorio fue puesta a prueba por el autor del trabajo de titulación, con la intención de buscar posibles errores. Estas pruebas se hicieron en tiempo real, es decir, mientras toda la red LoRaWAN se encontraba en funcionamiento. Ya que la aplicación consta de dos partes se ejecutó cada una por separado, empezando por la de consola, cuya función es almacenar la información recibida desde el bróker MQTT en la base de datos y mostrar los datos más relevantes en la pantalla.

La primera parte del programa se ejecutó sin inconvenientes al inicio, sin embargo no cumplía la funcionalidad de recibir los mensajes MQTT desde TTN, esto se debía a que para la V3 de esta plataforma el formato de los tópicos a suscribirse había cambiado en comparación a su versión anterior, para corregir este problema se actualizó el código del programa, quedando la sección de la siguiente figura.

```
//método para suscribirse a un tópico del bróker MQTT
clienteMqtt.Subscribe(
    new[] { $"v3/{usuario}/devices/{idDispositivo1}/up" },
    new[] { MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE });
```

Figura 3.9 Tópico de suscripción actualizado a la versión 3 de TTN.

Otro error encontrado durante esta etapa tuvo que ver con el tipo de dato declarado para un atributo de las clases transformadas desde las cadenas JSON. Esto se pudo originar por la naturaleza de la herramienta JSON2CSharp, que interpreta el objeto JSON y lo asigna a un tipo determinado C#. En este caso el atributo timestamp de las clases RxMetadata y Settings en un inicio se encontraba declarado como int, pero al momento de realizar las pruebas se encontró que si la distancia entre nodos y gateway aumentaba este atributo podía incrementar sustancialmente su valor, por tal motivo y considerando que la distancia entre estos dispositivos puede ser de hasta 1km se decidió modificar este tipo de dato por el mayor disponible en el lenguaje de programación, asignándole como ulong.

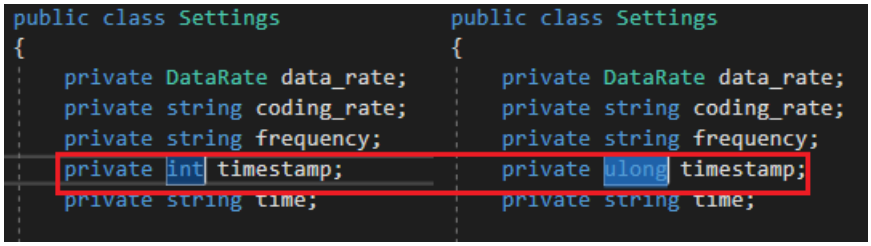


Figura 3.10 Tipo de dato cambiado tras encontrar el error.

Este error también afectó a las tablas correspondientes en la base de datos, ya que los tipos de datos no debían diferir con el fin que LINQ no tenga conflictos al momento de guardar los datos, debido a esto también se alteró su código, asignándole el correspondiente del lenguaje C# a SQL.

```
create table tblSettings_(
settingsID int identity(1,1) not null primary key,
data_rate int,
coding_rate varchar(10),
frequency varchar(20),
timestamp bigint,
time varchar(50)
)
go
```

Figura 3.11 Cambio en la tabla tblSettings_ de la base de datos.

Durante esta fase se encontró que el publicador MQTT anunciaba los datos a los suscriptores de los tópicos con horario UTC (Hora universal coordinada), por lo que fue necesario implementar una sección de código que traduzca este dato a la hora local, o UTC -05:00, para guardar correspondencia. En la siguiente figura se ilustra lo mencionado en el párrafo actual.

```
string fecha = rootAux.Received_at.Substring(0, 19);
fecha = fecha.Replace("T", " ");
DateTime fechaD = DateTime.ParseExact(fecha, "yyyy-MM-dd HH:mm:ss", System.Globalization.CultureInfo.InvariantCulture);
TimeZoneInfo ecTime = TimeZoneInfo.FindSystemTimeZoneById("SA Pacific Standard Time");
fechaD = TimeZoneInfo.ConvertTimeFromUtc(fechaD, ecTime);
```

Figura 3.12 Segmento de código para adaptar la hora a la correspondiente local.

En este segmento de la aplicación no se encontraron más errores, por lo que se pudo pasar al de la interfaz gráfica.

Aquí se utilizó consultas LINQ para acceder a la información la base de datos, de todas maneras al ejecutar la aplicación esta no fue cargada correctamente en el DataGridView, tras indagar se encontró que el error se originó porque este control visual estaba conectado a la base de datos de manera incorrecta; esto quiere decir que se enlazó directamente cuando esto se debía hacer mediante código para obtener los resultados deseados.

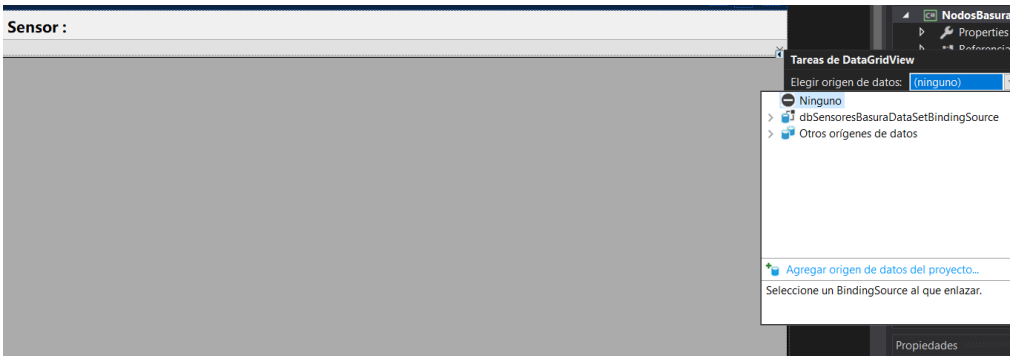


Figura 3.13 Corrección de conexión.

```
var regTod = (from iterA in this.miDB.tblDecoded_Payload.AsEnumerable()
              join iterB in this.miDB.tblUplink_Message
              on iterA.DecodedPayloadID equals iterB.Decoded_payload
              join iterC in this.miDB.tblRoot_.AsEnumerable()
              on iterB.UplinkMessageID equals iterC.Uplink_message
              join iterD in this.miDB.tblEndDevice_Ids.AsEnumerable()
              on iterC.End_device_ids equals iterD.EndDeviceID


              select new
              {
                  RegistroNº = iterC.RootID1,
                  ID_Dispositivo = iterD.Device_id,
                  Ocupado = 100 - Convert.ToInt32(iterA.Level.Replace("mm", " ")) / 8 + "%",
                  Fecha_Hora = iterC.Received_at,
                  Lleno = iterA.Fulll,
                  Bateria_Por_Agotarse = iterA.Battery,
                  RiesgoIncendio = iterA.Fire,
                  Caido = iterA.Tilt

              }).OrderByDescending(x => x.RegistroNº).ToList();
```

Figura 3.14 Consulta utilizada para acceder a las tablas y ordenar los datos.

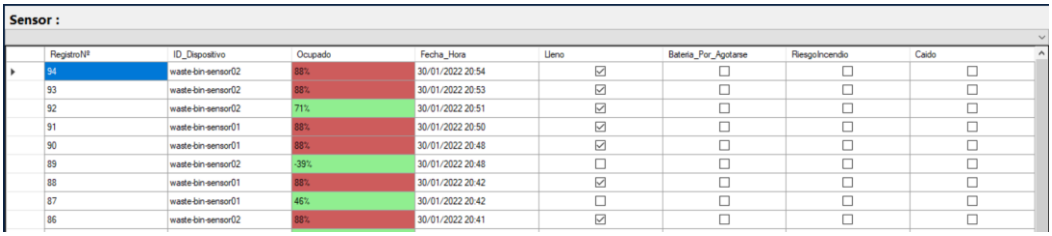


Figura 3.15 Resultado de la corrección.

Dentro de la consulta en la figura 3.14 se encontró otro error de forma, ya que el sensor de los dispositivos finales devuelve el dato de altura en milímetros con respecto a su otro extremo, o en otras palabras, indica la capacidad libre de los contenedores, y tal como se estableció la interfaz debía mostrar la ocupación del contenedor en porcentaje. Con ese fin se modificó una línea de la consulta, introduciendo operaciones matemáticas al respecto.

```
Ocupado = 100 - Convert.ToInt32(iterA.Level.Replace("mm", " ")) / 8 + "%",
```

Figura 3.16 Operaciones sobre un dato de la DB para mostrar la ocupación.

Otro problema encontrado fue que al momento de elegir el sensor o dispositivo final todos los datos anteriores se acumulaban en el DataGridView, es decir, no se borraban los anteriores, por lo cual se añadió una línea de código en cada caso para previamente limpiar esta tabla.

```
else if (cbxSensores.Text == "waste-bin-sensor01")
{
    dgvDatos.Refresh();
```

Figura 3.17 Línea de código para reiniciar el DataGridView.

En la parte visual se encontraron ciertos problemas con la ubicación de etiquetas e imágenes, sin embargo esto se fue corrigiendo en la marcha, obteniendo el resultado final mostrado en la siguiente figura.

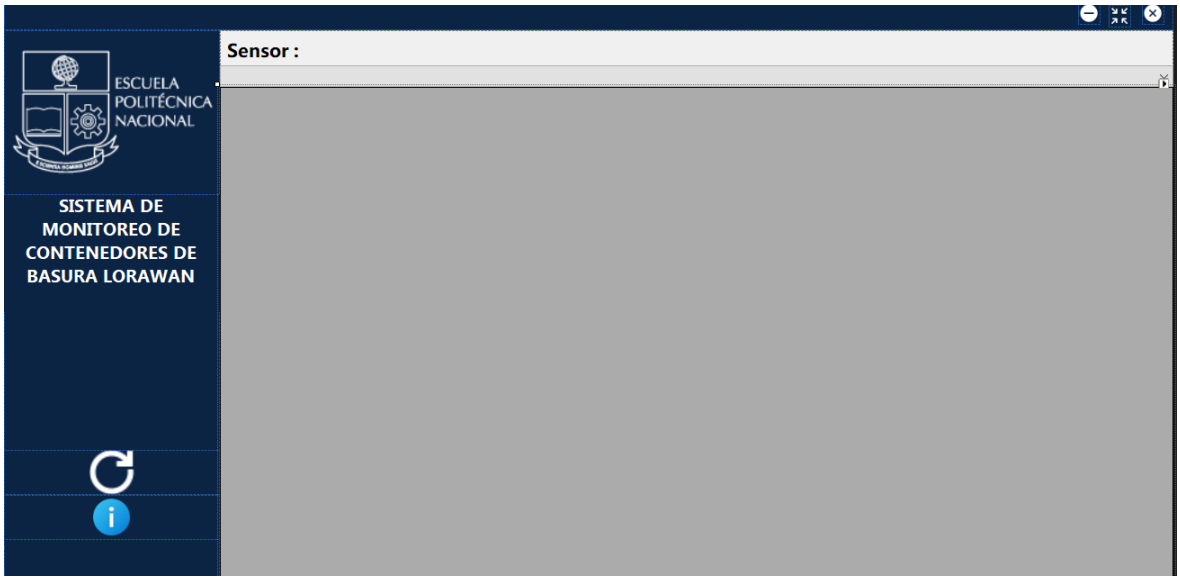


Figura 3.18 Conformación final de los elementos en la ventana principal de la interfaz gráfica.

Una implementación en esta interfaz tiene que ver con la de los mapas, los cuales muestran la ubicación de cada contenedor. Uno de los problemas encontrados en este apartado fue que al momento de incrementar o reducir el zoom sobre uno de estos mapas el marcador original se desplazaba hacia una ubicación totalmente diferente, se examinó el código y se encontró que una de las coordenadas se encontraba declarada incorrectamente, razón por la cual ocurría este error. Al corregir este dato el problema desapareció, permitiendo acercarse o alejarse del marcador sin algún inconveniente subsecuente.

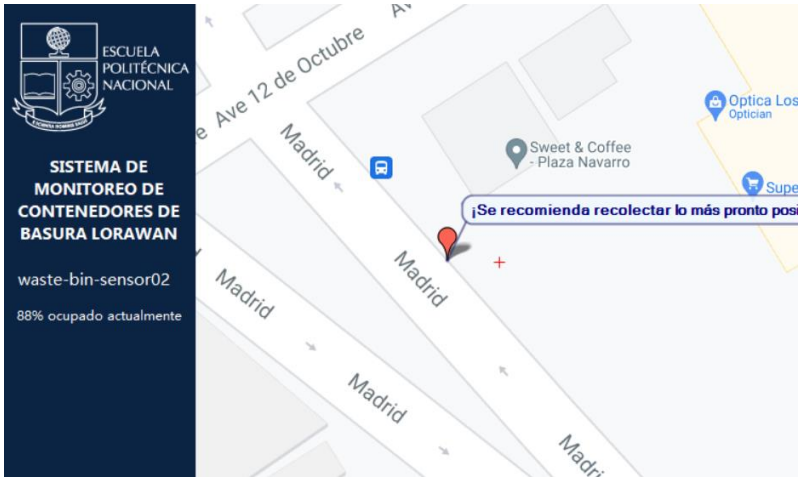


Figura 3.19 Zoom al máximo en uno de los mapas de contenedor.

Visualmente el contenedor del mapa no aumentaba de tamaño cuando se maximizaba la ventana contenedora, esto fue resuelto al cambiar una propiedad de este control visual, el cual se muestra a continuación.

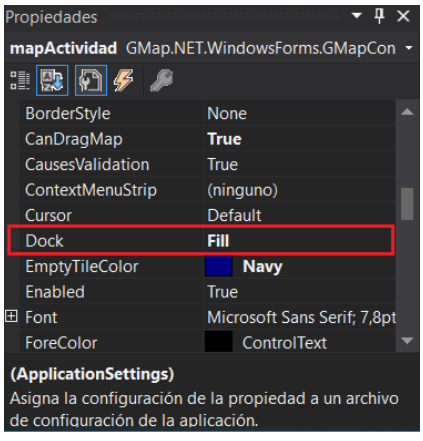


Figura 3.20 Propiedad alterada para ajustar el control visual al formulario.

3.2.2 PRUEBAS DE VERIFICACIÓN

Las pruebas hasta aquí realizadas fueron ejecutadas fuera del ambiente controlado de pruebas, es decir, fuera de los contenedores tipo con dimensiones similares a los utilizados por EMASEO EP. Para esta etapa los dispositivos se configuraron con nuevos parámetros, adaptados lo más cercanamente posible a las condiciones ambientales del medio y con otros elegidos arbitrariamente, ya que aquí la aplicación de escritorio fue puesta a prueba por un número determinado de usuarios.

```
Enter SetBasicInformat
AlarmHeight: 50 cm
Enter SetBasicInformat
AlarmTemp: 60
Enter SetBasicInformat
AlarmAngle: 45
Enter SetBasicInformat
AlarmBatt: 25
Enter SetBasicInformat
TestTimes: 10 min
Enter SetBasicInformat
WorkMode: 2
```

Figura 3.21 Parámetros de configuración nuevos para los dispositivos.

Entrando en detalles sobre la figura anterior, se estableció una alarma de altura de 50cm, es decir, que si la medición arrojada por los sensores es menor a esta cantidad se activara la alarma de capacidad (full), se configuró una alarma de temperatura de 60 grados centígrados, alarma de inclinación de 45 grados, alarma de batería de 25% restante o menos, periodos de medición y reporte cada 10 minutos y modo de trabajo 2 (modo demo).

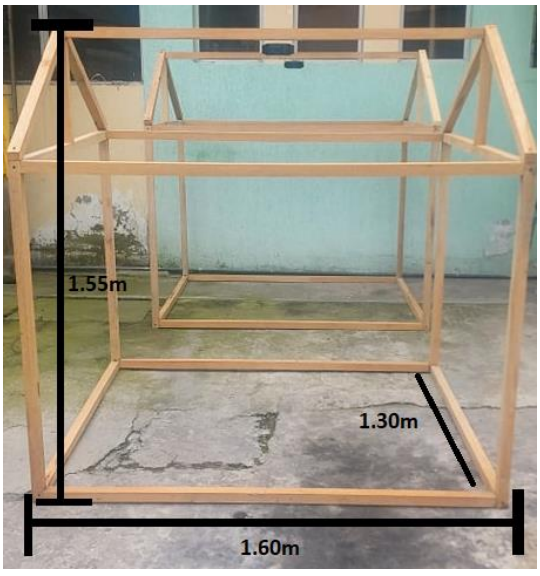


Figura 3.22 Parámetros de configuración nuevos para los dispositivos.

Figura 3.23 Nodo incorporado a uno de los contenedores tipo.

Ya que los dispositivos finales se acoplaron al entorno de pruebas fue necesario modificar unas líneas de código del programa, específicamente las referentes al nivel de ocupación en porcentaje que se muestra tanto en consola como en el DataGridView de la interfaz gráfica, ya que en etapas previas estaba configurado con otra referencia de altura, y como lo muestra la figura 3.22 esta debe ser ahora de 1.55m.

```
Console.WriteLine("Capacidad utilizada: " + ((int)Math.Ceiling((100 -
    (Convert.ToInt32(decAux.Level.Replace("mm", " ")) * (0.065359477)))) + "%");
```

Figura 3.24 Código en el programa de consola modificado para ajustarse a la altura de los contenedores.

```
Ocupado = ((int)Math.Ceiling((100 - Convert.ToInt32(iterA.Level.Replace("mm", " ")) * (0.065359477)))) + "%",
```

Figura 3.25 Código en la interfaz gráfica modificado para ajustarse a la altura de los contenedores.

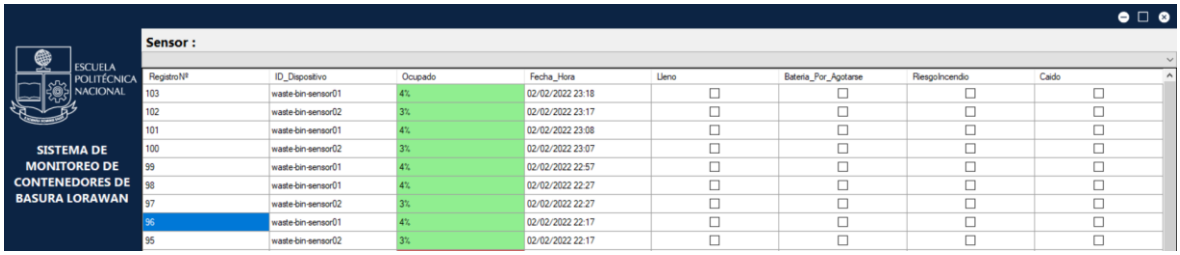


Figura 3.26 Resultados obtenidos con el nuevo código y nuevas mediciones.

Una vez que los nodos fueron introducidos en el ambiente controlado de pruebas y los códigos de los programas adaptados a este nuevo escenario se puso el sistema a prueba de los usuarios, quienes al terminar la comprobación llenaron una encuesta de satisfacción y opinión respecto al mismo. Ya que la aplicación es un prototipo la cantidad de encuestas aplicadas fue menor en comparación a las realizadas durante la etapa de definición de requisitos, además que la persona no debía cumplir el requisito de ser programador o alguien relacionado con LoRaWAN, ya que en caso de que el programa sea aplicado a mayor escala podría ser utilizado por cualquier usuario.

El resultado de estas encuestas se puede encontrar en la sección de Anexos.

3.2.3 CORRECCIÓN DE ERRORES

Aunque ciertos errores fueron corregidos en la etapa de Pruebas Iniciales otros aparecieron en la sección de Pruebas de Verificación, la intervención de tres usuarios en la segunda mencionada permitió visibilizar dichos inconvenientes, además expresaron su opinión respecto al prototipo mediante encuestas. En esta sección se recogió estos fallos y se los corrigió de la manera más oportuna posible; es decir, sin alterar la estructura del programa y evitando provocar nuevos errores.

En cuanto a la aplicación de consola no se reportó ningún error, ya que esta parte del programa está pensado para funcionar sin la necesidad de que el usuario intervenga o efectúe alguna operación sobre este. De todas formas fue mostrado a los usuarios para darles a entender su rol en el sistema. Por otro lado la interfaz gráfica mostró un par de problemas que se explica a continuación, así como su solución.

Para acceder a los mapas que contienen la ubicación de cada contenedor se debía hacer click sobre la fila del dispositivo deseado, sin embargo el seleccionar el cuadro enmarcado de la Figura 3.27 provocaba que se abran varias ventanas con la ubicación de los contenedores.

	RegistroNº	ID_Dispositivo	Ocupado	Fecha_Hora
▶	103	waste-bin-sensor01	4%	02/02/2022 23:18
	102	waste-bin-sensor02	3%	02/02/2022 23:17
	101	waste-bin-sensor01	4%	02/02/2022 23:08
	100	waste-bin-sensor02	3%	02/02/2022 23:07
	99	waste-bin-sensor01	4%	02/02/2022 22:57
	98	waste-bin-sensor01	4%	02/02/2022 22:27

Figura 3.27 Cabecera de fila en DataGridView.

Figura 3.28 Error provocado al hacer click sobre la primera cabecera de fila.

Para corregir este problema se utilizó otro evento del DataGridView, además se deshabilitó la opción de mostrar estas cabeceras de fila mediante una propiedad del mismo control visual.

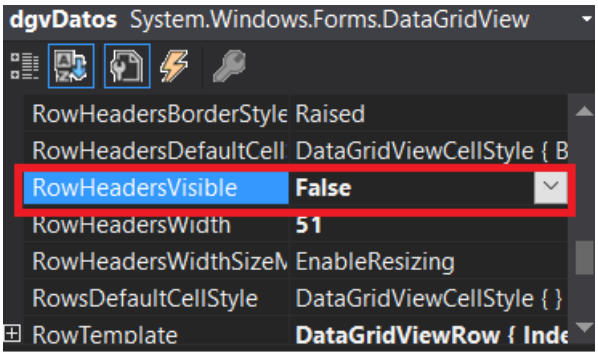


Figura 3.29 Desactivación de propiedad del DGV.

Este nuevo evento utilizado se denomina CellContentClick, y a diferencia del anteriormente utilizado no actúa cuando se selecciona toda una fila de datos, sino cuando se hace un click sobre una celda determinada de la tabla. Esto alteró ligeramente la forma en la que se puede acceder a la ventana con los mapas que contienen la ubicación de cada contenedor, ya que a partir de esta corrección se debía hacer click sobre el ID del dispositivo en el DataGridView.

```
1 referencia
private void dgvDatos_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    string nodo = dgvDatos.CurrentRow.Cells[1].Value.ToString();

    if (nodo == "waste-bin-sensor01")
    {
        double newLat = Convert.ToDouble(-0.2197791);
        double longit = Convert.ToDouble(-78.5070088);

        FrmUbicacion frmGraf = new FrmUbicacion(newLat, longit, nodo);
        frmGraf.Show();
    }
    else if (nodo == "waste-bin-sensor02")
    {
        double newLat = Convert.ToDouble(-0.2067698);
        double longit = Convert.ToDouble(-78.4876112);

        FrmUbicacion frmGraf = new FrmUbicacion(newLat, longit, nodo);
        frmGraf.Show();
    }

}
```

Figura 3.30 Nuevo evento DGV utilizado y código.

La siguiente figura muestra el resultado de este cambio, donde ahora para acceder a la ubicación de un contenedor se hace click sobre el casillero con el ID del dispositivo deseado.

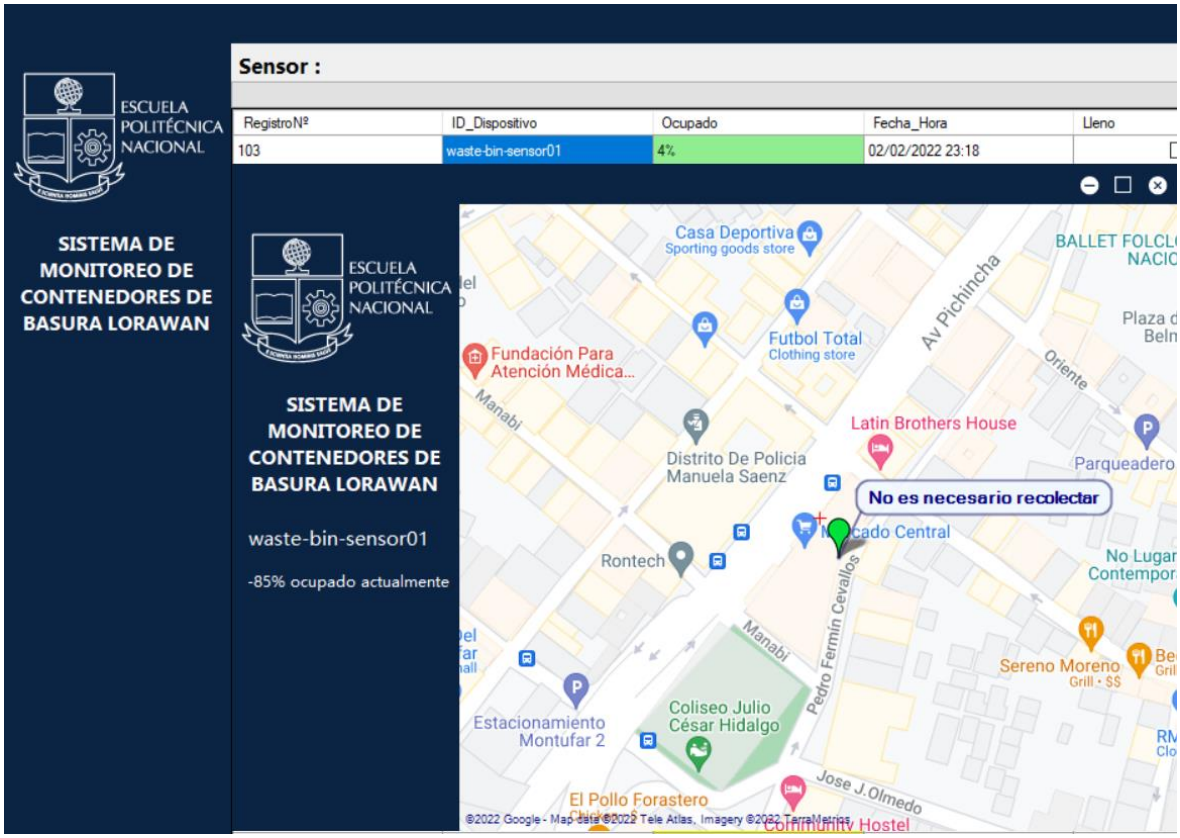


Figura 3.31 Cambio implementado en la interfaz gráfica.

Otro de los cambios realizados en base a la opinión de los usuarios tuvo que ver con el color de los casilleros correspondientes a la columna de ocupación, ya que los elegidos por defecto causaban conflicto en la visualización del dato obtenido, como lo indica la siguiente figura.

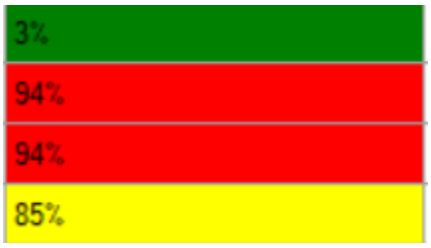


Figura 3.32 Colores usados antes.

Tomando en cuenta la opinión emitida por los usuarios que probaron la interfaz se implementó cambios en el código, para adoptar colores más visibles en el método correspondiente.

```
private void dgvDatos_CellFormatting(object sender, DataGridViewCellFormattingEventArgs e)
{

    if (e.ColumnIndex == 2 && e.Value != null)
    {

        string todo = Convert.ToString(e.Value).Replace("%"," ");

        int comp = Convert.ToInt32(todo);

        if (comp <= 75 )
        {
            e.CellStyle.BackColor = Color.LightGreen;
        }
        if (comp > 75 && comp <= 85)
        {
            e.CellStyle.BackColor = Color.Yellow;
        }
        if (comp > 85)
        {
            e.CellStyle.BackColor = Color.IndianRed;
        }
    }
}
```

Figura 3.33 Sección de código modificado.

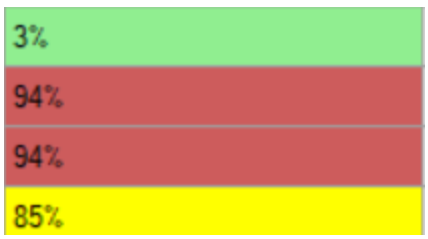


Figura 3.34 Nuevos colores implementados.

Otro error señalado tuvo que ver con el formulario o ventana correspondiente a los mapas, ya que en el panel izquierdo se informa sobre el último dato obtenido de ocupación, de todas maneras este error fue de forma ya que para solucionarlo únicamente se adaptó el código a las dimensiones del contenedor tipo fabricado.

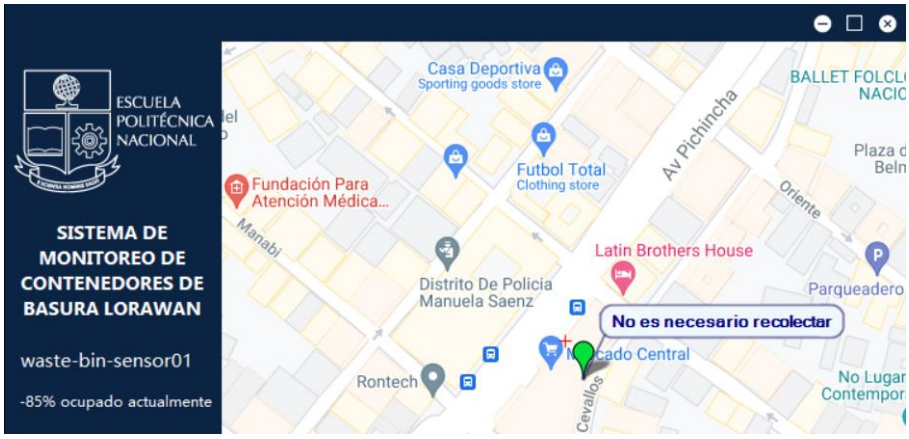


Figura 3.35 Error en la representación de datos.

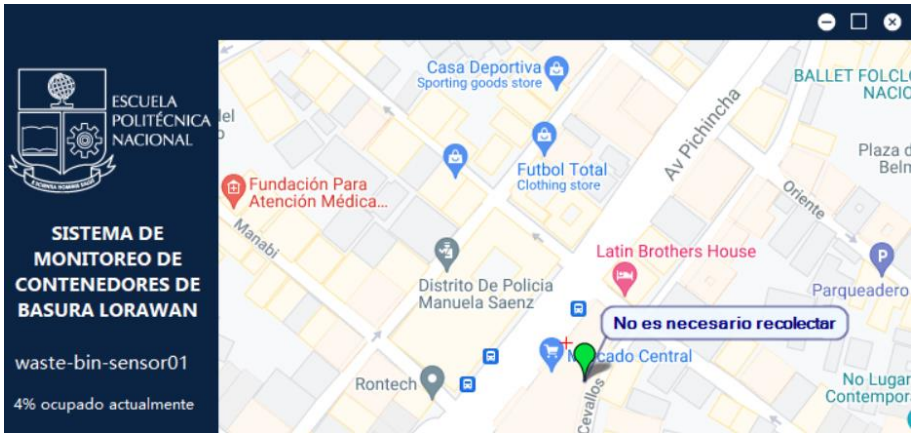


Figura 3.36 Error corregido.

En el entorno controlado de pruebas se ejecutó una prueba más, en donde uno de los contenedores tipo simuló estar lleno entre el 50 y 55%, mientras que el segundo simuló estar ocupado entre el 80 y 85%, en el primero se colocó el nodo denominado waste-bin-sensor01 mientras en el otro el waste-bin-sensor02. Como lo muestra la siguiente figura.

Figura 3.37 Contenedores tipo simulando estar a capacidad media y casi completa.

Con la red LoRaWAN en funcionamiento la plataforma de TTN captó los siguientes datos:

Figura 3.38 Datos mostrados en la consola de TTN.

En la línea correspondiente al waste-bin-sensor01, en el cuadro enmarcado rojo, se puede ver que la medición de altura es de 823mm, que tras hacer los cálculos respectivos utilizando la altura del contenedor (1500 mm aproximadamente) se obtiene una ocupación del 45.13%. Mientras que para el waste-bin-sensor02 se obtuvo una altura de 270mm, que tras realizar los cálculos se obtiene una ocupación del 82%.

Ejemplo de cálculo:

$$Ocupación\ waste-bin-sensor02 = 100 - \left(\frac{270\ mm}{1500\ mm} * 100\right)$$

$$Ocupación\ waste-bin-sensor02 = 82\%$$

Los sensores no reflejan directamente estos resultados, por lo que es necesario en el código de prototipo de aplicación ejecutar estas operaciones, como lo muestra la Figura 3.24. Caso similar con las banderas de mediciones como nivel de batería, riesgo de incendio, lleno o volteado, que no retornan un resultado del todo comprensible para el usuario, y que deben ser procesados en el código con tal fin. La siguiente figura muestra como la aplicación de consola muestra estas banderas, de forma tal que el usuario las comprenda sin demasiada explicación de fondo.

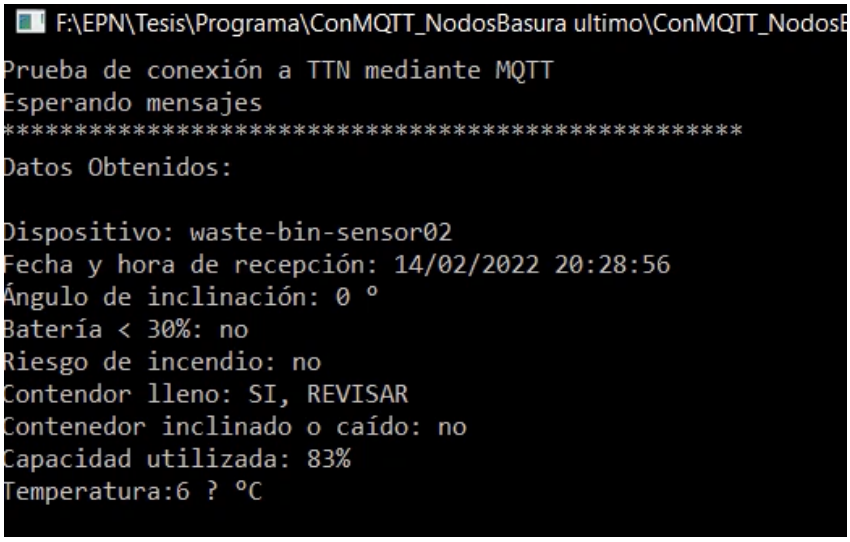


Figura 3.39 Datos escuchados desde el bróker MQTT mostrados en la aplicación de consola.

La otra parte del programa, es decir la interfaz gráfica, interpretó los datos mostrados en la Figura 3.38 de la siguiente forma:

Figura 3.40 Datos de los sensores interpretados por la interfaz gráfica.

Una muestra extendida de este ejemplo se puede encontrar en un video descargable en la sección de Anexos.

3.2.4 PRUEBAS DE RENDIMIENTO ENERGÉTICO

Los dispositivos finales utilizados en este proyecto incorporan una batería de cloruro de tionilo de litio por sensor, tienen una capacidad de 3.6V y son diseñadas específicamente para tareas en industrias o áreas comerciales, ofrecen un largo tiempo de duración para mediciones, seguridad o alarmas. El cloruro de litio-tionilo en su fórmula permite que tengan una descarga lenta y densidad de energía alta [83].

Para esta sección se puso a prueba los equipos en distintos periodos de tiempo durante 24 horas, estos periodos fueron de 10 minutos, 1 hora y 12 horas. Al iniciar y finalizar estos periodos se midió la capacidad de las baterías, y se obtuvo los resultados mostrados en la siguiente tabla.

Tabla 3.1 Resultados de eficiencia energética.

Periodo de tiempo	Voltaje Inicial	Voltaje Final	Voltaje estimado tras un mes de uso	Voltaje estimado tras un año de uso
10 minutos	3.68V	3.64V	2.48V	0V
1 hora	3.64V	3.63V	3.33V	0.03V
12 horas	3.63V	3.62V	3.59V	3.31V

Como se esperaba, a mas cantidad de mediciones menor tiempo útil de batería, pero tal y como lo especifica el fabricante en el datasheet del producto, se puede esperar más de 3 años de funcionalidad con periodos de medición de 4 horas. Estos periodos podrían variar dependiendo de la ubicación del contenedor y el ritmo de utilización de estos, pero en caso de que el prototipo sea aplicado a mayor escala se podrían establecer periodos de 4 o 5 horas.

3.2.5 VERIFICACIÓN DE TRAMAS ENTRE EXTREMOS DEL SISTEMA

En esta sección se comprobó que los datos de interés para el sistema se mantengan íntegros durante el periodo de comunicación, para esto se utilizó herramientas anteriores como el programa UartAssistant, la plataforma web de TTN y la aplicación de escritorio programada. Durante un periodo de envío de datos se comparó estos datos, obteniendo los resultados mostrados en la Figura 3.37. La primera parte de esta ilustración muestra los datos capturados desde los dispositivos directamente desde el conversor UART a USB incluido en el paquete de los dispositivos finales, se puede interpretar que la capacidad de la batería se mantiene en un 99%, registró una temperatura de 17.24ºC, una altura de 409mm y las banderas de incendio e inclinación en 0, pero la de capacidad utilizada en 1, o lleno. La segunda parte de la ilustración muestra una parte del payload capturado en la plataforma de TTN, con datos y banderas similares a los explicados recientemente, al final de esta figura se muestra los resultados de esta cadena de datos interpretados para el usuario, donde se puede verificar que los resultados son similares. Se ejecutó además otra serie de mediciones y comprobaciones para asegurar la fiabilidad del sistema, terminando en resultados similares a los obtenidos en este ejemplo.

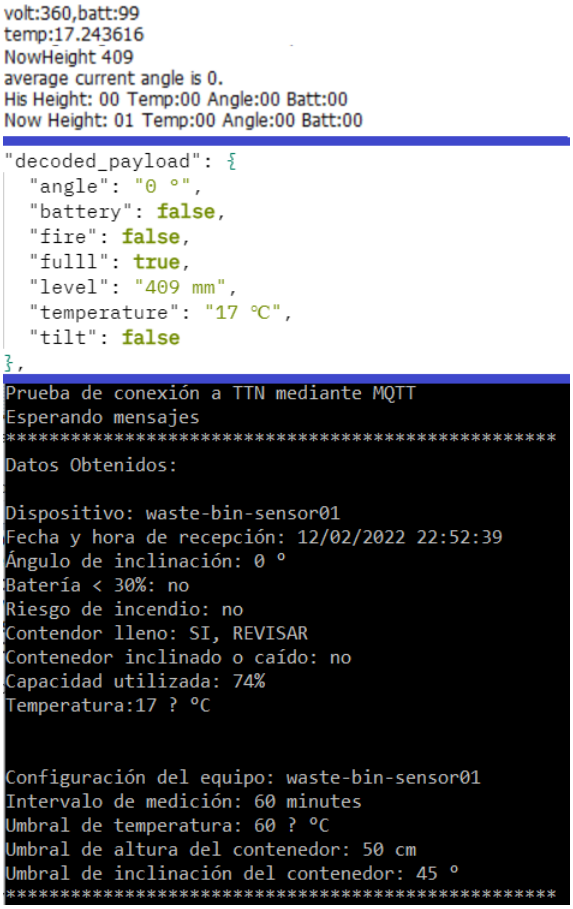


Figura 3.41 Resultados arrojados por los programas/plataformas.

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

El prototipo de aplicación que se obtuvo al finalizar este trabajo de titulación permite monitorear el nivel de ocupación de los contenedores de basura de gran capacidad a través de mediciones realizadas durante periodos de tiempo configurados. De esta manera uno o varios usuarios podrán visualizar el estado de estos depósitos y posteriormente ordenar su recolección, previniendo un posible colapso y problemas subsecuentes como contaminación ambiental, visual, entre otros.

El desarrollo de este trabajo de titulación implicó la búsqueda de información y adquisición de conocimiento respecto al Internet of Things, redes LPWAN, protocolo LoRaWAN, APIs para C# e interfaces gráficas. También sobre temas derivados como el protocolo MQTT, servidores de red y aplicación locales y en la nube.

El emplear LoRaWAN en el apartado de comunicaciones garantiza redes de gran alcance, además de dispositivos finales o nodos sensores con baterías de larga duración sin la necesidad de circuitos externos adicionales.

La metodología en cascada permitió desarrollar el prototipo de aplicación de forma progresiva y ordenada, la ausencia de uno o varios clientes finales permitió adoptar este procedimiento. Además, la aplicación no es multinivel, por lo que no fue necesario dividir las tareas en etapas muy distantes entre sí.

The Things Network es una plataforma de código abierto que permite utilizar su infraestructura para configurar servidores de red y aplicación, parte fundamental de la arquitectura LoRaWAN, para la naturaleza de este proyecto resultó conveniente, ya que sus clústeres se encuentran en el Internet, y los nodos sensores en la práctica podrían ubicarse en distintos puntos de la ciudad, haciendo despreciable la idea de implementar los servidores mencionados de manera local.

La implementación de una base de datos que almacene los datos generados por los dispositivos finales abre la posibilidad de incorporar herramientas en la aplicación que permitan planificar rutas de recolección más eficientes, basadas en el ritmo de acumulación de desperdicios dentro de los contenedores, época del año y sector donde se ubiquen. Por otra parte, incluir una característica que permite visualizar la ubicación de los contenedores mediante un mapa dentro del mismo programa posibilita diferenciarlos cuando se encuentren dentro de un mismo sector o cuadra, además de llevar un registro digital de su localización.

Las pruebas de ejecución no fueron ejecutadas por una cantidad elevada de usuarios, sin embargo fue suficiente para evidenciar errores y problemas en la aplicación. Estas pruebas además de verificar el funcionamiento del programa permitieron realizar correcciones en el diseño de la interfaz gráfica.

LoRaWAN permite plantear soluciones para problemas dentro de la ciudad o en zonas agrícolas, y debería ser considerado para el desarrollo de otros prototipos por su apertura de información y diversificación. Las integraciones que ofrecen sus servidores con aplicaciones externas facilitan la programación de aplicaciones en distintos lenguajes y plataformas; ya que, por ejemplo, el presente trabajo de titulación pudo plantearse como un prototipo de programa web.

4.2 RECOMENDACIONES

Bajo el contexto de la pandemia por el COVID-19 se recomienda plantear futuros prototipos como aplicaciones web o multiplataforma, ya que ciertos usuarios probablemente se vean obligados a monitorear el estado de los contenedores en locaciones fuera de sus oficinas o estaciones de trabajo cotidianas.

En caso de que este prototipo se utilice a gran escala se recomienda configurar los periodos de medición de los sensores por separado, ya que dependiendo de la ubicación de los contenedores estos podrían llenarse a un ritmo mayor o menor, y establecerlos por igual podría implicar un uso inadecuado de sus baterías, lo cual conllevaría a un reemplazo prematuro.

Se recomienda para futuros prototipos similares analizar los datos que se deberían almacenar en la base de datos, ya que el ritmo de generación de información podría terminar en un uso innecesario de espacio de disco del servidor donde esté funcionando el programa.

Se recomienda poner en prueba el prototipo en un ambiente real, ya que el presente trabajo de titulación utilizó uno controlado, sin la presencia de edificaciones o posibles obstrucciones que alteren la comunicación.

5. BIBLIOGRAFÍA

[1] RedHat, “¿Qué es el Internet de las cosas?” [Online]. Available: https://www.redhat.com/es/topics/internet-of-things/what-is-iot. [Accessed: 12-Nov-2021].

[2] R. Hernandez, “¿Qué es la tecnología LoRa y por qué es importante para IoT? - Post - The Things Network.” [Online]. Available: https://www.thethingsnetwork.org/community/santa-rosa/post/que-es-la-tecnologia-lora-y-por-que-es-importante-para-iot. [Accessed: 12-Nov-2021].

[3] Semtech, “Smart Environment | Applications | Semtech LoRa Technology | Semtech.” [Online]. Available: http://www.semtech.com/lora/lora-applications/smart-environment. [Accessed: 12-Nov-2021].

[4] J. R. Alvarado-López, R. F. Correa-Quezada, M. del C. Tituaña-Castillo, J. R. Alvarado-López, R. F. Correa-Quezada, and M. del C. Tituaña-Castillo, “Migración interna y urbanización sin eficiencia en países en desarrollo: evidencia para Ecuador,” *Papeles de población*, vol. 23, no. 94, pp. 99–123, 2017.

[5] “La ruta de las 2.200 toneladas diarias de basura producidas en Quito.” [Online]. Available: https://www.primicias.ec/noticias/sociedad/conozca-ruta-basura-quito/. [Accessed: 11-Nov-2021].

[6] el Universo, “Emaseo EP de Quito reporta altos costos por daños de los contenedores de basura | Ecuador | Noticias | El Universo.” [Online]. Available: https://www.eluniverso.com/noticias/2020/09/24/nota/7989343/emaseo-ep-quito-reporta-altos-costos-danos-contenedores-basura/. [Accessed: 12-Nov-2021].

[7] EMASEO EP, “Sistema de Recolección Mecanizada Quito - EMASEO EP.” [Online]. Available: http://www.emaseo.gob.ec/servicios/recoleccion-mecanizada/. [Accessed: 13-Nov-2021].

[8] el Telégrafo, “El Telégrafo - La contenerización de basura registra problemas en Quito.” [Online]. Available: https://www.eltelegrafo.com.ec/noticias/quito/1/la-contenerizacion-de-basura-registra-problemas-en-quito. [Accessed: 13-Nov-2021].

[9] EMASEO EP, “Cifras - Emaseo EP.” [Online]. Available: http://www.emaseo.gob.ec/servicios-de-gente-para-gente/cifras/. [Accessed: 13-Nov-2021].

[10] Plan V, “Quito generó 600 toneladas diarias más de desechos durante la pandemia | Plan V.” [Online]. Available: https://www.planv.com.ec/historias/sociedad/quito-genero-600-toneladas-diarias-mas-desechos-durante-la-pandemia. [Accessed: 13-Nov-2021].

[11] Dinkgtek, “DF703 Waste Bin Detector LoRaWAN Data sheet.”

[12] Anovo, “Qué es la IP68 y cómo proteger los dispositivos de la humedad - ANOVO.” [Online]. Available: https://www.anovo.es/que-es-la-ip68-y-como-proteger-los-dispositivos-de-la-humedad/. [Accessed: 15-Nov-2021].

[13] Laird, “Sentrius RG1xx LoRaWAN Gateway + Wi-Fi / Ethernet + Optional LTE (US Only) | Laird Connectivity.” [Online]. Available:

https://www.lairdconnect.com/wireless-modules/lorawan-solutions/sentrius-rg1xx-lorawan-gateway-wi-fi-ethernet-optional-lte-us-only. [Accessed: 15-Nov-2021].

[14] The Things Network, “The Things Network Stack V3 Update.” [Online]. Available: https://www.thethingsnetwork.org/article/the-things-network-stack-v3-update. [Accessed: 15-Nov-2021].

[15] The Things Network, “The Things Network V2 is Permanently Shutting Down (scheduled) - TTN Announcements - The Things Network.” [Online]. Available: https://www.thethingsnetwork.org/forum/t/the-things-network-v2-is-permanently-shutting-down-scheduled/50710. [Accessed: 15-Nov-2021].

[16] Deloitte España, “¿Qué es IoT (Internet Of Things)? .” [Online]. Available: https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html. [Accessed: 13-Feb-2022].

[17] El Español, “Todos los sensores que hay en un teléfono Android y para qué sirven.” [Online]. Available: https://www.elespanol.com/elandroidelibre/20200112/sensores-telefono-android-sirven/459204673_0.html. [Accessed: 17-Nov-2021].

[18] L. K. Ramasamy and S. Kadry, “Internet of things (IoT),” in *Blockchain in the Industrial Internet of Things*, IOP Publishing, 2021.

[19] Fracttal, “The 9 most important applications of the Internet of Things (IoT).” [Online]. Available: https://www.fracttal.com/en/blog/the-9-most-important-applications-of-the-internet-of-things. [Accessed: 17-Nov-2021].

[20] Intel Latinoamérica, “IoT Aplicado a la Fabricación: Fabricación Inteligente Intel.” [Online]. Available: https://www.intel.la/content/www/xl/es/manufacturing/manufacturing-industrial-overview.html. [Accessed: 17-Nov-2021].

[21] Cerem, “IoT Smart city: internet de las cosas y ciudades inteligentes.” [Online]. Available: https://www.cerem.ec/blog/iot-en-la-smart-city. [Accessed: 18-Nov-2021].

[22] IoT Analytics, “The top 10 Smart City use cases that are being prioritized now.” [Online]. Available: https://iot-analytics.com/top-10-smart-city-use-cases-prioritized-now/. [Accessed: 18-Nov-2021].

[23] Behrtech, “6 Leading Types of IoT Wireless Technologies | BehrTech Blog.” [Online]. Available: https://behrtech.com/blog/6-leading-types-of-iot-wireless-tech-and-their-best-use-cases/. [Accessed: 18-Nov-2021].

[24] Industrial M2M, “LPWAN: qué son y para qué se utilizan | M2M - Logitek.” [Online]. Available: https://www.m2mlogitek.com/lpwan-que-son-y-para-que-se-utilizan/. [Accessed: 18-Nov-2021].

[25] PandoraFms, “Qué es LPWAN: entrada al protocolo de comunicaciones de IoT.” [Online]. Available: https://pandorafms.com/blog/es/que-es-lpwan/. [Accessed: 18-Nov-2021].

[26] LoRa Alliance, “Member Directory - LoRa Alliance®.” [Online]. Available: https://lora-alliance.org/member-directory/?_sft_member_level=sponsor. [Accessed: 18-Nov-2021].

[27] B. Martinez, F. Adelantado, A. Bartoli, and X. Vilajosana, “Exploring the performance boundaries of NB-IoT,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5702–5712, Jun. 2019.

[28] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, “A comparative study of LPWAN technologies for large-scale IoT deployment,” *ICT Express*, vol. 5, no. 1, pp. 1–7, Mar. 2019.

[29] Semtech, “A Brief History of LoRa: Three Inventors Share Their Story.” [Online]. Available: https://blog.semtech.com/a-brief-history-of-lora-three-inventors-share-their-personal-story-at-the-things-conference. [Accessed: 19-Nov-2021].

[30] Manrique Miguel, Buitrago Marcela, and Hernández-Gutiérrez Jairo, “Redes LoRaWAN. Revisión de componentes funcionales en aplicaciones IoT.,” *Universidad Distrital Francisco José de Caldas*, 2019.

[31] LoRa Alliance, “What is LoRaWAN® Specification - LoRa Alliance®.” [Online]. Available: https://lora-alliance.org/about-lorawan/. [Accessed: 20-Nov-2021].

[32] The Things Stack, “Join Server | The Things Stack for LoRaWAN.” [Online]. Available: https://www.thethingsindustries.com/docs/reference/components/join-server/. [Accessed: 20-Nov-2021].

[33] Ghosliya Sakshama, “All About LoRa and LoRaWAN: LoRa: Symbol Generation.” [Online]. Available: http://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html. [Accessed: 19-Nov-2021].

[34] The Things Network, “Spreading Factors | The Things Network.” [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/spreading-factors/. [Accessed: 19-Nov-2021].

[35] Radio Bridge, “Understanding LoRaWAN End Devices - Radio Bridge.” [Online]. Available: https://radiobridge.com/blog/understanding-lorawan-end-devices. [Accessed: 20-Nov-2021].

[36] The Things Netwokr, “Device Classes .” [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/classes/. [Accessed: 21-Nov-2021].

[37] The Things Network, “End Device Activation .” [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/. [Accessed: 21-Nov-2021].

[38] Tech Play On, “LoRa - Device Activation Call Flow (Join Procedure) using OTAA and ABP.” [Online]. Available: https://www.techplayon.com/lora-device-activation-call-flow-join-procedure-using-otaa-and-abp/. [Accessed: 21-Nov-2021].

[39] The Things Network, “Gateways .” [Online]. Available: https://www.thethingsnetwork.org/docs/gateways/. [Accessed: 21-Nov-2021].

[40] I. O. Monfort, “Estudio de la arquitectura y el nivel de desarrollo de la red LoRaWAN y de los dispositivos LoRa.”

[41] The Things Network, “Limitations .” [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/limitations/. [Accessed: 21-Nov-2021].

[42] RF Wireless World, “LoRa | Advantages of LoRaWAN | Disadvantages of LoRaWAN.” [Online]. Available: https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Lora-or-LoRaWAN.html. [Accessed: 21-Nov-2021].

[43] The Things Stack, “The Things Network.” [Online]. Available: https://www.thethingsindustries.com/docs/getting-started/ttn/. [Accessed: 22-Nov-2021].

[44] The Things Network, “The Things Network Console .” [Online]. Available: https://www.thethingsnetwork.org/docs/network/console/. [Accessed: 22-Nov-2021].

[45] The Things Stack, “Components.” [Online]. Available: https://www.thethingsindustries.com/docs/reference/components/. [Accessed: 22-Nov-2021].

[46] The Things Stack, “Application Server .” [Online]. Available: https://www.thethingsindustries.com/docs/reference/components/application-server/. [Accessed: 22-Nov-2021].

[47] The Things Stack for LoRaWAN, “Gateway Server .” [Online]. Available: https://www.thethingsindustries.com/docs/reference/components/gateway-server/. [Accessed: 22-Nov-2021].

[48] The Things Stack for LoRaWAN, “Identity Server.” [Online]. Available: https://www.thethingsindustries.com/docs/reference/components/identity-server/. [Accessed: 22-Nov-2021].

[49] Llamas Luis, “¿Qué es MQTT? Su importancia como protocolo IoT.” [Online]. Available: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/. [Accessed: 22-Nov-2021].

[50] The Things Stack, “MQTT Server.” [Online]. Available: https://www.thethingsindustries.com/docs/integrations/mqtt/. [Accessed: 22-Nov-2021].

[51] IOT Factory, “Best LORAWAN Network Servers .” [Online]. Available: https://iotfactory.eu/products/software-platform/best-lorawan-network-servers/. [Accessed: 22-Nov-2021].

[52] Laird, “User Guide/Datasheet - RG1xx + RG191+LTE .” [Online]. Available: https://www.lairdconnect.com/documentation/user-guidedatasheet-rg1xx-rg191lte. [Accessed: 23-Nov-2021].

[53] Laird, “Sentrius RG1xx LoRaWAN Gateway + Wi-Fi / Ethernet + Optional LTE (US Only) .” [Online]. Available: https://www.lairdconnect.com/wireless-modules/lorawan-solutions/sentrius-rg1xx-lorawan-gateway-wi-fi-ethernet-optional-lte-us-only. [Accessed: 23-Nov-2021].

[54] “LoRa ® Technology Gateway User’s Guide,” 2016.

[55] Microchip Technology, “LORA(R) TECHNOLOGY EVALUATION KIT - 800 .” [Online]. Available: https://www.microchip.com/en-us/development-tool/dv164140-1#. [Accessed: 24-Nov-2021].

[56] Microchip Technology, “RN2903 .” [Online]. Available: https://www.microchip.com/en-us/product/RN2903#buy-from-store. [Accessed: 24-Nov-2021].

[57] NetOP Technology, “WASTE BIN SENSOR.” [Online]. Available: https://iot-shops.com/wp-content/uploads/2020/04/SN-LR-Waste_Bin_Sensor_V2.1.pdf. [Accessed: 24-Nov-2021].

[58] IoT-Shops, “WASTE BIN SENSOR-LR .” [Online]. Available: https://iot-shops.com/product/waste-bin-sensor-lr/?wmc-currency=USD. [Accessed: 24-Nov-2021].

[59] Besoftware, “¿Qué es C# en programación y para que sirve?” [Online]. Available: https://bsw.es/que-es-c/. [Accessed: 26-Nov-2021].

[60] GeeksforGeeks, “Introduction to Visual Studio.” [Online]. Available: https://www.geeksforgeeks.org/introduction-to-visual-studio/. [Accessed: 26-Nov-2021].

[61] Computerweekly, “¿Qué es SQL (Structured Query Language o Lenguaje de consultas estructuradas)? .” [Online]. Available: https://www.computerweekly.com/es/definicion/SQL-Structured-Query-Language-o-Lenguaje-de-consultas-estructuradas. [Accessed: 26-Nov-2021].

[62] Datademia, “¿Qué es SQL? .” [Online]. Available: https://datademia.es/blog/que-es-sql. [Accessed: 26-Nov-2021].

[63] Intelequia, “¿Qué es Microsoft SQL Server y para qué sirve?” [Online]. Available: https://intelequia.com/blog/post/2948/qu%C3%A9-es-microsoft-sql-server-y-para-qu%C3%A9-sirve. [Accessed: 26-Nov-2021].

[64] Xataka, “API: qué es y para qué sirve.” [Online]. Available: https://www.xataka.com/basics/api-que-sirve. [Accessed: 27-Nov-2021].

[65] Microsoft Docs, “Object-Oriented Programming (C#) .” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/oop. [Accessed: 30-Nov-2021].

[66] Ok Hosting, “Metodologías del Desarrollo de Software.” [Online]. Available: https://okhosting.com/blog/metodologias-del-desarrollo-de-software/. [Accessed: 01-Dec-2021].

[67] Lucidchart, “¿Qué es un diagrama entidad-relación? .” [Online]. Available: https://www.lucidchart.com/pages/es/que-es-un-diagrama-entidad-relacion. [Accessed: 02-Dec-2021].

[68] GitHub, “eclipse/paho.mqtt.m2mqtt.” [Online]. Available: https://github.com/eclipse/paho.mqtt.m2mqtt. [Accessed: 27-Nov-2021].

[69] CodeProject, “GMap.NET - Great Maps for Windows Forms and Presentation .” [Online]. Available: https://www.codeproject.com/Articles/32643/GMap-NET-Great-Maps-for-Windows-Forms-and-Presenta. [Accessed: 27-Nov-2021].

[70] Alfonzo Guzmán Milton Iván, “Análisis Técnico y Regulatorio de Nuevas Redes de Transmisión para Dispositivos IoT (Internet of Things) en la ciudad de Guayaquil, Ecuador,” Guayaquil, 2020.

[71] The Things Stack for LoRaWAN, “LoRa Basics™ Station .” [Online]. Available: https://www.thethingsindustries.com/docs/gateways/lora-basics-station/. [Accessed: 11-Dec-2021].

[72] The Things Stack for LoRaWAN, “LoRaWAN Network Server (LNS) .” [Online]. Available: https://www.thethingsindustries.com/docs/gateways/lora-basics-station/lns/. [Accessed: 11-Dec-2021].

[73] The Things Stack for LoRaWAN, “Configuration and Update Server (CUPS) .” [Online]. Available: https://www.thethingsindustries.com/docs/gateways/lora-basics-station/cups/. [Accessed: 11-Dec-2021].

[74] Let’s Encrypt, “Acerca de Let’s Encrypt.” [Online]. Available: https://letsencrypt.org/es/about/. [Accessed: 11-Dec-2021].

[75] Viafirma, “¿Qué es una autoridad de certificación? .” [Online]. Available: https://www.viafirma.com/faq/es/que-es-una-autoridad-de-certificacion/. [Accessed: 11-Dec-2021].

[76] The Things Stack for LoRaWAN, “Connect Laird Sentrius™ RG1xx with LoRa Basics™ Station .” [Online]. Available: https://www.thethingsindustries.com/docs/gateways/laird-sentrius/lbs/. [Accessed: 10-Dec-2021].

[77] The Things Network, “Can’t get Laird RG1xx Basics Station Forwarder working with TTN - Gateways .” [Online]. Available: https://www.thethingsnetwork.org/forum/t/cant-get-laird-rg1xx-basics-station-forwarder-working-with-ttn/42901/15. [Accessed: 11-Dec-2021].

[78] The Things Network, “Connect your Laird RG1xx gateway to The Things Stack using LoRa Basics Station - Erik Lins (Laird).” [Online]. Available: https://www.youtube.com/watch?v=Q6r6uxqXFCY&t=378s. [Accessed: 11-Dec-2021].

[79] STM, “Introduction LoRaWAN ® AT commands for STM32CubeWL.”

[80] Dingtek, “DF703Waste Bin Detector Configuration Manual,” 2021.

[81] Dingtek, “DF702&DF703 Waste Bin Sensor Protocol_LoRaWAN,” 2021.

[82] “Pila AA de litio - cloruro de tionilo, 3.7V, 2.6Ah | RS Components.” [Online]. Available: https://es.rs-online.com/web/p/pilas-aa/0596602. [Accessed: 10-Feb-2022].

[83] Homey, “What is Zigbee? Explaining the World’s Most Popular Smart Light Network Technology | Homey.” [Online]. Available: https://homey.app/en-au/wiki/what-is-zigbee/. [Accessed: 18-Nov-2021].

[84] ABR, “What is RFID and How Does RFID Work? - AB&R®.” [Online]. Available: https://www.abr.com/what-is-rfid-how-does-rfid-work/. [Accessed: 18-Nov-2021].

[85] PC Mag, “Definition of ISM band | PCMag.” [Online]. Available: https://www.pcmag.com/encyclopedia/term/ism-band. [Accessed: 18-Nov-2021].

[86] LoRa Developers, “An In-depth look at LoRaWAN® Class A Devices | DEVELOPER PORTAL.” [Online]. Available: https://lora-

developers.semtech.com/documentation/tech-papers-and-guides/lorawan-class-a-devices/. [Accessed: 18-Nov-2021].

[87] Speed Check, “What is the payload?” [Online]. Available: https://www.speedcheck.org/wiki/payload/. [Accessed: 18-Nov-2021].

[88] Tutorials Point, “Reference Models in Computer Network.” [Online]. Available: https://www.tutorialspoint.com/Reference-Models-in-Computer-Network. [Accessed: 19-Nov-2021].

[89] “Introducción al conjunto de protocolos TCP/IP (Guía de administración del sistema: servicios IP).” [Online]. Available: https://docs.oracle.com/cd/E19957-01/820-2981/6nei0r0r9/index.html. [Accessed: 19-Nov-2021].

[90] Abierto al Público, “Código abierto: conceptos y aplicaciones .” [Online]. Available: https://blogs.iadb.org/conocimiento-abierto/es/codigo-abierto/. [Accessed: 27-Nov-2021].

[91] GDX Group, “Clustering de servidores en tu empresa .” [Online]. Available: https://gdx-group.com/clustering-de-servidores-en-tu-empresa/. [Accessed: 27-Nov-2021].

[92] Tutorial LTE, “MIMO - Multiple Input Multiple Output.” [Online]. Available: http://www.ipv6go.net/lte/mimo.php. [Accessed: 27-Nov-2021].

[93] The Indian Express, “IP certifications: What do they mean and how to decode them .” [Online]. Available: https://indianexpress.com/article/technology/techook/ip-certifications-what-do-they-mean-and-how-to-decode-them-7465620/. [Accessed: 27-Nov-2021].

ANEXOS

ANEXO A. Preguntas y resultados de encuestas para definir los requisitos de la aplicación.

ANEXO B. Diagramas de clase.

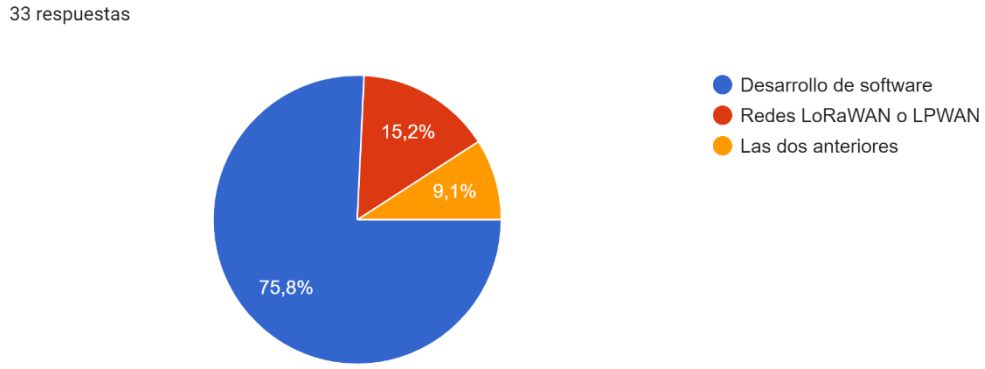
ANEXO C. Preguntas y resultados de encuestas ejecutadas en las pruebas de verificación del programa.

ANEXO D. Video que muestra el funcionamiento del sistema.

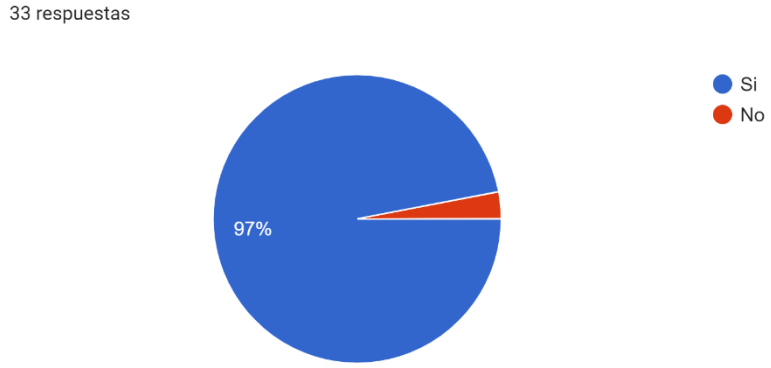
ANEXO A

Pregunta 1: Campo en el que se desenvuelve.

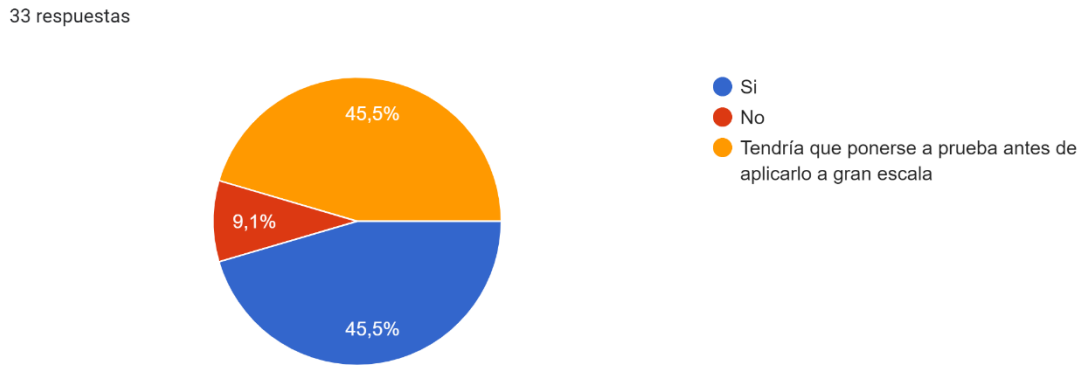
Resultados:



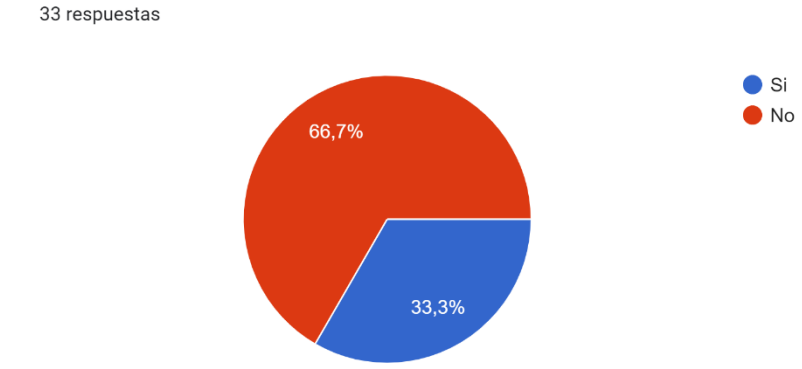
Pregunta 2: Dentro de su ciudad, o en alguna que usted visitó, ¿encontró que uno o más contenedores de basura de gran capacidad había desbordado su capacidad? de manera tal que contaminaba sus alrededores.



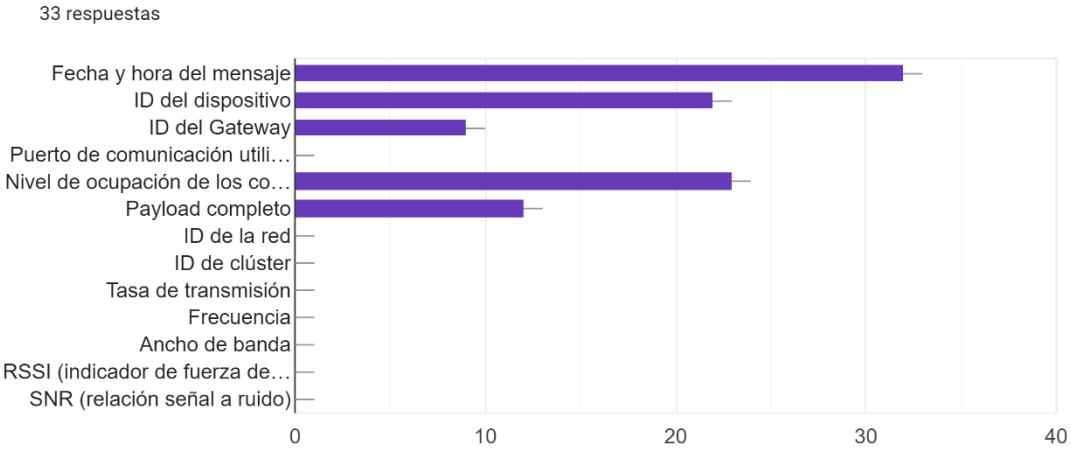
Pregunta 3: Con base en su experiencia y lo que ha podido evidenciar en las calles de su ciudad u otras, ¿cree que es oportuna la intervención de la tecnología para ayudar a mitigar el problema del desbordamiento de contenedores de basura?



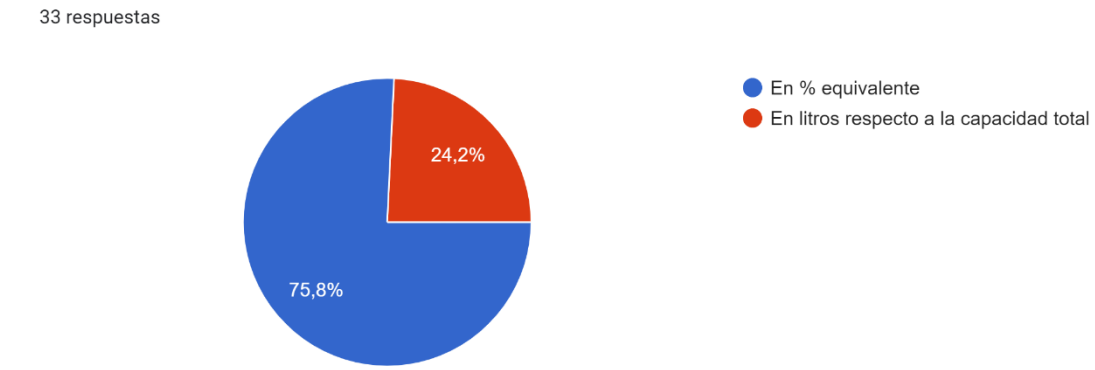
Pregunta 4: Considerando que la aplicación tiene un enfoque informativo (respecto al estado de los contenedores), ¿considera necesario mostrar información técnica (frecuencia, canales, calidad de la señal, etc.) mediante la interfaz gráfica?



Pregunta 5: A continuación encontrará una lista de datos los cuales son generados por los dispositivos de la red LoRaWAN, seleccione 3 que usted piense que se deberían mostrar en la interfaz de la aplicación.

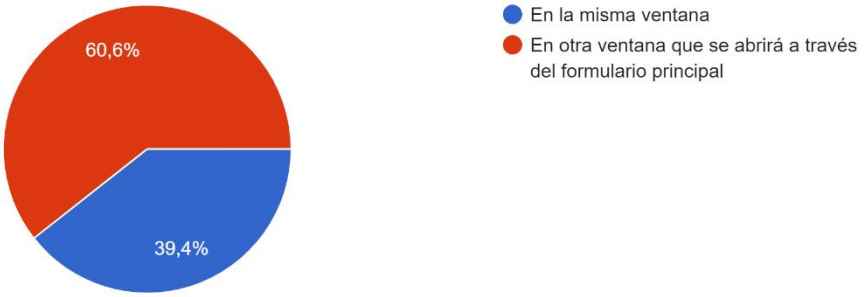


Pregunta 6: ¿Considera que en la interfaz de usuario sería mejor que se mostrara el nivel de ocupación de los contenedores de basura en porcentaje equivalente a la capacidad de estos o en litros?



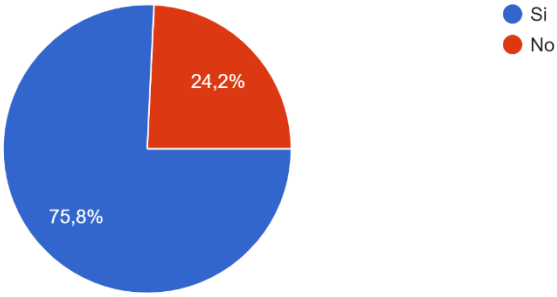
Pregunta 7: El programa está ideado para mostrar la ubicación de los contenedores a través de un mapa, ¿considera que esta característica debería ser implementada en la misma ventana de datos o a través de otra?

33 respuestas



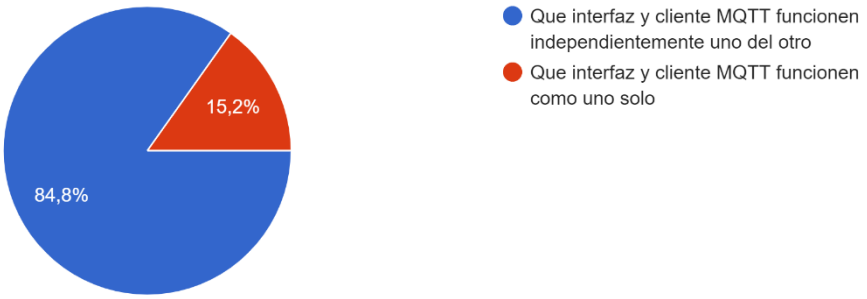
Pregunta 8: ¿Considera necesario implementar en la interfaz gráfica una característica que permita ver los datos de los sensores por separado o en conjunto?

33 respuestas

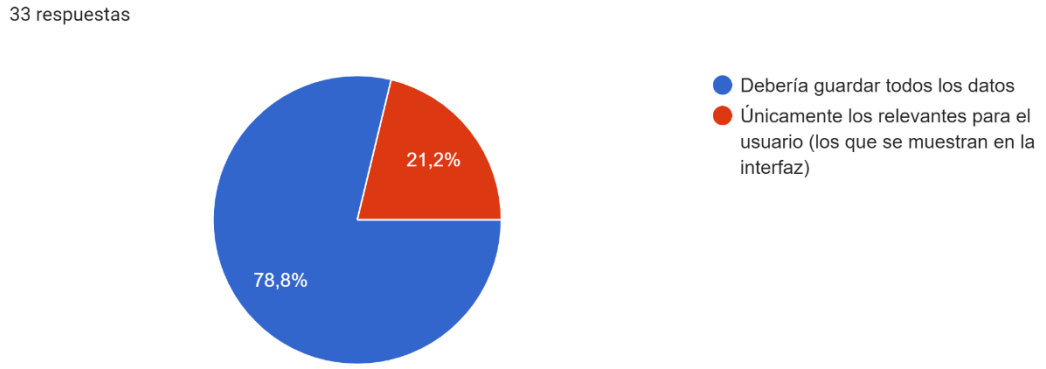


Pregunta 9: La aplicación consta de 3 componentes, sin embargo dos de ellos (interfaz y cliente MQTT) pueden ser implementados como uno solo. Considerando lo anterior, para usted sería mejor:

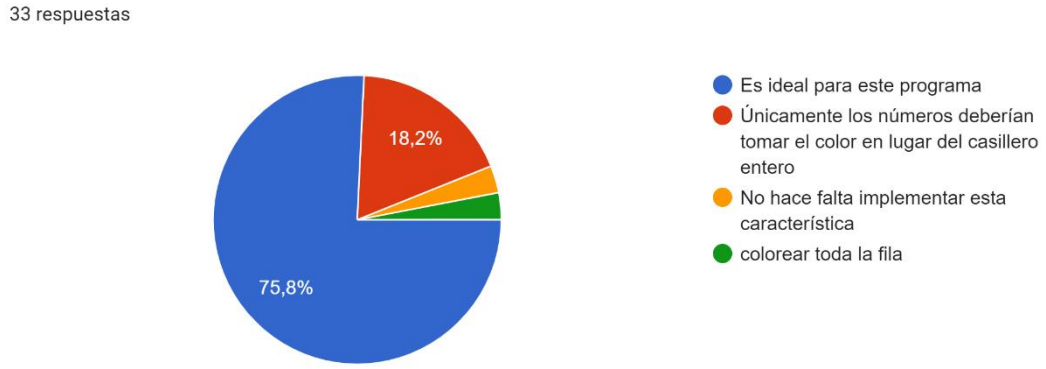
33 respuestas



Pregunta 10: ¿Considera que la base de datos a implementar en paralelo con el programa debería guardar todos los datos generados por los nodos? Independiente de si estos son mostrados o no en la interfaz de datos del usuario.



Pregunta 11: La imagen presentada es un bosquejo de cómo se mostrarían los datos en la interfaz de usuario, como se puede observar, para el nivel de ocupación, se planteó la utilización de colores en los casilleros correspondientes para mostrarlos de manera más gráfica. ¿Considera este una decisión de diseño ideal para este programa, o considera que sería mejor no utilizar este tipo de infografía?



ANEXO B

Diagrama de clases para guardar la información en la base de datos mediante procedimientos LINQ.

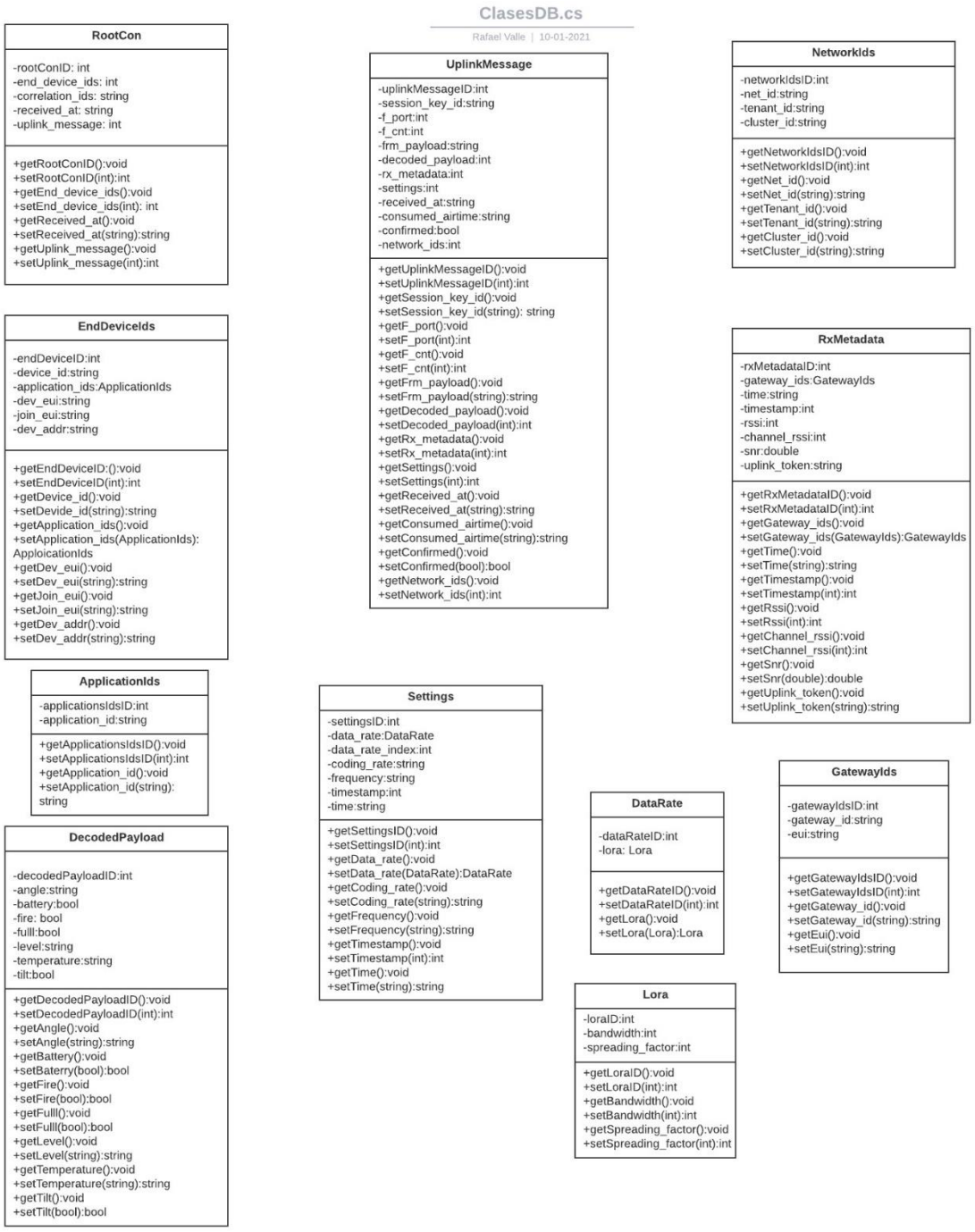


Diagrama de clase para procesar los mensajes MQTT que contienen información útil para el proyecto.

Diagrama de clase para procesar los mensajes MQTT que contienen información de configuración de los equipos.

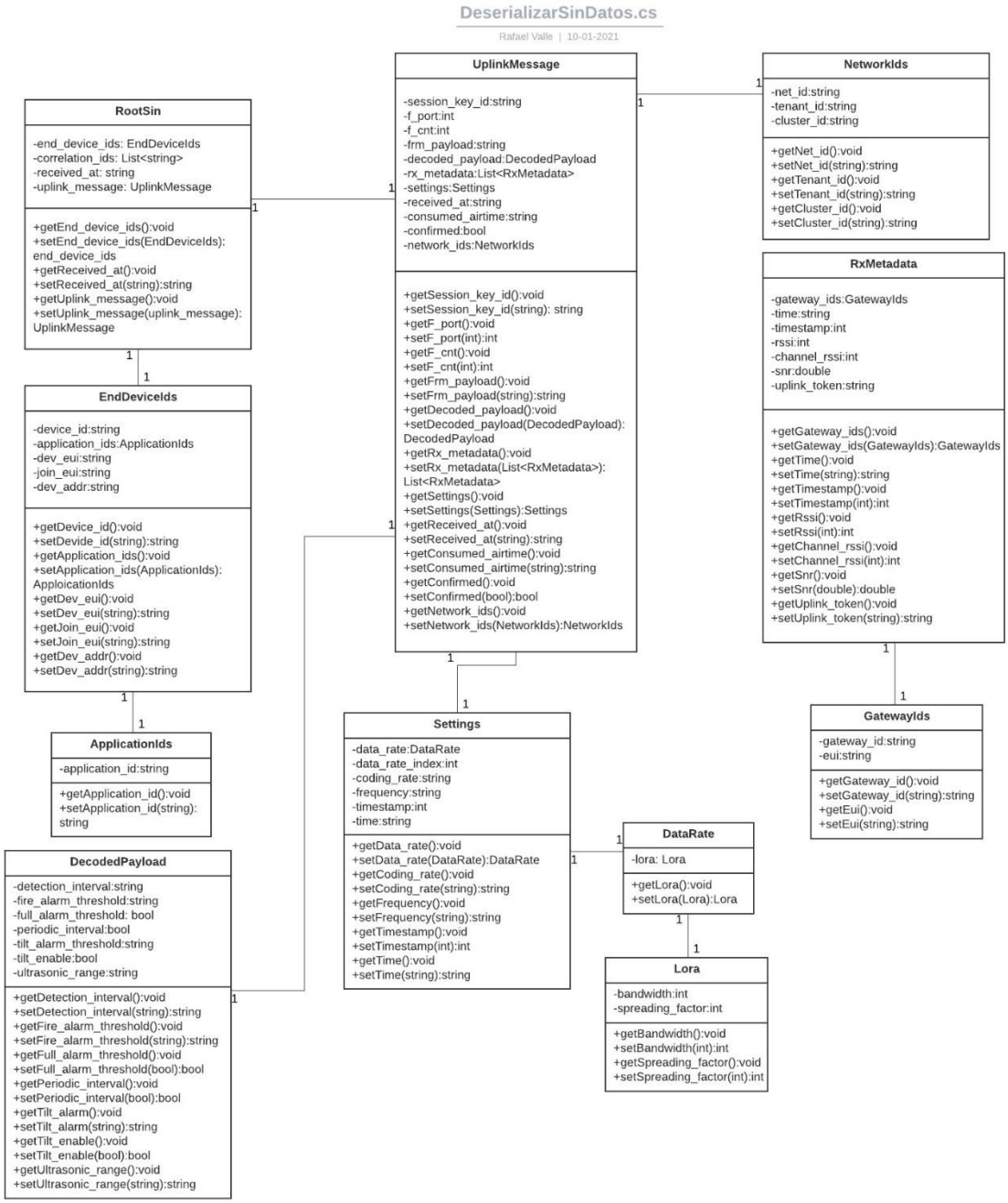


Diagrama de clases que contiene métodos para conectarse al bróker MQTT y almacenar los datos en la DB una vez recibidos.

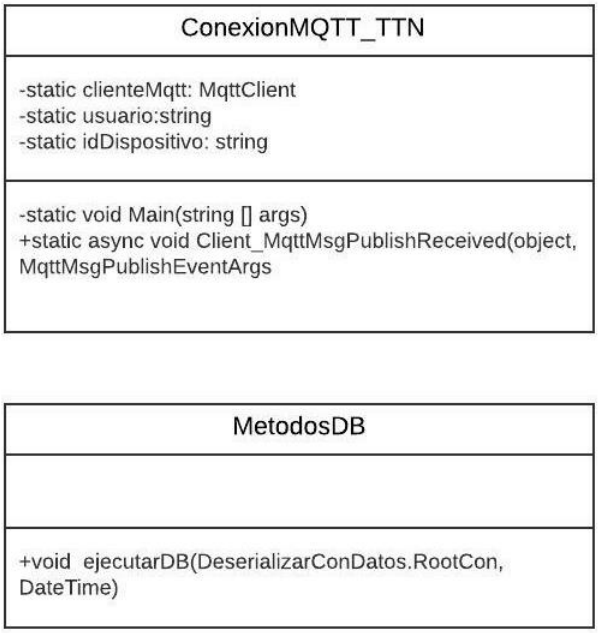
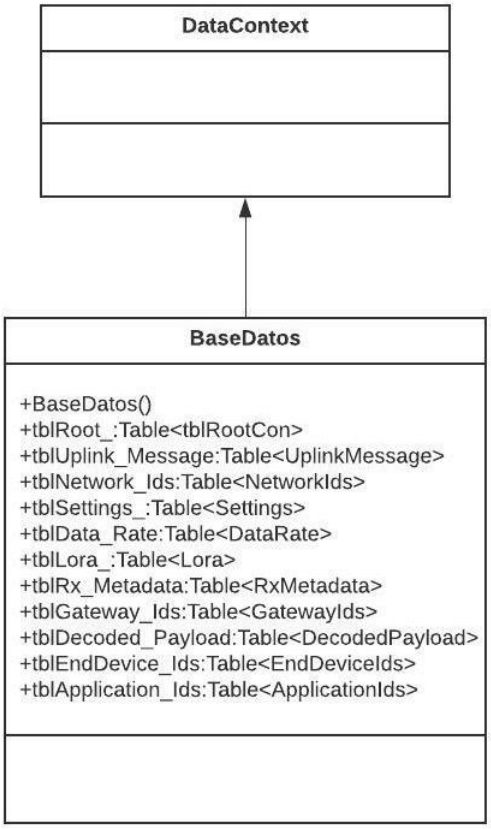
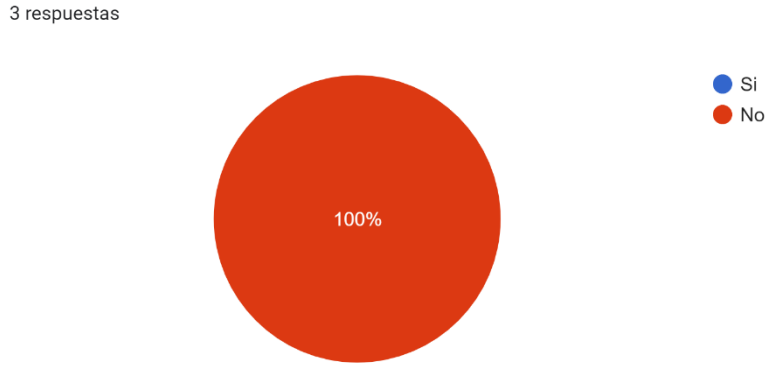


Diagrama de clases con el método para conectar a la base de datos.

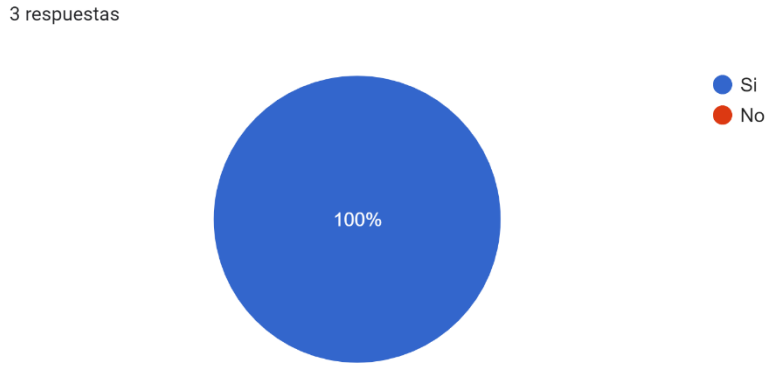


ANEXO C

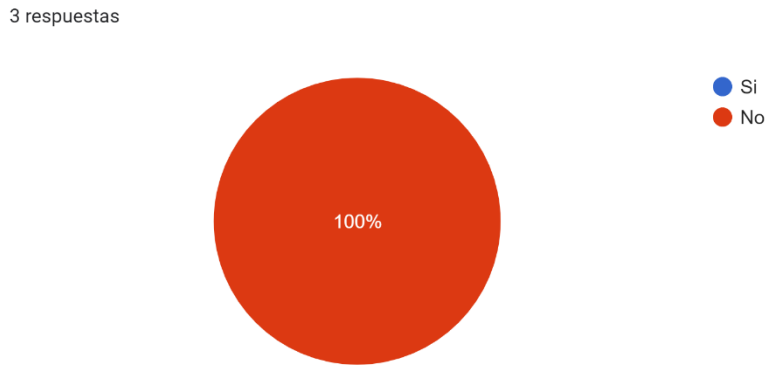
Pregunta 1: ¿Tuvo algún problema al momento de ejecutar la aplicación de consola?



Pregunta 2: ¿La aplicación de consola mostró los datos de forma ordenada y clara?



Pregunta 3: Al manipular la aplicación de consola, ¿encontró algún problema, como por ejemplo, cierre inesperado o similar?



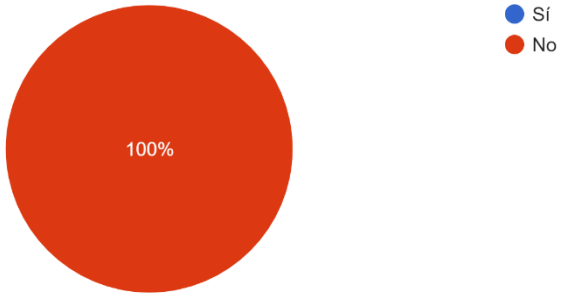
Pregunta 4: En caso de tener alguna recomendación u observación respecto a las preguntas anteriores, lo puede escribir a continuación

3 respuestas

Ninguna
No
nada

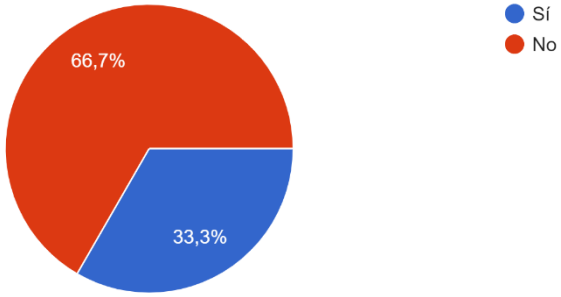
Pregunta 5: ¿Tuvo algún problema al momento de ejecutar la interfaz gráfca?

3 respuestas



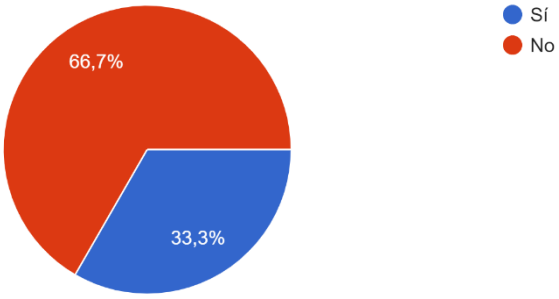
Pregunta 6: ¿Pudo ver de forma clara y legible los datos que se mostraron en la ventana principal?

3 respuestas



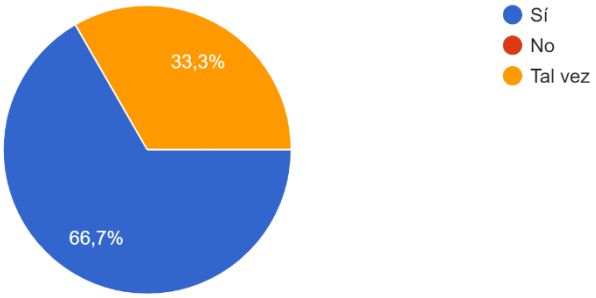
Pregunta 7: ¿Tuvo algún problema al manipular la aplicación?

3 respuestas



Pregunta 8: ¿La forma de acceder a la ventana con los mapas que contienen la ubicación de los contenedores fue lo suficientemente intuitiva?

3 respuestas

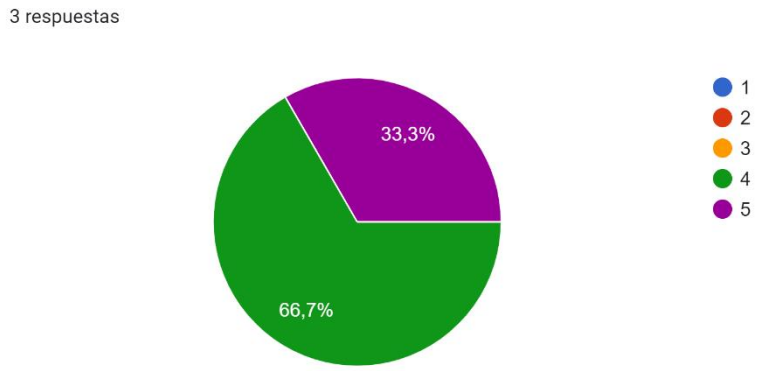


Pregunta 9: En caso de tener alguna recomendación u observación respecto a las preguntas anteriores, lo puede escribir a continuación

3 respuestas

- los colores en el casillero ocupacion no permitian ver los numeros de forma clara
- cambiar los colores para que los numeros se vean mejor
- los numeros pintados no se veian bien y al hacer click sobre el casillero superior derecho se abrieron muchas ventanas con mapas

Pregunta 10: Del 1 al 5, siendo 1 muy mala y 5 muy buena, ¿Cómo calificaría la interfaz gráfica de usuario?



ANEXO D

Enlace para descargar el video con el sistema puesto en funcionamiento.

https://drive.google.com/file/d/1GHtWgmmMnT3KMvadQ5KBHBWAePRW816y/view?usp=sharing