UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE ELECTROTECNIA Y COMPUTACIÓN

TRABAJO MONOGRÁFICO PARA OPTAR AL TÍTULO DE INGENIERO ELECTRÓNICO

Sistema para la Ubicación y Predicción de Arribo en Tiempo Real de Unidades de Transporte Colectivo Utilizando Nodos LoRa-GPS en Managua, Nicaragua

AUTORES:

- Br. Jehú Josué Guerrero Urrutia. 2017 – 0269U
- Br. Bryan Alexander Tinoco Baldelomar. 2017 – 0776U

TUTOR

Ing. Marcos Antonio Munguía Mena

Managua, Nicaragua

Julio 2023

DEDICATORIA

Dedicamos el presente trabajo monográfico realizado con mucho esfuerzo primeramente a Dios por habernos dado la vida, sabiduría e inteligencia para culminar con éxito nuestros estudios universitarios.

A nuestros padres quienes nos han guiado por el camino del bien y nos han brindado toda su comprensión y apoyo incondicional todos estos años.

A nuestros familiares y seres queridos que nos han brindado su apoyo a lo largo de este camino.

AGRADECIMIENTOS

Agradecemos primeramente a Dios por habernos permitido alcanzar este logro tan importante en nuestras vidas y darnos de su sabiduría para culminar esta monografía, proveyendo siempre lo necesario.

A nuestros padres, por su ayuda incondicional de toda una vida de esfuerzo y sacrificio, sus palabras de aliento, oraciones elevadas a Dios por nosotros y disposición para apoyarnos económicamente a pesar de las dificultades.

A la Universidad Nacional de Ingeniería por darnos la oportunidad de estudiar y ser parte de ella.

Agradecemos también a nuestro tutor, Ing. Marco Munguía, por brindarnos el apoyo científico-técnico para poder realizar este proyecto, y por sus recomendaciones y consejos durante el desarrollo de nuestro trabajo monográfico.

RESUMEN

El uso de nuevas tecnologías en los servicios de transporte urbano colectivo en las grandes ciudades es cada vez más frecuente, y evoluciona constantemente con el fin de mejorar la eficiencia operativa en general y proporcionar información en tiempo real sobre las unidades de transporte a los usuarios. Desafortunadamente, en Nicaragua no existe un sistema tecnológico implementado en el circuito de rutas que brinde información detallada y accesible acerca de los buses en operación.

En respuesta a estas carencias, que impiden a los usuarios tener información precisa sobre las próximas unidades de transporte colectivo, se decidió desarrollar un prototipo de Sistema para la ubicación y predicción de arribo en tiempo real de unidades de transporte colectivo en Managua, Nicaragua.

El resultado de este trabajo monográfico es un prototipo de sistema de ubicación que consta de nodos LoRa-GPS conectados a una red LoRaWAN gestionada por The Things Network. Este sistema también incluye la integración del servidor ThingSpeak, encargado de recolectar datos reales en la nube y ejecutar un algoritmo de predicción utilizando Matlab. Con ThingSpeak, es posible hacer uso de las API REST (Transferencia de Estado Representacional) para solicitar los resultados de los datos procesados, los cuales se muestran en un sistema que permite la visualización de información sobre la llegada y ubicación de las unidades de transporte urbano a través de una pantalla. La interfaz proyectada en la pantalla ha sido diseñada de tal modo que sea dinámica y fácil de interpretar para el usuario.

Este documento presenta los aspectos técnicos más relevantes del desarrollo de este proyecto monográfico. Se proporcionan los resultados obtenidos para que futuros colegas interesados puedan continuar mejorando este prototipo o utilizarlo como referencia en proyectos relacionados con este tema.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El transporte es una actividad necesaria y de gran importancia para el desarrollo económico y social de un país, ya que a través de él se mueven los principales elementos que lo componen, y es debido a su importancia que es necesario darle la atención que merece, procurando una solución moderna y dinámica que permita también el desarrollo y mejoramiento de las actividades (Sánchez, 2002).

Los buses urbanos de Managua, sin lugar a duda, son el principal medio de transporte de los habitantes y visitantes de esta ciudad. Según el INIDE[1] (2017), esta ciudad cuenta con un aproximado de 1,521,612 habitantes, de las cuales aproximadamente 853,000 personas son usuarios de 835 unidades de transporte colectivo que circulan diariamente (El Nuevo Diario, 2017).

El movimiento vehicular en Managua se torna caótico a lo largo del día, con problemas tales como: embotellamiento del tráfico, alta demanda de pasajeros, irregularidad en los horarios de los buses urbanos, y retardos inesperados en general. Muchos pasajeros se enfrentan al problema de llegar tarde a su destino por esperar una unidad de transporte de la cual no poseen la información concreta de su horario, incluso si tienen la posibilidad de tomar otras unidades o medios de transporte alternativos para movilizarse.

En el Reglamento de la Ley General de Transporte Terrestre se establece que:

> "El chequeo y control del tiempo de recorrido en las rutas intermunicipales e intramunicipales deberá realizarse por medio de puestos de control fijos o móviles a lo largo del recorrido de cada ruta, por medio de relojes mecánicos o electrónicos, **u otros sistemas de controles electrónicos**, en los que se hará constar la hora exacta en que la unidad de transporte pasó por determinado puesto de control, y cualquier otra información adicional necesaria para mejorar las operaciones" (Ley General de Transporte Terrestre, 2005, Arto.30).

[1] INIDE: Instituto Nacional de Información de Desarrollo

Debido a la problemática del movimiento vehicular e irregularidad en los horarios de las unidades, entre otros problemas, y que la ley permite controles de tiempo con **sistemas electrónicos** administrado por el Instituto Regulador de Transporte del Municipio de Managua (IRTRAMMA), se propone un prototipo de sistema para la ubicación y predicción de arribo en tiempo real de unidades de transporte colectivo en Managua utilizando tecnología LoRa-GPS implementado en una red LoRaWAN, en beneficio tanto de los usuarios como de las autoridades reguladoras de transporte.

Habitualmente, para el desarrollo de proyectos de geolocalización se utilizan tecnologías basadas en GPS y GSM, debido a que son tecnologías estándares de telecomunicaciones móviles y están extendida por todo el mundo, sin embargo, muchos sistemas de geolocalización pueden ser remplazados utilizando tecnología LoRa debido a las múltiples ventajas que ofrece, tales como: bajo consumo, largo alcance, costo de inversión y sostenibilidad (GSMA, 2019).

La tecnología LoRa está adquiriendo mucho impulso en su uso a través de LoRaWAN, que es un estándar de control de acceso al medio. Mediante esta especificación, es posible gestionar la comunicación de dispositivos inalámbricos, y así revolucionar las soluciones de IoT.

En este documento se presenta el desarrollo del prototipo de sistema para la ubicación y predicción de arribo en tiempo real de unidades de transporte colectivo en el cual se implementan nodos finales basado en tecnología LoRa-GPS capaces de transmitir datos de geolocalización de las unidades de transporte a un gateway[2], el cual se encarga de recibir estos datos y enviarlos a la nube. Finalmente, el sistema consta de una Raspberry Pi y una pantalla la cual debe ubicarse en un lugar estratégico en las estaciones de buses de la ciudad de Managua, con el fin de mostrar la información de llegada y ubicación de las unidades de Transporte Urbano Colectivo (TUC).

[2]gateway: Puerta de enlace.

OBJETIVOS

Objetivo general

- Desarrollar un prototipo de sistema para la localización y predicción de arribo en tiempo real para unidades de transporte colectivo utilizando tecnología LoRa-GPS en Managua, Nicaragua.

Objetivos específicos

- Elaborar el diagrama funcional para interconexiones y configuraciones apropiadas de los dispositivos electrónicos.

- Diseñar nodos LoRa-GPS funcionales para monitorear la ubicación de las unidades de transporte.

- Implementar el algoritmo basado en el Método Real-Time Travel Time (RTTT) para predecir el tiempo de llegada del autobús.

- Desarrollar un sistema con Raspberry Pi que permita la visualización de información de llegada y ubicación de las unidades de transporte urbano a través de una pantalla.

- Implementar un ambiente de pruebas que conste de una red LoRaWAN con nodos LoRa-GPS para validar el funcionamiento.

JUSTIFICACIÓN

El prototipo de sistema para la ubicación en tiempo real de unidades de TUC en Managua, proporcionará información precisa y actualizada sobre el arribo de las próximas unidades de TUC, lo que mejorará la experiencia de los usuarios al permitirles tomar decisiones informadas y evitar aglomeraciones innecesarias. Asimismo, se fomentará una mayor eficiencia en la operación de las unidades de transporte, permitiendo una asignación más precisa de recursos. Además, este prototipo de sistema es la base para la realización de un proyecto integral que busca interconectar la red de transporte público de Managua mediante la implementación de la tecnología LoRaWAN.

El desarrollo de este prototipo, busca implementar una solución eficiente y económica para la transmisión de datos de ubicación en tiempo real de las unidades de transporte urbano en la ciudad de Managua, pues si cada unidad de transporte urbano transmitiera su ubicación geográfica periódicamente a un servidor GPS a través de una red celular, se debería pagar un plan de internet por cada unidad, teniendo en cuenta que son aproximadamente 835 unidades (El nuevo diario, 2017) y se debe pagar $20.99 al mes por el plan de internet (precio cotizado a la empresa de Tigo) se genera un gasto al mes de aproximadamente $17,500, sin embargo, cuando se transmite información utilizando tecnología LoRa solo se pagarían planes de internet para los gateway, lo que supone un ahorro considerable ya que la geolocalización se haría en la red local LoRaWAN.

Se consideró de vital importancia llevar a cabo este trabajo monográfico, para evidenciar que se puede resolver el problema de la insuficiencia de información en los horarios de las unidades de transporte colectivo en tiempo real utilizando tecnologías de vanguardia como es LoRa. Cabe destacar que en la Universidad Nacional de Ingeniería (UNI), no hay antecedentes monográficos en el cual se haya utilizado tecnología LoRa.

La investigación y conocimientos aportados quedarán respaldados en los repositorios de la UNI, para que futuras generaciones realicen posibles mejoras o sirva de referencia para posteriores estudios en otras áreas que se vinculen de alguna manera con este tema.

CAPÍTULO I. MARCO TEÓRICO

A continuación, se presentará los soportes teóricos necesarios para fundamentar el desarrollo del prototipo de sistema propuesto en el presente documento.

1.1. Sistema de monitoreo de transporte

Hoy en día, con el surgimiento de nuevas tecnologías, los Sistemas Inteligentes de Transporte (ITS), se utilizan cada vez más en los sistemas de transporte público con el fin de brindar información en tiempo real (RTI[3]) tanto a pasajeros como a operadores (ver Figura 1)

El término ITS se refiere a la aplicación integrada de tecnologías en comunicaciones, control e información, que, después de un diseño e implementación, se convierten en una gran herramienta en el mercado de soluciones orientadas a mejorar la calidad, seguridad, accesibilidad y gestión del sistema de transporte (PIARC, s.f.).

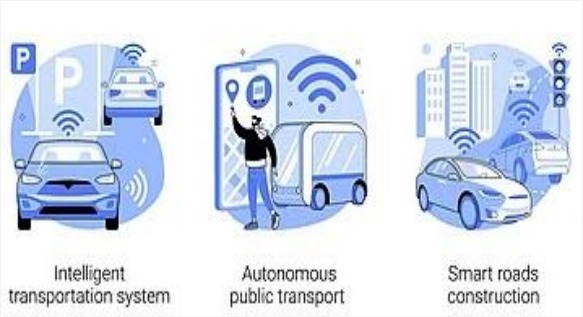


Figura 1. Sistema de transporte inteligente

1.1.1. Información de los sistemas inteligentes de monitoreo

Debido al protagonismo que han tomado los ITS estos están siendo desplegados por las cooperativas y agencias en los sistemas de transporte moderno, estas últimas implementan con éxito o están en proceso de transformar sus sistemas adaptando aplicaciones ITS en la planificación del transporte público, así como en el mantenimiento y operación de infraestructuras.

[3] RTI: Por sus siglas en inglés: Real Time Information

Con el surgimiento de estos sistemas, como el mostrado en la Figura 2, las cooperativas de transporte público tienen a su disposición datos sobre el tiempo de viaje del autobús, su ubicación, velocidad, cantidad de pasajeros a bordo, entre otros; siendo la información relacionada con la predicción del tiempo restante hasta la llegada de la próxima unidad de transporte la RTI más comúnmente proporcionada y uno de los principales focos de interés de los pasajeros (Loutos, 2013).

Figura 2. Información de sistema de transporte inteligente

1.1.2. Actualidad del sistema de Transporte Colectivo en Managua

De acuerdo a la visita realizada por Cruz y Muñoz (2021), al encargado de una de las cooperativas de transporte de Managua, el control de las flotas de unidades de transporte colectivo funciona de la siguiente manera:

Cada unidad de transporte viene dotada de un dispositivo GPS ubicado en un lugar no visible del vehículo, dispositivos que están conectados a un centro de control vía internet. Existen marcadores de posición ubicados en puntos establecidos de la ruta en que transitan las unidades de esta cooperativa; cuando las unidades transitan por estos puntos se envía la información del dispositivo GPS al centro de control de la cooperativa; información que es utilizada para elaborar reportes de actividad y de cumplimiento de itinerario de los conductores de los buses para que de esta manera el director de la cooperativa pueda tomar las medidas correctivas pertinentes.

Cabe destacar que los únicos que tienen acceso a esta información son los encargados de la cooperativa, por lo que los usuarios y público en general no disponen de información acerca de las unidades de transporte de su interés que están en circulación en ese momento.

1.1.3. Sistema LoRa – GPS implementado en una red LoRaWAN

Un sistema de monitoreo en tiempo real de unidades de transporte colectivo implementado en una red LoRaWAN, como el que se muestra en la Figura 3, podría ayudar a solventar la falta de información a los usuarios en Managua. Para esto se requiere de:

- El diseño nodos LoRa-GPS funcionales que permitan obtener la ubicación en tiempo real de una unidad TUC, resaltando que dicha información es vital para la predicción del tiempo de llegada.
- La utilización de un gateway LoRa de código abierto que permita unir la red inalámbrica LoRa a una red IP a través de Wi-Fi, Ethernet o alguna red móvil.
- La creación de una aplicación en un servidor basado en la nube capaz de registrar y procesar los datos proporcionado por el nodo.
- El desarrollo de un sistema de visualización de información de las unidades de transporte urbano a través de una pantalla.

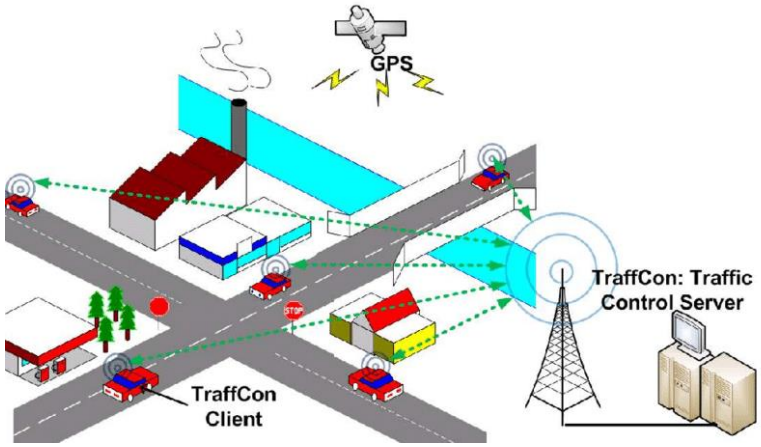


Figura 3. Monitoreo GPS de vehículos en tiempo real

1.2. LoRa (Long Range)

LoRa es la capa física o tipo de modulación utilizada para crear enlaces de comunicación de largo alcance, bajo consumo de energía, baja tasa de datos y una transmisión de datos segura. La modulación del espectro Spread LoRa está basada en CSS (Chirp Spread Spectrum), patentada y desarrollada por Semtech[4] (Pradeeka, 2019).

[4] Semtech: Proveedor de semiconductores analógicos y de señal mixta y algoritmos avanzados.

LoRa es una técnica de modulación inalámbrica que codifica información en ondas de radio utilizando pulsos de chirp, donde los chirp (también conocidos como símbolos) son el portador de datos. La transmisión modulada LoRa es resistente a las perturbaciones y se puede recibir a través de grandes distancias.

Esta modulación es ideal para aplicaciones que transmiten pequeños fragmentos de datos con tasas de bits bajas. Los datos se pueden transmitir a una distancia mayor en comparación con tecnologías como WiFi, Bluetooth o ZigBee. Estas características hacen que LoRa sea ideal para sensores y actuadores que funcionan en modo de bajo consumo, además de funcionar en las bandas sin licencia , por ejemplo, 915 MHz, 868 MHz y 433 MHz (The Things Network, s.f.).

1.2.1. Chirp Spread Spectrum

La modulación Chirp Spread Spectrum (CSS) mantiene las mismas características de baja potencia que la modulación FSK. Spread spectrum es una técnica que utiliza pulsos chirp modulados en frecuencia lineal de banda ancha para codificar información.

CSS fue desarrollado para aplicaciones de radar en la década de 1940. Se ha utilizado en comunicaciones militares y espaciales durante décadas debido a sus largas distancias de comunicación, bajos requisitos de potencia de transmisión y menos interferencias.

LoRa utiliza el espectro de dispersión de chirp para codificar datos. Cada bit se extiende por un factor de chirp. El número de bits por símbolo se denomina factor de ensanchamiento (SF)[5]. CSS utiliza factores de propagación del 7 al 12. Los factores de propagación pequeños proporcionan altas tasas de datos y requieren menos tiempo en el aire, en cambio, los factores de propagación grandes proporcionan bajas tasas de datos, mayor alcance y requieren más tiempo en el aire.

[5] SF: Spreading Factor

La modulación LoRa es más compleja y resistente al ruido de fondo en comparación a FSK. En lugar de usar solo las dos frecuencias de FSK, barre entre las dos frecuencias. En la Figura 4 se presenta algunos detalles de la modulación LoRa. Las imágenes de la parte superior muestran los barridos de frecuencia con los distintos factores de propagación, y ancho de banda dado, característicos de LoRa. Mientras que las imágenes de la parte inferior muestran la estructura de un paquete LoRa en la capa física y los barridos de frecuencia de arriba hacia abajo (up-down chirp) y de abajo hacia arriba (down-up chirp) (Pradeeka, 2019).

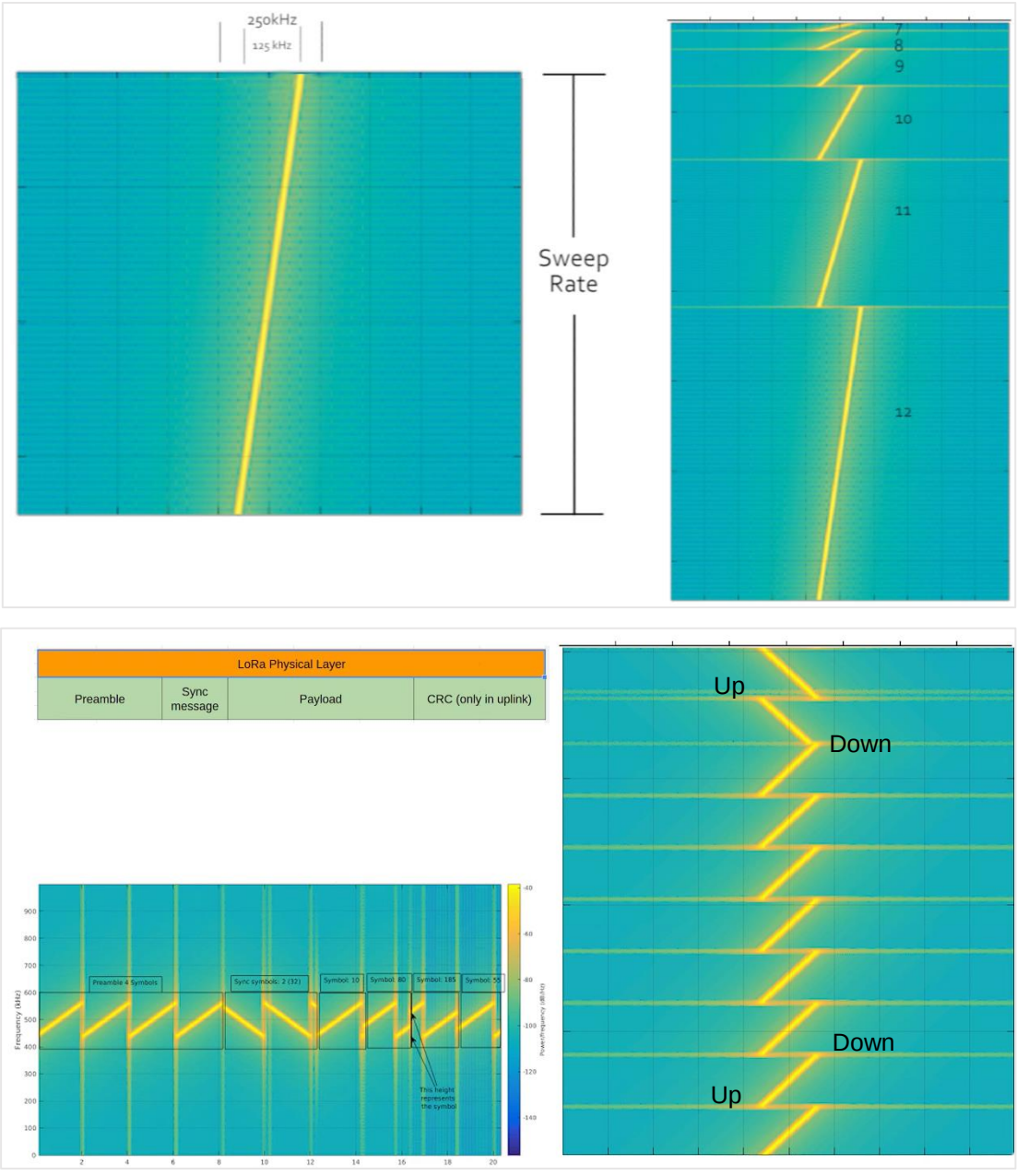


Figura 4. Modulación Chirp Spread Spectrum (Wenner, 2017)

La modulación de espectro ensanchado empleada en LoRa tiene las siguientes propiedades: Ancho de banda escalable, envolvente constante/bajo consumo de energía, alta robustez, resistencia a la multipath/fading, resistencia al efecto Doppler, capacidad de largo alcance, capacidad de red mejorada.

1.2.2. Rendimiento de LoRa ante el efecto Doppler

Es bien sabido que cuando la fuente de una onda (transmisor) se mueve con respecto a un observador (receptor), el observador recibe una frecuencia que difiere de la radiada. La diferencia depende de la dirección y la velocidad del movimiento de la fuente. Cuando se trata de comunicaciones inalámbricas, como LoRa, este efecto puede dificultar la correcta recepción de la señal, notándose un deterioro de la señal a medida que se aumenta la velocidad. Según el alcance de este efecto, la señal se recibe o se descarta. Sin embargo, se conoce que el CSS es resistente al efecto Doppler, es decir CSS está diseñado para mitigar los efectos negativos de fluctuaciones de frecuencia causadas por el movimiento. LoRa, al ser una modulación que emplea la técnica de CSS, hereda un grado de resiliencia de este.

Según Petajäjärvi et al. (2017) en su estudio sobre el rendimiento de LoRa y su robustez ante el efecto Doppler, con la realización de diversas pruebas experimentales utilizando la puerta de enlace LoRa IoT de Kerlink y un dispositivo final equipado con el transceptor SX1272 a bordo de un automóvil que viajaba a velocidades entre 40 y 100 Km/h. Utilizando SF con valores de 12 y 11, cuando la velocidad supera los 40 Km/h y 76 Km/h respectivamente, el rendimiento de la comunicación se deteriora, debido a que la duración del símbolo modulado por LoRa excede el tiempo de coherencia, mayormente cuando la distancia entre el transmisor y receptor aumenta, apareciendo otros factores influyentes como la disminución del SNR y el RSSI.

Sin embargo, en un estudio más reciente, realizado por Liando et al. (2019) para cuantificar el efecto del desplazamiento Doppler en los chirps LoRa, se configuró una puerta de enlace, basada en el chip SX1301, ubicado cerca de la carretera, y se adjuntó un nodo que incluía el transceptor SX1276 a un automóvil utilizado como transmisor móvil. Se condujo el automóvil realizando pruebas individuales para las velocidades de 50, 60, 70 y 80 km/h.

Cada transmisión de 50 bytes se realizó con un SF12 con una duración de paquete de 2.35 segundos para garantizar tiempo suficiente para registrar un paquete en los estados de acercamiento y pasaje.

Teniendo como resultado de la prueba una buena recepción de paquetes (>85%) en la puerta de enlace para las velocidades probadas, no obstante, es de vital importancia mencionar que a las velocidades de 50 y 80 Km/h cuándo el vehículo se acercaba al gateway se perdían el 50% de los paquetes enviados desde en nodo.

En ambos casos se concluyó que la viabilidad del uso de LoRaWAN en aplicaciones móviles en presencia del efecto Doppler va a depender de factores como la velocidad promedio de los nodos, la velocidad del cambio de dirección de incidencia de la señal y los parámetros de LoRa utilizados. Vale la pena señalar que paquetes de menor tamaño y con SF más bajos se ven menos afectados por el efecto Doppler, y, por lo tanto, pueden ser más adecuados para escenarios móviles.

1.3. LoRaWAN

El protocolo de interconexión de la red a utilizar en el presente proyecto es LoRaWAN (Long Range Wide Area Network) el cual define la comunicación y la arquitectura del sistema. El protocolo y la arquitectura de la red están directamente relacionados para determinar la optimización de la batería del nodo, la capacidad de la red, la calidad del servicio y las aplicaciones atendida por la red (LoRa Alliance, 2015).

LoRaWAN es un protocolo de capa de control de acceso a medios (MAC) construido sobre la modulación LoRa. Esta capa de software define cómo los dispositivos usan el hardware LoRa, por ejemplo, cuando transmiten y el formato de los mensajes.

En la mayoría de los casos, LoRaWAN usa la modulación LoRa, lo que hace que funcione bien con el ruido del canal, el desvanecimiento multitrayecto y el efecto Doppler, incluso a baja potencia.

La velocidad de datos depende del ancho de banda utilizado (BW) y del factor de dispersión (SF). LoRaWAN puede usar canales con un ancho de banda de

125 KHz, 250 KHz o 500 KHz, según la región o el plan de frecuencia. El factor de dispersión lo elige el dispositivo final e influye en el tiempo que se tarda en transmitir una trama (The Things Network, s.f.).

LoRaWAN está diseñada para conectar inalámbricamente cualquier “cosa”, móvil o estática, a internet en redes regionales, nacionales e internacionales. En la Figura 5 se puede observar el modelo en capas de una red LoRaWAN.

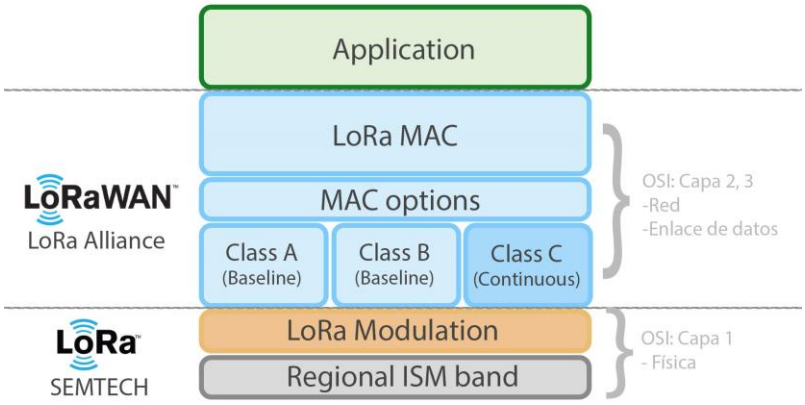


Figura 5. Modelo en capa de LoRaWAN (Becolve Digital, 2022).

1.3.1. Elementos de una red LoRaWAN

LoRaWAN, a como se puede apreciar en la Figura 6, se conforma de nodos finales (dispositivos finales), puerta de enlaces (concentradores o gateway), un servidor de red y servidores de aplicaciones. En una red LoRaWAN, los datos son transmitidos por un nodo final, y se reciben a través de una o varias puertas de enlaces. Una vez que se reciben los datos, la puerta de enlace reenvía los paquetes a través de una red celular, ethernet o satelital. El software que se ejecuta en la puerta de enlace es el encargado de reenviar cualquier paquete de dato entrante al servidor de red. Este software se conoce como *packet forwarder (reenviador de paquetes)*. El servidor de red envía y recibe mensajes LoRaWAN hacia y desde los dispositivos y se comunica con los servidores de aplicaciones ascendentes. El servidor de aplicaciones es el destino de los datos de la aplicación que se envían como carga útil en mensaje de LoRaWAN (Pradeeka, 2019).

Estos datos serán procesados para mostrar a través de una pantalla informativa y monitorear en tiempo real los buses de TUC.

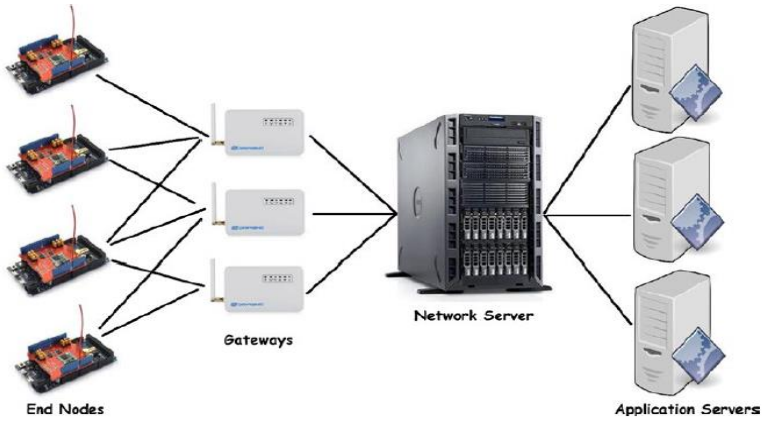


Figura 6. Elementos de una red LoRaWAN (Pradeeka, 2019)

1.3.2. Clases y métodos de activación

El protocolo LoRaWAN tiene tres clases diferentes de dispositivos finales para abordar las diferentes necesidades reflejadas en la amplia gama de aplicaciones, estas son: Clase A, Clase B y Clase C, diferenciándose entre ellas por la relación entre su latencia y duración de la batería.

Todos los dispositivos implementan Clase A, siendo la B y C extensiones de las especificaciones de los dispositivos de Clase A, y admitiendo toda la comunicación bidireccional (enlace ascendente[6] y descendente[7]). Según LoRa Alliance (2022), las clases se definen de la siguiente manera:

Clase A: Es la clase compatible con todos los dispositivos finales; en esta la comunicación siempre la inicia el nodo y es totalmente asíncrona. Cada transmisión de enlace ascendente va seguida de dos breves ventanas de enlace descendente (RX1 y RX2), lo que brinda la oportunidad de comunicación bidireccional. Una de las ventajas de esta clase es su bajo consumo energético al entrar en modo de suspensión al tiempo definido por su propia aplicación, al mismo tiempo que permite la comunicación de enlace ascendente en cualquier momento.

Clase B: Los dispositivos finales de clase B permiten más espacios de mensajes de enlace descendente, reduciendo la latencia de los mensajes, pero al mismo tiempo hace que sea menos eficiente energéticamente. Estos se

[6] Enlace ascendente: Dispositivo final envía un mensaje a la puerta de enlace (Uplink)
[7] Enlace descendente: Mensaje recibido de la puerta de enlace (Downlink)

sincronizan con la red mediante balizas periódicas y abren “ranuras de ping” de enlace descendente en horarios programados para recibir mensajes, todo esto a cambio de un mayor consumo energético en el nodo, aunque sigue siendo lo suficientemente bajo para aplicaciones alimentadas por batería.

Clase C: Además de la estructura de clase A de enlace ascendente seguida de dos ventanas de enlace descendente, la clase C reduce aún más la latencia en el enlace descendente al mantener abierto el receptor del dispositivo final en todo momento en que el dispositivo no está transmitiendo es decir enlace ascendente activo. Esta ventana de recepción continua hace que sea el menos eficiente energéticamente, y en la mayoría de los casos necesita una fuente de energía constante para operar.

Métodos de activación

Para que un dispositivo final pueda ser parte de una red LoRaWAN tiene que ser personalizado y activado. La personalización se realiza al darle un identificador único a cada dispositivo con el fin de conocer el dispositivo final y saber a qué aplicación corresponde. La activación puede ser lograda de dos maneras, por medio de OTAA[8] o por ABP[9] (The Things Industries, 2023)

OTAA: Los dispositivos finales OTAA se aprovisionan con claves raíz. El dispositivo final realiza un procedimiento de unión con una red LoRaWAN, durante el cual se asigna una DevAddr dinámica a un dispositivo final y se utilizan claves raíz para derivar claves de sesión. Por lo tanto, las claves de sesión cambian a medida que se establece cada nueva sesión.

ABP: En este una DevAddr fija y las claves de sesión para una red preseleccionada se codifican en el dispositivo final, y permanecen iguales durante toda la vida útil de un dispositivo final ABP. Con este modo, un dispositivo final omite el procedimiento de unión que parece ser más simple.

1.3.3. Tasa de transferencia de datos (Data Rates)

LoRaWAN es una tecnología que utiliza un salto de frecuencia para las comunicaciones entre dispositivos finales y puertas de enlace. Además del

[8] OTAA: Activación por aire
[9] ABP: Activación por personalización

salto de frecuencia, cada paquete de comunicación incluye una configuración de Velocidad de Datos (DR)[10] variable. La selección del DR permite equilibrar dinámicamente el rango de comunicación y la duración del mensaje.

La tecnología de espectro ensanchado utilizada en LoRaWAN garantiza que las comunicaciones con diferentes velocidades de datos no interfieran entre sí y crea un conjunto de canales virtuales de 'código' que aumentan la capacidad de la puerta de enlace.

Para maximizar la duración de la batería de los dispositivos finales y la capacidad general de la red, el servidor de red LoRaWAN administra la configuración de DR y la potencia de salida de RF para cada dispositivo final de forma individual. Esto se logra mediante un esquema de tasa de datos adaptable (ADR).

Las velocidades de LoRaWAN oscilan entre 0,3 Kbps y 50 Kbps. La tasa de transferencia de datos es un factor importante, ya que indica la cantidad de datos que se pueden transmitir de manera efectiva en la red, evitando sobrecargas. La selección adecuada de la tasa de transferencia de datos permite equilibrar el rango de comunicación y la duración del mensaje (The Things Network, s.f.)

1.3.4. Banda ICM

LoRaWAN define 64 canales de enlace ascendente de 125 KHz en el intervalo de frecuencia de 902.3 MHz a 914.9 MHz en incrementos de 200 kHz. Además, contiene ocho canales adicionales de enlace ascendente de 500 KHz en incrementos de 1,6 MHz en el intervalo de 903 MHz a 914.9 MHz. Los ocho canales de enlace descendente tienen un ancho de 500 kHz desde 923.3 MHz hasta 927.5 MHz. La potencia de salida máxima en la banda de 902-928 MHz en América del Norte es de +30 dBm, pero, para la mayoría de los dispositivos es suficiente con +20 dBm. Según la FCC[11], no hay limitaciones de ciclo de trabajo, pero hay un tiempo de permanencia máximo de 400 ms por canal (LoRa Alliance, 2015).

[10] DR: Data Rate
[11] FCC: Federal Communication Commission

LoRaWAN también trabaja en otras bandas de frecuencia de ICM[12], dichas bandas están reservadas internacionalmente para uso no comercial de radiofrecuencia electromagnética en áreas industrial, científica y médica. En Nicaragua, TELCOR[13] es el ente regulador de las telecomunicaciones y servicio postal, este mismo, en el Reglamento de uso del espectro radioeléctrico y de los servicios de radiocomunicaciones, 1997, Arto.69, expone:

> "Los equipos para aplicaciones industriales, científicas y médicas, denominados ICM registrados ante TELCOR, no requerirán de permiso para operar dentro de las bandas de frecuencias designadas por TELCOR para operar en aplicaciones industriales, científicas y médicas. Dichos equipos podrán operar en bandas de frecuencias diferentes a las designadas, si cumplen con las especificaciones particulares que señale TELCOR para cada caso, debiendo adoptar todas las medidas necesarias para garantizar el no causar interferencia perjudicial a los equipos, sistemas y redes de radiocomunicaciones autorizados o que se autorice en las bandas de frecuencia de que se trate.
>
> No podrá realizarse la operación de los equipos ICM en las bandas de frecuencias 490 - 510 KHz, 2170 -2194 KHz, 8354 - 8374 KHz, 121.4 - 121.6 MHz, 156.7 - 156.9 MHz, 242.8 - 243.2 MHz y en las demás bandas de frecuencias atribuidas nacional e internacionalmente para socorro, seguridad, búsqueda y salvamento" (Reglamento de uso del espectro radioeléctrico y de los servicios de radiocomunicaciones, 1997, Arto.69).

En La Gaceta[14] (2021) en el Acuerdo Administrativo N.º 002-2021 sobre el Cuadro Nacional de Atribución de Frecuencias se informa sobre la actual atribución de bandas de frecuencias internacional en la Región 2 (en la cual se incluye Nicaragua) y su compatibilidad con la asignación nacional. En la Figura

[12] ICM: Industrial, Científica y Médica
[13] TELCOR: Instituto Nicaragüense de Telecomunicaciones y Correos
[14] La Gaceta: Diario Oficial del Gobierno de la República de Nicaragua. Se encarga de publicar leyes, decretos, acuerdos, resoluciones y demás actuaciones de los poderes del Estado.

7 se puede observar la asignación de bandas libres ICM, de las cuales resalta la banda 902-928 MHz para nuestra región.

5.150 Las bandas:

13 553-13 567 kHz	(frecuencia central 13 560 kHz),
26 957-27 283 kHz	(frecuencia central 27 120 kHz),
40,66-40,70 MHz	(frecuencia central 40,68 MHz),
902-928 MHz	en la Región 2 (frecuencia central 915 MHz),
2 400-2 500 MHz	(frecuencia central 2 450 MHz),
5 725-5 875 MHz	(frecuencia central 5 800 MHz) y
24-24,25 GHz	(frecuencia central 24,125 GHz)

Están designadas para aplicaciones industriales, científicas y médicas (ICM). Los servicios de radiocomunicación que funcionan en estas bandas deben aceptar la interferencia perjudicial resultante de estas aplicaciones. Los equipos ICM que funcionen en estas bandas estarán sujetos a las disposiciones del número **15.13**

Figura 7. Bandas para aplicaciones ICM (La Gaceta, 2021)

En lo que refiere a la banda de 902-928 MHz, se puede observar en la Figura 8 que existe una compatibilidad entre la Región 2 (izquierda) y la asignación nacional (derecha) para el servicio de Aficionados, que es definido como un servicio dirigido a la instrucción individual, la intercomunicación y los estudios técnicos, efectuado por aficionados que se interesan en la radiotecnia con carácter exclusivamente personal y sin fines de lucro (La Gaceta, Diario Oficial, 2021).

902 - 928	**902 - 928**	
FIJO	FIJO	
Aficionados	Aficionados	
Móvil salvo móvil aeronáutico 5.325A	Móvil salvo móvil aeronáutico	
Radiolocalización	Radiolocalización	**NCG10 NCG32**

Figura 8. Rango de frecuencia de 902-928 MHz (La Gaceta, 2021)

Sabiendo esto, podemos afirmar que, de forma legal, se pueda llevar a cabo este prototipo de sistema para la ubicación en tiempo real de unidades de transporte colectivo utilizando tecnología LoRa-GPS implementado en una red LoRaWAN en la banda de 915 MHz, al entrar en la categoría de Aficionados en este rango de frecuencia.

1.3.5. Seguridad

La seguridad es indispensable para cualquier implementación masiva de IoT, con el fin de proteger los datos e información. En este proyecto, una de las

características fundamentales es la extensa incorporación de nodos LoRa-GPS conectado a internet por medio de una red LoRaWAN.

Afortunadamente LoRaWAN incluye dos capas de criptografía:

- **Network Session Key:** Clave de sesión de red única de 128 bits que garantiza la seguridad entre el dispositivo final y el servidor de red.
- **Application Session Key:** Clave de sesión de aplicación única de 128 bits que garantiza la seguridad de extremo a extremo a nivel de aplicación.

Se utilizan algoritmos AES (Advanced Encryption Standard) para proporcionar autenticación e integridad de paquetes al servidor de red y cifrado de extremo a extremo al servidor de aplicación.

Las claves pueden ser activadas mediante personalización (ABP), o pueden ser activadas por aire (OTAA) en el campo. OTAA permite que los dispositivos sean reencriptados si es necesario (LoRa Alliance, 2017).

1.4. Nodo LoRa-GPS (Dispositivo final)

Un nodo LoRa-GPS, a como se muestra en la Figura 9, está conformado por tres dispositivos electrónicos fundamentales: GPS (determina la ubicación real), microcontrolador (prepara la información para el transmisor) y módulo de transmisión y recepción inalámbrico con tecnología LoRa.

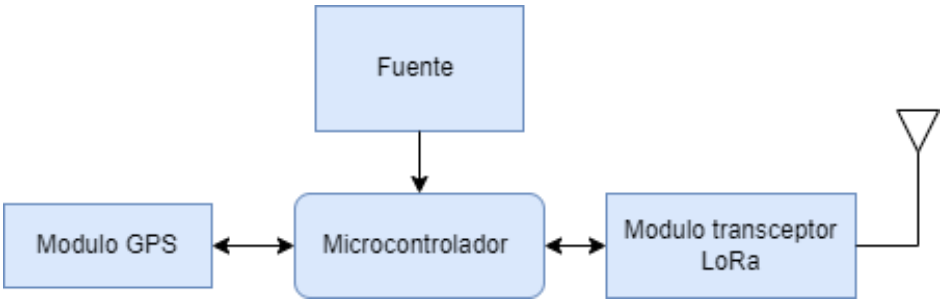


Figura 9. Diagrama de bloques Nodo LoRa-GPS. [Autor]

1.4.1. Módulo GPS

Los Sistemas de Posicionamiento Global (GPS[15]), puede considerarse uno de los componentes fundamentales de los sistemas inteligentes de transporte,

[15] GPS: Por sus siglas en inglés Global Positioning System

éstos indican referencias geográficas precisas en tiempo real y de amplia cobertura en un área rural o urbana, lo cual permite realizar un control preciso de los sistemas de transporte teniendo disponible la información de la ubicación por medio de receptores GPS (2008 y Prieto, 2015).
GPS es un sistema de localización diseñado por el Departamento de Defensa de los Estados Unidos con fines militares para proporcionar estimaciones precisas de posición, velocidad y tiempo.

La arquitectura del GPS se descompone en tres segmentos básicos:

Segmento espacio: Formado por 24 satélites GPS con una órbita de 26560 Km de radio y un periodo de 12 horas.

Segmento control: Consta de cinco estaciones monitoras encargadas de mantener en órbita los satélites y supervisar su correcto funcionamiento, tres antenas terrestres que envían a los satélites las señales que deben transmitir y una estación experta de supervisión de todas las operaciones.

Segmento usuario: Formado por las antenas y los receptores pasivos situados en tierra. Los receptores, a partir de los mensajes que provienen de cada satélite visible, calculan distancias y proporcionan una estimación de posición dónde se encuentra un objeto en el instante que se quiera (Pozo et al., s.f.).
El GPS es un elemento indispensable para el desarrollo del proyecto, pues provee la ubicación en tiempo real, siendo esta unas de las variables de entrada al algoritmo de estimación de tiempo de llegada de los buses.

El módulo GY-NEO6MV2, mostrado en la Figura 10, es un dispositivo de costo accesible y alto rendimiento de la serie NEO-6 de módulos GPS basada en u-blox 6, este incluye un motor de posicionamiento operando en la banda de frecuencia GPS L1 (1575.42 MHz). Además, su capacidad de integración con opciones de conectividad flexible en paquete pequeños lo hace perfectamente adecuados para productos finales de mercado masivo con tamaño estricto y requisitos de costos, tal como el prototipo para este proyecto. Además de tener una precisión en posición horizontal de alrededor de 2.5 m, en velocidad 0,1m/s y en orientación 0.5º, valores más que aceptables para un sistema de posicionamiento GPS (Ublox, s.f.).

Figura 10. Módulo GPS GY-NEO6MV2 (Amazon, 2019).

1.4.2. Microcontrolador

Un microcontrolador es un circuito integrado que en su interior contiene una Unidad Central de Procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada/salida y periféricos.

El microcontrolador es un computador dedicado a diversas aplicaciones. En su memoria sólo reside un programa destinado a gobernar una aplicación determinada; sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar, y todos los recursos complementarios disponibles tienen como única finalidad atender sus requerimientos (Novas, 2008).

El microcontrolador se encarga de procesar la información proporcionada por el GPS y prepararla para transmitirla de forma inalámbrica a través del módulo transceptor LoRa al gateway.

- **Criterios para la elección del microcontrolador**

Según Mazidi et al.(2008), para la selección del microcontrolador se deben tomar en cuenta algunos criterios:

- El primer y más importante criterio en la elección del microcontrolador es que debe cumplir con la tarea de forma eficiente y económica. Se debe elegir en base a las necesidades del proyecto. A partir de esto, se debe considerar si se necesita un microcontrolador de 8, 16 o 32 bits para el trabajo de computación. También se debe tomar en cuenta la velocidad, encapsulado,

potencia de consumo, cantidad de RAM y ROM, número de pines I/O, costo por unidad, tamaño, entre otros.

- El segundo criterio en la elección es qué tan fácil es desarrollar productos en torno al él.
- El tercer criterio para elegir un microcontrolador es su disponibilidad inmediata en cantidades necesarias tanto ahora como en el futuro.

Tomando en cuenta estos criterios se ha seleccionado el microcontrolador ATMEGA328P montado en una placa de desarrollo de Arduino. Con este microcontrolador se satisfacen los requisitos del diseño en cuanto a memoria, entradas y salidas, tamaño, e interfaz SPI (Serial Peripheral Interface) y comunicación serial UART (Universal Asynchronous Receiver-Transmitter) para la interacción con el módulo de comunicación inalámbrica y GPS.

- **Arduino Nano basado en Atmega328P**

El Arduino Nano, mostrado en la Figura 11, es una placa pequeña, completa y compatible con placas de prueba basada en ATmega328P (Arduino Nano 3.x). Además, se caracteriza por ser una placa microcontroladora pequeña, fácil de usar y muy flexible. Su tamaño compacto lo hace ser fácil de usar e ideal para proyectos que requieren un factor de forma pequeño, lo que resulta ser bastante beneficioso, puesto que en sí las funcionalidades que posee son iguales a las otras placas programadoras (ECDA, 2020).

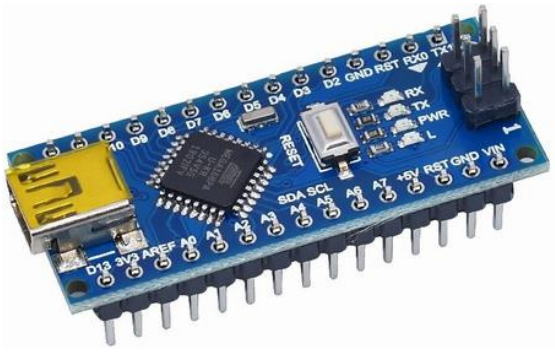


Figura 11. Placa Arduino basado en ATMEGA328P (Mercado Libre, s.f.)

Especificaciones técnicas:

Las especificaciones del Arduino Nano según la página oficial del fabricante se pueden observar en la Tabla 1.

Tabla 1. Especificaciones del Arduino Nano

Microcontrolador	**ATmega328P**
Arquitectura	AVR
Bus de datos	8 bits
Tensión de funcionamiento	5 voltios
Memoria flash	32 KB (2 KB para el bootloader)
SRAM	2 KB
Velocidad de reloj	16 MHz
Pines de entrada analógica	8
EEPROM	1 KB
Corriente CC por pines de E/S	40 mA (pines de E/S)
Voltaje de entrada	7-12V
Pines de E/S digitales	22 (6 de los cuales son PWM)
Salida PWM	6
El consumo de energía	19mA
Tamaño de placa	18x45mm
Peso	7g
Interfaces de comunicación	I2C, SPI, UART

Nota. Adaptado de Arduino Nano, Arduino Oficial Store, s.f.

1.4.3. Módulo transceptor RF LoRa

En el mercado existen una gran variedad de módulos de transmisión y recepción de largo alcance basados en chips fabricados por Semtech, su método especial de modulación LoRa aumenta la distancia de comunicación por lo cual puede enlazarse con uno o más gateway, teniendo un envío de información rápido y alta inmunidad a las interferencias (E22-900M30S User Manual, 2020).

1.4.3.1. Transceptor LoRa RFM95W

Los transceptores RFM95W cuentan con el modem de largo alcance LoRa que proporciona comunicación de espectro extendido de larga distancia y alta inmunidad a las interferencias mientras se minimiza el consumo de corriente, operando en la banda de 868 MHz o 915 MHz; utilizando la técnica de modulación LoRa se puede lograr una sensibilidad de hasta -136 dBm utilizando un cristal de bajo costo. La alta sensibilidad combinada con el amplificador de potencia integrado +20 dBm produce un presupuesto de enlace que lo hace óptimo para cualquier aplicación que requiera alcance o robustez.

Estos son radios de paquete LoRa que tienen una modulación de radio especial; pueden recorrer 2 km en línea de vista con antenas de cable simples, o hasta 20 km con antenas direccionales y ajustes de configuración (HOPERF, 2018).

Especificaciones técnicas:

- Paquete de radio con bibliotecas Arduino listas para usar.
- Banda ICM sin licencia: 868 MHz (Europa) o a 915 MHz (América).
- Antena de cable simple o un punto para el conector de radio uFL o SMA.
- Módulo basado en SX1276 LoRa con interfaz SPI.
- Potencia: De +5 a +20 dBm hasta 100 mW.
- Consumo: Pico de ~100 mA en transmisión de +20 dBm, ~30 mA durante la escucha activa de radio.
- Rango: Aproximadamente 2 kilómetros, dependiendo de obstrucciones, frecuencia, antena y potencia de salida.
- Tasa de bits: 0.293 - 37.5 kbps
- Ancho de banda: 125 - 500 kHz
- Presupuesto de enlace máximo de 164 dB
- Alimentación de 3 – 5 VDC

Figura 12. Módulo de RF LoRa Adafruit RFM95W (Adafruit, s.f.)

En la Figura 12 se puede observar el módulo Adafruit RFM95W, que incorpora el chip RFM95 en una pequeña placa de desarrollo o acondicionamiento de 29×25 mm con regulación de voltaje adecuada para su uso con interfaz SPI por la distribución de pines que ofrece para su montaje en bases para PCB (Geek, 2023).

1.5. LoRaWAN gateway

Una puerta de enlace LoRaWAN de código abierto, permite unir una red inalámbrica LoRa a una red IP a través de Wifi, red Ethernet o una red móvil. La tecnología inalámbrica LoRa permite a los usuarios enviar datos y alcanzar rangos extremadamente largos con velocidades de datos bajas. Dependiendo del fabricante se dispone de varias frecuencias de operación del gateway entre las que se debe elegir de acuerdo a la red LoRa que se está creando. Estos utilizan un reenviador de paquetes Semtech, además la conexión de la estación LoRaWAN es totalmente acorde con el protocolo LoRaWAN, a la vez que disponen de concentradores LoRa que son chips LoRa de banda base de nueva generación para puerta de enlace, los cuales tiene un menor consumo de corriente y mayor cantidad de tráfico (LPS8N LoRaWAN Gateway User Manual, 2022).

En la Figura 13 se pueden observar algunas de las aplicaciones de un gateway LoRaWAN dentro de una red.

Figura 13. LoRaWAN Gateway (Dragino, 2022)

1.5.1. Dragino LPS8N 915 MHz

El LPS8N es una puerta de enlace interior LoRaWAN de código abierto fabricada por Dragino. Permite unir la red inalámbrica LoRa a una red IP.

El LPS8N utiliza el reenviador de paquetes Semtech y la conexión de la estación LoRaWAN, y es totalmente compatible con el protocolo

LoRaWAN. Incluye un concentrador SX1302 LoRaWAN, que proporciona 10 rutas de demodulación paralelas programables.

Esta puerta de enlace tiene bandas de frecuencia LoRaWAN estándar preconfiguradas para usar en diferentes países. El usuario también puede personalizar las bandas de frecuencia para usar en su propia red LoRa.

Este dispositivo puede emplearse en aplicaciones de Medición inteligente, Ciudades inteligentes (transporte), Agricultura inteligente, Domótica, Logística y Gestión de la Cadena de Suministro (LPS8N LoRaWAN Gateway User Manual, 2022). En la Figura 14 se muestra la arquitectura en la que puede trabajar esta puerta de enlace.

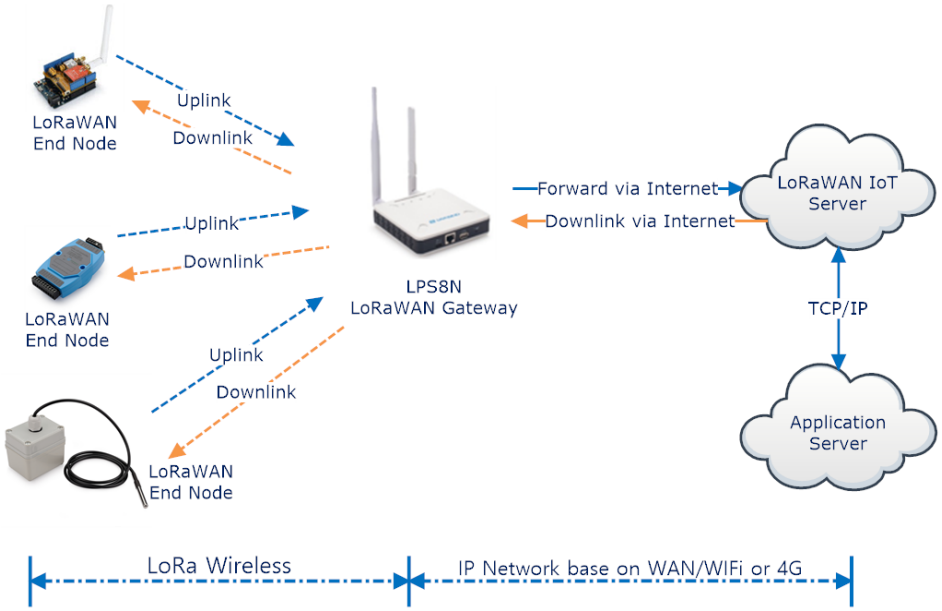


Figura 14. LPS8N en una red LoRaWAN IoT (Dragino, 2022)

Especificaciones técnicas

Entre las especificaciones técnicas del gateway y características más importantes, dadas por Dragino, se tienen las siguientes:

- Posee un procesador 400Mhz ar9331, RAM de 64MB y Flash de 16MB.
- Interfaces: Puerto RJ45 de 10M/100M, Wi-Fi: 802.11 b/g/n, conector USB 2.0, Concentrador LoRa SX1302 + 2 SX1250
- Sensibilidad LoRa de hasta -140 dBm y una potencia máxima de transmisión de 27dBm.

- Emula 49 demoduladores LoRa y 1 demodulador (G)FSK.
- 10 rutas de demodulación paralelas programables (10 canales).
- Sistema OpenWrt de código abierto.
- Administración por Web GUI, SSH a través de WAN o WiFi.
- Admite el reenviador de paquetes de Semtech
- Admite estación básica LoRaWAN.
- Bandas LoRa disponible 868/915 MHz
- Conexión celular 3G/4G opcional

En la Figura 15 se muestra la estructura del hardware que compone el sistema de la puerta de enlace de acuerdo a las especificaciones técnicas antes mencionadas.

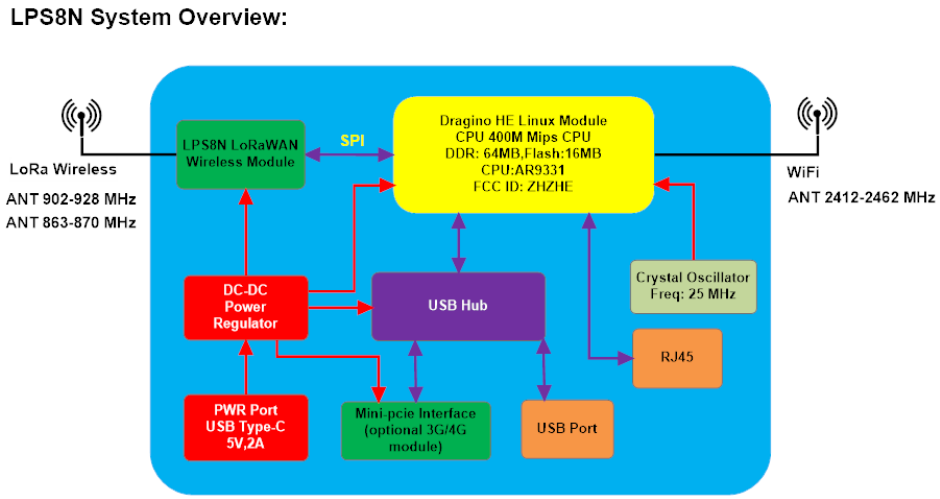


Figura 15. Estructura del hardware del Sistema LPS8N (Dragino, 2022)

1.6. Servicio basado en la nube

El sistema de servicio basado en la nube es el encargado de analizar y procesar la información obtenida para que sea visualizada por el usuario final, este debe estar conectado a una fuente de datos que en este caso es el gateway LoRaWAN, para luego procesarla. Algunas de las plataformas más empleadas son: ThingSpeak y The Things Network (TTN).

1.6.1. The Things Network

Es un ecosistema colaborativo global de IoT desarrollado por The Things Industries, que crea redes, dispositivos y soluciones utilizando LoRaWAN. Cuenta con Things Stack como servidor de red LoRaWAN que es el componente crítico para cualquier solución LoRaWAN. Este es muy utilizado por empresas y desarrolladores de todo el mundo, ya que gestiona de forma segura aplicaciones, dispositivos finales y puertas de enlace (The Things Network, s.f.).

1.6.2. Servidor ThingSpeak

ThingSpeak™ es un servicio de plataforma de análisis de IoT que permite agregar fuentes de datos, visualizar y analizar flujos de datos en vivo en la nube. ThingSpeak proporciona visualizaciones instantáneas de los datos publicados por sus dispositivos, además de contar con la capacidad de ejecutar código en MATLAB para realizar análisis y procesar datos a medida que ingresan, tal como puede visualizarse en la Figura 16. ThingSpeak se usa a menudo para la creación de prototipos y la prueba de concepto de sistemas IoT que requieren análisis (ThingSpeak, 2019).

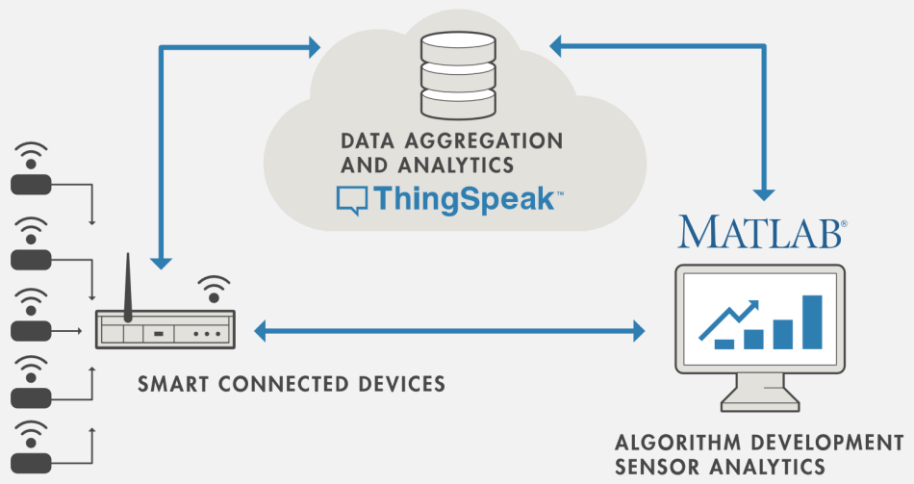


Figura 16. Análisis de datos en el servidor ThingSpeak (ThinSpeak, 2019)

Ambas plataformas incluyen características que son de utilidad para el prototipo de sistema propuesto por la compatibilidad con LoRaWAN, puesto que se pueden interconectar a través de integraciones para que trabajen en conjunto.

1.7. Sistema de visualización

El sistema de visualización se compone por un Raspberry Pi y una pantalla informativa, tal como se ve en la Figura 17. La Raspberry Pi se enlazará a una aplicación web obteniendo los datos previamente analizado y listo para ser mostrado en la pantalla para los usuarios de las unidades de TUC. Debido a las grandes cantidades de información y datos, es importante presentar la información en una pantalla de forma intuitiva proyectando y actualizando los tiempos de arribo de las unidades de transporte.

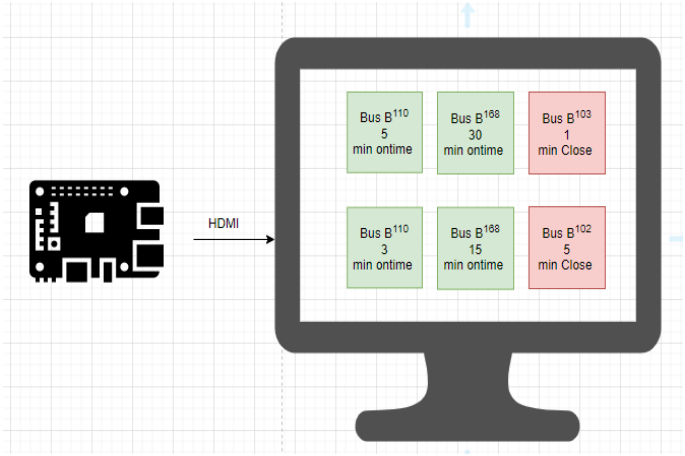


Figura 17. Sistema de visualización. [Autor]

1.7.1. Raspberry Pi 3B+

La Raspberry Pi es una computadora de bajo costo, con un tamaño similar al de una tarjeta de crédito, que se puede conectar a un monitor de computadora o TV, teclado y mouse, entre otros periféricos. Este pequeño dispositivo permite a personas de todas las edades explorar en la informática y aprender a programar en lenguajes como Scratch y Python principalmente. Es capaz de hacer todo lo que esperaría que hiciera una computadora de escritorio o laptop, desde navegar por Internet y reproducir contenido multimedia, hasta hacer hojas de cálculo, procesamiento de textos y jugar. También es una opción muy económica en comparación con otras opciones de hardware similares, lo que la hace ideal para una amplia gama de proyectos de creación digital, educativos y de investigación en universidades y centros de enseñanza en general. Asimismo, gracias a su pequeño tamaño y bajo consumo de energía, es ideal para proyectos que requieren de un dispositivo

compacto y eficiente, como la monitorización de sensores o la recolección y visualización de datos en tiempo real (Raspberry Pi, s.f.)

Estas microcomputadoras son la base del sistema de visualización, ya que en ellas se ejecuta la aplicación en Python desde la cual se pueden observar los tiempos de predicción en una interfaz animada. La Raspberry Pi debe estar ubicada en las estaciones acompañada de una pantalla conectada a través de HDMI para que los usuarios puedan tener la información de los buses que están por llegar. El código que ejecuta la Raspberry Pi conectada a internet descarga los tiempos de predicción procesados en Matlab ThingSpeak por medio de peticiones HTTP y realizar las animaciones de llegada y disminución de los tiempos para que sea más interactivo.

Figura 18. Raspberry Pi 3 Model B+ (Raspberry Pi, s.f.)

La Raspberry Pi 3 Model B + (ver Figura 18) es el último producto de la gama Raspberry Pi 3, con un procesador quad-core de 64 bits con 1,4 GHz, red inalámbrica de banda dual 2.4/5 GHz, Bluetooth 4.2 / BLE, Ethernet más rápido y capacidad PoE a través de un PoE HAT separado (Raspberry Pi, s.f.)

1.8. Algoritmo de predicción de llegada de unidades TUC

En el estudio Development of Prediction Schemes for Real-Time Bus Arrival Information realizado por Lautos (2013), se describe que, para hacer una predicción de la llegada del próximo autobús a una estación específica de la red, se necesita la ubicación del vehículo que se aproxima. Debido a que existen múltiples unidades de TUC, tener la capacidad de monitorear la ubicación en tiempo real de varios autobuses por medio de los nodos LoRa-

GPS nos proporcionará información valiosa sobre la distribución del tiempo de viaje a lo largo de la ruta. Teniendo en cuenta la heterogeneidad del tráfico, pistas y el tiempo de viaje en las rutas, el uso de información en tiempo real de las trayectorias de los autobuses anteriores podría ofrecer una perspectiva más precisa.

Es oportuno resaltar que la idea original del algoritmo RTTT, desarrollado por Lautos (2013), fue modificado para funcionar en condiciones reales de tráfico, pistas, estaciones y tiempos de espera del sistema de transporte público de la ciudad de Managua.

El sistema requiere como entrada el nombre de la estación y la hora exacta, en que la unidad de TUC arriba, además el esquema de pronóstico se basa en la suposición de que los autobuses salen de la terminal (o estación de referencia en el ambiente de prueba) acorde al itinerario designado por las autoridades de transporte público de Managua.
Dado que los itinerarios no están disponible al público se asignarán predicciones por defecto basadas en mediciones reales realizadas previamente en el ambiente de prueba.

A continuación, se presenta la formulación matemática del esquema de predicción:

- Encontrar el número de viajes denotado por la variable **bustrips.** (Además, se utiliza para identificar el bus al cual se realiza la predicción).
- Encontrar la última estación visitada en el viaje de la unidad de TUC.
- Se denota $\boldsymbol{PIksa}_{\boldsymbol{alfa}_{\mathbf{bustrips},\boldsymbol{stopx}}}$ como un arreglo de celdas donde cada celda $\boldsymbol{PIksX}$ contiene los tiempos reales en los que el bus **bustrips** arribó a la estación $\boldsymbol{stopx}$, donde $\boldsymbol{X} \in \boldsymbol{l}$ y $\boldsymbol{l}$ es el conjunto de estaciones de una ruta específica, por ejemplo, la **110**.
- Se busca obtener un vector de tiempo de predicciones, $\boldsymbol{vectkpsAsB}$, de dimensión **Nstop**, siendo **Nstop** el número de estaciones que aún deben ser visitadas y que requieren una predicción. El resto de las estaciones que ya han sido visitadas tomarán una predicción por defecto según el itinerario censado, que serán actualizada cuando una nueva unidad de TUC llegue a la estación

de referencia. Es importante mencionar que cada componente del vector $\boldsymbol{vectkpsAsB}$ está determinado por $\boldsymbol{tkpsAsB}^{\boldsymbol{t,stopA\rightarrow stopB}}_{\mathbf{bustrips}}$.

➢ Se denota $\boldsymbol{tkpsAsB}^{\boldsymbol{t,stopA\rightarrow stopB}}_{\mathbf{bustrips}}$ como el tiempo de viaje promedio ponderado de una cantidad de autobuses precedentes, definida por la variable **bustrips** para el segmento definido entre la estación A y la estación B.

Para realizar las predicciones se mostrará la forma de ponderar los tiempos de viaje de los autobuses anteriores.

Se utiliza un método matemático simple para ponderar, el cual consiste en ponerle peso que corresponde al inverso del intervalo de tiempo entre el autobús precedente y el bustrips, por lo tanto $tkpsAsB^{t,stopA\rightarrow stopB}_{\text{bustrips}}$ se puede calcular con Ecuación 1:

$$\boldsymbol{tkpsAsB}^{\boldsymbol{t,stopA\rightarrow stopB}}_{\mathbf{bustrips}} = \sum_{j=1}^{\text{bustrips}-1} \frac{\frac{1}{PIksA_{\text{bustrips}} - PIksA_{\text{bustrips}-j}}}{\Gamma} * \left(PIksB_j - PIksA_j\right) \quad \textbf{(1)}$$

Donde, Γ es la suma del peso de cada autobús anterior y bustrips el número de autobuses anteriores utilizados para la predicción (ver Ecuación 2).

$$\boldsymbol{\Gamma} = \sum_{j=1}^{\text{bustrips}-1} \frac{1}{PIksA_{\text{bustrips}} - PIksA_{\text{bustrips}-j}} \quad \textbf{(2)}$$

Por lo tanto, el esquema de predicción basado en RTTT (Real-Time Travel Time Method) consta de un proceso en el que se genera el tiempo de predicción de la próxima llegada de la unidad de TUC a las siguientes estaciones del recorrido.

En general, el último autobús anterior al autobús actual contribuirá en mayor medida al tiempo de viaje promedio ponderado que aquellos encontrados más adelante en la ruta.

CAPÍTULO II. ANÁLISIS Y PRESENTACIÓN DE RESULTADOS

2.1. Diseño metodológico

La investigación para llevar a cabo este trabajo monográfico fue de carácter exploratorio, ya que se analizó información específica bajo condiciones del campo de prueba en la ciudad de Managua sin ningún antecedente del uso de la tecnología propuesta. Se exploraron factores críticos que afectan directamente la comunicación del sistema, tales como: la cobertura de los módulos LoRa, el efecto de la movilidad de las unidades TUC, la infraestructura de la ciudad y demás fenómenos físicos que pueden deteriorar la comunicación inalámbrica. A la vez, fue de tipo aplicada, ya que está dirigida a la solución de un problema como lo es la desinformación de los usuarios del transporte público en cuanto a la hora de llegada de las unidades de transporte o la ubicación de estas en tiempo real. El presente proyecto se llevó a cabo en seis etapas fundamentales que se detallan a continuación:

En la primera fase se recolectó y analizó la información disponible acerca de los componentes en sus hojas de datos y de esta manera se diseñaron los nodos LoRa-GPS e interconexión de toda la red en un diagrama general, para posteriormente, centrarse en la funcionalidad de cada subsistema.

En la segunda fase se desarrollan los nodos LoRa-GPS basados en el diseño inicial, adaptable a las unidades de transporte. Se fabrican las tarjetas de circuito impreso las cuales se alojarán en una caja diseñada para brindar protección y soporte a los componentes electrónicos.

En la tercera fase se implementó la red LoRaWAN con los nodos LoRa-GPS basados en los módulos GPS, para la ubicación de cada nodo, y LoRa RFM95W para la comunicación inalámbrica con la puerta de enlace Dragino LPS8N. Además, se estableció la comunicación del gateway con la red de las cosas (TTN), que a su vez se conectó mediante una integración con ThingSpeak para el procesamiento de los datos.

En la cuarta fase se implementó el algoritmo de predicción basado en RTTT en Matlab Analysis de ThingSpeak. Este servidor en la nube es capaz de

obtener los datos de TTN para procesarlos en Matlab, obteniendo así las predicciones para su posterior visualización.

En la quinta fase se implementó una estación de visualización formada por una interfaz de visualización, desarrollada en Python y ejecutada en una Raspberry Pi 3B+, en la que se muestra las predicciones de arribo de las unidades de TUC e información relevante para el usuario.

En la sexta y última fase se realizó la integración de todos los subsistemas y las pruebas finales del sistema en un ambiente de pruebas real, ubicando el nodo en un vehículo de pruebas. En esta fase se recopila información que evidencie el correcto funcionamiento del sistema, tomando datos de la interfaz gráfica, y del mismo servidor de ThingSpeak donde se obtienen gráficos para medir la eficiencia de las predicciones.

2.2. Desarrollo

2.2.1. Requerimientos del sistema

En el desarrollo de un prototipo final es necesario tener en cuenta requerimientos técnicos y funcionales desde el diseño de cada componente hasta las funciones más específicas del sistema con el fin de obtener resultados satisfactorios.

2.2.1.1. Técnicos

A continuación, se detallan los requerimientos técnicos para el desarrollo de un ambiente de pruebas y diseño del sistema que se adapte al entorno en el cual se está trabajando y sea funcional.

- **Autonomía:** La autonomía o tiempo en el que los dispositivos finales pueden mantenerse activos con una fuente de alimentación independiente es un aspecto muy importante para la implementación de nodos en las unidades de transporte. La autonomía va a depender de dos factores: consumo total de los nodos y de la capacidad de la fuente de alimentación.

En Managua los buses de TUC comienzan a circular a las 4:30 de la mañana y culminan su servicio a las 9:00 de la noche, por lo que se necesita una autonomía de alrededor de 16.5 horas con la cual se pueda llegar al final de la

jornada teniendo el nodo siempre activo, tomando en cuenta que los nodos solamente transmiten cuando están en una de las estaciones y el resto del tiempo están en reposo. Debido a lo mencionado anteriormente, el nodo debe cumplir con los siguientes requerimientos:

- El consumo energético del nodo LoRa-GPS debe ser lo más bajo posible para asegurarse de tener una autonomía de al menos 16.5 horas sin perder de vista un tamaño compacto.

- Debe ser un tipo de nodo LoRa que se ajuste a las necesidades de consumo energético como lo es un nodo de Clase A.

- **Precisión de ubicación:** En este sistema la precisión de la tecnología de ubicación o localización es uno de los factores cruciales para el correcto funcionamiento de los sistemas de ubicación y para este sistema en específico. La precisión de sistemas de posicionamiento como GPS, suele ser bastante fiable, con un error horizontal de 2.5 metros aproximadamente, aunque en zonas urbanas puede verse afectada por la densidad de edificios y la falta de contacto directo con el dispositivo receptor. Sin embargo, la precisión ofrecida por GPS es más que aceptable para la implementación en un nodo de ubicación, además de su facilidad de uso y accesibilidad. Tomando en cuenta que la longitud promedio de las estaciones de buses de TUC en Managua es de 20 a 30 metros, se necesita que se pueda ubicar con precisión aceptable un bus en este rango para identificar la estación a la que ha arribado.

- **Tecnología de comunicación:** En la red LoRaWAN a implementar en este sistema se asegura comunicación inalámbrica a media y larga distancia gracias a la tecnología LoRa, que ya fue abordada en la sección 1.2, en una banda de libre operación.

- Se hará uso de la banda 902-928 MHz para la comunicación entre los nodos y la puerta de enlace, por lo cual se deben seleccionar los componentes adecuados y diseñar los nodos para operar en esta banda de frecuencia.

- Idealmente, el dispositivo final o al menos las antenas deben ubicarse en una parte externa del bus para evitar problemas de comunicación LoRa y principalmente tener una buena recepción GPS.

2.2.1.2. Funcionalidades

El sistema debe ser fácil de usar tanto para el usuario final como para los conductores de los buses en lo referido a las opciones que se ofrecen, interacciones, información mostrada y carga de los dispositivos finales.

- **Interacción con el usuario final (pasajeros)**

La interfaz gráfica es el medio con el cual se interactúa con los usuarios, en esta se deben mostrar los tiempos de arribo en las estaciones teniendo en cuenta que debe ser fácil de entender e intuitiva para todos los usuarios. Además, se mostrará información relevante tal como la ubicación actual del bus que viene de camino (última estación que visitó, trayecto en el que está) y el tiempo en el que se estima vendrá a la estación en que se ubica.

- **Interacción con el conductor (proveedor del servicio)**

Los nodos LoRa-GPS, como un dispositivo final que funciona con una fuente de alimentación independiente, ofrece la posibilidad de recargarse al final de una jornada o cuando se necesite por medio de un cable micro-USB a USB tipo A.

Además, se tiene una segunda interacción con el conductor que consta de un botón de emergencia que debe presionarse en caso de que la unidad de transporte sufra algún problema técnico o situación anormal que ocasione que el bus quede inoperativo.

También se dispone de indicadores LED que informarán si el dispositivo está activo (encendido) y si encuentra en estado de transmisión cuando arribe a una de las estaciones.

2.2.2. Ambiente de pruebas LoRaWAN

Con el objetivo de comprobar el correcto funcionamiento del prototipo desarrollado para el sistema propuesto, se debe implementar un ambiente de pruebas en el cual se pueda evaluar su desempeño en un entorno real.

En el ambiente de pruebas se incluye detalles sobre la arquitectura de la red LoRaWAN, la configuración de sus componentes y servidores; y la implementación del algoritmo de predicción.

2.2.2.1. Arquitectura de la red LoRaWAN

La arquitectura de red LoRaWAN implementada en el desarrollo de este prototipo de sistema se puede visualizar en la Figura 19.

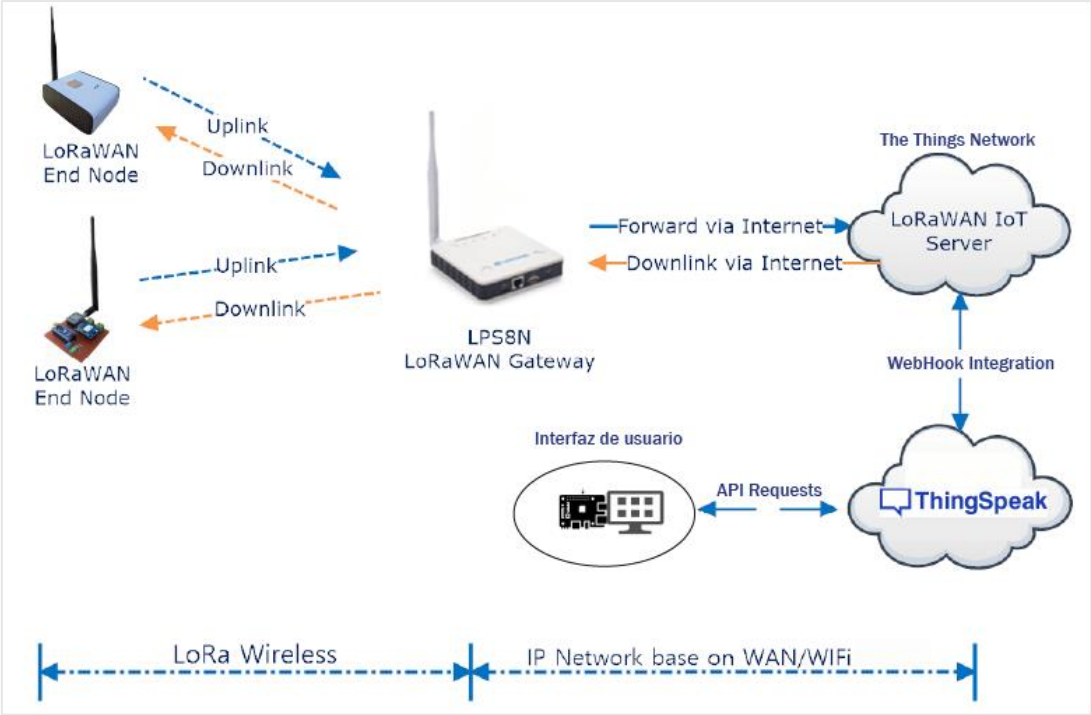


Figura 19. Arquitectura LoRaWAN [Modificado de "LPS8N- LoRaWAN Gateway User Manual" por Autor, 2022, "What is the LPS8N"].

Esta arquitectura está formada por cinco elementos principales, los cuales son:

- **Nodos:** Para este prototipo se tienen dos nodos funcionales que se desarrollaron. Estos, gracias al módulo GPS que integran, son capaces de brindar la latitud y longitud del bus que lo equipa, y una vez el microcontrolador calcula que ha entrado en ubicaciones establecidas se procede a transmitir por medio del módulo transceptor LoRa hacia la puerta de enlace.

- **Puerta de enlace:** El gateway LPS8N es el encargado de recibir los paquetes enviados desde los nodos por LoRa y subirlos a internet, específicamente al servidor LoRaWAN The Things Network.

- **Servidor LoRaWAN IoT:** The Things Network es el servidor orientado por el manual de usuario del gateway Dragino que permite conectar dispositivos IoT a la nube con una orientación a LoRaWAN. Este ofrece la posibilidad de decodificar la carga útil del paquete LoRa por medio de un formato de Uplink y

enviarlo a otro servidor para su procesamiento en Matlab por medio de una Integración - Webhook de ThingSpeak que ofrece.

- **Servidor ThingSpeak:** Es uno de los servidores de análisis IoT gratuitos con más prestaciones. Una vez obtenidos los datos de TTN se encarga de procesarlos en Matlab, y posteriormente ofrece la facilidad de poder consumir estos datos por medio de API de lectura.

- **Interfaz de usuario o estación de visualización:** Su finalidad es mostrar a los usuarios de forma amigable las predicciones generadas por el algoritmo implementado en Matlab a partir de la ubicación de las unidades de transporte. La Raspberry Pi se encarga de obtener las predicciones de ThingSpeak por medio de solicitudes HTTP y mostrarlas en una interfaz desarrollada en Python.

2.2.3. Configuración de elementos del ambiente de pruebas

Para poder implementar la arquitectura LoRaWAN primeramente se deben configurar los equipos y el servidor a utilizar.

2.2.3.1. Configuración de la puerta de enlace Dragino

Para realizar la configuración del gateway LPS8N inicialmente es necesario estar conectado al punto de acceso WiFi que genera el mismo dispositivo al energizarse, y ya estando conectado, después de utilizar la clave de acceso por defecto que aparece en el manual de usuario, se debe navegar a la dirección IP 10.131.1.1 de la interfaz de configuración y acceder con las credenciales por defecto, como puede visualizarse en la Figura 20.

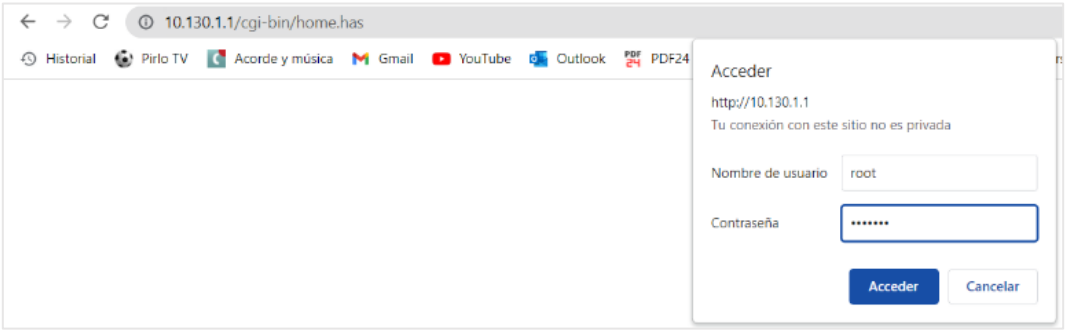


Figura 20. Acceso a la Web UI de configuración Dragino [Autor].

Una vez se tiene acceso hay una gran variedad de configuraciones divididas en varios menús en forma de listas desplegables, de los cuales se deben configurar los apartados de LoRa, LoRaWAN y Network (WiFi). En la Figura 21 se puede visualizar la vista general del sistema en la Web UI de Dragino.

Figura 21. Vista general del sistema Dragino [Autor].

La primera configuración por hacer respecta a la red LoRa; en este caso se debe establecer la banda y sub-banda de frecuencia en la cual se va a trabajar. A como se detalló anteriormente, para el diseño de esta red se ha seleccionado la banda ICM de 915 MHz que va de 902 a los 928 MHz, que en Nicaragua es libre para aficionados, por ello se selecciona la banda US915 y la segunda sub-banda, de 903.9 a 905.3 MHz, que es recomendada por el sistema, esta configuración se presenta en la Figura 22.

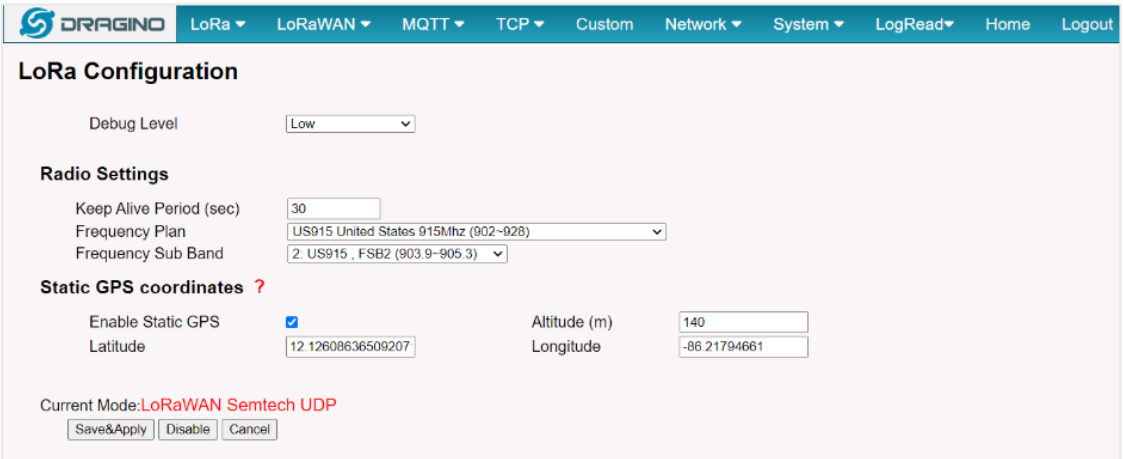


Figura 22. Configuración de frecuencia de la red LoRa [Autor].

Ya configurada la red local LoRa es momento de configurar la red LoRaWAN para conectarla a internet, dirigiéndose al desplegable LoRaWAN y seleccionando el proveedor de servicio The Things Network y la dirección de servidor más cercana a nuestra ubicación que es la correspondiente a Norte América, como puede observarse en la Figura 23. Es importante recordar esta última configuración y el código Gateway EUI para su posterior registro en el servidor de la red de las cosas TTN.

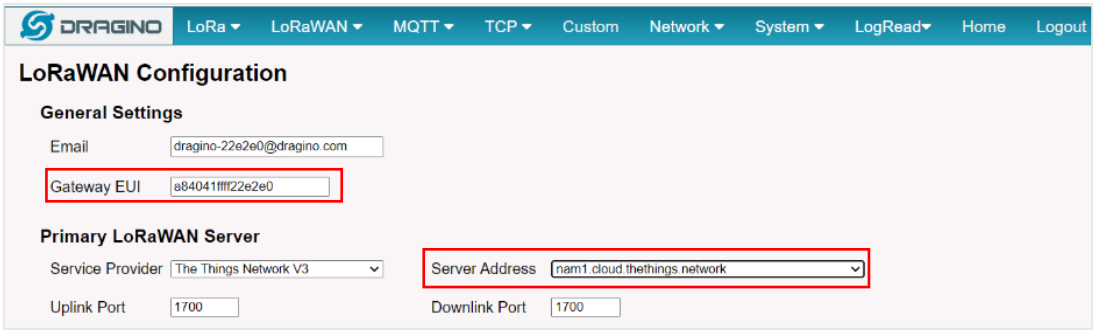


Figura 23. Configuración LoRaWAN [Autor].

Por último, para que el gateway pueda cumplir su función de puerta de enlace entre una red local LoRa e internet debe estar conectado a una red con acceso a internet (ser un cliente WiFi WAN). Para esta configuración se despliega la pestaña Network → WiFi y se habilita el cliente WiFi para conectarlo a una red con acceso a internet, a como se visualiza en la Figura 24.

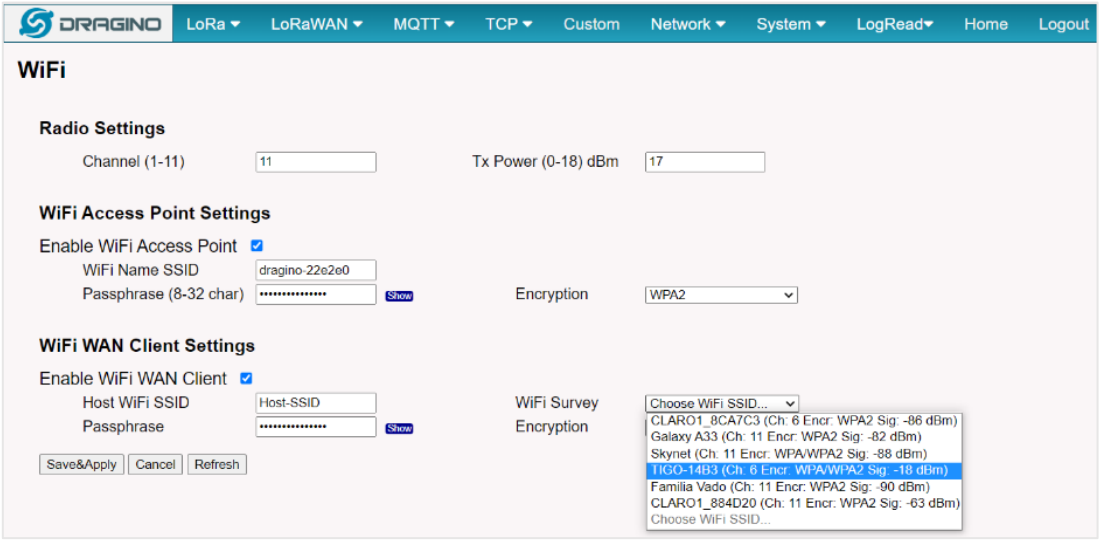


Figura 24. Habilitación para ser cliente WiFi conectado a internet [Autor].

Al hacer esta configuración el gateway está habilitado para cumplir con su función de forma adecuada en este ambiente de pruebas. En la Figura 25 se muestra la vista general después de realizadas las configuraciones necesarias.

Figura 25. Configuración del gateway Dragino LPS8N completada [Autor].

2.2.3.2. Registro de la puerta de enlace en la red de las cosas (TTN)

Con el gateway ya configurado de acuerdo a los requerimientos, se procede a hacer el registro en The Things Network, servidor que se eligió como preveedor del servicio LoRaWAN. Para esto es necesario el parámetro de Gateway EUI que es único para cada puerta de enlace LoRaWAN, y también se seleccionará el mismo plan de frecuencia y sub-banda configurada en el gateway mostrado anteriormente en la Figura 22. En la Figura 26 se muestra el registro del gateway en TTN.

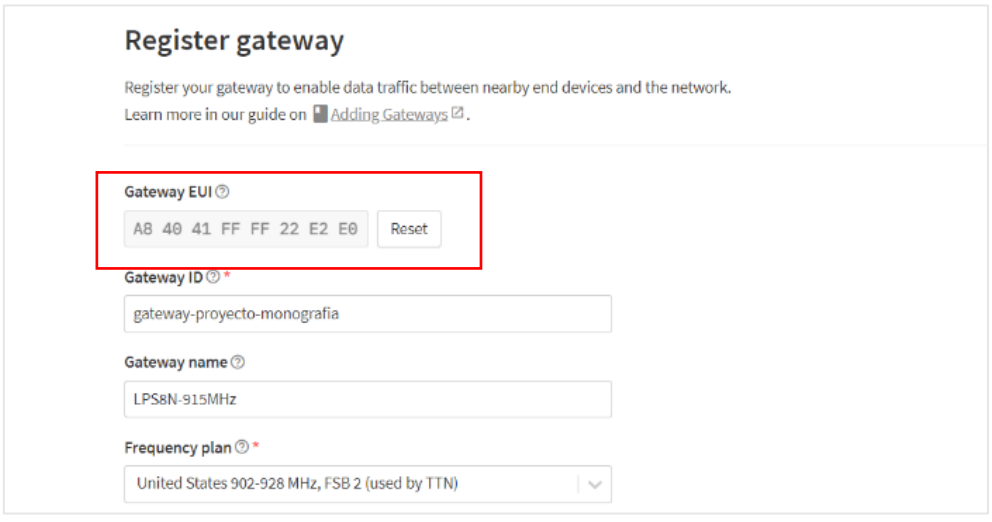


Figura 26. Registro del Gateway en TTN [Autor].

Con la culminación del registro se puede observar en TTN que el gateway ya está conectado a internet y estableciendo conexión con el servidor, a como verse en el Live data de la Figura 27.

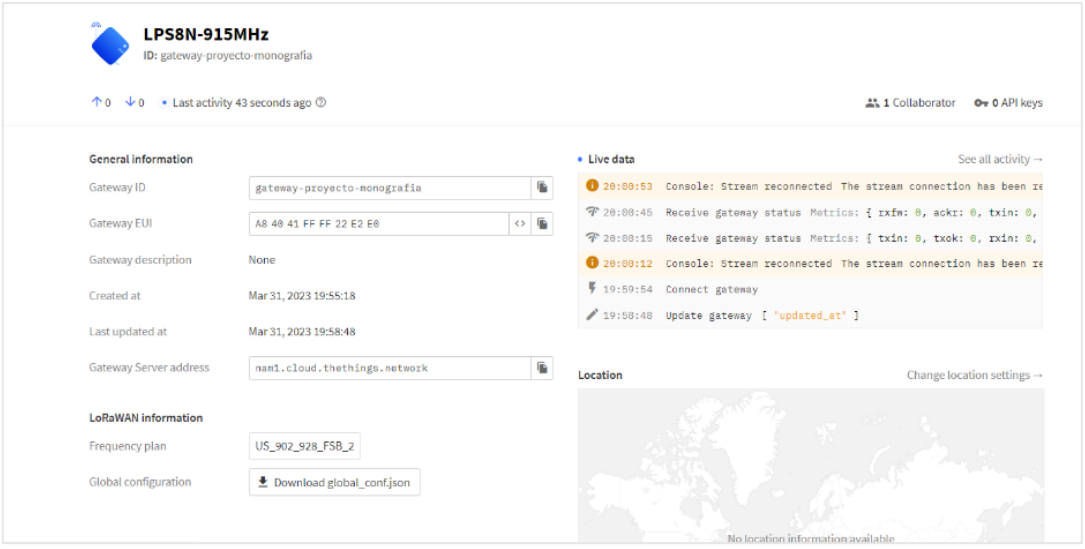


Figura 27. Registro del Gateway en TTN finalizado [Autor].

2.2.3.3. Dispositivos finales de la red en TTN

Una vez asegurado que la red LoRa está conectada a internet en el servidor TTN, se procede con la creación de una aplicación en la cual se agregarán los dispositivos finales de la red y se integrarán los servicios de ThingSpeak. En la Figura 28 se observa la aplicación creada con el fin agregar nodos finales y decodificar la carga útil de los paquetes provenientes de ellos.

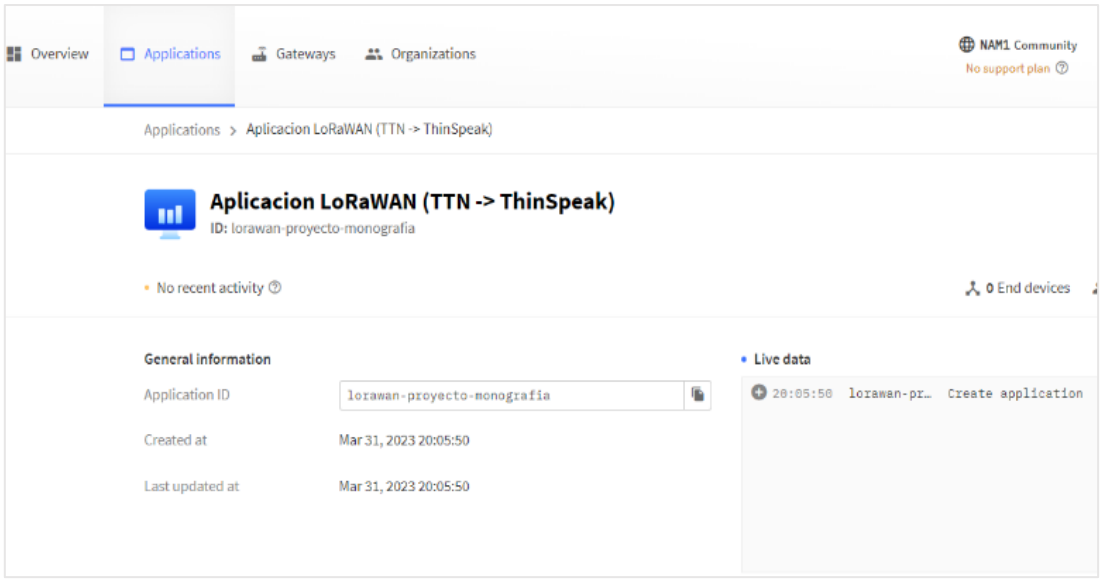


Figura 28. Creación de aplicación en TTN para el sistema [Autor].

Ya creada la aplicación se agregan los nodos a utilizar en el prototipo introduciendo los datos manualmente, al ser un nodo no comercial, y seleccionando el plan de frecuencia configurado para el gateway, como puede visualizarse en la Figura 29.

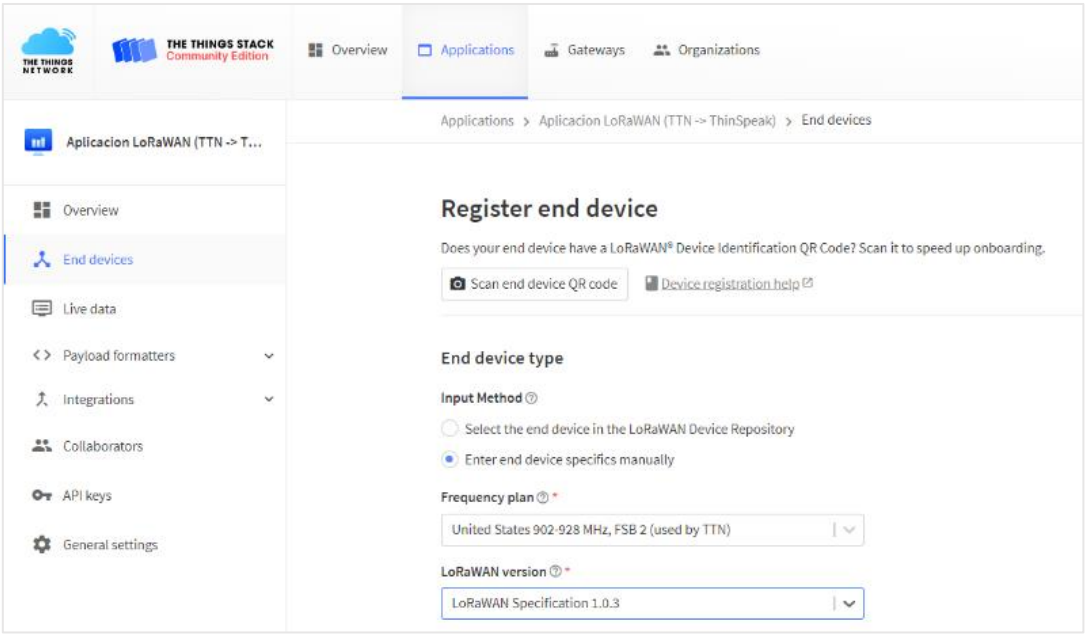


Figura 29. Registro de dispositivos finales [Autor].

Además, se selecciona la clase del nodo, que debe ser tipo A, y el método de activación ABP, debido a que es la mejor opción para el prototipo al considerar el corto tiempo que necesita para ser aceptada su solicitud de unión a la red, además, se habilita la función de reset del contador de frames para que el nodo pueda unirse a la red sin problema al ingresar nuevamente al rango del gateway, teniendo así acceso a una de las ventajas del método OTAA sin perder la ventaja de ABP de registrarse en la red solamente una vez. También se identifican las llaves para inicio de sesión e identificador del dispositivo, datos que serán usados en la programación de los nodos.

En la Figura 30 se muestra la configuración de los dispositivos finales al agregarlos en la aplicación en The Things Network

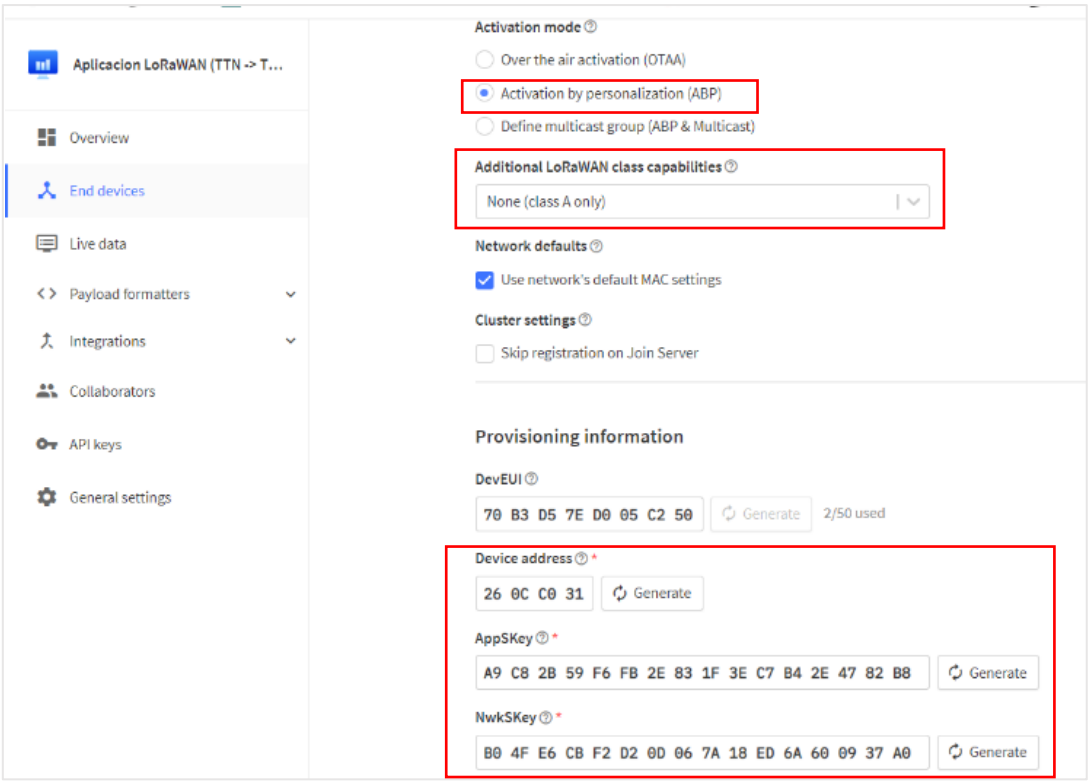


Figura 30. Configuración de los nodos [Autor].

En la Figura 31 se puede notar que a la aplicación se han agregado dos dispositivos finales, que son los que se han fabricado como parte del prototipo para este ambiente de pruebas.

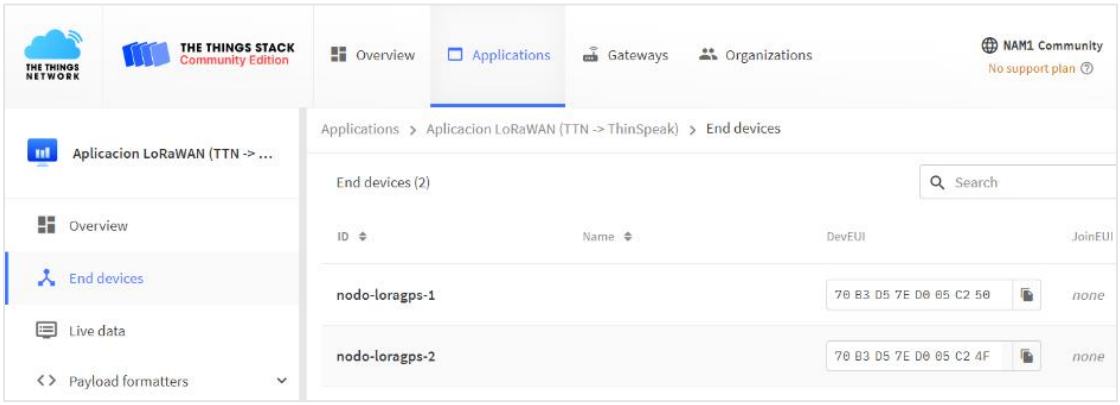


Figura 31. Dispositivos finales agregados [Autor].

2.2.3.4. Formato de carga útil de enlace ascendente predeterminado.

El “formateador de carga útil de enlace ascendente predeterminado” es una herramienta que permite dar formato a los datos recibidos desde un nodo

LoRa-GPS en sentido ascendente y se configura a través de una pestaña de donde se pueden definir ajustes personalizados mediante JavaScript.

Por lo tanto, se aprovechó el uso del formateador de JavaScript personalizado para escribir un código que permitiera asignar a la cadena decodificada propiedades específicas al objeto resultante. Esto facilitó la identificación de las cadenas que pertenecen a las estaciones correspondientes al trayecto de "ida" y al trayecto de "regreso".

En la Figura 32 se muestra el diagrama de flujo del proceso de decodificación y asignación de cadenas a los campos field1 y field2. El último carácter de la cadena se utiliza como identificador: "1" para el trayecto de ida y "2" para el trayecto de “regreso” respectivamente.

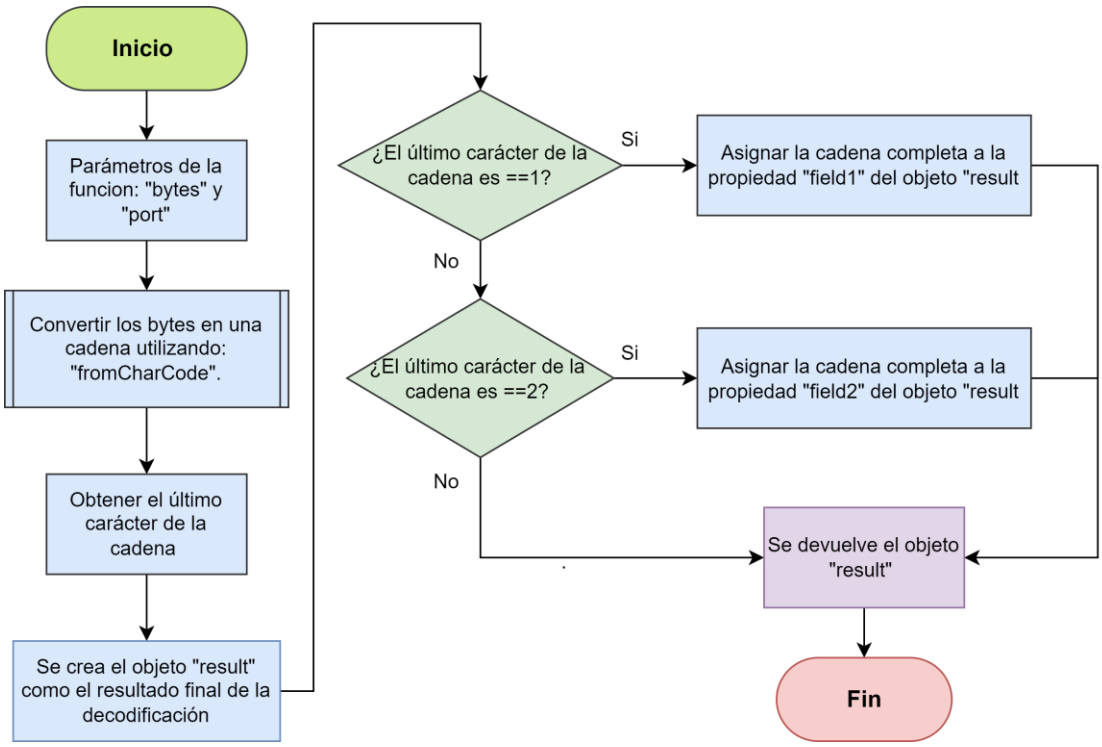


Figura 32. Flujograma de decodificación de carga útil (Uplink) [Autor].

2.2.3.5. Integración de ThingSpeak en TTN

Una vez lista la aplicación y los nodos creados se podrán decodificar los paquetes enviados desde los nodos y visualizarlos en los datos entrantes de TTN, sin embargo, aún se necesita que estos datos de ubicación lleguen a ThingSpeak para utilizarlos como entradas del algoritmo en su herramienta de

procesamiento MATLAB Analysis. Para esto, TTN ofrece la posibilidad de conectarse con el servidor ThingSpeak de una forma sencilla en el apartado de Integraciones → Webhooks, a como se ilustra en la Figura 33.

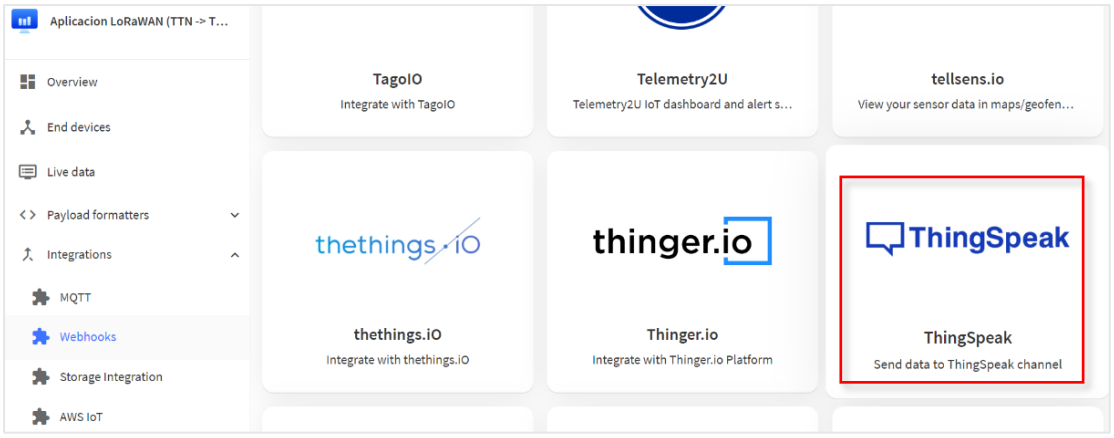


Figura 33. Integración de ThingSpeak en TTN [Autor]

En el siguiente paso, se registra la integración del canal de ThingSpeak introduciendo el ID del canal y su respectiva llave API de escritura; de esta manera se podrán escribir los datos provenientes de los nodos que llegan a TTN en el canal de ThingSpeak para su posterior procesamiento (ver Figura 34).

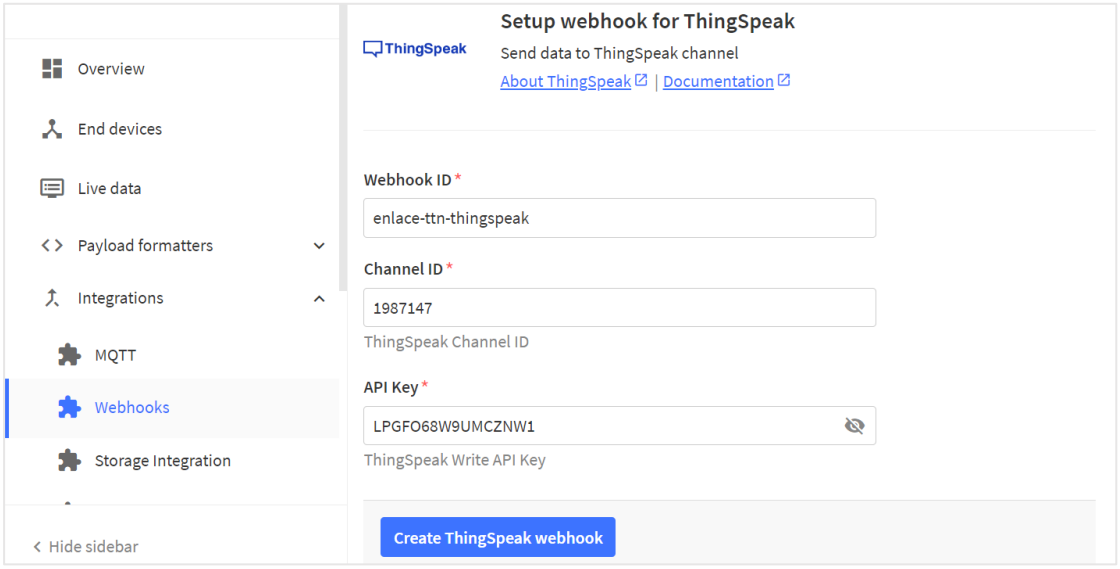


Figura 34. Creación de Webhook ThingSpeak para el canal [Autor].

2.2.4. Diseño e implementación de los nodos LoRa-GPS

Para la construcción de un nodo LoRa-GPS funcional es necesario realizar la programación y diseño electrónico y estructural que cumplan con los requerimientos tratados anteriormente para su correcto funcionamiento dentro del sistema y condiciones de trabajo.

2.2.4.1. Fuente de alimentación

La fuente de alimentación en los dos nodos para todo el circuito está basada en una batería 2500 mAh a 3.7V, la cual es elevada a un nivel de 5V a través de una placa de regulación y carga de la batería. Estas baterías permiten operar durante una jornada aproximada de 16.5 horas de actividad y ser recargadas al final del día de ser necesario o mientras está en actividad, de tenerse disponible una fuente de carga con salida de puerto USB. Los datos de consumo del nodo pueden verse en la Tabla 2.

Tabla 2. Consumos del nodo LoRa-GPS

Dispositivo	Consumo en reposo	Consumo en transmisión(máximo)
Nodo LoRa-GPS	80 mA – 90 mA	185 mA

Conociendo estos consumos, la capacidad de la batería empleada, y considerando que el nodo se encuentra la mayor parte del tiempo en reposo, podemos calcular el tiempo aproximado de autonomía de la fuente de alimentación usando la Ecuación 3:

$$\boldsymbol{t_{descarga}} = \frac{Capacidad\ de\ batería\ (mAh)}{Consumo\ del\ dispositivo\ (mA)} \quad \textbf{(3)}$$

Al remplazar los valores de capacidad de batería (mAh) y consumo del dispositivo (mA) en la ecuación mostrada, resulta:

$$\boldsymbol{t_{descarga}} = \frac{2500\ mAh}{90\ mA}$$

$$\boldsymbol{t_{descarga}} = 27.7\ horas$$

Se calcula que el tiempo de descarga es de aproximadamente 28 horas, con lo que se cumple con el requisito de autonomía para una jornada de servicio a tomarse en cuenta para este diseño.

2.2.4.2. Diseño electrónico y fabricación de PCB

El diseño electrónico se realizó a partir de las funcionalidades requeridas para el nodo, abordadas anteriormente. Tomando en cuenta lo anterior, el diseño incluye módulo GPS y transceptor LoRa, además de la placa Arduino Nano y algunos elementos de señalización lumínica (LED) para el estado del nodo y estado de la comunicación. Para la alimentación del circuito se ha elegido una batería de 3.7 V con su respectivo módulo de carga y suministración de energía de salida a 5V, a como se ha detallado antes, y a partir de la elección de esta fuente de alimentación se han seleccionado módulos que operen a ese nivel de tensión. En la Figura 35 se muestra el circuito electrónico para el nodo LoRa-GPS con todos los componentes utilizados.

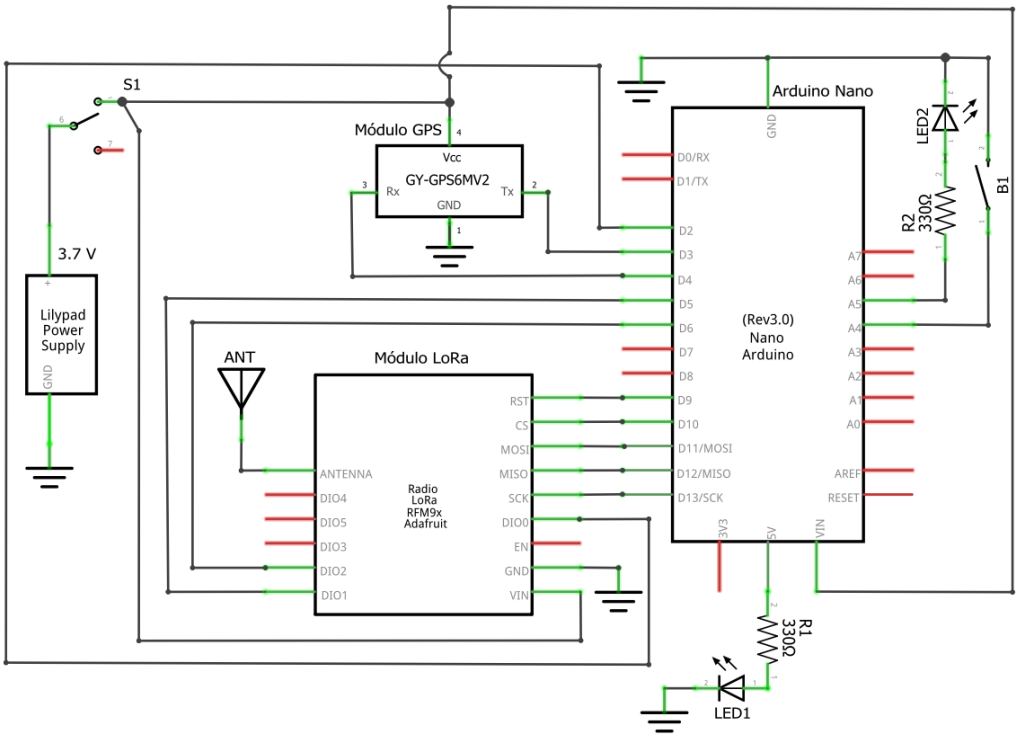


Figura 35. Esquemático electrónico del nodo LoRa-GPS [Fritzing, Autor]

El módulo GPS GY- NEO6MV2 es utilizado en este diseño y su comunicación con el microcontrolador es a través del protocolo serial UART con los pines 3 y 4 (Rx/Tx) del Arduino Nano, teniendo en cuenta que el Rx del GPS debe ir al Tx del Arduino y viceversa. El módulo GPS también se ha empleado ya que se puede alimentar con una tensión de hasta 5 V al poseer un regulador integrado dentro de sí para adecuar a un voltaje de 3.7 V para la correcta operación del chip NEO-6M de Ublox que incluye este módulo; además de

incluir una memoria EEPROM que utiliza el chip para guardar la configuración y para retener los últimos datos de posicionamiento, también un conector U.FL para conectar una antena externa de cerámica que utiliza para recibir la señales de geolocalización. La Tabla 3 muestra las conexiones entre el Arduino Nano y el módulo GPS

Tabla 3. Conexiones entre Arduino Nano y Módulo GPS

Arduino Nano	GY- NEO6MV2
Vcc (5V)	Vcc (5V)
D3 (Rx)	Tx
D4 (Tx)	Rx
GND	GND

De igual manera el módulo LoRa seleccionado alberga un regulador de voltaje de 3.3 V y un cambiador de nivel que puede manejar potencia y lógica de 3 a 5 V para que pueda ser utilizado con dispositivos de 3 V o 5 V. Este módulo se comunica por medio de la interfaz SPI con el Arduino y emplea otros cuatro pines de control digital que usa la librería LMIC, la cual es utilizada para programar el nodo con el protocolo para comunicarse en una red LoRaWAN. El módulo RFM95W basado en el chip de Semtech SX1276 cuenta con un conector SMA Edge-Mount para la conexión de una antena de 915 MHz para su comunicación con la puerta de enlace LoRaWAN. En la Tabla 4 se pueden observar las conexiones entre el Arduino Nano y el transceptor LoRa.

Tabla 4. Conexiones entre Arduino Nano y Módulo Adafruit RFM95W

Arduino Nano	Adafruit RFM95W
Vcc (5V)	Vcc (5V)
D9	RST
D10	CS
D11	MOSI
D12	MISO
D13	SCK
D2, D5, D6	G0, G1, G2
GND	GND

El diseño de la PCB se realizó tomando en cuenta que debe ser un nodo de tamaño compacto y buscando que las terminales queden distribuidas de forma

estratégica para el posterior diseño de la caja del nodo. El software utilizado para el diseño de la tarjeta electrónica fue Proteus v8.12, en el cual primeramente se realiza el esquemático electrónico y posteriormente se procede a realizar el diseño PCB, distribuyendo los componentes a conveniencia. En la Figura 36 se puede observar el diseño PCB para el nodo, con unas dimensiones de 70 x 80mm.

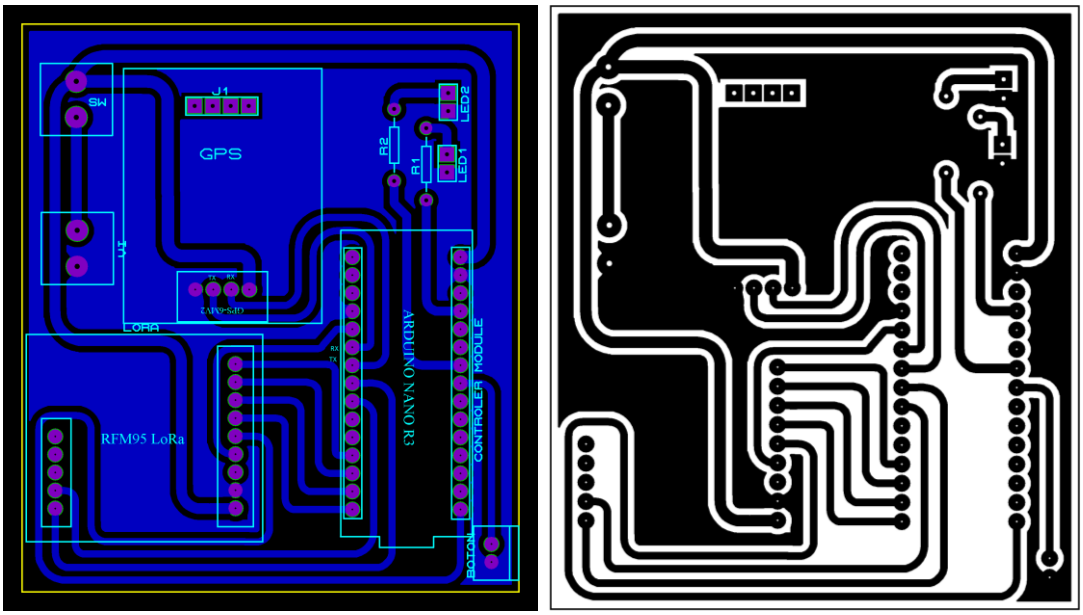


Figura 36. Diseño del circuito impreso [Proteus, Autor]

Luego de terminado el diseño PCB se fabricaron dos prototipos en tarjetas vírgenes de baquelita utilizando la técnica o método del planchado para la transfusión de la tinta de impresión en papel fotográfico a la placa, y la introducción de la placa, con las pistas adheridas, en ácido nítrico hasta que desaparezca el excedente de cobre, solamente quedando el cobre cubierto por la tinta. Una vez terminado esto, se procede a realizar las perforaciones en la placa y colocar los componentes y pines hembra para soldarlos, culminando con las pruebas de continuidad y de funcionalidad para verificar que no haya errores y todas las conexiones estén en perfecto estado.

En la Figura 37 se muestra la tarjeta electrónica del nodo LoRa-GPS con todos los componentes electrónicos para su correcto funcionamiento

Figura 37. Tarjeta PCB del nodo LoRa-GPS [Autor]

2.2.4.3. Diseño estructural para exteriores

Un nodo para aplicaciones como la abordada necesita prestar las condiciones para ser utilizado en un vehículo sin comprometer la integridad de los componentes electrónicos que componen la tarjeta electrónica y que permita que este sea más manejable.

La estructura creada para el nodo es una caja hecha a medida, de modo que quepa la tarjeta y la batería, además de incluirse perforaciones para la visibilidad de componentes como el botón de emergencia, interruptor de apagado y encendido, puerto de carga, LED indicadores, al igual que se asegura la ubicación externa de antenas de GPS y LoRa para tener una mejor recepción y transmisión. En la Figura 38 se muestra una vista interna de la caja con los componentes ubicados.

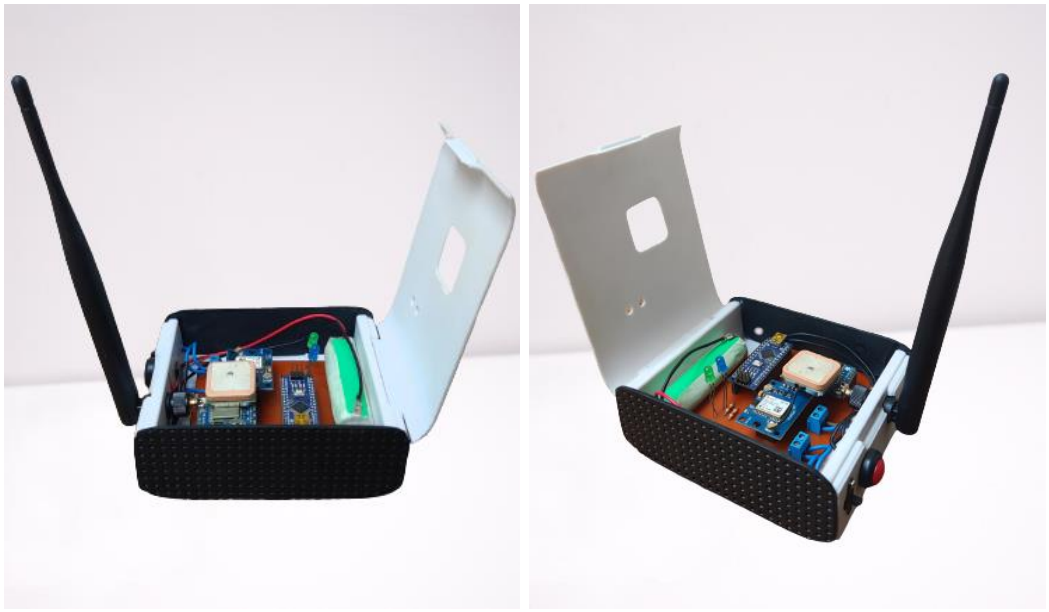


Figura 38. Vista interna de la caja del nodo [Autor].

La caja tiene unas dimensiones de 100 mm de ancho, 125 mm de largo y 45mm de alto, esta se puede apreciar en la Figura 39.

Figura 39. Estructura para el nodo [Autor].

2.2.4.4. Programación de los nodos LoRa-GPS

En este proceso se utilizó el IDE de Arduino para programar el microcontrolador de la placa de desarrollo Arduino Nano con las funcionalidades requeridas para los dispositivos finales de este sistema, teniendo en cuenta los parámetros necesarios para la comunicación en la red LoRaWAN, obtención y procesamiento de los datos del GPS, además de la funcionalidad del botón de emergencia. El nodo se programa como dispositivo

final con método de activación ABP con las claves de sesión generadas al agregar un dispositivo en la red del gateway registrado en TTN.

En la Figura 40 se ve reflejada la lógica de la programación del nodo LoRa-GPS en un diagrama de flujo.

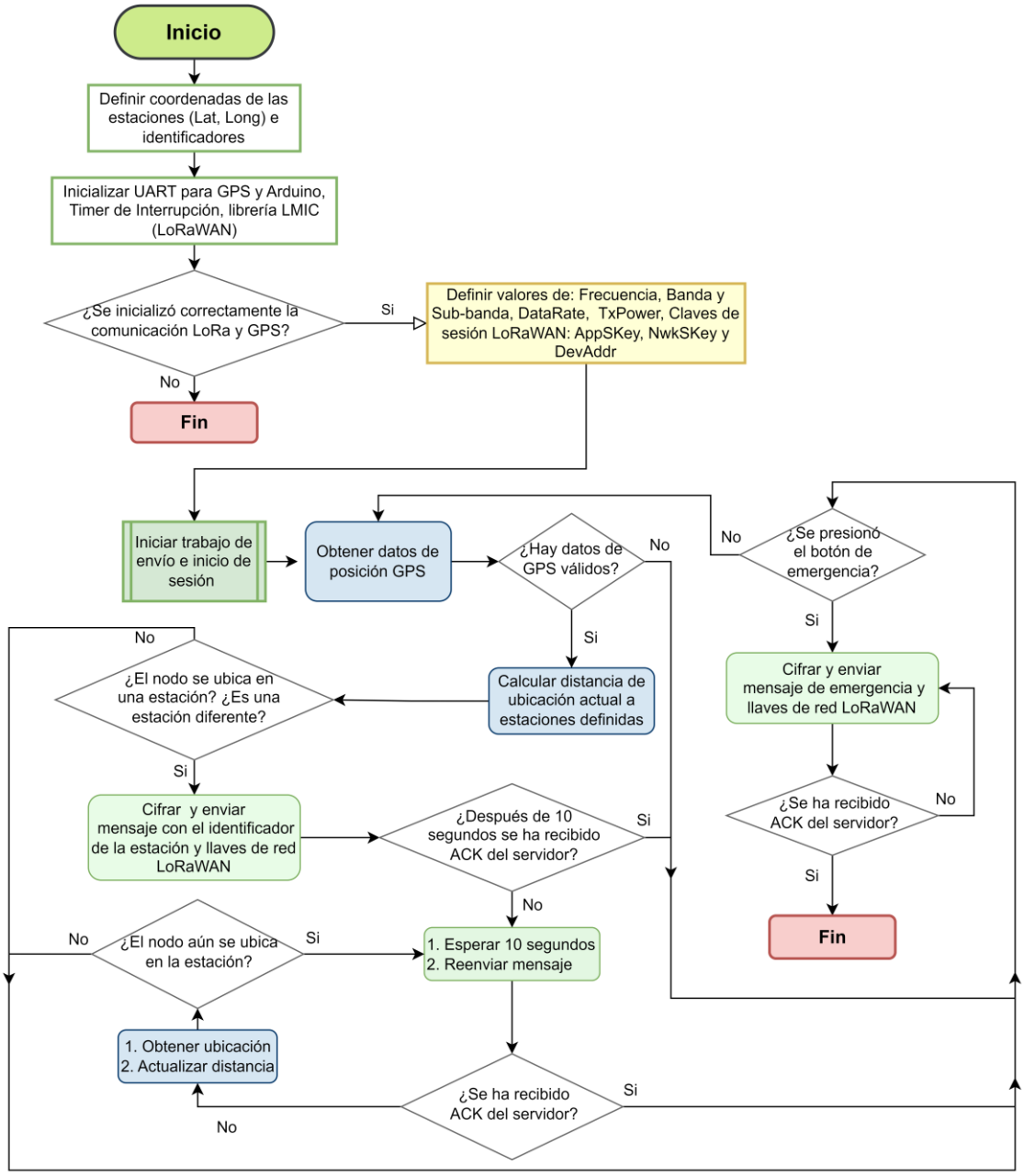


Figura 40. Flujograma de programación del Nodo LoRa-GPS [Autor].

Para la implementación de un nodo en una red LoRaWAN se empleó la librería para LoRaWAN LMIC, que proporciona una implementación completa para nodos de Clase A y Clase B, para las bandas EU-868 y US-915. Haciendo uso de LMIC se tiene la posibilidad de enviar datos a The Things Network. Esta

librería LMIC se adecua perfectamente a este prototipo al estar disponible y ser compatible con plataformas de programación como Arduino y módulos transceptores basados en el chip SX1276 (Arduino Reference, s.f.)

Una vez cargado el código con todas las claves de sesión para LoRaWAN en el Arduino Nano por primera vez el dispositivo final solicita al servidor TTN ser activado y culminar su registro y estar listo para su operación normal. En la Figura 41 se puede observar el proceso de activación y unión de los nodos ABP en la aplicación, después de iniciada sesión por primera vez.

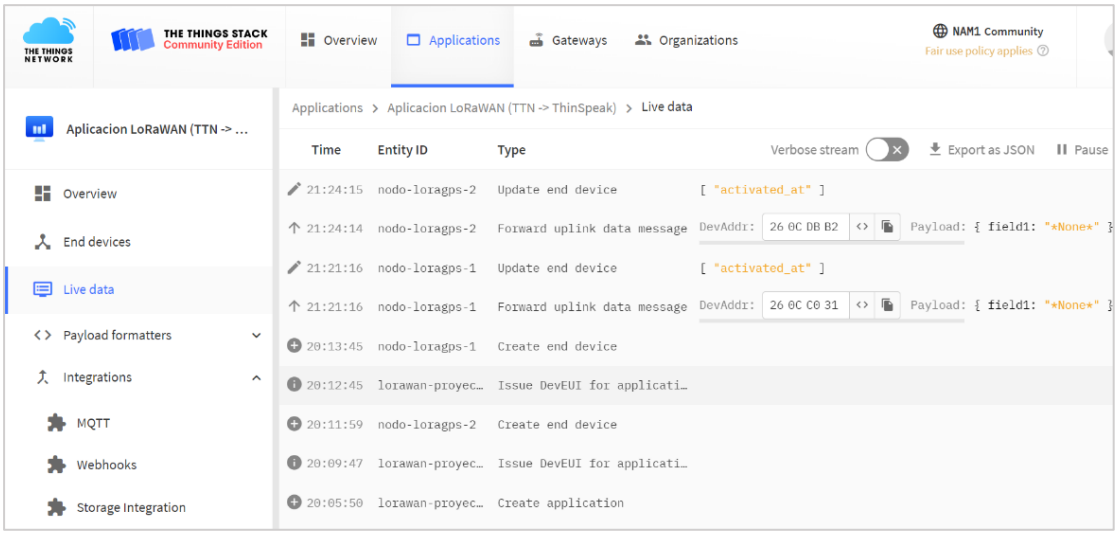


Figura 41. Activación y unión de nodos LoRa-GPS a la red [Autor].

2.2.5. Implementación del algoritmo de predicción RTTT

Para la implementación del algoritmo basado en RTTT se decidió utilizar el servidor ThingSpeak, ya que este ofrece una integración de Matlab en la nube. Para llevar a cabo la implementación del algoritmo de predicción basado en RTTT se siguieron los siguientes pasos:

Paso 1. Creación de cuenta y canales en ThingSpeak.

Luego de registrarse y obtener una cuenta de usuario gratuito, se crearon dos canales: uno para recopilar los datos de ubicación de los nodos enviados al gateway y provenientes de The Things Network por medio de la integración (Ubicación) y otro para almacenar las predicciones generadas por el algoritmo de predicción ejecutado en Matlab Analysis (Predicciones).

En la Figura 42 puede observarse la vista general de los dos canales creados.

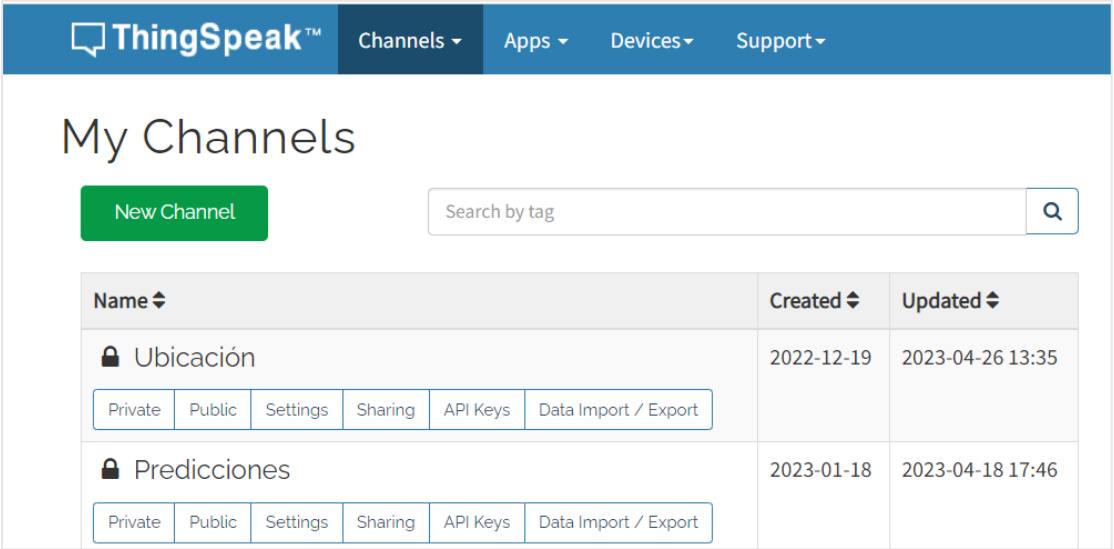


Figura 42. Canales del servidor ThingSpeak [Autor].

Canal Ubicación: Establece comunicación con TTN para registrar la información de los arribos de las unidades de TUC en tiempo real gracias a la Integración Webhook de ThingSpeak en TTN.

Canal Predicciones: Se configura para almacenar los tiempos de predicción de acuerdo a la última estación visitada por una unidad de TUC, con el fin de que sean leídos por el sistema de visualización que ejecuta la Raspberry Pi.

Paso 2. Crear una Reacción automática en ThingSpeak

Se agrega y configura una reacción para ejecutar el algoritmo basado en RTTT desarrollado en Matlab Analysis cada vez que se reciba un nuevo dato de los nodos en el canal de Ubicación. En la Figura 43 se puede observar los detalles de la reacción creada.

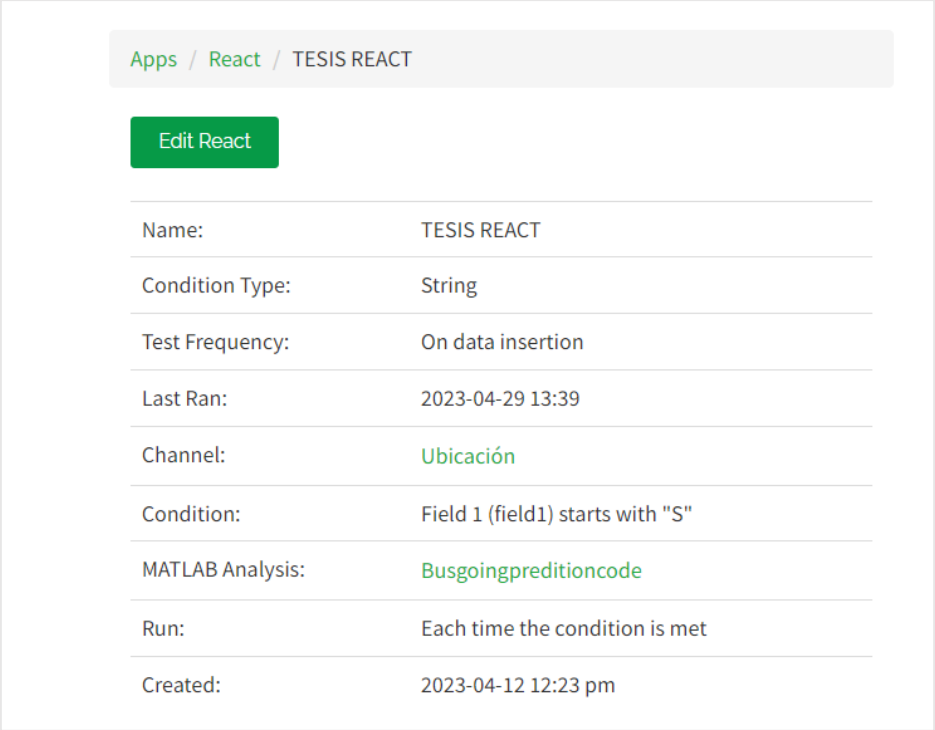


Figura 43. Configuración de la reacción [Autor].

Esta reacción se configura de manera tal que cada vez que se escriba un nuevo dato en el canal llamado “Ubicación” se ejecute el código de Matlab, en el cual se procesa el algoritmo de predicción.

Paso 3. Desarrollo de código en Matlab Analysis

Con el objetivo de leer los datos del canal “Ubicación”, procesarlos y escribir las predicciones en el canal “Predicciones” y sus respectivos campos, se desarrolló un código en la sección de Matlab Analysis, el cual se ejecuta cada vez que las condiciones de la reacción se cumplen. Es importante mencionar que el cálculo de las predicciones está basado en la Algoritmo de predicción de arribo de unidades TUC detallado en la sección 1.8.

En la Figura 44, se detalla el diagrama de flujo que describe la programación del código de Matlab y su función dentro del sistema.

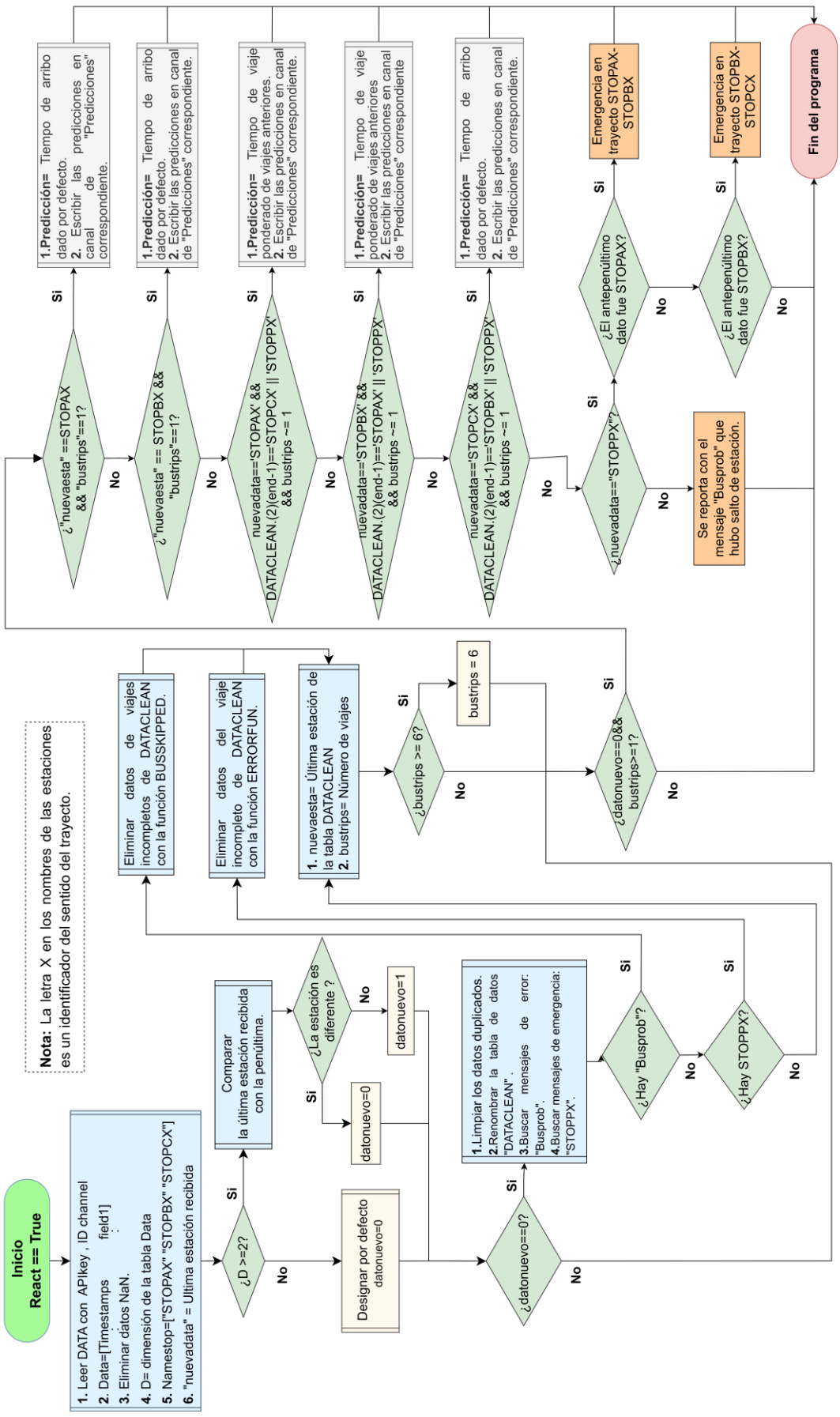


Figura 44. Flujograma del código de implementación en Matlab [Autor].

2.2.6. Desarrollo de la interfaz gráfica

La implementación del sistema de visualización de información de arribo y ubicación de las unidades de transporte urbano se llevó a cabo mediante el desarrollo de una interfaz gráfica informativa programada en Python, que se ejecuta en una Raspberry Pi. Esta debe estar ubicada en las estaciones de buses como un puesto de visualización donde los usuarios puedan informarse acerca de las predicciones de los tiempos de llegada hechas por el sistema de predicción basado en el algoritmo RTTT.

Es importante mencionar que la interfaz gráfica fue diseñada para que sea agradable e intuitiva para el usuario, permitiendo una fácil interpretación de las predicciones y alertas relacionadas con las unidades de TUC.

2.2.6.1. Programación en Python

Como se detalló anteriormente, el software de la interfaz gráfica fue programado en el lenguaje de programación de alto nivel Python, utilizando Tkinter, que es una librería del lenguaje de programación Python que funciona para la creación, posicionamiento y desarrollo de una interfaz gráfica de escritorio con Python (Keepcoding, 2023).

La obtención de datos del servidor ThingSpeak se realiza utilizando la biblioteca **urllib** de Python, la cual envía una solicitud HTTP a la URL especificada en formato JSON.

Tal como se muestra en la Figura 45, en estas líneas de código se envía una solicitud GET a la API de ThingSpeak, que devuelve los datos de los campos correspondientes del canal 2112563. La solicitud incluye el parámetro **api_key** que se utiliza para autenticar la solicitud y garantizar que solo los usuarios autorizados puedan acceder a los datos. Además, el parámetro **results=1** especifica que solo se debe devolver el último registro de datos del canal.

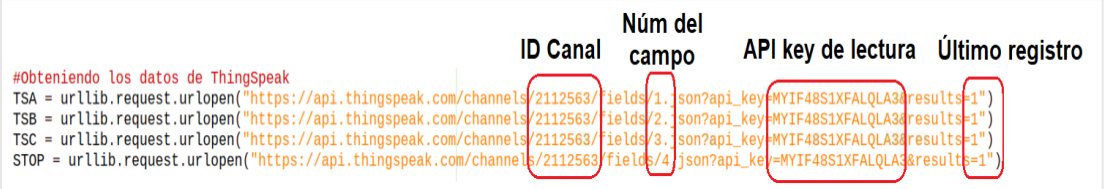


Figura 45. Líneas de código para enviar solicitud HTTP [Autor].

La solicitud HTTP devuelve una cadena de caracteres como la que se muestra en la Figura 46 a continuación.

{"channel":{"id":2112563,"name":"Prediction Coming","latitude":"0.0","longitude":"0.0","field1":"Field Label 1","field2":"Field Label 2","field3":"Field Label 3","field4":"Field Label 4","created_at":"2023-04-19T03:31:23Z","updated_at":"2023-04-19T03:31:23Z","last_entry_id":30},"feeds":[{"created_at":"2023-04-29T19:52:32Z","entry_id":30,"field4":"STOPA2"}]}

Figura 46. Cadena de respuesta a solicitud HTTP del campo [Autor].

Estas cadenas contienen la información completa de cada viaje y su respectiva predicción dependiendo del campo que se solicite, La Figura 47 muestra el fragmento de código que se encarga de leer el contenido solicitado a través de HTTP, convertirlo de cadena a diccionario o lista y acceder a los datos de interés.

```
# Lee el contenido de una respuesta HTTP
responseA = TSA.read()
responseB = TSB.read()
responseC = TSC.read()
responseSTOP = STOP.read()

#Convierte la cadena de texto responseA en un objeto de Python,
#para ser manipulado y accedido como un diccionario o lista de Python

dataA = json.loads(responseA)
dataB = json.loads(responseB)
dataC = json.loads(responseC)
dataSTOP = json.loads(responseSTOP)

#Manipulación y el acceso a los datos de interés.
entreidnewA = dataA['channel']['last_entry_id']
entreidnewB = dataB['channel']['last_entry_id']
entreidnewC = dataC['channel']['last_entry_id']
#Ultima estacion visitada
ultdataSTOP = dataSTOP['feeds'][0]['field4']
```

Figura 47. Lectura y acceso de datos de interés a través de HTTP [Autor].

Finalmente, una vez procesada la información obtenida de las solicitudes HTTP, se podrán tener las predicciones, funcionalidades y alertas ya listas para ser presentadas en la interfaz.

2.2.6.2. Visualización en pantalla

La interfaz de visualización, como medio para informar al usuario acerca de los acontecimientos del sistema, consta de tres ventanas o pantallas: Inicio, Menú y Predicciones. En la Figura 48 se puede visualizar el modelo de la interfaz gráfica desarrollada, funciones y transiciones.

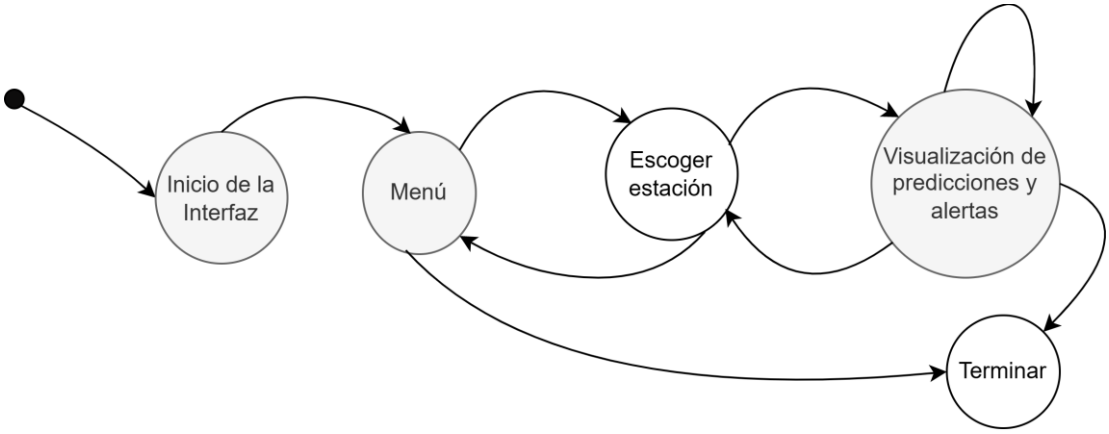


Figura 48. Modelo de la interfaz gráfica [Autor].

La pantalla de inicio muestra detalles acerca del sistema. Desde esta se puede avanzar hacia el menú de inicio para seleccionar la estación de interés, y de esta manera se podrá acceder a las predicciones e información de las unidades de transporte cercanas a la estación.

En la Figura 49 se puede apreciar el diseño de la pantalla de Inicio con una presentación sobre el sistema y un botón para hacer transición a la pantalla de Menú.

Figura 49. Pantalla de Inicio de la interfaz de usuario [Autor].

La pantalla de Menú (Figura 50) consta de una lista de estaciones de bus, entre las cuales se puede elegir en torno a qué estación se centrará la visualización

de las predicciones. El personal que instala la estación de visualización debe seleccionar la estación en la cual estará la interfaz informativa, esto con el fin de ver los tiempos de arribo de las estaciones que están en su entorno.

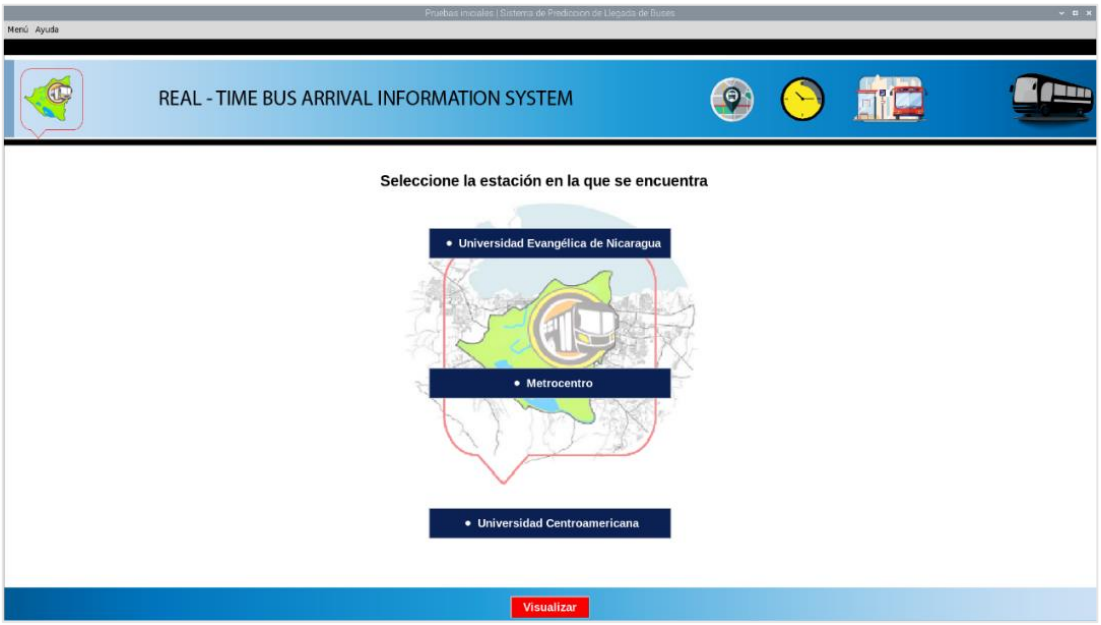


Figura 50. Menú de estaciones a visualizar [Autor].

Por último, estará la pantalla principal de predicciones, mostrada en la Figura 51, que es la que el usuario final podrá visualizar una vez el módulo de visualización ha sido instalado con una estación seleccionada.

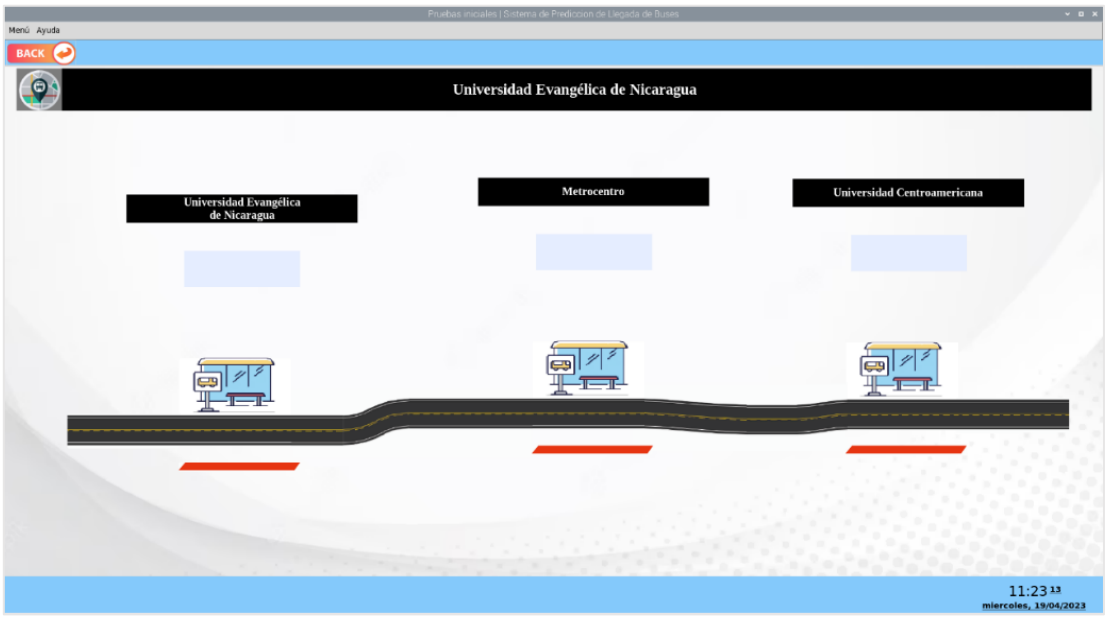


Figura 51. Pantalla de visualización de predicciones [Autor].

En esta pantalla se podrá observar el tiempo faltante para que la próxima unidad de transporte arribe a las estaciones del entorno, este tiempo irá disminuyendo a medida que pase el tiempo a partir de la llegada del bus a estaciones anteriores; si el tiempo se agota se muestra un mensaje que informe acerca del retraso.

Cabe destacar que estos tiempos se actualizarán cada vez que el bus llegue a las estaciones y se hagan nuevas predicciones, a la vez que la animación del bus irá apareciendo cuando llega a una nueva estación, y abandonará la estación una vez culminado el tiempo aproximado que se ha establecido para el abordaje y bajada de pasajeros. En la Figura 52 se puede observar uno de los muchos escenarios que se pueden dar; en este específico el bus va de camino a la siguiente estación teniendo un retraso en su llegada.

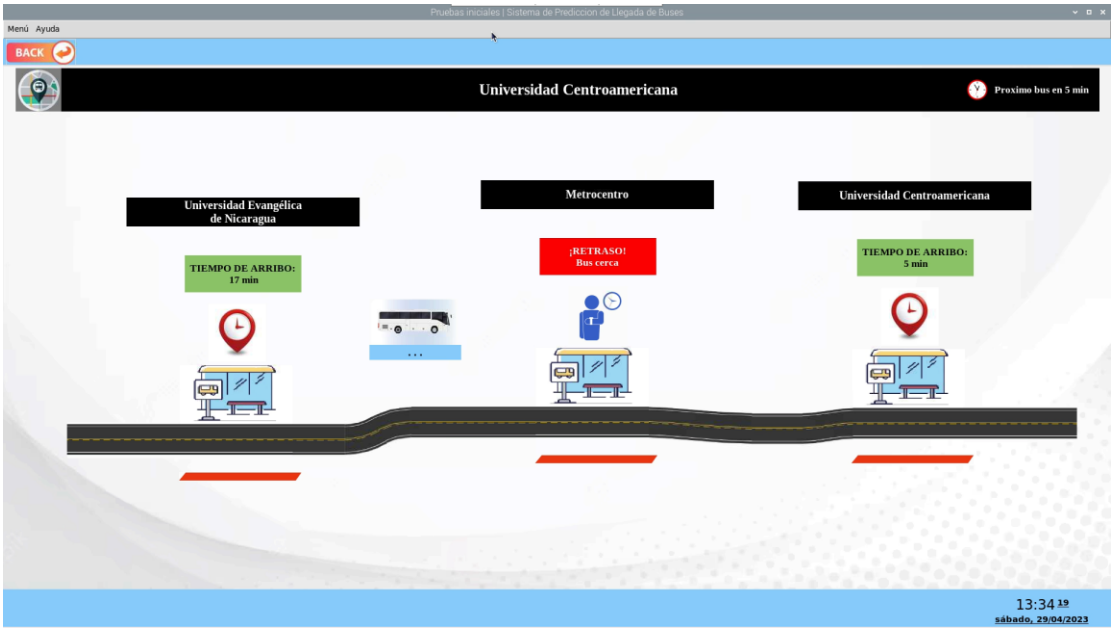


Figura 52. Escenario de ejemplo para la llegada de buses [Autor].

Además, a como se mencionó en los requerimientos del sistema, se integró la funcionalidad del botón de emergencia en caso de que el bus presente algún desperfecto que lo haga estar inoperativo. En caso de que el conductor presione este botón, en la interfaz gráfica se mostrará un mensaje que informa acerca del evento, y se presenta una animación en el lugar de ocurrencia a como se puede ver en la Figura 53.

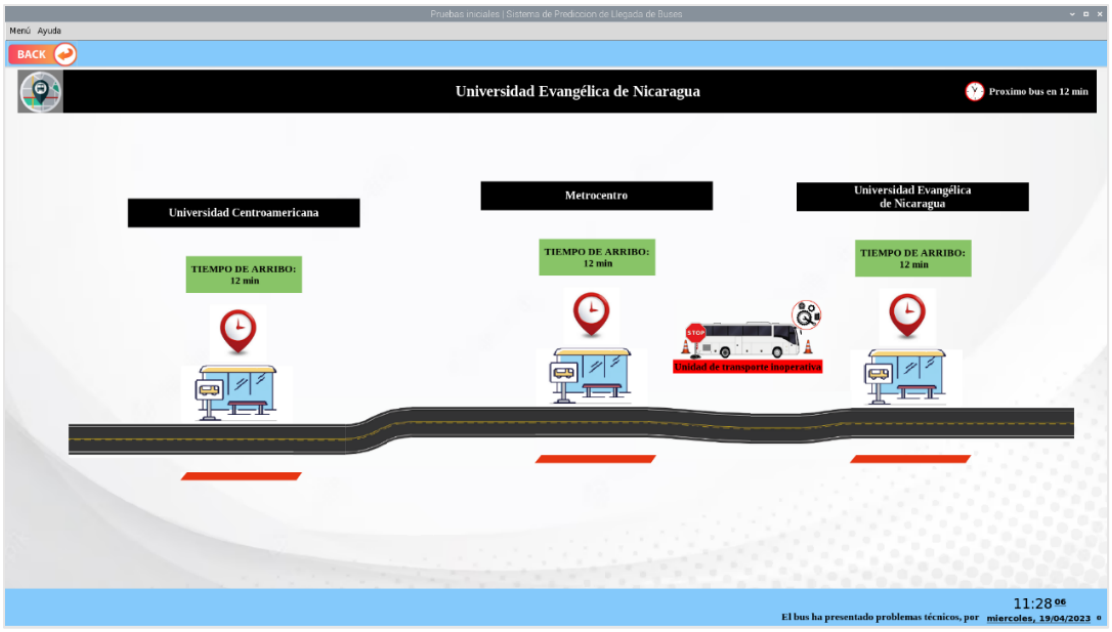


Figura 53. Escenario de ejemplo para una emergencia [Autor].

2.3. Resultados

2.3.1. Hardware

Para la realización de las pruebas, y como parte del desarrollo de un prototipo para el sistema propuesto, se desarrolló el hardware necesario para la implementación de la red LoRaWAN con la adquisición de una unidad de Gateway Dragino de 915 MHz y la fabricación de nodos finales con su correspondiente caja protectora que ayuda a mantener el circuito y sus componentes en buen estado. En la Figura 54 se muestra el hardware para la red LoRa.

Con estos nodos se realizarán las pruebas en un ambiente real con el gateway ubicado en un lugar estratégico, de modo que los dispositivos finales se encuentren en el área de cobertura durante el recorrido seleccionado.

Figura 54. Dispositivos finales y Gateway LoRaWAN [Autor].

En la Figura 55 se muestra un módulo de visualización compuesto por una Raspberry Pi 3B+ para la obtención de datos y ejecución de la interfaz y una laptop que funciona como una pantalla o monitor improvisado para la visualización. En este caso la computadora se conecta a la Rapsberry Pi por medio de un software de conexión remota, lo que sería fácilmente sustituible por un monitor con puerto HDMI de tenerse la disponibilidad.

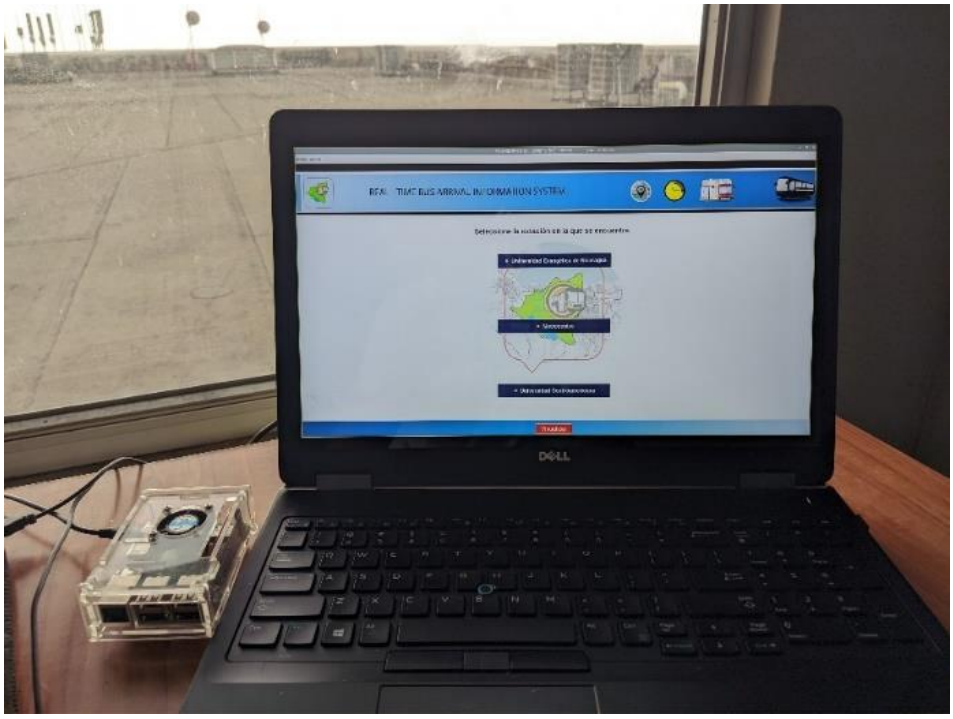


Figura 55. Módulo de visualización de la interfaz [Autor].

2.3.2. Comunicación LoRa

La comunicación LoRa en este sistema es parte esencial, específicamente el alcance máximo a la que esta se puede dar, ya que influye directamente en la cantidad de puertas de enlaces que se tendrían que incluir en un diseño que comprenda una mayor área.

Con el objetivo de seleccionar el recorrido para las pruebas finales en un ambiente real, primeramente, se realizó una prueba de alcance de la comunicación LoRa para conocer la distancia máxima a la que se podría tener comunicación entre el gateway (ubicación fija) y los nodos (móviles).

Esta prueba se realizó con el gateway ubicado en el lugar con mayor altitud al que se tuviese acceso; la azotea del Edificio Rigoberto López Pérez de la Universidad Nacional de Ingeniería, desde la cual se puede tener una buena vista de gran parte de la Pista Juan Pablo II, Paseo Tiscapa, Estadio Nacional Soberanía y sus alrededores. La Figura 56, muestra la ubicación específica.

Figura 56. Ubicación del gateway [Autor].

2.3.2.1. Alcance de la comunicación

Según el fabricante de los módulos LoRa utilizados, el alcance de estos puede llegar a ser de hasta 2 Km en campo abierto; esto dependiendo de

obstrucciones, señales de la misma frecuencia, antena, potencia de salida y la cantidad de señales presentes en el entorno.

Con el fin de analizar el comportamiento de la tecnología LoRa en un entorno urbano, como lo es la ciudad de Managua, se realizó el recorrido marcado en la Figura 57, en las cercanías de la ubicación del gateway, en la UNI.

Figura 57. Recorrido alrededor del gateway en prueba de alcance [Autor].

En esta prueba, el nodo envía un paquete cada 10 segundos de forma continua hacia el gateway, mientras se recorren las calles alrededor de él, principalmente pista principal y calles más traficadas.

Analizando los resultados de esta prueba se puede concluir que la efectividad de la comunicación va a depender de factores como el estado del tráfico vehicular y la altura a la que se encuentra el nodo en relación a la altura del gateway y del suelo.

En la Figura 57 se puede distinguir el recorrido de color verde en el cual se obtuvo señal con calidad entre excelente y buena, zona en la cual se recibían el 100% de los paquetes enviados por el nodo a pesar de no tenerse una línea de vista tan despejada. Además, se pudo observar que a medida que el nodo

se alejaba del gateway, naturalmente, la intensidad de la señal disminuía, y al haber mucha afluencia vehicular o mucha vegetación esto empeoraba a tal punto de perderse el enlace con el gateway en zonas de señales con RSSI[16] de -95 dBm o menos (recorrido amarillo); las cuales no pueden considerarse para el ambiente de pruebas para el sistema ya que la comunicación no es muy fiable. En la Figura 58 se muestra el panorama alrededor del gateway.

Figura 58. Panorama alrededor del gateway [Autor].

Los resultados podrían mejorar en caso de ubicar el gateway en punto más alto y cercano a la pista principal en la cual se requiera tener cobertura para las unidades de transporte. En esta prueba se logró tener una distancia máxima de comunicación de aproximadamente 1.5 Km con RSSI de señal cercano a los -100 dBm y un SNR muy bajo de -7.8 dB, que evidencia una señal muy débil y con mucha interferencia, cifra bastante considerable tomando en cuenta que no se tenía línea de vista despejada y que es un entorno urbano que genera mucho ruido.

2.3.2.2. Área geográfica de las pruebas

Una vez hecho el análisis de la zona con calidad de señal adecuada para implementar el sistema, teniendo en cuenta que debe asegurarse que haya comunicación fiable aun en condiciones desfavorables de tráfico y atenuación

[16] **RSSI:** Indicador de Fuerza de la Señal Recibida

por vegetación, y demás fenómenos que afectan la comunicación inalámbrica, se procedió a seleccionar una pista principal para la realización de las pruebas finales del sistema, optándose por un recorrido propio de una gran cantidad de unidades de transporte colectivo de diferentes rutas o cooperativas como lo es la Pista Juan Pablo II, el cual puede observarse en la Figura 59.

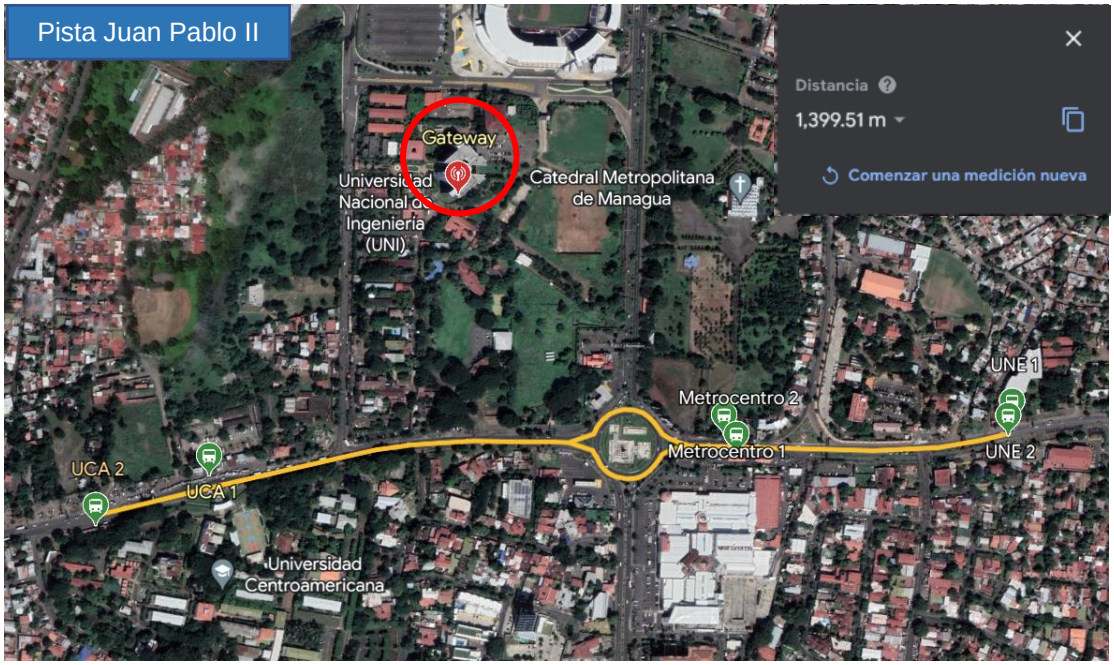


Figura 59. Recorrido seleccionado para las pruebas del sistema [Autor].

A lo largo de este recorrido de aproximadamente 1.4 Km (vía color amarillo en la Figura 59) se puede observar una considerable densidad poblacional y edificaciones, sin embargo, ya se ha verificado en la prueba de alcance descrita anteriormente que en este trayecto es viable implementar un ambiente de pruebas aprovechando que es un recorrido real de las unidades de TUC, para demostrar el funcionamiento del sistema de predicción. En el trayecto se ubican 6 estaciones de buses reales (marcadores verdes) las cuales formarán parte del sistema que se ha desarrollado como puntos de interés, las cuales son: Universidad Evangélica Nicaragüense Martin Luther King (UENIC), Metrocentro y la Universidad Centroamericana (UCA); teniéndose una distancia de aproximada de 900 metros entre el gateway y la estación más lejana (UENIC).

2.3.3. Prueba de funcionamiento del sistema

Habiendo culminado las anteriores etapas se procedió a realizar una prueba en un recorrido real de los buses de Managua con el objetivo de validar el correcto funcionamiento del sistema y todas sus etapas de preparación relacionadas con la red LoRaWAN y el subsistema de visualización para los usuarios.

Para esto se hizo uso de un vehículo particular, que se puede visualizar en la Figura 60, con el cual se estuvo recorriendo la pista seleccionada en ambos sentidos (ida y retorno) hasta completar el número de viajes necesarios para medir la eficiencia del sistema y evaluar el comportamiento general del mismo.

Figura 60. Vehículo particular para las pruebas del sistema [Autor].

En la realización de la prueba se tomaron en cuenta las siguientes condiciones:

- Al arribar a las estaciones de buses del recorrido se deberá detener la marcha por un tiempo cercano a un minuto simulando el tiempo de abordaje y bajada de los pasajeros del bus (tiempo promedio medido en las estaciones)
- El nodo debe ubicarse cercano a una de las ventanillas para facilitar la comunicación LoRa y sobre todo mejorar la recepción GPS, ya que trabaja en una banda de frecuencia superior a LoRa y es más susceptible al multitrayecto y ruido por obstáculos como el vehículo mismo.
- La velocidad debe ser al ritmo del tráfico en el momento, tratando de copiar el comportamiento de los buses, a una velocidad máxima de 45 Km/h, que es

límite de velocidad en el perímetro urbano (Ley para el Régimen de Circulación Vehicular e Infracciones de Tránsito, 2022, Arto.37 *bis*).

- Para realizar este recorrido se requirió tomar la rotonda Cristo Rey y hacer un giro en la zona de Enel para ingresar en el sistema en el sentido contrario una vez culminado el viaje de ida o vuelta (Figura 61), con lo que se configuró el tiempo en el que el vehículo arriba a las estaciones a 20 minutos. Es decir, aproximadamente cada 20 minutos el vehículo arribará a cada una de las estaciones contando desde su última visita. Sin embargo, en un sistema final se tomaría en cuenta que cada bus de la misma cooperativa o ruta (ruta 110, por ejemplo) arriba a una estación aproximadamente cada 10 minutos (tiempo medido y frecuencia con la que salen los buses de la terminal)

Figura 61. Puntos de retorno del recorrido [Autor].

En la Figura 62 se pueden observar los elementos del prototipo funcional del sistema que se utilizarán.

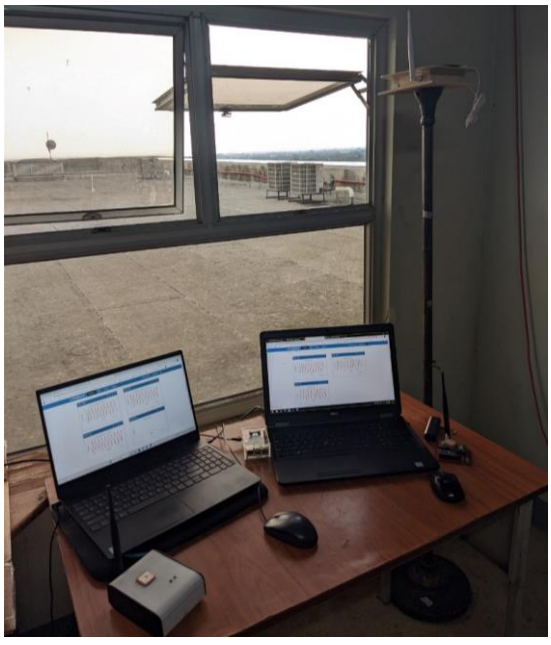


Figura 62. Elementos utilizados para el prototipo del sistema [Autor].

Estando en la localización más alta, con vista hacia el trayecto de pruebas, al que tuviéramos acceso. En la Figura 63 se muestra la ubicación del gateway en un pequeño mástil para ganar altura en la esquina de la azotea del edificio con mejor vista hacia la pista de interés, con el fin de crear mejores condiciones de recepción

Figura 63. Gateway en azotea del Edificio Rigoberto López Pérez [Autor].

Asimismo, en la Figura 64, se muestra un pequeño puesto de monitoreo improvisado desde el cual se hace un seguimiento a la interfaz gráfica informativa, y a la llegada y procesamiento de los datos provenientes del nodo a bordo del vehículo de pruebas.

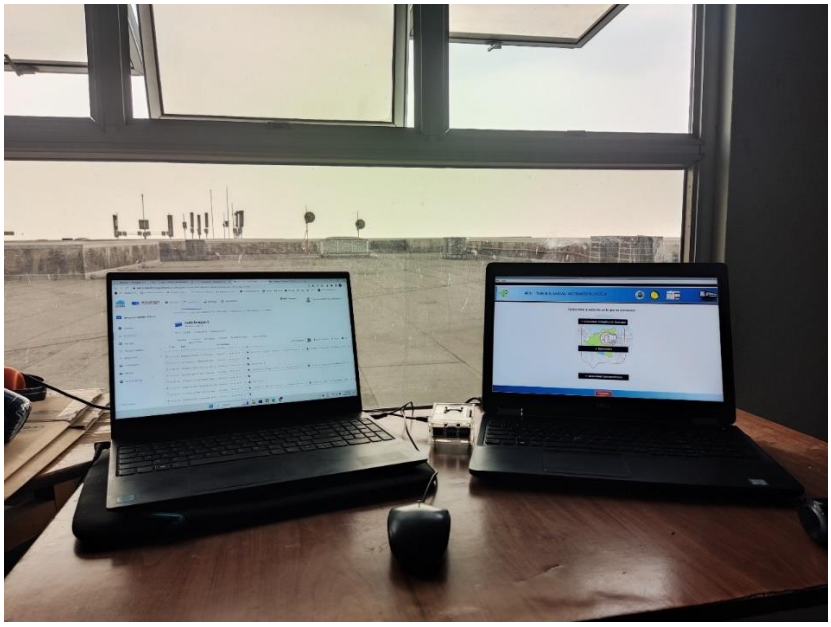


Figura 64. Estación de monitoreo del sistema durante las pruebas [Autor].

Una vez acondicionado todo el equipo se dio inicio a la prueba, que consistió en transitar 10 veces a través del trayecto antes especificado en ambos sentidos, respetando todos los puntos tratados anteriormente, teniéndose un excelente comportamiento del sistema en todas sus etapas.

2.3.4. LoRaWAN e Interfaz gráfica en Python

Al llegar a una de las estaciones cuyas coordenadas están registradas en el microcontrolador como paradas de buses oficiales en el recorrido, el nodo procede a enviar una cadena con la ubicación (estación) en la que se encuentra. Estos paquetes al ser recibidos en el gateway son subidos a la red de las cosas, en donde son decodificados con ayuda del formato de Uplink, obteniéndose la carga útil del paquete enviado desde el dispositivo final y algunos detalles sobre la recepción de la señal y propiedades de esta, a como se puede visualizar en la Figura 65.

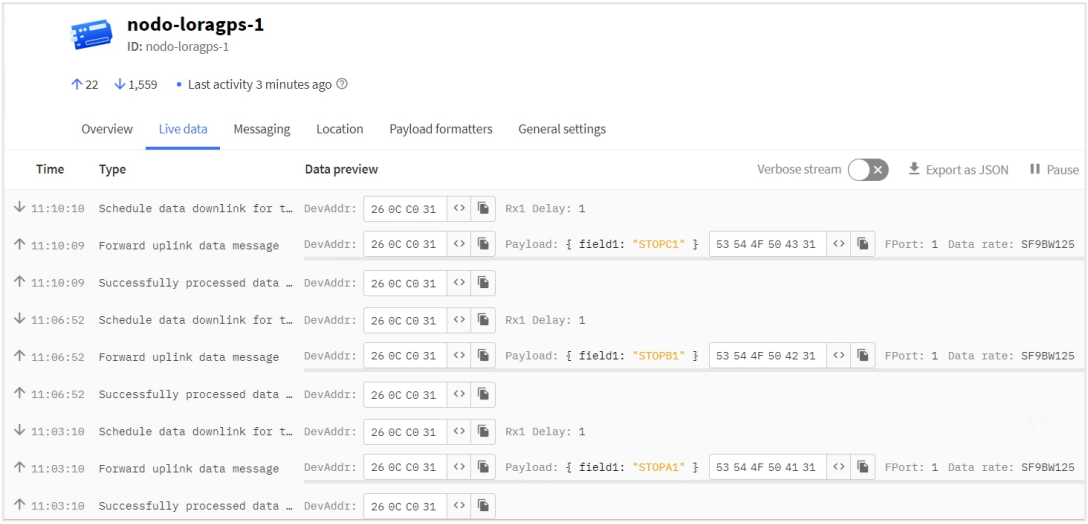


Figura 65. Recepción de paquetes y Uplink a TTN [Autor].

Estos datos son subidos a un canal de ThingSpeak a través de la integración Webhook mencionada en la sección 2.2.3 y sirven de entrada para el algoritmo de predicción que los toma para predecir nuevos tiempos de llegada a partir de ellos y escribirlos en un canal de salida.

Al recorrer el circuito de prueba 10 veces (10 viajes) se han hecho las predicciones adecuadamente para ambos sentidos (ida y vuelta), reflejándose

estas en las gráficas del canal de salida para los viajes de ida y de retorno a la terminal en la Figura 66.

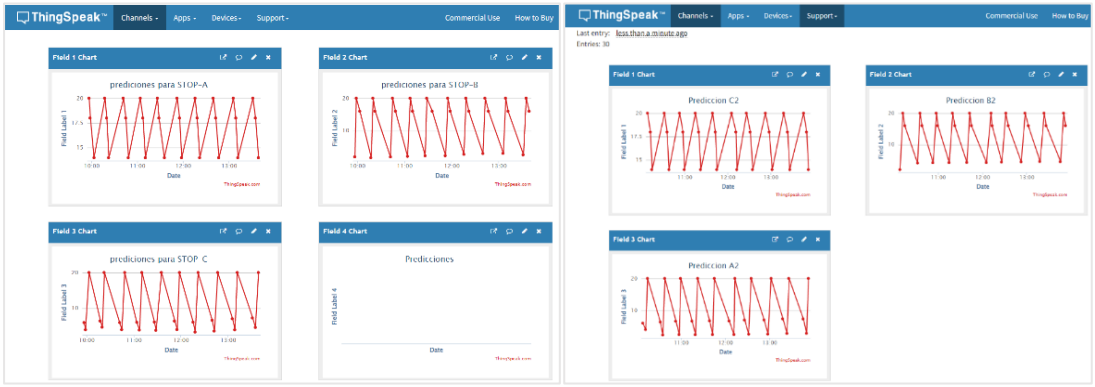


Figura 66. Gráficas en ThingSpeak de predicciones en 10 viajes [Autor].

Al predecirse nuevos tiempos de llegada a las próximas estaciones, a como pude verse en la figura anterior, los datos son almacenados en el canal de salida establecido en el código de Matlab y estos están listos para ser leídos para una aplicación externa a través de API Request, que son enlaces autorizados para escribir y leer datos del canal y sus campos desde cualquier navegador o dispositivo con acceso a internet.

Teniendo los datos de las predicciones en el canal de ThingSpeak, la Rapsberry Pi conectada a internet, con el código que ejecuta, lee estas predicciones para mostrarlas en la interfaz informativa a como se muestra en la Figura 67.

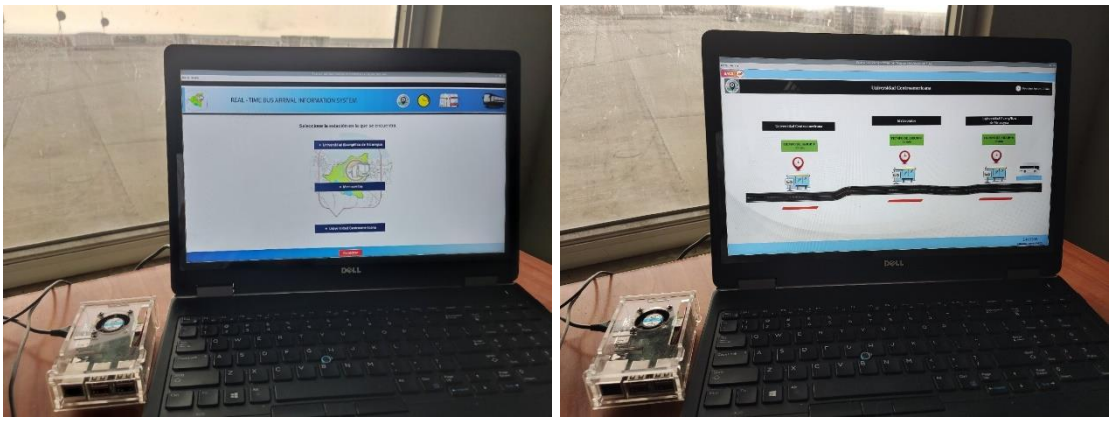


Figura 67. Interfaz corriendo en Raspberry Pi vista en pantalla [Autor].

Al leerse del canal predicciones con un identificador diferente, son tomados los datos de estación actual y predicciones para mostrarlos. En la Figura 68 se

muestra la interfaz cuando el vehículo arribó a la estación de la UENIC en sentido de ida en la décima vuelta y se actualizan las predicciones para las próximas estaciones y para la misma en la que se encuentra el bus.

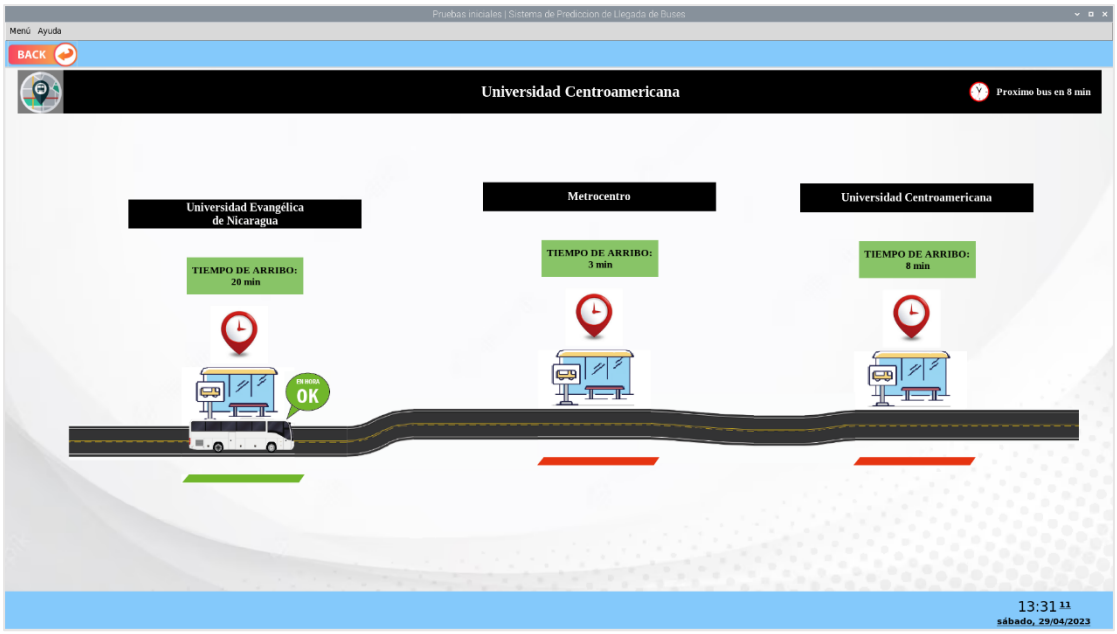


Figura 68. Llegada del vehículo a la estación de la UENIC [Autor].

Una vez ha terminado el tiempo de abordaje simulado durante la prueba vemos en la Figura 69 que en la interfaz el bus abandona la estación y se encamina a la estación de Metrocentro.

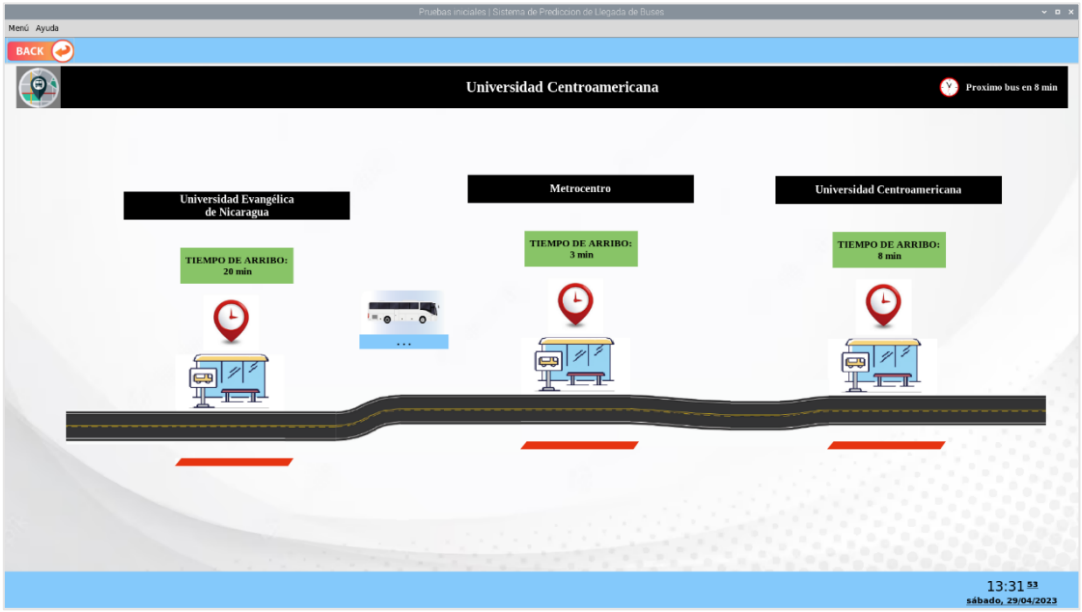


Figura 69. Bus después de transcurrido el tiempo de abordaje [Autor].

Otro elemento importante del diseño del nodo y la interfaz fue la inclusión del botón de emergencia, en caso de darse algún desperfecto mecánico o situación no esperada que impidiera la operación de algún bus. Se puso a prueba esta función en una de las vueltas en el sentido de retorno, al presionar el botón en el recorrido de regreso entre la UCA y Metrocentro, recibiéndose este aviso y registrándose en la interfaz en forma de alerta, quedando inoperativo el nodo y descartado el viaje, tal como se observa en la Figura 70.

Figura 70. Prueba del botón de emergencia en el nodo e interfaz [Autor].

2.3.5. Eficiencia del sistema de predicción

Con el fin de estudiar la eficiencia del algoritmo de predicción basado en RTTT presentado en la sección 1.8, se calculó el error de predicción en diferentes escenarios y segmentos del trayecto, tanto de “ida” como de “regreso”, de viajes reales del ambiente de prueba. Para mantener la generalidad en la variable "Error" y en el nombre de las estaciones de autobuses, que pueden variar en el sistema, la fórmula para el cálculo del error de predicción, descrita en la Ecuación 4, puede explicar de la siguiente manera:

La variable "Error" para el trayecto desde "StopA" hasta "StopB" se define como la diferencia entre la hora real de llegada de la unidad de TUC a "StopB" y la hora real de llegada de la unidad de TUC a "StopA", sumada a la predicción de la hora de llegada del autobús desde "StopA" hasta "StopB". Por lo tanto:

$$\boldsymbol{Error}_{(\mathbf{StopA-StopB})} = Hora\ real\ de\ arribo_{(StopB)} - Hora\ real\ de\ arribo_{(StopA)} + \cdots\ Predicción_{(StopA \rightarrow StopB)} \quad \textbf{(4)}$$

Las sentencias principales en Matlab para realizar el error de predicción se muestran en la Figura 71.

```
%Calculando error de predicción  STOP-A A STOP-B
ErrorAB=time(between(tiempodeprediccionB+PIksA,  PIksB))
difeAB=[seconds( PIksB-(tiempodeprediccionB+PIksA))];

%Calculando error de predicción DE STOP-B A STOP-C
ErrorBC=time(between(tiempodeprediccionC+PIksB,  PIksC));
difeBC = [seconds(PIksC-(tiempodeprediccionC+PIksB))];
```

Figura 71. Código en Matlab para calcular el error de predicción [Autor].

2.3.5.1. Análisis de sensibilidad para el algoritmo basado en RTTT.

El algoritmo basado en el Método RTTT consiste en monitorear el tiempo de llegada de una unidad de TUC a una estación del sistema y utilizar la información del tiempo de viaje entre estaciones de unidades anteriores que se registraron previamente para realizar la predicción de arribo en las próximas estaciones. Con el propósito de entender la contribución de los viajes anteriores de las unidades de TUC se llevó a cabo un estudio de sensibilidad. El análisis consistió en identificar el número óptimo de viajes anteriores que podrían utilizarse para proporcionar predicciones más precisas.

Para llevar a cabo el análisis de sensibilidad, se aplicaron medidas estadísticas como error absoluto medio y la desviación estándar a los conjuntos de errores de predicción que se generaban en diferentes escenarios en los cuales se utilizaron de 1 a 5 viajes anteriores para predecir. Considerando que se llevaron a cabo diez viajes en el entorno de prueba, es posible realizar este análisis a partir del sexto viaje.

1) **Análisis de sensibilidad en los segmentos de trayecto de “ida”**

La gráfica correspondiente al segmento de UENIC→Metrocentro, que se encuentra en la parte superior de la Figura 72, demuestra que, la media y la desviación estándar se estabilizan alrededor de los 30 segundos. Sin embargo, se observa una irregularidad en la desviación estándar de RTTT (5*), consecuencia de un error “abrupto” en el viaje número 10.

Asimismo, en la Figura 72, en la parte inferior se encuentra la gráfica del análisis para el segmento de Metrocentro→UCA. Esta muestra que la media

de error se estabiliza alrededor de 50 segundos; sin embargo, la desviación estándar desciende a medida que se agregan viajes anteriores a la predicción.

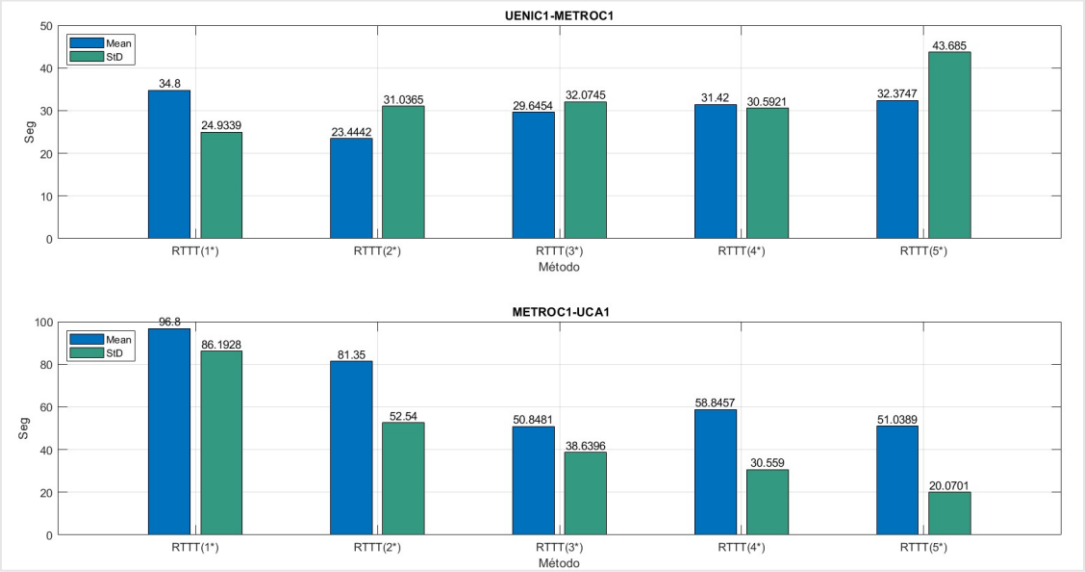


Figura 72. Media y Desviación estándar del error de predicción del método basado RTTT en los segmentos del trayecto de "ida" UENIC-METRO y METROC-UCA [Autor].
Nota: #* indica el número de viajes anteriores utilizados en el algoritmo basado en RTTT.

2) Análisis de sensibilidad en los segmentos de trayecto de "regreso"

Analizando la sensibilidad del algoritmo en trayecto de "regreso", cabe destacar que, los resultados fueron más estables, mientras que los viajes del trayecto de ida fueron más irregulares, lo cual se corrobora al comparar los comportamientos de las medidas estadísticas.

La gráfica superior de la Figura 73 muestra que en el segmento desde la UCA→Metrocentro, la media y la desviación estándar de los errores absoluto de predicción son estables en alrededor de 35 y 18 segundos, respectivamente. Es importante destacar que en este segmento el método RTTT (5*) presenta la media de error absoluto más baja, aunque con un ligero aumento en la desviación estándar.

Por otro lado, la gráfica ubicada en la parte inferior de la Figura 73 corresponde al comportamiento de la sensibilidad del algoritmo en el segmento de Metrocentro→UENIC. Dicha gráfica demuestra que la media y la desviación estándar de los errores absoluto de predicción son relativamente estables en torno a los 30 y 32 segundos, respectivamente. Se observa que el método RTTT (5*) nuevamente presenta la media de error absoluto más baja (27.23

segundos), con una desviación estándar muy similar a los métodos RTTT (4*) y RTTT (3*).

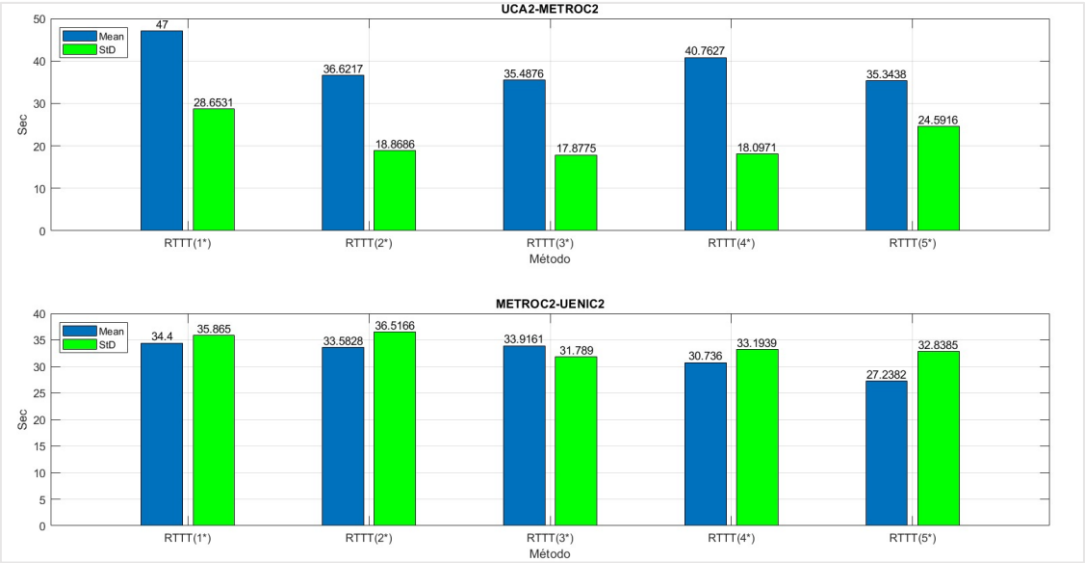


Figura 73. Media y desviación estándar del error de predicción del método basado RTTT en los segmentos del trayecto de "regreso" UCA2-METROC2 y METROC2 -UENIC2 [Autor].

Después de analizar la sensibilidad en todos los segmentos de los trayectos de ida y regreso, podemos concluir que:

- RTTT (5*) es el mejor escenario en tres de los cuatro segmentos analizados, debido a que generalmente la media de error absoluto de predicción es más baja en comparación con los otros métodos. Sin embargo, es importante tener en cuenta que en el análisis del segmento de Metrocentro →UCA, el método RTTT (3*) presenta una media ligeramente más baja (1 segundo) en comparación con RTTT (5*), pero con una desviación estándar 10 segundos menor que RTTT (3*), lo hace que RTTT (5*) en ese segmento siga siendo el de mejor rendimiento.

- Generalmente, a medida que aumenta el número de viajes anteriores para realizar una predicción, la media de error absoluto y la desviación estándar tienden a estabilizarse y mejorar el rendimiento.

2.3.6. Evaluación del sistema de pruebas

La trayectoria de ida de un viaje en el ambiente de prueba está compuesta por tres estaciones: UENIC, Metrocentro y UCA. De manera similar, la trayectoria

de "regreso" contiene las mismas tres estaciones pero en sentido contrario: UCA, Metrocentro y UENIC. Por lo tanto, se llevó a cabo una evaluación del sistema, analizando cada segmento del trayecto de forma particular, teniendo como referencia las estaciones iniciales de cada trayecto: la estación UENIC en el trayecto de ida y la estación UCA en el trayecto de regreso.

2.3.6.1. Evaluación para viajes con trayectorias de ida.

- **Segmento UENIC → Metrocentro**

En la Figura 74 se presentan los 10 errores de predicción correspondientes al trayecto comprendido entre las estaciones UENIC y Metrocentro, generados por los 10 viajes realizados en las pruebas funcionales de este prototipo, tal como se detalla en la sección 2.3.3. A como se puede observar, todos los errores están dentro del intervalo de [-16, 104.2] segundos y la distribución de los datos sigue una distribución normal truncada a la derecha. Además, al realizar una observación detallada en el histograma de errores de predicción UENIC1-METROC1, se revela que el 70% de los errores de predicción están comprendidos en el intervalo de [-20, 60) segundos.

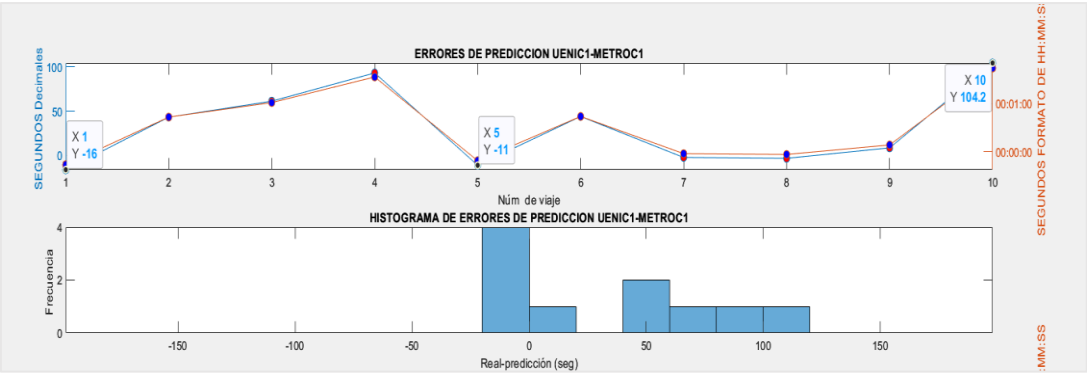


Figura 74. Gráfica de línea y distribución de errores de predicción en el trayecto de ida del segmento UENIC1-METRO1: análisis de 10 viajes en ambiente de prueba [Autor].

- **Segmento Metrocentro → UCA.**

En la Figura 75 se puede observar que todos los errores de la predicción del segmento Metrocentro → UCA están comprendidos en el intervalo de [-120.2, 173.6] segundos. Además, se nota una cierta estabilización del error de

predicción a partir del sexto viaje. Cabe resaltar que el comportamiento de la distribución de los datos se asemeja a una distribución normal.

En este segmento de trayecto, el 60% de los errores se encuentra en el intervalo de [-60, 60) segundos; sin embargo, vemos que el 20% está en el intervalo de [-80, -60] segundos, lo cual permite afirmar que estos errores no discrepan tanto de los otros errores distribuidos en los intervalos multimodales.

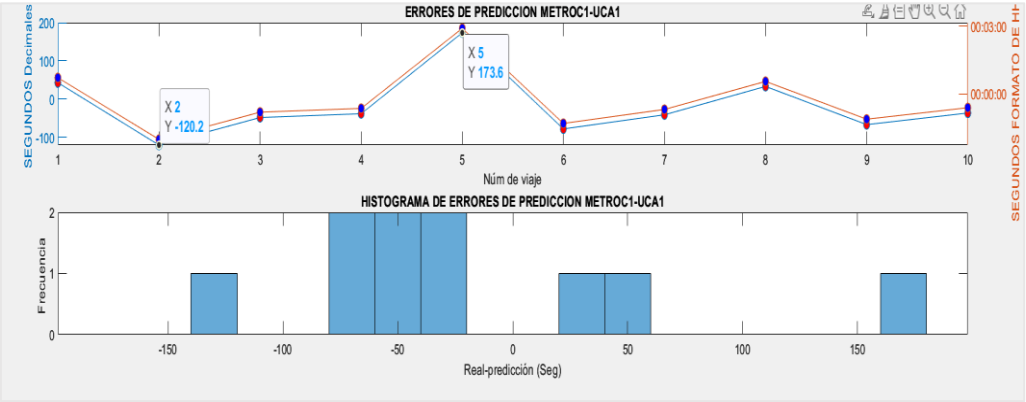


Figura 75. Gráfica de línea y distribución de errores de predicción en el trayecto de ida del segmento METRO1-UCA1: Análisis de 10 viajes en ambiente de prueba [Autor].

2.3.6.2. Evaluación para viajes con trayectorias de regreso.

- **Segmento UCA → Metrocentro**

En la Figura 76 se presentan dos gráficas que muestran el comportamiento de los errores de predicción en los 10 viajes de trayectoria de regreso. En particular, se analiza el segmento UCA → Metrocentro, donde los errores de predicción oscilan entre [-33.2, 124] segundos, y se observa que el 90% de estos errores se encuentran dentro del intervalo [-60, 60) segundos. Es interesante destacar que el único error de predicción fuera de este intervalo corresponde al primer viaje y es generado por la predicción por defecto para el inicio del sistema.

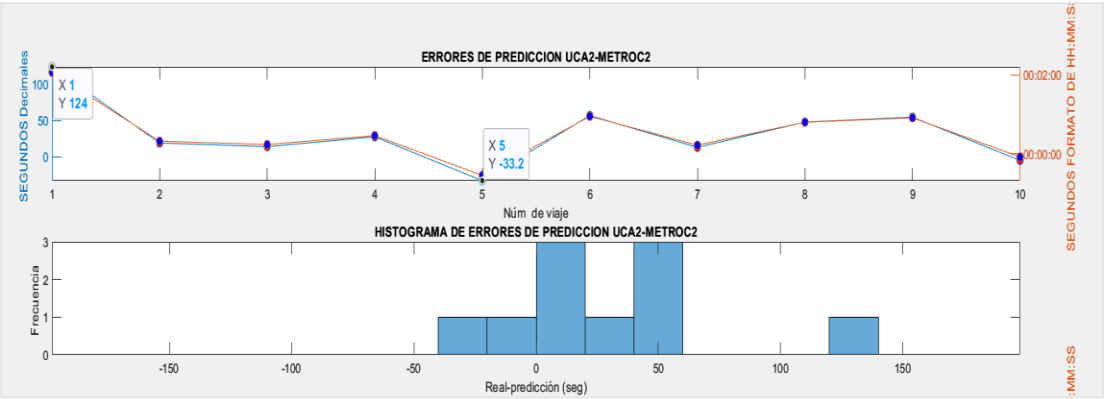


Figura 76. Gráfica de línea y distribución de errores de predicción en el trayecto de regreso del segmento UCA2-METROC2: Análisis de 10 viajes en ambiente de prueba [Autor].

- **Segmento Metrocentro → UENIC.**

Finalmente, en la Figura 77 se presentan las gráficas correspondientes a los 10 errores de predicción del último segmento de una vuelta completa, es decir, el segmento Metrocentro → UENIC. En este trayecto, los errores de predicción se encuentran entre [-101, 82.4] segundos, lo que genera un histograma que se asemeja a una distribución normal. Es importante destacar que el 80% de los errores de predicción se encuentra en el intervalo de [-20, 40) segundos, lo que indica una mayor precisión en la predicción en este segmento en particular.

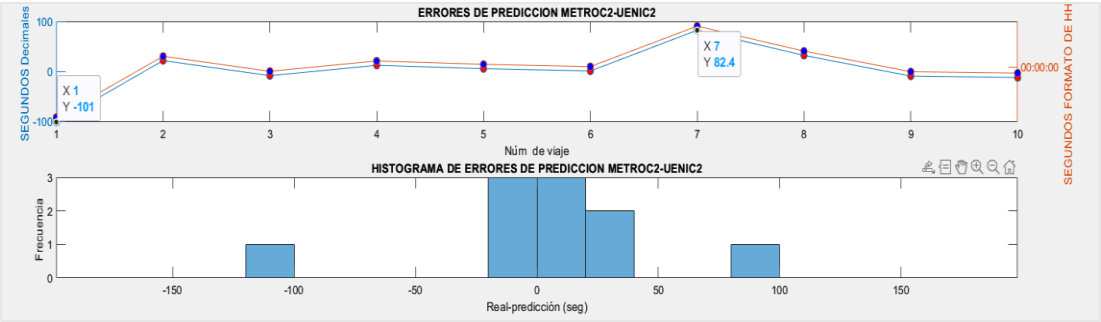


Figura 77. Gráfica de línea y distribución de errores de predicción en el trayecto de regreso del segmento METRO2-UENIC2: análisis de 10 viajes en ambiente de prueba [Autor].

Con el fin de resumir la evaluación completa de las predicciones realizadas por el algoritmo basado en RTTT, se presenta la Tabla 5, que muestra la frecuencia de los errores de predicción agrupados en los intervalos propuestos. Esta información permite una comprensión detallada del desempeño del algoritmo y su capacidad para predecir con precisión en diferentes rangos de valores

Tabla 5. Resumen de errores de predicción en intervalos.

Intervalos (dado en segundos)	**Frecuencia**	**Porcentaje relativo**
[-180 -60)	4	10%
[-60 60)	30	75%
[60-180)	6	15%
Total	40	100%

2.3.6.3. Cantidad de gateways para la cobertura total de Managua

Para determinar la cantidad de puertas de enlace necesarias para brindar cobertura en la red LoRaWAN a las vías de Managua, que reciben el servicio de TUC, es necesario hacer un análisis profundo debido a que en cada lugar se debe evaluar cómo se comporta la comunicación LoRa, al tenerse distintas condiciones en cada sitio de la ciudad. Sin embargo, en este análisis se hará una sencilla aproximación basada en el área aproximada de cobertura que se logró con este prototipo, a partir de la cual se aproximará la cantidad de puertas de enlace necesaria para cubrir las zonas de Managua que disponen del servicio de buses.

Partiendo del hecho de que el gateway utilizado en este prototipo fue colocado en un punto estratégico para cubrir la pista Juan Pablo II, y que según las pruebas de alcance detalladas anteriormente se cubre un área de casi 2 Km^2 , se puede aproximar que la cantidad de gateways para el área de Managua por la cual circulan buses de TUC (aproximadamente 70 Km^2 según mediciones de área realizadas en Google Earth) es de 35 unidades. También se deben ubicar puertas de enlace, además de las 35 unidades calculadas, para crear zonas de solapamiento o redundancia en caso de que un alguna de las unidades deje de operar, y esta cantidad va a depender de varios factores como la topografía del área y la densidad de edificios y árboles, por lo que el número de puertas de enlace aumentaría a 50 aproximadamente.

Se debe tomar en cuenta que esta cantidad aproximada se ha realizado con una simple división, sin embargo, esta cantidad podría disminuir o aumentar al realizarse un estudio que tome en cuenta otros factores para una implementación total en el casco urbano de Managua.

2.3.7. Costo del prototipo de sistema

Los principales componentes y dispositivos que se utilizaron para la implementación de este prototipo fueron importados, ya que en el territorio nacional no se encuentran disponible. En cambio, los componentes más básicos tales como: resistores, botones, interruptores y LED se adquirieron en el mercado nacional. En la Tabla 6 se muestra el costo en dólares de los componentes importados y su equivalencia en córdobas. La tasa de cambio del córdoba respecto al dólar que se usó fue de 36.17 córdobas.

Tabla 6. Costos de elaboración del prototipo.

Artículo	Cantidad	Precio unitario [$]	Precio unitario [C$]	Importe [C$]
Compras importadas.				
Módulo Adafruit LoRa RFM95	2	19.95	721.5915	1,443.183
Módulo GPS GY'NEO6MV2 + Antena cerámica	2	8.99	325.1683	650.3366
Antena 5 dBi 915MHz	2	6.5	235.105	470.21
Arduino Nano	2	8.33	301.2961	602.5922
Gateway LPS8N	1	179	6,474.43	6,474.43
Raspberry Pi 3B+	1	119	4,304.23	4,304.23
Compras en el país.				
Batería 3.7V 2500mAh + Placa de carga y regulación	2		176.4	352.8
Tarjeta de baquelita	2		65	130
Pines hembra 40 pines	4		15	60
Resistencias	4		4	16
Diodo Emisor de Luz (LED)	4		5	20
Interruptor deslizante	2		20	40
Botón pulsador	2		50	100
Terminal Blocks	4		12	48
Insumo de fabricación varios				350
Envío				1,400
Mano de obra				4,000
Total				20,461.7

CAPÍTULO III. CONCLUSIONES Y RECOMENDACIONES

3.1. Conclusiones

Al finalizar este trabajo monográfico se logró desarrollar e implementar un prototipo de sistema para la localización y predicción en tiempo real de la llegada de unidades de transporte colectivo mediante el uso de tecnología LoRa-GPS en los nodos de la red LoRaWAN. Para la verificación del funcionamiento se realizaron pruebas abarcando seis estaciones reales pertenecientes al sistema de transporte público de la ciudad de Managua, con tres estaciones en el trayecto de ida y tres estaciones en el de regreso, en la Pista Juan Pablo II.

Las siguientes conclusiones se basan en los objetivos específicos que se plantearon para lograr el objetivo general de este trabajo monográfico.

- Se diseñaron los nodos LoRa-GPS, seleccionando el módulo GPS GY-NEO6MV2, capaz de ubicar con una precisión más que aceptable, y el transceptor LoRa RFM95W capaz de tener una comunicación de hasta 1.5 Km en el área geográfica de las pruebas presentada en este proyecto monográfico.

- Se elaboró el diagrama de interconexiones y las configuraciones de comunicación inalámbrica para la transmisión y recepción de datos entre el nodo de LoRa-GPS y gateway LPS8N, logrando una efectividad en el envío y recepción del 100% durante las pruebas en el área con buena señal.

- Se implementó el algoritmo de predicción basado en RTTT en Matlab Analysis de ThingSpeak adaptado para la aplicación en el ambiente de prueba presentado en este trabajo monográfico, resultando que el 75% de todos los errores de predicción se encuentran en el intervalo de ±60 segundos.

- El sistema de visualización desarrollado, utilizando software libre, permite a los usuarios finales obtener información sobre la llegada y ubicación de las unidades de transporte urbano a través de una interfaz dinámica e intuitiva proyectada en una pantalla.

- Se comprobó satisfactoriamente el óptimo rendimiento de la comunicación a las velocidades experimentadas durante las pruebas, ya que el envío de los paquetes se realizaba mientras la unidad de transporte reducía su velocidad al

ingresar a una bahía de abordaje de pasajeros. Además, se realizó la prueba del botón de emergencia con el vehículo en movimiento a una velocidad aproximada de 40 Km/h, dado que la autopista estaba libre, recibiéndose exitosamente el paquete la primera vez que se envió. Es importante destacar que las velocidades alcanzadas durante la prueba se encuentran en conformidad con la ley de tránsito, la cual establece un límite de velocidad de 45 Km/h.

- Las pruebas realizadas en condiciones reales, como tráfico, pistas, estaciones y tiempos de espera del sistema de transporte público de la ciudad de Managua validan el correcto funcionamiento de la red LoRaWAN con nodos LoRa-GPS.

3.2. Recomendaciones

El prototipo de sistema para la localización y predicción de arribo en tiempo-real para unidades de transporte colectivo utilizando tecnología LoRa-GPS en Managua, Nicaragua, funciona correctamente. Sin embargo, es necesario implementar algunas mejoras que ayuden a incrementar el nivel de alcance, y de esta manera aspirar a un sistema que dé cobertura a todo el sistema de trasporte público de Managua. Tales recomendaciones son las siguientes:

- Agregar notificaciones a la interfaz del sistema, con el fin de informar a los administradores y usuarios cuando una unidad de TUC se desvía de su ruta planificada. Esto permitiría una rápida respuesta ante posibles incidentes y asegurar que el servicio se brinde de manera efectiva y segura.

- Se propone ampliar la red presentada en este prototipo mediante la adición de más puertas de enlace LoRaWAN. Se recomienda adquirir puertas de enlace para exteriores con disponibilidad de conexión a la red móvil para el acceso a internet, las cuales incluyen antenas con mayor ganancia y son más adecuadas para aplicaciones en exteriores como la presentada. Pueden explorarse puertas de enlace outdoor del mismo fabricante (Dragino), como los modelos: DLOS8N, HP0A o HP0D; o considerarse otros fabricantes.

- Realizar un análisis del comportamiento de LoRa en diferentes escenario de la ciudad de Managua y así determinar con mejor precisión la cantidad de puertas de enlace necesarias para una implementación del sistema.

- Se propone ubicar las puertas de enlace en lugares estratégicos con buena altura a lo largo de las pistas; lo que minimizará las pérdidas por obstrucciones y mejorará la comunicación a mayores distancias, de tal manera que se puedan aumentar el número de estaciones con cobertura por un solo gateway y así minimizar los costos y el número de estos en la red.

- Para ampliar la funcionalidad del sistema presentado se propone generalizar su implementación para cubrir todas las estaciones de cualquier ruta dentro del sistema de transporte público de la ciudad de Managua. De ampliarse este sistema también se recomienda mejorar el diseño de los dispositivos finales incorporando antenas GPS y LoRa para exteriores.

- Se sugiere realizar pruebas de estrés al gateway del sistema. Esta medida proactiva garantizará la robustez y rendimiento de la puerta de enlace en escenarios desafiantes donde se tenga un número considerable de nodos en la red transmitiendo simultáneamente. Al realizar estas pruebas se podrá evaluar el desempeño de las puertas de enlaces recomendadas y de ser necesario se deberán considerar nuevas alternativas de gateways con mejores capacidades de tráfico simultáneo; de esta manera se podrán prevenir fallas futuras y así se brindará una experiencia confiable a los usuarios finales.

- Tomando en cuenta que para este prototipo se empleó la banda ICM de US915 MHz, y sabiendo que al trabajar en una banda de frecuencia menor se puede obtener un mayor alcance, aunque con un ancho de banda menor; se recomienda explorar la banda de 433 MHz para dar continuidad a este sistema u otra aplicación, considerando que deben utilizarse los equipos LoRa y librería LoRaWAN adecuados para esta frecuencia.

Se prevé que trabajando en esta frecuencia la cantidad de puertas de enlace para cubrir toda el área de Managua que recibe el servicio de TUC disminuye considerablemente y con esto los costos de una implementación completa.

- Se propone utilizar está tecnología de comunicación inalámbrica de largo alcance en proyectos o trabajos monográficos que incluyan otras aplicaciones que requieran una tasa de datos baja, como: sistemas de monitoreo, alarmas, rastreadores, agricultura inteligente, o cualquier otra aplicación en la que se envíen paquetes pequeños considerando las limitantes de ancho de banda.

BIBLIOGRAFÍA

Mazidi, M., McKinlay, R. y Causey, D. (2008). PIC Microcontroller and Embedded Systems. New Jersey, United States: Pearson Prentice Hall.

Alzamora, P. y Bautista, A. (2010, enero). Control y monitorización del recorrido de los buses de transporte público mediante tecnología GPS y GSM. Universidad Politécnica Salesiana. Recuperado de: http://dspace.ups.edu.ec/handle/123456789/2356

Arduino Reference. (s.f.). MCCI LoRaWAN LMIC library. Recuperado de: https://www.arduino.cc/reference/en/libraries/mcci-lorawan-lmic-library/

Arduino. (s.f.). Arduino Nano. Arduino Store. Recuperado de: https://store.arduino.cc/products/arduino-nano

Asociación Mundial de la Carretera (PIAR). (s.f.). Sistemas inteligentes de transporte - Conceptos básicos. Recuperado de: https://rno-its.piarc.org/es/conceptos-basicos-its

Banco Interamericano de desarrollo (BID). (2021, abril). Sistemas Inteligentes de Transporte (ITS). Recuperado de: https://www.iadb.org/es/transporte/sistemas-inteligentes-de-transporte-its

Cano, K. y Hernández, J. (2019, septiembre). Propuesta de un sistema de notificación de paradas para una ruta 106 del transporte colectivo de Managua. Recuperado de: https://repositorio.unan.edu.ni/13673/

Cruz, D y Muñoz, C. (2022, marzo). Sistema Inteligente de Monitoreo en Tiempo Real, de Arribo por Parada y por Unidad del Transporte Urbano Colectivo (TUC) de Managua, Nicaragua. Recuperado de: http://ribuni.uni.edu.ni/4653/

Dragino. (2022, julio). LPS8N LoRaWAN Gateway Manual. Recuperado de: http://wiki.dragino.com/xwiki/bin/view/Main/User%20Manual%20for%20All%20Gateway%20models/LPS8N%20-%20LoRaWAN%20Gateway%20User%20Manual/

Ebyte. (2020). E22-900M30S User Manual SX1268 868/915MHz 1 W SPI LoRa module. Recuperado de: https://www.ebyte.com/en/downpdf.aspx?id=453

ECDA (El cajón de Arduino). (2020). Arduino Nano: Todo lo que necesitas saber. Recuperado de: https://www.elcajondeardu.com/2020/11/arduino-nano-todo-lo-que-necesitas-saber.html

El Nuevo diario. (2017). Modernización del Transporte Público. Recuperado de: https://www.elnuevodiario.com.ni/nacionales/managua/438316-modernizacion-transporte-publico-demanda-reorganiz/

Geek. (2023). RFM95W LoRa Radio 915 MHz Adafruit .Recuperado de: https://www.geekfactory.mx/tienda/modulos/radiofrecuencia/rfm95w-lora-radio-915-mhz-modulo-transceptor-adafruit/

GSMA (Representing the worldwide mobile communications). (2019). Mobile Technology GSM. Recuperado de: https://www.gsma.com/aboutus/gsm-technology

HOPERF. (2018). RFM95W: LoRa Module. Recuperado de: https://www.hoperf.com/modules/lora/RFM95.html

Instituto Nacional de Información de Desarrollo (INIDE). (2017). Anuario estadístico 2017. Recuperado de: https://www.inide.gob.ni/docs/Anuarios/Anuario2017.pdf

Keepcoding. (2023, marzo). ¿Qué es Tkinter? Recuperado de: https://keepcoding.io/blog/que-es-tkinter/

La Gaceta, Diario Oficial. (2021, julio). Acuerdo Administrativo N.º 002-2021: Cuadro Nacional de Atribución de Frecuencias (CNAF). Recuperado de: https://www.pgr.gob.ni/PDF/2021/GACETA/GACETA_21_07_2021.pdf

La Gaceta, Diario Oficial. (2022, febrero). Ley para el Régimen de Circulación Vehicular e Infracciones de Tránsito. Recuperado de: http://legislacion.asamblea.gob.ni/normaweb.nsf/b92aaea87dac762406257265005d21f7/dddcd831d507891d06258844005a7f39?OpenDocument

Liando J., Gamage, A., Tengourtius, A., y Li, M. (2019). Known and Unknown Facts of LoRa: Experiences from a Large-scale Measurement Study. Recuperado de: https://dr.ntu.edu.sg/handle/10356/142869

LoRa Alliance. (2017). LoRaWAN. Recuperado de https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/

LoRa Alliance. (2022, noviembre). What is LoRaWAN® Specification. LoRa Alliance. Recuperado de: https://lora-alliance.org/about-lorawan/

Loutos, G. (2013). Development of Prediction Schemes for Real-time Bus Arrival Information. Recuperado de: https://www.diva-portal.org/smash/get/diva2:732548/FULLTEXT01.pdf

Normas Jurídicas de Nicaragua. (2005). Ley General de Transporte Terrestre. Recuperado de: http://legislacion.asamblea.gob.ni/normaweb.nsf/3133c0d121ea3897062568a1005e0f89/dd843b119b6029510625712300595 5cb?OpenDocument

Novas, D. (2008). Microcontroladores Arquitectura, programación y aplicación. Recuperado de:

https://www.aiu.edu/applications/DocumentLibraryManager/upload/Despradel%20Novas%20Pe%C3%B1a.pdf

Petäjäjärvi, J., Mikhaylov, K., Pettissalo, M., Janhunen, J. y Iinatti, J. (2017). Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage. Recuperado de: https://journals.sagepub.com/doi/full/10.1177/1550147717699412

Pozo, A., Ribeiro, A., García, M., García, L., Guinea, D. y Sandoval, F. (s.f.). SISTEMA DE POSICIONAMIENTO GLOBAL (GPS): DESCRIPCIÓN, ANÁLISIS DE ERRORES, APLICACIONES Y FUTURO. Recuperado de: https://www.peoplematters.com/Archivos/Descargas/GPS.pdf

Pradeeka, S. (2019). Beginning LoRa Radio Networks with Arduino Build Long Range, Low Power Wireless IoT Networks. Recuperado de: https://doi.org/10.1007/978-1-4842-4357-2

Quintero, J. y Prieto, L. (2015, marzo). Sistemas inteligentes de transporte y nuevas tecnologías en el control y administración del transporte. Recuperado de: https://repository.upb.edu.co/bitstream/handle/20.500.11912/7281/SISTEMAS%20INTELIGENTES%20DE%20TRANSPORTE.pdf?sequence=1&isAllowed=y

Raspberry Pi. (s.f.). ¿Qué es una Raspberry Pi? Recuperado de: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/

Raspberry Pi. (s.f.). Raspberry Pi 3 Model B+. Recuperado de: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/

Rodríguez, J. y Rojas, J. (2021). Diseño de un Nodo LoRa-GPS para la localización de bicicletas implementado en una Red LoRaWAN. Recuperado de: https://repository.usta.edu.co/handle/11634/35661

Sánchez, J. (2002). Derecho Administrativo y la Regulación del Transporte Urbano Colectivo. Managua, Nicaragua: Monografía.

The Things Industries. (2023). ABP vs OTAA. Recuperado de: https://www.thethingsindustries.com/docs/devices/abp-vs-otaa/

The Things Network. (s.f). Documentation LoRaWAN. Recuperado de: https://www.thethingsnetwork.org/docs/lorawan/

ThingSpeak. (2019). About ThingSpeak. Recuperado de: https://thingspeak.com/pages/learn_more

Ublox. (s.f.). Datasheet u-blox 6 GPS Modules. Recuperado de: https://content.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf

Wenner, R. (2017*). LoRa CHIRP* [Archivo de Vídeo]. YouTube. Recuperado de: https://youtu.be/dxYY097QNs0

ANEXOS

1. Código de nodo LoRa-GPS

```
/*******************************************************************************
 * Copyright (c) 2015 Thomas Telkamp and Matthijs Kooijman
 * Permission is hereby granted, free of charge, to anyone
 * obtaining a copy of this document and accompanying files,
 * to do whatever they want with them without any restriction,
 * including, but not limited to, copying, modification and redistribution.
 * NO WARRANTY OF ANY KIND IS PROVIDED.*/
/* Modify by Jehú Guerrero and Bryan Tinoco*/


#include <lmic.h>
#include <hal/hal.h>
#include <TimerOne.h>
#include <SPI.h>
#include <SoftwareSerial.h>
#include <TinyGPS.h>

TinyGPS gps;
SoftwareSerial ss(3, 4);  // Arduino (RX, TX)

bool aux;
static void smartdelay(unsigned long ms);
float calculodistancia(float latitude, float longitude);

unsigned long age;
float flat, flon, distancia, distanciaact;
const char LED = A5;         ////////LED
const char EMERGENCIA = A4;  ////////Boton de emergencia
bool emer = false;

/*----Coordenadas de las estaciones de prueba real---*/
float latitudestop[] = { 12.131497, 12.130445, 12.130101, 12.129035,
12.127333, 12.125437};
float longitudestop[] = { -86.261631, -86.261165, -86.265145, -86.264633, -
86.271614, -86.272470 };

// -- -Mensajes que se enviarán-- -  //
int i = 0;

char* bus_stop[] = { "STOPA1", "STOPA2", "STOPB1", "STOPB2", "STOPC1",
"STOPC2"};
char* parada_emer[] = { "STOPP1", "STOPP2" };
char mydata[7] = "*None*";
char cadena_confirmada[7];
char cadena_actual[7];

// LoRaWAN NwkSKey, network session key
static const PROGMEM u1_t NWKSKEY[16] = { 0xB0, 0x4F, 0xE6, 0xCB, 0xF2,
0xD2, 0x0D, 0x06, 0x7A, 0x18, 0xED, 0x6A, 0x60, 0x09, 0x37, 0xA0 };
```

```
// LoRaWAN AppSKey, application session key
static const u1_t PROGMEM APPSKEY[16] = { 0xA9, 0xC8, 0x2B, 0x59, 0xF6,
0xFB, 0x2E, 0x83, 0x1F, 0x3E, 0xC7, 0xB4, 0x2E, 0x47, 0x82, 0xB8 };

// LoRaWAN end-device address (DevAddr)
static const u4_t DEVADDR = 0x260CC031;

// These callbacks are only used in over-the-air activation, so they are
left empty here (we cannot leave them out completely unless
void os_getArtEui(u1_t* buf) {}
void os_getDevEui(u1_t* buf) {}
void os_getDevKey(u1_t* buf) {}

static osjob_t sendjob;

// Programacion de envío cada tantos segundos
const unsigned TX_INTERVAL = 12;

// Mapeo de pines de módulo LoRa (pines necesarios para la librería LMIC)
const lmic_pinmap lmic_pins = {
  .nss = 10,
  .rxtx = LMIC_UNUSED_PIN,
  .rst = 9,
  .dio = { 2, 5, 6 },
};

void setup() {

  Serial.begin(9600);
  ss.begin(9600);
  pinMode(LED, OUTPUT);
  pinMode(EMERGENCIA, INPUT_PULLUP);
  Serial.println(F("Starting"));

  //Timer para interrupción temporizada que saltara cada 150 ms
  Timer1.initialize(500000);           // Dispara cada 500 ms. Intervalo de
disparo en microsegundos
  Timer1.attachInterrupt(emergencia);  //Interrupcion temporizada llama a
la funcion emergencia

#ifdef VCC_ENABLE
  // For Pinoccio Scout boards
  pinMode(VCC_ENABLE, OUTPUT);
  digitalWrite(VCC_ENABLE, HIGH);
  delay(1000);
#endif
  // LMIC init
  os_init();
  // Restablecer el estado MAC. Las transferencias de sesión y de datos
pendientes se descartarán.
  LMIC_reset();

#ifdef PROGMEM
  uint8_t appskey[sizeof(APPSKEY)];
  uint8_t nwkskey[sizeof(NWKSKEY)];
  memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
  memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
  LMIC_setSession(0x1, DEVADDR, nwkskey, appskey);
```

```
#else
  LMIC_setSession(0x1, DEVADDR, NWKSKEY, APPSKEY);
#endif
/*Frecuencia utilizada. En este caso se modificó los archivos de la
librería para que sea en la banda de 915MHz*/
#if defined(CFG_eu868)
  LMIC_setupChannel(0, 915000000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);  // g-band
#elif defined(CFG_us915)
  LMIC_selectSubBand(1);
#endif

  // Deshabilitar la validación de verificación de enlaces
  LMIC_setLinkCheckMode(0);
  // TTN usa SF9 para su ventana RX2.
  LMIC.dn2Dr = DR_SF9;
  //Configurando Data rate y la potencia de transmisión para el enlace
ascendente (nota: la biblioteca parece ignorar txpow)
  LMIC_setDrTxpow(DR_SF7, 14);

  //Iniciar trabajo de envío
  do_send(&sendjob);
  unsigned long start = millis();
  do {
    digitalWrite(LED, HIGH);
    os_runloop_once();
  } while (millis() - start < 30000);
  digitalWrite(LED, LOW);
}

void loop() {
  /*Haciendo recorrido de la funcion principal para las 6 estaciones*/
  GPSEstacion(0);  GPSEstacion(1);   GPSEstacion(2);   GPSEstacion(3);   GP
SEstacion(4);   GPSEstacion(5);
}

//----------------------------Funciones-----------------------//

//--Función de interrupción para botón de emergencia--//
void emergencia() {
  //Una vez se presione el botón de emergencia el nodo quedará invalidado y
no podrá seguir enviando. Para reanudar deberá ser apagado
  if (digitalRead(EMERGENCIA) == LOW && aux == false) {
    emer = true;  //Variable para invalidar unidad de transporte en caso de
emergencia
    Serial.println("El bus presentó un problema");
    int ultimo = cadena_actual[strlen(cadena_actual) - 1];  //Obteniendo el
sentido del viaje del paquete de la estación anterior

    Serial.print("El ultimo valor fue:");
    Serial.println(ultimo);
    if (ultimo == 49) {  //El 1 equivale a 49 en decimal
      Serial.println("Para la 1");
      strncpy(mydata, parada_emer[0], 6);
    }
    else if (ultimo == 50) {  //El 2 equivale a 50 en decimal
      Serial.println("Para la 2");
      strncpy(mydata, parada_emer[1], 6);
    }
```

```
    strncpy(cadena_actual, mydata, 6);
    while (strcmp(cadena_confirmada, cadena_actual) != 0) {
      Serial.println("Cadenas diferentes, emergencia...");
      unsigned long start = millis();
      do {
        digitalWrite(LED, HIGH);
        os_runloop_once();
      } while (millis() - start < 10000);
      smartdelay(100);
      aux = true;
      digitalWrite(LED, LOW);
    }
    digitalWrite(LED, LOW);
  }
  else if (digitalRead(EMERGENCIA) == LOW && aux == true){
    digitalWrite(LED, LOW);
  }
}
//----------Función para eventos de envío---------//
void onEvent(ev_t ev) {
  Serial.print(os_getTime());
  Serial.print(": ");
  switch (ev) {
    case EV_SCAN_TIMEOUT:
      Serial.println(F("EV_SCAN_TIMEOUT"));
      break;
    case EV_BEACON_FOUND:
      Serial.println(F("EV_BEACON_FOUND"));
      break;
    case EV_BEACON_MISSED:
      Serial.println(F("EV_BEACON_MISSED"));
      break;
    case EV_BEACON_TRACKED:
      Serial.println(F("EV_BEACON_TRACKED"));
      break;
    case EV_JOINING:
      Serial.println(F("EV_JOINING"));
      break;
    case EV_JOINED:
      Serial.println(F("EV_JOINED"));
      break;
    case EV_RFU1:
      Serial.println(F("EV_RFU1"));
      break;
    case EV_JOIN_FAILED:
      Serial.println(F("EV_JOIN_FAILED"));
      break;
    case EV_REJOIN_FAILED:
      Serial.println(F("EV_REJOIN_FAILED"));
      break;
    case EV_TXCOMPLETE:
      Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));

      if (LMIC.txrxFlags & TXRX_ACK)
        strcpy(cadena_confirmada, mydata);
      Serial.print("ACK para paquete: ");
      Serial.println(cadena_confirmada);

      if (LMIC.dataLen) {
```

```
        Serial.println(F("Received "));
        Serial.println(LMIC.dataLen);
        Serial.println(F(" bytes of payload"));
      }
      // Programando próxima transmisión
      os_setTimedCallback(&sendjob, os_getTime() +
sec2osticks(TX_INTERVAL), do_send);
      break;
    case EV_LOST_TSYNC:
      Serial.println(F("EV_LOST_TSYNC"));
      break;
    case EV_RESET:
      Serial.println(F("EV_RESET"));
      break;
    case EV_RXCOMPLETE:
      // Datos recibidos en la ranura de ping
      Serial.println(F("EV_RXCOMPLETE"));
      break;
    case EV_LINK_DEAD:
      Serial.println(F("EV_LINK_DEAD"));
      break;
    case EV_LINK_ALIVE:
      Serial.println(F("EV_LINK_ALIVE"));
      break;
    default:
      Serial.println(F("Unknown event"));
      break;
  }
}
//-----------Función de envío-----------//
void do_send(osjob_t* j) {
  Serial.print("Enviando: ");
  Serial.println((char*)mydata);
  if (LMIC.opmode & OP_TXRXPEND) {
    Serial.println(F("OP_TXRXPEND, not sending"));
  } else {
    // Preparación de la transmisión de datos se subida para la siguiente
próxima vez
    LMIC_setTxData2(1, mydata, sizeof(mydata) - 1, 1);
    Serial.println(F("Paquete en cola"));
    Serial.print("LMIC.freq:");
    Serial.println(LMIC.freq);
    Serial.println("");
  }
}
//Función de Smartdelay
static void smartdelay(unsigned long ms) {
  unsigned long start = millis();
  do {
    while (ss.available()) {
      gps.encode(ss.read());
    }
  } while (millis() - start < ms);
}
/*Función para el cálculo de distancia de la ubicación a las estaciones*/
float calculodistancia(float latitud, float longitud) {
  gps.f_get_position(&flat, &flon, &age);
  distancia = TinyGPS::distance_between(flat, flon, latitud, longitud);
  return distancia;
```

```
}
//Identificación de estacion y envío
void GPSEstacion(int i) {
  /*Calculando la distancia de ubicación actual a las ubicaciones
prefijadas*/
  distanciaact = calculodistancia(latitudestop[i],
longitudestop[i]);  //Calculando la distancia de ubicación actual a las
ubicaciones prefijadas
  smartdelay(500);
  /*Mientras el bus se encuentre en un radio de 30 metros de cualquiera de
las estaciones y el botón de emergencia no se presione, se enviará el
nombre de la estación actual*/
  while (distanciaact <= 60 && emer == false) {
    strncpy(mydata, bus_stop[i], 6);  //Copiando mensaje de estación actual
en la cadena de envio
    strncpy(cadena_actual, mydata, 6);
    if (strcmp(cadena_confirmada, cadena_actual) != 0) {
      Serial.println("Cadenas diferentes, haciendo envio...");
      digitalWrite(LED, HIGH);
      unsigned long start = millis();
      //Llamando función para el envío
      do {
        os_runloop_once();
      } while (millis() - start < 10000);
    } else if (strcmp(cadena_confirmada, cadena_actual) == 0) {
      digitalWrite(LED, LOW);
      smartdelay(500);
      digitalWrite(LED, HIGH);
      smartdelay(500);
      Serial.println("Esta cadena ya fue enviada");
    }
    distanciaact = calculodistancia(latitudestop[i], longitudestop[i]);
/*Actualizando distancia*/
  }
  if (distanciaact > 60) {
    if (emer == false) {
      digitalWrite(LED, LOW);
      smartdelay(100);
    }
  }
}
```

2. Formato de Uplink en The Things Network

```
function Decoder(bytes, port) {
  // Decode an uplink message from a buffer
  const str = String.fromCharCode.apply(null, bytes);

  const lastChar = str.charAt(str.length-1);
 const result = {};

  if (lastChar === '1') {
    result.field1 = str;
  } else if (lastChar === '2') {
    result.field2= str;
  }

  return result;

}
```

3. **Calidad de señal por RSSI**

Señal excelente: -55 a -70 dBm.

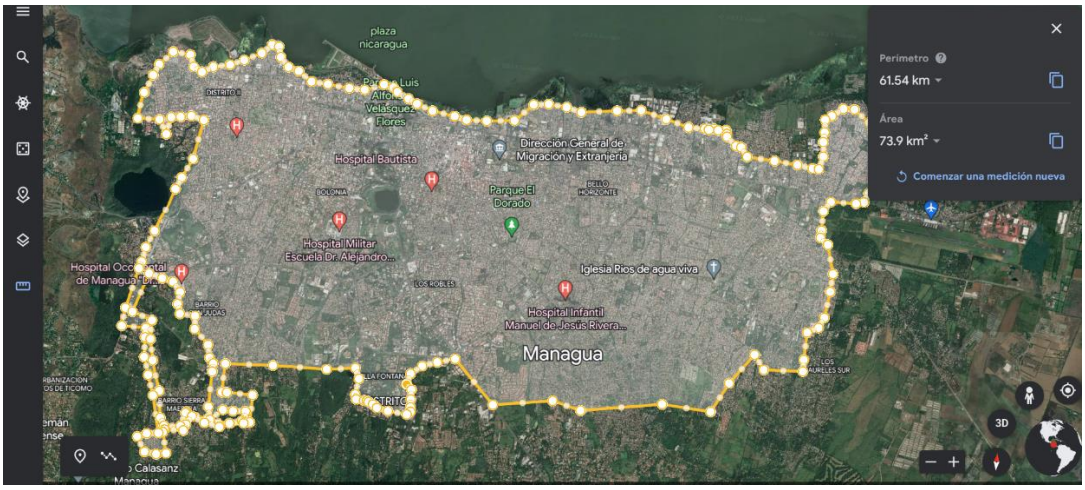
Señal buena: -70 a -85 dBm.

Señal aceptable: -85 a -100 dBm

Señal débil: -100 dBm o menos.

4. **Área cubierta con el prototipo**

5. **Área de Managua con servicio TUC**



6. Función de Matlab para la realización de predicciones

```
% Función principal que realiza las predicciones
function[vectkp1sAsB]=prediccion_alfa(PIksa_alfa,bustrips,Nstop,Nordenstop)
for k=1:Nstop
    Gamma=0;
    tkp1sAsB=0;


    for j=1:(bustrips-1)
        Gamma=Gamma+(1/minutes((PIksa_alfa{Nordenstop,1}(1,bustrips)-
PIksa_alfa{Nordenstop,1}(1,bustrips-j))));
    end
    for j=1:bustrips-1
        tkp1sAsB=
tkp1sAsB+((1/minutes((PIksa_alfa{Nordenstop,1}(1,bustrips)-
PIksa_alfa{Nordenstop,1}(1,bustrips-
j)))/Gamma)*minutes(PIksa_alfa{k+Nordenstop,1}(1,j)-
PIksa_alfa{Nordenstop,1}(1,j));
    end
    vectkp1sAsB(k)=tkp1sAsB;
end
vectkp1sAsB=round(vectkp1sAsB,2);
end
```