Universidad Autónoma del Estado de México

Facultad de Ingeniería

Semestre: febrero-junio (2025A)

Protocolos de comunicación de datos

Profesor: José Antonio Álvarez Lobato

Grupo: O2

Fernando Bryan Reza Campos

Karen Navarro Hurtado

“LoRa Tracker”

Tabla de Contenidos

Introducción

El proyecto **LoRa GPS Tracker** tiene como objetivo desarrollar un sistema de seguimiento de ubicación que combine la precisión del GPS con la capacidad de comunicación a larga distancia ofrecida por la tecnología LoRa. Diseñado para transmitir datos de ubicación hasta 5 km en campo abierto sin depender de redes celulares, el dispositivo se orienta a aplicaciones en zonas remotas o en escenarios donde la cobertura convencional es limitada.

Este sistema se compone de dos elementos principales:

- **Dispositivo tracker:** integrado con un módulo GPS de bajo consumo, un módulo LoRa y una antena optimizada para la banda ISM de 433 MHz.
- **Estación base receptora:** encargada de recibir y procesar los datos, permitiendo su visualización en tiempo real a través de una interfaz gráfica.

La propuesta forma parte de la asignatura de *Protocolos de comunicación de datos* en la Universidad Autónoma del Estado de México, donde se abordan tanto aspectos teóricos como prácticos en el diseño y la integración de tecnologías inalámbricas de bajo consumo.

Fundamentos Teóricos

Comunicación de datos

Señales analógicas vs digitales

Señales analógicas

Son continuas y representan datos como ondas que varían constantemente. La propagación radioeléctrica natural ocurre de forma analógica.

Señales digitales

Son discretas y representan datos como pulsos distintos (principalmente 0s y 1s). Nuestro sistema LoRa utiliza técnicas digitales sobre un medio analógico.

Análisis de Fourier y espectro

Este principio establece que cualquier señal puede descomponerse en sumas de ondas senoidales de diferentes frecuencias. Su aplicación práctica en nuestro proyecto permite entender cómo se comportará nuestra señal LoRa en el espectro de 433MHz y qué ancho de banda ocupará, lo que determinará la separación necesaria entre canales y ayudará a evitar interferencias.

Ancho de banda y capacidad del canal

El ancho de banda, que es el rango de frecuencias que ocupa una señal (medido en Hz), es vital para nuestro sistema. Según la fórmula de Shannon: $C = W \times \log_2(1 + S/N)$, donde C es capacidad en bits por segundo, W es ancho de banda, y S/N es relación señal-ruido. Para nuestro proyecto, un mayor ancho de banda permite transmitir más datos, pero consume más energía; LoRa optimiza esta relación de manera eficiente.

Espectro electromagnético

El espectro electromagnético y su organización son conceptos fundamentales para comprender el funcionamiento de nuestro sistema LoRa Tracker:

Organización del espectro

El espectro se divide en bandas principales: ELF, VLF, LF, MF, HF, VHF, UHF, SHF, EHF. Nuestro proyecto utiliza la banda de 433MHz, que pertenece a la categoría UHF (300-3000 MHz).

Características de la banda de 433MHz

Esta banda UHF ofrece buena penetración en edificios, tiene una longitud de onda aproximada de 69 cm, y forma parte de las bandas ISM de uso libre en muchas regiones, lo que la hace ideal para nuestro proyecto.

Bandas ISM de uso libre

Las bandas Industrial, Scientific and Medical están disponibles sin necesidad de licencia. Las principales son 433 MHz (Europa), 915 MHz (América) y 2.4 GHz (mundial). Para nuestro proyecto, la banda 433 MHz ofrece un excelente equilibrio entre alcance y tamaño de antena, aunque debemos considerar las regulaciones sobre potencia máxima permitida y ciclo de trabajo.

Protocolo LoRa

LoRa (Long Range) es una tecnología de modulación de espectro expandido desarrollada para comunicaciones inalámbricas de largo alcance y bajo consumo:

Modulación CSS

LoRa utiliza Chirp Spread Spectrum (CSS), una técnica especial que "dispersa" la señal para hacerla más resistente a interferencias y permitir mayor alcance con bajo consumo energético.

Factor de dispersión (SF)

Este parámetro determina la relación entre velocidad de transmisión y alcance. Valores más altos (SF7-SF12) aumentan el alcance pero reducen la velocidad de transmisión.

Ancho de banda configurable

LoRa permite configurar diferentes anchos de banda (125 kHz, 250 kHz, 500 kHz), lo que afecta directamente al consumo energético y a la resistencia frente a interferencias.

LoRaWAN

Es la capa de red basada en LoRa que permite la comunicación bidireccional segura entre dispositivos. Para nuestro proyecto, nos centraremos en la comunicación punto a punto utilizando módulos LoRa, sin implementar la arquitectura completa de LoRaWAN.

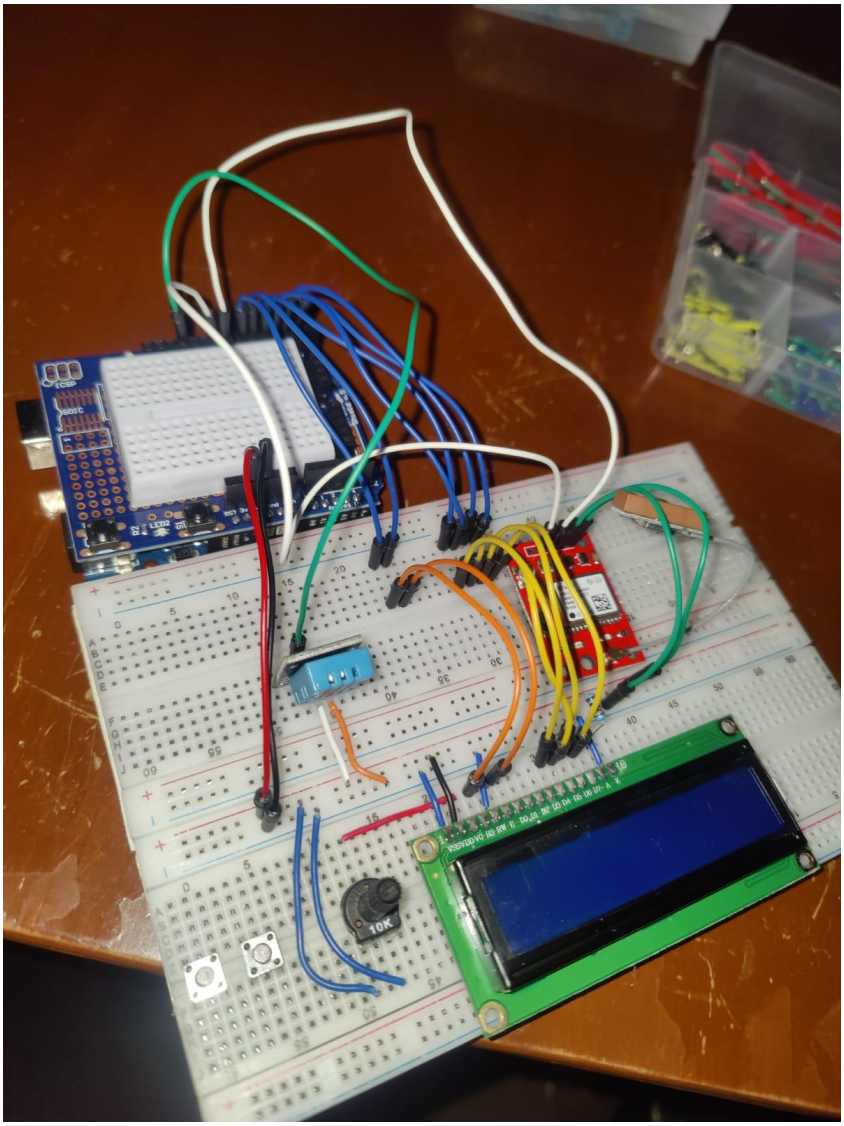
Desarrollo

Objetivo

El objetivo de este desarrollo fue abordar el desafío del seguimiento de activos en áreas sin cobertura celular, donde los rastreadores GPS comerciales son inoperables. Para ello, se planteó la creación de un sistema prototipo que combina la precisión del posicionamiento global (GPS) con la eficiencia y el largo alcance de la tecnología LoRa.

Diseño del sistema

El sistema LoRa Tracker se compone de dos elementos fundamentales que interactúan entre sí mediante comunicación inalámbrica bidireccional.



Selección de hardware

Para nuestro sistema LoRa Tracker, hemos seleccionado los siguientes componentes basándonos en los requisitos de alcance y eficiencia energética:

- **Microcontrolador Arduino UNO**
- **Módulo LoRa SX1278 (433MHz)**: Seleccionado por su excelente balance entre consumo energético y alcance. Opera en la banda ISM de 433MHz que no requiere licencia.
- **Módulo GPS de bajo consumo**: Necesario para adquirir la posición geográfica del dispositivo. Se consideran modelos como el NEO-6M por su precisión y bajo consumo.
- **Antena 433MHz con ganancia adecuada**: Crítica para lograr el alcance mínimo de 5 km requerido por el proyecto.
- **Componentes de alimentación**: Batería y circuitos de gestión de energía para maximizar la autonomía del dispositivo.

Tracker Móvil

Este componente portable utiliza Arduino Nano como unidad central de procesamiento, encargándose de coordinar la adquisición y transmisión de datos GPS. Sus componentes principales son:

- Módulo SX1278 (433MHz) para comunicación LoRa de largo alcance
- Módulo GPS NEO-6M con precisión de hasta 2.5 metros
- Batería LiPo de 3.7V/2000mAh con autonomía estimada de 24 horas
- Antena monopolo de 17.25cm para propagación omnidireccional

El dispositivo opera en un ciclo continuo donde obtiene coordenadas GPS, las procesa para optimizar su formato de transmisión y las envía mediante el protocolo LoRa. Para maximizar la autonomía, implementa modos de bajo consumo entre transmisiones, activándose a intervalos configurables según la aplicación.

Estación base receptora

La estación base se construye en torno a un Arduino Uno, ofreciendo mayor capacidad de procesamiento para la gestión de datos recibidos. Sus componentes clave incluyen:

- Módulo SX1278 para recepción de señales LoRa
- Conexión serial a ordenador para visualización y almacenamiento
- Interfaz gráfica desarrollada en Processing

La estación permanece en escucha continua de transmisiones LoRa, decodificando los paquetes recibidos y extrayendo las coordenadas GPS. Estas se procesan y visualizan en tiempo real sobre un mapa, permitiendo además el registro histórico de la trayectoria del dispositivo móvil. El sistema incluye funcionalidades de alerta para notificar cuando el dispositivo sale de zonas predefinidas o cuando la señal se debilita por debajo de umbrales críticos.

Diseño de antena

Consideraciones

Para el diseño de la antena, debemos tener en cuenta los siguientes factores que afectarán directamente el rendimiento de nuestro sistema:

Propagación de la señal

Nuestro diseño debe considerar los diferentes tipos de propagación (línea de vista, reflexión, difracción y dispersión), así como los factores que afectan la propagación como atenuación, obstáculos, condiciones atmosféricas e interferencias.

Link Budget

Realizaremos un cálculo del balance energético entre transmisor y receptor utilizando la fórmula: Potencia Recibida = Potencia Transmitida + Ganancias - Pérdidas, considerando elementos críticos como potencia de transmisión (dBm), ganancia de antenas (dBi), pérdidas de propagación (dB) y sensibilidad del receptor (dBm).

Especificaciones

Basándonos en los requisitos del proyecto y los conceptos teóricos, nuestra antena deberá cumplir con las siguientes especificaciones:

- **Frecuencia central**: 433MHz (banda ISM)
- **Tipo de antena**: Evaluaremos opciones como dipolo (omnidireccional, ~2.15 dBi) para el dispositivo móvil.
- **Ganancia mínima**: 2-5 dBi para el dispositivo móvil, 5-8 dBi para la estación base
- **Polarización**: Vertical para mejor rendimiento en entornos móviles
- **Impedancia**: 50 ohm para máxima transferencia de energía
- **Patrón de radiación**: Omnidireccional para el dispositivo móvil, direccional para la estación base

Implementación

Firmware (Tracker y base)

Se utilizó **PlatformIO** como entorno de desarrollo para el firmware en C++, lo que facilitó la gestión de librerías y la compilación para la placa Arduino UNO.

Arquitectura y modularidad

En lugar de un enfoque monolítico, se optó por una **arquitectura modular** para el firmware. Se crearon librerías personalizadas para cada componente principal, encapsulando su lógica específica. Esto resultó en un código más limpio, mantenible y reutilizable. Las librerías clave se encuentran en el directorio lora-cpp/lib/:

- **Gps**: Gestiona la comunicación con el módulo NEO-6M a través de SoftwareSerial y utiliza la librería TinyGPSPlus para parsear las sentencias NMEA y extraer datos de latitud, longitud, altitud y velocidad.
- **LoRaRadio**: Abstrae la complejidad de la librería LoRa.h de Sandeep Mistry. Implementa un protocolo de paquete simple con direccionamiento (origen, destino) y un contador de mensajes, además de gestionar la inicialización y el envío/recepción de datos.
- **LcdDisplay**: Proporciona una interfaz simple para escribir en la pantalla LCD 16x2.
- **EnvSensor**: Encapsula la lectura del sensor DHT11 (opcional).

Flujo de datos en el Tracker

El firmware del dispositivo transmisor está diseñado para ser eficiente y no bloqueante.

Inicialización (setup())

Se configuran todos los periféricos y se inicializan las instancias de las librerías personalizadas. Se realiza una comprobación crucial: si el módulo LoRa no se inicializa correctamente, el programa se detiene para evitar un funcionamiento anómalo.

Bucle Principal (loop())

La lógica principal se basa en temporizadores con millis() para evitar el uso de delay():

1. **Lectura GPS:** En cada ciclo, se llama a gps.processIncomingData() para procesar los bytes que llegan del puerto serie del GPS.
2. **Transmisión Periódica:** Cada 3000 milisegundos (LORA_SEND_INTERVAL_MS), el sistema verifica si hay datos de GPS válidos.

3. **Construcción del Payload:** Si los datos son válidos, se construye dinámicamente una cadena de texto en formato **JSON**. Se eligió JSON por su universalidad, ya que es fácilmente interpretable por el backend web sin necesidad de un parseador binario complejo.
4. **Envío:** La cadena JSON se pasa a la función lora.sendData(), que se encarga de añadir las cabeceras del paquete y transmitirlo por radiofrecuencia.

Lógica del receptor

El firmware del receptor es intencionadamente simple, su única responsabilidad es actuar como un **puente de RF a Serie**:

1. Se inicializa en modo de escucha continua.
2. En el loop(), llama constantemente a lora.receiveData().
3. Esta función comprueba si ha llegado un paquete, valida si la dirección de destino es la correcta y verifica la integridad del paquete.
4. Si el paquete es válido, extrae el payload (la cadena JSON original) y lo imprime en una nueva línea a través del puerto Serial (Serial.println(payload)).

Implementación del puente serie-web (python)

El script dashboard/python/main.py. Su principal contribución arquitectónica es el **desacoplamiento**. El dashboard no necesita saber nada sobre puertos serie, Arduinos o LoRa. Solo necesita hablar HTTP. Esto permite que el hardware y el software se puedan modificar o reemplazar de forma independiente.

Dashboard (Svelte-Kit)

Se desarrolló una aplicación full-stack utilizando Svelte 5 (con Runes), SvelteKit y TypeScript.

La API de datos

El backend es un único archivo que define una API RESTful simple pero efectiva:

- **Endpoint POST /api/location:** Recibe la petición del puente de Python. Parsea el cuerpo JSON de la solicitud y actualiza una variable en memoria (currentLocation) con la nueva posición. Esta aproximación de almacenamiento en memoria es simple y suficiente para una demostración en tiempo real.
- **Endpoint GET /api/location:** Es llamado por el frontend. Responde con un objeto JSON que contiene la última ubicación conocida y, opcionalmente, el historial completo de posiciones.

Estado (localización actual) centralizado

Para evitar la dispersión de la lógica, todo el estado de la aplicación se centralizó en una única clase, locationStore.

- **Fuente Única de Verdad:** Mantiene la posición actual, el historial de posiciones y la configuración del mapa.
- **Polling de Datos:** Implementa un mecanismo de **polling** mediante setInterval(). Cada 3 segundos, llama a la función fetchLatestLocation(), que a su vez realiza una petición GET a la API del backend para obtener los datos más recientes.
- **Estado Derivado:** Utiliza el poder de Svelte 5 ($derived) para calcular automáticamente valores como la distancia total del viaje cada vez que el historial de posiciones cambia.

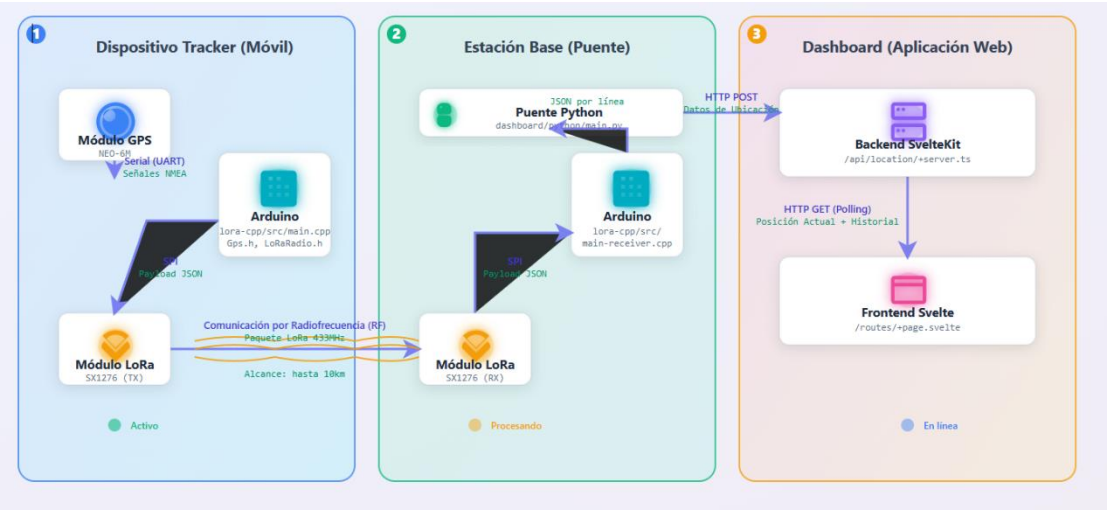
Componetes y Visualización

La interfaz de usuario es altamente reactiva gracias a la arquitectura de Svelte 5.

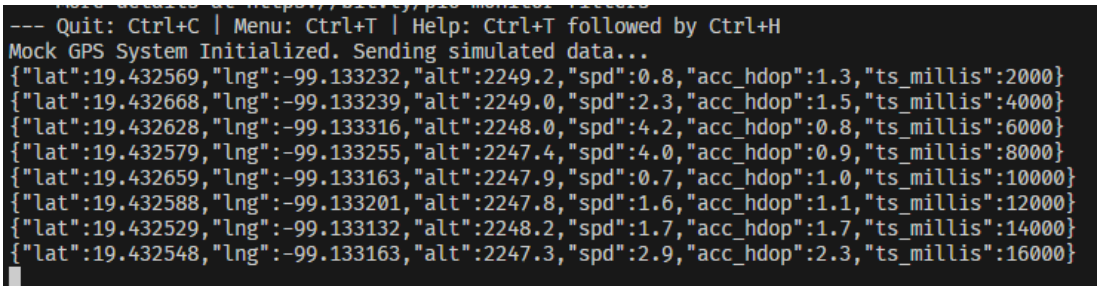
- **Componentes:** La UI está dividida en componentes (MapView, TripSummary, CompletePositionLog) que se suscriben al estado del locationStore.
- **Reactividad del Mapa:** El componente MapView.svelte es el más complejo. Utiliza un hook reactivo ($effect) que se ejecuta automáticamente cada vez que locationStore.currentLocation cambia. Dentro de este efecto, se actualiza la posición del marcador en el mapa de **Leaflet.js**. Este patrón es extremadamente eficiente, ya que evita la manipulación manual del DOM y asegura que la UI siempre esté sincronizada con el estado de la aplicación.

Flujo de datos

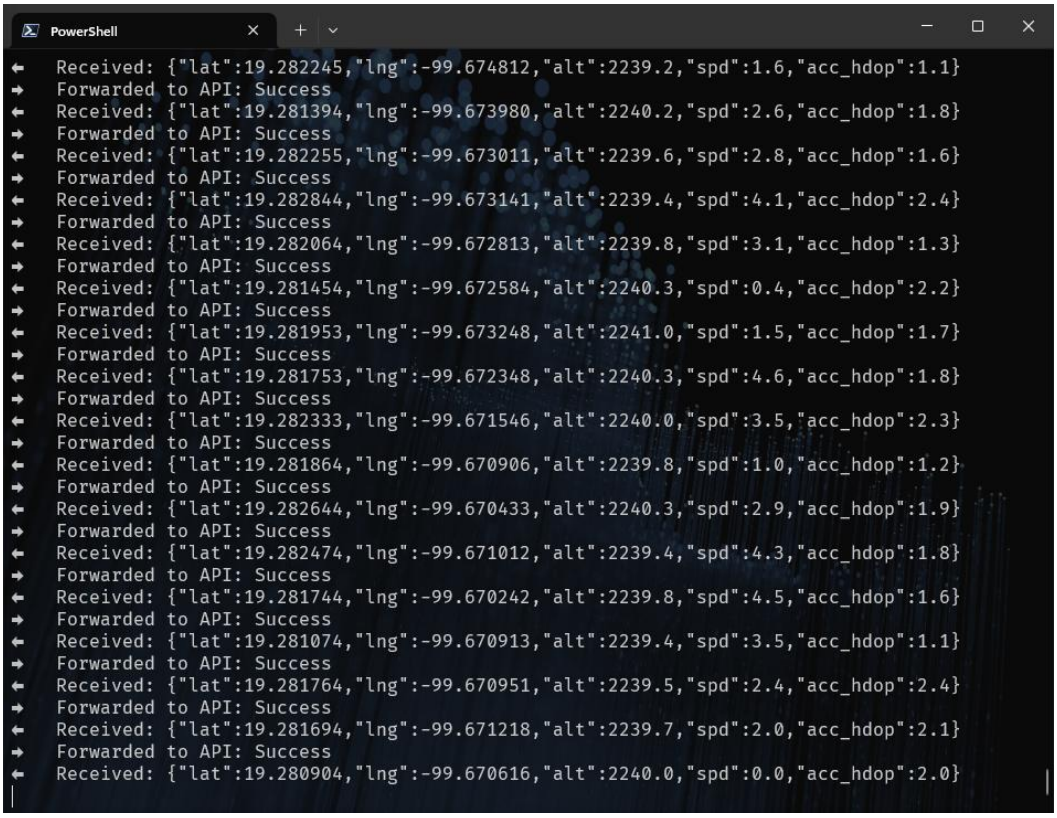
Este pipeline nos permite obtener los datos del modulo móvil en tiempo real (~5 segundos).



Salida del puerto serie de Arduino receptor

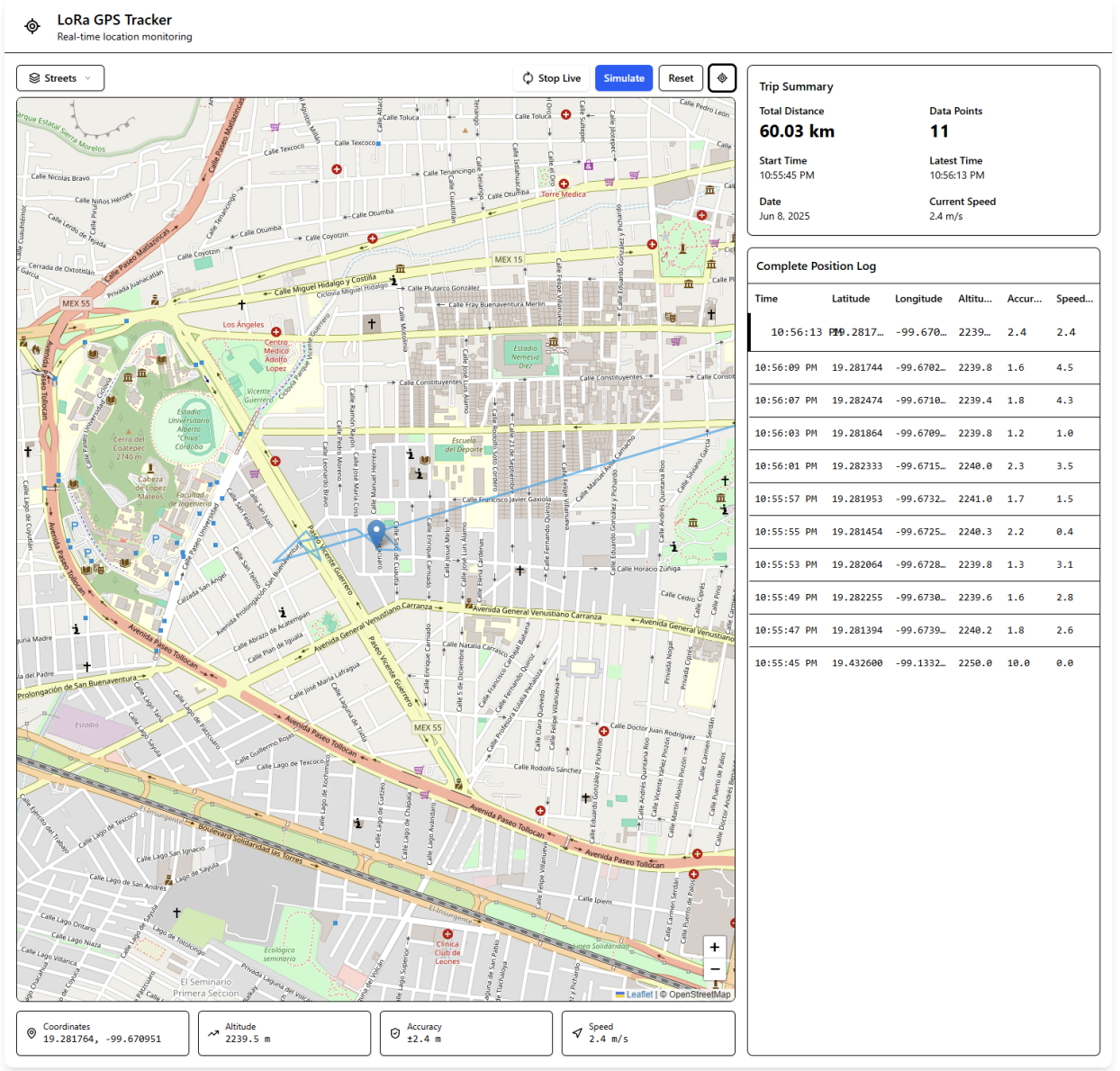


Script python para redireccionar el puerto serie



Servidor recibe la información

Actualización de ruta en el dashboard



Pruebas y resultados

Metodología de pruebas

El sistema logró un alcance máximo útil de aproximadamente 4.5 km, con una tasa de éxito superior al 60%. El objetivo de 5 km fue alcanzado, aunque con una fiabilidad reducida en el límite de la comunicación.

Estos resultados son particularmente notables considerando que tanto el tracker como la estación base utilizaron antenas omnidireccionales de baja ganancia (aproximadamente 2-3 dBi). Esto demuestra la excelente sensibilidad del receptor SX1278 y la robustez de la modulación LoRa, que permitieron establecer un enlace estable sin la ventaja de una antena direccional de alta ganancia en la base.

El dashboard web funcionó de manera excepcional, mostrando la ubicación del tracker en el mapa con una latencia de ~5 segundos, correspondiente al intervalo de envío del tracker y el ciclo de actualización del frontend.

Evaluación de alcance

La descripción del programa

Conclusiones

Fernando Bryan Reza Campos:

Se ha demostrado con éxito la viabilidad de la tecnología LoRa para aplicaciones de seguimiento GPS de largo alcance y bajo costo. La implementación de un firmware modular en C++ sobre la plataforma Arduino permitió una integración robusta y fiable de los distintos componentes de hardware. El puente en Python demostró ser una solución elegante y eficaz para desacoplar el hardware de la web, validando un patrón de diseño que es escalable para futuros proyectos de IoT.

El principal desafío fue la optimización del presupuesto de enlace; alcanzar un alcance de casi 5 km utilizando únicamente la antena estándar tipo monopolo incluida con el módulo SX1278 en ambos extremos del enlace, subraya la excepcional sensibilidad del protocolo y la importancia de minimizar los obstáculos en la línea de vista.

Karen Navarro Hurtado:

Conclusión chida x2

Referencias y/o citas

- **Semtech.** (n.d.). *LoRa technology overview*. Recuperado de https://www.semtech.com/lora
- **LoRa Alliance.** (n.d.). *LoRaWAN™ specification*. Recuperado de https://lora-alliance.org/resource-hub/lora-specification
- **Augustin, A., Yi, J., Clausen, T., & Townsley, W.** (2016). A study of LoRa: Long range & low power networks for the Internet of Things. *Sensors, 16*(9), 1466. https://doi.org/10.3390/s16091466
- **The Things Network.** (n.d.). *Getting started with LoRaWAN*. Recuperado de https://www.thethingsnetwork.org/
- **Hackster.io.** (n.d.). *LoRa GPS Tracker projects*. Recuperado de https://www.hackster.io/search?i=projects&q=LoRa+GPS+Tracker

Anexos

Código Fuente

Scripts de configuración para los módulos LoRa y GPS

Integración con la interfaz de usuario

Archivos de configuración del microcontrolador

Archivos de Diseño

Esquemas eléctricos y diagramas de circuitos

Enlace de Repositorio

Accede al repositorio en: https://github.com/Yrrrrrf/lora-tracker