LoRa-Based Wireless Localization System (v2.1)

A full-stack IoT solution for tracking assets in GPS-denied environments using Arduino, LoRa, and Python.

1. Project Overview

This project implements a wireless localization system that estimates the 2D position (x, y) of a mobile node using LoRa (Long Range) technology. Unlike GPS, which is often unreliable indoors or underground, this system uses RSSI (Received Signal Strength Indicator) from three fixed "Anchor" nodes to triangulate the position of a "Mobile" node.

The system features a real-time web dashboard that performs:

1. **Signal Processing:** Converts RSSI to distance using a Log-Distance Path Loss model.
2. **Error Reduction:** Applies a 1D Kalman Filter to smooth out noisy signal data caused by multipath interference.
3. **Visualization:** Displays the live location on a dynamic HTML5 Canvas map.

2. File Structure

```
/project-root
│
├── backend/
│   └── server.py        # Python Flask Server (Bridges Arduino Serial -> Web Socket)
│
├── frontend/
│   ├── assets/          # Images (Module.jpg, Team photos) & Project Report
│   └── index.html       # The Main Dashboard (UI, Kalman Logic, Trilateration Math)
│
├── G24-WriteUp.pdf      # Detailed project write-up
└── Readme.txt           # This file
```

3. Hardware Requirements

To replicate this project, you need the following components:

- **Arduino UNO (x4):** 3 for Anchors, 1 for Mobile Node.
- **LoRa Module (SX1278) (x4):** 433 MHz operating frequency.
- **Antenna (x4):** Custom 17cm wire antennas (optimized for 433 MHz).
- **Power Supply (x4):** Battery packs for mobile, DC sources for anchors.
- **Jumper Wires:** ~50 Male-to-Male/Male-to-Female.

4. Implementation Roadmap

Phase 1: Hardware Connections

Connect the LoRa SX1278 module to the Arduino UNO using the standard SPI pins. Repeat this for **ALL 4 units**.

Pin	Connection	Note
VCC	3.3V	**WARNING: Do not connect to 5V!**
GND	GND	
MISO	Pin 12	
MOSI	Pin 11	
SCK	Pin 13	
NSS	Pin 10	
DIO0	Pin 2	
RST	Pin 9	

Phase 2: Flashing the Arduino Code

You do not need to write code from scratch. The source code is embedded directly in the dashboard documentation.

1. Navigate to `frontend/index.html` and open it in your browser.
2. Click the **"PROJECT INFO"** button in the top navigation bar.
3. Select the **"Sender and Receiver Codes"** tab on the left.
4. **For the 3 Anchor Nodes:**
   - Copy the "Sender Code".
   - **IMPORTANT:** Change the ID for each board before uploading (e.g., replace `ANCHOR_1` with `ANCHOR_2` and `ANCHOR_3` for the respective boards).
   - Upload using Arduino IDE.
5. **For the Mobile Node:**
   - Copy the "Receiver Code".
   - Upload to the 4th Arduino.

Phase 3: Backend Setup

The backend acts as a bridge, reading data from the Mobile Node via USB and sending it to the web dashboard via WebSockets.

1. **Install Python Dependencies:**

   ```
   pip install pyserial flask flask-socketio
   ```

2. **Configure the Port:**
   - Connect the Mobile Node Arduino to your PC.
   - Check your Device Manager (Windows) or `/dev/tty` (Linux/Mac) to find the Port (e.g., `COM3`, `COM10`, `/dev/ttyUSB0`).
   - Open `backend/server.py` in a text editor.
   - Find the line `SERIAL_PORT = 'COM10'` and change it to your actual port.

Phase 4: Launching the System

1. Close Arduino IDE Serial Monitor (The Python script needs access to the port).
2. Run the server:

   ```
   cd backend
   python server.py
   ```

3. If successful, you will see `SUCCESS: Connected to COM...`.
4. Open your browser and go to: `http://localhost:5000`.

5. Mathematical Modelling & Calibration

Once the system is running, you need to calibrate it for your specific environment using the "Config Modal" on the website.

A. RSSI to Distance Conversion

The system calculates distance using the Log-Distance Path Loss model:

```
RSSI = A - 10 * n * log10(d)
```

Where:

- **RSSI:** Received Signal Strength Indicator (dBm).
- **A:** Reference RSSI value at 1 meter (Calibrated constant).
- **n:** Path loss exponent (Environmental constant).

- **d:** Distance in meters.

To calibrate:

1. Place the Mobile Node 1 meter from an Anchor.
2. Input the observed RSSI as `A` in the Settings Modal.
3. Adjust `n` (typically 2.0 for free space, 1.6-3.0 for indoor) until the distance matches reality.

B. Error Reduction (Kalman Filter)

Raw RSSI data fluctuates (+/- 5-8 dBm) due to noise. We use a 1-Dimensional Kalman Filter to smooth these values. The filter operates recursively:

Prediction & Update Steps:

```
K = P / (P + R)
X_new = X_prev + K * (Measurement - X_prev)
P_new = (1 - K) * P
```

Where:

- **R:** Process Noise (System uncertainty).
- **Q:** Measurement Noise (Sensor uncertainty).
- **Tuning Tip:** Increase `Q` in the Settings Modal for smoother (but slower) tracking.

C. 2D Trilateration

The dashboard determines the position (x, y) by solving the intersection of three circles:

```
(x - x_i)^2 + (y - y_i)^2 = d_i^2   for i=1, 2, 3
```

The system solves these simultaneous equations numerically to pinpoint the mobile node on the Canvas map.

6. The Team (G24)

LoRa Localization Group

- N V Poorna Vadhan (2024MCB1306)
- Mekala Sathwik Varma (2024MCB1305)
- Pola Sai Sujith (2024MCB1309)
- Bukke Balaji Naik (2024MCB1290)
- Jatin (2024MCB1296)

For detailed theoretical background, please refer to the `Project Report.pdf` or `G24-WriteUp.pdf` included in the assets folder.