Intern/Project Trainee Report June 2025

Reg No:

PROTOTYPE OF A MULTI-NODE LoRa IoT SYSTEM WITH CLOUD-CONNECTED DASHBOARD FOR REAL-TIME MONITORING

Aditya Krishna Jaiswal

Indian Institute of Information Technology Guwahati

Submitted to

Dr. Anjan Debnath

Scientist - F
Department of Space, Government of India
North Eastern Space Applications Centre, Meghalaya

North Eastern Space Applications Centre

Department of Space, Government of India
Umiam – 793103, Meghalaya

Abstract

This report details the design, development, and implementation of a Long Range (LoRa) based multi-node sensor network. The primary objective of this project was to gain a comprehensive understanding of LoRa and LoRaWAN technologies and apply this knowledge to create a practical environmental monitoring system. The project involved the creation of LoRa end-nodes (transmitters and a receiver), which evolved from initial prototypes using Ai-Thinker modules to a more robust solution with Wio-E5 development boards. A multi-transmitter and single-receiver network was established to collect sensor data, including temperature, humidity, and a simulated landslide detection alert. The receiver also functioned as a gateway, forwarding the collected data to a PostgreSQL database hosted on Supabase. A local dashboard was developed for real-time data visualization. This report documents the entire project lifecycle, from the initial learning phase and hardware selection to the challenges faced with signal strength, the successful implementation of the multi-node network, and the final data management and visualization pipeline.

Acknowledgement

I would like to express my deepest gratitude to Dr. Anjan Debnath, Scientist-F at the North Eastern Space Applications Centre (NE-SAC), Department of Space, Government of India, Meghalaya for his invaluable mentorship, guidance, and support throughout the course of this internship project.

A special thanks to my friend, co-intern, and batchmate **Pratham Agarwal**, whose constant encouragement, collaborative spirit, and technical support were instrumental to the project's success.

I would also like to extend my appreciation to the other team members of **Team Antriksha**, whose participation in the *Bhartiya Antriksha Hackathon 2024* laid the foundation for this internship opportunity. In particular, I am grateful to **Krish Kahnani** and **Keshav Jindal** for their dedication, teamwork, and shared vision.

Finally, I am thankful to everyone who contributed directly or indirectly to the successful completion of this project.

Aditya Krishna Jaiswal
Indian Institute of Information Technology Guwahati

Contents

Chapter 1

Introduction

1.1 Background and Motivation

The Internet of Things (IoT) has seen exponential growth, connecting billions of devices that sense, process, and communicate data from the physical world. A significant challenge in many IoT applications, particularly in environmental monitoring, agriculture, and smart cities, is the need for long-range, low-power communication. Traditional technologies like Wi-Fi and Bluetooth are limited in range, while cellular networks can be power-hungry and costly for simple sensor applications. Low-Power Wide-Area Networks (LPWANs) have emerged to fill this gap, offering multi-kilometer range with battery life that can last for years. LoRa (Long Range) is a leading LPWAN technology that has gained significant traction due to its open-standard approach and excellent performance characteristics. This project was motivated by the need to explore this technology's potential for creating cost-effective and scalable monitoring solutions.

1.2 Problem Statement

The core problem addressed by this project was to design and build a functional, end-to-end IoT monitoring system using LoRa technology. This involved overcoming the initial steep learning curve of a new communication protocol, navigating the practical challenges of RF hardware implementation, and creating a complete data pipeline from sensor to a user-facing dashboard. A key sub-problem that emerged was the significant performance discrepancy between different LoRa hardware modules, which necessitated a thorough investigation and a strategic pivot in hardware selection to achieve the desired reliability.

1.3 Project Objectives

The primary objectives of this project were defined as follows:

1. To conduct a detailed study of LoRa and LoRaWAN technologies to understand their underlying principles, architecture, and use cases.
2. To prototype a simple LoRa transmitter-receiver pair and evaluate its real-world performance.
3. To identify and troubleshoot hardware-related performance issues, specifically concerning signal strength (RSSI) and signal-to-noise ratio (SNR).

4. To design and implement a reliable multi-node network where multiple sensor nodes transmit data to a single receiver gateway.

5. To integrate sensors for collecting environmental data (temperature, humidity) and a simulated event trigger (landslide detection).

6. To develop a gateway that not only receives LoRa packets but also uploads the data to a cloud-based PostgreSQL database using a modern API service (Supabase).

7. To create a simple dashboard for visualizing the collected data in real-time.

1.4 Scope and Limitations of the Work

The scope of this project encompasses the entire lifecycle of an IoT product prototype, from theoretical research to hardware implementation and software development. It covers the use of 433/8 MHz LoRa modules, sensor integration, multi-node communication logic, and cloud data integration.

The limitations of this project include:

- The network was implemented using LoRa modulation in a point-to-multipoint topology, not a full LoRaWAN stack with a dedicated Network Server. This simplified the development but does not explore the full capabilities of LoRaWAN (like adaptive data rate or class B/C devices).

- The "landslide detection" is a simulation (sending a binary 1 or 0) and does not involve an actual geophysical sensor.

- The data dashboard is hosted locally and is designed for proof-of-concept visualization rather than as a production-grade, multi-user application.

1.5 Structure of the Report

This report is structured into eight chapters. Chapter 2 provides the theoretical background on LoRa and LoRaWAN. Chapter 3 details the initial, challenging prototyping phase with Ai-Thinker modules. Chapter 4 describes the successful reimplementation with Wio-E5 boards. Chapter 5 focuses on the design of the multi-sensor network. Chapter 6 covers the data backend and visualization pipeline. Chapter 7 presents and discusses the final results. Finally, Chapter 8 concludes the report and suggests avenues for future work.

Chapter 2

Theoretical Fundamentals: LoRa and LoRaWAN

2.1 The Rise of Low-Power Wide-Area Networks (LPWANs)

LPWANs are a category of wireless communication technologies designed for IoT applications that require long-range transmissions at a low bit rate. They are optimized for low power consumption, allowing devices to operate on small batteries for several years. This makes them ideal for deploying large-scale sensor networks in locations where power and internet connectivity are scarce. Key players in the LPWAN space include LoRaWAN, Sigfox, and NB-IoT.

2.2 Introduction to LoRa

LoRa is the proprietary physical layer (PHY) protocol for a spread spectrum modulation technique based on Chirp Spread Spectrum (CSS). Developed by Cycleo of France and acquired by Semtech, LoRa is the "long-range" technology itself. It is not a full network protocol but the method by which a signal is sent and received.

2.2.1 The LoRa Physical Layer (LoRa PHY)

The LoRa PHY defines how a signal is encoded and modulated to be transmitted wirelessly over the air. It operates in the sub-gigahertz ISM (Industrial, Scientific, and Medical) bands, such as 433 MHz, 868 MHz (Europe), and 915 MHz (North America). The use of these lower frequencies allows for better signal penetration through obstacles compared to higher-frequency bands like 2.4 GHz (Wi-Fi, Bluetooth).

2.2.2 Chirp Spread Spectrum (CSS) Modulation

LoRa's resilience and range are derived from its use of Chirp Spread Spectrum modulation. In CSS, information is encoded in "chirps," which are signals that continuously vary in frequency, either increasing (up-chirp) or decreasing (down-chirp) over time. This technique spreads a narrow-band signal over a wider channel bandwidth. The key advantages of CSS are:

- **Resilience to Interference:** Because the signal is spread out, it is less susceptible to narrow-band interference.
- **Robustness against Multipath and Fading:** The CSS technique is inherently robust to signal reflections and Doppler shift.
- **High Sensitivity:** LoRa receivers can detect signals that are significantly below the noise floor (as low as -20 dB). This is a primary reason for its long-range capability.

2.2.3 Key LoRa Parameters

The performance of a LoRa link can be tuned by adjusting several key parameters, creating a trade-off between data rate, range, and power consumption.

Spreading Factor (SF)

The Spreading Factor determines the duration of each chirp. It ranges from SF7 to SF12. A higher SF means a longer chirp duration, which spreads the signal more.

- **Higher SF (e.g., SF12):** Longer range, more robust signal, lower data rate, higher time on air, more power consumption per transmission.
- **Lower SF (e.g., SF7):** Shorter range, higher data rate, lower time on air.

Bandwidth (BW)

This is the width of the frequency band used for the chirp. Common values are 125 kHz, 250 kHz, and 500 kHz. A wider bandwidth allows for a higher data rate but can reduce receiver sensitivity slightly.

Coding Rate (CR)

LoRa uses forward error correction (FEC) to detect and correct bit errors at the receiver. The Coding Rate (e.g., 4/5, 4/6, 4/7, 4/8) defines how much redundant data is added. A higher coding rate (e.g., 4/8) adds more redundancy, increasing reliability but also increasing the time on air.

2.3 Understanding LoRaWAN

While LoRa is the physical layer, LoRaWAN is the MAC (Media Access Control) layer protocol that sits on top of it. LoRaWAN is an open standard maintained by the LoRa Alliance. It defines the network architecture and communication protocol, turning LoRa into a scalable networking solution.

2.3.1 LoRaWAN Network Architecture

A LoRaWAN network typically has a star-of-stars topology.

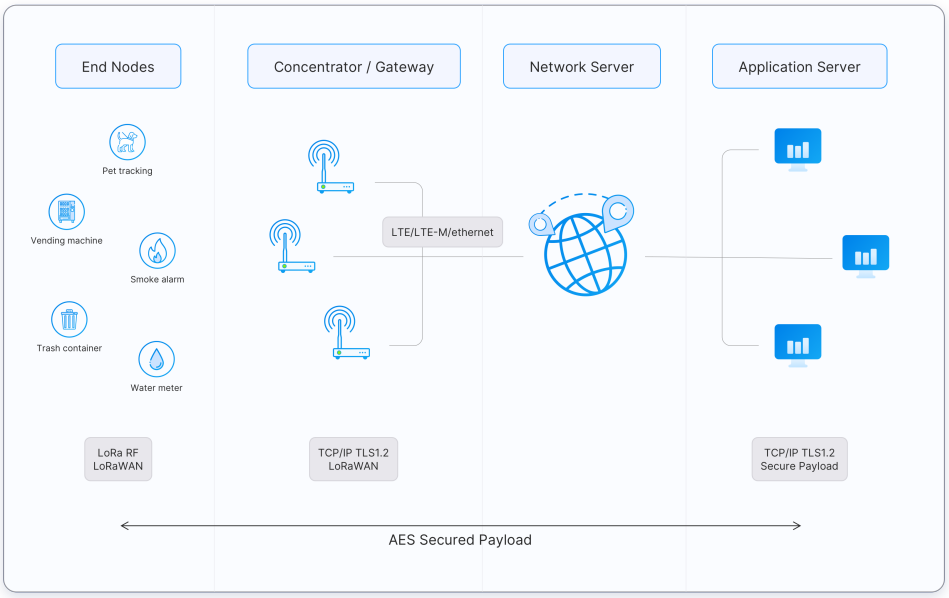

Figure 2.1: Typical LoRaWAN Network Architecture.[2]

End Devices:

These are the sensors and actuators in the field that collect data and have a LoRa radio.

Gateways:

A gateway receives LoRa packets from end devices and forwards them to a Network Server, typically over a standard IP connection (Wi-Fi, Ethernet, or Cellular).

Network Server:

The Network Server is the brain of the network. It deduplicates messages from multiple gateways, handles security checks, manages the device sessions, and routes messages to the correct Application Server.

Application Server:

This server handles the application-specific logic, processing the sensor data and integrating with user dashboards or other cloud services.

Chapter 3

Initial Prototyping with Ai-Thinker SX127x Modules

3.1 Hardware Selection and System Setup

The initial phase of the project focused on creating a proof-of-concept using widely available and low-cost components. The Ai-Thinker Ra-02 module, based on the Semtech SX1278 chip for the 433 MHz band, was chosen for this purpose. This bare-chip module was interfaced with an Arduino Uno for control and communication via the SPI protocol. The following diagram shows the pinout of a bare LoRa module.

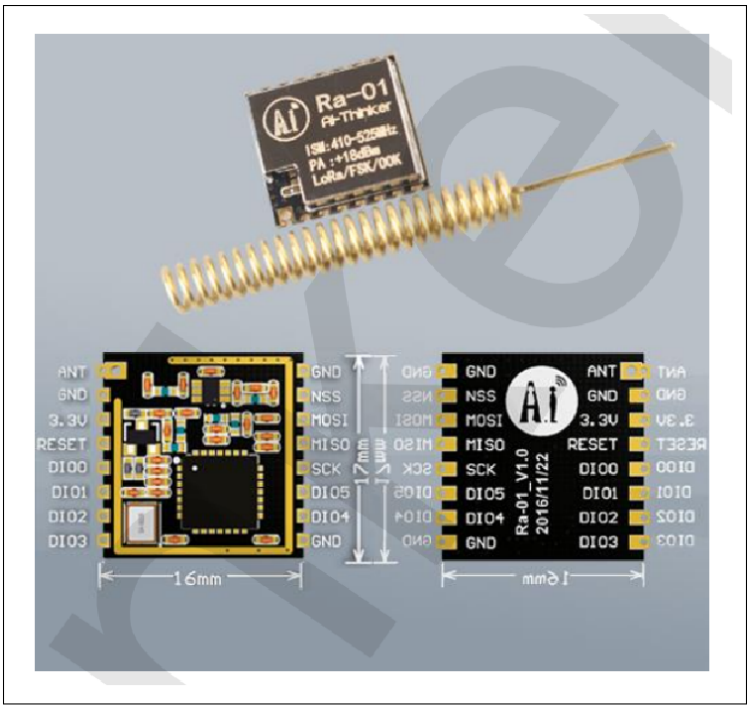

Figure 3.1: Pinout of Ai-Thinker Ra-01 (SX1278) LoRa Module [12]

3.2 Development of a Basic Transmitter and Receiver

The Arduino IDE was employed as the development environment, leveraging the `arduino-LoRa` library by Sandeep Mistry to simplify the initialization of the LoRa chip and facilitate

data packet transmission and reception. The transmitter and receiver setups, as shown in Figures ?? and **??**, were implemented using Arduino-compatible hardware interfaced with LoRa modules, enabling reliable wireless communication for sensor data. See Appendix for circuit schmatics.

- **Transmitter Firmware:** The transmitter was programmed to send dummy sensor values in a custom packet format (see Section 5.2.1) every two seconds, containing sensor ID, temperature, humidity, and landslide alert status.
- **Receiver Firmware:** The receiver was configured to listen continuously on the same frequency, parsing received packets to extract sensor ID, temperature, humidity, and landslide alert status, then displaying these values along with RSSI and SNR on the Serial Monitor.

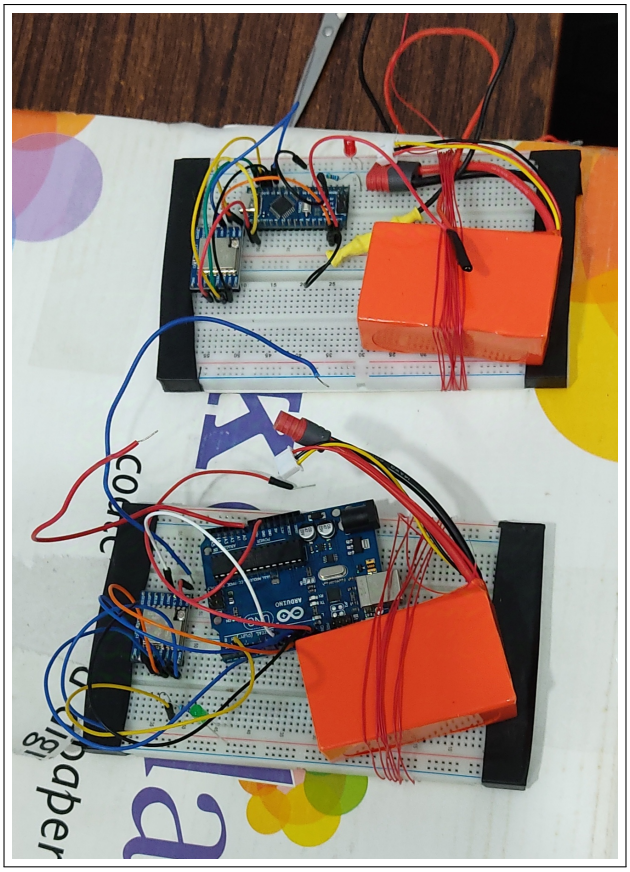

Figure 3.2: LoRa Transmitter and Receiver Setups

3.3 Performance Evaluation and Critical Challenges

Upon testing this basic setup, the performance was immediately identified as a major issue. Even with the transmitter and receiver placed within a meter of each other, the signal quality was extremely poor.

Analyzing Received Signal Strength Indicator (RSSI)

RSSI is an absolute measure of the received signal power in dBm. The values observed were around **-120 dBm**. For comparison, a good LoRa link at close range should be well

above -50 dBm. An RSSI of -120 dBm is typically associated with signals at the limit of a multi-kilometer range, not a one-meter distance.

3.3.1 The Problem of Poor Signal Integrity

Experimental Results with On-chip Antenna

The Ai-Thinker Ra-02 module has a connection for an external antenna. Initially, simple spring coil antennas were used. The poor performance suggested a severe impedance mismatch, poor antenna design, or potential issues with the RF circuitry on the module itself.

Attempts to Improve Performance with Makeshift Antennas

To rule out the antenna as the sole problem, several makeshift monopole antennas were created by cutting wires to a quarter wavelength of the 433 MHz frequency (approx. 17.3 cm). While this brought a marginal improvement, the results were still highly unsatisfactory:

- **Within 1 meter:** SNR improved to -16 dB, RSSI improved to -70 dBm.
- **Within 5 meters:** SNR dropped to -18 dB, RSSI dropped to -86 dBm.

These values were still far too low for a reliable communication link of any meaningful distance.

3.3.2 Conclusions from the Initial Prototyping Phase

The severe performance issues with the Ai-Thinker modules were deemed a major roadblock. While it's possible that a batch of modules was faulty or that a more sophisticated antenna and ground plane design was needed, the out-of-the-box performance was not viable for the project's goals. A decision was made to pivot to a more integrated and professionally designed hardware platform to mitigate these low-level RF issues.

Chapter 4

Advanced Implementation with Wio-E5 Development Boards

4.1 Transitioning to a More Robust Hardware Platform

To overcome the challenges faced with the Ai-Thinker modules, the project transitioned to the Seeed Studio Wio-E5 development board. This board was chosen for its integrated design and superior RF performance.

4.1.1 Introduction to the Seeed Studio Wio-E5 Development Board

The Wio-E5 is a compact development board featuring the Wio-E5 module. This module contains the STM32WLE5JC SoC, which is a significant step up from the previous setup. Fig. 4.1 and Fig. 4.2 show the technical specification and pin out of the board

4.1.2 The STM32WLE5JC System-on-Chip (SoC)

The key advantage of the STM32WLE5JC is that it integrates both a general-purpose microcontroller (ARM Cortex-M4) and a sub-GHz radio onto a single chip. This tight integration, designed and manufactured by STMicroelectronics, ensures optimized RF performance, proper impedance matching, and a certified design, eliminating the guesswork involved in using bare modules. The sub-GHz radio consists of:

- An analog front-end transceiver, capable of outputting up to +15 dBm maximum power on its RFO_LP pin and up to +22 dBm maximum power on RFO_HP pin.
- A digital modem bank providing the following modulation schemes:
  - LoRa Rx/Tx with bandwidth (BW) from 7.8 -500 kHz, spreading factor (SF) 5 -12, bit rate (BR) from 0.013 to 17.4 Kbit/s (real bitrate).
  - FSK and GFSK Rx/Tx with BR from 0.6 to 300 Kbit/s.
  - (G)MSK Tx with BR from 0.1 to 10 Kbit/s.
  - BPSK Tx only with bitrate for 100 and 600 bit/s.

- A digital control including all data processing and sub-GHz radio configuration control.
- A high-speed clock generation.

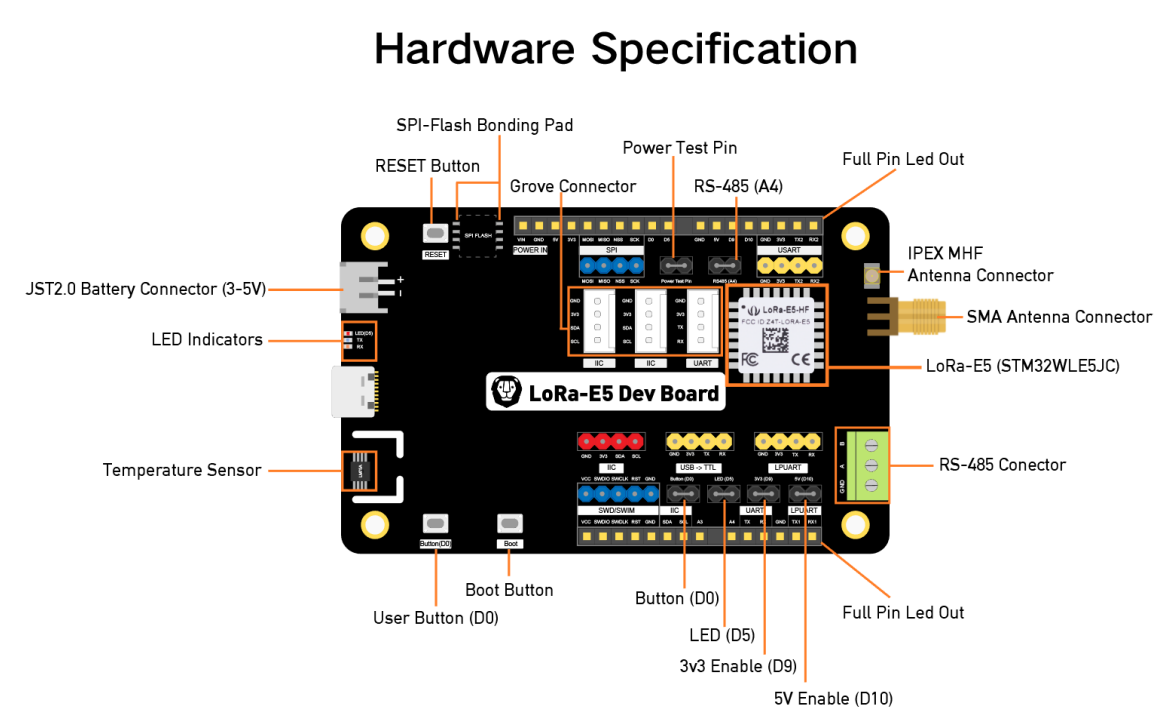

Figure 4.1: Wio-E5 Development Board Technical Specifications [9]

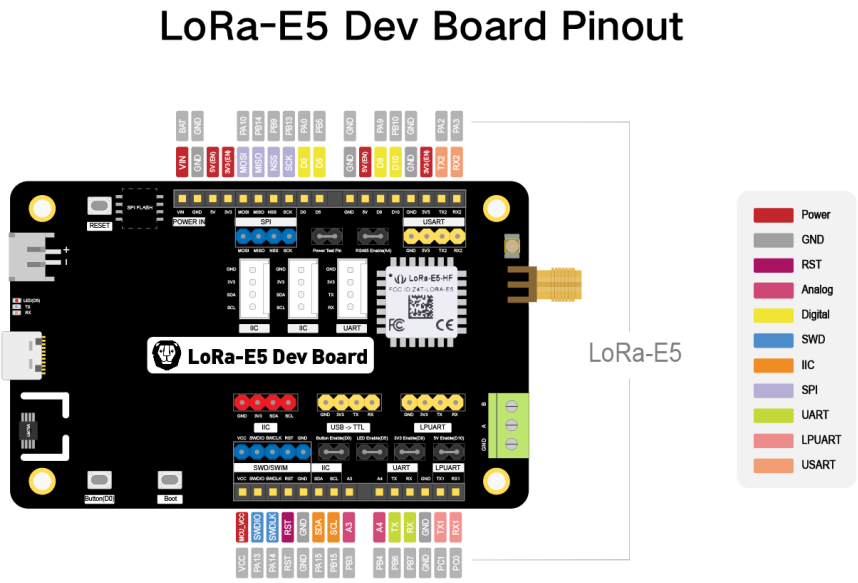

Figure 4.2: Wio-E5 Development Board Pinout[9]

4.2 Setting up the Wio-E5 Development Environment

The board was programmed only using AT commands to verify and control its functionality, with no custom firmware development. This allowed for rapid prototyping and testing of LoRa communication features using the built-in AT command interface. Some of the basic AT commands have been listed below.

Basic AT Commands

```
AT+ID                // Read all, DevAddr(ABP), DevEui(OTAA), AppEui(OTAA)
AT+ID=DevAddr                    // Read DevAddr
AT+ID=DevEui                     // Read DevEui
AT+ID=AppEui                     // Read AppEui
AT+ID=DevAddr,"devaddr"         // Set new DevAddr
AT+ID=DevEui,"deveui"           // Set new DevEui
AT+ID=AppEui,"appeui"           // Set new AppEui

AT+KEY=APPKEY,"16 bytes key"    // Change application session key

AT+DR=band                      // Change the Band Plans
AT+DR=SCHEME                    // Check current band
AT+CH=NUM, 0-7                  // Enable channel 0~7

AT+MODE="mode"                  // Select work mode: LWOTAA, LWABP or TEST

AT+JOIN                         // Send JOIN request

AT+MSG="Data to send"          // Send unconfirmed string frame
AT+CMSG="Data to send"         // Send confirmed string frame
AT+MSGHEX="xx xx xx xx"        // Send unconfirmed hex frame
AT+CMSGHEX="xx xx xx xx"       // Send confirmed hex frame
```

The board was used in test mode for both the transmitter and the receiver side in the LoRa band corresponding for usage in India, i.e 865 MHz.

- **Transmitter Side:**

  ```
  AT+MODE=TEST
  AT+TEST=RFCFG,865.0625,SF12
  AT+TEST=TXLRPKT,"{II,TT,HH,L}"
  ```

- **Receiver Side:**

  ```
  AT+MODE=TEST
  AT+TEST=RFCFG,865.0625,SF12
  AT+TEST=RXLRPKT
  ```

4.3 Comparative Performance Analysis

A simple transmitter-receiver test, identical to the one performed with the Ai-Thinker modules, was conducted using two Wio-E5 boards. The results were dramatically better, highlighting the advantages of using a professionally integrated development board.

4.3.1 Significant Improvements in SNR and RSSI

With standard antennas provided with the boards, the following measurements were recorded under two different conditions:

1. **Short-Range Testing (1 meter):**

- **SNR:** A very healthy **+2 to +4 dB**. This indicates the signal power was significantly higher than the noise floor.
- **RSSI:** An excellent **-10 to -15 dBm**. This is a very strong signal, as expected for devices in close proximity.

2. **Long-Range Testing (1 km):**

- **SNR:** Dropped significantly to **-17 to -18 dB**, indicating the signal was just above the noise floor in the worst-case scenario.
- **RSSI:** Decreased to **-112 to -120 dBm**, which is near the receiver sensitivity threshold but still within the acceptable reception range for LoRa.

4.3.2 Impact of Terrain and Line-of-Sight Obstruction

The long-range test was conducted in a hilly region, with a considerable elevation difference between the transmitter and receiver nodes. The exact locations of the transmitter (**TX**) and receiver (**RX**) are illustrated in Figure 4.3. The terrain and elevation posed natural challenges for radio propagation.

Notably, the signal reception dropped completely beyond a certain point, which coincided with a school building coming directly into the line-of-sight between the TX and RX. It is highly probable that the building acted as a physical barrier, attenuating or blocking the LoRa signal. Beyond this obstruction, no packets were received.

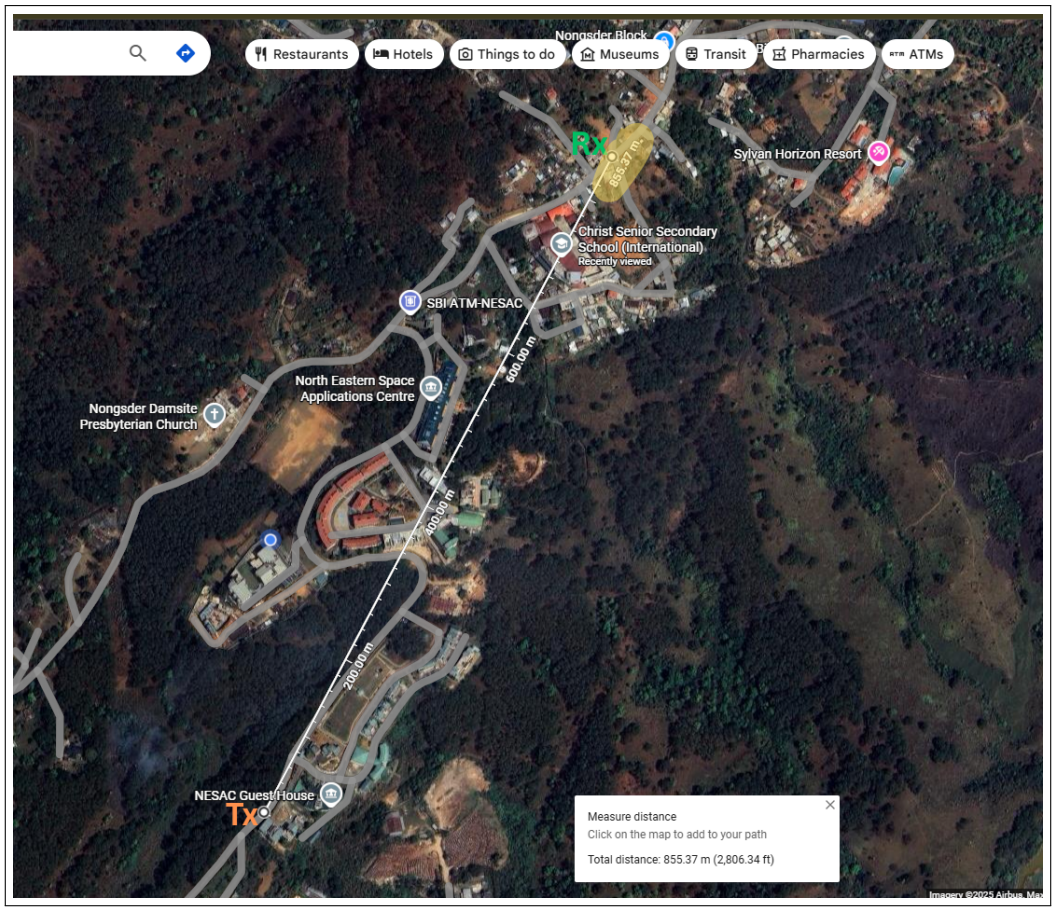

Figure 4.3: Test Location Map Showing Transmitter (TX) and Receiver (RX) Sites with Hilly Terrain and Obstruction

These results underline both the strengths and limitations of LoRa communication in real-world scenarios—demonstrating strong near-range performance and long-range capability under ideal conditions, but also susceptibility to significant obstructions in complex terrains.

4.3.3 Transmitter Node Hardware Integration

For ease of prototyping and rapid development, the Wio-E5 development board was interfaced with an Arduino Nano on the transmitter side. The Nano, being Arduino-compatible and breadboard-friendly, allowed for a compact and modular design. Communication between the Arduino Nano and the Wio-E5 board was established via a UART serial interface, enabling the Nano to send AT commands directly to the Wio-E5 module.

Additionally, a DHT11 temperature and humidity sensor was connected to the Arduino Nano. The sensor data (temperature and humidity) was read using the Nano and then formatted into a fixed structure packet {`II,TT,HH,L`}, where the `L` field represented a manually triggered landslide alert via a push-button.

4.3.4 Receiver Node with Wi-Fi Integration

On the receiver side, the Wio-E5 development board was interfaced with an ESP8266 board. The ESP8266 enabled the receiver node to connect to a Wi-Fi network and acted

as a bridge between the LoRa packet reception and potential cloud-based services or web interfaces.

The ESP8266 received the data over UART from the Wio-E5 and could be programmed to:

- Parse the received packet.
- Upload the data to a cloud service or local server.
- Display the data on a local web dashboard or serial monitor.

This setup effectively demonstrated a basic LoRa-to-Wi-Fi gateway, enabling long-range sensor data to be transmitted over LoRa and then pushed to the internet via the ESP8266.

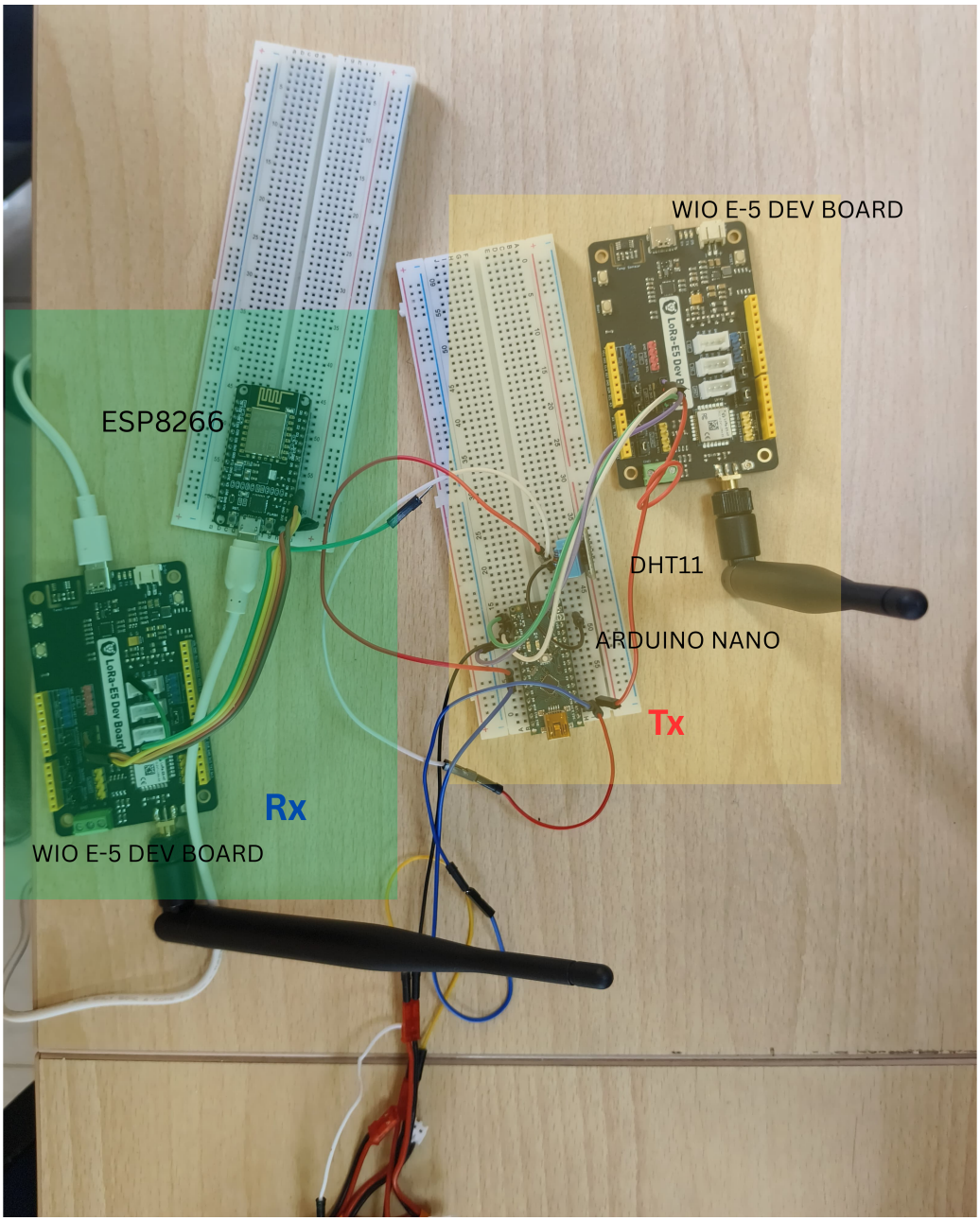

Figure 4.4: System-Level Overview of Transmitter (Arduino + Wio-E5 + DHT11) and Receiver (Wio-E5 + ESP8266) Setup

Chapter 5

Designing and Implementing a Multi-Sensor LoRa Network

5.1 Network Architecture and Topology

A star topology was chosen for the network, with multiple end-nodes (transmitters) sending data to a single central receiver/gateway. To manage data from multiple sources, a simple addressing scheme was implemented. Each end-node was assigned a unique ID (e.g., Node 1, Node 2).

5.2 End-Node Development with Integrated Sensors

Each transmitter node was built using a Wio-E5 board connected to a DHT11 sensor for temperature and humidity readings.

5.2.1 Data Packet Structure

A custom data packet was defined to send the data efficiently. The format follows a compact and fixed-length structure consisting of:

`{II,TT,HH,L}`

Where:

- `II` – 2-digit sensor (node) ID
- `TT` – 2-digit temperature value (in °C)
- `HH` – 2-digit humidity value (in %)
- `L` – 1-digit landslide alert flag (0 for no alert, 1 for alert)

For example, a packet from sensor 01 reporting a temperature of 25°C, humidity of 60%, and no landslide alert would look like:

`0125600`

This fixed-format structure ensures compact transmission over LoRa and easy parsing at the receiver end. ‘

5.2.2 Implementing a Dummy Landslide Detection Mechanism

A push button was added to each transmitter node to simulate a landslide alert. When the button is pressed, the `LandslideAlert` field in the data packet is set to `1`; otherwise, it remains `0`. This setup enables a demonstration-driven approach, allowing critical alerts to be sent alongside routine sensor data.

5.3 The Central Receiver as a LoRa Gateway

One Wio-E5 board was designated as the central receiver. Its firmware was designed to:

1. Listen continuously for incoming LoRa packets.
2. On reception, parse the data string to extract the Node ID and the sensor values.
3. Print the received data to the serial port for debugging.
4. Call the function to upload the parsed data to the cloud database.

Chapter 6

Data Persistence and Visualization

6.1 Gateway to Cloud Communication

To store the data for analysis and visualization, a cloud-based backend was required. Supabase was chosen for this purpose.

6.1.1 Introduction to Supabase

Supabase is an open-source Firebase alternative. It provides a suite of tools built around a PostgreSQL database, including instant APIs, authentication, and storage. Its key benefit for this project was the auto-generated RESTful API that allows for easy interaction with the database via standard HTTP requests.

6.1.2 Setting up the Postgres Database on Supabase

A new project was created in Supabase. A single table named `sensor_data` was created with the following schema:

- `created_at` (Timestamp, default now())
- `sensor_id` (integer)
- `temperature` (float)
- `humidity` (float)
- `latitude` (float)
- `longitude` (float)
- `landslide_alert` (integer)

sensor_data

NEW

Filter | Sort | Insert | Auth policy | Role postgres | Realtime off | API Docs

id int4	sensor_id int4	temperature float8	humidity float8	landslide int4	rssi int4	snr int4	latitude float8	longitude float8	timestamp timestamptz
1	1	27.5	74.7	1	80	20	1.246446	3.235645	NULL
2	1	25.5	60.2	0	-70	8	40.7128	-74.006	NULL
3	1	29.5	45	0	-10	4	25	80	NULL
4	1	24.5	71	0	-23	8	25	80	NULL
5	1	30.3	76	0	-8	3	23	91	NULL
6	1	30.7	81	1	-8	3	22	91	NULL
7	1	31.7	85	1	-8	3	24.3	91	NULL
8	1	32.3	88	1	-8	3	25	91	NULL
9	1	32.1	88	1	-8	3	25.5	91	NULL

Figure 6.1: Supabase `sensor_data` table schema as seen in the dashboard

6.2 Uploading Data to the Cloud

Since the Wio-E5 development board does not have built-in Wi-Fi capabilities, the LoRa receiver node (Wio-E5) was interfaced with an ESP8266 board using UART. The ESP8266 connects to a Wi-Fi network and acts as the internet gateway for the system.

It receives parsed sensor data from the Wio-E5 over UART and independently makes HTTP POST requests to the Supabase API. These requests insert the sensor data into a Postgres database hosted on Supabase. This approach eliminated the need for an intermediate host computer or Raspberry Pi, making the system lightweight and fully embedded.

6.2.1 ESP8266 Firmware for Supabase Integration

The ESP8266 firmware was written to:

- Listen for serial data from the Wio-E5.
- Parse the incoming fixed-format packets.
- Format the extracted fields (Node ID, Temperature, Humidity, Landslide Alert) into a JSON object.
- Perform an authenticated HTTP POST request to the Supabase REST API endpoint.

This direct communication pipeline ensured real-time data uploading to the cloud with minimal hardware and maximum efficiency.

6.3 Sensor Data Dashboard: Real-time Data Visualization

The dashboard serves as the primary interface for visualizing and interpreting sensor data in real time. Designed as part of a cloud-native system, it offers an integrated view of environmental conditions such as temperature, humidity, and landslide alerts across various sensor nodes. A snapshot of the dashboard interface is shown in Figure 6.2, demonstrating the interactive map and live-updating charts.

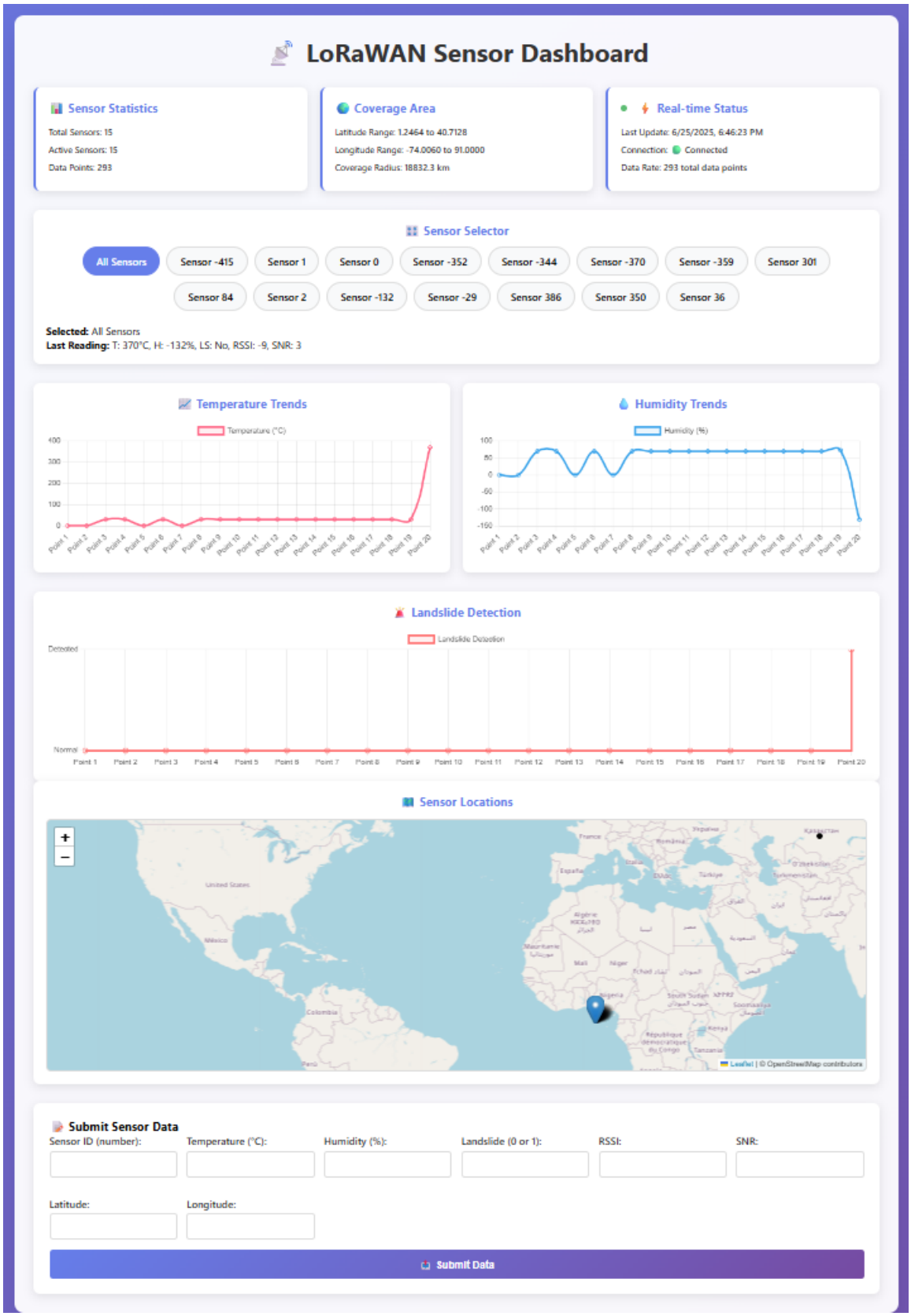

Figure 6.2: Sensor Data Dashboard showing real-time geospatial and temporal data

6.3.1 Frontend: Interactive and Real-time Visualization

The dashboard is built as a single-page application (SPA) using the Next.js framework, ensuring a fast and responsive user experience. It leverages Leaflet.js for geospatial visualization, allowing real-time plotting of sensor nodes on an interactive map with custom markers. Chart.js is used for rendering dynamic time-series charts that display the latest temperature, humidity, and landslide alert data for each node.

To ensure continuous data updates, the frontend establishes a persistent WebSocket connection using the Socket.io library. This enables push-based updates from the server, eliminating the need for manual refreshes and ensuring that operators always see the most recent sensor readings.

6.3.2 Backend: Serverless, Event-Driven Architecture

The backend is implemented using serverless functions hosted on the Vercel platform. These functions handle incoming API requests from sensors, process the data, and store it in the database. The use of Node.js ensures high efficiency for handling concurrent I/O operations typical in an IoT setting. The serverless model allows the system to scale automatically in response to load, particularly useful during events involving simultaneous data influx from multiple sensors.

6.3.3 Database: Real-time and Managed Storage

Data persistence is handled by a managed PostgreSQL database hosted on Supabase. Supabase's built-in real-time subscription engine allows the frontend to receive database updates as soon as new sensor data is inserted. This direct integration between the database and the frontend further enhances the real-time nature of the dashboard. The use of PostgreSQL ensures reliable and structured storage, with full support for advanced querying when needed.

Chapter 7

Conclusion and Future Scope

7.1 Summary of the Project

This project successfully navigated the journey from theoretical learning to the practical implementation of a multi-node LoRa sensor network. It began with an in-depth study of LoRa/LoRaWAN, followed by a critical prototyping phase that highlighted the importance of hardware quality in RF applications. By transitioning from problematic Ai-Thinker modules to robust Wio-E5 development boards, the project overcame initial signal integrity issues. A functional multi-sensor network was then built, capable of collecting and transmitting environmental data from multiple points to a central gateway. Finally, a complete data pipeline was established, where the gateway forwards data to a Supabase PostgreSQL database, with a live dashboard providing real-time visualization. All initial project objectives were successfully met.

7.2 Key Learnings and Takeaways

- **Hardware is Critical:** The single most important lesson was that for RF-based projects, the quality and design of the hardware (module, antenna, power supply) are paramount. The difference in performance between the bare-chip module and the integrated SoC solution was night and day.
- **Incremental Development:** Tackling the project in stages—from a simple single-node test to a multi-node network and finally cloud integration—was an effective strategy for managing complexity and troubleshooting issues systematically.
- **Power of Modern BaaS:** Using a Backend-as-a-Service platform like Supabase dramatically simplified the data persistence layer, allowing the project to focus on the IoT hardware and networking aspects rather than on complex backend development.

7.3 Recommendations for Future Work

This project serves as a strong foundation for several exciting future enhancements:

- **Full LoRaWAN Implementation:** The next logical step is to replace the current custom LoRa implementation with a full LoRaWAN stack. This would involve

setting up a LoRaWAN Network Server (e.g., The Things Stack) and registering the end-devices and gateway. This would enable more advanced features like Adaptive Data Rate (ADR) and scalability to thousands of devices.

- **Power Consumption Analysis:** Conduct a detailed power consumption analysis of the end-nodes. Implement sleep modes (e.g., MCU deep sleep) between transmissions to drastically extend battery life, making the nodes suitable for long-term, remote deployment.
- **Integration of Additional Sensors:** Expand the sensing capabilities by adding more relevant environmental sensors, such as soil moisture sensors for agriculture, air quality sensors (PM2.5) for urban monitoring, or accelerometers for more sophisticated landslide/vibration detection.
- **Real-World Deployment:** Deploy the network in a challenging real-world environment, such as a farm or a hilly area, to test its range, reliability, and resilience over a long period.

Appendix A

Schematics and Wiring Diagrams

This section would contains schematic showing the connections between the Wio-E5 boards, sensors (DHT11), and land slide alert button & also the Ai thinker loRa RA-02 based Transmitters/receivers.
Note:- To see readings on the Serial monitor,we can use Arduino IDE's serial monitor on a PC after ensuring correct port and Baud rate has been selected. For the entire project, a baud rate of 9600 has been selected. We have also used two LiPo batteries of 3.7V each connected in series for powering the boards but any power source which can provide a voltage between 7 -9V and should work.

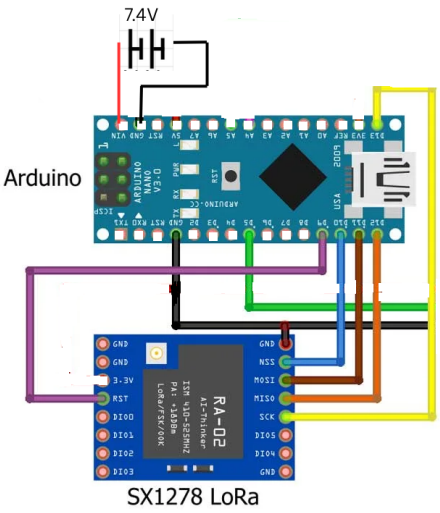

Figure A.1: Wiring Diagram for the Basic sx1276 based LoRa Transmitter/Receiver.

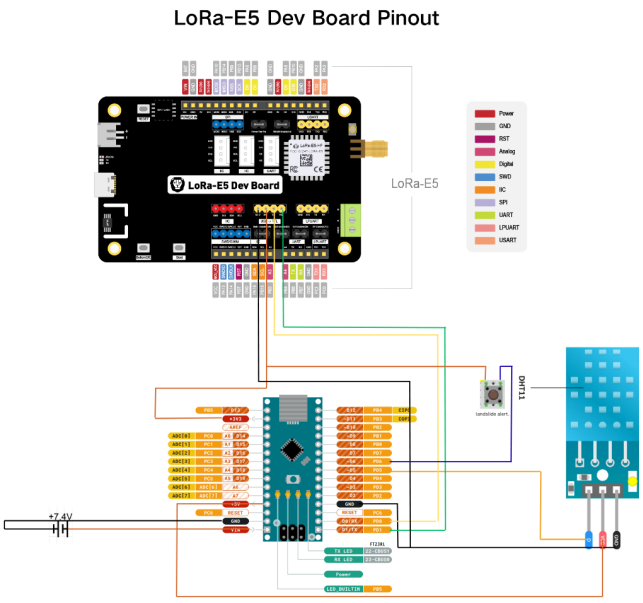

Figure A.2: Wiring Diagram for a Transmitter End-Node.

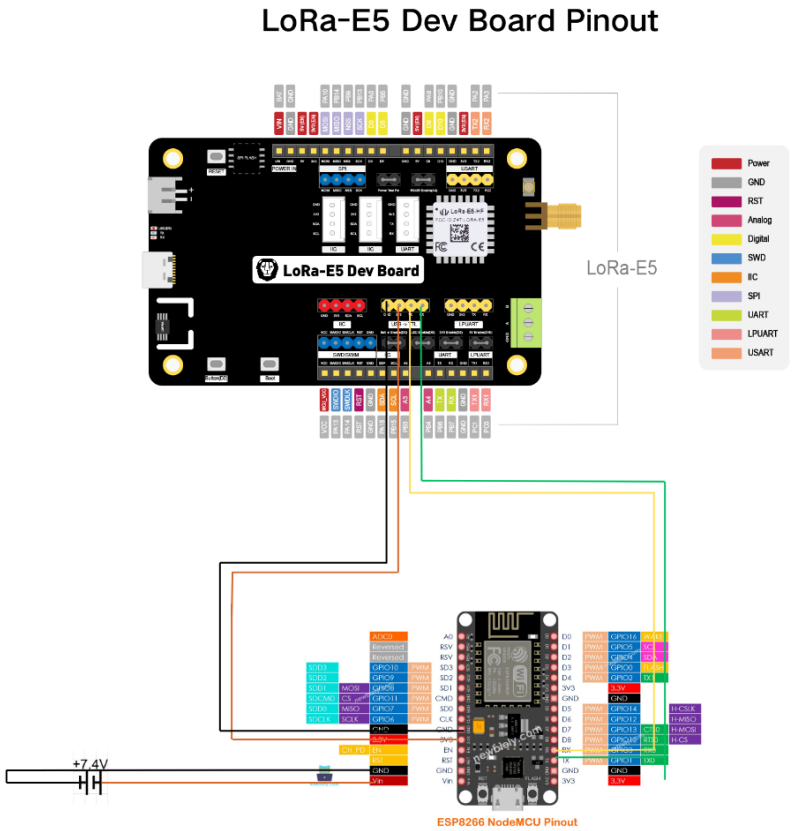

Figure A.3: Wiring Diagram for the E-5 Receiver/Gateway.

Appendix B

Source Code Listings

The complete, commented source code for the project firmware and scripts has been uploaded to the GitHub repository here.

Serial Monitor Snapshots

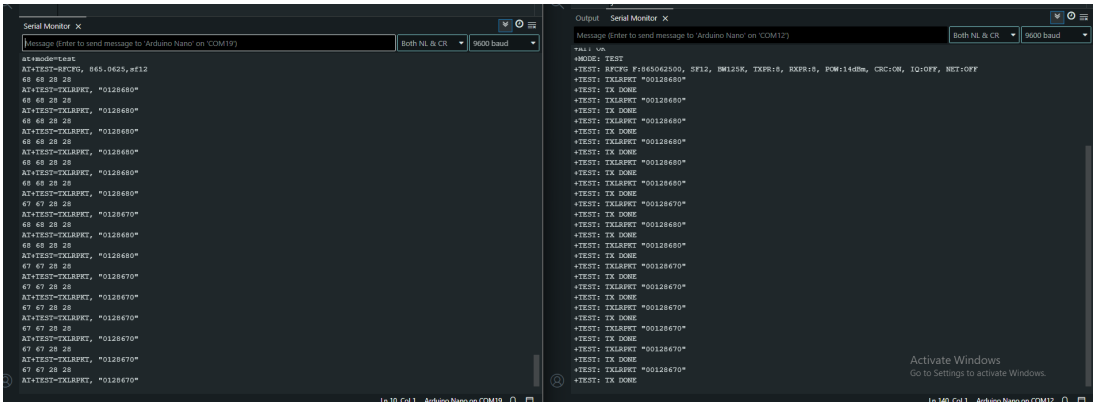

Figure B.1: Serial Monitor Output: Left: Transmitter Arduino Nano Right: Transmitter Wio E-5 (Ignore the initial 0 received)

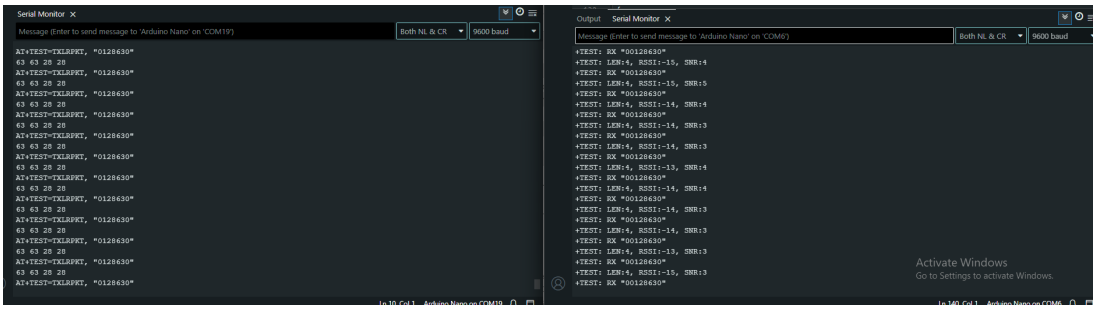

Figure B.2: Serial Monitor Output: Left: Transmitter Wio Board, Right: Receiver Wio Board (Ignore the initial 0 received)

Figure B.3: Serial Monitor Output: Receiver Node Gateway Logging

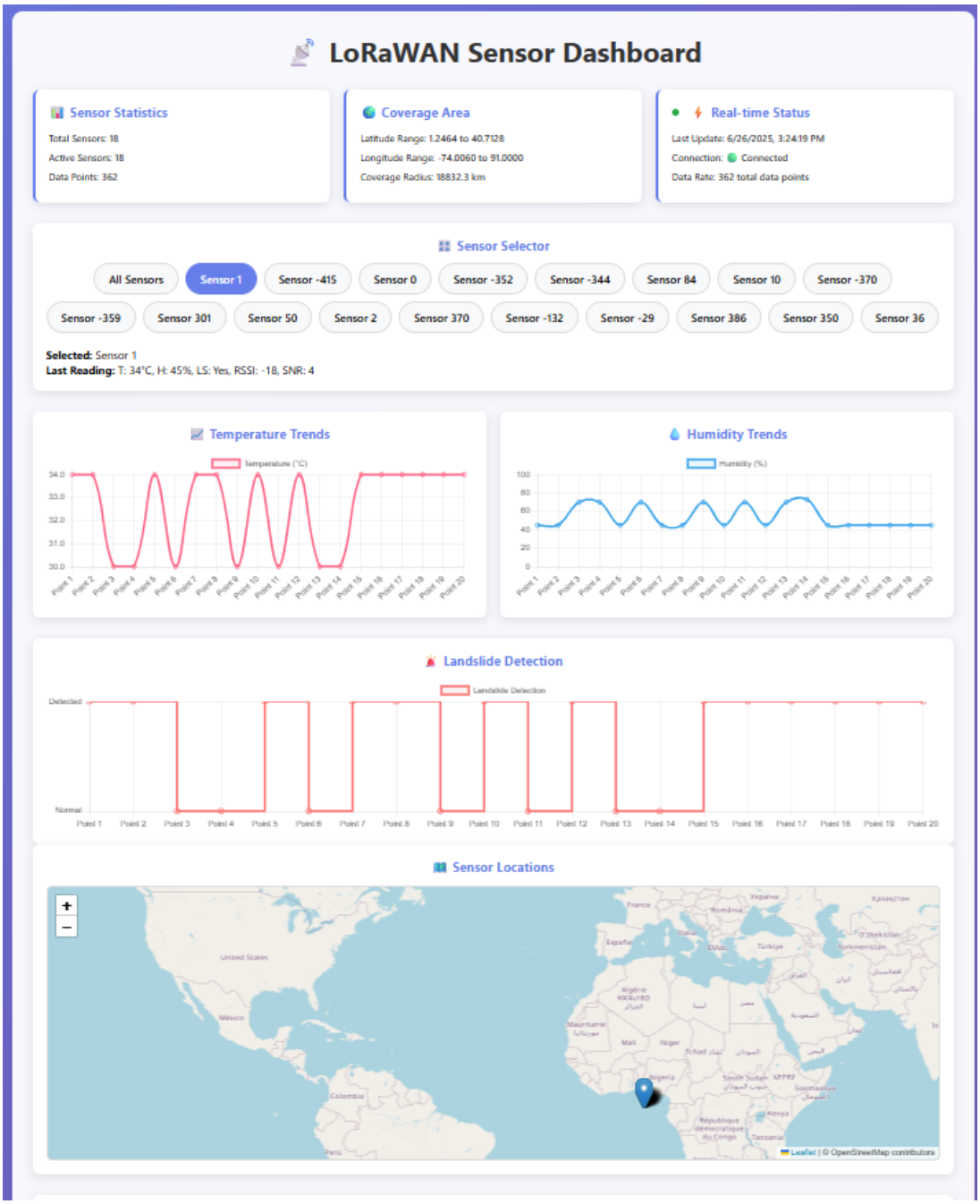

Figure B.4: Dashboard View

Bibliography

[1] Semtech Corporation, *What is LoRa?* [Online]. Available: https://www.semtech.com/lora/what-is-lora

[2] LoRa Alliance, *LoRaWAN Specification.* [Online]. Available: https://lora-alliance.org/

[3] Seeed Studio, *Wio-E5 Wireless Module - STM32WLE5JC.* [Online]. Available: https://www.seeedstudio.com/Wio-E5-Wireless-Module-p-4745.html

[4] Supabase, *The Open Source Firebase Alternative.* [Online]. Available: https://supabase.com/

[5] *LoRa SX1278 based Two Way Wireless Communication System with Arduino* [Video]. [Online]. Available: https://www.youtube.com/watch?v=08WNTuHj_QU

[6] Seeed Studio, *[Device Overview] Seeed Studio's Wio-E5 LoRa Development Boards* [Video]. [Online]. Available: https://www.youtube.com/watch?v=4ojRbUSfYdo&t=2s

[7] *[DIY Project] Application of Wio-E5 Dev Boards in LoRa WSN* [Video]. [Online]. Available: https://www.youtube.com/watch?v=B6kpMtoYYbQ&t=1835s

[8] Vinay Y.N., *Single Channel LoRa Gateway using Wio-E5 LoRa and Blynk.* [Online]. Available: https://www.hackster.io/vinayyn/single-channel-lora-gateway-using-wio-e5-lora-and-blynk-20b9ec

[9] Seeed Studio, *Wio-E5 Development Board Wiki.* [Online]. Available: https://wiki.seeedstudio.com/LoRa_E5_Dev_Board/

[10] STMicroelectronics, *STM32WLE5JC Product Overview.* [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32wle5jc.html

[11] *LoRa - Long-Range Radio for IoT — Arduino, ESP32, RPI Pico* [Video]. [Online]. Available: https://www.youtube.com/watch?v=YQ7aLHCTeeE

[12] Ai-Thinker, *LoRa Series Module User Manual.* [Online]. Available: https://docs.ai-thinker.com/en/lora/man