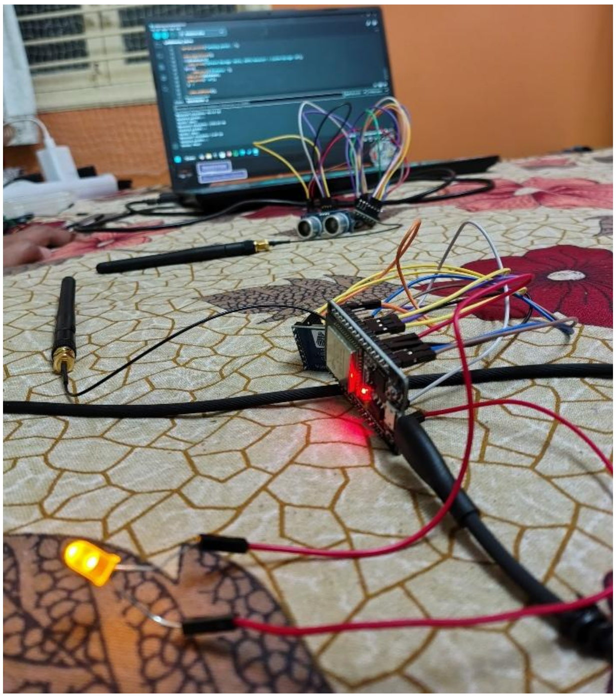
Project Documentation: LoRa-Based Smart Lighting System

1. Introduction

This project explores the capabilities of the **LoRa RA-02 module** for wireless communication and develops a **smart lighting system** that automatically controls room lighting based on object detection. The system uses **an Arduino (transmitter) with an ultrasonic sensor** and **an ESP32 (receiver) with an LED setup**.



2. Objectives

- Understand the **LoRa communication module** and its basic functionalities.
- Establish **one-way communication** between an Arduino (transmitter) and an ESP32 (receiver).
- Integrate an **ultrasonic sensor** to detect object presence.
- Automate a **lighting system** to turn on when an object is detected and off after a delay when removed.
- Gain insights into **LoRa communication parameters** such as **SPI setup, bandwidth, frequency settings, RSSI values, and SNR analysis**.

3. Components Used

Hardware

1. **Arduino UNO** (Transmitter)
2. **ESP32** (Receiver)
3. **LoRa RA-02 Module** (x2)

4. **Ultrasonic Sensor (HC-SR04)**
5. **LED Strip or Light Setup**
6. **Connecting Wires & Breadboard**
7. **Power Supply (Battery or Adapter)**

Software

- **Arduino IDE** (for coding and uploading firmware)
- **LoRa Library** for communication

4. Understanding LoRa Communication

4.1 How LoRa Works

LoRa (Long Range) is a wireless communication protocol that allows for **long-distance, low-power** data transmission. Unlike Wi-Fi or Bluetooth, LoRa operates using **spread spectrum modulation** to ensure high **interference resistance** and **low data rates**.

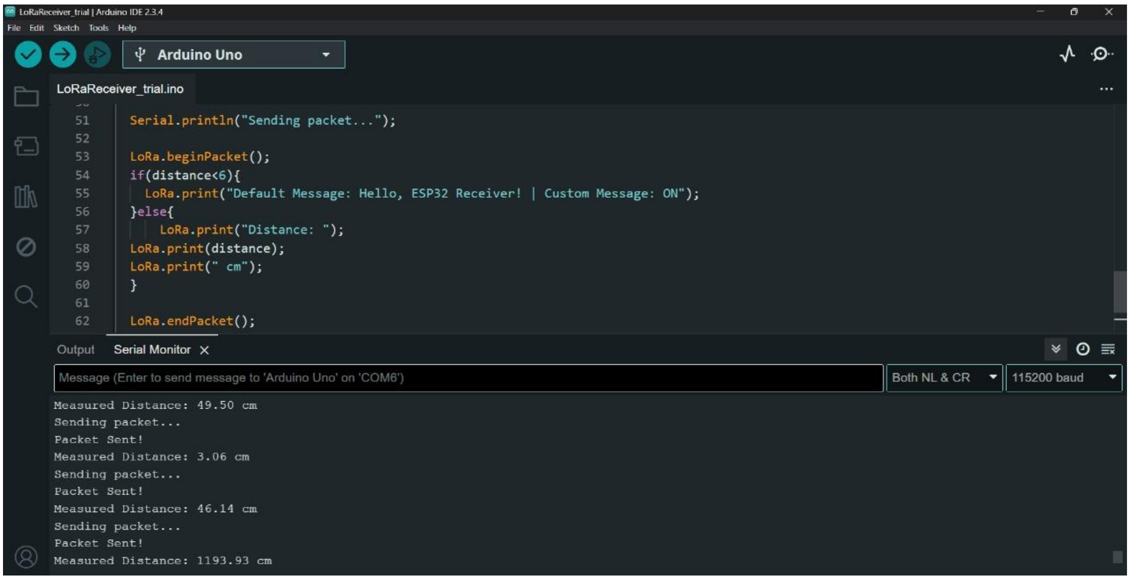
4.2 SPI Communication Setup

LoRa RA-02 modules communicate with microcontrollers using **SPI (Serial Peripheral Interface)**. The SPI pins are as follows:

- **MOSI (Master Out Slave In)**: Data sent from the microcontroller to the LoRa module.
- **MISO (Master In Slave Out)**: Data received from the LoRa module.
- **SCK (Serial Clock)**: Synchronizes data transmission.
- **NSS (Chip Select)**: Enables communication between the microcontroller and the LoRa module.

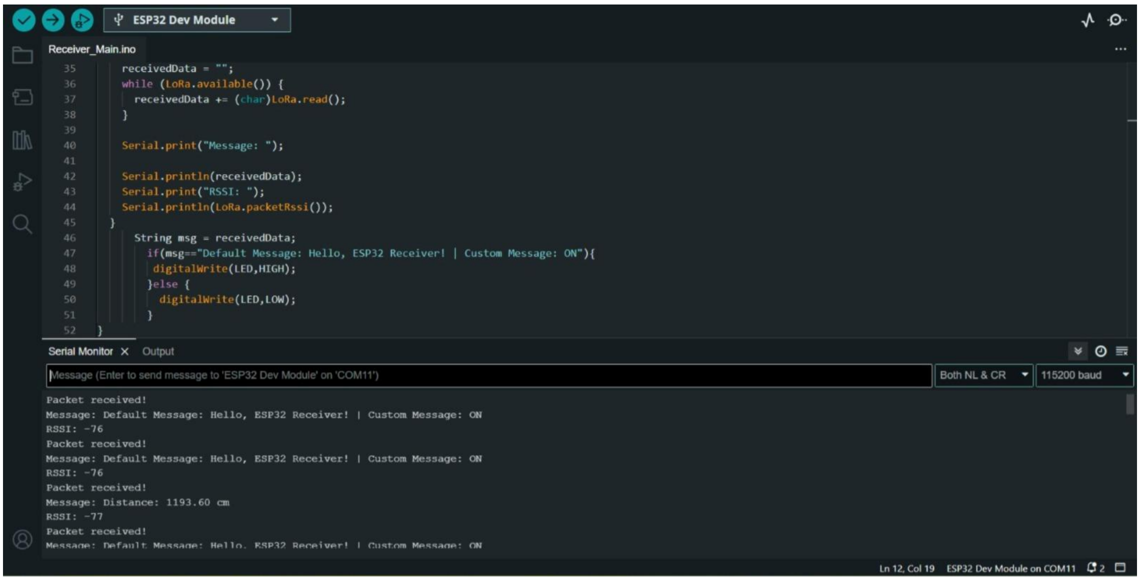
For **Arduino UNO**:

- MOSI: Pin 11
- MISO: Pin 12
- SCK: Pin 13
- NSS: Any digital pin (customizable)



For **ESP32**:

- MOSI: GPIO 23
- MISO: GPIO 19
- SCK: GPIO 18
- NSS: Any digital pin (customizable)



4.3 Frequency and Bandwidth

LoRa operates in different frequency bands depending on the region:

- **868 MHz** (Europe)
- **915 MHz** (North America)
- **433 MHz** (Asia & Custom Use Cases)

For this project, we set the **frequency to 433 MHz** (or adjust based on regional requirements). Bandwidth and spreading factors impact **range and power consumption**:

- **Higher bandwidth (e.g., 500 kHz)** allows for faster data rates but reduces range.
- **Lower bandwidth (e.g., 125 kHz)** increases range but limits data speed.

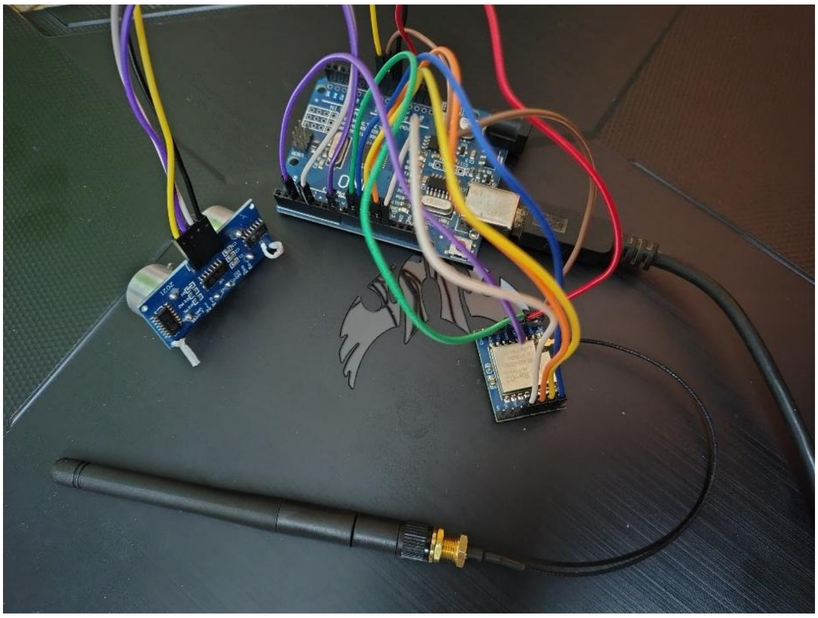
4.4 RSSI and SNR Values

- **RSSI (Received Signal Strength Indicator)**: Measures signal strength received by the ESP32.
    - Values range from **-30 dBm (strong signal) to -120 dBm (weak signal)**.
- **SNR (Signal-to-Noise Ratio)**: Evaluates signal clarity.
    - **Positive SNR** means a strong signal, while **negative SNR** indicates high noise interference.

5. System Architecture

5.1 Transmitter (Arduino UNO):

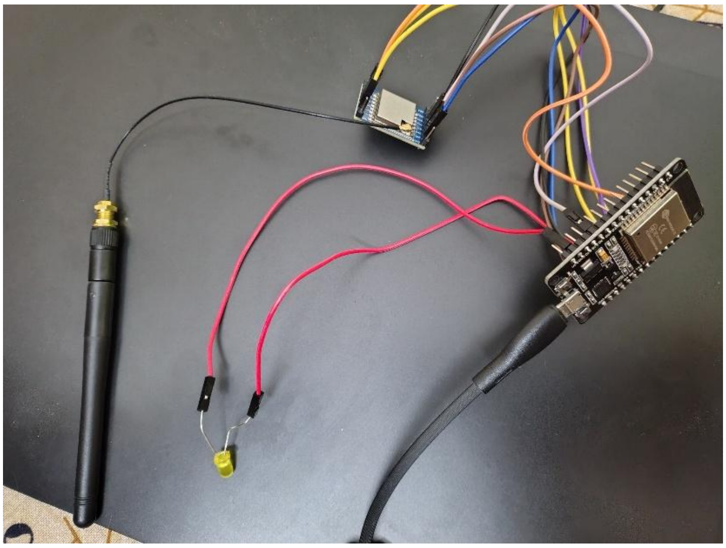
- Measures **distance using an ultrasonic sensor**.
- If an object is placed **within a defined threshold (e.g., <10cm)**, it sends a signal via **LoRa**.
- Sends default messages at intervals if no object is detected.



5.2 Receiver (ESP32):

- Continuously **listens for LoRa signals**.
- If a valid signal is received, it **turns on the LED lighting system**.
- If no object is detected for a certain duration, it **turns off the lights** after a delay.

- Logs **RSSI and SNR values** for debugging and range testing.



6. Implementation Details

6.1 Transmitter (Arduino) Code Overview

- Initializes **LoRa module** and **ultrasonic sensor**.
- Measures **distance** at regular intervals.
- Sends a **LoRa packet** when an object is detected.
- Allows **manual messages via the serial monitor**.

6.2 Receiver (ESP32) Code Overview

- Initializes **LoRa module**.
- **Listens** for incoming packets.
- If an object detection signal is received, it **activates the LED**.
- If the object is removed, it **deactivates the LED** after a set delay.
- **Reads RSSI and SNR values** for analysis.

7. Working Mechanism

1. **User places an object (bag) on a rack.**
2. **Ultrasonic sensor detects the object** and measures distance.
3. **LoRa transmitter sends data to the receiver (ESP32).**
4. **ESP32 turns ON the LED lighting system.**
5. **When the object is removed, ESP32 turns OFF the lights after a delay.**

8. Challenges and Learnings

- **Simultaneous transmission issue:** LoRa is a simplex communication protocol, meaning both devices **cannot send data at the same time**.
- **Message acknowledgment (ACK) not implemented:** Future upgrades could include bidirectional communication for **confirmation messages**.
- **Power consumption considerations:** Optimizations such as **low-power mode** can be explored.
- **Signal strength and range testing:** LoRa's **RSSI and SNR** were analyzed to ensure **reliable transmission**.

9. Future Enhancements

- Implement **ACK messages** for reliable transmission.
- Use **low-power modes** to extend battery life.
- Add **multiple sensors** for enhanced automation.
- Develop a **mobile app interface** to monitor and control the lighting system remotely.

10. Conclusion

This project successfully demonstrates **long-range, low-power wireless communication** using LoRa for **automated lighting control**. The setup is efficient, scalable, and can be further enhanced with advanced features such as **two-way communication, cloud integration, and remote control**.
