Overview

This project implements a **reliable text file transfer system** over LoRa using two **ESP32-S3** boards running **MicroPython**. The system allows uploading a `.txt` file via a web interface on the **sender** node, splits it into packets, and transmits them over LoRa to a **receiver** node. Each packet is acknowledged (ACKed) by the receiver, ensuring **reliable and complete** data transfer. The receiver displays the reconstructed file content on its own web interface.

🧠 Features

- 📁 File upload via web interface (HTML form)
- 📦 Chunked transmission of file content
- ✅ Per-packet acknowledgment (ACK) for reliable delivery
- 🧩 Packet reassembly at the receiver
- 🌐 Simple HTTP server interface for both sender and receiver
- ⚠️ UTF-8 safe messaging for cross-platform compatibility
- 📝 Final file preview on the receiver's web page

🧰 Hardware Required

- 2× ESP32-S3 boards with LoRa (SX1262)
- USB cables
- Computer with **Thonny IDE**
- Wi-Fi-enabled device (for file upload via web UI)

🔧 Software Stack

- MicroPython (custom firmware flashed to ESP32-S3)
- HTML (web interface)
- SX1262 LoRa library
- Thonny (for development and deployment)

🚀 Getting Started

1. Flash MicroPython Firmware

Use esptool or Thonny to flash MicroPython on both ESP32-S3 boards.

2. Upload Files

- Use Thonny's **Files** tab to upload `main.py` and `index.html` to each board.
- Reboot both ESP32 boards after uploading.

3. Connect to Access Point

- The sender board will create a Wi-Fi SoftAP (`LoRaSenderAP`, default IP: `192.168.4.1` `LoRaReciverAP`, default IP: `192.168.4.1`)
- Connect via your browser and visit `http://192.168.4.1` to upload your text file and Download text file

🔁 How It Works

1. **Sender node** serves an HTML page via HTTP, allowing the user to upload a `.txt` file.
2. Once uploaded, the content is split into fixed-size chunks (e.g., 100 bytes).
3. Each chunk is sent over LoRa with a header including:
    - Packet index
    - Total packets
4. **Receiver node** listens for incoming packets:
    - Sends ACK for each valid packet
    - Stores packets in order
    - Reconstructs original content after receiving all packets
5. Final content is displayed on the receiver's web page.