📤 Sender Algorithm (Upload → Chunk → Send with ACK)

🧠 Purpose:

To accept a text file via a web interface, break it into chunks, and transmit those chunks reliably over LoRa with acknowledgment from the receiver.

🔄 Steps:

1. **Web Interface Setup**
    - ESP32 starts as a Wi-Fi access point (AP) and serves a simple HTML page.
    - User uploads a `.txt` file via an HTML form.
2. **Receive File on Server**
    - MicroPython HTTP server parses the multipart form.
    - The uploaded text file is saved in the ESP32 file system (e.g., `example.txt`).
3. **Chunking the File**
    - The file is read and split into fixed-size chunks (e.g., 100 bytes each).

Each chunk is tagged with:
`[chunk_index]/[total_chunks]|[chunk_data]`

4. **LoRa Transmission with ACK**
    - For each chunk:
        - The sender transmits it via LoRa.
        - Waits for an ACK like `ACK:<chunk_index>` from the receiver.
        - Retries if ACK is not received within a timeout.

        - Stops after `MAX_RETRIES`.

5. **Final Message**
    - After all packets are sent, the sender may send an `END` or `EOF` marker.

🔁 Summary in Pseudocode:

python

```
chunks = split_file_into_chunks()
for i, chunk in enumerate(chunks):
    formatted = f"{i}/{len(chunks)}|{chunk}"
    retries = 0
    while retries < MAX_RETRIES:
        send(formatted)
        if wait_for_ack(i):
            break
        retries += 1
```

📥 Receiver Algorithm (Listen → ACK → Reconstruct)

🧠 Purpose:

To listen for incoming LoRa packets, acknowledge receipt, and reconstruct the original file after receiving all chunks.

🔄 Steps:

1. **Web Interface Setup**
    - ESP32 starts as a Wi-Fi AP and serves a basic webpage displaying the received content.
    - This is updated dynamically as packets are received.
2. **Listening for Packets**
    - The receiver continuously listens for LoRa messages.
    - On receiving a packet:
        - It parses the header to extract `chunk_index`, `total_chunks`, and `data`.
        - Sends back `ACK:<chunk_index>`.
3. **Storing Chunks**
    - Each chunk is saved in a dictionary with its index.
    - Duplicates are ignored.
4. **File Reconstruction**
    - When all `total_chunks` are received:
        - The chunks are ordered and concatenated.
        - The full text is saved (e.g., `received.txt`) and shown on the web interface.

🔁 Summary in Pseudocode:

python

```
received_chunks = {}
```

```
while True:

    packet = receive_lora()

    if is_valid_packet(packet):

        index, total, data = parse(packet)

        if index not in received_chunks:

            received_chunks[index] = data

        send_ack(index)


    if len(received_chunks) == total:

        reconstruct_file(received_chunks)
```