LoRa Communication for Image Transmission and Reception

Soumadeep De

February 23, 2025

Introduction

This paper presents a system for transmitting and receiving images using LoRa (Long Range) communication with Raspberry Pi Zero devices equipped with SX1276 LoRa modules. LoRa is a low-power, long-range communication technology that is suitable for applications that require efficient data transmission over significant distances. In this system, Raspberry Pi devices are used as LoRa nodes, where the transmitter sends image data, and the receivers reconstruct the image from the received packets.

The process involves converting an image into smaller packets, transmitting them over LoRa, and reconstructing the image on the receiver side. This system is optimized for low-power and long-range communication, making it suitable for remote and IoT applications. The first topology implements communication between one transmitter and one receiver, while the second topology utilizes two receivers, reducing transmission time by operating on two frequencies simultaneously.

Topology 1: 1 Transmitter — 1 Receiver

In Topology 1, the system consists of a single transmitter and a single receiver. The transmitter's task is to split the image into smaller packets, which are then transmitted sequentially using LoRa. The receiver listens for incoming packets, reconstructs the image, and saves or displays it. Each packet is sent in sequence, and the final packet includes an "ABC" end marker to signal the end of the image data.

The following algorithms describe the process for both the transmitter and the receiver:

Transmitter Algorithm for Topology 1

Algorithm 1 Image Transmitter Algorithm (1 Transmitter — 1 Receiver)

Initialize LoRa module with frequency 915 MHz for communication
Load image from file "Image_xx.jpg"
Compress the image using JPEG with quality 85
Convert the image to byte stream
Set packet size to 240 bytes (max packet size for LoRa)
LoRa communication can only transmit a limited amount of data per packet, so the image is divided into packets of 240 bytes each.
for each packet of image data in the byte stream **do**
    Convert each byte packet to an integer list to fit the LoRa packet format
    Send the packet over LoRa communication to the receiver
    The packet is transmitted over LoRa communication to the receiver.
    Wait for 1 second to allow receiver to process data
    A small delay is introduced between transmissions to give the receiver time to process the data.
end for
Send the final packet with identifier "ABC" to indicate the end of the transmission
The final packet contains the "ABC" marker, signaling that the image transmission is complete.
Exit the transmission loop
The transmission process ends after sending all packets.

Receiver Algorithm for Topology 1

Algorithm 2 Image Receiver Algorithm (1 Transmitter — 1 Receiver)

Initialize LoRa module and set frequency to 915 MHz
The receiver is initialized to listen for incoming data on the same frequency as the transmitter (915 MHz).
Set LoRa mode to receive (RX)
The LoRa module is set to receive mode so it can accept incoming packets in interrupt mode.
Initialize variable final_data to store received data
Initialize list rssi_values to store RSSI values
while True **do**
    Wait for incoming packets
    On receiving a packet:
        Store packet data in final_data
    Each incoming packet is appended to the final data variable.
        Store RSSI value in rssi_values
    The RSSI value for each packet is recorded to assess signal quality.
    if final_data contains "ABC" **then**
        Reconstruct the image from final_data
        Once the "ABC" marker is detected, the receiver knows that all packets have been received, so the image is reconstructed.
        Save the reconstructed image as "Received.jpg"
        Calculate performance metrics such as RSSI, SNR, BER, and PRR
        The receiver finishes the process after saving the image and calculating the performance metrics.
        Exit the receiver loop
    end if
end while

Topology 2: 1 Transmitter — 2 Receivers

In Topology 2, the system consists of one transmitter and two receivers operating at different frequencies. The transmitter alternates between two frequencies, 915 MHz for the master receiver and 914.5 MHz for the slave receiver. This allows the master and slave receivers to process different parts of the image data concurrently, reducing the total transmission time.

The following explains how the system works:

- The transmitter alternates between the two frequencies, sending packets of image data on each frequency.
- The master receiver listens on 915 MHz, while the slave receiver listens on 914.5 MHz.
- The data is received by both receivers in parallel, and once all packets are transmitted, the final data is combined from both receivers to reconstruct the image.
- The slave receiver sends its received data to the master receiver via UART for combining.

This method significantly reduces transmission time by utilizing both receivers in parallel, thus improving the system's efficiency.

Transmitter Algorithm for Topology 2

Algorithm 3 Image Transmitter Algorithm (1 Transmitter — 2 Receivers)

Initialize LoRa module with frequency 915 MHz
Load image from file "Image_xx.jpg"
Compress the image using JPEG with quality 85
Convert the image to byte stream
Set packet size to 240 bytes
Set frequency alternation between 915 MHz (master) and 914.5 MHz (slave)
The transmitter will alternate between the master and slave frequencies for sending the packets.
for each packet of image data in the byte stream **do**
    if packet index is even **then**
        Set frequency to 915 MHz for master receiver
    else
        Set frequency to 914.5 MHz for slave receiver
    end if
    Send the packet over LoRa communication
    The packet is transmitted over LoRa communication to the receiver.
    Wait for 0.5 seconds to allow processing by receivers
    A short wait time is added between transmissions to allow for processing by the receivers.
end for
Send the end marker "ABC" at both frequencies
The final marker is sent to indicate the end of the transmission.
Exit the transmission loop
The transmission process ends after sending all packets.

Master Receiver Algorithm for Topology 2

Algorithm 4 Master Receiver Algorithm (1 Transmitter — 2 Receivers)

Initialize LoRa module with frequency 915 MHz
Set LoRa mode to receive (RX)
Initialize variable final_data to store received data
while True **do**
    Wait for incoming packets
    On receiving a packet:
      Store packet data in final_data
    if final_data contains "ABC" **then**
        Wait for slave receiver to finish receiving data
        Combine the received data from master and slave receivers
        Reconstruct and save the image as "ReceivedImage.jpg"
        Calculate performance metrics
        Performance metrics like RSSI, SNR, BER, and PRR are calculated.
        Exit the receiver loop
        The process is complete, and the loop ends.
    end if
end while

Slave Receiver Algorithm for Topology 2

Algorithm 5 Slave Receiver Algorithm (1 Transmitter — 2 Receivers)

Initialize LoRa module with frequency 914.5 MHz
Set LoRa mode to receive (RX)
Initialize variable final_data to store received data
while True **do**
    Wait for incoming packets
    On receiving a packet:
      Store packet data in final_data
    if final_data contains "ABC" **then**
        Send the data over UART to master receiver
        Exit the receiver loop
        The process completes after sending the data to the master.
    end if
end while

The communication between the master and slave receivers is further optimized using Ready to Receive (RTR) logic. The RTR mechanism helps prevent data overflow in the UART buffer of the Raspberry Pi, which is limited to 4096 bytes. This is especially crucial when large images are transmitted.

The RTR logic ensures that the slave does not send data to the master faster than it can be processed, helping maintain synchronization and prevent data loss. The slave checks a GPIO pin (configured as an input) to determine whether the master is ready to accept more data. If the pin is **LOW**, the slave holds off on sending the next packet until the master signals it is ready.

This mechanism prevents the UART buffer from overflowing and ensures that each chunk of data is properly processed before the next one is sent.

Conclusion

This paper demonstrated two different topologies for image transmission using LoRa communication with Raspberry Pi Zero devices and SX1276 LoRa modules. The first topology involved one transmitter and one receiver, while the second topology used two receivers working in parallel on different frequencies to reduce transmission time.

Both systems effectively show how LoRa can be used for low-power, long-range image transmission. The second topology, in particular, offers a significant improvement in reducing transmission time by utilizing multiple receivers. This setup has potential applications in remote sensing, IoT networks, and other fields requiring efficient long-range communication.

Furthermore, the introduction of Ready to Receive (RTR) logic ensures reliable data transfer by managing the UART buffer and preventing overflow, especially when transmitting large images. This is crucial for maintaining synchronization between the receivers and ensuring that each packet is processed in a timely manner.

The results of this study show that LoRa communication, coupled with strategic optimizations like dual receivers and RTR logic, can be a highly effective solution for efficient image transmission in remote and low-power environments.