# LoRa-Based Image Transmission System

## Introduction

In this project, we have developed a cutting-edge system for transmitting images over a LoRaWAN network. The system leverages a Raspberry Pi Zero 2W, a LoRa pHAT node from RAK, and a USB-connected camera to capture, process, and transmit images efficiently. The Raspberry Pi takes pictures using the attached camera, converts pixel data into an optimized binary format, and sends it to The Things Network (TTN) using the LoRaWAN protocol. TTN acts as an intermediary cloud platform, forwarding the data to a laptop via a webhook integration enabled by ngrok. On the laptop, the binary information is processed to reconstruct the original image and updated dynamically as new data is received.

The system operates in an infinite loop, continuously capturing new images, identifying changes compared to the previous image, and transmitting only the affected pixel data. The first image is sent entirely, while subsequent transmissions focus on pixel differences, drastically reducing the amount of data sent after the initial transmission.

This project demonstrates the practical application of LoRaWAN in multimedia communication, highlighting how even limited bandwidth networks can handle image data transmission effectively. The approach balances the constraints of low-power IoT devices with the need for real-time multimedia updates, making it suitable for various applications like remote monitoring, surveillance, and industrial IoT.

## Components Used in the LoRa Node

### Raspberry Pi Zero 2W

The Raspberry Pi Zero 2W is the computational core of the LoRa node, responsible for handling all image processing, optimization, and communication tasks. This compact single-board computer offers the following features:

* Processor: Equipped with a 1 GHz quad-core ARM Cortex-A53 processor, the Pi Zero 2W is capable of handling computationally intensive tasks such as image resizing, pixel difference calculation, and binary data optimization.
* Memory: With 512 MB of RAM, it supports efficient operation of the Python programs required for this project.
* Connectivity: The Pi Zero 2W offers USB connectivity, which facilitates the use of a USB webcam for capturing images.
* Power Efficiency: Designed for IoT applications, its low power consumption makes it ideal for battery-powered deployments in the field.

The Raspberry Pi runs the Python-based LoRa node program, integrating image capture and data optimization seamlessly with LoRa communication.

### LoRa pHAT Node from RAK

The LoRa pHAT node from RAK brings LoRaWAN communication capability to the Raspberry Pi. This module supports the US915 frequency band and features:

* Robust Long-Range Communication: The LoRa pHAT is designed to transmit data over several kilometers, even in challenging environments.
* Low Power Consumption: Essential for IoT devices, it minimizes energy usage while maintaining reliable data transmission.
* Compatibility with TTN: The module seamlessly integrates with The Things Network, leveraging its cloud capabilities for data routing and storage.

The module is configured through the Raspberry Pi using the **rak811** Python library, which simplifies communication tasks like joining the network, setting operational parameters, and sending data.

### Camera Module

The camera module used in this project is a USB-connected webcam, chosen for its ease of integration and versatility. Key features include:

* Resolution: The camera captures images at resolutions up to 128x128 pixels, which are resized as needed for this project.
* Image Format: Images are saved in JPEG format, balancing quality and storage efficiency.
* Integration: Controlled using the fswebcam command-line tool, the camera captures images automatically in the required format, facilitating smooth operation with the Raspberry Pi.

The camera provides real-time visual data, which forms the basis for image transmission and difference detection.

### LoRa Node Program

The LoRa node software operates as follows:

*Image Capture:*

* The program captures an image using the camera and saves it as a JPEG file.
* The image is resized to 128x128 pixels to reduce the data size while maintaining sufficient detail for transmission.

*1D Pixel Data Optimization:*

* The image is processed row by row to identify horizontal runs of pixels with the same color. Each run is represented using the parameters:
  + x1 and x2: Start and end x-coordinates of the run.
  + y: The y-coordinate of the row.
  + r, g, b: RGB values of the pixels in the run.
* This approach significantly reduces the amount of data to be transmitted, especially for images with large uniform areas.

*Difference Calculation:*

* For subsequent images, the program compares the current image with the previous image pixel by pixel.
* The ImageChops.difference() function identifies the pixels that have changed. Only these "affected pixels" are included in the transmitted data, further reducing the data size.

*Data Transmission:*

* The optimized or difference data is divided into chunks, each containing up to 10 pixel segments.
* The first chunk of each transmission includes metadata specifying the total number of chunks.
* Data is sent to TTN in an infinite loop, ensuring continuous updates.

## Data Retrieval Using ngrok

### What is ngrok?

ngrok is a tunneling tool that allows local services to be exposed to the internet securely. It plays a critical role in this project by enabling the Flask server running on the laptop to receive data from TTN via webhook.

### Workflow

#### Local Server Setup:

A Flask server is hosted on the laptop, listening for incoming POST requests on port 5000.

#### ngrok Integration:

* ngrok is configured to forward requests from a public URL (e.g., https://<ngrok\_subdomain>.ngrok.io/webhook) to the local Flask server (<http://localhost:5000/webhook>).
* This public URL is used as the webhook endpoint in the TTN application settings.

#### Data Security and Monitoring:

* ngrok provides HTTPS encryption for secure data transfer.
* A web dashboard offers real-time monitoring of incoming requests, helping debug and validate the data flow.

By bridging TTN and the laptop, ngrok ensures seamless data transfer in real time.

## Laptop Code: Image Reconstruction

### Overview

The laptop code is designed to:

1. Receive image data from TTN via the webhook.
2. Decode and process the binary data.
3. Reconstruct and update the image in real time.

### Workflow in Detail

#### Receiving Data via Flask:

* + The Flask server handles POST requests from TTN. Each request contains Base64-encoded binary data in the frm\_payload field.
  + The data is decoded and passed to the process\_chunk() function.

#### Chunk Processing:

* + Each chunk contains multiple segments of pixel data in the format (x1, x2, y, r, g, b).
  + The data is unpacked using Python's struct module.
  + For each segment, the corresponding pixels in the reconstructed image are updated with the specified RGB values.

#### Image Reconstruction:

* + The reconstructed image starts as a blank 128x128 RGB image.
  + Segments are applied sequentially, updating the image in real time as data is received.

#### Dynamic Image Saving:

* + The first full image is saved as reconstructed\_image0.png.
  + Updated images are saved incrementally as reconstructed\_image1.png, reconstructed\_image2.png, and so on.

#### Counters and State Management:

* + Total chunks and processed chunks are tracked to ensure complete reconstruction of each image.
  + Once all chunks for an image are processed, counters are reset for the next image.

### Features

* **Real-Time Reconstruction**: Updates are applied as soon as data is received, minimizing latency.
* **Incremental Saving**: Each reconstructed image is saved separately, enabling step-by-step visualization of changes.

## **Conclusion**

This project demonstrates an innovative approach to transmitting and reconstructing images over a LoRaWAN network. By combining efficient data optimization techniques, robust communication protocols, and real-time image reconstruction, the system balances bandwidth constraints with performance. The use of ngrok and TTN simplifies cloud integration, while the modular Python programs ensure adaptability for future enhancements. This system is a testament to the potential of LoRaWAN in multimedia and IoT applications, paving the way for advancements in remote monitoring and low-power communication.