A Long-Range Transmission Network for Animal Sighting in the Wilderness

By

Yan Zhang

A Project Report Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

University of Victoria

SUPERVISORY COMMITTEE

A Long-Range Transmission Network for Animal Sighting in the Wilderness

By

Yan Zhang

University of Victoria 2023

Supervisory Committee

Dr. Kin Fun Li, Department of Electrical and Computer Engineering

Supervisor

Dr. Hong-Chuan Yang, Department of Electrical and Computer Engineering

Departmental member

Table of Contents

Acronyms vi

Acknowledgements vii

Abstract 1

Chapter 1 Introduction 2

1.1 Project Objectives 3

1.2 Report Outline 3

Chapter 2 Related Works 4

2.1 Background of LoRa and LoRaWAN 4

2.2 Characteristics of LoRaWAN 7

2.3 Comparison of LoRa and other LPWANs 9

Chapter 3 System Design 12

3.1 Architecture 12

3.2 Camera-embedded System 14

3.2.2 Movement Detection 16

3.2.3 Animal Identification 17

3.2.4 Concurrent Multi-threads for Animal Identification 18

3.3 LoRa Nodes Construction 24

3.3.1 End node 24

3.3.2 Relay node 24

3.4 Gateway 27

3.5 LoRaWAN Service 28

Chapter 4 Experimental Evaluation 30

4.1 Camera-embedded System Performance 30

4.2 LoRa Device Deployment 31

4.3 LoRaWAN Service 33

4.4 Client 34

4.4.1 LoRa Frames on Web 34

4.4.2 Local Web UI through Wi-Fi Hotspot 35

4.5 Implementation 36

Chapter 5 Conclusions 38

List of Figures

Figure 1.1. System Modules…… 2
Figure 2.1. LoRa Symbol and Chips…… 5
Figure 2.2. LoRa and LoRaWAN…… 7
Figure 3.1. LoRa Relay Network Architecture …… 13
Figure 3.2. Data Flow Chart……14
Figure 3.3. System Design Structure ……15
Figure 3.4. System Flow Chart……16
Figure 3.5. MS COCO Dataset Examples……18
Figure 3.6. Camera-embedded System Structure……22
Figure 3.7. Circular Queue A……23
Figure 3.8. Circular Queue B……23
Figure 3.9. LoRaWAN Class A Models……25
Figure 3.10. LoRaWAN Packet Structure……26
Figure 3.11. LoRaWAN Class C Model……27
Figure 3.12. LoRaWAN Gateway Design Structure……28
Figure 3.13. LoRaWAN Timing Diagram……30
Figure 4.1. Animal Identification with annotation……31
Figure 4.2. Deployment of Camera and LoRa Components……32
Figure 4.3. LoRa Gateway Location……33
Figure 4.4. LoRa Service Component Status……34
Figure 4.5. LoRa Gateway Data on Web Console……34
Figure 4.6. LoRa Data Saved in Redis……35
Figure 4.7. LoRa Web Client……35
Figure 4.8. Web Client UI……36
Figure 4.9. Star Network and Relay Network……37

List of Tables

Table 2.1. Relationship of Primary Parameters of LoRa in Theory……6

Table 2.2. Overview of LPWANs: LoRaWAN, NB-IoT, Sigfox and Wi-Fi HaLow……11

Table 3.1. LoRa Feed Structure……14

Table 3.2. Specification of the Camera……17

Table 3.3. Movement Detection Process……18

Table 4.1. Parameters of Antenna……34

Table 4.2. Equipment Cost……38

Acronyms

ABP: Activation By Personalization
ACK: Acknowledgement
ADR: Adaptive Data Rate
AES: Advanced Encryption Standard
AI: Artificial Intelligence
BPSK: Binary Phase Shift Keying
BR: Bit Rate
BW: Band Width
CMOS: Complementary Metal-oxide Semiconductor
CODR: Coding Rate
CPU: Central Processing Unit
CRC: Cyclic Redundancy Check
CSI: Camera Serial Interface
CSS: Chirp Spread Spectrum
DATR: Data Rate
DB: Database
DevAddr: Device Address
EUI: Extended Unique Identifier
FCtrl: Frame Control
FDMA: Frequency-Division Multiple Access
FHDR: Frame Header
FOpts: Frame Options
FPGA: Field Programmable Gate Arrays
FPS: Frames Per Second
FSK: Frequency Shift Keying
GPIO: General Purpose Input/Output
GPS: Global Positioning System
RPC: Remote Procedure Calls
HTTP: Hypertext Transfer Protocol
IoT: Internet of Things
IP: Internet Protocol
ISM: International Safety Management
JIT: Just-in-Time
JPG: Joint Photographic Experts Group
JSON: JavaScript Object Notation
LoRa: Long Range
LoRaWAN: Long Range Wide Area Network
LPWAN: Low Power, Wide Area Network
LSNR: Link Signal-to-Noise Ratio
LTE: Long-Term Evolution
MAC: Medium Access Control
MHDR: Medium Access Control Header
MIC: Message Integrity Code
MIMO: Multiple-Input Multiple-Output
MS COCO: Microsoft Common Objects in Context
MQTT: Message Queuing Telemetry Transport
NB-IoT: Narrowband Internet of Things
OFDM: Orthogonal Frequency-Division Multiplexing
OpenCV: Open-Source Computer Vision
PHDR: Physical Layer Header
PHY: Physical
QPSK: Quadrature Phase Shift Keying
RC: Rate Code
RF: Radio Frequency
RSSI: Received Signal Strength Indicator
RTOS: Real-Time Operating System
SF: Spreading Factor
SMA: Sub-Miniature version A
SNR: Signal-to-Noise Ratio
SPI: Serial Peripheral Interface
SSL: Secure Sockets Layer
TF: Trans-Flash
TFL: Tensor Flow Lite
TPEE: Thermoplastic Polyester Elastomer
TTN: The Things Network
TTS: The Things Stack
UART: Universal Asynchronous Receiver/Transmitter
UI: User Interface

Acknowledgements

I would like to express sincere gratitude to my supervisor Dr. Kin Fun Li for invaluable guidance and enlightening support throughout my research project. His insights and expertise were instrumental in shaping the direction of my research, and his dedication to teaching and research is an inspiration to me.

I am truly grateful for the time that Professor Li dedicated to meeting with me and discussing my research. He provided me with valuable feedback on my research proposal, which helped me to refine my research questions and design.

Thank you Professor Li, for all that you have done for me.

Abstract

When wild animals are monitored in the vast wilderness of Canada, data transmission is considerably challenging due to the lack of effective network service provided by telecom operators or carriers, especially in sparsely populated areas. A Long-Range Transmission Network for a wildlife detection system using low-power and low-cost embedded software and hardware is designed and implemented. The objective of the system is to transmit the results of wildlife identification with environmental data through independent long-range networking. The system consists of a Camera-embedded System for wildlife image capturing and environmental data logging, a user system for scanning images and notifications, and a LoRaWAN networking for Long-Range Transmission. Once a targeted animal is detected and identified, the system issues an alarm in the monitored area and sends a LoRa data frame to an application server for further analysis and user notification. The transmission distance of data is effectively extended through the relay between nodes. The system can process up to nine frames per second from the camera and identify the designated wildlife with high accuracy by asynchronous multi-threading in a low-cost embedded system. The application could be beneficial for a variety of purposes in the vast and diverse wilderness areas, such as traffic alarms for large wild animals' crossing, monitoring wildlife migrations by biologists, or a warning system in urban areas when there is a potential threat to the public such as approaching dangerous animals.

Chapter 1 Introduction

In recent years, there has been an increasing demand for accurate and reliable data on wildlife movements and behaviors to inform conservation efforts and improve wildlife management practices. However, traditional monitoring methods may be costly or not able to carry out long-distance transmission in the wild areas. Therefore, longer-range and more cost-effective methods are needed to monitor wildlife. For instance, detecting wild animals in vast national parks or keeping animals off the highway has a strong demand for long-range and low-power IoT data transmission.

The objective of the system is to transmit wildlife identification with environmental data through long-range network. As shown in Figure 1.1, the system consists of five functional modules. Wild animals are identified on the captured images by AI technology, and environmental data is collected through various sensors, such as temperature, humidity, air pressure, light intensity, atmospheric composition, noise intensity, etc. When animals are detected by movement detection algorithms, TensorFlow Lite inference is used for identification, which are encapsulated in LoRa frames and transmitted through the LoRaWAN network to the application servers. If the target animals are identified, alarms are raised and users are notified immediately.

Figure 1.1. System Modules

Processed images with environmental data are saved in the database. Users can download them through Wi-Fi HaLow within 1 kilometer. The system provides the option to transmit data with AES128 encryption to ensure security.

The proposed system can be used for a variety of purposes, including monitoring animal populations as well as habitation and providing early warning of potential threats to public traffic, parks, and communities.

1.1 Project Objectives

The major project objective is to design and deploy a long-range communication network system for low-power and low-cost transmission of environmental data and animal identification results in the wilderness, which can be summarized as follows:

1. Design a long-range and low-power consumption wireless network system, which can be strategically deployed to cover large wilderness areas.
2. Implement a relatively low-cost embedded system for detecting and monitoring animal movement in remote wilderness areas with environmental sensors and alarms.
3. Develop data processing and analysis services, including machine learning algorithms for animal classification and identification of meaningful insights from the sensor data.
4. Test and validate the system in simulated or real-world wilderness environments to ensure it is robust.
5. Improve the system's functionality and performance. Evaluate the potential limitations and challenges associated with the system and the potential for false positives.

The ultimate goal of the project is to provide a valuable tool for wildlife identification and protection. By detecting and monitoring animal movement in remote wilderness areas, the proposed system can help improve our understanding of animal populations and behaviors, guiding efforts to conserve animals and protect ourselves.

1.2 Report Outline

The remaining chapters are structured as follows: Chapter 2 explains the technical background and selection reasons of LoRaWAN and compares the advantages and disadvantages of the four prevailing LPWAN IoT communication methods. Chapter 3 describes the essentials of system design and analyzes the modules and LoRaWAN network architecture. Chapter 4 shows the actual deployment and results of the project as well as the verification in a real setting. Chapter 5 concludes and proposes future works.

Chapter 2 Related Works

Since LoRa technology was invented and launched in 2012, it has been widely used in the long-distance information transmission of agriculture [4] and urban applications [7], while animal classification and identification algorithms also have a relatively mature application framework. On this basis, this project applies low-power embedded technologies to achieve high frame rate monitoring and provides a visual data client and real-time alarm system.

2.1 Background of LoRa and LoRaWAN

LoRa is a Physical layer protocol using the Chirp Spread Spectrum (CSS) technique [1]. LoRa provides for long-range communications, with respect to ranges, up to 3 miles in dense urban areas and up to 10 miles or more in rural areas. A key characteristic of the LoRa-based solutions is ultra-low power requirements, which allow for the creation of battery-operated devices that can last for up to 10 years. Deployed in a star topology, a network based on the open LoRaWAN protocol is perfect for applications that require long-range or deep in-building communication among a large number of devices that have low power requirements and that collect small amounts of data.

LoRa uses CSS technology for long-range and low-power transmission, which are used in military and space communication for decades due to its robust nature. This allows the network to preserve battery life for connected end nodes through adaptive optimization of individual end node power levels and data rates. CSS is a signal whose frequency increases or decreases over time because the frequency change over time creates a frequency modulation that spreads the signal's energy over a wide frequency range, which means that the signal can be transmitted using lower power without sacrificing the signal's quality or data rate.

CSS uses a wide bandwidth to transmit a narrowband signal, which means that the transmitted signal is spread over a large frequency range. This allows multiple signals to be transmitted simultaneously without interfering with each other. Chirp signals have a higher signal-to-noise ratio (SNR) than other types of signals, which makes them more resistant to noise and interference. This is because the frequency modulation of the signal spreads the energy over a wider frequency range, which makes it easier to distinguish the signal from background noise.

As a result, the spectral efficiency of CSS is high, which means that it can transmit data using a relatively low transmission power.

The LoRa modulation bitrate can be expressed as the relation between bandwidth BW, spreading factor SF and Rate Code [2] [9]:

$$R_b = SF \frac{Rate\ Code}{\frac{2^{SF}}{BW}} \quad [\text{bit/s}]$$

Rb: Bitrate refers to the data transfer rate in bits per second. In IoT applications, Bitrate is generally low, usually between hundreds of bps to several kbps.

BW: Bandwidth is usually 125kHz. Smaller bandwidths provide longer communication distances and higher sensitivity, but communication speeds may be reduced.

SF: The spreading factor is usually between 7 and 12. A higher spreading factor can improve the transmission range and signal-to-noise ratio, whereas the communication speed may be decreased.

Rate Code: Rate code for encoding typically uses four levels, 4/5, 4/6, 4/7, and 4/8, where 4/5 provides the highest data rate and lowest redundancy, and 4/8 provides the lowest data rate and highest redundancy. In this project, 4/5 is used as usual.

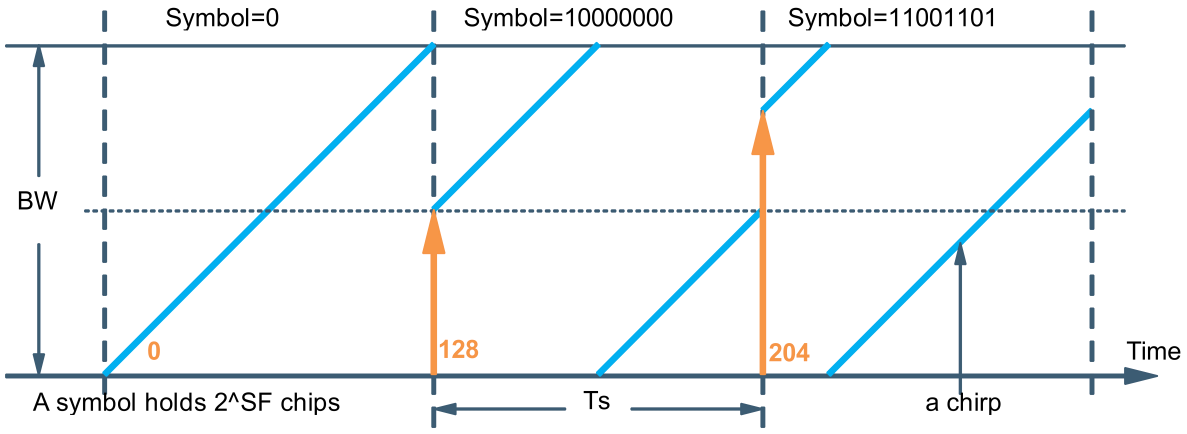


Figure 2.1. LoRa Symbol and Chips

Figure 2.1 shows the relationship between SF, Symbol, chirps, chips, and data bits in LoRa CSS [3]. Each blue slash is a chirp. Each symbol represents SF bits, and each symbol is divided into 2^SF chips, Ts=2^SF/BW, and the value of the chips rate is equal to the value of BW. An increase in Rate code and SF values decreases the effective data rate and causes an increase in time on air of the LoRa frame. The data in Table 2.1 shows the theoretical transmission Range and the consumed Time on Air achieved by LoRa data transmission corresponding to different parameters [8]. Among them, Bit Rate indicates the data transmission rate, Range indicates the

transmission distance, and Time on Air indicates the data frame transmission time. With the support of a perfect antenna, when LoRa uses a larger SF parameter, the longer the transmission range, the longer the corresponding transmission Time on Air. In the current project, SF=12 is used uniformly as a parameter of LoRa. For example., with a payload of 27-byte, the size of the LoRa frame is 40-byte, as BW=125 kHz, the relationships of parameters are shown in Table 2.1 [8] [13]:

SF	Chips per Symbol	Bit Rate (kbps)	Time on Air (ms)
7	128	5.47	58.5
8	256	3.13	102
9	512	1.76	182
10	1024	0.98	326
11	2048	0.54	592
12	4096	0.29	1103

Table 2.1. Relationship of Primary Parameters of LoRa in Theory

To meet the needs of building large-scale IoT accessing Internet scenarios, LoRaWAN (Long Range Wide Area Network), is an open networking protocol located in the Media Access Control (MAC) layer that delivers secure bi-directional communication and promotes LoRa signals to wider applications, designed for large-scale networks. LoRaWAN implements based on the physical layer technology of LoRa that enables long-range communication, and LoRaWAN converts the LoRa frame into a protocol format acceptable to the Internet sending it to the server. LoRaWAN devices use low-power communication methods, support multi-device connections, and provide end-to-end encryption and authentication mechanisms.

Figure 2.2 shows the networking layer relationship between LoRa and LoRaWAN. The LoRaWAN specification defines the MAC protocols, and the PHY layer mainly uses LoRa modulation or FSK modulation for some frequency bands.

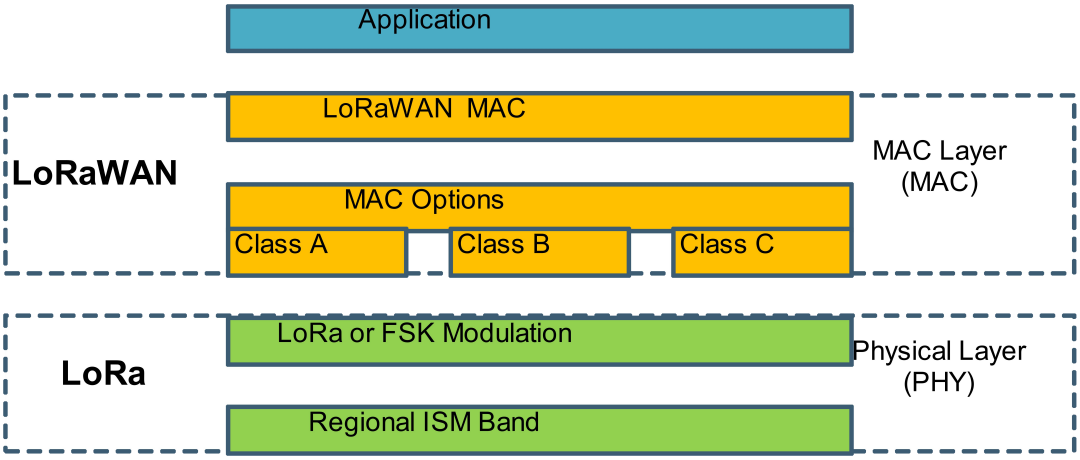


Figure 2.2. LoRa and LoRaWAN

2.2 Characteristics of LoRaWAN

LoRaWAN is a data link layer protocol developed by LoRa Alliance in 2015 to provide a low-power connectivity solution to battery-powered devices. LoRaWAN's modulation method is based on Chirp Spread Spectrum technology adopted by LoRa, which can realize long-distance data transmission under the premise of low power consumption. LoRaWAN nodes have low power consumption characteristics, which can run for a long time and support battery power. Furthermore, LoRaWAN adopts a random-access protocol, so nodes do not need to pre-negotiate access rights when sending packets directly to the network. This method may cause multiple nodes to send data at the same time, resulting in conflicts. Therefore, a certain collision detection and retransmission mechanism is required to ensure the reliability of data transmission. LoRa frame confirmation mechanism and conflict detection mechanism are used to improve the stability and reliability of the network and achieve low power consumption, related features including:

1. LoRaWAN uses Adaptive Data Rate (ADR) ALOHA [15] as the MAC (Medium Access Control) protocol, which is a variant of the Aloha-style multiple access technique. The ADR dynamic adjustment [21] of data rate and flexible configuration options is better suited for the diverse and challenging communication environments that LoRaWAN aims to address. It enables LoRaWAN to adapt to varying channel conditions and optimize performance for different IoT use cases. By default, LoRaWAN employs an adaptive data rate mechanism based on ALOHA, which dynamically adjusts the data transmission rate according to channel quality and device transmission conditions. This means that LoRaWAN devices can select the appropriate Spreading Factor (SF) and Bandwidth (BW) based on the actual

communication environment and channel quality to optimize communication performance and energy efficiency. When communication distances are far or channel quality is poor, LoRaWAN devices may use higher spreading factors and narrower bandwidths to achieve better signal propagation and longer communication ranges. Conversely, when communication distances are closer or channel quality is better, LoRaWAN devices can use lower spreading factors and wider bandwidths to support higher data rates. Through this adaptive approach, LoRaWAN can achieve flexible data transmission in various communication environments while considering both communication range and energy efficiency requirements.

2. Sleep mode in LoRaWAN means that the device enters a low-power mode to save energy. In sleep mode, the device will turn off some unnecessary circuits or reduce the circuit frequency to reduce power consumption. The sleep mode can be entered by the device autonomously, or it can be triggered by a command issued from the gateway in the network. Class A, Class B, and Class C [12] in LoRaWAN are used to describe the communication behaviors and power sleep management methods of devices.
    a. Class A is the most common device class in LoRaWAN, which is shown in Figure 3.9. In Class A, the communication behavior of devices is asynchronous to save power consumption. After sending data, the device waits for a fixed time window to receive downlink data, and then enters sleep mode to reduce power consumption. In this mode, the device consumes the lowest power consumption and is suitable for most low-power, low-latency application scenarios.
    b. Class B equipment adds an additional receiving window on the basis of Class A to receive beacon signals at fixed time intervals for time synchronization and additional downlink data reception. Class B devices go into sleep mode intermittently. The power consumption of the device in this mode is relatively high.
    c. Class C devices are the most flexible, and their receive window is always open for real-time communication without sleep. Class C devices can continue to receive downlink data when not sending data, and the power consumption is the highest. This mode is suitable for application scenarios that require fast response and real-time communication.

In summary, LoRaWAN's Class A is the most energy-efficient device category, suitable for most low-power application scenarios, which can effectively reduce the power consumption of the device and prolong the battery life. Class C provide higher real-time capability and flexibility, but relatively high-power consumption. Choosing the appropriate device class should be a trade-off based on application requirements and power consumption constraints. The Class A mode is mainly used in end nodes, while the class C mode is used in relay nodes.

2.3 Comparison of LoRa and other LPWANs

LPWANs (Low-Power Wide Area Networks) [5] are a class of low-power, long-distance communication networks designed for Internet of Things (IoT) applications. LPWANs have lower power consumption and thus can extend the battery life of IoT devices. LPWANs also have a wide coverage area, which can cover communications within several kilometers. There are many ways to implement LPWANs, currently mainly including, NB-IoT, Sigfox, Wi-Fi HaLow, LoRaWAN, and so on. Due to their advantages, they are suitable for different application scenarios.

NB-IoT stands for Narrow Band Internet of Things and is based on the existing LTE network infrastructure. Theoretically, the transmission rate of NB-IoT can reach more than 100 kbps, and the delay is lower than that of LoRa. NB-IoT is based on FDMA and therefore requires infrequent synchronization, which results in higher battery consumption than LoRa technology. NB-IoT has better latency and data rate, so those applications that require low latency and high data rate can use NB-IoT. However, NB-IoT requires service fees and depends on the deployment of operator network base stations, and there may be no signal in wilderness areas. It should be noted that common cellular networks like 4G are designed for high data throughput, and the construction cost of large-scale base stations is extremely high, which is not suitable for long-range, low-frequency, small-scale data transmission in wild fields.

Sig Fox supports narrowband technology using a standard radio transmission method called Binary Phase Shift Keying. The transmission distance of Sigfox can reach 50 kilometers, and the transmission time in a day is very short, which is suitable for fields without real-time communication requirements. Although the applicable application scenarios of Sigfox and LoRa are very similar, for example, both LoRa and Sigfox have the characteristics of long-distance and low power consumption, which can prolong battery life and form a large-scale message

transmission. Both have the several same features, like using the license-free Sub-1GHz ISM frequency band, without additional licensing fees, due to breakthroughs in electronic chip manufacturing technology, and the hardware manufacturing costs of the two technologies continue to decrease, but the business model and technical principles behind the company are completely different. The most typical difference is that Sigfox needs the base station equipment of the mobile service provider. It requires inexpensive endpoint radios and more complex base stations to manage the network. Therefore, Sigfox cannot be used in remote and underground areas where mobile signals cannot be covered, such as underground operations like mining and tunnel excavation, mountainous areas, etc., although the platform solution is complete. Whereas LoRa does not need to pass through the operator's base station, so there is no geographical restriction, and people can build and manage the network by themselves, and the cost is low.

Wi-Fi HaLow is IEEE802.11ah, with a maximum bandwidth of 86.7MHz and a transmission distance of around a 1km radius. Wi-Fi HaLow employs multiple-input multiple-output (MIMO) technology and other techniques to improve its transmission range, reliability, and energy efficiency. Wi-Fi HaLow has many of the same advantages as traditional Wi-Fi technology, such as high bandwidth and ease of use. It does not need a dedicated gateway to directly access the Internet. It natively supports IP networks, but chip suppliers are scarce and costly. Compared with other LPWAN technologies, Wi-Fi HaLow still consumes relatively high-power consumption, which may affect the battery life of devices. The Wi-Fi HaLow signal transmission range is about 1km, which is far inferior to LoRa and Sigfox. However, its relatively high bandwidth can effectively realize the transmission of data such as video and images, so it is used in the Camera-embedded System of this project to realize the transmission of images and real-time streaming.

LoRaWAN is based on LoRa technology which is highly resistant to multipath and fading and provides long-range communication. Because its frequency uses coding gain to improve receiver sensitivity and it has an absolute advantage in terms of battery power and hardware price, LoRa has an advantage at the receiving end. LoRaWAN supports the establishment of base stations and networks by itself, with a higher degree of freedom. For instance, with a well-designed antenna, a single LoRaWAN gateway can cover an area of 20 kilometers. LoRaWAN devices that use Class A mode to send 20 messages per day can last up to ten years on standby. The four prevailing LPWAN networks are compared in Table 2.2.

Empty Cell	LoRaWAN	NB-IoT	Sigfox	Wi-Fi HaLow
Modulation	CSS	QPSK	BPSK	OFDM
Frequency	Unlicensed ISM bands (868 MHz in Europe, 915 MHz in North America, and 433 MHz in Asia)	Licensed LTE frequency bands	Unlicensed ISM bands (868 MHz in Europe, 902 MHz in North America)	750~930 MHz
Sensitivity	-120~-140 dBm	-130 dBm	-126 dBm	-106 dBm
Data Rate	0.3~50 kbps	20-127 kbps	100 bps or 600 bps	150 kbps~86.7 Mbps
Bandwidth	250 kHz and 125 kHz	200 kHz	100 Hz	1,2,4,8MHz
Bidirectional	Yes / Half-duplex	Yes / Half-duplex	Limited / Half-duplex	Yes
Max payload length	243 bytes	1600 bytes	12 bytes (UL), 8 bytes (DL)	8,191 bytes
Range	5 km (urban), 15 km (rural)	1 km (urban), 10 km (rural)	10 km (urban), 40 km (rural)	1km
Encryption	AES 128bit	LTE	Not supported	WPA3 &TLS
Adaptive data rate	Yes	No	No	Yes
Localization	Yes (TDOA)	No (under specification)	Yes (RSSI)	Yes (RSSI, ToF, AoA)
Private Network	Yes	No	No	Yes
Standardization	LoRa-Alliance	3GPP	ETSI	Wi-Fi Alliance
Spectrum Cost	Free	Commercial	Free	Commercial
Carrier Service Fee	Free	$5-7/device year	$5-7/device year	Free
Deployment device cost	>$150/gateway >$1500/base station	>$25000/base station	>$6000/base station	>$100/router
End-device cost	$7-10	>$30	<$5	$40-70

Table 2.2. Overview of LPWANs: LoRaWAN, NB-IoT, Sigfox and Wi-Fi HaLow.

To sum up, in terms of functions, LoRaWAN, NB-IoT, and Sigfox are all narrow-band and suitable for wide-area transmission with low-power consumption, while WiFi HaLow bandwidth is sufficient for image streaming transmission but not suitable for long-range networking. In terms of cost, both NBIoT and Sigfox rely on telecom carriers to deploy base stations and pay a carrier service fee according to the annual equipment quantity, but the independent network construction of LoRaWAN and WiFi HaLow makes the cost optimal. However, the network maintenance cost with the stability of LoRaWAN should be considered. Therefore, this project is designed by combining LoRaWAN for low-power, low-cost, and long-range networking and WiFi HaLow for image transmission.

Chapter 3 System Design

3.1 Architecture

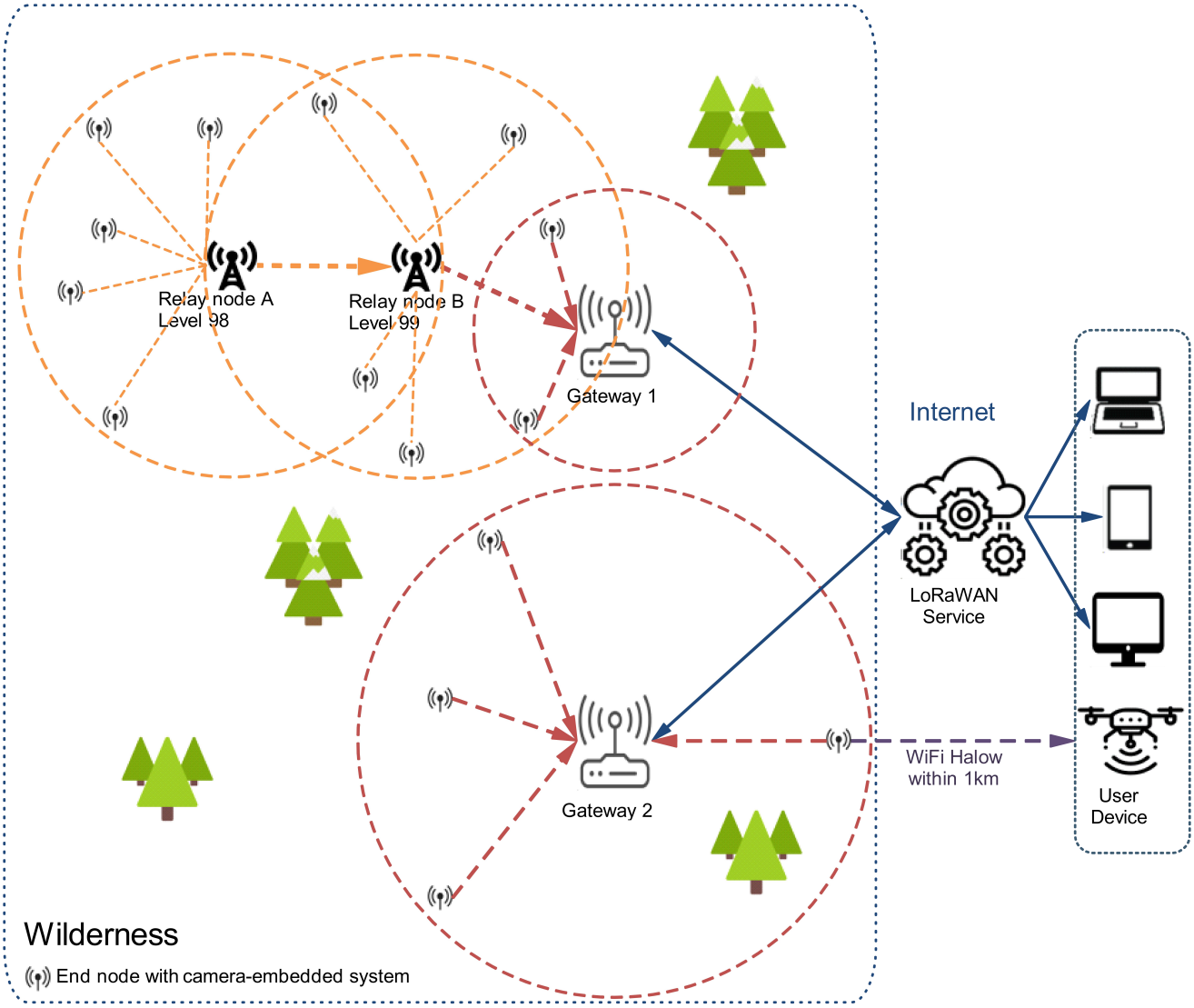


Figure 3.1. LoRa Relay Network Architecture

The LoRa Relay Network Architecture used in this project is shown in Figure 3.1. End nodes with a Camera-embedded System are widely distributed in wilderness fields. Images captured by a camera and environmental data collected by a variety of sensors are saved in the database of the Camera-embedded System. Feeds, as shown in Table 3.1, are created in a Camera-embedded System. Then a Feed as a member of a LoRa frame is encapsulated in an end node. LoRa frames

are transmitted to the gateway directly or via the low-to-high-level relay nodes for a longer range when the signal strength is not strong. The LoRaWAN transmission network that does not depend on the business carrier uses hierarchical relay nodes to realize long-distance transmission of LoRa frames. Then the LoRa frames are converted to JSON format data in the gateway and passed through the Internet to the cloud application servers. The processed images and sensor data saved in the database of the Camera-embedded System can be fetched directly through a Wi-Fi HaLow hotspot within a 1km area. User client software is provided to access the LoRa Feeds records through web service in the cloud.

```
struct Feed {
string      PicID;          #base64 string
string []   Labels;         #multi-label string array
float       Temperature;
float       Pressure;
float       Humidity;
float       Light;
uint        DiskUsage; }
```

Table 3.1. LoRa Feed Structure

The data format transmitted between each module is shown in Figure 3.2.

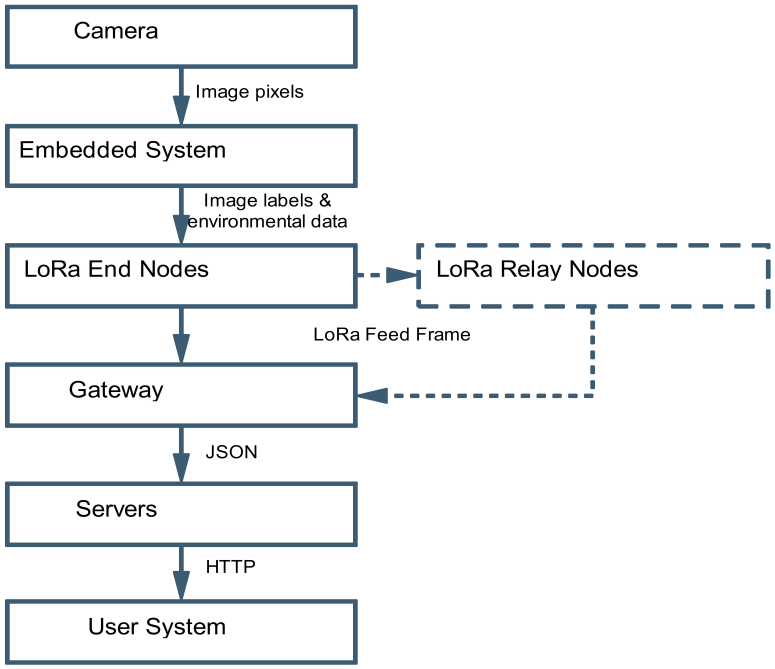


Figure 3.2. Data Flow Chart

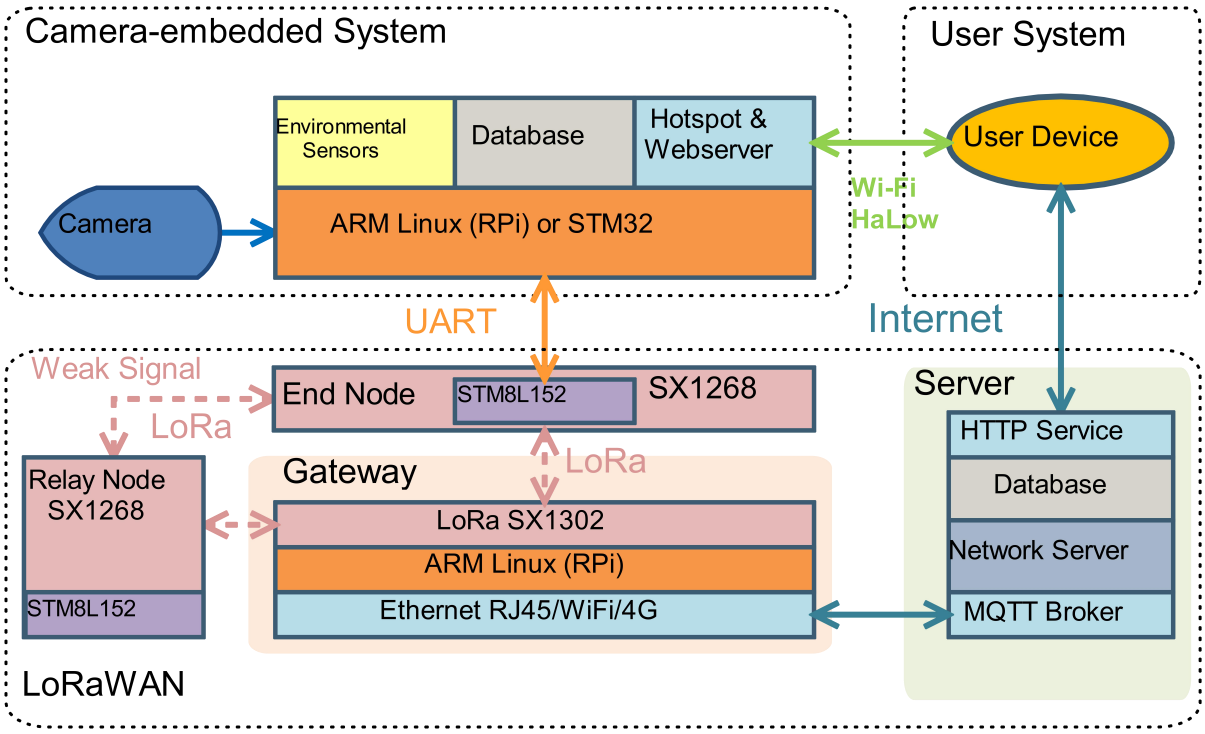


Figure 3.3. System Design Structure
(Dashed lines represent alternative communications)

The System Design Structure is shown in Figure 3.3, including Camera-embedded System, LoRaWAN Networking, and User System.

3.2 Camera-embedded System

The Camera-embedded System consists of a camera for wildlife image capturing, an ARM Linux system with a database, sensors for environmental data logging, and a Wi-Fi HaLow hotspot with web service. The Camera-embedded System is based on the ARM architecture and embedded Linux system where getting image frames and detecting movement are done by OpenCV functions. Artificial intelligence (AI) and deep learning techniques are used to identify animals. The TensorFlow Lite model is used for animal identification. Processed images and environmental data are stored in the database, which can be transferred to user devices through a Wi-Fi HaLow hotspot within 1km. User devices may be laptops, smartphones, tablets, drones, or specific hardware. Figure 3.4 demonstrates the whole process of image capturing and processing. If the identified animal is the target animal, a real-time application will run, such as an alarm to scare away animals. After drawing bounding boxes and annotating, the processed images and sensor data are stored, and the corresponding Feeds are transmitted simultaneously.

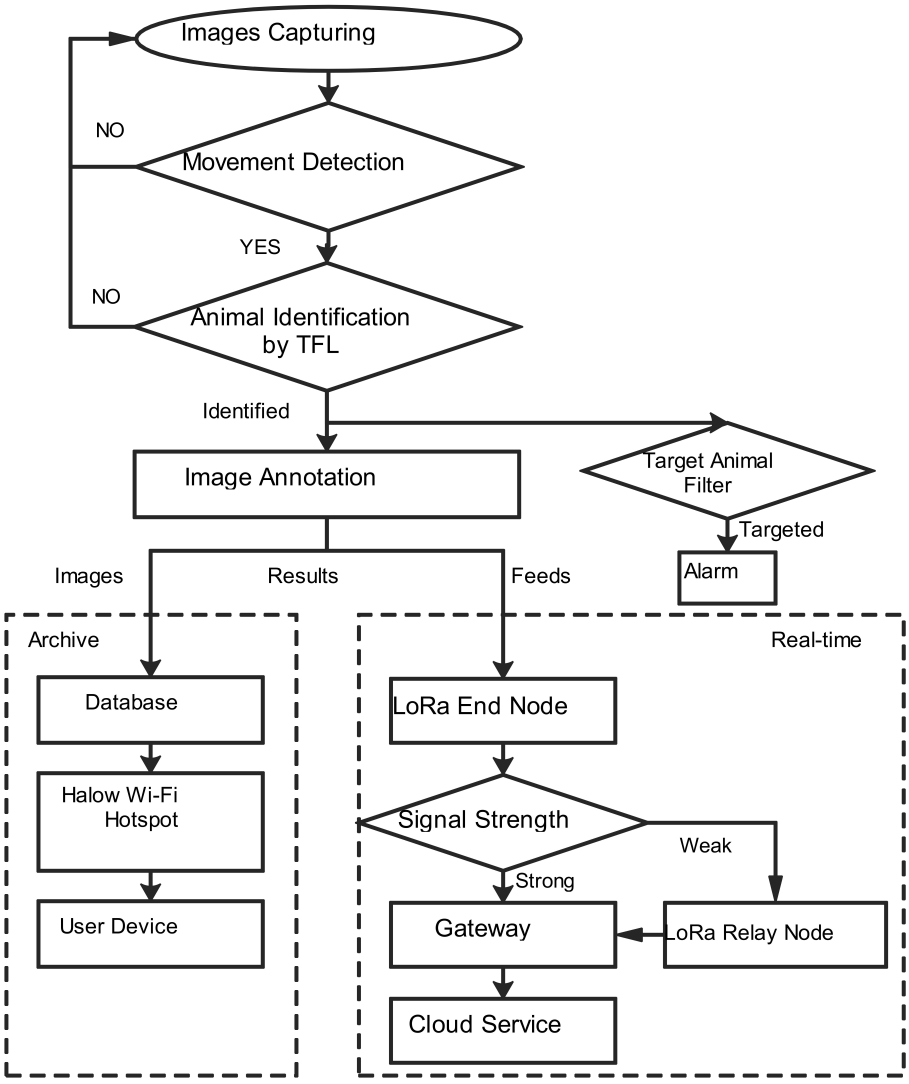


Figure 3.4. System Flow Chart

3.2.1 Camera

In this project, the camera adopts an OV5647 CMOS image sensor [14] that provides 2592*1944 image output. the camera is connected to the Raspberry Pi using the CSI dedicated high-speed interface. The specific parameters of the camera are shown in Table 3.2.

Considering the performance as well as the power consumption of the embedded system and actual requirements, after experimentation, images with a resolution of 640*480 and a 'jpeg' format are most suitable to be captured by OpenCV function for processing.

Under a light sensor, the camera can switch to an infrared filter for night view automatically, so the Camera-embedded System continues to capture and identify animals at night. However,

image processing and identifying effects at night based on infrared filters are not the objective of the current project and will be carried out in the future.

Camera	HBV-RPI-AUTO-IRCUT-3.6
Max pixels	5 million
Resolution	2592*1944
Angle of view	90°
Size	32*26.5*24 mm
CMOS size	1/4 inch
Aperture	1.2
Focal length	3.6 mm
Photosensitive chip	OV5647

Table 3.2. Specification of the Camera

3.2.2 Movement Detection

Animal identification will be performed when a moving animal is detected. The power consumption of an animal identification process is relatively high, if animal identification is performed on each captured image, it results in high system load and power costs. Therefore, using movement detection prior to identification can effectively reduce the overall power consumption of the system.

The movement detection algorithm performs differential operations on two consecutive frames of images and subtracts the pixels corresponding to different frames to determine the absolute value of the gray level difference as contours. When a contour value exceeds a certain threshold, e.g., 200 pixels, it can be judged as a moving animal. Specifically, the algorithm for implementing movement detection is described below:

1. Image frame capture. (by VideoCapture() function in OpenCV)
2. Image decode. (by imdecode() function in OpenCV)
3. Convert to grayscale. (by cvtColor() function in OpenCV)
4. Reduce image noise. (by GaussianBlur() function in OpenCV)
5. Image dilation operation. (by dilate() function in OpenCV)
6. Calculate the difference between the current frame and the previous frame. (by absdiff() function in OpenCV)
7. Find object contours. (by findContours() function in OpenCV)

8.	Traverse all contours. If the area of one contour is less than 200 pixels, ignore this contour and continue traversing for next contour, otherwise put the image into the queue for animal identification. (by contourArea() function in OpenCV) (200 pixels is selected from experiments as the image comparison threshold value of movement detection because it is based on the image with a resolution of 640 x 480 to ensure effective movement detection and avoid false positives under actual test experience.)

Table 3.3. Movement Detection Process

3.2.3 Animal Identification

Images whose movement detection is true will be processed by animal identification. Movement detection not only meets actual requirements but effectively reduces system load and power consumption. TensorFlow Lite is used for animal identification using the MS COCO (Microsoft Common Objects in Context) dataset [6], which contains photos of 91 common object categories with a total of 2.5 million labeled instances in 328k images.

Figure 3.5. MS COCO Dataset Examples (taken from [6])

Figure 3.5 shows the MS COCO dataset examples of complex everyday scenes containing common objects in their natural context. In this project, models trained on MS COCO are used

for animal identification. Specifically, a total of 11 types of animals in the MS COCO dataset are identified, including birds, dogs, cats, horses, sheep, cows, elephants, bears, zebra, giraffes, and raccoons.

When an image is identified as having animals, it will be stored in the database of the Camera-embedded System. In this project, after animal identification, a processed 640*480 image in JPG format is about 100KB. 120GB of database space in the Camera-embedded System is reserved for storing processed images, so no more than 1,200,000 processed images can be stored. When the number of images processed per second is set to FPS=5, assuming that 5% of the images are finally stored in the database through movement detection and animal identification, the number of images that can be stored per day is 21,600, that is, the 120G storage can be used for about 55 days. When the system increases the capacity of the database or reduces the FPS, it can effectively expand the storage to extend the duration. For example, as we use FPS=1 and a Database with 600G, the duration time can be extended to 1388 days, about 3.8 years. For remaining available space for the database in a Camera-embedded System, the usage of the database will be reported by Feeds. The Feed as LoRa frame payload is composed of the results of animal identification and environmental data as well as the usage of the database. Therefore, users can evaluate when to extract and delete images from the database depending on their usage. Once the database is full, the earlier images will be overwritten by the new images.

3.2.4 Concurrent Multi-threads for Animal Identification

It is found that processing Tensorflow Lite for animal identification is the bottleneck of the system. With a multi-core CPU, multi-threading can be used to improve image processing capabilities through asynchronous and concurrent design. The use of multi-threading can avoid the overall system time loss caused by process switching, which makes the design more appropriate for real-time situations.

In the Camera-embedded System, image capturing with movement detection is considered a producer, and output the images as tasks to the consumers which are the threads that execute animal identification by calling the TensorFlow Lite interface. The system design needs to ensure that consumers can always fully process the tasks from producers without causing a task backlog, especially in the worst-case scenario. The worst-case scenario means that every captured image is handed over to the processes of animal identification. Let the number of

producer processing threads be Np, the best time required to produce a single image as a task be Tp_b, the number of consumer processing threads be Nc, and the worst time required to process a single image task be Tc_w. To ensure that the consumer can fully process the output of the producer without task backlog, it needs to meet the following requirements:

Consumer worst capacity = Nc/Tc_w >= Producer best capacity = Np/Tp_b, (Capacity means the number of tasks that can be processed per unit of time)

Then Nc : Np >= Tc_w : Tp_b

Through experiments, we get that the best execution time of a producer is 0.11s, and the worst execution time of a consumer is 0.55s. Under the hardware of Raspberry Pi 3, the CPU has 4 cores. If the number of threads exceeds the number of CPU cores, the operating system needs to schedule each thread to execute on the physical core in turn in different time slices. The overhead of this context switching and scheduling increases because the operating system needs to switch thread contexts frequently and threads compete for limited CPU time slices. This can lead to degraded concurrency performance, as switching and scheduling overhead between threads becomes more expensive, while actual parallel execution time decreases. Therefore, the total number of threads of producer and consumer should not exceed the number of CPU cores. That is:

Since Tc_w : Tp_b = 55:11, and Nc+Np <= 4 cores ,

When Nc : Np >= 55 : 11 ,

Therefore, Nc=5 and Np=1 should be used in this system.

However, the CPU of Raspberry Pi 3 has only 4 cores, and the producer and consumer have 6 threads together, so problems are inevitable, such as thread blocking and waiting and relying on the operating system scheduling, resulting in insufficient real-time processing. Moreover, system design with the best producer state and the worst consumer state will not cause task backlog problems, but it will cause overdesign because in most cases the system will run in an average state. Through experiments, the average execution time of a consumer is 0.29s as Tc_a, and the average execution time of a producer is 0.19s as Tp_a. When we choose the average capacity of the consumer to process the output of the producer in the best state:

Since Tc_a : Tp_b = 29:11, and Nc : Np >= 29 : 11 .

Then Nc=3, Np=1.

When the number of consumer threads is reduced to 3 in this project, the number of producer and consumer threads are 4, which are respectively one-to-one bound to the 4 processing cores of the CPU of Raspberry Pi 3, so that the waiting caused by multi-threading preemption of computing resources can be avoided. Furthermore, as shown in Figure 3, using Circular Queue1 to replace old tasks and discarding timeout tasks are both protection strategies to ensure the real-time performance of the system, avoiding task backlog problems caused by abnormal fluctuations, which will be elaborated below. Therefore, the ratio of the number of threads between the consumer and producer is set to 3:1, which is completely feasible after experimental verification.

The average throughput of consumer and producer from experiments:

Consumer average capacity = Nc/Tc_a = 3/0.29 = 10 tasks per second,

Consumer best capacity = Nc/Tc_b = 3/0.23 = 13 tasks per second,

Producer average capacity = Np/Tp_a = 1/0.19 = 5 tasks per second,

Producer best capacity = Np/Tp_b = 1/0.11 = 9 tasks per second,

Max FPS = Min {Consumption best capacity, Production best capacity} = 9 fps

Average FPS = Min {Consumption average capacity, Production average capacity} = 5 fps,

The Average FPS represents the image processing capability of the system. Achieving higher FPS with the assistance of multi-threading ensures that the system can perceive moving animals and run timely applications.

As Figure 3.6 shows, there are 3 identical threads named ‘Thread-TFL’, as the consumer, to run animal identification with image annotation, sharing 3 cores of CPU. Another thread named ‘Thread-capture’, as producer, handles image capturing and movement detection using 1 core of the CPU.

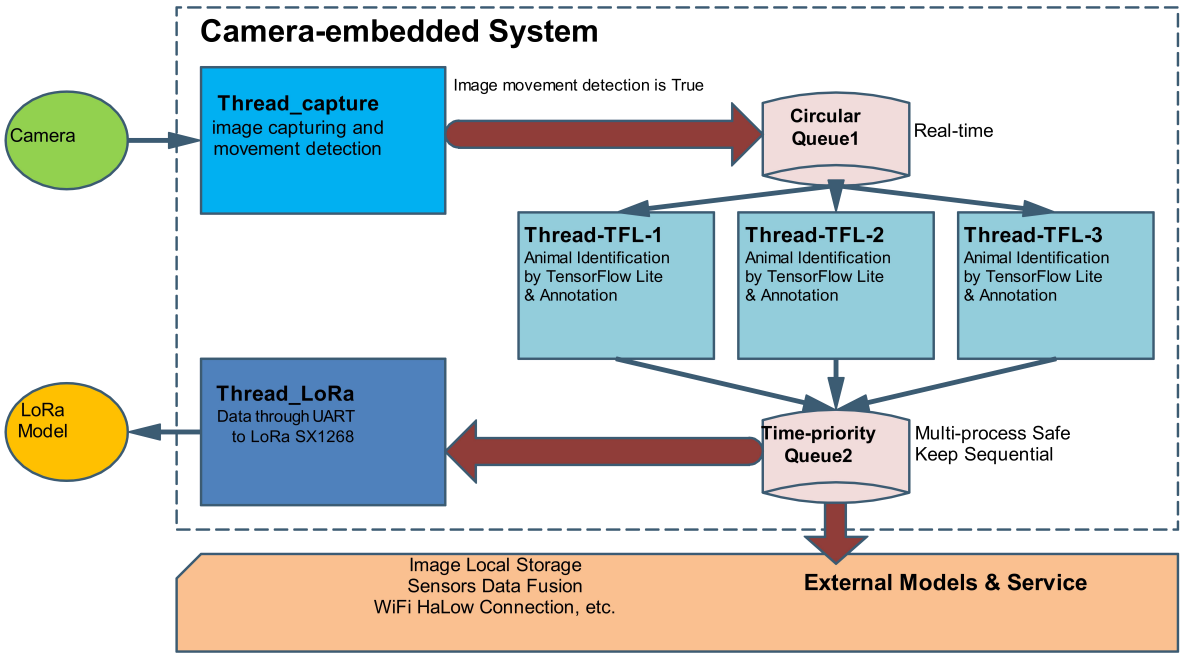


Figure 3.6. Camera-embedded System Structure

The delay fluctuations of the system are inevitable, when the system processes switching or thread lock contention occurs. Circular Queue1 is used to keep processing the latest tasks of images. When an image as a task to be processed is enqueued, the generation timestamp of the image is recorded. Before the image task is taken out of the Circular Queue1 for execution, it will be checked if the timestamp generated is timed out from the current time, that is:

current_time - image_timestamp < time_out

If the image task has been timed out, it will be discarded to ensure the real-time performance. Otherwise, the animal identification process will be performed. The 'time_out' can be configured according to user scenarios. In this project, 'time_out' is set to 0.3s to ensure all the images to be identified are generated within 0.3s. From the experimental experience, 0.3s as the timeout threshold can ensure real-time performance and avoid a large number of tasks being discarded due to timeout conditions. This is also one of the strategies to protect Circular Queue1 from creating image task backlog. Meanwhile, Circular Queue1 is a circular queue where new tasks from the producer are enqueued continuously. When the image task backlog exceeds the queue length, the latest image tasks will overwrite the oldest one to ensure the real-time performance of the system. The length of 'Circular Queue1' is set to 3, for there are 3 tasks that can be taken away by consumer threads at the same time.

The image task is denoted as a Task in Figure 3.7. Circular Queue1 shows that the producer generates Task3 and it is enqueued at the end of Circular Queue1. Task1 pointed by pointer Head is dequeued and it is handed over to the consumer for processing because its timestamp is still valid. When performing enqueue and dequeue operations, there are thread locks to protect the critical resources of Circular Queue1.

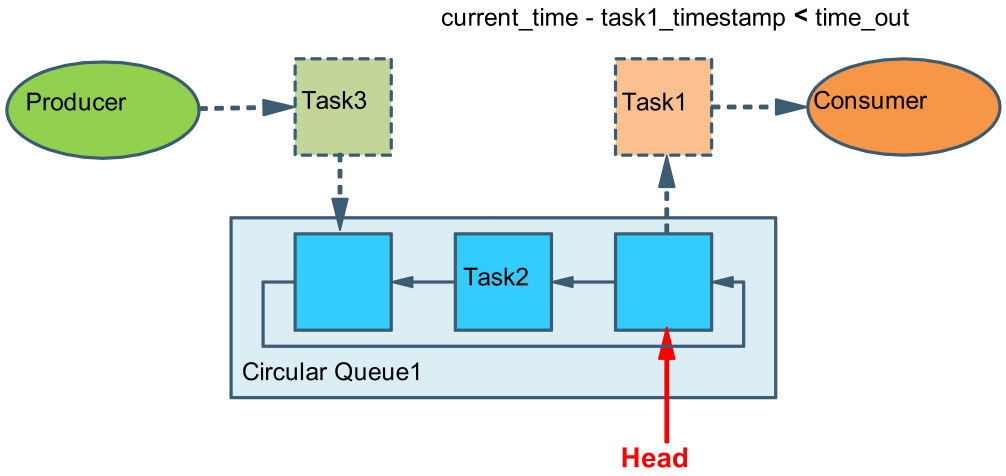


Figure 3.7. Circular Queue A

Figure 3.8. Circular Queue B shows when the producer generates Task4, the pointer Head moves backward one position to make new room for Task4 to enqueue. When Task2 is dequeued, Task2 is directly discarded for real-time performance because its timestamp is timed out.

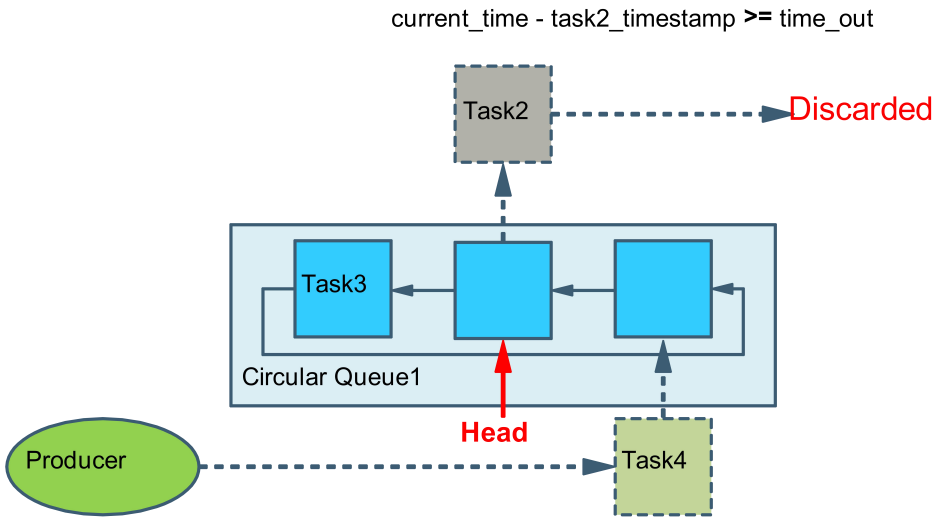


Figure 3.8. Circular Queue B

The images processed by concurrent Thread-TFL threads are put into Time-priority Queue2 which is based on a heap structure. Because concurrent Thread-TFL threads cannot ensure that

the order of task input is consistent with the order of output, the image generation timestamps are used as the key in the heap of Time-priority Queue2 in Figure 3.6 to make sure that the processed images are sorted to output and their dequeue sequence is identical with their generation order.

Environmental data derived from Sensors. Among them, temperature, pressure, and humidity are provided by the BME280 sensor which was developed by the German company Bosch Sensortec [17]. For temperature measurement, BME280 uses the internal temperature sensor to obtain the ambient temperature. It uses digital temperature compensation technology to provide accurate temperature measurement results by correcting and calibrating the raw data output by the sensor. The temperature measurement range of BME280 is usually between -40°C and 85°C, with humidity ranging from 0% to 100%, and air pressure ranging from 300 hPa to 1100 hPa. It has high accuracy and stability. Light strength is provided by the LTR-559 sensor which is developed by LITE-ON Technology Corporation [18]. It generates a corresponding electrical signal by detecting the intensity of light in the environment. The sensor's output signal can be converted to a numerical value of light intensity to provide an accurate measurement of ambient light intensity, ranging from 0.01 lux to 64k lux. In this project, the above two sensors are connected to the Raspberry Pi and transmit data through the I2C interface.

The Camera-embedded System uses the Wi-Fi HaLow module to provide a hotspot. The user client devices with the Wi-Fi HaLow module can connect to the Camera-embedded System within 1 kilometer and fetch the processed images saved in the database through Wi-Fi HaLow.

When the Camera-embedded System uses Raspberry Pi 3 for image processing which often involves plenty of calculation-intensive tasks, the CPU load reaches 60% usually, resulting in the average current range is 160-500 mA and the instantaneous peak value may reach 800 mA. Generally, we use a 20,000mAh lithium battery for power supply in experiments. Based on the average current of 300mA, the system can be used for about 66 hours. When outdoors, we use 5 sets of 20,000mAh lithium battery packs for power supply, which can be used for 2 weeks. However, the cost will increase significantly. In the future, it is planned to use a high-performance and low-power STM32 system, such as a module based on the STM32H7B3I chip, to replace the current Raspberry Pi 3, which can greatly reduce the system power consumption.,

and to use solar panels and rechargeable battery packs for power supply to meet the demand of long-term duration outdoor.

3.3 LoRa Nodes Construction

As shown in Figure 3.3, LoRa nodes are divided into end nodes and relay nodes, both of which are low-power and low-cost using Semtech SX1268 [10] RF chips for transmitting LoRa frames in this project. STM8L152 processors are used for LoRaWAN data encapsulating in end nodes and forwarding LoRa frames in relay nodes.

3.3.1 End node

End nodes are one-to-one connected to the Camera-embedded System, and the result of image processing as a LoRa Feed is transmitted from the Camera-embedded System through UART. End nodes use low-power STM8L152 to process LoRa frames encapsulation, which is transmitted through Semtech SX1268.

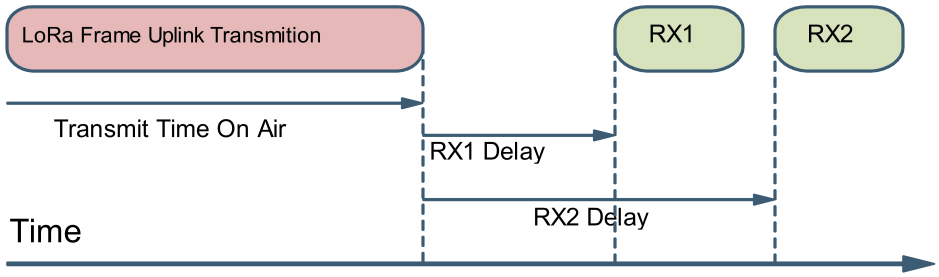


Figure 3.9. LoRaWAN Class A Model

Figure 3.9 shows that end nodes adopt the Class A model [12] to send data for saving energy. Once the uplink transmission is completed, the end nodes open two short receive windows in sequence, known as RX1 and RX2, waiting for a downlink message from gateways or relay nodes. As measured during the experiment, LoRa data frames usually are sent through Semtech SX1268 with transmit current of 110mA, receive current of 11mA, and sleep current of 2μA. End nodes have very low power consumption under the Class A model.

3.3.2 Relay node

When LoRa is used in a large area, the relay is required. Based on STM8L152, relay nodes are used for relaying LoRa data frames, for low power consumption and low cost. As Figure 3.1 shows, the LoRa relay nodes have a levels setting, which stipulates that LoRa frames must be

relayed from the lower level to the higher level. Levels of relay nodes can be changed by LoRa down frames remotely.

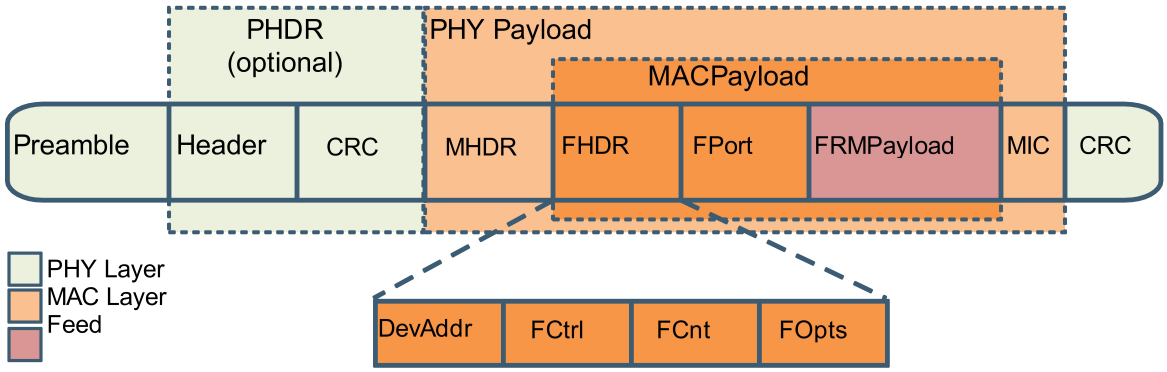


Figure 3.10. LoRaWAN Packet Structure

LoRaWAN employs a maximum frame size of 256 bytes. As shown in Figure 3.10, according to the protocol layer, the LoRaWAN frame is defined as follows [3]:

i. Preamble field is used for synchronization purposes.

ii. PHDR (Physical Layer Header): The PHY header contains information related to the physical layer settings, such as payload length, the spreading factor (SF), bandwidth (BW), coding rate, and frequency used for the transmission. The first byte of the header field specifies the payload length. The header is optional.

iii. PHY Payload: This field is used for MAC (Medium Access Control) layer and contains the following fields:

1. MHDR (Medium Access Control Header): defines the frame type (data or acknowledgment), protocol version, and its direction (uplink or downlink).
2. MACPayload: The FRMPayload field contains the actual application data LoRa Feeds sent by the end nodes. Maximum payload varies from 2 to 255 bytes. FHDR (Frame Header) is an essential part of the MAC layer frame structure, and it consists of the following fields:

a) DevAddr (Device Address): This field is 4 bytes long and uniquely identifies the end node within the LoRaWAN network. The network uses the DevAddr to identify the target device for downlink messages or to validate the source device for uplink messages.

b) FCtrl (Frame Control): This field is 1 byte long and contains control information related to the frame, such as information about the Adaptive Data Rate (ADR) flag, ACK (Acknowledgment) flag, and more.

c) FCnt (Frame Counter): This field is 2 bytes long and is used to keep track of the sequence number of the frame. Both the end node and the network maintain FCnt values to ensure the reliable delivery and sequencing of messages.

d) FOpts (Frame Options): This field is optional and can have varying lengths. It carries additional control information if needed for specific operations.

3. MIC: is used as the digital signature of the payload.

iv. CRC: is optional and comprises cyclic redundancy check (CRC) bytes for error protection for payload (2 bytes).

Specifically, DevAddr+FPort+FCnt fields of each LoRa frame as an ID are recorded in the relay node's ID-hashmap which keeps the latest 100 IDs of data frames that have been relayed. As Figure 3.10 shows, both DevAddr and FCnt fields are in the FHDR structure of MACPayload. If the latest received data frame ID is found in the ID-hashmap record, it will not be relayed again. The above can avoid the problem of LoRa data frames being sent repeatedly. Figure 3.11 shows the rules of the LoRaWAN Class C model [12], which is used by relay nodes to keep listening to all reachable LoRa frames from end nodes, whereas power consumption is continuously required. Relay nodes under the Class C model need two receive windows, known as RX1 and RX2. When a relay node sends a LoRa frame, a short RX2 receive window opens, followed by a short RX1 receive window, and then the reactive RX2 receive window remains open until the next LoRa frame transmission.

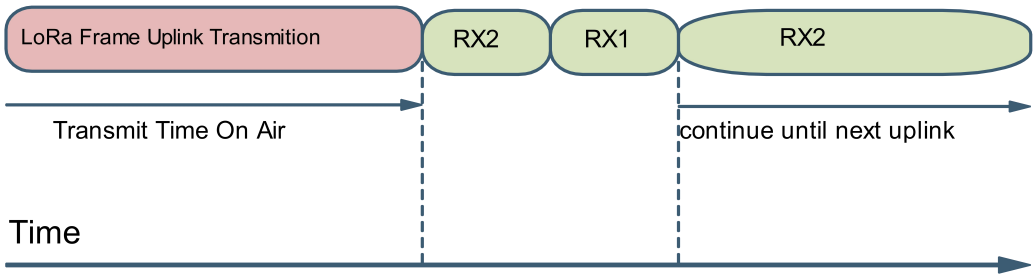


Figure 3.11. LoRaWAN Class C Model

3.4 Gateway

LoRaWAN wirelessly connects end nodes to the Internet through a gateway device. Gateways are used for forwarding packets to a Cloud Server through the Internet, without understanding the content of each packet. The purpose is to convert the LoRa data frame into a JSON packet and send it to the server via the Internet. There are 8 channels for receiving and forwarding by the SX1302 [11]. In this project, the LoRa SX1302 board is connected to the Raspberry Pi through SPI as a gateway. The Serial Peripheral Interface (SPI) is a communication protocol used to transfer data between the Raspberry Pi and peripheral devices. Specific GPIO pins of Raspberry Pi are used as the SPI interface [19].

A gateway is a two-way protocol converter, and the web server is responsible for decoding the packets sent by the device and generating the packets that should be sent back to the device. As Figure 3.12 shows, there are mainly three working threads in the gateway. Thread 1 receives LoRa frames from nodes and forwards them to the server. Thread 2 requests and processes the LoRa data packets from the LoRa server and inserts them into the JIT Queue. Thread 3 sends the LoRa data in JIT Queue to nodes through the LoRaWAN protocol. JIT (Just-in-Time) queue is a special software component implemented in the gateway, which is mainly used in the gateway to cache the data from the server to nodes. The data of the JIT queue is LoRa frames to be sent.

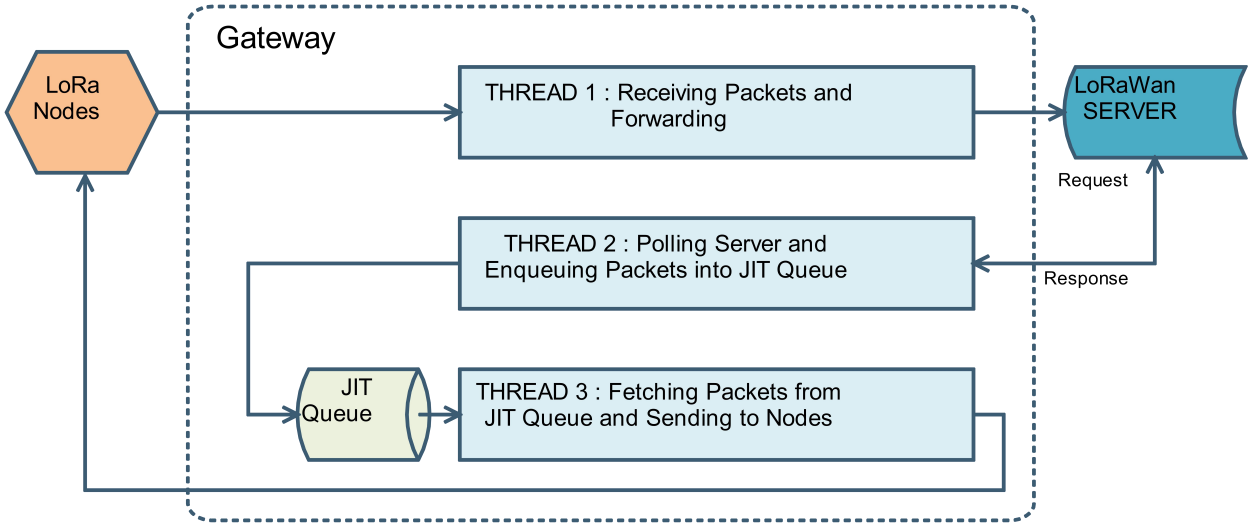


Figure 3.12. LoRaWAN Gateway Design Structure

3.5 LoRaWAN Service

The LoRaWAN-stack provides a robust and flexible architecture for receiving data from LoRaWAN gateways, supporting multiple protocols, and providing a scalable and extensible platform for LoRaWAN network applications. Servers are made to process LoRaWAN packets by Golang [20] which is a programming language created at Google in 2007 to improve programming productivity in multi-core networked machines and large codebases. A LoRaWAN service contains the following components and processes:

a) Network Server: This process is responsible for processing data messages and command messages of LoRaWAN devices. It manages the device sessions in the LoRaWAN network and generates corresponding MAC commands and uplink data messages according to the device's MAC commands and the application's downlink data messages.

b) Identity Server: This process is responsible for managing users and organizations in The Things Network Stack. It provides functions such as user authentication, access control, and rights management. It also provides standard authentication and authorization mechanisms such as OAuth2 and OpenID Connect.

c) Application Server: This process is responsible for processing application data messages. It receives the uplink data messages passed from the Network Server process, then processes these messages according to the business logic of the application and generates corresponding downlink data messages to send to the Network Server process. The Application Server handles the LoRaWAN application layer, including uplink data decryption and decoding, downlink queuing, and downlink data encoding and encryption.

d) MQTT Broker: This process provides an MQTT broker to support communication between The Things Network Stack and external MQTT clients. It converts MQTT messages into data messages of The Things Network Stack and converts data messages of The Things Network Stack into MQTT messages and sends them to external MQTT clients. The MQTT protocol is used as a communication protocol between the device and the Application Server to ensure data security and reliability. By using the MQTT protocol, we can isolate the Redis database from devices and applications, thereby improving the security and stability of the system. At the same time, the MQTT protocol also provides a flexible topic subscription mechanism, enabling applications to subscribe

and process specific data streams, thereby achieving more efficient and flexible data processing and control.

e) Redis: It is mainly used as DB to save the state information of different processes in The Things Network Stack, so that state information can be shared between different processes. For instance, MQTT Broker can store connected client state information in Redis, then other processes can query and use this information. Redis is also used for saving Feeds data and images.

The timing of communication between components in the LoRaWAN Service is shown in Figure 3.13.

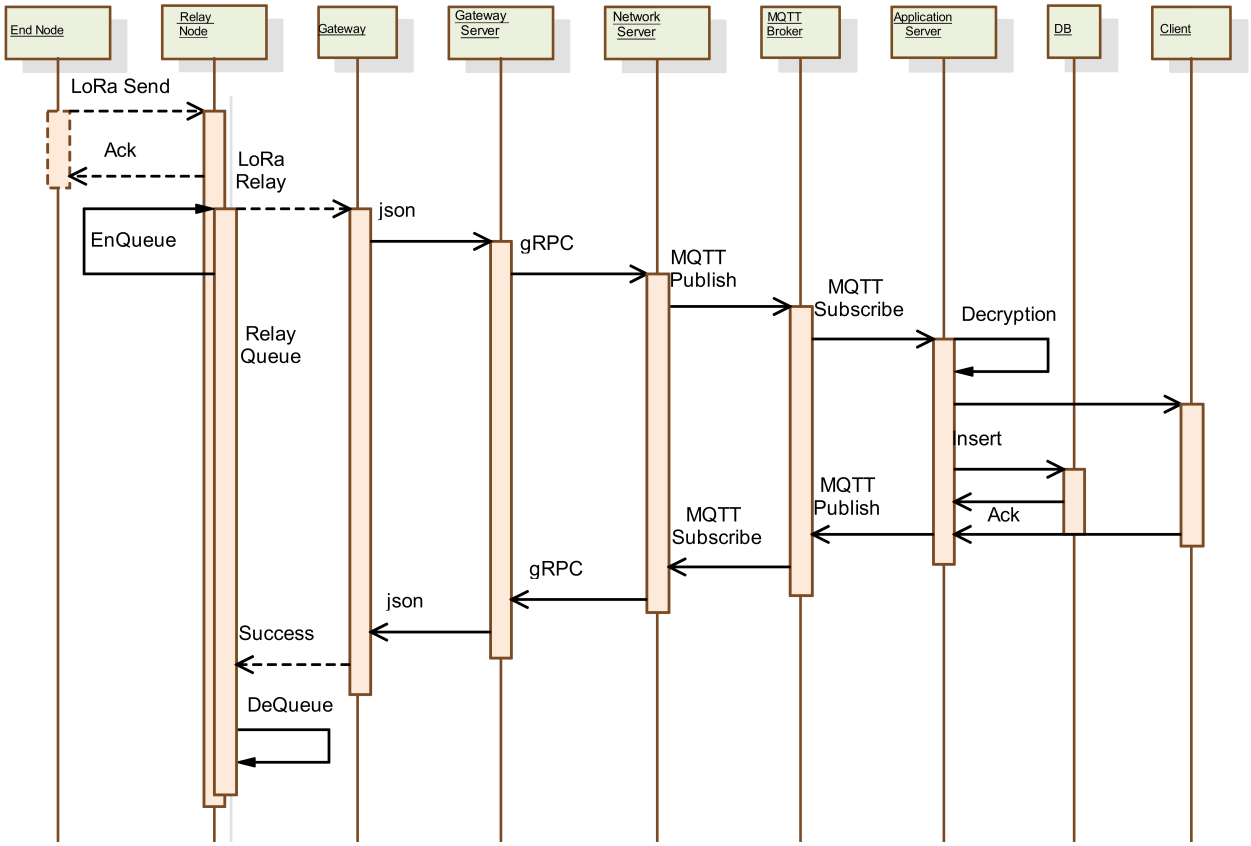


Figure 3.13. LoRaWAN Timing Diagram

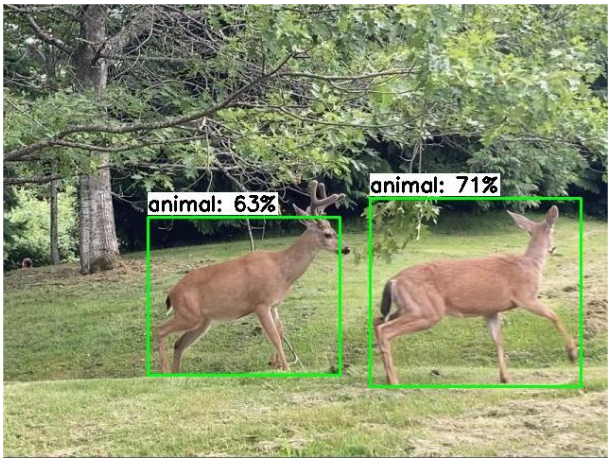
Chapter 4 Experimental Evaluation

Based on the above system design, the system was implemented, deployed, verified, and tested for function and performance.

4.1 Camera-embedded System Performance

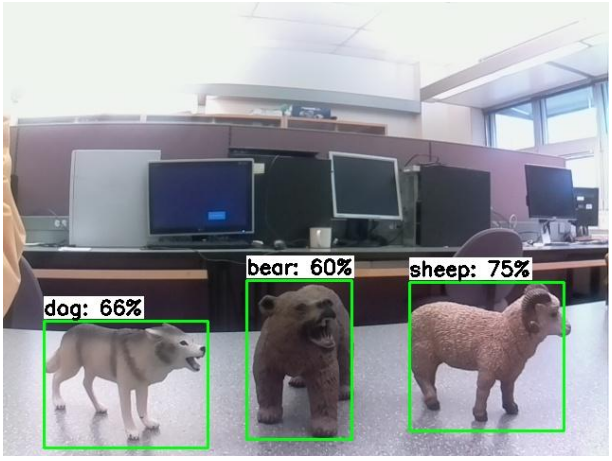
Animals are identified by their moveent using the camera-embedded system.

Since using real animals for testing is expensive, time-consuming, and impractical, low-cost scaled model figurines of animals and real animal photos are used to verify this AI-enhanced wildlife identification system.

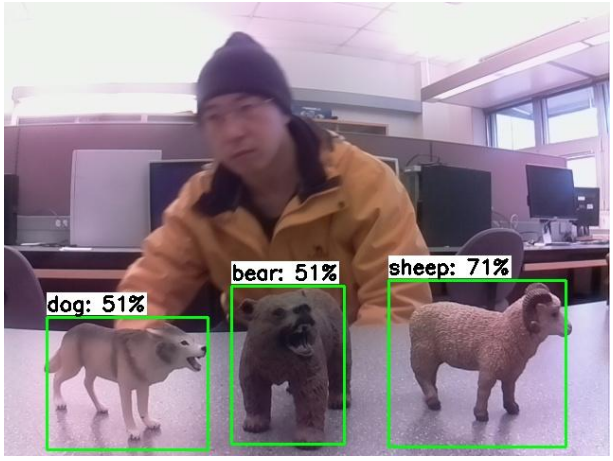


(a) Two identified animals in the campus

(b) One identified animal in the community



(c) Three identified animal figurines



(d) Animal identification is performed even with the presence of a human.

Figure 4.1. Animal Identification with Annotation

Figure 4.1 shows the results of the identification using low-cost scaled model figurines of animals. Identified animals will have bounding boxes and labels with predicting scores displayed on them in real time.

4.2 LoRa Device Deployment

(a) End node with antenna connecting with Camera-embedded system

(b) Camera-embedded system with alarm

(c) Gateway deployed on the roof

(d) Gateway with antennas

Figure 4.2. Deployment of Camera and LoRa Components

Figure 4.2 shows the actual deployment of LoRa devices. A LoRa end node is connected with the Camera-embedded System via UART, which is set in a Lab of the ELW building on the

UVic campus. The gateway is deployed in a student community housing roof one kilometer from the Lab.

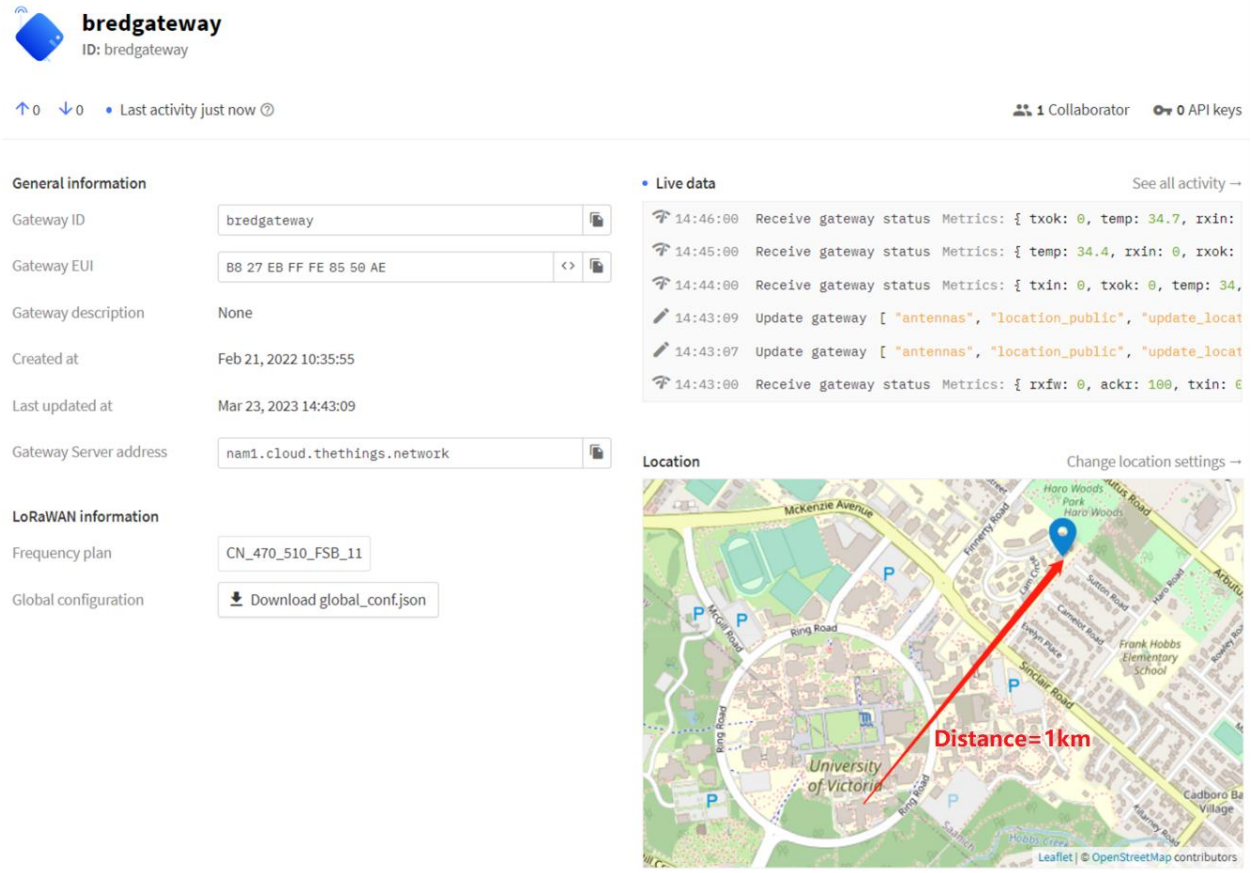


Figure 4.3. LoRa Gateway Location

The Things Network (TTN) [22] is an online service for LoRaWAN hardware device management and data processing. After registering the gateway on TTN, the TTN provide web pages for users to scan the information of the gateways and data of the LoRaWAN network protocol. Figure 4.3 shows the detailed information of the gateway, with gateway ID, EUI which is a unique number of the gateway, the latitude and longitude of the location in a map located by the GPS chip inside the Gateway, TTN server address, configure information, and the live LoRa frames received by the gateway. The gateway received the LoRa frames from nodes that are converted to JSON format and sent to the TTN server, corresponding to the Gateway Server address shown in Figure 4.3.

The parameters of the antenna used by the end node and the gateway are shown in Table 4.1 below:

Length	200mm
Weight	18g
Central Frequency	915MHz
Gain	3dBi
Interface	SMA-J
Voltage Standing Wave Ratio	<=1.5
Shell Material	Thermoplastic polyester elastomer (TPEE)
Impedance	50Ω
Power Capacity	20W
Operating Temperature	-40 to 85℃

Table4.1. Parameters of Antenna

4.3 LoRaWAN Service

The Things Stack [23] which is an open-source LoRaWAN network stack, is deployed with the web server we built on the cloud. The web console is used for scanning and controlling LoRa devices. Figure 4.4 shows the states of LoRaWAN online servers in this project. These servers are elaborated in Section 3.5.

Figure 4.4. LoRa Service Component Status

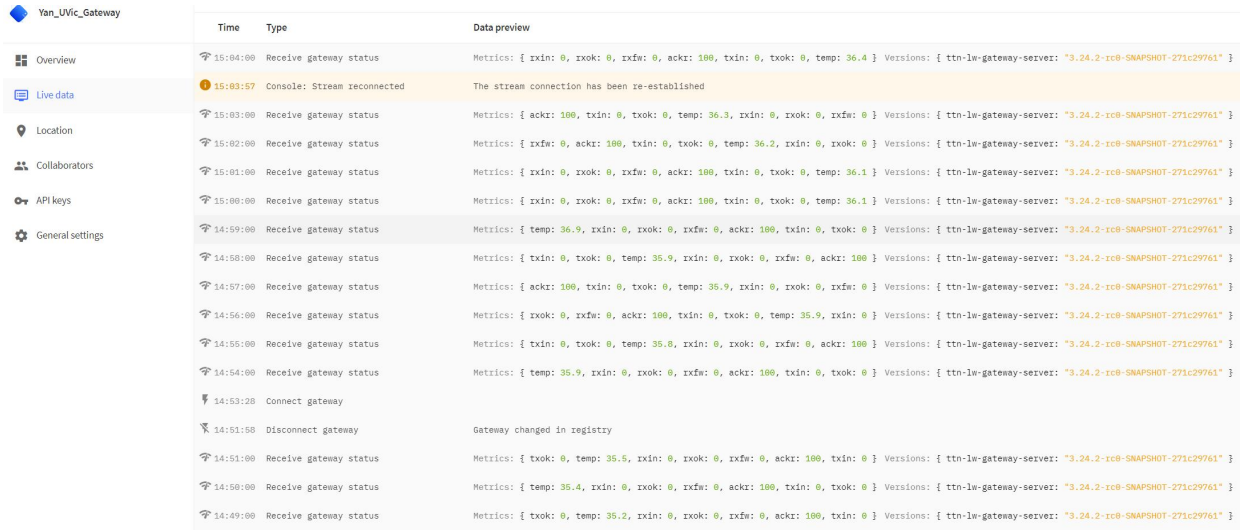


Figure 4.5. LoRa Gateway Data on Web Console

Figure 4.5 shows the LoRa frames from end nodes are received by a gateway and displayed on the user's web console.

134) "{\"stat\":{\"time\":\"2023-01-20 19:20:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":32.9}}"
135) "{\"stat\":{\"time\":\"2023-01-20 19:19:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
136) "{\"stat\":{\"time\":\"2023-01-20 19:18:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
137) "{\"stat\":{\"time\":\"2023-01-20 19:17:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
138) "{\"stat\":{\"time\":\"2023-01-20 19:16:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
139) "{\"stat\":{\"time\":\"2023-01-20 19:15:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
140) "{\"stat\":{\"time\":\"2023-01-20 19:14:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.0}}"
141) "{\"stat\":{\"time\":\"2023-01-20 19:13:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":32.9}}"
142) "{\"stat\":{\"time\":\"2023-01-20 19:12:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.0}}"
143) "{\"stat\":{\"time\":\"2023-01-20 19:11:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":32.9}}"
144) "{\"stat\":{\"time\":\"2023-01-20 19:10:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.0}}"
145) "{\"stat\":{\"time\":\"2023-01-20 19:09:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":32.9}}"
146) "{\"stat\":{\"time\":\"2023-01-20 19:08:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
147) "{\"stat\":{\"time\":\"2023-01-20 19:07:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
148) "{\"stat\":{\"time\":\"2023-01-20 19:06:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
149) "{\"stat\":{\"time\":\"2023-01-20 19:05:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
150) "{\"stat\":{\"time\":\"2023-01-20 19:04:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.0}}"
151) "{\"stat\":{\"time\":\"2023-01-20 19:03:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":32.0}}"
152) "{\"stat\":{\"time\":\"2023-01-20 19:02:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":32.0}}"
153) "{\"stat\":{\"time\":\"2023-01-20 19:01:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":32.9}}"
154) "{\"stat\":{\"time\":\"2023-01-20 19:00:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
155) "{\"stat\":{\"time\":\"2023-01-20 18:59:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
156) "{\"stat\":{\"time\":\"2023-01-20 18:58:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
157) "{\"stat\":{\"time\":\"2023-01-20 18:57:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
158) "{\"stat\":{\"time\":\"2023-01-20 18:56:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
159) "{\"stat\":{\"time\":\"2023-01-20 18:55:04 GMT\",\"rxnb\":8,\"rxok\":6,\"rxfw\":6,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
160) "{\"rxpk\":[{\"jver\":1,\"tmst\":875078383,\"chan\":2,\"rfch\":0,\"freq\":470.500000,\"mid\": 0,\"stat\":1,\"modu\":\"LORA\",\"datr\":\"
AAACMTglZDA4ODU3NWZfMzQlXzI2MF8xMDI5N18yMTJfNzFfMXowMIM=\",\"picid\":\"185d088575\"}]}"
161) "{\"rxpk\":[{\"jver\":1,\"tmst\":869843548,\"chan\":2,\"rfch\":0,\"freq\":470.500000,\"mid\": 0,\"stat\":1,\"modu\":\"LORA\",\"datr\":\"
AAACMTglZDA4ODQzMGZfMzQlXzI2MV8xMDI5N18yMTJfNzFfMeJLfW4=\",\"picid\":\"185d088430\"}]}"
162) "{\"rxpk\":[{\"jver\":1,\"tmst\":861828510,\"chan\":2,\"rfch\":0,\"freq\":470.500000,\"mid\": 0,\"stat\":1,\"modu\":\"LORA\",\"datr\":\"
AAACMTglZDA4ODFlOWJfMzQzXzI2MI8xMDI5N18yMTJfNzFfMYVJfC=\",\"picid\":\"185d0881e9\"}]}"
163) "{\"rxpk\":[{\"jver\":1,\"tmst\":857786974,\"chan\":2,\"rfch\":0,\"freq\":470.500000,\"mid\": 0,\"stat\":1,\"modu\":\"LORA\",\"datr\":\"
AAACMTglZDA4N2ZlMzJfMzQyXzI2NF8xMDI5N18yMTJfNzFfMU0COIc=\",\"picid\":\"185d087fe3\"}]}"
164) "{\"rxpk\":[{\"jver\":1,\"tmst\":853534124,\"chan\":2,\"rfch\":0,\"freq\":470.500000,\"mid\": 0,\"stat\":1,\"modu\":\"LORA\",\"datr\":\"
AAACMTglZDA4N2ZlMzJfMzQyXzI2NF8xMDI5N18yMTJfNzFfMU0COIc=\",\"picid\":\"185d087fe3\"}]}"
165) "{\"rxpk\":[{\"jver\":1,\"tmst\":849492803,\"chan\":2,\"rfch\":0,\"freq\":470.500000,\"mid\": 0,\"stat\":1,\"modu\":\"LORA\",\"datr\":\"
AAACMTglZDA4N2YzNGVfMzQyXzI2MI8xMDI5N18yMTJfNzFfMToIueI=\",\"picid\":\"185d087f34\"}]}"
166) "{\"stat\":{\"time\":\"2023-01-20 18:54:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
167) "{\"stat\":{\"time\":\"2023-01-20 18:53:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
168) "{\"stat\":{\"time\":\"2023-01-20 18:52:04 GMT\",\"rxnb\":0,\"rxok\":0,\"rxfw\":0,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"
169) "{\"stat\":{\"time\":\"2023-01-20 18:51:04 GMT\",\"rxnb\":8,\"rxok\":7,\"rxfw\":7,\"ackr\":100.0,\"dwnb\":0,\"txnb\":0,\"temp\":31.9}}"

Figure 4.6. LoRa Data saved in Redis

Figure 4.6 shows the LoRa frames are recorded in Redis storage in JSON format.

4.4 Client

4.4.1 LoRa Frames on Web

The user device can read all the LoRa frames saved in the LoRaWAN server as Figure 4.7 shows.

Each record of LoRa frames will be given a unique ID, displayed on a row on the page. The web server address is http://54.176.75.44/lorafeeds.

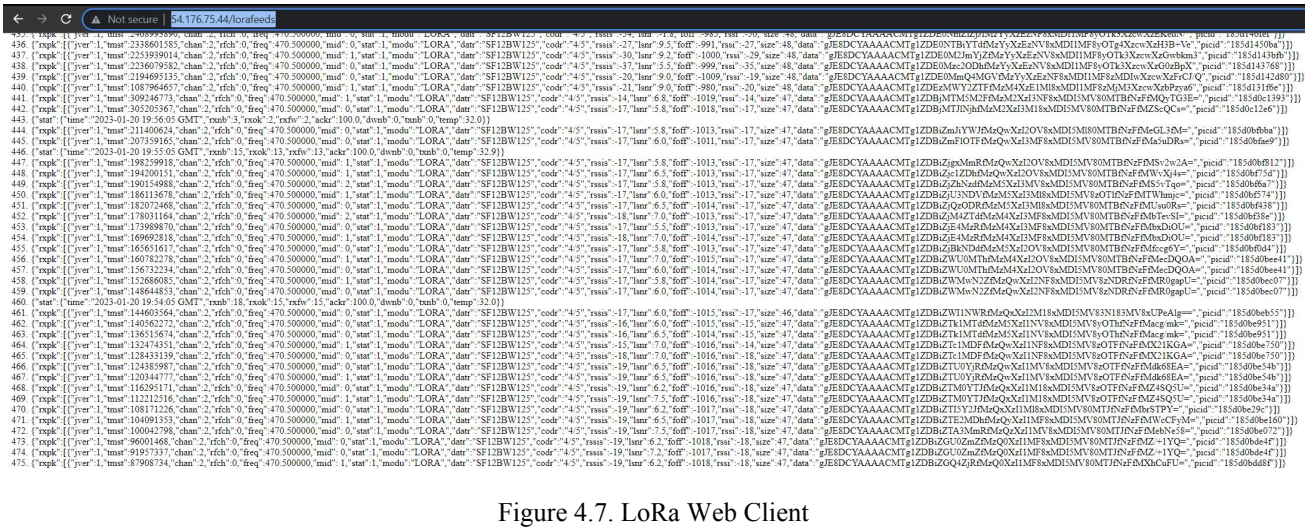


Figure 4.7. LoRa Web Client

4.4.2 Local Web UI through Wi-Fi Hotspot

When users use a terminal device (laptop, tablet, or drone) with a Wi-Fi HaLow module to connect, they can download images and environmental data through the web application as Figure 4.8 shows. There are two function buttons on the page. When the user presses the ‘Download & Clear’ button, the images and data will be packaged and downloaded to the user's device. After the download is complete, the Camera-embedded system will clear the downloaded data to free up space, which provides the hotspot based on Wi-Fi HaLow (IEEE 802.11ah) within a 1km range. When the user presses the ‘Real-time pic show’ button, the latest images will be played on the client device in real-time through Wi-Fi HaLow via web socket. Every image has a unique Pic ID, e.g., the name of the 3068th image is ‘1869ccaba9c’, with relevant environmental data.

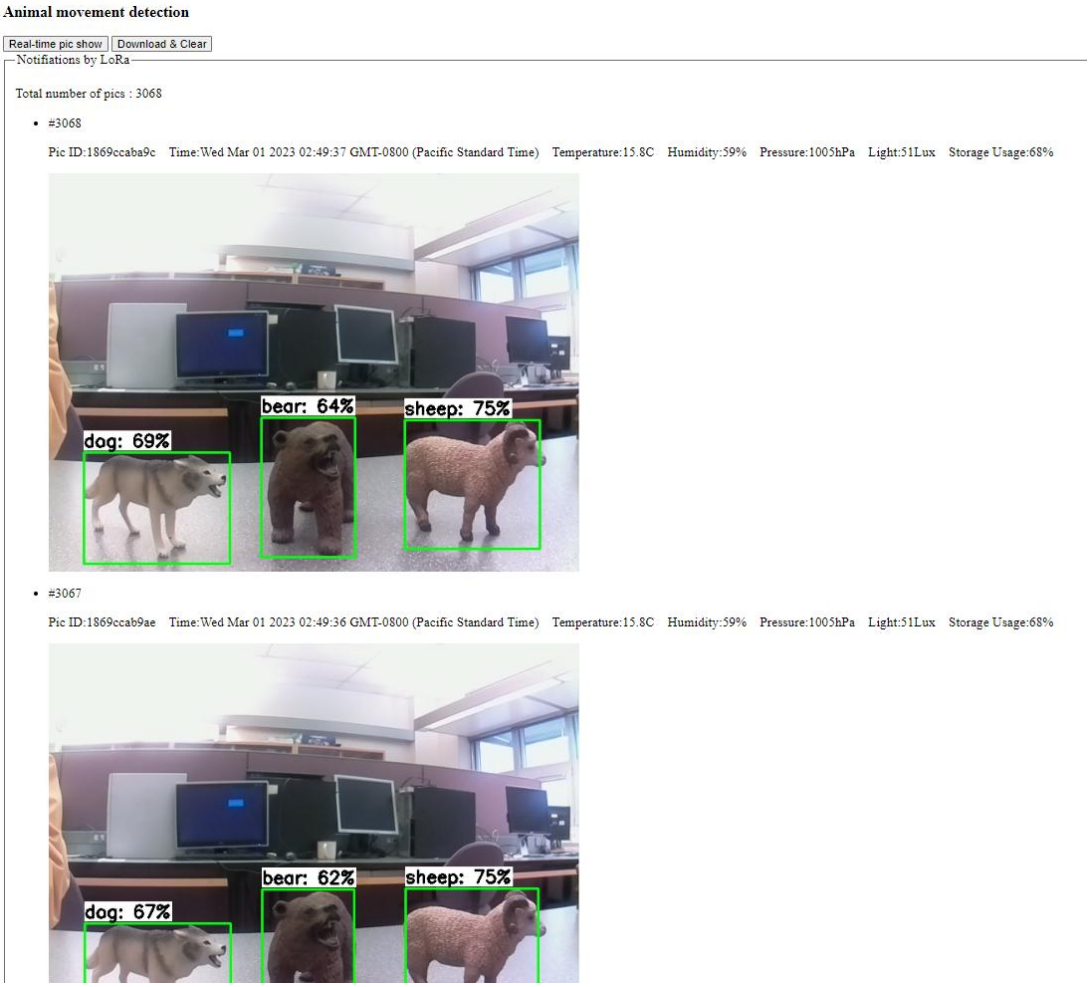
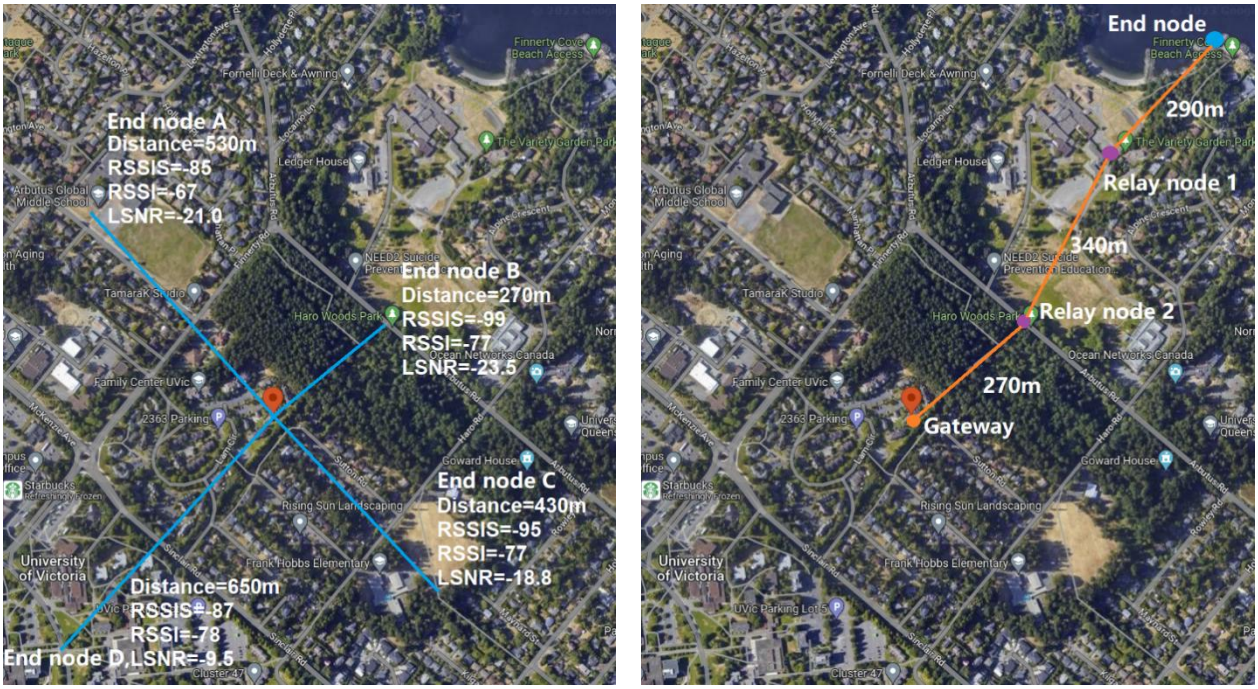


Figure 4.8. Web Client UI

4.5 Implementation



(a) End nodes use star topology network to send LoRa data to gateway

(b) One end node sends LoRa data through 2 relay nodes to gateway under a linear network.

Figure 4.9. Star Network and Relay Network

During the tests, end nodes first use star topology for message transmission to the gateway. The gateway is surrounded by a large number of obstacles such as tall forests and dense buildings. Taking the gateway as the center, LoRa sending tests are carried out in four directions respectively, using 3dB general antennas. The diameter covered by the gateway is about 1km as Figure 4.9 (a) shows. Taking the End node B as an example, the LoRa frame is shown as follows:


```
{"rxpk":[{
            "jver":1,
            "tmst":4050618957,
            "chan":2,
            "rfch":0,
            "freq":915.5,
            "mid":0,
            "stat":1,
            "modu":"LORA",
            "datr":"SF12BW125",
            "codr":"4/5",
            "rssis":-99,
            "lsnr":-23.5,
            "foff":-892,
            "rssi":-77,
            "size":27,
            "feed":" gJE8DCYAAAACMTg1YjI4NTIwMzQXXgG0gAz6",
        }]
}
```


The RSSI stands for "Received Signal Strength Indicator", the value is typically used to estimate the signal quality and the distance between the transmitting end node and the receiving

LoRa gateway. It is seen that there is a dense forest between end node B and the gateway. When the propagation distance is 270m long, RSSI is -77 dBm, indicating the signal has been attenuated to weak. LSNR stands for "LoRa Signal-to-Noise Ratio", the value is -23.5 dB, indicating a high probability of packet errors or loss and the communication is not good. The feed is a base64 encoded string, including the values of picid as "185b285203", humidity as 94%, temperature as 23℃, light strength as 55lux, and disk usage as 76%.

After the relay node is deployed in the middle of the transmission, using a linear network [16], which helps reduce relay collisions and extend the transmission range as Figure 4.9 (b) shows. The transmission can effectively avoid the influence of buildings and forests. The deployment location and number of relay nodes depend on the environment and the transmission target.

The system hardware equipment composition and cost details are shown in Table 4.2, and the total cost is $439.

Hardware Items	**Specification**	**Quantity**	**Price**
Camera	HBV-RPI-AUTO-IRCUT-3.6	1	$12
Raspberry Pi 3	ARM Cortex-A53 1.2 GHz CPU, 1GB LPDDR2 SDRAM	2	$70
Environment Sensors	BME280 & LTR-559	2	$16
Alarm	CMC-600-SG	1	$17
TF Card	Samsung 128G	2	$70
LoRa End Node	SX1268 + STM8L152	1	$22
LoRa Relay Node	SX1268 + STM8L152	2	$44
LoRa Gateway	SX1302 shield board	1	$110
Wi-Fi HaLow Module	AH-38	2	$58
Antenna		5	$20
Total			$439

Table 4.2. Equipment Cost

Chapter 5 Conclusions

This project uses LoRaWAN to build a completely independent network to meet long-range and low-power wilderness networking demands and provide original image synchronizing through Wi-Fi HaLow. A low-power and low-cost embedded system is implemented to realise animal motion detection and species identification. The principle of LoRa technology is introduced, and the advantages of LoRaWAN are compared with the existing LPWAN technologies used in the Internet of Things. Especially, LoRaWAN technology is proven to be appropriate in the vast Canadian wilderness environment to transmit animal identification results and environmental data information.

Advances in wireless sensor technology and long-range communication networks have made it possible to develop a system that can detect and monitor animal movements in real time over vast wilderness areas. Compared with traditional star network transmission, the deployment and implementation of relay nodes can avoid the impact caused by forests and buildings and effectively increase the transmission distance.

Through figurines and real photos of animals, the animal identification algorithm has been verified and performed well even in the presence of a human being. The system is able to significantly improve our understanding of animal populations and behavior in remote wilderness areas, which can be useful for a wide range of applications, including the safety of traffic, conservation efforts, wildlife management, and outdoor monitoring and alarm.

In this system, the Camera-embedded system costs $144, including a camera, a Raspberry Pi 3 with a TF card, a Wi-Fi HaLow module, two sensors, and one alarm. The end node with an antenna costs $26. The gateway is composed of a Raspberry Pi 3 with a TF card and an SX1302 shield with antennas, and the cost is $188. Two relay nodes with antennas cost $52. The user system has a Wi-Fi HaLow module that costs $29. The total system cost is $439.

In future work, the Raspberry Pi used in the current embedded system can be replaced by a lower cost and lower power STM32 system using RTOS. For instance, the STM32H7B3I-based board can support the complete workflow of image processing and TensorFlow Lite. An FPGA can be used to accelerate the motion detection processing of images to further improve the frame rate of embedded systems for image processing. There are possibilities for optimization and

improvement in LoRa's antenna and structural design to further promote the effective physical distance of transmission. Furthermore, the development of machine learning algorithms for animal identification can enable the extraction of meaningful insights from identified images and sensor data.

Reference

[1] Augustin, A., Yi, J., Clausen, T., & Townsley, W. (2016). A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. Sensors, 16(9), 1466. https://doi.org/10.3390/s16091466

[2] Reynders, B., Meert, W., & Pollin, S. (2017). Power and spreading factor control in low power wide area networks. In Proceedings of the 2017 IEEE International Conference on Communications (ICC) (pp. 1-6). Paris, France. https://doi.org/10.1109/ICC.2017.7996380.

[3] Banti, K., Karampelia, I., Dimakis, T., Boulogeorgos, A.-A., Kyriakidis, T., & Louta, M. (2022). LoRaWAN Communication Protocols: A Comprehensive Survey under an Energy Efficiency Perspective. Telecom, 3(2), 322-357. https://doi.org/10.3390/telecom3020018

[4] Davcev, D., Mitreski, K., Trajkovic, S., Nikolovski, V., & Koteli, N. (2018). IoT agriculture system based on LoRaWAN. In 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS) (pp. 1-4). Imperia, Italy. https://doi.org/10.1109/WFCS.2018.8402368

[5] Kais, M., Eddy, B., Frederic, C., & Fernand, M. (n.d.). A comparative study of LPWAN technologies for large-scale IoT deployment. https://doi.org/10.1016/j.icte.2017.12.005

[6] Lin, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (n.d.). Microsoft COCO: Common Objects in Context. In Computer Vision – ECCV 2014 (pp. 740–755). Springer International Publishing. https://doi.org/10.1007/978-3-319-10602-1_48

[7] Loriot, M., Aljer, A., & Shahrour, I. (2017). Analysis of the use of LoRaWAN technology in a large-scale smart city demonstrator. In 2017 Sensors Networks Smart and Emerging Technologies (SENSET) (pp. 1-4). Beiriut, Lebanon. https://doi.org/10.1109/SENSET.2017.8125011

[8] Georgiou, O., & Raza, U. (2017). Low Power Wide Area Network Analysis: Can LoRa Scale? IEEE Wireless Communications Letters, 6(2), 162-165. https://doi.org/10.1109/LWC.2016.2647247

[9] Vangelista, L., Zanella, A., & Zorzi, M. (2015). Long-Range IoT Technologies: The Dawn of LoRa™. In V. Atanasovski & A. Leon-Garcia (Eds.), Future Access Enablers for Ubiquitous and Intelligent Infrastructures. FABULOUS 2015. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 159. Springer. https://doi.org/10.1007/978-3-319-27072-2_7

[10] Semtech. (2019). SX1268 Long Range, Low Power, sub-GHz RF Transceiver. https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R000000Q2Kq/31nDblsvXMY94atnuZBurFOXI4KAa2uKsr26Vyh0snc

[11] Semtech. (2020). SX1302 LoRa Gateway Baseband Processor. https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R000000Hkyg/U8CIV3e9yI9T_aILFMxuzLNs_6_0Io1WIaksrNYyCMQ

[12] Devalal, S., & Karthikeyan, A. (2018). LoRa Technology - An Overview. In 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 284-290). Coimbatore, India. https://doi.org/10.1109/ICECA.2018.8474715

[13] Noreen, U., Bounceur, A., & Clavier, L. (2017). A study of LoRa low power and wide area network technology. In 2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP) (pp. 1-6). Fez, Morocco. https://doi.org/10.1109/ATSIP.2017.8075570

[14] OmniBSI Technology (2009). "OV5647 Datasheet Preliminary Specification". 1/4" color CMOS QSXGA image sensor. https://cdn.sparkfun.com/datasheets/Dev/RaspberryPi/ov5647_full.pdf

[15] Beltramelli, L., Mahmood, A., Österberg, P., & Gidlund, M. (2021). LoRa Beyond ALOHA: An Investigation of Alternative Random Access Protocols. IEEE Transactions on Industrial Informatics, 17(5), 3544-3554. https://doi.org/10.1109/TII.2020.2977046

[16] Abrardo, A., & Pozzebon, A. (2019). A Multi-Hop LoRa Linear Sensor Network for the Monitoring of Underground Environments: The Case of the Medieval Aqueducts in Siena, Italy. Sensors, 19(2), 402. https://doi.org/10.3390/s19020402

[17] BOSCH. (2022). BME280 Data Sheet. January 2022. https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf

[18] LITE-ON Technology Corp. (2014). LTR-559ALS-01 Data Sheet. September 2014. https://optoelectronics.liteon.com/upload/download/ds86-2013-0003/ltr-559als-01_ds_v1.pdf

[19] Raspberry Pi Ltd. (n.d.). Raspberry Pi hardware- GPIO and the 40-pin Header. Retrieved from https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[20] Google. (2009). Golang. Retrieved from https://go.dev/doc/

[21] Hauser, V., & Hégr, T. (2017). Proposal of Adaptive Data Rate Algorithm for LoRaWAN-Based Infrastructure. In 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud) (pp. 85-90). Prague, Czech Republic. https://doi.org/10.1109/FiCloud.2017.47

[22] TTN. (n.d.). Retrieved from https://www.thethingsnetwork.org/

[23] TTN. (n.d.). The Things Stack. Retrieved from https://www.thethingsnetwork.org/docs/quick-start/, https://github.com/TheThingsNetwork/lorawan-stack