UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Combining MLOps and low-power networking in edge computing applications

Diploma Thesis

Georgios Kapetanos

Supervisor: Christos Antonopoulos

May 2023

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Combining MLOps and low-power networking in edge computing applications

Diploma Thesis

Georgios Kapetanos

Supervisor: Christos Antonopoulos

May 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Συνδυασμός μηχανικής μάθησης και δικτύωσης χαμηλής κατανάλωσης ισχύος σε εφαρμογές στο άκρο του δικτύου

Διπλωματική Εργασία

Γεώργιος Καπετάνος

Επιβλέπων: Αντωνόπουλος Χρήστος

Μάιος 2023

Approved by the Examination Committee:

Supervisor **Christos Antonopoulos**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Spyros Lalis**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Nikolaos Bellas**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

I would like to express my gratitude to my supervisor, Professor Christos D. Antonopoulos, for his invaluable guidance and support throughout the entire duration of my thesis. His expertise and dedication have been instrumental in shaping the outcome of this research. I would also like to thank Professors Spyros Lalis and Nikolaos Bellas for their participation in the examination committee. Lastly, I extend my gratitude to the dedicated team at Oliveex, who during my internship there, provided me with invaluable knowledge that proved crucial for the accomplishment of this project.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Georgios Kapetanos

Diploma Thesis

Combining MLOps and low-power networking in edge computing applications

Georgios Kapetanos

Abstract

The objective of this Thesis is the design, development and evaluation of an Internet of Things (IoT) sensor system, based on the edge computing paradigm and utilizing a low power network for message delivery. Cloud platforms have conventionally been responsible for the processing of the data acquired from sensor devices owing to the computational power required for such operations. During the past years, technological advances in the semiconductor industry have allowed for IoT devices to feature significant processing power, therefore provoking a shift from the traditional cloud-based computing to an edge computing architecture, where the majority of processing happens near the source of the data.

Such a paradigm shift can bring considerable advantages in the IoT sector, ranging from more efficient network resource utilization by minimizing the data needed to be transferred to the cloud, to providing more security on the sensor data and improving the scalability of future sensor network expansions. The reduction in the volume of data transmitted to the cloud facilitates the use of low power networking, thus providing substantial area coverage and improved power efficiency at the expense of lower data throughput.

We design and implement a network system utilizing the LoRa physical communication technique, along with sample deployment configurations. The system has been evaluated for its range coverage capabilities in both an urban and rural setting, proving a range of 6 kilometers in the latter. Subsequently, we evaluated the processing capabilities of the IoT device used, by deploying an object detection configuration and measuring the power efficiency of the system.

Διπλωματική Εργασία

Συνδυασμός μηχανικής μάθησης και δικτύωσης χαμηλής κατανάλωσης ισχύος σε εφαρμογές στο άκρο του δικτύου

Γεώργιος Καπετάνος

Περίληψη

Ο σκοπός αυτής της εργασίας είναι η σχεδίαση, ανάπτυξη και πειραματική εκτίμηση ενός συστήματος αισθητήρων του Διαδικτύου των Πραγμάτων (IoT). Εφαρμόζουμε την αρχιτεκτονική υπολογισμού στο άκρο του δικτύου ενώ παράλληλα χρησιμοποιούμε ένα δίκτυο χαμηλής κατανάλωσης ισχύος για μετάδοση των μηνυμάτων. Παραδοσιακά, το υπολογιστικό νέφος ήταν υπεύθυνο για την επεξεργασία των συνολικών δεδομένων που παράγονται από τις συσκευές-αισθητήρες, λόγω της απαιτούμενης υπολογιστικής ισχύος τέτοιων εργασιών. Κατά τα τελευταία χρόνια, τεχνολογική πρόοδος έχει επιτρέψει την εδραίωση συσκευών IoT σημαντικής επεξεργαστικής ισχύος. Αυτό προκαλεί μία μετάβαση από τα υπολογιστικά νέφη σε αρχιτεκτονικές εστιασμένες στο άκρο του δικτύου, οι οποίες στοχεύουν στην επεξεργασία της πλειοψηφίας των δεδομένων κοντά στην πηγή τους.

Μία τέτοια μετάβαση μπορεί να επιφέρει ουσιώδη πλεονεκτήματα στον τομέα του IoT, τα οποία εκτείνονται από πιο αποδοτική χρήση πόρων των δικτύων λόγω ελαχιστοποίησης των δεδομένων που μεταφέρονται στο νέφος, μέχρι την αύξηση ασφάλειας στα δεδομένα και τη διευκόλυνση μελλοντικών επεκτάσεων των δικτύων. Η μείωση στον όγκο των δεδομένων που μεταδίδονται επιτρέπει τη χρήση δικτύων χαμηλής κατανάλωσης ενέργειας, αύξηση στην κάλυψη του δικτύου και βελτιωμένη ενεργειακή απόδοση, αφού πλέον είναι ανεκτό το κόστος της μείωσης του ρυθμού μεταφοράς δεδομένων του δικτύου.

Σχεδιάστηκε και αναπτύχθηκε σύστημα βασιζόμενο στην ασύρματη μέθοδο επικοινωνίας LoRa, το οποίο διαμορφώθηκε σε συνδυασμό με ενδεικτική εφαρμογή αναγνώρισης αντικειμένων. Το σύστημα αξιολογήθηκε ως προς την εμβέλειά του σε αστικό και επαρχιακό περιβάλλον, επιτυγχάνοντας εμβέλεια 6 χιλιομέτρων στο δεύτερο. Στη συνέχεια, δοκιμάστηκαν οι ικανότητες επεξεργασίας δεδομένων της χρησιμοποιούμενης συσκευής, εφαρμόζοντας έναν αλγόριθμο αναγνώρισης αντικειμένων και αποτιμώντας την ενεργειακή απόδοση του συστήματος.

Table of contents

List of figures

List of tables

Abbreviations

AES	Advanced Encryption Standard
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CRC	Cyclic Redundancy Check
CSI	Camera Serial Interface
CSS	Chirp Spread Spectrum
dBm	decibel-milliwatts
DC	Direct Current
DNS	Domain Name System
DTB	Device Tree Binary
DTO	Device Tree Overlay
EVP	EnVeloPe
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input Output
GSM	Global System for Mobile communication
IoT	Internet of Things
ISA	Instruction Set Architecture
ISM	Industrial, Scientific and Medical
IV	Initialization Vector
JSON	JavaScript Object Notation
LPDDR	Low-Power Double Data Rate
LPWAN	Low Power Wide Area Network
LTE	Long Term Evolution
LTE-FDD	Long Term Evolution - Frequency Division Duplex
MSL	Mean Sea Level

MQTT	Message Queuing Telemetry Transport
NB-IoT	Narrow band Internet of things
NMEA	National Marine Electronics Association
QS	Quality of Service
SBC	Single Board Computer
SF	Spreading Factor
SMA	SubMiniature version A
SNR	Signal to Noise Ratio
SoC	System On Chip
SSL	Secure Sockets Layer
SPI	Serial Peripheral Interface
TLS	Trasnport Layer Security
TCP/IP	Transmission Control Protocol/Internet Protocol
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
USB OTG	Universal Serial Bus On-The-Go
UUID	Universally Unique Identifier
WAN	Wide Area Network
WCDMA	Wideband Code Division Multiple Access

Chapter 1

Introduction

The constant technological development in integrated circuits during the last decades resulted in the wide availability of low power System On Chip (SoC) devices, capable of providing high computational power. Such devices can be found on modern Single Board Computers (SBCs), usually featured on Internet of Things (IoT) applications. The constantly increasing computational power has enabled the shift of the traditional paradigm of performing complex computational tasks on servers, to on-site devices closer to the source of the data, a so-called Edge Computing architecture. Edge computing brings various advantages to a computing system, by reducing the IoT device to server bandwidth utilization, improving reliability with a distributed device network and enhancing security by limiting the transmission of sensitive data from edge devices.

However, deployment of IoT devices requires a network link from the source of the data to the cloud server for final processing, and for the distribution of the data to its clients. While the server-to-client and part of edge device-to-server paths can utilize existing Internet infrastructure, sensor devices near the source of the data may be located far from existing infrastructure, in places with limited or no network, or power availability.

Low-power wide-area networking (LPWAN) provides a solution to the previously described challenges through the use of optimized wireless protocols, capable of covering wide areas, often providing kilometers of range with minimum cost due to the absence of wired communications and low power consumption by implementing efficient modulation techniques, adaptive data rate and low bandwidth compared to other technologies. Nonetheless, the disadvantage of decreased bandwidth is not of high significance as IoT applications often require the transmission of a few sensor readings during relatively large time intervals.

The combination of edge computing with low-power wide-area networking appears to further alleviate the issue of low data rate transmissions as computations near the sensor devices provide the opportunity for reducing the total data payload needed to be transferred to the cloud for final processing, or in the case of full processing in the edge, only the generated result of the computation would be transferred instead of the sensor output.

Various competing LPWAN technologies have emerged, each one optimized for a different subset of use cases concerning range, power consumption, bandwidth, deployment cost, and network topology. A private LPWAN may be established specifically for covering the needs of the operator, or access to the network may be offered as a third-party service so that sensor device operators avoid the cost of gateway infrastructure. Though less frequent, LPWANs may follow a mesh or point-to-point topology, eliminating the need for a gateway. Some of the most notable LPWAN technologies include:

- LoRa: A proprietary communication technology using the chirp spread spectrum method to encode data which allows for noise robustness and low power consumption, it offers a long-range, exceeding 10 kilometers with a data rate between 0.3 to 27 kbit/s while operating in sub-GHz bands, mainly 433 and 862 through 930 MHz depending on location. It features adaptive data rate, a technique that dynamically adjusts transmission parameters by monitoring channel Signal to Noise ratio (SNR), allowing for optimized network performance and power efficiency. Its absence of collision detection and avoidance methods introduces the impediment of requiring duty cycle limitations on transmission duration between devices operating on the same channel [2].

- NB-IoT (Narrowband Internet of things): A cellular-based technology that uses existing cellular network infrastructure to implement a LPWAN. It utilizes a subset of the long-term evolution (LTE) standard with a bandwidth of 200 KHz while operating on usual LTE frequencies from 800 to 2100 MHz. NB-IoT is suitable for higher data rate applications as it operates on licensed spectrums by third-party providers and is free of duty cycle restrictions.

- IEEE 802.11ah: It uses similar modulation as other 802.11 Wi-Fi standards but operates on sub-GHz frequencies similar to LoRa. Compared to LoRa it is more optimized for higher data rate applications with certain modulation and coding scheme configurations exceeding 100 Mbit/s when utilizing an 8 or 16 MHz channel. However, LoRa is more

suitable for long-range communications and low power devices [3].

1.1 Thesis Objective

This Thesis will study the combination of carrying out a computationally complex task on a low power single board computer, appropriate for IoT applications, together with low power-wide area networking to make the software system suitable for remote location deploying, where other network connectivity is either not available or strict data rate limitations apply. This system implements an edge computing architecture where the data is processed close to its source, the sensor device, allowing for a drastic reduction in data which need to be transmitted to a cloud-based server and/or subsequently relayed to client devices.

Machine learning operations have been chosen as a computationally complex task. More specifically we experiment with object detection, namely the differentiation of objects from their background. A video feed is provided from a sensor attached to an edge single-board computer and instead of streaming the feed to a remote cloud-based server, an operation requiring high data transfer volume and rate, it is processed locally using a pre-trained model for computer vision. Subsequently, a message containing the detected objects is broadcasted to the server. Since the local device may have limitations concerning processing power and available energy budget, certain compromises regarding detection accuracy may be evaluated, such as the video feed resolution, in order to optimize the performance of the software system.

Overall, the resulting edge computing, LPWAN-based system is expected to have the following advantages compared to a traditional cloud based one:

- Wide area coverage: The mentioned LPWAN technologies may provide more than 10 kilometers of range. Use of common wireless media would limit the spanning of devices to less than 1 km.

- Efficient use of network resources: The transmission of the video feed is not needed, resulting in bandwidth savings in the edge to cloud path. Avoidance of high data rate wireless protocols and use of LPWANs also permits considerate energy savings on the edge devices which may be battery or solar-powered [4].

- Latency: The locally calculated result may be used immediately locally or near the

network edge, instead of waiting for a remote server response.

- Security concerns: An edge-to-cloud network path may consist of several nodes from which the possibly sensitive video feed will be transmitted, keeping the original data on the local devices may result in increased privacy.

- Scalability and application heterogeneity: Each edge device may be customized for its specific use case while the cloud would provide a general interface compatible with the rest of the system. Therefore, new functionality can be introduced into the system just by the addition of appropriate edge nodes. More importantly, the expansion of the sensor network leaves a less significant impact on the bandwidth to the cloud. In a traditional cloud processing architecture, a large number of sensor devices constantly outputting an example of 1 Mbit/s feed would saturate the server's downlink, when on an edge computing architecture each device could provide orders of magnitude less data [5].

1.2 Significance and Contributions

The wireless network system developed is optimized to serve the edge computing architecture that was previously described. It is designed to transmit data from IoT devices in remote areas with no other existing infrastructure, assessed to reliably convey packets at six kilometers when on a relatively unobstructed line of sight with the receiver. Furthermore, it features end-to-end encryption, packet integrity validation, and node identification on the network.

On further evaluation, in addition to the advantage of remote network coverage, our system was estimated to have 92% less power consumption on the wireless network subsystem, compared to alternative cellular-based solutions.

Apart from the wireless network subsystem, we integrated Message Queuing Telemetry Transport (MQTT) server infrastructure into the system and we developed a number of test cases to demonstrate the edge computing architecture, LoRa capabilities, and possible use scenarios.

1.3 Organization of the Thesis

The general design and implementation of the system are described in Section 2, including the implementation of the network layer developed on top of the LoRa physical layer, publishing of messages to the MQTT server, a sample edge computing use case that combines object detection with the long-range network and the deployment of mobile network nodes for demonstrating the capabilities of the wireless subsystem.

Section 3 initially presents the results of two experiments concerning the network range and coverage. We evaluate the performance, power consumption, and security of the system, assuming an object detection application running at the edge of the network. Finally, in order to highlight its advantages, some aspects of this system are compared to a traditional server-based one, using cellular wireless infrastructure for connecting the sensor devices.

Section 4 discusses the conclusions from the development and evaluation of the system developed for the purpose of this Thesis.

Chapter 2

Design

2.1 General Design

The general design of the implemented system is displayed in Figure 2.1. The central functionality is the transmitter and receiver software. They constitute the network layer built on top of the LoRa physical layer, responsible for the formation of network packets that carry an application payload. On the transmitter side, the sensor input is provided by a camera module forwarding an image feed to the object detection model, which in turn outputs a sequence of characters (string) describing the detected semantic objects. The generated string is then processed in a library and passed on to the network library. Due to the network library being implemented in the C language and the object detection model in Python, a Python/C bind Application Programming Interface (API) exists between them to enable the calls of C functions by Python code.

The network library provides a Universal Asynchronous Receiver/Transmitter (UART) interface for connecting the LoRa transceiver modules and is responsible for message checksum calculation and validation, node identification, encryption and formation of the final packet.

On the receiver side, the received message is decrypted, validated and forwarded to a Message Queuing Telemetry Transport (MQTT) server using the TCP/IP protocol, from where the messages can be distributed to the clients.

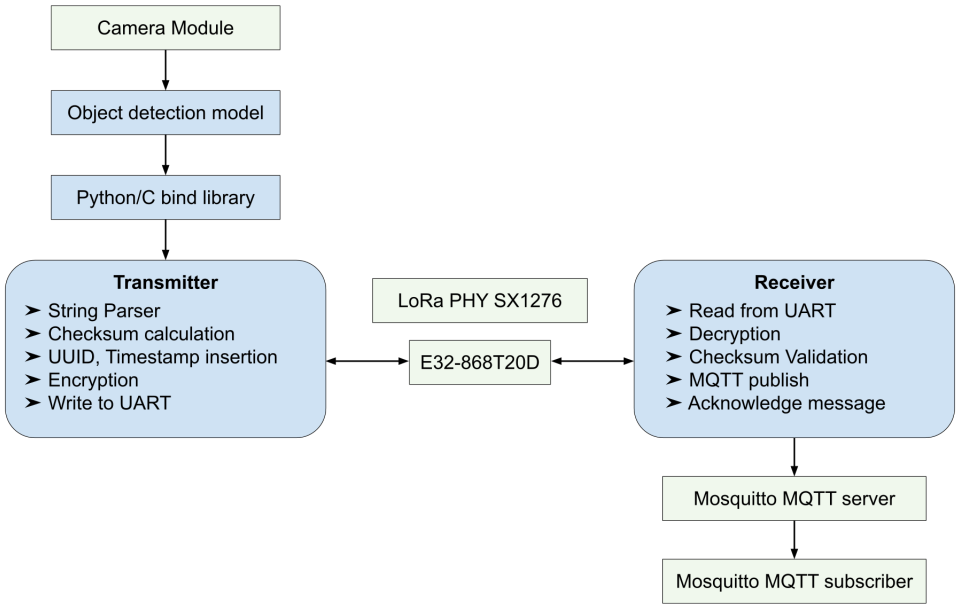

Figure 2.1: General design of the implemented system

2.2 LoRa Physical Layer

For the implementation of the network connectivity between the sensor device and the base station, whose purpose is to forward messages to a cloud server, LoRa was chosen as a lower physical layer communication technique, due to its long range and ability to function without any infrastructure, either self-deployed or third-party.

The Low Power Wide Area Network (LPWAN) we implemented does not include any gateway-oriented hardware. It is similar to a peer-to-peer wireless connection in the sense that the same hardware that functions as a transmitter from a sensor device is used on the concentrator, however, is has been designed to allow multiple transmitters, limited only by the congestion of the frequency spectrum used. More than one receivers can be configured to run independently in order to take advantage of multiple LoRa channels existing on different frequencies, so as to avoid interference due to network congestion.

The LoRa transceiver modules used for this system are the E32-868T30D and E32-868T20D, which have a maximum transmission power of 1 and 0.1 watt respectively. They are based on the Semtech SX1276 RF chip, while featuring an internal microcontroller for communication between the chip's Serial Peripheral Interface (SPI) and the (UART) protocol

of the transceiver module. The supported data rates are between 0.3 and 19.2 kbps depending on the environment conditions and the distance. The transmission range is reported by the datasheet at a theoretical maximum of 8 kilometers [6] [7].

As LoRa does not feature any collision detection and avoidance algorithms on its physical layer, devices should take into account the regulations concerning the frequency bands used for their communication. In Europe, LoRa uses the ISM 868 MHz band, consisting of a number of sub-bands with different restrictions. Table 2.1 lists the restrictions regarding duty cycle, the maximum percentage of time during which a device could transmit data [8].

Table 2.1: EU868 band regulations

Frequency (MHz)	Duty Cycle (%)
863-865	0.1
865-868	1
868-868.6	1
868.7-869.2	0.1
869.4-869.65	10
869.7-870	100
869.7-870	1

2.3 Network Layer Implementation

The wireless system implemented on top of LoRa is responsible for transferring JavaScript Object Notation (JSON) formatted text containing a payload, among other protocol-related information essential for recipient designation, sender identification, and data integrity.

JSON is a data exchange format consisting of key-value pairs, JSON objects recursively containing other key-value pairs, and arrays with either objects or values. It was chosen due to its low size overhead, its structure and its easy implementation in C and Python programming languages.

The specific structure of the packet consists of the following fields:

- A key describing the type of the JSON packet; for instance, value ’3’ corresponds to a usual plain string payload.

- A recursively embedded JSON object containing the string payload.
- A timestamp with hour:minute:second format capturing the formation time of the JSON packet.
- An universally unique identifier in the form 8-4-4-4-12 for a total of 32 hexadecimal characters (plus 4 dashes in the string representation) is the identifier of the specific node in the network.
- A two-byte checksum located outside the root JSON object, that is stripped once the data integrity is validated.

LoRa is subject to maximum packet length limitations which is determined by the spreading factor (SF) used for the transmission. While higher SF values enable increased data rates, they lower effective communication range and further limit the maximum packet length. Using 7 to 12 SFs, the maximum packet size is varied from 222 to 51 bytes [9]. However, as the payloads of those packets are concatenated by the microcontroller of the LoRa module, the network layer implementation is not affected by this limitation. The Lora module is also limited by a 256-bytes internal buffer which can also be bypassed by using appropriate low-level code interfacing the serial port. Overall, the network layer maximum packet size is limited to 4096 bytes.

2.3.1 Receiver Packet Integrity Validation

An optional cyclic redundancy check (CRC) is implemented in the LoRa physical layer for error detection [10]. However, an additional 2-byte checksum code is generated at the network layer, mainly used for confirming the successful decryption of a received packet.

Upon the formation of the final JSON string containing the payload and the other protocol fields, a checksum code is generated with the aid of a XOR bitwise operation between all the characters of the JSON structure. The resulting output is an 1-byte binary code, which is represented as a 2 digit character hexadecimal value and then inserted to the end of the JSON string, outside the JSON object.

The receiver calculates the same XOR bitwise operation across the packet except for the last 2 characters which are the transmitter side checksum code, and subsequently compares those two in order to validate data integrity. After the comparison, an appropriate operation is executed according to the acknowledgment configuration of the node.

2.3.2 Packet Acknowledgement

When a packet is received by a node, an appropriate acknowledgment is transmitted in order to complete a two-way communication, otherwise, the sender would constantly transmit packages without feedback whether they reached their destination.

The receiver checks the integrity of the received packet as described in Section 2.3.1 and transmits an acknowledgment packet containing a JSON object with 3 fields. The first key-value pair describes the nature of this packet being an acknowledgment one, the second contains the checksum of the original packet, and the third indicates whether this checksum was found to match the calculated checksum from the receiver side.

Once the receiver of the original packet obtains the acknowledgment, they can decide on a second attempt to transmit the message, based on their configuration.

In the event that no acknowledgment is received, the recipient may have failed to acquire the packet so the sender would try for retransmission up to a predefined number of attempts.

2.3.3 Packet Encryption

This software system features end-to-end encryption on wireless transmissions by utilizing the OpenSSL library. High-level access to OpenSSL cryptographic functions is provided by the OpenSSL EVP (EnVeloPe) [11].

The encryption uses the Advanced Encryption Standard specification combined with a 256-byte key and a 128-byte initialization Vector (IV). The block cipher mode of operation (the algorithm used for encryption) is configurable by the user and the following modes are supported by the library:

- CTR (Counter mode)(default recommended): The initial data are divided into blocks, and a counter value is incremented for each block of input data which is encrypted and used to generate a key stream that is XORed with the data before encryption.

- CBC (Cipher Block Chaining): The initial data are divided into blocks that are then XORed with the IV before being encrypted using the key.

- CFB (Cipher Feedback): The previous encrypted block of data is XORed with the current block of input data in order to produce the new encrypted block.

- OFB (Output Feedback): A key stream is generated by encrypting the initialization vector and XORed with the input data in order to produce the ciphertext thus using the same key stream for every block.

The key and IV are passed to the node during its software setup and stored in a configuration file. Both are required for encrypted communication between two nodes. The user can select the cipher mode of operation with an appropriate string on the same file.

Encryption is the last stage of packet formation on the transmitter when propagation to the LoRa module through the UART interface immediately follows. While the initial data are exclusively composed of ASCII (American Standard Code for Information Interchange) characters, as the payload is character-based, encryption outputs binary data, requiring the encrypted packet to be treated as a binary one for the rest of the transmission path. This affects the host to LoRa module UART communication, thus necessitating the configuration of the serial port for raw data handling. This issue will be discussed further in Section 2.3.4 of this Thesis.

2.3.4 UART Serial Communication Implementation

The selected LoRa modules for this system provide a UART interface for connecting to the host. UART is a serial communication protocol used for communicating between two wired devices. It consists of a transmitter (Tx) and a receiver (Rx) terminal, therefore providing full duplex communication. Consideration should be given to the fact that LoRa is a half-duplex protocol, thus transmitting and receiving on the same channel, as a result, the full duplex advantages of UART can not be taken advantage of.

UART connectivity is widely available on single-board and embedded computing devices. Other devices can join the network with either an external USB UART adapter or an RS-232 port. However, an RS-232 port should be coupled with an external circuit for electric signal handling translation as it is a higher voltage protocol compared to other standards on embedded devices that operate on either 3.3 or 5 volts. Overall, the UART interface selection enables supporting most existing computer devices.

During initialization of the serial interface, the following settings are configured in order to enable a binary data communication, discarding any interpretations of ASCII characters [12]:

- PARENB: Disable parity bit in output and input

- CSTOPB: Use only one stop bit per character
- CS8 CSIZE: 8 bit per byte character size
- CRTSCTS: Disable RTS (Request to Send) and CTS (Clear to Send) hardware flow control
- ECHO: Disable echo of input characters. This feature is used on remote terminals where the typed character is both displayed on the local terminal and transmitted to the remote host. However, it results in retransmissions of whatever the hardware is receiving, wasting LoRa bandwidth.
- ECHOE: Disable erasure of preceding input character
- ECHONL: Disable echo of the newline character
- ISIG: Disable interpretation of INTR, QUIT, SUSP, and DSUSP characters
- IXOFF: Disable software flow control on input
- IXON: Disable software flow control on output
- IGNBRK, BRKINT, PARMRK, ISTRIP, INLCR, IGNCR, ICRNL: Disable interpretation of various bytes
- OPOST: Disable implementation-defined output processing
- ONLCR: Disable interpretation of newline character to carriage return
- VTIME: VTIME is a timeout interval determining the return of read calls on the serial port, it is set to 1 second.
- VMIN: VMIN is the minimum bytes required for the return of a read call, set to 0.

Serial baud rate is set to 9600 baud/second (bps), the default rate of the used LoRa modules, but it can be set between 1200-115200 bps after configuring the module appropriately.

For data transmission, a simple write call to the serial port is needed. For the reception, a wrapper receive function is implemented using consecutive read calls (after each one-second read timeout defined on the serial port configuration, thus avoiding wasting processor cycles) until a whole packet is received with a maximum size of 4096 bytes.

2.4 MQTT Messaging

MQTT (Message Queuing Telemetry Transport) is a messaging protocol designed for connecting remote devices with limited resources and Internet of Things (IoT) communications [13].

The MQTT protocol employs a publisher and subscriber architecture for sharing messages between devices. Transmitters publish a message to an MQTT server, referred to as a broker, under a specific topic, while receivers subscribe to a topic of their choice and receive messages forwarded from the broker. This scheme allows for some abstraction between the sensor devices and the frontend responsible for arranging the sensor data to a more user-friendly format as the devices are not required to know each other's addresses by having the broker handle the connection of each end [14].

We selected Eclipse Mosquitto [15] as the MQTT broker implementation for this Thesis, as it is an open-source implementation of the MQTT protocol, ideal for real-time communication between devices, providing a number of features such as Quality of Service (QoS) and SSL support for encrypted network connections and authentication.

The Mosquitto MQTT broker is deployed on a remote, off-site server, where the LoRa receiver, typically an Internet-connected node which receives the sensor data wirelessly, forwards its messages, which are then conveyed by the broker to the data recipients.

As previously described, the LoRa receiver obtains a packet, decrypts, and validates its integrity generating a plain JSON object. This JSON object is published to an off-site MQTT broker, whose location could be in the cloud, under a specific topic. This completes the transportation and delivery of the system data, allowing a front-end application to take over the final presentation of the data.

2.5 Network Node configuration

A configuration file based on the YAML syntax is provided to each node of the network in order to define and configure its functionality. The configurations available define the universally unique identifier (UUID) of the node, the serial port where the LoRa transceiver is connected, a number of MQTT messaging parameters, encryption modes, credentials, and general receiver/transmitter node functionality.

General node functionality options:

- uuid: 128-bit hexadecimal label represented by 36 characters.
- serial: Serial Port device.
- acknowledge_packets: Whether the node should acknowledge received messages.
- broadcast_to_mqtt: Whether the node should broadcast incoming messages to an MQTT server.
- enforce_uuid_whitelist: Only broadcast incoming messages containing a set of predefined UUIDs.

The 36-character uuid is generated randomly on the first run of the node but can also be updated manually. It is essential for identification between different nodes coexisting on the network, permitting the end user to recognize the node from where the current sensor readings are received. Along with the EVP encryption, it strengthens security as it allows for implementing a whitelist for not permitting the forwarding of packets from disallowed third-party devices.

The UUID whitelist is stored in a different YAML file which is parsed on the initialization of the receiver node. In addition to the security and identification purposes served as described previously, it provides a method to correspond transmitter devices to specific receivers used as gateways. When multiple receivers coexist in an area, an uplink device may be in the range of multiple of them causing double MQTT message broadcasts. However, by assigning each sensor to a specific gateway we can overcome this issue.

The MQTT parameters for publishing received messages are listed below. The use of the MQTT message forwarding will be discussed in Section 2.4:

- mqtt_hostname: Hostname or IP address of available MQTT server.
- mqtt_port: Port used by the server. For the Eclipse Mosquitto used on this implementation, 1883 and 8883 are usually used.
- mqtt_keep_alive: The time interval, in seconds, after which the MQTT server should send a PING message if no messages have been transmitted or received during the interval.
- mqtt_password & mqtt_username: Credentials for connecting to the server.

Finally, the OpenSSL EVP encryption parameters are:

- encryption_mode: Block cipher mode of operation used by the Advanced Encryption Standard (AES) algorithm. The modes ctr, cbc, cfb, ofb are available for use.
- encryption_key: 32-byte Encryption key.
- encryption_iv: 16-byte Initialization vector.

2.6 Python-C Bind interface

The software system described so far in this section is implemented in the C language, by making use of the mosquitto, uuid and crypto libraries while providing a library for utilizing this LoRa network. It can either be included in other C software utilities by calling these functions or executables can be called by scripting languages with sensor data in calling arguments.

However, most object detection libraries, required for the processing of the camera input for this project, make use of the Python language. Therefore, in order to avoid redundant string parsing and passing data as arguments with the use of scripting languages, a Python/C API can be provided that creates a C binding interface for a Python module. More specifically, a Python library is created that can be imported in a Python script with methods that translate to the C functions of the C network library.

The following methods are implemented and available to be used by Python scripts:

- *transmit*: Parse a string containing a sensor output, insert it as a payload on a JSON object, and transmit it over a LoRa interface.
- *transmit_encrypted*: Same as transmit but providing OpenSSL EVP encryption.
- *process_detections_str*: Process a string containing object detections and format it before transmission.
- *receive_encrypted*: Receive a LoRa message from the serial interface, decrypt it, and return the plaintext as a string.
- *generate_json_str*: Generate a JSON object with a string payload and return it as a string.

- *lora_str_translate*: Replace JSON object initials from LoRa message (truncated in order to reduce packer overhead) to full words. Used before publishing to an MQTT server where air time and bandwidth limitations are less severe.

2.7 Security Considerations

Security is essential for a wireless communication system, such as the implemented LoRa network, especially when used for IoT applications, as devices may often transmit messages containing sensitive data including location (as demonstrated in Section 3.1.2 where a device transmits its current coordinates for determining communication range) or other sensor readings. Therefore, transmitted data need to be secured using End-to-end encryption (E2EE) along their entire path, from the sensor device to the user frontend.

Figure 2.2, presents the different encryption methods employed. AES (128 or 256 bit) [16] was selected for the wireless transmission over LoRa. In combination with the CTR mode, which is considered as one of the most secure block modes of operation, it provides both confidentiality and data integrity. Other modes of operation are also supported, including the CBC, but may be vulnerable to certain attacks, such as the padding oracle attack [17]. After being received by the LoRa base station, the message is published to an MQTT server which can provide encryption using the SSL (or more specifically the newer Transport Layer Security, TLS) cryptographic protocol and can then be propagated using the same protocol to the subscribed users.

In order to implement the TLS encryption, the server is assigned a domain name by a Domain Name System (DNS) provider and is then provided with a certificate from a Certificate Authority (CA).

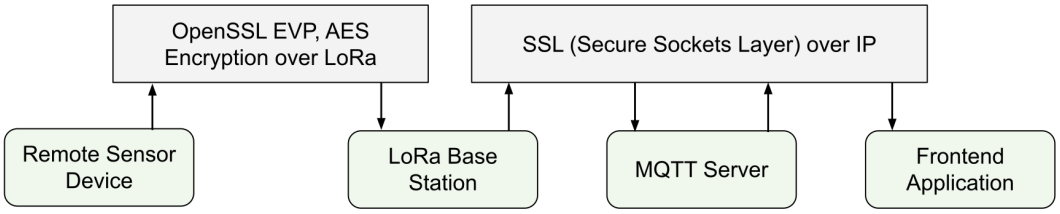

Figure 2.2: End-to-End encryption from the sensor device to the frontend.

The edge computing architecture, depending on its implementation, can be more secure than the traditional server-based processing architecture. On the latter, data acquired from sensors need to travel long distances through the network path where vulnerable nodes can expose sensitive information. On the contrary, in edge computing, the majority of the data information can remain at the edge of the network, transmitting only results after processing, thus hiding sensitive information contained in the initial sensor readings. Overall, the security of an edge computing system is determined by its implementation and use case. For instance, in the implemented object detection system, the acquired images remain in the sensor device and only the detection results are propagated to the network path. Subsequently, the privacy of the images is preserved as long as this single sensor device, in contrast to the whole system, can be secured against malicious attacks.

2.8 Object Detection and Camera Interface

As previously noted, the computationally expensive task chosen for implementing an edge-computing use case is object detection, namely the identification of objects of interest in an image or video feed. In this project, object detection is happening in real-time using camera hardware that provides the appropriate image data to the detection algorithm.

2.8.1 Object Detection Algorithm

The object detection algorithm used in this system is the YOLOv5 (You Only Look Once version 5) [18] which provides real-time detection. It offers a number of object detection models based on the Microsoft COCO (Common Objects in Context) [19], a large-scale dataset with the aim of solving three fundamental research challenges in the field of scene understanding. These challenges include the detection of non-traditional viewpoints or angles of objects, contextual reasoning between different objects, and the accurate 2D detection of semantical objects [20].

Table 2.2 lists the available pre-trained models from the YOLOv5 library with various parameter sizes, each requiring different processing needs [21].

The model with the lowest computational complexity, while providing sufficient accuracy should be chosen. Model selection is of particular importance in the use-case context described, as common edge network devices have limited processing capabilities and often

Table 2.2: YOLOv5 Pretrained Checkpoints

Model	Size (pixels)	CPU speed (ms)	Parameters (million)
YOLOv5n	640	45	1.9
YOLOv5s	640	98	7.2
YOLOv5m	640	224	21.2
YOLOv5l	640	430	46.5
YOLOv5x	640	766	86.7

lack GPGPU (General Purpose Computing on Graphics Processing Units) that would accelerate these tasks. Model performance will be discussed in Section 3.

2.8.2 Camera Serial Interface (CSI) & Camera Hardware

Concerning camera hardware, a Raspberry Pi Camera Module 3 containing the Sony IMX708 sensor was selected [22]. The camera module connects with the host board using the Camera Serial Interface (CSI), which is preferred over USB as it provides a dedicated bus with a maximum bandwidth of 5.8 Gbit/s and has the ability to transfer raw uncompressed image data allowing for more precise control over image processing parameters. The output protocol used for this specific device is RAW10, which represents each pixel with 10 bits.

2.8.3 Object detection & Camera Python Scripting

Python development support for this specific camera is provided with the picamera2 [23] module. We implemented a python script that provides camera functionality, YOLOv5 model execution, result parsing, and transmission with the LoRa network library.

The script initializes the camera module, captures a single image, and stores it as a file on a ramdisk location so as to minimize input/output latency. A predefined YOLOv5 model (yolov5n for instance) processes the image and generates a string with the object detections. The output is parsed to a Python-to-C bind function that adds the detections from previous image instances and transmits the generated data after a defined time period. The transmitted message along with the detected objects contains a value describing the total images processed in that time period. Code implementing time measurements exists between method calls in order to generate model execution statistics which are then used in Section 3 for

evaluation purposes.

In the current system configuration, the camera output consists of images instead of a video feed. The script contains two functions, one that captures the image from the camera and another responsible for executing the detection model. This execution style has the advantage of higher resolution availability from the camera module, as the maximum image resolution supported is 4608 × 2592 while the supported video resolution is up to 1920 x 1080, and more efficient resource usage due to the high likelihood of dropped video frames caused by the bottleneck of model processing. For instance, a video feed of 10 frames per second oversaturating the detection model that would be only capable of processing 5 frames per second would result in the rest of the frames being dropped, thus wasting CPU cycles, camera utilization and bus bandwidth.

Because the function that captures the images requires as much as 300 ms to execute, which is comparable to the object detection algorithm execution time (about 700 ms), these functions need to be called concurrently. To avoid concurrency execution issues, the frames captured are stored interchangeably in two files, when the first is being used for storing an image, the second one is being processed by the model. In this execution model, compared to object detection, the execution time of the rest of the functionality in the script is negligible.

After the message containing the detections for a time duration is received by the gateway, a percentage value is calculated for each detected object that describes the cumulative, repetitive appearance of that object for this time interval. For instance, a single object appearing in half of the processed images during a one-minute interval will produce a value of 0.50 (50%).

$$Cumulative\ prediction\ value = \frac{Total\ detections\ of\ this\ object}{Number\ of\ images\ processed}$$

This cumulative prediction value is useful for reducing false positives, namely objects falsely detected that did not actually exist in an image, as lower values would suggest a higher possibility of a false positive, and values closer to 1.00 (100%) indicate a certainty in the prediction (considering a static, fixed camera setup which this project is aimed for).

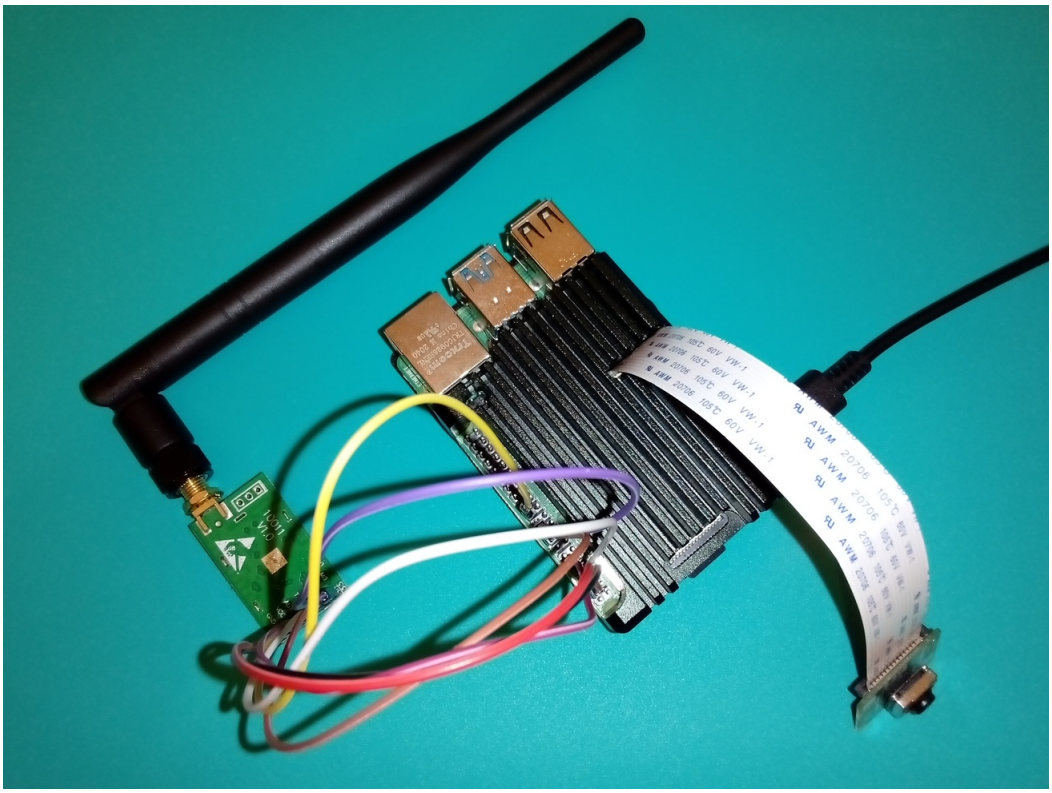

Figure 2.3: Raspberry Pi 4 equipped with LoRa and camera module.

2.9 Sample Implementation of the System

Depending on the specific use case, the deployment of the network and object detection nodes can manifest in many forms. In this Section, a reference configuration is presented in order to provide some examples of possible applications and highlight the technologies used.

2.9.1 Reference Implementation featuring Object Detection in the Edge

A simple transmitter-receiver setup where the transmitter is executing an object detection model and periodically relays its results is pictured in Figure 2.3 where a LoRa module is connected to the UART interface of the Raspberry Pi4 along with 3 GPIO pins for transmission mode selection and is supplied from the 5V power pin. The Camera module is connected to the CSI interface.

This node is deployed to the remote location needing the object detection functionality, for example for surveillance of an area, executes yolo inference, and relays the results. Multiple nodes could be deployed in several locations served by a single base station, a device containing another LoRa module, responsible for receiving those results and relaying them

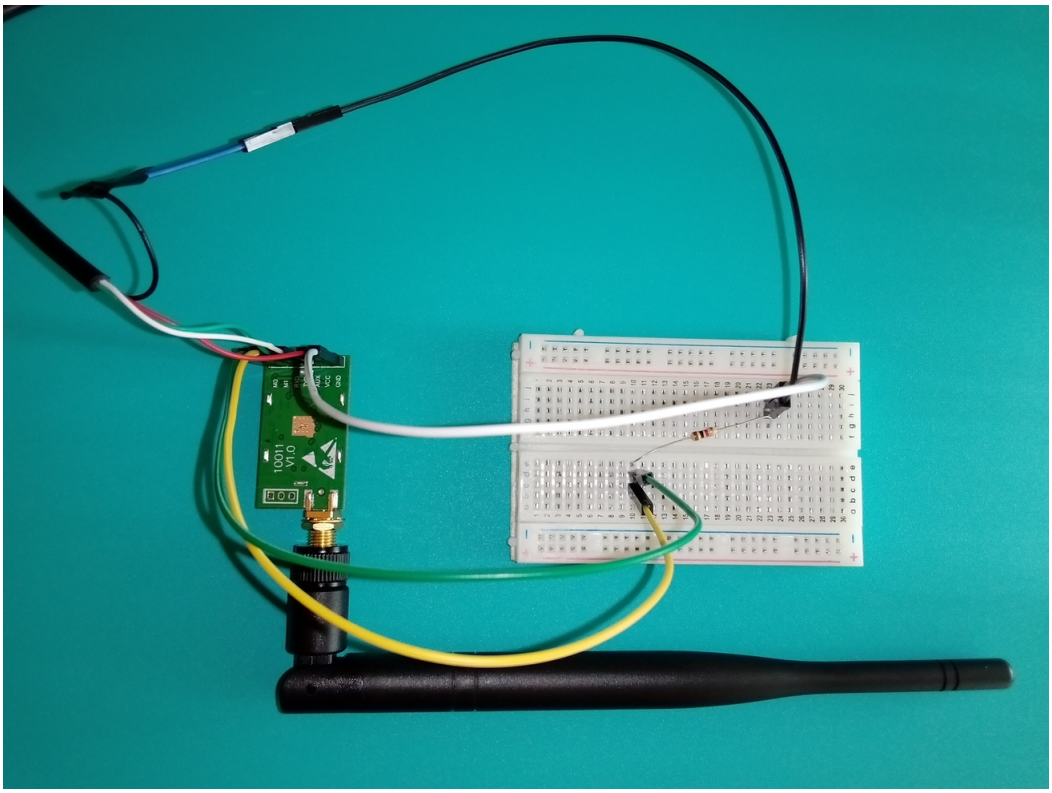

Figure 2.4: LoRa deployment to non UART-capable devices.

to an MQTT server.

The base station could be implemented by using either a Single Board Computer (SBC), which natively features a UART interface or a regular server by using a USB to UART adapter. Such a connection is displayed in Figure 2.4 where the LoRa module is supplied by the adapter's 5V pin. Due to the absence of programmable GPIOs on the host computer, the 2 control pins of the module need to be set to a logical zero value by connecting them to the ground through the use of a breadboard and a 1k-ohm resistor, otherwise the pins are measured to have a high enough floating voltage that it is interpreted as a logical one value and the module disables the transmission/reception mode. The third auxiliary pin outputs a high or low value depending on the transmission availability status of the module, however, it is not required by the design of this network Library and can be left unconnected.

An MQTT server is then deployed to a server either locally to the base station or in a remote Internet location, relaying the JSON structs of the LoRa packets to the front-end applications. In this implementation, the MQTT server is accessible through the TCP/IP protocol at port 1883 or for additional security over the WAN TCP/IP connection using the port 8883 featuring SSL (Secure Sockets Layer) cryptographic protocol.

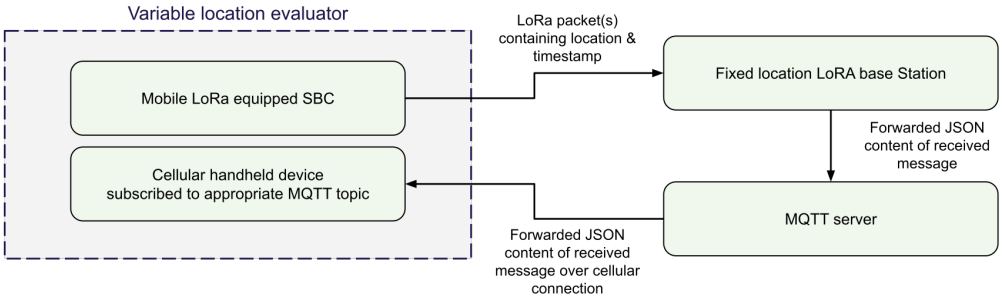

Figure 2.5: Overall system for testing LoRa reception, enabling live feedback to the evaluator.

2.9.2 Development of a mobile network node for range experimentation

Additional to the sample object detection system, some nodes using the same LoRa network library were implemented in order to evaluate the performance of the wireless subsystem. These nodes do not feature a camera module, however additional equipment for wireless range experimentation is used. Instead of transmitting packets containing the results of the object detection model, the packets contain essential data for the experiment such as the location of the node, indicating the versatility of the system for an assortment of applications.

Range and wireless coverage evaluation requires testing from multiple locations, where deploying a fixed station is not often possible. This necessitated the prototyping of a mobile, battery-powered LoRa device for carrying out range tests in various cases.

Figure 2.6 describes the hardware used for implementing the device. The mobile LoRa-equipped device is based on the Raspberry Pi Zero single-board computer which features low power consumption and small size while providing the necessary interfaces for the peripherals. The Raspberry Pi Zero SBC is equipped with a Global Navigation Satellite System (GNSS) module for determining its location and transmitting it with the LoRa module by using the developed network library. The transmitted packet is then received by an internet-connected base station on a fixed location, which then forwards the packet to an MQTT server. In order to avoid additional hardware complexity and power consumption, the mobile device is not equipped with a display. The evaluator can monitor the reception of packets from the base station in real-time by subscribing to the MQTT server using an appropriate application on a cellular handheld device, enabling immediate feedback on location and signal reception changes, as shown in Figure 2.5. Live feedback on packet reception enables

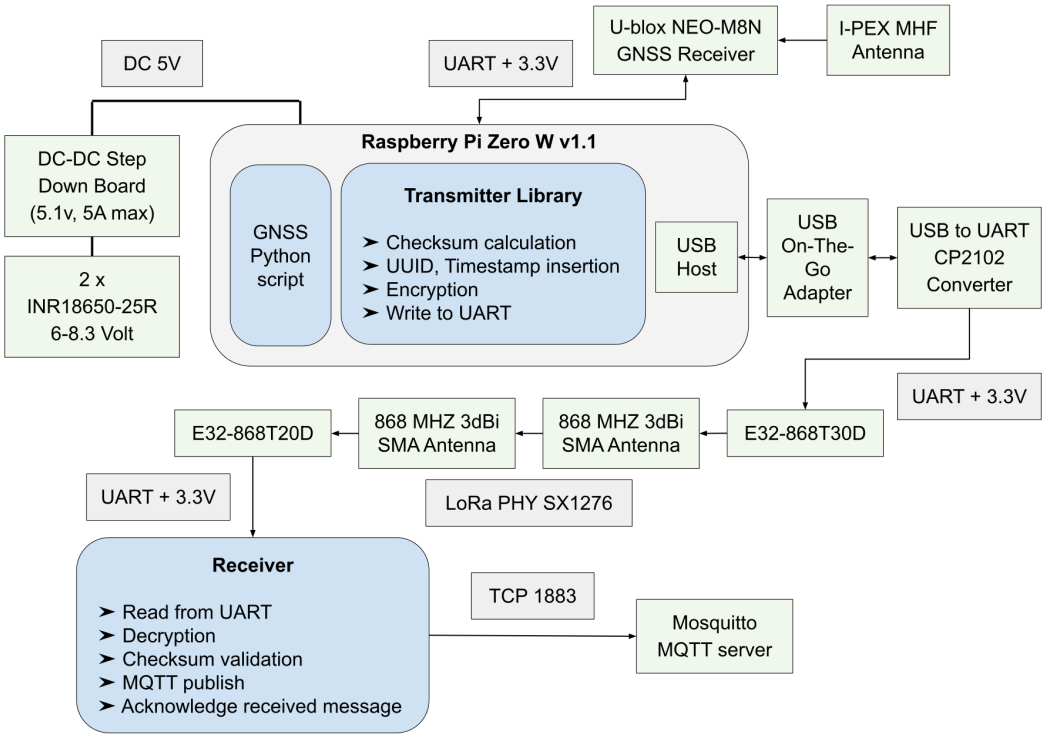

Figure 2.6: Design of the test system used for evaluating LoRa PHY and network range.

the evaluator to experiment with fine-tuning the reception by either performing subtle corrections to the device placement and antenna orientation or testing the impact of obstacles in the signal path.

For satisfying the power requirements of the mobile device and the peripherals, a power supply with a stable 5 volt output is required. The power supply is composed of two 18650 Lithium-ion batteries (Samsung INR18650-25R cells) connected in series, each providing 2.5 - 4.2 volts for a total of 5 - 8.4 volts [24], depending on the charge level. The voltage supplied to the modules is kept constant at 5V ± 5% by using a XL4015-based step-down DC to DC converter, able to provide an output of up to 5 amperes [25] for a total power of 25 watt (at 5 volt) according to the equation:

$$Power = Voltage \cdot Current$$

The Raspberry Pi Zero W, like most single-board computers, can be powered directly by providing 5 volts to the 5V GPIO power pins, depicted in Figure 2.8, avoiding the use of micro USB headers, as they are connected to the same 5V rail with the USB ports which is then connected to an onboard voltage regulator for providing the 3.3 and 1.8 voltage [26].

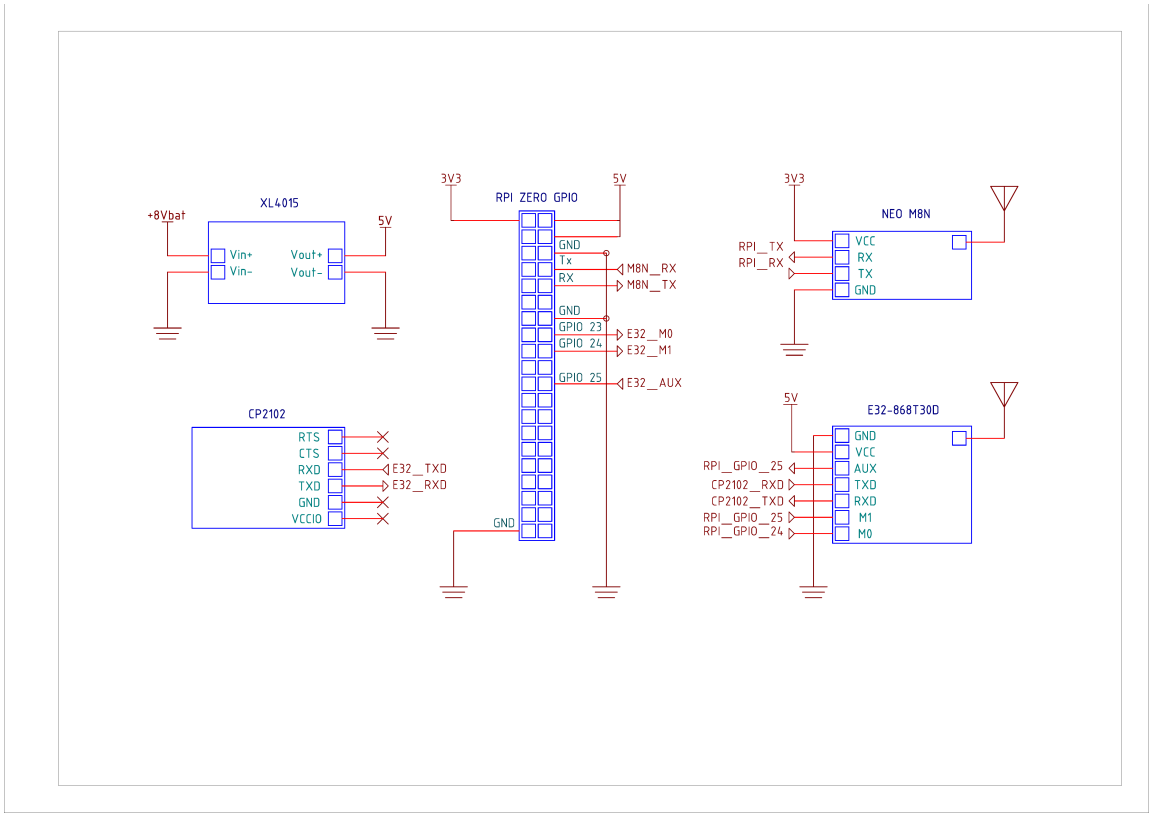

Figure 2.7: Mobile node circuit schematic.

Although powering most Raspberry Pi boards through the GPIO power pins bypasses the polyfuse (a resettable fuse, used to protect against overcurrent faults) located between the USB port and the 5V rail [27], the Raspberry Pi zero doesn't contain such fuse. An external fuse could be inserted between the power supply and the 5V power pin for overcurrent protection.

The second 5V power pin is connected to the power input of the E32-868-T30D transmitter, as the latter is recommended to be supplied 5 volts for operating at 30 dBm (1 watt) transmission power. The LoRa transmitter is also connected to GPIO pins 23, 24, and 25 where the first two configure the module's transmission mode and need to be set to a low value for regular operation, while pin 25 is set to a high value by the module when the receiver buffer is empty and can commence transmission. More details about connections with the GPIO pins are visible in the circuit schematic in Figure 2.7.

Figure 2.9 depicts the final form of the node's internals. The power supply consists of two INR18650-25R cells connected to a DC to DC converter. The GNSS module is located in the bottom right, supplied by the 3.3 volt pins and connected to the UART interface of the Raspberry Pi. A USB to UART converter (CP2102) is visible above the power supply, connected

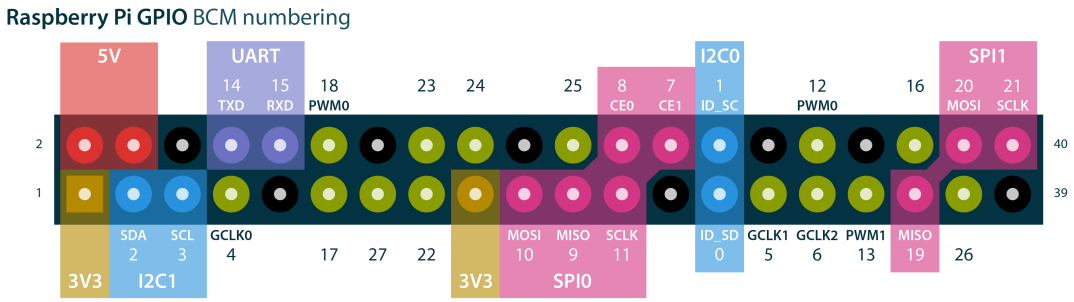

Figure 2.8: General Purpose Input Output (GPIO) of Raspberry Pi compatible headers [1].

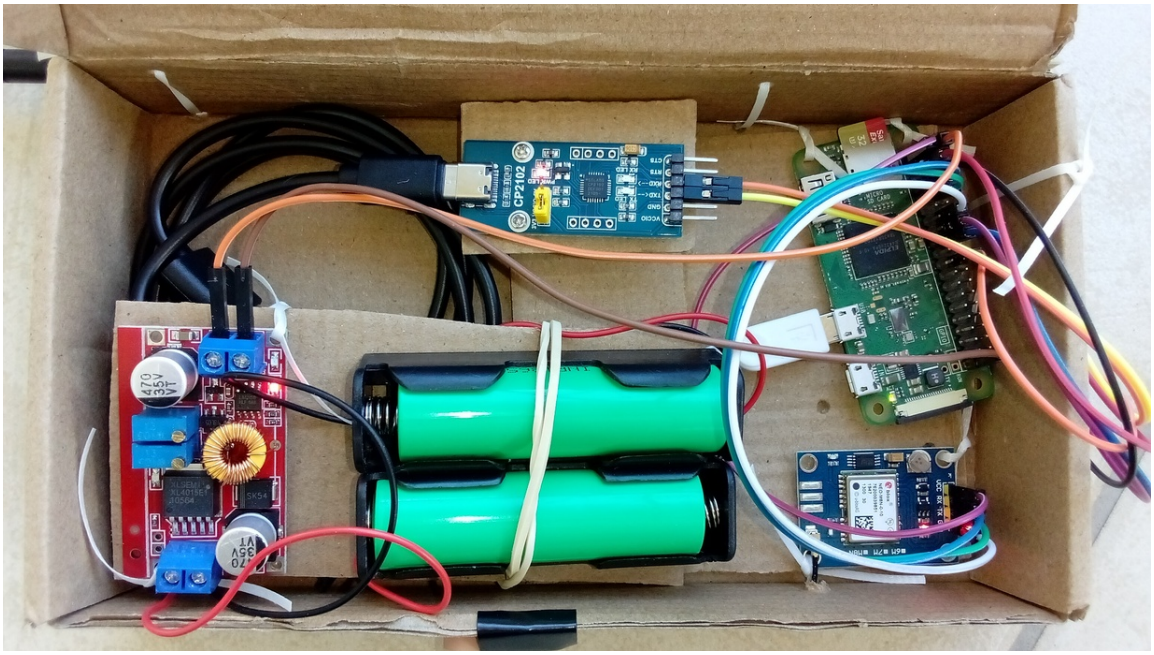

Figure 2.9: Internal view of the prototyped mobile node.

to the board with a microUSB On-the-Go adapter (for providing host USB capabilties to the board) and used for connecting the external LoRa module, as in the current configuration the GPIO pins don't provide a second UART interface. The 5V GPIO power pin supplies power to the Raspberry Pi Zero, while the power-only micro USB port is left unconnected. The USB adapter could be replaced with a USB-A one in order to eliminate the need for the 1 meter long USB-A to USB-C cable.

On the software side, a python script translates the NMEA (National Marine Electronics Association) strings provided by the GNSS module, mainly the "GGA - System Fixed Data" message which lists time and position data when a GNSS fix is acquired [28]. A message containing latitude, longitude, East/West, North/South indicators, altitude from mean sea level (MSL), and the number of satellites used is composed and transmitted as a payload using the

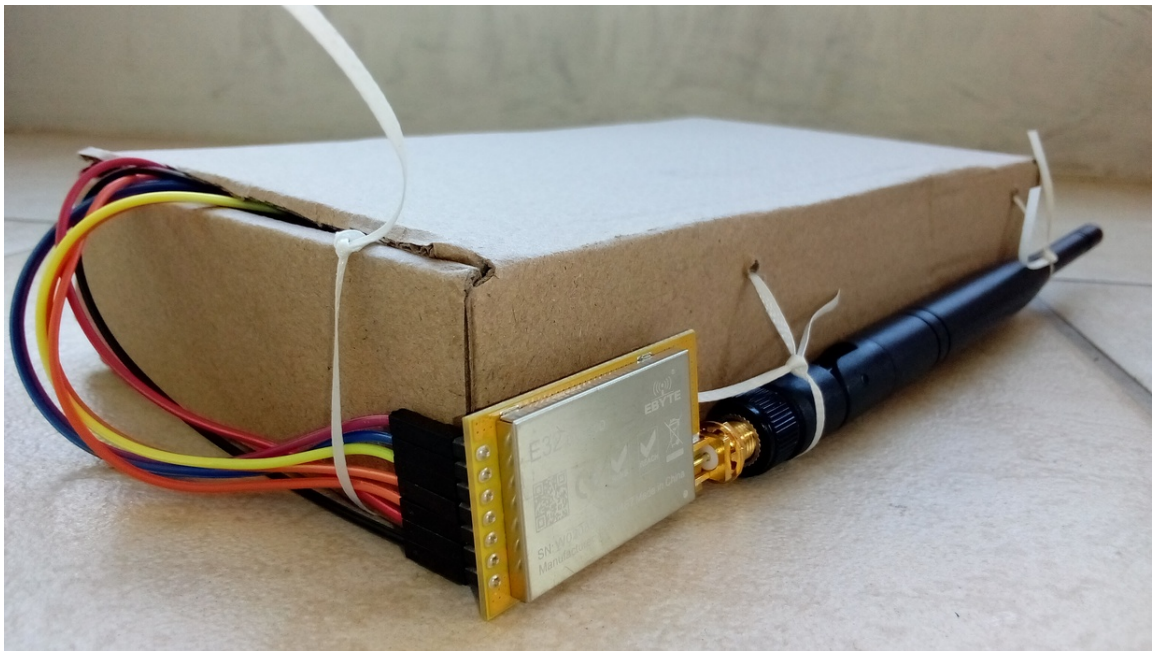

Figure 2.10: External view of the prototyped mobile node.

Python network library we implemented. In order to conform to regional duty cycle limitations in 868 MHz frequency plans, the transmission occurs periodically after a predefined time interval (set to one minute during the experimentation).

The LoRa module is visible in Figure 2.10, connected to an SMA, 3 dBi antenna. The antenna length is measured to be approximately 0.175 m, half the wavelength of the 868 MHz frequency used, thus efficient for these transmissions.

$$Wavelength = \frac{Speed\ of\ light}{Frequency} = \frac{299.792.458\ m/s}{868MHz} = 0,345m$$

The dBi value is the gain of the antenna. Lower values mean the antenna distributes energy more uniformly to all directions while higher values distribute more energy to the axis perpendicular to the antenna length. While a higher dBi value could extend the range of the transmission, it would impact the signal when the receiver and transmitter antennas are in different elevations or not at an ideal orientation.

2.9.3 Range Evaluation without WAN MQTT Feedback

With the intention of testing ranges spanning multiple kilometers, the base station should be positioned in a position that eliminates any building obstruction. Consequently, in the test described in Section 3.1.3, the base station is located on the sixth floor of an apartment

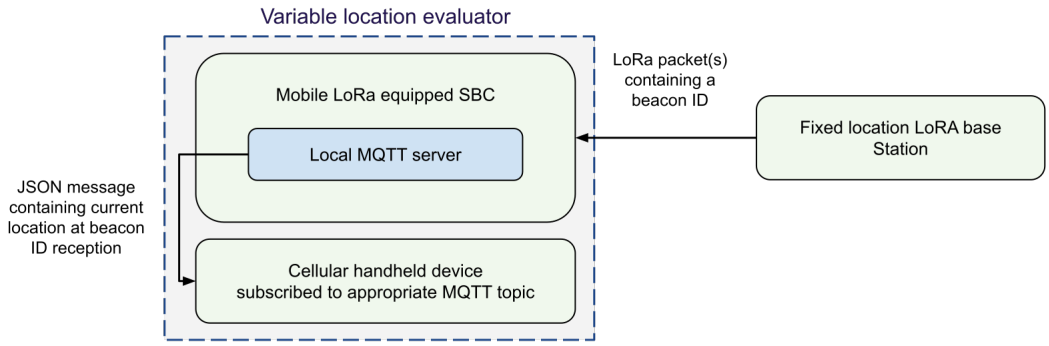

Figure 2.11: Updated system for testing LoRa reception in order to eliminate the need for Internet connectivity on the base station.

building. However, in order to eliminate the need for internet connectivity in the nodes and maintain the live reception feedback capabilities to the evaluator, the experimentation system has been reconfigured. As described in Figure 2.11, the base station is responsible for transmitting beacon packets with a predefined time period. When a beacon is received, the mobile node both saves it locally and publishes a JSON message containing its current location to an MQTT server hosted on the same device. The evaluator can receive these messages by subscribing to the local MQTT server with her mobile device using 802.11 connectivity. The overall system can now evaluate LoRa transmissions without depending on other WAN networking (for providing immediate packet reception feedback to the user), enabling the positioning of the base station to more locations.

For the base station, a VisionFive 2 SBC was used, featuring the JH7110 SoC with a RISC-V architecture (RV64GC ISA) and a Raspberry Pi-compatible 40-Pin GPIO Header. The SBC is powered by the same power module used on the mobile node through its 5V GPIO power pins. The UART interface needed for the LoRa module isn't provided out-of-the-box, as the exposed GPIO UART is used for serial console debugging. However, it is possible to enable additional UART interfaces by modifying the Device Tree Overlay (DTO) used for describing the board hardware during the initialization of the operating system.

The usual SoCs found on single-board computers provide multiple buses, utilized for communicating with peripherals. For instance, the JH7110 supports a maximum of six UARTs, seven SPIs (Serial Peripheral Interface), and seven I2C buses [29], but not all of them are ex-

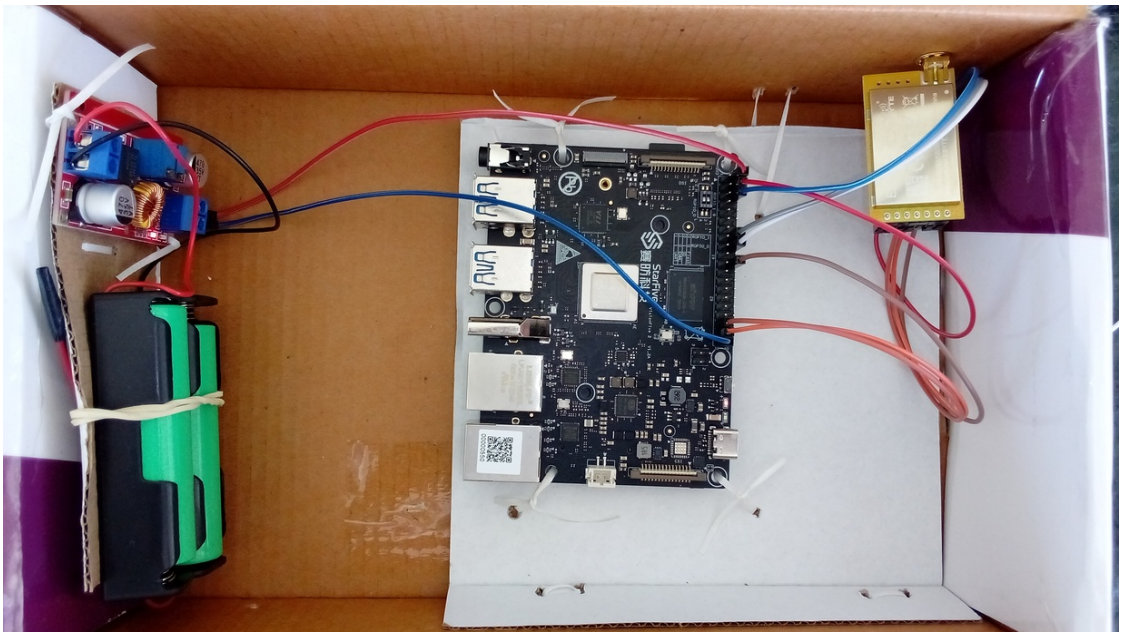

Figure 2.12: VisionFive 2 SBC on its prototyped enclosure.

posed due to the limited number of pins on the GPIO header. Instead, apart from the power and ground, the GPIO pins are multiplexed with a variety of functions that can later be configured by the developer, by loading an appropriate DTO file containing a device tree that describes non-discoverable hardware to the kernel.

Hardware vendors provide a device tree source file with structures describing the device hardware. After configuring the source file for exposing our desirable configuration, it is compiled with a device tree compiler to a binary device tree blob (DTB) which is then used by the bootloader [30].

As for the purpose of this project, an appropriate DTB file for exposing a second UART interface to the GPIO header of the Raspberry Pi Zero did not provide any effect, resulting in the requirement of the USB to UART adapter. However, this operation was successful for the VisionFive 2 SBC where two additional UART interfaces were exposed to the GPIO pin pairs 60 with 63 and 38 with 40 for a total of 4:

- Uart0 used for serial console debugging, mapped to /dev/ttyS0.
- Uart1 enabled for LoRa module communication, mapped to /dev/ttyS1.
- Uart2 enabled for an optional GNSS module mapped to /dev/ttyS2.
- Uart3, Uart4, Uart5 disabled.

The assembled form of the node containing the VisionFive 2 SBC is presented in Figure 2.12, where the battery-backed power supply module is visible powering the system through the 5V power pins of the SBC. The E32-868T30D transceiver is located on the right, connected to the GPIO pins 54, 51, and 50 (positioned at pin numbers 16, 18, and 22) for operation signaling and 63, 60 for the UART interface (pin numbers 35 and 37). A 3 dBi antenna is installed on the outside in order to eliminate interference from the enclosure.

Chapter 3

Evaluation

After the design and prototype implementation of the system, we performed a series of experiments to validate its functionality and evaluate its performance. The following aspects were evaluated:

- Range of the LoRa network that allows reception of packets with an acceptable loss rate. This is a first-class validation metric of the functionality of the system, as range is the main advantage of LoRa over other wireless Internet of Things (IoT) technologies, allowing communication with remote areas.

- Object detection model execution performance (measured in frames processed per second) and power consumption on a Single Board Computer (SBC).

- Comparison with other systems not implementing the edge computing paradigm (as transmission of the otherwise processed frames requires a significant amount of bandwidth).

3.1 LoRa Range Evaluation

The physical layer utilized in this system, LoRa, features a wide area coverage as a main advantage of this technology, which is the reason it was chosen for this implementation. The ability to establish long-range links enables the deployment of Wide Area Networks (WANs) being served by a single base station without the aid of repeaters, thus (i) decreasing deployment costs, and (ii) providing coverage to remote, unconnected areas. In the following

sections, we describe the parameters that impact the LoRa communication range and validate the range capabilities in real conditions.

3.1.1 LoRa Modulation Parameters

LoRa range and data transmission rate performance can be tuned by configuring the spreading factor, a parameter of its modulation as previously discussed in Section 2.3, the transmission bandwidth, and the coding rate.

LoRa utilizes the Chirp Spread Spectrum technology where symbols called chirps carry the transmitted bits. Spreading factor (SF) refers to the number of chirps used to encode each bit of information. SF can take 6 values between 7 and 12 with higher values providing a longer range but lower data rate. Essentially, an increase of SF by one value halves the data rate [9] [31].

Bandwidth refers to the frequency spectrum where the majority of the transmission power is directed. It is proportional to the data rate, where doubling the bandwidth results in approximately double throughput.

Finally, the coding rate is the proportion of the total transmitted bits that carry network layer data. In LoRa, the coding rate can take values between 4/5, 4/6, 4/7, and 4/8.

The following equation can be applied to the LoRa modulation parameters in order to estimate the data rate of the communication [32]:

$$Datarate = SF \cdot \frac{\frac{4}{4+CR}}{\frac{2^{SF}}{BW}} \cdot 1000$$

Where:

- Datarate is the transmission data rate in bits per second
- SF is the spreading factor with possible integer values 7, 8, 9, 10, 11 and 12
- CR is the coding rate with the possible values 1, 2, 3 and 4
- BW is the transmission bandwidth in KHz

The LoRa modules used, E32-868T20D and E32-868T30D, provide a programmable air data rate between 0.3 and 19.2 kbps. The module has a predefined configuration of LoRa parameters that correspond to each data rate. Using a wireless spectrum analyzer, the SDR++

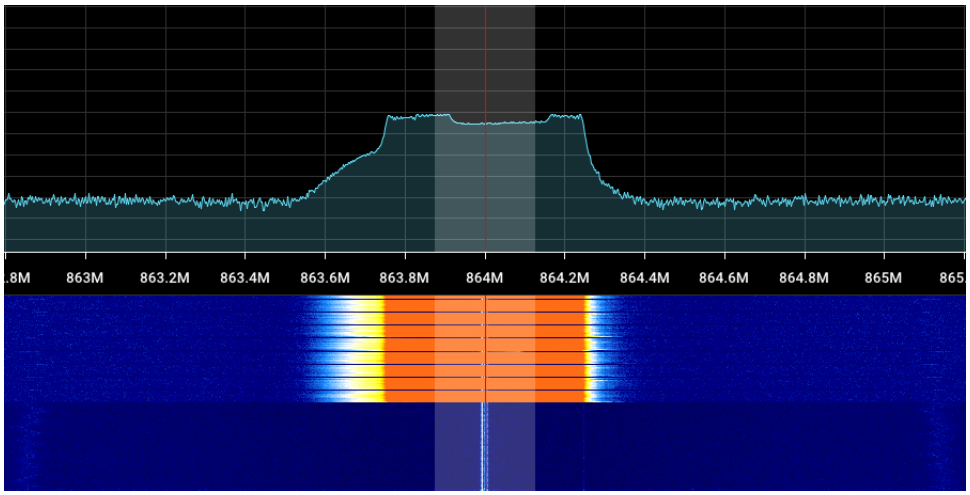

Figure 3.1: Snapshot of the spectrum analyzer during a LoRa transmission at 864 MHz, with an air data rate of 2.4 kbps.

software [33] combined with an SDR (Software Defined Radio) receiver, the transmission bandwidth was observed after transmission at different rates. In Figure 3.1, a snapshot of the spectrum analyzer is displayed, where the transmission bandwidth can be observed at 500 KHz in the orange waterfall area that corresponds to a high power spectral density. Small periodic intervals with no transmission can also be observed in the waterfall. These occur between transmissions of successive LoRa packets in the physical layer. The spreading factor value was estimated using the above LoRa modulation equation, assuming a coding rate of 4/5 and 12 SF for a 0.3 kbps transmission as noted on the module's manual [34]. Table 3.1 lists the calculated LoRa parameters for the modules used.

Table 3.1: E32-860T20D & E32-860T30D modulation schemes.

Programmable value	Data rate (kbps)	Bandwidth (KHz)	Estimated SF
1	0.3	125	12
2	1.2	250	11
3	2.4	500	11
4	4.8	250	9
5	9.6	500	9
6	19.2	500	7

By configuring the module data rate, a different spreading factor can be chosen, directly affecting receiver sensitivity. Application payload should also be contemplated, as higher

payload size will result in longer transmission times or even separation of data in multiple LoRa packets, increasing the possibility of interference and bit error.

In the following sections, we present the results of wireless range experiments with various interference and distance parameters. The main objective is to prove that in physical environments similar to those in our evaluation, a reliable wireless link can be established where application payload can be delivered in packets without data loss.

3.1.2 Urban & Suburban Network Coverage

An urban area is defined by densely located high rise buildings which, by having dimensions of several meters in the 3D space, are detrimental to signal propagation. While the full range capabilities of LoRa ca npt be demonstrated in this setting, a well positioned transceiver can provide hundreds of meters of effective connectivity, surpassing the usual 802.11 wireless networks. However, LoRa and more specifically this network implementation are oriented towards networking of remote locations, where other wireless interconnections are not feasible. Initial evaluation in urban areas provides a way to analyze the effect of object interference and other transmission properties on signal propagation, and to debug network nodes before a multi-kilometer range experimentation.

The most important requirement for successful exchange of data between two nodes is the existence of a relatively unobstructed path between them, called the Fresnel zone [35], an ellipsoidal area where any object existing in it, even the ground, can directly impact the transmission. The Fresnel zone explains the possible sources of disturbance and the multipath propagation for deflected waves following different paths to the destination, possibly arriving out of phase and destructively interfering with the main one.

Generally, the following parameters were considered in order to maximize communication range:

- Height of transmitter and receiver antennas from the ground, which limit the possibility of obstacles and ground interference as explained by the Fresnel zone.
- Line of sight availability between the nodes.
- Transmission power, which however is not proportionally connected to range.
- Packet length, as larger packets have a higher probability of bit flips, thus causing failure of checksum validation.

- LoRa modulation parameters: For this series of tests they were kept constant, with an air data rate of 2.4 kbps, corresponding to a spreading factor of 11, and 500 KHz bandwidth.

The first test was conducted using a 100 mW mobile transmitter, transmitting encrypted, 128-byte length packets and a base station positioned on the second floor of an apartment, about 10 meters above ground, using the system described in Figure 2.5. Packet reception is considered successful when its decrypted contents can be validated with its checksum value, meaning that no data corruption occurred. The maximum range achieved was 385 meters, where most successful transmissions were located in a straight unobstructed line with the receiver, after which building obstruction was apparent.

For the following tests an 1 watt transmitter was chosen, in order to enhance building penetration. In the second test, urban coverage improvement was evident since propagation was possible after multiple consecutive apartment buildings. Range higher that 500 meters was achieved with medium sized buildings in between the two nodes, and a maximum of 2120 meters was accomplished with relatively clear line of sight conditions.

A third test was executed using the setup of the second one, however assessing the range in a different geographical sector and positioning the base station better in order to avoid obstruction by nearby objects which are known to negatively affect the reception of radio waves. As a result the reception in the urban setting was more uniform and covering a larger area. Again, a clear path between the nodes was the determining factor for establishing a successful link, with a maximum distance of 1600 meters in a different direction from the previous experiment. While this was less than the previous experiment, this was caused by the lower elevation of the tested area, limiting the antenna height compared to obstructing buildings.

In summary, no correlation was noticed between distance increase and link deterioration for distances up to 2 km. Multiple high rise buildings obstructing the traversing path of the radio wave can totally block the communication. In cases of partial obstruction of the Fresnel zone, the packet reception performance can be improved substantially by increasing the height of the mobile node, even with height differences of as low as 0.5 to 1 meter. Moreover, the antenna alignment contributed greatly to overall reception effectiveness, as it was proved crucial to retain the antenna of the receiver perpendicular to the ground.

The results from the urban tests are depicted in the Figure 3.2, using the following color-

Figure 3.2: Map showing the successful LoRa reception points on the first set of experiments, with a maximum range of 2120 m.

coding:

- Red: First test with 100 mW transmitter.
- Green: Second test with 1000 mW transmitter.
- Blue: Third test with more efficient receiver positioning.
- Purple: Test for debugging the setup of the following experiment.

3.1.3 Rural, Open Area Network Coverage

The base station is configured to transmit a beacon, a 96-byte encrypted packet containing its Universally Unique Identifier (UUID) and a beacon ID starting from 0, approximately every one minute. During the previous experiments it was observed that the approximately one meter tall iron balustrade of the apartment had a significant impact on both transmission and reception of the node when it was positioned below it, as it was situated in the Fresnel zone and absorbed a part of the radiated wave. To overcome this obstacle, the station is raised on a 2 meter pole which is then secured to the iron balustrade, as seen in Figure 3.3. From

Figure 3.3: Position of the fixed base station on the sixth floor of an apartment building.

this position most other obstacles are below the antenna, enabling a direct path between the station and remote rural areas.

The use of an antenna extension cable was avoided purposely, given that the available coaxial cable could considerably attenuate the signal at 868 MHz, adding another parameter that would affect the range performance of the system, and increasing the possible failure points of the setup. Instead, the device was designed to be lightweight in order to remain supported on its elevated placement.

In an effort to validate the correct functionality of the system before proceeding to the remote test area, it is firstly assessed on a hill adjacent to the urban area, about 2800 meters from the station, where line of sight between the receiver and transmitter was confirmed during the placement of the base station on the sixth floor. In this location, reliable LoRa reception was validated in multiple spots, even in places featuring dense tree cover.

Reception was then tested on six more locations of increasing distance, where multiple LoRa packets where successfully received. A maximum range of 6320 meters was achieved, after which no more tests where conducted due to the limited traversability of the area. These locations are visible in Figure 3.4, noted as orange points along with those of the previous tests. Table 3.2 lists the range on each point, together with the elevation recorded on each site.

This environment is considered to be the ideal use case of the system, where wide land-

Figure 3.4: Map showing the successful reception points.

scapes being utilized for either agricultural purposes or solar panel array installations are not provided any landline communications connection, and power connectivity is usually already available for serving agricultural infrastructure. An example view of this environment where packet reception was successful is depicted in Figure 3.5. Clear view of the urban area where the base station is located, about 6 kilometers away, can be observed.

3.2 Power consumption of the transceiver nodes

The power consumption of the devices used in the last experiment was measured in order to estimate the battery life expectancy. A single INR18650-25R cell featuring 2500 mAh discharge capacity at nominal 3.6 V provides 9 Watt-hours of energy, for a total of 18 Watt-hours on the power supply modules used.

The base station with the VisionFive 2 SBC was measured at 2.4 Watts between beacon transmissions when connected to an external power supply, including the power losses of the power supply. This measurement does not take into account the consumption of the E32-868T30D module, which, at a 30 dBm transmission increases the total power draw by approximately 1 Watt. We can estimate the total average power consumption by calculating the energy requirement of a single beacon transmission:

$$Energy\_requirement(Watt \cdot hours) = \frac{Dt \cdot Power}{3600}$$

Figure 3.5: View from a typical rural area where packet reception from the mobile node was successful.

Where the Dt transmission duration is:

$$Dt = \frac{Data\_length}{Air\_data\_rate}$$

$$Energy\_requirement = \frac{Data\_length \cdot Power}{3600 \cdot Air\_data\_rate}$$

For a 96-byte beacon, with a 2.4 Kbps air data rate, the resulting energy requirement is 0.000089 Watt-hours per transmission. Supposing a beacon every minute, the average power consumption of the transmitter is 0.0053 Watt for this time frame, which is negligible compared with the consumption of the rest of the device. Therefore, the estimated battery life is 7.5 hours.

The mobile node with the Raspbery Pi zero W SBC had a power draw of 0.8 Watts, providing an effective battery autonomy of 22.5 hours.

3.3 Performance and Power metrics

The SBC device used for executing the object detection model is the Raspberry Pi 4, with a Broadcom BCM2711 System on Chip (SoC). As seen in Figure 3.6, this SBC features four Cortex-A72 (ARM v8) cores at 1.8GHz (SoC core frequency is 1.5 GHz, latest distributions default to 1.8 GHz) with 48 KiB Level 1 (L1) instruction cache and 32 KB L1 data cache for

Table 3.2: Receiver - Transmitter separation at LoRa reception.

Base station to mobile node separation (m)	GPS Elevation above mean sea level (m)
0 (Base Station)	95
2690-2910	94-108
3590	104
4010	116
5590	137
6140	140
6320	125
6300	126

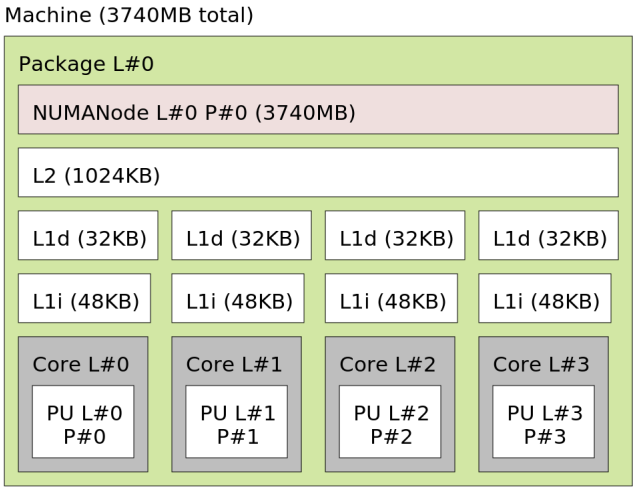

Figure 3.6: Raspberry Pi 4's core configuration.

each core following a 1 MiB L2 Cache. This specific model offers 4 GiB of LPDDR4-3200 memory. [36]

The script developed for object detection is responsible for capturing the frames from the camera module, executing the model, performing string parsing functions on the results, and transmitting the final data. However, the only computationally complex task is object detection with the rest of the functions providing statistically insignificant overhead.

Depending on the image resolution, the processing duration on this SBC ranges between 300 to 700 ms. per frame. Higher resolutions allow for more detail, decreasing the possibility of false negatives. The relation between image resolution and frames per second value can be seen in Figure 3.7, along with the throughput (Mpixels per second) achievd in Figure 3.8.

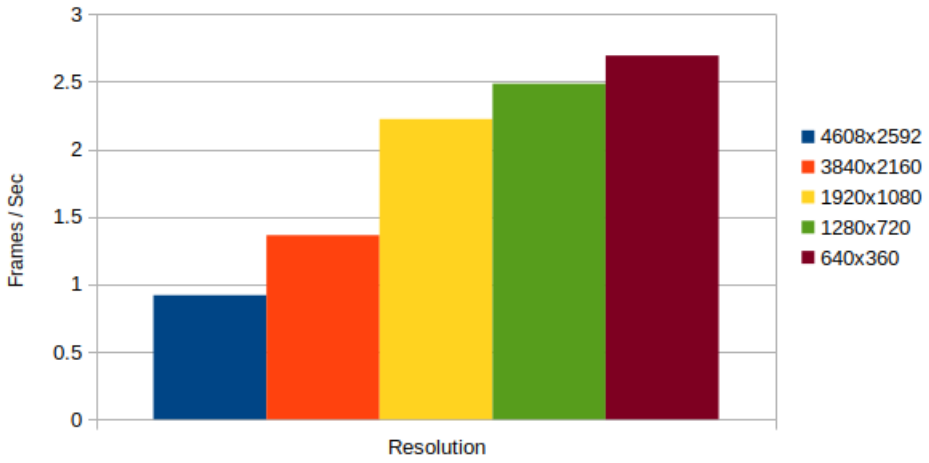

Figure 3.7: Resolution versus frames per second.

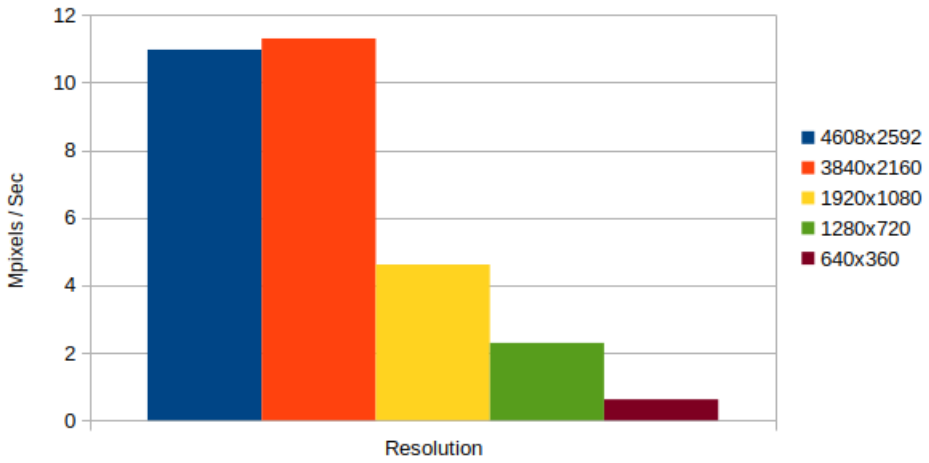

Figure 3.8: MPixels versus seconds.

Notice the roughly inverse relation of these values where higher resolutions allow for more efficient processing regarding the number of Mpixels.

As long as object detection precision remains at acceptable levels, tuning the size of the data to be processed can lead to performance gains as demonstrated in the experiment where we vary image resolution. Another way to increase the framerate performance of the system is to optimize the hardware parameters that affect performance. The model scales to multi-core configurations with all 4 cores of the SBC being utilized during execution. The frequency of the processor cores is another parameter that directly affects both the performance and the power consumption of the system according to the following equation [37]:

$$Pdyn = C \cdot V^2 \cdot f$$

- Pdyn is the dynamic power consumption of the CPU, caused by the switching states of

its logic gates, triggering the constant charge and discharge of the included capacitors.

- C is the switching load capacitance.
- V is the voltage of the processor core.
- f is the processor frequency.

An increase in processor frequency causes a theoretically proportional increase in performance (for computationally intensive tasks). However, higher frequencies require faster state switching of the logic gates, reaching a point where due to the logic gate's capacitance causing a time delay between voltage changes, it may not be possible for it to reach a voltage value low or high enough for it to be interpreted as the next logic state, causing data or instruction corruption.

In order to allow higher frequency values in the processor core, an increase in frequency must be accompanied by an increase in voltage, in order to allow faster logic state switching. Yet, the dynamic power consumption is related to the square root of the processor voltage, inducing a disproportional power increase, thus limiting energy efficiency. Consequently, there needs to be a compromise between energy efficiency and performance by selecting a high enough frequency value for acceptable performance, but at the same time remaining at a comfortable power consumption that does not stress the power supply or the cooling mechanism of the system.

We conducted an experiment for measuring the power consumption and model performance between different maximum processor frequency values. The SBC was supplied by the official Raspberry Pi 15W USB-C power supply [38], featuring a minimum efficiency of 81%, while being cooled by a passive heat sink. The results can be seen in Figure 3.3. The power consumption values were acquired using a power meter between the wall socket and the power supply therefore taking into account power supply inefficiencies.

The first section of Table 3.3 lists the parameters for each processor frequency. Four frequency values were tested, 1200, 1500, 1800, and 2000 MHz each accompanied by its operating voltage for that value and having a theoretical maximum performance improvement of 25%, 50%, and 66.67% compared with the initial frequency of 1200 MHz. The default frequency for this SoC is 1500 MHz, however, on relatively recent board revisions 1800 MHz can be attained with the option *arm_boost=1* on */boot/config.txt* configuration file (default

Table 3.3: Performance & Power Consumption per CPU frequency

Parameters				
Frequency (MHz)	1200	1500	1800	2000
(Core) Voltage	0.85	0.85	0.916	0.96
Max. Perf. gain	-	25.00%	50%	66.67%
Object Detection Test				
Average FPS	1.085	1.309	1.379	1.396
Standard Deviation (FPS)	0.005	0.0122	0.011	0.016
Perceived Perf. gain	-	20.64%	27.09%	28.66%
Power Usage (W)	4.8	5.1	5.9	6.5
Temperature (°C)	60.4	64.3	66.7	69.6
Power Efficiency (FPS/Watt)	0.226	0.257	0.234	0.215
CPU Benchmark Test				
Average Events/second	1878.678	2348.162	2816.032	3121.3
Standard Deviation (E/s)	1.085	0.377	2.205	1.305
Perceived Perf. gain	-	24.99%	49.89%	66.14%
Power Usage (W)	4.0	4.2	5.1	6.0
Temperature (°C)	61.3	63.8	66.2	70.1
Power Efficiency ((E/s)/Watt)	469.67	559.09	552.16	520.217

on recent distributions). The rest of the frequencies can be requested by setting the option *arm_freq=frequency_value* in MHz.

The second and third sections list the statistics and readings during the execution of object detection and a CPU benchmark for each frequency. These include the performance denoted as processed frames per second (or events per second, a benchmark statistic specific to the task it executed, a higher value means better performance), the perceived performance gain in comparison with the execution of the same task at 1200 MHz, the total power consumption of the device and the CPU temperature. The power efficiency is also noted; it is calculated from the ratio of useful work done to the average power consumption of the device during the execution of that test.

As expected, when running the object detection test, the maximum theoretical perfor-

mance increments were not achieved and the highest gain attained was 28.66% when moving from 1200 to 1800 MHz. Frequency increase does not cause a proportional performance increase in this test, instead, performance converges to a value. This is due to the fact that the algorithm is bottlenecked by other processor subsystems such as the memory bandwidth.

Power consumption readings were also acquired. When executing a benchmark test on all four CPU cores, there is an initial power draw of 4 Watts at 1200 MHz which is increased to 6 Watts at 2000 MHz. However during the execution of the object detection test the device power draw was 4.8 Watts at 1200 MHz, up to 6.5 Watts at 2000. This is caused by the object detection using more peripherals, such as the camera, LoRa module and possibly more RAM bandwidth than the benchmark which is designed to specifically stress the CPU processing capabilities.

In both tests the power efficiency peaked at 1500 MHz, having a slight difference with 1800 and decreasing abruptly at 2000. Lower CPU frequencies are characterized by less efficiency as execution time increases, during which the power consumption cost of the peripherals remains constant. At higher frequencies, the voltage in the core power plane causes a disproportional power draw as described by the dynamic CPU power consumption equation mentioned previously in this section.

The temperature remained at low levels even at 2000 MHz, recording a highest value of 70.1 °C. The room temperature during the experiment fluctuated between 25 and 26 °C, therefore operating at conditions with significantly higher temperature may require either liming the CPU frequency or employing active CPU cooling.

Overall, the most suitable frequencies from the four tested, were the 1500 and 1800 MHz, with 1800 MHz offering significantly more performance at the cost of slightly less power efficiency.

3.4 Comparison with alternative non-Edge implementations

An equivalent system that is not following the edge computing architecture, thus requiring the transmission of the image frames to a remote server accessible through a WAN connection would have two main disadvantages: (i) increased power consumption due to the increased volume of the transmitted data and (ii) additional operating costs in the case of a metered connection.

Assuming a 1280x720 resolution, the captured images are approximately 105 KB each. This figure can fluctuate depending on the efficiency of the image encoder in the current environment setting. Using the edge computing-based system, the object detection can process 2.5 frames (of 1280x720 resolution) per second, producing 3600 seconds * 105 KB * 2.5 frames per second = 945 MB of data per hour.

3.4.1 LoRa vs LTE implementation comparison

LTE (Long Term Evolution) provides a variety of frequency bands, mostly in 3 categories, sub-1GHz used for relatively long-range communications, 1500-1800 MHz for intermediate usage, and 5 GHz bands for hotspot coverage in locations with increased concentrations of subscribers. For long-range communications that approximate the LoRa system described in this thesis, the sub-1GHz is more suitable, therefore the extended GSM (Global System for Mobile Communication) 900 MHz (B8) band will be examined in particular. For theoretical power consumption comparisons, the specifications of an M.2 LTE card will be used, the Quectel EM05-G Category. This card features support for the B8 band using the Wideband Code Division Multiple Access (WCDMA) (symmetrical 384 kbps downlink, uplink bandwidth) and LTE-FDD (Frequency Division Duplex) standards [39].

The maximum peak output power in the GSM 900 MHz band is 2 Watts for a class 4 user equipment (UE) device. An 1-Watt average transmission power will be considered, equal to the maximum power of the LoRa modules used in the experiments in Section 3.1.3. Using the equations in Section 3.2 (assuming a 384 kbps WCDMA connection) an image frame can be transmitted with a transmission duration of 2.1875 seconds and an energy requirement of 2.1875 Watt-seconds. Transmitting a frame every 3 seconds will result in an average power draw of 0.73 Watts which is significantly higher than the 0.0053 Watts for a LoRa module with a transmission every minute as calculated in Section 3.2. Additionally, the cloud-based system will require the transmission of 126 MB of data per hour.

3.4.2 Edge vs Cloud Processing & Power Consumption

While the comparison with cellular technologies concludes that the LoRa implementation is more efficient concerning the power consumption of the transceiver modules, the processing of the data itself, which is significantly more power-demanding, is determinant for the total power efficiency of the system.

The SBCs selected for IoT applications usually feature CPUs based on the ARM architecture, focusing on low power consumption while maintaining considerable performance in contrast to x86 based processors used on most cloud platforms [40]. Some x86 microarchitectures, targeted at mobile devices thus competing with ARM alternatives, are also optimized for power efficiency. On the contrary, the main advantage of cloud-based computing is the sharing of server resources and peripheral power consumption with multiple services whereas an SBC device may be deployed for a single purpose.

In this section we compare the power consumption of executing the object detection algorithm between the Raspberry Pi 4 used so far and a headless x86-based device. The x86-based device features an Intel Core i7-3770 processor with a 4-core, 8-thread, up to 3.9 GHz configuration [41] and 16 GiB DDR3 RAM memory.

For the power consumption comparison test, both devices executed the object detection algorithm by processing 2.5 frames of 1280x720 resolution per second, as mentioned before in Section 3.4. Two power readings were acquired from the x86 device, one including the consumption of the whole device, including the CPU and its peripherals, measured from a sensor before the device's power supply unit, the other being only the CPU consumption, reported from the Running Average Power Limit (RAPL) domain of the CPU. The results are visible in Table 3.4. The high standard deviation value of the server is caused by the CPU idling between frame processing.

Table 3.4: Server vs SBC power consumption in watt.

	Server Average (STDEV)	RPi Average (STDEV)
Device Idle Power	23.0 (6.4)	2.3 (0.2)
CPU Idle Power	3.64 (0.03)	- (-)
Device Exec. Power	32.0 (18)	5.7 (0.2)
CPU Exec. Power	10.6 (1.22)	- (-)

Supposing an ideal scenario, where the server is simultaneously utilized by other services, the power usage of its peripherals is not included in the comparison with the SBC; instead, the difference between power while idling is subtracted by the power during object detection execution. As a result, the power consumption caused by the execution is measured to be 9 Watts of which 6.96 is from the CPU and the rest from RAM memory, power supply losses, and other motherboard and peripheral circuitry.

However, the average power consumption of the SBC, measured at 5.7 Watts, is 36.7 % less than the difference in power noticed on the server thus suggesting that deploying SBCs for performing specific computational tasks provides the benefit of increased power efficiency compared to Cloud-based computing, due to the more optimized CPU architecture of SBCs for this purpose.

Chapter 4

Conclusion

In summary, this Thesis has explored the advantages of low power wide area networking and, in combination with the edge computing paradigm, investigated an alternative to the traditional server-based computing architecture. Through an in-depth analysis of LPWANs and deployment of computationally expensive algorithms on the edge, this study has demonstrated their positive aspects concerning network coverage in remote locations, power efficiency of wireless communications, and security considerations. The findings have highlighted the potential of deploying LPWANs for connecting distant IoT devices without relying on other infrastructure while processing the sensor data close to their origin.

4.1 Summary and conclusions

This Thesis investigated the potential synergies between LPWANs and edge computing, proving their advantages in IoT applications. By examining deployment cases such as object detection on the network edge, this research has shown the cost-effectiveness, efficiency, and processing capabilities of edge computing which have been steadily improving over time, bridging the processing gap between server and SBC devices. Moreover, the study has highlighted the positive impacts that these systems can bring to the IoT segment, including connectivity with remote, not otherwise accessible areas and reducing resource utilization on existing networks.

While the findings of this thesis have been promising, it is important to acknowledge certain limitations. The feasibility and scalability of the aforementioned architectures are limited by topographical factors, where the presence of a line of sight between the transmitter and

lenges and opportunities in this domain. It has enabled the writer to make informed design decisions and optimize system performance based on the results obtained from the evaluation.

The insights gained from this experience not only contributed to the successful completion of this Thesis, but will also serve as a valuable asset in the pursuit of future endeavors in the field of edge computing and low-power networking.

receiver affects the availability of a long-range connection. Moreover, the processing power of SBCs in the IoT field, while constantly improving, can still bottleneck the throughput of an edge computing system, a disadvantage aggravated by the fact that such a device setup may be required to operate under certain power consumption limitations.

In conclusion, this Thesis has shed light on the significant role that edge computing and low-power networking can play in more efficient sensor systems, by distributing data processing away from a central server. By harnessing the strengths of both paradigms, organizations can unlock new possibilities and pave the way for a more connected and efficient future in IoT systems.

4.2 Knowledge acquired during the development

Extensive and invaluable knowledge was acquired during the process of developing and evaluating the edge computing and LPWAN system.

Initially, a deep understanding of edge computing in combination with LPWAN architecture and its benefits was acquired. The development process provided insights into the challenges and considerations involved in a network system optimized for IoT applications that requires a focus on limiting resource utilization, security, and reliability aspects. The utilization of the LoRa modules for the purposes of this system provided experience in low-level programming with the UART protocol and the GPIO pins typically used in embedded software systems.

Furthermore, the evaluation of the wireless networking capabilities proved to be challenging in various ways as it necessitated the development of fixed and mobile LoRa-equipped nodes. The experiments that were carried out during the evaluation required the implementation of solutions spanning software development and hardware prototyping that conformed to each test case. The systems needed in each test case entailed lengthy debugging and quality assurance before proceeding to the final evaluation, in order to ensure the validity of the produced results and avoid repetition of time-consuming field testing, while the evaluation itself provided a practical understanding of the relation between LoRa packet reception and factors such as range, object obstruction, and antenna orientation.

Overall, the knowledge acquired throughout the development of the edge computing and low-power networking system has provided a solid foundation in understanding the chal-

Bibliography

[1] Pinout.xyz Project. Raspberry pi pinout. https://github.com/pinout-xyz/Pinout.xyz/blob/master/graphics/, 2016. Online; accessed April 17, 2023.

[2] Ylli Rama and M. Alper Özpınar. A comparison of long-range licensed and unlicensed lpwan technologies according to their geolocation services and commercial opportunities. Technical report, Mechatronics Engineering Department Istanbul Commerce University Istanbul, Turkey, 2018.

[3] Weiping SunMunhwan, ChoiSunghyun, and ChoiSunghyun Choi. Ieee 802.11ah a long range 802.11 wlan at sub 1 ghz. Technical report, Department of ECE and INMC, Seoul National University, Seoul, Korea, 2013.

[4] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges, 2016. IEEE internet of things journal.

[5] Mahadev Satyanarayanan. The emergence of edge computing. Technical report, Carnegie Mellon University, 2017.

[6] SEMTECH. *SX1276/77/78/79 Datasheet - 137 MHz to 1020 MHz Low Power Long Range Transceiver*, 2020. Rev. 7.

[7] EBYTE. *E32-868T30D User Manual*, 2020. Rev. 1.6.

[8] Dimitrios Zorbas. Design considerations for time-slotted lora(wan). Technical report, Tyndall National Institute, University College Cork CONNECT Centre, 2020.

[9] Anup Marahatta 1, Yaju Rajbhandari, Ashish Shrestha, Ajay Singh, Anup Thapa, Francisco Gonzalez-Longatt, Petr Korba, and Seokjoo Shin. Evaluation of a lora mesh network for smart metering in rural locations. *electronics MDPI*, 2021.

[10] Taoufik Bouguera, Jean-François Diouris, Jean-Jacques Chaillout, and Randa Jaouadi 3and Guillaume Andrieux. Energy consumption model for sensor nodes based on lora and lorawan. *MDPI, Sensors 2018*, 2018.

[11] The OpenSSL Project Authors. Man page documentation; evp - high-level cryptographic functions. `https://www.openssl.org/docs/man1.0.2/man3/evp.html`. [Online; accessed June 2023].

[12] kernel.org/doc/man-pages. *termios — Linux manual page*, 2021. 5.13 of the Linux man-pages project.

[13] MQTT.org. Mqtt: The standard for iot messaging. `https://mqtt.org/`. [Online; accessed June 2023].

[14] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. Technical report, IBM Zurich Research Laboratory, Switzerland, IBM UK Laboratories, Hursley, UK, 2008.

[15] Eclipse Foundation. Eclipse mosquitto, an open source mqtt broker. `https://mosquitto.org/`. [Online; accessed June 2023].

[16] National Institute of Standards and Technology (NIST). *Advanced Encryption Standard (AES)*, May 2023.

[17] Kenneth G. Paterson and Arnold Yau. Padding oracle attacks on the iso cbc mode encryption standard. Technical report, Royal Holloway, University of London, 2004.

[18] Ultralytics. Yolov5 github repository. `https://github.com/ultralytics/yolov5`. [Online; accessed May 2023].

[19] COCO contributors and collaborators. Coco dataset homepage. `https://cocodataset.org/`. [Online; accessed June 2023].

[20] Tsung-Yi Lin et al. Microsoft coco: Common objects in context. *Springer International Publishing, 2014*, 2014.

[21] Ultralytics. Yolov5 github repository. `https://github.com/ultralytics/yolov5`, 2023. [Online; accessed 28-March-2023].

[22] Raspberry Pi Ltd. *Raspberry Pi Camera Module 3 Product Brief*, 2023. Published January 2023.

[23] picamera2 contributors. New libcamera based python library. `https://github.com/raspberrypi/picamera2`. [Online; accessed June 2023].

[24] Samsung SDI Co. Ltd. Energy Business Division. *Lithium-ion rechargeable cell for power tools Model name : INR18650-25R*, 3 2014. Rev. 1.0.

[25] XLSEMI. *5A 180KHz 36V Buck DC to DC Converter XL4015*. Rev. 1.8.

[26] Raspberry Pi. *Raspberry Pi ZeroW reduced schematics*, 2015. Rev. 1.1.

[27] Raspberry Pi. *Raspberry Pi 3 Model B+ (Reduced Schematic)*, 2018. Rev. V1.0.

[28] SiRF Technology Inc. *NMEA Reference Manual*, December 2017. Rev. 2.1.

[29] StarFive. *StarFive JH7110 Datasheet*, March 2023. Rev. 1.51.

[30] The kernel development community. Linux and the devicetree. `https://docs.kernel.org/devicetree/usage-model.html`. Online; accessed April 23, 2023.

[31] et al Petäjäjärvi, Juha. Performance of a low-power wide-area network based on lora technology: Doppler robustness, scalability, and coverage. *International Journal of Distributed Sensor Networks 13.3*, 2017.

[32] Moroz Vladimir. Low-frequency band lora network: Data rate optimization. Technical report, Metropolia University of Applied Sciences, 2021.

[33] Sdrplusplus github repository. `https://github.com/AlexandreRouma/SDRPlusPlus`. [Online; accessed June 2023].

[34] EBYTE. *SX1276/SX1278 Wireless Modules E32 Series User Manual*, 2018. Rev. 1.3.

[35] Athanasiadou Georgia E. Incorporating the fresnel zone theory in ray tracing for propagation modelling of fixed wireless access channels. *IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, 2007.

[36] Raspberry Pi (Trading) Ltd. *Raspberry Pi 4 Model B Datasheet*, June 2019. Release 1.

[37] Massoud Pedram Jan M. Rabaey. *Very-Large-Scale Integration*. Springer Science & Business Media, 2012.

[38] Raspberry Pi (Trading) Ltd. *Raspberry Pi 15W USB-C Power Supply Product Brief*, November 2021.

[39] Qectel Wireless Solutions Co., Ltd. *Quectel EM05 Series Product Specifications*, November 2023.

[40] Gupta Khushi and Tushar Sharma. Changing trends in computer architecture: A comprehensive analysis of arm and x86 processors. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology 7*, 2021.

[41] Intel® core™ i7 desktop processor comparison chart. `https://www.intel.com/content/dam/support/us/en/documents/processors/corei7/sb/intelcorei7comparisonchartrev8.pdf`. Online; accessed 26-May-2023.

APPENDICES

Appendix

LoRa Network layer library

The implemented network library for use with LoRa modules featuring a UART interface. The library itself can be used between any serial connection or pseudotty:

https://github.com/georkapetanos/p2plink

Sample use cases can be found in *python_scripts* and *Evaluation* directories which feature the system used during the evaluation.