

UNIVERSITY OF CAPE COAST

BUS TRACKING:
A REALTIME MONITORING SYSTEM FOR SHUTTLES

SETH ADDO

TEYE SHINE SACKITEY

© 2023

Seth Addo

Teye Shine Sackitey

UNIVERSITY OF CAPE COAST

BUS TRACKING:

A REALTIME MONITORING SYSTEM FOR SHUTTLES

BY

SETH ADDO

TEYE SHINE SACKITEY

A project work submitted to the Department of Computer Science and Information Technology of the School of Physical Science, College of Agriculture and Natural Sciences, University of Cape Coast, in partial fulfilment of the requirements for the award of Bachelor of Science Degree in Computer Science.

AUGUST, 2023

DECLARATION

Candidates' Declaration

We hereby declare that this project work is the result of our own original research and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature: Date:

Name: Seth Addo

Candidate's Signature: Date:

Name: Teye Shine Sackitey

Supervisor's Declaration

I hereby declare that the preparation and presentation of this project work was supervised in accordance with the guidelines on supervision of the project laid down by the University of Cape Coast.

Supervisor's Signature: Date:

Name: Mr. Isaac Armah-Mensah

ABSTRACT

This development research aims to increase productivity and reduce time mismanagement among students who rely on buses for transportation. The study proposes a system that includes both hardware and software components to track multiple buses on campus and display their location data in real time through a mobile application. The methodology involves building bus nodes that transmit their location data using the LoraWan gateway and developing an app that receives and displays the location data on a map. The preliminary results indicate that the researchers were able to set up the bus nodes and LoraWan gateway, retrieve data from the nodes, and display the location on a map. The findings suggest that the proposed system has the potential to enhance transportation efficiency and reduce wait times for students. Further research is needed to assess the usability and effectiveness of the system on a larger scale.

ACKNOWLEDGEMENTS

Our profound gratitude and indebtedness go to the Almighty God and to our supervisor Mr. Isaac Armah-Mensah of the University of Cape Coast for his guidance and suggestions towards the success of our work. We also are grateful to our various guardians for their emotional, psychological and financial support given to us to be able to come out with this project.

DEDICATION

This project work is dedicated to our parents, and our various family members, and friends for their love and support.

TABLE OF CONTENTS

DECLARATION	ii
ABSTRACT.....	iii
ACKNOWLEDGEMENTS.....	iv
DEDICATION	v
TABLE OF CONTENTS.....	vi
TABLE OF FIGURES	ix
CHAPTER ONE	1
INTRODUCTION	1
Background to the Study.....	1
Statement of Problem.....	3
Research Questions.....	3
Purpose of the Study	4
Significance of Study.....	4
Objectives	5
Organization of the Rest of the Study	6
CHAPTER TWO	7
LITERATURE REVIEW.....	7
Related Work	8

CHAPTER THREE	22
RESEARCH METHODOLOGY	22
Introduction.....	22
Requirement Gathering and Analysis	22
Functional Requirements	23
Non-functional Requirements	23
System Design	23
General System Design.....	24
Flowchart	26
Use Case Diagram	26
Web interface Design	27
Language Justification	29
Building the Hardware.....	31
Devices/sensors and usage:.....	31
Building the Front End	35
Building the parts - Hardware.....	37
Integration and Testing.....	37
System Deployment.....	38
CHAPTER FOUR.....	39
RESULTS AND DISCUSSION.....	39

Introduction.....	39
Design and Implementation	39
Testing and Evaluation.....	39
Unit Testing.....	40
Testing the nodes:	40
Testing the LoRaWAN gateway connection:	40
Testing the Web App:	41
Hardware Setup.....	51
Integration Testing	52
System Testing	52
Results.....	52
System Requirements	52
Conclusion	53
CHAPTER FIVE	54
SUMMARY, CONCLUSIONS AND RECOMMENDATIONS	54
Summary.....	54
Conclusion	55
Recommendations.....	55
REFERENCES	57
APPENDIX.....	60

TABLE OF FIGURES

Figure 1: System Design.....	24
Figure 2: Flowchart.....	26
Figure 3: Use Case Diagram	27
Figure 4: Web Interface Main	28
Figure 5: Web Interface Pickup section	28
Figure 6: Breakdown of the gateway	30
Figure 7: Raspberry Pi 3b	31
Figure 8: LoRaTTGO Node.....	33
Figure 9: 868MHz LoRa Antenna 5.8dBi Outdoor Fiberglass	34
Figure 10: 3D printed cases for Nodes	37
Figure 11: Login Page.....	42
Figure 12: Homepage.....	43
Figure 13: Available Shuttles	45
Figure 14: Set Reminder	46
Figure 15: Shuttle stations	47
Figure 16: Profile Page	48
Figure 17: About Page	49
Figure 18: Geolocation	50
Figure 19: Mounted Antenna	51
Figure 20: Device (Node with mini Antenna).....	51

CHAPTER ONE

INTRODUCTION

Background to the Study

The aim of this background study is to provide a comprehensive overview of the campus bus tracking system that utilizes LoRaWAN (Long Range Wide Area Network) technology. The system's primary objective is to reduce the waiting times of students on campus by providing real-time tracking and accurate arrival predictions of the campus buses. This study will explore the context, significance, and relevant concepts related to the project.

One of the common challenges faced by students on campus is the uncertainty and inconvenience associated with the campus bus transportation system. Students often experience long waiting times, unpredictability in bus schedules, and overcrowded buses. These issues can significantly impact their daily commute and overall campus experience.

LoRaWAN is a low-power, wide-area networking protocol that enables long-range communication between devices, even in areas with limited connectivity. It provides efficient and cost-effective connectivity for Internet of Things (IoT) applications, making it suitable for the implementation of a campus bus tracking system.

By leveraging LoRaWAN technology for bus tracking, the campus can experience several advantages, including:

- Real-time bus location tracking: Students can have access to the precise location of the buses in real-time, allowing them to plan their journeys more efficiently.
- Accurate arrival predictions: The system can utilize the bus location data to estimate the arrival time of the buses at different stops, enabling students to minimize waiting times.
- Improved service reliability: With accurate tracking and monitoring, the campus transportation department can better manage bus schedules, optimize routes, and respond promptly to any disruptions or delays.
- Enhanced student experience: Reduced waiting times and improved bus service reliability can contribute to a more convenient and pleasant campus commuting experience for students.

The bus tracking system utilizing LoRaWAN technology typically consists of the following components:

GPS-enabled tracking devices installed on buses: These devices collect and transmit the real-time location data of the buses to the LoRaWAN network.

LoRaWAN gateways: These devices receive the location data from the buses and forward it to the LoRaWAN network server.

LoRaWAN network server: It manages the communication between the gateways and the application server, ensuring the secure and reliable transmission of data.

Application server: It processes the received location data, performs data analytics, and provides the necessary interfaces for students to access the bus tracking information via a mobile app or web portal.

Statement of Problem

Waiting for shuttles at the university campus can be a frustrating and time-consuming experience for students. The lack of real-time information on when the shuttle will arrive often leads to long wait times at the shuttle station, which could result in missed classes or other important activities. To address this issue, there is a need for an efficient and reliable bus tracking system that utilizes LoRaWAN technology to provide accurate and real-time information on the location and ETA of the shuttles.

Research Questions

- I. What is the current state of bus tracking systems using LoRaWAN technology, and what are their limitations and advantages?
- II. How can LoRaWAN technology be utilized to design an efficient and reliable bus tracking system for university shuttles?
- III. What are the hardware and software requirements for developing a LoRaWAN-based bus tracking system, and how can they be implemented effectively?

- IV. How accurate and reliable is the location and ETA information provided by the proposed bus tracking system?
- V. What are the potential benefits of implementing the bus tracking system in terms of reducing wait times, improving shuttle service, and enhancing overall student experience on campus?

Purpose of the Study

The purpose of this study is to design and develop an efficient and reliable bus tracking system that utilizes LoRaWAN technology for university shuttles. The study aims to address the problem of long wait times at shuttle stations by providing accurate and real-time information on the location and ETA of shuttles. The study also seeks to evaluate the effectiveness and accuracy of the proposed bus tracking system in improving shuttle service and enhancing overall student experience on campus. The findings of this study are expected to contribute to the development of more efficient and reliable transportation systems for universities and other similar settings

Significance of Study

The proposed bus tracking system using LoRaWAN technology has the potential to significantly improve shuttle service and reduce wait times for students on university campuses. By providing accurate and real-time information on the location and ETA of shuttles, the system can help students better plan their

schedules and reduce the likelihood of missed classes or other important activities. Additionally, the system may also help reduce congestion and pollution on campus by encouraging more students to use shuttle services instead of driving their own vehicles. The findings of this study are expected to contribute to the development of more efficient and sustainable transportation systems for universities and other similar settings, ultimately benefiting the wider community.

Objectives

1. Deploy a LoRaWAN-based bus tracking system to track the location and arrival times of campus buses.
2. Develop and test LoRaWAN-enabled sensors for use in the bus tracking system that can transmit data over long distances with low power consumption.
3. Implement a real-time tracking dashboard accessible to students and staff that will display accurate bus arrival times and locations.
4. Integrate the LoRaWAN-based bus tracking system into the existing transportation management system to optimize routes, enhance scheduling, and improve overall efficiency.
5. Test and optimize the performance of the LoRaWAN network to ensure reliable and consistent connectivity across the entire campus.
6. Train transportation personnel and other relevant stakeholders on how to use the LoRaWAN-based bus tracking system to maximize its effectiveness.

7. Conduct user feedback sessions to evaluate the effectiveness and usability of the new system and make necessary adjustments.
8. Gather and analyze data collected by the LoRaWAN-based bus tracking system to identify areas for improvement and to inform future decision-making related to campus transportation.

Organization of the Rest of the Study

This research comprises five chapters, numbered from 1 to 5. The second chapter will focus on conducting a literature review, delving into previous projects and their methodologies. It will encompass an analysis of existing literature, the theoretical framework underpinning the study, its contributions to current knowledge, and the rationale for the investigation. Chapter three will provide an overview of the research methodology, outlining how the system will be constructed. Chapter 4 will comprehensively examine the research results and include discussions regarding the project's development. Finally, Chapter 5 will summarize the work, draw conclusions, and make recommendations.

CHAPTER TWO

LITERATURE REVIEW

Introduction

In recent years, university campuses have become increasingly complex and sprawling, making it difficult for students, faculty, and staff to navigate and efficiently utilize campus transportation services. To address this issue, researchers have proposed the use of LoRaWAN-based bus tracking systems as a means of improving the efficiency and effectiveness of campus transportation. LoRaWAN is a low-power, wide-area network technology that enables cost-effective and reliable communication over long distances, making it an ideal solution for campus-wide vehicle tracking.

The purpose of this literature review is to provide a comprehensive analysis and synthesis of the existing research on LoRaWAN-based bus tracking systems for university campuses. By critically examining the current literature, this review aims to identify key design considerations, implementation challenges, and benefits associated with such systems. Additionally, this review will explore the potential impact of LoRaWAN-based bus tracking systems on campus transportation services, including their ability to improve route optimization, reduce wait times, and enhance overall user experience. Ultimately, this review will help to inform future research and guide the implementation of effective LoRaWAN-based bus tracking systems for university campuses.

Terminologies Used:

Global Positioning System (GPS)

The Global Positioning System (GPS) is a satellite-based navigation system that allows users to determine their precise geographic location and track their movement anywhere on Earth. It operates through a network of orbiting satellites, ground control stations, and GPS receivers

LoRaWAN

LoRaWAN, which stands for Long Range Wide Area Network, is a wireless communication technology designed for low-power, long-range communication between Internet of Things (IoT) devices and a central network. It is part of the broader family of technologies under the LoRa (Low-Power Wide-Area Network) umbrella.

GSM/GPRS

GSM (Global System for Mobile Communications) and GPRS (General Packet Radio Service) are two related mobile communication technologies that were widely used for voice and data transmission in mobile networks

Related Work

(James, 2017), in their "*Efficient, real-time tracking of public transport, using LoRaWAN and RF transceivers*" proposes an innovative alternative to

traditional public transport tracking systems, shifting from the conventional use of GPS to a LoRa-based wireless transmission model. The authors seek to address the limitations and costs associated with GPS-based systems while providing an efficient and cost-effective solution.

The proposed model involves equipping buses with RF transmitters that continuously emit data regarding their identity. This information is transmitted via LoRa wireless communication to RF receivers placed at bus stops. The RF receivers detect the buses when they are within range and instantly relay this information to the base station using LoRa technology. This approach eliminates the need for individual GPS units on each bus, reducing implementation costs and power consumption.

At the base station, a LoRa receiver collects transit information from all bus stops within its range. This information can be modified as necessary and stored in a database for further processing and analysis. By centralizing the data collection and storage process, the system streamlines the tracking and monitoring of public transport, providing real-time information on bus locations and movements. The authors highlight the cost-effectiveness and scalability of the proposed system. The prototype developed for this research demonstrated a significant reduction in costs compared to traditional tracking systems, requiring only one-seventh of the expenses. Moreover, the system consumes less power, contributing to its sustainability and efficiency. The reduced costs and power consumption make the system highly scalable, as the expenses do not significantly increase with the number of buses utilized. The LoRa-based alternative presents several advantages

for public transport tracking systems. The reliance on LoRa wireless transmission eliminates the need for complex GPS infrastructure, reducing costs and power requirements. The real-time communication between buses and bus stops enables accurate and up-to-date tracking of bus movements, improving overall efficiency and user experience.

In conclusion, this research introduces a promising alternative to conventional public transport tracking systems, replacing GPS with a LoRa-based wireless transmission model. The proposed system leverages RF transmitters on buses, LoRa communication between bus stops and the base station, and a centralized database for efficient tracking and monitoring. The prototype's low costs and power consumption indicate the potential for scalability and widespread implementation. This study contributes to the development of cost-effective and sustainable public transport tracking solutions, enhancing overall transportation efficiency and user satisfaction.

(Hattarge, 2018), in their “*LoRaWAN based GPS tracking of city-buses for smart public transport system*” proposed a wireless tracking system based on LoRaWAN for efficient management of smart city buses. The authors highlight the pressing need to address air pollution caused by vehicular emissions, emphasizing the promotion of public transport as a viable solution. By incorporating real-time tracking of city buses, the system aims to enhance accessibility and convenience for commuters. One of the key challenges in deploying traditional GPS trackers for bus tracking is the high maintenance cost associated with GSM/GPRS modules. To overcome this hurdle, the authors advocate the use of LoRaWAN technology, which

offers a cost-effective alternative. LoRaWAN's long-range communication capability and low-power requirements make it suitable for tracking city buses over large areas with reduced maintenance costs. The proposed system consists of an end-to-end architecture that includes a custom gateway, server, and cloud database. This architecture provides more flexibility in designing network parameters compared to commonly used third-party gateways or server applications. The custom gateway acts as a bridge between the LoRaWAN network and the server, facilitating secure and reliable data transmission. Data from the buses, including their real-time location information, is transmitted over the LoRaWAN network and stored in a cloud database. To provide a user-friendly interface, an Android application is developed that allows users to visualize the current location of the buses. This feature enables commuters to plan their journeys more efficiently, reducing waiting times and improving overall transportation experience.

The paper's main contribution lies in the development of a custom gateway and server, which provides enhanced flexibility in network parameter design. By customizing these components, the system can be tailored to specific requirements and optimized for efficient bus tracking. The authors emphasize that this approach leads to reduced maintenance costs and increased scalability for smart city bus management systems. In conclusion, the proposed wireless tracking system based on LoRaWAN offers a promising solution for smart city bus management. By leveraging the advantages of LoRaWAN technology, such as cost-effectiveness and long-range communication, the system can effectively track city buses and enhance their accessibility. The development of a custom gateway and server adds flexibility

and scalability to the system, allowing for efficient network parameter design. Overall, this research contributes to the advancement of smart transportation systems and promotes the adoption of public transport as a means to mitigate air pollution caused by vehicular emissions.

(Manuel, 2022), in their “*Smart Campus IoT real-time bus tracking system and web app using LoRaWAN*” presented an innovative approach to enhance the mobility services and time management of university students through the real-time tracking of university buses using LoRaWAN technology. By proposing an alternative to traditional GPRS/GSM communication, the authors aim to improve the efficiency and accuracy of bus tracking systems. The proposed solution included the design and implementation of a prototype, consisting of an Arduino UNO development board, which serves as the control and instruction unit. LoRaWAN IoT technology is employed through the RAK 811 development board, connected to the Arduino UNO as a shield, enabling wireless communication. A GPS module, specifically the Neo6M, is utilized to obtain real-time location data of the buses.

To ensure the prototype's autonomy, a battery was integrated, along with additional components for protection and indicators for charging. These features contribute to the reliability and practicality of the tracking system, enabling seamless operation even in challenging environments. In parallel to the hardware development, a web application is implemented to provide users with a convenient interface for accessing real-time bus location information. The front end of the web app is built using Vue.js, a popular JavaScript framework known for its versatility and

performance. For the backend, Firebase, a cloud-based platform, is utilized to handle data storage, retrieval, and real-time updates.

The proposed solution offers several benefits for university bus tracking systems. By leveraging LoRaWAN technology, the system ensures long-range and low-power communication, enabling efficient and cost-effective tracking capabilities. Real-time bus location data enhances students' time organization, allowing them to plan their journeys more effectively and minimize waiting times. The integration of the prototype and web app provides users with an intuitive and accessible means of viewing the bus locations. The Vue.js frontend offers a responsive and interactive user experience, while Firebase facilitates efficient data management and synchronization between the prototype and the web app.

Overall, this research presents a practical and innovative solution for real-time tracking of university buses using LoRaWAN technology. By incorporating the Arduino UNO development board, LoRaWAN IoT technology, and a GPS module, the prototype demonstrates the feasibility and effectiveness of the proposed system. The implementation of the web app further enhances the user experience and accessibility of the tracking system. This study contributes to the advancement of smart transportation solutions in the university setting, improving mobility services and students' time management.

(Arian, 2022) in their study titled *“Performance Evaluation of LoRaWAN for Smart Shuttle Bus System Support in Campus Area”* focused on evaluating the performance of LoRaWAN technology for supporting a smart transportation system, specifically a campus shuttle bus system. The authors highlight the

importance of improving the accessibility of public transportation and address the challenges posed by high operational costs and power consumption in existing tracking system technologies.

The proposed solution involved the utilization of LoRaWAN, an IoT-based wireless communication technology. The study aimed to assess the performance of LoRaWAN by conducting tests using different data rate configurations (DR0 - DR5). To measure performance parameters, LoRaWAN-based GPS trackers are placed on the roofs of shuttle buses. Several metrics, including Signal to Noise Ratio (SNR), Received Signal Strength Indication (RSSI), and packet loss, are measured during the tests.

The test results were analyzed, and the performance parameter values were compared. Additionally, heatmaps based on SNR and RSSI values are created to visualize the performance along the shuttle bus route. The analysis indicated that there is a slight difference in the performance parameters (SNR, RSSI, and packet loss) when using different data rate configurations. However, overall, the performance of LoRaWAN remained reliable across all tested data rates.

Based on the performance evaluation, the average SNR value was above 0dB, indicating satisfactory signal quality. The average value of RSSI is above -100dBm , suggesting good signal strength. The maximum packet loss value is under 3% for each data rate configuration, implying reliable communication. These results demonstrated that LoRaWAN is capable of providing robust communication for the smart transportation system under evaluation.

The findings of this study contributed to the understanding of LoRaWAN's performance in the context of smart transportation systems. The analysis indicated that LoRaWAN is a viable technology choice, offering reliable communication with acceptable performance parameters. The study provided valuable insights for the deployment of LoRaWAN-based tracking systems, helping to improve the accessibility and efficiency of public transportation.

In conclusion, this research evaluates the performance of LoRaWAN for supporting a smart transportation system in a campus setting. The study demonstrates that LoRaWAN provides reliable communication, as indicated by satisfactory SNR, RSSI, and packet loss values. The findings contribute to the body of knowledge surrounding IoT-based tracking systems, paving the way for the implementation of cost-effective and efficient solutions in the field of smart transportation.

In their study titled “*Compression Method of Position Information for IoT-based Bus Location System Using LoRaWAN*”, (Boshita, 2018) focused on achieving an IoT-based bus location system with low operational costs by leveraging LoRaWAN technology. However, in the specific context of Japan, LoRaWAN operated in the mode with the longest communication distance had a limitation of transmitting only 11 bytes of data at once. To overcome this constraint and efficiently transmit time and bus travel position information, the authors proposed a location information compression method.

Traditionally, transmitting time and location information acquired from GPS using the general method required 280 bits of data. In contrast, the proposed compression method significantly reduced the data size to only 49 bits. By employing this compression technique, the authors aimed to optimize the use of LoRaWAN in the bus tracking system, ensuring efficient utilization of the limited data capacity.

The proposed compression method targeted the efficient transmission of time and location information within the 11-byte constraint. By compressing the data, the system could transmit the necessary information while minimizing the impact on the communication capacity of LoRaWAN. This approach allowed for more efficient utilization of the available data bandwidth and improved the overall performance of the bus tracking system.

In conclusion, this research addressed the operational cost limitations of an IoT-based bus location system by utilizing LoRaWAN technology. The proposed location information compression method aimed to optimize data transmission within the constraints of LoRaWAN communication, particularly in Japan. By compressing the time and bus travel position information, the authors achieved efficient utilization of the limited data capacity, significantly reducing the required data size from 280 bits to 49 bits. This study contributes to the development of cost-effective and resource-efficient bus tracking systems, enhancing the overall functionality and performance of IoT-based transportation solutions.

(Shree, 2019), in their “*Real Time Bus Tracking and Location Updating System*” addressed the need for automation and real-time tracking in the public transportation system, recognizing its significant role in various aspects of life and

economic development. The manual operation of major services in the system results in approximate data and limits ease of access for the public. To overcome these challenges, the project focused on automating the services by implementing a real-time tracking system for public transport buses.

The proposed system employed RFID technology by placing RFID tags on the buses and installing RFID readers at each bus stop. The RFID readers capture the bus information, such as bus identity, and transmit it to the central controller, which is an Arduino board. This central controller serves as the brain of the system, processing the data received from RFID readers.

To ensure continuous monitoring, a GSM module was utilized to send tracking messages to authorized individuals. This feature allowed for real-time updates and enabled efficient management of the public transportation system. Additionally, GPS technology was integrated to acquire accurate bus locations, enhancing the precision of the tracking system.

To provide user-friendly access to bus tracking information, the system utilized IoT technology. Users receive notifications on their mobile devices, delivering real-time bus tracking details. This approach enhanced accessibility and convenience for passengers, improving their overall travel experience.

The data collected by the Arduino board from RFID readers is processed and transmitted to the cloud. The cloud serves as the interface between the user and the system, facilitating seamless interaction and retrieval of real-time bus tracking

information. This integration of cloud technology enables efficient data management and enhances the responsiveness of the system.

In conclusion, this research project addresses the automation and real-time tracking challenges faced by the public transportation system. By implementing RFID technology, GSM modules, GPS, and IoT, the system offers a comprehensive solution for tracking and monitoring public transport buses. The integration of Arduino as the central controller and the cloud as the interface enhances data processing, accessibility, and overall system performance. The proposed system improves the accuracy, efficiency, and user experience of the public transportation system, contributing to the advancement of smart transportation solutions.

(Durón, 2019) in their study titled “*Mobile Positioning for IoT-based Bus Location System Using LoRaWAN*” focused on the implementation of a Mobile LoRaWAN Gateway, exploring the benefits of leveraging the LoRaWAN communication protocol and system architecture in combination with IoT for various applications. The project aimed to provide an alternative solution for scenarios where a mobile gateway is required or where resources are limited for deploying a conventional LoRaWAN gateway.

By utilizing LoRaWAN, the proposed Mobile LoRaWAN Gateway offers distinct advantages over alternative solutions that do not incorporate this communication protocol. The article highlights the importance of testing the capabilities of LoRaWAN and demonstrating its superiority in specific applications.

The implementation of a Mobile LoRaWAN Gateway offers flexibility and mobility, allowing for the deployment of LoRaWAN infrastructure in various settings. This solution caters to applications that require a portable or temporary gateway, enabling connectivity in remote areas, mobile environments, or situations where conventional gateways are impractical.

The article emphasizes the significance of leveraging IoT and LoRaWAN to address the limitations of traditional gateway solutions. By integrating IoT principles, the Mobile LoRaWAN Gateway enhances connectivity and data transfer capabilities, providing a robust and efficient solution for IoT applications.

The project's main objective is to validate and showcase the advantages of LoRaWAN over alternative solutions that do not utilize this communication protocol. By demonstrating the capabilities of the Mobile LoRaWAN Gateway, the article aims to promote the adoption and utilization of LoRaWAN in diverse applications, highlighting its benefits in terms of flexibility, mobility, and resource optimization.

In conclusion, this article presents the implementation of a Mobile LoRaWAN Gateway, emphasizing the benefits of the LoRaWAN communication protocol and system architecture in conjunction with IoT for various applications. The project aims to test the capabilities of LoRaWAN from a unique perspective and demonstrate its advantages over similar solutions that do not incorporate this protocol. The proposed Mobile LoRaWAN Gateway offers flexibility, mobility, and resource optimization, making it a valuable solution for applications that require a

portable or temporary gateway. The article contributes to the advancement of LoRaWAN technology and encourages its adoption in IoT applications.

In their “*Deployment of a LoRaWAN network and evaluation of tracking devices in the context of smart cities*” (de Camargo, 2021), demonstrated the collaboration between the Federal University of Technology - Parana (UTFPR) and the Toledo Municipality focuses on implementing smart city concepts, with one application being the real-time tracking of recyclable garbage collector trucks using LoRaWAN technology. Fleet vehicle tracking is a crucial component of smart cities, and LoRaWAN is recognized for its suitability in this context due to its open frequency range, long-distance coverage capabilities, low power consumption, and cost-effective equipment.

However, (de Camargo, 2021) stated that the performance and coverage of LoRaWAN are influenced by various factors, including the environment and configuration parameters. To address these considerations, the paper presents experimental investigations that evaluate four LoRaWAN tracking devices. These devices include both off-the-shelf options and devices that were assembled and programmed specifically for the study.

The behavior of the tracking devices is analyzed as they traverse representative urban areas, covering a total area of 10.71 km². The evaluation focuses on several aspects, including received signal strength indication (RSSI), signal-to-noise ratio (SNR), packet delivery ratio (PDR), and spreading factor (SF) for the received

geographic coordinates. By assessing these parameters, the effectiveness and reliability of the tracking devices can be determined.

Furthermore, the two most efficient tracking devices are subjected to additional analysis in a specific stretch covering a distance of 3.5 km. The devices are tested at various speeds ranging from 0 to 30 km/h, 0 to 50 km/h, and 0 to 100 km/h. This evaluation provides insights into the devices' performance under different speed conditions, which is essential for real-time tracking applications.

In addition to evaluating the tracking devices, the paper also presents the development and evaluation of the LoRaWAN network for the smart city application. The coverage planning throughout the city is a key consideration, as it directly affects the quality and reliability of the tracking system. The results of the network development and evaluation are included to provide a comprehensive assessment of the overall system's performance.

In conclusion, this study focuses on evaluating LoRaWAN tracking devices for real-time vehicle tracking in smart cities. The experimental investigations analyze the behavior of the devices in representative urban areas and assess their performance under different speed conditions. The evaluation includes various parameters such as RSSI, SNR, PDR, and SF. Additionally, the paper presents the development and evaluation of the LoRaWAN network, highlighting the importance of coverage planning for ensuring reliable and effective tracking. The findings of this study contribute to the understanding and advancement of LoRaWAN technology in the context of smart city applications.

CHAPTER THREE

RESEARCH METHODOLOGY

Introduction

This chapter aims to provide you with an extensive overview of the approaches employed to complete this project and render it functional. It will encompass an examination of the project's structure, its practical applications, the assembly process, configuration details, front-end programming, and communication aspects.

Requirement Gathering and Analysis

The phase of requirement gathering and analysis holds paramount importance within the software development life cycle. It entails the identification of stakeholders' needs and limitations, followed by the translation of these findings into a set of requirements. These requirements serve as a guiding framework for the subsequent stages of the development process.

The stakeholders of our project comprise of the University Administration since they will be responsible for ensuring that the system meets the needs of the university community and doesn't infringe on any rules and regulations. The transportation department and Bus drivers are also responsible for managing the buses and ensuring that the end nodes that are placed on the buses are not tampered with or stolen. The students would be the primary users of the system as they would

use it to track the location of the buses and plan their commute accordingly. Finally, we have faculty and staff would also benefit from the system as it will also help plan their commutes.

Functional Requirements

1. The system should be able to be accessed remotely
2. The system should provide the accurate location of each bus at all times
3. The system should update the location of the bus every four (4) seconds
4. The system should be able to transmit data locally and not over the internet

Non-functional Requirements

1. The system should have excellent user experience
2. The system should be portable
3. The system should be safe and easy to use
4. The system should work fully without lagging

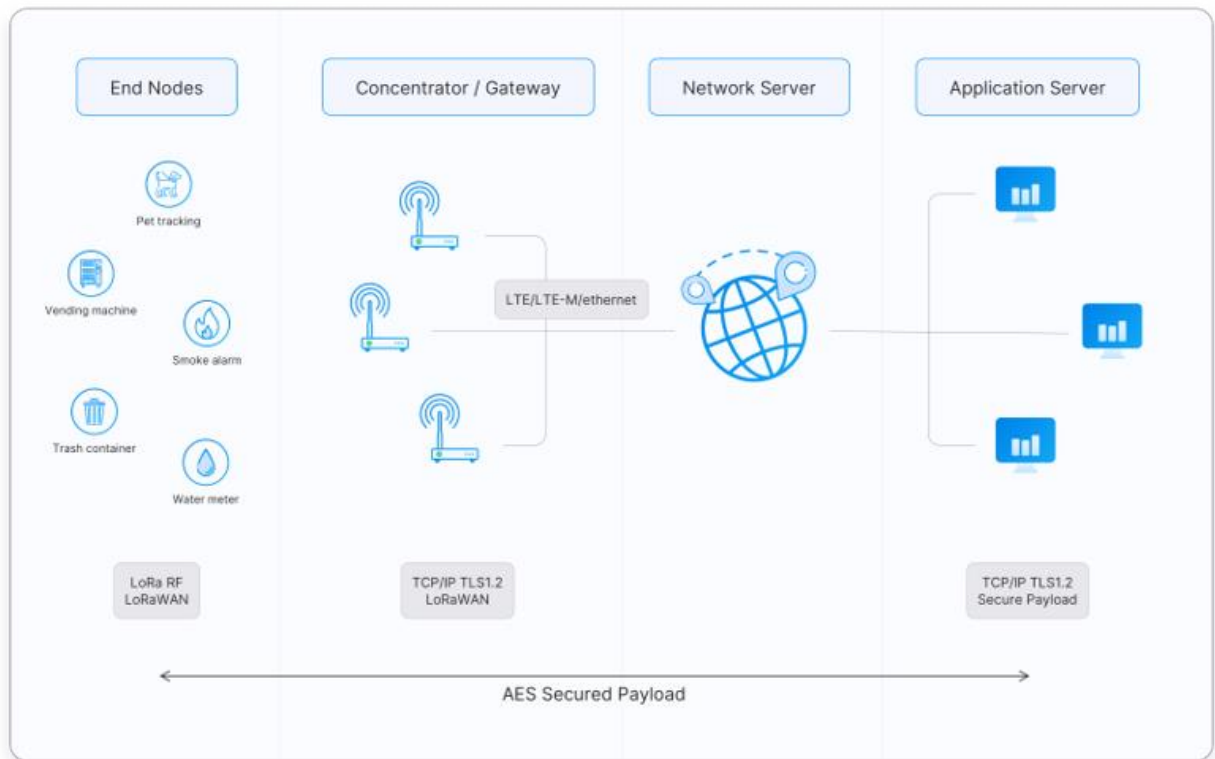
System Design

The system design is a detailed description of the system requirements, operating environment, architecture. It provides a comprehensive overview of the system's architecture and how it is expected to function. It also assists in designing the overall system architecture as well as describing hardware and system

requirements needed to produce the results. Hardware designs, use case diagrams, sequence diagrams, transition diagrams, and prototypes are among the deliverables created at this stage.

General System Design

Figure 1: *System Design*



End Devices (Nodes):

End devices are the physical IoT devices that collect data or perform actions. These can be sensors, actuators, or other devices. They are typically battery-powered and designed for low power consumption.

Gateway:

Gateways are the intermediaries between the end devices and the network server. They receive data from the end devices and forward it to the network server.

Gateways can cover a relatively large geographic area and are often connected to the internet through wired or wireless connections.

Network Server:

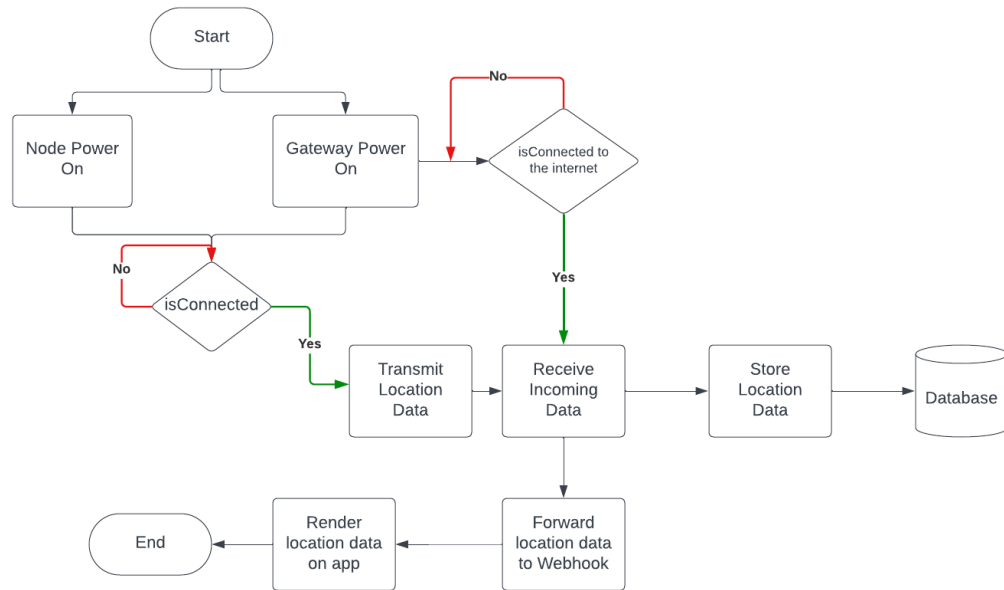
The network server manages the network, controls the gateways, and routes data from end devices to the appropriate application server. It is responsible for security, device management, and handling network-related functions. Data received from the gateways is decrypted and validated by the network server.

Application Server:

Application servers receive the data from the network server and process it for the specific application. They can trigger actions, store data, and provide a user interface for data visualization. Application servers are responsible for processing and interpreting the raw sensor data.

Flowchart

Figure 2: *Flowchart*



Use Case Diagram

A use case diagram is a graphical representation that illustrates how a system interacts with external entities, known as actors, to achieve specific goals or functionalities. It is one of the Unified Modeling Language (UML) diagrams commonly used in software engineering and systems analysis to model and visualize the interactions between different components of a system

Figure 3: Use Case Diagram



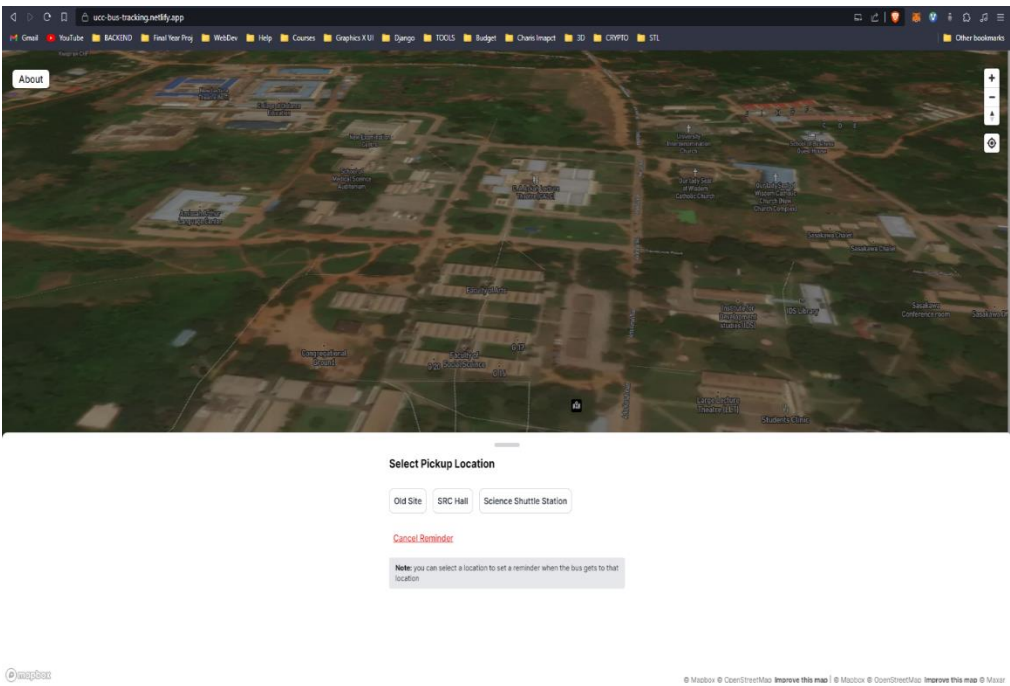
Web interface Design

Web interface design refers to the process of creating the visual and interactive elements of a website or web application that users interact with. It encompasses the layout, appearance, and functionality of the user interface, with the goal of providing an intuitive, engaging, and efficient user experience.

Figure 4: Web Interface Main



Figure 5: Web Interface Pickup section



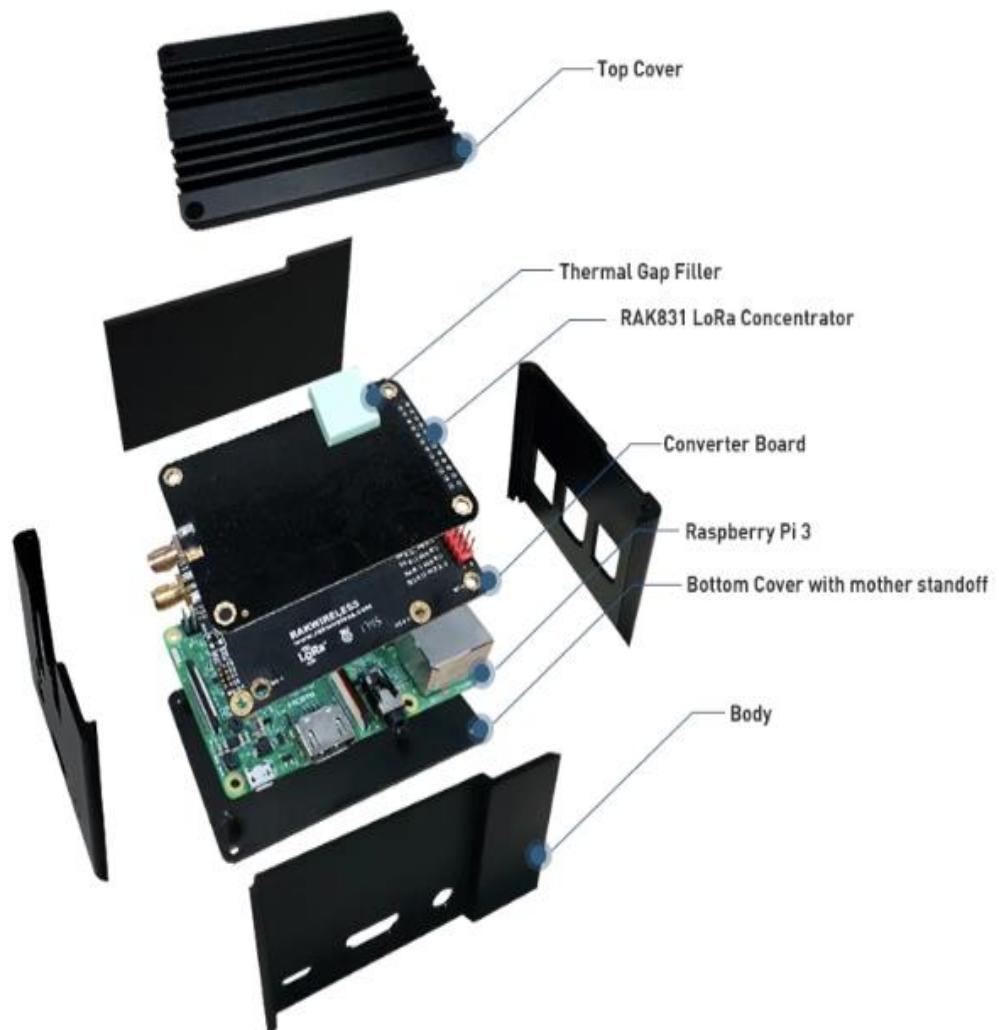
Language Justification

C++ for Arduino and JavaScript for the web app.

C++ is chosen for Arduino programming due to its efficiency, object-oriented capabilities, extensive libraries, and compatibility with microcontrollers. It offers flexibility for both low-level hardware access and high-level abstractions, while a vibrant community and resources support its use in creating complex and optimized projects.

JavaScript was the preferred language for implementing the web app due to the speed and efficiency of the React framework. React is a JavaScript framework for crafting interactive user interfaces. It employs reusable components to build complex UIs efficiently. Using a virtual DOM, it updates only necessary parts of the actual DOM, enhancing performance and supporting one-way data flow.

Figure 6: *Breakdown of the gateway*



Source: (Pi Supply, 2019)

Building the Hardware

The steps taken to build the hardware portion of this project are discussed.

Devices/sensors and usage:

The following devices and sensors are used to develop the Bus Tracking System.

Raspberry Pi.

This is the microcontroller being used for the project because of its compatibility with the Lora Module and the concentrator and also the RAK831 pilot gateway requires us to use it for as it possesses the ability to run an Operating System and other systems such as a database manager and an app server.

Figure 7: *Raspberry Pi 3b*

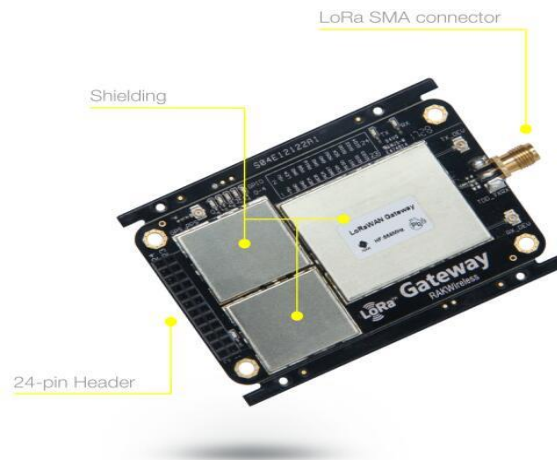


Source: (Reichelt, 2018)

RAK831 Concentrator module:

The RAK831 is a LoRaWAN gateway concentrator module that can act as the core of a full 8-channel LoRaWAN gateway 12. It is designed to work with a Raspberry Pi and provides the possibility to enable robust communication between an LPWAN gateway and a huge amount of LPWAN end-nodes spread over a wide area 1. The RAK831 is based on the Semtech SX1301 chip and supports 8 channels LoRaWAN stack 1.0.2 1. It has an Rx sensitivity down to -139 dBm (@293bps) and Tx power up to 23 dBm 2. The module supports full band coverage for 433MHz, 470MHz, 865MHz, 868MHz, 915MHz, 920MHz, and 923MHz. In our Case we are using the 868MHz module in compliance with our geographic location.

Figure 4: *RAK831 Concentrator module*



Source: (rakwireless, 2017)

TTGO Module Node:

The TTGO T-Beam is a development board that combines an ESP32 with a LoRa radio and onboard GPS 1. It provides an ideal development environment for long-range RF (LoRa) and GPS 2. The board is built around a dual-core ESP32 chip,

with 4MB of SPI flash onboard, providing both Wi-Fi and Bluetooth LE via a “3D antenna” on the PCB

Figure 8: *LoRaTTGO Node*



Source: (tinytronics, 2020)

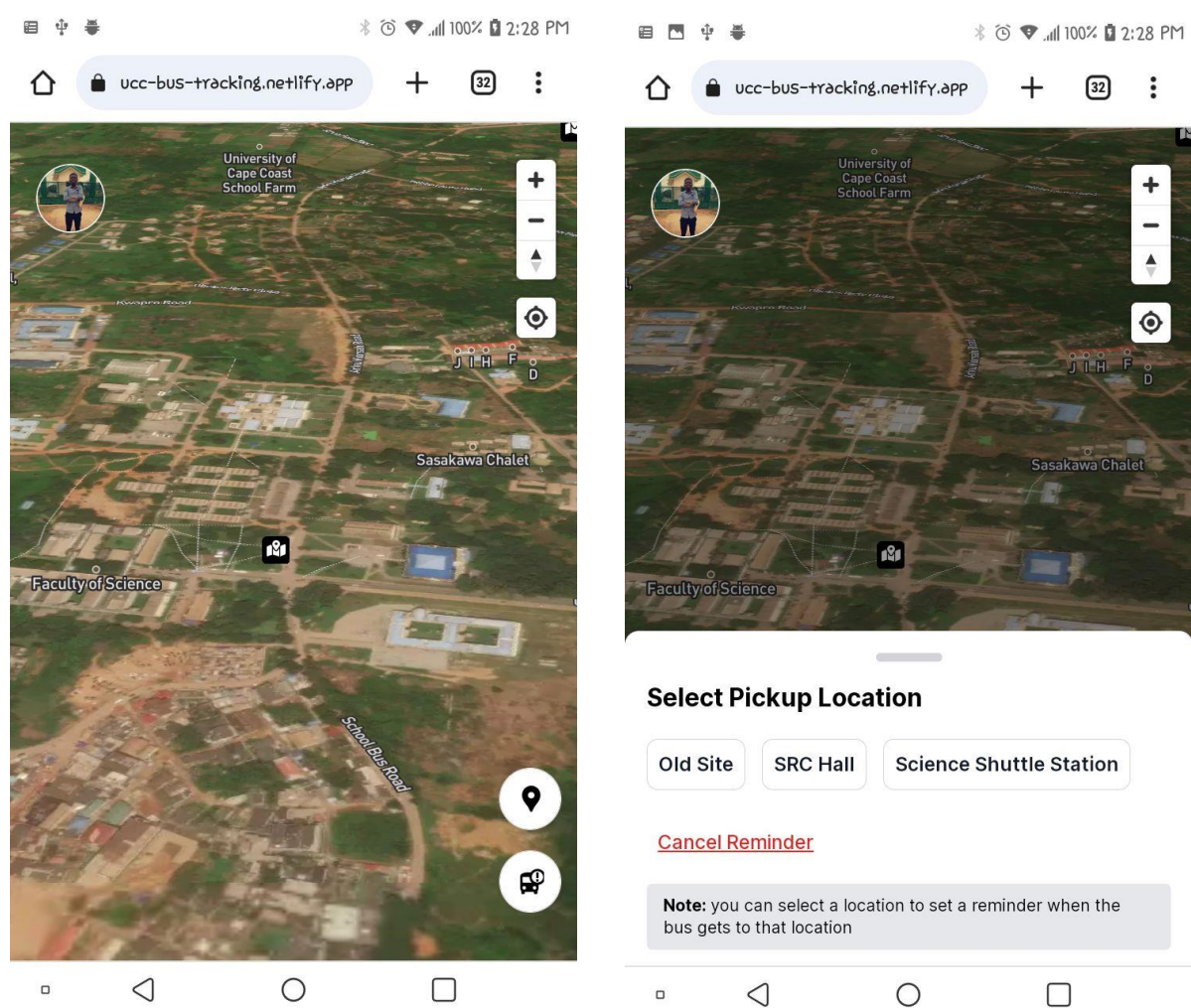
Figure 9: 868MHz LoRa Antenna 5.8dBi Outdoor Fiberglass

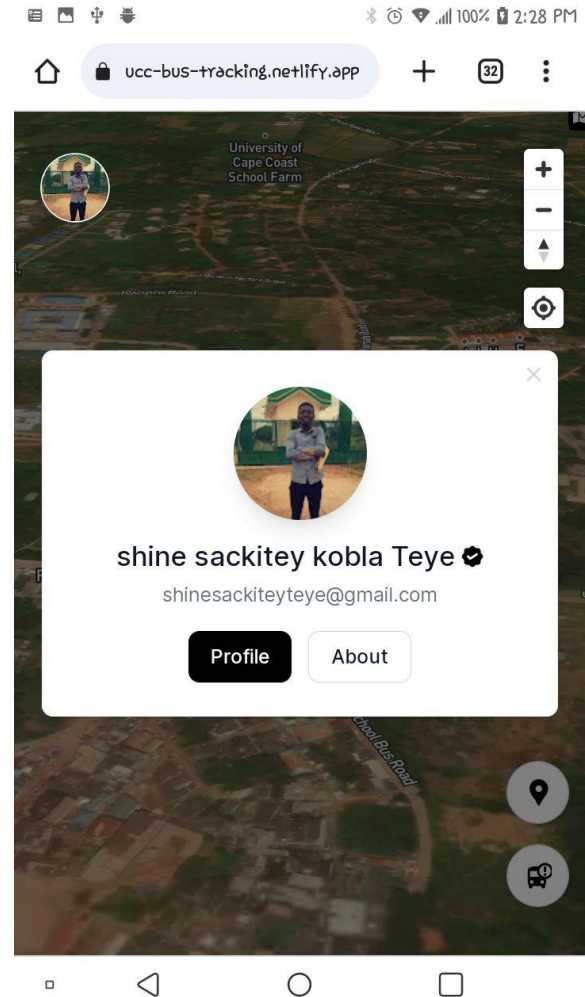
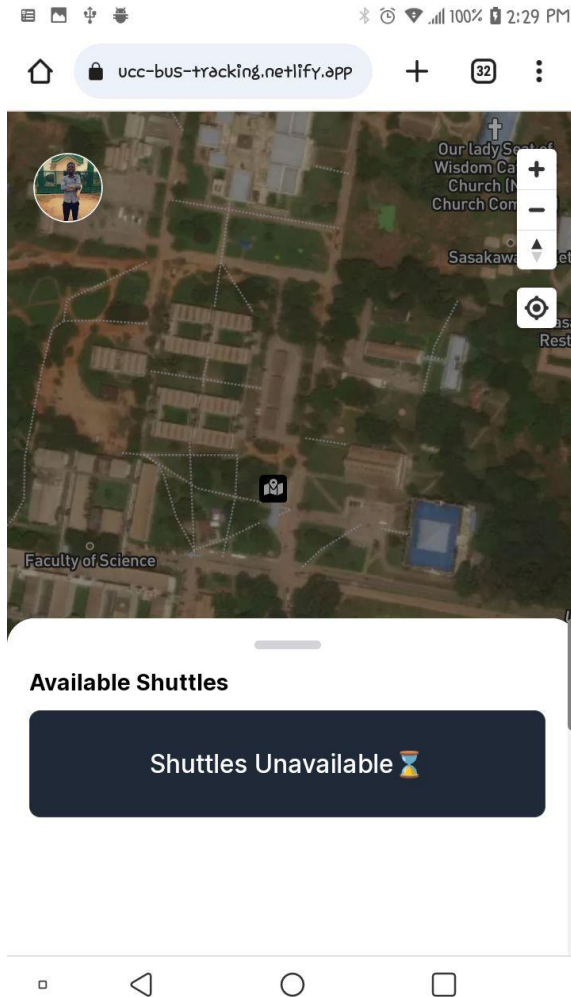


Source: <https://store.rakwireless.com/products/5-8dbi-fiber-glass-antenna>

Building the Front End

The user can view real-time bus locations on the front end.





Building the parts - Hardware

Each container for the bus nodes were designed and 3D printed.

Figure 10: *3D printed cases for Nodes*



Integration and Testing

Tests were performed device by device to make sure they were functioning perfectly. This was done to ensure that all errors would be resolved before they are put together and a full test is run.

To detect flaws and failure, the hardware and the web app were combined, and tests were performed several times to ensure that they worked perfectly.

Final tests involved mounting the devices on the buses and allowing them to move around to really evaluate the range at which they can still maintain communication. There were multiple tests in this phase also since the gateway had

to be relocated several times to ensure that the optimal point for successful long-range communication was achieved.

System Deployment

At this point the LoRaWAN bus tracking system will cover network setup with gateways, device deployment on buses, data transmission security, central server configuration, user interface explanation, testing, scalability considerations, maintenance protocols, integration with campus operations, user training, benefits, and future plans.

CHAPTER FOUR

RESULTS AND DISCUSSION

Introduction

This chapter's goal is to provide an explanation of the outcomes of our solution using the LoRaWAN technology.

Design and Implementation

Fritzing was used to visualize the connection layout when designing the systems. Figma was utilized to create the front-end prototype. For 3D printing, the sketched prototypes for the 3D cases were converted into 3D models using FreeCad. Additionally, Git/GitHub was used as a version control system throughout the development process so that we could continue to collaborate in real time.

Testing and Evaluation

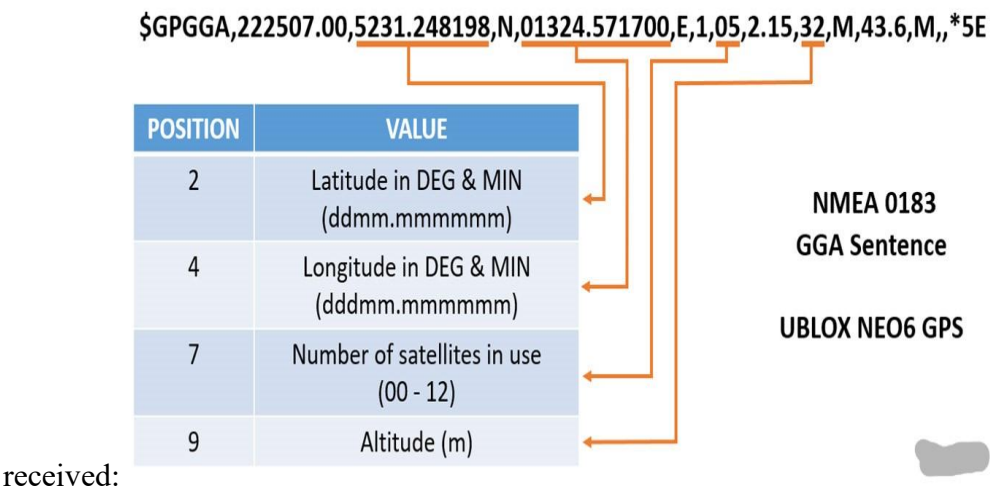
Software testing involves comparing the software to the specifications of the system, as well as code development and design. Software testing, which includes the validation and verification of the developed system, is a crucial component of software quality assurance. Software testing is important because it can help identify flaws and mistakes that were made throughout the development process. Additionally, software testing offers an unbiased, impartial picture of the product so that the owners may recognize and comprehend the dangers associated

with its implementation. Unit testing, compatibility testing, and system testing were all considered for our projects.

Unit Testing

Testing the nodes:

The GPS data from the nodes were collected, then formatted to get the actual latitude and longitude of each device. Other info such as number of satellites and altitudes was also received from this process. Here is a breakdown of GPS data



The formatted data was then successfully sent to the gateways available.

Testing the LoRaWAN gateway connection:

The LoRaWAN gateway comprises several key components, including the ChirpStack technology and the Semtech UDP Packet Forwarder which uses the

Semtech UDP protocol. The gateway's functionality was assessed through a series of tests.

Firstly, the gateway's ChirpStack technology was evaluated to ensure seamless communication with the connected devices. This involved verifying the integration of the gateway bridge, which serves as a crucial link between the LoRaWAN network server and the physical gateway.

Next, the gateway's performance was assessed in a real-world scenario. This encompassed testing its ability to receive and transmit data from the end devices, within its coverage area.

Additionally, the gateway's network connectivity and reliability were examined under different environmental conditions. This included assessing its range, signal strength, and interference susceptibility.

Overall, the LoRaWAN gateway demonstrated robust functionality and reliability in facilitating wireless communication between the connected devices and the ChirpStack infrastructure.

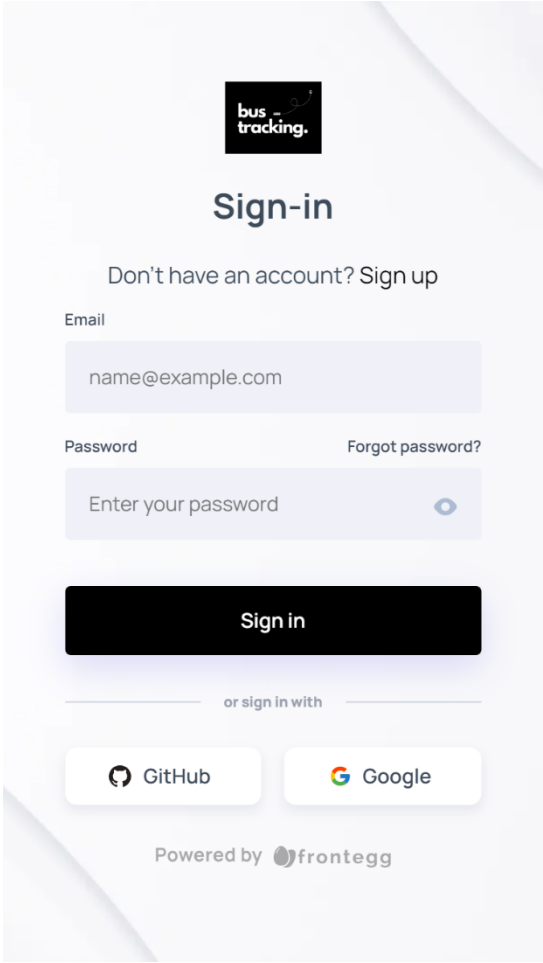
Testing the Web App:

GPS data was pushed from the network server to an application server hosted on render, a web service platform (<https://bt-server.onrender.com>). The payload received from the network server was then accessed by the web application. The

web application was tested to see if it could be installed locally as an app, this test yielded successful results.

The web app can be found at <https://ucc-bus-tracking.netlify.app>. when you open it, you first need to login as shown below:

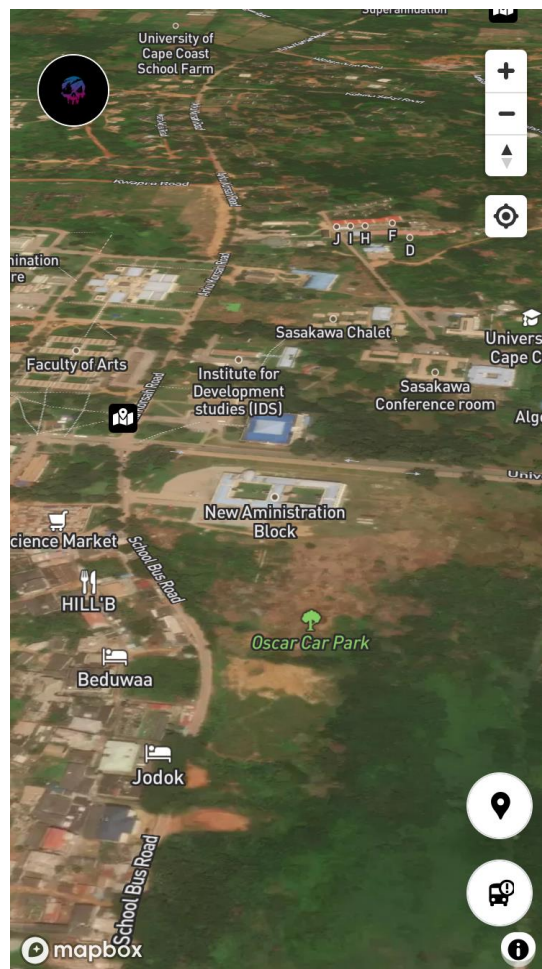
Figure 11: *Login Page*

The image shows a login page for a 'bus tracking' application. At the top, there is a logo with the text 'bus tracking.' inside a black square. Below the logo, the heading 'Sign-in' is displayed in a large, bold, dark font. Underneath the heading, there is a link that says 'Don't have an account? Sign up'. The form consists of two input fields: an 'Email' field with the placeholder text 'name@example.com' and a 'Password' field with the placeholder text 'Enter your password'. To the right of the password field is a link that says 'Forgot password?' and a toggle icon. Below the input fields is a large, black 'Sign in' button. Underneath the button, there is a horizontal line with the text 'or sign in with' in the center. Below this line are two buttons: one for 'GitHub' and one for 'Google'. At the bottom of the page, there is a footer that says 'Powered by frontegg' with the frontegg logo.

On the login form, you must enter your login details (email and password) to be able to access the platform. After a successful login, the homepage is shown. The user can see the map with available shuttles (if any). If the user has no account,

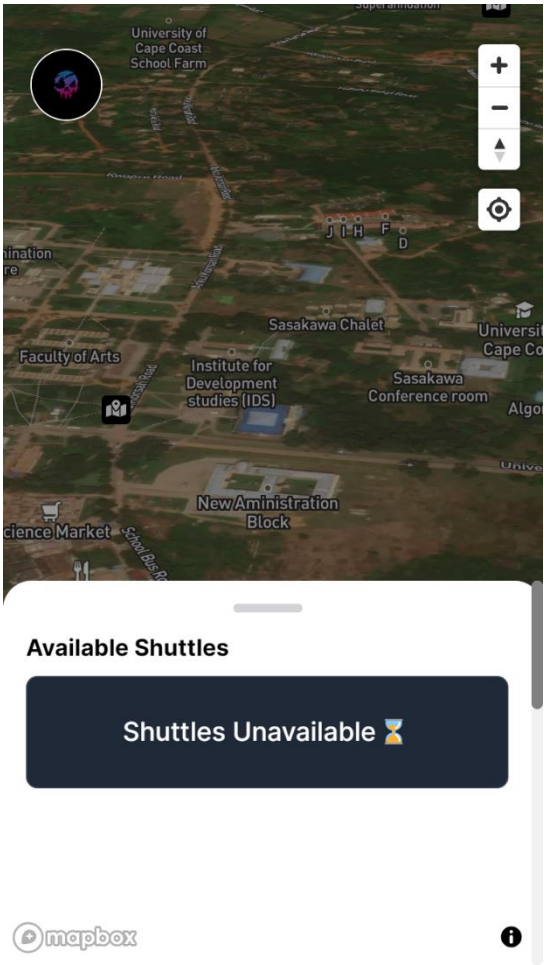
the user signs up by accessing the sign-up page OR by using the quick sign-up options available (i.e., GitHub, Google, etc.)

Figure 12: *Homepage*



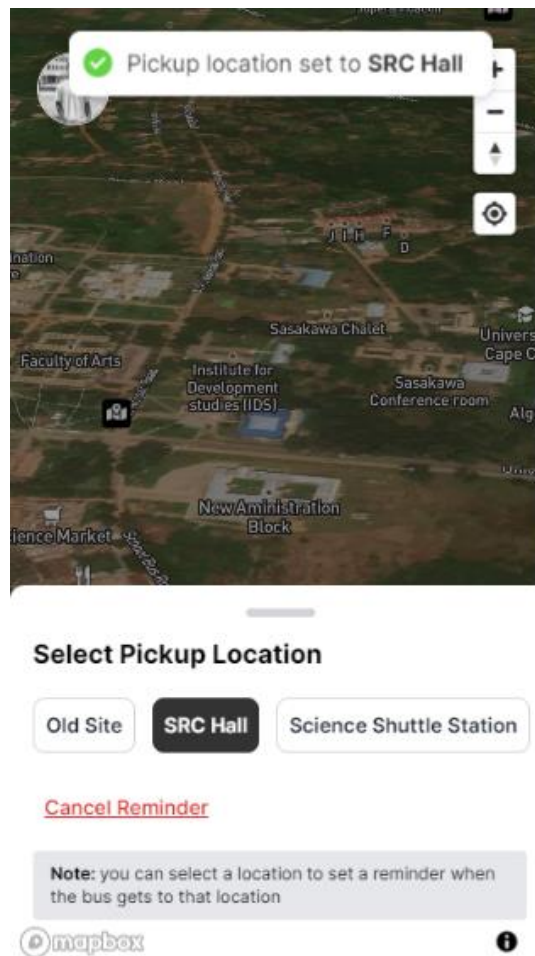
The homepage shows a map with a viewport set at UCC Campus. The page contains several buttons for Geolocation, viewing available shuttles and a button to set a reminder for when shuttles reach a certain station. Available shuttles and Shuttle stations will be seen on the map as different markers.

Figure 13: *Available Shuttles*



This page shows information about the available shuttles.

Figure 14: *Set Reminder*



This page allows the user to set a reminder for a station. When the shuttle gets to that shuttle station, a push notification is sent to the user's phone. You can also clear a reminder you've already set.


Figure 15: *Shuttle stations*



Shows info about a particular shuttle station.

Figure 16: *Profile Page*





Profile



seths10
(Developer)

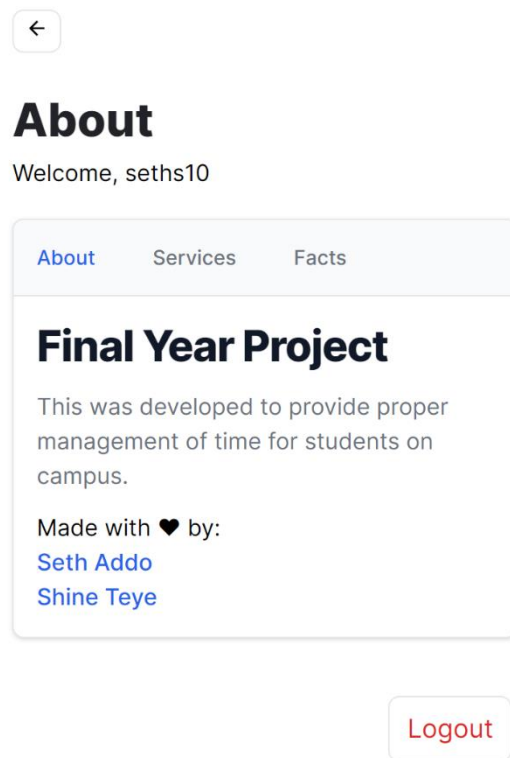
addoseth35@gmail.com

Basic Profile Information

Email	addoseth35@gmail.com	
Name	seths10	
Phone Number	0559372716	
Addresses		
Job Title	Developer	

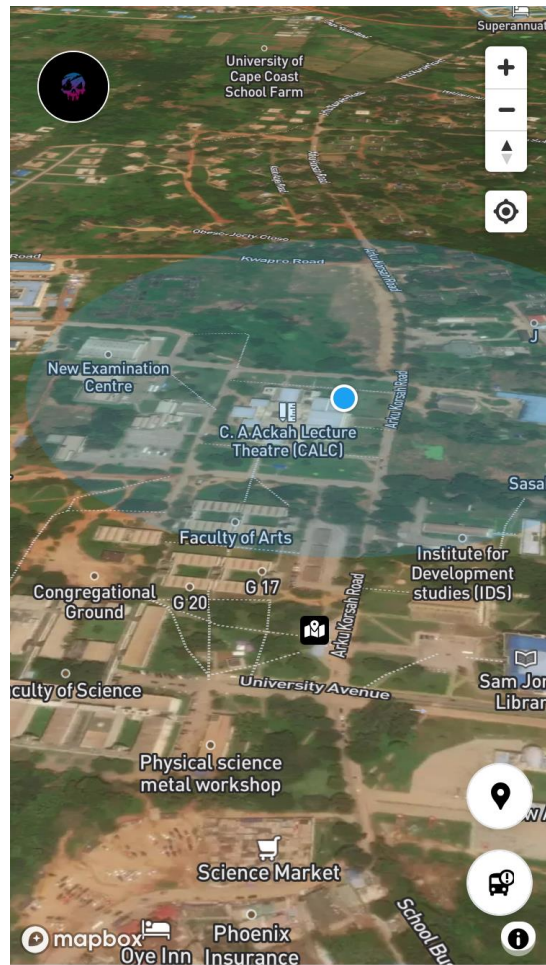
Shows info about the logged in user. The user is also able to change their profile details such as phone number, address, job title, name, and password. This page also allows the user to log out of all previous sessions.

Figure 17: *About Page*



The about page contains info such as facts about the software, creator's info, and more. Users can also log out of the software using this page.

Figure 18: *Geolocation*



The page allows the user to find his/her own location on the map.

Hardware Setup

Figure 19: *Mounted Antenna*



Figure 20: *Device (Node with mini Antenna)*



Integration Testing

The next test performed after unit tests is integration testing, which combines the hardware and software components that have undergone unit testing. A group of components are joined to provide output during an integration test.

Integration testing of the base (Sensors):

All the sensors were tested together.

System Testing

This is where the final software and hardware are tested together as a whole unit. First, the hardware (end device), which is LoRa TTGO node (with ESP32 microcontroller, GPS module installed on it), was tested to see if data is being sent. The second part, the software, was tested to see if it would get the right data. The GPS data was sent to the app server and then displayed on the web page.

Results

Economic Benefits

1. Shows available buses to help in efficient time management by students.
2. Set reminders for when a bus reaches a shuttle station thus improving productivity.

System Requirements

Based on the testing results of the system, the following system requirement were discovered and documented below:

1. The antennae on the gateway must be placed at a high point to get the maximum range possible.

Conclusion

In this chapter, a complete description of the entire system was shown. Both the hardware and the website. The user must log into his account to be able to see available shuttles on the map. Reminders can also be set for when a bus reaches a shuttle station

CHAPTER FIVE

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

Summary

The initial concept for this project was to create an application to help students locate shuttles on campus. The fundamental ideas underlying the approach are requirements gathering and analysis, design and development, testing and evaluation, and implementation. In the Requirements Gathering and Analysis stage, we did our research and found out all we needed to make the system we wanted to implement work. We came up with what we wanted the system to be able to do—functional requirements and some other non-functional requirements. Hardware prototype design, sequence diagrams, use case diagrams, transition diagrams, low fidelity, the design of the user interface, and coding were all included in the design and development stage. At this point, we have identified the programming languages and libraries needed to carry out the system's implementation. For the frontend, we decided to use ReactJS, VitePWA to power the native web application aspects of the projects, NodeJS for the application server and some Arduino libraries like MCCI and LMIC were used. In the testing and evaluation stage, we kept testing all the devices and sensors individually and then as a whole unit. Also, we put a lot of effort into developing the system, especially the software to be very basic and straightforward to use.

Conclusion

In summary, our system successfully displayed available buses on the map within the application or website. However, we encountered several challenges during implementation. Firstly, our reliance on a single gateway posed limitations. Additionally, our antennae placement, which was not at the optimal height, hindered our ability to achieve an extended range. Consequently, due to these challenges and constraints, our system falls short of providing complete coverage across the entire width of the UCC campus but still capable of reaching about 65% of the entire bus routes on campus.

Recommendations

Considering the challenges, we faced in achieving full coverage for our system on the UCC campus, we recommend several future enhancements to address these limitations:

1. **Multi-Gateway Deployment:** Expanding the system with multiple gateways strategically positioned across the campus can significantly enhance coverage. This would mitigate the range limitations encountered with a single gateway.
2. **Optimal Antenna Placement:** Reassessing and potentially relocating antennae to higher points on buildings or structures can improve signal propagation and extend the range of the system.

3. **Signal Strength Analysis:** Conduct a thorough analysis of signal strength and potential interference factors within the campus environment. This can guide decisions on gateway placement and antenna orientation.
4. **Signal Repeaters or Boosters:** Consider incorporating signal repeaters or boosters in areas with challenging coverage. These devices can amplify signals and help bridge gaps in connectivity.
5. **Regular Maintenance and Monitoring:** Establish a maintenance schedule to ensure the system remains in optimal working condition. This includes checking for any hardware malfunctions, updating software, and recalibrating components as necessary.

By implementing these recommendations, the system can overcome its current limitations and achieve more comprehensive coverage across the UCC campus. This will enhance the overall effectiveness and usability of the application or website for tracking available buses.

REFERENCES

- (2018). A survey of LoRaWAN for IoT: From technology to application. *Sensors*, 18(11), 3995.
1. 3. (2018). A survey of LoRaWAN for IoT. *From technology to application. Sensors*, 18(11), 3995.
- Arian, R. D. (2022, December). Performance Evaluation of LoRaWAN for Smart Shuttle Bus System Support in Campus Area. *2nd International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)*, pp. 47-52.
- Bankov, D. K. (2016). On the limits of LoRaWAN channel access. *International conference on engineering and telecommunication*, 10-14.
- Boshita, T. S. (2018, October). Compression method of position information for iot-based bus location system using lorawan. *Eleventh International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, pp. 1-2.
- de Camargo, E. T. (2021). Deployment of a LoRaWAN network and evaluation of tracking devices in the context of smart cities. *Journal of Internet Services and Applications*, 12(1), 1-24.
- de Carvalho Silva, J. R. (2017). LoRaWAN—A low power WAN protocol for Internet of Things: A review and opportunities. *2nd International multidisciplinary conference on computer and energy science (SpliTech), IEEE*, pp. 1-6.

- Durón, J. I. (2019, October). Mobile positioning for IoT-based bus location system using LoRaWAN. *IEEE International conference on engineering veracruz (ICEV)*, pp. Vol. 1, pp. 1-7.
- Hattarge, S. K. (2018 , December). LoRaWAN based GPS tracking of city-buses for smart public transport system. *first international conference on secure cyber computing and communication (ICSCCC)*, pp. 265-269.
- James, J. G. (2017, November). Efficient, real-time tracking of public transport, using lorawan and rf transceivers. *TENCON 2017-2017 IEEE Region 10 Conference*, pp. 2258-2261.
- Manuel, A. V. (2022, September). FIEE Smart Campus IoT real-time bus tracking system and web app using LoRaWAN. *IEEE International Smart Cities Conference (ISC2)*, pp. 1-7.
- Marais, J. M.-M. (2019, November). A review of LoRaWAN simulators: Design requirements and limitations. *international multidisciplinary information technology and engineering conference (IMITEC)*, pp. pp. 1-6.
- San Cheong, P. B. (2017, November). Comparison of LoRaWAN classes and their power consumption. *IEEE symposium on communications and vehicular technology (SCVT)*, pp. pp. 1-6.
- Shree, A. D. (2019, March). Real time bus tracking and location updation system. *5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, pp. 242-245.

- Tomasin, S. Z. (2017, March). Security analysis of lorawan join procedure for internet of things networks. *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. pp. 1-6.
- Yang, X. K. (2018, April). Security vulnerabilities in LoRaWAN. *IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pp. 129-140.

APPENDIX

Code for End Device

```
#include <lmic.h>

#include <hal/hal.h>

#include <SPI.h>

#define SERIAL1_RX 34 // GPS_TX -> 34

#define SERIAL1_TX 12 // 12 -> GPS_RX

String read_sentence;

uint8_t tx_payload[11];


boolean join_success = false;


//

// For normal use, we require that you edit the sketch to replace FILLMEIN

// with values assigned by the TTN console. However, for regression tests,

// we want to be able to compile these scripts. The regression tests define

// COMPILE_REGRESSION_TEST, and in that case we define FILLMEIN to a

non-

// working but innocuous value.
```

```

//

#ifdef COMPILE_REGRESSION_TEST

#define FILLMEIN 0

#else

#warning "You must replace the values marked FILLMEIN with real values from
the TTN control panel!"

#define FILLMEIN (#dont edit this, edit the lines that use FILLMEIN)

#endif


// This EUI must be in little-endian format, so least-significant-byte

// first. When copying an EUI from ttncctl output, this means to reverse

// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,

// 0x70.

static const u1_t PROGMEM APPEUI[8] = { APPEUI };

void os_getArtEui(u1_t* buf) {

    memcpy_P(buf, APPEUI, 8);

}


// This should also be in little endian format, see above.

```



```

static const u1_t PROGMEM DEVEUI[8] = { DEVEUI };

void os_getDevEui(u1_t* buf) {

    memcpy_P(buf, DEVEUI, 8);

}

// This key should be in big endian format (or, since it is not really a

// number but a block of memory, endianness does not really apply). In

// practice, a key taken from ttncctl can be copied as-is.

static const u1_t PROGMEM APPKEY[16] = { APPKEY };

void os_getDevKey(u1_t* buf) {

    memcpy_P(buf, APPKEY, 16);

}

static osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty

// cycle limitations).

const unsigned TX_INTERVAL = 30;

//LoRa pin mapping ESP32 (LILYGO Board V1.1)

const lmic_pinmap lmic_pins = {

    .nss = 18,

```

```

.rxtx = LMIC_UNUSED_PIN,

.rst = 23,

.dio = { 26, 33, 32 },

};

void printHex2(unsigned v) {

    v &= 0xff;

    if (v < 16)

        Serial.print('0');

    Serial.print(v, HEX);

}

void onEvent(ev_t ev) {

    Serial.print(os_getTime());

    Serial.print(": ");

    switch (ev) {

        case EV_SCAN_TIMEOUT:

            Serial.println(F("EV_SCAN_TIMEOUT"));

            break;

        case EV_BEACON_FOUND:

```

```

Serial.println(F("EV_BEACON_FOUND"));

break;

case EV_BEACON_MISSED:

Serial.println(F("EV_BEACON_MISSED"));

break;

case EV_BEACON_TRACKED:

Serial.println(F("EV_BEACON_TRACKED"));

break;

case EV_JOINING:

Serial.println(F("EV_JOINING"));

break;

case EV_JOINED:

Serial.println(F("EV_JOINED"));

{

u4_t netid = 0;

devaddr_t devaddr = 0;

u1_t nwkey[16];

u1_t artKey[16];

```

```

LMIC_getSessionKeys(&netid, &devaddr, nwkKey, artKey);

Serial.print("netid: ");

Serial.println(netid, DEC);

Serial.print("devaddr: ");

Serial.println(devaddr, HEX);

Serial.print("AppSKey: ");

for (size_t i = 0; i < sizeof(artKey); ++i) {

    if (i != 0)

        Serial.print("-");

    printHex2(artKey[i]);

}

Serial.println("");

Serial.print("NwkSKey: ");

for (size_t i = 0; i < sizeof(nwkKey); ++i) {

    if (i != 0)

        Serial.print("-");

    printHex2(nwkKey[i]);

}

```

```

    Serial.println();

    join_success = true;

}

// Disable link check validation (automatically enabled

// during join, but because slow data rates change max TX

// size, we don't use it in this example.

LMIC_setLinkCheckMode(0);

break;

/*

|| This event is defined but not used in the code. No

|| point in wasting codespace on it.

||

|| case EV_RFU1:

||   Serial.println(F("EV_RFU1"));

||   break;

*/

case EV_JOIN_FAILED:

    Serial.println(F("EV_JOIN_FAILED"));

```

```

    break;

case EV_REJOIN_FAILED:

    Serial.println(F("EV_REJOIN_FAILED"));

    break;

case EV_TXCOMPLETE:

    Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));

    if (LMIC.txrxFlags & TXRX_ACK)

        Serial.println(F("Received ack"));

    if (LMIC.dataLen) {

        Serial.print(F("Received "));

        Serial.print(LMIC.dataLen);

        Serial.println(F(" bytes of payload"));

    }

    // Schedule next transmission

    os_setTimedCallback(&sendjob, os_getTime() +
sec2osticks(TX_INTERVAL), do_send);

    break;

case EV_LOST_TSYNC:

```

```

    Serial.println(F("EV_LOST_TSYNC"));

    break;

case EV_RESET:

    Serial.println(F("EV_RESET"));

    break;

case EV_RXCOMPLETE:

    // data received in ping slot

    Serial.println(F("EV_RXCOMPLETE"));

    break;

case EV_LINK_DEAD:

    Serial.println(F("EV_LINK_DEAD"));

    break;

case EV_LINK_ALIVE:

    Serial.println(F("EV_LINK_ALIVE"));

    break;

/*

|| This event is defined but not used in the code. No

|| point in wasting codespace on it.

```

```

||

|| case EV_SCAN_FOUND:

||   Serial.println(F("EV_SCAN_FOUND"));

||   break;

*/

case EV_TXSTART:

    Serial.println(F("EV_TXSTART"));

    break;

case EV_TXCANCELED:

    Serial.println(F("EV_TXCANCELED"));

    break;

case EV_RXSTART:

    /* do not print anything -- it wrecks timing */

    break;

case EV_JOIN_TXCOMPLETE:

    Serial.println(F("EV_JOIN_TXCOMPLETE: no JoinAccept"));

    break;

```



```

default:

    Serial.print(F("Unknown event: "));

    Serial.println((unsigned)ev);

    break;

}

}

void do_send(osjob_t* j) {

    // Check if there is not a current TX/RX job running

    if (LMIC.opmode & OP_TXRXPEND) {

        Serial.println(F("OP_TXRXPEND, not sending"));

    } else {

        // Prepare upstream data transmission at the next possible time.

        Serial.print("Payload: ");

        int x = 0;

        while (x < sizeof(tx_payload)) {

            printHex2(tx_payload[x]);

```

```

        Serial.print(" ");

        x++;

    }

    Serial.println();

    LMIC_setTxData2(1, tx_payload, sizeof(tx_payload), 0);

    Serial.println(F("Packet queued"));

}

// Next TX is scheduled after TX_COMPLETE event.

}

void setup() {

    Serial.begin(115200);

    Serial.println("Starting");

    Serial1.begin(9600, SERIAL_8N1, SERIAL1_RX, SERIAL1_TX);

    //SPI pin mapping ESP32 (LILYGO Board V1.1)

    SPI.begin(5, 19, 27, 18);

```

```

#ifdef VCC_ENABLE

    // For Pinoccio Scout boards

    pinMode(VCC_ENABLE, OUTPUT);

    digitalWrite(VCC_ENABLE, HIGH);

    delay(1000);

#endif

    // LMIC init

    os_init();

    // Reset the MAC state. Session and pending data transfers will be discarded.

    LMIC_reset();

    // Start job (sending automatically starts OTAA too)

    do_send(&sendjob);

    //Set FS for RX2 (SF9 = TTN,TTS EU)

    LMIC.dn2Dr = DR_SF9;

}

```

```

void loop() {

  os_runloop_once();

  if (join_success == true) {

    read_sentence = Serial1.readStringUntil(13); //13 = return (ASCII)

    read_sentence.trim();

    if (read_sentence.startsWith("$GPGGA")) {

      String gps_lat = sentence_sep(read_sentence, 2); //Latitude in degrees &
minutes

      String gps_lon = sentence_sep(read_sentence, 4); //Longitude in degrees &
minutes

      String gps_sat = sentence_sep(read_sentence, 7);

      String gps_hgt = sentence_sep(read_sentence, 9);

      String gps_lat_o = sentence_sep(read_sentence, 3); //Orientation (N or S)

      String gps_lon_o = sentence_sep(read_sentence, 5); //Orientation (E or W)

      // /*

      // Serial.print("LAT: ");

      // Serial.print(Latitude);

```

```

// Serial.print(" LON: ");

// Serial.print(Longitude);

// Serial.print(" ALT: ");

// Serial.print(Altitude);

// Serial.print(" SAT: ");

// Serial.println(Sat);

// */

float Latitude = convert_gps_coord(gps_lat.toFloat(), gps_lat_o);

float Longitude = convert_gps_coord(gps_lon.toFloat(), gps_lon_o);

float Altitude = gps_hgt.toFloat();

int Sat = gps_sat.toInt();

generate_payload(Latitude, Longitude, Altitude, Sat);

}

}

}

String sentence_sep(String input, int index) {

int finder = 0;

```

```

int strIndex[] = { 0, -1 };

int maxIndex = input.length() - 1;

for (int i = 0; i <= maxIndex && finder <= index; i++) {

    if (input.charAt(i) == ',' || i == maxIndex) {

        finder++;

        strIndex[0] = strIndex[1] + 1;

        strIndex[1] = (i == maxIndex) ? i + 1 : i;

    }

}

return finder > index ? input.substring(strIndex[0], strIndex[1]) : "";

}

```

```

float convert_gps_coord(float deg_min, String orientation) {

    double gps_min = fmod((double)deg_min, 100.0);

    int gps_deg = deg_min / 100;

    double dec_deg = gps_deg + (gps_min / 60);

    if (orientation == "W" || orientation == "S") {

```

```

    dec_deg = 0 - dec_deg;

}

return dec_deg;

}

void generate_payload(double lat, double lon, int alt, int sat) {

    uint32_t LatitudeBinary = ((lat + 90) * 1000000);

    uint32_t LongitudeBinary = ((lon + 180) * 1000000);

    uint8_t payload[11];

    payload[0] = (LatitudeBinary >> 24) & 0xFF;

    payload[1] = (LatitudeBinary >> 16) & 0xFF;

    payload[2] = (LatitudeBinary >> 8) & 0xFF;

    payload[3] = LatitudeBinary & 0xFF;

    payload[4] = (LongitudeBinary >> 24) & 0xFF;

    payload[5] = (LongitudeBinary >> 16) & 0xFF;

```

```
payload[6] = (LongitudeBinary >> 8) & 0xFF;
```

```
payload[7] = LongitudeBinary & 0xFF;
```

```
payload[8] = (alt >> 8) & 0xFF;
```

```
payload[9] = alt & 0xFF;
```

```
payload[10] = sat & 0xFF;
```

```
int i = 0;
```

```
while (i < sizeof(payload)) {
```

```
    tx_payload[i] = payload[i];
```

```
    i++;
```

```
}
```

```
}
```