PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA MINISTRY
OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF YAHIA FARES MEDEA



Faculty of Technology

End of Studies Thesis

Department: Electrical Engineering

Specialty: Instrumentation Electronics

A Communication System for Student Access Control and Bus Location Monitoring

Presented by: Abdelmadjid Marmouze & Youcef Belaidi

Supervised by: DR. Kherkhar Abdelkrim

Graduating Class of 2025

Abstract

This project addresses the inefficiencies and security challenges in traditional university transportation systems, which rely heavily on manual supervision and the decisions of bus drivers. These outdated practices often result in transportation unavailability, student overcrowding, and unauthorized access.

The proposed solution uses Intelligent Transport Systems principles to develop an automatic transportation management system. Key technologies employed include RFID for student identification and access control, GPS for real-time bus tracking, and 4G LTE communication for synchronizing data between embedded hardware (T-SIM A7670E) and a centralized backend utilizing SQLite and MySQL databases. A semi-automatic tripod turnstile mechanism is integrated to enforce secure access procedures.

The system was successfully implemented and functionally tested in realistic scenarios. Results showed effective management of student access, reliable real-time tracking of buses, and consistent synchronization of data between the local and central databases. The system also provided clear performance metrics, including access control accuracy and tracking latency.

الملخص

يعالج هذا المشروع أوجه القصور والتحديات الأمنية في أنظمة النقل الجامعي التقليدية، التي تعتمد بشكل كبير على الإشراف اليدوي وقرارات سائقي الحافلات. وغالبًا ما تؤدي هذه الممارسات القديمة إلى عدم توفر وسائل النقل، وازدحام الطلاب، والدخول غير المصرح به.

يعتمد الحل المقترح على مبادئ أنظمة النقل الذكية (ITS) لتطوير نظام ألي لإدارة النقل. ومن بين التقنيات الأساسية المستخدمة: تقنية تحديد الهوية بموجات الراديو (RFID) للتعرف على الطلاب والتحكم في الدخول، ونظام تحديد المواقع العالمي (GPS) لتتبع الحافلات في الوقت الحقيقي، وتقنية الاتصال 4 LTE Gلمزامنة البيانات بين العتاد المدمج-T) (SIM A7670Eونظام خلفي مركزي يعتمد على قواعد بيانات SQLite و .MySQLكما تم دمج آلية بوابة دوارة ثلاثية الأرجل شبه أوتوماتيكية لتطبيق إجراءات دخول آمنة.

تم تنفيذ النظام بنجاح واختباره وظيفيًا في سيناريوهات واقعية، وأظهرت النتائج قدرة فعالة على إدارة دخول الطلاب، وتتبع موثوق للحافلات في الوقت الفعلي، ومزامنة مستقرة للبيانات بين القاعدة المحلية والمركزية. كما وفر النظام مؤشرات أداء واضحة مثل دقة التحكم في الدخول وزمن تأخير التتبع.

Keywords

Intelligent Transport Systems, RFID Access Control, Real-Time Tracking, GPS Location, 4G LTE Communication, IoT in Transportation, Student Transportation Management, Embedded Systems, Data Synchronization, ESP32.

الكلمات المفتاحية

أنظمة النقل الذكية، التحكم في الوصول باستخدام RFID ، التتبع في الوقت الحقيقي، تحديد الموقع باستخدام GPS ، الاتصال عبر شبكة LTE 4G ، إنترنت الأشياء في النقل، نظام إدارة النقل الطلابي، الأنظمة المدمجة، مزامنة البيانات ESP32.

Table of Contents

Abstract i
Table of Contents ii
List of Figures v
List of Tables vi
Notations and Symbols vii
General Introduction 1
1. CHAPTER 1 Theoretical Background 2
1.1. Introduction 3
1.2. Introduction to ITS 3
1.2.1. Definition of ITS 3
1.2.2. ITS Architecture 3
1.2.3. ITS Applications 3
1.2.4. Real-World Implementations of ITS 4
1.3. Security and Management in ITS 5
1.3.1. Identification, Authentication, Authorization, and Accounting (IAAA) 5
1.3.1.1. Identification 5
1.3.1.4. Accounting 6
1.3.1.5. IAAA application in ITS 6
1.4. Technologies Used in ITS 6
1.4.1. Data Acquisition Technologies 7
1.4.1.1. Radio Frequency Identification (RFID) Technology 7
1.4.1.2. GPS Technology 8
1.4.2. Communication Technologies 8
1.4.2.1. Cellular Communication (4G/LTE) 9
1.4.2.2. Long-Range Low-Power Communication (LoRa) 9
1.4.2.3. Communication Protocols 10
1.4.2.3.1. HyperText Transfer Protocol 10
1.4.2.3.2. Message Queuing Telemetry Transport (MQTT) Protocol 11
1.4.3. Real-Time Data Processing 12
1.4.3.1. Computing Models 12

1.4.3.1.1. Cloud Computing 12
1.4.3.1.2. Edge Computing 12
1.5. Conclusion 13
2. CHAPTER 2 System Design and Implementation 14
2.1. Introduction 15
2.2. System Architecture 15
2.2.1. System Overview 15
2.2.2. System Work Flow 16
2.3. Hardware Design 18
2.3.1. T-SIM A7670E (Microcontroller) 18
2.3.1.1. Overview of the T-SIM A7670E Board 18
2.3.1.2. ESP32 Core 19
2.3.1.3. GNSS Support 19
2.3.1.4. SD Card support 20
2.3.1.5. SIM A7670E LTE Modem 20
2.3.1.6. Power Handling 20
2.3.2. RC522 reader (RFID Reader Module) 21
2.3.3. Semi-automatic Tripod Turnstile (Blocking Mechanism) 21
2.3.3.1. The Blocking Mechanism Core 22
2.3.3.2. Solenoid Actuation System 23
2.3.4. Driver Interface 24
2.4. System Data Flow & Communication 25
2.4.1. Overview of Data Flow 25
2.4.2. Local database (SQLite) 25
2.4.2.1. Local database structure 25
2.4.2.2. Local data processing 26
2.4.3. Central Database (MySQL) 27
2.4.3.1. Central database structure 27
2.4.3.2. Data Sync Strategy and Error Handling 28
2.5. Conclusion 30
3. CHAPTER 3 System Testing and Evaluation 31

3.1. Introduction 32
3.2. Testing Environment 32
3.3. Test Cases and Results 33
3.3.1. Functional Testing 33
3.3.2. Performance Testing 34
3.3.3. Security Testing 35
3.4. System Evaluation 36
3.4.1. Performance Metrics 36
3.4.1.1. Accuracy of Real-Time Tracking 36
3.4.1.2. Prediction Accuracy for Arrival Times 36
3.4.1.3. real-time data sync latency 37
3.4.2. Limitations and Improvements 38
3.5. Conclusion 40
General Conclusion 41
References 42
Appendix B 45
Appendix C 47
Appendix D 48
Appendix E 49
Appendix F 50
Appendix G1 52
Appendix G2 53
Appendix H 54

List of Figures

Figure 1.1: Intelligent Transportation System architecture 3
Figure 1.2: Passive RFID Concept 7
Figure 1.3: GPS Satellite Positioning [8]. 8
Figure 1.4: LoRa/LTE Bandwidth vs Range [10]. 9
Figure 1.5: HTTP protocol architecture 10
Figure 1.6: MQTT Publish-subscribe Architecture [12]. 11
Figure 1.7: Edge computing contrasted with cloud computing [14]. 12
Figure 2.1: System General Architecture 15
Figure 2.2: System Work Flow Organizational Chart 17
Figure 2.3: LilyGO T-SIM A7670E Development Board 18
Figure 2.4: Tripod Turnstile Blocking mechanism [18] 22
Figure 2.5: Solenoid Driver Circuit 23
Figure 2.6: Driver Interface Work Flow 25
Figure 2.7: Visualization of Students Table (SQLite) 26
Figure 2.8: Visualization of BusStats Table (SQLite) 27
Figure 2.9: General Database Network 27
Figure 2.10: Visualization of buses Table (MySQL) 29
Figure 2.11: Visualization of buses Table (MySQL) 29
Figure 2.12: Visualization of EntryLogs Table (SQLite) 29
Figure 3.1: Real-Time Tracking Accuracy per Trip 36
Figure 3.2: ETA Prediction Error per Trip 37
Figure 3.3: Real-Time Data Sync Latency per Trip 38

List of Tables

Table 1.1: IAAA Working Principle 6
Table 2.1: T-SIM A7670E Specifications [15] 18
Table 2.2: SD Card Pinout [15] 20
Table 2.3: RC522 Pinout [17] 21
Table 2.4: Solenoid Characteristics 23
Table 2.5: Interface Pinout. 24
Table 2.6: SQLite Database Structure 26
Table 2.7: MySQL Database Structure 28
Table 3.1: Software List 32
Table 3.2: Functional Testing Results 34
Table 3.3: Performance Testing Results 35
Table 3.4: Security Testing 35

Notations and Symbols

Abbreviations

ADC – Analog-to-Digital Converter

ANPR – Automatic Number Plate Recognition

DAC – Digital-to-Analog Converter

ESP – Espressif Systems

ETA – Estimated Time of Arrival

GNSS – Global Navigation Satellite System

GPIO – General Purpose Input/Output

GPS – Global Positioning System

GSM – Global System for Mobile Communications

HTTP – HyperText Transfer Protocol

I2C – Inter-Integrated Circuit

IAAA – Identity, Authentication, Authorization, Accounting

ID – Identifier

IP – Internet Protocol

IoT – Internet of Things

ITS – Intelligent Transportation System

LTE – Long-Term Evolution

LoRa – Long Range

LPWAN – Low Power Wide Area Network

MAC – Media Access Control

MQTT – Message Queuing Telemetry Transport

PWM – Pulse Width Modulation

RFID – Radio Frequency Identification

SNMP – Simple Network Management Protocol

SIM – Subscriber Identity Module

SPI – Serial Peripheral Interface

UART – Universal Asynchronous Receiver-Transmitter

General Introduction

In recent years, the evolution of technologies has significantly impacted the public transportation sector [1], especially with the integration of Intelligent Transport Systems (ITS) [2]. These systems aim to improve the efficiency, safety, and convenience of transportation services by incorporating advanced data acquisition, processing, and communication technologies.

The transportation system adopted by universities has proven to be inefficient due to its poor management of transportation security and availability, as it depends on outdated and manual processes such as security agents to control student access and bus drivers' decisions to provide transportation services. This traditional system provided multiple issues such as:

Transportation unavailability: it's often caused by a shortage of buses, which happens due to road traffic or drivers' delay.

Student overcrowding: it usually occurs when most buses are occupied during busy times. This often leads students to either waste time by waiting for the next bus or squeeze their way into the crowd, forcing them to stand in uncomfortable and unsafe conditions.

Potential security risks: unauthorized access, lost items, and theft are some of the risks that arise from the lack of supervision at bus stations due to limited coverage by security agents.

These issues highlighted the need for a more controlled, and well-monitored transportation system that meets the modern demands for automation, monitoring, and responsiveness.

To address these issues, we proposed in this thesis a design and implementation of a student transportation management system based on ITS principles. The proposed solution aims to modernize university transportation by integrating key technologies such as real-time GPS tracking for bus location monitoring, RFID-based access control to ensure secure and authorized access, and predictive analytics to estimate arrival times and manage passenger capacity.

This thesis is structured into three main chapters. Chapter 1 provides the theoretical background, including a comprehensive overview of ITS technologies and their role to public transportation. Chapter 2 covers the design and development of the system architecture, detailing the hardware and software components used in the solution. Finally, Chapter 3 presents the testing process, system evaluation, and performance analysis based on real-world trials, highlighting both the strengths and limitations of the developed prototype.

CHAPTER 1
Theoretical Background

1.1. Introduction

With the integration of modern technology, the development of transportation systems has advanced significantly, leading to the creation of what is now known as Intelligent Transport Systems (ITS). These systems aim to enhance the efficiency, safety, and sustainability of transportation networks by incorporating advanced technologies such as data acquisition devices, wireless communication, and real-time data processing.

This chapter offers a solid theoretical foundation for understanding the fundamental components of ITS, including everything from its architecture and real-world applications to understanding the technologies that are used within ITS.

1.2. Introduction to ITS

1.2.1. Definition of ITS

An ITS is a modern application of the Internet of Things (IoT) that integrates into transportation systems a variety of information and electronics technologies that are based on wireless and wired communication protocols. These systems aim to enhance the efficiency, safety, and reliability of transportation networks by enabling intelligent decision-making and real-time data exchange between vehicles (V2V) and between vehicles and infrastructure (V2I), which leads to a better user experience [1].

1.2.2. ITS Architecture

No matter the purpose behind creating an intelligent transportation system, the principles behind how it works usually have some similarities from one system to another. An ITS model starts by collecting real-time data, which is then transmitted to processing units for analysis (Figure 1.1). These units interpret the information and make real-time decisions that are usually physical changes or instructions in the form of control signals. The system continuously monitors its performance and adapts over time [1].

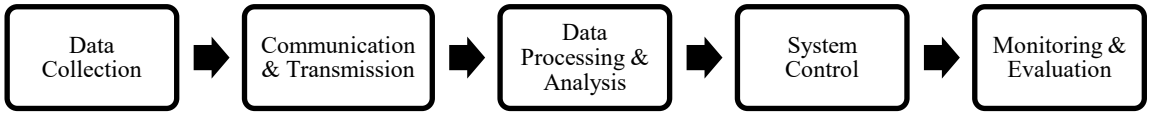


Figure 1.1: Intelligent Transportation System architecture

1.2.3. ITS Applications

These systems can offer variety of applications to the public transportation like:

- **Automation**

Automation has invaded multiple domains, and transportation is one of them. It can significantly help in reducing human intervention in daily transportation operations which

minimize human error, and ensure consistent performance. For example, automated ticketing systems, RFID-based identification systems....

- **Real-time data collection monitoring and analytics**

ITS have the ability to collect and process real-time data from various sensors which can help us achieve a continuous monitoring of vehicles, traffic flow, and passenger activity.

- **Multi-device access real-time data**

Operators, drivers, and even passengers can use various devices to interact with ITS real-time data. This interaction can help users stay informed and react instantly, track vehicle locations, and receive real-time schedule or delay information. Administrators can also monitor and operate the system remotely.

1.2.4. Real-World Implementations of ITS

several cities worldwide have managed to integrate intelligence into their existing transportation systems.

- Stockholm's congestion pricing system uses RFID and ANPR (Automatic Number Plate Recognition) technologies for vehicle identification. These systems communicate over a secure TCP/IP-based network, with roadside units transmitting data to a centralized control center via cellular (4G) or fiber-optic links. The control center uses real-time data exchange protocols such as HTTPS and WebSocket for dashboard updates and system alerts. Message Queuing Telemetry Transport (MQTT) may be used in subsystems for lightweight, real-time telemetry data exchange between traffic sensors and backend servers [1].
- London's Congestion Charging Zone is powered by a network of CCTV and ANPR cameras, integrated via wired Ethernet (LAN) and fiber-optic connections. These devices feed data into a central system over HTTPS using secure APIs. Internally, the system may use SOAP/RESTful APIs for module communication and SNMP (Simple Network Management Protocol) for device monitoring. Communication between various city transport services is standardized via DATEX II, a European data exchange protocol for traffic and travel information [2].
- Singapore's ITS leverages multiple technologies, including GPS, V2I (Vehicle-to-Infrastructure) communication, and cellular networks (4G/5G). Real-time data from taxis, buses, and road sensors is sent via UDP/TCP/IP protocols to the Land Transport Authority's backend servers. The system also uses DSRC (Dedicated Short-Range Communication) in some scenarios for low-latency vehicle-to-infrastructure messaging. IoT middleware platforms often rely on MQTT or AMQP (Advanced Message Queuing

Protocol) to collect and route data streams to machine learning services for traffic prediction [3].

- The TAP system uses RFID/NFC for contactless fare collection. Card readers installed on buses and subway turnstiles communicate over RS-485 or CAN Bus for local hardware interconnection and then forward data to the central server via Wi-Fi, Ethernet, or cellular 4G/LTE depending on vehicle location. Transactions are uploaded using HTTPS REST APIs to ensure secure communication. Backend systems use database replication protocols and JSON-based data exchange to synchronize fare and usage data across agencies [4].

1.3. Security and Management in ITS

1.3.1. Identification, Authentication, Authorization, and Accounting (IAAA)

IAAA is a foundational concept in modern intelligent systems, especially in systems that rely on networks and require strong security. It stands for four core processes that ensure secure and controlled access to digital services and infrastructure.

Here's how each process works and why it's critical:

1.3.1.1. Identification

Identification is the process of claiming an identity. It's a way of answering: *"Who are you?"*. Before access is taken into consideration, this is the initial stage in identifying who is attempting to access. Access can be granted if a user, device, or system presents an identifier, such as a username, email, RFID tag ID, or device MAC address [5].

1.3.1.2. Authentication

Authentication is the process of checking the reliability of who is attempting to access. In this process we use various Authentication factors to prevent unauthorized access and provide a strong security [5]. Those factors are typically divided into:

- Something you know (e.g., password)
- Something you have (e.g., security token)
- Something you are (e.g., fingerprint)

1.3.1.3. Authorization

Authorization defines the permissions and roles that each authenticated person is allowed to have which ensures that users can only access what they're supposed to. For example, a user might be allowed to view data but not modify it [5].

1.3.1.4. Accounting

Accounting tracks and monitors the actions taken by users and systems. It answers: *"What did you do, when, and for how long?"*. It can provide a record of logging session activities, resource usage, login times, duration, access patterns, and more [5].

Table 1.1: IAAA Working Principle

Function	Purpose	Example
Identification	Claiming an identity	Typing a username
Authentication	Proving identity	Entering a password
Authorization	Granting rights	Access to dashboard only
Accounting	Tracking usage	Logging user login/logout times

1.3.1.5. IAAA application in ITS

The principles of IAAA are applied across both the software and hardware components of Intelligent Transport Systems.

On the software side, IAAA ensures that only authorized users can access or interact with systems such as ticketing platforms, traffic management dashboards, or passenger data services. This protects sensitive information and ensures secure operations.

On the hardware side, IAAA processes are used to control physical access. The Identification, Authentication and Authorization mechanisms are used to allow only verified individuals to access Intelligent Transport like buses, trains, or station gates. This verification can take many forms depending on the purpose of the system. For example, it can check if a person's public transport subscription is still active, confirm whether a student's ID exists in the transportation database, or verify if an employee is listed for company bus use before granting access. In each case, the system ensures that only authorized individuals are allowed access based on predetermined standards.

The accounting part of IAAA is responsible for tracking and recording access events. It keeps a detailed log of who accessed the system, when, and for what purpose. This helps in monitoring usage patterns, detecting unusual activities, and generating reports for analysis. In transportation systems, for example, accounting can log each time a person enters or exits a vehicle or facility, which is useful for both security and service optimization.

1.4. Technologies Used in ITS

The efficiency of ITS depends on the integration of various technologies that are responsible for acquiring, communicating, and processing data in real time. The following sections highlight some of the technological components used in ITS.

1.4.1. Data Acquisition Technologies

1.4.1.1. Radio Frequency Identification (RFID) Technology

RFID is an automatic identification technology that uses radio waves for data transmission and identification. It stores information in electronic tags and communicates via radio waves between reading devices (readers) and tags, enabling the tracking and identification of items [6]. This technology is classified into two types: Active and Passive. Active RFIDs are battery-powered, which enables the ID chip to emit signals on its own. On the other hand, Passive RFIDs require a scanner to transmit a radio signal to the ID chip to prompt it to emit signals in response [6].

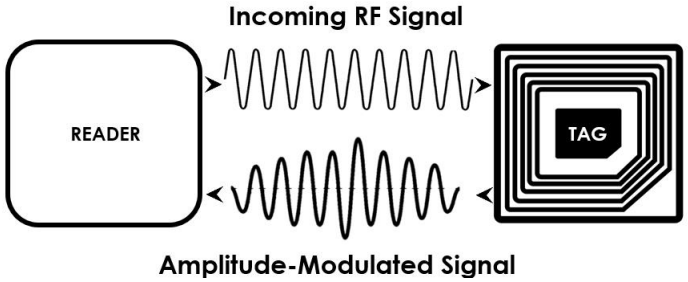


Figure 1.2: Passive RFID Concept

The RFID technology has been integrated into public transportation through a variety of practical applications, such as:

- **Ticketing System:** In public transportation, RFID technology helped in automating the ticketing procedure. All passengers have to do is hold their RFID card near the ticketing machine, which allows the system to quickly finish identification verification and fee deduction. This approach significantly helped to reduce wait times and the requirement for manual processes, which minimizes the mistake rate, while also improving passenger convenience [7]. RFID also facilitates travel without paper. Instead of using ordinary paper tickets, travelers can use electronic transportation cards or smartphone applications. This concept reduces paper waste, which not only makes travel more convenient but also advances sustainable development goals [7].
- **Passenger Management:** The RFID system can monitor passengers' activity on transportation in real-time. The managers can reasonably plan capacity after a detailed analysis of peak hours and passenger behaviors to ensure sufficient service and comfort for passengers during busy periods [7]. In addition, when passengers use RFID cards, their identity information is recorded and tracked for verification and security checks when needed. This measure effectively prevents unauthorized travel and improves public safety [7].

- **Data Analysis and Optimization:** The RFID system can collect substantial data related to passenger travel, including entry and exit times and passenger flow counts. Analyzing this data allows transportation management to better understand passenger needs, optimizing operational strategies and service arrangements to enhance the overall travel experience [7].

1.4.1.2. GPS Technology

GPS stands for Global Positioning System. It operates by calculating the distance between a user's receiver and multiple satellites in orbit. By combining signals from at least four satellites, the receiver can determine its exact position on Earth through a method called trilateration.[8]

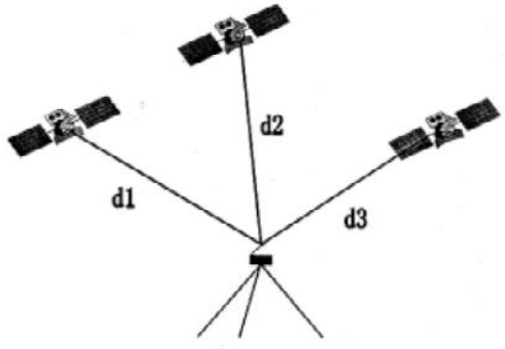


Figure 1.3: GPS Satellite Positioning [8].

This process allows for real-time location tracking which enables several critical applications in various aspects of public transportation systems such as:

- **Real-Time Vehicle Tracking:** GPS allows operators to monitor the location of buses, trains, or other public transport vehicles in real-time. Knowing such type of information can help reduce passenger wait times and improve schedule accuracy.
- **Traffic Management:** GPS data helps identify traffic congestion in real-time, allowing authorities to adjust signals, redirect traffic, or issue alerts.
- **Emergency Response and Safety:** In case of accidents or emergencies, GPS can help locate the exact position of a vehicle quickly leading for an improvement in response time.

1.4.2. Communication Technologies

Communication technologies are responsible of transmitting the data collected from Data Acquisition Technologies to remote servers, databases, or control centers. Real-time monitoring, control, and decision-making wouldn’t be possible without a strong communication setup.

1.4.2.1. Cellular Communication (4G/LTE)

4G LTE stands for Fourth Generation Long-Term Evolution. It's a globally adopted standard for high-speed wireless communication, offering lower latency and greater bandwidth compared to previous generations. While this technology can reach speeds of up to 100 Mbps under ideal conditions, real-world performance varies depending on location and network quality. It can reach a maximum speed of 15Mbps which makes it very interesting for the Internet of Things [9].

4G LTE consumes high power due to several factors, including constant connectivity through SIM cards that relies on active service subscriptions, high data transmission rates, and complex signal processing requirements. These characteristics make it less suitable for applications that rely on long battery life, such as low-power IoT devices. However, features like constant connectivity and seamless communication with cell towers, make LTE an ideal choice for fast-moving devices in ITS, where real-time data transmission and continuous network access are essential [9].

1.4.2.2. Long-Range Low-Power Communication (LoRa)

LoRa is a Low Power Wide Area Network (LPWAN) technology designed for long-range communication with minimal energy consumption. It operates over distances up to 15–20 km in rural areas and 2–5 km in urban settings. Although it supports only very low data rates (0.3 to 50 kbps) and has higher latency, its extremely low power usage makes it ideal for battery-powered devices that need to operate for years without maintenance. LoRa uses narrowband channels (125–500 kHz), keeping costs low and making it suitable for private deployments in applications like environmental monitoring, smart agriculture, and remote sensor networks where high-speed data transfer is not required [10].

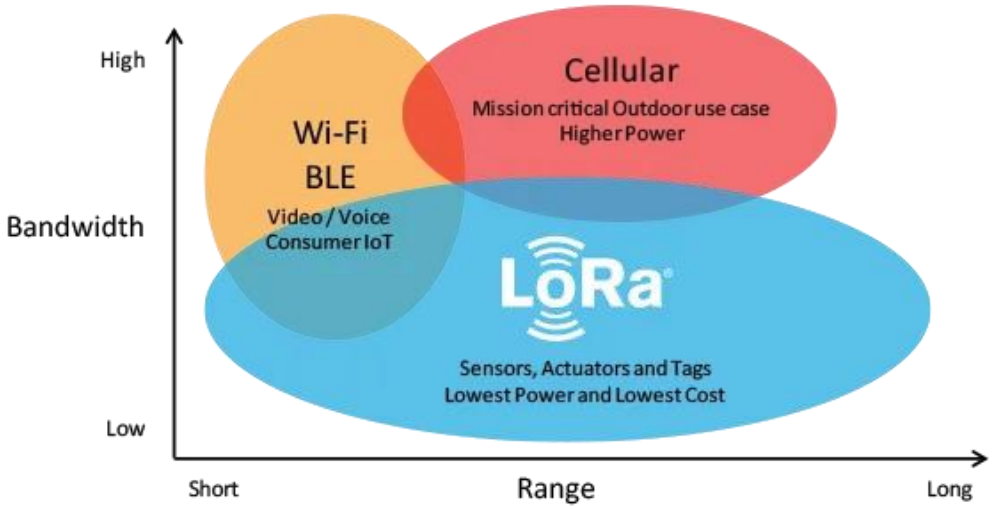


Figure 1.4: LoRa/LTE Bandwidth vs Range [10].

1.4.2.3. Communication Protocols

Communication protocols are fundamental to the IoT, enabling devices to exchange data reliably and efficiently. Among the most widely used protocols in IoT are HTTP and MQTT protocols which serves our communication needs.

1.4.2.3.1. HyperText Transfer Protocol

HTTP Protocol is the foundation of data communication on the web. It enables the exchange of information between clients (like browsers or microcontrollers) and backend servers. The communication between clients and servers is done by exchanging individual messages (Figure 1.5). The messages sent by the client are called *requests* and the messages sent by the server as an answer are called *responses* [11].

Clients can request a server to perform a variety of actions on a resource (like data on a database). These actions are called HTTP methods which define the type of operation being carried out in a web communication. Some of the most relevant HTTP methods are:

- **GET**: Retrieve data from a specified resource without modifying it [11]. Example: Fetching student records from a transportation database.
- **POST**: Send data to the server to create/process a new resource [11]. Example: Submitting a new student's RFID access credentials.
- **PUT** or **PATCH**: Update existing resources. PUT replaces the entire resource, while PATCH applies partial modifications [11]. *Example*: Updating a bus's real-time GPS coordinates (PUT) or correcting a typo in a student's name (PATCH).
- **DELETE**: Remove a specified resource from the server [11]. *Example*: Deleting an expired RFID access record.

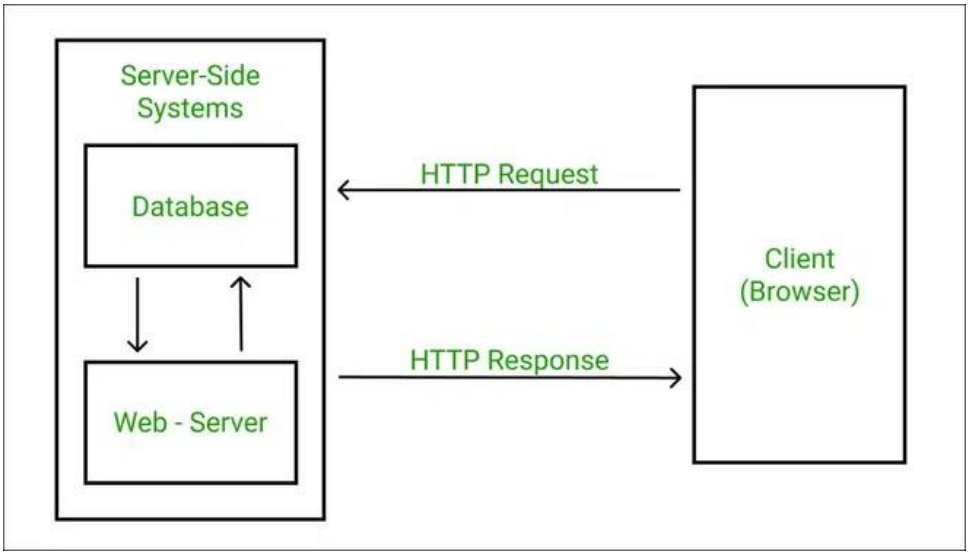


Figure 1.5: HTTP protocol architecture

The HTTP protocol is commonly used in IoT systems, enabling devices such as sensors, microcontrollers, and embedded systems to send and receive data over the internet. This communication often follows a structured approach using RESTful APIs (Representational State Transfer Application Programming Interfaces), which define how devices can access and manipulate web-based resources [11].

1.4.2.3.2. Message Queuing Telemetry Transport (MQTT) Protocol

MQTT is an OASIS standard messaging protocol for the IoT. It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth (Figure 1.6). MQTT today is used in a wide variety of industries, such as automotive, manufacturing, telecommunications, oil and gas, etc. [12]

The MQTT Publish-Subscribe pattern has four main components:

- ***MQTT Broker:*** is like a post office. It receives messages from devices (clients) and sends them to the right subscribers. It's the central part that manages all communication in the MQTT system [12].
- ***MQTT Client*****:** is any device or application that connects to the broker. It can send (publish) messages, receive (subscribe to) messages, or do both [12].
- ***Publishing:*** means sending a message to the broker on a specific topic. For example, a temperature sensor can publish data like "25°C" to a topic called "home/livingroom/temperature" [12].
- ***Subscribing:*** means a client asks the broker to send it any messages published to a specific topic. For example, an app subscribed to "home/livingroom/temperature" will receive the temperature updates from the sensor [12].

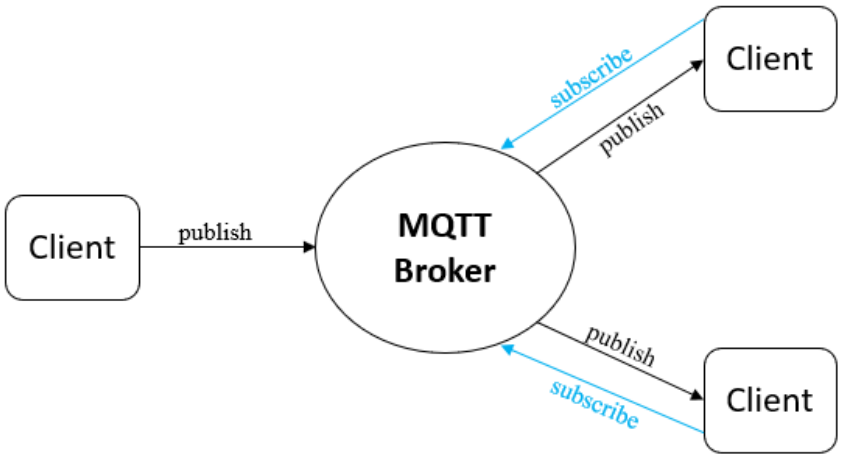


Figure 1.6: MQTT Publish-subscribe Architecture [12].

1.4.3. Real-Time Data Processing

Real-time data processing is what stands out for intelligent transport systems. Without it, ITS would lose its “intelligence” and become just a static transport management system. That's due to its crucial role in making immediate decisions based on multiple integrated data streams.

1.4.3.1. Computing Models

1.4.3.1.1. Cloud Computing

cloud computing uses internet to connect users to a cloud platform where they request and access rented computing services [13]. cloud platforms host databases, offering both data storage and processing capabilities, enabling applications to access, manage, and analyze data remotely.

This model relies on continuous communication with remote server programs that located in data centers, these data centers are often located thousands of miles away, often in areas that have cheap electricity and cold climate, so the amount of energy required for cooling is minimized [13].

1.4.3.1.2. Edge Computing

Edge Computing is a computing framework that processes and analyses data closer to the point where it's created (real-time) rather than sending all data to a centralized data center or cloud to reduce latency, minimize bandwidth usage and improve response time [14].

This framework plays a critical role by enabling small local processors (like microcontrollers or edge servers) to process real-time data coming from a network of IoT devices, like sensors and actuators Instead of sending every bit of data to the cloud, to analyze and respond to the data right where it’s collected [14].

Edge technology is used in many domains that requires real-time data processing such as healthcare monitoring, Agriculture, industrial automation and ITS [14].

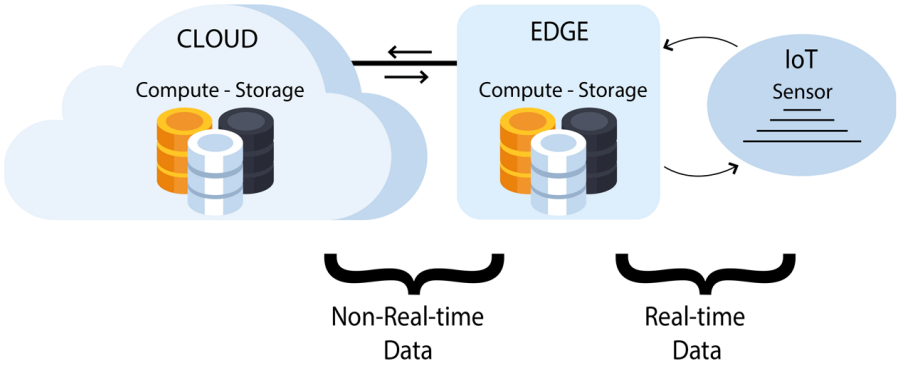


Figure 1.7: Edge computing contrasted with cloud computing [14].

1.5. Conclusion

In this chapter, we explored the fundamental concepts that work together to achieve intelligent, secure, and efficient transport solutions. By understanding how data is acquired, transmitted, processed, and protected, we can see how ITS is capable in solving modern transportation challenges. After having this theoretical background, we are now ready to move into the next chapter, which will focus on the design, implementation, and integration of a real-world ITS prototype.

CHAPTER 2
System Design and Implementation

2.1. Introduction

This chapter will demonstrate how the concepts and technologies introduced above are practically applied through a combination of hardware components and intelligent software design to create a connected intelligent transportation system for students.

We begin by highlighting the overall architecture and work flow of the proposed system and its main components, which integrates RFID-based identification, real-time location tracking, and centralized data synchronization. Next, we discuss how the system processes data locally and how it uses wireless communication to provide real-time monitoring and tracking remotely.

By the end of this chapter, there will be a clear understanding of how our proposed intelligent transportation system will deliver a seamless and responsive transportation solution.

2.2. System Architecture

2.2.1. System Overview

The proposed system is divided into two main segments (Figure 2.1):

- On-Bus Unit (Embedded System)
- Cloud Infrastructure (Remote Monitoring & Control)

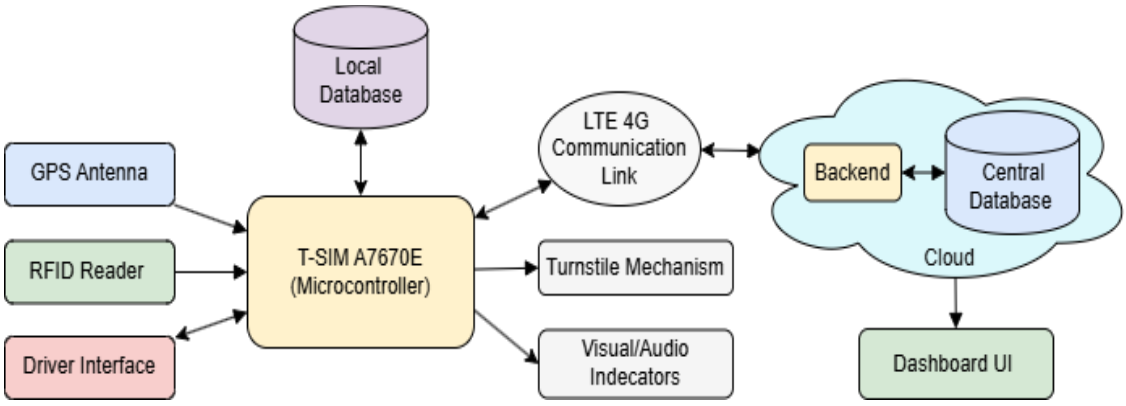


Figure 2.1: System General Architecture

The on-bus unit (see **Appendix B** for General Circuit Schematic) is an embedded system designed to manage access control, location tracking, and data logging in real-time. The core of the system is the T-SIM A7670E module, which combines a microcontroller, cellular connectivity (for data transmission), local storage and multi-GNSS (for location tracking). This module is paired with an RFID reader (for student authentication), a driver interface, a blocking mechanism (to secure access), and a stable power supply (to ensure uninterrupted operation).

The cloud infrastructure is composed of a backend server that handles incoming data from the on-bus unit and a centralized database that securely stores this information. The backend is responsible for processing and validating the data before inserting it into the database. This architecture ensures reliable data flow from mobile units to the cloud, enabling real-time monitoring, analytics, and future scalability.

2.2.2. System Work Flow

As showed in the flow chart below (figure 2.2), the system starts by initializing all the sensors and modules used within the system such as the RFID Reader, SD card, LTE, and GPS. If one of the modules fails to initialize, the system then displays an error code and stop afterwards. Otherwise, the system displays “swipe your ID” and wait for students to scan their identification tag.

If a student swipes their identification tag on the RFID reader, the system checks its authenticity according to two conditions: whether it exists in the authorized students list within the local SQLite database and whether it has not been used within a defined cooldown period. If either condition fails. The blocking mechanism stays locked and indicates that access to transportation is denied (LEDs, buzzer...). If access is granted, the system unlocks the blocking mechanism, indicates that the access is granted (LEDs, buzzer...), records the entry within the history access record, and increases the current passenger capacity in the database.

At the same time, transportation drivers can interact with the system using a dedicated interface which has a menu section that can be accessed by pressing on the menu button to set the next destination or configure maximum capacity of the bus. This feature ensures flexibility for route adjustments or overcrowding prevention.

The system constantly checks both of the internet connectivity and the backend response status. If both are available the system provides Real-time synchronization and continuous updates of dynamic data such as the vehicle’s location, current passenger capacity, and next destination, enabling remote monitoring of the estimated time of arrival (ETA), occupancy level, and Incoming transportation availability. On the other hand, If the system is offline or the server isn’t responding, the data get stored within the local database until connection is restored.

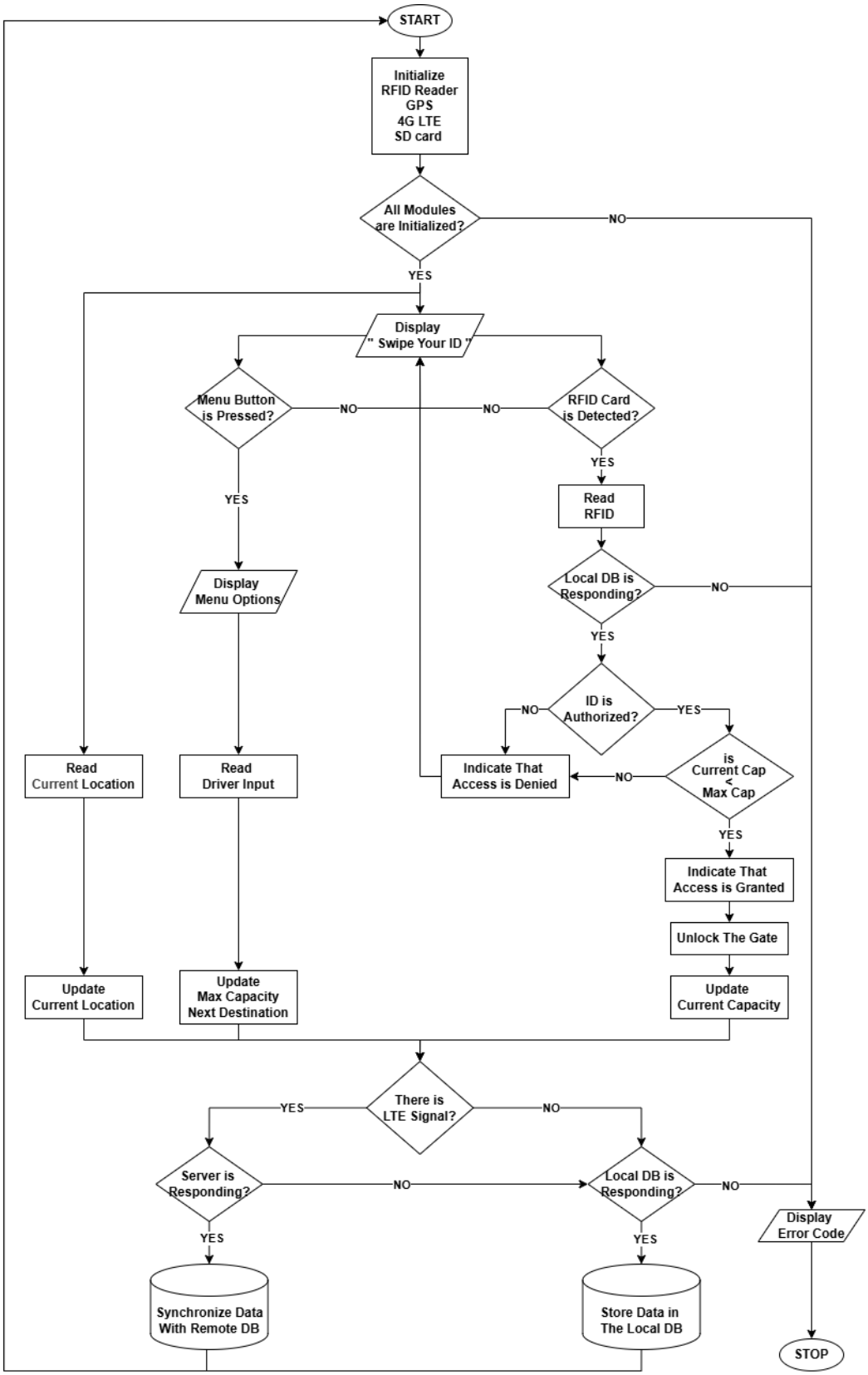


Figure 2.2: System Work Flow Organizational Chart

2.3. Hardware Design

2.3.1. T-SIM A7670E (Microcontroller)

2.3.1.1. Overview of the T-SIM A7670E Board

The T-SIM A7670E is a development board module from LILYGO that integrates an ESP32 WROVER-B microcontroller with a SIM7670E cellular module, GNSS support via a dedicated antenna, and a "18650" Li-ion battery slot. This combination provides efficient processing, wireless connectivity, and power flexibility, making it an ideal choice for mobile applications such as our transportation system.



Figure 2.3: LilyGO T-SIM A7670E Development Board

Here's a table containing most crucial technical specifications and features that made us choose the T-SIM A7670E for our system (see [15] for more details):

Table 2.1: T-SIM A7670E Specifications [15]

Category	**Specification**
Microcontroller	ESP32 WROVER-B (dual-core 32-bit, 240 MHz, 520 KB SRAM, 8 MB PSRAM)
Cellular Module	SIM7670E – supports 2G and 4G (LTE Cat 1)
GNSS Support	Integrated GPS, GLONASS, BeiDou (via SIM7670E) with external active GNSS antenna port
Battery Support	"18650" Li-ion battery slot with onboard charging and protection circuit

LTE Bands	B1/B3/B5/B7/B8/B20 (works with Algerian mobile network operators)
Supported Networks	LTE-FDD, GSM, GPRS, EDGE
SIM Card	1.8V/3.0V (Supports Nano/Micro SIM)
Data Rates	LTE: 10 Mbps DL / 5 Mbps UL
	GPRS/EDGE: 236.8 Kbps DL/UL
Interfaces	SPI, I2C, UART, ADC, DAC, PWM
Protocols	TCP/IP, HTTP/HTTPS, MQTT, FTP/FTPS, DNS, TLS
Supply Voltage	3.4V–4.2V
Power consumption	LTE : ~3.8 mA (idle) /GSM : ~3.5 mA (idle)

2.3.1.2. ESP32 Core

The core processing unit of the T-SIM A7670E board is based on the ESP32 WROVER-B module, a powerful dual-core 32-bit microcontroller operating at a clock speed of 240 MHz It features 520 KB of SRAM and an additional 8 MB of PSRAM, offering sufficient memory resources for multitasking and concurrent execution of embedded applications.

The ESP32 is responsible for managing RFID operations, peripheral communication (e.g., SD card, visual indicators), and logic-based decision-making. Its support for interfaces such as SPI, I²C, UART, ADC, DAC, and PWM makes it highly adaptable for various I/O operations required by the system.

2.3.1.3. GNSS Support

The T-SIM A7670E includes integrated GNSS functionality via the SIM7600 series modem, supporting major global satellite systems such as GPS, GLONASS, and BeiDou which improves satellite visibility and ensures more reliable positioning, particularly in urban environments.

The GNSS system uses an external active GNSS antenna that typically achieve horizontal positioning accuracy within 2.5 to 5 meters Under ideal conditions. However, this accuracy may vary depending on environmental conditions, antenna quality, and satellite availability.

2.3.1.4. SD Card support

The T-SIM A7670E development board features a built-in TF card slot that is fully compatible with microSD cards which eliminates the need for additional SD card modules. The TF card slot operates via SPI communication protocol, utilizing the following pin configuration:

Table 2.2: SD Card Pinout [15]

Pin Type	Pin Description	T-SIM A7670E pins
MOSI	Master In Slave Out	GPIO 15
MOSI	Master Out Slave In	GPIO 02
SCK	Serial Clock	GPIO 14
CS	Chip Select	GPIO 13

The microSD is used as a local database storage unit in our system. it stores sensitive and essential data such as the list of authorized student IDs used for RFID-based access authentication, which can be remotely updated. it also stores a predefined list of bus stations from which the driver can select the next destination through a dedicated interface. Additionally, the system logs unsynchronized access events locally when network connectivity is unavailable.

2.3.1.5. SIM A7670E LTE Modem

The board integrates a SIMCom A7670E modem, which enables cellular connectivity through both 2G (GSM) and 4G (LTE Cat 1) networks. This module allows for remote communication with the base station via mobile networks available in Algeria, using LTE-FDD and GSM/EDGE protocols [16].

The integrated modem supports data rates of up to 10 Mbps (downlink) and 5 Mbps (uplink) in LTE mode, and 236.8 Kbps for GPRS/EDGE, making it suitable for sending passenger data, location updates, and access logs in near real-time using multiple protocols such as TCP/IP, HTTP/HTTPS, and MQTT [16].

2.3.1.6. Power Handling

The T-SIM A7670E features a dedicated power management system designed to operate on a single 18650 Li-ion battery. It includes onboard charging and overcurrent protection circuitry, ensuring safe and reliable energy delivery to all modules. The board supports a voltage range between 3.4V and 4.2V.

Power consumption varies based on communication mode, with idle current draw around 3.3 mA for LTE and approximately 3.5 mA for GSM. This efficient power design enables prolonged operation in mobile environments like buses, even when disconnected from external power sources.

2.3.2. RC522 reader (RFID Reader Module)

The RC522 is a compact 13.56 MHz RFID reader module based on the MFRC522 chip, commonly used for short-range (~3–5 cm) wireless identification due to its reader/writer antenna that's designed to communicate with ISO/IEC 14443A cards and transponders without additional active circuitry [16].

This module was selected for its low cost, extended support, and reliable performance in embedded applications as it operates at 3.3V and communicates via SPI, making it directly compatible with the ESP32 through the following pin configuration:

Table 2.3: RC522 Pinout [17]

Pin Type	Pin Description	T-SIM A7670E pins
MOSI	Master In Slave Out	GPIO 23
MOSI	Master Out Slave In	GPIO 19
SCK	Serial Clock	GPIO 18
SDA	Chip Select	GPIO 5
RST	Reset	3.3 V
3.3V	Voltage Input	3.3 V
GND	Ground	GND

The RC522 module supports secure sector-based memory access and uses cryptographic keys (Key A and Key B) to control read/write operations. In this system, MIFARE Classic 1K RFID tags are pre-initialized with encrypted identifiers written into protected data blocks. During access control, the RC522 authenticates with Key A to securely read the stored data, enabling a reliable identification of passengers.

2.3.3. Semi-automatic Tripod Turnstile (Blocking Mechanism)

A Semi-automatic Tripod Turnstile is a type of access control barrier commonly used in places such as stadiums, museums, and public transport stations. It consists of three metallic arms (shaped like tripod) that rotates in one direction when access is granted, allowing the user to pass through.

2.3.3.1. The Blocking Mechanism Core

The engineering behind the blocking mechanism core (Figure 2.3) that's used to block the tripod arms from rotating until access is granted mainly consists of:

- ***Disk Wheel***: This is a circular plate fixed to the rotating shaft that holds the tripod arms. It usually has notches or cutouts that align with the locking mechanism.
- ***Lock Rod:*** A spring-loaded metal rod that slides in and out to lock or unlock the disk wheel.
- ***Turnstile-opening Solenoid:*** This is an electromechanical actuator responsible for pulling the lock rod out of the disk wheel notch it.
- ***Position Sensor:*** Often a hall effect sensor or mechanical limit switch mounted to detect the rotation angle or completion of a turn.

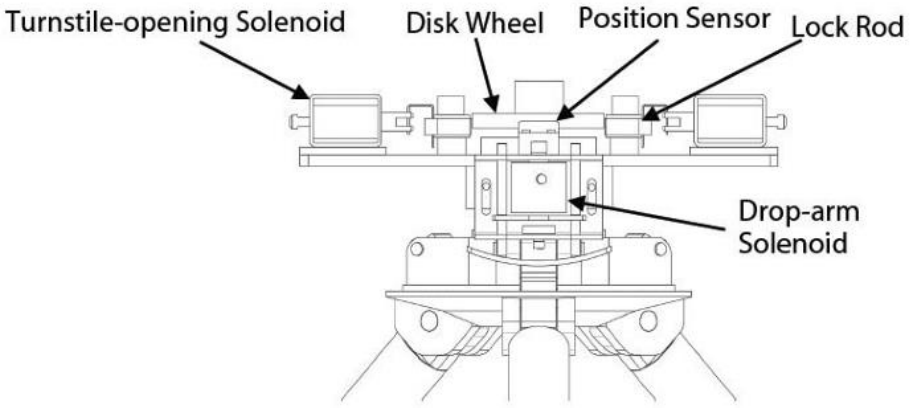


Figure 2.4: Tripod Turnstile Blocking mechanism [18]

The disk wheel is mechanically coupled to the rotating shaft of the tripod arms and contains notches that define fixed locking positions. A spring-loaded lock rod engages with these notches to prevent unauthorized rotation. When access is granted, the microcontroller activates the solenoid, which temporarily retracts the lock rod, allowing the disk wheel and attached arms to rotate freely.

Once the user has passed through and the disk wheel completes its rotation, a position sensor detects the correct alignment for re-locking. The solenoid is then de-energized, and the spring mechanism pushes the lock rod back into the next available notch, securely locking the system in place until the next authorized access. This mechanism ensures both mechanical reliability and controlled, one-way passage through the turnstile.

2.3.3.2. Solenoid Actuation System

When selecting a solenoid, it is important to consider the electrical characteristics and the mechanical characteristics of the solenoid, as well as the specific requirements of the application. Here's a summary of the most common characteristics:

Table 2.4: Solenoid Characteristics

Electrical Characteristics			Mechanical Characteristics	
Characteristic	Typical Value		Characteristic	Typical Value
Voltage	12V DC /24V DC		Type	*Push/ Pull type*
Current Draw	0.3 A – 1.2 A		Stroke Length	5 mm – 10 mm
Power Rating	5W – 15W		Holding Force	5 N – 15 N
Coil Resistance	10Ω – 40Ω		Response Time	$< 100\ ms$

The solenoid actuator used in the system is the JF-0530B model, which operates at 12V and consumes approximately 0.8A, requiring 9.6W of power during activation [19] which can be supplied by the bus's buttery.

To control the solenoid, we created a separate circuit (Figure 2.4) based on the IRLZ44N logic-level N-channel MOSFET which basically works as an electronic switch that gets triggered by a digital 3.3V output coming from the SIM A7670E which allows us to either energize or de-energize the solenoids to engage or disengage the lock rod.

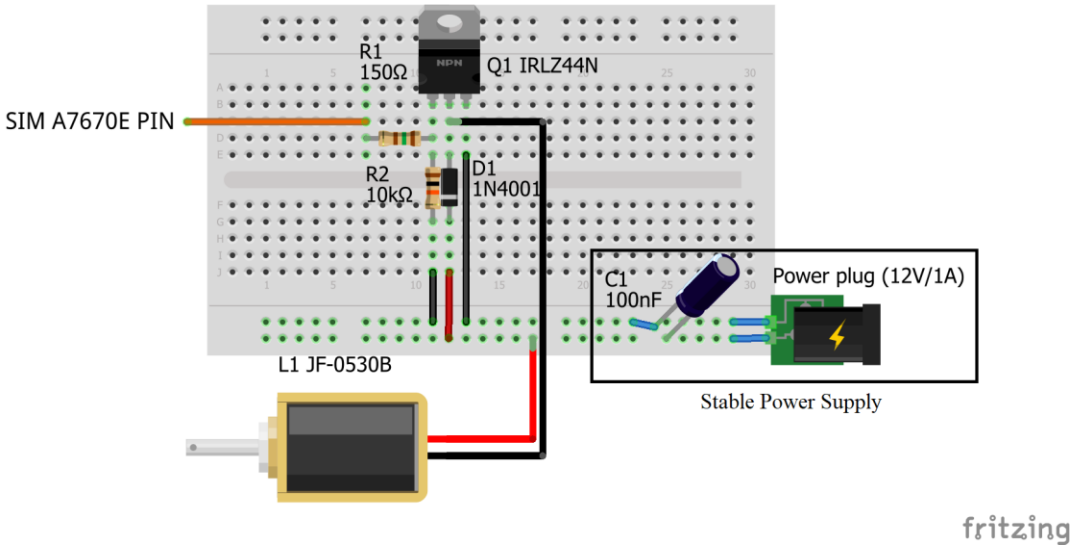


Figure 2.5: Solenoid Driver Circuit

The solenoid driver circuit has a gate resistor ($R_G = 150\Omega$) that limits the current flow coming from the GPIO pin, a pull-down resistor ($R_{pd} = 10k\Omega$) to ensure that the MOSFET remains off during idle states, a 1N4001 flyback diode connected in parallel with the coil to suppress inductive voltage spikes generated when the solenoid turns off and a ceramic capacitor ($C = 100nF$) is added across the power rails to reduce voltage ripple and ensure stable performance.

2.3.4. Driver Interface

The driver interface is built using a 16x2 LCD connected via the I2C protocol, along with four push buttons connected to the T-SIM A7670E's GPIO pins:

Table 2.5: Interface Pinout.

Part		Description	T-SIM A7670E pins
16x2 LCD	SDA	Serial Data	GPIO 21
	SCL	Serial Clock	GPIO 22
	GND	Ground	GND
	VCC	Voltage Input	VBUS (5V)
Button 1		Navigation (up/right/increase)	GPIO 32
Button 2		Select	GPIO 33
Button 3		Navigation (down/left/decrease)	GPIO 25
Button 4		Options menu access	GPIO 26

The interface logic is designed for intuitive interaction by the bus driver and the students. The driver can manage max capacity and the next destination of the bus by accessing the settings menu through the interface by holding the menu button (Button 4) for a few seconds. This brings up two menu options:

The driver can then navigate through these options using navigation buttons (Buttons 1 and 3) and select using the select button (Button 2). If the "Max Passengers" option is selected, the driver can adjust the allowed number of passengers that can access the bus and If "Destination" option is selected, the driver can scroll through predefined destinations (e.g., Destination A, B…).

When the driver updates one of these options, they get saved and synchronized with the remote database for real-time tracking and monitoring.

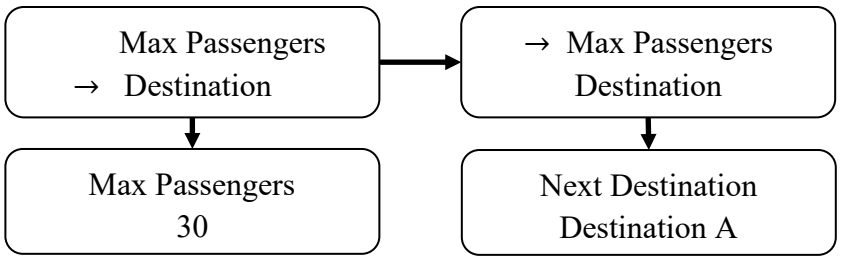


Figure 2.6: Driver Interface Work Flow

2.4. System Data Flow & Communication

2.4.1. Overview of Data Flow

The in-bus system communicates with two databases: a local database (built using SQLite) and a remote central database (built using MySQL) (see **Appendix C** for the system general Data flow) This dual-database setup ensures that the system remains functional even in areas with limited or no network connectivity.

All collected data including access logs, current capacity, maximum capacity, next destination, and bus location are stored locally in the SQLite database. If an LTE connection is available, the system attempts to synchronize this data with the remote MySQL database in real time. If the signal is unavailable, data is cached locally and will be synchronized once connectivity is restored.

2.4.2. Local database (SQLite)

2.4.2.1. Local database structure

The local database uses SQLite, a lightweight and efficient database engine that supports standard SQL (Structured Query Language) commands like:

- **SELECT:** to retrieve data,
- **INSERT:** to add new records,
- **UPDATE:** to modify entries, and
- **DELETE:** to remove data.

SQLite was chosen for its offline functionality, minimal setup, and reliability for small to medium workloads, making it ideal for embedded systems like our system, where low resource usage and offline data processing are critical.

The local database contains four main tables that stores bus real-time stats, students and stations lists and scan logs (Table 2.7) (see **Appendix G1** for the local database code structure).

Table 2.6: SQLite Database Structure

Table Name	Fields	Description
BusStats	BusID PlateNumber CurrentCapacity NextStationID MaxCapacity LastSyncTime	Stores real-time stats for each bus such as maximum capacity, current passenger count, upcoming station, and when is last time the bus stats have been synced with the remote server.
Students	RFID StudentID AllowedAccess LastScanTime	Contains information about each student, their RFID tag, access permissions, and the time of their last scan.
EntryLogs	LogID RFID BusID ScanTime SyncedStatus	Logs every RFID scan event, recording the student's RFID, the bus, the time, and sync status.
Stations	StationID StationName	Holds the list of bus station identifiers and their respective names. Used to track destinations.

2.4.2.2. Local data processing

Local data processing is critical to ensure real-time functionality even in the absence of internet connectivity. The following tasks are handled locally:

- **RFID Authorization**: The system checks the scanned RFID against the Students table to determine access.

	StudentID	RFID	AllowedAccess	SyncedStatus
	Filtre	Filtre	Filtre	Filtre
1	202032036180	8E85767D	0	1
2	202032036181	03B35C97	1	1

Figure 2.7: Visualization of Students Table (SQLite)

- **Capacity Management**: The BusStats table is updated after each successful entry to reflect the current onboard count.
- **Driver Input Handling**: Any updates from the driver (e.g., new destination or max capacity) are saved immediately in BusStats.

	BusID	PlateNumber	MaxCapacity	CurrentCapacity	NextStationID	LastSyncTime ▲
	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	BUS01	123456789	5	4	A	0

Figure 2.8: Visualization of BusStats Table (SQLite)

- **Log Entry**: Each granted entry scan is logged into the EntryLogs table with a timestamp (Figure 2.12).

2.4.3. Central Database (MySQL)

2.4.3.1. Central database structure

The central database uses MySQL; a high-performance relational database system designed for large data workload and enterprise-grade reliability. Unlike the embedded SQLite database used locally.

The central database collects data from each bus's local databases, creating a centralized system for real-time monitoring, analytics, and control. Each bus synchronizes its data, including access logs, current location and capacity, and destination updates, to the central MySQL database whenever a stable LTE connection is available.

This architecture combines the data sent and synced from multiple discrete datasets into one centralized dataset to provide centralized visibility across the entire fleet of buses. Creating a network of datasets (Figure 2.7) ensures real-time monitoring and tracking of multiple systems and facilitates data analysis remotely.

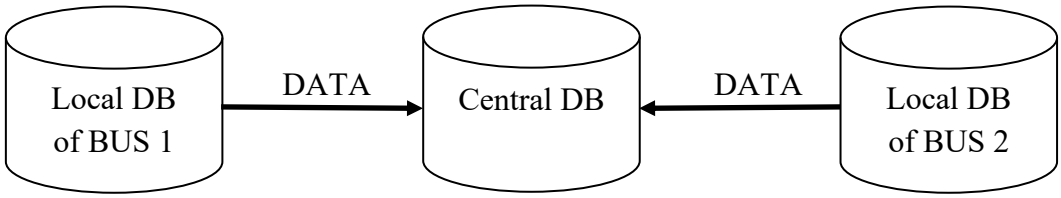


Figure 2.9: General Database Network

The central database contains more detailed and comprehensive tables than the ones stored in the local databases, such as a detailed students' list table, a multi-buses stats table, and a global scan logs table (Table 2.8) (see **Appendix G2** for the remote database code structure).

Table 2.7: MySQL Database Structure

Table Name	Fields	Description
buses	BusID PlateNumber MaxCapacity CurrentCapacity NextStationID LastSyncTime Latitude Longitude	Stores data about each bus including its ID, license plate, capacity details, upcoming station, location coordinates and the last time it synced.
students	StudentID FullName RFID Class AllowedAccess ContactInfo SyncedStatus	Contains comprehensive records of students, including full identity, RFID tag, educational info, and scan/access details.
entrylogs	LogID StudentID BusID Timestamp	Logs every RFID scan event with contextual data: which student boarded or left which bus, at what station, and whether it has been synced.
stations	StationID StationName	Holds the list of bus station identifiers and their respective names. Used to track destinations.

2.4.3.2. Data Sync Strategy and Error Handling

The system employs a hybrid data synchronization strategy that adapts based on the type of data being handled and the availability of network connectivity. Sync operations are selectively triggered depending on data priority and update frequency, ensuring efficient communication between each bus's local database and the central server.

- **Real-time Location Updates:** When network connectivity is available, the bus's GPS location coordinates (Longitude/ Latitude) are continuously transmitted to the backend in real time then saved in the central database within the buses table.
- **Driver Updates (Next Station / Max Capacity):** When the driver modifies configuration data such as the next destination or the bus's maximum passenger capacity, these updates are first saved to the local database and then immediately synced to the central database when network connectivity is available.
- **Capacity Tracking (Current Capacity):** Each time a passenger accesses the bus, the current capacity value is updated locally. This change is then synced with the central database in near real-time, assuming network availability.

BusID	PlateNumber	MaxCapacity	CurrentCapacity	NextStationID	LastSyncTime	latitude	longitude
BUS01	123456789	5	1	A	NULL	NULL	NULL
BUS02	987654321	10	3	B	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 2.10: Visualization of buses Table (MySQL)

- **Access Logs (EntryLogs):** RFID scan events are first stored in the local database. The logs are synced to the central database at regular entry times (sync unsynced logs each X entries) to ensure data consistency without overloading the network.

LogID	RFID	BusID	ScanTime
1	63FB4B97	BUS01	2025-01-01 00:00:21
2	223D981E	BUS01	2025-01-01 00:02:19
3	63FB4B97	BUS01	2025-01-01 00:02:30
4	223D981E	BUS01	2025-01-01 00:02:40
5	03B35C97	BUS01	2025-01-01 00:02:48
6	223D981E	BUS01	2025-01-01 00:03:00

Figure 2.11: Visualization of buses Table (MySQL)

In the event of a network failure or server unavailability unsynced data is flagged using a ***SyncedStatus*** field and retried until a successful acknowledgment is received from the backend (see **Appendix H** for the backend structure). Data is transferred in JSON format over a secure HTTP connection. Once the transaction is successfully into the central database, a flag is sent back the local system to mark those records as synced ($SyncedStatus = 1$).

	LogID	RFID	BusID	ScanTime	SyncedStatus
	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	63FB4B97	BUS01	2025-01-01 00:00:21	0

Figure 2.12: Visualization of EntryLogs Table (SQLite)

2.5. Conclusion

This chapter outlined the complete design and implementation of the automatic bus access and tracking system. It detailed the system's architecture, covering both hardware and software components such as the microcontroller, communication modules, sensors, and the driver interface.

The chapter also presented the structure and roles of the local and central databases, focusing on how they manage student access, track bus status, and handle data synchronization. With SQLite supporting offline data handling on the bus and MySQL managing centralized records, the system ensures reliable performance in varying network conditions. The data flow, synchronization strategy, and modular design enable the system to operate efficiently while allowing for future expansion.

These foundations prepare the system for evaluation in the next chapter, where its functionality and effectiveness will be further tested.

CHAPTER 3
System Testing and Evaluation

3.1. Introduction

Initially, we approached this project by testing each hardware component individually before integrating them one by one into a complete system. This modular testing approach helped us isolate and resolve issues efficiently, ensure hardware compatibility, and simplify troubleshooting during the integration phase. We adopted an iterative testing approach, where each component was tested multiple times under a variety of possible conditions. This process involved simultaneous monitoring of hardware behavior and software logic to enable real-time validation, using tools such as the Serial Monitor for data output and debugging. This method was particularly effective in tracing communication errors and validating sensor outputs. The overall goal of our testing methodology was to guarantee stability, operational efficiency, and reliable performance of the complete system.

3.2. Testing Environment

The testing environment consisted of both hardware and software tools that facilitated validation, debugging, and performance monitoring. On the hardware side, the LilyGO T-A7670 (ESP32-based) development board was connected with the computer through USB Type-c for programming and serial debugging and each paired component was wired on a breadboard for flexibility and visibility during initial tests. On the software side, the programs needed for the code development (see **Appendix D and E** for a code snippet) and testing are:

Table 3.1: Software List

Software	**Description**
PlatformIO IDE (VS Code)	Used to develop the main code of the development board and analyze results using Serial Monitor for logging outputs, capturing sensor data, and tracing errors.
Python (Flask) (VS Code)	Used to build the backend system, providing RESTful API endpoints.
Postman	Used to validate API endpoints, verify response formats, and confirm proper error handling.
SQLite DB Browser	Employed to view, query, and manage the structure and updates of the local database stored in the microcontroller. This helped ensure correct data insertion and reading during testing.
MySQL Workbench 8.0 CE	Responsible of visualizing synced data, managing queries, and comparing expected data flow with actual results.

3.3. Test Cases and Results

3.3.1. Functional Testing

- **RFID Card Detection**

Testing showed that the RFID module consistently detected student cards within a typical scan range of 2–4 cm. Multiple trials confirmed perfect acceptance rates and quick response times (~300 ms).

- **GPS Tracking and Location Accuracy**

The GPS module integrated into the T-SIM A7670E device retrieved the current location, speed and timestamp at regular intervals. In open-sky environments, the average positional accuracy remained within 3–5 meters.

However, in urban or obstructed areas, signal quality occasionally degraded, affecting location precision. Despite this, the data remained usable for route and arrival-time estimation.

- **Driver Menu Interface**

Functional testing of the I2C LCD and GPIO-based button interface confirmed that drivers could interact with the menu to update the current destination and maximum passenger limit.

Navigation between menu options and data confirmation was done smoothly. The interface provided clear feedback to the user, and driver inputs of the next destination/max capacity values were updated immediately in the local database.

- **Offline Operation and Local Logging**

In offline scenarios, the system continued to log RFID entries and to update variables such the current capacity locally. Once connectivity was restored, all data was successfully uploaded to the backend server. This verified the reliability of the local SQLite offline functionality.

- **Real-Time Data Synchronization**

The ESP32 module successfully transmitted data to the Flask backend over a 4G network. Both Buses and EntryLogs tables were synchronized without corruption (see **Appendix F** for the serial monitor outputs). The latency chart (see Section 3.4.1) shows that real-time sync delays averaged around 1.2 seconds in optimal conditions. When internet connection is lost, the retry mechanisms handled transmission failures effectively, ensuring consistency.

The table below (Table 3.2) summarizes all the results of our functional testing:

Table 3.2: Functional Testing Results

Test Case	Description	Expected Result	Actual Result	Status
TC1	RFID card detection	Detect UID of scanned cards and increase capacity each valid entry	UID displayed in Serial Monitor	Pass
TC2	GPS data reading	Get accurate latitude, longitude, speed, and timestamp	Data shown correctly in Serial Monitor	Pass
TC3	Driver interface (LCD + buttons)	Navigate menus and update values	LCD updates values in real time	Pass
TC4	Read/write SQLite database	Update bus stats, next station, location and capacity	Values updated successfully	Pass
TC5	SQLite local authentication	Check the scanned UID validity within the local DB	Access granted or denied message	Pass
TC6	Logging access within the local DB	Write log entry with timestamp within the local DB	Local Log table updated correctly	Pass
TC7	Remote database communication via LTE	Send bus stats and logs to remote DB	bus stats and entries appeared in remote DB	Pass

3.3.2. Performance Testing

The performance testing evaluated the system's technical capabilities under operational conditions to ensure all components meet specified requirements (within the acceptable range). Both hardware and software interactions were measured, including RFID processing delays, data logging speeds, GPS acquisition times, network latency, and interface responsiveness. The table below (Table 3.3) demonstrates the system's ability to handle critical operations and maintain real-time functionality.

Table 3.3: Performance Testing Results

Metric	Description	Measured Value	Acceptable Range	Result
RFID read delay	Time between card scan and UID output	~300ms	< 500ms	Pass
SD card write speed	Time to append log to file	~120ms	< 250ms	Pass
GPS fix time	Time to acquire satellite lock	~20s cold start	< 30s	Pass
LTE ping delay	Round-trip time to remote server	~150ms	< 300ms	Pass
UI response time	LCD update on button press	<200ms	<500ms	Pass

3.3.3. Security Testing

Security Testing was performed to put into test the system's ability to control access of unauthorized tag usage according to multiple scenarios. The results confirm proper handling of different unauthorized access cases such as duplicate scans, unauthorized RFID and bus capacity saturation.

Table 3.4: Security Testing

Test	Description	Expected Result	Actual Result
Duplicate RFID scan	Prevent multiple logging by same student in short time	"Access Denied" response	Pass
Unauthorized RFID use	Prevent access from unknown tags	"Access Denied" response	Pass
Student is not allowed to access	Prevent access if the student isn't allowed to access $AllowedAccess = 0$	"Access Denied" response	Pass
Current Capacity reached Max Capacity	Prevent access due to bus capacity saturation *Current Capacity=Max Capacity*	"Bus is Full" response	Pass

3.4. System Evaluation

To evaluate our system's performance, we put it the under the test during 10 trips from location A to location B where we analyzed the real-time data sync latency, real-time tracking accuracy, and arrival times accuracy.

3.4.1. Performance Metrics

3.4.1.1. Accuracy of Real-Time Tracking

The accuracy of Real-Time tracking represents how closely the system-reported location matches the actual GPS location at specific intervals. We used equation (1) To evaluate this accuracy. We needed to compare actual GPS coordinates and the reported GPS coordinates from our in-bus system every sampled every N minutes.

$$\text{Accuracy (\%)} = \big(1 - (\text{Average GPS Error}/\text{Total Route Distance})\big) \times 100 \text{ ------ (1)}$$

As shown in Figure 3.1, the real-time tracking accuracy varied across the 10 trips. Lower accuracy in Trips 4 (88%) and 9 (87%) was likely caused by urban signal blockage, which disrupts GPS signals. In contrast, Trip 10 achieved high accuracy (95%) due to open-road conditions and strong satellite visibility. To improve reliability in dense urban areas, hybrid positioning systems combining GPS, GLONASS, and inertial sensors could be implemented to mitigate signal loss and maintain accurate tracking performance.

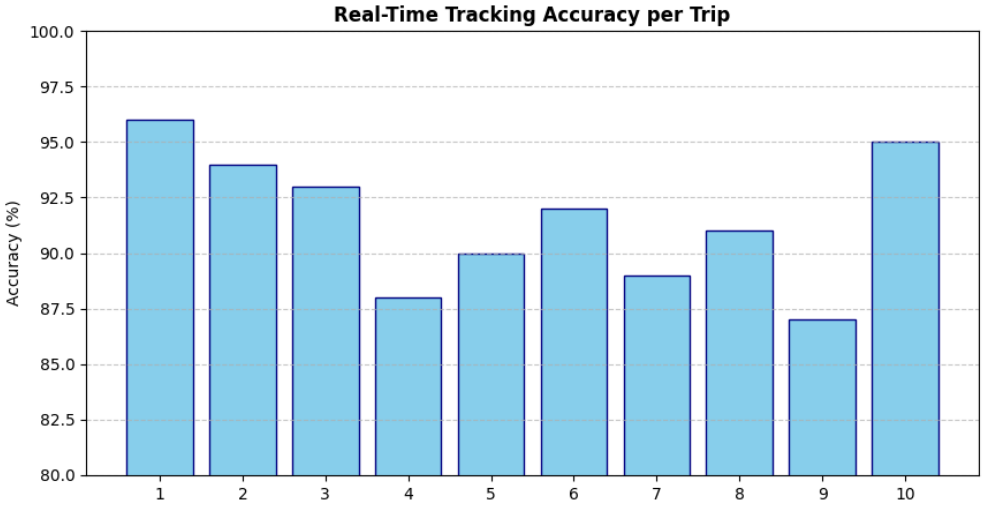


Figure 3.1: Real-Time Tracking Accuracy per Trip

3.4.1.2. Prediction Accuracy for Arrival Times

It represents how close our system's predicted arrival times are to the actual arrival times at station. To evaluate this accuracy, we needed to calculate the Error tolerance threshold between the "Actual Arrival Time" at the station and the "Estimated Arrival Time" that is

calculated by our system's logic. The Estimated Arrival Time was calculated using equation (2) mid-trip based on real-time GPS speed and road length between position.

$$\text{Prediction } Accuracy = |Predicted\ ETA - Actual\ Arrival\ Time| \text{ ----------(2)}$$

As illustrated in Figure 3.2, the largest ETA prediction errors occurred in Trip 9 (3.3 minutes) and Trip 4 (2.9 minutes), primarily due to unexpected traffic jams. In contrast, Trips 1 (0.6 minutes) and 10 (0.8 minutes) showed minimal errors, benefiting from clear traffic conditions and more predictable transit times.

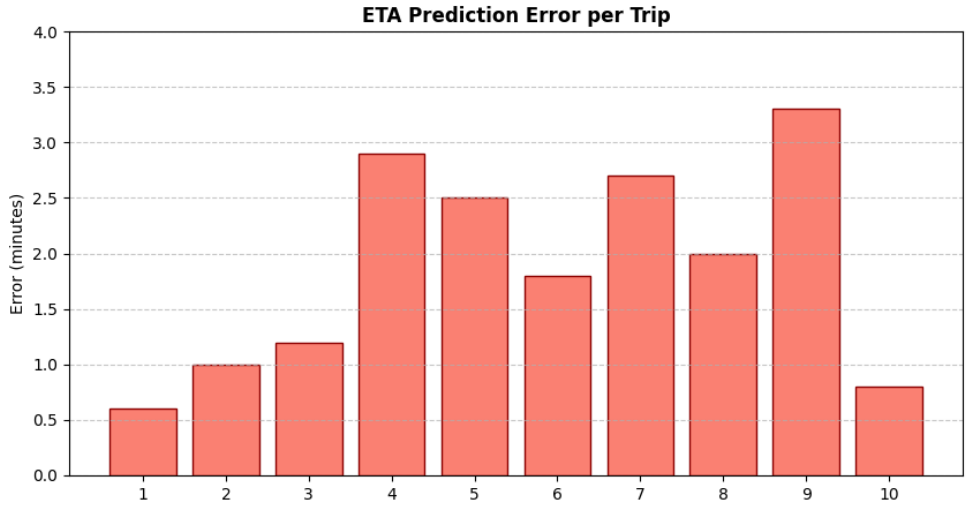


Figure 3.2: ETA Prediction Error per Trip

3.4.1.3. real-time data sync latency

Sync latency was measured as the time delay (seconds) between data transmission from the in-bus system to the backend server and confirmation of receipt via 4G LTE communication (SIM module).

We used equation (3) to calculate Sync latency considering the exact time (T1) when a data packet is sent by the bus's onboard system and the time (T2) when the backend server receives and processes the packet.

$$\text{Sync Latency (ms)} = \text{T2 (server receive time)} - \text{T1 (device send time)} \text{ -------(3)}$$

As we observe in the chart below (Figure 3.3), Trip 9 (6.2 s) and Trip 4 (5.9 s) experienced the worst latency due to 4G network congestion during peak hours and weak signal zones. In contrast, Trips 1 (2.5 s) and 10 (2.8 s) benefited from strong network coverage.

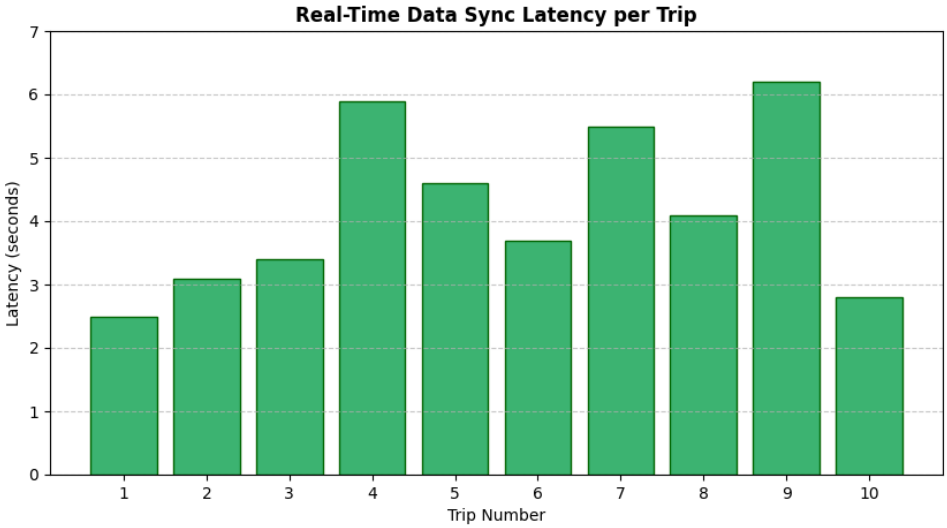


Figure 3.3: Real-Time Data Sync Latency per Trip

3.4.2. Limitations and Improvements

- **Bus Capacity Decreasing**

increasing the capacity when there is an entry can be done with using RFID identification but decreasing the bus capacity when a student exits the bus is a feature that needs to be integrated to achieve a final passenger counting accuracy. this feature can be applied by adding a second RFID with Weight and proximity Sensors on the exit.

- **Network Dependency**

Since the system transmits data to a centralized server using cellular (4G) connection, its real-time performance is based on network availability and stability. In areas with weak mobile coverage or during network congestion, data synchronization delays may occur, resulting in outdated information being displayed to stakeholders. This dependence on external network infrastructure introduces variability that may not be fully controllable from the system side, especially in remote or rural transit routes.

- **GPS Signal Interference**

The real-time tracking system relies heavily on GPS signals for accurate location updates. However, in dense urban environments with tall buildings or during adverse weather conditions, satellite signals may become reflected, leading to reduced positioning accuracy. This can cause a lack of accuracy between the actual and reported locations of the bus, which may impact the reliability of route tracking and estimated time of arrival (ETA) calculations for users.

- **Scalability Challenges**

While the system performs well under typical loads, handling a large number of passengers or multiple buses simultaneously introduces scalability concerns. As the volume of RFID scans, GPS updates, and log entries increases, the system may experience longer processing and synchronization times. These delays can compromise the “real-time” nature of the service, particularly when low-cost microcontrollers and local databases are used without optimizations for parallel processing or load distribution.

- **Hardware Challenges**

Implementing a scalable solution across multiple buses is effective but at the same time it requires advanced IoT hardware components such as the ones used in this project. these components can significantly increase deployment costs especially when considering a wide coverage of transportation.

- **Energy Efficiency**

The current system is powered by 18650 lithium-ion batteries. These batteries are cost-effective and widely available but may not offer sufficient stability and durability for continuous operations especially under heavy computational load. Voltage drops or insufficient charging cycles could lead to unexpected shutdowns or data loss. An integration of an alternative or supplementary power source, such as a solar panel or a higher-capacity rechargeable battery is recommended to enhance system stability. These additions would reduce maintenance requirements and ensure consistent power delivery in real-world deployments.

- **Lack of a Real-Time User Application**

While the system effectively collects and synchronizes real-time data such as bus location, next destination, and passenger capacity, it currently lacks a mobile or web application for student interaction. This limits the ability of students to monitor bus positions or receive estimated arrival times remotely. Adding an app that can monitor and track buses stats in real time to keep students updated anywhere about buses location and an estimation of their arrival time

3.5. Conclusion

This chapter tested and evaluated how well the system works in real-world conditions. The results showed that the RFID-based access control, GPS tracking, and 4G data syncing worked correctly in most cases.

The system was able to track buses in real time and store trip data successfully. Some issues were found, such as weak GPS signals in certain areas, reliance on stable 4G networks, and limited battery life. Despite these, the system performed well overall and proved that it can help improve student transportation by making it more secure, efficient, and easy to monitor.

.

General Conclusion

This thesis presented the design, implementation, and evaluation of a automatic student transportation management system inspired by the principles of Intelligent Transport Systems. The system was developed in response to critical challenges observed in traditional university transportation setups, including inefficiencies in student access control, lack of real-time tracking, and lack of bus availability.

By combining technologies such as GPS tracking, RFID authentication, embedded microcontrollers, and real-time data communication, the proposed solution offers an automated and responsive approach to managing student transport. The system enhances security by controlling students' access through RFID authentication, minimized students overcrowding by limiting the bus capacity, and optimizes transportation unavailability and user experience by providing up-to-date bus status and ETA predictions.

Through functional and performance testing, the system demonstrated its capability to operate successfully under real-world conditions, although certain limitations such as limited power dependency and network reliability highlight areas for future improvement. The project shows that automatic, IoT-driven transportation systems can provide a scalable and effective solution to the growing demands of modern transport systems.

References

[1] I. Chipilska, “What Is an Intelligent Transport System and How Does It Work?” Modeshift. Accessed: Apr. 13, 2025. [Online]. Available: https://www.modeshift.com/what-is-an-intelligent-transport-system-and-how-does-it-work/

[2] A. A. Christensen, “Intelligent Transport Systems for Smarter Cities | Smart Cities Dive.” Accessed: Apr. 13, 2025. [Online]. Available: https://www.smartcitiesdive.com/ex/sustainablecitiescollective/intelligent-transport-systems-smarter-cities/18319/

[3] “Land Transport Authority (LTA).” Accessed: Apr. 20, 2025. [Online]. Available: https://www.lta.gov.sg/content/ltagov/en.html

[4] TAP, “TAP Overview.” Accessed: Apr. 20, 2025. [Online]. Available: https://www.taptogo.net/articles/en_US/Website_content/about-tap

[5] “Identification, Authentication, Authorization, and Accountability,” Learning Security with Mayur. Accessed: Apr. 20, 2025. [Online]. Available: https://mayurpahwa.com/blog/cybersecurity/identification-authentication-authorization-and-accountability

[6] Y.-Y. Chen, M.-L. Tsai, and J.-K. Jan, "The design of RFID access control protocol using the strategy of indefinite-index and challenge-response," *Computer Communications*, vol. 34, no. 3, pp. 250–256, Mar. 2011. [Online]. Available: https://researchmgt.monash.edu/ws/portalfiles/portal/522993685/514718015_oa.pdf

[7] Luo, “RFID in Public Transport: A Comprehensive Guide,” RFID Card. Accessed: Apr. 15, 2025. [Online]. Available: https://www.rfidcard.com/rfid-in-public-transport-a-comprehensive-guide/

[8] Wang, Yingchun & Li, Yuan. (2021). The positioning principle of global positioning system and its application prospects. IOP Conference Series: Earth and Environmental Science. 781. 022085. 10.1088/1755-1315/781/2/022085.

[9] “What is 4G LTE? Understanding 4G LTE Technology and Its Impact,” Taoglas. Accessed: Apr. 20, 2025. [Online]. Available: https://www.taoglas.com/blogs/what-is-4g-lte-understanding-4g-lte-technology-and-its-impact/

[10] “What are LoRa and LoRaWAN?” The Things Network. Accessed: Apr. 20, 2025. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/

[11] “An overview of HTTP - HTTP | MDN.” Accessed: May 25, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview

[12] E. Team, “Introduction to MQTT Publish-Subscribe Pattern,” www.emqx.com. Accessed: Apr. 20, 2025. [Online]. Available: https://www.emqx.com/en/blog/mqtt-5-introduction-to-publish-subscribe-model

[13] “What is Cloud Computing?” Google Cloud. Accessed: Apr. 21, 2025. [Online]. Available: https://cloud.google.com/learn/what-is-cloud-computing

[14] “Edge computing key driver for wider IoT deployments | Industrial Ethernet Book.” Accessed: Apr. 19, 2025. [Online]. Available: https://iebmedia.com/technology/edge-cloud/edge-computing-key-driver-for-wider-iot-deployments/

[15] “T-A7670E/G/SA R2,” LILYGO®. Accessed: Apr. 21, 2025. [Online]. Available: https://lilygo.cc/products/t-sim-a7670e

[16] SIMCom, *A7670E Specification V1.00*, Feb. 2020. [Online]. Available: https://assets-global.website-files.com/619d4b4fac06f4717b357041/623069fff692f0693016f1af_A7670E_SPEC%25C2%25A0EN_202002.pdf

[17] NXP Semiconductors, "MFRC522 Standard performance MIFARE and NTAG frontend," Rev. 3.8, Jan. 2023. [Online]. Available: https://www.handsontec.com/dataspecs/RC522.pdf

[18] ZKTeco, *Tripod Turnstile Series Installation Guide*, Mar. 1, 2021. [Online]. Available: https://new-website-file.s3.ap-southeast-1.amazonaws.com/files/20211218/15d17a3be89e282638de7ad04430335f.pdf

[19] JFC Company, "JF-0530B Solenoid Technical Specifications," ver. 2.1, 2022. [Online]. Available: https://www.botnroll.com/en/solenoides/4622-solenoid-jf-0530b-12v-1a-5n-10mm-range.html

[20] “In-Depth: Interfacing an I2C LCD with Arduino,” Last Minute Engineers. Accessed: May 25, 2025. [Online]. Available: https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/

Appendix A

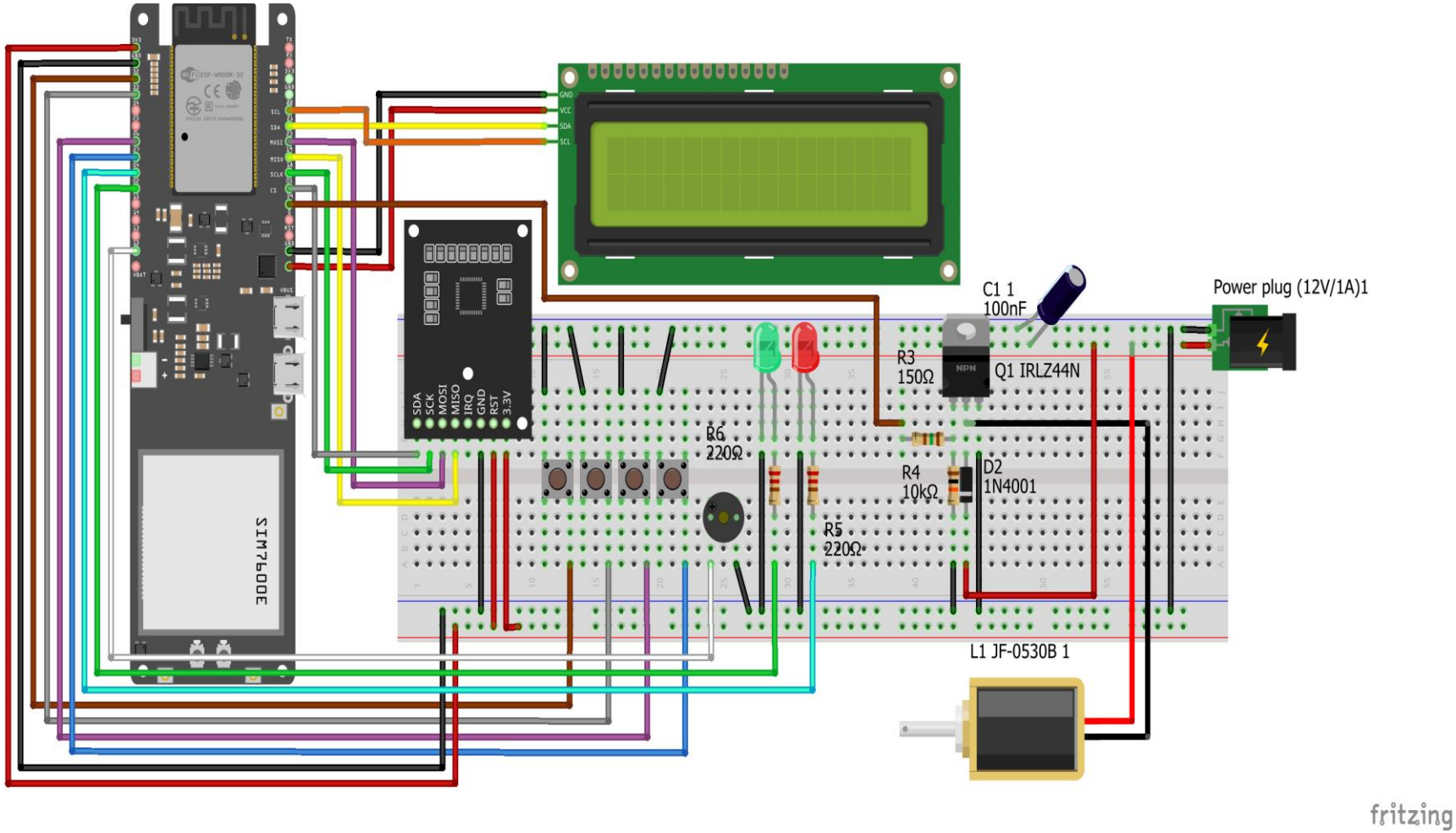
Hardware Components List

The table below summarizes all the major hardware components used in the implementation of the embedded system including their respective models and quantities:

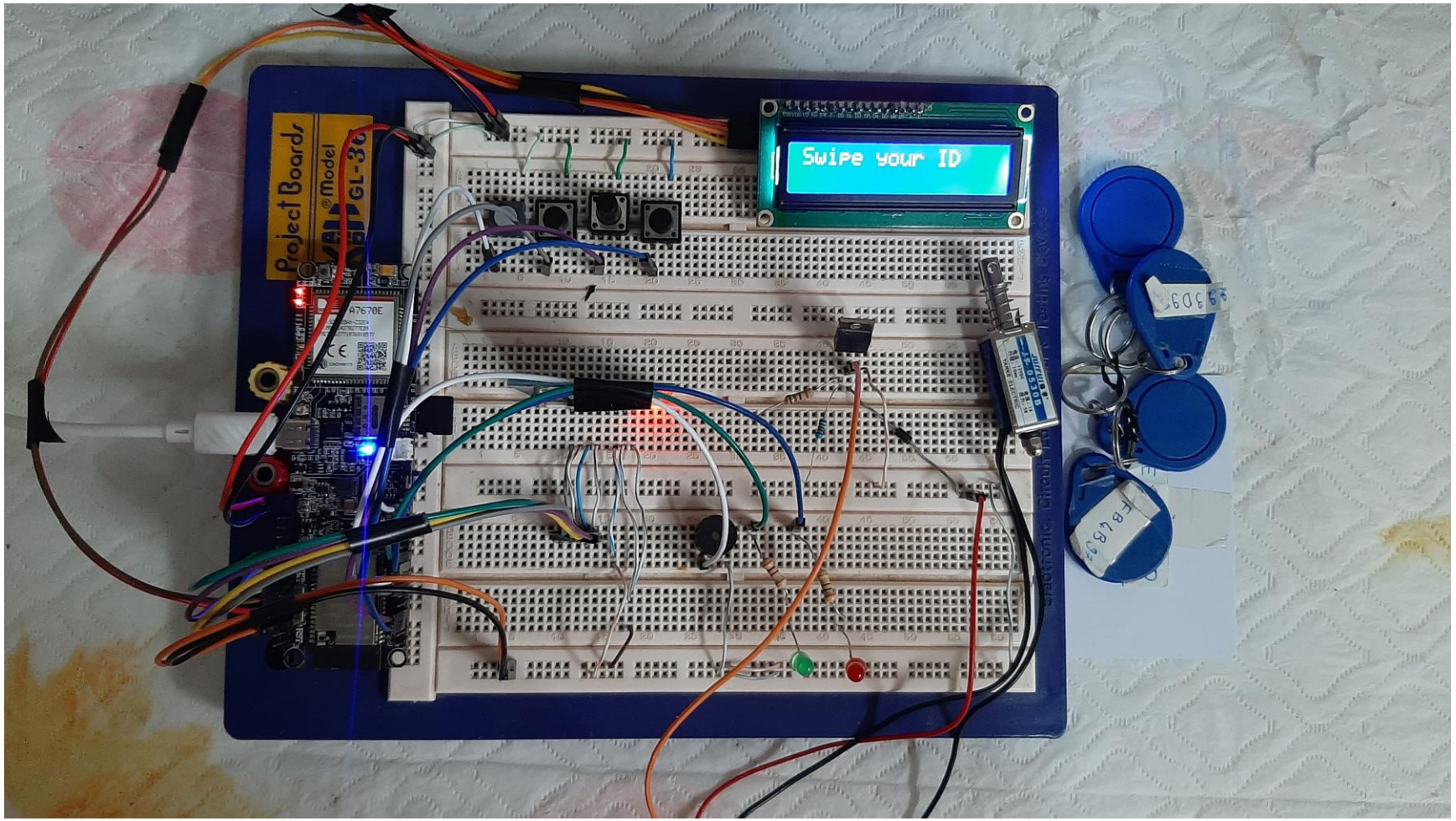
N°	Components	Model	Quantity
1	Development board	$LilyGO\ T-A7670$	**1**
2	RFID Reader	$MFRC522$	**1**
3	LCD display	$16x2\ I2C$	**1**
4	Buttons	/	**4**
5	Solenoid	$JF-0530B\ (12V/1A)$	**2**
6	Diode	$1N4007$	**1**
7	MOSFET	$IRL540N$	**1**
8	Buzzer	$Passive\ 5V$	**1**
9	LEDs	$Green/Red$	**2**
10	Micro SD	$16\ GB$	**1**
11	Resistors	$10k\Omega$	**1**
		220Ω	**2**
		150Ω	**1**
12	Capacitor	$100\ \mu F$	**1**
13	Jack adapter	$Female$	**1**
14	Power supply	$12V\ 1A$	**1**
15	USB adapter	Type c	**1**
16	Breadboard	$Large$	**1**

Appendix B

General Circuit Schematic (Fritzing)

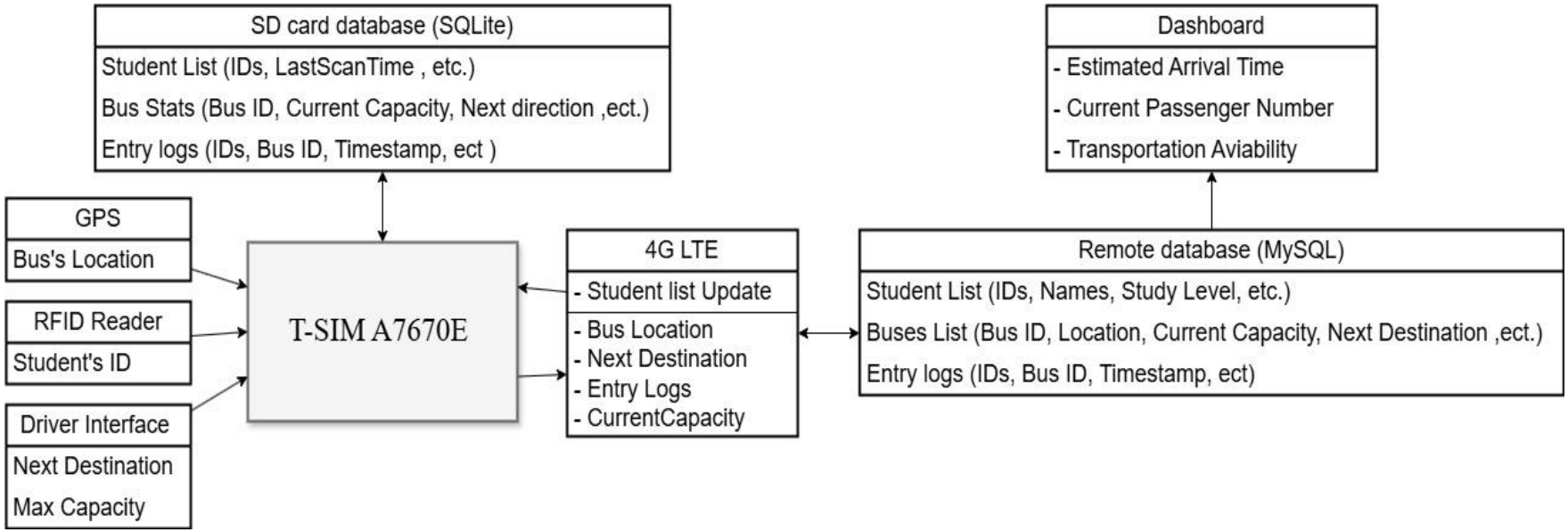


General System Circuit



Appendix C

the system general Data flow



Appendix D
ESP32 Main Code Snippet

```
void detectTag() {
  // Look for new RFID cards
  if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
    String uid = getRFIDUID();

    Serial.print("Detected UID: ");
    Serial.println(uid);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("UID:");
    lcd.setCursor(0, 1);
    lcd.print(uid);
    delay(500);

    if (checkAccess(uid)) {
      // Increment Curent capacity if access is valid
      incrementCurrentPassengers();
      // log entry in the local database
      logEntry(uid);
      Serial.println("ACCESS GRANTED");
      lcd.clear();
      lcd.print("Welcome!");
      accessAllowed();
      Serial.print("there are ");
      Serial.print(currentPassengers);
      Serial.println(" people in the bus");
      delay(500);
    } else {

      Serial.println("ACCESS DENIED");
      lcd.clear();
      lcd.print("ACCESS DENIED");
      accessDenied();
      delay(500);
    }
    sqlite3_close(db);
    chooseRFID();
    // Halt PICC
    rfid.PICC_HaltA();
    rfid.PCD_StopCrypto1();
    scanMessage = false;
  }
}
```

Appendix E

Key Libraries Used in ESP32 Code

Library	Purpose / Functionality
utilities.h (LILYGO)	Initialization and board configuration specific to the **T-SIM A7670E** module.
Wire.h	Enables I2C communication, used for interfacing with the I2C LCD screen.
LiquidCrystal_I2C.h	Controls the 16x2 I2C LCD display for the driver menu interface.
FS.h, SD.h, SPI.h	Provide access to the SD card module for optional local storage and data logging.
MFRC522.h	Allows communication with the RFID reader (MFRC522 module) via SPI.
sqlite3.h	Embedded SQL database library used for local data management on the ESP32.
WiFi.h	Handles Wi-Fi connectivity to test cloud/back-end communication.
HTTPClient.h	Allows the ESP32 to send HTTP POST requests to the Flask backend.
ArduinoJson.h	Serializes and parses JSON data for sending structured data via HTTP.

Appendix F

Serial Monitor Outputs:

Valid Access:

```
Detected UID: 223d981e
Opened /sd/LSDB.db successfully
Checking RFID: 223D981E
RFID found in database.
Bus Capacity is incremented
Fetched BusID: BUS01
Bus Capacity is updated successfully
Syncing payload:
{"BusID":"BUS01","PlateNumber":"123456789","MaxCapacity":5,"CurrentCa
pacity":1,"NextStationID":"A"}
Update sent successfully
Fetched BusID: BUS01
Entry logged successfully.
Unsynced logs count: 1
ACCESS GRANTED
there are 1 people in the bus
```

Invalid Access:

```
Detected UID: 8e85767d
Opened /sd/LSDB.db successfully
Checking RFID: 8E85767D
RFID not found in database.
ACCESS DENIED
```

Full Bus Capacity:

```
Detected UID: 63fb4b97
Opened /sd/LSDB.db successfully
Checking RFID: 63FB4B97
RFID not found in database.
BUS IS FULL
ACCESS DENIED
```

Syncing unsynced logs each 5 logs:

```
Detected UID: 63fb4b97
Opened /sd/LSDB.db successfully
Checking RFID: 63FB4B97
RFID found in database.
Bus Capacity is incremented
Fetched BusID: BUS01
Bus Capacity is updated successfully
Syncing payload:
{"BusID":"BUS01","PlateNumber":"123456789","MaxCapacity":10,"CurrentC
apacity":10,"NextStationID":"A"}
Update sent successfully
Fetched BusID: BUS01
Entry logged successfully.
Unsynced logs count: 5
Logs payload:
[
  {
    "RFID": "223D981E",
    "BusID": "BUS01",
    "ScanTime": "1970-01-01 00:00:39"
  },
  {
    "RFID": "223D981E",
    "BusID": "BUS01",
    "ScanTime": "1970-01-01 00:00:50"
  },
  {
    "RFID": "223D981E",
    "BusID": "BUS01",
    "ScanTime": "1970-01-01 00:04:05"
  },
  {
    "RFID": "223D981E",
    "BusID": "BUS01",
    "ScanTime": "1970-01-01 00:04:32"
  },
  {
    "RFID": "63FB4B97",
    "BusID": "BUS01",
    "ScanTime": "1970-01-01 00:05:16"
  }
]
Update sent successfully
Marked 5 logs as synced
Synced entries deleted successfully.
ACCESS GRANTED
there are 5 people in the bus
```

Appendix G1

Local Database Structure:

```
CREATE TABLE Stations (
    StationID TEXT PRIMARY KEY,
    StationName TEXT NOT NULL
);

CREATE TABLE Buses (
    BusID TEXT PRIMARY KEY,
    PlateNumber TEXT,
    MaxCapacity INTEGER NOT NULL,
    CurrentCapacity INTEGER DEFAULT 0,
    NextStationID TEXT,
    LastSyncTime DATETIME,
    FOREIGN KEY (NextStationID) REFERENCES Stations(StationID)
);

CREATE TABLE Students (
    StudentID TEXT PRIMARY KEY,
    RFID TEXT UNIQUE NOT NULL,
    AllowedAccess INTEGER DEFAULT 1,
    LastScanTime DATETIME
);

CREATE TABLE EntryLogs (
    LogID INTEGER PRIMARY KEY AUTOINCREMENT,
    RFID TEXT NOT NULL,
    BusID TEXT NOT NULL,
    ScanTime DATETIME DEFAULT CURRENT_TIMESTAMP,
    SyncedStatus INTEGER DEFAULT 0,
    FOREIGN KEY (RFID) REFERENCES Students(RFID),
    FOREIGN KEY (BusID) REFERENCES Buses(BusID)
);
```

Appendix G2

Remote Database Structure:

```
USE bus_tracking;

CREATE TABLE Stations (
    StationID VARCHAR(2) PRIMARY KEY,
    StationName VARCHAR(50) NOT NULL,
    INDEX (StationName)
) ENGINE=InnoDB;

CREATE TABLE Buses (
    BusID VARCHAR(10) PRIMARY KEY,
    PlateNumber INT NOT NULL,
    MaxCapacity INT NOT NULL,
    CurrentCapacity INT DEFAULT 0,
    NextStationID VARCHAR(2),
    LastSyncTime DATETIME,
    FOREIGN KEY (NextStationID) REFERENCES Stations(StationID) ON
DELETE SET NULL,
    INDEX (NextStationID)
) ENGINE=InnoDB;

CREATE TABLE Students (
    StudentID BIGINT PRIMARY KEY,
    FullName VARCHAR(100) NOT NULL,
    RFID VARCHAR(50) UNIQUE NOT NULL,
    Class VARCHAR(20),
    AllowedAccess BOOLEAN DEFAULT TRUE,
    ContactInfo TEXT,
    SyncedStatus BOOLEAN DEFAULT TRUE,
    INDEX (RFID)
) ENGINE=InnoDB;

CREATE TABLE EntryLogs (
    LogID INT PRIMARY KEY AUTO_INCREMENT,
    StudentID BIGINT NOT NULL,
    BusID VARCHAR(10) NOT NULL,
    StationID VARCHAR(2) NOT NULL,
    ScanTime DATETIME NOT NULL,
    SyncedStatus BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID) ON DELETE
CASCADE,
    FOREIGN KEY (BusID) REFERENCES Buses(BusID) ON DELETE CASCADE,
    FOREIGN KEY (StationID) REFERENCES Stations(StationID) ON DELETE
RESTRICT,
    INDEX (ScanTime),
    INDEX (SyncedStatus)
) ENGINE=InnoDB;
```

Appendix H
Backend Structure

Backend Technology Stack

- Language: Python
- Framework: Flask
- Database: MySQL hosted locally

Main Responsibilities of the Backend

- Receives HTTP POST requests from the in-bus system.
- The backend includes two main endpoints: the first updates the bus status data in the buses table, while the second logs RFID entry events in entrylogs table within MySQL database.
- send confirmation or error responses back to ESP32.

