Palestine Polytechnic University

College of IT and Computer Engineering

Computer System Engineering Department

Prepaid Parking System

BY Student names

Omar Srur

Bashar Shweiki

Waleed nasereddin

Supervisor Dr.khalid Tomizi

Submitted to the College of IT and Computer Engineering

in partial fulfillment of the requirements for the

Bachelor degree in Computer System Engineering

Hebron, May 2025

Abstract

Modern technology has brought significant developments to various sectors and the increasing number of vehicles, the need for smart and efficient parking management solutions has become essential to alleviate congestion and provide a comfortable experience for drivers.

The prepaid parking system consists of a set of integrated electronic components aimed at efficiently organizing the reservation and payment process. The system relies on a microcontroller linked to a set of sensors, a screen, and a camera. Payment is made via a coin acceptor or a mobile phone application, which allows the user to reserve a parking space for a specific period of time based on the amount paid. After completing the payment, a quick response (QR) code is displayed on the screen for the user to scan via the mobile phone application. The system also includes an ultrasonic sensor to detect the presence of a vehicle within the parking space.

Through the application, the user can reserve a parking space electronically using a Visa card. The application also allows scanning the QR code that appears after payment to activate the reservation. This allows the user to view the remaining parking time and view an interactive map, enabling the user to precisely locate available parking spaces. The application also includes an electronic lock control system for the parking space, allowing the user to open or close it remotely, in addition to a visual map that helps accurately locate the parking space within the area.

The system's website provides a dedicated control panel for administrators, enabling them to monitor the system's performance comprehensively and view detailed reports on the most frequently used parking spaces, along with the number of times each space was booked during the month.

Acknowledgement

We extend our sincere thanks and gratitude to everyone who contributed to the completion of this project and supported and guided us.

We especially thank our supervisor, Dr. Khalid Tomizi, for his valuable guidance and constructive comments, which significantly steered us in the right direction.

We also express our gratitude to Palestine Polytechnic University for providing the appropriate environment and resources that helped us implement the project optimally.

We also thank our colleagues and friends for their continuous support and encouragement throughout the project, as well as our generous families, who provided us with all the psychological and moral support.

Finally, we extend our thanks to everyone who contributed, directly or indirectly, to the success of this project and to everyone who had a hand in this achievement.

Table of Contents

List of Figures:

List of Tables:

CHAPTER ONE :Introduction

1.1 Preface

Technologies are transforming the world around us, including sectors that already seem to have clear and well-established procedures. Public parking services may seem straightforward enough, but a deeper look reveals lots of challenges for both drivers and parking operators, from the difficulty of finding an available parking spot in a popular location to issues involving convenient and secure payments.

The project aims to create a prepaid parking system that helps manage and organize parking spaces for users. The system relies on the use of advanced control devices, including a controller and sensors, to detect the presence of vehicles in front of the meter and manage parking spaces efficiently. Payment is made using a coin-accepting device, which in turn initiates parking time based on the paid currency. Users can find vacant parking spaces in real time through a mobile application, which opens an Open Street Map to view prepaid parking lots in a specific city, for example. Upon arrival, a map is opened, pinpointing the exact location of the meter so that the car can be parked at the appropriate vacant meter.

The app also allows users to check the status of their vehicles, see the time remaining on the meter, view the vehicle via a camera, and control the parking lock, which can be opened and closed through the app. The app also allows users to scan the QR code that appears on the screen after payment. Through the website, the administrator can obtain detailed reports on the most booked parking spaces, as well as the number of times each meter was booked during the month. These reports help improve parking management and provide better services to users, which contributes to regulating traffic and better meeting their needs.

1.2 Project Aims and Objectives

1.2.1Project Objectives:

Prepaid parking system has many objectives, including:

1. Organizing and managing parking lots, reducing chaos, and facilitating users' access to vacant parking spaces.
2. Empty parking spaces can be displayed before arriving at them via the phone application map, which saves time and effort for users.
3. Vehicles can be monitored remotely via the camera installed on the parking .
4. The presence of the vehicle in front of the parking is verified through an ultrasonic sensor.
5. The administrator can obtain detailed reports on the most booked parking spaces.
6. The ability to view a open Street Map to display a list of prepaid parking .
7. Identify coins using a coin acceptor.
8. View detailed reports on the most booked spaces during the month.
9. Reservations can be made via the app by entering Visa card details.

1.2.2Project aim :

1. Reduce parking congestion by knowing in advance about empty parking spaces.
2. Through the prepaid process a more convenient and easy to use parking system is created.
3. The project provides an effective tool for managing parking spaces as the reports and analyses provided to officials help make better decisions.
4. Raising the level of confidence and security among users thanks to camera follow up and time out alerts.
5. The system helps raise the level of progress and development of countries.
6. Reports help improve parking management and provide better services to users.

1.3 Problem Statement

Many cities suffer from the problem of finding empty parking spaces which leads to users wasting time and effort while searching for parking spaces. In addition traditional parking systems lack the ability to manage and organize the movement of cars inside the parking spaces which leads to additional congestion and delays in parking operations as well as the lack of an easy and safe mechanism to monitor the condition of vehicles after parking, which increases users' concern about their vehicles and their commitment to the duration of parking inside the paid parking space.

Traditional methods rely mainly on manual work or unconnected systems, which leads to a lack of accurate data that helps in making better decisions to manage parking spaces.

The idea of the prepaid parking system came to solve these problems by using modern technology in organizing and managing parking spaces which contributes to facilitating the process for users and providing a more efficient and safe service.

1.4 Project Requirements

1.4.1 Functional Requirements:

- Detect the presence of a vehicle in the parking lot using ultrasonic sensors and update the parking status immediately.
- Possibility of viewing parking spaces through the mobile application map and viewing the status of parking spaces if they are available or unavailable.
- The app also allows users to check the remaining parking time.
- Controls the lock through the app, allowing it to be opened and closed manually.
- The ability to scan the QR code on the screen in the parking lot to view the car through the camera that opens after scanning the code
- Detailed reports on the most booked parking spaces during the month.

1.4.2 NON -Functional Requirements:

- Security: Protect data from any unauthorized access.
- Ease of use: The application should be easy to use and navigate.
- System reliability: The system is able to operate without interruption for long periods as the parking status is updated instantly.
- Performance: Quickly detect the status of available and unavailable parking spaces and accordingly the application and the management panel are updated in real time.
- Scalability: It can be expanded to include more parking spaces or different locations when needed.

1.5 System description

The prepaid parking system consists of several interconnected components that manage parking meters. The system relies on a controller linked to a number of sensors, including a coin acceptor through which payments are made, an ultrasonic sensor to detect the presence of vehicles within the parking space, and a screen to display the QR code that appears after the payment process. Additionally, the system is linked to a mobile application, which features several features, including electronic reservations using a Visa card, a map displaying parking spaces within the city, and the ability to view the exact parking space using a map. The application supports displaying the area, position, and node of parking spaces. Additionally, manual lock control is available via the application. The camera can be opened to view the vehicle, confirm its presence, and determine the remaining parking time. The system is supported by a website for system management, which displays detailed reports on the most booked spaces throughout the month. This data helps improve resource allocation and make better organizational decisions.

1.6 Project Limitations/Constraints

Here are some limitations of the prepaid parking management system:

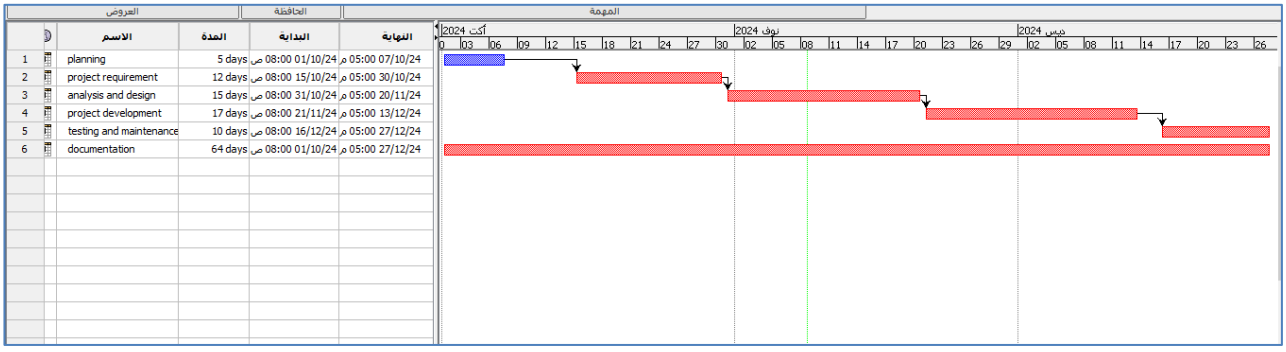
1. Restrictions on technical support for the project at all times.
2. Sensor reading errors, i.e. poor accuracy of the sensors used, which leads to system weakness.
3. Any power outage can lead to service disruption.
4. Users without a mobile internet connection or mobile app may not fully benefit from features such as electronic payment or vehicle monitoring.
5. Exposing devices to weather-related issues such as rain, extreme temperatures, or dust, which may impact device performance.

1.7 Project Schedule

Table1. 1Project Schedule

	الاسم	المدة	البداية	النهاية
1	planning	5 days	08:00 01/10/24 ص	05:00 07/10/24 م
2	project requirement	12 days	08:00 15/10/24 ص	05:00 30/10/24 م
3	analysis and design	15 days	08:00 31/10/24 ص	05:00 20/11/24 م
4	project development	17 days	08:00 21/11/24 ص	05:00 13/12/24 م
5	testing and maintenance	10 days	08:00 16/12/24 ص	05:00 27/12/24 م
6	documentation	64 days	08:00 01/10/24 ص	05:00 27/12/24 م

Table1. 2Project Gannet



CHAPTER TWO: Theoretical Background

2.1 Preface

This chapter provides a short review of some related systems and introduces a theoretical background of the project, including a description of the hardware and software components used in the system.

2.2 Theoretical Background

The theory of the prepaid parking management system aims to integrate different technologies that solve many challenges related to parking management including space discovery, payment processing, and data communication. The main theoretical concepts and technologies that form the basis of this project are as follows:

1. Generate a Quick Response (QR) code that acts as a digital ticket that the user scans using the mobile app to confirm the reservation and track the remaining parking time. This code appears on the screen once payment is made.
2. Open Street Map : by integrating Open Street Map into the parking application, users can view a real-time, interactive map showing the precise locations of smart parking slots.
3. Web and mobile application development: where the system relies on an application for users and a web interface for administrators where the application provides real-time updates on available parking spaces, time management and vehicle monitoring. The web interface allows administrators to manage data.
4. Embedded systems: where ESP32 and ESP32-CAM modules act as embedded systems designed to control and communicate with peripheral devices such as ultrasonic sensors, coin acceptor ,lock and cameras.

2.3 Literature Review

2.3.1 RFID and HDL Based Pre-Paid car Parking System

In today's era one of the most common problems which the world is facing is an exponential increase in population. This has indirectly increased a lot of other issues; one of them being the quantity of automobiles on the road. The increased number of vehicles have resulted in heavy traffic and shortage of parking space which gives rise to illegal parking which itself is a threat to the security of cars. The main aim of this paper is to study different kinds of car parking systems and come up with an Intelligent Prepaid Car Parking system which addresses the problem of parking space, vacancy issues together with the safety of cars. It utilizes the perks of RFID (Radio Frequency Identification) using VHDL. It decreases manual work and makes the system more efficient. The design has been tested and simulated on Xilinx Vivado 15.4 Software tool.[1]

2.3.2 Support for Self-service Automated Parking Systems

Due to the increasing numbers of cars in cities, organizations of parking zones and fees are crucial to the efficient functioning of traffic in cities. For most urban areas, the number of parking spaces, especially on-street parking, is usually not enough. On-street parking is the most common form of parking in cities. There are several ways for on-street parking payments such as special parking vouchers, text messages via mobile phones, special smartphone applications, self-service parking machines, and parking meters. Over 95% of the on-street parking in the world is paid by parking meters and self-service parking machines. In order for constant improvement of the parking services, one approach is by extending the functionality of self-service parking machines with the introduction of new technologies. This paper describes the support for on-street parking payment systems, by introducing new parking kiosk (self-service parking machine) that can be easily integrated with other parking payment systems in the cities.[2]

2.3.3 Streamline Your Drive: Pre-Booking & Pre-Paid Parking slots, Locating CNG Stations & EV Charging Stations.

Urbanization has led to the increase in Commercial places frequently visited by people. Having so many options for travelling, humans usually prefer their own vehicles when commuting to these areas. Due to the availability of limited parking slots in malls and public places, parking of vehicles is one of the major concerns in the existing scenario. Also, transformation from petroleum-based vehicles to ecofriendly Electrical and CNG vehicles demands dynamic location of EV Charging points and CNG stations. The need of the hour is to develop an application to streamline the parking and location of nearby charging stations. Streamline your drive revolutionizes urban mobility by helping users effortlessly locate nearby parking spaces including pre-booking and prepaid options for both free and paid spots in crowded public spaces.[3]

The importance of the project compared to previous studies:

All previous studies focused on the importance of parking spaces or organizing parking spaces without adding features that enhance the efficiency and comfort of the parking process and work to provide real-time alerts for the remaining parking time, or monitor vehicles directly through cameras, or integrate advanced sensors to detect the presence of vehicles with high accuracy. In addition, previous systems do not feature an electronic payment system and rely on traditional kiosks or RFID-based solutions, which are primitive systems that do not accept development and are easy to use. Therefore, this prepaid parking project was developed to address these limitations by creating a comprehensive system that organizes parking spaces and also provides real-time updates, vehicle monitoring, and push notifications. And send all information directly to the mobile application for people to work on managing parking spaces properly.

2.4 Hardware System Components

This section describes all hardware used in our project, it presents a figure for each one with short description about its work principle and why it is used in the system.

ESP32 microcontroller

ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor or a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules..It is a successor to the ESP32 microcontroller.

Features of the ESP32 include the following:

1. CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor operating at 160 or 240 MHz and performing at up to 600 DMIPS.
2. Ultra low power (ULP) co-processor.
3. Memory: 320 KiB RAM, 448 KiB ROM.
4. Wireless connectivity.
5. Wi-Fi: 802.11 b/g/n.

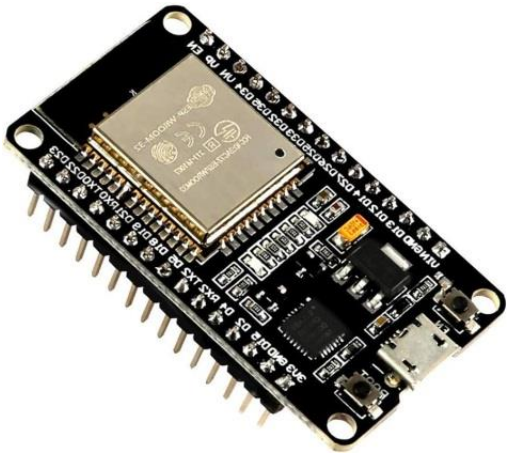


Figure2. 1ESP32 Microcontroller[4]

Coin Acceptor

A coin acceptor is an essential tool used in prepaid machines such as vending machines, arcades, public laundromats, and more. The device is capable of recognizing coins based on characteristics such as size, weight, metal material, and inscriptions, and can be programmed to accept only certain types of coins. When a coin is inserted, the device carefully scans it using advanced sensors, and if it does not meet the programmed criteria, it is automatically rejected and returned to the user via the coin rejection strip. The device sends precise electrical signals to the main electronic system of the device to perform the required operations, such as turning on the device or starting the service. It is characterized by its simple design and easy installation, as it contains mounting accessories such as screws and nuts, in addition to its support for integration with various control systems such as Arduino or Raspberry Pi. The device is also made of durable, corrosion-resistant materials, making it suitable for use in various environments and supports a long service life.

Common uses:

1. Vending machines (selling drinks or snacks).
2. Arcade games.
3. Charging stations for electronic devices.
4. Public laundry facilities.
5. Smart parking devices.

Figure2. 2Coin Acceptor[5]

Ultrasonic sensors

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e. the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target). In order to calculate the distance between the sensor and the object, the sensor measures the time it takes between the emission of the sound by the transmitter to its contact with the receiver. The formula for this calculation is D = ½ T x C (where D is the distance, T is the time, and C is the speed of sound ~ 343 meters/second).

Ultrasonic Sensor Features

1. Distance Measurement: Uses high-frequency sound waves.
2. Wide Range: From a few centimeters to several meters.
3. Non-contact Measurement: Ideal for sensitive materials.
4. Weatherproof: Operates in harsh environments.
5. Easy Integration: Supports interfaces like Analog, I2C, UART.
6. Low Power Consumption: Suitable for portable devices.
7. Versatile Applications: Robotics, vehicles, industry.

Figure2. 3Ultrasonic Sensors[6]

ESP-CAM

The ESP32-CAM is a very small camera module with the ESP32-S chip that costs approximately $10. Besides the OV2640 camera, and several GPIOs to connect peripherals, it also features a microSD card slot that can be useful to store images taken with the camera or to store files to serve to clients. The ESP32-CAM doesn’t come with a USB connector, so you need an FTDI programmer to upload code through the U0R and U0T pins (serial pins).

Here is a list with the ESP32-CAM features:

1. The smallest 802.11b/g/n Wi-Fi BT SoC module
2. Low power 32-bit CPU ,can also serve the application processor
3. Up to 160MHz clock speed, summary computing power up to 600 DMIPS
4. Built-in 520 KB SRAM, external 4MPSRAM
5. Supports UART/SPI/I2C/PWM/ADC/DAC
6. Support OV2640 and OV7670 cameras, built-in flash lamp
7. Support image Wi-Fi upload
8. Support TF card
9. Supports multiple sleep modes
10. Embedded Lwip and FreeRTOS

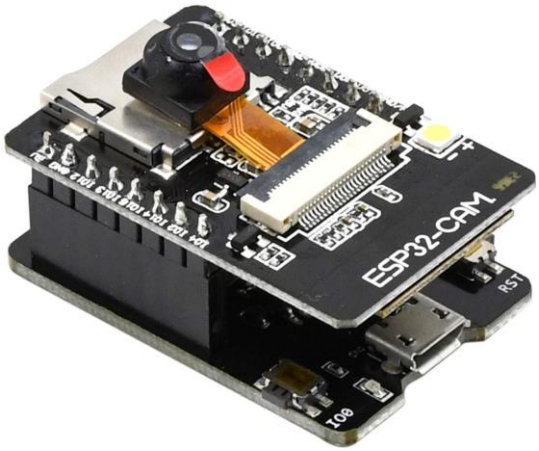


Figure2. 4ESP-CAM[7]

DC motor

A DC motor is an electrical motor that uses direct current (DC) to produce mechanical force. The most common types rely on magnetic forces produced by currents in the coils. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor.

DC motors were the first form of motors to be widely used, as they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor, a lightweight brushed motor used for portable power tools and appliances can operate on direct current and alternating current. Larger DC motors are currently used in propulsion of electric vehicles, elevator and hoists, and in drives for steel rolling mills. The advent of power electronics has made replacement of DC motors with AC motors possible in many applications.

Figure2. 5 DC Motor[8]

L298N DC Motor Driver Module

This L298N Motor Driver Module is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control.

Features & Specifications

1. Driver Model: L298N 2A
2. Driver Chip: Double H Bridge L298N
3. Motor Supply Voltage (Maximum): 46V
4. Motor Supply Current (Maximum): 2A
5. Logic Voltage: 5V
6. Driver Voltage: 5-35V
7. Driver Current:2A
8. Logical Current:0-36mA
9. Maximum Power (W): 25W
10. Current Sense for each motor
11. Heatsink for better performance
12. Power-On LED indicator

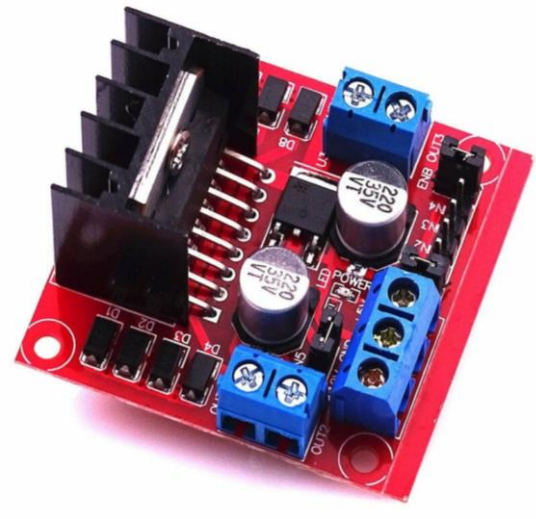


Figure2. 6 L298N Motor Driver[9]

Table 2.1 as show the difference between Raspberry Pi VS Arduino VS ESP32 :

Comparing ESP32 and Arduino and Raspberry Pi involves considering their features and applications and use cases. Each of these platforms has strengths and weaknesses making them suitable for different type project. compare them based on various criteria to determine which one might be best choice for the project :

Table2. 1Differrnce between RPi And Arduino And ESP32

Component	Arduino	Raspberry Pi	ESP32
Ease of Use	beginners and for quick prototyping.	requires some basic knowledge of Linux and programming	especially for IoT projects. relatively easy to use.
Processing Power	limited processing power	dual-core processor	single-board computer
Connectivity	limited connectivity options.	built-in Wi-Fi and Bluetooth capabilities	Ethernet, USB, Wi-Fi ,Bluetooth connectivity
Cost	10-20$	40-55$	20-25$
Speed	16MHz.	1.4GHz.	240MHz.
Connectivity	I2C UART GPIO	DSI UART IOCSI GPIO	RT, GPIO
st choose	x	x	✓
ge			

- In our project we will use the ESP32 controller because it meets the project requirements that we need.

Table 2.2 as show the difference between Genera Ultrasonic sensor VS A02YYUW ultrasonic sensor :

Table2. 2 Differrnce Between Ultrasonic And A02YYUW Ultrasonic Sensor

Feature	**eneral Ultrasonic Sensor**	**2YYUW Ultrasonic Sensor**
Waterproofing	Usually not waterproof	Fully waterproof
Range	Varies (2cm–4m typical)	20mm–4500mm (4.5m)
Accuracy	Moderate, varies with model.	±5mm
Output Signal	Basic (PWM or analog)	UART (digital communication)
Price	Generally lower	Slightly higher
plications	oor, non-harsh ironments	tdoor, industrial, and harsh ironments
st choose	✓	x
age		

- Ultrasonic sensor is the best choice for prepaid parking system project due to its affordable price and accuracy which makes it convenient to use and easy to integrate with smart systems to ensure reliable performance.

2.5 System Software (Website)

Websites play a vital role in the modern digital world allowing access to a global audience without any restrictions and providing continuous presence at all times which ensures the provision of services and information at any time. In addition websites are an effective tool for many projects so we will work on designing a website for a prepaid parking system where detailed reports are displayed on the most reserved parking spaces and the number of times each meter was reserved during the month, which helps improve parking management and provide better services to users and contributes to organizing traffic and meeting the needs of users better. Reservations are also displayed to users, and zones are managed. This is all done through several interfaces, including a login interface where a username and password are entered. All of this is done through several interfaces, including the login interface, through which the username and password are entered. This data is completely encrypted using the SHA-256 algorithm which is a cryptographic hash function that creates a hash value with a fixed size of 256 bits . The purpose of the SHA-256 algorithm is to create a unique digital fingerprint for a piece of data such as a password. SHA-256 refers to a cryptographic hash algorithm where the input data is processed through a complex mathematical function resulting in a unique output hash. This hash acts as digital fingerprint that uniquely represents the original data. This is to ensure the security and protection of data on the site as each person's data is protected from hacking.

In addition to the internal data that is transferred from the hardware part to the website which is protected through SSL/TLS (Secure Sockets Layer) is a security protocol used to create an encrypted and secure connection between the browser such as Chrome or Firefox and server that hosts the website as this protocol aims to protect data exchanged between the two parties from being hacked or intercepted while it is transmitted over the Internet.

2.6 System Software (Mobile application)

The importance of obtaining mobile applications has increased significantly with the digital development of smartphones and the increase in the number of users of these phones, so obtaining applications has become an important means for companies and monitoring systems in addition to immediate access to data and receiving and sending notifications and alerts. Applications also have the ability to connect to cloud systems to collect and store data, so we worked on developing a smartphone application in the language of filters to monitor vehicles in the prepaid parking system.

The application helps users find empty parking spaces in real time by entering the mobile application, where a map is opened on Open Street Map to know the prepaid parking lots in a specific city, for example, and upon arrival, the municipality map is opened to determine the exact location of the meter so that the car can be parked in the appropriate empty meter, and it is possible to navigate from one map to another through the application interfaces. The app also allows users to check their vehicles, see the time remaining on the meter, and book using a Visa card. The app allows users to enter data and book, in addition to controlling the lock, which can be opened and closed manually. The app also allows users to open the camera and view the vehicle with ease.

Security features have been added to the application to ensure data protection from hacking, as data is transferred and protected through the TLS/SSL protocol, which protects data when moving between systems over networks, especially on the Internet. TLS is an evolution of an earlier protocol known as SSL (Secure Sockets Layer), and is now the primary standard for securing communications.

2.7 System Software Component

This section will provide some information about the main programs used in our project.

2.7.1 Arduino IDE

Arduino code is written in C+ with an addition of special methods and functions. C++ is a human-readable programming language. The Arduino Integrated Development Environment (IDE) is the main text editing program used for Arduino programming. It is where you'll be typing up your code before uploading it to the board you want to program. Arduino code is referred to as sketches, It is processed and compiled to machine language, and Arduino sketch code is used to program wemos microcontrollers.[10]

2.7.2 Flutter language

Flutter is an open-source UI software development kit developed by Google. The features of Flutter allows Flutter app development services to create cross-platform applications from a single codebase for web browser, Fuschia, Android, iOS, Linux, macOS, and Windows.

The initial version of Flutter was known as “Sky”, unveiled in 2015 at a DART developer’s summit. In 2018, its actual version 1 was released for use. Right now, it is one of the biggest rivals of React Native, which is backed by Facebook, now known as Meta.[11]

Flutter has a wide range of applications that enables Flutter app development companies to:

- Create mobile apps for both the operating system, i.e. Android and iOS
- Develop web apps that can run on any browser
- Develop desktop applications for Windows, mac OS, and Linux
- Develop apps for embedded systems such as smart TVs and wearables

2.7.3 Fritizing

It is an open source initiative to support designers, artists, researchers and hobbyists to take the step from physical prototypes to actual product. We make this software in the spirit of processing and Arduino, developing a tool that allows users to document their Arduino and other electronic models, and create a PCB layout for manufacturing. The complementary website helps users share and discuss drafts and experiences as well as reduce manufacturing costs. Fritizing is essentially electronic design automation software with a low barrier to entry, suitable for the needs of designers and artists. It uses the metaphor of a breadboard, so that it is easy to transfer your hardware drawing to the software. From there, it is possible to create PCB layouts to turn them into a robust PCB on your own or with the help of a manufacturer.[12]

2.7.4 XAMPP

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server possible. XAMPP is regularly updated to the latest releases of Apache, MariaDB, PHP and Perl. It also comes with a number of other modules, including OpenSSL, phpMyAdmin, MediaWiki, Joomla, WordPress and more .Self-contained, multiple instances of XAMPP can exist on a single computer, and any given instance can be copied from one computer to another. XAMPP is offered in both a full and a standard version (Smaller version).[13]

2.7.5 OpenStreetMap

OpenStreetMap (abbreviated OSM) is a free, open map database updated and maintained by a community of volunteers via open collaboration. Contributors collect data from surveys, trace from aerial photo imagery or satellite imagery, and import from other freely licensed geodata sources. OpenStreetMap is freely licensed under the Open Database License and is commonly used to make electronic maps, inform turn-by-turn navigation, and assist in humanitarian aid and data visualisation. OpenStreetMap uses its own data model to store geographical features which can then be exported into other GIS file formats. The OpenStreetMap website itself is an online map, geodata search engine, and editor.[14]

CHAPTER THREE: SYSTEM DESIGN

3.1 Preface

In this chapter, we will delve into the design prepaid parking system. We will explore different design options, provide a conceptual description of the system, discuss the algorithms and methodologies used, and present schematic diagrams to illustrate the system's components and connections. This chapter serves as a comprehensive guide to the design phase of the project.

3.2 Block diagram

The figure 3.1 represents the block diagram of a prepaid parking system, which shows the overall structure of the project by connecting the ESP32 microcontroller to the ultrasonic sensors, coin reader, LN298 driver, and camera. It shows the relationships and interactions between the system components, as shown in the inputs and outputs of the system. This helps in understanding the organizational framework of the project and how each part integrates with the others.

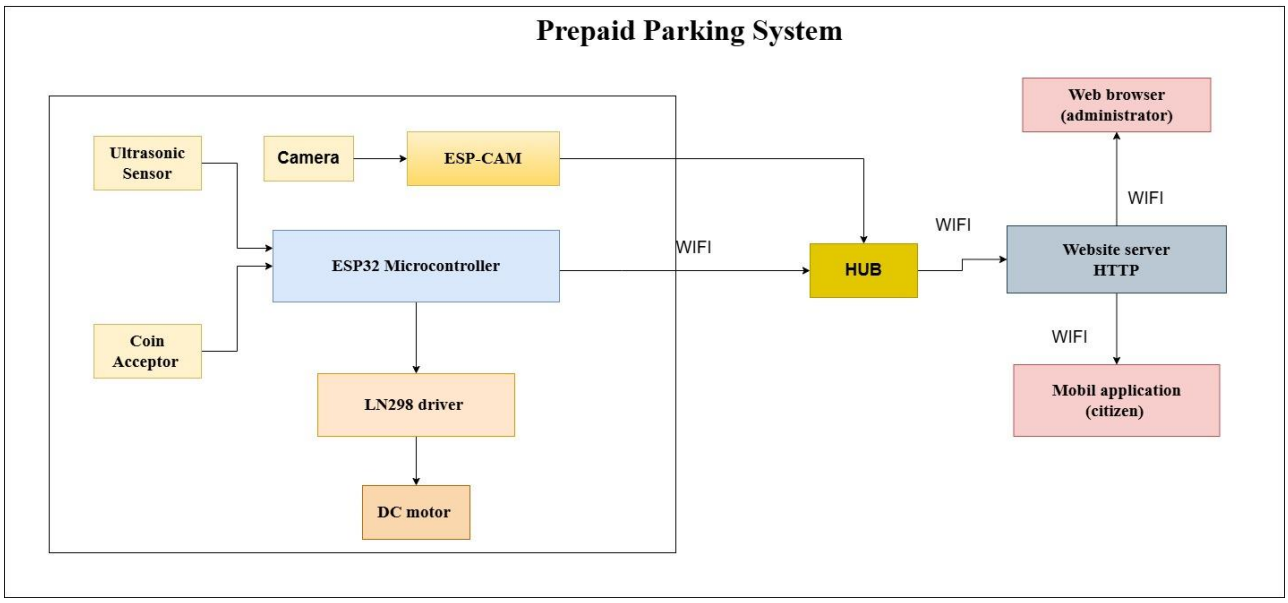


Figure3. 1Block Diagran

3.3 Flowchart

Figure 3.2 represents a flowchart of the application. It displays a set of boxes, including "Electronic Booking," which allows users to select a location and navigate to the map or plan view interface. Alternatively, "Book Now," which allows users to enter data and complete the booking. Alternatively, users can select a box to scan the QR code, open the camera to view the vehicle, or control the lock, either opening or closing it. Additionally, the time remaining for the person in the parking space is displayed.

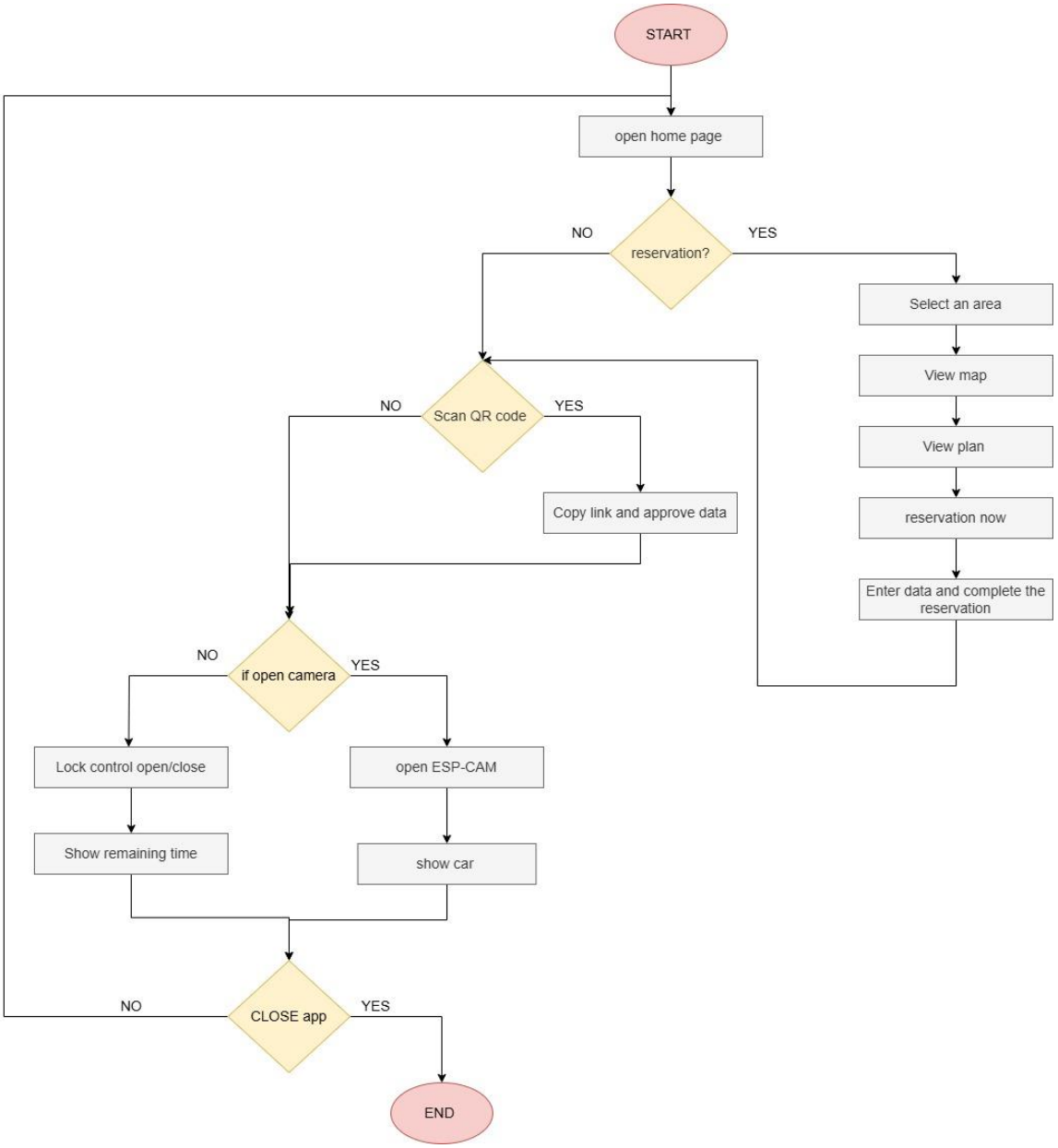


Figure3. 2Flowchart

The following figure 3.3 represents the system's flow chart, showing how to park a vehicle in the parking lot. The presence of a vehicle is detected and the amount is verified to have been paid into the device. Then a parking space is reserved, a QR is displayed, the timer is started, and the presence of the vehicle is verified. If payment is not made, refer to confirm sensor.

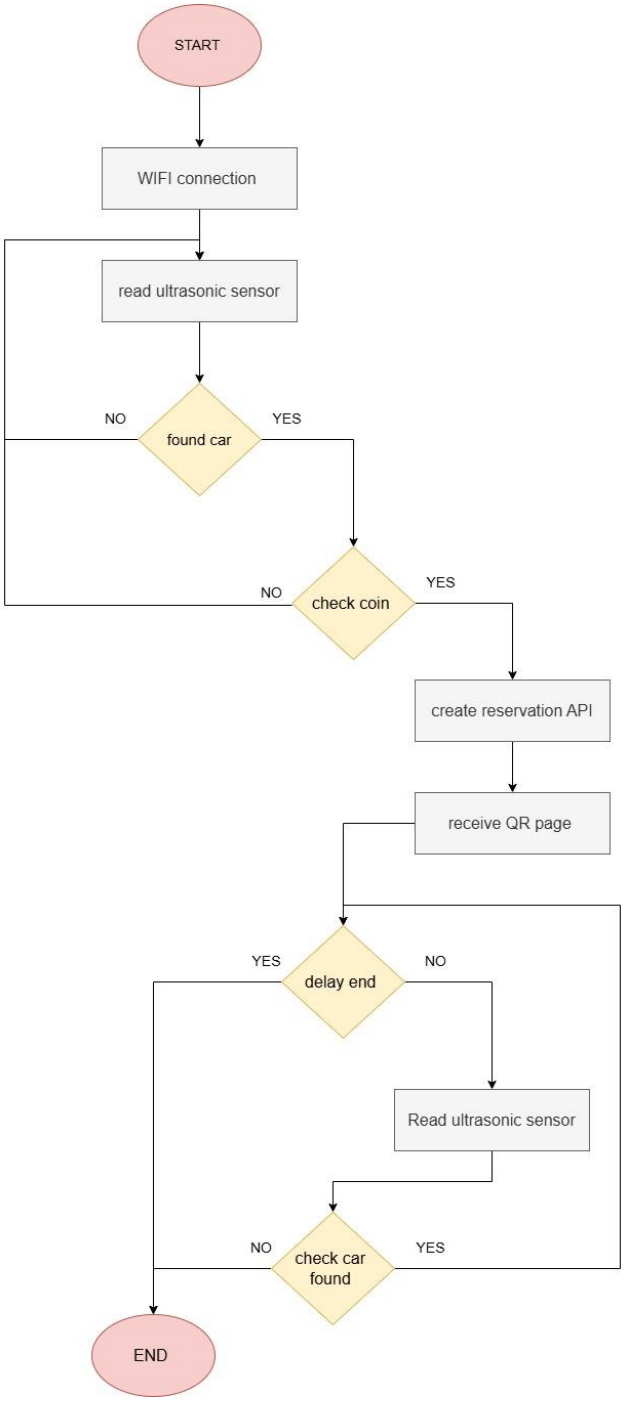


Figure3. 3Flowchart for Hardware

3.4 Physical layout diagram

The following figure 3.4 illustrates the physical design diagram which is an essential part of designing a prepaid parking system because it shows the physical arrangement of the various components of the system and how these components interact with each other. The following figure is a vital tool for understanding the physical architecture of a prepaid parking system as it shows what the components used in the project would look like in their actual form.

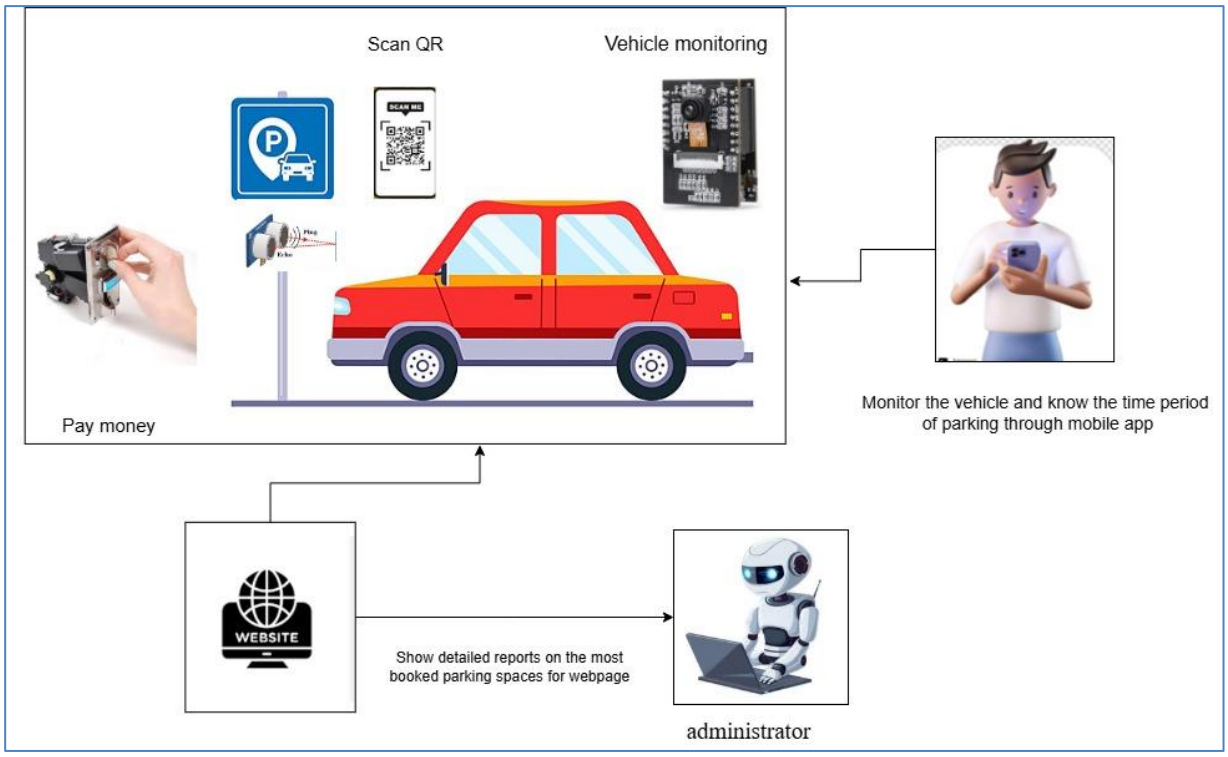


Figure3. 4Physical Layout Diagram

3.5 Architecture diagram (layer architecture)

The following figure 3.5 shows the Layered Architecture of the software system used in the project, where the system depends on dividing tasks into different levels as shown:

1. User Interface Layer: This layer shows the interaction between the user and the system, for example, accessing the website is through browser interfaces to open the site.
2. Service Layer This layer consists of two parts:
    a. Service Layer 1 (for public sessions) provides public services that can be accessed by all users, i.e. in its system, it represents those responsible for the site who work on displaying reports and the rest of the data.
    b. Service Layer 2 (for private sessions): It represents the secure layer for data, i.e. protecting it when moving from one party to another using the data protection system.

3. Database Layer (Database Layer): Dealing with data stored in the system.

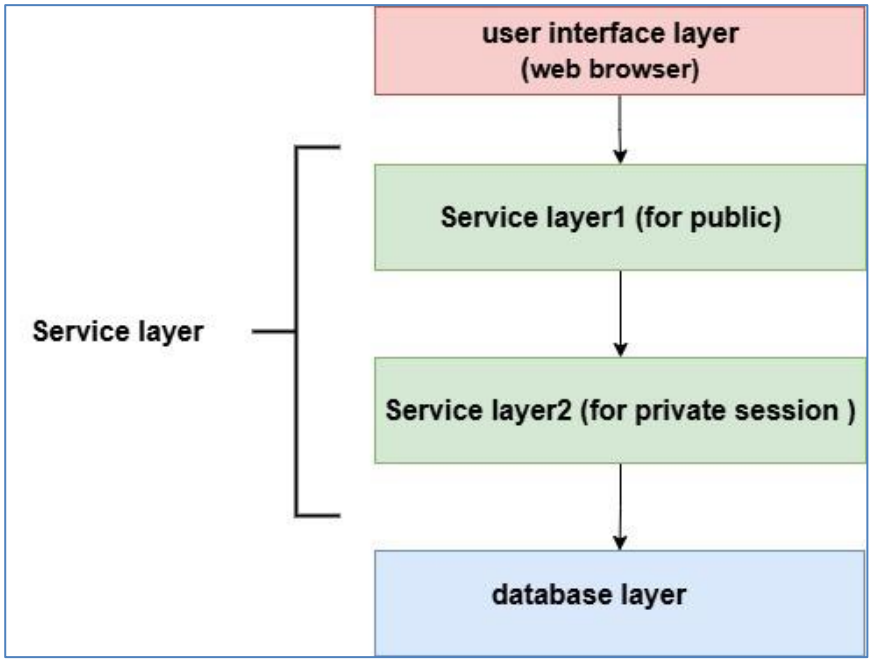


Figure3. 5Architecture Diagram

3.6 ERD diagram

Figure 3.6 illustrates an ERD, short for Entity-Relationship Diagram, which is a diagram used to design and illustrate the database structure of a prepaid parking system. It illustrates the entities (tables, such as admin, reservation, devicenode, positions ,area , log_user_info), attributes (columns for each table, such as name, date, password), and relationships (links between tables, such as each user can make multiple reservations).

This diagram helps design the database logically and systematically, helps programmers and designers build accurate databases, and also documents the database to facilitate development and maintenance.

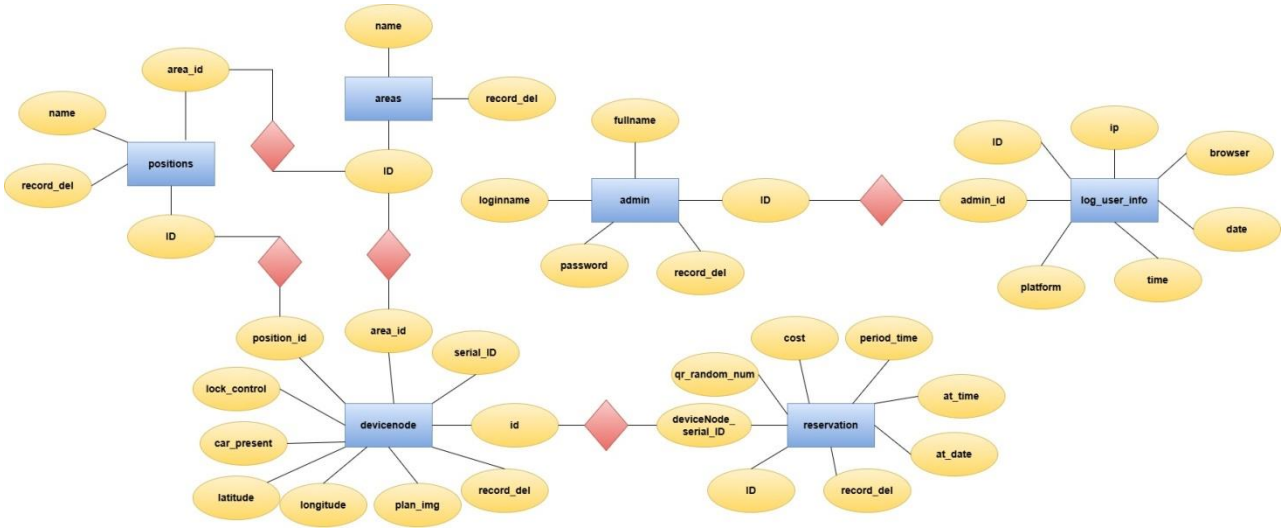


Figure3. 6ERD Diagram

3.7 Security diagram

A Security Diagram helps identify the security components of a system such as users and devices and SSL/TLS protocols and shows the relationships between them. The diagram helps in detecting security vulnerabilities managing access and privileges and ensuring compliance with security standards. It also helps in analyzing risks and developing effective strategies to counter attacks which enhances the protection of data and system infrastructure as show in the figure 3.7.

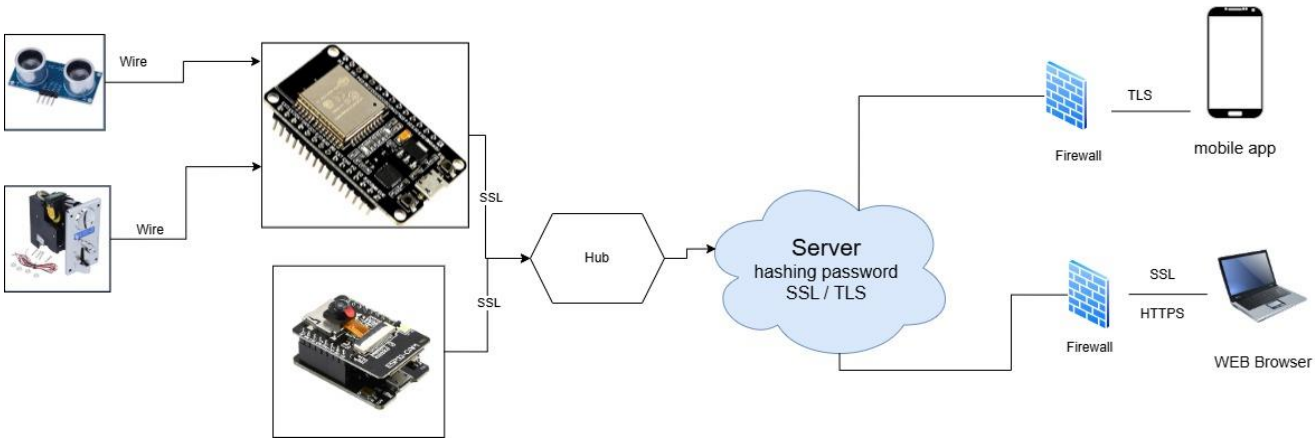


Figure3. 7Security Diagram

3.8 Sequence diagram

The following figure 3.8 represents the sequential diagram of the process of logging into the site by entering the username and password. The entered data is verified and checked in the database. If the data is incorrect, an error message is given, and if it is correct, a session is opened to enter the main page.

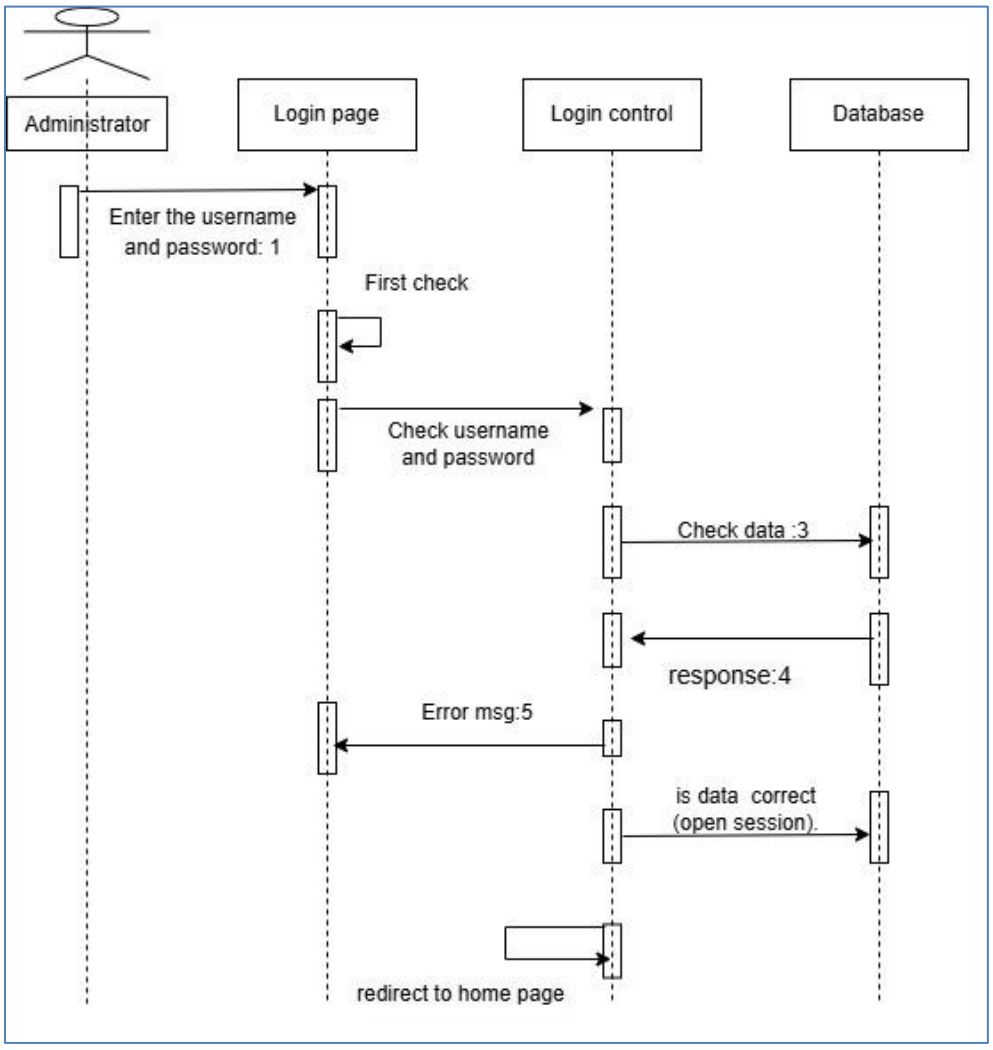


Figure3. 8Sequence Diagram

3.9 System Wiring Diagrams

The system wiring diagrams offer a detailed view of the physical connections between components. This includes the intricate network of wires connecting sensors and microcontrollers and communication modules. The wiring diagrams provide a practical guide for the assembly and maintenance of the device, ensuring a clear understanding of the hardware layout. In this section some components are shown with their connection to the ESP32. The pins of each equipment will be physically connected as shown in the following figures.

- ESP32 connect with ultrasonic sensor :

Table3. 1ESP32 Connect with Ultrasonic

ESP32 microcontroller	Ultrasonic sensor
VCC	VCC
GND	GND
GPIO27	TRIG
GPIO26	ECHO

- Connection wiring diagram between the ESP32 controller and a ultrasonic sensor:

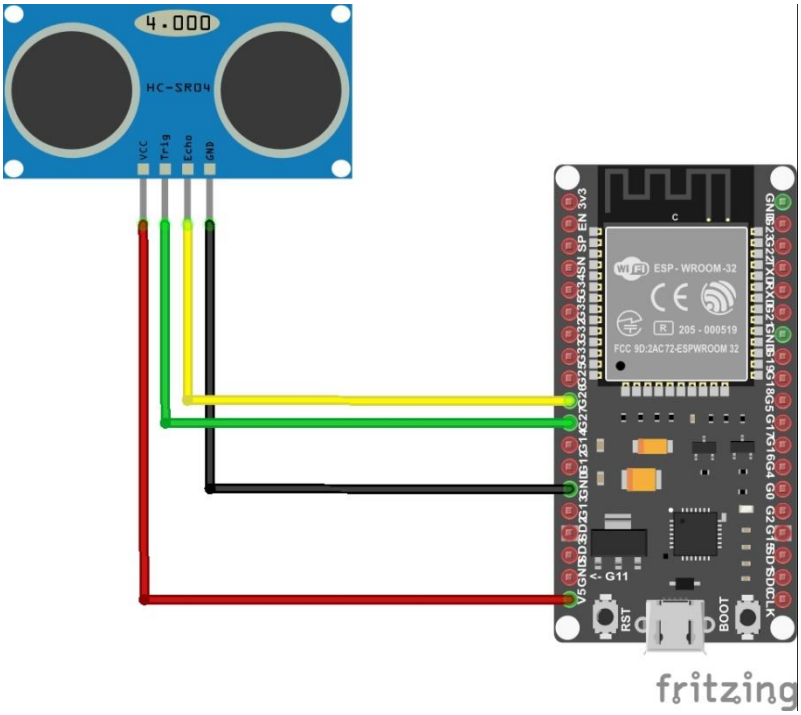


Figure3. 9Wiring diagram between ESP32& Ultrasonic

- ESP32 connect with Coin acceptor :

Table3. 2ESP32 Connect with Coin Acceptor

ESP32 microcontroller	Coin acceptor
VCC	12V
GND	GND
GPIO19	SIG

- Connection wiring diagram between the ESP32 controller and a Coin acceptor:

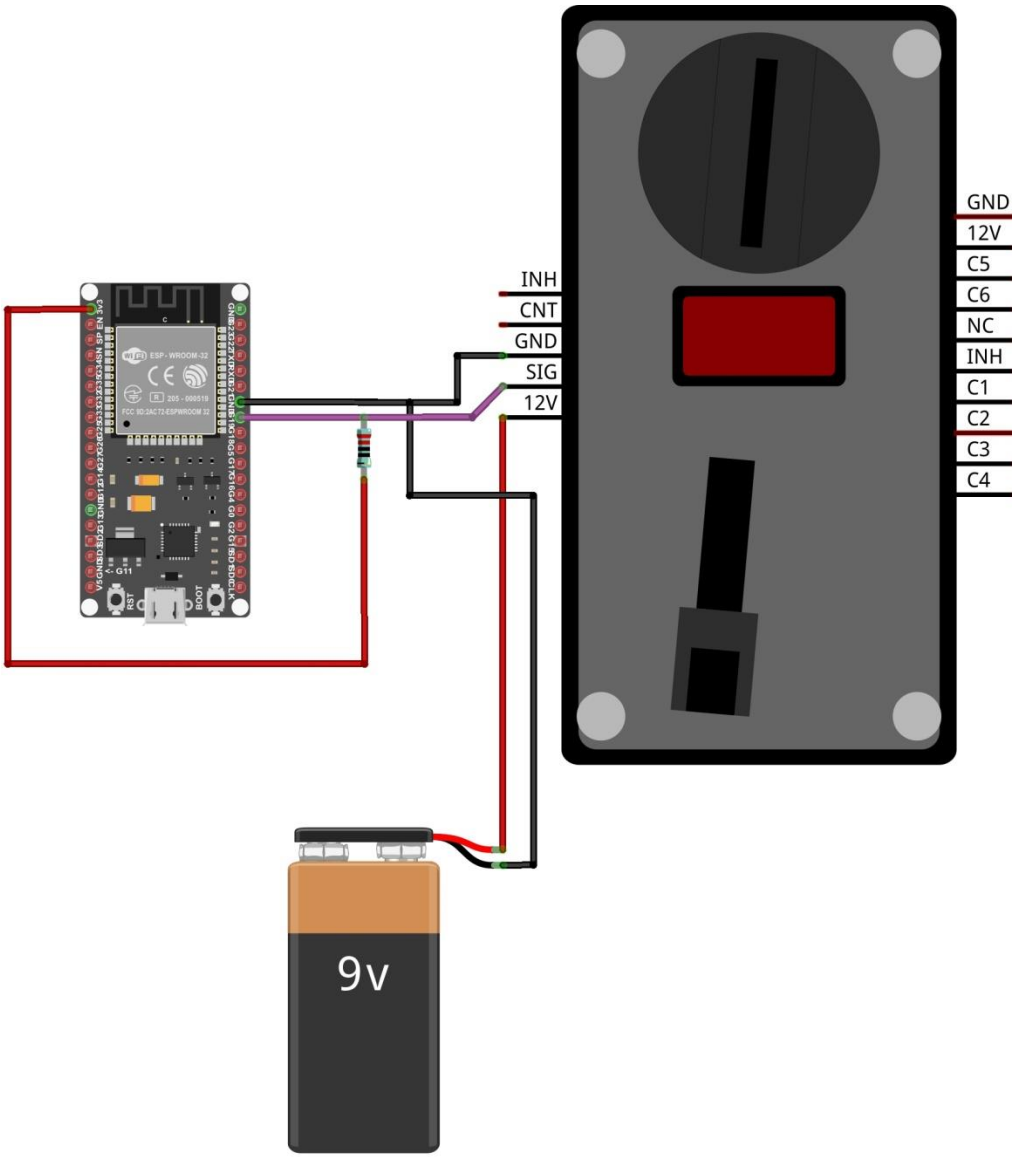


Figure3. 10Wiring Diagram between ESP32& Coin Acceptor

- ESP32 connect with LN298N driver ：

Table3. 3ESP32 Connect with LN298N

ESP32 microcontroller	LN298N
VCC	VCC
GND	GND
GPIO6	IN1
GPIO7	IN2

- Connection wiring diagram between the ESP32 controller and a LN298N:

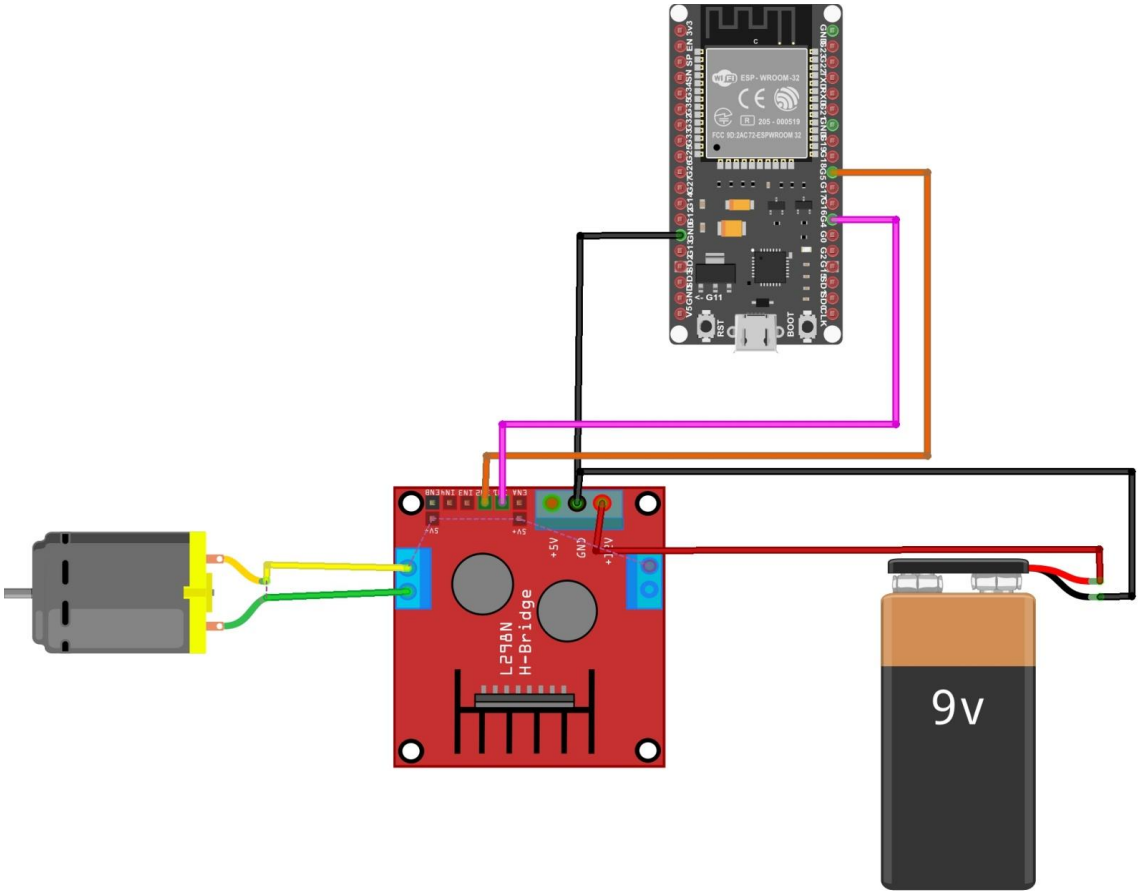


Figure3. 11Wiring Diagram between ESP32& LN298N

Wiring diagram for all systems :

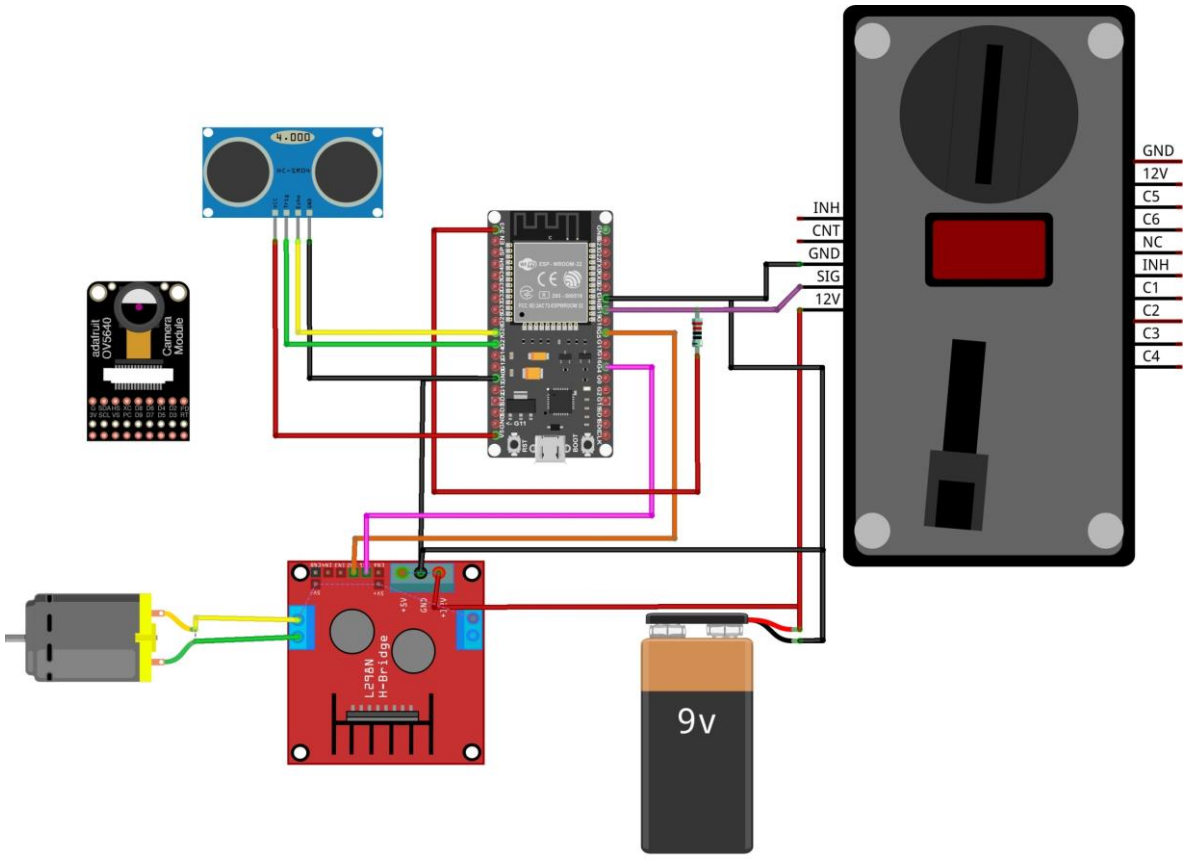


Figure3. 12Wiring Diagram

3.10 System Schematic diagram

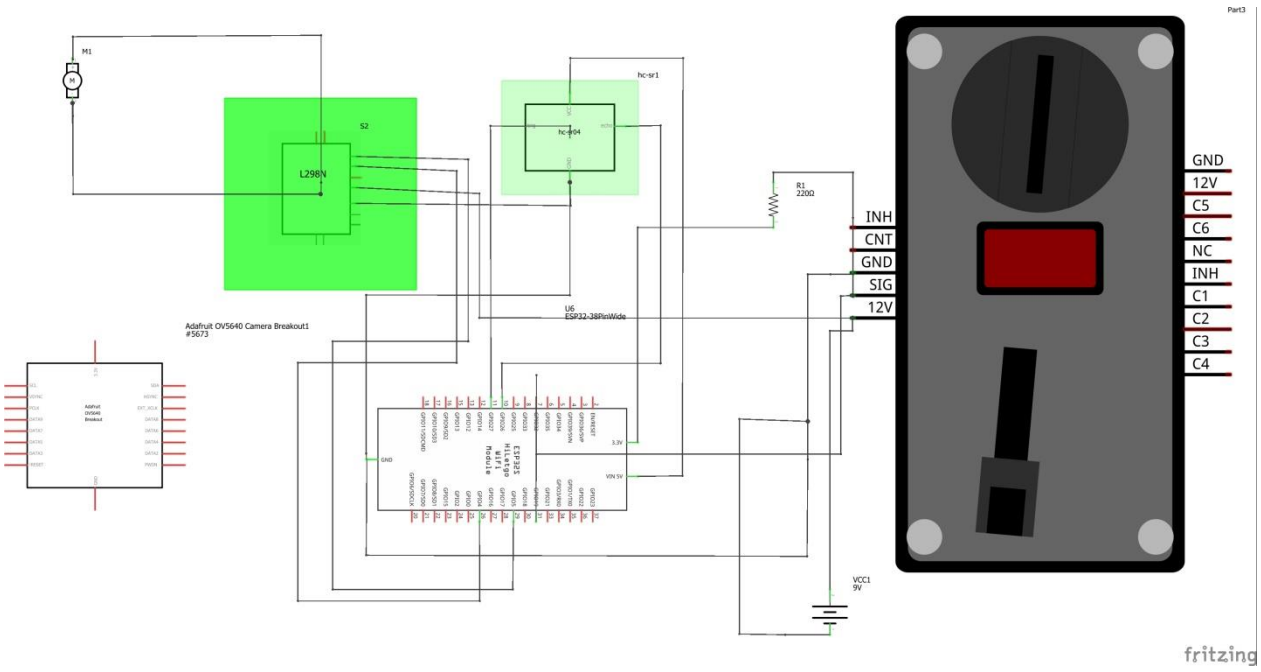


Figure3. 13 System Schematic diagram

CHAPTER FOUR : IMPLEMENTATION AND TESTING

4.1 Preface

In this chapter, we will discuss the implementation of the prepaid parking system components, both software and hardware. We will begin by discussing implementation challenges, explaining the challenges we encountered during the project, and then present the mobile application and its features. We will also review the project website, including the connections of the electronic components, and discuss implementation challenges, as well as project validation and testing.

4.2 Implementation issues

During the implementation of the prepaid parking system, we encountered several implementation issues that needed to be resolved to complete the system's operation. One of the most significant of these issues was the coin acceptor configuration mechanism. The device must be fully configured, its connection terminals properly identified, and its connection to the controller properly ensured accurate operation. The following figure 4.1 illustrates the device configuration and operation process.

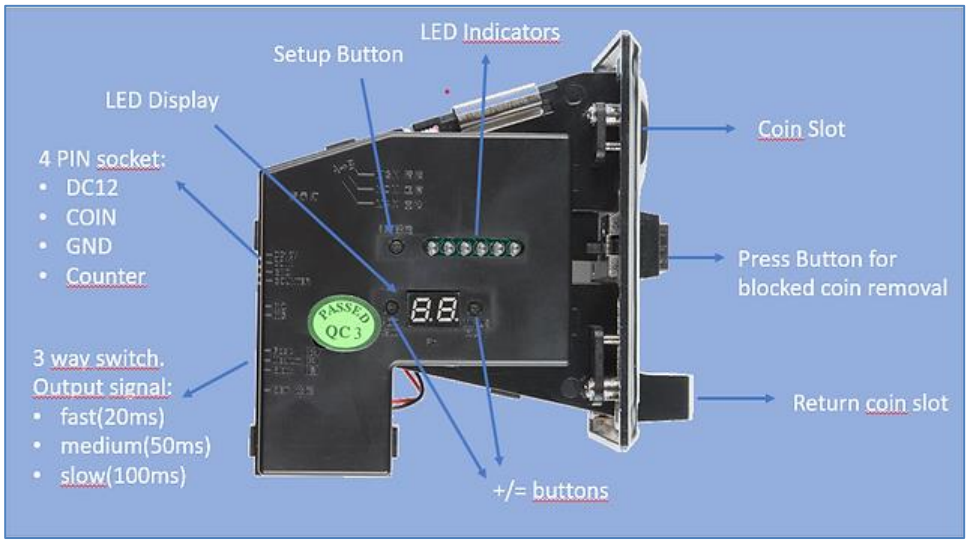


Figure4. 1Adjusting coin acceptor

Another implementation issue we encountered was counting the pulses for coins. The following figure 4.2 illustrates the pulse diagram for coins. In the Coin Acceptor, coins are identified based on physical characteristics such as size, weight, and material. When a coin is inserted into the device, it passes through a set of sensors that analyze the coin and ensure it matches specific values previously stored within the device. If the coin is successfully recognized, the device sends a specific number of electrical pulses via the signal output to the microcontroller. Each coin sends a different number of pulses. For example, one pulse might represent a 2 shekel coin, two pulses mean 5 shekel, and so on. The microcontroller counts these pulses over a short period of time and then determines the coin type based on the number of pulses received.

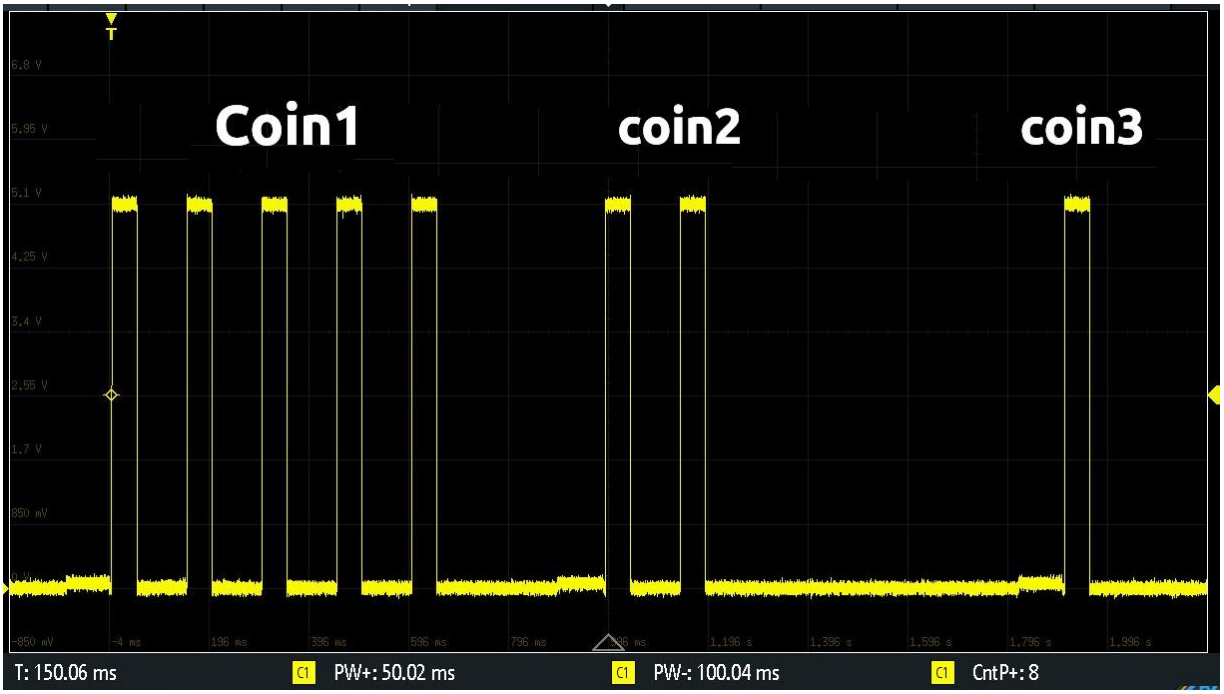


Figure4. 2 Pulses For Coins

In addition, we faced the problem of choosing the most appropriate way to display the QR code on the LCD , as the following figure 4.3 shows the QR code display code.

```
<div class="qr-code">
  <?php
    $data = urlencode($qr_random_num);
    $qr_base64 = base64_encode(file_get_contents("https://api.qrserver.com/v1/create-qr-code/?size=200x200&data=" .
$data));
    echo "<img src='{$qr_base64}' alt='QR Code' />";
  ?>
</div>
```

Figure4. 3 Code display QR LCD

Another issue we encountered was opening the map in the app. This was done using Open Street Map, an open-source mapping platform that provides geographic data that can be freely used in web and mobile applications. To display a map in your app using OSM, you typically use a middleware library or API such as Leaflet.js (for web applications) or the Leaflet or Open Street Map SDK (for mobile applications). The following figure 4.4 shows the code for the connection links to get the map.

```
bool useHttps = false; // Change this based on your requirement
final String config_URL = "192.168.1.105";//
final String path_URL="/project/api-flutter/";
final String config_unencodedPath = path_URL+"index.php";
final String config_unencodedPath_upload_img = path_URL+"upload_img.php";


final String show_img_on_server = path_URL+"show_img.php?img=";

final Map<String, String> config_post_headers = {'Content-Type': 'application/json; charset=UTF-8'};
```

Figure4. 4 Connection Code to Get Map

4.3 Implementation challenges

We encountered several challenges during the implementation of the prepaid parking system, which required significant effort. One of the main difficulties was establishing stable and synchronous communication between the hardware components (such as the coin receiver, ultrasonic sensor, servo motor, and tractor) and the software components running on the ESP32 microcontroller and communicating with the web API.

We also encountered synchronization issues between the ESP32 module and the remote server. This required continuous HTTP POST requests to send data such as the entered amount, vehicle presence, and door status updates. We implemented custom timers (using the elapsedMillis library) to schedule these interactions efficiently without disrupting the main program loop.

Another challenge we faced was the process of creating a QR code, which required us to understand and study how it works. The process of creating a QR code involves converting a string of data (such as a website link, a number, or text) into a grid of black and white squares. The following figure 4.5 illustrates the QR code:

```
class _NewQRScanState extends State<NewQRScan> {
  final globleFormKey = GlobalKey<FormState>();
  TextEditingController _c_reservation_item_list = TextEditingController();
  // Barcode scanner instances with prefixes
  final mobile_scanner.MobileScannerController mobileScannerController =
  mobile_scanner.MobileScannerController();
  bool validate = false;
  bool circular = false;
  String? errorText;
  bool isScanned = false;
  @override
  void initState() {
    super.initState();
    if (globals.qr_code != null) {
      _c_reservation_item_list.text = globals.qr_code!;
    }
  }
  @override
  void dispose() {
    _c_reservation_item_list.dispose();
    mobileScannerController.dispose();
    super.dispose();
  }
```

Figure4. 5 QR Generation

4.4 Software Implementation

This section will show case both the website and the application for the prepaid parking system, detailing the software for each, from the login interface to the rest of the details.

4.4.1 Mobile application Software Implementation

The following figure 4.6 illustrates the main interface of the application. Users access the system via a mobile app, which allows them to scan a QR code, open the camera, check the remaining time on the meter, and control the lock, in addition to electronic booking.

Figure4. 6Main Page

The following figure 4.7 shows how to scan the QR code through the application. During the payment process, the code appears on the screen and then opens in the application.

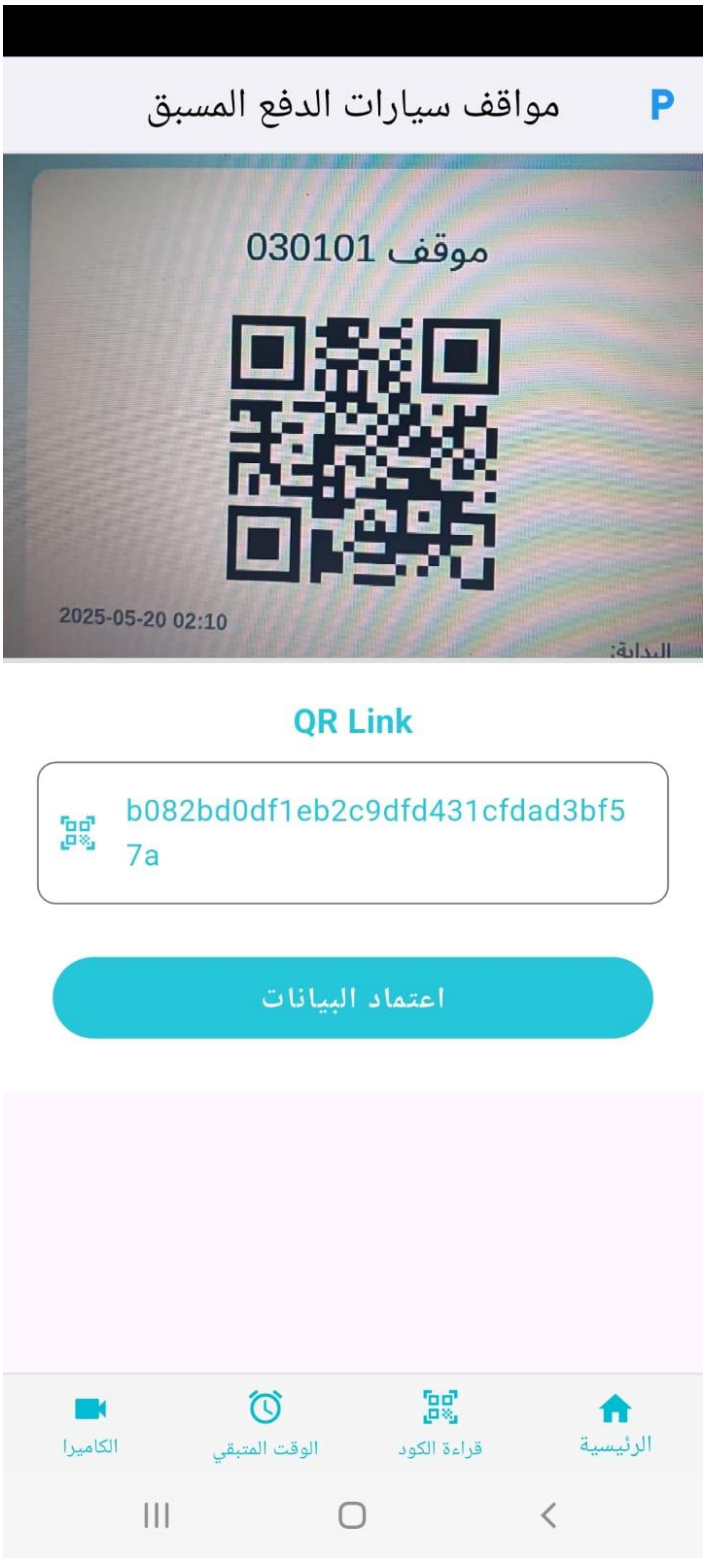


Figure4. 7 QR Link

The following figure 4.8 shows how to control the lock, as it can be opened and closed manually through the application.

Figure4. 8Control Lock

Figure 4.9 below shows the areas that contain parking spaces. Through this interface, you can select the appropriate area and then move to the map and plans interface and book now.

Figure4. 9 Select Appropriate Area

Figure 4.10 below shows the time remaining for the vehicle in the parking space, whether the space is closed or open, and whether there is a vehicle or not. The camera can also be opened to view the vehicle.

Figure4. 10 Time Remaining

Figure 4.11 below shows the online booking interface where card data is entered, the number of hours is specified, and the booking is completed.

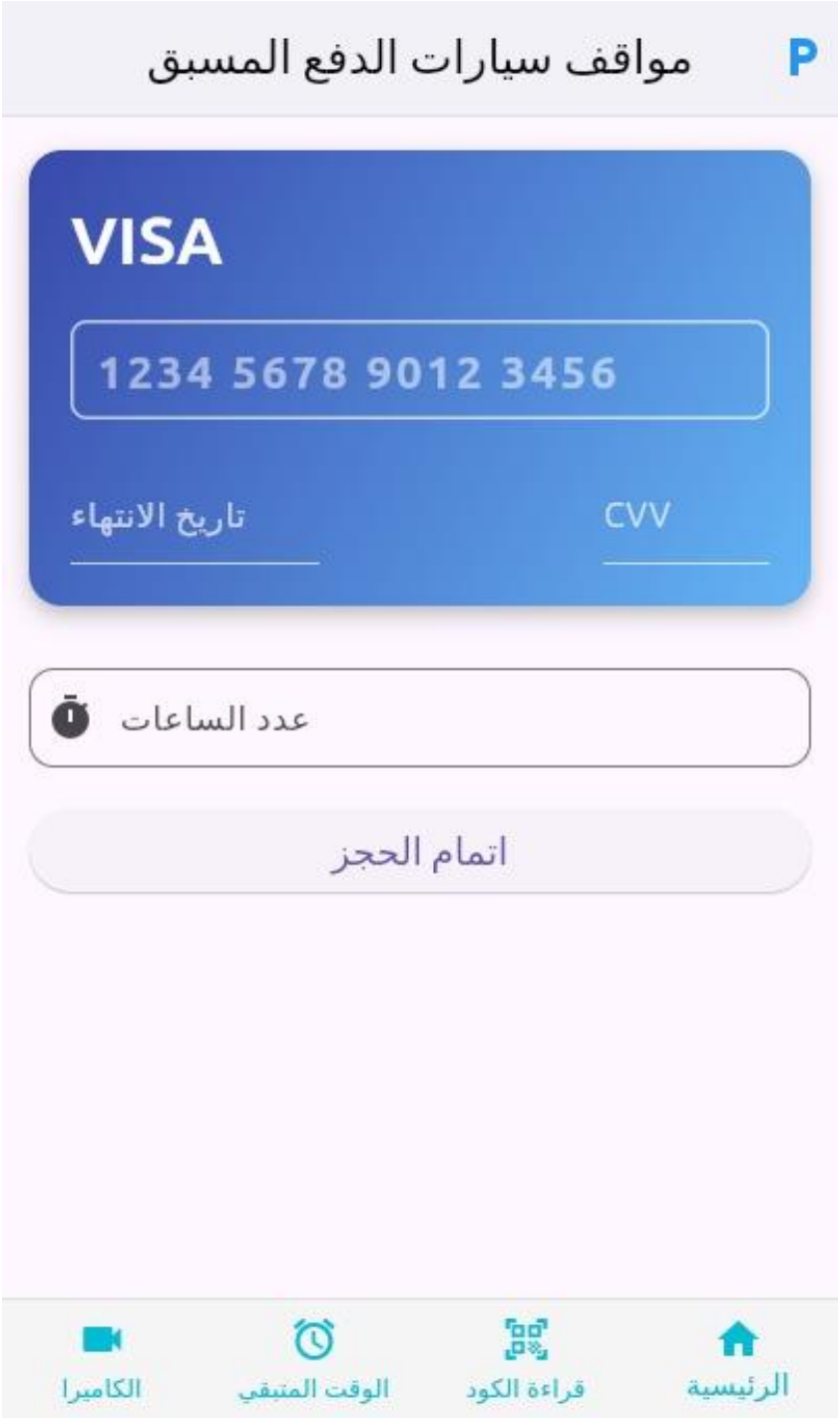


Figure4. 11 Entering Card Information

Figure 4.12 below shows the area, position, and node ID , as well as opening the map, viewing the plan, or working on the reservation now.

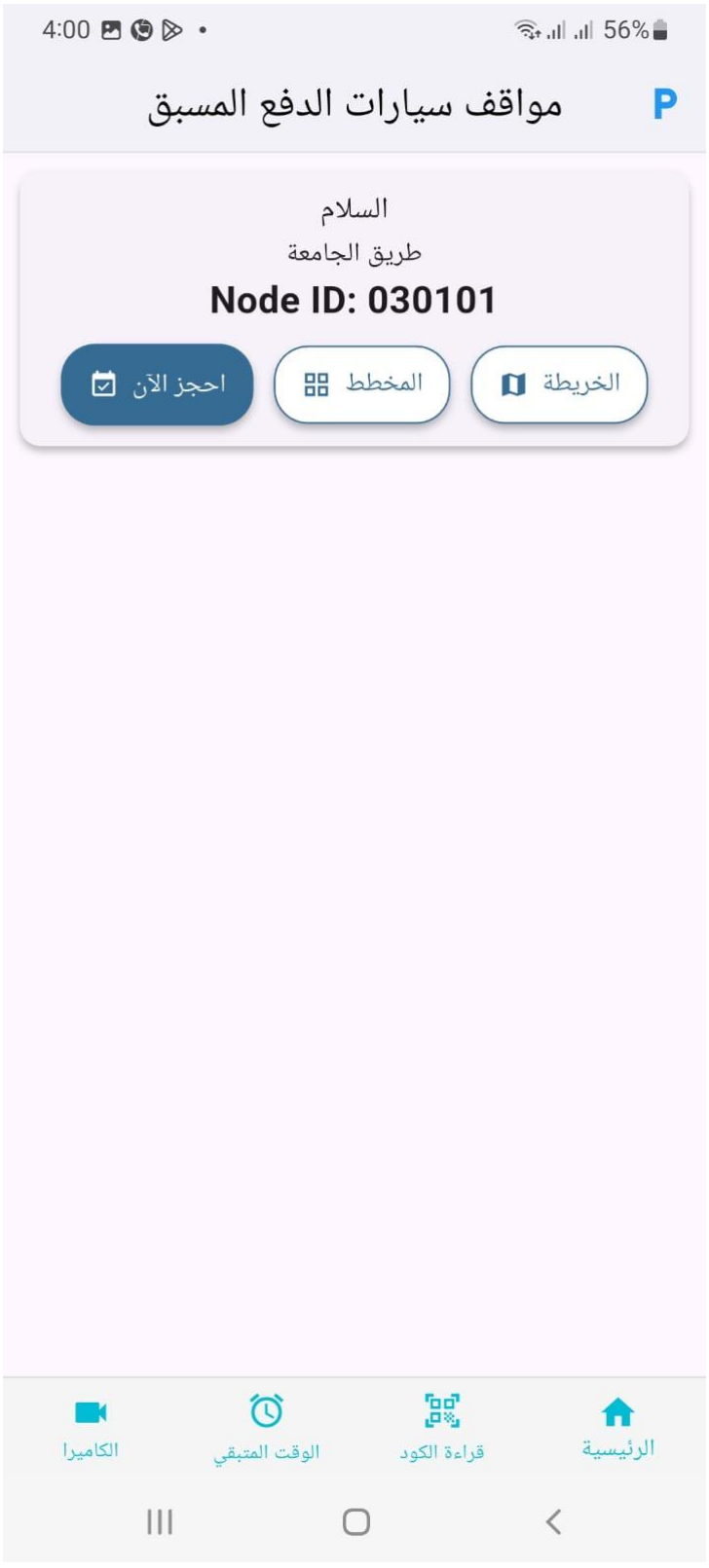


Figure4. 12 Shows The Area

Figure 4.13 shows a map showing areas with parking spaces.

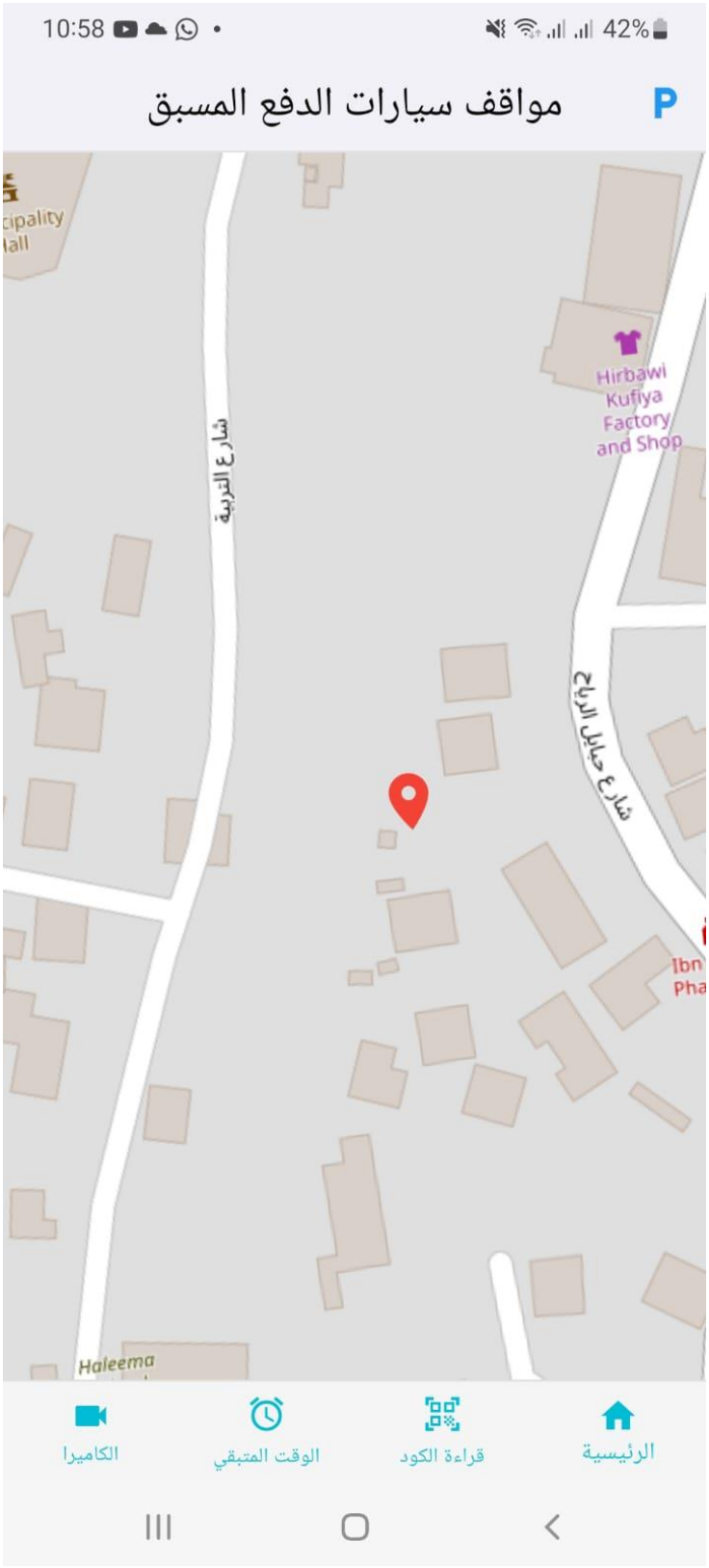


Figure4. 13 Shows The Map

Figure 4.14 shows a plan that helps you pinpoint the exact location of the parking space by placing a check mark in the specified location.

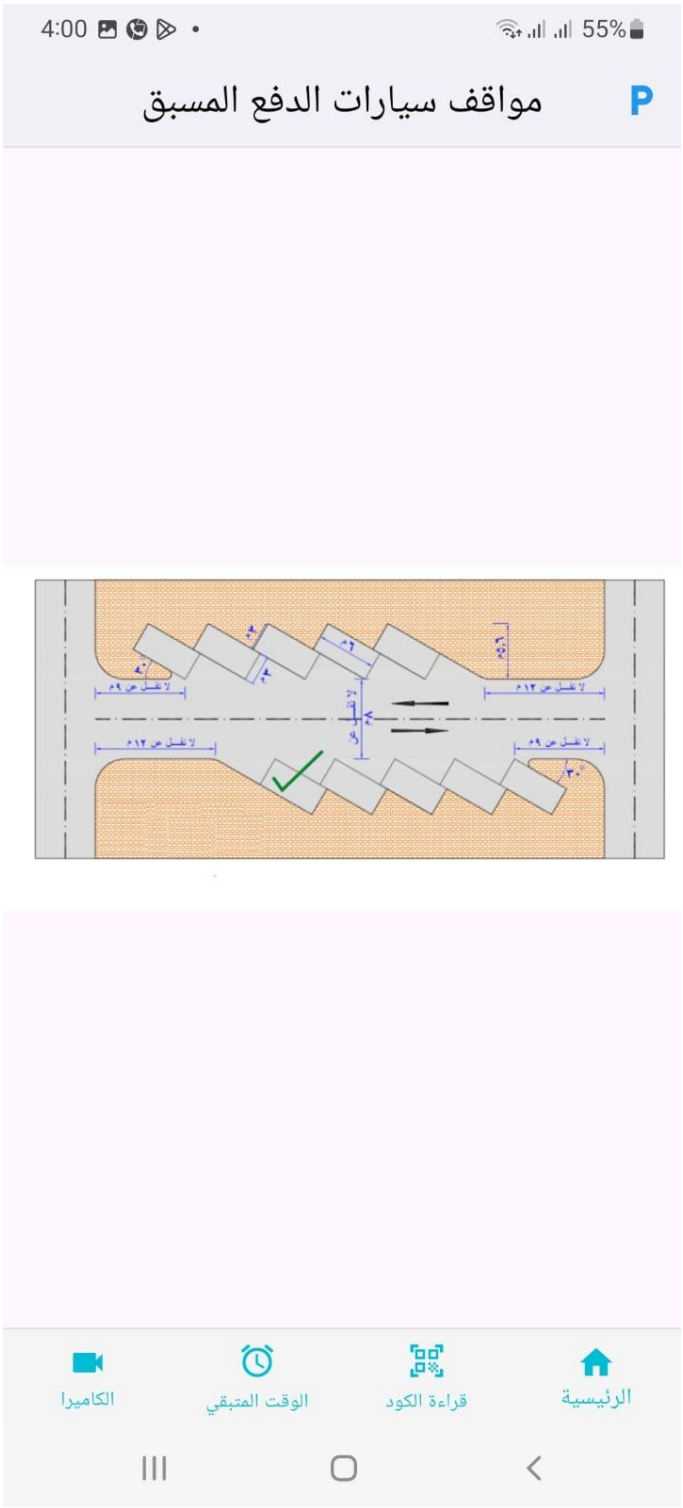


Figure4. 14 Shows The Plan

4.4.1 Website Software Implementation

The following figure 4.15 shows the login interface for the website, where restrictions have been placed, as only authorized persons who have a username and password are allowed to enter the website.

Figure4. 15Login Page

The following figure 4.16 shows the reservation record, where the time, date, and parking number are specified, in addition to the cost and QR code link, as well as the ability to modify or delete any record.

#	الموقف	التاريخ	الوقت	المدة(ساعة)	التكلفة	QR Random	تعديل	حذف
1	030101	2025-04-15	00:00	1	2₪	489557d98683f98dc1b5f98d81f572c0		
2	030101	2025-04-15	00:00	1	2₪	f494fa5ddcfb713c3f9f35566ad0f080		
3	030101	2025-04-15	00:00	1	2₪	b03e93521a37945417075442b767028b		
4	030101	2025-04-15	00:00	1	2₪	6ada12b5efa6603d77a3e39d61b8965c		
5	030101	2025-04-15	00:00	1	2₪	98806541f4de487ee3cfe0a2c889cb26		
6	030101	2025-04-15	00:00	1	2₪	e206fc8d4aa17b304be4e75e63a0bccd		
7	020201	2025-03-04	22:00	3	56₪	56dcf9e65ff78cf35f243c1279707c20		
8	020201	2025-01-30	02:05	3	3₪	e8545815e8239c363c32a0518ea42750		
9	020201	2025-05-08	15:07	2	4₪	d68386e2d4fa84b87922a039e5c2be74		

The following figure 4.17 shows the possibility of reservation, which is done by entering the following data: the parking number, time and date, in addition to specifying the cost and time period.

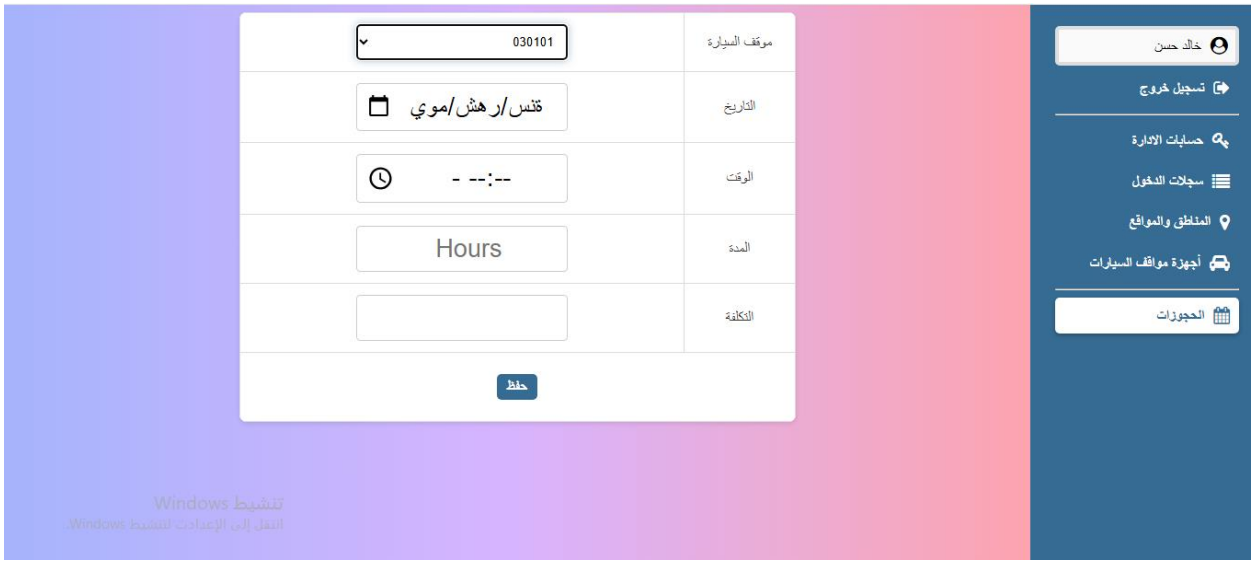


Figure4. 17 Reservation Page

The following figure 4.18 shows the areas that contain parking spaces, where the area and location can be specified, in addition to deleting and modifying areas.

Figure4. 18Location Parking

Figure 4.19 shows the parking device screen, where you can add a new device, modify or delete an existing device, view the serial number, map and diagram, and open the camera and view the location of each device.

Figure4. 19 Parking device

The following figure 4.20 shows the reports screen where a group of reports can be displayed, one for areas and another for positions, as well as parking reports, where each one displays a screen for the report's graph.

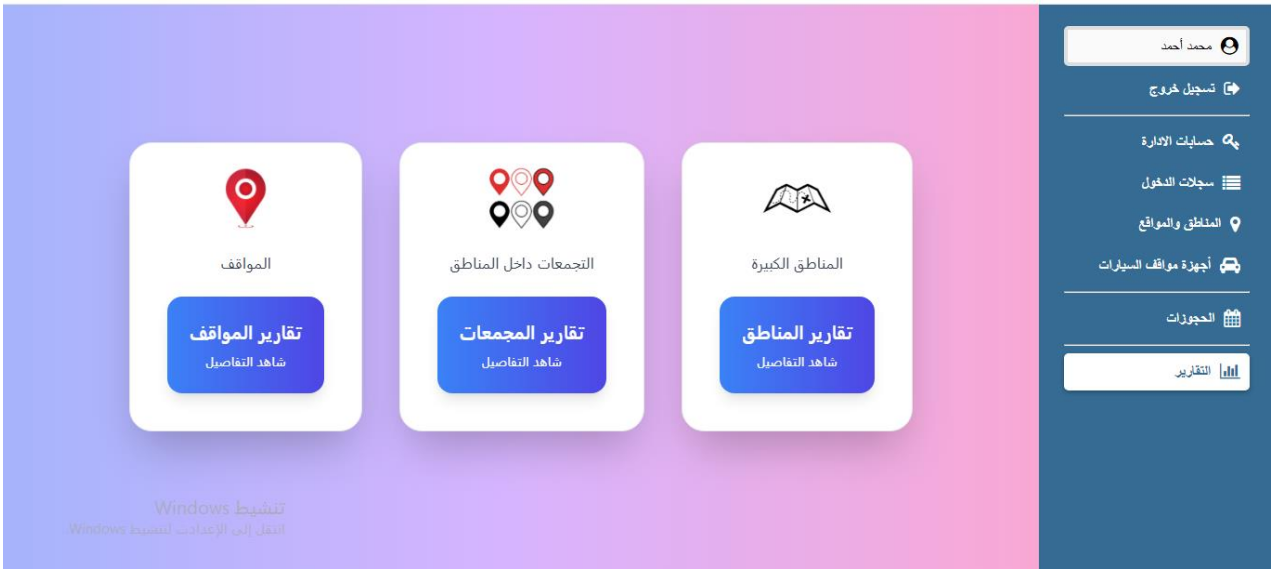


Figure4. 20 Reports Screen

The following figure 4.21 shows the chart of the areas report where the region and cost percentage are displayed.

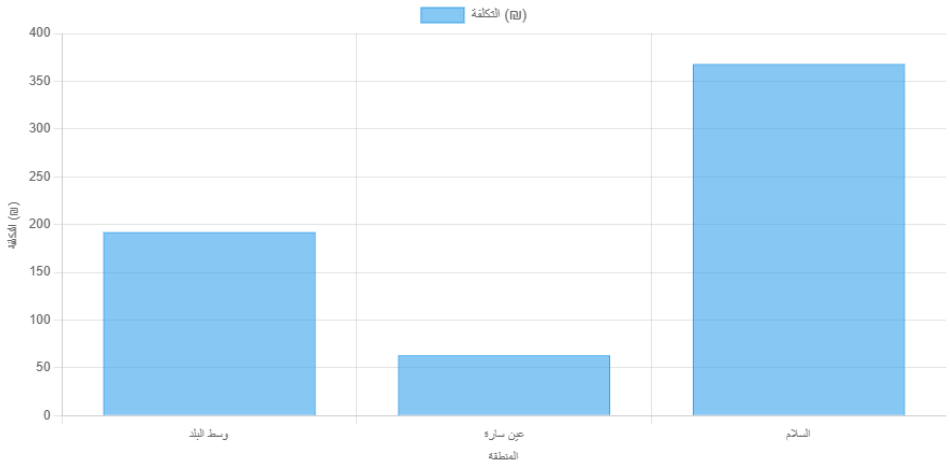


Figure4. 21 Chart Of Areas

The following figure 4.22 shows the chart of the position report within the regions where the cluster and cost are displayed.

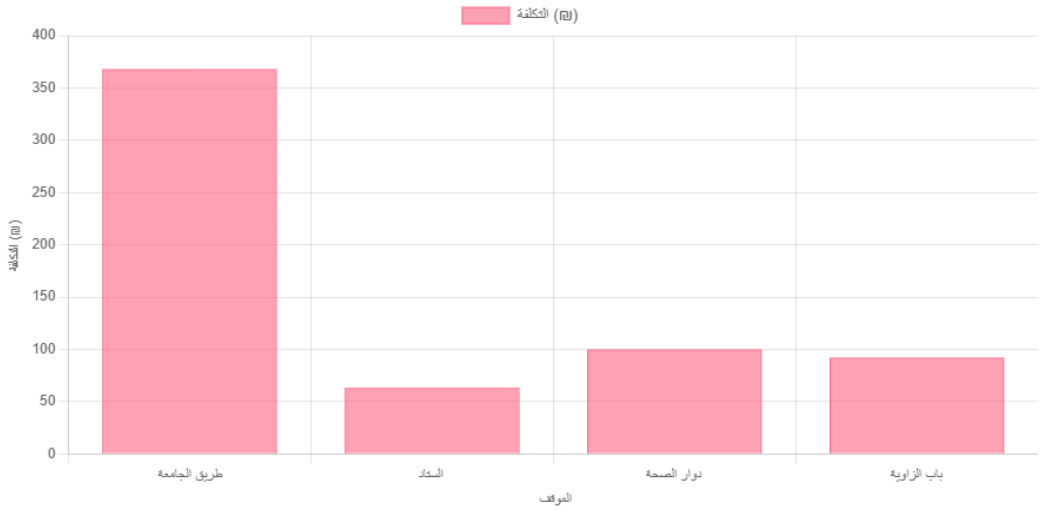


Figure4. 22 Chart Of Position

The following figure 4.23 shows the chart of the node report and the cost for each one.

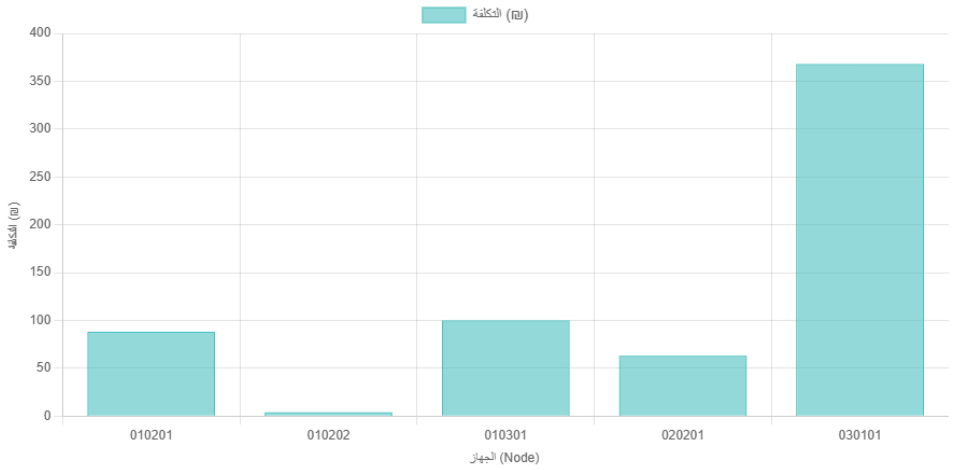


Figure4. 23 Chart Of Node

4.5 Hardware Implementation

The implementation of the proposed system includes the design of several main components of a prepaid parking system, where the coin receiver is connected to the controller, the ultrasonic sensor, and the driver. The following figure shows how the electronic components are connected to operate them and ensure accurate readings and communication between them.

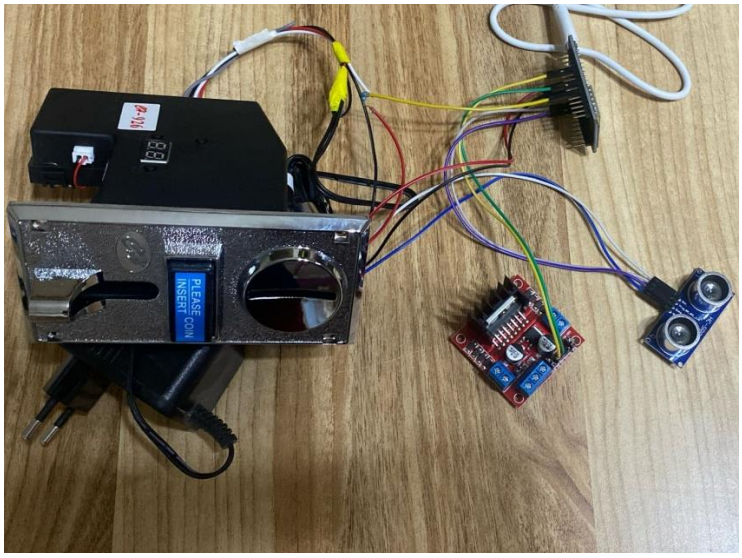


Figure4. 24Hardware Implementation

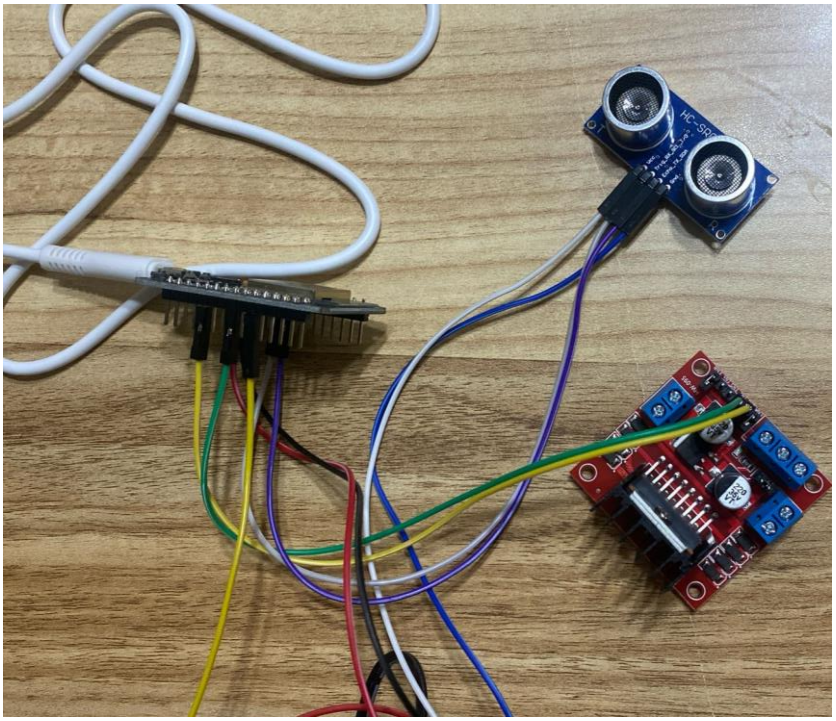


Figure4. 25 Hardware component

4.6 Validation and testing

Validation and testing were carried out to ensure the system performed efficiently under various conditions. Each hardware component was individually tested to ensure proper operation before full integration. The coin acceptor was tested using multiple currencies to verify recognition accuracy. Communication between the ESP32 and the web application was tested using simulated and real user inputs.

QR code generation and scanning were tested using mobile devices to verify the validity of the association with user accounts. We also performed system stress testing to evaluate response times and performance under continuous use. The system's effectiveness and correct performance were verified.

Additionally, the map was tested, as we worked with a variety of maps and linked them to the app and website. Open Street Map proved to be the best solution for displaying maps on the website and app, and for ease of connection and integration.

CHAPTER FIVE: RESULTS AND DISCUSSION

5.1 Detailed analysis of the results/experiments

Many tests and experiments were conducted on the prepaid parking system to determine the efficiency and capacity of the system. These experiments were represented in the following points:

1. Parking reservations are made in two ways: the first is through the mobile phone application, where reservations are made using a Visa card, and the second is through a coin acceptor, where payment is made directly.
2. Through the site map, the user can choose a location by first selecting a position, then an area, and finally one or more nodes. To make it easier for users to locate the nearest parking space for example (Downtown - Al Manara Roundabout - Hebron Center A)
3. Connecting all parts of the system with the controller and ensuring real-time communication, where the processes are sequenced from payment and obtaining the QR until the remaining parking time expires.
4. Vehicle monitoring is done using a camera that is opened through a QR code. This enables the user to track and monitor his vehicle at all times.
5. Integration between the system parts, where once payment is made, a QR code appears and the timer starts counting. If 2 shekels are paid, parking is allowed for an hour from the start of payment.
6. Flexibility in the prepaid parking system, where the user can extend the parking time by paying again, and the system then extends the time period for the vehicle.
7. Detailed reports identify the most booked parking meters over a single month. This analysis helps understand user behavior and areas with high demand.

5.2 Error / success rate calculations

To evaluate the performance of the prepaid parking system under various user conditions and behaviors, the error and success rates were calculated based on system responsiveness, payment accuracy, and real-time updates. Success and error rates were calculated by determining the total number of attempts, the number of successful transactions, and the number of failures, in addition to the success rate (%) and error rate (%) .

Table5. 1Success and error rates calculated

Test Scenario	**Total Tests**	**Successful Operations**	**Failed Operations**	**Success Rate (%)**	**Error Rate (%)**
Mobile App Reservation	50	48	2	96%	4%
Coin acceptor Payment	40	38	2	95%	5%
Location Selection via Site Map	30	30	0	100%	0%
QR Code Generation	45	44	1	97.7%	2.3%
Vehicle Monitoring via Camera	25	23	2	92%	8%
Parking Time Extension	35	34	1	97.1%	2.9%

Overall Average Success Rate:

$$\frac{96 + 95 + 100 + 97.7 + 92 + 97.1}{6} = 96.3\%$$

Overall Average Error Rate:

$$100\% - 96.3\% = 3.7$$

These results indicate high system reliability and user satisfaction potential, with occasional failures mainly due to unstable internet connection or user error during payment procedures.

5.3 Justifications of the obtained results

The results achieved during the implementation and testing of the prepaid parking system demonstrate high efficiency and a high success rate. These results can be explained and justified through the following points:

1. The system was designed with a flexible and integrated architecture, where each module works together, helping to reduce failures and ensure reliable and consistent system continuity.
2. Continuous research and study to choose the best ways to display the site map and ensure continuous and effective communication with the application and the site.
3. Choose two payment methods to make it easier for users, as payment can be made directly via the coin acceptor or via Visa card, which opens the way for reservations in all ways and possibilities.
4. Using advanced technologies in the system, a quick response code has been created for each person who pays and reserves a reservation within the parking lot. This is an effective and easy method for users.
5. To ensure the safety and security of the system, immediate monitoring of the vehicle inside the parking lot was carried out by opening the camera and viewing the vehicle in real time.

5.4 Summary

This chapter presents the project results in terms of efficiency, accuracy, and high effectiveness in the operation of the prepaid parking system, by integrating the best technologies in communication and linking between the physical components and devices to implement the system in the best possible way.

CHAPTER SIX: CONCLUSION AND FUTURE WORK

6.1 Conclusion

In conclusion, the prepaid parking system has proven its effectiveness and efficiency in light of technological advancements and the increasing population and vehicle density in urban areas. This has made it necessary to devise smart and practical solutions for parking management. Therefore, we designed and implemented the prepaid parking system as an effective response to community needs. It combines smart device technologies, mobile applications, and online platforms to provide a seamless and safe user experience for all users.

The system relies on an ESP controller that manages the basic operations of the system, connecting both the hardware and software components of the prepaid parking system. The mobile application is designed with an easy-to-use interface that allows users to reserve a parking space remotely using a Visa card and track the remaining parking time using a QR code generated after payment. The application also displays an interactive map based on Open Street Map technology to accurately locate available parking spaces. It also allows remote control of parking locks for increased security and flexibility.

On the administrative side, the system's website provides a control panel for administrators to view monthly reports for each parking space, analyze data to identify the most frequently used parking spaces through a graph, view reservations, and manage parking devices by adding, modifying, or deleting a device. Additionally, it manages user accounts.

Finally, this project represents a practical step towards creating a sustainable smart infrastructure that supports smart city concepts and contributes to the development of urban transportation systems by utilizing modern technologies.

6.2 Future work

In the future, the system can be supported with numerous technologies and systems that enhance operational efficiency, including:

1. A notification and alert system for users, which is sent in case of danger.
2. Adding new sensors to the system to increase efficiency and accuracy.
3. Using machine learning algorithms to analyze usage data, predict busiest times, and send alerts to users.
4. Adding artificial intelligence to the system through license plate recognition to enhance safety.
5. Deploying the system on the largest possible scale to include all parking spaces in Hebron and other cities.
6. Improving the application's user interface by adding features such as voice search.
7. Developing the system to automatically notify users before the reservation time expires or when a nearby parking space becomes available.

Reference

[1] S. P. S. Aithal, G. S. Kulkarni, and M. K. Mhatre, "RFID and HDL Based Pre-Paid Car Parking System," in 2018 International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, Oct. 2018, pp. 712-716. "[Online]. Available https://ieeexplore.ieee.org/document/8474606

[2] P. R. Singh and N. D. Gupta, "Support for Self-service Automated Parking Systems," in 2020 International Conference on Emerging Trends in Engineering and Technology (ICETET), Nagpur, India, Oct. 2020, pp. 345-350. [Online]. Available: https://ieeexplore.ieee.org/document/9066319

[3] J. K. Verma, A. K. Patel, and R. S. Yadav, "Streamline Your Drive: Pre-Booking & Pre-Paid Parking Slots, Locating CNG Stations & EV Charging Stations," in 2024 International Conference on Intelligent Transportation Systems (ITS), Bangalore, India, Nov. 2024, pp. 456- [Online]. Available: https://ieeexplore.ieee.org/document/10731066

[4] Espressif , "ESP32 NODMCU microcontroller," Make-It, Oct. 25, 2024. [Online]. Available: https://www.make-it.ca/nodemcu-details-specifications/

[5] "Coin acceptor," DEC 10, 2024. [Online]. Available: https://www.amazon.com/coin-acceptor/s?k=coin+acceptor

[6] "Ultrasonic sensor," MaxBotix, DEC 20, 2024. [Online]. Available: https://maxbotix.com/blogs/blog/how-ultrasonic-sensors

[7] "ESP-CAM" OCT 15, 2024. [Online]. Available:

https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/

[8] "DC Motor "Datasheet, TowerPro.pdf (accessed on 6 January 2025). [Online]. Available:

https://byjus.com/physics/dc-motor/

[9] " L298N Motor Driver, TowerPro.pdf ,accessed on 10OCT 2020. [Online]. Available:

https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/

[10]"Arduino IDE," Arduino, Nov. 10, 2024. [Online]. Available:

https://www.arduino.cc/en/software

[11] "FLUTTER language to build android APK" Nov. 12, 2024. [Online]. Available: https://flutter.dev/

[12] "Fritizing," Fritizing. [Online]. Available:

https://fritzing.org

[13] "XAMPP SERVER " NOV 5, 2024. [Online]. Available:

https://www.apachefriends.org/

[14] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," IEEE Pervasive Computing, vol. 7, no. 4, pp. 12–18, Oct.–Dec. 2008, doi: 10.1109/MPRV.2008.80. [Online]. Available: https://www.openstreetmap.org/#map=4/36.81/35.68

Appendix A

ESP32 CODE

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <EEPROM.h>
#include <elapsedMillis.h>

#define TRIG_PIN 27
#define ECHO_PIN 26
const int coinSensorPin = 19;
#define BUZZER_PIN 2  // تحديد الطرف المتصل بالصفارة

//=====l298n
#define IN1_PIN 4
#define IN2_PIN 5
const char* ssid = "ps";
const char* password = "00000000";

const char* url_reservation =
"http://172.20.10.2/project/api/api_reservation.php";

String deviceID = "030101";

WiFiClient client;
HTTPClient http;

// note: المتغيرات الخاصة بالعملات
int totalAmount = 0;
int old_totalAmount = 0;
volatile bool updateDisplay = false;
volatile bool coinInserted = false;
elapsedMillis timer;
elapsedMillis countDownTimer;
const long intervalPerCoin = 10000; // الزمن المخصص لكل وحدة نقدية
bool countDownStarted = false;

// note: المتغيرات الخاصة بالمسافة
bool carPresentNotified = false;
float distanceThreshold = 10.0; // سنتيمتر

// note: دالة الاتصال بالشبكة
void connectWiFi() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
  }
  Serial.println("\nWiFi Connected: " + WiFi.localIP().toString());
}

// note: دالة حساب المسافة باستخدام الألتراسونك
float getDistance() {
  digitalWrite(TRIG_PIN, LOW); delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  return pulseIn(ECHO_PIN, HIGH) * 0.034 / 2.0;
}

// note: دالة إرسال البيانات إلى السيرفر
void postReservation(String payload) {
  http.begin(client, url_reservation);
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
  int code = http.POST(payload);
  if (code > 0) {
    Serial.println("Server response: " + http.getString());
  } else {
    Serial.println("POST failed");
  }
  http.end();
}

// note: دالة جلب الوقت المتبقي من الـ API
int getRemainingTimeFromAPI() {
  String payload = "action=get_remaining_time&deviceNode_serial_ID=" +
deviceID;
  http.begin(client, url_reservation);
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");

  int code = http.POST(payload);
  int remainingTime = -1;

  if (code > 0) {
    String response = http.getString();
    Serial.println("API Response: " + response);

    DynamicJsonDocument doc(256);
    DeserializationError error = deserializeJson(doc, response);
    if (!error && doc["status"] == "success") {
      remainingTime = doc["remaining_time"];
    }
  } else {
    Serial.println("Failed to get remaining time");
  }

  http.end();
```

```
  return remainingTime;
}
String get_lock_control_FromAPI() {
  String payload = "action=get_lock_control&deviceNode_serial_ID=" +
deviceID;
  http.begin(client, url_reservation);
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");

  int code = http.POST(payload);
  String lock_control = "";

  if (code > 0) {
    String response = http.getString();
    Serial.println("API Response: " + response);

    DynamicJsonDocument doc(256);
    DeserializationError error = deserializeJson(doc, response);
    if (!error && doc["status"] == "success") {
      lock_control = String(doc["lock_control"]);
    }
  } else {
    Serial.println("Failed to get remaining time");
  }

  http.end();
  return lock_control;
}

// مقاطعة استشعار العملات
void IRAM_ATTR coinISR() {
  coinInserted = true;
  updateDisplay = true;
}

elapsedMillis customTimer; // مؤقت خاص جديد للتحقق كل 2 ثانية
const unsigned long customInterval = 5000; // التحقق كل 2 ثانية


void setup() {
  Serial.begin(115200);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(coinSensorPin, INPUT_PULLUP);
  pinMode(BUZZER_PIN, OUTPUT);  // تحديد الطرف الخاص بالصفارة
  pinMode(IN1_PIN, OUTPUT);
  pinMode(IN2_PIN, OUTPUT);

  attachInterrupt(digitalPinToInterrupt(coinSensorPin), coinISR,
FALLING);
```

```
  connectWiFi();
  Serial.println("System Ready...");
}

int last_update = 0;

String lock_control="";
String old_lock_control="";
int car_present=0;

float period_time=0;

int reading_coin_now=0;
void loop() {

  if (coinInserted)
  {
    coinInserted = false;

    // هنا يمكن تعديل قيمة العملة بناءً على مدخلات أو زمن مرور العملة عبر الحساس
    int coinValue = 1; // يمكن تغييرها لـ 5 أو 10 بناءً على منطق معين
    totalAmount += coinValue;
    reading_coin_now += coinValue;

    Serial.print("Coin inserted: ");
    Serial.print(coinValue);
    Serial.print(" Baht, Total: ");
    Serial.println(totalAmount);

    if (totalAmount > 0 && !countDownStarted)
    {
      countDownStarted = true;
      timer = 0;
      countDownTimer = timer;
    }
  }
  if(totalAmount!=old_totalAmount && millis()-last_update> 5000 )
  {
      last_update=millis();
      int sub_amount=totalAmount-old_totalAmount;
      if(sub_amount>=4 && sub_amount<=6 )
      {
          Serial.println("-- coin is 1$ --");
          period_time=0.5;
          String payload = "action=reservation&deviceNode_serial_ID=" +
deviceID + "&period_time=" + String(period_time);
          postReservation(payload);
          reading_coin_now=0;
```

```
      }
      else if(sub_amount>=8 && sub_amount<=12 )
      {
          Serial.println("-- coin is 2$ --");
          period_time=1;
          String payload = "action=reservation&deviceNode_serial_ID=" +
deviceID + "&period_time=" + String(period_time);
          postReservation(payload);
          reading_coin_now=0;
      }
      else if(sub_amount>=13 && sub_amount<=17 )
      {
          Serial.println("-- coin is 5$ --");
          period_time=2.5;
          String payload = "action=reservation&deviceNode_serial_ID=" +
deviceID + "&period_time=" + String(period_time);
          postReservation(payload);
          reading_coin_now=0;
      }
      else if(sub_amount>=18 && sub_amount<=22 )
      {
          Serial.println("-- coin is 10$ --");
          period_time=5;
          String payload = "action=reservation&deviceNode_serial_ID=" +
deviceID + "&period_time=" + String(period_time);
          postReservation(payload);
          reading_coin_now=0;
      }

      old_totalAmount=totalAmount;

  }

  //each 2 second  ==========start
    if (customTimer >= customInterval && !reading_coin_now ) {
        customTimer = 0; // إعادة التايمر الخاص

        int serverTimeLeft = getRemainingTimeFromAPI();
        Serial.print("================================================
=============");
        Serial.println(serverTimeLeft);
        if (serverTimeLeft <= 0)
        {
          Serial.println("Time is up! Closing the door...");
          //close it if time out
          if(old_lock_control!="close")
          {
            digitalWrite(IN1_PIN, LOW);
            digitalWrite(IN2_PIN, HIGH);
```

```
            delay(4000);
            digitalWrite(IN2_PIN, LOW);

            old_lock_control="close";
          }
        }
        else if (serverTimeLeft > 0)
        {
            Serial.print("Time left from API: ");
            Serial.println(serverTimeLeft);
            lock_control = get_lock_control_FromAPI();
            if(lock_control=="open" && old_lock_control!="open")
            {
              digitalWrite(IN1_PIN, HIGH);
              digitalWrite(IN2_PIN, LOW);
              delay(4000);
              digitalWrite(IN1_PIN, LOW);
              old_lock_control="open";
            }
            else if(lock_control=="close" && old_lock_control!="close")
            {
              digitalWrite(IN1_PIN, LOW);
              digitalWrite(IN2_PIN, HIGH);
              delay(4000);
              digitalWrite(IN2_PIN, LOW);
              old_lock_control="close";
            }
        }

        float distance = getDistance();
        Serial.println("Distance: " + String(distance) + " cm");
        car_present=0;
        if (distance < distanceThreshold) {
            if (serverTimeLeft <= 0) {
                digitalWrite(BUZZER_PIN, HIGH);
                delay(500);
                digitalWrite(BUZZER_PIN, LOW);
            }
            car_present=1;
        }
        String payload =
"action=set_car_present&car_present="+String(car_present)+"&deviceNode_
serial_ID=" + deviceID;
        postReservation(payload);

   }


}
```

Appendix B

CODE ESP32- CAM

```
#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG
quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is
selected
//            Partial images will be transmitted if image exceeds
buffer size
//
//            You must select partition scheme from the board menu that
has at least 3MB APP space.
//            Face Recognition is DISABLED for ESP32 and ESP32-S2,
because it takes up from 15
//            seconds to process single frame. Face Detection is
ENABLED if PSRAM is enabled as well

// ===================
// Select camera model
// ===================
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE  // Has PSRAM
//#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
//#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
//#define CAMERA_MODEL_M5STACK_CAMS3_UNIT  // Has PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
//#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
// ** Espressif Internal Boards **
//#define CAMERA_MODEL_ESP32_CAM_BOARD
//#define CAMERA_MODEL_ESP32S2_CAM_BOARD
//#define CAMERA_MODEL_ESP32S3_CAM_LCD
//#define CAMERA_MODEL_DFRobot_FireBeetle2_ESP32S3 // Has PSRAM
//#define CAMERA_MODEL_DFRobot_Romeo_ESP32S3 // Has PSRAM
#include "camera_pins.h"

// ===========================
// Enter your WiFi credentials
// ===========================
const char *ssid = "ps";
const char *password = "00000000";
```

```
void startCameraServer();
void setupLedFlash(int pin);

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG;  // for streaming
  //config.pixel_format = PIXFORMAT_RGB565; // for face
detection/recognition
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;

  // if PSRAM IC present, init with UXGA resolution and higher JPEG
quality
  //                     for larger pre-allocated frame buffer.
  if (config.pixel_format == PIXFORMAT_JPEG) {
    if (psramFound()) {
      config.jpeg_quality = 10;
      config.fb_count = 2;
      config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
      // Limit the frame size when PSRAM is not available
      config.frame_size = FRAMESIZE_SVGA;
```

```
      config.fb_location = CAMERA_FB_IN_DRAM;
    }
  } else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t *s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit
saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);        // flip it back
    s->set_brightness(s, 1);   // up the brightness just a bit
    s->set_saturation(s, -2);  // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  if (config.pixel_format == PIXFORMAT_JPEG) {
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

#if defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

#if defined(CAMERA_MODEL_ESP32S3_EYE)
  s->set_vflip(s, 1);
#endif

// Setup LED FLash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
  setupLedFlash(LED_GPIO_NUM);
#endif
```

```
  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // Do nothing. Everything is done in another task by the web server
  delay(10000);
}
```

Appendix C

SET APPLICATION PERMISSIONS CODE

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.INTERNET" />

    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" /> <!-- Add this -
->
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" /> <!-- Add this
-->

    <application
        android:label="prepaid_parking"
        android:name="${applicationName}"
```

```
        android:icon="@mipmap/ic_launcher">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:launchMode="singleTop"
            android:taskAffinity=""
            android:theme="@style/LaunchTheme"
            android:configChanges="orientation|keyboardHidden|keyboard|sc
reenSize|smallestScreenSize|locale|layoutDirection|fontScale|screenLayout
|density|uiMode"
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize">
            <!-- Specifies an Android theme to apply to this Activity as
soon as
                 the Android process has started. This theme is visible
to the user
                 while the Flutter UI initializes. After that, this theme
continues
                 to determine the Window background behind the Flutter
UI. -->
            <meta-data
              android:name="io.flutter.embedding.android.NormalTheme"
              android:resource="@style/NormalTheme"
              />
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category
android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <!-- Don't delete the meta-data below.
             This is used by the Flutter tool to generate
GeneratedPluginRegistrant.java -->
        <meta-data
            android:name="flutterEmbedding"
            android:value="2" />
    </application>
    <!-- Required to query activities that can process text, see:
         https://developer.android.com/training/package-visibility and
         https://developer.android.com/reference/android/content/Intent#A
CTION_PROCESS_TEXT.

         In particular, this is used by the Flutter engine in
io.flutter.plugin.text.ProcessTextPlugin. -->
    <queries>
        <intent>
            <action android:name="android.intent.action.PROCESS_TEXT"/>
            <data android:mimeType="text/plain"/>
        </intent>
```

```
        <intent>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.BROWSABLE" />
            <data android:scheme="http" />
        </intent>
        <intent>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.BROWSABLE" />
            <data android:scheme="https" />
        </intent>

    </queries>

</manifest>
```

Appendix D

LIBRARIES SETTINGS FOR THE APPLICATION CODE

```
name: prepaid_parking
description: A new Flutter project.
publish_to: 'none' # Remove this line if you wish to publish to pub.dev
version: 1.0.0+1

environment:
  sdk: ">=2.12.0 <3.0.0"


# Dependencies specify other packages that your package needs in order
to work.
# To automatically upgrade your package dependencies to the latest
versions
# consider running `flutter pub upgrade --major-versions`.
Alternatively,
# dependencies can be manually updated by changing the version numbers
below to
# the latest version available on pub.dev. To see which dependencies
have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter

  mobile_scanner: ^3.2.0
  webview_flutter: ^4.0.7
```

```
  #sqflite: ^2.3.0
  path: ^1.8.0
  flutter_map: ^6.1.0
  latlong2: ^0.9.0

  http: ^1.1.0

  geolocator: any
  url_launcher: ^6.2.5


  flutter_lints: ^4.0.0

  custom_info_window: ^1.0.1
  api_cache_manager: ^1.0.2

  hexcolor: ^3.0.1
  snippet_coder_utils: ^1.0.12
  logger: ^2.0.2+1
  dio:
  fluttertoast:
  intl: ^0.19.0

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  image_picker: ^1.0.5
  dob_input_field: ^2.0.0
  dropdown_button2: ^2.3.9
  flutter_holo_date_picker: ^2.0.0
  flutter_secure_storage: ^9.0.0

  #flutter pub remove fl_chart
  #flutter pub add fl_chart
  fl_chart: ^0.70.0
  #url_launcher: ^6.1.7

dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below contains a set of recommended
lints to
  # encourage good coding practices. The lint set provided by the
package is
  # activated in the `analysis_options.yaml` file located at the root
of your
```

```
  # package. See that file for information about deactivating specific
lint
  # rules and activating additional ones.

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like
this:
  assets:
    - assets/images/


  # An image asset can refer to one or more resolution-specific
"variants", see
  # https://flutter.dev/assets-and-images/#resolution-aware.

  # For details regarding adding assets from package dependencies, see
  # https://flutter.dev/assets-and-images/#from-packages

  # To add custom fonts to your application, add a fonts section here,
  # in this "flutter" section. Each entry in this list should have a
  # "family" key with the font family name, and a "fonts" key with a
  # list giving the asset and other descriptors for the font. For
  # example:
  # fonts:
  #   - family: Schyler
  #     fonts:
  #       - asset: fonts/Schyler-Regular.ttf
  #       - asset: fonts/Schyler-Italic.ttf
  #         style: italic
  #   - family: Trajan Pro
  #     fonts:
  #       - asset: fonts/TrajanPro.ttf
  #       - asset: fonts/TrajanPro_Bold.ttf
  #         weight: 700
  #
  # For details regarding fonts from package dependencies,
  # see https://flutter.dev/custom-fonts/#from-packages
```

Appendix E

THE QR GENARATION CODE

```
import 'package:flutter/material.dart';
import 'package:hexcolor/hexcolor.dart';
import 'package:snippet_coder_utils/FormHelper.dart';

import 'package:prepaid_parking/api/api_backend_method.dart';

import 'package:prepaid_parking/reservation/show_timer.dart';
import 'package:prepaid_parking/public/alert_dialog.dart';
import 'package:prepaid_parking/public/globals.dart' as globals;
import 'package:prepaid_parking/top_bottom/bottomBar_static.dart';
import 'package:prepaid_parking/top_bottom/topBar_static.dart';

import 'package:mobile_scanner/mobile_scanner.dart' as mobile_scanner;
import 'package:image_picker/image_picker.dart';

class NewQRScan extends StatefulWidget {
  const NewQRScan({Key? key}) : super(key: key);

  @override
  State<NewQRScan> createState() => _NewQRScanState();
}

class _NewQRScanState extends State<NewQRScan> {
  final globleFormKey = GlobalKey<FormState>();
  TextEditingController _c_reservation_item_list =
TextEditingController();

  // Barcode scanner instances with prefixes
  final mobile_scanner.MobileScannerController mobileScannerController
=
      mobile_scanner.MobileScannerController();

  bool validate = false;
  bool circular = false;
  String? errorText;
  bool isScanned = false;
  @override
  void initState() {
    super.initState();
    if (globals.qr_code != null) {
      _c_reservation_item_list.text = globals.qr_code!;
    }
  }

  @override
```

```
  void dispose() {
    _c_reservation_item_list.dispose();

    mobileScannerController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        //backgroundColor: Colors.transparent,
        appBar: topBar(context),
        bottomNavigationBar: footer(context),
        body: Container(
          color: HexColor("#ffffff"),
          child: Form(
            key: globleFormKey,
            child: _registerUI(context),
          ),
        ),
      ),
    );
  }

  Widget _registerUI(BuildContext context) {
    return SingleChildScrollView(
      child: Column(
        children: [
          Container(
            width: MediaQuery.of(context).size.width,
            height: 300,
            decoration: BoxDecoration(
              border: Border(
                bottom: BorderSide(
                  color: Color(0xffdfdfdf),
                  width: 3.0,
                ),
              ),
            ),
            child: mobile_scanner.MobileScanner(
                    controller: mobileScannerController,
                    onDetect: (capture) {
                      final List<mobile_scanner.Barcode> barcodes =
capture.barcodes;
                      if (!isScanned && barcodes.isNotEmpty &&
barcodes.first.rawValue != null) {
                        setState(() {
                          isScanned = true;
```

```
                        _c_reservation_item_list.text =
barcodes.first.rawValue!;
                      });
                    }
                  },
                ),
          ),
          SizedBox(height: 20),
          Text(
            "QR Link",
            style: TextStyle(
              fontWeight: FontWeight.bold,
              fontSize: 18,
              color: Colors.cyan.shade400,
            ),
          ),
          Padding(
            padding: EdgeInsets.symmetric(horizontal: 20, vertical:
10),
            child: TextFormField(
              controller: _c_reservation_item_list,
              style: TextStyle(color: Colors.cyan.shade400),
              validator: (value) {
                if (value!.isEmpty) return "Amount can't be empty";
                return null;
              },
              maxLines: null,
              decoration: InputDecoration(
                border: OutlineInputBorder(
                  borderSide: BorderSide(width: 2, color:
Colors.cyan.shade400),
                  borderRadius: BorderRadius.circular(11.0),
                ),
                hintText: "QR Link",
                hintStyle: TextStyle(fontSize: 15.0, color:
Colors.cyan.shade400),
                errorText: validate ? null : errorText,
                prefixIcon: Icon(Icons.qr_code_scanner, color:
Colors.cyan.shade400),
                focusedBorder: OutlineInputBorder(
                  borderSide: BorderSide(width: 2, color:
Colors.cyan.shade400),
                  borderRadius: BorderRadius.circular(11.0),
                ),
              ),
            ),
          ),
          SizedBox(height: 20),
          circular
```

```
                ? Center(child: CircularProgressIndicator())
                : Center(
                    child: FormHelper.submitButton(
                      "اعتماد البيانات",
                      () async {
                        if (_c_reservation_item_list.text.isEmpty) {
                          setState(() {
                            validate = false;
                            errorText = "amount Can't be empty";
                          });
                          return;
                        }
                        //save in variable
                        globals.qr_code=_c_reservation_item_list.text;

                        Navigator.push(
                          context,
                          MaterialPageRoute(builder: (context) =>
TimerReservation()),
                        );
                      },
                      btnColor: Colors.cyan.shade400,
                      borderColor: HexColor("#ffffff"),
                      txtColor: HexColor("#ffffff"),
                      borderRadius: 30,
                      width: 355,
                    ),
                  ),
            SizedBox(height: 30),
          ],
        ),
      );
  }
}
```

Appendix F

MAP CODE

```
import 'package:flutter/material.dart';
import 'package:flutter_map/flutter_map.dart';
import 'package:latlong2/latlong.dart';
import 'package:prepaid_parking/top_bottom/bottomBar_static.dart';
import 'package:prepaid_parking/top_bottom/topBar_static.dart';

class MapPage extends StatelessWidget {
  final String latitude;
  final String longitude;

  MapPage({required this.latitude, required this.longitude});

  bool get isLocationDefined => latitude != 0 && longitude != 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: topBar(context),
      bottomNavigationBar: footer(context),
      body: isLocationDefined
          ? FlutterMap(
              options: MapOptions(
                initialCenter: LatLng(double.parse(latitude),
double.parse(longitude)),
                initialZoom: 18.0,
              ),
              children: [
                TileLayer(
                  urlTemplate:
'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
                  subdomains: ['a', 'b', 'c'],
                  userAgentPackageName: 'com.example.app',
                ),
                MarkerLayer(
                  markers: [
                    Marker(
                      point: LatLng(double.parse(latitude),
double.parse(longitude)),
                      width: 40,
                      height: 40,
                      child: Icon(
                        Icons.location_on,
                        color: Colors.red,
                        size: 40,
                      ),
```

```
                    ),
                  ],
                ),
              ],
            )
          : Center(
              child: Text(
                'الموقع غير محدد',
                style: TextStyle(fontSize: 20, color: Colors.red),
              ),
            ),
      );
  }
}
```

Appendix G

WEBSITE DATABASE TABLE

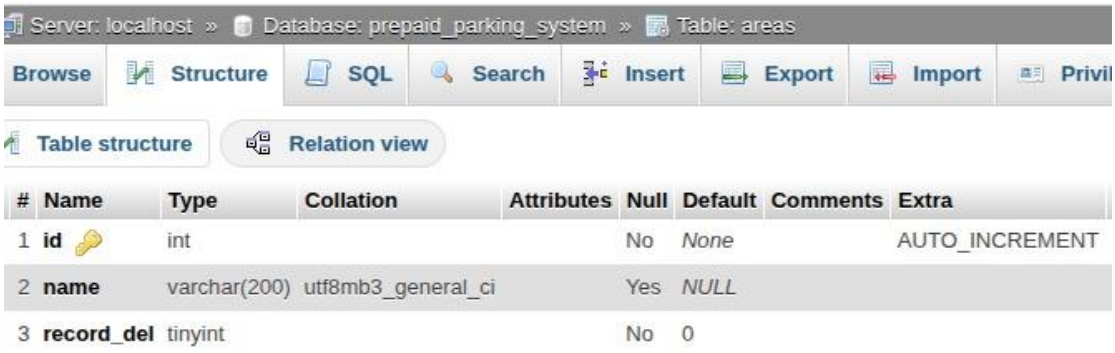
Figure G. 1Database Table

ADMIN TABLE

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int		UNSIGNED	No	None		AUTO_INCREMENT
2	fullname	varchar(100)	utf8mb3_general_ci		Yes	NULL		
3	loginname	varchar(100)	utf8mb3_general_ci		Yes	NULL		
4	password	varchar(100)	utf8mb3_general_ci		Yes	NULL		
5	record_del	tinyint(1)			No	0		

Figure G. 2Admin Table

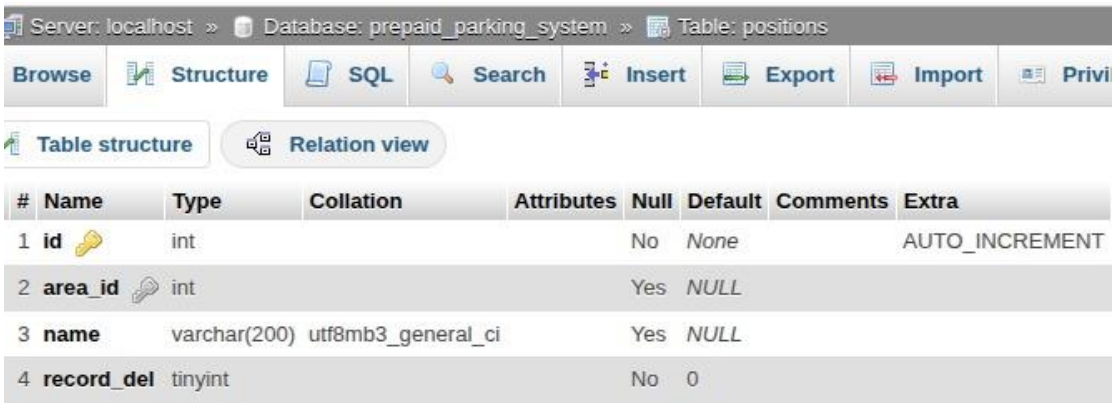
AREA TABLE



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int			No	None		AUTO_INCREMENT
2	name	varchar(200)	utf8mb3_general_ci		Yes	NULL		
3	record_del	tinyint			No	0		

Figure G. 3Area Table

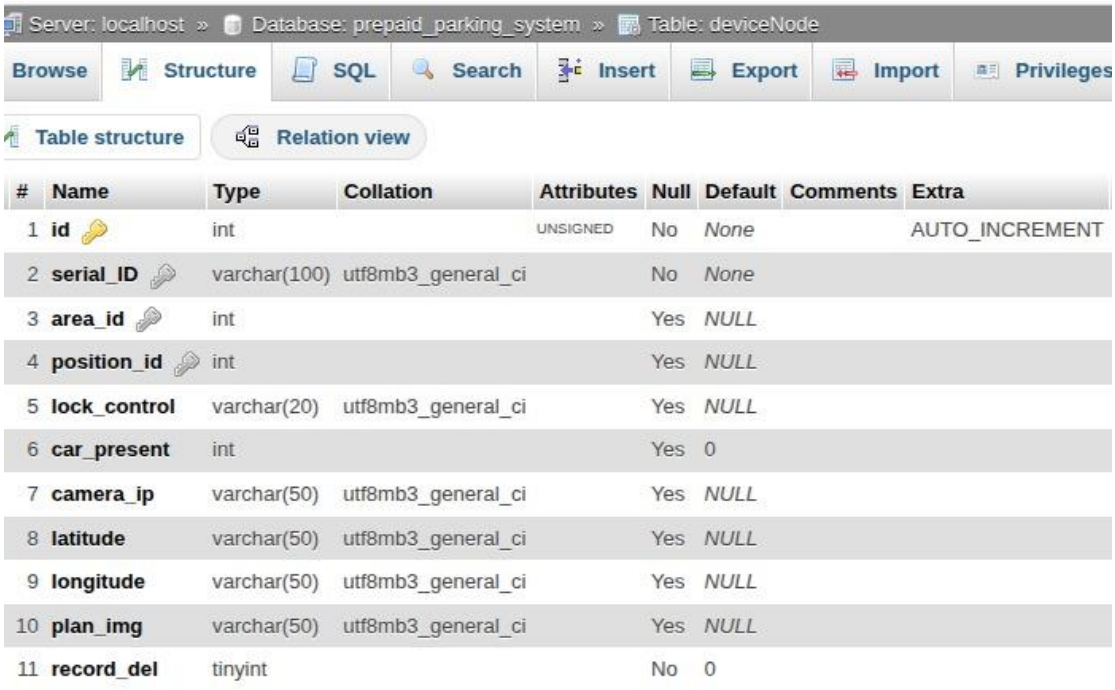
POSITIONS TABLE



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int			No	None		AUTO_INCREMENT
2	area_id	int			Yes	NULL		
3	name	varchar(200)	utf8mb3_general_ci		Yes	NULL		
4	record_del	tinyint			No	0		

Figure G. 4Positions Table

DEVICENODE TABLE



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int		UNSIGNED	No	None		AUTO_INCREMENT
2	serial_ID	varchar(100)	utf8mb3_general_ci		No	None		
3	area_id	int			Yes	NULL		
4	position_id	int			Yes	NULL		
5	lock_control	varchar(20)	utf8mb3_general_ci		Yes	NULL		
6	car_present	int			Yes	0		
7	camera_ip	varchar(50)	utf8mb3_general_ci		Yes	NULL		
8	latitude	varchar(50)	utf8mb3_general_ci		Yes	NULL		
9	longitude	varchar(50)	utf8mb3_general_ci		Yes	NULL		
10	plan_img	varchar(50)	utf8mb3_general_ci		Yes	NULL		
11	record_del	tinyint			No	0		

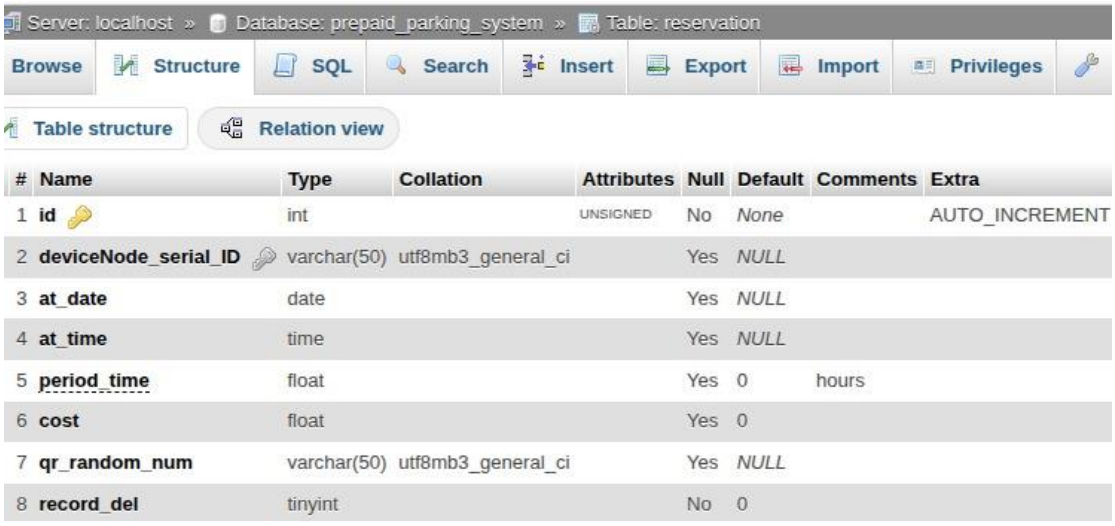
Figure G. 5 DeviceNode Table

LOG_USER_INFO_TABLE

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int			No	None		AUTO_INCREMENT
2	admin_id	int		UNSIGNED	Yes	NULL		
3	ip	varchar(60)	utf8mb3_general_ci		Yes	NULL		
4	browser	varchar(60)	utf8mb3_general_ci		Yes	NULL		
5	date	varchar(60)	utf8mb3_general_ci		Yes	NULL		
6	time	varchar(60)	utf8mb3_general_ci		Yes	NULL		
7	platform	varchar(60)	utf8mb3_general_ci		Yes	NULL		

Figure G. 6 Log_User_Info Table

RESERVATION TABLE



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**id**	int		UNSIGNED	No	*None*		AUTO_INCREMENT
2	**deviceNode_serial_ID**	varchar(50)	utf8mb3_general_ci		Yes	*NULL*		
3	**at_date**	date			Yes	*NULL*		
4	**at_time**	time			Yes	*NULL*		
5	**period_time**	float			Yes	0	hours	
6	**cost**	float			Yes	0		
7	**qr_random_num**	varchar(50)	utf8mb3_general_ci		Yes	*NULL*		
8	**record_del**	tinyint			No	0		

Figure G. 7 Reservation Table