KLE Technological University | Creating Value, Leveraging Knowledge

Dr. M. S. Sheshgiri Campus, Belagavi

Department of
Electronics and Communication Engineering

Minor Project 1

Report

on

Intelligent Vehicle parking system

By:

1. **Bhakti Betageri** USN: 02FE21BEC019
2. **Bhuvan Budavi** USN: 02FE21BEC021
3. **Komal Melavanki** USN: 02FE21BEC042
4. **Praveen Magadum** USN: 02FE21BEC064

Semester: VI, 2023-2024

Under the Guidance of

Prof. Shweta K

KLE Technological University,
Dr. M. S. Sheshgiri College of Engineering and Technology
BELAGAVI-590 008
2023-2024

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that project entitled **"Intelligence vehicle parking system"** is a bonafide work carried out by the student team of **" Bhakti Betageri (02FE21BEC019), Bhuvan Budavi (02FE21BEC021), Komal Melavanki (02FE21BEC042), Praveen M (02FE21BEC064)"**. The project report has been approved as it satisfies the requirements with respect to the mini project work prescribed by the university curriculum for B.E. (VI Semester) in Department of Electronics and Communication Engineering of KLE Technological University Dr. M. S. Sheshgiri CET Belagavi campus for the academic year 2023-2024.

Prof. Shweta K	**Dr. Dattaprasad A. Torse**	**Dr. S. F. Patil**
Guide	**Head of Department**	**Principal**

External Viva:

Name of Examiners	**Signature with date**
1.	
2.	

ACKNOWLEDGMENT

We would like to extend our heartfelt gratitude to our supervisor, Prof. Shweta K, for his invaluable guidance, support, and mentorship throughout the duration of this project. Their expertise, patience, and constructive feedback have been instrumental in shaping the direction of this endeavor and enriching our learning experience. We are sincerely grateful for his dedication and commitment to fostering our academic and professional growth. We would also like to thank our course coordinator Dr. U. L. Naik for their continuous support, guidance, and coordination throughout this project. Their efforts in overseeing the course and ensuring its alignment with our project goals have been invaluable. We would also like to express our appreciation to the Head of the Department, Dr. Dattaprasad A. Torse, for their encouragement, support, and facilitation of resources that have enabled the smooth execution of this project. Their leadership and guidance have played a significant role in ensuring the success of this endeavor. Furthermore, we extend our gratitude to the Principal, Dr. S. F. Patil, for their vision, leadership, and un- wavering support for academic and research pursuits within the institution. Their commitment to excellence and innovation has provided a conducive environment for the development and execution of this project.

We are also thankful to KLE Technological University for providing the necessary infrastructure, facilities, and resources that have been indispensable for the completion of this project. The institution's commitment to academic excellence and research initiatives has been a driving force behind the successful execution of this endeavor. This project would not have been possible without the guidance, support, and encouragement of all individuals and entities mentioned above. Thank you for your invaluable contributions to this endeavor.

-The project team

ABSTRACT

Intelligent parking vehicle detection project presents an Intelligent Vehicle Parking System utilizing an ESP32 microcontroller, an IR sensor, a servo motor, and the Arduino Cloud platform. The ESP32 is programmed to control the servo motor, which manages a barrier arm mechanism. The IR sensor detects vehicle presence, sending signals to the ESP32 to open or close the barrier arm accordingly. When a car approaches, the IR sensor triggers the ESP32 to raise the barrier arm, allowing entry. After the car parks and is no longer detected, the barrier arm lowers. Upon vehicle exit, the sensor signals the ESP32 to raise the barrier arm again, lowering it once the car has left. Integration with Arduino Cloud enables real-time monitoring and remote management, enhancing the system's flexibility and scalability for efficient parking management.

Contents

1 Introduction **9**
- 1.1 Motivation 9
- 1.2 Objectives 9
- 1.3 Literature survey 10
- 1.4 Problem statement 10
- 1.5 Application in Societal Context 10
- 1.6 Project Planning and bill of materials 12
- 1.7 Organization of the report 13

2 System design **16**
- 2.1 Functional diagram 16
- 2.2 Design alternatives 16
- 2.3 Final design 17

3 Implementation details **19**
- 3.1 Specifications and final system architecture 19
  - 3.1.1 Specifications 19
  - 3.1.2 System Architecture 19
- 3.2 Algorithm 20
- 3.3 Flowchart 21

4 Optimization **22**
- 4.1 Introduction to optimization 22
- 4.2 Types of Optimization 22

5 Results and discussions **24**
- 5.1 Results 24
- 5.2 Result Analysis 25
- 5.3 Discussion on optimization 27

6 Conclusions and future scope **28**
- 6.1 Conclusion 28
- 6.2 Future scope 28

References **28**

List of Figures

1.1 block diagram 13

2.1 Functional Block Diagram 16

3.1 Flowchart 21

5.1 Blynk console 24
5.2 Hardware Implemented 25
5.3 Comparison of aspects 27

Chapter 1

Introduction

1.1 Motivation

Developing the Intelligent Vehicle Parking System stems from the increasing need for efficient and automated parking solutions in urban environments. Traditional parking systems often suffer from inefficiencies such as human error, time consumption, and lack of real-time data management. By leveraging the ESP32 microcontroller, IR sensors, and servo motors, this project aims to create a cost-effective, reliable, and scalable solution that automates the entry and exit process in parking spaces. Additionally, integrating with the Arduino Cloud enhances the system's capabilities for real-time monitoring and remote control, addressing the growing demand for smart city infrastructure. This project not only improves the user experience by reducing wait times and manual intervention but also contributes to better resource management and operational efficiency in parking facilities.

1.2 Objectives

- Automation of Parking Space Management: To design and implement a system that automatically controls the entry and exit of vehicles in a parking space using an ESP32 microcontroller, IR sensor, and servo motor. This eliminates the need for manual intervention, reducing human error and improving efficiency.

- Real-Time Monitoring and Control: To integrate the parking system with the Arduino Cloud platform, enabling real-time monitoring and remote management. This allows for seamless oversight and control of the parking operations, enhancing the system's flexibility and scalability.

- Enhanced User Experience and Efficiency: To provide a reliable and user-friendly solution that reduces wait times for drivers, ensures accurate vehicle detection, and optimizes the overall operation of the parking facility. This leads to improved customer satisfaction and better utilization of parking resources.

1.3 Literature survey

[1]. Smart Parking Systems Using IoT and Embedded Systems: Recent studies have focused on leveraging IoT (Internet of Things) and embedded systems to create smart parking solutions. Research indicates that the integration of sensors, microcontrollers, and cloud platforms significantly enhances the efficiency and user experience of parking systems. For instance, systems using Arduino, ESP8266, and other microcontrollers with sensors have been shown to automate vehicle detection and parking space management effectively .

[2]. IR Sensors for Vehicle Detection: Infrared (IR) sensors are widely used in smart parking systems for their ability to accurately detect vehicle presence. Literature reviews suggest that IR sensors provide a cost-effective solution for monitoring parking spaces. They can reliably detect vehicles by sensing the change in infrared radiation, which triggers automated processes such as opening or closing barrier arms. Studies demonstrate that IR sensors are effective in various environmental conditions, making them a practical choice for parking systems .

[3]. Microcontroller-Based Automation: The use of microcontrollers like ESP32 and Arduino in parking systems has been extensively explored. These microcontrollers are chosen for their processing power, versatility, and ease of integration with other components such as sensors and motors. Research highlights the effectiveness of these microcontrollers in automating parking operations, controlling barrier mechanisms, and communicating with cloud platforms for real- time data management. Projects using ESP32 have demonstrated its capability in handling multiple inputs and outputs, crucial for complex parking systems .

[4]. Cloud Integration for Smart Parking: Integrating cloud platforms with parking systems offers numerous advantages, including real-time monitoring, remote control, and data analytics. Literature emphasizes the importance of cloud connectivity in enhancing the functionality and scalability of parking solutions. By using platforms like Arduino Cloud, systems can provide features such as occupancy tracking, remote diagnostics, and user notifications. Studies show that cloud-based systems improve resource management, reduce operational costs, and offer better user services compared to traditional parking systems .

1.4 Problem statement

Design and implement a vehicle presence detection system utilizing a ESP 32 Microcontroller board to control entry barriers in parking lots.

1.5 Application in Societal Context

1. Urban Traffic Management: Intelligent vehicle parking systems can significantly alleviate traffic congestion in urban areas by streamlining the parking process. Efficient parking solutions reduce the time drivers spend searching for parking spots, leading to decreased traffic flow and lower emissions. This contributes to a more sustainable urban environment and improved air quality.

2. Enhanced Security and Safety: Automated parking systems improve security in parking facilities by regulating vehicle entry and exit. The integration of sensors and microcontrollers ensures that only authorized vehicles can access the parking area, reducingthe risk of theft and unauthorized access. Additionally, real-time monitoring via cloud platforms can provide immediate alerts in case of any security breaches or anomalies.

3. Convenience and Accessibility: The implementation of smart parking systems offers enhanced convenience for users by providing real-time information on parking availability and reducing wait times. This is particularly beneficial in busy commercial areas, hospitals, and airports where finding parking quickly is essential. Furthermore, the system can be designed to include features that assist disabled drivers, ensuring accessible parking options are available.

4. Economic Efficiency: Smart parking systems can lead to significant cost savings for both operators and users. For operators, automation reduces the need for manual labor and lowers operational costs. For users, efficient parking management translates to less fuel consumption and time savings. Additionally, businesses in the vicinity of well-managed parking facilities may see increased patronage as easier parking improves customer satisfaction.

5. Data-Driven Urban Planning: The data collected by intelligent parking systems can pro- vide valuable insights for urban planners and policymakers. Analyzing parking patterns and usage data helps in making informed decisions about infrastructure development, zoning, and transportation policies. This data-driven approach supports the creation of smarter, more livable cities.

6. Environmental Impact: By reducing the time vehicles spend idling or circling to find parking, intelligent parking systems contribute to lower greenhouse gas emissions. This aligns with broader environmental goals and supports initiatives aimed at combating climate change. Efficient use of parking spaces also promotes sustainable land use, minimizing the need for extensive parking infrastructure.

1.6 Project Planning and bill of materials

- **Project scope:** The future scope of the Intelligent Vehicle Parking System project is extensive, with numerous opportunities for enhancement and expansion. One potential direction is the integration of advanced technologies such as AI and machine learning to improve vehicle detection accuracy and predict parking space availability based on historical data and usage patterns. Incorporating additional sensors like ultrasonic or LIDAR could further refine the system's capability to manage complex parking environments. Furthermore, expanding the system's connectivity through IoT advancements can enable seamless integration with smart city infrastructure, facilitating comprehensive traffic management and urban planning. The development of a mobile application interface can provide users with real-time updates, reservation capabilities,and personalized parking recommendations.

Technical Requirements:

1. Hardware Components:

- ESP32 Microcontroller: Central control unit for processing signals and managing operations.
- IR Sensors: For detecting the presence of vehicles at entry and exit points.
- Servo Motor: To control the barrier arm for opening and closing.
- Barrier Arm: Mechanism for regulating vehicle entry and exit.
- Power Supply: Suitable power source for ESP32, servo motor, and sensors.
- Connecting Wires and Breadboard: For connecting components and prototyping the circuit.
- Mounting Hardware: Brackets, screws, and other hardware for installing the sensors and barrier arm.

2. Software Requirements:

- Arduino IDE: For programming the ESP32 microcontroller.
- Arduino Cloud Account: For real-time monitoring and control of the system via the cloud.
- Libraries for ESP32: Such as WiFi, Servo, and other relevant libraries for sensor and motor control.

Project steps:

- Connect the ESP32 to the IR sensor and the servo motor.
- Install the Arduino IDE on your computer.
- Create an Arduino Cloud account.
- Install the Arduino Cloud library in the Arduino IDE.
- Download the code for the smart car parking system from GitHub.
- Open the code in the Arduino IDE.
- In the code, enter your Arduino Cloud credentials.
- Upload the code to the ESP32.
- Open the Arduino Cloud dashboard.
- Click on the “Parking” tab.
- You should see the status of the parking spaces in real time.
-

1.7 Organization of the report

block diagram:

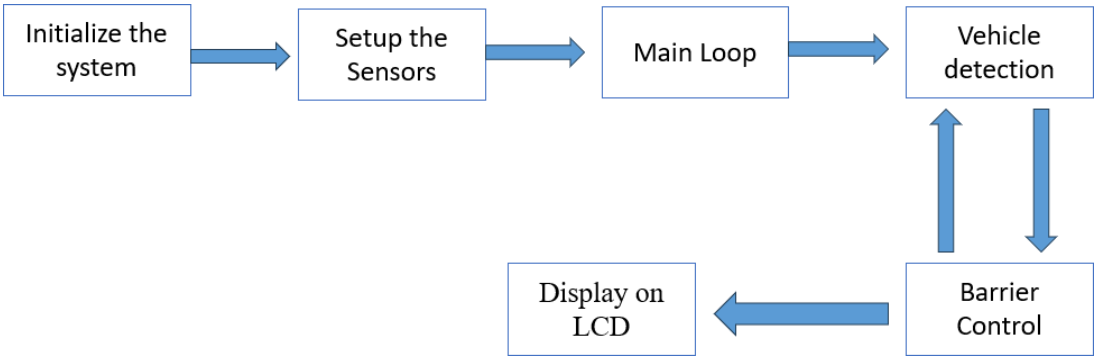


Figure 1.1: block diagram

Here is the block diagram for the Intelligent Vehicle Parking System project:

- IR Sensor: Detects the presence of a car and sends a signal to the ESP32.
- ESP32: Receives signals from the IR sensor and controls the servo motor. It also communicates with the Arduino Cloud for real-time monitoring and remote management.

- Servo Motor: Activated by the ESP32 to open or close the barrier arm.
- Barrier Arm: Mechanically operated by the servo motor to regulate vehicle entry and exit.
- Arduino Cloud: Provides real-time monitoring and remote control capabilities.

Design alternatives:

1. Basic Standalone System:
   - Single Microcontroller: Use only the ESP32 without Arduino Cloud.
   - Local Control: ESP32 directly controls the servo and communicates with the IR sensor.
   - No Cloud Connectivity: Eliminate the Arduino Cloud account.
   - Simple Logic: IR sensor triggers the servo directly without cloud interaction.
2. ESP32 with Local Web Server:
   - ESP32 Web Server: ESP32 hosts a web server to control the barrier.
   - WiFi Connectivity: Allows controlling the barrier through a web interface.
   - IR Sensor Integration: IR sensor triggers the barrier locally.
   - No Cloud Dependency: Doesn’t rely on Arduino Cloud.

Final design:

The final design of the parking barrier system utilizes an ESP32 microcontroller board integrated with an IR sensor and a servo motor for barrier control. The system operates independently without relying on cloud services for enhanced reliability. When a car approaches, the IR sensor detects its presence and triggers the ESP32 to open the barrier using the servo motor. Similarly, when the car leaves, the IR sensor signals the ESP32 to close the barrier. This design ensures efficient and real-time vehicle detection and barrier control locally.

Implementation design:

- The implementation design of the parking barrier system involves several key components and interactions. The ESP32 microcontroller serves as the central control unit. Upon detecting a vehicle approaching or leaving, the IR sensor sends signals to the ESP32.
- The ESP32 then processes these signals and controls the servo motor to open or close the barrier arm accordingly. The system can be implemented using the Arduino IDE and libraries for ESP32 development. Communication between the ESP32 and the IR sensor is achieved through digital input/output pins, while the servo motor is controlled via PWM signals.
- The system can be powered either by a battery or a power adapter. Additionally, the implementation can include features like error handling, power-saving modes, and integration with external interfaces such as Bluetooth or Wi-Fi for remote monitoring and control. The code can be structured to ensure modularity, readability, and flexibility for future enhancements or modifications. Testing and calibration procedures should be included to ensure proper functionality of the system once deployed.

Chapter 2

System design

2.1 Functional diagram

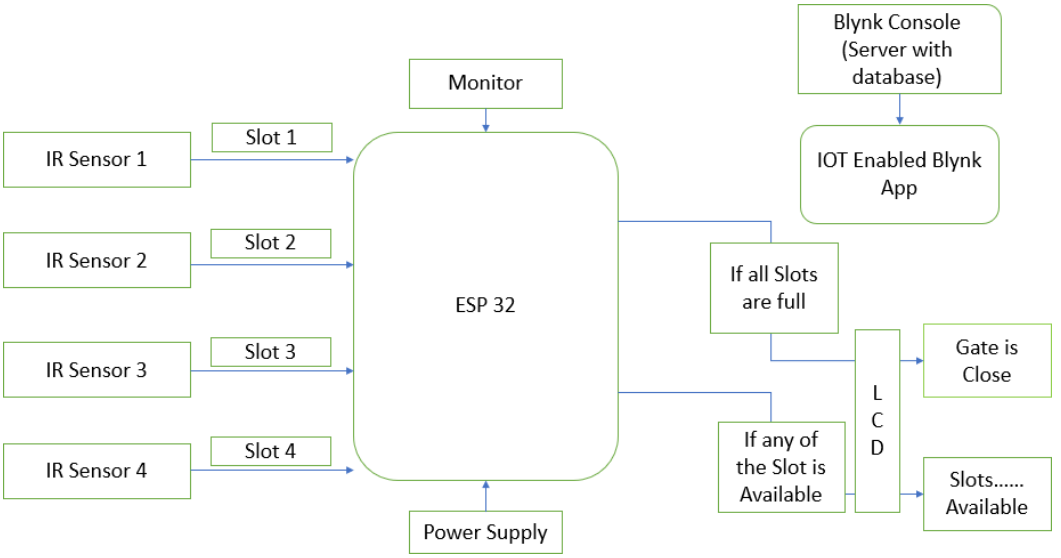


Figure 2.1: Functional Block Diagram

2.2 Design alternatives

1. Local Control with ESP32: Keep the control entirely local without relying on cloud services. ESP32 can directly control the servo motor based on IR sensor inputs. This eliminates the dependency on an Arduino Cloud account, making the system more inde- pendent.

2. WiFi Communication: Instead of Arduino Cloud, use WiFi communication with the ESP32 to connect to a local server or a custom-built web interface. This allows remote monitoring and controlof the barrier, providing flexibility without relying on a third-party cloud service.

3. Bluetooth Control: Utilize Bluetooth communication instead of WiFi, allowing users to control the barrier arm using a smartphone app. This eliminates the need for internet connectivity and provides a more direct control method.

4. MQTT Protocol: Implement MQTT protocol for communication, enabling the system to be part of a larger IoT network. This can facilitate communication with other devices and services, offering scalability and interoperability.

5. Offline Mode with Manual Override: Implement a manual override mechanism or physical button in case of network failure or sensor malfunction, ensuring that the barrier can still be controlled locally even without network connectivity.

6. Integration with Other Sensors: Besides IR sensor, consider using alternative sensors like ultrasonic sensors, magnetic sensors, or cameras for vehicle detection, providing redun- dancy and improving accuracy, especially in challenging weather conditions.

7. Mobile Network Integration: If remote monitoring and control are crucial, integrate a SIM module to enable communication over the mobile network, allowing users to control the barrier from anywhere via SMS or a dedicated mobile app.

8. Cloud Integration for Data Logging: Use cloud services like AWS IoT, Google Cloud IoT, or Azure IoT for data logging and analytics purposes while retaining local control for immediate actions, providing insights into parking space usage over time.

9. Voice Control Interface: Implement a voice control interface using services like Alexa or Google Assistant, allowing users to open/close the barrier using voice commands, which can be convenient in certain scenarios.

10. Solar Power Integration: Incorporate solar panels and batteries to power the system, mak- ing it self-sustainable and environmentally friendly, especially if the installation location lacks easy access to power sources.

2.3 Final design

Working Principle:

1. Initialization: The ESP32 microcontroller initializes upon power-up or reset. It sets up the necessary peripherals including the IR sensor and servo motor.

2. Vehicle Detection: The IR sensor continuously monitors the parking space. When a car approaches, it detects the presence of the vehicle based on infrared radiation. The sensor sends a signal to the ESP32 indicating the presence of the car.

3. Barrier Opening: Upon receiving the signal from the IR sensor, the ESP32 activates the servo motor. The servo motor is controlled to open the barrier arm, allowing the car to enter the parking space safely.

4. Monitoring: While the car is parked, the system remains in a monitoring state. The IR sensor continues to detect if the car is present.

Ease of Implementation:

- Hardware Integration: The hardware components, including the ESP32, IR sensor, and servo motor, are relatively easy to connect since they often come with clear documentation and readily available libraries. Wiring connections and mounting the components can be straightforward with basic electronics knowledge.
- Programming: Programming the ESP32 to control the servo motor based on IR sensor input is moderately complex but manageable, especially with resources and example codes available online. Libraries for servo motor control and IR sensor interfacing simplify the task, making it achievable for beginners with some programming experience.
- Testing and Debugging: Testing the system requires setting up a test environment mimicking real-world conditions. Debugging issues related to sensor readings, motor control, and communication can take some time but is feasible with thorough testing and troubleshooting techniques.
- Power Supply: Powering the system might need consideration, but using a standard power source for the ESP32 and servo motor is typically uncomplicated. Battery-powered or solar-powered options might require additional considerations but can be implementedwith proper planning.
- Mechanical Design: Designing the physical barrier mechanism for the servo motor to actuate may vary in complexity depending on the specific requirements of the barrier arm. However, basic mechanical designs can be implemented without extensive expertise.
- Integration with Arduino Cloud: Integrating with Arduino Cloud for remote monitoring adds some complexity, requiring setup of the cloud account and handling communication protocols. However, Arduino provides documentation and examples that ease the integration process.
- Scalability and Expansion: The project can be expanded with additional features like mobile app control, data logging, or integration with other IoT devices, which may increase complexity but also provide opportunities for customization based on requirements.
- Overall Complexity: While individual components are relatively easy to work with, the overall complexity may increase when integrating all components seamlessly. However, with step-by-step planning, starting with basic functionality and then expanding features gradually, the project is achievable for hobbyists and beginners in electronics and programming.

Considering these factors, the project offers a moderate level of ease of implementation for those withbasic electronics and programming skills, and it can provide a good learning experience for beginnerswhile allowing room for expansion and customization.

Chapter 3

Implementation details

3.1 Specifications and final system architecture

3.1.1 Specifications

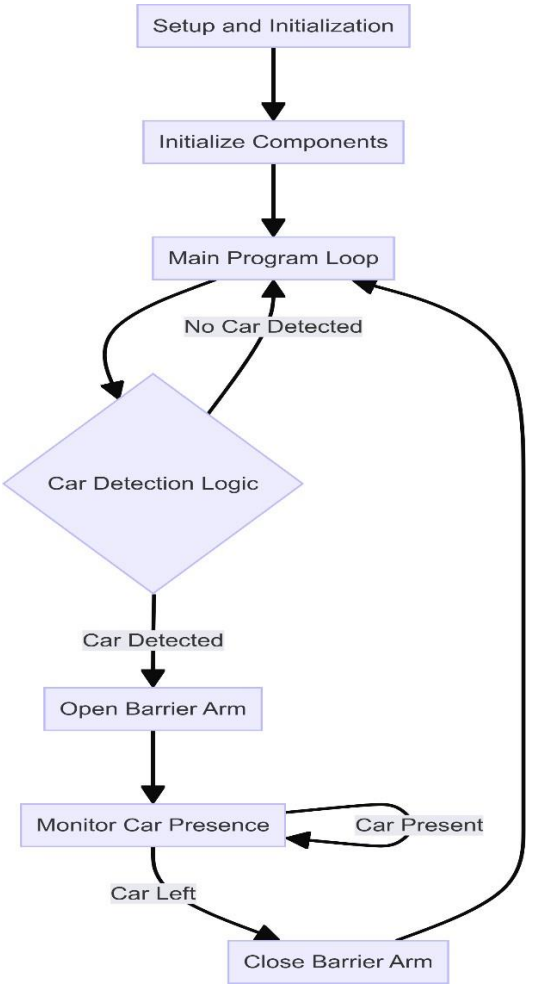
1. ESP32 Microcontroller:
   - Specifications: Dual-core Tensilica processor, WiFi and Bluetooth connectivity, ample GPIO pins for sensor and motor connections, and sufficient processing power for control tasks.
   - Role: Central control unit responsible for interfacing with the IR sensor, servo motor, and optionally with external services like Arduino Cloud.
2. IR Sensor:
   - Specifications: Infrared proximity sensor capable of detecting objects within a certain range.
   - Role: Detects the presence or absence of vehicles in the parking space by measuring infrared radiation. Sends signals to the ESP32 upon detection.
3. Servo Motor:
   - Specifications: Standard servo motor with torque and speed suitable for operating the barrier arm.
   - Role: Actuates the barrier arm to open and close based on commands received from the ESP32.

3.1.2 System Architecture

- Sensor Layer: IR sensor detects vehicles and sends signals to the ESP32.
- Control Layer: ESP32 processes sensor inputs and controls the servo motor to open/close the barrier arm.
- Communication Layer: Optionally connects to Arduino Cloud for remote monitoring and control.
- Power Supply: Typically powered by a stable DC power source, though alternative power options like battery or solar power can be implemented.

- Main Control Program: Written for ESP32, it manages sensor inputs, controls the servo motor, and handles communication if needed.
- Sensor Interface: Code to read data from the IR sensor and interpret vehicle presence.
- Motor Control: Code to control the servo motor's position to open and close the barrier arm.
- Cloud Interface (if used): Code to communicate with Arduino Cloud for remote access.
- Implement safety measures to prevent accidents, such as emergency stop mechanisms or obstacle detection to avoid closing the barrier arm on a vehicle.

3.2 Algorithm



3.3 Flowchart

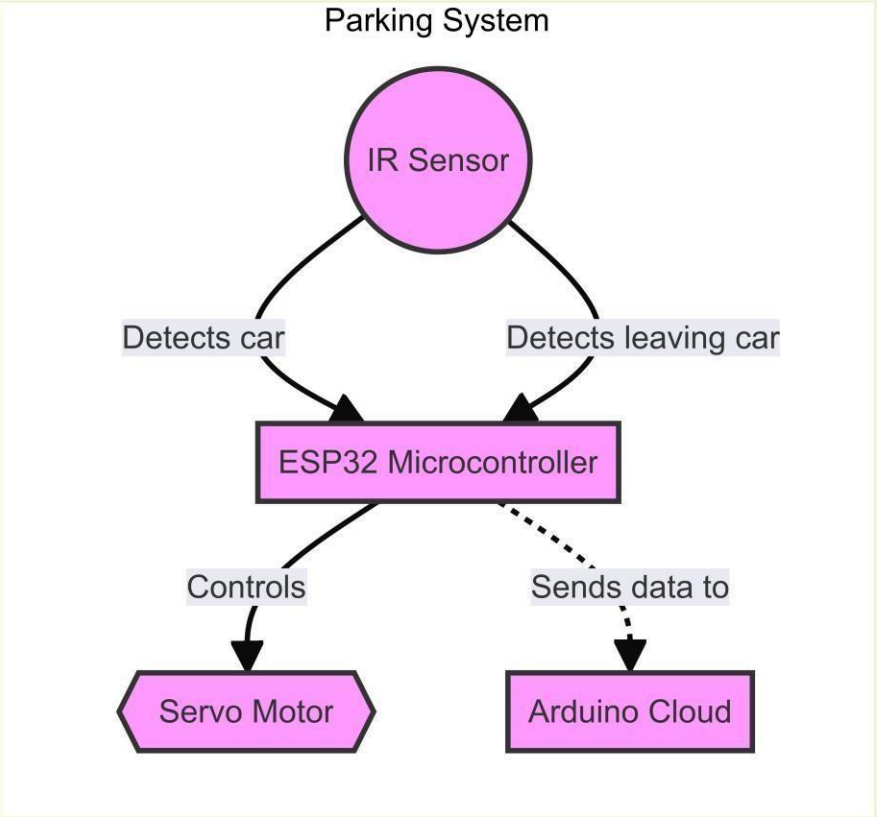


Figure 3.1: Flowchart

- parking system that uses an ESP32 microcontroller to control a servo motor and send data to the Arduino Cloud. Here's a breakdown of how it works:
- IR Sensor: This sensor detects the presence of a car. It likely uses infrared (IR) light, which is invisible to the human eye. When a car enters or leaves the parking space, the sensor sends a signal to the ESP32 microcontroller.
- ESP32 Microcontroller: This is the brain of the system. It receives signals from the IR sensor and controls the servo motor based on that information. The ESP32 microcontroller is also likely programmed to send data about the parking space to the Arduino Cloud.
- Servo Motor: This motor is used to control a barrier or gate at the entrance of the parking space. When the IR sensor detects a car entering the parking space, the ESP32 microcontroller sends a signal to the servo motor to open the gate. Conversely, when a car leaves the parking space, the servo motor closes the gate.
- Arduino Cloud: This is an Internet of Things (IoT) platform that allows the ESP32 microcontroller to send and receive data over the internet. The data from the parking space sensor could be used to create a real-time parking availability app, for example.

Chapter 4

Optimization

4.1 Introduction to optimization

Optimizing the project involves enhancing efficiency, reliability, and resource utilization to ensure smooth operation and better performance. In this project, optimization can focus on several aspects such as improving response time, minimizing power consumption, maximizing sensor accuracy, and enhancing system robustness. By optimizing the code, hardware, and system architecture, we aim to create a parking barrier system that operates seamlessly, conserves energy,and provides accurate detection of vehicles while ensuring timely barrier control. Optimization efforts will not only improve the overall effectiveness of the system but also contribute to its longevity and scalability, making it more adaptable to varying environmental conditions and user requirements.

4.2 Types of Optimization

1. Power Optimization:
   - Implementing sleep modes or low-power states to conserve energy when the system is idle.
   - Optimizing hardware components and software algorithms to minimize power consumption.
   - Using efficient power sources such as solar panels or low-power components where applicable.
2. Response Time Optimization:
   - Streamlining sensor data processing and motor control algorithms to reduce latency.
   - Minimizing delays in vehicle detection and barrier activation for faster response.
   - Utilizing interrupts or parallel processing to improve real-time responsiveness.
3. Sensor Optimization:
   - Calibrating sensors for optimal performance and accuracy in detecting vehicles.
   - Implementing noise filtering techniques to reduce false readings and improve reliability.
   - Using sensor fusion techniques if multiple sensors are employed for better accuracy.

4. Code Optimization:

- Writing efficient and optimized code for the microcontroller to reduce execution time and memory usage.
- Employing algorithmic optimizations and minimizing unnecessary loops or computations.
- Utilizing hardware features and specialized instructions for performance improvement where available.

5. Hardware Optimization:

- Selecting appropriate components with the right specifications to meet project requirements without overkill.
- Optimizing hardware layout and connections for better signal integrity and reliability.
- Using hardware peripherals or dedicated ICs for specific tasks to offload processing from the microcontroller.

Chapter 5

Results and discussions

5.1 Results

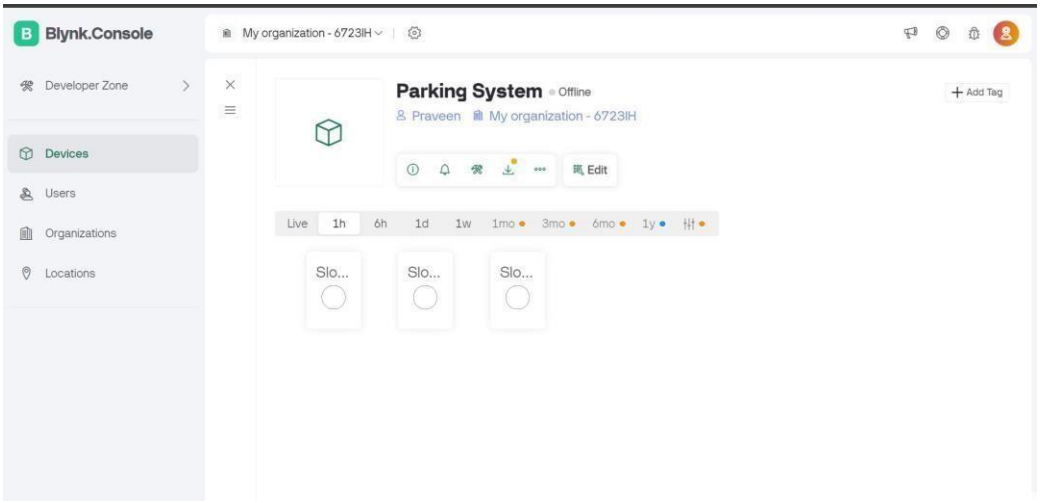


Figure 5.1: Blynk console

The dashboard shows various sections including Devices, Users, Organizations, and Locations. It also shows a lot number, “8 Praveen,” which could be a user or device ID. The dashboard says “Parking System. Offline,” which means the system is not currently connected to the internet.

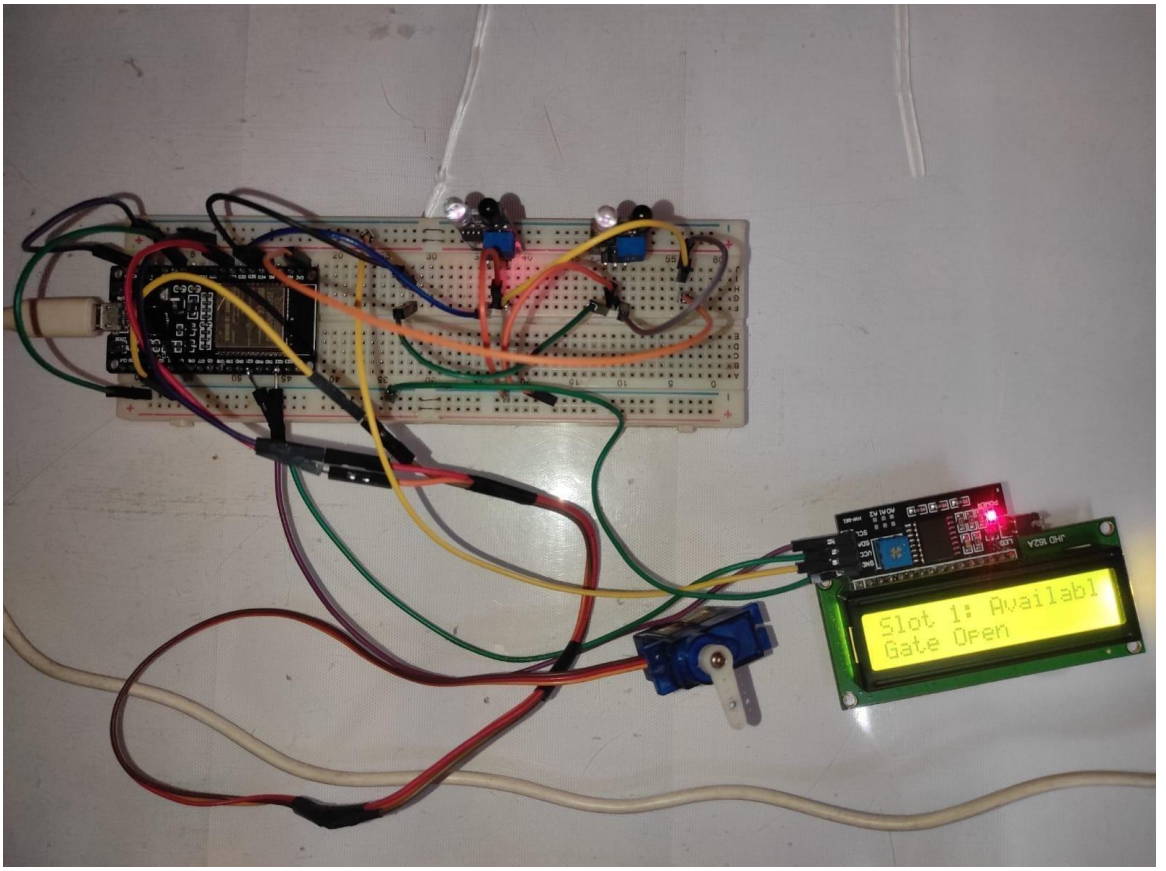


Figure 5.2: Hardware Implemented

5.2 Result Analysis

1. System Performance:
   - Evaluate the overall performance of the system in terms of response time from vehicle detection to barrier activation and closure.
   - Measure the reliability of vehicle detection by the IR sensor under various lighting and environmental conditions.
   - Assess the accuracy of barrier arm movement controlled by the servo motor.
2. Response Time Analysis:
   - Measure the average response time from vehicle detection to barrier opening and closing.
   - Analyze any delays and identify potential causes such as sensor processing time or motor control latency.
   - Optimize system components and algorithms to minimize response time where necessary.
3. Accuracy of Vehicle Detection:

- Determine the accuracy of vehicle detection by comparing sensor readings with actual vehicle presence.
- Analyze false positives or false negatives and adjust sensor parameters or algorithms accordingly.
- Validate the sensor's performance over time and in different environmental conditions.

4. Reliability and Robustness:
   - Assess the system's reliability by testing its performance over an extended period, including continuous operation and under varying conditions.
   - Conduct stress tests to identify potential failure points and weaknesses in the system.
   - Evaluate fault tolerance mechanisms and how well the system handles unexpected situations.
5. Cloud Integration Analysis:
   - If Arduino Cloud is used, analyze the reliability and responsiveness of remote monitoring and control features.
   - Evaluate the effectiveness of cloud communication in providing real-time status updates and notifications.
   - Assess the security measures implemented for cloud connectivity to ensure data privacy and integrity.
6. User Experience:
   - Gather feedback from users regarding the usability and convenience of the parking barrier system.
   - Evaluate any issues encountered during installation, operation, or maintenance.
   - Identify areas for improvement to enhance user experience, such as adding manual override options or improving feedback mechanisms.

5.3 Discussion on optimization

Aspect	LPC1768	ESP32
Processing Power	Moderate	Higher
Connectivity	Limited (requires additional module)	Built-in WiFi and optional Bluetooth
GPIO Pins	Limited, may need expansion boards	Abundant, more flexibility
Memory	Limited Flash and RAM	More Flash and RAM
Cost	Usually lower	Slightly higher

Figure 5.3: Comparison of aspects

The table compares two microcontroller boards, LPC1768 and ESP32, across five key aspects relevant to a parking barrier project.

- Processing Power: LPC1768 offers moderate processing power, while ESP32 provides higher processing power.
- Connectivity: LPC1768 has limited connectivity, usually requiring additional modules, whereas ESP32 has built-in WiFi and optional Bluetooth capabilities.
- GPIO Pins: LPC1768 has limited GPIO pins and may require expansion boards, while ESP32 offers abundant GPIO pins for more flexibility.
- Memory: LPC1768 has limited Flash and RAM compared to ESP32, which offers more Flash and RAM.
- Cost: LPC1768 is usually lower in cost compared to ESP32, which tends to be slightly higher.

In summary, while LPC1768 may be sufficient for basic applications with lower processing and connectivity requirements, ESP32 offers more power, built-in connectivity options, abundant GPIO pins, more memory, albeit at a slightly higher cost. The choice between the two depends on the specific requirements and budget of the parking barrier project.

Chapter 6

Conclusions and future scope

6.1 Conclusion

In conclusion, implementing a parking barrier project using either LPC1768 or ESP32 microcontroller boards offers distinct advantages. The choice depends on factors such as processing power, connectivity needs, GPIO requirements, memory, and budget constraints. Both microcontrollers can effectively control the barrier arm using an IR sensor and servo motor. However, ESP32 stands out with its higher processing power, built-in WiFi and optional Bluetooth connectivity, abundant GPIO pins, and more memory, making it well-suited for advanced features and IoT applications. Additionally, ESP32 provides easier integration with platforms like Arduino Cloud, simplifying remote monitoring and control. Overall, while LPC1768 may suffice for simpler applications, ESP32 offers greater capabilities and versatility for more sophisticated parking barrier systems, providing scalability and potential for future expansions.

6.2 Future scope

The parking barrier project has several future scopes for enhancement and expansion. One possibilityis to integrate advanced features such as:

1. Smart Parking Management: Implementing a centralized system to monitor multiple parking spaces, providing real-time data on parking availability and occupancy.
2. Vehicle Detection and Recognition: Enhance the system with advanced vehicle detection and recognition technology, possibly using cameras or RFID for automatic identification of vehicles.
3. Payment Integration: Integrate payment systems to allow for automated parking fee collection, supporting cashless transactions through mobile apps or RFID cards.
4. IoT Connectivity: Expand IoT capabilities to enable remote monitoring and management of the parking system, allowing administrators to control barriers and access data from anywhere.
5. Integration with Navigation Apps: Integrate with navigation applications to provide users with real-time information on parking availability and navigation to vacant spots.
6. Security Enhancements: Implement additional security features such as surveillance cameras, automatic alerts for suspicious activities, and access control systems.

Bibliography

[1] Suresh S., J. Bhavya, S. Sakshi, K. Varun and G. Debarshi, "Home Monitoring and Security system," 2016 International Conference on ICT in Business Industry Government (ICTBIG), 2016, pp. 1-5, doi: 10.1109/ICTBIG.2016.7892665

[2] C. Visvesvaran, S. Kamalakannan, K. N. Kumar, K. M. Sundaram, S. M. S. S. Vasan and S. Jafrrin, "Smart Greenhouse Monitoring System using Wireless Sensor Networks," 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), 2021, pp. 96- 101, doi: 10.1109/ICOSEC51865.2021.9591680.

[3] S. P. Parikh, D. R. Shah and P. R. Shah, "Park Indicator: Book Your Parking Spot," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-3, doi: 10.1109/ICCUBEA.2018.8697378.

[4] Anand, A. Kumar, A. N. M. Rao, A. Ankesh and A. Raj, "Smart Parking System (S- Park) – A Novel Application to Provide Real-Time Parking Solution," 2020 Third International Conference on Multimedia Processing, Communication Information Technology (MPCIT), 2020,pp. 93-96, doi: 10.1109/MPCIT51588.2020.9350429.