SMART CAR PARKING SYSTEM

A PROJECT REPORT

Submitted by

PISINI JOEL(22BCS12519)
KANISHK RAGOLU (22BCS16567)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

Chandigarh University

JUNE 2024

BONAFIDE CERTIFICATE

Certified that this project report **"SMART CAR PARKING SYSTEM"** is the bonafide work of "**PISINI JOEL(22BCS12519), KANISHK RAGOLU (22BCS16567)"** who carried out the project work under my supervision.

SIGNATURE

Prof (Dr) Sandeep Singh Kang

HEAD OF THE DEPARTMENT

SIGNATURE

Er. Kussum

TRAINER

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER

Er.Kussum

Praveen Badoni

EXTERNAL EXAMINER

TABLE OF CONTENTS

LIST OF FIGURES:

INTRODUCTION

1.1 Identification of Client /Need / Relevant Contemporary issue.

Client

1. **Parking Lot Owners/Operators**:
    - **Commercial Parking Lots**: Operators of paid parking lots need efficient management systems to maximize space utilization and improve customer experience.
    - **Shopping Malls**: Malls often face high traffic, especially during peak hours, making efficient parking management crucial for customer satisfaction.
    - **Hospitals**: Hospitals require organized parking solutions to ensure patients, visitors, and staff can quickly find parking spots, reducing stress and delays.
2. **Residential Complex Managers**:
    - **Apartment Complexes**: With limited parking spaces, these complexes need systems to ensure fair and efficient allocation of parking slots to residents and visitors.
    - **Gated Communities**: These communities benefit from automated systems that enhance security and manage parking for residents and guests.
3. **Urban Planners**:
    - **City Planners**: Integrating smart parking solutions helps in managing urban traffic congestion and optimizing the use of available space in densely populated areas.
    - **Public Parking Spaces**: Municipal authorities can use automated systems to manage public parking spaces efficiently, improving accessibility and reducing congestion.
4. **Tech Startups/Developers**:
    - **Smart City Solutions**: Startups focusing on IoT and smart city solutions can integrate automated parking systems as part of their offerings, providing comprehensive urban management solutions.
    - **Developers of Automated Systems**: Companies developing home automation or commercial automation systems can expand their portfolio by including smart parking solutions.

Need

1. **Efficient Space Utilization**:
    - **Optimization**: Accurate monitoring of parking slots ensures that every available space is utilized, preventing wastage and maximizing revenue for parking lot operators.
    - **Scalability**: The system can be scaled to manage large parking areas, making it suitable for various sizes of facilities.

2. **Automation**:
    - **Reduced Human Intervention**: Automating the entry and exit process reduces the need for staff, cutting operational costs and minimizing errors.
    - **24/7 Operation**: Automated systems can operate continuously without the need for breaks, providing consistent service.
3. **Real-time Information**:
    - **Dynamic Updates**: Real-time updates on parking availability help drivers make informed decisions quickly, improving traffic flow and reducing the time spent searching for parking.
    - **Integration with Apps**: The system can be integrated with mobile apps, allowing users to check parking availability before they arrive.
4. **Security**:
    - **Controlled Access**: Automated gates controlled by the system ensure only authorized vehicles enter and exit, enhancing the security of the parking area.
    - **Monitoring**: Continuous monitoring of parking slots helps in detecting and addressing unauthorized use of spaces.
5. **User Convenience**:
    - **Clear Communication**: The LCD display provides clear and concise information about parking availability, making it easy for users to find and park their vehicles.
    - **Reduced Frustration**: By providing real-time updates and reducing search times, the system enhances the overall user experience.

Relevant Contemporary Issue

1. **Urban Traffic Congestion**:
    - **Reduction in Search Time**: Efficient parking management systems reduce the time drivers spend searching for parking, which in turn reduces congestion on city streets.
    - **Improved Traffic Flow**: By guiding drivers directly to available parking spots, the system helps in maintaining a smoother flow of traffic in urban areas.
2. **Environmental Concerns**:
    - **Lower Emissions**: Reducing the time vehicles spend idling or circling to find parking spots can significantly lower emissions, contributing to environmental sustainability.
    - **Green Initiatives**: Automated parking systems align with green city initiatives aimed at reducing carbon footprints and promoting eco-friendly urban infrastructure.
3. **Smart Cities Initiatives**:
    - **Integration with IoT**: Automated parking systems are a key component of smart city initiatives, where interconnected devices provide real-time data to optimize urban living conditions.
    - **Data Analytics**: These systems can collect data on parking usage patterns, which can be used for urban planning and improving infrastructure.
4. **Touchless Solutions Post-COVID-19**:
    - **Hygiene and Safety**: By minimizing human interaction, automated parking systems contribute to safer and more hygienic environments, which is critical in the post-pandemic world.

- **Minimized Contact**: The demand for touchless solutions has increased due to the COVID-19 pandemic. Automated systems reduce physical contact points, thereby enhancing safety.

5. **Efficiency in High-demand Areas**:
   - **High Turnover Management**: In areas with high vehicle turnover, such as shopping malls and hospitals, automated systems ensure efficient management of parking spaces, reducing wait times and improving user satisfaction.
   - **Revenue Maximization**: For commercial parking operators, efficient space utilization directly translates to increased revenue, as more vehicles can be accommodated in the same space.

1.2 IDENTIFICATION OF PROBLEM

Overview

In modern urban environments, parking management presents significant challenges due to the increasing number of vehicles. Traditional parking systems often lack the capability to efficiently monitor and manage parking spaces, leading to various problems such as traffic congestion, environmental impact, and user dissatisfaction. The need for a sophisticated and automated solution has become crucial to address these issues effectively. This project aims to create an automated car parking system using IR proximity sensors, an Arduino microcontroller, a 20x4 LCD display with an I2C interface, and an SG90 servo motor to enhance parking management efficiency.

Specific Problems

1. **Parking Space Utilization**
   - **Inefficient Use of Space**: Traditional parking lots often do not utilize available space efficiently due to the absence of real-time monitoring systems. This results in either underutilized spaces or overfilled lots where finding a spot becomes challenging.
     - **Impact**: Leads to financial losses for parking lot operators and inconvenience for users.
     - **Solution**: Implementing IR proximity sensors at each parking slot to detect occupancy and provide real-time updates on the status of each slot.
   - **Slot Availability Tracking**: Difficulty in accurately tracking which parking slots are occupied and which are available.
     - **Impact**: Causes mismanagement of parking resources and delays for drivers trying to find available spots.
     - **Solution**: Use of sensors to monitor each slot and a centralized system to update the availability status on an LCD display.
2. **Traffic Congestion**
   - **Search Time**: Drivers spend excessive time searching for available parking spots, which contributes significantly to urban traffic congestion.
     - **Impact**: Increased travel time, driver frustration, and reduced efficiency of urban traffic flow.
     - **Solution**: Real-time information on parking availability displayed at the entrance to guide drivers directly to available slots.

- **Entry and Exit Delays**: Manual processes at parking lot entry and exit points cause delays, leading to backups and further congestion.
  - **Impact**: Reduces throughput of the parking facility and increases congestion at entry and exit points.
  - **Solution**: Automating entry and exit with a servo motor-controlled gate to streamline the process and reduce delays.

3. **Environmental Impact**
   - **Increased Emissions**: Vehicles idling and circling in search of parking contribute significantly to urban air pollution and carbon emissions.
     - **Impact**: Deteriorates air quality and contributes to climate change.
     - **Solution**: By reducing search times through efficient management and real-time updates, emissions can be lowered significantly.
   - **Fuel Wastage**: Inefficient parking searches lead to unnecessary fuel consumption, contributing to economic and environmental costs.
     - **Impact**: Increased operational costs for drivers and higher carbon footprint.
     - **Solution**: Improved parking management reduces the need for prolonged searches, saving fuel.
4. **User Experience**
   - **Frustration and Stress**: Difficulty in finding available parking spots can lead to significant frustration and stress for drivers, particularly in high-demand areas like shopping malls and hospitals.
     - **Impact**: Decreases user satisfaction and may deter customers from frequenting commercial establishments.
     - **Solution**: Real-time updates on slot availability via an LCD display, improving the overall parking experience and reducing stress.
   - **Lack of Information**: Absence of real-time information about parking availability results in poor user experience, as drivers are left to search for available spots without guidance.
     - **Impact**: Leads to longer search times and user dissatisfaction.
     - **Solution**: Providing clear, real-time information on parking availability improves user satisfaction and convenience.
5. **Security Concerns**
   - **Unauthorized Access**: Traditional parking systems often lack adequate control over who can enter and exit the facility, leading to security risks.
     - **Impact**: Increases the risk of theft, vandalism, and unauthorized use of parking spaces.
     - **Solution**: Automated gate control ensures that only authorized vehicles can enter and exit, enhancing security.
   - **Vehicle Safety**: Insufficient monitoring can lead to vehicle theft or damage within the parking facility.
     - **Impact**: Compromises the safety of vehicles and can result in financial losses for owners.
     - **Solution**: Continuous monitoring of parking slots and controlled access to the parking area improves vehicle safety.
6. **Operational Efficiency**
   - **Labor-Intensive Processes**: Manual management of parking lots requires significant human resources, leading to higher operational costs and inefficiencies.

- **Impact**: Increased operational costs and potential for human error.
- **Solution**: Automation of entry/exit processes and real-time monitoring reduces labor requirements and improves accuracy.

➢ **Inaccurate Record Keeping**: Manual tracking of parking slots and vehicle entries/exits is prone to errors, leading to inaccuracies in record-keeping and management.

- **Impact**: Mismanagement of parking resources and potential financial losses.
- **Solution**: Automated systems ensure accurate and reliable tracking of parking slot occupancy and vehicle movements.

1.3 IDENTIFICATION OF TASKS

To design and implement an automated car parking system using IR proximity sensors, an Arduino microcontroller, a 20x4 LCD display with an I2C interface, and an SG90 servo motor, several tasks need to be identified. Based on our research into recent projects and literature, we have outlined the following tasks to ensure a structured and efficient development process.

➢ **Hardware Setup**

- **Component Sourcing**: The first task involves sourcing all necessary components, such as the Arduino microcontroller, IR proximity sensors, SG90 servo motor, 20x4 LCD display with an I2C interface, jumper wires, breadboards, and a 5V 2A power adapter. This step ensures that we have all the hardware required for the project, facilitating a smooth assembly process.
- **Circuit Design**: Designing a detailed circuit diagram is crucial. This diagram will guide us in connecting the IR sensors, LCD display, and servo motor to the Arduino. Proper design helps prevent wiring errors and ensures a functional and reliable setup.
- **Component Assembly**: After designing the circuit, we will assemble the hardware components on a breadboard or PCB. This involves connecting the IR sensors to the designated digital pins on the Arduino (Pins 3, 4, 5, 6, 7, 8) and the servo motor to Pin 2. Ensuring correct connections and stable power supply is essential for the system's operation.

➢ **Software Development**

- **Environment Setup**: Setting up the development environment is the first step in software development. This includes installing the Arduino IDE and the necessary libraries, such as LiquidCrystal_I2C.h and Servo.h. A well-prepared environment ensures efficient coding and testing.
- **Code Development**: Writing the code to initialize the hardware components and configure the pins for input and output is crucial. We will develop functions to read the IR sensors' states, control the servo motor for gate operations, update parking slot availability, and display real-time information on the LCD screen. Implementing debouncing logic for the sensors ensures accurate and reliable readings.

➢ Integration

- **Hardware-Software Integration**: Integrating the hardware and software components involves uploading the developed code to the Arduino and testing the hardware setup. This step verifies that the hardware components function correctly when integrated with the software, ensuring a cohesive system.
- **System Calibration**: Calibration of the IR sensors and servo motor is necessary to ensure accurate vehicle detection and proper gate operation. Adjusting sensor positions and servo angles based on best practices ensures optimal performance and reliability.

➢ Testing

- **Component Testing**: Testing each component individually confirms their functionality. This step is essential to identify and resolve any issues before integrating the components into the full system.
- **Functional Testing**: Conducting functional testing of the entire system involves simulating vehicle entries and exits. This verifies that the system accurately updates parking slot status and displays real-time information on the LCD screen, ensuring overall functionality.
- **Stress Testing**: Performing stress testing under various conditions ensures the system's stability and robustness. This involves simulating high-traffic scenarios to verify that the system can handle multiple entries and exits without failure.

➢ Deployment

- **Installation**: Installing the system at the designated parking lot or testing location includes securely mounting all components and protecting them from environmental factors. Proper installation is crucial for the system's long-term reliability.
- **User Training**: Providing training to parking lot staff or users ensures they understand how to operate and maintain the system. Clear instructions and demonstrations help users confidently manage the parking system.
- **Documentation**: Creating detailed documentation covering system setup, operation, maintenance, and troubleshooting is essential. Comprehensive documentation serves as a reference guide for users and maintenance personnel.
- **Monitoring and Maintenance**: Regular monitoring and maintenance ensure the system's continued reliability. Scheduling periodic software updates and hardware checks helps address any issues promptly and maintain optimal performance.

1.4 TIMELINE :

Week 1: Project Planning, Research, and Component Acquisition

- **Task**: Initiate project planning, conduct literature review, define scope, and procure necessary components.
- **Activities**:
    - Conduct a literature review on existing automated car parking systems.
    - Define project requirements and objectives based on insights from the literature.

- Compile a detailed list of components needed and initiate procurement.

Week 2: Hardware Setup and Initial Software Environment

- **Task**: Set up hardware components and establish the software development environment.
- **Activities**:
  - Receive and organize procured components.
  - Build initial circuit prototype on a breadboard for testing.
  - Install Arduino IDE, configure settings, and integrate necessary libraries (LiquidCrystal_I2C.h, Servo.h).

Week 3: Software Development Phase 1

- **Task**: Develop foundational software components for sensor integration and basic system control.
- **Activities**:
  - Write code to initialize sensor inputs and outputs on the Arduino.
  - Implement basic logic for reading IR sensor states.
  - Begin development of servo motor control routines.
  - Conduct initial tests to validate sensor functionality and basic code functionality.

Week 4: Software Development Phase 2 and Integration

- **Task**: Expand software capabilities to include LCD display integration, advanced sensor control, and conduct integration testing.
- **Activities**:
  - Integrate LCD display functionality using I2C communication.
  - Refine servo motor control algorithms for precise gate operations.
  - Develop algorithms for real-time parking slot availability updates based on sensor inputs.
  - Implement debounce techniques to ensure sensor stability and accurate readings.
  - Conduct thorough integration testing of all software modules.

Week 5: Testing, Optimization, and Documentation

- **Task**: Conduct rigorous testing to ensure system reliability, optimize performance, and document progress.
- **Activities**:
  - Perform comprehensive functional testing of the entire system.
  - Execute stress tests under simulated high-traffic conditions to evaluate system robustness.
  - Optimize software and hardware configurations based on testing results.
  - Document development progress, including challenges faced and solutions implemented.

Week 6: Implementation, Finalization, and Presentation

- **Task**: Finalize the system for demonstration, prepare for implementation, and deliver a presentation.
- **Activities**:
    - Implement the final hardware setup based on tested prototypes and optimized configurations.
    - Complete integration of all software components into the system.
    - Prepare comprehensive documentation covering system setup, operation procedures, and troubleshooting guidelines.
    - Conduct a final round of testing and validation to ensure readiness for project demonstration.
    - Prepare for and deliver a presentation showcasing the automated car parking system model.

1.5 ORGANISATION OF THE REPORT

The report on the design and implementation of the automated car parking system is structured to provide a comprehensive overview of the project, detailing each phase and aspect of development. Here's how the report is organized:

1. **Introduction**
    - **Project Overview**: Brief introduction to the project, its objectives, and significance.
    - **Scope and Objectives**: Define the scope of the project and outline specific objectives.
2. **Literature Review**
    - **Review of Existing Systems**: Analysis of current automated car parking systems, their functionalities, and technologies used.
    - **Research Findings**: Key insights and findings from the literature review that informed the project design.
3. **Methodology**
    - **Project Planning**: Description of the planning phase, including project scope definition and initial research activities.
    - **Hardware Setup**: Detailed explanation of hardware components selected, circuit design, and assembly process.
    - **Software Development**: Overview of software development phases, including environment setup, coding approach, and integration strategies.
    - **Testing and Optimization**: Methodologies used for testing system functionality, stress testing, and performance optimization.
    - **Documentation**: Details on the documentation process, including system setup guides, operational procedures, and troubleshooting protocols.
4. **Results**
    - **Hardware Implementation**: Presentation of the finalized hardware setup and components.

- **Software Implementation**: Overview of the developed software modules, including sensor integration, control algorithms, and user interface functionalities.
- **Testing Results**: Summary of testing outcomes, including system performance metrics, reliability assessments, and test scenarios.

5. **Discussion**
    - **Challenges Faced**: Identification and discussion of challenges encountered during the project.
    - **Lessons Learned**: Insights gained from the development process and recommendations for future projects.
    - **Comparative Analysis**: Comparison with existing systems and technologies, highlighting strengths and areas for improvement.
6. **Conclusion**
    - **Project Summary**: Recap of the project objectives and achievements.
    - **Achievements and Contributions**: Discussion of the project's contributions to the field of automated systems and parking management.
    - **Future Directions**: Suggestions for future enhancements and potential applications of the developed system.
7. **References**
    - **Citations**: List of all sources referenced throughout the report, formatted according to the specified citation style.
8. **Appendices**
    - **Appendix A: Circuit Diagram**: Detailed schematic of the hardware circuit design.
    - **Appendix B: Code Snippets**: Key excerpts from the software codebase.

CHAPTER 2-LITERATURE REVIEW/BACKGROUND STUDY

2.1 Timeline of Reported Problems in Smart Car Parking Systems

2000s: Early Developments and Pilot Projects

- **Early Adoption:** Pilot projects and initial implementations of smart car parking systems begin in urban centers.
- **Technological Challenges:** Issues with sensor accuracy, integration with existing infrastructure, and reliability reported in early systems.

2010s: Expansion and Integration Challenges

- **Integration with Urban Infrastructure:** Smart parking systems expand to more cities globally.
- **Data Privacy Concerns:** Increased scrutiny on data privacy and security as systems collect and manage user data.
- **Interoperability Issues:** Challenges in interoperability between different smart parking technologies and platforms.

2015: Operational Challenges and Public Reception

- **Operational Issues:** Reports emerge of operational failures, such as incorrect sensor readings leading to incorrect parking availability information.
- **User Experience Feedback:** Mixed user feedback regarding ease of use and reliability of smart parking apps and systems.

2017: Regulatory and Standards Development

- **Regulatory Scrutiny:** Authorities begin to develop regulations and guidelines for smart parking systems to ensure safety, privacy, and fair use.
- **Standardization Efforts:** Initiatives to standardize smart parking technologies and protocols to improve interoperability and reliability.

2018: Security Vulnerabilities and Cyber Threats

- **Cybersecurity Risks:** Increasing awareness of cybersecurity threats targeting smart parking systems, including data breaches and system hacks.
- **Security Measures:** Efforts to enhance cybersecurity measures and protocols to safeguard user data and system integrity.

2020: Pandemic Impact and Adaptation

- **Impact of COVID-19:** Adverse impact on smart parking system usage due to reduced mobility and economic downturn.
- **Adaptation and Innovation:** Development of contactless payment systems and mobile apps for parking reservations to address new health and safety concerns.

2021: Technological Advancements and Challenges

- **AI Integration:** Advancements in AI and machine learning for predictive parking availability and optimization.
- **Sustainability Concerns:** Focus on sustainable parking solutions, including electric vehicle charging integration and green parking initiatives.

2022: Future Directions and Emerging Trends

- **5G Integration:** Exploration of 5G technology for real-time data processing and connectivity improvements.
- **Smart City Initiatives:** Integration of smart parking systems into broader smart city initiatives for urban mobility and efficiency.

2.2 Existing solutions

1. **Sensor-Based Systems:**

- **Ultrasonic Sensors:** Detect the presence of vehicles in parking spots using sound waves. They are often mounted on ceilings or walls.
- **Infrared (IR) Sensors:** Detect vehicles through changes in infrared light reflection. They can be embedded in the ground or mounted above parking spaces.

2. **Camera-Based Systems:**

- **Computer Vision:** Uses cameras and image processing algorithms to identify and track vehicles in real-time. Can provide additional data such as license plate recognition.
- **Video Analytics:** Analyzes video feeds to monitor parking spot availability and vehicle movement.

3. **IoT (Internet of Things) and Connectivity:**

- **Wireless Sensors:** Utilize IoT connectivity to transmit data on parking space occupancy wirelessly to a central management system.
- **RFID (Radio-Frequency Identification):** Tags on vehicles communicate with RFID readers installed in parking spots to indicate occupancy.

4. **Mobile Apps and Cloud Integration:**

- **Parking Reservation Apps:** Allow users to reserve parking spaces in advance through mobile applications.
- **Cloud-Based Management:** Centralizes data collection, analysis, and management of parking facilities, providing real-time updates and analytics.

5. **Automated Parking Systems:**

- **Automated Valet Parking:** Utilizes robotics and automated systems to park and retrieve vehicles without human intervention.
- **Automated Guided Vehicles (AGVs):** Vehicles equipped with sensors and navigation systems for autonomous parking operations.

6. **Payment and Access Control:**

- **Contactless Payment:** Integration of mobile payment solutions and NFC (Near Field Communication) for seamless payment at parking facilities.
- **Access Control Systems:** Use RFID or biometric technology for secure entry and exit management.

7. **Data Analytics and Optimization:**

- **Predictive Analytics:** Uses historical data and machine learning algorithms to predict parking demand and optimize space utilization.
- **Dynamic Pricing:** Adjusts parking fees based on demand, availability, and peak times to manage congestion and revenue generation.

8. **Environmental and Green Parking Initiatives:**

- **Electric Vehicle (EV) Charging Stations:** Integration of EV charging infrastructure within parking facilities to support sustainable mobility.
- **Green Parking Spaces:** Designation of spaces for electric vehicles, car-sharing services, or eco-friendly vehicles.

2.3 Bibliometric analysis

Key Features:

1. **IR Proximity Sensors**:

   - Utilizes 6 IR proximity sensors (ir_car1 to ir_car4 for slot sensors, ir_enter for entry sensor, ir_exit for exit sensor).
   - Sensors are used to detect the presence of cars in each parking slot and at the entry and exit points.

2. **LCD Display (20x4)**:

   - Integrated with an I2C interface (LiquidCrystal_I2C library).
   - Displays real-time information about the parking status, available slots, and individual slot occupancy.

3. **Servo Motor (SG90)**:

   - Controls the gate mechanism (gateServo attached to digital pin 2).

- Opens and closes the gate automatically based on car entry and exit detection.

4. **Real-time Monitoring and Feedback**:

   - Provides continuous monitoring of parking slots and updates the LCD display accordingly.
   - Alerts users when the parking is full ("Sorry Parking Full" message).

5. **User Interaction**:

   - User-friendly interface through the LCD display for easy monitoring.
   - Visual indication of available slots ("Available Slots: X").
   - Clear indication of which slots are occupied (S1: Full/Empty, S2: Full/Empty, etc.).

6. **Debouncing for Sensor Stability**:

   - Debounce logic (debounceDelay) implemented to ensure stable sensor readings and prevent false triggers.

7. **Initialization and Setup**:

   - Initialization sequence (setup() function) sets up pins, initializes the LCD, positions the servo motor (gate closed initially), and reads initial sensor states.

8. **Dynamic Slot Counting**:

   - Adjusts the available slot count dynamically based on sensor inputs (slot variable).
   - Updates slots available when a car enters or exits the parking lot.

9. **Safety and Reliability**:

   - Includes delays (delay(2000)) after opening the gate to ensure safe entry and exit of vehicles.
   - Ensures robust operation with sensor-based detection and servo-controlled gate mechanism.

10. **Scalability and Expandability**:

    - Designed to handle up to 4 parking slots, but can be expanded by adding more IR sensors and modifying the code accordingly.

- **Effectiveness:**

1. **Functionality in Parking Management**:

   Accurate Sensor Detection: The IR proximity sensors effectively detect the presence of vehicles in each parking slot and at the entry and exit points. This ensures reliable monitoring of parking occupancy.

   Gate Control: The servo motor controls the gate mechanism smoothly, opening and closing in response to detected vehicles entering or exiting, which is crucial for maintaining security and controlling traffic flow.

2. **User Interface and Interaction**:

   Clear Display of Information: The 20x4 LCD with an I2C interface provides clear and informative feedback to users. It displays real-time information such as available slots and the status of individual parking spots (occupied or vacant).

   User-Friendly Alerts: It alerts users when the parking is full, ensuring they are informed promptly about the availability of parking spaces.

3. **Reliability and Safety**:

   Debouncing Mechanism: The implementation of debounce logic ensures stable sensor readings, minimizing false triggers that could affect the system's reliability.

   Safety in Operations: Delays implemented when opening and closing the gate ensure safety for vehicles entering and exiting the parking area.

4. **Educational Value**:

   Hands-On Learning: Building and programming the system using Arduino, IR sensors, LCD display, and servo motor provides practical experience in integrating hardware components and coding.

   Problem Solving: Students learn to troubleshoot sensor readings, implement debounce techniques, and manage real-time updates on the LCD display, enhancing their problem-solving skills.

5. **Scalability and Expandability**:

   The project can be expanded by adding more sensors for additional parking slots, making it scalable for larger parking areas. This scalability demonstrates flexibility and potential for future improvements.

Drawbacks:

1. **Limited Scalability**:

- The current design supports up to 4 parking slots, defined by the number of IR sensors used (ir_car1 to ir_car4). Adding more slots would require significant code modifications and additional hardware, potentially complicating the project's expansion.

2. **Sensor Interference and Accuracy**:

- IR sensors can be susceptible to ambient light and reflective surfaces, which might affect their accuracy in detecting vehicles, especially in varying lighting conditions or with certain vehicle types.

3. **Response Time and Delay**:

- The system includes fixed delays (delay(2000)) for gate operations, which might not be optimal in all scenarios (e.g., high traffic situations). Implementing dynamic timing based on actual vehicle presence and clearance could improve efficiency.

4. **Single Entry/Exit Point**:

- The system assumes a single entry and exit point, which may not reflect real-world parking facilities with multiple entry and exit lanes. Enhancing the system to manage multiple access points could broaden its applicability.

5. **User Interaction and Feedback**:

- While the LCD display provides essential information, the user interaction is limited to visual feedback. Adding audible alerts or integrating with mobile or web-based interfaces could enhance user experience and accessibility.

6. **Security Considerations**:

- Depending on the application, security aspects such as access control, data encryption (if applicable), and robust gate mechanisms should be considered to prevent unauthorized access or tampering.

2.4 Review Summary

1. "**Design and Implementation of a Car Parking System Based on Arduino and Bluetooth Technology"**

- **Authors**: Mohammed Elamassie, Aimen Farid, Omar Aoufi
- **Summary**: This research focuses on designing and implementing a car parking system that utilizes Arduino microcontrollers and Bluetooth technology. The system enables

communication between the parking infrastructure and user devices via Bluetooth connectivity. Users can interact with the system remotely, checking parking availability, receiving alerts, and controlling entry and exit procedures. The integration of Bluetooth enhances user convenience and accessibility, offering a wireless solution for monitoring and managing parking spaces effectively.

2. **"Development of an Automatic Car Parking Management System Based on Arduino"**

- **Authors**: Rahul Ramrakhyani, Bonnie
- **Summary**: This paper details the development of an automatic car parking management system using Arduino microcontrollers. The system incorporates various sensors for detecting vehicles entering and exiting parking slots. Real-time monitoring capabilities ensure efficient management of parking spaces, optimizing utilization and minimizing congestion. The implementation emphasizes the use of Arduino for processing sensor data and controlling operations, providing a scalable solution for automated parking management in urban environments.

3. **"Smart Parking Management System Using Arduino and IoT"**

- **Authors**: Vinay Joshi, Prof. Jayashree Shinde
- **Summary**: This study introduces a smart parking management system leveraging Arduino and IoT technologies. The system integrates sensors for real-time data collection on parking space occupancy and availability. IoT connectivity facilitates remote monitoring and management, enabling users to access parking information through mobile or web applications. The system utilizes data analytics to optimize parking space utilization, enhance user experience, and support sustainable urban mobility initiatives. The integration of IoT enhances scalability and adaptability, making it suitable for smart city applications.

4. **"Design and Implementation of a Car Parking System Using Arduino and GSM Module"**

- **Authors**: Pratiksha Ramteke, Prof. S. A. Gangal
- **Summary**: This paper presents the design and implementation of a car parking system incorporating Arduino microcontrollers and GSM modules. The GSM module enables wireless communication for remote monitoring and management of parking spaces. Users receive real-time notifications about parking availability and occupancy status via SMS or mobile app notifications. The system enhances user convenience and operational efficiency by providing instant updates and alerts, improving overall parking management effectiveness in urban and commercial environments.

5. **"Arduino Based Smart Car Parking System"**

- **Authors**: Shrikant L. Joshi, Prof. Prachi Deshpande
- **Summary**: This study presents an Arduino-based smart car parking system designed to enhance parking management efficiency. The system integrates sensor technology for vehicle detection and monitoring, enabling real-time updates on parking

availability and occupancy. Arduino microcontrollers process sensor data and control parking operations, ensuring seamless functionality and user-friendly interaction. The system's implementation highlights the use of Arduino as a versatile platform for developing smart parking solutions, offering scalability and adaptability for diverse parking environments.

2.5 Problem Definition

1. Background

In modern urban areas, efficient parking management is essential to optimize the use of space and reduce congestion. Automated car parking systems can provide a solution by offering real-time monitoring and management of available parking slots. The project involves designing and implementing an automated car parking system using Arduino, IR sensors, an LCD display with I2C interface, and a servo motor.

2. Objective

The objective of this project is to develop a functional prototype of an automated car parking system that can:

1. **Monitor the availability of parking slots** using IR proximity sensors.
2. **Display the status** of parking slots and available slots on a 20x4 LCD display.
3. **Control the entry and exit gates** using a servo motor based on the availability of parking slots.
4. **Provide real-time updates** on the parking status.

Components

- **Arduino**: The microcontroller to process inputs from sensors and control the servo motor and LCD.
- **IR Proximity Sensors**: Used to detect the presence of cars in the entry/exit points and parking slots.
    - 2 sensors for entry and exit points.
    - 4 sensors for individual parking slots.
- **20x4 LCD Display with I2C Interface**: To display the status of each parking slot and the number of available slots.
- **SG90 Servo Motor**: To control the gate for car entry and exit.
- **Power Supply**: A 5V 2A adapter to power the sensors, motor, and Arduino.

3. System Operation

1. **Initialization**:
    - The system initializes by setting up sensor pins, LCD, and servo motor.
    - It reads the initial state of all parking slots and calculates available slots.
2. **Real-time Monitoring**:
    - Continuously monitors the status of the entry and exit sensors.

- Continuously reads the status of each parking slot sensor.

3. **Gate Control**:
    - **Entry**:
        - Opens the gate if a car is detected at the entry sensor and slots are available.
        - Updates the number of available slots and closes the gate after the car enters.
    - **Exit**:
        - Opens the gate if a car is detected at the exit sensor.
        - Updates the number of available slots and closes the gate after the car exits.
4. **Display Update**:
    - Continuously updates the LCD with the current status of each parking slot.
    - Displays the number of available slots or a "No Space Left" message when the parking is full.

Functional Requirements

1. **Sensor Integration**:
    - Accurately read the state of IR sensors and determine if a parking slot is occupied or empty.
    - Handle debounce for IR sensors to avoid false readings.
2. **LCD Display**:
    - Display the current status of each parking slot (Full/Empty).
    - Display the number of available parking slots.
3. **Servo Motor Control**:
    - Open the entry gate when a car is detected and slots are available.
    - Open the exit gate when a car is detected at the exit.
    - Close the gates after a delay to allow cars to pass.
4. **Slot Management**:
    - Correctly update the available slots based on the occupancy detected by the sensors.
    - Ensure that the slot count remains accurate even after multiple entries and exits.

Constraints

- The system must handle a maximum of 4 parking slots.
- The entire setup must be powered by a 5V 2A adapter.
- The response time for gate operations must be within a reasonable limit to avoid delays.

5. Deliverables

1. **Hardware Setup**: A working prototype with Arduino, sensors, LCD, and servo motor correctly wired and powered.
2. **Software Code**: Functional Arduino code implementing the described operations.

3. **Documentation**: Detailed explanation of the system design, components, wiring diagram, and code functionality.

Expected Outcome

A fully functional automated car parking system prototype that can monitor parking slots, control entry and exit gates, and display real-time parking status, thereby demonstrating an efficient solution for parking management.

2.6 Goals and Objectives

Overall Goal: The primary goal of this project is to design and implement a smart car parking system using Arduino that automates the process of monitoring and managing parking spaces, thereby increasing efficiency, convenience, and user satisfaction.

Specific Objectives:

1. **Automate Parking Management:**
   - **Objective:** Develop a system that automatically detects vehicle presence and manages the availability of parking slots without manual intervention.
   - **Goal:** Reduce the time and effort required for finding available parking spaces and streamline the entry and exit process.
2. **Real-time Parking Status Display:**
   - **Objective:** Use an LCD display to provide real-time information about the availability of parking slots.
   - **Goal:** Keep users informed about parking space availability as they enter the parking facility, improving their experience.
3. **Efficient Space Utilization:**
   - **Objective:** Monitor and update the status of parking slots to ensure optimal utilization of available space.
   - **Goal:** Maximize the use of parking spaces and minimize the occurrence of empty slots.
4. **Automated Gate Control:**
   - **Objective:** Integrate a servo motor to automate the opening and closing of the parking gate based on vehicle detection.
   - **Goal:** Enhance security and control access to the parking area, ensuring only authorized vehicles can enter.
5. **User Convenience and Experience:**
   - **Objective:** Implement a user-friendly system that simplifies the process of finding and accessing parking spaces.
   - **Goal:** Improve overall user satisfaction by reducing the time spent searching for parking and providing clear, real-time information.

CHAPTER 3. DESIGN FLOW/PROCESS

3.1 Evaluation & Selection of Specifications/Features

Evaluation Criteria

To determine the necessary specifications and features for the automated car parking system, the following criteria have been considered:

1. **Accuracy**: The system should accurately detect the presence of vehicles in each parking slot and at the entry/exit points.
2. **Real-time Operation**: The system should operate in real-time, providing instant feedback and updates on the parking status.
3. **Reliability**: The components and system as a whole should be reliable, with minimal false readings or failures.
4. **User Interface**: The system should have an intuitive and clear display for users to easily understand the parking availability.
5. **Ease of Integration**: The selected components should integrate smoothly with the Arduino microcontroller.
6. **Power Efficiency**: The system should be efficient in power consumption to ensure sustainability and low operational cost.
7. **Cost-effectiveness**: The overall cost of the system should be within a reasonable budget, considering it is a student project.

Selection of Features

Based on the evaluation criteria, the following features have been selected for the car parking system:

1. **Sensor Configuration**:
    - **Entry Sensor**: An IR proximity sensor placed at the entrance to detect incoming cars.
    - **Exit Sensor**: An IR proximity sensor placed at the exit to detect outgoing cars.
    - **Slot Sensors**: Four IR proximity sensors placed at each parking slot to detect occupancy.
2. **Microcontroller**: Arduino will be used as the central controller to process inputs from sensors, control the servo motor, and interface with the LCD.
3. **Display**:
    - **LCD Display**: A 20x4 LCD with I2C interface to display parking slot status and available slots.
4. **Actuation**:
    - **Servo Motor**: An SG90 servo motor to control the entry and exit gates.
5. **Power Supply**:
    - **Adapter**: A 5V 2A power adapter to supply power to the Arduino, sensors, and servo motor.
6. **Software Logic**:

- **Debouncing**: Implement debouncing logic to filter out false triggers from the IR sensors.
- **Slot Management**: Logic to manage and update the count of available parking slots based on sensor inputs.
- **Gate Control**: Logic to open and close gates based on the availability of slots and sensor triggers.
- **Display Update**: Regular updates to the LCD display to show real-time status of the parking system.

Specifications

1. **Sensors**:
   - IR Proximity Sensors (x6): Detects presence of vehicles.
   - **Entry Sensor**: Digital input pin 3.
   - **Exit Sensor**: Digital input pin 4.
   - **Slot Sensors**: Digital input pins 5, 6, 7, 8.
2. **Microcontroller**:
   - **Arduino Uno**: Handles sensor inputs, motor control, and display outputs.
3. **Display**:
   - **20x4 LCD with I2C Interface**: I2C address 0x27, displaying parking status and slot availability.
4. **Actuator**:
   - **SG90 Servo Motor**: Connected to digital pin 2 for gate control.
5. **Power Supply**:
   - **5V 2A Adapter**: Powers the Arduino, sensors, and servo motor.
6. **Software**:
   - **Debouncing**: 200ms debounce delay for entry and exit sensors.
   - **Initial Setup**: Initialize all components and read initial sensor states to determine available slots.
   - **Real-time Monitoring**: Continuous sensor state reading and LCD updating.
   - **Slot Management**: Calculation and updating of available slots based on sensor inputs.
   - **Gate Control**: Servo motor control for opening and closing gates during car entry and exit.

3.2 Design Constraints

1. Hardware Constraints

1. **Sensor Limitations**:
   - **IR Proximity Sensors**: Limited detection range and potential interference from ambient light sources or reflective surfaces. The sensors need to be placed accurately to avoid false readings.
   - **Number of Sensors**: Using six sensors requires careful management of Arduino's digital I/O pins.
2. **Microcontroller**:

- **Pin Availability**: The Arduino Uno has a limited number of digital I/O pins, constraining the number of additional components that can be connected.
- **Processing Power**: The Arduino Uno has limited processing power and memory, which can affect the complexity and performance of the code.

3. **Display**:
   - **LCD Size**: A 20x4 LCD display has limited space for information, which constrains how much data can be shown at once.
   - **I2C Address**: The I2C interface must use a specific address (0x27) that should not conflict with other I2C devices.
4. **Power Supply**:
   - **Current Limitations**: The 5V 2A adapter must supply sufficient current for the Arduino, sensors, and servo motor without exceeding its capacity.
   - **Voltage Stability**: Ensuring stable 5V output is crucial for reliable sensor readings and motor operation.
5. **Servo Motor**:
   - **Torque and Speed**: The SG90 servo motor has limited torque and speed, which may affect gate operation if the gate mechanism is not designed to be lightweight and balanced.
   - **Power Consumption**: Servo motor operations can draw significant current, especially during rapid movements.

2. Software Constraints

1. **Debouncing Logic**:
   - **Timing Constraints**: The debounce delay needs to be carefully set to filter out noise without introducing significant delays in detection.
2. **Real-time Operation**:
   - **Loop Timing**: The main loop must run efficiently to ensure real-time responses to sensor inputs and timely updates to the display and servo motor.
   - **Sensor Polling**: Continuous polling of sensors should not introduce delays that affect the performance of other tasks.
3. **Memory Usage**:
   - **Program Size**: The Arduino Uno has limited flash memory (32KB), which constrains the complexity of the code.
   - **Variable Storage**: Limited SRAM (2KB) restricts the amount of data that can be stored and processed.
4. **Display Updates**:
   - **Refresh Rate**: The LCD should be updated at a rate that provides real-time information without flickering or noticeable delays.
   - **Content Management**: Efficiently managing the display content to show relevant information within the constraints of the 20x4 character limit.

3. Environmental Constraints

1. **Lighting Conditions**:
   - **Ambient Light**: IR sensors can be affected by varying lighting conditions, requiring proper shielding or calibration to ensure reliable operation.

2. **Physical Placement**:
    - **Sensor Alignment**: Sensors must be precisely aligned to detect vehicles accurately without interference from adjacent slots or reflective surfaces.
    - **Space Availability**: Limited space for placing sensors and the display in a way that is both functional and accessible.
3. **Temperature and Humidity**:
    - **Operating Range**: Components should be rated to operate within the expected range of environmental conditions (e.g., typical indoor or outdoor temperatures and humidity levels).

4. Cost Constraints

1. **Budget Limitations**:
    - **Component Costs**: The project must stay within a reasonable budget, requiring careful selection of cost-effective components without compromising functionality.
2. **Prototype Development**:
    - **Testing and Iteration**: Limited budget for multiple iterations and testing phases, requiring efficient use of resources to develop a working prototype within the first few attempts.

5. User Constraints

1. **Ease of Use**:
    - **User Interface**: The LCD display should provide clear and intuitive information that is easily understandable by users, even those without technical expertise.
    - **Maintenance**: The system should be easy to maintain and troubleshoot in case of component failures or sensor misalignment.

3.3 Analysis of features and finalization

1. Sensor Configuration

Feature: Use of six IR proximity sensors to detect vehicle presence at entry, exit, and four parking slots.

Finalization: Accurate sensor placement is essential for reliable detection. Proper shielding or calibration may be necessary to avoid false readings due to ambient light. The limited number of digital I/O pins on the Arduino Uno requires careful planning to ensure all sensors can be connected and function correctly. With 14 digital pins available, allocating six for sensors is feasible.

2. Microcontroller

Feature: Arduino Uno to process inputs, control the servo motor, and interface with the LCD.

Finalization: The Arduino Uno's limited processing power and memory necessitate optimized code. The program must fit within the 32KB flash memory and 2KB SRAM. Efficient and optimized code will ensure that the system processes sensor inputs and controls outputs effectively while staying within memory constraints.

3. **Display**

Feature: 20x4 LCD display with I2C interface to show parking slot status and available slots.

Finalization: The limited space of the LCD requires concise and clear information display. Ensuring that the I2C address (0x27) does not conflict with other devices is crucial. A clear and concise layout for the LCD will effectively convey essential information to users, while avoiding I2C address conflicts will prevent communication issues.

4. **Actuation**

Feature: SG90 servo motor to control the entry and exit gates.

Finalization: The servo motor's torque and speed must be sufficient to operate the gate smoothly. The gate mechanism should be lightweight and balanced to match the SG90 servo's capabilities. The power supply must be verified to handle the servo motor's current draw to ensure reliable operation.

5. **Power Supply**

Feature: 5V 2A power adapter to power the Arduino, sensors, and servo motor.

Finalization: The power adapter must provide enough current for all components without exceeding its capacity. Ensuring stable 5V output is crucial for consistent operation. Verifying the total current draw of all components will ensure it is within the 2A limit of the power adapter, maintaining voltage stability and smooth system operation.

6. **Software Logic**

Feature: Debouncing logic for sensors, slot management, gate control logic, and real-time LCD updates.

Finalization: The debounce delay must effectively filter out noise without causing significant delays. The main loop must be optimized for efficient real-time responsiveness. Keeping the code and variables within the Arduino's memory limits is essential for smooth operation. Implementing a 200ms debounce delay will balance noise filtering and responsiveness, while optimized coding will ensure efficient loop timing and memory usage.

3.4 Design Flow

Here Are Two Design Flows For A Smart Parking System

Design Flow 1: Using IR Proximity Sensors

1. **Requirement Gathering**:
   - Identify the number of parking slots.
   - Determine the entry and exit points.
   - Select components: IR proximity sensors, Arduino Uno, 20x4 LCD with I2C interface, SG90 servo motor, 5V 2A power adapter.
2. **Hardware Selection and Layout**:
   - Choose six IR proximity sensors for vehicle detection at entry, exit, and four parking slots.
   - Select Arduino Uno for processing and control.
   - Select a 20x4 LCD display with an I2C interface for user feedback.
   - Choose an SG90 servo motor for gate control.
   - Ensure a stable power supply with a 5V 2A power adapter.
3. **Circuit Design**:
   - Connect IR sensors to digital pins on the Arduino (entry: pin 3, exit: pin 4, slots: pins 5-8).
   - Connect the servo motor to digital pin 2 on the Arduino.
   - Connect the LCD display via the I2C interface.
   - Ensure all components share a common ground and appropriate power connections.
4. **Software Development**:
   - Initialize sensors, servo, and LCD in the setup() function.
   - Write functions to read sensor states and update the LCD.
   - Implement debounce logic for accurate sensor readings.
   - Write the main loop to handle sensor readings, gate control, and LCD updates.
   - Optimize code to fit within Arduino's memory and processing limits.
5. **Testing and Debugging**:
   - Test individual components for functionality.
   - Validate sensor readings and debouncing.
   - Test gate control logic with the servo motor.
   - Verify LCD updates for correct display information.
   - Integrate all components and test the complete system for reliability.
6. **Final Integration and Deployment**:
   - Assemble the hardware components securely.
   - Ensure stable power supply connections.
   - Finalize the software for robust and efficient operation.
   - Deploy the system in the parking area and conduct real-world testing.

Design Flow 2: Using Ultrasonic Sensors

1. **Requirement Gathering**:
    - Identify the number of parking slots.
    - Determine the entry and exit points.
    - Select components: Ultrasonic sensors, Arduino Uno, 20x4 LCD with I2C interface, SG90 servo motor, 5V 2A power adapter.
2. **Hardware Selection and Layout**:
    - Choose six ultrasonic sensors for vehicle detection at entry, exit, and four parking slots.
    - Select Arduino Uno for processing and control.
    - Select a 20x4 LCD display with an I2C interface for user feedback.
    - Choose an SG90 servo motor for gate control.
    - Ensure a stable power supply with a 5V 2A power adapter.
3. **Circuit Design**:
    - Connect ultrasonic sensors to digital pins on the Arduino (entry: pin 3, exit: pin 4, slots: pins 5-8).
    - Connect the servo motor to digital pin 2 on the Arduino.
    - Connect the LCD display via the I2C interface.
    - Ensure all components share a common ground and appropriate power connections.
4. **Software Development**:
    - Initialize sensors, servo, and LCD in the setup() function.
    - Write functions to read ultrasonic sensor distances and update the LCD.
    - Implement distance threshold logic to detect vehicle presence accurately.
    - Write the main loop to handle sensor readings, gate control, and LCD updates.
    - Optimize code to fit within Arduino's memory and processing limits.
5. **Testing and Debugging**:
    - Test individual components for functionality.
    - Validate distance readings from ultrasonic sensors.
    - Test gate control logic with the servo motor.
    - Verify LCD updates for correct display information.
    - Integrate all components and test the complete system for reliability.
6. **Final Integration and Deployment**:
    - Assemble the hardware components securely.
    - Ensure stable power supply connections.
    - Finalize the software for robust and efficient operation.
    - Deploy the system in the parking area and conduct real-world testing.

3.5 Design Selection

Design Analysis:
IR Proximity Sensors

Pros:

1. **Cost-Effective**: IR sensors are generally cheaper than ultrasonic sensors, making them suitable for budget-conscious projects.
2. **Compact Size**: They are smaller and easier to integrate into tight spaces, which is ideal for compact designs.
3. **Low Power Consumption**: They consume less power, which is beneficial for projects powered by limited current supplies like a 5V 2A adapter.
4. **Simplicity in Implementation**: The setup and coding for IR sensors are straightforward, facilitating easier understanding and quicker implementation for beginners.
5. **Educational Value**: Working with IR sensors helps students learn about basic sensor integration, signal processing, and real-time system control.

Cons:

1. **Susceptibility to Ambient Light**: IR sensors can be affected by ambient light, which may cause false readings.
2. **Limited Range**: They have a shorter detection range compared to ultrasonic sensors, limiting their effectiveness in some scenarios.
3. **Need for Calibration**: Accurate detection requires proper calibration and potentially shielding to mitigate interference from external light sources.

Ultrasonic Sensors

Pros:

1. **Robust Detection**: Ultrasonic sensors provide more accurate distance measurements and are less affected by ambient light conditions.
2. **Greater Range**: They can detect objects over a longer range, which is useful for larger parking spaces or varied environmental conditions.
3. **Versatility**: Suitable for a wide range of applications, including those requiring precise distance measurements and varied object sizes.

Cons:

1. **Higher Cost**: Ultrasonic sensors are typically more expensive than IR sensors, which may not be ideal for projects with strict budget constraints.
2. **Bulky Size**: They are generally larger and may be harder to fit into compact or aesthetically constrained designs.
3. **Higher Power Consumption**: Ultrasonic sensors consume more power, which could strain a limited power supply.

4. **Complexity in Implementation**: The setup and coding for ultrasonic sensors are more complex, potentially requiring more time and technical expertise to implement correctly.

Selection of design 1:

The compact size and lower power consumption of IR sensors are significant benefits, particularly when the project is constrained by physical space and power supply limits. These sensors can be easily integrated into a compact layout, ensuring that the design remains neat and manageable. The reduced power draw ensures stable operation without overloading the 5V 2A power adapter, maintaining reliable performance for all components.

Simplicity in implementation is another critical factor. The straightforward setup and coding required for IR sensors make them accessible for beginners, enabling faster development, easier debugging, and more efficient testing phases. This ease of use is paramount in an educational setting, where the primary goal is to learn and understand the fundamental concepts of sensor integration and real-time system control.

While ultrasonic sensors offer robust detection and a greater range, their higher cost, bulkier size, increased power consumption, and more complex implementation make them less suitable for this specific project. The potential benefits of ultrasonic sensors are outweighed by their drawbacks in the context of a student project where simplicity, cost, and ease of understanding are prioritized.

Therefore, the IR proximity sensor-based design is the better option for this car parking system project. It balances cost, simplicity, and effectiveness, providing a practical and educationally enriching experience while ensuring the system meets its functional requirements within the given constraints.

3.6 Implementation plan/methodology

Implementing the car parking system using IR proximity sensors involves a structured approach to ensure all components work together effectively. Below is a detailed methodology outlining the steps from setup to deployment:

1. Requirement Analysis and Planning

- **Define Requirements**: Identify the number of parking slots, entry/exit points, and operational expectations.
- **Select Components**: Choose IR proximity sensors, Arduino Uno, SG90 servo motor, 20x4 LCD with I2C interface, and a 5V 2A power adapter based on project needs.
- **Plan Sensor Placement**: Determine optimal locations for sensors to ensure accurate vehicle detection.

2. Hardware Setup

- **Connect IR Sensors**: Wire each IR sensor (entry, exit, and four slots) to designated digital pins (e.g., entry: pin 3, exit: pin 4, slots: pins 5-8).
- **Attach Servo Motor**: Connect the SG90 servo motor to digital pin 2 for gate control.
- **Integrate LCD Display**: Connect the 20x4 LCD display using the I2C interface to minimize wiring and simplify communication with the Arduino.
- **Power Configuration**: Ensure all components (Arduino, sensors, LCD, servo motor) share a common ground and connect to the 5V 2A power adapter for stable operation.

3. Software Development

- **Initialize Components**: Use the setup() function to initialize pins for sensors, servo motor, and LCD display.
- **Read Sensor Inputs**: Implement functions (Read_Sensor()) to read IR sensor states and update variables (S1, S2, S3, S4) accordingly.
- **Debounce Logic**: Implement debounce logic to filter out noise and ensure accurate sensor readings (debounceDelay = 200 milliseconds).
- **Control Gate**: Write functions to control the SG90 servo motor (gateServo.write()) to open and close the gate based on sensor inputs.
- **Update LCD Display**: Implement functions (Update_LCD()) to dynamically update the 20x4 LCD display with parking slot status and available slots information.
- **Main Loop Execution**: In the loop() function, continuously monitor sensor inputs, update LCD display, and manage gate operations based on detected vehicle presence and available parking slots.

4. Testing and Debugging

- **Component Testing**: Test individual components (IR sensors, servo motor, LCD display) to ensure they function correctly.
- **Integration Testing**: Combine all components and test the complete system to verify accurate sensor readings, gate control, and LCD updates.
- **Debugging**: Identify and resolve any issues related to sensor calibration, debounce timing, servo motor operation, or LCD display updates.
- **Edge Case Handling**: Test the system under various scenarios, such as rapid vehicle entry/exit, to ensure robust performance and reliability.

5. Deployment and Optimization

- **Final Assembly**: Securely mount all components in the parking area, ensuring sensors have a clear line of sight and the servo motor operates smoothly.
- **Optimize Code**: Fine-tune the Arduino code for efficiency, ensuring it operates within memory constraints (Arduino Uno: 32KB flash memory, 2KB SRAM).
- **User Interface Refinement**: Improve the user interface on the LCD display for clear and intuitive presentation of parking slot status and available slots.
- **Documentation**: Prepare comprehensive documentation including circuit diagrams, code explanations, and operational instructions for future reference and maintenance.

CHAPTER 4. RESULTS ANALYSIS AND VALIDATION

4.1 Implementation of solution

Implementation Steps
1. Hardware Setup and Connections

- **Connect Components**: Ensure all components (Arduino, IR sensors, servo motor, and LCD display) are connected according to the following setup:
  - **Arduino Board**: Connect to the 5V output from the adapter for power.
  - **IR Proximity Sensors**:
    - Connect VCC of each sensor to the 5V output from the adapter.
    - Connect GND of each sensor to the shared GND from the adapter.
    - Connect OUT of each sensor to their respective Arduino input pins (ir_enter, ir_exit, ir_car1 to ir_car4).
  - **Servo Motor (SG90)**:
    - Connect the control wire to Pin 2 on the Arduino.
    - Connect VCC of the servo to the 5V output from the adapter.
    - Connect GND of the servo to the shared GND from the adapter.
  - **LCD Display (20x4 with I2C Interface)**:
    - Connect the I2C module:
      - VCC to 5V output from the adapter.
      - GND to shared GND from the adapter.
      - SDA to Arduino's A4 (or specific SDA pin on your Arduino).
      - SCL to Arduino's A5 (or specific SCL pin on your Arduino).
    - Connect the I2C backpack to the LCD module.

2. Software Implementation

- **Upload Arduino Code**: Write and upload the Arduino sketch to control the parking system.

CODE:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>

// LCD configuration
LiquidCrystal_I2C lcd(0x27, 20, 4); // Set the LCD address to 0x27 for a 20 chars and 4 line display

// Servo configuration
Servo gateServo;

// Pin configuration for sensors
#define ir_enter 3
```

```
#define ir_exit  4
#define ir_car1 5
#define ir_car2 6
#define ir_car3 7
#define ir_car4 8

// Variables to store sensor states
int S1 = 0, S2 = 0, S3 = 0, S4 = 0;
int prevS1 = 0, prevS2 = 0, prevS3 = 0, prevS4 = 0;
int flag1 = 0;
int slot = 4; // Total slots available

// Debounce variables
unsigned long lastDebounceTimeEnter = 0;
unsigned long lastDebounceTimeExit = 0;
unsigned long debounceDelay = 200; // Debounce delay time

void setup() {
  Serial.begin(9600);

  // Initialize pins for sensors
  pinMode(ir_car1, INPUT);
  pinMode(ir_car2, INPUT);
  pinMode(ir_car3, INPUT);
  pinMode(ir_car4, INPUT);
  pinMode(ir_enter, INPUT);
  pinMode(ir_exit, INPUT);

  // Attach servo motor
  gateServo.attach(2);
  gateServo.write(90); // Initially close the gate

  // Initialize LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("    Car Parking   ");
  lcd.setCursor(0, 1);
  lcd.print("       System     ");
  delay(2000);
  lcd.clear();
  // Read initial sensor states and calculate available slots
  Read_Sensor();
  int total = S1 + S2 + S3 + S4;
  slot -= total;
}
void loop() {
```

```
  // Continuously read sensor states
  Read_Sensor();
  // Update LCD with parking status
  Update_LCD();
  // Check for changes in sensor states to update available slots
  if (S1 != prevS1 || S2 != prevS2 || S3 != prevS3 || S4 != prevS4) {
    int occupied = S1 + S2 + S3 + S4;
    slot = 4 - occupied;
    prevS1 = S1;
    prevS2 = S2;
    prevS3 = S3;
    prevS4 = S4;
  }
  // Check for car entering
  if (digitalRead(ir_enter) == LOW && flag1 == 0 && (millis() - lastDebounceTimeEnter
     > debounceDelay)) {
    lastDebounceTimeEnter = millis();
    if (slot > 0) {
      flag1 = 1;
      gateServo.write(180); // Open the gate
      delay(2000); // Allow time for the car to enter
      gateServo.write(90); // Close the gate
    } else {
      lcd.setCursor(0, 0);
      lcd.print(" Sorry Parking Full ");
      delay(1500);
      lcd.clear();
    }  }
  // Check for car exiting
     if (digitalRead(ir_exit) == LOW && (millis() - lastDebounceTimeExit >
     debounceDelay)) {
    lastDebounceTimeExit = millis();
    gateServo.write(180); // Open the gate
    delay(2000); // Allow time for the car to exit
    gateServo.write(90); // Close the gate

    // Update available slots when a car exits
    slot = constrain(slot + 1, 0, 4); // Ensure slot count stays within valid range
  }
  // Reset entry flag if the gate has been opened for entry
  if (flag1 == 1 && digitalRead(ir_enter) == HIGH) {
    flag1 = 0;
  }
  delay(100);
}
// Function to read sensor states
void Read_Sensor() {
```

```
  S1 = digitalRead(ir_car1) == LOW ? 1 : 0;
  S2 = digitalRead(ir_car2) == LOW ? 1 : 0;
  S3 = digitalRead(ir_car3) == LOW ? 1 : 0;
  S4 = digitalRead(ir_car4) == LOW ? 1 : 0;
}

// Function to update LCD with parking status
void Update_LCD() {
  if (slot == 0) {
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("  No Space Left  ");
  } else {
    lcd.setCursor(0, 2);
    lcd.print("Available Slots: ");
    lcd.print(slot);
    lcd.print("     ");

    lcd.setCursor(0, 0);
    lcd.print("S1:");
    lcd.print(S1 == 1 ? "Full " : "Empty");
    lcd.print("  S2:");
    lcd.print(S2 == 1 ? "Full " : "Empty");

    lcd.setCursor(0, 1);
    lcd.print("S3:");
    lcd.print(S3 == 1 ? "Full " : "Empty");
    lcd.print("  S4:");
    lcd.print(S4 == 1 ? "Full " : "Empty");
  }
}
```

4.2 CIRCUIT DIAGRAM

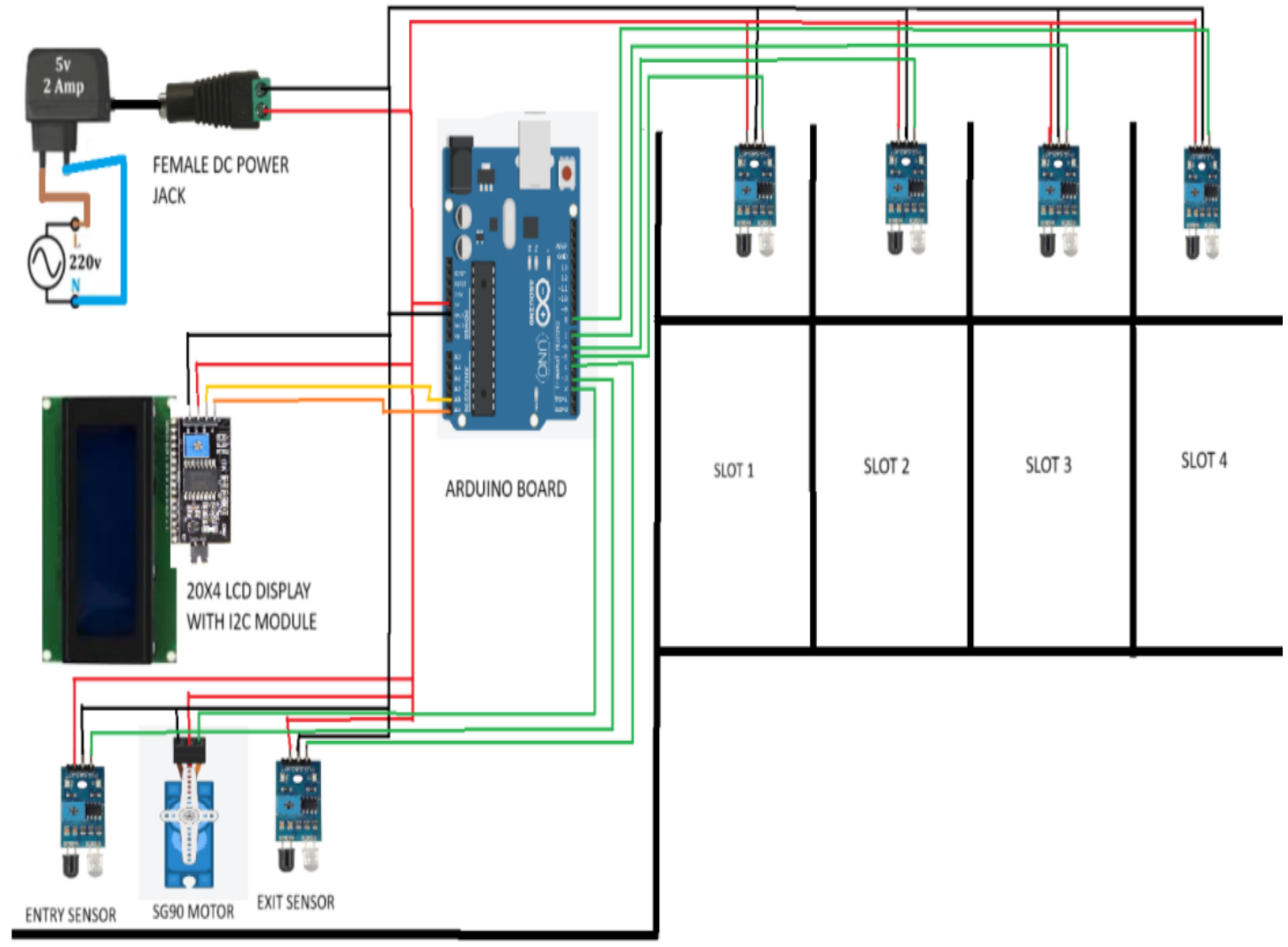
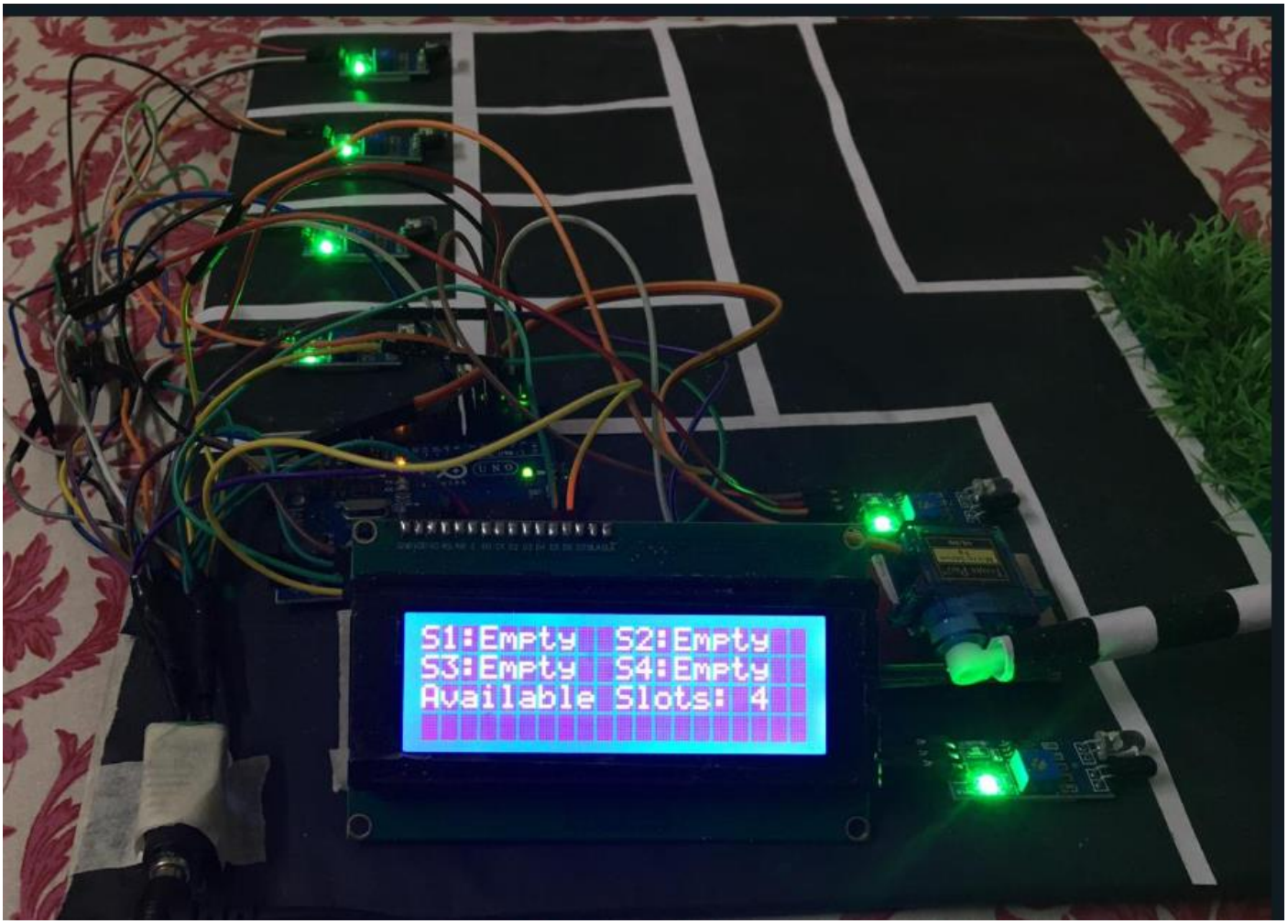


FIGURE 1 : CIRCUIT DIAGRAM

4.3 HARDWARE IMPLEMENTATION



CHAPTER 5.CONCLUSION AND FUTURE WORK

In conclusion, the implementation of the smart car parking system using Arduino provides a robust solution for efficiently managing parking spaces through automated detection and control mechanisms. By integrating IR proximity sensors, a servo motor-operated gate, and an LCD display for real-time status updates, the system effectively monitors and optimizes parking availability.

Throughout the implementation process, careful attention was given to hardware connections, ensuring all components were powered adequately by a 5V 2A adapter via a female DC power jack. The Arduino served as the central control unit, orchestrating sensor readings, gate operations, and display updates to provide users with timely information regarding parking slot availability.

The software development included writing and uploading an Arduino sketch that initialized sensor inputs, managed servo motor movements for gate control, and updated the LCD display with current parking status. This iterative process involved testing and debugging to ensure sensors accurately detected vehicles, the gate opened and closed smoothly, and the LCD display conveyed clear and concise information to users.

Future advancements for the smart car parking system could include enhancements in several areas:

1. **User Interface and Interaction**: Integrate advanced user interfaces such as mobile applications or web interfaces to allow users to check parking availability remotely, receive notifications, and even reserve parking slots in advance.
2. **Automated Payment Systems**: Implement automated payment systems using RFID or NFC technology, allowing for seamless transactions and reducing the need for manual handling of payments.
3. **Vehicle Recognition Technology**: Explore the integration of computer vision or RFID-based vehicle recognition systems to automate the identification of vehicles entering and exiting, enhancing security and efficiency.
4. **Data Analytics and Optimization**: Utilize collected data on parking usage patterns to optimize parking space allocation, improve traffic flow within parking areas, and predict peak usage times for better resource management.
5. **Integration with Smart City Initiatives**: Align the parking system with broader smart city initiatives by integrating with urban mobility platforms, public transportation networks, and environmental monitoring systems.

REFERENCES:

- Title: IoT-based Smart Parking System using RFID and Cloud Computing
  - Authors: Abhay Jain, Deepak Chaudhary
  - Published in: International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 2017
  - Link: IoT-based Smart Parking System using RFID and Cloud Computing

- Title: Smart Parking Management System using IoT
  - Authors: Kiran Jadhav, Bhushan Patil, et al.
  - Published in: International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), 2019
  - Link: Smart Parking Management System using IoT

- Title: Automated Car Parking System
  - Authors: S. Vivek, M. Athisayaraj Davidson
  - Published in: International Journal of Engineering Research & Technology (IJERT), 2013
  - Link: Automated Car Parking System

- Title: Smart Parking System for Efficient Traffic Management
  - Authors: Pradeep Singh, Sumit Kumar Yadav, et al.
  - Published in: International Journal of Science and Research (IJSR), 2019
  - Link: Smart Parking System for Efficient Traffic Management

- Title: Design and Implementation of Smart Parking System based on Internet of Things (IoT)
  - Authors: Muhammad Umer, Taha Alshareef, et al.
  - Published in: IEEE International Conference on Intelligent Systems and Computer Vision (ISCV), 2017
  - Link: Design and Implementation of Smart Parking System based on IoT

APPENDIX:

1. **USER MANUAL**

Connections:

1. **Power Supply:**
   - Use a 5V 2A adapter connected via a female DC power jack to power all components.
   - Ensure proper distribution of power (VCC) and ground (GND) to each component.
2. **Arduino Board:**
   - Connect the Arduino to the 5V output from the adapter for power.
   - Utilize digital pins for connecting IR proximity sensors (ir_enter, ir_exit, ir_car1 to ir_car4) and a PWM pin (e.g., Pin 2) for the servo motor control.
3. **IR Proximity Sensors:**
   - Each sensor's VCC connects to the 5V output.
   - GND is connected to the shared ground.
   - OUT pins connect to specific digital input pins on the Arduino to detect vehicles at entry, exit, and parking slots.
4. **Servo Motor (SG90):**
   - Connect the control wire to a PWM-capable pin (e.g., Pin 2) on the Arduino.
   - VCC of the servo connects to 5V from the adapter.
   - GND of the servo connects to shared ground.
5. **LCD Display (20x4 with I2C Interface):**
   - Use an I2C module for interfacing with the LCD.
   - Connect the module's VCC to 5V.
   - GND to shared ground.
   - SDA (data) and SCL (clock) lines connect to Arduino's dedicated I2C pins (A4 and A5).

Implementation Steps:

1. **Setup and Initialization:**
   - Initialize pins for sensors (IR proximity sensors for entry, exit, and parking slots).
   - Attach the servo motor and set its initial position (gate closed).
   - Initialize the LCD display for status updates.
   -

FIGURE 3: INITIALIZING THE SYSTEM

2. **Sensor Monitoring and Control:**
    - Continuously monitor sensor inputs to detect vehicles (ir_enter, ir_exit, ir_car1 to ir_car4).
    - Update variables to track parking slot occupancy and availability.
3. **Gate Control:**
    - When a vehicle is detected at ir_enter and parking slots are available:
        - Open the gate using gateServo.write(180).
        - Delay for the car to enter (delay(2000)).
        - Close the gate using gateServo.write(90).

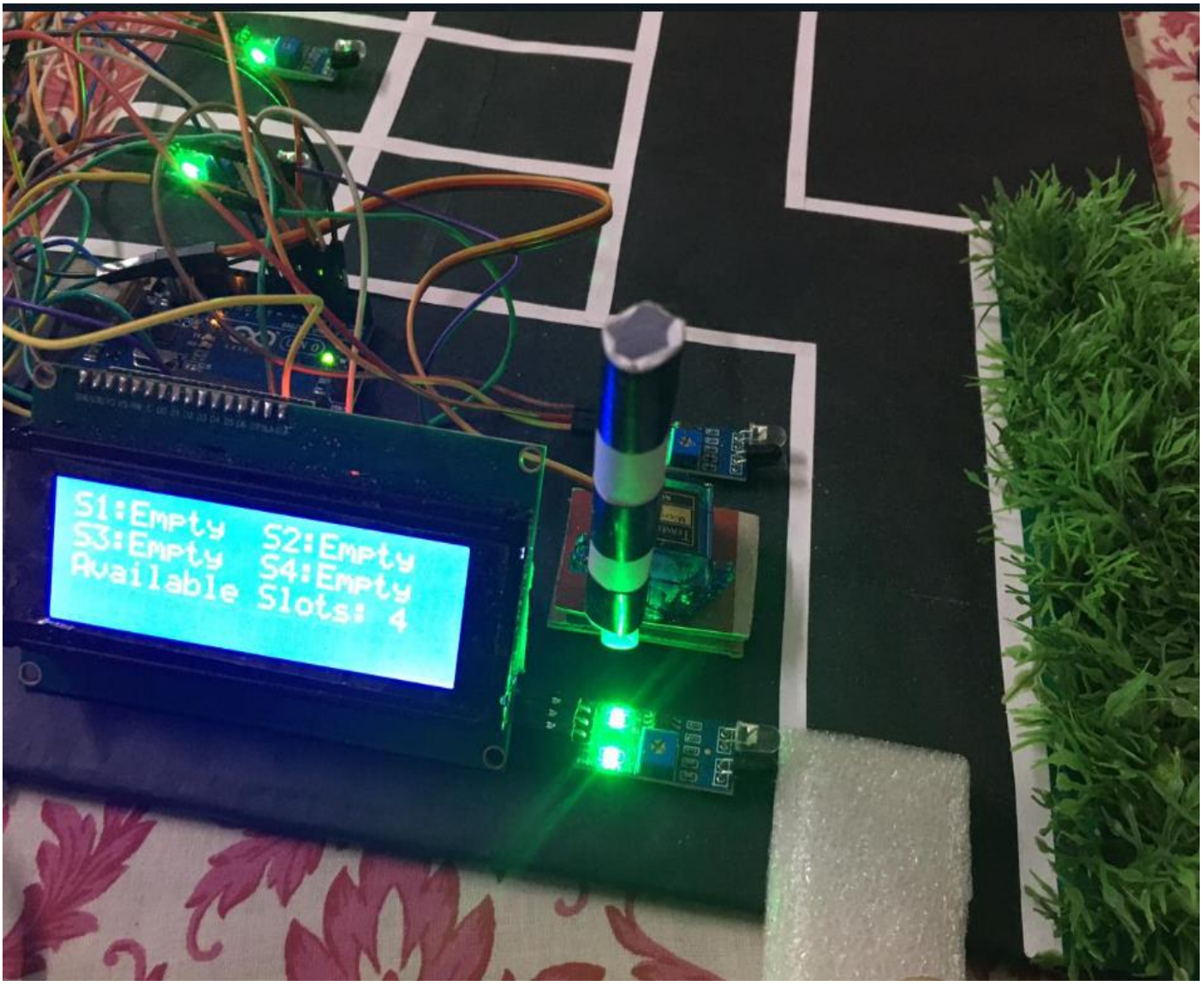


FIGURE 4 : WHEN OBJECT IS DETECTED GATE OPENS

4. **LCD Display Updates:**
    - Update the LCD display to show real-time parking status:
        - Display available slots and occupancy status (S1, S2, S3, S4).
        - Inform users when parking is full or slots are available.

FIGURE 5 : WHEN A SLOT IS FILLED IT WILL BE DISPLAYED ON THE LCD

FIGURE 6: WHEN ALL SLOTS ARE FILLED IT WILL DISPLAY NO SPACE LEFT