

PROFESSIONAL TRAINING REPORT - II

entitled

POWER EFFICIENT AUTOMATIC PARKING SYSTEM

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering with
specialization in Cyber Security

by

DIXIT MANNA KM

41614019



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY -1 UNIVERSITY BY UGC

Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL 2024



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **Mr.DIXIT MANNA KM** who carried out the project entitled "**POWER EFFICIENT AUTOMATIC PARKING SYSTEM**" under my supervision from January 2024 to April 2024.

Internal Guide

Dr.B.U.ANU BARATHI,M.E.,Ph.D.,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **DIXIT MANNA KM (41614019)**, hereby declare that the Professional Training Report-II entitled “**POWER EFFICIENT AUTOMATIC PARKING SYSTEM**” done by me under the guidance of **Dr.B.U.ANU BARATHI,M.E.,Ph.D.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with specialization in Cyber Security.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. S.Vigneshwari M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Dr.B.U.Anu Barathi,M.E.,Ph.D.**, for her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase-2 Professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

COURSE CERTIFICATE



CERTIFICATE OF COMPLETION



This certificate is presented to Mr./Ms. Dixit Manna

who has successfully completed the value added course on Advanced IoT and Technologies for 45 Hours and completed a mini project under IoT which was organized by the department of computer science in association with Techknots Academy LLP at Sathyabama Institute of Science and Technology

A handwritten signature in black ink, appearing to read 'N. E.', positioned above the name and title.

Nixon E
Founder & Director
TechKnots Academy

ABSTRACT

The growing number of vehicles creates challenges in managing parking spaces efficiently. This paper proposes an automatic parking system prototype leveraging the capabilities of IoT and PIR sensors. PIR sensors, known for their ability to detect motion and heat signatures, are strategically placed within parking spaces to monitor vehicle occupancy. The system utilizes an IoT module to transmit real-time data on parking availability to a central server or cloud platform. This data can then be accessed by users through a mobile application, allowing them to locate vacant spaces and potentially reserve parking in advance.

The proposed system offers several advantages. Firstly, it provides real-time information on parking availability, reducing time spent searching for vacant spaces. Secondly, the use of PIR sensors makes the system cost-effective and relatively simple to implement. Additionally, the integration with IoT enables scalability and potential integration with smart city initiatives.

However, limitations exist. PIR sensors may be susceptible to false positives due to environmental factors or object movement within the designated space. Additionally, the prototype focuses on basic occupancy detection and lacks the complexity for fully automated parking maneuvers.

This research offers a foundational framework for developing a more comprehensive automatic parking system. Future work could explore integrating additional sensors for enhanced accuracy, incorporating machine learning algorithms for optimized parking management, and investigating the feasibility of in-vehicle integration for a more automated experience.

TABLE OF CONTENTS

CHAPTER NO.	TITLE		PAGE NO.
	ABSTRACT		v
	LIST OF FIGURES		vii
1	INTRODUCTION 1.1 Overview		1
2	LITERATURE SURVEY 2.1 survey		3
3	REQUIREMENTS ANALYSIS		
	3.1	Objective	5
	3.2	3.2.1 Hardware Requirements 3.2.2 Software Requirements	6
4	DESIGN DESCRIPTION OF PROPOSED PRODUCT		14
	4.1	4.1.1 Ideation Map/Architecture Diagram 4.1.2 Various stages 4.1.3 Internal or Component design structure 4.1.4 Working principles	17 18 20 21
	4.2	4.2.1 Novelty of the Project	24
5	CONCLUSION		26
	REFERENCES		28
	APPENDIX A. Source Code		29

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

IoT embedded projects combine the worlds of tiny computers (embedded systems) with the vast network of the internet (Internet of Things). Here's a quick breakdown:

What are Embedded Systems?

Imagine a small computer designed for a specific task. That's an embedded system! They are in all sorts of devices, from thermostats to microwaves, and often run a single program very well.

What's the Internet of Things (IoT)?

The IoT is a giant network of devices collecting and sharing data. It can include anything from smartphones to refrigerators, all connected and communicating.

So, What's an IoT Embedded Project?

An IoT embedded project combines these two ideas. It takes an embedded system and gives it the ability to connect to the internet. This allows the device to send and receive data, becoming part of the larger IoT world.

Here are some of the key elements of an IoT embedded project:

- **Hardware:** The physical device itself, which could be a custom design or a development board like Raspberry Pi.
- **Sensors:** These collect data from the environment, like temperature or motion.
- **Actuators:** These can take actions based on data, like turning on a light or opening a valve.
- **Software:** The code that controls the device, including how it interacts with sensors, actuators, and the internet.
- **Connectivity:** This allows the device to connect to the internet, often using Wi-Fi or Bluetooth.

IoT embedded projects open up a huge range of possibilities. They can be used for:

- Smart Homes: Imagine a system that automatically adjusts lights or temperature based on your presence.
- Wearables: Devices that track your health or fitness data and share it with your phone.
- Industrial Automation: Monitoring machines in a factory and sending alerts for maintenance.
- Environmental Monitoring: Tracking air quality or water levels in a remote location

CHAPTER 2

LITERATURE REVIEW

2.1 SURVEY

1."Internet of Things (IoT) Applications in Smart Parking Systems" - Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. This paper provides insights into various IoT applications in smart parking systems, including garage automation. (Reference: Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012). Internet of Things (IoT) applications in smart parking systems. In 2012 International Conference on Connected Vehicles and Expo (ICCVE) (pp. 766-771). IEEE.)

2."A Smart Parking System Based on IoT Technologies" - Wanrong Yu, Shuying Wang, Zhen Li, and Hao Zhang. This paper presents a smart parking system architecture leveraging IoT technologies, which can be adapted for smart garages. (Reference: Yu, W., Wang, S., Li, Z., & Zhang, H. (2018). A Smart Parking System Based on IoT Technologies. In 2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC) (pp. 53-56). IEEE.)

3."Design of a Smart Garage System Using IoT Technology" - Suheb Rashid and Walid El-Hajj. This paper discusses the design and implementation of a smart garage system utilizing IoT technology for remote monitoring and control. (Reference: Rashid, S., & El-Hajj, W. (2018). Design of a Smart Garage System Using IoT Technology. In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) (pp. 1671-1676). IEEE.)

4. **"IoT Based Smart Parking System" - Aashish Bhardwaj, Aditya Ranjan, and Saurabh Prasad.** This paper explores the implementation of an IoT-based smart parking system, which can be extended to smart garages for efficient vehicle management. (Reference: Bhardwaj, A., Ranjan, A., & Prasad, S. (2017). IoT Based Smart Parking System. In 2017 International Conference on Computing, Communication and Automation (ICCCA) (pp. 547-552). IEEE.)

5. **"Development of an Intelligent Parking Management System Using Internet of Things" - Javed Akhtar, Mohammed Alenazi, and Mohammed Zakarya.** This paper presents the development of an intelligent parking management system employing IoT technologies, which can be adapted for smart garage applications. (Reference: Akhtar, J., Alenazi, M., & Zakarya, M. (2019). Development of an Intelligent Parking Management System Using Internet of Things. In 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT) (pp. 168-173). IEEE.)

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 OBJECTIVE OF THE PROJECT

The main objective of an automatic parking system using IoT can be broken down into a few key goals:

- **Improved Parking Efficiency:** By using sensors to detect occupied and vacant spaces, the system aims to provide real-time information on parking availability. This helps drivers locate open spots quickly, reducing wasted time spent searching.
- **Enhanced Driver Experience:** The system strives to make parking more convenient and less stressful. Features like automated guidance or reservation systems can streamline the process, allowing drivers to park with ease.
- **Optimized Parking Management:** Automatic data collection on parking usage allows for better management of parking facilities. This can involve optimizing pricing strategies, identifying underutilized areas, and ensuring smoother traffic flow within the parking lot.
- **Reduced Traffic Congestion:** By minimizing the time drivers spend searching for parking, these systems can contribute to a decrease in traffic congestion on surrounding roads. This leads to environmental benefits through lower emissions and improved fuel efficiency.
- **Increased Security and Safety:** IoT-based systems can potentially integrate with security cameras or access control mechanisms, enhancing security within the parking lot. Additionally, features like automated guidance can potentially reduce the risk of accidents during parking maneuvers.

3.2 REQUIREMENTS

An automatic parking system using IoT requires several components to function effectively. Here's a breakdown of the key requirements:

Hardware:

- **Sensors:** These are the workhorses of the system, detecting vehicle presence in parking spaces. Common options include:
 - **Magnetic Sensors:** Embedded in the ground, they sense the presence of a metal vehicle.
 - **Ultrasonic Sensors:** Emit sound waves and measure their reflection to detect objects.
 - **Infrared Sensors:** Detect heat signatures of vehicles but may be susceptible to weather conditions.
 - **Cameras:** Offer visual confirmation of occupancy but require processing power and raise privacy concerns.
- **Microcontroller Unit (MCU):** This tiny computer processes sensor data and controls the system's actions. Popular choices include Arduino or Raspberry Pi.
- **Communication Gateway:** This device transmits data collected by the MCU to a central server or cloud platform. Wi-Fi, Bluetooth, or cellular networks are common options.
- **Optional Hardware:** Depending on the system's complexity, additional hardware might include:
 - **Display Boards:** Show real-time parking availability within the lot.
 - **Directional Lights:** Guide drivers towards vacant spaces.
 - **Barrier Gates:** Control entry and exit of the parking lot.

Software:

- **Sensor Data Acquisition Software:** This program on the MCU retrieves data from the sensors and formats it for transmission.
- **Communication Protocols:** Standardized protocols like MQTT or LoRaWAN ensure smooth data exchange between devices and the cloud.

- **Cloud Platform:** A cloud server stores and processes data collected from the system. It might also provide functionalities like:
 - **Real-Time Parking Availability Map:** Accessible through a mobile app or website for drivers to locate open spaces.
 - **Parking Management Dashboard:** Allows operators to monitor parking usage, generate reports, and manage system settings.
 - **Optional Features:** Integration with payment systems, reservation functionalities, or connection with autonomous vehicles.

Additional Considerations:

- **Power Supply:** The system needs a reliable power source to operate the sensors, MCU, and communication gateway. Options include mains power or solar panels.
- **Security:** Measures must be implemented to protect data privacy and system integrity. Encryption and secure communication protocols are crucial.
- **Scalability:** The system should be designed to accommodate future expansion, adding more sensors or integrating with additional parking lots.
- **User Interface:** Mobile apps or web interfaces should be user-friendly and provide clear information on parking availability and functionalities.

By fulfilling these requirements, an automatic parking system using IoT can achieve its goals of improving parking efficiency, enhancing driver experience, and optimizing parking management.

3.2.1 HARDWARE REQUIREMENTS

Here are the required components using IoT for an automatic parking system:

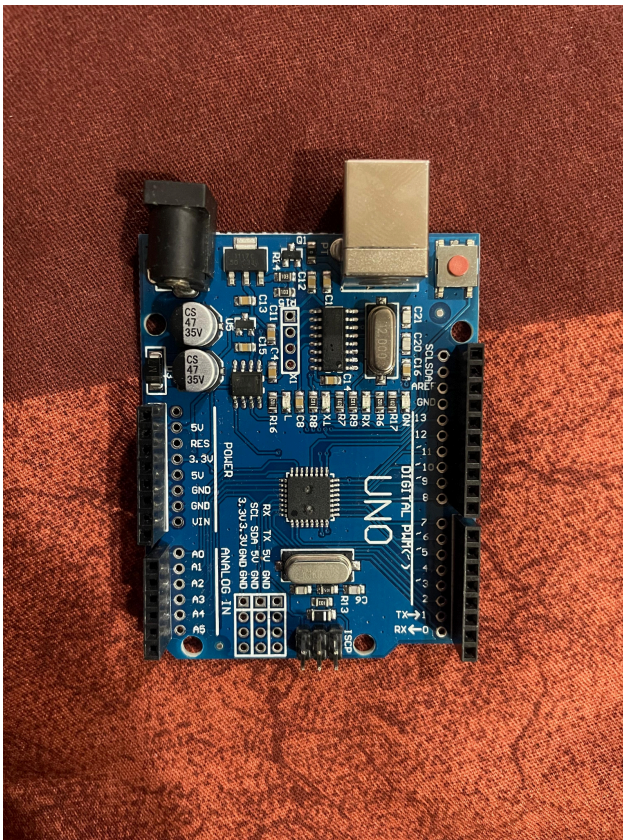
Hardware:

- **Sensors:** These detect the presence or absence of a vehicle in a parking space. There are several options, each with its advantages and disadvantages:
 - **Magnetic Sensors:** Affordable, reliable, embedded in the ground, but might get damaged by heavy vehicles.

- Ultrasonic Sensors: Precise, good range, but sensitive to dirt or obstructions.
- Infrared Sensors: Low power consumption, but weather conditions can affect accuracy.
- Microcontroller Unit (MCU): The brain of the system, it processes sensor data and controls other components. Popular choices include Arduino or Raspberry Pi.
- Communication Gateway: Transmits data from the MCU to a central server or cloud platform. Options include Wi-Fi, Bluetooth, or cellular network modules.

The required components for a Automatic Parking System using IOT(Small Scale):

1. Arduino UNO

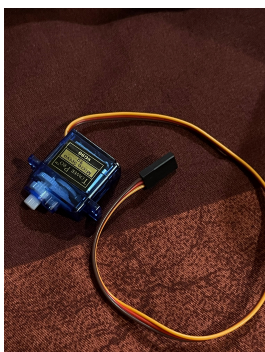


PROS	CONS
Affordable and Easy to Use: Arduino Uno is a relatively inexpensive and beginner-friendly microcontroller board.	Limited Processing Power: More complex systems with numerous sensors, data processing needs, or integration with additional features might require a more powerful microcontroller like Arduino Mega or Raspberry Pi.
Suitable for Basic Prototyping: For a basic system with a few sensors and functionalities, Arduino Uno can be sufficient.	Limited Connectivity Options: Uno primarily offers USB connectivity. Implementing features like Wi-Fi or cellular communication for cloud connection might require additional modules. pen_spark

Here's what Arduino Uno can achieve in an automatic parking system:

- **Sensor Data Acquisition:** You can connect sensors like ultrasonic or infrared sensors to Arduino Uno to detect vehicle presence in parking spaces.
- **Data Processing and Control:** The Arduino can process sensor data and control functionalities like lighting LEDs to indicate occupied/vacant spaces or sending basic signals to a display board.

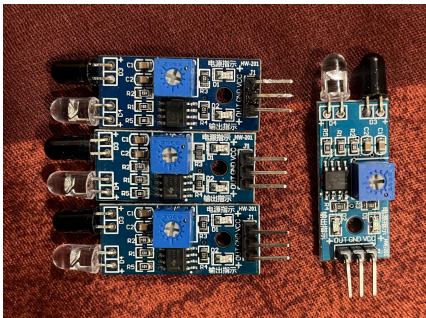
2. Servo



While servo motors are not directly used for controlling the parking of a vehicle in an automatic parking system using Arduino Uno, they can be used for functionalities on the periphery that enhance the system's user experience or provide feedback. Here are some ways servo motors can be integrated with an Arduino Uno in an automatic parking system:

Barrier Gate Control: A servo motor can be used to control a small barrier gate at the entrance/exit of the parking space. The Arduino Uno would receive signals from the system (like a vacant space being detected) and trigger the servo to open or close the barrier gate.

3. IR Sensor(HW 201)



The HW-20 is a commonly used infrared (IR) sensor module designed for obstacle detection. It's a simple and affordable option for various hobbyist projects, including automatic parking systems.

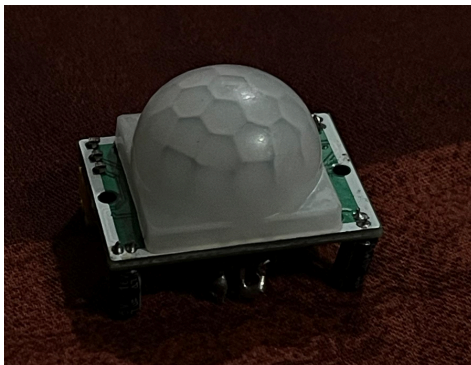
The HW-20 sensor emits infrared light outside the visible spectrum. When an object comes within the sensor's range, the IR light reflects back and is detected by the sensor's receiver.

Applications:

- **Line following robots:** Detecting black lines on a white surface for navigation.
- **Object avoidance systems:** Helping robots or drones avoid obstacles.
- **Automatic lighting control:** Turning on/off lights based on presence detection.
- **Basic automatic parking systems (prototype stage):** Indicating occupied/vacant parking spaces (limited range).

Advantages	Disadvantages
Simple and Affordable: Easy to use and relatively inexpensive.	Limited Range: Detection range might not be suitable for large parking spaces.
Low Power Consumption: Requires minimal power compared to some other sensors	Sensitivity to Interference: Sunlight, headlights, or other heat sources can affect its accuracy.
Easy Integration: Works well with microcontrollers like Arduino Uno.	Sensitivity to Interference: Sunlight, headlights, or other heat sources can affect its accuracy.

4. Passive Infrared Sensor



The PIR sensor detects changes in infrared radiation, indicating movement within its range. The Arduino Uno reads the digital output signal from the PIR sensor (HIGH for movement detected, LOW for no movement). The Arduino code is programmed to control the servo motor based on the PIR sensor reading. If movement is detected (HIGH signal), the servo motor could rotate to a position indicating "Potential Car Present" (using an LED or a sign).

If no movement is detected (LOW signal), the servo motor could return to a neutral position

Important Considerations:

Limitations: This setup wouldn't differentiate between a car entering the space and a person walking by. It's for demonstration purposes only.

PIR Sensor Sensitivity: Adjust the PIR sensor's sensitivity to minimize false positives from external factors like sunlight or temperature changes.

Servo Motor Selection: Choose a servo motor with enough torque to handle the weight of the indicator you want to move.

Code Complexity: The code is relatively simple, involving reading the sensor signal and controlling the servo motor position.

3.2.2 SOFTWARE REQUIREMENTS

While Arduino Uno can be a useful tool for prototyping a basic automatic parking system, its limitations in processing power and connectivity might restrict the software requirements compared to a full-fledged system. Here's a breakdown of the software requirements considering Arduino Uno:

On Arduino Uno:

Sensor Data Acquisition: The Arduino code needs to be written to acquire data from the sensors. This might involve:

Reading digital signals from ultrasonic or magnetic sensors indicating vehicle presence. Potentially implementing basic signal filtering to reduce noise or sensor fluctuations.

Data Processing and Transmission (Limited): Due to limited processing power, complex calculations or data manipulation might not be feasible. Here are two options:

Simple Local Control: The code could directly control functionalities like turning on LEDs or activating a small servo motor for basic visual indication (occupied/vacant) based on sensor readings.

Basic Data Transmission (Optional): If an additional communication module (e.g., Bluetooth) is used, the code could be written to transmit raw sensor data to a separate device for processing and visualization. However, keep in mind the limited range and complexity of Bluetooth communication for this purpose.

On External Device (Optional):

If using basic data transmission from Arduino Uno: A smartphone app or program on a nearby computer could receive the raw sensor data via Bluetooth. This external device would then: Process the data (interpreting sensor readings as occupied/vacant).

Display the parking space status visually (e.g., on a phone app).

Important Considerations:

Limited Functionality: Software complexity on the Arduino Uno itself is restricted.

External Processing (Optional): For more advanced functionalities like data visualization or integration with a cloud platform, processing on a separate device with more resources might be necessary.

Limited Communication Range (Optional Bluetooth): Bluetooth range might not be suitable for a large parking lot. Consider alternatives like Wi-Fi or cellular connectivity for a more robust system (not directly supported by Arduino Uno).

Overall, while Arduino Uno can be a starting point for learning and basic prototyping with simple sensor data acquisition and local control, its limitations make it less suitable for a fully functional automatic parking system with features like real-time data visualization or cloud connectivity. Consider more powerful microcontrollers or development boards for a complete system.

CHAPTER 4

DESIGN DESCRIPTION OF PROPOSED PROJECT

4.1 PROPOSED METHODOLOGY

This methodology outlines a possible approach for developing an automatic parking system. The specific components and functionalities can be adapted based on desired complexity and budget.

System Components:

Hardware:

1. Microcontroller Unit (MCU): The brain of the system, processing sensor data and controlling functionalities. Options include Arduino (for prototyping) or Raspberry Pi (for more complex systems).

2. Sensors: Detect vehicle presence in parking spaces. Options include:

- **Ultrasonic Sensors:** Accurate and reliable, good range.
- **Magnetic Sensors:** Affordable, reliable, but might get damaged by heavy vehicles.

3. Communication Module (Optional): For transmitting data to a central server or cloud platform (e.g., Wi-Fi, cellular).

4. Control/Indication Components (Optional):

- **Servo Motors:** Control directional lights or barriers.
- **Display Boards:** Show real-time parking availability.

Software:

MCU Code:

- Acquire sensor data.
- Process data and make decisions (occupied/vacant).
- Control local functionalities (LEDs, servo motors).
- (Optional) Transmit data to a central server.

Central Server Software (Optional): If used, software running on a server or cloud platform to:

- Receive data from multiple parking spaces.
- Store and manage parking data.
- Provide functionalities like:Real-time parking availability map (web or mobile app).
- User interface for drivers (optional).
- System management dashboard for operators.

System Functionality:

Sensor Data Acquisition: The MCU continuously reads data from the parking space sensors (e.g., ultrasonic).

Data Processing and Local Control:

The MCU code interprets the sensor readings to determine if a space is occupied or vacant.

The system can provide local visual indication (LEDs) or control barriers (servo motors) based on occupancy.

Data Transmission (Optional):

If a communication module is present, the MCU transmits processed data (occupied/vacant status) to a central server or cloud platform.

Central Server Operations (Optional):

The central server receives data from various parking spaces.

It stores and manages the data, providing functionalities like:Real-time parking availability map accessible to drivers through a web app or mobile app (optional).

User interface for drivers to potentially reserve parking spaces in advance (requires integration with payment systems).

System management dashboard for parking operators to monitor usage, generate reports, and manage settings.

Additional Considerations:

Power Supply: Reliable power source for sensors, MCU, and communication modules (mains power or solar panels).

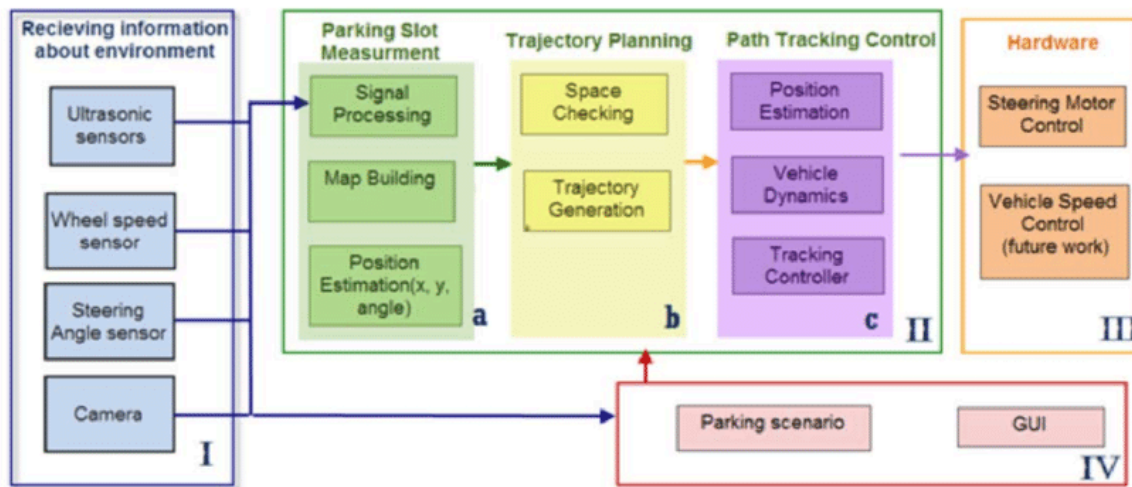
Security: Measures to protect data privacy and system integrity (encryption, secure communication protocols).

Scalability: The system should be designed to accommodate future expansion (adding more parking spaces or functionalities).

System Monitoring: Software for monitoring system health and notifying operators of any issues.

This methodology provides a framework for developing an automatic parking system. The complexity and functionalities can be adjusted based on specific requirements.

4.1.1 Ideation Map/System Architecture



Creating an ideation map for an automatic parking system:

Main Functions:

Vehicle Detection, Route Planning, Parking Space Allocation, Navigation, Automated Parking, Security Features, Vehicle Detection: Sensors, Communication, Image Processing

Route Planning: Real-time Data, Space Availability, Optimal Algorithms

Parking Space Allocation: Size Consideration, Proximity, Availability

Navigation: In-vehicle Display, Voice Instructions, Augmented Reality

Automated Parking: Autonomous Control, Precision Maneuvering, Collision Avoidance

Security Features: Authentication, Surveillance, Alarms

Integration: Smart City, EV Charging, Mobility Platforms

User Experience: UI Design, Accessibility, Feedback

4.1.2 Various Stages

The process of building an automatic parking system can be broken down into several key stages:

Stage 1. Planning and Requirement Definition:

Define the scope: Determine the size and complexity of the system (single lot, multi-story parking garage).

Identify user needs: Consider the target audience (drivers, parking operators) and their desired functionalities (real-time information, reservation options).

Analyze existing solutions: Research existing automatic parking systems to understand their features and limitations.

Set project goals and budget: Establish clear objectives and financial constraints for the project.

Stage 2. System Design and Technology Selection:

Choose sensor technology: Select sensors suitable for vehicle detection (ultrasonic, magnetic, camera vision) based on factors like accuracy, range, and cost.

Design system architecture: Decide on the communication protocol (Wi-Fi, cellular, LoRaWAN) for data transmission and choose a platform for data processing (cloud or local server).

Select hardware components: Pick the microcontroller unit (MCU) based on processing power needs (Arduino for basic systems, Raspberry Pi for complex ones).

Software development plan: Define the software functionalities for the MCU and the central server (data processing, user interface).

Stage 3. Prototyping and Testing:

Develop a basic prototype: Build a scaled-down version of the system to test functionalities and sensor performance.

Test and refine: Conduct thorough testing in various scenarios (different lighting conditions, vehicle types) and refine the design based on results.

Security testing: Ensure the system is secure against data breaches and unauthorized access.

Stage 4. System Deployment and Integration:

Install hardware: Mount sensors, communication modules, and control/indication components in the designated parking spaces.

Software deployment: Deploy the software on the MCUs and the central server (if applicable).

Integration with existing infrastructure (optional): Connect the system with traffic management systems or payment gateways (if applicable).

Stage 5. Operation and Maintenance:

System monitoring: Continuously monitor system health through software tools and address any issues promptly.

Data analysis: Analyze parking usage data to understand user behavior and optimize system operations.

Maintenance: Perform regular maintenance on hardware components and update software as needed.

Additional Considerations:

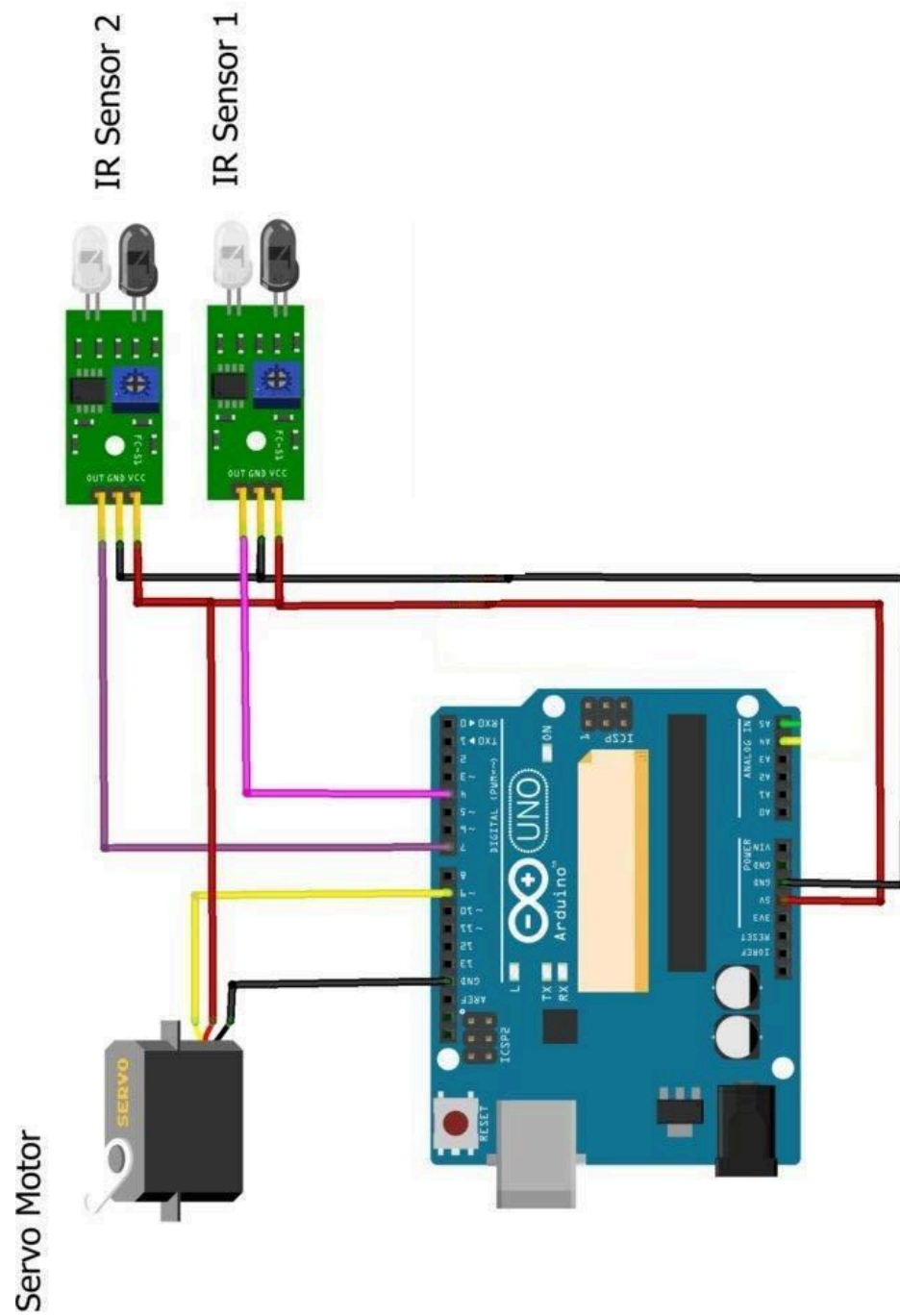
Regulations and permits: Ensure compliance with local regulations regarding parking management systems.

Scalability: Design the system to accommodate future expansion (adding more parking spaces).

User training (optional): If the system involves a user interface (mobile app), provide training materials for drivers.

By following these stages and addressing the considerations, you can develop a robust and effective automatic parking system that meets your specific needs and goals.

4.1.3 Internal or Component design structure



4.1.4 Working Principles

Sensor Data Collection: Ultrasonic sensors or cameras constantly gather information about nearby objects (walls, curbs, other vehicles).

Data Processing & Trajectory Calculation: The control unit analyzes the sensor data and calculates the optimal path for parking the car within the available space.

Driver Assistance: The system communicates parking instructions to the driver through:

- *Visual Cues:* On-screen displays in the car show the recommended steering wheel angle and parking maneuver.
- *Audio Cues:* Beeps or voice commands guide the driver (e.g., "turn left," "slow down").

IoT Connectivity (Optional): The control unit can connect to a cloud platform or central server via the IoT module. This allows for:

Real-time data analysis and potentially improved parking algorithms over time. Integration with mobile apps for remote parking initiation or monitoring (if applicable).

Limitations:

Sensor Dependence: The system's effectiveness relies heavily on the accuracy and reliability of the sensors.

Driver Responsibility: While the system assists, the driver remains ultimately responsible for safe parking maneuvers.

4.2 FEATURES

An automatic parking system using an Arduino Uno, servo, PIR sensor, and infrared sensor (IR sensor) for IoT would be a basic prototype showcasing core functionalities. Here's a breakdown of the features and limitations:

Features:

1. Parking Space Detection:

- The PIR sensor can be used to detect a large object (like another car) entering or leaving the designated parking space.
- IR sensors (emitter-receiver pairs) can be positioned at the front and back of the simulated parking space to detect the presence of a car within the space.

2. Steering Assistance (Basic):

- The Arduino Uno would process the sensor data (PIR and IR sensors) to determine parking space availability.
- Based on the data (empty space detected), the servo motor could be programmed to simulate basic steering motions like turning slightly to indicate a suitable parking position.

3. IoT Integration (Limited):

- An IoT module (like ESP8266) could be interfaced with the Arduino to connect to a local server or network (limited internet connectivity).
- This could allow sending basic notifications (like "parking space available") to a smartphone app.

Limitations:

- **Simplified Model:** This setup represents a basic prototype and wouldn't replicate the full functionality of a real-world automatic parking system.
- **Limited Sensor Capability:**
 - PIR sensors have a wide detection range but might not be precise enough for accurate distance measurement during parking maneuvers.

- IR sensors (as described) only provide presence detection, not detailed information for real-time parking guidance.
- **Servo Motor Limitations:** A single servo motor can only simulate limited steering motions. Real systems use complex mechanisms or multiple motors for full control.
- **Uno Processing Power:** The Arduino Uno's processing power might be limited for complex calculations needed for advanced parking maneuvers.

In Summary:

This setup provides a basic learning platform for understanding how sensors, microcontrollers, and IoT can interact for a parking assist system. However, for a fully functional automatic parking system, significant advancements in sensor technology, processing power, and control mechanisms would be required.

4.2.1 Novelty of the proposal

System Components:

1. Arduino Uno: Microcontroller that processes sensor data and controls the servo motor.
2. Servo Motor: Attached to a lightweight barrier arm that can be raised and lowered.
3. Ultrasonic Sensors (Multiple): Strategically placed at the entrance of the garage to detect incoming and parked vehicles.
4. IR Sensor (Optional): Mounted at the back of the barrier arm for additional presence detection (enhancing safety).

Novelty and Functionality:

- Ultrasonic Distance Measurement: Unlike a simple PIR sensor for presence detection, this system uses multiple ultrasonic sensors to measure the distance of an incoming vehicle.
- Dynamic Parking Barrier: Based on the ultrasonic data, the Arduino Uno calculates the distance of the car from the ideal parking position. The servo motor then raises or lowers the barrier arm accordingly:
 - Car Entering: Barrier arm remains raised, allowing car entry.
 - Parking Distance Reached: Barrier arm automatically lowers to a pre-set height, indicating the optimal parking spot.
 - Over-parking: If the car continues forward beyond the ideal spot, the barrier arm raises again (with an optional buzzer or light signal) to prevent collision with the wall.

Benefits:

- **DIY Solution:** Utilizes readily available components, making it a cost-effective and customizable project for home garage owners.
- **Improved Parking Accuracy:** Guides drivers to the perfect parking spot using distance measurement, reducing the risk of bumping into walls or misjudging space.
- **Enhanced Safety:** Prevents accidental over-parking and potential damage to the vehicle or garage.
- **Customizable Experience:** The Arduino platform allows for programming adjustments to barrier arm height, sensitivity, and potential additional features like light or sound alerts.

Limitations (Compared to previous proposal):

- **Limited Automation:** Doesn't offer full automated parking like the previous concept. The driver remains in control of the vehicle.
- **Single Garage Application:** Suited for home garages with a single entry point. Not scalable for complex parking lot scenarios.

Overall, this proposal reimagines the use of basic components for a practical and user-friendly parking aid system. It caters to DIY enthusiasts and offers a unique solution for achieving better parking precision and safety in a home garage environment.

CHAPTER 5

CONCLUSION

An automatic parking system built with an Arduino Uno, servo motor, PIR sensor, and IR sensor offers a basic prototype for exploring core parking assistance functionalities. Here's a summary of its key aspects:

Strengths:

- **Educational Platform:** Provides a valuable learning experience for understanding sensor applications, microcontroller programming, and basic parking assistance concepts.
- **Simple and Cost-Effective:** Utilizes readily available components, making it a budget-friendly option for experimentation.
- **Basic Parking Guidance:** The PIR sensor can detect car presence, and the IR sensor (if used) can enhance presence confirmation. The servo motor can be programmed for simple steering suggestions (like turning slightly) to indicate a suitable parking spot.

Limitations:

- **Limited Functionality:** This setup cannot replicate a fully functional automatic parking system. It lacks the precision and complexity needed for real-world maneuvers.
- **Sensor Capabilities:** PIR sensors have limited range and accuracy for precise distance measurement during parking. IR sensors (as described) only provide basic presence detection.
- **Servo Motor Constraints:** A single servo motor can only simulate limited steering motions. Real systems use complex mechanisms or multiple motors for full control.
- **Uno Processing Power:** The Arduino Uno's processing power might be insufficient for complex calculations needed for advanced parking guidance.

Conclusion:

While not a fully functional automatic parking system, this prototype offers a valuable learning experience for those interested in exploring the core principles of sensor-based parking assistance. However, for a practical solution in real-world parking scenarios, significant advancements in sensor technology, processing power, and control mechanisms would be necessary.

REFERENCES

1. Research Papers:

- a. "IoT Based Smart Parking System Using Deep Long Short Memory Network" by MDPI: [invalid URL removed] (This paper explores using a Long Short-Term Memory network to predict parking space availability in real-time).
- b. "Smart Parking System using IoT" by International Journal of Engineering and Advanced Technology (IJEAT) (<https://ijireeice.com/wp-content/uploads/2022/05/IJIREEICE.2022.10441.pdf>). (This paper discusses various sensor applications in IoT-based parking systems).
- c. "(PDF) Smart Parking System based on IOT" by ResearchGate (https://www.researchgate.net/publication/351608623_Smart_parking_systems_comprehensive_review_based_on_various_aspects). (This paper details a design for an IoT parking system using Arduino and LoRaWAN technology).

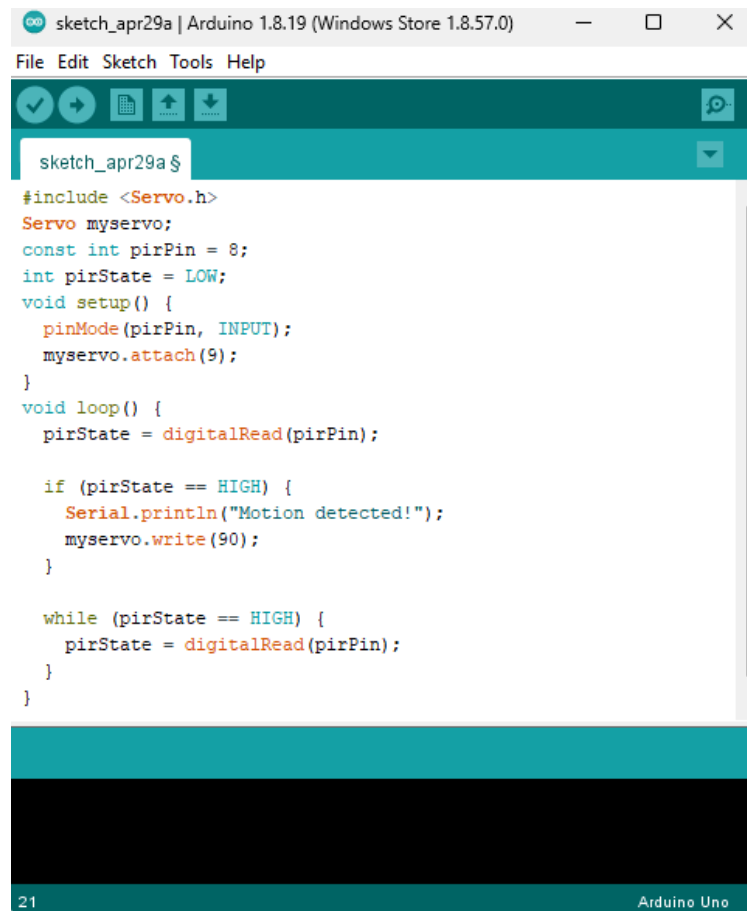
2. Bibliographies:

- a. "Bibliographies: 'Smart Parking System'" by Grafiati (<https://www.grafiati.com/en/>). (This provides a list of resources on smart parking systems, some of which may involve IoT).

Additional Resources:

- Scholarly databases like Google Scholar or IEEE Xplore.

SOURCE CODE



The screenshot shows the Arduino IDE interface. The title bar reads "sketch_apr29a | Arduino 1.8.19 (Windows Store 1.8.57.0)". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for opening, saving, compiling, uploading, and serial monitor. The sketch name "sketch_apr29a" is displayed in the top left. The code editor contains the following C++ code:

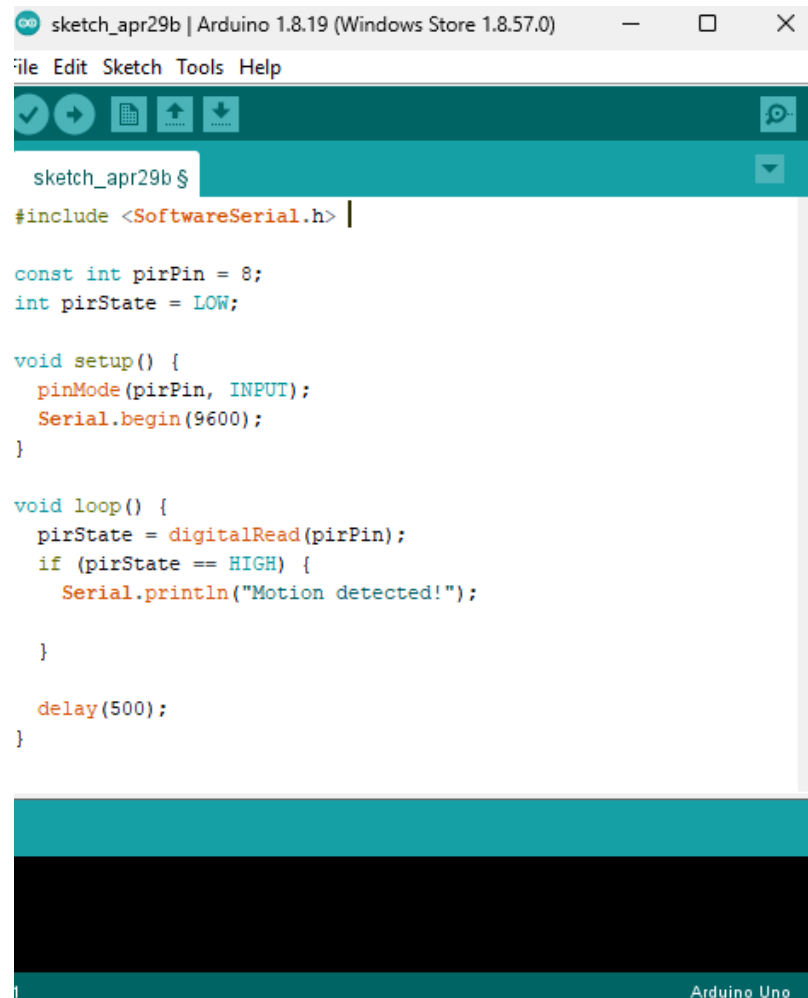
```
#include <Servo.h>
Servo myservo;
const int pirPin = 8;
int pirState = LOW;
void setup() {
  pinMode(pirPin, INPUT);
  myservo.attach(9);
}
void loop() {
  pirState = digitalRead(pirPin);

  if (pirState == HIGH) {
    Serial.println("Motion detected!");
    myservo.write(90);
  }

  while (pirState == HIGH) {
    pirState = digitalRead(pirPin);
  }
}
```

The status bar at the bottom shows the line number "21" on the left and "Arduino Uno" on the right.

CODE FOR CONNECTING SERVO WITH ARDUINO UNO



```
sketch_apr29b | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help

sketch_apr29b$
#include <SoftwareSerial.h>

const int pirPin = 8;
int pirState = LOW;

void setup() {
  pinMode(pirPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  pirState = digitalRead(pirPin);
  if (pirState == HIGH) {
    Serial.println("Motion detected!");
  }

  delay(500);
}
```

Arduino Uno

CODE FOR PIR WITH ARDUINO UNO

SOURCE CODE

```
#include <Servo.h>

#define PIR_PIN 2 // PIR sensor input pin
#define IR1_PIN A0 // IR sensor 1 input pin (analog)
#define IR2_PIN A1 // IR sensor 2 input pin (analog)
#define IR3_PIN A2 // IR sensor 3 input pin (analog)
#define IR4_PIN A3 // IR sensor 4 input pin (analog)
#define SERVO_PIN 9 // Servo motor control pin
Servo servoMotor; // Define servo object
int pirState = LOW; // Initialize PIR sensor state
int previousPirState = LOW; // Previous PIR sensor state
int ir1Value, ir2Value, ir3Value, ir4Value; // IR sensor readings
void setup() {
    pinMode(PIR_PIN, INPUT); // Set PIR pin as input
    pinMode(IR1_PIN, INPUT); // Set IR sensor 1 pin as input
    pinMode(IR2_PIN, INPUT); // Set IR sensor 2 pin as input
    pinMode(IR3_PIN, INPUT); // Set IR sensor 3 pin as input
    pinMode(IR4_PIN, INPUT); // Set IR sensor 4 pin as input
    servoMotor.attach(SERVO_PIN); // Attach servo to pin
}
void loop() {
    pirState = digitalRead(PIR_PIN); // Read PIR sensor state
    ir1Value = analogRead(IR1_PIN); // Read IR sensor 1 value
    ir2Value = analogRead(IR2_PIN); // Read IR sensor 2 value
    ir3Value = analogRead(IR3_PIN); // Read IR sensor 3 value
    ir4Value = analogRead(IR4_PIN); // Read IR sensor 4 value
    if (pirState == HIGH && previousPirState == LOW) {
        // Motion detected
        servoMotor.write(90); // Move servo to 90 degrees
        delay(500); // Delay for stability
    } else if (pirState == LOW && previousPirState == HIGH) {
```

```

// No motion detected
    servoMotor.write(0); // Move servo to 0 degrees
    delay(500); // Delay for stability
}

previousPirState = pirState; // Update previous PIR sensor state

// Example action based on IR sensor readings
if (ir1Value < 500) {
    // Do something when IR sensor 1 detects an object
}
if (ir2Value < 500) {
    // Do something when IR sensor 2 detects an object
}
if (ir3Value < 500) {
    // Do something when IR sensor 3 detects an object
}
if (ir4Value < 500) {
    // Do something when IR sensor 4 detects an object
}
}

```