SMART CAR PARKING SLOT SYSTEM

MINI PROJECT REPORT

Submitted by

ABHINAND K ANNA 9517202109001

SAKTHI JEGANATHAN R 9517202109045

CHANDRA PRAKASH C 9517202109251

RAMKUMAR K 9517202109253

in

19AD752 – INTELLIGENT SYSTEMS FOR IOT LABORATORY

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MEPCO SCHLENK ENGINEERING COLLEGE

SIVAKASI

October 2024

MEPCOSCHLENKENGINEERINGCOLLEGE,

SIVAKASI

AUTONOMOUS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

BONAFIDE CERTIFICATE

This is to certify that it is the bonafide work of **"ABHINAND K ANNA 9517202109001, SAKTHI JEGANATHAN R 9517202109045, CHANDRA PRAKASH C 9517202109251, RAMKUMAR K 9517202109253"** for the Mini project titled **"Smart Car Parking Slot System"** in 19AD752–Intelligent Systems for IOT Laboratory during theseventh semester June 2024– October 2024 under my supervision.

SIGNATURE

Dr.A.Shenbagarajan ,
Associate Professor,
AI&DS Department,
Mepco Schlenk Engineering College
Sivakasi

SIGNATURE

Dr. J. Angela Jennifa Sujana,
Professor & Head,
AI&DS Department,
Mepco Schlenk Engineering College
Sivakasi

ABSTRACT

The Smart Car Parking System is designed to streamline the process of vehicle parking, ensuring efficient use of parking space and ease of vehicle management. The Arduino Uno as the central microcontroller to manage and monitor parking slots using Infrared (IR) sensors and control gate access with a servo motor. The system includes four IR sensors to monitor the availability of four parking slots and two additional IR sensors to detect car entry and exit. Furthermore, light posts are installed at each parking slot to indicate their availability. The project aims to automate the parking process, reduce human intervention, and enhance user convenience by providing real-time parking slot availability information.

ACKNOWLEDGEMENT

First and foremost we **praise and thank "The Almighty",** the lord of all creations, who by his abundant grace has sustained us and helped us to work on this project successfully.

We really find unique pleasure and immense gratitude in thanking our respected management members**,** who is the backbone of our college.

A deep bouquet of thanks to respected Principal **Dr.S.Arivazhagan M.E.,Ph.D.,** for having provided the facilities required for our mini project.

We sincerely thank our Head of the Department **Dr. J. Angela Jennifa Sujana M.E.,Ph.D.,** Professor & Head, Department of Artificial Intelligence and Data Science, for herguidance and support throughout the mini project .

We extremely thank our project coordinator **Dr.A.Shenbagarajan M.E.,Ph.D,** Associate Professor and **Dr.E.Emerson Nithiyaraj M.E., Ph.D,** Assistant Professor, Department of Artificial Intelligence and Data Science, who inspired us andsupported us throughout the mini project.

We extend our heartfelt thanks and profound gratitude to all the faculty members of Artificial Intelligence and Data Science department for their kind help during our mini projectwork.

We also thank our parents and our friends who had been providing us with constant support during the course of the mini project work.

TABLE OF CONTENT

LIST OF FIGURES

1. INTRODUCTION

1.1 INTRODUCTION

As urbanization accelerates and vehicle ownership continues to rise, efficient parking management has become a critical challenge in urban areas. Traditional parking systems often lead to congestion, inefficient use of space, and frustration for drivers searching for available parking spots. To address these issues, the Smart Car Parking System offers a modern, IoT-enabled solution designed to streamline the parking process, improve space utilization, and enhance user convenience.

1.2 OBJECTIVES

2 **Automate Parking Management**: Utilize IR sensors and an Arduino Uno to automatically monitor and manage parking slots.

3 **Real-Time Slot Availability**: Provide real-time updates on parking slot status to inform drivers of available spaces.

4 **Efficient Space Utilization**: Ensure optimal use of parking spaces through automated entry and exit control.

5 **User Convenience**: Reduce the time and effort required for drivers to find available parking slots.

6 **Scalability**: Design a system that can be easily scaled to accommodate more parking slots or larger parking facilities.

1.3 SCOPE OF THE PROJECT

The Automatic Water Dispenser project focuses on developing an IoT-based system that automates water dispensing and refilling processes. The scope of the project includes the following:

1. Hardware Design

- **Integration of IR Sensors**:
  - Deployment of four IR sensors to monitor the occupancy status of four individual parking slots.
  - Utilization of two additional IR sensors to track vehicle entry and exit at the parking lot entrance and exit points.
- **Use of Arduino Uno**:
  - Employing Arduino Uno as the central microcontroller to process sensor data and control various system components.
- **Servo Motor Control**:
  - Implementing a servo motor to manage the gate barrier, enabling it to open and close based on parking slot availability.
- **Light Indicator System**:
  - Installing light posts at each parking slot to provide visual indicators. Lights turn off when a slot is occupied and remain on when the slot is available.
  - Activating a red light at the entrance to signal when the parking lot is full.

2. Software Development

- **Sensor Data Processing**:
  - Writing programs to read data from IR sensors and determine the occupancy status of parking slots.
  - Implementing logic to control the servo motor for gate operation based on real-time parking slot availability.
- **Real-Time Monitoring and Control**:
  - Developing a system to continuously monitor parking slot status and update light indicators in real time.
  - Ensuring efficient control of the gate barrier to allow or restrict vehicle entry based on the availability of parking slots.

3. Testing and Validation

- **System Testing**:
  - Conducting extensive testing under various scenarios to ensure accurate detection of vehicle presence and reliable operation of the gate barrier and light indicators.
- **Performance Validation**:
  - Validating the system's reliability in maintaining accurate real-time updates of parking slot status and effective control of vehicle entry and exit.

4. Real-World Applications

- **Urban Parking Management**:
  - Deploying the system in urban parking facilities to reduce congestion, optimize space utilization, and enhance user convenience.
- **Commercial and Residential Buildings**:
  - Integrating the system into commercial and residential buildings to provide efficient and automated parking management solutions.
- **Scalable Parking Solutions**:
  - Adapting the system for various sizes and types of parking facilities, from small lots to large multi-level parking structures.

5. Future Enhancements

- **Integration with Mobile Apps**:
  - Developing mobile applications to provide drivers with real-time updates on parking slot availability and facilitate easy navigation to available spots.
- **Remote Monitoring and Control**:
  - Expanding the system to include remote monitoring and control capabilities via cloud-based platforms for enhanced scalability and management.
- **Advanced Sensor Integration**:
  - Incorporating additional sensors such as ultrasonic or camera-based systems for more precise vehicle detection and slot monitoring.

2. PROPOSED SOLUTION

2.1 BLOCK DIAGRAM

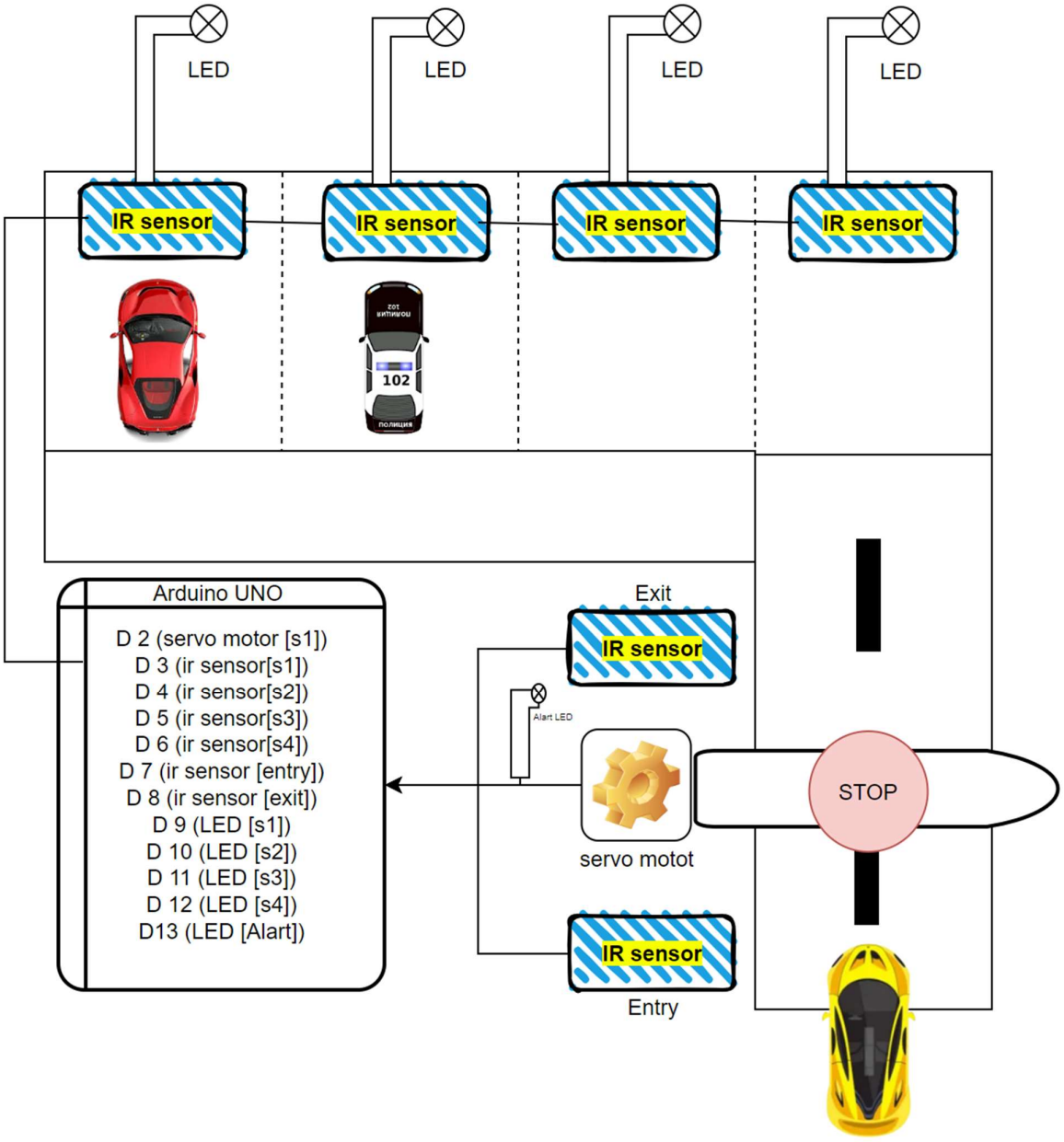

Figure 2.1.1 Block Diagram

2.2 SYSTEM BEHAVIOR:

2.2.1 Parking Slot Monitoring

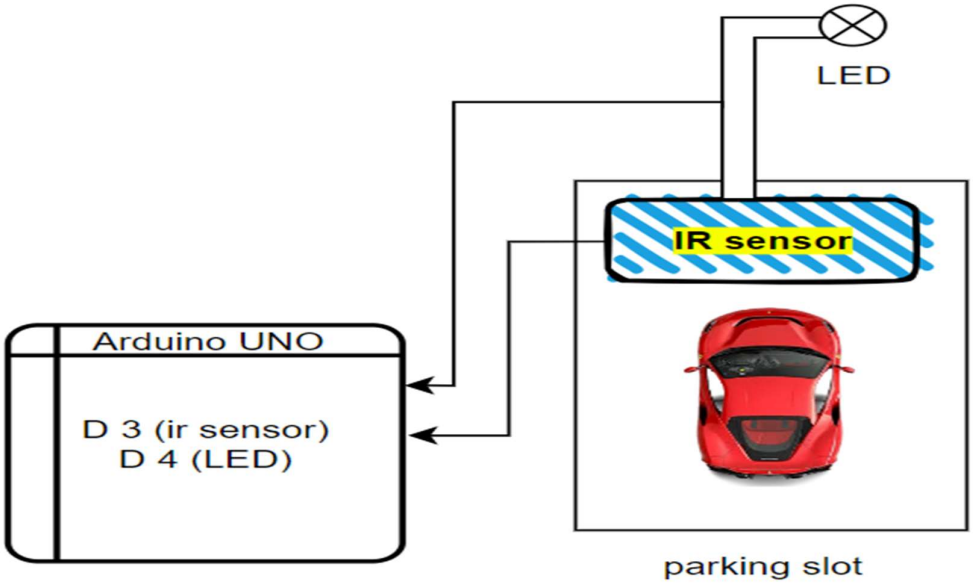

Figure 2.2.1 Parking Slot Monitoring

1. **Occupancy Detection**:
    - Each of the four parking slots is equipped with an IR sensor.
    - The IR sensors continuously monitor the presence of vehicles in their respective slots.
    - When a vehicle occupies a slot, the corresponding IR sensor detects the obstruction and sends a signal to the Arduino Uno.
2. **Visual Indicators**:
    - Each parking slot has a light post acting as a visual indicator.
    - When a slot is occupied, the light post for that slot turns off, indicating the slot is taken.
    - When a slot is available, the light post remains on, signaling the availability to incoming drivers.

2.2.2 Vehicle Entry and Exit Monitoring

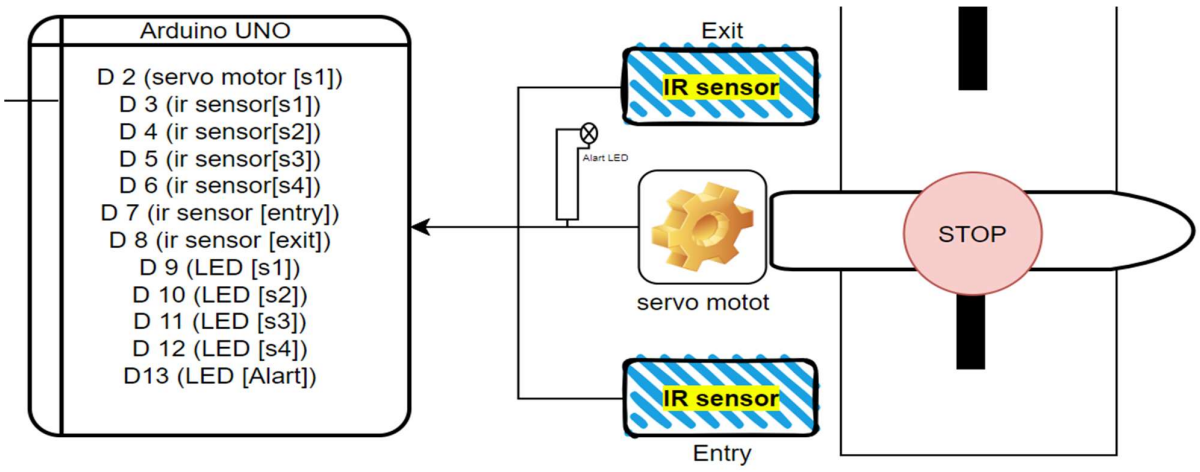

Figure 2.2.2 Vehicle Entry and Exit Monitoring

1. **Entry Point Monitoring**:
    - An IR sensor is placed at the entry point to detect incoming vehicles.
    - When a vehicle is detected, the system checks the availability of parking slots.
    - If at least one slot is available, the system proceeds to open the gate.
2. **Exit Point Monitoring**:
    - An IR sensor is placed at the exit point to detect vehicles leaving the parking area.
    - When a vehicle exits, the system updates the availability status of the parking slots.

2.2.3 Gate Control

1. **Automatic Gate Operation**:
    - A servo motor controls the gate barrier.
    - When an incoming vehicle is detected and parking slots are available, the servo motor is activated to open the gate.
    - Once the vehicle passes through, the gate automatically closes after a short delay.

2. **Full Capacity Management**:
    - If all parking slots are occupied, the system activates a red light indicator at the entrance.
    - The red light signals to incoming drivers that the parking lot is full, and the gate remains closed to prevent entry.

2.2.4 Real-Time Monitoring and Updates

1. **Continuous Monitoring**:
    - The system continuously monitors the status of all IR sensors and updates the occupancy status of parking slots in real time.
    - The Arduino Uno processes the sensor data and updates the light posts and gate control mechanisms accordingly.
2. **Dynamic Slot Availability**:
    - The system dynamically updates the availability status of parking slots as vehicles enter and exit.
    - Real-time updates ensure that the system accurately reflects the current state of the parking lot.

2.2.5 System Alerts and Notifications

1. **Full Parking Alert**:
    - When the parking lot reaches full capacity, the system activates the red light at the entrance to alert incoming drivers.
    - The gate remains closed to prevent additional vehicles from entering until a slot becomes available.
2. **Maintenance Notifications**:
    - The system can be programmed to send alerts or notifications to maintenance personnel if any sensor malfunctions or if there is an issue with the gate mechanism.

2.3 CIRCUIT DIAGRAM

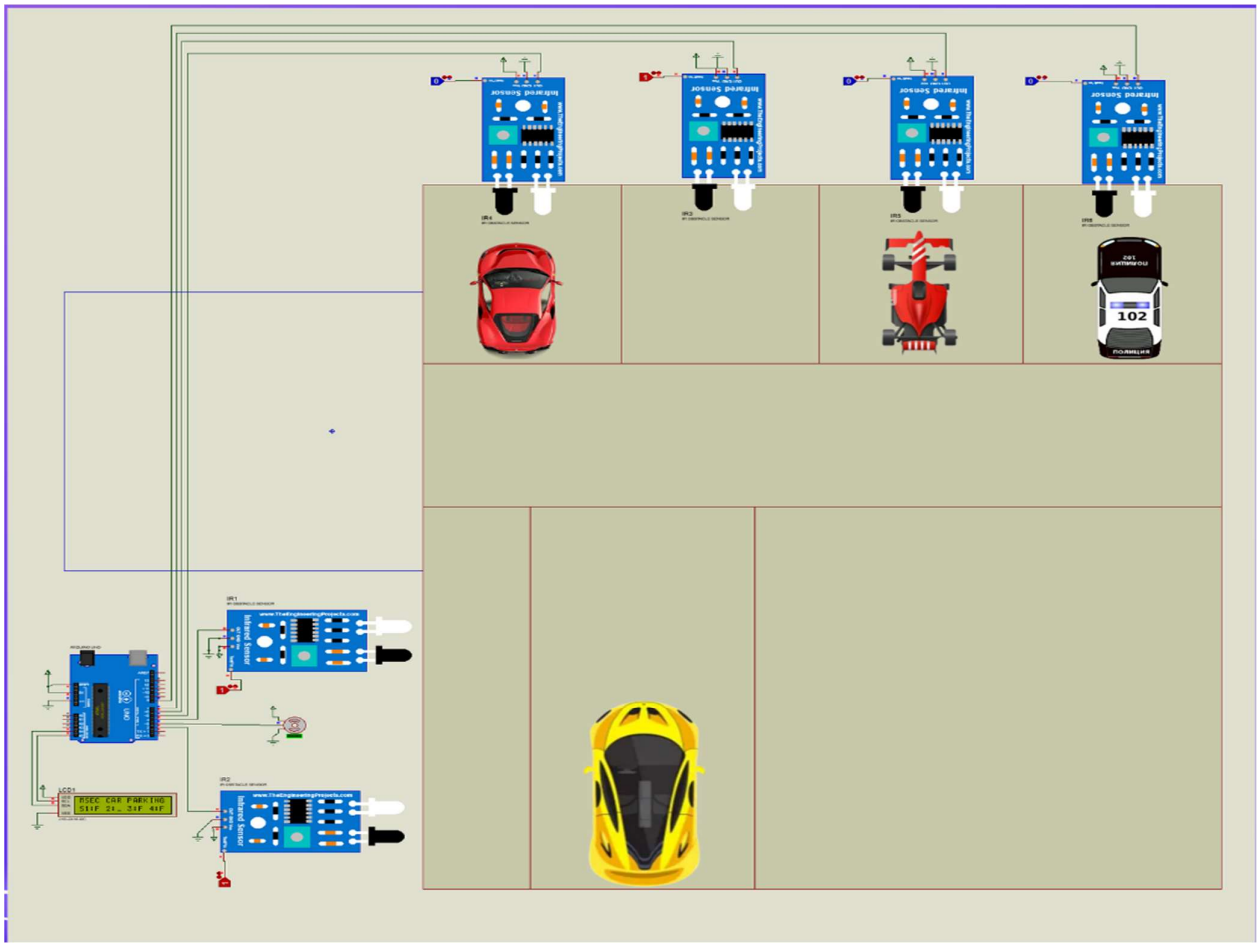

Figure 2.3.1 Circuit Diagram

2.4 HARDWARE REQUIREMENTS

- Arduino Uno
- Jumper wires
- USB Cable
- LED
- IR sensors
- Breadboard
- Servi motor

2.5 SOFTWARE REQUIREMENT

- Arduino IDE Software

2.6 ARDUINO UNO

- Arduino UNO is based on an Atmega328P microcontroller.
- It is easy to use compared to other boards, such as the Arduino Mega board, etc.
- The board consists of digital and analog Input/Output pins (I/O), shields, and other circuits.
- The Arduino UNO includes 6 analog pin inputs, 14 digital pins, a USB connector, a powerjack, and an ICSP (In-Circuit Serial Programming) header.
- It is programmed based on IDE, which stands for Integrated Development Environment.
- It can run on both online and offline platforms.

2.6.1 ARDUINO UNO DIAGRAM

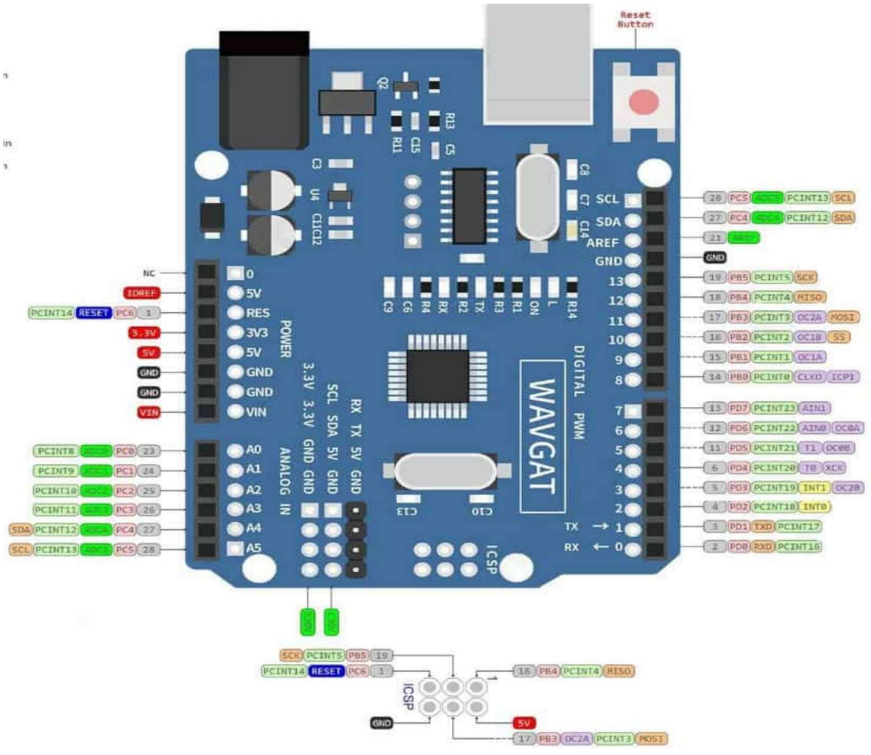

Figure 2.6.1 Arduino Uno R3 Board

- **ATmega328 Microcontroller**- It is a single chip Microcontroller of the ATmel family. Theprocessor code inside it is of 8-bit. It combines **Memory (SRAM, EEPROM, and Flash), Analog to Digital Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator.**
- **ICSP pin**- The In-Circuit Serial Programming pin allows the user to program using thefirmware of the Arduino board.
- **Power LED Indicator**- The ON status of LED shows the power is activated. When thepower is OFF, the LED will not light up.
- **Digital I/O pins**- The digital pins have the value HIGH or LOW. The pins numbered fromD0 to D13 are digital pins.
- **TX and RX LED's**- The successful flow of data is represented by the lighting of theseLED's.
- **AREF-** The Analog Reference (AREF) pin is used to feed a reference voltage to the ArduinoUNO board from the external power supply.
- **Reset button**- It is used to add a Reset button to the connection.
- **USB**- It allows the board to connect to the computer. It is essential for the programming ofthe Arduino UNO board.
- **Crystal Oscillator**- The Crystal oscillator has a frequency of 16MHz, which makes theArduino UNO a powerful board.
- **Voltage Regulator**- The voltage regulator converts the input voltage to 5V.
- **GND**- Ground pins. The ground pin acts as a pin with zero voltage.
- **Vin**- It is the input voltage.
- **Analog Pins**- The pins numbered from A0 to A5 are analog pins. The function

of Analog pins is to read the analog sensor used in the connection. It can also act as GPIO (General Purpose Input Output) pins.

2.7 IR Sensor

An Infrared (IR) sensor is a device that uses infrared radiation to detect objects and measure distances. These sensors are commonly used in various applications such as obstacle detection, distance measurement, and motion detection. IR sensors operate based on the principles of infrared light reflection or interruption.

Working Principle

1. **Emission**:
    - The IR sensor contains an IR LED (Light Emitting Diode) that emits infrared light.
    - The IR LED is continuously or periodically turned on to emit IR light towards the target area.
2. **Detection**:
    - The emitted IR light travels through the air and, if it encounters an object, it is reflected back toward the sensor.
    - The reflected IR light is detected by a photodiode or phototransistor in the sensor.
    - In some IR sensors, the light intensity changes when an object is detected, while in others, the IR beam is interrupted by the object, causing a change in the received signal.
3. **Signal Processing**:
    - The sensor processes the received signal to determine the presence and distance of the object.

- The intensity of the received signal or the interruption of the beam is analyzed to provide output data.

Typical Specifications

- **Range**: Typically ranges from a few centimeters to several meters, depending on the sensor model.
- **Wavelength**: Infrared light is usually emitted at wavelengths around 850 nm to 950 nm.
- **Field of View**: Typically, 30° to 90°, depending on the sensor design and application.
- **Response Time**: Generally very fast, in the order of milliseconds.
- **Operating Voltage**: Commonly 3.3V or 5V.
- **Current Consumption**: Varies from a few milliamps to tens of milliamps, depending on the model.

Applications

- **Obstacle Detection**: Used in robotics and automation to detect obstacles and avoid collisions.
- **Distance Measurement**: Used to measure the distance to an object based on the intensity of the reflected IR light.
- **Motion Detection**: Used in security systems and automatic lighting systems to detect the presence and movement of people or objects.
- **Proximity Sensing**: Used in touchless switches and proximity sensors for detecting the presence of objects without physical contact.

Advantages

- **Non-Contact Measurement**: IR sensors can detect objects and measure distances

without physical contact, making them ideal for applications where contact is not possible or desirable.

- **Fast Response Time**: IR sensors provide quick detection and response, making them suitable for real-time applications.
- **Versatility**: Can be used in a wide range of applications, including industrial automation, consumer electronics, and automotive systems.

Limitations

- **Sensitivity to Ambient Light**: IR sensors can be affected by ambient light conditions, which may interfere with their accuracy.
- **Limited Range**: The detection range of IR sensors is typically shorter than other types of sensors such as ultrasonic or laser sensors.
- **Surface Dependence**: The reflectivity of the object's surface can affect the accuracy of the sensor. Shiny or reflective surfaces may cause inaccuracies in detection.

2.7.1 Diagram of IR Sensor

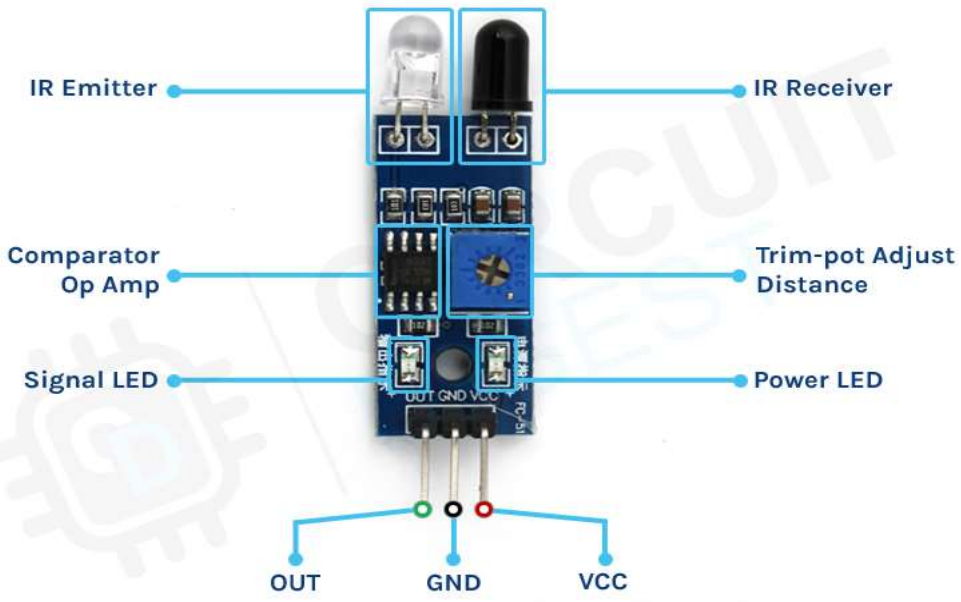

Figure 2.7.1 IR sensor Diagram

2.7.2 Components of IR Sensor

1. **IR LED (Transmitter)**:
    - Emits infrared light.
    - The emitted light is typically modulated to prevent interference from ambient light.
2. **Photodiode/Phototransistor (Receiver)**:
    - Detects the reflected IR light.
    - Converts the received light into an electrical signal.
3. **Signal Processing Circuit**:
    - Processes the electrical signal from the receiver.
    - Determines the presence or distance of the object based on the received signal.
    - Outputs the processed data as an analog or digital signal.
4. **Output Signal**:
    - The processed data is output as a voltage level or digital signal, indicating the presence or distance of the detected object.

2.8 Servo Motors

A servo motor is an electromechanical device used for precise control of angular or linear position, velocity, and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Key Features of Servo Motors

- **Precision Control**: Servo motors provide precise control of position, speed, and acceleration, making them ideal for applications requiring accurate movements.
- **Feedback Mechanism**: They incorporate a feedback system (typically a potentiometer) that constantly adjusts the motor's operation to match the desired position.
- **High Efficiency**: Servo motors are highly efficient, converting electrical energy into mechanical energy with minimal losses.
- **Compact Size**: Despite their power and functionality, servo motors are relatively small and compact, making them suitable for use in tight spaces.

1. Advantages of Servo Motors

1.1. High Torque

- Servo motors can produce high torque relative to their size, making them suitable for demanding applications requiring significant force.

1.2. Accurate Positioning

- The feedback mechanism in servo motors allows for precise control of position, ensuring accurate and repeatable movements.

1.3. Fast Response

- Servo motors can quickly respond to control signals, making them ideal for dynamic applications where rapid changes in position or speed are required.

2.8.1 Servo Motor Diagram

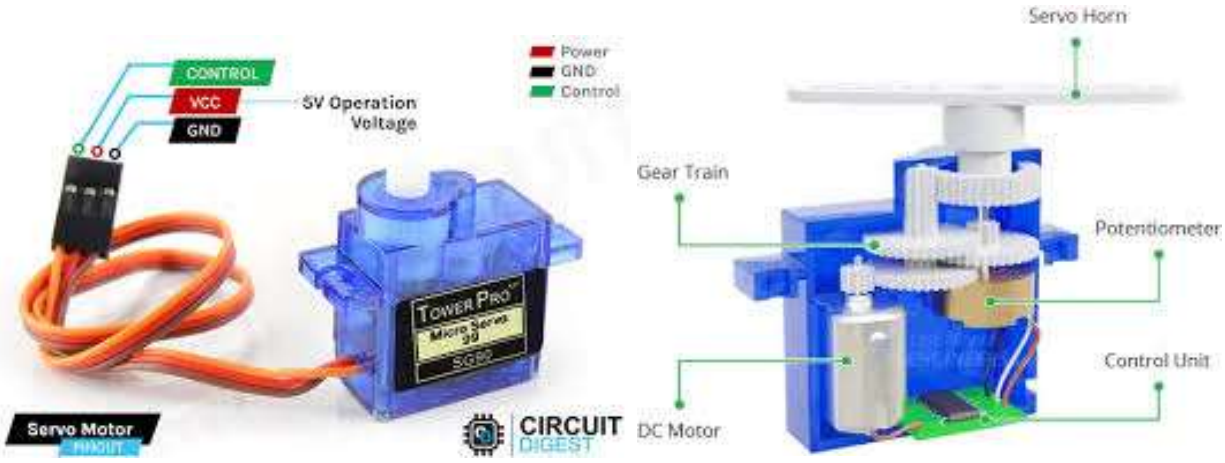

Figure 2.8.1 Servo motor

1. Working

Control Signal

- The position of a servo motor is controlled by sending a Pulse Width Modulation (PWM) signal to its control wire. The PWM signal dictates the position of the servo by varying the width of the pulse.

Feedback Loop

- The servo motor includes a feedback mechanism (usually a potentiometer) that measures the actual position of the motor. This information is sent back to the controller, which adjusts the motor's movement to achieve the desired position.

1.1. Internal Components

- **Motor**: The primary mover within the servo system, usually a DC motor.
- **Control Circuit**: Processes the input PWM signal and the feedback from the potentiometer to control the motor's movement.

Typical Specifications

- **Torque**: The amount of force the motor can exert. Measured in kg.cm or oz.in.
- **Speed**: The rate at which the servo motor can rotate. Measured in seconds per 60 degrees.
- **Voltage**: The operational voltage range, typically 4.8V to 6.0V for most hobby servos.
- **Size and Weight**: Physical dimensions and weight, which vary based on the motor's application.

Applications of Servo Motors

- **Robotics**: Used for precise control of robotic arms and joints.
- **RC Vehicles**: Commonly used in remote-controlled cars, boats, and airplanes for steering and throttle control.
- **Industrial Automation**: Employed in CNC machines, conveyor belts, and other automated systems for precise control.

2.9 LED

- Energy Efficient: LEDs consume less energy compared to traditional light sources like incandescent or fluorescent bulbs.
- Long Lifespan: LEDs can last tens of thousands of hours due to their efficient operation and solid-state design.
- Compact Size: LEDs are typically small in size, which makes them versatile for various applications.
- Variety of Colors: By changing the material used in the semiconductor, LEDs can emit different colors, including visible light and infrared.
- Durability: LEDs are resistant to shock and vibration since they are solid-state devices.

1. Advantages of LEDs

1.1. Durability

- Unlike conventional bulbs, LEDs are more durable due to their **solid-state design**. They are resistant to shock, vibrations, and external impacts.

1.2. Eco-Friendly

- LEDs do not contain harmful substances like **mercury** and are fully recyclable, making them an environmentally friendly lighting option.

1.3. Versatility

- LEDs are available in a wide range of colors and sizes, and they are easily integrated into various systems. They are used in everything from **traffic lights** to **displays** and **automotive lighting**.

2.9.1 LED DIAGRAM

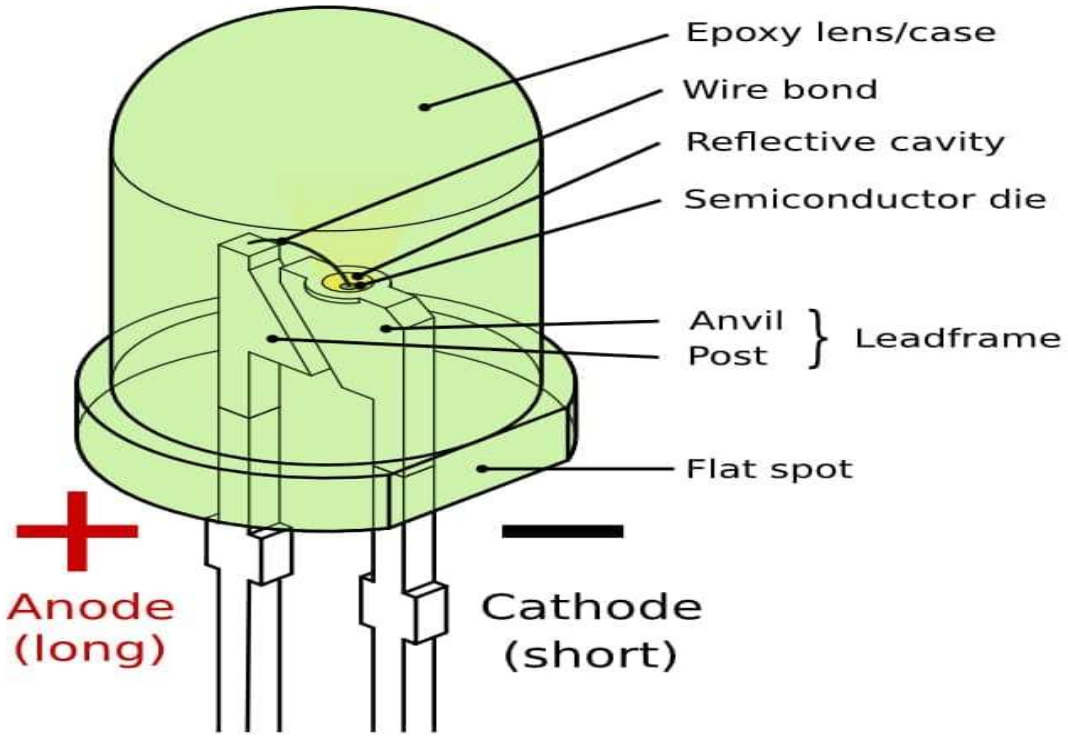

Figure 2.9.1 PWD LED Diagram

1. WORKING

- Duty Cycle: The duty cycle is the percentage of time the signal is high (on) compared to the total time of one cycle. For example, a 50% duty cycle means the LED is on for half of the cycle and off for the other half.

1.1. Electroluminescence Phenomenon

- The core of LED operation is the phenomenon of **electroluminescence**. When a current passes through the diode, electrons recombine with holes, releasing energy in the form of light. The color of the emitted light depends on the energy gap of the semiconductor material used.

1.2. Construction of LEDs

- LEDs are made from **p-n junctions** of semiconductor materials like **gallium arsenide** (GaAs), **gallium phosphide** (GaP), or **indium gallium nitride** (InGaN). The type of material and its composition determine the wavelength and color of the light emitted.

2.10 ARDUINO SOFTWARE

1. Main Components

- Menu Bar: Located at the top, this includes options for File (new, open, save), Edit (cut, copy, paste), Sketch (verify, upload), Tools (board selection, port selection), and Help (documentation, examples).
- Toolbar: This provides quick access to frequently used actions such as verifying code, uploading to the board, and opening the serial monitor.
- Code Editor: The central area where users write their Arduino sketches (programs). It supports syntax highlighting, which helps differentiate between keywords, variables, functions, and comments for better readability.

2. Code Editor Features

- Line Numbers: Displayed on the left side of the editor to help keep track of code lines.
- Auto-Completion: Offers suggestions as you type, helping speed up coding and reducing errors.
- Error Highlighting: Compiles code and highlights any syntax errors or issues, providing feedback before uploading.

3. Output and Serial Monitor

- Output Window: Located at the bottom, this shows messages related to the compilation process, including errors and warnings. It provides feedback after attempting to upload the code to the board.
- Serial Monitor: Accessible via the toolbar, this allows you to view and send data between the Arduino and your computer. It is useful for debugging and monitoring real-time data.

4. Sketch Area

- Setup Function: This function runs once when the program starts. It's used for initializing variables, pin modes, etc.
- Loop Function: This function runs continuously after the setup function. It contains the main code that controls the Arduino's behavior.

5. Library Management

- Library Manager: Accessible through the Sketch menu, it allows users to include and manage libraries, which are collections of code that simplify complex tasks and expand the Arduino’s capabilities.

6. Board and Port Selection

- Board Selector: Located in the Tools menu, this lets you select the specific Arduino board you are using, ensuring that the code is compiled for the correct hardware.
- Port Selector: This allows you to select the correct port for uploading the code to your Arduino board, which is essential for communication between the computer and the board.

3. IMPLEMENTATION

3.1 SOURCE CODE

```
#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
Servo myservo;
#define ir_enter 2
#define ir_back 4
#define ir_car1 5
#define ir_car2 6
#define ir_car3 7
#define ir_car4 8
#define l1 9
#define l2 10
#define l3 11
#define l4 12
#define r 13
int S1 = 0, S2 = 0, S3 = 0, S4 = 0;
int flag1 = 0, flag2 = 0;
int slot = 4;
int full = 0;
void setup() {
  Serial.begin(9600);

  pinMode(ir_car1, INPUT);
```

```
pinMode(ir_car2, INPUT);
pinMode(ir_car3, INPUT);
pinMode(ir_car4, INPUT);
pinMode(ir_enter, INPUT);
pinMode(ir_back, INPUT);
pinMode(l1, OUTPUT);
pinMode(l2, OUTPUT);
pinMode(l3, OUTPUT);
pinMode(l4, OUTPUT);
pinMode(r, OUTPUT);
myservo.attach(3);
myservo.write(90);
lcd.init();
lcd.backlight();
lcd.setCursor(0, 0);
lcd.print(" Hi Welcome To ");
lcd.setCursor(0, 1);
lcd.print("smart car parking project");
delay(4000);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(" Today's Project ");
lcd.setCursor(0, 1);
lcd.print("smart Parking  ");
delay(4000);
lcd.clear();
Read_Sensor();
int total = S1 + S2 + S3 + S4;
```

```
  slot = 4 - total; // Initialize slots based on occupied spaces
}
void loop() {
  Read_Sensor();
  // Display the number of available slots or Parking Full
  if (slot > 0) {
    lcd.setCursor(0, 0);
    lcd.print("Slots: ");
    lcd.print(slot);
    lcd.print("     "); // Clear any previous characters
    digitalWrite(r, LOW);
  } else {
    lcd.setCursor(0, 0);
    lcd.print("Parking Full    ");
    digitalWrite(r, HIGH);
  }
  // Display the status of each slot
  lcd.setCursor(0, 1);
  lcd.print("1:");
  lcd.print(S1 == 1 ? "F" : "E");
  lcd.print(" 2:");
  lcd.print(S2 == 1 ? "F" : "E");
  lcd.print(" 3:");
  lcd.print(S3 == 1 ? "F" : "E");
  lcd.print(" 4:");
  lcd.print(S4 == 1 ? "F" : "E");
  // Handle car entry
  if (digitalRead(ir_enter) == 0 && flag1 == 0) {
```

```
  if (slot > 0) {  // Only open gate if there is a free slot
    flag1 = 1;
    if (flag2 == 0) {
      myservo.write(180);
      digitalWrite(r, LOW);
      slot--;
    }    } else {
    lcd.setCursor(0, 0);
    lcd.print("Parking Full    ");
    digitalWrite(r, HIGH);
  }  }
 // Handle car exit
 if (digitalRead(ir_back) == 0 && flag2 == 0) {
  flag2 = 1;
  if (flag1 == 0) {
    myservo.write(180);
    slot++;
  }  }
 // Close the servo after both entry and exit have been processed
 if (flag1 == 1 && flag2 == 1) {
  delay(1000);
  myservo.write(90);
  flag1 = 0;
  flag2 = 0;
 }  delay(1);
}
void Read_Sensor() {
 S1 = 0; S2 = 0; S3 = 0; S4 = 0;
```

```
if (digitalRead(ir_car1) == 0) {
  S1 = 1;
  digitalWrite(l1, LOW);
} else {
  digitalWrite(l1, HIGH);
}
if (digitalRead(ir_car2) == 0) {
  S2 = 1;
  digitalWrite(l2, LOW);
} else {
  digitalWrite(l2, HIGH);
}
if (digitalRead(ir_car3) == 0) {
  S3 = 1;
  digitalWrite(l3, LOW);
} else {
  digitalWrite(l3, HIGH);
}
if (digitalRead(ir_car4) == 0) {
  S4 = 1;
  digitalWrite(l4, LOW);
} else {
  digitalWrite(l4, HIGH);
}
int total = S1 + S2 + S3 + S4;
slot = 4 - total;
}
```

3.2 RESULTS

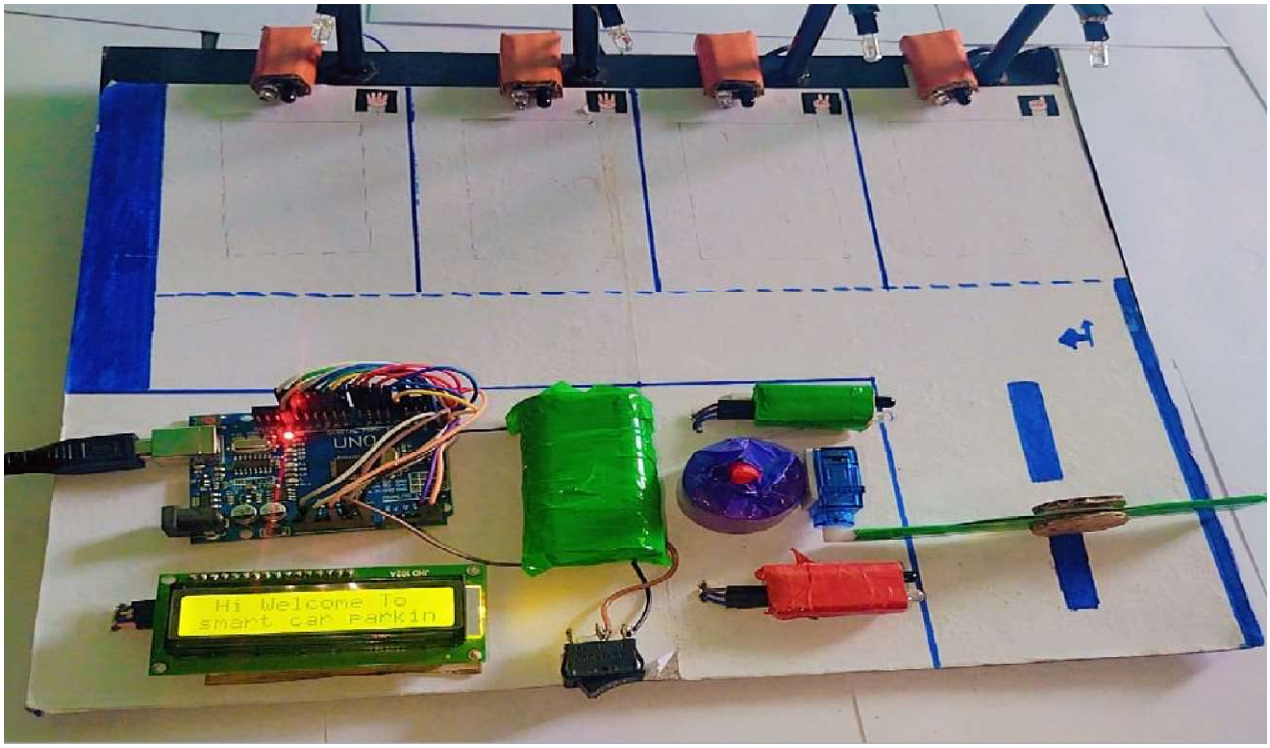

Figure 3.2.1 Initial Stage system Switch On

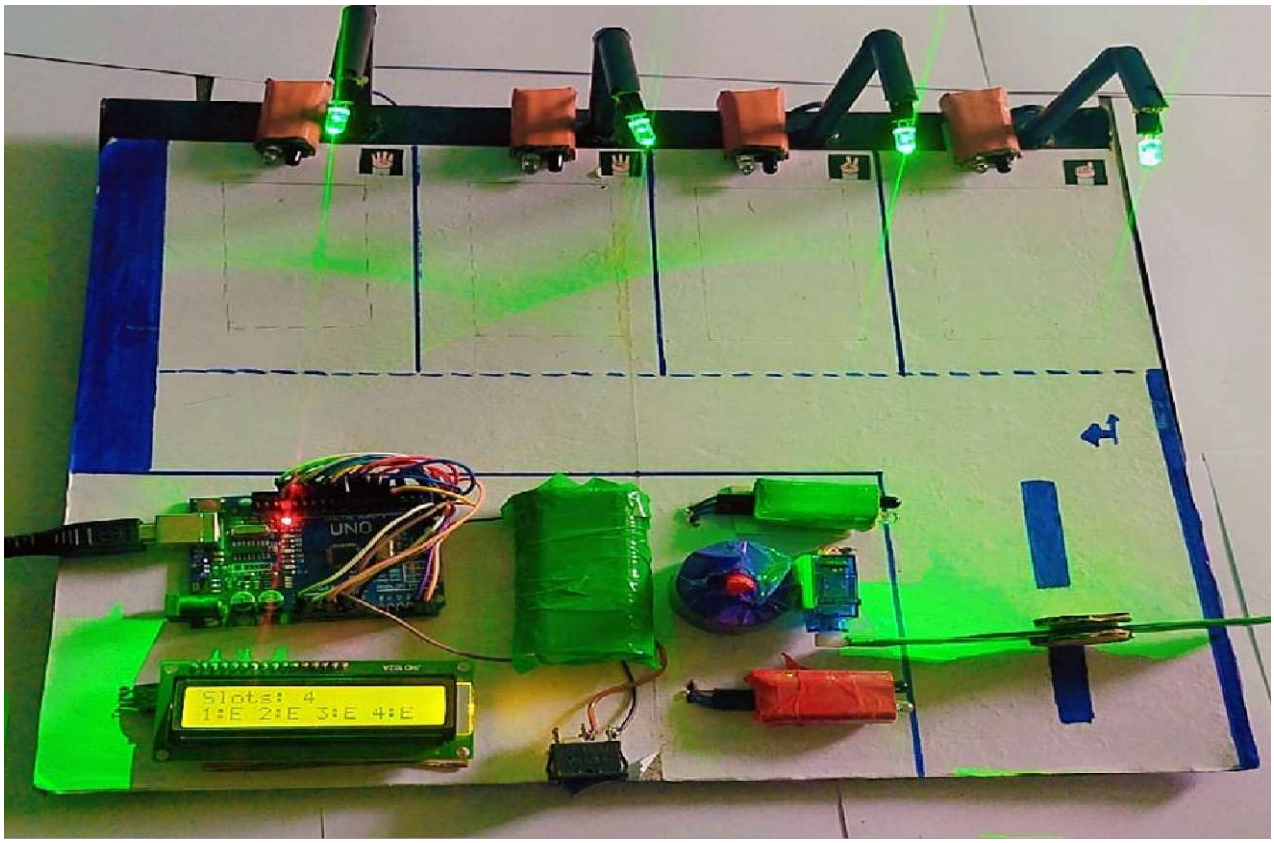

Figure 3.2.2 All Slot Empty

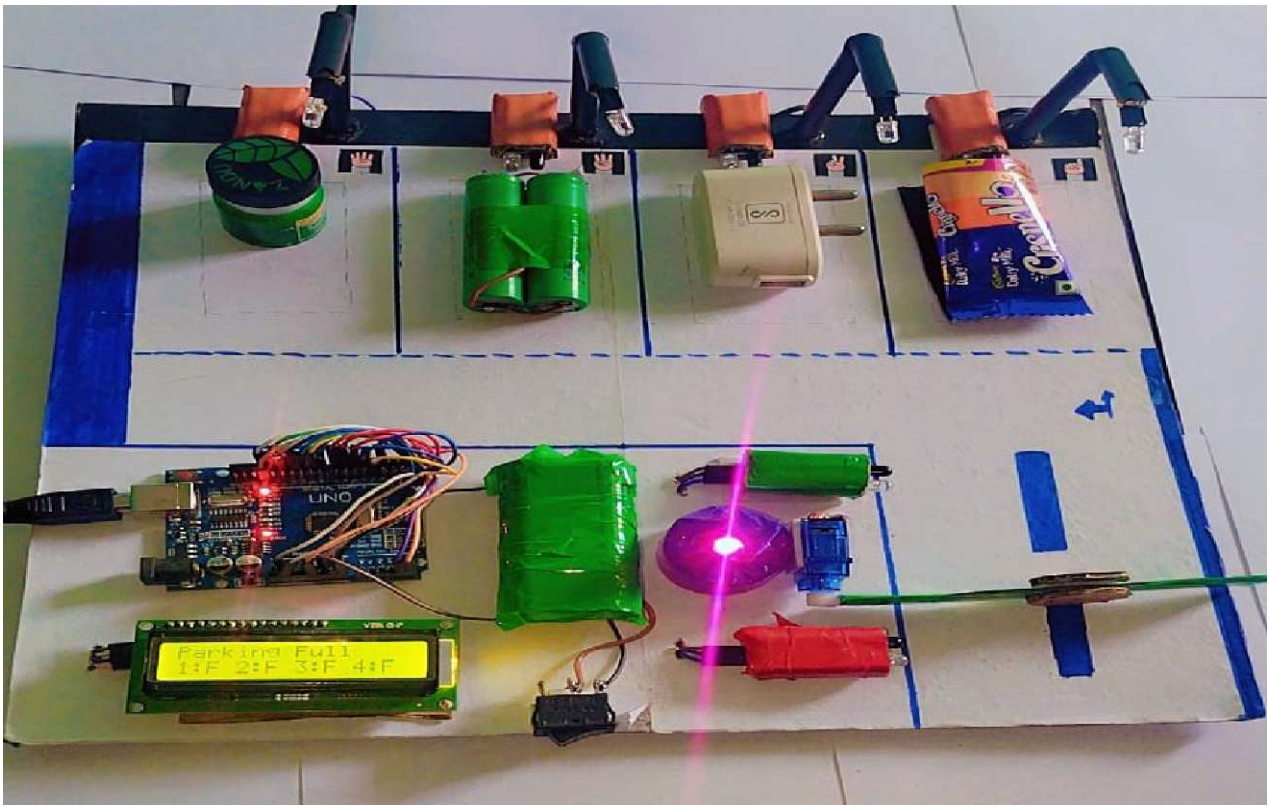

Figure 3.2.3 All Slot Full

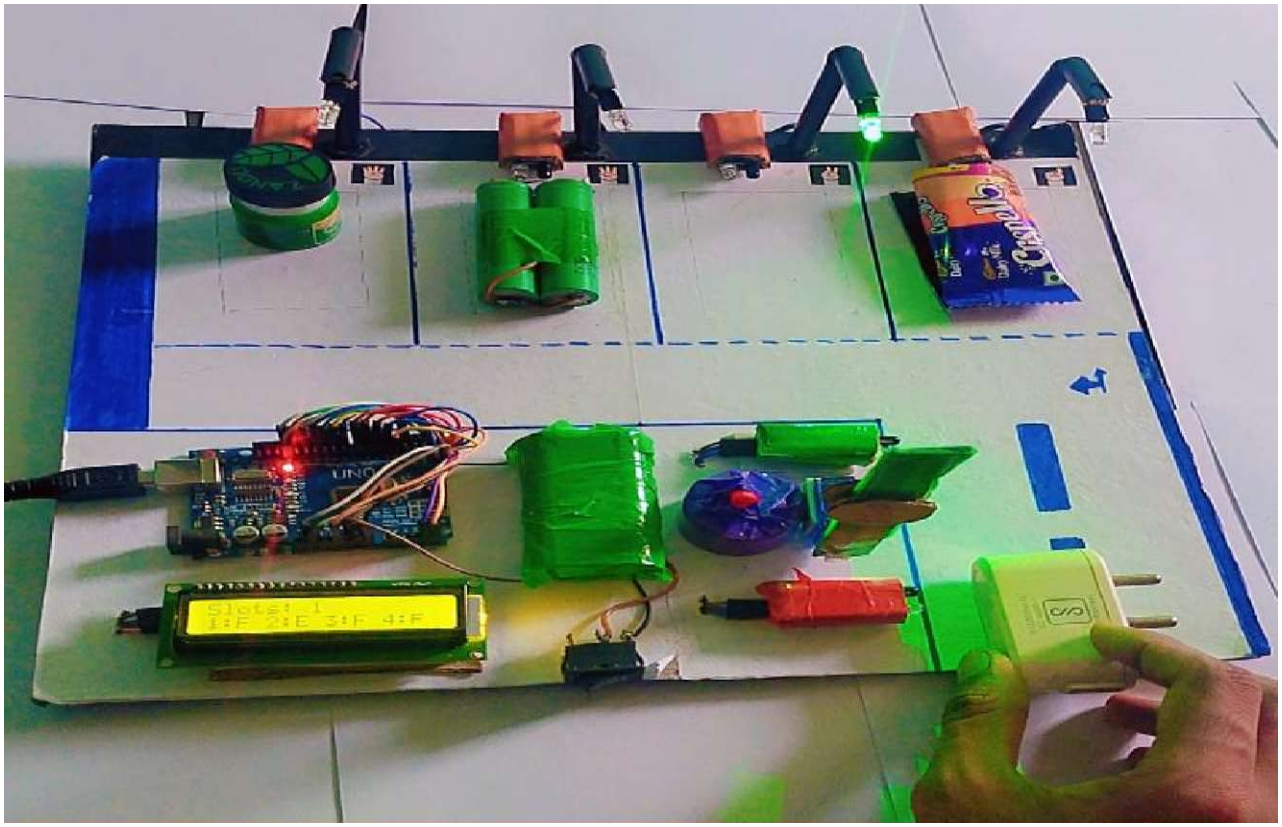

Figure 3.2.4 Car Enter Gate Open

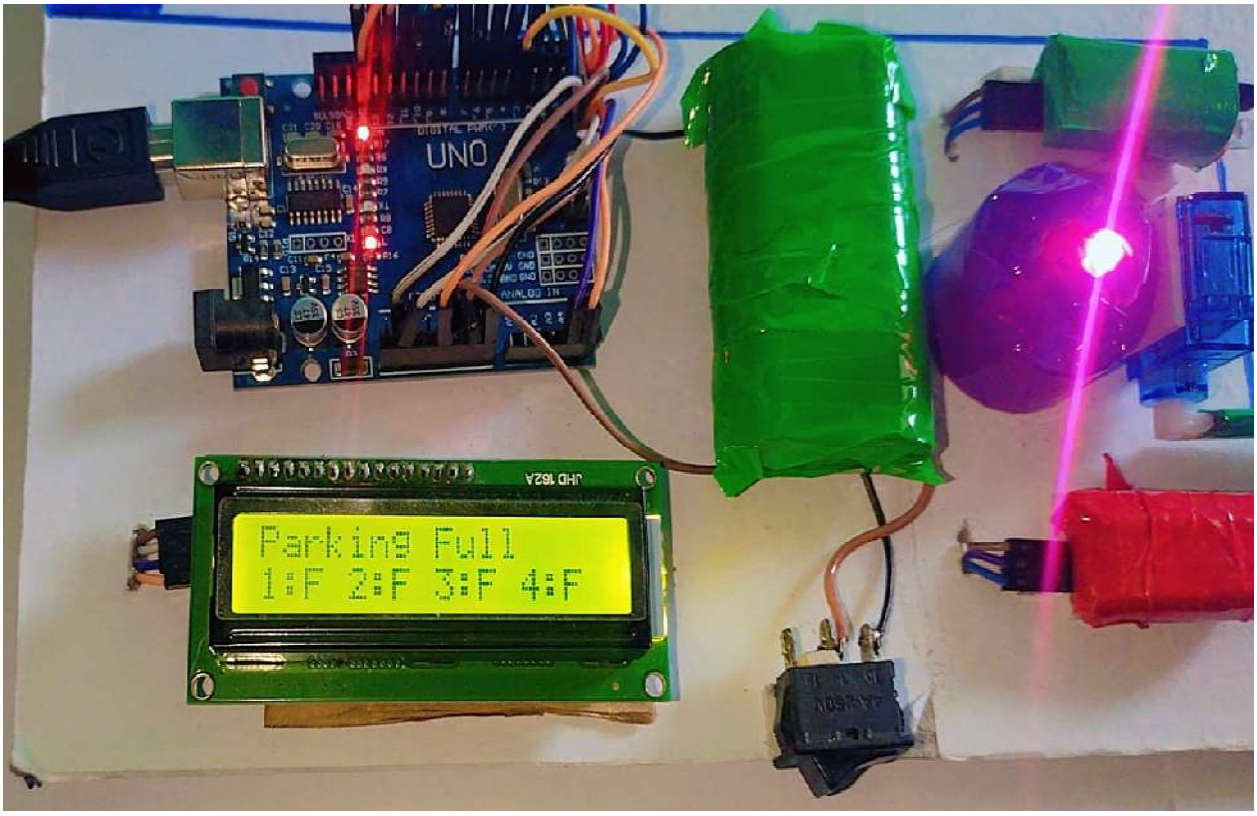

Figure 3.2.4 Slot Full Alert LED On

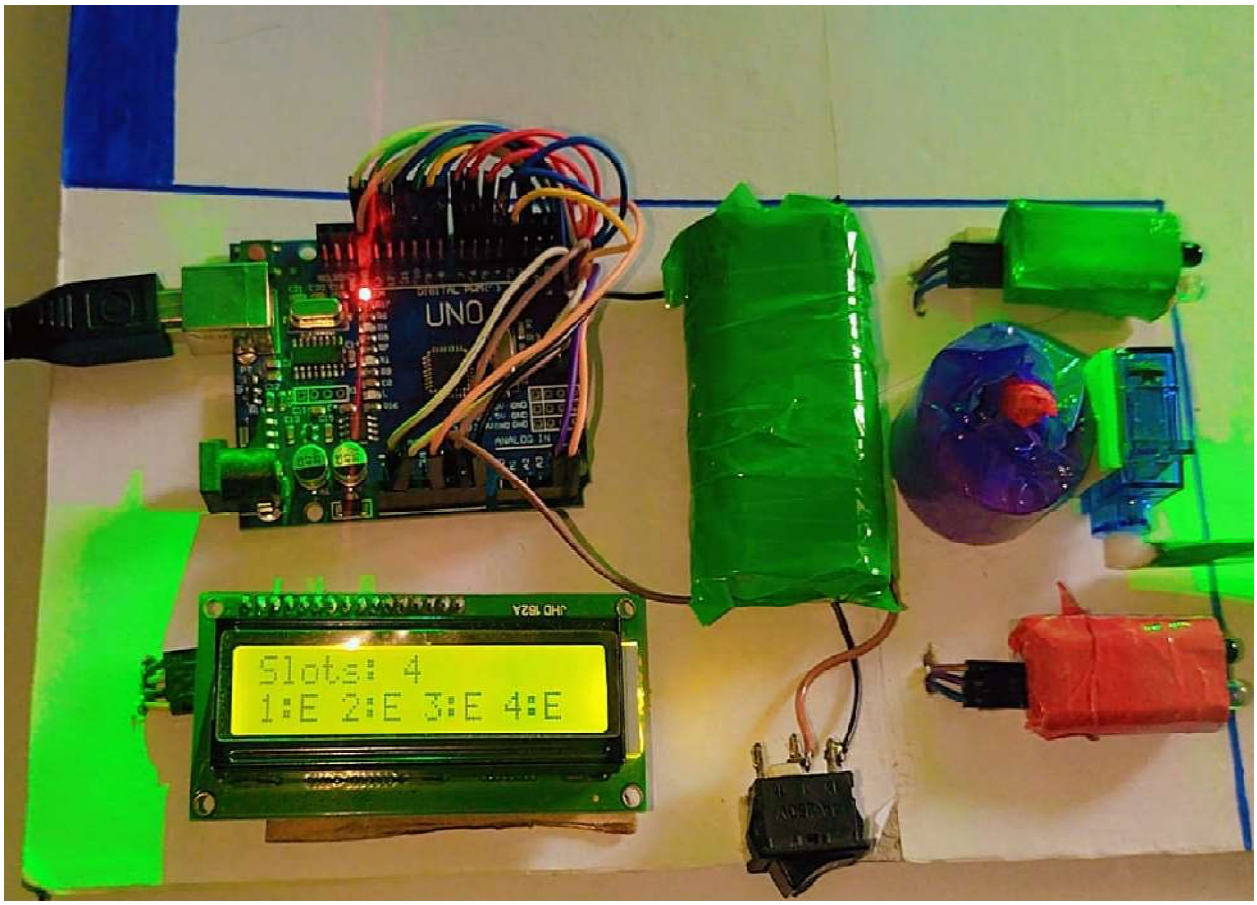

Figure 3.2.3 Lcd Display Show All Slot Empty

4. CONCLUSION

4.1 CONCLUSION

The Smart Car Parking System project demonstrates a practical application of IoT technology to address common challenges associated with urban parking management. By leveraging multiple IR sensors, an Arduino Uno microcontroller, and a servo motor, the system effectively monitors the availability of parking slots, controls access to the parking area, and provides real-time feedback through visual indicators.

Key Outcomes and Benefits

- **Efficient Parking Management**: The system automates the process of monitoring parking slot availability, reducing the need for manual oversight and minimizing the chances of parking errors. This leads to better utilization of available parking space.
- **Real-Time Monitoring and Control**: With the integration of IR sensors and the Arduino Uno, the system offers real-time monitoring of parking slots and dynamic control of the entry gate. This ensures that only available slots are filled, preventing overcrowding and enhancing user convenience.
- **Energy Efficiency**: The use of LEDs to indicate slot availability ensures low energy consumption. The system only activates the servo motor when necessary, further optimizing energy use.
- **Scalability**: The project is designed with scalability in mind, allowing for easy expansion to accommodate more parking slots or additional features, such as remote monitoring and integration with mobile applications.
- **Cost-Effectiveness**: Utilizing readily available components like the Arduino Uno and IR sensors makes the system cost-effective and accessible for various applications, from small-scale parking lots to larger commercial or residential facilities.

Future Enhancements

While the current implementation successfully addresses the primary objectives, there are several areas for future enhancement:

- **Advanced Sensors**: Incorporating advanced sensors such as ultrasonic sensors for more accurate distance measurement and slot detection.
- **Remote Monitoring and Control**: Developing a mobile application or web interface to allow users to check slot availability and reserve spots remotely.
- **Enhanced Security**: Integrating security features such as camera-based monitoring and automated number plate recognition to enhance safety and prevent unauthorized access.
- **Environmental Adaptation**: Adding sensors to monitor environmental conditions like temperature and light, allowing the system to adapt its operations for better performance under varying conditions.

5. REFERENCE

Arduino Uno Documentation:

- Arduino.cc. (n.d.). Arduino Uno Rev3. Retrieved from https://www.arduino.cc/en/Main/ArduinoBoardUno

IR Sensor Technical Specifications and Usage:

- Adafruit Industries. (n.d.). IR Sensor. Retrieved from https://learn.adafruit.com/ir-sensor

Servo Motor Control with Arduino:

- Margolis, M. (2011). Arduino Cookbook. O'Reilly Media, Inc.

Introduction to IoT and Applications:

- McEwen, A., & Cassimally, H. (2013). Designing the Internet of Things. Wiley.

LED Indicators in Smart Systems:

- Banzi, M., & Shiloh, M. (2014). Getting Started with Arduino: The Open Source Electronics Prototyping Platform (3rd ed.). Maker Media, Inc.

Smart Parking Systems Overview:

- Barone, C., Giuffrè, T., Siniscalchi, S. M., Morgano, M. A., & Tesoriere, G. (2014). "A Smart Parking System based on IoT protocols and emerging enabling technologies". In *Applied Sciences*, 4(4), 1576-1595. MDPI. DOI: 10.3390/app4041576

Energy-Efficient Lighting Technologies:

- Tsao, J. Y., Coltrin, M. E., Crawford, M. H., & Simmons, J. A. (2010). "Solid-state lighting: an integrated human factors, technology, and economic perspective". In *Proceedings of the IEEE*, 98(7), 1162-1179. DOI: 10.1109/JPROC.2010.2042430

Real-Time Monitoring and Control in IoT:

- Khan, S., Parkinson, S., & Qin, Y. (2017). "A survey of real-time embedded systems". In *International Journal of Computer Science & Network Solutions*, 5(2).

Smart Car Parking Systems: A Survey:

- Idris, M. Y. I., Tamil, E. M., Noor, N. M., Razak, Z., & Zulkarnain, M. S. (2009). "Parking guidance system utilizing wireless sensor network and ultrasonic sensor". In *Information Technology Journal*, 8(2), 138-146. DOI: 10.3923/itj.2009.138.146

Programming and Controlling Motors with Arduino:

- Blum, J. (2013). Exploring Arduino: Tools and Techniques for Engineering Wizardry. Wiley.