SMART PARKING

Problem definition:

The objective of this project is to design, develop, and deploy an innovative Parking management system that leverages Internet of Things (IoT) technology. The system will optimize parking space utilization, improve user experience, and reduce traffic congestion in urban areas. An IoT-based parking system is a centralized management that enables drivers to search for and reserve a parking spot remotely through their smartphones. It offers a convenient arrangement for drivers to park their cars when they are looking to avoid potential traffic congestion.

Detailed Explanation:

Parking Monitoring:

- Implement IoT sensors in each parking space or at entrance/exit points of parking slots. These sensors detect the presence or absence of vehicles and send this data to a central server or cloud platform.

Occupancy Monitoring:

- Real-time data on parking space occupancy is collected and this information can be used to optimize parking space allocation and pricing strategies.

Parking Guidance:

- Develop a system to guide a drivers to the nearest available parking spaces. This reduces the time and fuel wasted in searching for parking.

Design thinking:

Project objectives:

- To develop a centralized management system based on IOT concepts that allows the drivers to use their smart phone to search for and reserve parking spot.

IoT sensors:

- Various type of IoT sensors used for detect the occupancy status of the parking spaces. The most commonly used sensors are Ultra Sonic sensors and Infra-Red sensors.
- These sensors collect the information about the parking spaces and transmit the data to a central server or a cloud platform via wired or wireless connectivity.

Real Time Display:

- Users can see the real-time notification about parking slot availability and occupancy availability through LCD display.

Innovation:

After thorough research and analysis, we arrived at an innovative solution to solve the above problem as detailed in phase 1 of our project. We implement an smart parking system with help of Arduino Uno that enables effective parking spot usage. We use IR sensors to detect parking space occupancy.

Components:

1. Arduino

The Arduino Uno is an open source microcontroller board based on the Microchip ATmega8p microprocessor. The board includes digital and analogue I/O pins that can be used to connect to expansion board and other circuits.

2. Servo motor

A Servo motor is a linear or rotatory actuator that permits exact control of angular and linear position, velocity and acceleration.

3. IR Sensors

An Infrared (IR) sensor is a type of sensor that detects and measure infrared radiation in the surroundings. It will detect the parking availability in the parking slot.

Workflow:

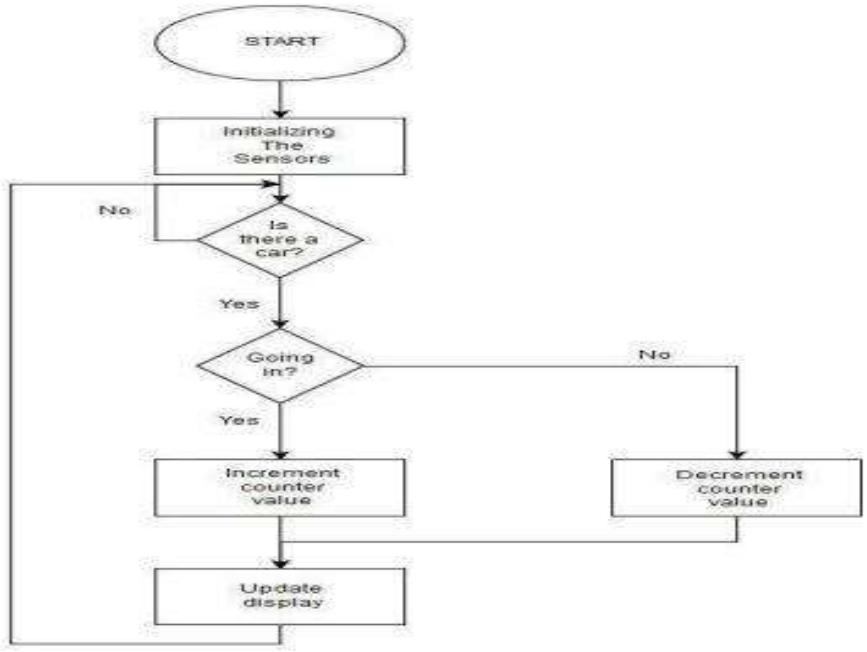
- The Arduino UNO microcontroller is utilized. The infrared sensors are attached to the Arduino's digital pins.

- When a car approaches the entry gate, the entry IR sensor sends a signal to Arduino, which instructs it to turn on the motor. When a car is recognized, the motor acts as a checkpoint or gate, allowing or prohibiting certain actions.

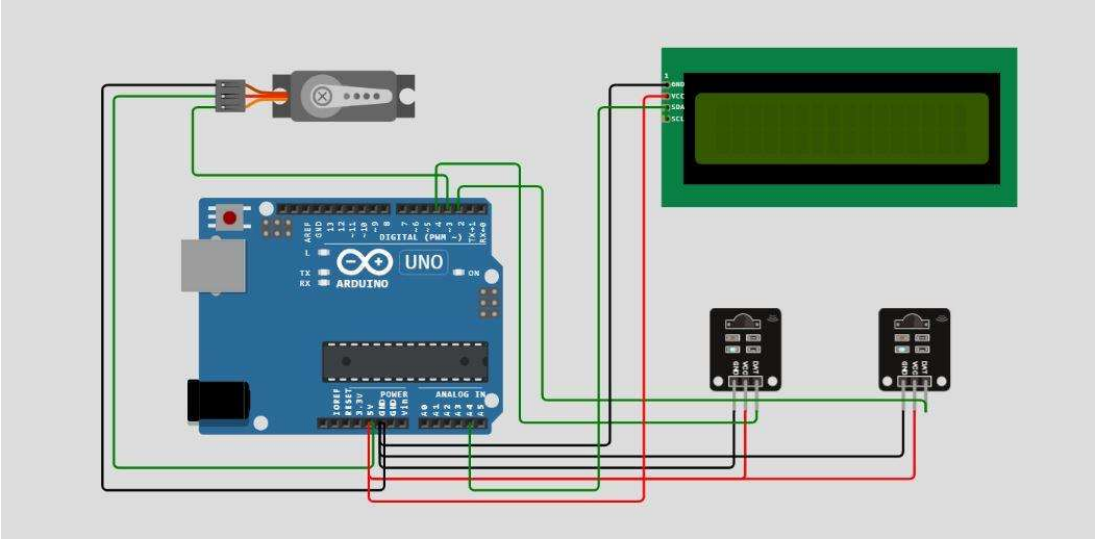
- There are already a certain amount of parking spots accessible. When the car pulls into the parking space, both IR sensors pick it up. If the car is first detected by the IR sensor-1 (located outside the check post), it is entering the parking area, and the total number of parking spaces will be decreased by one unit.

- The total number of parking spaces is increased by one unit if the IR sensor-2 (located within the check post) recognizes the car first. According to the vehicle detection, the motor constantly opens and closes the checkpoint.

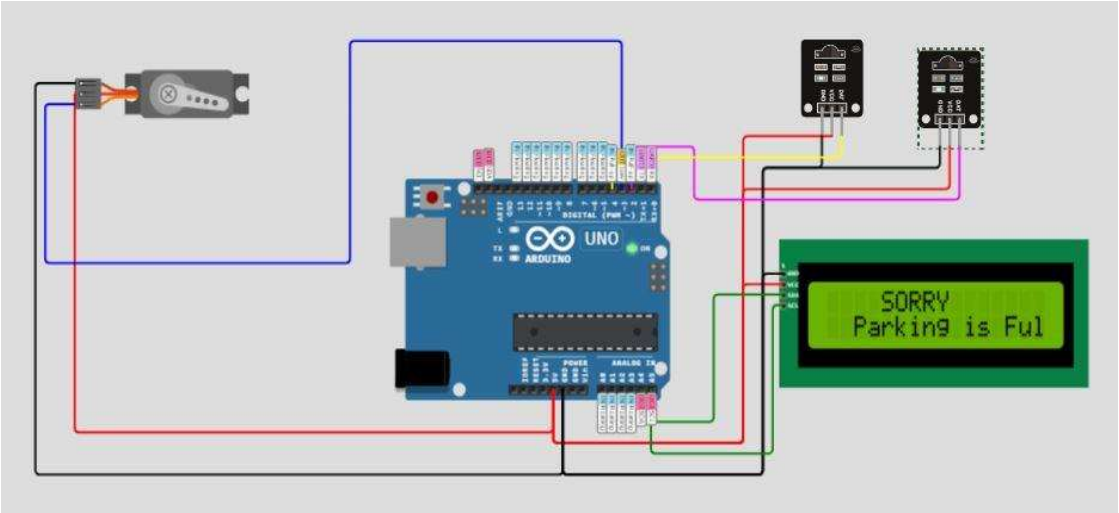
Flowchart:



Software Implementation:
Circuit diagram:



Output:



Source Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Change the HEX address
#include <Servo.h>
Servo myservo1;

int IR1 = 2;          // Parking Entrance
int IR2 = 4;          // Parking Exit
int Slot = 4;        // Enter Total number of parking Slots
bool flag1 = false;
bool flag2 = false;
```

```
unsigned long lastLcdUpdate = 0;  // Variable to track the time of the last LCD update
unsigned long lcdUpdateInterval = 1000;  // Update the LCD every 1000 milliseconds (1 second)


void setup() {
  lcd.begin(16, 2);  // Initialize LCD with 16 columns and 2 rows
  lcd.backlight();
  pinMode(IR1, INPUT);
  pinMode(IR2, INPUT);
  myservo1.attach(3);
  myservo1.write(100);
  lcd.setCursor(0, 0);
  lcd.print("     SMART    ");
  lcd.setCursor(0, 1);
  lcd.print(" PARKING SYSTEM ");
  delay(2000);
  lcd.clear();
  Serial.begin(9600);  // Start serial communication for debugging
}


void loop() {
  if (digitalRead(IR1) == LOW && !flag1) {
    if (Slot > 0) {
      flag1 = true;
```

```
    if (!flag2) {
      myservo1.write(0);
      Slot--;
    }
  } else {
    displayMessage("    SORRY  ", "  Parking is Full  ");
  }
}

if (digitalRead(IR2) == LOW && !flag2) {
  flag2 = true;
  if (!flag1 && Slot < 4) {
    myservo1.write(0);
    Slot++;
  }
}

if (flag1 && flag2) {
  delay(1000);
  myservo1.write(100);
  Serial.println("Servo returned to Initial Position");
  flag1 = false;
  flag2 = false;
}
```

```
  // Update the LCD display with a delay
  if (millis() - lastLcdUpdate >= lcdUpdateInterval) {
    updateLcdDisplay();
    lastLcdUpdate = millis();
  }
}

void updateLcdDisplay() {
  displayMessage("Parking Status", "Slots Left: " + String(Slot));
}

void displayMessage(const char *line1, const String &line2) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(line1);
  lcd.setCursor(0, 1);
  lcd.print(line2);
}
```

Team Members

1. Hariharan S (2021504007)
2. Sanmugavel R (2021504036)
3. Madhavan T (2021504020)
4. Harish Keran E (2021504519)