SMART PARKING

PHASE-5

Team Members

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

Objective:

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 and reduce traffic congestion in urban areas. An IoT-based parking system is that enables drivers to search a parking spot. 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 with help of this data provide the information about the parking slots.

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

Parking Guidance:
- Develop a system to guide a drivers to available parking spaces using IoT devices and sensors. This reduces the time and fuel wasted in searching for parking.

Design thinking:

Project objectives:

- To develop a parking system based on IOT concepts that allows the drivers to know the available parking spaces.

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 microcontroller.

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 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 UNO:

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 used. The infrared sensors are connected to the Arduino's digital pins.
- When a car approaches the entry gate, the entrance 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.
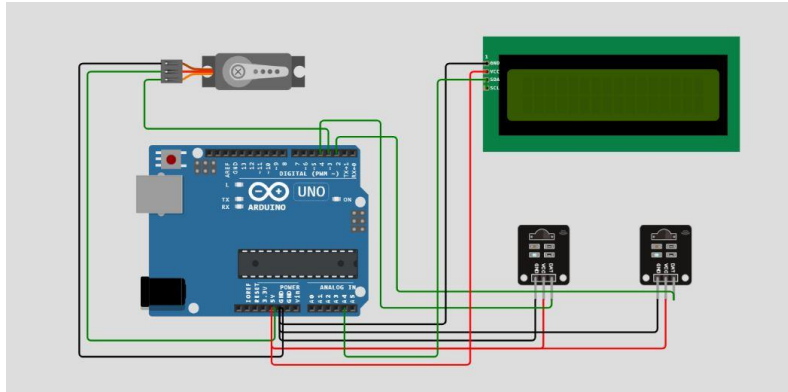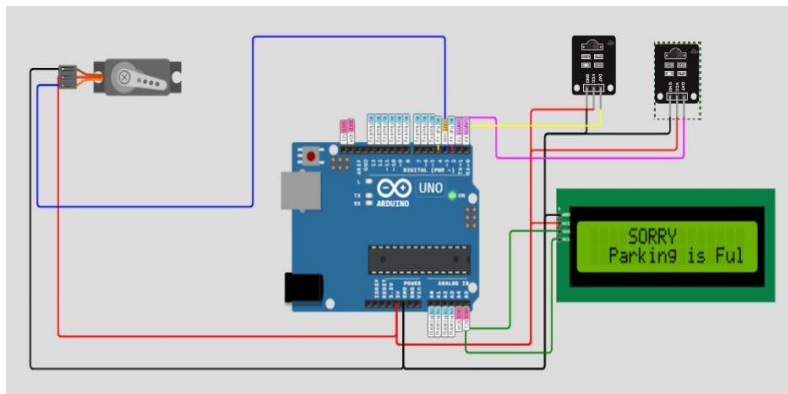
Block Diagram:
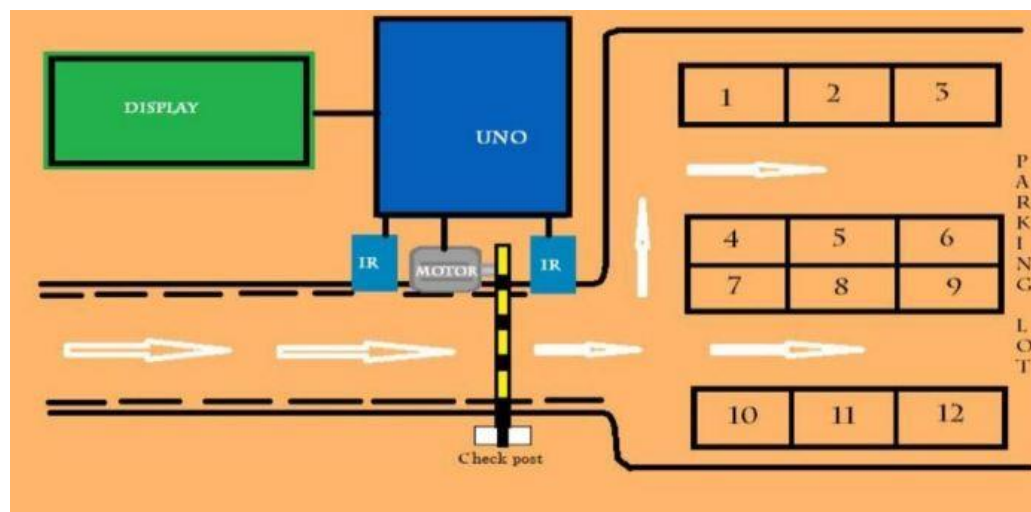
Flowchart:
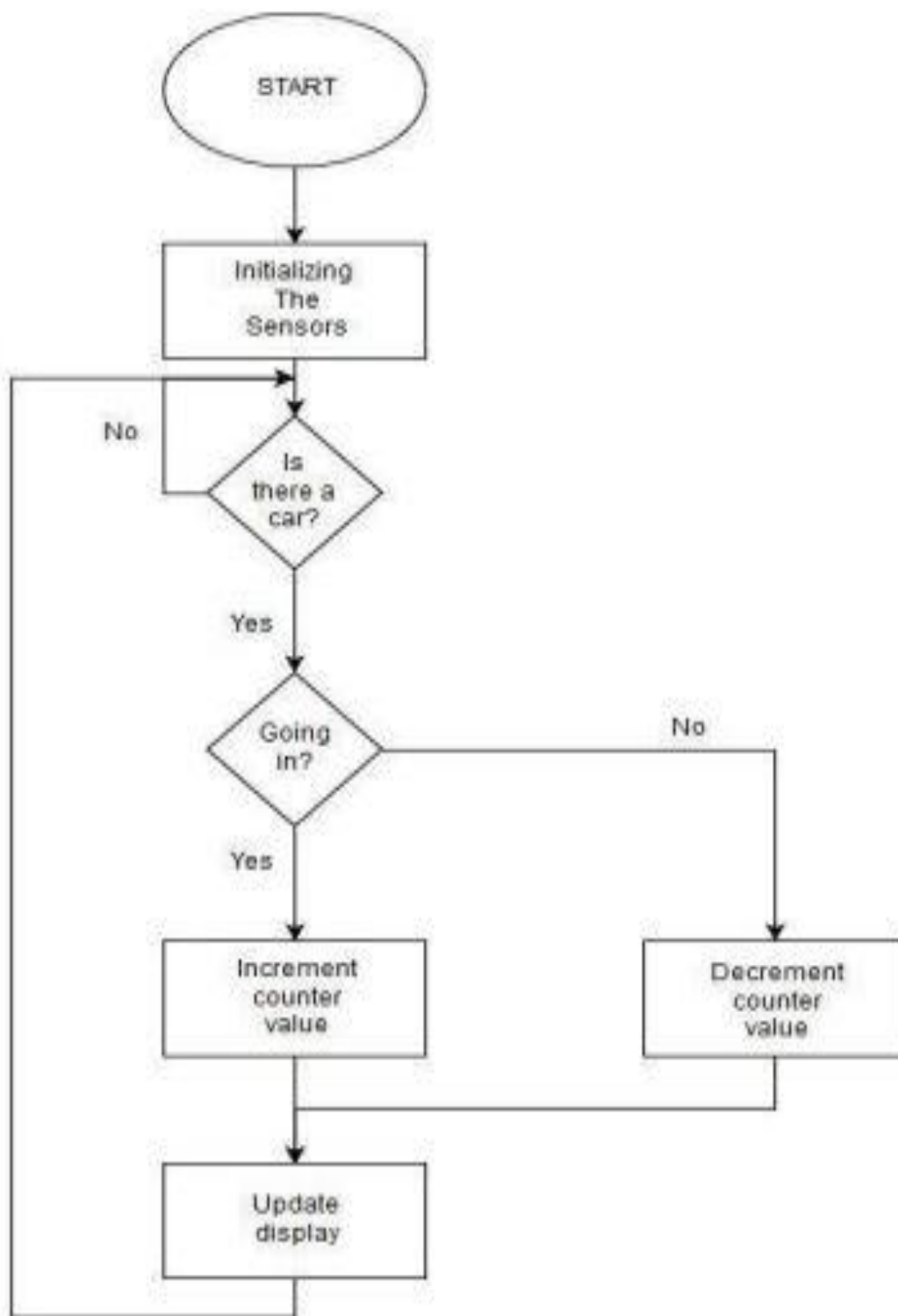
Software Implementation:

Circuit diagram:
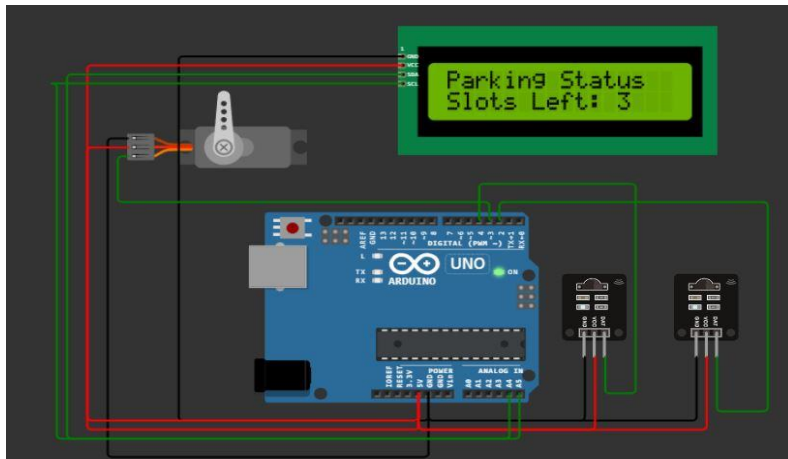

Output:

Output when no slot available:

Output when car entering (slot reduced):

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);
```

```cpp
  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;
```

```arduino
    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);
}
```

Arduino Code:

```
// FOR CAR PARKING SYSTEM
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F,16,2);  //Change the HEX address
#include <Servo.h>
Servo myservo1;
int IR1 = 2;
int IR2 = 4;
int Slot = 4;        //Enter Total number of parking Slots
int flag1 = 0;
int flag2 = 0;

void setup() {
  lcd.begin();
```

```
  lcd.backlight();

pinMode(IR1, INPUT);

pinMode(IR2, INPUT);

myservo1.attach(3);

myservo1.write(100);

lcd.setCursor (0,0);

lcd.print("    ARDUINO    ");

lcd.setCursor (0,1);

lcd.print(" PARKING SYSTEM ");

delay (2000);

lcd.clear();

}

void loop(){

if(digitalRead (IR1) == LOW && flag1==0){

if(Slot>0){flag1=1;

if(flag2==0){myservo1.write(0); Slot = Slot-1;}

}

else{

lcd.setCursor (0,0);

lcd.print("    SORRY :(    ");

lcd.setCursor (0,1);

lcd.print("  Parking Full  ");

delay (3000);

lcd.clear();
```

```
      }

      }

      if(digitalRead (IR2) == LOW && flag2==0){flag2=1;

      if(flag1==0){myservo1.write(0); Slot = Slot+1;}

      }

      if(flag1==1 && flag2==1){

      delay (1000);

      myservo1.write(100);

      flag1=0, flag2=0;

      }

      lcd.setCursor (0,0);

      lcd.print("   WELCOME!   ");

      lcd.setCursor (0,1);

      lcd.print("Slot Left: ");

      lcd.print(Slot);

      }


      // FOR LCD DISPLAY

      #include <Wire.h>

      void setup()

      {

      Wire.begin();

      Serial.begin(9600);
```

```
Serial.println("\nI2C Scanner");

}

void loop()

{

byte error, address;

int Devices;

Serial.println("Scanning...");

Devices = 0;

for(address = 1; address < 127; address++ )

{

Wire.beginTransmission(address);

error = Wire.endTransmission();

if (error == 0)

{

Serial.print("I2C device found at address 0x");

if (address<16)

Serial.print("0");

Serial.print(address,HEX);

Serial.println("  !");

Devices++;

}

else if (error==4)

{

Serial.print("Unknown error at address 0x");
```

```
if (address<16)

Serial.print("0");

Serial.println(address,HEX);

}

}

if (Devices == 0)

Serial.println("No I2C devices found\n");

else

Serial.println("done\n");

delay(5000);

}
```