IoT_Phase5 – Documentation

SMART PARKING SYSTEM USING IoT

Introduction:

A smart parking system using IoT (Internet of Things) is a system that uses sensors and other IoT devices to collect real-time data on parking lot occupancy and transmits this information to the cloud or a local network. This data can then be used to provide drivers with information on available parking spaces, guide them to open spots, and even reserve parking spaces in advance.

Smart parking systems can also be used to optimize parking operations. For example, parking lot managers can use the data collected by IoT sensors to track parking occupancy trends and identify areas where additional parking is needed. They can also use the data to identify and address parking violations.

Components:

The following are the needed components for a smart parking system using IoT:

Hardware:

- **Microcontroller (Arduino):**

  The microcontroller is the brain of the system. It is responsible for collecting data from the sensors and processing it to determine whether a parking space is occupied or vacant. The microcontroller also communicates with the cloud platform.
- **Sensors (IR Sensors):**

  Sensors are used to detect the presence of vehicles in parking spaces.
- **Actuators (Servo motors):**

  It is a part of a device or machine that helps it to achieve physical movements by converting energy, often electrical, air, or hydraulic, into mechanical force. Simply put, it is the component in any machine that enables movement.
- **Communication modules (Wi-Fi, Bluetooth):**

  Communication modules are used to connect the microcontrollers to the cloud. Some common types of communication modules used in smart parking systems include:

  Wi-Fi modules

  Cellular modules
- **Power supply:**

  Batteries in this project provide power at the required voltage for the electronic system that handles sensor input and data transfer to the internet.
- **Liquid crystal displays (LCDs):**

LCD displays can be used to display the number of available parking spaces in a given area, as well as the status of individual parking spaces (occupied or vacant).

- **12c Module:**
  I2C serial interface to the parallel interface required by a character LCD display. This makes it easy to connect an LCD to a microcontroller, such as an Arduino, without having to use a lot of pins.
- **Jumper Wires:**
  Jumper wires are used for making connections between items on your breadboard and your Arduino's header pins.
- **Mounting hardware:**
  Screws
  Bolts

Software:

- **Adafruit IO Platform:**
  It is a cloud server that can be used to connect to IoT devices through wifi and to control these devices through a dashboard. It can be used as a free service and it has got a simple easy-to-use interface to design dashboards.

The specific components need is vary depending on the design of the smart parking system. For example, we design a simple system so it uses only a microcontroller and a few sensors to detect the presence of vehicles in parking spaces.

The following is a basic overview of how a smart parking system using IoT works:

- The microcontroller collects data from the sensors and actuators.
- The microcontroller processes the data and determines the availability of parking spaces.
- The microcontroller sends the data to the cloud platform (optional).
- The mobile app (optional) displays the parking availability data to users.
- Users can use the mobile app to find and reserve parking spaces.

When a vehicle enters the parking lot, the sensors detect its presence and send this information to the microcontroller. The microcontroller then updates the parking availability data in the cloud platform. When a vehicle leaves a parking space, the sensors detect its absence and send this information to the microcontroller. The microcontroller then updates the parking availability data accordingly.

Deployment of Components:

To deploy IR sensors for a smart parking system:

1. Connect the IR sensors to the microcontroller:

Each IR sensor has three wires: power, ground, and signal. Connect the power wire of each IR sensor to a 5V pin on the microcontroller. Connect the ground wire of each IR sensor to a ground pin on the microcontroller. Connect the signal wire of each IR sensor to a digital pin on the microcontroller.

2.Program to read the IR sensor data:

The program read the digital pin for each IR sensor and determine whether it is high or low. A high value indicates that the IR sensor is detecting an object, while a low value indicates that the IR sensor is not detecting an object.

Program:

```
#include <IRremote.h>

const int IR_PIN = 12; // The pin that the IR sensor is connected to

IRrecv irrecv(IR_PIN);

void setup() {

  Serial.begin(9600);

  irrecv.enableIRIn(); // Start receiving IR signals

} void loop() {

  if (irrecv.decode()) { // If an IR signal is received

    Serial.println("Vehicle detected!");

  } }
```

3.Determine to detecting vehicle:.

You can use the IR sensor data to determine when a vehicle enters or exits a parking space.

To deploy LCD for a smart parking system:

1.Choose the right LCD:

There are many different types of LCDs available, so we are choose one that is compatible with our microcontroller and that meets our needs in terms of size, resolution, and other features.

2.Connect the LCD to the microcontroller:

The LCD will typically be connected to the microcontroller using I2C.

3.Program to display the parking lot data on the LCD:

This program read the parking lot data from the central server and display it on the LCD in a readable format.

Program:

```
#include <stdio.h>

#include <stdlib.h>

#include <wiringPi.h>

#define LCD_I2C_ADDRESS 0x27

Void write_char_to_lcd(char c) {

  wiringPiI2CWriteReg8(LCD_I2C_ADDRESS, 0x00, c); }

Void write_string_to_lcd(char *string) {

  While (*string != '\0') {

    Write_char_to_lcd(*string);

    String++;  } }

Int main() {

  wiringPiSetup();

  // Initialize the I2C interface

  wiringPiI2CInit(LCD_I2C_ADDRESS);

  // Clear the LCD

  wiringPiI2CWriteReg8(LCD_I2C_ADDRESS, 0x00, 0x01);

  // Write a string to the LCD

  Write_string_to_lcd("Welcome to the smart parking system!");

  // Wait for 5 seconds

  Delay(5000);

  // Clear the LCD

  wiringPiI2CWriteReg8(LCD_I2C_ADDRESS, 0x00, 0x01);

  // Write another string to the LCD

  Write_string_to_lcd("There are 10 available parking spaces.");

  // Wait for 5 seconds

  Delay(5000);

  // Clear the LCD

  wiringPiI2CWriteReg8(LCD_I2C_ADDRESS, 0x00, 0x01);

  // Write a final string to the LCD

  Write_string_to_lcd("Thank you for using the smart parking system!");

  // Wait for 5 seconds

  Delay(5000);

  Return 0;

}
```

To Use the AdaFruit IO platform for smart parking system:

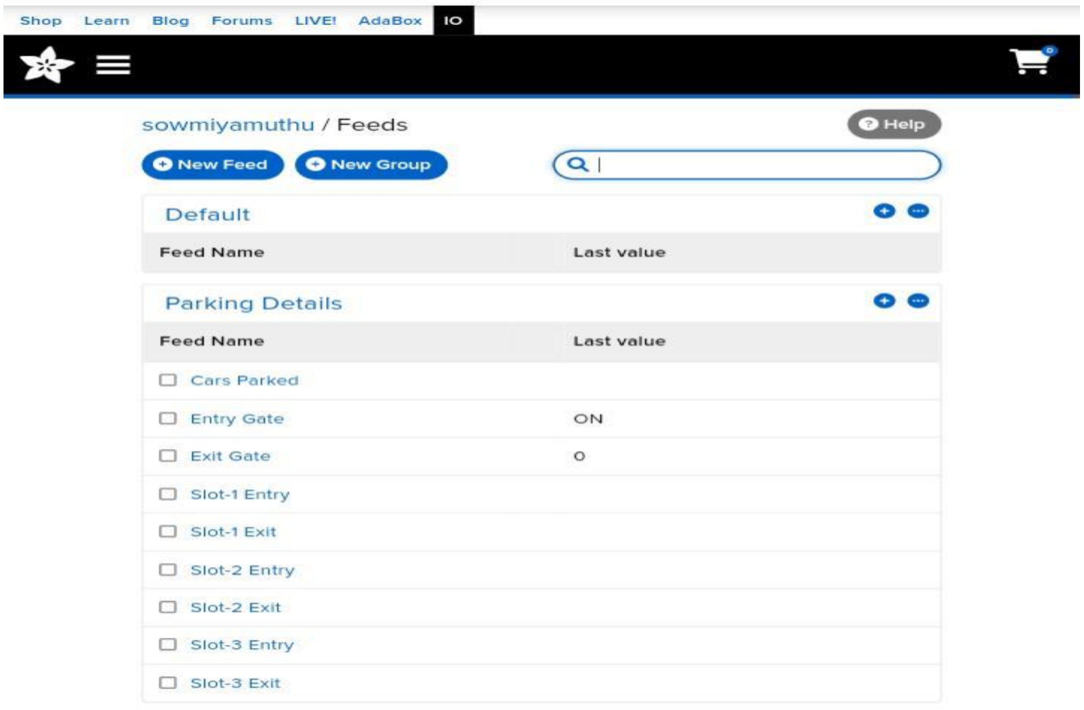
Instead of creating a mobile application or web applications as a beginner I choose the Adafruit IO platform to display the parking availability datas on remotely.

1.Connect an IR sensor to an Arduino microcontroller.

2.Program to read the IR sensor and send the data to Adafruit IO.

Program:

```
#include <WiFi.h>

#include <Adafruit_IO.h>

// Replace the following with your AdaFruit IO username and key

Const char* AIO_USERNAME = “YOUR_ADAFRUIT_IO_USERNAME”;

Const char* AIO_KEY = “YOUR_ADAFRUIT_IO_KEY”;

// Replace the following with the feed keys for your parking slot feeds

Const char* PARKING_SLOT_1_FEED_KEY = “YOUR_PARKING_SLOT_1_FEED_KEY”;

Const char* PARKING_SLOT_2_FEED_KEY = “YOUR_PARKING_SLOT_2_FEED_KEY”;

Const char* PARKING_SLOT_3_FEED_KEY = “YOUR_PARKING_SLOT_3_FEED_KEY”;

// Create an Adafruit_IO object

Adafruit_IO_Class aio(AIO_USERNAME, AIO_KEY);

// Set up the IR sensors

Const int IR_SENSOR_1_PIN = D1;

Const int IR_SENSOR_2_PIN = D2;

Const int IR_SENSOR_3_PIN = D3;

Void setup() {

  // Connect to WiFi

  Serial.begin(115200);

  While (!Serial);

  Serial.println(“Connecting to WiFi”);

  WiFi.begin(“YOUR_WIFI_SSID”, “YOUR_WIFI_PASSWORD”);

  While (WiFi.status() != WL_CONNECTED) {

    Delay(500);
Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  // Connect to Adafruit IO
  Aio.connect();
}
Void loop() {
  // Read the IR sensors
  Int irSensor1Value = digitalRead(IR_SENSOR_1_PIN);
  Int irSensor2Value = digitalRead(IR_SENSOR_2_PIN);
  Int irSensor3Value = digitalRead(IR_SENSOR_3_PIN);
  // Publish the parking slot availability to Adafruit IO
  Aio.publish(PARKING_SLOT_1_FEED_KEY, irSensor1Value);
  Aio.publish(PARKING_SLOT_2_FEED_KEY, irSensor2Value);
  Aio.publish(PARKING_SLOT_3_FEED_KEY, irSensor3Value);
  // Wait for 1 second
  Delay(1000);
}
```



sowmiyamuthu / Feeds

Default

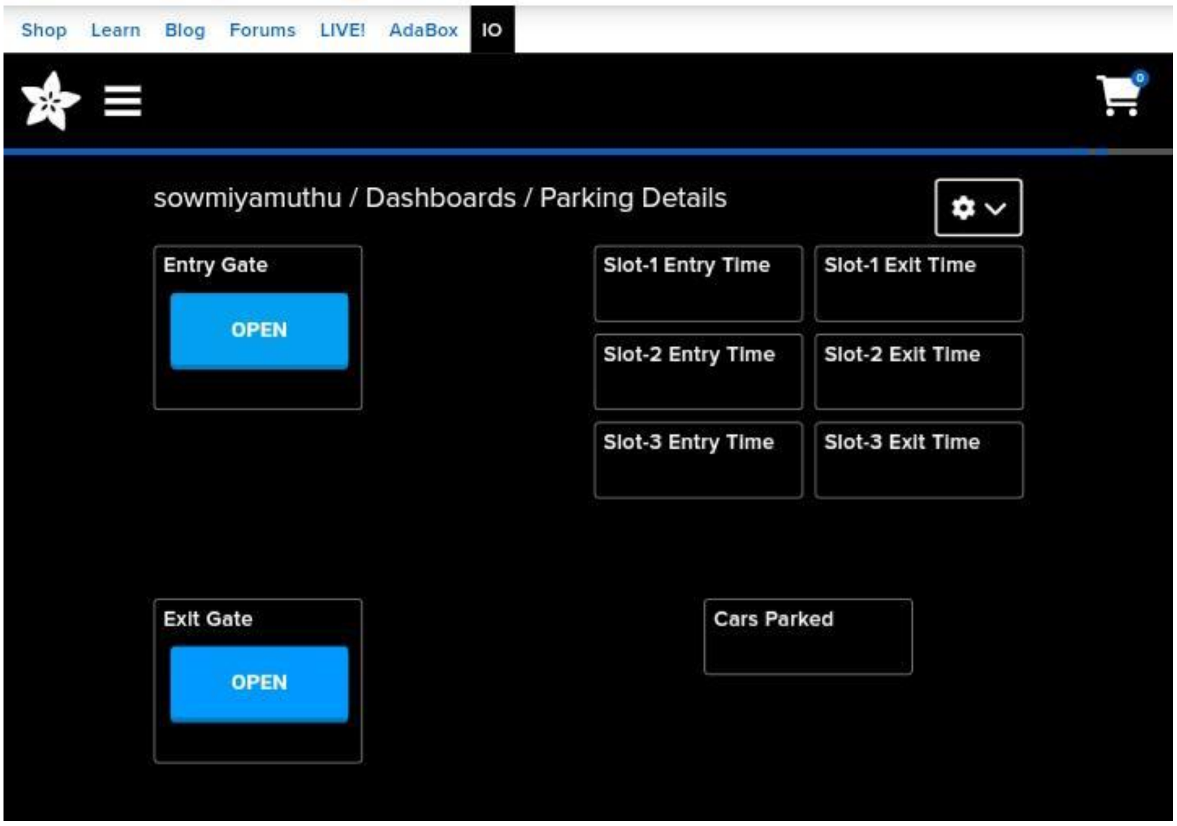
Feed Name	Last value

Parking Details

Feed Name	Last value
Cars Parked	
Entry Gate	ON
Exit Gate	0
Slot-1 Entry	
Slot-1 Exit	
Slot-2 Entry	
Slot-2 Exit	
Slot-3 Entry	
Slot-3 Exit	

3.Create an Adafruit IO dashboard to display the parking space occupancy data.

4.Set up an alert to notify you when a parking space becomes available.



sowmiyamuthu / Dashboards / Parking Details

Entry Gate

OPEN

Slot-1 Entry Time

Slot-1 Exit Time

Slot-2 Entry Time

Slot-2 Exit Time

Slot-3 Entry Time

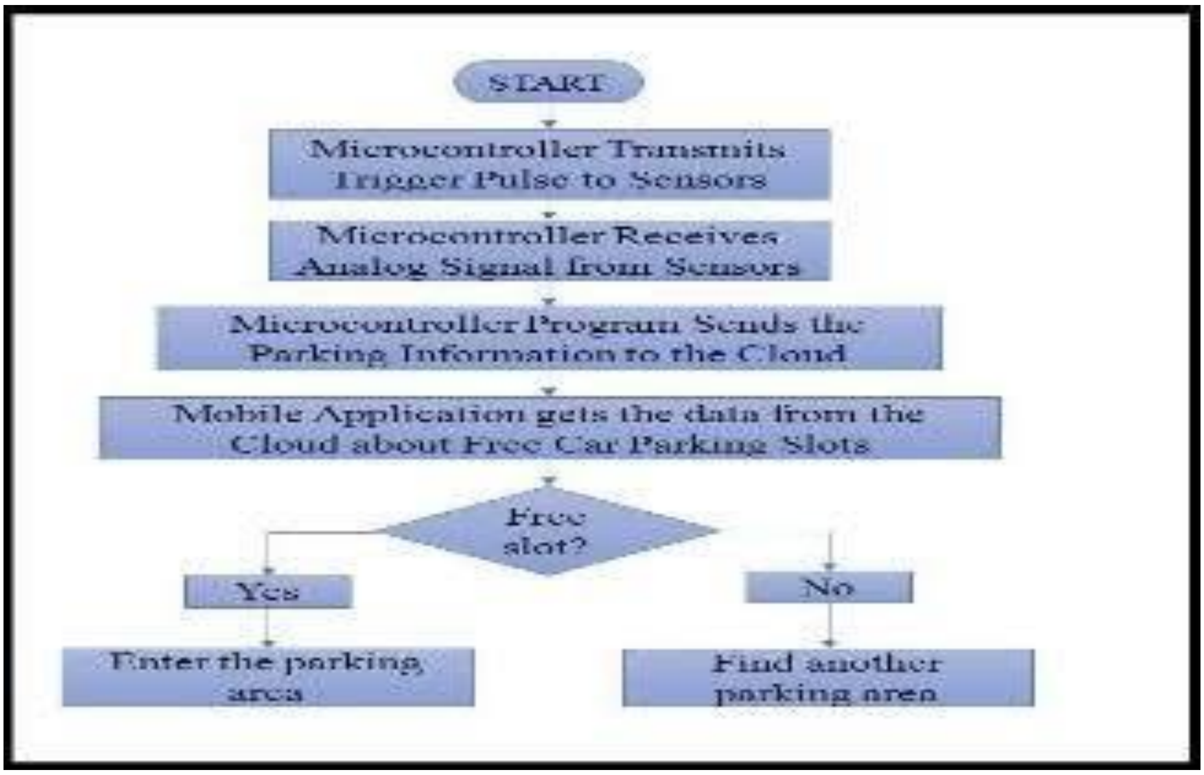
Slot-3 Exit Time

Exit Gate

OPEN

Cars Parked

Flowchart:



START
Microcontroller Transmits Trigger Pulse to Sensors
Microcontroller Receives Analog Signal from Sensors
Microcontroller Program Sends the Parking Information to the Cloud
Mobile Application gets the data from the Cloud about Free Car Parking Slots
Free slot?
Yes
No
Enter the parking area
Find another parking area

Coding For Arduino (Integration on Codes) :

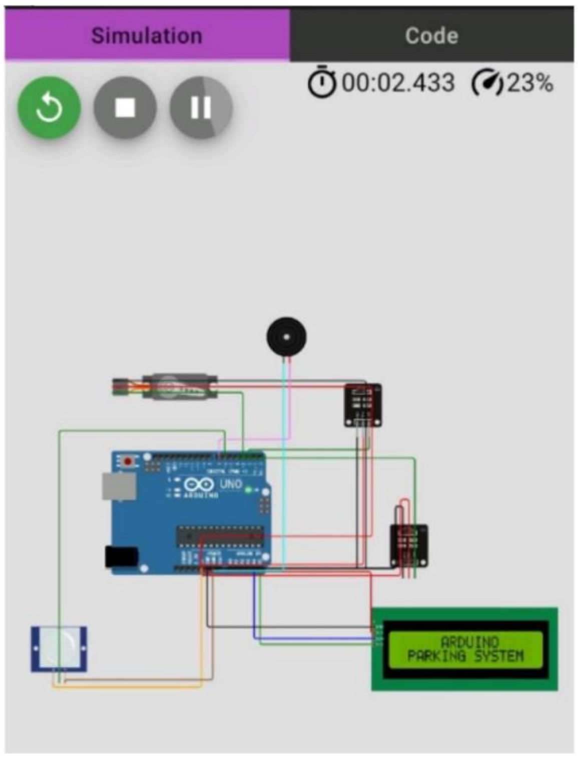
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Change the HEX address
#include <Servo.h>
Servo myservo1;
int IR1 = 2;
int IR2 = 4;
int SmokeDetectorPin = 6;  // Digital pin for the smoke detector
int BuzzerPin = 7;         // Digital pin for the buzzer
int 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);
  pinMode(SmokeDetectorPin, INPUT);
  pinMode(BuzzerPin, OUTPUT);
  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();
  Serial.begin(9600);  // Start serial communication for debugging
}
```

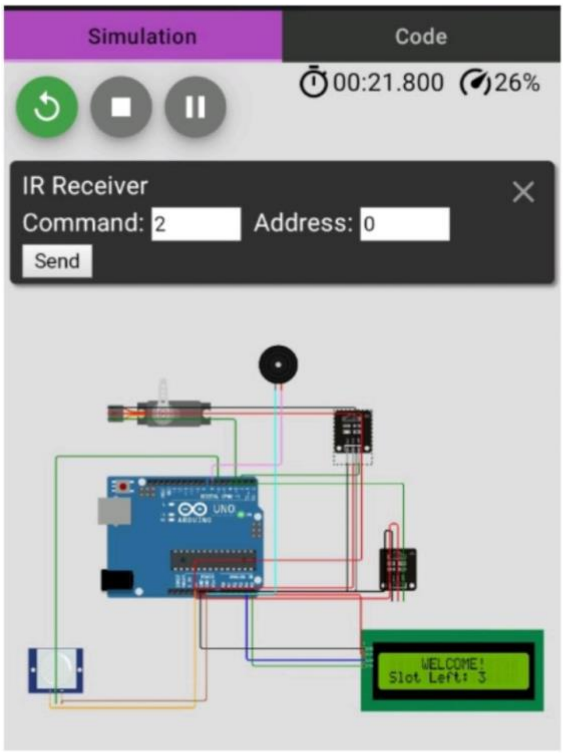
```
void loop() {
 If (digitalRead(IR1) == LOW && !flag1) {
  If (Slot > 0) {
   Flag1 = true;
   If (!flag2) {
    Myservo1.write(0);
    Slot--;}
  } else {
   displayMessage("    SORRY ☹    ", "  Parking Full  ");
  } }
 If (digitalRead(IR2) == LOW && !flag2) {
  Flag2 = true;
  If (!flag1) {
   Myservo1.write(0);
   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() {
 If (digitalRead(SmokeDetectorPin) == HIGH) {
  displayMessage("   WARNING!   ", " Smoke Detected ");
  digitalWrite(BuzzerPin, HIGH);  // Turn on the buzzer
 } else {
```

```
displayMessage("    WELCOME!    ", "Slot Left: " + String(Slot));
    digitalWrite(BuzzerPin, LOW);   // Turn off the buzzer
  } }
Void displayMessage(const char *line1, const String &line2) {
  Lcd.clear();
  Lcd.setCursor(0, 0);
  Lcd.print(line1);
  Lcd.setCursor(0, 1);
  Lcd.print(line2);
}
```

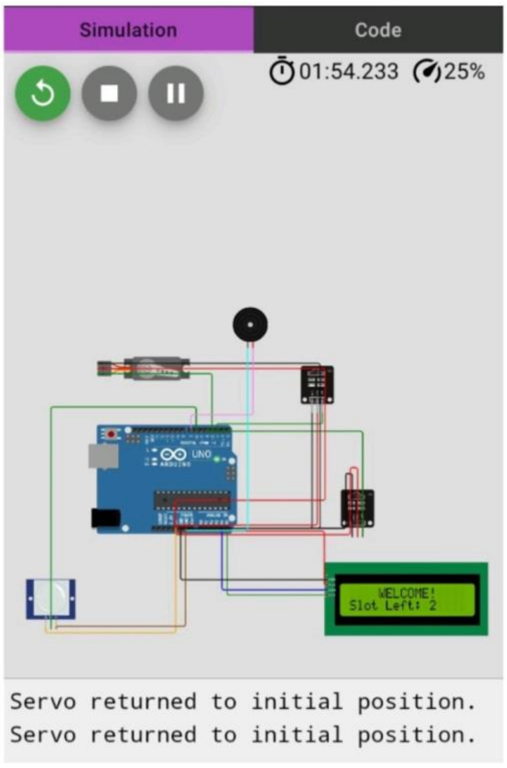
Simulation (on WOKWI)

Since we had trouble using Wokwi Simulator in the PC web browser, we have implemented the project in the mobile web browser.

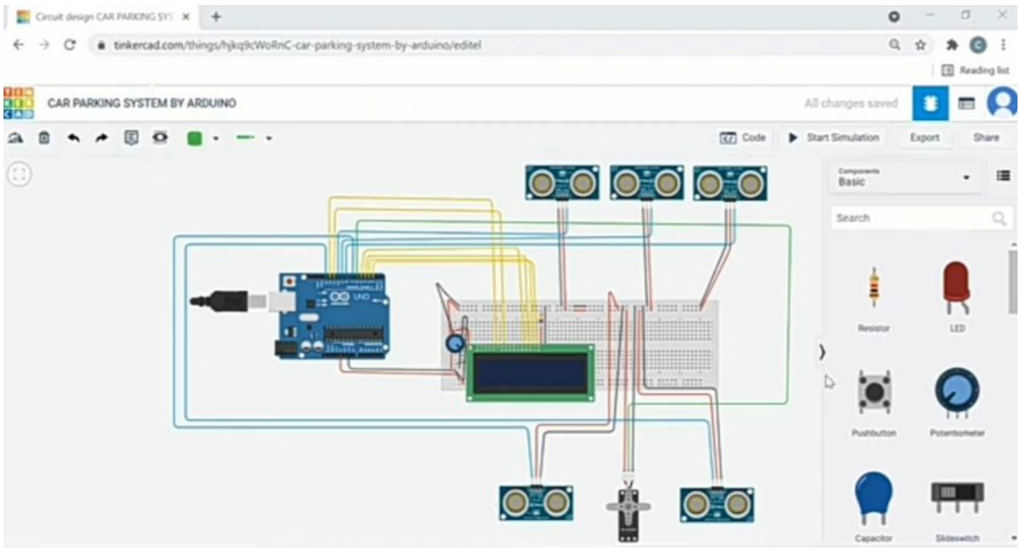


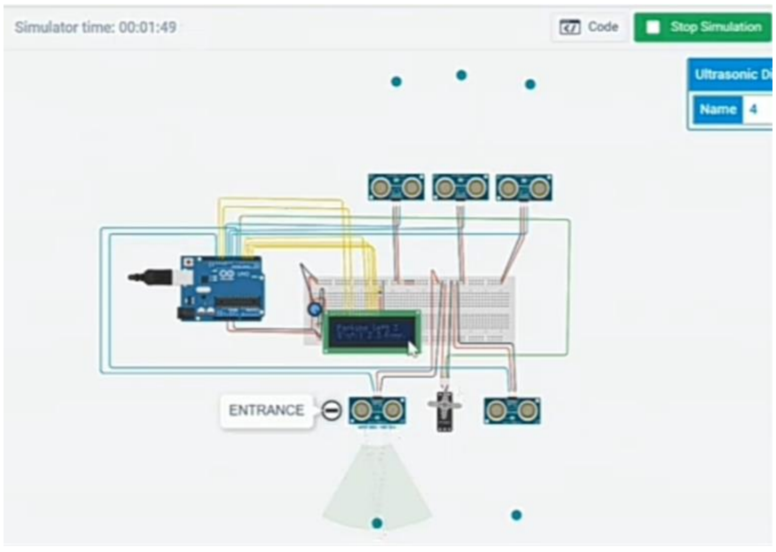


IR Receiver
Command: 2 Address: 0



```
Servo returned to initial position.
Servo returned to initial position.
```





Benefits of Using Smart Parking System:

Smart parking systems offer a number of benefits, including:

- Reduced traffic congestion: Smart parking systems can help to reduce traffic congestion by helping drivers to find available parking spaces more easily.
- Improved air quality: Smart parking systems can help to improve air quality by reducing the amount of time that drivers spend idling while looking for parking.
- Increased revenue for cities and businesses: Smart parking systems can help to generate revenue for cities and businesses by charging drivers for parking.
- Improved efficiency of parking lots: Smart parking systems can help to improve the efficiency of parking lots by providing parking lot operators with data about parking usage patterns.

Conclusion:

This project has described the design and implementation of a smart parking system using IoT technology. The system can be used to provide drivers with real-time information about parking availability, and to help them find and reserve parking spaces remotely. This can save drivers time and frustration, and can also help to reduce traffic congestion.

Team Members:

R.Deepika Sri-921021104006

T. Muthuvaisali-921021104030

L.Luckymalathi-921021104023

M.Sowmiya-921021104047