## I.OBJECTIVE:

This research project explores the development of a Smart Parking system by integrating Internet of Things (IOT) technology with Raspberry Pi, complemented by the creation of a mobile application using Python. The goal of this innovative system is to enhance the efficiency and convenience of parking management, providing real-time parking information to users via a mobile app. The project involves the deployment of IOT sensors in parking spaces, which communicate with a Raspberry Pi-based central server. The mobile app, developed in Python, interfaces .

**Keywords**: **Android application, smart car parking system, IR sensor**

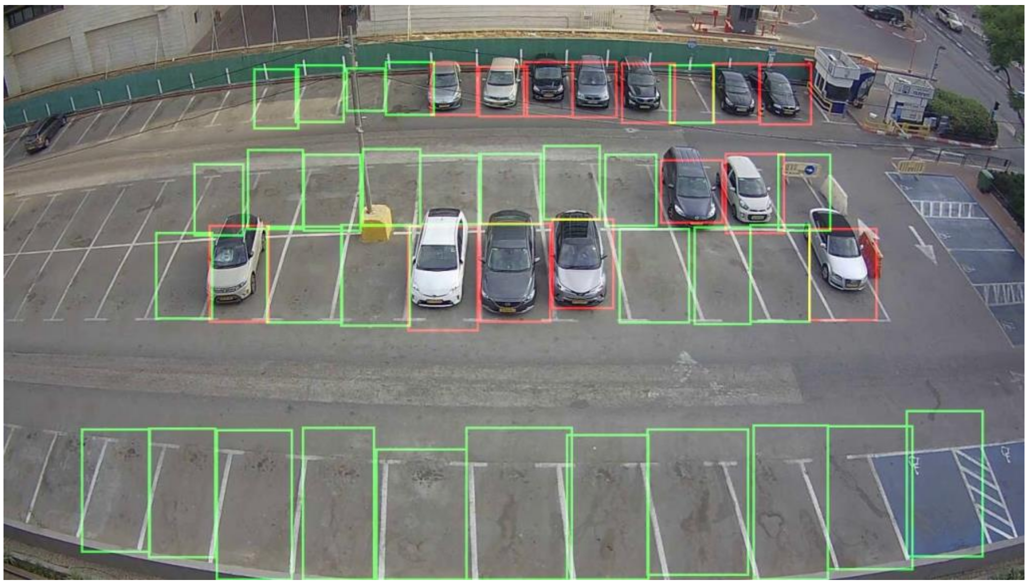
## II. INTRODUCTION:

A smart parking project refers to the implementation of technology and innovative solutions to optimize parking management and enhance the overall parking experience for both users and operators. Smart parking projects aim to address common parking challenges, such as congestion, inefficient space utilization, and the difficulty of finding available parking spots. These projects typically involve the use of various technologies and data-driven approaches to make parking more convenient, efficient, and sustainable.

## DEFINITION:

Smart parking is an IOT (Internet of Things) solution that uses sensors and/or cameras in combination with a software to inform users of vacant parking spaces in a certain area. Most of the time, people can also directly reserve the spot and pay for it with an app.

As a result of using Smart Parking, people who are looking to find a parking spot will find it in the most efficient way possible and companies or municipalities can optimize their parking territories. It also makes cities more liveable, safer and less congested.



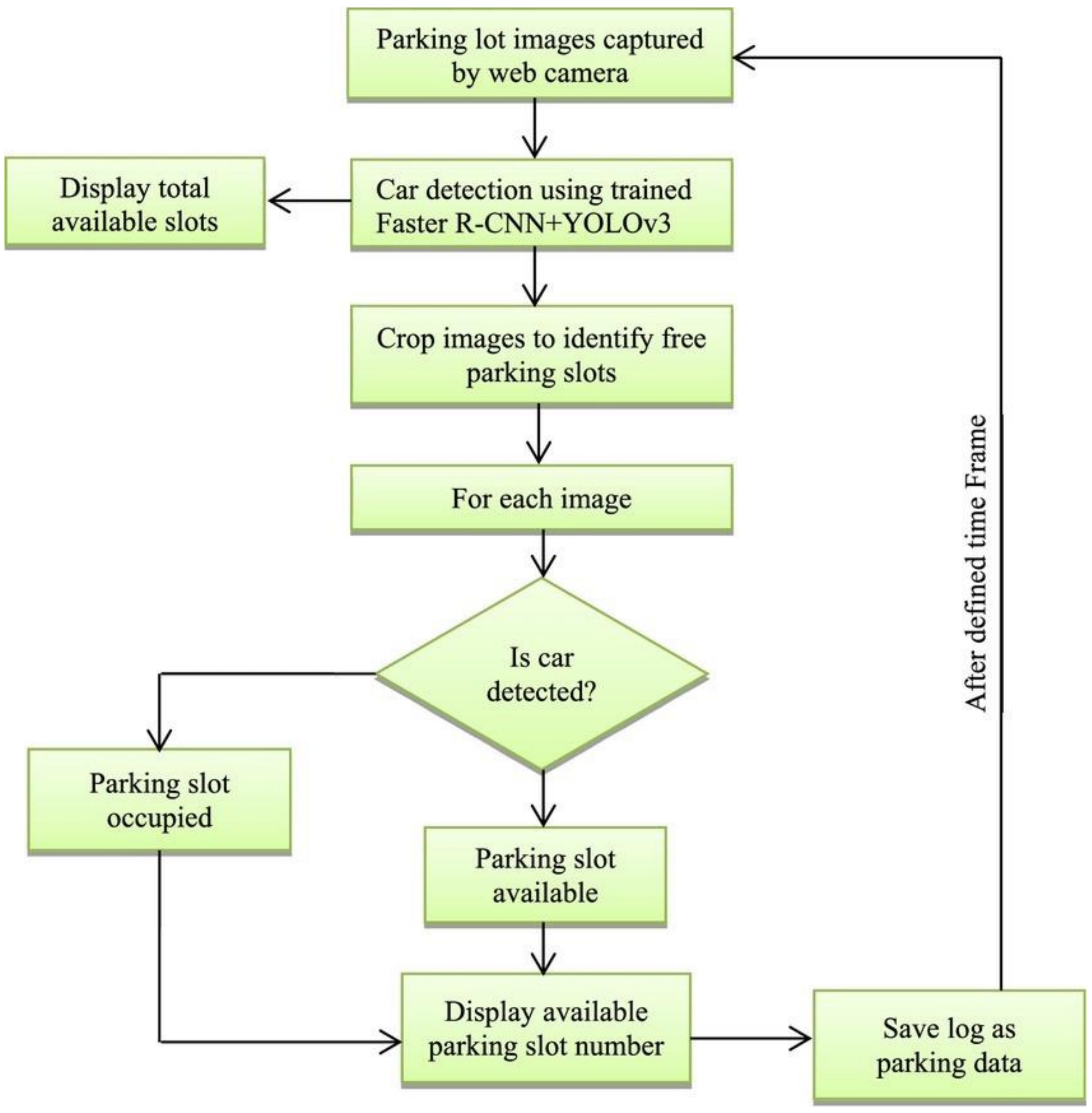
## IOT SENSOR:

IOT sensors are pieces of hardware that detect changes in an environment and collect data. They’re the pieces of an IOT ecosystem that bridge the digital world to the physical world. IOT sensors may detect things like temperature, pressure, and motion, and if they are connected to a network, they share data with the network.

## IOT BASED SMART PARKING:

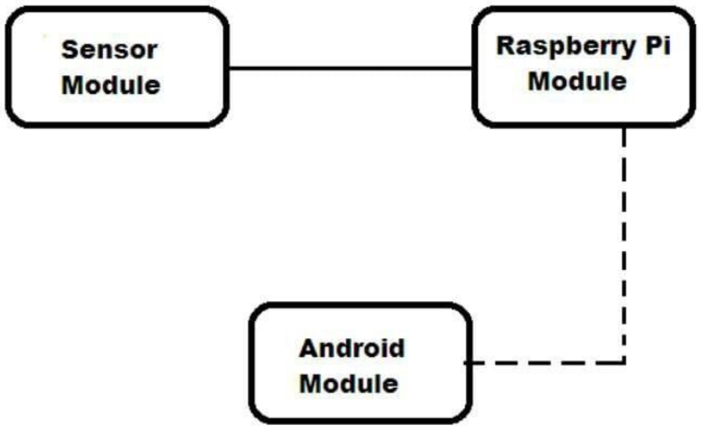
An IOT-based parking system is a centralized management that enables drivers to search for and reserve a parking spot remotely through their smartphones. It offers a convenient arrangement for drivers to park their cars when they are looking to avoid potential traffic congestion.

## FLOWCHART:



Parking lot images captured by web camera
Car detection using trained Faster R-CNN+YOLOv3
Display total available slots
Crop images to identify free parking slots
For each image
Is car detected?
Parking slot occupied
Parking slot available
Display available parking slot number
Save log as parking data
After defined time Frame

## Block Schematic Diagram



The system will be implemented in 3 modules

**Sensor Module**: This module will be installed in the parking place, there will be a sensor node for each parking space.

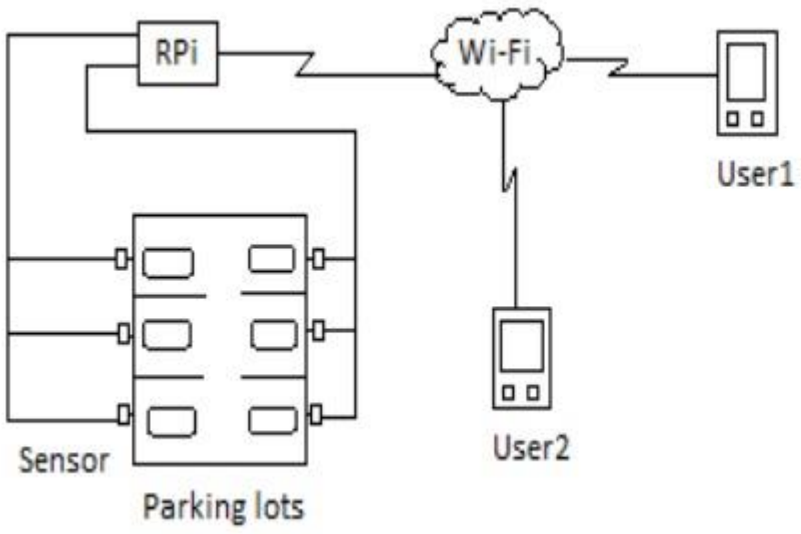
The sensors which will be used are IR- sensors.

**Raspberry Pi Module:** This module will be connected to the internet and will have connections from all sensor nodes.

**Android Module:** This module will be installed as an Android app in the users phones. And will display the parking lot status.

The system will require a Raspberry Pi with various IR sensors attached to it. The IR sensors will determine the parking status. The operating system of the raspberry Pi is Raspbian and to see the status of the parking in the parking lot we use the Android App. The parking lot setup (Raspberry Pi and IR sensor) will be accessible to the Android app over Wi-Fi network. The Android application will be used by users to check the parking status on their cell phones, and hence it will be the User interface of our project. The Raspberry Pi is interfaced with the IR sensors to determine the parking status will be the hardware setup of the project. Hence the raspberry pi becomes the hardware module of the system.

## System Architecture:



The system architecture has been divided into following 3 modules:

1) Raspberry Pi module 2) Sensors module

3) Android module

## Requirements:

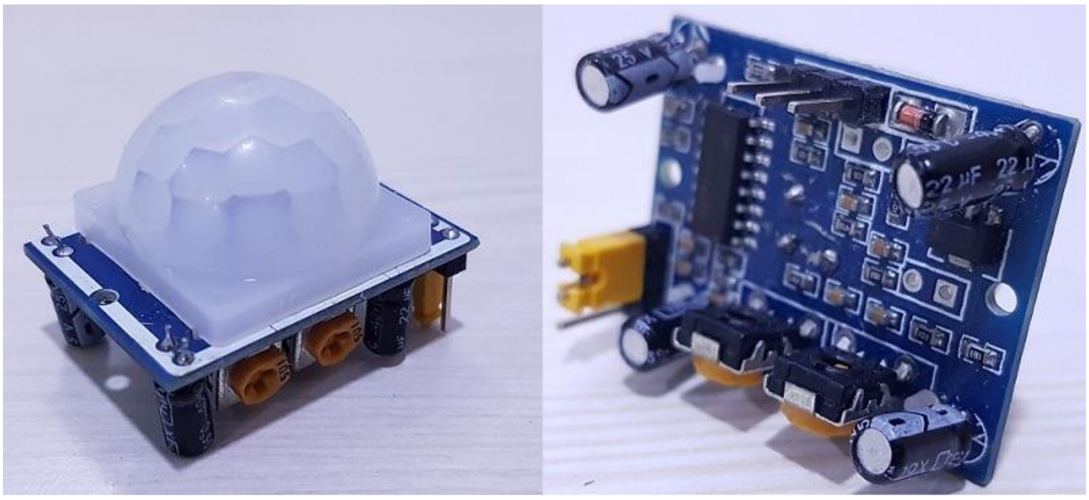
- PIR Sensor
- Raspberry pi

## PIR SENSOR:

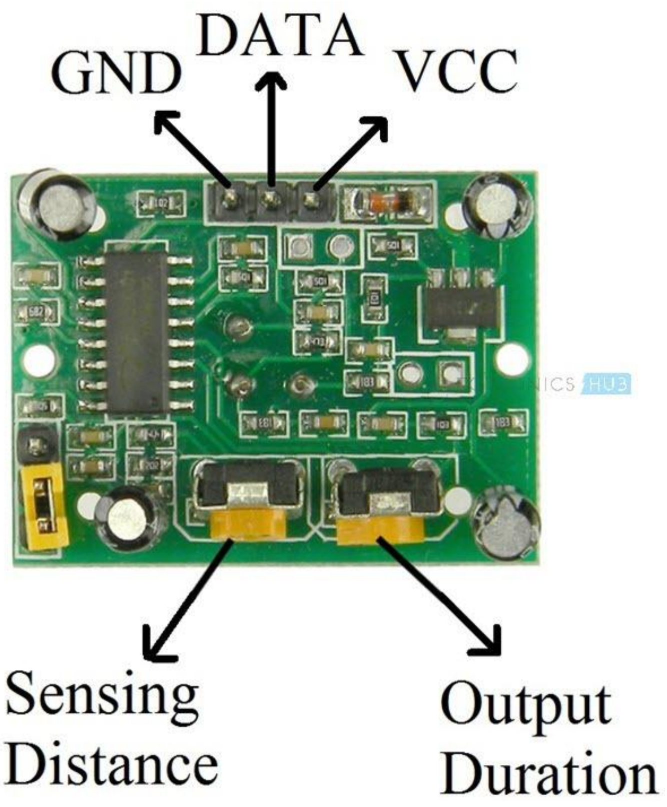
PIR sensor is used for detecting infrared heat radiations. This makes them useful in the detection of moving living objects that emit infrared heat radiations.

The output (in terms of voltage) of the PIR sensor is high when it senses motion; whereas it is low when there is no motion (stationary object or no object).

PIR sensors are used in many applications like room light control using human detection, human motion detection for security purposes at home, etc.



## PIR SENSOR ADJUSTMENT:



## RASPBERRRY PI:

It is a cheap, credit-card-sized device that uses a daily keyboard and mouse and joins to a TV or computer monitor. It is a thin weighable computer that let every person of all ages to discover programming and gain how to programme in variant languages like Python and Scratch. From exploring the internet and watching high-definition video, word-processing, to creating spreadsheets, and it can do every possible thing we'd expect a desktop computer to do and playing sports.

## CIRCUIT DESIGN:

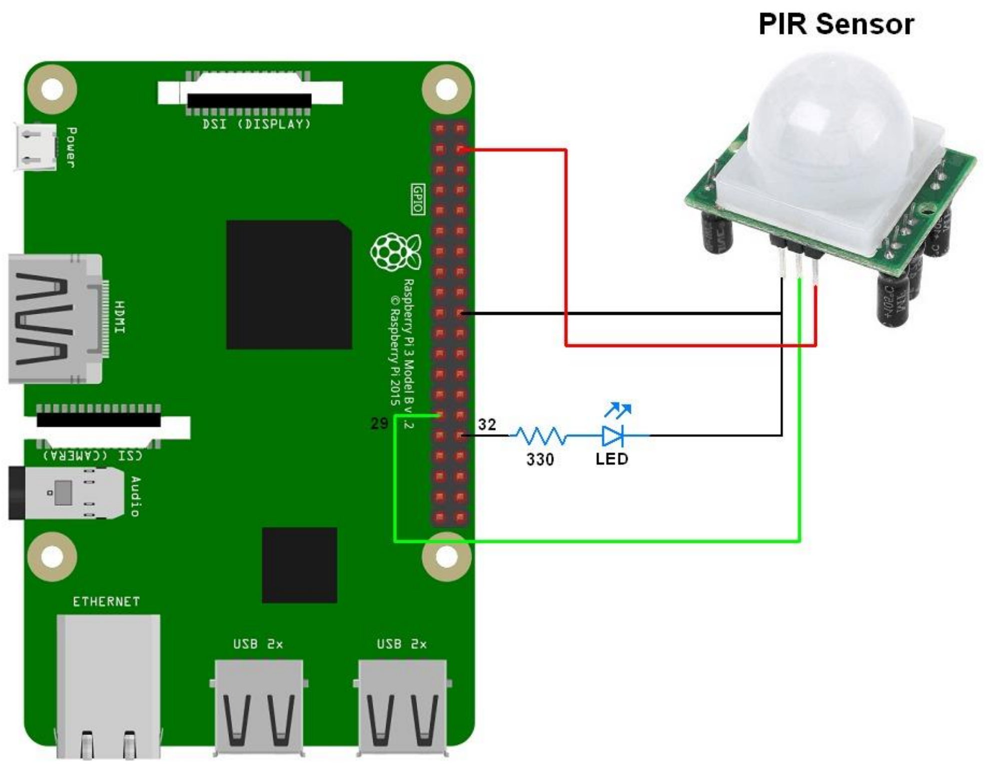
Connect the VCC and GND pins of the PIR Motion Sensor to +5V and GND pins of the Raspberry Pi. Connect the DATA Pin of the PIR Sensor to GPIO23 i.e. Physical Pin 16 of the Raspberry Pi.

A 5V Buzzer is connected to GPIO24 i.e. Physical Pin 18 of the Raspberry Pi. The other pin of the buzzer is connected to GN

## Requirement:

- Raspberry Pi 3 Model B
- PIR Sensor
- 5V Buzzer
- Connecting Wires
- Mini Breadboard
- Power Supply
- Computer

## CIRCUIT DIAGRAM:



## CONNECTION:

- Hardware Setup:Connect the VCC (power) pin of the PIR sensor to a 5V pin on the Raspberry Pi.Connect the GND (ground) pin of the PIR sensor to a ground pin on the Raspberry Pi.Connect the OUT pin of the PIR sensor to GPIO pin 7 (P7) on the Raspberry Pi.

- Software Configuration:Make sure you have Raspbian or another compatible operating system installed on your Raspberry Pi.Depending on the programming language you're using (e.g., Python), you can use libraries such as RPi.GPIO to interact with the GPIO pins.

**Applications:**

The applications of the PIR Motion Sensor using Raspberry Pi project have already been mentioned. Some of them are:

- ➢ Automatic Room Light
- ➢ Motion Detection
- ➢ Intruder Alert
- ➢ Automatic Door Opening
- ➢ Home Security System

**Operation:**

When a vehicle or a person moves within the PIR sensor's range, it will detect the motion and trigger the sensor's output pin. Your Raspberry Pi will read this signal, and you can use it to track parking space occupancy or trigger other actions, such as capturing an image with a camera module.

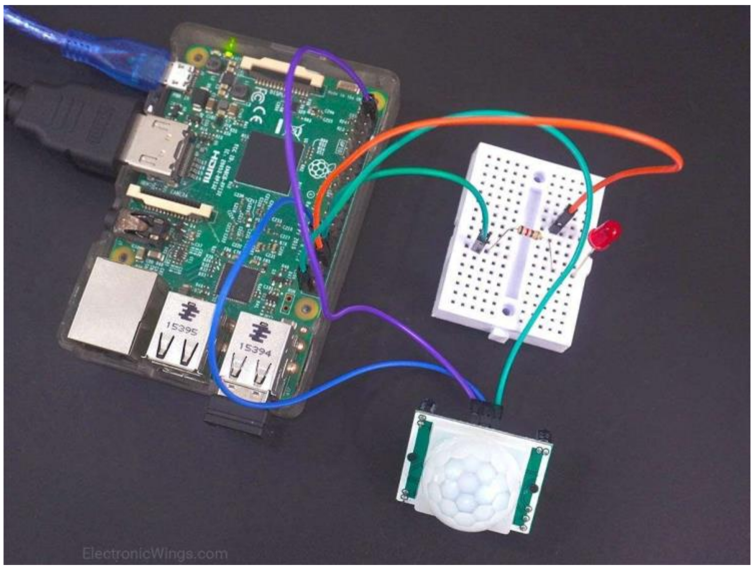
**Considerations:**

Adjust the PIR sensor's sensitivity and delay settings as needed. Secure the PIR sensor in a way that minimizes false detections from nearby movement. Combine PIR sensors with other sensors or cameras for a more robust smart parking system.

**Working:**

The working of the PIR Motion Sensor using Raspberry Pi is very simple. If the PIR Sensor detects any human movement, it raises its Data Pin to HIGH.

Raspberry Pi upon detecting a HIGH on the corresponding input pin, will activate the Buzzer.



**MOBILE APPLICATION:**

A mobile application, often referred to as a mobile app, is a software program designed specifically to run on mobile devices, such as smartphones and tablets. These applications are developed to provide various functions and services to users, enhancing their experience with the device. Mobile apps can be downloaded and installed from app stores like the Apple App Store for iOS devices and the Google Play Store for Android devices.

## Mobile application in IOT:

IOT mobile app development involves creating apps that connect physical objects, such as phones and watches, to the internet. These apps give users more control and convenience in their daily lives.

Developing these kinds of apps comes with unique challenges and opportunities for businesses.

Coding :

```
include <ESP8266WiFi.h>

#include <Servo.h>

#include <NTPClient.h>
```

```
#include <WiFiUdp.h>
#include <NTPClient.h>;
#include <WiFiUdp.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
const char *ssid = "Galaxy-M20"; // Enter your WiFi Name
const char *pass = "ac312124"; // Enter your WiFi Password
define MQTT_SERV "io.adafruit.com"
#define MQTT_PORT 1883
#define MQTT_NAME "aschoudhary"
#define MQTT_PASS "1ac95cb8580b4271bbb6d9f75d0668f1"
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 19800,60000);
Servo myservo; //servo as gate
Servo myservos; //servo as gate
int carEnter = D0; // entry sensor
int carExited = D2; //exi sensor
int slot3 = D7;
int slot2 = D6;
int slot1 = D3;
int count =0;
int CLOSE_ANGLE = 80; // The closing angle of the servo motor arm
int OPEN_ANGLE = 0; // The opening angle of the servo motor arm
int hh, mm, ss;
int pos;
int pos1;
```

```
String h, m,EntryTimeSlot1,ExitTimeSlot1, EntryTimeSlot2,ExitTimeSlot2,
EntryTimeSlot3,ExitTimeSlot3;

boolean entrysensor, exitsensor,s1,s2,s3;

boolean s1_occupied = false;

boolean s2_occupied = false;

boolean s3_occupied = false;

WiFiClient client;

Adafruit_MQTT_Client mqtt(&client, MQTT_SERV, MQTT_PORT, MQTT_NAME,
MQTT_PASS);

//Set up the feed you're subscribing to

Adafruit_MQTT_Subscribe EntryGate = Adafruit_MQTT_Subscribe(&mqtt,
MQTT_NAME "/f/EntryGate");

Adafruit_MQTT_Subscribe ExitGate = Adafruit_MQTT_Subscribe(&mqtt,
MQTT_NAME "/f/ExitGate");

//Set up the feed you're publishing to

Adafruit_MQTT_Publish CarsParked =
Adafruit_MQTT_Publish(&mqtt,MQTT_NAME "/f/CarsParked");

Adafruit_MQTT_Publish EntrySlot1 =
Adafruit_MQTT_Publish(&mqtt,MQTT_NAME "/f/EntrySlot1");

Adafruit_MQTT_Publish ExitSlot1 = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME
"/f/ExitSlot1");

Adafruit_MQTT_Publish EntrySlot2 =
Adafruit_MQTT_Publish(&mqtt,MQTT_NAME "/f/EntrySlot2");

Adafruit_MQTT_Publish ExitSlot2 = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME
"/f/ExitSlot2");

Adafruit_MQTT_Publish EntrySlot3 =
Adafruit_MQTT_Publish(&mqtt,MQTT_NAME "/f/EntrySlot3");

Adafruit_MQTT_Publish ExitSlot3 = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME
"/f/EExitSlot4”) ;

void setup() {
  delay(1000);
  Serial.begin (9600);
  mqtt.subscribe(&EntryGate);
  mqtt.subscribe(&ExitGate);
  timeClient.begin();
  myservo.attach(D4);      // servo pin to D6
  myservos.attach(D5);       // servo pin to D5
  pinMode(carExited, INPUT);    // ir as input
  pinMode(carEnter, INPUT);     // ir as input
  pinMode(slot1, INPUT);
  pinMode(slot2, INPUT);
  pinMode(slot3, INPUT);
  WiFi.begin(ssid, pass);                                   //try to connect with wifi
  Serial.print("Connecting to ");
  Serial.print(ssid);                          // display ssid
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");                         // if not connected print this
    delay(500);
  }
  Serial.println();
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP Address is : ");
  Serial.println(WiFi.localIP());                                    //print local IP address
}

void loop() {
```

```
MQTT_connect();
timeClient.update();
hh = timeClient.getHours();
mm = timeClient.getMinutes();
ss = timeClient.getSeconds();
h= String(hh);
m= String(mm);
h +" :" + m;

entrysensor= !digitalRead(carEnter);
exitsensor = !digitalRead(carExited);
s1 = digitalRead(slot1);
s2 = digitalRead(slot2);
s3 = digitalRead(slot3);

  if (entrysensor == 1) {                     // if high then count and send data
  count=  count+1;                            //increment count
  myservos.write(OPEN_ANGLE);
  delay(3000);
  myservos.write(CLOSE_ANGLE);
  }

  if (exitsensor == 1) {                          //if high then count and send
  count= count-1;                                 //decrement count
  myservo.write(OPEN_ANGLE);
  delay(3000);
```

```
   myservo.write(CLOSE_ANGLE);
  }
  if (! CarsParked.publish(count)) {}

  if (s1 == 1 && s1_occupied == false) {
       Serial.println("Occupied1 ");
       EntryTimeSlot1 =  h +" :" + m;
       //Serial.print("EntryTimeSlot1");
       //Serial.print(EntryTimeSlot1);

       s1_occupied = true;
       if (! EntrySlot1.publish((char*) EntryTimeSlot1.c_str())){}
    }
  if(s1 == 0 && s1_occupied == true) {
      Serial.println("Available1 ");
      ExitTimeSlot1 =  h +" :" + m;
      //Serial.print("ExitTimeSlot1");
      //Serial.print(ExitTimeSlot1);

      s1_occupied = false;
       if (! ExitSlot1.publish((char*) ExitTimeSlot1.c_str())){}
}
  if (s2 == 1&& s2_occupied == false) {
     Serial.println("Occupied2 ");
     EntryTimeSlot2 =  h +" :" + m;
     //Serial.print("EntryTimeSlot2");
     //Serial.print(EntryTimeSlot2);
```

```
if (s2 == 1&& s2_occupied == false) {
     Serial.println("Occupied2 ");
     EntryTimeSlot2 =  h +" :" + m;
     //Serial.print("EntryTimeSlot2");
     //Serial.print(EntryTimeSlot2);

     s2_occupied = true;
     if (! EntrySlot2.publish((char*) EntryTimeSlot2.c_str())){}
    }
  if(s2 == 0 && s2_occupied == true) {
      Serial.println("Available2 ");
      ExitTimeSlot2 =  h +" :" + m;
      //Serial.print("ExitTimeSlot2");
      //Serial.print(ExitTimeSlot2);

      s2_occupied = false;
       if (! ExitSlot2.publish((char*) ExitTimeSlot2.c_str())){}
  }
  if (s3 == 1&& s3_occupied == false) {
     Serial.println("Occupied3 ");
     EntryTimeSlot3 =  h +" :" + m;
    //Serial.print("EntryTimeSlot3: ");
     //Serial.print(EntryTimeSlot3);
     s3_occupied = true;
      if (! EntrySlot3.publish((char*) EntryTimeSlot3.c_str())){}
    }
  if(s3 == 0 && s3_occupied == true) {
      Serial.println("Available3 ");
      ExitTimeSlot3 =  h +" :" + m;
      //Serial.print("ExitTimeSlot3: ");
      //Serial.print(ExitTimeSlot3);
      s3_occupied = false;
       if (! ExitSlot3.publish((char*) ExitTimeSlot3.c_str())){ }
  }

  Adafruit_MQTT_Subscribe * subscription;
  while ((subscription = mqtt.readSubscription(5000)))
     {

   if (subscription == &EntryGate)
     {
     //Print the new value to the serial monitor
     Serial.println((char*) EntryGate.lastread);

   if (!strcmp((char*) EntryGate.lastread, "ON"))
      {
      myservos.write(OPEN_ANGLE);
      delay(3000);
      myservos.write(CLOSE_ANGLE);
    }
}
  if (subscription == &ExitGate)
     {
     //Print the new value to the serial monitor
```

```
      Serial.println((char*) EntryGate.lastread);

   if (!strcmp((char*) ExitGate.lastread, "ON"))
      {
       myservo.write(OPEN_ANGLE);
       delay(3000);
       myservo.write(CLOSE_ANGLE);
    }
}
}
}
void MQTT_connect()
{
  int8_t ret;

  // Stop if already connected.
  if (mqtt.connected())
  {
    return;
  }

  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) // connect will return 0 for connected
  {
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;
      if (retries == 0)
      {
        // basically die and wait for WDT to reset me
        while (1);
      }
  }
}
```