



# VIT<sup>®</sup>

**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

## Smart Parking System using ESP8266

By

**VATTIKONDA SAI JAYANTH– 21BIT0733**

During

**FALL SEMESTER (2023-2024)**

SUBMITTED TO:

**BHARANI DHARAN N**

## **Abstract:**

Now a day's growth in population is increasing very highly as that comparing numbers of car owners are also increasing. In the modern days, all are updating towards automation. We need to make our requirements to be available in faster, automatic and reliable way. In our country, many cities have developed and some will develop into smart cities in a while. When Since the cities are getting developed in a rapid way, automobiles scan the RFID are increasing in number. For the smart cities with increasing There will be automobiles need smart technology as in their comfort. So, people of the parking find difficult in parking their vehicles. Technology is getting into every domain and so as in parking places.

Each developer is developing according to their own visions application and implementing the project. We propose a Smart parking the parking Slot system which provide an optimal solution for parking problems to demonstrate. in crowded cities, because people waste more time for chasing free parking slots. So, using this system in real time we can help many other domains like decreasing of air pollution, reducing traffic problems, decreasing sound pollution. Instead of creating different solutions for each and other problems we can interlink them and create a common solution for every interlinked instance. Hence, the customer can easily access the parking status of a particular parking zone via internet in his mobile or car. We make their search for a parking slot makes more comfortable and efficient. So, we can decrease man power involved, since we connect it to IoT chain. We store the overall statistics of the parking slots.

This project is implemented using RFID Technology and IR sensors. For a particular user we assign a RFID tag attached to BLY the gate. We send some of the data to the cloud to process the data like average parking time of each car and car parking density in that particular parking lot using Wi-Fi modules. IR sensors are used to check for the parking slot status and communicate with the ESP32 and displays in the webpage.

Instead of parking the car idle, we provide a charging point for the electric cars and a car wash motors which are controlled according to the customer interest.

## **Introduction:**

Smart parking system are very important in smart cities because finding of free parking space is big challenges for car drivers or owners. Reason of this problem is large number of car owners in the country. It creates air, sound pollution and traffic congestion and waste of more fuel. So, our project gives the solution of that problem. It will be informed to driver in advance about the free parking slot on your web application.

For Implementation of the project we used ESP8266 which has an ATmega328P of AVR family. When the driver enters the parking lot, he/she should wait for IR sensor at the Entry gate side to detect and opens the door using Servo Motor and park in the available free slots and same happens when a car is exited, the IR Sensor will detect and send message to Servo to open the door. There will be a monitor in front of the gate for the status people of the parking slots. We use IR sensors for monitoring the status of each slot. If the car is present in the slot, web visions application and the android application indicates the status of parking the parking slot. We are using six parking slots in this project roblems to demonstrate.

## Components:

### Hardware:

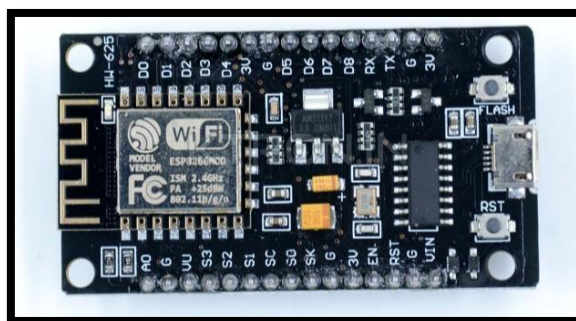
- NodeMCU ESP8266
- IR Sensor(4)
- Servo Motor(2)

### Online Services:

- Adafruit IO

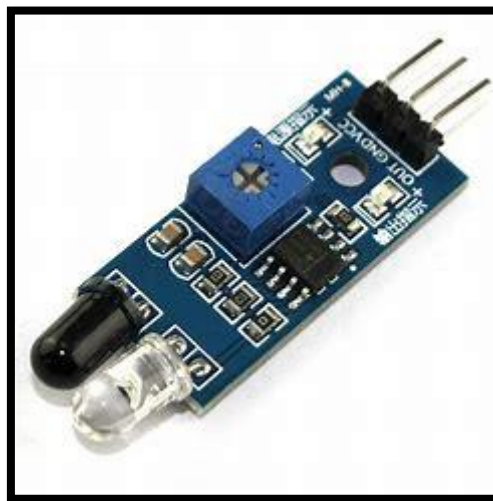
## ESP8266 Wi-Fi Module:

The ESP8266 Wi-Fi Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your Wi-Fi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Arduino device and get about as much Wi-Fi ability as a Wi-Fi Shield offers (and that's just out of the box)! The ESP8266 module is an extremely cost-effective board with a huge, and ever growing, community.



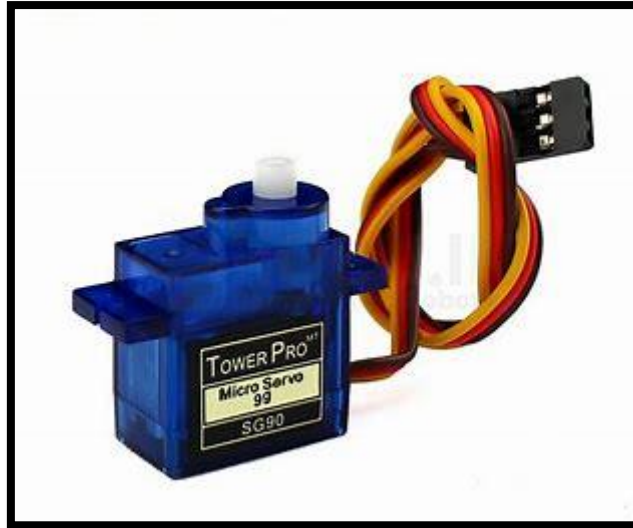
## IR Sensors:

An infrared sensor is an electronic device, that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. These types of sensors measures only infrared radiation, rather than emitting it that is called as a passive IR sensor. Usually in the infrared spectrum, all the objects radiate some form of thermal radiations. These types of radiations are invisible to our eyes, that can be detected by an infrared sensor. The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode which is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, the resistances and these output voltages, change in proportion to the magnitude of the IR light received.



## Servo Motor:

A servo motor is a self-contained electrical device that moves parts of a machine with high efficiency and great precision. In simpler terms, a servo motor is a BLDC motor with a sensor for positional feedback. This allows the output shaft to be moved to a particular angle, position, and velocity that a regular motor cannot do. However, a servo motor is only one part of a closed-loop motion control system. A complete motion system includes an amplifier, control circuit, drive gears, potentiometer, shaft, and either an encoder or resolver as well as the servo motor.

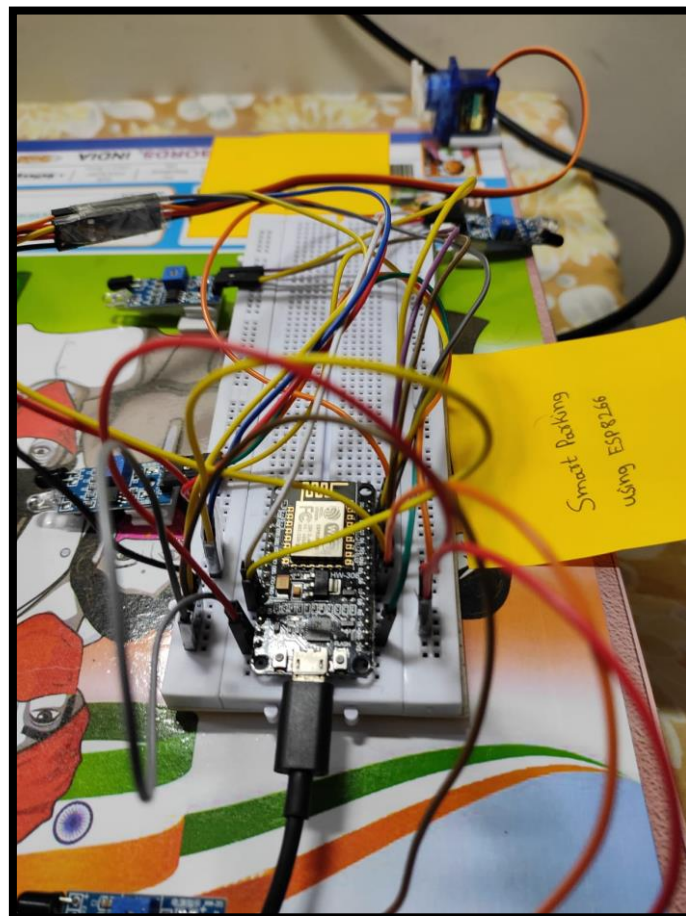
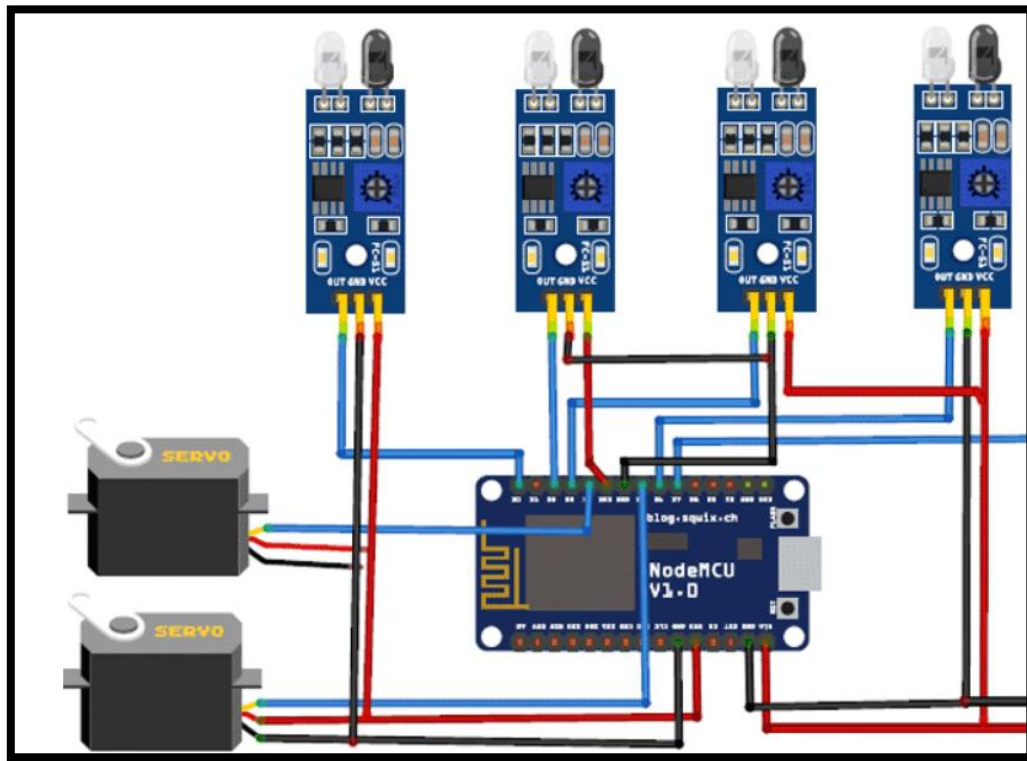


## Adafruit IO:

Adafruit IO is a platform designed to display, respond, and interact with your project's data. It is a system that makes data useful and focuses on ease of use, allowing simple data connections with little programming required. Adafruit IO includes client libraries that wrap their REST and MQTT APIs. It is built on Ruby on Rails, and Node.js.

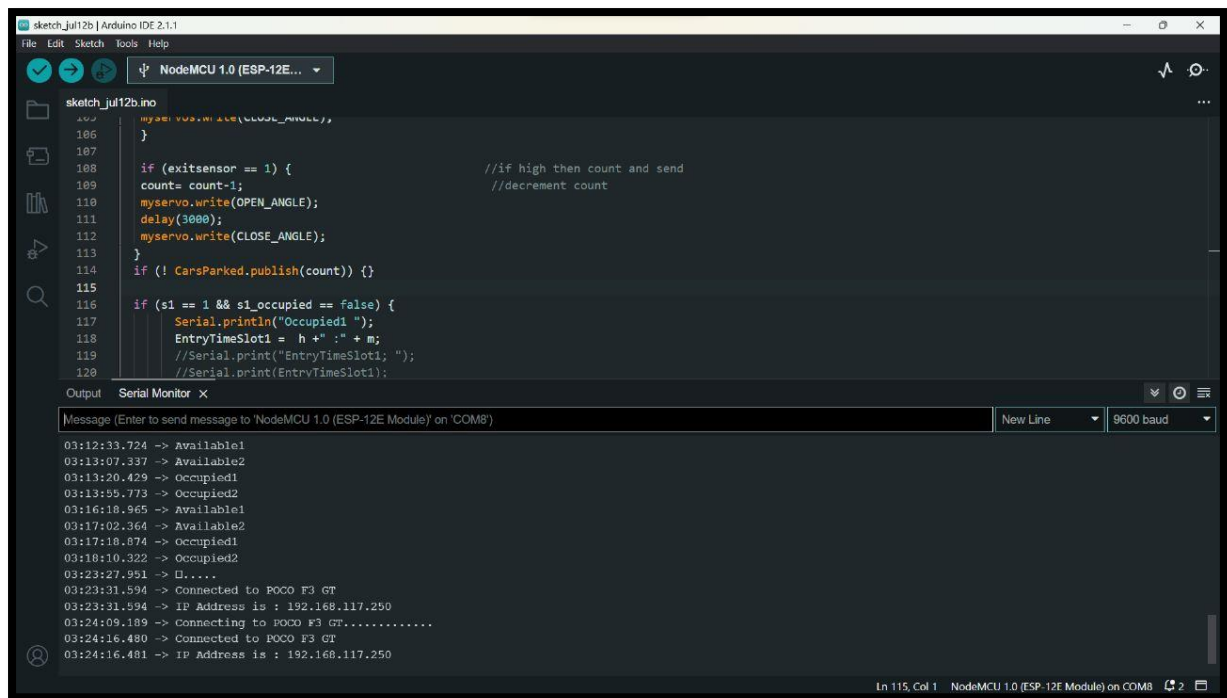


## Circuit Diagram:







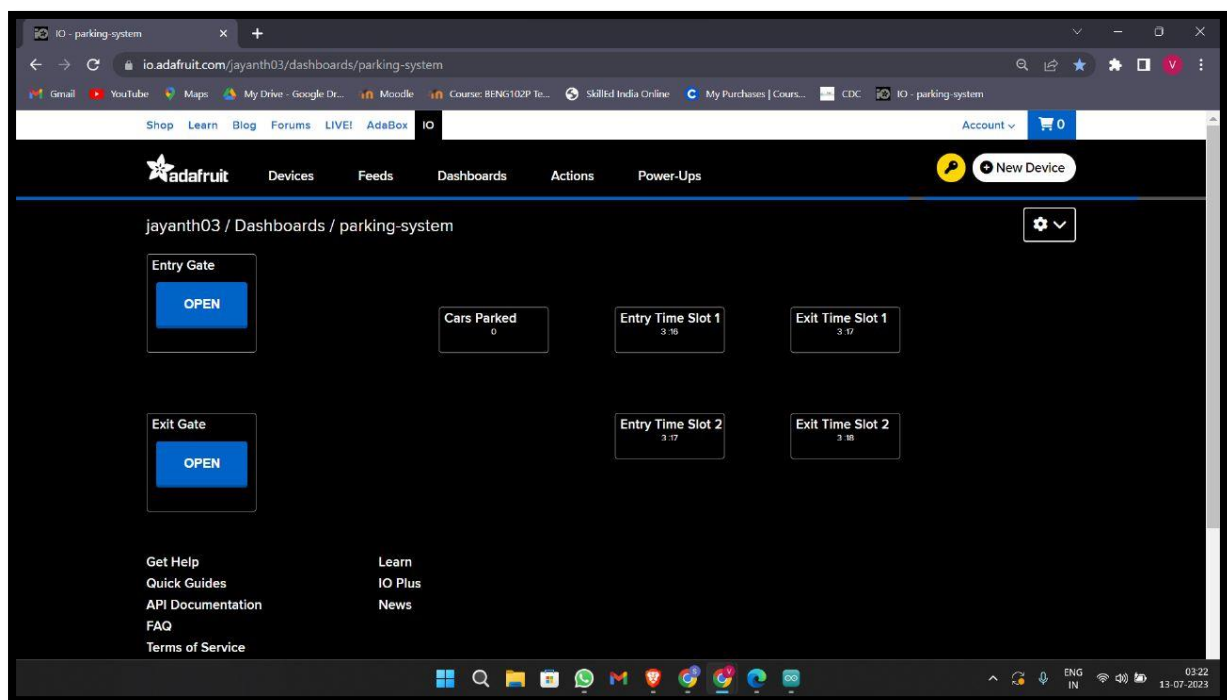


```
sketch_jul12b.ino
106
107
108 if (exitsensor == 1) {
109     count= count-1;
110     myservo.write(OPEN_ANGLE);
111     delay(3000);
112     myservo.write(CLOSE_ANGLE);
113 }
114 if (! CarsParked.publish(count)) {}
115
116 if (s1 == 1 && s1_occupied == false) {
117     Serial.println("Occupied1 ");
118     EntryTimeSlot1 = h + ":" + m;
119     //Serial.print("EntryTimeSlot1: ");
120     //Serial.print(EntryTimeSlot1);
121 }
```

Output Serial Monitor

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM8')

03:12:33.724 -> Available1  
03:13:07.337 -> Available2  
03:13:20.429 -> Occupied1  
03:13:55.773 -> Occupied2  
03:16:18.965 -> Available1  
03:17:02.364 -> Available2  
03:17:18.874 -> Occupied1  
03:18:10.322 -> Occupied2  
03:23:27.951 -> D.....  
03:23:31.594 -> Connected to POCO F3 GT  
03:23:31.594 -> IP Address is : 192.168.117.250  
03:24:09.189 -> Connecting to POCO F3 GT.....  
03:24:16.480 -> Connected to POCO F3 GT  
03:24:16.481 -> IP Address is : 192.168.117.250



**Drive Link:**

<https://drive.google.com/file/d/1k2028f2N9mWH7LcDi6qwcseC8Xftsshf/view?usp=drivesdk>

<https://io.adafruit.com/jayanth03/dashboards/parking-system>



## Code:

```
#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 = "POCO F3 GT"; // Enter your WiFi Name
const char *pass = "123456789"; // Enter your WiFi Password


#define MQTT_SERV "io.adafruit.com"
#define MQTT_PORT 1883
#define MQTT_NAME "jayanth03"
#define MQTT_PASS "aio_vnhT46E3lm8Zf1r981IU5VK7i7Z5"

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP, "pool.ntp.org", 19800,60000);

Servo myservo; //servo as gate

Servo myservos; //servo as gate


int carEnter = D1; // entry sensor
int carExited = D6; //exi sensor

//int slot3 = D7;

int slot2 = D3;

int slot1 = D2;

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/ExitSlot3");


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);
```

```
    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);
    }
}
}

```

## References:

Smart Parking System using IoT ElakyaR Juhi Seth, Pola Ashritha, R Namith

- 1) T. N. Pham, M. Tsai, D. B. Nguyen, C. Dow and D. Deng. "A Cloud- Based Smart Parking System Based on Internet of Things Technologies," in IEEEAC- cess, vol. 3, pp. 1581-1591. 2015 doi:10.1109/ACCESS 2015.2477299
- 2) Thanh Nam Phaml. Ming Fong Tsail, Duc Bing Nguyen, Chyi- Ren Dowland Der- Jiunn Deng2. "A CloudBased Smart- Parking System Based on Internet- of Things Technologies". IEEE Access, volume 3, pp. 1581-1591. september 2015.
- 3) Callum Rhodes, William Blewitt, Craig Sharp. Gary Ushaw and Graham Morgan. "Smart Routing: A Novel Application of Collaborative Path- finding to Smart Parking Systems". Business Informatics (CBI), 2014 IEEE Conference on volume 1, pp. 119- 126, 2014
- 4) Cui Shiyao, Wu Ming. Liu Chen, Rong Na. "The Re search and Implement of the Intelligent Parking Reser vation Management System Based on ZigBee Tech- nology". Measuring Technology and Mechatronics Au tomation (ICMTMA), pp. 741-744, January 2