Introduction :

A person can control the devices installed in his home or office from anywhere in the world by just using a smartphone or any internet connected devices.

parking solution will be built which will use an ultrasonic sensor to detect vehicle presence and trigger the gate to open or close automatically. The ESP8266 NodeMCU will be used here as the main controller to control all the peripherals attached to it.

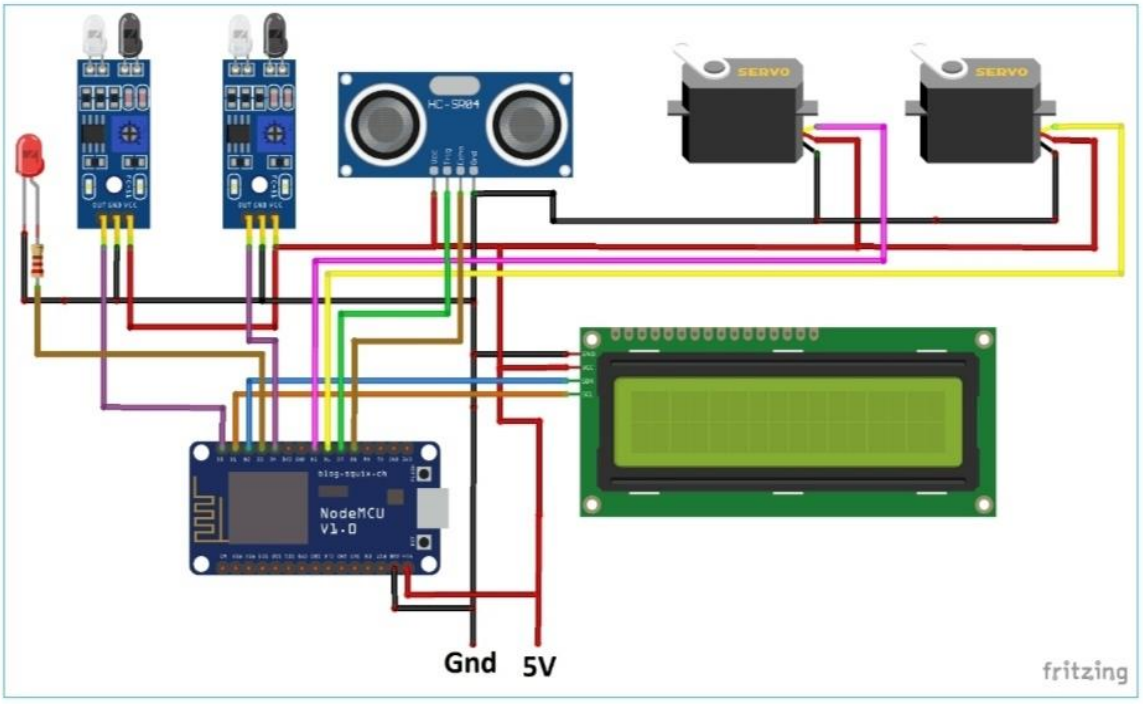
ESP8266 is the most popular controller to build IoT based applications as it has inbuilt support for Wi-Fi to connect to internet. We previously used it build many IoT projects like:

In this IoT Smart Parking System, we will send data to webserver for looking up the availability of space for vehicle parking. Here we are using firebase as Iot database to get the parking availability data.

Components Required :

1. ESP8266 NodeMCU
2. Ultrasonic Sensor
3. DC Servo Motor
4. IR Sensors
5. 16x2 i2c LCD Display
6. Jumper

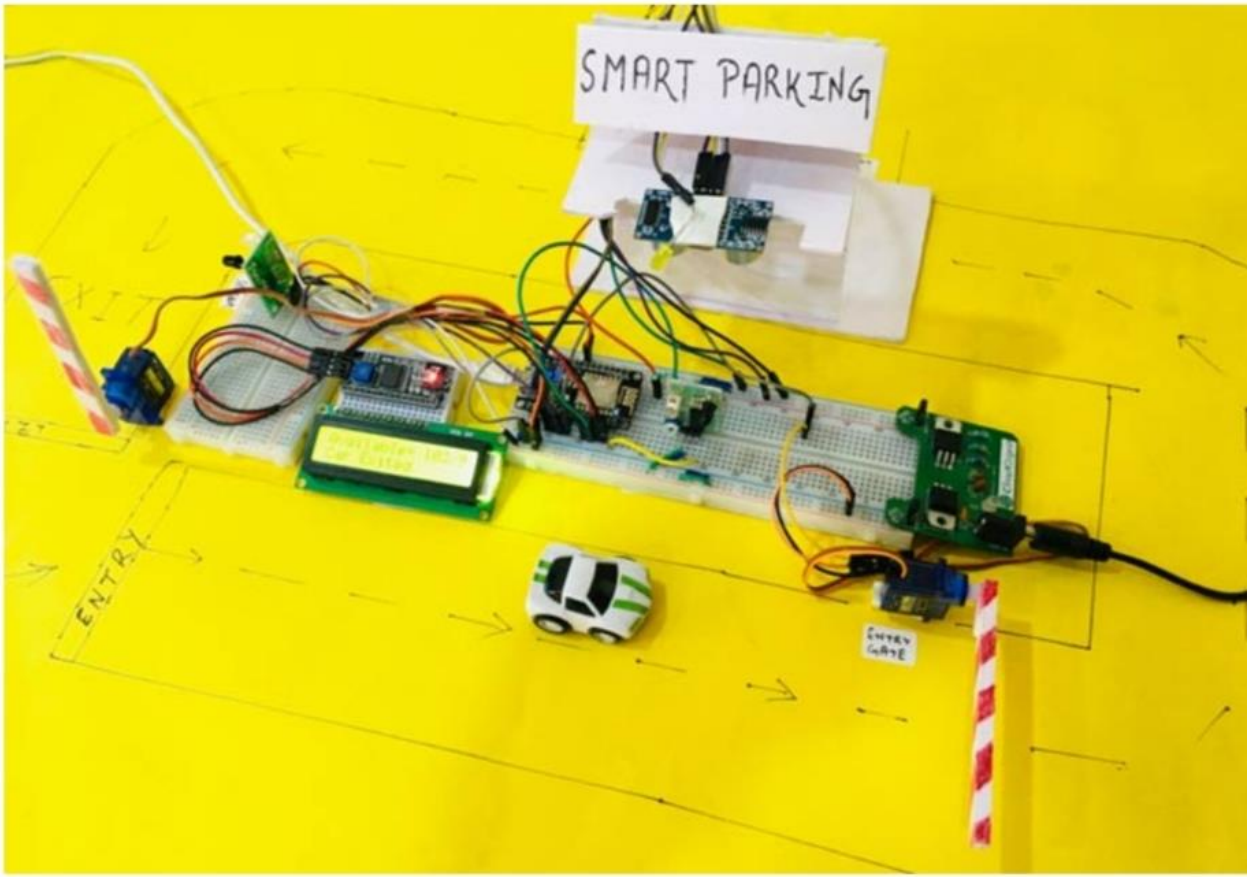
Circuit diagram :



Circuit diagram for this IoT based vehicle parking system is given below. It involves two IR sensor, two servo motors, one ultrasonic sensor and one 16x2 LCD.

Two IR sensors are used at entry and exit gate to detect the presence of car and automatically open or close the gate. IR Sensor is used to detect any object by sending and receiving the IR rays,

Two servos will act as entry and exit gate and they rotate to open or close the gate. Finally an Ultrasonic sensor is used to detect if the parking slot is available or occupied and send the data to ESP8266 accordingly.



Programming using node mcu :

For programming NodeMCU, just plug the NodeMCU to Computer with a Micro USB Cable and open Arduino IDE.

The libraries are required for I2C Display and Servo Motor. The LCD will display the availability of Parking Spaces and the Servo motors will be used to open and close the Entry and Exit gates.

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/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);
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);
      }
  }
}
```