

INTERNET OF THINGS

SMART PARKING



In this project we are going to construct a smart vehicle parking system that connects to internet to help a car driver or any vehicle owner to check if there is a vacant parking spot exist in a parking lot even before the driver reach the intended parking lot destination.

The proposed design:

As mentioned above, the proposed smart parking lot circuit will be equipped with several sensors, inexpensive microcontrollers and Wi-Fi module using which a car / any vehicle owner can check if there is a vacant space in a parking lot using his / her phone or tablet or even on computer.

The number of vacant spaces in the smart parking lot can be viewed from anywhere in the world using a URL link or the user can scan a QR code. The scanned / shared URL can be browsed on any web browser to know how many empty parking spot exist in real time.

IoT smart parking system for electric vehicle (EV) charging stations:

The proposed smart parking system is very useful in electric vehicle charging stations and this technology is going to a boon for those who are passing beside the charging stations equipped with this system. Now the motorists can see number of vacant chargers on their smartphone and plan their journey accordingly.

In conclusion, the main purpose of a smart vehicle parking system is to save time and reduce hassle for motorists to find a parking lot with a vacant parking spot; otherwise a driver may need to spend their time to find if there are any vacant parking spot left or should they move on to an another parking lot and this situation may put many motorists to mental stress especially those who are in an urgent circumstances.

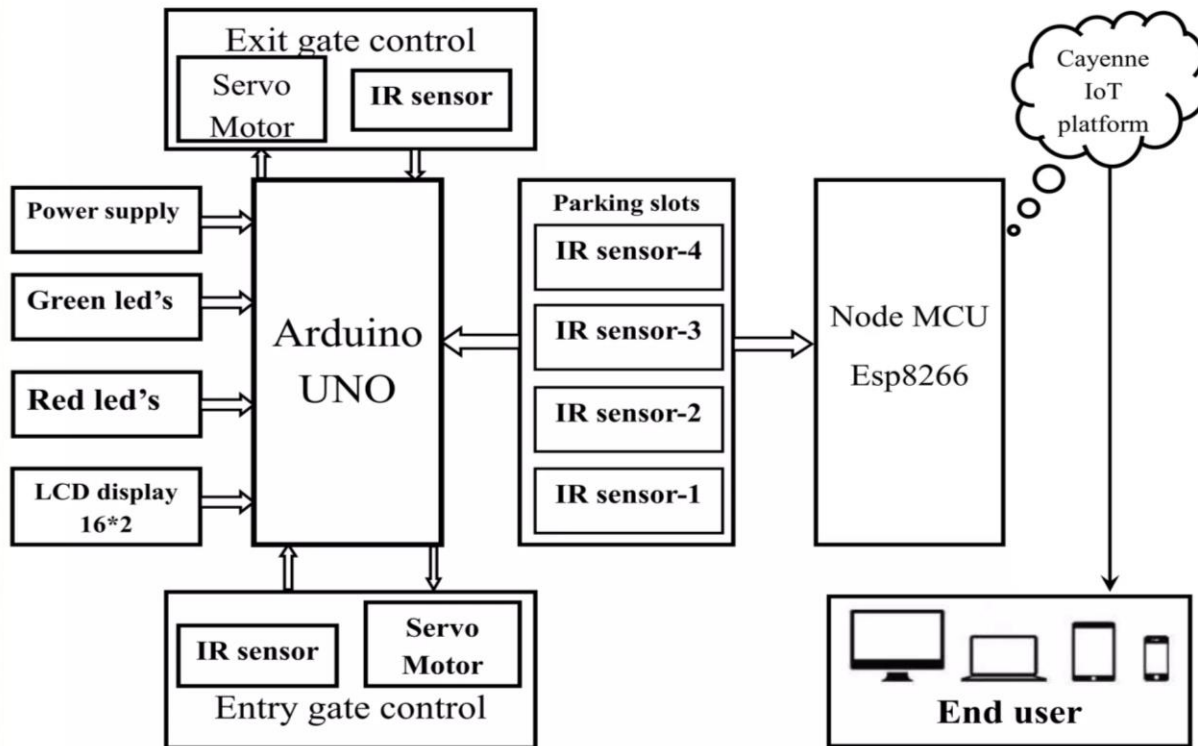
Hardware Requirements:

- **Nodemcu ESP8266**
- **IR Sensor**
- **LCD 16X2**
- **DC Motor**

Software Requirements:

- **Arduino IDE**
- **Orcad Design**

Block Diagram:

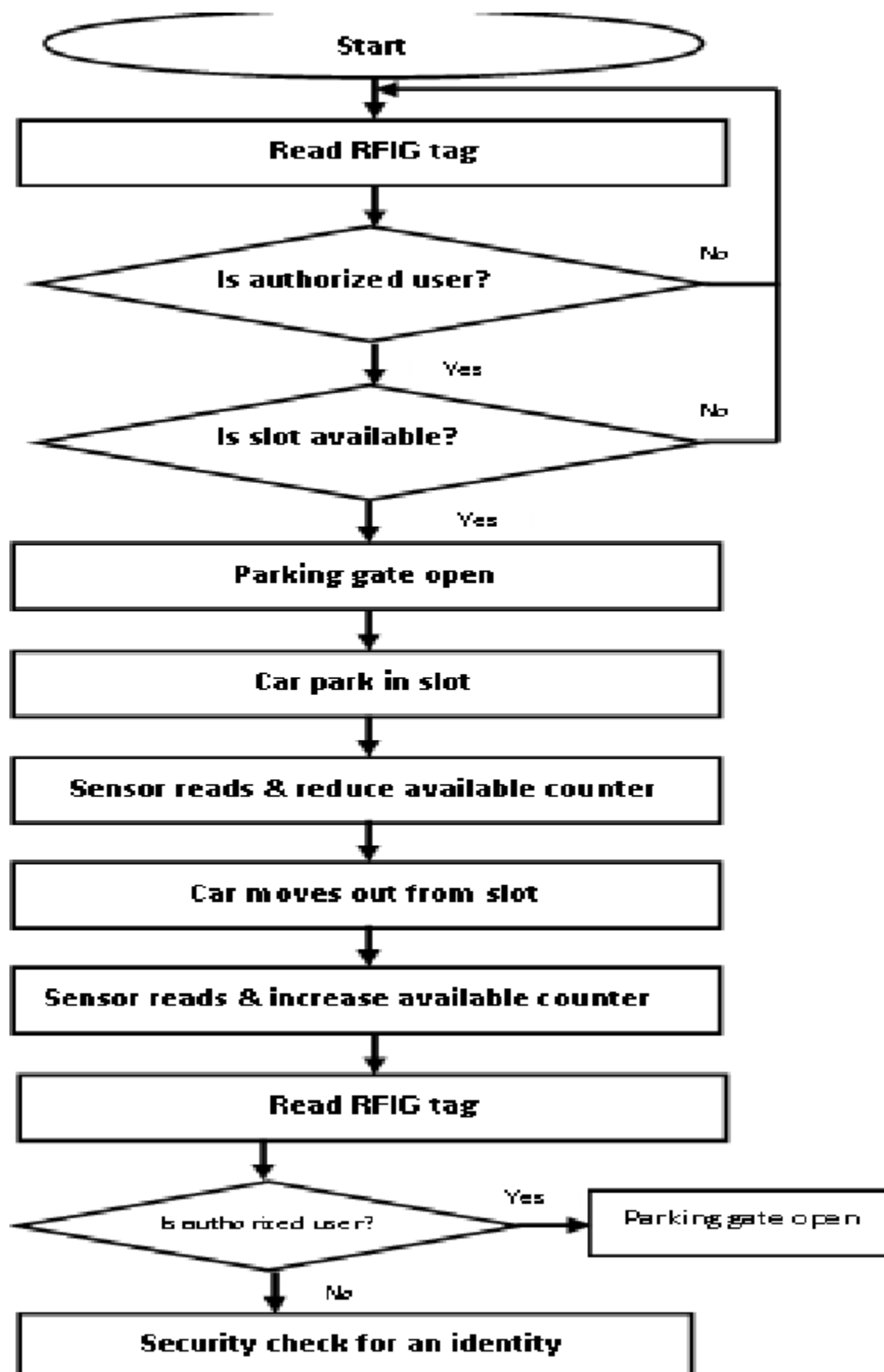


The circuit we are going to build will be based on the above architecture. An inexpensive Arduino board is going to be the brain of the project.

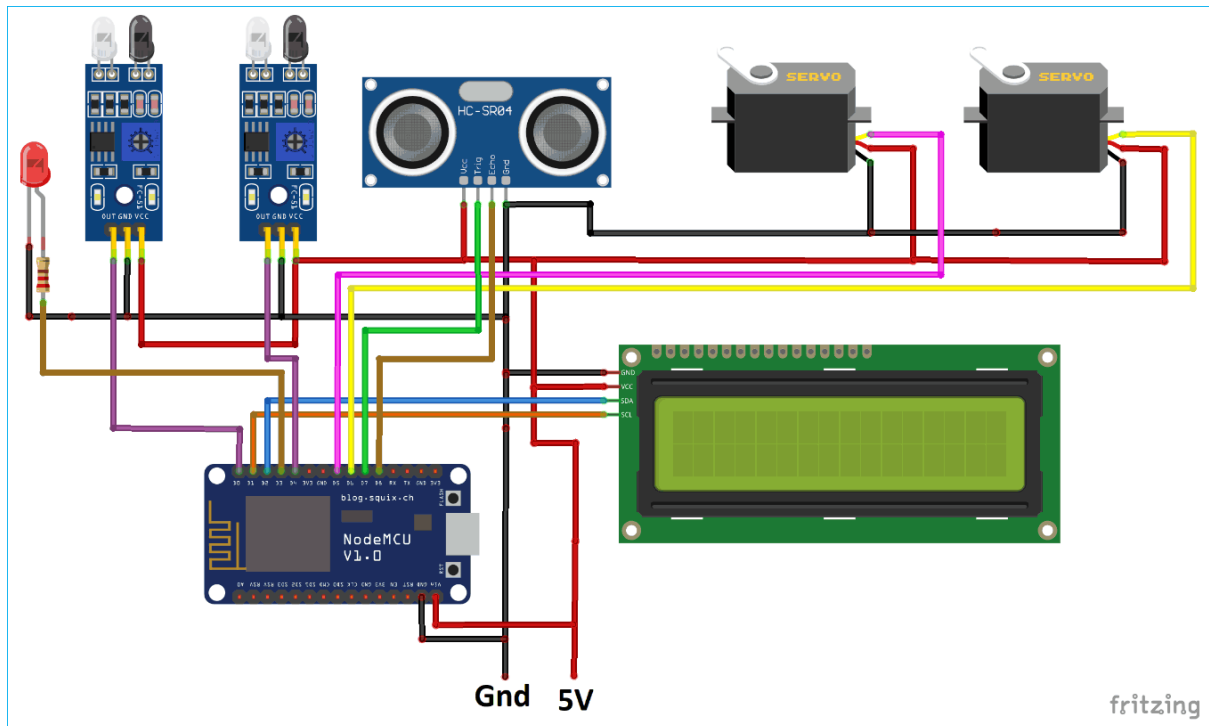
A 16 x 2 LCD is utilized for displaying the number of vacant spots locally (without internet). An I2C module is utilized for driving the LCD with just four wires so that GPIO pins can be saved for interfacing the sensors and other modules.

There are three ultrasonic sensors for detecting 3 cars / vehicles on the parking spot, we are using ultrasonic sensors instead of IR based sensors because if the parking lot is situated outdoors, infrared light from sunlight may interfere with IR sensors and may give incorrect detection of the vehicle, whereas ultrasonic sensor acts like a mini radar and environmental factors affecting its functionality is minimal.

Flow Chart:



Circuit diagram for IoT based car park monitoring system:



The above illustrated schematic consists of commonly available and easy to find modules. The brain of the project is an Arduino board and you can use any Arduino board with ATmega328p microcontroller.

Program code for Arduino:

```
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11);

LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display

const int trig_1 = 2;
const int echo_1 = 3;
const int trig_2 = 4;
```

```
const int echo_2 = 5;
const int trig_3 = 6;
const int echo_3 = 7;
float distanceCM_1 = 0, resultCM_1 = 0;
float distanceCM_2 = 0, resultCM_2 = 0;
float distanceCM_3 = 0, resultCM_3 = 0;
long Time_1, Time_2, Time_3;
float car_1, car_2, car_3;
float Dist_1 = 8.0, Dist_2 = 8.0, Dist_3 = 8.0;
int total = 0, timer_cnt = 0;
void setup()
{
  mySerial.begin(115200);
  pinMode(trig_1, OUTPUT);
  pinMode(trig_2, OUTPUT);
  pinMode(trig_3, OUTPUT);
  pinMode(echo_1, INPUT);
  pinMode(echo_2, INPUT);
  pinMode(echo_3, INPUT);
  digitalWrite(trig_1, LOW);
  digitalWrite(trig_2, LOW);
  digitalWrite(trig_3, LOW);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print(" IoT CAR PARK");
  lcd.setCursor(0, 1);
  lcd.print(" MONITOR SYSTEM");
  delay(2000);
  lcd.clear();
}
```

```
void loop()
{
    total = 0;
    car_1 = sensor_1();
    car_2 = sensor_2();
    car_3 = sensor_3();
    lcd.setCursor(0, 0);
    lcd.print("CAR1:");
    if (car_1 <= Dist_1)
    {
        lcd.print("OK ");
    }
    else
    {
        total += 1;
    }
    if (car_1 > Dist_1) lcd.print("NO ");
    lcd.print("CAR2:");
    if (car_2 <= Dist_2)
    {
        lcd.print("OK ");
    }
    else
    {
        total += 1;
    }
    if (car_2 > Dist_2) lcd.print("NO ");
    lcd.setCursor(0, 1);
    lcd.print("CAR3:");
    if (car_3 <= Dist_3)
```

```

{
    lcd.print("OK ");
}
else
{
    total += 1;
}
if (car_3 > Dist_3) lcd.print("NO ");
lcd.print("FREE:");
lcd.print(total);
if (timer_cnt >= 50)
{
    mySerial.print('*');
    mySerial.print(total);
    mySerial.println('#');
    timer_cnt = 0;
}
timer_cnt += 1;
delay(200);
}

float sensor_1(void)
{
    digitalWrite(trig_1, HIGH);
    delayMicroseconds(10);
    digitalWrite(trig_1, LOW);
    Time_1 = pulseIn(echo_1, HIGH);
    distanceCM_1 = Time_1 * 0.034;
    return resultCM_1 = distanceCM_1 / 2;
}

```



```

float sensor_2(void)
{
    digitalWrite(trig_2, HIGH);
    delayMicroseconds(10);
    digitalWrite(trig_2, LOW);
    Time_2 = pulseIn(echo_2, HIGH);
    distanceCM_2 = Time_2 * 0.034;
    return resultCM_2 = distanceCM_2 / 2;
}

```

```

float sensor_3(void)
{
    digitalWrite(trig_3, HIGH);
    delayMicroseconds(10);
    digitalWrite(trig_3, LOW);
    Time_3 = pulseIn(echo_3, HIGH);
    distanceCM_3 = Time_3 * 0.034;
    return resultCM_3 = distanceCM_3 / 2;
}

```

Program code for ESP8266:

```

#include "ThingSpeak.h"
#include <ESP8266WiFi.h>

//----- WI-FI details -----//
char ssid[] = "SSID"; //SSID here
char pass[] = "PASSWORD"; // Password here
//-----//

//----- Channel details -----//
unsigned long Channel_ID = 123456; // Your Channel ID
const char * myWriteAPIKey = "ACBDE12345"; //Your write API key

```

```
//-----//
```

```
const int Field_Number_1 = 1;
```

```
String value = "";
```

```
int value_1 = 0;
```

```
WiFiClient client;
```

```
void setup()
```

```
{
```

```
  Serial.begin(115200);
```

```
  WiFi.mode(WIFI_STA);
```

```
  ThingSpeak.begin(client);
```

```
  internet();
```

```
}
```

```
void loop()
```

```
{
```

```
  internet();
```

```
  if (Serial.available() > 0)
```

```
  {
```

```
    delay(100);
```

```
    while (Serial.available() > 0)
```

```
    {
```

```
      value = Serial.readString();
```

```
      if (value[0] == '*')
```

```
      {
```

```
        if (value[2] == '#')
```

```
        {
```

```
          value_1 = value[1] - 0x30;
```

```
        }
```


```
      }
```

```
    }  
  }  
  upload();  
}  
  
void internet()  
{  
  if (WiFi.status() != WL_CONNECTED)  
  {  
    while (WiFi.status() != WL_CONNECTED)  
    {  
      WiFi.begin(ssid, pass);  
      delay(5000);  
    }  
  }  
}
```

```
void upload()  
{  
  ThingSpeak.writeField(Channel_ID, Field_Number_1, value_1, myWriteAPIKey);  
  delay(15000);  
  value = "";  
}
```


This is how the **availability of parking can be tracked online on Firebase** as shown in the snapshot below:


← → ↻ 🔒 https://console.firebase.google.com/project/smart-parking-7f5b6/database/smart-parking-7f5b6/data


 **Firebase**


Project Overview ⚙️


Develop


 Authentication

 **Database**

 Storage

 Hosting

 Functions

 ML Kit

Quality

Crashlytics, Performance, Test Lab

Analytics


Dashboard, Events, Conversions, A/B Experiments

Spark

Free \$0/month **Upgrade**

smart-parking ▾

Database


 Realtime Database ▾


Data

Rules


Backups

Usage

 https://smart-parking-7f5b6.firebaseio.com/

 **Your security rules are defined as public, so anyone can steal, modify or delete your data.**

smart-parking-7f5b6

 Parking Status

-LiRj8MVilzSScNXkWaG: "Available=90/90"

-LiRj9DcO88_DcLIJaak: "Available=89/90"

-LiRjDuDe4y2tejSJ_X8: "Available=88/90"

-LiRjEiXXI4UHKITNvCL: "Available=87/90"

-LiRjJsmEEqknwU5-knN: "Available=86/90"

-LiRjM85RGwUdU9Z6xcA: "Available=85/90"

-LiRjQU4CnZuuDxXY0WK: "Available=86/90" ✕

-LiRjVNJk0FwAwI9ebyo: "Available=87/90"