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.

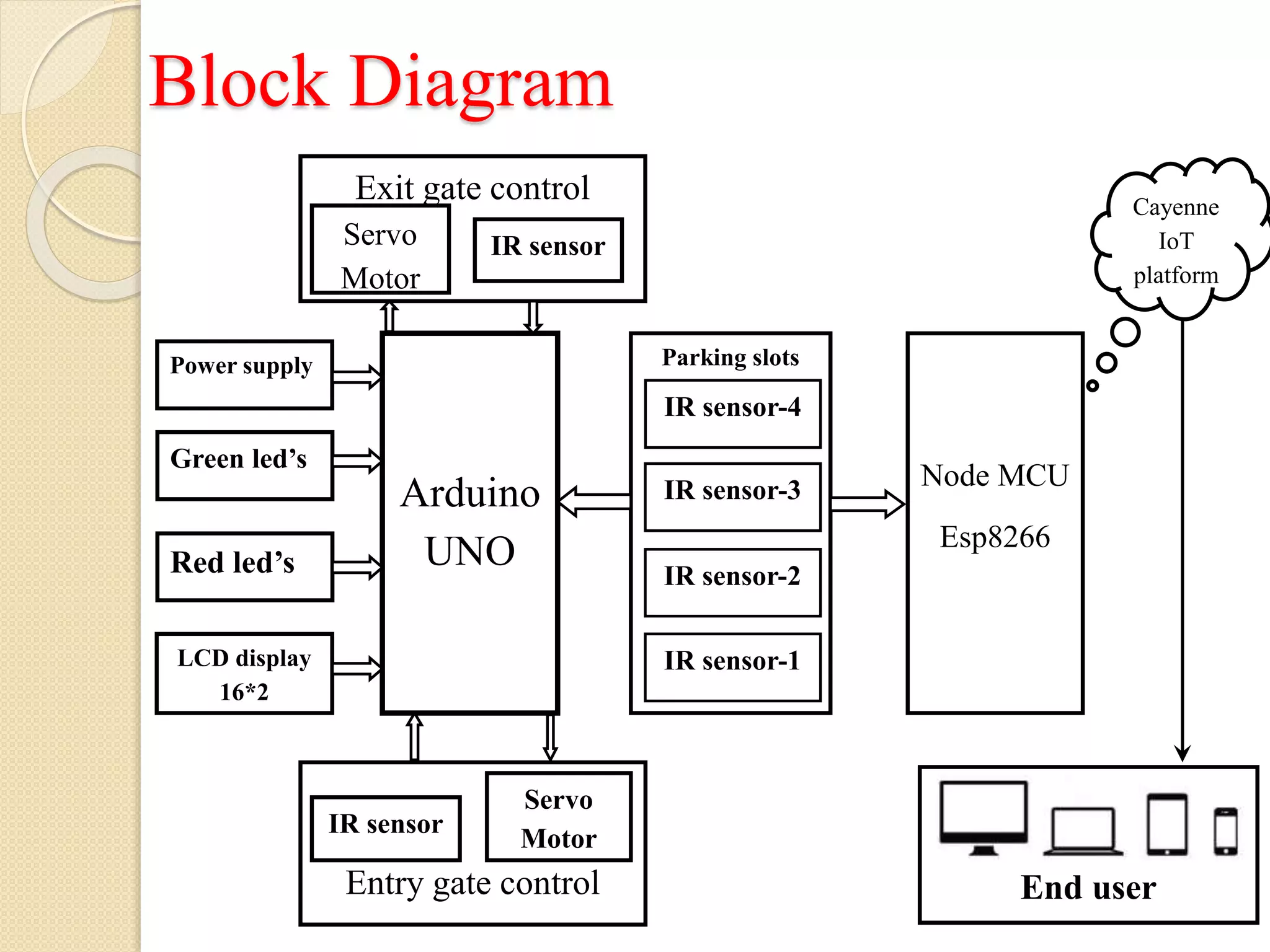
**Hardware Requirements:**

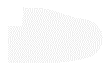
* **Nodemcu ESP8266**
* **IR Sensor**
* **LCD 16X2**
* **DC Motor**

**Software Requirements:**

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

**Block Diagram:**



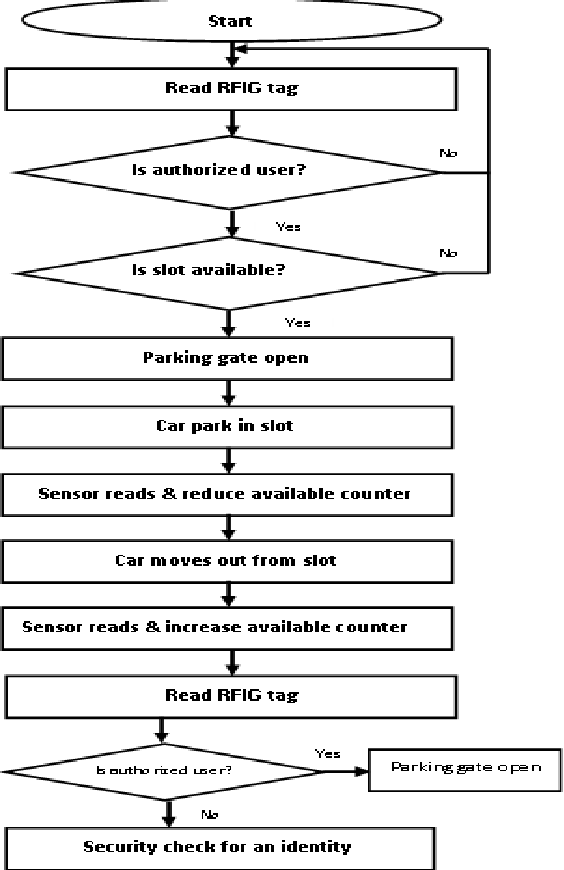


The circuit we are going to build will be based on the above architecture. An inexpensive Arduino board is going to be the brain of the project.

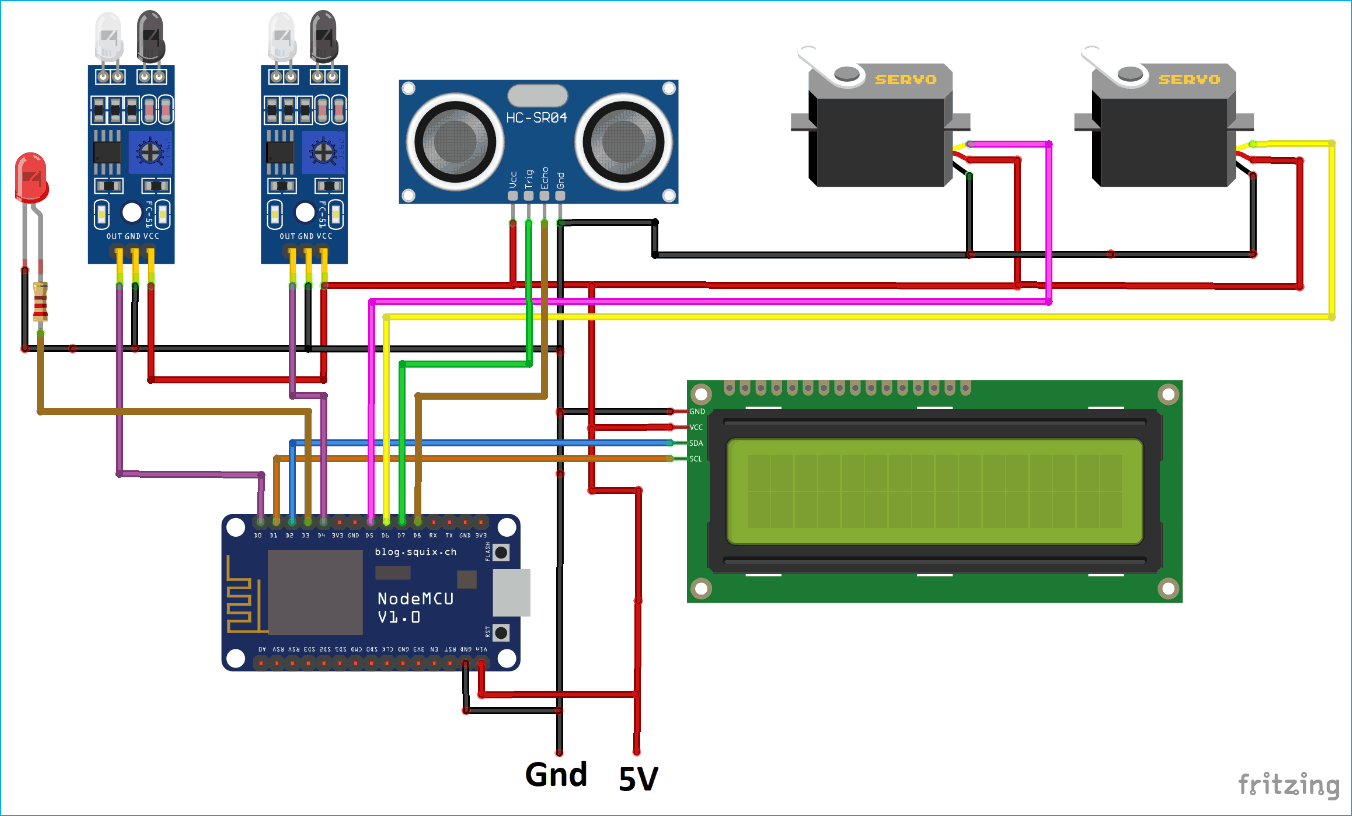
 A 16 x 2 LCD is utilized for displaying the number of vacant spots locally (without internet). An I2C module is utilized for driving the LCD with just four wires so that GPIO pins can be saved for interfacing the sensors and other modules.

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:

