Full name: Mohamed Ibrahim Adam

ID:3016

1 Contents

1 INTRODUCTION

1.1 PROJECT DETAILS:

A major problem in day to day life is parking of vehicles especially the car parking at an appropriate place. And this issue indirectly leads to traffic congestion. This project presents the basic concept of using app based smart parking services in smart cities as an important application of the Internet of Things (IoT) paradigm. This system will be accessible through a mobile app and can be used to monitor or find the empty slots in that area.

1.2 PURPOSE:

Moving towards smart city application, smart parking is a good example for a common citizen of how the Internet-of-Things (IoT) will be effectively and efficiently used in our daily living environments to provide different services to different users. Any citizen may use his mobile device, a computer having Internet to access the smart city application from anywhere in the world to find a free parking spot in the city and get to know the which parking spot is still available. The main purpose of this application is to reduce the on-road traffic and fuel consumption and make travelling eco-friendly and social. This entire process is made easier by the means of an application which people can use from their smartphones.

2 PROBLEM STATEMENT

To create an IoT-based Smart Parking System which allows users to check for empty slots in parking lots thereby reducing wastage of time. The users should be able to monitor several parking lots in real-time.

3 LITERATURE SURVEY

In today's world parking lots have become redundant and needs lot of manpower to handle and maintain it. These parking lots are not user friendly and do not provide data regarding availability of free spaces. Many researchers have contributed to this issue and formalized with various methods to better optimize the parking lot to serve the needs.

One author proposed smart parking reservation system using short message services (SMS), for that he uses Global System for Mobile(GSM) with microcontroller to enhances security. The ZigBee technique is used along with the GSM module for parking management and reservation. The author uses Global Positioning System (GPS) and Android platform to show available parking spaces.

Another author uses wide angle camera as a sensor which detect only free parking spaces and records them. These records are then used to assign parking space to the incoming user. Intelligent Transport System (ITS) and Electronic toll collection (ETC) using optical character recognition (OCR) creates a record for all entering vehicle. This creates tag less entry for all vehicles in the parking lot, but it does not assign a slot to the user.

4 H/W & S/W COMPONENTS

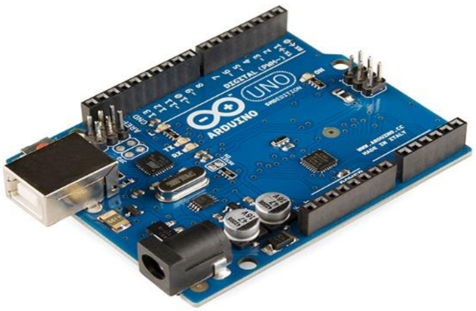
4.1 Hardware Components:

4.1.1 Arduino UNO

Arduino Uno is a microcontroller board developed by Arduino.cc which is an open-source electronics platform mainly based on AVR microcontroller Atmega328.

First Arduino project was started in Interaction Design Institute Ivrea in 2003 by David Cuartielles and Massimo Banzi with the intention of providing a cheap and flexible way to students and professional for controlling a number of devices in the real world.

The current version of Arduino Uno comes with USB interface, 6 analog input pins, 14 I/O digital ports that are used to connect with external electronic circuits. Out of 14 I/O ports, 6 pins can be used for PWM output. It allows the designers to control and sense the external electronic devices in the real world. This board comes with all the features required to run the controller and can be directly connected to the computer through USB cable that is used to transfer the code to the controller using IDE. Directly AC/DC power supply can also be supplied instead of using a USB cable.

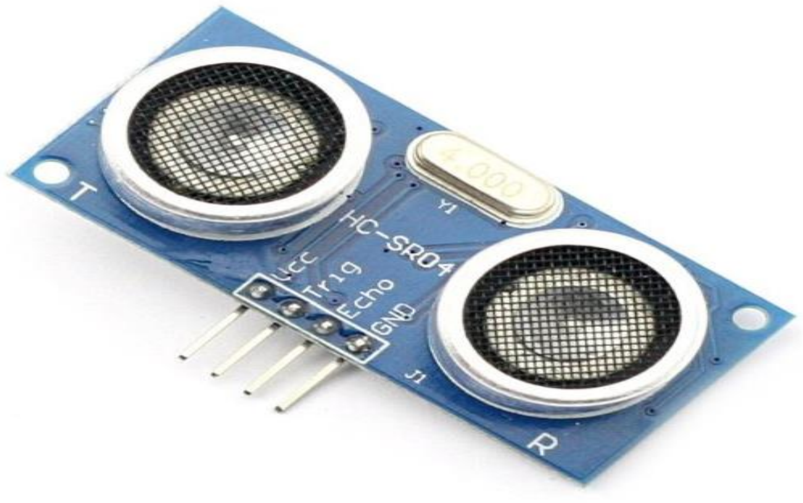


4.1.2 Node MCU

Node MCU is an open source LUA based firmware developed for ESP8266 Wi-Fi chip. By exploring functionality with ESP8266 chip, NodeMCU firmware comes with ESP8266 Development board/kit i.e.

NodeMCU Development board. Since NodeMCU is open source platform, their hardware design is open for edit/modify/build. NodeMCU Dev Kit/board consist of ESP8266 Wi-Fi enabled chip. The ESP8266 is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol. For more information about ESP8266, you can refer ESP8266 Wi-Fi Module.

4.1.3 **Ultrasonic sensors** – Ultrasonic sensors transmit pressure waves of sound energy at frequencies above the human audible range (25-50 kHz). These sensors use the portion of energy that returned after bouncing on a surface to measure the distance of objects from the sensor. In the absence of vehicles, a sensor installed above the road surface will measure the distance to the road surface. When a vehicle enters the sensing field, a change a shorter distance will be measured, thus triggering a vehicle presence signal to be generated.



4.1.4 **RFID tags** – RFID systems uses radio waves to exchange data between a reader and an electronic tag attached to an object. This type of identification has long been used by toll authorities to enable the automated collection of tolls. They are similar in concept to a smart card (which can in many cases be using RFID technology for communication). In the context of parking facilities, RFID readers may be used to scan vehicle entering a lot. Vehicles for which a valid tag is detected would then be granted access of registered as having entered the facility. If each tag is linked to a specific account, parking fees could then automatically be debited from the account upon entry in the case of fixed fees, or upon exit in the case of time-based fees.

4.1.5 Servo Motor

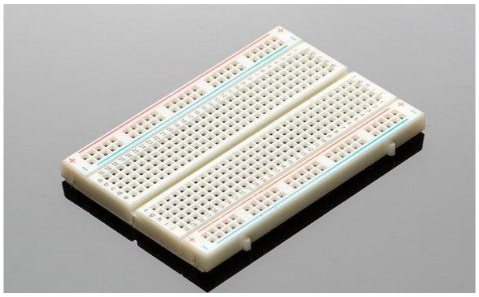
It is a rotator device that allows the control of angular as well as linear motion. A servo motor is used for the opening and closing of the gate. Servo drive transmits electrical signals to the servo motor for producing motion.

4.1.6 **Light Emitting Diodes (LED) signs** – Sign displaying current parking space situations using green, or red light-emitting diodes placed in matrix arrays. LED arrangements can be realized to show specific words, such as “Free”, “Closed”, or to offer freely programmable options. A particular advantage of LED displays is their high visibility at night, durability, and low maintenance needs.

4.1.7 **LCD signs** – LCD technology permits the display of current parking situation using highly legible, reflecting LCD characters. Additional informative text using highly legible proportional fonts is also easy to implement. Nighttime illumination of LCD displays can further be realized by back lighting (white LEDs, or fluorescent tubes).

4.1.8 Breadboard

A breadboard is a solderless device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. The breadboard has strips of metal underneath the board and connect the holes on the top of the board.



4.1.9 Jumper wires

Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboards and other prototyping tools in order to make it easy to change a circuit as needed.

4.1.10 USB Cable

The term USB stands for "Universal Serial Bus". USB cable assemblies are some of the most popular cable types available, used mostly to connect computers to peripheral devices such as cameras, camcorders, printers, scanners, and more. Devices manufactured to the current USB Revision 3.0 specification are backward compatible with version 1 .1.

4.2 Software Components:

4.2.1 Arduino IDE

The **Arduino integrated development environment (IDE)** is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

4.2.2 Blynk for Android

Blynk is a hardware-agnostic IoT platform with customizable mobile apps, private cloud, rules engine, and device management analytics dashboard. Blynk platform is easy to use for developers and saves a lot of time on integration. It also allows us to publish our apps into the market faster.

5 DESIGN

5.1 Basic Workflow Design and Diagram

The Ultrasonic sensors are used for car detection. Once the car is detected then a command is sent to the Node MCU module which then sends the desired command to the virtual pin of the Blynk App.

The basic design for this project involves sending data from Ultrasonic Sensors to the Arduino board which is connected to the Node MCU Board that uses Wi-Fi to send data to Blynk cloud which in turn is displayed on the Android app. The end user can use the app to check out empty parking slots.

The flow can be understood from the below diagram as well.

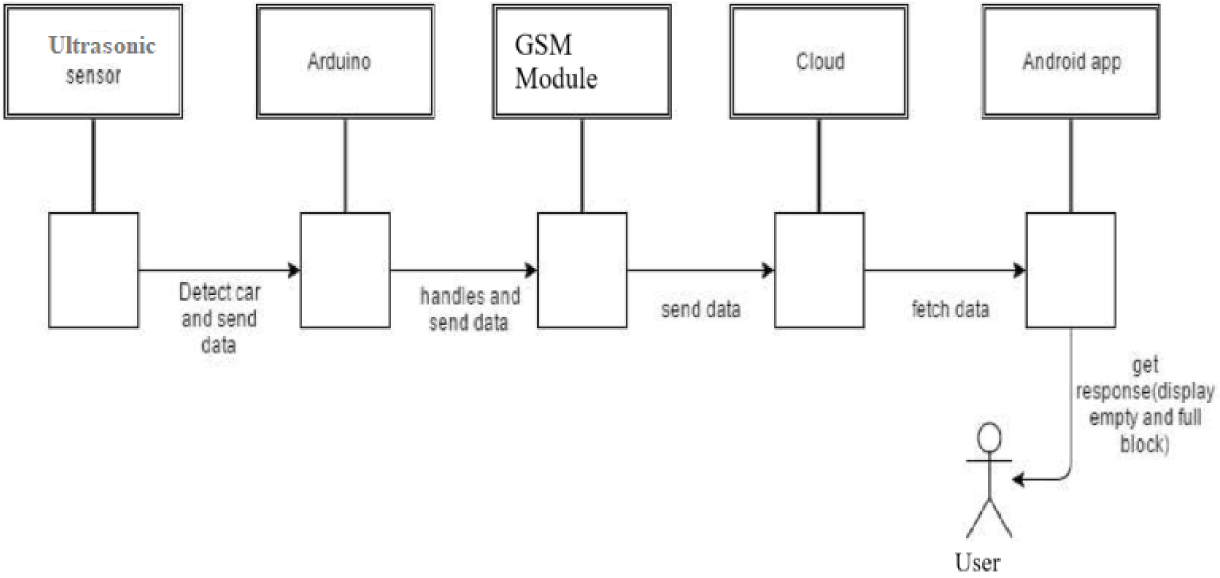


Diagram depicting the Workflow

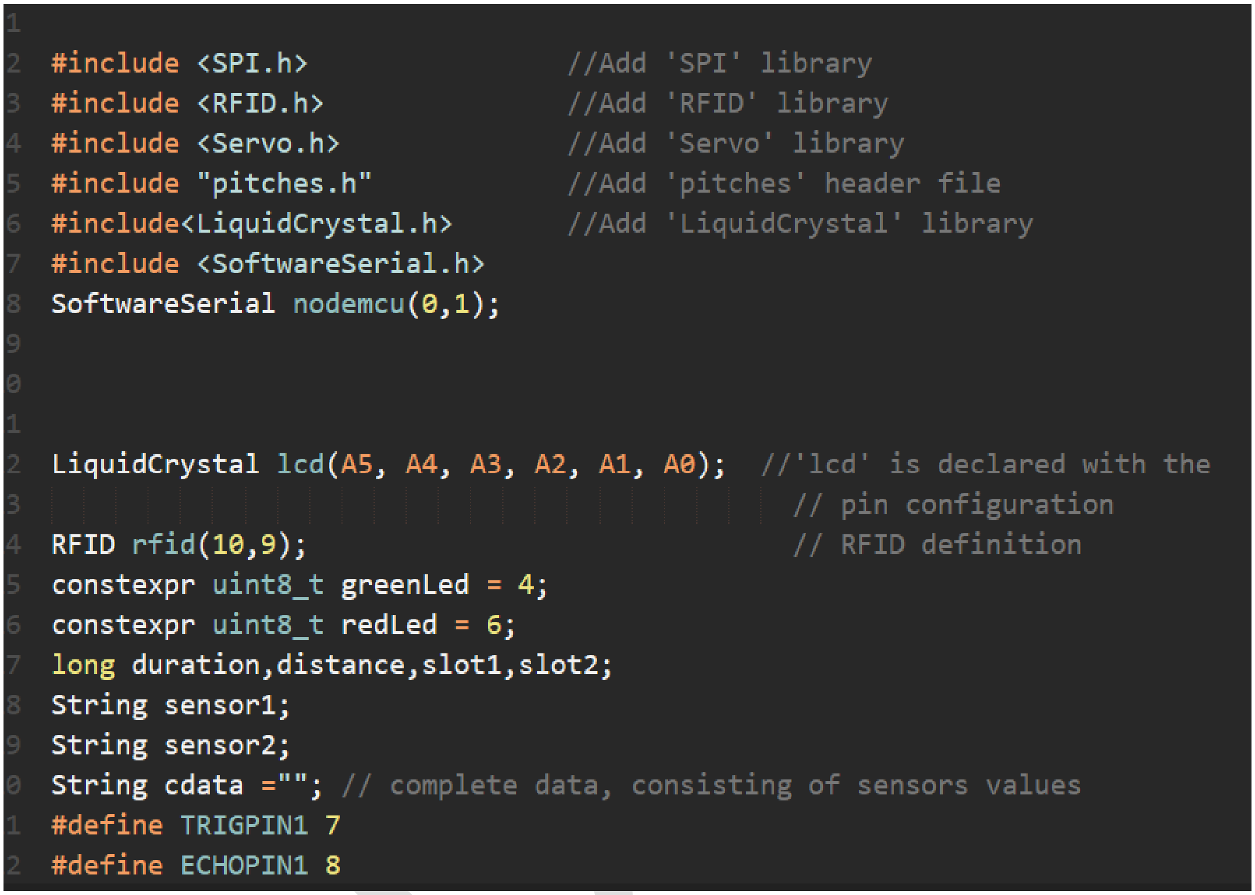
6 PROJECT IMPLEMENTATION

6.1 Steps in Implementation:

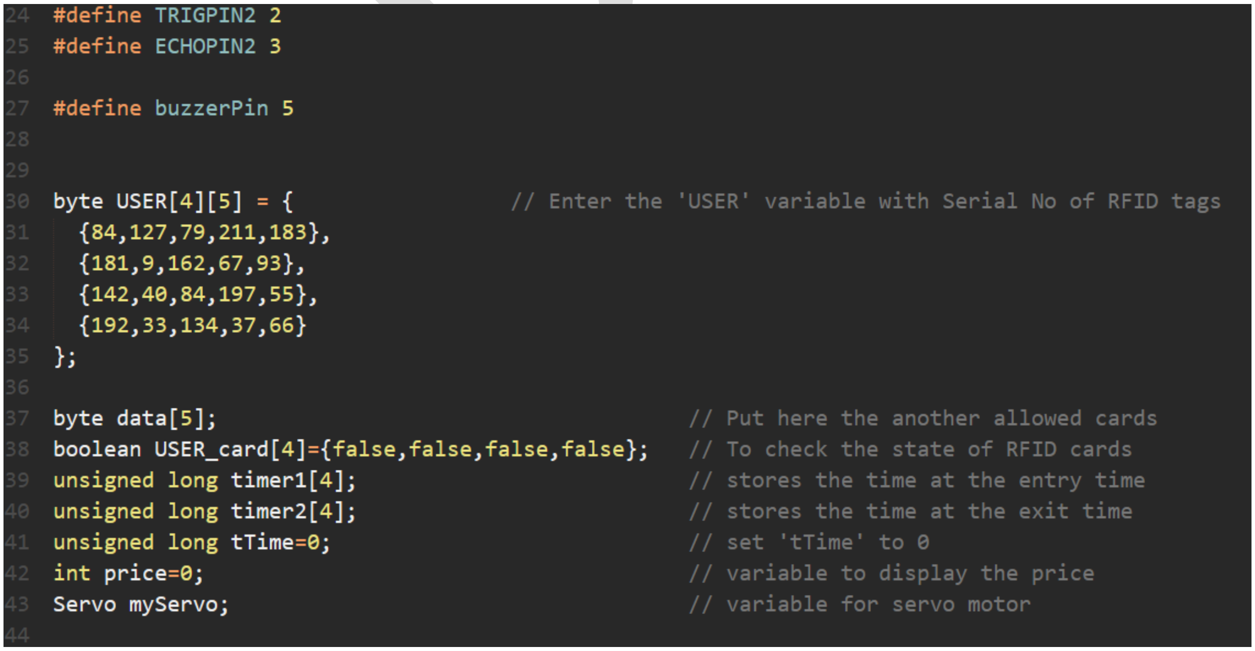
- Step 1: In order to establish connection between the client and the server, the Wi-Fi option in the Smartphone is enabled. However, it is also possible to extend the connection using any other means like Cellular Data.
- Step 2: Internet connectivity ensures that the Blynk app is able to communicate with NodeMCU over the server with its unique Auth Token.
- Step 3: Each sensor in the parking system is connected to the digital pins on the Arduino Board which in turn is connected to the Node MCU Board. This board will automatically connect to the Wi-Fi using the SSI D and Password details which is dumped into the board prior to the start of the operation.
- Step 4: A C-program is loaded on to the microprocessor chip on the Arduino Uno board that allows it collect sensor data.
- Step 5: A C-Program is loaded on the microprocessor chip on the Node MCU board that allows it to send data to the Android app using Blynk's servers.
- Step 6: The Blynk Android app enables the end user to monitor the various parking lots and the available slots I them.

6.2 Code

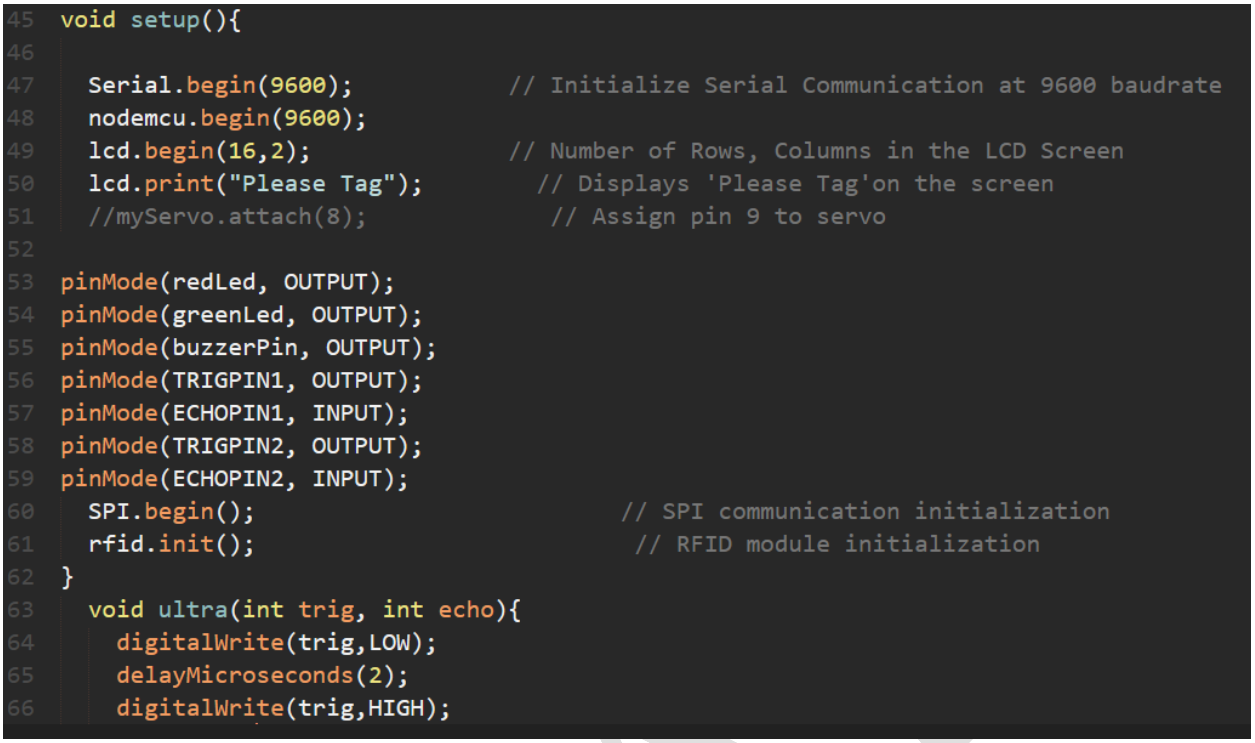
6.2.1 Arduino UN O Code



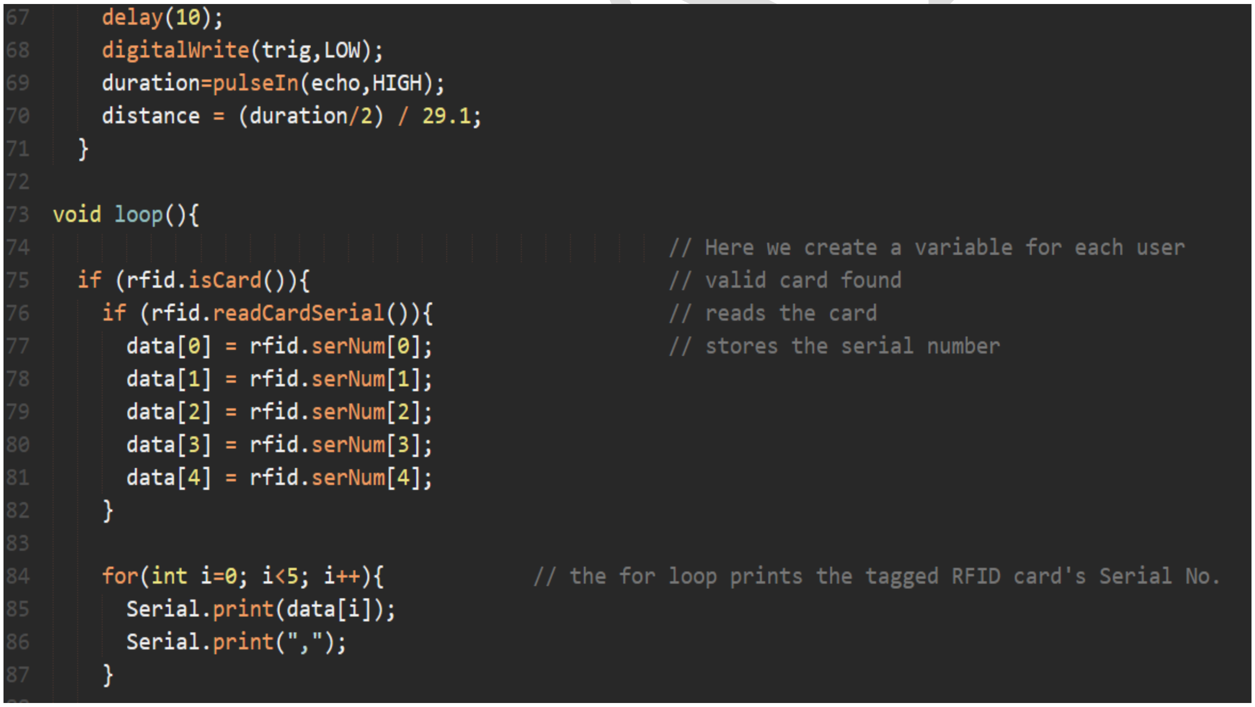
```

#include <SPI.h>                //Add 'SPI' library
#include <RFID.h>               //Add 'RFID' library
#include <Servo.h>              //Add 'Servo' library
#include "pitches.h"            //Add 'pitches' header file
#include<LiquidCrystal.h>       //Add 'LiquidCrystal' library
#include <SoftwareSerial.h>
SoftwareSerial nodemcu(0,1);



LiquidCrystal lcd(A5, A4, A3, A2, A1, A0);  //'lcd' is declared with the
                                              // pin configuration
RFID rfid(10,9);                              // RFID definition
constexpr uint8_t greenLed = 4;
constexpr uint8_t redLed = 6;
long duration,distance,slot1,slot2;
String sensor1;
String sensor2;
String cdata =""; // complete data, consisting of sensors values
#define TRIGPIN1 7
#define ECHOPIN1 8
```



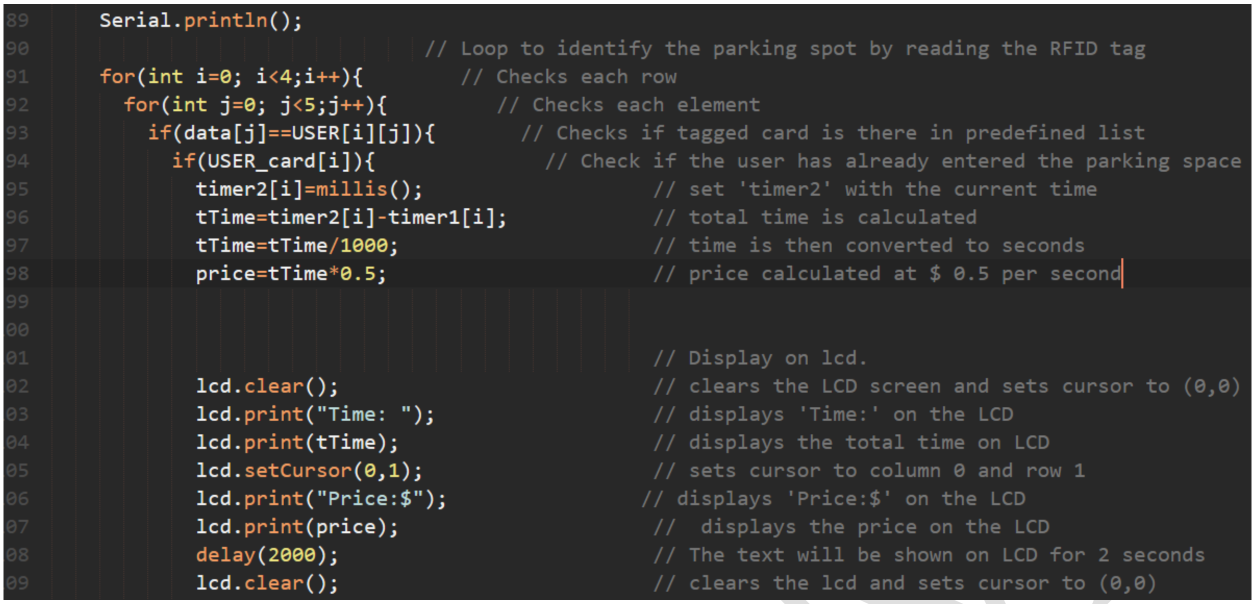
```
#define TRIGPIN2 2
#define ECHOPIN2 3

#define buzzerPin 5


byte USER[4][5] = {                   // Enter the 'USER' variable with Serial No of RFID tags
  {84,127,79,211,183},
  {181,9,162,67,93},
  {142,40,84,197,55},
  {192,33,134,37,66}
};

byte data[5];                                   // Put here the another allowed cards
boolean USER_card[4]={false,false,false,false};  // To check the state of RFID cards
unsigned long timer1[4];                        // stores the time at the entry time
unsigned long timer2[4];                        // stores the time at the exit time
unsigned long tTime=0;                          // set 'tTime' to 0
int price=0;                                    // variable to display the price
Servo myServo;                                  // variable for servo motor

```



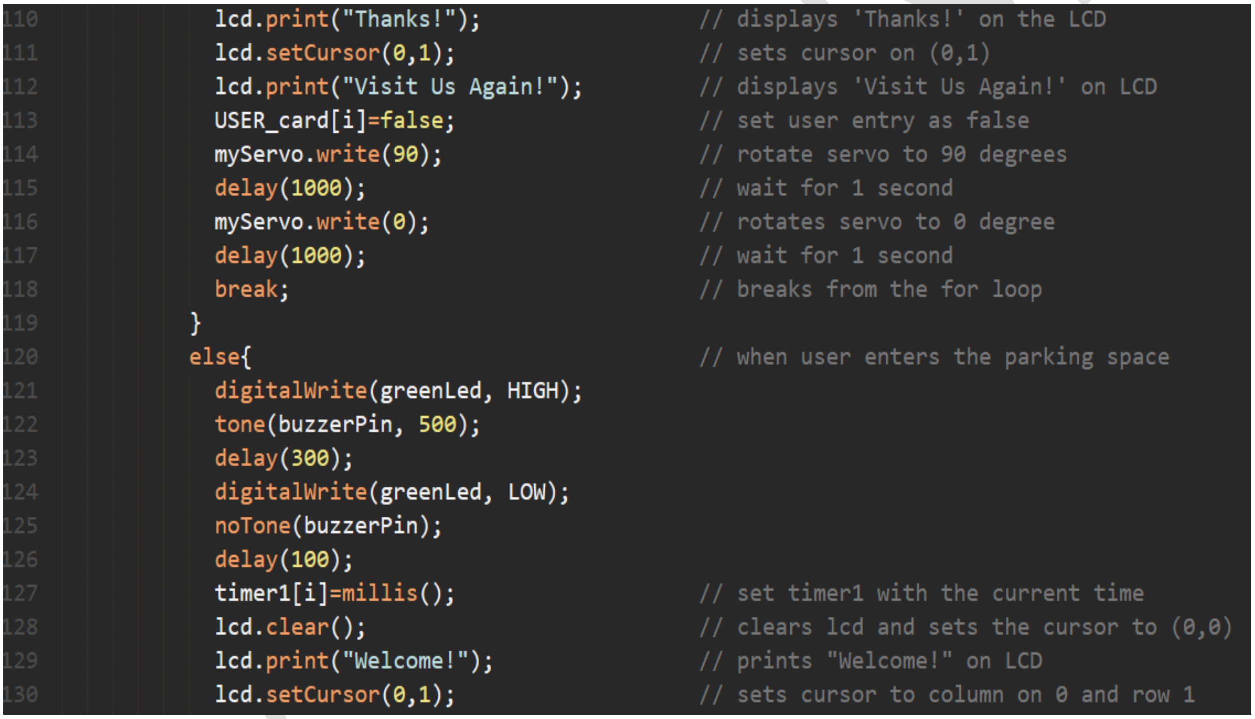
```
void setup(){

  Serial.begin(9600);          // Initialize Serial Communication at 9600 baudrate
  nodemcu.begin(9600);
  lcd.begin(16,2);             // Number of Rows, Columns in the LCD Screen
  lcd.print("Please Tag");       // Displays 'Please Tag'on the screen
  //myServo.attach(8);            // Assign pin 9 to servo

pinMode(redLed, OUTPUT);
pinMode(greenLed, OUTPUT);
pinMode(buzzerPin, OUTPUT);
pinMode(TRIGPIN1, OUTPUT);
pinMode(ECHOPIN1, INPUT);
pinMode(TRIGPIN2, OUTPUT);
pinMode(ECHOPIN2, INPUT);
  SPI.begin();                         // SPI communication initialization
  rfid.init();                          // RFID module initialization
}
  void ultra(int trig, int echo){
    digitalWrite(trig,LOW);
    delayMicroseconds(2);
    digitalWrite(trig,HIGH);
```



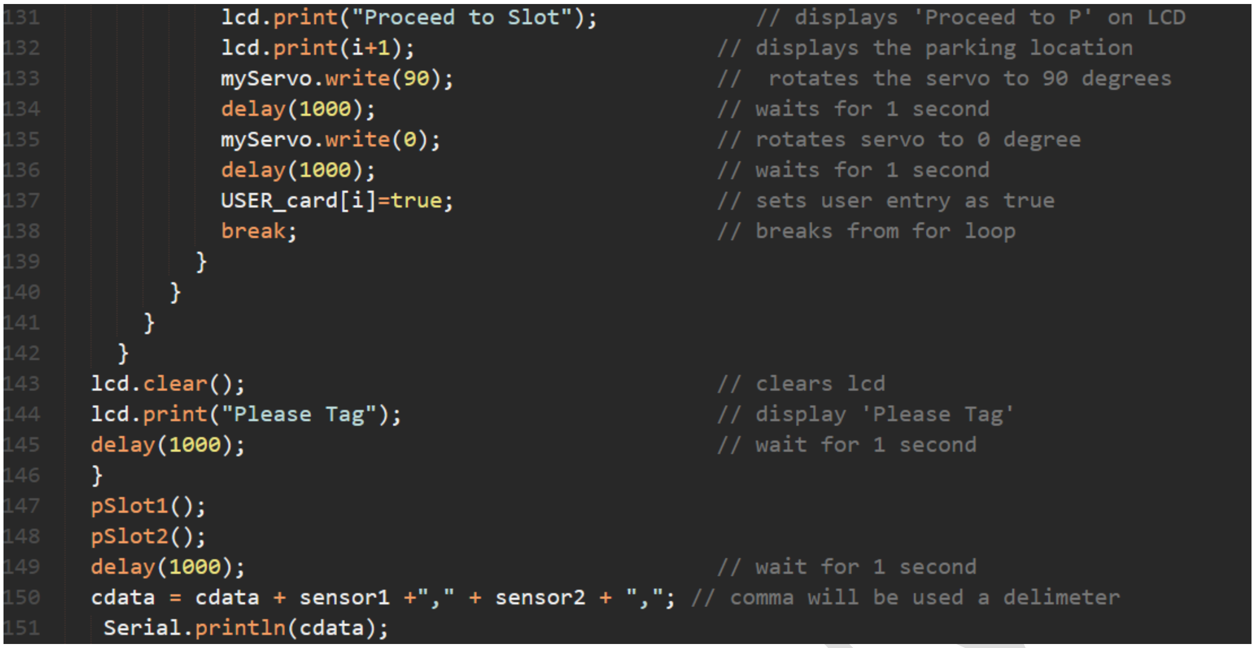
```
    delay(10);
    digitalWrite(trig,LOW);
    duration=pulseIn(echo,HIGH);
    distance = (duration/2) / 29.1;
  }

void loop(){
                                               // Here we create a variable for each user
  if (rfid.isCard()){                          // valid card found
    if (rfid.readCardSerial()){                // reads the card
      data[0] = rfid.serNum[0];                // stores the serial number
      data[1] = rfid.serNum[1];
      data[2] = rfid.serNum[2];
      data[3] = rfid.serNum[3];
      data[4] = rfid.serNum[4];
    }

    for(int i=0; i<5; i++){            // the for loop prints the tagged RFID card's Serial No.
      Serial.print(data[i]);
      Serial.print(",");
    }
```



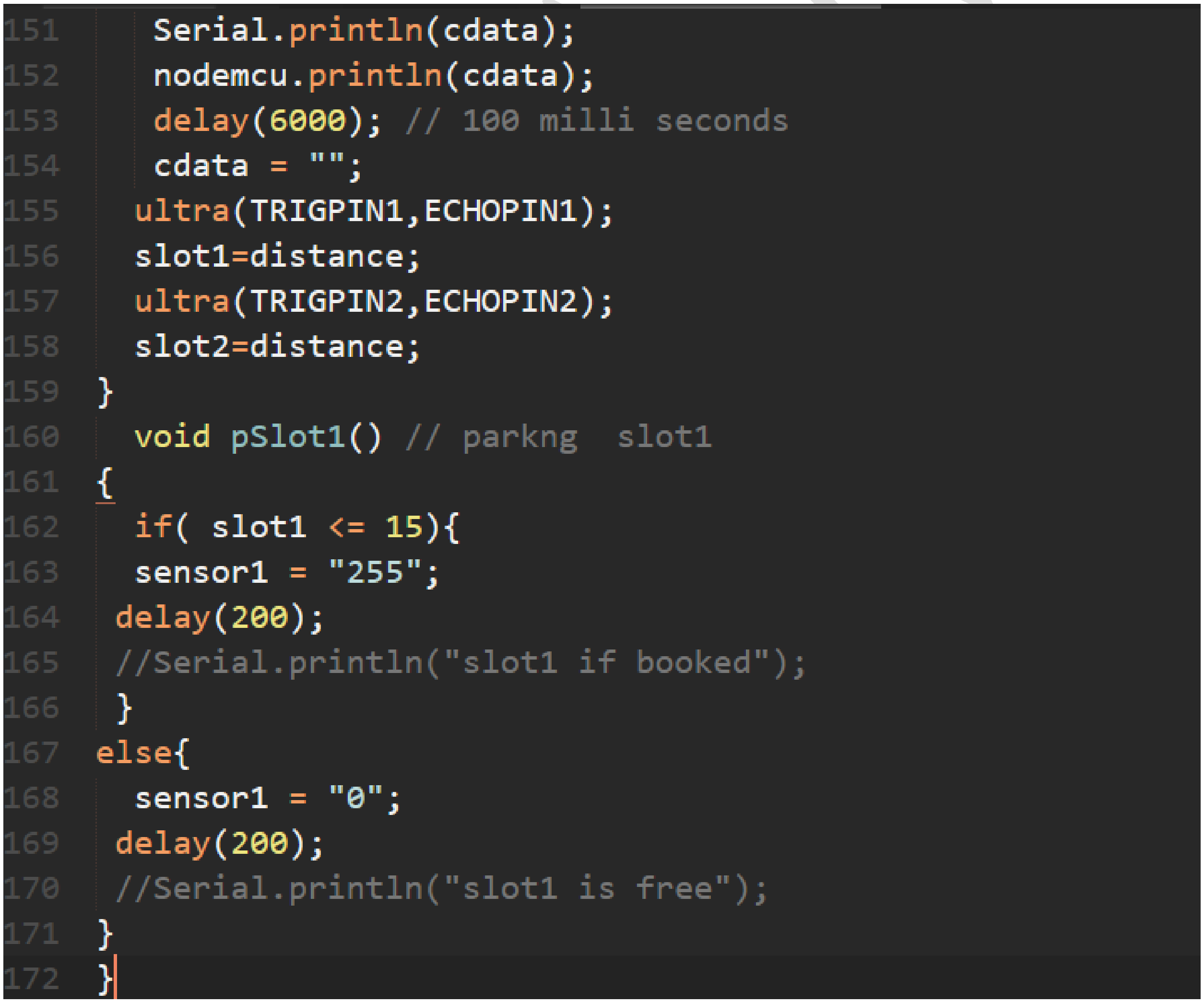
```
    Serial.println();
                               // Loop to identify the parking spot by reading the RFID tag
    for(int i=0; i<4;i++){        // Checks each row
      for(int j=0; j<5;j++){        // Checks each element
        if(data[j]==USER[i][j]){      // Checks if tagged card is there in predefined list
          if(USER_card[i]){             // Check if the user has already entered the parking space
            timer2[i]=millis();                   // set 'timer2' with the current time
            tTime=timer2[i]-timer1[i];            // total time is calculated
            tTime=tTime/1000;                     // time is then converted to seconds
            price=tTime*0.5;                      // price calculated at $ 0.5 per second


                                                  // Display on lcd.
            lcd.clear();                          // clears the LCD screen and sets cursor to (0,0)
            lcd.print("Time: ");                  // displays 'Time:' on the LCD
            lcd.print(tTime);                     // displays the total time on LCD
            lcd.setCursor(0,1);                   // sets cursor to column 0 and row 1
            lcd.print("Price:$");                // displays 'Price:$' on the LCD
            lcd.print(price);                     //  displays the price on the LCD
            delay(2000);                          // The text will be shown on LCD for 2 seconds
            lcd.clear();                          // clears the lcd and sets cursor to (0,0)
```



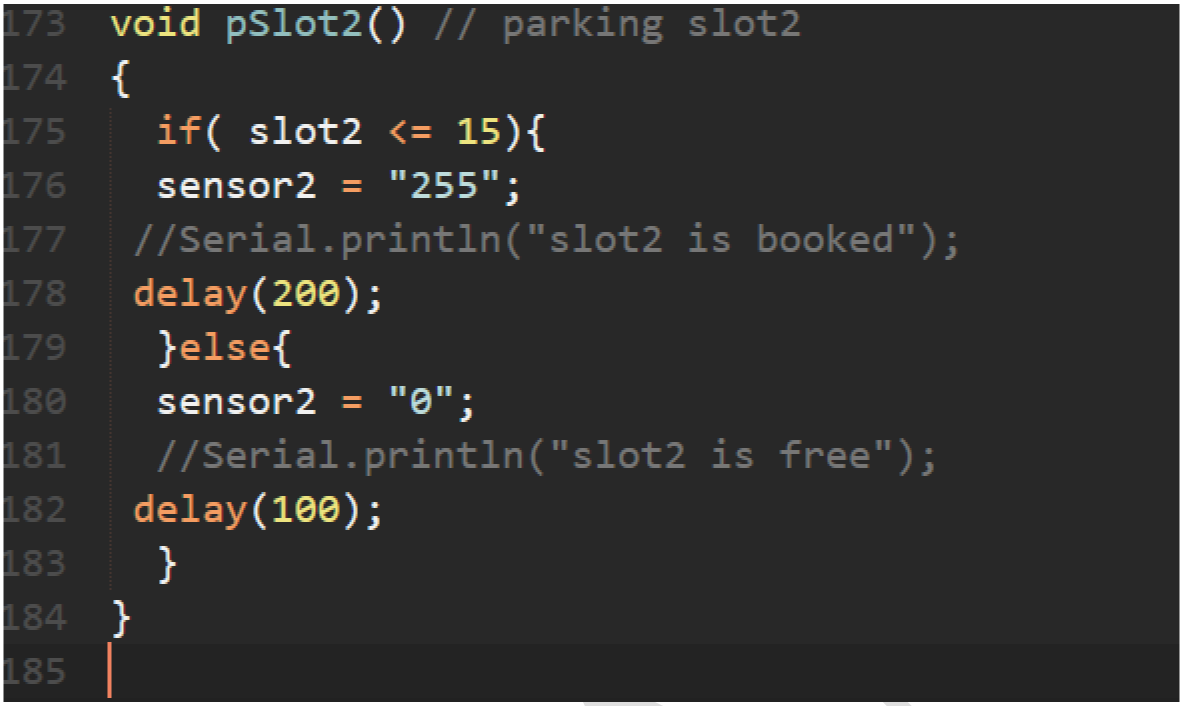
```
            lcd.print("Thanks!");                 // displays 'Thanks!' on the LCD
            lcd.setCursor(0,1);                   // sets cursor on (0,1)
            lcd.print("Visit Us Again!");         // displays 'Visit Us Again!' on LCD
            USER_card[i]=false;                   // set user entry as false
            myServo.write(90);                    // rotate servo to 90 degrees
            delay(1000);                          // wait for 1 second
            myServo.write(0);                     // rotates servo to 0 degree
            delay(1000);                          // wait for 1 second
            break;                                // breaks from the for loop
          }
          else{                                   // when user enters the parking space
            digitalWrite(greenLed, HIGH);
            tone(buzzerPin, 500);
            delay(300);
            digitalWrite(greenLed, LOW);
            noTone(buzzerPin);
            delay(100);
            timer1[i]=millis();                   // set timer1 with the current time
            lcd.clear();                          // clears lcd and sets the cursor to (0,0)
            lcd.print("Welcome!");                // prints "Welcome!" on LCD
            lcd.setCursor(0,1);                   // sets cursor to column on 0 and row 1
```



```
          lcd.print("Proceed to Slot");           // displays 'Proceed to P' on LCD
          lcd.print(i+1);                      // displays the parking location
          myServo.write(90);                   //  rotates the servo to 90 degrees
          delay(1000);                         // waits for 1 second
          myServo.write(0);                    // rotates servo to 0 degree
          delay(1000);                         // waits for 1 second
          USER_card[i]=true;                   // sets user entry as true
          break;                               // breaks from for loop
        }
      }
    }
  }
lcd.clear();                                   // clears lcd
lcd.print("Please Tag");                       // display 'Please Tag'
delay(1000);                                   // wait for 1 second
}
pSlot1();
pSlot2();
delay(1000);                                   // wait for 1 second
cdata = cdata + sensor1 +"," + sensor2 + ","; // comma will be used a delimeter
 Serial.println(cdata);
```



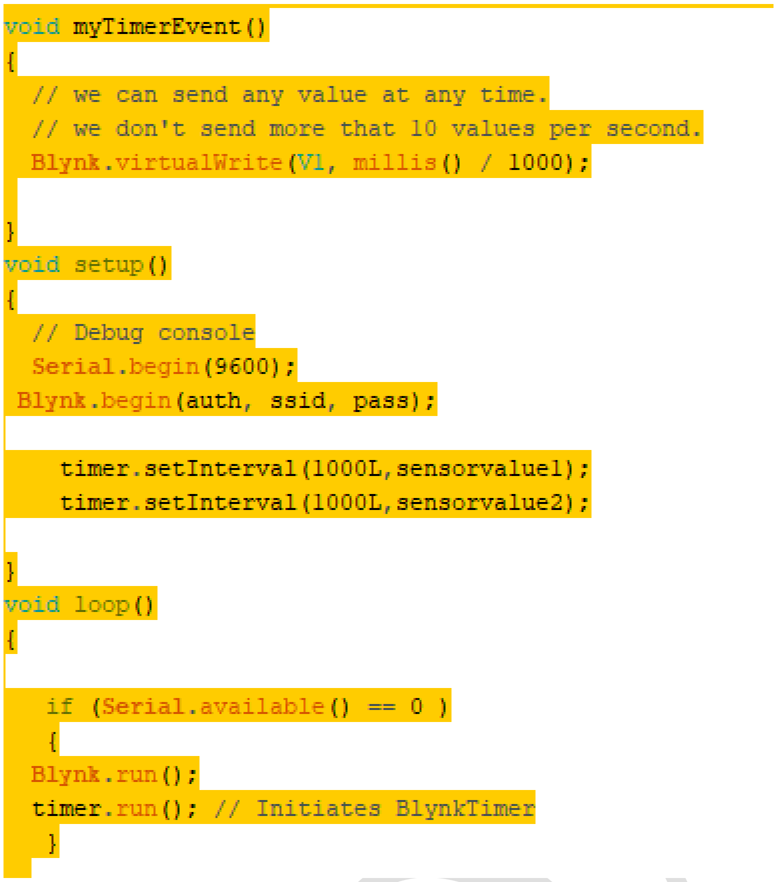
```
   Serial.println(cdata);
   nodemcu.println(cdata);
   delay(6000); // 100 milli seconds
   cdata = "";
  ultra(TRIGPIN1,ECHOPIN1);
  slot1=distance;
  ultra(TRIGPIN2,ECHOPIN2);
  slot2=distance;
}
  void pSlot1() // parkng  slot1
{
  if( slot1 <= 15){
  sensor1 = "255";
 delay(200);
 //Serial.println("slot1 if booked");
 }
else{
  sensor1 = "0";
 delay(200);
 //Serial.println("slot1 is free");
}
}
```



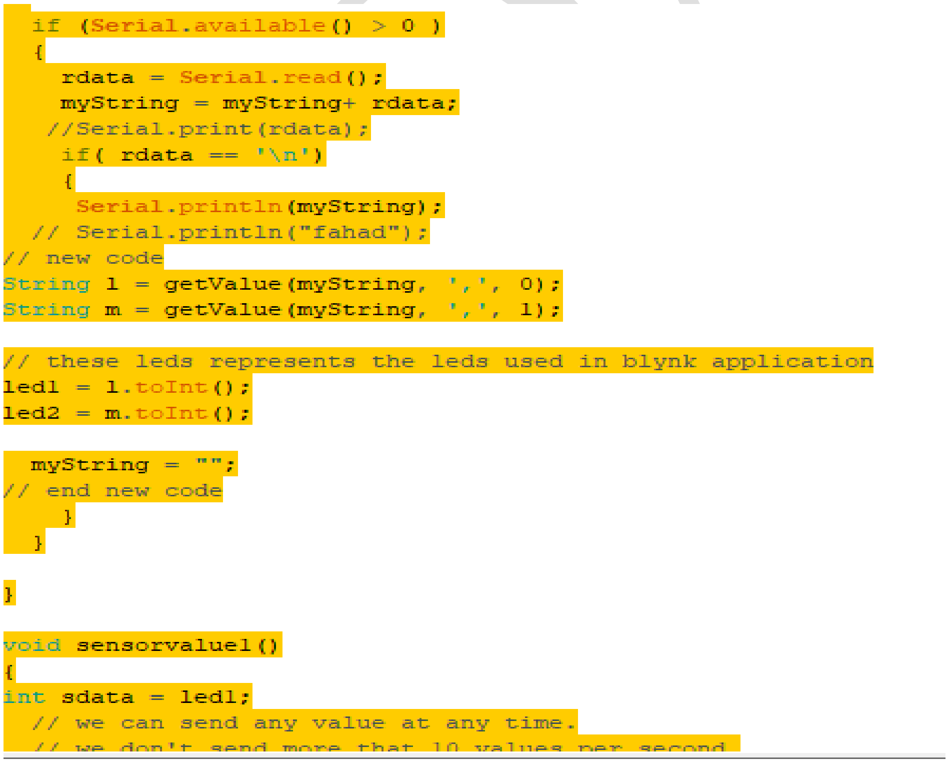
```
void pSlot2() // parking slot2
{
  if( slot2 <= 15){
  sensor2 = "255";
 //Serial.println("slot2 is booked");
 delay(200);
  }else{
  sensor2 = "0";
  //Serial.println("slot2 is free");
 delay(100);
  }
}
```

6.2.2 Node MCU Code:

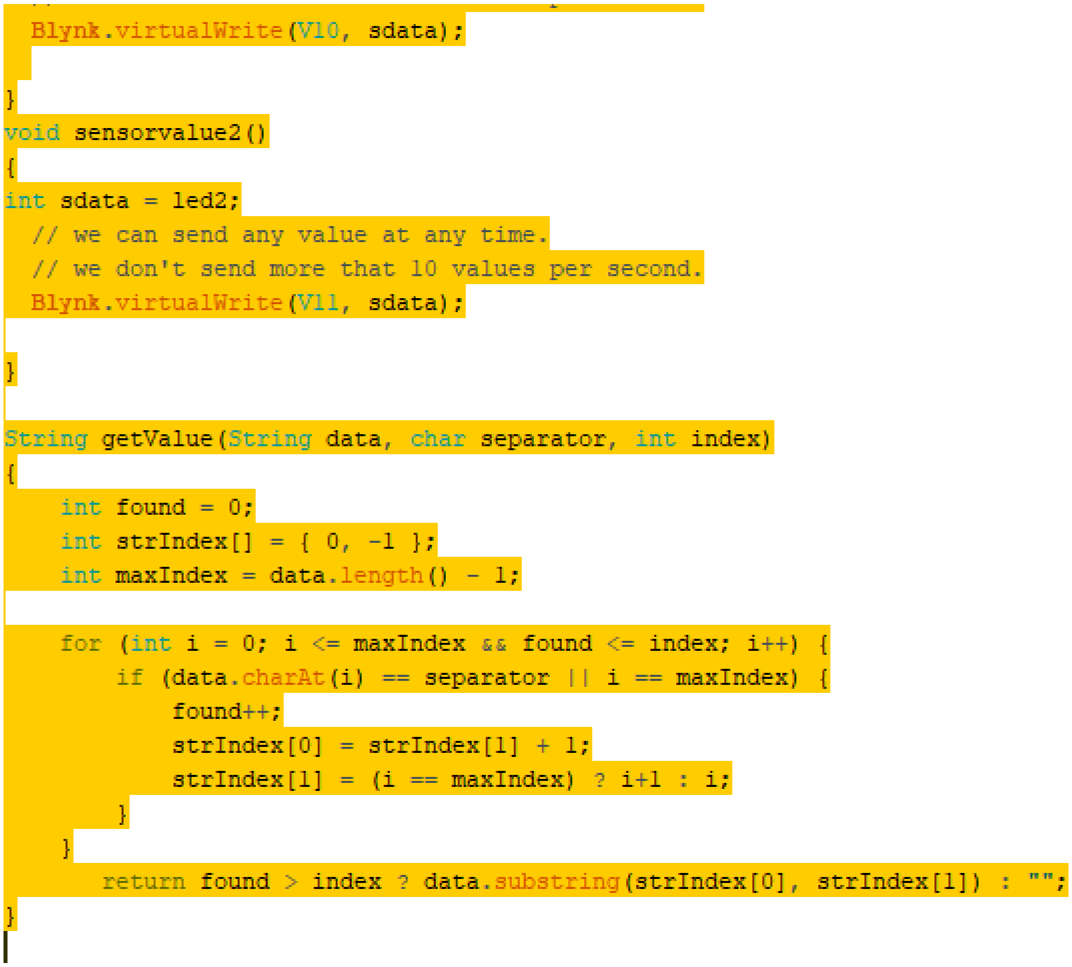
```
#define BLYNK_PRINT Serial


#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <SoftwareSerial.h>
#include <SimpleTimer.h>

char auth[] = " pKsfpdQNmictHTx45_PhHx_ISfTNiVb-";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "doctor nourdine";
char pass[] = "mhandani269";

SimpleTimer timer;

String myString; // complete message from arduino, which consistors of snesors data
char rdata; // received charactors

int led1,led2;
// This function sends Arduino's up time every second to Virtual Pin (1).
// In the app, Widget's reading frequency should be set to PUSH. This means
// that you define how often to send data to Blynk App.
```



```
void myTimerEvent()
{
  // we can send any value at any time.
  // we don't send more that 10 values per second.
  Blynk.virtualWrite(V1, millis() / 1000);

}
void setup()
{
  // Debug console
  Serial.begin(9600);
 Blynk.begin(auth, ssid, pass);

    timer.setInterval(1000L,sensorvalue1);
    timer.setInterval(1000L,sensorvalue2);

}
void loop()
{

   if (Serial.available() == 0 )
   {
  Blynk.run();
  timer.run(); // Initiates BlynkTimer
   }
```



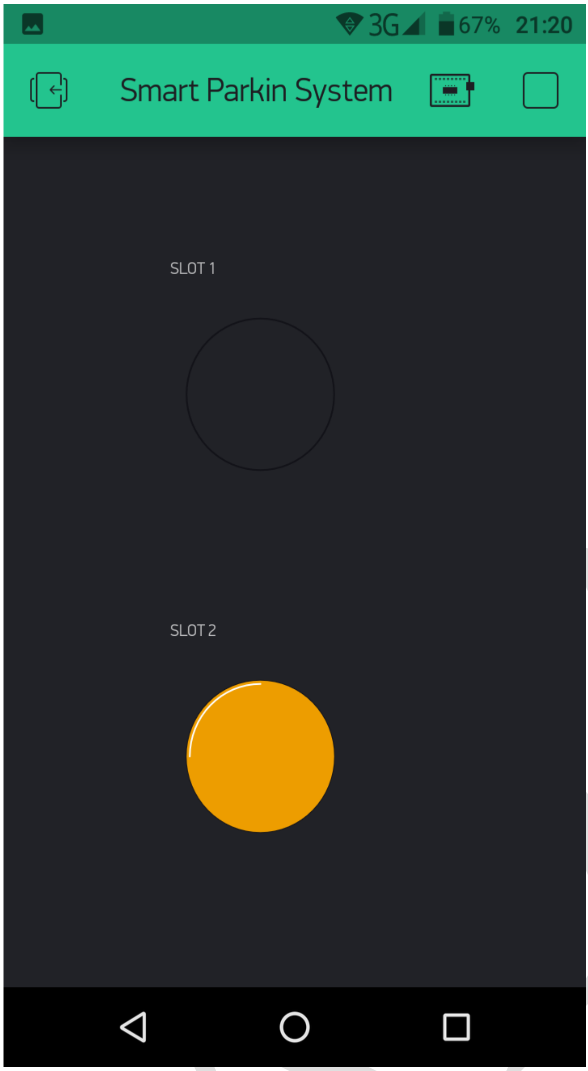
```
  if (Serial.available() > 0 )
  {
    rdata = Serial.read();
    myString = myString+ rdata;
   //Serial.print(rdata);
    if( rdata == '\n')
    {
     Serial.println(myString);
  // Serial.println("fahad");
// new code
String l = getValue(myString, ',', 0);
String m = getValue(myString, ',', 1);

// these leds represents the leds used in blynk application
led1 = l.toInt();
led2 = m.toInt();

  myString = "";
// end new code
    }
  }

}

void sensorvalue1()
{
int sdata = led1;
  // we can send any value at any time.
  // we don't send more that 10 values per second
```



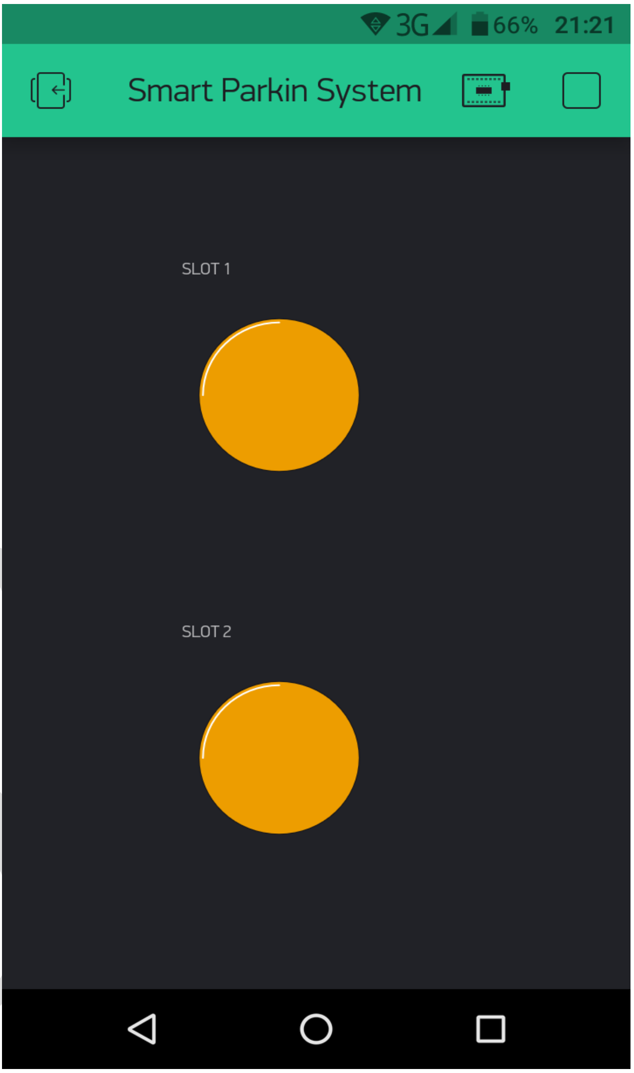
```
  Blynk.virtualWrite(V10, sdata);

}
void sensorvalue2()
{
int sdata = led2;
  // we can send any value at any time.
  // we don't send more that 10 values per second.
  Blynk.virtualWrite(V11, sdata);

}

String getValue(String data, char separator, int index)
{
    int found = 0;
    int strIndex[] = { 0, -1 };
    int maxIndex = data.length() - 1;

    for (int i = 0; i <= maxIndex && found <= index; i++) {
        if (data.charAt(i) == separator || i == maxIndex) {
            found++;
            strIndex[0] = strIndex[1] + 1;
            strIndex[1] = (i == maxIndex) ? i+1 : i;
        }
    }
       return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```

6.2.3 Output:

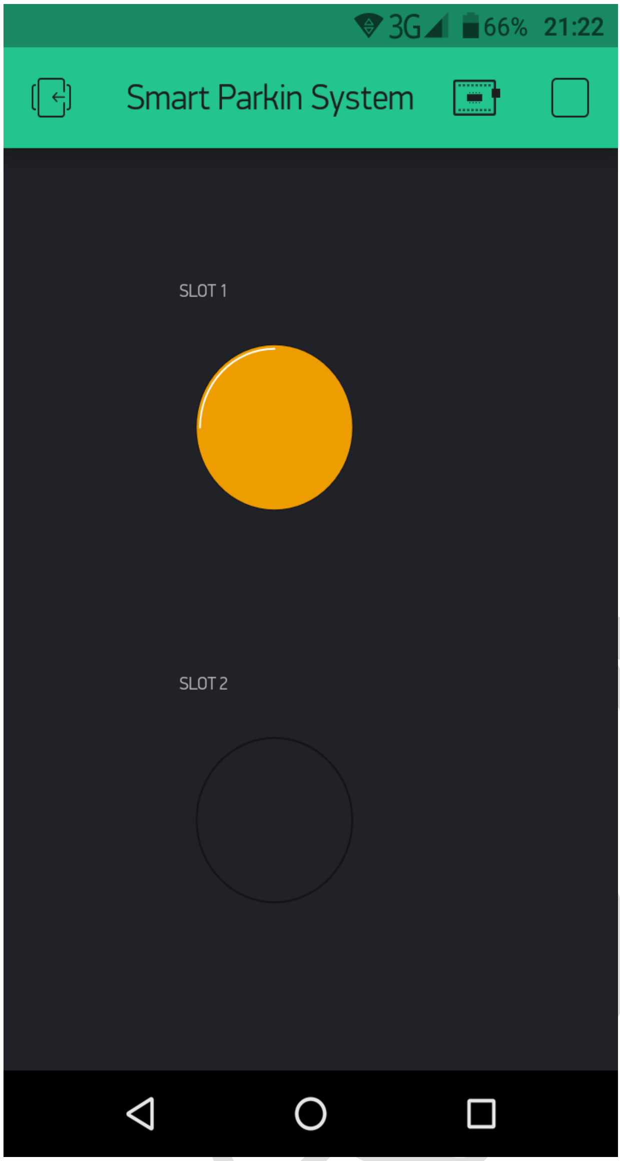
When car entered slot 2:



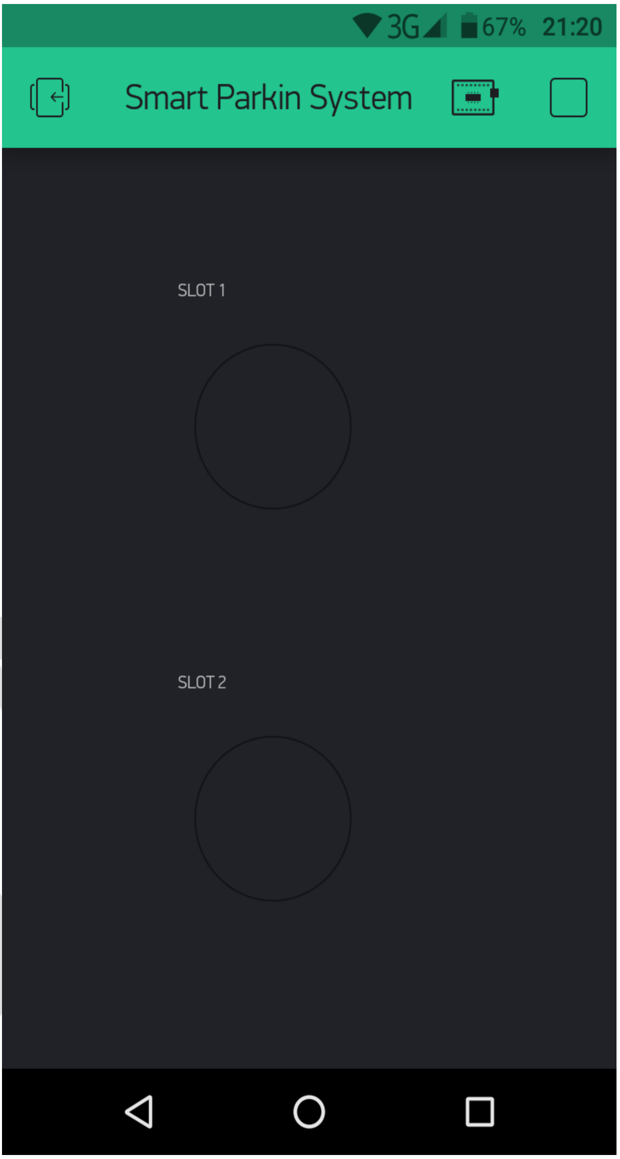
Later, when car entered



When car from slot 2 exits



When car from slot 1 also exits



7 CONCLUSION

With the ever-increasing volume of traffic, it will be crucial to implement smart parking system world-wide to reduce the hassle and save the time it takes to find a parking lot.

The proposed system can further be developed in future to enable billing & payments and an integration with Maps that allows users to pre-book his nearest parking lot and reach there as quickly as possible.