IOT WORKSHOP
EIECC19

Smart Parking

Mini Project Documentation

Submitted By:

2020UEI2804 Sahil Kumar Yadav

2020UEI2827 Mukesh Kumar Mandal

Submitted To:

Prof. Dhananjay V. Gadre

Contents

1 Acknowledgement

We would like to express our sincere gratitude towards Professor Dhananjay V. Gadre for his remarkable guidance, monitoring, and constant encouragement throughout the project.

We would also like to thank our friends who supported us greatly and were always willing to help us.

2 Introduction

The project aims to develop a smart parking system that utilizes inductive loops and IoT technology to optimize parking availability and improve user convenience. The system comprises several components, including an Inductive Loop detector, Colpitts Oscillator circuit, ESP32 for communication, Arduino Nano for frequency detection, LM-7808 Voltage Regulator, and Bidirectional I2C Logic Level Converter from 5V to 3.3V.

The system includes a Colpitts oscillator circuit that is designed to detect the presence of a vehicle by measuring changes in the frequency of oscillation. The electrical properties of an inductor can be influenced by the presence of a large metallic body, such as that of a vehicle. By measuring these changes in the properties, we can detect the presence of a vehicle. Any other non-metallic body would not show up as a vehicle in this system. Different values of capacitors are used to tune the oscillator circuit for optimal performance.

The inductive loop detector is embedded beneath the parking lot surface and shows a deflection of up to 20KHz. The frequency changes are then measured by an Arduino Nano board, which sends the parking status to an ESP32 board for communication with the cloud server and updates parking availability in real-time. Overall, our smart parking system offers a scalable and efficient solution to parking management problems.

3 Six-Box Model

Let us first list down the requirements of the proposed project as follows:

- An Inductive Loop to detect changes in frequency caused by the presence of a vehicle.
- An Oscillator Circuit to generate a stable sine wave.
- A cloud platform to display the analyzed data.
- A microcontroller to measure the frequency of the oscillator.
- A communication channel between the microcontroller and cloud platform.
- A supply to power up the whole circuit.

Now we will try to place the above requirements in the six-box model:

1. INPUT: An Inductive Loop to detect the presence of a vehicle created using a 10-core FRC Cable with a Copper core. The physical size of the inductor is important and should be about 4 feet in length and 4 feet in width. Ten turns of the copper loop give us an inductance of about 500μH to 1000μH which when combined with 2 capacitors C1 and C2 of 0.1μF gives frequency in the range of 20kHz to 60 kHz, which can be easily detected by an Arduino Nano.

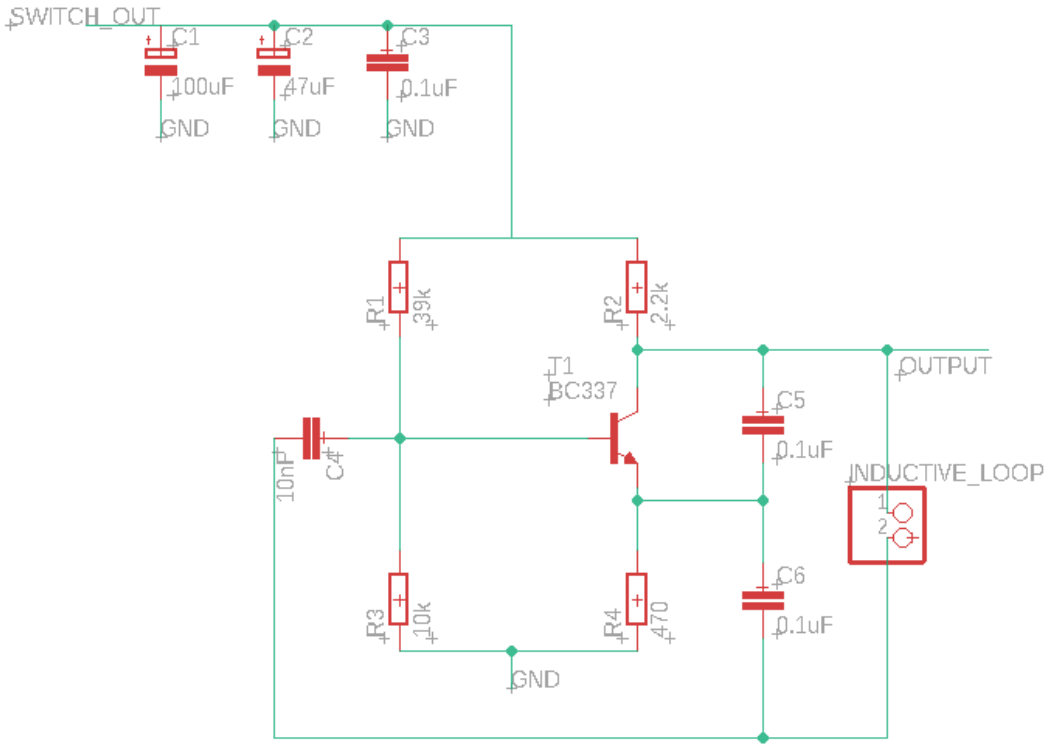
(a) FRC Cable

(b)Inductive Loop

Figure 1: Inductive Loop for detecting a vehicle

2. OSCILLATOR: The inductive loop created above forms the detector of vehicles. When metallic material approaches the center of the inductor (the detector coil), it enters the magnetic field created by the inductor. This changes the magnetic permeability of the inductor's core, causing the inductance to change. The change in inductance, in turn, changes the oscillating frequency of the Colpitts Oscillator circuit. The Colpitts Oscillator provides us with sustained oscillations at a frequency that can be measured by a microcontroller.

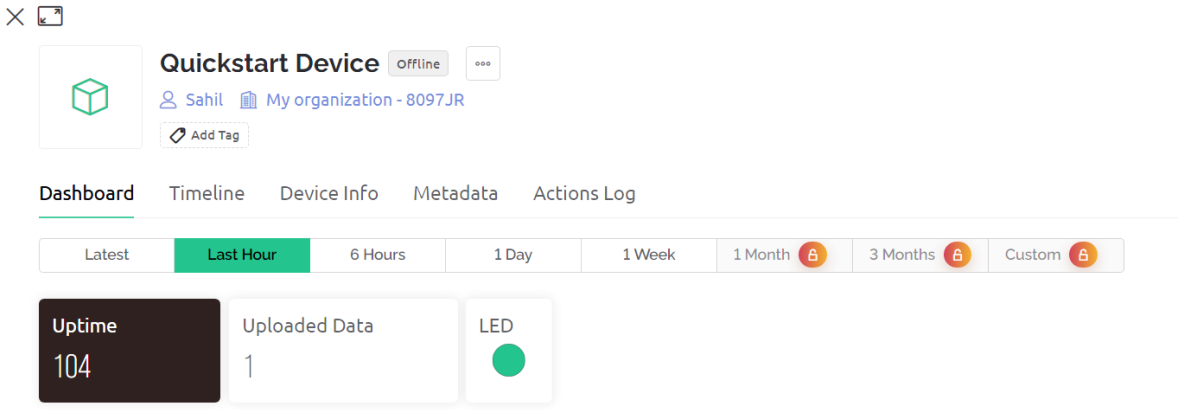
(a) Colpitts Oscillator



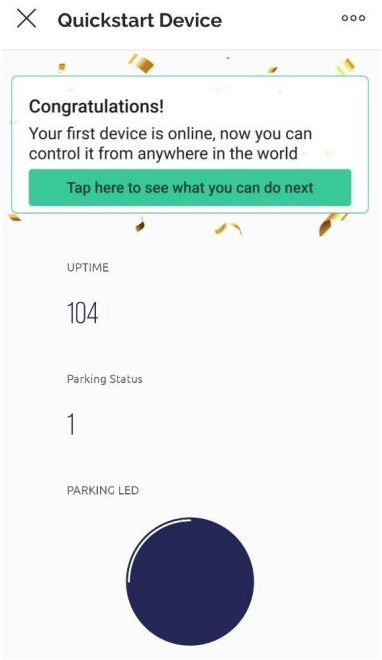
(b)Schematic

Figure 2: Colpitts Oscillator for generating a sine wave

3. OUTPUT: The output of the Smart Parking project is shown on an IoT platform where the data collected from the circuit is analyzed and visualized for the user. In this case, the IoT platform is Blynk. The uploaded data is shown in the form of a label and also in the form of an LED.



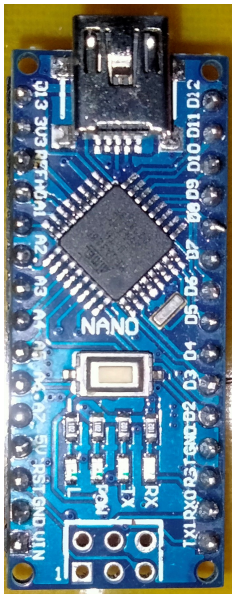
(a) Blynk Dashboard



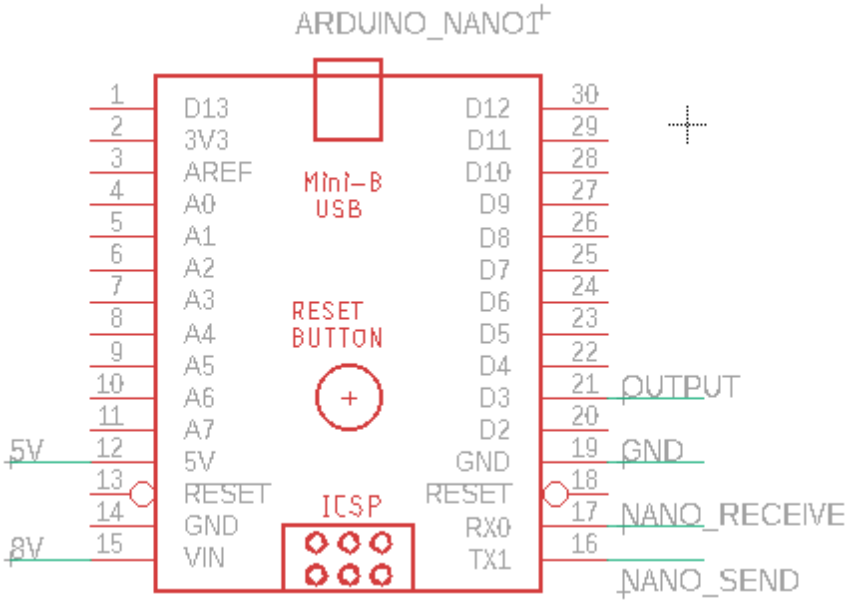
(b) Blynk App

Figure 3: Blynk Cloud Platform to access uploaded data

4. MICRO CONTROLLER: The microcontroller used to measure the frequency of the Colpitts Oscillator is Arduino Nano. It'll constantly detect the oscillator frequency and when the frequency goes above a certain threshold, the presence of a vehicle is detected. The data is then sent to ESP32 over the Serial line.



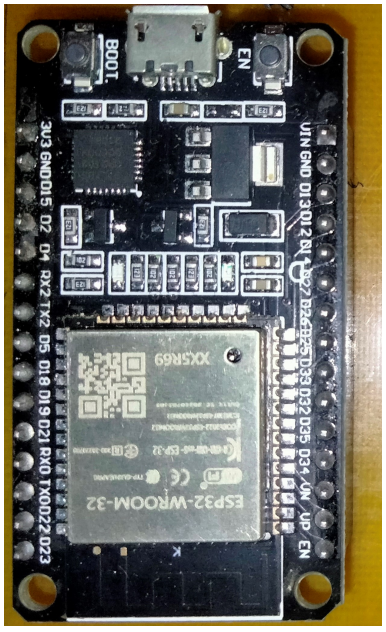
(a) Arduino Nano Microcontroller



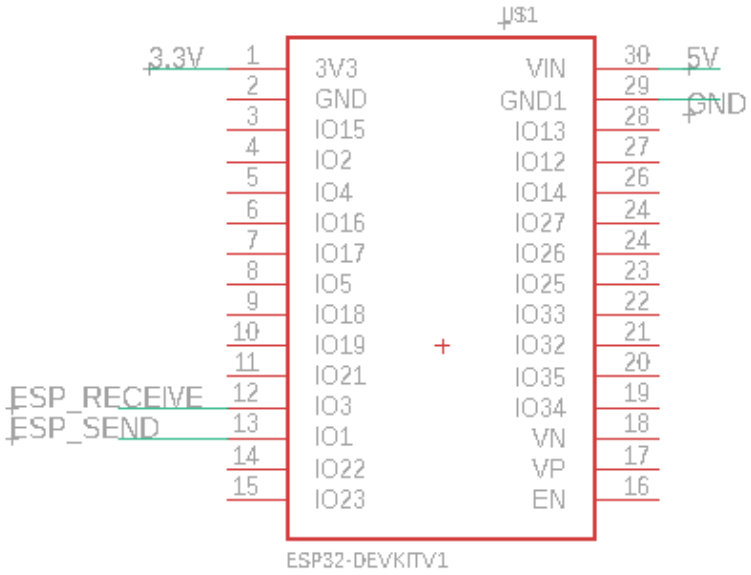
(b) Schematic

Figure 4: Arduino Nano for frequency detection

5. COMMUNICATION: The communication between Arduino Nano and the Cloud is provided through the use of the ESP32 microcontroller. It sends the data received over its Serial port to the Blynk IoT Cloud using the onboard Wifi module.



(a) ESP32 Microcontroller



(b) Schematic

Figure 5: ESP32 for cloud communication

6. POWER SUPPLY: The power supply to the whole circuit is a 12V Lithium-ion rechargeable battery. The Arduino Nano microcontroller is powered using a constant 8V regulated voltage obtained using the LM7808 Voltage Regulator. The 5V output pin from Arduino Nano is used to power the ESP32 microcontroller. The communication between Arduino Nano and ESP32 is done through the use of a Bidirectional I2C Logic Level Converter 5V to 3.3V to prevent damaging the ESP32 from the Arduino Nano's high voltage.

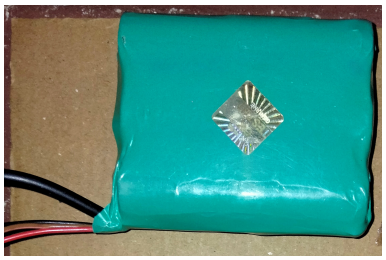
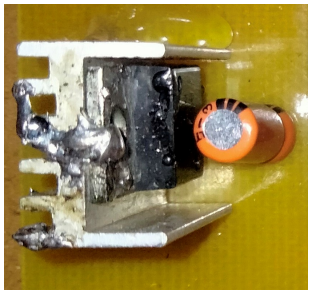
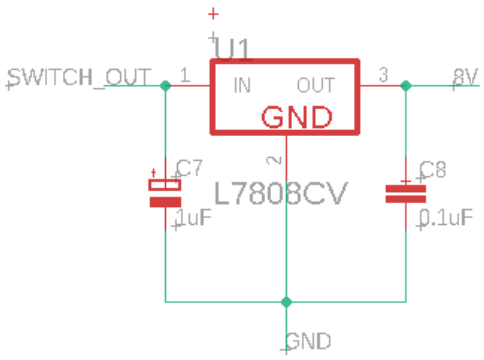


Figure 6: 12V Lithium-ion Rechargeable Battery

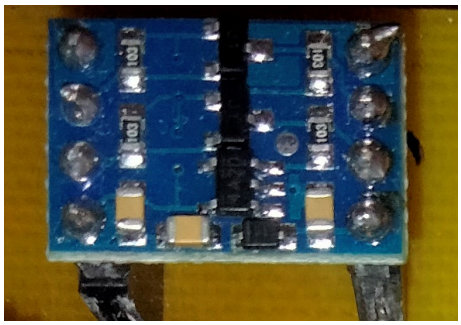


(a) LM7808 Voltage Regulator

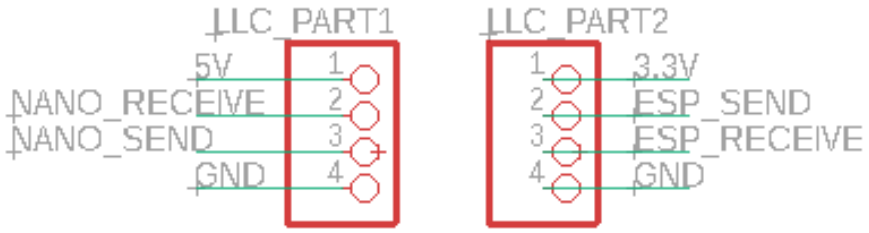


(b) Schematic

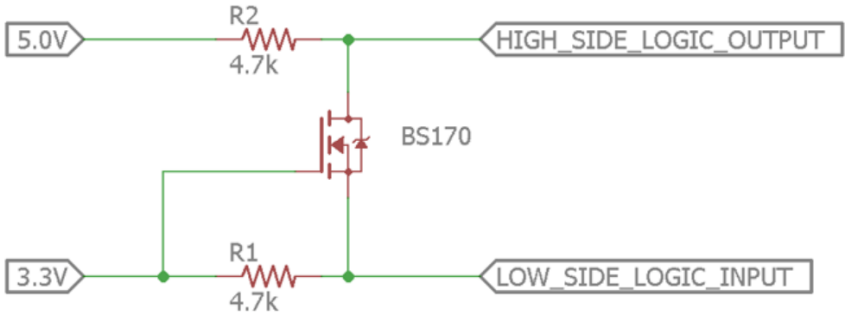
Figure 7: LM7808 for 8V output to power Arduino Nano



(a) Bidirectional I2C Logic Level Converter 5V to 3.3V



(b) Schematic



(c) Circuit

Figure 8: Bidirectional Logic Level Converter for communication between Arduino Nano and ESP32

4 Project Description

4.1 Block Diagram

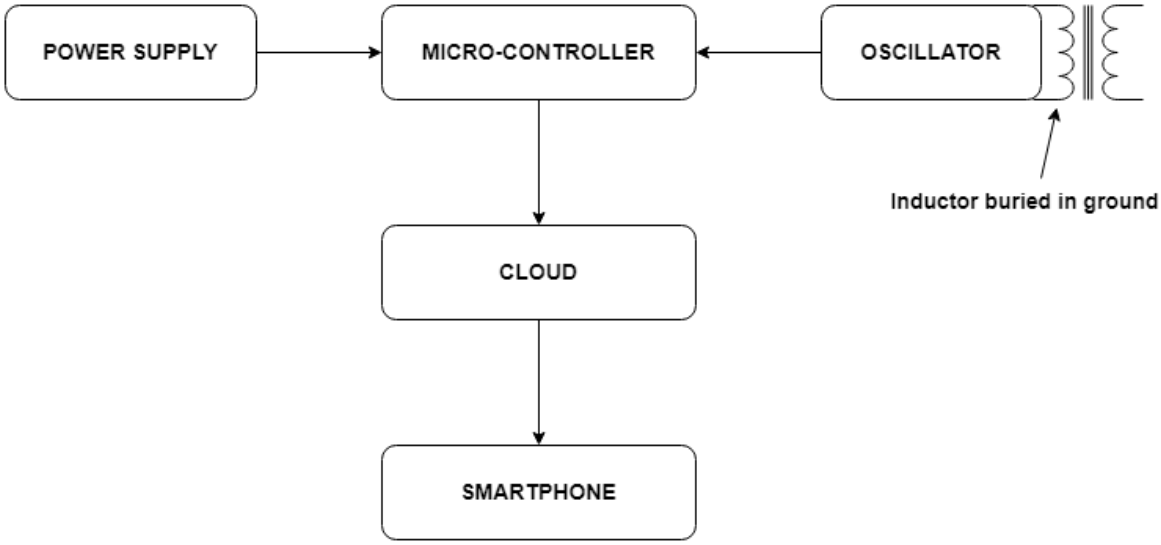


Figure 9: Smart Parking Block Diagram

4.2 Hardware Description

The Smart Parking System is made up of several hardware components that work together to detect the presence of a vehicle in a parking spot and communicate this information to the cloud server in real-time.

The 12V Battery output is filtered through 4 coupling capacitors of values 100µF, 47µF, 0.1µF, and 1µF to remove noise and voltage fluctuations.

A Common Emitter amplifier circuit with a feedback path is used to create sustained oscillations in the Colpitts oscillator. The Inductive Loop is embedded beneath the parking lot surface and is designed to detect the presence of a vehicle. When a vehicle enters the inductive loop, the loop's inductance changes, which causes a change in the oscillator circuit's frequency.

The Arduino Nano board is responsible for measuring the frequency changes and sending this data to the ESP32 board for communication with the cloud. The LM7808 Voltage Regulator provides a constant regulated supply of 8V to power the Arduino Nano microcontroller.

The ESP32 module is powered using the 5V output pin from Arduino Nano. It communicates with the cloud server to update parking availability in real-time.

The communication between Arduino Nano and ESP32 is interfaced using the Bidirectional I2C Logic Level Converter which converts the 5V Serial data from Arduino Nano to 3.3V Serial data for the ESP32.

4.3 Software Description

The software for the smart parking system based on an inductive loop primarily involves programming the Arduino Nano and ESP32 using the Arduino IDE to perform their respective functions.

For the Arduino Nano, the software is responsible for detecting the frequency changes in the inductive loop caused by the presence of a vehicle. This is achieved by programming the Arduino to read the frequency using the PulseIn function. An average value of 1000 continuous frequencies is used to reduce fluctuations in the detected frequency. If the frequency is below a certain threshold, a string "Empty" to denote the vacant parking spot is sent to ESP32 over the Serial Port. Although if the frequency is above the threshold frequency, a string "Parked" to denote an occupied parking spot is sent to ESP32 over the Serial Port.

The ESP32 board is responsible for communication with the cloud server to update parking availability in real time. Upon receiving data on Serial Port from the Arduino Nano, the ESP32 sets a 'park-status' variable depending on the received string. If the received string is "Empty", 'park-status' is set to 0, if the received string is "Parked", 'park-status' is set to 1, else 'park-status' is set to 8. The value 8 is used for error detection and debugging purposes. The ESP32 then sends this 'park-status' variable to the Blynk IoT Cloud where we can visualize the change in the parking status using available widgets.

5 Circuit Design

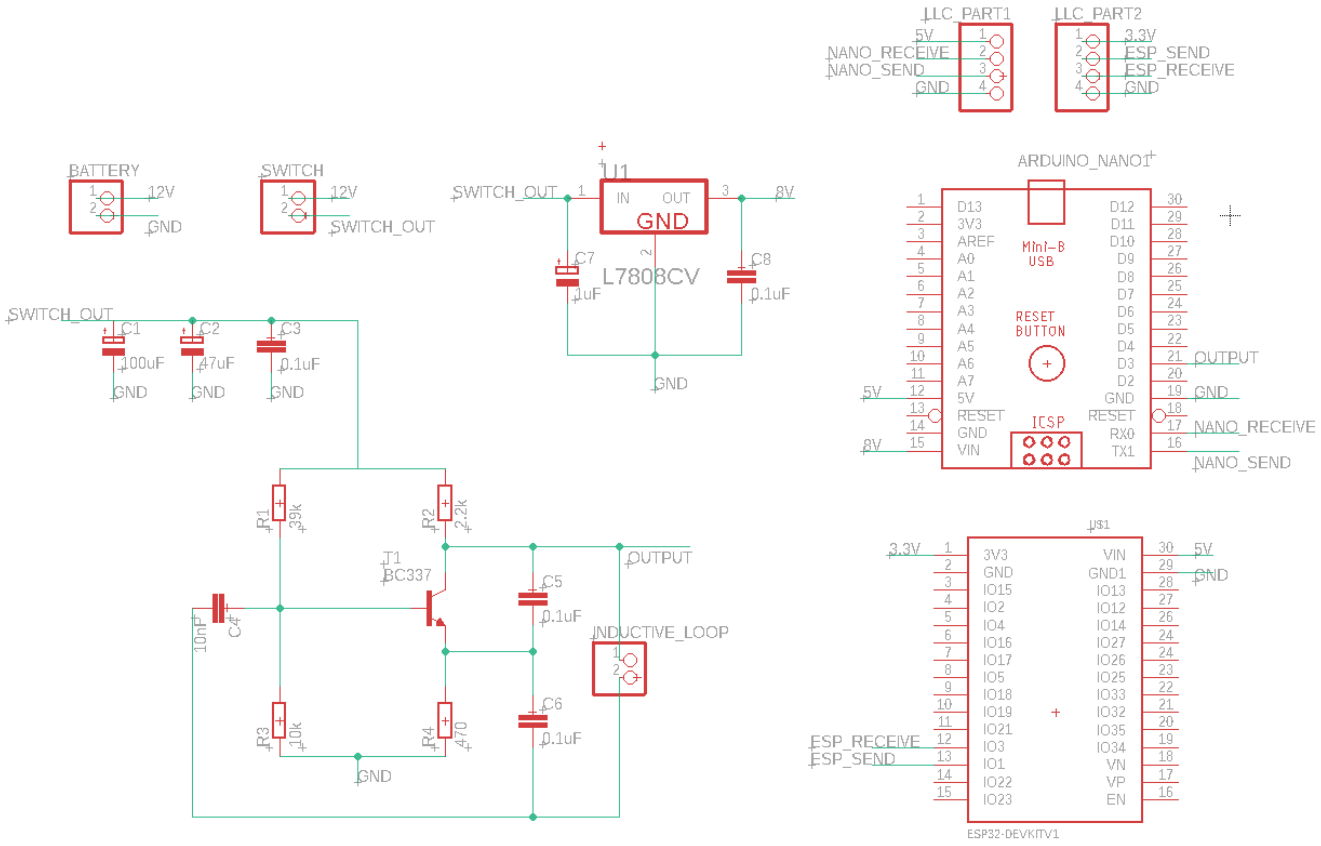


Figure 10: Schematic for Smart Parking

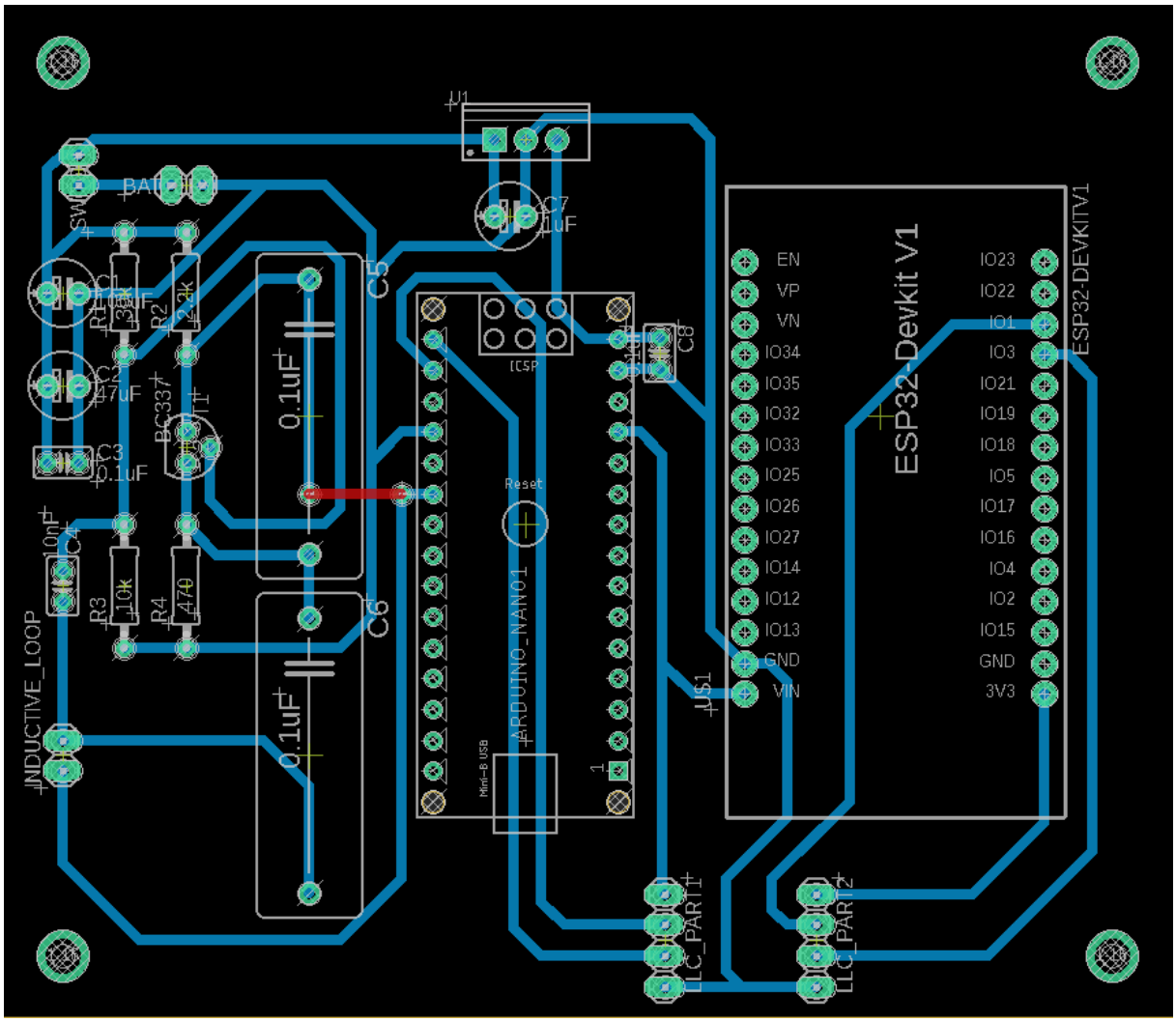


Figure 11: PCB Layout for Smart Parking

6 Code

6.1 Arduino Code

```
int Htime;   //integer for storing high time
int Ltime;   //integer for storing low time
float Ttime; // integer for storing total time of a cycle
float frequency; //storing frequency
float avg_frequency;
String park_string;
void setup()
{
  pinMode(3,INPUT);
  Serial.begin(9600);
}
void loop()
{
  avg_frequency=0;
  for(int i=0; i<1000; i++)
  {
    Htime=pulseIn(3,HIGH); //read high time
    Ltime=pulseIn(3,LOW);  //read low time
    Ttime = Htime+Ltime;
    frequency=1000000/Ttime;
    avg_frequency=avg_frequency+frequency;
  }
  avg_frequency=avg_frequency/1000;
  // Serial.println("Frequency of signal");
  // Serial.print(avg_frequency);
  // Serial.println(" Hz");
  if(avg_frequency<42000)
  {
    park_string="Empty";
  } else {
    park_string="Parked";
  }
  Serial.println(park_string);
  delay(2000);
}
```

6.2 ESP32 Code

```
1
2 /* Fill-in information from Blynk Device Info here */
3 #define BLYNK_TEMPLATE_ID           "TMPL3H-oy1mU2"
4 #define BLYNK_TEMPLATE_NAME         "Quickstart␣Template"
5 #define BLYNK_AUTH_TOKEN            "
    KbAZB0Fz3tsz65TL7JS1SCkeHgmuKHBo"
6
7 /* Comment this out to disable prints and save space */
8 #define BLYNK_PRINT Serial
9
10 // My variables for Serial Transmission
11 #define RXp 3
12 #define TXp 1
13 String park_string;
14 int park_status;
15
16 #include <WiFi.h>
17 #include <WiFiClient.h>
18 #include <BlynkSimpleEsp32.h>
19
20 // Your WiFi credentials.
21 // Set password to "" for open networks.
22 char ssid[] = "iQOO_9T_Sahil";
23 char pass[] = "nhibtarha";
24
25 BlynkTimer timer;
26
27 // This function is called every time the Virtual Pin 0 state
     changes
28 BLYNK_WRITE(V0)
29 {
30   // Set incoming value from pin V0 to a variable
31   int value = param.asInt();
32
33   // Update state
34   Blynk.virtualWrite(V1, value);
35 }
36
37 // This function is called every time the device is connected
     to the Blynk.Cloud
38 BLYNK_CONNECTED()
```

```
39 {
40   // Change Web Link Button message to "Congratulations!"
41   Blynk.setProperty(V3, "offImageUrl", "https://static-image.
       nyc3.cdn.digitaloceanspaces.com/general/fte/
       congratulations.png");
42   Blynk.setProperty(V3, "onImageUrl",  "https://static-image.
       nyc3.cdn.digitaloceanspaces.com/general/fte/
       congratulations_pressed.png");
43   Blynk.setProperty(V3, "url", "https://docs.blynk.io/en/
       getting-started/what-do-i-need-to-blynk/how-quickstart-
       device-was-made");
44 }
45
46 // This function sends Arduino's uptime every second to
     Virtual Pin 2.
47 void myTimerEvent()
48 {
49   // You can send any value at any time.
50   // Please don't send more that 10 values per second.
51   Blynk.virtualWrite(V2, millis() / 1000);
52 }
53
54 void setup()
55 {
56   Serial.begin(115200);
57   Serial2.begin(9600, SERIAL_8N1, RXp, TXp);
58   Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
59
60   // Setup a function to be called every second
61   timer.setInterval(1000L, myTimerEvent);
62 }
63
64 void loop()
65 {
66   Blynk.run();
67   timer.run();
68
69   park_string=Serial2.readString();
70   park_string.trim();
71   Serial.println(park_string);
72   if(park_string=="Empty")
73   {
74     park_status=0;
```

```
  }
  else if(park_string=="Parked")
  {
    park_status=1;
  }
  else
  {
    park_status=8;
  }
  Serial.println(park_status);
  Blynk.virtualWrite(V1,park_status);
  Blynk.virtualWrite(V1,park_status);
}
```

7 References

1. tinyAVR Microcontroller Projects for the Evil Genius, Dhananjay V. Gadre & Nehul Malhotra

2. https://www.ijrte.org/wp-content/uploads/papers/v7i4s/E1996017519.pdf

3. https://www.researchgate.net/publication/332875468_Real_Time_Smart_Car_Parking_System_Using_Internet_of_Things

4. https://iopscience.iop.org/article/10.1088/1742-6596/1424/1/012021/pdf

5. https://blog.getmyparking.com/2019/02/14/issues-with-parking-in-indian-metropolises