TPJ Technical Report

SPOT SAVY SMART PARKING SYSTEM

GROUP D6

Group Member 1: **Krusha Vipulkumar Mistry**

Group Member 2: **Tony Braxton Tchio Ngoumeza**

Course Code: **TPJ655 NDDL**

Instructors: **Rakesh Kantariaony and Iakov Romanovski**

TPJ Technical Report

The "Spot Savy Smart Parking System" addresses urban parking challenges through advanced technology. With increasing vehicle numbers in cities, efficient parking management is crucial for reducing congestion and emissions. This report details the system's design and implementation, leveraging sensors, real-time data, and user-friendly interfaces to optimize parking. By providing features such as spot reservation and efficient space utilization, the system aims to enhance user experience and environmental sustainability. The study underscores the potential for smart parking solutions to significantly lower greenhouse gas emissions and improve city infrastructure. Through real-time monitoring and intelligent management, Spot Savy contributes to smarter, eco-friendly urban environments.

TPJ Technical Report

Table of Contents

Table of Figures

The "Spot Savy Smart Parking System" is an innovative solution designed to address the increasing challenges of urban parking. In fast growing cities with a rise in the number of vehicles, efficient parking management becomes crucial to reduce congestion, lower emissions, and enhance urban living. Li and Jasmine's (2021) study on why the implementation of smart parking would leads to the growth of smarter cities, elaborates on how the different tools that are available from the implementation of smart parking systems could help with the day-to-day struggle of vehicle drivers in crowded cities. This report presents the development and implementation of the Spot Savy Smart Parking System, a comprehensive approach utilizing advanced technologies to streamline the parking experience. Our aim is to provide options such as spot reservation, easy access and a friendly user interface which will help make parking a less stressful experience for vehicle drivers.

The parking industry has a long history and has undergone significant evolution. To enhance this sector, extensive research was conducted to assess its current state, identify existing challenges, and explore potential improvements. According to Köln auf dem Weg Zur's study (as cited in Geoffrey Garnier's 2015-2024) on what the future of smart parking is, approximately 900,000 tons of greenhouse gas emission could be avoided through more smart parking systems implementation. According to INRIX's study (as cited in Geoffrey Garnier's 2015-2024) a typical vehicle driver spends 17hours on average looking for a parking space over the course of a year. By addressing these issues, the industry can better meet the needs of modern society and accommodate future advancements. The primary objective of this system is to enhance the utilization of parking spaces in urban areas through real-time monitoring and intelligent management. Utilizing an IoT platform, the system gathers and analyzes data on parking availability. A user-friendly web application allows users to access this information, helping them find available parking spaces quickly and efficiently. This approach not only optimizes parking space usage but also aims to reduce traffic congestion and lower vehicle emissions.

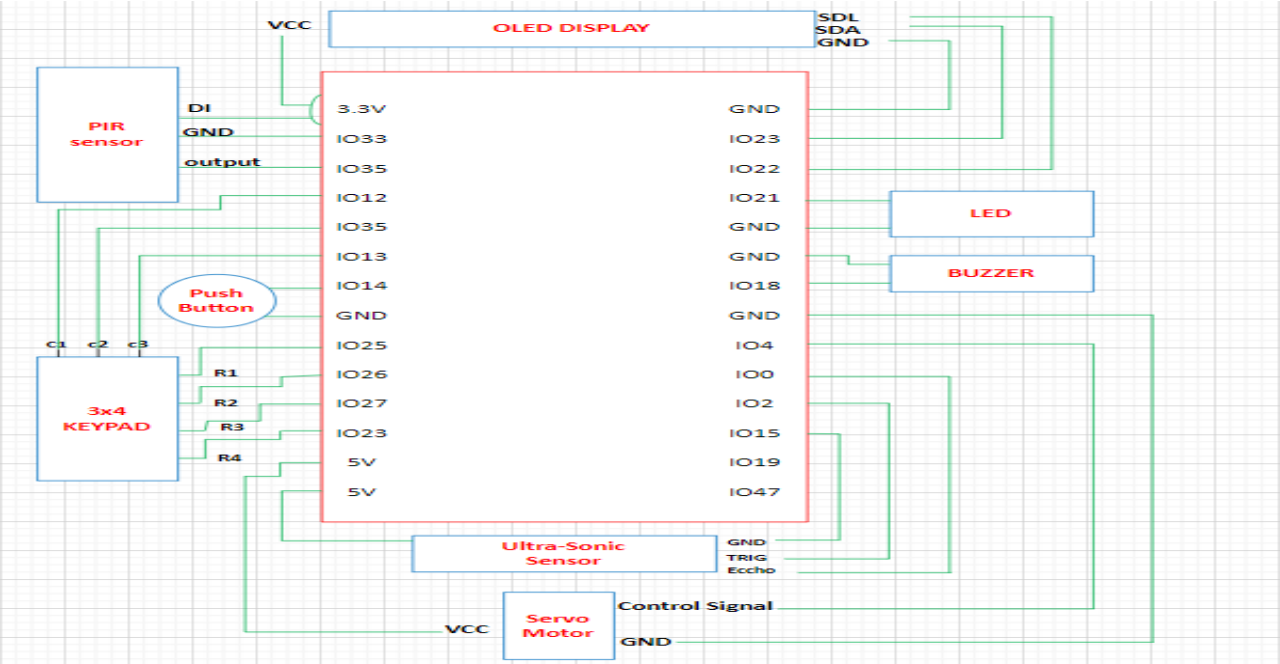


Figure 1: Mechanical Diagram-1

From the above mechanical diagram, we can see the system uses a combination of sensors, LEDs, a buzzer, a servo motor, an OLED display, a keypad, a push button, and an ESP32 microcontroller. This combination ensures efficient monitoring, control, and user interface for the smart system.

For the purpose of our project, we decided to use the

following list of input and output devices.

Input Devices

The input devices include sensors, a keypad, and a push button, which provide data and user interaction.

Figure 2 :Ultrasonic sensor image

Robocraze. (n.d.). What is Ultrasonic Sensor? Robocraze. Retrieved June 21, 2024, from https://robocraze.com/blogs/post/what-is-ultrasonic-sensor

*****Ultrasonic Sensors (HC-SR04): Detect approaching cars to automatically open the gate.

Figure 3:Infrared sensor image

Amazon India. (n.d.). REES52 Infrared Obstacle Avoidance Sensor Module for Arduino. Amazon. Retrieved June 21, 2024, from https://www.amazon.in/REES52-Infrared-Obstacle-Avoidance-Arduino/dp/B0731XLRBL

*IR Sensors (HC-SR501): Ensure the gate remains open until the vehicle fully passes through.

Figure 4: Keypd image

PiShop. (n.d.). Membrane 3x4 matrix keypad. PiShop. Retrieved June 21, 2024, from https://www.pishop.ca/product/membrane-3x4-matrix-keypad-extras-3x4/

*Keypad (3x4 KPSM): Allows drivers to enter their pre-booked slot number.

Figure 5:camera module on ESP32 CAM

Amazon Canada. (n.d.). CANADUINO ESP32-CAM ESP32 CAM MB Programming Adapter. Amazon. Retrieved June 21, 2024, from https://www.amazon.ca/CANADUINO-ESP32-CAM-ESP32-CAM-MB-Programming-Adapter/dp/B08XYLSH15

*Camera Module (Integrated to ESP32): Provides real-time visual monitoring and verification.

Figure 6: Push button image

Amazon Canada. (n.d.). QTEATAK 100pcs Momentary Tactile Button Switch. Amazon. Retrieved June 21, 2024, from https://www.amazon.ca/QTEATAK-Momentary-Tactile-Button-Switch/dp/B07VSNN9S2

*Push Buttons (DAOKI-6x6x5): Activates the buzzer in case of an emergency.

Output Devices

The output devices, such as LEDs, a buzzer, a servo motor, and an OLED display, deliver feedback and visual information.

Figure 7: OLED image

Rajguru Electronics. (n.d.). 0.91 inch OLED display 12832 LCD display device white colour. Rajguru Electronics. Retrieved June 21, 2024, from https://rajguruelectronics.com/ProductView?product=0.91%20inch%20OLED%20display%2012832%20LCD%20display%20device%20whitecolour&tokDatRef=MzkyNg==&tokenId=MQ==

*****OLED Display (1814-FA2): Displays instructions and relevant information at the entrance.

Figure 8: LEDs image

Walmart. (n.d.). 100pcs 5mm Red Green Round LED Diode Lights Electronic Component Emitting Light. Walmart Canada. Retrieved June 21, 2024, from https://www.walmart.ca/en/ip/100pcs-5mm-Red-Green-Round-LED-Diode-Lights-Electronic-Component-Emitting-Light/PRD6L0YSDEMZFH5

*LEDs: Indicate the status of parking spots; green for available and red for occupied.

Figure 9: Buzzer image

PCB Board. (n.d.). Mini Piezo Buzzer. PCB Board Canada. Retrieved June 21, 2024, from https://www.pcboard.ca/minipiezo-buzzer

*Buzzer (2223-CMI-1295IC-0585T-ND): Alerts in emergency situations when the push button is pressed.

Figure 10: Servo Motor image

PiShop. (n.d.). SG90 180 degrees 9g micro servo motor - Tower Pro. PiShop. Retrieved June 21, 2024, from https://www.pishop.ca/product/sg90-180-degrees-9g-micro-servo-motor-tower-pro/

*****Servo Motor (SG90): Operates the gate, opening it upon detecting a car and ensuring it stays open until the IR sensor confirms the vehicle has passed.

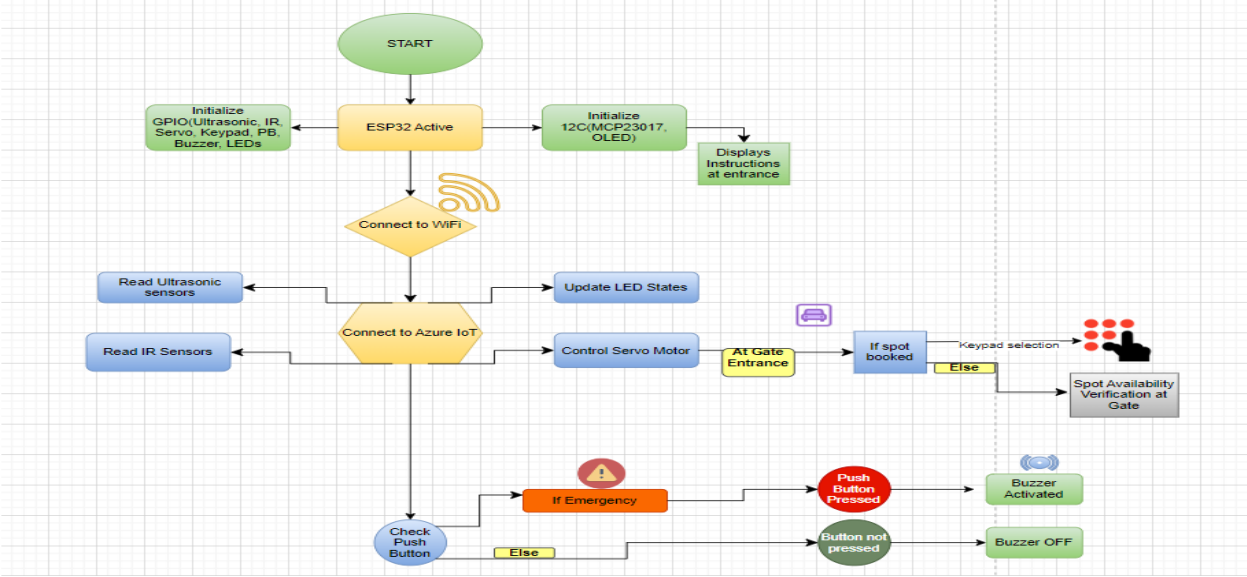


Figure 11: Flowchart

The Azure IoT platform and web application coordinate the inputs and outputs. Data from the sensors, camera, and keypad is sent to Azure IoT, which processes it to manage parking operations. The web application allows users to book slots and interact with the system. When a booking is made, the information is sent to the ESP32 microcontroller via Azure IoT. The ESP32 then controls the output devices based on real-time data and booking details, ensuring seamless parking management and user experience.

The Smart Parking System is intended to effectively manage and monitor parking slots utilizing a blend of several components and microcontroller. This system allows users to book parking slots, ensures security, and alerts staff in case of emergencies. It operates as follow:

•As a vehicle approaches the Parking Plaza barrier, the entrance IR sensor detects its presence.

- The ESP32 processes the sensor data and give command to servo motor to open the barrier.

- The OLED display shows the web server credentials for the user to connect and book a slot.

Slot Booking and Management:

- The user will then connect to the web server using the displayed credentials, for slot reservation.
- Through the web interface, the user can view the availability of parking slots and book an available slot.
- Upon slot booking, the corresponding LED lights up to indicate reservation.
- The user parks their vehicle in the reserved slot, and the respective IR glow up for indication.

Slot Disengagement:

- After using slot, the user can disengage the booked slot through the web server.
- The LED indicator turns off, marking the slot as available again.

Fire Detection and Security:

- The fire sensor continuously monitors the parking area for any signs of fire.
- In case of fire detection, the buzzer sounds an alarm to alert the staff.
- The Camera provides live video, allowing the security team to monitor the area.
- The security staff can also control the buzzer and the barrier through the web server and via physical button to manage emergencies and prevent unauthorized access.

Web Server Control

- The web server provides a user-friendly interface for booking and managing parking slots.
- It also allows the security team to control the barrier, and activate or deactivate the buzzer.

Maintaining a smart parking system requires ongoing attention to ensure its efficiency, security, and user satisfaction. Regular updates, sensor calibration, database management, and user interface enhancements are critical components of an effective maintenance plan. These efforts ensure the system remains reliable, secure, and user-friendly, providing a seamless parking experience.

System Updates:

Regular updates are crucial for maintaining the smart parking system. These updates protect the system from vulnerabilities, enhance functionality, and ensure compatibility with new devices and technologies. Regular maintenance schedules should be established to apply these updates without disrupting the system's operations.

Sensor Calibration and Testing:

To maintain accuracy, all sensors must be periodically calibrated and tested. Regular checks ensure the sensors provide reliable data, detect vehicle presence accurately, and maintain seamless gate operations. Any malfunctioning sensors should be promptly repaired or replaced to prevent system inefficiencies.

Database Management and Optimization:

Efficient database management is vital for the smart parking system's performance. Regular database optimization ensures quick data retrieval and smooth operation. This includes routine backups and getting rid of outdated information.

User Interface and Experience Enhancements:

Continuous improvement of the user interface is essential for maintaining a positive user experience. Regular feedback collection from users can identify areas for enhancement. This helps in retaining user satisfaction and engagement.

Here is a list of suggestions on what can be added to this project in the future for improvement.

creating mobile Application:

Creating a mobile application that users can install on their phones to book parking spots easily.

Payment System:

Incorporating a payment system within the mobile application to help users to book and pay for parking spots ahead of time.

Navigation Assistance:

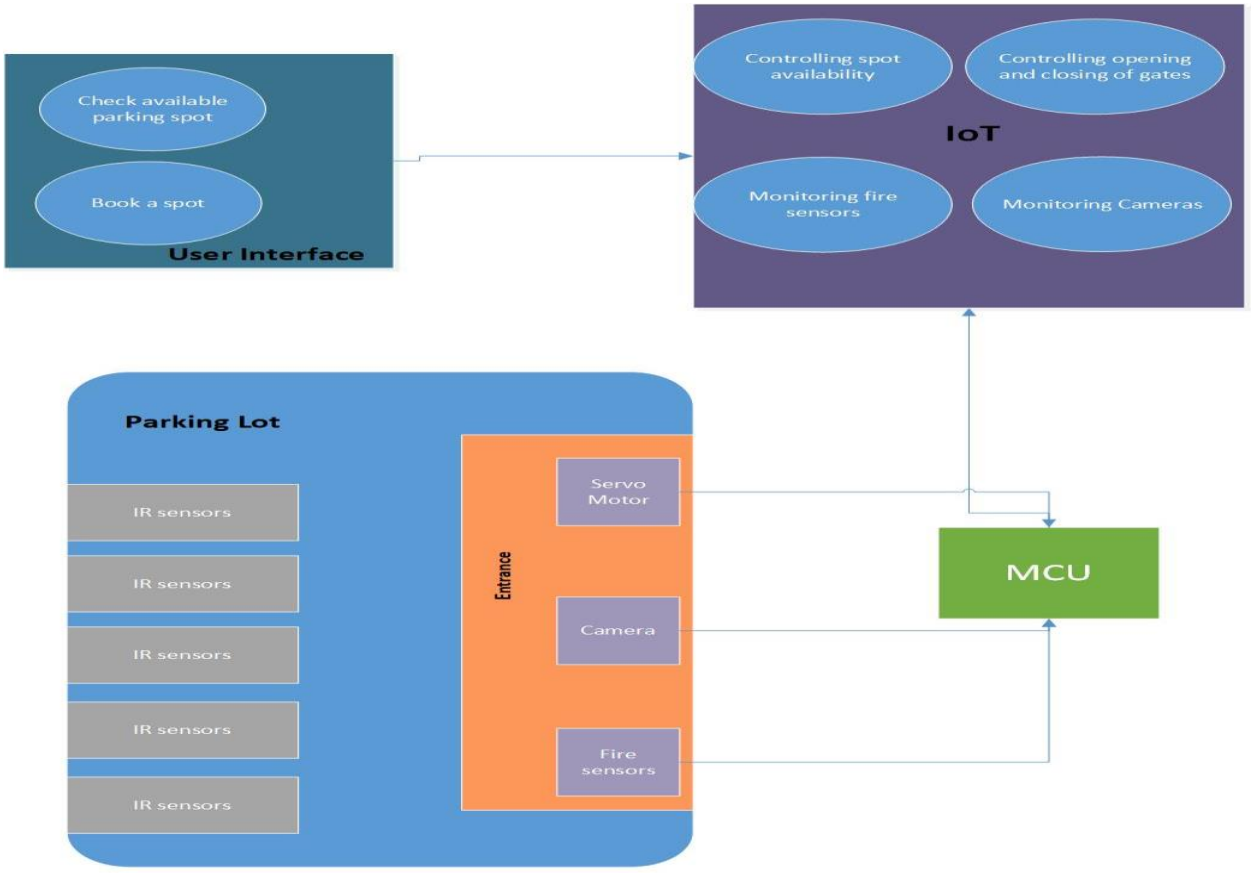
Adding a map feature to the application to guide users directly to their reserved parking spot.

In conclusion, the Spot Savy Smart Parking System offers a scalable and efficient solution for modern urban parking challenges, promoting a smarter, more connected city infrastructure.

References

Garnier, Geoffrey. "What Future for Smart Parking?" Canadian Parking Association, https://canadianparking.ca/what-future-for-smart-parking/. Accessed 18 June 2024.

Li, Jasmine. "Why Smart Parking Leads To Smarter Cities." *Parking Industry Canada*, 15 Jan. 2021, www.parkingindustry.ca/feature-articles/why-smart-parking-leads-to-smarter-cities. Accessed 18 June 2024.



Spot Savvy: Cutting Edge Mall Parking System

Figure 12: system block diagram

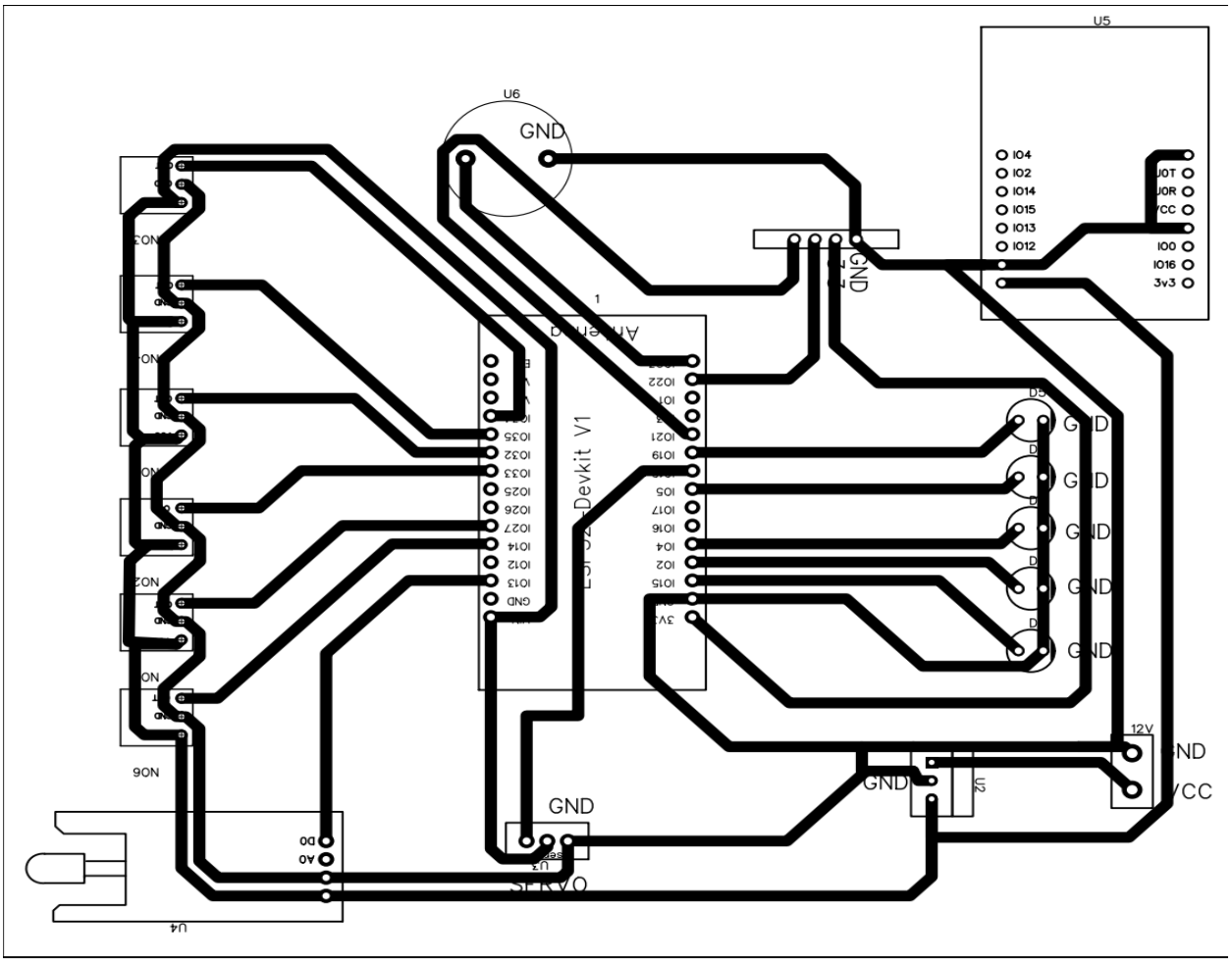


Figure 13:Electrical Diagram

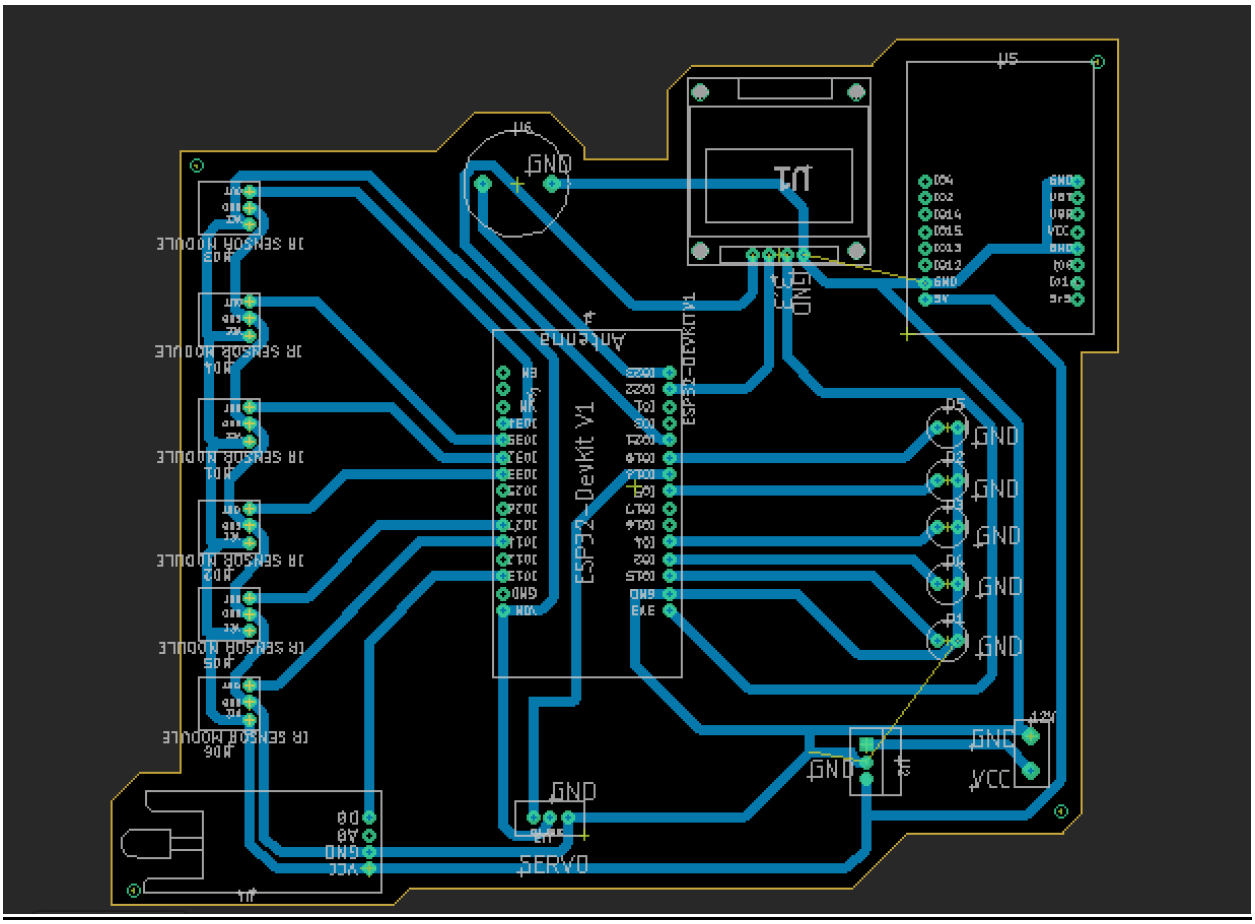


Figure 14:PCB Design Diagram

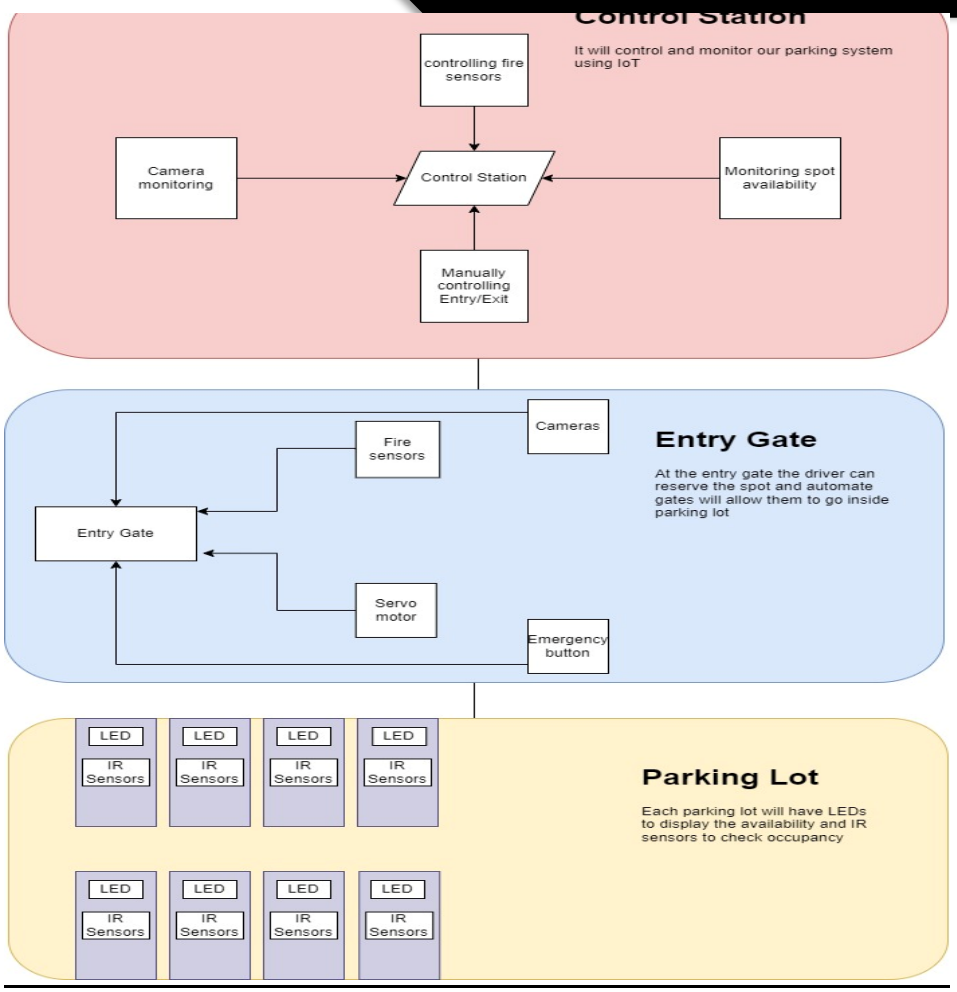


Figure 111: Mechanical Diagram-2

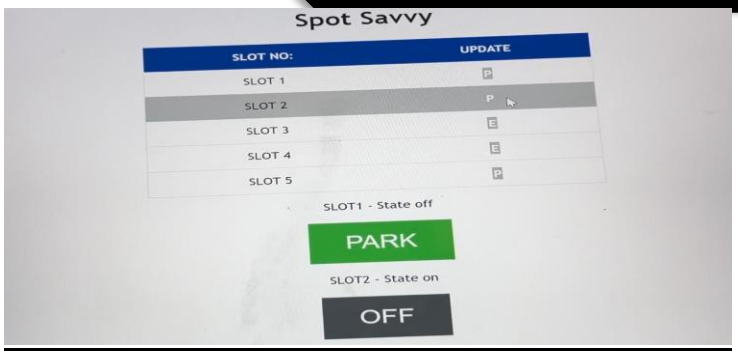


Figure 16: IOT Dashboard-1

Figure 17: IOT Dashboard-2

Figure 18: Capture of parking from ESPCAM

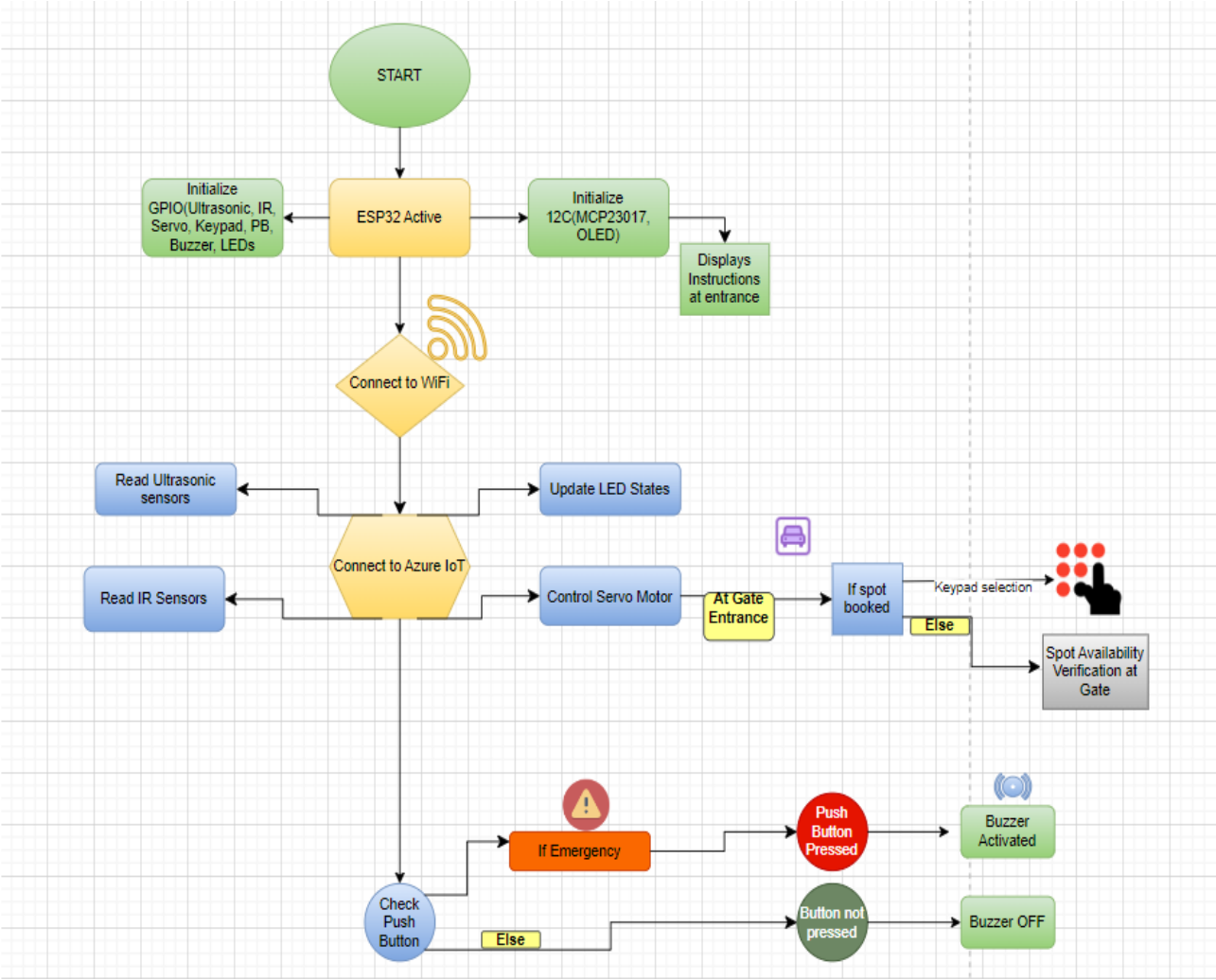


Figure 19: Flowchart

```
#include <WiFi.h>

#include <Servo.h>

#include <SPI.h>

#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>


#define SCREEN_WIDTH 128

#define SCREEN_HEIGHT 64

#define OLED_RESET     -1

#define SCREEN_ADDRESS 0x3C

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);


const char* ssid     = "KRUSHALAPI 3980";

const char* password = "L3@224y5";


WiFiServer server(80);


String header;

unsigned long currentTime = millis();

unsigned long previousTime = 0;

const long timeoutTime = 2000;


// Auxiliar variables to store the current output state

String output1State = "off";

String output2State = "off";
```

```
String output3State = "off";

String output4State = "off";

String output5State = "off";

String output6State = "off";

String output7State = "off";


#define IR1 32

#define IR2 33

#define IR3 34

#define IR4 35

#define IR5 27

#define IR6 14

int P1,P2,P3,P4,P5,E;

char S1 = 'E', S2 = 'E', S3 = 'E', S4 = 'E', S5 = 'E';


#define SERVO_PIN 18

Servo servoMotor;


#define LED1 2

#define LED2 4

#define LED3 5

#define LED4 15

#define LED5 19

#define BUZZ 23
```

```
#define FLAME 13

int F;


#define Button 25

int B = 0;

bool servoWebState = false;

bool buzzerWebState = false;


void setup() {

 Serial.begin(115200);

 pinMode(IR1,INPUT);

 pinMode(IR2,INPUT);

 pinMode(IR3,INPUT);

 pinMode(IR4,INPUT);

 pinMode(IR5,INPUT);

 pinMode(IR6,INPUT);

 pinMode(FLAME, INPUT);

 pinMode(Button, INPUT);

 pinMode(LED1,OUTPUT);

 pinMode(LED2,OUTPUT);

 pinMode(LED3,OUTPUT);

 pinMode(LED4,OUTPUT);

 pinMode(LED5,OUTPUT);

 pinMode(BUZZ,OUTPUT);

 digitalWrite(BUZZ, LOW);

 servoMotor.attach(SERVO_PIN);

 Serial.print("Connecting to ");
```

```
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.print("WiFi Name: ");
  display.println(ssid);
  display.setCursor(0,10);
  display.print("Password: ");
  display.println(password);
  display.setCursor(0,28);
  display.println("IP address: ");
  display.setCursor(0,40);
```

```
  display.println(WiFi.localIP());
  display.display();
}

void loop(){
  P1 = digitalRead(IR1);
  P2 = digitalRead(IR2);
  P3 = digitalRead(IR3);
  P4 = digitalRead(IR4);
  P5 = digitalRead(IR5);
  E = digitalRead(IR6);
  F = digitalRead(FLAME);
  B = digitalRead(Button);

  if(P1 == 1) {
    digitalWrite(LED1,LOW);
    S1 = 'E';
  } else {
    digitalWrite(LED1,HIGH);
    S1 = 'P';
  }

  if(P2 == 1) {
    digitalWrite(LED2,LOW);
    S2 = 'E';
  } else {
    digitalWrite(LED2,HIGH);
```

```
  S2 = 'P';
 }

 if(P3 == 1) {
  digitalWrite(LED3,LOW);
  S3 = 'E';
 } else {
  digitalWrite(LED3,HIGH);
  S3 = 'P';
 }

 if(P4 == 1) {
  digitalWrite(LED4,LOW);
  S4 = 'E';
 } else {
  digitalWrite(LED4,HIGH);
  S4 = 'P';
 }

 if(P5 == 1) {
  digitalWrite(LED5,LOW);
  S5 = 'E';
 } else {
  digitalWrite(LED5,HIGH);
  S5 = 'P';
 }
```

```
// if(E == 1) {
//   servoMotor.write(90);
// } else {
//   servoMotor.write(0);
// }
//
// if(F == 1) {
//   digitalWrite(BUZZ,HIGH);
// } else {
//   if(B == 1) {
//     digitalWrite(BUZZ,HIGH);
//   } else {
//     digitalWrite(BUZZ,LOW);
//   }
// }

if (!servoWebState) {
    servoMotor.write(E ? 90 : 0);
  }

  // Update buzzer based on flame sensor and button states, unless overridden by web server
```

```
  if (!buzzerWebState) {
    if (F == 1 || B == 1) {
      digitalWrite(BUZZ, HIGH);
    } else {
      digitalWrite(BUZZ, LOW);
    }
  }


 WiFiClient client = server.available();
 if (client) {
  currentTime = millis();
  previousTime = currentTime;
  Serial.println("New Client.");
  String currentLine = "";
  while (client.connected() && currentTime - previousTime <= timeoutTime) {
   currentTime = millis();
   if (client.available()) {
    char c = client.read();
    Serial.write(c);
    header += c;
    if (c == '\n') {
     if (currentLine.length() == 0) {
      client.println("HTTP/1.1 200 OK");
      client.println("Content-type:text/html");
      client.println("Connection: close");
      client.println();
```

```
            if (header.indexOf("GET /1/on") >= 0) {
              output1State = "on";
              S1 = 'P';
            }
            else if (header.indexOf("GET /1/off") >= 0) {
              output1State = "off";
              S1 = 'E';
            }
            else if (header.indexOf("GET /2/on") >= 0) {
              output2State = "on";
              S2 = 'P';
            }
            else if (header.indexOf("GET /2/off") >= 0) {
              output2State = "off";
              S2 = 'E';
            }
            else if (header.indexOf("GET /3/on") >= 0) {
              output3State = "on";
              S3 = 'P';
            }
            else if (header.indexOf("GET /3/off") >= 0) {
              output3State = "off";
              S3 = 'E';
            }
            else if (header.indexOf("GET /4/on") >= 0) {
              output4State = "on";
```

```
            S4 = 'P';
        }
        else if (header.indexOf("GET /4/off") >= 0) {
            output4State = "off";
            S4 = 'E';
        }
        else if (header.indexOf("GET /5/on") >= 0) {
            output5State = "on";
            S5 = 'P';
        }
        else if (header.indexOf("GET /5/off") >= 0) {
            output5State = "off";
            S5 = 'E';
        }
        else if (header.indexOf("GET /6/on") >= 0) {
                    output6State = "on";
                    buzzerWebState = true;
                    digitalWrite(BUZZ, HIGH);
                } else if (header.indexOf("GET /6/off") >= 0) {
                    output6State = "off";
                    buzzerWebState = false;
                    digitalWrite(BUZZ, LOW);
                } else if (header.indexOf("GET /7/on") >= 0) {
                    output7State = "on";
                    servoWebState = true;
                    servoMotor.write(0);
                }
```

```
 else if (header.indexOf("GET /7/off") >= 0) {
                    output7State = "off";
                    servoWebState = false;
                    servoMotor.write(90);
                }


        client.println("<!DOCTYPE html><html>");
        client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
        client.println("<link rel=\"icon\" href=\"data:,\">");
        client.println("<script>function refreshPage() { setTimeout(() => { location.reload(); }, 5000); }</script>");
        client.println("<style>body { text-align: center; font-family: \"Trebuchet MS\", Arial;}");
        client.println("table { border-collapse: collapse; width:35%; margin-left:auto; margin-right:auto; }");
        client.println("th { padding: 12px; background-color: #0043af; color: white; }");
        client.println("tr { border: 1px solid #ddd; padding: 12px; }");
        client.println("tr:hover { background-color: #bcbcbc; }");
        client.println("td { border: none; padding: 12px; }");
        client.println(".sensor { color:white; font-weight: bold; background-color: #bcbcbc; padding: 1px; }");
        client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;");
        client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
        client.println(".button2 {background-color: #555555;}</style></head>");


        client.println("<body onload=\"refreshPage()\"><h1>Spot Savvy</h1>");
        client.println("<table><tr><th>SLOT NO:</th><th>UPDATE</th></tr>");
        client.println("<tr><td>SLOT 1</td><td><span class=\"sensor\">");
```

```
client.println(S1);
client.println("</span></td></tr>");
client.println("<tr><td>SLOT 2</td><td><span class=\"sensor\">");
client.println(S2);
client.println("</span></td></tr>");
client.println("<tr><td>SLOT 3</td><td><span class=\"sensor\">");
client.println(S3);
client.println("</span></td></tr>");
client.println("<tr><td>SLOT 4</td><td><span class=\"sensor\">");
client.println(S4);
client.println("</span></td></tr>");
client.println("<tr><td>SLOT 5</td><td><span class=\"sensor\">");
client.println(S5);
client.println("</span></td></tr>");
client.println("</table>");
// Display current state, and ON/OFF buttons
client.println("<p>SLOT1 - State " + output1State + "</p>");
// If the output26State is off, it displays the ON button
if (output1State=="off") {
  client.println("<p><a href=\"/1/on\"><button class=\"button\">PARK</button></a></p>");
} else {
  client.println("<p><a href=\"/1/off\"><button class=\"button button2\">OFF</button></a></p>");
}
client.println("<p>SLOT2 - State " + output2State + "</p>");
```

```
            // If the output26State is off, it displays the ON button
            if (output2State=="off") {
              client.println("<p><a href=\"/2/on\"><button class=\"button\">PARK</button></a></p>");
            }
          else {
              client.println("<p><a href=\"/2/off\"><button class=\"button button2\">OFF</button></a></p>");
            }
            client.println("<p>SLOT3 - State " + output3State + "</p>");
            // If the output26State is off, it displays the ON button
            if (output3State=="off") {
              client.println("<p><a href=\"/3/on\"><button class=\"button\">PARK</button></a></p>");
            } else {
              client.println("<p><a href=\"/3/off\"><button class=\"button button2\">OFF</button></a></p>");
            }
            client.println("<p>SLOT4 - State " + output4State + "</p>");
            // If the output26State is off, it displays the ON button
            if (output4State=="off") {
              client.println("<p><a href=\"/4/on\"><button class=\"button\">PARK</button></a></p>");
            } else {
              client.println("<p><a href=\"/4/off\"><button class=\"button button2\">OFF</button></a></p>");
            }
            client.println("<p>SLOT5 - State " + output5State + "</p>");
            // If the output26State is off, it displays the ON button
            if (output5State=="off") {
              client.println("<p><a href=\"/5/on\"><button class=\"button\">PARK</button></a></p>");
            }
```

```
else {

          client.println("<p><a href=\"/5/off\"><button class=\"button button2\">OFF</button></a></p>");

        }

          client.println("<tr><td>Buzzer</td><td><span class=\"sensor\">" + output6State + "</span></td><td><a href=\"/6/on\"><button class=\"button\">ON</button></a><a href=\"/6/off\"><button class=\"button button2\">OFF</button></a></td></tr>");

          client.println("<tr><td>Gates</td><td><span class=\"sensor\">" + output7State + "</span></td><td><a href=\"/7/on\"><button class=\"button\">ON</button></a><a href=\"/7/off\"><button class=\"button button2\">OFF</button></a></td></tr>");

          client.println("</body></html>");

          client.println();

          break;

        }

        else {

          currentLine = "";

        }

      }

      else if (c != '\r') {

        currentLine += c;

      }

    }

  }

  header = "";

  client.stop();

  Serial.println("Client disconnected.");

  Serial.println("");

 }

}
```

No. #	PART NAME	QUANTITY	PART NUMBER	PRICE/QUANTITY	TOTAL
1	Ultrasonic sensors	7	HC-SR04	$6.95	$48.65
2	Keypad	1	4x4KPSM	$7.95	$7.95
3	IR sensor	1	HC-SR501	$17.95	$17.95
4	LEDs	7		$0.28	$1.96
5	Breadboard	2		$12.55	25.1
6	ESP32 + Camera module	1	ESP32-WROVER-E	$23.95	$23.95
7	Servo motor	1	SG90	$5.26	$5.26
8	Buzzer	1	2223-CMI-1295IC-0585T-ND	$0.85	$0.85
9	Push Button	1	DAOKI-6x6x5	$2.69	$2.69
10	OLED Display	1	1814-FA2	$39.95	$39.95
11	I2C GPIO expander	1	MCP23017T-E/SO	$2.51	$2.51
12	Jumper wires	1		8.95	8.95
TOTAL					$185.77

Bill of Materials

Contact Info:

*Name: Tony Braxton Tchio Ngoumeza

Phone number: +1 (437) 684-1777

Email: ttchio-ngoumeza@myseneca.ca

*Name: Krusha Vipulkumar Mistry

Phone number: +1 (647) 668-3303

Email: kmistry23@myseneca.ca