LBTMS

Lithium-Ion Battery Thermal Runaway System

Muhammed Nihal C
Rajagiri School of engineering and technology

The Problem

- The problem at hand is the lack of an optimal and dependable battery monitoring system specifically designed for lithium-ion batteries in electric bikes.
- As the core component of an EV, the power battery directly affects the performance and safety.In order to improve the safety of power batteries, the internal failure mechanism and behavior characteristics of internal short circuit (ISC) and thermal runaway (TR) in extreme cases need to be tested and studied.
- Existing solutions primarily focus on either hardware or software components individually, without integrating them into a cohesive system.
- This gap hinders the ability to detect and predict deviations from ideal battery conditions, potentially leading to severe consequences such as battery failure or even safety hazards for the user.

A 7-year old boy dies of injuries in an e-scooter battery blast in Maharashtra

The deceased, identified as Shabbir Ansari, was a Class 2 student. He died during treatment for injuries suffered in an electric scooter battery blast in Vasai area of Palghar, Maharashtra.

Objective

- To address these challenges, there is a need for an integrated hardware and software battery monitoring system that can continuously monitor lithium-ion batteries in electric bikes, collect real-time data on crucial battery conditions, analyze deviations from ideal conditions, and provide users with actionable insights.

- This comprehensive system should offer a user-friendly interface that displays real-time data variations and alerts the user of any potential negative impacts on the battery's condition.

- By implementing such a system, electric bike users can ensure the optimal performance and longevity of their batteries while minimizing the risk of battery damage and safety hazards.

The Solution

Current

Measure, Compare and store current variations with preset tolerance/peak value

Voltage

Measure,Compare and store voltage variations with the preset tolerance/peak value

Temperature

Measure, Compare and store temperature variations with the preset tolerance/peak value

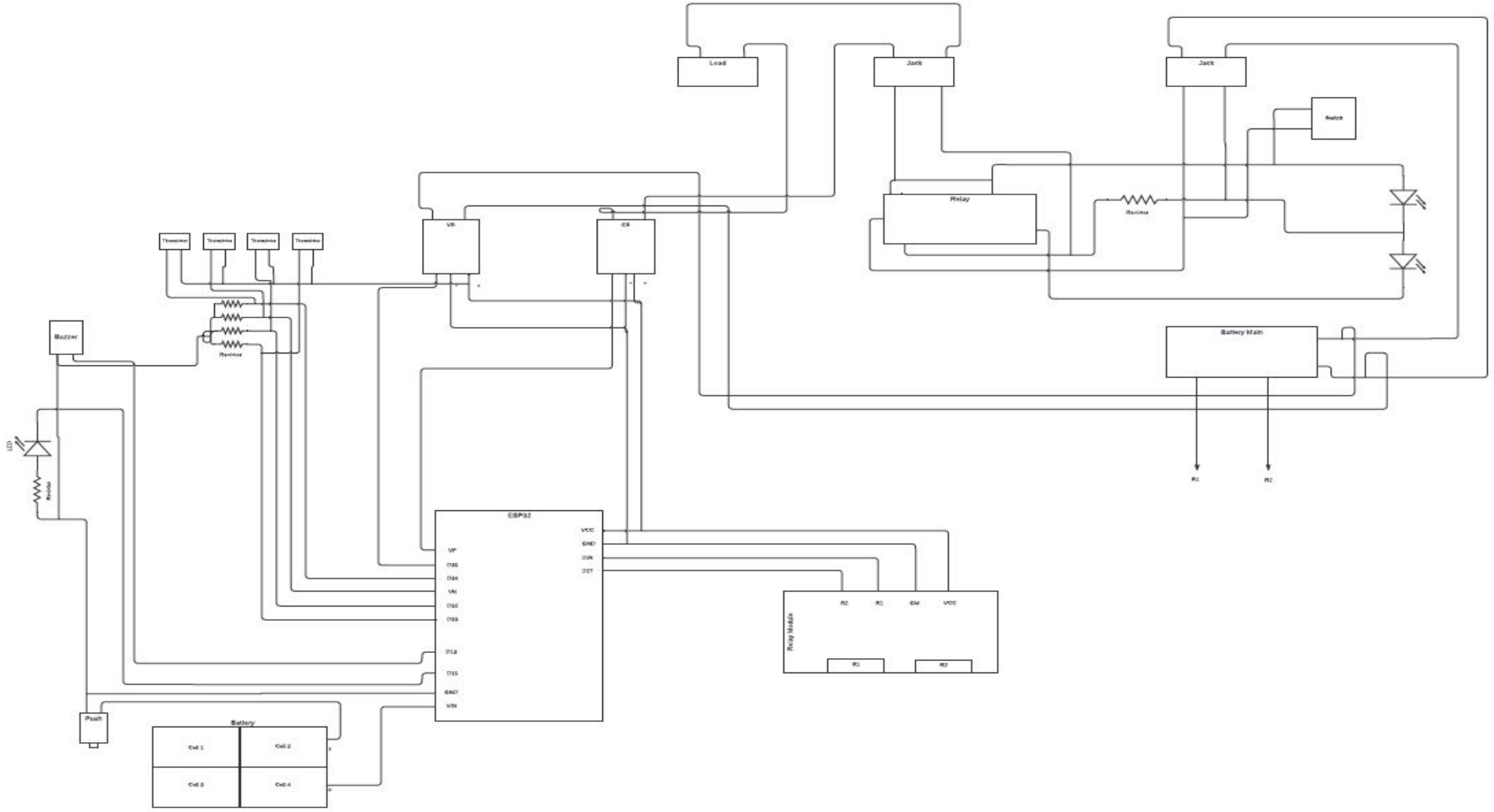
Isolate

To isolate the fault from rest of the circuit by comparison of measured real time values with preset values

Components

1. ESP32
2. Voltage sensor
3. Current sensor
4. Temperature sensor
5. Relay module
6. Buzzer
7. LED
8. Switch

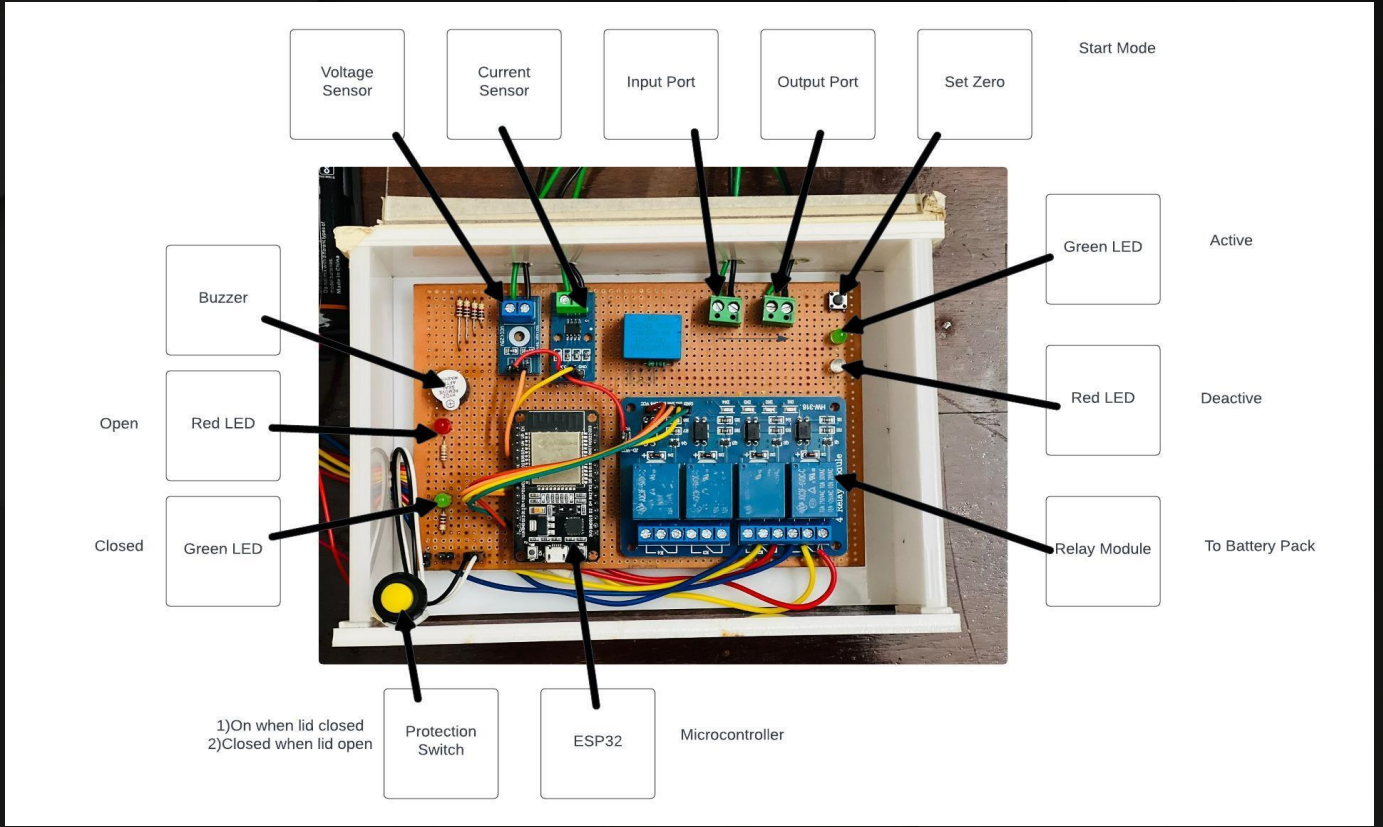
Circuit Diagram

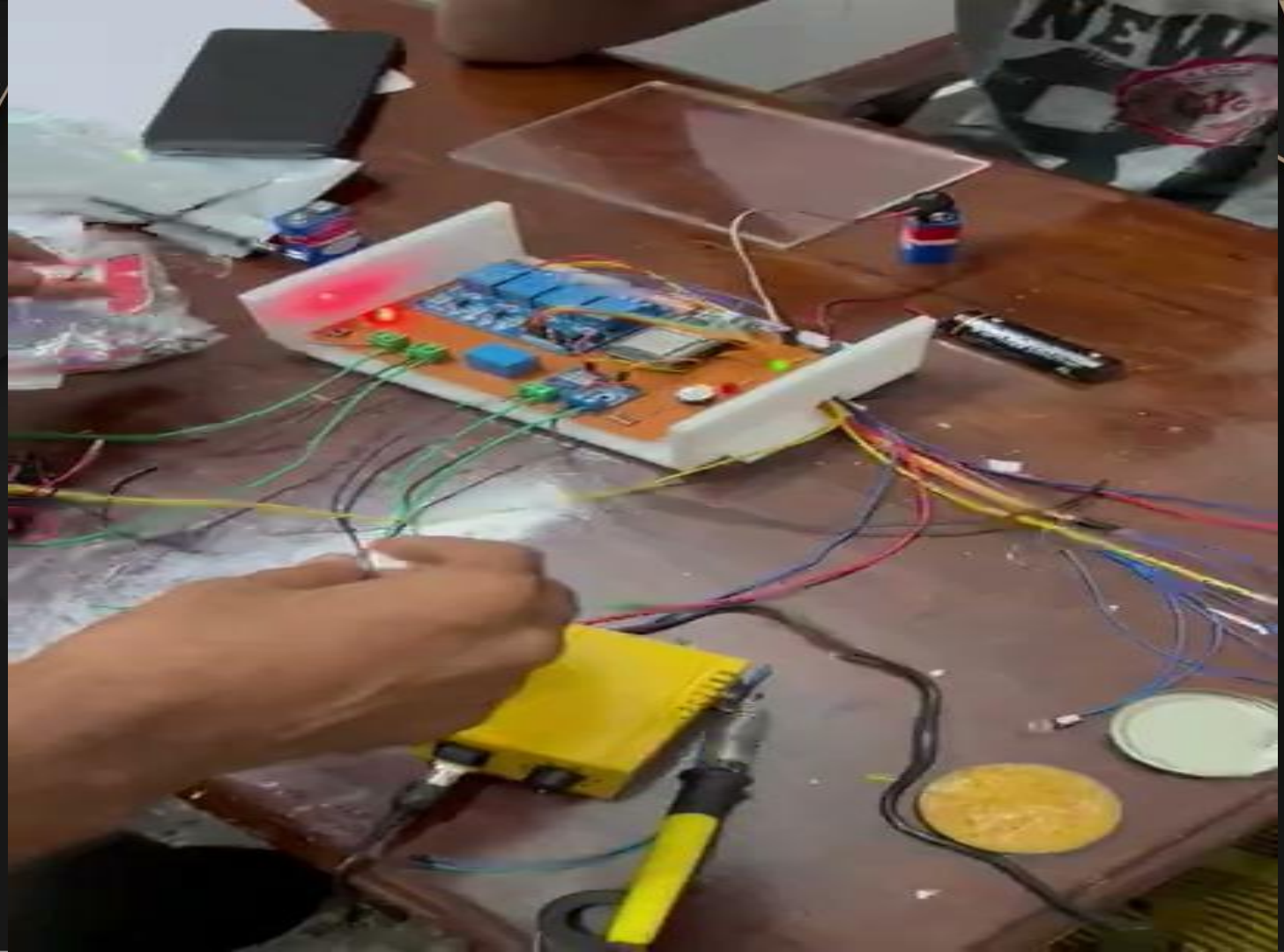

Lead
Jack
Jack
Switch
Relay
Resistor
Transistor
Transistor
Transistor
Transistor
Buzzer
Resistor
LED
Resistor
Battery Main
ESP32
VCC
GND
D26
D27
VP
D35
D34
VN
D32
D33
D12
D15
GND
VIN
Relay Module
IN2
IN1
GM
VCC
IN1
IN2
Push
Battery
Cell 1
Cell 2
Cell 3
Cell 4

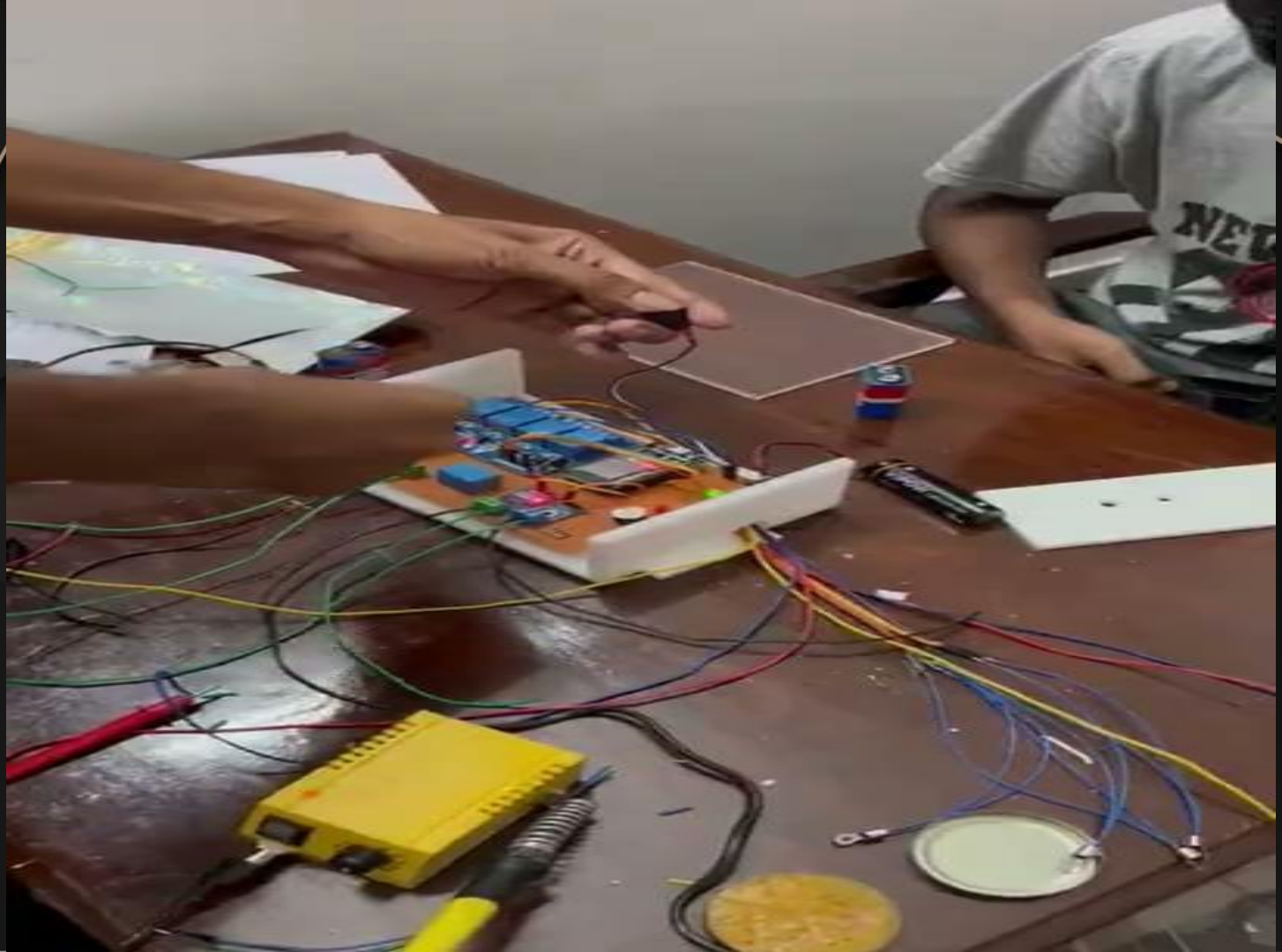
Working

- Objective is to predict and detect fault. Fault is detected using LM35 heat sensor, Current and Voltage sensor.
- LM35 heat sensor is attached to battery, regulator and motor.
- Voltage and current sensors are attached to line from battery.
- The flow of voltage and current will be monitored by microcontroller and provide information to user.
- When there's a change is heat above the tolerance level, it'll be detected as fault. If the temperature is rising rapidly, the faulty part will be isolated using relay.
- When there's a fault in cells of battery, there'll be change in output voltage and current. The output datas will be monitored by microcontroller and compare will past data and predict the level of fault. If the change in output voltage and current is more than the tolerance value, it is considered as fault and relay will be activated with the help of microcontroller and the details will be send to user.
- There's a high chance of short circuit, overloading, overcharging in EVs. To provide protection to EVs, we include emergency turn off system. If such a fault is detected, the entire system will be turned off and only be turned on after rectifying the issue manually.

Hardware


Voltage Sensor
Current Sensor
Input Port
Output Port
Set Zero
Start Mode
Buzzer
Green LED
Active
Open
Red LED
Red LED
Deactive
Closed
Green LED
Relay Module
To Battery Pack
1)On when lid closed
2)Closed when lid open
Protection Switch
ESP32
Microcontroller





Program - *(https://github.com/Nihalrahoof/OEN-Battery-management/tree/main)*

FRONT END :
Wix developer tools

BACK END :
Python Anywhere
Flask

DATA BASE :
MySQL

DATA BASE :
ESP32

Flask code

/home/oenlbtms/mysite/lbtms.py

Keyboard shortcuts: Normal | Share | Save | Save as... | >>> Run

```python
from flask import Flask,session,jsonify,request
from flask_session import Session
from flask_cors import CORS
from flask_httpauth import HTTPBasicAuth
from flask_mysqldb import MySQL
import MySQLdb.cursors



app = Flask(__name__)
CORS(app)   # So that the frontend and backend can make requests with necessary permissions
# auth = HTTPBasicAuth()

# Setting Up DB
app.config['MYSQL_HOST'] = 'oenlbtms.mysql.pythonanywhere-services.com'
app.config['MYSQL_USER'] = 'oenlbtms'
app.config['MYSQL_PASSWORD'] = 'insaneinsane'
app.config['MYSQL_DB'] = 'oenlbtms$sensordatas'
mysql = MySQL(app)

app.secret_key = '\xcb\x9e\x84(#/\t\xf74\xfd\x10\x06~2.\xe7\xed\x90hGNNX\xc7'

app.config["SECRET_KEY"] = 'DOTHACK2022'
app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
app.config.update(SESSION_COOKIE_SAMESITE="None", SESSION_COOKIE_SECURE=True)
Session(app)
```

```
 * Running on http://10.0.0.107:3001
Press CTRL+C to quit
```

/home/oenlbtms/mysite/lbtms.py *(unsaved changes)*

Keyboard shortcuts: Normal | Share | Save | Save as... | >>> Run

```
@app.route('/mysite/sensordata', methods=['GET', 'POST'])
def sensordata():

    data=request.get_json()
    value1 = data['value1']
    value2 = data['value2']
    value3 = data['value3']
    value4 = data['value4']
    value5 = data['value5']
    value6 = data['value6']
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute('INSERT INTO lbtms_datasheet VALUES (NULL,CURRENT_TIMESTAMP, %s, %s,%s,%s,%s,%s)', (value1, value2,value3,value4,value6,value5,))
    mysql.connection.commit()

@app.route('/mysite/sensordataerror', methods=['GET', 'POST'])
def sensordataerror():


    data=request.get_json()
    value1 = data['value1']
    value2 = data['value2']
    value3 = data['value3']
    value4 = data['value4']
    value5 = data['value5']
    value6 = data['value6']
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute('INSERT INTO fault_data VALUES (NULL,CURRENT_TIMESTAMP, %s, %s,%s,%s,%s,%s)', (value1, value2,value3,value4,value6,value5,))
    mysql.connection.commit()

```

```
 * Running on http://10.0.0.107:3001
Press CTRL+C to quit
```

```
58
59
60  @app.route('/mysite/senddata', methods=['GET', 'POST'])
61  def senddata():
62      cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
63      cursor.execute("SELECT * FROM lbtms_datasheet ORDER BY id DESC LIMIT 1")
64      data = cursor.fetchone()
65      cursor.close()
66      if data['temperature_cell1'] > 1 or data['temperature_cell2'] > 1 or data['temperature_cell3'] > 1 or data['temperature_cell4'] > 1 or data['Voltage']
67          data['status'] = "Fault detected"
68      else:
69          data['status'] = "Normal"
70      return jsonify(data)
71
72  @app.route('/mysite/senddatafault', methods=['GET', 'POST'])
73  def senddatafault():
74      cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
75      cursor.execute("SELECT * FROM fault_data ORDER BY id DESC LIMIT 1")
76      data = cursor.fetchone()
77      cursor.close()
78      return jsonify(data)
79
80
81
82  if __name__ == '__main__':
83      app.run(host='0.0.0.0',port=3001)
84
85      app.run(host='0.0.0.0',port=3001)
86  if __name__ == '__main__':
87      app.run(host='0.0.0.0',port=3001)
```

ESP32 CODE

oen_updated

```
nclude <WiFi.h>
nclude <HTTPClient.h>




nst int voltageSensorPin = 35;     // Analog pin for voltage sensor on ESP-WROOM-32 (GPIO35 / A3)
nst int currentSensorPin = 36;     // Analog pin for current sensor on ESP-WROOM-32 (GPIO36 / A4)
nst int tempSensorPin1 = 34;       // Analog pin for thermistor 1 on ESP-WROOM-32 (GPIO34 / A2)
nst int tempSensorPin2 = 39;       // Analog pin for thermistor 2 on ESP-WROOM-32 (GPIO39 / A5)
nst int tempSensorPin3 = 32;       // Analog pin for thermistor 3 on ESP-WROOM-32 (GPIO32 / A0)
nst int tempSensorPin4 = 33;       // Analog pin for thermistor 4 on ESP-WROOM-32 (GPIO33 / A1)

nst int relayPin1 = 26;            // Digital pin for relay 1 control on ESP-WROOM-32
nst int relayPin2 = 27;            // Digital pin for relay 2 control on ESP-WROOM-32
nst int ledPin = 13;               // Digital pin for LED output on ESP-WROOM-32
nst int buzzerPin = 15;            // Digital pin for buzzer output on ESP-WROOM-32


nst float maxVoltage = 8.0;        // Maximum allowable voltage (in volts)
nst float maxCurrent = 10.0;       // Maximum allowable current (in Amperes)
nst int maxTemperature = 600;       // Maximum allowable temperature (in degrees Celsius)

oat voltage, current;
t temp1, temp2, temp3, temp4;

initialising wifi
nst char* ssid = "Galaxy M5142E0";
nst char* password = "tomo0694";

id setup() {

//connecing to wifi
 Serial.begin(115200);
delay(50);
Serial.println("Connecting to WiFi...");
```

oen_updated

```
  WiFi.begin((char*)ssid, password); // Cast ssid to char* type

  while (WiFi.status() != WL_CONNECTED) {
    delay(50);
    Serial.println("Connecting to WiFi...");
  }

  Serial.println("Connected to WiFi");

  pinMode(relayPin1, OUTPUT);
  pinMode(relayPin2, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
}

void loop() {


  int voltageRaw = analogRead(voltageSensorPin);
  voltage = voltageRaw * (5.0 / 1023.0); // Assuming 10-bit ADC and 5V reference voltage

  // Read current from current sensor
  int currentRaw = analogRead(currentSensorPin);
  current = (currentRaw * 5.0 / 1023.0) * 5.0; // Assuming 10-bit ADC, 5V reference voltage, and 5A full-scale current range

  // Read temperature from thermistor 1
  temp1 = analogRead(tempSensorPin1);

  // Read temperature from thermistor 2
  temp2 = analogRead(tempSensorPin2);

  // Read temperature from thermistor 3
  temp3 = analogRead(tempSensorPin3);

  // Read temperature from thermistor 4
  temp4 = analogRead(tempSensorPin4);
```

oen_updated §

```
// Check for tolerance conditions
 if (voltage > maxVoltage || current > maxCurrent || temp1 > maxTemperature || temp2 > maxTemperature ) {
   // If any of the tolerance conditions are exceeded, trip 1st relay, turn on the LED, and sound the buzzer
   digitalWrite(relayPin1, HIGH);
   digitalWrite(ledPin, HIGH);
   digitalWrite(buzzerPin, HIGH);
 }
 else if (voltage > maxVoltage || current > maxCurrent || temp3 > maxTemperature || temp4 > maxTemperature) {
   // If any of the tolerance conditions are exceeded, trip 2nd relay, turn on the LED, and sound the buzzer
   digitalWrite(relayPin2, HIGH);
   digitalWrite(ledPin, HIGH);
   digitalWrite(buzzerPin, HIGH);
 }else {
   // If all sensor values are within tolerance, turn off the relay, LED, and buzzer
   digitalWrite(relayPin1, LOW);
   digitalWrite(relayPin2, LOW);
   digitalWrite(ledPin, LOW);
   digitalWrite(buzzerPin, LOW);
 }




 // Print the sensor values
 Serial.print("Voltage: ");
 Serial.print(voltage);
 Serial.println(" V");

 Serial.print("Current: ");
 Serial.print(current);
 Serial.println(" A");

 Serial.print("Temperature 1: ");
 Serial.print(temp1);
 Serial.println(" (raw)");
```

oen_updated §

```
  Serial.print("Temperature 2: ");
  Serial.print(temp2);
  Serial.println(" (raw)");

  Serial.print("Temperature 3: ");
  Serial.print(temp3);
  Serial.println(" (raw)");

  Serial.print("Temperature 4: ");
  Serial.print(temp4);
  Serial.println(" (raw)");




  delay(10); // Delay before taking the next reading

if(WiFi.status()== WL_CONNECTED){   //Check WiFi connection status

   //sending sensor data to flask page
   HTTPClient http;
   Serial.println("starting to send ");

   http.begin("https://oenlbtms.pythonanywhere.com/mysite/sensordata");
   http.addHeader("Content-Type", "application/json");
   String httpRequestData = "{\"value1\":\"" + String(temp1) + "\",\"value2\":\"" + String(temp2) +"\",\"value3\":\"" + String(temp3) +"\",\"value4\":\"" + String(temp4) +"\",\"value5\":\"" + String(curr
   int httpResponseCode = http.POST(httpRequestData);

   if(httpResponseCode>0){

    Serial.println(httpResponseCode);

   }else{

    Serial.println("Error on sending POST");
```

oen_updated §

```
  http.end();  //Free resources


  if (voltage > maxVoltage || current > maxCurrent || temp1 > maxTemperature || temp2 > maxTemperature || temp3 > maxTemperature || temp4 > maxTemperature) {
   // If any of the tolerance conditions are exceeded, send that values
   HTTPClient http;
  Serial.println("starting to send ");

  http.begin("https://oenlbtms.pythonanywhere.com/mysite/sensordataerror");
  http.addHeader("Content-Type", "application/json");
  String httpRequestData = "{\"value1\":\"" + String(temp1) + "\",\"value2\":\"" + String(temp2) +"\",\"value3\":\"" + String(temp3) +"\",\"value4\":\"" + String(temp4) +"\",\"value5\":\"" + String(curr
  int httpResponseCode = http.POST(httpRequestData);

  if(httpResponseCode>0){

   Serial.println(httpResponseCode);

  }else{

   Serial.println("Error on sending POST");

  }

  http.end();  //Free resources

 }

}else{

   Serial.println("Error in WiFi connection");

}

 delay(30);  //Send a request every 10 seconds

}
```

WIX CODE

sensor data

Run

```
const axios = require('axios');
$w.onReady(function () {
});
    axios
        .post("https://oenlbtms.pythonanywhere.com/mysite/senddata", {
            auth: {
                username: 'admin',
                password: 'insane',
            },
        })
        .then(function (response) {
            console.log("Authenticated");
            console.log(response.data);
            $w('#text28').text = response.data.date.toString();
            $w('#text29').text = response.data.Current.toString();
            $w('#text30').text = response.data.Voltage.toString();
            $w('#text42').text = response.data.temperature_cell1.toString();
            $w('#text43').text = response.data.temperature_cell2.toString();
            $w('#text44').text = response.data.temperature_cell2.toString();
            $w('#text45').text = response.data.temperature_cell3.toString();
            $w('#text37').text = response.data.id.toString();
            if (response.data.status.toString() === "Fault detected") {
             $w('#text46').text = "Fault detected";
             $w('#text47').text = "See Fault data";
    }

        })
        .catch(function (error) {
            console.log("Error on Authentication : ", error);
        });
```

Select an element to add events, edit its ID, and more.
Learn more

Fault data

Run

```
const axios = require('axios');
$w.onReady(function () {
});
    axios
        .post("https://oenlbtms.pythonanywhere.com/mysite/senddatafault", {
            auth: {
                username: 'admin',
                password: 'insane',
            },

        })
        .then(function (response) {
            console.log("Authenticated");
            console.log(response.data);
            $w('#text28').text = response.data.date.toString();
            $w('#text29').text = response.data.Current.toString();
            $w('#text30').text = response.data.Voltage.toString();
            $w('#text42').text = response.data.temperature_cell1.toString();
            $w('#text43').text = response.data.temperature_cell2.toString();
            $w('#text44').text = response.data.temperature_cell2.toString();
            $w('#text45').text = response.data.temperature_cell3.toString();
            $w('#text37').text = response.data.id.toString();


        })
        .catch(function (error) {
            console.log("Error on Authentication : ", error);
        });

```

Select an element to add events, edit its ID, and more.
Learn more

Output - (*https://u2005037.wixsite.com/website*)

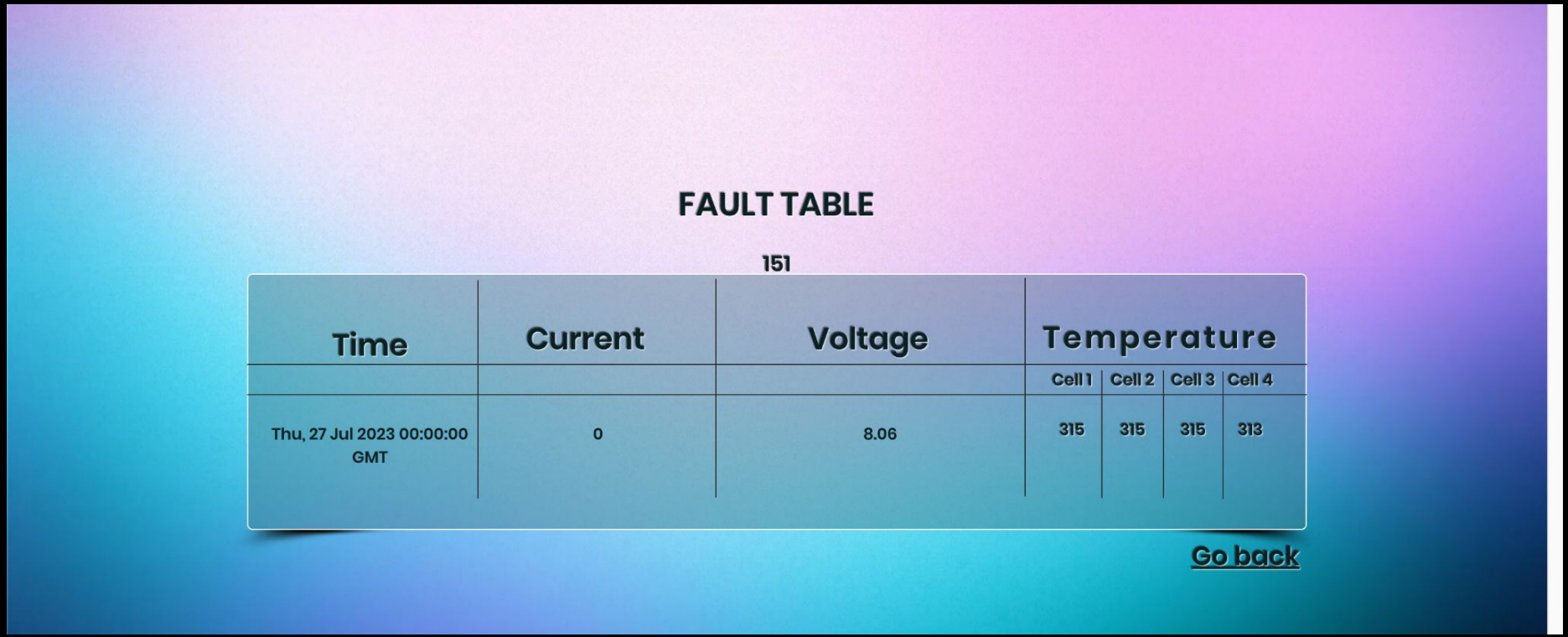
Latest Data

782

Time	Current	Voltage	Temperature			
			Cell 1	Cell 2	Cell 3	Cell 4
Thu, 27 Jul 2023 00:00:00 GMT	0	0	315	311	311	312

System Optimal condition

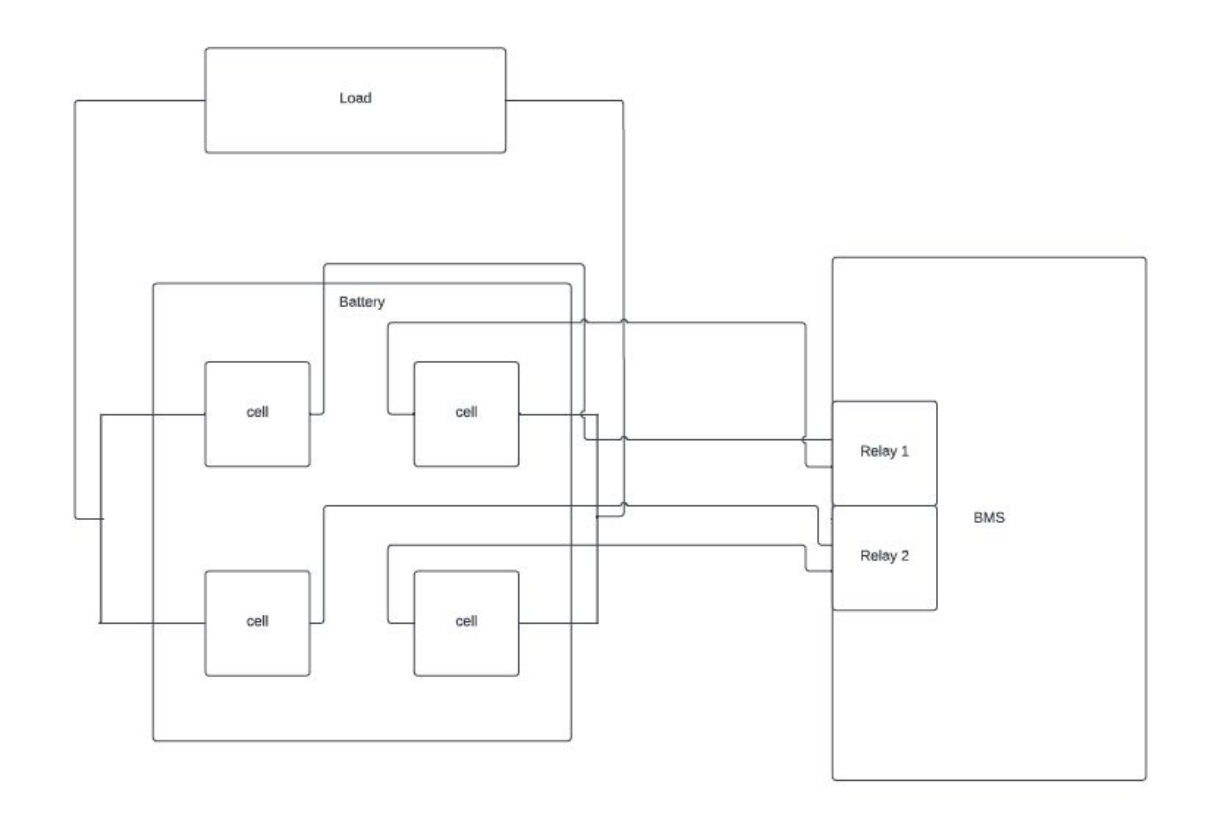
Output - (*https://u2005037.wixsite.com/website*)



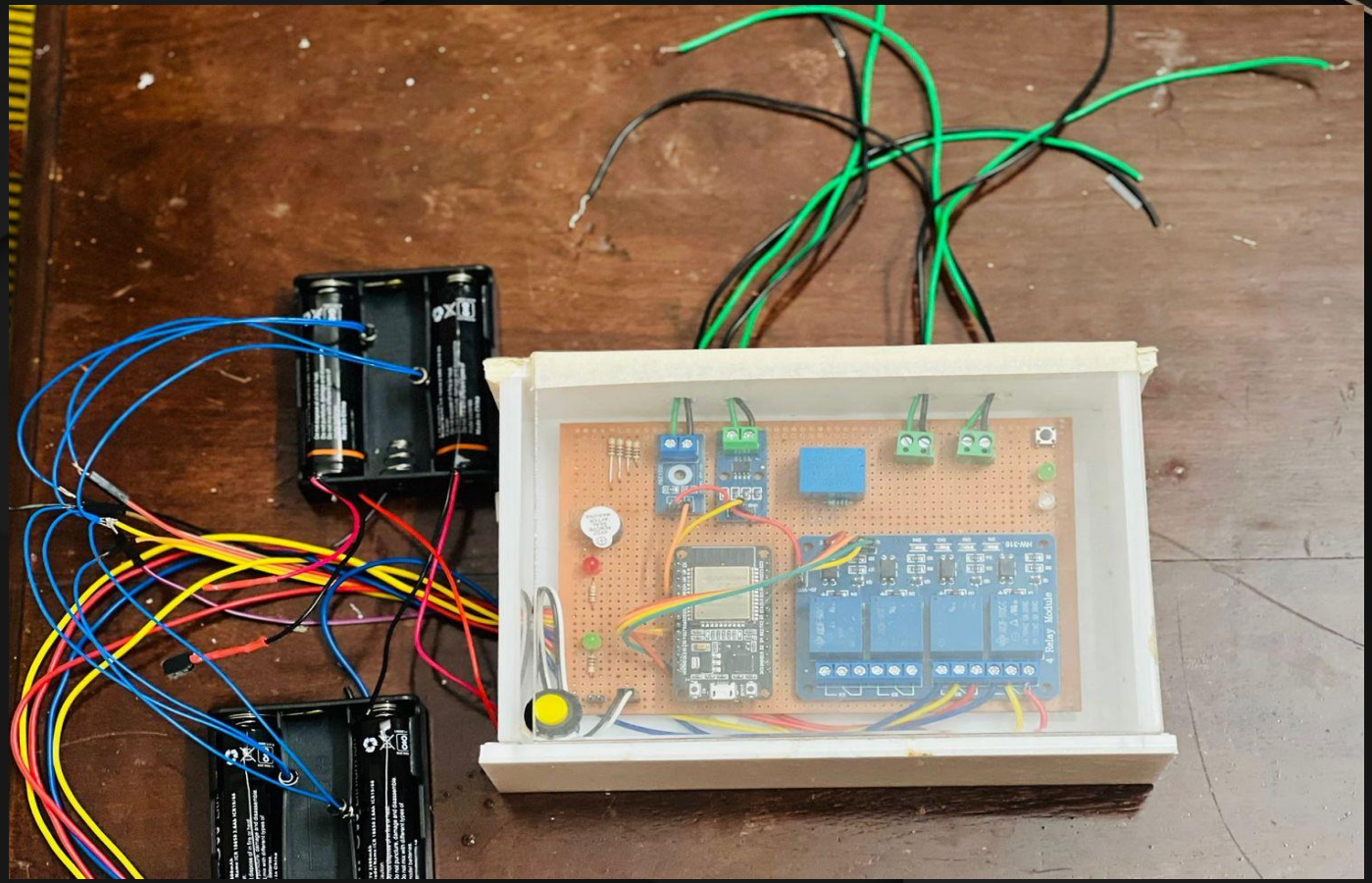
Cell Planning

- Centralized BMS System.
- Has one central BMS in the battery pack assembly. All the battery packages are connected to the central BMS directly
- Advantage- It is more compact, and it tends to be the most economical since there is only one BMS.
- Disadvantage- Since all the batteries are connected to the BMS directly, the BMS needs a lot of ports to connect with all the battery packages.
- Working- They relay senses the temperature across 2 cells connected in series(pair of 2) and opens the faulty cell pair and isolates that particular pair from rest of the cell pairs connected in parallel (battery pack) when the measured value crosses the threshold value.

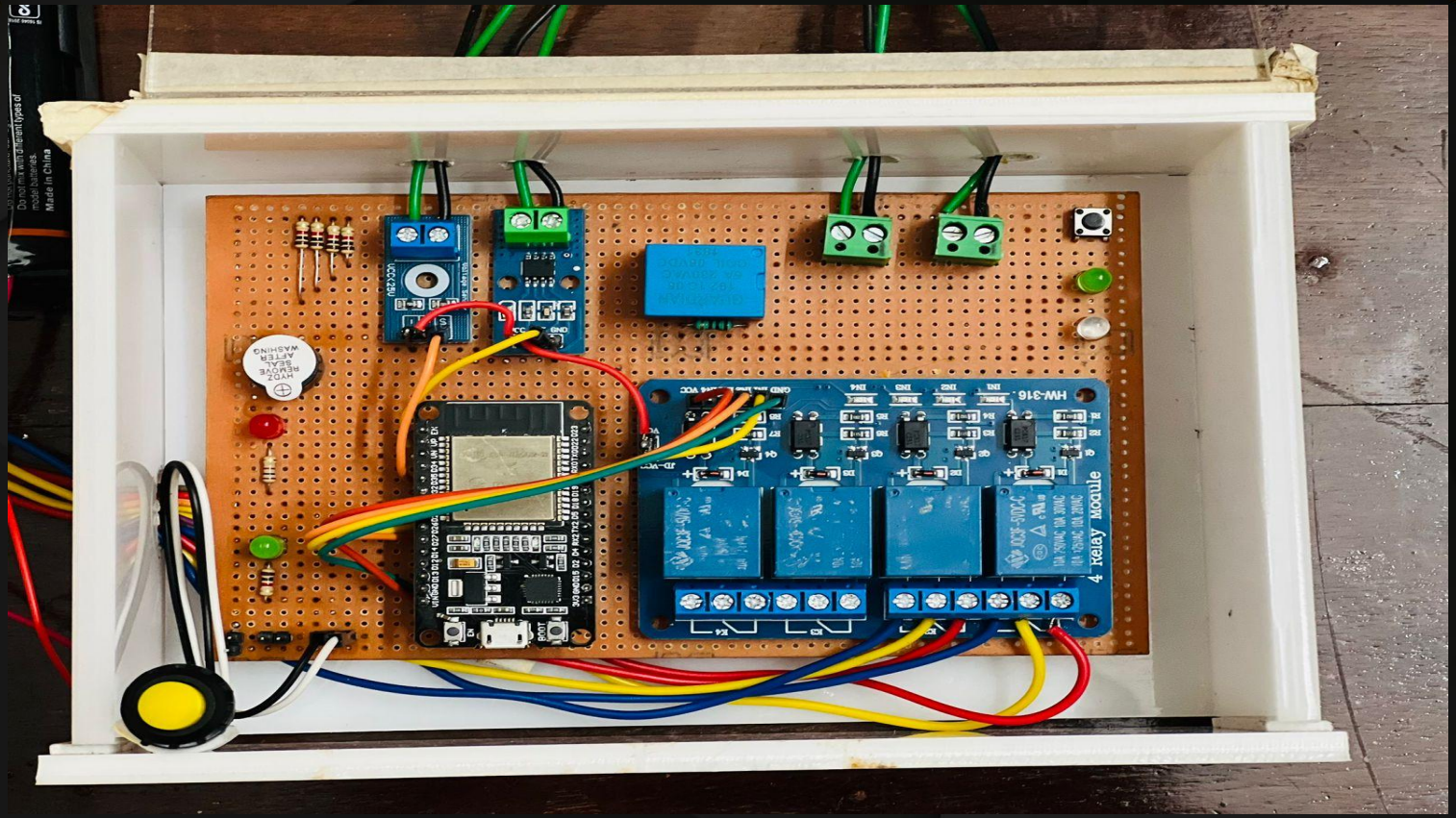
Cell Planning



Product Prototype



Product Prototype



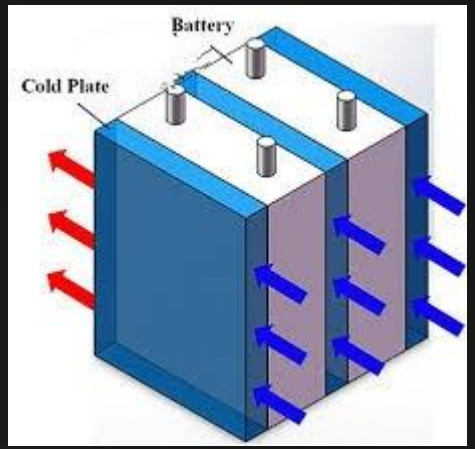
Prototype Budget

PROTOTYPE BUDGET			
Name of Components	Quantity	Rate	Price
Lithium ion battery	5	120	600
PCB	1	100	100
ESP	1	672	672
Relay module	1	250	250
Relay	1	30	30
Current sensor	1	250	250
voltage sensor	1	100	100
Thermister	4	80	320
LED	4	2	8
Buzzer	1	15	15
Wire		20	20
Battery case	2	50	100
Total			2465

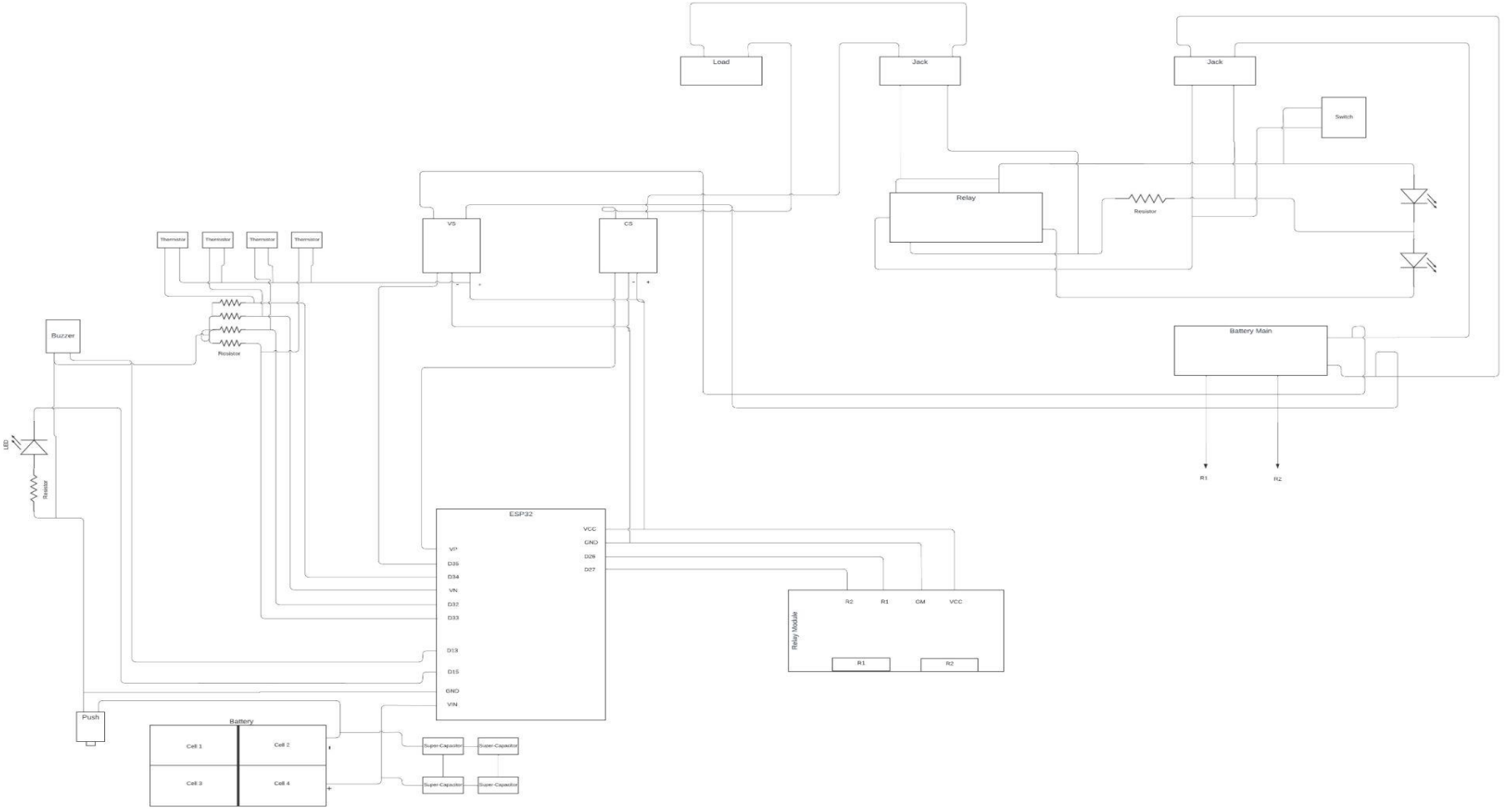
Future Scope

Cooling System

- Cooling - The battery cooling system in electric vehicles regulates the temperature of the battery pack.
- Cold plates-
    - Stabilizes battery cell temperature and provides optimal temperature uniformity.
    - The cells are packed in a box and the outer surface of these having cold plates mounted.



Inclusion of Supercapacitor


Load
Jack
Jack
Switch
Relay
Resistor
VS
CS
Thermistor
Thermistor
Thermistor
Thermistor
Buzzer
Resistor
Battery Main
LED
Resistor
R1
R2
ESP32
VCC
GND
D26
D27
VP
D35
D34
VN
D32
D33
D13
D15
GND
VIN
R2
R1
GM
VCC
Relay Module
R1
R2
Push
Battery
Cell 1
Cell 2
Cell 3
Cell 4
Super-Capacitor
Super-Capacitor
Super-Capacitor
Super-Capacitor

Ansys Simulation

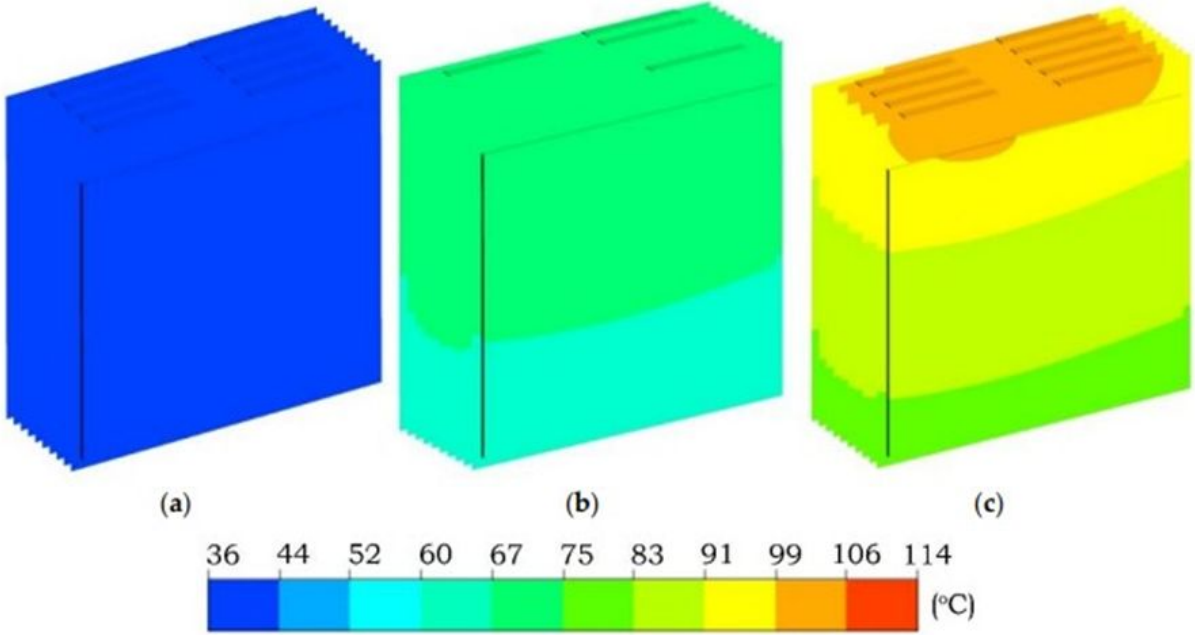


Figure 5. Temperature distribution of the battery module after discharging at (**a**) 1C, (**b**) 3C, and (**c**) 5C.

Temperature History of Battery

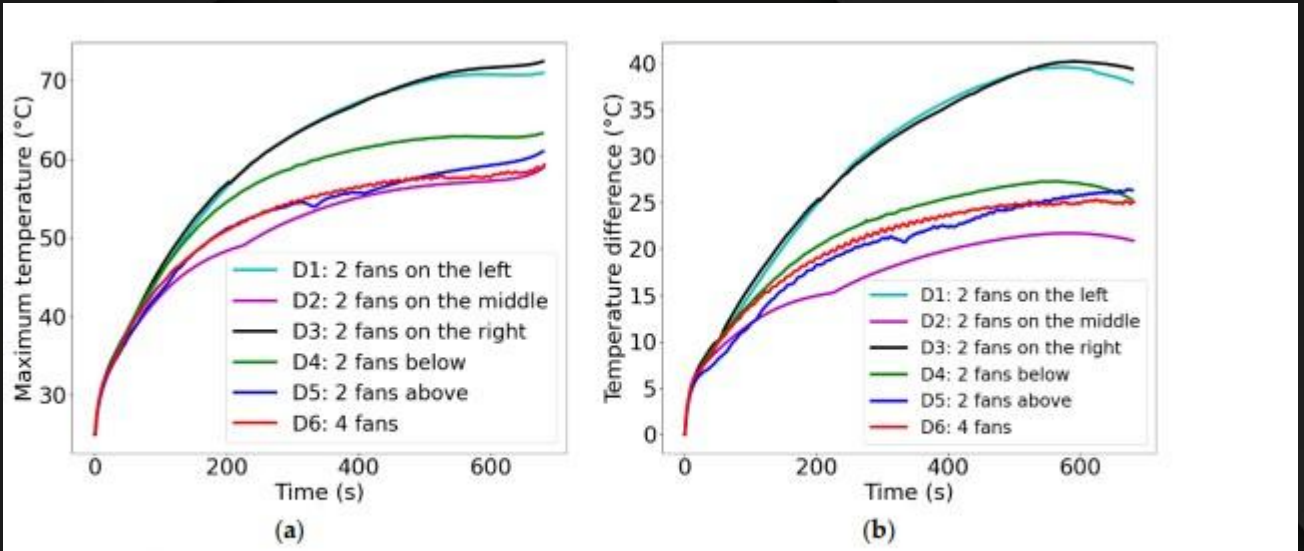


Figure 8. Temperature histories of a battery module with various fan systems: (**a**) maximum temperature curves and (**b**) temperature difference curves.

Thankyou