PROJECT REPORT

"Smart Instrumented Battery Module Design"

(Battery Module Optimization with Sensor Fusion)

NAME	Muhammad Ahtisham Sudheer
Degree	43
Domain	Electrical Engineering
Mentors	Prof Dr Mazhar Abbas Sir Harris Attaullah

Department of Electrical Engineering, National University of Science and Technology (NUST), Pakistan.

Smart Instrumented Battery Module Design

Abstract

The Smart Instrumented Battery Module (SIBM) represents a pivotal advancement in battery management systems, catering to the growing demand for sophisticated monitoring solutions. The SIBM is meticulously designed to offer a comprehensive suite of monitoring functionalities, addressing critical parameters such as voltage, current, temperature, and energy. Its primary objective is to empower efficient and reliable battery monitoring by integrating advanced sensors, including thermocouples for temperature estimation, Hall-effect sensors for current measurement, thermistors for thermal management, and voltage sensors for electrical potential monitoring. This amalgamation positions the SIBM as an exemplary system for multi-parameter monitoring, providing a holistic understanding of battery performance.

The architectural foundation of the SIBM involves a seamless integration of hardware and software components. The Arduino-based code serves as the software backbone, orchestrating communication, and data processing from the diverse array of sensors. The hardware integration includes meticulous circuit design and sensor placement to ensure optimal accuracy and reliability in monitoring. The project's flexibility allows it to accommodate two cylindrical battery cells, connected either in parallel or series, showcasing adaptability to various battery configurations.

Introduction

Battery management systems (BMS) are essential for ensuring the optimal performance, safety, and longevity of battery packs, especially for applications such as electric vehicles, renewable energy storage, and uninterruptible power supplies. BMS monitors and regulates the parameters of each battery cell, such as voltage, current, temperature, and state of charge, and performs actions such as balancing, charging, and discharging. BMS also communicates with external devices, such as chargers, inverters, and user interfaces, to provide information and feedback on the battery pack status and health.

However, designing and implementing a BMS can be challenging, as it requires a combination of hardware and software components, such as sensors, controllers, communication protocols, and algorithms, that need to work together reliably and efficiently. Moreover, different battery types, such as lithium-ion, lead-acid, or nickel-metal hydride, have different characteristics and requirements, which need to be considered by the BMS. Therefore, it is desirable to have a flexible and adaptable BMS that can be customized and modified according to the specific needs and applications of the battery pack.

In this project, we aim to develop a BMS using passive elements and Arduino, which is a popular and widely used open-source platform for prototyping and developing electronic projects. Arduino consists of a microcontroller board, such as the Arduino Uno, and a software environment, such as the Arduino IDE, that allows users to program and upload code to the board using a simple and intuitive language. Arduino also provides a large variety of libraries and shields, which are additional boards that can be stacked on top of the Arduino board to extend its functionality and compatibility with other devices and sensors.

The **main advantages** of using Arduino for BMS are:

- Arduino is easy to use and learn, as it has a user-friendly interface and a simple programming language that is based on C/C++.
- Arduino is low-cost and widely available, as it can be purchased online or from local stores, and it has a large and active community of users and developers that provide support and resources.
- Arduino is flexible and adaptable, as it can be interfaced with various sensors and devices, such as voltage and current sensors, temperature sensors, LCD displays, Bluetooth modules, and relays, and it can be programmed to perform different functions and algorithms, such as data acquisition, processing, communication, and control.

In this project, we design a battery management system (BMS) using passive elements and Arduino to monitor and regulate the parameters of a battery pack composed of two cylindrical lithium-ion cells. The BMS uses various sensors to measure the voltage, current, temperature, and temperature difference of the cells, and an Arduino Uno board to process the data and control a relay that can connect or disconnect the battery pack from the charger, or the load can be used to for safety of battery. The BMS also displays the information and feedback on the battery pack status and health on an LCD display using the I2C protocol.

Literature Review

Battery technologies play a pivotal role in various industries, from portable electronics to electric vehicles and renewable energy systems. Ensuring the optimal performance, safety, and longevity of batteries requires advanced monitoring systems. This literature review delves into the extensive research on battery monitoring, highlighting the importance of monitoring systems, advancements in sensor technologies, challenges faced, and future directions. The review covers key parameters such as voltage, current, temperature, and energy, exploring diverse approaches to enhance battery management systems (BMS).

1.1. Importance of Battery Monitoring Systems

Battery Monitoring Systems (BMS) are critical for managing the state and performance of batteries. In traditional lead-acid batteries and modern lithium-ion batteries, monitoring is essential to prevent overcharging, over-discharging, and thermal issues. The implementation of BMS enables real-time monitoring, balancing, and management of individual cells within a battery pack [1].

Accurate monitoring contributes to the improvement of battery life, efficiency, and overall safety.

1.2. Voltage Monitoring

Voltage monitoring is fundamental for understanding a battery's state of charge (SOC) and state of health (SOH). Various techniques have been explored to measure voltage accurately. Coulomb counting, open-circuit voltage measurements, and impedance spectroscopy are among the methods employed for precise voltage monitoring [2].

The advancement in voltage sensors ensures reliability in capturing voltage variations, enabling early detection of cell imbalances, and improving overall battery performance.

1.3. Current Monitoring

Accurate current measurement is crucial for evaluating the energy flow within a battery system. Hall-effect sensors have emerged as a popular choice due to their non-intrusive nature and ability to provide galvanic isolation [3]. Integrating Hall-effect sensors into BMS enhances accuracy and safety in current monitoring. The development of high-precision current sensors contributes to the overall efficiency of battery systems.

1,4. Temperature Monitoring

Temperature monitoring is paramount for battery safety and performance. Elevated temperatures can accelerate aging and lead to thermal runaway. Thermocouples and thermistors are widely employed for temperature sensing in BMS. Advanced algorithms, such as the Steinhart-Hart equation, are utilized to calculate precise temperatures from sensor readings [4]. Integrating temperature sensors aids in early fault detection and thermal management, ensuring the longevity of batteries.

1.5. Energy Monitoring

Energy monitoring involves measuring power and energy consumption or generation within a battery system. Power, defined as the rate of energy transfer, is calculated by integrating sensors for voltage and current. Energy monitoring provides insights into the efficiency of a battery system and aids in optimizing its operation for specific applications [5].

Continuous advancements in energy monitoring technologies contribute to the sustainable and efficient use of battery resources.

1.6. Advanced Sensor Integration

Recent research has focused on integrating multiple sensors into a unified monitoring system to provide a holistic view of battery parameters. The Smart Instrumented Battery Module (SIBM) introduced in this project exemplifies this trend. By combining thermocouples, Hall-effect sensors, thermistors, and voltage sensors, the SIBM offers comprehensive battery monitoring capabilities. This integrated approach allows for precise control and management, addressing the complexities of modern battery systems.

1.7 Multi-Sensor Integration Challenges

While the integration of multiple sensors offers a comprehensive solution, it poses challenges related to data fusion, calibration, and system complexity. Researchers are actively addressing these challenges to ensure seamless integration and interpretation of data from diverse sensors [6]. Calibration techniques are being developed to harmonize the measurements from different sensors, providing a unified and accurate representation of the battery state.

1.8 Wireless Sensor Networks

The deployment of wireless sensor networks in battery monitoring is gaining attention. Wireless communication reduces wiring complexities, offering scalability and flexibility in monitoring large battery arrays. This approach is particularly beneficial in electric vehicles and renewable energy systems, where minimizing weight and complexity is essential [7].

The use of wireless technologies enhances the accessibility and ease of installation of battery monitoring systems.

1.9. Challenges and Future Directions

Despite significant progress in battery monitoring systems, challenges persist in developing cost-effective and scalable solutions. The complexities associated with multi-sensor integration, data processing, and communication protocols demand continuous research efforts. Machine learning algorithms are being explored to enhance predictive modeling and fault detection in battery systems [8]. These algorithms can leverage large datasets generated by sophisticated sensor arrays to predict and prevent potential issues.

2.0 Machine Learning for Predictive Modeling

Machine learning algorithms, including neural networks and support vector machines, are being applied to predict battery behavior and health based on historical data. These predictive models can identify patterns and anomalies, enabling proactive maintenance and extending battery life. life [9].

Integrating machine learning into BMS contributes to the development of intelligent and adaptive monitoring systems.

2.1 Addressing Scalability Issues

Scalability remains a challenge, especially in large-scale energy storage applications. Researchers are exploring strategies to design monitoring systems that can scale seamlessly with the size of the battery array. This involves the development of modular sensor units, efficient communication protocols, and data aggregation techniques [10].

Ensuring scalability is crucial for the widespread adoption of advanced battery monitoring. technologies.

The literature reviewed underscores the dynamic nature of battery monitoring systems and the continuous efforts to enhance their capabilities. The integration of advanced sensors, wireless technologies, and machine learning algorithms represents a paradigm shift in battery management. The Smart Instrumented Battery Module introduced in this project exemplifies the latest trends in BMS, demonstrating the potential for comprehensive and efficient battery monitoring in diverse applications.

Objectives

1. Design a versatile Smart Instrumented Battery Module (SIBM) integrating advanced sensors for parallel or series battery configurations.

2. Implement real-time multi-parameter monitoring functionality, measuring voltage, current, temperature, and energy for enhanced battery performance.

3. Validate integrated sensor accuracy and reliability through rigorous testing and calibration procedures.

4. Explore SIBM adaptability by connecting two cylindrical battery cells in different configurations, assessing performance across diverse architectures.

5. Contribute to Battery Management Systems (BMS) advancements, influencing the trajectory of monitoring technologies for more efficient energy storage solutions.

Method

The following is the complete method of this project:

- Initialize the Arduino Uno microcontroller and configure the necessary I/O pins.
- Connect the voltage sensor to one of the analog input pins of the Arduino Uno.
- Connect the current sensor to another analog input pin of the Arduino Uno.
- Connect the temperature sensor to another analog input pin of the Arduino Uno.
- Connect the thermocouple sensor to another analog input pin of the Arduino Uno, using the MAX6657 chip to amplify and convert the thermocouple signal.
- Connect the relay to one of the digital output pins of the Arduino Uno, using a transistor and a diode to drive the relay coil.
- Connect the LCD display to the Arduino Uno using the I2C protocol, using the SDA and SCL pins of the Arduino Uno.
- Implement a function to read the voltage from the voltage sensor using the analogRead() function of the Arduino Uno, and convert the analog value to the voltage value using the map() function and the voltage sensor specifications.
- Implement a function to read the current from the current sensor using the analogRead() function of the Arduino Uno, and convert the analog value to the current value using the map() function and the current sensor specifications.
- Implement a function to read the temperature from the temperature sensor using the analogRead() function of the Arduino Uno, and convert the analog value to the temperature value using the map() function and the thermistor equation.
- Implement a function to read the temperature difference from the thermocouple sensor using the analogRead() function of the Arduino Uno, and convert the analog value to the temperature difference value using the map() function and the thermocouple sensor specifications.
- Develop a control algorithm to monitor and regulate the parameters of the battery pack, such as the voltage, current, temperature, and state of charge, and perform actions such as balancing, charging, and discharging, using the data from the sensors and the relay.
- Write a function to set the state of the relay using the digitalWrite() function of the Arduino Uno, and control the connection or disconnection of the battery pack from the charger or the load, depending on the conditions of the cells.
- Implement a function to display the parameters and status of the battery pack on the LCD display, using the LiquidCrystal_I2C library and the lcd.print() function of the Arduino Uno.
- Set up a timer to periodically call the functions to read the sensors, update the relay, and display the LCD, using the millis() function of the Arduino Uno and a variable to store the previous time.

- Build the circuit by connecting the components according to the schematic and layout diagrams of the BMS, using prototyping techniques such as soldering, breadboarding.
- Program the Arduino Uno microcontroller using the Arduino IDE software tool and upload the firmware using the USB cable.
- Test the BMS using passive elements and Arduino by varying the voltage, current, and temperature of the battery pack and observing the changes in relay state and LCD display.
- Document the results, including voltage range, current range, temperature range, state of charge range, relay response, and any observations or issues encountered during testing.

CODE

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <max6675.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);  // Set the LCD address to 0x20 for a 16x2 display
// Floats for ADC voltage & Input voltage
float adc_voltage = 0.0;
float in_voltage = 0.0;

// Floats for resistor values in divider (in ohms)
float R1 = 30000.0;
float R2 = 7500.0;

// Float for Reference Voltage
float ref_voltage = 5.0;
// Integer for ADC value
int adc_value = 0;

// define the thermistor pins in an array
const int thermistor_output[] = {A2, A3};
const int thermistor_Pack= A4;
// define the number of thermistors
const int thermistor_count = 2;

// define the constants for the Steinhart-Hart equation
#define THERMISTORNOMINAL 10000
#define TEMPERATURENOMINAL 25
#define NUMSAMPLES 5
#define BCOEFFICIENT 3950
#define SERIESRESISTOR 10000

// Declare variables to store the time values
unsigned long t1, t2, dt;
```

```
   void setup() {
    lcd.init();                      // Initialize the LCD
    lcd.backlight();
    lcd.clear();
    lcd.print("<<----BMS---->>");
    delay(2000);
    lcd.setCursor(0, 0);
    lcd.print("Smart Battery");
     lcd.setCursor(0, 1);
      lcd.print("Module Design");
    delay(1000);
    lcd.clear();
    Serial.begin(9600); /* Define baud rate for serial communication */


    for (int i = 0; i < thermistor_count; i++) {
      pinMode(thermistor_output[i], INPUT);
}

}

void loop() {
// Read the Analog Input
adc_value = analogRead(A0);

// Determine voltage at ADC input
adc_voltage = (adc_value * ref_voltage) / 1024.0;

// Calculate voltage at divider input
in_voltage = adc_voltage / (R2 / (R1 + R2));

// Display Voltage on LCD
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("V: ");
if(in_voltage<1){in_voltage=0;}
lcd.print(in_voltage, 3);
lcd.print("V");

float Current = 0;
 t1 = millis();//time befor current measurement
for(int i = 0; i < 1000; i++)
{
  Current = Current + (.0264 * analogRead(A1) -13.51) / 1000;
  delay(1);
}
// Get the second time value after current measurement
t2 = millis();
// Calculate the time difference in seconds
```

```
 dt = (t2 - t1) / 1000.0;


 // Display Current on LCD
 lcd.setCursor(0, 1);
 if(in_voltage<1){Current=0;}
 lcd.print("Current: ");
 lcd.print(Current, 3);
 lcd.print("A");
 Serial.println(Current,4);
 delay(2000);




// Multiply voltage and current to get power
 float power = in_voltage * Current;

 // Display Power on LCD
 lcd.clear();
 lcd.setCursor(0, 0);
   lcd.print("P: ");
   lcd.print(abs(power),4);
   lcd.print("W");

   // Use the formula with the time difference
 float Energy = in_voltage * Current * dt;


 // Display Energy on LCD
 lcd.setCursor(0, 1);
   lcd.print("E: ");
   lcd.print(Energy, 4);
   lcd.print("kWh");

 delay(2000);
 // Thermistor for T-discrepency.............................
 // declare an array to store the temperature values
 float temperature[thermistor_count];
 Serial.println(thermistor_count);
 // iterate over the thermistor pins and calculate the temperature values
 for (int i = 0; i < thermistor_count; i++) {
   // declare a variable to store the average of the samples
   float average = 0;
   // take NUMSAMPLES samples from the thermistor and add them to the average
   for (int j = 0; j < NUMSAMPLES; j++) {
     average += analogRead(thermistor_output[i]);
     delay(10);
   }
   // divide the average by NUMSAMPLES
```

```
  average /= NUMSAMPLES;
  // convert the average to resistance
  average = 1023 / average - 1;
  average = SERIESRESISTOR / average;
  // apply the Steinhart-Hart equation to get the temperature in Celsius
  temperature[i] = average / THERMISTORNOMINAL; // (R/Ro)
  temperature[i] = log(temperature[i]); // ln(R/Ro)
  temperature[i] /= BCOEFFICIENT; // 1/B * ln(R/Ro)
  temperature[i] += 1.0 / (TEMPERATURENOMINAL + 273.15); // + (1/To)
  temperature[i] = 1.0 / temperature[i]; // Invert
  temperature[i] -= 273.15; // convert to C
 }



// calculate the temperature discrepancy between the first and second thermistors
 float discrepancy = temperature[1] - temperature[0];

// print the results on the serial monitor
 for (int i = 0; i < thermistor_count; i++) {
  Serial.print("Temperature ");
  Serial.print(i + 1);
  Serial.print(" = ");
  Serial.print(temperature[i]);
  Serial.println(" *C");
 }
 Serial.print("Discrepancy = ");
 Serial.print(discrepancy);
 Serial.println(" *C");

 // display the first and second temperatures on the LCD
 lcd.clear(); // clear the LCD screen
 lcd.setCursor(0, 0); // set the cursor to the first row
 lcd.print("T_Cell_1="); // print the label for the first temperature
 lcd.print(temperature[0],2); // print the value of the first temperature
 lcd.print("*C"); // print the unit
 lcd.setCursor(0, 1); // set the cursor to the second row
 lcd.print("T_Cell_2="); // print the label for the second temperature
 lcd.print(temperature[1],2); // print the value of the second temperature
 lcd.print("*C"); // print the unit
 delay(2000); // wait for 2 seconds

 // display the discrepancy
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("T_D="); // print the label for the discrepancy
 lcd.print(abs(discrepancy));
 lcd.print("*C");
```

```
// ////////////////////T PAck///////////////
 int thermistor_adc_val;
double output_voltage, thermistor_resistance, therm_res_ln, temperature_p;
thermistor_adc_val = analogRead(thermistor_Pack);
output_voltage = ( (thermistor_adc_val * 5.0) / 1023.0 );
thermistor_resistance = ( ( 5 * ( 10.0 / output_voltage ) ) - 10 ); /* Resistance in kilo ohms
*/
thermistor_resistance = thermistor_resistance * 1000 ; /* Resistance in ohms   */
therm_res_ln = log(thermistor_resistance);
/*  Steinhart-Hart Thermistor Equation: */
/*  Temperature in Kelvin = 1 / (A + B[ln(R)] + C[ln(R)]^3)   */
/*  where A = 0.001129148, B = 0.000234125 and C = 8.76741*10^-8  */


     temperature_p = ( 1 / ( 0.001129148 + ( 0.000234125 * therm_res_ln ) + ( 0.0000000876741
    * therm_res_ln * therm_res_ln * therm_res_ln ) ) ); /* Temperature in Kelvin */
temperature_p =( (temperature_p - 273.15)/2.3);  /* Temperature in degree Celsius */
//////Serial monitor display
Serial.print("Temperature in degree Celsius = ");
Serial.print(temperature_p);
Serial.print("\t\t");
Serial.print("Resistance in ohms = ");
Serial.print(thermistor_resistance);
Serial.print("\n\n");

 lcd.setCursor(0, 1);
lcd.print("T_Pack="); // print the label for the discrepancy
lcd.print(temperature_p);
lcd.print("*C");
delay(2000)
```

Components List

1. The **circuit components** used in Proteus:
2. Battery pack: two cylindrical lithium-ion cells (21700 format) with a nominal voltage of 3.7 V and a capacity of 5 Ah each
3. Voltage sensor: a device that measures the voltage across the battery pack, with a resolution of 10 mV and a measurement range of 0 to 5 V.
4. Current sensor: a device that measures the current flowing through the battery pack, based on the hall-effect principle, with a sensitivity of 185 mV/A and a measurement range of -5 to 5 A
5. Temperature sensor: a device that measures the ambient temperature around the battery pack, based on a thermistor, with a nominal resistance of 10 kΩ at 25 °C and a temperature coefficient of -4.4%/°C.
6. Thermocouple sensor: a device that measures the temperature difference between the positive and negative terminals of each cell, with a sensitivity of 40 µV/°C and a measurement range of -40 to 125 °C.
7. Arduino Uno board: a microcontroller board that acts as the main controller of the BMS, with 14 digital input/output pins and 6 analog input pins.
8. Relay: We can add a switch that can connect or disconnect the battery pack from the charger or the load, controlled by the Arduino Uno board, with a coil voltage of 5 V and a contact rating of 10 A
9. LCD display: a user interface that can display information and feedback on the battery pack status and health, connected to the Arduino Uno board using the I2C protocol, with a resolution of 16x2 characters and a backlight.

Simulation Circuit Diagram

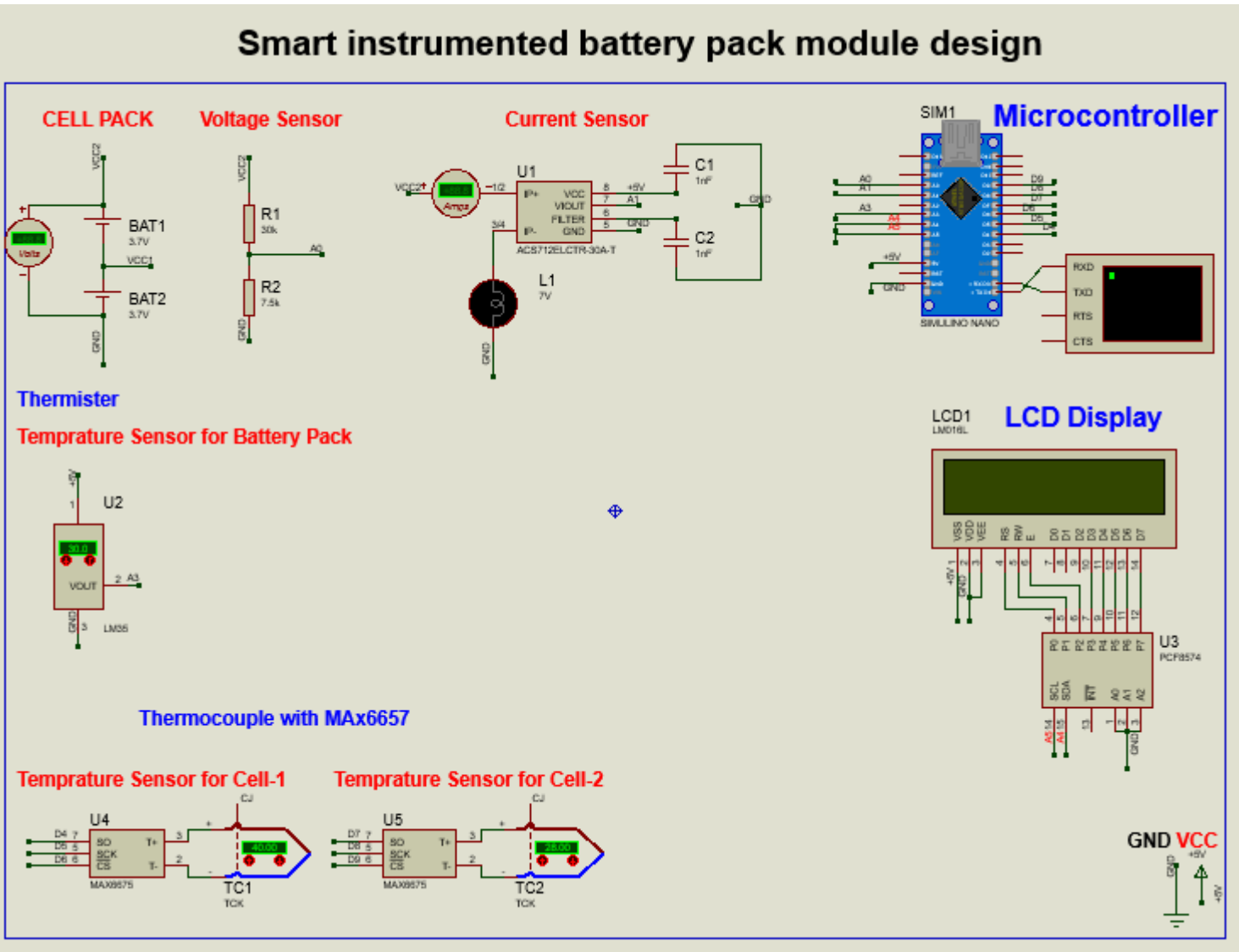

Fig1: simulation design with lm35 and Thermocouple and max6657

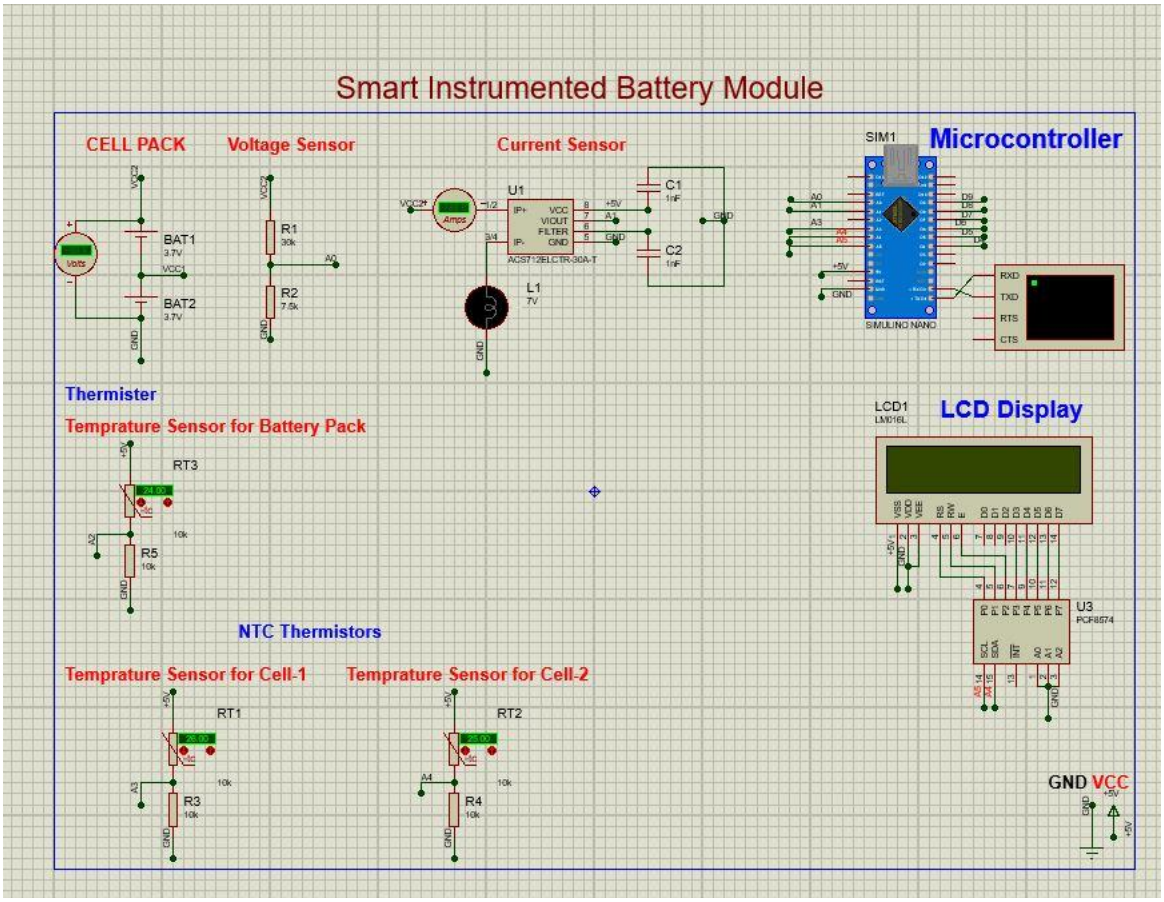

Fig2: simulation design with thermistors

Simulation Results

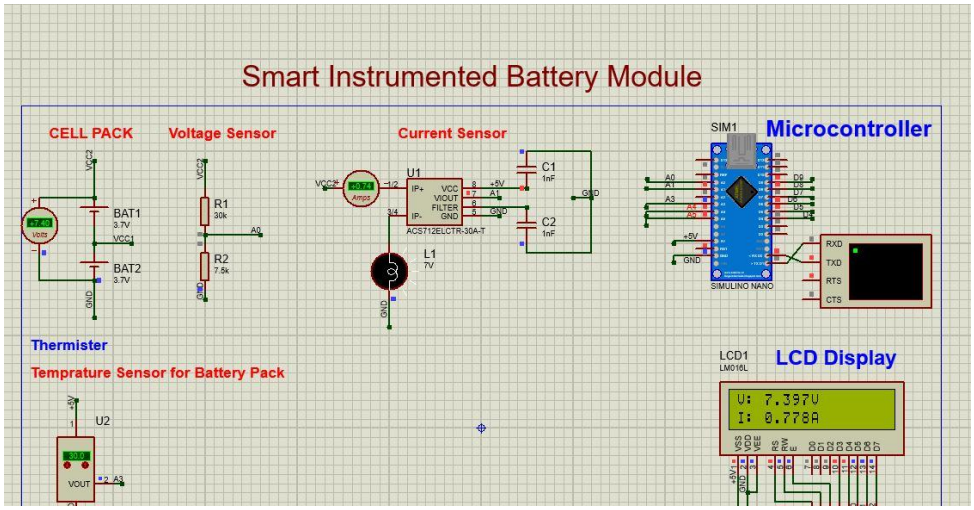

Fig3: Voltage and Current display

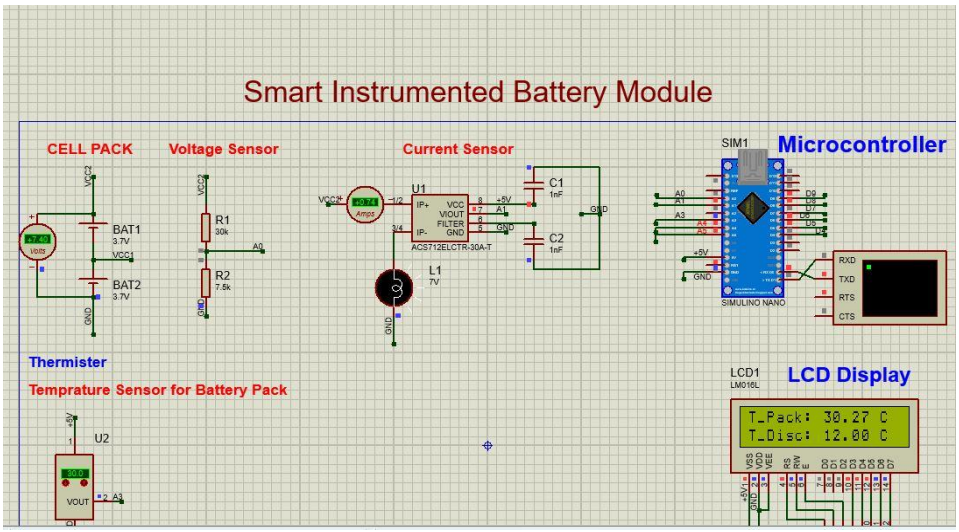

Fig4:T Disc and Temp of Pack display

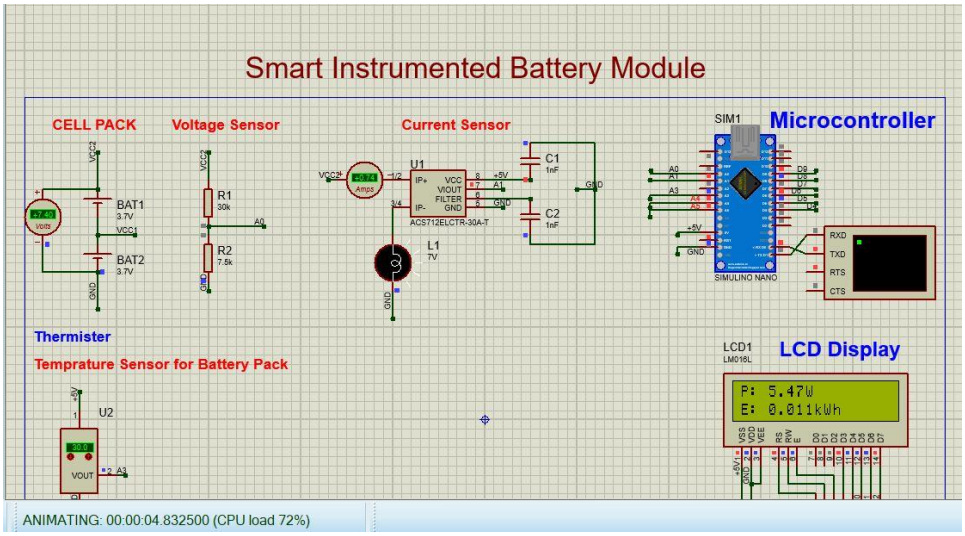

Fig5: Power and Energy display

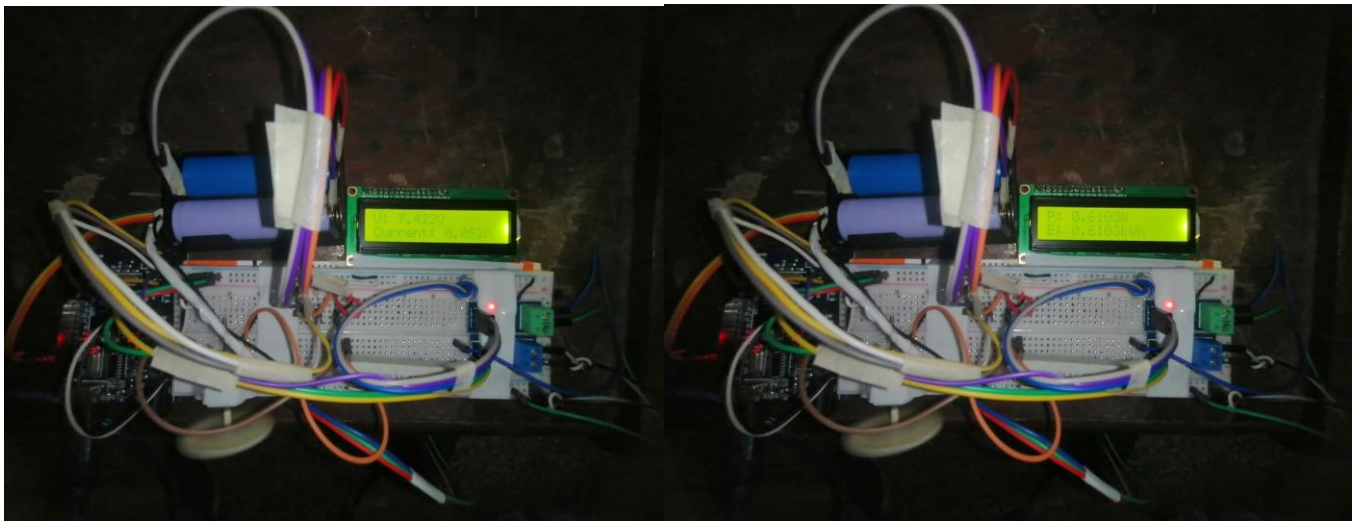

Fig6: Display of Voltage, Current, Power & Energy.

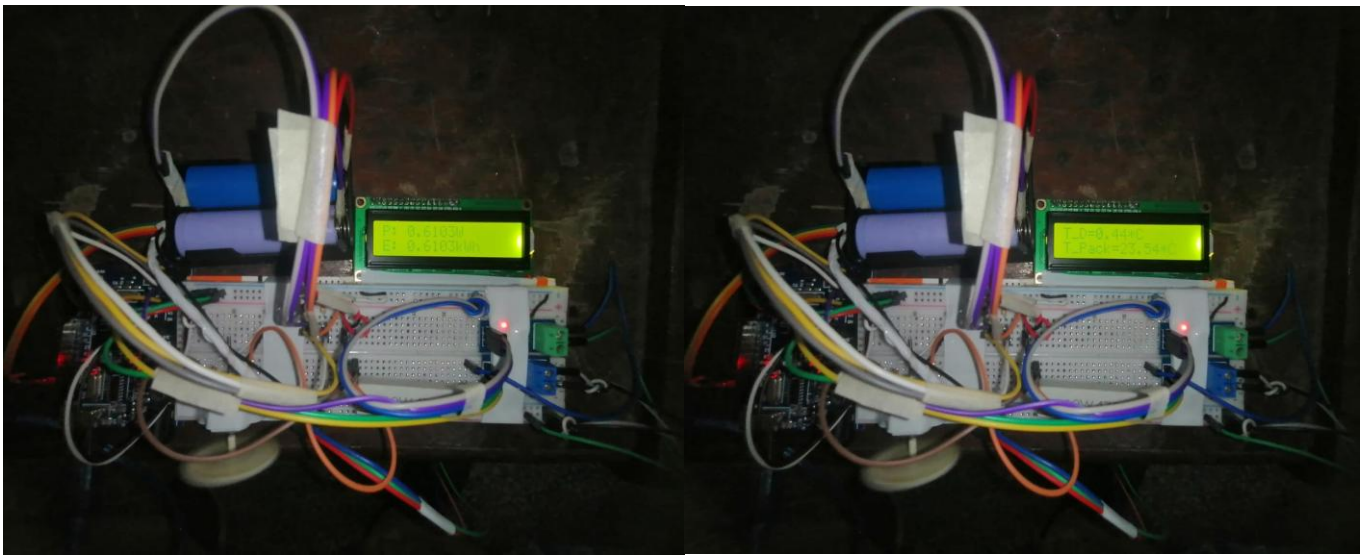

Fig7: Display of Tcell1, Tcell2, T discrepancy & Temp of pack.

Conclusion

In conclusion, the Smart Instrumented Battery Module stands as a versatile and effective solution, revolutionizing the monitoring of crucial battery parameters. The seamless integration of multiple sensors ensures precise measurements, paving the way for enhanced battery management across various applications. The successful implementation of the SIBM in measuring power, energy, and temperature discrepancies validates its potential for widespread adoption in diverse battery systems. With its innovative design and comprehensive functionality, the SIBM not only addresses current monitoring needs but also sets the stage for the next generation of intelligent and adaptive Battery Management Systems such as Machine Learning based Algorithm design for Better Optimization of battery.

References

1. Valoen, L. O., & Nærland, T. (2003). Battery Management Systems for Large Lithium-Ion Battery Packs. IEEE Transactions on Energy Conversion, 18(3), 392–399.

2. Karden, E., & Ho, C. (2011). Battery Management Systems for Large Lithium-Ion Battery Packs. Journal of Power Sources, 196(1), 668–677.

3. Ayers, J. K., & Hadsell, E. (2002). Current Sensor Developments for Battery Management Systems. IEEE Aerospace and Electronic Systems Magazine, 17(1), 15–19.

4. Steinhart, H., & Hart, S. R. (1968). Calibration Curves for Thermistors. Deep Sea Research and Oceanographic Abstracts, 15(4), 497–503.

5. Zhang, Y., & Zhao, H. (2008). Power and Thermal Characterization of Lithium-Ion Battery Packs: Experimental and Simulation Investigation. Journal of Power Sources, 185(2), 1433–1440.

6. Wan, C., Xu, J., Li, Y., & Hu, L. (2019). A Review on Current Sensing Technologies for Battery Management System: Challenges and Opportunities. Journal of Energy Storage, 25, 100844.

7. Zhang, C., Huang, Z., Li, J., Wang, J., & Cao, J. (2018). Wireless Sensor Networks in Battery Management Systems for Electric Vehicles: A Review. Energies, 11(12), 3354.

8. Wang, H., & Kroposki, B. (2011). Advanced Battery Management and Control in Vehicle Electrification. IEEE Transactions on Industrial Electronics, 58(4), 1066–1075.

9. Hu, X., Zhang, H., Han, Z., & Cao, B. (2017)