TABLE OF CONTENT :

1.ABSTRACT

In recent years, the demand for efficient and safe battery systems has significantly increased due to the growing use of batteries in electric vehicles, renewable energy storage, and portable electronics. One of the critical challenges in battery operation is thermal management. Improper temperature regulation can lead to reduced battery efficiency, shortened lifespan, or even dangerous thermal runaway conditions. This project proposes a Battery Thermal Management System (BTMS) designed to monitor and regulate battery temperature using a microcontroller-based control mechanism..

The system employs the LM35 temperature sensor to accurately detect the real-time temperature of the battery. The analog signal generated by the LM35 is fed into the arduino, which processes the temperature data. If the temperature exceeds a predefined threshold, the system activates a cooling fan to lower the battery's temperature. Simultaneously, an alarm buzzer is triggered to alert the user of the overheating condition, and the current temperature is displayed on an LCD module.

Integration of an LCD display ensures user-friendly monitoring of the battery status, while the buzzer provides a immediate auditory warning. The system also includes a transistor-based driver circuit to control the high-power fan using the microcontroller's output. A freewheeling diode is placed across the fan to protect the transistor from back EMF.

This BTMS can be particularly beneficial in applications where battery health is critical, such as electric vehicles, solar power storage systems, and industrial automation. The simplicity and low cost of the components make it a practical solution for small- to medium-scale applications. Additionally, the system can be easily enhanced with wireless communication for remote monitoring and control.

In conclusion, this project demonstrates a cost-effective, real-time solution for battery temperature monitoring and control. The intelligent use of sensors, actuators, and microcontrollers makes the system reliable and scalable. Future improvements can include automatic logging of temperature data and integration with IoT platforms for predictive maintenance and analytics.

PREPARATION OF PCB LAYOUT OF A CIRCUIT USING SUITABLE SOFTWARE

1.INTRODUCTION

Printed Circuit Boards (PCBs) play a vital role in the functioning and fabrication of modern electronic devices. They provide the mechanical support and electrical interconnections necessary for components to function as a complete circuit. The process of preparing a PCB layout involves translating a theoretical circuit design into a physical layout that can be manufactured. This experiment was undertaken by a group of four members with the objective of understanding the steps required to design a PCB layout using suitable electronic design automation (EDA) software, such as KiCad or Eagle. The focus was on learning the complete design cycle, from schematic capture to final layout generation and Gerber file preparation.

2. OBJECTIVE OF THE EXPERIMENT

The primary aim of this study experiment was to familiarize the students with the practical aspects of PCB design. The specific objectives were:

- To understand the purpose and structure of a PCB.
- To learn how to use suitable PCB design software.
- To convert an electronic circuit into a schematic using the software.
- To design the corresponding PCB layout.
- To generate output files required for PCB fabrication.
- To understand best practices and design constraints in PCB layout design.

3. MATERIALS AND SOFTWARE

The following resources were used during the course of the experiment:

- Hardware:

  Desktop/Laptop with appropriate system specifications.

- Software:

  KiCad EDA (open-source PCB design tool) or Eagle (Autodesk).

- Documents and References:

  Circuit diagrams (manually designed or provided).

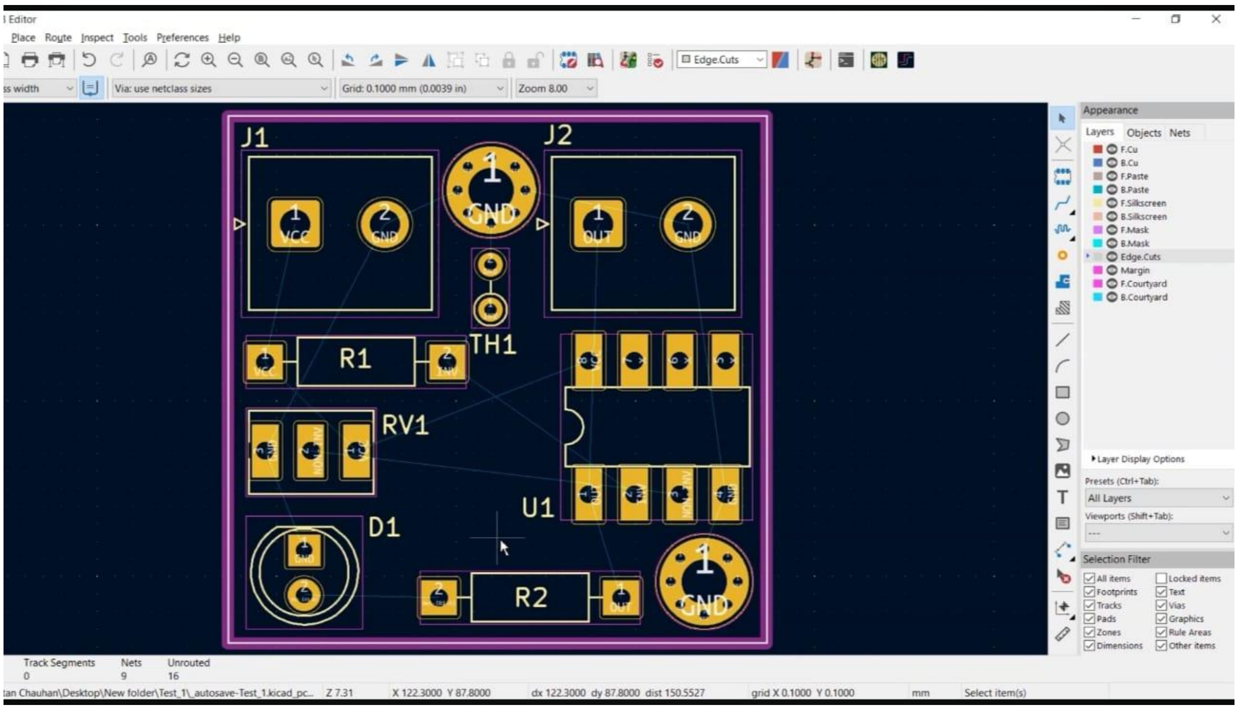
  Component datasheets.

  PCB design tutorials and manuals for the selected software.

4. METHODOLOGY

The experiment followed a step-by-step methodology, which is outlined below:

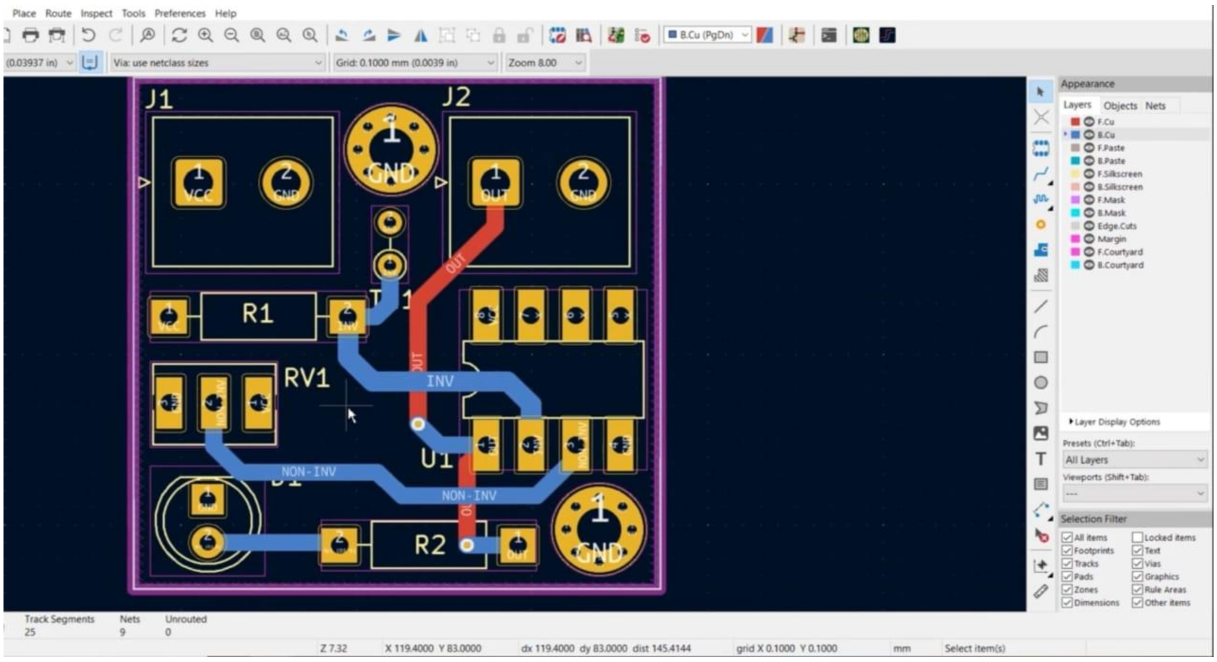
1. Selection of Circuit The group selected a simple electronic circuit for which the PCB layout was to be prepared. The circuit was chosen based on ease of understanding, number of components, and suitability for layout within limited board dimensions. In this case, a 555 timer-based LED flasher circuit was selected.
2. Schematic Design The chosen circuit was drawn in the schematic editor of the selected software. Each component was placed, named, and interconnected using nets. care was taken to assign appropriate values and reference designators to each component.
3. Electrical Rules Check (ERC) Once the schematic was complete, the design was checked for electrical errors such as unconnected pins or multiple net labels. The ERC tool provided warnings and suggestions for corrections.
4. Footprint Assignment Each component in the schematic was assigned a physical footprint. These footprints represent the actual land patterns of components on the PCB. Libraries provided by the software were used, and in some cases, custom footprints were created.
5. PCB Layout Creation Using the layout editor, a new PCB file was created and the schematic was imported into it. All component footprints appeared in a separate area, which were then manually arranged on the board area based on considerations like signal flow, heat dissipation, and space optimization.
6. Design Rule Check (DRC) After placing and routing all tracks, a Design Rule Check was performed. This verified that the layout conformed to manufacturing constraints like minimum track width, clearance, and pad sizes.
7. Generating Output Files Once the design was validated, Gerber files were generated. These files are used by manufacturers to fabricate the PCB. Additional files such as drill files and BOM (Bill of Materials) were also prepared.



```
Track Segments   Nets   Unrouted
0                9      16
```

```
tan Chauhan\Desktop\New folder\Test_1\_autosave-Test_1.kicad_pc...   Z 7.31   X 122.3000  Y 87.8000   dx 122.3000  dy 87.8000  dist 150.5527   grid X 0.1000  Y 0.1000   mm   Select item(s)
```

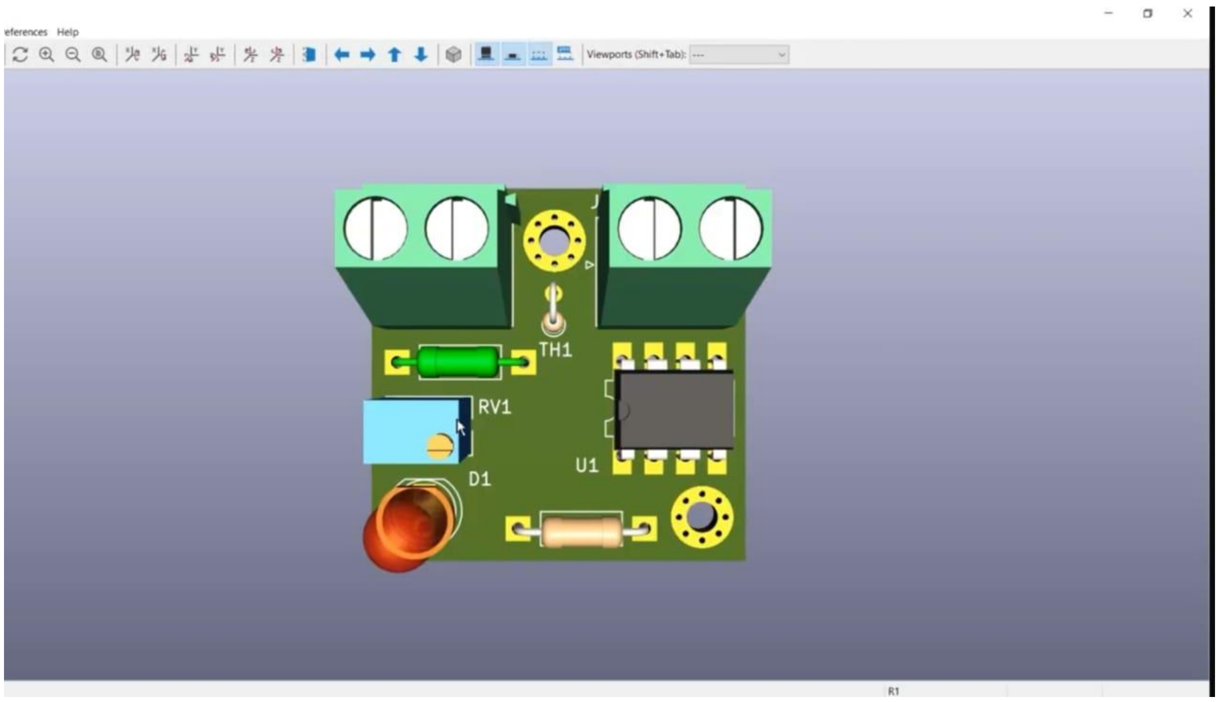
EDITING FOOTPRINT



```
Track Segments   Nets   Unrouted
25               9      0
```

```
Z 7.32   X 119.4000  Y 83.0000   dx 119.4000  dy 83.0000  dist 145.4144   grid X 0.1000  Y 0.1000   mm   Select item(s)
```

PCB ROUTING.



EXTERNAL 3D MODEL.

5. PCB DESIGN CONSIDERATIONS

Several technical considerations were kept in mind while preparing the PCB layout:

1. Component Placement: Components were placed to minimize the path of critical signals and reduce electromagnetic interference.
2. Trace Width and Spacing: Appropriate trace widths were used depending on current requirements.
3. Power and Ground Planes: Ground and power traces were given priority in routing for circuit stability.
4. Via Usage: Minimal vias were used to maintain signal integrity and reduce complexity.
5. Board Shape and Size: The final layout was confined to a predefined board size for compatibility with the enclosure.

6. CHALLENGES FACED

During the course of this experiment, the group encountered several challenges:

1. Some component footprints were not readily available and had to be created manually.

2. Routing in tight spaces proved difficult, especially when dealing with multiple components and signal crossings.
3. Understanding the different output file formats and their purposes required additional research.
4. Minor errors in the schematic resulted in repeated corrections during layout.

However, these challenges provided valuable learning opportunities and helped the team develop problem-solving skills.

7. RESULTS AND OBSERVATIONS

The group successfully created a functional PCB layout for the selected circuit using KiCad software. The Gerber files were visually verified using a Gerber viewer to ensure accuracy. Observations included:

1. Proper placement of decoupling capacitors improved the layout.
2. The use of grid alignment tools made component arrangement more systematic.
3. Using color-coded nets helped in visualizing signal paths during routing.
4. The final layout was neat, compact, and manufacturable, adhering to basic design principles.

8. ADVANTAGES OF USING PCB DESIGN SOFTWARE

The use of EDA tools like KiCad or Eagle offers numerous advantages:

- Speeds up the design process through component libraries and auto-routing.
- Ensures accuracy through design rule checks.
- Simplifies the manufacturing process via standardized output files.
- Offers 3D visualization of the board for verification.
- Supports multi-layer design for complex circuits.

9. SCOPE FUTURE

Further study and practice in PCB layout design can lead to the following:

- Design of multi-layer PCBs for high-speed circuits.
- Implementation of surface-mount technology (SMT) for compact devices.

- Integration with simulation tools for real-time analysis.
- Automation of component placement using AI tools.
- Contribution to open-source hardware design repositories

10. CONCLUSION

This experiment enabled the group to gain hands-on experience with PCB layout design. From schematic capture to output file generation, each step involved critical thinking and application of theoretical concepts. The knowledge of using PCB design software equips students with an essential skill for future projects and real-world electronics design. The success of this experiment has laid a strong foundation for more advanced PCB design tasks, including multi-layer boards and embedded system layouts

STUDY OF SENSORS AND COMMUNICATION PROTOCALS IN IOT:

SENSORS

WHAT IS SENSOR?

A sensor is a device that detects and responds to changes in its environment by converting physical, chemical, or biological signals into electrical signals. Sensors are used in various applications,including automation, healthcare, transportation, and environmental monitoring.

WORKING:

Basic Working Principle of Sensors

Detection – The sensor detects a physical, chemical, or biological change (e.g., temperature, pressure, motion).

Conversion – The detected change is converted into an electrical signal.

Processing – The signal is amplified, filtered, or conditioned for accuracy.

Output – The processed signal is sent to a display, controller, or system for further action.

EXAMPLE:

Temperature Sensors: Thermocouples generate voltage based on temperature differences, while thermistors change resistance with temperature variations.

Pressure Sensors: Piezoelectric sensors generate electrical charge when pressure is applied.

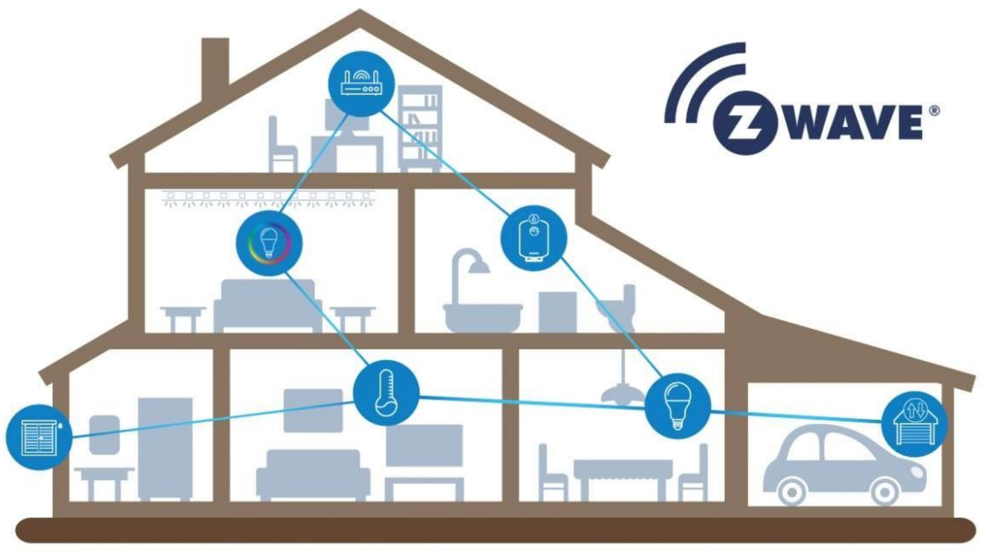
COMMUNICATION PROTOCALS:

1.HTTP (HYPERTEXT TRANSFER PROTOCOL)

HTTP is a high-level application-layer protocol used for web communication and is not a physical transmission protocol like WiFi. It operates over TCP/IP networks and is one of the most widely used protocols in IoT devices that interact with cloud-based platforms. HTTPS (secured with SSL/TLS encryption) ensures data security and integrity during transmission. Cloud-connected IoT applications (smart home hubs, industrial monitoring dashboards).

2. ZIGBEE

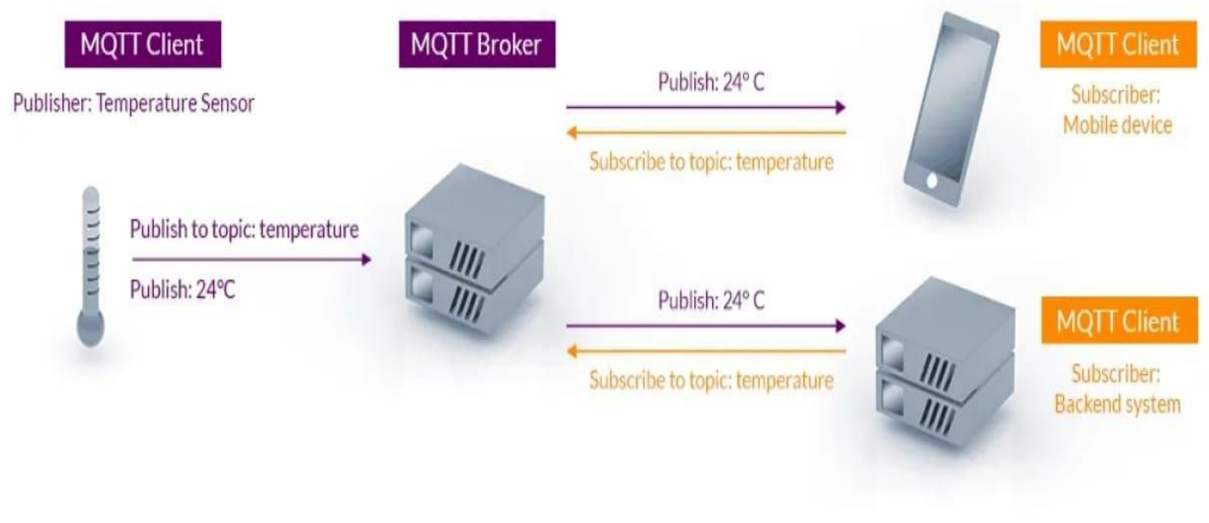
Zigbee is a low-power, low-data-rate protocol that uses mesh networking for extensive coverage. It operates across multiple frequencies (868 MHz, 915 MHz, and 2.4 GHz). Low power consumption and short-range wireless communication. Operates on IEEE 802.15.4 standard. Mesh networking for better coverage. Ideal for home automation, smart lighting, and healthcare devices. Supports AES-128 encryption for security.



5.MQTT

MQTT is a lightweight messaging protocol using a publish/subscribe model, optimized for low-bandwidth and low-power IoT devices. It is widely adopted in IoT platforms and supports real-time communication between devices and cloud services. Lightweight and efficient for low-bandwidth networks. Publish-subscribe messaging model. Supports Quality of Service (QoS) levels for reliable message delivery. Secure communication using TLS encryption. Ideal for industrial IoT, smart homes, and remote monitoring

MQTT has become one of the most popular protocols in IoT communication because of its ability to operate reliably even in constrained network environments. It uses a central broker to manage all message exchanges, which helps simplify the architecture of connected systems. Devices do not need to know about each other directly; instead, they only need to know which topics to publish or subscribe to, making the system highly decoupled and scalable. Security in MQTT is typically handled through TLS encryption and username/password authentication, making it suitable for use in sensitive applications. Overall, MQTT offers a balance of performance, reliability, and simplicity, which makes it a preferred choice for developers building connected systems and real-time data monitoring solutions.



SENSORS AND COMMUNICATION PROTOCALS USED IN BATTERY THERMAL MANAGEMENT IN ELECTRIC VEHICLE:

Sensors Used in Battery Thermal Management is

1. **Temperature Sensors** – These are crucial for detecting heat variations in battery cells. They help maintain the battery within safe operating temperatures to prevent thermal runaway.

The **LM35** is a precision temperature sensor that provides an output voltage linearly proportional to the temperature in Celsius. It is widely used in electronics and embedded systems for temperature monitoring.The internal block diagram of the LM35 consists of several key functional blocks that work together to measure temperature and convert it into a proportional analog voltage. At the core of the LM35 is a temperature-sensing element, usually a silicon bandgap-based sensor, which produces a small voltage that changes predictably with temperature. This voltage signal is very small, so it is first fed into a precision amplifier circuit that increases its level to a usable range while maintaining high accuracy and linearity.

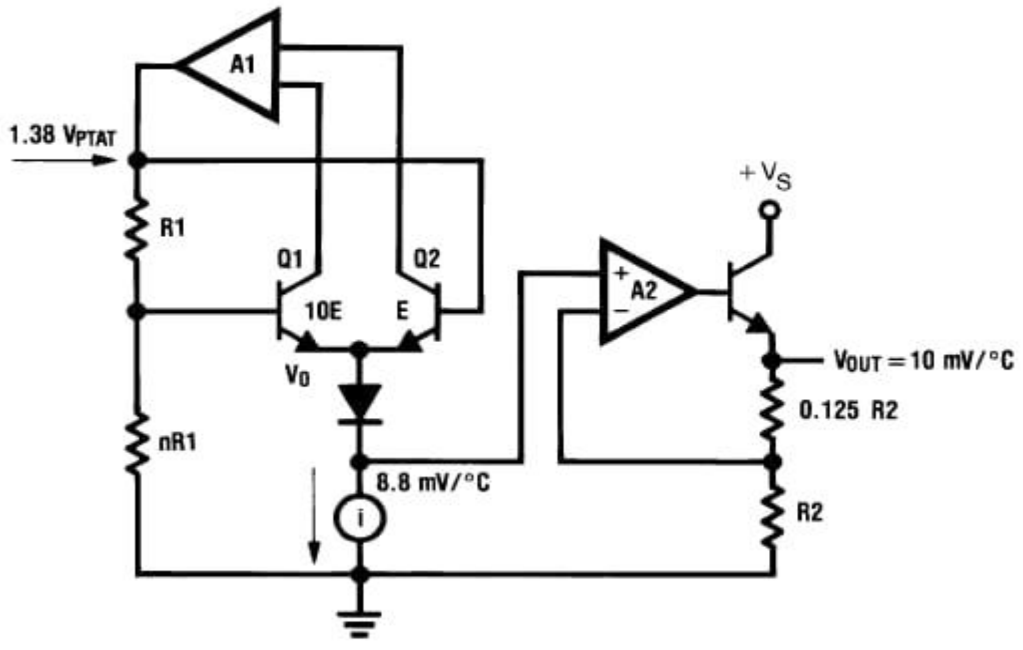
To ensure stability and consistent performance, the sensor also includes biasing circuitry and voltage reference blocks, which help maintain a fixed operating point for internal components regardless of external variations. In addition, linearity correction circuitry is included to ensure that the output voltage remains directly proportional to temperature over the operating range. Finally, the processed and amplified analog signal is sent to the output stage, which delivers a voltage of 10 mV per degree Celsius (e.g., 250 mV for 25°C), making it easy to interpret using an ADC or even a basic voltmeter.

KEY FEATURES OF LM35

- **Calibrated directly in Celsius** (no need for conversion from Kelvin).
- **Linear output**: 10mV per °C.
- **High accuracy**: ±0.5°C at 25°C.
- **Low power consumption**: Draws only 60µA.
- **Wide operating range**: -55°C to 150°C.
- **Low self-heating**: Less than 0.1°C in still air.

WORKING PRINCIPLE

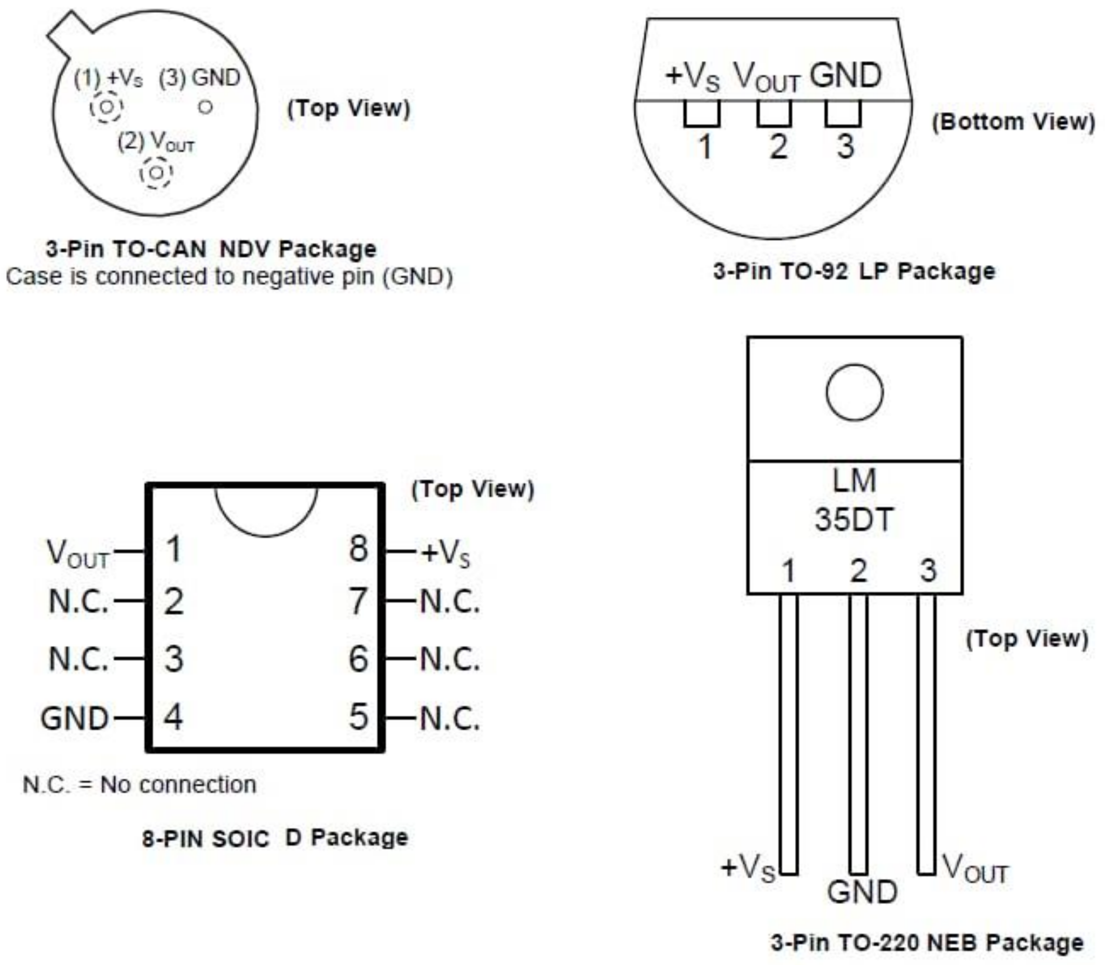
The LM35 operates by measuring temperature and converting it into a proportional voltage. The output voltage increases by 10mV for every 1°C rise in temperature. This voltage can be read by an ADC (Analog-to-Digital Converter) in microcontrollers like Arduino or Raspberry Pi.



INTERNAL BLOCK DIAGRAM OF LM35

ROLE OF LM35 IN BATTERY THERMAL MANAGEMENT

- **Real-time Temperature Monitoring** – Continuously measures battery temperature to prevent overheating.
- **Linear Output (10mV/°C)** – Provides precise temperature readings without complex calibration.
- **Low Power Consumption (60µA)** – Ensures minimal energy drain on the battery system.
- **Fast Response Time** – Quickly detects temperature fluctuations for efficient thermal regulation.



LM35 Sensor Packaging

COMMUNICATION PROTOCALS IN BATTERY THERMAL MANAGEMENT:

CAN Bus (Controller Area Network) – A widely used protocol in EVs for real-time data exchange between the Battery Management System (BMS) and other vehicle components.

How CAN Bus Works in Battery Thermal Management?

Data Transmission – CAN Bus allows seamless communication between battery sensors, cooling systems, and the vehicle control unit (VCU).

Real-Time Monitoring – It continuously tracks battery temperature, voltage, current, and state of charge (SoC).

Fault Detection – CAN Bus helps identify overheating issues and triggers safety mechanisms.

Energy Optimization – Ensures balanced energy distribution among battery cells to enhance performance.

CONCLUSION

In battery thermal management systems for electric vehicles, the combined use of various sensors and communication protocols ensures safe and efficient operation. Sensors like LM35, DHT22, pressure, voltage, and current sensors provide critical real-time data, while protocols such as CAN Bus, I2C, UART, Zigbee, and MQTT facilitate reliable and timely communication. These technologies collectively form the backbone of smart, safe, and energy-efficient battery management in modern electric vehicles.

PROTOTYPE

BATTERY THERMAL MANAGAMENT SYTSEM

1.ABSTRACT

This project presents a Thermal Battery Management System (TBMS) designed to enhance the safety of lithium-ion batteries. It monitors both temperature and voltage using an LM35 temperature sensor and a voltage divider circuit interfaced with an Arduino Uno. When the battery temperature exceeds a preset threshold (e.g., 40°C), a BC547 transistor activates a fan to cool the system, and a buzzer alerts the user. A 3.7V lithium-ion battery is connected through a switch to measure voltage, which is displayed along with the temperature on a 16x2 LCD. The system is compact, cost-effective, and ideal for small-scale battery-powered applications.

2. INTRODUCTION

Lithium-ion batteries are widely used in portable electronics but are prone to overheating, which can lead to degradation or even explosions. Managing the thermal state and voltage level is critical to ensuring battery longevity and safety. This project introduces an embedded solution that actively monitors these parameters using Arduino. It automatically triggers safety mechanisms like fans and buzzers and provides real-time readings to users through an LCD, thus serving as a low-cost educational or prototyping tool for thermal battery safety.

3. OBJECTIVES

- Monitor real-time battery temperature using LM35 sensor.
- Measure the battery voltage of a 3.7V lithium-ion cell using a voltage divider.
- Turn ON a cooling fan through a BC547 transistor upon overheating.
- Trigger a buzzer for user alerts when unsafe conditions arise.
- Display temperature and voltage on a 16x2 LCD.
- Use a push-button to display battery voltage on demand.
- Ensure a reliable, low-power embedded system for safety monitoring.

4. LITERATURE REVIEW

Past studies emphasize the importance of thermal and voltage monitoring in battery systems. Microcontroller-based BMS designs have proven effective for low-cost, scalable applications. Sensors like LM35 provide accurate linear temperature readings, and voltage divider circuits are commonly used for analog voltage measurements. Previous work often focuses on either thermal or voltage aspects separately. This project uniquely integrates both for enhanced safety in compact systems, reflecting findings from industry practices and academic research.

5. COMPONENTS AND MATERIALS

- Arduino Uno microcontroller
- LM35 temperature sensor
- 3.7V lithium-ion battery
- 5V DC fan
- BC547 NPN transistor
- Buzzer (5V)
- 16x2 LCD (or I2C LCD module)
- Push button switch
- Voltage divider (two 10kΩ resistors)
- Resistors: 1kΩ, 10kΩ
- Breadboard and jumper wires
- USB cable, battery holder

6. SYSTEM ARCHITECTURE

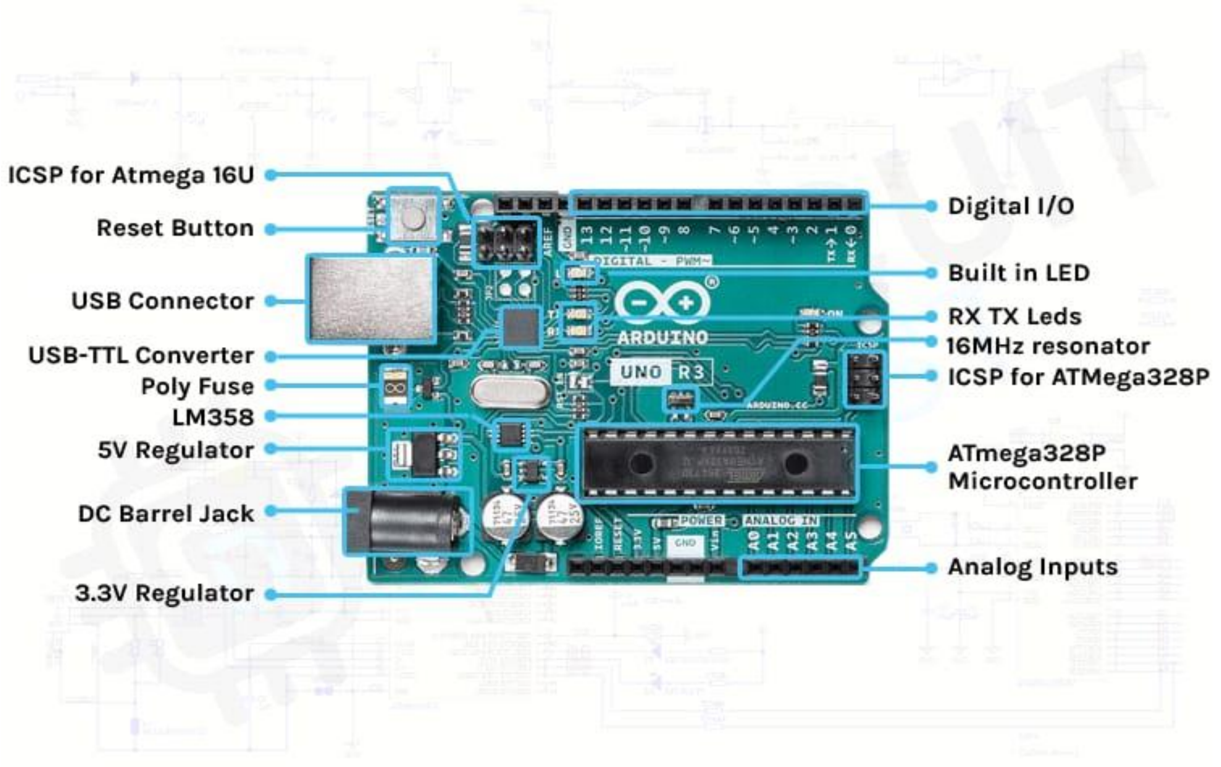
The architecture includes four major subsystems: sensing (LM35 and voltage divider), control (Arduino Uno), actuation (fan and buzzer), and output display (LCD). The LM35 and battery voltage feed analog data to Arduino. The microcontroller processes this data and triggers the fan and buzzer through digital output pins. The system uses a push-button for selective voltage display and powers the circuit through the 3.7V battery interfaced via USB or external adapter.

7. HARDWARE IMPLEMENTATION

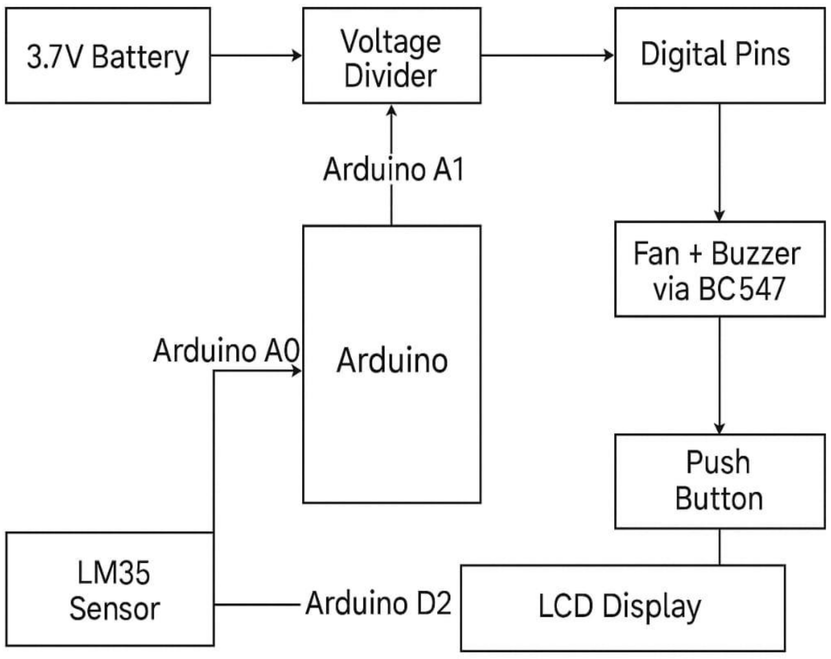
LM35 is attached near the battery to accurately measure its temperature. The 3.7V battery is connected to a voltage divider to scale its voltage for Arduino's 5V analog input. The transistor controls the fan, receiving a signal from Arduino's digital pin. The buzzer connects to another digital pin. A 16x2 LCD displays readings, and the push-button allows on-demand voltage display. All components are mounted on a breadboard and powered from a regulated source for testing.

8. SOFTWARE IMPLEMENTATION

The Arduino code initializes input and output pins, the LCD, and continuously reads analog values from the LM35 and voltage divider. It converts the analog values into temperature (°C) and voltage (V). If the temperature crosses the threshold, the code activates the fan and buzzer. If the push-button is pressed, the voltage is displayed. The LCD shows temperature and voltage in a loop, making the system interactive and reliable.



9. BLOCK DIAGRAM



10. CIRCUIT DIAGRAM

LM35: Vcc → 5V, GND → GND, Output → A0

Voltage Divider: Battery + → R1 → R2 → GND; tap between R1 & R2 → A1

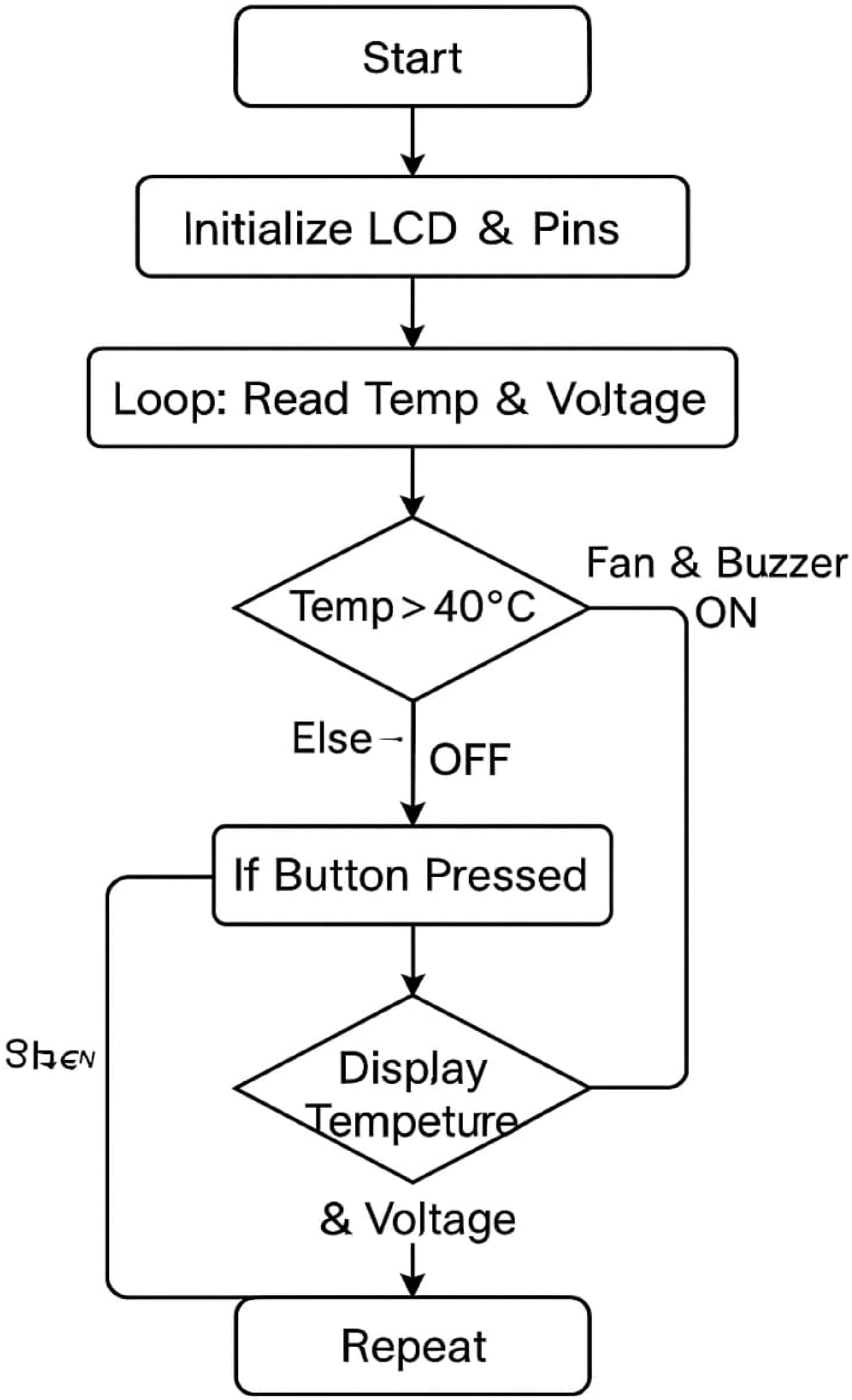
Fan Circuit: BC547 Base → D3 (with 1kΩ), Collector → Fan → 5V, Emitter → GND

Buzzer: D4 → Buzzer +, Buzzer - → GND

LCD: Connected in 4-bit mode to D5-D8 (or I2C to SDA/SCL)

Button: One end → D2, other end → GND (with pull-up)

11. FLOWCHARTS AND LOGIC



12. TESTING AND RESULTS

The system was tested under controlled heating. The LM35 accurately triggered the fan and buzzer at 40°C. The voltage divider circuit provided reliable battery readings (~3.7V) verified with a multi meter. Button response for voltage display was consistent. LCD output was clear and updated in real-time. The project successfully demonstrated combined thermal and voltage management.

13. ADVANTAGES AND LIMITATIONS

ADVANTAGES:

- Real-time monitoring
- Low cost and easy to build
- Protects battery from overheating
- Interactive and educational

LIMITATIONS:

- Only one temperature point
- Manual calibration of voltage
- Not suitable for high-current batteries

14. APPLICATIONS AND FUTURE SCOPE

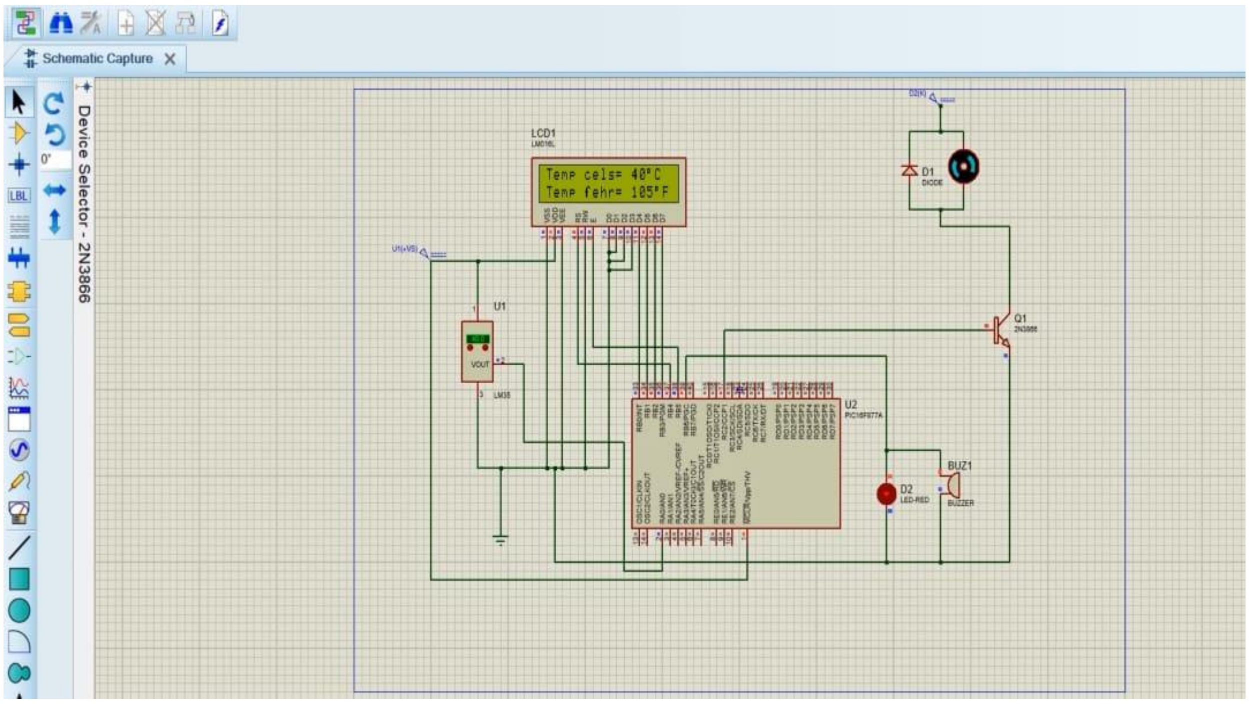
APPLICATIONS:

- Portable battery-powered devices
- DIY power banks
- Educational kits for embedded systems

Future Scope:

- Add IoT alerts (SMS/Bluetooth)
- Measure current & charge level
- Integrate solar charging
- Expand to multi-cell battery packs

15.SIMULATION:



16. REFERENCES

1. Texas Instruments – LM35 Datasheet

2. Arduino.cc – Arduino Uno Documentation

3. BC547 Transistor Datasheet

4. Tutorials Point, Electronics Hub – Voltage Divider Basics

5. Research articles on Battery Management Systems

17. APPENDIX

- ➢ Arduino Source Code
- ➢ Complete Circuit Diagrams
- ➢ LCD and Transistor Pinouts
- ➢ Component Datasheets

18.CONCLUSION

This project provides a compact and affordable thermal battery monitoring system that uses Arduino, LM35, and a 3.7V lithium-ion battery. It demonstrates how embedded systems can safeguard batteries by monitoring temperature and voltage, controlling cooling mechanisms, and alerting users. It is suitable for students, DIY enthusiasts, and developers interested in battery safety and automation

DESIGN THINKING WORKSHEET

(COMBINED HARDWARE AND SOFTWARE PROJECTS)

1.EMPHATHIZE- Understand the user and the Contest

Who is your Target user?

The target users are primarily individuals and organizations that rely on rechargeable battery systems, such as **electric vehicle (EV) manufacturers, renewable energy system**. Other users may include drone manufacturers and mobile power device designers. These users deal with batteries that require strict thermal management to avoid failures. In EVs especially, overheating batteries can result in performance drops, fire hazards, or system shutdowns.

What is their Environment (home, industry, office/college etc.)?

Users typically operate in varied environments such **as industrial factories, automotive assembly lines, research centers, and even some residential setups for solar storage**. In industrial environments, batteries are used for backup and power distribution systems. In EV manufacturing, battery packs are integral and operate under fluctuating temperature conditions. In startups and product prototyping labs, systems are often tested with limited infrastructure. Therefore, these environments require automated, low-maintenance, and accurate monitoring systems.

What specific problem or inconvenience do they face?

The main problem is the **lack of real-time battery temperature management**. Users currently face challenges such as overheating of battery cells, which can lead to battery failure or thermal runaway. Often, the batteries are sealed or placed in inaccessible areas, making manual monitoring impossible.

Without real-time alerts, users may not know when a battery is overheating or releasing hazardous gases. Even if they monitor the battery manually, there is a risk of delay in reacting to a temperature spike. This can result in **reduced battery life, unsafe operation, or complete system failure.**

2.DEFINE – PROBLEM STATEMENT :

In battery-powered systems, especially in electric vehicles and portable electronics, batteries are prone to overheating and gas emission due to continuous operation or charging cycles. Without an automatic monitoring and control system, this can lead to performance degradation, system failure, or even fire hazards. Manual temperature monitoring is not reliable or feasible in real-time applications.

What problem are you solving, and why is it important to the user?

We are solving the problem of **real-time thermal monitoring and automatic cooling in battery systems**. By using temperature and gas sensors connected to a Arduino uno, our system continuously monitors the environment around the battery. When dangerous temperature or gas levels are detected, the system automatically triggers a buzzer for alert and activates a fan (motor) to cool the battery. The LCD display shows current values to the user for transparency and monitoring.

This solution **prevents overheating, reduces the risk of damage, extends battery life, and enhances overall system safety**. It is especially important for users in electric vehicles, power backups, and robotics who need reliable and safe battery performance without constant supervision. By automating the response, the system removes human error and ensures immediate protective action. This is **a low-cost, easy-to-implement solution** that improves battery safety and efficiency across various environments.

3. IDEATE- POSSIBLE SOLUTIONS :

List hardware and software ideas that could solve the problem:

There are several possible solutions to prevent battery overheating and gas leakage.

- Thermal cutoff switches.
- Battery Management Systems (BMS) with advanced algorithms.
- IoT-based monitoring systems.
- Smart cooling jackets.
- Thermal imaging with AI analysis.

What solution are you choosing and why does it stand out?

Our chosen solution:

We are choosing a Aduino-based thermal management system. It uses temperature and gas sensors to continuously monitor the battery environment. If the values cross the safety threshold, a buzzer is triggered and a fan (DC motor) turns on to cool the battery. An LCD displays real-time temperature and gas levels. This solution is cost-effective, easy to build, and suitable for small- to medium-scale applications.

Unlike high-end BMS or IoT systems, it does not rely on internet connectivity or expensive components, making it ideal for educational, prototype, or small industry use. It provides an automatic, fast-responding, and portable safety solution for battery systems.

4. PROTOTYPE – STIMULATE AND MODEL

What hardware components and circuts are used ?

The main hardware components include a **Arduino UNO, LM35 temperature sensor, a 16x2 LCD display, a buzzer, a red LED, a DC motor (fan), and relay** for switching. The LM35 outputs an analog voltage corresponding to temperature, which is connected to the microcontroller's analog pin. The motor and buzzer are connected through relay, allowing the Arduino to switch them ON/OFF based on threshold temperatures.

What software or code modules will control or support the hardware?

We used **Arduino IDE** for arduino to write and compile the Embedded C code that reads the sensor data using ADC (analog-to-digital conversion), processes it, and controls outputs. **Proteus Design Suite** was used to simulate the circuit, test the code, and verify real-time behavior. LCD library functions are used to display temperature in both Celsius and Fahrenheit formats.

Describe how the input is received (sensor, signal, user input):

The system receives inputs from two key sensors: **the LM35 temperature sensor and a gas sensor (e.g., MQ-2 or MQ-135).** The LM35 provides an analog voltage proportional to the ambient temperature, which is read by the microcontroller's ADC channel.

The gas sensor detects harmful gases (like LPG, CO, or smoke) around the battery and also outputs an analog signal based on gas concentration. This signal is connected to another analog pin on the microcontroller. The microcontroller continuously monitors these inputs in real time. No manual user input is required during normal operation, making it fully autonomous and responsive to changing conditions.

Describe how the output is generated (actuator, alert, display):

When the measured temperature exceeds the predefined limit, the microcontroller sends a signal to the **buzzer (BUZ1) and activates the fan (DC motor)** via the transistor Q1. The **LCD display**s both the Celsius and Fahrenheit values, providing clear feedback to the user. The **red LED (D2)** may also be used as an overheat warning indicator.

Does the simulation show correct and useful behavior?

Yes, the Proteus simulation clearly shows that the system works correctly. When the temperature reaches 40°C, the LCD updates, the fan turns ON, the buzzer activates, and the red LED glows. When the temperature falls below the threshold, the system resets the outputs accordingly. We also tested multiple scenarios like gradual temperature increase and sudden gas release, and the response was timely. The simulation confirms that the logic, sensor reading, and output controls are functioning as expected.

5.**TEST- REAL WORLD VALIDATION :**

How will you implement and test the actual hardware/software?

We will first assemble the components on a breadboard , then load the embedded C code into the microcontroller using a programmer. The sensors will be placed near a heated object or simulated battery to test real response. We will gradually heat the area to test the temperature sensor response and use LPG or smoke to trigger the gas sensor. The fan and buzzer should turn on/off based on preset limits.

We will also monitor the LCD to ensure data is readable. Several test cycles will be run to ensure consistent behavior. External interference and noise will be minimized using proper grounding. Logs may be recorded manually or by adding a serial monitor for data tracking.

Who will evaluate your solution (user, teacher, peer)?

Our solution will be evaluated by our project guide/teacher, who will assess technical functionality and safety aspects. Peers in our class may review our simulation and hardware results to give feedback. If possible, we will demonstrate the project to an end user or domain expert like a technician in the EV department or a faculty member with research in battery systems.

What feedback did you receive?

We received suggestions to include more accurate sensors, such as **digital temperature sensors** like DS18B20. Some suggested adding manual override for fan control in case the sensor fails. We were also advised to make the threshold values adjustable through user input or software. Feedback also included improving the alert system using different buzzer tones for different warning levels. Others suggested adding a **cooling-off timer to prevent fan cycling rapidly.**

What improvements will you make?

Based on the feedback, we plan to upgrade to more accurate sensors, use adjustable thresholds via buttons or serial commands, and add fan delay timers. We'll also improve the buzzer alert by using different beeping patterns.

In future versions, we might **integrate wireless communication (e.g., Bluetooth) to send temperature alerts to a smartphone app**. We may also add a **data logging module** for analyzing long-term performance. Additionally, we’ll consider enclosing the setup in a compact case to improve durability and usability.