Project Name:-

Design and simulate a Battery Monitoring and Safety Alert System for a two wheeler Electric Vehicle using Arduino UNO on Tinkercad Circuits.

By

Ankit Kumar

1st March Batch

PROJECT DESCRIPTION

Project Objective :-

The main objective of project is to build a Simplified Battery Management System (BMS) which monitors battery voltage levels, detects over-temperature, trigger alert like buzzer and LED in critical conditions, display system status on LCD.

Important Pre-requisite :-

Before starting the project it is crucial to have a good understanding of basic electronics concepts, circuit design principle components, Arduino Code. Also, knowledge of how to use TinkerCad simulation software is essential, as it will be platform for designing, simulating and analyzing the output of battery temperature, battery voltage.

Require Components

Sl No.	Component Name	Function
1.	Arduino Uno r3	Microcontroller
2.	LCD 16x2	For display
3.	Breadboard	For connection
4.	Temperature Sensor	For measuring temperature
5.	LED	For lighting different status
6.	Buzzer	For audio signalling
7.	Resistor	Resistance
8.	Potentiometer	For varying the
9.	Jumper or Wires	For connecting components

Components and its input value:-

1. Arduino Uno r3

Arduino UNO is a popular open-source microcontroller board based on the ATmega328P. It has 14 digital I/O pins (6 PWM capable), 6 analog inputs, a 16 MHz quartz crystal, USB connection, power jack, and reset button. It's widely used for learning, prototyping, and hobby electronics due to its simplicity, reliability, and compatibility with the Arduino IDE.

Arduino UNO Pin Description:

Digital I/O Pins (D0 – D13):
- Used for digital input or output.
- Pins D3, D5, D6, D9, D10, and D11 support PWM.

Analog Input Pins (A0 – A5):
- Used to read analog voltages (0–5V).
- Can also be used as digital I/O if needed.

Power Pins:
- 5V and 3.3V: Provide regulated power output.
- GND: Ground connections.
- Vin Input voltage if using external power supply.

PWM Pins:
- D3, D5, D6, D9, D10, D11: Used for Pulse Width Modulation (analog-like output).

Serial Communication:
- D0 (RX) and D1 (TX): Used for serial communication via USB or Serial Monitor.

SPI Pins:
- D10 (SS), D11 (MOSI), D12 (MISO), D13 (SCK): Used for SPI communication with sensors and devices.

I2C Pins:
- A4 (SDA) and A5 (SCL): Used for I2C communication with modules like LCD, RTC, etc.

AREF Pin:
- AREF: Provides reference voltage for analog inputs (optional use).

Reset Pin:
- RESET: Resets the microcontroller when triggered.

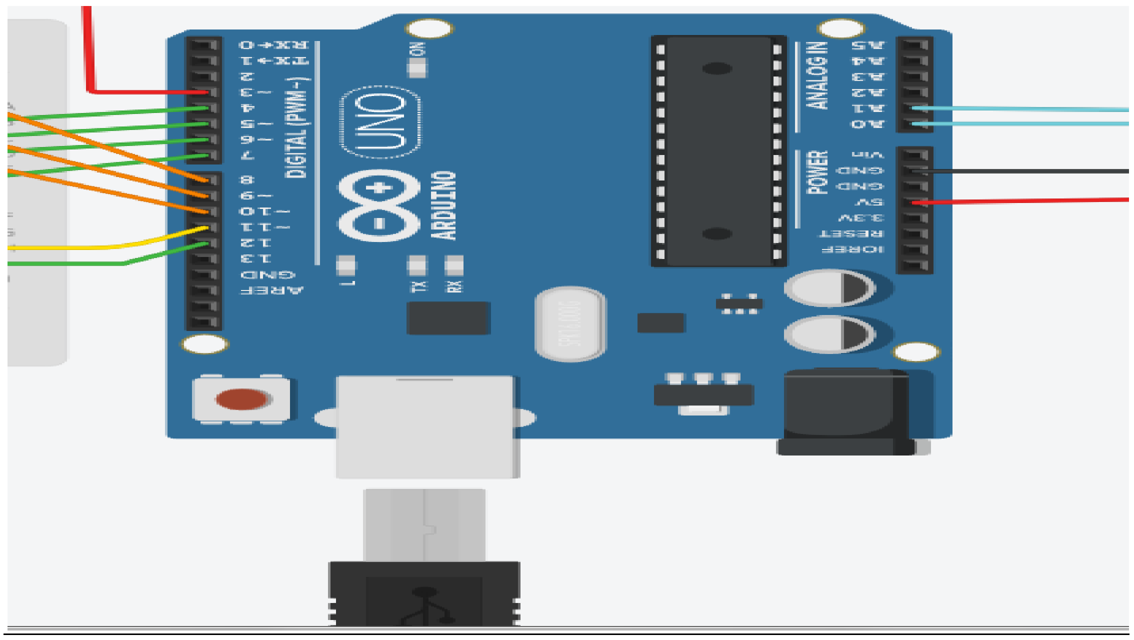


Fig. Arduino UNO

2. LCD 16x2

The LCD used in TinkerCAD is typically a 16x2 alphanumeric display, which means it can show 2 lines with 16 characters each. It is based on the Hitachi HD44780 controller and communicates with the Arduino using either 4-bit or 8-bit parallel mode. In TinkerCAD, it's commonly wired in 4-bit mode to save digital pins. It is used to display text such as sensor readings, status messages, or other data from the Arduino.

16x2 LCD Pin Descriptions:

1. GND → Ground (GND)- Connects to GND of Arduino

2. VCC → +5V - Connects to 5V supply from Arduino

3. VO → Contrast Adjust - Connects to the middle pin of a potentiometer (for adjusting display contrast)

4. RS → Register Select- Selects command register (0) or data register (1); Connect to a digital pin (e.g., D11)

5. RW → Read/Write- Selects Read (1) or Write (0); commonly connected to GND (write-only mode)

6. E → Enable- Triggers LCD to read data on data pins- Connect to a digital pin (e.g., D12)

7. D0–D3 → Data Pins (optional in 8-bit mode) - Not used in 4-bit mode; leave unconnected

8. D4–D7 → Data Pins - Used to send data in 4-bit mode; Common connections: - D4 → D4 on Arduino- D5 → D5 - D6 → D6 - D7 → D7

9. LED+ → Backlight Anode- Connect to 5V through a resistor (e.g., 220Ω)

10. LED– → Backlight Cathode- Connect to GND

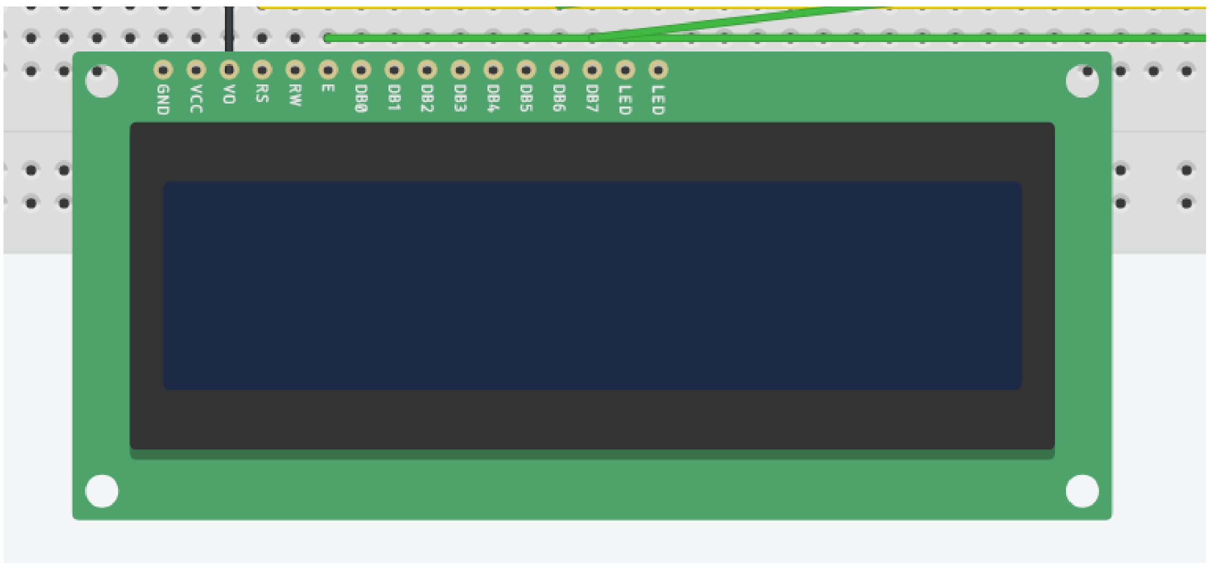


Fig. LCD 16x2

3. BreadBoard

A breadboard is a rectangular plastic board used for building and testing electronic circuits without soldering. It has a grid of interconnected holes where electronic components and wires can be inserted. Breadboards allow quick prototyping and are reusable, making them ideal for beginners and experimenters.

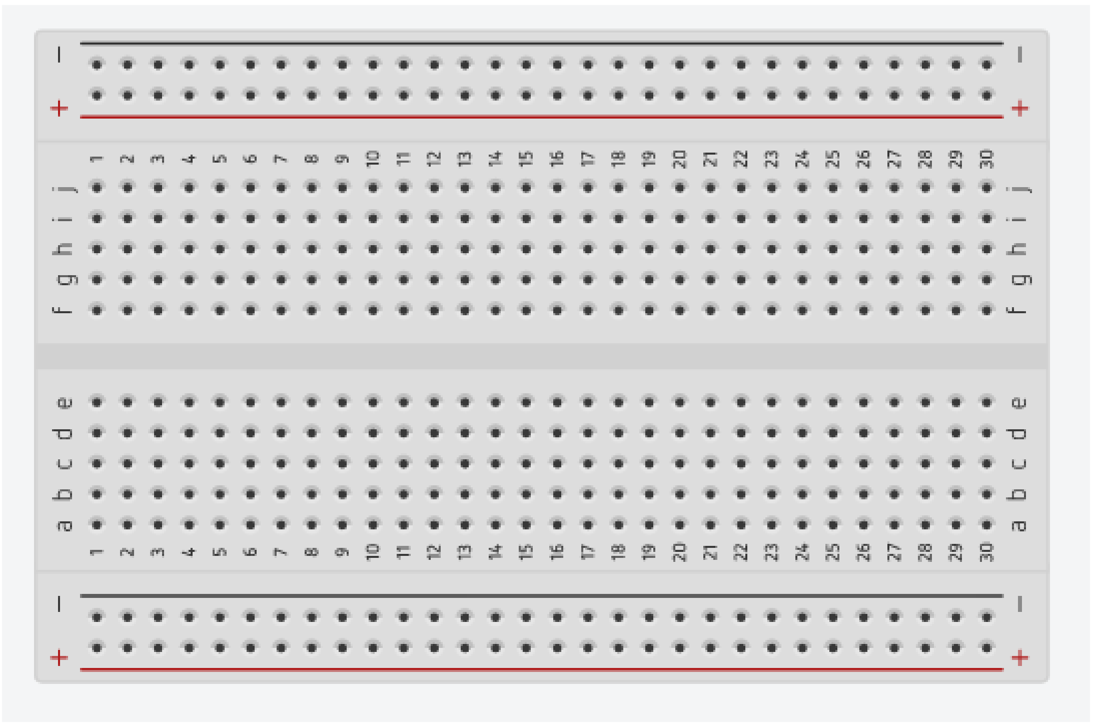


Fig. BreadBoard

4. Temperature Sensor

The LM35 is a commonly used analog temperature sensor that provides a voltage output directly proportional to the temperature in Celsius. It is simple, accurate, and easy to use with Arduino.

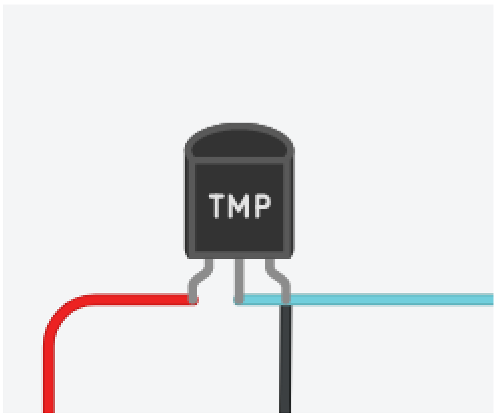


Fig. Tempeature Sensor

5. LED (light emitting diode)

LEDs are commonly used in Arduino projects for indicators, signals, or status displays. To use an LED safely, a resistor (typically 220Ω or 330Ω) is connected in series to limit the current and prevent damage. Here I have used 3 led of colour Red, Green, yellow.

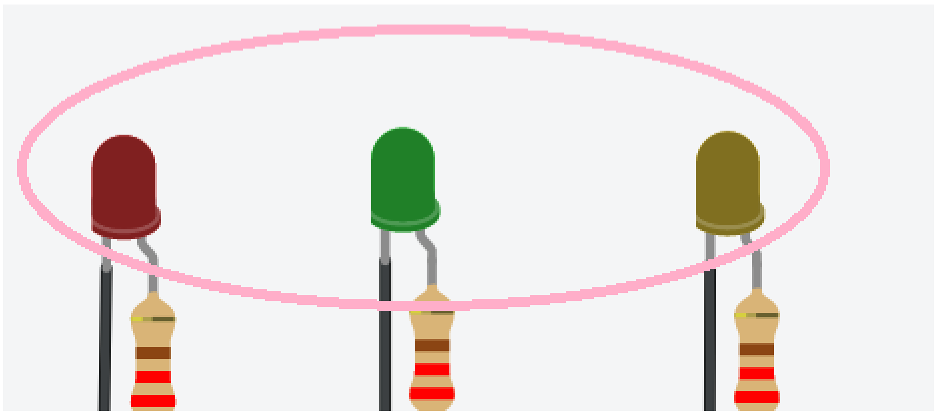


Fig. LED

6. Buzzer

A buzzer is an electronic component that produces sound when powered. It's often used for alarms, notifications, or feedback in Arduino projects.

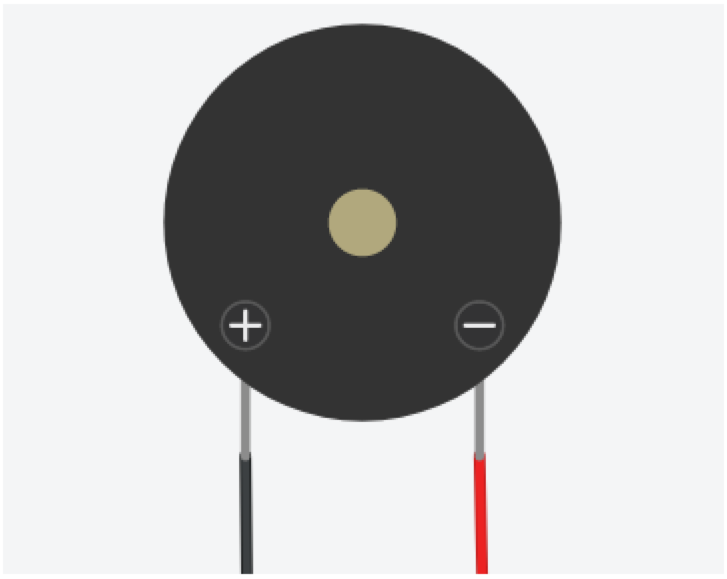
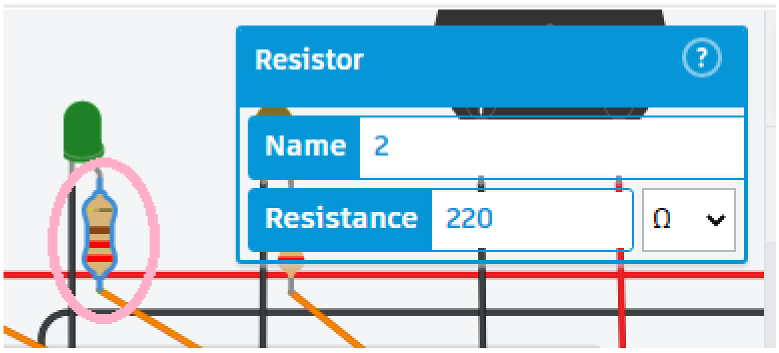


Fig. Buzzer

7. Resistor

A resistor is an electronic component that limits or controls the flow of electric current in a circuit. It's used to protect components like LEDs, buzzers, and sensors from too much current. Here I have used 4 resistor 3 for LED's having value 220 ohms and 1 for LCD having value 1kOhms.



Resistor

Name: 2

Resistance: 220 Ω

Fig. Resistor and its value selection

8. Potentiometer

A potentiometer is a variable resistor that can be used to control voltage. It has three pins and is commonly used to adjust values like brightness, volume, or contrast in circuits. In Arduino projects, potentiometers are often used as analog inputs to control sensors, motors, or display settings. When you rotate the knob, the output voltage changes from 0V to 5V. Here I have used two potentiometer to one to adjust the LCD display brightness and another to adjust the voltage and I have used the scaling factor to change the reading of voltage from 0 to 15V when knob is rotated

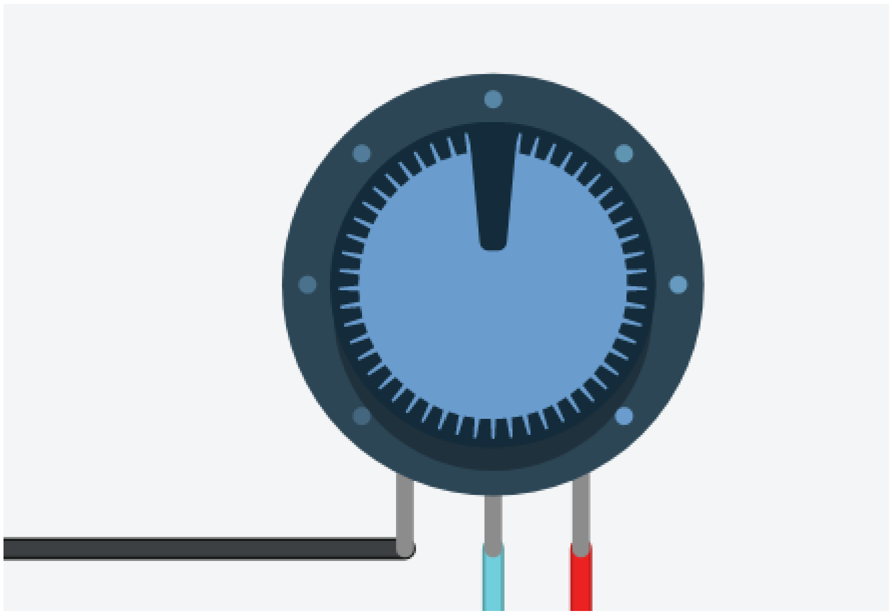


Fig. Potentiometer

Circuit Diagram of BMS (Battery Monitoring System) :-

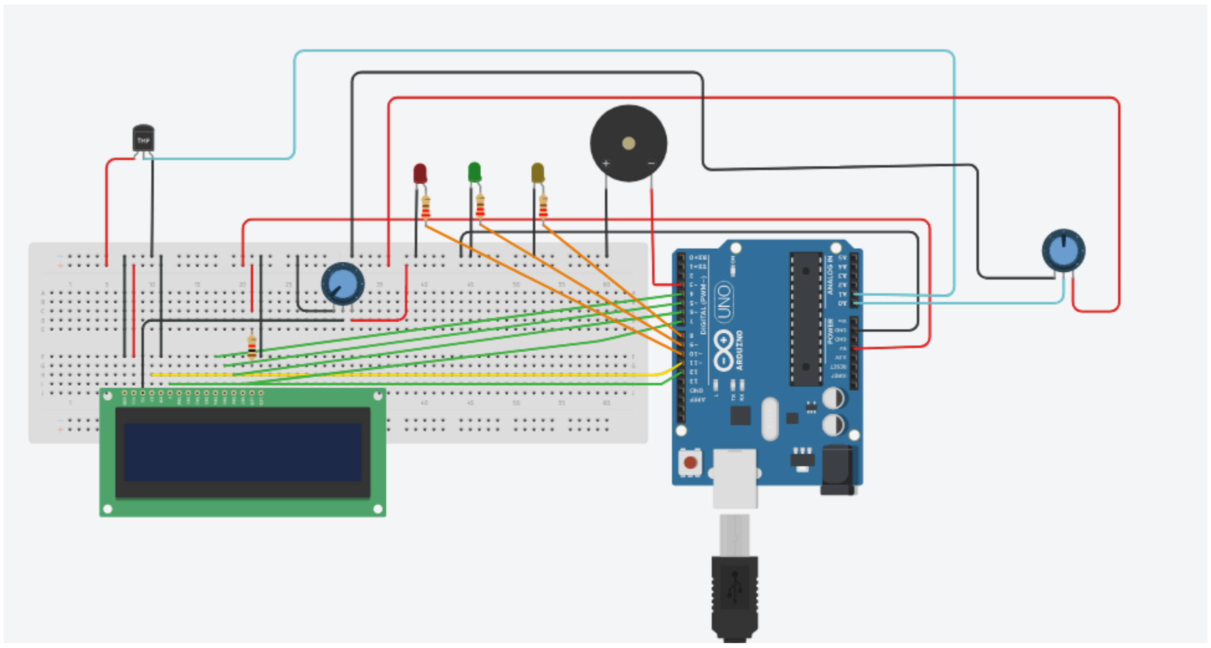


Fig. Circuit Diagram of BMS

Here I have connected the LCD to Arduino Uno board, I have also connected the Led (Red, Green, yellow) to Uno board (providing resistor in their anode to prevent them for getting damaged) to get our desired signal from LED. Buzzer is also connected to Uno board to get sound notification. I have also connected temperature sensor to Uno board to detect temperature change. I have connected one potentiometer to Uno board so change the voltage. One potentiometer is connected LCD to adjust the brightness.

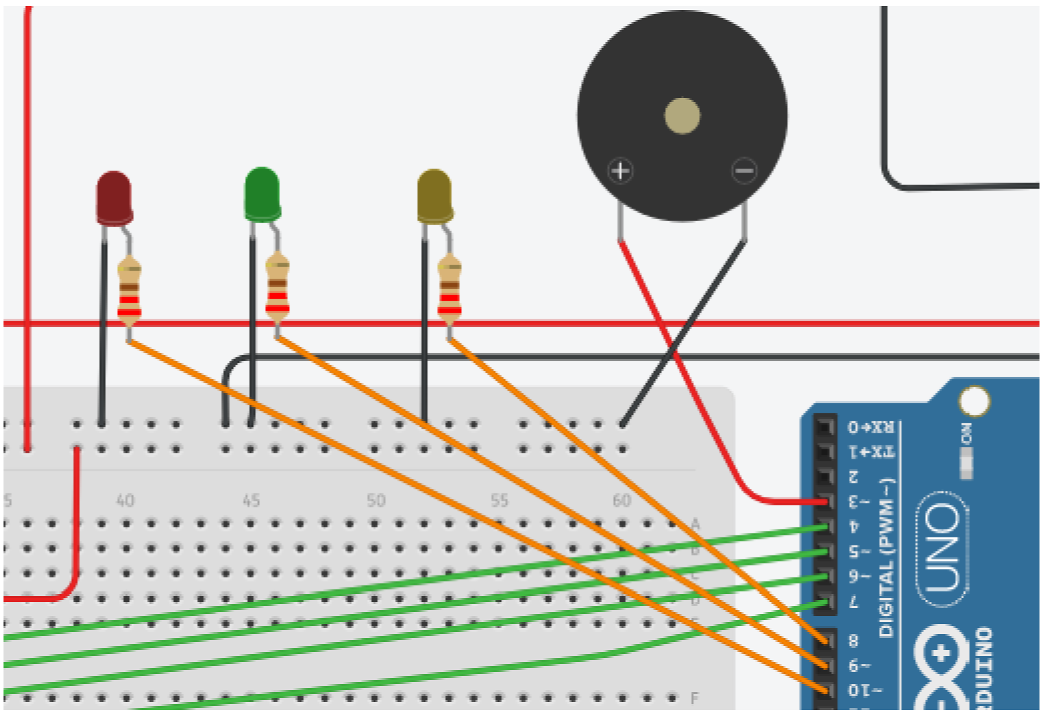


Fig. Circuit Diagram showing connection of LED and Buzzer

Here I have connected the cathode of LED & Buzzer to ground of Uno board using breadboard .The LED Red, Green, Yellow with Resistor having resistance of 220 ohm is connected to anode and with help of jumper wire these are connected to Digital PIN 10,9,8 respectively. Buzzer positive is connected to Digital PIN 3.

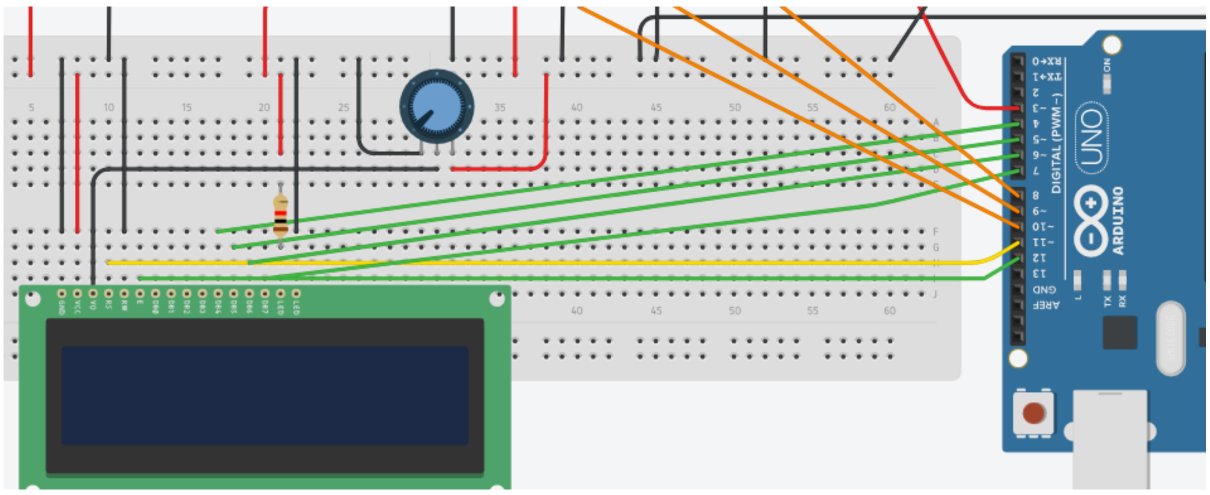


Fig. Circuit Diagram showing connection of LCD

Here I have connected the LCD to Uno board, I have connected the GND to GND, VCC to 5V power pin, RW to GND, V0 to slider of potentiometer, using breadboard to Uno board and RS , E, DB4, DB5, DB6, DB7 directly to Digital pin 11, 12, 4, 5, 6,7 respectively and last 2 LED, LED are connected to 5V and GND respectively using breadboard to Uno Board. Here Potentiometer legs are connected to GND and 5V to adjust the brightness of LCD.

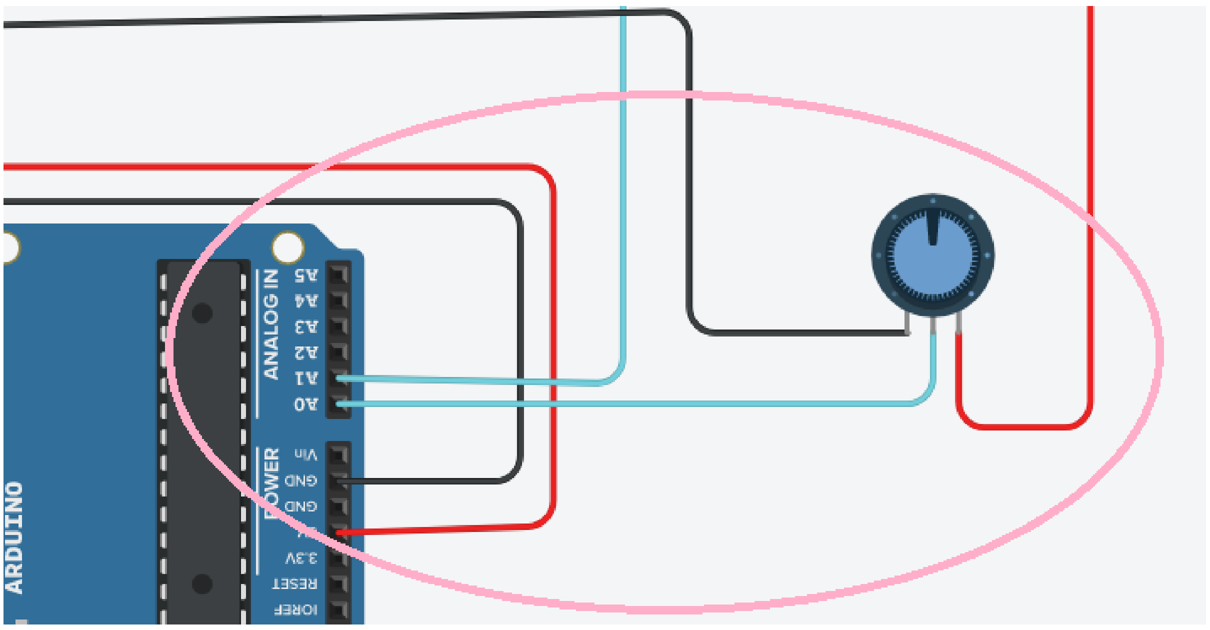


Fig. Circuit Diagram showing connection of temperature sensor & Potentiometer

Here I have connected the analog pin A0 to potentiometer to adjust the voltage and A1 to temperature sensor to adjust the temperature and also connected their rest legs to positive and negative to supply the current.

Embedded Code

```
#include <LiquidCrystal.h>

// LCD pin configuration

const int rs = 11, en = 12, d4 = 4, d5 = 5, d6 = 6, d7 = 7;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// Pin definitions

const int voltPin = A0;

const int tempPin = A1;

const int redLED = 10;

const int greenLED = 9;

const int yellowLED = 8;

const int buzzer = 3;

void setup() {

  // Set pin modes

  pinMode(redLED, OUTPUT);

  pinMode(greenLED, OUTPUT);

  pinMode(yellowLED, OUTPUT);

  pinMode(buzzer, OUTPUT);

  lcd.begin(16, 2);

  lcd.print("System Starting...");

  delay(1000);

  lcd.clear();

}

void loop() {

  // Read and convert battery voltage

  int rawVolt = analogRead(voltPin);

  float voltage = (rawVolt * 5.0 / 1023.0) * 3.0; // Scaled to 15V
  // Read and convert temperature
  int rawTemp = analogRead(tempPin);
  float temperature = (rawTemp * 5.0 / 1023.0) * 100.0; // LM35 = 10mV/°C
// Clear LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Volt:");
  lcd.print(voltage, 1);
  lcd.print("V");
  lcd.setCursor(0, 1);
  lcd.print("Temp:");
  lcd.print(temperature, 1);
  lcd.print("C");
  // Default all outputs OFF
  digitalWrite(redLED, LOW);
  digitalWrite(greenLED, LOW);
  digitalWrite(yellowLED, LOW);
  digitalWrite(buzzer, LOW);
  delay(1000); // Small delay before status check
  // Battery status logic
  if (voltage < 11.0) {
    lcd.clear();
    lcd.print("Low Batt!");
    digitalWrite(redLED, HIGH);
    digitalWrite(buzzer, HIGH);
  }
  else if (voltage >= 11.0 && voltage <= 12.6) {
    lcd.clear();
    lcd.print("Battery OK");
    digitalWrite(greenLED, HIGH);
    digitalWrite(buzzer, LOW);
  }
  else if (voltage > 13.0) {
    lcd.clear();
    lcd.print("Not Needed");
    digitalWrite(redLED, HIGH);
    digitalWrite(buzzer, HIGH);
  }
  delay(2000); // Allow time to read LCD
  // Temperature check
  if (temperature > 45.0) {
    lcd.clear();
    lcd.print("Overheat!");
    digitalWrite(yellowLED, HIGH);
    digitalWrite(buzzer, HIGH);
    delay(2000); // Wait before next loop
  }
}
```

For Battery Voltage logic

```
// Battery status logic
if (voltage < 11.0) {
  lcd.clear();
  lcd.print("Low Batt!");
  digitalWrite(redLED, HIGH);
  digitalWrite(buzzer, HIGH);
}
else if (voltage >= 11.0 && voltage <= 12.6) {
  lcd.clear();
  lcd.print("Battery OK");
  digitalWrite(greenLED, HIGH);
  digitalWrite(buzzer, LOW);
}
else if (voltage > 13.0) {
  lcd.clear();
  lcd.print("Not Needed");
  digitalWrite(redLED, HIGH);
  digitalWrite(buzzer, HIGH);
}
```

Here I have used decision making logic to check the voltage of battery and voltage is adjusted by potentiometer.

If voltage is below 11.0V then LCD will display "Low Batt! " , red LED will glow & buzzer will also produce sound.

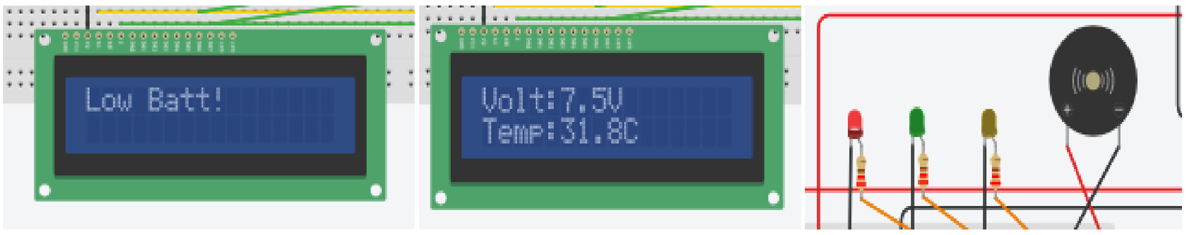


Fig. At 7.5V LCD showing Low Batt! & Red LED and Buzzer are also On

If voltage is in between 11.0V &12.6V including both extreme then LCD will display "Battery OK ", green LED will glow & buzzer will not produce sound.

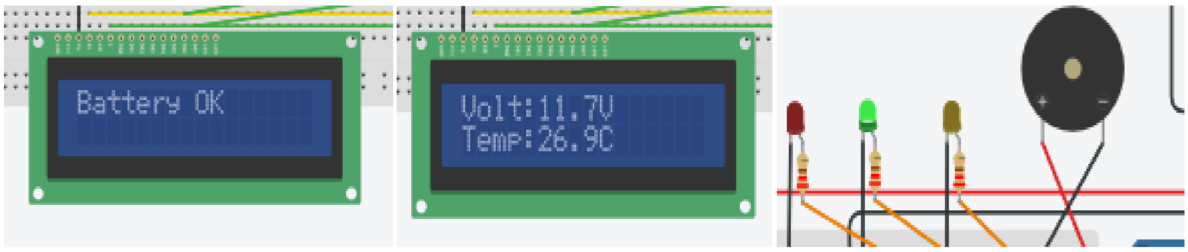


Fig. At 11.7V LCD showing Battery OK & Green LED On and Buzzer Off

If voltage is above 13.0 then LCD will display “Not Needed ” , red LED will glow & buzzer will also produce sound.

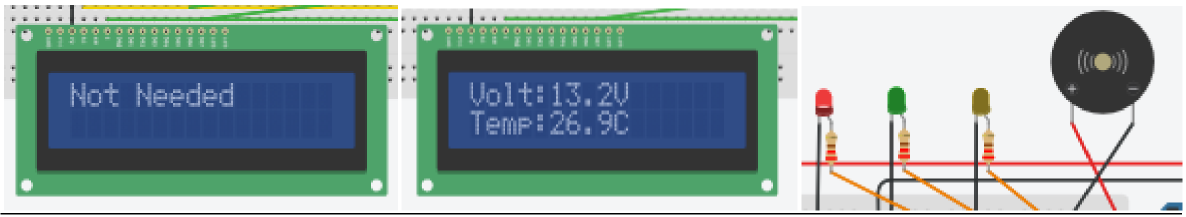
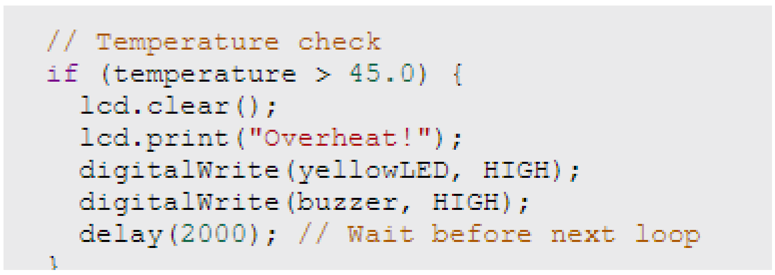


Fig. At 13.2V LCD showing Not Needed & Red LED and Buzzer are also On

For Temperature logic



```
// Temperature check
if (temperature > 45.0) {
  lcd.clear();
  lcd.print("Overheat!");
  digitalWrite(yellowLED, HIGH);
  digitalWrite(buzzer, HIGH);
  delay(2000); // Wait before next loop
}
```

Here I have used decision making logic to check the temperature of battery and temperature is adjusted by temperature sensor.

If temperature is above 45.0 LCD will display “Overheat” , yellow LED will glow and buzzer will also produce sound.

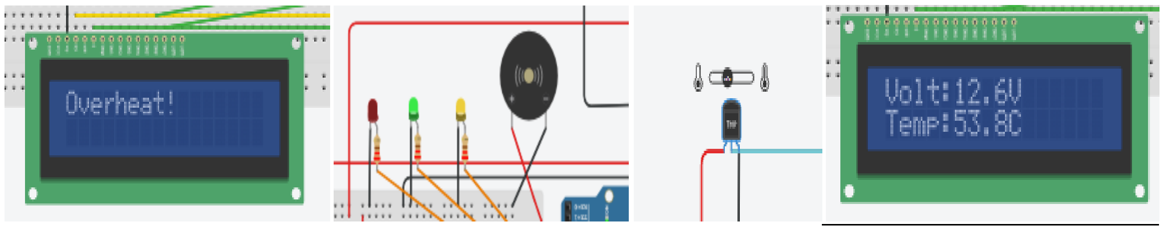


Fig. At Temp 53.8C LCD showing Overheat & yellow LED and Buzzer are also On

Figure below are showing 10.5 volt, Temp 33.2C and low battery with Red LED and buzzer ON

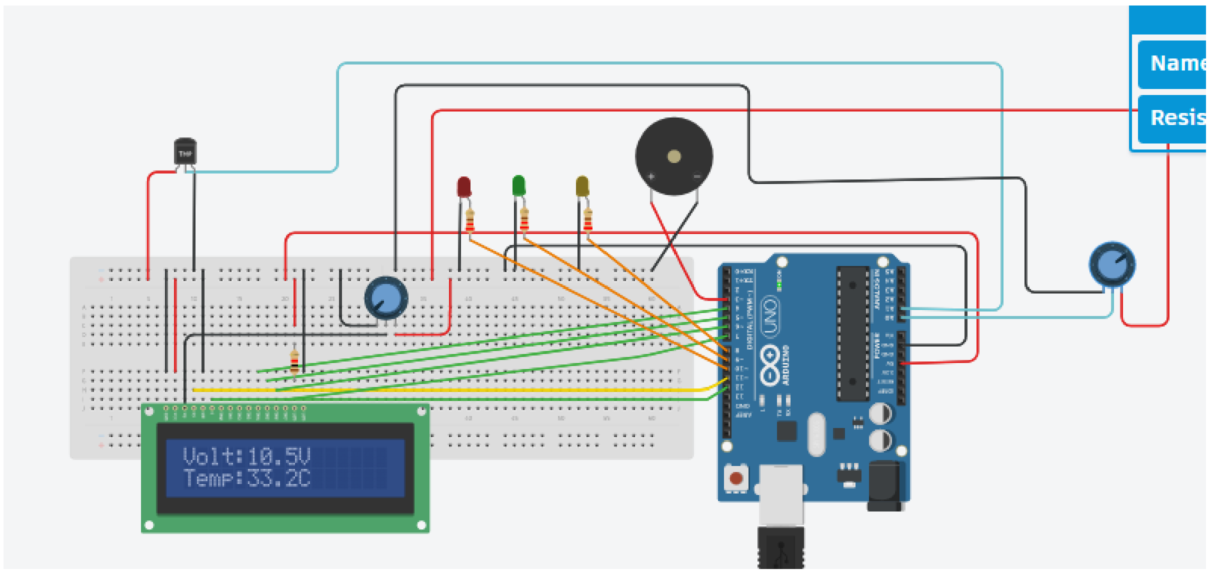


Fig. Test Result for simulation at 10.5V and 33.2C

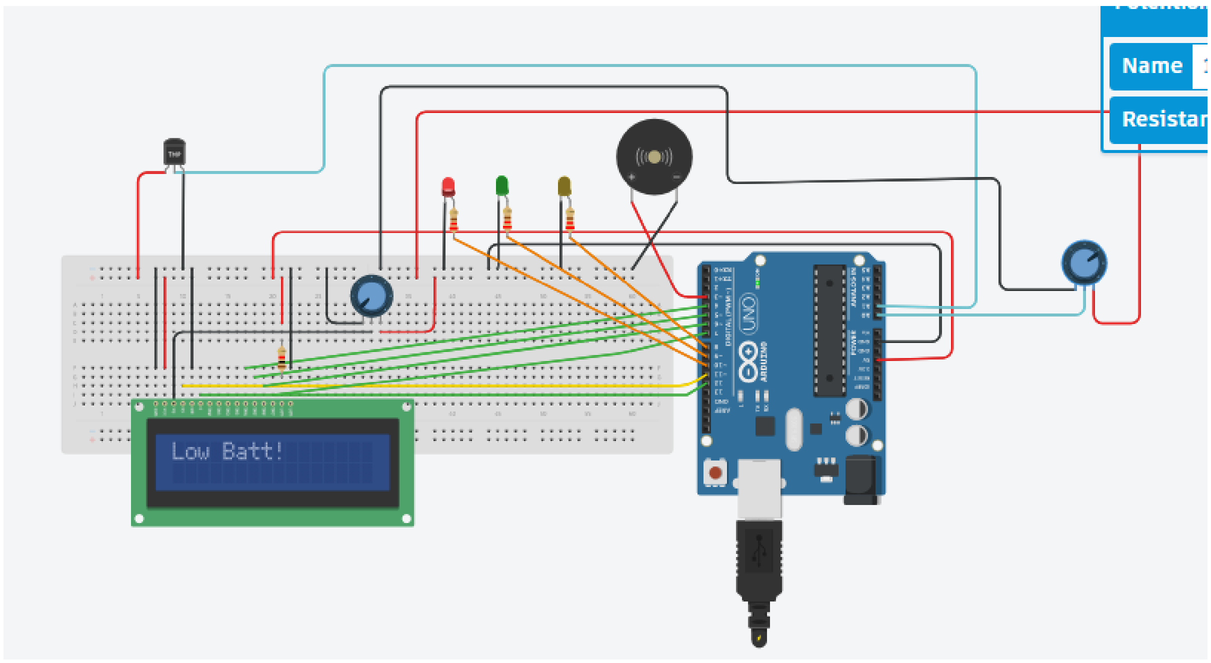


Fig. Test Result for simulation at 10.5V and 33.2C

Here when voltage is 10.5V and temperature is 33.2 C LCD shows the reading and after delay LCD shows Low Batt! and Red LED and Buzzer ON.

Figure below are showing 12 volt, Temp 33.2C and low battery with Red LED and buzzer ON

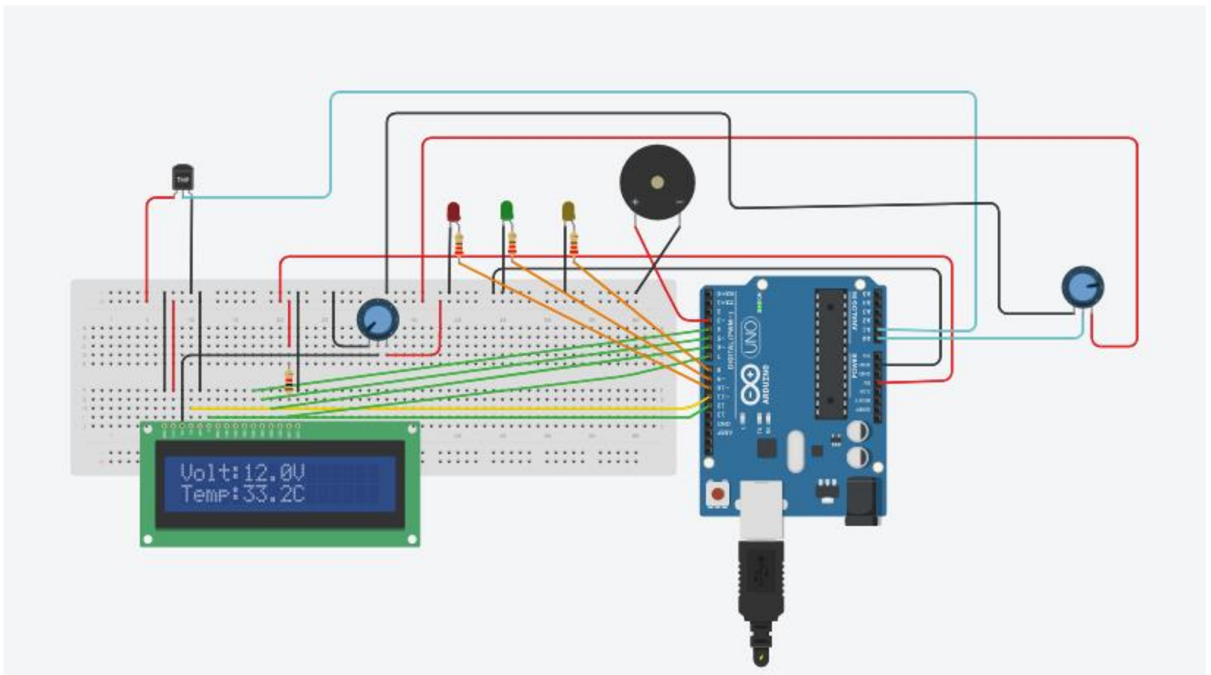


Fig. Test Result for simulation at 12V and 33.2C

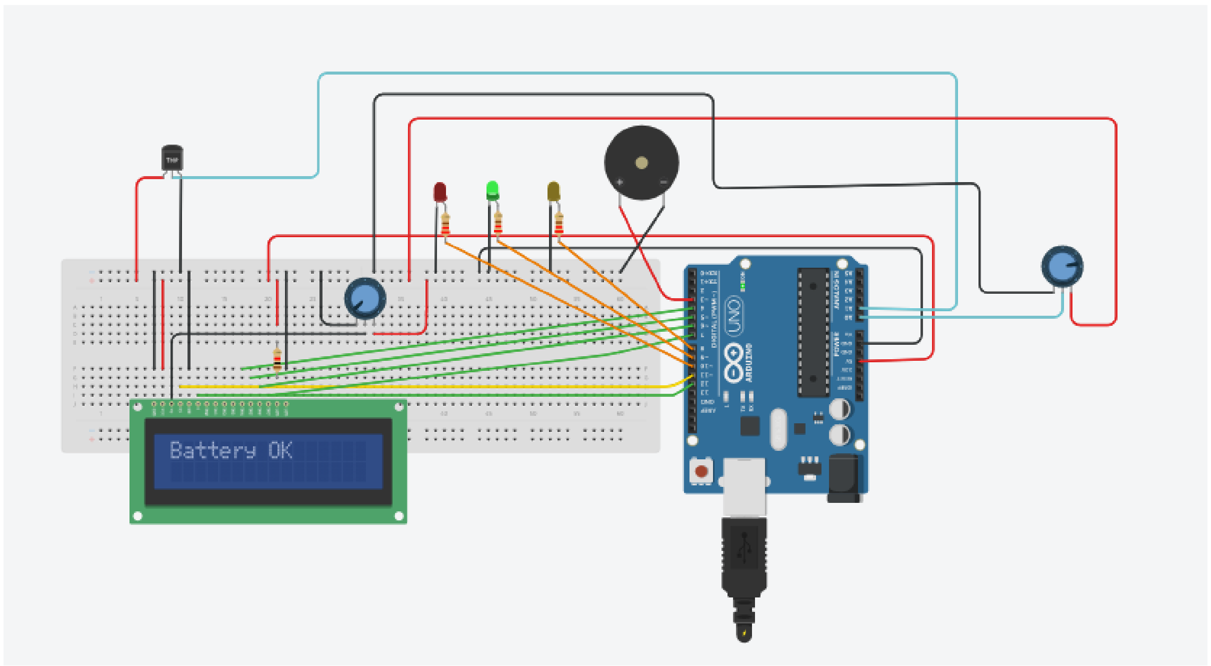


Fig. Test Result for simulation at 12V and 33.2C

Here when voltage is 12V and temperature is 33.2 C LCD shows the reading and after delay LCD shows Battery OK and Green LED ON.

Figure below are showing 14.4 volt, Temp 33.2C and low battery with Red LED and buzzer ON

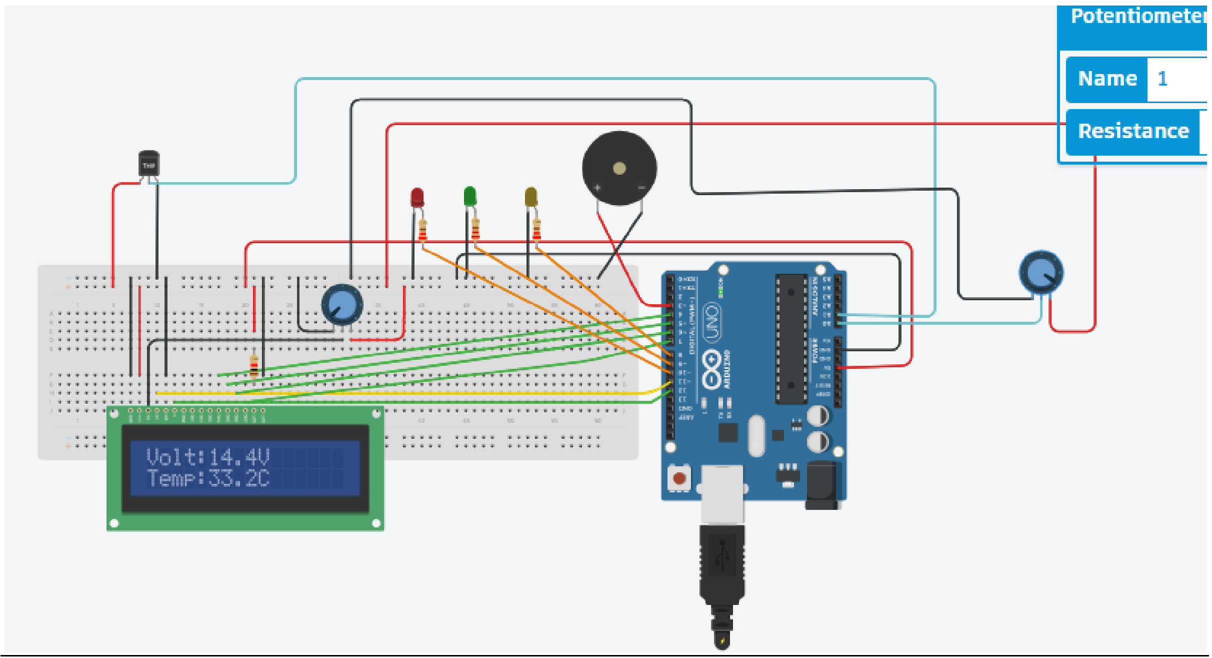


Fig. Test Result for simulation at 14.4V and 33.2C

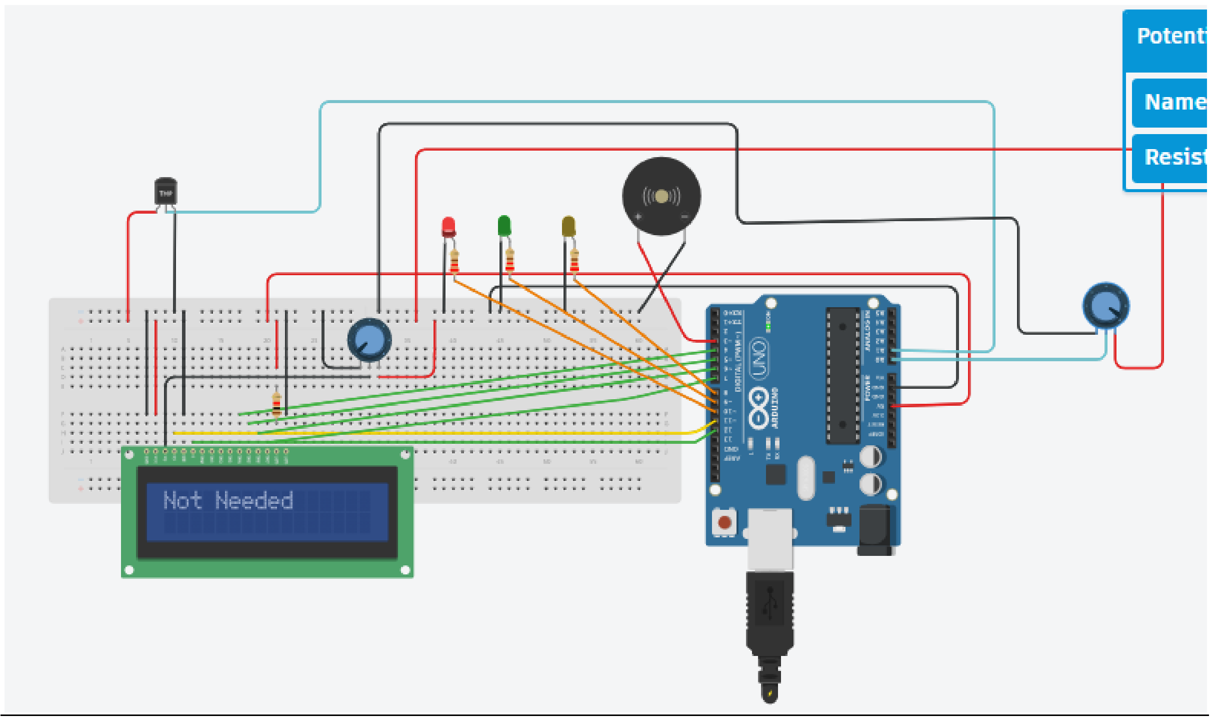


Fig. Test Result for simulation at 14.4V and 33.2C

Here when voltage is 14.4V and temperature is 33.2 C LCD shows the reading and after delay LCD shows Not Needed and Red LED & Buzzer ON.

Figure below are showing 12.6 volt, Temp 59.1C and with Green & Yelllow LED ON and buzzer ON.

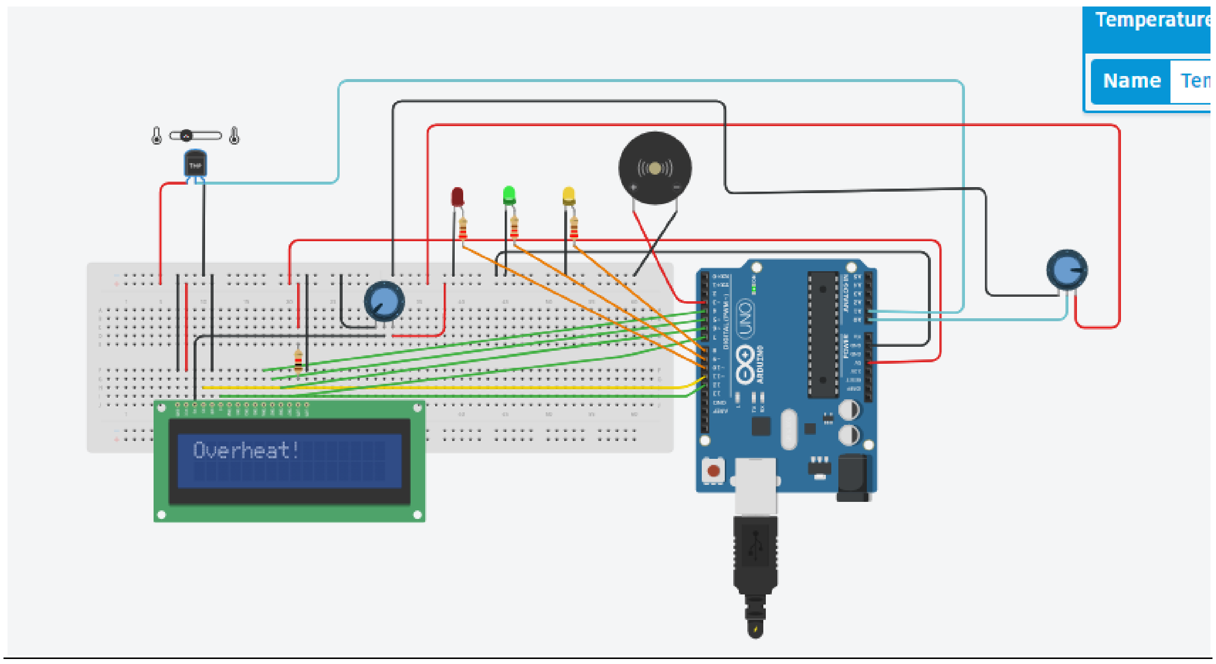


Fig. Test Result for simulation at 12.6V and 59.1C

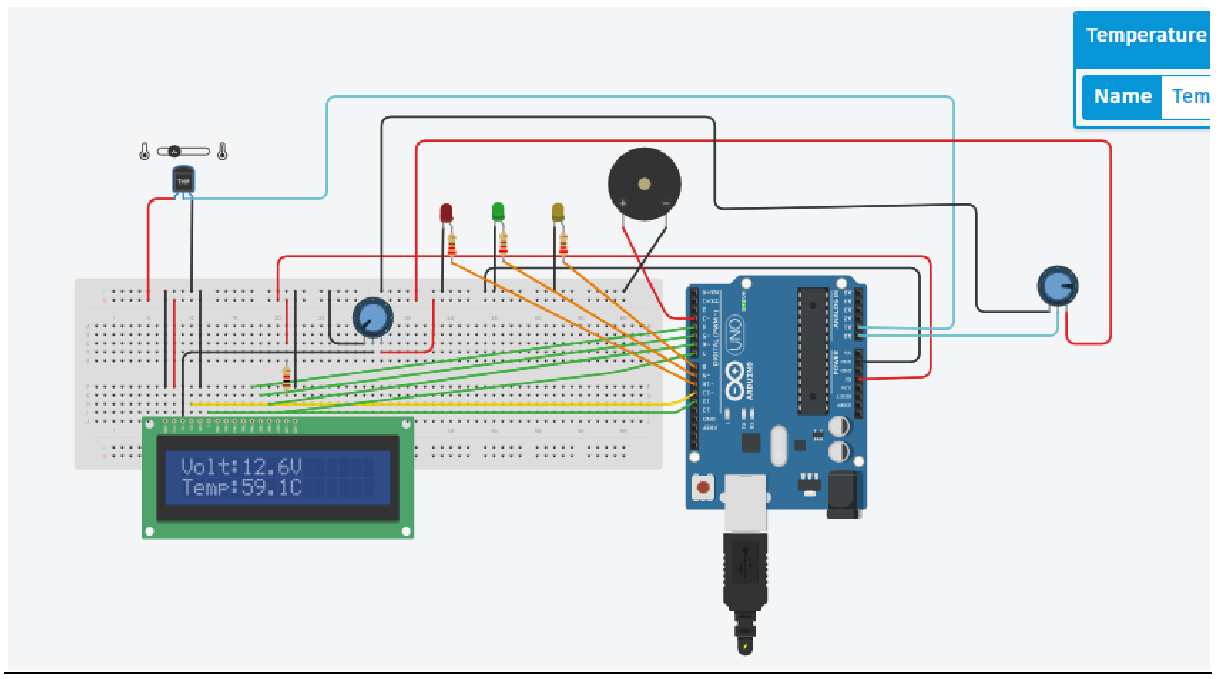


Fig. Test Result for simulation at 12.6V and 59.1C

Here when voltage is 12.6V and temperature is 59.1C LCD shows the reading and after delay LCD shows Overheat! and Yellow, Green LED & Buzzer ON (Green LED showing Battery voltage is between 11 & 12.6 V , Yellow LED showing Battery Temperature is above 45C)

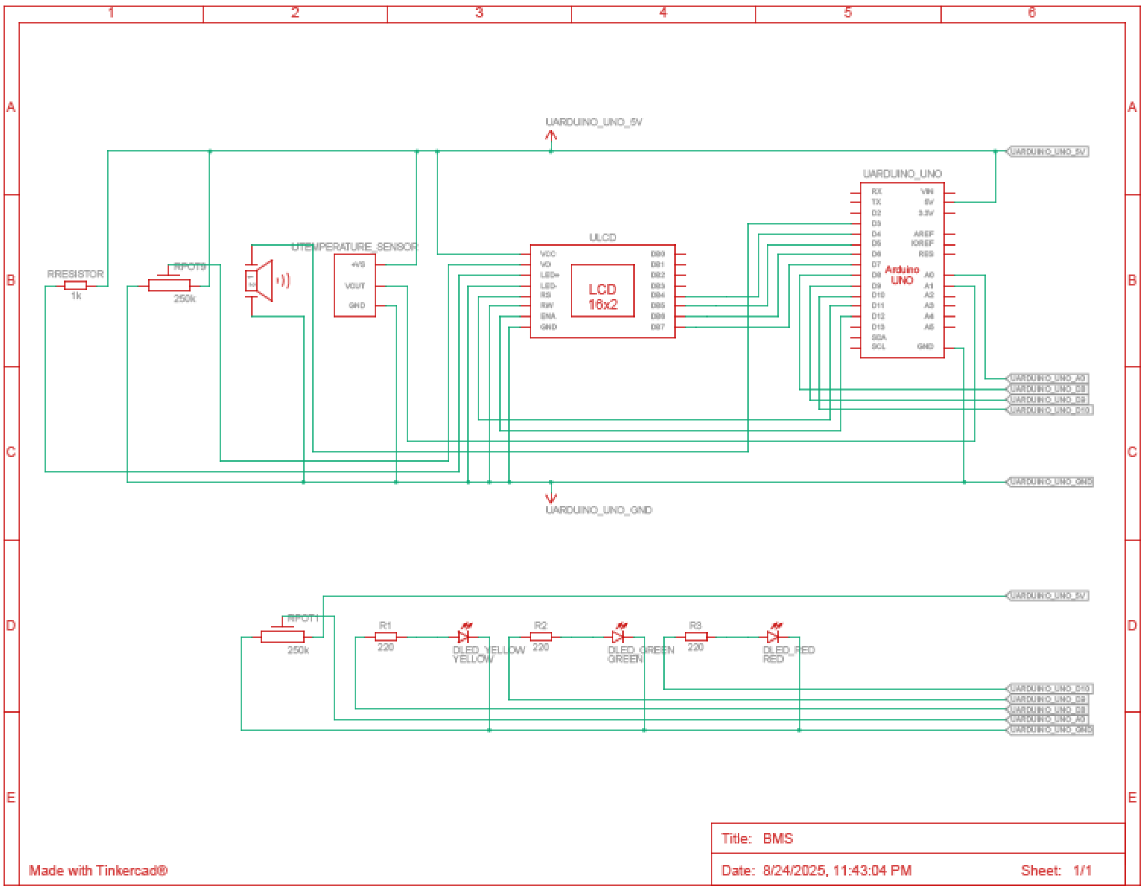


Fig. Schematic View of Circuit

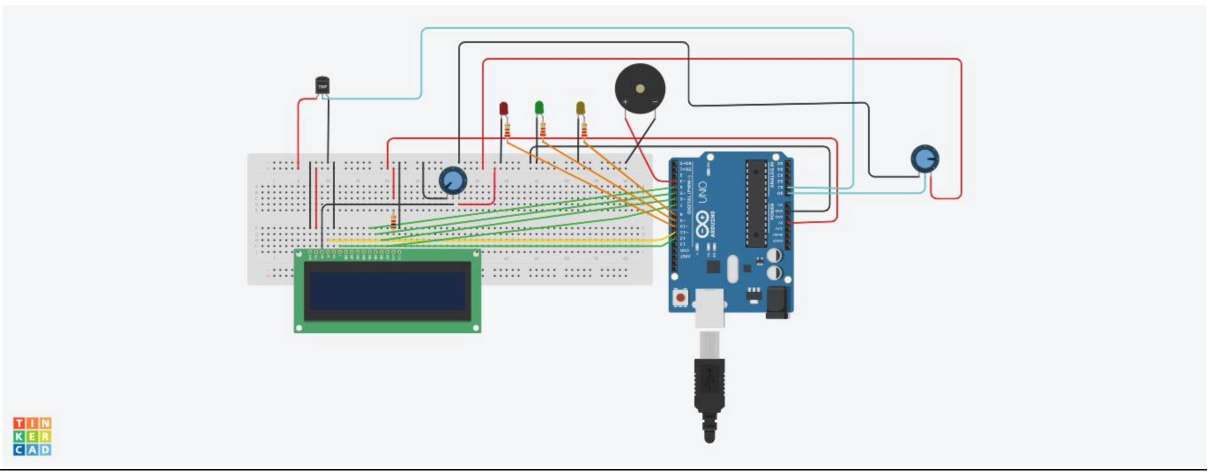


Fig. Circuit Diagram

<u>Code Explanation</u>

1. #include <LiquidCrystal.h> – Includes the LiquidCrystal library for using an LCD.

2. const int rs = 11, en = 12, d4 = 4, d5 = 5, d6 = 6, d7 = 7;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7); – Sets the LCD control and data pins, then initializes the LCD object.

3. const int voltPin = A0; const int tempPin = A1; const int redLED = 10…………… – Pin assignments for battery voltage, temperature sensor, LEDs, and buzzer.

4. void setup() function: - Sets the pin modes for the LEDs and buzzer as OUTPUT;

5. lcd.begin(16, 2); - Initializes the LCD to 16 columns and 2 rows;

6. lcd.print("System Starting..."); - Displays "System Starting..."

7. delay(1000); -displays "System Starting..." for 1 second,

8 lcd.clear(); - then clears the display.

9. void loop() function -runs continuously.

10. int rawVolt -analogRead(voltPin); -Reads the voltage on a specified analog input pin, converts it to a numerical value, and stores that value in the integer variable rawVolt.

11. float voltage - (rawVolt * 5.0 / 1023.0) * 3.0; - Scale the rawVolt to 15v and stores that value in the float variable voltage.

12. int rawTemp - analogRead(tempPin); Reads the temperature on a specified analog input pin, converts it to a numerical value, and stores that value in the integer variable rawTemp.

13. float temperature - (rawTemp * 5.0 / 1023.0) * 100.0; Converts analog value to Celsius assuming LM35 sensor.

14. lcd.clear(); -Clear the entire LCD ,deleting previoud text.

15. lcd.setCursor(0, 0); -Set the cursor position to column 0,row 0 of LCD.

16. lcd.print("Volt:"); - Print Volt: at column 0,row 0 of LCD.

17. lcd.print(voltage, 1); -Print the value of voltage formatted to 1 decimal place.

18. lcd.print("V"); -Print the letter V to the LCD

19. lcd.setCursor(0, 1); - Set the cursor position to column 0,row 1 of LCD.

20. lcd.print("Temp:"); - Print Temp: at column 0,row 1 of LCD

21. lcd.print(temperature, 1); - Print the value of temperature formatted to 1 decimal place

22. lcd.print("C"); - Print the letter C to the LCD

23. digitalWrite(redLED, LOW); digitalWrite(greenLED, LOW);digitalWrite(yellowLED, LOW); digitalWrite(buzzer, LOW); - Turns off all LEDs and buzzer before running condition checks.

24. Battery voltage logic:

```
if (voltage < 11.0) {
lcd.clear();
lcd.print("Low Batt!");
digitalWrite(redLED, HIGH);
digitalWrite(buzzer, HIGH);
}
else if (voltage >= 11.0 && voltage <= 12.6) {
lcd.clear();
lcd.print("Battery OK");
digitalWrite(greenLED, HIGH);
digitalWrite(buzzer, LOW)
}
else if (voltage > 13.0) {
lcd.clear();
lcd.print("Not Needed");
digitalWrite(redLED, HIGH);
digitalWrite(buzzer, HIGH);
```

Here we have use else if logic to check the voltage if voltage is below 11v print “Low batt!” , turn red LED On, Buzzer On ;else if voltage is in between 11.0v & 12.6v including both print “Battery OK” , turn green LED On, Buzzer Off ;else if voltage is above 13v print “Not Needed” , turn red LED On, Buzzer On.

25. delay(2000); -Waits 2 seconds before checking temperature.

26. if (temperature > 45.0) {

```
lcd.clear();
lcd.print("Overheat!");
digitalWrite(yellowLED, HIGH);
digitalWrite(buzzer, HIGH);
```

- When temperature is above 45, Displays "Overheat!", turns on yellow LED and buzzer On.

27. delay(2000); - Waits another 2 seconds.
28. Repeats the loop.

Tinkercad simulation Link-

https://www.tinkercad.com/things/bcmtNJvwkaV-bms-/editel?returnTo=https%3A%2F%2Fwww.tinkercad.com%2Fdashboard&sharecode=xUimBF7JxwRinpP-KWiqb3IpUa4U_aizn2fti7bn3LI

Conclusion-

This embedded system code and circuit successfully monitors battery voltage and temperature using analog sensors, displays the readings on an LCD, and uses visual (LEDs) and audible (buzzer) alerts to indicate system status. The logic distinguishes between low, normal, and overcharged battery conditions, and warns of overheating. It offers a simple, reliable interface for real-time diagnostics.

Future enhancements-

Here we can integrate fan for cooling when battery temperature is reached beyond the threshold temperature limit and also we can add disconnect the battery when overheated or overcharged.