EMBEDDED SYSTEMS TASK-1

Submitted in panel for the presentation of Internship Training

Submitted in partial fulfillment of the requirements for the completing of the first task in Internship Training

by

P.Shiva Shankara Vara Prasad

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

From

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

DECEMBER(2025)

ABSTRACT

Smart temperature monitoring is an advanced system designed to automatically measure, record, and evaluate temperature levels in real time using sensors, microcontrollers, and communication technologies. Traditional manual temperature checking methods are slow, prone to errors, and lack continuous tracking. The smart monitoring approach overcomes these limitations by providing accurate and instant temperature observations that support better control, decision-making, and safety management.

The system typically integrates temperature sensors with a processing unit such as Arduino, PIC, or IoT-based controllers to detect environmental or device-level temperature changes. Once the temperature data is collected, the controller processes it and displays it on LCD or mobile applications for user visibility. Wireless communication modules like Wi-Fi, Bluetooth, or GSM enable remote monitoring so that users can track temperature conditions without being physically present at the location.

In addition, the system incorporates alert mechanisms to notify users when temperature readings cross predefined safety limits. These alerts may be sent through SMS, alarms, mobile notifications, or email depending on the application. This feature is particularly useful in industries like food storage, medical refrigeration, server rooms, agriculture, and manufacturing, where temperature fluctuations can cause damages or reduce product quality.

Furthermore, smart temperature monitoring systems can store historical data, which supports analytics and performance evaluation. By analyzing stored temperature patterns, organizations can predict failures, schedule maintenance, and improve energy efficiency. This makes the system not just a monitoring tool but an intelligent solution for long-term environmental control and automation.

Overall, smart temperature monitoring enhances reliability, safety, and responsiveness across various fields. It provides continuous observation, improves operational efficiency, and reduces risk by ensuring temperature remains within required limits. Its real-time capabilities, combined with automation and connectivity, make it a practical and modern solution for both industrial and domestic applications.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 EMBEDDED SYSTEMS

An embedded system is a specially designed computing system that performs dedicated functions within a larger electrical or mechanical system. Unlike general-purpose computers (like laptops or smartphones that handle multiple tasks), embedded systems are built to perform a specific task efficiently, reliably, and with minimal human intervention. They combine hardware and software to control devices and processes in real time.

The core of an embedded system usually includes a microcontroller or microprocessor that acts as the brain of the device. Microcontrollers like Arduino, PIC, AVR, ARM Cortex, and 8051 are commonly used because they have built-in processing and memory resources suitable for embedded applications. The software running on an embedded system is optimized to accomplish a single job, ensuring fast performance and low power consumption.

Embedded systems are present in everyday life and are used in industries, homes, automobiles, healthcare, agriculture, aviation, and consumer electronics. Examples include washing machines, smart TVs, printers, ATM machines, traffic control systems, medical devices, and automotive control units. This wide usage shows how embedded technology has become an essential part of modern electronics.

One major characteristic of embedded systems is that they work in real-time. Real-time systems must react instantly to input signals and generate output responses without delay. For example, an airbag system in a car must deploy within milliseconds during a collision. Any delay or incorrect response could lead to serious consequences. Hence, reliability and timing accuracy are crucial features.

Embedded hardware typically includes components like sensors, actuators, timers, memory units, communication interfaces, displays, and power supply units. Sensors collect input signals from the environment, while actuators perform physical actions based on the processed output. The embedded controller reads sensor data, processes it according to software logic, and controls actuators to complete the task.

Embedded software, also called firmware, is written using languages such as Embedded C, C++, Python (for some boards), and assembly language. The software directly interacts with the hardware and is optimized for performance and memory efficiency. Unlike traditional software applications.

Embedded systems can be classified into different types based on performance and functionality, such as small-scale, medium-scale, and large-scale systems. Small-scale systems use basic 8-bit or 16-bit controllers, while large-scale systems use advanced 32-bit processors or SoCs (System on Chip) that support complex tasks like multimedia processing, artificial intelligence, and IoT connectivity.

With the evolution of technology, embedded systems have moved from simple standalone controllers to connected and intelligent devices. Modern embedded systems often use communication protocols like UART, SPI, I2C, Wi-Fi, Bluetooth, CAN, and Ethernet for data transfer. This connectivity allows them to become part of Internet of Things (IoT) networks, enabling remote monitoring and automation.

The design and development of embedded systems require knowledge of electronics, programming, communication protocols, debugging tools, and board-level interfacing. Engineers must test both hardware and software to ensure proper functioning. Tools like Keil, MPLAB, Arduino IDE, Proteus, and STM32CubeIDE are commonly used for coding, simulation, and hardware testing.

In conclusion, embedded systems are at the heart of modern automation and smart technology. They provide efficient control, reduce human effort, improve accuracy, and enhance safety across various applications. As industries move toward smart manufacturing, electric vehicles, medical automation, and IoT, the importance of embedded systems continues to grow, making it a vital field for students and engineers to learn and build a successful career in.

CHAPTER 2

COMPONENTS

2.1 ARDUINO UNO

The Arduino Uno is a widely used microcontroller board based on the ATmega328P processor. It provides a simple platform for building electronic prototypes and automation systems. It comes with 14 digital input/output pins, 6 analog inputs, and dedicated power pins, allowing users to connect sensors, modules, and external circuits easily.

Fig:2.1:Arduino uno

Arduino Uno is programmed using the Arduino IDE software, which supports C/C++-based coding. The board is beginner-friendly and includes a USB interface, making code uploading fast and convenient. It has a built-in bootloader that removes the need for a dedicated programmer, which is why it is commonly used in academic projects and industrial prototyping.

Because of its flexibility, the Arduino Uno is used in robotics, home automation, medical devices, smart monitoring systems, IoT devices, and sensor-based projects. It supports various communication protocols like UART, SPI, and I2C, making it suitable for advanced interfacing. Its low cost, reliability, and vast community support make it a top choice in embedded electronics.

2.2 TEMPERATURE SENSOR

A temperature sensor is an electronic component that measures heat levels and converts temperature readings into electrical signals. Common types include LM35, DHT11, and DS18B20, each offering different accuracy ranges and output formats. These sensors help monitor environmental, industrial, and equipment temperature in real time.

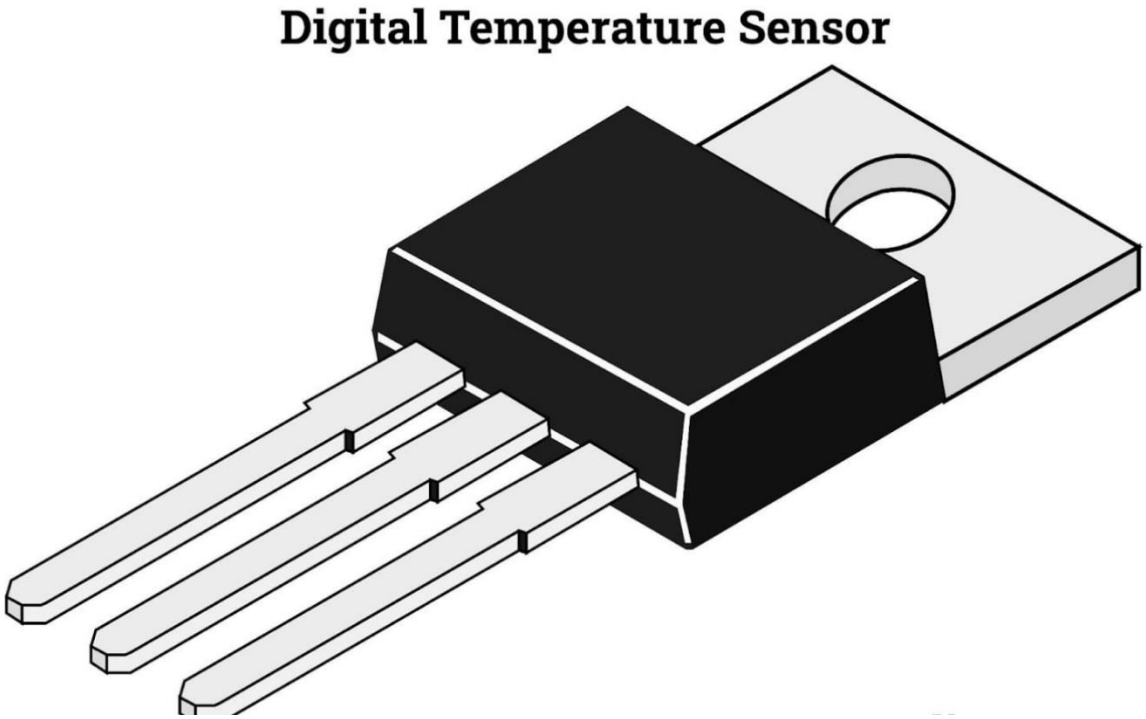

Fig:2.2:Temperature sensor

Temperature sensors can output analog or digital signals depending on their type. Sensors like LM35 provide analog voltage proportional to temperature, while DHT11 transmits digital signals with coded temperature and humidity data. When connected to a microcontroller, these readings are processed and used for monitoring, control, or automation.

Temperature sensors are used in refrigerators, air conditioners, industrial machines, medical devices, agriculture greenhouses, and IoT monitoring systems. They play a critical role in safety, as excessive heat can damage components.

2.3 POTENTIOMETER

A potentiometer is a variable resistor used to adjust voltage or signal levels in a circuit. It consists of three terminals: two connected to a fixed resistor and one movable wiper terminal that changes resistance as the knob is rotated. This allows users to manually control the electrical output.

Potentiometers are often used for calibration or tuning in embedded systems. In Arduino projects, they are used to adjust LCD contrast, control LED brightness, set motor speed, or change sensor threshold values in real time. They act as voltage dividers, providing smooth and adjustable voltage to the controller.

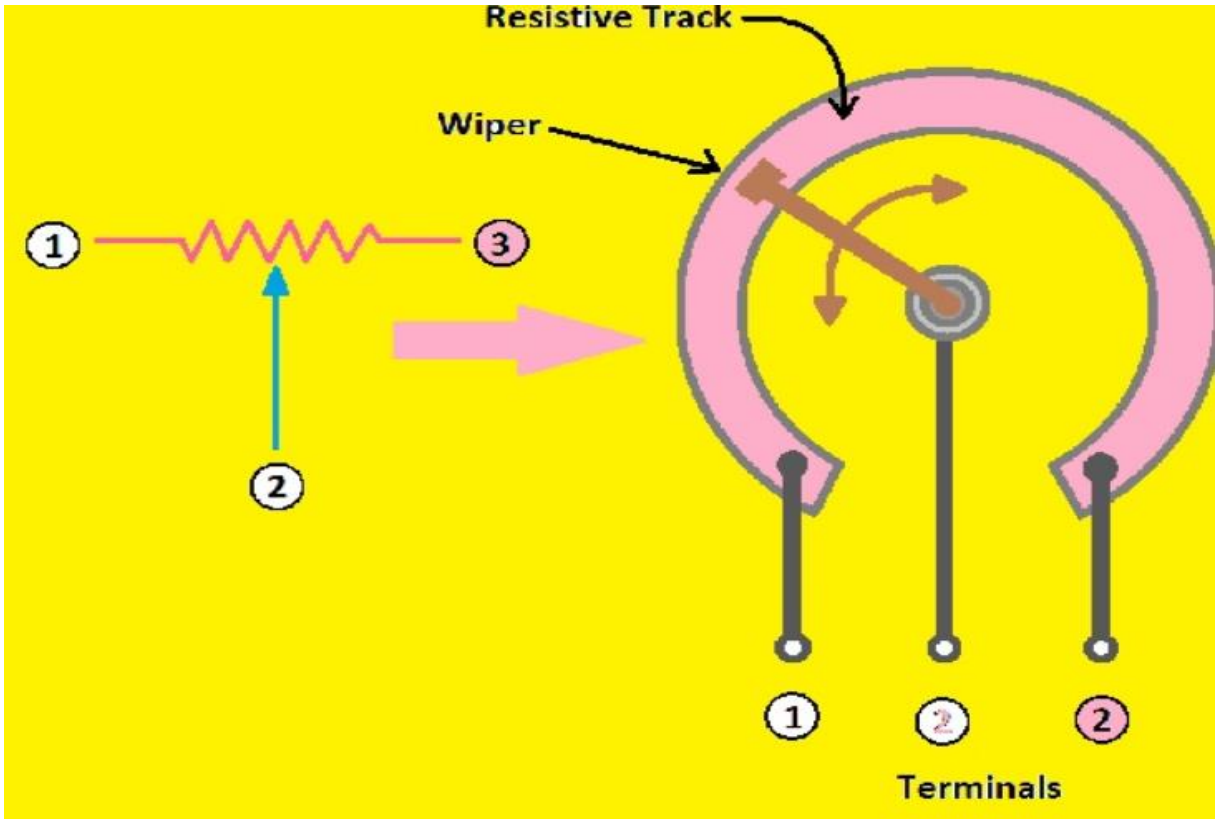

Fig:2.3:Potentiometer

These components are common in audio equipment, power supplies, industrial machines, and consumer electronics. They allow user interaction with the system without needing to modify the program each time. Their reliability, low cost, and ease of use make them a standard component in control circuits.

2.4 RESISTOR

A resistor is a passive electronic component that limits or controls the flow of electrical current in a circuit. It is measured in ohms (Ω) and is essential for protecting delicate components from excessive current. Without resistors, many components.

Fig:2.4:Resistor

Resistors are used to create voltage dividers, reduce signal levels, and control current flow. They also help stabilize circuits by setting correct biasing conditions for transistors and amplifiers. The value of a resistor is identified by a color code printed on its body, which represents its resistance level.

In Arduino and embedded system circuits, resistors are frequently paired with sensors, LEDs, relays, and communication modules. They ensure safe operation, reliable performance, and long component life. Because of their importance, resistors are present in almost every electronic device, from small toys to industrial machinery.

2.5 LCD (LIQUID CRYSTAL DISPLAY)

A Liquid Crystal Display (LCD) is a display module used to show numbers, characters, and messages. The commonly used type is a 16x2 LCD, which displays 16 characters per line across 2 lines. It helps users view system output like sensor.

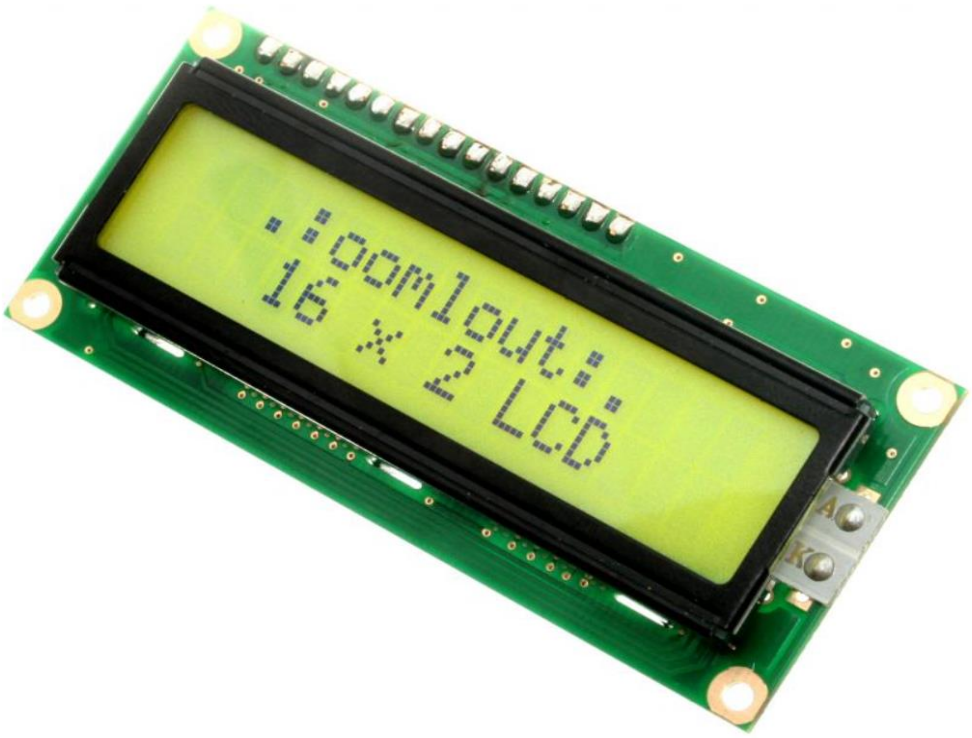

Fig:2.5:LCD

The LCD communicates with microcontrollers through parallel pins or using an I2C interface that reduces wiring. In Arduino-based systems, LCDs are used for debugging, displaying sensor outputs, menu controls, and status indication. They are easy to interface and require basic coding with built-in libraries.

LCDs are widely used in home appliances, medical devices, measurement systems, industrial displays, and portable electronics. Their low power consumption, clear visibility, and compatibility with embedded systems make them ideal for real-time projects. When combined with sensors and controllers, LCDs help in automation and monitoring applications.

CHAPTER 3

PROJECT

3.1 Smart Temperature Monitoring

The smart temperature monitoring project is designed to measure and display real-time temperature using a temperature sensor, Arduino Uno, and an LCD display. The system is assembled on a breadboard to simplify wiring, testing, and hardware understanding. This project helps monitor heat levels in environments like rooms, storage areas, and equipment sections.

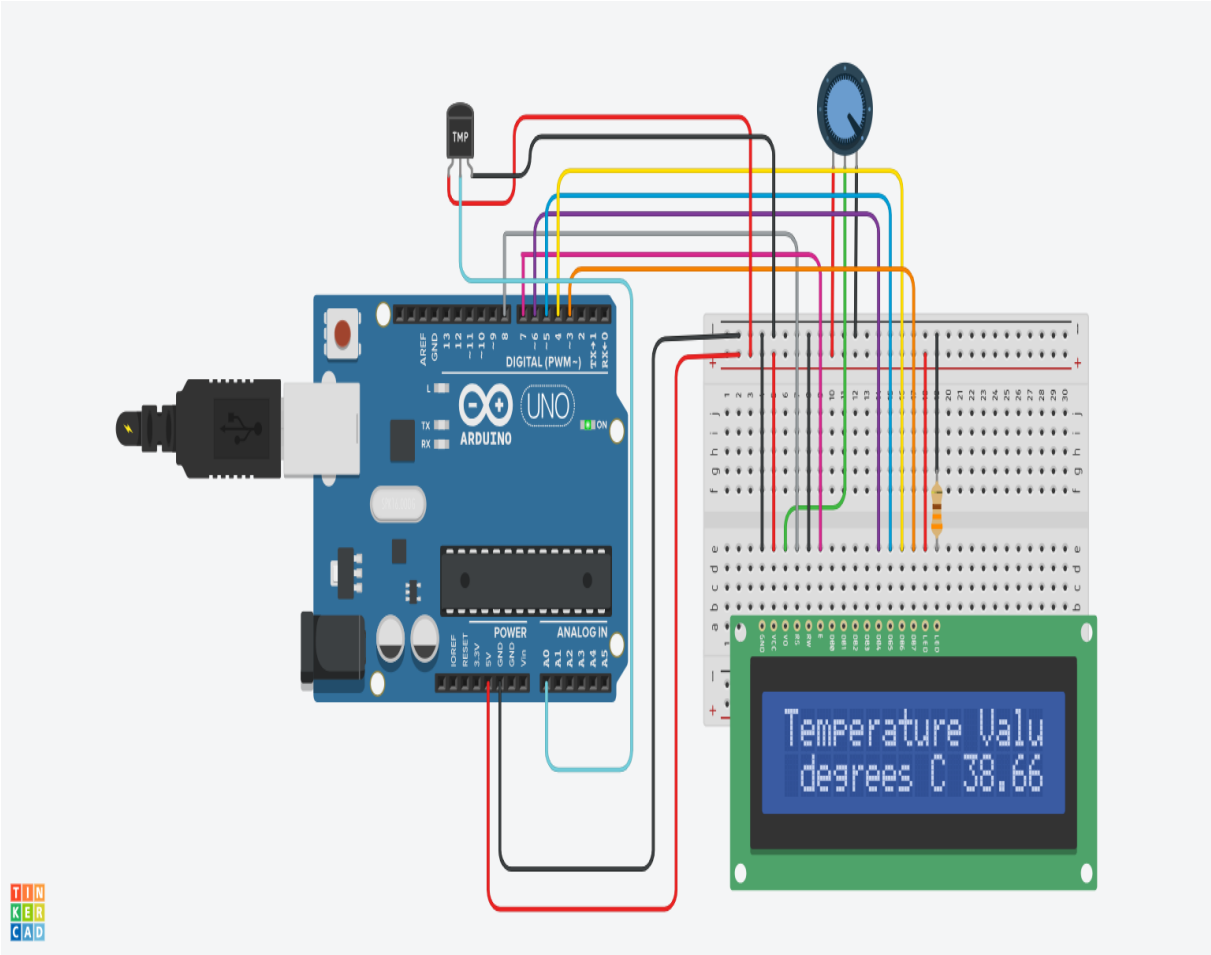

Fig:3.1:Circuit diagram

The **Arduino Uno** acts as the main controller of the entire system. It collects data from the temperature sensor, processes the input signal, and drives the LCD to show the result. The Arduino board is powered using a USB cable or a 9V external power supply. Digital and analog pins are used for communication with other components.

A **temperature sensor** (such as LM35 or DHT11) is used to measure the surrounding temperature. This sensor converts temperature into an electrical signal that the Arduino can read. The output pin of the sensor is connected to one of the analog input pins (A0) of the Arduino for voltage-based temperature conversion.

The sensor is placed on the breadboard, and jumper wires connect the VCC and GND pins to the power rails. The signal pin is routed to the analog input of the Arduino. As the temperature changes, the sensor's output voltage also changes, allowing the controller to measure temperature continuously.

A **resistor** is used in the circuit to protect components and ensure proper voltage regulation. Resistors prevent excessive current from damaging the sensor or LCD backlight. The resistor may also be paired with the LCD's backlight LED or the sensor power line to ensure safe operation.

A **potentiometer** is placed on the breadboard to adjust the contrast of the LCD screen. It is connected to the V0 contrast pin of the LCD, and by rotating the knob, the text visibility can be improved. Without the potentiometer, characters may appear too faint or too dark.

The **LCD display** (usually a 16x2) is used to show the temperature value in degrees Celsius. It is connected to Arduino through digital pins in 4-bit or 8-bit communication mode. The LCD receives data from the controller and prints the measured temperature so the user can easily read it.

The breadboard acts as the base for arranging components. It eliminates the need for soldering, making the project flexible and error-free. Horizontal and vertical rails on the breadboard allow common power distribution so all components receive stable supply from Arduino.

When the system powers ON, the Arduino initializes the LCD using programmed instructions. The temperature sensor starts sensing the surrounding heat and sends voltages proportional to the measured temperature. The Arduino's analog-to-digital converter reads the analog voltage and converts it into a temperature value.

The software logic programmed into the Arduino IDE calculates the temperature based on the sensor formula. For example, the LM35 provides 10mV per degree Celsius, so Arduino converts this voltage into a readable temperature. The calculated value is refreshed continuously to give real-time monitoring.

The LCD updates the temperature reading every second or according to the programmed delay. The display might show:
“Temperature:38.66°C”. This live monitoring helps the user respond if temperature rises beyond a normal limit.

This project demonstrates how sensors interact with microcontrollers in real-time systems. It also shows the importance of supporting components: potentiometer for LCD clarity, resistor for protection, and breadboard for hardware prototyping. The working principle follows: **Sense → Process → Display.**

This setup is useful in practical fields like food storage monitoring, greenhouse temperature control, industrial machine safety, and medical refrigeration. It can also be extended for IoT usage by adding Wi-Fi modules like ESP8266 to send data to a smartphone.

In conclusion, the smart temperature monitoring project using Arduino Uno, temperature sensor, potentiometer, resistor, LCD, and breadboard provides a complete foundation for learning embedded system automation. It teaches interfacing, sensor handling, display control, and real-time processing—making it an ideal beginner-to-intermediate level project.

CHAPTER 4

SOURCE CODE

```
#include "LiquidCrystal.h"        // Include the library to control the LCD display

LiquidCrystal lcd(8,7,6,5,4,3); // Create LCD object with pin connections (RS=8, E=7, D4=6, D5=5, D6=4, D7=3)

int sensorPin = 0;                // Analog pin A0 is used to read temperature sensor

void setup()

{

  Serial.begin(9600);           // Start serial communication at 9600bps for Serial Monitor

  lcd.begin(16,2);              // Initialize LCD with 16 columns and 2 rows

}

void loop()

{

int reading = analogRead(sensorPin);  // Read the analog value (0-1023) from temperature sensor at A0

  // Multiply by 4.68V (your measured board voltage, normally 5V but USB varies)

  float voltage = reading * 4.68;       // Convert ADC reading to voltage reference

  voltage /= 1024.0;                    // Divide by ADC resolution (1024 steps)

  float temperatureC = (voltage - 0.5) * 100;  // Convert voltage to temperature in Celsius

  Serial.print(temperatureC);           // Print temperature value to Serial Monitor

  Serial.println(" degrees C");         // Print unit "degrees C" on next line

  lcd.setCursor(0,0);                   // Position cursor on first row, first column

  lcd.print("Temperature Value ");// Print title text on LCD first line

  lcd.setCursor(0,1);                   // Move cursor to start of second row

  lcd.print(" degrees C");              // Print text "degrees C" on LCD

  lcd.setCursor(11,1);                  // Move cursor to print temperature at correct position

  lcd.print(temperatureC);              // Display actual temperature reading on LCD

  delay(100);                           // Small delay (100ms) before next reading

}
```

CHAPTER 5

CONCLUSION

The smart temperature monitoring project successfully demonstrates how embedded systems can be used for real-time environmental data tracking. By integrating an Arduino Uno with a temperature sensor, the system measures heat levels continuously and displays the values on an LCD. This proves that automated monitoring offers more accuracy and reliability than manual temperature checking methods.

Each component plays an important role in the working of the project. The temperature sensor captures the real-time temperature, the Arduino processes the data, and the LCD displays the output clearly. The potentiometer helps adjust the LCD contrast for better visibility, while the resistor ensures safe current flow for component protection. The breadboard makes the circuit flexible, easy to test, and beginner-friendly.

The project highlights the importance of real-time monitoring in daily life and industrial applications. It can be used in homes for room temperature tracking, in agriculture for greenhouse control, in hospitals to maintain medical storage conditions, and in food storage warehouses where temperature stability is crucial. The system can be expanded further to include alerts or automated cooling mechanisms.

This project also improves understanding of interfacing sensors, coding in Arduino IDE, handling analog inputs, and managing electrical components. It introduces the user to basic automation concepts, which can be scaled up to Internet of Things (IoT)-based monitoring by adding wireless modules like ESP8266 or GSM for remote data access. This makes the project a good foundation for advanced development.

Overall, the smart temperature monitoring system is an efficient, low-cost, and practical embedded project that reflects modern automation needs. It proves that simple hardware components, when combined with logical programming, can create effective solutions for real-world problems. This project not only measures temperature but also opens the path for innovation, improvement, and future technological expansion.