**Title, intro, components, working of sensor, specs of sensor, working of actuator, spec of actuator, final outcome, code, images**

**Continuous Assessment 1**

**CA\_1 : Interfacing of sensors and actuators with three different hardware platforms**

**Title: Temperature Logging and Alert System**

1. **Introduction**:

The Temperature Monitoring and Alert System is a project that utilizes Arduino's capabilities to monitor real-time temperature using a DHT11 sensor and provides visual feedback through an LED indicator. It is designed to demonstrate a simple yet effective application of Arduino in environmental monitoring and alerting. Applications of this would be - Monitor room temperature, Plant Growth Monitoring: Take action if temperature levels become unfavorable for growth, Server Room Monitoring: Deploy the system in server rooms to prevent equipment overheating.

Functionality -

* The DHT11 sensor continuously measures the temperature of the environment and sends the data to Arduino.
* It processes the temperature data and compares it with a predefined threshold value, in this case, 25 degrees.
* If the temperature exceeds the threshold value, the Arduino activates the digital pin associated with the buzzer and the buzzer alerts us.

**Components Used:**

* Arduino Uno: The brain of the system that controls the entire operation. It is a widely-used microcontroller board featuring the ATmega328P microcontroller, 14 digital input/output pins, 6 analog input pins, USB interface for programming. It operates at 5V. It has 2 pins providing 5V and 3.3V
* DHT11: Temperature and humidity sensor
* Buzzer: For alerting with sound
* Jumper wires: For connecting different components to each other
* Breadboard: To organize connections

**Working of sensor:**

* A sensor capable of accurately measuring temperature and humidity in the surroundings. It is important to note that it may not be suitable for applications that require high precision measurements. It has limitations in terms of its refresh rate, accuracy, and performance in extreme conditions.

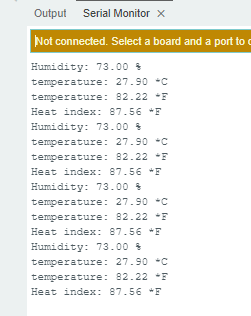
It works on the principle of capacitance measurement. The temperature-sensing component uses a thermistor (resistor whose resistance changes with temperature) to measure the ambient temperature. The resistance of the thermistor changes with temperature, and by measuring its resistance, the sensor can determine the temperature.

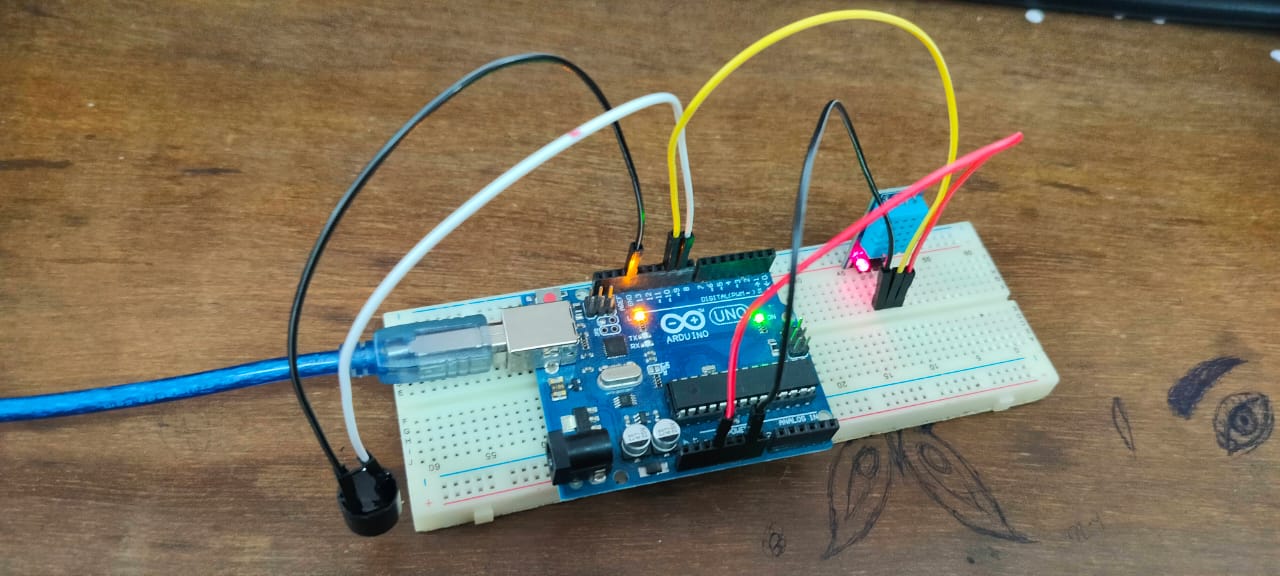
The humidity-sensing component of the DHT11 consists of a moisture-absorbing substrate that changes its electrical conductivity based on the surrounding humidity. This substrate is placed between two electrodes. As humidity increases, the conductivity changes, and the sensor measures the time it takes for the capacitance to discharge, providing a humidity value.

**Specifications of sensor:**

* Operating Voltage: 3.3V to 5.5V DC
* Humidity Measurement Range: 20% to 90% RH (Relative Humidity)
* Humidity Accuracy: ±5% RH
* Temperature Measurement Range: 0°C to 50°C
* Temperature Accuracy: ±2°C
* Sampling Rate: 1 Hz (1 reading per second)
* Output: Digital signal (Single-wire interface)
* Data Transmission: 40-bit data packet, including humidity and temperature information
* Dimensions: Approx. 15.5mm x 12mm x 5.5mm

**Working of actuator**





**Code :**

#include <DHT.h>

#define DHTPIN 10 // DHT11 data pin

#define DHTTYPE DHT11 // DHT11 sensor type

DHT dht(DHTPIN, DHTTYPE);

#define BUZZER\_PIN 9 // Digital pin for buzzer control

void setup() {

pinMode(BUZZER\_PIN, OUTPUT);

Serial.begin(9600);

dht.begin();

}

void loop() {

// Wait a few seconds between measurements.

delay(2000);

// Reading temperature or humidity takes about 250 milliseconds!

float h = dht.readHumidity();

float t = dht.readTemperature();

float f = dht.readTemperature(true);

if (isnan(h) || isnan(t) || isnan(f)) {

Serial.println("Failed to read from DHT sensor!");

return;

}

// Check if the temperature is above a threshold

if (t > 25.0) {

digitalWrite(BUZZER\_PIN, HIGH); // Turn the buzzer on

delay(1000); // Keep the buzzer on for 1 second

digitalWrite(BUZZER\_PIN, LOW); // Turn the buzzer off

}

// Print sensor data

float hi = dht.computeHeatIndex(f, h);

Serial.print("Humidity: ");

Serial.print(h);

Serial.println(" %\t");

Serial.print("temperature: ");

Serial.print(t);

Serial.println(" \*C ");

Serial.print("temperature: ");

Serial.print(f);

Serial.println(" \*F\t");

Serial.print("Heat index: ");

Serial.print(hi);

Serial.println(" \*F");

delay(5000); // Wait for 5 seconds before the next loop

}

47 - Urvi :

67 - Manav :

const int trigPin = 7;

const int echoPin = 8;

const int ledPin = 9;

void setup() {

Serial.begin(9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(ledPin, OUTPUT);

}

void loop() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long duration = pulseIn(echoPin, HIGH);

int distance = duration \* 0.0343 / 2;

Serial.print("Distance: ");

Serial.println(distance);

if (distance <= 10) { // Change the threshold distance as needed

digitalWrite(ledPin, HIGH);

} else {

digitalWrite(ledPin, LOW);

}

}

32 - Hrishikesh

**Title: Temperature Logging and Alert System**

**Introduction:**

The Temperature Monitoring System using Arduino Uno and LM35 sensor is a versatile and cost-effective solution designed to accurately measure and monitor temperature in various environments. This system utilizes the Arduino Uno microcontroller and the LM35 analog temperature sensor to collect temperature data, process it, and present the results in a meaningful way.

**Components Used:**

**1. ESP32:** The core component of the system, the ESP32 is a powerful microcontroller module based on the Espressif ESP32 SoC (System-on-Chip). It offers substantial processing power, extensive I/O capabilities, and built-in Wi-Fi and Bluetooth connectivity. The ESP32 is responsible for interfacing with the LM35 temperature sensor and displaying temperature data, making it a versatile choice for IoT and sensor applications.

**2. LM35 Temperature Sensor:** The LM35 is a precision analog sensor that provides an accurate voltage output proportional to the temperature it senses. It is easy to use and does not require external calibration or signal conditioning.

**3. Breadboard and Jumper Wires:** These components are used to create the necessary connections between the Arduino Uno, LM35 sensor, and any additional components like displays or indicators.

**4. Display:** Arduino Serial Monitor to visually present the measured temperature.

**Functionality:**

**1.** **Temperature Sensing:** The LM35 sensor detects the ambient temperature and generates an analog voltage output linearly proportional to the temperature.

**2. Analog-to-Digital Conversion:** The analog voltage from the LM35 is read by one of the analog input pins on the Arduino Uno. The Arduino's built-in analog-to-digital converter (ADC) converts this analog signal into a digital value.

**3. Data Processing:** The digital temperature value is processed by the Arduino using mathematical calculations to convert the digital reading into actual temperature in degrees Celsius or Fahrenheit.

**4. Display :** The Arduino will show the temperature reading on the display, providing real-time information to the user.

**Benefits:**

**1. Accuracy:** The LM35 sensor is known for its high accuracy and linearity over a wide temperature range, making the system reliable for various applications.

**2. Ease of Use:** The LM35 is straightforward to connect and use, and the Arduino platform simplifies programming and interfacing with the sensor.

**3. Cost-Effective:** Both the Arduino Uno and LM35 sensor are affordable components, making the overall system cost-effective for temperature monitoring.

**4. Customization:** The system can be easily customized by incorporating additional components like displays, data logging capabilities, or remote connectivity options.

**5. Versatility:** This system can be employed in a variety of settings, such as home temperature monitoring, industrial process control, environmental monitoring, and more.

**Applications:**

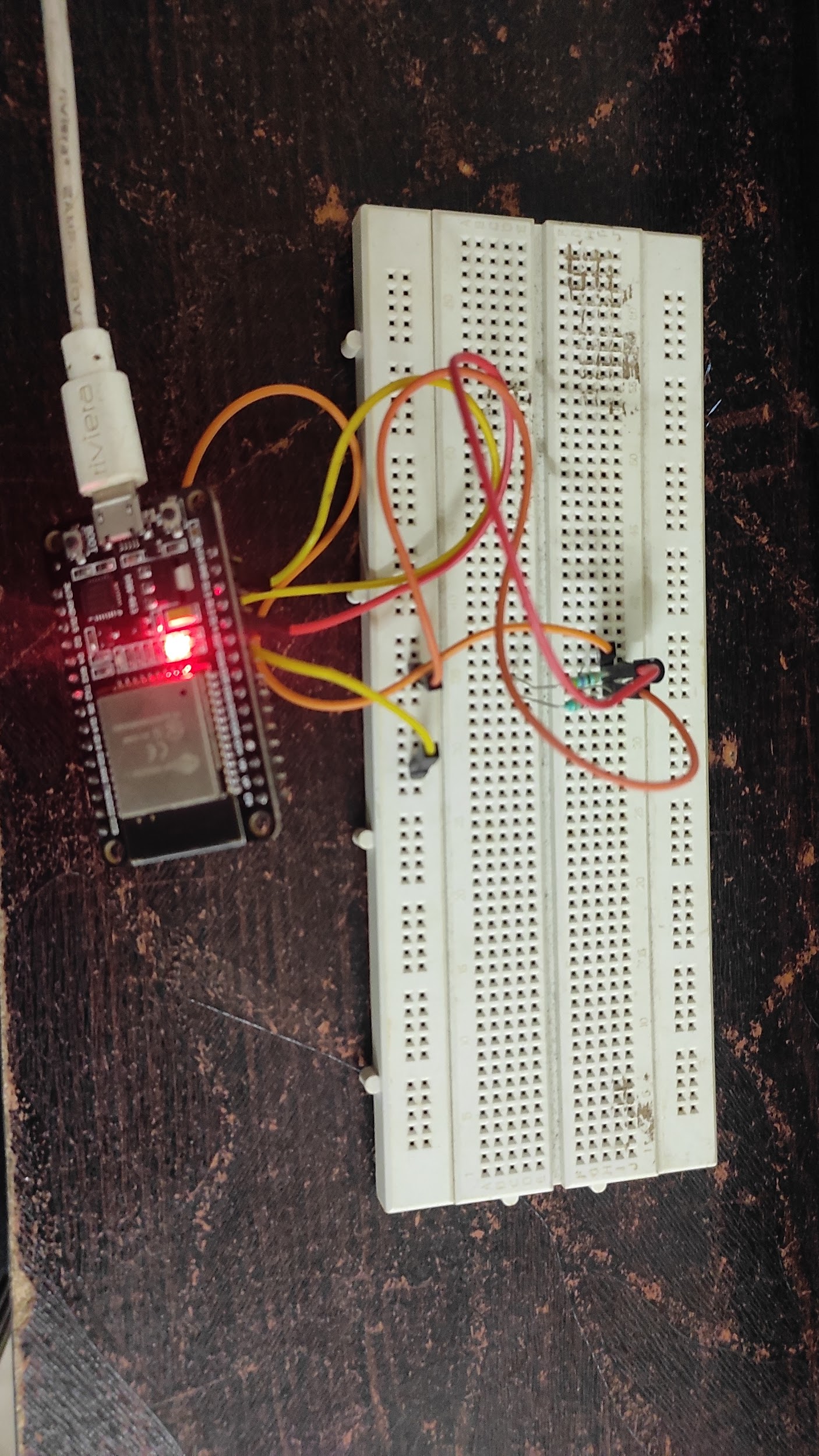
**Home Climate Control:** The system can be used to monitor indoor temperatures and control heating, ventilation, and air conditioning (HVAC) systems to maintain a comfortable living environment.

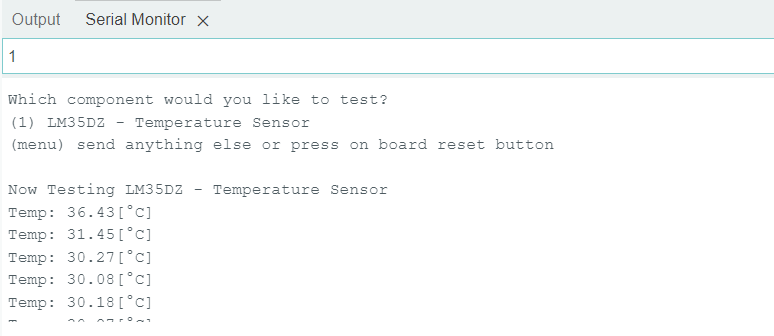
**Industrial Process Control:** Many industrial processes require precise temperature control. This system can monitor and adjust temperatures in manufacturing processes such as chemical reactions, food production, and more.

**Environmental Monitoring:** It can be employed to monitor temperature variations in outdoor environments, helping with weather stations, agriculture, and ecology studies.

**Data Centers:** Temperature monitoring is critical in data centers to prevent overheating and ensure the stable operation of servers and networking equipment.

**Food Storage and Transport:** The system can be used to monitor temperatures in refrigerators, freezers, and transport containers, ensuring that perishable goods are stored and transported under proper conditions.





**Code:**

// Include Libraries

#include "Arduino.h"

#include "LM35.h"

// Pin Definitions

#define LM35\_PIN\_VOUT 14

// Global variables and defines

// object initialization

LM35 lm35(LM35\_PIN\_VOUT);

// define vars for testing menu

const int timeout = 10000; //define timeout of 10 sec

char menuOption = 0;

long time0;

// Setup the essentials for your circuit to work. It runs first every time your circuit is powered with electricity.

void setup()

{

// Setup Serial which is useful for debugging

// Use the Serial Monitor to view printed messages

Serial.begin(9600);

while (!Serial) ; // wait for serial port to connect. Needed for native USB

Serial.println("start");

menuOption = menu();

}

// Main logic of your circuit. It defines the interaction between the components you selected. After setup, it runs over and over again, in an eternal loop.

void loop()

{

if(menuOption == '1') {

// LM35DZ - Temperature Sensor - Test Code

//Get Measurment from LM35 temperature sensor.

float lm35TempC = lm35.getTempC();

Serial.print(F("Temp: ")); Serial.print(lm35TempC); Serial.println(F("[°C]"));

}

if (millis() - time0 > timeout)

{

menuOption = menu();

}

}

// Menu function for selecting the components to be tested

// Follow serial monitor for instrcutions

char menu()

{

Serial.println(F("\nWhich component would you like to test?"));

Serial.println(F("(1) LM35DZ - Temperature Sensor"));

Serial.println(F("(menu) send anything else or press on board reset button\n"));

while (!Serial.available());

// Read data from serial monitor if received

while (Serial.available())

{

char c = Serial.read();

if (isAlphaNumeric(c))

{

if(c == '1')

Serial.println(F("Now Testing LM35DZ - Temperature Sensor"));

else

{

Serial.println(F("illegal input!"));

return 0;

}

time0 = millis();

return c;

}

}

}

In conclusion, the Temperature Monitoring System using Arduino Uno and LM35 sensor offers an accurate, cost-effective, and versatile solution for measuring and displaying temperature data. Whether for personal use or more complex applications, this system provides a foundation for temperature monitoring and data processing.