# Introduction

In today’s technological landscape, the use of sensors for real-time monitoring has become increasingly critical across various industries and applications. Sensors allow systems to detect and respond to environmental changes without the need for human intervention. This project focuses on creating a responsive and efficient system that integrates two different sensors — an ultrasonic sensor and an LM35 temperature sensor — with the ESP32 microcontroller to control a buzzer and an LED for alert signaling.

## Why ESP32?

The ESP32 microcontroller was chosen for this project due to its versatility, built-in wireless capabilities, and ability to handle multiple sensors simultaneously. It offers:

- **Multiple GPIO pins**: Essential for interfacing with multiple input/output devices like sensors and actuators.
- **Analog-to-digital conversion (ADC)**: Required for reading analog signals from the LM35 temperature sensor.
- **Fast processing**: Crucial for real-time applications where delays in response could lead to system inefficiencies.

## Functionality of the System

The core functionality of this system is based on monitoring two parameters:

1. **Distance**: Using the HC-SR04 ultrasonic sensor, the system continuously measures the distance between the sensor and any object in its line of sight. When an object is detected within a certain threshold (less than 20 cm), the system triggers an auditory (buzzer) and visual (LED) alert.
2. **Temperature**: The LM35 temperature sensor measures the surrounding temperature. When the temperature exceeds 40°C, the system triggers the same alarm components.

The buzzer and LED act as the system's alarm indicators, providing both visual and auditory signals when the thresholds for distance or temperature are breached. This makes the system versatile enough to handle multiple environmental conditions in real time.

## Components Used

In this project, we use a combination of sensors, actuators, and the ESP32 microcontroller to build an intelligent alarm system. Each component plays a vital role in ensuring the functionality of the system, and its integration with the ESP32 makes it possible to read, process, and act on environmental data. Below is a detailed explanation of each component and its purpose:

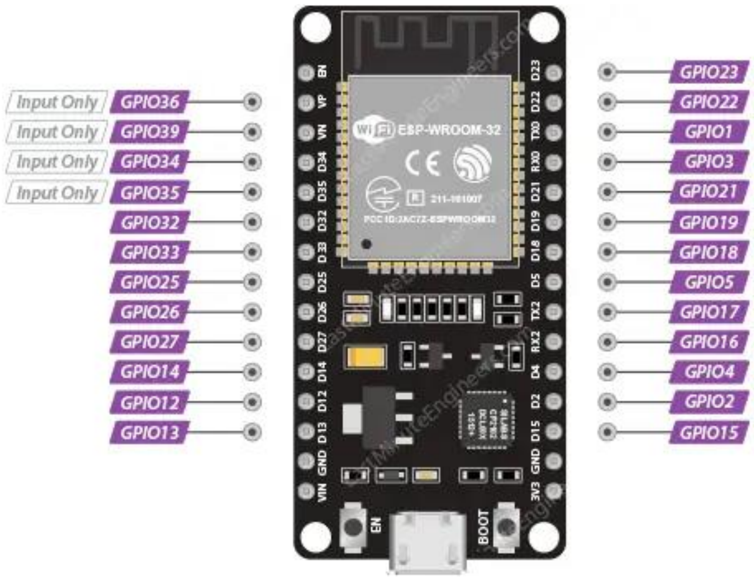
### 1. ESP32 Microcontroller

The ESP32 is a powerful, low-cost microcontroller with integrated Wi-Fi and Bluetooth capabilities. It is widely used in IoT (Internet of Things) projects due to its flexibility, multiple GPIO (General Purpose Input/Output) pins, and robust processing power.

**Key Features:**

- **Dual-core processor**: Enables simultaneous task handling, such as reading from sensors while controlling actuators like the buzzer and LED.
- **Built-in ADC (Analog-to-Digital Converter)**: Allows the ESP32 to interface with analog sensors like the LM35 temperature sensor, converting analog voltage to a digital value for further processing.
- **Wi-Fi and Bluetooth**: Although not used in this project, the ESP32 can connect to the internet or other devices, making it ideal for remote monitoring systems.
- **Multiple GPIO pins**: Provides several input and output connections for interfacing with multiple components like sensors, LEDs, and buzzers.
- **Low power consumption**: Suitable for battery-operated systems, the ESP32 efficiently balances performance and energy use.

In this project, the ESP32 is the central controller. It reads data from the ultrasonic sensor (for distance) and the LM35 sensor (for temperature), processes this information, and activates the buzzer and LED when preset conditions are met.



## 2. Ultrasonic Sensor (HC-SR04)

The **HC-SR04** ultrasonic sensor is commonly used for measuring distance by emitting ultrasonic sound waves and detecting the time it takes for them to bounce back after hitting an object. It is a popular choice for proximity and obstacle detection in robotics and automation projects due to its accuracy and ease of use.

**Key Features:**

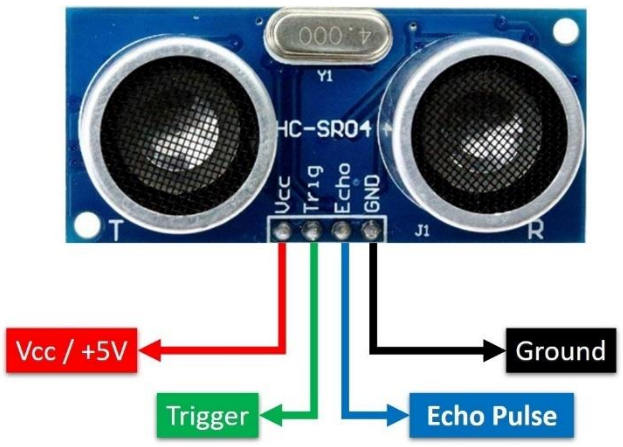
- **Operating Voltage**: 5V
- **Measuring Range**: 2 cm to 400 cm
- **Accuracy**: ±3 mm
- **Sound Wave Frequency**: 40 kHz
- **Working Principle**: Time-of-flight measurement (calculating the time taken by ultrasonic pulses to travel to an object and reflect back to the sensor).

**Working:**

- The sensor consists of two main parts: a **transmitter** (which emits ultrasonic waves) and a **receiver** (which listens for the reflected waves).
- When an object is detected, the sensor measures the time taken for the ultrasonic wave to bounce back, calculates the distance using the formula:

  Distance = (Time × Speed of Sound)/2 , where the speed of sound is approximately 343 m/s.

In this project, the ultrasonic sensor is connected to the ESP32, which continuously monitors the distance to nearby objects. If the distance falls below 20 cm, the system triggers an alarm (buzzer and LED). The HC-SR04 is an ideal choice for this task due to its affordability and ability to measure both short and long distances.



## 3. LM35 Temperature Sensor

The **LM35** is a precision temperature sensor that provides an analog voltage output directly proportional to the temperature in Celsius. It is widely used for temperature monitoring due to its accuracy and linear output. Unlike thermistors, the LM35 does not require external calibration and is factory calibrated for high accuracy.

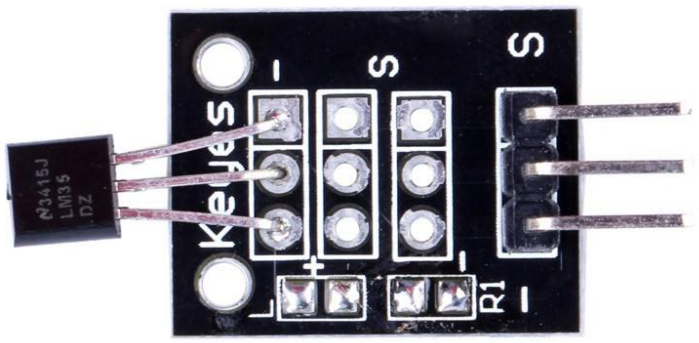
**Key Features:**

- **Operating Voltage**: 4 to 30V
- **Temperature Range**: -55°C to +150°C
- **Accuracy**: ±0.5°C at room temperature
- **Output Voltage**: 10 mV per degree Celsius rise in temperature.
- **Low Self-Heating**: Ensures minimal influence of the sensor's own heat on the temperature reading.

**Working:**

- The LM35 outputs an analog voltage proportional to the temperature, with 10 mV corresponding to each degree Celsius. For example, at 25°C, the output will be 250 mV.
- The ESP32’s ADC (Analog-to-Digital Converter) reads this voltage and converts it into a digital value, which is then processed to display the temperature in both Celsius and Fahrenheit.

In this project, the LM35 continuously monitors the ambient temperature. If the temperature exceeds 40°C, the system triggers an alarm, alerting users to possible overheating or dangerous temperature levels. The LM35’s precision and simple interfacing make it a perfect choice for this real-time monitoring system.



## 4. Buzzer

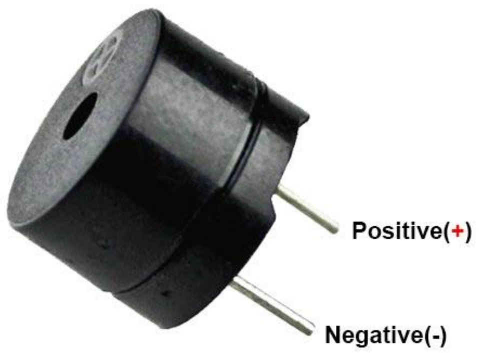
The **buzzer** is an electroacoustic component that generates a loud sound when powered. It is used in a wide range of applications where an audible alert or warning signal is required.

**Key Features:**

- **Operating Voltage**: 3V to 12V
- **Sound Output**: Varies between different types of buzzers (typically 70-90 dB for small buzzers)
- **Simple ON/OFF Control**: Can be easily triggered by a microcontroller's GPIO pin.

**Working:**

- The buzzer is driven by the ESP32's GPIO pin. When the conditions (either low distance or high temperature) are met, the GPIO pin goes high, activating the buzzer and generating sound.
- In this project, the buzzer serves as an auditory alert system. When the distance falls below 20 cm or the temperature exceeds 40°C, the buzzer is activated, alerting users of potential hazards or proximity warnings.



## Circuit Connections

• Ultrasonic **Sensor (HC-SR04)**

- **VCC** → 5V (ESP32)
- **GND** → GND (ESP32)
- **TRIG** → GPIO 5 (ESP32)
- **ECHO** → GPIO 18 (ESP32)

• Buzzer

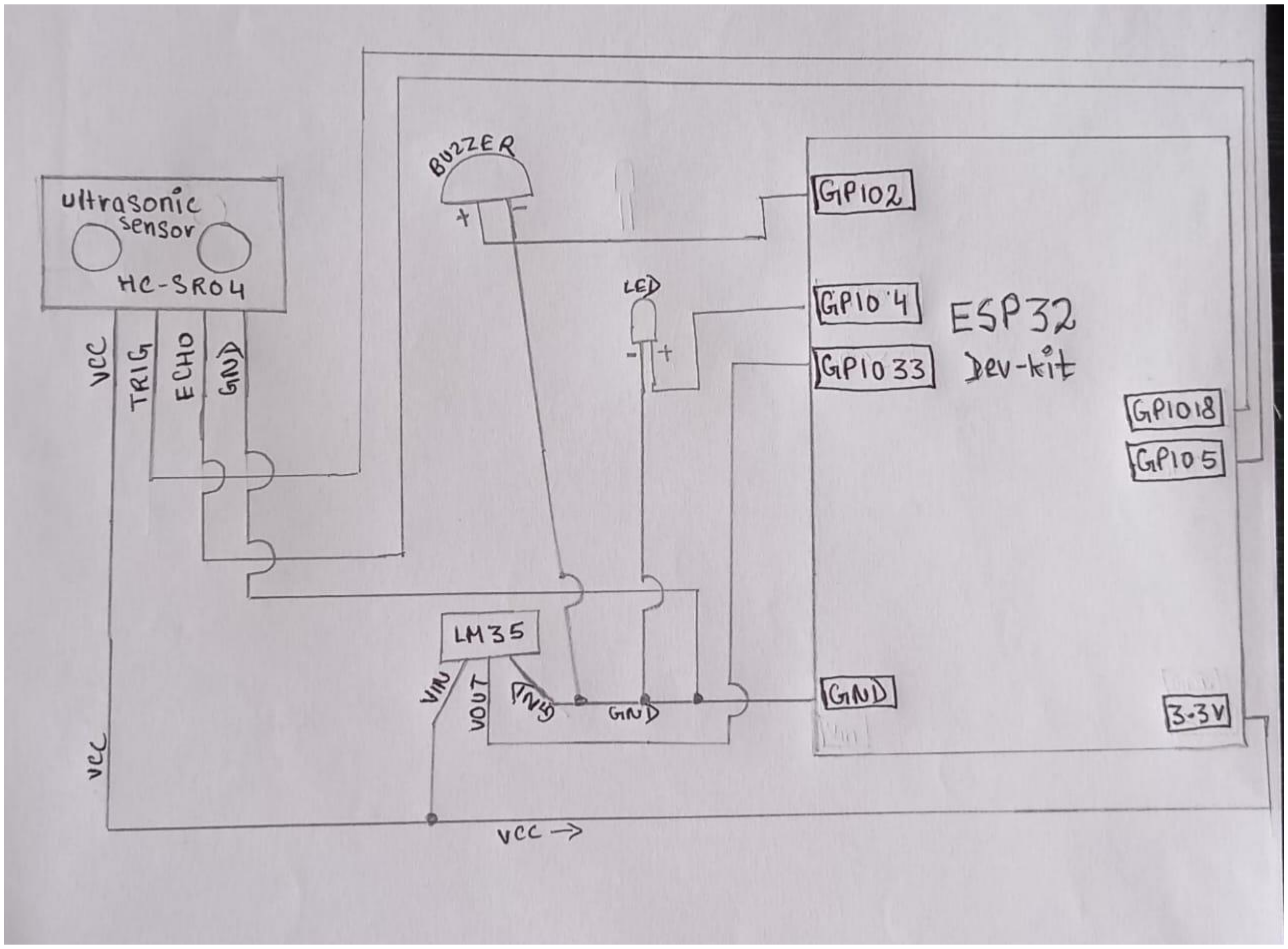
- **VCC** → 5V (ESP32)
- **GND** → GND (ESP32)
- **Signal** → GPIO 2 (ESP32)

• LM**35 Temperature Sensor**

- **VCC** → 3.3V (ESP32)
- **GND** → GND (ESP32)
- **Output** → GPIO 33 (ESP32)

• LED

- **VCC** → GPIO 4 (ESP32)
- **GND** → GND (ESP32)



## Code Implementation

```
// Distance Sensor Definitions
#define BUZZER_PIN 2
#define TRIG_PIN 5
#define ECHO_PIN 18
#define LED_PIN 4           // Define the LED pin
#define MAX_DISTANCE 100     // Maximum distance threshold (100 cm)

// Temperature Sensor Definitions
#define ADC_VREF_mV 3300.0   // ESP32's ADC reference voltage in millivolts
#define ADC_RESOLUTION 4095.0 // ESP32's ADC resolution (12-bit)
#define PIN_LM35 33          // Pin connected to LM35 sensor
#define TEMP_THRESHOLD 40.0  // Temperature threshold in °C for buzzer

void setup() {
  // Distance Sensor Setup
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);  // Set LED pin as output
  Serial.begin(115200);      // Initialize serial communication for distance
readings

  // Temperature Sensor Setup
  Serial.begin(9600);        // Initialize serial communication for temperature
readings
}
long readDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);      // Standard 10us pulse width
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH, 6000);  // Timeout for 1 meter (6ms)

  // If duration is 0, no echo was received, return 0 cm
  if (duration == 0) {
    return 0;  // No object detected
  }
  // Convert the time into a distance in cm
  long distance = duration * 0.034 / 2;

  // Return distance if it's less than the maximum distance threshold
  if (distance <= MAX_DISTANCE) {
    return distance;
  } else {
    return 0;  // Return 0 if the distance is out of valid range
  }
}
void loop() {
  // Distance Measurement
  long distance = readDistance();
  // Print the distance
  if (distance > 0) {
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");
  } else {
    Serial.println("No object detected");
  }
  // Temperature Measurement
  int adcVal = analogRead(PIN_LM35);  // Read the ADC value from the LM35
  // Convert ADC value to voltage in millivolts
  float milliVolt = (adcVal * ADC_VREF_mV) / ADC_RESOLUTION;
  // Convert voltage to temperature (LM35 gives 10mV per degree Celsius)
  float tempC = milliVolt / 10.0;
  float tempF = tempC * 9.0 / 5.0 + 32.0;
  // Print ADC value, temperature in both Celsius and Fahrenheit
  Serial.print("ADC Value: ");
  Serial.println(adcVal);
  Serial.print("Temperature: ");
  Serial.print(tempC);
  Serial.print("°C");
  Serial.print("  ~  ");
  Serial.print(tempF);
  Serial.println("°F");

  // Activate the buzzer and LED if the object is within 20 cm or if temperature
is above 40°C
  if ((distance > 0 && distance < 20) || tempC > TEMP_THRESHOLD) {
    digitalWrite(BUZZER_PIN, HIGH);
    digitalWrite(LED_PIN, HIGH);  // Turn LED on
  } else {
    digitalWrite(BUZZER_PIN, LOW);
    digitalWrite(LED_PIN, LOW);   // Turn LED off
  }
  delay(2000);  // Delay for 2 seconds between readings
}
```

## Code Overview

The executed code integrates an ultrasonic sensor (HC-SR04) and a temperature sensor (LM35) with an ESP32 microcontroller to create a dual-purpose alarm system. The system triggers a buzzer and an LED when either the distance to an object is less than 20 cm or the temperature exceeds 40°C. The code is divided into several sections for sensor initialization, data reading, and condition-based alarm activation.

Distance Sensor Definitions:

```
#define BUZZER_PIN 2
#define TRIG_PIN 5
#define ECHO_PIN 18
#define LED_PIN 4          // Define the LED pin
#define MAX_DISTANCE 100     // Maximum distance threshold (100 cm)
```

- BUZZER_**PIN**: Pin 2 of the ESP32 is assigned to control the buzzer.
- TRIG_**PIN**: Pin 5 is connected to the **TRIG** pin of the ultrasonic sensor. The ESP32 sends a signal to this pin to generate an ultrasonic pulse.
- ECHO_**PIN**: Pin 18 is connected to the **ECHO** pin of the ultrasonic sensor. This pin receives the pulse reflected back from the object.
- LED_**PIN**: Pin 4 is assigned to the LED, which lights up when the alarm conditions are met.
- MAX_**DISTANCE**: Set to 100 cm, this defines the upper limit for the distance sensor to trigger an object detection response.

Temperature Sensor Definitions:

```
#define ADC_VREF_mV 3300.0   // ESP32's ADC reference voltage in millivolts
#define ADC_RESOLUTION 4095.0 // ESP32's ADC resolution (12-bit)
#define PIN_LM35 33          // Pin connected to LM35 sensor
#define TEMP_THRESHOLD 40.0  // Temperature threshold in °C for buzzer
```

- ADC_**VREF_mV**: Reference voltage of the ESP32's ADC is 3300 mV (or 3.3V). This is used to convert the ADC reading into a voltage that corresponds to the temperature.
- ADC_**RESOLUTION**: The ESP32 has a 12-bit ADC, so the resolution is 4095 (2^12 - 1). This value is used to map the analog input (0-3.3V) to a digital output range of 0-4095.
- PIN_**LM35**: Pin 33 is connected to the LM35 temperature sensor to read analog data.
- TEMP_**THRESHOLD**: Defines the temperature threshold at 40°C, above which the alarm (buzzer and LED) will be activated.

Setup Function:

```
void setup() {
  // Distance Sensor Setup
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);  // Set LED pin as output
  Serial.begin(115200);      // Initialize serial communication for distance readings
  // Temperature Sensor Setup
  Serial.begin(9600);        // Initialize serial communication for temperature readings
}
```

• pinMode(): Configures the buzzer, TRIG, ECHO, and LED pins as either inputs or outputs. The TRIG pin and LED are outputs, while the ECHO pin is an input to receive the reflected pulse from the ultrasonic sensor.
• Serial.begin(): Initializes the serial monitor for debugging and printing data at a baud rate of 115200 for distance sensor readings and 9600 for temperature sensor readings.

Distance Reading Function:

```
long readDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);      // Standard 10us pulse width
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH, 6000);  // Timeout for 1 meter (6ms)
  // If duration is 0, no echo was received, return 0 cm
  if (duration == 0) {
    return 0;  // No object detected
  }
  // Convert the time into a distance in cm
  long distance = duration * 0.034 / 2;
  // Return distance if it's less than the maximum distance threshold
  if (distance <= MAX_DISTANCE) {
    return distance;
  } else {
    return 0;  // Return 0 if the distance is out of valid range
  }
}
```

• digitalWrite(): Sends a HIGH pulse to the **TRIG_PIN** for 10 microseconds to trigger the ultrasonic sensor, which emits an ultrasonic pulse.

• pulseIn**()**: This function measures the duration in microseconds that the **ECHO_PIN** remains HIGH, which corresponds to the time taken for the ultrasonic wave to bounce back after hitting an object.
• Duration **Calculation**: The formula converts the duration into distance using the speed of sound (0.034 cm/µs). The distance is halved because the signal travels to the object and back.
• Distance **Threshold**: The distance is only valid if it's less than or equal to the **MAX_DISTANCE** (100 cm). Otherwise, the function returns `0`, indicating no object detected or the object is out of range.

Main Loop Function:

```
void loop() {
  // Distance Measurement
  long distance = readDistance();
  // Print the distance
  if (distance > 0) {
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");
  } else {
    Serial.println("No object detected");
  }
  // Temperature Measurement
  int adcVal = analogRead(PIN_LM35);  // Read the ADC value from the LM35
  float milliVolt = (adcVal * ADC_VREF_mV) / ADC_RESOLUTION;  // Convert ADC value to
voltage in millivolts
  float tempC = milliVolt / 10.0;  // Convert voltage to temperature (10mV per °C)
  float tempF = tempC * 9.0 / 5.0 + 32.0;  // Convert Celsius to Fahrenheit
  // Print temperature readings
  Serial.print("ADC Value: ");
  Serial.println(adcVal);
  Serial.print("Temperature: ");
  Serial.print(tempC);
  Serial.print("°C  ~  ");
  Serial.print(tempF);
  Serial.println("°F");
  // Alarm Activation
  if ((distance > 0 && distance < 20) || tempC > TEMP_THRESHOLD) {
    digitalWrite(BUZZER_PIN, HIGH);  // Turn buzzer on
    digitalWrite(LED_PIN, HIGH);     // Turn LED on
  } else {
    digitalWrite(BUZZER_PIN, LOW);   // Turn buzzer off
    digitalWrite(LED_PIN, LOW);      // Turn LED off
  }
  delay (2000);  // Delay for 2 seconds between readings
}
```

**Distance Measurement and Display:**

- **readDistance()**: Calls the distance function to calculate and return the current distance. If an object is detected within range, the distance is printed to the serial monitor.
- **Serial.print()**: Prints distance or "No object detected" message to the serial monitor for debugging.

**Temperature Measurement and Display:**

- **analogRead()**: Reads the analog value from the LM35 sensor, representing the temperature.
- **milliVolt**: Converts the ADC value into millivolts using the reference voltage (3300 mV) and ADC resolution (4095).
- **tempC**: Converts the millivolt reading into a temperature in Celsius (since the LM35 outputs 10 mV per degree Celsius).
- **tempF**: Converts the Celsius temperature into Fahrenheit for users who prefer this unit.

**Alarm Activation:**

- **if condition**: The buzzer and LED are activated if either of these conditions is met:
  - Distance is less than 20 cm.
  - Temperature exceeds the threshold of 40°C.
- **digitalWrite()**: Turns the buzzer and LED on if conditions are met, or turns them off if no alert is needed.

**Delay:**

- **delay (2000)**: Adds a 2-second delay between readings to prevent constant spamming of data and to allow proper functioning of the sensors.

## Conclusion

The project successfully demonstrates the integration of an ultrasonic sensor, LM35 temperature sensor, and buzzer alarm circuit with the ESP32 microcontroller. By combining these components, we have developed a functional dual-purpose alarm system capable of monitoring both distance and temperature in real-time.

The system effectively triggers visual (LED) and auditory (buzzer) alerts when:

1. The distance to an object is less than 20 cm, indicating proximity.
2. The temperature exceeds the defined threshold of 40°C, signaling a potential overheating scenario.

This implementation can be applied in various practical contexts, such as safety monitoring, environmental control, and proximity-based warning systems. The modular design and simple code structure also make it flexible for further enhancements, such as adding more sensors, sending notifications, or integrating with IoT platforms.

Satisfied Course Outcomes,

**CO 1**: Describe the architecture of embedded systems

**CO 3**: Identify the features & architecture of the ESP modules

**CO 6**: Select & use the appropriate ESP module for real world application

## References

• ESP**32 Documentation** – Official ESP32 technical reference manual for GPIO usage and ADC configuration.

- https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

• HC**-SR04 Ultrasonic Sensor Datasheet** – Provides details on the working and specifications of the ultrasonic sensor used for distance measurement.

- https://www.piborg.org/sensors1136/hcsr04#:~:text=The%20HC%2DSR04%20has%20a,to%20the%20nearest%200.3cm.

• LM**35 Temperature Sensor Datasheet** – Provides details on the working principle, output characteristics, and usage of the LM35 temperature sensor.

- https://www.ti.com/lit/ds/symlink/lm35.pdf