**CHAPTER 1**

**INTRODUCTION**

**1.1 INTRODUCTION:**

Industrial temperature control refers to the management and regulation of temperature within various industrial processes and environments. It involves the use of specialized equipment, technologies, and systems to maintain optimal temperature conditions for efficient operation, product quality, and worker safety. In industrial settings, temperature control is crucial across a wide range of applications, including manufacturing, chemical processing, food and beverage production, pharmaceuticals, energy generation, and many others. Proper temperature control ensures that industrial processes function smoothly and reliably, preventing equipment malfunctions, product defects, and safety hazards. Industrial temperature control systems typically consist of several components, including temperature sensors, controllers, actuators, and heat transfer mechanisms. These systems are designed to monitor and adjust temperatures within specific parameters, ensuring that they remain within the desired range. And we have used DHT11 for temperature and humidity sensor.

Efficient temperature control also has significant implications for energy consumption and environmental impact. By employing advanced control strategies and energy-efficient technologies, industrial facilities can reduce energy waste, minimize greenhouse gas emissions, and lower operational costs.

Industrial temperature control plays a vital role in several industries, including manufacturing, pharmaceuticals, food and beverage, petrochemicals, automotive, and many others. In manufacturing processes, precise temperature control is essential for achieving desired material properties.

**1.2 OBJECTIVES:**

* To maintaining safe temperature ranges, temperature control systems can help to prevent accidents and injuries.
* It can help to optimize the efficiency of industrial processes and reduce energy waste.

**1.3 PROBLEM STATEMENT:**

* The problem statement for industrial temperature control is that many industrial processes require precise and consistent temperature control, but achieving and maintaining this control can be challenging. Factors such as variations in ambient temperature, equipment malfunction, and human error can all affect temperature control and lead to issues such as decreased efficiency, reduced product quality, increased maintenance costs, and safety hazards.
* Therefore, the problem statement for industrial temperature control is how to design, implement, and maintain effective temperature control systems that can meet the uniquetemperature requirements of different industrial processes, respond quickly to temperature changes, and minimize the impact of external factors on temperature control.

**CHAPTER 2**

**LITERATURE REVIEW**

**2.1 OVERVIEW OF LITERATURE REVIEW:**

Control Strategies and Algorithms: Numerous studies have investigated advanced control strategies and algorithms for industrial temperature control systems. These include model predictive control (MPC), fuzzy logic control, adaptive control, and neural network-based control. Researchers have explored the effectiveness of these approaches in achieving precise temperature regulation, energy efficiency, and disturbance rejection.

Developments in sensor technologies have significantly impacted industrial temperature control. Researchers have explored the use of various sensor types, such as thermocouples, resistance temperature detectors (RTDs),thermistors, and infrared sensors, for accurate temperature measurement and monitoring. Additionally, research has focused on sensor placement optimization and fault detection techniques to ensure reliable temperature sensing.

Energy efficiency is a crucial aspect of industrial temperature control due to its impact on operational costs and environmental sustainability. Literature has addressed energy optimization techniques, including optimal scheduling, load balancing, and equipment coordination to minimize energy consumption while maintaining temperature requirements.

Heat transfer mechanisms and optimizing thermal management are critical for effective temperature control in industrial settings. Research has explored different heat transfer techniques, such as conduction, convection, and radiation, to improve heat dissipation,temperature uniformity, and thermal stability within industrial processes.

Literature has investigated the integration of renewable energy sources, such as solar and geothermal, into industrial temperature control systems. This research aims to leverage renewable energy for heating and cooling processes, thereby reducing reliance on traditional energy sources and promoting sustainability.

Literature reviews often include case studies and practical applications of industrial temperature control in specific industries. These studies highlight the challenges faced,the solutions implemented, and the outcomes achieved in areas such as food processing, pharmaceutical manufacturing, chemical reactions, and semiconductor fabrication.

**2.2 CONCLUSIONS FROM THE LITERATURE REVIEW:**

Industrial temperature control is a critical aspect of various industries, ensuring optimal process conditions, product quality, and safety. This literature review has explored the fundamentals of industrial temperature control, temperature sensors and instrumentation, control strategies and algorithms, advanced techniques, and emerging trends. Further research is needed to address challenges, such as nonlinearity and energy efficiency, and to explore the application of advanced control techniques in specific industrial processes.

Literature review is a general overview based on the information available to me. For a comprehensive and up-to-date literature review, I recommend consulting specific books and research papers on industrial temperature control.

**CHAPTER 3**

**METHODOLOGY**

**Measurement:**

The first step in temperature control is measuring the temperature of the system or process.Temperature sensors, such as thermocouples, thermistors, or RTDs (Resistance Temperature Detectors), are commonly used to measure temperature.

**Control system:**

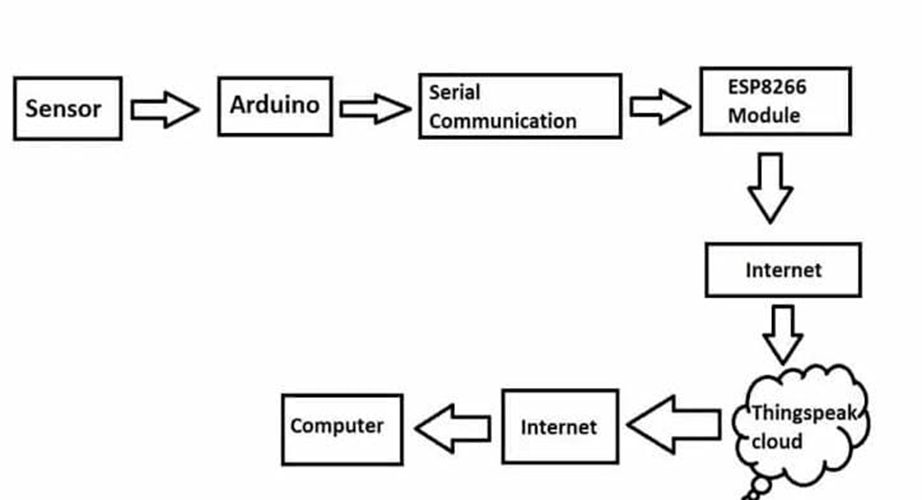
The temperature control system consists of a controller, which receives input from the temperature sensor, and an actuator, which adjusts the temperature of the system. The controller compares the measured temperature to the desired temperature and sends a signal to the actuator to adjust the temperature if necessary.

**Control algorithm:**

The control algorithm determines how the controller adjusts the actuator to maintain the desired temperature. There are several types of control algorithms, including on-off control,proportional control, and PID (Proportional-Integral-Derivative) control.

**3.1 BLOCK DIAGRAM OF THE DESIGN:**

The overall view of this project is shown in figure 3.1. The block diagram describes our project idea, Which is going to be implemented. The idea of the project is to detect the heat rate and temperature level for industrial temperature control.

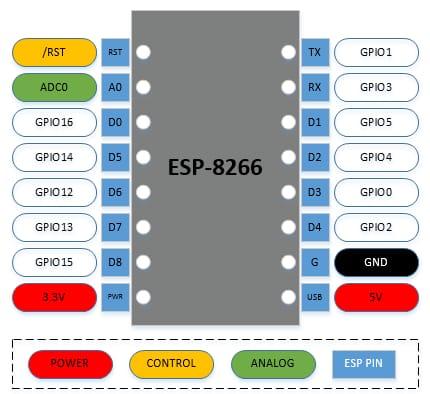
****

**Figure 3.1: Block diagram of the proposed Design**

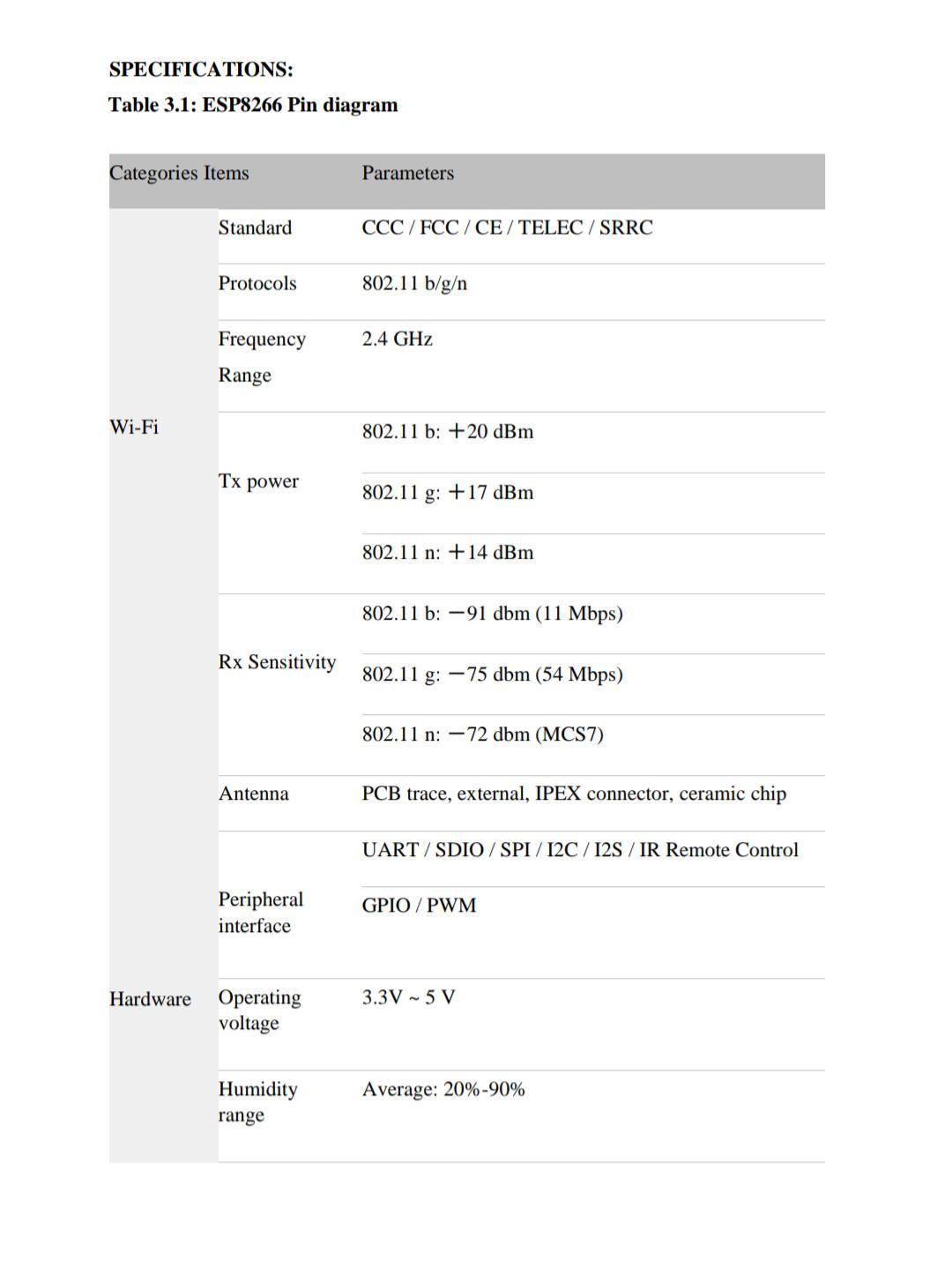
**3.2 COMPONENTS DESCRIPTION:**

**3.2.1 ESP8266**

The ESP8266 is a popular Wi-Fi module widely used in the field of Internet of Things (IoT) and embedded systems. It combines a microcontroller with built-in Wi-Fi capabilities, allowing devices to connect to the internet and communicate with other devices or cloud services, Once your development environment is set up, you can start writing code for your ESP8266. Depending on your preferred programming language, you can use either the Arduino programming language (C/C++) or the ESP-IDF framework (C/C++ or MicroPython) to develop applications for the ESP8266.Once your code is thoroughly tested and working as expected, you can deploy your ESP8266-based device in the target environment. This typically involves connecting it to the intended hardware or circuitry, ensuring power supply, and configuring any necessary settings.



**Figure 3.2: PIN DIAGRAM FOR ESP8266**





**WORKING PRINCIPLE:**

The ESP8266 is a highly popular and versatile Wi-Fi module used in various Internet of Things (IoT) applications. It combines a microcontroller unit (MCU) with built-in Wi-Fi connectivity, making it capable of connecting to the internet and interacting with other devices or services.

The ESP8266 module is powered by a 32-bit Tensilica L106 RISC microcontroller, which operates at a clock speed of 80 MHz. It has a limited amount of RAM and Flash memory for program storage, but it is usually sufficient for most IoT applications.

The module features integrated Wi-Fi functionality, allowing it to connect to wireless networks and communicate using the TCP/IP protocol stack. It supports various security standards such as WPA and WPA2 for secure communication.

The ESP8266 can be programmed using the Arduino IDE or other compatible development environments. It provides a set of GPIO (General Purpose Input/Output) pins that can be used to interface with sensors, actuators, and other electronic components. These pins can be configured as digital input or output, and some of them can handle analog signals.

When programming the ESP8266, you can write code that runs on the microcontroller to perform various tasks. For example, you can read data from sensors, control actuators, connect to Wi-Fi networks, and exchange data with remote servers or other devices.

The ESP8266 can operate in two main modes: a standalone mode and a network mode. In standalone mode, it can function as a microcontroller with Wi-Fi capabilities, running your application code locally. In network mode, it can connect to an existing Wi-Fi network and communicate with other devices over the internet.

**PIN DEFINTION:**

The ESP8266 has a large number of GPIO pins, which can be used for a variety of purposes. Here is a list of some of the most commonly used pins and their functions:

1. VCC: Power supply pin. Connect to a 3.3V power source.
2. GND: Ground pin. Connect to the ground of your power supply.
3. GPIO0: General-purpose input/output pin. It is used during boot-up for flashing new firmware. It can also be used as a digital input or output pin.
4. GPIO2: General-purpose input/output pin. It can be used as a digital input or output pin.
5. GPIO4: General-purpose input/output pin. It can be used as a digital input or output pin. GPIO5: General-purpose input/output pin. It can be used as a digital input or output pin.
6. GPIO5: General-purpose input/output pin. It can be used as a digital input or output pin.
7. GPIO12: General-purpose input/output pin. It can be used as a digital input or output pin. It can also be used as an analog input (ADC).
8. GPIO13: General-purpose input/output pin. It can be used as a digital input or output pin.
9. G PIO14: General-purpose input/output pin. It can be used as a digital input or output pin. It can also be used as an analog input (ADC).
10. GPIO15: General-purpose input/output pin. It can be used as a digital input or output pin.
11. GPIO16: General-purpose input/output pin. It can be used as a digital input or output pin.
12. RST: Reset pin. When pulled LOW, it resets the ESP8266 module.

**ADVANTAGES OF ESP8266:**

1. Cost-effective: The ESP8266 module is highly cost-effective compared to many other microcontrollers with built-in Wi-Fi capabilities.
2. Small Size: The ESP8266 module is compact in size, making it suitable for projects with space constraints or those requiring miniaturization.
3. Low Power Consumption: The ESP8266 is designed to operate with low power consumption, which is essential for battery-powered or energy-efficient IoT applications.
4. GPIO Pins and Flexibility: The ESP8266 provides a sufficient number of GPIO pins that can be configured as digital input/output, analog input, or used for serial communication (UART), I2C, and SPI.
5. Extensive Connectivity Options: In addition to Wi-Fi, the ESP8266 can be used with other communication protocols such as Bluetooth, MQTT, and HTTP, providing flexibility in connecting to different devices, platforms, and cloud services.

**POWER SUPPLY:**

**DIGITAL POWER SUPPLY:**

The ESP8266 module requires a regulated 3.3V power supply as its digital power supply. It is important to provide a stable and clean power source to ensure proper functionality and avoid potential issues.

The digital power supply voltage for the ESP8266 should be 3.3V. It is crucial not to exceed this voltage, as the module is not designed to handle higher voltages. Applying higher voltage can damage the ESP8266.

**POWER-ON SEQUENCE:**

Connect a regulated 3.3V power supply to the VCC and GND pins of the ESP8266 module. It is important to provide a stable and clean power source within the specified voltage range.

**RESET:**

If you want to ensure a clean boot and avoid any residual states from previous operations, you can briefly set the RST (Reset) pin of the ESP8266 module to a LOW state. This can be done by connecting the RST pin to the GND (Ground) pin and then disconnecting it.

**3.2.2 DHT11**

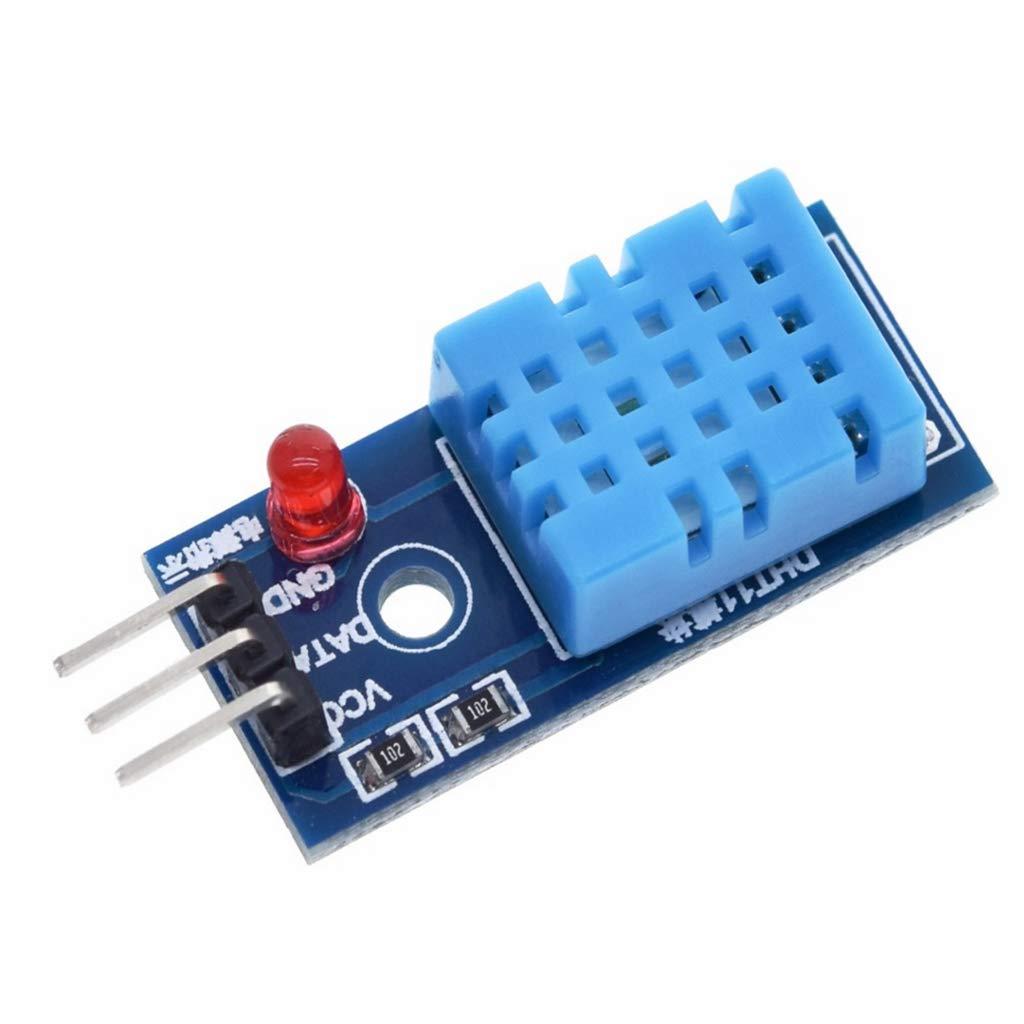
The DHT11 sensor consists of a capacitive humidity sensor and a thermistor for temperature sensing. These sensing elements are embedded within the sensor module.

The DHT11 uses a single-wire communication protocol called the 1-Wire protocol. This means that it requires only one digital pin to communicate with a microcontroller or other devices.

To start the communication with the DHT11, the microcontroller sends a start signal to the sensor. The start signal consists of a low-voltage signal for a specific duration to initiate the data transmission.

After receiving the start signal, the DHT11 sensor sends data to the microcontroller in the form of a series of pulses. Each data bit is represented by the duration of a high signal followed by a low signal. The duration of the high signal determines whether it represents a "0" or a "1" bit.

The DHT11 sensor has specific timing requirements for signal durations and delays during the communication process. The microcontroller must adhere to these timing requirements to ensure accurate data transmission and reception.



**Figure 3.3: DHT11**

**WORKING OF DHT11:**

The DHT11 sensor consists of a capacitive humidity sensor and a thermistor for temperature sensing. These sensing elements are embedded within the sensor module.

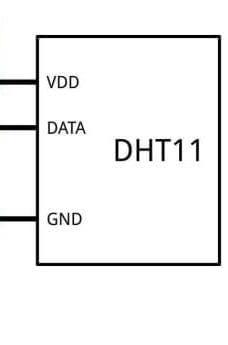
The DHT11 uses a single-wire communication protocol called the 1-Wire protocol. This means that it requires only one digital pin to communicate with a microcontroller or other devices.

To start the communication with the DHT11, the microcontroller sends a start signal to the sensor. The start signal consists of a low-voltage signal for a specific duration to initiate the data transmission.

After receiving the start signal, the DHT11 sensor sends data to the microcontroller in the form of a series of pulses. Each data bit is represented by the duration of a high signal followed by a low signal. The duration of the high signal determines whether it represents a "0" or a "1" bit.

The DHT11 sensor has specific timing requirements for signal durations and delays during the communication process. The microcontroller must adhere to these timing requirements to ensure accurate data transmission and reception.**PIN DIAGRAM FOR DHT11:**

1. This component consists of a thermistor that measures the ambient temperature.
2. Analog-to-Digital Converter: The analog output from the temperature sensing element is converted into a digital signal that can be processed by the microcontroller.
3. Data Encoding and Transmission: This component encodes the temperature and humidity data into a specific format suitable for transmission over a single-wire digital interface.
4. Microcontroller Interface: The encoded data is received and processed by a microcontroller.
5. Output Pins: The microcontroller provides output pins to transmit the temperature and humidity readings to external devices or the user interface of the project.

****

**Figure 3.4: Pin Diagram for DHT11**

**PIN DEFINITION FOR DHT11:**

1. VCC: This pin is the power supply pin and should be connected to a 5V power source.
2. Data: The data pin is used for bidirectional communication between the DHT11 sensor and the microcontroller or other devices.
3. NC (No Connect) or Not Used: Some DHT11 modules may have a third pin labeled as NC, which stands for No Connect. This pin is not used and is left unconnected in the circuit.
4. GND: The GND pin is the ground pin and should be connected to the ground of the power supply or the common ground of the system.

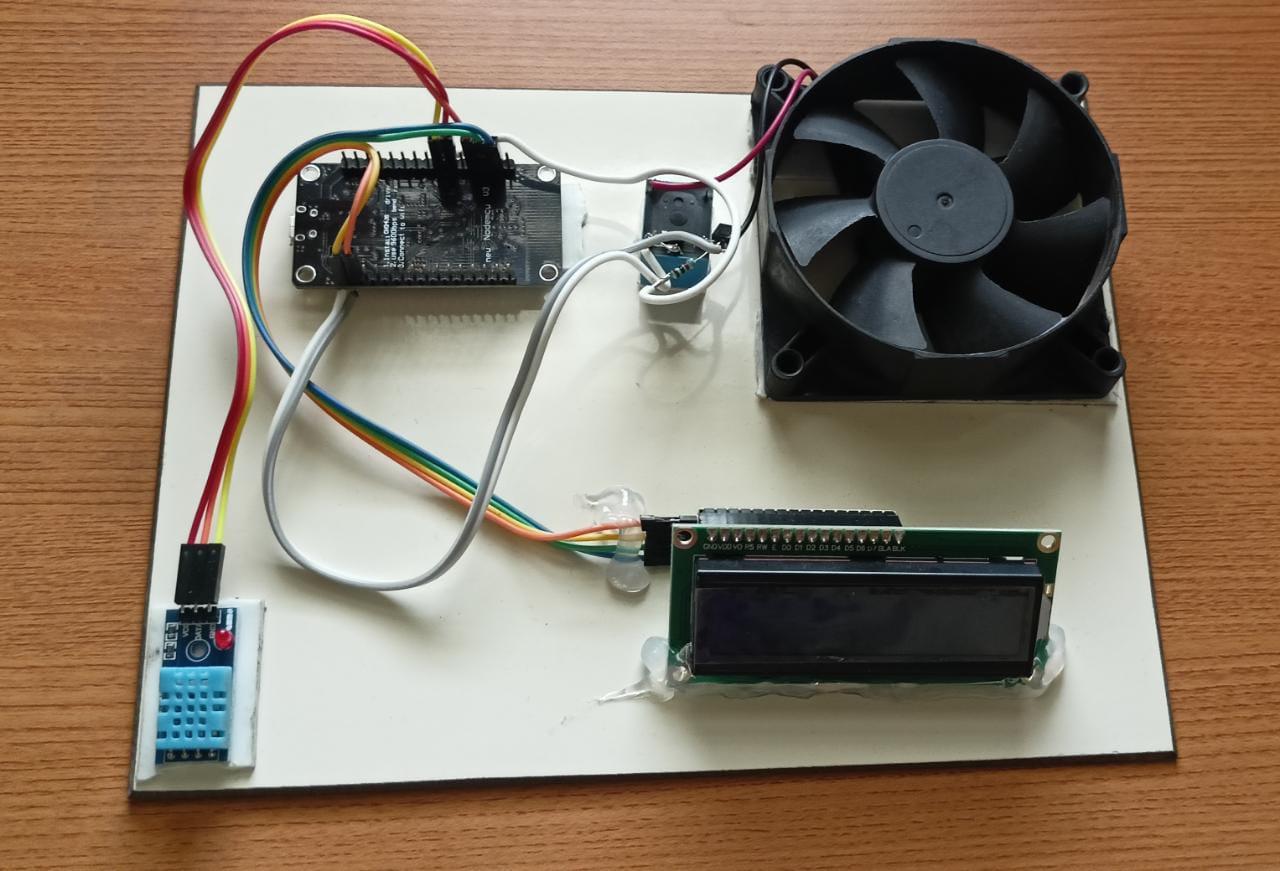
**ADVANTAGES FOR DHT11:**

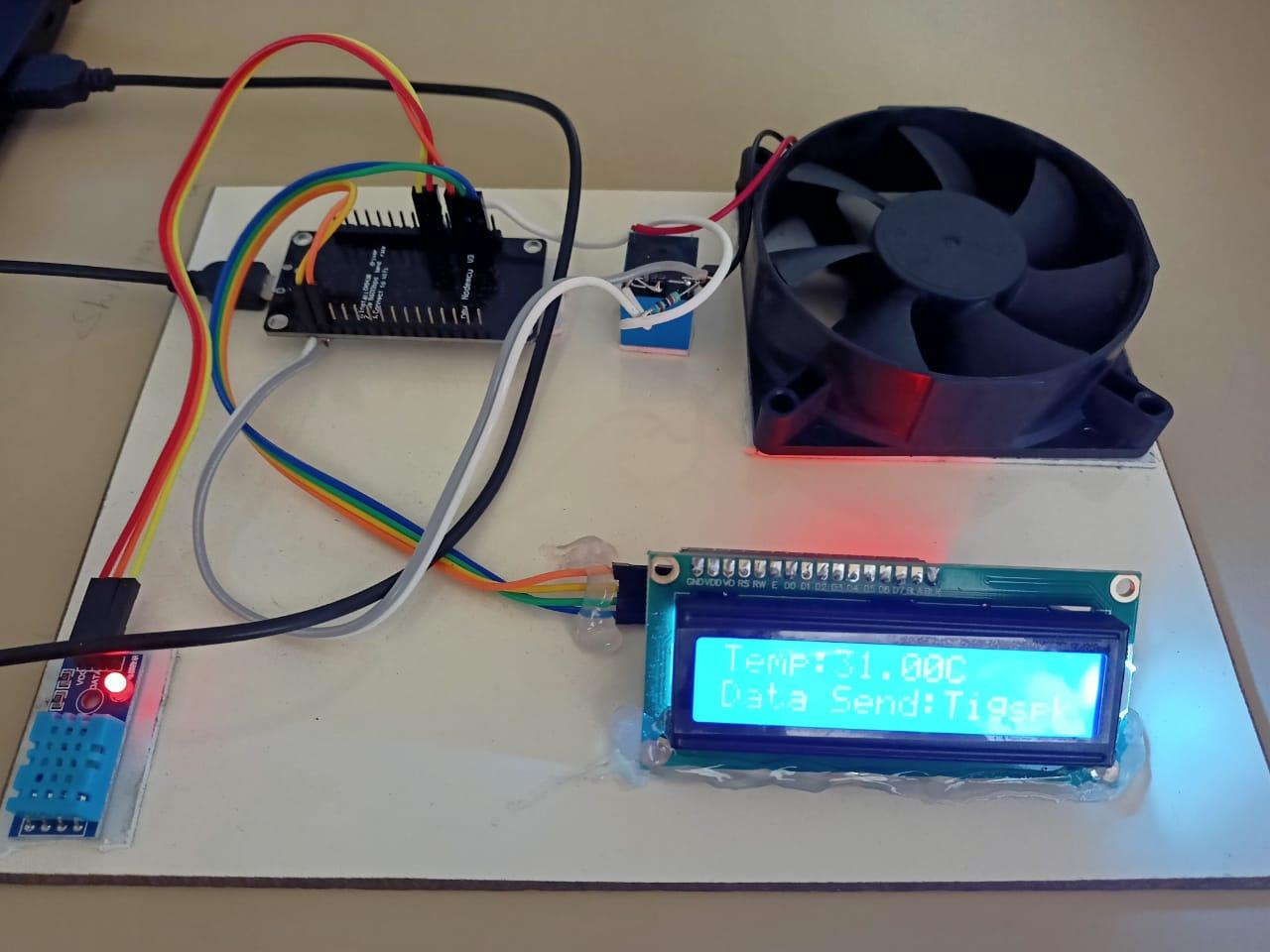
1. Cost-Effective: The DHT11 sensor is highly affordable and readily available at a low cost.
2. Ease of Use: The DHT11 sensor is straightforward to use, even for beginners. It has a simple digital interface, requiring only a single data pin for communication with a microcontroller or other devices.
3. Wide Operating Voltage Range: The DHT11 sensor can operate within a wide voltage range, typically from 3.3V to 5V.
4. Adequate Accuracy for Basic Applications: While the DHT11 sensor may not provide the highest level of accuracy compared to more advanced and expensive sensors
5. Low Power Consumption: The DHT11 sensor operates with low power consumption, making it suitable for battery-powered or energy-efficient applications.
6. Adequate Measurement Range: The DHT11 sensor offers a suitable measurement range for many applications. It can measure temperature within a range of 0°C to 50°C (32°F to 122°F) with a resolution of 1°C and humidity within a range of 20% to 80% with a resolution of 1%.

**CHAPTER 4**

**EXPERIMENTAL SETUP AND PROCEDURE**

**4.1 EXPERIMENTAL SETUP:**



blob:https://web.whatsapp.com/2079ef0d-ffb2-4a07-bd17-698006029f63 **blob:https://web.whatsapp.com/12555140-14e3-4f98-9ed1-aef57f08c916** 

**Figure 4.1: Hardware Setup**

Figure 4.1 represents the hardware setup of the proposed project. In this hardware setup, the power supply is obtained from the laptop and passed to the circuit which contains an ESP8266 and DHT11. This is the hardware setup used to detect the industrial temperature.

**4.2 ARDUINO:**

This program uses the Adafruit\_MAX31855 library for reading temperature from a thermocouple sensor. Make sure you have the library installed in your Arduino IDE before uploading the program to your Arduino board.

The program starts by defining the connections for the thermocouple sensor (CLK, CS, DO) and the relay (RELAY\_PIN). It also sets the temperature thresholds for controlling the relay.

In the setup() function, it initializes the serial communication for debugging purposes and sets the relay pin as an output.

The main loop() function reads the temperature from the thermocouple using thermocouple.readCelsius(). It then prints the temperature to the serial monitor.

Next, it checks the temperature against the defined thresholds. If the temperature is below the lower threshold, it turns on the relay by setting the relay pin to HIGH. If the temperature is above the upper threshold, it turns off the relay by setting the relay pin to LOW.

**4.3 THINGSPEAK CLOUD PLATFORM:**

ThingSpeak provides APIs and libraries that enable users to send data from their IoT devices or sensors to the cloud platform. It supports various communication protocols such as HTTP, MQTT, and ThingSpeak's own ThingSpeak IoT protocol.

ThingSpeak organizes data into channels, which are essentially virtual containers for data streams. Each channel consists of multiple fields that represent different data points or variables. Users can define their own fields based on their specific requirements.

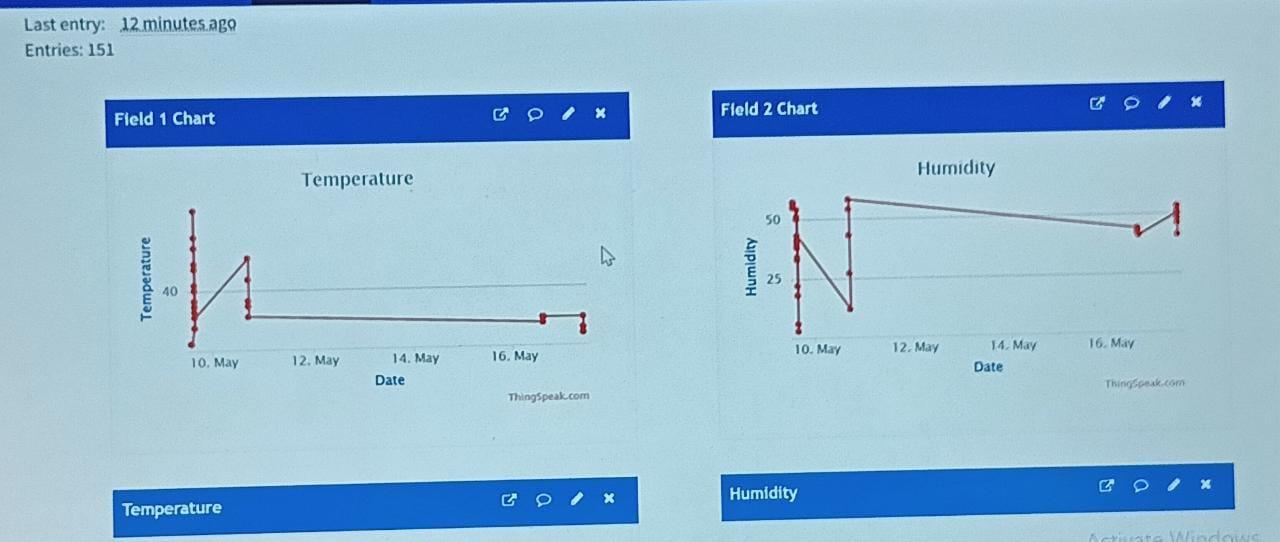
ThingSpeak stores the data sent to it in a time-series format, allowing users to store and access historical data. Data is automatically timestamped upon arrival, and users can retrieve specific data points based on time intervals or other criteria.

ThingSpeak offers built-in tools for visualizing data in real-time. Users can create customizable charts, gauges, and maps to display their data. The platform also supports MATLAB visualizations, enabling advanced data analysis and visualization capabilities.

**CHAPTER 5**

**RESULT AND DISCUSSION**

* 1. **RESULT:**



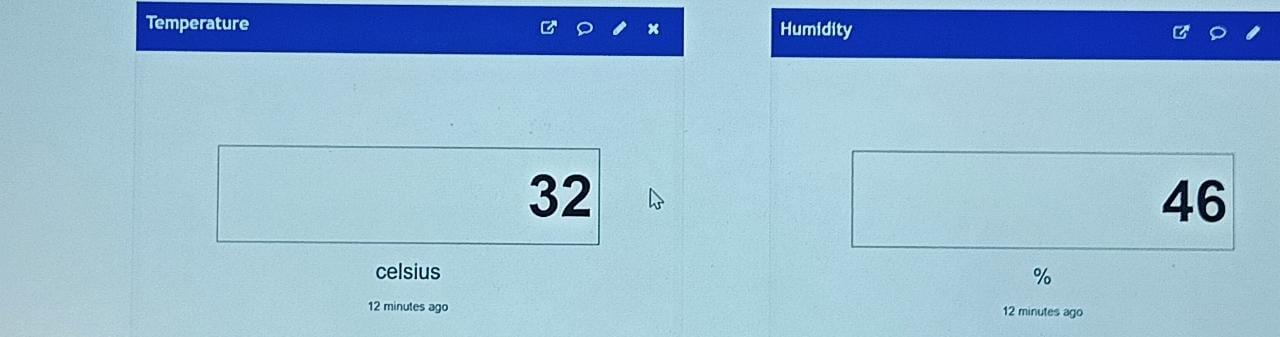


Figure 5.1: SOFTWARE OUTPUT



Figure 5.2: HARDWARE OUTPUT

### 5.2 OUTPUT DESCRIPTION

From this output, we can determine the temperture of the machine in the current scenario. In this project, we used an DHT11 to capture the temperature level, and processed it using processed it through Embedded C to check its temperature level based on celcius and humidity . Our stored temperature level contain data of the heat and cooling, which is used as a comparison tool to determine the industrial temperature using the embedded C Code. Therefore, we can see the temperature level of the industrial machines.

**CHAPTER 6**

### CONCLUSION

In conclusion, industrial temperature control plays a vital role in ensuring optimal operating conditions, product quality, and safety in various industrial processes. This literature review has provided insights into the fundamentals, control strategies, advanced techniques, challenges, and real-world applications of industrial temperature control.

The review highlighted the basic principles of temperature measurement and control, including different types of temperature sensors and the widely used PID control theory. It also discussed various control strategies, such as on-off control, proportional control, PID control, and model predictive control (MPC). These strategies offer different levels of sophistication and performance, allowing for precise temperature regulation in industrial settings.

The challenges associated with industrial temperature control were discussed, including nonlinearity, sensor calibration, robust system design, external disturbances, thermal inertia, and energy efficiency considerations. Addressing these challenges requires further research and development efforts to enhance control system performance and optimize energy usage.

Finally, the review presented real-world applications of industrial temperature control, ranging from chemical and petrochemical industries to food processing, power generation, HVAC systems, industrial furnaces, and semiconductor manufacturing. Each application has its unique requirements and constraints, demanding tailored temperature control solutions.

**APPENDIX 1:**

#include <stdio.h>

#include <stdbool.h>

// Function to read temperature from sensor

float readTemperature() {

// Code to read temperature from sensor

// Replace this with your actual sensor reading code

// For example, you might use analog-to-digital conversion or a communication protocol

// Return the temperature value as a float

return 0;

}

// Function to control the fan based on the temperature

void controlFan(float temperature) {

static bool fanState = false; // Stores the current state of the fan

// Check if temperature is increasing by 35 degrees

if (temperature >= 35.0) {

fanState = true; // Turn on the fan

} // Check if temperature is decreasing below 30 degrees

if (temperature <= 30.0) {

fanState = true; // Turn on the fan to maintain temperature

}

// Code to control the fan

// Replace this with your actual fan control code

// For example, you might use digital outputs or a communication protocol

// Implement the logic to turn on or off the fan based on the fanState variable }

int main() {

// Variable to store the desired temperature

float desiredTemperature = 25.0;

// Main loop

while (true) {

// Read the temperature

float currentTemperature = readTemperature();

// Print the current temperature

printf("Current Temperature: %.2f\n", currentTemperature);

// Control the fan based on the temperature

controlFan(currentTemperature);

// Check if the current temperature is within an acceptable range

if (currentTemperature >= desiredTemperature - 1.0 && currentTemperature <= desiredTemperature + 1.0) {

// Temperature is within an acceptable range

// Implement any necessary actions or alerts here }

// Delay between temperature readings and fan control

// Adjust the delay based on your system requirements

// This is just an example delay for demonstration purposes

for (int i = 0; i < 10000000; i++) {

// Do nothing, just waste some time

} }

return 0;

}

**APPENDIX 2:**

#include <ThingSpeak.h>

#include <WiFi.h>

// WiFi credentials

const char\* ssid = "your\_wifi\_ssid";

const char\* password = "your\_wifi\_password";

// ThingSpeak settings

const char\* server = "api.thingspeak.com";

const char\* apiKey = "your\_thingspeak\_api\_key";

const unsigned long channelID = your\_channel\_id;

// Thermocouple connections

#define THERMOCOUPLE\_CLK 13

#define THERMOCOUPLE\_CS 10

#define THERMOCOUPLE\_DO 12

// Relay connection

#define RELAY\_PIN 9

// Temperature thresholds

#define LOWER\_THRESHOLD 50.0

#define UPPER\_THRESHOLD 100.0

WiFiClient client;

Adafruit\_MAX31855 thermocouple(THERMOCOUPLE\_CLK, THERMOCOUPLE\_CS, THERMOCOUPLE\_DO);

void setup() {

// Initialize serial communication

Serial.begin(115200);

// Connect to Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi...");

}

Serial.println("Connected to WiFi");

// Initialize ThingSpeak

ThingSpeak.begin(client);

}

void loop() {

// Read temperature from thermocouple

double temperature = thermocouple.readCelsius();

// Print temperature to serial monitor

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.println(" °C");

// Send temperature data to ThingSpeak

ThingSpeak.setField(1, temperature);

int httpCode = ThingSpeak.writeFields(channelID, apiKey);

if (httpCode == 200) {

Serial.println("Data sent to ThingSpeak successfully");

} else {

Serial.println("Error sending data to ThingSpeak"); }

**REFERENCE**

<https://www.engineersgarage.com/industrial-temperature-control-system/>

<https://www.ia.omron.com/support/guide/53/introduction.html>

<https://www.renesas.com/us/en/application/industrial/factory-automation/monitoring-control/precision-industrial-temperature-control>

<https://thermalprocessing.com/the-plc-based-industrial-temperature-control-system-design-and-implementation/>

<https://www.researchgate.net/publication/314781614_The_PLC-based_Industrial_Temperature_Control_System_Design_and_Implementation>

<https://www.slideshare.net/Edgefx/industrial-temperature-controller-system>