A Mini-Project II Report

On

TEMPERATURE BASED FAN SPEED CONTROL USING ARDUINO

Submitted By:

ANSHIKA RAJENDRA PATIL,

KAJAL PRAMOD PATIL

And

GAURAV ANIL DESALE

Of

Third Yr. B. Tech. (Electronics & Telecommunication Engineering) Semester-VI

Guided By:

Prof. Mahesh H. Patil

As per syllabus laid by

Dr. Babasaheb Ambedkar Technological University, Lonere – Raigad

Godavari Foundation's

GODAVARI COLLEGE OF ENGINEERING, JALGAON

Yr. 2024-2025

Dr. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY, LONERE

CERTIFICATE

This is to certify that KAJAL PRAMOD PATIL (PRN No.-2251711372018) and GAURAV ANIL DESALE (PRN NO.-2251711372007) of Third Year B. Tech. (Electronics & Telecommunication Engineering) have satisfactorily completed the Mini Project-II Report entitled

Temperature Based Fan Speed Controller

as per laid down in the syllabus of Dr. Babasaheb Ambedkar Technological University, Lonere - Raigad and this report includes her bonafide work in the B. Tech., sixth semester, in the academic year 2024-25.

Prof. Mahesh H. Patil
Guide

Dr. Hemant Ingale
Head, E&TC Engg Dept.

Dr. Vijaykumar
PRINCIPAL

Godavari Foundation's
GODAVARI COLLEGE OF ENGINEERING, JALGAON

ACKNOWLEDGEMENT

I am immensely thankful to my Teachers at Godavari College of Engineering, Jalgaon for teaching me the importance of self-learning & essence of keeping up with emerging technology during Third year B. Tech. course in Electronics & Telecommunication Engg. I wish to express my thanks towards **Prof. Mahesh H. Patil**, my guide for providing the technical guidelines and suggestions regarding the line of this work. He encourages me all the times for doing the said work in tuned with the target puts before me. He has provided excellent suggestions while carefully reviewing my Mini Project-I report. I have been very fortunate to have him as my guide and his appreciable guidance during my stay at the department**.**

I would also like to thank **Dr. Hemant. Ingale**, Head, E&TC Engineering department for his constant encouragement towards technical improvements and appreciation of my efforts. It's a moment of immense satisfaction for me to express my sense of profound gratitude towards my Principal **Dr. Vijaykumar Patil** and Vice Principal **Prof. P. V. Phalak**, whose zeal & enthusiasm were a source of inspiration for us.

I also appreciate the support given by all the staff members of E & TC Department without which it was impossible to complete this work. Last but not least thanks to my friends those who rendered their precious help during the completion of this work.

Place: JALGAON

Miss. Anshika Rajendra Patil

PRN No. 2251711372003

Miss. Kajal Pramod Patil

PRN No. 2251171372018

Mr. Gaurav Anil Desale

PRN No. 2251711372007

ABSTRACT

The **Temperature-Based Fan Speed Control** system is designed to automatically adjust the speed of a fan based on the ambient temperature to ensure optimal cooling and energy efficiency. By utilizing a temperature sensor, such as the **DHT11**, the system continuously monitors the surrounding temperature. The data from the sensor is fed to a microcontroller, like **Arduino**, which processes the temperature readings and adjusts the fan speed accordingly. The fan operates at a lower speed when the temperature is within a comfortable range and increases speed as the temperature rises, providing efficient cooling when necessary.

This approach ensures that the fan operates only when needed, conserving energy and reducing noise levels. The system can be integrated into various applications, including cooling systems for electronic devices, computers, or even home ventilation systems, to maintain an optimal temperature while minimizing power consumption. The integration of components like motor drivers, a microcontroller, and temperature sensors makes the system highly adaptable and scalable for different cooling needs.

INDEX

NAME OF FIGURES

1. Introduction

This project aims to design and implement a temperature-based fan speed control system that automatically adjusts the speed of a fan according to the ambient temperature. The system utilizes an **Arduino UNO** microcontroller along with a variety of sensors and actuators to monitor temperature and adjust fan speed accordingly. The system's primary goal is to maintain an optimal cooling environment by providing efficient fan control based on real-time temperature data. Such a system can be beneficial in various applications, including:

- **Cooling electronic devices** like computers, servers, and power supplies, where overheating can lead to performance degradation or hardware failure.
- **Indoor climate control**, especially in smart homes or data centers, where maintaining a stable temperature is crucial for comfort and energy efficiency.

The system will display the current temperature on an **LCD** and will dynamically adjust the fan speed to cool the environment when the temperature crosses a certain threshold.

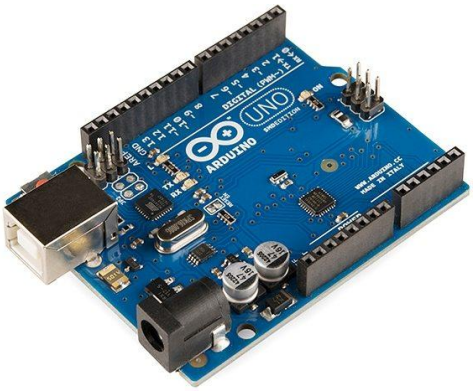
1.1 Objective:

The objective of this project is to design a fully functional temperature control system that uses the **Arduino UNO** to monitor the temperature from a **DHT11 sensor** and adjust the speed of a **12V exhaust fan** accordingly. The fan speed will be regulated using a **L298 motor driver**, which is capable of controlling the fan's speed through Pulse Width Modulation (PWM). The system will be equipped with an **LCD with I2C module** to display the real-time temperature and the fan's status.

1.2 Components:

1. **Arduino UNO**:

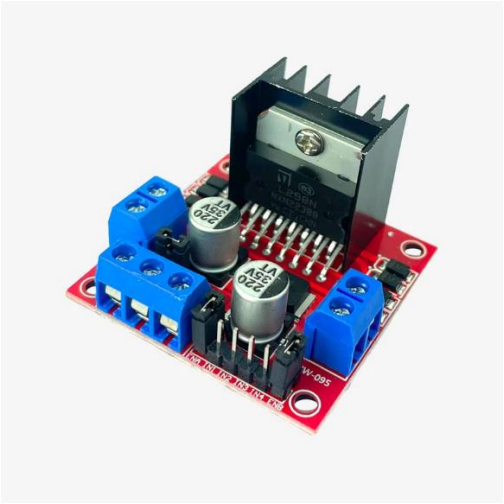
   The **Arduino UNO** acts as the central control unit in this project. It reads temperature data from the **DHT11 sensor** and processes the data to determine the appropriate speed for the fan. Based on predefined temperature thresholds, the Arduino sends PWM signals to the **L298 motor driver**, controlling the fan speed accordingly. The Arduino also handles the interface with the **LCD**, updating it with the current temperature and fan status.



(Fig. 1.2.1 Arduino UNO)

2. **L298 Motor Driver**:

The **L298 Motor Driver** is a dual H-Bridge motor driver that allows the Arduino to control the speed and direction of the **12V exhaust fan**. By using Pulse Width Modulation (PWM) signals, the L298 regulates the voltage supplied to the fan, thereby controlling its speed. The **L298** is powered by a **12V adapter** to supply the necessary voltage to the fan.

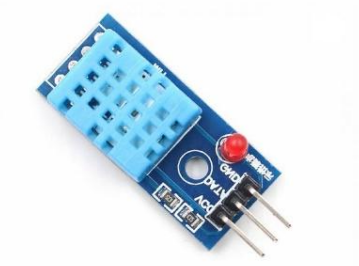
- 

(Fig. 1.2.2 L298 Motor Driver)

3. **DHT11 Sensor**:

The **DHT11** sensor is a low-cost temperature and humidity sensor that provides accurate readings of ambient temperature. The sensor communicates with the

Arduino using a digital signal, allowing the system to measure the temperature and decide the required fan speed. The sensor has a working range of 20-50°C for temperature and 40-80% for humidity, making it suitable for our purpose of controlling fan speed based on temperature changes.



(Fig. 1.2.3 DHT11 Sensor)

4. **12V Exhaust Fan**:

The 12V exhaust fan is used to cool the environment. It is powered through the L298 motor driver, which adjusts its speed based on the temperature read by the DHT11 sensor. The fan operates within a 12V power supply, making it ideal for applications that require moderate cooling without complex setups.

(Fig. 1.2.4 12v Exhaust Fan)

5. **LCD with I2C Module**:

   The **LCD with I2C module** is used to display real-time information about the temperature and the current fan speed. The **I2C module** simplifies wiring by using only two data lines (SDA and SCL) for communication, making it an efficient choice for projects that require an easy interface with displays. The LCD screen shows the current temperature, fan speed, and any status updates or error messages.

(Fig. 1.2.5 LCD with I2c Module)

6. **12V Adapter**:

   The **12V adapter** provides the required power for the **12V exhaust fan** and the **L298 motor driver**. It ensures that the fan has the necessary voltage to operate efficiently and that the system can handle the load without overloading the components.

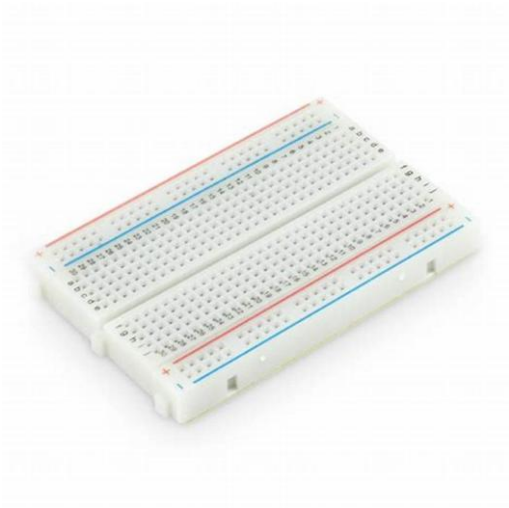
7. **Female DC Connector**:

   The **female DC connector** is used to connect the **12V adapter** to the power supply input on the system. It ensures secure and stable power delivery to the fan and motor driver.

(Fig. 1.2.6 12v Adapter and Female DC Connector)

8. **Breadboard and Jumper Wires**:

Breadboard and **jumper wires** are used for prototyping and connecting all the components without the need for soldering. The breadboard provides a temporary connection platform, allowing easy testing and modification of the circuit before final assembly.



(Fig. 1.2.7 Breadboard)

2. Circuit Design

1. **DHT11 Sensor**:
    - **VCC** → 5V pin on the Arduino.
    - **GND** → GND pin on the Arduino.
    - **DATA** → Digital pin 7 on the Arduino (you can choose another pin if necessary).
2. **L298 Motor Driver** (to control the 12V exhaust fan):
    - **ENA** → 5V (this enables the motor driver).
    - **IN1** → Arduino pin 9.
    - **IN2** → Arduino pin 10.
    - **OUT1** → One terminal of the 12V DC fan.
    - **OUT2** → Other terminal of the 12V DC fan.
    - **12V Input** → 12V adapter.
    - **GND** → Common ground connected to the GND of the Arduino and the power supply.
3. **LCD with I2C Module**:
    - **SDA** → A4 pin on the Arduino UNO.
    - **SCL** → A5 pin on the Arduino UNO.
    - **VCC** → 5V pin on the Arduino.
    - **GND** → GND pin on the Arduino.

4. **12V Adapter**:
    - **Positive (12V)** → Connected to the **12V input** pin of the L298 motor driver.
    - **Ground (GND)** → Connected to the GND of the Arduino and the L298 motor driver.

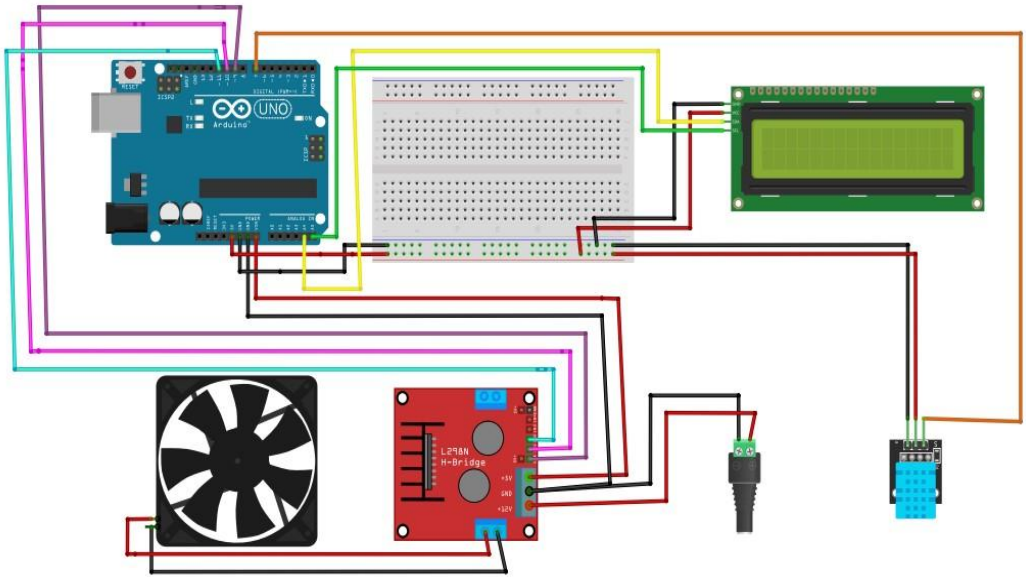
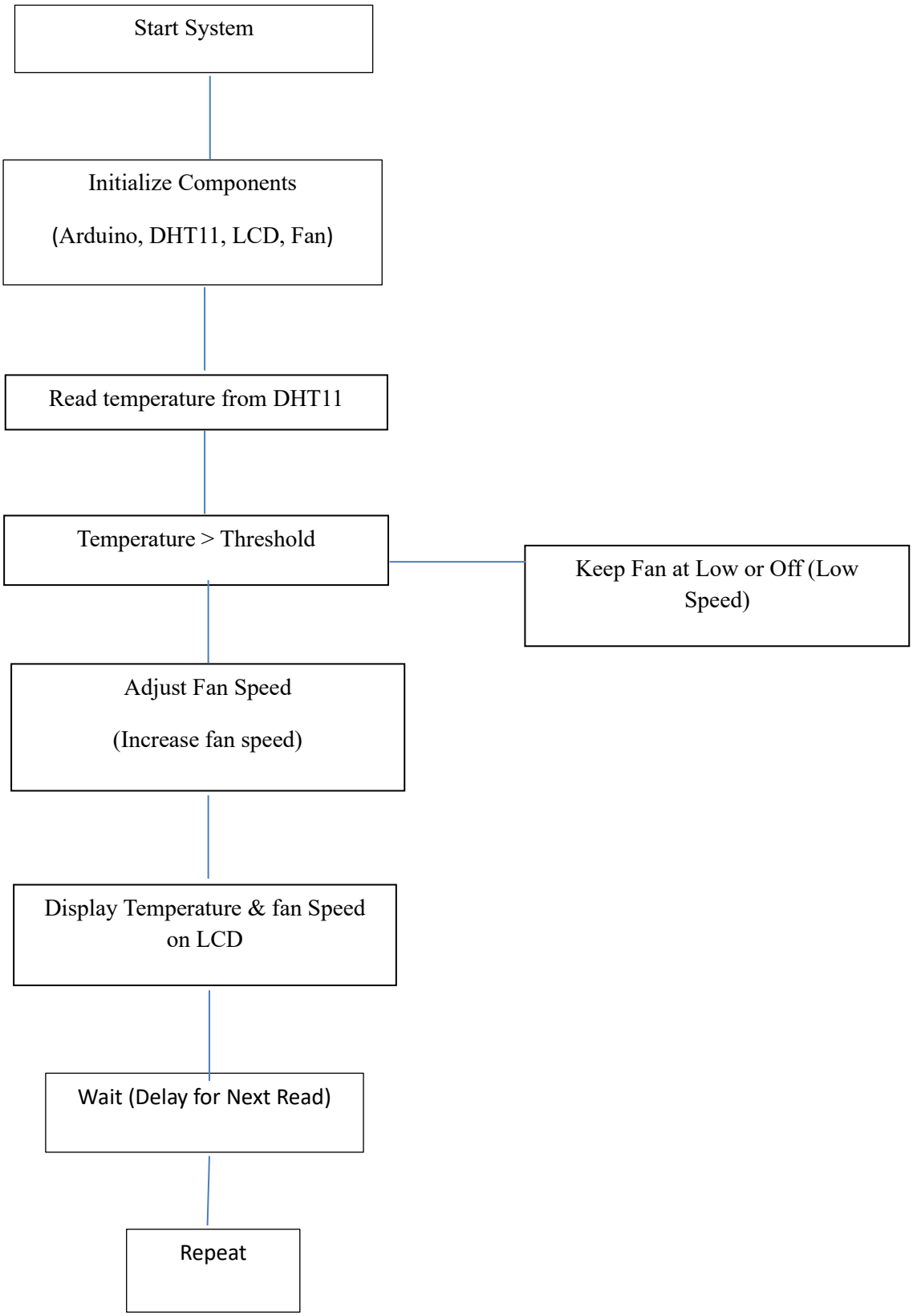


Fig. 2.1 Circuit Diagram

3. Flowchart


Start System
Initialize Components
(Arduino, DHT11, LCD, Fan)
Read temperature from DHT11
Temperature > Threshold
Keep Fan at Low or Off (Low Speed)
Adjust Fan Speed
(Increase fan speed)
Display Temperature & fan Speed on LCD
Wait (Delay for Next Read)
Repeat

4. Working Principle

4.1 Temperature Sensing (DHT11 Sensor)

The **DHT11 sensor** plays a crucial role in this system by measuring the **ambient temperature**. The sensor provides digital output, allowing the **Arduino UNO** to read the temperature data via a single digital pin. Here's a step-by-step explanation of how the temperature sensing works:

1. **Sensing the Temperature**:
    - The DHT11 sensor continuously monitors the surrounding temperature and outputs a signal representing the current temperature in Celsius.
    - The **DHT11 sensor** is powered by the **5V** pin from the Arduino and transmits the data to a specified digital pin (e.g., **pin 7**) via a single wire.
2. **Data Transmission to Arduino**:
    - The Arduino reads the **digital signal** sent from the DHT11 sensor. Using an external library (such as the **DHT** library), the Arduino can easily decode the data and extract the temperature reading.
    - The DHT11 sensor has a limited range of **20°C to 50°C** with an accuracy of ±2°C, which is suitable for most temperature-sensitive cooling applications.
3. **Processing the Data**:
    - The Arduino processes the temperature data, which can be read as a numerical value (e.g., 25°C), and uses this information to determine the fan's speed.
4. **Decision Making**:
    - The Arduino compares the current temperature with **predefined thresholds** to decide whether the fan should run at full speed, partial speed, or be turned off. For example:
        - **If the temperature is below 25°C**, the fan may be turned off or run at low speed.

- **If the temperature is above 30°C**, the fan may run at higher speeds to cool the environment.

4.2 Fan Speed Control (Using L298 Motor Driver)

The **L298 Motor Driver** controls the **12V exhaust fan**, adjusting its speed according to the temperature reading provided by the **DHT11 sensor**. The **L298** uses **Pulse Width Modulation (PWM)** to adjust the voltage and current supplied to the fan, thus controlling its speed.

Here's how the fan speed control mechanism works:

1. **PWM Signal Generation**:
    - Based on the temperature data, the **Arduino UNO** generates a **PWM signal** on one of its digital output pins (e.g., **pin 9** for controlling the motor speed via the L298).
    - The **PWM** signal is a digital square wave, where the duty cycle (the proportion of the time the signal is HIGH) determines the effective voltage supplied to the fan.
        - **A higher duty cycle** (e.g., 100%) sends a stronger signal, increasing the fan speed.
        - **A lower duty cycle** (e.g., 0%) sends a weaker signal, reducing the fan speed.
2. **Controlling Fan Speed**:
    - The **L298 Motor Driver** receives the PWM signal from the Arduino on the **ENA pin** and adjusts the output voltage accordingly.
    - The **IN1** and **IN2** pins control the direction of the fan, ensuring it runs in the correct direction (i.e., exhausting air out of the system).
    - The **PWM signal** sent from the Arduino will control the fan's speed, increasing or decreasing based on the ambient temperature.

3. **Temperature vs Fan Speed**:
    - The Arduino uses the **map()** function to map the temperature to a suitable PWM value. For example:
        - **Temperature 25°C → PWM 0%** (Fan off or slow speed)
        - **Temperature 30°C → PWM 50%** (Moderate fan speed)
        - **Temperature 35°C → PWM 100%** (Full fan speed)

4.3 LCD Display (Real-Time Feedback)

The **LCD with I2C** module provides real-time feedback to the user about the current temperature and the fan's operating status. Here's how the LCD plays a role:

1. **Displaying the Temperature**:
    - The Arduino reads the temperature data from the DHT11 sensor and formats it into a string (e.g., "Temp: 25°C").
    - This data is displayed on the **LCD screen** (16x2 character display) in real-time. The **I2C module** simplifies the wiring by using only two data lines (SDA and SCL), allowing communication with the Arduino.
2. **Fan Speed Status**:
    - The Arduino calculates the fan speed based on the temperature and the PWM signal being sent to the fan.
    - The current **fan speed** or the **status (e.g., "Fan On" or "Fan Off")** is displayed on the LCD screen to provide users with feedback on the system's operation.

For example:

- **If the fan is off** (at a temperature below a threshold), the display will show something like:
  Temp: 25°C, Fan: Off

- **If the fan is running at moderate speed** (e.g., at 30°C), the display might show:
  Temp: 30°C, Fan: Medium

3. **Continuous Updates**:
   - The **LCD** is updated at regular intervals (e.g., every 2 seconds) so that the user can always see the current temperature and fan status.
   - This provides the user with an ongoing real-time update on the system's operation, making it easy to monitor the cooling process.

4.4 Flow of the System:

1. **Sensing the Temperature:**

   The **DHT11 sensor** measures the ambient temperature and sends the data to the **Arduino UNO**.

2. **Processing the data:**
   The **Arduino** processes the data and compares the temperature against predefined thresholds to determine the appropriate fan speed.

3. **Controlling the Fan:**

   Based on the temperature, the Arduino sends a **PWM signal** to the **L298 motor driver** to adjust the **12V exhaust fan's speed**. The fan's speed is proportionate to the temperature.

4. **Displaying Information:**
   The **LCD with I2C module** updates every 2 seconds to display the **current temperature** and **fan speed** to the user.

5. **Automatic Fan Control**

   As the temperature rises or falls, the fan speed changes automatically to maintain an optimal cooling environment.

5. Source Code

```
#include <Adafruit_Sensor.h>

#include <DHT.h>

#include <DHT_U.h>

#include <Wire.h>

#include <LiquidCrystal_I2C.h>

#define DHTPIN 7         // DHT11 sensor data pin

#define DHTTYPE DHT11    // DHT sensor type

#define MOTOR_PIN_ENA 9  // Enable pin for motor driver (PWM pin)

#define MOTOR_PIN_IN1 10  // Input pin 1 for motor driver

#define MOTOR_PIN_IN2 11  // Input pin 2 for motor driver

#define TEMPERATURE_THRESHOLD 29

#define TEMPERATURE_THRESHOLD1 32 // Temperature threshold to adjust motor speed

DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal_I2C lcd(0x27, 16, 2);  // Adjust the I2C address if necessary

void setup() {

  Serial.begin(9600);

  dht.begin();

 Wire.begin();

  lcd.begin();  // Initialize the LCD

  lcd.backlight();  // Turn on the backlight

  lcd.setCursor(0, 0);
```

```
  lcd.print("Temperature:");
}
void loop() {
  delay(2000); // Delay between readings (adjust as needed)
  float temperature = dht.readTemperature(); // Read temperature in Celsius
  if (isnan(temperature)) {
    Serial.println("Failed to read temperature from DHT sensor!");
    return;
  }
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");
  lcd.setCursor(0, 1); // Set cursor to the second line
  lcd.print("                "); // Clear the previous motor speed
  // Adjust motor speed based on temperature
  if (temperature > TEMPERATURE_THRESHOLD1) {
    // Increase motor speed
    analogWrite(MOTOR_PIN_ENA, 255);    // Set the motor speed to maximum (255)
    digitalWrite(MOTOR_PIN_IN1, HIGH);  // Set motor direction (forward)
    digitalWrite(MOTOR_PIN_IN2, LOW);
    lcd.setCursor(0, 1);
    lcd.print("Motor Speed: Max");
  }
```

```
  else if (temperature > TEMPERATURE_THRESHOLD) {
    // Increase motor speed
    analogWrite(MOTOR_PIN_ENA, 100);    // Set the motor speed to a value (100)
    digitalWrite(MOTOR_PIN_IN1, HIGH);  // Set motor direction (forward)
    digitalWrite(MOTOR_PIN_IN2, LOW);
    lcd.setCursor(0, 1);
    lcd.print("Motor Speed: Med");
  }
  else {
    // Decrease motor speed
    analogWrite(MOTOR_PIN_ENA, 45);    // Set the motor speed to a lower value (45)
    digitalWrite(MOTOR_PIN_IN1, HIGH);  // Set motor direction (forward)
    digitalWrite(MOTOR_PIN_IN2, LOW);
    lcd.setCursor(0, 1);
    lcd.print("Motor Speed: Low");
  }
 lcd.setCursor(12, 0); // Set cursor to the temperature position on the first line
  lcd.print(temperature); // Print temperature
}
```

6. Testing and Results

6.1 Testing Procedure:

The testing phase involves verifying that the system operates as intended under various conditions. Here's a detailed procedure for testing the system:

1. **Power On the System**:
    - **Connect the 12V power supply** to the system, ensuring all components are connected properly.
    - Turn on the **Arduino** and the system by plugging in the power adapter.
    - The **LCD with I2C module** should light up, and the **Arduino** should begin reading data from the **DHT11 sensor**.
2. **Verify the LCD Display**:
    - Check if the **LCD** correctly displays the initial temperature reading from the **DHT11 sensor**.
    - The LCD should show a message similar to: Temp: 25°C, Fan: Off
    - Make sure the **temperature** reading is within a reasonable range based on the current environment (e.g., the room temperature should be around 25°C or the expected value).
3. **Simulate Temperature Changes**:
    - Adjust the environmental temperature to test the system's response. This can be done by:
        - **Using a heat source** such as a **lamp** or a **hair dryer** (keeping it at a safe distance to avoid damaging the components) to raise the temperature near the DHT11 sensor.
        - Alternatively, the temperature can be lowered by **moving the sensor to a cooler location**.

- Monitor the **LCD screen** as the temperature changes and observe the real-time update of the temperature.

4. **Observe the Fan Speed**:
   - As the temperature rises above a certain threshold (e.g., 30°C), the **fan speed** should start increasing.
   - Initially, when the temperature is below the threshold, the **fan may remain off** or run at a **low speed** (depending on your programming).
   - As the temperature crosses the threshold, the **fan should speed up**, and the **PWM signal** sent to the fan will increase.
   - The LCD should display both the **current temperature** and **fan speed**, such as:
     - For temperatures **below 30°C**: Temp: 28°C, Fan: Low
     - For temperatures **above 30°C**: Temp: 32°C, Fan: High
5. **Check the Fan Response**:
   - Verify that the **fan speed** corresponds to the changing temperature values.
   - The fan speed should increase proportionally as the temperature rises, and the PWM signal sent to the L298 motor driver should control this speed.
   - If the temperature continues to rise, the **fan speed** should increase to its maximum limit (if the PWM is set to 100% at a certain threshold, such as 35°C).
   - If the temperature falls back below the threshold, the **fan should slow down** or **turn off** as the system cools.

6.2 Results:

1. **Temperature Measurement**:
    - As the system operates, the **LCD** will continuously display the temperature in real-time, updating every 1-2 seconds.
    - The displayed temperature should match the expected values based on the environmental conditions. For instance, the temperature may read **25°C** initially, and once the heat source is applied, it should increase gradually.
2. **Fan Speed Control**:
    - The **fan speed** should increase progressively as the **temperature exceeds certain thresholds**.
        - Below **30°C**, the fan should be **off** or run at a low speed (e.g., 10-20% PWM).
        - At **30°C**, the fan should start running at **medium speed** (e.g., 50% PWM).
        - At **35°C or higher**, the fan should run at **full speed** (e.g., 100% PWM).
3. **LCD Feedback**:
    - As the temperature rises and the fan speed changes, the **LCD display** will show the current temperature and corresponding fan speed. The displayed information should look like:
        - **Low Speed (Fan off)**: Temp: 25°C, Fan: Off
        - **Moderate Speed (Fan running at medium speed)**: Temp: 30°C, Fan: Medium
        - **Full Speed (Fan running at high speed)**: Temp: 35°C, Fan: High
4. **Consistency**:
    - The system should exhibit consistent behavior: each time the temperature changes, the fan speed should adjust smoothly in proportion to the temperature.

- The system should be responsive to changes in the environment, whether the temperature rises or falls.

5. **Error Handling**:
   - If there is any error with the DHT11 sensor or communication issues between the sensor and the Arduino, the system should handle it gracefully.
   - A potential error state could be indicated on the **LCD** (e.g., "Sensor Error" or "Check Connections").

Example Testing Scenarios:

1. **Room Temperature (25°C)**:
   - The system starts with the fan **off**.
   - The LCD displays: Temp: 25°C, Fan: Off
2. **After Heating Up the Room (30°C)**:
   - The temperature rises, and the fan speed increases to a **moderate level** (PWM at 50%).
   - The LCD displays: Temp: 30°C, Fan: Medium
3. **Further Increase in Temperature (35°C)**:
   - The temperature continues to rise, and the fan reaches **full speed** (PWM at 100%).
   - The LCD displays: Temp: 35°C, Fan: High
4. **Cooling Down the Room (25°C)**:
   - After removing the heat source, the temperature drops, and the fan speed decreases or turns off.
   - The LCD displays: Temp: 25°C, Fan: Off

7. Conclusion

This project successfully demonstrates an efficient method for controlling the speed of a fan based on ambient temperature. The system uses an Arduino UNO, a DHT11 sensor for temperature measurement, an L298 motor driver to regulate the fan's speed, and an LCD with I2C module to display real-time feedback. The core logic is based on the PWM (Pulse Width Modulation) control of the fan, which adjusts its speed proportionally to the temperature. The system functions by continuously reading the temperature data from the DHT11 sensor, processing it in the Arduino, and sending appropriate signals to the L298 motor driver, which in turn adjusts the fan's speed.

The LCD provides clear real-time updates on both the temperature and the fan's operating status. This setup serves as a simple, yet effective solution for maintaining desired temperature conditions in a variety of applications.

Applications:

1. Cooling Electronics:
2. Ventilation Systems:
3. Smart Homes:
4. Greenhouses and Agriculture:
5. HVAC Systems:

Future Improvements:

1. Add Humidity Control in Addition to Temperature:
2. Improve the Fan Control Algorithm for Smoother Speed Transitions:
3. Incorporate Wi-Fi or Bluetooth for Remote Monitoring and Control:
4. Enhance User Interface:
5. Automatic Calibration and Self-Adjustment:
6. Integration with Other Environmental Sensors:

8. References

1. G. M. Debele and X. Qian, "Automatic Room Temperature Control System Using Arduino UNO R3 and DHT11 Sensor," *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, Chengdu, China, 2020, pp. 428-432, doi: 10.1109/ICCWAMTIP51612.2020.9317307.

2. A. Gupta, M. Gupta and K. Markam, "Temperature Control Smart Fan: Model Analysis and Working Analysis," 2023 1st International Conference on Innovations in High Speed Communication and Signal Processing (IHCSP), BHOPAL, India, 2023, pp. 527-530, doi: 10.1109/IHCSP56702.2023.10127120.

3. Arduino Documentation:

   URL: Arduino Documentation

4. DHT11 Sensor Datasheet:

   URL: DHT11 Sensor Datasheet

5. L298 Motor Driver Datasheet:

   URL: L298 Motor Driver Datasheet

6. LCD with I2C Module Documentation:

   URL: LCD I2C Documentation

7. L298 Motor Driver Overview:

   URL: L298 Motor Driver Overview

8. Arduino Libraries for DHT Sensors:

   URL: DHT Sensor Library on GitHub