Room temperature monitoring & control

Bachelor of Technology

In

ELECTRONIC AND COMMUNICATION ENGINEERING

Specialization in

Artificial intelligence and cybernetics

Submitted to

VIT BHOPAL UNIVERSITY (M.P.)

Submitted by

RAKESH LODHI(21BAC10034)

Under the Supervision of

Dr.KUMKUM DUBEY
VIT BHOPAL UNIVERSITY

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGG

BHOPAL (M.P.)-466114

May – 2024

Chapter 1

1. Introduction

In the realm of automation and smart systems, the integration of sensors with control mechanisms has become a cornerstone of modern engineering. This project delves into the realm of temperature-based automation, specifically focusing on the control of a fan in response to ambient temperature variations. By utilizing an Arduino microcontroller, a DHT11 temperature and humidity sensor, and an L298N motor driver, this project showcases how a simple yet effective system can be designed to regulate a fan's operation based on environmental conditions.

The primary objective of this project is to demonstrate the practical application of temperature sensing in controlling a fan. The ability to automate the fan's activation and deactivation based on predefined temperature thresholds not only showcases the versatility of such systems but also highlights their potential in energy conservation and comfort management in various environments.

Through this project, we aim to explore the intricacies of sensor integration, microcontroller programming, fan control mechanisms, and real-time data display. By providing a comprehensive overview of the circuit design, code implementation, testing procedures, and results analysis, this project report aims to serve as a valuable resource for enthusiasts, students, and professionals interested in temperature-based automation and control systems.

1.1 Motivation of the study

The motivation behind this study stems from the increasing demand for energy-efficient and intelligent systems in various sectors such as home automation, industrial control, and environmental monitoring. Temperature-based automation is a crucial aspect of such systems, allowing for precise control and optimization of energy consumption based on environmental conditions. The motivation to explore temperature-based control of a fan lies in its practical applications, including:

Energy Conservation: By activating the fan only when necessary based on temperature thresholds, energy wastage can be minimized, leading to reduced electricity consumption and lower utility costs.

Comfort Management: Automated fan control ensures optimal comfort levels by adjusting airflow based on ambient temperature changes, enhancing user experience and well-being.

Environmental Impact: Efficient use of resources through automation contributes to sustainability efforts, reducing carbon footprint and environmental impact.

1.2 Objective of the work

Objective of Work:

1. Design a circuit layout integrating Arduino, DHT11 sensor, L298N motor driver, and fan for temperature-based control.
2. Implement Arduino code to read temperature data, process it, and control fan operation based on temperature thresholds.
3. Utilize PWM to regulate fan speed and direction, ensuring efficient and smooth operation.
4. Integrate an LCD display for real-time temperature and fan status feedback.
5. Conduct thorough testing to validate system functionality, temperature control accuracy, and responsiveness.
6. Analyze test results to assess system performance, energy efficiency, and practical usability in temperature-controlled environments.

Chapter 2

3.Problem formulation and proposed methodology

The problem addressed in this project is the need for an efficient and user-friendly display system for microcontroller-based applications. Traditional methods of interfacing LCD displays with microcontrollers often involve complex wiring and limited pin availability, which can hinder the development of interactive and dynamic display systems. Therefore, the challenge is to create a display solution that overcomes these limitations and offers a streamlined interface for displaying information, controlling display features, and enhancing user interaction.

Key aspects of the problem formulation include:

Simplification of Wiring: The project aims to simplify the wiring required for interfacing an LCD display with a microcontroller, reducing the number of pins needed for communication.

Efficient Communication: Utilizing I2C communication protocol enables efficient data transfer between the microcontroller and the display, enhancing system performance.

User-Friendly Interface: Designing a user-friendly interface allows for easy control of display features such as text output, cursor positioning, scrolling, and custom character display.

Scalability and Flexibility: The display solution should be scalable and adaptable to accommodate future enhancements, such as sensor integration, menu systems, touchscreen interfaces, and data logging capabilities.

Reliability and Performance: Ensuring the reliability and performance of the display system under various operating conditions is crucial for its practical application in real-world projects.

3.2 Proposed methodology

Firstly, we know the Adress of I2C that we using in LCD then program we program ARDUINO UNO & ARDUINO IDE. Next.
Further, we connect the DHT11 module to L298N motor driver as per the circuit diagram. Next we connect the 1 DC motors to the the L298N motor driver. We also connect a 12 -20V DC power supply to power the complete setup.

After this, we power the robot through the DC power supply. Once the Circuit response, we now take the room temperature.

th

● 3.2.1 Components

Arduino :A microcontroller board contains on-board power supply, USB port to communicate with PC, and an Atmel microcontroller chip. In 2005, building upon the work of Hernando Barragán (creator of Wiring), Massimo Banzi and David Cuartielles created Arduino, an easy-to-use programmable device for interactive art design projects, at the Interaction Design Institute Ivrea in Ivrea, Italy. It is an open source hardware, anyone can get the details of its design and modify it or make his own one himself. Basically there are different types of Arduino boards are available. In this project we used Ardunio uno.

Fig 3.2.1.1Arduino

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version

2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

Fig 3.2.1.2 LDR

- **The LCD** -(Liquid Crystal Display) 16x2 is a common type of alphanumeric display module that can show 16 characters per line and has 2 lines. It uses liquid crystal technology to display text and numbers, making it suitable for various applications, including microcontroller-based

projects like Arduino. The module typically includes a backlight for visibility in different lighting conditions and can be interfaced with microcontrollers using protocols such as I2C or parallel communication.

I2C (Inter-Integrated Circuit) is often used to interface with LCD (Liquid Crystal Display) modules, especially in projects where multiple components need to communicate over a single bus. When using I2C for an LCD display, you typically need an I2C adapter or module, such as the PCF8574, which converts the parallel data from the LCD to serial data for I2C communication.

The advantage of using I2C for an LCD display is that it reduces the number of pins required to connect the LCD to a microcontroller like Arduino. Instead of using many digital pins for data and control signals, I2C allows you to use just two pins (SDA and SCL) for communication. This simplifies the wiring and frees up digital pins for other purposes in your project.

Fig 3.2.1.3 LCD DISPLAY I2C

L298N MOTOR DRIVER.: The L298N Motor Driver module consists of an L298 Motor Driver IC, 78M05 Voltage Regulator, resistors, capacitor, Power LED, 5V jumper in an integrated circuit. 78M05 Voltage regulator

will be enabled only when the jumper is placed. When the power supply is less than or equal to 12V, then the internal circuitry will be powered by the voltage regulator and the 5V pin can be used as an output pin to power the microcontroller. The jumper should not be placed when the power supply is greater than 12V and separate 5V should be given through 5V terminal to power the internal circuitry.

ENA & ENB pins are speed control pins for Motor A and Motor B while IN1& IN2 and IN3 & IN4 are direction control pins for Motor A and Motor B.

DHT11 sensor- The DHT11 sensor is a digital temperature and humidity sensor known for its simplicity and affordability. It provides accurate temperature and humidity readings and communicates data to microcontrollers like Arduino through a single digital pin. Its low cost and ease of use make it popular for various projects requiring environmental monitoring and control.

3.3 Implementation

Overview of the Circuit:

The circuit consists of an Arduino board, an LCD display (16x2), an I2C module (e.g., PCF8574), and necessary connections using jumper wires. The I2C module acts as an interface between the Arduino and the LCD display, simplifying the wiring and reducing the number of pins required for communication. The Arduino communicates with the I2C module over the I2C bus (SDA and SCL lines), while the I2C module communicates with the LCD display using standard parallel communication.

Working of the Circuit:

Hardware Connections:

Connect the SDA and SCL pins of the I2C module to the corresponding pins on the Arduino (A4 and A5, respectively). Connect VCC and GND of the I2C module to the Arduino's 5V and GND pins, respectively. Connect the VCC and GND pins of the LCD display to the I2C module. Connect the SDA and SCL pins of the LCD display to the I2C module.

Initialization:

In the Arduino code, include the necessary libraries for I2C communication and LCD control (e.g., Wire.h and LiquidCrystal_I2C.h).

Initialize the LiquidCrystal_I2C object with the I2C address of the module (commonly 0x27 for many modules) and the LCD dimensions (16x2).

Initialize the LCD display using lcd.init() and turn on the backlight with lcd.backlight().

Code Execution:

In the setup() function, set the cursor position using lcd.setCursor(row, column) and print desired text using lcd.print("text").

Optionally, you can control the cursor blinking, scrolling, and other display features using LCD library functions.

The loop() function can be used for continuous operation or updating the display as needed, such as displaying sensor data, menu options, or dynamic information.

Uploading and Testing:

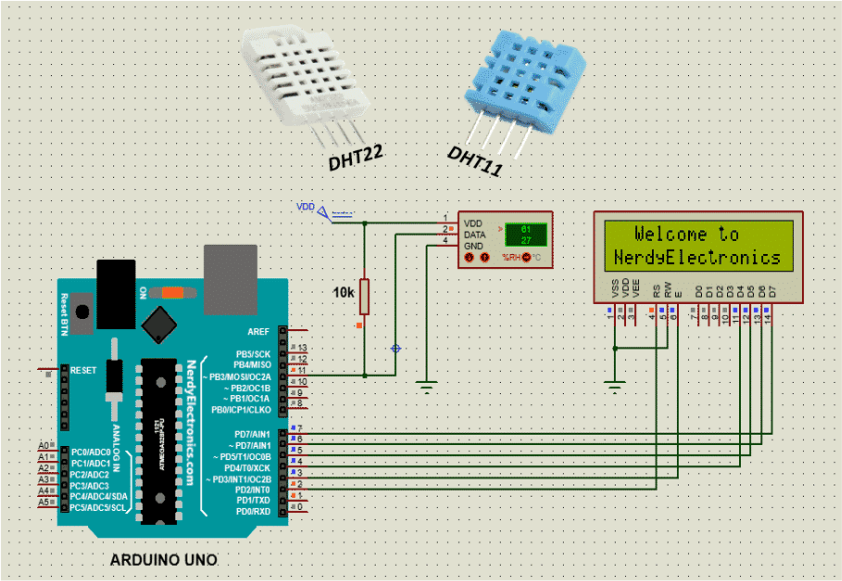
Upload the code to the Arduino board and observe the LCD display for the printed text.

The LCD should display the specified text on the specified row and column position as defined in the code.

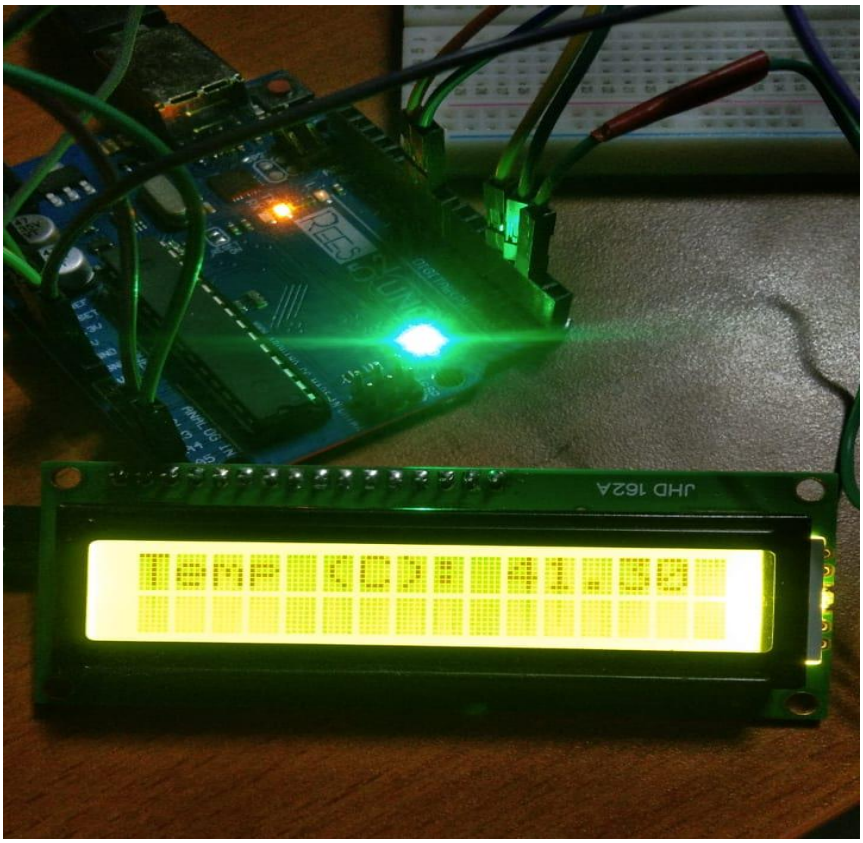
Customization:

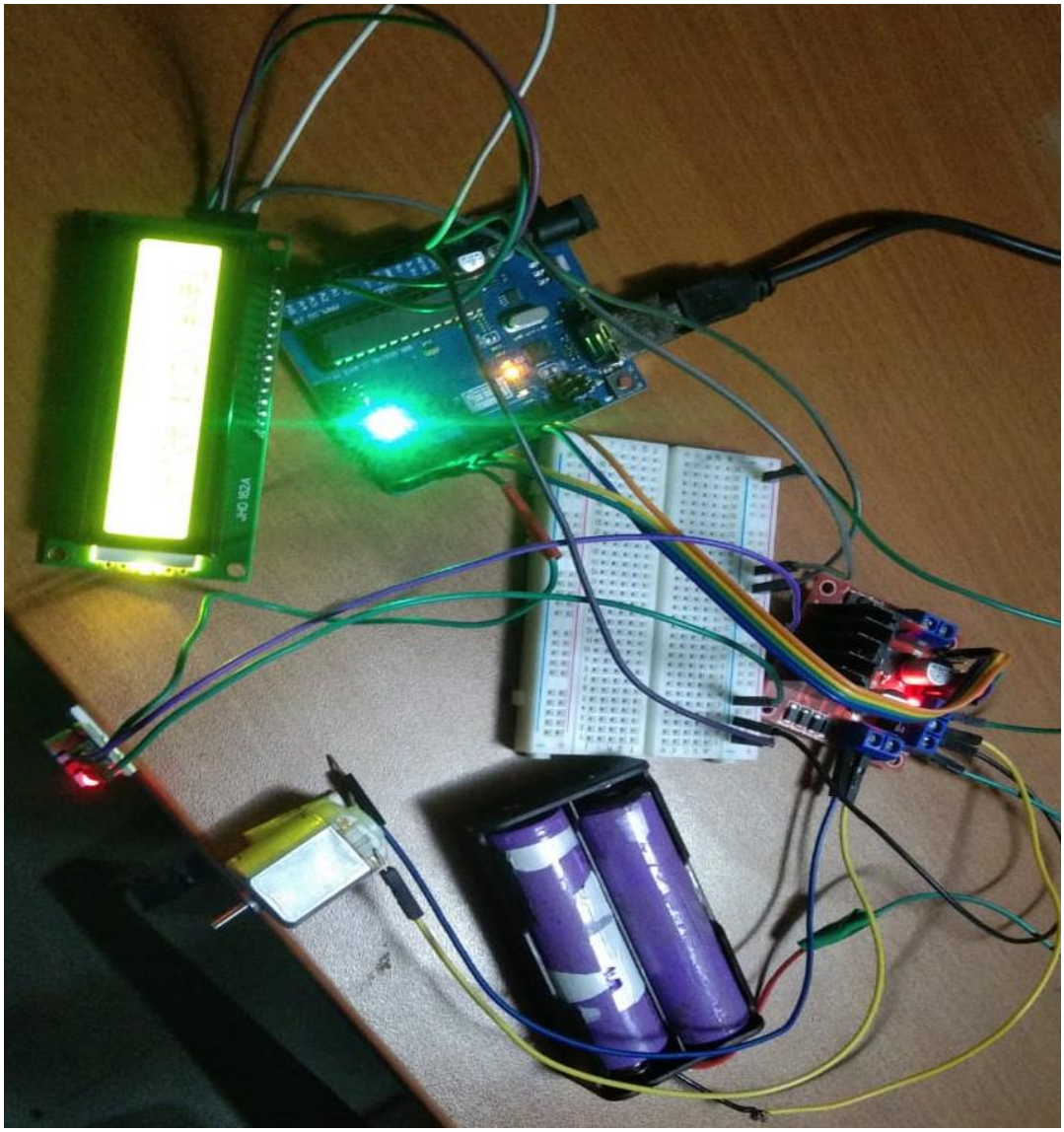
Modify the code to display different messages, control the cursor, clear the display, or scroll text as per your project requirements.

Explore additional LCD library functions for advanced display control, such as creating custom characters, adjusting contrast, or displaying numeric values.



Circuit Snap




JHD 162A

Chapter 5

4. Conclusion and future scope

Conclusion:

The project successfully demonstrated the interfacing of an LCD display with Arduino using I2C communication. The use of I2C simplified the wiring and reduced pin usage, making the circuit more efficient and manageable. The code implementation allowed for displaying text and controlling the display's features, showcasing the versatility of the Arduino platform. The project achieved its objective of creating a functional and interactive display system.

Future Scope:

Enhanced Display Functionality: Implement scrolling text, custom characters, and graphical elements for more dynamic and engaging displays.

Sensor Integration: Integrate sensors to display real-time data such as temperature, humidity, or environmental parameters.

Menu System: Develop a menu system for user interaction, allowing navigation through different options and actions.

Remote Control: Implement remote control functionality using wireless communication modules like Bluetooth or Wi-Fi for remote display control.

Data Logging: Incorporate SD card modules or cloud connectivity to log and display data over time, enabling data analysis and visualization.

Touchscreen Interface: Upgrade to a touchscreen LCD for intuitive user interaction and advanced display capabilities.

Customized Interfaces: Design customized interfaces for specific applications like IoT dashboards, home automation controls, or industrial monitoring systems.

By exploring these avenues, the project can be expanded into more sophisticated display systems with enhanced functionality and usability, catering to a wide range of applications and user requirements.

References

[1] Dilruba Yamin, Automatic Room Temperature Controlled Fan Speed Controller Using PT100,

International Journal of Scientific and Engineering Research, Volume 6, Issue 8, August 2015.

[2] M. Kumaran, I. Vikram, S. Kishore Kumar, R. Rajesh Kumar, S. Lokesh, Design of An Automatic Fan Speed Controlling System Using Arduino UNO. International Journal of Intellectual Advancements and Research in Engineering Computations, Volume-6 Issue-2, 2018 [2039-2042].

[3] Joseph Chuma, Design and Simulation of an Automatic Room Heater Control System, Botswana International University of Science and Technology, 2018.

[4] Siddika, S. F. Nasrin, Design, and Arduino-based Automatic Fan Control System Using PIR Sensor and LM35 Sensor. GSJ: Volume 6, Issue 8, August 2018.

[5] M. M. San, C. K. Win & K. Z. Mon, Design and Simulation of Fan Speed Control using Arduino UNO and LM35DZ, International Journal of Engineering

Research and Advanced Technology (IJERAT), (EISSN: 2454-6135) Volume.6, Issue 6 June -2020.

[6] Suraj Kaushik, Automatic Fan Speed Control using Temperature and Humidity Sensor and Arduino, International Journal of Advance Research Ideas and Innovations in Technology Volume 2, Issue 8, 2018.

[7] Components101, January 5, 2018, DHT11 –Temperature and Humidity sensor, Available:

APPENDICES

//CODE

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

#define DHTPIN 2       // Digital pin connected to the DHT11 sensor
#define DHTTYPE DHT11  // DHT11 sensor type
DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal_I2C lcd(0x27, 16, 2);  // Initialize the LCD with I2C address 0x27, 16 columns, and 2 rows

const int fanPin1 = 3;  // Digital pin connected to IN1 of the motor driver
const int fanPin2 = 4;  // Digital pin connected to IN2 of the motor driver
const int motorSpeedPin = 9;  // PWM pin connected to ENA of the motor driver

void setup() {
  Serial.begin(9600);  // Initialize serial communication for debugging
  lcd.init();         // Initialize the LCD
  lcd.backlight();    // Turn on the backlight (if available)
```

```
  dht.begin();        // Initialize the DHT11 sensor

  lcd.setCursor(0, 0);     // Set cursor to the first row, first column
  lcd.print("Temp (C): "); // Display "Temp (C): " on LCD

  pinMode(fanPin1, OUTPUT);       // Set fan control pins as output
  pinMode(fanPin2, OUTPUT);
  pinMode(motorSpeedPin, OUTPUT); // Set motor speed control pin as output
}

void loop() {
  float temperature = dht.readTemperature();  // Read temperature from DHT11 sensor in Celsius
  Serial.print("Temperature: ");
  Serial.println(temperature);

  lcd.setCursor(10, 0);    // Set cursor to the first row, eleventh column (after "Temp (C): ")
  lcd.print(temperature);  // Display temperature on LCD

  if (temperature > 36) {
    Serial.println("Turning fan on");
    analogWrite(motorSpeedPin, 255);  // Set motor speed to maximum (255) when turning the fan on
    digitalWrite(fanPin1, HIGH);      // Set IN1 to HIGH
```

```
    digitalWrite(fanPin2, LOW);       // Set IN2 to LOW
  } else {
    Serial.println("Turning fan off");
    analogWrite(motorSpeedPin, 0);    // Set motor speed to 0 when turning the fan off
    digitalWrite(fanPin1, LOW);       // Set IN1 to LOW
    digitalWrite(fanPin2, LOW);       // Set IN2 to LOW
  }

  delay(2000);  // Delay for 2 seconds before updating readings and fan control
}
```