Automatic Temperature Control System

Aliyan Hassan Khan	211390022
Shahwaiz	211390001

Subject: Introduction to Embedded Systems

Department of Electrical Engineering

GIFT University, Gujranwala

Pakistan

March 2024

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

Abstract

Abstract: Automatic temperature control systems are vital in numerous industries and applications, ensuring precise temperature regulation. These systems employ sensors to measure temperature, micro-controllers to process data, and actuators to adjust heating or cooling mechanisms as needed. This abstract explores the design, implementation, and advantages of automatic temperature control systems, emphasizing their importance in maintaining optimal temperatures for processes such as food production, HVAC (heating,ventilation and air conditioning systems, and environmental control. By incorporating advanced algorithms and sensing technologies, these systems offer increased efficiency, energy savings, and improved product quality. Furthermore, advancements in connectivity enable remote monitoring and control, enhancing their usefulness in modern industrial and residential contexts. This abstract highlights the significance of automatic temperature control systems as essential tools for achieving accuracy, consistency, and reliability in temperature-sensitive applications.

Chapter 1

Introduction

Automatic temperature control systems have become essential components in a wide range of industries and applications, helping to maintain ideal temperature settings for processes, surroundings, and comfort. These systems use innovative technology to constantly monitor, evaluate, and modify temperatures, ensuring precision, consistency, and efficiency in temperature regulation. Automatic temperature control systems have various advantages, including improved product quality, energy savings, and increased comfort. This introduction presents an overview of the design, operation, and significance of automatic temperature control systems, emphasizing their usefulness in modern applications and the technological developments that are boosting their evolution.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Project Details

2.1 Explanation

The code begins by importing necessary libraries like EEPROM and LiquidCrystal for data storage and display. LCD pins are specified, and the LCD is initialized with its dimensions.

Variables are declared for ADC results, resistance calculations, temperature readings, threshold settings, fan speed control, and display flags.

In the setup function, the thermistor pin, temperature threshold buttons, fan, and buzzer are configured. The LCD displays a welcome message if it's the first use and initializes using default temperature readings from EEPROM.

In the loop function:

* Temperature is read by averaging ADC values over 100 readings and computing resistance and temperature using the Steinhart-Hart equation. * Fan speed is adjusted based on temperature and threshold settings to maintain desired temperature range. * Buzzer sounds if temperature exceeds high threshold. * Buttons are checked for adjusting temperature thresholds, with long presses toggling between upper and lower limits. * LCD is updated to display temperature, fan speed, and settings, with blinking indicating the value being changed. * Buzzer is deactivated after sounding alarm. * Delays and flags are used for button debouncing and setup mode control.

This code ensures smooth operation of the Automatic Temperature Control System, handling temperature regulation, user interaction, and display updates.

2.2 Components Details

2.2.1 Arduino

Arduino is an electronics platform available as open-source software and hardware components. Because of its accessible and user-friendly design, interactive electronic projects and prototypes are made by professionals, students, and hobbyists alike.

A microcontroller board, which functions as the system's brain, is part of the Arduino hardware. Typically, this board has buttons and LEDs as built-in components, along with digital and analog input/output pins. There are many different kinds of Arduino boards available, from basic models like the Arduino Uno to more sophisticated ones like the Arduino Mega.

The Arduino Integrated Development Environment (IDE), which is used to write, compile, and upload code to the Arduino board, makes up the software component. The code, which is usually written in a language like C or C++, regulates how the Arduino board behaves and interacts with the sensors, actuators, and other electrical parts that are attached to it. Arduino helps users to create projects from simple Led blinkers to complex robotics.

2.2.2 NTC Thermistor 10K

An NTC (Negative Temperature Coefficient) thermistor is a semiconductor-based temperature sensor. Its electrical resistance decreases with increasing temperature. This property makes it suitable for temperature measurement in electronic circuits. The resistance-temperature relationship follows a predictable curve, usually represented by a Beta () value. The temperature of the environment can be determined by measuring the thermistor's resistance change.

2.2.3 IRFZ44N (MOSFET)

The IRFZ44N is a power MOSFET, used for switching and amplification. It offers high current capability and low on-resistance, enabling efficient power handling. With a typical voltage rating of 55V and current capacity up to 49A, it suits various applications like motor control and power supplies.

2.2.4 LCD (16x2)

The 16x2 LCD (Liquid Crystal Display) is a commonly used alphanumeric display module that consists of 16 columns and 2 rows, allowing it to display up to 16 characters per row and 2 rows of characters.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Simulation Working

In this project,simulation starts with the use of an Arduino UNO micro-controller, which serves as the micro-controller for our Automatic Temperature Control System. To detect temperature changes, we use a 10k NTC Thermistor with a 10k pull-down resistor. The system's user interface includes three push buttons for temperature setting and a 16×2 LCD for data presentation. The circuit has a 10k potentiometer to ensure optimal display contrast. The FET IRFZ44N allows for more efficient fan speed management, while a normal fan provides the necessary air circulation. In addition, a buzzer is used to sound an alarm when temperature limit is exceeded. After compiling the code in the Arduino IDE and generating the hex file, it is easily implemeted into the simulation environment.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Proteus Circuit

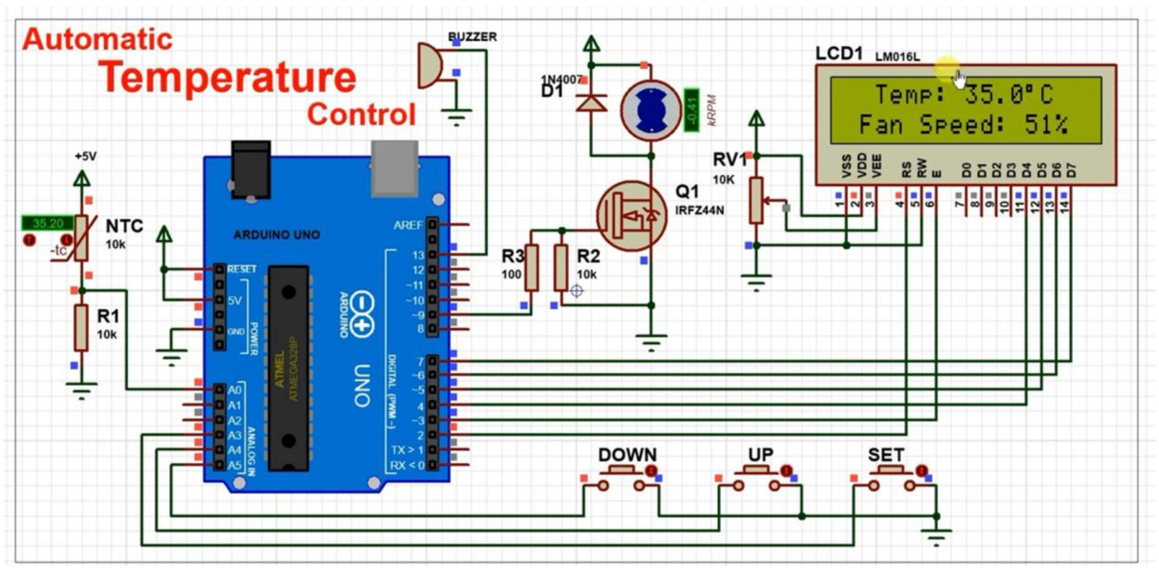

Figure 4.1: Circuit Diagram of Automatic Temperature Control System

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

PROGRAMMING

A.1 Automatic Temperature Control System

Listing A.1: Arduino IDE Code

```

#include <EEPROM.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

#define ThermistorPin A0 // for Arduino microcontroller

#define bt_set  A3
#define bt_up   A4
#define bt_down A5

#define fan 9
#define buzzer 13

long ADC_Value;
float R1 = 10000; // value of R1 on board
float logR2, R2, T;

//steinhart-hart coeficients for thermistor
```

```
float c1 = 0.001129148, c2 = 0.000234125, c3 =
   0.0000000876741;

float setL_temp = 30.5;
float setH_temp = 39.5;
float temp=0;

int duty_cycle;
int Set=0, flag=0, flash=0;

void setup() {// put your setup code here, to run once:

pinMode(ThermistorPin, INPUT);

pinMode(bt_set,  INPUT_PULLUP);
pinMode(bt_up,   INPUT_PULLUP);
pinMode(bt_down, INPUT_PULLUP);

pinMode(fan, OUTPUT);
pinMode(buzzer, OUTPUT);

lcd.begin(16, 2); // Configura lcd numero columnas y filas
lcd.clear();
lcd.setCursor (0,0);
lcd.print("   Welcome To   ");
lcd.setCursor (0,1);
lcd.print("  Temp Control  ");

if(EEPROM.read(0)==0){}
else{
```

```
EEPROM.put(10, setL_temp);
EEPROM.put(15, setH_temp);
EEPROM.write(0, 0);
}
EEPROM.get(10, setL_temp);
EEPROM.get(15, setH_temp);

delay(2000); //wait 1000 mS for next measure
lcd.clear();
}

void loop(){

ADC_Value=0;
for(int i=0; i<100; i++) {
ADC_Value = ADC_Value+analogRead(ThermistorPin);
delay(1);
}

ADC_Value=ADC_Value/100;
R2 = R1 * (1023.0 / (float)ADC_Value - 1.0); //calculate
    resistance on thermistor
logR2 = log(R2);
temp = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2)); //
    temperature in Kelvin
temp = temp - 273.15; //convert Kelvin to Celcius

int value1 = temp*10;
int value2 = setL_temp*10;
int value3 = setH_temp*10;
```

```

duty_cycle = map(value1, value2, value3, 0, 100);
if(duty_cycle>100)duty_cycle=100;
if(duty_cycle<0)duty_cycle=0;

if(temp<setL_temp){digitalWrite(fan, 0);}
              else{analogWrite(fan, (duty_cycle*2)+55); }

if(temp>setH_temp){digitalWrite(buzzer, HIGH);  delay(100)
   ;}

if(digitalRead(bt_set)==0){
digitalWrite(buzzer, HIGH);
 if(flag==0){ flag=1;
  Set=Set+1;
  if(Set>2){Set=0; flash=0;}
delay(200);
 }
}else{flag=0;}

if(digitalRead(bt_up)==0){
digitalWrite(buzzer, HIGH);
if(Set==1){setL_temp = setL_temp+.1;EEPROM.put(10,
   setL_temp);}
if(Set==2){setH_temp = setH_temp+.1;EEPROM.put(15,
   setH_temp);}
delay(10);
}

if(digitalRead(bt_down)==0){
```

```
digitalWrite(buzzer, HIGH);
if(Set==1){setL_temp = setL_temp-.1;EEPROM.put(10, setL_temp);}
if(Set==2){setH_temp = setH_temp-.1;EEPROM.put(15, setH_temp);}
delay(10);
}

if(Set==0){

lcd.setCursor(0,0);
lcd.print("  Temp: ");
lcd.print(temp,1);
lcd.write(223);
lcd.print("C   ");

lcd.setCursor(0,1);
lcd.print(" Fan Speed:");
if(duty_cycle<100)lcd.print(" ");
lcd.print(duty_cycle);
lcd.print("%   ");

}else{
lcd.setCursor(0,0);
lcd.print("  Setting Mode  ");

lcd.setCursor(0,1);
lcd.print("L:");
if(Set==1 && flash==1){lcd.print("    ");}
else{lcd.print(setL_temp,1);}
```

```
lcd.print("C   ");

lcd.setCursor(9,1);
lcd.print("H:");
if(Set==2 && flash==1){lcd.print("    ");}
else{lcd.print(setH_temp,1);}
lcd.print("C   ");
}

if(Set>0){flash=!flash;}
delay(1); //wait 1 mS for next measure
digitalWrite(buzzer, LOW);
}
```

[1]

Bibliography

[1] “circuit basics,” https://marobotic.com/2024/02/08/automatic-temperature-control-system-using-arduino/, accessed: 08-03-2024.