
University of Messina

Department of Engineering

Engineering and Computer Science

# Embedded Systems Project:
# Digital Thermometer with a temperature control system

Teacher:

Prof. Francesco Longo

Student:

Abdirashova Aruzhan

2022

# Contents

# Chapter 1

## Introduction

The purpose of this project is to make a digital thermometer with a temperature control system.

In this project I have an opportunity to understand the concepts and use of the Arduino R3 board, and actually run an experiment using a selection of this device.

Digital room temperature is very important since it is needed to keep in check a certain temperature of a room or an atmosphere digitally. That means it overrides the stress of using an analog temperature reader which might involve more calculations to get the current temperature of the environment. This system enables the user to obtain a more precise representation of the temperature in the room using a thermistor.

# Chapter 2

## Hardware

This chapter provides a detailed explanation on how the system was designed and constructed.

In this project, we interfaced a thermistor with an Arduino to design a digital thermometer. The measured temperature will be directly displayed on a 16*2 LCD. Thermistor is capable of reading the temperature in Centigrade scale. The output voltage of the sensor is directly proportional to the temperature in centigrade.

At the end of the chapter, we can observe how all the components have been connected together.

## 2.1 Overview

Components used:

1. Arduino Uno: microcontroller;
2. Arduino IDE;
3. LCD 1602 Module Display: widely used in Arduino's projects LCD display, it has 2 lines on which can be displayed 16 characters;
4. RGB Led;
5. RTC Module: used to store clock information also when the main device is without power;
6. Resistors;
7. Power supply.

### 2.1.1 Arduino UNO R3 Board

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; all it is needed is to simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. It is possible to tinker with the Uno without worrying too much about doing something wrong, worst case scenario the user can replace the chip for a few dollars and start over again.

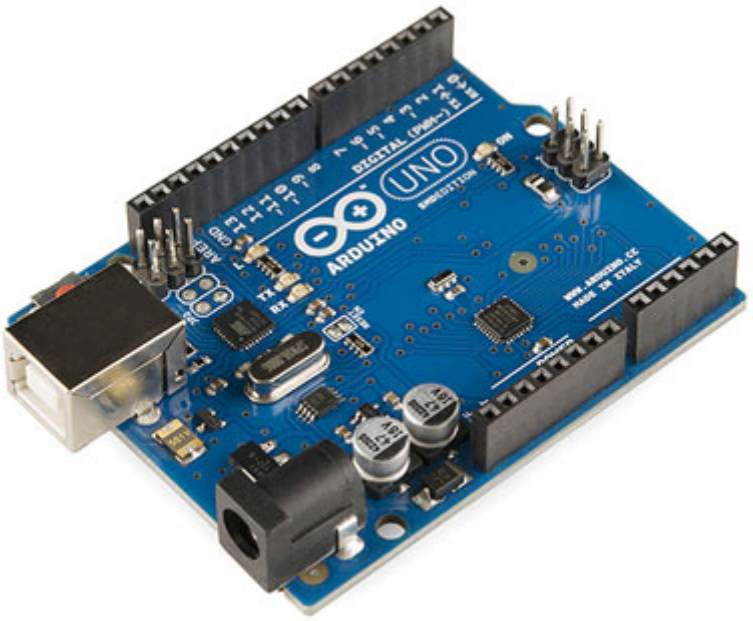


Figure 1: Arduino UNO R3 board.

In this project we have used a Arduino to control the whole process of the system. Arduino is a controller which runs on an ATmega AVR controller. Arduino is an open source hardware platform and very useful for project development purposes.

### 2.1.2 RGB LED

An RGB LED has 4 pins, one for each colour (Red, Green, Blue) and a common cathode. It has three different colour-emitting diodes that can be combined to create all sorts of colour. Any color is possible depending on how bright each diode is. So it actually consists of 3 separate LEDs red, green and blue packed in a single case. That's why it has 4 leads, one lead for each of the 3 colours and one common cathode or anode depending on the RGB LED type.

There are two type of RGB LEDs you will encounter and they are wired very differently

1. Common Anode (+)
   This type of LED has four pins. A cathode (-) for each color and a shared anode (+). The LED shares a common 3.3V to 5V input voltage. Grounding each anode will light the different colors.
2. Common Cathode (-)
   This type of LED has four pins. An anode (+) for each color and shared cathode (-). The LED shares a common ground, applying 3.3V to 5V to each Anode will light the different colors. This is the most common RGB LED.

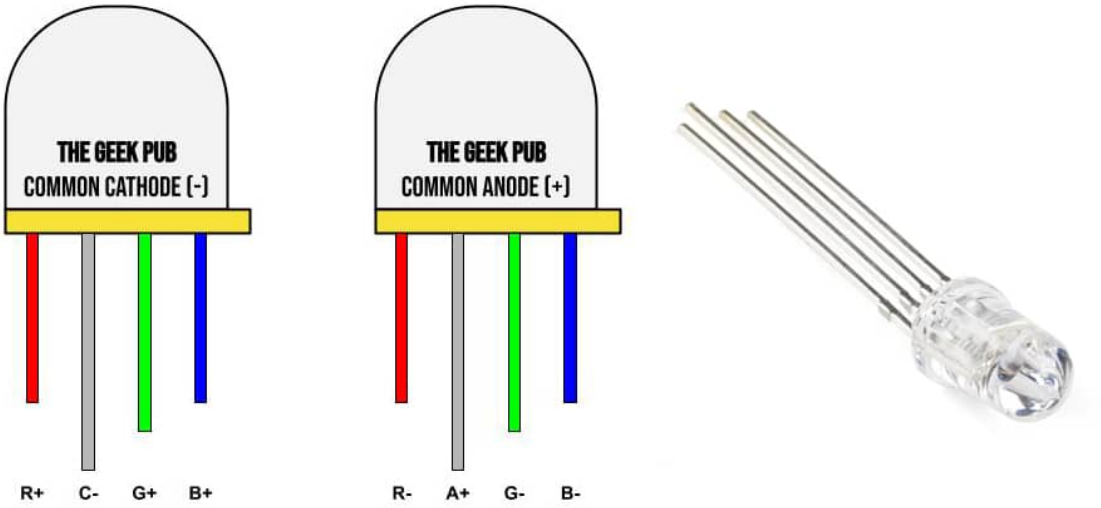


Figure 2: RGB LED.

First of all, we will start with a test, the LED starts in the Red color state, then fade to Green, then fade to Blue. While we do this we show on the LCD Display the expected

color, in this way we can check if the LED matches with the shown color on the display, if yes we know that our system is showing information as expected.

### 2.1.3 Thermistor

Thermistors are variable resistors that change their resistance with temperature. They are classified by the way their resistance responds to temperature changes. In Negative Temperature Coefficient (NTC) thermistors, resistance decreases with an increase in temperature. In Positive Temperature Coefficient (PTC) thermistors, resistance increases with an increase in temperature.
NTC thermistors are made from a semiconducting material (such as a metal oxide or ceramic) that's been heated and compressed to form a temperature sensitive conducting material.
The conducting material contains charge carriers that allow current to flow through it. High temperatures cause the semiconducting material to release more charge carriers. In NTC thermistors made from ferric oxide, electrons are the charge carriers. In nickel oxide NTC thermistors, the charge carriers are electron holes.

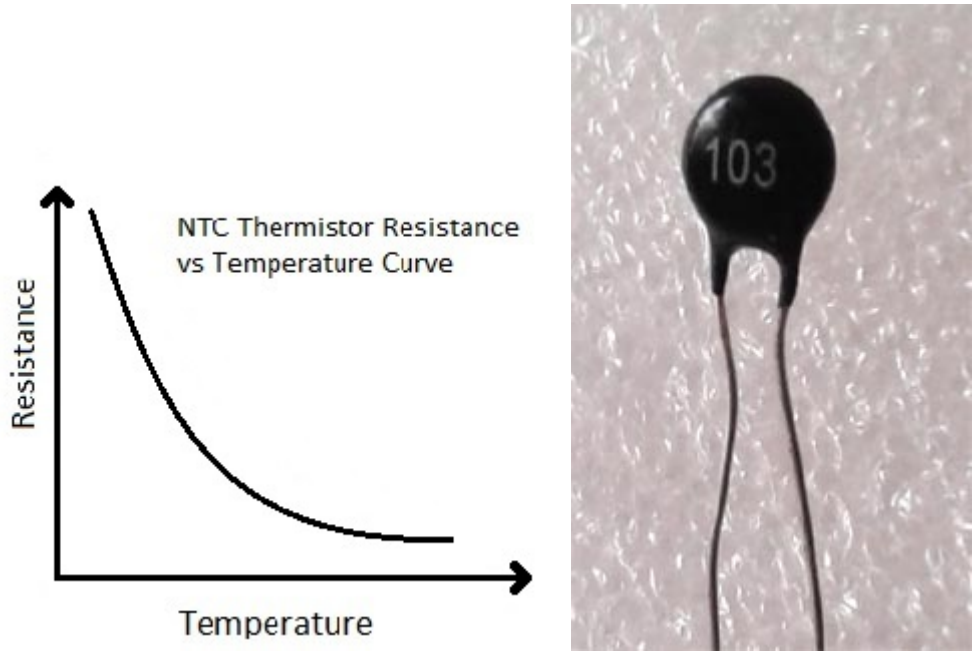


Figure 3: Thermistor.

PTC's are often used as resettable fuses - an increase in temperature increases the resistance which means that as more current passes through them, they heat up and 'choke back' the current, quite handy for protecting circuits.

### 2.1.4 LCD Display

LCD (Liquid Crystal Display) screen is an electronic display module and finds a wide range of applications. A 16x2 LCD display is a very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.


Figure 4: 16x2 LCD Display.

A 16x2 LCD means it can display 16 characters per line and there are 2 such lines as shown in figure 4 above. In this LCD each character is displayed in a 5x7 pixel matrix. This LCD has two registers, namely, Command and Data. The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD.

### 2.1.5 DS1307 RTC module

In many electronic projects it is necessary to run an operation according to the time or date And the calculation of the time and date shouldn't stop when the system shuts down. For this purpose, Real Time Clock (RTC) modules are used.

Real Time Clock or RTC is a system that keeps track of the current time and can be used in any device which needs to keep accurate time.

DS1307 module is one of the most affordable and common RTCs modules. It can accurately keep track of seconds, minutes, hours, days, months, and years.
Some of the DS1307 important features are:

- Ability of Generating Programmable Square-Wave
- Low Current Use; under 500nA in Battery Backup mode
- The Ability to Set the Date Up to Year 2100
- I2C Serial Interface

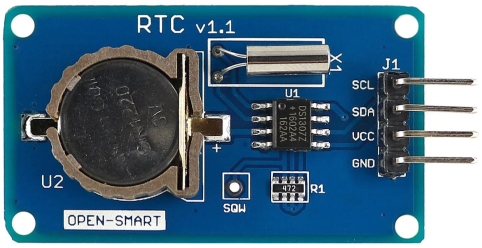


Figure 5: RTC module.

Using this module is simple, RTC part communicates with the microcontroller using I2C protocol.

### 2.1.6 Power supply

All Arduino boards need electric power to function. A power supply is what is used to provide electric power to the boards and typically can be a battery, USB cable, AC adapter or a regulated power source device.
There are different ways to power your Arduino board. The most common way is through the USB connector available on every board.
Arduino Board already have an inbuilt power supply section. Here we only need to connect a 9 volt or 12 volt adaptors with the board.

### 2.1.7 Connecting wires

Connecting wires allows an electrical current to travel from one point on a circuit to another, because electricity needs a medium through which to move. In the case of computers, wires are embedded into circuit boards, carrying pulses of electricity.

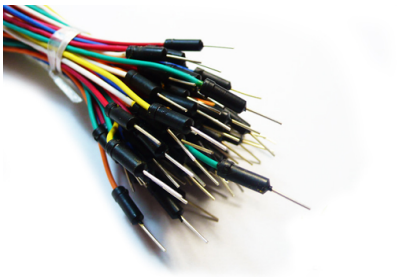

Figure 6: Connecting wires.

In this part, we will analyze the different connection parts separately to have a better comprehension of the circuit. All hardware, places and wire components were drawn on a virtual circuit.

## 2.2 Project Setup

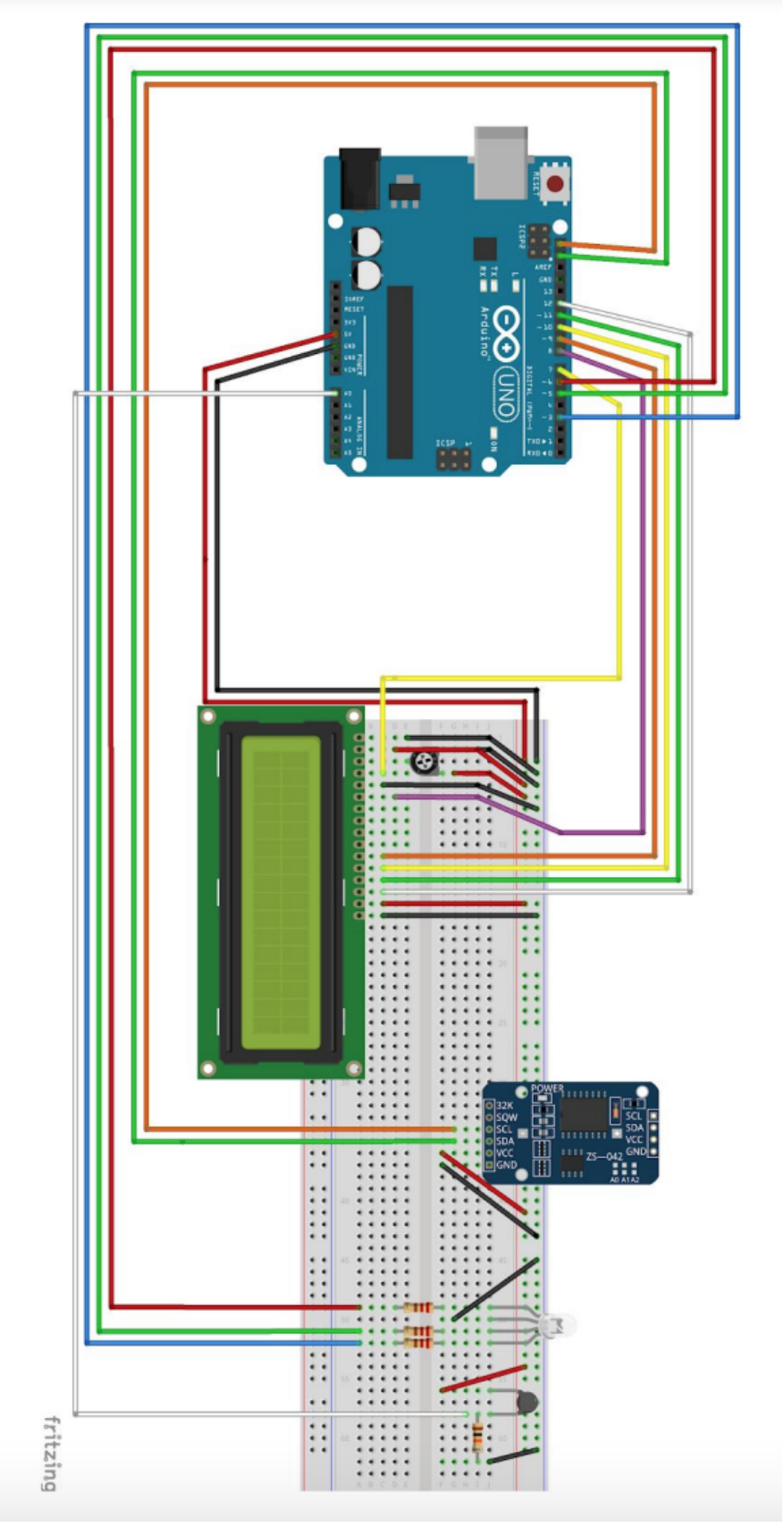


Figure 7: Schematic of main circuit.

### Connecting LCD to the breadboard

There are 11 bus lines: D0 through D7 (8 data lines) and RS, EN, and RW. D0-D7 are the pins that have the raw data we send to the display. The RS pin lets the microcontroller tell the LCD whether it wants to display that data or whether it is a command byte (like, change position of the cursor). The EN pin is the 'enable' line; we use this to tell the LCD when data is ready for reading. The RW pin is used to set the direction - whether we want to write to the display (common) or read from it.

The Arduino board is connected to a PC via USB which will be used as both a power source and serial monitoring.

# Chapter 3

## Software

In the following sections the most critical aspects of the sketch will be described, including the theory behind them.

## 3.1 Libraries

The sketch uses the following libraries:

• LiquidCrystal allows the Arduino board to control the LCD display;

• Wire library allows you to communicate with I2C / TWI devices;

• RTClib is a fork of JeeLab's fantastic real time clock library for Arduino.

## 3.2 Arduino Sketch Description

| | |
|---|---|
| #include <LiquidCrystal.h><br>#include <Wire.h><br>#include "RTClib.h" | Libraries used to control LiquidCrystal displays (LCDs), I2C devices and the RTC module. |
| #define BLUE 6<br>#define GREEN 5<br>#define RED 3<br>#define TEMP 0<br>#define LTD 23.0<br>#define HTD 28.0<br>#define LTN 10.0<br>#define HTN 21.0<br>#define STARTDAYHOUR 9 | Constants with color names are associated with the pin number that manages this color that is connected to the LED. TEMP is used to read the analog value of the pin 0, the raw data from the thermistor. The constants LTD,HTD, LTN and HTN are used to set values of boundaries for temperature in the day and night. With |

| | |
|---|---|
| #define STARTDAYMINUTES 4<br>#define ENDDAYHOUR 21<br>#define ENDDAYMINUTES 30 | STARTDAYHOUR, STARTDAYMINUTES, ENDDAYHOUR and ENDDAYMINUTES we set the time for day mode, before and after the set time is considered night time. |
| char str[16];<br>bool isday = true;<br>RTC_DS3231 rtc;<br>LiquidCrystal lcd(7, 8, 9, 10, 11, 12) | The str variable is used to show information about time and the active mode (Daily or Nightly). isday is used to store the part of the time period (day or night mode). rtc is to interact with RTC module. lcd is to interact with LCD module, the numbers in the parenthesis are the 12 pin used to connect the lcd. |
| void setup() {<br>Serial.begin(9600);<br>lcd.begin(16, 2);<br>pinMode(RED, OUTPUT);<br>pinMode(GREEN, OUTPUT);<br>pinMode(BLUE, OUTPUT);<br>digitalWrite(RED, HIGH);<br>lcd.clear();<br>lcd.setCursor(0, 0);<br>lcd.print("Check RED");<br>delay(1500);<br>digitalWrite(RED, LOW);<br>digitalWrite(GREEN, HIGH);<br>lcd.clear();<br>lcd.setCursor(0, 0);<br>lcd.print("Check GREEN");<br>delay(1500);<br>digitalWrite(GREEN, LOW);<br>digitalWrite(BLUE, HIGH);<br>lcd.clear();<br>lcd.setCursor(0, 0);<br>lcd.print("Check BLUE");<br>delay(1500);<br>digitalWrite(BLUE, LOW); | The setup function is once when the microcontroller starts up. In this function are initialized the pin used for the RGB LED (all set as OUTPUT), the lcd is also initialized with the size (number of columns and rows) of the screen. Here we also set up the serial bus, we use this for debug information and for statistical use as described forward. Here we made the test of the RGB Led and of the LCD display, these tests are made showing a text on the first row of the lcd (we begin to write with the cursor at coordinates 0,0) writing a text that will match the color of the led. To change the color of this led we set the desired pin color at high and the others to low, then we wait for 1.5 seconds (with the delay function) then we clear the lcd and continue with the next color. In this section we also check if the RTC is working and have a valid date and time. If it isn't working we show debug information on the LCD, if |

| | |
|---|---|
| if (!rtc.begin()) {<br>lcd.print("check connection");<br>while (true); }<br>if (rtc.lostPower()) {<br>rtc.adjust(DateTime(F(__DATE__),<br>F(__TIME__))); } } | it hasn't a valid date and time we set. For this purpose we use the function from the RTClib. |
| void loop() {<br>int tempReading=analogRead(TEMP);<br>DateTime now = rtc.now();<br>int h = now.hour();<br>int m = now.minute();<br>int s = now.second();<br>float logRt, tempK;<br>logRt = log(10000.0 * ((1024.0 /<br>float(tempReading) - 1.0)));<br>tempK = 1.0 / (0.001129148 +<br>(0.000234125 + (0.0000000876741 *<br>logRt * logRt)) * logRt);<br>float tempC = tempK - 273.15;<br>Serial.println(tempC);<br>lcd.clear();<br>lcd.setCursor(0, 0);<br>lcd.print("Temp C");<br>lcd.setCursor(6, 0);<br>lcd.print(tempC);<br>isday = (h > STARTDAYHOUR \|\| (h ==<br>STARTDAYHOUR && m >=<br>STARTDAYMINUTES)) && (h <<br>ENDDAYHOUR \|\| (h == ENDDAYHOUR<br>&& m <= ENDDAYMINUTES));<br>if ((tempC >= LTD && tempC <= HTD<br>&& isday) \|\| (tempC >= LTN && tempC<br><= HTN && !isday)) {<br>analogWrite(RED, 0);<br>analogWrite(GREEN, 128);<br>analogWrite(BLUE, 0); }<br>if ((tempC > HTD && isday) \|\| (tempC ><br>HTN && !isday)) { | The loop function will be executed as long as the Arduino controller is enabled.<br>The program reads the analog value from the pin when the thermistor is connected. Also the current time is read from the RTC and the value of hour, minute and second stored as a variable to be used later. The raw value from the thermistor is converted in Kelvin degrees and then in Celsius. This last value is printed on the serial bus, we will use it for debug information but also for statistics. The temperature value in Celsius degrees is also printed on the first line of the LCD. The time values readed before are compared with the boundaries of the time interval to define if the Daily or the Nightly mode must be used. The information about the temperature and about the time are used to show the status coloring of the LED according to the temperature range and the time. The RGB led will be GREEN if the temperature is in the desired range, BLUE if is lower and RED if is greater. With the same criteria we will also manage other devices according to the temperature change. On the second LCD line we show the actual time and the active program. All these datas are updated every half |

| | |
|---|---|
| analogWrite(RED, 128);<br>analogWrite(GREEN, 0);<br>analogWrite(BLUE, 0); }<br>if ((tempC < LTD && isday) \|\| (tempC <<br>LTN && !isday)) { analogWrite(RED, 0);<br>analogWrite(GREEN, 0);<br>analogWrite(BLUE, 128); }<br>if (isday) { snprintf(str, sizeof(str),<br>"%02d:%02d:%02d DM", h, m, s); }<br>else {<br>snprintf(str, sizeof(str),<br>"%02d:%02d:%02d NM", h, m, s); }<br>lcd.setCursor(0, 1);<br>lcd.print(str);<br>delay(500); } | second. |

## 3.3 Working principle of the device

I would like to discuss about more openly how the code in our Arduino IDE works, so let us see some more details.

First of all, we create rtc for the DS3231 RTC module, and address is fixed at 0x68. Then we add the code rtc.begin() to initialize the rtc object.
After getting the user input, we can update the RTC's internal clock by using the function rtc.adjust() from the RTCLib.h library. The rtc.adjust() function receives a parameter with type DataTime which it uses to update the rtc's internal time and date.
The function rtc.now() in our code returns a DateTime data type that contains the current date and time of the rtc. We then assign the data to different variables for additional formatting on the LCD. After assigning the variables, we use the functions lcd.setCursor() and lcd.print() from the LiquidCrystal_I2C.h to position the cursor and to display the text respectively on the LCD. The code below shows how these functions come together to get the rtc time, format the text and display it to the LCD.

Secondly, we need to calculate the temperature using thermistor. We know from the voltage divider circuit, that ouput voltage:

$$Vout= (Vin * Rt) / (R + Rt)$$

So the value of Rt will be:

$$Rt = R (Vin/Vout) – 1$$

Here, Rt will be the resistance of thermistor and R will be 10k ohm resistor. This equation is used for the calculation of thermistor resistance from the measured value of output voltage Vo. We can get the value of Voltage Vout from the ADC value at pin A0 of Arduino as shown in the Arduino Code given below.
Mathematically the thermistor resistance can only be compute with the help of the Stein-Hart equation.

$$T = 1 / (A + Bln(Rt) + Cln (Rt)3 )$$

Where,
T is the temperature in Kelvins.
R is the resistance of the thermistor at T (in Ohms).
A, B and C are the Steinhart–Hart coefficients.

The constant value for the thermistor used in the project is A = 1.009249522×10–3, B = 2.378405444×10–4, C = 2.019202697×10–7. You can either get these constant values directly from the datasheet of the Thermistor.
This equation is used for the calculation of thermistor resistance from the measured value of output voltage Vo. We can get the value of Voltage Vout from the ADC value at pin A0 of Arduino as shown in the Arduino Code given below.

To give the supply to the Arduino you can power it via USB to your laptop or connecting 12v adapter. A LCD and RTC is interfaced with Arduino to display temperature and datetime values and Thermistor is connected as per circuit diagram. The analog pin (A0) is used to check the voltage of thermistor pin at every moment

and after the calculation using Stein-Hart equation through the Arduino code we are able to get the temperature and display it on LCD in the Celsius and Fahrenheit.

## 3.4 I2C protocol

I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line). I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master.

I2C only uses two wires to transmit data between devices:

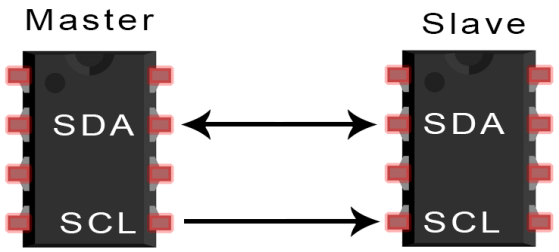


Figure 8: I2C Interface.

SDA (Serial Data) – The line for the master and slave to send and receive data.
SCL (Serial Clock) – The line that carries the clock signal.

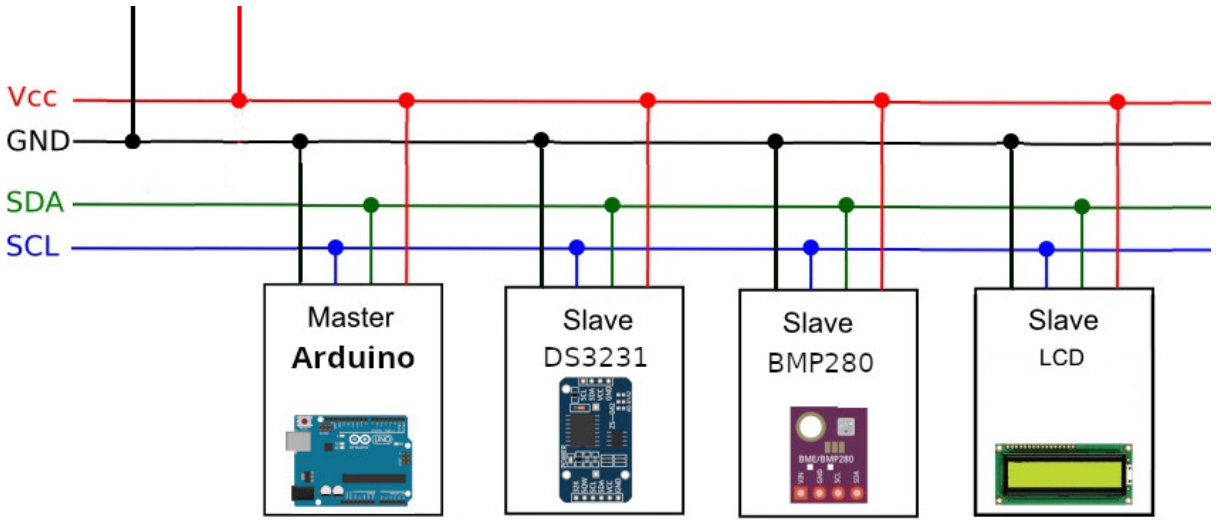


Figure 9: I2C Communication.

## 3.5 Software environment

The entire project was developed using the Arduino IDE 1.8.19. This software makes it possible to program the Arduino microcontroller with a C like language.

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

The main features of the new Arduino Pro IDE are:

- Modern, fully featured development environment
- New Board Manager
- New Library Manager
- Board List
- Basic Auto Completion
- Git Integration
- Serial Monitor
- Dark Mode

# Chapter 4

## Conclusions

This project has been surprisingly successful in accomplishing its main goal despite the modest potency and relatively cheap cost of the components used. The main aim of this work was to design and construct a microcontroller based digital thermometer.  The device has been tested and is working. This project illustrates the use of embedded systems particularly in instrumentation design and generally in the design of electronic devices.
Digital thermometer is an innovation to end the error due to parallax reading in liquid in glass thermometer and also comfort the easy access and accurate reading of temperature. Digital thermometer can further more be advanced into home automations, use in cold rooms, food temperature reserve and so on.

# References


1. https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/
2. https://circuitdigest.com/microcontroller-projects/arduino-thermistor-interfacing-code-circuit
3. https://it.aliexpress.com/item/32232621848.html
4. https://hartmut-waller.info/arduinoblog/i2c/
5. https://www.researchgate.net/publication/329781961_LM35_Based_Digital_Room_Temperature_Meter_A_Simple_Demonstration
6. https://circuitdigest.com/microcontroller-projects/digital-thermometer-using-arduino
7. https://fritzing.org/download/