DC Fan

Temperature regulated DC fan

March 21, 2025

Uppsala University
1TE717, Digitalteknik och elektronik HT2024
Group 1

Authors:

Tuva Björnberg	tuva.bjornberg.5452@student.uu.se
William Mårback	william.marback.7699@student.uu.se
Nora Reneland	nora.reneland.1770@student.uu.se
Matilda Stenbaek	matilda.stenbaek.1554@student.uu.se

Contents

1 Objectives

The goal of this project is to design and implement a temperature monitoring and control system using an Arduino.

Using the system it should be possible to set a desired temperature, ranging between 18°C and 28°C. The user should be able to change this set-temperature while the system is running, i.e. without changing anything in the code.

The actual temperature in the room should be measured using a temperature sensor and should handle temperatures within the expected temperature range, between 15°C and 35°C. The difference between the desired set-temperature and the actual temperature in the room should then be compared in order to decide whether or not to cool down the room using the fan.

The fan should be turned off anytime the set-temperature is higher than the real temperature, since that means that no cooling is needed. If the set-temperature is lower than the temperature recorded by the sensor, meaning the room is too hot, the fan should be turned on. The speed of the fan should vary depending on how much cooling is needed, i.e. the difference between the current- and set-temperature.

The final part of the system is the user interface, which should let the user know whether the system is working to decrease the temperature or if the temperature is acceptable. It should also signal to which extent the cooling is being done, for example signaling when the fan is going at full capacity.

2 Design

The system is composed of four subsystems. The temperature setpoint-system, the temperature sensor-system, the cooling system, and the LED interface. These subsystems were designed and tested individually before being put together. All the subsystems integrated into a complete circuit are illustrated as a circuit diagram in Figure 1 and shown in a photo in Figure 2.

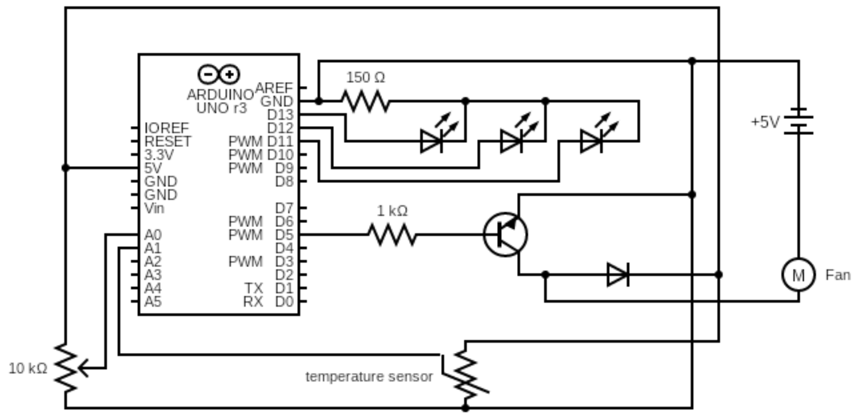

Figure 1: Circuit diagram of the complete system

2.1 Temperature setpoint

The temperature setpoint is set by adjusting a potentiometer. The potentiometer's terminals are connected to the Arduino's voltage source of 5V, the analog pin A0, and ground. The configuration can be seen in the lower-left corner of Figure 1 or in the lower center part of Figure 2. The orange wire connects to Vcc, the green connects to ground and the yellow to the analog pin A0.

The potentiometer will adjust the voltage drop across the first and second terminals. The voltage is then read through the A0 pin by the program and translated to a temperature. This is the setpoint temperature chosen by the user and the temperature which the rest of the system will strive to achieve, by using the fan.

By turning the potentiometer completely to the left, the voltage drop is 5V, which corresponds to a set temperature of 28°C. By turning it gradually to the right, the voltage drop between terminals one and two will decrease and consequently decrease the wanted temperature. The voltage drop is converted into degrees Celsius through the equation:

$$T = v_d p \times \frac{5.0}{1023.0} \times 2 + 18 \tag{1}$$

where *vdp* is the voltage drop from the potentiometer read through the A0

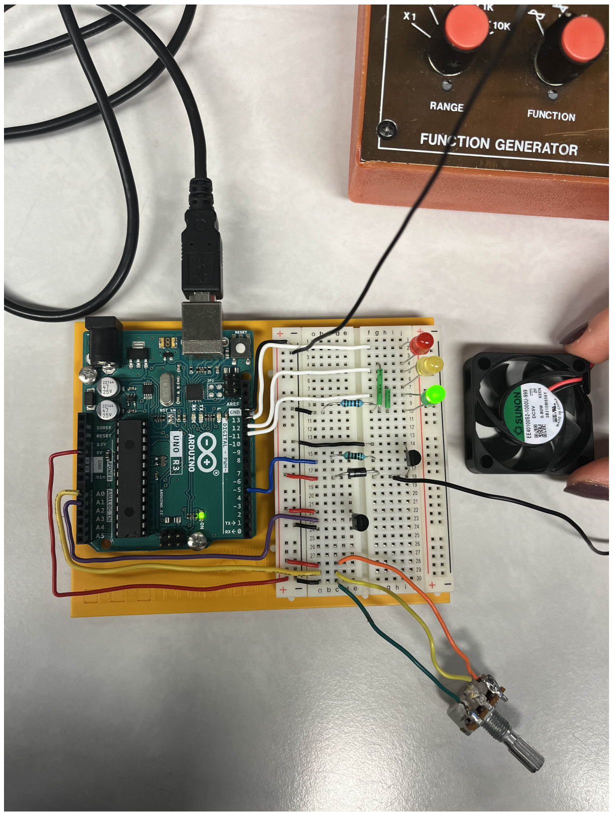

Figure 2: The configuration of all subsystems to the complete system

pin. The voltage read through the analog pin A0 is a raw voltage with a resolution of 1024 readings, from the analog to digital converter [1]. This is why there is a need to modify the resolution into a standard voltage. With the formula in equation 1 the set voltage from the potentiometer is then converted to a temperature between 18°C and 28°C.

The choice of having a potentiometer instead of a voltage divider is to have a more gradual shift between voltages. If the subsystem were to be implemented with a voltage divider, the set temperature would be static. The user would be able to change it by manually switching out resistors, but this method could be seen as impractical. The potentiometer allows for real-time changes of the voltage to the A0 pin without opening the circuit.

2.2 Temperature sensor

The temperature sensor, LM60 from Texas Instruments, can sense temperatures of -40°C to 125°C [2], but will for this project mostly be used around room temperatures.

The temperature sensor's terminals are connected to the 5V voltage source from the Arduino, the analog pin A1, and ground. The configuration can be seen in the lower center in Figure 1 or above the potentiometer subsystem in Figure 2.

The temperature sensor outputs a voltage through terminal 2 that corresponds to the current temperature based on equation 2.

$$V_o = (+6.25mV/°C \times T°C) + 424mV \quad [2] \tag{2}$$

In a room of 25°C the temperature sensor should output a voltage of 0.580V [2]. The higher the drop the sensor outputs, the higher the current temperature it has measured.

The temperature sensor should output voltage at least within the interval $[0.51775V, 0.64275V]$, which corresponds to the objective that a room is usually in the range of 15°C and 35°C. The voltage interval is calculated using equation 2.

Similar to the temperature setpoint-subsystem, the voltage read through the analog pin A1 is a resolution of the voltage. This resolution also needs to be converted to a standard voltage using equation 3, where vts is the voltage out of the temperature sensor.

$$V_o = vts \times \frac{5.0}{1023.0} \tag{3}$$

The voltage V_o from equation 3 is then transformed into a temperature through equation 2.

2.3 Cooling

The cooling subsystem contains different components to ensure that the desired outcome is achieved in an effective and safe manner. The fan itself, a generic DC 5V fan, is in charge of the cooling, but other components are needed to control and operate it.

2.3.1 Fan

The fan is powered using a separate power supply since the one built into the Arduino is not powerful enough to power both the fan as well as the rest of the circuit. When connecting both the fan and the rest of the circuit to the Arduino 5V supply, the current drawn by the fan causes the voltage to drop by approximately 0.2V. This drop is significant enough that it will cause the LEDs to dim, as well as the temperature sensor to present faulty data.

2.3.2 PWM

To regulate the speed of the fan, it is necessary to either adjust the voltage fed to the fan, or the interval at which the power source is active. These two approaches will mostly have the same result and both could work for this project. By directly regulating the voltage fed to the fan, it will run faster when fed higher voltages and slower when fed a lower voltage. Pulse width modulation (PWM) on the other hand, regulates how often the voltage source is active. By having it be active more often, the average voltage over time is higher, registering as a higher voltage by the fan. It follows equation 4, where V_{in} is the peak voltage and D is the duty cycle.

$$V_{avg} = V_{in} \times D \tag{4}$$

PWM will not work for all purposes, but it does for a fan. While the pulse is high, the fan RPM will ramp up. While the pulse is low, the coils in the fan will operate as an inductor, hindering the current from dropping to 0 immediately and thereby smoothing out the fan operation.

In order for the fan to spin, a minimum of 3,5V [3] is needed. As the fan is supplied with 5V, the duty cycle needed for the fan to spin becomes

$$\tfrac{3,5}{5} = 0,7 = 70\%$$

of max. When the analogWrite() is at state 255, the duty cycle is 1, 100%. Likewise, when the analogWrite() is at state 0, the duty cycle is 0, 0%. Thus, the minimum state needed for the fan to spin is

$$0,7 \times 255 = 178,5 \approx 180$$

The implication of this becomes that the usable PWM range for the fan in theory is

$$255 - 180 = 75$$

steps, or 30% of the PWM range available on the Arduino. However, reality will not always correspond perfectly with the theoretical values and could be more flexible. Thus, the usable spectrum could be either larger or smaller in reality. The spectrum was visually noted to be larger, but to what degree is uncertain.

2.3.3 Transistor

A transistor is used as a switch to control whether the fan is on or off, as well as the speed of the fan, using the setpoint-temperature. The base of the transistor receives a signal from terminal 5 in the Arduino. This terminal was chosen due to its PWM frequency of 980Hz [4], which is higher than other terminals. The higher frequency allows for more precise control of the fan speed since the fan will recognize a more even and regular voltage.

The duty cycle of the PWM signal is controlled based on the difference between the setpoint-temperature and the current temperature. If the current temperature exceeds the setpoint-temperature by more than 5°C, the PWM signal will have a duty cycle of 100%. This causes the transistor switch to close and current to flow from the collector to the emitter, turning the fan on at maximum speed.

For a smaller difference in temperature, the duty cycle will linearly decrease depending on the difference, leading to a lower collector-emitter current and a proportionally slower fan speed.

When the set-temperature is equal to or higher than the current temperature, the duty cycle of the PWM signal will be 0%, meaning the switch is opened and the fan is completely off.

The base of the transistor is connected in series with a resistor that controls the current flow into the base, protecting the transistor from excessive current flow. The collector current, I_c, is decided by the current used by the fan and was measured to be 125mA when the motor was connected to a 5V voltage source.

However, since there is a voltage drop over the transistor,

$$V_{CEsat} = 0,4V \text{ [5]}$$

, the voltage to the fan ends up being

$$5 - 0,4 = 4,6V$$

Due to this, the current to the fan becomes approximately

$$\tfrac{4,6}{5} \times 125mA = 115mA$$

The current gain of the transistor, 2N3904, for a collector current in the 100mA range is 30 [5], making the equation for the base current, I_b, equation 5.

$$I_b = \frac{I_c}{h_{fe}} = \frac{115mA}{30} \approx 3,8mA \tag{5}$$

Using KVL, the required resistance can therefore be calculated using the base current (I_b), the input voltage (V_{in}), and the base-emitter voltage (V_{BE}), as seen in equation 6.

$$R_b = \frac{v_{in} - v_{BE}}{I_b} = \frac{5 - 0,7}{3,8mA} \approx 1,1k\Omega \tag{6}$$

2.3.4 Diode

A diode is used in parallel with the fan to prevent damaging voltage spikes. Since the fan acts as an inductive load in the circuit, the current through the fan cannot change instantaneously. When the transistor switch is opened, the fan will therefore cause big voltage spikes in an attempt to resist the change in current.

These spikes in voltage could possibly damage the transistor and for that reason they have to be avoided. This is done using a diode that is placed in parallel with the fan. The diode is placed in such a way that it is reverse-biased when the switch is closed, so as to not lead current away from the fan. Conversely, when the switch is opened, the diode will be forward-biased, leading the current away and letting it dissipate safely.

2.4 LED interface

The system provides a simple-to-read visual interface of the relationship between the desired and current temperature, as well as the status of the fan.

It consists of three LEDs; one green, one yellow, and one red. The green LED signals when the desired temperature is met, or when the current temperature is lower than the desired temperature since the fan is only able to cool. The yellow LED is lit when the current temperature doesn't exceed the desired temperature by more than 5°C. If the current temperature does exceed the set-temperature by more than 5°C, the red LED is lit. These three states reflect the states the DC fan operates on.

The green light means that the fan is off and the setpoint-temperature is met, the yellow light means the fan is on, but not on full force, and finally, the red light means that the fan is operating at maximum capacity.

To operate the LEDs, the current must be regulated. The LEDs all have roughly the same forward voltage $V_f \approx 2.0V$ at a current of $20mA$. For this reason, they can all share a common resistor limiting the current to around $20mA$. Using Ohm's law, seen in equation 7, as well as the source and LED voltages from the KVL equation 8, the value of the resistor required by the LEDs can be calculated in equation 9.

$$R = \frac{V}{I} \quad [6] \tag{7}$$

$$V = V_s - V_d \Rightarrow \begin{cases} V_s = 5V \\ V_d = 2V \end{cases} \tag{8}$$

$$R = \frac{V_s - V_d}{I} = \frac{5-2}{20 \times 10^{-3}} = \frac{3}{20 \times 10^{-3}} = 150\Omega. \tag{9}$$

Since the resistor value is shared across all LEDs, the resistor is connected in series with the common ground of the LEDs. This gives the same result as if the three LEDs had their own resistor connected to the positive lead.

3 The program

Other than the hardware, a program is needed to control the logic of the system and make the necessary calculations, before outputting the controlling signals.

Firstly, an interface between the output PWM pins is created for the LEDs and the fan. The variables are set globally, at lines 1-4 in listing A, and then set up as output pins, at lines 8-11. When the system is running, the output pins will send signals out to the circuit with the help of digitalWrite() [7], for the LEDs, and analogWrite() [4], for the fan. The digitalWrite(), seen at lines 29-45 in listing A, will send out either a 1 or 0, corresponding to a 5V or 0V signal, to turn on or off the LEDs. The analogWrite(), which is shown on lines 33, 39, and 47 in listing A, sends out signals with a regulated voltage for the fan to vary its speed.

As described throughout, the voltages from the different sensor inputs, i.e. the potentiometer and temperature sensor, are transformed into °C. The difference in temperature is the main driver of changing outputs. If there is no difference in temperature there is no need to use the fan, nor change the LED values from green, as seen in lines 42-48 in listing A. If there is a change in the setpoint temperature and actual temperature then the fan logic is run and can be seen in lines 26-40 in listing A.

4 Process

All work for this project was done during sessions with all group members present. By everyone working together, we were able to both problem-solve the current issues and multitask, i.e. someone could make adjustments to the code while others changed the hardware, all while everyone was aware of what was being done. The roles of editing code and testing hardware were also shifted around during the process.

The work was divided into the different subsystems described above; the temperature setpoint system, the temperature sensor system, the cooling system, and the LED interface. The subsystems were designed, configured, and tested one at a time. When all subsystems worked as expected on their own, they were integrated with each other and tested.

Ultimately, the hardware and code for the project were completed by all members of the group, and the group collectively wrote the report.

5 Result

The project has produced a functional system for temperature monitoring and control of a DC-fan, using an Arduino. When the user set temperature is lower than the actual temperature, a DC fan will run at variable speed until it has cooled the current temperature to the desired.

The circuit is simple, easy to analyze, and troubleshoot because of the neatly organized subsystem setup, as seen in Figure 2.

References

[1] "Analogread()." (), [Online]. Available: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/ (visited on 10/09/2024).

[2] T. Instruments. "Lm60 2.7-v, sot-23 or to-92 temperature sensor." (), [Online]. Available: https://www.ti.com/document-viewer/lm60/datasheet (visited on 2017).

[3] "Ee40100s2-1000u-999." (), [Online]. Available: https://www.sunon.com/PROSEARCH_FILES/(D04121200G-01)-1.pdf (visited on 10/09/2024).

[4] "Analogwrite()." (), [Online]. Available: https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/ (visited on 10/09/2024).

[5] "2n3903 - general purpose transistors." (), [Online]. Available: https://www.onsemi.com/download/data-sheet/pdf/2n3903-d.pdf (visited on 10/09/2024).

[6] P. Scherz and S. Monk, *Practical Electronics For Inventors*, 4th ed. United States of America: McGraw-Hill Education, 2016, ISBN: 978-1-25-958754-2.

[7] Arduino. "Digitalwrite()." (), [Online]. Available: https://docs.arduino.cc/language-reference/en/functions/digital-io/digitalwrite/ (visited on 05/15/2024).

A Arduino code

```
int redLed = 13;
int yellowLed = 12;
int greenLed = 11;
int fan = 5;

void setup() {
  Serial.begin(9600);
  pinMode(redLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(fan, OUTPUT);
}

void loop() {
  int setpointVoltage = analogRead(A0);
  int rawVoltageTmpSensor = analogRead(A1);

  float setTemp = setpointVoltage * (5.0 / 1023.0) * 2 + 18; //Degrees celsius, set through pot. meter

  float voltageOutTmpSensor = rawVoltageTmpSensor * (5.0 / 1023.0);
  float tempSensorNow = (voltageOutTmpSensor - 0.424)/0.00625; //OBS! Be wary of bottow view of temp sensor, function from tmp sensor datasheet

  float tempDiff = tempSensorNow-setTemp;
  Serial.print("Tempdiff: ");
  Serial.println(tempDiff);
  if (tempDiff > 0) {

    if (tempDiff > 5) { //Actual temp more than 5 degrees from set temp
      digitalWrite(redLed, HIGH);
      digitalWrite(yellowLed, LOW);
      digitalWrite(greenLed, LOW);

      analogWrite(fan, 255); //Run at full speed
    } else {
      digitalWrite(redLed, LOW);
      digitalWrite(yellowLed, HIGH);
      digitalWrite(greenLed, LOW);

      analogWrite(fan, (255/5)*(tempDiff));
    }

  } else {
    digitalWrite(redLed, LOW);
    digitalWrite(yellowLed, LOW);
    digitalWrite(greenLed, HIGH);

    analogWrite(fan, 0); //No speed
  }

  Serial.print(setTemp);
  Serial.println(" grader POTENTIOMETER");

  Serial.print(voltageOutTmpSensor);
  Serial.println(" volt TMP SENSOR");

  Serial.print(tempSensorNow);
  Serial.println(" grader nu!");

  delay(1000);
}
```