



Computer Engineering
Micro-controllers – 10636426

Assignment #2: Heat/Cool Temperature Control – Fall 2024

Instructor: Dr. Raed Alqadi

Academic Year: 2024/2025

Semester: Fall

Credit Hours: 3

Date: Oct. 19

Student Name:

Registration Number:

Section: : 1

Assignment Grade: 15

Assignment Weight: 7.5%

Evaluation Part	Description	Points	CLO	Grade
Part 1	Coding Style	4	I	
Part 2	Correct code and Output	5	I	
Part 3	Discussion and Demo	6	I	
Student Grade				

Notes

1. *Submit every Part of Hardware or Software On Time*
2. *Use good programming practices and style. .*
3. *Read the specs of the Assignment/project on the next page carefully.*

Assignment #2

Heat/Cool Temperature Control

Proportional Control- Fall 2024

Dr. Raed Alqadi

In this assignment, you will design and Control a Heater and Cooler in order to Control the heat of a Plant or a house by using Pulse Width Modulation and Proportional Control as Follows.

Analog Input 0 (AI0): It is the (Pot.P1), It will be used to set the **Set Point**. It is a Potentiometer of 10K Ohm. Scale the value read to 0 – 100C.

We will Refer to it as **SP (Set Point Temperature)**.

Analog Input 0 (AI1): It is the (Pot. P2), It will be used to Represent the Outside Temperature **Outside Temperature**. It is a Potentiometer of 10K Ohm. Scale the value read to 0 – 100C.

We will Refer to it as **OT (Outside Temperature)**.

Analog Input 2 (AI2): This is already connected to a simulator that reads a voltage already connected to a sensor. Read the Voltage (0 – 5) then scale it to temperature by just multiplying it by 100 since an increase of 10mv represent an increase by 1C. The minimum Value that it reads is around 27 degrees without doing extra changes. We will leave it as is to simplify things and we will control Temperatures above the minimum Value. The Value of this sensor is the actual temperature of the room.

We shall Refer to it as **T (Temperature) or RT (Room Temperature)**.

Heater: This is already connected to **RC5**. We will turn Control the RC5 by generating a Pulse width Modulated Signal. We will not use an actual PWM signal but we by use Time3 Interrupt to generate a Pulse Width Modulation

Cooler (Fan): This is already connected to **RC2** which is the CCP1. We will use actual Pulse width Modulation to control the amount of Cooling.

Hysteresis (HS): This is a Value that we should set Digitally through Interrupt INT1 at RB1 (Decrement) and INT2 at RB2 (Increment). The HS should between **0 – 3**. INT2 will increment HS until it reaches **3** and will stay at **3**. INT1 will decrement HS until it reaches 0 and will stay at 0. Hysteresis has meaning only when in the Temperature Control Modes. INT1 and INT2 will be used to increment/decrement different values depending on the Operating mode in a similar to the description of setting the HS. **The Hysteresis can only be changed in this mode.**

Cooling and Heating Amount: In Control Temperature mode, the amount of Heating or Cooling will be proportional to the difference between the Actual Temperature and the Set Point as explained in the Modes below.

Operation Modes: This is a Value that we should set Digitally through Interrupt INT0 at RB0. There shall be **3 operation Modes plus the Off Mode**. main mode that we can select (Mode 1: Heat Mode, Mode 2: Cool Mode and Mode 3: Auto Heat-Cool Mode) as described below. INT0 will circulate through the Modes (You can use 3 values e.g. (1 – 3) and circulate through them. Rollover when the value exceeds 3.

Mode 3 is the Auto Heat Mode which we shall refer to as Auto HC mode. So, there will be 3 modes which are: Mode 1: **Heat Mode**, Mode 2: **Cool Mode** and Mode 3: **Auto HC Mode**.

The details of the modes are explained below.

The System Shall be turned **OFF Heater OFF(0%) and Cooler OF(0%)** if the user Presses **RB3** which shall be read in the main Loop. This is Mode 0 below.

4. **Mode 0 (OFF):** The system shall start in this mode on Power-up. Once the user presses INT0 then we shall go to one of the three operational Modes(**Heat, Cool, Auto**). Use **RB3**. If the user presses this button which is read in the main loop, then turn the system OFF.
5. **Mode 1 (Heat):** In this mode, the Heater can be set to 20 levels of Heating. **The Cooler is always Off**. The amount of heating is set by the Heat percentage and should be between 0 –100%. The amount is incremented/decremented by a step of 5%. To implement this, use a **Percent Heat Counter** that you will increment/decrement by 1 (value 0—20). Each increment/decrement corresponds to 5%. INT2 will increment the **Percent Heat Counter** by 1 until it reaches 10 and will stay at 10. INT1 will decrement **Percent Heat Counter** until it reaches 0 and will stay at 0. To implement a Simulated Pulse Width Modulation, use Timer 3 and make it cause an Interrupt every **20ms**. For example, If the desired heating percentage is 60%, generate a pulse width signal of $TON = 240ms$ (12×20) and a Period of $400ms$ (20×20). The RC5 should be ON for **Percent Heat Counter=12** interrupts. And should be OFF for the Next 4 Interrupts. Use an Interrupt Counter on every interrupt and reset it to zero when it reaches 20. Compare the Interrupt to the **Percent Heat Counter** and set the RC5 as explained.

Notice that: **Heating Percentage** = $5 \times \text{Percent Heat Counter}$; ✓

In this mode, the display screen should be as shown below and the RC5 is as shown on the scope.

Actual Temperature (T).
Shown as RT.

Set Point

Actual Temperature (OT).
Shown as OT.

Clock (MHz)

4

Debug

LCD

hd44780 16x4

Pot. P1

Pot. P2

Heater

Cooler

Temp: 58.31°C

RT: 58.2C

SP: 50.4C

OT: 54.5C

MD: Heat

H C

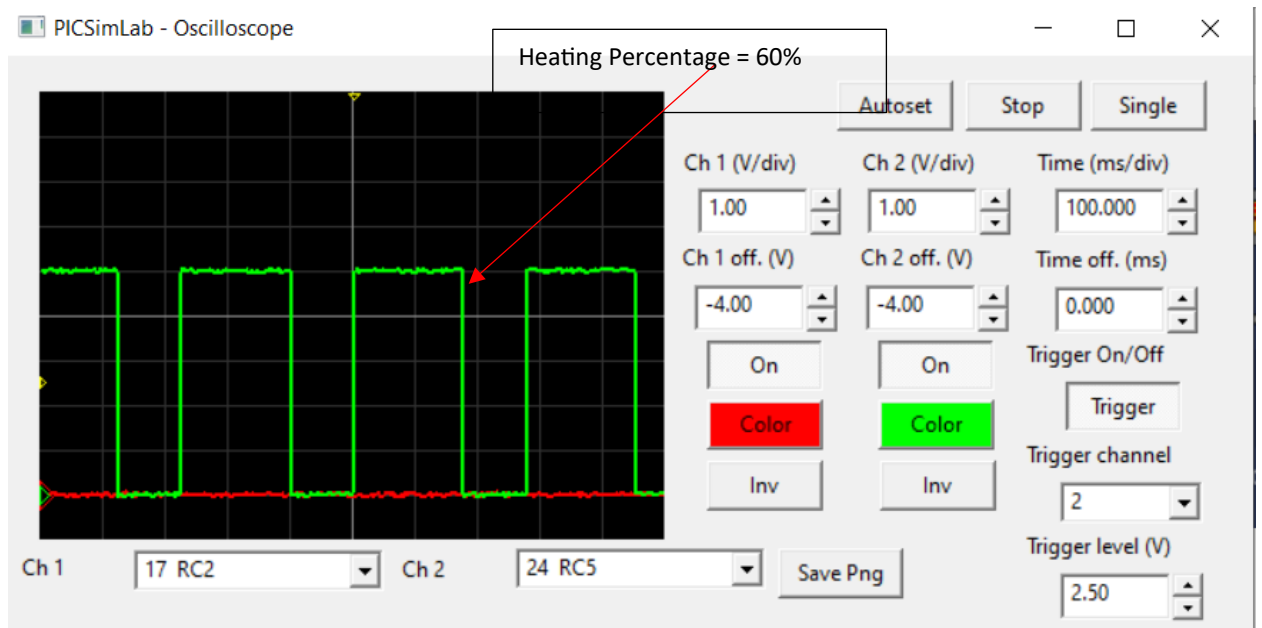
Y N

H: 60%

Operation Mode change with RB0

Increment

Decrement



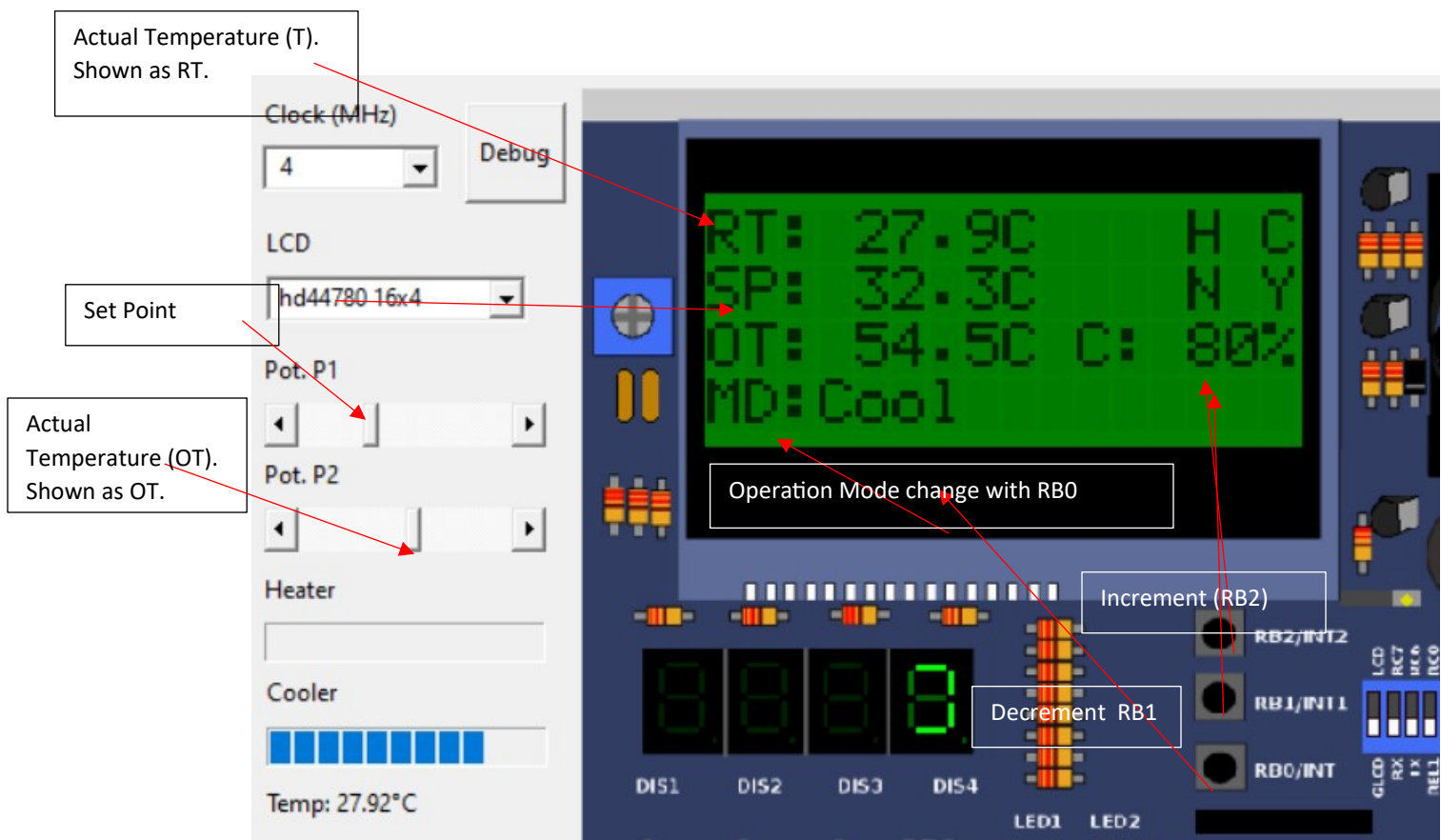
6. **Mode 2 (Cool):** In this mode, the Cooler can be set to 10 levels of Cooling. The Heater is always Off. The amount of Cooling is set by the Cool percentage should be between 0 – 100%. The amount is incremented/decremented by a step of 5%. This is simply implemented by the set_pwm1_percent() function to the Value selected. The Speed of the fan should be proportional to the pulse width modulation Value selected.

INT2 will increment Percentage by 5% until it reaches 100% and will stay at 100%. INT1 will decrement Percentage by 5% until it reaches 0 and will stay at 0.

The Heater should be off in this Mode.

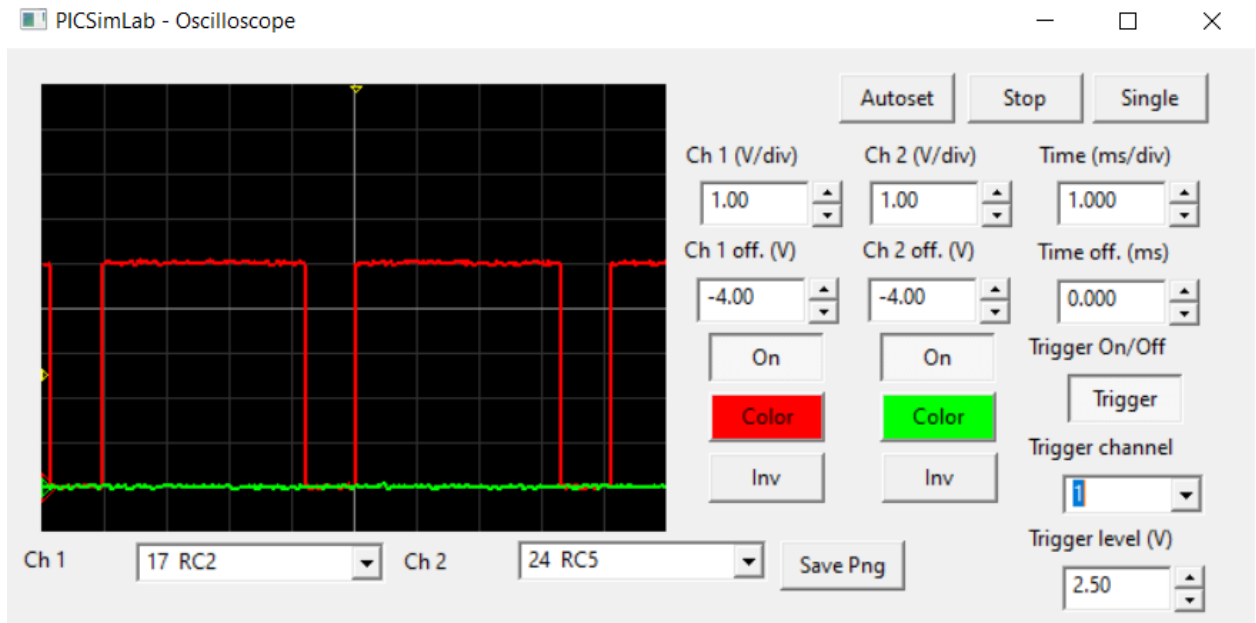
In this mode, the display screen should be as shown below:

Note : this screen is from an old version of the Simulator: P1, P2 are just AN0, AN1)



The Oscilloscope should display the followings in the Mode:

Pay attention to the Settings. RC2 is PWM1 and is set by using set_pwm_1()



7. Auto Mode: Heat/Cool (Display as **Auto HC**):

In this Mode, the Temperature will be automatically Controlled to become Almost Equal to the Set Point. In other words, we bring it to the Set Point to within a Hysteresis(H) Value (A small value, we will restrict to a Max of 4 as explained in Page 1 it can be from (0 --5). This Means that the Difference between the actual Temperature and the Setpoint should be very small (No more than the H Value).

In **winter we should Heat** and In **Summer we should Cool**. We should Not Heat and Cool at the Same Time. So, we will use AIN1 (Outdoor Temperature) to Simulate Summer or Winter. However, we also will Allow Heating and Cooling in some cases as shown below.

In this mode, display the hysteresis on Line %, it can be incremented, decremented in this node only using INT1, INT2.

Let OT be Outdoor Temperature,

```
#define WINTER_T 40
```

```
#define SUMMER_T 60
```

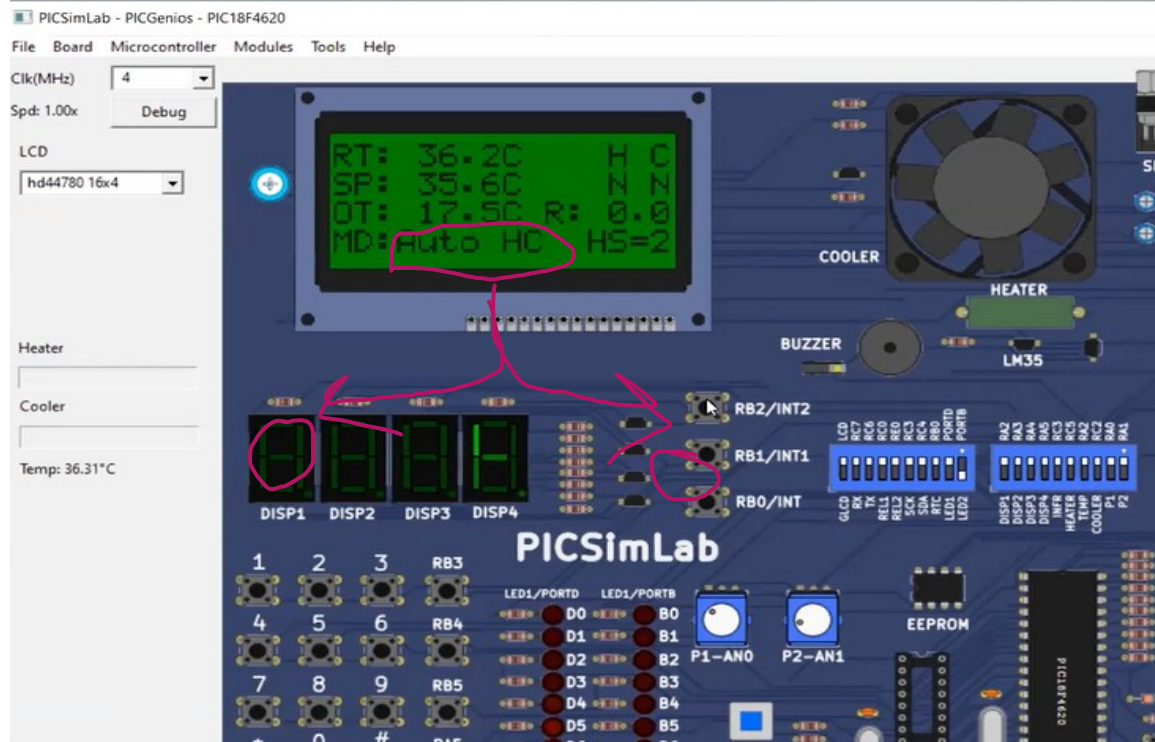
If (OT > SUMMER_T) Control Cooling as described below in **Auto Cool**. See **a.** below

Else If (OT < WINTER_T) Control Heating as described below in **Auto Heat**. See **b.** below

Else Do Not Heat Or Cool turn Both Heater and Cooler Off

Example Screen for **Auto HC** mode

(Displaying the **R** which is the RPS(Fan Speed0 is optional)



- a. **Auto Cool**: In this mode the Temperature should be controlled such that the Actual Temperature should be around the Set Point referred to as **SP (SetPoint)**. The **Cooler** should be controlled by the Pulse Width Modulation signal CCP1 at RC2. The Value of the PWM percentage should be proportional to the **CoolError = T – SP** if $T > SP$ and should be turned Off if $T < (SP - HS)$. The percentage of Cooling (PWM percentage value) should be set as follows.

$$\text{CoolError} = T - SP$$

If(CoolError > 0)

$$\text{PWM percentage value} = \text{CError} * 100 / 10;$$

Set the CCP1 PWM to PWM percentage value

If this value is less than 25% set it to 25%

If($T < (SP - HS)$)

PWM percentage value = 0 ;// Cooler OFF

In real life, the Heater should be always off but here we are forced to make it **50% ON** when T fall below $(SP - HS)$, that is $(T < (SP - HS))$ *because we cannot increase the temperature if we do not do that. in real life, in summer, the temperature rises when we turn the cooler off, we cannot do that here.*

In this Mode, **line 3** should display the **Actual Fan Speed**. To measure the Fan speed, use Timer 1 and Timer 0 as we did in the [pwm asl.x example](#).

After Sometime The Actual temperature will be within H degrees(In this case 2C) from the Setpoint. (Displaying the Fan Speed is Optional, you Can choose to Not display it)

- b. Auto Mode Heat(Control Heat Mode)** (Display as **Auto HT**): In this mode the Temperature should be controlled such that the Actual Temperature should be around the Set Point referred to as **SP (SetPoint)**. The **Heater** should be controlled by the Pulse width Modulation generated as described in Mode 1 by using Timer 3 Interrupt. The Value of the PWM percentage should be proportional to the **HeatError = SP - T** if **SP > T** and should be turned Off if **T > (SP+HS)**. The percentage of Heating (PWM percentage value) should be set as follows. In Other words, set the **Percent Heat Counter** to be proportional to the Heat Error. The **Percent Heat Counter** is described in **Mode 1**

HeatError = TSP - T

If(HeatError > 0){

Percent Heat Counter = 2*HeatError ; //for 10 C difference do 100% heat
If this Value is greater than 20 set it to 20

// The percentage is Percent Heat Counter * 5;

Do not change the code for Interrupt 3, Percent Heat Counter is set by the Heat Error in this case not the Increment / Decrement buttons.

If (Percent Heat Counter < 10) Percent Heat Counter = 10; //min

heat

if(SetPoint > 52) Percent Heat Counter to 10 ;// Max value ..special

case

}

If(T > (SP +HS)) turn the Heater off


The Cooler should be off in this Mode (Control Heat Mode))

Serial Interface:

We will add a very simple Serial Interface to get the Values that we show in the display:

Use the Serial RX interrupt to read a character, if the user sends the character 'M': Write the Values to the Serial Terminal. Write the values of **RM, OT, SP, Cooler ON/OFF, Heater On or OFF, Cooling Percentage, Heating Percentage.**

RM — OT — SP — C —



For Example RM: 35, OT= 30, H: ON ..etc. Put every Two Values in a line (use "\r\n").

If the user enters the letter 'S' then put the System in the OFF mode.

Important: Read the character using the RX Interrupt. Bot do not send serial data during the Interrupt. Set a Variable(flag) in the Interrupt and write to the serial in the main Loop. See what I did for sending the data in the [pwm_asl.x example](#) however the receiving the character will be should be in the Interrupt.

Notes:

1. The temperature should be automatically updated as well as HS and the Mode if changed.
2. The main program should an infinite loop and there should be noticeable flicker in the screen. It should be continuously reading the Analog Inputs and displaying them. You can put a small delay if needed.
3. The Interrupt code should be minimal, just change the Variables) and do actual displaying in the Main Program.
4. If your program is correct in Auto HC mode, the Temperature can start with any temperature and it will go the HSP or CSP, respectively.
5. Keep your set points less than 60 . There are special cases for the Heater, since this is a simulator. You cannot make the Temperature more than 76 and cannot be less than 27.
6. Make sure when you enable your Interrupts to Enable the Most Two significant bits bit 7 and 6 in INTCON. (GIEH, and GIEL) Otherwise your Timer 3 interrupt will not work correctly. Disable the Priority.
7. Work in Groups of 2 and start as soon as possible.