

# CS0010 Introduction to Computing for Systems Engineers

## Project 3

### Overview

In this project, we will build a thermostat. A thermostat is a device that controls a heater, an air conditioner, and a fan so as to keep the temperature within a certain range. Since we don't have a heater, an air conditioner, or a fan, we'll turn on a red LED to represent turning on the heat, a blue LED to represent turning on the air conditioner, and a green LED to represent turning on the fan.

Here are the basic rules governing the operation of our thermostat:

- If the temperature is below the desired range, turn on the heater and the fan.
- If the temperature is above the desired range, turn on the air conditioner and the fan.
- If the temperature is within the desired range, turn off any of the air conditioner, heater, and fan that are on.

We'll also have two buttons to change the desired temperature range. Pressing the first will shift both endpoints of the desired temperature range up by one degree. For example, if the current range is between 20 and 25 degrees Celsius, pressing this button will shift the desired range to between 21 and 26 degrees Celsius. Pressing the second button will similarly shift both endpoints of the range down by a degree.

Note that the heater and the air conditioner should never be on at the same time. Also, whenever either the heater or the air conditioner is on, the fan should also be on.

### Wiring

First, we'll work on the wiring for this project. *Do not remove the wiring and devices you used for Project 1.* The projects for CS0010 are designed such that you will not need to remove any wiring from past projects, allowing you to run all of the programs you have written so far throughout the term without re-wiring.

With Pi Cobbler plugged in to d1-20 and h1-20 (GPIO pin 21 on h20):

- black wire from a20 (GND on the Pi Cobbler @d20) to Neg rail
- 560 Ohm resistor (Green, Blue, Brown, Gold) from Neg rail to a30

- red LED short leg (cathode) to e30 (to resistor/Neg rail), long leg (anode) to f30 (to red wire/GPIO pin 21)
- red wire from j20 (GPIO pin 21 @h20) to j30
- 560 Ohm resistor (Green, Blue, Brown, Gold) from Neg rail to a35
- blue LED short leg (cathode) to e35 (to resistor/Neg rail), long leg (anode) to f35 (to blue wire/GPIO pin 12)
- blue wire from j16 (GPIO pin 12 @h16) to j35
- 560 Ohm resistor (Green, Blue, Brown, Gold) from Neg rail to a40
- green LED short leg (cathode) to e40 (to resistor/Neg rail), long leg (anode) to f40 (to green wire/GPIO pin 25)
- green wire from j11 (GPIO pin 25 @h11) to j40
- purple wire from a1 (3.3V power @d1) to Pos rail
- BME280 e45-e51 (VIN @e45, CS @e51)
- purple wire from Pos rail to a45 (VIN @e45)
- black wire from Neg rail to a47 (GND @e47)
- orange wire from a48 (SCK @e48) to a3 (SCL @d3)
- yellow wire from a50 (SDI @e50) to a2 (SDA @d2)
- white wire from j3 (GND on the Pi Cobbler @h3) to other Neg Rail
- button with pins in e55, e57, h55, h57
- red wire from j9 (GPIO pin 24 @h9) to j55
- grey wire from Neg rail to j57
- button with pins in e60, e62, h60, h62
- blue wire from j8 (GPIO pin 23 @h8) to j60
- grey wire from Neg rail to j62

## Part 1: Design a thermostat class

First, we will design a class `Thermostat` to represent our thermostat. Write this class in a file named `thermostat.py`.

Your class should include the following attributes:

- Desired temperature range (represented by the 2 endpoints of the range)

- An object for each of the red, blue, and green LEDs. To do this, recall that you will need to use the LED class in the `gpiozero` module. You can import this class by including this line at the top of your file:

```
from gpiozero import LED
```

Recall that you can then obtain an object for the red LED (connected to GPIO pin 21) using the following code:

```
red_led = LED(21)
```

The methods `red_led.on()` and `red_led.off()` turn the LED on and off, respectively. The attribute `red_led.is_lit` is `True` if the corresponding LED is lit and is `False` otherwise.

You can create objects for the blue LED (connected to GPIO pin 12) and green LED (connected to GPIO pin 25) in the same way; they will have the same attributes and methods.

- An object for each of the two buttons. To do this, you will need to use the `Button` class in the `gpiozero` module. You can import this class by including this line at the top of your file:

```
from gpiozero import Button
```

You can then obtain an object for the up button (connected to GPIO pin 24) using the following code:

```
up_button = Button(24)
```

The `Button` class includes the attribute `is_pressed`, a boolean that is `True` if the button is currently being pressed and `False` otherwise. For example, if the button tied to GPIO pin 24 was being pressed, `up_button.is_pressed` would have value `True`.

While the `Button` class also includes the method `wait_for_press` which halts the execution of your program until the button is pressed and then returns. Note that this could be problematic for our project. Eventually, we will want our thermostat to be constantly checking for both button presses and changes in temperature. To accomplish this, we will not be able to stop executing the program until a button is pressed. For this project, you will want to use `is_pressed`.

The object, methods, and attributes for the down button, connected to GPIO pin 23, will be similar.

Write the following methods:

- `__init__` should create the attributes listed above. You should start with a desired temperature range of 20 to 25 degrees Celsius.

- `__str__` should return a string giving the upper and lower endpoints of the desired temperature range. For example, if the temperature range 20 to 25 degrees Celsius, the `__str__` method should return the following as a string:

Thermostat set to 20-25 degrees C

- `up_temp` should increase both endpoints of the desired temperature range by 1.
- `down_temp` should decrease both endpoints of the desired temperature range by 1.
- `check_btns` if either button is currently being pressed, it should adjust the desired temperature range accordingly. Recall that pressing the up button should shift both endpoints of the desired temperature range up by one degree, while pressing the down button should shift both endpoints of the desired temperature range down by one degree.
- `fan_on` should turn the fan LED (green) on, if it is not already on.
- `fan_off` should turn the fan LED (green) off, if it is already on.
- `ac_on` should turn the AC LED (blue) on, if it is not already on.
- `ac_off` should turn the AC LED (blue) off, if it is already on.
- `heat_on` should turn the heat LED (red) on, if it is not already on.
- `heat_off` should turn the heat LED (red) off, if it is already on.

When writing `ac_on`, `ac_off`, `heat_on`, and `heat_off`, keep in mind that if the AC or heat is turned on, the fan must be turned on as well (and, similarly, if the heat or AC is turned off, the fan should be turned off). These functions will need to call `fan_on` and `fan_off`.

Write a test `main` function at the bottom of `thermostat.py` in which you create an instance of your class and test each of these methods. This function should only be run when `thermostat.py` is explicitly executed, not when it is imported.

## Part 2: Add temperature reading

Next, we will add temperature reading to our Thermostat class. Recall that we will need to import the `board`, `busio`, and `adafruit_bme280` modules to do this. As in Project 2, you will be able to create an object representing the sensor in this way:

```
i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_bme280.Adafruit_BME280_I2C(i2c)
```

You will then be able to access the current temperature with `sensor.temperature`. Make the following changes to your `Thermostat` class:

1. First, in the `__init__` function, add an attribute `sensor` to the class whose value is the sensor object above.

2. Next, update the `__str__` method to display the current temperature as well as the desired temperature range. Now this method should return a string in the following format:

```
Will turn on AC at or above 25 C
Will turn on heat at or below 20 C
Current temperature: 22 C
```

3. Write a method named `check_temp` that reads the current temperature. If the temperature is below the desired range, the method should turn on the heat if it isn't already on. If the temperature is above the desired range, the method should turn in the air conditioning if it isn't already on. If the temperature is within the desired range and if one of the heat or air conditioner is on, it should be turned off.

This method should return `True` if either the heater or the air conditioner is turned on or off. If nothing is turned on or off, it should return `False`.

Note that you will need some way of keeping track of whether or not the air conditioner, the heater, and the fan are currently on or off.

4. Update the `check_btns` function to return `True` if a button has been pressed and the desired temperature range has been changed. If nothing changes, return `False`.

Finally, you should modify the `main` function in `thermostat.py` to simply start running the thermostat. First, it should create an object of class `Thermostat`. It should then run `check_btns` and `check_temp` every 0.2 seconds and continuing forever (Hint: Use the `sleep` function from the `time` module). Any time there is any change (increase or decrease in desired temperature range, heat turns on or off, air conditioner turns on or off) print the `Thermostat` object to the screen.

Again, make sure that if the heater is on, the fan is on. Make sure that if the air conditioner is on, the fan is on. Make sure the heater and air conditioner are never on at the same time.

Be sure to make liberal use of comments to clarify the code that you are writing. You will be graded on the readability of your code (including commenting!).

Once you have completed all parts of the lab, be sure to show your work to the lab instructor.

## Rubric

Your program will be evaluated according to the following rubric:

Red LED tied to heater and works properly	10
Blue LED tied to AC and works properly	10
Green LED tied to Fan and works properly	10
Up botton works as specified	10
Down botton works as specified	10
<code>check_temp</code> works as specified	10
<code>check_btns</code> works as sepcified	15
<code>main</code> written as specified	15
Code is well laid-out and commented	20