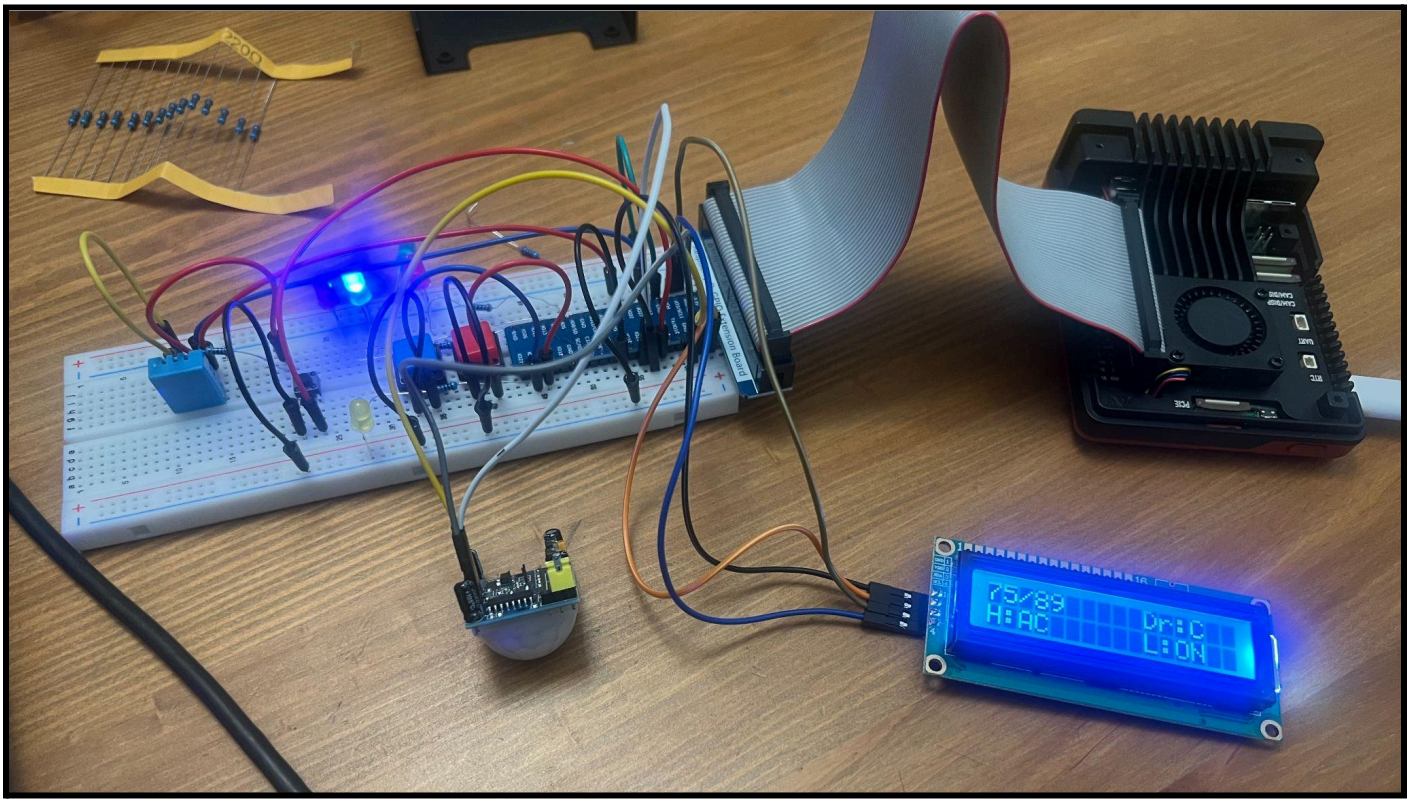
Miles Jennings
June 11, 2025

Building HVAC System



Approach:

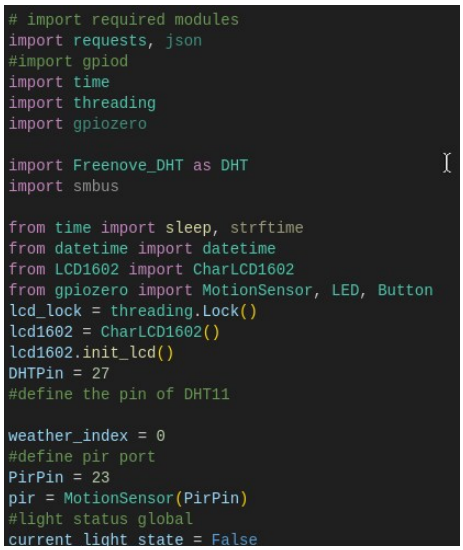
Much like many projects I have done, my approach consisted of 4 things.

1. Understanding the problem/collecting data
2. Proposing a solution
3. Implement the solution
4. Follow up with the solution

I followed my usual approach by first understanding and reading the instructions in the assignment. I then came up with a plan since I now knew what components I needed from the assignment instructions: Get the components together → start connecting to breadboard/gpio ports → code the logic. I then implemented the solution. However, I ran into a few issues as finding the correct manual that had instructions on using the components was difficult. After I imported all of the needed libraries/code, I was able to look into them to find the functions I would need to call to use the components. This took a while because I had to decode what each function did. However, after that, all I had to do was test and troubleshoot. Submitting this report is my follow up as I will see if I did it correctly based on the grade.

The layout of my code is as follows:

1. Library/Module imports and setup:



```
# import required modules
import requests, json
#import gpiod
import time
import threading
import gpiozero

import Freenove_DHT as DHT
import smbus

from time import sleep, strftime
from datetime import datetime
from LCD1602 import CharLCD1602
from gpiozero import MotionSensor, LED, Button
lcd_lock = threading.Lock()
lcd1602 = CharLCD1602()
lcd1602.init_lcd()
DHTPin = 27
#define the pin of DHT11

weather_index = 0
#define pir port
PirPin = 23
pir = MotionSensor(PirPin)
#light status global
current_light_state = False
```

2. Hardware component initialization:

```
#define ports of LEDs and Buttons using dictionaries
LEDS = {
    'HEATER':LED(5),   #reg LED - RED - HEATER - on if weather
    'AC':LED(6),   #reg LED - BLUE - AC - on if weather index >=
    'GREEN':LED(26),  #reg LED - GREEN - Light on/off when moti
    'FIRE_ALARM':LED(21),  #reg LED - YELLOW - Fire Alarm (1 se
}

BUTTONS = {
    'inc': Button(16, bounce_time=0.3),  #red - increase tempera
    'dec': Button(20, bounce_time=0.3),   #blue - decrease temp
    'security': Button(18)  #changes door/window status
}
```

3. Functions Used in the main function (Hvac control, light control, fire alarm, etc)

```
def hvac_ctrl(weather_index):
    global hvac_status
    global hvac_prev
    global D_W_open
    global desired_temp

    #prevent invalid states at start
    if weather_index is None:
        return
    #if door/window is open, turn off HVAC
    if D_W_open:
        LEDS['HEATER'].off()
        LEDS['AC'].off()
        hvac_status = "OFF"
    else:
        #AC on if temp is hotter than desired
        if weather_index >= desired_temp + 3:
            LEDS['HEATER'].off()
            LEDS['AC'].on()
            hvac_status = "AC"
            #log_events("HVAC AC")
        #heater on if colder than desired
        elif weather_index <= desired_temp - 3:
            LEDS['HEATER'].on()
            LEDS['AC'].off()
            hvac_status = "HEAT"
            #log_events("HVAC HEAT")
        #if within 3 degrees, turn it off
        else:
            LEDS['HEATER'].off()
            LEDS['AC'].off()
            hvac_status = "OFF"
            log_events("HVAC OFF")
    #announce on LCD if change happens
    if hvac_status != hvac_prev:
        if hvac_status == "AC":
```

4. Main function that controls flow of program and calls the hardware functions

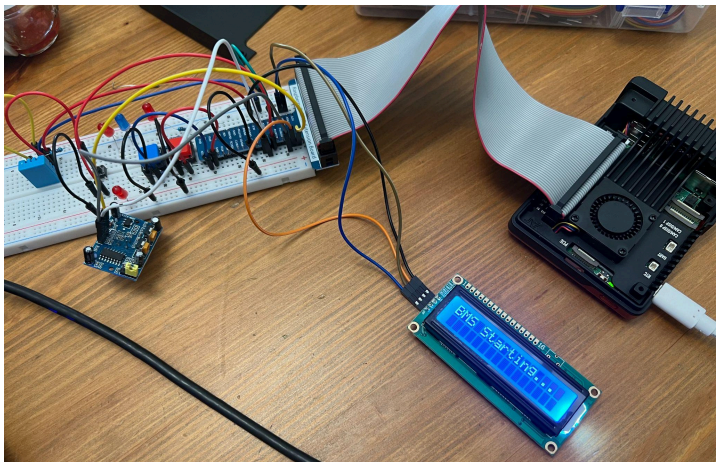
Below are the results

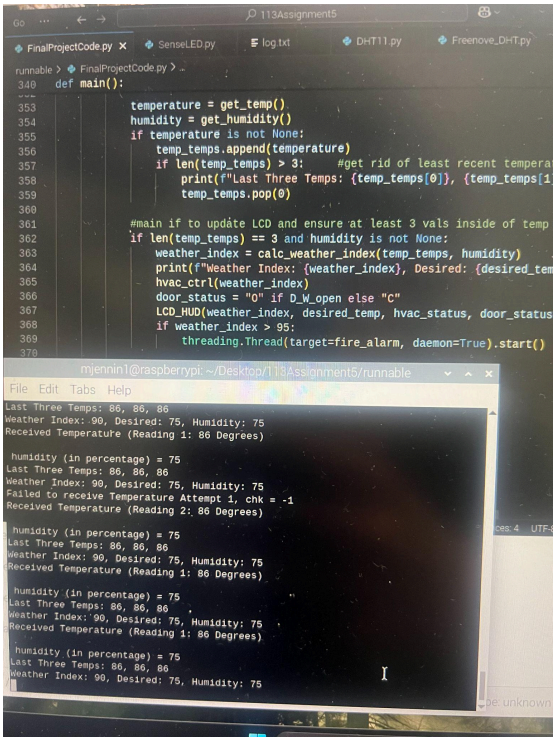
```python
def main():
    LCD_announce("BMS Starting...")
    global current_light_state
    global fire_active
    global weather_index
    temp_temps = [] #temp list to store last 3 temps
    threading.Thread(target=light_ctrl, daemon=True).start()
    print("Motion Detection System has started. Lights will turn on if detected.")
    #init leds
    for led in LEDS.values():
        led.off()
    try:
        while True:

            temperature = get_temp()
            humidity = get_humidity()
            if temperature is not None:
                temp_temps.append(temperature)
                if len(temp_temps) > 3:     #get rid of least recent temperature
                    print(f"Last Three Temps: {temp_temps[0]}, {temp_temps[1]}, {temp_temps[2]}")
                    temp_temps.pop(0)

            #main if to update LCD and ensure at least 3 vals inside of temp list
            if len(temp_temps) == 3 and humidity is not None:
                weather_index = calc_weather_index(temp_temps, humidity)
                print(f"Weather Index: {weather_index}, Desired: {desired_temp}, Humidity: {humidity}")
                hvac_ctrl(weather_index)
                door_status = "O" if D_W_open else "C"
                LCD_HUD(weather_index, desired_temp, hvac_status, door_status, "ON" if current_light_state
                if weather_index > 95 and not fire_active:
                    threading.Thread(target=fire_alarm, daemon=True).start()


            else:
                print("Calculating Weather. . .")
```

Results:

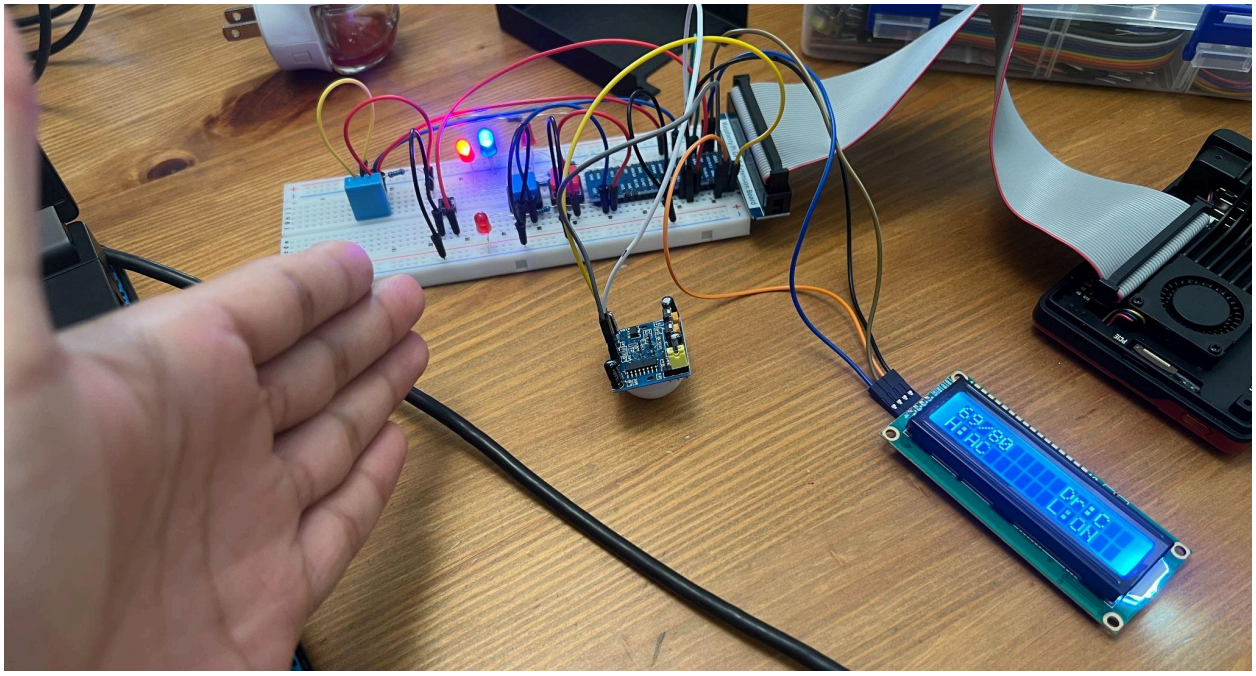
During the running of the code, log.txt updates with All of the events that occured. When program is exited, Log.txt is wiped clean.

Code compiles and program starts:

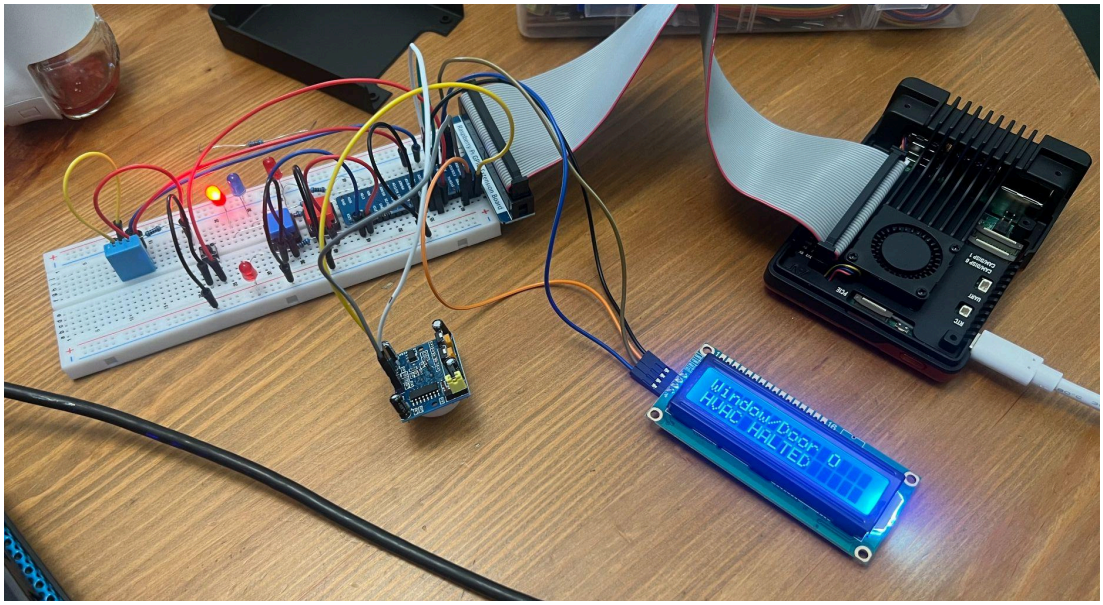




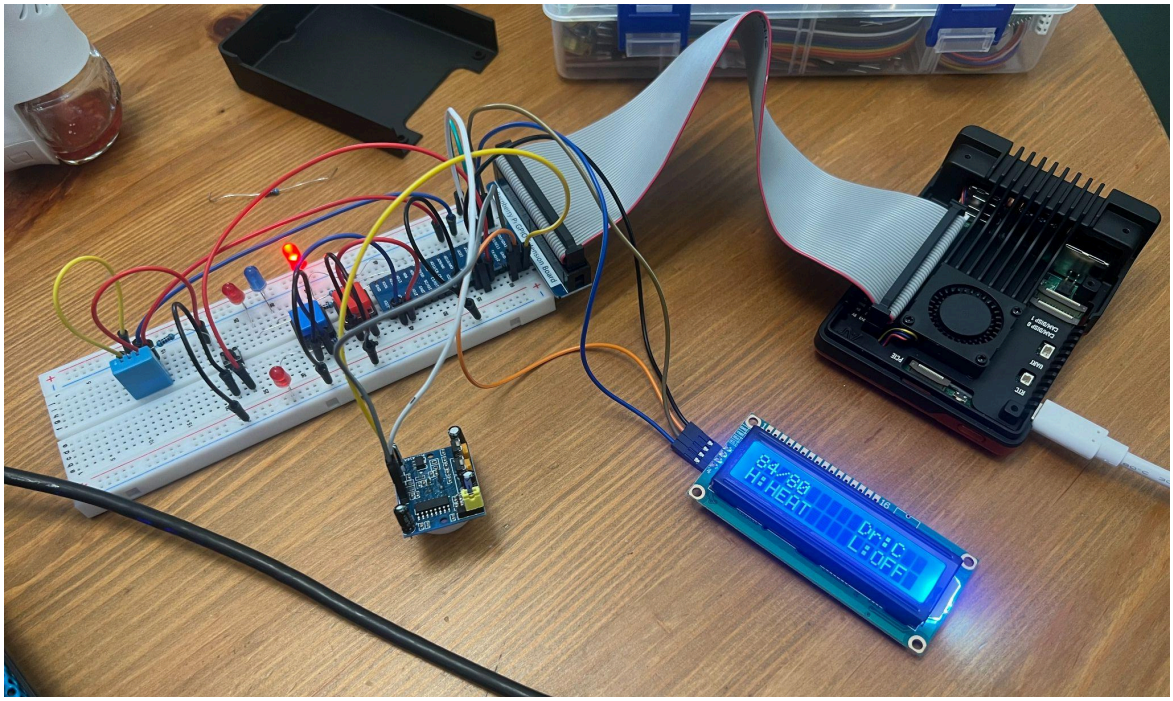
Motion detection leads to light on and desired temp + 3 < weather index leads to AC:



Door Opened leads to HVAC halting and display Open:

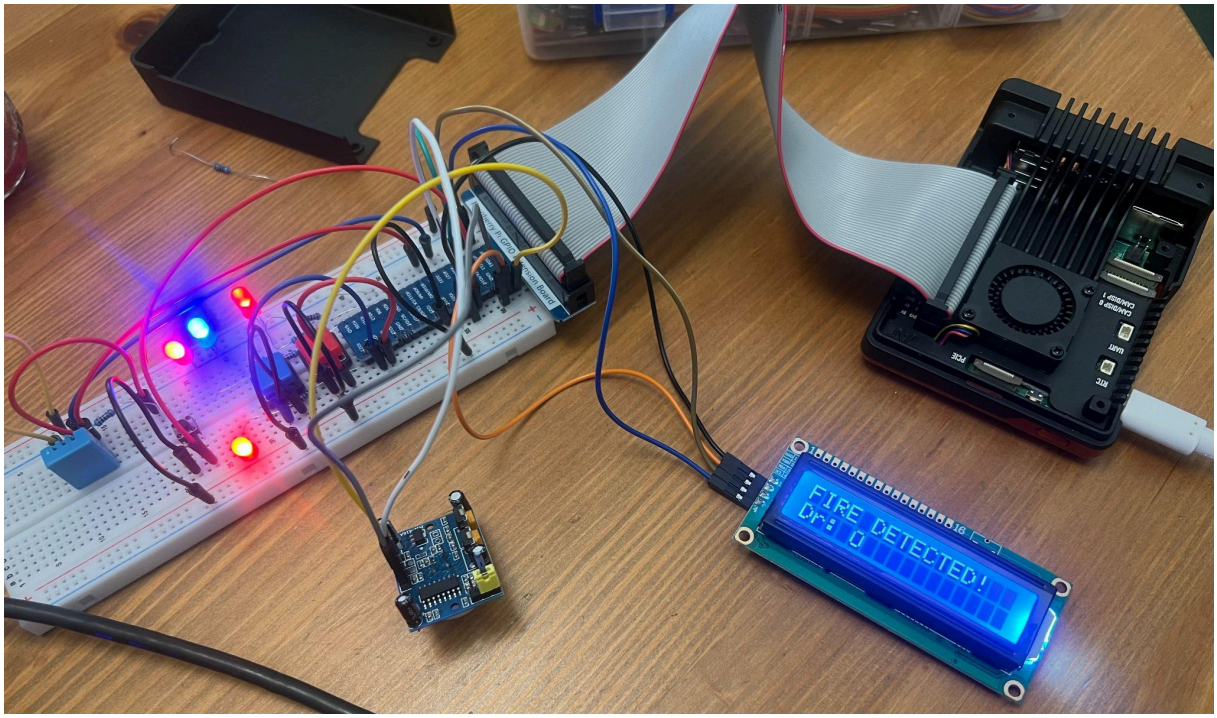


Door closed and desired temp >= weather index

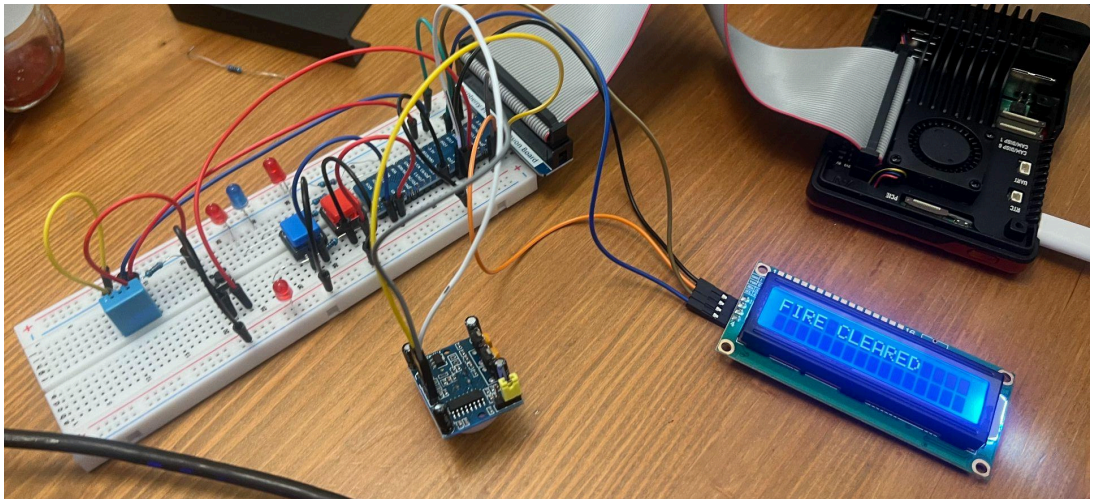


Weather index > 95 leads to fire alarm activated, all lights blink while LCD displays emergency message and door status:

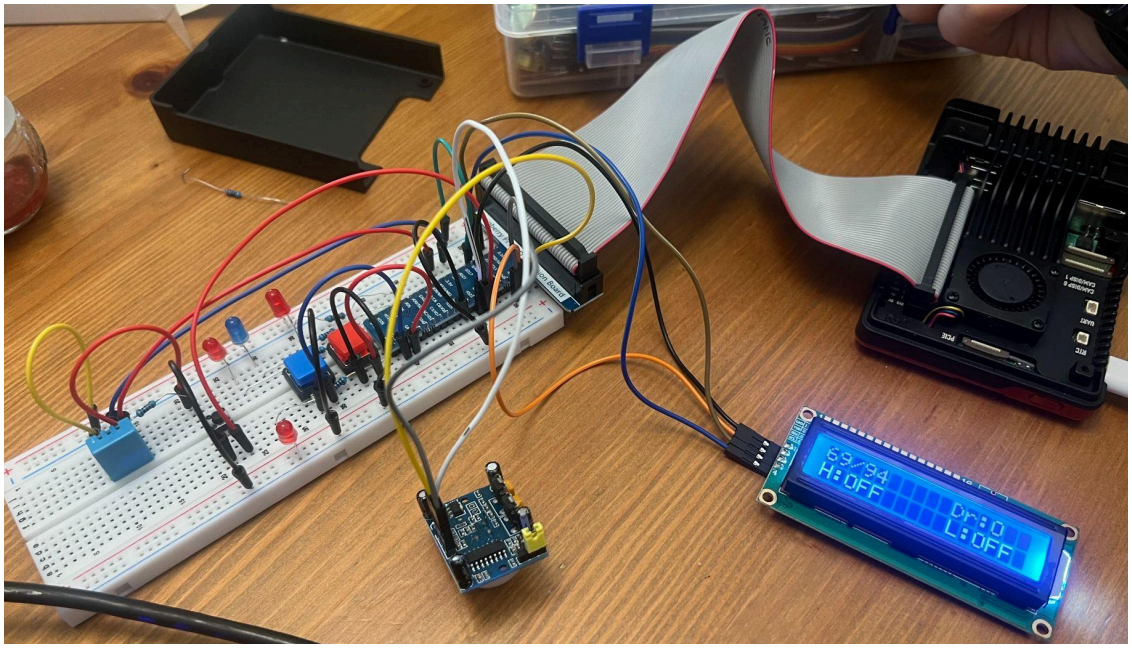
1. Fire detected → display emergency message and door status



2. Fire cleared as weather index gets back to 95 or below



3. Door is still open and temperature back below 95



Issues I ran into:

- Fried my DHT11
  - → Ordered a whole new one on amazon to finish
- Motion Sensor was not detecting and turning on LED
  - Had to replace a resistor and change the sensitivity on the PIR sensor
- Faulty Resistors
  - Some LEDs just weren't lighting up but after replacing resistors, they worked
- LCD Display would display gibberish after certain button sequences
  - Had to clear and do changes to the spacing of the words being displayed
  - Needed to add in mutex locks since there are multiple threads trying to write to the LCD screen simultaneously
  - (seen when using "with lcd_lock)