IoT Experimental Learning
Week 1 Journal

1. **Describe the experience and what you hope to gain from participating in the experience.**
    - During this week I was introduced to the Raspberry Pi and the IoT extension, GrovePi+. This period allowed me to initialize the Raspberry Pi and mate it with the GrovePi+ IoT extension board. My goal of this exercise was to understand the physical and logical aspects of the Raspberry Pi hardware platform, as well as the GrovePi+ extension board. This was accomplished through the IoT week 1 assignment 'Temperature and Humidity Sensors'. After some initialization issues I was able to successfully initialize and interface with the Temperature and Humidity sensor through the Raspberry Pi GPIO interface and the GrovePi+ board.
    - It has been a while since I have had an opportunity to engage with the Python software language, this gave me the opportunity to refresh myself with the Python.
    - I was also able to engage with other classmates through discussions to help resolve the issues I was experiencing.

2. **Provide an overview of tasks and key activities (training, discussions, labs, assessments, etc.) in which you were engaged during the week.**

    For week 1 I accomplished the following tasks in chronological order;

❖ **Monday January 20, 2020**, I began by unboxing my Raspberry Pi 4 and reading all supplied instructions. Upon completion, I began the task of initializing my Raspberry Pi with the supplied 32G micro SD card which contained the Noobs interface. I installed noobs and then the Raspian OS (code named Buster). Once installed I began familiarizing myself with the base UI interface and its base contents. I noticed that it was based on the Debian platform.

- **I** performed an internet search and found the OS home page at https://www.raspbian.org/. I then browsed the documentation in order to understand its basic concepts and features.
- I then went to the IoT Week 1 Project instructions at https://learn.snhu.edu/content/enforced/343560-IT-697-X3325-OL-TRAD-GR.20TW3/IoT%20Project%201%20Directions.pdf and began the week 1 project.
- I then began this by installing and initializing the GrovePi+ software at http://www.dexterindustries.com/GrovePi/get-started-with-the-grovepi/setting-software/
    - This was, according the documentation, accomplished successfully based on the functional tests on steps 9 and 10 of the instructions.

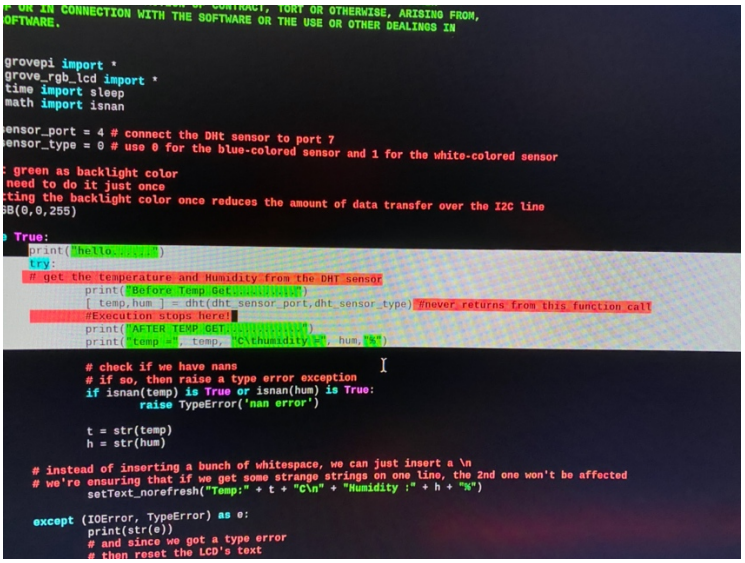
- I then installed the Temperature and Humidity sensor proved as part of the GrovePi+ Starter set and ran the Home_Weather_Display.py python code as instructed and came up with the following observations and conclusions;
  - The call to [temp,hum] = dht(dht_sensor_port,dht_sensor_type) in the python script blocks there and never actually returns any values for temp, hum. From that point it does not continue to execute, and the program hangs until forced to close. Upon termination of the program, the LCD continues to to stay back lit. I am able to set and change the LCD Display back lighting from green to blue and this works just fine.
  - I came to the conclusion that I needed to dive deeper into this issue and posted my 1st day in the week 1 discussion board.

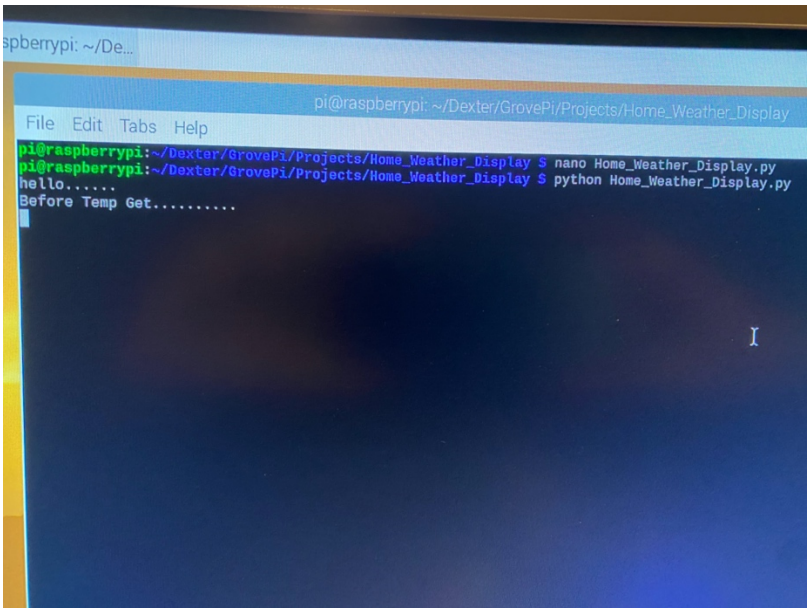
- **Tuesday January 21, 2020**, performed additional research online for the issues encountered on Monday, but nothing really was identifiable as a possible solution presented itself.
  - I then downloaded and installed PyCharm IDE on my Raspberry Pi. I choose this because of its fully integrated debugger.
  - I ran the Home_Weather_Display.py in PyCharm and attached the debugger to the main thread and had the following observations and conclusions

- Here is an image of the RGB LED with power applied, no running code, notice the black squares. They will remain during code execution.

- Python code for the Home_Weather_Display project has been executed, notice that the back lighting has been changed to Blue but the black squares are still present. The expected result should be a reading of the Temperature and Humidity received from the sensor.

- The following two images are associated with the same code execution from above, these illustrate what the code should be performing and what is actually not being performed. Notice on the second image that the code seems to stall after printing 'Before temp Get…'.



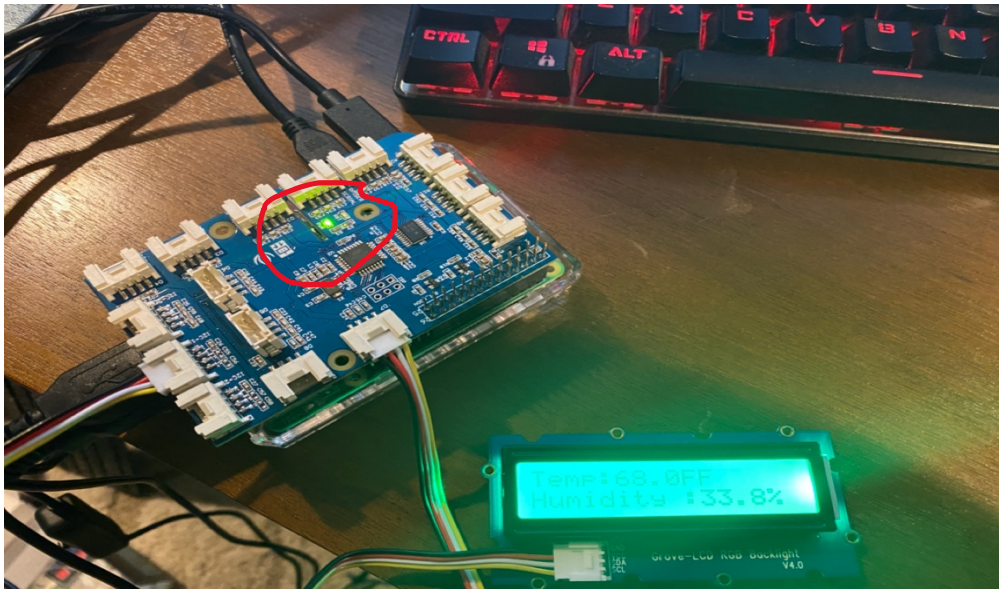


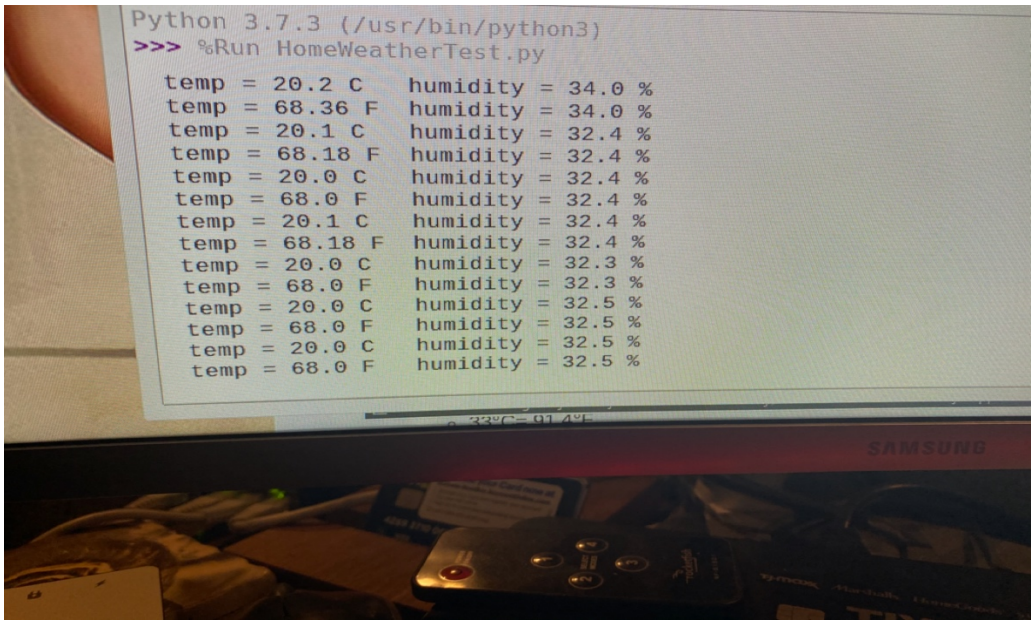

```
            counter += 1
            time.sleep(0.003)

    return data

def read_identified_i2c_block(read_command_id, no_bytes):  read_command_id:
    data = [-1]  data: [23]
    while data[0] != read_command_id[0]:
        data = read_i2c_block(no_bytes + 1)

    return data[1:]

# Arduino Digital Read
def digitalRead(pin):
    write_i2c_block(dRead_cmd + [pin, unused, unused])
    data = read_identified_i2c_block( dRead_cmd, no_bytes = 1)[0]
    return data

# Arduino Digital Write
def digitalWrite(pin, value):
    write_i2c_block(dWrite_cmd + [pin, value, unused])
    read_i2c_block(no_bytes = 1)
```

- While executing in the debugger, I noticed that the code is stalled in an infinite loop at the above function while reading from the I2C Block. It will never return from this.
- I then posted an update to my original post with my finding to the week 1 discussion board.

➢ **Wednesday, January 22, 2020**, read the post from Mr. Martin on his success for the same issue in which he provided a resource list which I researched fully and produced a plan of action against.

- **Thursday, January 23, 2020,** I began by following Mr. Martins resource on firmware upgrade at https://www.dexterindustries.com/GrovePi/get-started-with-the-grovepi/updating-firmware/
  - I re-flashed the GrovePi+ firmware based on the above instructions and re initialized the Raspberry Pi and GrovePi+ extension board.
  - I restarted the Home_Weather_Display.py python code, but I was still experiencing the same issue. I then noticed on the GrovePi+ board two small LEDs one green and the other red. My gut reaction was that red was bad. I did some research and was not really able to find a single source that identified the 'red' LED or what it denoted. I was able to ascertain that the 'red' LED of an indication of a bus fault.
    - I then removed power from the Raspberry Pi and removed the GrovePi+ board.
    - I applied power to the Raspberry Pi, this time without the GrovePi+ board attached.
    - I then, with power applied to the Raspberry Pi, installed the GrovePi+ board to the GPIO connecter.
      - Observed that the 'red' LED was no longer lit, and the only LED was that of the 'green'.
    - I then ran the Home_Weather_Display.py script from and found that it executed correctly.
    - So it seems that there were two issue, firmware was out dated and the bus fault. After doing a little more reading I did find that all say not to install the GrovePi+ board until after the Raspberry Pi has fully completed boot and power on.





- Program output to console from Home_Weather_Display.py

- Here is the final code with updated to include;
  - Temperature conversion from Celsius to Fahrenheit
  - LED back-lighting; RED if > 68 or Blue if lower.

Home_Weather_Display.py

```
from grovepi import *
from grove_rgb_lcd import *
from time import sleep
from math import isnan

dht_sensor_port = 7 # connect the DHt sensor to port 7
dht_sensor_type = 1 # use 0 for the blue-colored sensor and 1 for the white-colored sensor

# set the default back lighting to BLUE
setRGB(0, 0, 255)

# clear the LED Screen
setText_norefresh(" ")

while True:
    try:
        # get the temperature and Humidity from the DHT sensor
```

```
    [ temp,hum ] = dht(dht_sensor_port,dht_sensor_type)
    print("temp =", temp, "C\thumidity =", hum,"%")

    # check if we have nans
    # if so, then raise a type error exception
    if isnan(temp) is True or isnan(hum) is True:
        raise TypeError('nan error')

        # Convert the aquired temperature to Fahrenheit
    tempf = (temp * 1.8) + 32

        # Print the converted temperature and humidity
        # to the shell/console
    print('temp =', tempf, 'F\thumidity =', hum, '%')

        # If the temperature in Fahrenheit is greater than 68
        # change the LeD back lighting to RED
        # and print this condition
    if tempf > 68:
        print('Temp is RED.....')
        setRGB(255,0,0)
        pass
    else:
            # If its not greater then 68
            # set the backlighting to BLUE and
            # print color condition to console
        print('Temp is BLUE')
        setRGB(0,255.0)

    t = str(temp)
    h = str(hum)

    # set temp string to Fahrenheit value
    tf = str(tempf)

    # setText_norefresh("Temp:" + tf + "F\n" + "Humidity:" + h + "%")

        # allowing the LED to refresh on writes
    setText("Temp:" + tf + "F\n" + "Humidity:" + h + "%")

except (IOError, TypeError) as e:
    print(str(e))
    # and since we got a type error
    # then reset the LCD's text
```

```
except KeyboardInterrupt as e:
    print(str(e))
    # since we're exiting the program
    # it's better to leave the LCD with a blank text
    break

# wait some time before re-updating the LCD
sleep(2))
```