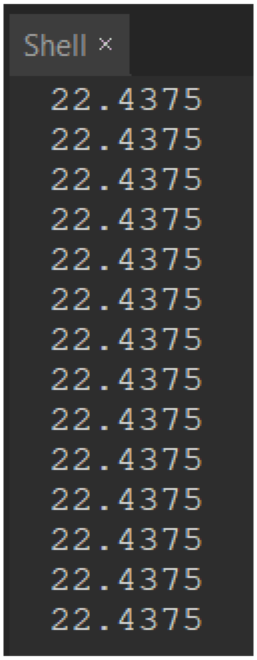
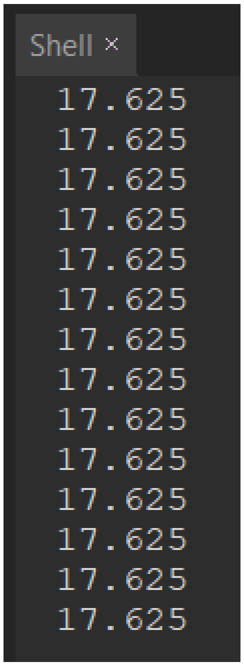
1.



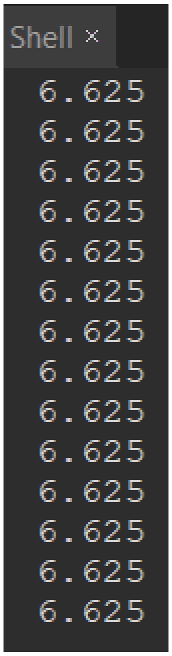
```
22.4375
22.4375
22.4375
22.4375
22.4375
22.4375
22.4375
22.4375
22.4375
22.4375
22.4375
22.4375
22.4375
22.4375
```

Ambient Air Temp



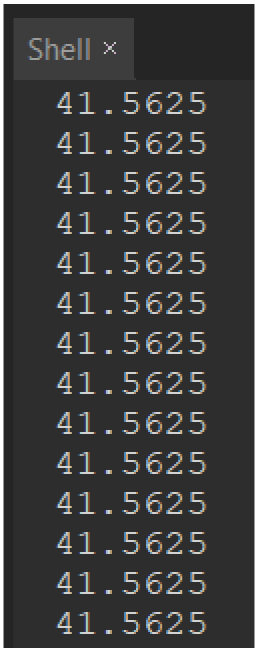
```
17.625
17.625
17.625
17.625
17.625
17.625
17.625
17.625
17.625
17.625
17.625
17.625
17.625
17.625
```

Cold Tap Water



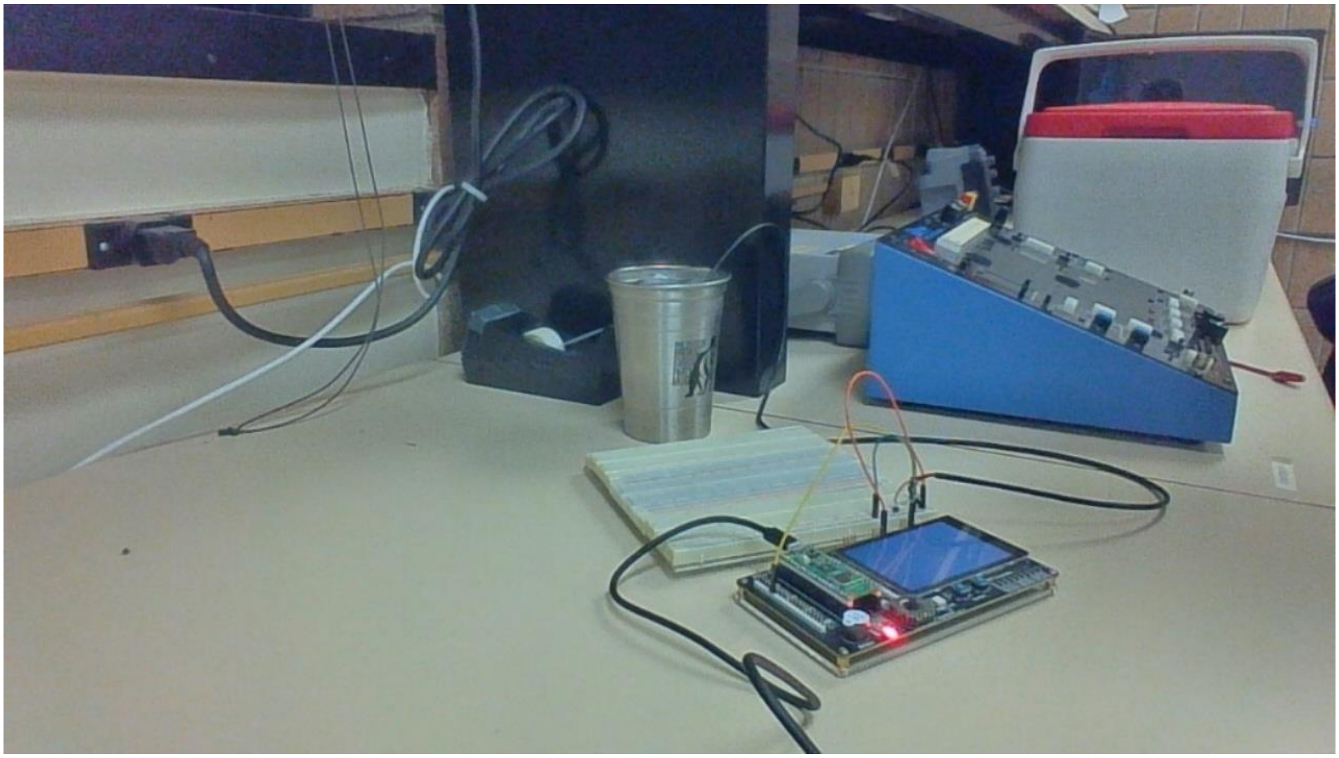
```
6.625
6.625
6.625
6.625
6.625
6.625
6.625
6.625
6.625
6.625
6.625
6.625
6.625
6.625
```

Ice Water

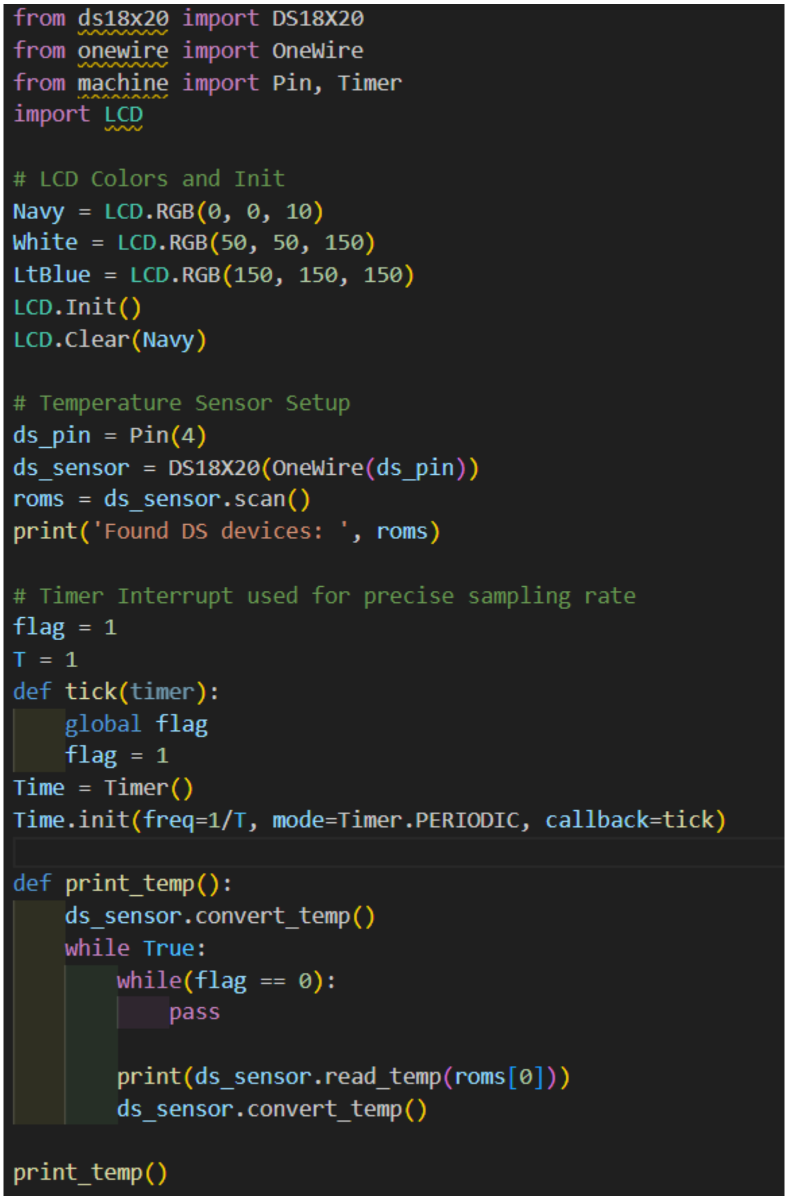


```
41.5625
41.5625
41.5625
41.5625
41.5625
41.5625
41.5625
41.5625
41.5625
41.5625
41.5625
41.5625
41.5625
41.5625
```

Hot Tap Water



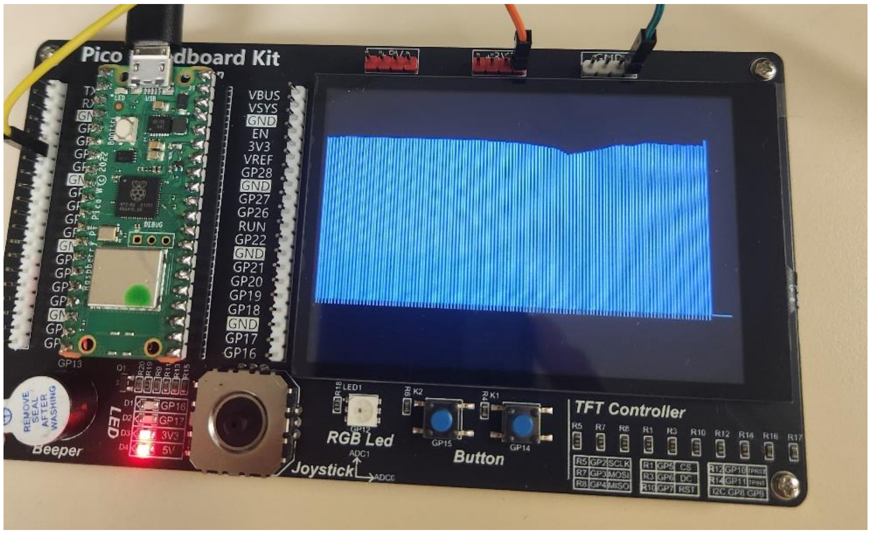
Picture of Setup

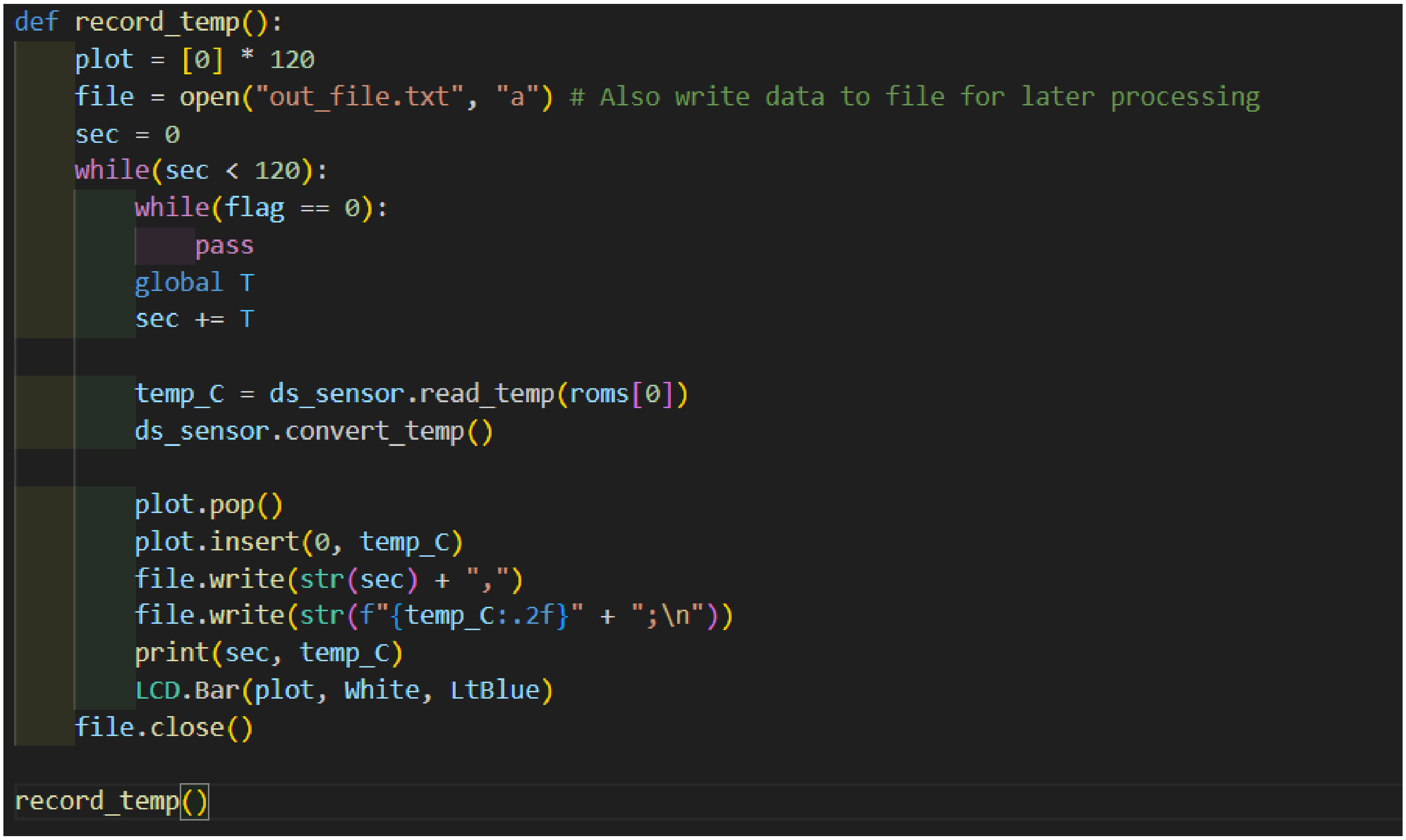


```python
from ds18x20 import DS18X20
from onewire import OneWire
from machine import Pin, Timer
import LCD

# LCD Colors and Init
Navy = LCD.RGB(0, 0, 10)
White = LCD.RGB(50, 50, 150)
LtBlue = LCD.RGB(150, 150, 150)
LCD.Init()
LCD.Clear(Navy)

# Temperature Sensor Setup
ds_pin = Pin(4)
ds_sensor = DS18X20(OneWire(ds_pin))
roms = ds_sensor.scan()
print('Found DS devices: ', roms)

# Timer Interrupt used for precise sampling rate
flag = 1
T = 1
def tick(timer):
    global flag
    flag = 1
Time = Timer()
Time.init(freq=1/T, mode=Timer.PERIODIC, callback=tick)

def print_temp():
    ds_sensor.convert_temp()
    while True:
        while(flag == 0):
            pass

        print(ds_sensor.read_temp(roms[0]))
        ds_sensor.convert_temp()

print_temp()
```

Code for #1 (hw_7a.py)

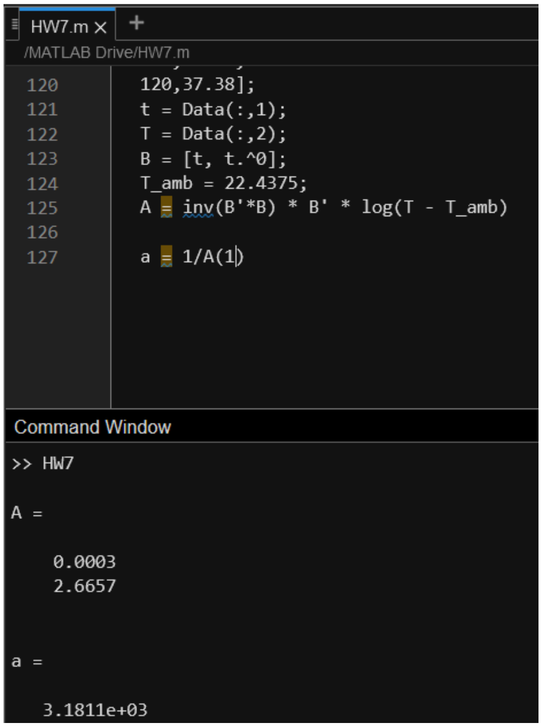
2.





```python
def record_temp():
    plot = [0] * 120
    file = open("out_file.txt", "a") # Also write data to file for later processing
    sec = 0
    while(sec < 120):
        while(flag == 0):
            pass
        global T
        sec += T

        temp_C = ds_sensor.read_temp(roms[0])
        ds_sensor.convert_temp()

        plot.pop()
        plot.insert(0, temp_C)
        file.write(str(sec) + ",")
        file.write(str(f"{temp_C:.2f}" + ";\n"))
        print(sec, temp_C)
        LCD.Bar(plot, White, LtBlue)
    file.close()

record_temp()
```

Code for #2 (hw_7a.py)

3. 

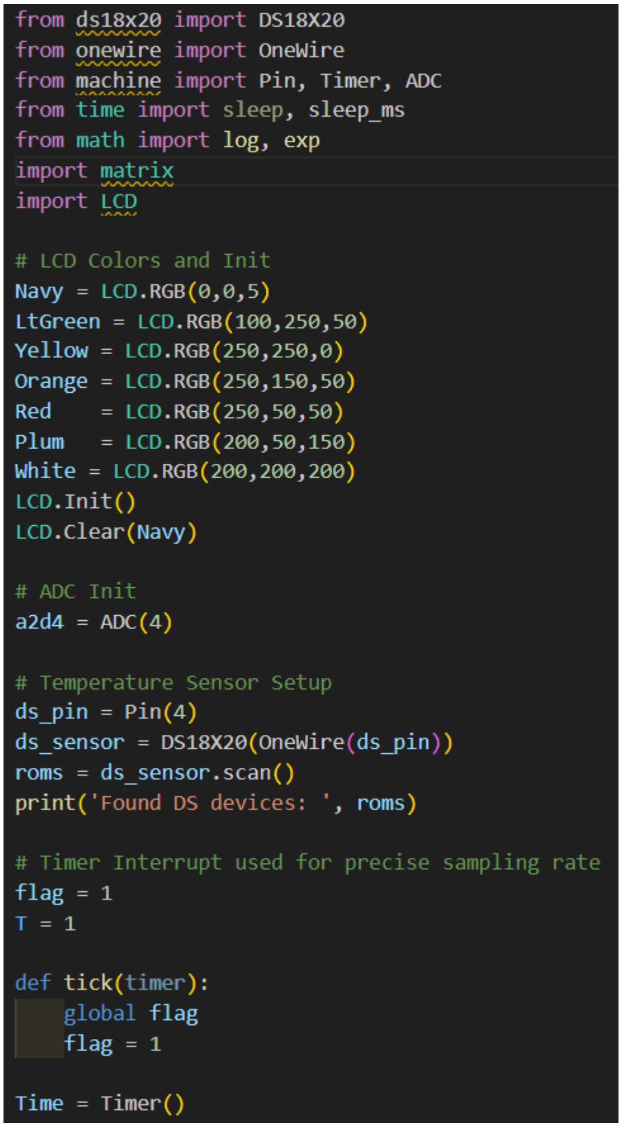
```matlab
120,37.38];
t = Data(:,1);
T = Data(:,2);
B = [t, t.^0];
T_amb = 22.4375;
A = inv(B'*B) * B' * log(T - T_amb)

a = 1/A(1)
```

```
>> HW7

A =

    0.0003
    2.6657


a =

   3.1811e+03
```

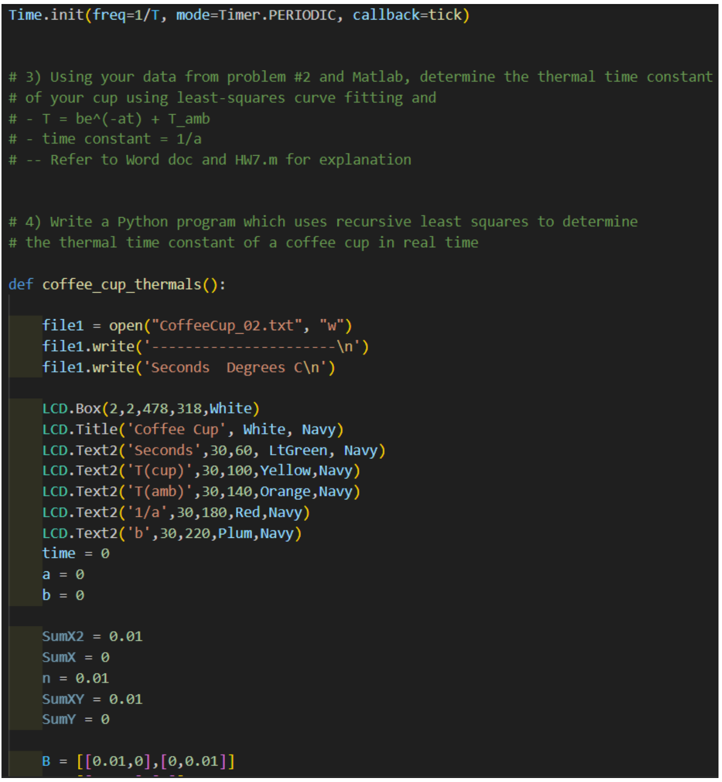
HW7.m

Matlab script with time constant of 3181. Sample 1 was 38.75, and sample 120 was 37.38.

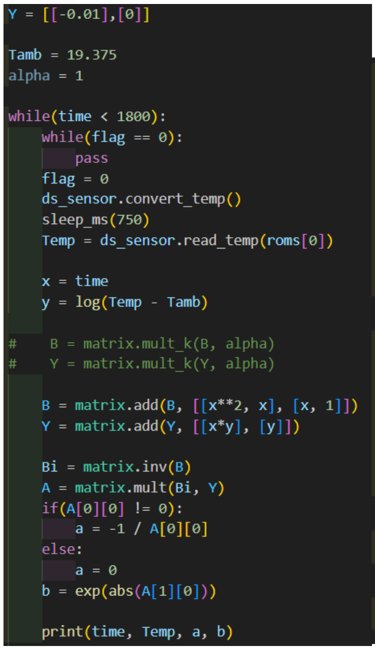
4.



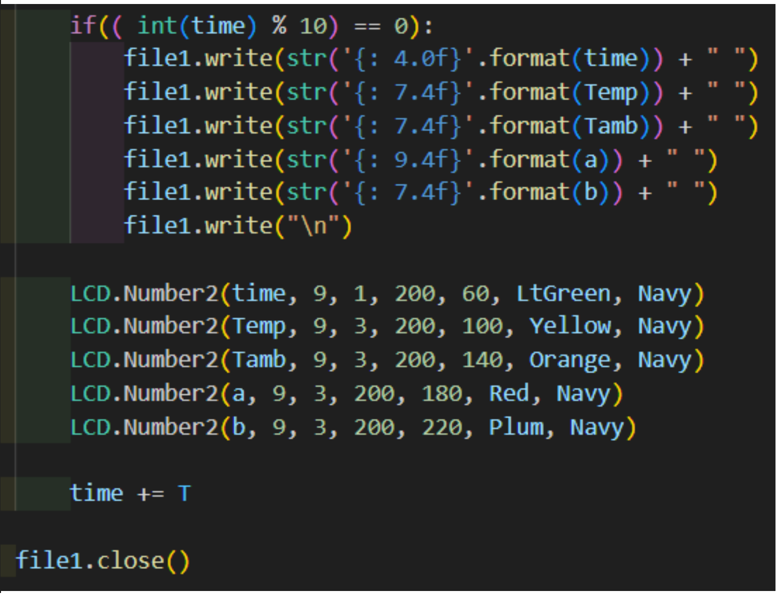
```
from ds18x20 import DS18X20
from onewire import OneWire
from machine import Pin, Timer, ADC
from time import sleep, sleep_ms
from math import log, exp
import matrix
import LCD

# LCD Colors and Init
Navy = LCD.RGB(0,0,5)
LtGreen = LCD.RGB(100,250,50)
Yellow = LCD.RGB(250,250,0)
Orange = LCD.RGB(250,150,50)
Red    = LCD.RGB(250,50,50)
Plum   = LCD.RGB(200,50,150)
White = LCD.RGB(200,200,200)
LCD.Init()
LCD.Clear(Navy)

# ADC Init
a2d4 = ADC(4)

# Temperature Sensor Setup
ds_pin = Pin(4)
ds_sensor = DS18X20(OneWire(ds_pin))
roms = ds_sensor.scan()
print('Found DS devices: ', roms)

# Timer Interrupt used for precise sampling rate
flag = 1
T = 1

def tick(timer):
    global flag
    flag = 1

Time = Timer()
```



```
Time.init(freq=1/T, mode=Timer.PERIODIC, callback=tick)


# 3) Using your data from problem #2 and Matlab, determine the thermal time constant
# of your cup using least-squares curve fitting and
# - T = be^(-at) + T_amb
# - time constant = 1/a
# -- Refer to Word doc and HW7.m for explanation


# 4) Write a Python program which uses recursive least squares to determine
# the thermal time constant of a coffee cup in real time

def coffee_cup_thermals():

    file1 = open("CoffeeCup_02.txt", "w")
    file1.write('----------------------\n')
    file1.write('Seconds  Degrees C\n')

    LCD.Box(2,2,478,318,White)
    LCD.Title('Coffee Cup', White, Navy)
    LCD.Text2('Seconds',30,60, LtGreen, Navy)
    LCD.Text2('T(cup)',30,100,Yellow,Navy)
    LCD.Text2('T(amb)',30,140,Orange,Navy)
    LCD.Text2('1/a',30,180,Red,Navy)
    LCD.Text2('b',30,220,Plum,Navy)
    time = 0
    a = 0
    b = 0

    SumX2 = 0.01
    SumX = 0
    n = 0.01
    SumXY = 0.01
    SumY = 0

    B = [[0.01,0],[0,0.01]]
```

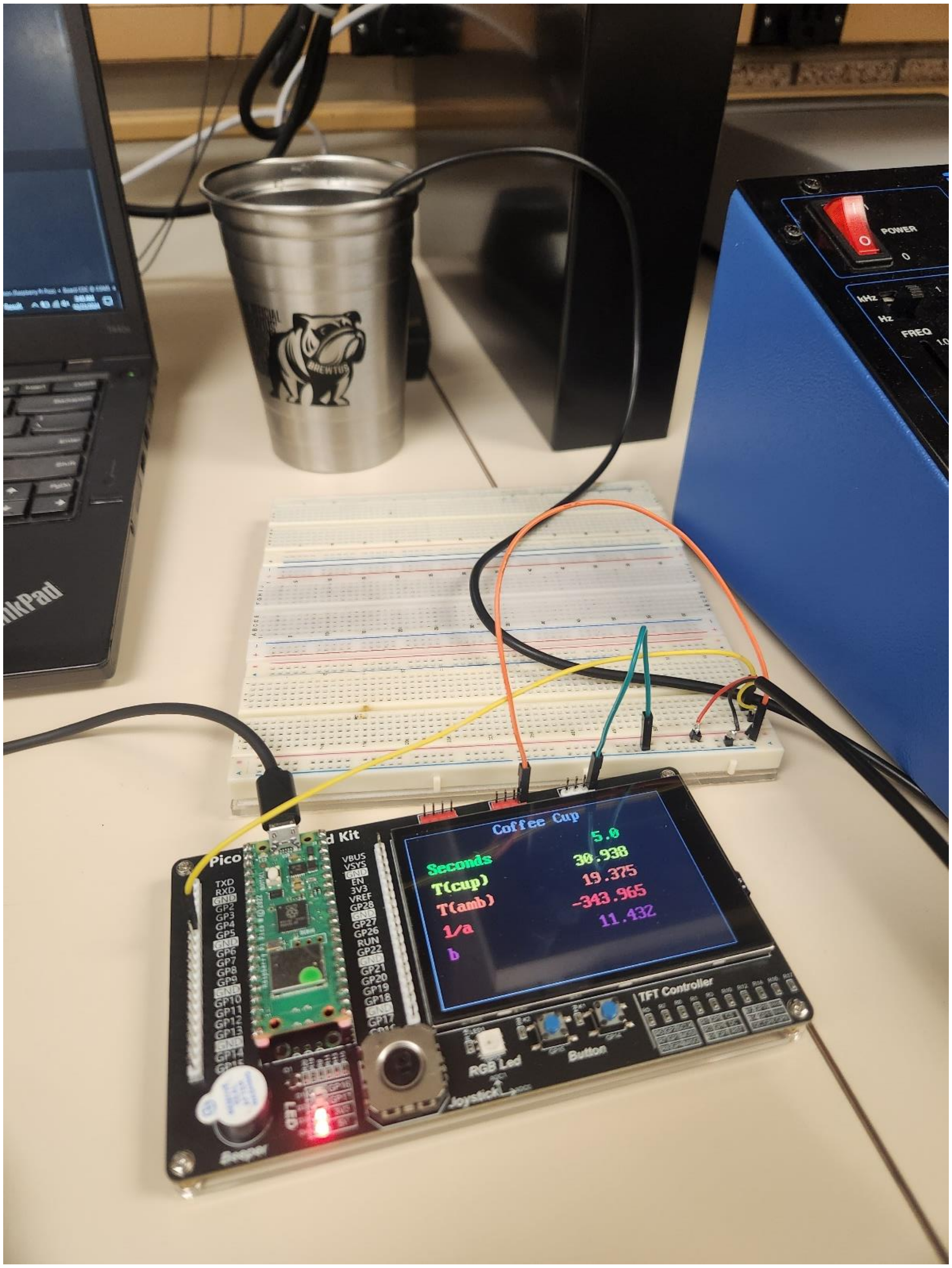


```
Y = [[-0.01],[0]]

Tamb = 19.375
alpha = 1

while(time < 1800):
    while(flag == 0):
        pass
    flag = 0
    ds_sensor.convert_temp()
    sleep_ms(750)
    Temp = ds_sensor.read_temp(roms[0])

    x = time
    y = log(Temp - Tamb)

#    B = matrix.mult_k(B, alpha)
#    Y = matrix.mult_k(Y, alpha)

    B = matrix.add(B, [[x**2, x], [x, 1]])
    Y = matrix.add(Y, [[x*y], [y]])

    Bi = matrix.inv(B)
    A = matrix.mult(Bi, Y)
    if(A[0][0] != 0):
        a = -1 / A[0][0]
    else:
        a = 0
    b = exp(abs(A[1][0]))

    print(time, Temp, a, b)
```



```
        if(( int(time) % 10) == 0):
            file1.write(str('{: 4.0f}'.format(time)) + " ")
            file1.write(str('{: 7.4f}'.format(Temp)) + " ")
            file1.write(str('{: 7.4f}'.format(Tamb)) + " ")
            file1.write(str('{: 9.4f}'.format(a)) + " ")
            file1.write(str('{: 7.4f}'.format(b)) + " ")
            file1.write("\n")

        LCD.Number2(time, 9, 1, 200, 60, LtGreen, Navy)
        LCD.Number2(Temp, 9, 3, 200, 100, Yellow, Navy)
        LCD.Number2(Tamb, 9, 3, 200, 140, Orange, Navy)
        LCD.Number2(a, 9, 3, 200, 180, Red, Navy)
        LCD.Number2(b, 9, 3, 200, 220, Plum, Navy)

        time += T

    file1.close()
```

Hw_7b.py (from BisonAcademy)

5. 

Thermal Constants:

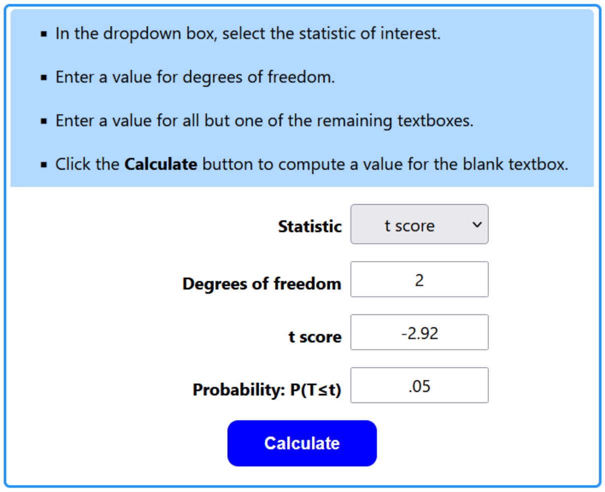
- Test 1: 4273
- Test 2: 4290
- Test 3: 4332

Sample Mean: 4298

Standard Deviation: 31.48

Sample Size: 3

Degrees of Freedom: 2



- In the dropdown box, select the statistic of interest.
- Enter a value for degrees of freedom.
- Enter a value for all but one of the remaining textboxes.
- Click the **Calculate** button to compute a value for the blank textbox.

Statistic t score

Degrees of freedom 2

t score -2.92

Probability: P(T≤t) .05

I am 90% confident the thermal coefficient is between 4206.08 and 4389.92