HW #10 – Loops, plots, statistics, and data collection

1. Write a program that retrieves a temperature measurement from the Temperature Control Lab (see https://apmonitor.com/heat.htm) and displays the value on a plot.

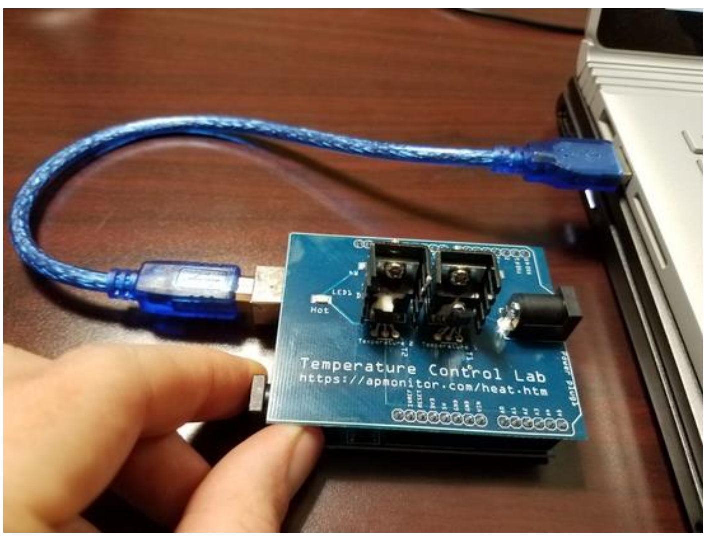
   Instructions for setting up and connecting to the TCLab:

   a. Install **tclab** as a Python module with pip package manager (command prompt in Windows with **cmd** or a new Terminal in MacOS):
      - pip install tclab

      or directly from within the Python script:

```
module='tclab'
try:
      from pip import main as pipmain
except:
      from pip._internal import main as pipmain
pipmain(['install',module])
# to upgrade: pipmain(['install','--upgrade',module])
```

   b. Plug in device to a USB port. The heater power supply is not needed for this exercise.



   c. Run the following test script to ensure that the device is connected. The LED red light should turn on and then off after 1 second.

```
import tclab
import time
# Connect to Arduino
a = tclab.TCLab()
print('LED On')
a.LED(100)
# Pause for 1 second
time.sleep(1.0)
```

```
print('LED Off')
a.LED(0)
a.close()
```

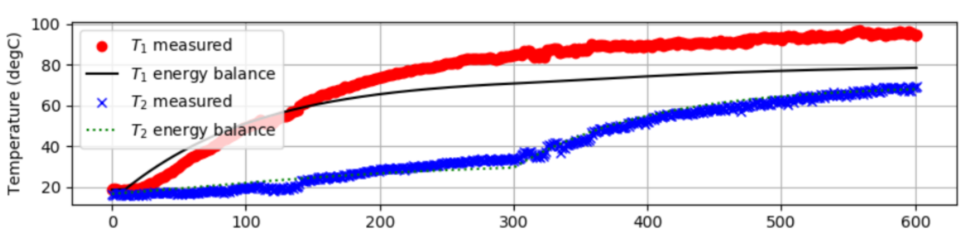
d. Test the temperature reading:

```
import tclab
import numpy as np
import time
# Connect to Arduino
a = tclab.TCLab()
# Temperature
print('Temperature (degC)')
print(a.T1)
# Sleep (10 sec) with LED on
a.LED(100)
time.sleep(10.0)
a.LED(0)
# Temperature
print('Temperature (degC)')
print(a.T1)
# Close connection to Arduino
a.close()
```

Write a program that retrieves a temperature value from the Temperature Control Lab each second for 1 minute. Create a plot of the temperature value versus the time is was collected (start from 0 sec). Report the average (mean), maximum, and minimum temperatures. The temperature should be reported in degrees Celsius.

Sample Output:

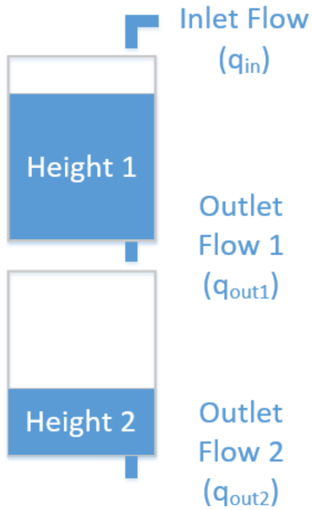
Max: 98.0 degC
Min: 18.0 degC
Average: 70.0 degC



Note that two temperatures are available (a.T1 and a.T2). For this assignment, please select just a.T1 for the summary statistics and plotting.

2. Gravity Drained Tanks in Python

Cylindrical dual gravity drained tanks with a constant cross sectional area (A_c=2 m^2) and maximum height of 1 *m*. If the tank overfills, the excess fluid is lost. There is an inlet flow q_{in}, an intermediate outlet flow from tank 1 to tank 2 as q_{out1}, and a final outlet flow as q_{out2}. All flows are in units of m^3/hr and heights are reported in units of *m*.



A mass balance on each tank is used to derive the following equations that relate inlet flow to the height of the tanks.

$$A_c \frac{dh_1}{dt} = q_{in} - q_{out1} \qquad A_c \frac{dh_2}{dt} = q_{out1} - q_{out2}$$

The outlet flow rate for each tank depends on the height in the tank according to Bernoulli's equation for incompressible fluids as:

$$q_{out1} = c_1\sqrt{h_1} \qquad q_{out2} = c_2\sqrt{h_2}$$

The tanks are initially empty when the inlet to tank 1 starts to flow at a rate of 0.5 m^3/hr.

a) **Solve** for the heights (h_1 and h_2) as functions of time with c_1=0.13 and c_2=0.20. Use a timestep size of **dt=0.5 *hr*** and solve to **t=10 hr**.

b) **Plot** the predicted heights h_1 and h_2 as functions of time on the same plot. Label the axes as "time (hr)" and "height (m)".

Hint: use an explicit Euler's equation applied to each *dh/dt* above: $dh/dt = f(h,t) \rightarrow h_{n+1} = h_n + dt*f(h_n,t_n)$.
Don't forget to add an IF statement to check for overfill conditions such as:

```
if (height[i]>=1.0 and dheight_dt>=0):
   height[i+1] = 1.0
else:
   height[i+1] = height[i] + dt * ( f(height[i], qin, qout1) )
```