BIOE3090

Controls and Circuits Homework

30 Pts

Due: 2/13/2018 (Before Class, see canvas)

# Part I: C++ Primer (20 points)

The software provided on the UCDBioe Github account (see below for downloading instructions) contains all of the Arduino firmware required to control the field incubator and provides data communications between the Arduino and an optional terminal running Matlab. However, the firmware is missing failsafe modes for overheating the box and for over discharging the battery. Since it would be difficult to run the native Arduino firmware without the associated hardware, you have been provided an incomplete C++ program that you can use to develop some of the functions that would be useful in implementing these failsafes.

**Overview of the bioe3090\_HW.cpp code**

The code consists of several function definitions and a main function. The main function defines variables and arrays used to make a randomly generated test dataset for battery voltages and temperature readings (battVals and tempVals). The batVals and tempVals array lengths can be changed and the arrays can be modified to have a low and high ranges that allow for the user to build a dataset that has a transition that would cause a failure alarm to be tripped. For example, the program sets up the battery arrays where the first 22 elements have an average value of 12.0V ±0.02V, and the rest of the elements have a value of 11.3V ±0.02V. The alarm should be tripped when the average battery voltage drops below 11.6V. See source code for further details.

The program also sets up two circular buffers to help filter some of the noise out of the data (tempBuffer, battBuffer). These circular buffers are a straightforward way to reduce some of the noise in real-world signals (emulated here as random noise). This is how they work:

For an incoming data stream

datastream = [12.1, 12.2, 12.3, 12.4, 12.7, 12.6, 12.5]

The buffer rolls along the data stream and holds a subset of the overall data

buffer = [12.2, 12.3, 12.4, 12.7]

average(buffer) = 12.4

When new data comes in, we shift the array values to the left and replace the last element with the new data.

buffer = [12.3, 12.4, 12.7, 12.6]

average(buffer) = 12.5

Gaussian distributed random noise is thus reduced by using an averaging filter.

**Fixing the Code**

The code provided should run, it just will not do what it is supposed to until you fix it. This should allow you to debug the software in a functional state. I would suggest compiling and test running the code early and often rather than trying to complete all of the sections then trying to debug the code. Also, the C-code required to complete this assignment should not be anything terribly advanced. Simple loops, print statements and arithmetic should be sufficient.

1. (5-pts)

Complete the code needed to print the values contained in the arrays in a nicely formatted way. Your output should look similar to Figure 1, where the values are aligned roughly in columns. The code sections are outlined in the bioe3090\_HW.cpp file using the tags:

(STUDENT COMPLETE THIS SECTION >> -- HWS1----)

Your code goes here!!!!!!!!!!!!!! Place your code inside the bioe3090\_HW.cpp file, but keep it between these tags so I can find it!

(STUDENT COMPLETE THIS SECTION << -- HWS1----)

1. (5-pts.)  
   Complete the code needed to print the average values of a given array. The code sections are outlined in the bioe3090\_HW.cpp file using the tags:

(STUDENT COMPLETE THIS SECTION >> -- HWS2----)

Your code goes here!!!!!!!!!!!!!! Place your code inside the bioe3090\_HW.cpp file, but keep it between these tags so I can find it!

(STUDENT COMPLETE THIS SECTION << -- HWS2----)

1. (5-pts)

Complete the code needed to shift the buffer array and replace the last value with new data. This is the circular buffer used to average out noise in the signals. The code sections are outlined in the bioe3090\_HW.cpp file using the tags:

(STUDENT COMPLETE THIS SECTION >> -- HWS3----)

Your code goes here!!!!!!!!!!!!!! Place your code inside the bioe3090\_HW.cpp file, but keep it between these tags so I can find it!

(STUDENT COMPLETE THIS SECTION << -- HWS3----)

1. (5-pts)

Complete the code needed to check if a shutdown condition has been met. The code sections are outlined in the bioe3090\_HW.cpp file using the tags:

(STUDENT COMPLETE THIS SECTION >> -- HWS4----)

Your code goes here!!!!!!!!!!!!!! Place your code inside the bioe3090\_HW.cpp file, but keep it between these tags so I can find it!

(STUDENT COMPLETE THIS SECTION << -- HWS4----)

You may get a warning when you compile the code before you have fixed all of the sections: **warning: control reaches end of non-void function [-Wreturn-type].** This warning simply means that you have a function defined to return some value and no value was returned. It will fix itself when you have valid return statements.

What to turn in for part 1

Your output should look similar to Figure 1. Turn in your .cpp file, put your name in the filename, i.e. steve\_lammers\_bioe3090\_HW.cpp

Your code should compile and run. You will be mostly graded based on whether your code produces the correct output when it is compiled and run. You should not need to include any other libraries to do this assignment.

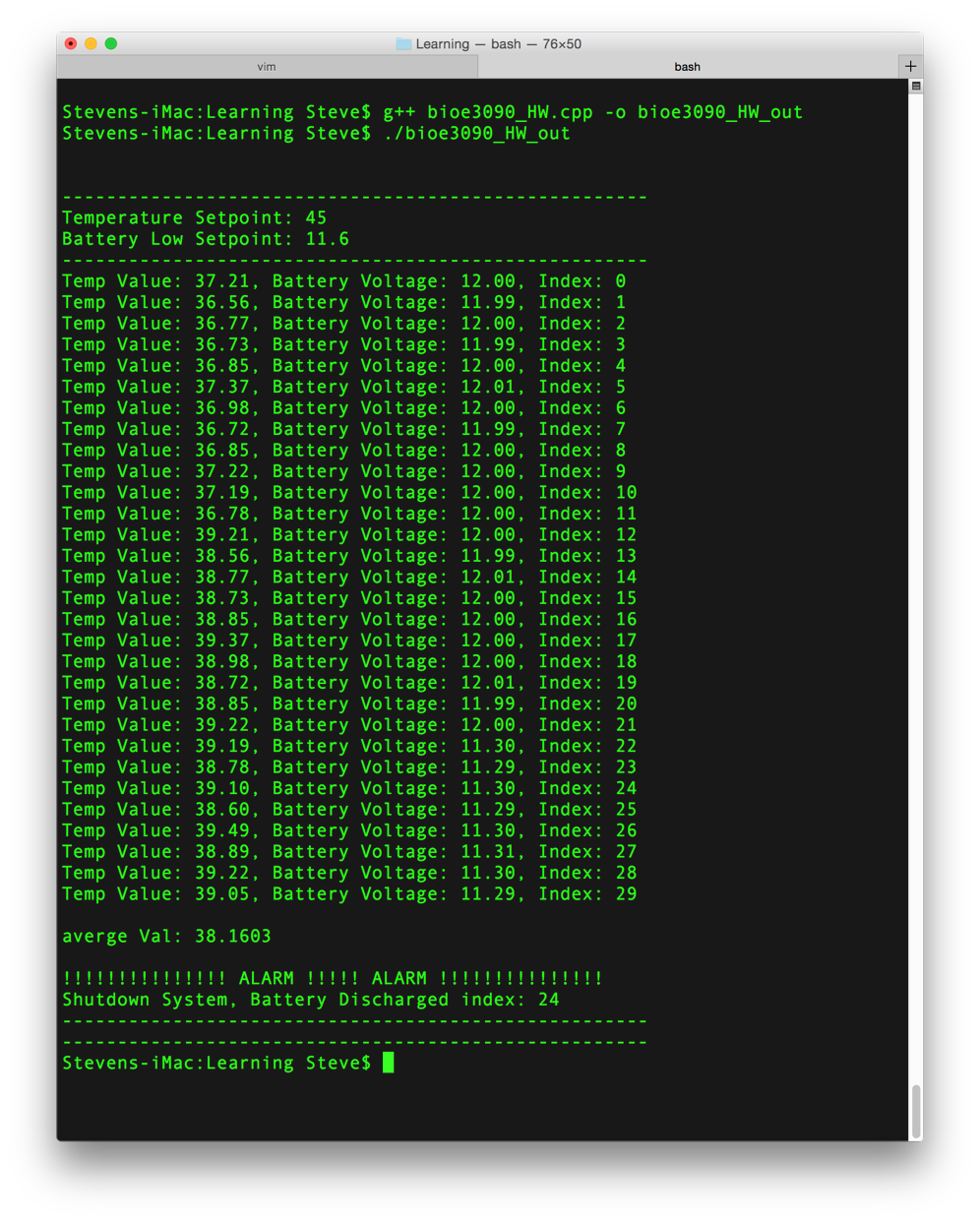


Figure 1: Output from completed C++ program.

**Downloading Instructions:**

The github repository is now public, so you can download the files needed to do this homework and to build your first design project.

<https://github.com/UCDBioe/Bioe3090_Incubator>

Remember to click the green "Clone or Download" button to be able to download the files. The homework is in the homework directory.

# Part 2: Soldering and Components (10 points)

Watch the “Soldering Video” and the “Multimeter Tutorial Video” posted to Canvas-Modules-Technical Content and Lab Module. Answer the following questions.

1. Why do you need to keep your solder tip clean?
2. Why do we tin the soldering iron?
3. What shape indicates a good solder joint on a printer circuit board?
4. What is continuity?
5. Can you use a multimeter to measure a resistor that is wired into a circuit?
6. (5-pts) What happens if you do not remember to splice in the multimeter when checking a circuit and instead use the probes as you would use to check a voltage reading? For example, you went to check the amps of a 120V wall outlet and plugged in the probes directly to the outlet as was demonstrated in the video when AC voltage was being measured? Describe how this could be dangerous and potentially damage equipment.