WHAT’S UP

LAB # 4

SECTION # 9

SUBMITTED BY: CONNER SPAINHOWER

OCTOBER 1, 2019

**Problem**

The problem for this lab was to get the DualShock 4 to output what direction it was facing depending on the rotation of the controller. This would need to include 3 or more of our own functions and use the magnitude function to determine the values of rotation. Along with this, we also needed code that set tolerances for the changing numbers. At the end, the program must be exited with the triangle button on the DualShock 4.

**Analysis**

Right off the bat, the C file gave us some numbers that ran through Cygwin. There was a list of numbers, and those numbers would change depending on the orientation of the controller. This gave me an input for my tolerances and helped me get a foundation of how this would be done. When the direction of the controller would change, so would the numbers.

**Design**

Given the numbers as input from the controller, I had to interpret what those meant. I looked at the C file and I saw the numbers were input from the controller and output as code I could manipulate. So, I then starting writing if statements inside the while loop. The while loop would keep going as long as there was input, basically, and the if statements would take the number in. After the number was input, it would be compared between two values, and if it was then it would output the direction as a print statement.

It was important to not put only if statements, because if there were 2 true values, then it would output two directions. So, I had to do else-if statements after the first if statement so that wouldn’t happen as easily. Another design flaw I had to meddle with was using 2 functions for the direction output. Some values were output negative when a certain direction, so it was much easier to use 2 functions that do opposite things to output the direction. For example, I had the function close\_to and close\_to2 that were different by either subtracting or adding the input numbers to compare to the tolerance and get the output.

The last thing to design was the exit protocol. This was rather simple, as when a button was pushed out of the four on the controller, the value would turn one. In return, all I had to do was put an else-if statement that would end the while loop and exit the program if that button was ever 1.

**Testing**

When testing this program, it was difficult at first. After designing the programs, I had to tweak the tolerances quite a bit to get the output I wanted. As you can see in the code, the tolerances are different depending on the direction because some directions are more sensitive than others. Another problem I had was designing the functions. Functions are complicated (for me) to put into the program correctly, but in the end make the process much simpler. As long as I can overcome that obstacle, the testing would workout.

**Comments**

The lab, in my opinion, was a challenging but rewarding task. It is nice to be able to say that you can program software that will output the direction a controller is facing, which is used by gaming companies everywhere to utilize the controller to movements. I would not change the lab, as the problems may need some direction, but in the end is achievable.

**Questions**:

**1. How did you approach the design?** My approach is always breaking the problem down into smaller parts. I ask myself, what steps do I need to achieve this goal? Sometimes this even requires getting out a notepad and writing them down and coding each small step into one big step in the end.

**2. What data did you have to read in?** We had to read in the data from the DualShock 4 controller. This was a list of 11 numbers and we had to decide which number meant what in the output.

**3. What functions did you choose to implement and why?** I chose functions that set tolerances for the input numbers so the output could be what I want. For example, if gz was between the numerical value of 0.25 and 0.75, then it would output the direction. This was my close\_to function. The close\_to2 function was the same thing but it added the values instead of subtracted them, since the inputs were negative. Lastly, of course, the magnitude function was implemented, as it was necessary to compare the values.

**4. What tolerance values did you pick and how did you decide on them?** This was a tricky one for me, because at first, I kept on getting 2 or 3 directions at a time. I learned that some tolerances had to be smaller or bigger because some directions are more sensitive than others. I usually tried to shoot between 0.25 and 0.75, but my last direction, BACK, was less sensitive than the others.



