

The DS4 Equalizer

LAB # 7

SECTION #

Riley Lawson

SUBMISSION DATE:

10/29/2019

DATE:

10/29/2019

Problem

The purpose of this lab was to create a bar graph that you set to correspond with the controller when it moves right or left, front to back, or moves with the analog sticks. Also, you press a button to change to the different modes. The program gives you a set of functions and very little code to work with and you ended up having to place a function within a function. This ended being separated by many functions and a very small amount of code in the main function.

Analysis

This wants us to focus on developing problem-solving skills in c, writing certain functions to a recommended specification, clarifying the use for loops, functions, and branching, and to practice using output parameters. Overall, this leads to a graph of the controller (for example: the more you tilt it to the right the more "R"s there will be and vis vera for "L") along with a graph for left-right, front-back, and analog stick.

Design

Within this program, I ended up doing a lot of research on how to even start it. I learned a little from my last mistake and actually had a half-decent plan to get started with it. This process is still vague but I think it represents my process a little better than last time:

1. Examine existing code
2. Focus on an existing function
3. Figure out what needs to be dealt with
4. Route the function within another function
5. Return code back to main

There were two equations used to scale the values and that included one for gyro_value and joy_value. gyro_value was $\text{gyro_value} * 39$ because that was the return value and for the joy_value it was $(-\text{joy_value} * (39 / 127.0))$ because of the 39 that comes from the return value and is divided by 127 (the magnitude) which gives you the values.

Testing

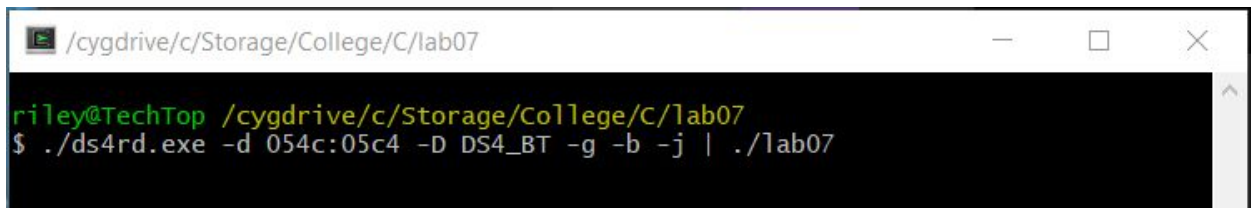
This program consisted of many trials and errors of trying to figure out the graph and how to display it. I knew when I first read the program I had to either develop a switch statement within the main program or develop many if, else-if statements to link it back to whatever function it needed. Just saying it didn't sound that bad, however in practice, this took way more work than I originally expected. There were many errors to sort through each bit of code. When it came to compiling though, it never really showed many errors even with -Wall. At first, I thought I had finally figured it out, but it only sort of work. It showed the "R"s and the "O"s properly but when I moved it to the left it just showed blank spaces. I realized that all of the errors that had occurred were extremely minor and it included not

having 127.0 instead of 127 or not adding a -1 somewhere. All of these simple things really messed up my code. In the end, these small tweaks ended up being way bigger setbacks than originally intended.

Comments

In this lab, I learned that having an actual process to go along with the development of your code can actually really help. I don't think I had a good process, but I definitely thought it was much better than the last lab's process. In the screenshot below I have forgotten to take a picture of the code running.

ScreenShots

A screenshot of a terminal window. The title bar shows the path "/cygdrive/c/Storage/College/C/lab07". The terminal content shows the prompt "riley@TechTop" followed by the command "./ds4rd.exe -d 054c:05c4 -D DS4_BT -g -b -j | ./lab07".

```
/cygdrive/c/Storage/College/C/lab07  
riley@TechTop /cygdrive/c/Storage/College/C/lab07  
$ ./ds4rd.exe -d 054c:05c4 -D DS4_BT -g -b -j | ./lab07
```