Introduction the DS4 and Functions

LAB REPORT # 3

SECTION # 1

Jesus Horacio Soto Gonzalez

SUBMISSION DATE:

9/23/2022

Problem

Problem # 1:

DualShock 4 Data Collection

Analysis

This problem introduces our class to flags and why they are important in order to analyze different kinds of information. It also demonstrates the great amount of data that can be extracted from an object like a controller and collected to help us understand computer interpretations of a code better, such as the graphs we created in this problem.

Design

For problem number one we collected the data from the DualShock 4 controller according to the instructions provided and created graphs to analyze and interpret the data we collected.

Testing

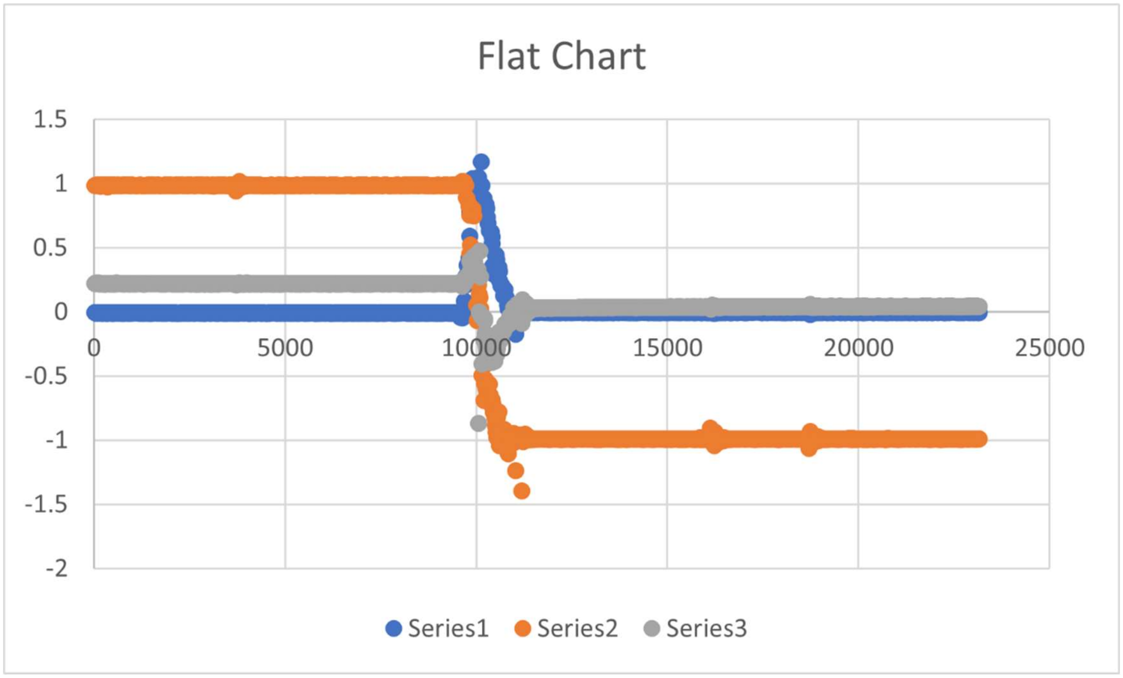
For the testing process we mainly made sure that the data we collected was appropriate and made sense according to the instructions. We had to observe that the movement and direction of the controller changed the output of the values interpreted by the computer.

Comments

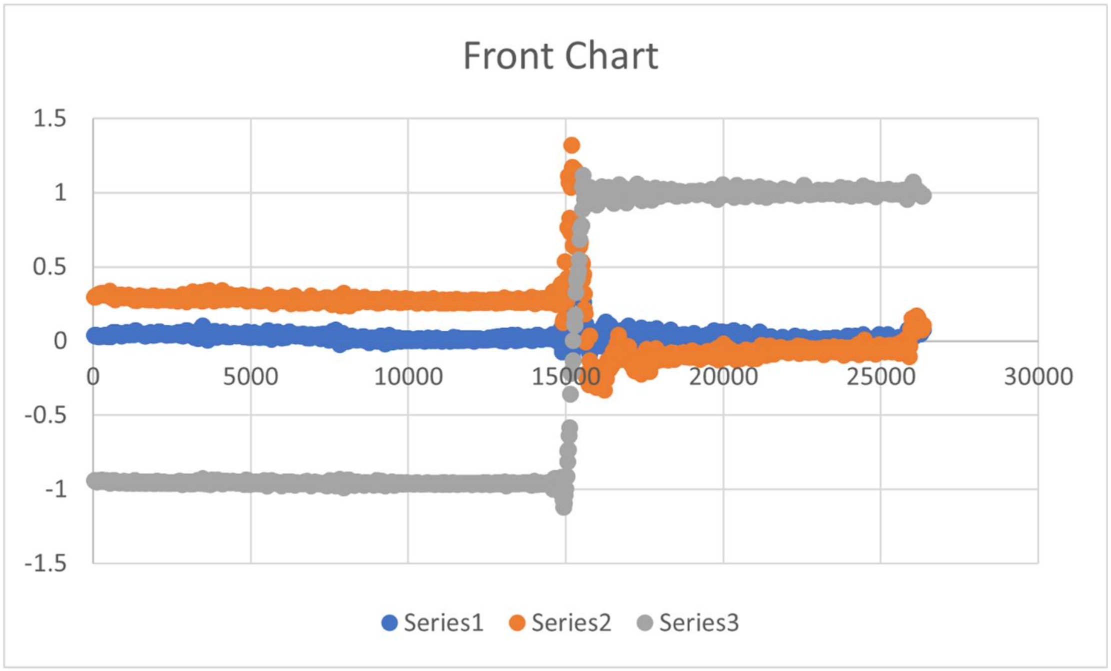
This problem helped us understand the importance analyzing data with an object that all of us are familiar with. It is impressive the amount of information that can be extracted from a controller and that needs to be examined and understand by a computer program without failed.

Screen Shots

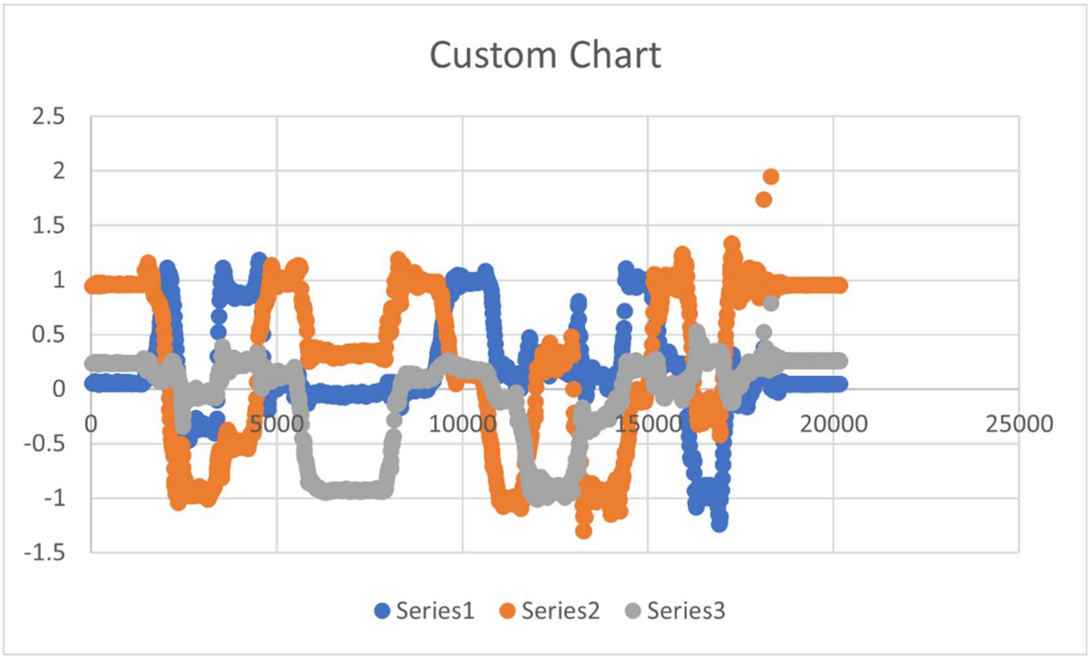
SS #1:



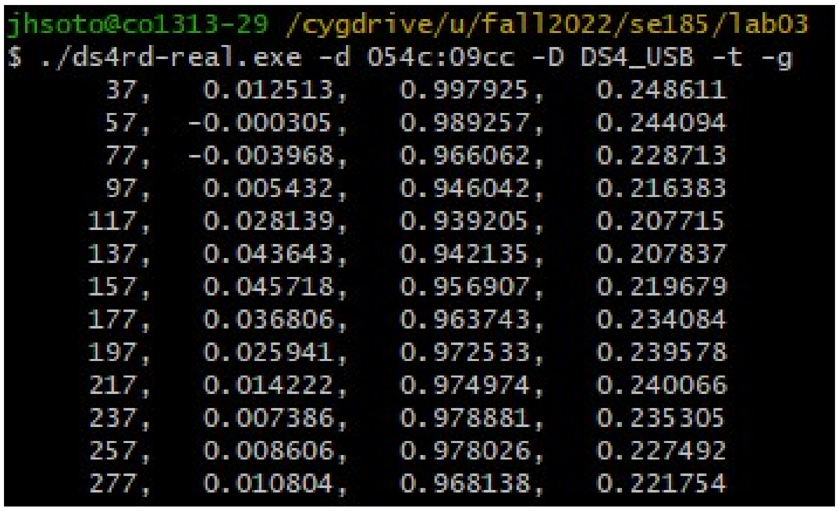
SS #2:



SS #3:



SS #4:



```
jhsoto@co1313-29 /cygdrive/u/fall2022/se185/lab03
$ ./ds4rd-real.exe -d 054c:09cc -D DS4_USB -t -g
      37,   0.012513,   0.997925,   0.248611
      57,  -0.000305,   0.989257,   0.244094
      77,  -0.003968,   0.966062,   0.228713
      97,   0.005432,   0.946042,   0.216383
     117,   0.028139,   0.939205,   0.207715
     137,   0.043643,   0.942135,   0.207837
     157,   0.045718,   0.956907,   0.219679
     177,   0.036806,   0.963743,   0.234084
     197,   0.025941,   0.972533,   0.239578
     217,   0.014222,   0.974974,   0.240066
     237,   0.007386,   0.978881,   0.235305
     257,   0.008606,   0.978026,   0.227492
     277,   0.010804,   0.968138,   0.221754
```

Problem

Problem 2:

Introduction to Functions and the DualShock 4

Analysis

This problem asks us to modify the given program so that we can adjust the value of the outputs of the computer from milliseconds to seconds as well as change the character area of the output to an eight-character area and three decimal places of precision for time and a seven-character area with four decimal digit precision for acceleration. Finally, the problem instructs us to create a function that can output the magnitude of acceleration from the three different acceleration values given.

Design

```
while (1)

   {

      scanf("%d, %lf, %lf, %lf", &t, &ax, &ay, &az);

      /* CODE SECTION 0 */

      printf("Echoing output: %08.3lf, %07.4lf, %07.4lf, %07.4lf\n", t / 1000.0, ax * 1000.0, ay * 1000.0, az
* 1000.0);

      /*   CODE SECTION 1 */

                    printf("At %d ms, the acceleration's magnitude was: %lf\n", t, magnitude(ax, ay, az));

  }


   return 0;

}

int minutes(int ms){

                    int min = ms / 60000;

                    return min;

}

int seconds (int ms){

                    int sec = (ms / 1000) % 60;

                    return sec;
```

```
}
int milliseconds (int ms){
                        int mil = ms % 1000;
                        return mil;
}
       /*   CODE SECTION 2 */
           printf("At %d minutes, %d seconds, and %d milliseconds it was: %lf\n",
           minutes(t), seconds(t), milliseconds(t), magnitude(ax, ay, az));


Calculates and returns the magnitude of three given values.
 * @param x - The x-axis scanned values from the DS4 controller.
 * @param y - The y-axis scanned values from the DS4 controller.
 * @param z - The z-axis scanned values from the DS4 controller.
 * @return - The magnitude of the given values.
 */
double magnitude(double x, double y, double z)
{
    return sqrt((x * x) + (y * y) + (z * z));
}
```

For this design we are given the mathematical formula that calculates the magnitude of acceleration. Our job is to change the formula to something that the computer can interpret and output a value with. To do this we use the math.h folder and utilize the sqrt() function and some multiplication which can also be done by using the pow() function. After we set up the function the computer can calculate the value and output the magnitude of acceleration.

Testing

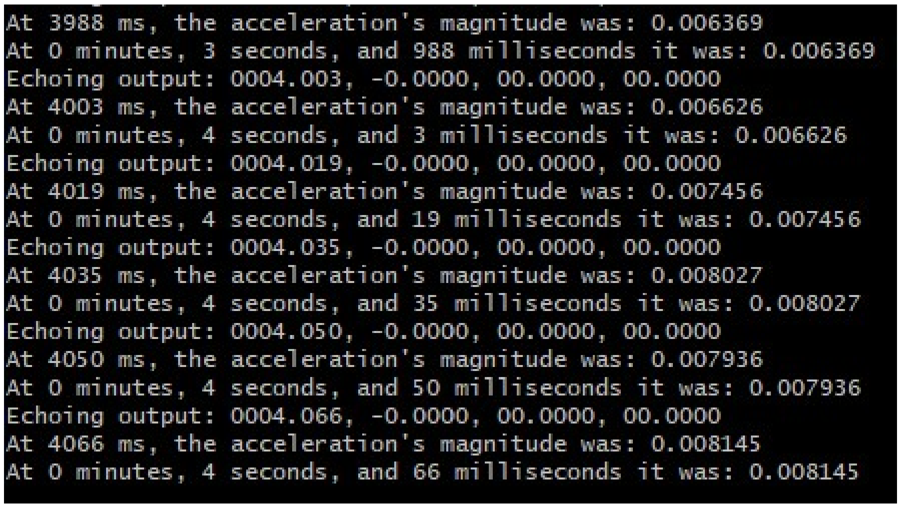
Testing went smoothly, after some taught I realize that I am able to understand the program better if fore the last part of the problem I just multiplied the variables by itself instead of using the pow() function and avoid confusion.

Comments

This problem showed us that it is possible to modify different input and output values to facilitate our comprehension of the information obtained by interacting with the controller and the computer. It also demonstrates that we can create more complicated calculations with the inputs of the device and that we can add programs to obtained very detailed values from those inputs.

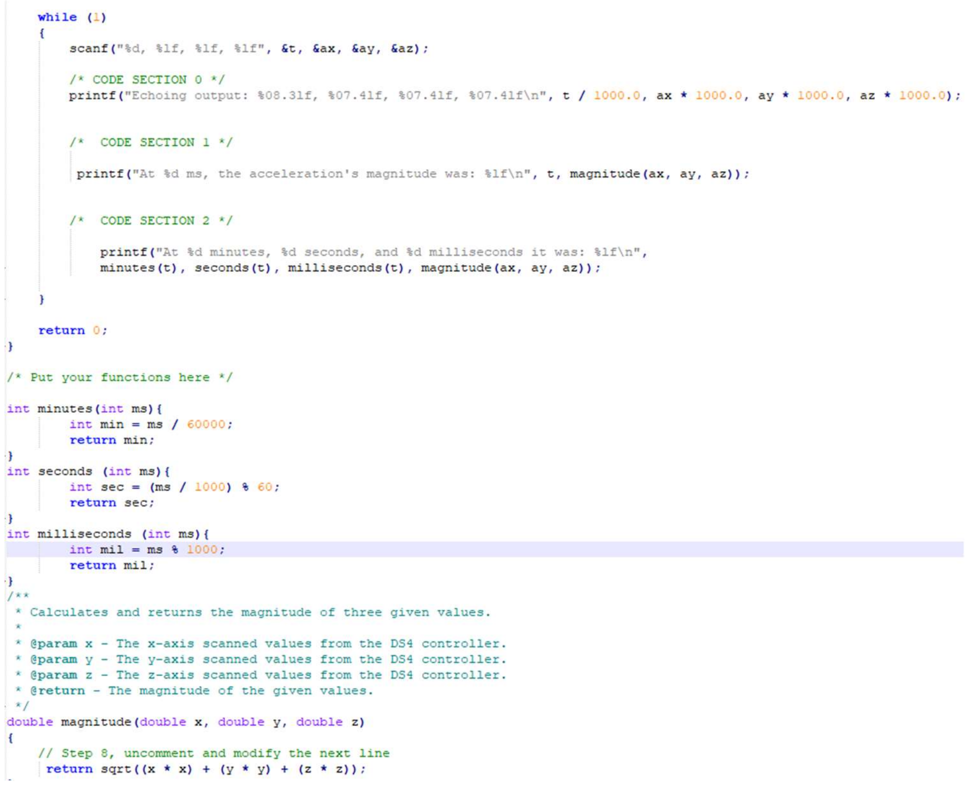
Screen Shots

SS #1



```
At 3988 ms, the acceleration's magnitude was: 0.006369
At 0 minutes, 3 seconds, and 988 milliseconds it was: 0.006369
Echoing output: 0004.003, -0.0000, 00.0000, 00.0000
At 4003 ms, the acceleration's magnitude was: 0.006626
At 0 minutes, 4 seconds, and 3 milliseconds it was: 0.006626
Echoing output: 0004.019, -0.0000, 00.0000, 00.0000
At 4019 ms, the acceleration's magnitude was: 0.007456
At 0 minutes, 4 seconds, and 19 milliseconds it was: 0.007456
Echoing output: 0004.035, -0.0000, 00.0000, 00.0000
At 4035 ms, the acceleration's magnitude was: 0.008027
At 0 minutes, 4 seconds, and 35 milliseconds it was: 0.008027
Echoing output: 0004.050, -0.0000, 00.0000, 00.0000
At 4050 ms, the acceleration's magnitude was: 0.007936
At 0 minutes, 4 seconds, and 50 milliseconds it was: 0.007936
Echoing output: 0004.066, -0.0000, 00.0000, 00.0000
At 4066 ms, the acceleration's magnitude was: 0.008145
At 0 minutes, 4 seconds, and 66 milliseconds it was: 0.008145
```

SS #2:



```c
    while (1)
    {
        scanf("%d, %lf, %lf, %lf", &t, &ax, &ay, &az);

        /* CODE SECTION 0 */
        printf("Echoing output: %08.3lf, %07.4lf, %07.4lf, %07.4lf\n", t / 1000.0, ax * 1000.0, ay * 1000.0, az * 1000.0);


        /*  CODE SECTION 1 */

         printf("At %d ms, the acceleration's magnitude was: %lf\n", t, magnitude(ax, ay, az));


        /*  CODE SECTION 2 */

            printf("At %d minutes, %d seconds, and %d milliseconds it was: %lf\n",
            minutes(t), seconds(t), milliseconds(t), magnitude(ax, ay, az));

    }

    return 0;
}

/* Put your functions here */

int minutes(int ms){
        int min = ms / 60000;
        return min;
}
int seconds (int ms){
        int sec = (ms / 1000) % 60;
        return sec;
}
int milliseconds (int ms){
        int mil = ms % 1000;
        return mil;
}
/**
 * Calculates and returns the magnitude of three given values.
 *
 * @param x - The x-axis scanned values from the DS4 controller.
 * @param y - The y-axis scanned values from the DS4 controller.
 * @param z - The z-axis scanned values from the DS4 controller.
 * @return - The magnitude of the given values.
 */
double magnitude(double x, double y, double z)
{
    // Step 8, uncomment and modify the next line
     return sqrt((x * x) + (y * y) + (z * z));
```

Problem

Problem 3:

Counting Buttons

Analysis

For this problem we had to write a program able to output the number of buttons being pressed on the DualShock 4.

Design

For the design of this code, I had to make sure that the computer collected the data on each of the buttons pressed and to create a function that added each of the buttons pressed correctly providing the sum of the buttons pressed as an output.

```
        int buttonsPressed(int triangle, int circle, int x, int square);


int main(int argc, char *argv[])
{
        int triangle, circle, x, square;
  while (1)
  {
        scanf("%d, %d, %d, %d", &triangle, &circle, &x, &square);
        printf("Buttons pressed = %d\n", buttonsPressed(triangle, circle, x, square));
  }

  return 0;
}


/* Put your functions here, and be sure to put prototypes above. */
        int buttonsPressed(int triangle, int circle, int x, int square)
        {
                int buttonspressed;
        buttonspressed = triangle + circle + x + square;
```

return buttonspressed;

}

Testing

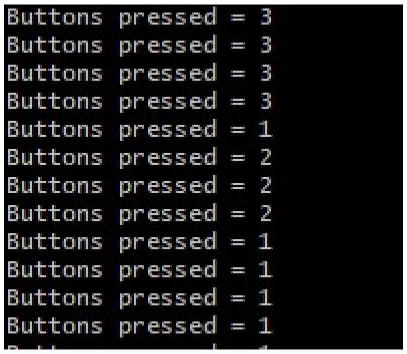
For this problem during testing, I had difficulties with the program not registering the number of buttons pressed. It provided an answer when I pressed one, but it was not able to count how many were pressed at the same time.

Comments

With this problem I was able to interact with the controller and observe the change of values and calculations made from my own code. Really interesting problem to show the capabilities of programming.

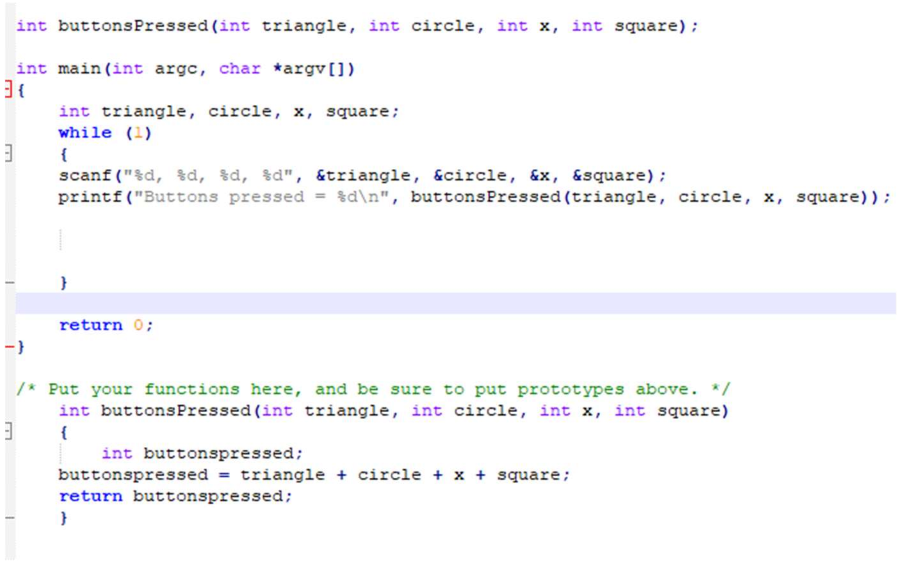
Screen Shots

SS #1



```
Buttons pressed = 3
Buttons pressed = 3
Buttons pressed = 3
Buttons pressed = 3
Buttons pressed = 1
Buttons pressed = 2
Buttons pressed = 2
Buttons pressed = 2
Buttons pressed = 1
Buttons pressed = 1
Buttons pressed = 1
Buttons pressed = 1
```

SS #2



```
int buttonsPressed(int triangle, int circle, int x, int square);

int main(int argc, char *argv[])
{
    int triangle, circle, x, square;
    while (1)
    {
    scanf("%d, %d, %d, %d", &triangle, &circle, &x, &square);
    printf("Buttons pressed = %d\n", buttonsPressed(triangle, circle, x, square));


    }

    return 0;
}

/* Put your functions here, and be sure to put prototypes above. */
    int buttonsPressed(int triangle, int circle, int x, int square)
    {
        int buttonspressed;
    buttonspressed = triangle + circle + x + square;
    return buttonspressed;
    }
```