Lab 4 Report

LAB 4

SECTION V

Zach Johnson

SUBMISSION DATE:

9/30/2014

9/26/2014

Problem

The given problem for this lab was to take data from the esplora and buy using a variety of different functions. These functions involved the manipulation of the data and some simple conversions. We were also asked to reproduce the magnitude function which was given to us and output them. Finally we needed to determine how many buttons on the controller were being used at a given time.

Analysis

To look at this problem we need to first asses how the computer displayed the time. From the time returned in milliseconds we were asked to convert this to display a more human friendly number in minutes, seconds and remaining milliseconds. For the button pressing function we simply needed to determine what data the buttons were sending and how many pieces of information to receive from the esplora.

Design

To design these functions we used common knowledge conversion functions to first convert milliseconds to seconds which is just a factor of 1000 from milliseconds, then to convert seconds to minutes. To determine the remaining seconds and remaining milliseconds we used the modulus function. The button mapping was a very simple addition of buttons pressed as the buttons either returned a 1 or 0 with the exception of the slider which we used a simple if statement to check if there was a positive value and is so we set it to 1.

Testing

To test our conversion functions we used 2 test cases as shown in screen shot 4-4. These 2 tests were done by hand to check our functions as both passed we implemented them in our functions. Beyond that we tested our button checking function by running it as a simple addition and found that the slider returned a scale of values. To fix this we used a simple if statement concerning that value.

Comments

The biggest comment I have is to mention the trouble we had with the esplora as we were overthinking the input and that the esplora.exe did most of the work we initially thought we were supposed to do and that we need to make sure we call the exact number of variables we want or the returned data is unpredictable and often useless.

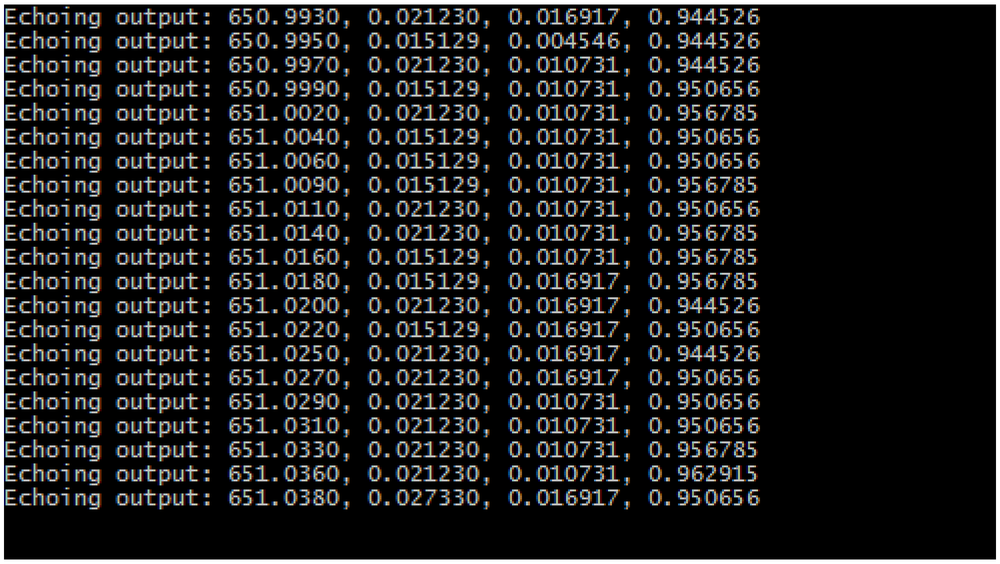
Source Code

```
/* Lab 4 Wrapper Program */

#include <stdio.h>
#include <math.h>

#define TRUE 1

/* Put your function prototypes here */
double mag(double, double, double);
int minutes(int);
int seconds(int);
int millis(int);
int main(void) {
    /* DO NOT MODIFY THESE VARIABLE DECLARATIONS */
      int t;
      double  ax, ay, az;


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

/* CODE SECTION 0 */
      double secConvert = t/1000.00;
            printf("Echoing output: %7.4lf, %lf, %lf, %lf\n", secConvert, ax,
ay, az);

/*    CODE SECTION 1 */
            printf("At %d ms, the acceleration's magnitude was: %lf\n",
                  t, mag(ax, ay, az));
/*    CODE SECTION 2*/

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

      }

return 0;
}
double mag(double ax, double ay, double az)
{
double tempAns =(ax * ax) + (ay * ay) + (az * az);
double ans = sqrt(tempAns);
return ans;
}
int seconds(int t)
{
      int timeTemp = t;
      int tempSec = timeTemp / 1000;
      int sec = tempSec % 60;
      return sec;
}
int millis(int t)
```

```
{
      int timeMils = t;
      int mils = timeMils % 1000;
      return mils;
}
int minutes(int t)
{
    int timeTemp = t;
      int tempMin = timeTemp / 1000;
      int min = tempMin / 60;
      return min;
}

/* Lab 4 Wrapper Program */

#include <stdio.h>
#include <math.h>

#define TRUE 1

int numButtons(int, int, int, int, int, int);

int main(void) {
    /* DO NOT MODIFY THESE VARIABLE DECLARATIONS */
      int t;
      double  ax, ay, az;
      int B1, B2, B3, B4, B5, B6;

      while (TRUE) {


            scanf("%d,%d,%d,%d,%d,%d", &B1, &B2, &B3, &B4, &B5, &B6);


            printf("Number of buttons = %d\n", numButtons(B1, B2, B3, B4, B5,
B6));
            fflush(stdout);




      }

return 0;
}
int numButtons(int B1, int B2, int B3, int B4, int B5, int B6)
{
      if(B6 > 0)
      {
            B6 = 1;
      }
      int numButtons = B1 + B2 + B3 + B4 + B5 + B6;
      return numButtons;
}
```

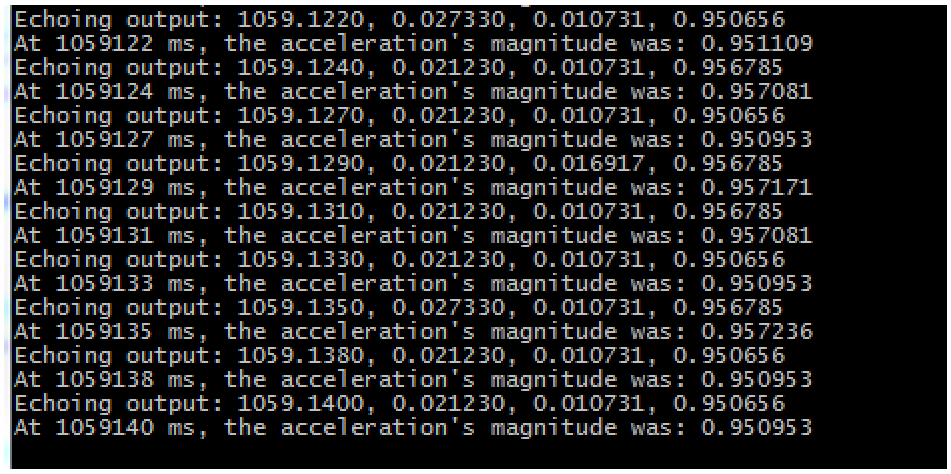
Screen Shots

Screen shot 4-1



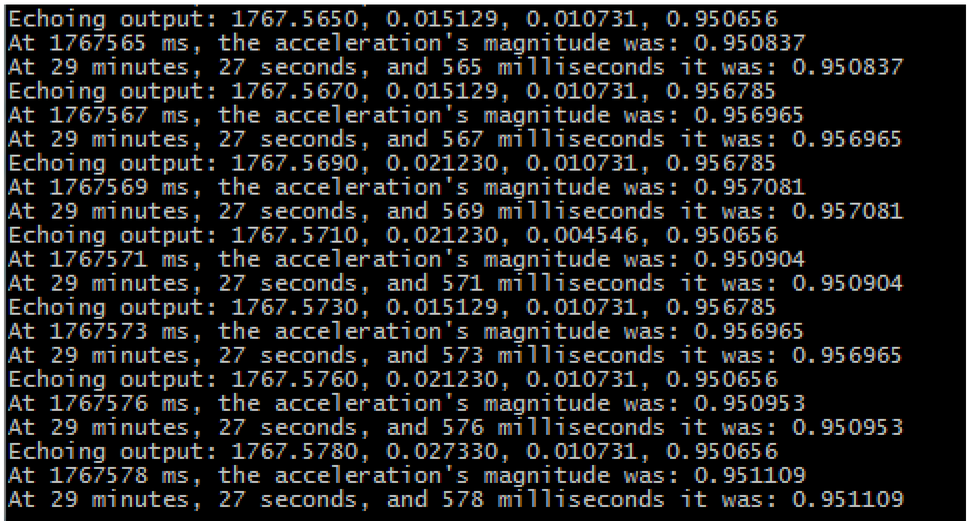
```
Echoing output: 650.9930, 0.021230, 0.016917, 0.944526
Echoing output: 650.9950, 0.015129, 0.004546, 0.944526
Echoing output: 650.9970, 0.021230, 0.010731, 0.944526
Echoing output: 650.9990, 0.015129, 0.010731, 0.950656
Echoing output: 651.0020, 0.021230, 0.010731, 0.956785
Echoing output: 651.0040, 0.015129, 0.010731, 0.950656
Echoing output: 651.0060, 0.015129, 0.010731, 0.950656
Echoing output: 651.0090, 0.015129, 0.010731, 0.956785
Echoing output: 651.0110, 0.021230, 0.010731, 0.950656
Echoing output: 651.0140, 0.021230, 0.010731, 0.956785
Echoing output: 651.0160, 0.015129, 0.010731, 0.956785
Echoing output: 651.0180, 0.015129, 0.016917, 0.956785
Echoing output: 651.0200, 0.021230, 0.016917, 0.944526
Echoing output: 651.0220, 0.015129, 0.016917, 0.950656
Echoing output: 651.0250, 0.021230, 0.016917, 0.944526
Echoing output: 651.0270, 0.021230, 0.016917, 0.950656
Echoing output: 651.0290, 0.021230, 0.010731, 0.950656
Echoing output: 651.0310, 0.021230, 0.010731, 0.950656
Echoing output: 651.0330, 0.021230, 0.010731, 0.956785
Echoing output: 651.0360, 0.021230, 0.010731, 0.962915
Echoing output: 651.0380, 0.027330, 0.016917, 0.950656
```

Screen shot 4-2



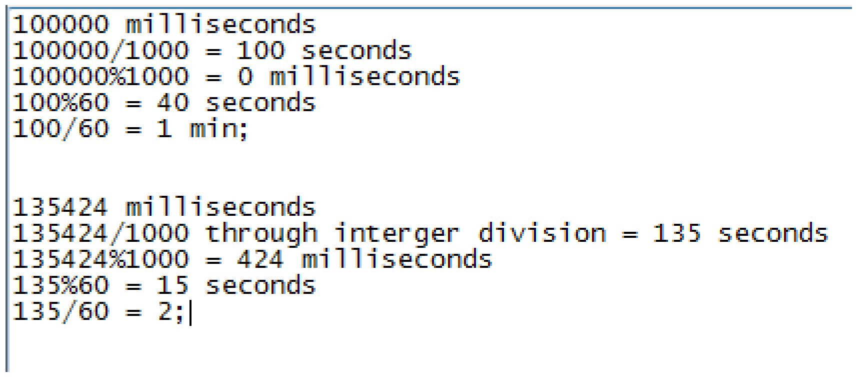
```
Echoing output: 1059.1220, 0.027330, 0.010731, 0.950656
At 1059122 ms, the acceleration's magnitude was: 0.951109
Echoing output: 1059.1240, 0.021230, 0.010731, 0.956785
At 1059124 ms, the acceleration's magnitude was: 0.957081
Echoing output: 1059.1270, 0.021230, 0.010731, 0.950656
At 1059127 ms, the acceleration's magnitude was: 0.950953
Echoing output: 1059.1290, 0.021230, 0.016917, 0.956785
At 1059129 ms, the acceleration's magnitude was: 0.957171
Echoing output: 1059.1310, 0.021230, 0.010731, 0.956785
At 1059131 ms, the acceleration's magnitude was: 0.957081
Echoing output: 1059.1330, 0.021230, 0.010731, 0.950656
At 1059133 ms, the acceleration's magnitude was: 0.950953
Echoing output: 1059.1350, 0.027330, 0.010731, 0.956785
At 1059135 ms, the acceleration's magnitude was: 0.957236
Echoing output: 1059.1380, 0.021230, 0.010731, 0.950656
At 1059138 ms, the acceleration's magnitude was: 0.950953
Echoing output: 1059.1400, 0.021230, 0.010731, 0.950656
At 1059140 ms, the acceleration's magnitude was: 0.950953
```

Screen shot 4-3



```
Echoing output: 1767.5650, 0.015129, 0.010731, 0.950656
At 1767565 ms, the acceleration's magnitude was: 0.950837
At 29 minutes, 27 seconds, and 565 milliseconds it was: 0.950837
Echoing output: 1767.5670, 0.015129, 0.010731, 0.956785
At 1767567 ms, the acceleration's magnitude was: 0.956965
At 29 minutes, 27 seconds, and 567 milliseconds it was: 0.956965
Echoing output: 1767.5690, 0.021230, 0.010731, 0.956785
At 1767569 ms, the acceleration's magnitude was: 0.957081
At 29 minutes, 27 seconds, and 569 milliseconds it was: 0.957081
Echoing output: 1767.5710, 0.021230, 0.004546, 0.950656
At 1767571 ms, the acceleration's magnitude was: 0.950904
At 29 minutes, 27 seconds, and 571 milliseconds it was: 0.950904
Echoing output: 1767.5730, 0.015129, 0.010731, 0.956785
At 1767573 ms, the acceleration's magnitude was: 0.956965
At 29 minutes, 27 seconds, and 573 milliseconds it was: 0.956965
Echoing output: 1767.5760, 0.021230, 0.010731, 0.950656
At 1767576 ms, the acceleration's magnitude was: 0.950953
At 29 minutes, 27 seconds, and 576 milliseconds it was: 0.950953
Echoing output: 1767.5780, 0.027330, 0.010731, 0.950656
At 1767578 ms, the acceleration's magnitude was: 0.951109
At 29 minutes, 27 seconds, and 578 milliseconds it was: 0.951109
```

Screen shot 4-4



```
100000 milliseconds
100000/1000 = 100 seconds
100000%1000 = 0 milliseconds
100%60 = 40 seconds
100/60 = 1 min;


135424 milliseconds
135424/1000 through interger division = 135 seconds
135424%1000 = 424 milliseconds
135%60 = 15 seconds
135/60 = 2;
```