**EXDROP**

**LAB #6**

**SECTION AA**

**SUBMITTED BY:**

**ANDREW HANCOCK**

**SUBMISSION DATE:**

**10/23/16**

**Lab 6 part 1 code**

// lab7.c

//

// This is the outline for your program

// Please implement the functions given by the prototypes below and

// complete the main function to make the program complete.

// You must implement the functions which are prototyped below exactly

// as they are requested.

#include <stdio.h>

#include <math.h>

#define PI 3.141592653589

//NO GLOBAL VARIABLES ALLOWED

//PRE: Arguments must point to double variables or int variables as appropriate

//This function scans a line of explore data, and returns

// True when left button is pressed

// False Otherwise

//POST: it modifies its arguments to return values read from the input line.

int read\_acc(double\* a\_x, double\* a\_y, double\* a\_z, int\* time, int\* Button\_UP, int\* Button\_DOWN, int\* Button\_LEFT, int\* Button\_RIGHT);

// PRE: -1.0 <= x\_mag <= 1.0

// This function computes the roll of the esplora in radians

// if x\_mag outside of -1 to 1, treat it as if it were 1 or -1

// POST: -PI/2 <= return value <= PI/2

double roll(double x\_mag);

// PRE: -1.0 <= y\_mag <= 1.0

// This function computes the pitch of the esplora in radians

// if y\_mag outside of -1 to 1, treat it as if it were 1 or -1

// POST: -PI/2 <= return value <= PI/2

double pitch(double y\_mag);

// PRE: -PI/2 <= rad <= PI/2

// This function scales the roll value to fit on the screen

// POST: -39 <= return value <= 39

int scaleRadsForScreen(double rad);

// PRE: num >= 0

// This function prints the character use to the screen num times

// This function is the ONLY place printf is allowed to be used

// POST: nothing is returned, but use has been printed num times

void print\_chars(int num, char use);

//PRE: -39 <= number <=39

// Uses print\_chars to graph a number from -39 to 39 on the screen.

// You may assume that the screen is 80 characters wide.

void graph\_line(int number);

int main()

{

double x, y, z; // magnitude values of x, y, and z accelerations

int b\_Up, b\_Down, b\_Left, b\_Right; // variables to hold the button statuses

double roll\_rad, pitch\_rad; // value of the roll measured in radians

int scaled\_valueP, scaled\_valueR, return1 = 1, return2; // value of the roll adjusted to fit screen display

int t, b\_Extra1, b\_Extra2, s, Setting;

//insert any beginning needed code here

return2 = 0;

int i = 0,k = 0,p = 0;

do

{

// Get line of input

scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d", &t, &x, &y, &z, &b\_Down, &b\_Up, &b\_Left, &b\_Right, &b\_Extra1, &s);

//printf("%d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d\n", t, x, y, z, b\_Down, b\_Up, b\_Left, b\_Right, b\_Extra1, s);

// check for button input

return1 = read\_acc(&x, &y, &z, &t, &b\_Up, &b\_Down, &b\_Left, &b\_Right);

// calculate roll and pitch

roll\_rad = roll(x);

if(roll\_rad == 0){

roll\_rad = 1.0;

}

pitch\_rad = pitch(y);

if(pitch\_rad == 0){

pitch\_rad = 1.0;

}

//printf("%lf\n", roll\_rad);

// switch between roll and pitch(up vs. down button)

if(p == 0){

return1 = 1;

p++;

}

//to switch to roll

if(return1 == 1){

return2 = 1;

}

//to switch to pitch

if(return1 == 2){

return2 = 2;

}

/\*

if(return2 == 2){

printf("pitch\n");

}else if(return2 == 1){

printf("roll\n");

}

\*/

// Scale your output value

scaled\_valueP = scaleRadsForScreen(pitch\_rad);

scaled\_valueR = scaleRadsForScreen(roll\_rad);

/\*

if(return2 == 2){

printf("%d\n", scaled\_valueP);

}else if(return2 == 1){

printf("%d\n", scaled\_valueR);

}

\*/

// Output your graph line

if(return2 == 2){

graph\_line(scaled\_valueP);

}else if(return2 == 1){

graph\_line(scaled\_valueR);

}

if(return1 == 3){

break;

}

fflush(stdout);

} while (1); // Modify to stop when left button is pressed

return 0;

}

int read\_acc(double\* a\_x, double\* a\_y, double\* a\_z, int\* time, int\* Button\_UP, int\* Button\_DOWN, int\* Button\_LEFT, int\* Button\_RIGHT){

scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d", &time, &a\_x, &a\_y, &a\_z, &Button\_DOWN, &Button\_UP, &Button\_LEFT, &Button\_RIGHT);

if(Button\_UP == 1){

return 1;

}else if(Button\_DOWN == 1){

return 2;

}else if(Button\_LEFT == 1){

return 3;

}else{

return 0;

}

}

double roll(double x){

double roll = asin(x);

if(roll > 1){

return 1;

}else if(roll < -1){

return -1;

}

else if(isnan(asin(x)) != 0 ){

return 0;

}

return roll;

}

double pitch(double y){

double pitch = asin(y);

if(pitch > 1){

return 1;

}else if(pitch < -1){

return -1;

}

else if(isnan(asin(y)) != 0 ){

return 0;

}

return pitch;

}

int scaleRadsForScreen(double rad){

int rads = rad \* 39;

return rads;

}

void print\_chars(int num, char use){

if(use == '0'){

printf("%40c\n", use);

}

if(use == 'r'){

for(int spaces = 39;spaces > 0;spaces--){

printf(" ");

}

for(int i = num;i > 0;i--){

printf("%c", use);

}

printf("\n");

}

if(use == 'l'){

for(int spaces = (39 - fabs(num));spaces > 0;spaces--){

printf(" ");

}

for(int i = fabs(num);i > 0;i--){

printf("%c", use);

}

printf("\n");

}

}

void graph\_line(int number){

char toUse;

if(number < 0){

toUse = 'l';

}else if(number > 0){

toUse = 'r';

}else{

toUse = '0';

}

print\_chars(number, toUse);

}

**Questions and Experiments**

1. Value \* 39. Since the max the value could go up to was 1 the max the returned value could be is 39.
2. Each letter represents approximately 1.5 degrees. Near the limits, the equation returns nan. In order to accommodate for this the equations just returns 1 if it has a non-numerical answer.

**Problem**

The program needs to represent the pitch or roll in the console in terms of up to 39 letters in each direction. The equation that returns the value is in radians and needs to be scaled up. The program needs to stop when the left button is pressed. The program needs to be able to switch between pitch and roll with the press of a button. The program needs to output a new line of code each time data is read in. It also needs to output the correct letter depending on which direction the Arduino is turned.

**Analysis**

The program has to be written using the functions given in the skeleton code and no global variables are allowed. The inputs are the buttons, accelerometer values, and time from the Arduino. The program will take the accelerometer values and put them through equations to find the angle that the Arduino is turned. The equations will return the value in radians so it needs to be scaled up and then given to the function that will output the lines in the console. Sometimes the equation will return non-numerical values so the program needs to know to instead of stopping, return 1. Also, although the radian value can go higher than 1, it will always return 1 unless it is less.

**Design**

First, I have all of the prototypes. In my main function the first thing that happens is the initialization of the variable that will be needed. No global variables are allowed. Next, the program is put into a do-while loop that will end when the left button is pressed. This will then end the program. The first thing in the do-while loop is a scan statement reading in data from the Arduino. Next I have a call to the read\_acc function which checks for button input. It returns 0, 1, 2 or 3 based on when certain buttons are pressed. Next, my program calculates both pitch and roll. It could be improved to only calculate one, which I guess would probably make it faster, but I didn’t think of it at the time. In the end only one is used depending on whether it wants pitch or roll. When my program first starts I have a piece of code that automatically puts it in roll mode. This happens because when nothing is pressed in an iteration of the do-while loop the read\_acc function just returns 0. The piece of code increments by 1 so that it is never executed again. Next the values for pitch and roll, which are in radians, are both sent to the scaleRadsForScreen function, which just multiplies them by 39 since the max they can be is 1. Next, I have an if statement that, depending on the mode, either graphs the roll or pitch. Finally, I have an if statement that will break the whole loop if the left button is pressed.

**Testing**

I had to test my program a lot during the process of building it. A lot of times it was just to make sure I was getting the correct value of either radians or scaled radians for the screen. Testing the graphing function went relatively fast. I just had to make sure it was outputting the correct letter on the correct side at the correct time. When I tested how the button switched the mode, I just had the program output either pitch or roll for whatever mode it was on.

**Comments**

As I said in the design section, I think I could make my program faster by instead of calculating both radian values and both scaled radian values, only calculate the one it’ll need. One of the major problems I encountered during the lab was getting the program to switch modes and also getting it to get put into roll mode initially. I was really happy with my solution and it also works well. Another problem I encountered was when the function returned nan, but that was easily fixed by using the function isnan, which I didn’t know existed until this lab.