|  |  |
| --- | --- |
| \\cetus.ece.missouri.edu\users\fischerjd\Desktop\Current Semester\common\logo-clear.png | University of Missouri – Columbia  Department of Electrical and Computer Engineering  *“A Proud Tradition, Engineering the Future!”* |

Following Random Directions Using a Thump Joystick

ECE 4220  Final Project Proposal

May 8, 2014

Abdullah Albelaiys, Electrical and Computer Engineering

— Course Instructor —

Dr. Guilherme DeSouza, PhD

Department of Electrical and Computer Engineering, University of Missouri

**Objectives and Project Description**

The objective of this project is to perform the different skills that we learned throughout the semester. Also, getting familiar with adding new hardware components to the board in the lab. Ideally, the project should implement joystick that follow random arrow heads generated in the screen. However, due to the concerns that the thumb joystick will not work with the board, I used the buttons on the auxiliary boards as the hardware components. Three buttons are corresponding to three different direction, left most button is for left direction, middle button for up direction, and the right most button corresponds to the right direction.



Figure Thumb joystick

**Implementation**

I went through many ideas and approaches in order to accomplish the goal of the project. However, I am describing two of those approaches. The first approach is to create two different files. The first one is the kernel module for the project. In the kernel module, I am setting up the real time tasks and interrupts for the buttons, also, the FIFO in order to send the action of the button to the other file which is the user space file. In the user space file, I am asking the user of the number of rounds that he/she want to play. Each round corresponds to one random direction. So having five rounds will give us 5 random directions. After generating this random directions, am waiting on the kernel to responds. Am getting the information from the kernel, and deciding in the user space program whether the button pushed is appropriate or not. If so, the user will see a greeting message and get a point, otherwise, it will deducts a point. Unfortunately, this approach was not implemented correctly. So I switched to the other approach which is having only one file, where we do mainly the same thing as in the user space file described above, the only different is, we have all buttons detected in the same file. After generating a certain direction, we let the program gives the user a time window to hit the button, if the button is pushed, a certain light will turns on just to make sure that the button is detected, and if the button being pushed is the right one, the user again will get a greeting message and a point, or vice versa.



Figure 2 A flowchart of the project

**Experiments and Results**

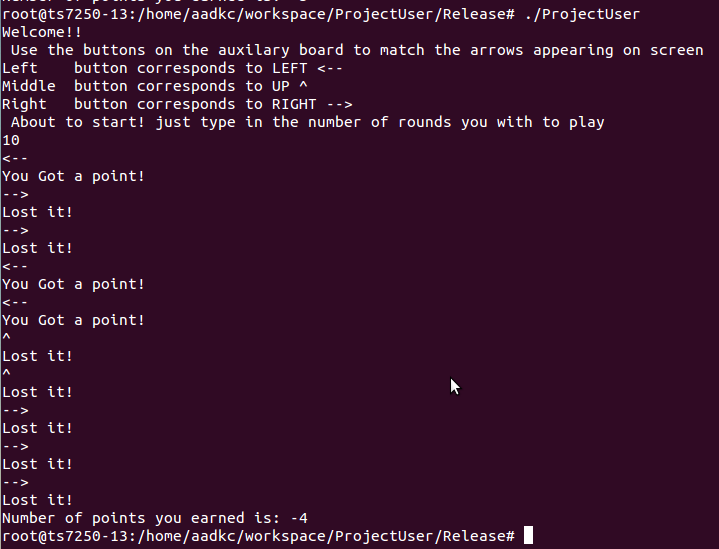
The way I tested the program is by dividing up the work into stages. The first stage is to code the generating of the arrows and print them. Simply like that to make sure that this part is done. The next step, is to set up the buttons, also test them to make sure they are mapped correctly. After that, I implemented the time window that the program waits on the user, and start hitting the buttons whenever I see corresponding arrow head on the screen. I ran the program for up to 9000 rounds, but after taking out the usleep function. The reason for testing this huge number is to see if the program is able to detect the push button at any time or not. At the beginning the program was counting points regarding of the arrow and if the button is pressed or not. After that, the program is taking off points all the time since it cannot detect the button within any window of time. The implemented program failed on detecting the buttons being pressed, however, an ideal case is shown in the demonstration videos, and also in the screenshot below.

Figure Ideal Output

**Discussion**

First of all, after investigating on the joystick, it needs a 5 pin header in order for it to fit into a breadboard. After that, I could not find a way to assign the pins on the board to the terminals of the thumb joystick. The only solution to this problem is to have the already installed push buttons on the auxiliary board. The second problem I faced was in exchanging information between the kernel and the user space program, after many trials, I switched to the second approach in order to have something to present. The first approach however makes more sense and apply most of the knowledge we gained through the semester. Unfortunately, I could not find the way in order to employ this knowledge to get a presentable result. I learned in this project that we can have many approaches to one problem, however, not all of them are effective the same way. Some solutions may have advantages over others.

**Code Section**

Code is attached.

**Conclusion**

For conclusion, the use of the push buttons were implanted as the directions of the thumb joystick. Also, the head arrows were generated and printed on the screen. One crucial and remarkable improvement to the project is to try and implement the project as the first approach. Having to implement it is this way will assure getting the desired results. Also, I can look for different joysticks that can be attached to the board in the lab in order to use it instead of the push buttons.

**Appendix A**

**Code**

**/\***

**============================================================================**

**Name : User**

**Author : Abdullah Albelaiys**

**Description : Arrows Game User**

**============================================================================**

**\*/**

**// call libraries**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <rtai.h>**

**#include <unistd.h>**

**#include <rtai\_lxrt.h>**

**#include <fcntl.h>**

**#include <sys/time.h>**

**#include <sys/types.h>**

**#include <semaphore.h>**

**#include <unistd.h>**

**#include <sys/mman.h>**

**#include <sys/stat.h>**

**#define ROUND 100**

**int main (void) {**

**// declare variables**

**unsigned long button;**

**int fd, rounds = 0;**

**unsigned long \*ptr;**

**unsigned long \*pfdr;**

**unsigned long \*pfddr;**

**unsigned long \*pbdr;**

**unsigned long \*pbddr;**

**int numrounds = 0;**

**int arrow;**

**int count = 0;**

**fd = open("/dev/mem",O\_RDWR | O\_SYNC ); //opens up special file /dev/mem**

**ptr = mmap(NULL, getpagesize(), PROT\_READ|PROT\_WRITE, MAP\_SHARED, fd, 0x80840000); // mapping**

**pfdr = (unsigned long \*) ((char \*) ptr + 0x04);//offsets to port B = 0x80840004**

**pfddr = (unsigned long \*) ((char \*)pfdr + 0x10); //offsets to DDR for port B**

**pbdr = ptr + 0x01; //pbdr = 0x80840004**

**pbddr = ptr + 0x05; //pbdr = 0x80840014**

**\*pbddr &= 0xFFFFFFFE; //input**

**\*pbddr |= 0xE0; //output**

**\*pbdr &= 0xFFFFFF1F; //turns off all lights**

**button = \*pfdr;**

**printf("Welcome!!\n Use the buttons on the auxilary board to match the arrows appearing on screen\n");**

**printf("Left button corresponds to LEFT <--\n");**

**printf("Middle button corresponds to UP ^\n");**

**printf("Right button corresponds to RIGHT -->\n");**

**printf(" About to start! just type in the number of rounds you with to play\n");**

**scanf("%d", &rounds);**

**while (numrounds < rounds) {**

**srand(time(NULL));**

**arrow = rand()% 3 + 1; // to generate a random number from 1 to 3**

**if (arrow == 1) { // if the random number is one, the print UP arrow**

**printf("^\n");**

**usleep(1000000);**

**if (((button & 0X01) == 0x00)) {// Left button**

**\*pbdr |= 0x20; // just to make sure the button is pressed**

**msync(ptr, getpagesize(), MS\_ASYNC);**

**sleep(1); //wait**

**\*pbdr = \*pbdr & 0xFFFFFF1F;**

**printf("You Got a point!\n");**

**count ++;**

**}**

**else {**

**printf("Lost it!\n");**

**count --;**

**}**

**numrounds++;**

**}**

**else if ( arrow == 2) { // if the random number is 2, the print right arrow**

**printf("-->\n");**

**usleep(1000000);**

**if (((button & 0X04) == 0x00)) {// middle button**

**\*pbdr |= 0x40; // just to make sure the button is pressed**

**msync(ptr, getpagesize(), MS\_ASYNC);**

**sleep(1); //wait**

**\*pbdr = \*pbdr & 0xFFFFFF1F;**

**printf("You Got a point!\n");**

**count ++;**

**}**

**else {**

**printf("Lost it!\n");**

**count --;**

**}**

**numrounds++;**

**}**

**else if (arrow == 3){**

**printf("<--\n");**

**usleep(1000000);**

**if (((button & 0X10) == 0x00)) {// right button**

**\*pbdr |= 0x80; // just to make sure the button is pressed**

**msync(ptr, getpagesize(), MS\_ASYNC);**

**sleep(1); //wait**

**\*pbdr = \*pbdr & 0xFFFFFF1F;**

**printf("You Got a point!\n");**

**count ++;**

**}**

**else {**

**printf("Lost it!\n");**

**count --;**

**}**

**numrounds++;**

**}**

**}**

**printf("Number of points you earned is: %d\n", count);**

**return 0;**

**}**