EE357 Fall 2012 Final Project
Final Report

Due: 5 PM., 12/10/2012

Sangmook Johnny Jung and Sheldon Kwok
sangmooj@usc.edu and sheldonk@usc.edu

1. Introduction

1.1 Motivation

Our main motivation for this project was to implement a game on the Coldfire board that would present an enjoyable user experience. This desire came from our interest in Human Computer Interaction and we originally planned to use the Wii Nunchuk as the main input source to control our game. We felt that using the Wii Nunchuk would provide a natural interface for the game that we have implemented.

1.2 Background

For this project, we have created an implementation of the arcade game, Phoenix[1], released in 1980 by Atari, with the Coldfire MCF52259 board. This is essentially a classic two-dimensional shooter game where the joystick will control a ship that can shoot lasers at incoming spaceships coming from the top of a LCD screen. The objective of the game is to survive and kill incoming waves of spaceships using bullets and bombs. Our original plans were to use the Wii Nunchuk peripheral to control the ship in this game. Wiimote[2] and Wii Nunchuk[3][4][5] peripherals have been used in the past often (especially with the Arduino boards), but we have not seen any uses of them with the Coldfire MCF52259 board, so we thought that this would present an interesting challenge.

2. Implementation

Our initial plans were to implement this game using the Wii Nunchuk controller as the main input source. As shown below, the Wii Nunchuk combines several different input sources into one controller, and additionally provides a familiar game-playing experience for the user. The joystick on the front of the Nunchuk would control the ship's movement from left to right, and the large button on the back of the Nunchuk (the Z button) would be used for firing standard bullets. Additionally, using the built-in accelerometer in the Nunchuk, the player could vigorously shake the Nunchuk controller to activate a bomb.

Figure 1:
Wii Nunchuk and Controls

The difficult with this project mainly came from actually integrating the Wii Nunchuk with the Coldfire board. The Nunchuk connects to the Coldfire board using the I2C bus protocol[6][7] by using a special WiiChuck adapter[8][9] as shown below. The I2C bus protocol would then allow the Coldfire board to read in the Nunchuk joystick, button and accelerator data and this would be used as input for the game. However, as we quickly realized, setting up the Wii Nunchuk using I2C was not a trivial affair.

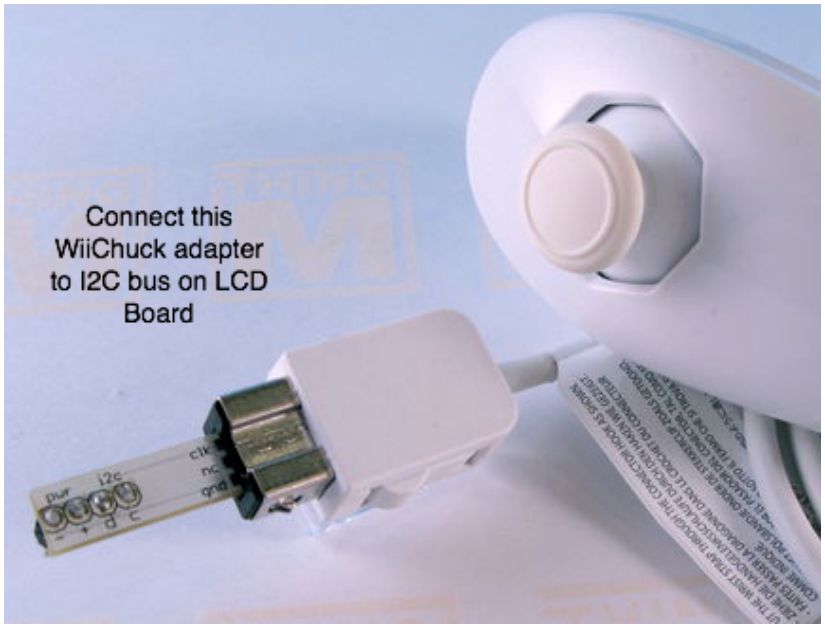

Figure 2:
WiiChuck and Wii Nunchuk

One of the main issues we had with setting up the Wii Nunchuk with the Coldfire board was that there were no online resources to refer to for the Coldfire board, and that the documentation was sparse at best. There were only a few resources for connecting the Nunchuk to the Arduino board[10][11][12][13]. In fact, Johnny was able to setup the Wii Nunchuk to his Arduino board with no problems, but it seems that no one has ever tried to connect the Nunchuk to the Coldfire board.

Initially, Johnny attempted to get the Wii Nunchuk working using the connections on the LCD module by utilizing existing code from the LCD lab, along with I2C code from previous labs and sample code given. However, he quickly realized that he didn't even know where to physically connect the Nunchuk! Upon receiving extensive help from the TA and other EE students, Johnny eventually figured out that the I2C bus protocol (clock and data) were located at ports A6 and A7 on the Coldfire board. In order to access these ports, a prototyping board was used as shown below. The Wii Nunchuk adapter was placed on this prototyping board, and various header pins and wires were soldered together to establish the

correct connections between the Wii Nunchuk adapter and the Coldfire board. Upon finishing this (and spending lots of time soldering), this prototype board was plugged into the Tower interface, and ultimately failed to work.

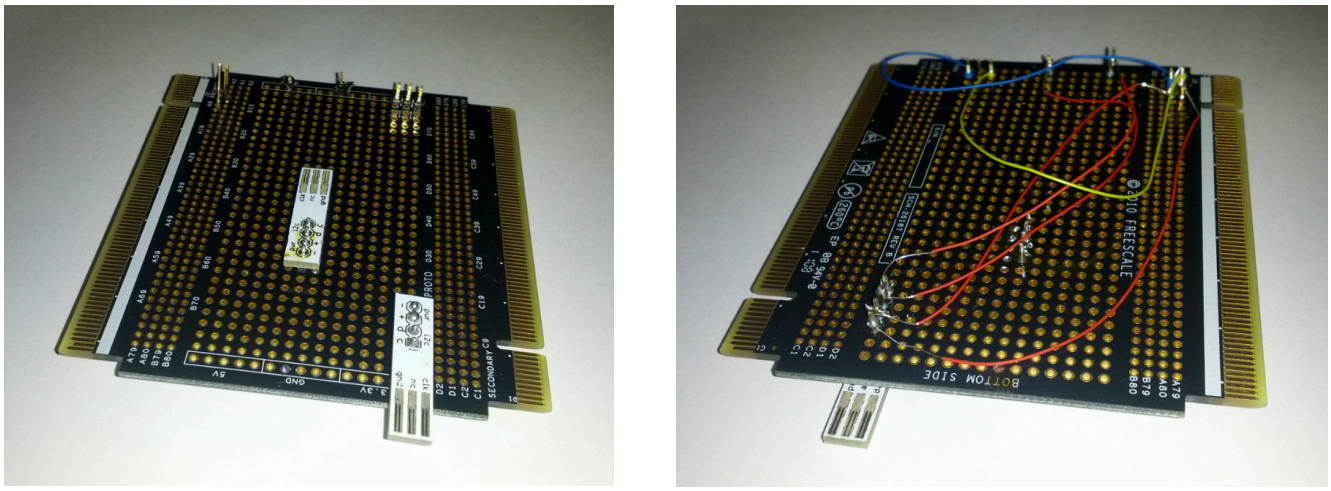

Figure 3: Prototyping Board with WiiChuck Adapter

Afterwards, to debug the Wii Nunchuk, Johnny connected up the I2C connection wires to an oscilloscope to see if he could get the I2C signal to be recognized. Despite Johnny's efforts and using all sorts of combinations of sample code and wiring of the Nunchuk connection, nothing showed any semblance of an I2C connection being established. At this point, Johnny could've spent more time trying to establish an I2C connection (perhaps with GPIO registers on the Coldfire board), but due to time constraints we decided to stop trying to use the Wii Nunchuk as the input source. The possibilities of error were quite large, and it was very difficult to debug and figure out what the issue was. It could've ranged anywhere from an issue with soldering and wiring, or perhaps even a software issue. It could've even been a defect or something that we failed to recognize with the Coldfire board!

Luckily, the features of the Wii Nunchuk were all present in one form or another on the LCD module and the Coldfire board. A joystick was present on the LCD module, and an accelerometer and several buttons were present on the Coldfire board. At this point, using the lab code and the sample code provided by the TA, Johnny was easily able to set up the joystick, button press, the accelerometer, the LCD output for this project, and the skeleton code needed to hold all of this input and output interaction.

For this project, the joystick on the LCD module was used to the control the player's ship on the screen. After establishing the connection with the Coldfire board, the joystick returned simple boolean values stating whether it was selected in a certain direction (i.e. up and down). In our project, we chose to focus on whether the user moved the joystick left and right, as we decided that our users wouldn't need to move up and down in the game. We also used the accelerometer in a useful way for our project. The accelerometer constantly returned xyz values related to its relative position from initialization. At first, the sensitivity of the accelerometer module seemed to be a problem, as the values would constantly fluctuate even when the board appeared to be still. However, we decided to take advantage of this and use these values as a random number generator to determine the game behavior of the incoming spaceships. We felt that this would make the game more exciting, as this is probably more random than any pseudo-random number generator we could come up with. Shaking the accelerometer also would cause the user's bomb to be released and wipe out all the enemies on the screen.

For the output for this project, we used the LCD module as the primary interface for the game to display all of the game events and graphics. A few of the LCD module functions were already provided for this project such as grphText, grphUpdate, and grphErase.

grphText was used to draw text to the screen buffer with xy specifications, font size, and the text we wanted to display. Some of the available fonts were initially commented out in the fonts header file and therefore needed to be uncommented. grphUpdate simply drew the graphics buffer to the screen with the latest drawing updates. This allows multiple characters to be drawn to the screen at once so the game appears to be more fluid. The grphErase function simply erased everything on the screen.

In the end we were able to successfully implement the Phoenix game on the Coldfire board, albeit without usage of the Wii Nunchuk. Despite this, we believe that the game provides a game experience that is enjoyable and takes advantage of various input sources, such as the joystick, accelerometer and several button presses.

3. Conclusion

In this project, we have learned about various aspects of working with hardware and microcontrollers. Johnny learned a lot about building and interfacing hardware components and how difficult it can be to setup communication between hardware. Sheldon learned a lot about the memory and processing constraints of the microcontroller. Due to the limited memory of the Coldfire board, we did not make any new variables after the program was initialized to avoid memory leaks. For example, random numbers were acquired from values that were already available. Both of us collectively realized that programming a microcontroller is often not very straightforward and may require various clever tricks and "hacks" to get the desired functionality.

The most challenging aspect of this project was definitely setting up the I2C bus protocol between the Wii Nunchuk and the Coldfire board. It was not easy to set up, as many hours were spent trying to establish this connection, which eventually did not work. The lack of documentation and perhaps even the lack of precedent made it prohibitively challenging to set this up. Another challenging aspect that we faced was the slow Coldfire processor and LCD module. Changes and updates of frames were very noticeable and they definitely impacted game performance and the user experience. Additionally, reading the joystick information also affected the frame rate. For this, we had to take special precautions to essentially optimize the frame rate.

Our main motivation was to create a game that was playable, but more importantly user-friendly and easy to play. We thus evaluated ourselves on creating a game that we ourselves would want to play. Despite not being able to get the Wii Nunchuk working, we were able to create a game that was completely playable using the joystick, accelerometer and button

process. We also included a title screen with a menu animation and pause button, as most games have in the present day. We believe that we have met our initial goals to create an intuitive and enjoyable game experience.

If given more time, many changes could be made to make this game better. Initially, it would be great if we could get the Wii Nunchuk working with the Coldfire board. We still believe that this would provide a superior gaming experience and that this would be the next logical step for us. Additionally, other input sources may be considered as well. Getting the actual WiiMote (using an IR Camera) to interface with the Coldfire board would probably be a very difficult and rewarding challenge. Also, more optimizations to the code could've probably been made to minimize frame rate issues. Additionally, more game features, such as a game shop or the concept of money, could be developed if given more time. In addition, more detailed graphics could have been made with pixel art to emulate the Space Invaders experience. With additional input devices, multiplayer functionality could have also been implemented easily as well to provide the nostalgic feeling of playing in an arcade with a friend.

4. References

[1] http://en.wikipedia.org/wiki/Phoenix_(video_game)

[2] http://johnnylee.net/projects/wii/

[3] http://www.windmeadow.com/node/42

[4] http://letsmakerobots.com/node/5684

[5] http://todbot.com/blog/2007/11/24/bionic-arduino-class-notes-3-4/

[6] http://wiibrew.org/wiki/Wiimote/Extension_Controllers/

[7] http://wiibrew.org/wiki/Wiimote/Extension_Controllers/Nunchuck

[8] https://www.sparkfun.com/products/9281

[9] http://todbot.com/blog/2008/02/18/wiichuck-wii-nunchuck-adapter-available/

[10] http://www.britishideas.com/2011/08/20/reading-a-wii-nunchuck-using-i2c/

[11] http://forum.allaboutcircuits.com/showthread.php?t=34422

[12] http://www.nerdkits.com/forum/thread/972/

[13] http://dangerousprototypes.com/2009/08/19/bus-pirate-wii-nunchuck-quick-guide/

Figure 1: http://express.howstuffworks.com/gif/autopsy-wii-nunchuk.jpg

Figure 2: http://todbot.com/blog/2008/02/18/wiichuck-wii-nunchuck-adapter-available/

Figure 3: Picture Taken by Johnny Jung

Sample Code from Blackboard and Sang Wook Do

Help with I2C + Wii Nunchuk from Sang Wook Do, Daniel Wong, and Professor Redekopp