Wii-B-Fit Adaptive Wii Remote

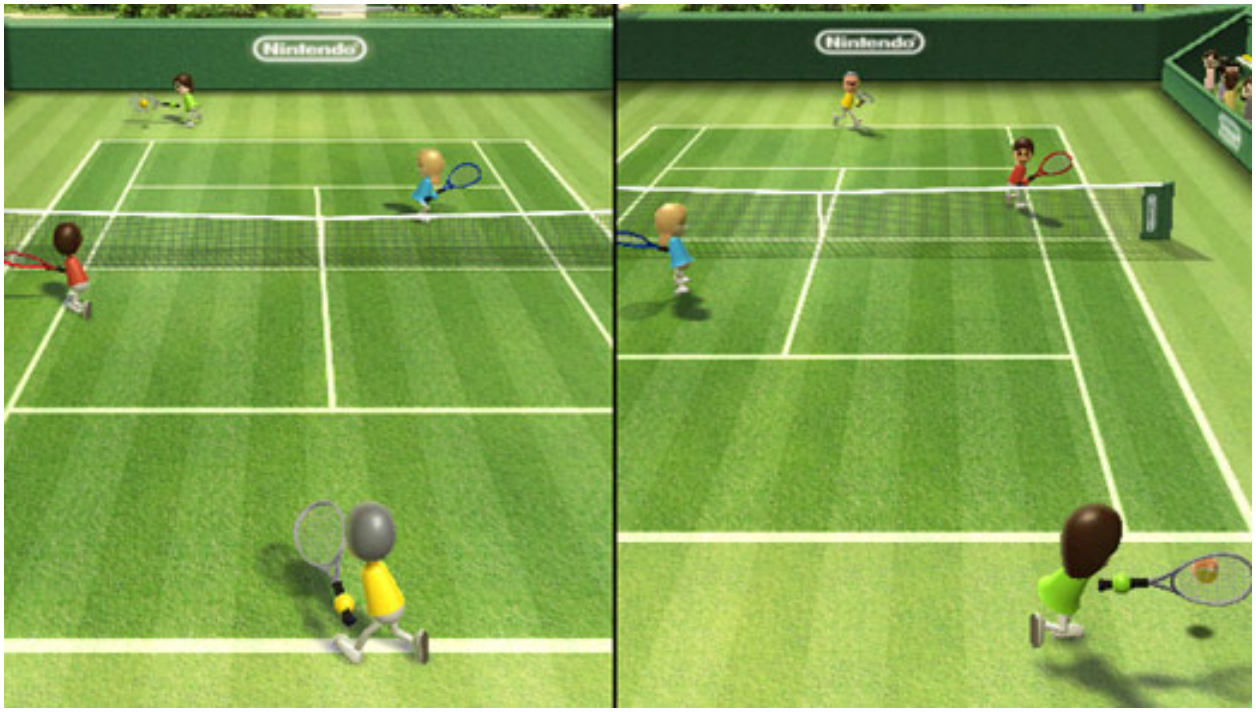

Figure 1. https://www.nintendo.com/games/detail/1OTtO06SP7M52gi5m8pD6CnahbW8CzxE

A Senior Project for

The Computer Engineering Department of

California Polytechnic State University, San Luis Obispo

By

Chris Lombardo

Advised by Lynne Slivovsky

March 21, 2014

Table of contents

1. Introduction

The goal of this project was to fix and supplement the original Wii-B-Fit system in order to get it running again and improve on the original design. Wii-B-Fit's original mission is to give people with paralysis the ability to participate in activities that they would not otherwise get to do, while creating a more fun, social, and independent lifestyle.Physical activity can help stimulate brain activity and is an integral part to a healthy lifestyle. Quadriplegia however makes it extremely hard to perform most forms of physical activity and severely limits the choices available to victims of the disease. The original project designed a system that allowed someone with partial paraplegia to interact with the Wii system using added sensors that would interact with a Wii controller. This system was completed for two of the Wii Sports games bowling and tennis. This project will also serve to showcase some of the diverse skills of a computer engineer. The project combines hardware and software, a fusion that epitomizes the computer engineering program. I will combine my C programming skills, my knowledge and experience with integrated systems, and my circuit construction skills in order to complete this project. This varied skillset allowed me to complete this project as my own one man team.

2. Client:

The client for this project Mike Ward is a partial quadriplegic who has only full movement with his head and limited movement with one of his arms. He is confined to a

wheelchair and cannot play the Wii using its standard control scheme Mr Ward was happy with and extensively used the original system until it failed.

3. Project Goals and Objectives:

The goal of this project is to return the system in a working state to the client. A secondary goal would be to supplement the original Wii-B-Fit system in order to improve on the original design. In order to accomplish this goal I set my main objective as diagnosing and fixing the cause of failure in the system. To improve the system I would explore the feasibility of converting the system to an Arduino based system.

4. Testing Of Previous System and Planning

a. Testing

I studied the original system architecture, design, and current condition to find the cause of failure. I sought to get a general overview of the current state of the system before I made any lasting plans. Immediately upon inspection I noticed melted wire and a burn spot on the original circuit board. After repeated attempts to power on the board failed I had to conclude that one or more integral components of the board had burned out. I next moved on to the three DACs and tested them with a simple Arduino based DAC tutorial that I found online for this particular DAC chipset.[1] The three DAC chips behaved normally and this allowed their reuse. Finally I set about

testing the ADXL330 accelerometer. Again I turned to an Arduino test program designed around this particular chip and found that it also still functioned. [2] With the main board out of action but the other parts of the system still operational I decided to implement an Arduino based solution to complete it.

5. System Architecture and Design

a. Overview

Since the goal of this project was mainly to return the system to a functional state I wanted to reuse as much of the previous system as possible. To this end the only piece of hardware I replaced was the broken Atmel board. The new Arduino Uno would serve the same purpose and have a much smaller footprint. The four buttons would interact directly with the Wiimote and function as larger more accessible versions of the Wiimote buttons connected with detachable jacks. The accelerometer is mounted on a hat worn by the user with users movement fed as X, Y, and Z outputs back to the Arduino. These signals are read by the Arduino then modulated to increase sensitivity before being fed to three output DACs. These DACs output back into the X, Y, and Z axis of the Wiimote to be read by the game.

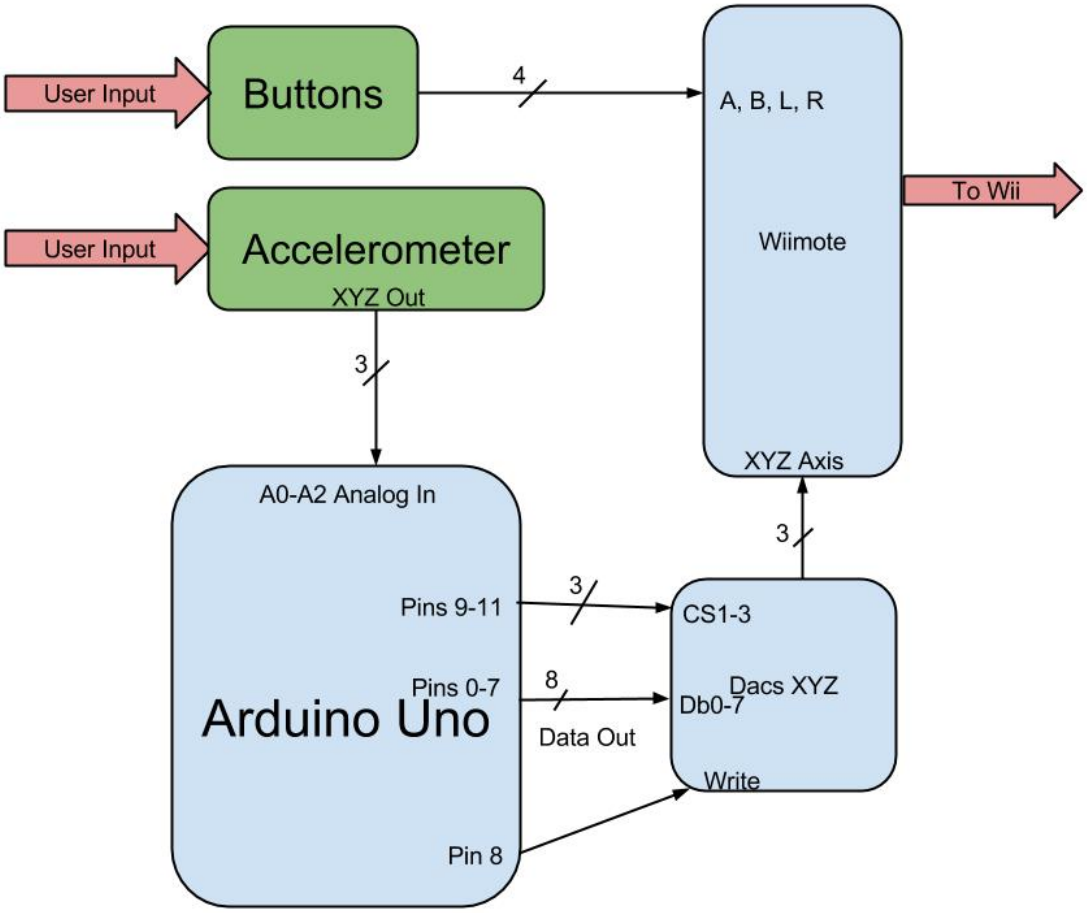

Figure 2.

b. Accelerometer

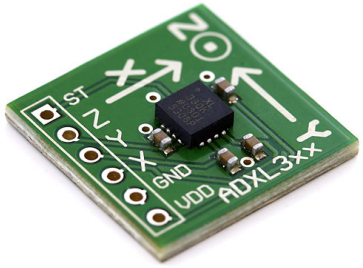

Figure 3. http://robosavvy.com/store/images/sparkfun/00692-03-L.jpg

The head mounted accelerometer has two power wires, a red Vcc wire set to 3.3V a black ground wire, both of wire are separate from the three data connections. The three data wires are connected on one header and correspond to the X-axis for the red wire, the Y-axis for the black wire, and the Z-axis for the green wire. These wires output an analog signal between 0-3 volts corresponding to -3g and +3g with a baseline around 1.5V depending on tolerances. [3]

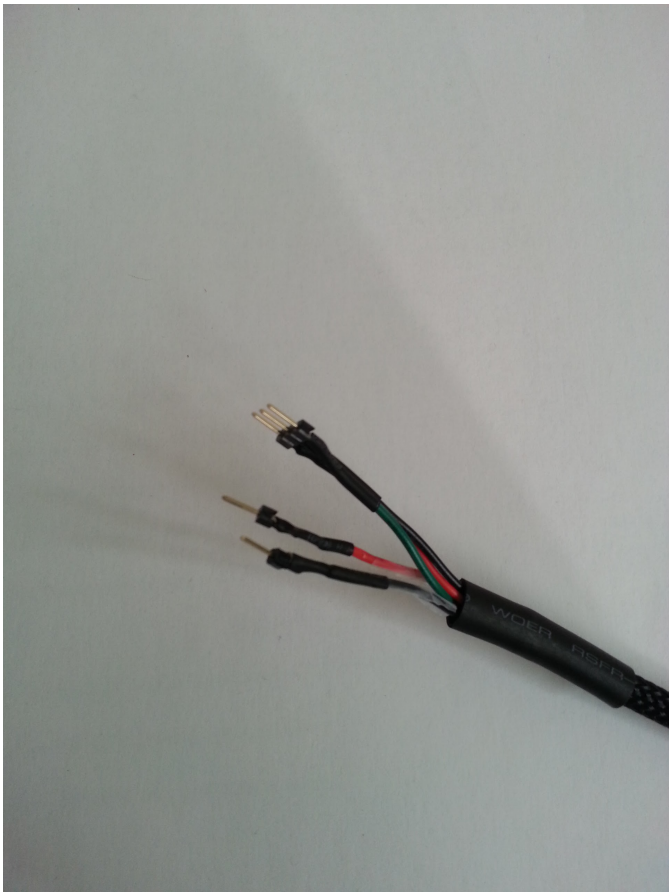

Figure 4. Wires from Accelerometer

c. Arduino

The Arduino is the main workhorse of the system as it powers all of the individual components of the system all well as reading the accelerometer inputs and

outputting their modulated results. The X-axis wire coming from the accelerometer(Red Wire) is connected to pin A0 with Y connected to A1 and Z connected to A2. The three voltages from the accelerometer are read using the built in analog to digital converter of the Arduino into a 10bit digital value with a reference voltage of 3.3 volts. The deviation of these readings from their normal rest is amplified so smaller motions can be used to simulate large swings of the Wiimote. This value is then mapped to an 8bit value and sent to the DACs through pins 0-7.

d. DAC

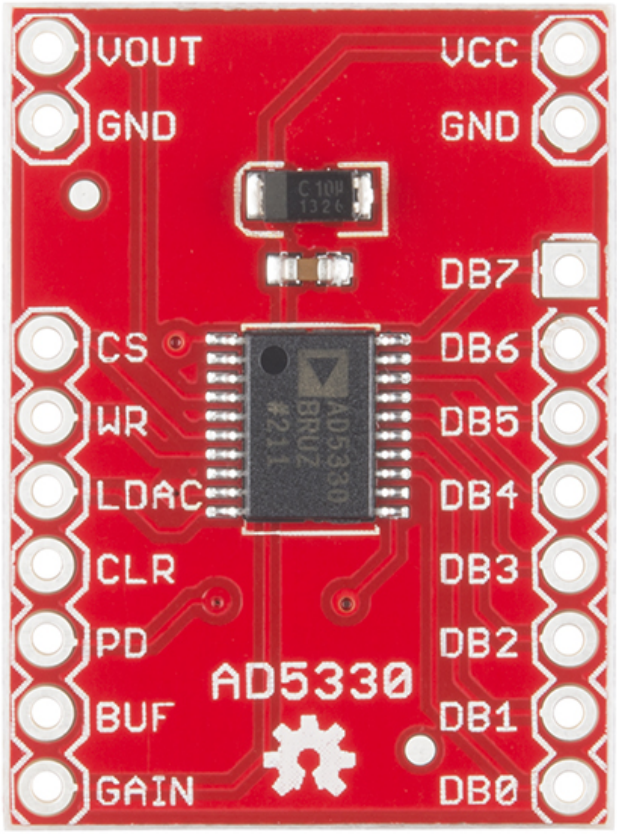

Figure 5. https://cdn.sparkfun.com//assets/parts/8/7/1/1/12082-04.jpg

There are three separate DACs each responsible for a separate axis. They have 8bit inputs DB0-7 which are all tied together meaning there are eight input lines but they are shared between all three DACs. GAIN is connected to ground to always be low and set gain to one. The BUF input is connected to Vcc so that it will not buffer. The clear pin along with the PD pin are also tied to Vcc to disable clearing and enable the chip respectively. These connections are all hardwired as they do not need to change with the only connections between the DACs and the Arduino being the WRITE pin and CS pins. To reduce the number of connections the DACs share a single write line but have three different selector lines corresponding to Arduino pins 9-11. The analog value is then output through vout to the Wiimote.

e. Software

The software takes in the accelerometer data and converts it to a 10bit value then boosts the difference between the read value and the normal value. This difference is multiplied by eight or ten in the case of the Z-axis and then added to the default value as shown by this equation ((Input-RestingValue)*8) + RestingValue. This value is then mapped to an 8bit value and sent to the DACs.

6. Construction

I reused the mounting for the DACs but with a conversion to an arduino based system the old project box was much larger than it needed to be so I bought a new one that was roughly half the size. I also kept the hat that the old system used as that was the one our client was used to but the wire connection between it and the box had deteriorated. I replaced the original felt and plastic wrapping with a snakeskin wire covering as well as soldered on header pins to the ends of the previously loose wire. I also added header pins to the Wiimote's wire connections which were lacking them previously. Several holes had to be drilled into the project box in order to allow the cables to the Wiimote and accelerometer head mount to pass through. In addition access was needed for the Arduino power cable and usb connection for

future tweaking of the firmware. The breadboard the DACs are mounted on along with the Arduino were then hot glued in place and the project box sealed.

7. Testing

a. Overview

Since this was mostly a user interface project most of the testing was done on the system as a whole with the results more based on the overall feel of the system.

b. Wii Sports Tennis

Since I first tested each of the components individually when trying to diagnose the problem with the original system I proceeded to full system testing after a cursory check of the different parts. With the wiimote powered and the buttons working I tried to play tennis but found that like the previous team who worked on this project the accelerometer is not sensitive enough to adequately pick up swings with your

head. To solve this problem I amplified the input signals difference from its resting value to increase sensitivity. I still found myself having difficulty getting swings recorded by the Wii. I experimented with different hard coded values for the DACs to see which combinations produced registered swings. I found for a normal right hand swing the output needs to be about 3 volts for the X and Y axis with the Z axis being close to .8-1.0V. A backhanded swing needs an X and Y axis value of close to 0 volts in addition to a Z axis value close to .8-1.0V. After finding these values I found it was too difficult to get a backhand swing with the current configuration so I switched the X-axis of the head based accelerometer to the Y-axis input of the Wii.This change also helped with bowling as it allows you to pull your head up to raise the ball instead of to the side. This tweak along with an increase in the sensitivity of the Y-axis allowed me to reliably detect normal and backhand swings.

c. Wii Bowling

Testing of the bowling proved much easier as after the tweaks from the tennis test the bowling was smooth although it does take some time to get used to moving your head as you would your arm while bowling.

8. Conclusion and Final Thoughts

Overall this was an excellent project to show off the diverse skillset of a computer engineer with electronic work combined with programming to make a finished product. Soldering along with rigorous testing coming were needed in equal parts over the course of this project and I don’t know if I would have been able to complete it without the unique Learn By Doing approach at Cal Poly. After finishing this

project even though it was not the flashiest I'm just glad to be able to positively contribute to our community.

9. Appendix

1. https://www.sparkfun.com/tutorials/160
2. http://arduino.cc/en/Tutorial/ADXL3xx
3. http://www.analog.com/en/mems-sensors/mems-inertial-sensors/adxl330/products/product.html
4. Previous group poster and whitepaper attached as pdfs.
5. Code

```
//Set up the Arduino Pin locations of the AD5330 Control Pins

#define  WR  8

#define  CS1 9

#define  CS2 10

#define  CS3 11

#define VOL 0x00

#define VOL1 0x00

#define VOL2 0xFF


const int xpin = A0;          // x-axis of the accelerometer

const int ypin = A1;          // y-axis

const int zpin = A2;          // z-axis

int AnalogInX = 0;

int AnalogInY = 0;

int AnalogInZ = 0;
```

```
int XBoost = 0;
int YBoost = 0;
int ZBoost = 0;

void setup();
void loop();

void setup()
{
  //All of the digital pins on the Arduino will be outputs to the AD5330
  for(int pin=0; pin<12; pin++)
  {
          pinMode(pin, OUTPUT);
  }
  analogReference(EXTERNAL);
  digitalWrite(CS1, HIGH);  //Set the CS high by default
  digitalWrite(CS2, HIGH);  //Set the CS high by default
  digitalWrite(CS3, HIGH);  //Set the CS high by default
  digitalWrite(WR, HIGH);  //Set the WR pin high by default

  //Clock in Gain and Buffer Values
  digitalWrite(CS1, LOW);
```

```
  digitalWrite(CS2, LOW);
  digitalWrite(CS3, LOW);
  delayMicroseconds(10);
  digitalWrite(WR, LOW);
  delayMicroseconds(10);

  digitalWrite(CS1, LOW);
  digitalWrite(CS2, LOW);
  digitalWrite(CS3, LOW);
  delayMicroseconds(10);
  digitalWrite(WR, LOW);
  delayMicroseconds(10);
  //Serial.begin(9600); //Remove comments to enable serial communication
  //Serial.println();
}
void loop()
{
  AnalogInX = (analogRead(xpin));
  delay(1);
  AnalogInY = (analogRead(ypin));
  delay(1);
  AnalogInZ = (analogRead(zpin));
```

```
AnalogInX = ((AnalogInX-510)*8) +510;
AnalogInY = ((AnalogInY-515)*10) +515;
AnalogInZ = ((AnalogInZ-600)*8) +600;

if(AnalogInX <0) { AnalogInX = 0; } //Set bounds for values
if(AnalogInY <0) { AnalogInY = 0; }
if(AnalogInZ <0) { AnalogInZ = 0; }
if(AnalogInX >1024) { AnalogInX = 1024; }
if(AnalogInY >1024) { AnalogInY = 1024; }
if(AnalogInZ >1024) { AnalogInZ = 1024; }

XBoost = map(AnalogInX, 0, 1023, 0, 255);
YBoost = map(AnalogInY, 0, 1023, 0, 255);
ZBoost = map(AnalogInZ, 0, 1023, 0, 255);

digitalWrite(CS1, LOW);              //Ready the 1st AD5330 for input
digitalWrite(WR, LOW);    //Enable Writes on the AD5330
PORTD = (unsigned char)XBoost; //Set the voltage on the AD5330
digitalWrite(WR, HIGH);   //Clock in the new data
digitalWrite(CS1, HIGH);
```

```
digitalWrite(CS2, LOW);             //Ready the 2nd AD5330 for input
digitalWrite(WR, LOW);    //Enable Writes on the AD5330
PORTD = (unsigned char)YBoost;        //Set the voltage on the AD5330
digitalWrite(WR, HIGH);   //Clock in the new data
digitalWrite(CS2, HIGH);

digitalWrite(CS3, LOW);             //Ready the 3rd AD5330 for input
digitalWrite(WR, LOW);    //Enable Writes on the AD5330
PORTD = (unsigned char)ZBoost;        //Set the voltage on the AD5330
digitalWrite(WR, HIGH);   //Clock in the new data
digitalWrite(CS3, HIGH);
delay(6);

/*digitalWrite(CS1, LOW);           //Ready the 1st AD5330 for input
digitalWrite(WR, LOW);    //Enable Writes on the AD5330
PORTD = 78; //Set the voltage on the AD5330
digitalWrite(WR, HIGH);   //Clock in the new data
digitalWrite(CS1, HIGH);

digitalWrite(CS2, LOW);             //Ready the 2nd AD5330 for input
digitalWrite(WR, LOW);    //Enable Writes on the AD5330
PORTD = 120;       //Set the voltage on the AD5330
```

```
digitalWrite(WR, HIGH);   //Clock in the new data
digitalWrite(CS2, HIGH);


digitalWrite(CS3, LOW);             //Ready the 3rd AD5330 for input
digitalWrite(WR, LOW);    //Enable Writes on the AD5330
PORTD = 140;       //Set the voltage on the AD5330
digitalWrite(WR, HIGH);   //Clock in the new data
digitalWrite(CS3, HIGH);*/
delay(100);


/*Serial.print(AnalogInX); //Enable serial communication
Serial.print("\t");
Serial.print(AnalogInY);
Serial.print("\t");
Serial.print(AnalogInZ);
Serial.println();


Serial.print("\t");
Serial.print(XBoost);
Serial.print("\t");
Serial.print(YBoost);
Serial.print("\t");
```

```
  Serial.print(ZBoost);
  Serial.println(); */
}

int main(void)
{
        init();

        setup();

        for (;;)
        loop();

        return 0;
}
```