Wii Remote Accuracy

By: Isaac Muttschall

Class: ECE 4220

School: University Of Missouri, Columbia

Date: 12/17/2009

Abstract:

The Wii Remote is a revolutionary Human Interface Device (HID) which uses embedded systems with limitless possibilities when put together. It is desirable to know the ability of the Wii Remote to accurately measure its pitch and roll through software. This paper presents the inner-workings of a Wii Remote, some methods of interfacing between the Wii Remote and a personal computer, and the accuracy of the Wii Remote's ability to correctly measure its pitch and roll. This report finds that the Wii Remote is able to correctly measure its pitch within 1.275 (degrees) and its roll within 0.701 (degrees).

Introduction:

The Wii remote contains many different systems, however, the ones of interest to this paper are the ADXL330 accelerometer chip, the BroadCom BCM 2042 Bluetooth chip, and the 128x96 IR camera. These systems are user together to provide the three-dimensional position of the WiiMote in relation the to the Wii system.

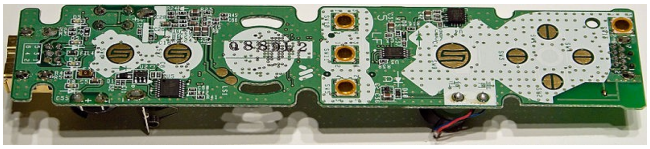

Illustration 1: (Top of WiiMote Circuit Board)

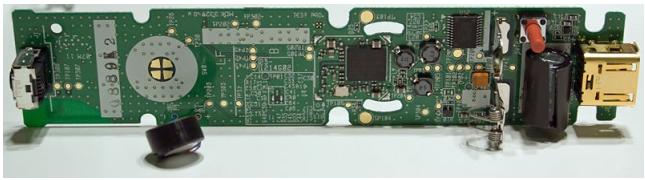

Illustration 2: (Bottom of WiiMote Circuit Board)

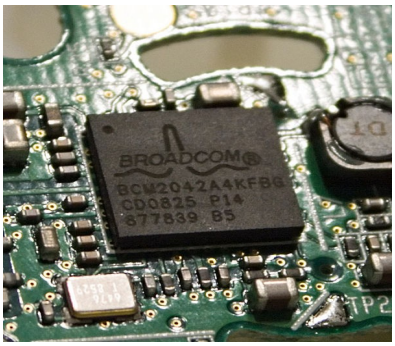

Illustration 3: (BroadCom 2042 Bluetooth Chip)

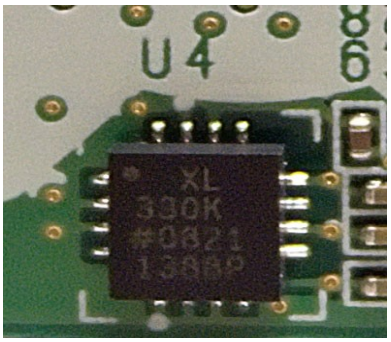

Illustration 4: (ADXL 330 Accelerometer Chip)

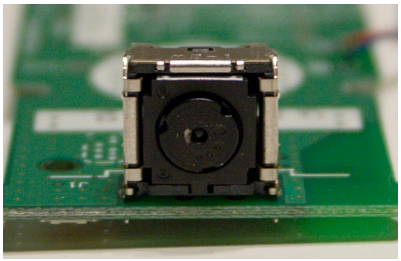

Illustration 5: (IR Camera)

The BroadCom Bluetooth chip provides communication to the Wii System via a wireless connection. This is done though sending packets of information back and forth, this is discussed in more detail in the next section. According to the data sheet for the ADXL330 “The accelerometer can measure the static acceleration of gravity in tilt sensing applications as well as dynamic acceleration resulting from motion, shock, or vibration” [1]. This data is used to provide the tilt and roll of the WiiMote. The infrared camera on the WiiMote has the ability to track up to 4 infrared moving points [2]. The sensor bar for the Wii is equipped with several infrared Light Emitting Didoes (LEDs) which are used by the IR camera in the WiiMote to define its position in relation the sensor bar. When the sensor bar is placed above or below the TV, it gives the position of the WiiMote in relation to the TV. Using the sensor bar, the Wii is able to determine the Z-axis rotation of the WiiMote through the IR camera.

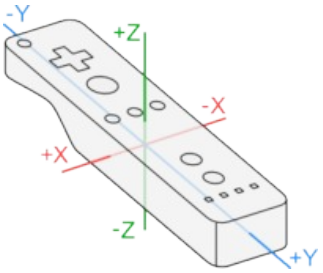

Illustration 6: (Coordinate System Used By WiiMote)


Illustration 7: (Wii Sensor Bar)

When these three systems are combined, they give the full three-dimensional position of the WiiMote in relation to the sensor bar and the ability to transmit that data to the Wii system. The details of transmitting data to and from the WiiMote are examined next.

Wii Remote interface:

The WiiMote interfaces with the Wii system through sending packets of data over a Bluetooth connection. The Human Interface Device (HID) standard block is used so that the WiiMotes can be self-describing. This block contains reports or commands that the device can interpret or understand. The WiiMote, like other HID devices, reports its HID block when queried using the SDP protocol. The HID block for the WiiMote is shown below. An output labeled with 'O', is data sent from the Wii system to the WiiMote. And input labeled with 'I', is data sent from the WiiMote to the Wii system. Through these commands and reports, the systems on the WiiMote can be controlled from the Wii system. The packets have the form “command/report_options_data”. The data of particular interest to this paper are the Accelerometer and the IR camera. This data is gathered from the Data Reporting function. The Data Reporting is started by sending a 6 byte command of the form “command_options_mode” each item is 2 bytes [2].

I/O	ID(s)	Size	Function
O	0x10	1	Unknown
O	0x11	1	Player LEDs
O	0x12	2	Data Reporting mode
O	0x13	1	IR Camera Enable
O	0x14	1	Speaker Enable
O	0x15	1	Status Information Request
O	0x16	21	Write Memory and Registers
O	0x17	6	Read Memory and Registers
O	0x18	21	Speaker Data
O	0x19	1	Speaker Mute
O	0x1a	1	IR Camera Enable 2
I	0x20	6	Status Information
I	0x21	21	Read Memory and Registers Data
I	0x22	4	Acknowledge output report, return function result
I	0x30-0x3f	2-21	Data reports

Table 1: (HID Descriptor Block For A WiiMote)

There are many different ways to collect data from the accelerometer and the IR camera depending on what the controller setup is. These different modes can be accessed by changing the mode bytes. There are many ways to access the data but this command will give both accelerometer data and IR data. 12 00 33 [2].

0x33 returns button, accelerometer, and IR data in the form. (2 bytes of button data) (3 bytes of Accelerometer data) (12 bytes of IR data).

According to HomeBrew.org's Wiki, Accelerometer data has the form XX YY ZZ where XX, YY, and ZZ are unsigned bytes representing the acceleration in each of the three axis, where zero acceleration is approximately 0x80 [2]. Homebrew.org also explains in detail the different reporting formats for the IR camera. For our knowledge, the data contains many different positional data bytes.

WiiMote Accuracy:

The WiiMote's X and Y position can be gathered from the data given from the reports mentioned above. The accuracy of these angles in degrees is to be determined. Data is collected by using a protractor to manually measure the angle, and hooking the WiiMote up to GlovePIE [3] to determine the reported pitch (X) and roll (Y) angle. To hole the WiiMote steady at a constant angle, a rotating clamp is used. The angle of the WiiMote can be easily measured from from this clamp, so semi-accurate results can be attained. The manually measured angle is compared against the software obtained angle to find an error between the actual angle and the reported angle.

Illustration 8: WiiMote Measuring Roll

Illustration 9: WiiMote Measuring Roll

Illustration 10: WiiMote Measuring Pitch

Illustration 11: WiiMote Measuring Pitch

The results of the measurement are shown below in Table 1 and Table 2. The mean error for the pitch was found to be 1.275 (degrees) with a standard deviation of 0.677 (degrees). The mean error for the roll was found to be 0.701 (Degrees) with a standard deviation of 0.525 (degrees).

PITCH		
Actual (Degrees)	Program (Degrees)	Error (Degrees)
0.00	-1.12	1.12
15.00	16.91	1.91
30.00	31.94	1.94
45.00	45.50	0.50
60.00	61.85	1.85
75.00	74.54	0.46
90.00	88.34	1.66
105.00	102.25	2.75
120.00	119.25	0.75
135.00	134.02	0.98
150.00	148.93	1.07
165.00	164.59	0.41
180.00	178.83	1.17
XBAR	-	1.275
SIGMA	-	0.677
MIN	-	0.410
MED	-	1.120
MAX	-	2.750

Table 2: Pitch Measurement Results

ROLL		
Actual (Degrees)	Program (Degrees)	Error (Degrees)
0.00	-1.17	1.17
15.00	14.86	0.14
30.00	29.91	0.09
45.00	44.15	0.85
60.00	59.86	0.14
75.00	74.20	0.80
90.00	88.96	1.04
105.00	104.34	0.66
120.00	120.14	0.14
135.00	136.03	1.03
150.00	151.06	1.06
165.00	165.14	0.14
180.00	181.86	1.86
XBAR	-	0.701
SIGMA	-	0.525
MIN	-	0.090
MED	-	0.800
MAX	-	1.860

Table 3: Roll Measurement Results

Conclusions:

According the the methods of testing the WiiMote is able to correctly measure its pitch to within an error of 1.275 (degrees) on average and 2.750 (degrees) at most. The WiiMote is also able to correctly measure its roll to within an error of 0.701 (degrees) on average and 1.860 (degrees) at most. Possible errors in the experiment could be from human error, using an inadequate instrument to measure the actual angle, and possible conversion errors in the code causing a loss of data. To make this experiment more accurate, a digital laser angle measurement device should be used. This report has explained the basic details on the operation of the WiiMote, some methods for interfacing with the WiiMote, and determined the accuracy of the WiiMotes ability to correctly measure its pitch and roll.

Works Cited:

Text:

[1] Analog Devices, “Small, Low Power, 3-Axis ±3 *g i* MEMS® Accelerometer ,” ADXL330 datasheet, 2006 Rev. 0. [Online]. Available: http://www.sparkfun.com/datasheets/Components/ADXL330_0.pdf [Accessed: Dec. 16, 2009].

[2] "WiiMote – WiiBrew," Dec. 1, 2009. [Online]. Available: http://wiibrew.org/wiki/Wiimote. [Accessed: Dec. 16, 2009].

[3] [Online]. Available: http://carl.kenner.googlepages.com/glovepie. [Accessed: Dec. 16, 2009].

Illustrations:

[1-6] http://wiibrew.org/wiki/Wiimote

[7] http://commons.wikimedia.org/wiki/File:Nintendo_Wii_Sensor_Bar.jpg

Tables:

[1] http://wiibrew.org/wiki/Wiimote

Appendix A: Code

Modified GlovePIE testWiiMote.PIE code

```
// This script is just to show you how WiiMote scripts look
// See the WiimoteScripts folder for real scripts

//Modified by Isaac Muttschall
//University of Missouri, Columbia
//Electrical and Computer Engineering Student
//For Class: ECE 4220 "Realtime Embedded Systems II"
//Final Project/Paper
//Date: 12/15/2009

// Set keys to wiimote/nunchuk buttons
//for my purposes these are not needed
//*************************************************************
//Up = Wiimote.Up or Wiimote.Classic.Up
//Down = Wiimote.Down or Wiimote.Classic.Down
//Left = Wiimote.Left or Wiimote.Classic.Left
//Right = Wiimote.Right or Wiimote.Classic.Right
//A = Wiimote.A or Wiimote.Classic.a
//B = Wiimote.B or Wiimote.Classic.b
//C = Wiimote.Nunchuk.CButton
//One = Wiimote.One
//Two = Wiimote.Two
//Home = Wiimote.Home or Wiimote.Classic.Home
//Minus = Wiimote.Minus or Wiimote.Classic.Minus
//NumPadPlus = Wiimote.Plus or Wiimote.Classic.Plus
//X = Wiimote.Classic.x
//Y = Wiimote.Classic.y
//Z = Wiimote.Nunchuk.ZButton or Wiimote.Classic.ZL or Wiimote.Classic.ZR
//L = Wiimote.Classic.L
//R = Wiimote.Classic.R

// Set PPJoy Virtual Joystick to nunchuk joystick
//PPJoy.Analog0 = Wiimote.Nunchuk.JoyX
//PPJoy.Analog1 = Wiimote.Nunchuk.JoyY

// Rumble when shift is pressed
//Wiimote.Rumble = Shift

//*************************************************************

//LEDs will be used for testing different modes
//LED1 for X axis mode
//LED2 for y axis mode
//LED3 for z axis mode
//LED4 for all mode
```

```
//Set to x axis mode initially
// Set the middle two LEDs to ON
//Wiimote.Led1 = true
//Wiimote.Led2 = false
//Wiimote.Led3 = false
//Wiimote.Led4 = false

//Not needed for my purposes
//*****************************************************************
// Show the IR dots as fake cursors
//Cursor1.Visible = Wiimote.dot1vis
//Cursor1.x = Wiimote.dot1x / 1023
//Cursor1.y = Wiimote.dot1y / 1023

//Cursor2.Visible = Wiimote.dot2vis
//Cursor2.x = Wiimote.dot2x / 1023
//Cursor2.y = Wiimote.dot2y / 1023

//Cursor3.Visible = Wiimote.dot3vis
//Cursor3.x = Wiimote.dot3x / 1023
//Cursor3.y = Wiimote.dot3y / 1023

//Cursor4.Visible = Wiimote.dot4vis
//Cursor4.x = Wiimote.dot4x / 1023
//Cursor4.y = Wiimote.dot4y / 1023

// Check expansion
//if Wiimote.HasNunchuk then
//  var.expansion = Wiimote.Nunchuk.JoyX+','+Wiimote.Nunchuk.JoyY+'
NC.Force='+Wiimote.Nunchuk.gx+','+Wiimote.Nunchuk.gy+','+Wiimote.Nunchuk.gz
//else if Wiimote.HasClassic then
//  var.expansion = 'CL: '+Wiimote.Classic.RawJoy1X+','+Wiimote.Classic.RawJoy1Y
//else
//  var.expansion = ""
//end if
//*****************************************************************

//if A is pressed goto X reporting mode
if Wiimote.A then
  Wiimote.Led1 = true
  Wiimote.Led2 = false
  Wiimote.Led3 = false
  Wiimote.Led4 = false

else if Wiimote.B then
  Wiimote.Led1 = false
  Wiimote.Led2 = true
  Wiimote.Led3 = false
  Wiimote.Led4 = false
```

```
else if Wiimote.One then
  Wiimote.Led1 = false
  Wiimote.Led2 = false
  Wiimote.Led3 = true
  Wiimote.Led4 = false

else if Wiimote.Two then
  Wiimote.Led1 = false
  Wiimote.Led2 = false
  Wiimote.Led3 = false
  Wiimote.Led4 = true
end if

if Wiimote.Led1 = true then
  debug = 'Bat='+Wiimote.Battery+'; Pitch= '+Wiimote.Pitch+'; Roll= '+Wiimote.Roll
      wait 200ms
else if Wiimote.Led2 = true then
  debug = 'Bat='+Wiimote.Battery+'; RelAcc= '+9*RemoveUnits(Wiimote.RelAcc)
      wait 200ms
else if Wiimote.Led3 = true then
  debug = 'Bat='+Wiimote.Battery+'; RawAcc= '+9*RemoveUnits(Wiimote.RawAcc)
      wait 200ms
else if Wiimote.Led4 = true then
  debug = 'Bat='+Wiimote.Battery+'; RawForce= '+9*RemoveUnits(Wiimote.RawForce)
      wait 200ms
end if
```

Link to Lunder Lander Code By: Wiim

http://digitalretrograde.com/projects/wiim/