Wiimote Hacking

Wiiing up your Wiisome Wiimote with some Wiitastic Wii Scripts.

Wii.

Anatomy of a Wiimote

- Communication:
  - Broadcom 2042 Bluetooth controller chip
  - Acts as a Bluetooth HID device
- Inputs
  - Buttons
  - Motion Sensor
  - Infrared (IR) Sensor
- Outputs
  - Player LEDs
  - Rumble
  - Speaker
- Memory
  - Flash Memory
  - Control Registers
- Expansion Port
- Batteries

Communication

Wiimote Bluetooth HID

- Bluetooth HID is directly based on the USB HID
- Queried with Bluetooth SDP, returns:
  - Name: Nintendo RVL-CNT-01
  - Vendor ID: 0x057e
  - Product ID: 0x0306
- Max Report frequency of 100/sec
- Does not use any auth/security features of Bluetooth HID standard

Wiimote Peering

- Press and hold the “1” and “2” buttons simultaneously or press the red “sync” button under the battery cover
- Query Wiimote via Bluetooth HID driver on the host device

Wiimote HID Interface

- HID standard allows devices to be self-describing using a HID descriptor block
- The HID descriptor block includes an enumeration of available Reports
- Reports are like network ports assigned to a particular service
- Reports are unidirectional
- Query using SDP to get descriptor block including reports, direction, and payload size

Wiimote HID Reports

- Output:
  - 0x11 (1): Player LEDs, Force Feedback
  - 0x12 (2): Report type / ID
  - 0x13 (1): Enable IR Sensor
  - 0x14 (1): Enable Speaker
  - 0x15 (1): Controller Status
  - 0x16 (21): Write data
  - 0x17 (6): Read data
  - 0x18 (21): Speaker data
  - 0x19 (1): Mute speaker
  - 0x1a (1): IR Sensor Enable 2
- Input:
  - 0x20 (6): Expansion Port
  - 0x21 (21): Read data
  - 0x22 (4): Write data
  - 0x30 (2): Buttons only
  - 0x31 (5), 0x33 (17):
    - Buttons | Motion Sensing Report
  - 0x32 (16), 0x34 (21), 0x36 (21), 0x3d (21):
    - Buttons | Expansion Port | IR
  - 0x35 (21), 0x37 (21):
    - Buttons | Motion Sensing Report | Expansion Port
  - 0x38 (21), 0x3e (21), 0x3f (21):
    - Buttons | Motion Sensing Report | IR

Inputs

Wiimote Buttons

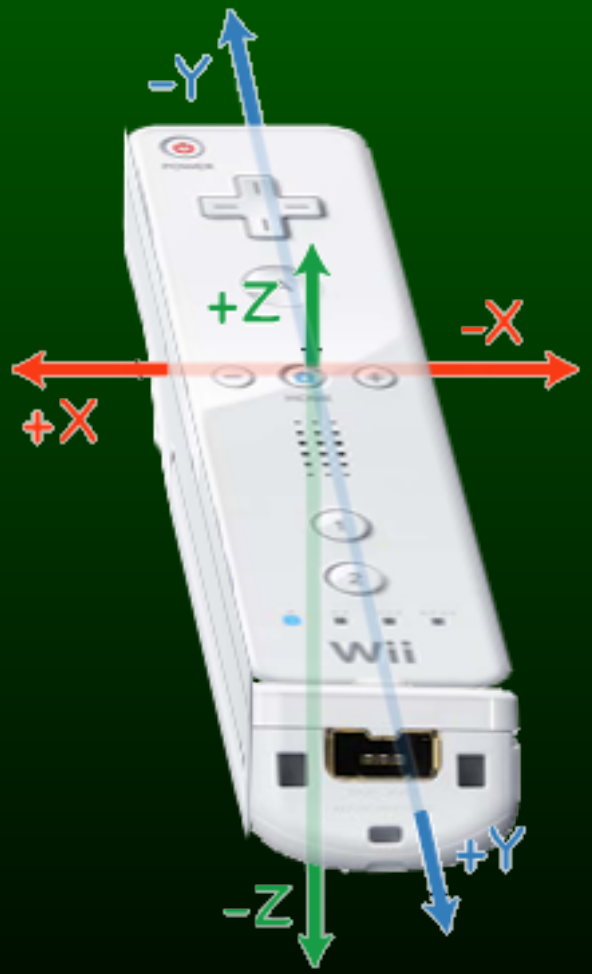
- 12 buttons on the Wiimote:
  - Power button
  - D-Pad: Up, Down, Left, Right
  - A and B buttons
  - -, Home, and + buttons
  - 1 and 2 buttons
- Button press or release generates Input Report 0x30 containing a 2-byte bitmask with current state of all buttons
- Button state mask is also included as first two bytes of all other Input Reports

Button Bitmask Values

- 0x0001: 2 button
- 0x0002: 1 button
- 0x0004: B button
- 0x0008: A button
- 0x0010: - button
- 0x0020: Used with 0x3e or 0x3f
- 0x0040: Used with 0x3e or 0x3f
- 0x0080: Home button
- 0x0100: Left button
- 0x0200: Right button
- 0x0400: Down button
- 0x0800: Up button
- 0x1000: Plus button
- 0x2000: Used with 0x3e or 0x3f
- 0x4000: Used with 0x3e or 0x3f
- 0x8000: Unknown

Motion Sensor

- Motion is sensed by an Analog Devices ADXL330 3-axis linear accelerometer located slightly left of the A button
- Measures acceleration over a range of +/- 3g with 10% sensitivity
- Forces on each axis are digitized into 8-bit unsigned integers, with the zero scale set to 0x80



Motion Sensor Reports

- Wiimote does not normally report motion sensor data, but can be requested by sending a SET_REPORT request to channel 0x12:
  - (a1) 12 00 31
- Wiimote will now send motion sensor reports to channel 0x31 at a freqency determined by parameters to the above SET_REPORT request:
  - (a1) 31 40 20 86 8a a5
  - Bytes 1 and 2 are the buttons bitmask
  - Bytes 3, 4, and 5 are X, Y, and Z axis measurements, respectively
- Other channels that can be used for motion sensor reports include 0x33, 0x35, 0x37, 0x3e and 0x3f
- 0x3e and 0x3f have special properties and extra bytes which may contain IR data
- Reports can be stopped setting the channel to 0x30

Motion Sensor Calibration

- Zero points and gravity values for the three axes are stored in the Wiimote's flash memory, starting at address 0x16
- Is repeated at address 0x20:
  - 0x16: zero point for X axis
  - 0x17: zero point for Y axis
  - 0x18: zero point for Z axis
  - 0x19: unknown
  - 0x1a: +1G point for X axis
  - 0x1b: +1G point for Y axis
  - 0x1c: +1G point for Z axis
  - 0x1d: unknown
  - 0x1e - 0x1f: possible checksum

IR Sensor

- PixArt IR sensor located at the front of the Wiimote housing
- Possibly a PixArt “System on a Chip” product
- Locates 2 IR beacons within the IR sensor’s field of view
- Can detect and transmit up to 4 IR hotspots back to the host
- Various data sets can be requested, including:
  - Only position values
  - Position and size
  - Position, size, and pixel values

IR Sensor Reports

- IR Sensor reports 3 bytes of data per dot recognized
- Bytes 0 and 1 are X and Y positions
- Byte 2 is the MSBs of X and Y and a size value:
  - xxxxxxx yyyyyyy yyxxssss

Outputs

Player LEDs

- 4 blue LEDs
- Used during play to indicate the player number of the controller
- All 4 blink when in Bluetooth discovery mode
- Independently controllable via Output Report to channel 0x11 containing LED bitmask:
  - (52) 11 10
- Most-significant 4 bits control each LED
- Updating the LED bitmask rapidly (>1 times per second) causes all 4 LEDs to blink as if in discovery mode, then returns to last known bitmask

Force Feedback (Rumble)

- Rumble is achieved via a motor with an unbalanced weight
- Can be activated by sending an Output Report to channels 0x11, 0x13, 0x14, 0x15, 0x19 or 0x1a with the LSB set:
  - (52) 13 01
- Can be disabled by clearing the LSB:
  - (52) 13 00

Speaker

- Small, low-quality internal speaker
- Used primarily for short sound effects during gameplay
- Sound is streamed directly from the host
- Speaker has adjustable parameters
- Controlled via 3 Output Reports together with a section of the register address space
- Report 0x14 is used to enable or disable the speaker by setting or clearing bit 2 of the payload:
  - Enable: (52) 14 04
  - Disable: (52) 14 00
- Report 0x19 is used to mute or un-mute the speaker, used exactly like 0x14

Speaker Initialization

- To initialize the speaker:
  - Enable speaker: Output Report 0x14 of value 0x04
  - Mute speaker: Output Report 0x19 of value 0x04
  - Write 0x01 to register 0x04a20009
  - Write 0x08 to register 0x04a20001
  - Write 7-byte configuration to registers 0x04a20001 - 0x04a20008
  - Write 0x01 to register 04a20008
  - Unmute speaker: Output Report 0x19 value 0x00

Speaker Configuration

- 7 bytes control all speaker settings:
  - 0x00: unknown
  - 0x01: unknown
  - 0x02: unknown
  - 0x03: Sample rate divisor, based on a start rate of approximately 48000Hz
  - 0x04: Volume control
  - 0x05: unknown
  - 0x06: unknown

Sample Rates and Volume

- Known sample rate values:
  - 0x0b: 4000/4364Hz (~4200Hz)
  - 0x0c: 3692/4000Hz (~3920Hz)
  - 0x0d: 3429/3692Hz (~3640Hz)
  - 0x0e: 3200/3429Hz (~3360Hz)
  - 0x0f: 3000/3200Hz (~3080Hz)
- Volume:
  - Any value from 0x00 to 0xff works
  - 0x40 seems to be generally accepted good default

Speaker Data

- Report 0x18 is used to send speaker data
- Up to 20 bytes may be sent at once
- Byte 1 of the payload indicates the length of data, shifted left by 3 bits
- Data must be padded if it is less than the indicated length
- Sound data must be sampled at roughly the proper rate
- Rate can be set during speaker initialization
- Format appears to be 4-bit ADPCM sound

Memory

Flash Memory

- Persistent RAM
- 5.5K of RAM
- Memory Addresses 0x0000 - 0x15ff
- Control Registers begin with 0x04 and are 4 byte addresses (0x04a10000)
- Addresses wrap after 0xffff

Flash Memory Addresses

- 2-byte Addresses (0x010000 == 0x0000)
- 0x16 and 0x20: Calibrated zero offsets for accelerometer
- 0x0040 - 0x0fc9: All zeroes on new Wiimote
- 0x0fca - 0x12b9: Mii data block 1
- 0x12ba - 0x15a9: Mii data block 2
- 0x15aa - 0x15ff: All zeroes on new Wiimote
- 0x1600 - 0xffff: Don’t exist, return error on read
- 0x010000 - 0xFF0000 used for control registers

Control Registers

- Bit 2 must be set in first byte of address
- Bit 1 is the rumble flag and not considered part of the address (0x05a20000 == 0x04a20000)
- Only 0x04a20000 - 0x04a30000 are readable

Control Register Addresses

- 0x04000000 - 0x049fffff: returns error 7 on read
- 0x04a00000 - 0x04a1ffff: doesn't exist
- 0x04a20000 - 0x04a30000: speaker
- 0x04b00000 - 0x04bfffff: returns error 7 on read
  - 0x04b00000 - 0x04b00008: IR sensitivity settings
  - 0x04b0001a - 0x04b0001b: IR sensitivity settings
  - 0x04b00030: IR toggle
  - 0x04b00033: IR mode
- 0x04c00000 - 0x04ffffff: returns error 7 on read

Reading Memory

- Output Report 0x17 reads memory:
  - (52) 17 XX XX XX XX YY YY
  - XX XX XX XX is big-endian formatted address
  - YY YY is big-endian formatted size in bytes
  - LSB of first byte is rumble flag and is not part of address, should be set to whatever state the rumble should be
- Responses look like:
  - `SE XX XX data...`
  - `(a1) 21 80 00 f0 11 f0 80 6c 8c ...`
  - S shifted right 4 bits is size in bytes, minus 1, of current packet
  - E is error flag:
    - 8 if reading from bytes that don't exist
    - 7 if reading from write-only registers
    - 0 if no error
  - XX XX is offset of current packet in big-endian format
  - Rest is data, 16 bytes maximum

Writing Memory

- Output Report 0x16 writes memory:
  - `XX XX XX XX SS data...`
  - `(52) 16 00 00 00 00 10 57 69 69 ...`
  - XX XX XX XX is the address being written to
  - SS is the size in bytes
  - Up to 16 bytes of data, padded
- Write acknowledgement is sent on Input Report 0x22
- 0x04 as first byte of payload indicates write to control register address
- Bit 0 of first byte of payload sets rumble feature
- Second byte of payload is ignored unless writing to control register

Expansion Port

Expansion Port

- Located on the bottom of the Wiimote
- Used to connect auxiliary controllers which augment the input options of the Wiimote
- Custom connector with 6 contacts
  - 2 of the contacts are longer and make contact first when the plug is inserted
- Communicates with the Wiimote via a 400kHz “fast” I2C, slave address 0x52
- Available expansions include:
  - Nunchuk controller
  - Classic controller

Expansion Port Reports

- 0x20: Expansion port status
  - Sent whenever status changes
  - Can be requested with Output Report 0x15
  - `bu bu ss uu uu bl`
  - `(a1) 20 00 00 02 00 00 c0`
  - bu contains the button state bitmask
  - ss contains the Expansion Port status bitmask
    - Bit 0 is unknown
    - Bit 1 indicates whether or not an attachment is plugged in
    - Bit 2 indicates whether the speaker is enabled
    - Bit 3 indicates whether the IR sensor is enabled
    - Bits 4 - 8 indicate the status of the 4 LEDs
  - uu contains some unknown bytes
  - bl contains the battery level

Batteries

Batteries

- 2 AA size batteries
- Expansion kits containing rechargeable Lithium-Ion batteries and a charging station have hit the market

Battery Reports

- 0x20: Read battery charge level
  - Sent when something is plugged into or unplugged from the expansion port
  - Can be requested with Output Report 0x15
  - `bu bu ss uu uu bl`
  - `(a1) 20 00 00 02 00 00 c0`
  - bl contains the battery level
  - Values as high as 0xc6 have been found
  - Suggests that a “fully charged” value may be 0xc8 (200 in decimal)

Expansion Devices

Expansion Devices

- Use address space 0x04a40000 - 0x04a400ff
- Must be initialized by writing value 0x00 to address 0x04a40040
- Byte 3 of address space appears to be ignored (0x04a4ff00 == 0x04a40000)
- Data is “encrypted” via simple XOR
  - Decrypt: value = (byte ^ 0x17) + 0x17

Expansion Device Addresses

- 0x04a40008 - 0x04a4000d:
  - 6-byte current state of device
- 0x04a40020 - 0x04a4002f:
  - Calibration data
- 0x04a40030 - 0x04a4003f:
  - Repeat of data at 0x04a40020
- 0x04a400f0 - 0x04a400ff:
  - Same on all similar devices, possible device Type ID

Expansion Device Reports

- Device must be initialized first
- Reports 0x32, 0x34, 0x35, 0x36, 0x37 and 0x3d will contain the 6-byte device status
- Data must be “decrypted”
- Can also retrieve these bytes by reading 16 bytes starting at address 0x04a40000
  - Device status will be at offset 0x08-0x0d
  - Different data is returned if you try to read directly from 0x04a40008, so don’t try.

Nunchuk Controller State

- 6-byte current state of device:
  - 0x00: X-axis value of analog stick
  - 0x01: Y-axis value of analog stick
  - 0x02: Accelerometer X-axis acceleration value
  - 0x03: Accelerometer Y-axis acceleration value
  - 0x04: Accelerometer Z-axis acceleration value
  - 0x05: Button state bitmask:
    - Bit 0: Z button
    - Bit 1: C button
    - Bits 2-3: LSBs from X-axis accelerometer
    - Bits 4-5: LSBs from Y-axis accelerometer
    - Bits 6-7: LSBs from Z-axis accelerometer

Classic Controller State

- 6-byte current state of device:
  - 0x00: X-axis value of both analog sticks:
    - Bits 0-5: X-axis of left analog stick
    - Bits 6-7: Bits 3-4 of X-axis of right analog stick
  - 0x01: Y-axis value of left analog stick:
    - Bits 0-5: Y-axis of left analog stick
    - Bits 6-7: Bits 1-2 of X-axis of right analog stick
  - 0x02: Y-axis value of right analog stick / Left shoulder button
    - Bits 0-4: Y-axis of right analog stick
    - Bits 5-6: Bits 3-4 of left shoulder button
    - Bit 7: Bit 0 of X-axis of right analog stick
  - 0x03: Left / Right shoulder buttons
    - Bits 0-4: Right shoulder button
    - Bits 5-7: Bits 0-2 of left shoulder button

Classic Controller State

- 6-byte current state of device (continued)
  - 0x04: Button state bitmask 1:
    - Bit 0: unused
    - Bit 1: R button fully pressed
    - Bit 2: + button
    - Bit 3: Home button
    - Bit 4: - button
    - Bit 5: L button fully pressed
    - Bit 6: Down button
    - Bit 7: Right button
  - 0x05: Button state bitmask 2:
    - Bit 0: Up button
    - Bit 1: Left button
    - Bit 2: ZR button
    - Bit 3: x button
    - Bit 4: a button
    - Bit 5: y button
    - Bit 6: b button
    - Bit 7: ZL button

GlovePIE

Or, why you didn't need to know any of that lower-level stuff...

What is GlovePIE?

- Glove Programmable Input Emulator
- Originally designed for VR gloves
- Emulates joystick and keyboard input when input is received from other devices
- Supports the Wiimote as an input device!
- Can use it to map Wiimote inputs to a game's standard controls
  - Now you can play WoW with your Wiimote!!!
- Abstracts away all that lower-layer stuff into a nice object-oriented scripting language
- Does not work with Microsoft's Bluetooth stack

GlovePIE Scripts

- GlovePIE creates objects for the hardware it supports
- Available Wiimote objects of course have methods which implement much of that lower-layer bit and byte necromancy for you
- Supports variables, flow control, conditionals, etc.
- Best shown by example...

Example #1: NES Emulator

```
A = Wiimote.Two          //A button "Two" Button
B = Wiimote.One          //B button "One" Button
S = Wiimote.Plus         //Start "Plus" Button
F = Wiimote.Minus        //Select "Minus" Button
Left = Wiimote.Up        //Up is "D pad Left"
Right = Wiimote.Down     //Down is "D pad Right"
Down = Wiimote.Left      //Left is "D pad Down"
Up = Wiimote.Right       //Right is "D pad Up"
D = Wiimote.Home         //Left Shoulder is "Home"
N = Wiimote.B            //Right Shoulder is "B"
```

Example #2: WoW

```
// Setting up the wiimote so that the controls on top of the wiimote works
// as WASD that is commonly used in FPS and other games.
w = Wiimote.Up
s = Wiimote.Down
a = Wiimote.Left
d = Wiimote.Right

// Here I set the nunchuk so it corresponts with my direction keys.
Right = 1 > Wiimote1.Nunchuk.JoyX > 0.5
Left = -1 < Wiimote1.Nunchuk.JoyX < -0.5
down = 1 > Wiimote1.Nunchuk.JoyY > 0.5
up = -1 < Wiimote1.Nunchuk.JoyY < -0.5

// Bind some keys to the mote, you can bind your own.
h = Wiimote.Plus
q = Wiimote.Minus
tab = Wiimote.Home
j = Wiimote.One
k = Wiimote.Two
// Nunchuck
u = Wiimote.Nunchuk.CButton
f = Wiimote.Nunchuk.ZButton
// B for left click and A for right click
mouse.LeftButton = Wiimote.B
mouse.RightButton = Wiimote.A
```

Objects

- Keyboard (also, Key)
- Wiimote

Keyboard Methods

- Most “keys” will work:
  - Up, Down, Left, Right, w, a, s, d, Enter, Space, etc.

Wiimote Methods

- Buttons:
  - Up, Down, Left, Right, A, B, Plus, Minus, Home, One, Two
- LEDs:
  - Toggles:
    - LED1, LED2, LED3, LED4
  - Bitmask: LEDs
- Rumble:
  - Rumble (Boolean, set to 0 or 1)

Demos!

PowerPoint

No demo necessary, I've been using it this entire time...

Knight Rider

Everybody loves blinky LEDs

Wiibrator 3

This one’s for the ladies...

References

- Wii Linux Project Wiimote Page
  - http://www.wiili.org/index.php/Wiimote
- GlovePIE
  - http://carl.kenner.googlepages.com/glovepie