Wii Remote Controller

Group 4 - TRON
Guillaume Antony, Maggie Hayes,
Kishore Rao, Tyler Siegel

Outline

1. WiiMote Overview
2. Configuration and Connection
3. WiiMote communication
4. Memory and Control Registers

Wii (Re)mote Controller

- Nintendo Wii & Wii U controller
- Released in 2006
- Motion Plus : 2009

- One handed control
- Autonomy : 30-60h
- Extensions : Nunchuk

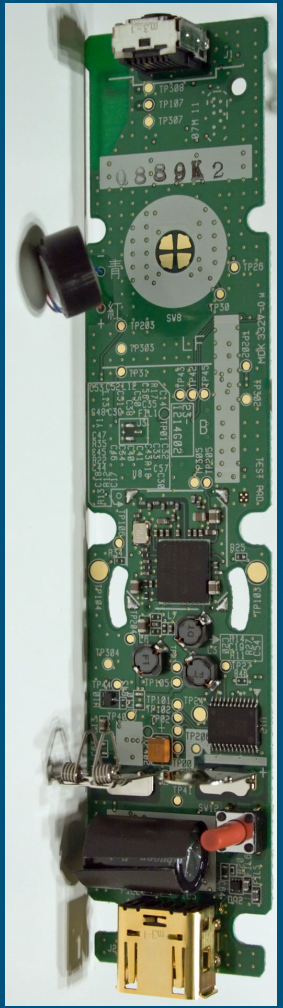
Source : https://www.amazon.com/Wii-Remote-Plus-White-Nintendo/dp/B0045FGET2

What’s inside ?

- Digital buttons and Dpad
- Speaker
- EEPROM 16kB
- Accelerometer
- Infrared sensors
- Gyroscope

Source : https://fr.wikipedia.org/wiki/Gyroscope

- Bluetooth chip : Broadcomm BCM2042
- Extension port (I2C)



Source : http://wiibrew.org/wiki/Wiimote

Typical Use

- Game controller
  - Basics buttons
  - Movements/Gestures
- New interactive player experience
  - Sport games

Source : http://wiiaccessories.org/wii-remote-accessories/

Possible extensions to embedded systems

- Control systems
  - RC Car, Home automation, Lawn mower
- Music instrument
- Keynote Tool
- Virtual reality
  - Head tracking using IR

Bluetooth on the WiiMote

- The WiiMote communicates with hosts through standard bluetooth
- Connecting with a device does not require a bluetooth pairing
- Bluetooth pairing is beneficial if a disconnection occurs
- Two communication protocols are used within bluetooth communication
    - HCI- Configure host to pair with WiiMote
    - HID- Communicate between WiiMote and host

Connecting to the Wiimote without pairing

Press the 1+2 buttons on the Wiimote or the sync button to make the Wiimote discoverable

Establish an HID connection through the Wiimote and bluetooth module through firmware on the host-side. This varies based on bluetooth module

Utilize the HID commands to configure data reporting, read from memory, etc

Configuring Data Formatting

- Use HID ID 0x12 to configure how the Wiimote reports its “status”
- Need to report in two instances, when setting report type manually and when receiving status report (0x20) unprompted

12 TT MM

HID ID	0x12, the ID of the configuration packet
Continuous	Bit 2, (0x04) signifies the Wiimote always sending data regardless of change. Else (0x00), the Wiimote will only send data when it has changed
Reporting Mode	Sets the type of report the Wiimote should sent. Each type sends a different combination of buttons, accelerometer, IR camera, etc.

Different Reporting Modes

Report Type	Format of Packet
0x30 - Core buttons	30 BB BB
0x31 - Buttons + Accelerometer	31 BB BB AA AA AA
0x33 - Buttons + Accelerometer + IR	33 BB BB AA AA AA II II II II II II II II II II II II
0x35 - Buttons + Accelerometer + 16 Extension Bits	35 BB BB AA AA AA EE (16x)

Buttons

- Packets of two bytes
- Select desired data from packet with mask
- 1 means pushed
- Power button and sync button are unique
- **30 BB BB**

Bit	Mask	First Byte	Second Byte
0	0x01	D-Pad Left	Two
1	0x02	D-Pad Right	One
2	0x04	D-Pad Up	B
3	0x08	D-Pad Down	A
4	0x10	Plus	Minus
5	0x20		
6	0x40		
7	0x80		Home

Accelerometer

- 3 Bytes of data: XX YY ZZ
- Unsigned bytes: Zero Acceleration is 0x80
- 10 bit values
- Extra bits encoded in buttons

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1		X[1]	X[0]					
2		Z[1]	Y[1]					

Speaker

- 21mm piezo-electric speaker
- HID 0x14 for Speaker Enable
- HID 0x19 for Speaker Mute
  - Ex: (a2) 19 04
- HID 0x18 to write Speaker Data
- First byte is data length shifted by 3 bits
- 1 - 20 data bytes + padding
  - Ex: (a2) 18 LL DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD

Rumble

- Implemented with small motor
- LSB of EVERY output report sets rumble
  - Ex: (a2) 19 05 (Speaker Mute with Rumble On)
- Need to store state of rumble locally

Wii Remote Memory

- Memory consists of 16 kB EEPROM chip
- One part contains firmware for microcontroller
- Other part is generic memory space
  - Written to Virtual Memory addresses 0x0000-0x16FF
  - Contains calibration data and data storage for Mii characters

http://wiibrew.org/wiki/File:Wii_Remote_Flash.jpg

Reading and Writing To Memory

- Both read and write commands use big endian format
- Each transaction can either read or write 16 bytes

Read (HID 0x17):

Write (HID 0x16):

Memory Map

0x0000	Calibration Data
0x002A	User Data
0X0FCA	Mii 1 Data
0X12BA	Mii 2 Data
0X15AA	Unknown
0X16D0	Unknown

IR Calibration Block

- Two blocks of calibration data
- Last byte of each is a checksum (all prior bytes XORed plus 0x55)

```
0000: A1 AA 8B 99 AE 9E 78 30 A7 74 D3 A1 AA 8B 99 AE
0010: 9E 78 30 A7 74 D3 82 82 82 15 9C 9C 9E 38 40 3E
0020: 82 82 82 15 9C 9C 9E 38 40 3E 00 00 00 00 00 00
```

Accelerometer Calibration Block

- Stores the calibrated zero offsets for the accelerometer
- The values are the force of gravity on each axis of the accelerometer (X, Y, and Z)

```
0000: A1 AA 8B 99 AE 9E 78 30 A7 74 D3 A1 AA 8B 99 AE
0010: 9E 78 30 A7 74 D3 82 82 82 15 9C 9C 9E 38 40 3E
0020: 82 82 82 15 9C 9C 9E 38 40 3E 00 00 00 00 00 00
```

Control Registers

Address	Name	Function
0xA20000	Speaker Settings	Sets sample rate, frequency, and volume of speaker
0xA40000	Extension Controller Settings	Stores unique ID of the external controllers plugged in to the remote
0xA60000	Wii Motion Plus Settings	Indicates whether a Wii Motion plus extension is plugged into the remote and its current state
0xB00000	IR Camera Settings	Stores settings for light intensity sensitivity

Questions?

Interested in learning more?

http://wiibrew.org/wiki/Wiimote

HCI vs. HID commands

HCI	HID
Pseudo-Protocol that standardizes communication between the host and actual bluetooth IC	Controls the actual bluetooth communication between devices, used for keyboards, mice, etc.
Used to set the settings in your embedded system to be able to connect with the Wii-mote	Each communication “packet” contains an ID detailing the context of the data being sent
Necessary to do a bluetooth pairing with devices	Bluetooth module needs to be able to communicate with HID to communicate with the Wiimote