User Guide

Wii Nunchuk Remote Controller for Arduino

A great invention of Nintendo is the Nunchuk, a cheap extension for the Wii remote control. As it uses I2C as transportation protocol, it's easy to access the raw data of the controller. The Wii Nunchuck is loaded with exciting features: 2-Axis joystick, two buttons (called button C and button Z) and a 3 axis ±2g accelerometer. Any device capable of I2C interface can communicate with it, including Arduino & Rasberry Pi controller board !

SKU: MCH-1040

Brief Data:

- Standard I2C interface.
- 3-axis accelerometer with 10-bit accuracy.
- 2-axis analog joystick with 8-bit A/D converter.
- 2 buttons.
- Operating Voltage: 3.3V.
- Cable length: 100cm/39.37".

Mechanical Dimension:

Figure-1; Mechanical Dimension

Connector Pin Assignment:

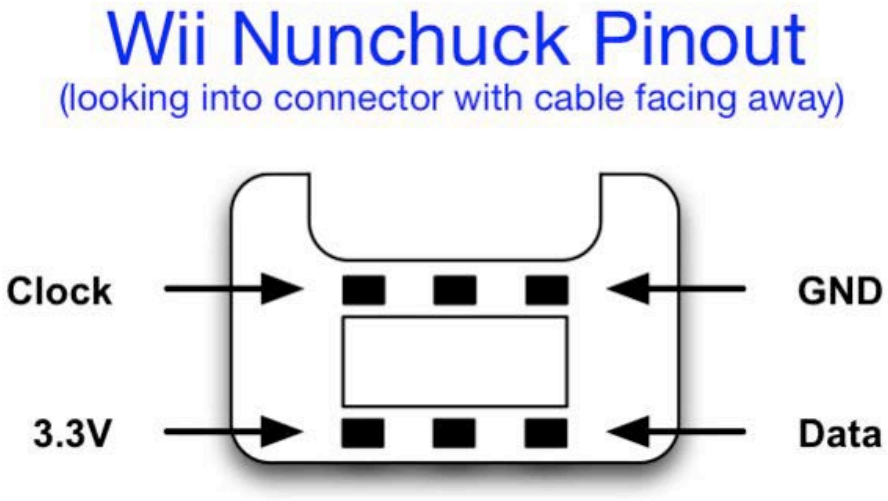


Figure-2: Nunchuk Socket Pin Assignment

Interface to Arduino Controller Board:

To make connection from Nunchuk's 6-pin connector to Arduino board without cutting the cable, we need a small break-out board. This break-out board can be easily inserted into the 6-pins Nunchuk connector securely. The other end of this board is standard 2.54mm header pin which can insert to breadboard or use with jumper wires.

Figure-3: Nunchuk to Arduino Adapter Board

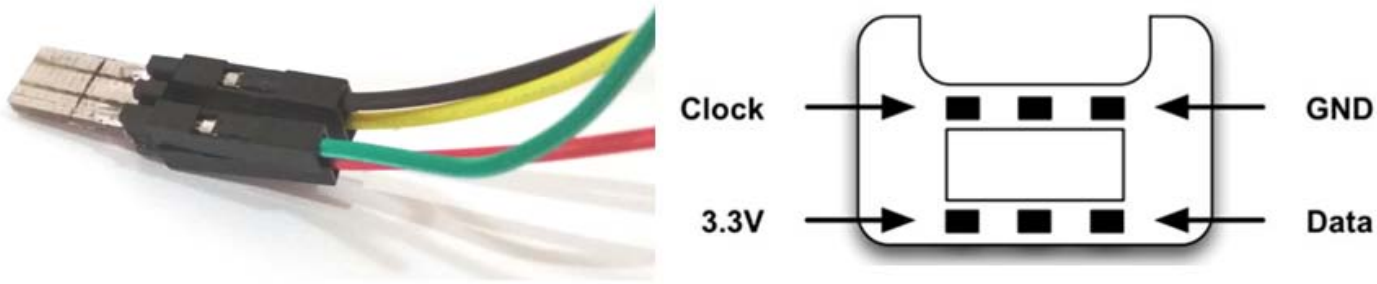


Figure-4

Connection from Nunchuk controller (using adapter board) to Arduino board:

Table-1:

Nunchuk Adapter Pin	Arduino Pin
Clock (Green Wire)	Analog-In A5 (SCL)
Data (Yellow Wire)	Analog-In A4 (SDA)
3.3V Supply (Red Wire)	3.3V
Ground (Black Wire)	GND

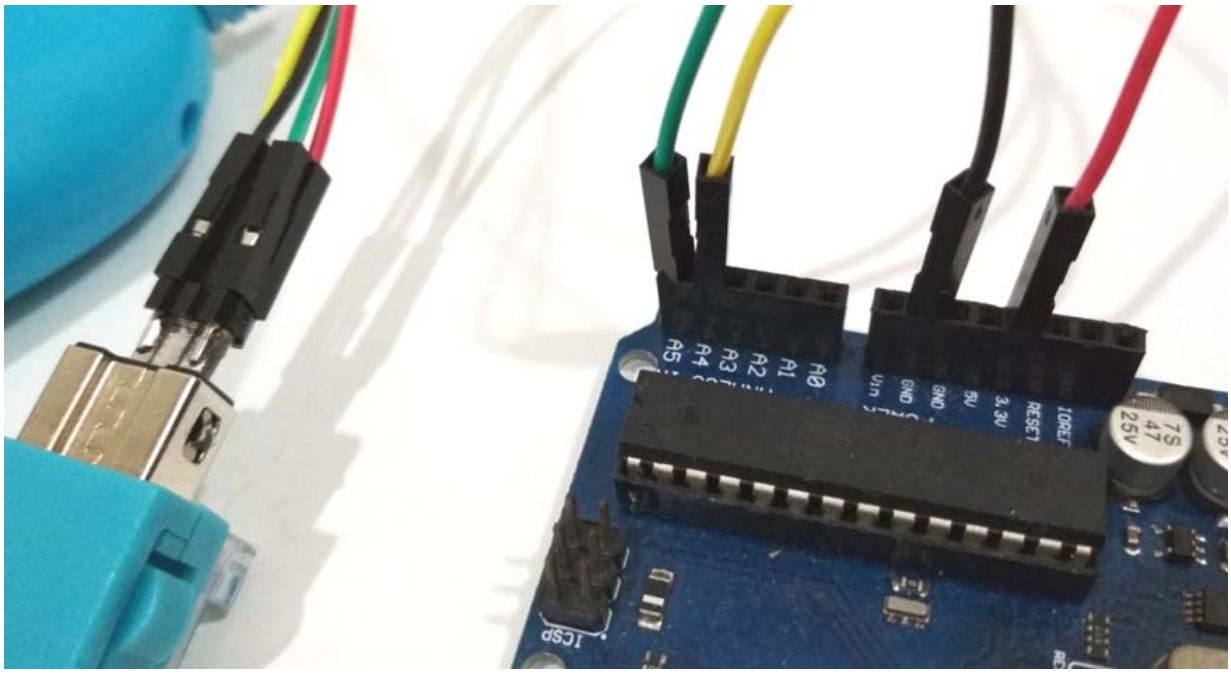
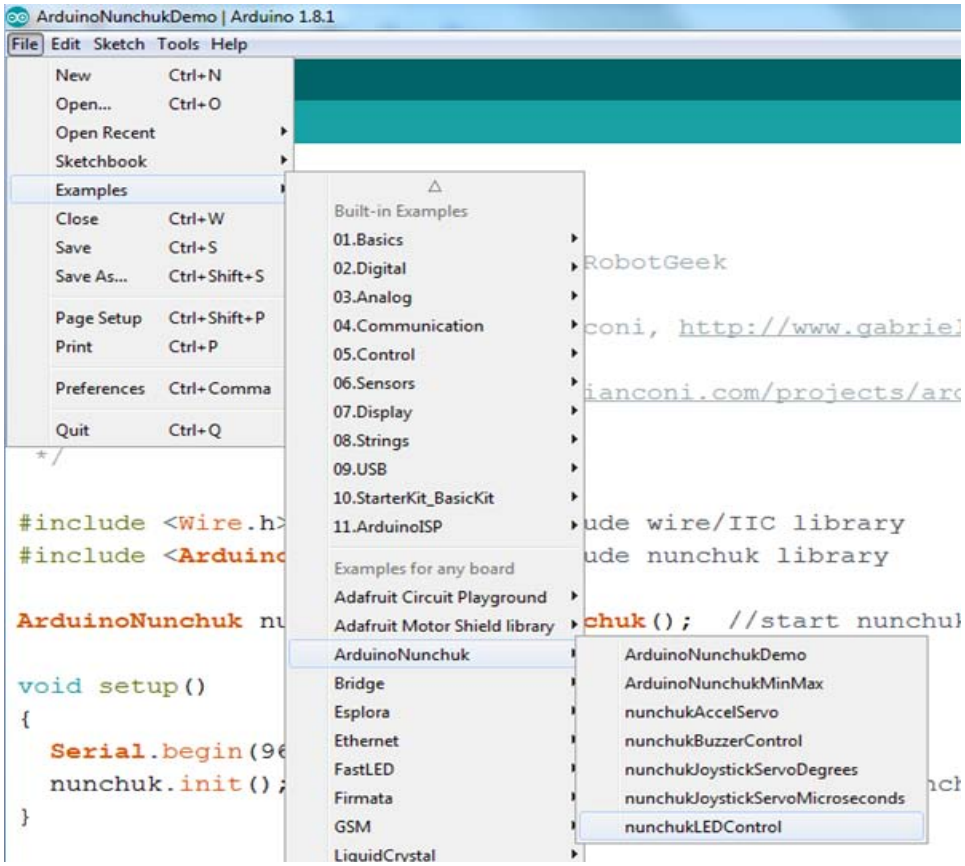


Figure-5: Nunchuk to Arduino Connection

Next we will perform a simple LED test by pressing the 2-buttons on the Nunchuck controller. First we need a library to do this job, download this library "ArduinoNunchuk" and extract it to Arduino IDE libraries folder.

Open the Arduino IDE and locate "nunchuk LEDControl" sketch:



```
/*=============================================================================
//  Author      : Handson Technology
//  Project     : Arduino Uno with Nunchuk Remote Controller
//  Description : Nunchuk LED Control
//  Source-Code : nunchukLEDControl.ino
//=============================================================================
*/

#include <Wire.h>
#include <ArduinoNunchuk.h>

ArduinoNunchuknunchuk=ArduinoNunchuk();

constint DIGTIAL_LED =13;
constint PWM_LED =3;

intpwmLEDValue;

void setup()
{
nunchuk.init();//start classy library
delay(100);
nunchuk.update();//read data from the classic controller
pinMode(DIGTIAL_LED, OUTPUT);
pinMode(PWM_LED, OUTPUT);
}

void loop()
{

nunchuk.update();//read data from the classic controller


//the joystick is already 0-255, so we can write it directly to PWM

analogWrite(PWM_LED,nunchuk.analogY);//write the PWM value to the LED

//if a is pressed, turn the digital LED on
if(nunchuk.zButton== true)
{
digitalWrite(DIGTIAL_LED, HIGH);
}

//if b is pressed, turn the digital LED off
if(nunchuk.cButton== true)
{
digitalWrite(DIGTIAL_LED, LOW);
}

}
```

Pressing the “Z Button” on the Nunchuk controller will turn On the Arduino on-board LED (L). Pressing “C Button” will turn Off the LED.

By connecting another LED to digital pin 3 of Arduino board and push forward and backward of the Nuchuk Joystick, you can do a dimming control of this LED.

Congratulation! you have successfully established the I2C communication of Nunchuk remote controller with Arduino controller board.

There are plenty of exciting application of this Nunchuk controller with Arduino board including robotics, self-balancing 2-wheels car, remote control drones…

Wii-Nunchuk Data Format:

The Wii-Nunchuk is a slave I2Cbus device that output six bytes of data as follow:

Data byte receive								Address
Joystick X								0x00
Joystick Y								0x01
Accelerometer X (bit 9 to bit 2 for 10-bit resolution)								0x02
Accelerometer Y (bit 9 to bit 2 for 10-bit resolution)								0x03
Accelerometer Z (bit 9 to bit 2 for 10-bit resolution)								0x04
Accel. Z bit 1	Accel. Z bit 0	Accel. Y bit 1	Accel. Y bit 0	Accel. X bit 1	Accel. X bit 0	C-button	Z-button	0x05

Byte 0x00 : X-axis data of the joystick

Byte 0x01 : Y-axis data of the joystick

Byte 0x02 : X-axis data of the accellerometer sensor

Byte 0x03 : Y-axis data of the accellerometer sensor

Byte 0x04 : Z-axis data of the accellerometer sensor

Byte 0x05 : bit 0 as Z button status - 0 = pressed and 1 = release

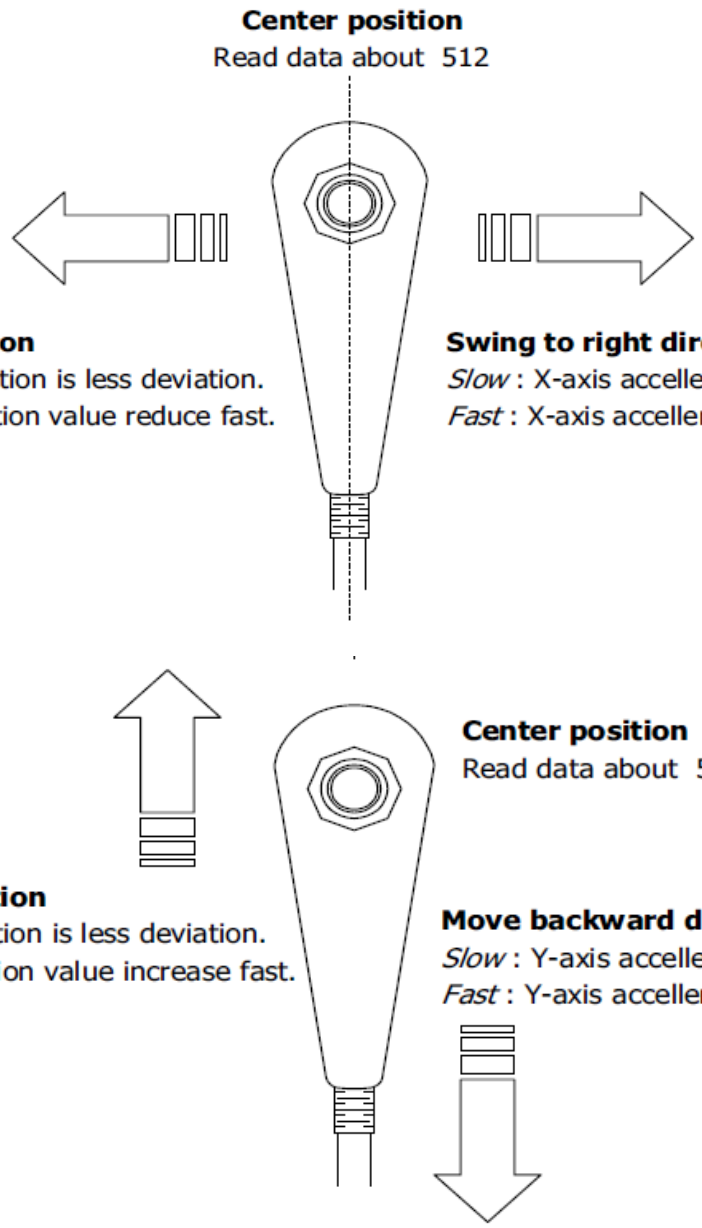
bit 1 as C button status - 0 = pressed and 1 = release

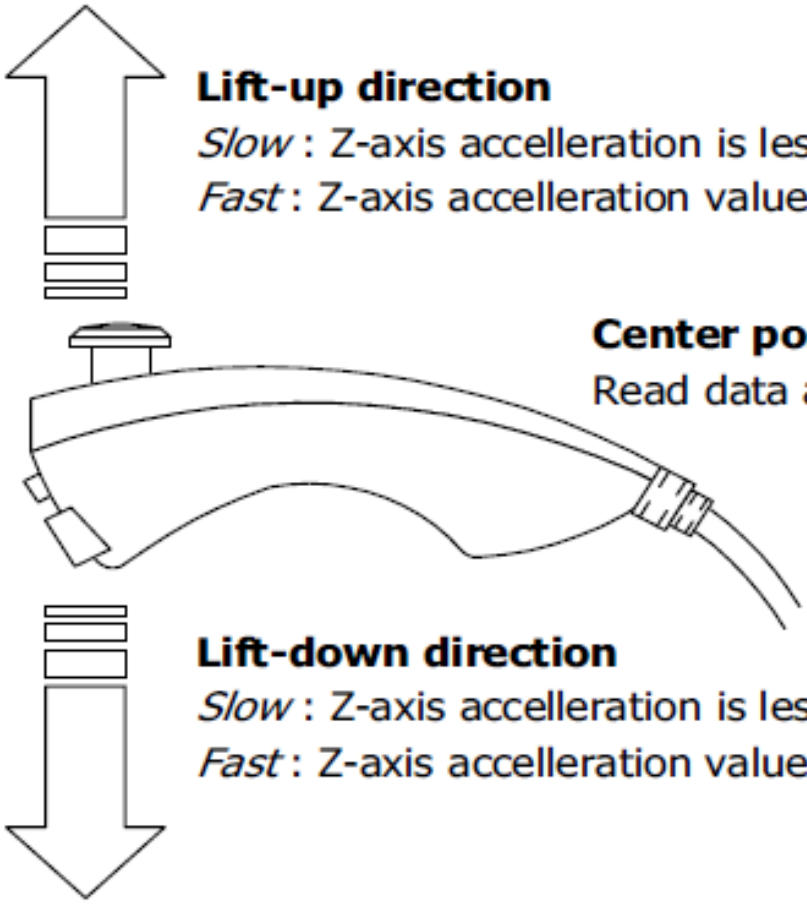
bit 2 and 3 as 2 lower bit of X-axis data of the accellerometer sensor

bit 4 and 5 as 2 lower bit of Y-axis data of the accellerometer sensor

bit 6 and 7 as 2 lower bit of Z-axis data of the accellerometer sensor

Wii-Nunchuk Postion Description:





Lift-up direction

Slow : Z-axis accelleration is less deviation.

Fast : Z-axis accelleration value increase fast.

Center position

Read data about 512

Lift-down direction

Slow : Z-axis accelleration is less deviation.

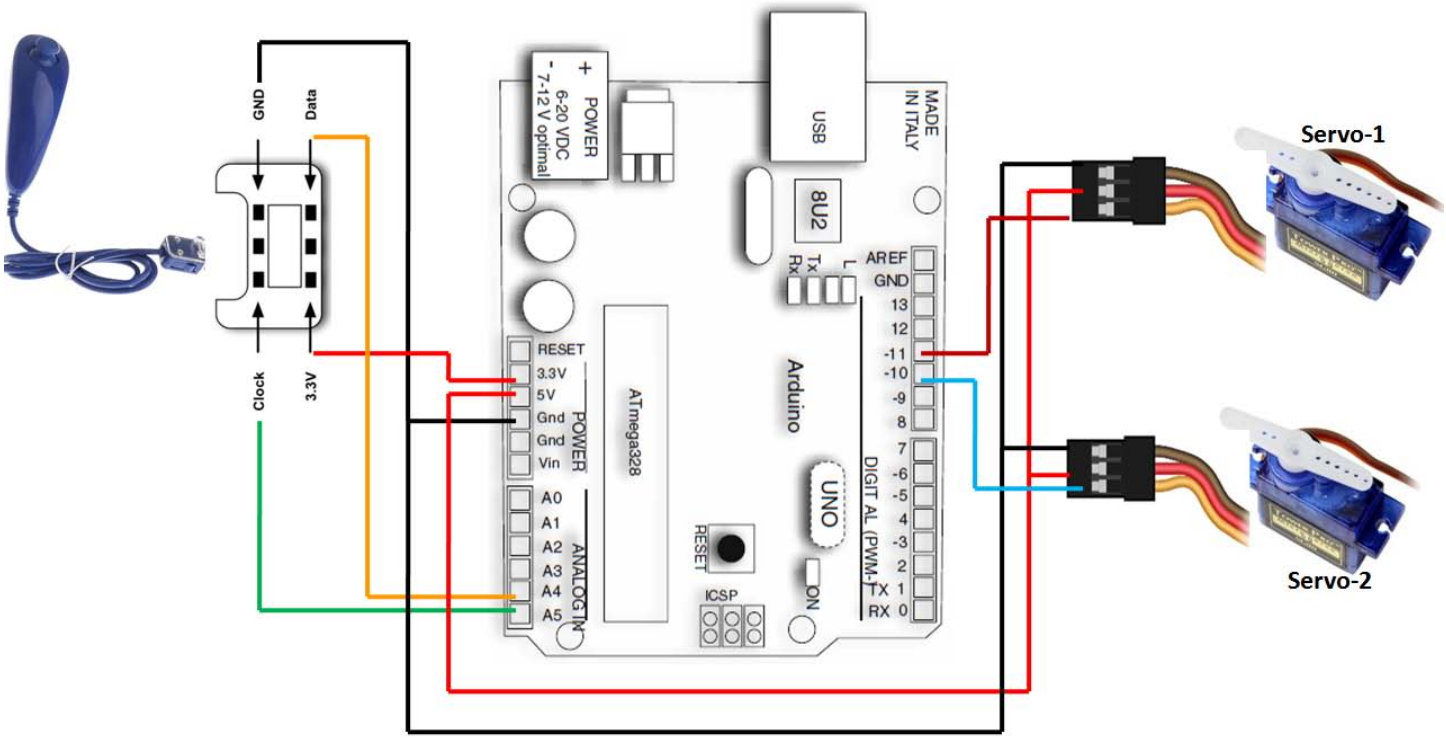
Fast : Z-axis accelleration value reduce fast.

Controlling Two Servo Motor with Wii Nunchuk.

Wii Nunchuk is versatile game controller that can be easily connected to Arduino board using I2C bus data format.
Wire up the circuit as shown in below diagram.

Parts needed in this task:

- Arduino Uno.
- Wii Nunchuk Controller with adapter board.
- Micro Servo Motor.
- Some jumper wired.



Arduino Sketch:

Copy and paste (or download from the below link) the below sketch into Arduino IDE and upload to Arduino Uno board. Once upload completed, try to move the Wii Nunchuk joystick forward, backward, left and right direction and observe the turning on the two servo motors attached to D10 & D11.

```
/*=============================================================================
//  Author      : Handson Technology (www.handsontec.com)
//  Project     : Wii Nunchuk Servo Control with Arduino Uno
//  Description : Controlling 2-servo motor with Wii Nunchuk Controller
//  Source-Code : nunchuk_2-servos.ino
//=============================================================================
*/

#include <Wire.h>
```

```
#include <Servo.h>

Servo servoLeft;          // Define left servo-1
Servo servoRight;         // Define right servo-2

static uint8_t nunchuck_buf[6];   // array to store nunchuck data,

void setup()
{
  Serial.begin(19200);
  servoLeft.attach(10);  // Set left servo-1 to digital pin 10
  servoRight.attach(11);  // Set right servo-2 to digital pin 9
  nunchuck_init(); // send the initilization handshake
  Serial.print ("Finished setup\n");
}

void loop()
{
  nunchuck_get_data();

// map nunchuk data to a servo data point
  int x_axis = map(nunchuck_buf[0], 23, 222, 180, 0);
  int y_axis = map(nunchuck_buf[1], 32, 231, 0, 180);

//move servo to desired position based on Wii nunchuk reading
  servoLeft.write(x_axis);
  servoRight.write(y_axis);

// un-comment next line to print data to serial monitor
//  nunchuck_print_data();

}

// Nunchuck functions
// initialize the I2C system, join the I2C bus,
// and tell the nunchuck we're talking to it

void nunchuck_init()
{   Wire.begin();                      // join i2c bus as master
    Wire.beginTransmission (0x52);
    Wire.write ((byte)0xF0);
    Wire.write ((byte)0x55);
    Wire.write ((byte)0xFB);
    Wire.write ((byte)0x00);
    Wire.endTransmission ();
}

// Send a request for data to the nunchuck

void nunchuck_send_request()
{
  Wire.beginTransmission(0x52);   // transmit to device 0x52
  Wire.write(0x00);               // sends one byte
  Wire.endTransmission();         // stop transmitting
}

// Receive data back from the nunchuck.
int nunchuck_get_data()
{
    int cnt=0;
    Wire.requestFrom (0x52, 6);     // request data from nunchuck
```

```
    while (Wire.available ()) {
      // receive byte as an integer
      nunchuck_buf[cnt] = nunchuk_decode_byte(Wire.read());
      cnt++;
    }
    nunchuck_send_request();  // send request for next data payload
    // If we recieved the 6 bytes, then go print them
    if (cnt >= 5) {
     return 1;   // success
    }
    return 0; //failure
}

// Print the input data we have received
// accel data is 10 bits long
// so we read 8 bits, then we have to add
// on the last 2 bits.  That is why I
// multiply them by 2 * 2

void nunchuck_print_data()
{
  static int i=0;
  int joy_x_axis = nunchuck_buf[0];
  int joy_y_axis = nunchuck_buf[1];

  int accel_x_axis = nunchuck_buf[2]; // * 2 * 2;
  int accel_y_axis = nunchuck_buf[3]; // * 2 * 2;
  int accel_z_axis = nunchuck_buf[4]; // * 2 * 2;

  int z_button = 0;
  int c_button = 0;

  // byte nunchuck_buf[5] contains bits for z and c buttons
  // it also contains the least significant bits for the accelerometer data
  // so we have to check each bit of byte outbuf[5]
  if ((nunchuck_buf[5] >> 0) & 1)
    z_button = 1;
  if ((nunchuck_buf[5] >> 1) & 1)
    c_button = 1;

  if ((nunchuck_buf[5] >> 2) & 1)
    accel_x_axis += 2;
  if ((nunchuck_buf[5] >> 3) & 1)
    accel_x_axis += 1;

  if ((nunchuck_buf[5] >> 4) & 1)
    accel_y_axis += 2;
  if ((nunchuck_buf[5] >> 5) & 1)
    accel_y_axis += 1;

  if ((nunchuck_buf[5] >> 6) & 1)
    accel_z_axis += 2;
  if ((nunchuck_buf[5] >> 7) & 1)
    accel_z_axis += 1;

  Serial.print(i,DEC);
  Serial.print("\t");

  Serial.print("joy:");
  Serial.print(joy_x_axis,DEC);
  Serial.print(",");
```

```
  Serial.print(joy_y_axis, DEC);
  Serial.print("  \t");

  Serial.print("acc:");
  Serial.print(accel_x_axis, DEC);
  Serial.print(",");
  Serial.print(accel_y_axis, DEC);
  Serial.print(",");
  Serial.print(accel_z_axis, DEC);
  Serial.print("\t");

  Serial.print("but:");
  Serial.print(z_button, DEC);
  Serial.print(",");
  Serial.print(c_button, DEC);

  Serial.print("\r\n");  // newline
  i++;
}

// Encode data to format that most wiimote drivers except
// only needed if you use one of the regular wiimote drivers
char nunchuk_decode_byte (char x)
{
  x = (x ^ 0x17) + 0x17;
  return x;
}
```

The above sketch is self explanatory with the detail comments; we will skip further code anatomy in the brief application guide.

Watch this demo in our youtube video.

https://www.youtube.com/watch?v=ULD0sUijLSw