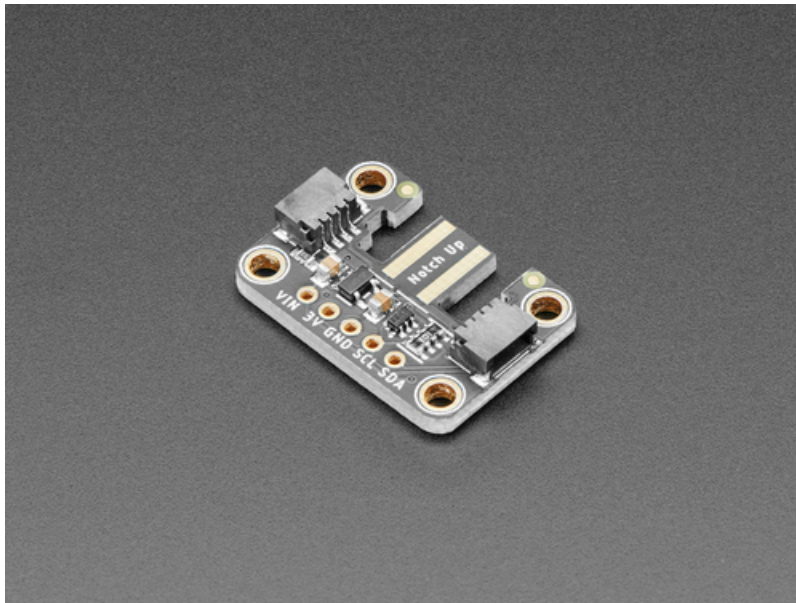




## Adafruit Wii Nunchuck Breakout Adapter

Created by Kattni Rembor



Last updated on 2021-10-22 11:44:48 AM EDT

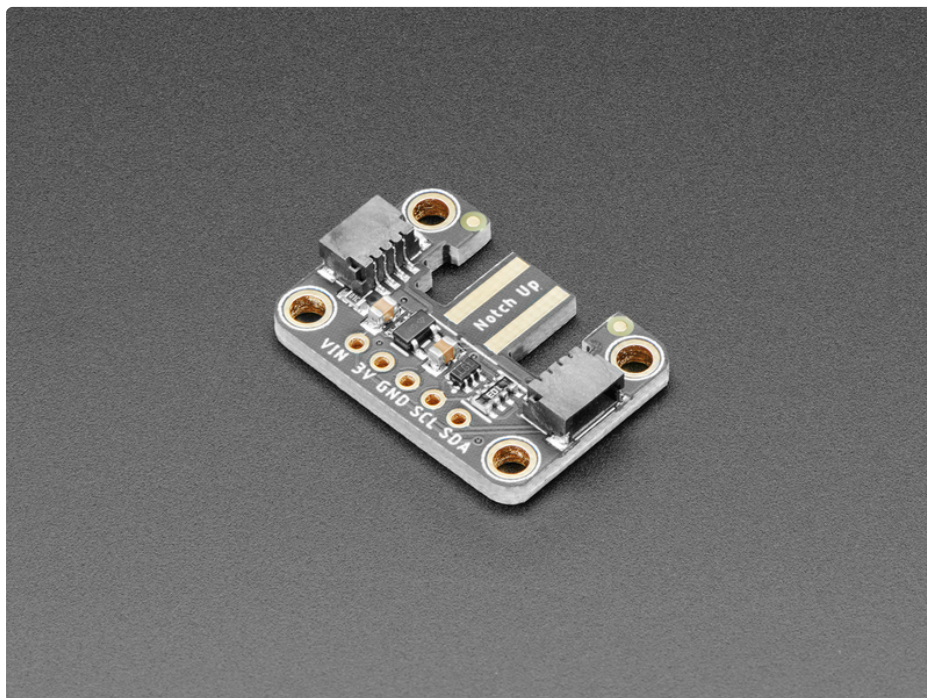
## Guide Contents

Guide Contents	2
Overview	3
Pinouts	6
Power Pins	6
I2C Logic Pins	6
Wii Nunchuck Connector	6
Arduino Use	8
Python & CircuitPython	11
CircuitPython Microcontroller Wiring	11
Python Computer Wiring	11
CircuitPython Installation of Nunchuck Library	12
Python Installation of Nunchuck Library	13
CircuitPython & Python Usage	13
Full Example Code	14
Python Docs	15
Downloads	16
Files:	16
Schematic	16
Fab Print	16
Sample Project: Nunchuck NeoPixel Ring	18
Parts	18
Wiring	19
Code	19

# Overview

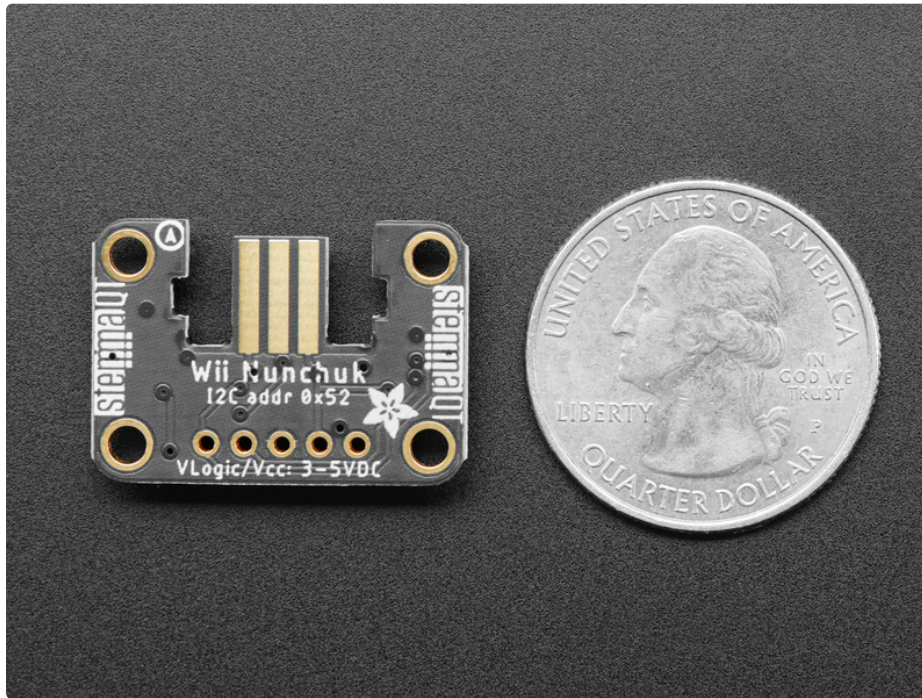


Dig out that old Wii controller and use it as a sleek controller for your next robot if you like. The Adafruit Adafruit Wii Nunchuck Breakout Adapter fits snugly into the Wii connector, and performs the level shifting and power regulation needed to use the controller with any microcontroller or microcomputer.

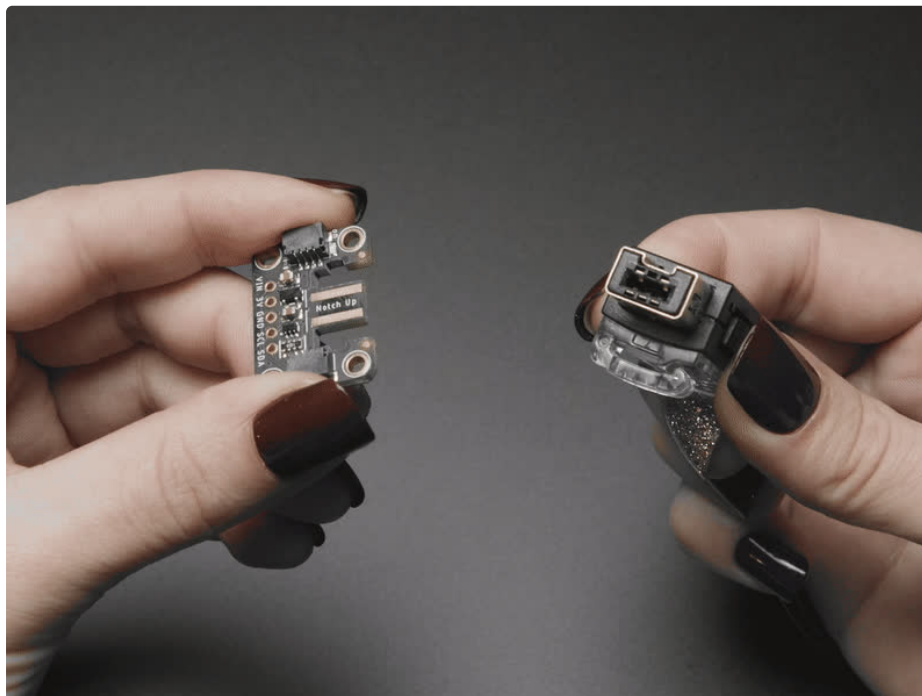


The Wii controllers use a standard I2C interface, and [there's existing code for both](#)

[Arduino \(https://adafru.it/PbS\)](https://adafru.it/PbS) and [CircuitPython/Python for quick integration \(https://adafru.it/PbT\)](https://adafru.it/PbT) with an Arduino UNO, Feather, or even a Raspberry Pi. We like to use these with the Wii Nunchuck, as you can get an X-Y joystick, two buttons and an accelerometer all in one hand-held package. All data is transmitted over I2C address 0x52, and the address can not be changed.

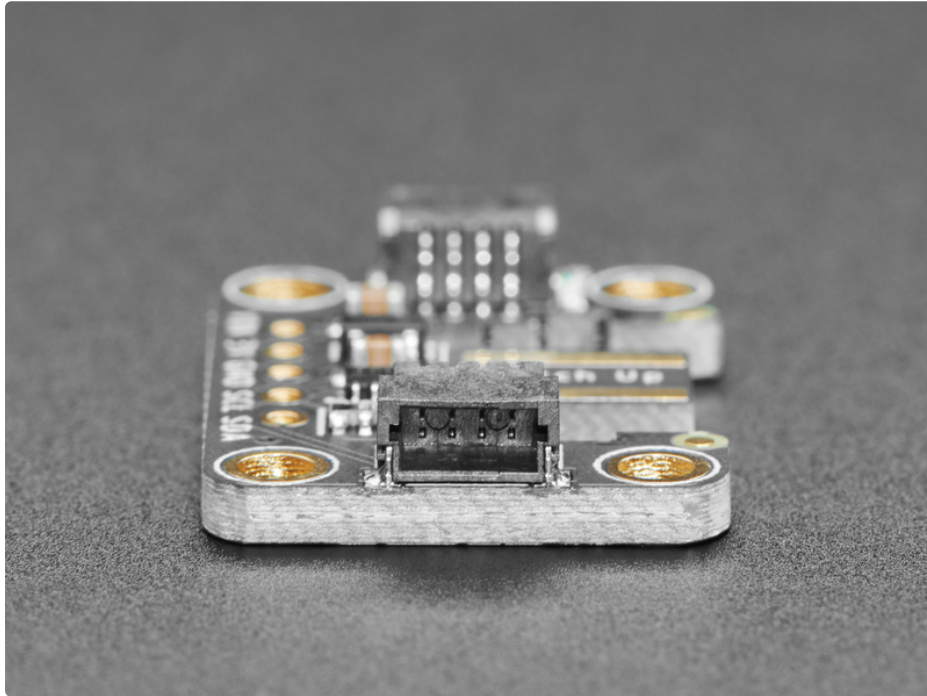


We use extra thicc 2.0mm PCBs for this breakout, and made cut-outs for the grabber-notches, so that the controller connection is snug, and wont rattle or come loose!

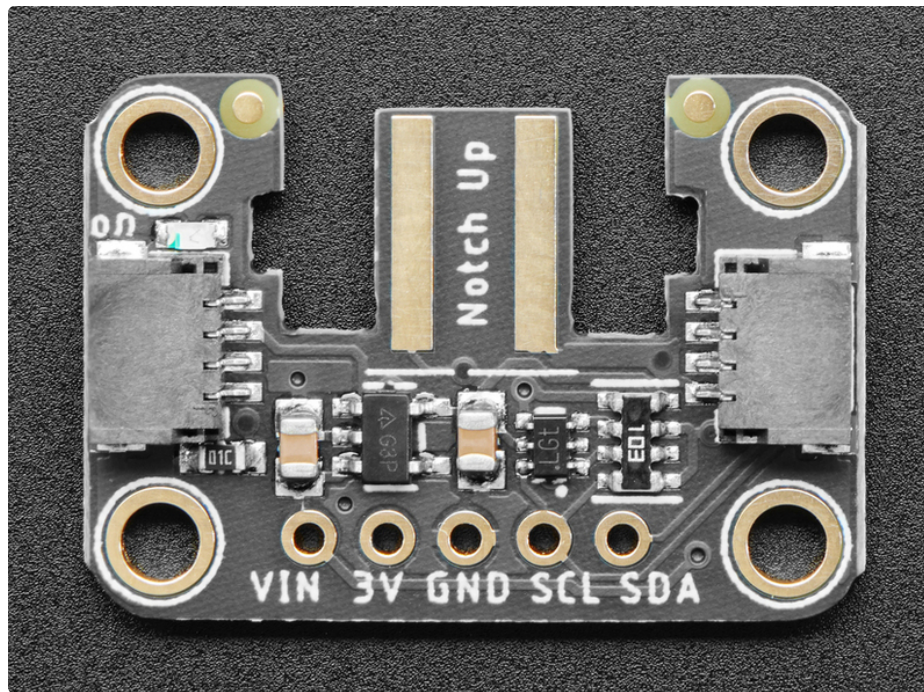


To make using it as easy as possible, we've created this breakout in [Stemma QT form factor](https://adafru.it/P3F) (<https://adafru.it/P3F>). You can either use a breadboard or the [SparkFun qwiic](https://adafru.it/Fpw) (<https://adafru.it/Fpw>) compatible [STEMMA QT](https://adafru.it/Ft4) (<https://adafru.it/Ft4>) connectors, and compatibility with 5V voltage levels as commonly found on [Arduinos](https://adafru.it/P4a) (<https://adafru.it/P4a>), as well as 3.3V logic used by many other boards like the Raspberry Pi or our Feathers. **QT Cable is not included**, but we have a variety in the [shop](https://adafru.it/JnB) (<https://adafru.it/JnB>) for quick plug-and-play support.

[Doesn't come with the Wii Nunchuck controller, that's sold separately](https://adafru.it/eQ5) (<https://adafru.it/eQ5>) (or just look in that plastic bin in your parent's attic next time you visit).







## Power Pins

- **VIN** - This is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller like Arduino, use 5V
- **3V** - This is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like.
- **GND** - common ground for power and logic

## I2C Logic Pins

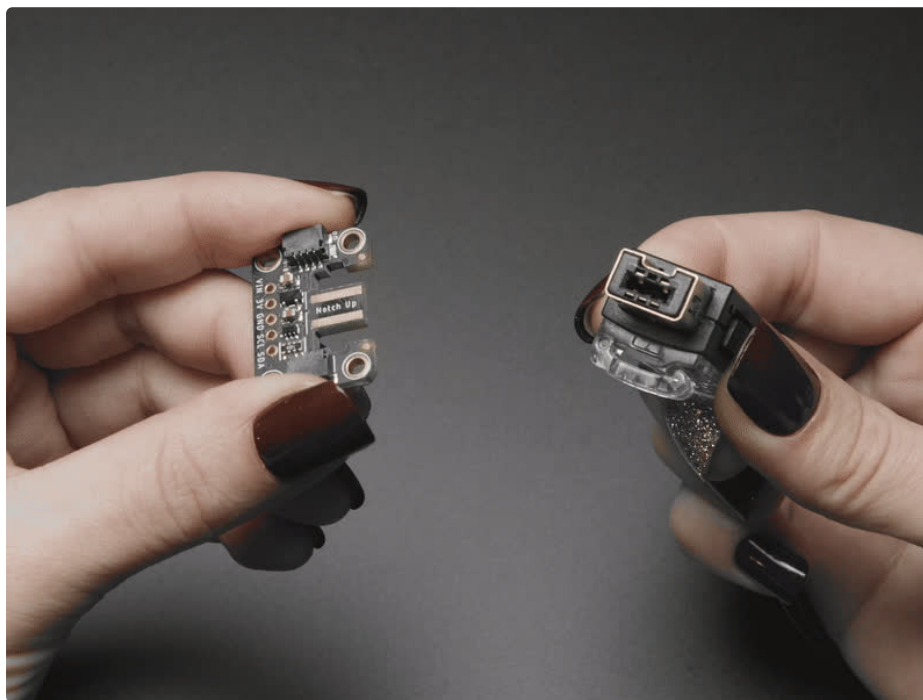
Default address is 0x52 - this cannot be changed.

- **SCL** - I2C clock pin, connect to your microcontroller I2C clock line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- **SDA** - I2C data pin, connect to your microcontroller I2C data line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- **STEMMA QT** (<https://adafru.it/Ft4>) - These connectors allow you to connect to dev boards with **STEMMA QT** connectors or to other things with [various associated accessories](https://adafru.it/Ft6) (<https://adafru.it/Ft6>)

## Wii Nunchuck Connector

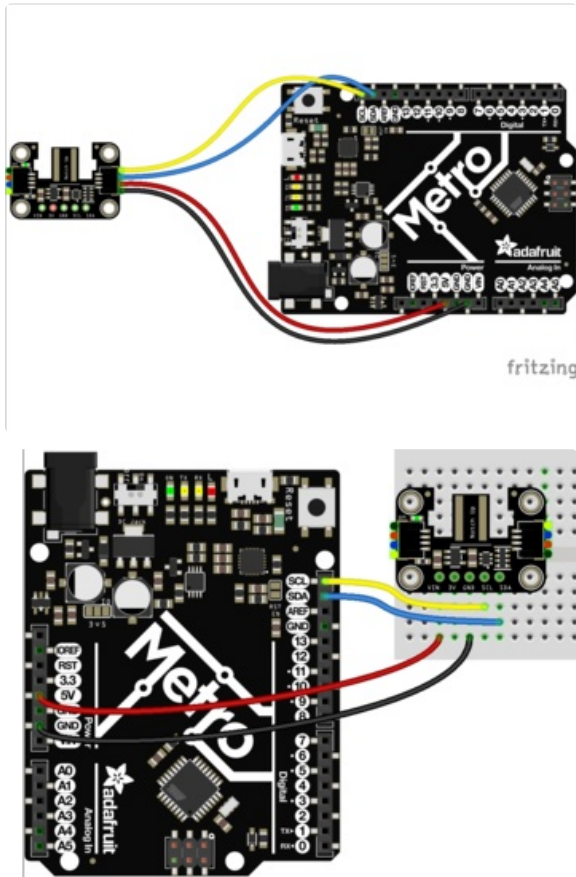
Wii controllers use a standard I2C interface. We use extra thick 2.0mm PCBs for this breakout, and made cut-outs for the grabber-notches, so that the controller connection is snug, and wont rattle or come loose!

See the GIF for how to install. The **notched side of the connector U** goes on the side that says **Notch Up!**



# Arduino Use

Connect the Wii Nunchuck Breakout Adapter as shown below using the STEMMA QT connector or a solderless breadboard.



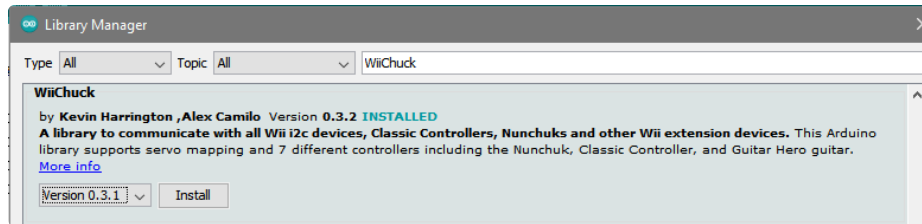
- Connect **board VIN** (red wire) to **Arduino 5V** if you are running a **5V** board Arduino (Uno, etc.). If your board is **3V**, connect to that instead.
- Connect **board GND** (black wire) to **Arduino GND**
- Connect **board SCL** (yellow wire) to **Arduino SCL**
- Connect **board SDA** (blue wire) to **Arduino SDA**

There are existing Arduino libraries you can use with the Wii series of devices. [We'll be using the popular WiiChuck library \(https://adafru.it/PbS\)](https://adafru.it/PbS) which has support for the following devices

- Nunchuk
- Classic Controller
- Guitar Hero Guitar
- Guitar Hero Drums
- DJ Hero
- Drawesome Tablet
- Taiko Drums

Install it by searching the Arduino Library Manager for **WiiChuck**





The WiiChuck library is very fully featured, using a lot of RAM and Flash and we don't recommend it for use with Arduino compatibles that use ATmega32x's like the UNO/Leonardo/32u4 because its easy to overuse the memory and go unstable. Please pick a board with over 4K of RAM!

You can use this sketch, again it barely fits on a 32u4 or 328 Arduino but it will connect and display data from a Nunchuk type controller!

```
#include <WiiChuck.h>

Accessory nunchuck;

void setup() {
  Serial.begin(115200);
  nunchuck.begin();
  if (nunchuck.type == Unknown) {
    nunchuck.type = NUNCHUCK;
  }
}

void loop() {
  nunchuck.readData();    // Read inputs and update maps

  Serial.print("X: "); Serial.print(nunchuck.getAccelX());
  Serial.print(" \tY: "); Serial.print(nunchuck.getAccelY());
  Serial.print(" \tZ: "); Serial.println(nunchuck.getAccelZ());

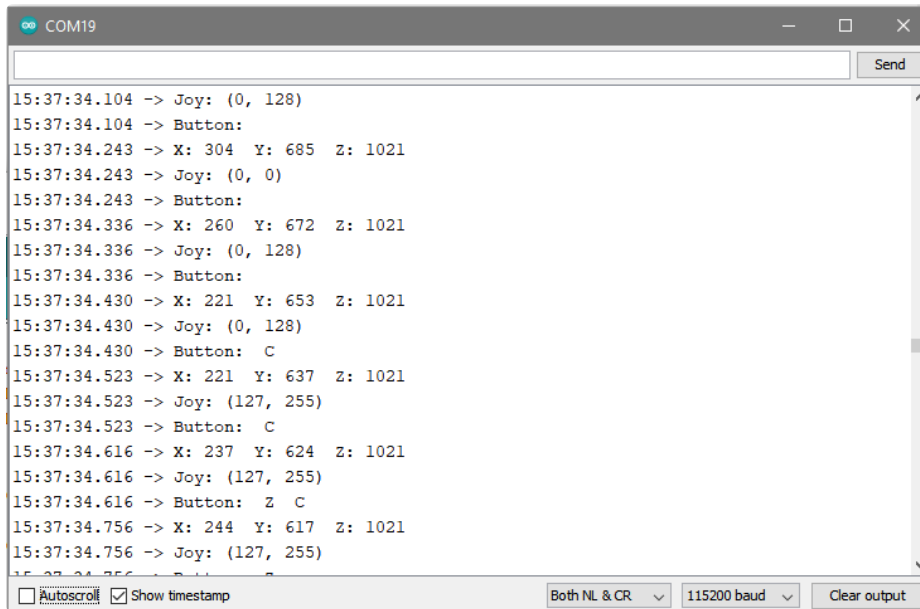
  Serial.print("Joy: (");
  Serial.print(nunchuck.getJoyX());
  Serial.print(", ");
  Serial.print(nunchuck.getJoyY());
  Serial.println(")");

  Serial.print("Button: ");
  if (nunchuck.getButtonZ()) Serial.print(" Z ");
  if (nunchuck.getButtonC()) Serial.print(" C ");

  Serial.println();
  delay(100);
}
```

Open the serial console at 115200 baud to see data streaming out, you can see the X Y Z accelerometer data (ranges from 0 to 1023), X and Y from the thumbstick (ranges from 0-255, with ~127 in the center

position), and the two trigger buttons Z and C



```
COM19
15:37:34.104 -> Joy: (0, 128)
15:37:34.104 -> Button:
15:37:34.243 -> X: 304 Y: 685 Z: 1021
15:37:34.243 -> Joy: (0, 0)
15:37:34.243 -> Button:
15:37:34.336 -> X: 260 Y: 672 Z: 1021
15:37:34.336 -> Joy: (0, 128)
15:37:34.336 -> Button:
15:37:34.430 -> X: 221 Y: 653 Z: 1021
15:37:34.430 -> Joy: (0, 128)
15:37:34.430 -> Button: C
15:37:34.523 -> X: 221 Y: 637 Z: 1021
15:37:34.523 -> Joy: (127, 255)
15:37:34.523 -> Button: C
15:37:34.616 -> X: 237 Y: 624 Z: 1021
15:37:34.616 -> Joy: (127, 255)
15:37:34.616 -> Button: Z C
15:37:34.756 -> X: 244 Y: 617 Z: 1021
15:37:34.756 -> Joy: (127, 255)
15:37:34.756 -> Button:

 Autoscroll  Show timestamp Both NL & CR 115200 baud Clear output
```

# Python & CircuitPython

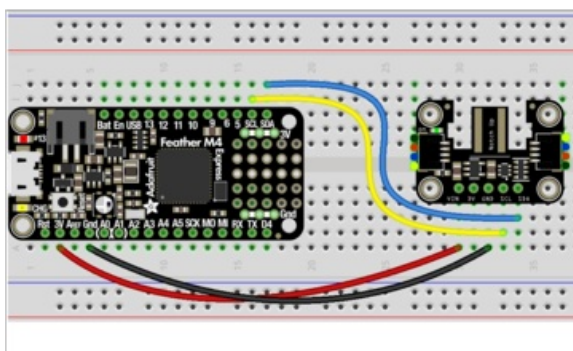
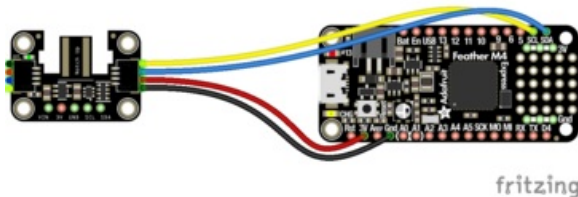
It's easy to use the Wii Nunchuck Breakout Adapter with CircuitPython and the [Adafruit CircuitPython Nunchuk \(https://adafru.it/PbT\)](https://adafru.it/PbT) module. This module allows you to easily write Python code that reads controls from the Wii Nunchuck.

You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python [thanks to Adafruit\\_Blinka, our CircuitPython-for-Python compatibility library \(https://adafru.it/BSN\)](https://adafru.it/BSN).

## CircuitPython Microcontroller Wiring

First wire up a Wii Nunchuck Adapter Breakout to your board exactly as follows. Here is an example of the Nunchuck Adapter wired to a Feather using I2C:

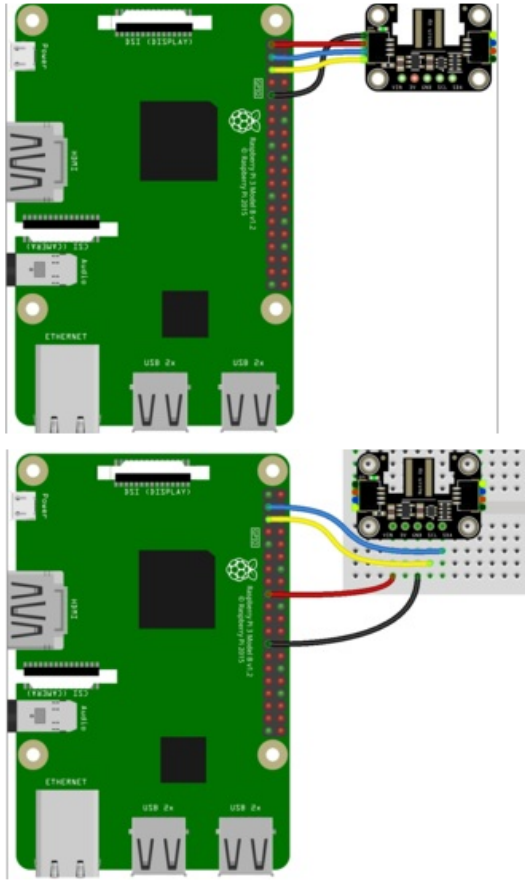
- Board 3V to sensor VIN (red wire)
- Board GND to sensor GND (black wire)
- Board SCL to sensor SCL (yellow wire)
- Board SDA to sensor SDA (blue wire)



## Python Computer Wiring

Since there's *dozens* of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported \(https://adafru.it/BSN\)](https://adafru.it/BSN).

Here's the Raspberry Pi wired with I2C:



- Pi 3V3 to sensor VIN (red wire)
- Pi GND to sensor GND (black wire)
- Pi SCL to sensor SCL (yellow wire)
- Pi SDA to sensor SDA (blue wire)

## CircuitPython Installation of Nunchuck Library

You'll need to install the [Adafruit CircuitPython Nunchuck](https://adafru.it/PbX) (<https://adafru.it/PbX>) library on your CircuitPython board.

First make sure you are running the [latest version of Adafruit CircuitPython](https://adafru.it/Amd) (<https://adafru.it/Amd>) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from [Adafruit's CircuitPython library bundle](https://adafru.it/ENC) (<https://adafru.it/ENC>). Our CircuitPython starter guide has [a great page on how to install the library bundle](https://adafru.it/ABU) (<https://adafru.it/ABU>).

Copy the following files from the bundle to the **lib** folder on your **CIRCUITPY** drive:

- **adafruit\_nunchuck.mpy**
- **adafruit\_bus\_device**

Before continuing make sure your board's **lib** folder or root filesystem has the **adafruit\_nunchuck.mpy**, and **adafruit\_bus\_device** file and folder copied over.

Next [connect to the board's serial REPL](https://adafru.it/Awz) (<https://adafru.it/Awz>) so you are at the CircuitPython >>> prompt.

## Python Installation of Nunchuck Library

You'll need to install the **Adafruit\_Blinka** library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. [Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready](#) (<https://adafru.it/BSN>)!

Once that's done, from your command line run the following command:

- `pip3 install adafruit-circuitpython-nunchuck`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

## CircuitPython & Python Usage

To demonstrate the usage of the sensor we'll initialize it and read from the controller in the board's Python REPL.

Run the following code to import the necessary modules and initialize the I2C connection with the adapter:

```
import time
import board
import adafruit_nunchuk

nc = adafruit_nunchuk.Nunchuk(board.I2C())
```

Now you're ready to read values from the controller using these properties:

- **joystick** - An (x, y) tuple of the joystick position.
- **button\_C** - The pressed state of button C.
- **button\_Z** - The pressed state of button Z.
- **acceleration** - An (x, y, z) tuple of acceleration data.

For example, to read the joystick position:

```
x, y = nc.joystick
print("joystick = {},{}".format(x, y))
```

To read button presses on the+ buttons:



```
print(nc.button_C)
print(nc.button_Z)
```

To read the acceleration data:

```
ax, ay, az = nc.acceleration
print("acceleration ax={}, ay={}, az={}".format(ax, ay, az))
```

That's all there is to using the Adafruit Wii Nunchuck Breakout Adapter with Python and CircuitPython!

## Full Example Code

This example reads all the data in a loop.

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_nunchuk

nc = adafruit_nunchuk.Nunchuk(board.I2C())

while True:
    x, y = nc.joystick
    ax, ay, az = nc.acceleration
    print("joystick = {},{}".format(x, y))
    print("acceleration ax={}, ay={}, az={}".format(ax, ay, az))

    if nc.buttons.C:
        print("button C")
    if nc.buttons.Z:
        print("button Z")
    time.sleep(0.5)
```

# Python Docs

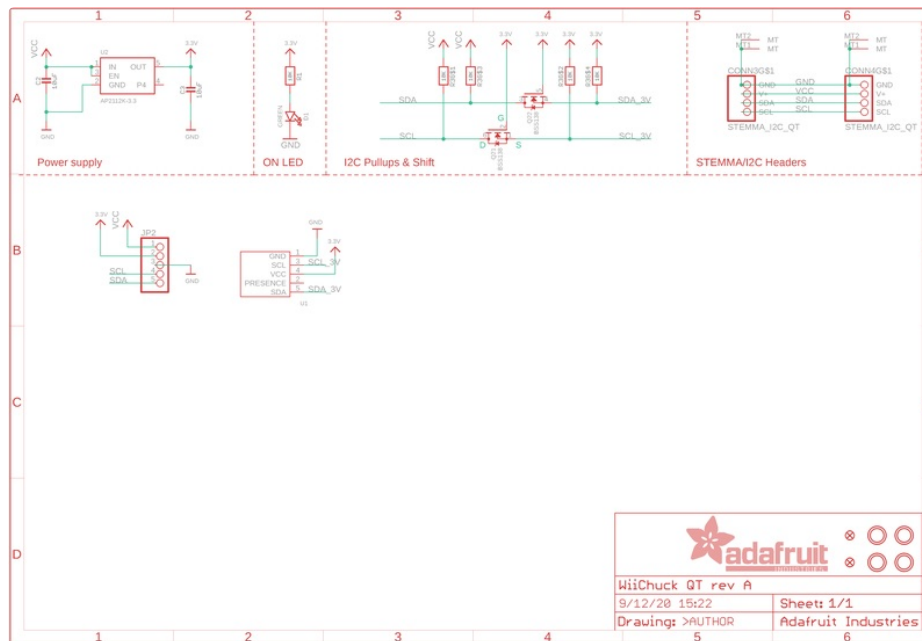
[Python Docs \(https://adafru.it/PbW\)](https://adafru.it/PbW)

# Downloads

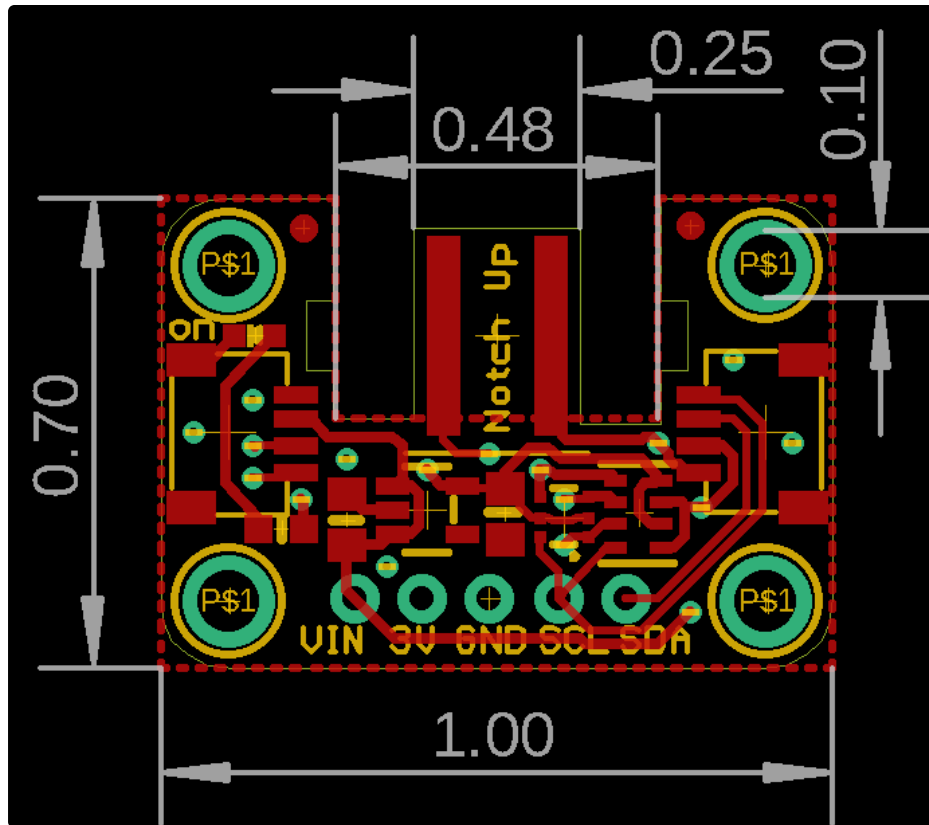
## Files:

- [Fritzing object in the Adafruit Fritzing Library \(https://adafru.it/PbU\)](https://adafru.it/PbU)
- [EagleCAD PCB files on GitHub \(https://adafru.it/PbV\)](https://adafru.it/PbV)

# Schematic



# Fab Print



# Sample Project: Nunchuck NeoPixel Ring

Here's an example project you can do with your Nunchuck breakout to control some NeoPixels with joystick, button, and motion from a Wii Nunchuck.

## Parts

Your browser does not support the video tag.

### [Adafruit Wii Nunchuck Breakout Adapter](#)

Dig out that old Wii controller and use it as a sleek controller for your next robot if you like. The Adafruit Adafruit Wii Nunchuck Breakout Adapter fits snugly into the Wii connector...

\$2.95

In Stock

Add to Cart

---

### [Wii controller \(Nunchuck / Wiichuck\)](#)

This is a generic Wii Nunchuck controller, we haven't tried it with a Wii but it does work great with the Video Game shield, and all the microcontroller code we tried. May come in...

\$12.50

In Stock

Add to Cart

---

You can use pretty much any CircuitPython-capable board for this, I had a Metro M4 Airlift handy.

### [Adafruit Metro M4 Express AirLift \(WiFi\) - Lite](#)

Give your next project a lift with AirLift - our witty name for the ESP32 co-processor that graces this Metro M4. You already know about the Adafruit Metro...

\$34.95

In Stock

Add to Cart

---

This will work with any NeoPixel strand, strip, or ring.

Your browser does not support the video tag.

### [NeoPixel 1/4 60 Ring - 5050 RGBW LED w/ Integrated Drivers](#)

What is better than smart RGB LEDs? Smart RGB+White LEDs! These NeoPixels now have 4 LEDs in them (red, green, blue and white) for excellent lighting effects. Round and round...

\$11.95

In Stock

Add to Cart

---



## STEMMA QT / Qwiic JST SH 4-pin to Premium Male Headers Cable

This 4-wire cable is a little over 150mm / 6" long and fitted with JST-SH female 4-pin connectors on one end and premium Dupont male headers on the other. Compared with the...

\$0.95

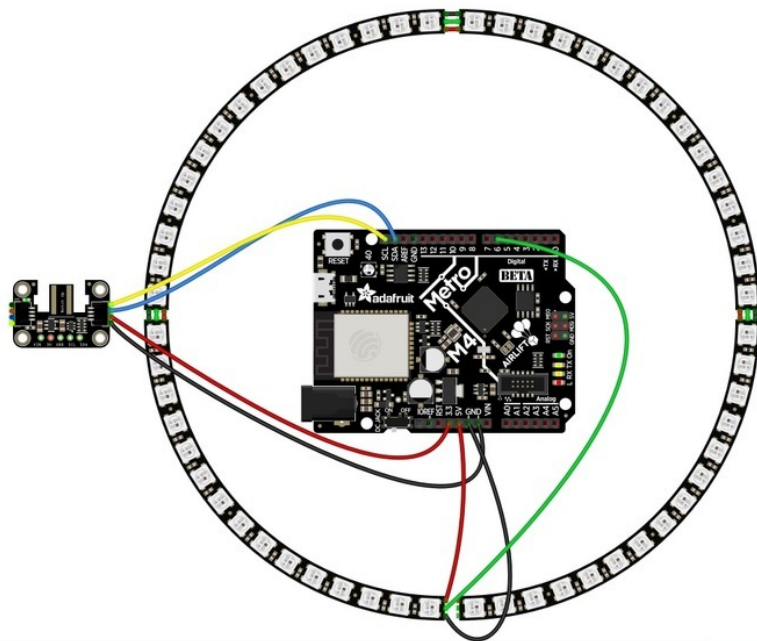
In Stock

Add to Cart

## Wiring

Use the STEMMA QT/qwiic to male header pin adapter to plug the Nunchuck breakout into your microcontroller's **3.3V**, **GND**, **SDA**, & **SCL** lines.

Wire your NeoPixels into the board's **5V**, **GND**, and **D6** (to NeoPixel **DIN**) pins (if your board only has 3.3V that's usually fine, too, but NeoPixels prefer 5V when available).



Then, plug your Nunchuck controller into the adapter, making sure to follow the silkscreened "NOTCH UP" guideline.

## Code

Set up your board as shown on the previous page, then download this code onto the board as code.py.

```

import time
import board
import adafruit_nunchuk
import neopixel
import simpleio

nc = adafruit_nunchuk.Nunchuk(board.I2C())
# create neopixel object
NEOPIN = board.D6
NEOLENGTH = 60
NEOORDER = neopixel.GRBW # set to GRB for 'regular' RGB NeoPixels
pixels = neopixel.NeoPixel(
    NEOPIN, NEOLENGTH, brightness=0.1, auto_write=False, pixel_order=NEOORDER
)

RED = (220, 0, 0)
PURPLE = (80, 0, 160)
PINK = (100, 0, 80)
GREEN = (0, 180, 0)
CYAN = (0, 80, 100)
BLUE = (0, 0, 255)
BLACK = (0, 0, 0)

COLORS = [RED, PURPLE, PINK, GREEN, CYAN, BLUE]
pix = 0 # selected pixel
color_pick = 0 # current color index
pixels.fill(BLACK)
pixels.show()

while True:
    x, y = nc.joystick # get joystick values
    ax, ay, az = nc.acceleration # get accelerometer values

    tilt_x = simpleio.map_range(ax, 300.0, 800.0, 0.0, 1.0) # remap tilt to brightness
    # remap y to current pixel
    pix = int(
        simpleio.map_range(y, 0, 255, 0, NEOLENGTH - 1)
    )

    if nc.button_C: # hold C button to use tilt for brightness
        pixels.brightness = tilt_x

    if nc.button_Z:
        color_pick = (color_pick + 1) % 4 # cycle through colors
        time.sleep(0.02) # debounce

    pixels.fill(BLACK) # turn off pixels
    for i in range(0, pix + 1): # light up all pixels up to the current one
        pixels[i] = COLORS[color_pick]

    pixels.show()

```

Now, you can control your LEDs with the nunchuck!

- move the joystick up and down to choose how many pixels are lit
- tap the Z button to change colors
- hold the C button while tilting the controller from side to side to adjust brightness. Let go of the C button to lock in that brightness level

