Guitar Hero MIDI Controller

Created by John Park

https://learn.adafruit.com/guitar-hero-midi-controller

Last updated on 2024-06-03 03:19:02 PM EDT

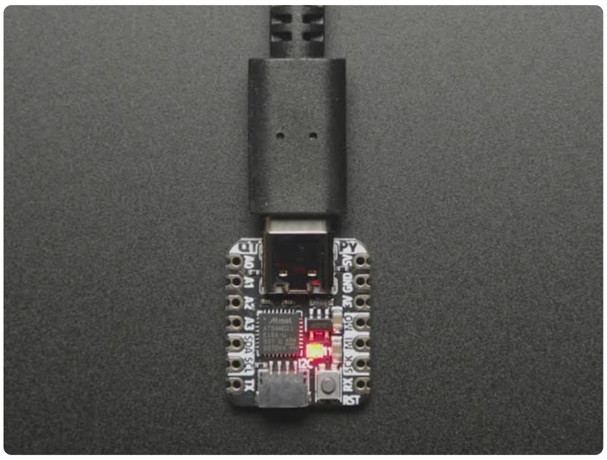
Table of Contents

Overview

You can turn an old Guitar Hero accessory into a USB MIDI controller for your synthesizer! Wii accessories use I2C to send all of their data, so we'll couple the Guitar Hero controller with a STEMMA QT Nunchuck Breakout and an Adafruit QT Py to read all the fret buttons, whammy bar, strum bar, and joystick data.

You'll be able to send MIDI notes, chords, octave changes, whammy bar pitch bends, and joystick CC messages to any software synth or a USB MIDI host-capable synthesizer.

Parts



Adafruit QT Py - SAMD21 Dev Board with STEMMA QT

What a cutie pie! Or is it… a QT Py? This diminutive dev board comes with our favorite lil chip, the SAMD21 (as made famous in our GEMMA M0 and Trinket M0 boards).This time it…

https://www.adafruit.com/product/4600

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…

https://www.adafruit.com/product/4836

STEMMA QT / Qwiic JST SH 4-pin Cable - 100mm Long

This 4-wire cable is a little over 100mm / 4" long and fitted with JST-SH female 4-pin connectors on both ends. Compared with the chunkier JST-PH these are 1mm pitch instead of...

https://www.adafruit.com/product/4210

USB C to USB C Cable - USB 3.1 Gen 4 with E-Mark - 1 meter long

As technology changes and adapts, so does Adafruit! Rather than the regular USB A, this cable has USB C to USB C plugs!USB C is the latest...

https://www.adafruit.com/product/4199

USB A to USB C Adapter

As technology changes and adapts, so does Adafruit, and speaking of adapting, this adapter has a USB A plug and a USB C socket so your older...

https://www.adafruit.com/product/4175

Wii Guitar Hero Controller

You have one in your closet. If not, ~$30 on eBay, less at a thrift shop.

Optional

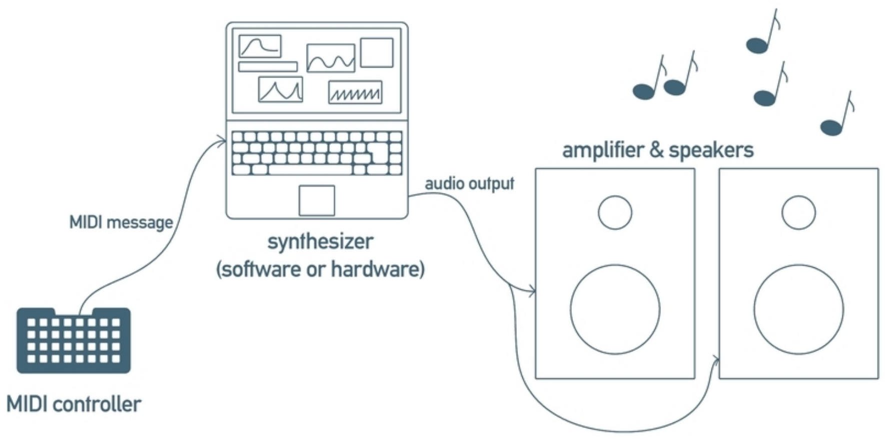
iOS Lightning to USB OTG Cable

Your iOS phone or tablet may not have a USB port on the bottom but that doesn't mean you can't use it to connect USB devices. Secretly known as a 'Camera Connector' or...

https://www.adafruit.com/product/3940

USB MIDI Essentials

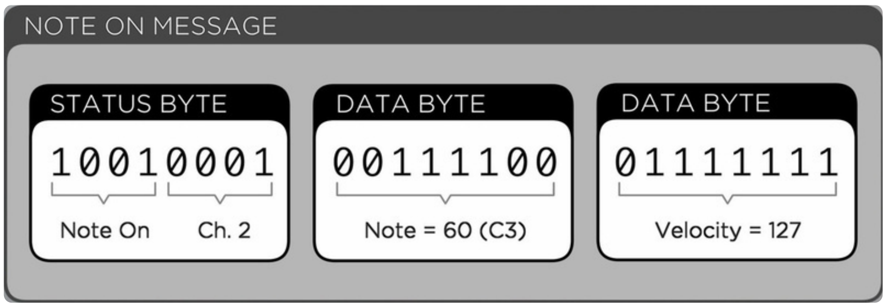
MIDI is a venerable protocol (dating back to 1983) that is used to communicate between synthesizers, controllers, sequencers, sample players, computers, mobile devices, drum machines, and other electronic music making devices.



MIDI Messages

A simple and very common use case is to have a controller, such as a piano-style keyboard, send **Note On** and **Note Off** data to a music synthesizer (including software synths on your computer or mobile device).

Press a key and a message is sent telling the synth to play a specific musical note. Release the key and a message is sent to the synthesizer telling it to stop playing that note.



Getting a bit fancier than simple on/off messages, MIDI can also be used to send "continuous controller" **CC** messages, typically the result of turning a knob or pushing a slider on the keyboard controller. These can be used to sweep through the cuttoff frequency of a low pass filter, or modulate a tremolo, and many, many other parameters.

A similar scheme is also used to send pitch bend info -- often represented by a pitch bend wheel on the left side of a MIDI keyboard.

USB MIDI uses the same message protocol as classic MIDI, but does so over a USB host/device transport.

This makes it very simple to plug MIDI devices/controllers into computers and iOS devices, which act as the MIDI host. This is great, because it means you can control software synthesizers using something nicer than a computer keyboard!

Wii Guitar Hero Controller MIDI

On its own, the Wii Guitar Hero controller sends analog signals via I2C for all of its seven buttons, strum bar buttons, x/y joystick, and whammy bar. We'll use the Wii Nunchuck breakout adapter with the QT Py microcontroller to convert those I2C messages into MIDI notes, CC messages, and pitch bend from the whammy bar.

With the QT Py plugged into your computer or iOS device via the USB-Lightning adapter (http://adafru.it/3940), you'll be able to play any software synth with your Guitar Hero controller!

Synthesizers

Nearly any software synth (or DAW (digital audio workstation) you find will allow you to use USB MIDI to control the notes, pitch bend, and CC input. Here are some good ones to try:

Linux / Windows / mac os

free open source

- Helm (https://adafru.it/C-a)
- VCV Rack (https://adafru.it/C-b)
- Pure Data (https://adafru.it/C-c)
- Vital (https://adafru.it/P-f)
- Ardour (https://adafru.it/C-d)

Chrome Web Browser

- Viktor NV-1 https://nicroto.github.io/viktor/ (https://adafru.it/C-4)
- Juno-106.js http://juno-106.js.org/ (https://adafru.it/C-5)

iOS

free open source

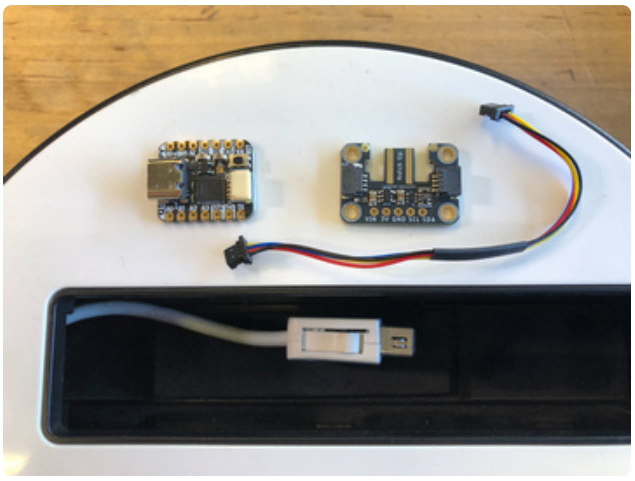
- AudioKit Synth One (https://adafru.it/C-6) (iPad only)

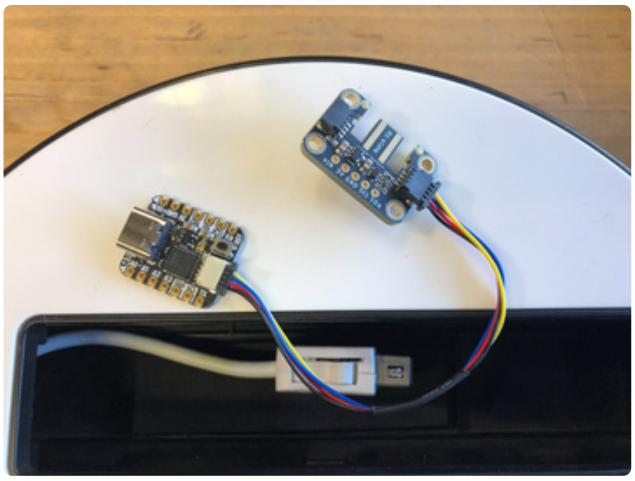
free

- GarageBand (https://adafru.it/C-7)
- DRC Polyphonic Synth by Imaginado (https://adafru.it/C-8) (iPhone and iPad)

Hook Up the Controller

Thanks to the Wii Accessory plug on the controller, we can make all of our connections without disassembling the guitar. The plug hooks up with the Wii Nunchuck breakout adapter very simply!

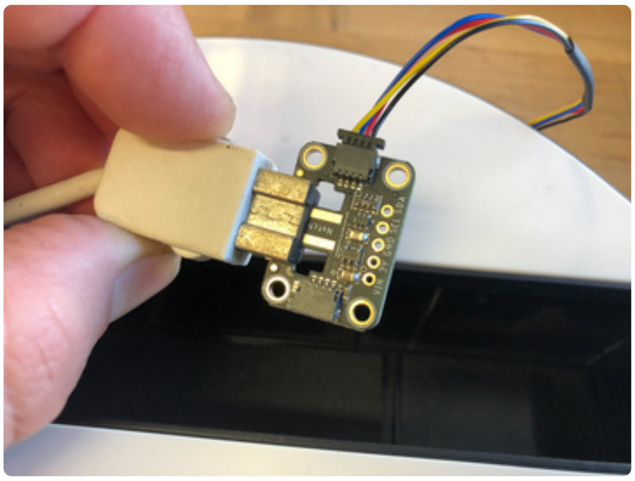


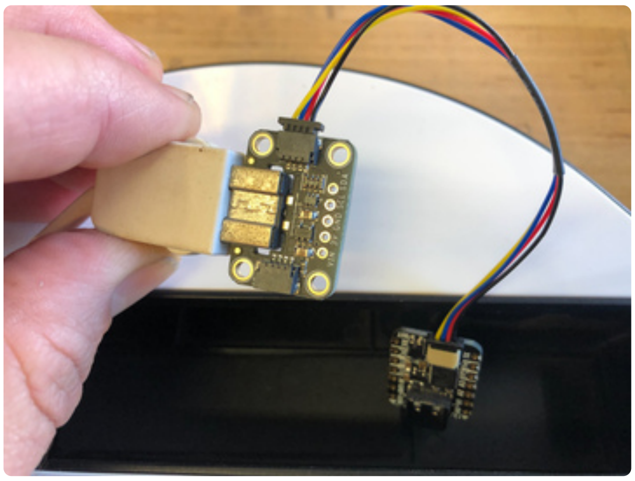


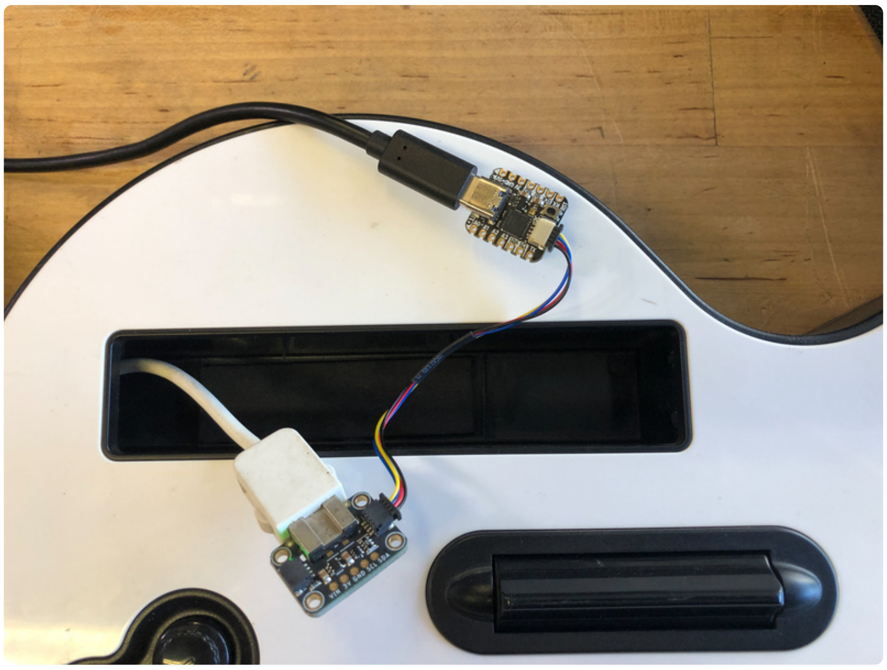
Plug Things In to Other Things

Here's all we need to connect:

Nunchuck adapter to QT Py with a STEMMA QT cable
Wii controller Accessory Plug to Nunchuck adapter (be sure to follow the "Notch Up" directions
QT Py to computer with USB cable







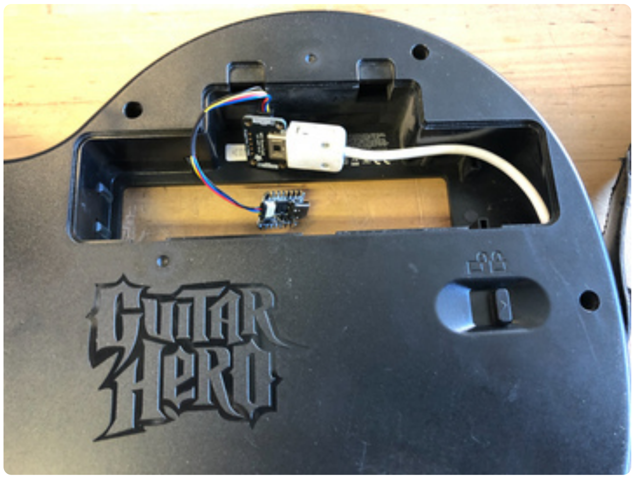
If connecting to an iOS device, you'll also need a Lightning to USB OTG adapter cable, such as adafru.it/3940

If you want to keep it super simple, this is all you need to do. Just push the parts into the empty Wiimote slot and you're done!

Fancy Mode

If you want to get a bit fancy with it, you can create a small notch in the access panel on the back of the guitar.

Be careful with cutting instruments and wear eye protection.

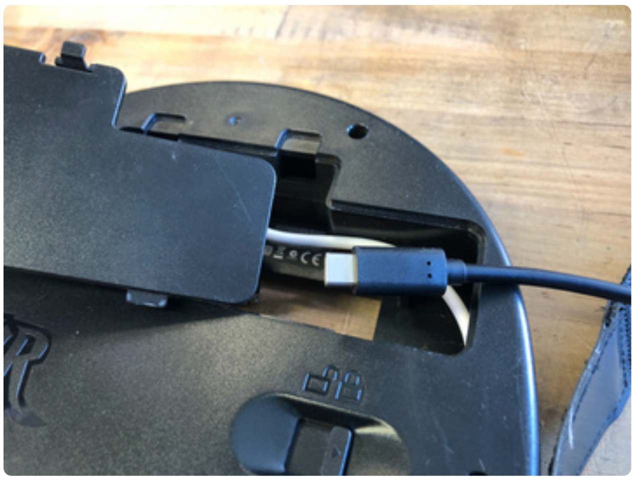


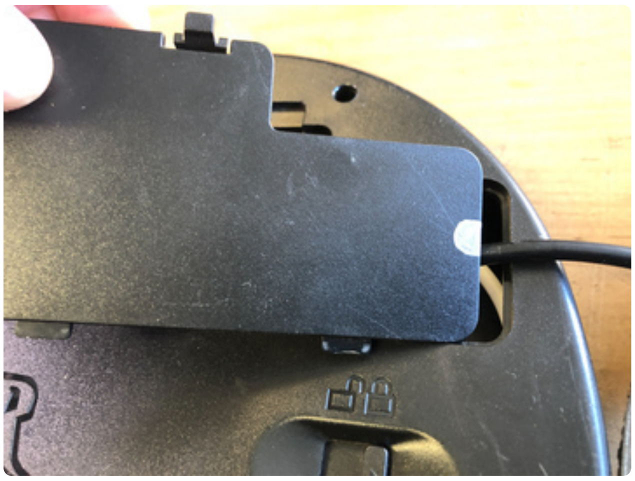
Open the access panel, then mark the door where you want to file out a notch for the USB cable.

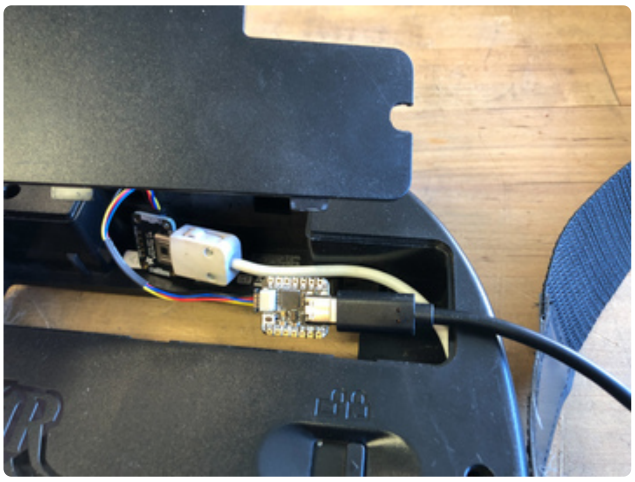
Use a round file (or utility knife, small saw, and sandpaper, etc.) to create the notch.

Plug the USB cable into the QT Py and then close the access panel.

You're all set!







Code the Guitar Hero MIDI Controller

In order to make it simple to get up and running with your Guitar Hero MIDI Controller **with no programming required**, we've created a drag-and-drop firmware you can use.

If you're just getting started with your QT Py, there is lots more info here in the main Learn Guide! (https://adafru.it/P-A)

Download

First, download the firmware file linked in the button below and save it to your computer hard drive somewhere you'll be able to find it, such as your **Downloads** folder.

Guitar_Hero_MIDI_QTPy.UF2

https://adafru.it/P-B

This program will replace CircuitPython. To get back to your CircuitPython projects, reinstall it by following https://learn.adafruit.com/adafruit-qt-py/circuitpython

Install the Firmware

Plug your QT Py into your computer with a good quality, data capable **USB** cable. Life is too short to go through the pain of accidentally using a **power-only** USB cable, so please round up any you own, cut them in half, travel to a distant land, bury them, and dance on their grave.

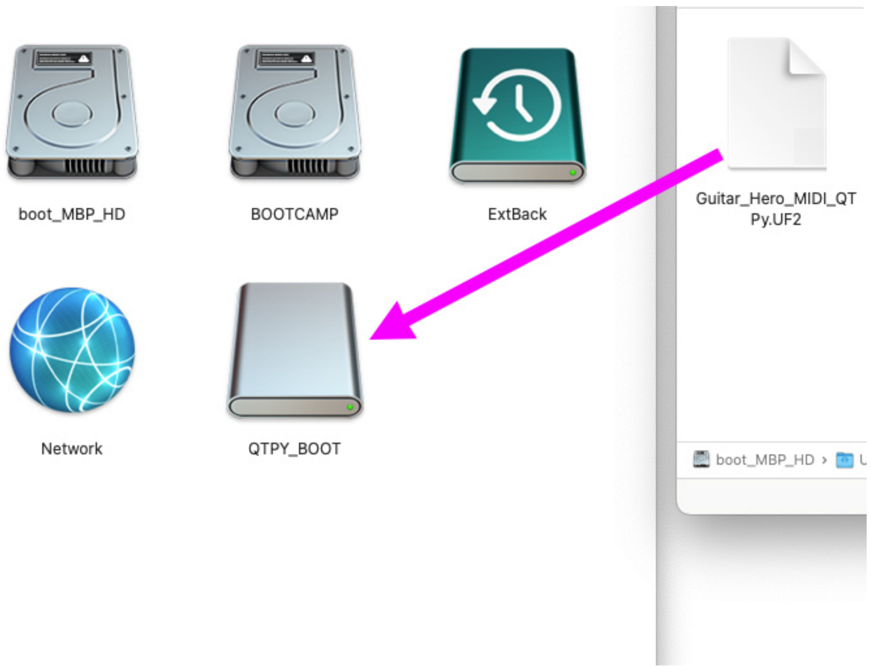
Bootloader Mode

Now, we'll put the QT Py into "bootloader" mode. In this mode it will appear as a USB drive on your computer and will be ready to receive a new .uf2 firmware file. Double-click the reset button on the top side of the board, next to the STEMMA QT connector port.

Once you have double-clicked the reset button, the indicator LED will turn green. You'll notice a new USB drive appear on your computer named **QTPY_BOOT**, this is the bootloader USB storage built right into the QT Py. It is now ready to receive the firmware file.

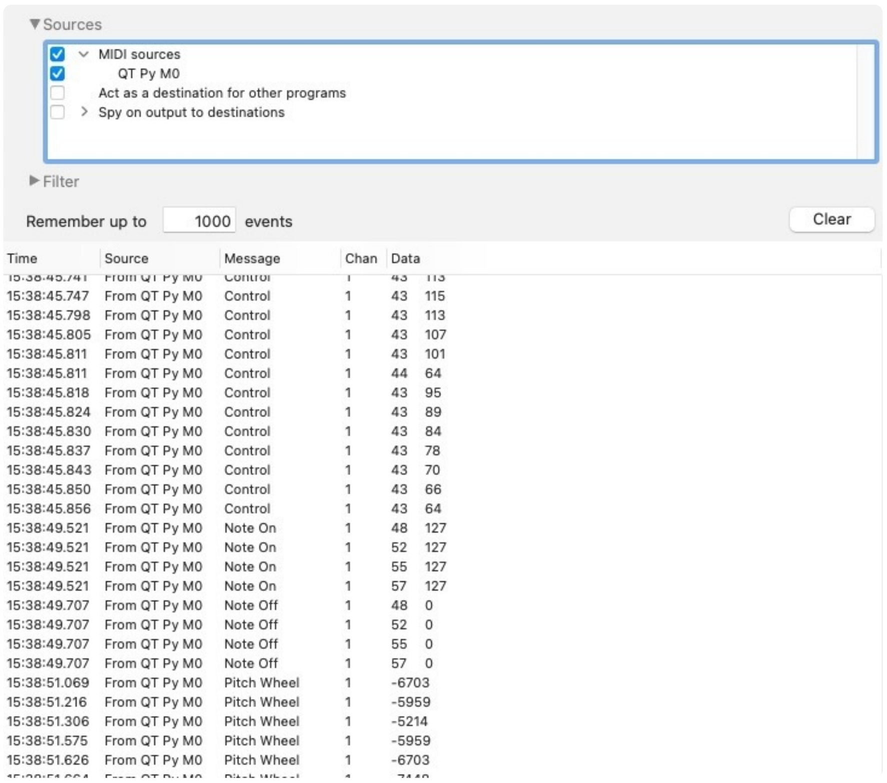
Drag and Drop

Now, drag the **Guitar_Hero_MIDI_QTPy.UF2** file onto the **QTPY_BOOT** drive. The file will copy over in a few seconds and then the QT Py will automatically restart itself (you'll see the **QTPY_BOOT** drive disappear, don't worry, this is normal!). The status LED will turn off, indicating regular operation mode.



You've updated the firmware and you're ready to play!

If you ever need to manually switch out of bootloader mode, simply press the reset button one time.



- [✓] MIDI sources
  - [✓] QT Py M0
- [ ] Act as a destination for other programs
- [ ] Spy on output to destinations

Remember up to 1000 events

```
Time          Source          Message      Chan  Data
15:38:45.741  From QT Py M0   Control      1     43   113
15:38:45.747  From QT Py M0   Control      1     43   115
15:38:45.798  From QT Py M0   Control      1     43   113
15:38:45.805  From QT Py M0   Control      1     43   107
15:38:45.811  From QT Py M0   Control      1     43   101
15:38:45.811  From QT Py M0   Control      1     44   64
15:38:45.818  From QT Py M0   Control      1     43   95
15:38:45.824  From QT Py M0   Control      1     43   89
15:38:45.830  From QT Py M0   Control      1     43   84
15:38:45.837  From QT Py M0   Control      1     43   78
15:38:45.843  From QT Py M0   Control      1     43   70
15:38:45.850  From QT Py M0   Control      1     43   66
15:38:45.856  From QT Py M0   Control      1     43   64
15:38:49.521  From QT Py M0   Note On      1     48   127
15:38:49.521  From QT Py M0   Note On      1     52   127
15:38:49.521  From QT Py M0   Note On      1     55   127
15:38:49.521  From QT Py M0   Note On      1     57   127
15:38:49.707  From QT Py M0   Note Off     1     48   0
15:38:49.707  From QT Py M0   Note Off     1     52   0
15:38:49.707  From QT Py M0   Note Off     1     55   0
15:38:49.707  From QT Py M0   Note Off     1     57   0
15:38:51.069  From QT Py M0   Pitch Wheel  1     -6703
15:38:51.216  From QT Py M0   Pitch Wheel  1     -5959
15:38:51.306  From QT Py M0   Pitch Wheel  1     -5214
15:38:51.575  From QT Py M0   Pitch Wheel  1     -5959
15:38:51.626  From QT Py M0   Pitch Wheel  1     -6703
```

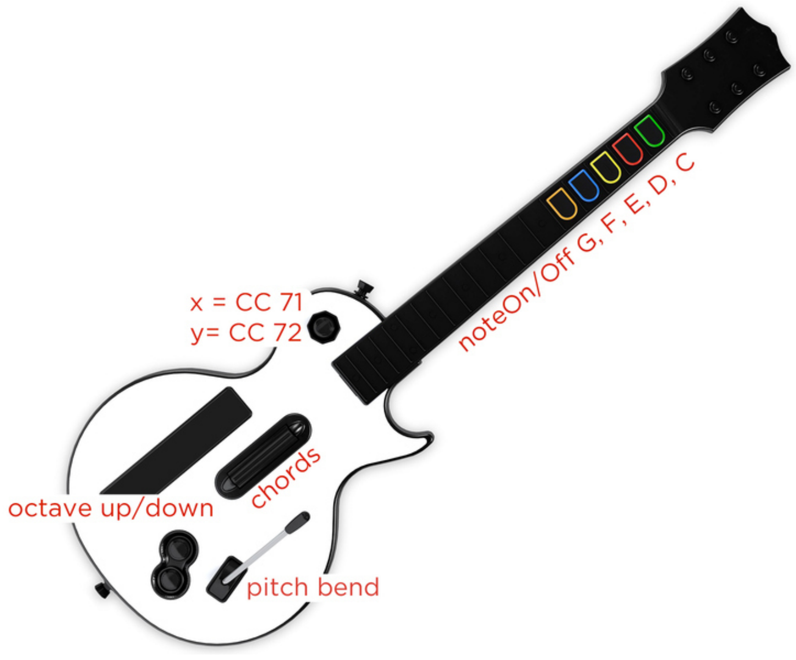
Test MIDI Output

You can test to see that your QT Py is outputting MIDI signals properly by using a MIDI utility. These will report all activity coming from any MIDI controller plugged into your system. Here are some to try:

- MIDI Monitor (https://adafru.it/rBH) for mac os
- MIDI-OX (https://adafru.it/nkF) for Windows
- MIDI Wrench (https://adafru.it/C-1) for iOS
- MIDI Scope (https://adafru.it/C-2) for Android
- KMidimon (https://adafru.it/rBI) for Linux
- Web MIDI Monitor (https://adafru.it/C-3) for Chrome browser

Play!

Here's how the controls are mapped:



Customize It

If you'd like to modify or customize your code, you can download the Arduino code linked below and use it along with the Arduino IDE. Then, you'll flash your QT Py with the updated firmware using the Arduino IDE Upload button.

```
// SPDX-FileCopyrightText: 2021 John Park for Adafruit Industries
//
// SPDX-License-Identifier: MIT
```

```
//Wii Guitar Hero MIDI Controller
// by John Park for Adafruit Industries
#include <WiiChuck.h>
#include <Adafruit_TinyUSB.h>
#include <MIDI.h>

Accessory guitar;
Adafruit_USBD_MIDI usb_midi;
MIDI_CREATE_INSTANCE(Adafruit_USBD_MIDI, usb_midi, MIDI);

int MIDI_OUT_CH = 1;  // pick your midi output channel here
bool DEBUG = 0;  // set to 1 to use serial monitor to check out controller values

int whammyBar;
int joyX;
int joyY;
int minusButton;  // drop an octave w each press
int plusButton;  // up and octave w each press
int strumDown;
int strumUp;
int fretButtons[5];

bool minusButtonOn = 0;
bool plusButtonOn = 0;
bool strumDownOn = 0;
bool strumUpOn = 0;
bool fretButtonOn[] = {0, 0, 0, 0, 0};

int octave = 12;  // note offset value, used to change octaves

int strumDownChord[] = {36, 40, 43, 45};  //all note values will be offset by
octave value
int strumUpChord[] = {36, 41, 43, 45} ;
int fretNotes[] = {24, 26, 28, 29, 31};

int lastWhammy = 16;  // Use the resting state value of your whammy bar
int whammyPitchVal = 8192;  // resting position of pitchwheel

int lastJoyX = 223;  // resting value of joyX
int joyXCCNum = 71; // VCF or whatever you assign in synth software
int joyXCCVal = 0;
int lastJoyY = 224;  // resting value of joyY
int joyYCCNum = 72; // VCA
int joyYCCVal = 0;

void setup() {
  Serial.begin(115200);
  MIDI.begin(MIDI_CHANNEL_OMNI);
  guitar.begin();
  guitar.type = GuitarHeroController;
}

void loop() {
  guitar.readData();    // Read inputs and update maps
  fretButtons[0] = guitar.values[10];  // green
  fretButtons[1] = guitar.values[11];  // red
  fretButtons[2] = guitar.values[12];  // yellow
  fretButtons[3] = guitar.values[13];  //blue
  fretButtons[4] = guitar.values[14]; // orange
  minusButton = guitar.values[6];
  plusButton = guitar.values[16];
  strumDown = guitar.values[7];
  strumUp = guitar.values[7];
  whammyBar = guitar.values[0];
  joyX = guitar.values[20];
  joyY = guitar.values[21];

  for(int i=0;i<5;i++){
    if(fretButtons[i]==255 && fretButtonOn[i]==0){
      MIDI.sendNoteOn(fretNotes[i]+octave, 127, MIDI_OUT_CH);
      fretButtonOn[i] = 1;}
    if(fretButtons[i]==0 && fretButtonOn[i]==1){
      MIDI.sendNoteOff(fretNotes[i]+octave, 0, MIDI_OUT_CH);
      fretButtonOn[i] = 0;}
    }

  if(whammyBar!=lastWhammy){
    whammyPitchVal=map(whammyBar, 15, 26, 8192, 16383); // remap to pitch value 
range, two semitones here
    MIDI.sendPitchBend(whammyPitchVal, MIDI_OUT_CH);
    lastWhammy=whammyBar;
  }
  if(joyX!=lastJoyX){
    joyXCCVal=map(joyX, 190, 255, 0, 127); // remap to bigger range
    MIDI.sendControlChange(joyXCCNum, joyXCCVal, MIDI_OUT_CH);
    lastJoyX=joyX;
  }
  if(joyY!=lastJoyY){
    joyYCCVal=map(joyY, 190, 255, 0, 127); // remap to bigger range
    MIDI.sendControlChange(joyYCCNum, joyYCCVal, MIDI_OUT_CH);
    lastJoyY=joyY;
  }

  if(minusButton==0 && minusButtonOn==0){
    octave = constrain((octave - 12), 0, 108);
    minusButtonOn = 1;}
  if(minusButton==128 && minusButtonOn==1){
    minusButtonOn = 0;}

  if(plusButton==255 && plusButtonOn==0){
    octave = constrain((octave + 12), 0, 108);
    plusButtonOn = 1;}
  if(plusButton==0 && plusButtonOn==1){
    plusButtonOn = 0;}

  if(strumDown==0 && strumDownOn==0){
    for(int c=0; c<4; c++){
      MIDI.sendNoteOn(strumDownChord[c]+octave, 127, MIDI_OUT_CH);}
    strumDownOn = 1;}
  if(strumDown==128 && strumDownOn==1){
    for(int c=0; c<4; c++){
      MIDI.sendNoteOff(strumDownChord[c]+octave, 0, MIDI_OUT_CH);}
    strumDownOn = 0;}

  if(strumUp==255 && strumUpOn==0){
    for(int c=0; c<4; c++){
      MIDI.sendNoteOn(strumUpChord[c]+octave, 127, MIDI_OUT_CH);}
    strumUpOn = 1;}
  if(strumUp==128 && strumUpOn==1){
    for(int c=0; c<4; c++){
      MIDI.sendNoteOff(strumUpChord[c]+octave, 0, MIDI_OUT_CH);}
    strumUpOn = 0;}

  delay(5);

  if(DEBUG){
    Serial.println("-------------------------------------------");
    guitar.printInputs();
    for (int i = 0; i < WII_VALUES_ARRAY_SIZE+3; i++) {
      Serial.println(
          "Controller Val " + String(i) + " = "
              + String((uint8_t) guitar.values[i]));
    }
    delay(50);  // keeps the terminal from flooding
  }
}
```