Table of Contents

Overview 9
- Parts
- Parts from Adafruit
- Wires & Connections
- Non-Adafruit Parts
- Author Credits

Circuit Diagram 15
- Adafruit Library for Fritzing
- Wired Connections
- Going Further
- Wired Connections
- Extra Parts for the Neck NeoPixel Effects

CircuitPython on the Grand Central M4 Express 20
- CircuitPython Setup for the Grand Central M4 Express
- Grand Central M4 Express Libraries
- Download the Code from GitHub

Code Walkthrough 27
- Libraries
- MIDI Setup
- LIS3DH Accelerometer Setup
- Analog Inputs
- Mode Switches
- Strumming Switches
- Guitar Neck Buttons
- Octave Selector
- Digital Input States
- Analog Input States
- Note Input Arrays
- Reading Analog Inputs
- Beginning REPL Message
- The Loop
- Mapping Analog Values to MIDI Parameter Values
- Sending MIDI Data with Analog Data
- How the MIDI Messages Are Being Sent
- Octave Selection
- Strum Mode
- Tapping Mode (No Strumming Required)

3D Printing 39
- 3D Printed Parts
- Body Parts
- Strumming Parts
- Whammy Bar Parts
- Design Source Files
- Slicing Parts
- Setup Parts
- Large Surface Area
Neck Head Assembly
- Hardware for Neck and Head
- Install Screws
- Secure Hardware
- Assembled Neck and Head

Head Cover Assembly
- Black LED Acrylic
- Glue Diffuser to Cover
- Secured Diffuser
- Installed Diffuser Cover

Body Assembly
- Guitar Body – Bottom Half
- Install Machine Screws
- Secure Machine Screws
- Guitar Body – Top Half
- Install and Secure Screws
- Installing Guitar Body and Neck
- Install Neck to Top Half Guitar Body
- Install Hardware into Neck
- Secure and Fasten Hardware
- Assembled Guitar Body and Neck

Strum Assembly
- Strum Hardware Screws
- Install Hinges
- Secure Hex Nuts
- Install Joints to Pick
- Hot Glue Joints to Pick
- Strum Pick with Joints
- Install Pick to Base
- Installed Pick
- Pick Assembly
- Parts for Micro Switches
- Secure Switches to Plate
- Installed Switches
- Wires for Micro Switches
- Tinning Wires
- Wire JST Connectors
- 2x 2-pin JST Cables
- Micro Switch Ground Wire
- Solder Micro Switch Pin
- Wired Micro Switches
- Strum Installation
- Install Strum Mounting Plate
- Install Micro Switches
- Secure Strum Parts
- Install Pick to Hinges
- Test Strum Pick
- Assembled Strum

Whammy Assembly
- Whammy Wheel Parts
- Glue Joint to Wheel
- Glue Potentiometer Holder to Wheel
• Glue Spring Plate
• Glue Spring Plate to Potentiometer Plate
• Spring Assembly
• Install Extension Spring
• Install Ext. Spring to Holders
• Install Spring Holder Set to Plate
• Installed Spring Holder Assembly
• Whammy Wires
• JST Wire
• JST Cables
• Solder Potentiometer
• Test JST Connection
• Center Potentiometer
• Install Potentiometer
• Spring Tension
• Whammy Body Parts
• Install Whammy Idler
• Install Wheel to Body
• Whammy Assembled

**Wiring 8-Way Switch**

• Wires for 8-way Switch
• Tin Ribbon Cable
• Ribbon Cable Wires
• Tin JST Cable Wires
• Tinned Wires
• Connect 8-way Pins
• Connect 8-way Ground
• Attaching JST connectors to ribbon cable
• Solder JST connectors
• Wired 8-way switch

**Wiring Pots**

• Wires for Potentiometers
• Tin Ribbon Cable Wires
• Solder JST Connectors
• Wired 3-pin JST Connectors
• Wire Potentiometers
• Wired Potentiometers

**Wiring Toggle Switches**

• Toggle Switch Wires
• Tin Toggle Switch Wires
• 2-pin JST Connectors
• Solder 2-pin JST wires
• Wired 2-pin JST
• Solder Toggle Switch Pins
• Wired Toggle Switches
• Tin 2-pin JST cables
• Wired Toggle Switch Cables

**Wiring Perma-Proto**

• Perma-Proto Power Rails
• 2-wire ribbon cable
• Solder Wires to Pins
• Wired Perma-Proto Power Rails
Wiring The LIS3DH

- Wires for LIS3DH
- Tin Ribbon Cable Wires
- Tin 4-pin JST Cable
- Solder Ribbon Wires
- Solder LIS3DH Wires
- Wired LIS3DH

Wiring MX Switches

- Wires for MX Switches
- Tin 4-wire JST cables
- Tin 2-wire JST Cable
- Tinned JST Cables
- Plate for MX Switches
- Installing MX Switches
- Install MX Switches
- Installed MX Switches (Bottom View)
- Installed MX Switches (Top View)
- Wires from Shared Ground
- Soldering Ground
- Shared Ground
- Ground Wired
- Check Ground Wiring
- Tin Longer Ground Wire
- Solder JST Connector
- Solder Last Ground MX Switch
- Wired JST Ground Cable
- Wiring 4-pin JST Connectors
- Tinning 4-pin JST Cables
- Wiring First Key
- 4 Keys Wired
- 8 Keys Wired
- 12 Keys Wired
- Wired MX Switches
- Group Wiring

Wiring NeoPixel Feather

- Feather and NeoPixel Wires
- Count, Cut and Unsheathe
- Tin 3-pin JST Cable
- Tin 3-wire ribbon cable
- Strip 3-pin JST cable solder
- Feather 3-wire ribbon cable solder
- Solder Slide Switch Ribbon Cable
- Solder Slide Switch to Feather
- Test Circuit

Strip Assembly

- NeoPixel Strip for Head
- Install Strip to Head
- Install Feather Screws
- Installing Feather to Neck
- Secure Feather to Neck
- Install Feather Slide Switch
- Connect Strip to Feather
- Install Strip JST Wiring
Install the LIS3DH
- Hardware for LIS3DH
- Install LIS3DH Screws
- Install LIS3DH to Neck
- Secure LIS3DH to Neck
- Install Battery

Wiring the M4
- Grand Central M4 Wiring
- Install M3 Standoffs
- Label JST Cables
- Wire Prema-Proto
- Wire Potentiometers
- Pots Ground and Voltage
- Wire 8-Way Rotary Switch
- Wire 8-Way Rotary Switch to Ground
- Wire the LIS3DH
- Wire LIS3DH Ground and Voltage
- Wire Micro Switches (Strum)
- Wire MX Switches
- M4 Wiring Mounted Components
- M4 Wire Whammy Pot
- Wire Whammy Ground and Voltage
- Wire Toggle Switches
- Wire Toggle Switches Ground and Voltage
- M4 Wiring Check Point
- Unplug Components

Panel Mount Components
- Panel Mounting Components
- Install 8-way switch and toggle switches
- Install 2x Potentiometers
- Panel Mounted Components

Assemble Grand Central
- Install Grand Central M4
- Secure Grand Central M4
- Connect Micro Switches for Strum
- Connect Whammy
- Installing micro USB Extension
- Secure USB Cable
- Installed USB Extension

Final Assembly
- Install the Neck Cover
- Snap Fit Neck Cover
- Snap Fit Head Cover
- Test NeoPixel Strip
- Make Connections
- Install Keycaps
- Installed Keycaps
- Check Connections
- Closing the Guitar Body
- Closing the Guitar (Continued)
- Install the Whammy Bar
- Install Knob for 8-way switch
• Secure Knob for 8-way switch
• Installed Knob for 8-way switch
• Install Knobs for Potentiometers
• Secured Knobs for Potentiometers
• Final Build

Let's Jam!
Overview

Inspired by the infamous Guitar Hero controller(), this project shapes a MIDI controller into a 3D printed guitar. Featuring 12x Cherry MX switches() in the neck, this gives you a tactile "hands-on" feeling that is satisfying. The whammy bar lets you pitch bend, making it feel like an electric guitar. It also features a strumming mode so you can power through chords and notes, giving you a playing style unlike typical MIDI controllers. Using an accelerometer, you can move the guitar to adjust modulation giving you an expressive playing style.

This project is coded in CircuitPython and uses the USB MIDI library. It features velocity, modulation, pitch bending and notes on/off. It features 8 different sets of notes for playing different octaves or sets of notes to craft your own chords. Two switches allow you to enable different features such the strum mode, trigger mode and accelerometer mode. Two potentiometers lets you adjust velocity and modulation.
The parts are 3D printed without any support material. The top and bottom half of the guitar body snap fit together allowing you easy access to open and close. Parts are designed to print on most 3D printers with a build volume of 250x210x200mm. The parts and components are secured together with M3 hardware. The design files are open source, freely available to download. The CAD files feature a parametric assembly to easily add/mod to create custom controller.

MX switches are available in different types and styles. The color of the key stem is used to classify its characteristics. We choose to use speed silvers for their quiet presses and low actuation point.

The head houses a strip of NeoPixels to diffuse a piece of black LED acrylic. The LEDs illuminate the panel making a neat lighting effect.

Parts

Adafruit Grand Central M4 Express featuring SAMD51
Are you ready? Really ready? Cause here comes the Adafruit Grand Central featuring the Microchip ATSAMD51. This dev board is so big, it’s not...
https://www.adafruit.com/product/4084

©Adafruit Industries
Adafruit LIS3DH Triple-Axis Accelerometer (+-2g/4g/8g/16g)
The LIS3DH is a very popular low power triple-axis accelerometer. It's low-cost, but has just about every 'extra' you'd want in...
https://www.adafruit.com/product/2809

Mini 8-Way Rotary Selector Switch - SP8T
When you need a lot of options, you need a Mini 8-Way Rotary Selector Switch. This is basically a single-pole to 8-throw switch. As you rotate the knob...
https://www.adafruit.com/product/2925

Potentiometer with Built In Knob - 10K ohm
Oh say can you seeBy the knob's early light...Sorry - we thought that was clever. And while it wasn't really, this potentiometer definitely...
https://www.adafruit.com/product/4133

Panel Mount 10K potentiometer (Breadboard Friendly)
This potentiometer is a two-in-one, good in a breadboard or with a panel. It's a fairly standard linear taper 10K ohm potentiometer, with a grippy shaft. It's smooth and easy...
https://www.adafruit.com/product/562
Micro Switch w/Lever - 2 Terminal
Micro-switches are often found in arcade buttons and joysticks but they’re also really handy in any kind of mechatronics project or when you need a basic sensor. They are always...
https://www.adafruit.com/product/818

Adafruit Perma-Proto Quarter-sized Breadboard PCB - Single
Customers have asked us to carry basic perf-board, but we never liked the look of most basic perf: it’s always crummy quality, with pads that flake off and no labeling. Then we...
https://www.adafruit.com/product/1608

Panel Mount Extension USB Cable - Micro B Male to Micro B Female
Check out this handy MicroUSB extension cable, which will make it easy for you to enclose a device that has a B type (micro USB host) port. Great if you need to extend the USB...
https://www.adafruit.com/product/3258

Mini Panel Mount SPDT Toggle Switch
This or that, one or the other, perhaps or perhaps not! So hard to make decisions these days without feeling like you're just going back and forth constantly. Deciding whether or...
https://www.adafruit.com/product/3221
Parts Galore!

There's a hand full of parts used in this project! If you have parts on-hand, be sure to scroll through the guide and look for anything that you may have on hand. Most of the panel mounted components feature standardized diameters. The mounting holes on the Grand Central M4 board are similar to the Arduino Mega.

Parts from Adafruit
1x Grand Central M4 ()
1x LIS3DH Accelerometer breakout ()
2x 10k Potentiometer ()
2x Micro Switch with Lever ()
1x Mini potentiometer ()
1x 8-way rotary selector switch ()
1x Panel Mount USB micro cable ()
2x Mini panel mount switch ()
1x 2m USB micro cable ()
1x Quarter-Size Perma-Proto ()

Wires & Connections

Because of the amount of components, several dozens of connections will need to be made. By using pre-crimped wires with JST PH connectors, you'll make the assembly more manageable.

4 x 2-pin JST Female
JST PH cable

4 x 2-pin JST Male
JST PH Cable

Pink and Purple Braided USB A to Micro B Cable - 2 meter long
This cable is super-fashionable with a woven pink and purple Blinka-like pattern!
First let's talk about the cover and over-molding. We got these in custom colors,...
https://www.adafruit.com/product/4148
### JST PH Cable

- **3 x 3-pin JST Female**
  - JST PH Cable
  - [https://www.adafruit.com/product/3893](https://www.adafruit.com/product/3893)
- **3 x 3-pin JST Male**
  - JST PH Cable
  - [https://www.adafruit.com/product/4046](https://www.adafruit.com/product/4046)
- **4 x 4-pin JST Male**
  - JST PH Cable
  - [https://www.adafruit.com/product/4045](https://www.adafruit.com/product/4045)
- **4 x 4-pin JST Female**
  - JST PH Cable
  - [https://www.adafruit.com/product/3950](https://www.adafruit.com/product/3950)
- **1 x JST PH Kit**
  - Assortment of JST PH Connectors
  - [https://www.adafruit.com/product/4422](https://www.adafruit.com/product/4422)
- **1 x 10-wire silicone ribbon cable**
  - 28AWG silicone cover stranded wire
  - [https://www.adafruit.com/product/3890](https://www.adafruit.com/product/3890)

### Non-Adafruit Parts

These parts are used in the project but not sold in the Adafruit Online Shop.

- **12 x MX Switches**
  - Cherry MX Speed Silver Keyswitch (10 Pack)
  - [https://www.amazon.com/gp/product/B07B79J3ZK](https://www.amazon.com/gp/product/B07B79J3ZK)
- **1 x Key Caps for MX Switches**
  - Available in Black and different colors
  - [https://www.amazon.com/gp/product/B06VWBV3Q7/](https://www.amazon.com/gp/product/B06VWBV3Q7/)
- **1 x Whammy Bar**
  - 6mm Bar Arm for Electric Guitar
  - [https://www.amazon.com/gp/product/B00CE1WO48/](https://www.amazon.com/gp/product/B00CE1WO48/)
- **2 x Knobs for Potentiometer**
  - GTE Knob - Small
  - [https://www.sparkfun.com/products/11951](https://www.sparkfun.com/products/11951)
- **1 x Knobs for Rotary Selector Switch**
  - Black Knob - 15x19mm
  - [https://www.sparkfun.com/products/9998](https://www.sparkfun.com/products/9998)
- **1 x Extension Springs**
  - Extension Spring Kit (6.35 x 22.2 mm)
  - [https://www.amazon.com/gp/product/B01KZP0WOS/](https://www.amazon.com/gp/product/B01KZP0WOS/)
- **1 x Black LED Acrylic**
  - Precut Sheet – 114mm x 144mm
Hardware List

Guitar Body – Bottom Half

16x M3 x 6mm pan head screws  
4x M3 x 8mm pan head screws  
4x M3 x 10mm pan head screws  
4x M3 x 16mm pan head screws  
20x M3 hex nuts  
4x M3 x 20mm standoffs  
1x Extension Spring (6.35 x 22.2 mm)

Guitar Body – Top Half

7x M3 x 6mm pan head screws  
7x M3 hex nuts

Guitar Neck

12x M2.5 hex nuts  
6x M2.5 x 8mm pan head screws  
2x M3 x 6mm pan head screws  
2x M3 hex nuts

1 x M2.5 Standoff Kit  
Black Nylon screws standoffs and hex nuts  
https://www.adafruit.com/product/3299

1 x M3 Standoff Kit  
Brass screws standoffs and hex nuts  
https://www.amazon.com/gp/product/B06XQ33Y9X/

Author Credits

CAD by Noe Ruiz () and Code by Liz Clark ()

Circuit Diagram

The diagram below provides a visual reference for wiring of the components. This diagram was created using the software package Fritzing ().
Adafruit Library for Fritzing

Use Adafruit's Fritzing parts library to create circuit diagrams for your projects. Download the library or just grab individual parts. Get the library and parts from GitHub - Adafruit Fritzing Parts.

Wired Connections

Make the following connections for the LISD3DH accelerometer.

- Grand M4 SDA to LIS3DH SDA
- Grand M4 SCL to LIS3DH SCL
- Grand M4 GND to LIS3DH GND
- Grand M4 3V to LIS3DH VIN

Pitch Bend/Whammy Potentiometer

- Grand M4 PIN1 to Pot PIN1
- Grand M4 GND to Pot PIN2
- Grand M4 3V to Pot PIN3

Modulation Potentiometer

- Grand M4 PIN2 to Pot PIN1
• Grand M4 GND to Pot PIN2
• Grand M4 3V to Pot PIN3

### Velocity Potentiometer

• Grand M4 PIN3 to Pot PIN1
• Grand M4 GND to Pot PIN2
• Grand M4 3V to Pot PIN3

### Strum Mode Switch

• Grand M4 PIN52 to Switch PIN1
• Grand M4 GND to Switch GND

### Accelerometer Mode Switch

• Grand M4 PIN53 to Switch PIN1
• Grand M4 GND to Switch GND

### 8-Octave Selector Switch

• Grand M4 PIN24-31 to 8-Way Switch PIN1-8
• Grand M4 GND to 8-Way Switch GND

### Strum Switches

• Grand M4 PIN23 to Switch 1 NO
• Grand M4 PIN22 to Switch 2 NO
• Grand M4 GND to Switch 2 GND
• Grand M4 GND to Switch 1 GND

### MX Switches

• Grand M4 PIN2-12, PIN14 to MX Switch 1-12 NO
• Grand M4 GND to MX Switch 1-12 GND
Going Further

The head of the guitar can house a strip of NeoPixels for additional lighting effects. Using a slide switch, a 500mAh battery and Adafruit Feather M4, you can create LED animations with CircuitPython. This is a separate system apart from the Grand Central M4.

Wired Connections

Feather M4, NeoPixel Strip and Switch

- Feather EN to Switch PIN1
- Feather GND to Switch PIN2
- Feather PIN5 to NeoPixel DIN
- Feather GND to NeoPixel GND
- Feather 3V to NeoPixel 5V
Extra Parts for the Neck NeoPixel Effects

Adafruit Feather M4 Express - Featuring ATSAMD51
It's what you've been waiting for, the Feather M4 Express featuring ATSAMD51. This Feather is fast like a swift, smart like an owl, strong like a ox-bird (it's half ox,...
https://www.adafruit.com/product/3857

Breadboard-friendly SPDT Slide Switch
These nice switches are perfect for use with breadboard and perfboard projects. They have 0.1" spacing and snap in nicely into a solderless breadboard. They're easy to switch...
https://www.adafruit.com/product/805

Lithium Ion Polymer Battery - 3.7v 500mAh
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/1578
Adafruit Mini Skinny NeoPixel Digital RGB LED Strip - 60 LED/m
So thin. So mini. So teeeeeny-tiny. It's the 'skinny' version of our classic NeoPixel strips! These NeoPixel strips have 60 digitally-addressable pixel Mini LEDs per...
https://www.adafruit.com/product/2964

CircuitPython on the Grand Central M4 Express

Take your Grand Central M4 Express board and plug it into your computer via a known good data + power USB cable. Your operating system will show a drive named CIRCUITPY when a board is plugged in. If you get a drive named GCM4BOOT, you'll likely need to install CircuitPython. This is easy, see the steps below to do this, get library files, and copy the code to your board.

CircuitPython Setup for the Grand Central M4 Express

First, to make sure you're running the most recent version of CircuitPython for the Grand Central M4 Express.
Grand Central M4 Express Libraries

You'll need a few CircuitPython libraries in the lib folder on the Grand Central M4 Express CIRCUITPY drive for the code to work. Head to https://circuitpython.org/libraries to download the latest library bundle matching the major version of CircuitPython now on your board (5 for CircuitPython 5.x, etc.). The procedure is available in the Grand Central M4 Express guide.

Once you've downloaded the libraries bundle, add these libraries to the lib folder on the Grand Central:

- adafruit_bus_device
- adafruit_lis3dh.mpy
- adafruit_midi
- simpleio.mpy

Your Grand Central CIRCUITPY drive should look like this after you load the code below:

![CIRCUITPY Drive Content]

Download the Code from GitHub

Once your Grand Central M4 Express is all setup with CircuitPython and the necessary libraries, you can click on the Download: Project Zip link above the code to get the code file.

```python
# SPDX-FileCopyrightText: 2020 Liz Clark for Adafruit Industries
# # SPDX-License-Identifier: MIT

import time
import board
import simpleio
```
import busio
import adafruit_lis3dh
import digitalio
from digitalio import DigitalInOut, Direction, Pull
from analogio import AnalogIn
import usb_midi
import adafruit_midi
from adafruit_midi.note_on          import NoteOn
from adafruit_midi.note_off         import NoteOff
from adafruit_midi.control_change   import ControlChange
from adafruit_midi.pitch_bend       import PitchBend

# imports MIDI
midi = adafruit_midi.MIDI(midi_out=usb_midi.ports[1], out_channel=0)

# setup for LIS3DH accelerometer
i2c = busio.I2C(board.SCL, board.SDA)
lis3dh = adafruit_lis3dh.LIS3DH_I2C(i2c)
lis3dh.range = adafruit_lis3dh.RANGE_2_G

# setup for 3 potentiometers
pitchbend_pot = AnalogIn(board.A1)
mod_pot = AnalogIn(board.A2)
velocity_pot = AnalogIn(board.A3)

# setup for two switches that will switch modes
mod_select = DigitalInOut(board.D52)
mod_select.direction = Direction.INPUT
mod_select.pull = Pull.UP
strum_select = DigitalInOut(board.D53)
strum_select.direction = Direction.INPUT
strum_select.pull = Pull.UP

# setup for strummer switches
strumUP = DigitalInOut(board.D22)
strumUP.direction = Direction.INPUT
strumUP.pull = Pull.UP
strumDOWN = DigitalInOut(board.D23)
strumDOWN.direction = Direction.INPUT
strumDOWN.pull = Pull.UP

# setup for cherry mx switches on neck
            board.D6, board.D7, board.D8, board.D9, board.D10, board.D11,
            board.D12]
note_buttons = []
for pin in note_pins:
    note_pin = digitalio.DigitalInOut(pin)
    note_pin.direction = digitalio.Direction.INPUT
    note_pin.pull = digitalio.Pull.UP
    note_buttons.append(note_pin)

# setup for rotary switch
                board.D30, board.D31]
octave_selector = []
for pin in oct_sel_pins:
    sel_pin = digitalio.DigitalInOut(pin)
    sel_pin.direction = digitalio.Direction.INPUT
    sel_pin.pull = digitalio.Pull.UP
    octave_selector.append(sel_pin)
# cherry mx switch states
note_e_pressed = None
note_f_pressed = None
note_fsharp_pressed = None
note_g_pressed = None
note_gsharp_pressed = None
note_a_pressed = None
note_asharp_pressed = None
note_b_pressed = None
note_c_pressed = None
note_csharp_pressed = None
note_d_pressed = None
note_dsharp_pressed = None

# state machines
strummed = None
pick = None
up_pick = None
down_pick = None

# states for analog inputs
pitchbend_val2 = 0
mod_val2 = 0
velocity_val2 = 0
acc_pos_val2 = 0
acc_neg_val2 = 0

# array for cherry mx switch states
note_states = [note_e_pressed, note_f_pressed, note_fsharp_pressed, note_g_pressed,
note_gsharp_pressed, note_a_pressed, note_asharp_pressed,
note_b_pressed,
note_c_pressed, note_csharp_pressed, note_d_pressed,
note_dsharp_pressed]

# array of MIDI note numbers
note_numbers = [21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100,
117, 118, 119, 120, 121, 120, 123, 124, 125, 126, 127]

# list of note name variables that are assigned to the note_numbers array
# this allows you to use the note names rather than numbers when assigning
# them to the cherry mx switches
(A0, Bb0, B0, C1, Db1, D1, Eb1, E1, F1, Gb1, G1, Ab1,
A1, Bb1, B1, C2, Db2, D2, Eb2, E2, F2, Gb2, G2, Ab2,
A2, Bb2, B2, C3, Db3, D3, Eb3, E3, F3, Gb3, G3, Ab3,
A3, Bb3, B3, C4, Db4, D4, Eb4, E4, F4, Gb4, G4, Ab4,
A4, Bb4, B4, C5, Db5, D5, Eb5, E5, F5, Gb5, G5, Ab5,
A5, Bb5, B5, C6, Db6, D6, Eb6, E6, F6, Gb6, G6, Ab6,
A6, Bb6, B6, C7, Db7, D7, Eb7, E7, F7, Gb7, G7, Ab7,
A7, Bb7, B7, C8, Db8, D8, Eb8, E8, F8, Gb8, G8, Ab8,
A8, Bb8, B8, C9, Db9, D9, Eb9, E9, F9, Gb9, G9) = note_numbers

# arrays for note inputs that are tied to the rotary switch
octave_8_cc = [E8, F8, Gb8, G8, Ab8, A8, Bb8, B8, C9, Db9, D9, Eb9]
octave_7_cc = [E7, F7, Gb7, G7, Ab7, A7, Bb7, B7, C8, Db8, D8, Eb8]
octave_6_cc = [E6, F6, Gb6, G6, Ab6, A6, Bb6, B6, C7, Db7, D7, Eb7]
octave_5_cc = [E5, F5, Gb5, G5, Ab5, A5, Bb5, B5, C6, Db6, D6, Eb6]
octave_4_cc = [E4, F4, Gb4, G4, Ab4, A4, Bb4, B4, C5, Db5, D5, Eb5]
octave_2_cc = [E2, F2, Gb2, G2, Ab2, A2, Bb2, B2, C3, Db3, D3, Eb3]
octave_1_cc = [E1, F1, Gb1, G1, Ab1, A1, Bb1, B1, C2, Db2, D2, Eb2]
octave_select = [octave_1_cc, octave_2_cc, octave_3_cc, octave_4_cc, octave_5_cc, octave_6_cc, octave_7_cc, octave_8_cc]

# function for reading analog inputs
def val(voltage):
    return voltage.value

# beginning script REPL printout
print("MX MIDI Guitar")
print("Default output MIDI channel:"), midi.out_channel + 1)

# loop
while True:
    # values for LIS3DH
    x, y, z = [value / adafruit_lis3dh.STANDARD_GRAVITY for value in lis3dh.acceleration]

    # mapping analog values to MIDI value ranges
    # PitchBend MIDI has a range of 0 to 16383
    # all others used here are 0 to 127
    pitchbend_val1 = round(simpleio.map_range(val(pitchbend_pot), 0, 65535, 0, 16383))
    mod_val1 = round(simpleio.map_range(val(mod_pot), 0, 65535, 0, 127))
    velocity_val1 = round(simpleio.map_range(val(velocity_pot), 0, 65535, 0, 127))
    acc_pos_val1 = round(simpleio.map_range(x, 0, 0.650, 127, 0))
    acc_neg_val1 = round(simpleio.map_range(y, -0.925, 0, 127, 0))

    # checks if modulation switch is engaged
    if not mod_select.value:
        # if it is, then get modulation MIDI data from LIS3DH
        # positive and negative values for LIS3DH depending on
        # orientation of the guitar neck
        # when the guitar is held "normally" aka horizontal
        # then the modulation value is neutral aka 0

        # compares previous LIS3DH value to current value
        if abs(acc_pos_val1 - acc_pos_val2) < 50:
            # updates previous value to hold current value
            acc_pos_val2 = acc_pos_val1
            # MIDI data has to be sent as an integer
            # this converts the LIS3DH data into an int
            accelerator_pos = int(acc_pos_val2)
            # int is stored as a CC message
            accWheel_pos = ControlChange(1, accelerator_pos)
            # CC message is sent
            midi.send(accWheel_pos)
            # delay to settle MIDI data
            time.sleep(0.001)

        # same code but for negative values
        elif abs(acc_neg_val1 - acc_neg_val2) < 50:
            acc_neg_val2 = acc_neg_val1
            accelerator_neg = int(acc_neg_val2)
            accWheel_neg = ControlChange(1, accelerator_neg)
            midi.send(accWheel_neg)
            time.sleep(0.001)

    else:
        # if it isn't then get modulation MIDI data from pot
        if abs(mod_val1 - mod_val2) > 2:
            # updates previous value to hold current value
            mod_val2 = mod_val1
            # MIDI data has to be sent as an integer
            # this converts the pot data into an int
            modulation = int(mod_val2)
            # int is stored as a CC message
modWheel = ControlChange(1, modulation)
# CC message is sent
midi.send(modWheel)
# delay to settle MIDI data
time.sleep(0.001)

# reads analog input to send MIDI data for Velocity
# compares previous velocity pot value to current value
if abs(velocity_val1 - velocity_val2) > 2:
    # updates previous value to hold current value
    velocity_val2 = velocity_val1
    # MIDI data has to be sent as an integer
    # this converts the pot data into an int
    # velocity data is sent with NoteOn message
    # NoteOn is sent in the loop
    velocity = int(velocity_val2)
    # delay to settle MIDI data
time.sleep(0.001)

# reads analog input to send MIDI data for PitchBend
# compares previous picthbend pot value to current value
if abs(pitchbend_val1 - pitchbend_val2) > 75:
    # updates previous value to hold current value
    pitchbend_val2 = pitchbend_val1
    # MIDI data has to be sent as an integer
    # this converts the pot data into an int
    # int is stored as a PitchBend message
    a_pitch_bend = PitchBend(int(pitchbend_val2))
    # PitchBend message is sent
    midi.send(a_pitch_bend)
    # delay to settle MIDI data
time.sleep(0.001)

# checks position of the rotary switch
# determines which notes are mapped to the cherry mx switches
for s in octave_selector:
    if not s.value:
        o = octave_selector.index(s)
        octave = octave_select[o]

# checks if strum select switch is engaged
if not strum_select.value:
    # if it is, then:
    # setup states for both strummer switches
    if strumUP.value and up_pick is None:
        up_pick = "strummed"
        pick = time.monotonic()
    if strumDOWN.value and down_pick is None:
        down_pick = "strummed"
        pick = time.monotonic()
    # bug fix using time.monotonic(): if you hit the strummer, but don't
    # hit a note
    # the state of the strummer switch is reset
    if (not pick) or ((time.monotonic() - pick) > 0.5 and
                      (down_pick or up_pick == "strummed")):
        up_pick = None
        down_pick = None

# if either strummer switch is hit
if (not strumUP.value and up_pick == "strummed") or (not strumDOWN.value
    and down_pick == "strummed"):
    # indexes the cherry mx switch array
    for i in range(12):
        buttons = note_buttons[i]
        # if any of the mx cherry switches are pressed
        # and they weren't previously pressed (checking note_states[i])
        # where i is the matching index from the note_buttons array
        if not buttons.value and not note_states[i]

© Adafruit Industries
# send the NoteOn message that matches with the octave[i]

# along with the velocity value
midi.send(NoteOn(octave[i], velocity))
# note number is printed to REPL
print(octave[i])
# note state is updated
note_states[i] = True
# updates strummer switch states
up_pick = None
down_pick = None
# delay to settle MIDI data
time.sleep(0.001)

# the next if statement allows for you to strum notes multiple times

# having to release the note

# if either strummer switch is hit
if (not strumUP.value and up_pick == "strummed") or (not strumDOWN.value and down_pick == "strummed"):
    # indexes the cherry mx switch array
    for i in range(12):
        buttons = note_buttons[i]
        # if any of the cherry mx switches are pressed
        # and they *were* previously pressed (checking note_states[i])
        # where i is the matching index from the note_buttons array
        if not buttons.value and note_states[i]:
            # send the NoteOn message that matches with the octave[i]

array

# along with the velocity value
midi.send(NoteOn(octave[i], velocity))
# note number is printed to REPL
print(octave[i])
# note state is updated
note_states[i] = True
# updates strummer switch states
up_pick = None
down_pick = None
# delay to settle MIDI data
time.sleep(0.001)

# the next for statement sends NoteOff when the cherry mx switches are released

# indexes the cherry mx switch array
for i in range(12):
    buttons = note_buttons[i]
    # if any of the cherry mx switches are released
    # and they *were* previously pressed (checking note_states[i])
    # where i is the matching index from the note_buttons array
    if buttons.value and note_states[i]:
        # send the NoteOff message that matches with the octave[i]

array

# along with the velocity value
midi.send(NoteOff(octave[i], velocity))
# note state is updated
note_states[i] = False
# updates strummer switch states
down_pick = None
up_pick = None
# delay to settle MIDI data
time.sleep(0.001)

# if strum select switch is not engaged
else:
    # indexes the cherry mx switch array
    for i in range(12):
        buttons = note_buttons[i]
        # if any of the mx cherry switches are pressed
        # and they weren't previously pressed (checking note_states[i])
        # where i is the matching index from the note_buttons array
        if not buttons.value and not note_states[i]:
            # send the NoteOn message that matches with the octave[i] array
            # along with the velocity value
            midi.send(NoteOn(octave[i], velocity))
            # note number is printed to REPL
            print(octave[i])
            # note state is updated
            note_states[i] = True
            # delay to settle MIDI data
            time.sleep(0.001)

    # if any of the cherry mx switches are released
    # and they *were* previously pressed (checking note_states[i])
    # where i is the matching index from the note_buttons array
    if (buttons.value and note_states[i]):
        # send the NoteOff message that matches with the octave[i]
        array
        # along with the velocity value
        midi.send(NoteOff(octave[i], velocity))
        # note state is updated
        note_states[i] = False
        # delay to settle MIDI data
        time.sleep(0.001)

    # delay to settle MIDI data
    time.sleep(0.005)

---

**Code Walkthrough**

**Libraries**

The code begins by importing the libraries.

```python
import time
import board
import simpleio
import busio
import adafruit_lis3dh
```
import digitalio
from digitalio import DigitalInOut, Direction, Pull
from analogio import AnalogIn
import usb_midi
import adafruit_midi
from adafruit_midi.note_on import NoteOn
from adafruit_midi.note_off import NoteOff
from adafruit_midi.control_change import ControlChange
from adafruit_midi.pitch_bend import PitchBend

MIDI Setup

After the libraries, `midi` is declared to bring in the `adafruit_midi` library. It declares that the USB port on the Grand Central board will be transmitting the MIDI data.

```python
midi = adafruit_midi.MIDI(midi_out=usb_midi.ports[1], out_channel=0)
```

LIS3DH Accelerometer Setup

Next the LIS3DH accelerometer is setup to be controlled via I2C, with its range being 2G.

```python
i2c = busio.I2C(board.SCL, board.SDA)
lis3dh = adafruit_lis3dh.LIS3DH_I2C(i2c)
lis3dh.range = adafruit_lis3dh.RANGE_2_G
```

Analog Inputs

The three potentiometers are then brought in to control three different MIDI parameters: pitch bend, modulation and velocity. The modulation and velocity
potentiometers are located on the body of the MIDI guitar and the pitch bend potentiometer is in the whammy bar assembly.

```python
pitchbend_pot = AnalogIn(board.A1)
mod_pot = AnalogIn(board.A2)
velocity_pot = AnalogIn(board.A3)
```

Mode Switches

This is followed by two switches, which will switch between two different modes. The first is for modulation control. It will allow you to switch between either using the potentiometer to control modulation or the LIS3DH, which will affect the modulation based on the physical position of the MIDI guitar.

The second switch is for strum mode. Since this is a MIDI guitar, you can strum the notes using the strummer, which will activate the switches inside the mechanism. However, this might not always be ideal. By using this switch you can turn off the strummer and just hit the note buttons on the neck of the MIDI guitar to play it like a traditional MIDI keyboard.

```python
mod_select = DigitalInOut(board.D52)
mod_select.direction = Direction.INPUT
mod_select.pull = Pull.UP

strum_select = DigitalInOut(board.D53)
strum_select.direction = Direction.INPUT
strum_select.pull = Pull.UP
```
Strumming Switches

Finally, the two switch inputs for the strumming mechanism are setup. There are two switches inside the strummer mechanism. They’re activated when you strum up and down with the strummer.

```python
strumUP = DigitalInOut(board.D22)
strumUP.direction = Direction.INPUT
strumUP.pull = Pull.UP

strumDOWN = DigitalInOut(board.D23)
strumDOWN.direction = Direction.INPUT
strumDOWN.pull = Pull.UP
```

Guitar Neck Buttons

Now two arrays of inputs are brought in. The first is for the note inputs, or buttons that are on the neck of the guitar. There are 12 of them.
```python

note_buttons = []

for pin in note_pins:
    note_pin = digitalio.DigitalInOut(pin)
    note_pin.direction = digitalio.Direction.INPUT
    note_pin.pull = digitalio.Pull.UP
    note_buttons.append(note_pin)
```

**Octave Selector**

This is followed by the inputs for the rotary switch. The switch has 8 inputs and as a result 8 pins are set up. The switch will allow you to change octaves for the notes that you can play. This expands the range of the MIDI guitar by quite a bit rather than being stuck with just the 12 notes setup with the buttons.

```python
    board.D30, board.D31]

octave_selector = []

for pin in oct_sel_pins:
    sel_pin = digitalio.DigitalInOut(pin)
    sel_pin.direction = digitalio.Direction.INPUT
    sel_pin.pull = digitalio.Pull.UP
    octave_selector.append(sel_pin)
```
Digital Input States

After the peripherals have been setup, the state machines can follow. There are 12 states setup for the 12 different notes to help with debouncing. Their names correspond with the note name that will be assigned to each button.

There are also two states, strummed and pick, that will be used to track the state of the strummer.

```python
note_e_pressed = None
note_f_pressed = None
note_fsharp_pressed = None
note_g_pressed = None
note_gsharp_pressed = None
note_a_pressed = None
note_asharp_pressed = None
note_b_pressed = None
note_c_pressed = None
note_csharp_pressed = None
note_d_pressed = None
note_dsharp_pressed = None
strummed = None
pick = None
```

Analog Input States

There are some variables setup to help track the positions of the potentiometers and the LIS3DH. These will be used to compare to the previous value reading to get the current reading for each input.

```python
pitchbend_val2 = 0
mod_val2 = 0
velocity_val2 = 0
acc_pos_val2 = 0
acc_neg_val2 = 0
```
Note Input Arrays

The 12 states that were just set up for the note inputs are put into an array called `note_states`. This array will be able to match with the `note_buttons` array in the loop.

```python
note_states = [note_e_pressed, note_f_pressed, note_fsharp_pressed, note_g_pressed, note_gsharp_pressed, note_a_pressed, note_asharp_pressed, note_b_pressed, note_c_pressed, note_csharp_pressed, note_d_pressed, note_dsharp_pressed]
```

This is followed by an array of the MIDI note numbers called `note_numbers`. Next is a list of the note names that correspond with those numbers. The list is stored in parenthesis and is set to equal `note_numbers`. Those note names are now variables that are holding the note numbers. This works since the names are in the same order as the numbers. This lets you easily type in the note names when creating your note arrays that will be tied to the buttons on the guitar neck. It tends to make more sense to look at note names than what can appear to be random numbers.

```python
(A0, Bb0, B0, C1, Db1, D1, Eb1, E1, F1, Gb1, G1, Ab1, A1, Bb1, B1, C2, Db2, D2, Eb2, E2, F2, Gb2, G2, Ab2, A2, Bb2, B2, C3, Db3, D3, Eb3, E3, F3, Gb3, G3, Ab3, A3, Bb3, B3, C4, Db4, D4, Eb4, E4, F4, Gb4, G4, Ab4, A4, Bb4, B4, C5, Db5, D5, Eb5, E5, F5, Gb5, G5, Ab5, A5, Bb5, B5, C6, Db6, D6, Eb6, E6, F6, Gb6, G6, Ab6, A6, Bb6, B6, C7, Db7, D7, Eb7, E7, F7, Gb7, G7, Ab7, A7, Bb7, B7, C8, Db8, D8, Eb8, E8, F8, Gb8, G8, Ab8, A8, Bb8, B8, C9, Db9, D9, Eb9, E9, F9, Gb9, G9) = note_numbers
```

Up next there are 8 arrays containing 12 note names each. These variables are being imported from the list above. Each note name corresponds to a MIDI note number. When these numbers are sent along with a NoteOn message in MIDI then the proper pitch is played.

These arrays are for the different octaves that can be selected using the rotary switch. Each array begins with an E natural note. This was chosen because the lowest string on a guitar in standard tuning is tuned to E natural. The notes are chromatic and since there are 12 (one for each button) you’re just one note shy of a full octave.
You can also customize the notes that correspond to each button by editing the arrays.

```
octave_8_cc = [E8, F8, Gb8, G8, Ab8, A8, Bb8, B8, C9, Db9, D9, Eb9]
octave_7_cc = [E7, F7, Gb7, G7, Ab7, A7, Bb7, B7, C8, Db8, D8, Eb8]
octave_6_cc = [E6, F6, Gb6, G6, Ab6, A6, Bb6, B6, C7, Db7, D7, Eb7]
octave_5_cc = [E5, F5, Gb5, G5, Ab5, A5, Bb5, B5, C6, Db6, D6, Eb6]
octave_4_cc = [E4, F4, Gb4, G4, Ab4, A4, Bb4, B4, C5, Db5, D5, Eb5]
octave_2_cc = [E2, F2, Gb2, G2, Ab2, A2, Bb2, B2, C3, Db3, D3, Eb3]
octave_1_cc = [E1, F1, Gb1, G1, Ab1, A1, Bb1, B1, C2, Db2, D2, Eb2]
```

These arrays are then put into the `octave_select` array. This allows for the rotary switch to select the different note ranges.

```
octave_select = [octave_1_cc, octave_2_cc, octave_3_cc, octave_4_cc, octave_5_cc,
octave_6_cc, octave_7_cc, octave_8_cc]
```

**Reading Analog Inputs**

This is followed by a fairly standard function when you work with analog inputs in CircuitPython. It allows you to read the value of an analog input continuously.

```
def val(pin):
    return pin.value
```

**Beginning REPL Message**

Finally, right before the loop, there are some debugging strings that are printed out to the REPL to let you know which MIDI channel output is active.

```
print("MIDI Guitar")
print("Default output MIDI channel:", midi.out_channel + 1)
```
The Loop

Mapping Analog Values to MIDI Parameter Values

At the beginning of the loop, \( x, y \) and \( z \) are defined to hold the values of the accelerometer.

Following that, there are five variables assigned to hold analog values of the three potentiometers and two accelerometer readings. The values for the potentiometers are being brought in with the `val()` function setup before the loop. The `map_range()` function is being used to convert the analog voltage value range to values that match with the MIDI CC ranges being controlled by the potentiometers and accelerometer.

The accelerometer setup is a little different than the potentiometers. The goal for the accelerometer is to control modulation. When the accelerometer is in a centralized, or neutral, position the modulation will be set to 0. However, when you tilt the guitar up or down the modulation amount will increase. That's why there are mapped values for negative and positive ranges.

```python
while True:
    x, y, z = [value / adafruit_lis3dh.STANDARD_GRAVITY for value in lis3dh.acceleration]
    pitchbend_val1 = round(simpleio.map_range(val(tonebend_pot), 0, 65535, 0, 16383))
    mod_val1 = round(simpleio.map_range(val(mod_pot), 0, 65535, 0, 127))
    velocity_val1 = round(simpleio.map_range(val(velocity_pot), 0, 65535, 0, 127))
    acc_pos_val1 = round(simpleio.map_range(x, 0, 0.650, 127, 0))
    acc_neg_val1 = round(simpleio.map_range(y, -0.925, 0, 127, 0))
```

You may need to experiment with your accelerometer value ranges depending on your preferences.

Sending MIDI Data with Analog Data

The rest of the loop defines what MIDI messages are sent as a result of using all of the different control interfaces. These are stored in a series of `if` statements.

The first `if` statement is checking the value of the `mod_select` switch. The `mod_select` determines whether modulation is being controlled by the potentiometer (`mod_pot`) or the accelerometer. For the accelerometer, a neutral position sends a value of
for modulation and then if you tilt the MIDI guitar up or down the value increases. The potentiometer allows you to set an unchanging value for modulation.

This is followed by an `if` statement for the potentiometer that controls the velocity value. Velocity in MIDI is the volume level that a note is recorded at. You won't necessarily notice it while you're actively playing (depending on the patch), but on playing back MIDI data it can make the music sound more human if there's variation in the levels.

The last analog potentiometer controls PitchBend and lives in the whammy bar mechanism. This parameter allows you to adjust the pitch up or down on a note that you're playing.

```python
if not mod_select.value:
    if (abs(acc_pos_val1 - acc_pos_val2) < 50):
        acc_pos_val2 = acc_pos_val1
        accelerator_pos = int(acc_pos_val2)
        accWheel_pos = ControlChange(1, accelerator_pos)
        midi.send(accWheel_pos)
        print(accelerator_pos)
        time.sleep(0.001)
    elif (abs(acc_neg_val1 - acc_neg_val2) < 50):
        acc_neg_val2 = acc_neg_val1
        accelerator_neg = int(acc_neg_val2)
        accWheel_neg = ControlChange(1, accelerator_neg)
        midi.send(accWheel_neg)
        print(accelerator_neg)
        time.sleep(0.001)
    else:
        if (abs(mod_val1 - mod_val2) > 2):
            mod_val2 = mod_val1
            modulation = int(mod_val2)
            modWheel = ControlChange(1, modulation)
            midi.send(modWheel)
            print(modulation)
            time.sleep(0.001)
        if (abs(velocity_val1 - velocity_val2) > 2):
            velocity_val2 = velocity_val1
            velocity = int(velocity_val2)
            time.sleep(0.001)
        if (abs(pitchbend_val1 - pitchbend_val2) > 75):
            pitchbend_val2 = pitchbend_val1
            a_pitch_bend = PitchBend(int(pitchbend_val2))
            midi.send(a_pitch_bend)
            print(int(pitchbend_val2))
            time.sleep(0.001)
```

How the MIDI Messages Are Being Sent

All of these `if` statements are using the same code structure to translate these analog values to MIDI values that can be successfully sent to your MIDI software. First, it checks to see if the value of the input has changed since the last time it was
checked. If it has, it checks to see if it has surpassed the threshold, which varies depending on the range of the input and MIDI parameter.

```python
if (abs(mod_val1 - mod_val2) &gt; 2):
```

If the value has changed enough, then the current value is logged and a new variable is setup to hold that value as an integer. MIDI message values need to be integers in order to be sent properly.

```python
mod_val2 = mod_val1
modulation = int(mod_val2)
```

Another variable is setup to hold the MIDI message, which includes the type of message, MIDI channel and the value as an integer.

```python
modWheel = ControlChange(1, modulation)
```

The MIDI message is finally sent with `midi.send()`. This is followed by a quick delay and that completes the process.

```python
midi.send(modWheel)
time.sleep(0.001)
```

**Octave Selection**

After the analog inputs, the code moves on to the digital inputs. This involves the rotary switch, the MX switches on the neck of the MIDI guitar and the two switches in the strummer.

This section begins by checking the position of the rotary switch to determine which octave is active. This is stored as `octave` and is called later when the NoteOn MIDI message is sent.

```python
for s in octave_selector:
    if not s.value:
        o = octave_selector.index(s)
        octave = octave_select[o]
```

**Strum Mode**

Next, the `strum_select` switch's position is checked. This switch determines whether you are strumming your MIDI guitar or just pressing the notes on the guitar's neck like a keyboard.
After checking if strum mode is active, the states of both switches located in the strum bar mechanism are checked. This takes care of debouncing the switches. There is also a variable called pick (named for a guitar pick) that is holding time.monotonic().

There is an if statement after the strummer switch state checks. It prevents the strummer from a glitch. If a note is not pressed down when you hit the strum bar, then the code is waiting for a note to be pressed on the neck. If you hit the strummer and then a few minutes later hit a note, it will sound. By having this line, the state of the two switches is reset to its initial none state in the event that this happens.

```python
if not strum.select.value:
    if strumUP.value and up_pick == None:
        up_pick = "strummed"
        pick = time.monotonic()
    if strumDOWN.value and down_pick == None:
        down_pick = "strummed"
        pick = time.monotonic()
    if (not pick) or ((time.monotonic() - pick) &gt; 0.5 and (down_pick or up_pick == "strummed")):
        up_pick = None
        down_pick = None
```

After the strumming is all setup, the code checks to see if the strummer has been strummed in either direction. If it has, then the 12 buttons on the neck are brought in with i being the index that identifies which button has been pressed. The button's values and states are then checked.

If the button has been pressed, then the NoteOn MIDI message is sent. That message includes the matching octave's note and the velocity value. The note's state and strummed status are updated and everything is closed out by a brief delay.

```python
if (not strumUP.value and up_pick == "strummed") or (not strumDOWN.value and down_pick == "strummed"):
    for i in range(12):
        buttons = note_buttons[i]
        if not buttons.value and not note_states[i]:
            midi.send(NoteOn(octave[i], velocity))
            note_states[i] = True
            print(octave[i])
        up_pick = None
        down_pick = None
        time.sleep(0.001)
```

With MIDI, if you play a note you also need to tell it to stop playing. Otherwise, it will go on forever. This next portion of code does just that. It checks to see if a note button is no longer being pressed. If that is true, then the note and strum states are set to their defaults and the NoteOff MIDI message is sent.
for i in range(12):
    buttons = note_buttons[i]
    if buttons.value and note_states[i]:
        note_states[i] = False
        strummed = None
        midi.send(NoteOff(octave[i], velocity))
        time.sleep(0.001)

Tapping Mode (No Strumming Required)

A wild `else` appears! If you scroll up a bit, you'll see that this `else` is the partner to the `if not strum_select.value` statement. This `else` statement allows you to play MIDI notes on this guitar without using the strum bar. You can just press the buttons on the guitar's neck just like a MIDI keyboard.

You'll see that the code matches our previous code to allow for NoteOn and NoteOff messages. The only thing missing is the portions about the strum bar.

else:
    for i in range(12):
        buttons = note_buttons[i]
        if not buttons.value and not note_states[i]:
            midi.send(NoteOn(octave[i], velocity))
            note_states[i] = True
            print(octave[i])
            time.sleep(0.001)
        if (buttons.value and note_states[i]):
            note_states[i] = False
            midi.send(NoteOff(octave[i], velocity))
            time.sleep(0.001)

---

3D Printing

3D Printed Parts

STL files for 3D printing are oriented to print "as-is" on FDM style machines. Parts are designed to 3D print without any support material. Original design source may be downloaded using the links below.

Download CAD from Fusion 360

Download CAD Files from Thingiverse
Body Parts
Here's a list of the body parts file names.

- body-left-top.stl
- body-left-bottom.stl
- body-right-top.stl
- body-right-bottom.stl
- body-middle-bottom.stl
- body-middle-top.stl
- head-case.stl
- head-diffuser.stl
- head-top.stl
- neck-bottom.stl
- neck-top.stl

Strumming Parts
These parts make the strumming mechanism.

- pick-base.stl
- pick-hinge.stl (2x)
- pick-joint.stl (2x)
- pick-mount.stl
- pick-plate-sw-left.stl
- pick-plate-sw-right.stl

Note: 2x means to print two copies of that part.
Whammy Bar Parts
These parts make the whammy bar assembly.

wmy-plate-idler.stl
wmy-plate-pot.stl
wmy-plate-spring.stl
wmy-spring-ring-a.stl
wmy-spring-ring-b.stl
wmy-wheel-joint.stl
wmy-wheel-mid.stl
wmy-wheel-pot.stl
Extension Spring (6.35 x 22.2 mm)

Design Source Files
The project assembly was designed in Fusion 360. This can be downloaded in different formats like STEP, STL and more. Electronic components like Adafruit's board, displays, connectors and more can be downloaded from the Adafruit CAD parts GitHub Repo().

Slicing Parts
The 3D models in the STL files have been oriented to print "as-is". With some 3D printers, the neck parts would not fit – With the parts rotated, it can fit diagonally across the bed. These parts do no require any support material.
Setup Parts
Multiple parts can be placed on the bed. By planning the arrangement, the placement of the parts can allow for several large parts like the left and right wings of the guitar body.

Large Surface Area
Some of the parts in the guitar body feature large surfaces and thus requires a well leveled bed. Warping in the corners could be an issue – Using a brim could be helpful and prevent corners from warping. Ensure the bed surface is clean and dust free for a perfect first layer.

Neck Head Assembly

Hardware for Neck and Head
Use the following hardware to attach the head of the guitar to the neck of the guitar.
Install Screws
Insert the M3 screws through the two mounting holes on the end of the neck-bottom.stl part. Place the head-case.stl part onto the threads of the screws and joint the two parts together.

Secure Hardware
Install and fasten M3 hex nuts onto the threads of the M3 screws. Use a pair of needle nose pliers to secure the neck-bottom.stl and head-case.stl parts together.

Assembled Neck and Head
The neck and head are now assembled and ready for the next set of assembly.
Head Cover Assembly

Black LED Acrylic
The head-diffuser.stl can either be 3D printed part or CNC/Laser cut from acrylic. This build uses Black LED Acrylic as the material for making the part. The diffuser is designed to press fit into the opening in the head-top.stl cover part.

Glue Diffuser to Cover
Hot glue can be used to seal the diffuser to the top cover.

Secured Diffuser
Without the hot glue the diffuser may pop out once the cover is installed into the head of the guitar.
Installed Diffuser Cover
With the diffuser now installed, the top cover is ready for the head of the guitar.

Body Assembly

Guitar Body – Bottom Half
Use the following hardware to assembly the bottom half of the guitar body.

4x M3 x 6mm pan head screws
4x M3 hex nuts

Install Machine Screws
Insert and fit the M3 machine screws through the tabs on the middle part of the bottom half of the guitar body.
Secure Machine Screws
Place the left and right parts of the bottom half of the guitar body onto the screw threads. Insert and fasten the M3 hex nuts onto the threads of the M3 screws. Use a pair of needle nose pliers to tightly fasten the hex nuts.

Guitar Body – Top Half
Use the following hardware to assembly the bottom half of the guitar body.

4x M3 x 6mm pan head screws
4x M3 hex nuts

Install and Secure Screws
Follow the same procedure for installing the hardware to the top half of the guitar body. Insert the M3 screw through the mounting tabs and use M3 hex nuts to secure the parts together.
Installing Guitar Body and Neck
Use the following hardware to secure the neck of the guitar to the top half of the guitar body.

3x M3 x 6mm pan head screws
3x M3 hex nuts

Install Neck to Top Half Guitar Body
Install the neck-bottom.stl part into the body-middle-top.stl part by press fitting them together. Check and ensure the three mounting holes are nicely lined up with each other.

Install Hardware into Neck
Insert the M3 screws into the neck-bottom.stl so the threads protrude through the body-middle-top.stl part.
Secure and Fasten Hardware
Insert and fasten the M3 hex nuts onto the threads of the M3 screws. Use a pair of needle nose pliers to get a holding of the M3 hex nuts.

Assembled Guitar Body and Neck
The top half and bottom half of the guitar body are assembled.

Strum Assembly

Strum Hardware Screws
Gather the following hardware for creating the strum mechanism.

Hardware
4x M3 x 10mm pan head screws
4x M3 hex nuts
Install Hinges
Insert a screw through the bottom of the pick-mount.stl part. Place one of the pick-hinge.stl parts on top of the standoffs. Insert and fasten an M3 hex nut onto the thread of the screw.

Secure Hex Nuts
Repeat the process for all four standoffs. A pair of needle nose pliers can assist in grasping the hex nuts for a secure fitting.

Install Joints to Pick
Install the two pick-joint.stl parts into the pick-base.stl by inserting them from the inside. The flange should be flush with the inner wall of the pick-base.stl.
Hot Glue Joints to Pick
Apply a moderate amount of hot glue to the inside of the pick-base.stl. This will prevent the pick-joint.stl from being pushed in and secured in place.

Strum Pick with Joints
Double check the joints are installed on both ends.

Install Pick to Base
Begin to install the pick-base.stl part into the pick-mount.stl. Fit one of the ends into one of the pick-hinge.stl parts. Flex the pick-mount.stl slightly to allow the second pick-joint.stl part to fit into the second pick-hinge.stl.
Installed Pick
Double check both ends are installed properly. The pick-base.stl should rotate freely.

Pick Assembly
The pick assembly is ready for the next set of parts.

Parts for Micro Switches
The two micro switches need 4x M3 x 16mm long machine screws and 4x M3 hex nuts. The switches will be mounted to the pick-plate-sw-left.stl and pick-plate-sw-left.stl parts.
Secure Switches to Plate
Place one of the micro switch onto either of the pick-plate-sw parts. Orient the micro switch so the mounting holes are lined up with the plate. Insert one of M3 x 16mm long screws through the micro switch and plate. Insert and fasten an M3 hex nut to secure the switch to the plate.

Installed Switches
Repeat this process for the second micro switch. Double check the orientation of the switches are matching with the photo.

Wires for Micro Switches
Use the following wires for connecting the micro switches.

- 2-pin JST female cable – 114mm long
- 4-wire ribbon cable – 180mm long
- 2x 2-pin JST male connectors
Tinning Wires
Separate the 4-wire ribbon cable so there's 2x 2-wire cables. Use wire strippers to remove a bit of insulation from the tips of each wire. Tin the ends using a bit of solder – This will prevent the strands of wires from fraying. Third helping hands can help assist while soldering.

Wire JST Connectors
Solder up the 2-wire ribbon cables to the male 2-pin JST connectors. Don't forget to add heat shrink tubing before soldering – This will insulate the exposed wires and prevents connections from shorting out.

2x 2-pin JST Cables
Repeat this process for the second 2-pin ribbon cable.
Micro Switch Ground Wire
Solder one of the wires from either 2-wire ribbon cable to the common ground on one of the micro switches. The ground is labeled on the side of the micro switch "COM". Tinning the pin of the micro switch can help solder adhere to the wire.

Solder Micro Switch Pin
Attach the other wire to the "NO" (Normally Open) pin on the micro switch.

Wired Micro Switches
Repeat this process for the second micro switch. Double check the wires are connecting to the correct pins and have solid connections.
Strum Installation
Get the parts for the strum assembly. The mounting plate, micro switches and pick base. This will need 4x M3 x 6mm long screws and 4x M3 hex nuts.

Install Strum Mounting Plate
Insert 4x M3 x 6mm long screws through the back of the guitar body. Use the four mounting holes in the center near the middle of the body. While holding screws in, place the pick-mount.stl part over the screws so the threads go through the four mounting holes.

Install Micro Switches
Place the two (pick-plate-sw-left.stl and pick-plate-sw-right.stl) micro switches over the threads of the screws. Reference for the photo for correct placement.
Secure Strum Parts
Insert 4x M3 hex nuts onto the screws and fasten tightly to secure the strum assembly to the body of the guitar.

Install Pick to Hinges
Fit the joints on the end of the pick-base.stl to the hinges on the pick-mount.stl. Start by fitting one end and then the other end. The parts can flex slightly to offer more clearance for the joints to be fitted into the hinges.

Test Strum Pick
Test out the micro switches by rocking the pick back and forth. The levers should easily be pressed. The pick should stay in the center when idle.
Assembled Strum
Double check the hex nuts in the strum assembly is nice and tight.

Whammy Assembly

Whammy Wheel Parts
The wmy-wheel-joint.stl and wmy-wheel-pot.stl parts will need to be super glued to the wmy-wheel-mid.stl part.

Glue Joint to Wheel
Start by applying a moderate amount of super glue to the wmy-wheel-joint.stl to one of the sides of the wmy-wheel-mid.stl part.
Glue Potentiometer Holder to Wheel
Add super glue to the wmy-wheel-pot.stl part and adhere it to the other side of the wmy-wheel-mid.stl part.

Glue Spring Plate
The wmy-plate-spring.stl part will need to be super glued to the wmy-plate-pot.stl part. Apply a moderate amount of super glue to the flat side of the wmy-plate-spring.stl part.

Glue Spring Plate to Potentiometer Plate
Adhere the wmy-plate-spring.stl part to the flat side of the wmy-plate-pot.stl part. Double check the edges are square on both of the parts.
Spring Assembly
An extension spring will be attached to the wmy-spring-ring-a.stl and wmy-spring-ring-a.stl parts to create the spring assembly.

Install Extension Spring
Install the extension spring to either the wmy-spring-ring-a.stl or wmy-spring-ring-a.stl part. Press fit one of the loop ends to the slot on the end of the spring rings.

Install Ext. Spring to Holders
Grab the second spring ring and install the remaining loop end. Hold the two spring rings together so they're together.
Install Spring Holder Set to Plate
Place the set of spring rings onto the cylindrical post on the wmy-plate-spring.stl part. The two spring rings are placed and fitted together. Reference the photo for correct placement.

Installed Spring Holder Assembly
The spring assembly is now ready to be installed into the rest of the whammy assembly.

Whammy Wires
Gather up the knobby potentiometer, 3-pin JST female cable, 10-wire silicone ribbon wire and the 3-pin JST male connector.

Measure and cut a piece of the 10-wire ribbon wire (88mm in length). Peel off a 3-wire set.
**JST Wire**
Wire up the 3-wire silicone cable to the 3-pin JST male connector. Don't forget to put heat shrink tubing on first!

**JST Cables**
Cut and trim the 3-pin JST female cable to prepare it for wiring. The following wire lengths are recommended.

- female cable – 74mm
- male cable – 88mm

**Solder Potentiometer**
Wire up the 3-wire silicone cable to the potentiometer. Connect the 3-pin JST female cable and follow the signal, voltage and ground wiring. Signal (white) in the middle pin. Voltage and ground on the side pins.
Test JST Connection
Double check the wires and cross reference the circuit diagram.

Center Potentiometer
Turn the knob so the white marker is centered with the middle pin.

Install Potentiometer
Press fit the knob on the potentiometer into the whammy wheel. Check the orientation of the pins are facing up.
Spring Tension
The tension in the spring needs to be tight in order for the wheel to snap back to the center. Using needle nose pliers can help assist in bending the loops so the spring tension is tighter.

Whammy Body Parts
The whammy assembly requires 4x M3 x 6mm screws and 4x M3 hex nuts. The two mounting brackets (wmy-plate-idler.stl and wmy-plate-pot.stl) will need to be secured and fastened to the bottom half of the body.

Install Whammy Idler
Insert 2x M3 x 6mm through the bottom of the guitar. Place the wmy-plate-idler.stl over the two screw threads. Insert and fasten 2x M3 hex nuts to secure the part to the body of the guitar.
Install Wheel to Body
Insert the remaining 2x M3 x 6mm screws into the mounting holes on the guitar. Place the wheel sub-assembly in the body of the guitar. Insert the cylinder from the wmy-wheel-pot.stl part into the hole in the center of the wmy-plate-idler.stl.

Whammy Assembled
Test out the check the whammy bar mechanism by moving the wheel back and forth. The hex nuts should be nice and tight.

Wiring 8-Way Switch

Wires for 8-way Switch
Get the wires ready for soldering to the 8-way selector switch. Use the following lengths to measure and cut the wires to size.

Wire Lengths
10-wire ribbon cable – 90mm
4-wire JST cable – 102mm
Tin Ribbon Cable
Use wire stripper to remove insulation from the tips of each wire. Tin the exposed wire by adding a bit of solder.

Ribbon Cable Wires
Repeat tinning on the other end of the ribbon cable. This helps prevent the strands of wire from fraying.

Tin JST Cable Wires
Use wire strippers to remove the tips from the 4-wire JST cable. Tin the strands of wire by adding a bit of solder.
Tinned Wires
The cables are ready to wire up.

Connect 8-way Pins
Add pieces of heat shrink tubing to the wires before soldering. Solder eight of the ten wires from the ribbon cable to the pins on the rotary switch.

Connect 8-way Ground
The single pin in the middle is the common ground. Solder the remaining two wires from the ribbon cable to the ground pin on the rotary switch.
Attaching JST connectors to ribbon cable
Get the following JST male connectors ready to wire up to the ribbon cable.

Solder JST connectors
Solder four of the wires from the rotary switch to one of the 4-pin JST male connectors. Add pieces of heat shrink tubing to the wires before soldering.

Wired 8-way switch
Repeat this process for the other 4-pin JST male connector. Solder the two ground wires from the rotary switch to the 2-pin JST male connector.
Wiring Pots

Wires for Potentiometers
Get the ribbon cable, JST cables and connectors ready to wire up to the potentiometers. Use the following wire lengths.

- 2x 3-pin JST cables – 80mm
- 6-wire ribbon cable – 126mm

Tin Ribbon Cable Wires
Use wire stripper to remove a bit of insulation from each of the wires. Tin the strands of wire by adding a bit of solder.

Solder JST Connectors
Tin the pins on the JST connectors by adding a bit of solder. Add pieces of heat shrink tubing to three of the six wires from the ribbon cable. Attach the three wires to the 3-pin JST male connector.
Wired 3-pin JST Connectors
Repeat this process for the last three wires.

Wire Potentiometers
Connect three from the ribbon cable to the pins on the potentiometer. Add pieces of heat shrink tubing before soldering. Follow the order of the colors to get the correct polarity.

White – Middle Pin
Red/Black – Side Pin

Wired Potentiometers
Repeat this process for the second potentiometer.
Wiring Toggle Switches

**Toggle Switch Wires**
Get the cables, wires and connectors ready for the toggle switches. Use the following wire lengths for the 2x 2-pin JST cable and 4-wire ribbon cable.

Wires Lengths

- 4-wire ribbon cable – 98mm
- 2-wire JST cables – 66mm

**Tin Toggle Switch Wires**
Use wire strippers to remove a bit of insulation from the tips of each wire. Tin the strands of wire by adding a bit of solder.

**2-pin JST Connectors**
Tin the pins on both of the 2-pin JST connectors by adding a bit of solder.
Solder 2-pin JST wires
Wire up two of the four wires from the ribbon cable to one of the 2-pin JST male connectors. Be sure to add pieces of heat shrink tubing before soldering.

Wired 2-pin JST
Repeat this process for the second 2-pin JST male connector.

Solder Toggle Switch Pins
Attach the ribbon wires to the middle and side pins of the toggle switch. Be sure to add pieces of heat shrink tubing before soldering.
Wired Toggle Switches
Repeat this process for the second toggle switch.

Tin 2-pin JST cables
Use wire strippers to remove insulation from the tips of the wires. Tin the strands of wire by adding a bit of solder.

Wired Toggle Switch Cables
Repeat this process for the second 2-pin JST cable. Double check the wiring for solid solder joints.
Wiring Perma-Proto

Perma-Proto Power Rails
A piece of perma-proto can be useful for extended the voltage and ground pins. Use a rotary power tool to cut the power rails from the quarter size perma-proto PCB – Use a vice to lock down the PCB to a tabletop. Use a shop vac to capture the dust.

2-wire ribbon cable
Use a 2-wire ribbon cable to connect the power and ground pins.

Wire Lengths
2-wire ribbon cable – 52mm

Solder Wires to Pins
Solder the wires to the voltage and ground on the PCB.
Wired Perma-Proto Power Rails
This will be wired to the power and ground pins on the Adafruit Grand Central M4.

Wiring The LIS3DH

Wires for LIS3DH
Get the wires ready for the LIS3DH breakout board. Use the following wire lengths.

- 4-pin JST cable – 116m
- 4-wire ribbon cable - 250mm

Tin Ribbon Cable Wires
Use wire strippers to remove insulation from the tips of each wire. Tin the exposed wire using a bit of solder.
Tin 4-pin JST Cable
Use wire strippers to remove insulation from the tips of each wire in the 4-wire JST cable. Tin the strands of wire by adding a bit of solder.

Solder Ribbon Wires
Connect the 4-pin male JST connector to the 4-pin JST cable. Attach the wires from the ribbon cable to the 4-pin JST male connector. Make sure to add pieces of heat shrink tubing before soldering.

Solder LIS3DH Wires
Solder the wires from the ribbon cable to the pins on the LIS3DH breakout board. Reference pin outs and the colored wiring.

Green Wire – SCL
White Wire – SDA
Red Wire – VIN
Black Wire – GND
Wired LIS3DH
The colored wires from the 4-pin JST cable should match the pinouts of the LIS3DH breakout.

Wiring MX Switches

Wires for MX Switches
Get the wires and Cherry MX switches ready. Use the following wire lengths.

Wire Lengths
3x 4-pin JST cable – 116mm
1x 2-pin JST cable – 116mm
2x 10-wire ribbon cable – 305mm

Tin 4-wire JST cables
Use wire strippers to remove a bit of insulation from the tips of each wire from the 4-pin JST cable. Tin the strands of wire by adding a bit of solder.
Tin 2-wire JST Cable
Use wire strippers to remove a bit of insulation from the tips of each wire in the 2-pin JST cable. Tin the strands of wire by adding a bit of solder.

Tinned JST Cables
Repeat the tinning process for the remaining JST cables. Pieces of heat shrink tubing will keep the wires bundled together.

Plate for MX Switches
Get the neck-top.stl part ready to install the Cherry MX switches.
Installing MX Switches
Cherry MX switches are press fitted into the square holes through the top of the neck-top.stl part. The MX switches can be oriented in any orientation – Use the orientation with the most optimal placement for the wired connections.

Install MX Switches
The switches are placed so the pins are following a consistent orientation.

Installed MX Switches (Bottom View)
Double check the placement of the switches once they're all installed into the neck-top.stl part. If any of the switches are out of place, they can be "popped" out.
Installed MX Switches (Top View)
The top of the switches feature the Cherry MX embossed labeling. They can be used to visually see if they’re all following in the same orientation. Note: a combination of "Blue" and "Silver" switches are used here unintentionally due to components on hand.

Wires from Shared Ground
Peel off a single wire from the 10-wire ribbon cable. This single wire is cut into several pieces that will be used to connect all of the ground pins together. The switches will share a common ground.

Each wire is about 35mm in length.

Soldering Ground
Secure the neck-top.stl part so it's doesn't move around while soldering. A Panavise or third helping hands can be used. Tin one of pins on the first Cherry MX switch.
Shared Ground
The second Cherry MX switch has two wires soldered to the ground pin. The second wire will be connected to the third switch. A pair of tweezers can be helpful in handling the wires while soldering.

Ground Wired
Repeat this process for nine of the remaining Cherry MX switches. The last switch (the 12th) in the arrangement will need a third wire for connecting to the ground pins on the Adafruit Grand Central M4.

Check Ground Wiring
Take a moment to inspect your wires to ensure the solder joints are solid.
Tin Longer Ground Wire
Measure and cut a 2-wire piece from the 10-wire ribbon cable. Use the following wire length.

2-wire ribbon cable – 117mm

Solder JST Connector
Solder the 2-wire ribbon cable to the 2-pin male JST connector. Add pieces of heat shrink tubing before soldering.

Solder Last Ground MX Switch
Solder the 2-pin ribbon cable to the ground pin on the last mx switch.
Wired JST Ground Cable
Double check the wiring in the MX switches.

Wiring 4-pin JST Connectors
Split the 10-wire ribbon (305mm long) cable so there's 3x 4-wire ribbon cables. These will be used to solder to the 3x 4-pin JST connectors.

Tinning 4-pin JST Cables
Repeat this process for the remaining 4-pin JST cables.
Wiring First Key
Wire up one of the 4-wire ribbon cables to the first MX switch.

4 Keys Wired
Proceed to wire the next three MX switches. Note, the wires are shortened in length inline with the MX switches.

8 Keys Wired
Proceed to solder the next 4-wire ribbon cable to four more MX switches. Follow the same process and shorten the wires per switch.
12 Keys Wired
Proceed to wire the third 4-wire ribbon cable to the last remaining 4x MX switches.

Wired MX Switches
Thoroughly inspect the wires and double check all of the solder joints are solid.

Group Wiring
The 3x ribbon cables are layered and bundled together. This keeps the wires nice and neat.

Wiring NeoPixel Feather
If you are installing the optional lighting for the neck of the guitar, follow the wiring below.
Feather and NeoPixel Wires
Get the Feather, NeoPixel strip, slide switch and wires ready. Use the following wire lengths.

Wire Lengths
- 3-pin JST cable – 70mm
- 3-wire ribbon cable – 74mm
- 2-wire ribbon cable – 65mm

Count, Cut and Unsheathe
This project is using a Mini Skinny NeoPixel strip with 144 pixels per meter. The NeoPixel LED strip will be hot glued to the head of the guitar. The following NeoPixel strip was used in this project.

- Mini Skinny NeoPixel Strip (144/m) – 35 NeoPixels

Tin 3-pin JST Cable
Use wire strippers to remove insulation from the tips of each wire. Tin the exposed wire using a bit of solder.
Tin 3-wire ribbon cable
Use wire strippers to remove insulation from the tips of each wire. Tin the exposed wire using a bit of solder.

Strip 3-pin JST cable solder
Solder the 3-pin JST cable to the NeoPixel strip. Check and find the first NeoPixel in the strip. Solder white wire to data In, red wire to 5V and black wire to ground. Add pieces of heat shrink tubing before soldering connections.

Feather 3-wire ribbon cable solder
Solder the 3-wire ribbon cable to the pins on the Feather. Follow the cable and match up the colored wires from the 3-pin JST cable. White wire to Pin 5, black wire to ground and red wire to 3V.
Solder Slide Switch Ribbon Cable
Solder a 2-wire ribbon cable to the slide switch. Use the middle pin and either pin on the side. Add pieces of heat shrink tubing before making wired connections.

Solder Slide Switch to Feather
Solder slide switch ribbon cable to EN and GND pins on the Feather.

Test Circuit
Connect a 500mAh battery to the JST battery port on the Feather. This project uses the CircuitPython NeoPixel demo code (). Use the slide switch to turn the circuit on and off.
Strip Assembly

NeoPixel Strip for Head
Get the NeoPixel strip ready to install into the head of the guitar.

Install Strip to Head
Fit the NeoPixel strip into the head of the guitar. Insert the 3-pin JST cable through the hole – route cable to the neck. Carefully bend the strip so it follows the shape of the head. Use hot glue to secure the strip to the inner walls.

Install Feather Screws
Insert 4x M2.5 x 8mm screws into the mounting holes on the Feather PCB. Insert and fasten 4x M2.5 hex nuts. Note: longer (10mm) M2.5 screws were added due to hardware on hand – These can be cut short.
Installing Feather to Neck
Get the Feather ready to install into the mounting holes on the neck.

Secure Feather to Neck
Place the Feather into the neck and fit the threads of the screws through the mounting holes. Insert and fasten four M2.5 hex nuts onto the threads of the screws to secure Feather to neck.

Install Feather Slide Switch
Insert body of slide switch at an angle into the built-in switch holder.
Connect Strip to Feather
Connect the 3-pin JST cable from the strip to the 3-pin JST cable on the Feather.

Install Strip JST Wiring
Fit the JST cable in between the Feather PCB and inner walls of the neck. Tuck in the cable so it's nice and neat.

Install the LIS3DH

Hardware for LIS3DH
Get the M2.5 x 8mm screws and M2.5 hex nuts ready for the LIS3DH.
Install LIS3DH Screws
Insert and install 2x M2.5 x 8mm screws into the mounting holes. Insert and fasten 2x M2.5 hex nuts on to the threads of the screws.

Install LIS3DH to Neck
Place the LIS3DH into the neck of the guitar. Fit the thread of the screws through the two mounting holes on the neck of the guitar.

Secure LIS3DH to Neck
Insert and fasten 2x M2.5 hex nuts onto the threads of the screws.
Install Battery
Place the 500mAh battery into the neck of the guitar. Note: The battery cable is routed underneath the LIS3DH PCB for extra neatness.

Wiring the M4

Grand Central M4 Wiring
Get the cables from the panel mounted components ready to wire into the Grand Central M4 Express. 2x toggle switches, 2x potentiometers, 8-way rotary switch and the perma-proto pcb. Use the following hardware.

8x M3 x 6mm screws
4x M3 x 20mm standoffs

Install M3 Standoffs
Place an M3 screw through the mounting holes in each corner of the Grand Central M4 PCB. Insert and fasten the M3 standoffs onto the threads of the screws. Finger tighten all four standoffs.
Label JST Cables
Take a moment to label the JST cables. This will help connect components together once it's been installed into the body of the guitar. I used the following letters to reference these components.

X – LIS3DH
A,B,C – MX Switches
P1, P2 – Potentiometers
W – Potentiometer for Whammy
1-4, 5-8 – Rotary Switch
S1, S2 – Toggle Switches

Wire Prema-Proto
Solder the perma-proto PCB to the ground and 3V pin on the Grand Central M4. Reference photo for best wire placement.

Wire Potentiometers
Solder the white wires from the 2x potentiometers to pin 2 and 3 on the Grand Central M4.
Pots Ground and Voltage
Solder the voltage (red) and ground (black) wires to the pins on the perma-proto PCB.

Wire 8-Way Rotary Switch
Solder the eight wires from the rotary switch to the following pins on the Grand Central M4.

24-31

Wire 8-Way Rotary Switch to Ground
Solder the 2-pin JST cable to the ground pins on the perma-proto.
Wire the LIS3DH
Solder the green wire to SCL on Grand Central M4. Solder the white wire to SDA on the Grand Central M4.

Wire LIS3DH Ground and Voltage
Solder the ground and voltage wires to the ground and voltage pins on the perma-proto.

Wire Micro Switches (Strum)
Solder the 2-pin JST cables from the two micro switches to the following pins on the Grand Central M4.

Pin 22
Pin 23
Wire MX Switches
Solder the 3x 4-pin JST cables from the MX switches to the following pins on the Grand Central M4.
12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 14

M4 Wiring Mounted Components
Get the bottom half of the guitar body ready. The panel mounted components will be wired to the Grand Central M4.

M4 Wire Whammy Pot
Solder the white wire from whammy potentiometer to Pin #1 on the Grand Central M4.
Wire Whammy Ground and Voltage
Solder the ground and voltage wires from the whammy potentiometer to the ground and voltage pins on the perma-proto.

Wire Toggle Switches
Solder the 2-pin JST cables from the 2x toggle switches to the following pins on the Grand Central M4.

Switch 1 – Pin #52
Switch 2 – Pin #53

Wire Toggle Switches Ground and Voltage
Solder the voltage and ground wires from the toggle switches to the voltage and ground pins on the perma-proto.
M4 Wiring Check Point
Take a moment to thoroughly inspect the wiring.

Unplug Components
Begin to disconnect the various JST connectors. The components will need to be unplugged in order to install the Grand Central M4 into the body of the guitar.

Panel Mount Components

Panel Mounting Components
Get the components ready to panel mount onto the top half of the guitar body.
Install 8-way switch and toggle switches
Panel mount the 8-way rotary switch and two toggle switches. Use the included hardware to secure the components. Reference the photo for correct placement.

Install 2x Potentiometers
Panel mount the two potentiometers. Use the included hardware to secure the components. Reference the photo for correct placement.

Panel Mounted Components
Double check the components are installed correctly. Double check all of the hardware has been tightly fastened.
Assemble Grand Central

Install Grand Central M4
Place the Grand Central M4 PCB into the bottom half of the guitar. Reference the photo for correct placement.

Secure Grand Central M4
Line up standoffs with mounting holes. Insert and fasten 4x M3 x 6mm screws into the bottom half of the guitar. Fasten tightly.

Connect Micro Switches for Strum
Reconnect the JST cables from the two micro switches.
Connect Whammy
Reconnect the JST cable from the whammy potentiometer.

Installing micro USB Extension
Get the microUSB extension cable ready to connect and panel mount.

Secure USB Cable
Use included hardware to panel mount the USB cable to the side of the guitar.
Installed USB Extension
Connect micro USB cable to the USB port on the Grand Central M4. Reference photo for best placement of the USB extension cable.

Final Assembly

Install the Neck Cover
Get the neck cover with MX switches ready to install into the neck of the guitar. Orient the neck cover using the photo to reference correct placement.

Snap Fit Neck Cover
Insert and pass JST cables through the hole in neck of the guitar. Firmly press cover into neck to snap fit shut. Ensure all of the wires are inside the neck.
Snap Fit Head Cover
Install the LED cover into the head of the guitar. Firmly press cover into head to snap fit shut.

Test NeoPixel Strip
Power on NeoPixel strip to test out diffusion in the head cover. Black LED acrylic is pretty amazing!

Make Connections
Take a moment to connect the rest of the components. This is where labeling your wires comes in handy!
Install Keypads
Get the key caps ready to install onto the 12 MX switches. Note, only narrow shaped key caps will fit. Spacing between switches is short.

Installed Keypads
Key caps are press fitted onto MX switches actuators (+). Test key caps by pressing them. If they're getting stuck pressed down, try using different keys or even sand the edges.

Check Connections
Take a moment to thoroughly inspect the connections.
Closing the Guitar Body
Gently begin to close up the guitar body by bring the two halves together. Check and ensure all of the wires and cables are fitted inside the body of the guitar. You need to ensure wires are not caught between the halves of the body, which might cause intermittent operation.

Closing the Guitar (Continued)
The cables and connectors will need to be layered and flattened nicely in order for the two halves to close properly. This may take a few attempts to get right. Thoroughly check how the wires are routed and position them so they're laying flat. Press the two halves together to snap fit them shut.

Install the Whammy Bar
Insert the threaded end of the whammy bar into the hole in the center of the whammy wheel. Twist to fasten and screw the bar into the wheel until it's in a preferred position.
Install Knob for 8-way switch
Optionally install the selector knob onto the 8-way rotary switch.

Secure Knob for 8-way switch
The knob features a grub screw to secure onto the 8-way rotary selector switch. Place knob over tip of selector switch and press down. Use a small flat head screwdriver to tighten the grub screw.

Installed Knob for 8-way switch
This knob has a nice shape for handling and features a line mark to note its position.
Install Knobs for Potentiometers
Optionally install knobs onto the two potentiometers. Use the built-in grub screws to secure knobs onto the tips of each pot.

Secured Knobs for Potentiometers
These knobs appear to be ideal for guitar amps. Labels numbers 1-11 are noted on knobs.

Final Build
And there we have it! Take a moment to test out the various components. Twist some knobs, press some buttons, bend some pitches and strum the pick.

Let's Jam!
With your MX MIDI Guitar all assembled, you're ready to rock! You can use it with any software that allows MIDI input. Most commonly, you'll use a MIDI controller with music production software such as Reason, Garage Band, FL Studio, Ableton Live, etc.
You could use the guitar for recording your own music with MIDI or for playing live. With the strum bar you can play in a really unique way that isn't possible with regular synthesizers that use a standard keyboard for note input. In a live situation, the added control of the accelerometer in the neck opens up a world of possibilities.