# Table of Contents

## Overview
- What is MIDI? 5
- Why build a MIDI controller? 5
- Parts 6

## How MIDI is Transmitted
- MIDI In vs. MIDI Out 8
- MIDI Channels 9
- Connections 10

## MIDI Messages
- NoteOn 15
- NoteOff 15
- ControlChange Messages 15
- PitchBend 16
- Reading MIDI In 16

## Let's Build!
- Prerequisite Guides 18

## Basic MIDI Keyboard
- Circuit Diagram 19
- Setup the Feather RP2040 19
- CircuitPython Code 19
- Upload the Code and Libraries to the Feather RP2040 21
- How the CircuitPython Code Works 21
- Usage 21

## MIDI CC Control with Pots
- Circuit Diagram 22
- Setup the QT Py RP2040 23
- CircuitPython Code 23
- Upload the Code and Libraries to the QT Py RP2040 24
- How the CircuitPython Code Works 25
- Usage 25

## BLE MIDI Sequencer
- Setup the CLUE 26
- CircuitPython Code 26
- Upload the Code and Libraries to the CLUE 28
- How the CircuitPython Code Works 29
- Usage 29

## Receive and Display MIDI Messages
- Circuit Diagram 30
- Setup the QT Py RP2040 30
- CircuitPython Code 30
- Upload the Code and Libraries to the QT Py RP2040 32
- How the CircuitPython Code Works 32
- Usage 32
Receive MIDI Over UART and Send Over USB

• Setup the Feather RP2040
• CircuitPython Code
• Upload the Code and Libraries to the Feather RP2040
• How the CircuitPython Code Works
• Usage

Control Motors with MIDI

• Circuit Diagram
• Setup the Metro M4 Express
• CircuitPython Code
• Upload the Code and Libraries to the Metro M4 Express
• How the CircuitPython Code Works
• Usage

MIDI Projects on the Learn System

• Raspberry Pi Pico and LED Arcade Button MIDI Controller
• Trellis M4 Expressive MIDI Controller
• MIDI Solenoid Drummer
• MIDI Melody Maker
• Power Glove Wireless MIDI Controller
Overview

Whether you're a classically trained pianist or a hobbyist who thinks synths sound really cool, the allure of DIY MIDI projects are incredibly tempting. It can be hard to know how to get started though. What board should you use? How does the code work? What even is MIDI?

This guide will tell you everything you need to know to get started with your own MIDI projects so that you can focus on building rather than searching everywhere on the internet.

What is MIDI?

Music Instrument Digital Interface (MIDI) is a 7-bit communication protocol that dates back to the 1980’s. It may sound antiquated, but it is the standard digital communication method for all things music. If you've ever had a piano keyboard control a piece of software or digital audio workstation (DAW) on a computer, then you were using MIDI.

MIDI is dependable, predictable, reliable and at this point is usually plug and play without too much fuss. This also means that it's approachable for folks to build their own MIDI devices.

Why build a MIDI controller?

There are a lot of aftermarket MIDI devices available, so why would you want to build your own? When you build your own MIDI controller, you can design it specifically for what you need. If you aren't quite sure what you want to do with MIDI, you can
experiment and test things before committing to a full build. Additionally, you can share your MIDI project's details with the community for other musicians to benefit from.

Parts

Adafruit MIDI FeatherWing Kit
Turn your Feather into a song-bird with this musically-enabled FeatherWing that adds MIDI input and output jacks to just about any Feather. You get both input and output DIN-5 MIDI...
https://www.adafruit.com/product/4740

Adafruit Feather RP2040
A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought "this chip is going to be awesome when we give it the Feather...
https://www.adafruit.com/product/4884

Adafruit QT Py RP2040
What a cutie pie! Or is it... a QT Py? This diminutive dev board comes with one of our new favorite chip, the RP2040. It's been made famous in the new
https://www.adafruit.com/product/4900
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Quantity</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 x FeatherWing Doubler</td>
<td>FeatherWing Doubler</td>
<td>1</td>
<td><a href="https://www.adafruit.com/product/2890">https://www.adafruit.com/product/2890</a></td>
</tr>
<tr>
<td>1 x Breadboard</td>
<td>Full Size</td>
<td>1</td>
<td><a href="https://www.adafruit.com/product/239">https://www.adafruit.com/product/239</a></td>
</tr>
<tr>
<td>1 x Breadboard wire bundle</td>
<td>Various size wires for breadboarding</td>
<td>1</td>
<td><a href="https://www.adafruit.com/product/153">https://www.adafruit.com/product/153</a></td>
</tr>
<tr>
<td><strong>Adafruit CLUE - nRF52840 Express with Bluetooth LE</strong></td>
<td>Do you feel like you just don't have a CLUE? Well, we can help with that - get a CLUE here at Adafruit by picking up this sensor-packed development board. We wanted to build some...</td>
<td></td>
<td><a href="https://www.adafruit.com/product/4500">https://www.adafruit.com/product/4500</a></td>
</tr>
<tr>
<td><strong>Adafruit Metro M4 feat. Microchip ATSAMD51</strong></td>
<td>Are you ready? Really ready? Cause here comes the fastest, most powerful Metro ever. The Adafruit Metro M4 featuring the Microchip ATSAMD51. This...</td>
<td></td>
<td><a href="https://www.adafruit.com/product/3382">https://www.adafruit.com/product/3382</a></td>
</tr>
<tr>
<td><strong>Monochrome 1.3&quot; 128x64 OLED graphic display - STEMMA QT / Qwiic</strong></td>
<td>These displays are small, only about 1.3&quot; diagonal, but very readable due to the high contrast of an OLED display. This display is made of 128x64 individual white OLED pixels,...</td>
<td></td>
<td><a href="https://www.adafruit.com/product/938">https://www.adafruit.com/product/938</a></td>
</tr>
</tbody>
</table>
How MIDI is Transmitted

MIDI In vs. MIDI Out

The MIDI communication protocol has two main options: MIDI in and MIDI out. MIDI in refers to receiving MIDI messages as an input. MIDI out refers to sending MIDI messages as an output.

Each type requires a different circuit to send or receive the messages properly. MIDI controllers will have the ports labeled as such. For homemade devices, you will define your device as either being an input or an output.
MIDI Channels

MIDI messages are sent via channels. There are 16 available channels, numbered 1-16. The channels allow for devices to know what and who to listen to, similar to setting up a remote to control a specific television.

For example, a MIDI controller could send a C3 note and CC messages via channel 1 to a synthesizer setup to receive messages on channel 1. At the same time, that MIDI controller could send the same or different messages on channel 8 to a drum machine setup to receive messages on channel 8.

In code, these channels are numbered 0 to 15 because you are counting with zero-based numbering (https://adafruit.it/YFG). For example, real-world channel 1 is coded as channel 0, real-world channel 2 is coded as channel 1, etc.

In CircuitPython, your MIDI device’s channel that it is listening to or sending messages on is defined at the beginning of your code. It looks like this:

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

where `in_channel` and `out_channel` are holding the channel values.

In Arduino, it's a little different. In the `setup()`, the `begin()` function is called with `MIDI_CHANNEL_OMNI`. `MIDI_CHANNEL_OMNI` means that the code is able to listen for any of the 16 MIDI channels. It looks like this:

```c
MIDI_CREATE_DEFAULT_INSTANCE();

void setup()
{
    MIDI.begin(MIDI_CHANNEL_OMNI);
}
```

More discrete MIDI channel definitions can be defined when sending individual MIDI messages. Examples for this syntax are available on the MIDI Messages (https://adafruit.it/YFI) page of this guide.
Connections

DIN-5

The original MIDI connector is a MIDI jack, or DIN-5 connector. These are chunky, hearty connectors. Though they are becoming less common on newer devices, they are reliable and provide a straightforward connection to transmit MIDI data.
TRS-A

A newer MIDI connection standard that has been implemented is TRS-A, which utilizes a TRS audio jack to transmit MIDI data over UART. When this first began appearing, the connection was not standardized so older devices may require specialized cables for full compatibility.

The standardized pinout for TRS-A is as follows:

- MIDI Sink to TRS Tip
- MIDI Source to TRS Ring
- MIDI Shield to TRS Sleeve

This new standard allows for the reliability of a DIN-5 connector, but in a smaller and more common form factor.

Beware of TRS-B! TRS-B is a different TRS wiring that was used before TRS-A was standardized and can be found on various MIDI controllers.

For more information on TRS MIDI connections, check out this resource from audionerd

https://adafru.it/YFp

MIDI over UART

When you see a connection with a DIN-5 or TRS-A connector, the device is using MIDI over UART. MIDI over UART transmits MIDI messages over the TX and RX serial connections at a baud rate of 31250.

You can use MIDI over UART to create MIDI devices using older boards with microcontrollers that do not have support for direct USB communication, such as the Atmega328. Additionally, you can have boards communicate with each other over MIDI directly using UART.
Adafruit METRO 328 Fully Assembled - Arduino IDE compatible
We sure love the ATmega328 here at Adafruit, and we use them a lot for our own projects. The processor has plenty of GPIO, Analog inputs, hardware UART SPI and I2C,...
https://www.adafruit.com/product/50

USB MIDI

USB MIDI has become more common for MIDI controllers. These devices have a configuration descriptor allowing them to be plug and play. These do not have direct compatibility with MIDI over UART devices, and, as a result, would require a converter to communicate directly.

For MIDI over USB devices, you can use them with your computer to interface with music tech software. You can also use them with a USB MIDI Host, which allows for USB devices to interface with each other. There are aftermarket USB MIDI hosts available, as well as DIY options (you can even use a Raspberry Pi (https://adafruit.it/YFq)).

Many current microcontrollers have the ability to be used for DIY USB MIDI projects, including boards that are based on the ATSAMD21 (M0 Express), ATSAMD51 (M4 Express) and RP2040. All of these boards have support for both CircuitPython and Arduino.

Adafruit ItsyBitsy M0 Express - for CircuitPython & Arduino IDE
What's smaller than a Feather but larger than a Trinket? It's an Adafruit ItsyBitsy M0 Express! Small, powerful, with a rockin' ATSAMD21 Cortex M0...
https://www.adafruit.com/product/3727
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

Adafruit QT Py RP2040
What a cutie pie! Or is it... a QT Py? This diminutive dev board comes with one of our new favorite chip, the RP2040. It's been made famous in the new
https://www.adafruit.com/product/4900

BLE MIDI
One of the more unique ways to transmit MIDI is over Bluetooth with BLE MIDI. BLE MIDI allows for wireless MIDI communication over Bluetooth. It can be the perfect choice for some projects that benefit from not being tethered by cables.

If you want to build a BLE MIDI project, you'll want to use an nRF52840 Express-based board, which has support for BLE.

Adafruit CLUE - nRF52840 Express with Bluetooth LE
Do you feel like you just don't have a CLUE? Well, we can help with that - get a CLUE here at Adafruit by picking up this sensor-packed development board. We wanted to build some...
https://www.adafruit.com/product/4500
MIDI Messages

You've read how MIDI is transmitted, but what exactly are MIDI messages? This section will go through the most commonly used messages and show how to send them with either CircuitPython or Arduino.

For more information on the Arduino MIDI library, check out this documentation on GitHub.IO

https://adafruit.it/YFr

For more information on the CircuitPython MIDI library, check out this documentation on readthedocs

https://adafruit.it/YFs
NoteOn

This message begins playing a note. A MIDI NoteOn message sends the MIDI note number and the velocity of the note.

MIDI note numbers range from 0 to 127, where 21 is A0. Inspired Acoustics (https://adafruit.it/YFt) has a great table here showing the MIDI note numbers with their corresponding English note name, piano key number and pitch frequency.

CircuitPython

```python
midi.send(NoteOn(note, velocity))
```

Arduino

```cpp
MIDI.sendNoteOn(note, velocity, midi channel);
```

NoteOff

This stops a note from playing. If a NoteOff message is not sent after a NoteOn message, the note will continue playing forever. A MIDI NoteOff message sends the note number and the velocity of the note.

CircuitPython

```python
midi.send(NoteOff(note, velocity))
```

Arduino

```cpp
MIDI.sendNoteOff(note, velocity, midi channel);
```

ControlChange Messages

ControlChange (CC) messages are utility messages that can have a range of functionality. CC messages are usually what is sent when you turn a knob or move a slider on a MIDI controller. Some examples of common messages are modulation and
sustain. Each CC message has an assigned number. The MIDI Association has a resource listing CC messages [here](https://adafru.it/YFu).

A MIDI CC message contains the CC number and the value that you want to send to that control change. The value range will differ depending on the CC message, but usually have a range of **0 - 127**.

**CircuitPython**

```python
midi.send(ControlChange(CC, value))
```

**Arduino**

```c
MIDI.sendControlChange(CC, value, midi channel);
```

**PitchBend**

**PitchBend** is a special MIDI message, with a range of **0** to **16383**. Since pitch can be bent up or down, the midpoint (no pitch bend) is **8192**.

**CircuitPython**

```python
x = PitchBend(value)
midi.send(x)
```

**Arduino**

```c
MIDI.sendPitchBend(value, midi channel);
```

**Reading MIDI In**

You can receive MIDI data and have your microcontroller read the incoming MIDI message.

**CircuitPython**

```python
midi.receive()
```
Let's Build!

In the next section, you'll be able to look at six example projects that apply the MIDI knowledge in this guide. You can reference these projects for your own creations.

The first three projects are examples for sending MIDI out:

- Basic MIDI Keyboard
  - Send MIDI notes with button presses
- MIDI CC Control with Pots
  - Affect control change messages with potentiometers
- BLE MIDI Sequencer
  - Send arpeggios to a synth over BLE MIDI. Wireless melodies!

The last three projects are examples for receiving MIDI in:

- Receive and Display MIDI Messages
  - Using an LCD screen, view the MIDI messages being sent in real-time from a MIDI controller
• Receive MIDI Over UART and Send Over USB
  ◦ Convert a UART MIDI device to a USB MIDI device
• Control Motors with MIDI
  ◦ You can control a variety of peripherals with MIDI messages. Motors are common since you can create MIDI-controlled robotic instruments

Prerequisite Guides

If this is your first time using electronics or CircuitPython, you will want to browse these guides first to have a better background when trying out the MIDI example projects.

Welcome to CircuitPython! Learn Guide
https://adafruit.it/cpy-welcome

CircuitPython Essentials Learn Guide
https://adafruit.it/Bfr

Which CircuitPython Board is Right for You? Learn Guide
https://adafruit.it/YFv

Basic MIDI Keyboard
Being able to send note messages is one of the most popular MIDI projects. In this example, you'll see how to use button inputs to send `NoteOn` and `NoteOff` messages to a synth or DAW.

Below, a simple circuit is built using a Feather RP2040 microcontroller board wired to four buttons. The components and wires are seated in a breadboard (https://adafruit.it/Bgk) which facilitates connections via wires.

The Feather will need to have header pins soldered on. The Breadboard can help hold them while you do this.

**Circuit Diagram**

- **Button 1**
  - Pin 1 to Feather pin D5
  - Pin 4 to Feather GND
- **Button 2**
  - Pin 1 to Feather pin D6
  - Pin 4 to Feather GND
- **Button 3**
  - Pin 1 to Feather pin D9
  - Pin 4 to Feather GND
- **Button 4**
  - Pin 1 to Feather pin D10
  - Pin 4 to Feather GND

**Setup the Feather RP2040**

For this example, you'll be using the Feather RP2040. To make sure it is setup properly with CircuitPython, please follow the steps in the guide below.

[Feather RP2040 CircuitPython Setup](https://adafruit.it/YFw)

**CircuitPython Code**

Once you've finished setting up your Feather RP2040 with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.
To do this, click on the Download Project Bundle button in the window below. It will download as a zipped folder.

```python
import board
import digitalio
import usb_midi
import adafruit_midi
from adafruit_midi.note_on import NoteOn
from adafruit_midi.note_off import NoteOff

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

# midi note numbers
midi_notes = [60, 64, 67, 72]

# digital pins for the buttons
key_pins = [board.D5, board.D6, board.D9, board.D10]

# array for buttons
keys = []

# setup buttons as inputs
for key in key_pins:
    key_pin = digitalio.DigitalInOut(key)
    key_pin.direction = digitalio.Direction.INPUT
    key_pin.pull = digitalio.Pull.UP
    keys.append(key_pin)

# states for buttons
key0_pressed = False
key1_pressed = False
key2_pressed = False
key3_pressed = False

# array for button states
key_states = [key0_pressed, key1_pressed, key2_pressed, key3_pressed]

while True:

    # iterate through 4 buttons
    for i in range(4):
        inputs = keys[i]

        # if button is pressed...
        if not inputs.value and key_states[i] is False:
            # update button state
            key_states[i] = True
            # send NoteOn for corresponding MIDI note
            midi.send(NoteOn(midi_notes[i], 120))

        # if the button is released...
        if inputs.value and key_states[i] is True:
            # send NoteOff for corresponding MIDI note
            midi.send(NoteOff(midi_notes[i], 120))
            key_states[i] = False

```
Upload the Code and Libraries to the Feather RP2040

After downloading the Project Bundle, plug your Feather RP2040 into the computer's USB port. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called CIRCUITPY. Unzip the folder and copy the following items to the Feather RP2040's CIRCUITPY drive.

- lib folder
- code.py

Your Feather RP2040 CIRCUITPY drive should look like this after copying the lib folder and the code.py file.

How the CircuitPython Code Works

The MIDI note numbers assigned to each button are setup in the `midi_notes` array. When a key press is detected, a `NoteOn` message with the corresponding MIDI note number is sent. When a key is released, then a `NoteOff` message is sent.

If you wanted to change the notes or buttons, you would edit the `midi_notes` and `key_pins` array.

Usage

You can connect your Feather RP2040 via USB to either your computer or a USB MIDI host. Then, press the buttons to play the notes of a C major triad.

A DIY MIDI keyboard is a very popular project. You can add more buttons, change the notes or create a fun enclosure to customize this project further.
MIDI CC Control with Pots

Potentiometers (often times shortened to the word pots, which are resistors varied by turning a knob) are a common peripheral on MIDI controllers. They also provide a greater sense of control than clicking and dragging when using a mouse with a DAW. In this example, you'll see how to update control change (CC) messages with pots.

Circuit Diagram

- **Pot 1**
  - Pin 1 to QT Py RP2040 GND
  - Pin 2 to QT Py RP2040 A0
  - Pin 3 to QT Py RP2040 3V

- **Pot 2**
  - Pin 1 to QT Py RP2040 GND
  - Pin 2 to QT Py RP2040 A1
  - Pin 3 to QT Py RP2040 3V

- **Pot 3**
  - Pin 1 to QT Py RP2040 GND
  - Pin 2 to QT Py RP2040 A2
  - Pin 3 to QT Py RP2040 3V

- **Pot 4**
  - Pin 1 to QT Py RP2040 GND
  - Pin 2 to QT Py RP2040 A3
  - Pin 3 to QT Py RP2040 3V
Setup the QT Py RP2040

For this example, you'll be using the QT Py RP2040. To make sure it is setup properly with CircuitPython, please follow the steps in the guide below.

CircuitPython Code

Once you've finished setting up your QT Py RP2040 with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the Download Project Bundle button in the window below. It will download as a zipped folder.

```python
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

import board
import usb_midi
import adafruit_midi
import simpleio
from analogio import AnalogIn
from adafruit_midi.control_change import ControlChange
from adafruit_midi.pitch_bend import PitchBend

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

# potentiometer setup
mod_pot = AnalogIn(board.A0)
pitchDown_pot = AnalogIn(board.A1)
pitchUp_pot = AnalogIn(board.A2)
sus_pot = AnalogIn(board.A3)

# function to read analog input
def val(pin):
    return pin.value

# variables for last read value
# defaults to 0
# no pitchbend is 8192
mod_val2 = 0
pitchDown_val2 = 8192
pitchUp_val2 = 8192
sus_val2 = 0

while True:
    # map range of analog input to midi values
    # pitchbend range is 0 to 16383 with 8192 centered or no pitchbend
```
mod_val1 = round(simpleio.map_range(val(mod_pot), 0, 65535, 0, 127))
pitchDown_val1 = round(simpleio.map_range(val(pitchDown_pot), 0, 65535, 0, 8192))
pitchUp_val1 = round(simpleio.map_range(val(pitchUp_pot), 0, 65535, 8192, 16383))
sus_val1 = round(simpleio.map_range(val(sus_pot), 0, 65535, 0, 127))

# if modulation value is updated...
if abs(mod_val1 - mod_val2) > 2:
    # update mod_val2
    mod_val2 = mod_val1
    # create integer
    modulation = int(mod_val2)
    # create CC message
    modWheel = ControlChange(1, modulation)
    # send CC message
    midi.send(modWheel)

# pitchbend down value is updated...
if abs(pitchDown_val1 - pitchDown_val2) > 75:
    # update pitchDown_val2
    pitchDown_val2 = pitchDown_val1
    # create PitchBend message
    pitchDown = PitchBend(int(pitchDown_val2))
    # send PitchBend message
    midi.send(pitchDown)

# pitchbend up value is updated...
if abs(pitchUp_val1 - pitchUp_val2) > 75:
    # updated pitchUp_val2
    pitchUp_val2 = pitchUp_val1
    # create PitchBend message
    pitchUp = PitchBend(int(pitchUp_val2))
    # send PitchBend message
    midi.send(pitchUp)

# sustain value is updated...
if abs(sus_val1 - sus_val2) > 2:
    # update sus_val2
    sus_val2 = sus_val1
    # create integer
    sustain = int(sus_val2)
    # create CC message
    sustainPedal = ControlChange(64, sustain)
    # send CC message
    midi.send(sustainPedal)

Upload the Code and Libraries to the QT Py RP2040

After downloading the Project Bundle, plug your QT Py RP2040 into the computer's USB port. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called CIRCUITPY. Unzip the folder and copy the following items to the QT Py RP2040's CIRCUITPY drive.

- lib folder
- code.py
Your QT Py RP2040 CIRCUITPY drive should look like this after copying the lib folder and the code.py file.

![CIRCUITPY (E:) folder]

How the CircuitPython Code Works

The `map_range()` function is used to map the analog values from the potentiometers to MIDI values. The `_val1` and `_val2` variables are compared to each other for each pot to see if the pot has been turned. If it has, then an updated MIDI value is sent.

Usage

You can connect your QT Py RP2040 via USB to either your computer or a USB MIDI host. Then, turn the knobs to affect modulation, sustain and pitch bend. This project works best with a software or hardware synth since the effects generally affect musical input happening in real time.

Going further, you could combine this project with the components and code from the MIDI keyboard example to create a larger MIDI interface.
BLE MIDI Sequencer

BLE MIDI lets you send MIDI data wirelessly over Bluetooth. In this example, you'll use a CLUE to send arpeggios to a DAW.

Setup the CLUE

This example uses an Adafruit CLUE microcontroller board. To make sure it is setup properly with CircuitPython, please follow the steps in the guide below.

CircuitPython Code

Once you've finished setting up your CLUE with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the Download Project Bundle button in the window below. It will download as a zipped folder.

```python
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import adafruit_ble
import touchio
import board
import adafruit_midi
import adafruit_ble_midi
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
```
from adafruit_midi.note_on import NoteOn
from adafruit_midi.note_off import NoteOff

# CLUE cap touch setup
c_touch = touchio.TouchIn(board.D0)
f_touch = touchio.TouchIn(board.D1)
g_touch = touchio.TouchIn(board.D2)

# array of touch pads
pads = [c_touch, f_touch, g_touch]

# BLE MIDI setup
midi_service = adafruit_ble_midi.MIDIService()
advertisement = ProvideServicesAdvertisement(midi_service)

ble = adafruit_ble.BLERadio()
if ble.connected:
    for c in ble.connections:
        c.disconnect()

# midi setup
midi = adafruit_midi.MIDI(midi_out=midi_service, out_channel=0)

print("advertising")
ble.start_advertising(advertisement)

# MIDI note numbers for C, F and G major triads

c_triad = (60, 64, 67)
f_triad = (65, 69, 72)
g_triad = (67, 71, 74)

# array of triads
triads = [c_triad, f_triad, g_triad]

# touch debounce states

c_pressed = False
f_pressed = False
g_pressed = False

# array of debounce states
triad_states = [c_pressed, f_pressed, g_pressed]

# beginning triad
active_triad = c_triad

# variable for triad index
z = 0

while True:
    # BLE connection
    print("Waiting for connection")
    while not ble.connected:
        pass
    print("Connected")
    time.sleep(1.0)

    # while BLE is connected...
    while ble.connected:
        # iterate through the touch inputs
        for i in range(3):
            inputs = pads[i]
            # if a touch input is detected...
            if inputs.value and triad_states[i] is False:
                # debounce state activated
                triad_states[i] = True
                # update triad
                active_triad = triads[i]
                print(active_triad)
                # after touch input...
                if not inputs.value and triad_states[i] is True:
# reset debounce state
triad_states[i] = False

# send triad arpeggios out with half second delay
midi.send(NoteOn(active_triad[z]))
time.sleep(0.5)
midi.send(NoteOff(active_triad[z]))
time.sleep(0.5)

# increase index by 1
z += 1

# reset index at end of triad
if z > 2:
    z = 0

# BLE connection
print("Disconnected")
print()
ble.start_advertising(advertisement)

Upload the Code and Libraries to the CLUE

After downloading the Project Bundle, plug your CLUE into the computer's USB port. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called CIRCUITPY. Unzip the folder and copy the following items to the CLUE's CIRCUITPY drive.

- lib folder
- code.py

Your CLUE CIRCUITPY drive should look like this after copying the lib folder and the code.py file.
How the CircuitPython Code Works

There are three sequences of MIDI note numbers setup: `c_triad`, `f_triad` and `g_triad`. These correspond with the CLUE's three capacitive touch inputs.

In the loop, `NoteOn` and `NoteOff` messages are sent every 0.5 seconds to play through the triad. If a touch input is detected, then the triad updates.

The code is using BLE MIDI to utilize the CLUE's nRF52840. The loop is dependent on a BLE connection being active, otherwise the CLUE will wait for a connection.

Usage

You can connect your CLUE via BLE to your computer or mobile device. After a BLE connection is established, open your preferred music software to send the arpeggios to a software synth. Tap the three capacitive touch inputs to change the chord.

Going further, you could change the notes being sent and also play with the timing for sending the notes. You could use an additional input to code up a tap tempo to set the beats per minute (BPM) for sending notes.

Receive and Display MIDI Messages

Reading incoming MIDI messages can come in handy for troubleshooting, hacking or general curiosity. In this example, you'll see how to receive MIDI in messages and display them on an LCD screen.
Circuit Diagram

You can easily connect the LCD to the QT Py RP2040 with a STEMMA QT cable.

- LCD 3V to QT Py RP2040 3V
- LCD GND to QT Py RP2040 GND
- LCD SDA to QT Py RP2040 SDA
- LCD SCL to QT Py RP2040 SCL

Setup the QT Py RP2040

This example uses an QT Py RP2040. To make sure it is setup properly with CircuitPython, please follow the steps in the guide below.

QT Py RP2040 CircuitPython Setup

https://adafru.it/YFx

CircuitPython Code

Once you've finished setting up your QT Py RP2040 with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the Download Project Bundle button in the window below. It will download as a zipped folder.

```python
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

import board
import busio
import usb_midi
import adafruit_midi
import displayio
import terminalio
from adafruit_display_text import label
import adafruit_displayio_ssd1306
from adafruit_midi.control_change import ControlChange
from adafruit_midi.note_off import NoteOff
from adafruit_midi.note_on import NoteOn
from adafruit_midi.pitch_bend import PitchBend

displayio.release_displays()

oled_reset = board.D1
```
# I2C setup for display
# STEMMA I2C setup pre-CP 7.2
i2c = busio.I2C(board.SCL1, board.SDA1)

# STEMMA I2C setup for CP 7.2+
# i2c = board.STEMMA_I2C()
display_bus = displayio.I2CDisplay(i2c, device_address=0x3D, reset=oled_reset)

# midi setup
print(usb_midi.ports)
midi = adafruit_midi.MIDI(
    midi_in=usb_midi.ports[0], in_channel=0, midi_out=usb_midi.ports[1],
    out_channel=0
)
msg = midi.receive()

# display width and height setup
WIDTH = 128
HEIGHT = 64
BORDER = 5

# display setup
display = adafruit_displayio_ssd1306.SSD1306(display_bus, width=WIDTH,
    height=HEIGHT)
splash = displayio.Group()
display.show(splash)

# text area setup
text = "MIDI Messages"
text_area = label.Label(
    terminalio.FONT, text=text, color=0xFFFFFF, x=30, y=HEIGHT // 2+1)
splash.append(text_area)

while True:
    # receive midi messages
    msg = midi.receive()
    if msg is not None:
        # if a NoteOn message...
        if isinstance(msg, NoteOn):
            string_msg = 'NoteOn'
            string_val = str(msg.note)
        # if a NoteOff message...
        if isinstance(msg, NoteOff):
            string_msg = 'NoteOff'
            string_val = str(msg.note)
        # if a PitchBend message...
        if isinstance(msg, PitchBend):
            string_msg = 'PitchBend'
            string_val = str(msg.pitch_bend)
        # if a CC message...
        if isinstance(msg, ControlChange):
            string_msg = 'ControlChange'
            string_val = str(msg.control)
        # update text area with message type and value of message as strings
        text_area.text = (string_msg + " " + string_val)
Upload the Code and Libraries to the QT Py RP2040

After downloading the Project Bundle, plug your QT Py RP2040 into the computer's USB port. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called CIRCUITPY. Unzip the folder and copy the following items to the QT Py RP2040's CIRCUITPY drive.

- lib folder
- code.py

Your QT Py RP2040 CIRCUITPY drive should look like this after copying the lib folder and the code.py file.

How the CircuitPython Code Works

`msg` is acting as a variable to hold the incoming MIDI messages received with `midi.receive()`. Every time a new message comes in, the text on the screen is updated using `text_area.text`. The message values are converted to strings using `str()` so that they can be displayed.

Usage

You can connect your QT Py RP2040 via USB to either your computer or a USB MIDI host. Setup your software to send MIDI messages out and you will see the messages update in real time on the screen.
You can use this project to figure out what MIDI messages are being sent out, especially with ControlChange messages.

Receive MIDI Over UART and Send Over USB

In this example, you'll receive MIDI in over UART with the Adafruit MIDI FeatherWing, using a DIN-5 connector, to send the received MIDI messages out over USB with the Feather M4 Express. Basically, converting a UART MIDI device to use USB MIDI.

Plug the Feather RP2040 and the Adafruit MIDI FeatherWing into a FeatherWing Doubler.

For more information on the Adafruit MIDI FeatherWing, check out this guide

https://adafruit.it/YFy
Setup the Feather RP2040

For this example, you'll be using the Feather RP2040. To make sure it is setup properly with CircuitPython, please follow the steps in the guide below.

CircuitPython Code

Once you've finished setting up your Feather RP2040 with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the Download Project Bundle button in the window below. It will download as a zipped folder.

```python
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

import board
import busio
import adafruit_midi
import usb_midi
from adafruit_midi.control_change import ControlChange
from adafruit_midi.pitch_bend import PitchBend
from adafruit_midi.note_off import NoteOff
from adafruit_midi.note_on import NoteOn

# uart setup
uart = busio.UART(board.TX, board.RX, baudrate=31250)
# midi channel setup
midi_in_channel = 1
midi_out_channel = 1
# midi setup
# UART is setup as the input
# USB is setup as the output
midi = adafruit_midi.MIDI(
    midi_in=uart,
    midi_out=usb_midi.ports[1],
    in_channel=(midi_in_channel - 1),
    out_channel=(midi_out_channel - 1),
    debug=False,
)

print("MIDI UART In/USB Out")
print("Default output channel: ", midi.out_channel + 1)

# array of message types
messages = (NoteOn, NoteOff, PitchBend, ControlChange)

while True:
    # receive MIDI input from UART
    msg = midi.receive()
```
# if the input is a recognized message...
if msg is not None:
    for i in range(0, 3):
        # iterate through message types
        # makes it so that you aren't sending any unnecessary messages
        if isinstance(msg, messages[i]):
            # send the input out via USB
            midi.send(msg)
            print(msg)

Upload the Code and Libraries to the Feather RP2040

After downloading the Project Bundle, plug your Feather RP2040 into the computer's USB port. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called CIRCUITPY. Unzip the folder and copy the following items to the Feather RP2040's CIRCUITPY drive.

- lib folder
- code.py

Your Feather RP2040 CIRCUITPY drive should look like this after copying the lib folder and the code.py file.

How the CircuitPython Code Works

The MIDI FeatherWing's DIN-5 (or TRS) connections use MIDI over UART to transmit MIDI messages. Here, the code is setting up `midi_in` to use UART and `midi_out` to use the Feather's USB connection.

In the loop, when a recognized MIDI message is received, it is sent out via USB. As a result, you have a small form factor DIN-5 MIDI to USB MIDI converter.
Usage

You can connect your Feather via USB to either your computer or a USB MIDI host. Then, plug your DIN-5 hardware controller’s output to the MIDI FeatherWing’s input jack. When you send a MIDI message from your hardware controller, it should transmit via USB.

This essentially creates a simple DIN-5 to USB converter box. You could also do the reverse (taking MIDI in over USB and sending MIDI out over UART) if you wanted to send software MIDI to a hardware synth that did not have a USB connection.

Control Motors with MIDI

Musical robots controlled by incoming MIDI messages are a popular project. This example will show how you can use MIDI in messages to affect a servo motor.

Circuit Diagram

- Servo GND to Metro M4 Express GND
- Servo power to Metro M4 Express 5V
- Servo signal to Metro M4 Express pin D2
Setup the Metro M4 Express

This example uses a Metro M4 Express microcontroller board. To make sure it is setup properly with CircuitPython, please follow the steps in the guide below.

Metro M4 Express CircuitPython Setup

https://adafru.it/YFz

CircuitPython Code

Once you've finished setting up your Metro M4 Express with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the Download Project Bundle button in the window below. It will download as a zipped folder.

```python
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT
import board
import pwmio
import usb_midi
import adafruit_midi
from adafruit_midi.note_off import NoteOff
from adafruit_midi.note_on import NoteOn
from adafruit_motor import servo

# pwm setup for servo
pwm = pwmio.PWMOut(board.D2, duty_cycle=2 ** 15, frequency=50)

# servo setup
motor = servo.Servo(pwm)

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

while True:
    # receive midi input
    msg = midi.receive()

    if msg is not None:
        # if a NoteOn message is received...
        if isinstance(msg, NoteOn):
            # servo set to 180 degrees
            motor.angle = 180
        # if a NoteOff message is received...
        if isinstance(msg, NoteOff):
            # servo set to 0 degrees
            motor.angle = 0
```

© Adafruit Industries
Upload the Code and Libraries to the Metro M4 Express

After downloading the Project Bundle, plug your Metro M4 Express into the computer's USB port. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called CIRCUITPY. Unzip the folder and copy the following items to the Metro M4 Express's CIRCUITPY drive.

- lib folder
- code.py

Your Metro M4 Express CIRCUITPY drive should look like this after copying the lib folder and the code.py file.

How the CircuitPython Code Works

This code is an example of having another peripheral react to a MIDI input. Every time a `NoteOn` message is received, the servo turns to 180 degrees. When a `NoteOff` message is received, the servo turns back to 0 degrees.

Usage

You can connect your Metro M4 via USB to either your computer or a USB MIDI host. Send MIDI `NoteOn` and `NoteOff` messages to the Metro and you should see the servo move back and forth accordingly.

Musical robots are a popular music technology project that often incorporate MIDI. Using techniques like this can also be useful in interactive installations where folks can see a physical reaction to a digital input.
MIDI Projects on the Learn System

There are many fantastic MIDI projects on the Adafruit Learning System ([https://adafruit.it/YFA](https://adafruit.it/YFA)) for you to get further inspiration from. These projects are full builds compared to the examples in this guide, so you can get some ideas for enclosures, hardware and more advanced code.

**Raspberry Pi Pico and LED Arcade Button MIDI Controller ([https://adafruit.it/YFB](https://adafruit.it/YFB))**

This project uses 16 light-up arcade buttons to send MIDI note messages out. Additionally, it has a 5-way joystick and a screen to change the MIDI note numbers assigned to each button on the go.

**Trellis M4 Expressive MIDI Controller ([https://adafruit.it/EJU](https://adafruit.it/EJU))**

This project uses the Trellis M4 to have 32 button inputs for sending MIDI out. Additionally, it uses the built-in accelerometer to control MIDI parameters by panning and tilting the board.

**MIDI Solenoid Drummer ([https://adafruit.it/YFC](https://adafruit.it/YFC))**

This project sends MIDI out to control solenoid motors. The solenoids are setup to hit percussive objects to create a robotic percussion instrument.

**MIDI Melody Maker ([https://adafruit.it/YFD](https://adafruit.it/YFD))**

This project sends MIDI note sequences out. There are parameters for changing the note pattern, key and note value. Additionally, there is a slider that affects the BPM, or the speed, at which notes are sent out.

**Power Glove Wireless MIDI Controller ([https://adafruit.it/YFE](https://adafruit.it/YFE))**

This project uses BLE MIDI to modify a classic Nintendo Power Glove to be a wireless MIDI controller. Flex sensors in the fingers are used to affect the values of the MIDI messages.