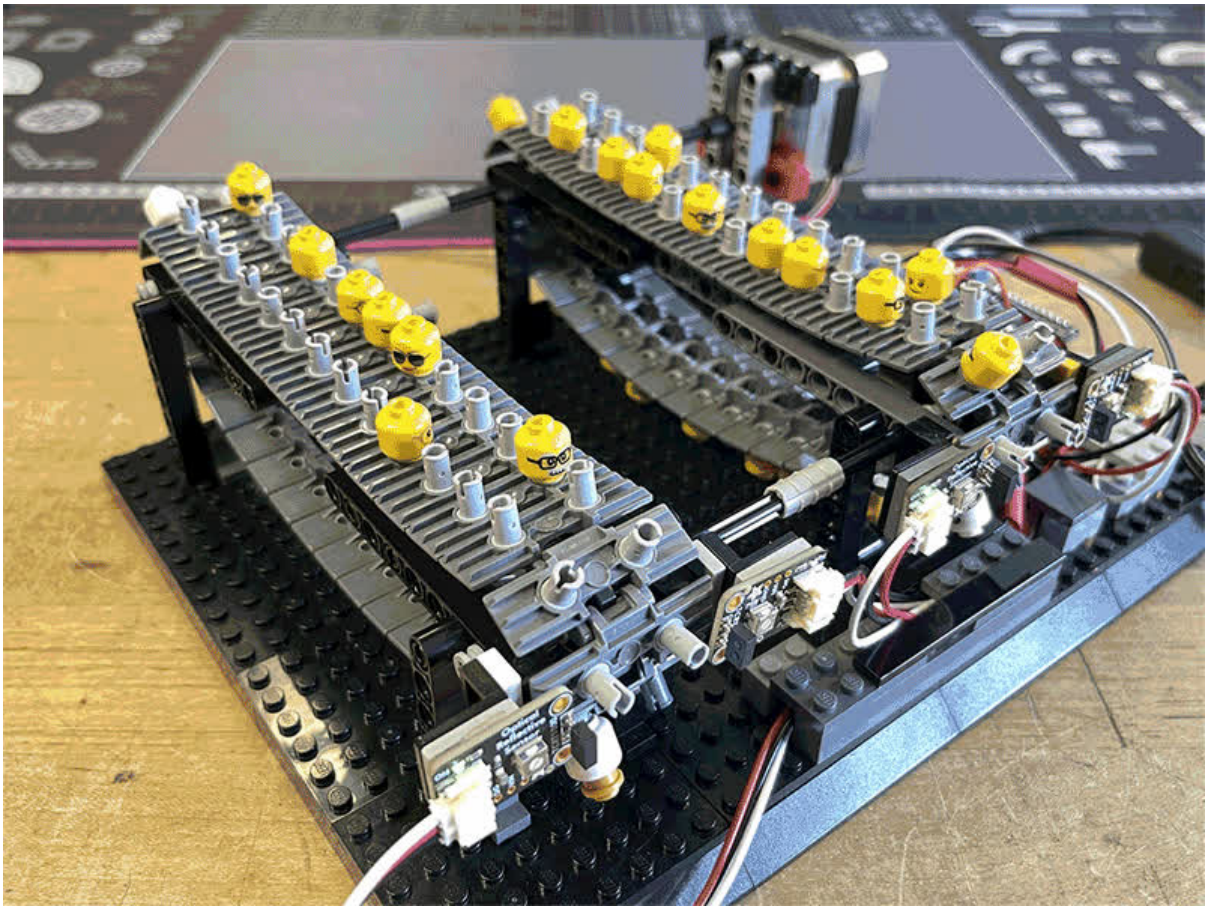
Optical Sensor Drum Track Sequencer

Created by John Park



https://learn.adafruit.com/drum-track-sequencer

Last updated on 2024-06-11 12:39:15 PM EDT

Table of Contents

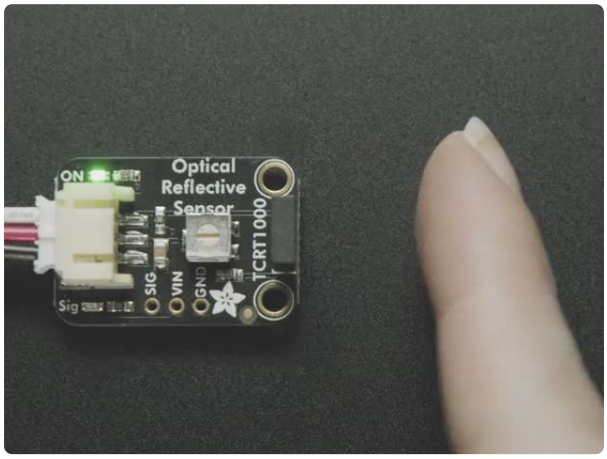
Overview

Build a physical drum sequencer so you can program beats right here in the real world! Four tracks of drums (say kick, snare, clap, cowbell, but it's your choice) and 32 steps for two four-bar patterns can trigger your favorite drum synthesizer or sample player.

Four optical reflection sensors detect LEGO minifigure heads as they roll on by, all driven by a Feather RP2040 with Motor FeatherWing running CircuitPython.

Parts

Four sensors:

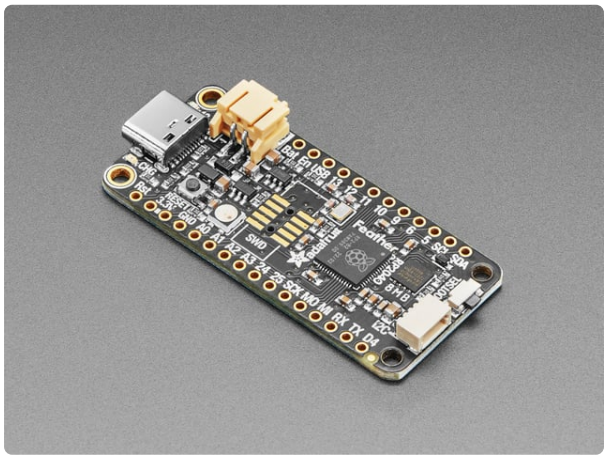


Adafruit STEMMA Reflective Photo Interrupt Sensor - TCRT1000

An optical reflective sensor is a composite electronic device with two elements - an IR LED and an IR photo-transistor. The IR LED blasts light, and when something bounces...

https://www.adafruit.com/product/5913

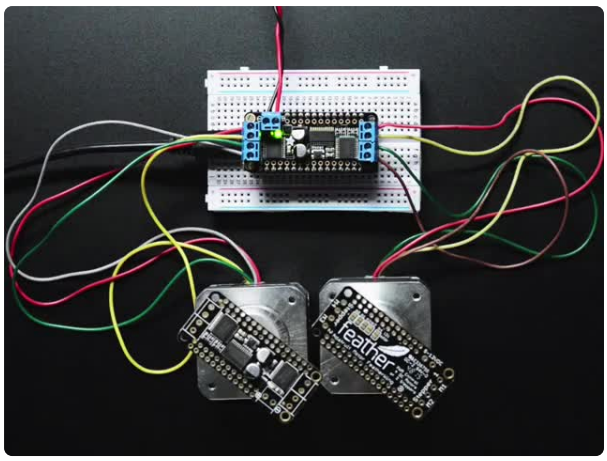
Then these:



Adafruit Feather RP2040

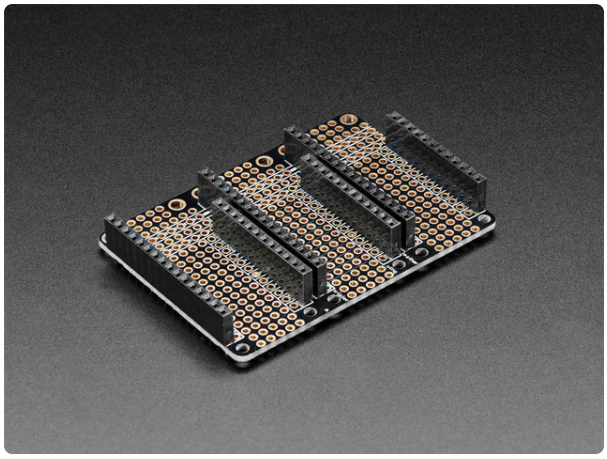
A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought "this chip is going to be awesome when we give it the Feather...

https://www.adafruit.com/product/4884



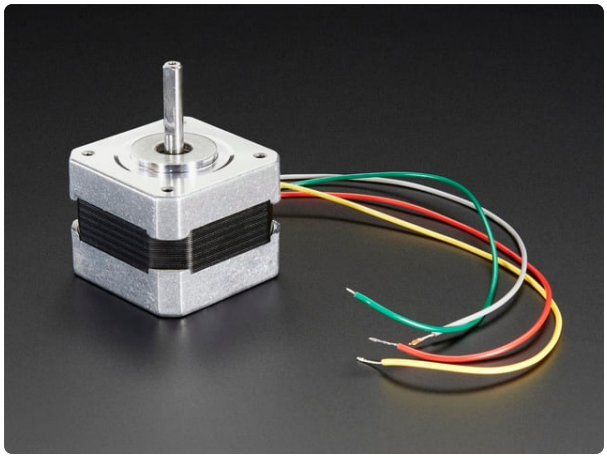
DC Motor + Stepper FeatherWing Add-on For All Feather Boards

A Feather board without ambition is a Feather board without FeatherWings! This is the DC Motor + Stepper FeatherWing which will let you use 2 x bi-polar...
https://www.adafruit.com/product/2927



FeatherWing Tripler Mini Kit - Prototyping Add-on For Feathers

This is the FeatherWing Tripler - a prototyping add-on and more for all Feather boards. This is similar to our
https://www.adafruit.com/product/3417



Stepper motor - NEMA-17 size - 200 steps/rev, 12V 350mA

A stepper motor to satisfy all your robotics needs! This 4-wire bipolar stepper has 1.8° per step for smooth motion and a nice holding torque. The motor was specified to have a max...
https://www.adafruit.com/product/324

1 x DC Gearbox "TT" Motor to LEGO® compatible Cross Axle — https://www.adafruit.com/product/3810

DC Gearbox "TT" Motor to LEGO® compatible Cross Axle

---

1 x 12V 5A switching power supply — https://www.adafruit.com/product/352

12VDC from 110VAC or 220VAC

---

4 x STEMMA JST PH 2mm 3-Pin to Male Header Cable — https://www.adafruit.com/product/3893

200mm

---

20 x 6"

1 x Premium Female/Male 'Extension' Jumper Wires https://www.adafruit.com/product/1954
20 x 6"

---

1 x In-line power switch https://www.adafruit.com/product/1125
for 2.1mm barrel jack

---

1 x Female DC Power adapter https://www.adafruit.com/product/368
2.1mm jack to screw terminal block

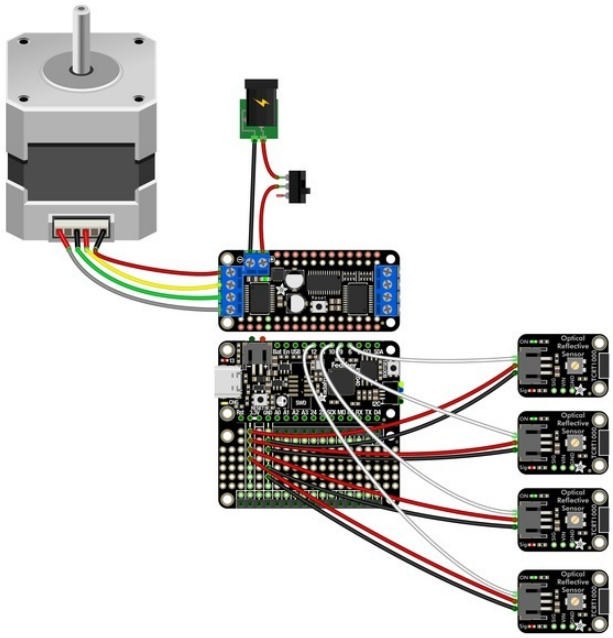
---

1 x Clear Adhesive Squares https://www.adafruit.com/product/4813
6 pack - UGlu Dashes

---

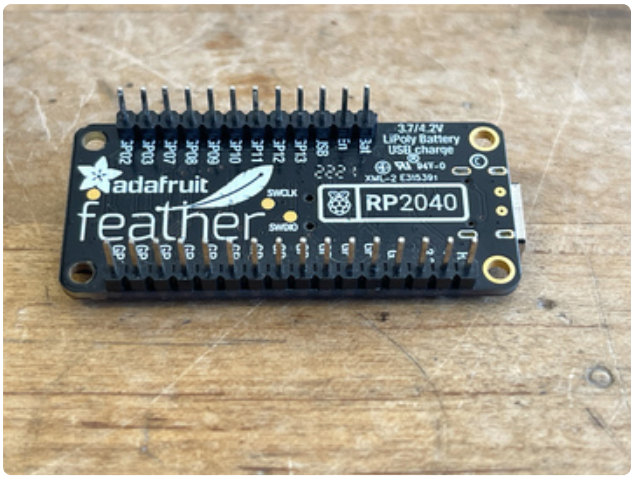
1 x 20-pin 0.1" Female Headers https://www.adafruit.com/product/4160
Rainbow Color Mix - 5 pack

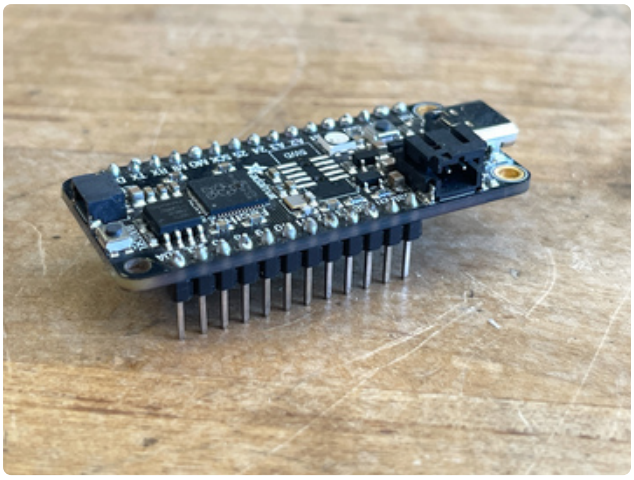
Build the Circuit



The circuit consists of four optical sensors running to the Feather's pins 12, 10, 9, and 6, and a Motor FeatherWing to drive the stepper motor. A 12VDC power supply runs into the FeatherWing via an inline switch for easy pause/play action.

Feather Prep

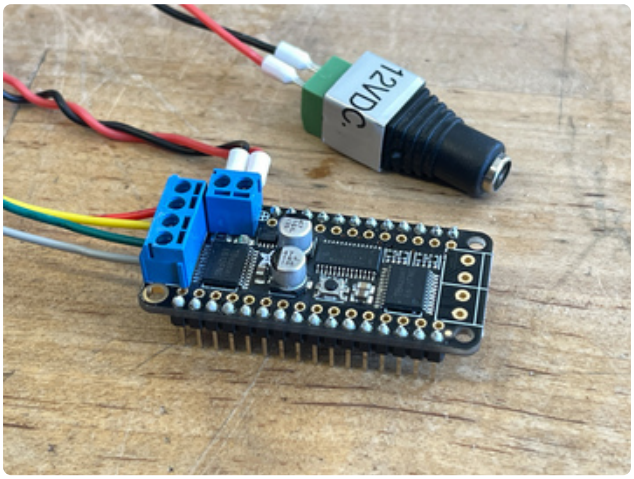




Solder male header pins underneath the Feather as shown here. These will be used to connect the Feather to the FeatherWing Tripler.

This guide (https://adafru.it/Rhd) covers the details of Feather pin soldering in depth.

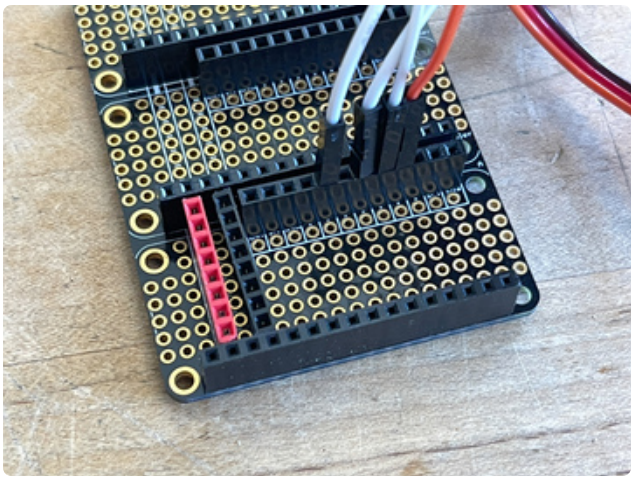
Motor FeatherWing Prep



Solder in the Terminal blocks and male header pins under the Motor FeatherWing (unless you bought the pre-soldered version).

Use two short lengths of wire to connect the DC power breakout **positive voltage** + to Motor FeatherWing **power input** + and the DC power breakout **ground (-)** to Motor FeatherWing **power input ground**.

Feather Tripler Prep



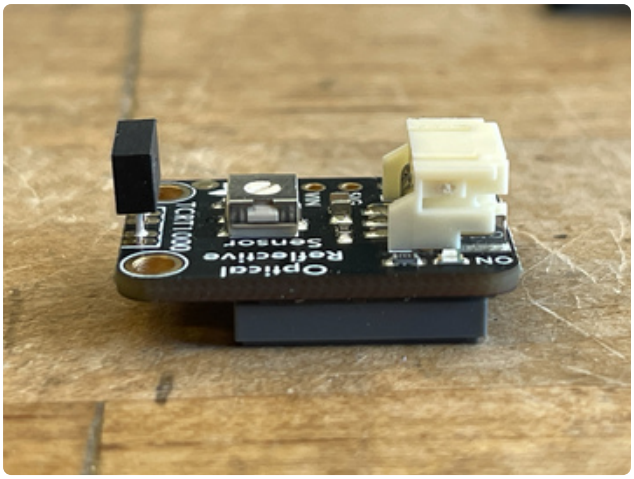
Assemble the FeatherWing Tripler in the usual way (as shown here) with female header sockets. Then, add an additional strip for the 3.3V and GND strips as shown here. These will make it easy to plug in the four optical sensors without running out of power/ground sockets.

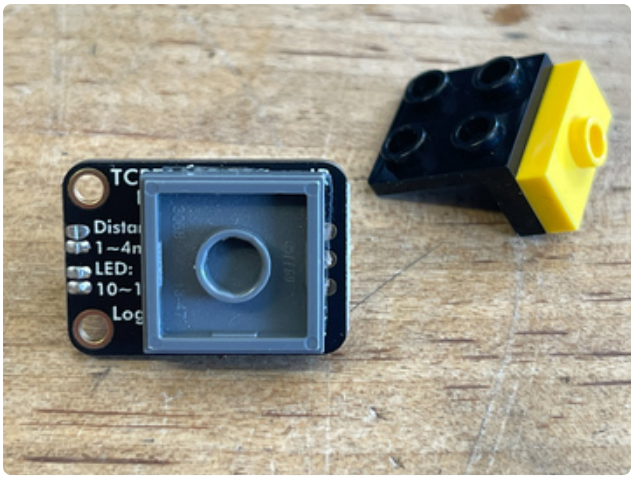
If you want to be fancy (and who doesn't), use color coded headers (http://adafru.it/4160)!

Optical Reflection Sensors Prep

An optical reflective sensor is a composite electronic device with two elements - an IR LED and an IR photo-transistor. The IR LED blasts light, and when something bounces the light back to the photo-transistor, the transistor turns on and the amount of current flowing through it increases.

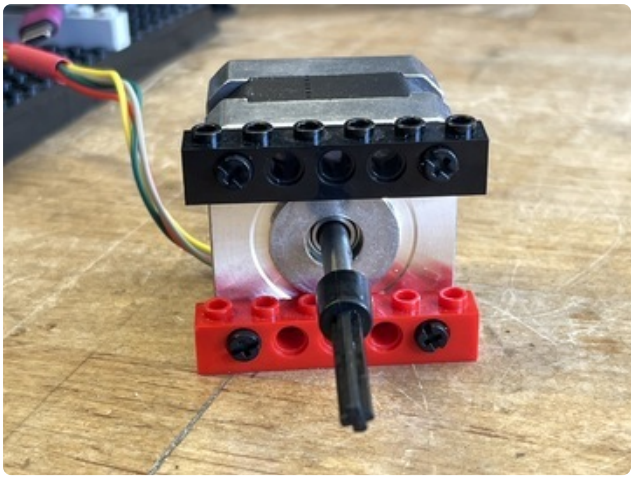
This makes the sensor great at detecting when something is in front of the sensor, such as a LEGO head to trigger a MIDI drum! when nothing is there the IR light never gets reflected back and the transistor stays off.

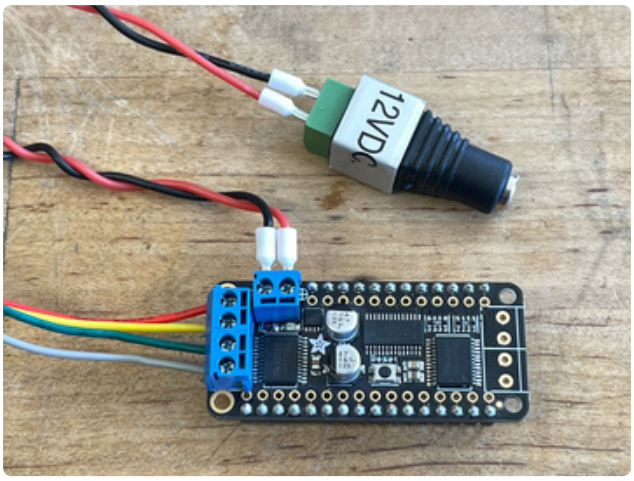




Use a Uglu square to affix a LEGO 2x2 tile under each reflection sensor breakout as shown here.

Prep the Stepper Motor



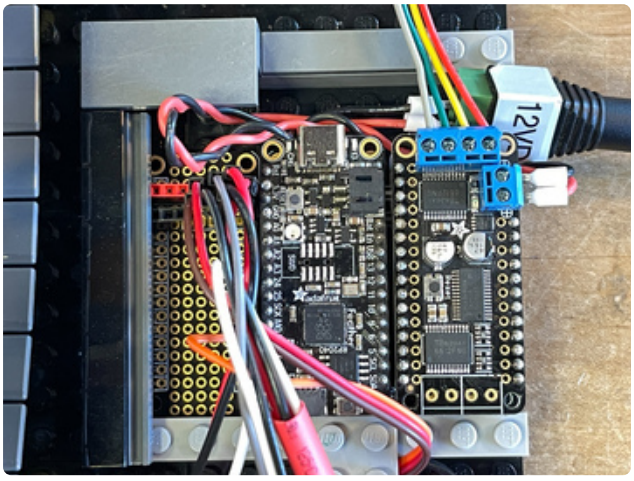


Use 4 M2.5 x 10mm nylon screws to connect two LEGO Technic 1x6 with holes bricks as shown.

Press fit the shaft to axle adapter -- this will probably require the removal of a bit of plastic inside the coupling shaft -- a hobby knife works pretty well for this.

Then, screw the four wires into the Motor FeatherWing terminal blocks as shown.

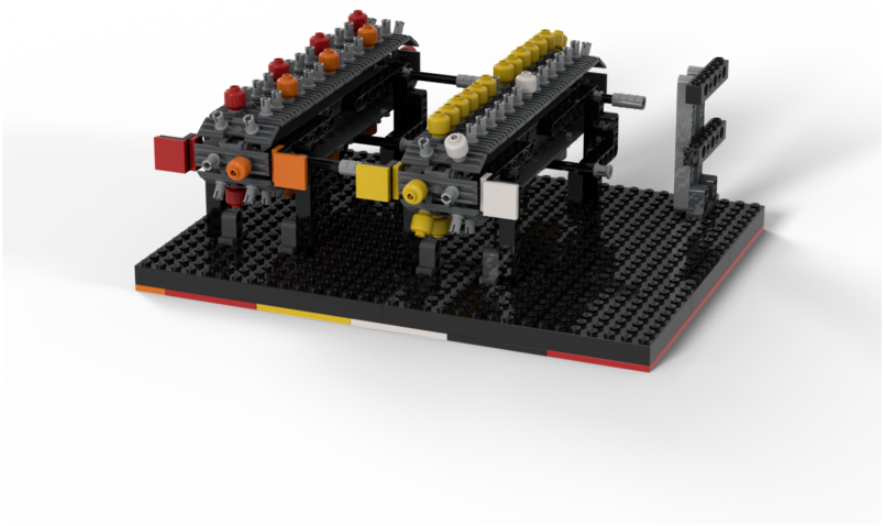
Full Circuit



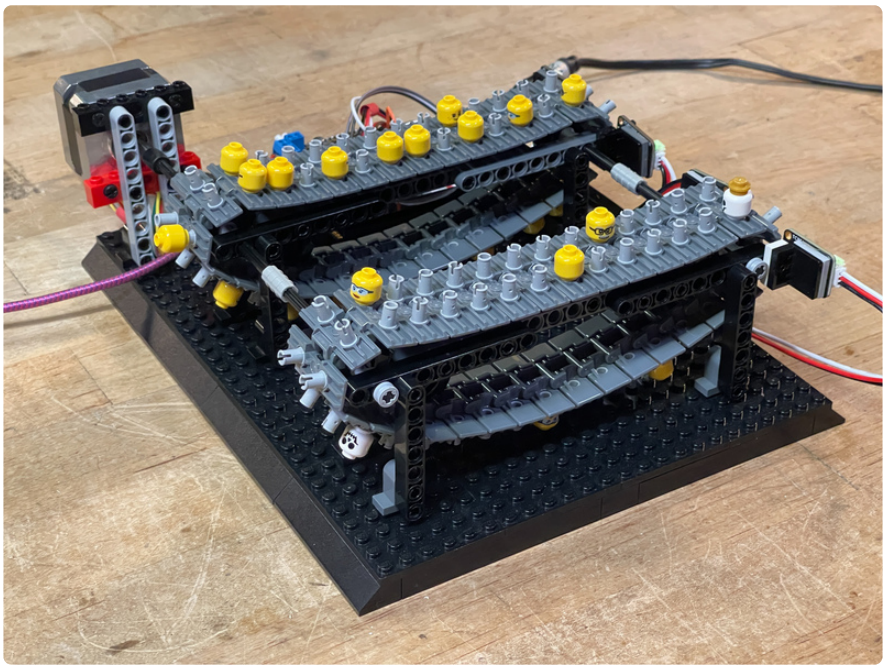
Assemble the parts by pressing the Motor FeatherWing and the Feather RP2040 into the Feather Tripler as shown.

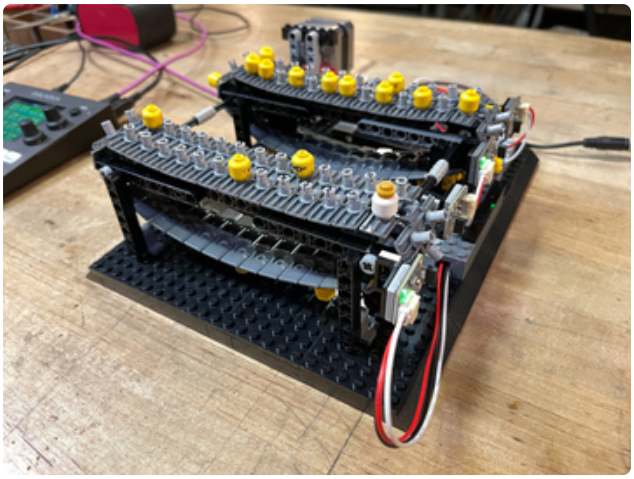
Then, use the JST connectors to connect the four optical reflection sensor breakouts to the power and ground strips and Feather pins 12, 10, 9, and 6.

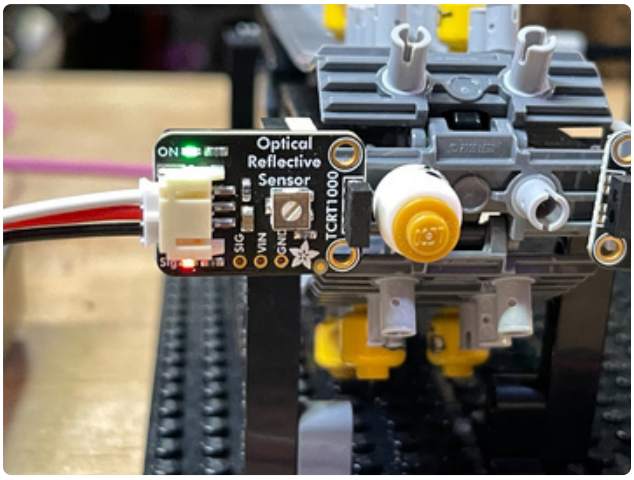
Build the Sequencer

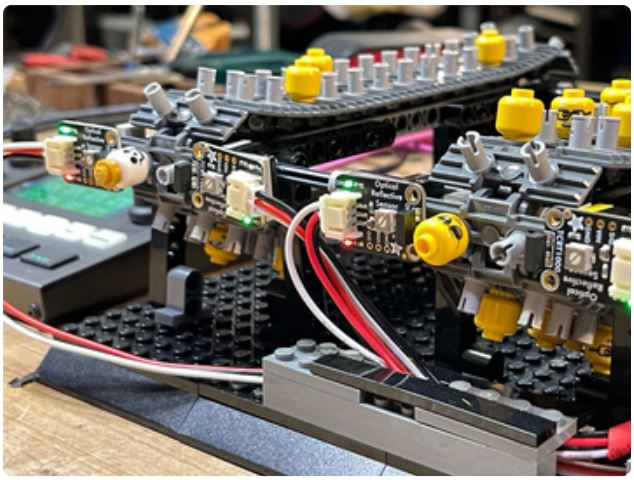


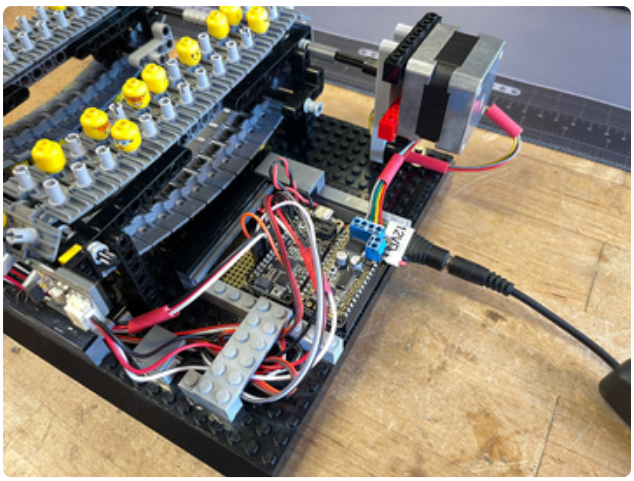
To build the LEGO portion of the sequencer, you can use the instructions found here (https://adafru.it/1a2d) which includes step-by-step build instructions and a 3D model you can spin around to inspect at each step.





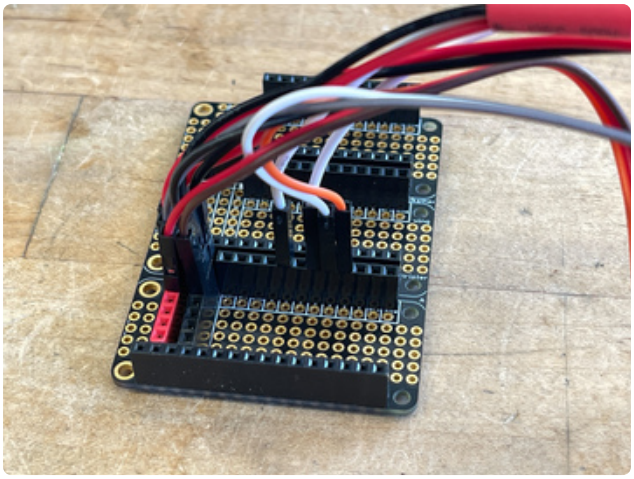






Optical Sensors

Plug in the JST cables to the four optical sensor breakouts. The two further ones can be extended with jumper cables so they reach the board.

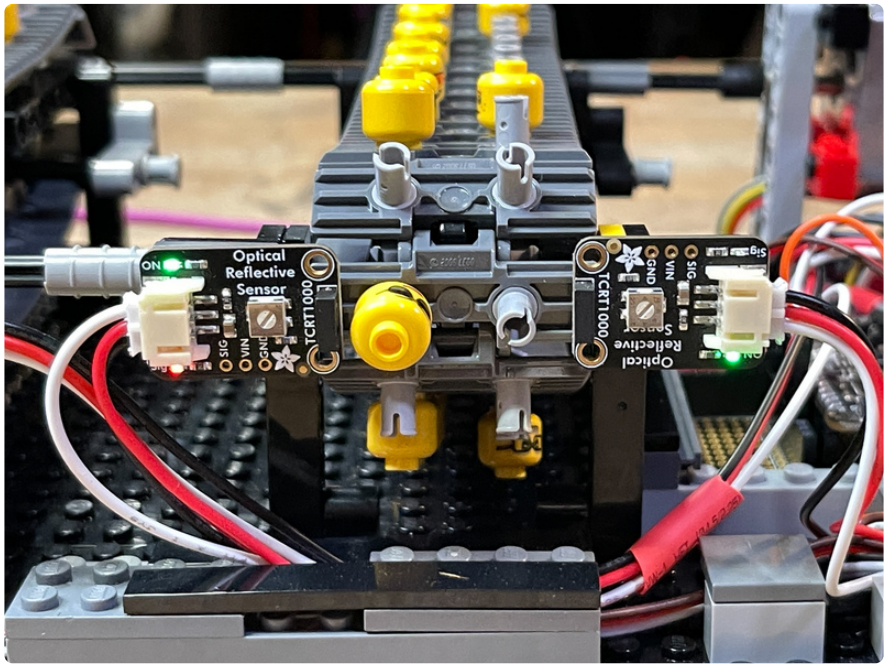


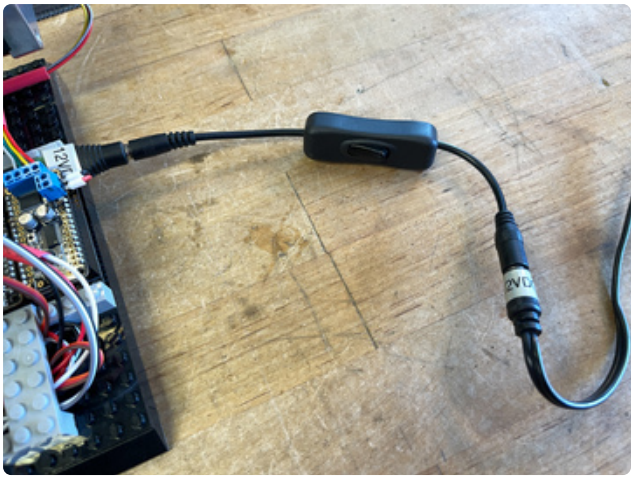
Sensor Wiring

Here you can see the isolated FeatherWing Tripler with the added power and ground rails. Each sensor plugs into power and ground, and then the GPIO pins 12, 10, 9, & 6.

Triggers

You can see in the image below, when a head is in front of the optical sensor, it lights up the Signal LED and sends the trigger signal to the Feather. If necessary, use a small slotted screwdriver to adjust the trimmer pot until the heads trigger reliably.





Power Switch

Insert the inline power switch between the 12VDC power supply and the Motor FeatherWing socket extension. This will allow you to cut power to the stepper motor.

Install CircuitPython

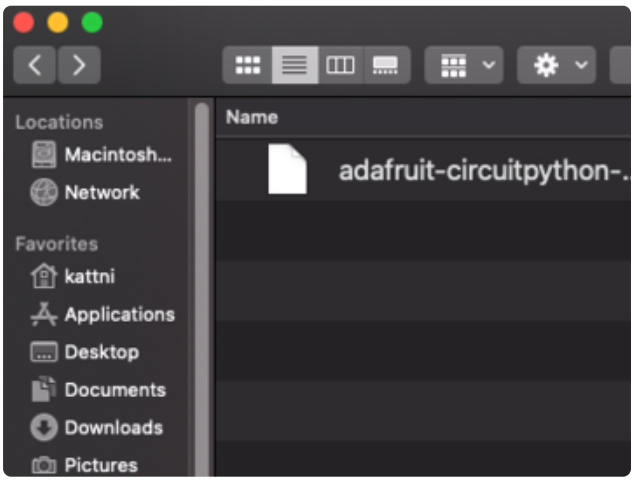
CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the **CIRCUITPY** drive to iterate.

CircuitPython Quickstart

Follow this step-by-step to quickly get CircuitPython running on your board.

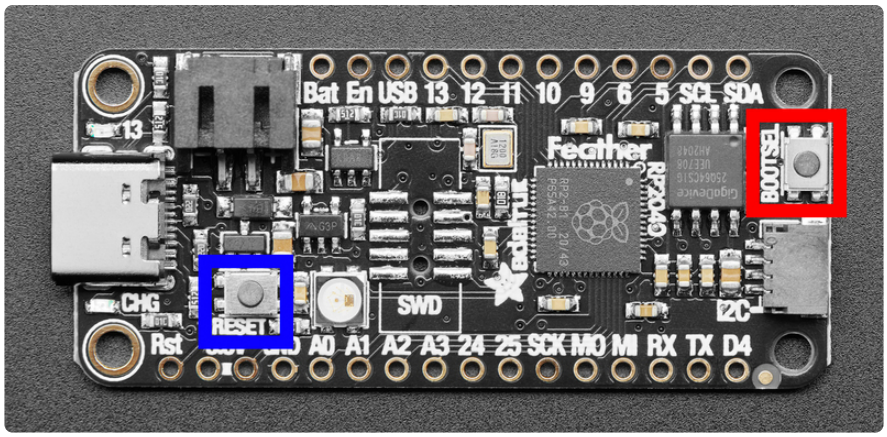
Download the latest version of CircuitPython for this board via circuitpython.org

https://adafru.it/R1D



Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.

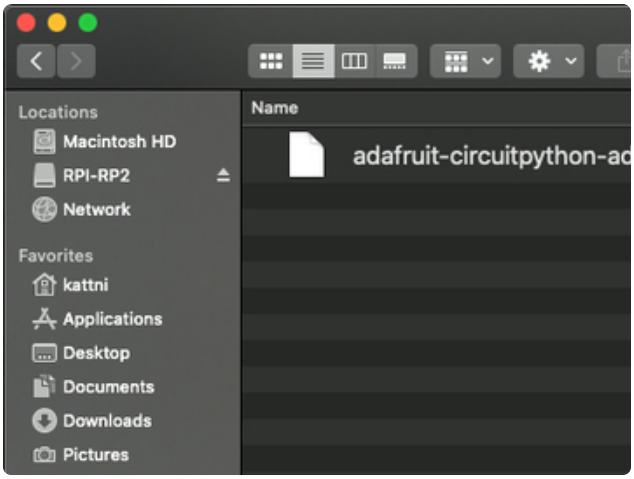


To enter the bootloader, hold down the **BOOT/BOOTSEL button** (highlighted in red above), and while continuing to hold it (don't let go!), press and release the **reset button** (highlighted in blue above). **Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!**

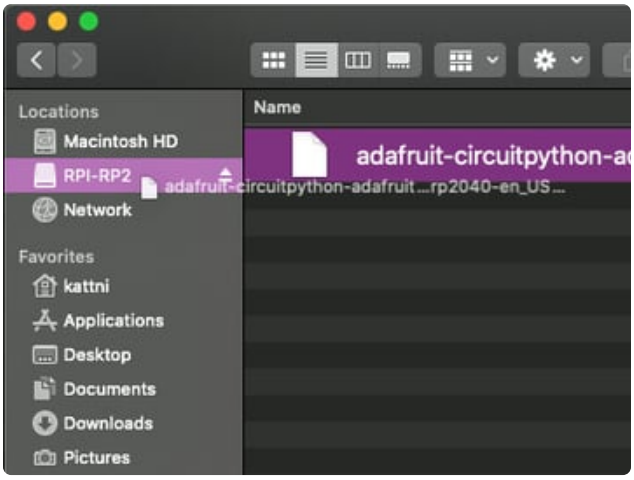
If the drive does not appear, release all the buttons, and then repeat the process above.

You can also start with your board unplugged from USB, press and hold the BOOTSEL button (highlighted in red above), continue to hold it while plugging it into USB, and wait for the drive to appear before releasing the button.

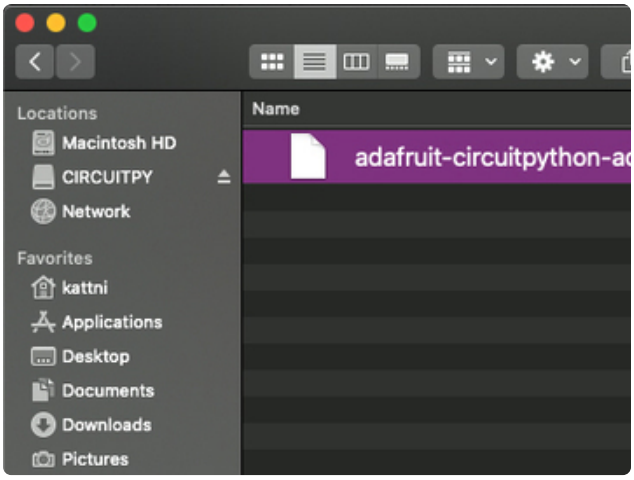
A lot of people end up using charge-only USB cables and it is very frustrating! **Make sure you have a USB cable you know is good for data sync.**



You will see a new disk drive appear called **RPI-RP2**.



Drag the **adafruit_circuitpython_etc.uf2** file to **RPI-RP2.**



The **RPI-RP2** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

That's it, you're done! :)

Safe Mode

You want to edit your **code.py** or modify the files on your **CIRCUITPY** drive, but find that you can't. Perhaps your board has gotten into a state where **CIRCUITPY** is read-only. You may have turned off the **CIRCUITPY** drive altogether. Whatever the reason, safe mode can help.

Safe mode in CircuitPython does not run any user code on startup, and disables auto-reload. This means a few things. First, safe mode bypasses any code in **boot.py** (where you can set **CIRCUITPY** read-only or turn it off completely). Second, it does not run the code in **code.py**. And finally, it does not automatically soft-reload when data is written to the **CIRCUITPY** drive.

Therefore, whatever you may have done to put your board in a non-interactive state, safe mode gives you the opportunity to correct it without losing all of the data on the **CIRCUITPY** drive.

Entering Safe Mode

To enter safe mode when using CircuitPython, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 1000ms. On some boards, the onboard status LED (highlighted in green above) will blink yellow during that time. If you press reset during that 1000ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

In Safe Mode

If you successfully enter safe mode on CircuitPython, the LED will intermittently blink yellow three times.

If you connect to the serial console, you'll find the following message.

```
Auto-reload is off.
Running in safe mode! Not running saved code.

CircuitPython is in safe mode because you pressed the reset button during boot.
Press again to exit safe mode.

Press any key to enter the REPL. Use CTRL-D to reload.
```

You can now edit the contents of the **CIRCUITPY** drive. Remember, your code will not run until you press the reset button, or unplug and plug in your board, to get out of safe mode.

Flash Resetting UF2

If your board ever gets into a really weird state and CIRCUITPY doesn't show up as a disk drive after installing CircuitPython, try loading this 'nuke' UF2 to RPI-RP2. which

will do a 'deep clean' on your Flash Memory. **You will lose all the files on the board**, but at least you'll be able to revive it! After loading this UF2, follow the steps above to re-install CircuitPython.

Download flash erasing "nuke" UF2

https://adafru.it/RLE

Code the Sequencer

Text Editor

Adafruit recommends using the **Mu** editor for editing your CircuitPython code. You can get more info in this guide (https://adafru.it/ANO).

Alternatively, you can use any text editor that saves simple text files.

Download the Project Bundle

Your project will use a specific set of CircuitPython libraries and the **code.py** file. To get everything you need, click on the **Download Project Bundle** link below, and uncompress the .zip file.

Drag the contents of the uncompressed bundle directory onto your board's **CIRCUITPY** drive, replacing any existing files or directories with the same names, and adding any new ones that are necessary.

```
# SPDX-FileCopyrightText: 2024 John Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT
"""
Drum Track Sequencer
 Feather RP2040, Motor FeatherWing, stepper motor,
 four reflection sensors, USB MIDI out
"""
import asyncio
import busio
import board
from adafruit_motorkit import MotorKit
from adafruit_motor import stepper
import keypad
import usb_midi

# Tempo setup
BPM = 100  # user set value
tempo_table = {  # motor speed seems non-linear, so we'll use a lookup table
    110: 0.0004,
    100: 0.001,
    90: 0.002,
    80: 0.003,
```

```
    75: 0.004,
    65: 0.005,
    60: 0.006,
    50: 0.008
}
def get_nearest_tempo(given_bpm):
    nearest_table_item = min(tempo_table.keys(), key=lambda k: abs(k - given_bpm))
    return tempo_table[nearest_table_item]
motor_pause = get_nearest_tempo(BPM)

i2c=busio.I2C(board.SCL, board.SDA, frequency=400_000)

# Motor setup
kit = MotorKit(i2c=i2c)
motor_run=True

# Sensor setup
optical_pins = (board.D6, board.D9, board.D10, board.D12)
optical_sensors = keypad.Keys(optical_pins, value_when_pressed=False, pull=True)

# MIDI setup
midi = usb_midi.ports[1]
midi_notes = (36, 37, 38, 39)  # typical drum voice notes

def play_drum(note):
    midi_msg_on = bytearray([0x99, note, 120])  # 0x90 noteOn ch1, 0x99 noteOn ch10
    midi_msg_off = bytearray([0x89, note, 0])
    midi.write(midi_msg_on)
    midi.write(midi_msg_off)

async def check_sensors():
    while True:
        optical_sensor = optical_sensors.events.get()
        if optical_sensor:
            if optical_sensor.pressed:
                track_num = optical_sensor.key_number
                # print("tripped", track_num)
                play_drum(midi_notes[track_num])
        await asyncio.sleep(0.008)  # don't check sensors constantly or motor speed
reduced

async def run_motor():
    while True:
        kit.stepper1.onestep(
                                direction=stepper.BACKWARD,
                                style=stepper.DOUBLE
        )
        await asyncio.sleep(motor_pause)  # motor speed-- smaller numbers are faster

async def main():
    motor_task = asyncio.create_task(run_motor())
    sensor_task = asyncio.create_task(check_sensors())
    await asyncio.gather(motor_task, sensor_task)

asyncio.run(main())
```

How It Works

The code's job is to run the stepper motor at a specific speed and read the four optical sensor inputs. When an input is triggered a MIDI note is sent over USB MIDI.

This code checks the sensor inputs while the stepper motor rotates at a tempo-defined speed. By combining asynchronous tasks, then it efficiently checks sensors and controls the motor simultaneously.

Libraries

First, the following libraries are imported:

```
import asyncio
import busio
import board
from adafruit_motorkit import MotorKit
from adafruit_motor import stepper
import keypad
import usb_midi
```

Constants

the `BPM` variable is an integer for storing the tempo. It is user defined and works along with the `tempo_table` dictionary to set the stepper motor step delay.

```
BPM = 100  # user set value
tempo_table = {
    110: 0.0004,
    100: 0.001,
    90: 0.002,
    80: 0.003,
    75: 0.004,
    65: 0.005,
    60: 0.006,
    50: 0.008
}
```

Tempo Calculation

The `get_nearest_tempo()` function returns the `motor_pause` value in milliseconds.

```
def get_nearest_tempo(given_bpm):
    nearest_table_item = min(tempo_table.keys(), key=lambda k: abs(k - given_bpm))
    return tempo_table[nearest_table_item]
motor_pause = get_nearest_tempo(BPM)
```

Setup

Next, the I2C bus, motor, sensor, and MIDI setup:

```
i2c = busio.I2C(board.SCL, board.SDA, frequency=400_000)

# Motor setup
kit = MotorKit(i2c=i2c)
motor_run = True

# Sensor setup
optical_pins = (board.D6, board.D9, board.D10, board.D12)
optical_sensors = keypad.Keys(optical_pins, value_when_pressed=False, pull=True)

# MIDI setup
midi = usb_midi.ports[1]
midi_notes = (36, 37, 38, 39)  # typical drum voice notes
```

Play Drum Function

This function will send the note on and note off messages when a drum track beat is triggered.

```
def play_drum(note):
    midi_msg_on = bytearray([0x99, note, 120])  # 0x90 noteOn ch1, 0x99 noteOn ch10
    midi_msg_off = bytearray([0x89, note, 0])
    midi.write(midi_msg_on)
    midi.write(midi_msg_off)
```

async Functions

We need the motor control and sensor checking to happen at the same time, otherwise there would be a mess of tempo shifting! The asyncio library allows for asynchronous functions to be defined and then run effectively at the same time without getting in each others way. You can find out much more here (https://adafru.it/ZwB).

```
async def check_sensors():
    while True:
        optical_sensor = optical_sensors.events.get()
        if optical_sensor:
            if optical_sensor.pressed:
                track_num = optical_sensor.key_number
                play_drum(midi_notes[track_num])
        await asyncio.sleep(0.008)  # don't check sensors constantly or motor speed
reduced
```

```
async def run_motor():
    while True:
        kit.stepper1.onestep(
            direction=stepper.BACKWARD,
            style=stepper.DOUBLE
        )
        await asyncio.sleep(motor_pause)  # motor speed-- smaller numbers are faster
```

Then, the `async main()` function is created and run.

```
async def main():
    motor_task = asyncio.create_task(run_motor())
    sensor_task = asyncio.create_task(check_sensors())
    await asyncio.gather(motor_task, sensor_task)

asyncio.run(main())
```

Use the Sequencer

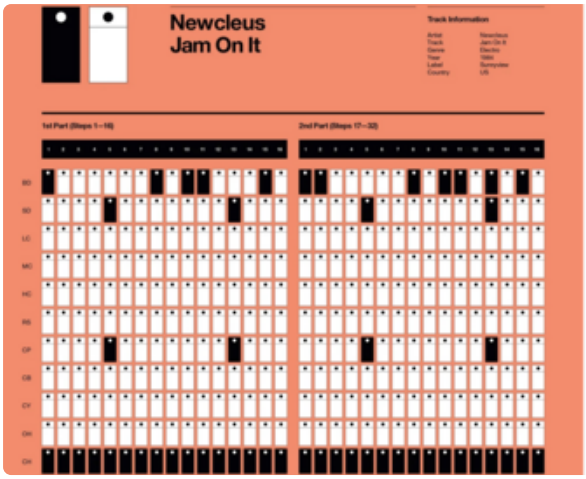
Now, you can fire up a drum synthesizer or sample player on your computer, such as Ableton Live, Garage Band, the free, open source VCV Rack (https://adafru.it/C-b) which runs on Windows, Mac, and Linux machines, or even this WebMIDI DrumKit (https://adafru.it/1a2n) by Adafruit community member Sam Blenny!

Here's an example VCV Rack patch that uses the four incoming MIDI notes from the track sequencer to trigger four drum sample modules.

trackseq.vcv

https://adafru.it/1a2e

Check out the video above for a demonstration on using your Drum Track Sequencer.



Drum Patterns

Try out some classic drum machine patterns such as those found in the classic Hal Leonard book **200 Drum Machine Patterns** (https://adafru.it/1a2f)

Or, have a look at these from a series of posters by Rob Ricketts. (https://adafru.it/1a2g)

Pick four key tracks and follow their patterns, such as bass drum, snare, closed hats, and clap.