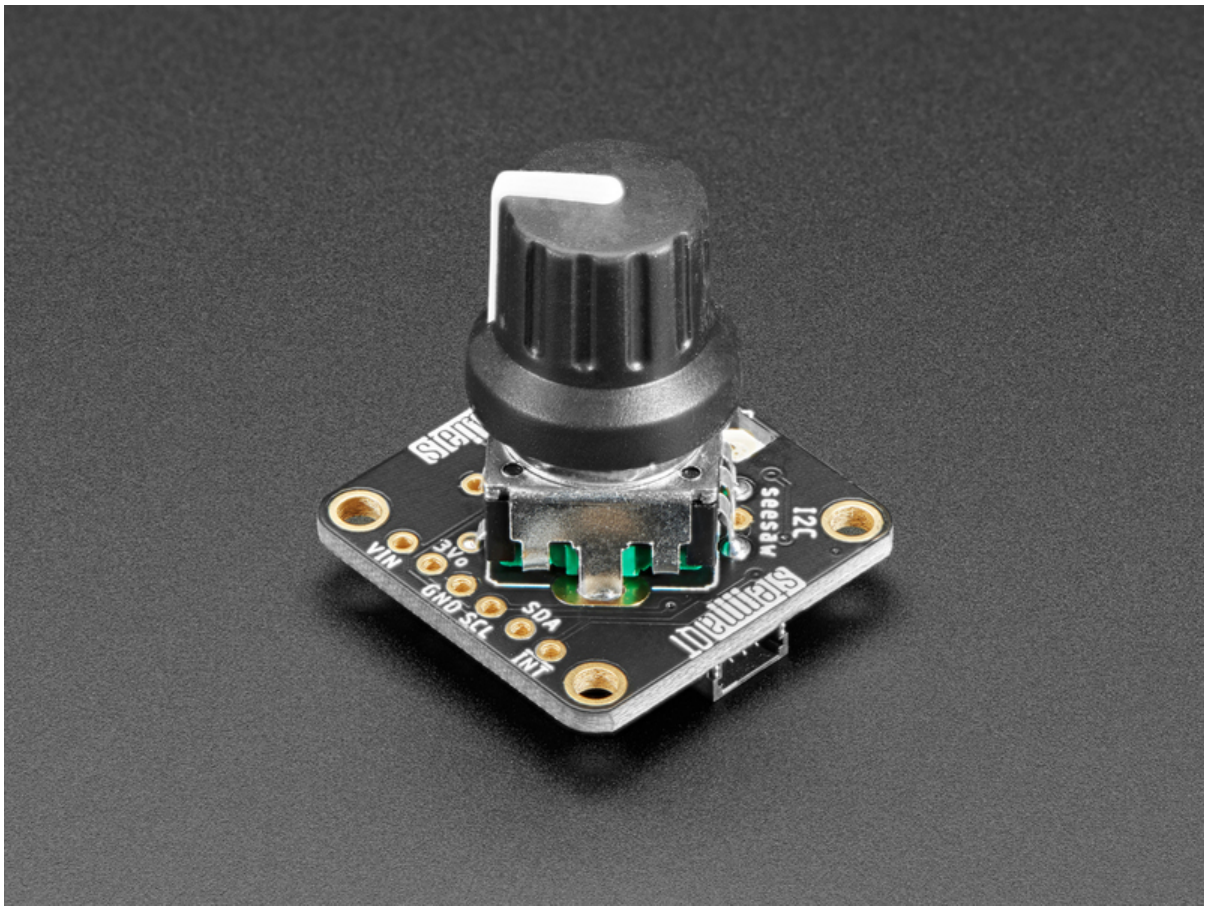
# Adafruit I2C QT Rotary Encoder

Created by Liz Clark

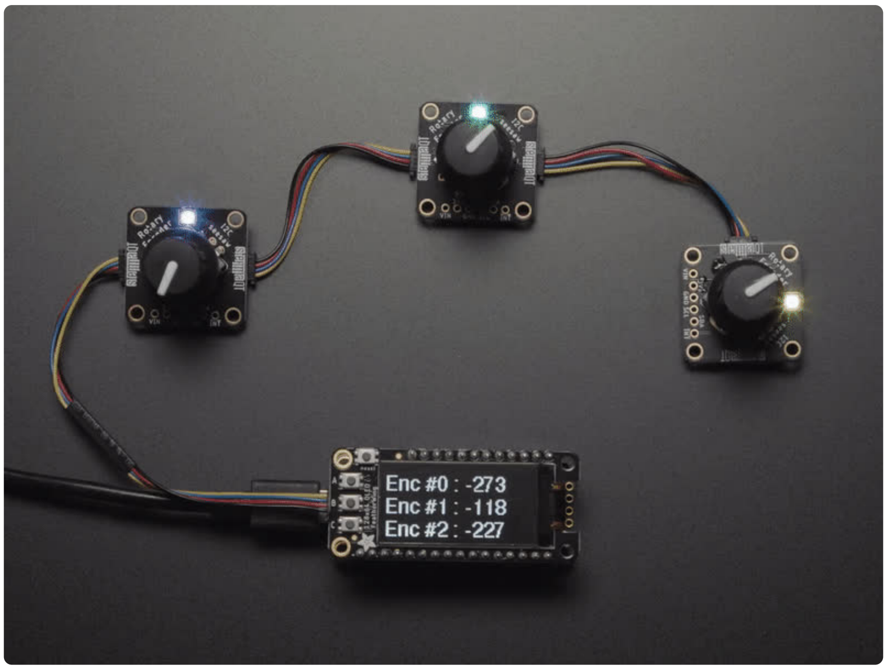


https://learn.adafruit.com/adafruit-i2c-qt-rotary-encoder

Last updated on 2024-06-03 03:24:00 PM EDT

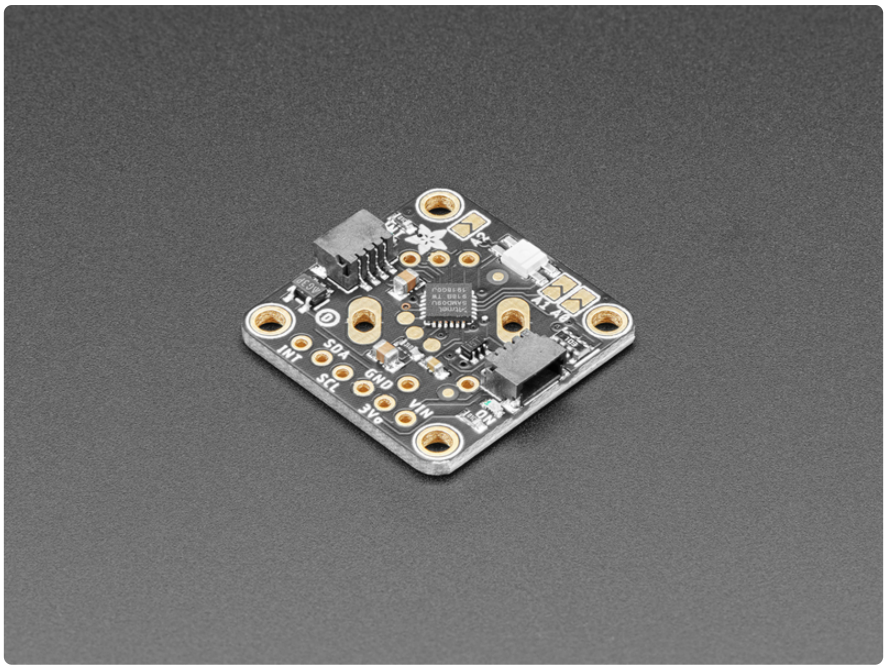
# Table of Contents

## Overview



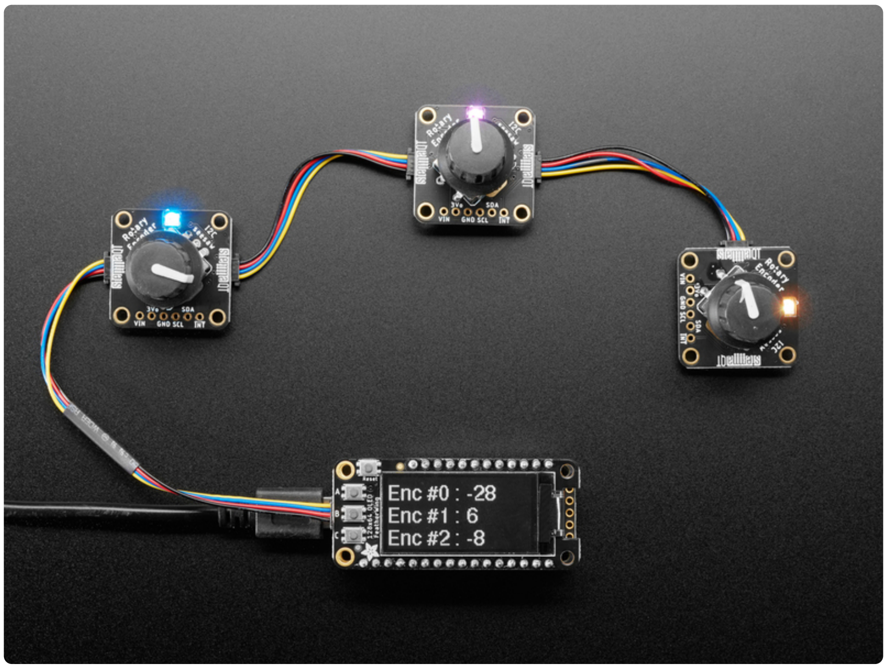
Rotary encoders are soooo much fun! Twist em this way, then twist them that way. Unlike potentiometers, they go all the way around, and often have little detents for tactile feedback. But, if you've ever tried to add encoders to your project you know that they're a real challenge to use: timers, interrupts, debouncing...

This Stemma QT breakout makes all that frustration go away - solder in any 'standard' PEC11-pinout rotary encoder with or without a push-switch. The onboard microcontroller is programmed with our seesaw firmware and will track all pulses and pins for you and then save the incremental value for querying at any time over I2C. Plug it in with a Stemma QT cable for instant rotary goodness, with any kind of microcontroller from an Arduino UNO up to a Raspberry Pi.



You can use our Arduino library to control and read data (https://adafru.it/BrV) with any compatible microcontroller. We also have CircuitPython/Python code (https://adafru.it/BrW) for use with computers or single-board Linux boards.

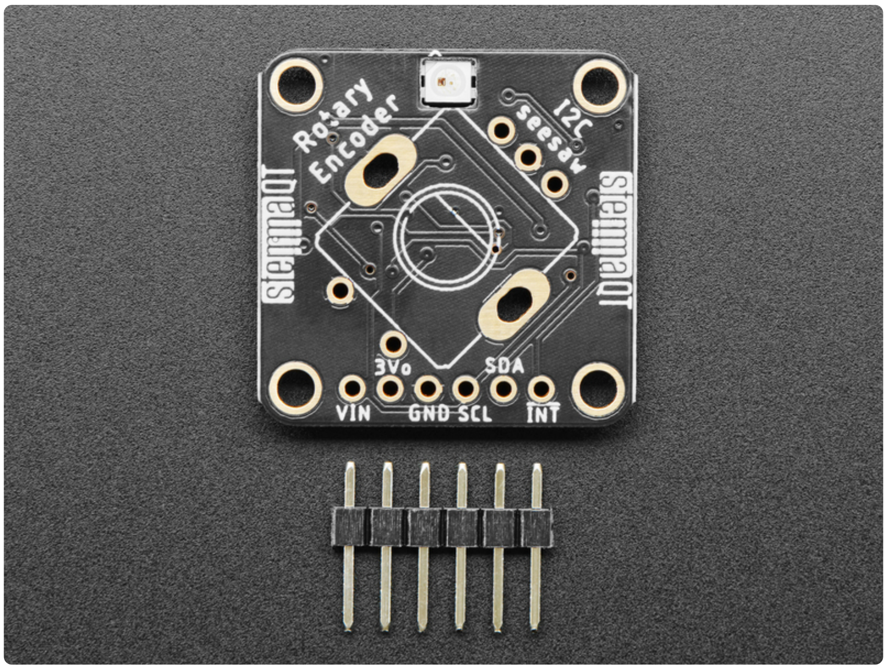
It's also easy to add this breakout to a breadboard - with six 0.1"-spaced breakout pads. Power with 3 to 5V DC and then use 3 or 5V logic I2C data. The INT pin can be configured to pulse low whenever rotation or push-buttoning is detected so you do not have to spam-read the I2C port to detect motion.



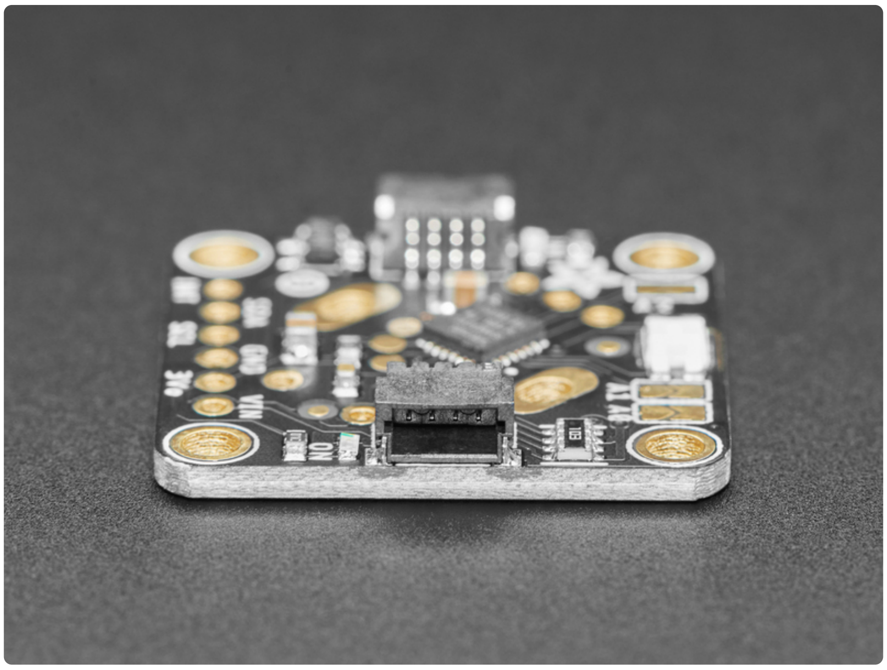
```
Enc #0 : -28
Enc #1 : 6
Enc #2 : -8
```

There's a NeoPixel onboard, that can display any color you like. It's also controlled over I2C for additional visual feedback or keep it off if you like. On the back there's a green power LED as well as a red INT LED that, if the interrupt is configured, will blink when the interrupt fires.

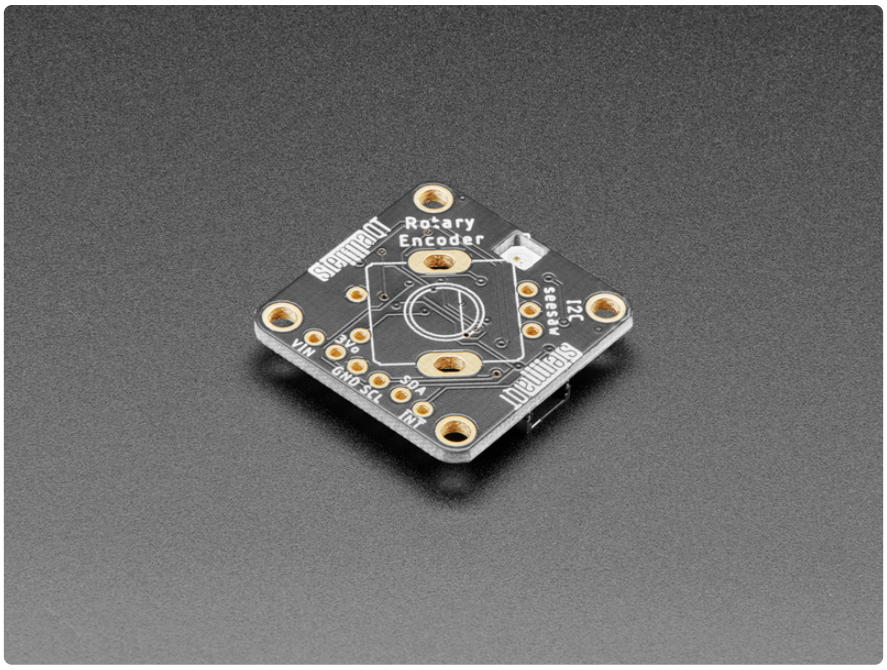
Using the three onboard address jumpers, you can connect up to 8 of these rotary encoders on a single I2C port. The first one will be at address 0x36, the last one at 0x3D when all three jumpers are soldered closed.



To keep the board nice and compact, only 1" x 1" we made the footprint for the rotary encoder at a 45-degree angle. Since it rotates around freely there's no need for it to be at a 90-degree angle to the PCB. Each order comes with one assembled and tested PCB breakout and a small piece of header.

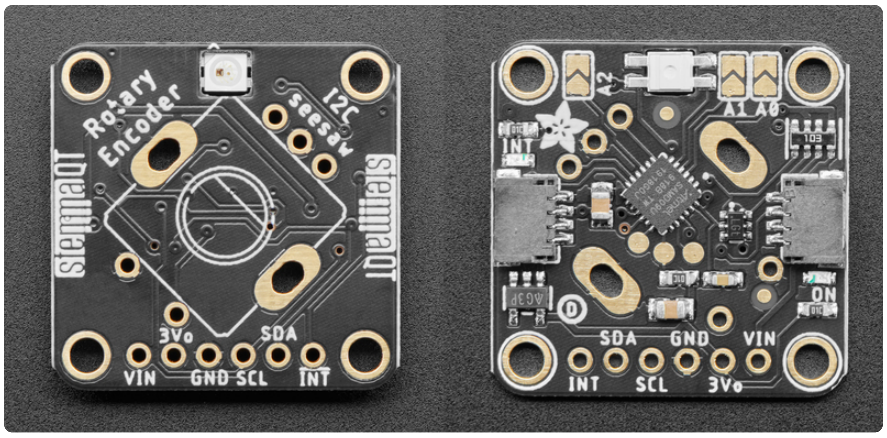


To get you going fast, we spun up a custom made PCB with the seesaw chip and all supporting circuitry, in the **STEMMA QT** form factor (https://adafru.it/LBQ), making them easy to interface with. The STEMMA QT connectors (https://adafru.it/JqB) on either side are compatible with the SparkFun Qwiic (https://adafru.it/Fpw) I2C connectors. This allows you to make solderless connections between your development board and the rotary encoder or to chain them with a wide range of other sensors and accessories using a **compatible cable** (https://adafru.it/JnB). QT Cable is not included, but we have a variety in the shop (https://adafru.it/17VE).



**This breakout does not come with an encoder soldered on**, so you can pick whatever encoder you like. We sell a common 24-detent-with-switch encoder here and it works wonderfully. (http://adafru.it/377) You can also use encoders without detents or with a different number of detents per rotation, of course! You'll need to solder the encoder and optional header onto the PCB to use with a solderless breadboard. but it's fairly easy and takes only a few minutes even for a beginner.

## Pinouts



The default I2C address is **0x36**.

Note that if you touch the back while using this board, you might add some conductivity to it, which can make it think you've pressed the button.

### Power Pins

- **VIN** - This is the **power pin**. Since the chip uses 3 VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller like Arduino, use 5V.
- **3Vo** - This is the **3.3V output** from the voltage regulator. You can grab up to 100mA from this if you like.

- **GND** - This is the **common ground** for power and logic.

If you notice any unusual behavior, especially with multiple encoders, consider hard-wiring a power supply to VIN and GND on the encoder.

## I2C Logic Pins

The default I2C address is 0x36.

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

## Other Pins

- **INT** - This is the **interrupt pin**. There is a **red INT LED** on the back of the board that, if this pin is configured, blinks when the interrupt fires.

## Address Jumpers

On the back of the board are **three address jumpers**, labeled **A0**, **A1**, and **A2**. These jumpers allow you to chain up to 8 of these boards on the same pair of I2C clock and data pins. To do so, you solder the jumpers "closed" by connecting the two pads.

If you happen to need more than 8, it's possible to set the I2C address with a special address-change command that is saved to the onboard non-volatile EEPROM memory.

The default I2C address is **0x36**. The other address options can be calculated by "adding" the **A0/A1/A2** to the base of **0x36**.

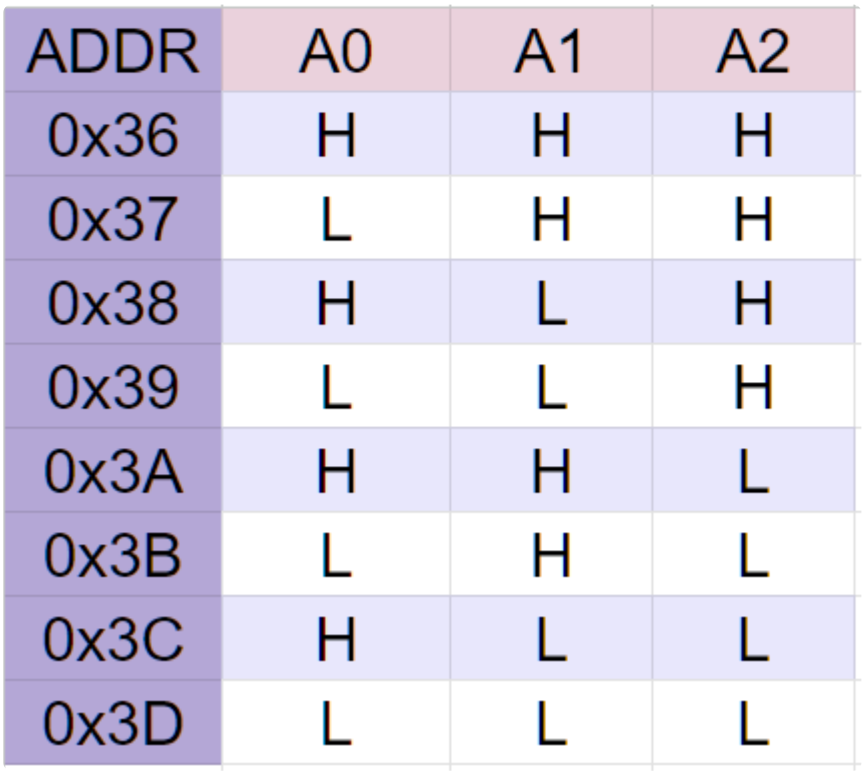
**A0** sets the lowest bit with a value of **1**, **A1** sets the next bit with a value of **2** and **A2** sets the next bit with a value of **4**. The final address is **0x36** + **A2** + **A1** + **A0** which would be **0x3D**.

If only **A0** is closed, the address is **0x36** + **1** = **0x37**

If only **A1** is closed, the address is **0x36** + **2** = **0x38**

If only **A2** is closed, the address is **0x36** + **4** = **0x3A**

The table below shows all possible addresses, and whether the pin(s) should be low (closed) or high (open).



| ADDR | A0 | A1 | A2 |
|---|---|---|---|
| 0x36 | H | H | H |
| 0x37 | L | H | H |
| 0x38 | H | L | H |
| 0x39 | L | L | H |
| 0x3A | H | H | L |
| 0x3B | L | H | L |
| 0x3C | H | L | L |
| 0x3D | L | L | L |

## seesaw Pins

- **Onboard NeoPixel LED** - This RGB LED is controlled over I2C using the seesaw library for additional visual feedback or keep it off if you like. It is connected to **seesaw pin 6**.
- **Rotary encoder** - The rotary encoder is mounted in the center of the board. You will have to solder it if you order PID4991 (http://adafru.it/4991). It comes pre-

soldered on PID5880 (http://adafru.it/5880). The push button on the rotary encoder is connected to **seesaw pin 24**.

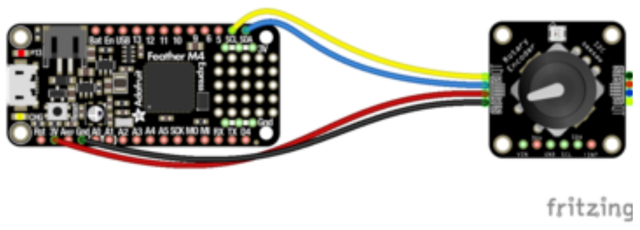
# Python & CircuitPython

It's easy to use the Adafruit I2C QT Rotary Encoder with CircuitPython using the Adafruit CircuitPython seesaw (https://adafru.it/BrW) library. This library allows you to write Python code that reads encoder position (relative to the starting position) and, if applicable to your rotary encoder, button presses.

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

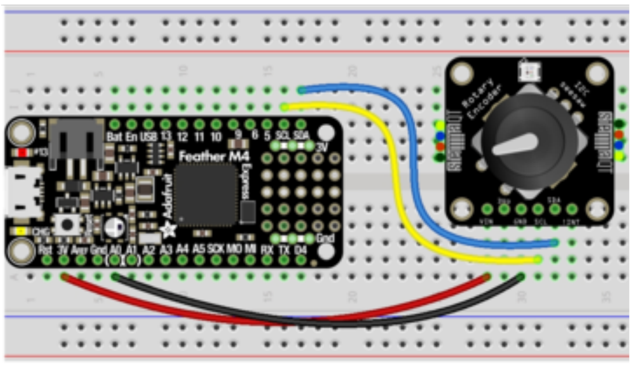
## CircuitPython Microcontroller Wiring

First wire up an I2C QT Rotary Encoder breakout to your board exactly as follows. The following is the breakout wired to a Feather using the STEMMA connector:



**Board 3V** to **breakout VIN (red wire)**
**Board GND** to **breakout GND (black wire)**
**Board SCL** to **breakout SCL (yellow wire)**
**Board SDA** to **breakout SDA (blue wire)**

The following is the breakout wired to a Feather using a solderless breadboard:

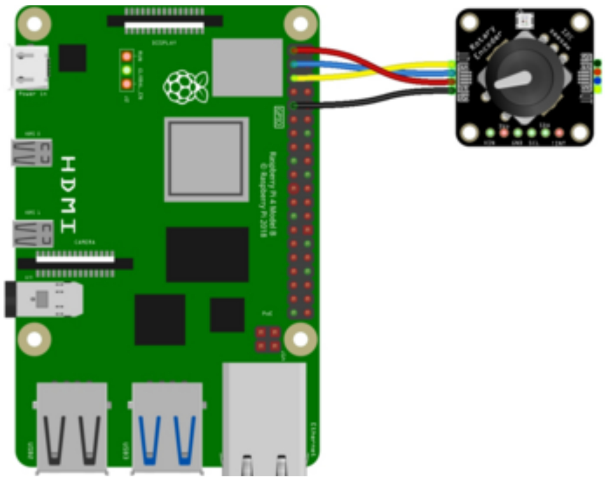


**Board 3V** to **breakout VIN (red wire)**
**Board GND** to **breakout GND (black wire)**
**Board SCL** to **breakout SCL (yellow wire)**
**Board SDA** to **breakout SDA (blue wire)**

## Python Computer Wiring

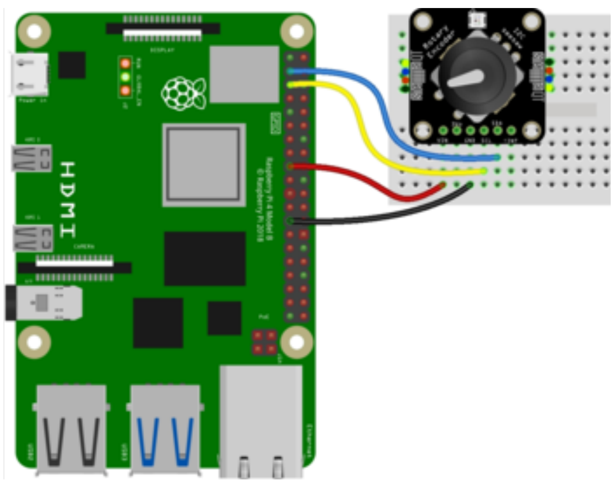
Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired with I2C using the STEMMA connector:



**Pi 3V** to **breakout VIN (red wire)**
**Pi GND** to **breakout GND (black wire)**
**Pi SCL** to **breakout SCL (yellow wire)**
**Pi SDA** to **breakout SDA (blue wire)**

Here's the Raspberry Pi wired with I2C using a solderless breadboard:



**Pi 3V** to **breakout VIN (red wire)**
**Pi GND** to **breakout GND (black wire)**
**Pi SCL** to **breakout SCL (yellow wire)**
**Pi SDA** to **breakout SDA (blue wire)**

## Python Installation of seesaw Library

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

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

- `pip3 install adafruit-circuitpython-seesaw`

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

## CircuitPython & Python Usage

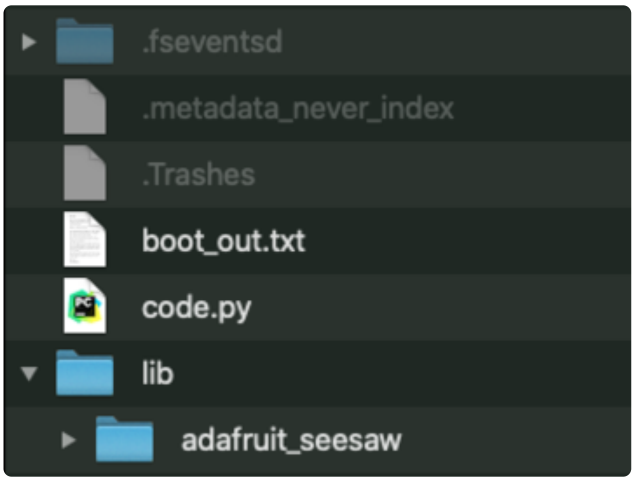
To demonstrate using this breakout with CircuitPython, you'll install the necessary library, update your code, and then connect to the serial console (https://adafru.it/Bec) to see the information printed out.

To use the I2C QT Rotary Encoder breakout with CircuitPython, you need to first install the seesaw library into the **lib** folder on your **CIRCUITPY** drive.

Then you need to update **code.py**.

Click the **Download Project Bundle** button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

```
# SPDX-FileCopyrightText: 2021 John Furcean
# SPDX-License-Identifier: MIT

"""I2C rotary encoder simple test example."""

import board
from adafruit_seesaw import seesaw, rotaryio, digitalio

# For use with the STEMMA connector on QT Py RP2040
# import busio
# i2c = busio.I2C(board.SCL1, board.SDA1)
# seesaw = seesaw.Seesaw(i2c, 0x36)

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
seesaw = seesaw.Seesaw(i2c, addr=0x36)

seesaw_product = (seesaw.get_version() >> 16) & 0xFFFF
print("Found product {}".format(seesaw_product))
if seesaw_product != 4991:
    print("Wrong firmware loaded?  Expected 4991")

# Configure seesaw pin used to read knob button presses
# The internal pull up is enabled to prevent floating input
seesaw.pin_mode(24, seesaw.INPUT_PULLUP)
button = digitalio.DigitalIO(seesaw, 24)

button_held = False

encoder = rotaryio.IncrementalEncoder(seesaw)
last_position = None

while True:
    # negate the position to make clockwise rotation positive
    position = -encoder.position

    if position != last_position:
        last_position = position
        print("Position: {}".format(position))

    if not button.value and not button_held:
        button_held = True
        print("Button pressed")

    if button.value and button_held:
        button_held = False
        print("Button released")
```



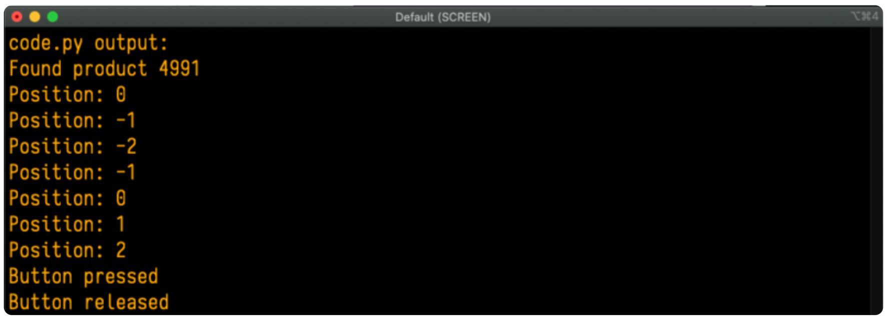
Your **CIRCUITPY** drive should resemble the image.

You should have in / of the **CIRCUITPY** drive:

**code.py**
And in the **lib** folder on your **CIRCUITPY** drive:

**adafruit_seesaw/**

Now, connect to the serial console (https://adafru.it/Bec). Try rotating the rotary encoder to see the position change, and, if applicable, press the button to see the button status printed out!



```
code.py output:
Found product 4991
Position: 0
Position: -1
Position: -2
Position: -1
Position: 0
Position: 1
Position: 2
Button pressed
Button released
```

That's all there is to using the I2C QT Rotary Encoder with CircuitPython!

## NeoPixel Color Picker Example

Update your **code.py** file to the following.

Click the **Download Project Bundle** button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **seesaw_rotary_neopixel.py** file to your **CIRCUITPY** drive.

Rename **seesaw_rotary_neopixel.py** to **code.py**.

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

"""I2C rotary encoder NeoPixel color picker and brightness setting example."""
import board
from rainbowio import colorwheel
from adafruit_seesaw import seesaw, neopixel, rotaryio, digitalio


# For use with the STEMMA connector on QT Py RP2040
# import busio
# i2c = busio.I2C(board.SCL1, board.SDA1)
# seesaw = seesaw.Seesaw(i2c, 0x36)

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
seesaw = seesaw.Seesaw(i2c, 0x36)

encoder = rotaryio.IncrementalEncoder(seesaw)
seesaw.pin_mode(24, seesaw.INPUT_PULLUP)
switch = digitalio.DigitalIO(seesaw, 24)

pixel = neopixel.NeoPixel(seesaw, 6, 1)
pixel.brightness = 0.5

last_position = -1
color = 0  # start at red

while True:
    # negate the position to make clockwise rotation positive
    position = -encoder.position

    if position != last_position:
        print(position)

        if switch.value:
            # Change the LED color.
            if position > last_position:  # Advance forward through the colorwheel.
                color += 1
            else:
                color -= 1  # Advance backward through the colorwheel.
            color = (color + 256) % 256  # wrap around to 0-256
            pixel.fill(colorwheel(color))

        else:  # If the button is pressed...
            # ...change the brightness.
            if position > last_position:  # Increase the brightness.
                pixel.brightness = min(1.0, pixel.brightness + 0.1)
            else:  # Decrease the brightness.
                pixel.brightness = max(0, pixel.brightness - 0.1)

    last_position = position
```

Rotate the rotary encoder to cycle through the NeoPixel rainbow. Press the rotary encoder switch and rotate while holding the switch to change the brightness.

That's all there is to using the NeoPixel with the rotary encoder using CircuitPython!

## Multiple QT Rotary Encoder Example

Here's an example that uses multiple boards. The first board is in the default state, while the second board has it's A0 jumper bridged to set the I2C address to `0x37`.

```
# SPDX-FileCopyrightText: 2021 John Park
# SPDX-License-Identifier: MIT

# I2C rotary encoder multiple test example.
# solder the A0 jumper on the second QT Rotary Encoder board

import board
from adafruit_seesaw import seesaw, rotaryio, digitalio, neopixel

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller

qt_enc1 = seesaw.Seesaw(i2c, addr=0x36)
qt_enc2 = seesaw.Seesaw(i2c, addr=0x37)

qt_enc1.pin_mode(24, qt_enc1.INPUT_PULLUP)
button1 = digitalio.DigitalIO(qt_enc1, 24)
button_held1 = False

qt_enc2.pin_mode(24, qt_enc2.INPUT_PULLUP)
button2 = digitalio.DigitalIO(qt_enc2, 24)
button_held2 = False

encoder1 = rotaryio.IncrementalEncoder(qt_enc1)
last_position1 = None
```

```
encoder2 = rotaryio.IncrementalEncoder(qt_enc2)
last_position2 = None

pixel1 = neopixel.NeoPixel(qt_enc1, 6, 1)
pixel1.brightness = 0.2
pixel1.fill(0xff0000)

pixel2 = neopixel.NeoPixel(qt_enc2, 6, 1)
pixel2.brightness = 0.2
pixel2.fill(0x0000ff)


while True:

    # negate the position to make clockwise rotation positive
    position1 = -encoder1.position
    position2 = -encoder2.position

    if position1 != last_position1:
        last_position1 = position1
        print("Position 1: {}".format(position1))

    if not button1.value and not button_held1:
        button_held1 = True
        pixel1.brightness = 0.5
        print("Button 1 pressed")

    if button1.value and button_held1:
        button_held1 = False
        pixel1.brightness = 0.2
        print("Button 1 released")


    if position2 != last_position2:
        last_position2 = position2
        print("Position 2: {}".format(position2))

    if not button2.value and not button_held2:
        button_held2 = True
        pixel2.brightness = 0.5
        print("Button 2 pressed")

    if button2.value and button_held2:
        button_held2 = False
        pixel2.brightness = 0.2
        print("Button 2 released")
```

## Python Docs

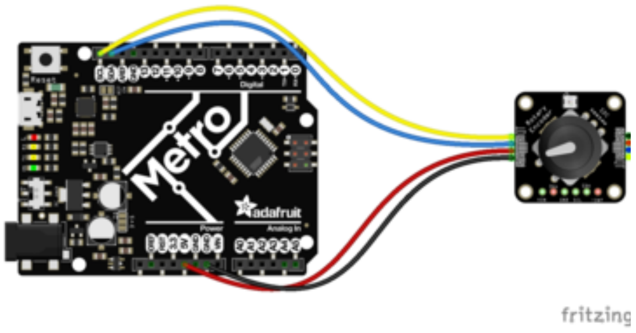
Python Docs (https://adafru.it/C5y)

## Arduino

The Adafruit I2C QT Rotary Encoder uses the seesaw chip. To use the QT Rotary Encoder with Arduino, you'll use the Adafruit seesaw library. With the STEMMA QT connectors, you can easily get started with minimal soldering required for the rotary encoder. Solder a rotary encoder to your breakout before starting.

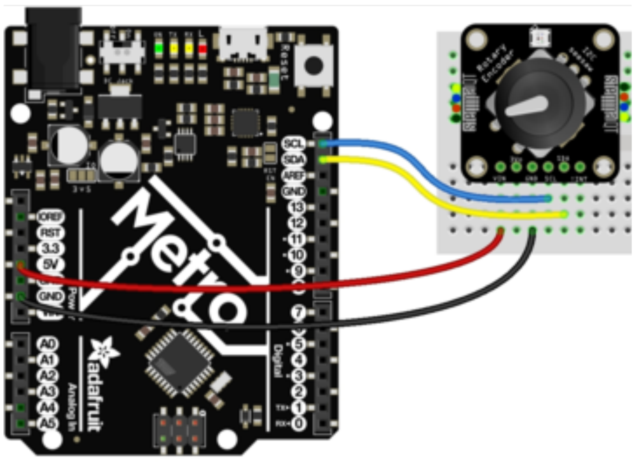
## I2C Wiring

Here is how to wire up the breakout using one of the **STEMMA QT** (https://adafru.it/Ft4) connectors. The examples show a Metro but wiring will work the same for an Arduino or other compatible board.



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

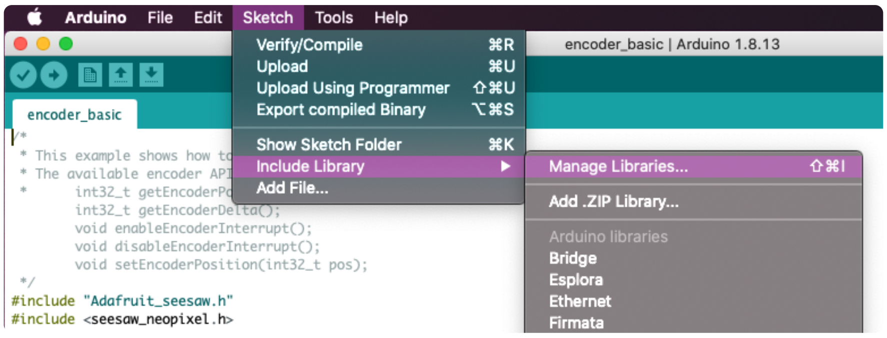
Here is how to wire the breakout to a board using a solderless breadboard. To do this, you must solder the header pins provided to the breakout.



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

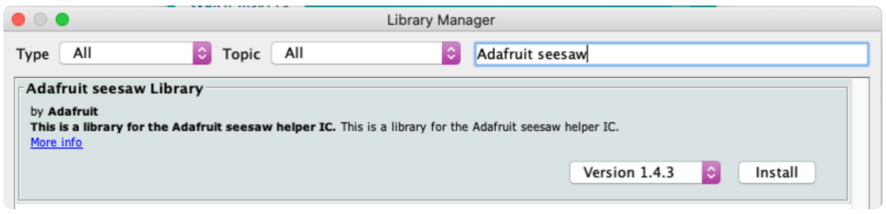
## Library Installation

You can install the Adafruit seesaw library for Arduino using the Library Manager in the Arduino IDE.



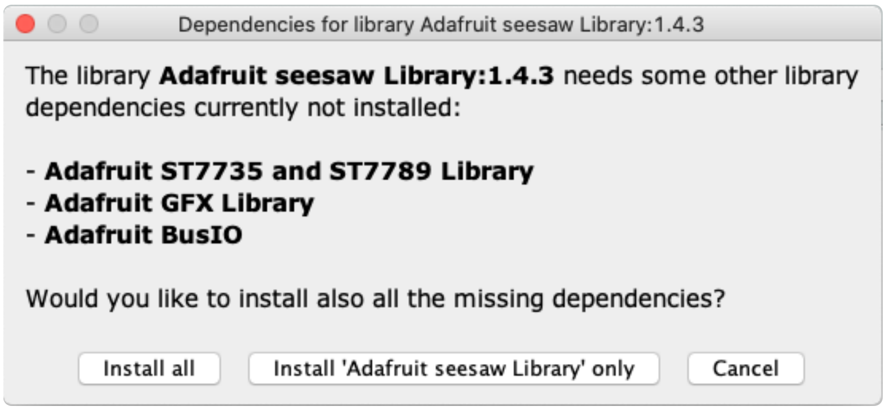
```
/*
 * This example shows how t
 * The available encoder API
 *      int32_t getEncoderPo
        int32_t getEncoderDelta();
        void enableEncoderInterrupt();
        void disableEncoderInterrupt();
        void setEncoderPosition(int32_t pos);
 */
#include "Adafruit_seesaw.h"
#include <seesaw_neopixel.h>
```

Click the **Manage Libraries ...** menu item, search for **Adafruit seesaw** , and select the **Adafruit seesaw** library:



**Adafruit seesaw Library**
by **Adafruit**
**This is a library for the Adafruit seesaw helper IC.** This is a library for the Adafruit seesaw helper IC.
More info

When asked to install the Adafruit seesaw library dependencies, click **Install all**.



Dependencies for library Adafruit seesaw Library:1.4.3

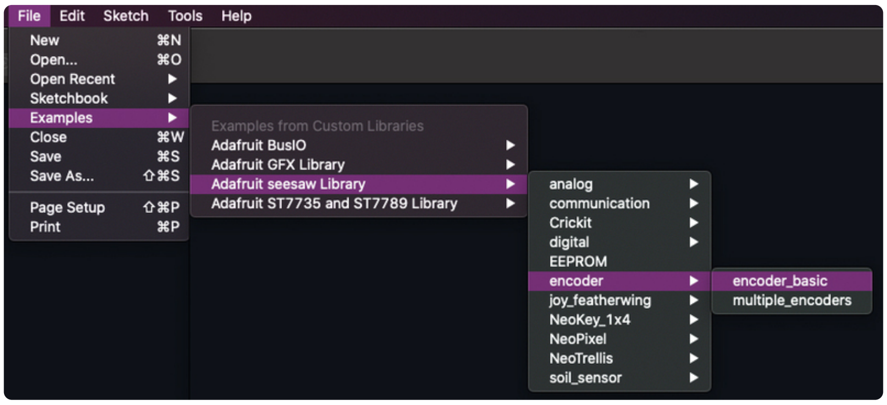
The library **Adafruit seesaw Library:1.4.3** needs some other library dependencies currently not installed:

- **Adafruit ST7735 and ST7789 Library**
- **Adafruit GFX Library**
- **Adafruit BusIO**

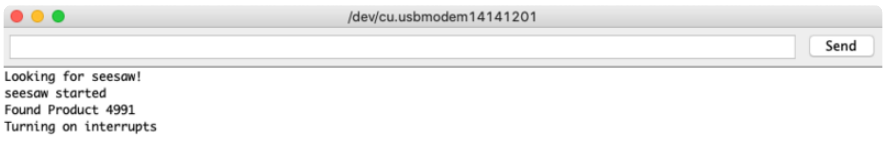
Would you like to install also all the missing dependencies?

## Load Example

Open up **File -> Examples -> Adafruit seesaw Library -> encoder -> encoder_basic**

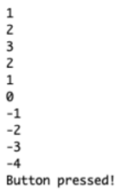


After opening the **encoder_basic** file, upload it to the Arduino wired to your rotary encoder. Open the **Serial Monitor** at **115200 baud**. You should see the following as the sketch starts up.



```
Looking for seesaw!
seesaw started
Found Product 4991
Turning on interrupts
```

Now, rotate the rotary encoder back and forth. And try pressing the button. You should see something similar to the following in the Serial Monitor.



```
1
2
3
2
1
0
-1
-2
-3
-4
Button pressed!
```

That's all there is to using the I2C QT Rotary Encoder with Arduino!

## Example Code

```
/*
 * This example shows how to read from a seesaw encoder module.
 * The available encoder API is:
 *      int32_t getEncoderPosition();
        int32_t getEncoderDelta();
        void enableEncoderInterrupt();
        void disableEncoderInterrupt();
        void setEncoderPosition(int32_t pos);
 */
#include "Adafruit_seesaw.h"
#include <seesaw_neopixel.h>

#define SS_SWITCH        24
#define SS_NEOPIX        6

#define SEESAW_ADDR          0x36

Adafruit_seesaw ss;
seesaw_NeoPixel sspixel = seesaw_NeoPixel(1, SS_NEOPIX, NEO_GRB + NEO_KHZ800);

int32_t encoder_position;

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);

  Serial.println("Looking for seesaw!");

  if (! ss.begin(SEESAW_ADDR) || ! sspixel.begin(SEESAW_ADDR)) {
    Serial.println("Couldn't find seesaw on default address");
    while(1) delay(10);
  }
  Serial.println("seesaw started");

  uint32_t version = ((ss.getVersion() >> 16) & 0xFFFF);
  if (version  != 4991){
    Serial.print("Wrong firmware loaded? ");
    Serial.println(version);
    while(1) delay(10);
  }
  Serial.println("Found Product 4991");

  // set not so bright!
  sspixel.setBrightness(20);
  sspixel.show();

  // use a pin for the built in encoder switch
  ss.pinMode(SS_SWITCH, INPUT_PULLUP);

  // get starting position
  encoder_position = ss.getEncoderPosition();

  Serial.println("Turning on interrupts");
  delay(10);
  ss.setGPIOInterrupts((uint32_t)1 << SS_SWITCH, 1);
  ss.enableEncoderInterrupt();
}

void loop() {
  if (! ss.digitalRead(SS_SWITCH)) {
    Serial.println("Button pressed!");
  }

  int32_t new_position = ss.getEncoderPosition();
  // did we move arounde?
  if (encoder_position != new_position) {
    Serial.println(new_position);         // display new position

    // change the neopixel color
    sspixel.setPixelColor(0, Wheel(new_position & 0xFF));
    sspixel.show();
    encoder_position = new_position;      // and save for next round
  }

  // don't overwhelm serial port
  delay(10);
}


uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if (WheelPos < 85) {
    return sspixel.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  }
  if (WheelPos < 170) {
    WheelPos -= 85;
    return sspixel.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
  WheelPos -= 170;
  return sspixel.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
}
```

## Arduino Docs

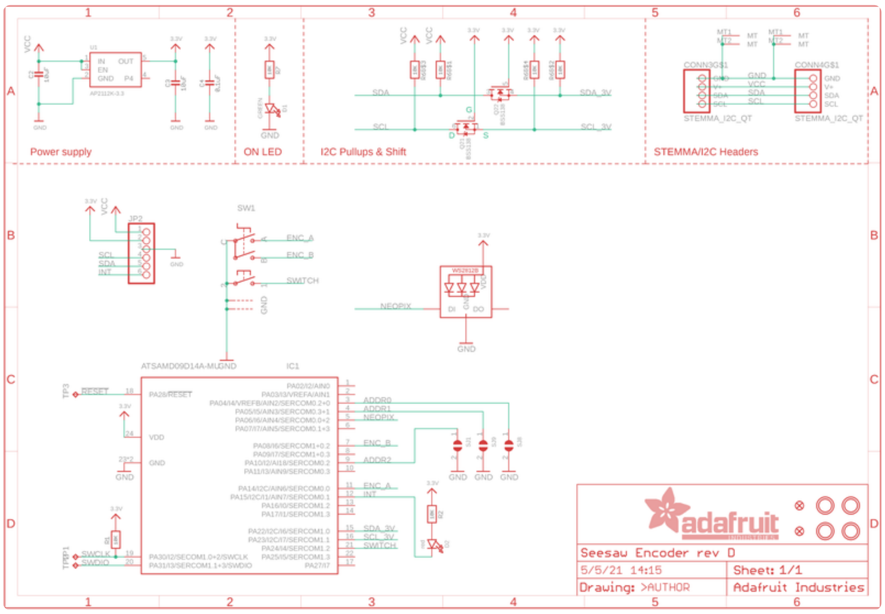
Arduino Docs (https://adafru.it/SdQ)

## Downloads

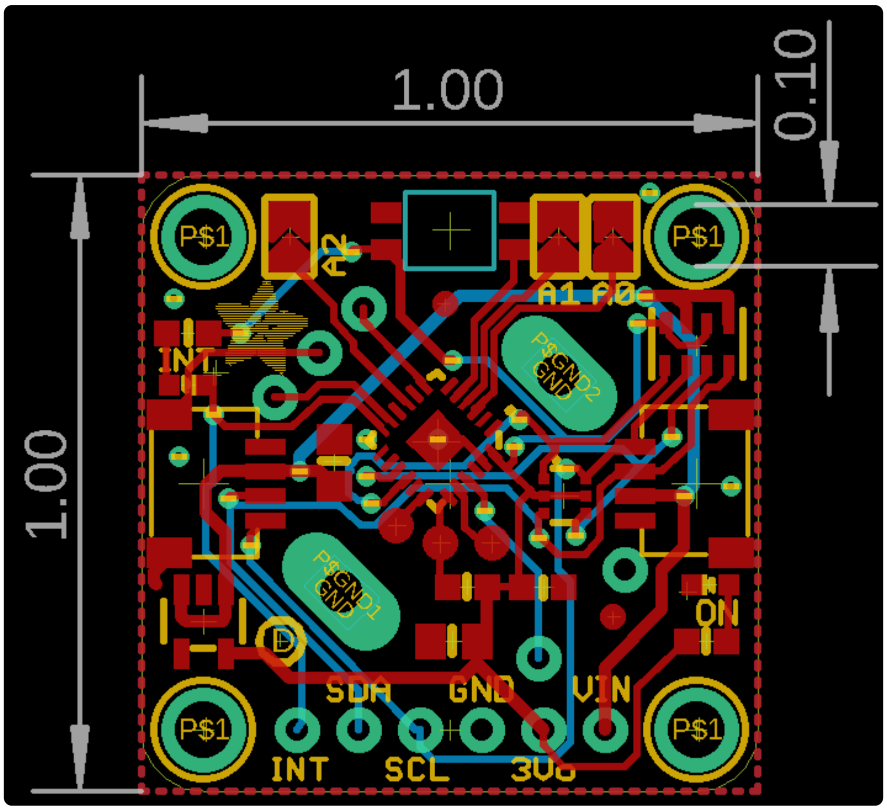
### Files:

- SAMD09 datasheet (https://adafru.it/BrY)
- EagleCAD PCB files on GitHub (https://adafru.it/Sdd)
- 3D Models on GitHub (https://adafru.it/SgF)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/Sde)

## Schematic



## Fab Print



## 3D Model