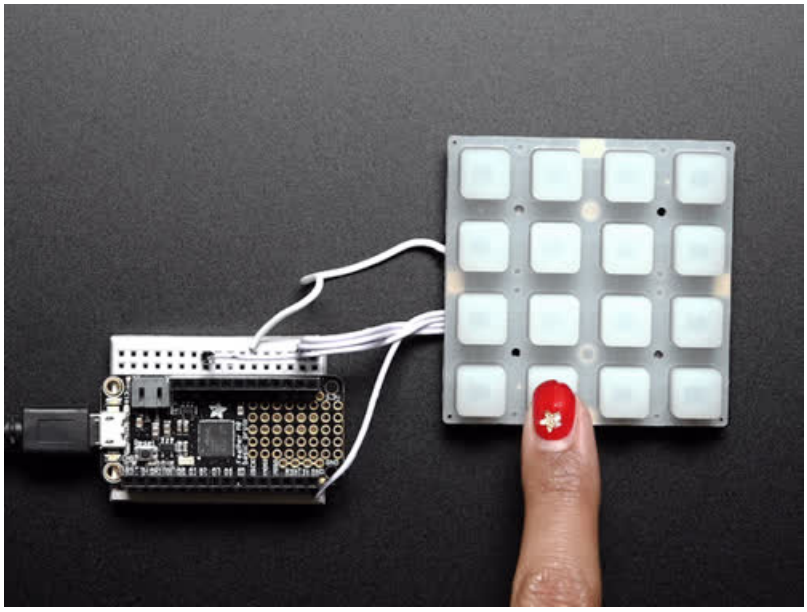
adafruit learning system

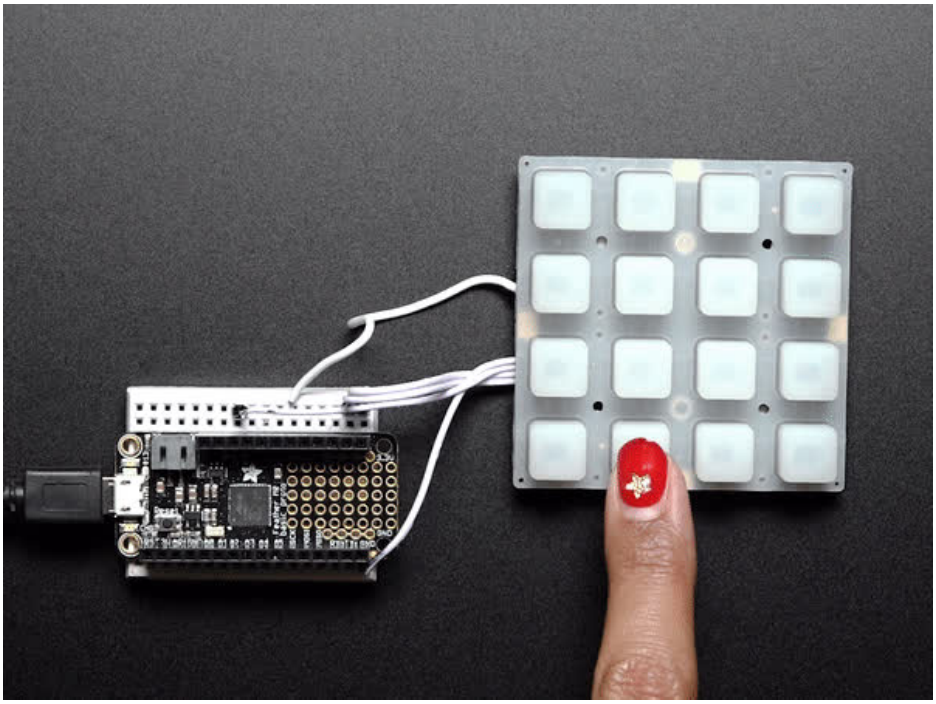
Adafruit NeoTrellis

Created by lady ada

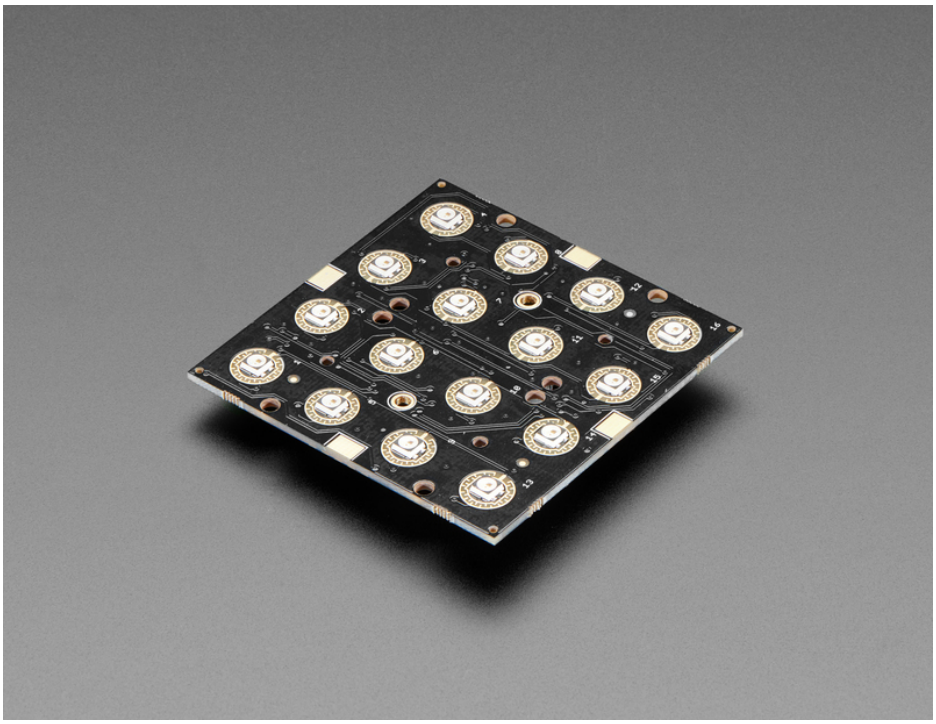


Last updated on 2019-05-13 03:17:58 PM UTC

Overview

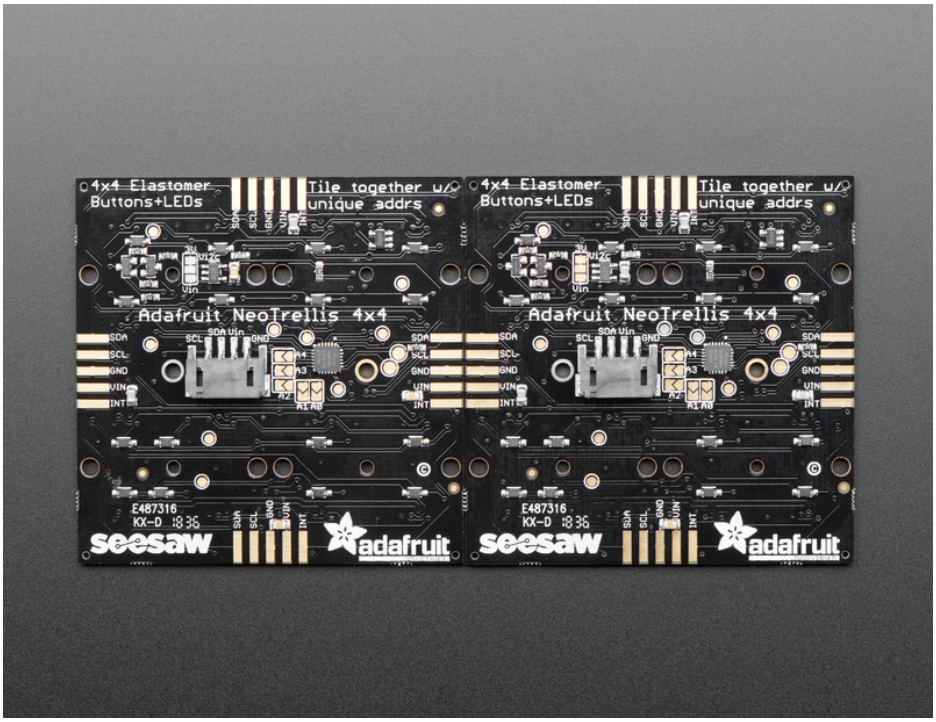


By popular request, we've upgraded our popular Trellis elastomer button kits to now have a PCB with *full color NeoPixel* support! You heard that right, no more single-color LEDs, you can now have any color you like under the fantastic rubbery button pads we sell.

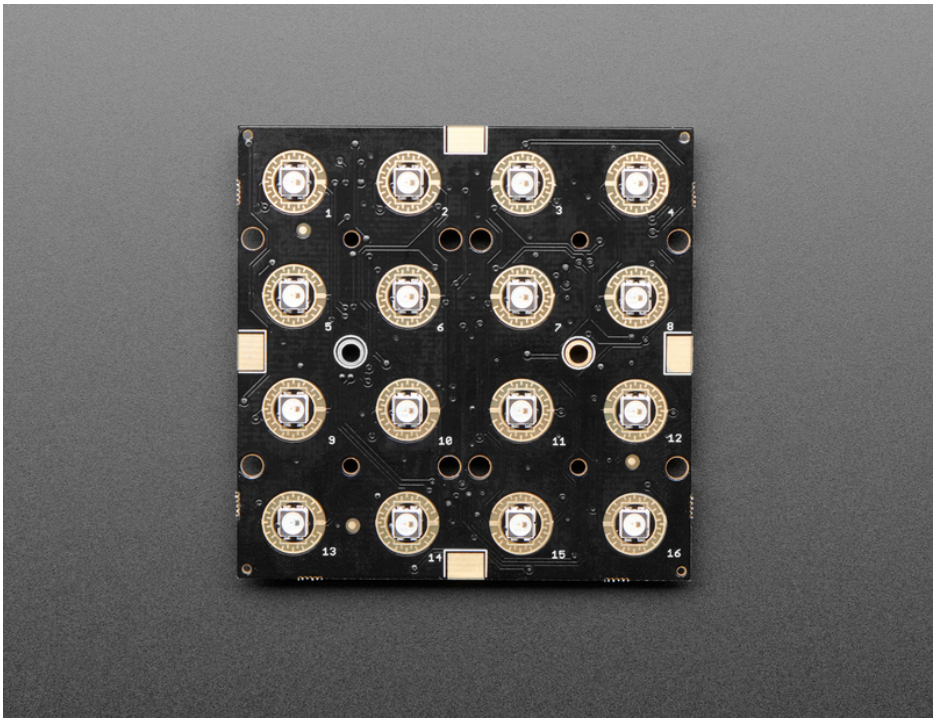


These 4x4 button pad boards are fully tile-able and communicate over I2C. With 5 address pins, you've got the ability to connect up to 32 together in any arrangement you like. With our trusty seesaw I2C-to-anything chip (https://adafru.it/Cmk), you don't even need to manage the NeoPixel driving. That's right! Both the button

management and LED driving is completely handled for you all over plain I2C. With both Arduino/C++ and CircuitPython/Python library support, you can use these pads with any and all microcontroller or computer boards.

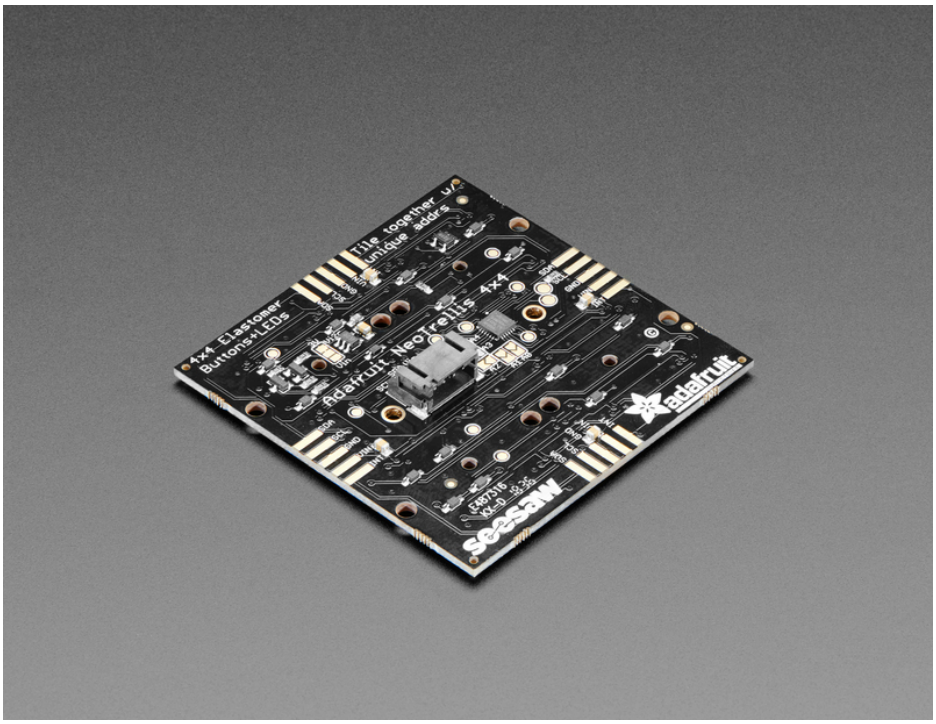


Perfect for your next cool interface, MIDI instrument, control panel...whatever could benefit from beautiful diffused colorful buttons. For fast connection to a single board, we include a JST-PH 4-pin connector that will give you power+data access. It's Grove compatible (the cable fits, even though its not the exact same connector), or you can use a JST-PH 4 pin cable (https://adafru.it/CzH).



Each order comes with one NeoTrellis board with seesaw chip and 16 NeoPixels already soldered in place. **Elastomer**

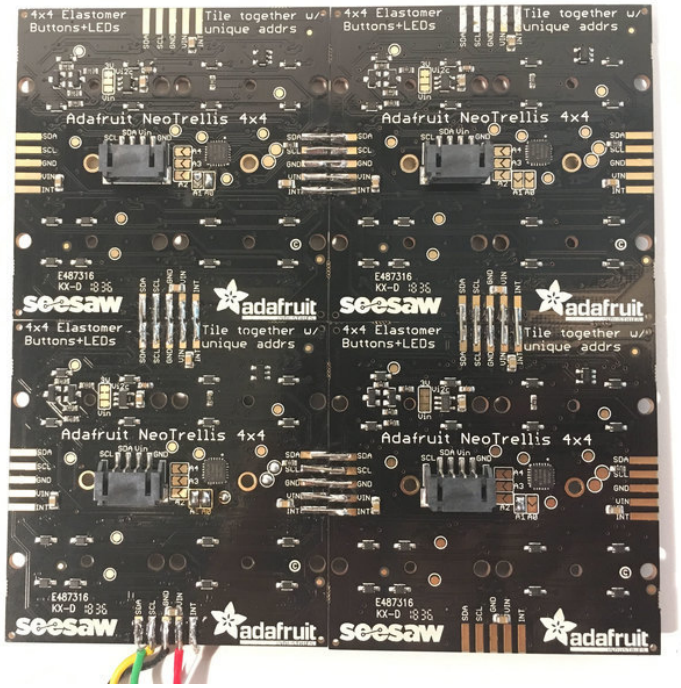
button pad not included, so be sure to pick one up (https://adafru.it/dLy)!



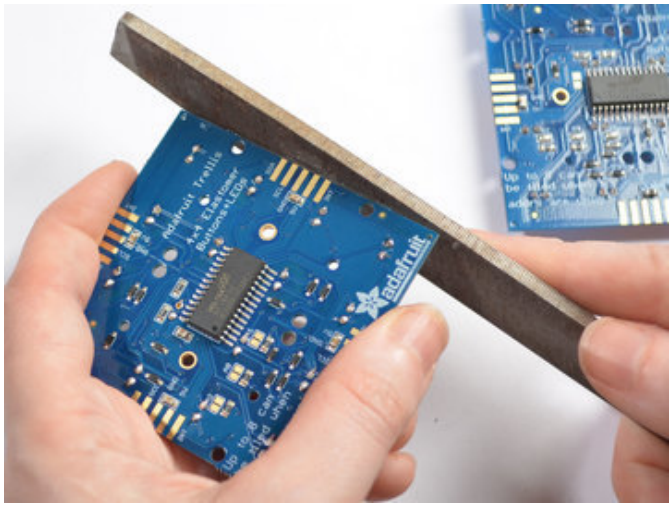
Tiling

You can tile up to 32 NeoTrellis PCBs on a single 2-wire I2C bus. This allows you to easily build up to 16x16 larger panels which can be lots of fun!

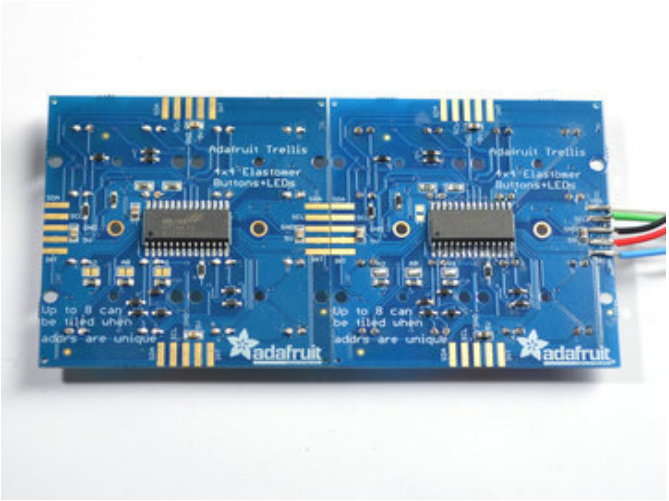
To start with, its a good idea to assemble and test each individually so you know each NeoTrellis works individually.



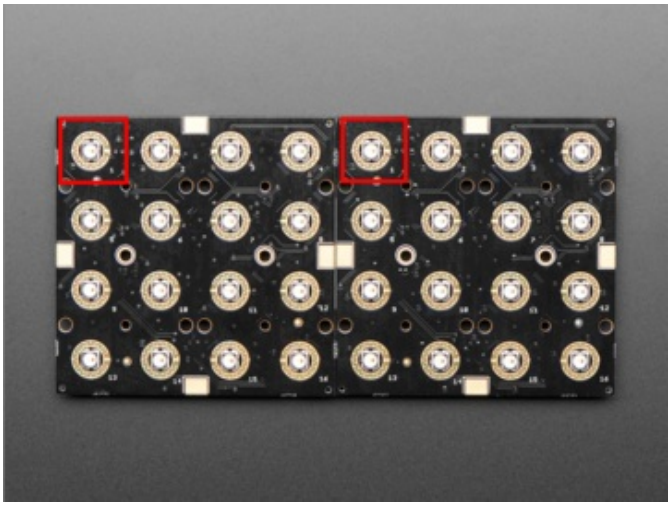
The photos below show the earlier non-Neo Trellis but the process is identical for NeoTrellis!



There's little nubs on the sides of some PCBs that keep them on the assembly panel, you can file them off with any file or sandpaper.

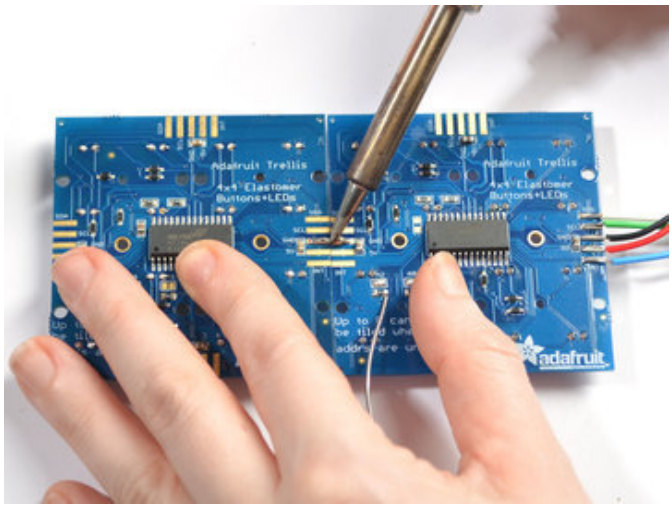


Arrange the tiles up the way you want, we'll start with two. Make sure the Adafruit logo is lined up the same.

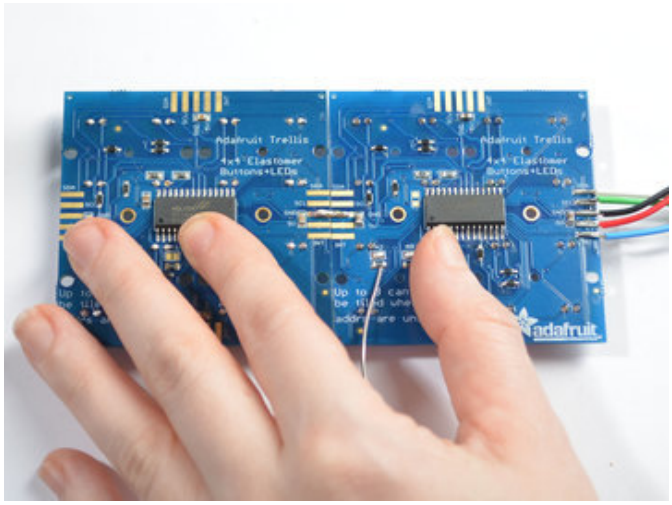


On the other side make sure LEDs #1 are in the same corner

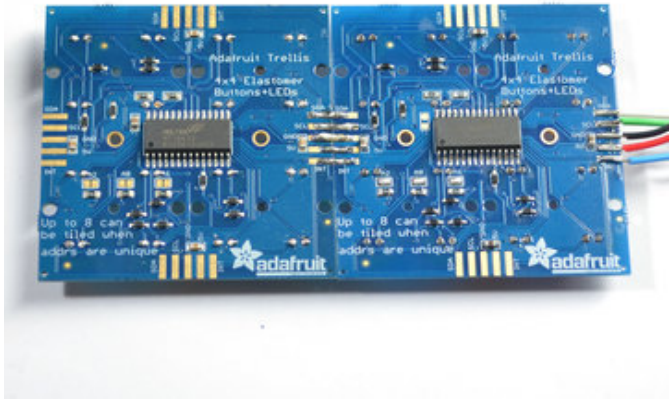
Solder two blobs of solder on two adjacent finger pads.

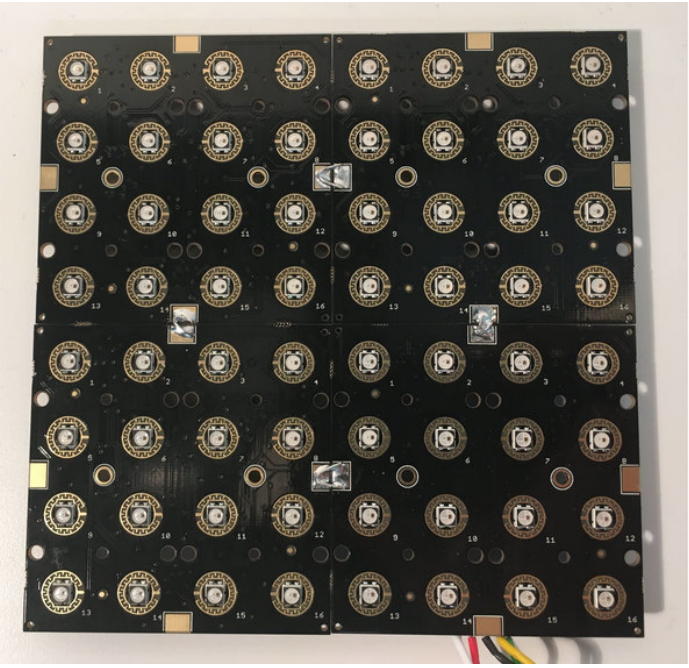


Use your soldering iron to drag solder from one pad to another, with a little effort they'll stick together and make a connection. You can add more solder to make the connection stronger. Its still not mechanically strong - so be careful not to bend or shake the arrangement



Repeat for the other 4 fingers





Addressing

Each NeoTrellis tile must have a unique address. You can set the addresses on the back of each panel using a little solder over the address jumpers.

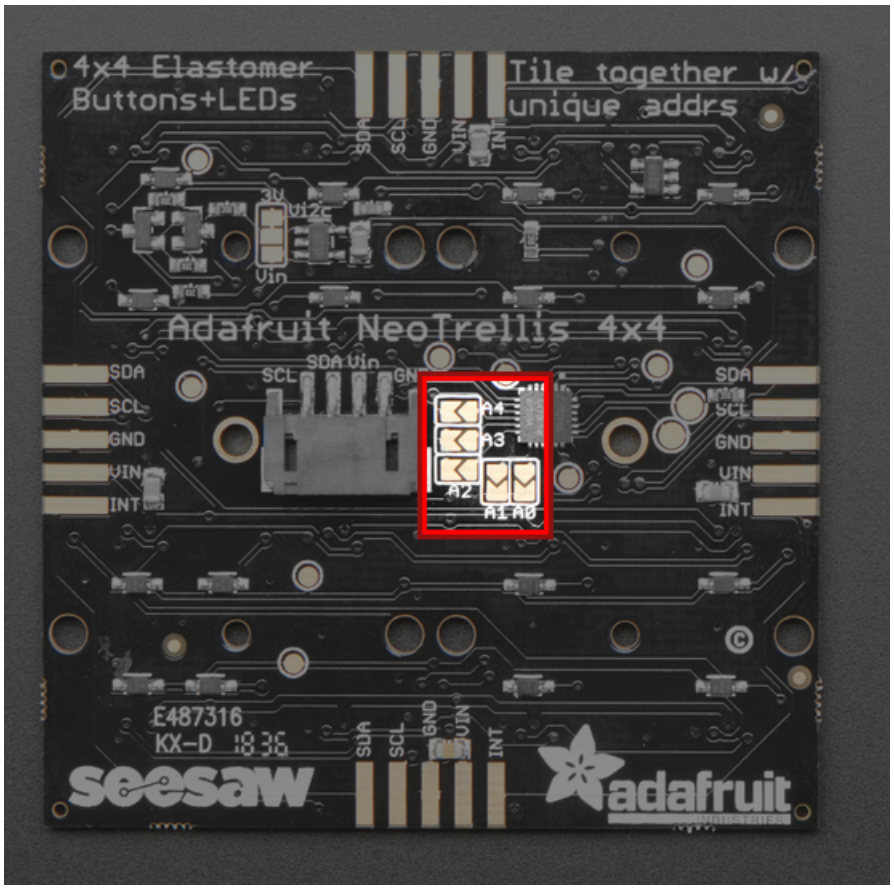
The seesaw driver chip on the Trellis has a default I2C address of **0x2F**. Since each device on an I2C bus must have a unique address, its important to avoid collisions or you'll get a lot of strange responses from your electronic devices!

Luckily, the seesaw has 5 address adjust pins, so that the address can be changed. Each pin changes one binary bit of the address, so you can set the address to any hex number between **0x2F** and **0x4E** inclusive

The panels don't have to have consecutive address #'s, they just have to be unique.

Changing Addresses

You can change the address of very easily. Look on the back to find the three **A0, A1, A2, A3** and **A4** solder jumpers.

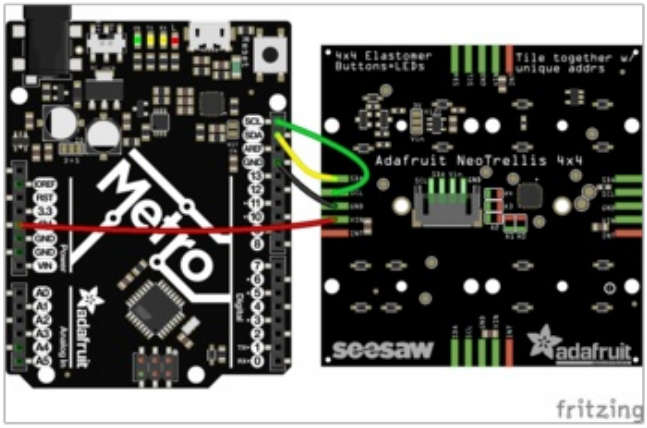


Each one of these is used to hard-code in the address. If a jumper is shorted with solder, that sets the address.

- **A0** sets the lowest bit with a value of **1**
- **A1** sets the bit with a value of **2**
- **A2** sets the bit with a value of **4**
- **A3** sets the bit with a value of **8**
- **A4** sets the bit with a value of **16**

The final address is **0x2F** + **A4** + **A3** + **A2** + **A1** + **A0.** So for example if **A2** is shorted and **A0** is shorted, the address is **0x2F** + **4** + **1** = **0x34**

Arduino Code

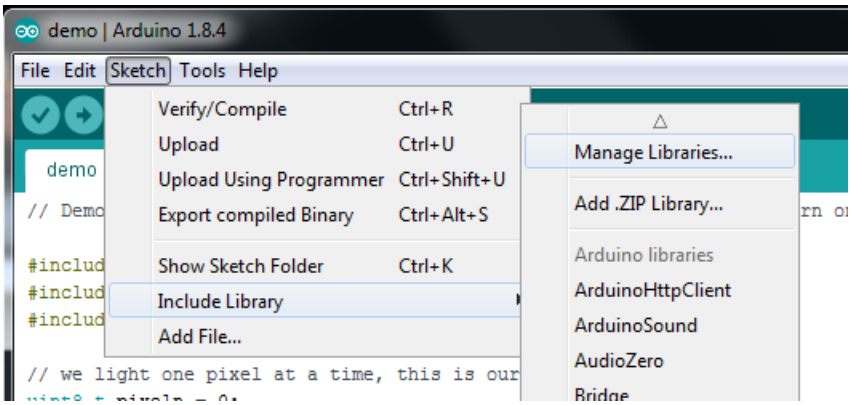


Wiring is easy, solder the four **VIN GND SDA SCL** pads to wires and then connect to your Arduino compatible:

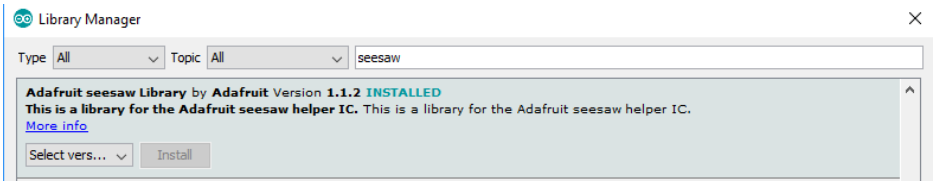
- **VIN** to **5V** (or 3.3V if 5V is not available)
- **GND** to **GND**
- **SDA** to I2C data **SDA**
- **SCL** to I2C clock **SCL**

Install Arduino Libraries

Lets begin by installing all the libraries we need. Open up the library manager in Arduino IDE

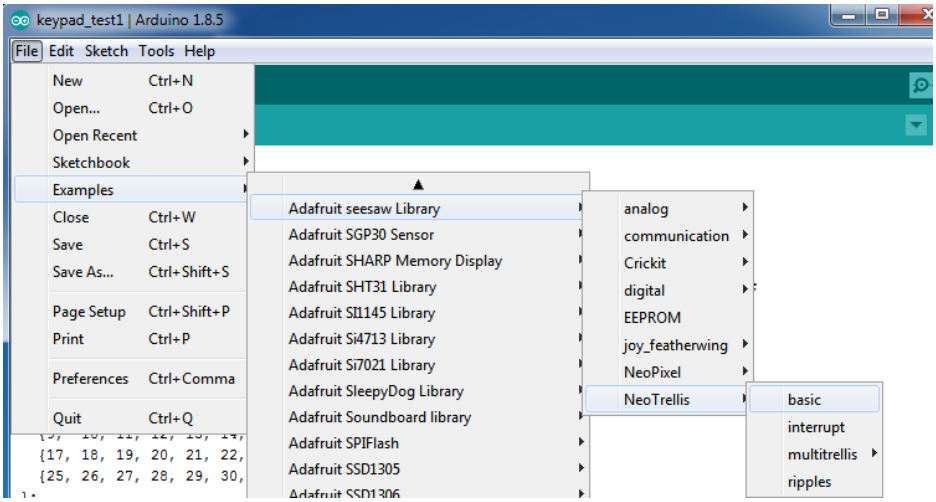


Search for and install the latest version of the Adafruit seesaw library



Basic Example

Start by opening up the **NeoTrellis -> basic** under the Adafruit Seesaw library:

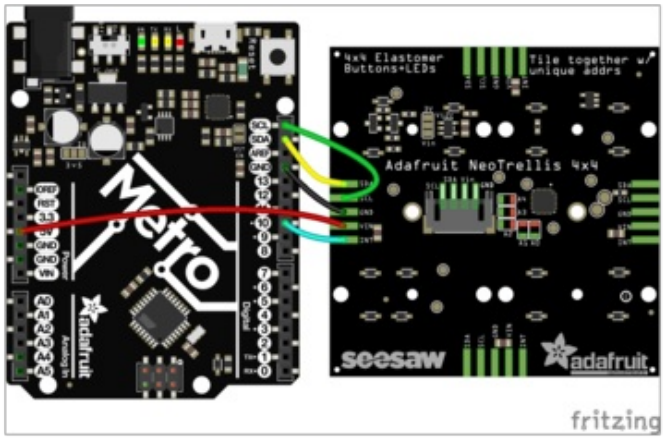


And upload it to your board!

You will see all the LEDs run a pattern in a color gradient. Then once the pattern is done, pressing any key will cause the led underneath to light up. Releasing the key will turn the LED off.

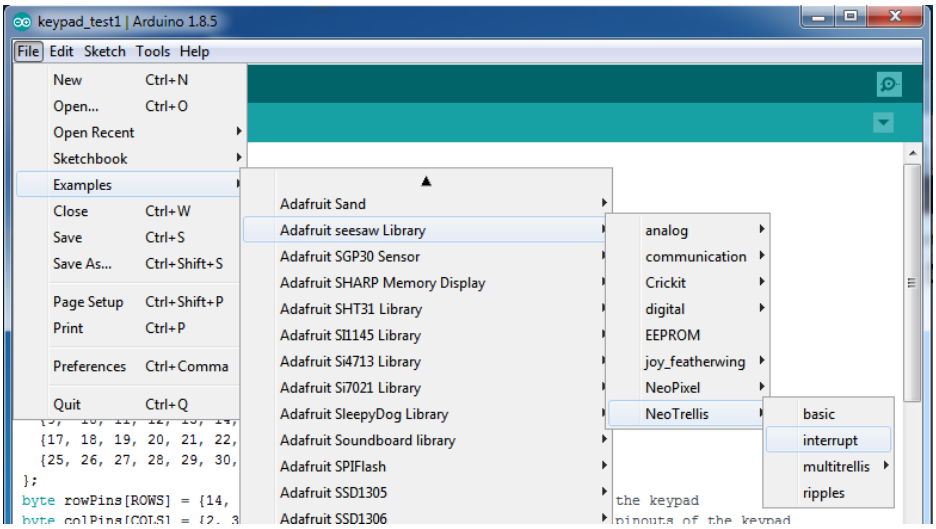
Using the Interrupt Pin

The pad labeled **INT** on the NeoTrellis board will be pulled LOW when an event the user has subscribed to is detected. To test this functionality, solder a wire to any of the four pads labeled **INT** and connect that to a digital pin on your host microcontroller board.



Use the same wiring from before, but this time connect **INT** to digital pin **#10** (you can change this later)

Open up the **NeoTrellis -> interrupt** under the Adafruit Seesaw library and upload to your board.



The behavior is identical to the **basic** example, but the INT pin is used instead of polling the NeoTrellis for new data. This frees up more time for your host microcontroller to do other things!

Tiling multiple NeoTrellis boards

Multiple NeoTrellis boards can be tiled together for a larger grid of buttons and lights without taking up any extra pins on your host microcontroller!

To do this, lay the boards you wish to connect side by side in the desired arrangement so that the pins match up. Then solder them together and set the unique I2C address on the back with solder jumpers (https://adafru.it/CzI)

The alignment of the boards does matter! The pad marked '1' should always be in the top lefthand corner.

The base I2C address for the NeoTrellis board is **0x2E** and to calculate the selected I2C address for a given jumper configuration use this formula:

```
Addr = 0x2E + (1*A0) + (2*A1) + (4*A2) + (8*A3) + (16*A4)
```

Where each A term is 1 if the jumper is soldered, and 0 if it is not. For example, if only the**A0** jumper is soldered, the I2C address would be 0x2E + 1 = 0x2F. If both the **A0** and **A1** jumper were soldered, the address would be 0x2E + 1 + 2 = 0x30.

Once your boards are connected and your I2C addresses selected, load up the **NeoTrellis -> multitrellis -> basic** example from the Adafruit Seesaw library.

Change the definitions and variables at the top of the file to match your current configuration:

```
#define Y_DIM 4 //number of rows of key
#define X_DIM 8 //number of columns of keys

//create a matrix of trellis panels
Adafruit_NeoTrellis t_array[Y_DIM/4][X_DIM/4] = {

    { Adafruit_NeoTrellis(0x2E), Adafruit_NeoTrellis(0x2F) }

};
```

The above code works for two NeoTrellis boards connected side by side. If you were using a 2x2 array of boards, the above lines would be:

```
#define Y_DIM 8 //number of rows of key
#define X_DIM 8 //number of columns of keys

//create a matrix of trellis panels
Adafruit_NeoTrellis t_array[Y_DIM/4][X_DIM/4] = {

    { Adafruit_NeoTrellis(0x2E), Adafruit_NeoTrellis(0x2F) },

    {Adafruit_NeoTrellis(YOUR_ADDR), Adafruit_NeoTrellis(YOUR_ADDR2)} }

};
```

Arduino Library Documentation

Arduino Library Documentation (https://adafru.it/CzJ)

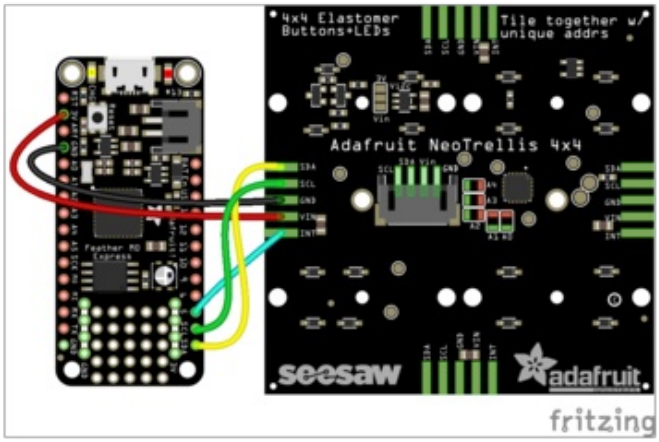
CircuitPython & Python

It's easy to use NeoTrellis panels with Python or CircuitPython and the Adafruit CircuitPython NeoTrellis (https://adafru.it/CzK) module. This module allows you to easily write Python code that reads the button presses, and then light up the LEDs

You can use NeoTrellis 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 a NeoTrellis to your board exactly as shown on the previous pages for Arduino.



Wiring is easy, solder the four **VIN GND SDA SCL** pads to wires and then connect to your CircuitPython device

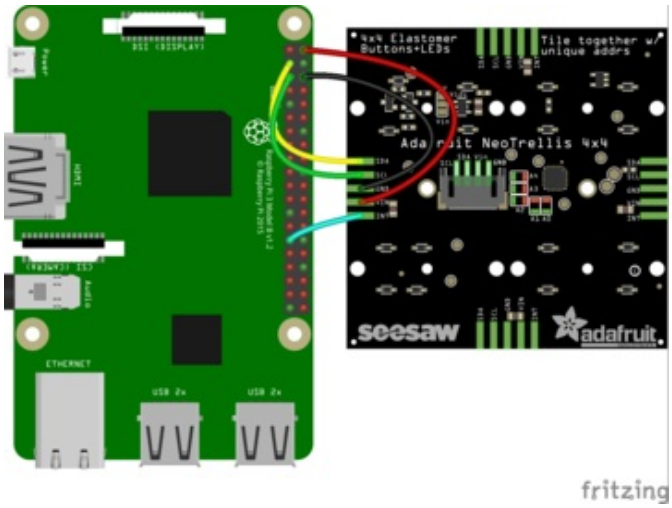
- **VIN** to **USB** (if you are going to have it always plugged into USB), **BAT** (if you're always going to have a lipo battery plugged in) or **3.3V** (if you're not sure)
- **GND** to **GND**
- **SDA** to I2C data **SDA**
- **SCL** to I2C clock **SCL**

If you want to use the interrupt pin, wire that as well to a digital I/O pin. We'll be using **#5** by default

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:



Wiring is easy, solder the four **VIN GND SDA SCL** pads to wires and then connect to your CircuitPython device

- **VIN** to **5V**
- **GND** to **GND**
- **SDA** to I2C data **SDA**
- **SCL** to I2C clock **SCL**

If you want to use the interrupt pin, wire that as well to a digital I/O pin. We'll be using GPIO **#5** by default

CircuitPython Installation of NeoTrellis Library

You'll need to install the Adafruit CircuitPython NeoTrellis (https://adafru.it/CzK) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/uap). Our CircuitPython starter guide has a great page on how to install the library bundle (https://adafru.it/ABU).

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- **adafruit_neotrellis**
- **adafruit_seesaw**
- **adafruit_bus_device**

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_seesaw, adafruit_neotrellis**, and **adafruit_bus_device** files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz) so you are at the CircuitPython >>> prompt.

Python Installation of NeoTrellis 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:

- `sudo pip3 install adafruit-circuitpython-neotrellis`

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 the usage of the board we will initialize and run an example.

If you're using an I2C connection run the following code to import the necessary modules and initialize the I2C connection with the board:

```
import time
from board import SCL, SDA
import busio
from adafruit_neotrellis.neotrellis import NeoTrellis

#create the i2c object for the trellis
i2c_bus = busio.I2C(SCL, SDA)

#create the trellis
trellis = NeoTrellis(i2c_bus)
```

Next lets define the function we want to call when a button is pressed or released:

```
OFF = (0, 0, 0)
CYAN = (0, 255, 255)

def blink(event):
    #turn the LED on when a rising edge is detected
    if event.edge == NeoTrellis.EDGE_RISING:
        trellis.pixels[event.number] = CYAN
    #turn the LED off when a rising edge is detected
    elif event.edge == NeoTrellis.EDGE_FALLING:
        trellis.pixels[event.number] = OFF
```

Now lets activate all the buttons and hook up the callbacks:

```
for i in range(16):
    #activate rising edge events on all keys
    trellis.activate_key(i, NeoTrellis.EDGE_RISING)
    #activate falling edge events on all keys
    trellis.activate_key(i, NeoTrellis.EDGE_FALLING)
    #set all keys to trigger the blink callback
    trellis.callbacks[i] = blink
```

Finally, we poll for new events in a loop:

```
while True:
    #call the sync function call any triggered callbacks
    trellis.sync()
    #the trellis can only be read every 10 millisecons or so
    time.sleep(.02)
```

This will turn on the corresponding LED when a button is being pressed, and then turn the LED off when the button is released.

Here is a complete example:

```
import time

from board import SCL, SDA
import busio
from adafruit_neotrellis.neotrellis import NeoTrellis

#create the i2c object for the trellis
i2c_bus = busio.I2C(SCL, SDA)

#create the trellis
trellis = NeoTrellis(i2c_bus)

#some color definitions
OFF = (0, 0, 0)
RED = (255, 0, 0)
YELLOW = (255, 150, 0)
GREEN = (0, 255, 0)
CYAN = (0, 255, 255)
BLUE = (0, 0, 255)
PURPLE = (180, 0, 255)

#this will be called when button events are received
def blink(event):
    #turn the LED on when a rising edge is detected
    if event.edge == NeoTrellis.EDGE_RISING:
        trellis.pixels[event.number] = CYAN
    #turn the LED off when a rising edge is detected
    elif event.edge == NeoTrellis.EDGE_FALLING:
        trellis.pixels[event.number] = OFF

for i in range(16):
    #activate rising edge events on all keys
    trellis.activate_key(i, NeoTrellis.EDGE_RISING)
    #activate falling edge events on all keys
    trellis.activate_key(i, NeoTrellis.EDGE_FALLING)
    #set all keys to trigger the blink callback
    trellis.callbacks[i] = blink

    #cycle the LEDs on startup
    trellis.pixels[i] = PURPLE
    time.sleep(.05)

for i in range(16):
    trellis.pixels[i] = OFF
    time.sleep(.05)

while True:
    #call the sync function call any triggered callbacks
    trellis.sync()
    #the trellis can only be read every 17 millisecons or so
    time.sleep(.02)
```

Connecting multiple NeoTrellis boards

Multiple NeoTrellis boards can be chained together if they are set to different I2C addresses. See the Arduino Code (https://adafru.it/CzL) section of this guide for information on how to connect the boards and set the I2C addresses, and then run this example to test your setup!

```
import time

from board import SCL, SDA
import busio
from adafruit_neotrellis.neotrellis import NeoTrellis
from adafruit_neotrellis.multitrellis import MultiTrellis

#create the i2c object for the trellis
i2c_bus = busio.I2C(SCL, SDA)

"""create the trellis. This is for a 2x2 array of NeoTrellis boards
for a 2x1 array (2 boards connected left to right) you would use:

trelli = [
    [NeoTrellis(i2c_bus, False, addr=0x2E), NeoTrellis(i2c_bus, False, addr=0x2F)]
    ]

"""
trelli = [
    [NeoTrellis(i2c_bus, False, addr=0x2E), NeoTrellis(i2c_bus, False, addr=0x2F)],
    [NeoTrellis(i2c_bus, False, addr=0x30), NeoTrellis(i2c_bus, False, addr=0x31)]
    ]

trellis = MultiTrellis(trelli)

#some color definitions
OFF = (0, 0, 0)
RED = (255, 0, 0)
YELLOW = (255, 150, 0)
GREEN = (0, 255, 0)
CYAN = (0, 255, 255)
BLUE = (0, 0, 255)
PURPLE = (180, 0, 255)

#this will be called when button events are received
def blink(xcoord, ycoord, edge):
    #turn the LED on when a rising edge is detected
    if edge == NeoTrellis.EDGE_RISING:
        trellis.color(xcoord, ycoord, BLUE)
    #turn the LED off when a rising edge is detected
    elif edge == NeoTrellis.EDGE_FALLING:
        trellis.color(xcoord, ycoord, OFF)

for y in range(8):
    for x in range(8):
        #activate rising edge events on all keys
        trellis.activate_key(x, y, NeoTrellis.EDGE_RISING)
        #activate falling edge events on all keys
        trellis.activate_key(x, y, NeoTrellis.EDGE_FALLING)
        trellis.set_callback(x, y, blink)
        trellis.color(x, y, PURPLE)
        time.sleep(.05)

for y in range(8):
    for x in range(8):
        trellis.color(x, y, OFF)
        time.sleep(.05)

while True:
```

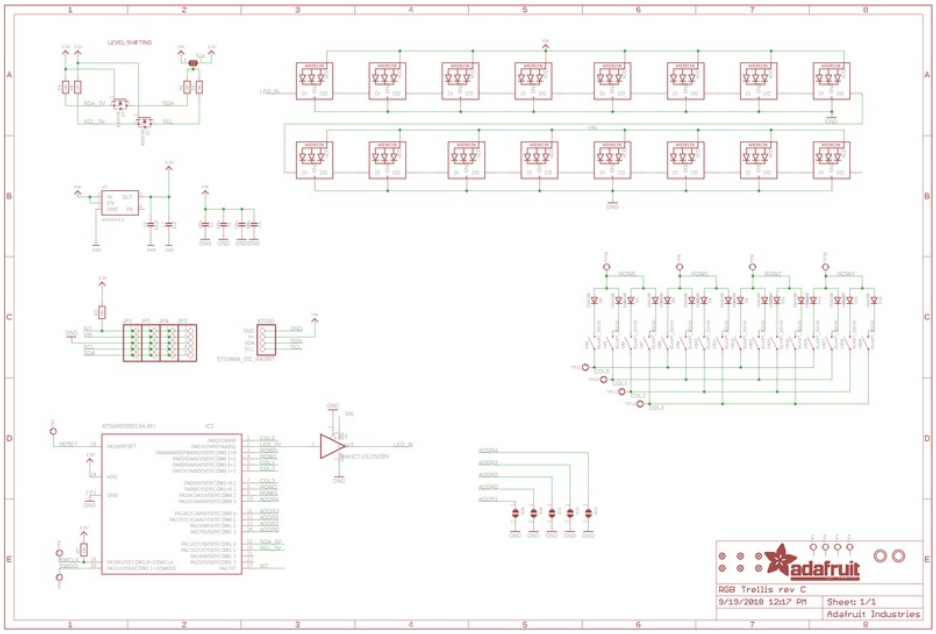
```
while True:
    #the trellis can only be read every 17 millisecons or so
    trellis.sync()
    time.sleep(.02)
```

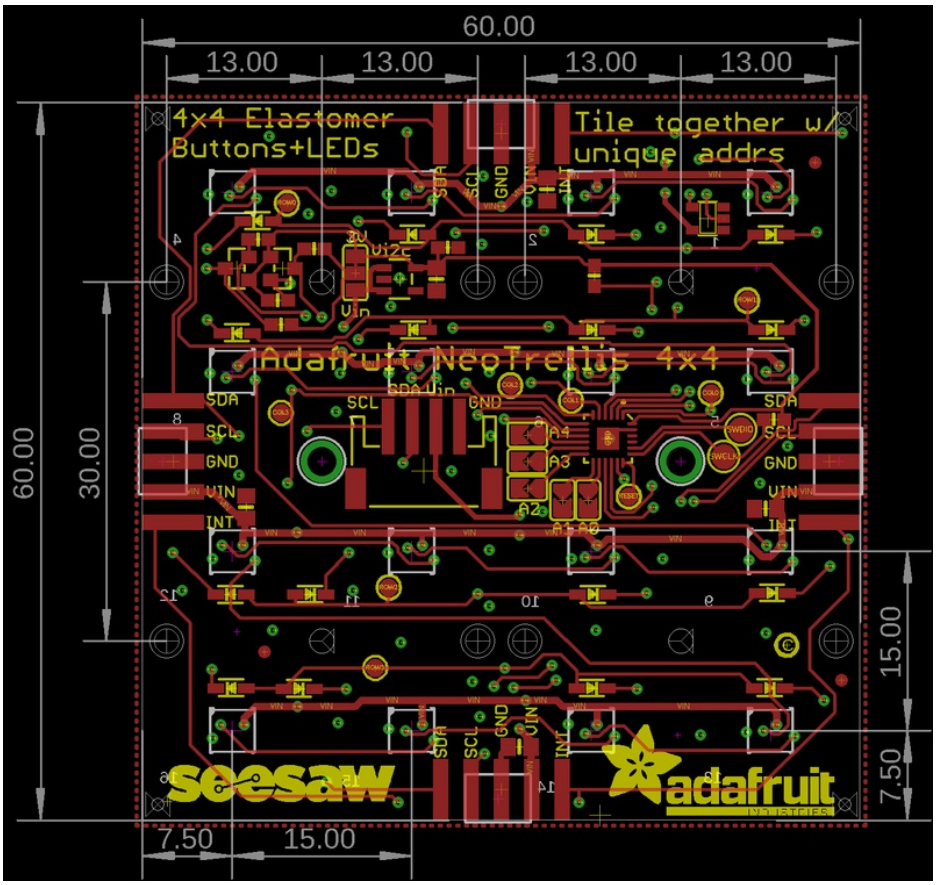
Downloads

Files:

- Fritzing object in Adafruit Fritzing library (https://adafru.it/c7M)
- EagleCAD PCB files in GitHub (https://adafru.it/Cz2)

Schematic and Fab Print




60.00
13.00
13.00
13.00
13.00
4x4 Elastomer
Buttons+LEDs
Tile together w/
unique addrs
Adafruit NeoTrellis 4x4
60.00
30.00
15.00
7.50
7.50
15.00
SDA
SCL
GND
VIN
INT
seesaw
adafruit