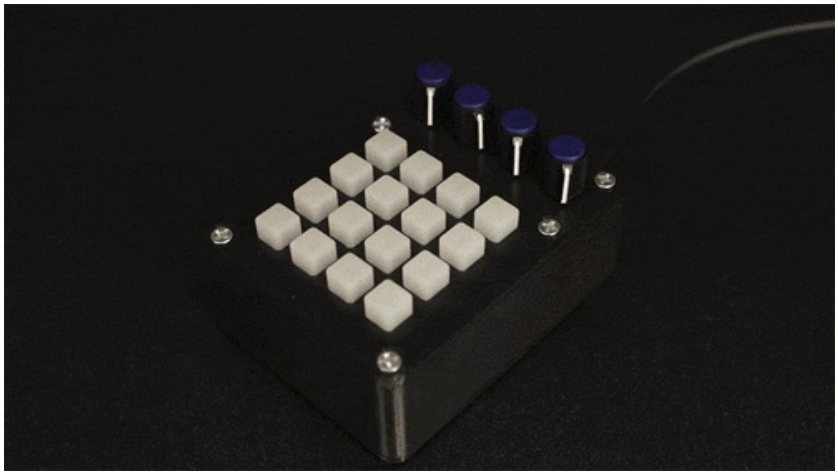
adafruit learning system

Mini OONTZ: 3D Printed MIDI Controller

Created by Noe Ruiz



Last updated on 2014-08-03 10:30:12 PM EDT

Guide Contents

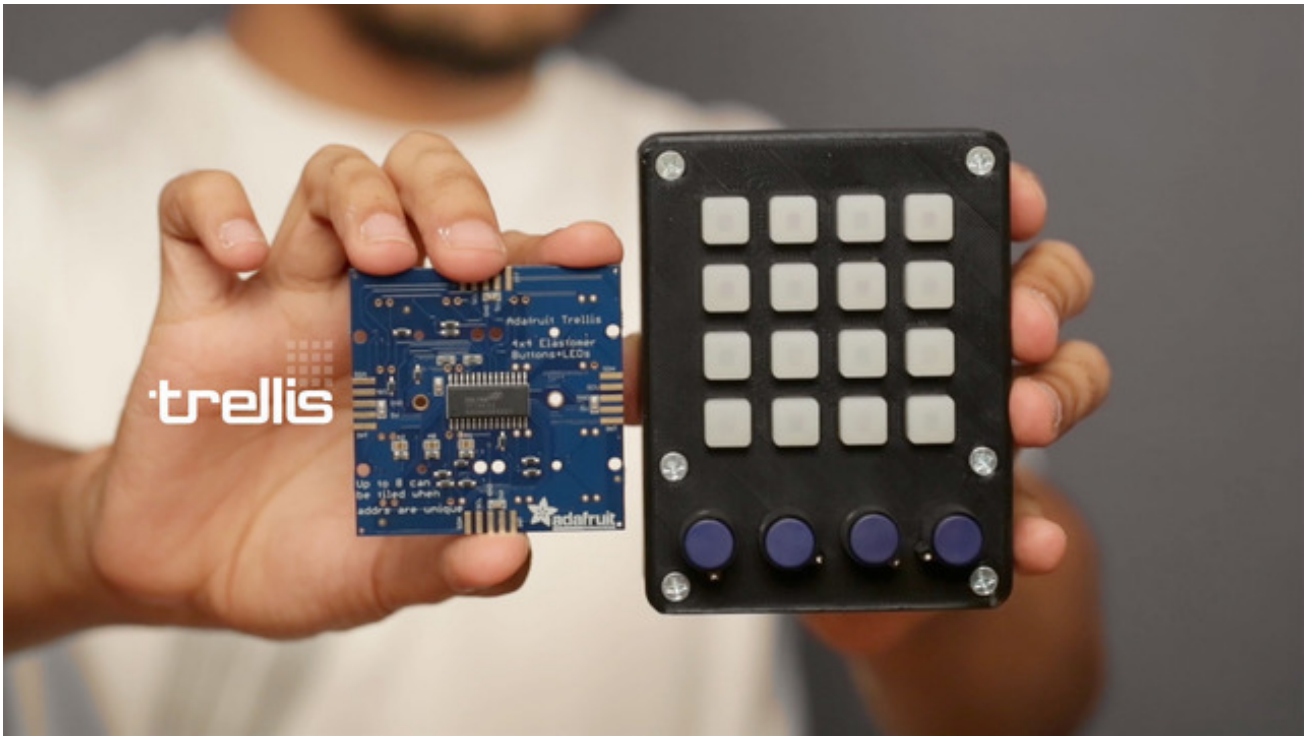
Overview

DIY MIDI Controller

Say hello to OONTZ (http://adafru.it/dLt), the open source button grid controller based on the Adafruit Trellis (http://adafru.it/dLu) button platform.

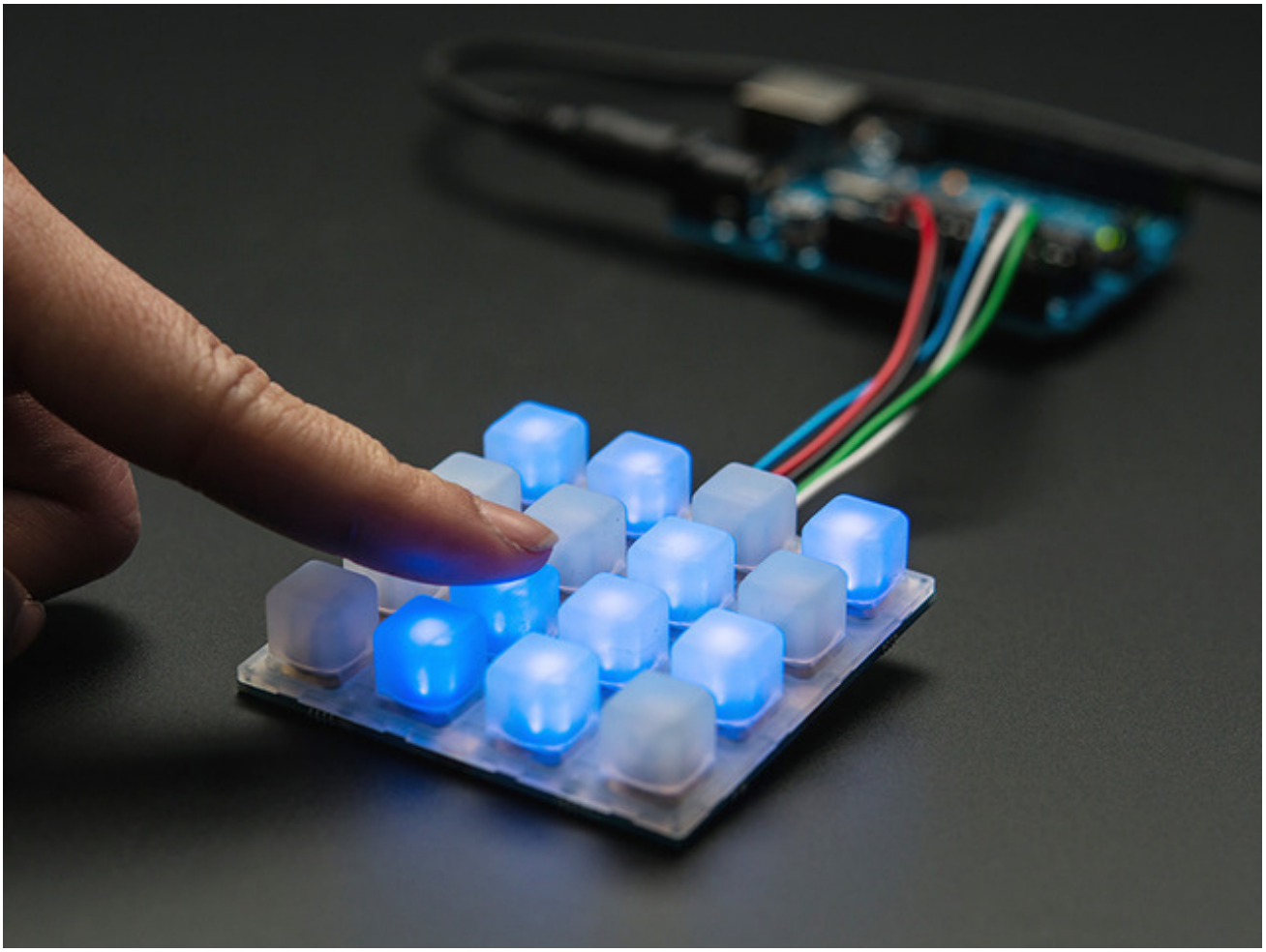
The mini OONTZ is a 3d printed case housing one trellis and 4 potentiometers ideal for a 16 button drum pad. In this project we're building a diy midi controller that works with ios devices and analog synths.

OONTZ (http://adafru.it/dLt) works with any MIDI software and hardware. No more virtual midi or serial to MIDI stuff, oontz is legit USB MIDI.



Adafruit Trellis

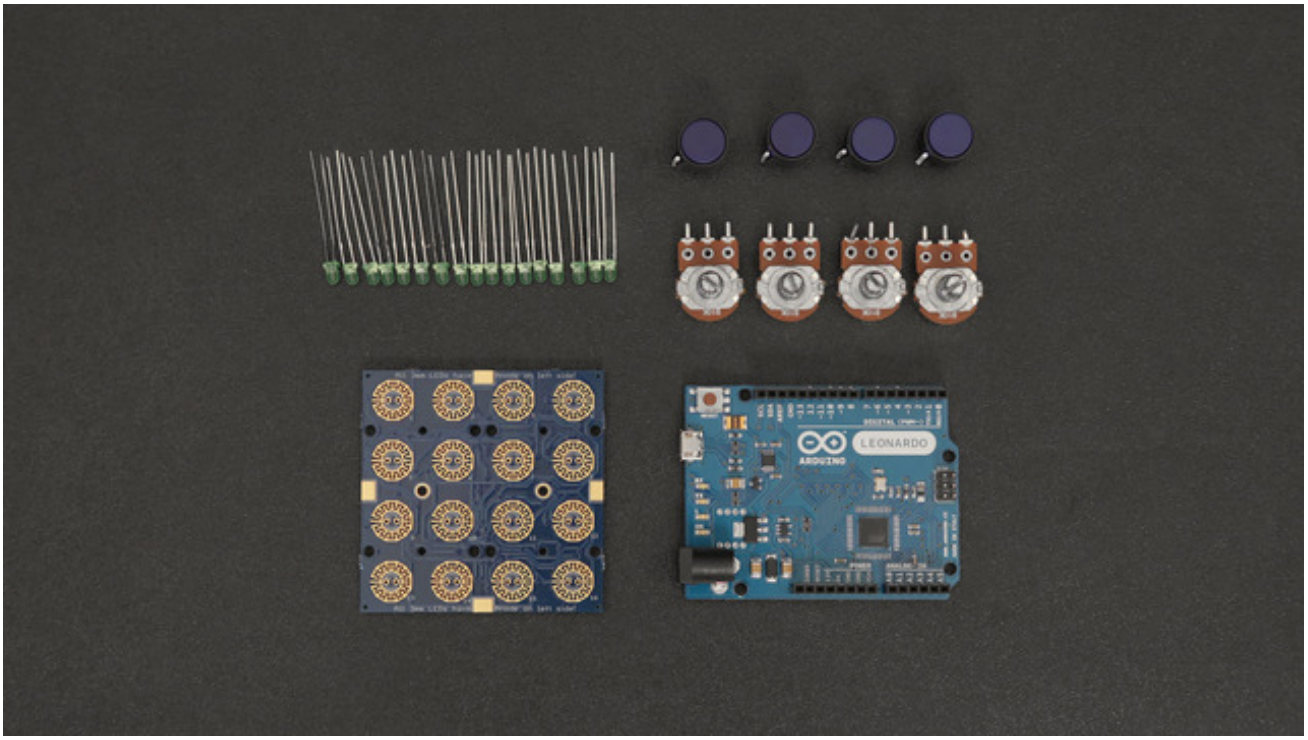
This PCB is specially made to match the Adafruit 4x4 elastomer keypad (http://adafru.it/1611). Each Trellis PCB (http://adafru.it/dLv) has 4x4 pads and 4x4 matching spots for 3mm LEDs (http://adafru.it/778). The circuitry on-board handles the background key-presses and LED lighting for the 4x4 tile. However, it does not have any microcontroller or other 'brains' - an Arduino (or similar microcontroller) is required to control the Trellis to read the keypress data and let it know when to light up LEDs as desired.



Prerequisite Guides

Before starting this project, we recommend doing a walk-through of the following guides to get you familiar with the hardware, and soldering (if your new to the craft!).

- OONTZ: a Trellis MIDI Instrument (http://adafru.it/dLw)
- Introducing Adafruit Trellis (http://adafru.it/dxx)
- Adafruit Guide to Excellent Soldering (http://adafru.it/drl)



Parts

We have all the parts in the shop for building this project. The potentiometer knobs can be 3D Printed (http://adafru.it/dLx) or get some nice ones on the internets.

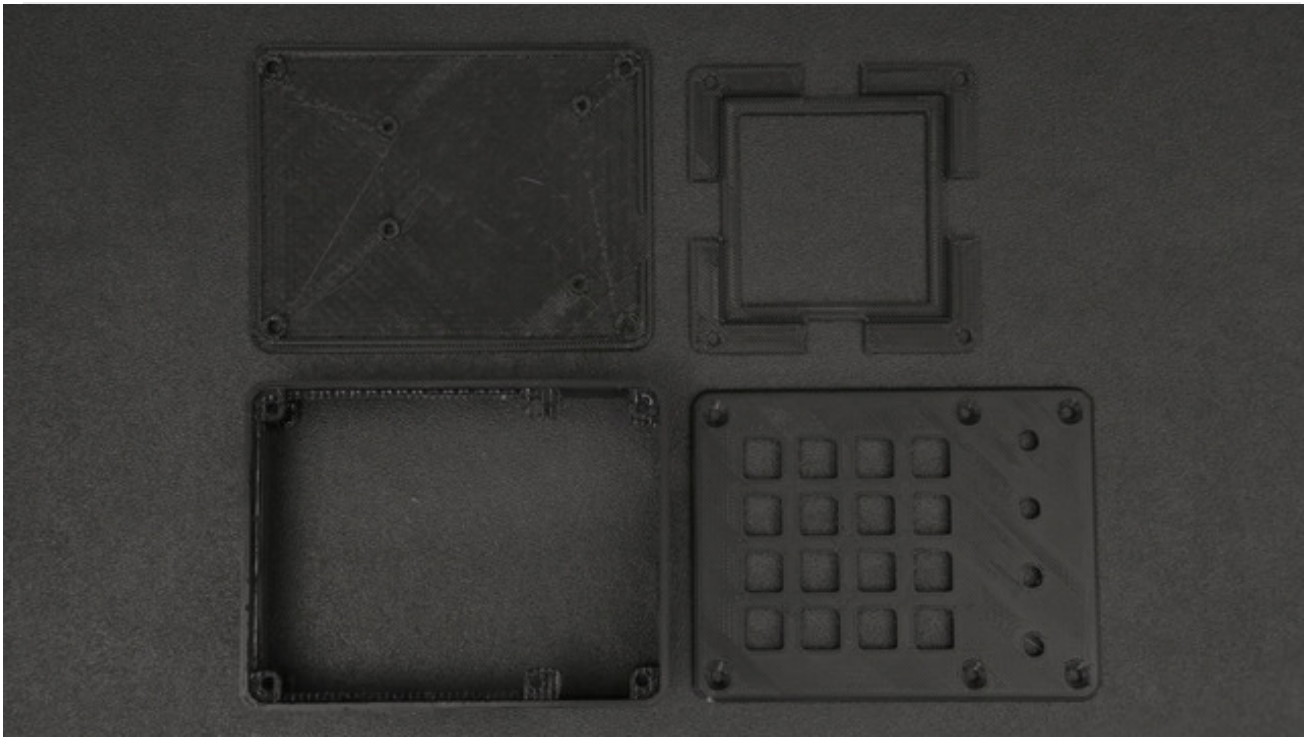
- 1 4x4 Adafruit Trellis Monochrome Driver (http://adafru.it/1616)
- 1 Silicone Elastomer 4x4 button keypad (http://adafru.it/dLy)
- 1 Arduino Leonardo (http://adafru.it/dy8)
- 4 10k Potentiometers (http://adafru.it/dzq)

Tools & Supplies

The following tools and supplies will get you started on your build.

- 3D Printer (http://adafru.it/doT)
- Soldering Iron (http://adafru.it/dLz)
- 16x 3mm LEDs (http://adafru.it/dLA)
- Flush diagonal cutters (http://adafru.it/dil)
- Premium Male Jumper Wires (http://adafru.it/759)
- 14 #6-32 1/2' Flat Phillip Screws

3D Printing



3D Printed Mini OONTZ Enclosure

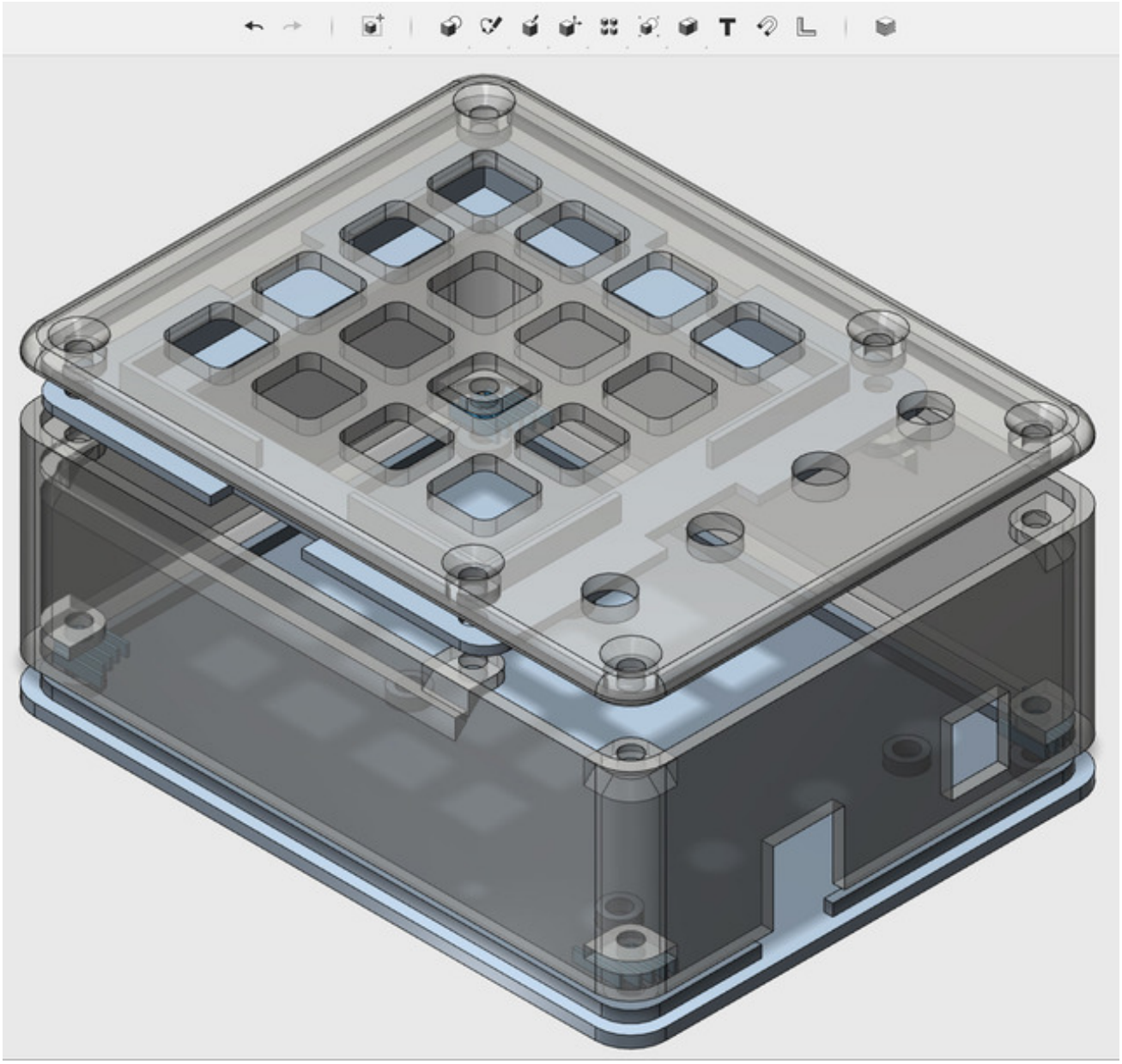
This 4 piece design houses the components and keeps them secured with machine screws. The Arduino Leonardo is mounted to the bottom cover with 4 screws. 4 potentiometers are mounted to the top panel and secured in place. The Trellis is mounted to a tray that is secured to the enclosure with another set of machine screws. A total of 14 screws makes this pretty solid and durable box.

Download STLs on Thingiverse

http://adafru.it/dLB

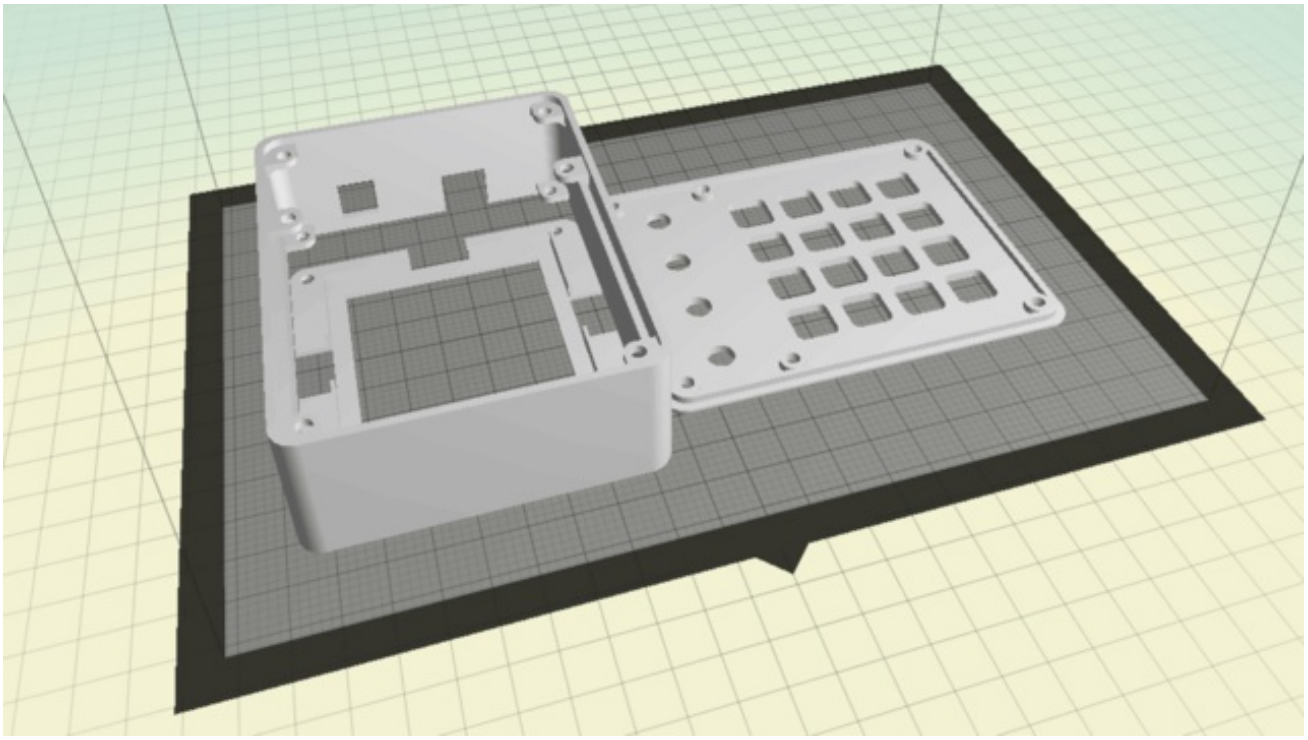
Customize Solids

If your interested in removing or adding new components, the original solids of the design are available for free to edit on Autodesk 123D.



Download Solids on AutoDesk 123D

http://adafru.it/dLC

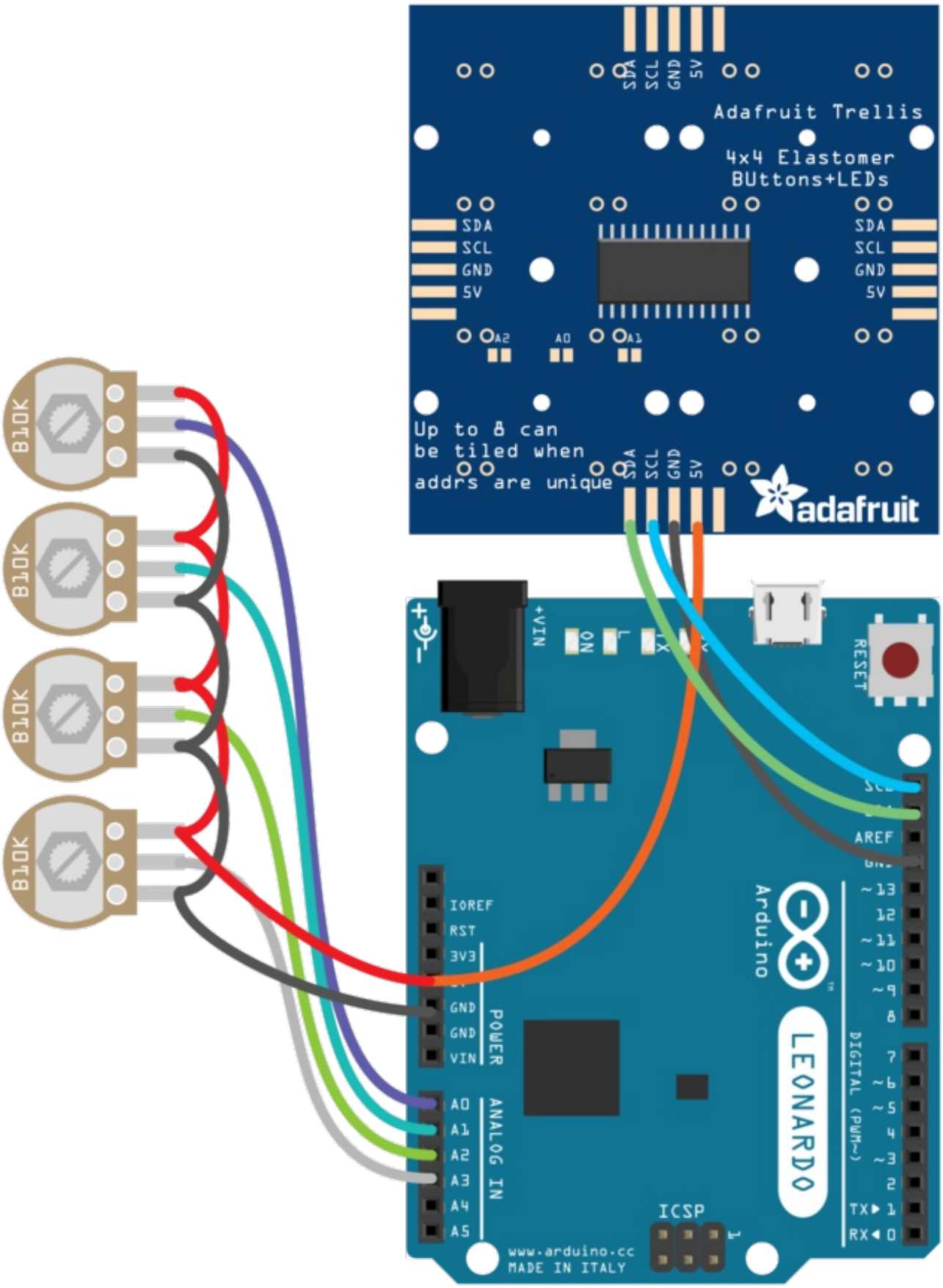


Slicer Settings

We recommend using the slice settings below. For really great quality prints, we recommend using PLA over ABS.

mini-oontz-cover.stl mini-oontz-tray.stl mini-oontz-frame.stl mini-oontz-bottom.stl	PLA@240 2 Shells 10% Infill 90/120 Speeds 0.2 Layer height	

Circuit Diagram


Adafruit Trellis
4x4 Elastomer
BUttons+LEDs
SDA
SCL
GND
5V
A2
A0
A1
Up to 8 can
be tiled when
addrs are unique
adafruit
B10K
B10K
B10K
B10K
+VIN
RESET
AREF
IOREF
RST
3V3
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
ICSP
Arduino
LEONARDO
DIGITAL (PWM~)
TX
RX
www.arduino.cc
MADE IN ITALY

Wire Connections

Follow the circuit diagram above to reference the wires are connected. You will need to solder wires to the Trellis PCB and 4 potentiometers. Use male jumper on the Trellis wires to easily connect to the headers on the Leonardo.

The Trellis PCB will have 4 wires, **SDA**, **SCL**, **GND** and **5V** that will connect the **SDA**, **SCL**, **GND** and **5V** pins on the Arduino Leonardo.

The 4 pots have common **ground** and **5V power** in series. The middle terminals connect the the **A0-A3** pins on the Arduino Leonardo.

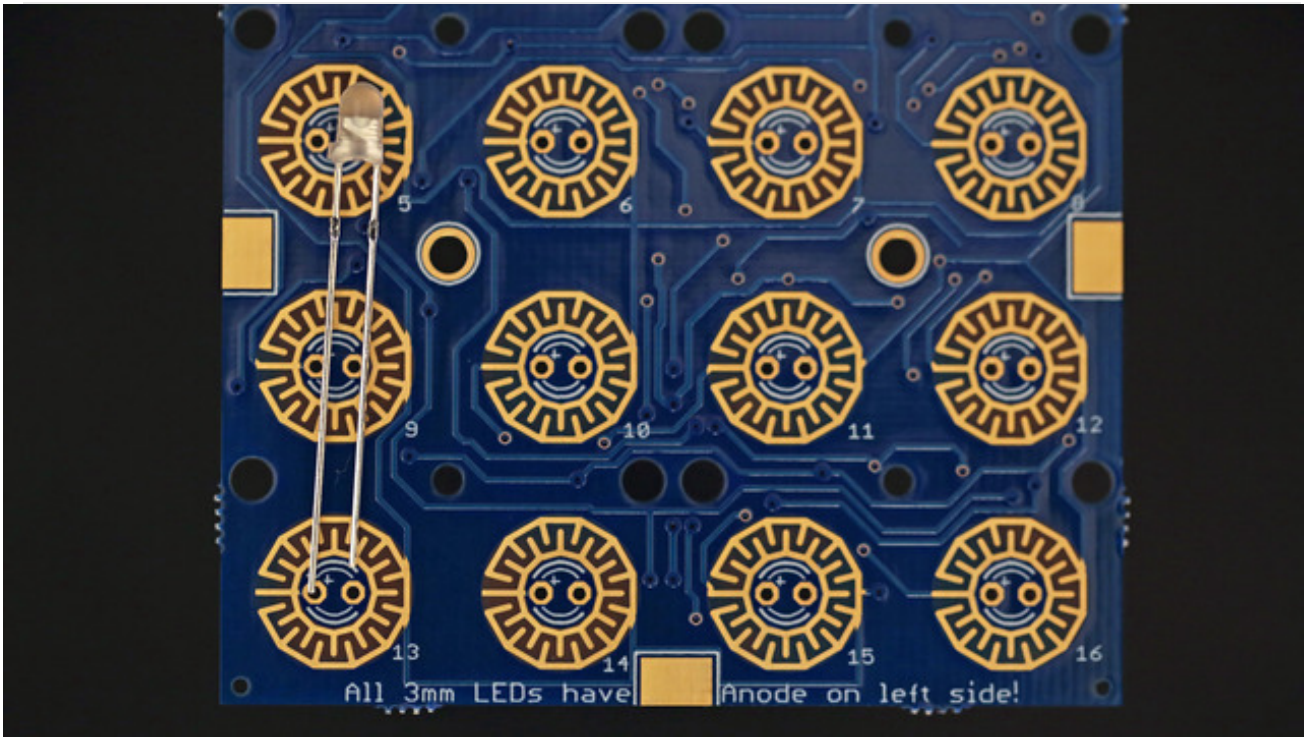
Install Trellis Library to Arduino

In order to control the trellis with the Arduino Leonardo, you will need to install the Trellis Arduino Library. Download Arduino IDE (http://adafru.it/d8v). Download the zip file (http://adafru.it/cZg) below. Add the folder to **~/Documents/Arduino/libraries/**. Extract it and rename the folder to **Adafruit_Trellis**. Restart the Arduino IDE. Goto File > Examples and check to see if **Adafruit_Trellis** is listed. If it is, move on to the next page!

Adafruit Trellis Library

http://adafru.it/cZg

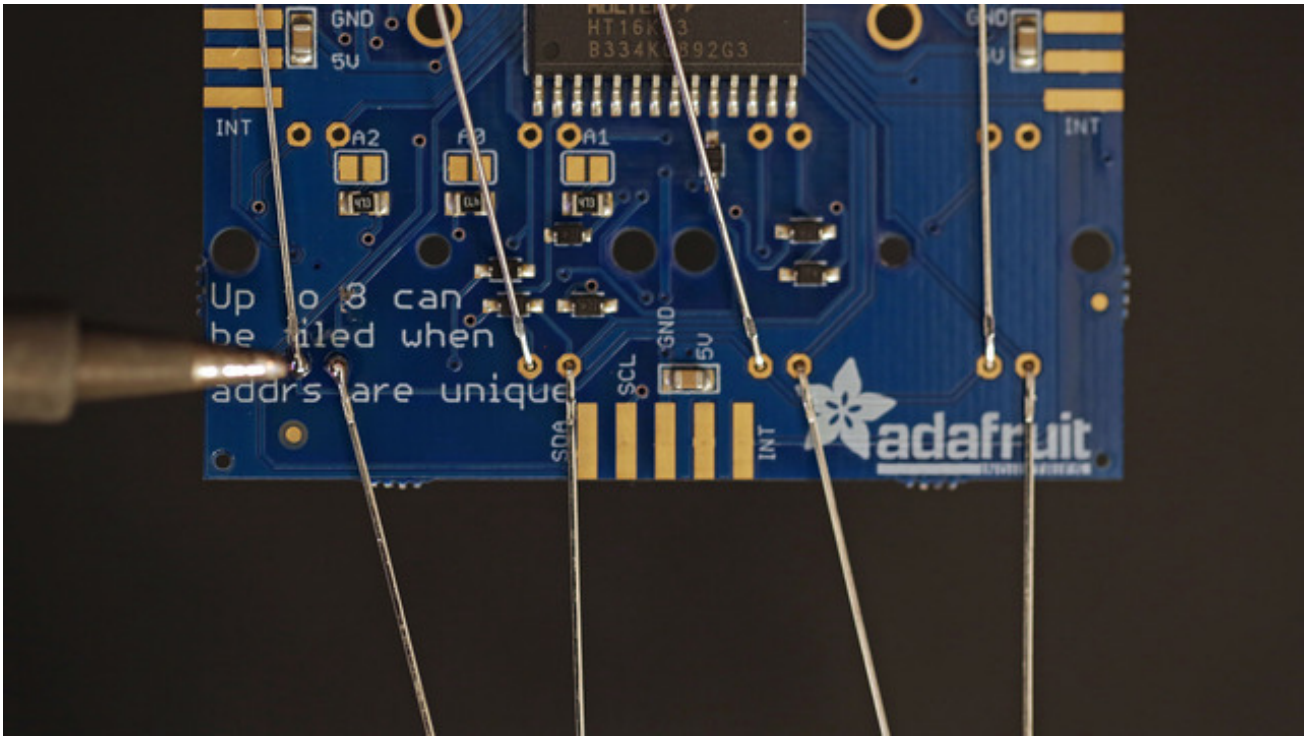
Prep Components



Insert 3mm LEDs to Trellis

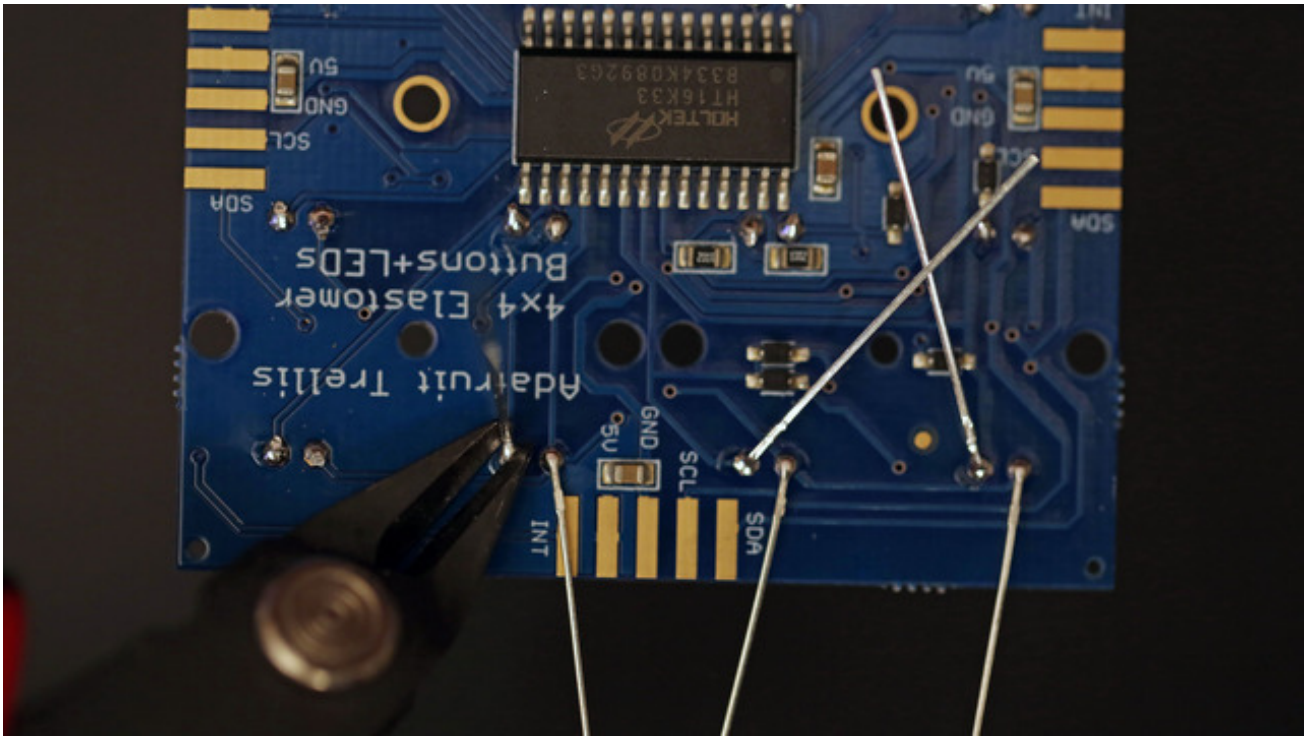
Secure the Trellis PCB to a panavise jr. or third helping hand. With the top of the facing up, insert the 3mm LED into the Trellis PCB with the longer leg going into the **+positive** pin. Bend the legs apart to secure the LED in place.

Make sure to triple check the polarity of the LEDs!



Solder LEDs to Trellis

Flip the Trellis over with the back side facing up. The LEDs should be secured in place. If not, make sure they are! Solder the 16 LEDs to the Trellis PCB.

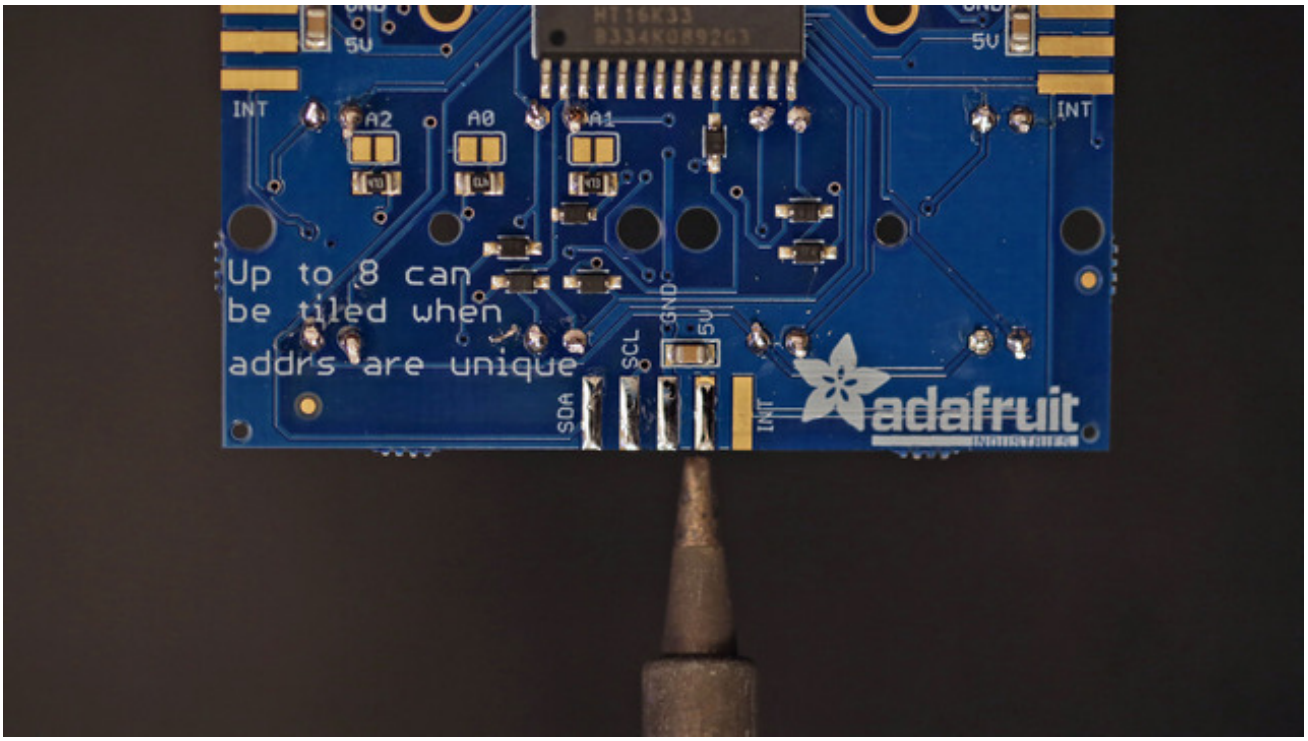


Trim Trellis LEDs

Use flush diagonal cutters to trim the excess terminals of the LEDs.

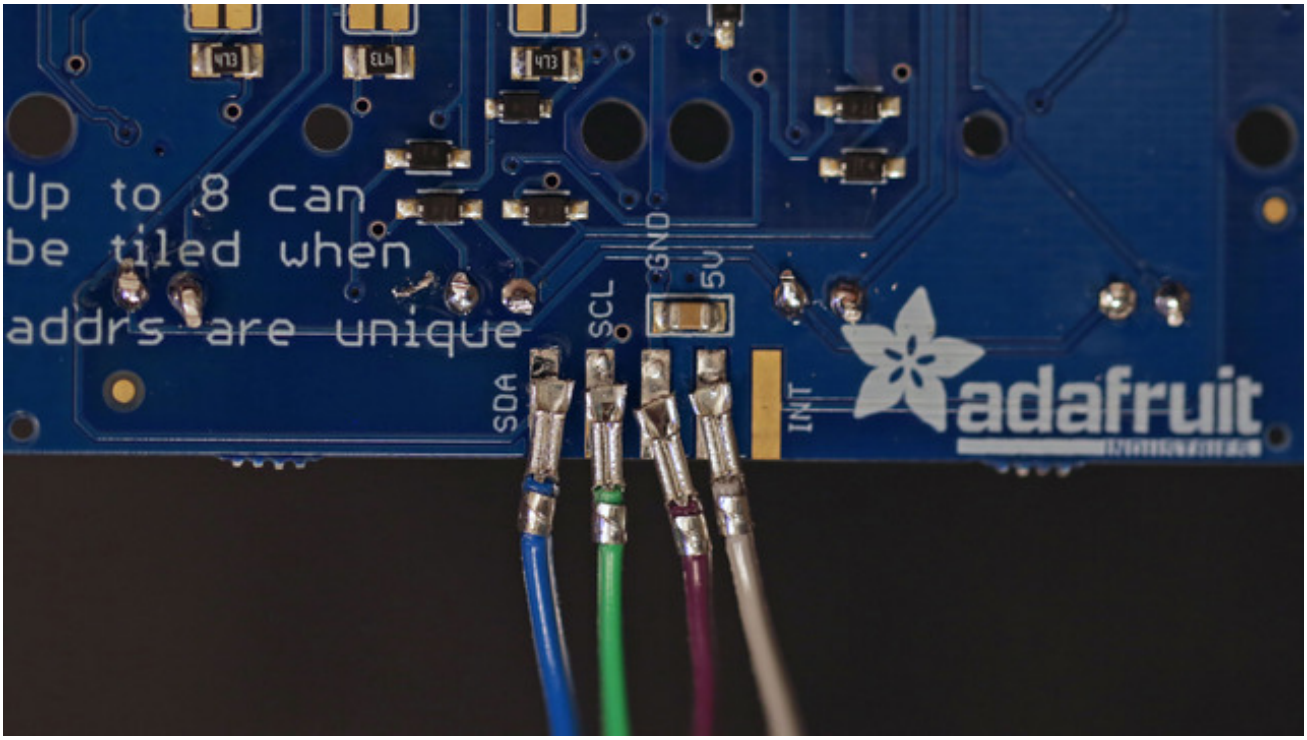
Male Jumper Wires

Grab 4 different colored male jumper wires and remove the plastic part from the one end of all 4 jumper wires. Trim off half of the exposed terminal, leaving just a small portion.



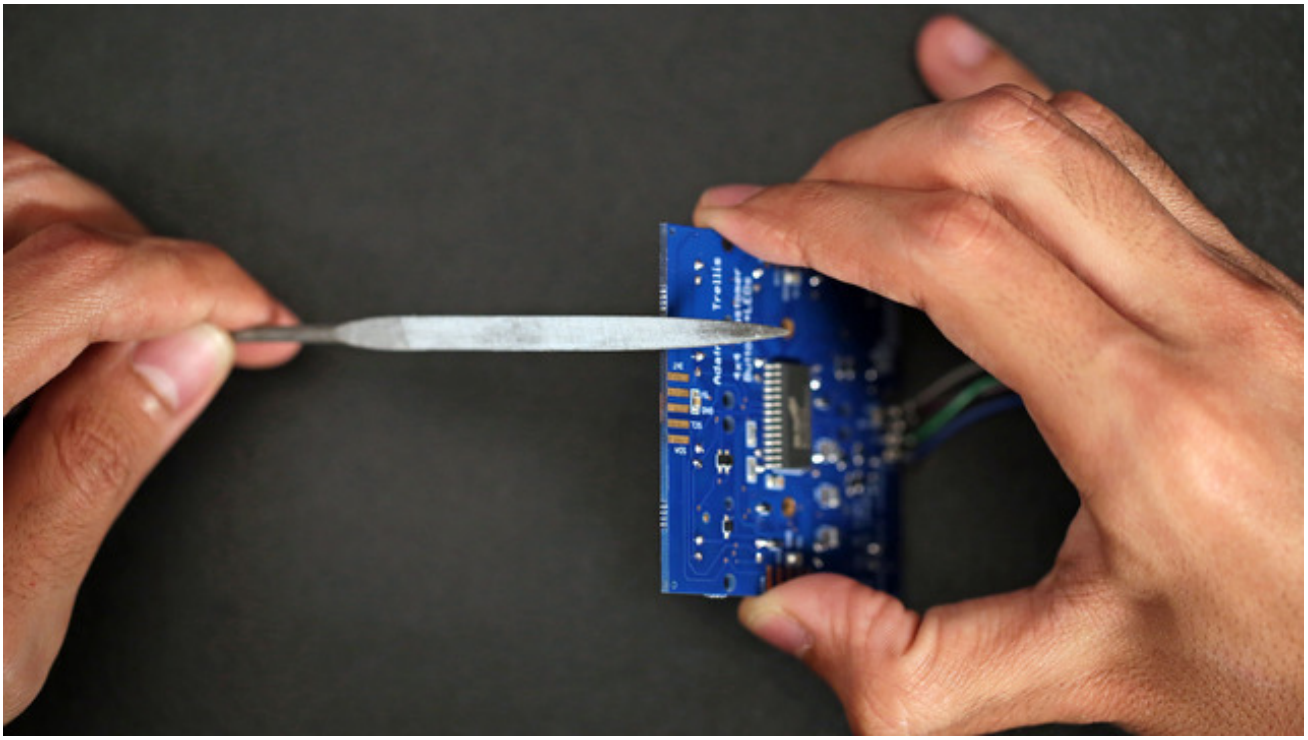
Tin Trellis PCB

Tin the **GND**, **SLC**, **SDA** and **5V** pads on Trellis PCB. There are four different groups of each, it doesn't matter which one as long as they are in the same group. In this project, we're using the one closest to the Adafruit logo.



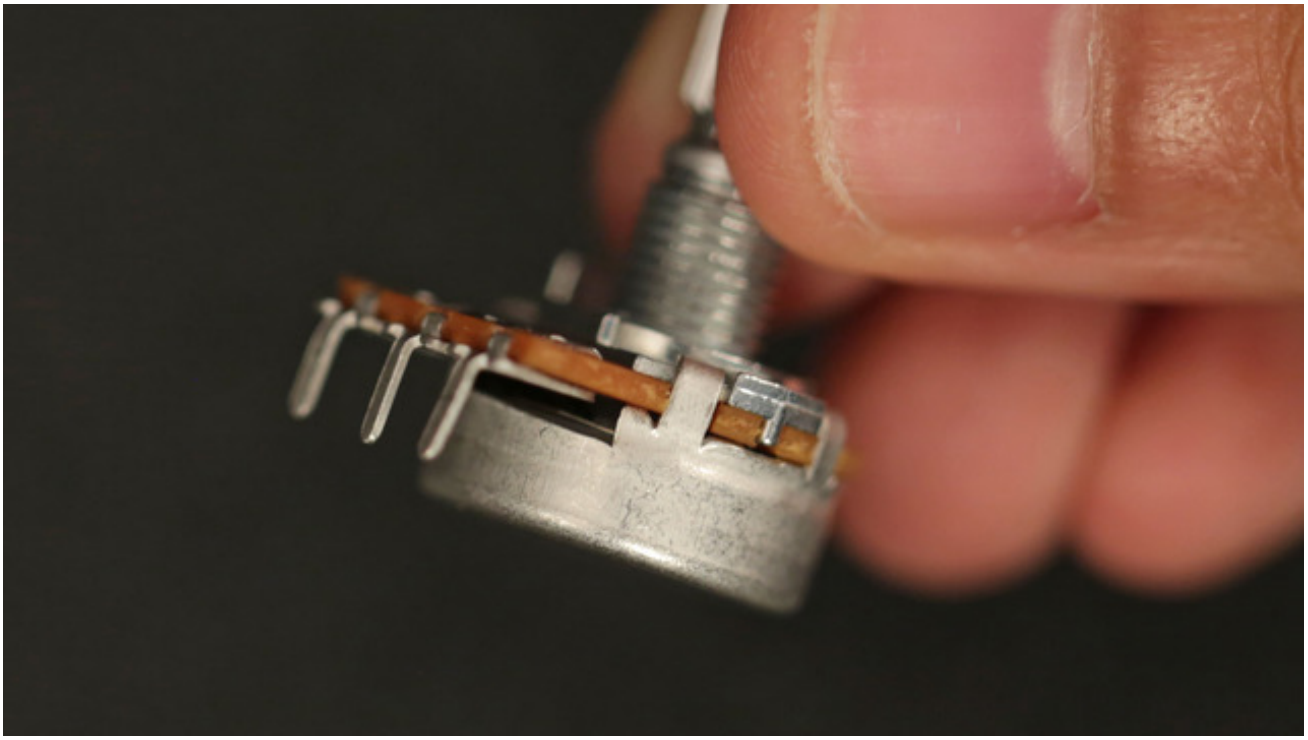
Solder Wires to Trellis PCB

Use a panavise Jr. to secure the Trellis PCB. Solder the 4 male jumper wires to the **GND**, **SCL**, **SDA** and **5V** pads on the Trellis PCB.



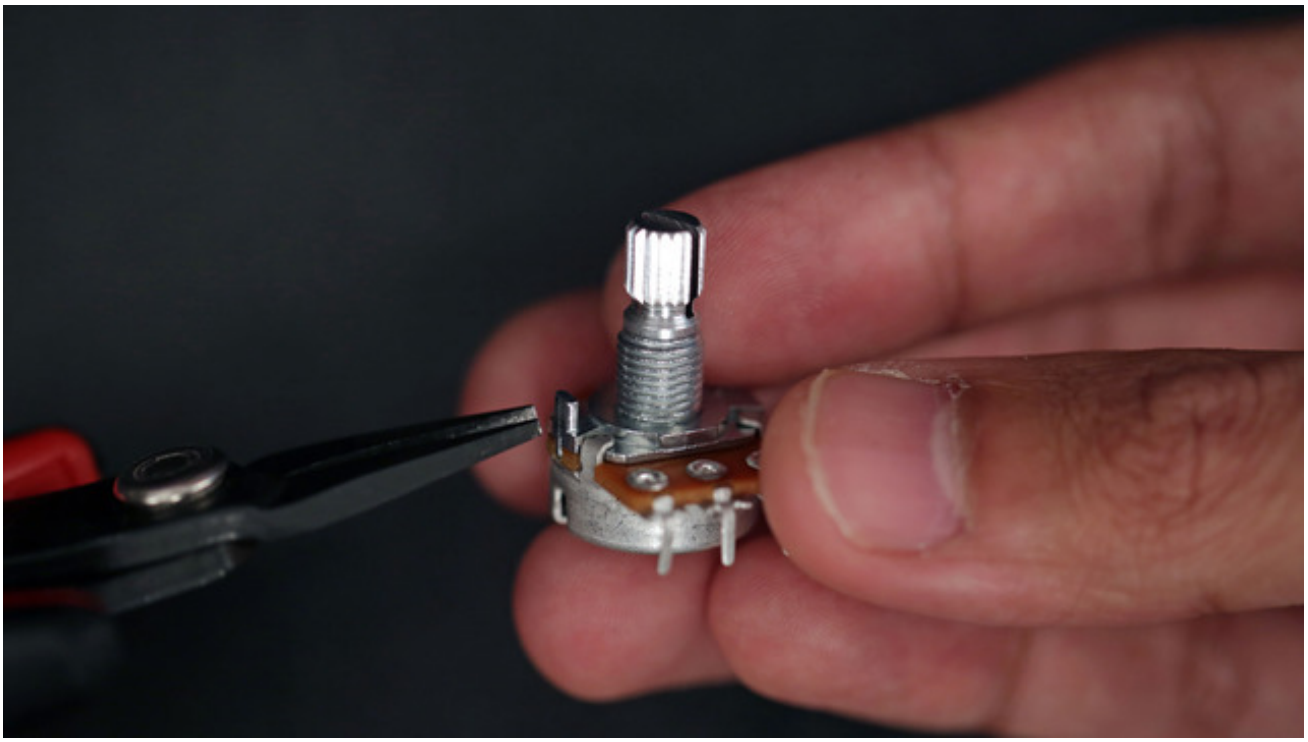
File Trellis Edges

In order to fit the Trellis into the tray, you will need to file down the edges to remove this excess material. There's two small parts on each side.



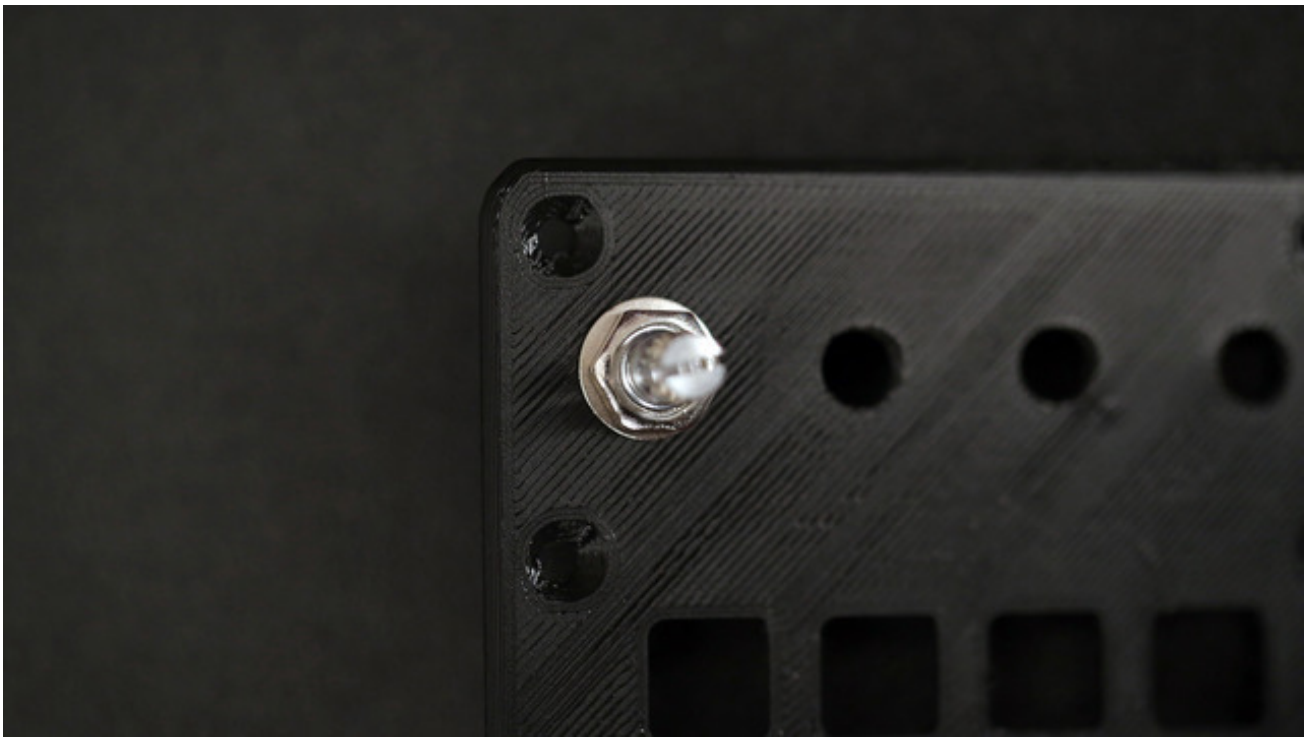
10k Potentiometers

Bend the three terminals on each of the 10k potentiometers like in the photo.



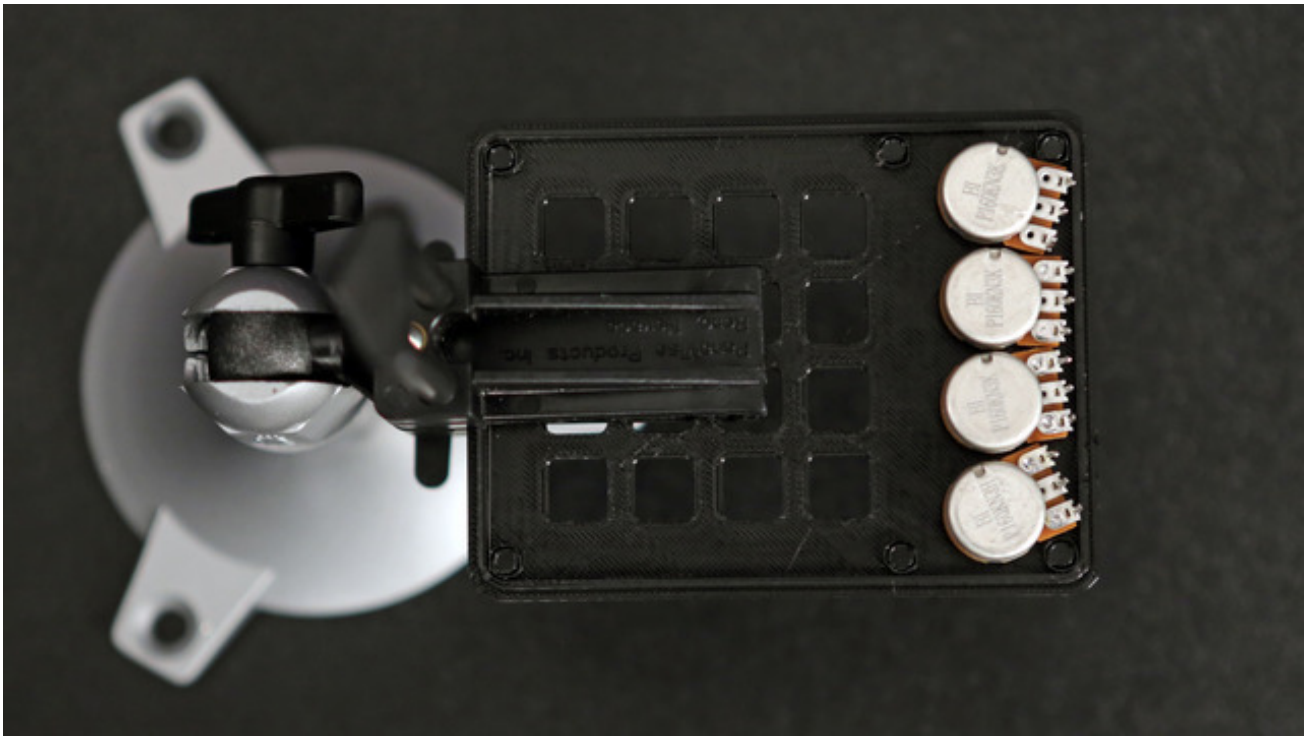
Remove 'stubs' from Pots

A small metal 'stub' on top of the base prevents the potentiometer from being flush with the cover. Remove this metal tip by bending it off with flat pliers.



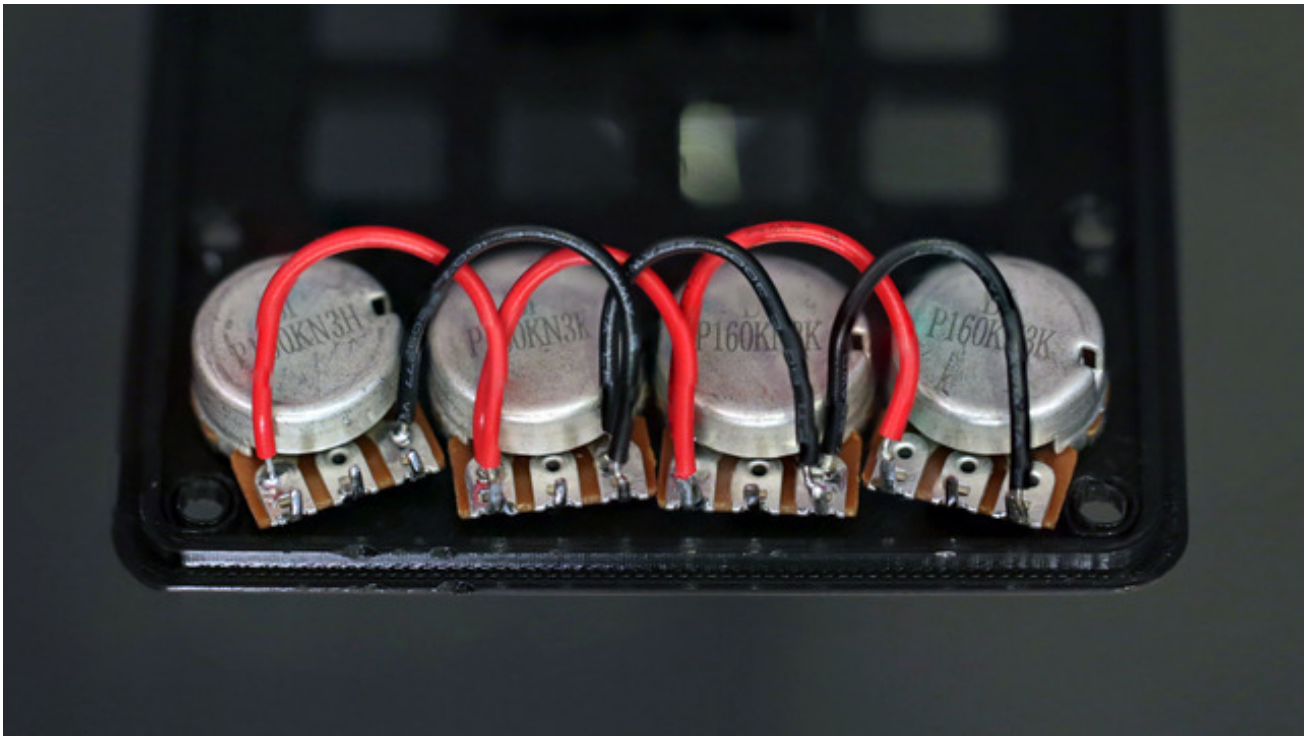
Add Pots to Enclosure Cover

Install the four potentiometers to the **mini-oontz-cover.stl** part with the knob facing the printed surface of the cover. They should pop into place. If the tolerance are too tight, use an x-acto knife or dremel to open up the mounting holes.



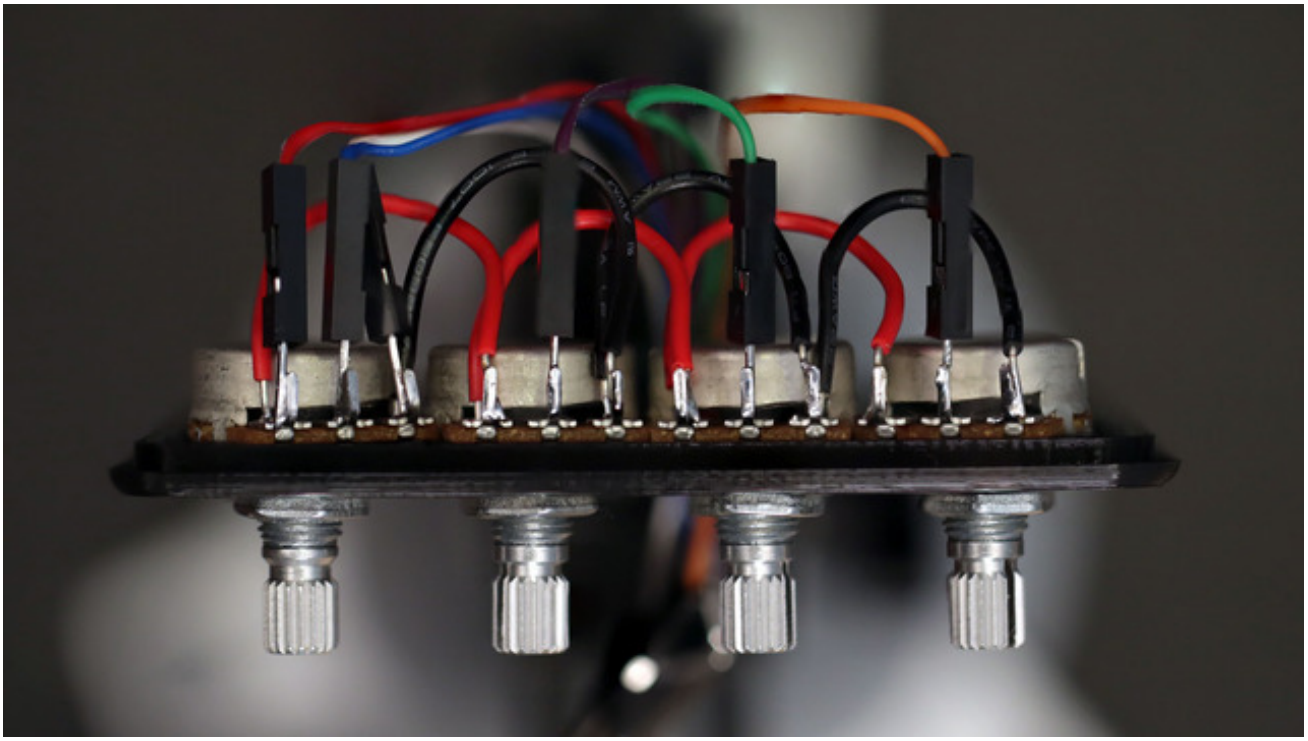
Secure Cover with Pots

You'll need to wire up the 4 pots so they share common **ground** and **5V**. Use a panavise jr. to secure the **mini-oontz-cover.stl** part that has the 4 pots installed.



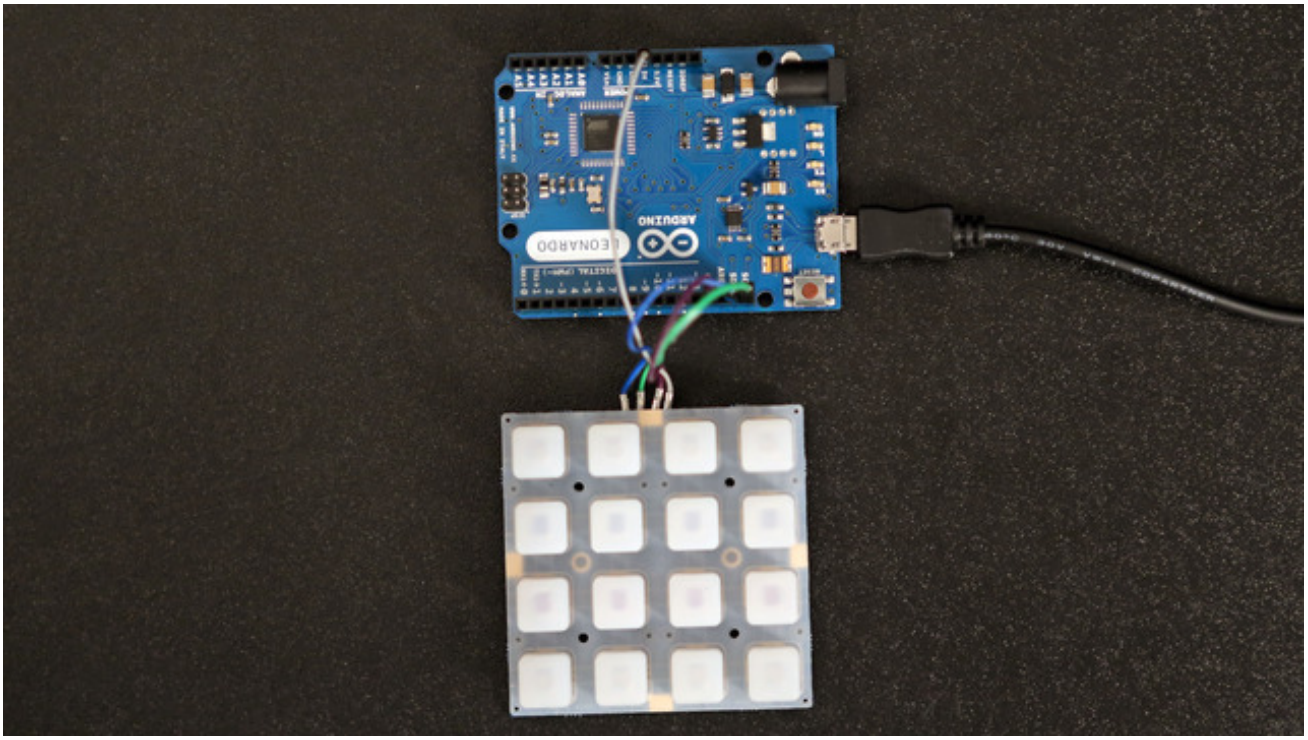
Wire common ground and 5V on Pots

Use 22AWG solid core wire to connect the common ground and 5V on the 4 pots together in series. Use a third-helping hand to secure the 22 gauge wire close to the pot terminal leads and solder in place.



Solder Jumper Wires to Pots

Solder one jumper wire to each of the middle terminals of the 4 potentiometers. This makes it easier to arrange and connect the wires.



Connect Trellis to Leonardo

Hook up the jumper wires on the trellis to the headers on the Leonardo to make the following connections:

- **SCL** to **SCL**
- **SDA** to **SDA**
- **GND** to **GND**
- **5V** to **5V**

Test Trellis

With the Trellis connect to the Leonardo, you can now upload a test sketch to check the polarity of the LEDs are all good. Copy + Paste the code below into a new sketch. Select Tools > Board > **Arduino Leonardo** in the top menu. Ensure programmer is set to **USBTinyISP**. Plug a USB micro cable into the Leonardo and connect it to the USB port of your computer, then hit the **upload** arrow button.

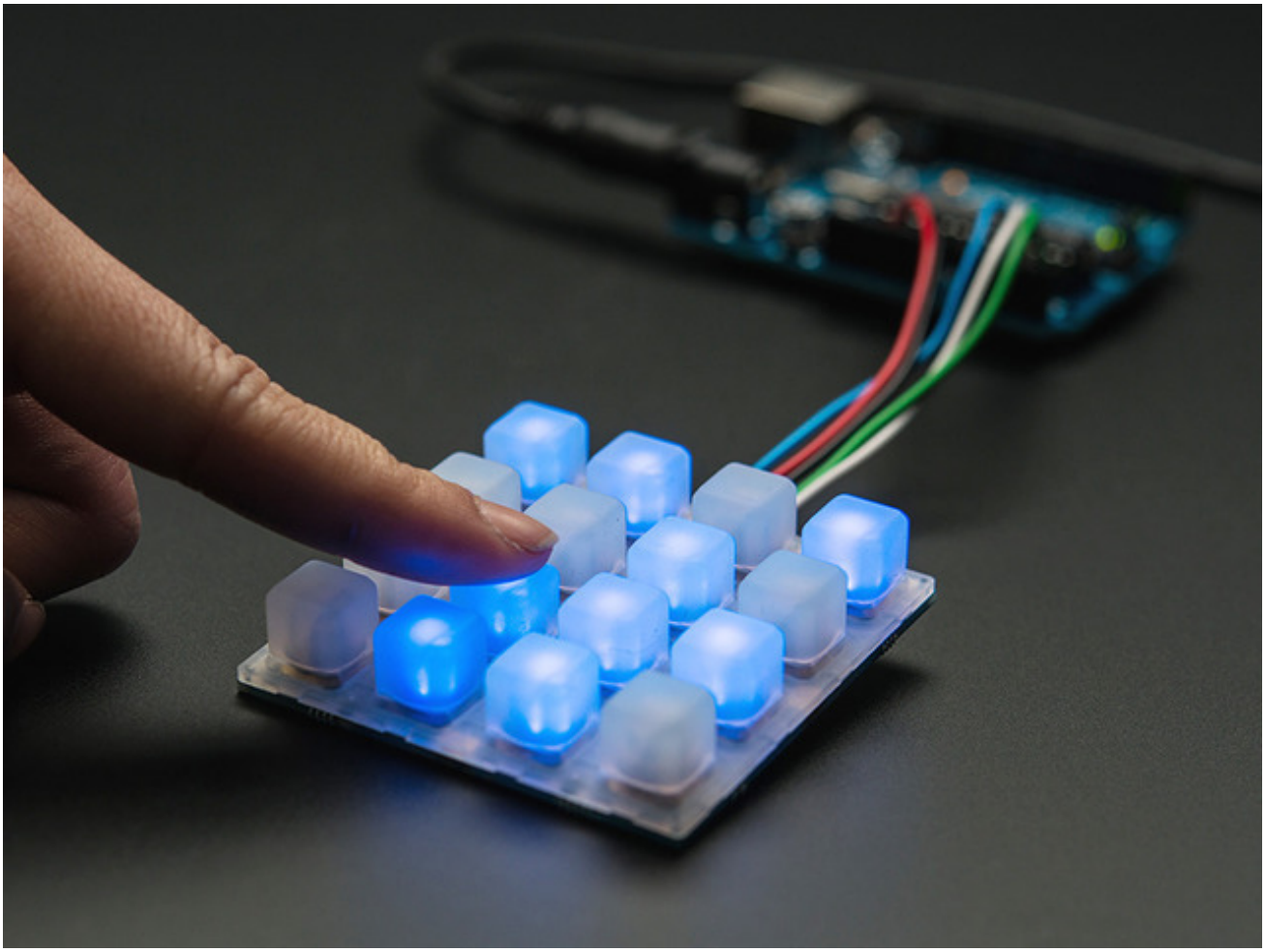
Install Trellis Arduino Library before uploading code.

```
/*************************************************
  This is a test example for the Adafruit Trellis w/HT16K33

  Designed specifically to work with the Adafruit Trellis
  ----> https://www.adafruit.com/products/1616
  ----> https://www.adafruit.com/products/1611

  These displays use I2C to communicate, 2 pins are required to
  interface
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 *************************************************/

#include <Wire.h>
#include "Adafruit_Trellis.h"

/*************************************************
  This example shows reading buttons and setting/clearing buttons in a loop
  "momentary" mode has the LED light up only when a button is pressed
  "latching" mode lets you turn the LED on/off when pressed

  Up to 8 matrices can be used but this example will show 4 or 1
 *************************************************/

#define MOMENTARY 0
#define LATCHING 1
// set the mode here
#define MODE LATCHING


Adafruit_Trellis matrix0 = Adafruit_Trellis();

// uncomment the below to add 3 more matrices
/*
Adafruit_Trellis matrix1 = Adafruit_Trellis();
Adafruit_Trellis matrix2 = Adafruit_Trellis();
Adafruit_Trellis matrix3 = Adafruit_Trellis();
// you can add another 4, up to 8
*/

// Just one
Adafruit_TrellisSet trellis =  Adafruit_TrellisSet(&matrix0);
// or use the below to select 4, up to 8 can be passed in
```

```
//Adafruit_TrellisSet trellis =  Adafruit_TrellisSet(&matrix0, &matrix1, &matrix2, &matrix3);

// set to however many you're working with here, up to 8
#define NUMTRELLIS 1

#define numKeys (NUMTRELLIS * 16)

// Connect Trellis Vin to 5V and Ground to ground.
// Connect the INT wire to pin #A2 (can change later!)
#define INTPIN A2
// Connect I2C SDA pin to your Arduino SDA line
// Connect I2C SCL pin to your Arduino SCL line
// All Trellises share the SDA, SCL and INT pin!
// Even 8 tiles use only 3 wires max


void setup() {
  Serial.begin(9600);
  Serial.println("Trellis Demo");

  // INT pin requires a pullup
  pinMode(INTPIN, INPUT);
  digitalWrite(INTPIN, HIGH);

  // begin() with the addresses of each panel in order
  // I find it easiest if the addresses are in order
  trellis.begin(0x70);  // only one
  // trellis.begin(0x70, 0x71, 0x72, 0x73);  // or four!

  // light up all the LEDs in order
  for (uint8_t i=0; i<numKeys; i++) {
    trellis.setLED(i);
    trellis.writeDisplay();
    delay(50);
  }
  // then turn them off
  for (uint8_t i=0; i<numKeys; i++) {
    trellis.clrLED(i);
    trellis.writeDisplay();
    delay(50);
  }
}


void loop() {
  delay(30); // 30ms delay is required, dont remove me!
```

```
 if (MODE == MOMENTARY) {
   // If a button was just pressed or released...
   if (trellis.readSwitches()) {
     // go through every button
     for (uint8_t i=0; i<numKeys; i++) {
// if it was pressed, turn it on
if (trellis.justPressed(i)) {
  Serial.print("v"); Serial.println(i);
  trellis.setLED(i);
}
// if it was released, turn it off
if (trellis.justReleased(i)) {
  Serial.print("^"); Serial.println(i);
  trellis.clrLED(i);
}
     }
     // tell the trellis to set the LEDs we requested
     trellis.writeDisplay();
   }
 }

 if (MODE == LATCHING) {
   // If a button was just pressed or released...
   if (trellis.readSwitches()) {
     // go through every button
     for (uint8_t i=0; i<numKeys; i++) {
       // if it was pressed...
if (trellis.justPressed(i)) {
  Serial.print("v"); Serial.println(i);
  // Alternate the LED
  if (trellis.isLED(i))
    trellis.clrLED(i);
  else
    trellis.setLED(i);
       }
     }
     // tell the trellis to set the LEDs we requested
     trellis.writeDisplay();
   }
 }
}
```

Confirm LEDs are working on Trellis



TrellisTest Sketch

If everything is correct (trellis arduino libraries installed in the right place, polarity of LEDs, etc), you should see a short animation sequence of the LED's lighting up. As the comments note: This example shows reading buttons and setting/clearing buttons in a loop "momentary" mode has the LED light up only when a button is pressed "latching" mode lets you turn the LED on/off when pressed.

Software

Install OONTZ + Trellis Arduino Libraries

Follow the guide on how to setup the arduino IDE with the special OONTZ libraries that will allow you to program the Leonardo to perform like a native USB MIDI device.

Install Arduino Libraries

http://adafru.it/dz5

Install Teensyduino to Mod Arduino IDE

You will need to install the Tennsyduino for your operating system. It's a special damon that updates your install of the Arduino IDE to have that special menu for uploading MIDI code to the Leonardo. Click the link below to get instructions on installing Teensyduino.

Install Teensyduino for Arduino

http://adafru.it/dy3

Ensure the TeeOnArdu + OONTZ library are installed!

Uploading Sketches to Leonardo MIDI

In certain cases, once the Arduino Leonardo has the MIDI uploaded, it won't be able to **upload new code** until the **reset button** is pressed. Keep this in mind when attempting to upload new sketches.

MIDI Loop

The code below uses the **usbMIDI.sendNoteOn** and **usbMIDI.sendNoteOff** calls for each button to change the state of the MIDI note. The note[i] variable is set in the table. 127 is the velocity of the midi note, being the highest/hardest. Channel is a reference that is defined in the top of the sketch.

```
for(uint8_t i=0; i<16; i++) { // For each button...
      if(trellis.justPressed(i)) {
        usbMIDI.sendNoteOn(note[i], 127, CHANNEL);

        trellis.setLED(i);
      } else if(trellis.justReleased(i)) {
        usbMIDI.sendNoteOff(note[i], 0, CHANNEL);
        trellis.clrLED(i);
      }
```

```
    }
```

MIDI Note Mapping

This table lists the midi notes that are mapped the 16 buttons on the Trellis. The values are ordered just like the 4x4 grid on the Trellis. "60" is equal to C3, while "48" will trigger note C2. The full range of notes in this table are visually represented below. This map syncs perfectly with the TRG-16 performance pad in NanoStudio (http://adafru.it/dLD). For a list of the flu range of available notes, check out this page (http://adafru.it/dLE).

```
uint8_t note[] = {
  60, 61, 62, 63,
  56, 57, 58, 59,
  52, 53, 54, 55,
  48, 49, 50, 51
};
```

Potentiometers MIDI CC

The 4 potentiometers are mapped to MIDI CC **1**, **11**, **12** and **13.** The blocks of code below use the **usbMIDI.sendControlChange** call to define which potentiometer will be mapped to a MIDI CC. The **analogRead(#)** call refers to the analog input on the Arduino Leonardo.

```
 mod = map(analogRead(0), 0, 1023, 0, 127);
 vel = map(analogRead(1), 0, 1023, 0, 127);
 fxc = map(analogRead(2), 0, 1023, 0, 127);
 rate = map(analogRead(3),0, 1023, 0, 127);
 usbMIDI.sendControlChange(1, mod, CHANNEL);
 usbMIDI.sendControlChange(11, vel, CHANNEL);
 usbMIDI.sendControlChange(12, fxc, CHANNEL);
 usbMIDI.sendControlChange(13, rate, CHANNEL);
```

pot = map(ArdiunoPin(#)), 0(lowest MIDI value), 1023(highest MIDI value);
usbMIDI.sencControlChange(Arduino Pin, pot, CHANNEL);

```
uint8_t newModulation = map(analogRead(0), 0, 1023, 0, 127);
    if(mod != newModulation) {
      mod = newModulation;
      usbMIDI.sendControlChange(1, mod, CHANNEL);
    }
```

MINI OONTZ Sketch

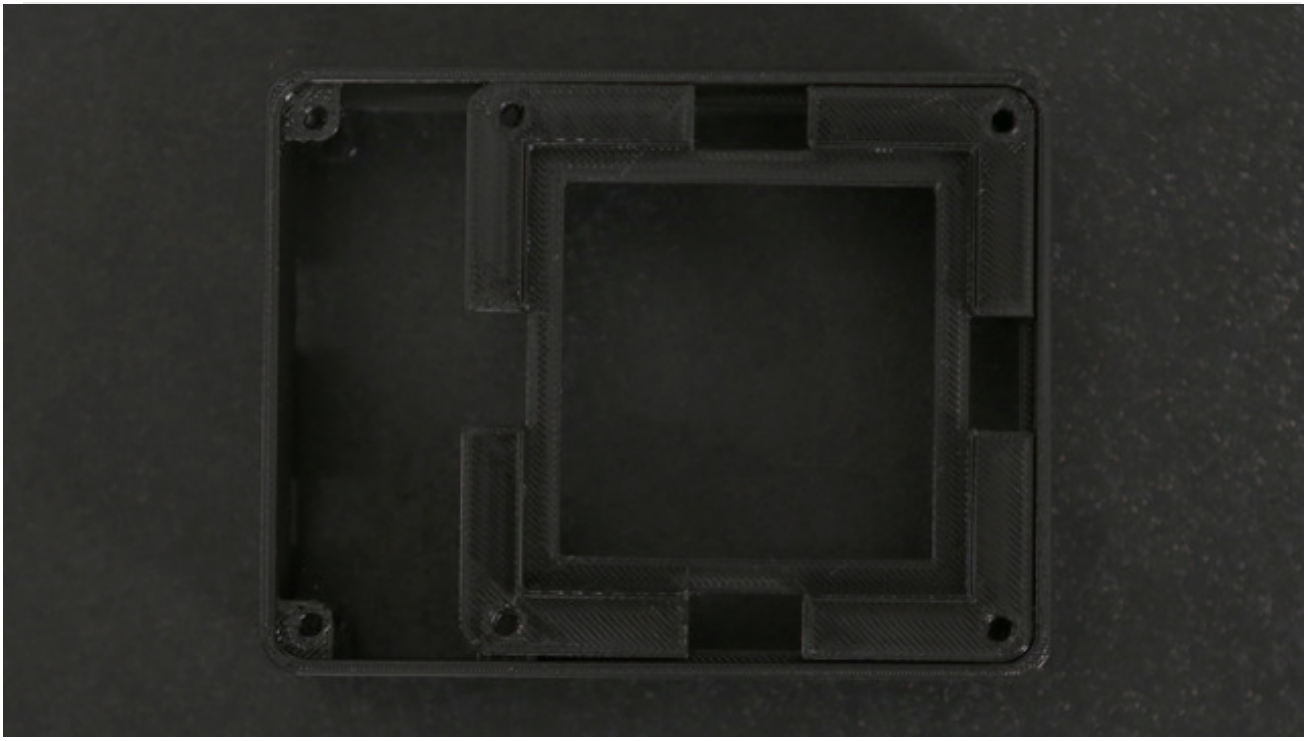
Copy and paste the full code below into a new sketch. The 16 buttons on the Trellis are mapped to trigger LEDs+MIDI notes. 4 potentiometers are mapped to MIDI CCs.

```
#include <Wire.h>
#include <Adafruit_Trellis.h>

#define LED     13 // Pin for heartbeat LED (shows code is working)
#define CHANNEL 1  // MIDI channel number

Adafruit_Trellis trellis;

uint8_t       heart       = 0;  // Heartbeat LED counter
unsigned long prevReadTime = 0L; // Keypad polling timer
uint8_t       mod;
uint8_t       vel;
uint8_t       fxc;
uint8_t       rate;

uint8_t note[] = {
  60, 61, 62, 63,
  56, 57, 58, 59,
  52, 53, 54, 55,
  48, 49, 50, 51
};

void setup() {
  pinMode(LED, OUTPUT);
  trellis.begin(0x70); // Pass I2C address
#ifdef __AVR__
  // Default Arduino I2C speed is 100 KHz, but the HT16K33 supports
  // 400 KHz.  We can force this for faster read & refresh, but may
  // break compatibility with other I2C devices...so be prepared to
  // comment this out, or save & restore value as needed.
  TWBR = 12;
#endif
  trellis.clear();
  trellis.writeDisplay();
  mod = map(analogRead(0), 0, 1023, 0, 127);
```

```
  vel = map(analogRead(1), 0, 1023, 0, 127);
  fxc = map(analogRead(2), 0, 1023, 0, 127);
  rate = map(analogRead(3),0, 1023, 0, 127);
  usbMIDI.sendControlChange(1, mod, CHANNEL);
  usbMIDI.sendControlChange(11, vel, CHANNEL);
  usbMIDI.sendControlChange(12, fxc, CHANNEL);
  usbMIDI.sendControlChange(13, rate, CHANNEL);
}

void loop() {
  unsigned long t = millis();
  if((t - prevReadTime) >= 20L) { // 20ms = min Trellis poll time
    if(trellis.readSwitches()) {  // Button state change?

      for(uint8_t i=0; i<16; i++) { // For each button...
        if(trellis.justPressed(i)) {
          usbMIDI.sendNoteOn(note[i], 127, CHANNEL);

          trellis.setLED(i);
        } else if(trellis.justReleased(i)) {
          usbMIDI.sendNoteOff(note[i], 0, CHANNEL);
          trellis.clrLED(i);
        }
      }
      trellis.writeDisplay();
    }
    uint8_t newModulation = map(analogRead(0), 0, 1023, 0, 127);
    if(mod != newModulation) {
      mod = newModulation;
      usbMIDI.sendControlChange(1, mod, CHANNEL);
    }
    uint8_t newVelocity = map(analogRead(1), 0, 1023, 0, 127);
    if(vel != newVelocity) {
      vel = newVelocity;
      usbMIDI.sendControlChange(11, vel, CHANNEL);
    }
    uint8_t newEffect = map(analogRead(2), 0, 1023, 0, 127);
    if(fxc != newEffect) {
      fxc = newEffect;
      usbMIDI.sendControlChange(12, fxc, CHANNEL);
    }
    uint8_t newRate = map(analogRead(3), 0, 1023, 0, 127);
    if(rate !=newRate) {
      rate = newRate;
      usbMIDI.sendControlChange(13, rate, CHANNEL);
    }
```

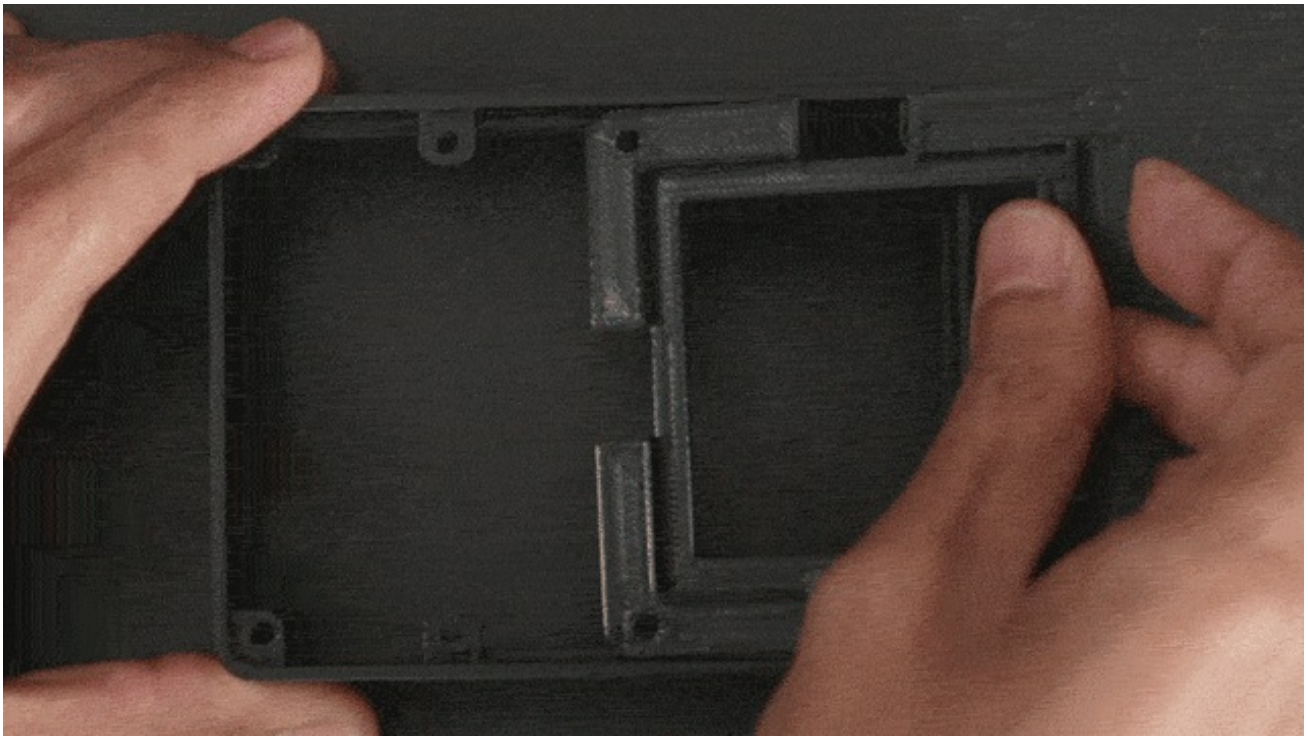
```
    prevReadTime = t;
    digitalWrite(LED, ++heart & 32); // Blink = alive
  }
  while(usbMIDI.read()); // Discard incoming MIDI messages
}
```

Assembly



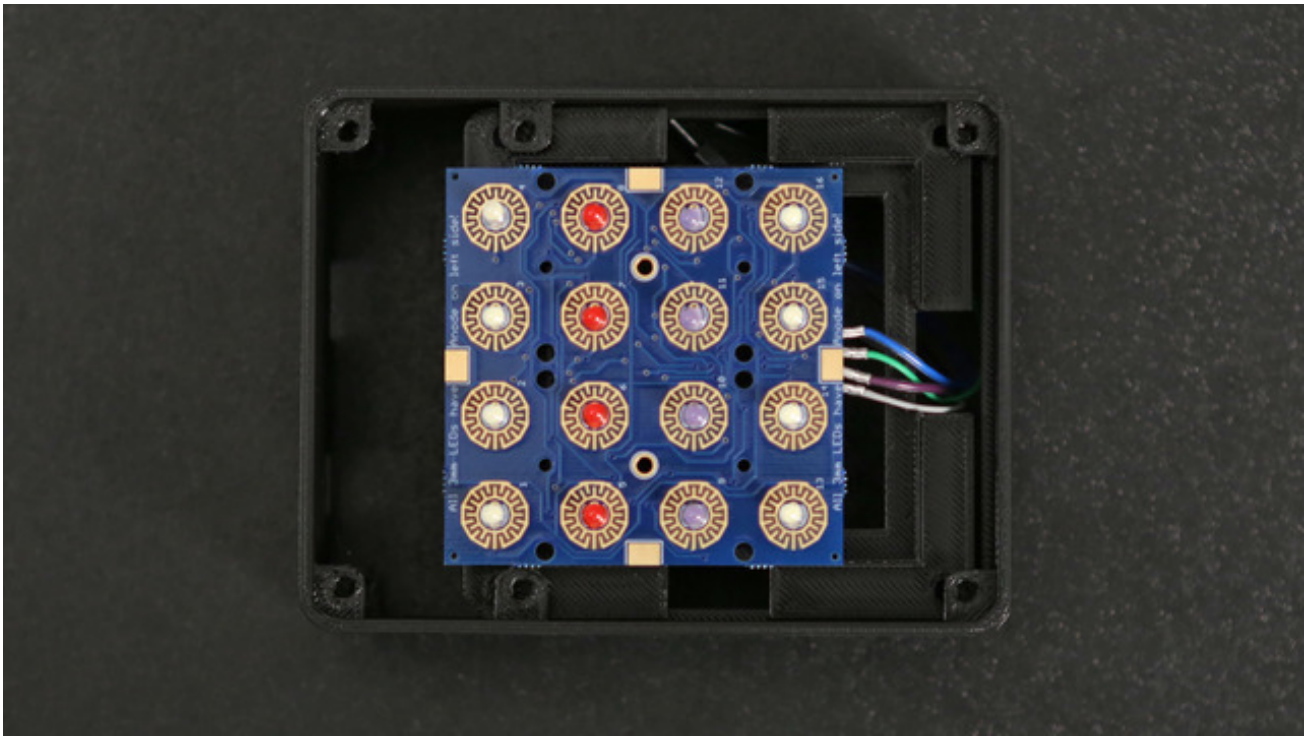
Add Tray to Enclosure frame

Position the tray over the frame and orient it so that the mounting holes are lined up precisely. You'll notice two sets of holes are positioned slightly different.



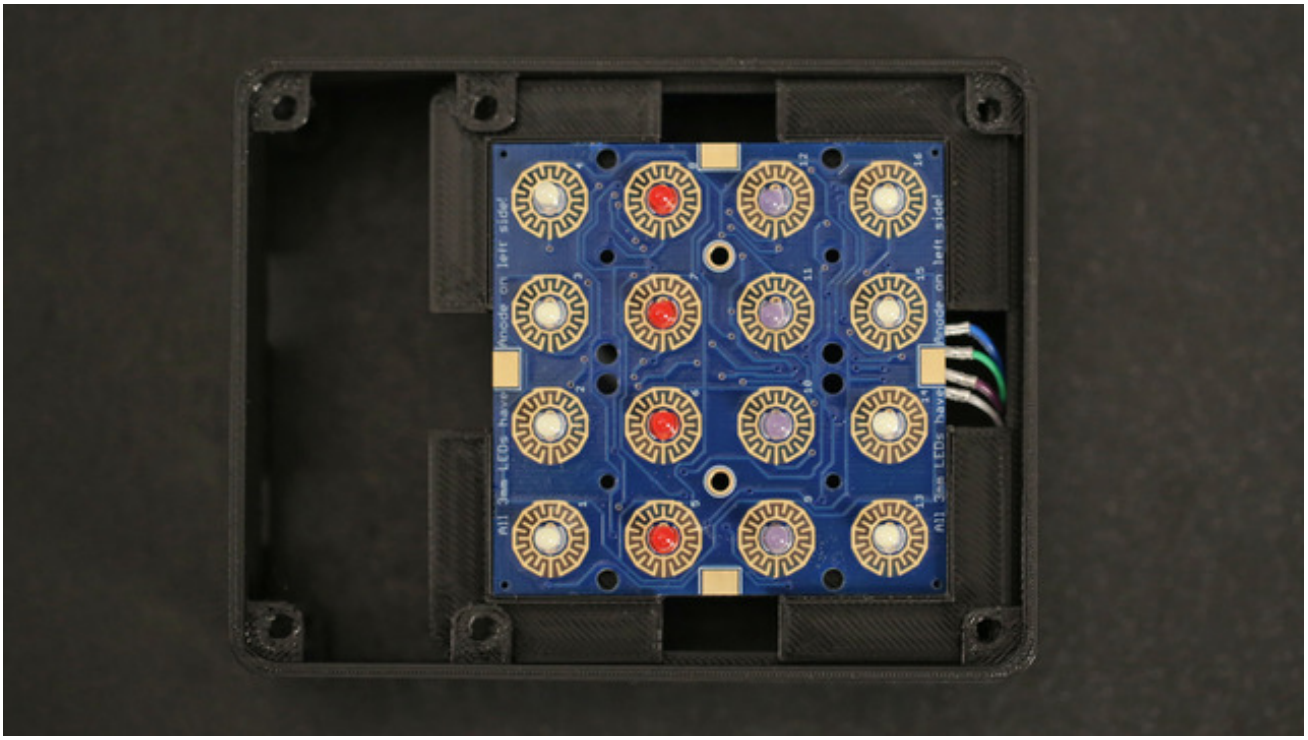
Install Tray to Enclosure frame

Insert the **mini-oontz-tray.stl** part into the mini-oontz-frame.stl with one end sliding in first. Make sure to line up the holes.



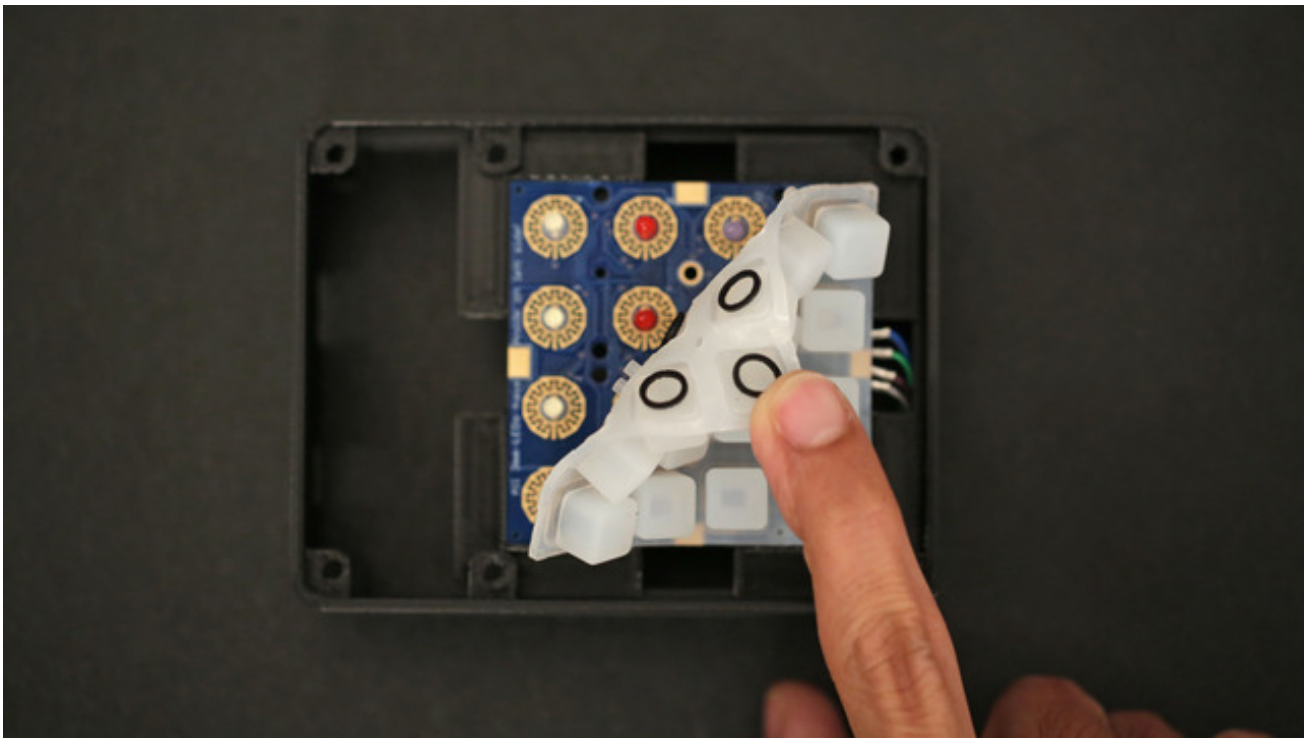
Install Trellis PCB to Tray

Position and rotate the Trellis so that the numbers are upright with the opening for the potentiometers. Group the jumper cables from the trellis and pull them through one of the opening of the tray.



Secure Trellis to Tray

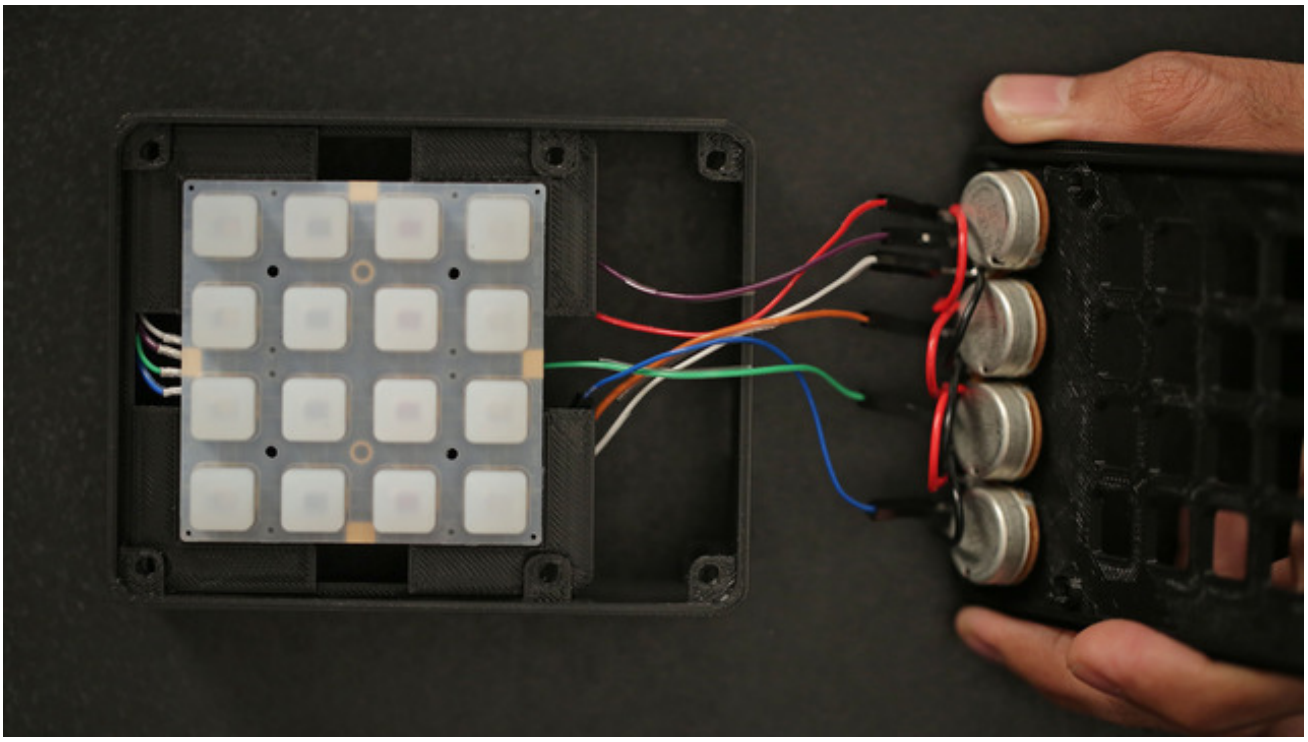
Place the Trellis down into the tray and snap it into place.



Install Elastomers to Trellis PCB

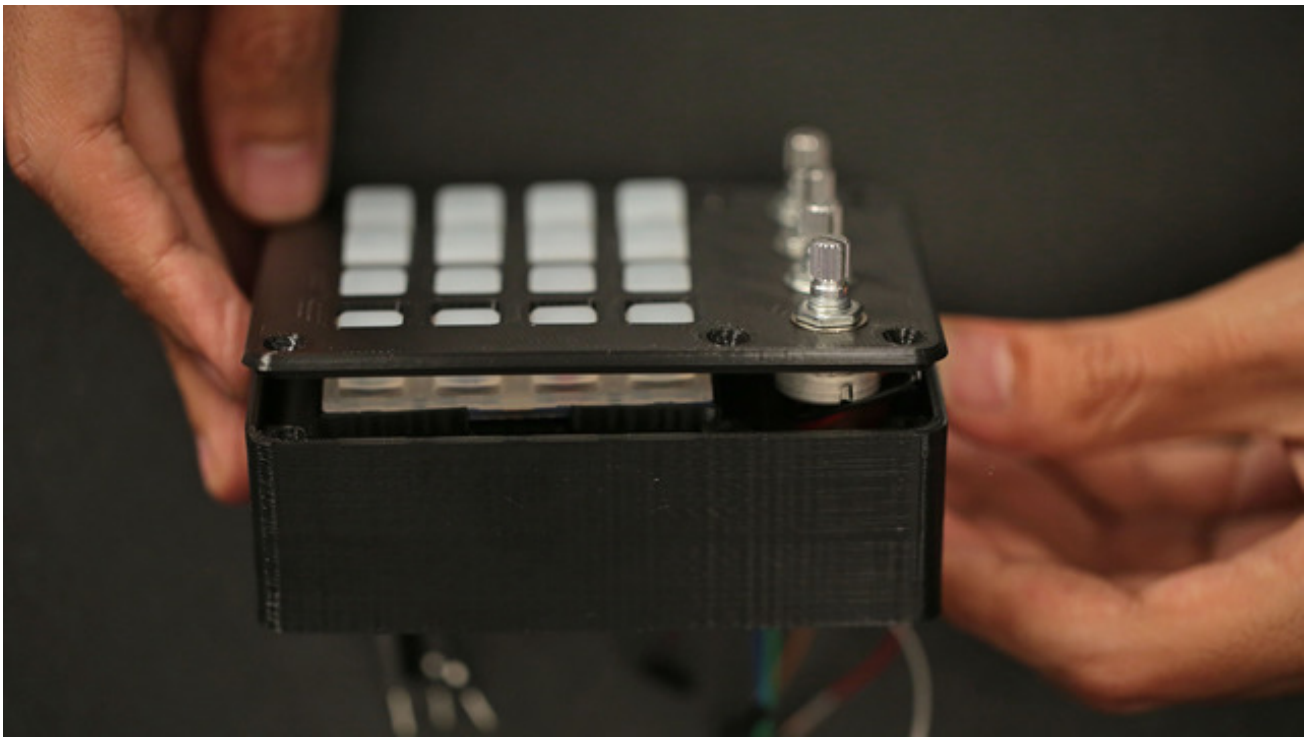
Grab the 4x4 silicone elastomers and position them over the Trellis PCB. Orient the keypad

so that the "nubs" are inserting into the holes on the Trellis.



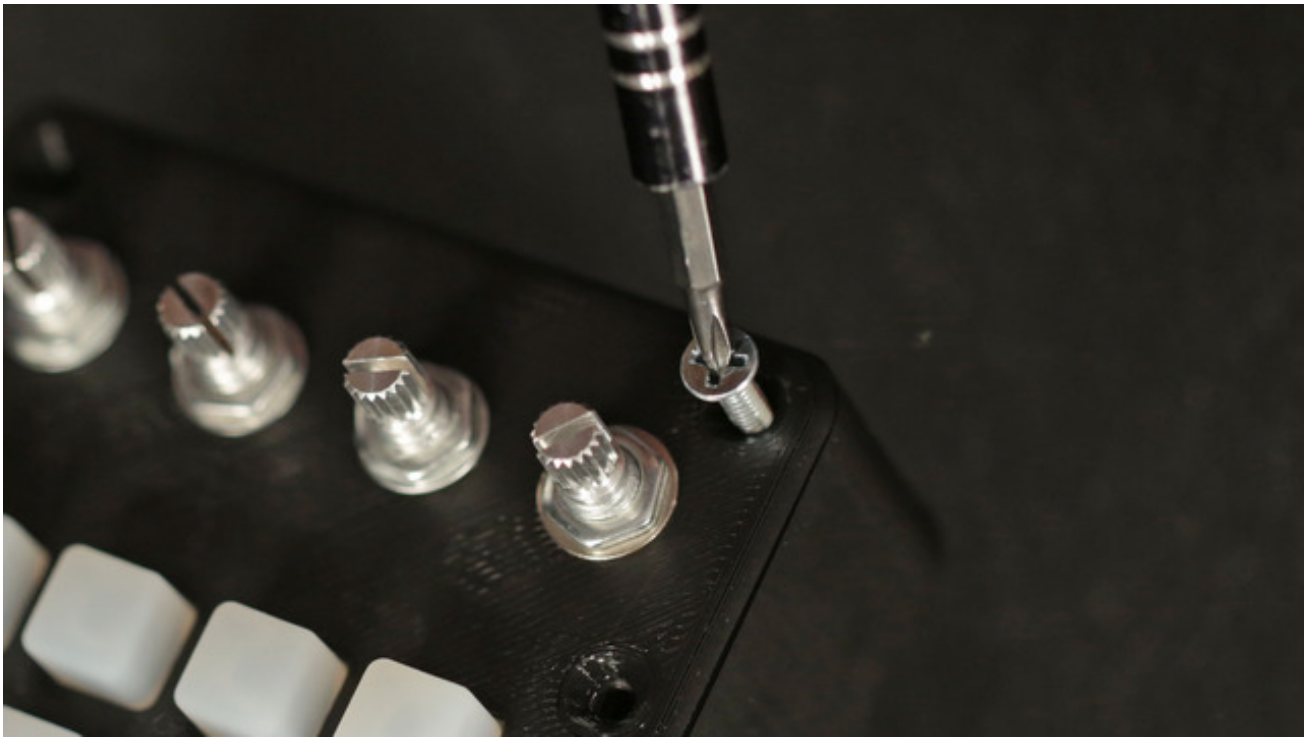
Insert Pot Wires into Enclosure

Group the jumper wires from the potentiometers and fit them through the top large opening of the enclosure. This gap is appropriately sized to fit the 4 pots.



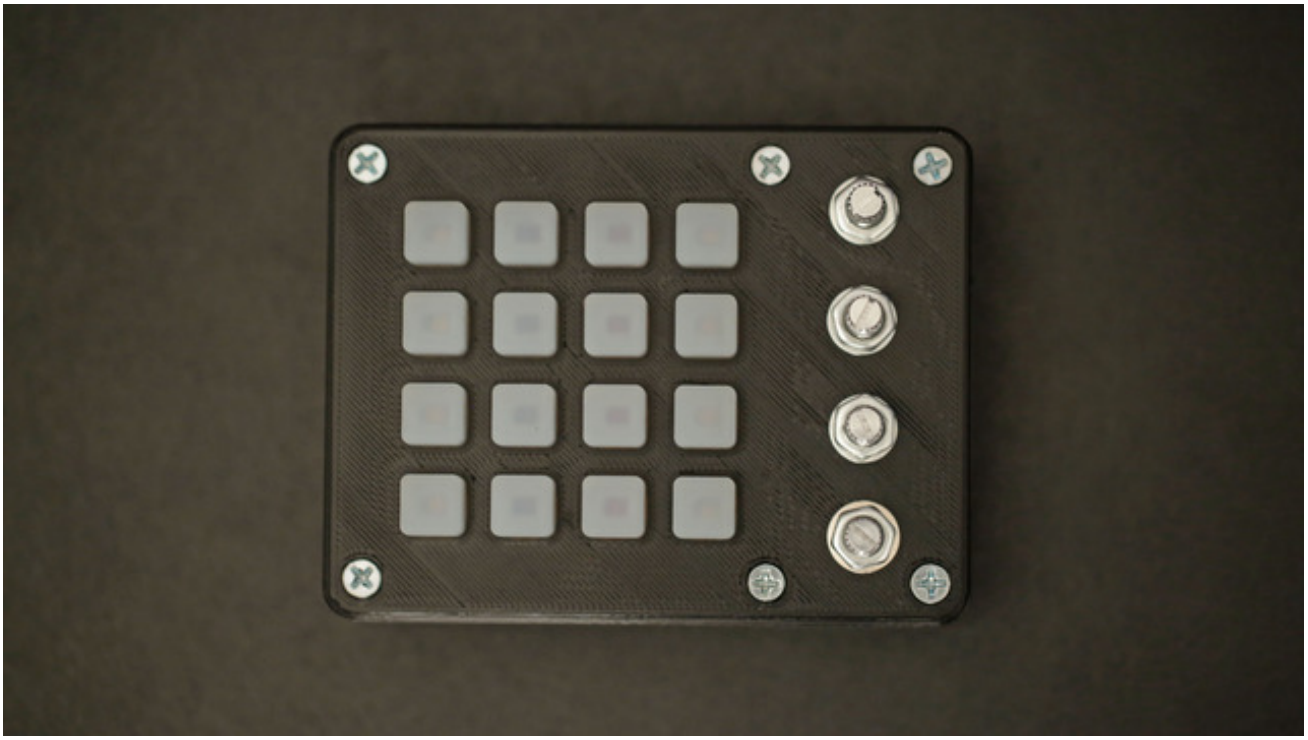
Add Cover to Enclosure Frame

Position the mini-oontz-cover.stl over the mini-oontz-frame.stl with the mount holes lined up.



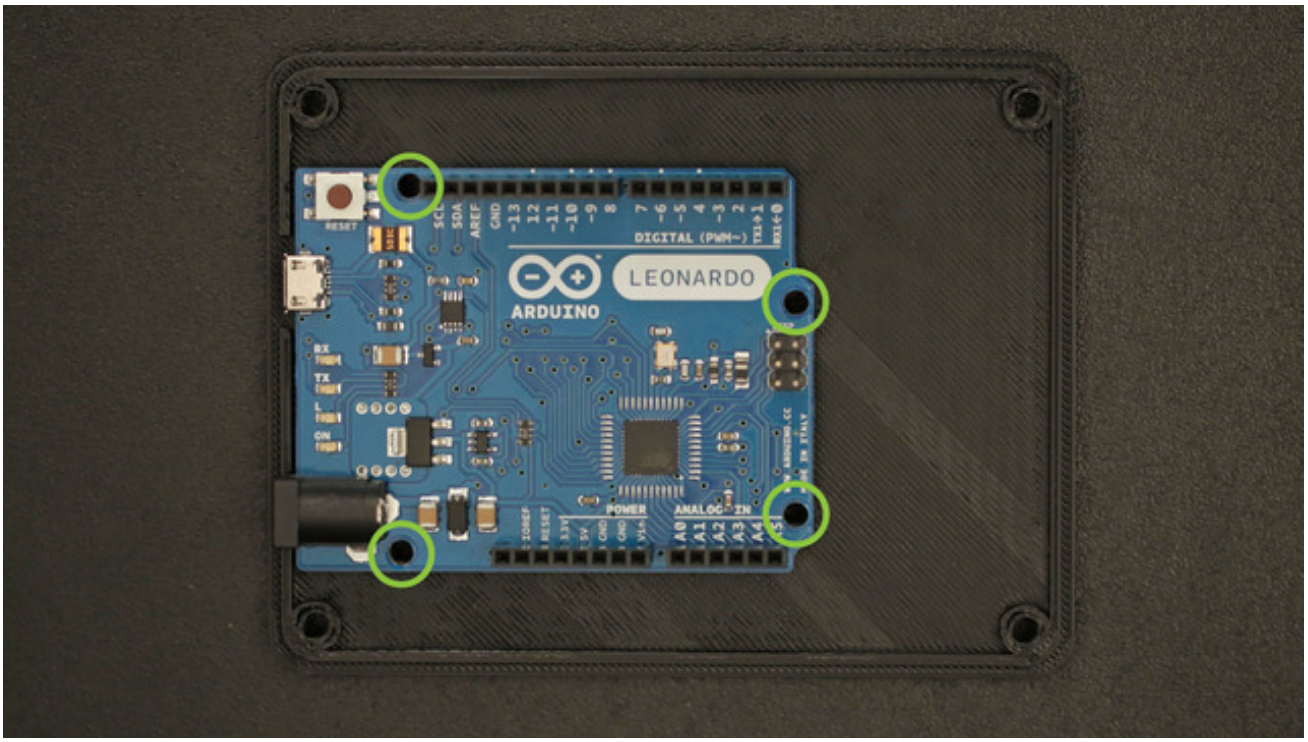
Secure Cover to Enclosure Frame

Place the cover down onto the frame and snap it shut. Use 6 #6-32 1/2' flat phillips screws to secure the cover to the frame.



Secured Enclosure and Cover

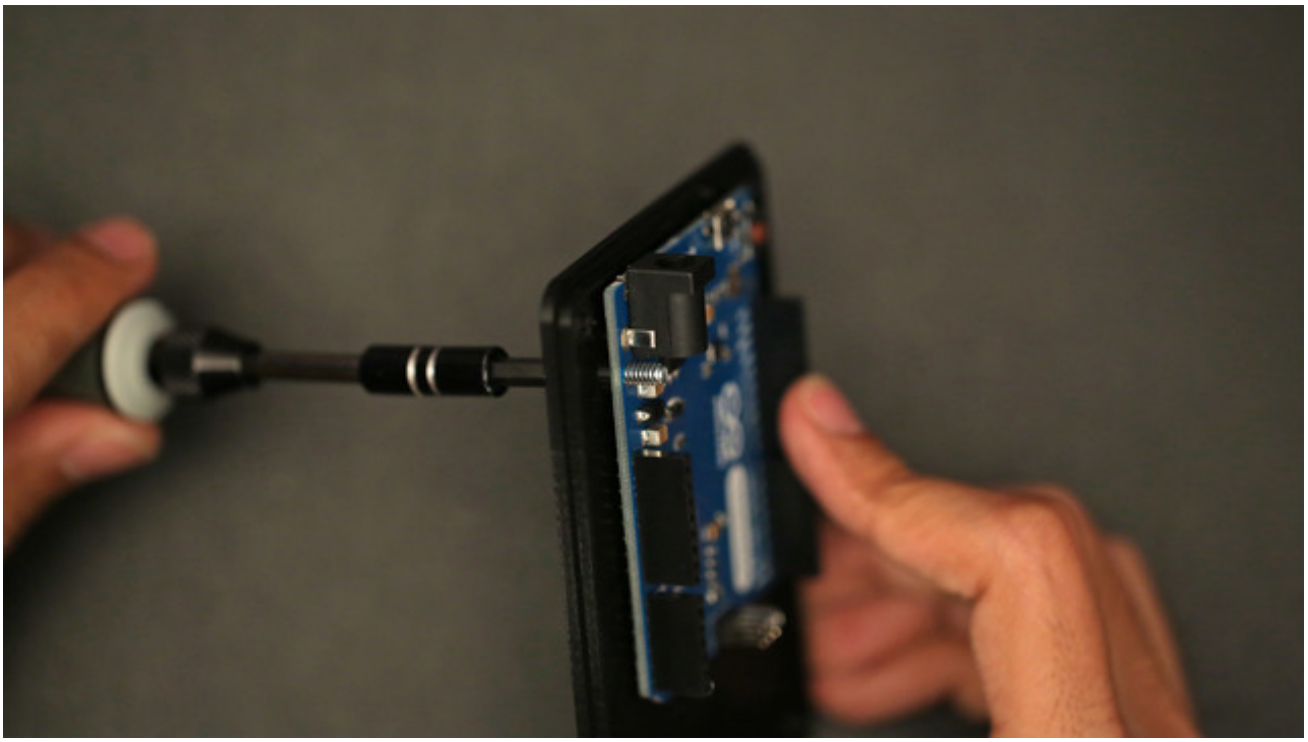
Yey! The 6 screws keep the Trellis PCB and 4 potentiometers secured inside the enclosure.



Add Leonardo to Bottom

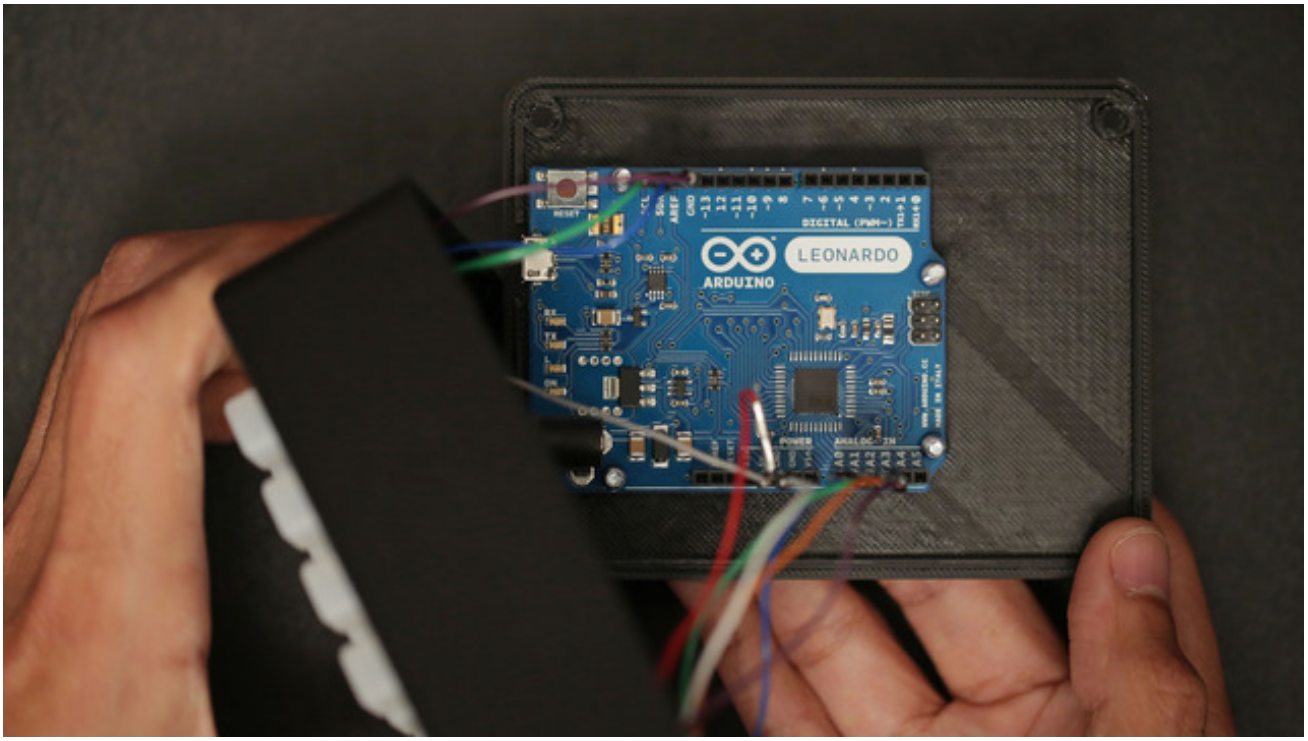
Place the Leonardo over the **mini-oontz-bottom.stl** part and make sure the mount holes

line up.



Mount Arduino Leonardo to Bottom

Use 4 #6-32 x 1/2' to secure the Leonardo board to the bottom cover.

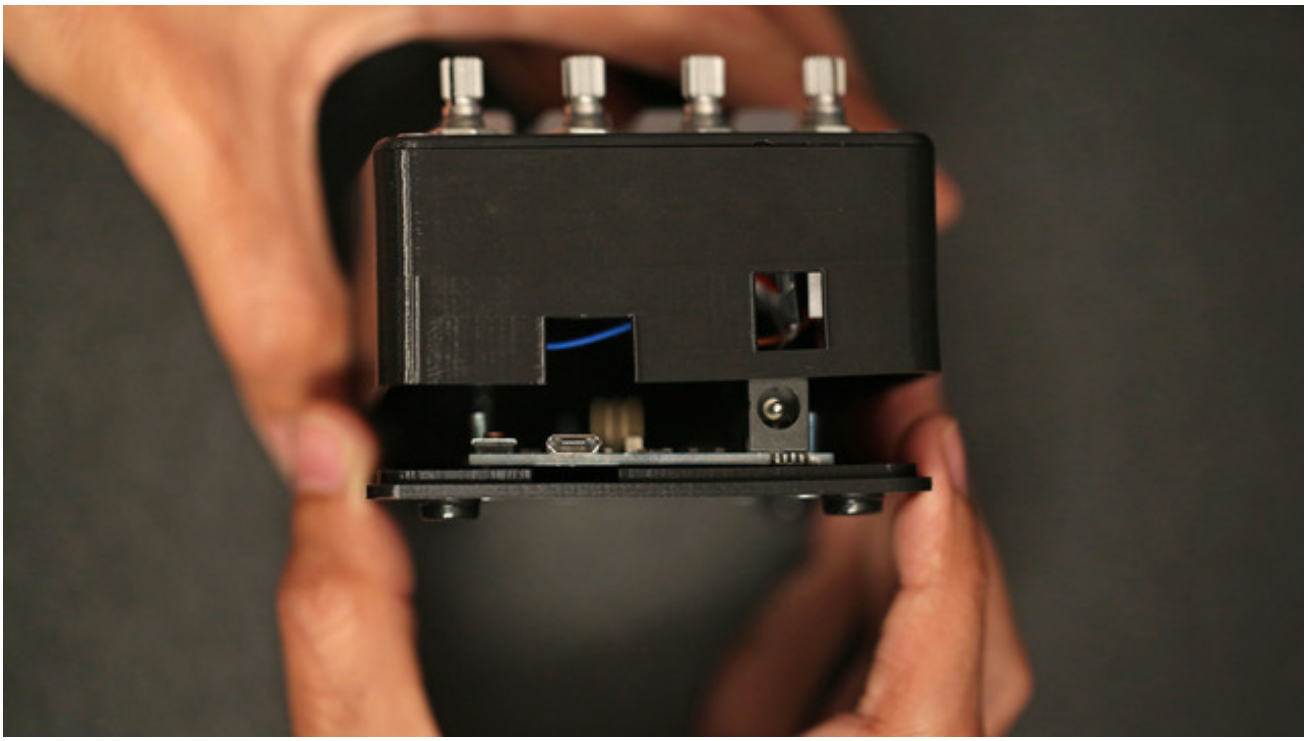


Connect Trellis+Pot jumper wires to Arduino Leonardo

Position the frame with the mounted cover towards the Arduino board. Make sure the openings are lines up to allow for the power jack and USB micro port. Connect the male jumper wires to the appropriate pins on the header of the Leonardo.

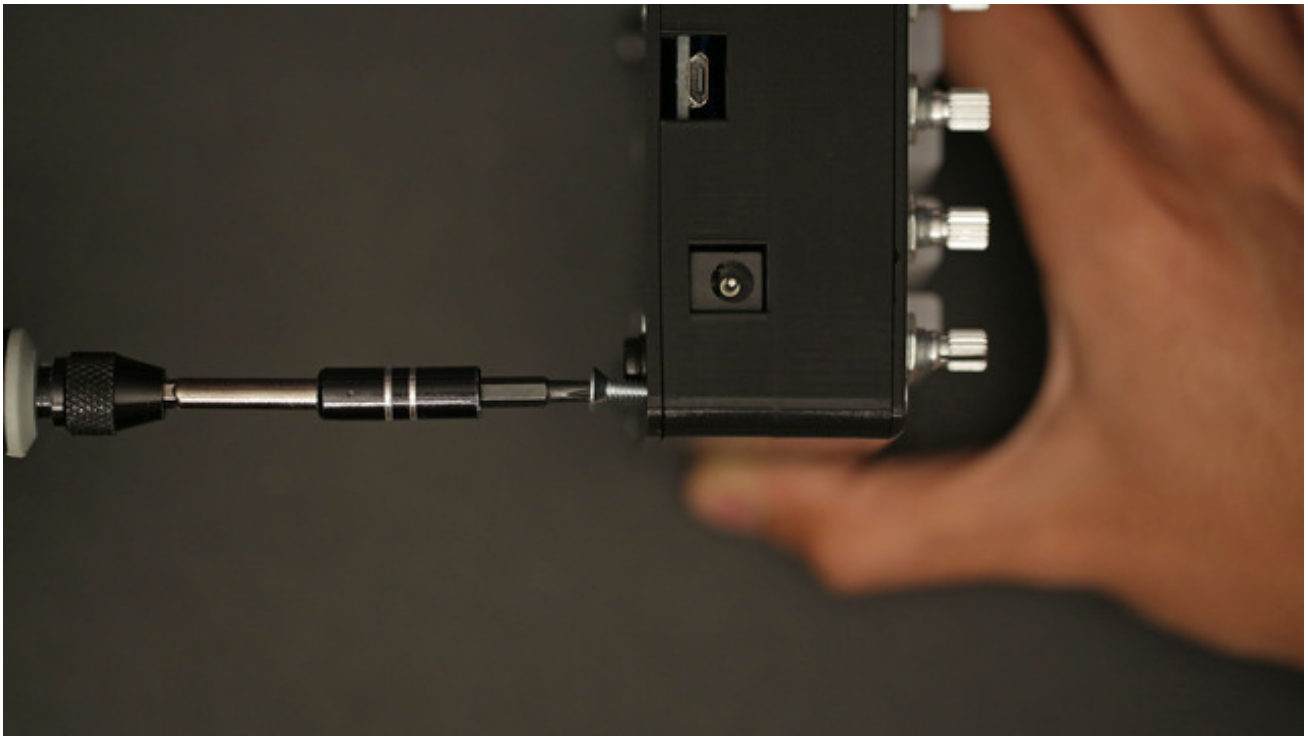
- SCL to SLC
- SDA to SDA
- GND to GND
- 5V to 5V

Connect the **GND** and **5V** jumper cables from the master potentiometer to the a spare **GND** on the Leonardo. The **5V** jumper cable will need to share a spot with **5V** Trellis jumper cable. You can easily remove the plastic end of the 5V jumper cable of the master pot, bend the terminal and insert it on top of the jumper of the Trellis.



Line up Bottom with Enclosure

Ensure the orientation of the Leonardo is lined up with the enclosure so that the USB port has an opening.

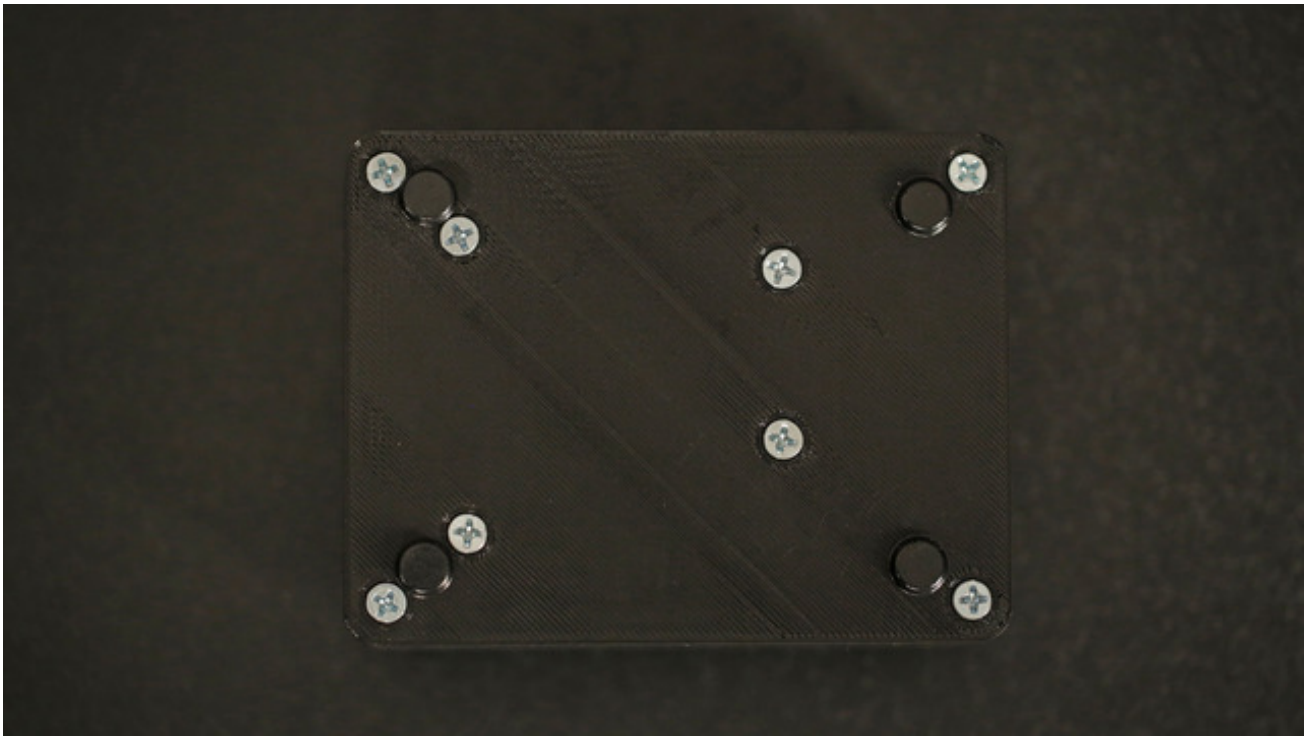


Secure Bottom to Enclosure Frame

With all the jumper cables connected, snap the enclosure to the frame. Use 4 #6-32 1/2' flat phillips screws to secure the two pieces together.

Potentiometer Knobs

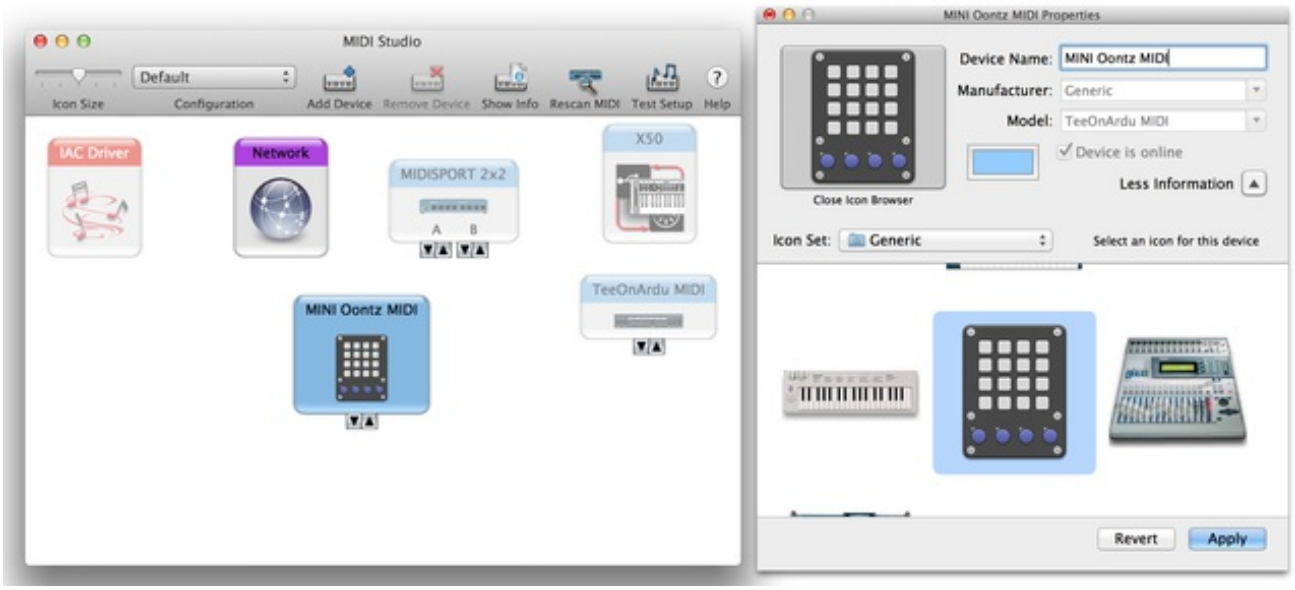
You can 3D print your own knobs or get some neat ones on the internets. I picked up 1/2' pot knobs from RadioShack (http://adafru.it/dLF).



Little Rubber Bumper Feet

Keep your mini OONTZ from going barefoot, give them little rubber feet (http://adafru.it/dLG)! These small sticky bumpers are our favorite accessory for any electronic kit or device. They are sticky, but not impossible to remove.

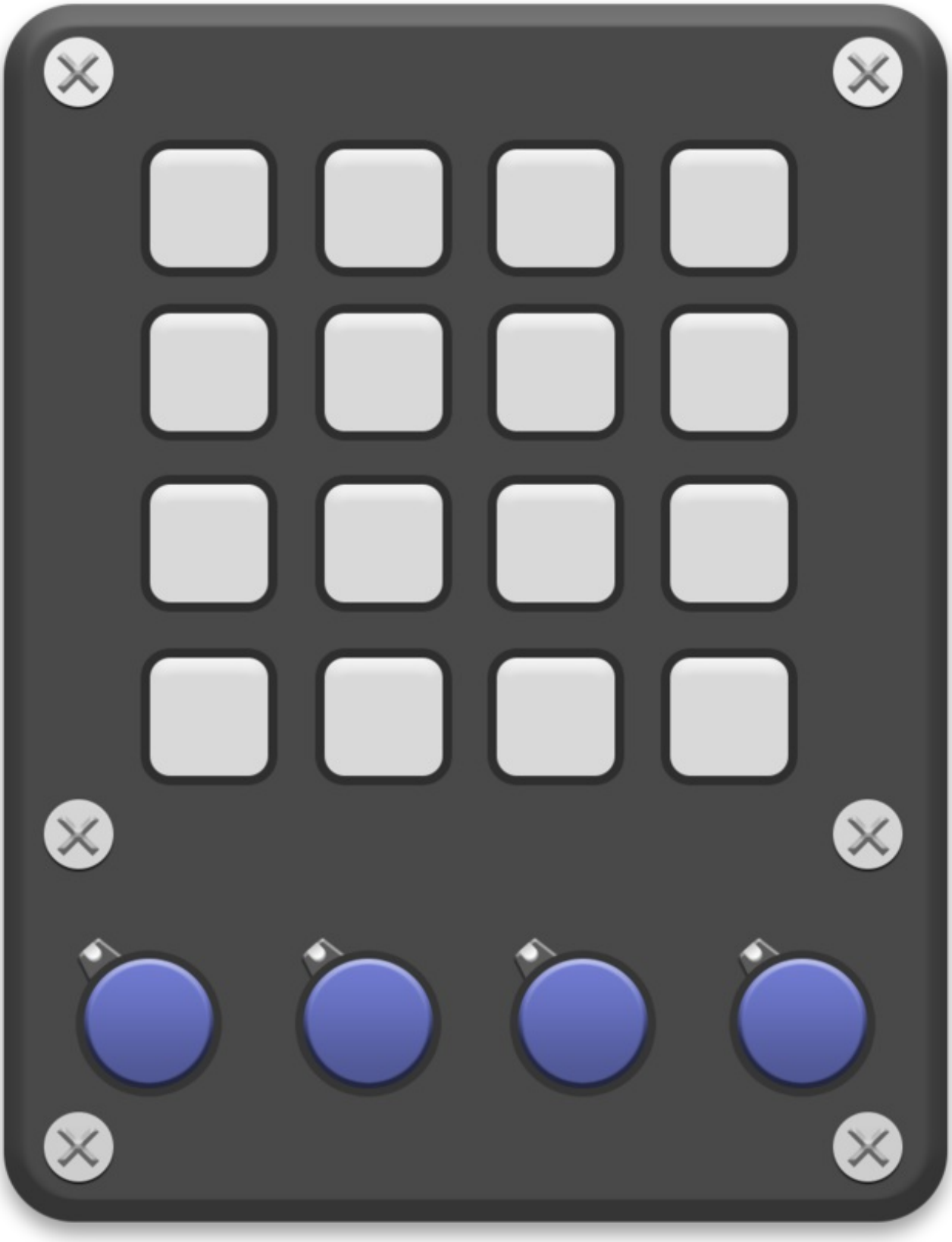
Configure MIDI



MIDI OSX Setup

To configure your OONTZ, you can launch the **Audio MIDI Setup** app by searching for it in Finder. It usually likes to reside in the **~/HD/Applications/Utilities** directory. Once launched, go to the top menu and select Window > Show MIDI Window.

You should see the TeeOnArdu MIDI object listed in the MIDI studio window. Double click it to see MIDI Ports and rename, change icon/color, etc if you'd like.



Mini OONTZ Device Icon

You can add a custom mini OONTZ icon to your software stuffs. To add the icon to the OSX

Midi window, simply drop the PNG icon to the following directory. You'll need to enter your password to set the proper permission. **~/HD/Library/Audio/MIDI Devices/Generic/Images**

Using with iOS Devices

The USB Camera Kit adapter (http://adafru.it/dLH) allows you to use the OONTZ with any MIDI enabled iOS App. Just plug it in, a wait second and discard the dialog pop up. You may need to enable omni mode in your iOS app in order for OONTZ to control MIDI.

Controlling Analog MIDI Synths

Analog MIDI synths that use MIDI cables can be controlled with the OONTZ using a MIDI to USB interface. Using an old school MIDI cable, you can connect the midi input of your analog synth to the output of the Midi to USB interface. Once configured and connect, OONTZ will act like any other USB classic MIDI device.