CompuCanvas 101 - Transitions

Created by Chris Daly

https://learn.adafruit.com/compucanvas-101

Last updated on 2024-03-08 03:33:27 PM EST

Table of Contents

Overview 3

Parts and Tools 9

- Optional Lighting Parts
- Cables
- Tools
- Art and Hardware Store Parts

Canvas Enclosure Parts 13

Preparing the Canvas 15

Computers on Canvas 22

Digressions in Code 44

- Preparing a microSD card with Raspbian
- Logging into a new CompuCanvas
- Initial system configuration

Internal Lighting 58

Alternate Internal Lighting 77

External Features 90

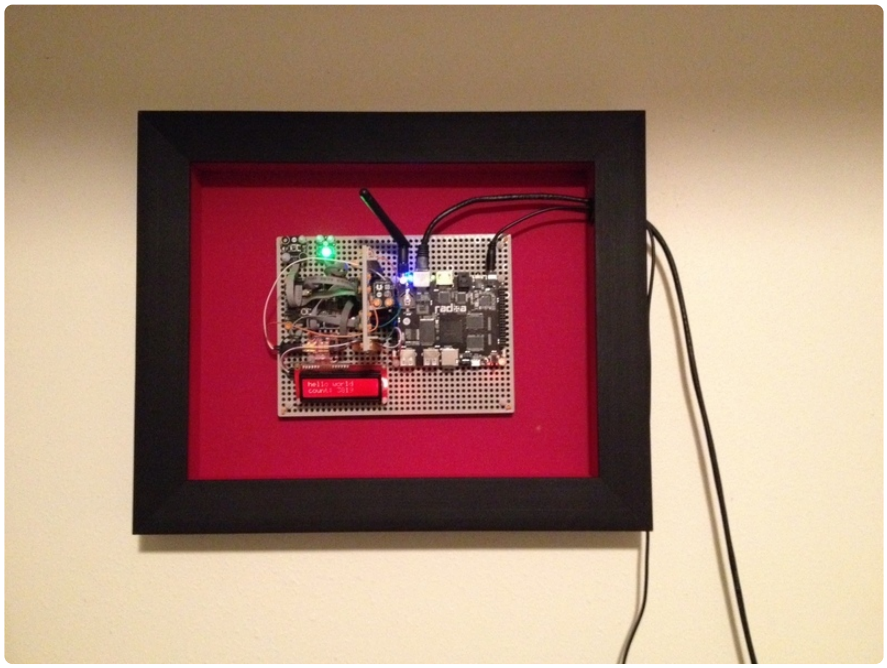
More External Features 106

Audio Addendum 118

Overview

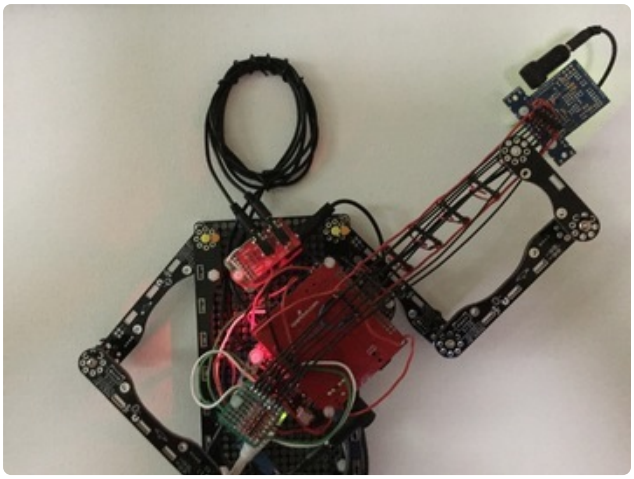
To make a long story short, several years ago I ran out of space for electronics projects on my desk, and I looked to the wall as a new frontier.

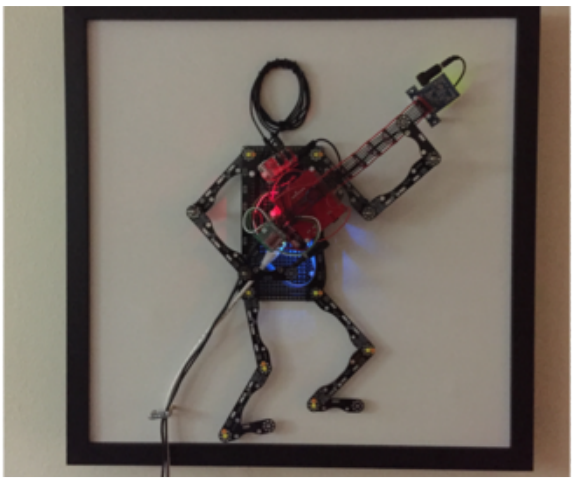
In previous decades, I had spent a lot of time building computer systems in large metal "tower" cases. With the advent of hacker board computers like Arduino, BeagleBone, Raspberry Pi and others it has become possible to assemble computer systems that are so light, they hang on the wall. So that's what I did.

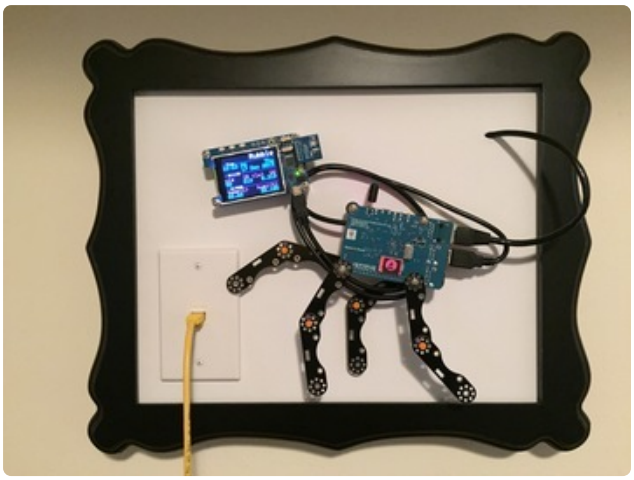


My first few attempts were pretty clunky, as shown above. I would first assemble a functional computer system on a plastic base with a grid of mounting holes. Then, when I decided the system was basically working I would fit the plastic base into a wall-hanging frame.

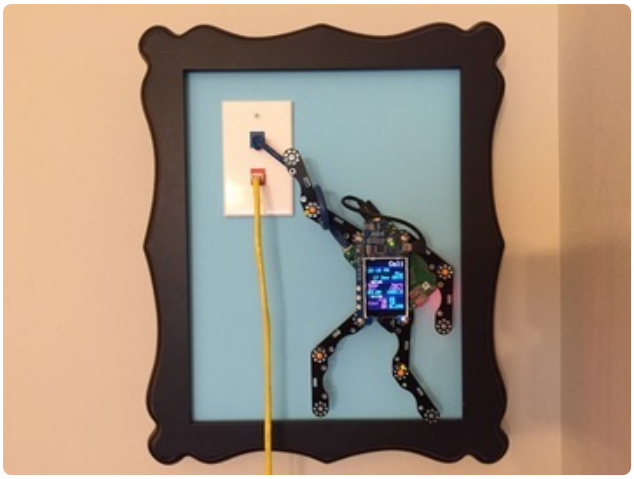
Later I started to think about designing characters and scenes on the canvas. I had sometimes seen collage-style wall-hanging art incorporating non-functional parts, like chips, wires, connectors, etc. But I had never seen wall-hanging art that used functional computer systems, so I decided to delve into that niche.

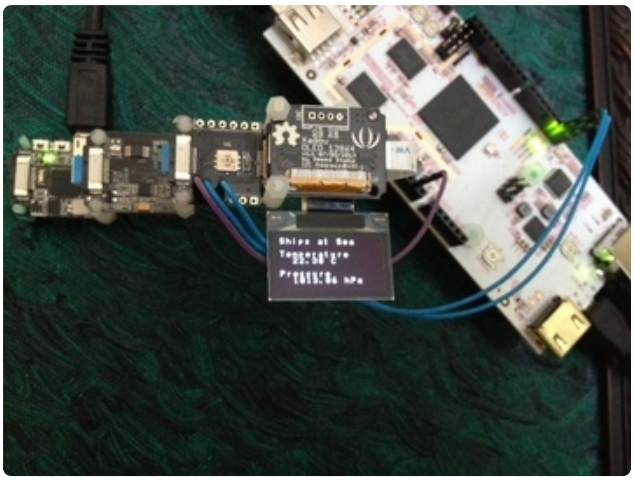






You can find more information about the projects pictured here in my GitHub repos, like GuitarThing (https://adafru.it/HfP), ShipsAtSea (https://adafru.it/HfQ) and fold (https://adafru.it/HfR).





At one point I thought ... what about a Kickstarter? Can this thing be scaled up? Will big manufacturers like Lenovo or Apple wake up and realize the wall is the new frontier for computing? (Nest is one answer to that.)

I ruled out a crowd-funded project. Overall, I don't really want to be involved in a manufacturing operation. What I enjoy most is designing each new system in some way different than what I've done before.

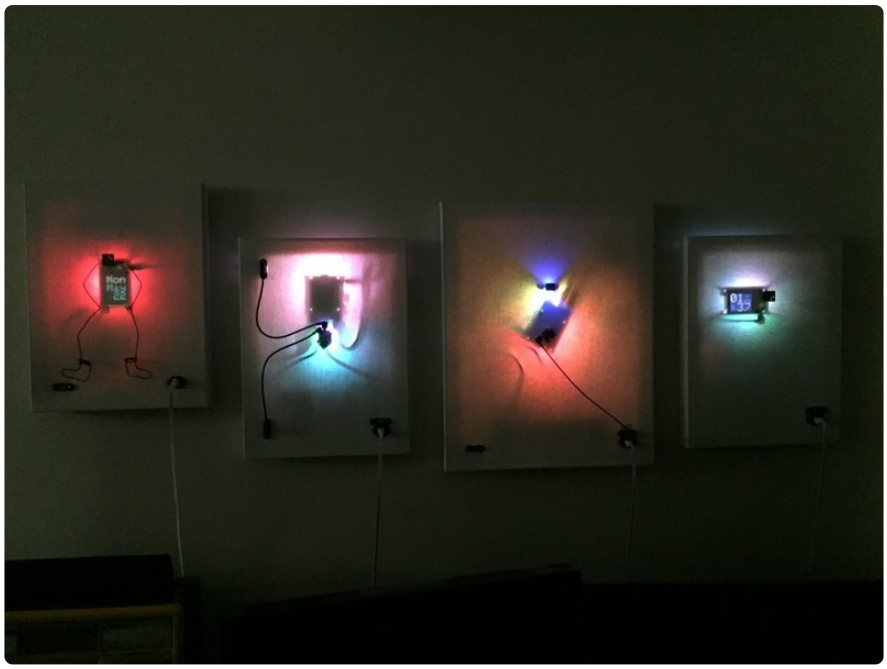
The process of thinking about scaling up the operation also caused me to reconsider the basic concept of computers on the wall and try to distill it to its most fundamental

essence. For example, I wondered what is the minimal viable wall-hanging canvas enclosure computer system that could be easily assembled and shipped in quantity? I named my answer to that question the CompuCanvas model A0 (https://adafru.it/HfS).

Here are some of the design ideas and lessons learned that guided the 'CompuCanvas' concept:

- Only one cable should extend from the canvas to the floor. Make it be Power over Ethernet if possible.
- Use a Raspberry Pi as the primary computer, with USB peripheral devices (like Arduinos or CircuitPython devices).
- Place a USB speaker behind the canvas so the system has audio output.
- Use panel mount cables to transition from one side of the canvas to the other.
- Don't worry about fancy framing around the sides of the canvas. Just hang the bare canvas on the wall.
- The canvas can be a proper 'enclosure' with all the parts inside, or it could be what I call an 'exclosure' where parts are placed on the canvas front. Exclosures will be harder to package for shipping and more susceptible to damaging drops and face-plants, but still worth doing for all the design possibilities they enable.

Even with constraints like outlined above, there are a multitude of possibilities. And these rules may be bent or broken as necessary to achieve even more.



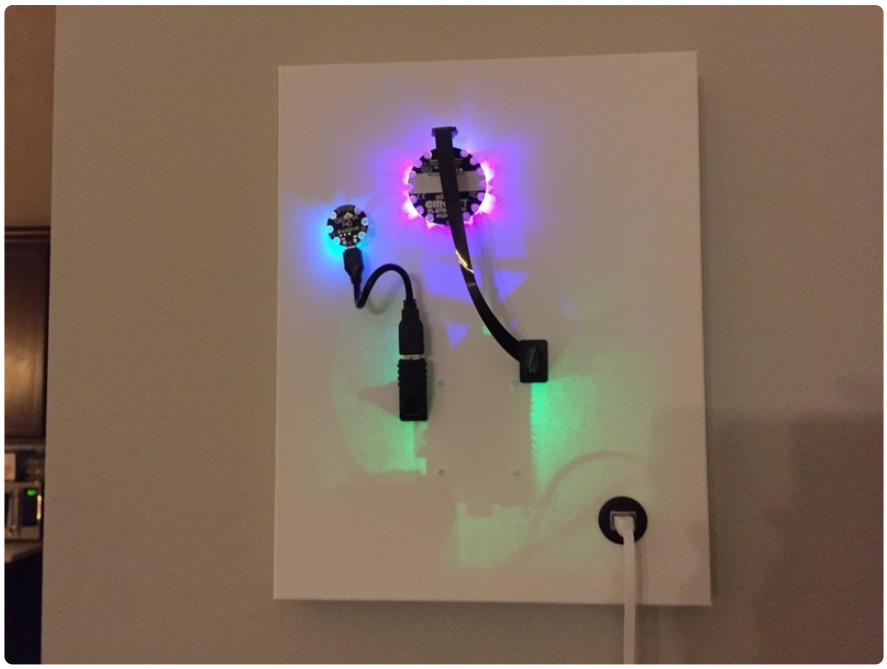
Over time I've experimented with a lot of different parts and arrangements, but I still have barely scratched the surface of ideas.

For this guide, I'll walk you through the process of constructing a CompuCanvas type system. I will present a primary path you can follow as well as some alternate routes you may explore.

We'll start by creating a plain enclosure style CompuCanvas housing a Raspberry Pi 3B and a speaker, as shown below.

After that I will look at several variations on placing lights inside the canvas. One example of this is shown below.

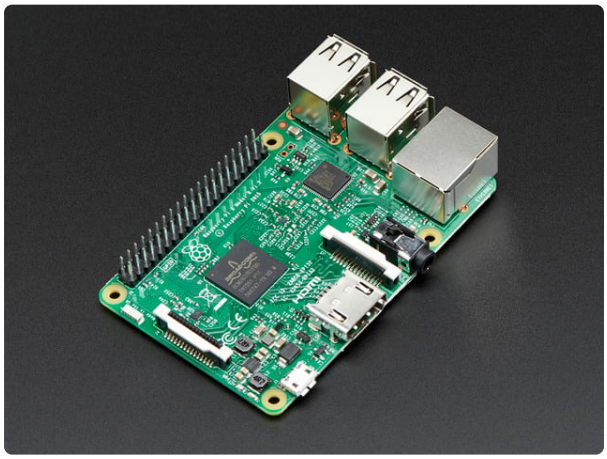
Finally, I'll show how to make USB transitions to the front of the canvas to add external features with illumination.



Parts and Tools

This page is a master list of the parts and tools used in the guide. Some pages of the guide present optional paths that you may choose to skip, so you may not need every part shown below.

I'll start with the most essential parts from Adafruit. The CompuCanvas we build will house a Raspberry Pi model 3B with a USB speaker and a Power over Ethernet module. It will also use USB connected CircuitPython devices to create a light show on the canvas. These key parts are listed just below.



Raspberry Pi 3 - Model B - ARMv8 with 1G RAM

Did you really think the Raspberry Pi would stop getting better? At this point, we sound like a broken record, extolling on the new Pi's myriad improvements like we're...

https://www.adafruit.com/product/3055

PoE Splitter with MicroUSB Plug - Isolated 12W - 5V 2.4 Amp

Power over Ethernet (PoE) splitters can make your project wiring a breeze - and best of all they work to connect any 802.3af-capable router to any Ethernet device, even if it...

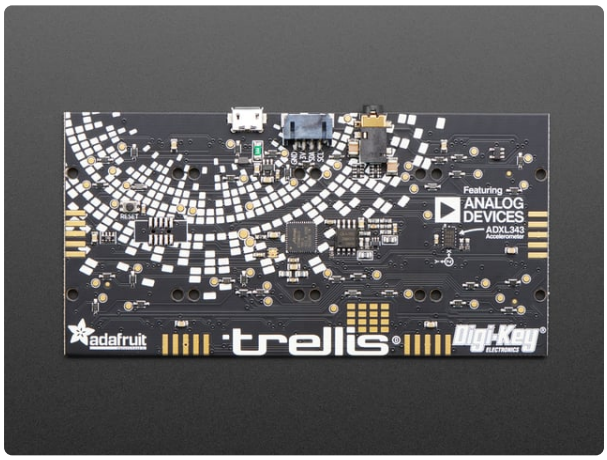
https://www.adafruit.com/product/3785

Note: for the other end of the Power over Ethernet splitter shown above, you will need an IEEE 802.3af compliant PoE 'injector', which takes power and Ethernet as input and will send them over an Ethernet cable to the PoE splitter shown above. The PoE injector part will be shown in more detail in upcoming sections of this guide.

Mini External USB Stereo Speaker

Add some crisp and compact audio to your project with this Mini External USB 2.0 Speaker. It's USB-only and does not require or use a 3.5mm audio jack....

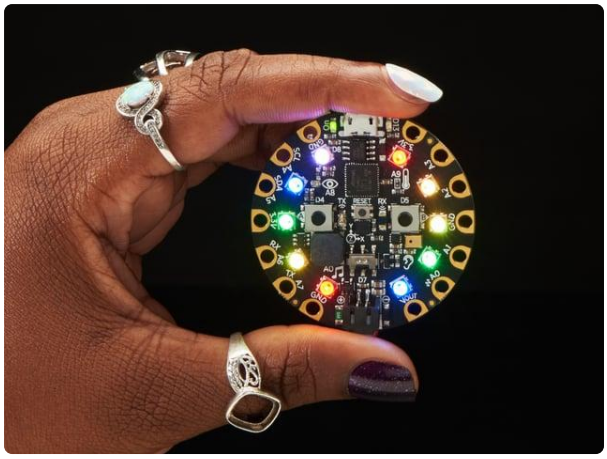
https://www.adafruit.com/product/3369



Adafruit NeoTrellis M4 Mainboard - featuring SAMD51

We got a big ol' blender and tossed in an ItsyBitsy M4, two NeoTrellis boards and an electret mic amp - turned on the 'mix' button and out came the NeoTrellis M4 - a super...

https://www.adafruit.com/product/3938



Circuit Playground Express
Circuit Playground Express is the next step towards a perfect introduction to electronics and programming. We've taken the original Circuit Playground Classic and...
https://www.adafruit.com/product/3333

Below are a number of additional parts from Adafruit used in this guide. Some of these are optional, as indicated in the description text.

1 x White Nylon Screw and Stand-off Set – M2.5 Thread — https://www.adafruit.com/product/3658
These nylon screw and standoff parts are essential for mounting computers on canvas.

---

1 x SD/MicroSD Memory Card — https://www.adafruit.com/product/2693
microSD card for Raspberry Pi

Optional Lighting Parts

1 x Adafruit NeoPixel LED Strip w/ Alligator Clips - 30 LEDs/meter - 1 Meter — https://www.adafruit.com/product/3812
30 NeoPixel strip to add light and color behind the canvas

---

1 x Adafruit GEMMA M0 — https://www.adafruit.com/product/3501
Gemmo M0, used to control NeoPixel strip or for external canvas lighting

---

Cables

I have indicated a quantity of 2 for some of these cables below, but you may only need one depending on which of the interior and exterior lighting options you choose.

Panel mount cable to bring Ethernet (and power) into the canvas.

1 x RJ-45 Ethernet Round Panel Mount Extension Cable - 30cm — https://www.adafruit.com/product/4215

Panel mount cable to bring Ethernet (and power) into the canvas.

---

2 x Ethernet Cable — https://www.adafruit.com/product/730

Ethernet cables for canvas exterior. Also consider the flat, thin style Ethernet cables shown in an upcoming page of this guide.

---

2 x USB cable - 6" A/MicroB — https://www.adafruit.com/product/898

Short USB cable for connecting CircuitPython devices to rPi.

---

2 x USB cable - USB A to Micro-B - 3 foot long — https://www.adafruit.com/product/592

Long USB cable for connecting CircuitPython devices to rPi

---

2 x Snap-In Panel Mount Cable - USB A Extension Cable — https://www.adafruit.com/product/4055

USB cable to transition from behind to front of canvas, for externally placed CircuitPython devices.

---

1 x DIY USB Cable Parts - Straight Type A Plug — https://www.adafruit.com/product/4109

USB A end of ribbon cable assembly

---

1 x DIY USB or HDMI Cable Parts - 20 cm Ribbon Cable — https://www.adafruit.com/product/3561

20 cm ribbon cable

---

1 x DIY USB Cable Parts - Right Angle Micro B Plug Up — https://www.adafruit.com/product/4104

micro USB end of ribbon cable assembly

Tools

This guide uses some common tools that you can find at a hardware store, or maybe already have:

- hammer
- Philips screwdriver

- wire cutter
- awl and/or large needles
- mechanical pencil

And here are a few additional tools used in the guide that can be found in the Adafruit store:

1 x USB MicroSD Card Reader/Writer — https://www.adafruit.com/product/939
USB adapter for programming microSD cards

---

1 x Adafruit PyRuler — https://www.adafruit.com/product/4319
ruler for measuring and marking cut points on canvas

---

1 x Slice Craft Knife with Ceramic Blade — https://www.adafruit.com/product/4306
blade tool for making cuts in the canvas

Art and Hardware Store Parts

Some of the parts you will need aren't available from Adafruit, but should be easy to find at an art supply store and/or a hardware store. These will be covered in full detail in the next section:

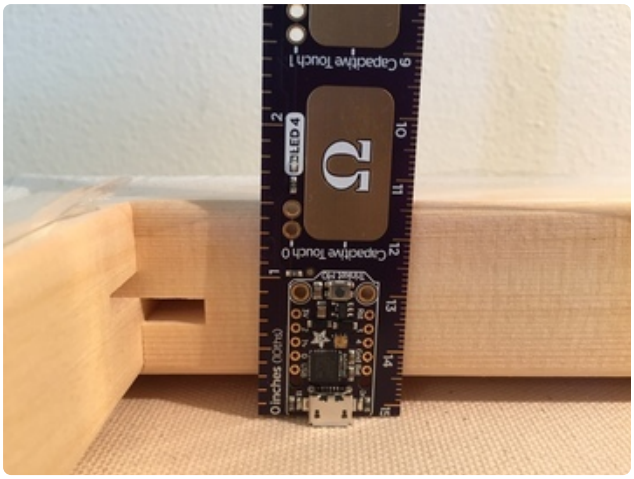
- Art canvas - 11x14 inch - "heavy duty"
- D-ring hangers, for attaching to back of canvas
- Hanging wire, attaches to D-ring hangers, for securing canvas to wall
- Cable staples, for holding cables to wood frame of canvas
- optional: foam board for back of canvas
- optional: offset clips to hold foam board to back of canvas

---

Canvas Enclosure Parts

To get started on a CompuCanvas, you will need a few parts that are probably most easily found at an art supply store. In my area, we have Michaels (https://adafru.it/HAa), which is where I bought the parts shown here. I wasn't able to find URLs for the exact parts I bought, so I will show and describe them here. I think you will be able to find similar parts at an art supply store in your area or online.

Perhaps the most essential part for a CompuCanvas is the canvas! For this project, I recommend an 11x14 inch "Heavy Duty" style art canvas. You could probably make it work with a 9x12 inch canvas, or go bigger with something like a 12x16 inch canvas and have more room to spread things out. (PyRulers (http://adafru.it/4319) for scale.)



Many of the art canvases I've seen are very shallow in the back - maybe half an inch - which is not a lot of room to work with. I have found some canvases labeled "Heavy Duty", which seems to designate a deeper design with about 1.5 inches of room to work with behind the canvas. I like to place USB speakers behind the canvas (so you hear, but don't see them), and the standard, shallow canvases would not have enough room in back for any USB speaker I've found.

Besides the canvas, there are two other essential parts for building this type of enclosure. The hanging wire and "D-ring hangers" shown here will allow for hanging the CompuCanvas assembly on a hook or nail in the wall.

The above parts are all you really need for a basic CompuCanvas enclosure. But for a more deluxe build, try to find foam board, like the 8x10 inch pieces shown here, and these 0.25 inch "offset clips". The offset clips will be used to hold the foam board in place behind the canvas. When designing a look for the canvas and laying out the electronic parts and cables, you may find that it works better to attach some parts to the foam board, instead of the canvas.

Preparing the Canvas

Before setting any computer parts on the canvas, I prepare the back so it can hang on the wall. These art canvases come wrapped in plastic and I recommend leaving most of the plastic wrap on at this stage to protect the front of the canvas. As you will see below, I'll cut part of the plastic wrap in the back to allow for attaching the D-ring hangers and the (optional) offset clips for the foam board backing.

In addition to the parts shown in the previous section, we will need several common tools for this section. First, we will need to cut away some of the plastic wrap in back with a knife or similar tool.

Shown here are 3 tools, any of which can be used to cut out a rectangular section of plastic wrap behind the canvas: an "X-Acto" style knife, a pocket knife and a ceramic blade like this Slice Craft Knife (http://adafru.it/4306) carried by Adafruit.

Choose your weapon and then cut away the protective plastic in the rectangular section behind the canvas. This will allow for leaving the front protective plastic wrap on while attaching the hanging hardware in back.

Now that the protective plastic has been removed in back we can access the space just behind the canvas. Here I have opened up the hanging wire and D-rings. Curiously, the D-rings come in packs of 3. We will only need 2 D-rings for this. And there is enough hanging wire in this pack for many, many CompuCanvases.

Shown here are two more basic tools we'll need: a wire cutter to cut a section of the hanging wire and a Phillips screwdriver for the D-ring screws.

I'm not sure if there is an optimal place for the hanging wire, but I recommend placing it near the top, a few inches down. On this canvas the paper labeling extends down about 3 inches. I poke a starter hole with a screw, as shown in the picture, in the middle of the wood just below the paper.

Then screw in this D-ring, and repeat the process of marking a spot and screwing in the D-ring on the other side of the canvas.

Now with both D-rings attached, cut a section of the hanging wire about 17 inches (my rule of thumb for this is to take the width of the canvas - 11 inches - and add 6 more inches to that).

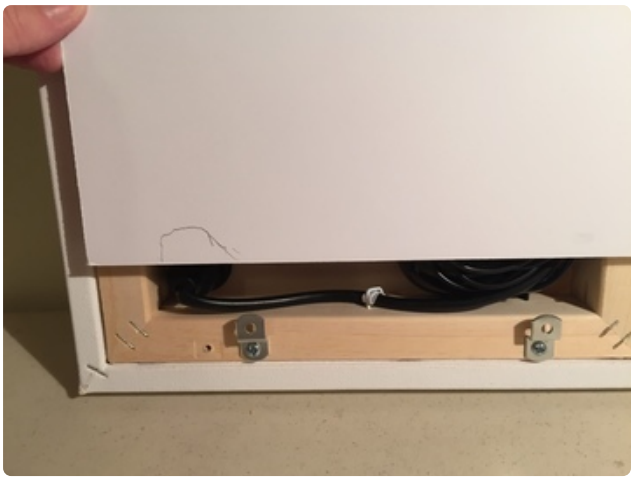
I crimp the wire with my fingers to measure out the middle section. It's important to leave a little slack so it's not too difficult to hang on the wall, but don't leave so much slack that the wire is visible over the top of the canvas.

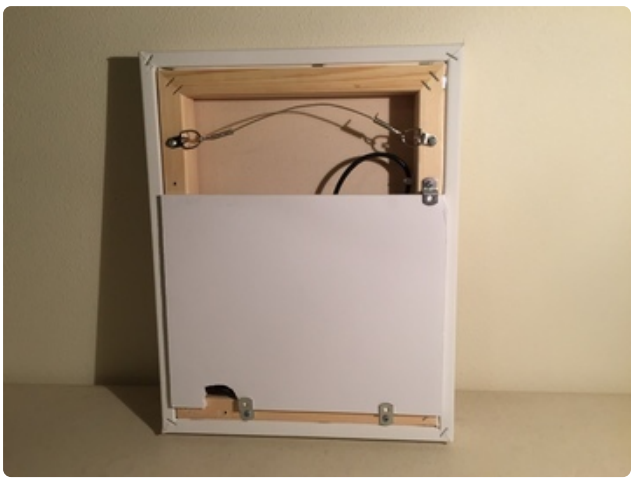
This series of pictures shows how I wrap the hanging wire around itself to attach it to the D-rings. Looping it around 7-10 times should be plenty to keep it on the wall, and you can cut away any extra hanging wire with the wire cutter tool.

Now, repeat the wire-looping exercise described above on the other D-ring and you will have a canvas that can hang on the wall!

Note in this picture how there is just about half an inch of slack in the hanging wire. It's not an exact science, but this is about right.

If you got the 8x10 inch foam board and offset clips for attaching parts behind the canvas, now is the time to add them. Get 2 of the offset clips with their screws ready for the next step.

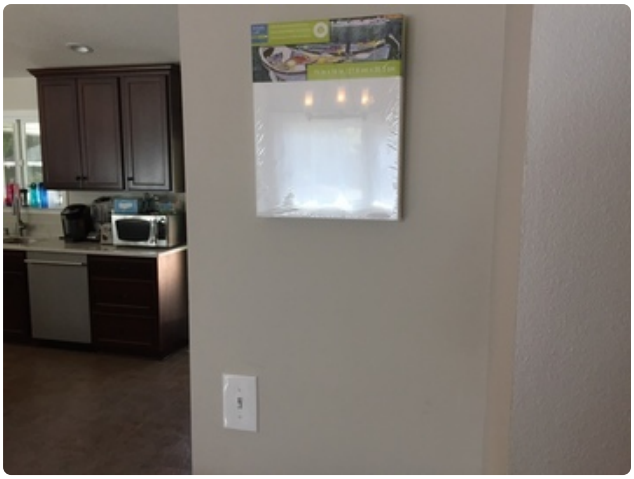




Warning! When I took the original pictures here I thought the offset clips would be fine about 2.5 inches from the edge of the canvas. But midway through the next section, I realized that on the left side, this will interfere with one of the cables.

The additional pictures here show how I moved the left offset clip another inch toward the center. I also added another offset clip on the top right (see the final picture).

Note also that the holes here are little below the center of the wood frame, but not so low that it might break through the edge of the wood as it gets screwed in. For the offset clip on top, make sure to leave a little extra space so the foam board can easily slide in and out.



Now you have a blank canvas that hangs on the wall. Proceed to the next section to learn how to place a Raspberry Pi and other computer parts into the canvas assembly.

Computers on Canvas

At the end of the last section, we had a blank canvas that can hang on the wall, but with no electronics. The goal for this section is to end up with a canvas that is still mostly blank - allowing for an infinity of further design possibilities - but it will have a running Raspberry Pi system behind the canvas, with one wire bringing in both power and Ethernet.

These pictures show the end goal for this section. Behind the canvas is a running Raspberry Pi model 3B with a Power over Ethernet adapter and a USB speaker. The front of the canvas is still primarily blank, except for 4 small nylon screws holding the Raspberry Pi behind the canvas and the circular Ethernet jack.

Next I will describe the parts and tools needed to build the Raspberry Pi system. Before I do that, I want to disclaim that I have some biases when it comes to building Raspberry Pi systems: I prefer Ethernet to WiFi and I like to use a rPi2B (http://adafru.it/2358), rPi3B (http://adafru.it/3055) or rPi3B+ (http://adafru.it/3775) because of the 4 CPU cores and 1 GB RAM.

But there are a zillion ways to build a CompuCanvas! You could take a more minimal (pure WiFi) approach with a Raspberry Pi Zero W (http://adafru.it/3400), or go even more minimal and build the system around something like a Circuit Playground Bluefruit (http://adafru.it/4333). Or you could go bigger with something like a

rPi4B (http://adafru.it/4296). However, changes like that will necessitate other changes to parts and layout that I haven't considered here. So you may want to continue following along with my selection of parts and steps.

The main picture here shows roughly how I will arrange the rPi system behind the canvas. The second picture shows the key parts (the PyRulers (http://adafru.it/4319) are just shown for scale).

The parts you will need with their Adafruit links are listed just below.

Note that we will use some parts from the nylon screw and standoff set in this section and then use more parts from this set in later sections.

1 x Raspberry Pi 3 - Model B — https://www.adafruit.com/product/3055

Raspberry Pi mainboard. You could also use a model 2B or 3B+ for this project.

---

1 x PoE Splitter with MicroUSB Plug (5V, 2.4 Amp) — https://www.adafruit.com/product/3785

Power over Ethernet splitter which will deliver 2.4 Amps of power to the rPi.

---

1 x RJ-45 Ethernet Round Panel Mount Extension Cable - 30cm — https://www.adafruit.com/product/4215

Panel mount cable to bring Ethernet (and power) into the canvas.

---

1 x Mini External USB Stereo Speaker — https://www.adafruit.com/product/3369

USB speaker to place behind the canvas and connect to the rPi,

---

Mounting hardware to allow for attaching the rPi and other electronic parts to the canvas.

1 x White Nylon Screw and Stand-off Set — https://www.adafruit.com/product/3658

Mounting hardware to allow for attaching the rPi and other electronic parts to the canvas.

Also, you will need a microSD card for the Raspberry Pi filesystem and you may need an adapter to write a Raspbian image on the microSD card. In case you don't already have these parts, the links are just below.

1 x MicroSD Memory Card — https://www.adafruit.com/product/2693

microSD card for Raspberry Pi

---

1 x USB MicroSD Card Reader/Writer — https://www.adafruit.com/product/939

USB adapter for programming microSD cards

The parts above will be placed inside the canvas (except for the USB microSD card reader/writer, which is a tool used for programming the microSD memory card). There are some additional parts you will need, to bring power and Ethernet into the canvas.

I like the way that flat Ethernet cables drape from the front of the canvas down to the floor, so I look for cables like the one shown here. This one is seven feet long, but you may need something longer depending on how you place the CompuCanvas in a room.

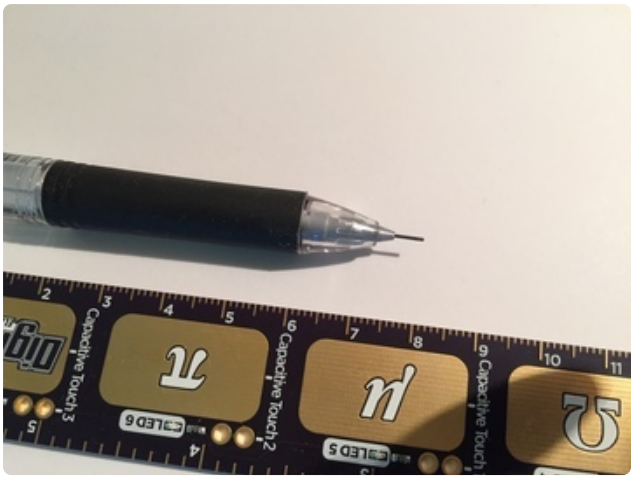
The other part here is a Power over Ethernet (PoE) injector. This will take power from a wall plug and Ethernet from your home router as input and send out PoE. You don't need the specific TrendNet part shown here, but you do need to use an IEEE 802.3af PoE injector to be compatible with the PoE splitter (http://adafru.it/3785) from the parts list above.

I couldn't find these parts at Adafruit, but they (and similar/compatible parts) are available from many online sources.

There are a few more parts and tools you will need for this section. For most of these, the hardware store is where you will likely find them.

The pictures here show hardware parts that I have used to attach cables to the wood frame of the canvas. I recommend the style of staples shown on the right, if you can find them. These can be pressed in with your thumb such that they will hold temporarily while you layout the parts. Then you can hammer them in more later when your layout is certain.

You will only need 2 or 3 of these staples for this project, so don't buy everything you see here! Just one pack will be enough.



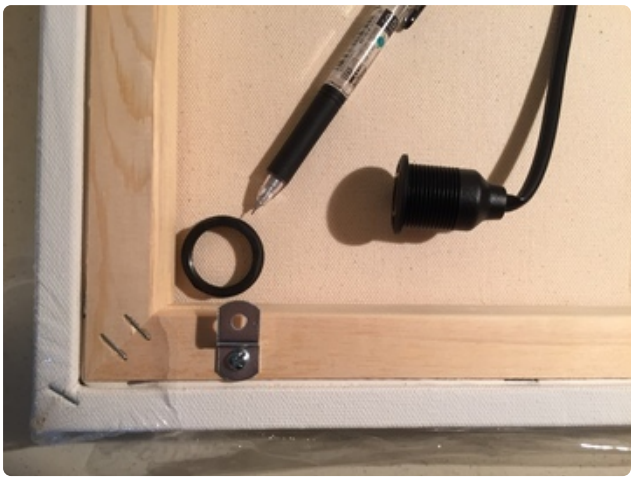
This picture shows some additional tools you will need. The hammer, shown on top, is used for final placement of the cable staples discussed above.

Shown on the lower left here are needles and awls which I use to poke small holes in the canvas. Adafruit sells a needle set (http://adafru.it/615), but I have also collected some other needles which are a bit larger. I often start by poking a hole with one of the needles to mark a spot and then use one of the awl tools to make the hole large enough for the nylon screws discussed above.

On the lower right are blades for making straight cuts or large rounded cuts in the canvas. These were already introduced in the previous section of this guide.

Finally, I use a mechanical pencil to mark spots on the canvas for cutting or poking holes. And the PyRulers (http://adafru.it/4319) shown here are not just for scale! You will want a ruler around to measure for some of the layout activities.

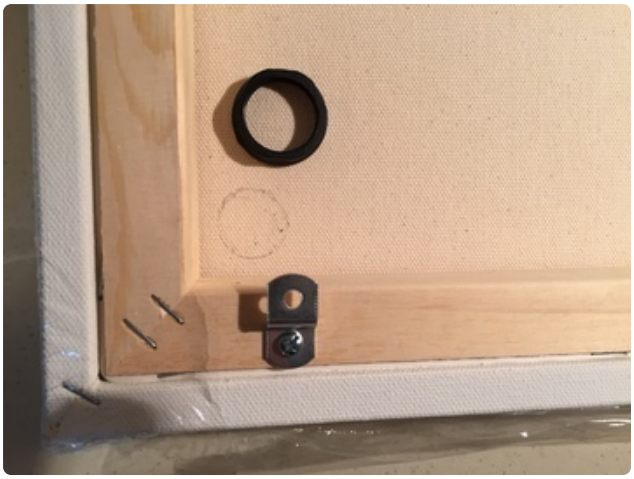
Now, with all of these parts and tools assembled we can get down to business. The next few steps will show how I arranged the computer parts behind the canvas and started to mark spots to cut through the canvas.

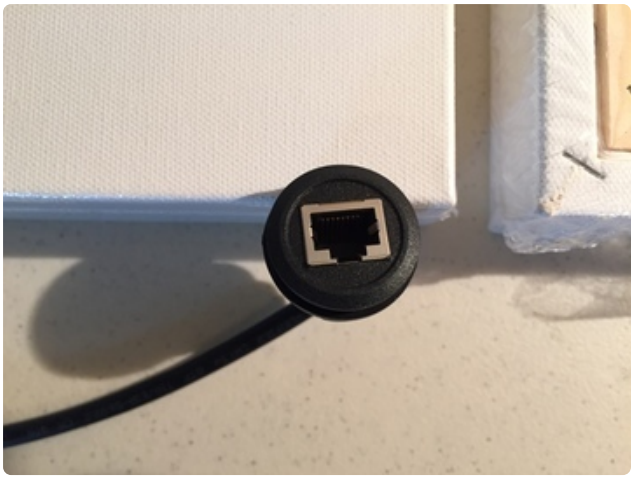


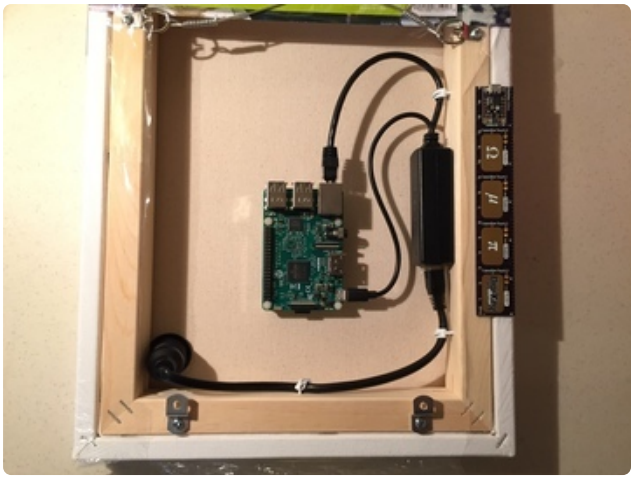
I'm going to start by laying out the panel mount Ethernet cable. The pictures here show how I removed the circular piece that holds the cable from behind and used it to trace out a circle where I will cut through the canvas.

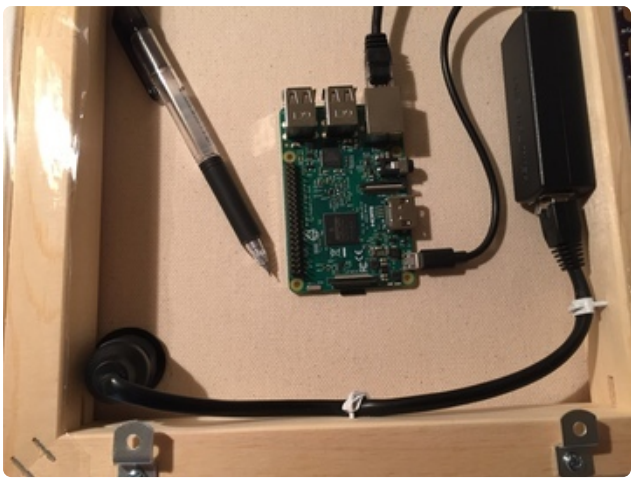
Some other things to note here:

I used one of the cable staples, placed at an angle, to direct the cable behind the canvas. Don't hammer the staple in yet because we still need to thread the cable through the front. For now just thumb-press it in to mark a spot for it near the center of the canvas.
Ethernet cables have a notch (as the last picture shows) and this is often oriented down (toward the floor), or it could be on the top, but you probably don't want it in some random orientation, so pay attention to this as you proceed!
Once you figure out the cable orientation you want, massage/crimp the cable at a right-angle so it will run toward the other side of the canvas.
This cable is deep enough that it will interact with the foam board. As mentioned in the previous section, I moved the offset clip near this cable transition to the right. This will be shown in more detail in an upcoming section.





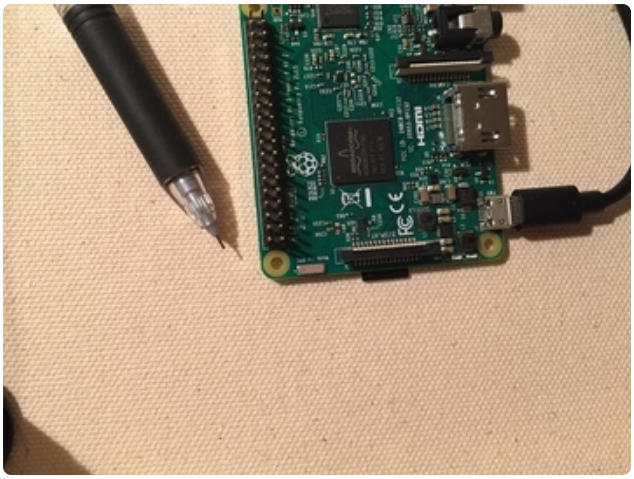


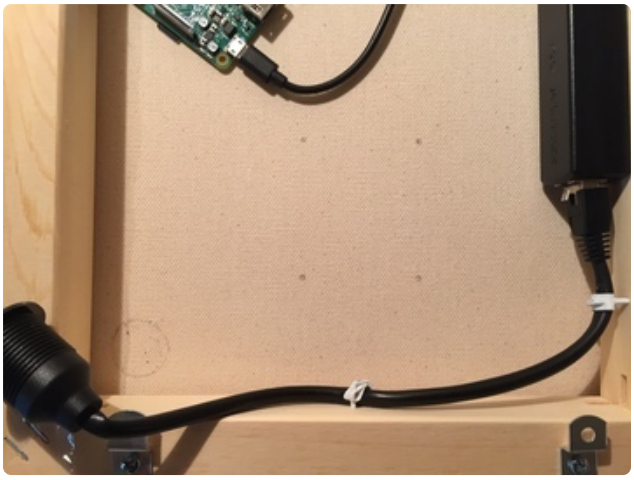


Here I am preparing to mark 4 spots to poke holes for mounting the Raspberry Pi to the canvas. The key things to note are:

In my experience, the actual length of cables can vary a bit from the advertised length. There is no substitute for plugging things together and observing where everything fits (and doesn't). Thus, I have the rPi, the PoE adapter and the Ethernet cable all connected together and pinned to the sides of the frame with staples.
The first pictures show 3 staples, but the center staple is not present in the last picture. When I got to fitting the USB speaker, I realized that this staple was interfering and should not be used.
I want the USB speaker to nestle behind the Ethernet cable, as shown in the last picture. I cut this very close! The rPi needs to be about 4.3 centimeters above the frame to fully clear the speaker. Use the speaker, the rPi and a ruler as necessary in the next step to find a positioning that will work.
Some of these pictures show the nylon screws in the rPi mounting holes. This will be covered in detail in upcoming steps.

When arranging computer parts on these art canvasses, be careful not to place any of the chips that may get hot directly against the canvas. Some of the chips on the Raspberry Pi can get quite hot. The design I present here leaves plenty of air gap between the hot parts of the rPi and the canvas.



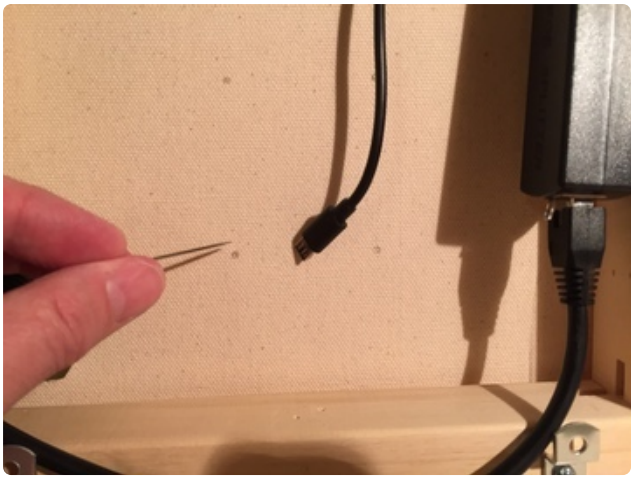


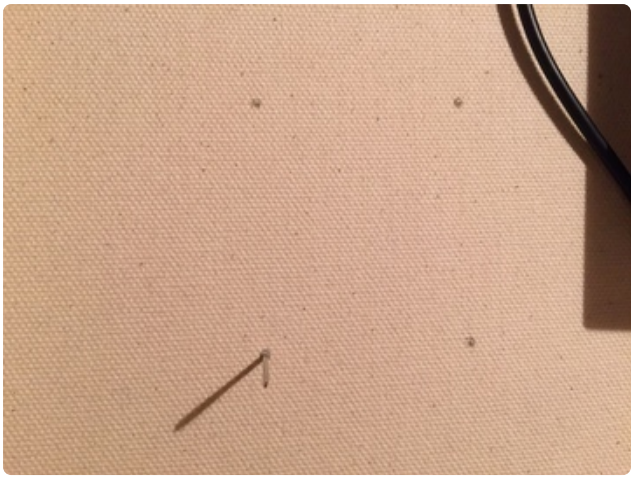
Now, hold the Raspberry Pi down in the exact spot where you want it on the canvas and use a mechanical pencil to mark the 4 mounting holes, as shown in the second picture here.

I find that I have to extend the lead from the pencil much more than I would for writing on paper to allow it to reach the canvas. This means it can break easily. But you don't need dark marks for this. Just gently circle the pencil around in each mounting hole and you should get clear enough markings for the next step.

Now it is finally time to make some cuts in the canvas!

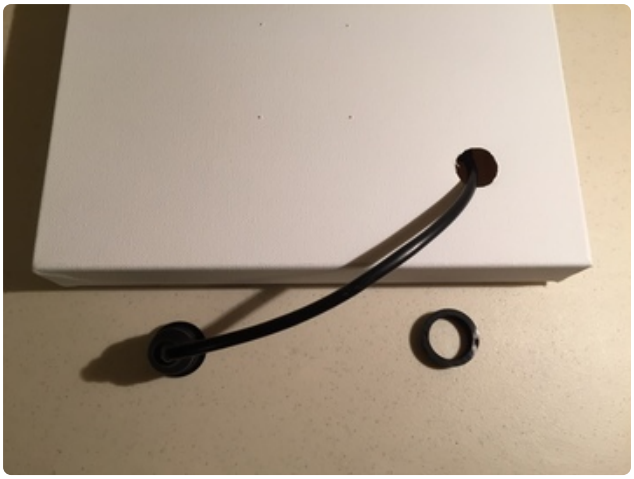
First, use a blade type tool to carefully cut around the circle you traced for the Ethernet cable.





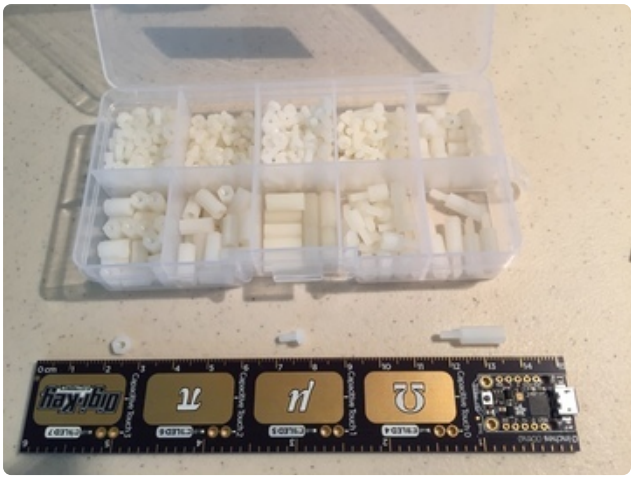
Next, use a needle to poke through the center of each of the 4 rPi mounting holes. We will make these holes larger (with the awl) in an upcoming step. For now you just need holes that are visible from the front of the canvas.

The last picture here shows the canvas from the front. Note that the protective plastic is still (mostly) covering the canvas. If you peek ahead to the next step, you'll notice the protective plastic is gone. You know what that means! Yes, go ahead and rip off all the extra plastic to fully reveal your canvas!



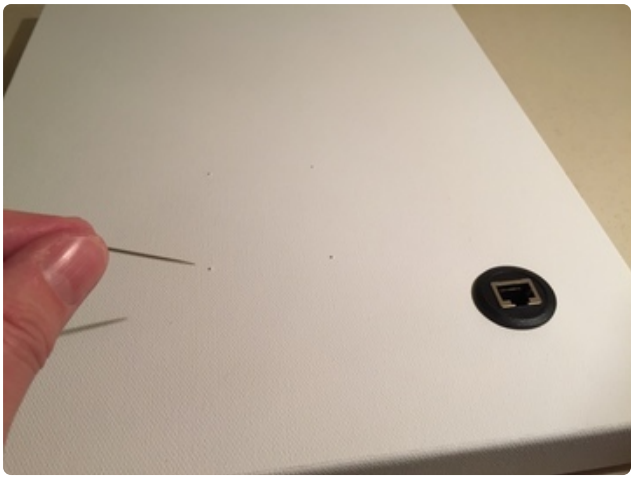
Here I have threaded the Ethernet cable through the front of the canvas, and screwed the plastic nut in back to secure it in place.

It was surprisingly difficult to turn the nut around with my fingers, so this step took a while. You may find that a pliers or wrench helps here, or just be patient and gently fiddle it around until it is flush with the canvas.



In the next step we will widen the 4 rPi mounting holes and place nylon screws through them. Here you just need to collect the nylon screw parts to use. The first picture shows how the nylon standoffs will be attached to the rPi. The second picture identifies which parts to use from the nylon screw kit, but you need 4 of each:

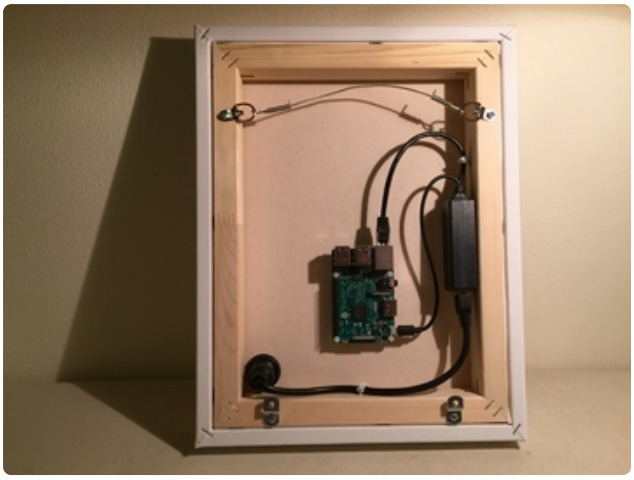
4 of the nuts
4 of the medium sized screws
4 of the longest standoffs
You can go ahead and attach the standoffs to your rPi as shown in the first picture here.



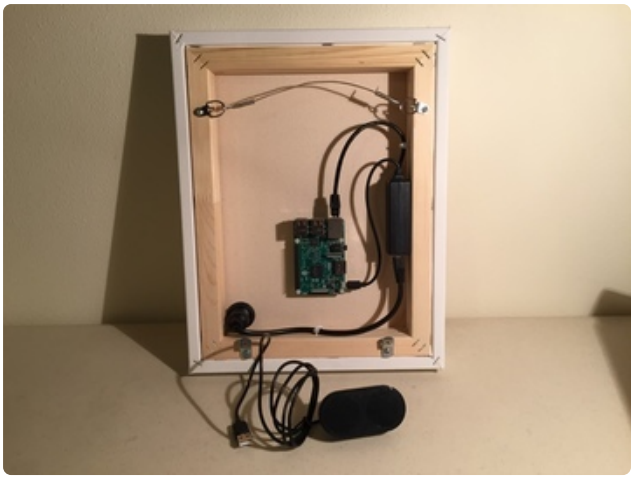
The 4 nylon screws need to be inserted through the front of the canvas now, so we can attach the rPi in back.

To accomplish this I start by poking a needle through the holes in front and wobbling it around to make the holes a little wider. Then I insert the awl and gently twist and push to make the hole even wider.

Be careful not to make the holes too big! This step can take some time. I gently work on making the holes bigger until they reach the point where they are just wide enough to place the screw and use a screwdriver to twist it all the way in.

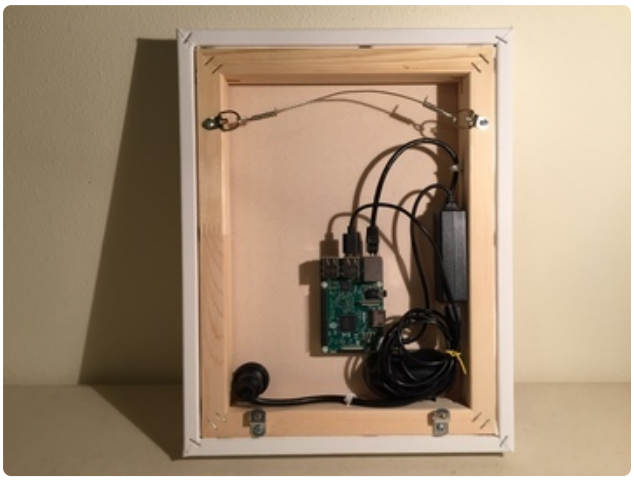


The next step is to secure the Raspberry Pi in place. Hold it up so the standoffs align with the screws and then use a screwdriver to attach each one.



Now I am positioning the USB speaker behind the Ethernet cable.

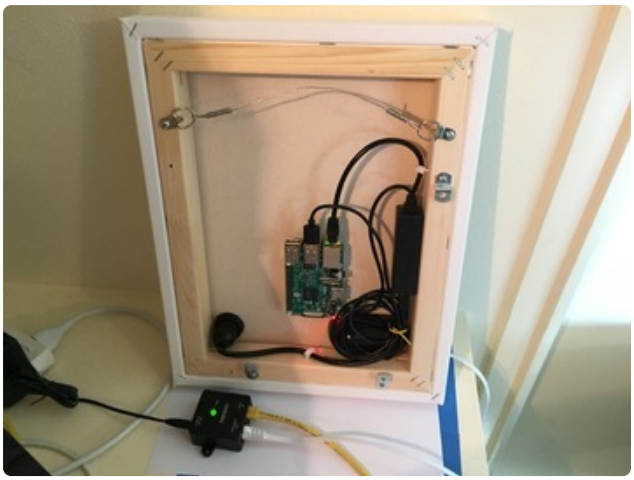
The speaker cable is much longer than is needed here. In the last picture, I've used a yellow twist-tie to attach the extra speaker cable to the Ethernet cable. This helps keep the speaker in place and keeps the extra cabling out of the way.



You will also need to plug in a microSD card with Raspbian (https://adafru.it/fQi), or your preferred OS.

As the pictures here show, the microSD slot is under the rPi on the bottom. It should be inserted with the traces facing toward you.

If you are not sure how to prepare the microSD, don't panic! The next section in this guide will cover the software side of getting your CompuCanvas up and running.



Now we're almost ready to boot this system up! The pictures here show how the PoE injector gets connected in front of the canvas. The yellow Ethernet cable connected to "DATA IN" needs to also be connected to a router or switch on the other end.

Ok, now take a deep breath and admire the pure, pristine beauty of your CompuCanvas enclosure. It's like the Beatles "White Album"! It's like a field covered with fresh snow!

But things are about to get messy. We've been doing physical construction, but now it's time to do software. The next section will cover details from how to setup a microSD card with Raspbian to how to find the CompuCanvas on your network, login and do initial configuration of the system.

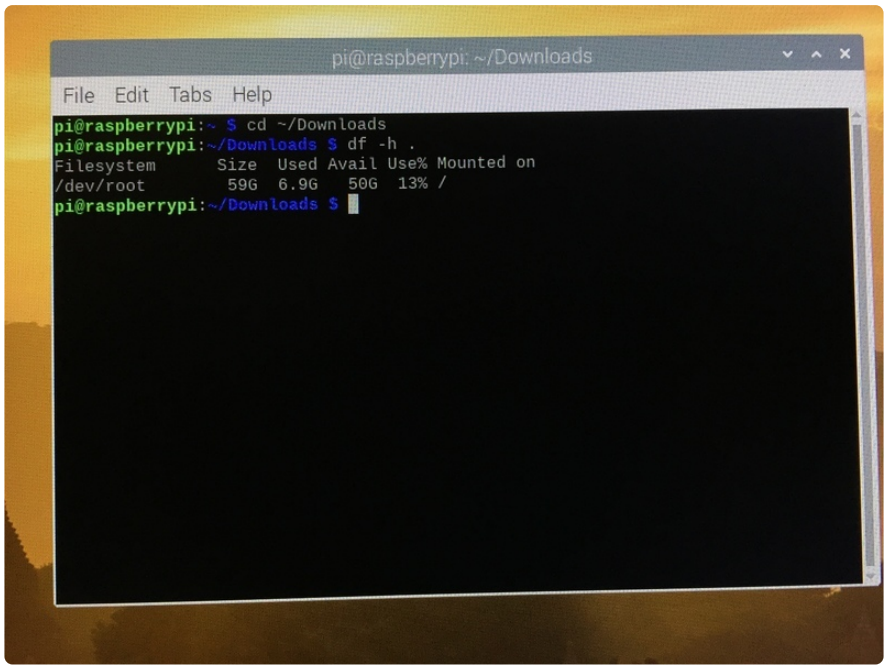
Digressions in Code

In this section I'll first show how to prepare a microSD card with Raspbian to boot on your CompuCanvas. Then I'll walk through some of the setup steps that I perform to configure a new system. If you already have your CompuCanvas running you can skip over the initial material here.

Preparing a microSD card with Raspbian

If you want to use a Windows machine or a Mac to prepare your microSD card, refer to this other learn guide about rPi setup (https://adafru.it/dDL). I noticed that guide does not show how to use a Raspberry Pi to do the setup. I often use one rPi system to prepare the microSD card for another, so I'll detail those steps here to complement the other guide. I'll be using a Raspberry Pi with a monitor, keyboard and mouse connected to it. But these steps could be performed with a headless rPi (like a CompuCanvas) as well.

The first thing to do is download a Raspbian image... but even before doing that you should check to make sure you have enough room for downloading and unzipping one of these large files. You can use the `df` command to figure out how much free space you have. The screenshot below shows how I changed to the `Downloads` directory and then ran `df -h .` and that shows I have about 50 Gigabytes free (on a 64GB microSD card), which will be plenty.



As I write this, the Raspbian downloads page (https://adafru.it/fQi) has 3 editions of "Buster". Any of them should be ok for running a CompuCanvas. I usually choose the medium sized one: "Raspbian Buster with desktop". It's ok to use the 'desktop' editions even though the CompuCanvas won't have a proper monitor setup.

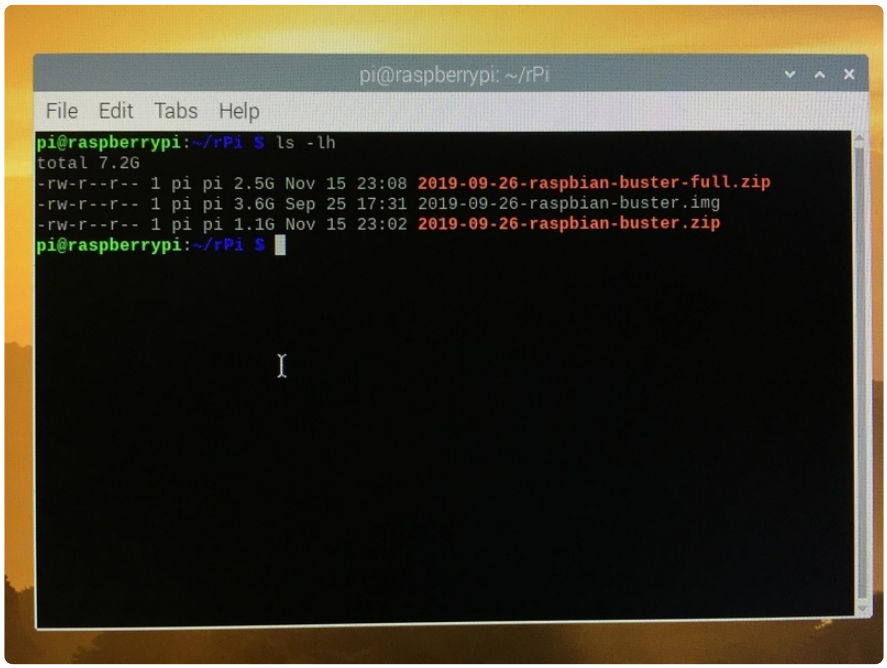
The download (https://adafru.it/jE0) I got is called `2019-09-26-raspbian-buster.zip`. The size of this file is about 1.1 Gigabytes, but we need to unzip it (with the `unzip` command), and that will produce a file with the same name but a `.img` extension. This 'image' file is about 3.6 Gigabytes. So I will need almost 5 GB of free space on my system to perform this operation. As shown in the picture above, I have plenty of space, but you should check this too before downloading and unzipping the image.

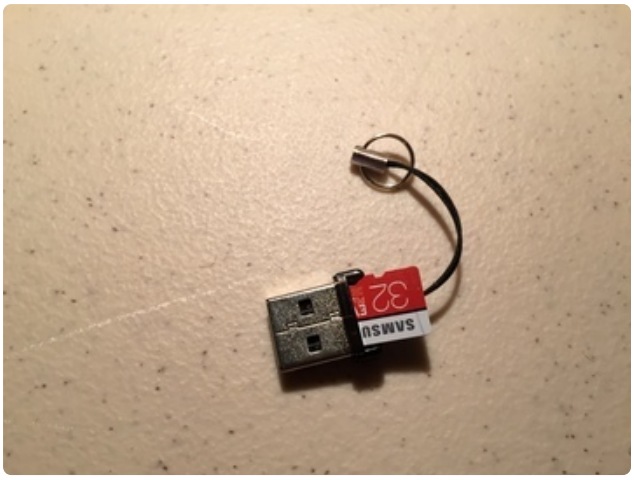
If you are working entirely from the command line, you can use the `wget` command to download the zip file, like this:

```
wget -O raspbian.zip https://downloads.raspberrypi.org/raspbian_latest
```

If you are using a browser, you can click the link and it will be downloaded to the `/home/pi/Downloads` directory.

In the picture below, I've download two of the Raspbian zips and moved them to a directory named `rPi` that I created to organize them. I have also unzipped one to get the `.img` file that will be copied to the microSD card.





The next step will be to use a USB to microSD adapter (http://adafru.it/939), like the part from Adafruit that was introduced in an earlier section of this guide. We will also need a microSD card.

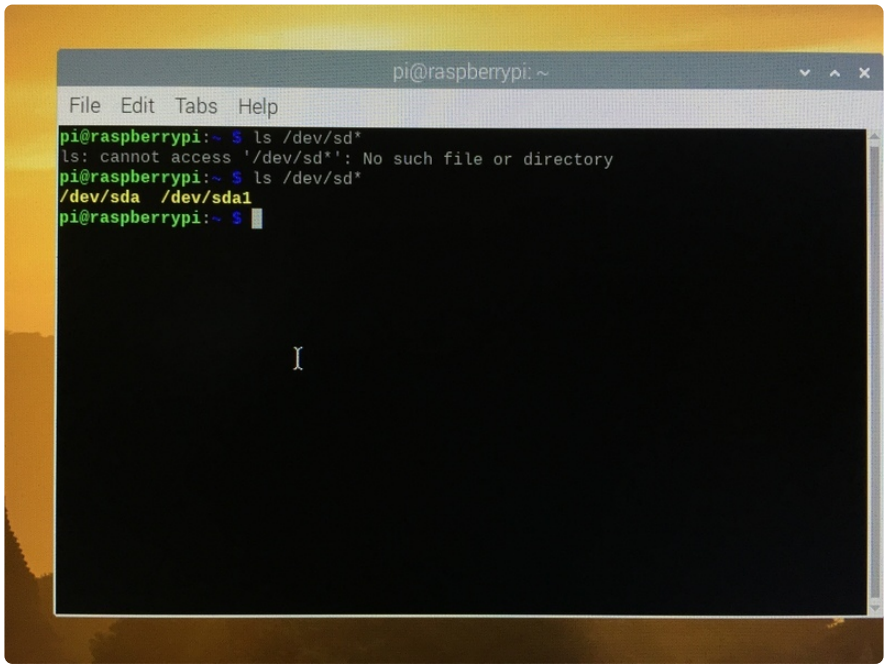
The pictures here show inserting the microSD card into the adapter.

Next we'll want to plug this into the rPi system where we have the Raspbian image file. When we plug in the adapter, it should become visible as a device with a filename like `/dev/sda`, `/dev/sdb` or similar.

In the picture below you can see I ran `ls /dev/sd*` before plugging in the microSD adapter. On the first run there were no files found matching that pattern. Then I plugged in the adapter and ran the same command again. Now I see a `sda` and `sda1` files in `/dev`. The first one represents the physical device and the second is a logical volume on that device. If we had a second of these adapters plugged in,

we would see `/dev/sdb` and some logical volumes on it matching the pattern `/dev/sdb*`.

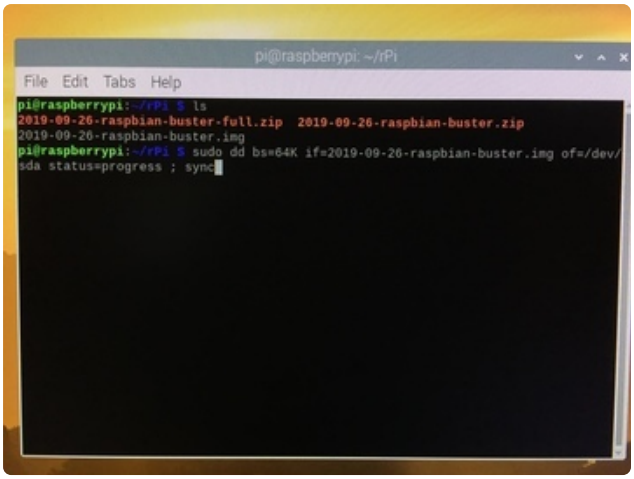
Checking this is important, because the next step is to write the image file to the microSD card. Doing this will erase and overwrite any data on the device we write to, so it's really important to identify the microSD correctly before proceeding. My before-and-after test shown below is a good way of confirming the device name.

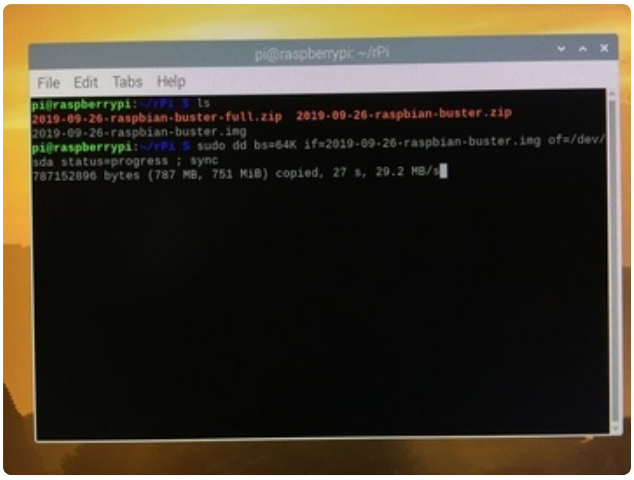


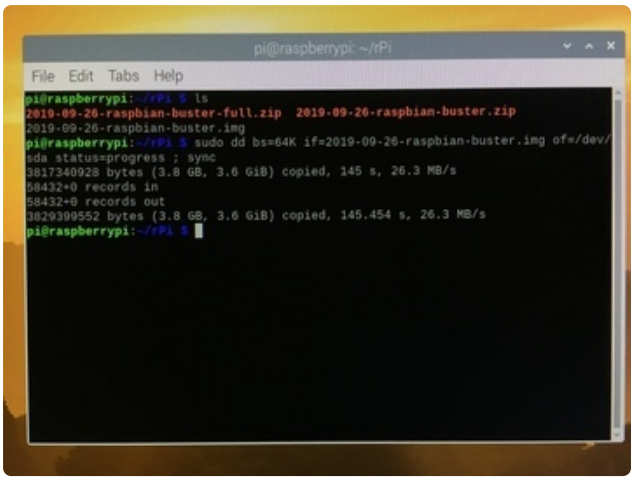
Now we can run the `dd` command to copy the image file to the microSD card. The magic command for this is shown just below. However, note that you will need to fix 2 things before running it. I used `raspbian.img` as the image file name, but you will need to replace this with the actual image filename from the zip download. Second, I used `/dev/sdXYZ` as an example device to write to. Replace this with the device we found in the previous step, which is probably `/dev/sda`, but I always double-check to avoid overwriting the wrong device.

```
sudo dd bs=64K if=raspbian.img of=/dev/sdXYZ status=progress ; sync
```

The `sync` command at the end is really important too! This makes sure that all the data gets written out to the physical disk (instead of being cached on the rPi) so that it will be safe to remove the microSD adapter.







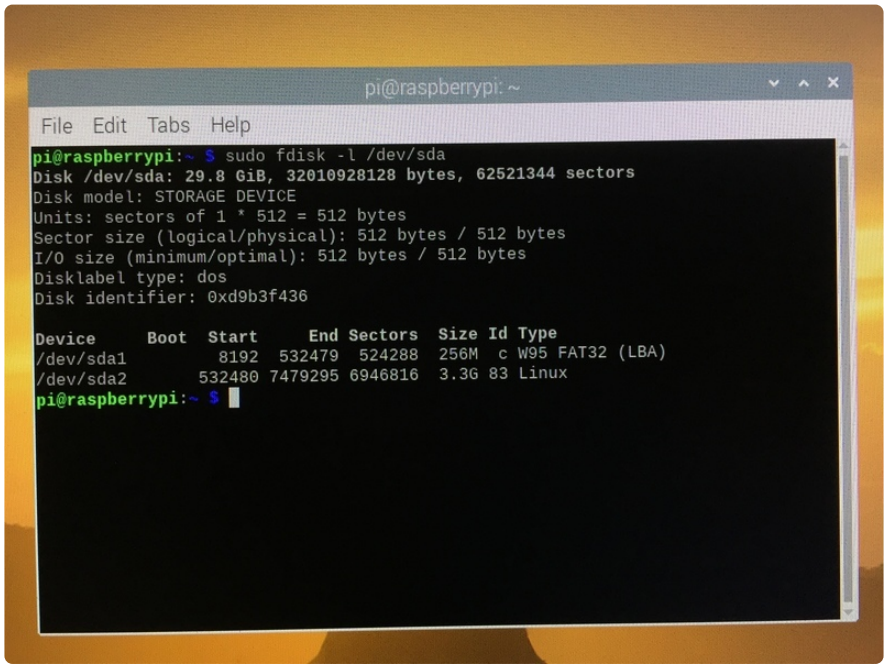
These pictures show how I constructed the `dd` command and then ran it.
The `status=progress` option gives feedback as the copy is happening, then a summary is printed when it finishes.

There is one more crucial step to perform before unplugging the microSD adapter. As a security measure, remote login (with the `ssh` command) is disabled by default on new Raspbian images. But remote login will be the primary way of controlling the CompuCanvas, so we need to enable it.

The method for enabling remote logins is to write a file named `ssh` at the top-level of the "boot" filesystem. But first we may need to mount the boot filesystem so we can write this file to it. When we wrote the Raspbian image file to the microSD card, it should have created 2 logical volumes, a "boot" volume and a "root" filesystem. If you are lucky, they will get auto-mounted under `/media/pi` but I checked there and was

not lucky, so I will need to manually mount the boot filesystem to write the `ssh` file on it.

The picture below shows how I ran `sudo fdisk -l /dev/sda` which prints some information about the 2 logical volumes (`sda1` and `sda2`) associated with the physical microSD card (`/dev/sda`). The device with "FAT32" in the type field is the boot volume that we need to mount.



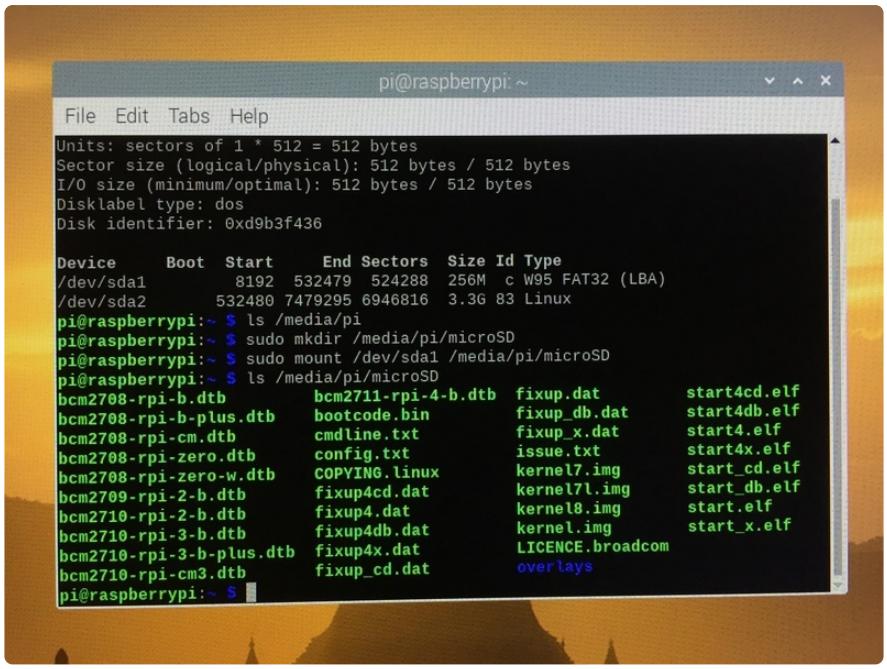
We will need an empty directory to mount the boot filesystem. When things get auto-mounted, they appear under `/media/pi` . In the picture below, I created a directory `/media/pi/microSD` to use as the mount point. This will only need to be done once on this machine:

```
sudo mkdir /media/pi/microSD
```

Then I ran the following command to mount the volume:

```
sudo mount /dev/sda1 /media/pi/microSD
```

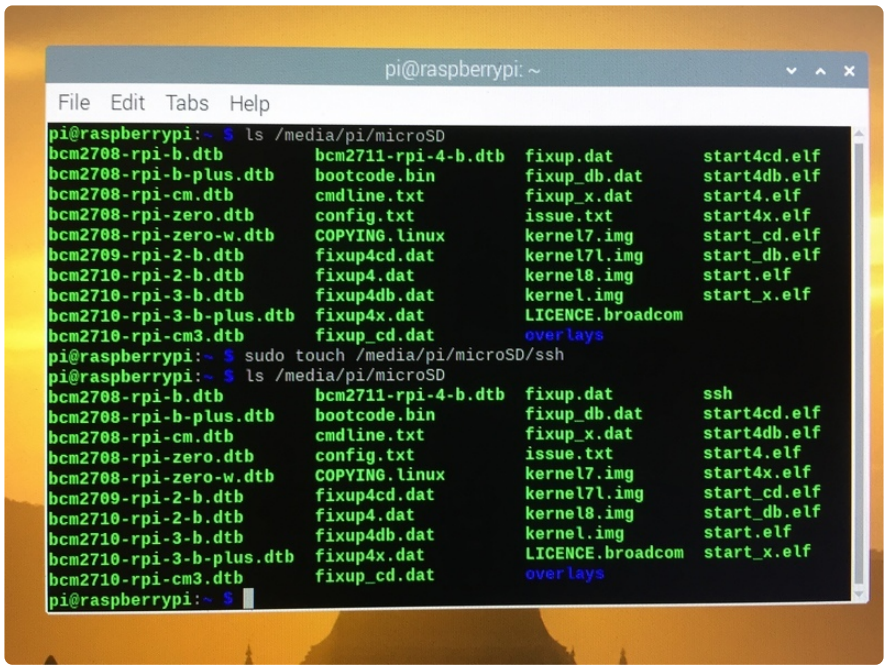
After this, as the picture shows, we can `ls /media/pi/microSD` and see a bunch of files that are used as the Raspberry Pi boots up.



Now I can create the magic `ssh` file that enables remote logins. This is accomplished with the `touch` command. which creates a new, empty file:

```
sudo touch /media/pi/microSD/ssh
```

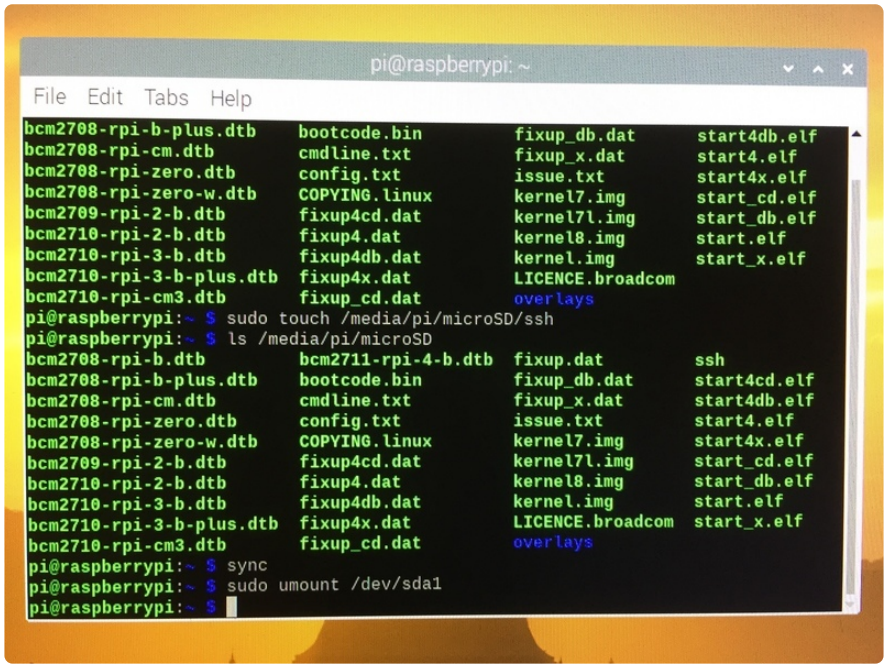
The picture below shows the before and after running the command. Notice the `ssh` file is present the second time we check that directory with `ls`.



Finally, you should run the `sync` command one more time to make sure all information is written out to the microSD card, and then unmount the root volume with this command:

```
sudo umount /dev/sda
```

This is shown in the next picture



Now you can remove the microSD adapter from this Raspberry Pi and put it into the CompuCanvas!

We can now proceed to the next sub-section of this software odyssey.

Logging into a new CompuCanvas

When you power up the CompuCanvas and it starts to boot up, it will get assigned an IP address on your network. You'll need to find that IP address and use it with the `ssh` command to login from the second Raspberry Pi. One way to find devices on your local network is with the `nmap` command. You might need to install it first, like this:

```
sudo apt update
```

```
sudo apt install nmap
```

Home routers often assign IP addresses where the first 3 digits are `192.168.1` and the final digit is between 1 and 254. So the CompuCanvas could pop up at an address like `192.168.1.33` or any one of about 254 other possibilities. With `nmap` you can search a range of addresses. You may need to adjust the IP addresses I show for your

own network/router configuration. The command below shows how to scan for all devices under `192.168.1`:

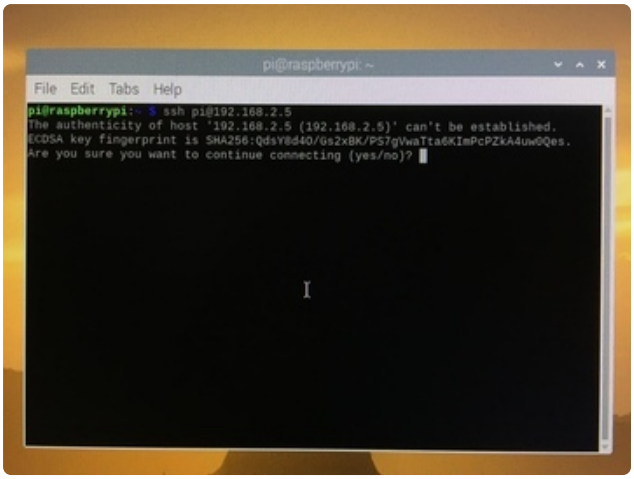
```
nmap -sP 192.168.1.1-254
```

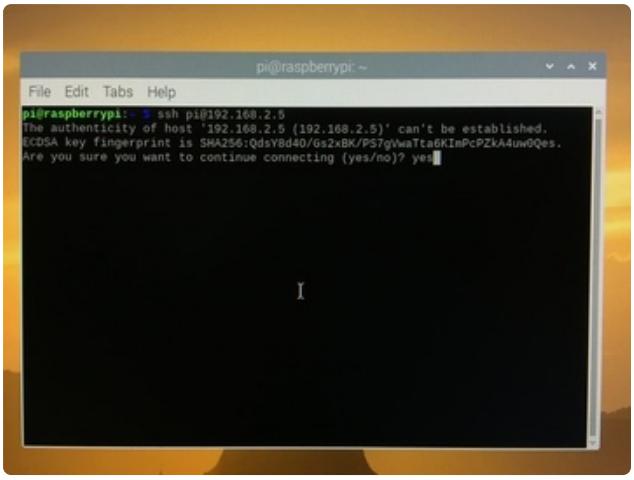
Similar to identifying the microSD card, you'll want to run this command before plugging in the CompuCanvas and get a list of devices currently on your network. Then plugin in the CompuCanvas and wait a few minutes for it to start up and get an IP address assigned. Then run the same `nmap` command again and look for a new device and note that IP address for the next sub-section.

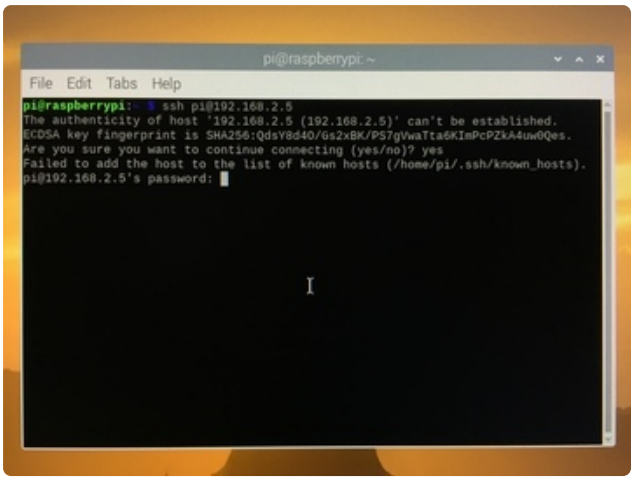
Suppose that your `nmap` command shows a new device at address `192.168.1.33` on your network. You can try to login with the `ssh` command like this:

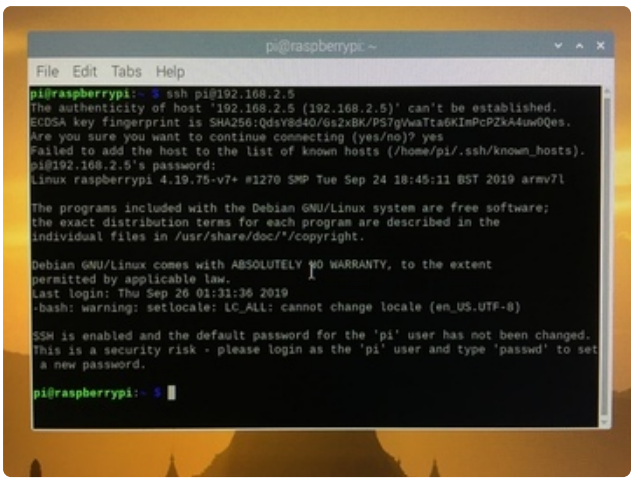
```
ssh pi@192.168.1.33
```

Sometimes I see several new/unknown devices in the `nmap` output and I have to guess. If you try to login to the wrong IP address, you will probably see a "connection refused" message.







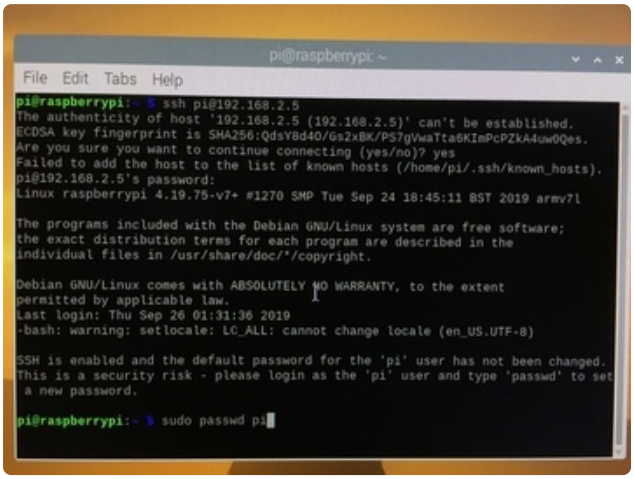


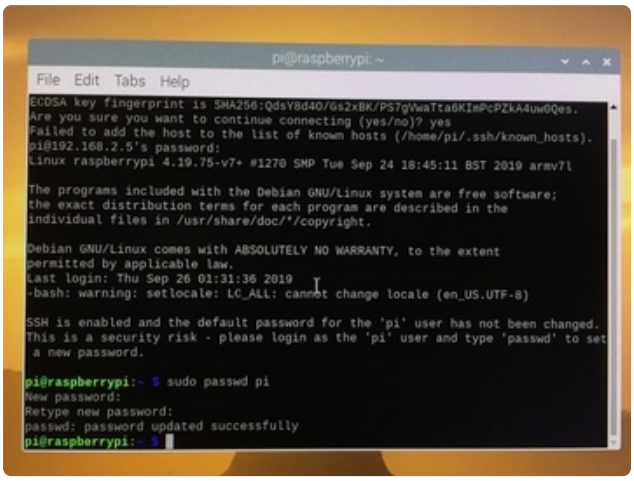
The pictures here show my initial login to the CompuCanvas. (Note that my subnet is actually `192.168.2` instead of `192.168.1`, as reflected in these pictures).

My host rPi doesn't trust this unknown machine, and I have to confirm the connection is ok by typing `yes` at the prompt.

Then I get a login prompt. In these fresh Raspbian images, there is a `pi` user account and the default password is `raspberry`. My `ssh` command specified the userid (with the `pi@...` syntax), so to complete the login I just need to type the default password.

Initial system configuration





The first thing I do when logging into a new Raspbian system is change the `pi` account password from the default to something else. As shown in these pictures, you can change your password with this command:

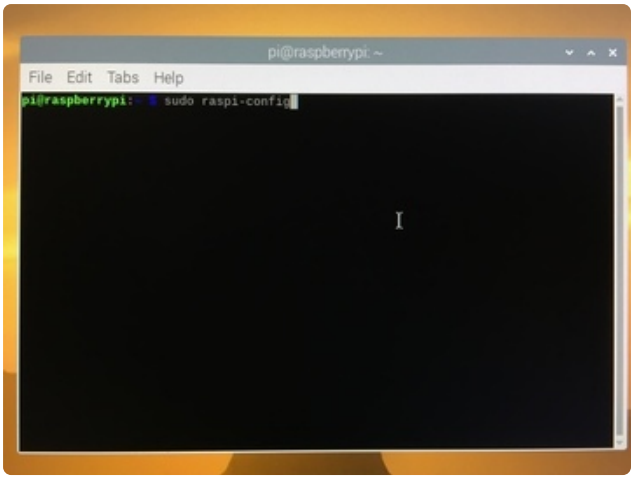
```
sudo passwd pi
```

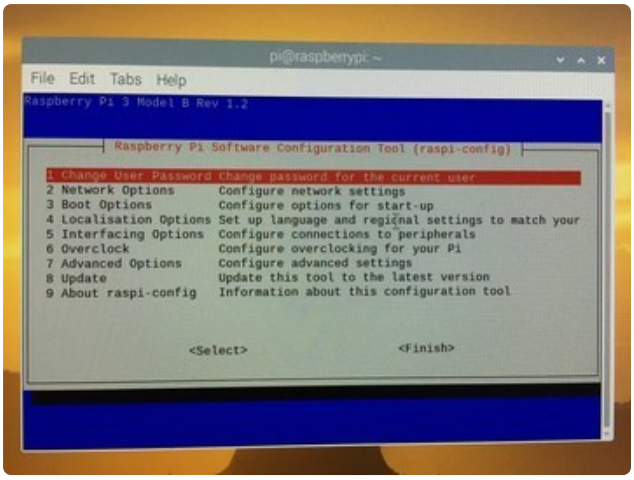
Then you need to type the new password twice to confirm the change. Please write down the new password in a safe place.

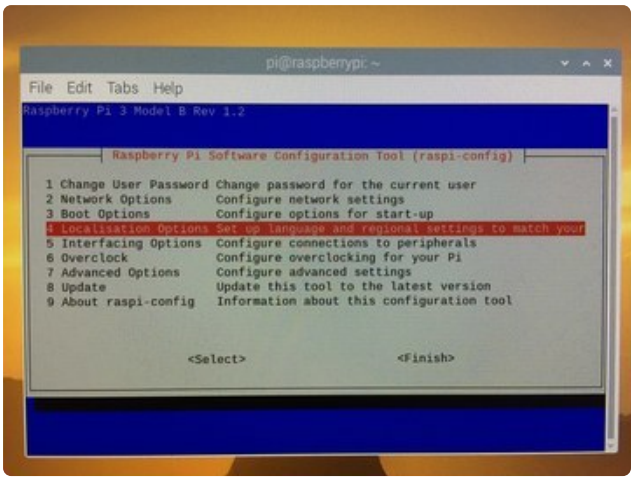
The next step I take in setting up a new CompuCanvas is to run the `raspi-config` utility. This command line tool can perform most of the initial configuration you may want for running the CompuCanvas on your network. To start this, type:

```
sudo raspi-config
```

The next series of pictures will show some of the features of `raspi-config`.







Some things to note about these pictures of the `raspi-config` utility:

We could have used `raspi-config` to change the password, instead of the command line way I showed above. Either way would work.

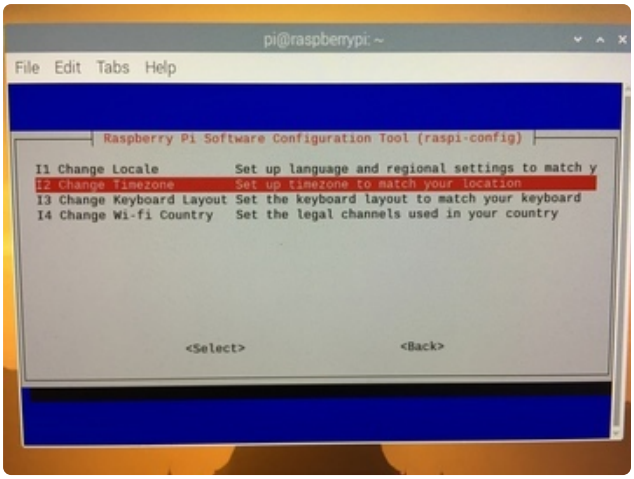
To navigate in `raspi-config` use the up and down arrow keys to select an action. Use the `Enter` key to perform the selected action. Use `Esc` to back out of a sub-menu. Use `Tab` to navigate between the top section and the actions like `<Select>` and `<Finish>` shown on the bottom of the screen.

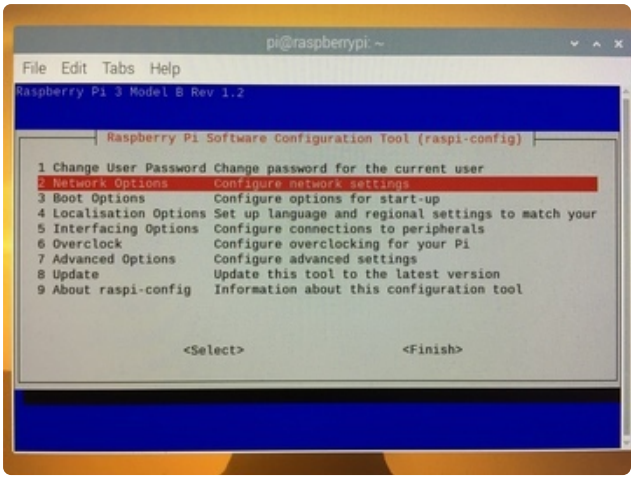
The first thing I do is set the system timezone, which is under "Localization Options".

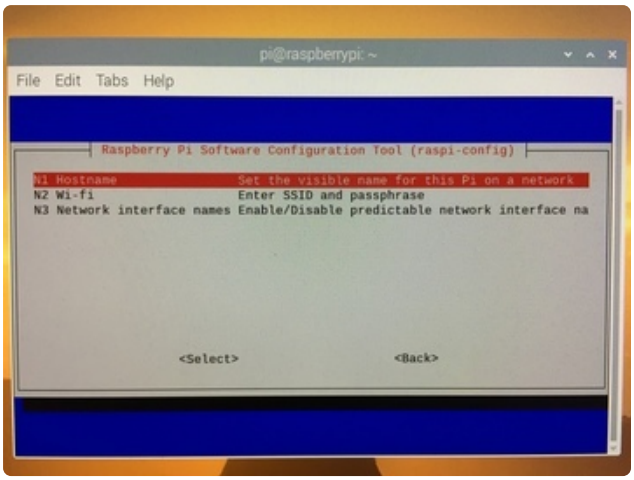
I also delve into the "Network Options" menu to set the system host name.

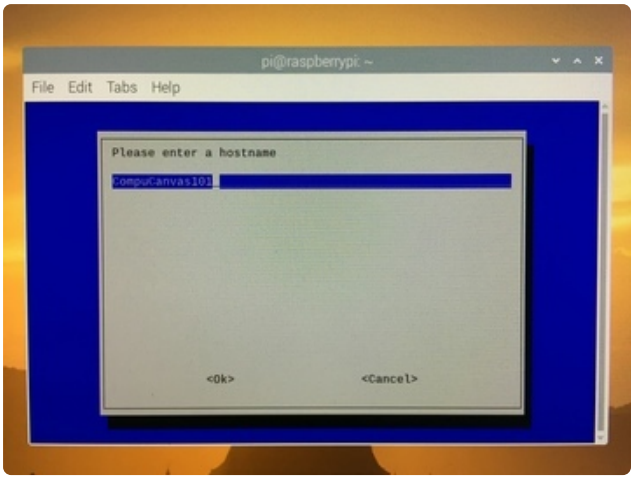
This is a "season to taste" kind of thing. You may have other configuration you want to do for your environment.

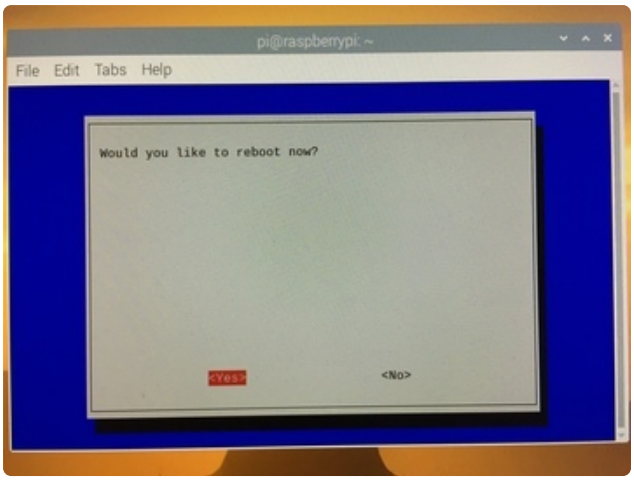
When exiting `raspi-config` it will prompt you to reboot. It is advisable to reboot after making the kind of changes I described above (e.g. hostname and timezone). You can go ahead and let `raspi-config` reboot, or do it from the command line with: `sudo reboot`.


pi@raspberrypi: ~
File Edit Tabs Help
Raspberry Pi Software Configuration Tool (raspi-config)
I1 Change Locale Set up language and regional settings to match y
I2 Change Timezone Set up timezone to match your location
I3 Change Keyboard Layout Set the keyboard layout to match your keyboard
I4 Change Wi-fi Country Set the legal channels used in your country
<Select>
<Back>


pi@raspberrypi: ~
File Edit Tabs Help
Raspberry Pi 3 Model B Rev 1.2
Raspberry Pi Software Configuration Tool (raspi-config)
1 Change User Password Change password for the current user
2 Network Options Configure network settings
3 Boot Options Configure options for start-up
4 Localisation Options Set up language and regional settings to match your
5 Interfacing Options Configure connections to peripherals
6 Overclock Configure overclocking for your Pi
7 Advanced Options Configure advanced settings
8 Update Update this tool to the latest version
9 About raspi-config Information about this configuration tool
<Select>
<Finish>


pi@raspberrypi: ~
File Edit Tabs Help
Raspberry Pi Software Configuration Tool (raspi-config)
N1 Hostname Set the visible name for this Pi on a network
N2 Wi-fi Enter SSID and passphrase
N3 Network interface names Enable/Disable predictable network interface na
<Select>
<Back>


pi@raspberrypi: ~
File Edit Tabs Help
Please enter a hostname
CompuCanvas101
<Ok>
<Cancel>

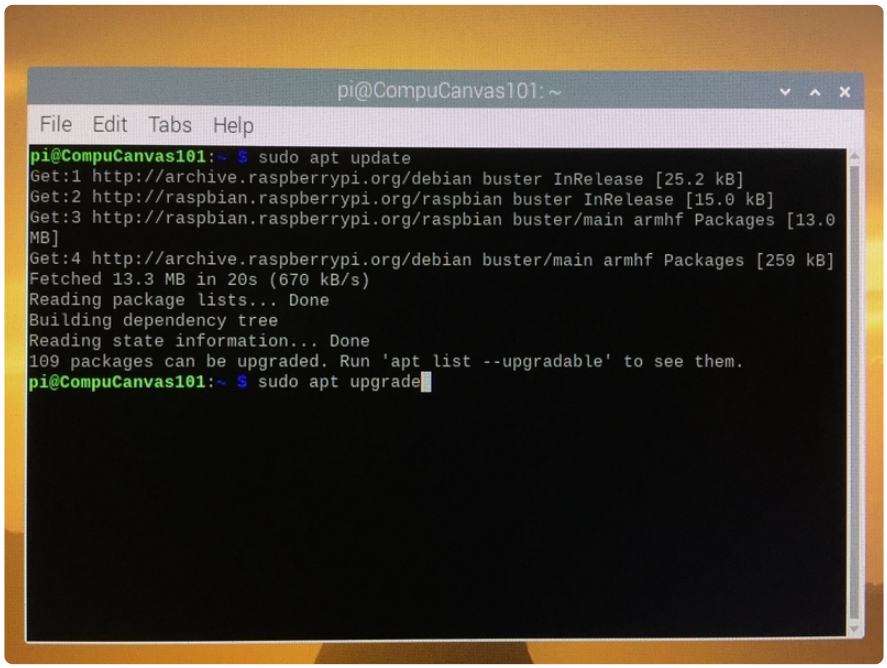


After the first reboot, there is still a bit more setup to do. I login to the CompuCanvas again with `ssh` and then run these 2 commands:

```
sudo apt update
```

```
sudo apt upgrade
```

This will upgrade the Raspbian operating system packages to their latest versions. The second command may take a long time, and there may be prompts where you need to type something to make it proceed. The picture below shows the beginning of this process.



It is a good idea to reboot again after this upgrading, but first you could install additional software to run on the CompuCanvas. There are 2 utilities that we will use later in the guide: `espeak` and `screen`. You should install those now with the following command:

```
sudo apt install espeak screen
```

You could install additional software at this point as well. After that, reboot again:

```
sudo reboot
```

The final bit of system configuration will be setting up the audio so we can hear sound from the USB speaker. Next, I'll transition back to physical construction topics - specifically lighting up the inside of the canvas.

Internal Lighting

In this section I'll show several different ways to place and control lights behind the canvas. I'll use the parts shown below as well as some more parts from the nylon screw and standoff kit.

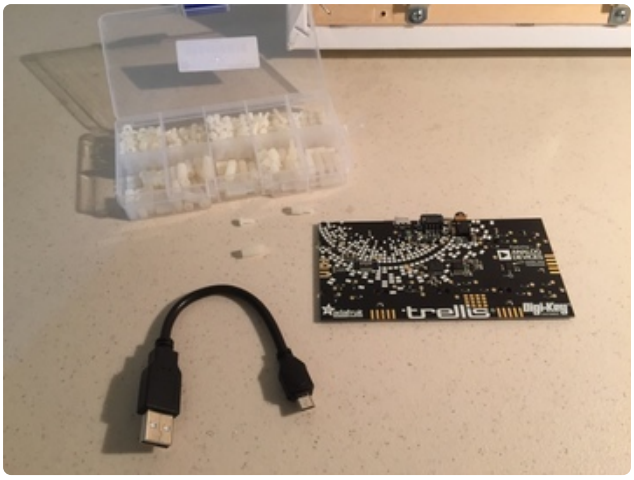
1 x NeoTrellis M4 Mainboard https://www.adafruit.com/product/3938
NeoTrellis M4 Mainboard with 32 NeoPixels in a 4x8 grid

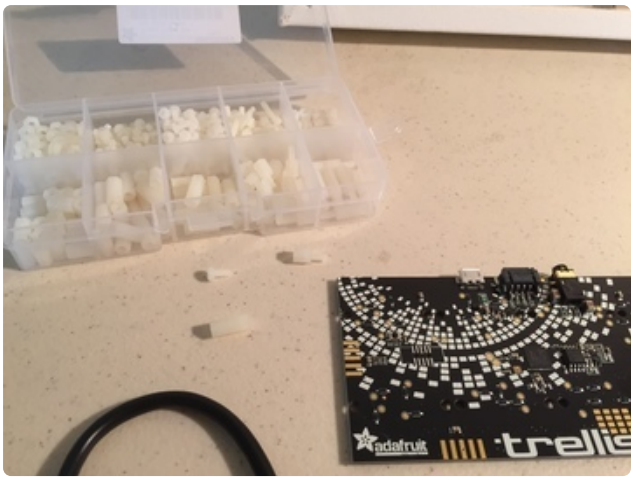
---

1 x USB cable - 6" A/MicroB https://www.adafruit.com/product/898
short (6 inch) USB cable

---

1 x USB cable - USB A to Micro-B - 3 foot long https://www.adafruit.com/product/592
long (3 foot) USB cable

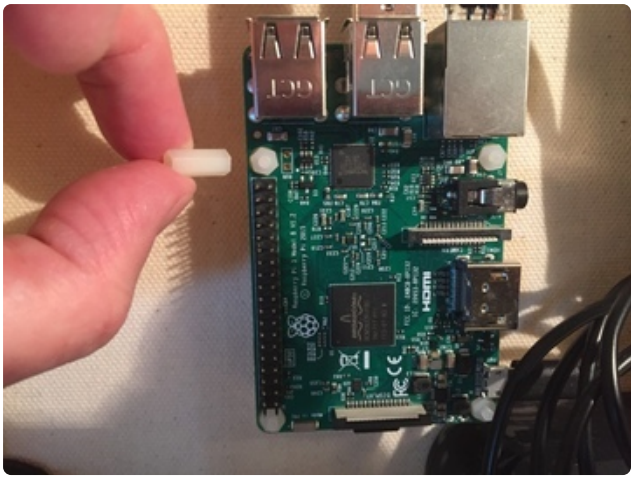


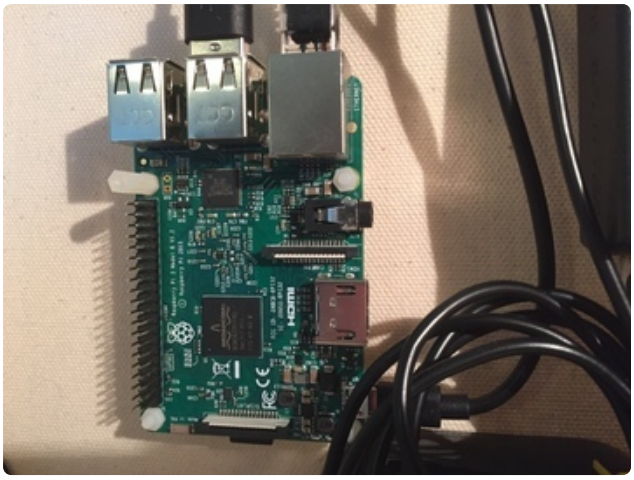


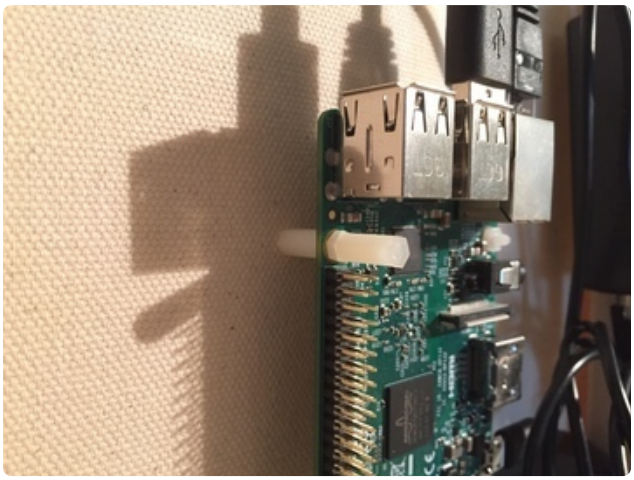
The parts list above includes 2 different USB cables. I'll first show how to use the short USB cable and some of the nylon screw parts to attach an Adafruit NeoTrellis M4 board directly to the Raspberry Pi.

Later, I'll use the longer USB cable to show how you could instead attach the NeoTrellis M4 to foam board placed behind the canvas. This configuration will allow for more variability on positioning of the NeoTrellis M4.

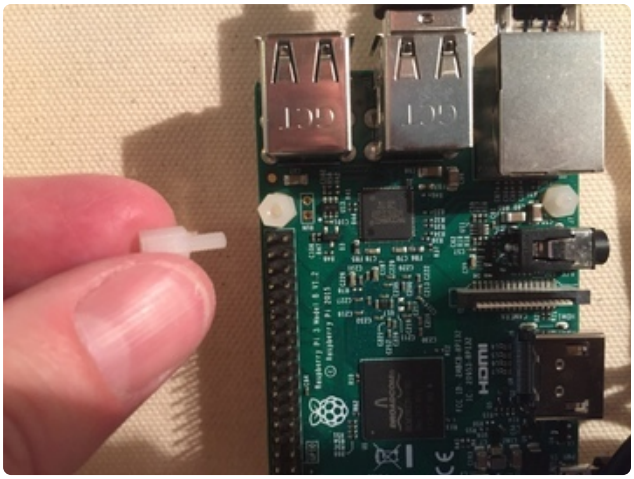
From the nylon screw kit, pull out one of the medium size screws, a shorter size standoff and the longest of the straight spacer pieces. Read on to see how these will be attached to the Raspberry Pi.

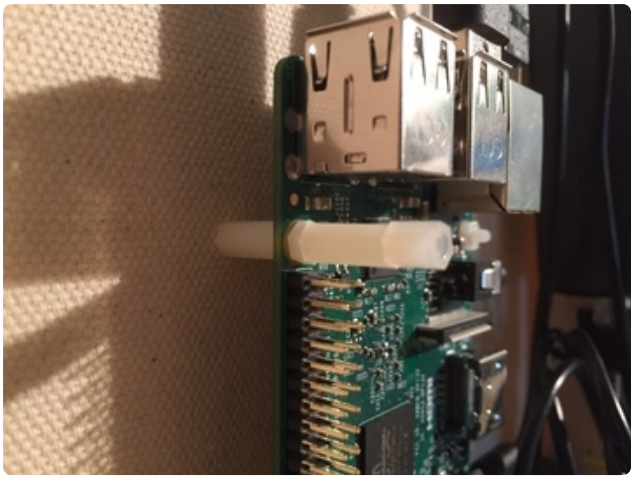




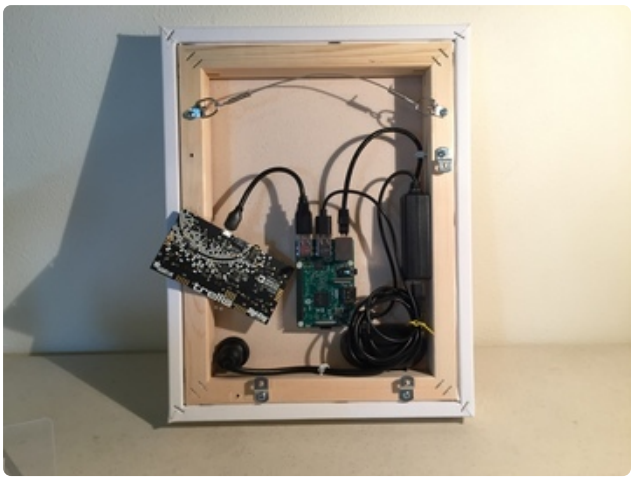


Here I am screwing the spacer part into the upper left of the 4 nylon screw assemblies holding the Raspberry Pi.



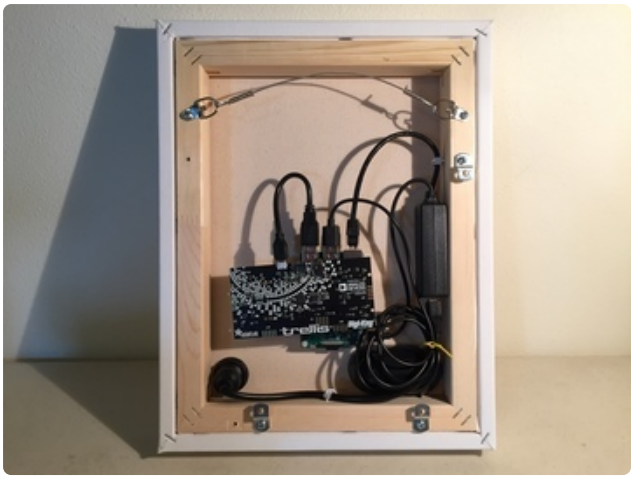


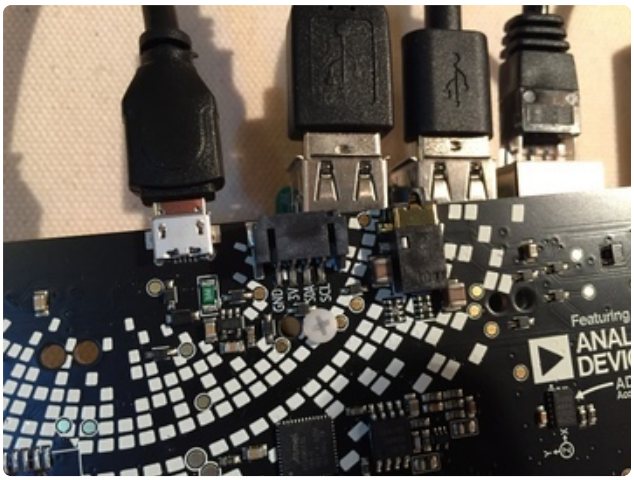
Now screw the standoff on top of the spacer as shown in these pictures.



In these pictures, you can see how I've connected the NeoTrellis M4 to the short USB cable and then plugged that into one of the USB ports in the Raspberry Pi.

At this point, the NeoTrellis M4 is hanging freely, but in the next step I'll show how I oriented it over the Raspberry Pi and then attached it with the nylon screw.





You'll probably need to massage this USB cable a bit to get these parts to hang well together.

Notice in these pictures how I've placed the nylon screw through the hole near the SDA and SCL markings on the board. But if your cable is a little longer or shorter, you may find that another mounting hole works better.

At this point we can boot up the Raspberry Pi system and use the `screen` utility to connect to the CircuitPython REPL on the NeoTrellis M4.

Usually the first connected CircuitPython device will be represented as `/dev/ttyACM0` on the Raspberry Pi, and you can connect and talk to the REPL with this shell command:

```
screen /dev/ttyACM0
```

If this REPL connection is working, you should be able to press `Enter` (or use `Control-D` to reset the board), and see a `>>>` prompt allowing Python commands to be entered. If there is a Python program running, you can use `Control-C` to stop it. To end the `screen` command session, type `Control-A` followed by the `K` and `Y` keys.

I used the adafruit_trellism4 (https://adafru.it/D2W) library to control the lights on the NeoTrellis M4 through the REPL. This required copying several libraries from the Adafruit library bundle. I downloaded the library bundle from the CircuitPython site (https://adafru.it/ENC) and unzipped it, and from the **lib** directory in there I copied 3 of the `.mpy` files to the NeoTrellis M4 drive as shown below:

```
cp adafruit_trellism4.mpy /media/pi/CIRCUITPY/lib ; sync
```

```
cp neopixel.mpy /media/pi/CIRCUITPY/lib ; sync
```

```
cp adafruit_matrixkeypad.mpy /media/pi/CIRCUITPY/lib ; sync
```

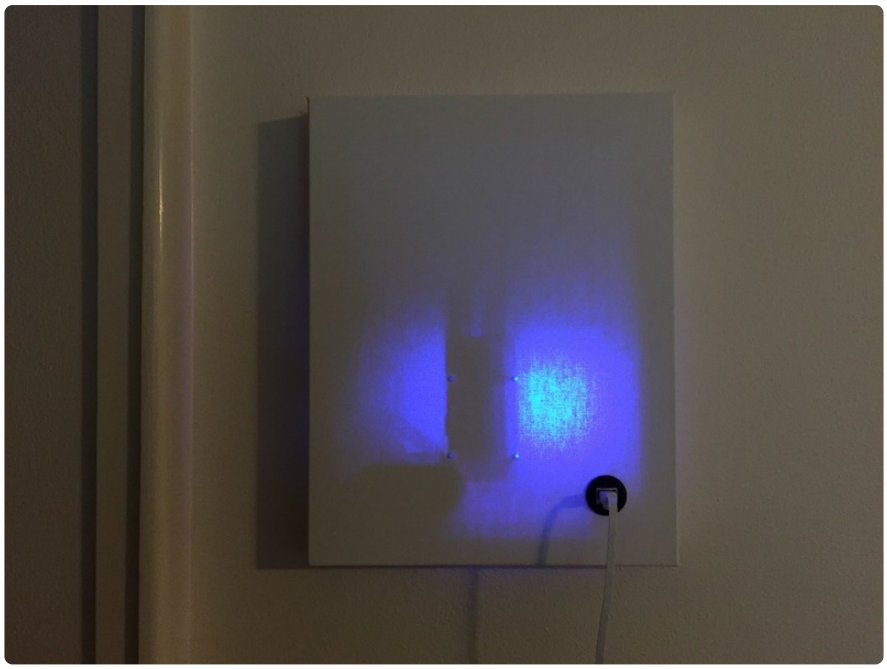
... now with that all ready, use `screen` to connect to the REPL. In the REPL, first do `Control-D` and then press `Enter` a few times and look for the `>>>` prompt. Then issue these commands:

```
import adafruit_trellism4
```

```
trellis = adafruit_trellism4.TrellisM4Express()
```

```
trellis.pixels.fill((0,0,128))
```

After doing this, my CompuCanvas looks like the picture below.



In the picture here, I have just one of the NeoPixels emitting green light and the rest are off. To get to this state, I first used this command in the REPL to turn all the NeoPixels off:

```
trellis.pixels.fill((0,0,0))
```

Then, I set a single NeoPixel like this:

```
trellis.pixels[0,0]=(0,128,0)
```

Note that this NeoTrellis M4 board has 32 NeoPixels in an 8x4 grid. So one corner can be addressed by `trellis.pixels[0,0]` and the far corner is `trellis.pixels[7,3]`.

To get the picture here, I turned the lower left NeoPixel blue with this command in the REPL:

```
trellis.pixels[0,3]=(0,0,128)
```

For this picture, I have illuminated the other 2 corner NeoPixels with the following commands:

```
trellis.pixels[7,0]=(0,0,128)
```

```
trellis.pixels[7,3]=(0,128,0)
```

I've found that a lot of interesting things happen when the lights are just behind the edges of the other parts. For these pictures, I first cleared all the NeoPixels:

```
trellis.pixels.fill((0,0,0))
```

Then I set 4 of them that seem to be aligned just behind the Raspberry Pi with the following:

```
trellis.pixels[0,0]=(0,0,128)
```

```
trellis.pixels[3,0]=(0,0,128)
```

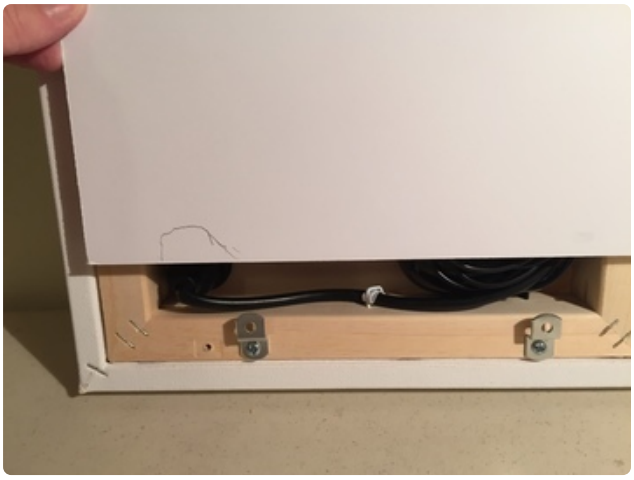
```
trellis.pixels[0,3]=(0,128,0)
```

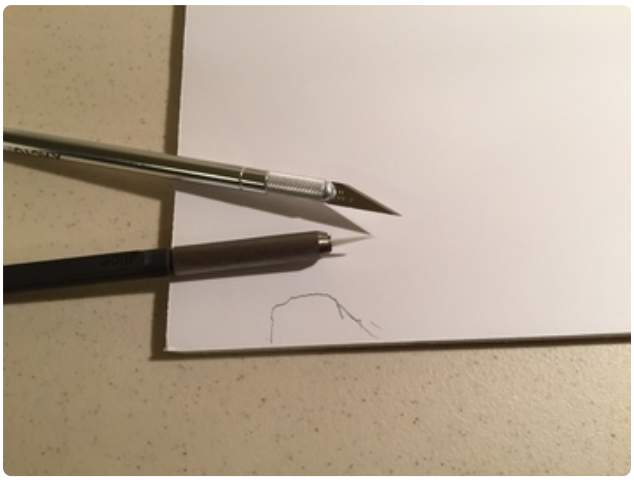
```
trellis.pixels[3,3]=(0,128,0)
```

I like the how the spikey shadows from the expansion pins are visible on the right side of the Raspberry Pi with this illumination pattern.

Ok, now let's try something completely different! What if we move the NeoTrellis M4 off of the Raspberry Pi and instead attach it to the foam board?

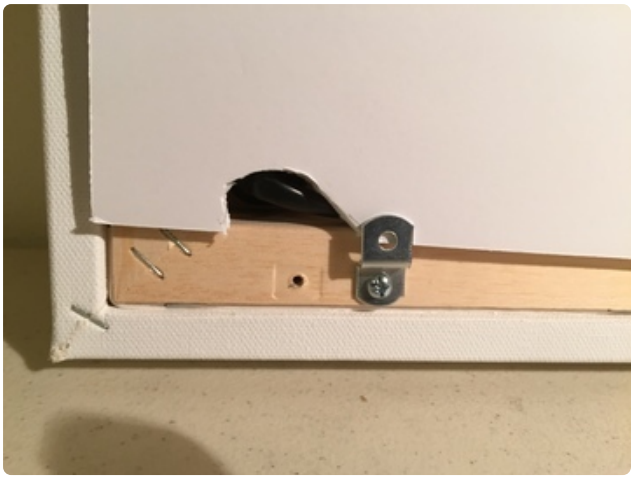
Before we get to that, I'll first go over this series of pictures that show how I fine-tuned the foam board to fit with the other parts behind the canvas.

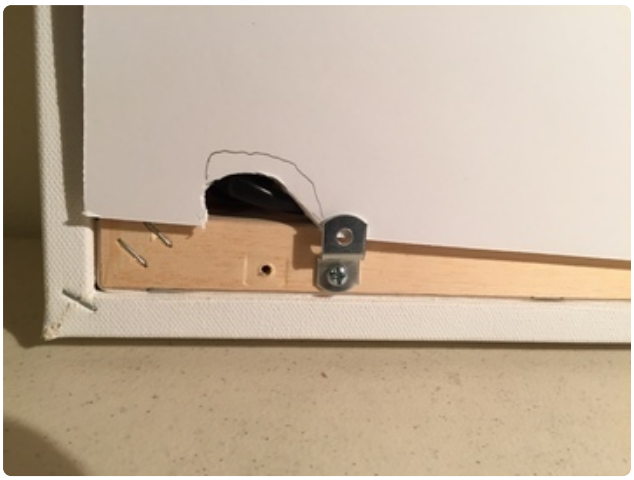


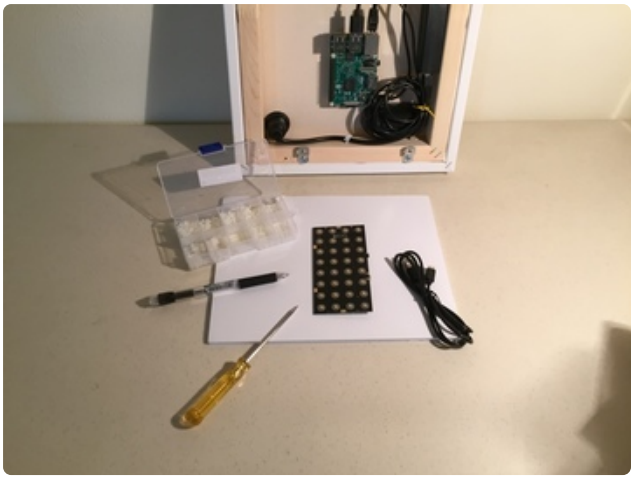


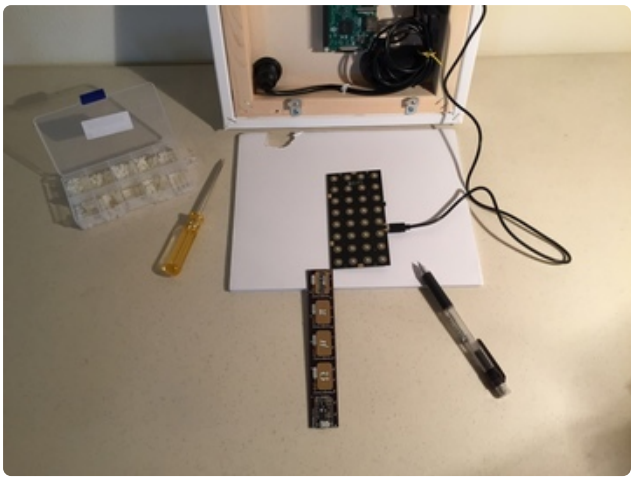
The Ethernet cable is just deep enough that it blocks the foam board from fitting in place.

The quick and easy solution to this kind of problem is to just cut away a bit of the foam board. The pictures here show how I - with a bit of trial and error - cut enough to allow the foam board to rest squarely against the offset clips attached to the bottom of the canvas.

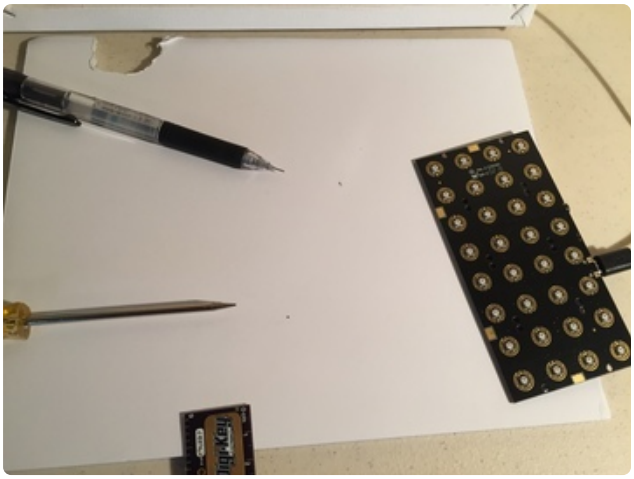


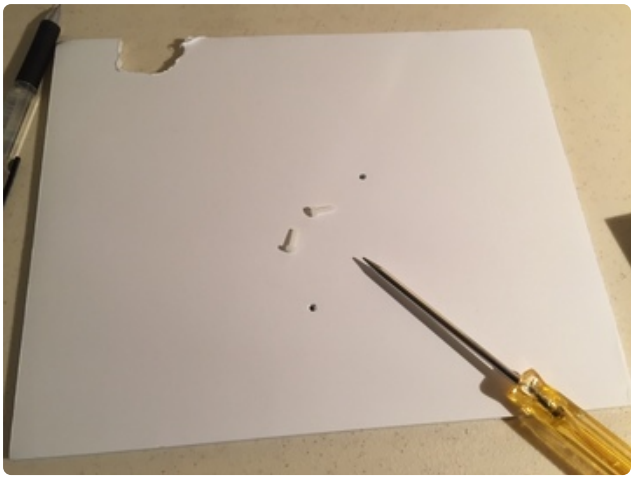


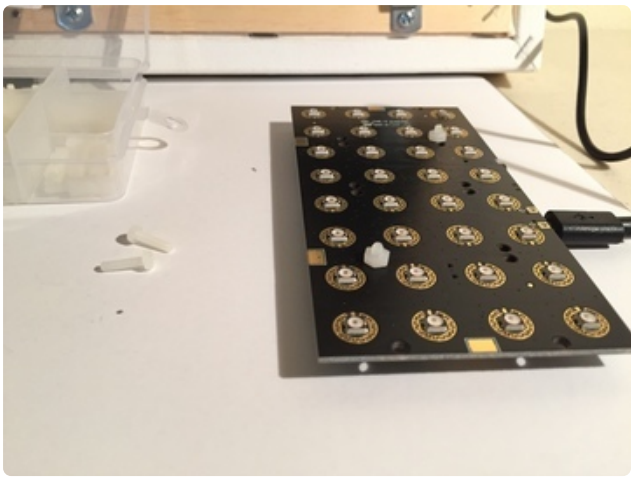


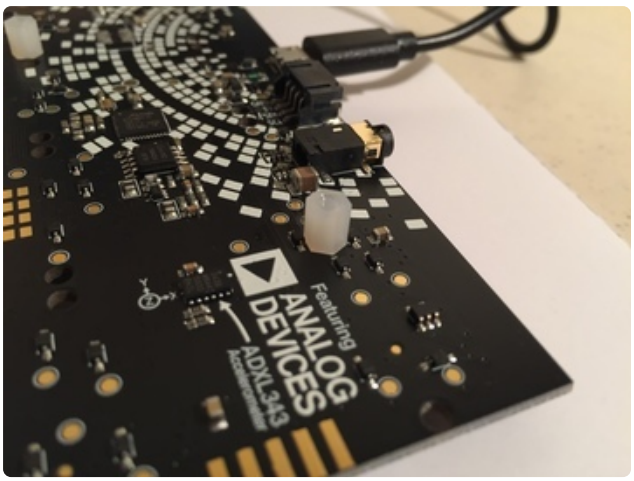


In these picture,s I'm sizing up how to position the NeoTrellis M4 on the foam board. I used the mechanical pencil to mark spots aligned with 2 of the mounting holes on the NeoTrellis M4. Then I used the awl to cut holes through these marked spots to fit the nylon screws.



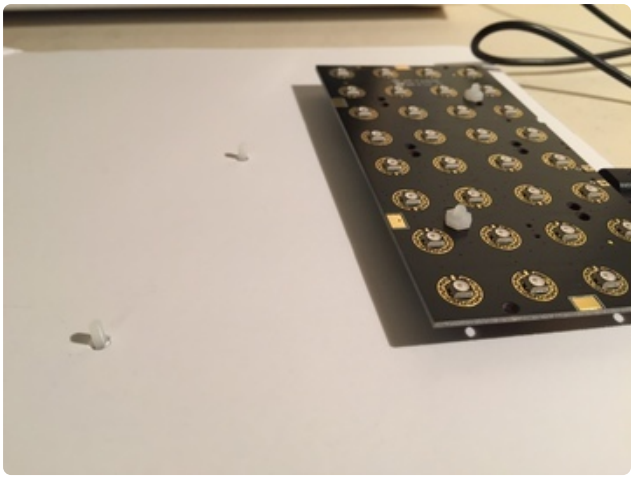


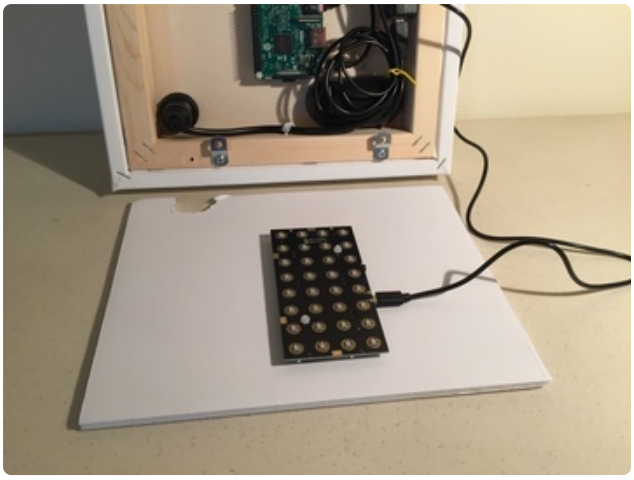




From the nylon screw set, pull out 2 of the hex nuts, 2 of the longest screws and 2 of the shorter standoffs. The pictures here show how I attached the standoffs at the points I marked on the foam board in the previous steps.

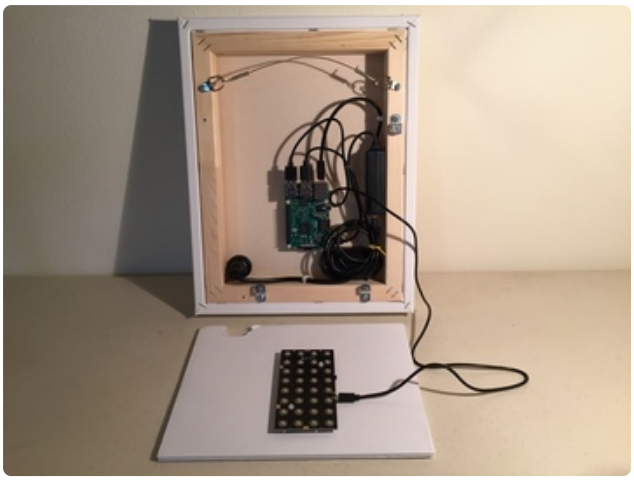
Note that the last picture shows how the standoffs can end up very close to some of the electronic parts. Be careful not to twist the standoffs too much against these surface mounted parts!

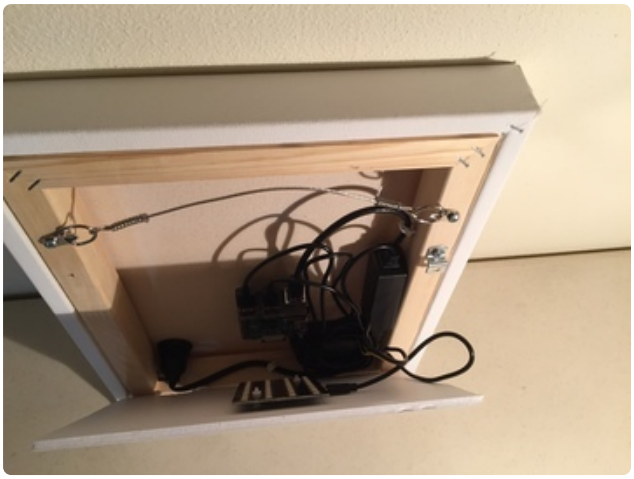


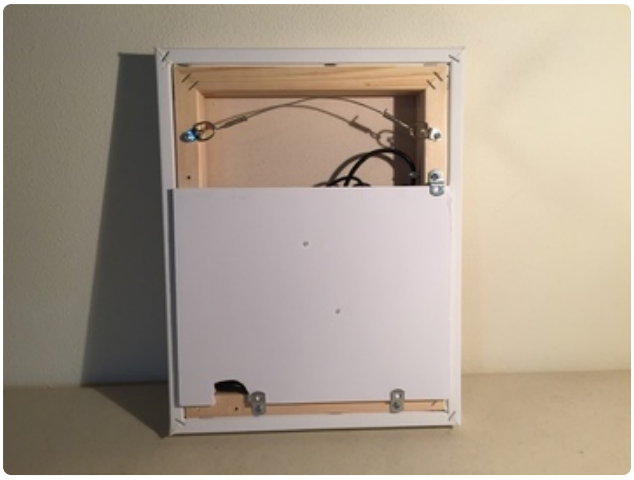


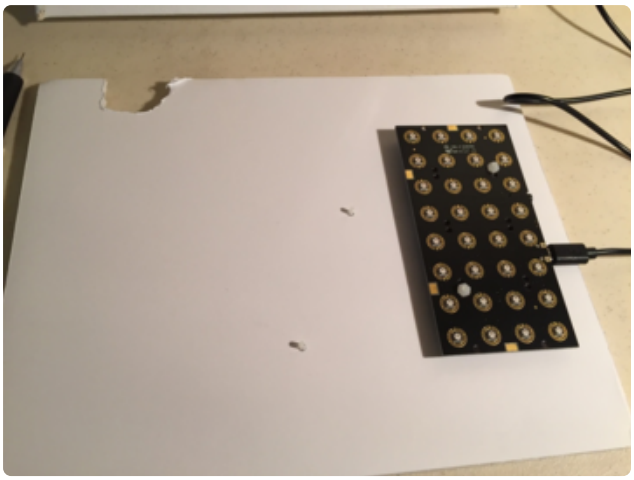
In these pictures I threaded the nylon screws through the holes in the foam board. Then I mounted the NeoTrellis M4 on the foam board as shown.

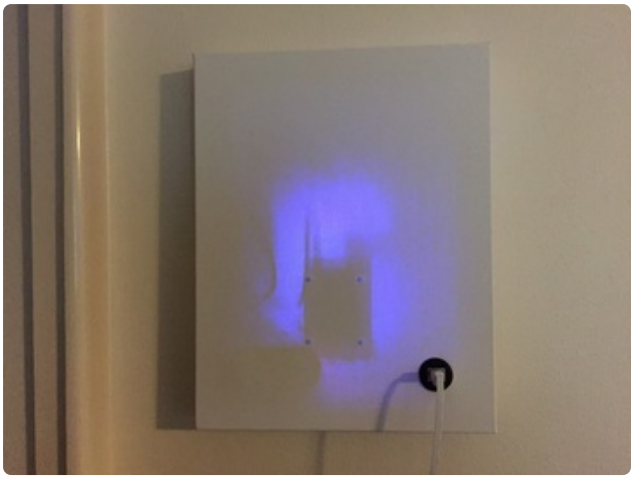
Now the foam board can be fit into the offset clips behind the canvas. It's a snug fit, but everything has a place.











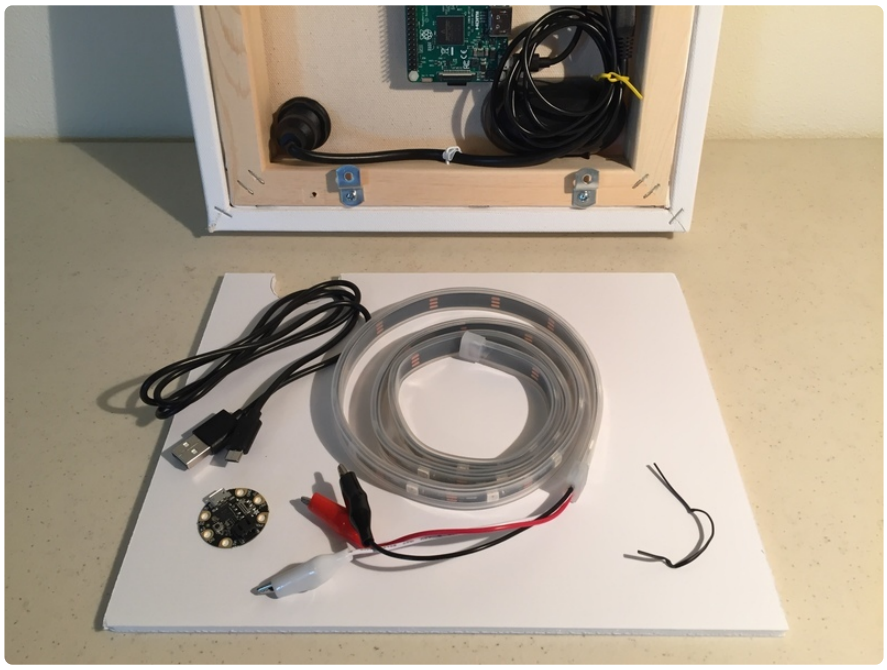
In these pictures I am testing the look of this new positioning of the NeoTrellis M4. As in the earlier examples above, I am using the `screen` command to connect to the REPL and issuing commands like shown below to light all the NeoPixels with one color:

```
import adafruit_trellism4

trellis = adafruit_trellism4.TrellisM4Express()

trellis.pixels.fill((0,128,0))
```

Here are some more lighting tests with the new NeoTrellis M4 positioning. In this orientation the NeoPixel at coordinate `[0,0]` is on the lower left, and `[7,3]` is in the upper right corner.

Continue to the next page to see an alternative approach to placing light behind the canvas.

Alternate Internal Lighting



In this section I'll show a different way of illuminating the back of the canvas using the parts shown above, which are also listed just below. The primary part is a NeoPixel strip with 30 RGB LEDs. This will be controlled by a Gemma M0. A USB cable is needed to connect the Gemma M0 to the Raspberry Pi. Finally, another piece of the 8 inch by 10 inch foam board will be used to secure these parts behind the canvas.

Before getting into the weeds of this section, I want to stress a few points about this approach to interior lighting:

- In the previous section, the NeoTrellis M4 had 32 NeoPixels directed straight out from behind the canvas. The arrangement here will have 30 NeoPixels directed sideways out from the center of the canvas. To help visualize this, refer to the picture above where the small white squares on the NeoPixel strip are the actual NeoPixels which will be aligned parallel to the foam board. Overall this should fill the interior of the canvas with lots of light and color, but the effect will be more diffuse than with the NeoPixels facing the canvas.
- The parts used here did not play well with the exterior lighting parts used in the next section. The main issue is of space behind the canvas. The NeoPixel strip used in this section will interfere with the USB panel mount cables used in the next section.
- There are many different NeoPixel strips to choose from. Another one that I have used before is this 20 NeoPixel strand (http://adafru.it/3630). Using this would allow for facing the LEDs straight out from behind the canvas and also allow for more positioning options than the part I chose. You would need a different way of attaching to the Gemma M0 - this "bolt-on" kit (http://adafru.it/4103) is what I would try first.

1 x Adafruit NeoPixel LED Strip w/ Alligator Clips https://www.adafruit.com/product/3812
NeoPixel strip for canvas interior

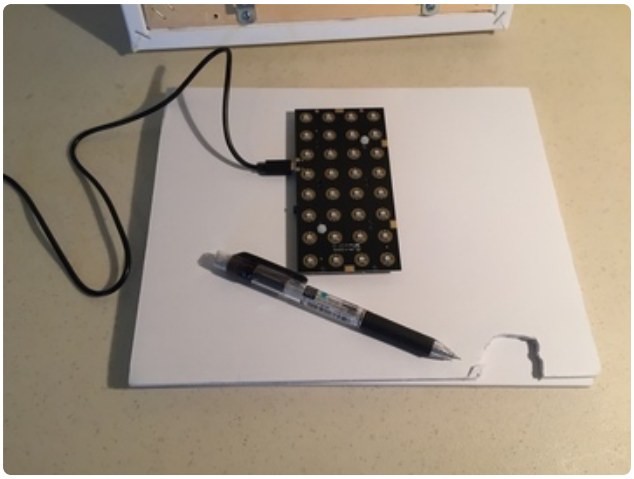
---

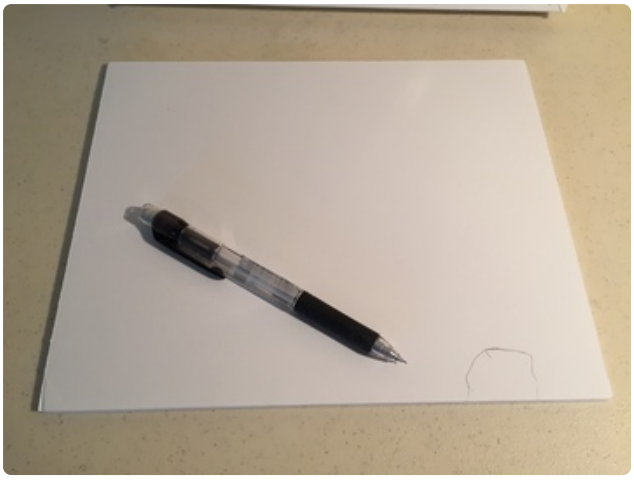
1 x Gemma M0 https://www.adafruit.com/product/3501
GemmaM0 to control NeoPixel strip

---

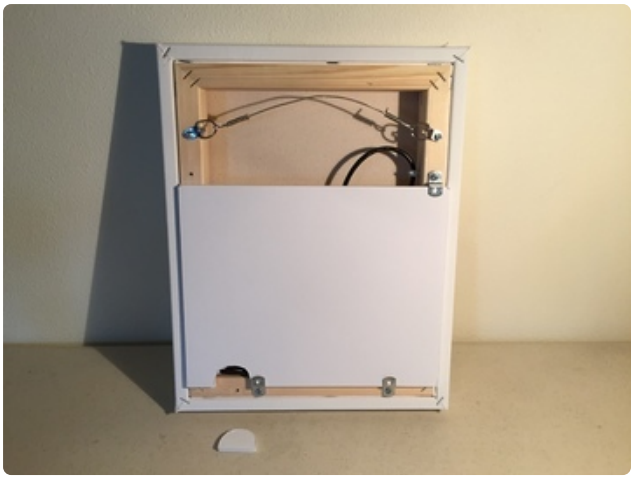
1 x USB cable - USB A to Micro-B - 3 foot long https://www.adafruit.com/product/592
USB cable to connect GemmaM0 to rPi

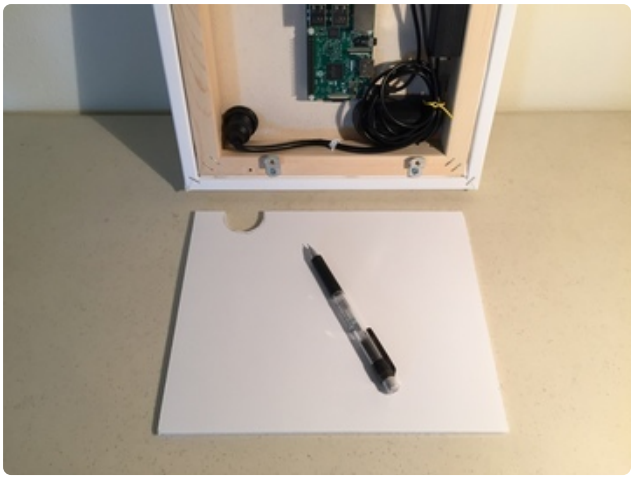
In these pictures, I'm trying to show that the NeoPixel strip will occupy a bit more depth behind the canvas than the NeoTrellis M4 does. When arranged on the foam board, it needs to avoid obstructions like the PoE adapter, the speaker, and the USB and Ethernet ports on the Raspberry Pi.

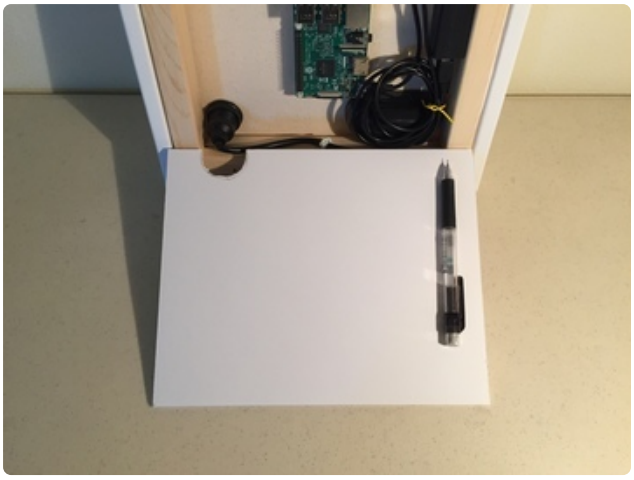




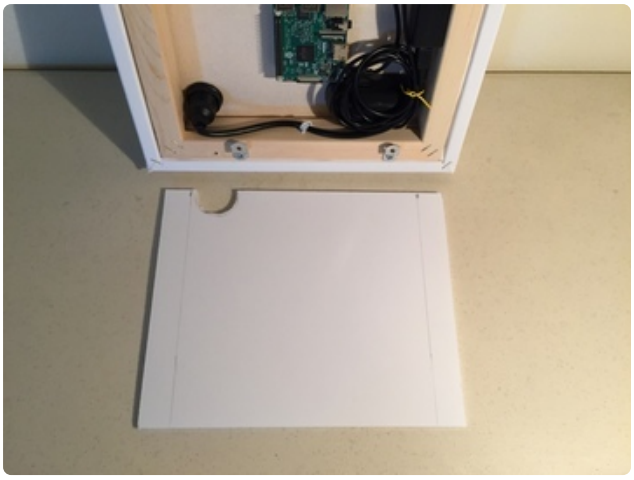
These pictures show how I use the foam board setup from the last section to trace out where to cut the new foam board for this section. The previous cut was a bit gnarly so I rounded out my trace and then made the cut and checked that the foam board does indeed fit behind the canvas.

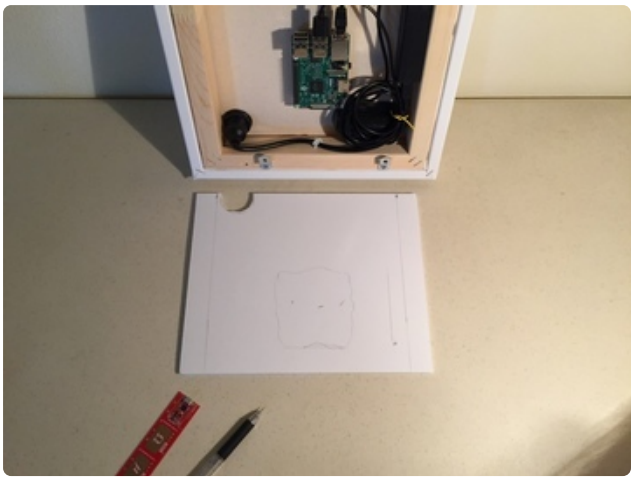


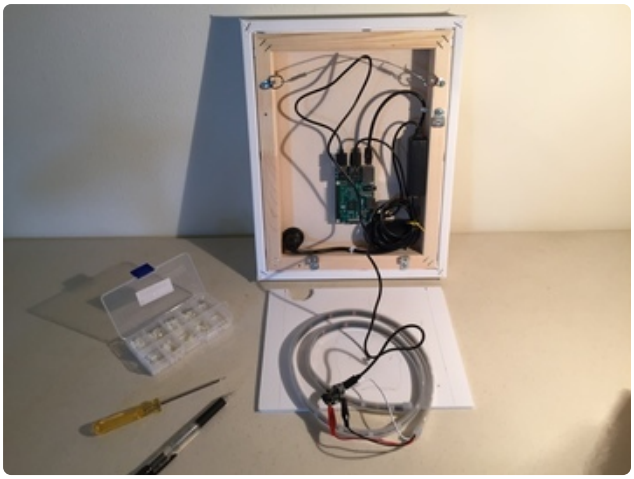


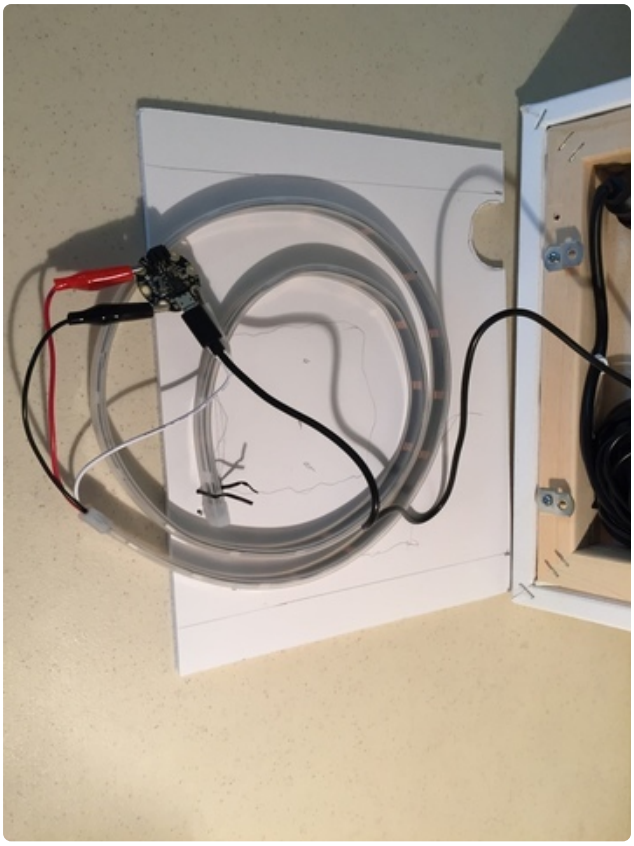


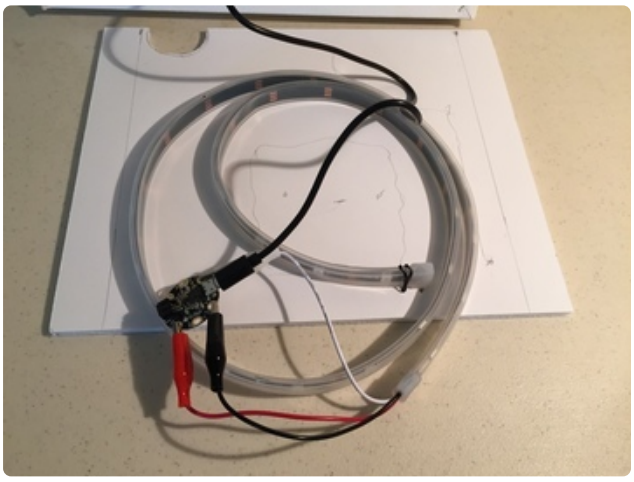
This series of pictures shows how I've tried to map out the canvas interior to identify where the NeoPixel strip can be placed. There is quite a bit of eye-balling and fuzzy logic to this process....







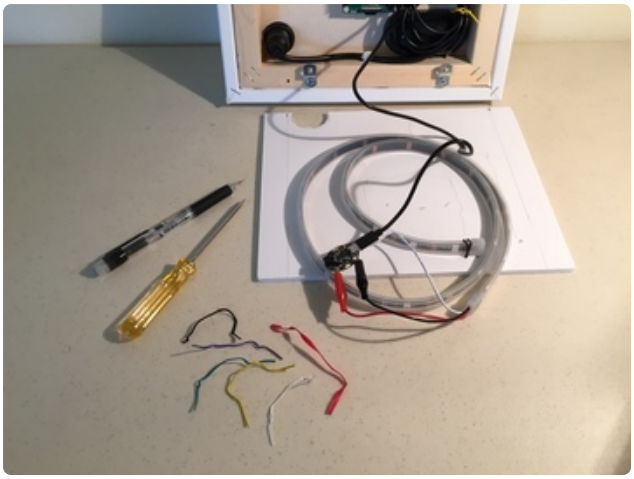


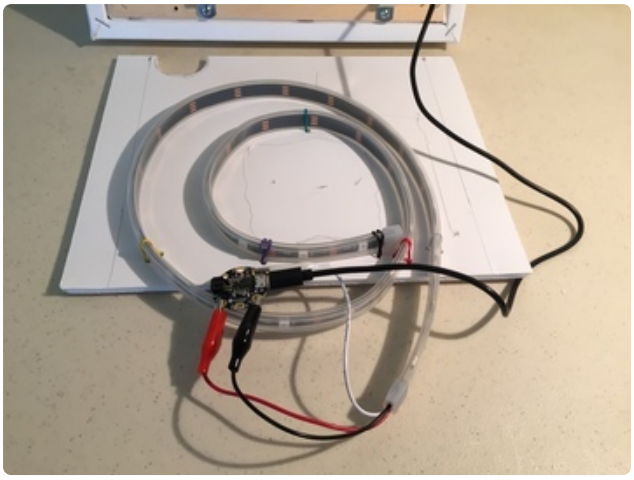


Here I'm starting to attach the NeoPixel strip to the foam board. I'm going to use the twist-tie that came with the USB cable. You could also use bits of wire or small zip ties for this.

Note how I'm using the awl to poke holes in the foam board and then threading the twist-tie through the foam board and around end of the NeoPixel strip.

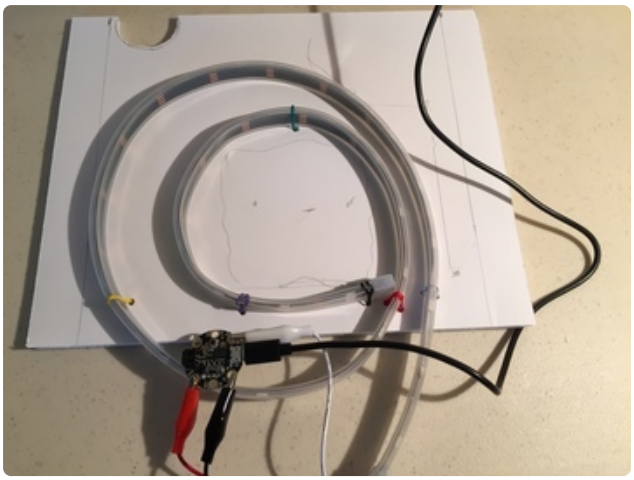
As I go forward here, I'm trying to keep the NeoPixel strip inside the lines I've marked on the foam board. In some cases my initial mount points didn't work, so I poked more holes and re-positioned the NeoPixel strip. The foam board can withstand a lot of damage and is relatively inexpensive, so buy a few extra pieces and don't fear about experimenting with alternate configurations.

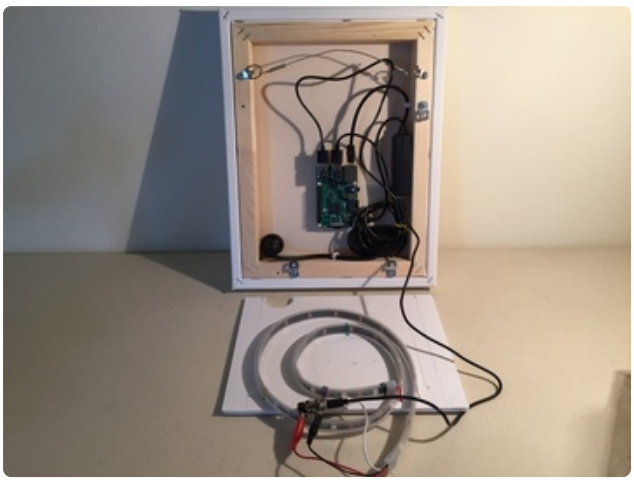


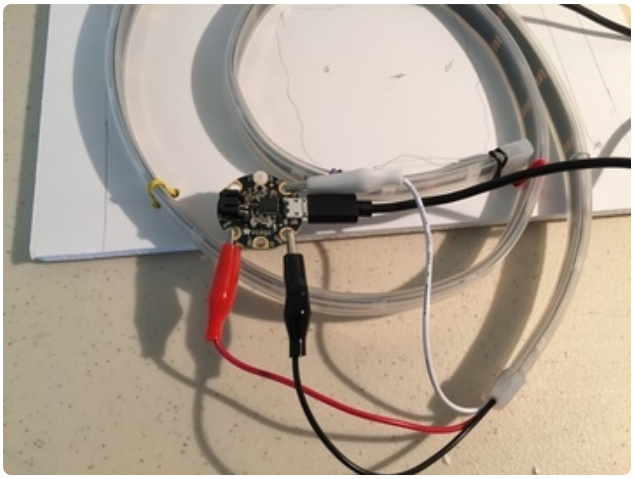


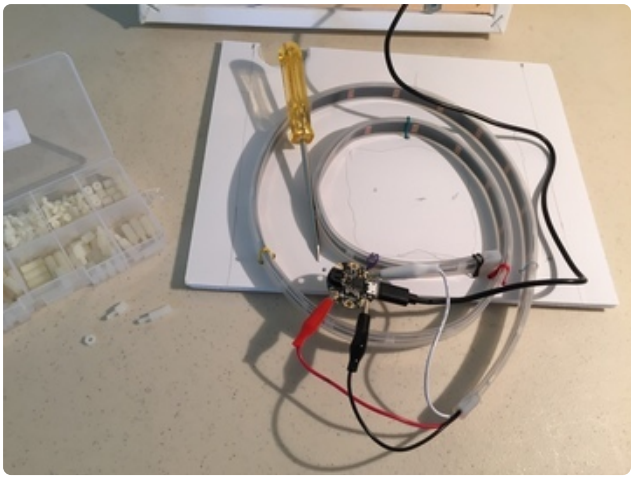
I actually have a collection of these twist-ties, saved from the purchase of many cables and other assorted things. Here I've selected a range of colors and I'll place them at different points to hold the NeoPixel strip to the foam board while avoiding obstructions of parts attached to the canvas.

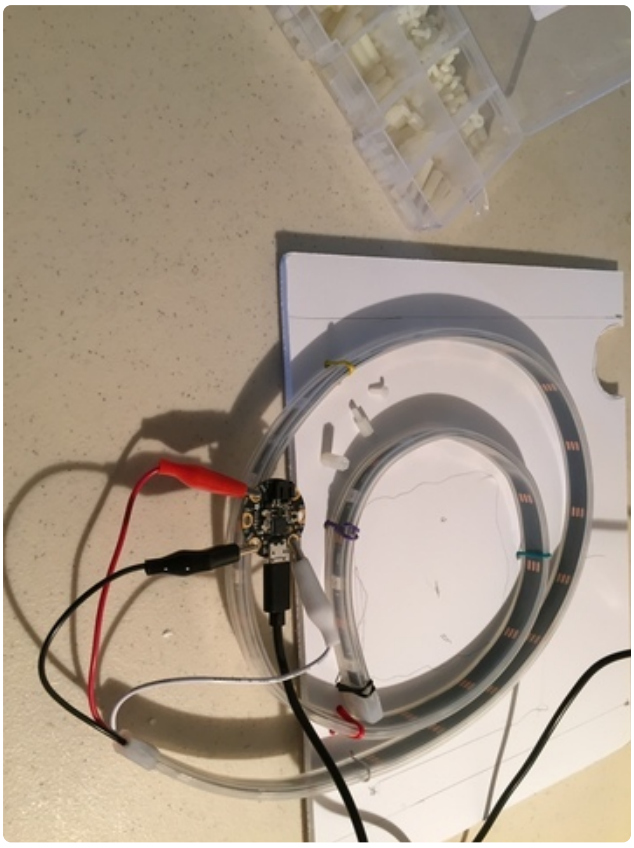
Bits of stiff wire can also be used.







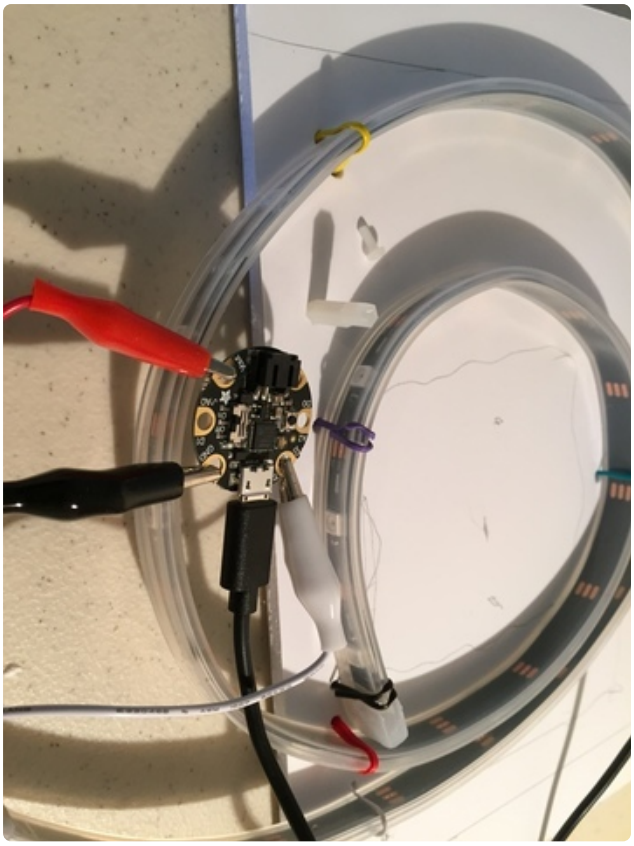


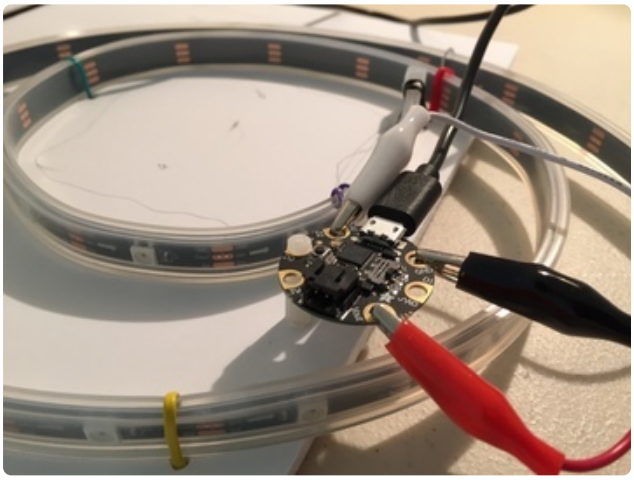


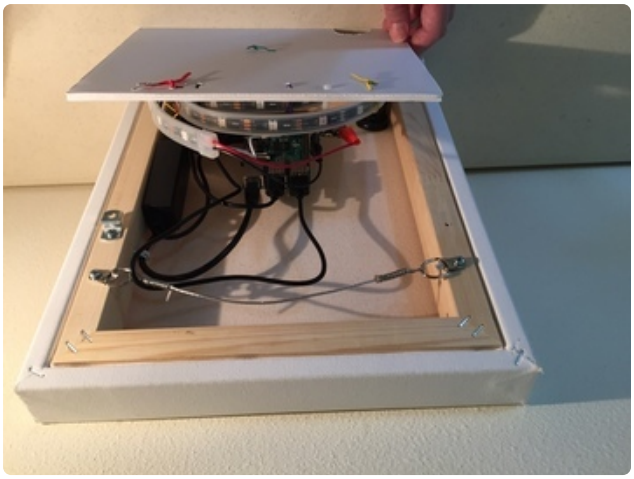
The first picture here shows how I ultimately connected the Gemma M0 to the foam board and the NeoPixel strip. The additional pictures show how I got to this point, measuring, marking and using the nylon screw parts to mount the Gemma M0 on the foam board.

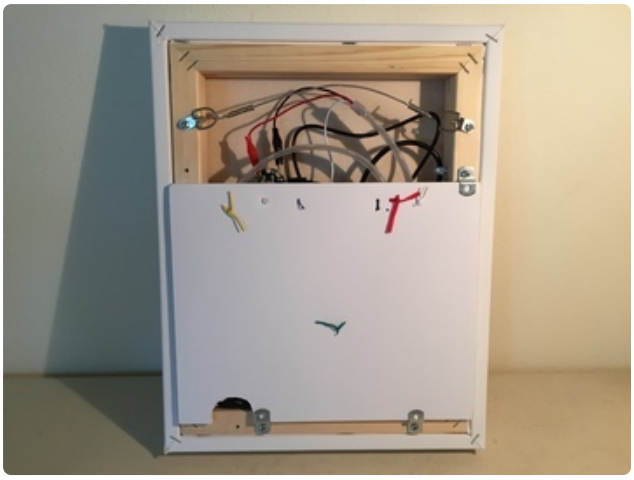
Note that the Gemma M0 is connected to the NeoPixel strip alligator clips as follows:

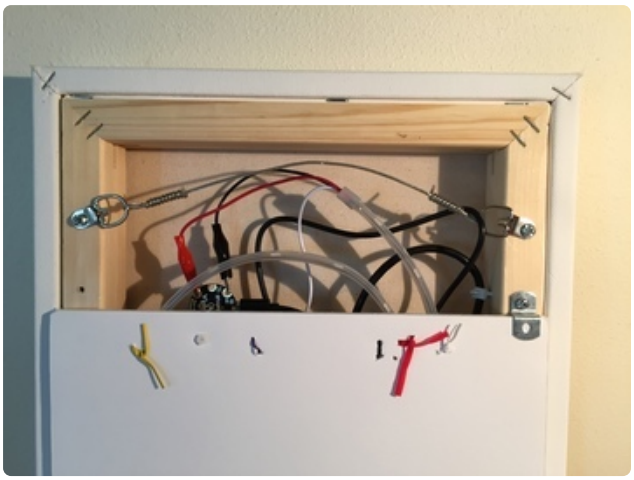
Red alligator clip to **Vout**
Black to **GND**
White to **A1/D2**
See also this NeoPixel page from the Gemma M0 guide (https://adafru.it/HAg) which shows a wiring diagram and has a CircuitPython code example.



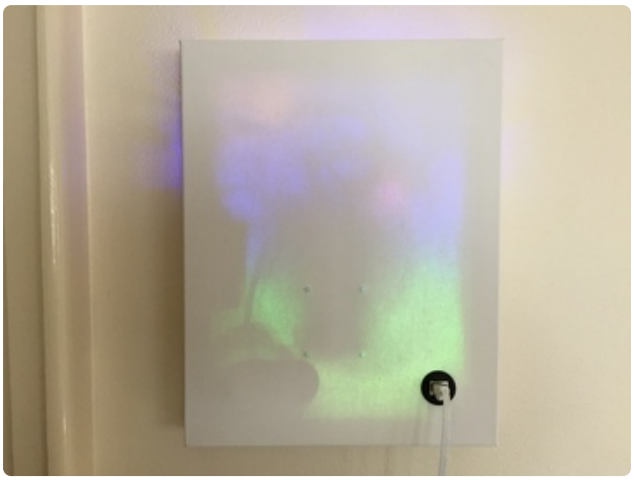








Here I am fitting this interior lighting setup onto the back of the canvas.



In these pictures I've booted up the system and am testing the lights. In the first picture, the green light is the NeoPixel at index 0. The rest of the NeoPixels on the strip are blue. The pinkish light is the DotStar on the Gemma M0.

To control these lights through the CircuitPython REPL, you will need code something like this:

```
import board, neopixel
pixels =
neopixel.NeoPixel(board.A1, 30)
pixels.fill((0,0,200))
pixels[0]=(0,200,0)
```

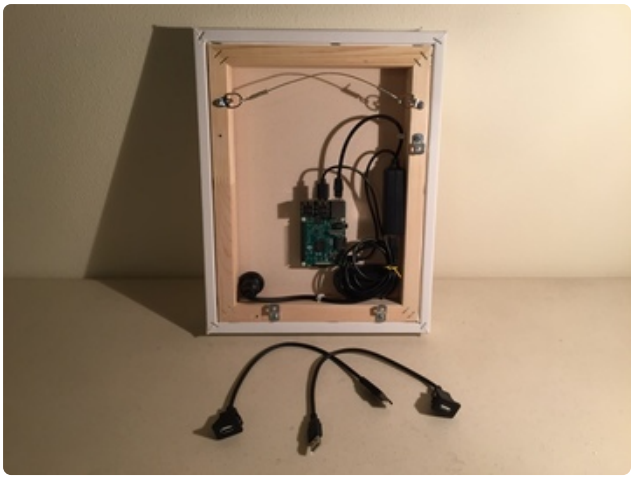
Due to the orientation of the NeoPixels, this is a more subdued lighting style than shown in the last section, but it covers more of the canvas interior.

External Features

In the previous sections, I've placed parts inside the canvas to create lighting effects emanating from the interior. In my experience, the most interesting lighting effects come from placing lights on both sides of the canvas, with interior lights directed out and exterior lights pointing back onto the canvas.

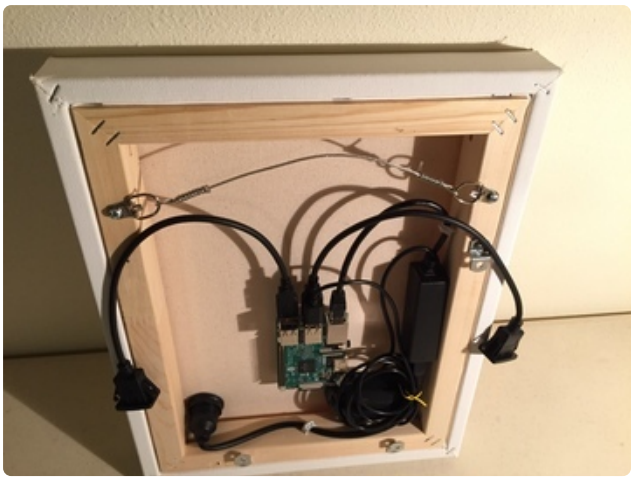
In this section I am going to show some ways of transitioning from behind the canvas to the exterior such that light may be directed at the front side of the canvas.

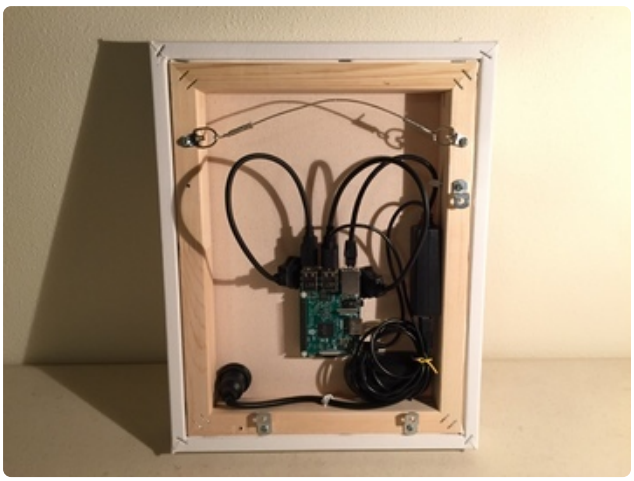
I want to stress here that I've left the front of the canvas plain up to this point in the guide because you may arrive at better, cooler, more stylish exterior designs than I can imagine. As I proceed, my general design idea is to make it look like flowering plants are sprouting out of the canvas. You may want to read ahead and then consider whether you want to follow me exactly or branch out with a different set of parts and designs for the front of your canvas.



The key part I've chosen for transitioning from the back to the front of the canvas are the USB panel mount cables shown here (and in the parts list just below).

In the pictures here, I am sizing up roughly how the cables will sit on the back of the canvas.





I'll add these cables one at a time. After seeing my results, you may choose to do only one or the other, or to find some other arrangement and set of parts.

What follows next is the parts list and process for adding the first external feature: a Circuit Playground Express. In the next section, I will show how to mount a Gemma M0 as a second external feature.

1 x Circuit Playground Express https://www.adafruit.com/product/3333
CPX with 10 NeoPixels to illuminate the canvas front

---

1 x Snap-In Panel Mount Cable - USB A Extension Cable https://www.adafruit.com/product/4055
USB cable which allows for clean transition from behind canvas

---

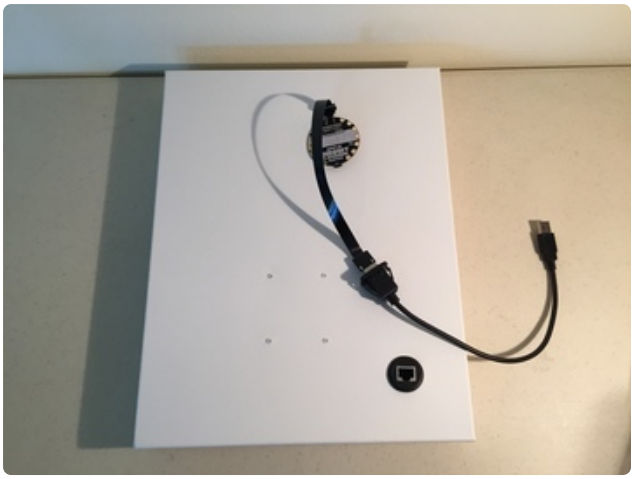
1 x DIY USB Cable Parts - Straight Type A Plug https://www.adafruit.com/product/4109
USB A end of ribbon cable assembly

---

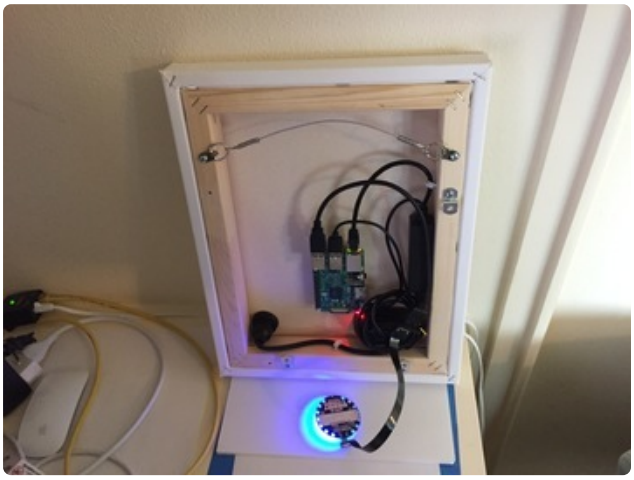
1 x DIY USB or HDMI Cable Parts - 20 cm Ribbon Cable https://www.adafruit.com/product/3561
20 cm ribbon cable

---

1 x DIY USB Cable Parts - Right Angle Micro B Plug Up https://www.adafruit.com/product/4104
Micro USB end of cable assembly; connects to CPX

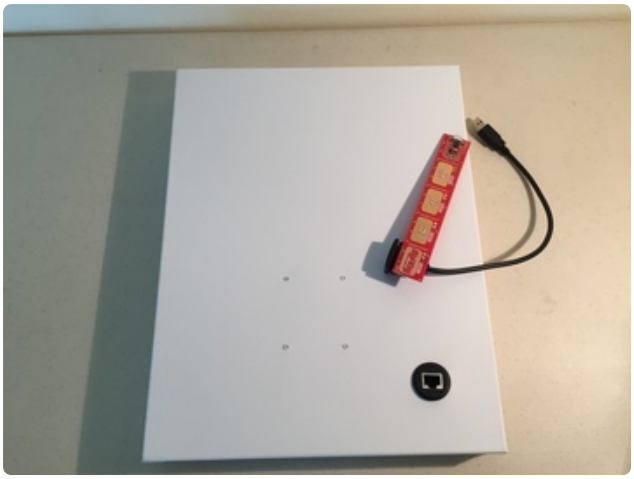


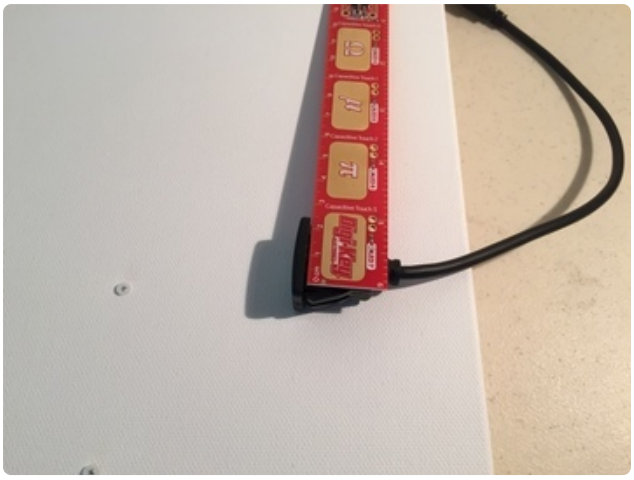
These pictures show roughly how the parts will be arranged from the front. In the second picture, the panel mount USB cable is oriented horizontally, which makes the ribbon cable flat. As you'll see in upcoming pictures, I chose instead to orient the panel mount USB cable vertically, which gives the ribbon cable a twist.

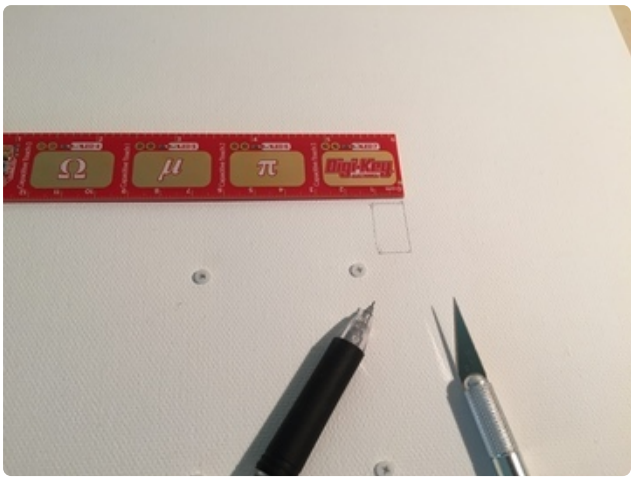


In this picture I have the CompuCanvas running, but turned around and resting against the wall. I have the exact cable assembly I plan to use and I've confirmed that I can access the CircuitPython REPL and control the NeoPixels.

This type of test is advisable before committing these parts to the canvas. In fact, the first cable assembly I tried did not work. The ribbon cables can be tricky to assemble! I ended up swapping a few parts (I was originally trying to use a 10 cm ribbon cable) and then was able to confirm the configuration shown here worked.

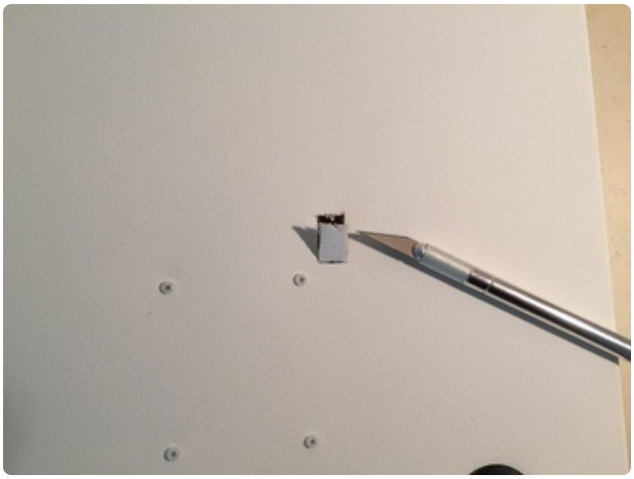


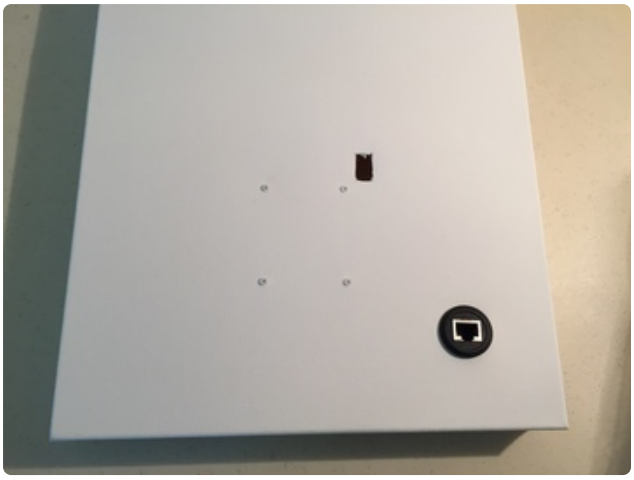




In these pictures I am measuring and marking a rectangle where I will cut a hole for the USB cable. Some things to highlight about this process:

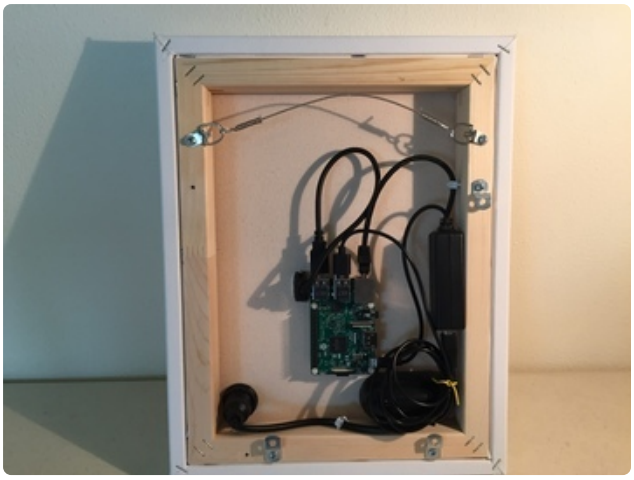
Using the nylon screws for orientation, I carefully examined both sides of the canvas before selecting a spot. It is crucial to make sure the USB cable will fit behind the spot you cut! I wanted the cable to be close to the rPi, but not too close.
The rectangle I trace here is about 1.1 cm by 1.9 cm. Having cut several of these holes now, these numbers seem just about right. If the hole is too small, the canvas could rip when placing it. With this 1.1x1.9 cm hole, the canvas should stretch a little around the cable and be a snug fit.





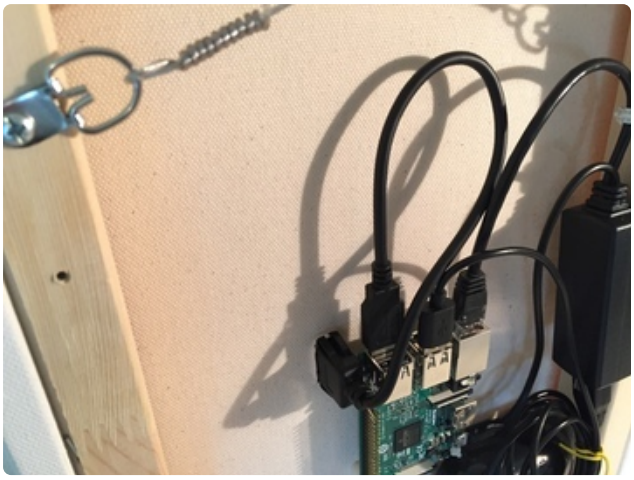
In these pictures, I show cutting the rectangular hole in the canvas and the beginning to place the USB cable.

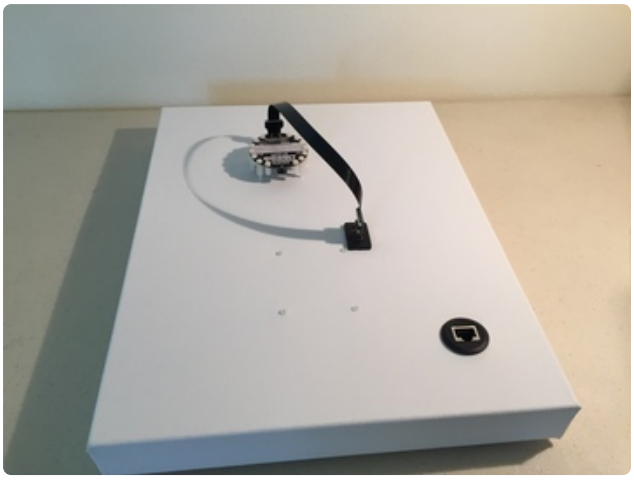
One thing to double-check is the orientation of the USB cable. It could be flipped in one of 2 directions. The last picture shows how I plugged in the Circuit Playground Express to make sure this orientation of the USB cable works.



In these pictures, I have pulled the USB cable into place. The sides of the USB cable have plastic prongs that bend a little as the cable slides in. I try to push the prongs in as I pull the cable through and then wiggle the cable a bit to nest it in place.

In the back, notice how the cable needs to be massaged away from where the foam board (or wall) will be.





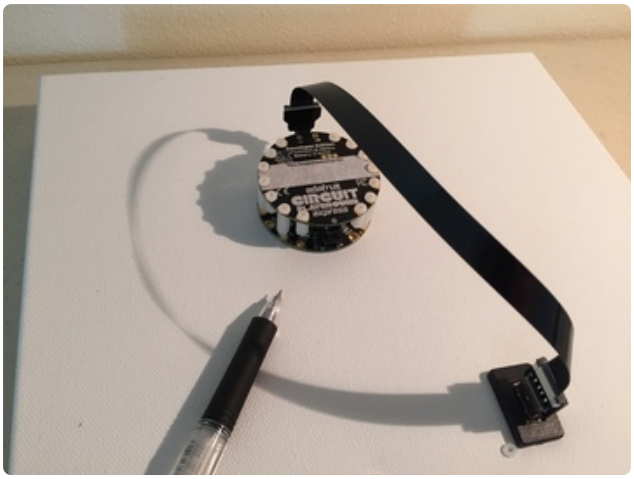
These pictures show how I marked 6 spots on the canvas where I will poke holes to attach the Circuit Playground Express (CPX).

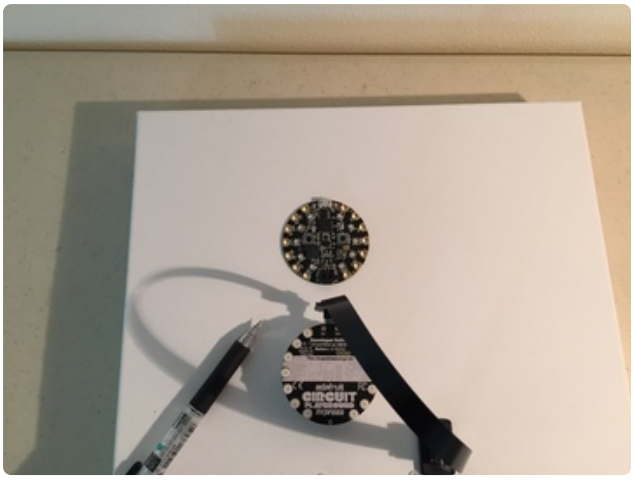
You may notice a lot of the nylon screw parts attached to the CPX in these pictures. This arrangement will be explained just below.

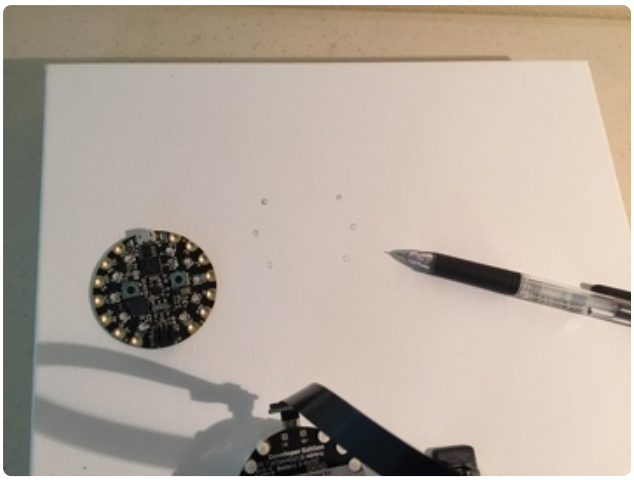
You may also notice that I 'cheated' when marking these spots by using a second CPX. If you have a second CPX, you can skip ahead and attach the nylon screw parts and then mark things the way I did. If you just have the one CPX, try to find and mark a place where it will hang well before attaching the nylon screws.

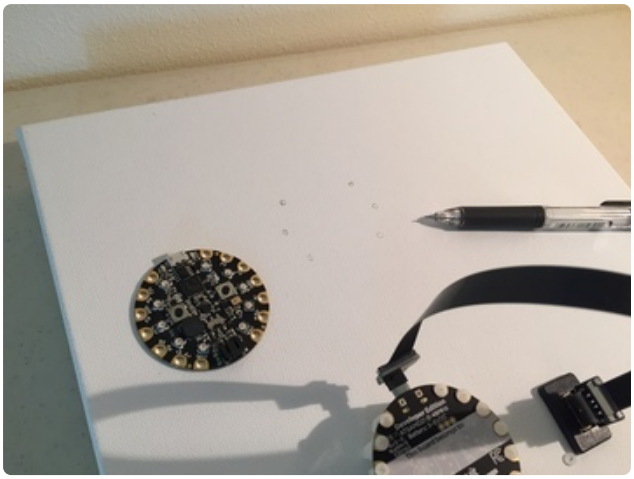
Either way, if you are following my lead here you should mark these 6 spots:

On one side: SCL/A4, 3.3V and TX/A7
On the other side: A3, GND and A0





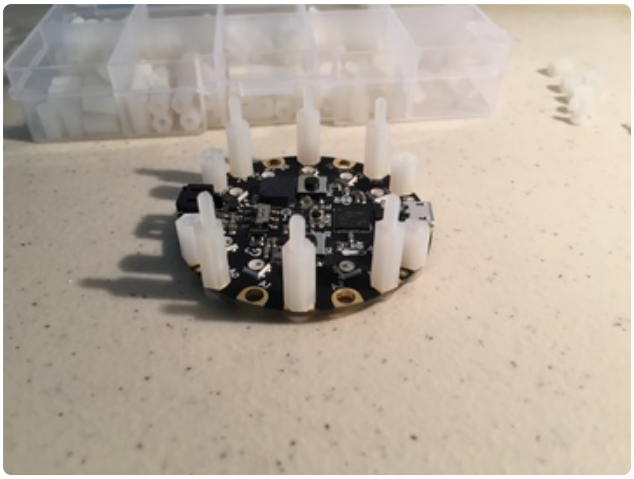


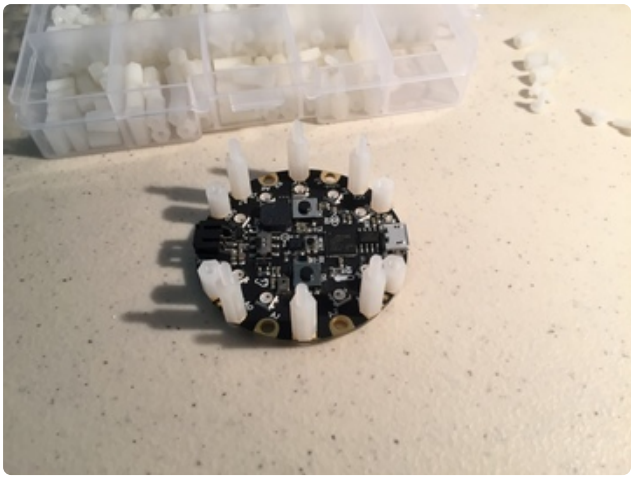


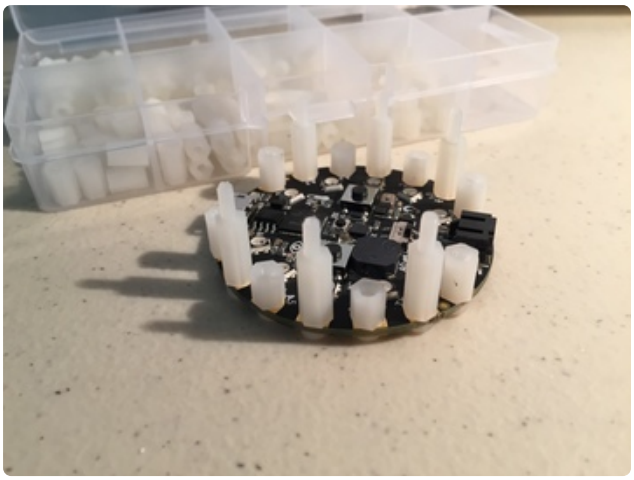
In these pictures, I'm starting to attach a bunch of the nylon screw parts to the Circuit Playground Express.

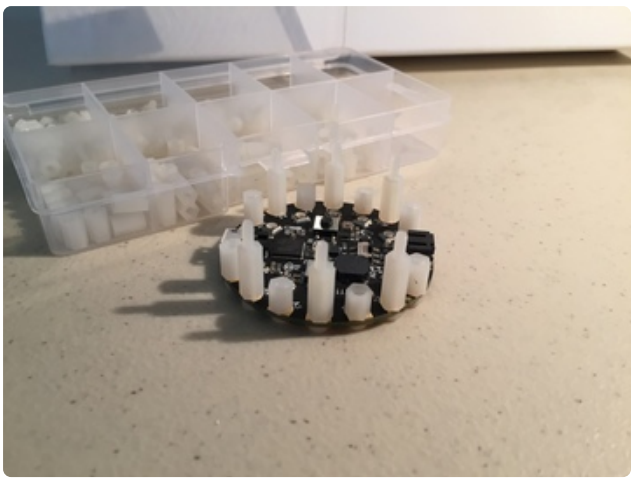
First, I attached 6 of the longest standoffs at the points shown. These 6 standoffs will be used to mount the CPX to the canvas, so they correspond to the 6 points marked in the previous step.

I actually ended up using the last 6 of the longest standoffs from the kit for this (I had to reclaim one from the Gemma M0 used in the alternate interior lighting setup). If you run out of long standoffs, 2 of the shorter standoffs are about the same length.





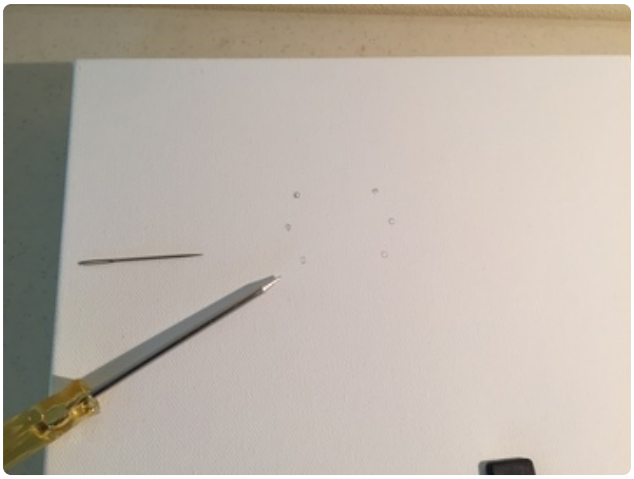


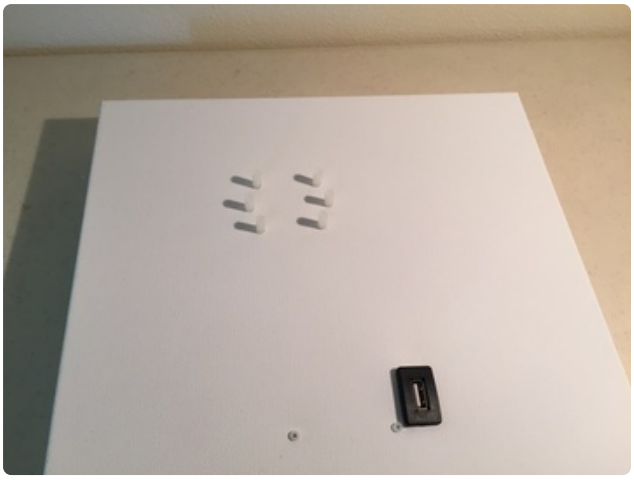


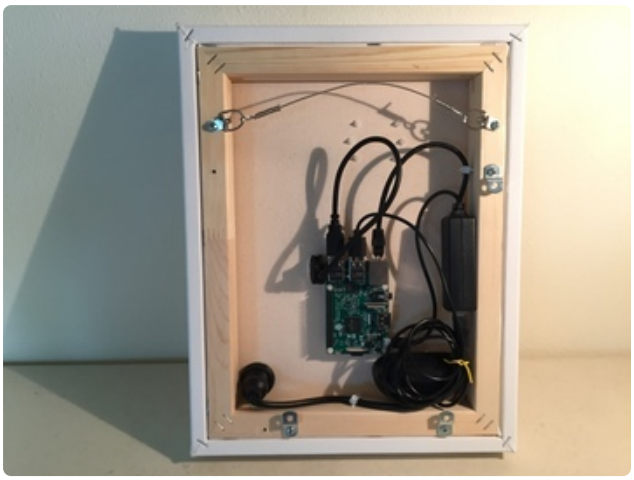
These pictures show how I also attached some of the nylon spacer parts to the CPX.

These are not for attaching to the canvas. Instead they should enhance the "light-show" by both capturing some of the light (and glowing) and also blocking some light, creating shadows on the canvas.

There are lots of variations that could be explored using the nylon screw parts to tune the NeoPixel light from the CPX.

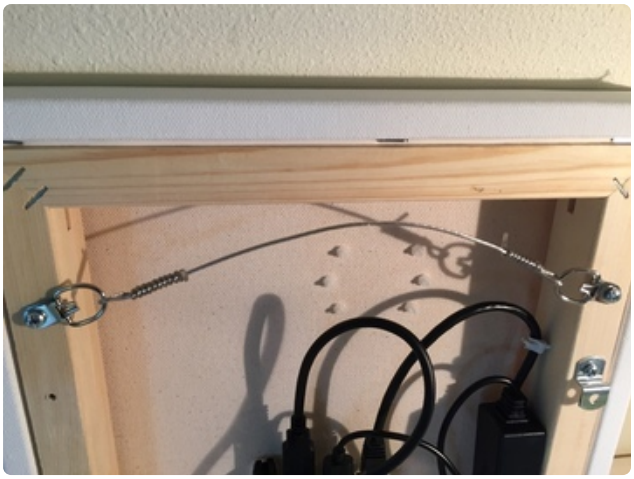






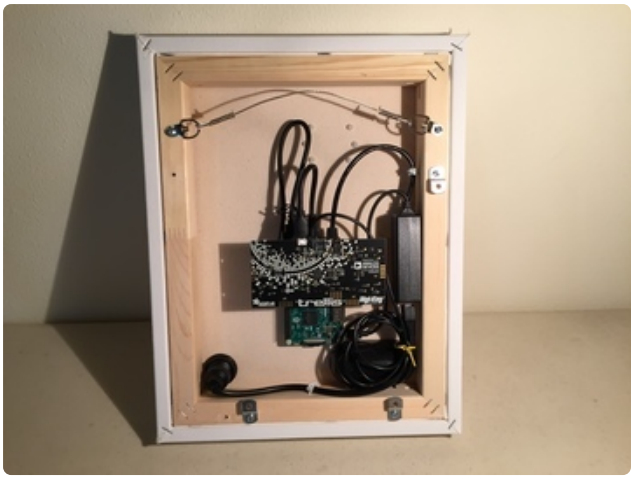
In this step I poked through the 6 spots in the canvas where I'm going to mount the Circuit Playground Express. I used the awl widen the holes just enough to screw the standoffs through the canvas.

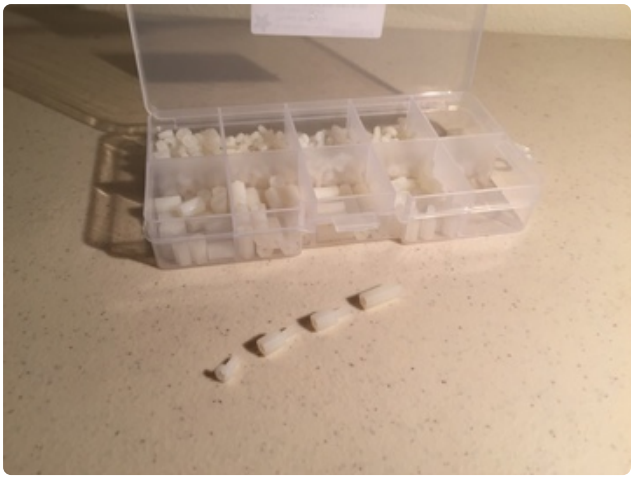
I had to detach the standoffs from the CPX initially here. The pictures show how I used 6 of the nylon hex nuts to attach the standoffs to the canvas. Finally, I placed the CPX on the front over the standoffs and screwed it in.



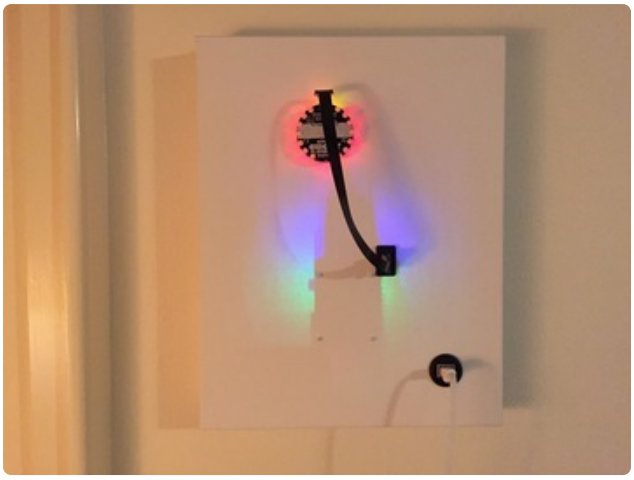
In these pictures, I've booted up the system and tried a few different light configurations. Once you get into the REPL, use commands like this to control the NeoPixels:

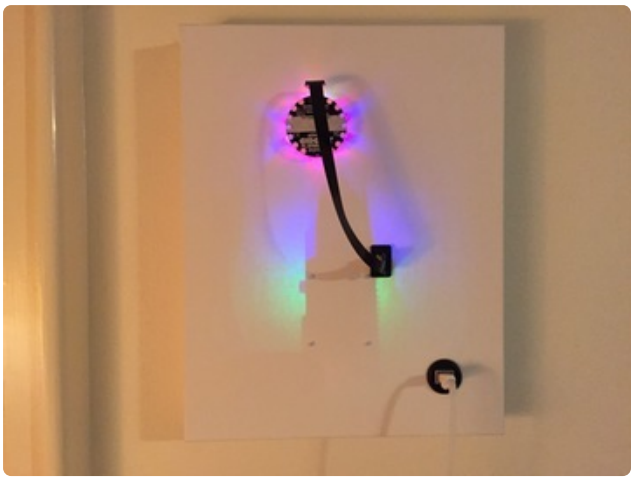
```
import board, neopixel
pixels =
neopixel.NeoPixel(board.NEOPIXEL,
10)
pixels.fill((0,0,30))
pixels[0]=(0,60,0)
```





These pictures show how I re-attached the NeoTrellis to the Raspberry Pi using the short USB cable. This time I used one more of the short nylon standoffs to avoid the NeoTrellis M4 from contacting the USB panel mount cable.





Here I am testing the lights on both the Circuit Playground Express and the NeoTrellis M4 to find combinations that look nice.

At this point we could declare this CompuCanvas finished from a hardware point of view, and refocus on creating animations for the lights or other programming exercises. But read on if you want to see how I placed another feature on the front of the canvas using a different type of cable assembly.

More External Features

One of the tricky things about making transitions through the canvas is finding cable parts that make right-angle type turns on each side. The ribbon cable system I

showed in the last section may be the best set of parts Adafruit currently offers for these right-angle USB cable turns.

But another way of making right (or left, or up, or down) cable turns is to find (or fabricate) a custom cable with the desired length and orientation.

This picture shows a whole bunch of USB cables I have collected that have various styles of right/left/up/down turns.

You can find these at sites like Amazon by searching for "short left right angle micro USB cable" or similar. These often come in pairs with both the "left" and "right" versions. I look for the pairs because it's sometimes hard to figure out which of the two I will need until I have them in hand.

I am going to use one of the "angled USB adapters" shown here - whichever one happens to angle the way I want.

In addition to one of the USB adapters shown above, I will use the following Adafruit parts. You may already have some of these from the earlier experiments in interior lighting. If so, you can repurpose them here as I did.

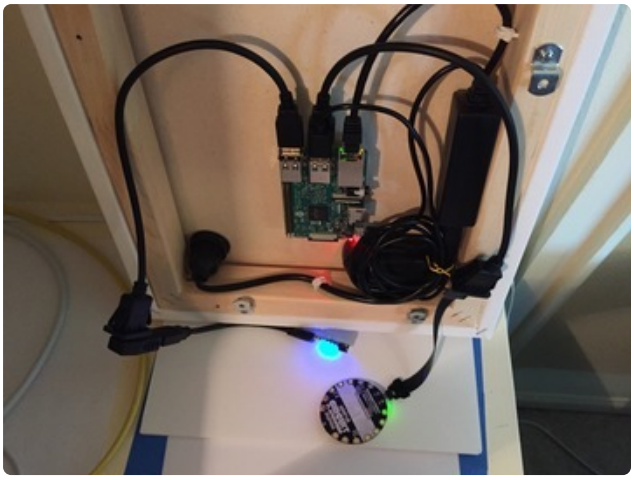
1 x Adafruit GEMMA M0 https://www.adafruit.com/product/3501
Gemma M0 with DotStar RGB LED

---

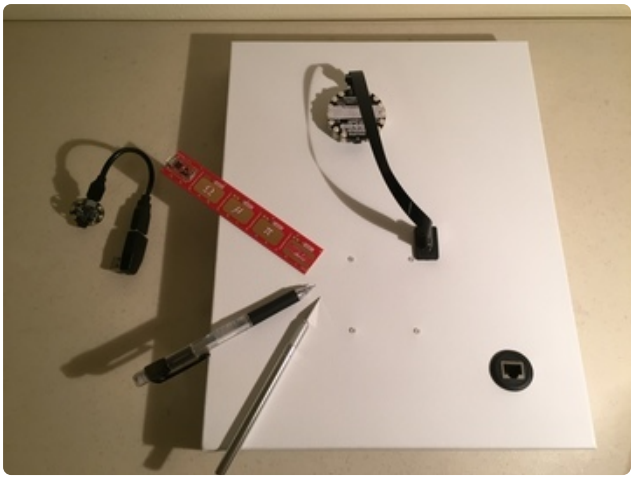
1 x USB cable - 6" A/MicroB https://www.adafruit.com/product/898
USB cable for Gemma M0

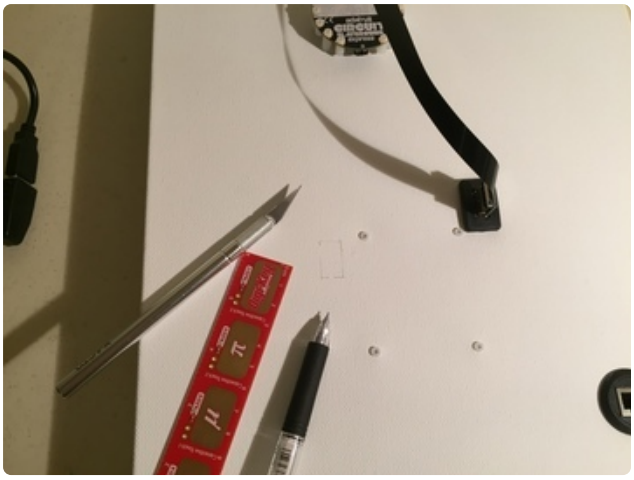
---

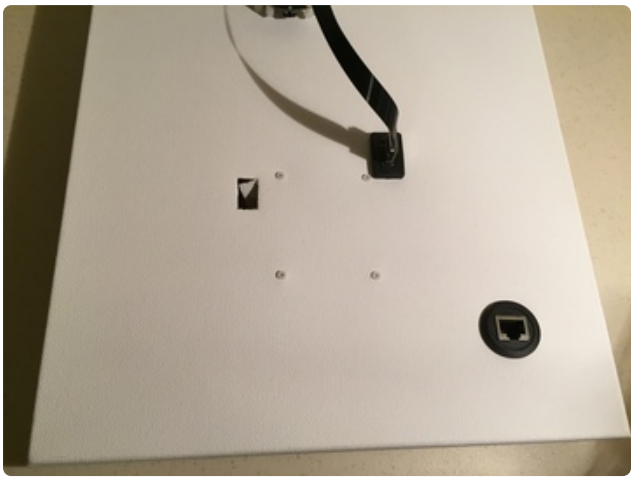
1 x Snap-In Panel Mount Cable - USB A Extension Cable https://www.adafruit.com/product/4055
another of the USB panel mount cables



These pictures shows the parts described above. As with the CPX, I tested the full cable assembly for the Gemma M0 before committing any of it to the canvas.



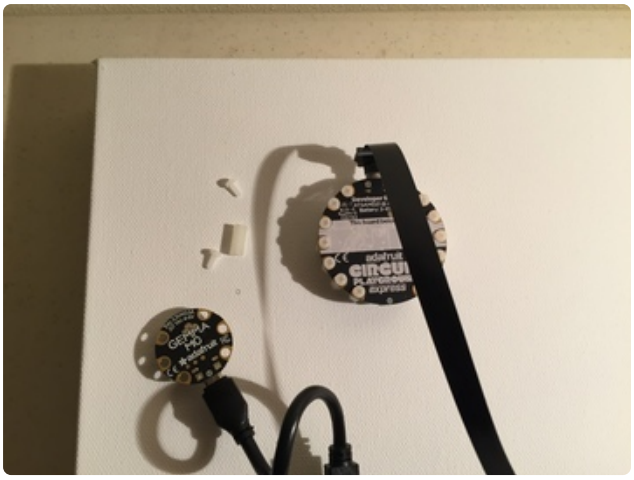




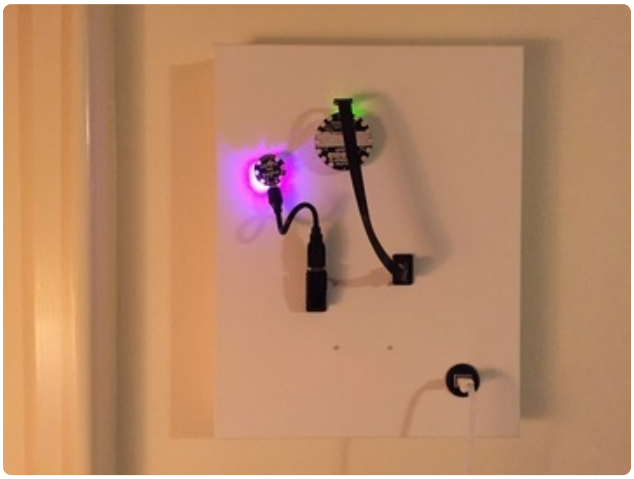
Here I measured and then cut another 1.1 by 1.9 cm hole in the canvas. I'm placing this one a little lower than the other. As before, I've checked both side of the canvas to make sure the cable will not bump into anything on the inside.

These pictures show how I've snapped the panel mount USB cable into the canvas - after making sure the orientation of the cable works with the USB adapter I chose.

Then I connected the cable assembly with the Gemma M0. This is another situation where I needed to do a lot of "cable massage" to get the short USB cable into the curly shape you can see here.

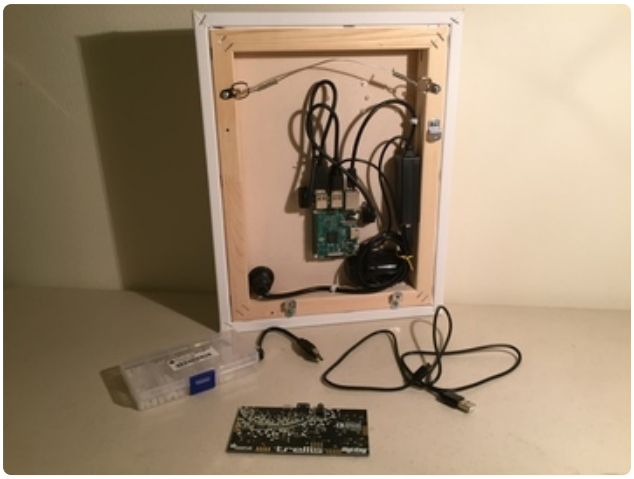


I think one set of the nylon screw parts will be enough to keep the Gemma M0 attached to the canvas. These pictures show how I marked a spot on the canvas to attach the Gemma M0 through the hole near the 'A' in 'GEMMA' on the silk screen.



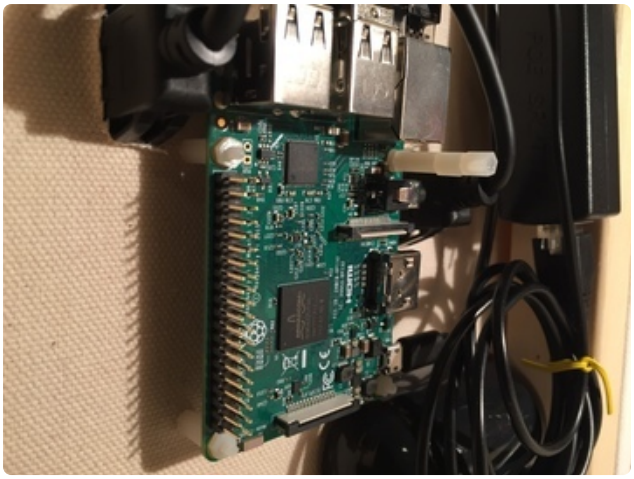
Now we can try out lighting tests with the DotStar RGB LED on the Gemma M0. See this Gemma M0 guide page (https://adafru.it/HAh) for all the details. You may need to first copy the `adafruit_dotstar` library to your Gemma M0. With that library you should be able to see some color from the DotStar in the CircuitPython REPL with commands like this:

```
import adafruit_dotstar
led =
adafruit_dotstar.DotStar(board.APA102_SCK,
board.APA102_MOSI, 1)
led[0] = (80, 0, 80)
```

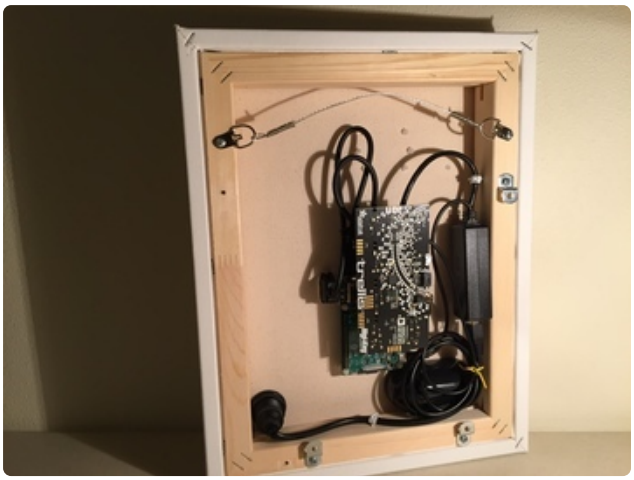


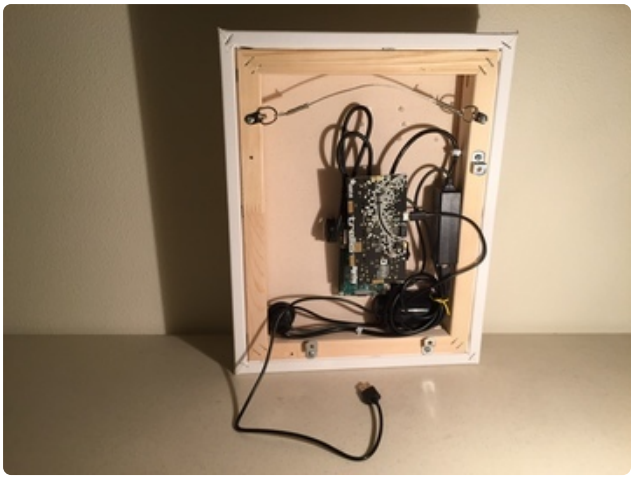
Here I will once more attach the NeoTrellis M4 to the Raspberry Pi using the nylon screw set. The first picture shows both USB cables, but I will only use the longer cable for this.

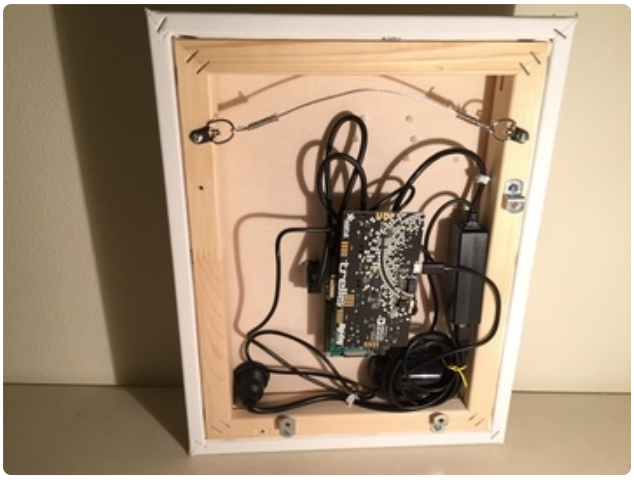
I want the NeoTrellis M4 to sit vertically, behind the rPi and the USB cables sprouting out of it. Note that I am connecting to a different corner of the rPi than in previous instances where I mounted the NeoTrellis M4.

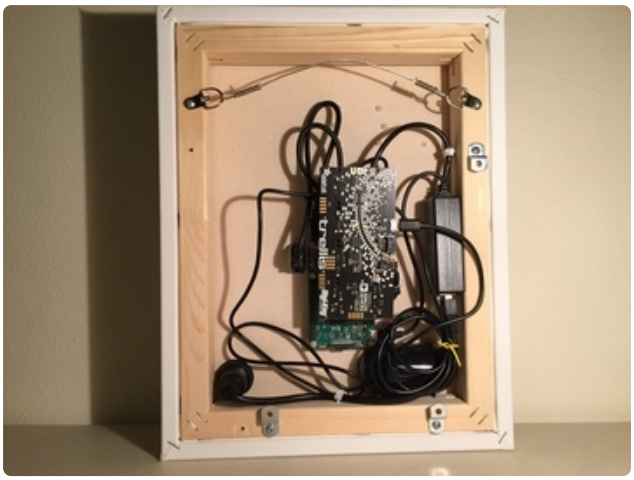


trellis
DigiKey
ANALOG DEVICES

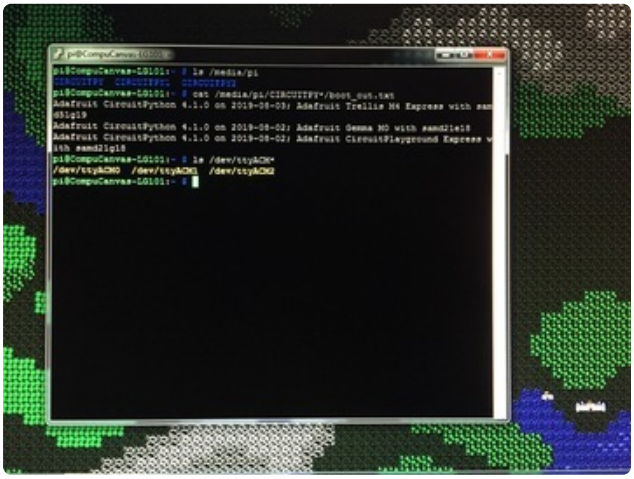








I threaded the USB cable for the NeoTrellis M4 through the USB speaker cable and then arranged it so the extra cabling rests on the bottom of the canvas frame, as shown here.



This picture shows what it looks like when you boot a Raspberry Pi with 3 CircuitPython devices attached.

Note how we can print out the `boot_out.txt` files to see which of the `CIRCUITPY` directories corresponds to which device.

Also note how we have the 3 `ttyACM` devices in `/dev`. I don't know how to tell which is which other than using `screen` to connect to the REPL of each one.

Here I am conducting light tests with all three CircuitPython devices.

I am now going to declare this CompuCanvas finished from a hardware point of view. The Python light show code could be tweaked endlessly, but I will leave that as an exercise to the reader.

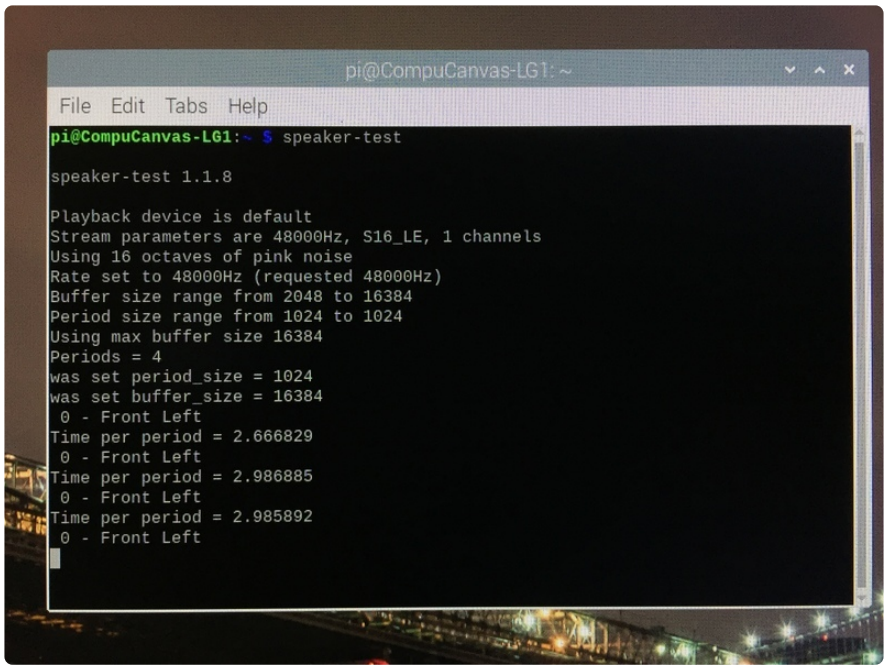
Here are several more pictures with different lighting configurations.

Now with the physical construction of this CompuCanvas finished, we are almost at the end of this guide.

But there is one loose end to return to: the configuration of the USB speaker. The next (and final) section will detail how to test and configure the USB audio.

Audio Addendum

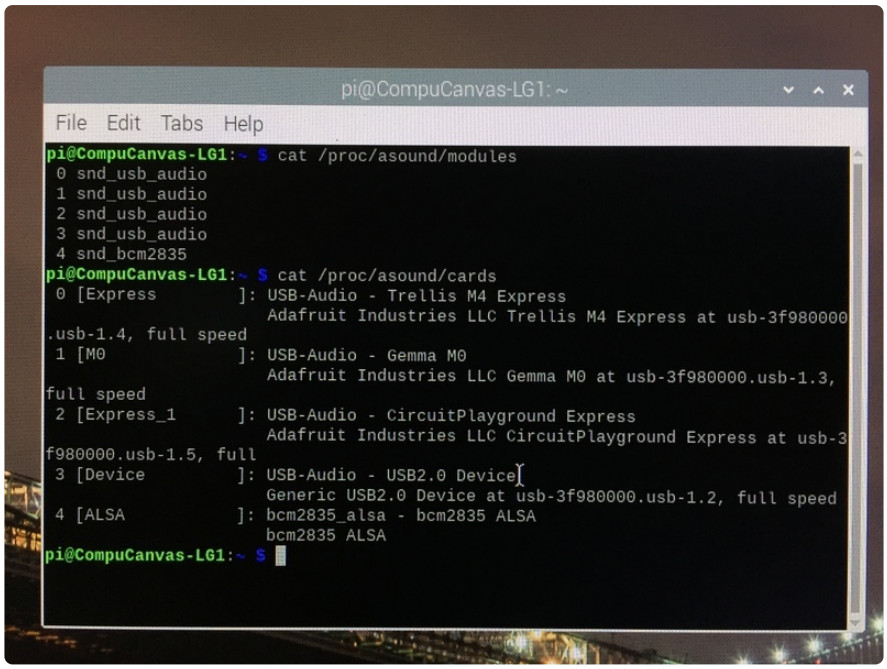
Raspbian comes with a command called `speaker-test` which is the first thing I try when setting up audio. The picture below shows what it looks like with the audio correctly configured. More importantly is what it sounds like! If everything is working, you will hear static ("pink noise") from the USB speaker. After listening for awhile you can press `Control-C` to end `speaker-test`.



So you might get lucky and find that your speaker just works, without any tweaking. I was not so lucky. When I initially ran `speaker-test`, it exited immediately with an error (and there was no sound).

If your speaker is already working, skip ahead to where I talk about the `espeak` command. Otherwise the next several steps will cover how I diagnosed and fixed the audio issue.

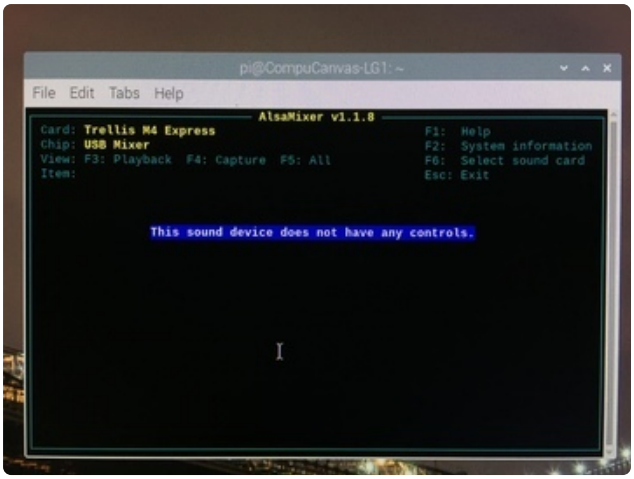
I found this StackExchange writeup about ALSA (https://adafru.it/HAi) to be very helpful. One thing it recommends is looking at the files `/proc/asound/modules` and `/proc/asound/cards`. The picture below shows how I used the `cat` command to print out the contents of those files.

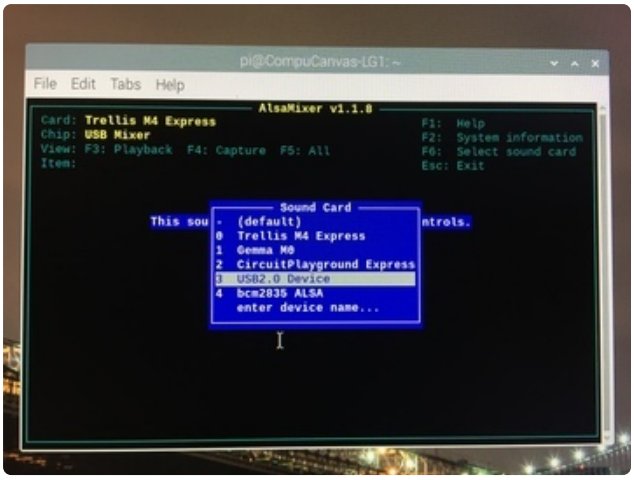


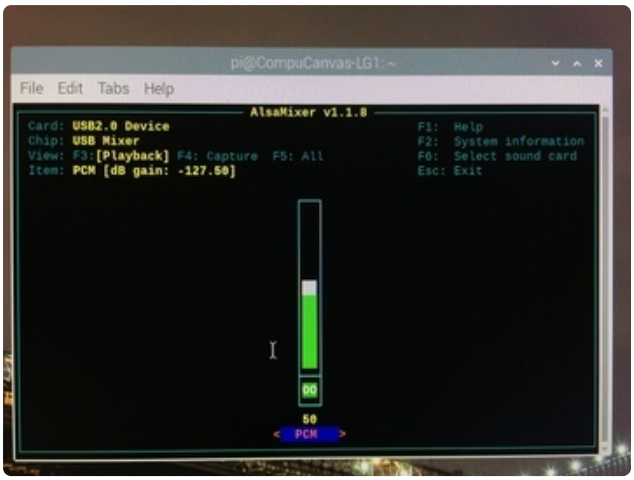
The picture above shows that 5 audio devices were detected! Why is this? Apparently CircuitPython devices register as audio devices, and in my case the first 3 devices listed are the NeoTrellisM4, the Gemma M0 and the CPX.

The device with index `3` is the real USB speaker. The last device ("bcm2835") is the audio system inside the Raspberry Pi which uses the rPi headphone jack.

The first device listed will be the default audio device, so in my case Raspbian was initially trying to use the NeoTrellis M4 as the speaker (and this failed `speaker-test`).







Another way to check the audio configuration is with the `alsamixer` command. The pictures here show that when I first ran it, the "Trellis M4 Express" card appeared first.

The second picture shows how you can press `F6` to switch to a different sound card.

The final picture shows how I selected the USB speaker. This allows for controlling the volume (set at 50 in the picture) with the up and down arrow keys.

You can change the volume here and Raspbian will continue to use that setting after you exit `alsamixer`. However we cannot change the default sound card from here. I'll show how to do that in the next step.

In the above steps, I determined that my USB speaker is the device with index `3` in `/proc/asound/cards`. By default, Rasbian is using device `0` (the NeoTrellis M4). So I need to change the default value from `0` to `3`... but where?

The answer to that question was in the StackExchange link (https://adafru.it/HAi). We can change the default audio card by editing this file: `/usr/share/alsa/alsa.conf`. This file is owned by the root user, so we need to launch an editor with `sudo`. You could do it with the `nano` editor like this:

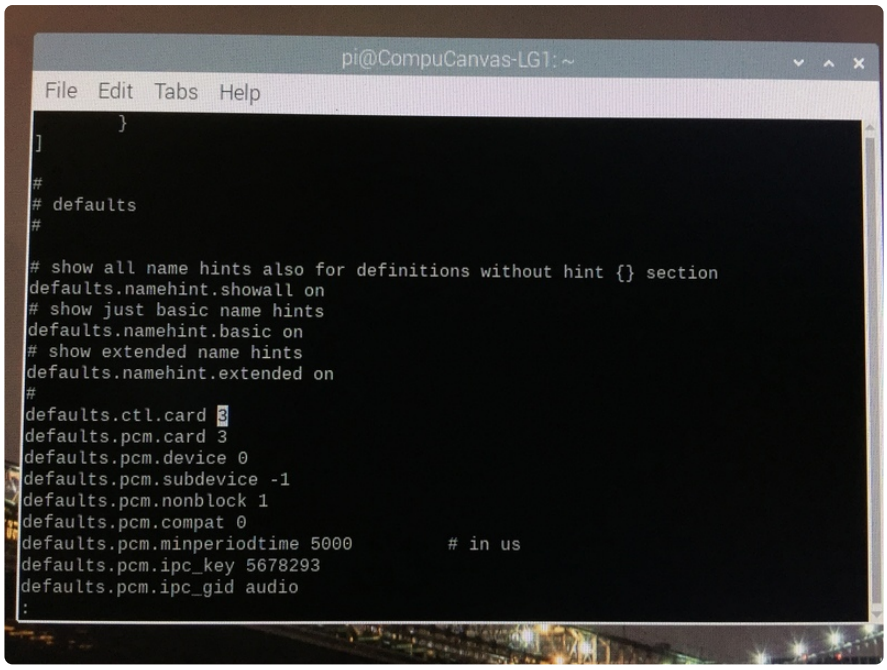
```
sudo nano /usr/share/alsa/alsa.conf
```

Two lines need to be changed in the `alsa.conf` file. Look for these 2 lines:

```
defaults.ctl.card 0
```

```
defaults.pcm.card 0
```

The picture below shows how I changed those values from `0` to `3` because my USB speaker was assigned to audio card `3`.



After changing the 2 values in the `alsa.conf` file, reboot to allow the changed settings to be processed:

```
sudo reboot
```

After this reboot, try the `speaker-test` command again and you should hear some noise from the speaker. You can run `alsamixer` again to adjust the volume.

Now, with all that setup done try this from the command line: `espeak hello`

You should hear your CompuCanvas say, "hello". You will also see a lot of error messages in the terminal window. If you are hearing sound, it's safe to ignore these messages. You can hide them in your command line invocations like this:

```
espeak "hello world" 2>/dev/null
```

The last bit (`2>/dev/null`) hides the error messages by sending them to the `/dev/null` device (sort of a digital trash can).

Below is a `bash` script I wrote that you can use to have your CompuCanvas announce itself and its IP address.

```
#!/bin/bash

function espeaker() {
  espeak -a 50 "$@" 2&gt;/dev/null
}

IP_ADDR=`hostname -I`
while [ -z $IP_ADDR ]; do
  sleep 1
  IP_ADDR=`hostname -I`
done
ESPEAK_IP_ADDR=`echo $IP_ADDR | sed -e 's/\(.\)/\1 /g' | sed -e 's/\./dot/g'`

espeaker "Hello from Comp U Canvas"
sleep 2

espeaker "I P address"
espeaker -s 100 "$ESPEAK_IP_ADDR"
sleep 2

espeaker "repeating I P address"
espeaker -s 100 "$ESPEAK_IP_ADDR"
```

To try this out, use an editor to create a file for the script:

`nano ~/cc-hello.sh`

This will open `nano` on a new file named `cc-hello.sh` in your home directory. You can paste the above script into that file and save it. Then run the following to make the file executable:

`chmod ugo+x ~/cc-hello.sh`

Now try to run the script:

`~/cc-hello.sh`

The `~` in the commands above refers to the user's home directory, which is `/home/pi`. So the full path of this script is: `/home/pi/cc-hello.sh`. I'm going to use that in the next (and final) step of this guide.

The last thing I want to show is how you can make the `cc-hello.sh` script run when the CompuCanvas boots up. This is useful when taking the CompuCanvas to a new place (like a meetup), because you will need the IP address to login to the CompuCanvas.

There is probably more than one way to tie `cc-hello.sh` into the Raspbian boot sequence. I do it by editing the file `/etc/rc.local`. That file is owned by the root user, so you'll need `sudo` to edit it, like this:

```
sudo nano /etc/rc.local
```

Notice that the very last line of this file has `exit 0`. Add the line shown below just before the `exit 0` line:

```
/home/pi/cc-hello.sh
```

Then save the changes, and `sudo reboot` one more time. When the CompuCanvas starts up again you should hear it announce its IP address.