A Framework For Making Affordable & Stylish Modular Controllers

by **Fuzzy-Wobble** on December 19, 2011

Table of Contents


Author:Fuzzy-Wobble author's website
I am a human from planet earth. I use my brain to make things.

Intro: A Framework For Making Affordable & Stylish Modular Controllers

****

I have designed a **framework** for making affordable and stylish modular controllers. You can use the content of this Instructable to make a wide range of controllers for a wide range of applications relevant to artists, DJs, VJs, gamers, producers, and the like. The DJ controllers I showcase in this document serve only as examples of (more conventional) interfaces you can create within the framework.

I focused on making this project affordable, stylish, and most important, builder friendly. The controllers can be re-programmed to send serial, MIDI, or HID messages. The modular design allows you to plug the controllers into one another, thus requiring only one USB port on your computer. Each module can have approximately 28 digital inputs/outputs, 23 analog inputs, and 4 rotary encoders. Those more savvy could add components such as touchscreens, sensors, pressure pads, etc., to the controllers using this framework. No special tools or equipment are required to build these controllers beyond a basic soldering iron and wire stripper.

I hope this Instructable document will become the standard for people wanting to build DIY controllers.

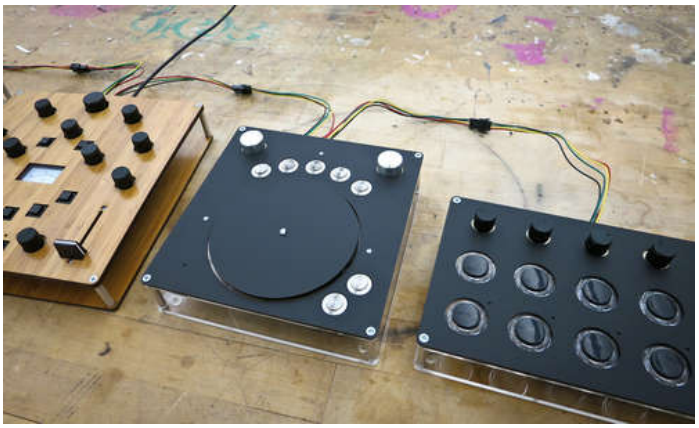
****

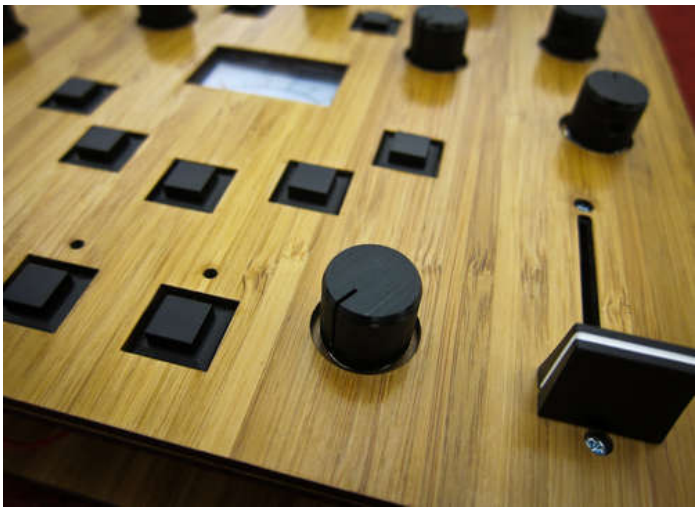
If you like this Instructable, vote for me so I can win a ShopBot and build more! I am a student of design & technology, and an avid supported of DIY.

More info here .

These photos and more found on my photostream .

Remember to read this entire Instructable before beginning any of the constituent steps





Step 1: Components

I have spent endless hours searching for the best and most reasonably priced components from around the world. I ordered many different components to test out and compare, most of them were rubbish and were not used in project. Here is the list of the best stuff I found.

Microprocessor

- Teensy++ @ PJRC (USA)

Pushbuttons

- Arcade @ DJ Tech Tools (USA)
- Translucent Arcade @ Adafruit (USA)
- Many arcade button options @ Twisted Quarter (USA)
- Square plastic & large round @ Futurlec (China)
- Stainless steel fancy pushbuttons @ Sunshine (China, shipping is >40$ to USA, I have some extras so message me if interested)
- Mini arcade @ Newark (USA)
- Many unusual pushbuttons @ All Electronics (USA)

(Many of these pushbuttons are available in different sizes and colors from the retailers)

Faders

- Small @ Mouser (USA)
- Medium @ Mouser (USA)

(If these are out of stock, they may also be available at digikey.com or newark.com)

Slide knob

- Many options available at Newark.

Potentiometers

- Very smooth D shaft @ Digikey (USA)

- Very smooth knurled shaft? (still trying to find a nice one)

Potentiometer Knobs

- Machined, aluminum, set-screw, 6.4mm @ Newark (USA) in 3 sizes, black or silver
- Smooth, aluminum, set-screw, 6.4mm @ Newark (USA) in 3 sizes
- Round, knurled, rubber, 6mm @ Newark (USA) many colors
- Large, knurled, metal @ SparkFun (USA) black or sliver

(Knurled knobs can only be attached to knurled shafts. There are many knobs available online but these are surely the best ones I tested.)

Encoders

- D shaft, 18 ppr, switch @ Newark (USA)

(Search 'PEC11-' on Newark to get a long list of nice and affordable encoders. I used PEC11-4115F-S0018 and PEC11-4215F-S0024 in my build. I tested many encoders and found the Bourns ones to be best. They have both D, and knurled shaft options available)

PCB

- Super nice PCB @ Sure Electronics (China)
- Protoboard 777 @ Futurlec (China)
- Mini @ Sparkfun (USA)

Wires & Connectors

- Pre crimped 12", Pre crimped 24" @ Pololu (USA)
- Module connectors @ All Electronics (USA)
- Crimp connector housings @ Pololu (USA)
- Wire @ All Electronics (USA)

Multiplexer

- CD4067BE @ Digikey (USA)

Header Pins

- Available from All Electronics, Sparkfun, and Adafruit
- Female @ Sparkfun

Standoffs

- Many 6-32 options @ All Electronics (USA)

LEDs

- Many LEDs @ Super Bright LEDs (USA)

Hardware
All hardware (nuts and bolts) can be found at McMaster (USA)

Other
Soldering iron, wire cutters/strippers.

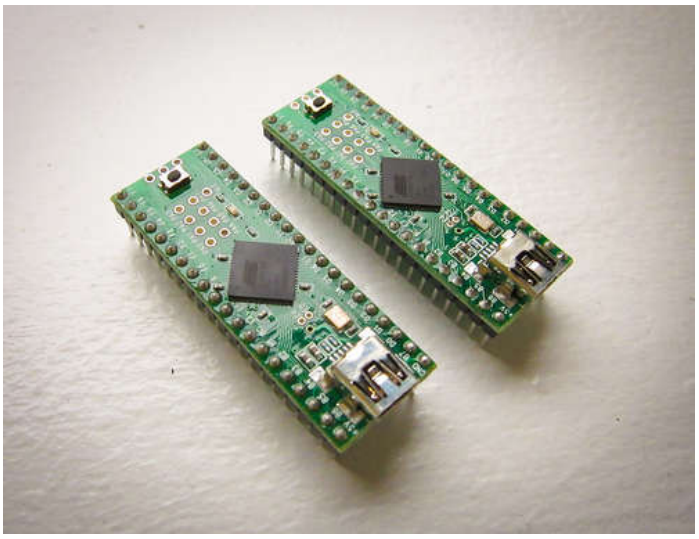
There are many other components you may want to include in your interface such as LCDs, touchscreens, trackballs, pressure pads, proximity sensors, etc. SparkFun and Adafruit are the best resources for these kinds of components. Although the controllers I showcased in this Instructable are somewhat conventional, I encourage you to get crazy. DIY gives you the power to do anything you can imagine - just wait till you see the next controller I am going to build!

Read the datasheets carefully before buying components.

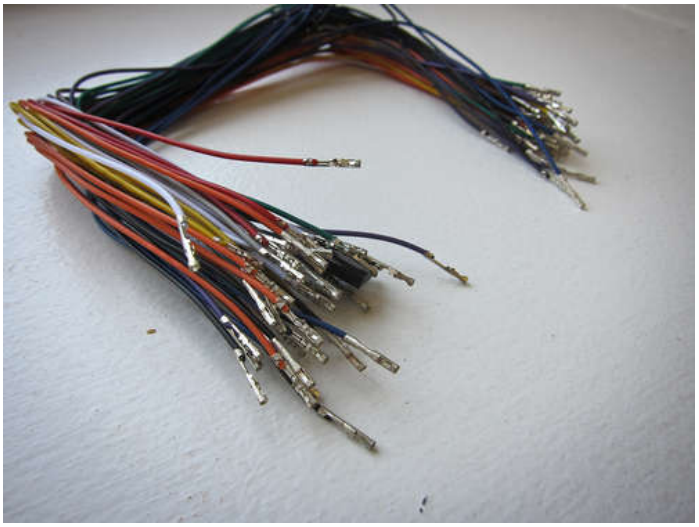
Discount/surplus stores like Futurlec & All Electronics have some components which are totally bogus so be extra careful before buying from these stores.

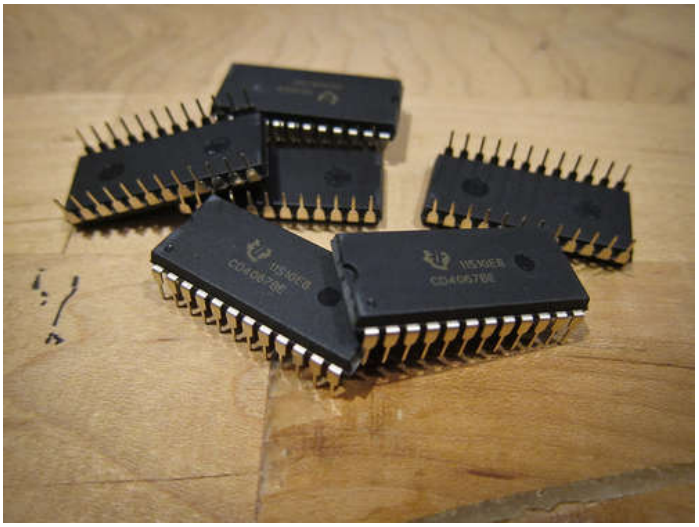
Note that all these components are panel mount (with exception to the LEDs). There is an important distinction between surface mount and panel mount components. Surface mount components fasten directly to the printed circuit board (PCB) while panel mount components fasten to the top panel and connect to the PCB via wire and solder connections. It is usually not possible to use surface mount components unless you fabricate custom PCBs and have expert soldering skills. Using panel mount components makes this project accessible to anyone.

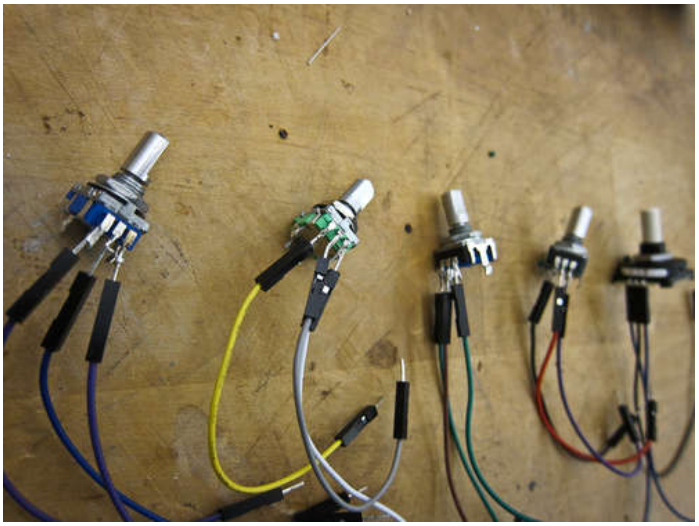
So how much do these controllers cost? I don't kow how much **your** controller is gonna cost but mine were cheap, really cheap!

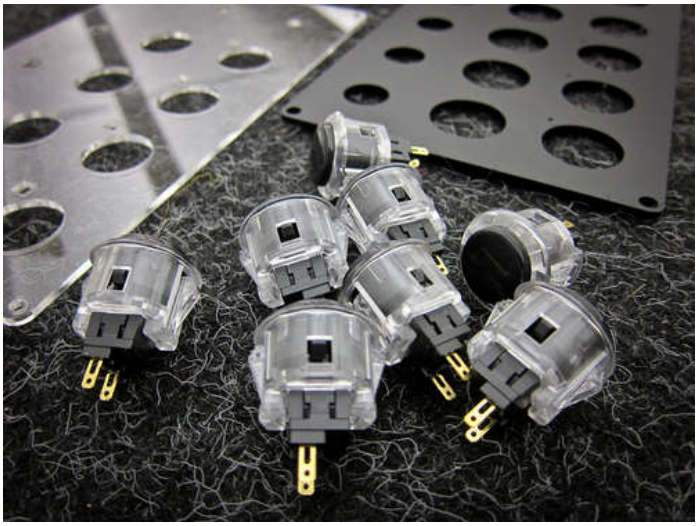


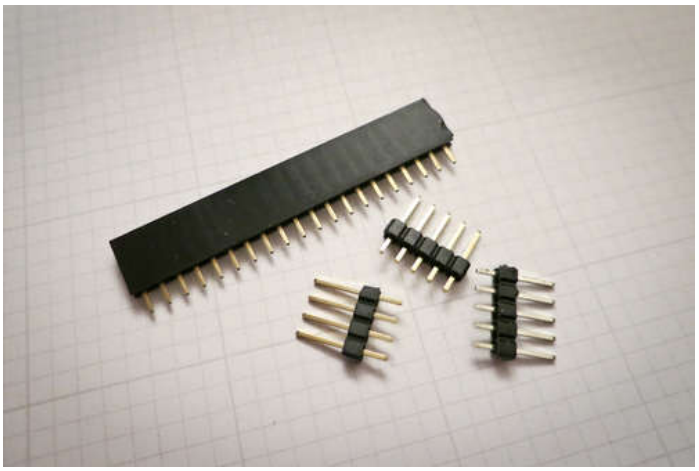

MB-B811114
SURE

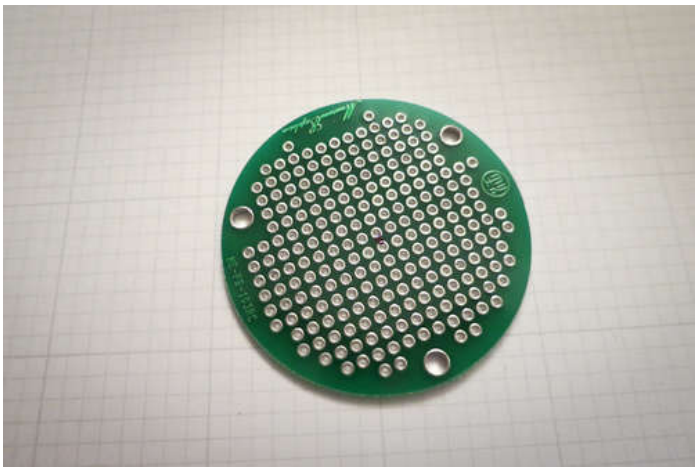












Step 2: Set Up Teensyduino (Arduino + Teensy)

Those of you who are familiar with Arduino may be wondering why I am using a Teensy instead of Arduino. Four good reasons. One, Arduino cannot act as a native MIDI/USB/HID device as Teensy can. Two, Teensy has many more inputs/outputs and more interrupt pins. Three, Teensy can be easily soldered to the protoboard. Four, Teensy is comparatively cheaper. I use Arduino for many projects, but not for controllers. We will, however, borrow the user friendly programming environment from Arduino and integrate it with Teensy to create the very awesome Teensyduino.

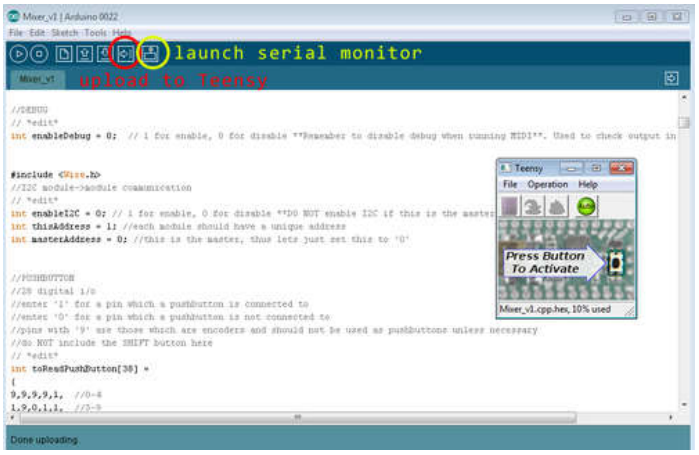
Some of you also may be wondering, what the hell is Teensy or Arduino? Well don't worry if you are not familiar with these. They are just the small, programmable computers (microcontrollers) that are the brains of your controller. They are easy to use - just edit the code on your computer and upload it!

Follow the Teensyduino tutorial carefully. It contains all the step-by-step guide to get your Teensyduino up and running.

If you are installing the Arduino environment on your computer for the first time I would recommend **not** installing the latest version (I still use v0022).

Once you have installed Teensyduino, proceed to the getting started tutorial to ensure everything is working. Dont forget to use the Teensy loader .

Although the Teensy loader tells you to 'Press Button To Activate' you can upload the code directly to the Teensy using the 'Upload' button in the Arduino environment (see image). This is important because once your Teensy is secure inside your controller the hardware button will not be easily accessible.



Step 3: Getting Familiar With Teensy++

The diagram above explains the Teensy++ pin configuration. You need to understand how these pins differ.

Digital pins
All numbered pins on the Teensy++ board can be used as digital inputs/outputs. A digital pin is simply an on/off pin. It can be on (HIGH) or off (LOW). We use digital pins as input to read pushbuttons, or as outputs to turn LEDs on.

PWM pins
Pulse width modulation (PWM) means we can send a variable voltage out of that pin between 0 volts and 5 volts. PWM pins are ideal for LEDs as they allow us to vary the brightness of the LED.

Interrupt pins
Each pin with 'INT' is an interrupt pin. Interrupt pins are important cause we need them to read encoders. They are simply digital pins which have special functions attached to them and will notify the Teensy when their value has changed. The Teensy++ board has 8 interrupt pins, allowing you to read four encoders.

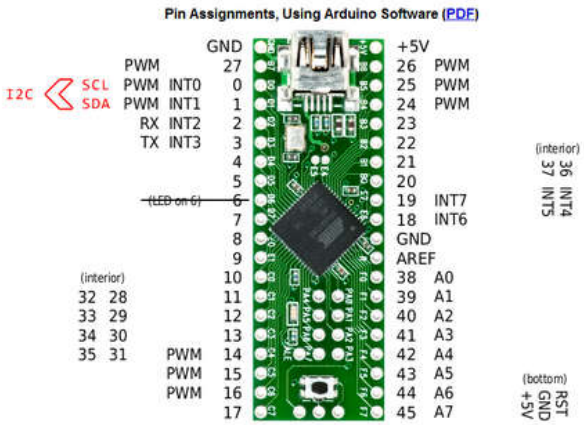
Analog input pins
Analog pins allow you to read variable voltages between 0 volts and 5 volts. We use analog pins to read potentiometers.

+5V/GND
In the top right corner we have power (+5V) and in the top left we have ground (GND). All Components will need to use ground, and all analog components will need to use both ground and power.

I2C pins
The I2C pins (SDA, SCL) are what we use to enable module->module communication. When module->module communication is enabled these two pins can no longer be used by a rotary encoder.

Other
Pin 6 is not used cause it already has an internal LED attached to it. Bottom pins (+5V, GND, RST) are not used, nor are the two side pins GND and AREF.

Internal pins
Pins 28-37 are located internally on the board.



Step 4: Testing Individual Components

This is the surely the most critical step of this Instructable. You need to build a test circuit so you can understand how the components are hooked up and used with the Teensy. This step will also cover serial communication using the Teensy.

In the diagram you will see a breadboard which has the following components (left to right)

- Teensy++ microcontroller
- Two potentiometers
- One rotary encoder with pushbutton
- One mini LED
- Three mini pushbuttons

Hookup the circuit as you see it in the diagram. Likely, your pushbuttons will look much different than mine, but don't worry about that cause all pushbuttons hook up the same - one end to ground and the other to a digital input. If you have pushbuttons which are not breadboard-friendly just solder some wire connections to them quickly.

You may also notice my connections between the pushbuttons look a bit goofy (the one yellow wire). I did this for a reason. It shows that instead of connecting the pushbutton ground directly to the ground linked to the Teensy (blue strip) you can plug the ground from one pushbutton to the next, so long as one of the pushbuttons is linked to the Teensy ground. This is important cause when you are wiring your components into your panel the ground connection on the Teensy is usually not as convenient as the ground of the neighboring components. This is an excellent way to reduce the number of wires in your controllers. I also did the same thing for the potentiometers with the grey wires.

Usually LEDs should have resistors in series to prevent them from burning out over time. Since this was just a quick test I left the resistor out of the circuit. Also note that LEDs are polar, meaning they have a power and ground end. If not hooked up correctly they with fail to light up.

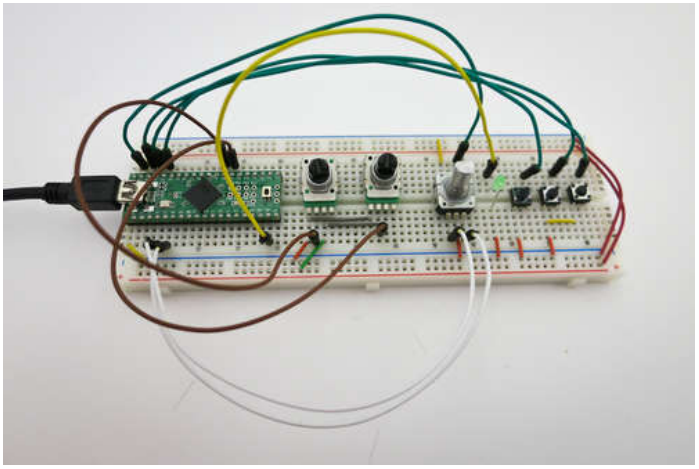
Once you have the circuit hooked up, proceed to opening to code. Read the many comments within the code to help you understand what is going on.

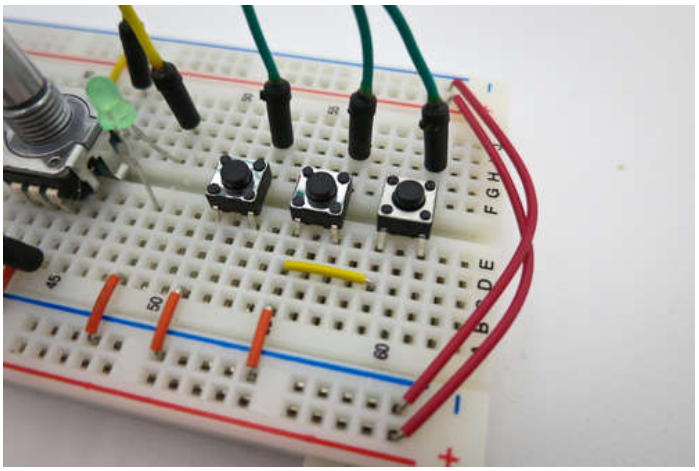
Upload the code attached to your Teensy++ and launch the serial monitor. The serial monitor should give you feedback as you interact with the components: 'button 24

pressed', 'analog 2 value: 344', 'encoder +'. If your serial monitor is constantly spewing out values then you have hooked up something incorrectly. Make sure your serial monitor is running at 9600bps.

The LED is set to flash at different brightness's depending on which pushbutton you click.

Before you move onto the next step ensure this is working and you (without doubt) understand how to hook up each of these components to the Teensy++.





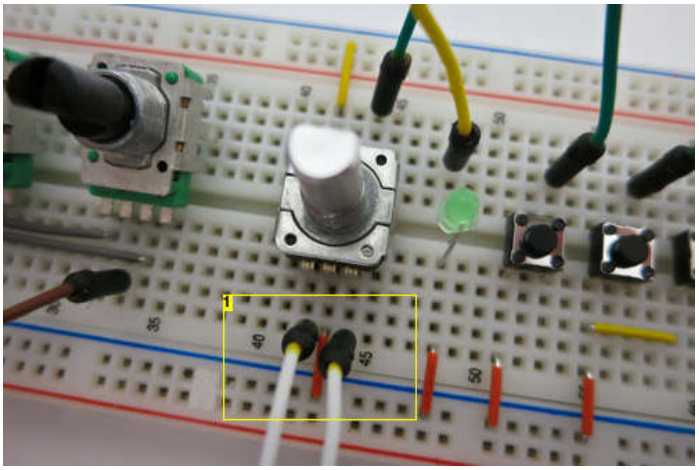


Image Notes
1. encoder 1 / ground / encoder 2

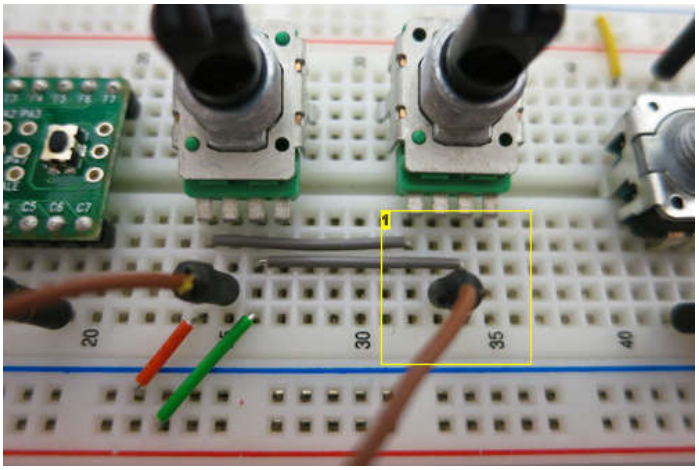
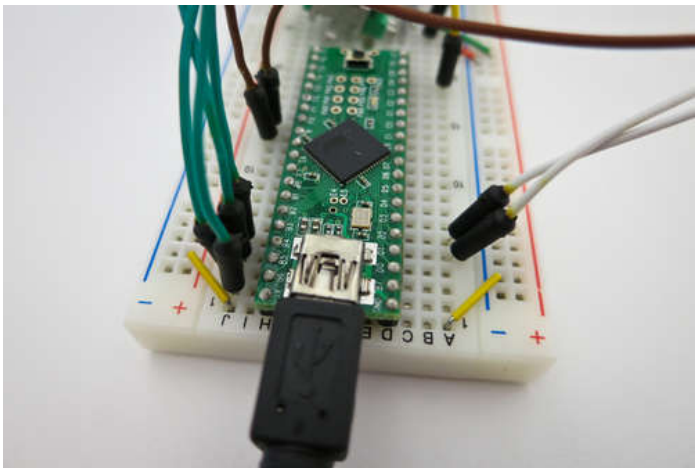
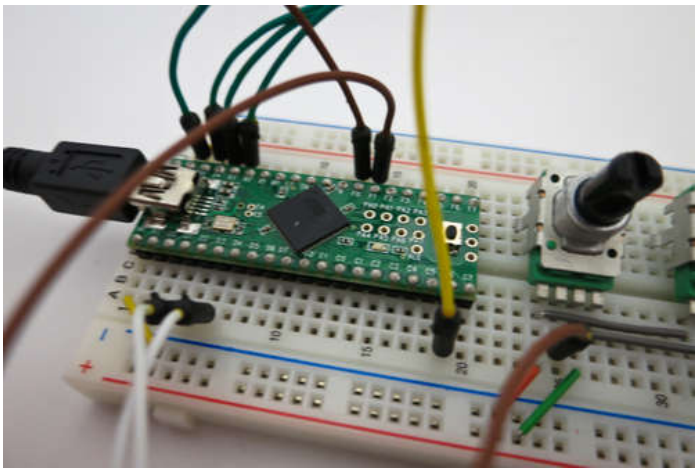
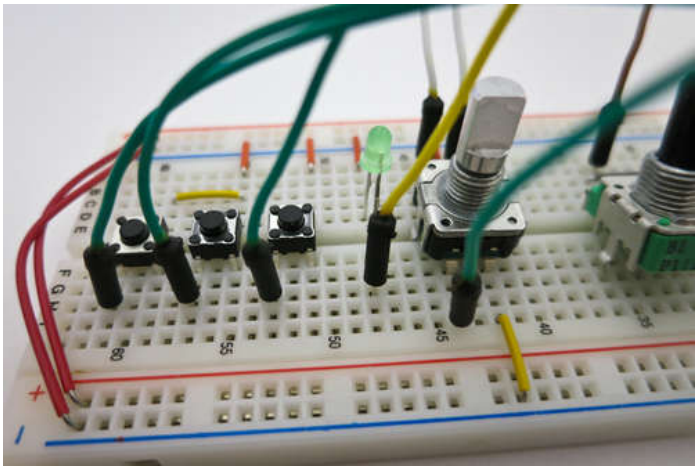


Image Notes
1. ground / output / power







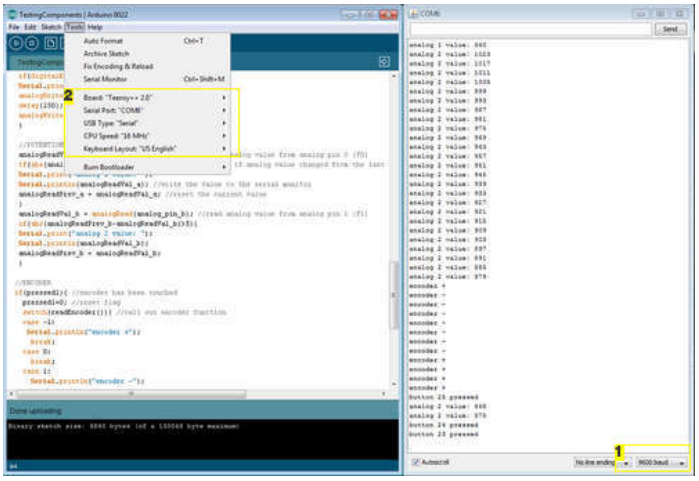


Image Notes
1. 9600bps speed
2. Teensy 2.0
USB Type 'Serial'

File Downloads

TestingComponents_v1.pde (3 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'TestingComponents_v1.pde']

Step 5: Design, Layout & Body.

You may suspect I am some kinda master craftsman or something. No, I am terrible at working with hands and thus must rely 100% on laser cutting robots.

Ponoko is great for laser cutting services, they have an excellent material selection and super friendly staff.

For my controllers I used the materials Bamboo Amber 3mm , Black Matte Acrylic 1/8" , White Matte Acrylic 1/8" , Clear Acrylic 4.5mm , and Clear Acrylic 3mm .

In the diagram above you will see the sketch for my three layer design. This method of design allows to components to be counter-sunk. Below is the breakdown of layers for both the decks and sampler.

- Layer 1: Clear Acrylic 3mm
- Layer 2: Matte 1/8"
- Bottom: Clean Acrylic 4.5mm
- Platter: Matter 1/8"

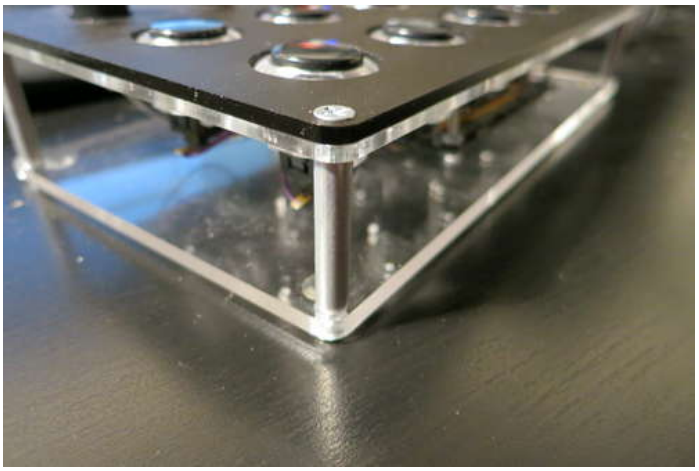
The bamboo mixer also uses a three layer design. Each layer is bamboo.

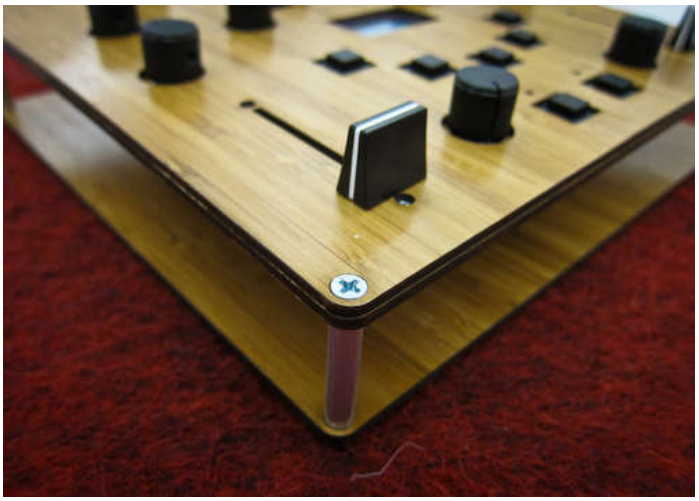
Very very very very very very helpful design tips (that I wish someone told me)

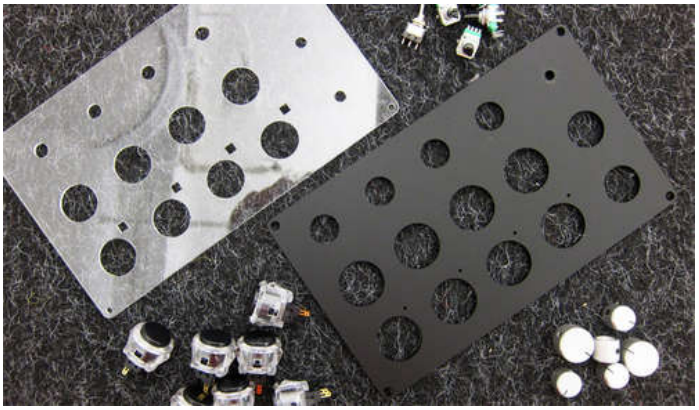
- Most important: Do not overcrowd your board! Leave lots of space between components. Think minimalist. Although your Teensy++ has many inputs available it is foolish to try and make use of all them. It is best to begin creating more modules rather than overcrowding a single board. Hookup will be a nightmare if your overcrowd the board.
- A shift button is a great way to reduce the number of pushbuttons. Using one shift button is highly recommended. When you hold your shift button, all the pushbuttons on your board will then send a different message thus doubling the total number of pushbuttons on your board.
- If you use non-matte acrylics on the front face they easily become covered will smudges.
- All wood material will require a coating of polyurethane (or some equivalent).
- Etching is possible (although I did not use it on my design).
- If you design your front face to be large, there will be undesirable flexing in the center when you engage a pushbutton. Keep your modules a reasonable size. My mixer is 230x230mm.
- Bamboo is more flexible than acrylic. The acrylic is quite stiff.
- To reduce flex you can add tiny bolts which join layer one and two (see image).
- Do not try to glue the top two layers together.
- When creating your layout you need to reference the datasheet constantly to ensure your cutouts match the components.
- If the specific components cutout dimensions arenot given, it is best to increase the size by 0.1mm. For example, the rotary potentiometers have a bore diameter of 9mm, thus the cutout should be 9.1mm.
- Errors in laser cutting can get pricey so be sure to check your design numerous times before sending it off.
- The upper and lower layers can be spaced using aluminum standoffs.
- I made the standoffs at the front slightly shorter than the ones at the back to created a tilted front face.
- You can 3D print some custom corners (shapeways, ponoko, imaterialise) and add walls to your design, much like I did in MK1. Personally, I prefer the no-wall design as it enhances the DIY aesthetic and reduces costs.
- The gap between the slide potentiometer body and slide knob needs to be measured carefully to ensure there will be enough clearance for the slider to slide.
- If you are using bamboo, I would recommend using 3 ply for the bottom instead of single ply. It is a good idea to use a heavy/thick bottom for all of the modules.
- Print out your layout on paper before laser cutting to get a feel for the spacing.
- Pay close attention to the height of your components. Ones which are too tall will cause your controllers to look bulky and odd.
- When using the software design all three layers at once then separate the layers before cutting (see attached pdf).

I designed my layouts in Adobe Illustrator (free one month trial). I have attached the files for the mixer to this document. I would not recommend copying these cutouts as It is best to make your own custom design. When using illustrator be sure to switch your units to millimeters and add a grid to aid in layout/spacing.

Get creative and have fun with the design!







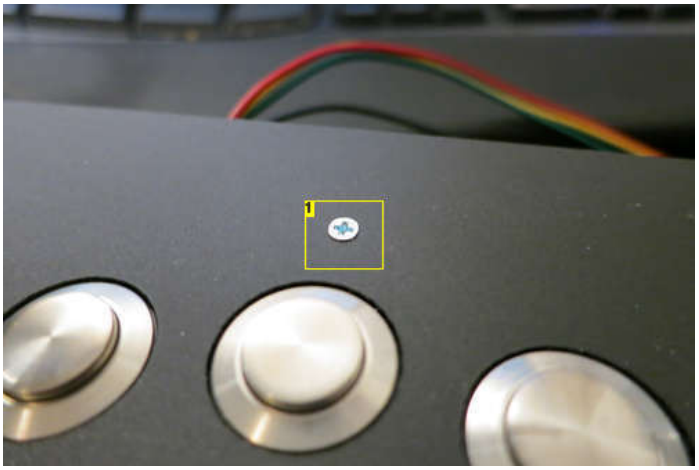
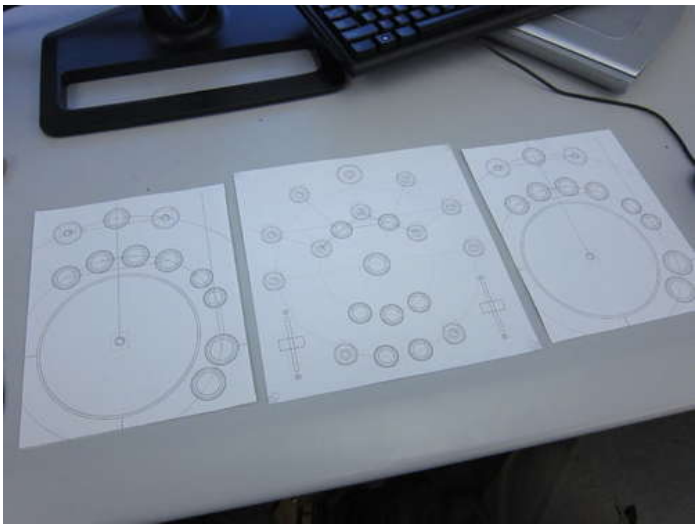
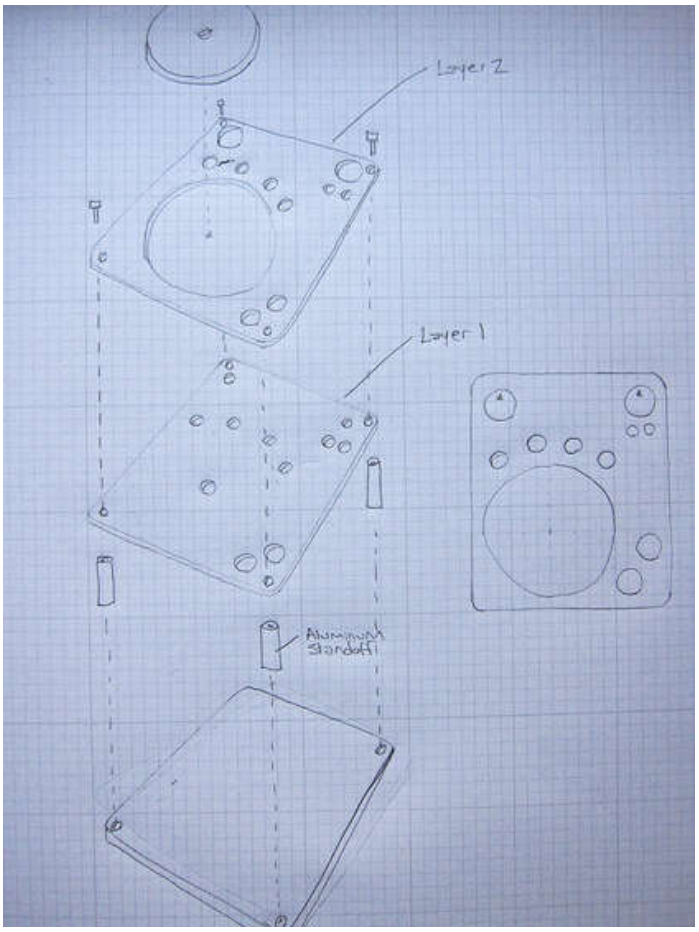
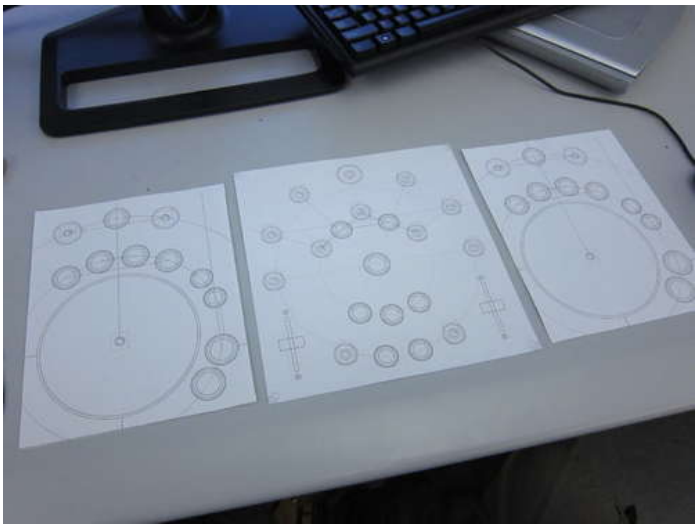


Image Notes
1. Tiny M2 bolt to join upper layers.







File Downloads

Mixer_v4.pdf (227 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'Mixer_v4.pdf']

MixerCut_v4.pdf (232 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'MixerCut_v4.pdf']

Step 6: Soldering Like A Pro

As you can probably tell there is going to be some soldering necessary to build your controller. If you have not soldered before don't fret, its really easy (if you follow my tips). If you don't have a soldering iron here is a super cheap station. You can also get irons at most hardware stores as well as Radio Shack. Below are four important tips to help you solder like a pro!

Buy a flux pen
This is imperative! They can be purchased here . Never ever try to solder without a flux pen no matter what anyone tells you.

Watch my videos

Use this method if you have a free hand available to feed the solder.

Use this method if you don't have a free hand available to feed the solder.

Clean your tip using a wet sponge
If you don't do this you iron will become useless very fast.

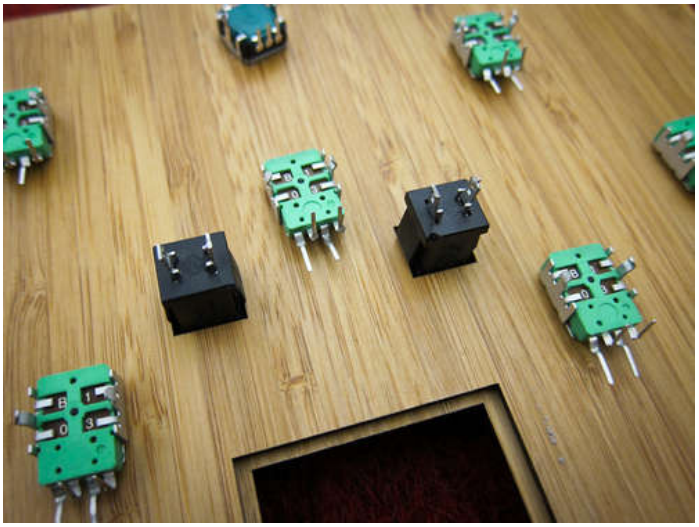
Use stranded wire
Solder stick better to wire with many strands. The crimped wire from Pololu is stranded.

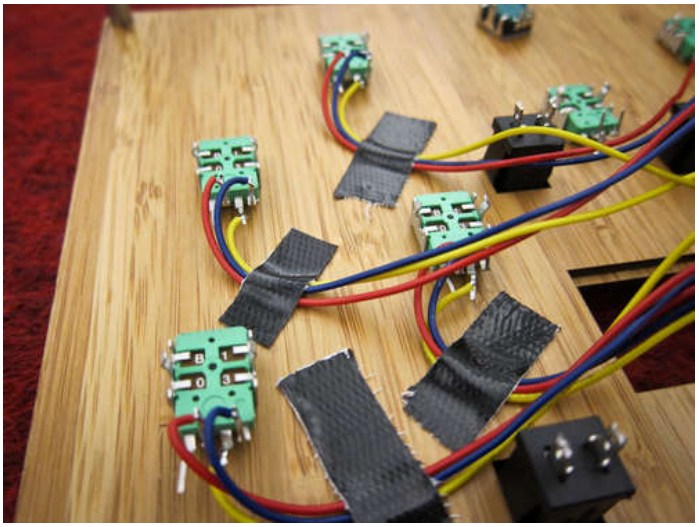
Step 7: Assembling The Top Panel

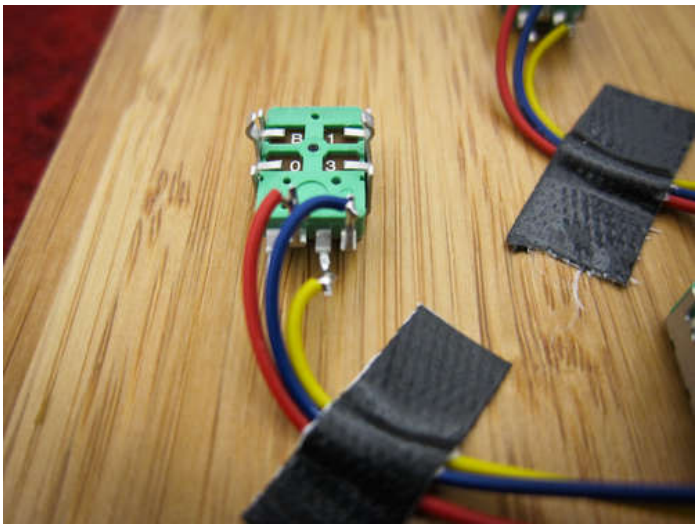
Fasten all your components
Some components will be snap fit, while others will have fastening nuts and bushings. All components should fit snug, if not, you may have made an error in the laser cutting.

Cut your crimped wires to appropriate lengths
Each crimped wire you bought then becomes two wires when you cut it in half. One end will be the crimped end and the other will be the exposed end which you cut. The exposed end solders directly to the components, while the crimped end plugs into the crimp housings. It is helpful if you make the wires long enough so that your top panel can plug into the bottom PCB while the top panel remains standing next to it (see image). If you are building a medium/large controller it would be wise to purchase 24" crimped wires, as opposed to 12". It is also important to know approximately where your PCB will be positioned on the bottom panel so you can cut your wire to the appropriate length. Some components (like arcade buttons) have large heights and will interfere with the components on your PCB. Notice on my sampler I had to position the PCB slightly off-center as to not interfere with the Teensy++ on my PCB - I wanted to keep the height as low as possible. If you cut your wire too long it may stick out the side, so take care when cutting - not to short and not too long.

Solder your exposed wire to your components
Color coating is not imperative, but helpful if you have enough wires. In the case you were foolish in your design and have an abundance of wires, it is helpful to begin connecting your ground/power connections between components (as shown in step 4). You should only do this if necessary. On my sampler I had a mess of wires, so connected all the ground connections from the arcade buttons to each other (see images). This way I only had one ground connection which needed to be plugged into the board. I did the same for the power and ground of the potentiometers (see images). You may want to add either tape or glue gun glue (see images) to your soldered connections. This way when you tug on the wire, there is less risk of it becoming detached from the component.







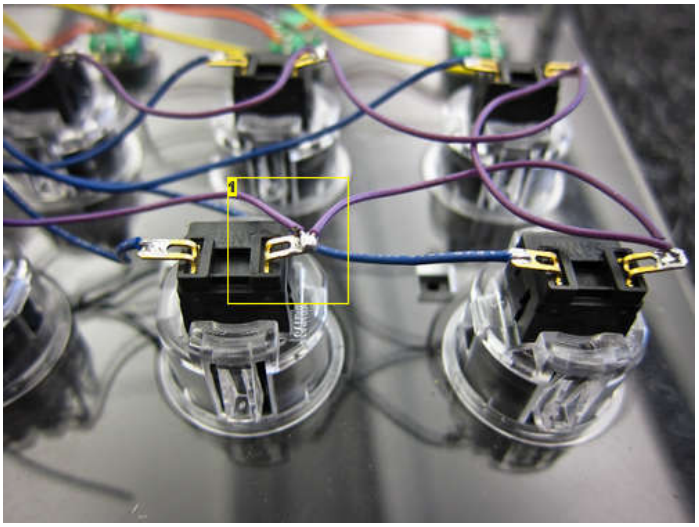


Image Notes
1. All ground connections for arcade buttons grouped together using purple wire.

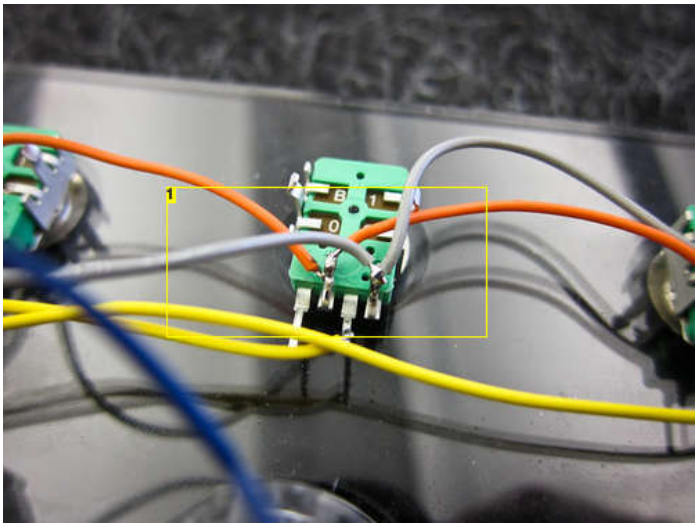
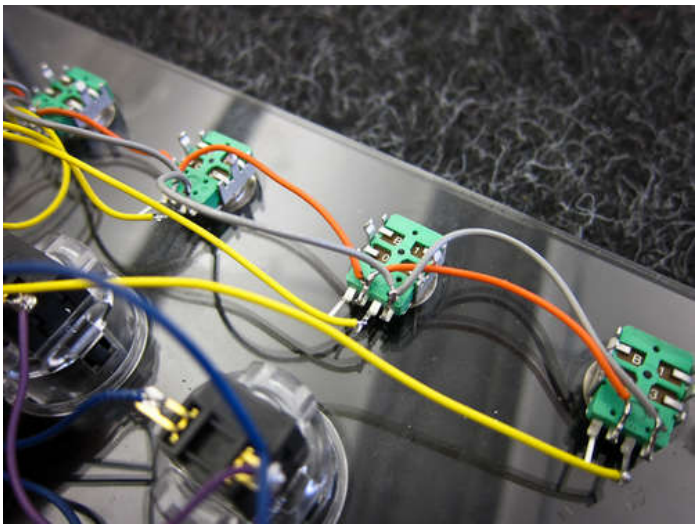
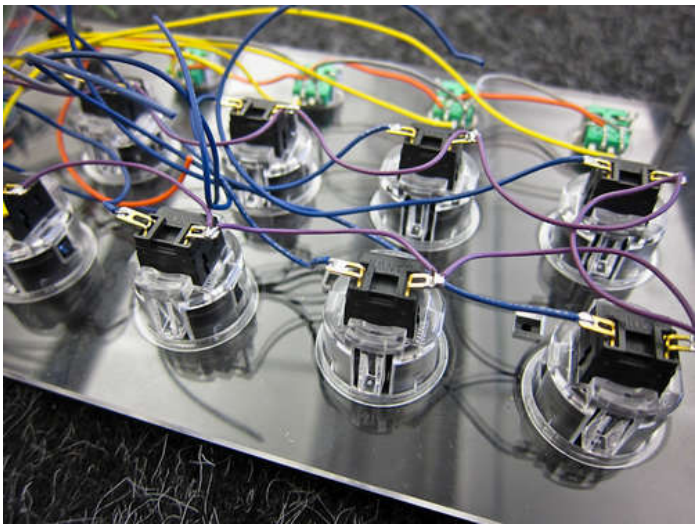


Image Notes
1. Ground/power connections for potentiometers grouped together using orange and grey wire.





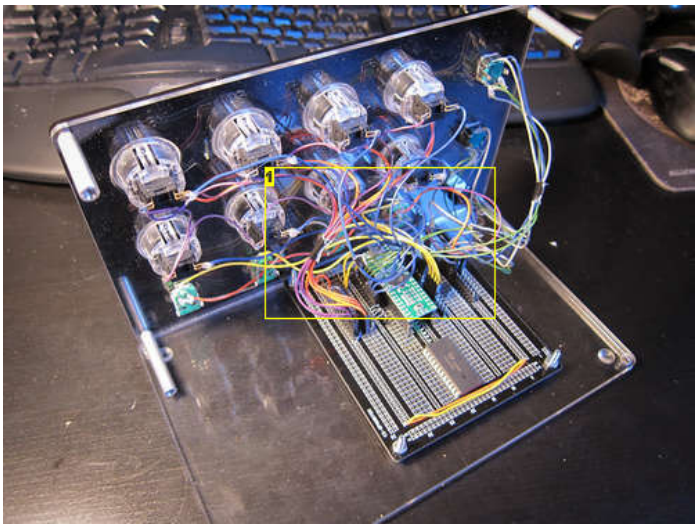


Image Notes
1. Wires are long enough so top panel can stand upright next to bottom.

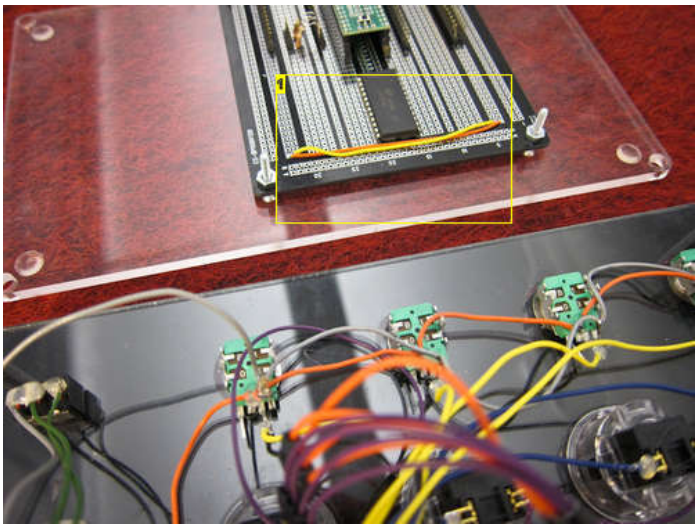


Image Notes
1. PCB mounted slightly off-center to avoid interference.

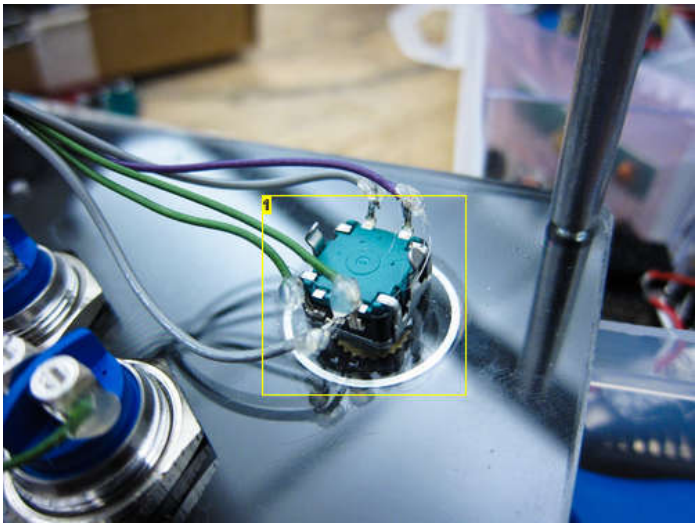


Image Notes
1. Rotary encoder with pushbutton.

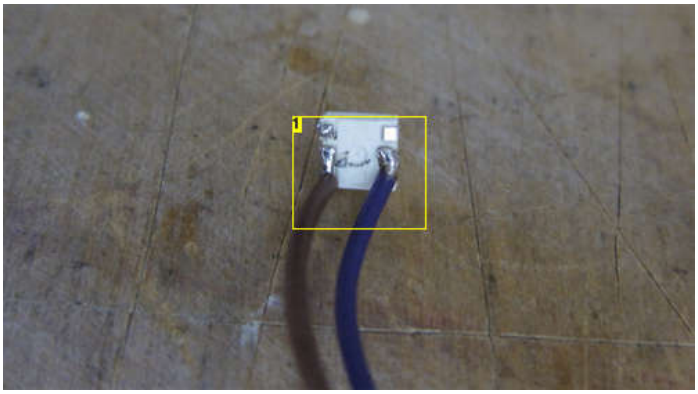
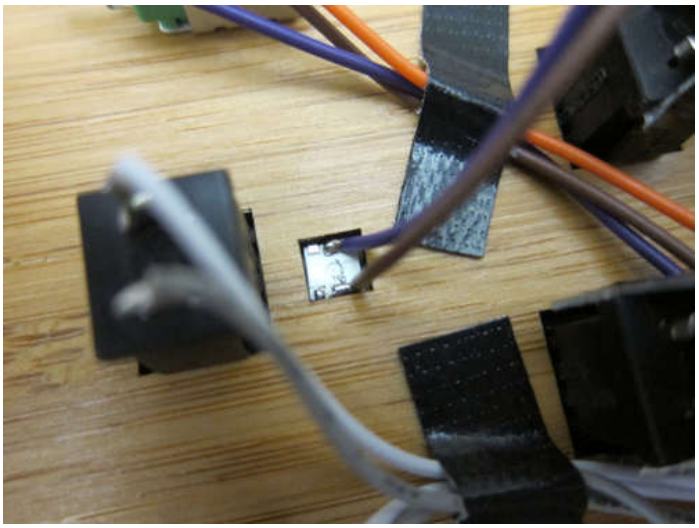


Image Notes
1. Surface mount LED.



Step 8: Assembling The PCB

Understanding the bare PCB
The first and most important thing to understand is which holes on your bare PCB are connected to one another. Notice how there are white lines/boxes outlining some of the holes. These outlines identify which holes are connected together. Although none of the holes look connected from the outside, the ones which are outlined by a white box are connected together internally. So if you solder any two connections within a box, think of this as soldering those connections directly together. Follow the +5V/ground connections from the Teensy++ board and notice how both the power/ground become connected around the entire perimeter of the board. The orange wires at the bottom of the board send power/ground from one side of the board to the other.

Header pins
You way also be wondering what all these header pins poking out are for. Well, it's simple, those pins are what we 'snap' our wires to using the plastic crimp housing. I have attached images for the full featured PCB. This PCB makes use of every Teensy++ pin, as well as the CD4067BE multiplexer. None of my modules used every Teensy++ pin (not even close). This board was built just as an example. For example, if you had 10 pushbuttons and 10 potentiometers you wanted to hookup, you would need a total of 50 header pins. Each pushbutton would require one header pin that connected to the board, and another one that connected to ground (total 20). Each potentiometer would require one header pin that connected to the board, one to ground, and one to power (total 30).

CD4067BE
Okay what the heck does this CD4067BE chip do? Good question. As you probably noticed from the pin configuration diagram the Teensy++ only has eight analog inputs. Yikes! In some cases (like my mixer) you will require more than eight analog inputs. The CD4067BE easily adds 16 analog input to your board. The four red wired linked between the Teensy++ and CD4067BE allow the Teensy++ to tell the CD4067BE which of the 16 pins it wants to read. The brown then sends the information for that analog input from the CD4067BE to the Teensy++. So if you require eight analog inputs or less, you will not need to add the CD4067BE to your PCB. That said, it is nice to have it on your PCB in case you later want to add more analog inputs. The CD4067BE chip only costs ~$1.50.

Resistors on the top right of the board.
This is the board I used for my mixer, which includes two LEDs on the interface. It is important you use resistors in series with your LEDs otherwise your LEDs will burn out. Different colors/types of LEDs require different resistor values usually around 200ohm-500ohm. Check online to make sure you have coupled the correct resistor with your LED. You will notice I soldered one end of the resistor directly to the header pin. This was a bit ugly. It is best to not include header pins where LEDs will be connected and instead solder these resistors directly to the PCB. Both work, the latter just looks a little nicer.

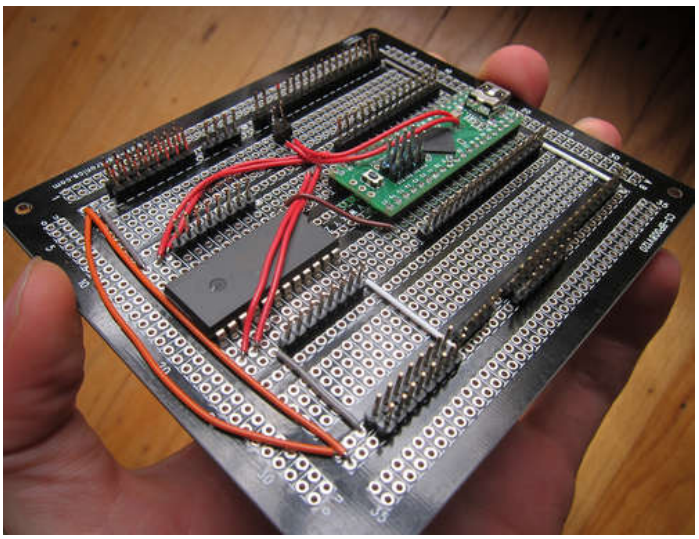
I2C module->module communication
In the top left corner we can see the connections required for I2C: power (orange), ground (grey), SDA (yellow), SCL (yellow), Even if you are only building one module I would still recommend adding these connections for future use. It is important to note that this is the I2C setup for the master module. The master module requires two 4.7 kilo ohm resistors connected between SDA/SCL and power. The other (slave) modules do not require these resistors, i.e. the slave modules look identical to this diagram except they are simply without the two resistors (see step 12 for more info).

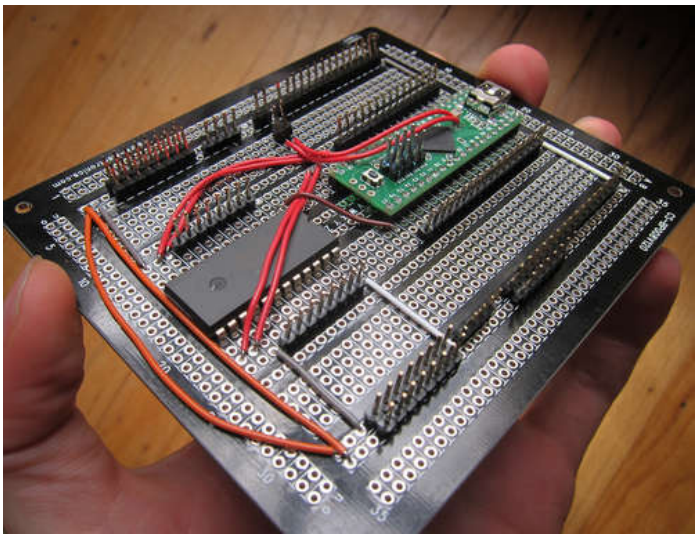
Teensy++ internal pins
You will notice eight headers and two wires coming out of the Teens++ board. This is because some of the pins on the Teensy++ board (for whatever reason) are located internally. We have to add these headers/wires so we can access these pins.

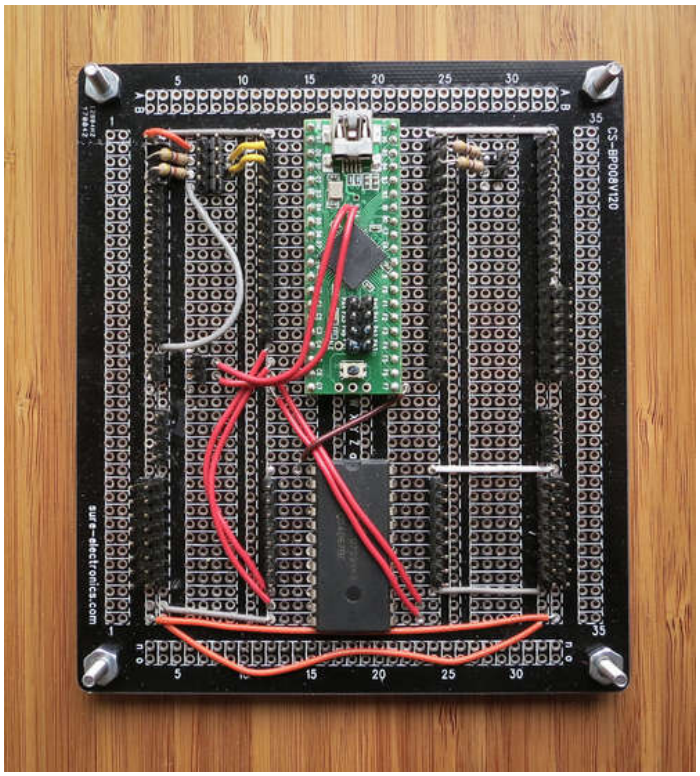
Snap in Teensy++
The last two images show an alternative way of connecting the Teensy++ to the PCB. If something breaks on your Teensy++ board it is very very very difficult to remove from the PCB. By carefully cutting the female headers to the appropriate length we can create a port that the Teensy++ snaps into. You could also do this for the CD4067BE. The disadvantage to doing this is that your height will increase. On my mixer I soldered the Teensy++ directly to the PCB (see images). On all my other modules I connected the Teensy++ via female headers.

Soldering to the PCB
The video (step 6) shows how to solder to the PCB. Ensure each connection has a nice cone of solder securing it. The solder should never bridge two holes/connections otherwise you may have some serious problems.

Okay that is all you need to know. Follow the diagrams carefully and build you board. You can build the fully featured board (exactly as you see it) or if you have a better idea what is going on here, add header pins only where you need them. In the latter case, ensure you have planned out where each component will connect to the board before soldering. Know what sizes of crimp housings you have available to help plan out your header layout; i.e. If you only have 1x4 crimp housings it would be foolish to solder five or six neighboring headers (read next step to better understand).







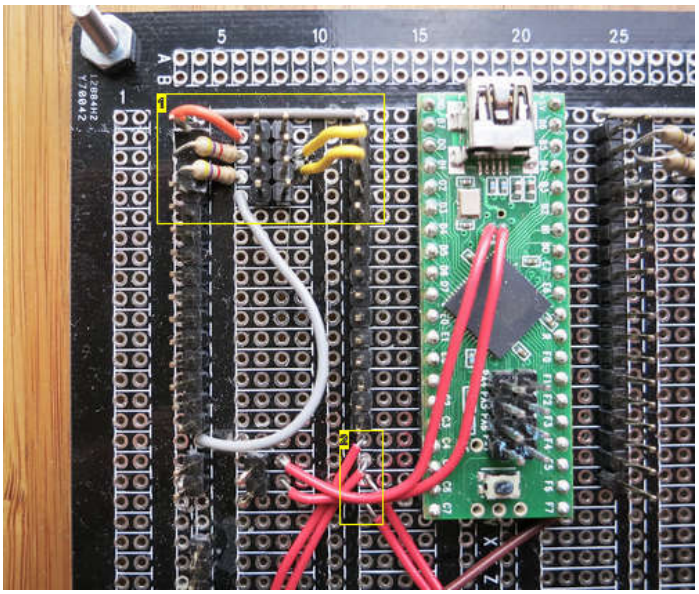


Image Notes
1. I2C module->module pins
2. Control pins for CD4067BE.

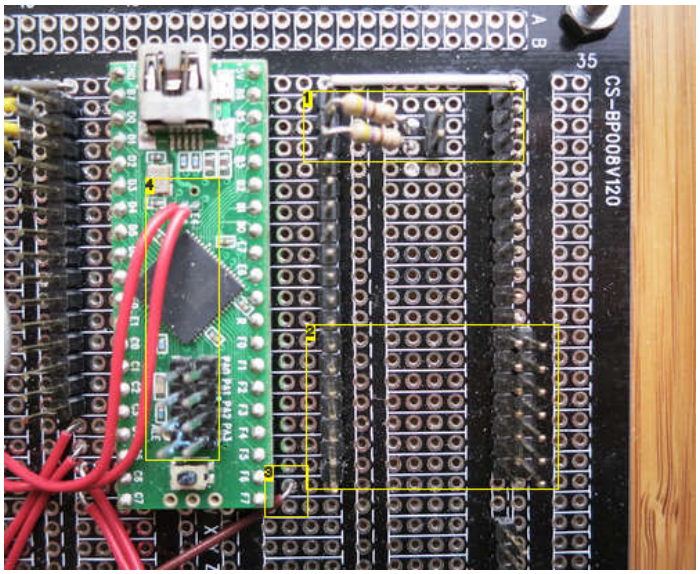


Image Notes
1. LED pins and resistors.
2. Teensy analog in pins
3. Read pin for CD4067BE.
4. Teensy internal pins.

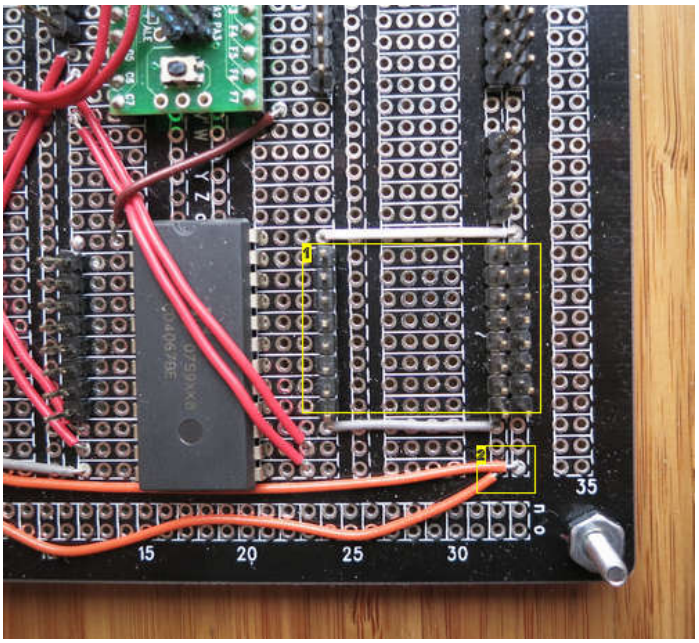


Image Notes
1. 8/16 analog in pins for CD4067BE.
2. Connection pwr/gnd from one side of the board to the other.

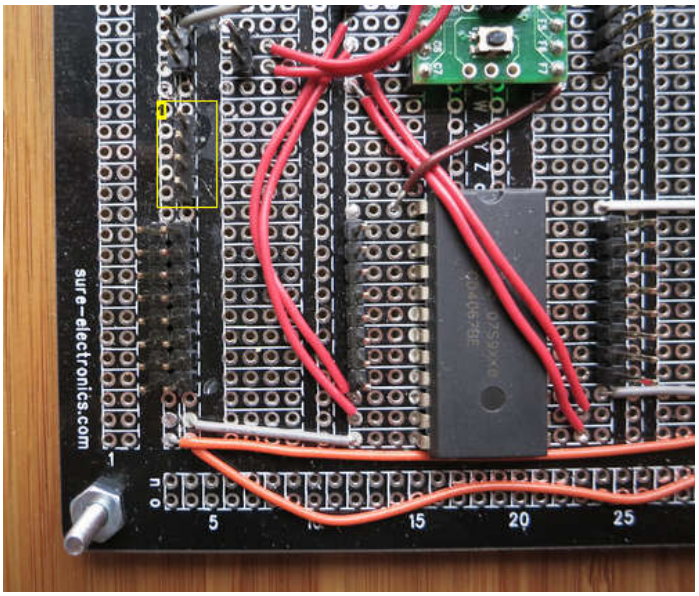


Image Notes
1. 4/8 ground pins for Teensy internal digital pins.

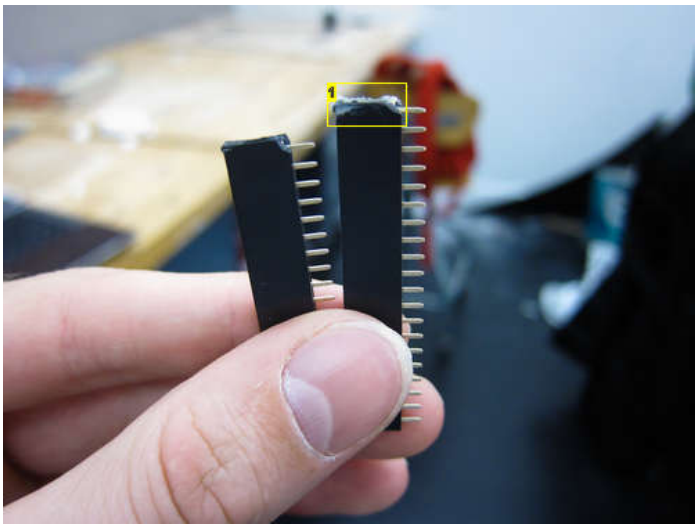
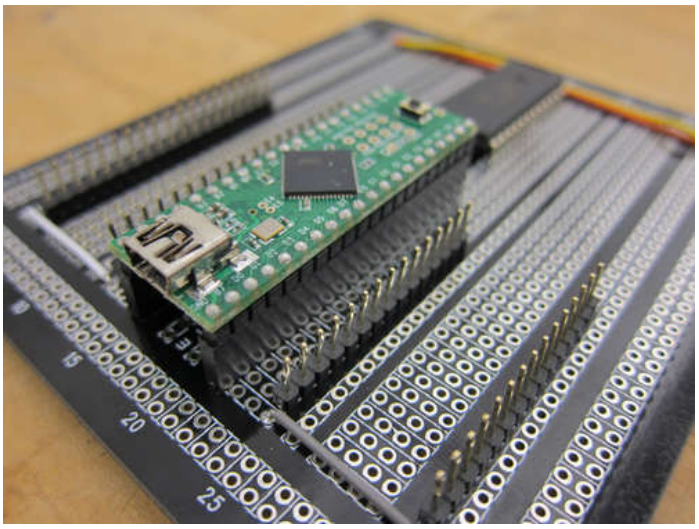
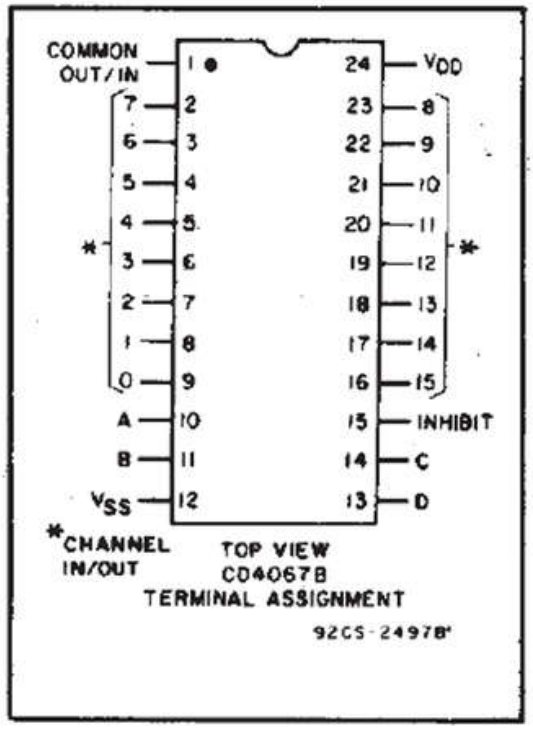


Image Notes
1. A little bit of glue gun glue added to keep the last terminal inside the plastic housing.



CD4067BE



Step 9: Connecting It Together

Alright now that you have built the top panel and the PCB it is time to connect them together.

First fasten your PCB to your bottom layer with some hardware. Note that I did not include the PCB mounting holes in my laser cutting files cause I knew I would want to specify the position of the PCB hereafter.

Hopefully you were wise and purchased many sizes of crimp housing connectors. You need to connect you crimp wire ends into the crimp housings. There is a specific orientation that causes the crimp connection to 'click' into the housing. Once the wire is connected properly to the housing you will notice that you cannot pull it out - perfect. In the case you have made a mistake and need to remove the crimp connection from the housing just tilt the tiny plastic flap in the front backwards and gently remove.

Now it is important you plug all ground/power/reading connections into different housings. Note the orientation of the ground/power/reading header pins on the PCB. If you understood step 4 it should become obvious how the crimp housings are to be created and connected. Using zip-ties or tape to group wires for certain types of components is highly recommended (see image).

Important you do not mix up the power/input/ground potentiometer connections when plugging them into your board. Since I did not color coat my wires very well I did this once. I plugged them incorrectly into my CD4067BE and smelt burning shortly after plugging in the Teensy++. I immediately unplugged the USB and luckily did not fry the chip - close call.

If you have many components you may just want to hookup and test a few at a time.

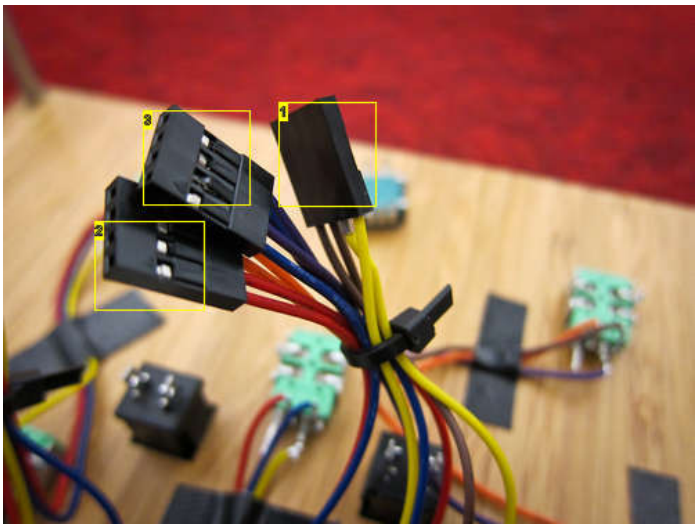


Image Notes
1. Analog reading housing.
2. Analog power housing.
3. Analog ground housing.

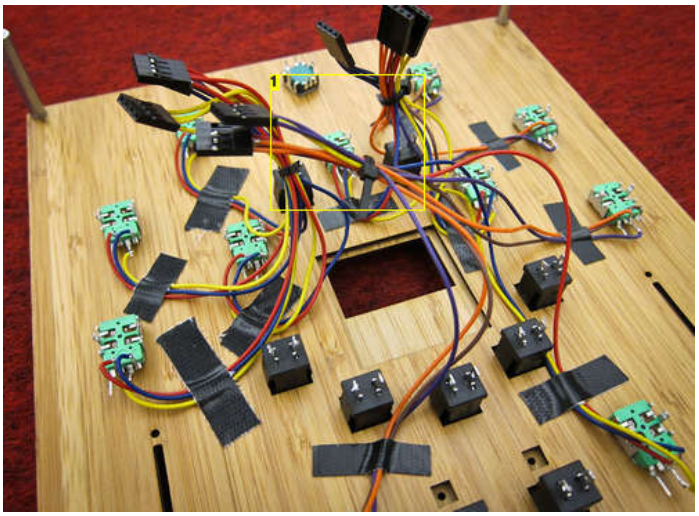
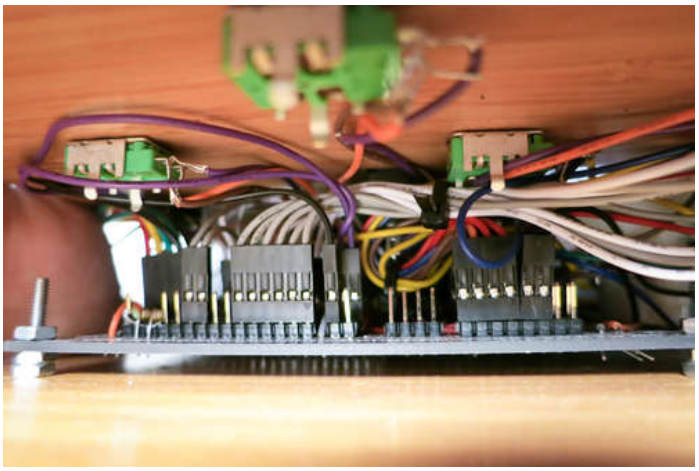
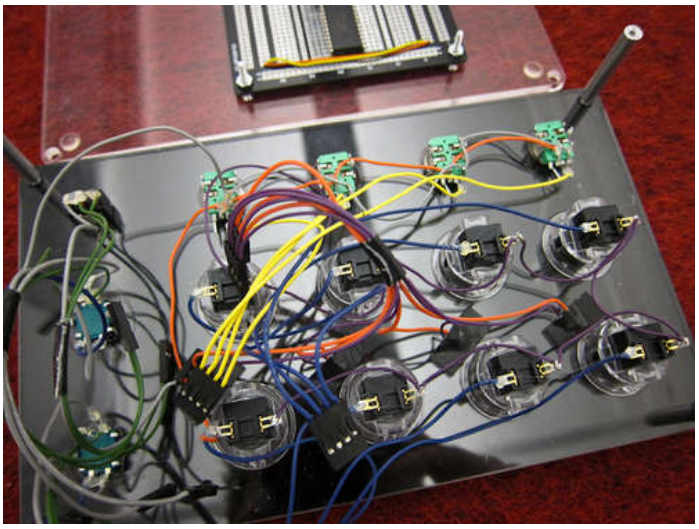
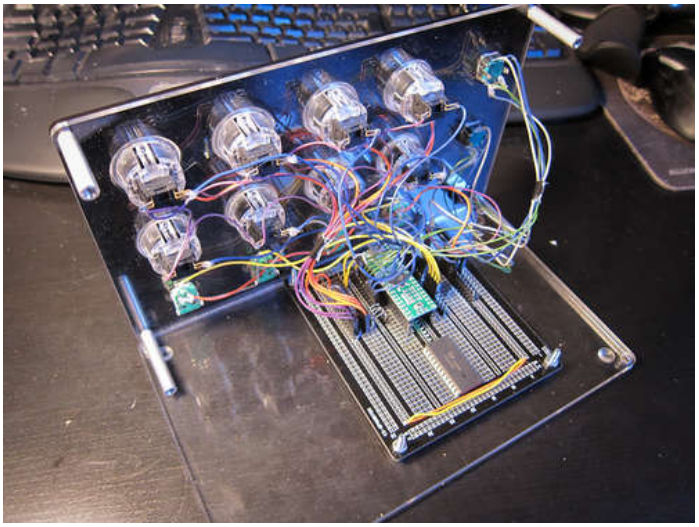
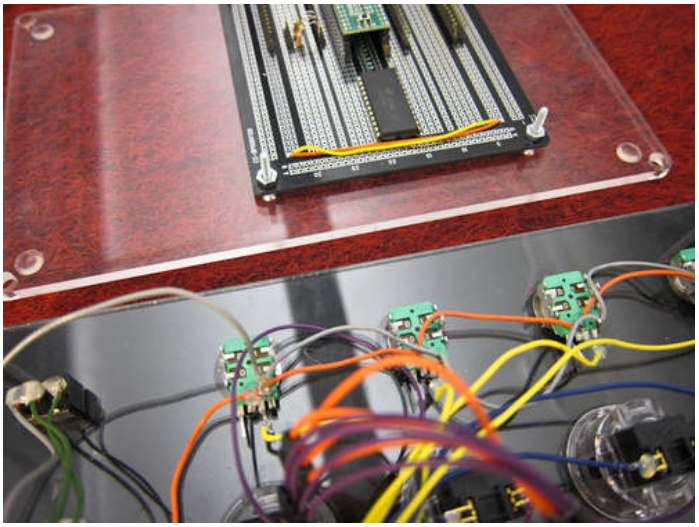


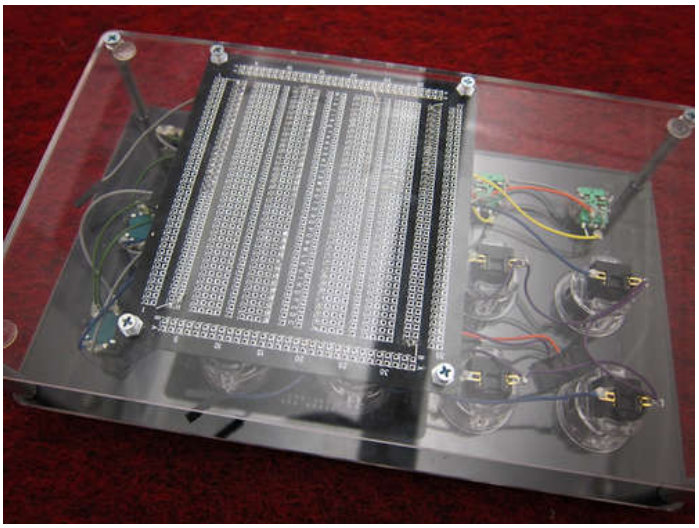
Image Notes
1. Three groups of potentiometers, all using 1x4 housings.











Step 10: Hey, What About The Jogwheels?

Jogwheels are nothing more than rotary encoders.

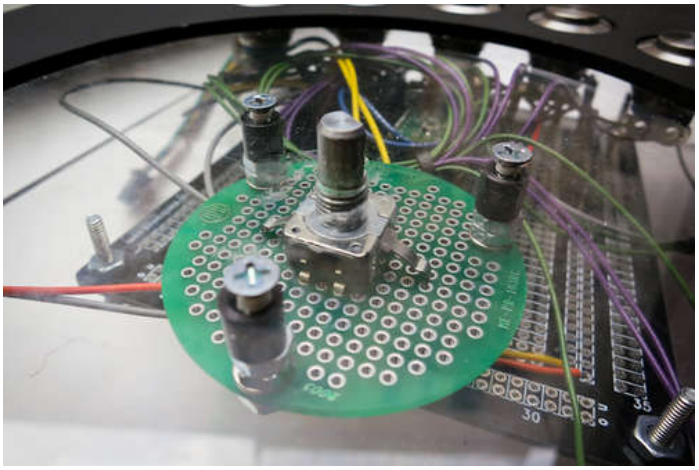
The only tricky party to the jogwheels is mounting them.You have to mount them to the underside of the top panel otherwise the lip/nut interferes when you try to spin them.

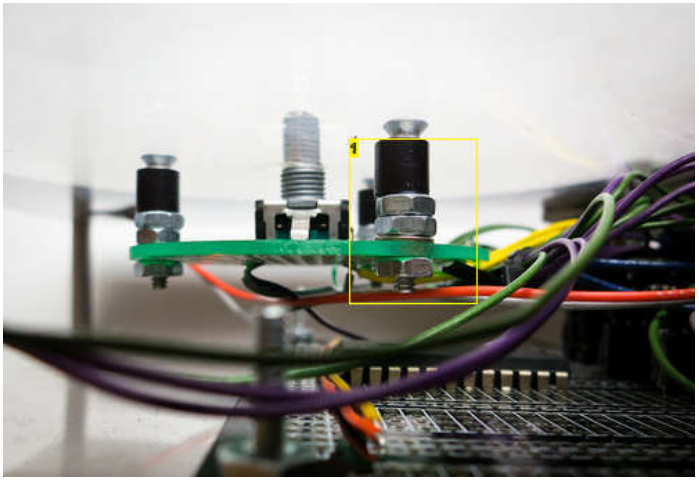
I found the best way to mount the jogwheels is by soldering the encoder to a mini PCB, then using the mounting holes on the PCB to secure the PCB/encoder to the top panel (see images). I cutout the hole in the top panel to be almost exactly the same diameter (6mm) as the shaft of the encoder. The cutout on the platter is a 'D' shape to match that of the encoder.

I did not include the cutouts for this mounting in the laser cutter file. I did it manually with a power drill. I suspect doing it on the laser cutter would be better.

Use felt feet on the underside of the platter to reduce friction.

MK1 used a similar design for the wooden jog wheels.





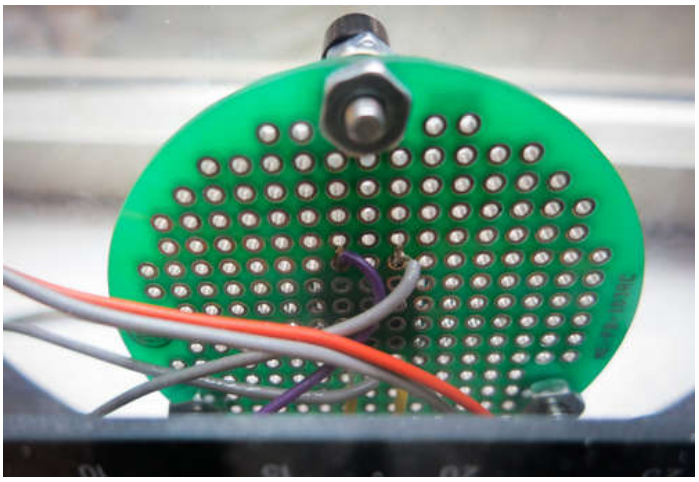
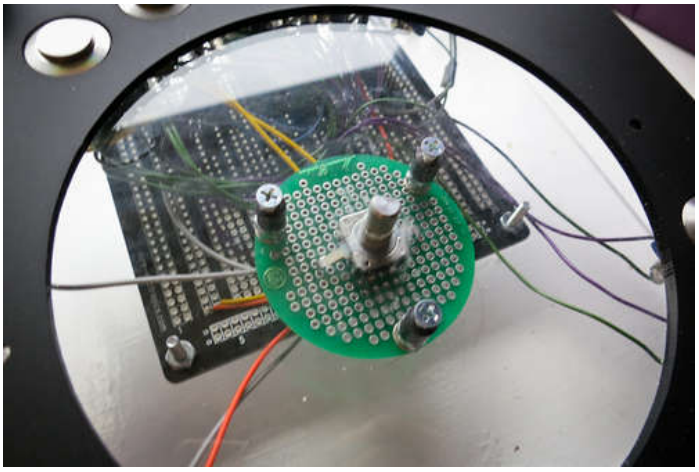


Image Notes
1. I need to get some proper spacers.



Step 11: All About The Code

I understand many people become puzzled and overwhelmed when trying to read code. I put a ton of work into making the code user-friendly. I was never trained as a coder, but I do consider this code to be pretty damn good! Thanks to my brother Neil for help with some of the more difficult sections.

The code includes **many** comments to help you understand what is going on.

If you have not been introduced to Arduino code before this Instructable it is good idea to read some beginner tutorials online before proceeding.

The code is broken up into sections (encoders, pushbuttons, LEDs, multiplexer, etc.). You are required to edit some parts of the code before the setup() and loop() functions. Each line which required edits has '// *edit*' written above it. These edits usually only require filling in a number '1' or '0' (kinda like fill in the blanks). You need to make edits which specify the following:

- Which pins have pushbuttons connected to them
- Which pins have LEDs connected to them
- Which pins have analog inputs connected to them
- Which pin is the shift button
- Which encoders are in use
- Should debug be enabled - this allows you to debug your code in the serial monitor
- Should I2C be enabled - this allows for module->module communication
- The address of this module - each module must have a unique address
- The master address - just leave this as '0'
- The channel of this module - each module must have a unique channel number

You will also need to write custom code which governs the behavior of your LEDs.

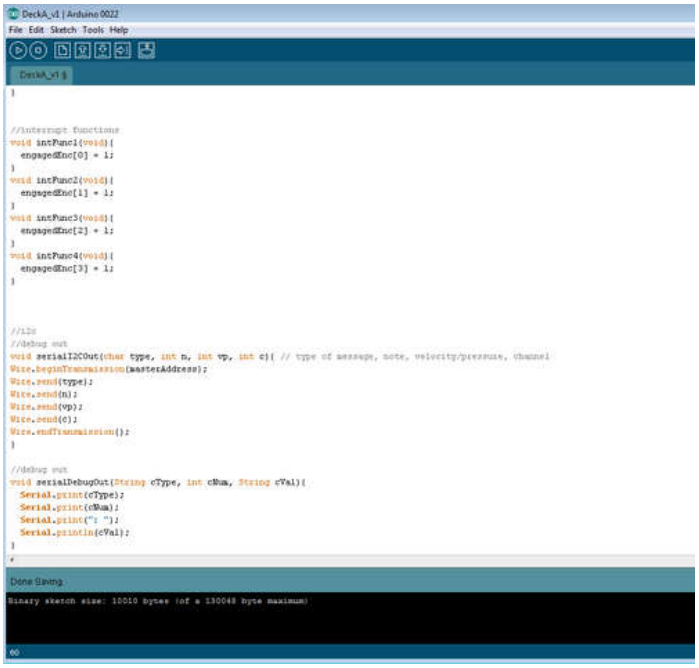
Although it is not imperative to understand all the code, I just want to explain one snippet of code which will help you understand how the control messages work. The code below sends analog messages for a potentiometer. There are three modes (debug, I2C, and MIDI) which you turn on/off in the edits. If debug is enabled, messages will be sent through USB to the serial monitor to help with debugging. 'serialDebugOut' is a very basic function I wrote which sends these serial messages and it can be viewed at the bottom of the code. If I2C is enabled, messages will be sent through I2C to the master module which will then forward the messages as MIDI through USB. If MIDI is enabled, then this controller will send MIDI messages directly through USB to the computer. 'usbMIDI.' is a specific message used by Teensy++ for MIDI messages. Read more about Teensy MIDI here . Keep in mind you can edit the code to send many kinds of messages: Teensy USB Serial , Teensy USB Keyboard , Teensy USB Mouse , Teensy USB Joystick , Teensy USB MIDI , and Arduino Serial.

```
if(enableDebug==1){ //SERIAL debug is on
serialDebugOut("Analog",i,tempAnalogInMap);
}
else if(enableI2C==1){ //I2C
serialI2COut('a',i,tempAnalogInMap,channelNumber);
}
else{ //MIDI
usbMIDI.sendControlChange(i,tempAnalogInMap,channelNumber);
}
```

Read the code/comments over and over, it make take a few iterations to wrap your head around. You can learn a lot from the code. Constantly reference the Arduino library for sections you don't understand.

The rotary encoder code is a nasty beast. If you are feeling ambitious you can read more about it here .

Likely there will be some improvements made to the code over time so continue to check back for the latest version.



File Downloads

http://www.instructables.com/id/A-Framework-For-Making-Affordable-Stylish-Modula/

ControllerCode_v1.pde (14 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'ControllerCode_v1.pde']

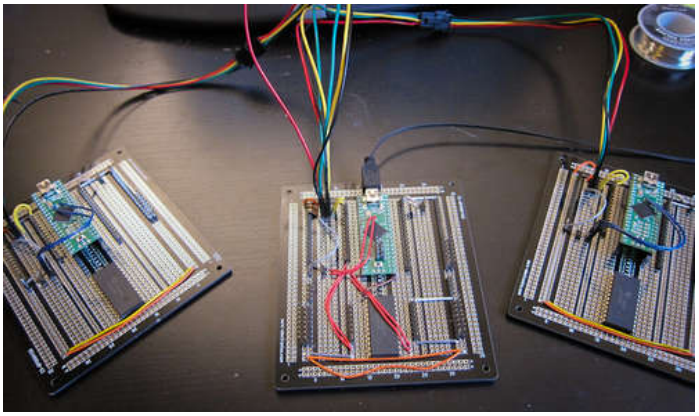
Step 12: Linking Module Together With I2C

To enable module->module communication you simply connect the modules together and enable I2C-mode in the code. Only your master module will plug into the computer via USB.

DO NOT plug the modular connectors in backwards to the PCB or bad things may happen. Ensure your red cable is lined up with the power and the black cable is lined up with ground at all times.

Below is a video showing module->module connections between three modules (one master, and two slaves).

You can link as many modules together as you like. As you can see from the video, each slave module had a free I2C slot for attaching more module in series.



Step 13: Testing Your Device

This section is a bit tricky because the modules can operate in one of three modes: serial (debug) mode, MIDI mode, or I2C mode. If you turn one mode on, you must disable the other modes. It is not recommended to jump straight into MIDI mode because it is impossible to debug your controller this way.

If you are using only one module here are the steps.

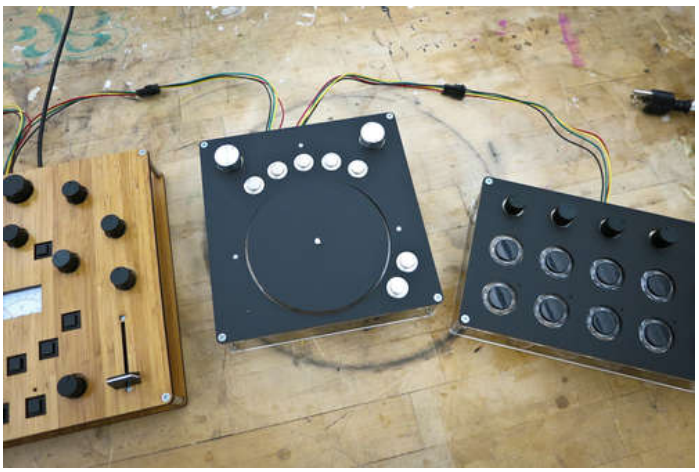
- Enable debug mode in the code.
- It is critical that you comment out **ALL** lines of code which have 'usbMIDI' in them otherwise the code will not compile (annoying, but necessary).
- Select 'serial' device from the Arduino tools->USB type menu.
- Upload your code and open your serial monitor. Ensure the connection speed in 9600bps.
- If you serial monitor is getting flooded with output you have hooked something up incorrectly or have improperly edited to the code to identify which pins are in use. Like I said earlier, it is best just testing one set of components at a time as opposed to all of them at once.
- If your serial monitor is working correctly you should see some output when you interact with the components. This output also informs you which mode number is associated with each pushbutton (so you can change the modes later if necessary).
- Once everything is working in the serial monitor you can now proceed to MIDI.
- Disable debug mode in the code.
- Select 'MIDI' device from the Arduino tools->USB type menu.
- Uncomment all the 'usbMIDI' lines you commented out earlier.
- Your controller is now be ready to send MIDI messages, and your computer should recognize it as a native MIDI device (as easy as that).
- Now I would not recommend jumping directly to hookup with your software, rather, download a free MIDI sniffer (there are many out there) and test your MIDI messages in that first.
- The sniffer, as well as your software, should recognize the device as 'Teensy MIDI'.
- Once you are happy with the MIDI output, your ready to roll.
- You may notice that as you switch between serial and MIDI modes that the code may not want to upload the first time. Only on the consecutive attempts will it manages to upload. I am not sure why this is.

If you are using more than one module here are the steps.

- Follow the one module steps above for each module (without them linked together).
- Enable I2C mode on each of the modules except the master module. Never enable I2C on the master module.
- Again, you need to comment out all 'usbMIDI' lines.
- Enable debug mode on the master module. So the master module will be running in serial mode and all other modules will be running in I2C mode.
- Connect all modules together.
- Check the output in the serial monitor. Messages sent from your other modules should now appear in the serial monitor with prefixes '(I2C)'.
- Now enable MIDI mode on only the master module.
- Uncomment all the 'usbMIDI' lines on your master module which you commented out earlier.
- You don't need to make any changes to your other modules, only the master.
- Now, test your MIDI output in the MIDI sniffer - voila.

Alright, this whole operation is a bit convoluted, but once you go through it one time you will understand what is happening here. It is surely annoying you have to continue commenting/uncommenting the 'usbMIDI' lines but unfortunately unavoidable.

One thing I have noticed about DIY electronics over the years is they **never work the first time** . There is always a small soldering blunder or connection error along the way. Don't get discouraged if things don't run so smoothly during the first attempt.

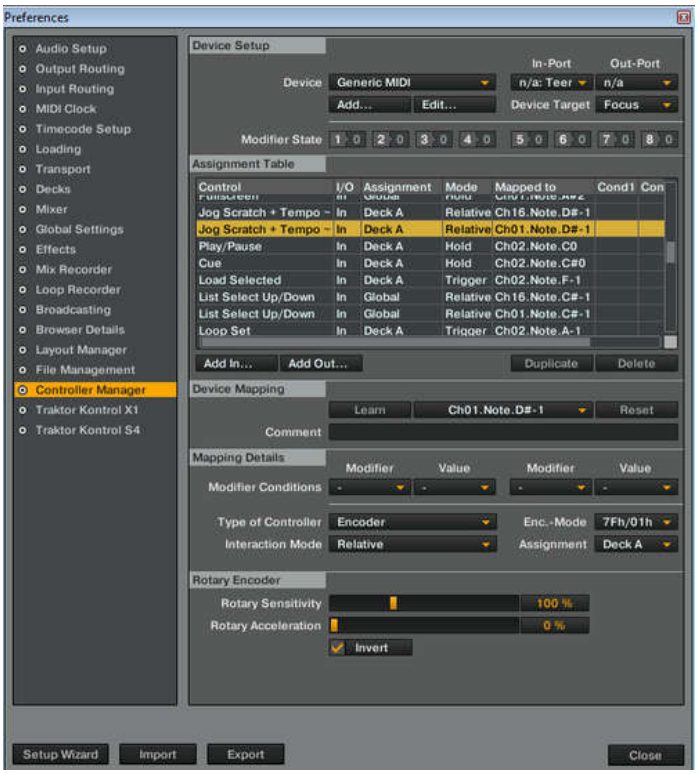


Step 14: Connecting To Software

You could connect these controllers to Ableton, Max/Msp, Traktor, custom software, etc.

I have attached my mapping file for Traktor II Pro. This file will only work for my device but you can load it into Traktor and get an idea of how I mapped my encoders, potentiometers and pushbuttons. Traktor has included many amazing features that make custom mapping a breeze - my hat goes off to the brilliant team at Native Instruments (who developed Traktor).

Since I have no idea what software you will be using your controller with I can not offer much help. You will need to rely on other online resources to help you connect your controller to your preferred software. It is important you check that the software you plan to use has custom mapping options available **AND** documentation (some do not).



Step 15: Outro

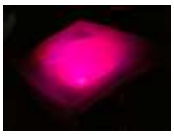
There you have it! Although it only took me a few days to create this Instructable, the design/testing for this projects has been ongoing for about 8 months. I was never trained in electronics/programming so there was a great deal of learning involved to reach this goal. Great fun. Great project. Onto the next.

Thanks to PJRC , NI and Ponoko for your excellent products and support.

Learn, make, think, repeat. Bridge the esoteric gap.

Related Instructables

Make an Amazing MIDI Controller by Fuzzy-Wobble



Arduino/Ableton Color Organ; MIDI controlled (video) by _Aias

Halloween Beat 2009 (video) by _Aias

How to Create a MIDI Map for the BCD3000 in Traktor Pro by cheft

Arcade Button MIDI Controller by fraganator

Cigar Box MIDI Controller by highly_liquid