instructables

How to Make a USB Laptop Keyboard Controller

by Frank_Adams

This is a snapshot of the original Instructable which was based on the Teensy LC and 3.2. These microcontrollers are now obsolete and difficult to find. Per PJRC, use the Teensy 4.0 or 4.1 for new designs. The LC and 3.2 board files and software discussed below are still available at my GitHub repository along with new board files and software for the Teensy 4.0 and 4.1.

This Instructable will provide a step by step procedure for building a USB laptop keyboard and touchpad controller. I created this guide and video to hopefully make it easier for people to re-purpose an old laptop. A typical laptop keyboard relies on the motherboard for the keyboard scanning and touchpad interface. I use a Teensy microcontroller mounted on a connector board to take over this function. Teensies are often used by the mechanical keyboard enthusiasts at Geekhack and Deskthority and the TMK or QMK software is the most popular controller code. The TMK/QMK code is a bit of an overkill if you just want a simple USB keyboard but it will certainly provide all the features you could ever need. If you would rather write your own keyboard software using Arduino, the Teensyduino functions give you total USB control. Whatever software you decide to use, it will require a key matrix that maps out how your keyboard is wired. One approach, (that I never want to do again) is to exhaustively check every connector pin combination with an ohm meter while holding down each key. I did this when I converted a Sony Vaio into a Raspberry Pi laptop. An Instructable from alpinedelta disassembles the keyboard in order for the connections to each switch to be visually traced back to the connector. Instead of taking the keyboard apart or using an ohm meter, this Instructable will load the Teensy with an automated continuity tester. The Teensy will report over USB, the two pin numbers that are connected when you press a key. After every key has been pressed, the results can be transferred to a row-column matrix and used by the TMK/QMK keyboard controller software or a home-brew Teensyduino routine. The touchpad code described in step 22 can be added to the keyboard routine to create a composite USB device.

I will include download buttons for many of the files in the Instructable but all the files are located at my GitHub repository**.** The best way to download files from a GitHub repository is to navigate to the top level of the project (USB_Laptop_Keyboard_Controller in this case) and click the green "Code" download button on the right. Choose the Download ZIP option from the Code pull-down menu. The ZIP file will contain the entire repository content, not just the code but also the board layouts and PDFs.

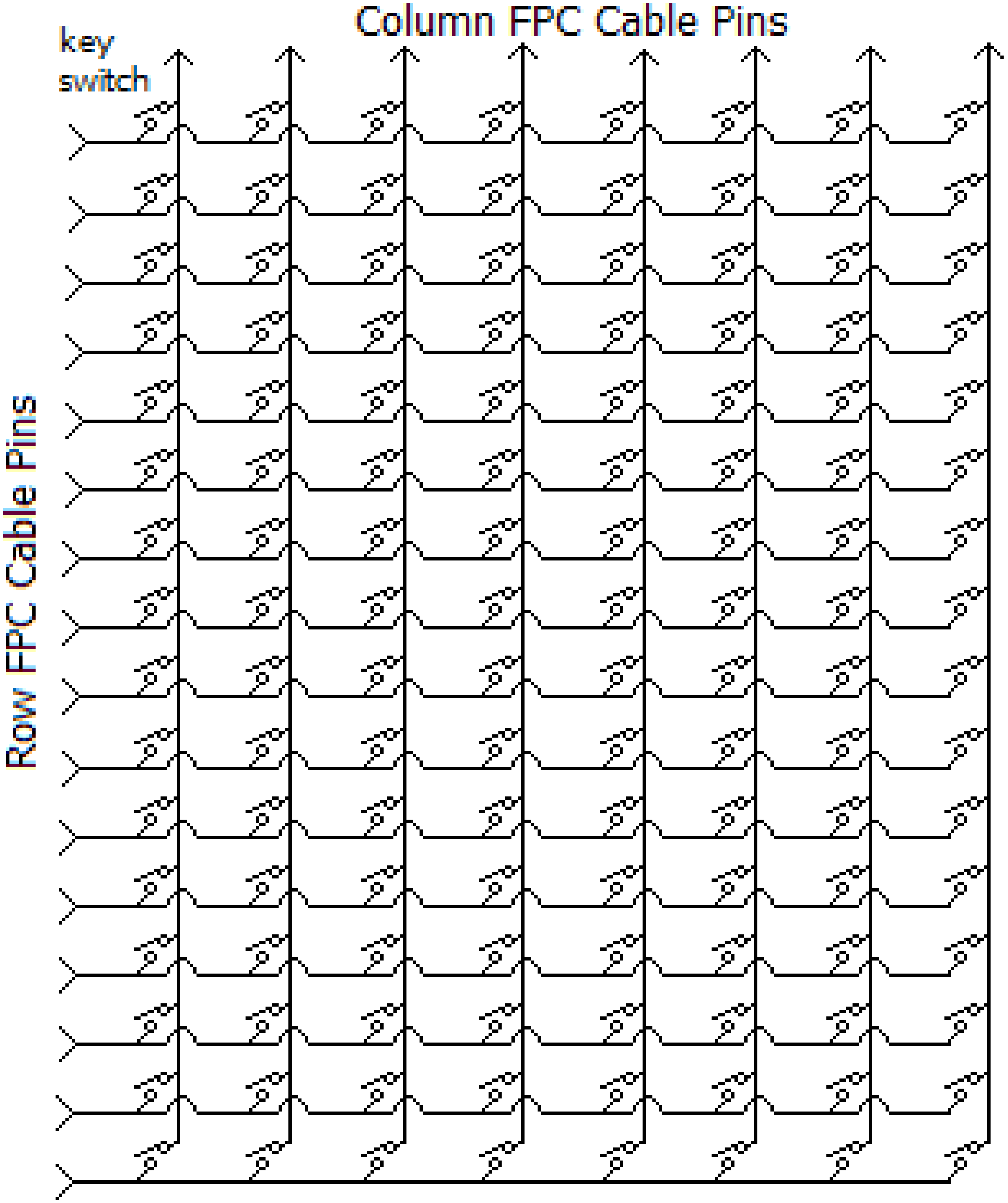
https://vimeo.com/458595950

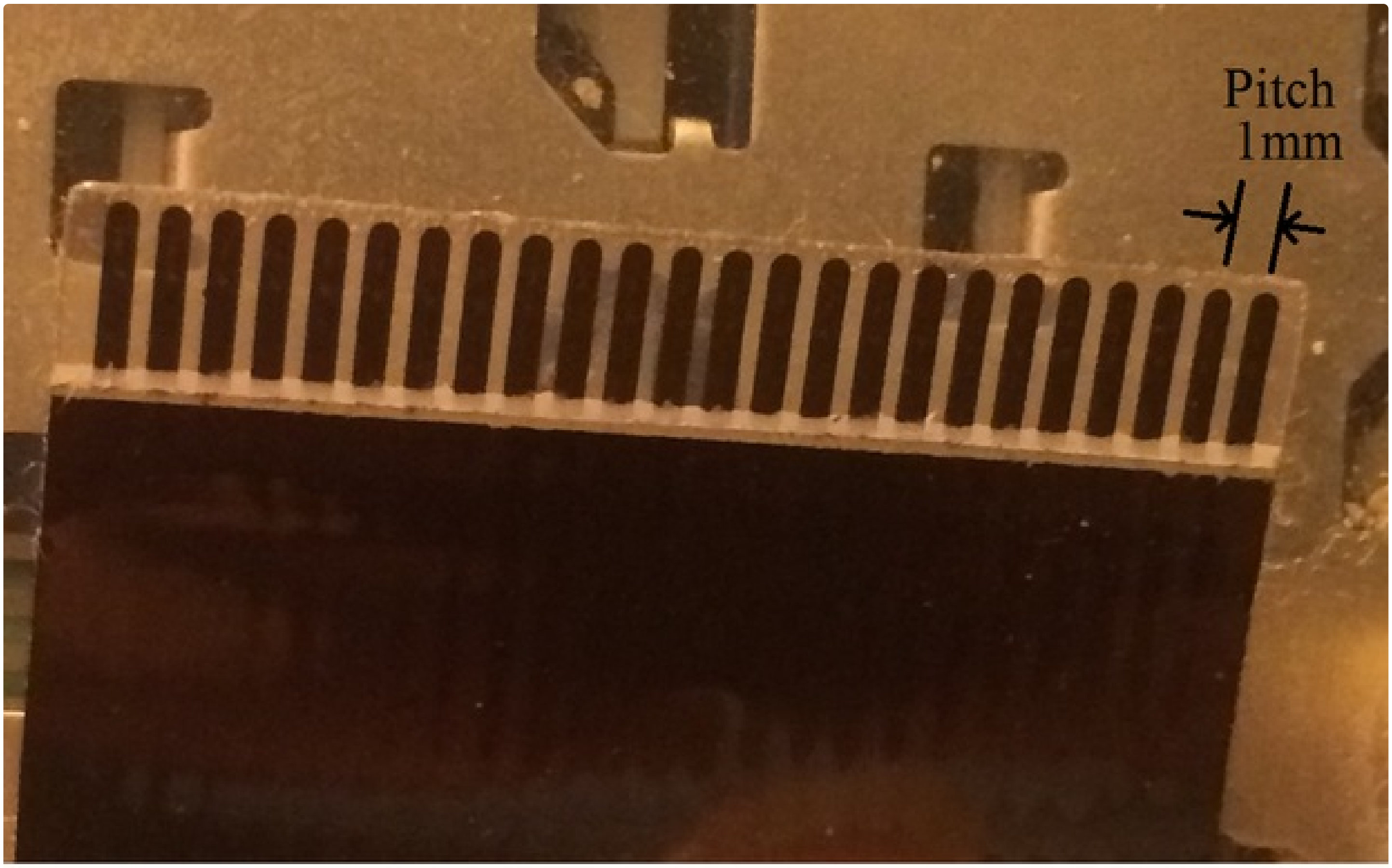
Step 1: Keyboard Cable Specifications

Laptop keyboards use a flexible printed circuit (FPC) that connects all the key switches in an array of rows and columns as shown above. The keyboard controller drives each row low, one at a time while reading the columns (with a pull up resistor) to see if any switch is pushed. The two bins of laptop keyboards shown above are from Re-PC, a local recycling store. About 75% of the keyboards have FPC cables that end with exposed metal traces on one side and a plastic backing on the other side. The plastic backing plus the FPC material typically measures about 0.30 mm thick. A typical keyboard without a number pad has 24 or 25 signal traces with a 1 mm pitch. If there is a number pad, then it's common to have 26 traces with a 1 mm pitch. It can be difficult to measure the pitch between traces so another method is to measure the width of the entire FPC cable and use this formula:

Pitch = Total width / (N + 1) where N is the number of pins

A few of the keyboards at Re-PC had 28 to 34 traces and some had a 0.8mm or 0.5mm pitch. There were also keyboards with dual FPC cables. I have included support for all of these variations. Connectors for keyboard cables are readily available from companies like Aliexpress, Mouser, or Digikey. The number of signal traces, the pitch, and whether the contacts are on the top or bottom are the parameters you will need when ordering. There were some old keyboards in the bins with rigid printed circuit board connectors and some other keyboards with specialized connectors soldered to the end of the FPC cable. These keyboards will not be the focus of this Instructable but a few examples will be given.

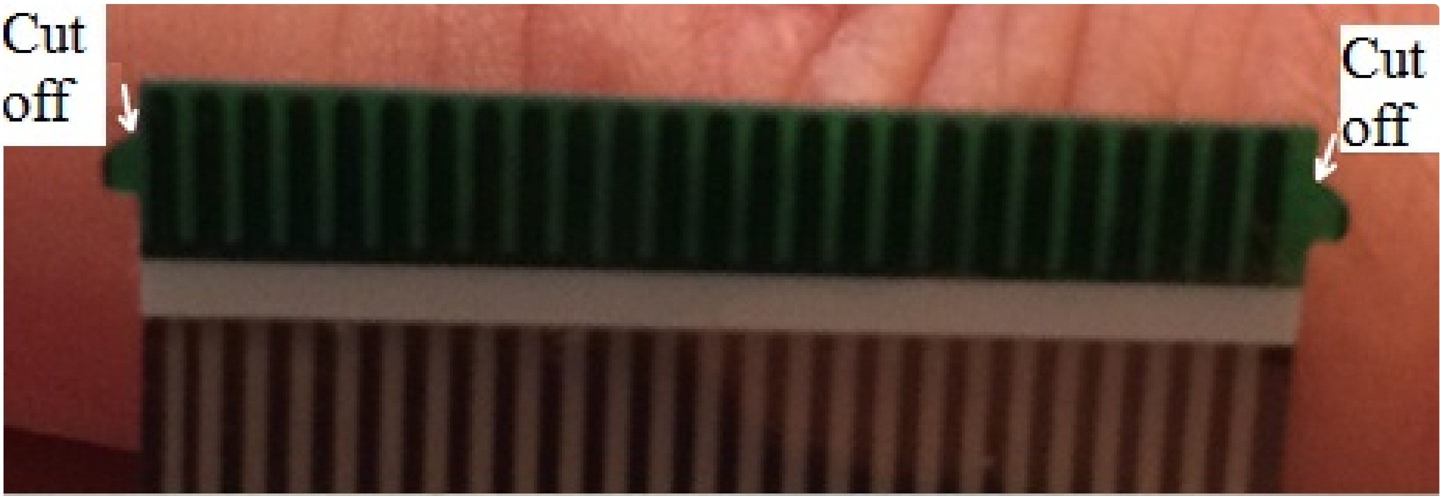

Column FPC Cable Pins
key
switch
Row FPC Cable Pins

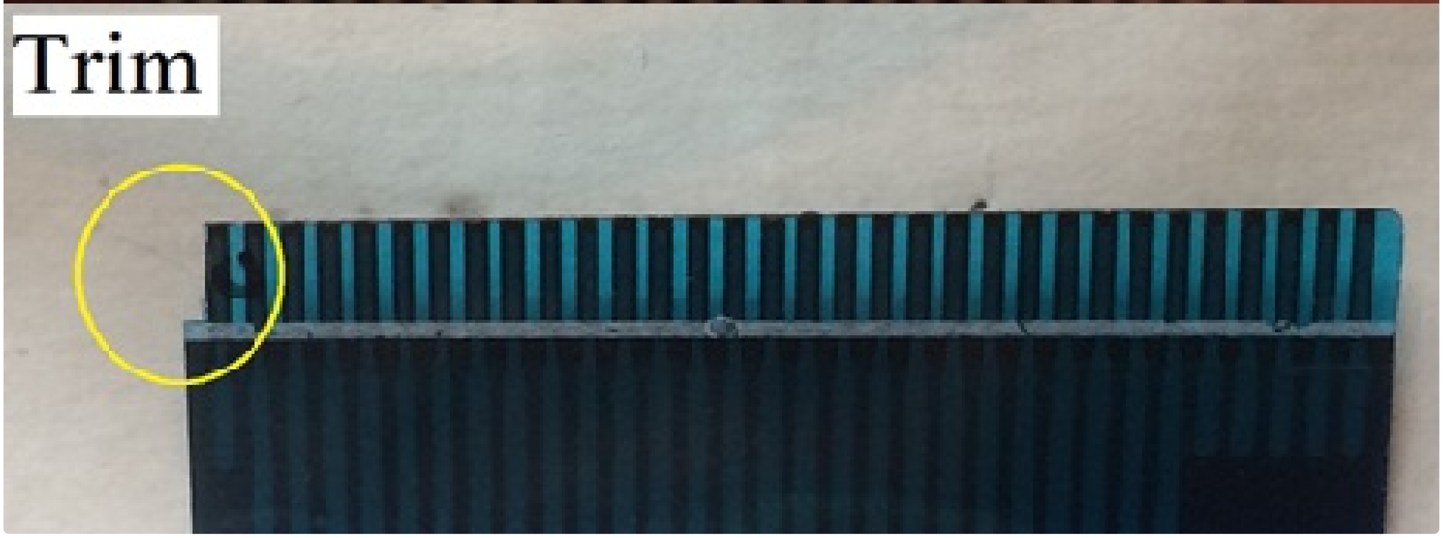


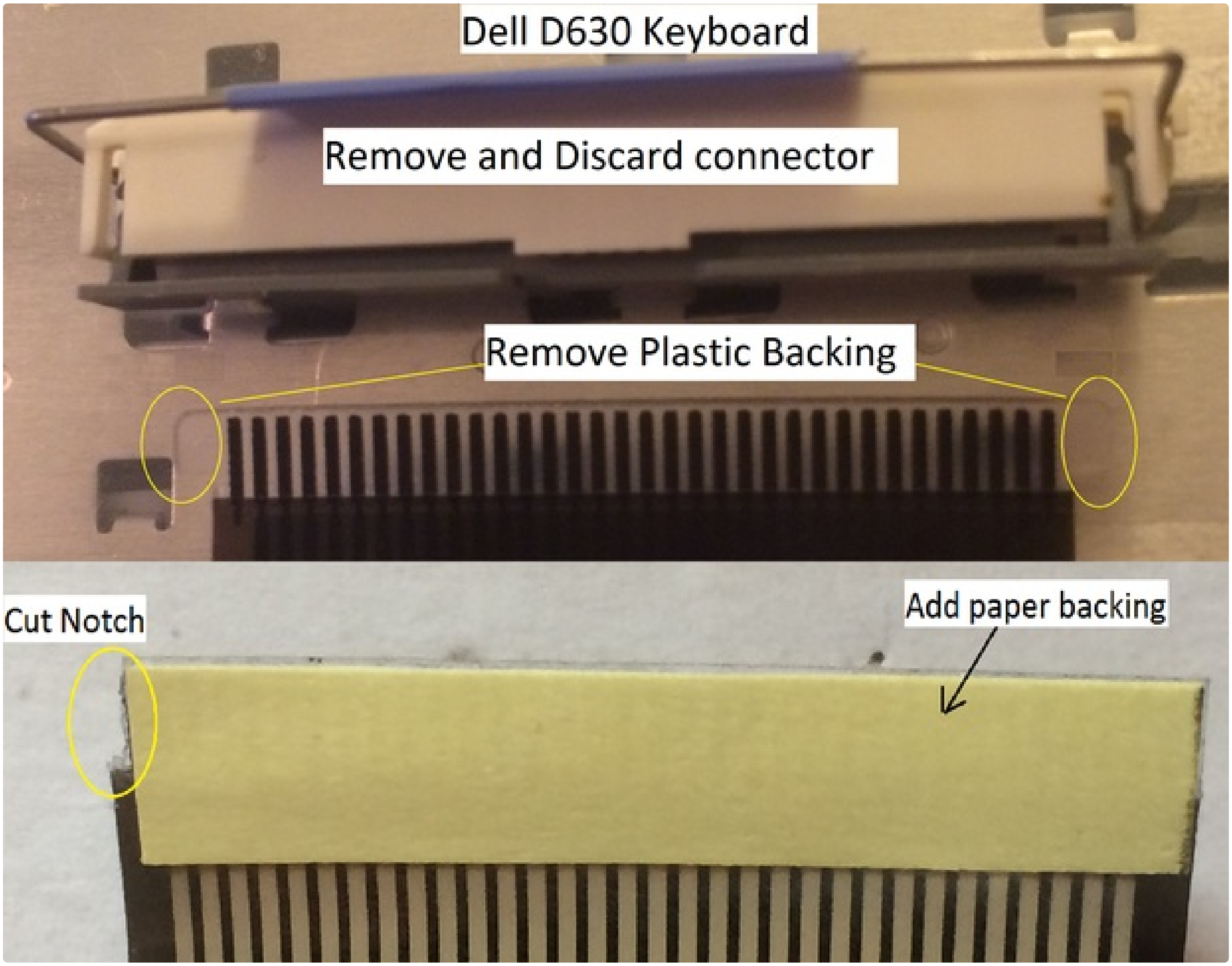
Standard 24 Pin FPC Cable with a 1mm Pitch

Step 2: Modify Your FPC Cable As Needed

Some FPC cables need to be modified to fit in a generic connector. Locking nubs on the side of the cable are easy to remove with scissors. If the FPC traces don't line up with the connector pins, use an X-ACTO knife to trim along the side of the cable. The Dell Latitude D630 keyboard needed the most modifications. It had a solder-less connector on the end of the FPC cable that was easily removed. Then I pulled off the extra thick plastic backing that was glued on the end of the cable and cut a notch on the side to align the contacts. To bring the thickness back to normal, I glued 2 pieces of paper to the end of the cable.


Cut off
Cut off

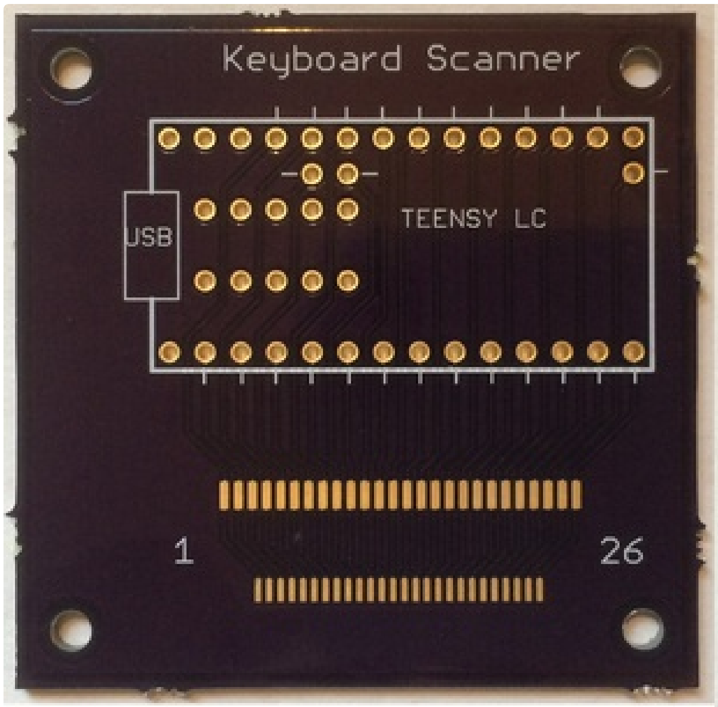

Trim



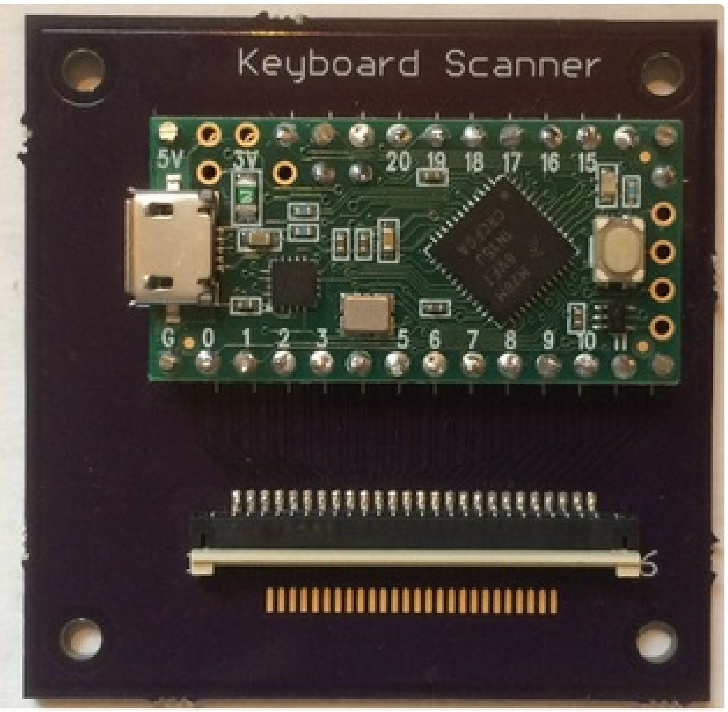
Step 3: Teensy LC FPC Connector Assembly - Surface Mount or Thru-Hole

I designed a circuit board using Eagle for the Teensy LC that routes its 26 I/O pins to 26 surface mount pads for an FPC connector with a 1mm pitch or a 0.8mm pitch. I created a second circuit board for keyboards that have a 0.5mm pitch. For those that don't want to solder surface mount connectors, I created the board shown on the right for thru-hole FPC connectors. A connector with up to 26 pins can be soldered to these boards based on your needs. For connectors with less than 26 pins, solder starting at pin 1 and leave the unused pads at the other end blank. I avoided using the 27th Teensy LC output because it's connected to an LED and 27 pin FPC connectors are rare.

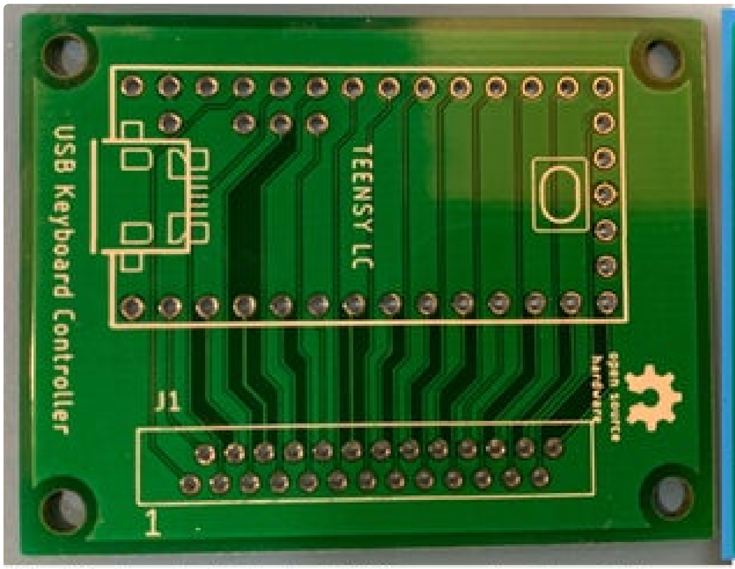
After soldering the FPC connector to the board, I soldered 4 header posts to the board to support the corners of the Teensy and then I soldered the Teensy to the header posts. The last step was to connect the rest of the Teensy I/O signals to the board with 30 gauge wire. I used wire instead of header posts to make it easy to reroute I/O pins or cut the Teensy off the board. If you buy a Teensy LC with pins, it will not have pins on I/O's 24, 25, and 26 so you'll need to add them.



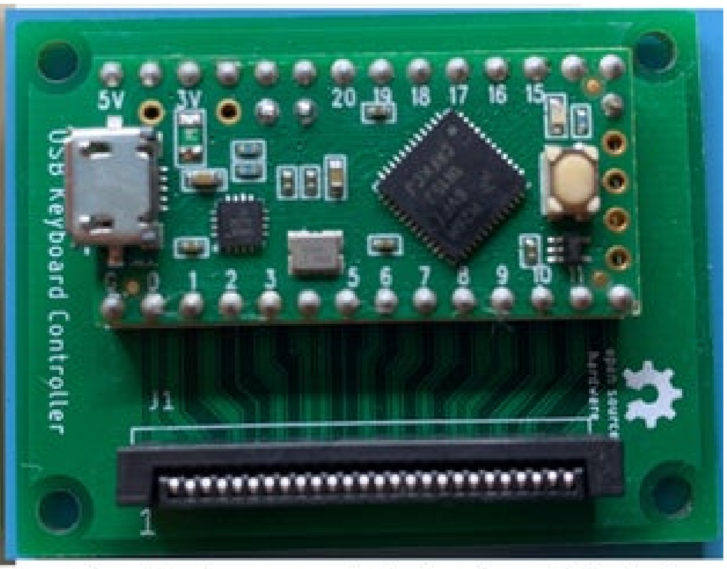
Teensy LC Side of Circuit Board



Teensy LC and FPC Connector Assembly



Use this board with thru-hole FPC connector, 1mm pitch, up to 26 pins. Install the connector with the contacts facing front or back depending on how the FPC cable is oriented.



Board has been assembled with a Teensy LC with pins plus pins added for I/O's 24,25, and 26.
26 pin FPC vertical connector Amphenol HLW26S-2C7LF

Step 4: Teensy 3.2 or 4.0 FPC Connector Assembly

I designed the back side of the board for a Teensy 3.2 with 34 I/O signals and an FPC connector with a 1 mm or 0.8 mm pitch. A different 3.2 circuit board is available if you have a 0.5mm pitch.

Whichever side of the board you mount the Teensy on, you must solder the FPC connector on the same side.

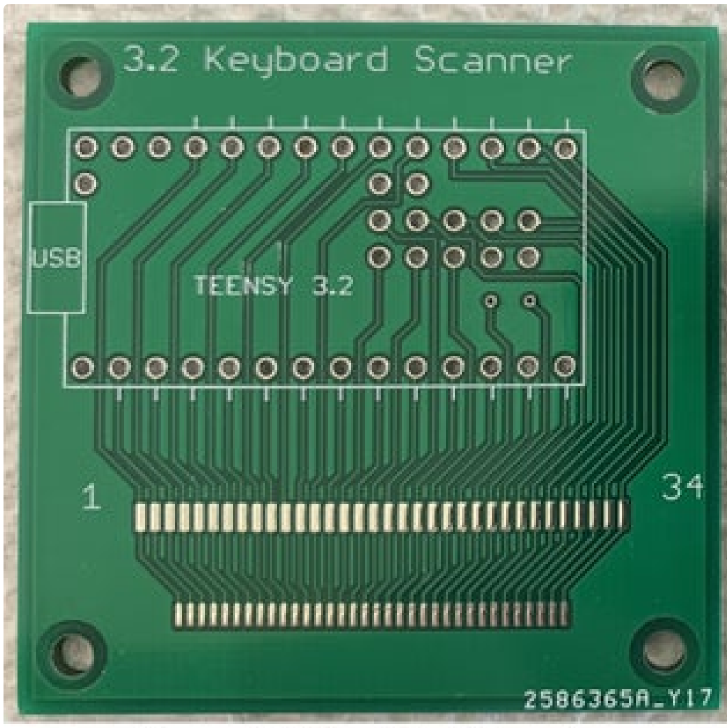
*****Update for Teensy Availability********

PJRC has changed the status of the Teensy LC and 3.2 to "Not Recommended for New Designs". They feel the 90nm process used for these older ICs will never recover from the chip shortage so there will always be a low stock and inflated prices. The Teensy 4.0 is made with a more modern 45nm process so there are boards for purchase at a reasonable price. I recommend you use the 4.0 for all future keyboard projects.

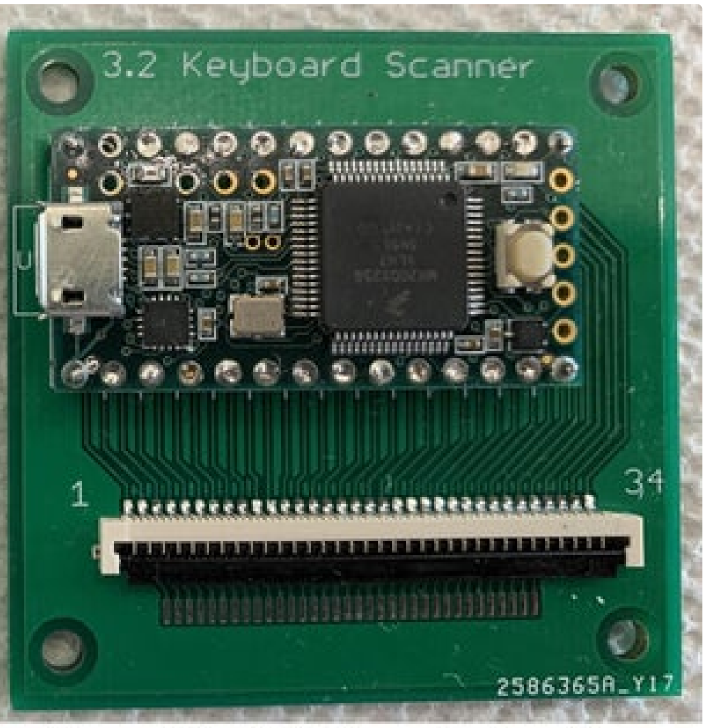
I created a new circuit board for the Teensy 4.0 that routes its 34 I/O signals to pads for an FPC connector with a 1 mm or 0.8 mm pitch (see pictures above).

If your keyboard uses FPC pin 34, you must disconnect the LED near I/O 13 on the Teensy 3.2 or 4.0 in order to avoid interfering with the keyboard scan. I find it easier to unsolder the current limit resistor right next to the LED instead of unsoldering the LED itself.

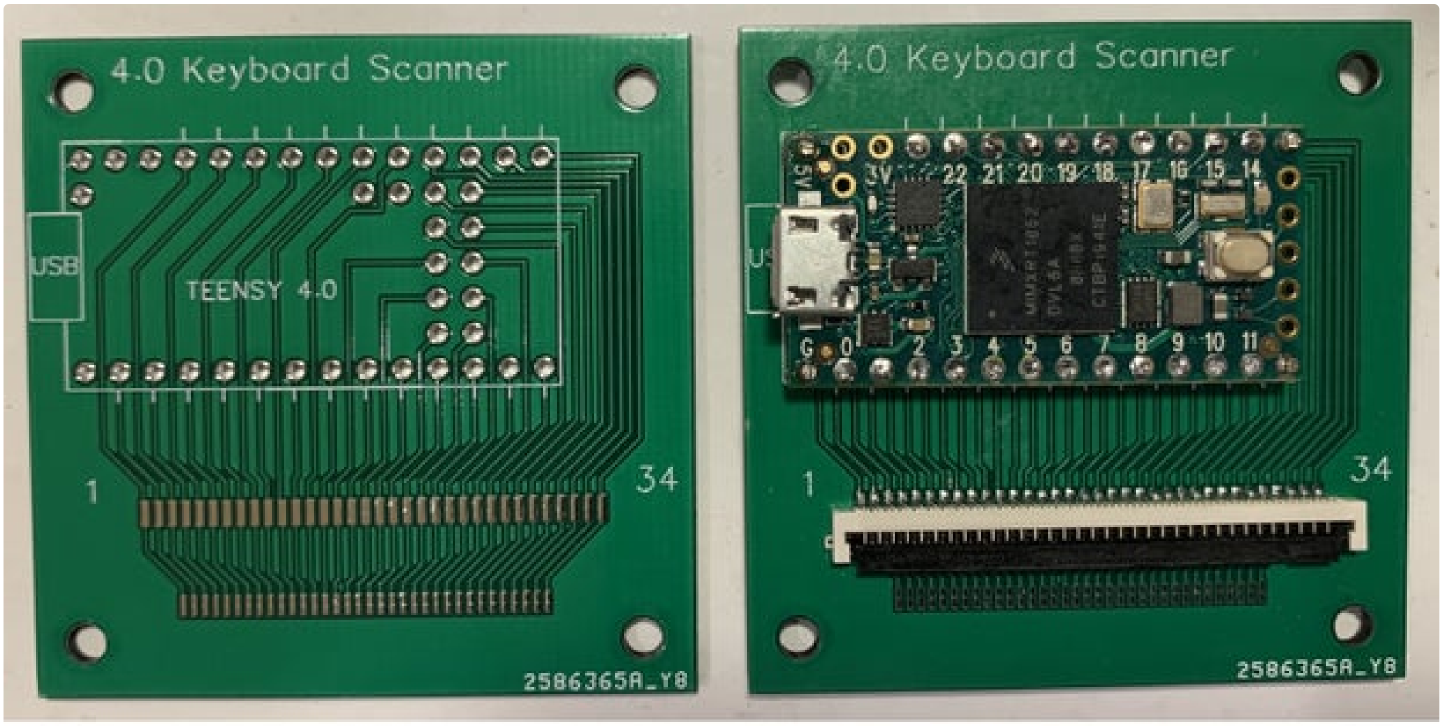
Solder the FPC connector to the 3.2/4.0 side of the board and then proceed to the next step.



Teensy 3.2 Side of Circuit Board



Teensy 3.2 and FPC Connector Assembly



Teensy 4.0 Side of Circuit Board

Teensy 4.0 and FPC Connector Assembly

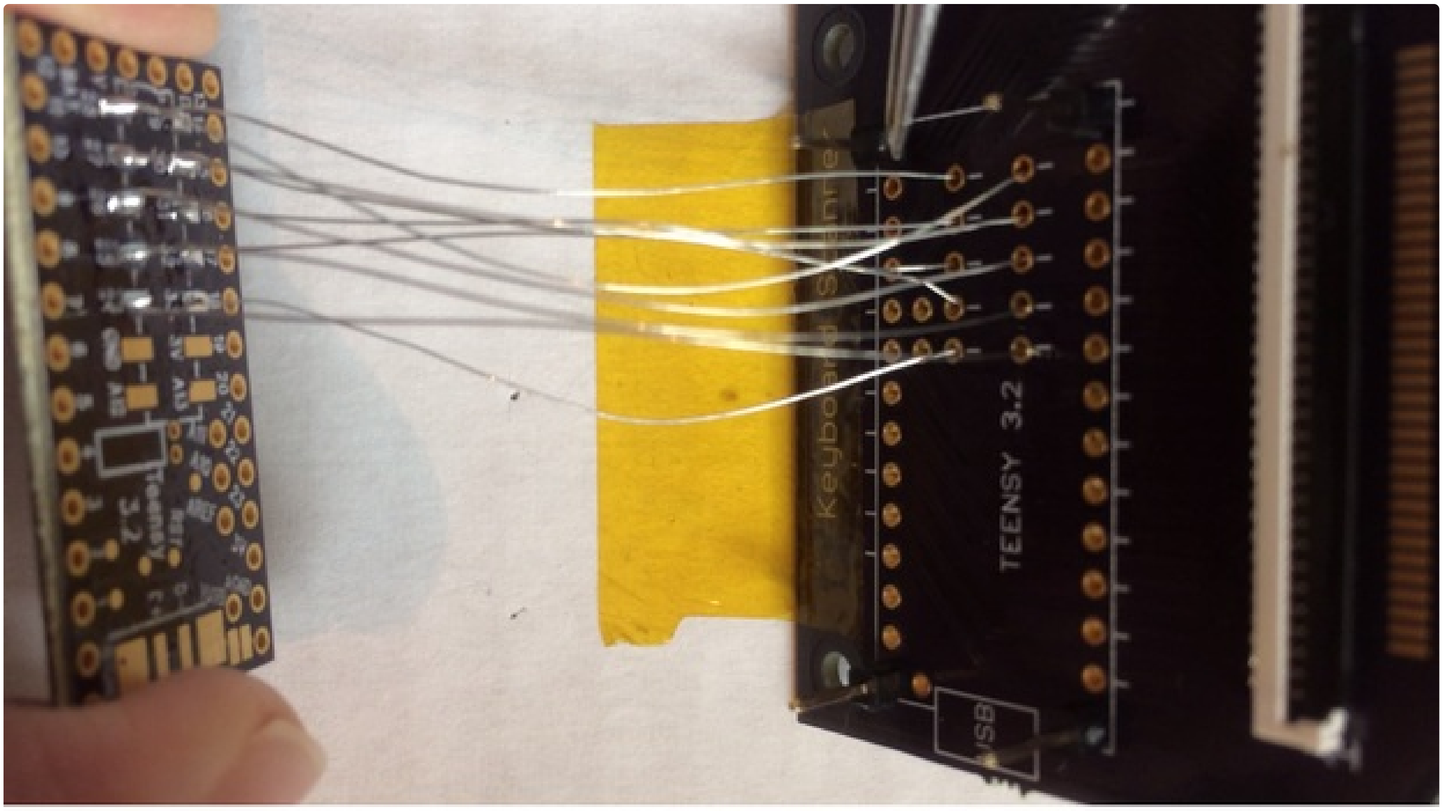
Step 5: Teensy 3.2 and 4.0 Surface Mount Pads

The Teensy 3.2 and 4.0 use surface mount pads for I/O signals 24 thru 33 so it's a little more work to solder them to the board. Solder "flying leads" to the surface mount pads on the Teensy 3.2 or 4.0 and then pass each wire thru the corresponding pad on the board for soldering. Finish the assembly by soldering wires to the remaining I/O signals.

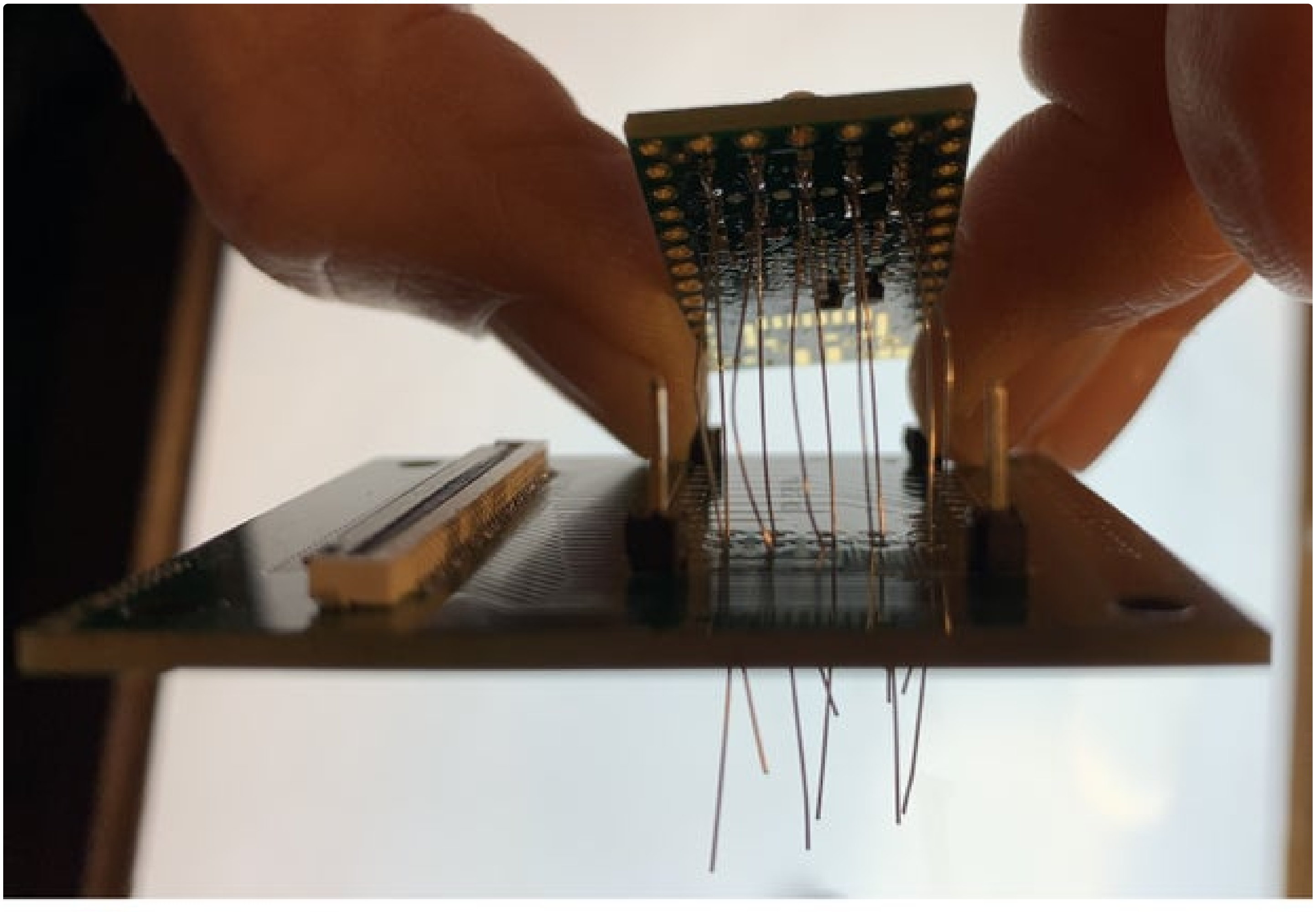
For those that dislike flying leads, I redesigned the Teensy 3.2 board and Teensy 4.0 boards so that header pins can be used for the surface mount pads. The pictures above show how to modify a 2 x 7 right angle header. Note that you can still use flying leads with these boards.

For the Teensy 3.2, cut the header down to 2 x 6, then partially pull and cut the two pins on one end so they are straight as shown above. These 2 pins will help align the header. A small perf board with holes can also be used to hold the header in alignment while soldering the surface mount pads. You will need a long soldering iron tip in order to reach the pads.

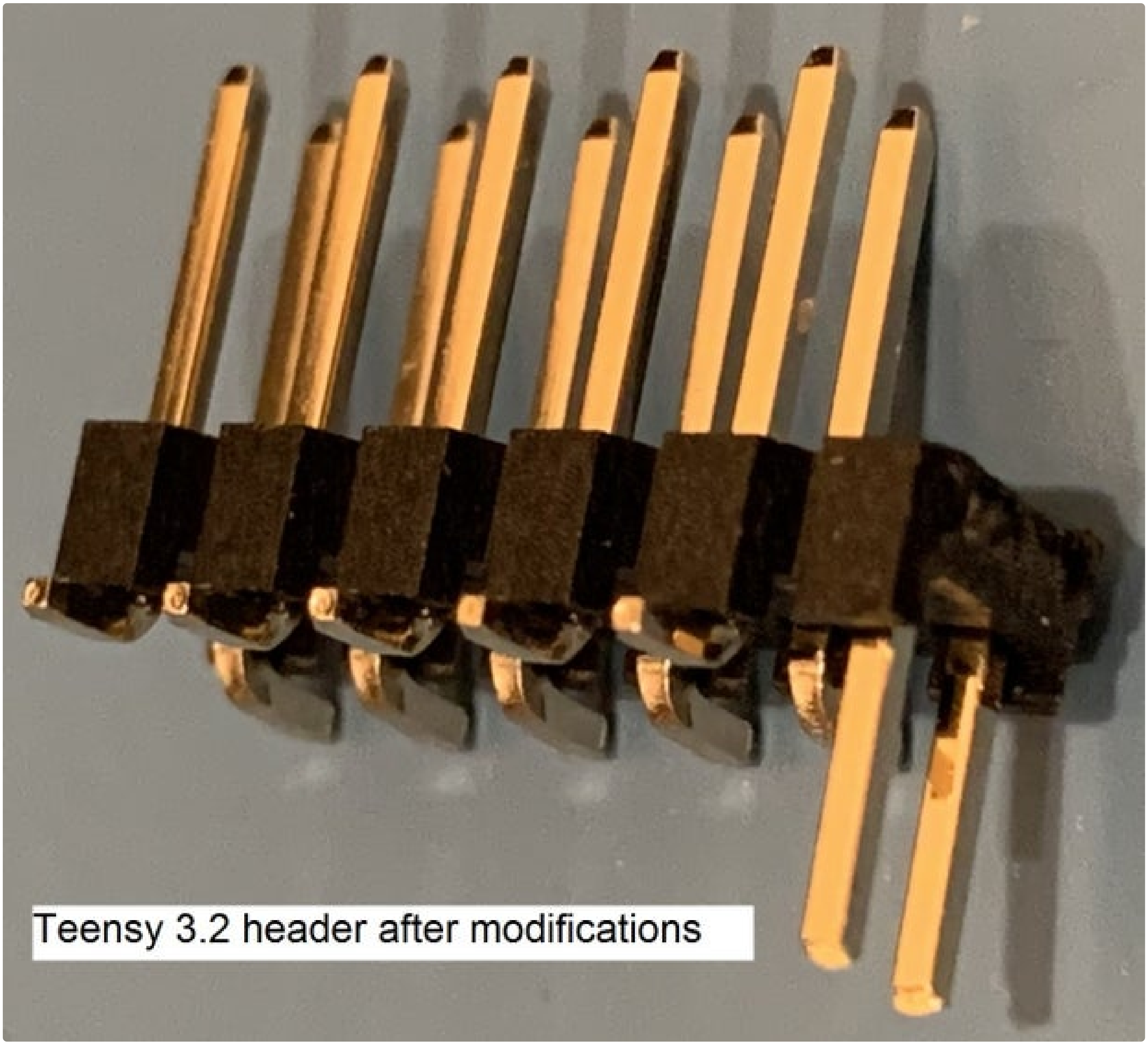
For the Teensy 4.0, pull out the 4 right angle leads at the ends and replace them with 4 straight pins that will hold the header in the correct position. The remaining right angle header pins are a touch too long so I recommend they be trimmed in order to not overhang the surface mount pads of the Teensy 4.0.



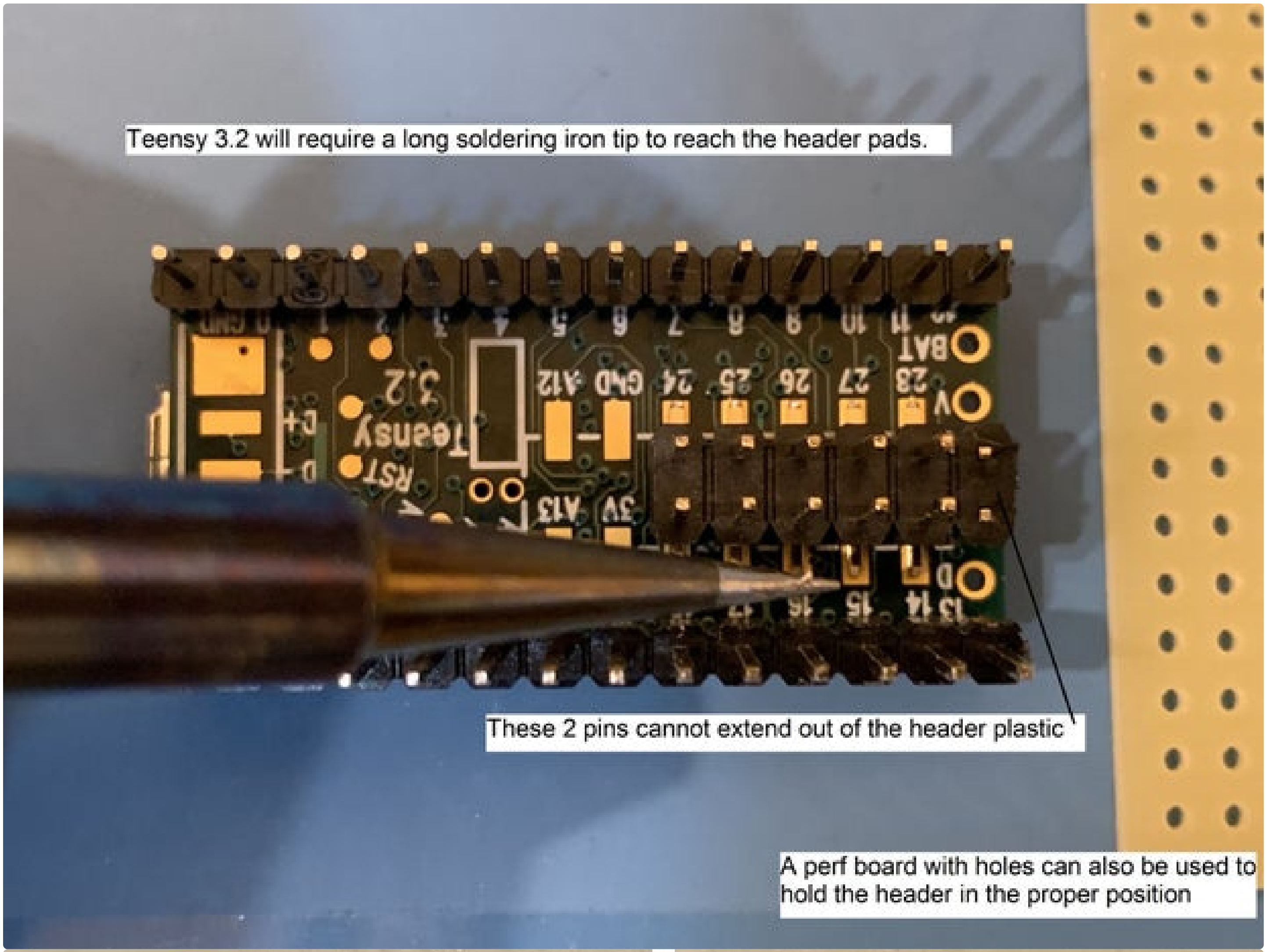
Flying Leads



Teensy 4.0 Flying Leads



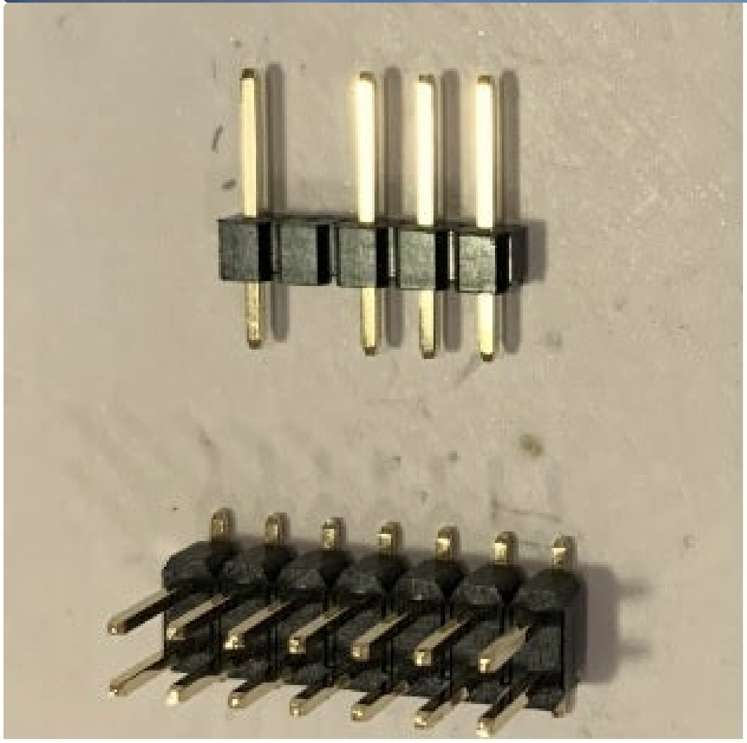
Teensy 3.2 header after modifications



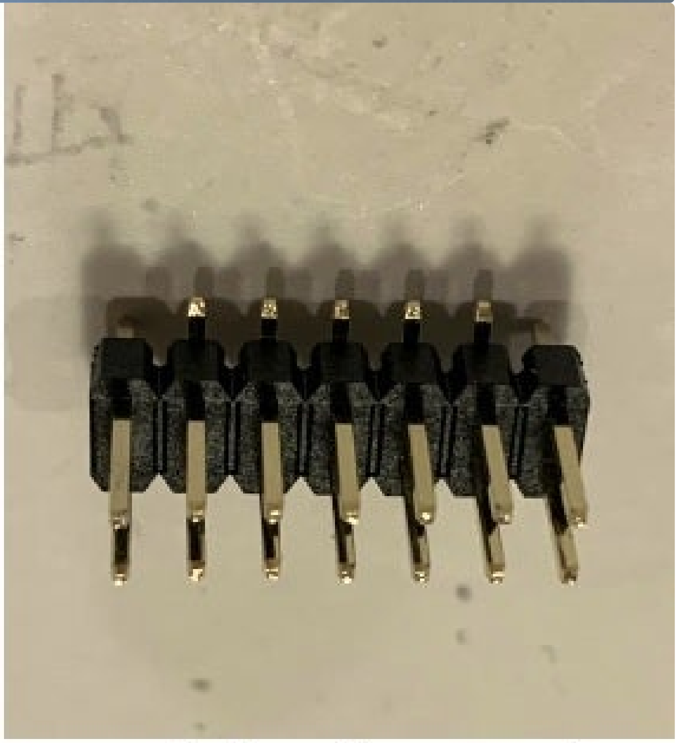
Teensy 3.2 will require a long soldering iron tip to reach the header pads.

These 2 pins cannot extend out of the header plastic

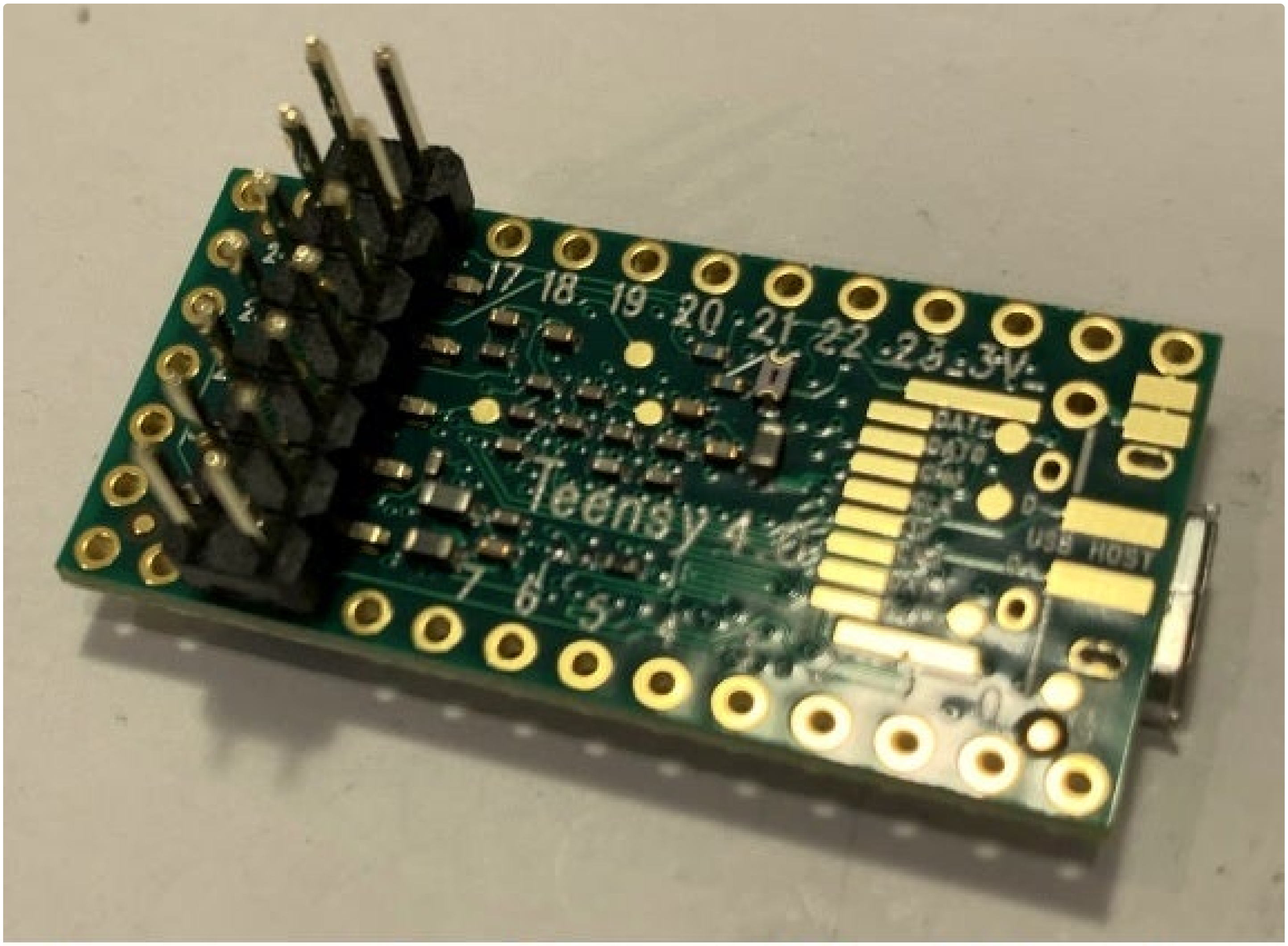
A perf board with holes can also be used to hold the header in the proper position



Original Header



Modified Header



Teensy 4.0 with modified Header

Step 6: Order the Board and Components

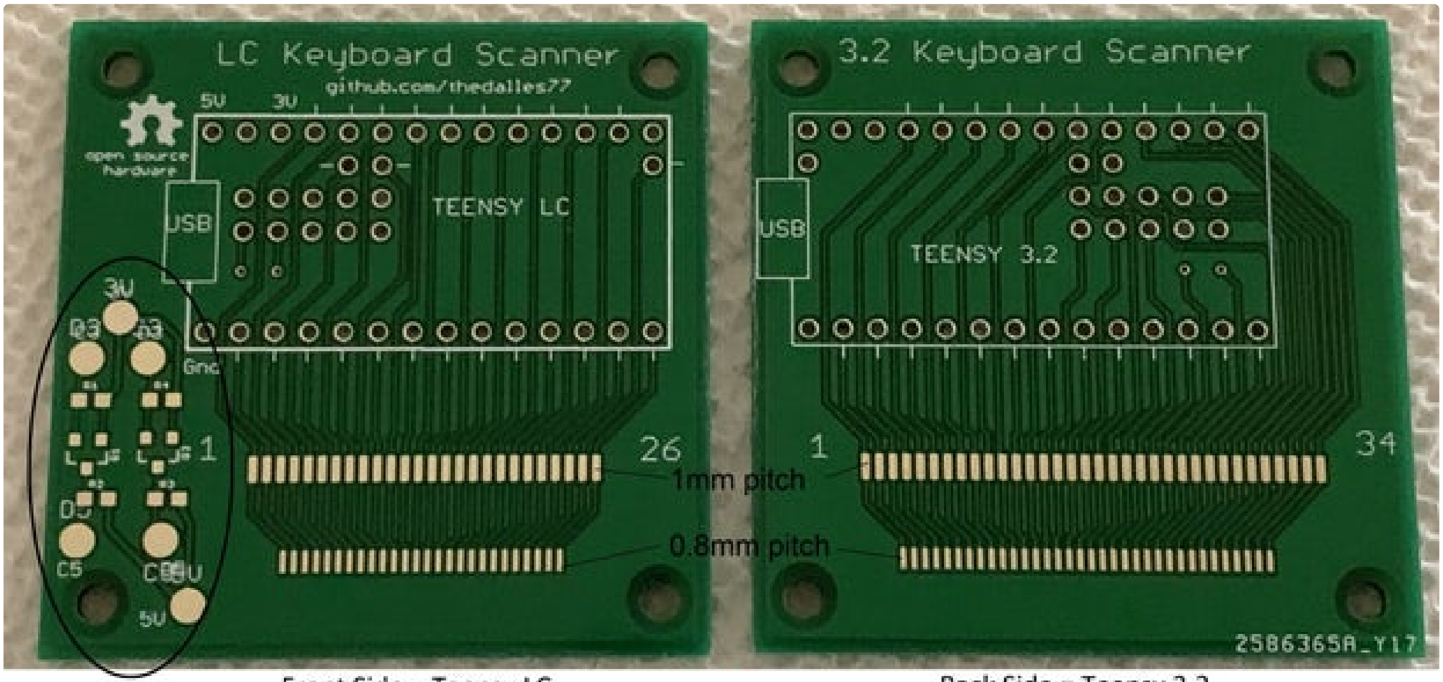
You will need header pins, wire, solder, flux, and a USB cable for this project in addition to the board, connector, and Teensy described below:

- Pick the Teensy that you want to use based on availability and the number of signals on your keyboard cable. The LC is the lowest cost Teensy but is limited to 26 I/O's. The 4.0 is the newest and fastest Teensy and is not affected by the chip shortage. The 3.2 is my favorite because it is 5 volt tolerant but the chip shortage has made supplies scarce. The ++2.0 is the oldest Teensy in this group and it has the most I/O's. It is no longer available from PJRC but there are clones available at AliExpress.
- Decide how you will route the FPC cable to the Teensy connector board and then examine whether the contacts at the end of the cable are facing up or down. It's common to flip the FPC cable 180 degrees so it ends up underneath the keyboard. The surface mount FPC connectors from AliExpress, Mouser, and Digikey can have top or bottom contacts (see example picture above). Besides right angle connectors, there are also vertical connectors. Here are 3 examples for the possible locking mechanisms; spring loaded with no locking bar, flip down locking bar or slide a locking drawer sideways. You should check the

thickness at the end of your FPC cable with a micrometer. A common thickness is 0.30mm but sometimes it's less and you'll need to increase the thickness with tape or a piece of paper.

The following paragraphs describe the various circuit boards that you can use for this project. The Eagle circuit board files have a .brd extension and can be downloaded from my repo. These files are accepted by some board fabricators such as OSH Park in North America and Eurocircuits in Europe. Three boards from OSH Park will cost $18 to $30 depending on the size. Also at my repo are the zipped Gerber files that I translated from the Eagle files. Gerber is accepted by all board houses such as JLCPCB in China. They will produce 5 boards for about $10 with economy shipping to the U.S. They use a HASL (aka HAL) surface finish which is not as good as the ENIG finish used by OSH Park. I have had no trouble with the JLCPCB HASL finish. The different PCB surface finishes are explained by Optimum Design Associates and Eurocircuits.

- To use a Teensy LC or 3.2 with a surface mount 1mm or 0.8mm pitch FPC connector, use the Keyboard_Scanner_LT2.brd or zip file. This board has pads for a 2 bit level translator in case you want your Teensy LC to talk to a 5 volt PS/2 touchpad. If you don't want the level translator, leave these pads empty. The Teensy 3.2 is 5 volt tolerant so it doesn't need a translator. The hole spacing for the Teensy 3.2 backside I/O signals are spaced for right angle header pins or you can use flying leads.
- To use a Teensy 4.0 with a surface mount 1mm or 0.8mm pitch FPC connector, use the Keyboard_Scanner_4p0.brd or zip file. For 0.5mm pitch FPC connector, use the Keyboard_Scannmer_4p0_0p5.brd or zip file. The hole spacing for the Teensy 4.0 backside I/O signals are spaced for right angle header pins or you can use flying leads. The Teensy 4.0 is not 5 volt tolerant so use the 2 bit level translator on the Teensy LC side of the board if you want to control a 5 volt PS/2 touchpad.
- If your keyboard needs a 0.5mm pitch FPC connector, use the Keyboard_Scanner_LT_0p5.brd or zip file. This board will work with a Teensy LC on one side or a Teensy 3.2 on the other side with flying leads.
- If you don't want to solder a surface mount FPC connector, I designed a board for a thru hole FPC connector. This board uses a Teensy LC and can take an FPC cable with up to 26 pins on a 1mm pitch. Use Eagle file Keyboard_Scanner_LC_thruhole.brd or the zip file for this board. The connector hole pattern on the board is designed for this AliExpress or Digikey connector.
- The Teensy ++2.0 (clone) is still available from AliExpress so I designed a board that will connect it to an FPC connector with either a 1mm, 0.8mm, or 0.5mm pitch and up to 36 pins. You must un-solder the LED located on the Teensy ++2.0 so that it does not interfere with the keyboard software. The Eagle board file Keyboard_Scanner_2pp.brd and the gerber zip file are at my repo.
- Some old keyboards have a 1.25mm pitch FPC cable so I made Eagle file TeensyLC_1p25.brd. It will take a thru hole connector with up to 26 pins and a 1.25mm pitch like these from AliExpress.

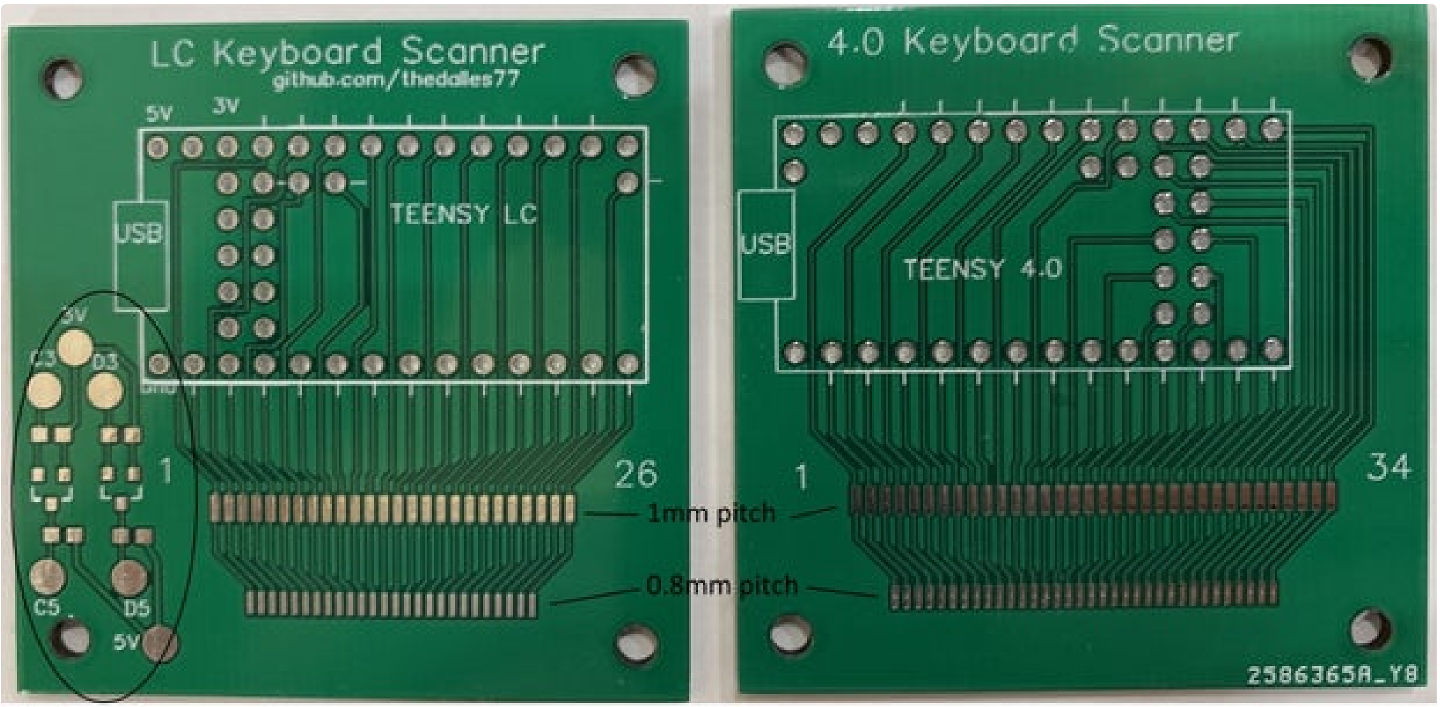


Front Side = Teensy LC

Back Side = Teensy 3.2

Optional 2 bit level translator for touchpad

File = Keyboard_Scanner_LT2.brd

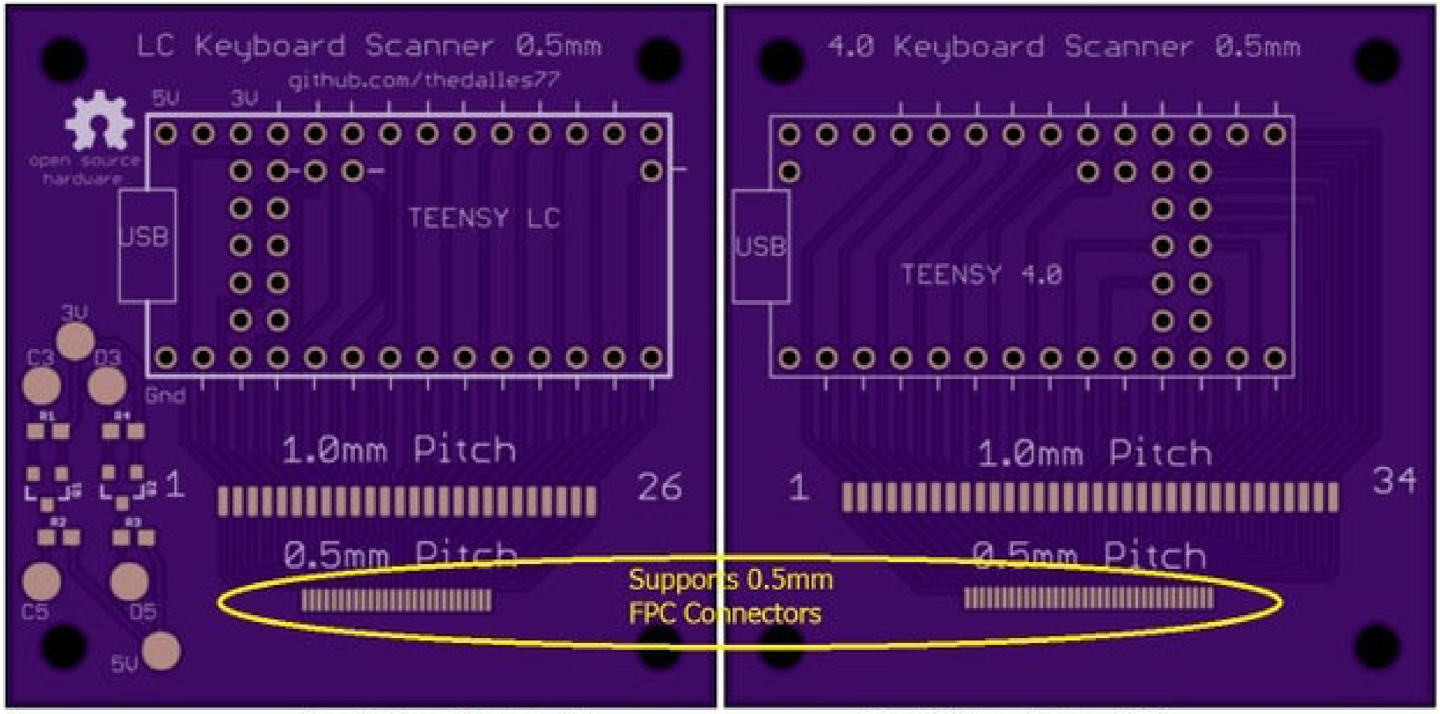


Optional 2 bit level translator for touchpad

Front Side = Teensy LC

Back Side = Teensy 4.0

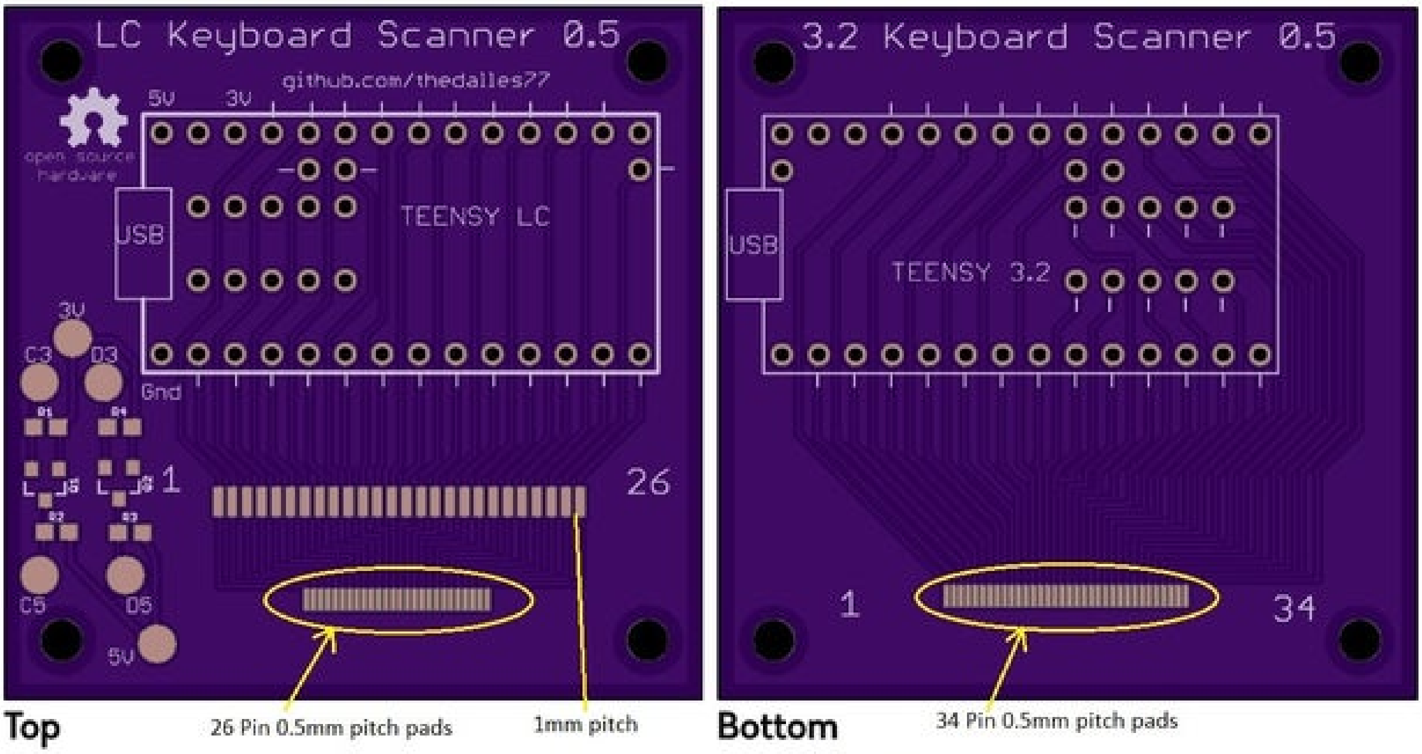
File = Keyboard_Scanner_4p0.brd



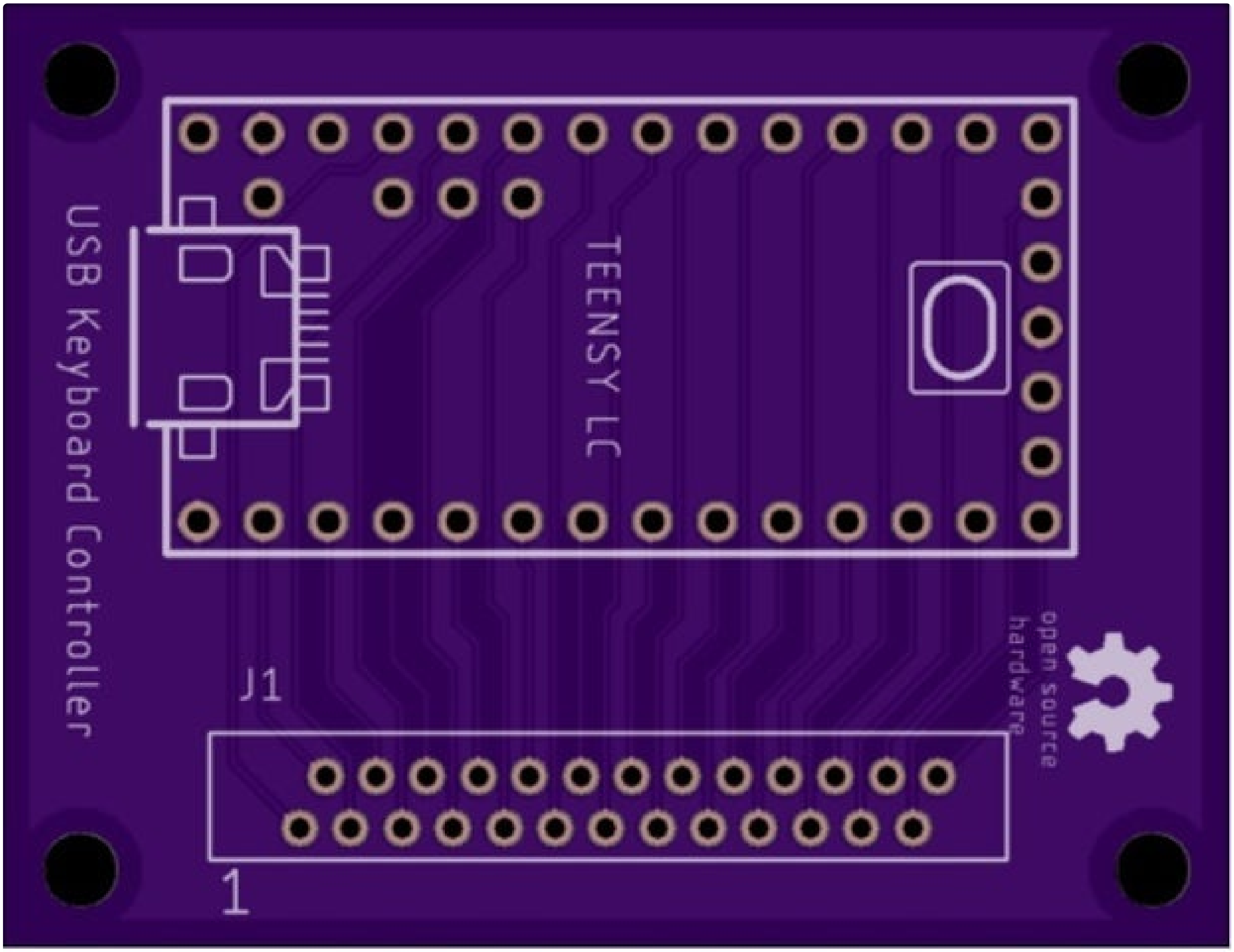
Front Side = Teensy LC

Back Side = Teensy 4.0

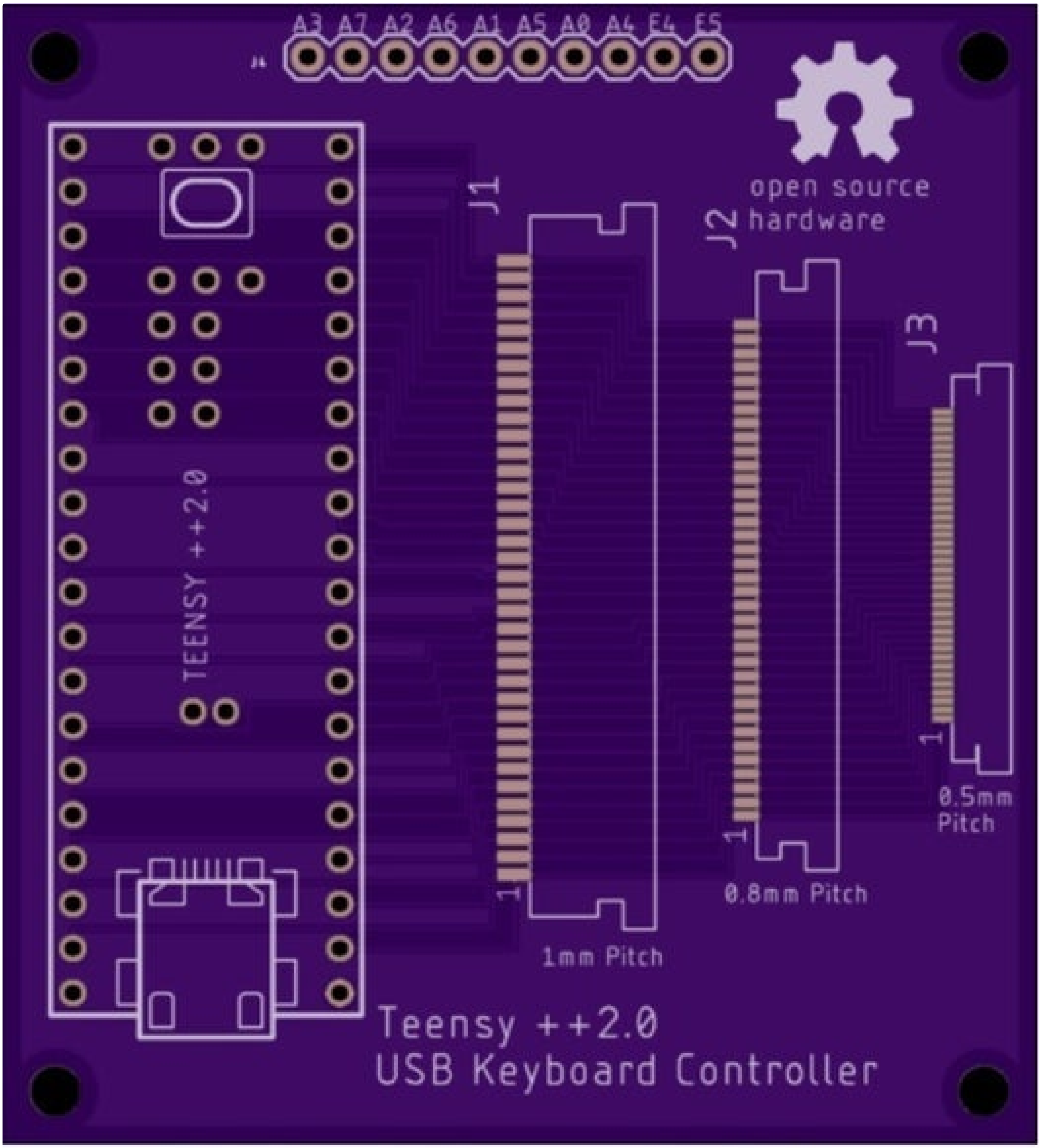
File = Keyboard_Scanner_4p0_0p5



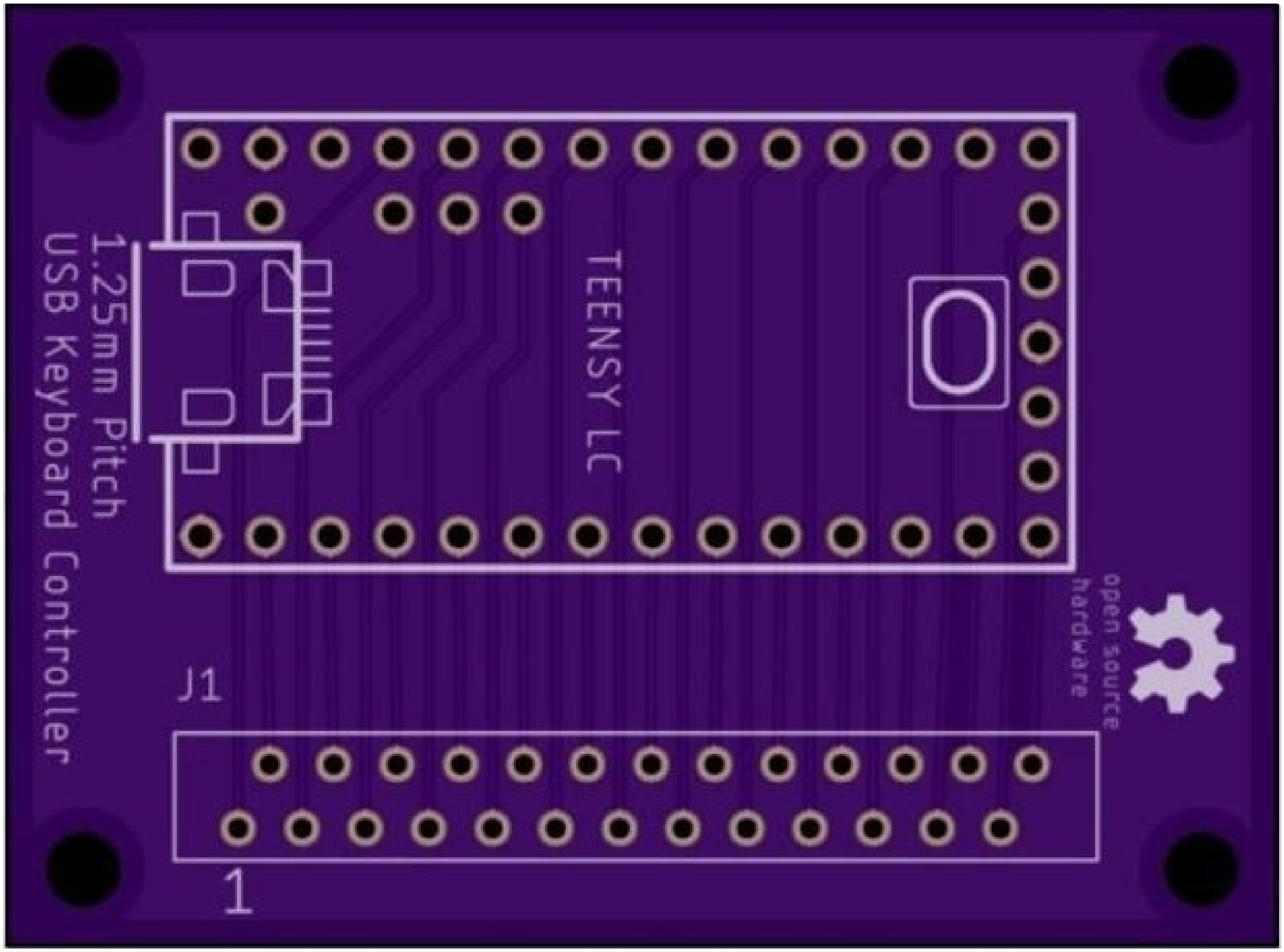
Keyboard_Scanner_LT_0P5.brd



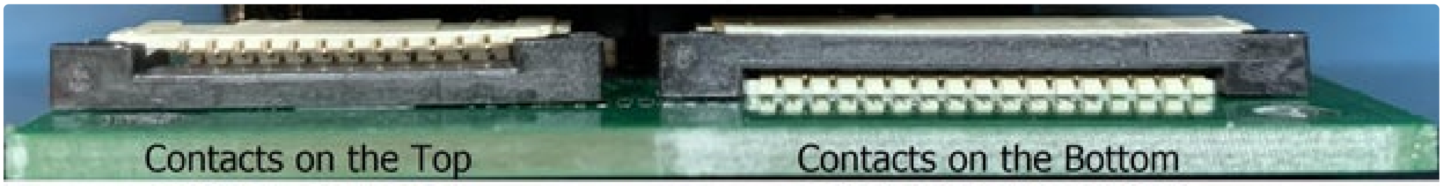
Keyboard_Scanner_LC_thruhole.brd



Keyboard_Scanner_2pp.brd



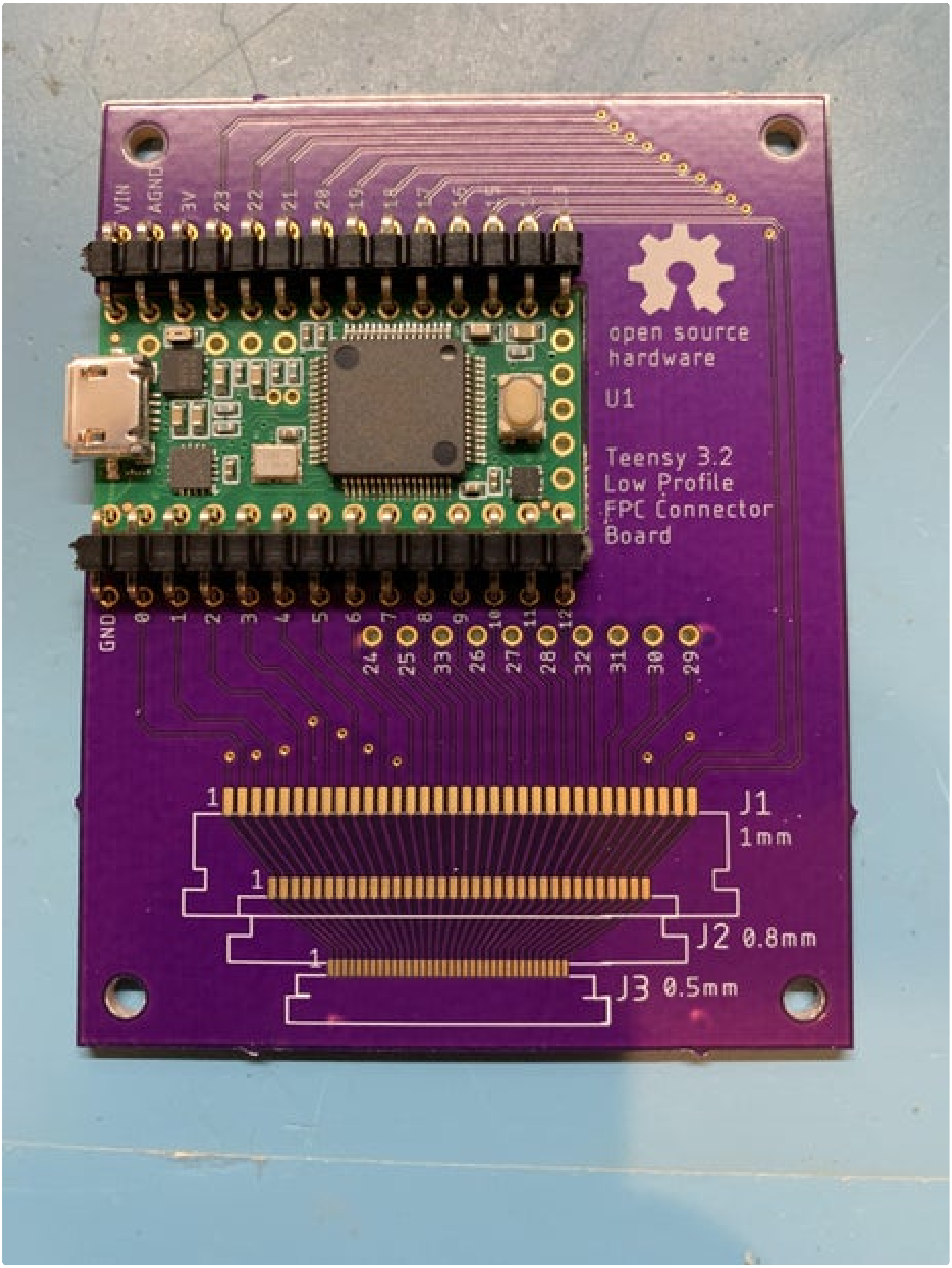
TeensyLC_1p25.brd

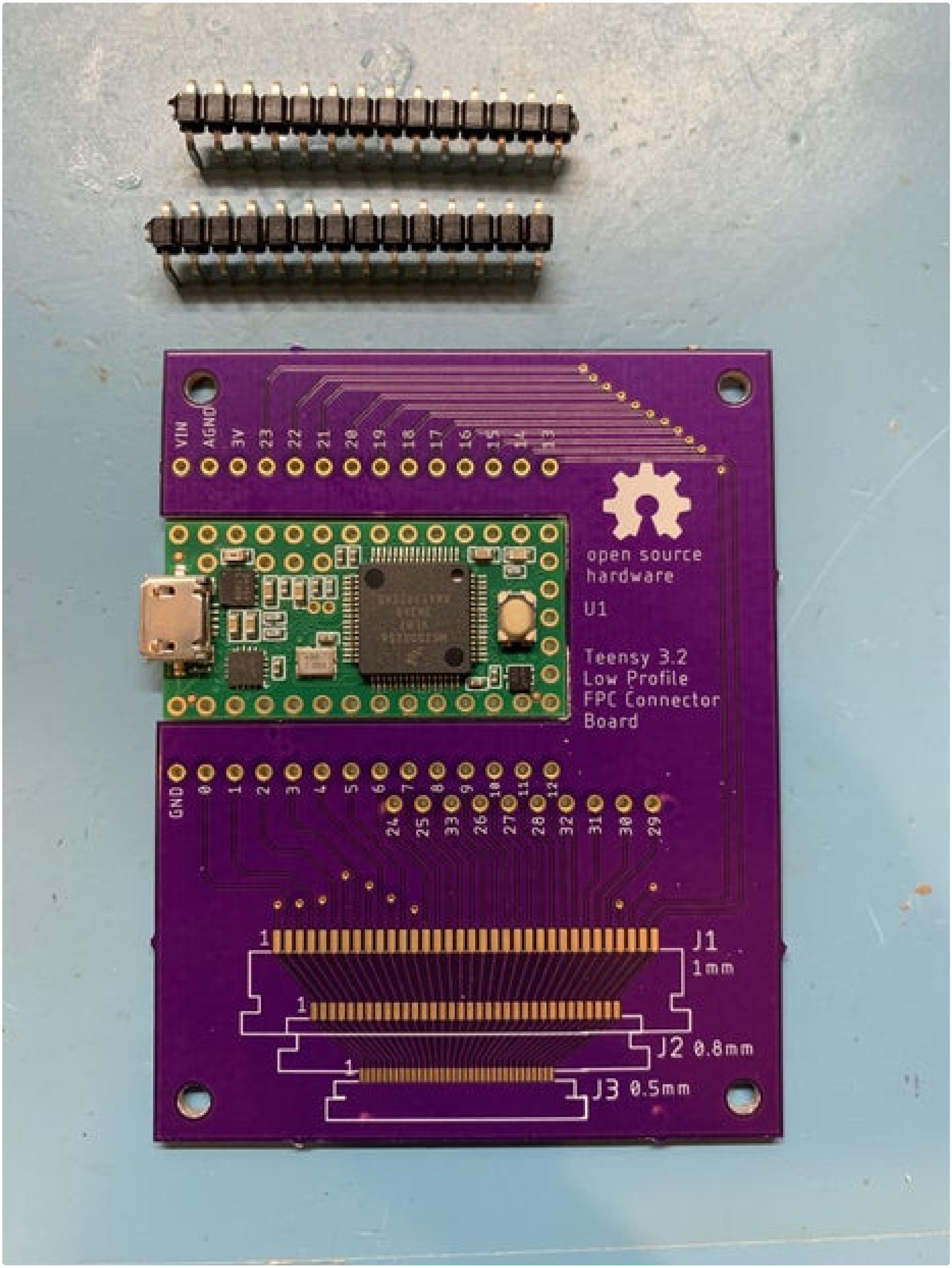


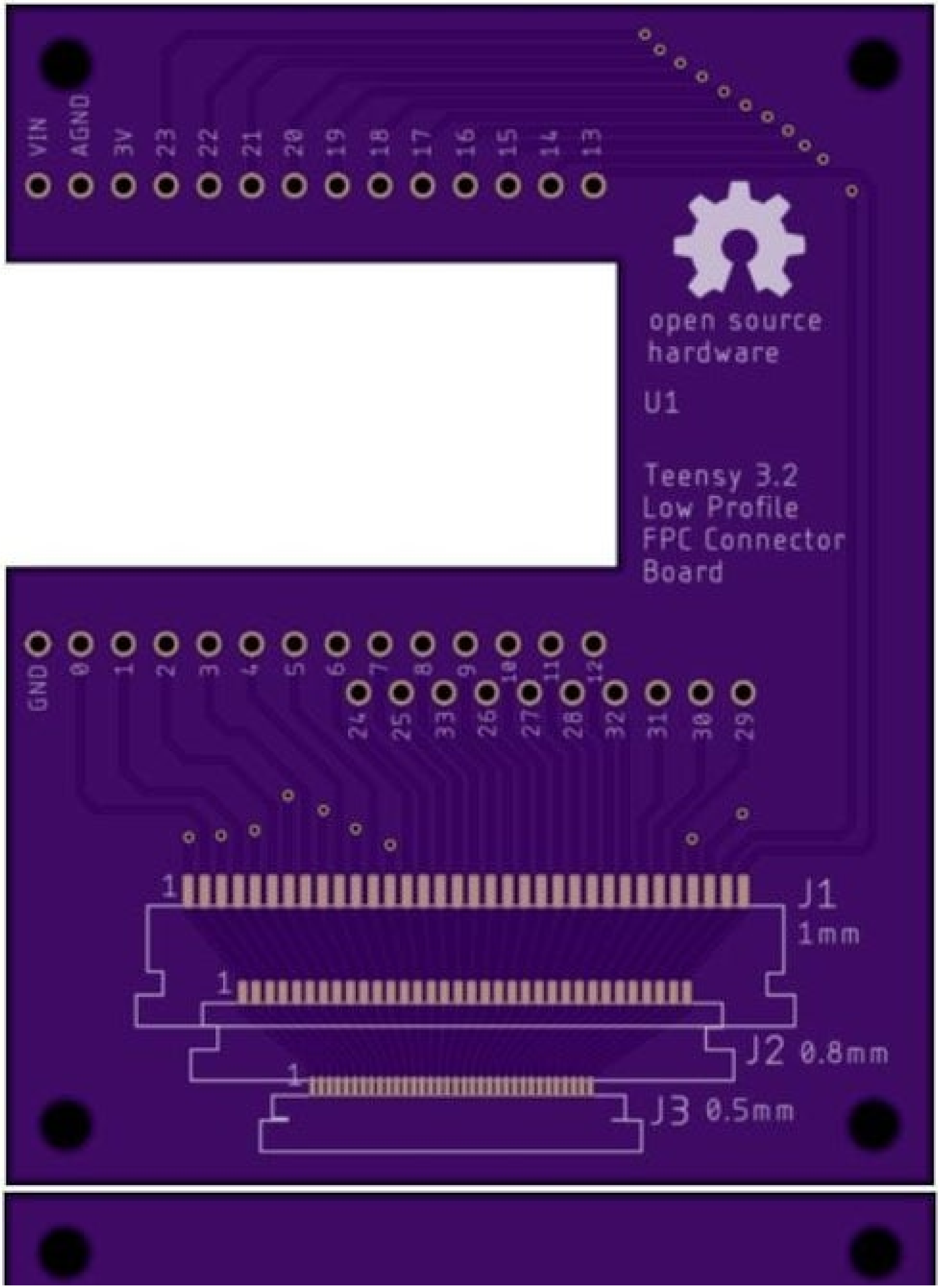
Right angle FPC connectors (drawer slide lock)

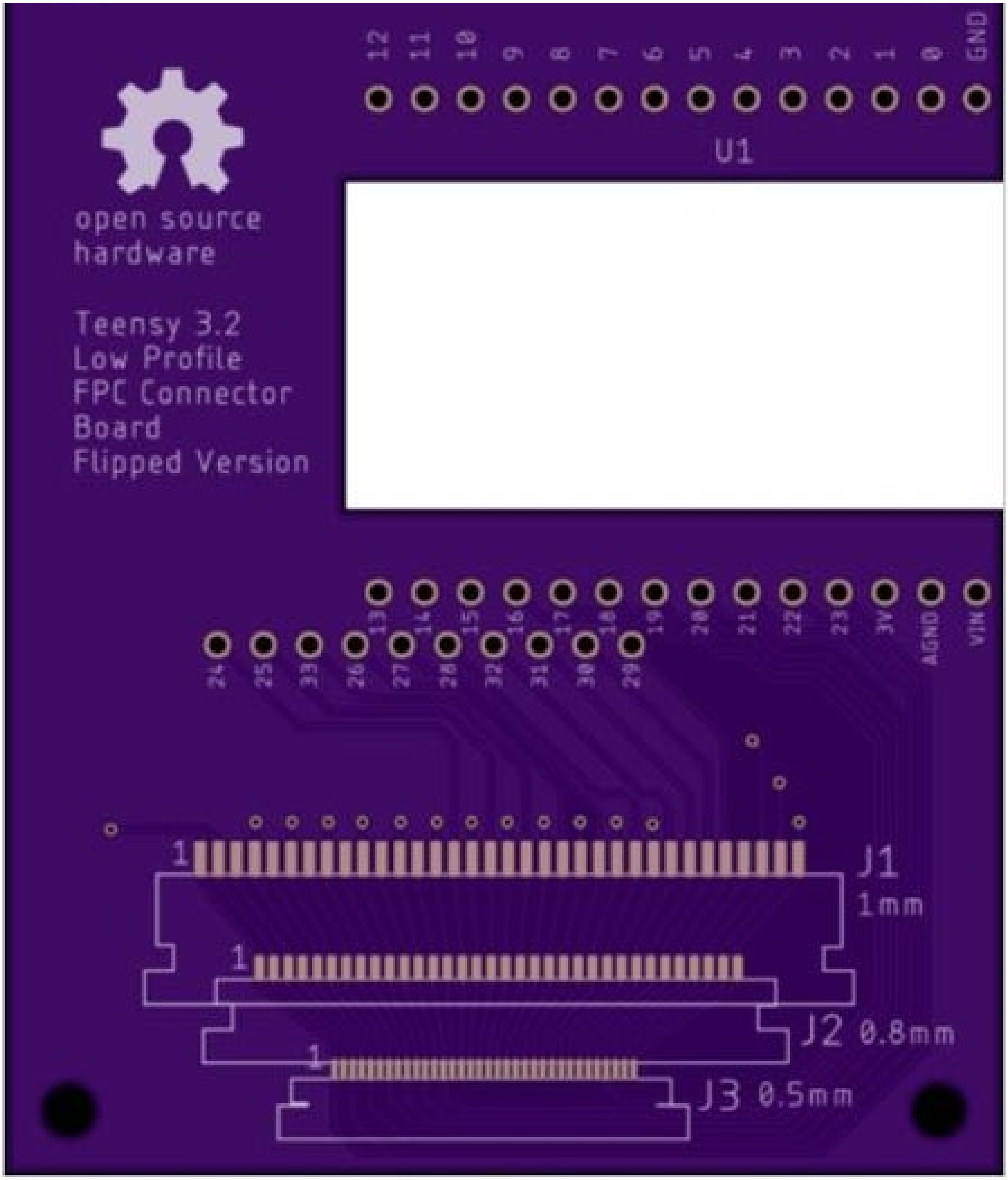
Step 7: Low Profile Connector Board - Teensy LC, 3.2, & 4.0

A Teensy with header pins that is mounted on an FPC connector board may be too tall to fit in a thin laptop case so I designed the board shown above. The Eagle board file, "Teensy3p2Cutout.brd" can be downloaded from my repo. A Teensy 3.2 (as well as an LC or 4.0) will fit in the cutout and U shaped header pins on each side will hold the Teensy in place. These pins will tie Teensy I/O numbers 0 thru 23 to the connector board. Teensy I/O numbers 24 thru 33 are on backside SMD pads and must be connected with jumper wires. I used "U" shaped header pins from Ali Express that have a width of 6mm (even though their documentation shows a 5mm width). The board will work with 1mm, 0.8mm, or 0.5mm pitch FPC connectors with up to 34 pins. The Eagle file "Teensy3p2CutoutFlip.brd" rotates the Teensy 180 degrees so it can fit in a cutout on the right side instead of the left (see picture above). Use this board if the USB connection is on the right side of the laptop base. The left cutout and right cutout boards route the I/O signals the same as the Keyboard_Scanner_LT2 connector board and will use the same Teensy 3.2 software. If you would rather use a Teensy 4.0, it will also fit in the board cutout. If you wire the backside I/O numbers 24 thru 33 per the silkscreen labeling, you can use the Teensy 3.2 matrix decoder software to get the correct translation. The Teensy LC can also be used in this manner with jumpers for I/O numbers 24 thru 26.


VIN
AGND
3V
23
22
21
20
19
18
17
16
15
14
13
open source hardware
U1
Teensy 3.2
Low Profile
FPC Connector
Board
GND
0
1
2
3
4
5
6
7
8
9
10
11
12
24
25
33
26
27
28
32
31
30
29
1
J1
1mm
1
J2 0.8mm
1
J3 0.5mm


VIN
AGND
3V
23
22
21
20
19
18
17
16
15
14
13
open source hardware
U1
Teensy 3.2
Low Profile
FPC Connector
Board
GND
0
1
2
3
4
5
6
7
8
9
10
11
12
24
25
33
26
27
28
32
31
30
29
1
J1
1mm
1
J2 0.8mm
1
J3 0.5mm

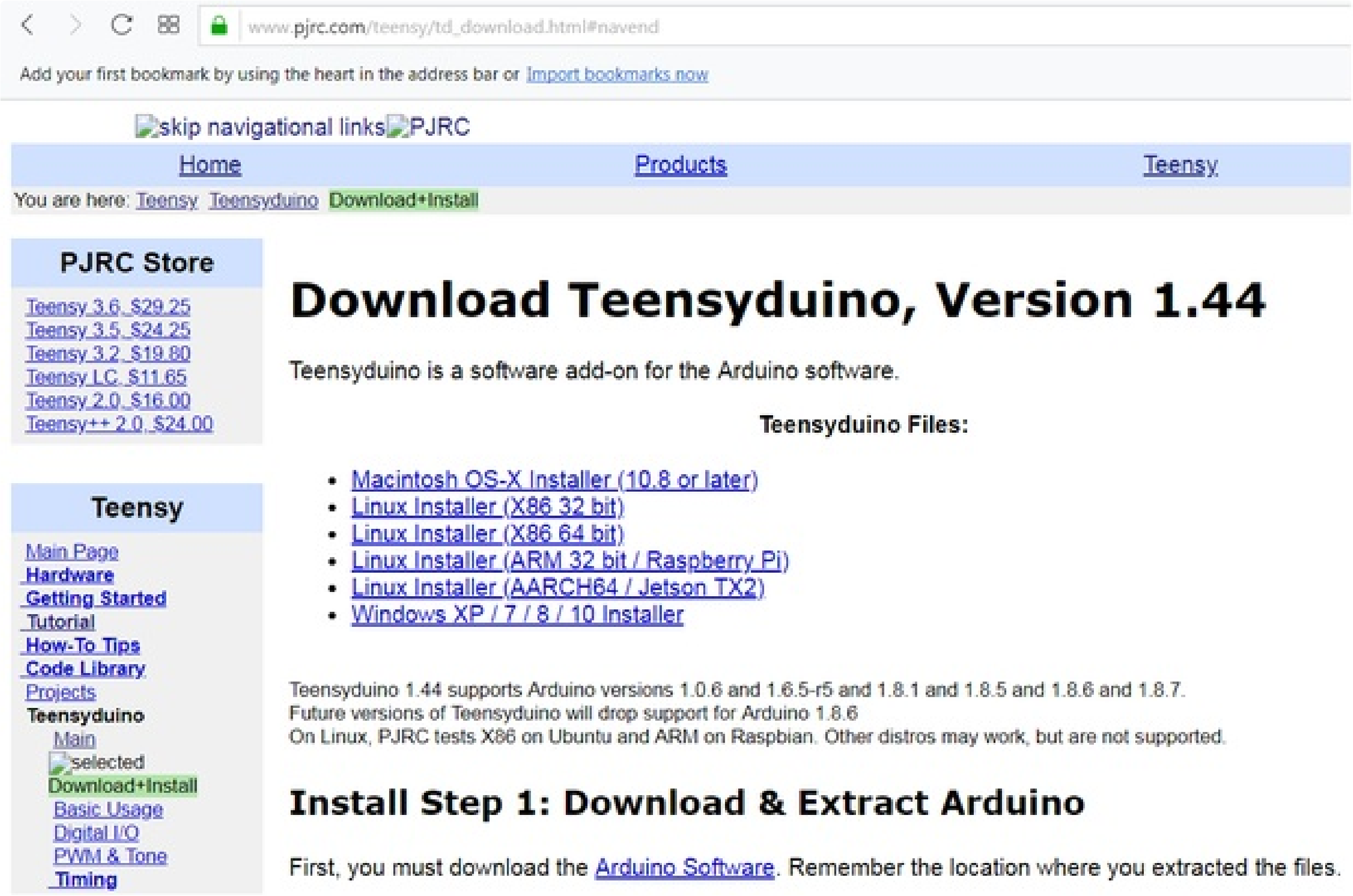

VIN
AGND
3V
23
22
21
20
19
18
17
16
15
14
13
open source hardware
U1
Teensy 3.2 Low Profile FPC Connector Board
GND
0
1
2
3
4
5
6
7
8
9
10
11
12
24
25
33
26
27
28
32
31
30
29
1
J1 1mm
1
J2 0.8mm
1
J3 0.5mm



The file "Teensy3p2CutoutFlip.brd" has the Teensy 3.2 on the right side instead of the left.

Step 8: Load the Continuity Tester Into the Teensy

- Follow the PJRC link for installing Arduino and Teensyduino on your computer.
- Download the Matrix_Decoder Arduino code. Use file Matrix_Decoder_LC.ino for a Teensy LC, Matrix_Decoder_3p2.ino for a Teensy 3.2, Matrix_Decoder_4p0.ino for a Teensy 4.0 or Matrix_Decoder_2pp.ino for a Teensy ++2.0.
- Load the Matrix_Decoder code into the Arduino integrated development environment (IDE).
- Connect a USB cable from the Teensy to the computer. Your computer should automatically load the necessary USB drivers.
- In the Arduino IDE, under "tools", "board", "Teensyduino", select Teensy LC, 3.2, 4.0, or ++2.0 depending on what you're using. Also under "tools", select USB type: Keyboard. If you forget to do this step, you will get an error message that says "Keyboard was not declared in this scope".
- Compile and load the Matrix_Decoder code into the Teensy. If it's your very first time loading the Teensy, you'll have to push the button on the Teensy to enable the loader.
- Disconnect the USB cable from the Teensy.
- If you are using a Teensy ++2.0, you must unsolder the LED on the Teensy so it does not interfere with the scan.



www.pjrc.com/teensy/td_download.html#navend

Add your first bookmark by using the heart in the address bar or Import bookmarks now

skip navigational links PJRC

Home Products Teensy

You are here: Teensy Teensyduino Download+Install

PJRC Store

Teensy 3.6, $29.25
Teensy 3.5, $24.25
Teensy 3.2, $19.80
Teensy LC, $11.65
Teensy 2.0, $16.00
Teensy++ 2.0, $24.00

Teensy

Main Page
Hardware
Getting Started
Tutorial
How-To Tips
Code Library
Projects
Teensyduino
Main
selected
Download+Install
Basic Usage
Digital I/O
PWM & Tone
Timing

Download Teensyduino, Version 1.44

Teensyduino is a software add-on for the Arduino software.

Teensyduino Files:

- Macintosh OS-X Installer (10.8 or later)
- Linux Installer (X86 32 bit)
- Linux Installer (X86 64 bit)
- Linux Installer (ARM 32 bit / Raspberry Pi)
- Linux Installer (AARCH64 / Jetson TX2)
- Windows XP / 7 / 8 / 10 Installer

Teensyduino 1.44 supports Arduino versions 1.0.6 and 1.6.5-r5 and 1.8.1 and 1.8.5 and 1.8.6 and 1.8.7.
Future versions of Teensyduino will drop support for Arduino 1.8.6
On Linux, PJRC tests X86 on Ubuntu and ARM on Raspbian. Other distros may work, but are not supported.

Install Step 1: Download & Extract Arduino

First, you must download the Arduino Software. Remember the location where you extracted the files.

Download

https://www.instructables.com/FSS/6WKW/K9700QSF/FSS6WKWK9700QSF.ino

Download

https://www.instructables.com/FYR/E1SX/K9700QSG/FYRE1SXK9700QSG.ino

Download

https://www.instructables.com/FE4/CTOH/K9700QSH/FE4CTOHK9700QSH.ino

https://www.instructables.com/FKJ/BL87/KDA8O9T4/FKJBL87KDA8O9T4.ino Download

Step 9: Load the Key-List File in the Editor

Open a text editor on your computer. I like to use Notepad++ on Windows or Geany on the Pi because they have column editing.

Original Method:

There are two “key-list” text files you can download, named Keyboard_without_number_pad and Keyboard_with_number_pad. The “key-list” file should have every key that you will push followed by tabs to make the results more readable and easy to copy into a spreadsheet.

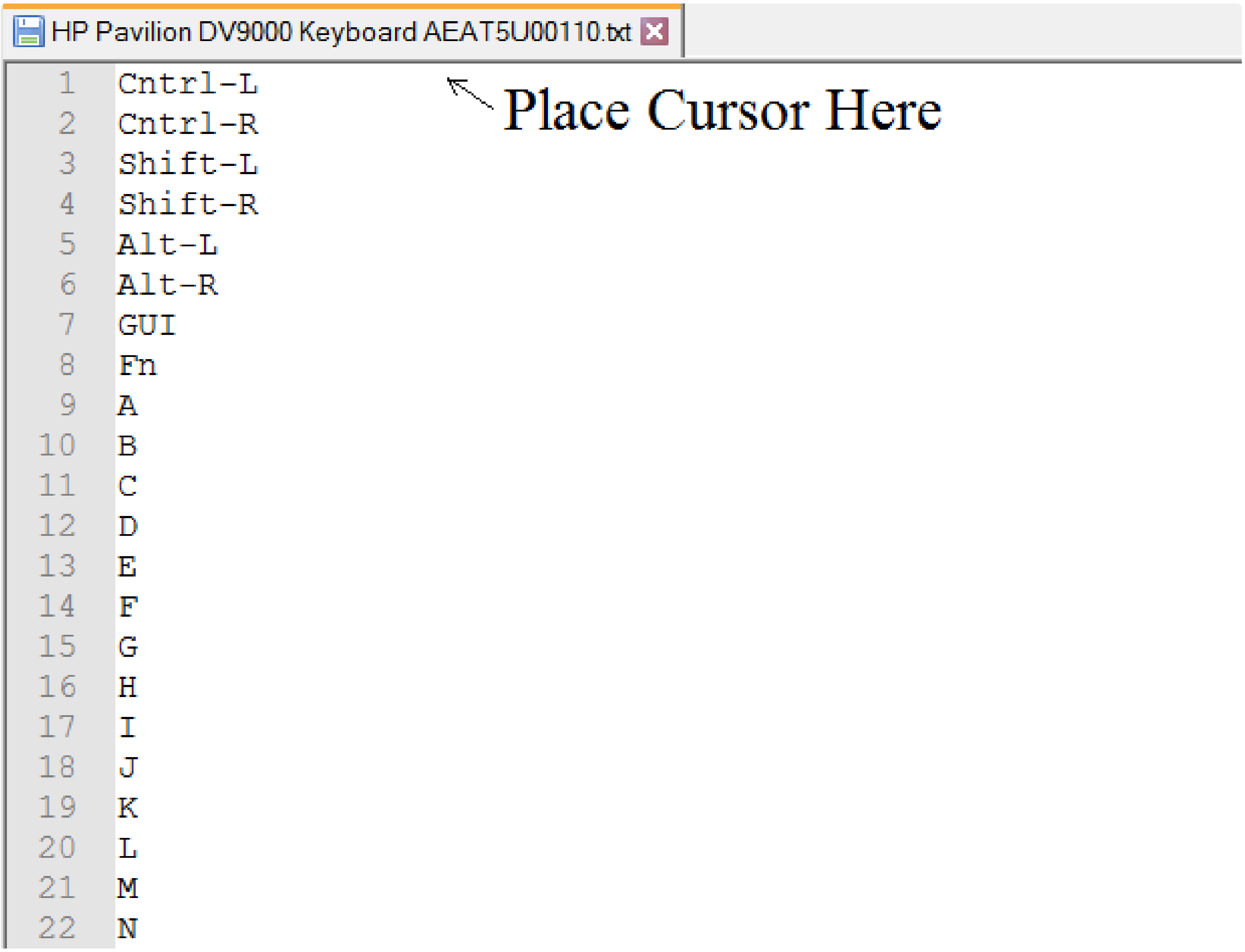
New Method:

Marcel Hillesheim has written a Python program that takes much of the manual labor out of my original process. The latest version of his program includes a menu which selects the Teensy LC, 3.2, or 4.0 (but not the Teensy ++2.0). The Python program is at his GitHub repo along with two blank key-list files that use the PJRC key codes. If you're comfortable running Python, download the key list text file (with or without number pad) and the matrixgenerator Python program. It will save you a lot of time.

Modify as needed:

You may need to modify the key list file slightly to match your keyboard’s keys. A non-US keyboard can still use this routine, just make a list of your keys and the Teensy will report pin connections. The GUI key is either the "windows key" from a PC or the "clover key" from a Mac. Place the cursor to the right of the very first key in the list as shown above. This will determine where the Teensy begins to display the pin numbers as you push each key.

French computers require the shift key to be pushed in order to display the numbers 0 thru 9. I have made French matrix decoder code for the Teensy LC, 3.2, and 4.0 that takes care of the shift key so that numbers are displayed instead of @ " ' (- _ etc. You can find this code at my Github repo.



Place Cursor Here

```
Cntrl-L
Cntrl-R
Shift-L
Shift-R
Alt-L
Alt-R
GUI
Fn
A
B
C
D
E
F
G
H
I
J
K
L
M
N
```

Download

https://www.instructables.com/FZ8/L5AQ/JOSY1ULO/FZ8L5AQJOSY1ULO.txt

Download

https://www.instructables.com/F4H/P3YR/JOSY1ULP/F4HP3YRJOSY1ULP.txt

Step 10: Connect the FPC Cable

Inspect your FPC connector to determine the correct orientation for the cable.

If your FPC connector has contacts on the bottom like the picture on the left, use your finger nail to gently lift the connector locking bar to the open position. Slide the FPC cable into the connector with the bare metal contacts pointed down (closest to the board) and the plastic backing strip pointed up. Lock the cable to the connector by gently pushing the bar down. The locking bar mashes the bare metal of the FPC cable down against pins on the bottom of the connector.

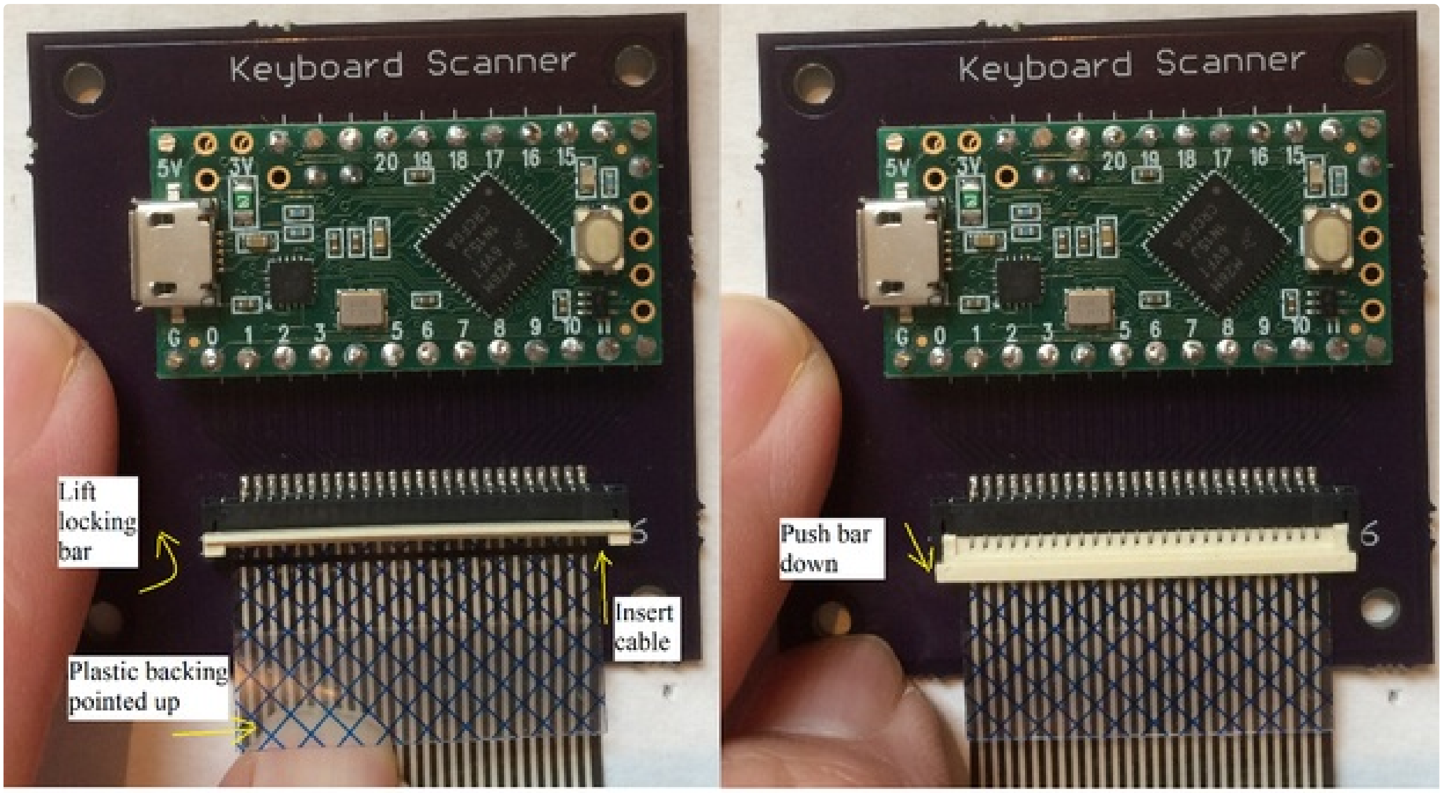
If your FPC connector has contacts on the top like the picture on the right, use your finger nail to slide the locking bar to the right. Insert the cable into the connector with the bare metal contacts pointed up and the plastic backing strip pointed down. The orientation is opposite from the first picture because when the locking bar is slid to the left, it pushes the cable up against pins on the top of the connector.

Connect a USB cable from the Teensy to the computer and **wait 20 seconds** for the Teensy to be recognized as a USB keyboard. This delay is in the code to make sure your computer is ready to receive numbers from the Teensy. If numbers are reported on the screen before any keys are pressed, these pins are shorted together and must be fixed. It may be a solder short on the connector which you can check by retrying without the keyboard connected. If it is still shorted without the keyboard, use flux and solder wick on the shorted connector pads. If the short goes away when the keyboard is removed, these pins are most likely ground pins that are tied together in the keyboard and must be excluded from the routine. Adjust the max_pin or min_pin values in the code to allow the routine to skip over these pins. Another reason to reduce the max_pin value is because some of the FPC traces may be visibly tied together as shown in the picture above. The original motherboard controller used these shorted pins to identify the language of the keyboard but the Teensy code will report the short and then hang.

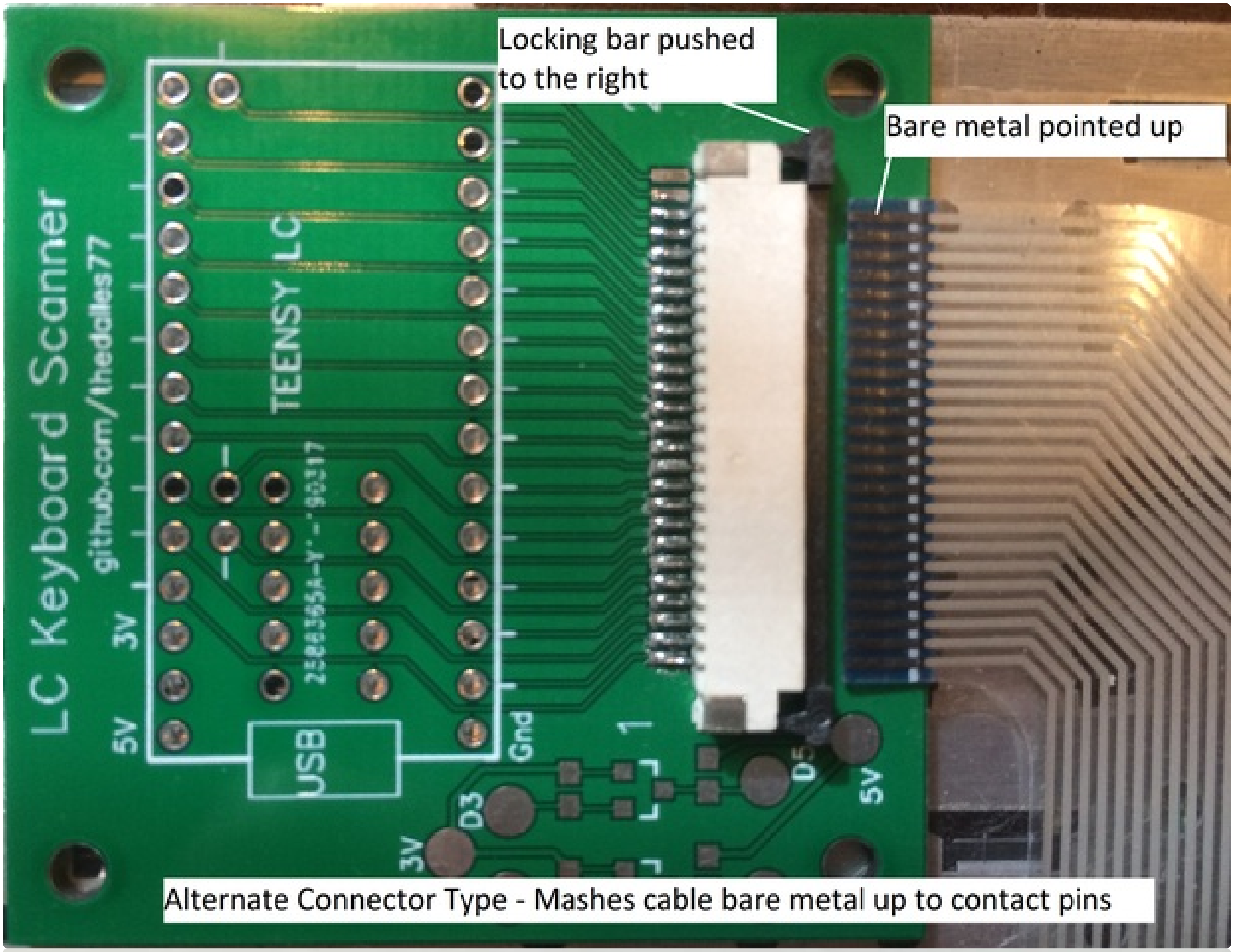
If your keyboard has shorted pins that are in the middle, adjusting max_pin or min_pin will not fix the problem. In that case load the Matrix_Decoder_LC_alternate, Matrix_Decoder_3p2_alternate, or Matrix_Decoder_4p0_alternate routine. These routines allow you to exclude any pins necessary to avoid shorts.

If you are using a Teensy ++2.0 and you get the numbers 1 and 15 reported on your screen, you forgot to unsolder the LED located on the Teensy. The Teensy 3.2 and 4.0 will report 1 and 34 as shorted if you are scanning a 34 pin keyboard cable and the LED has not been removed from the Teensy.

If certain keyboard keys do not give a response when pressed, it is usually caused by a bad solder connection, tarnished cable, or misaligned contacts. To prove the code will report all possible pin connections, use a small flat screwdriver to momentarily short pads 1 and 2 at the FPC connector, right on top of your solder joint. Move on to pads 2 and 3, 3 and 4....., proceeding up to the last two pads, taking note of any pad numbers that don't work. For the pads that don't give a response, reheat the solder joint on the FPC connector pad and if necessary, at the header pin on the Teensy. For the Teensy LC, don't forget to add header pins on I/O's 24, 25, and 26. Use the pin translation table in step 14 to map the FPC pin number to the I/O number on the Teensy. Keep fixing your solder joints until you get a response for every pin when you short them with a screw driver. If some keyboard keys are still not working, there may be a bad connection to the FPC cable. Gently clean the FPC cable with an eraser if the metal traces look tarnished. Use a magnifying glass to inspect inside the FPC connector to see if the contact points are in alignment with the cable traces. You may need to trim the side of the FPC cable to get the proper alignment. If alignment looks good, push down on the FPC connector or lift up on the cable while trying the bad keys to see if you can sometimes make them work. You may need to add a thin piece of paper or tape to the plastic backing on the FPC cable to get a tighter fit. Erratic keys can sometimes be fixed by popping off the key cap and cleaning with isopropyl alcohol. Sometimes the keyboard has a bad key switch that will not give a reliable response, no matter what you do. The only solution is to replace the keyboard.



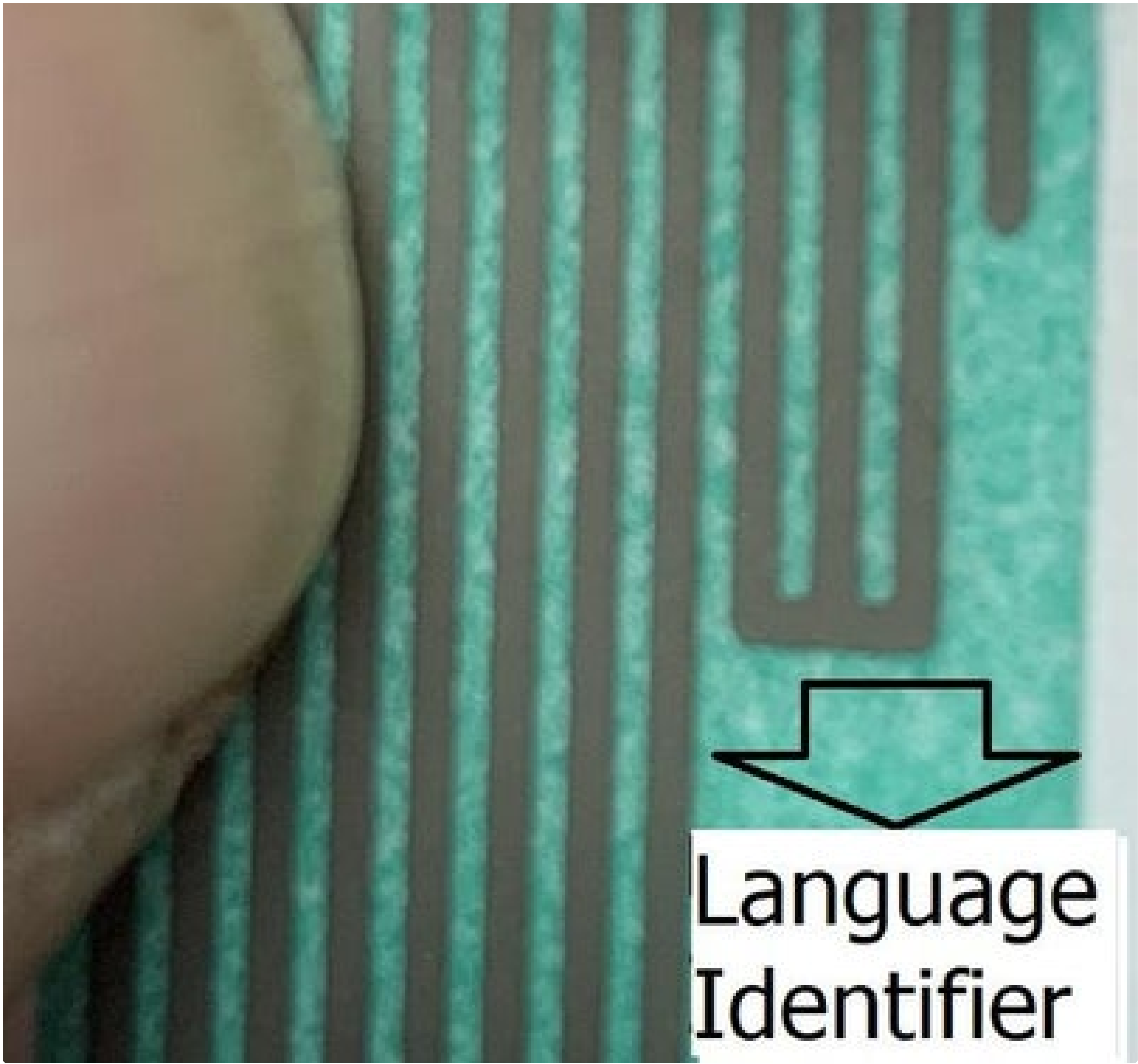
FPC Cable to Connector Procedure



Locking bar pushed to the right

Bare metal pointed up

Alternate Connector Type - Mashes cable bare metal up to contact pins



Download

https://www.instructables.com/FQJ/DDLW/K931G96H/FQJDDLWK931G96H.ino

Download

https://www.instructables.com/FMZ/S858/K931G96I/FMZS858K931G96I.ino

Download

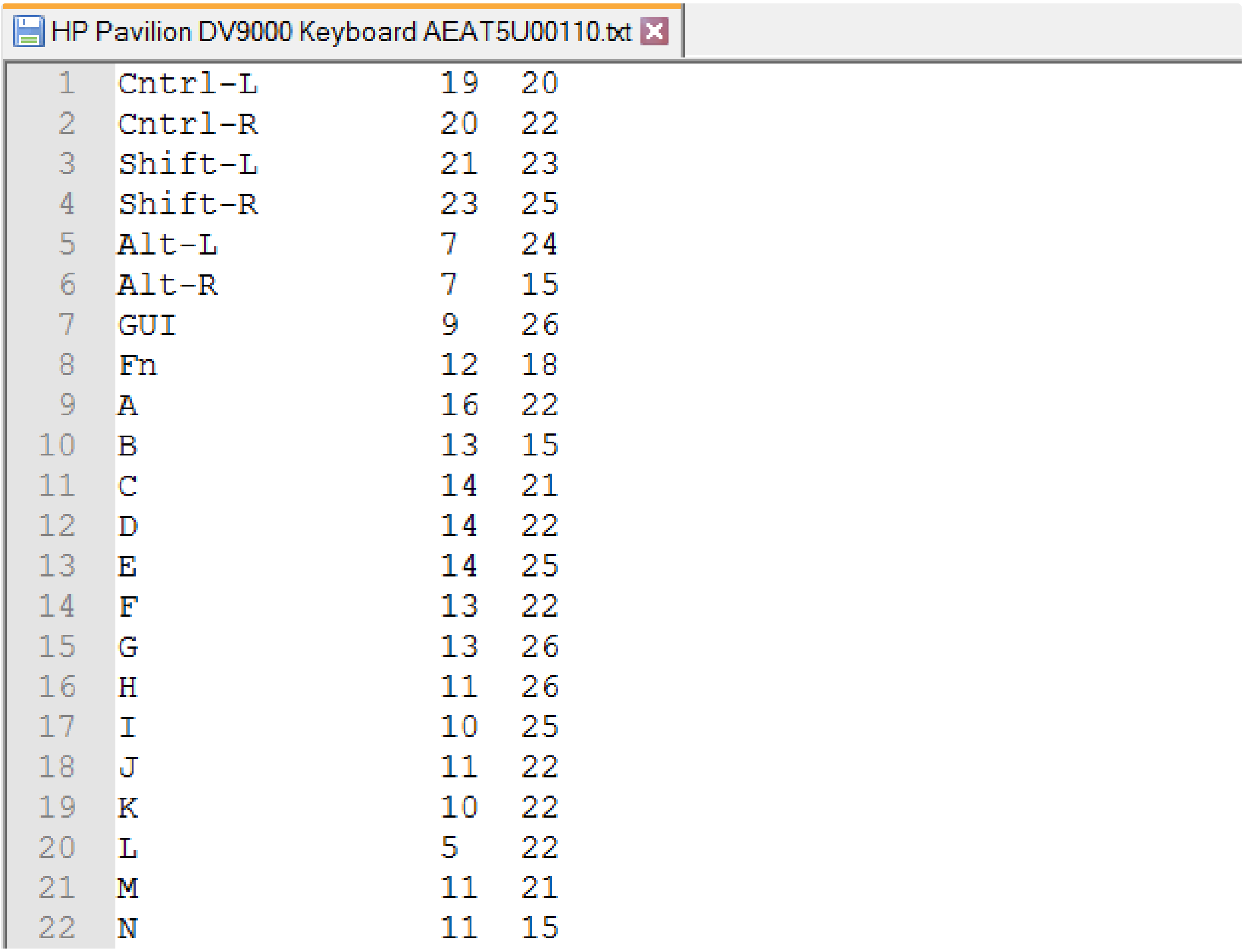
https://www.instructables.com/FGY/BBDJ/K931G96J/FGYBBDJK931G96J.ino

Step 11: Test the Keyboard

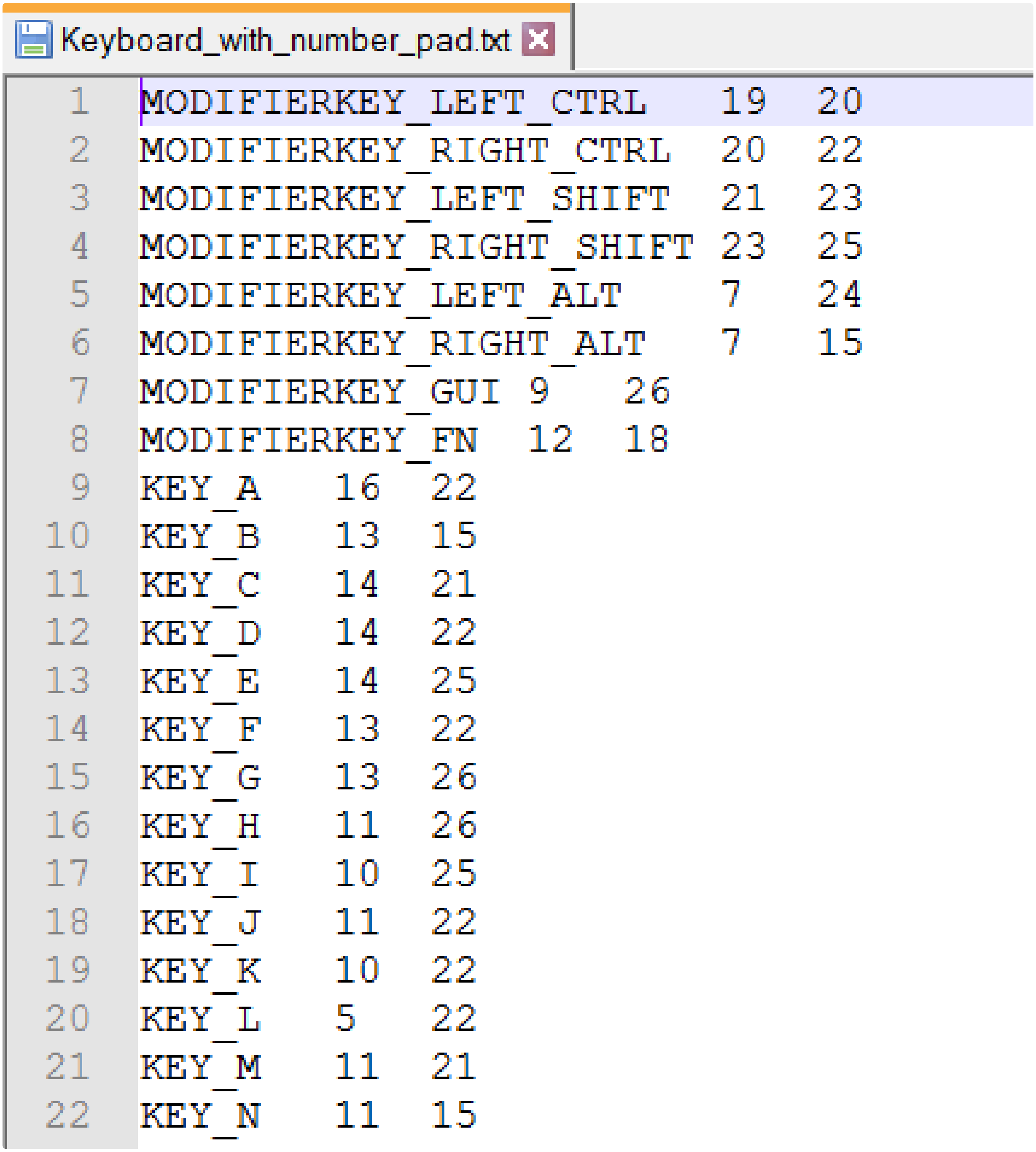
Press each key, one by one on the test keyboard as listed on the editor screen. The Teensy will send two pin numbers over USB that were connected when the key was pressed. The Teensy will then send a down arrow to position the cursor for the next key. If you make a mistake, just use your PC keyboard and mouse to erase the error and re-position the cursor.

After pressing every key on the keyboard and confirming that pin numbers were given for all, save the finished file for analysis. At this point you've created a very thorough keyboard tester.

The original key list on the left gives every key and the results are in columns for transfer to a spreadsheet. Marcel's key list on the right uses the PJRC key names to make it easier for his Python program to build the matrix. If a key is listed that is not on your keyboard, use your mouse or arrow keys on your PC to move the cursor to the next key. The Python program jumps over the unused keys so there's no need to edit them out manually. If you want to assign Media keys (a key event associated with the Fn-key), push the media key (do not push the Fn key). The letters "FN" are between the media keyname and the pin numbers to let the Python program know this belongs in the media matrix.



```
Cntrl-L         19  20
Cntrl-R         20  22
Shift-L         21  23
Shift-R         23  25
Alt-L           7   24
Alt-R           7   15
GUI             9   26
Fn              12  18
A               16  22
B               13  15
C               14  21
D               14  22
E               14  25
F               13  22
G               13  26
H               11  26
I               10  25
J               11  22
K               10  22
L               5   22
M               11  21
N               11  15
```



```
MODIFIERKEY_LEFT_CTRL   19  20
MODIFIERKEY_RIGHT_CTRL  20  22
MODIFIERKEY_LEFT_SHIFT  21  23
MODIFIERKEY_RIGHT_SHIFT 23  25
MODIFIERKEY_LEFT_ALT    7   24
MODIFIERKEY_RIGHT_ALT   7   15
MODIFIERKEY_GUI 9   26
MODIFIERKEY_FN  12  18
KEY_A   16  22
KEY_B   13  15
KEY_C   14  21
KEY_D   14  22
KEY_E   14  25
KEY_F   13  22
KEY_G   13  26
KEY_H   11  26
KEY_I   10  25
KEY_J   11  22
KEY_K   10  22
KEY_L   5   22
KEY_M   11  21
KEY_N   11  15
```

Step 12: Determine Input and Output Pins

If you are using Marcel's Python program, it will automatically determine the input and output pins. The completed pin list file should be in the same directory as the Python program so the start up menu (shown above) will list it. To load and run his Python 3 program on a Raspberry Pi 4, use the Thonny IDE under the programming tab. The results will list the

FPC connector pins that are inputs (columns) and the pins that are outputs (rows) plus it does the translation to the selected Teensy I/O numbers. The key codes are automatically placed into arrays for easy cut and paste into a USB keyboard routine (see matrix examples above).

Sometimes a strange keyboard will confuse Marcel's program and not everyone is familiar with running Python. For those cases or if you are using a Teensy ++2.0, you should use my original manual procedure. The following instructions will determine the keyboard pins based around the Modifier keys; Control, Alt, Shift, GUI, and Fn. As a general rule (that is not always followed) 8 of the keyboard pins will be inputs to the Teensy and the remainder will be outputs. The Modifier keys usually have an output row all to themselves which allows these keys to be held down while other keys are pressed. This avoids a sneak path which would cause ghosting as described in the PDF below. These "rules" are not always followed (especially by the Fn key) so you may need to do some trial and error as you build the matrix. I have lots of keyboard examples at my Github repo to help you out.

Control-Left and Control-Right will have a common pin between them. Example:

Cntrl-L 19 20

Cntrl-R 20 22

The common pin, Pin 20 in this example, will be a Teensy output, and 19 & 22 will be inputs.

Similarly Alt-Left and Alt-Right will have a common pin between them, just as Shift-Left and Shift-Right will also have a common pin. Example:

Alt-L 7 24

Alt-R 7 15

Shift-L 21 23

Shift-R 23 25

The Alt common pin will be a Teensy output, and 15 & 24 will be inputs.

The Shift common pin will be a Teensy output, and 21 & 25 will be inputs.

The GUI key is usually a single key as in this example;

GUI 9 26

Search all the other pins in the list to see if 9 or 26 are used on other keys. In this example, pin 9 was not used for any other key which means it will be a Teensy output and 26 will be an input. Sometimes both pins are used for other keys but one of the pins is used for common keys like letters and numbers and the other pin is used for less-common keys like page-up. In this case the pin used for common keys will be a Teensy input and the other pin will be an output. Note that the GUI key will still work if you swap the pins.

The Fn key is also a single key as in this example;

Fn 12 18

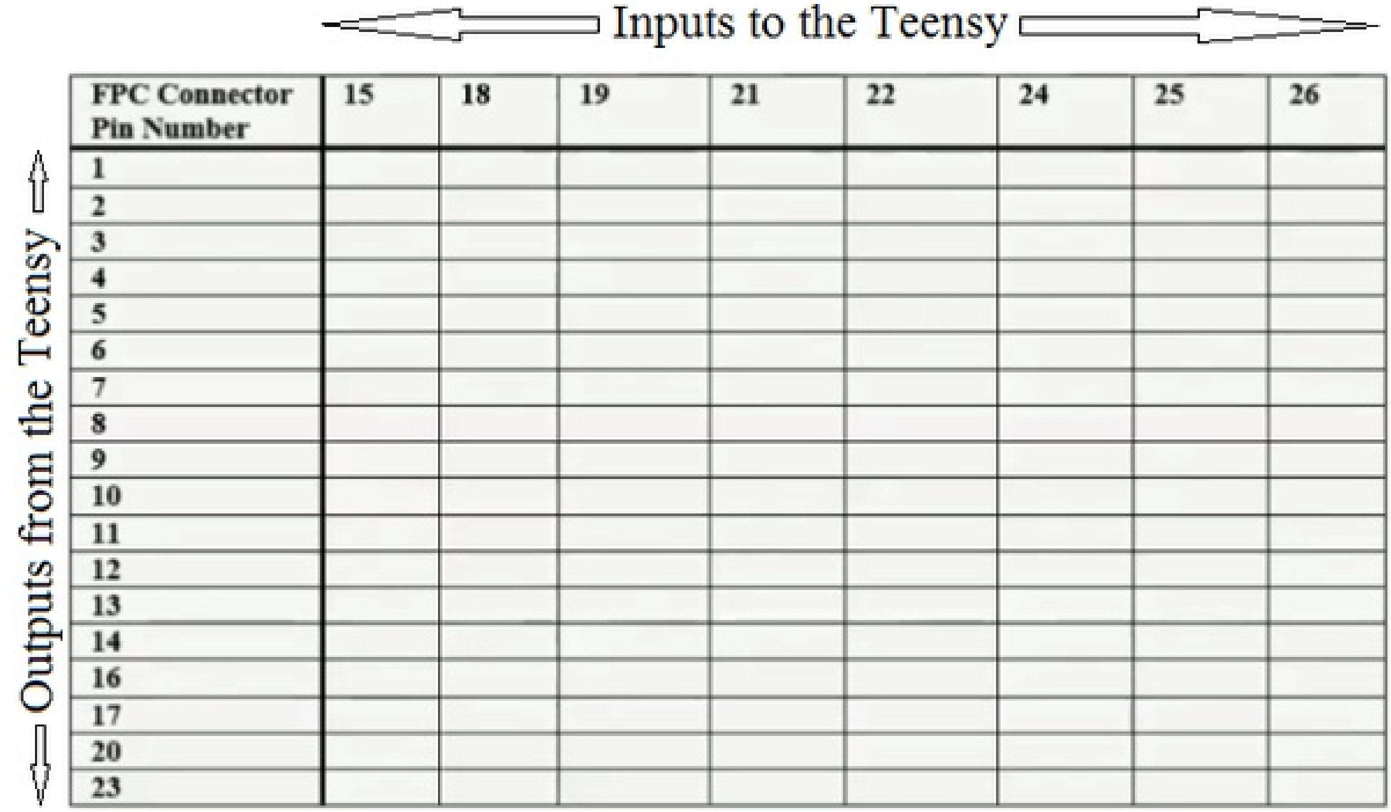
Using the same approach as the GUI key, search all the other pins to see if 12 or 18 are used on other keys. In this example, pin 12 was not used for any other key therefore it will be an output and 18 will be an input. If both pins are used on other keys, follow the same rules as the GUI example. Sometimes both of the Fn pins are used by common keys which means you can pick either pin as an input and the other as an output.

The eight input pins for the HP DV9000 example keyboard have been identified as; 15, 18, 19, 21, 22, 24, 25, and 26. All other pins will be Teensy outputs. Make a keyboard matrix table like the one shown above with the 8 input pins across the top in ascending order and all the other pins as outputs on the side, also in ascending order.

The orientation of the keyboard matrix is just my personal preference. You can swap the rows/columns and inputs/outputs if you want. Swapping pins may be necessary if you have a rare laptop keyboard that has diodes in series with each switch. With diodes, you need to make sure the cathode (first pin listed) is designated an output from the Teensy and the anode (second pin listed) is designated an input to the Teensy.

Sometimes only 7 pins can be identified as inputs because two modifier keys share the same input pin (usually the Shift-R and Control-R). I've even seen keyboards that had the same two input pins for the Shift, Alt, and Control keys so after determining the GUI and Fn input pins, only 4 inputs were identified. For some keyboards, the input pins are grouped together (i.e., 17 thru 24) which makes it easy to fill in the missing pins. Other keyboards have no grouping of pins which means you'll have to begin filling out the matrix even though some input pins are missing. The remaining input pins will be revealed when some of the keys can't be placed. Pick one of the pins from these keys as an input and continue filling out the matrix. If you get stuck, go back and pick the other pin instead.

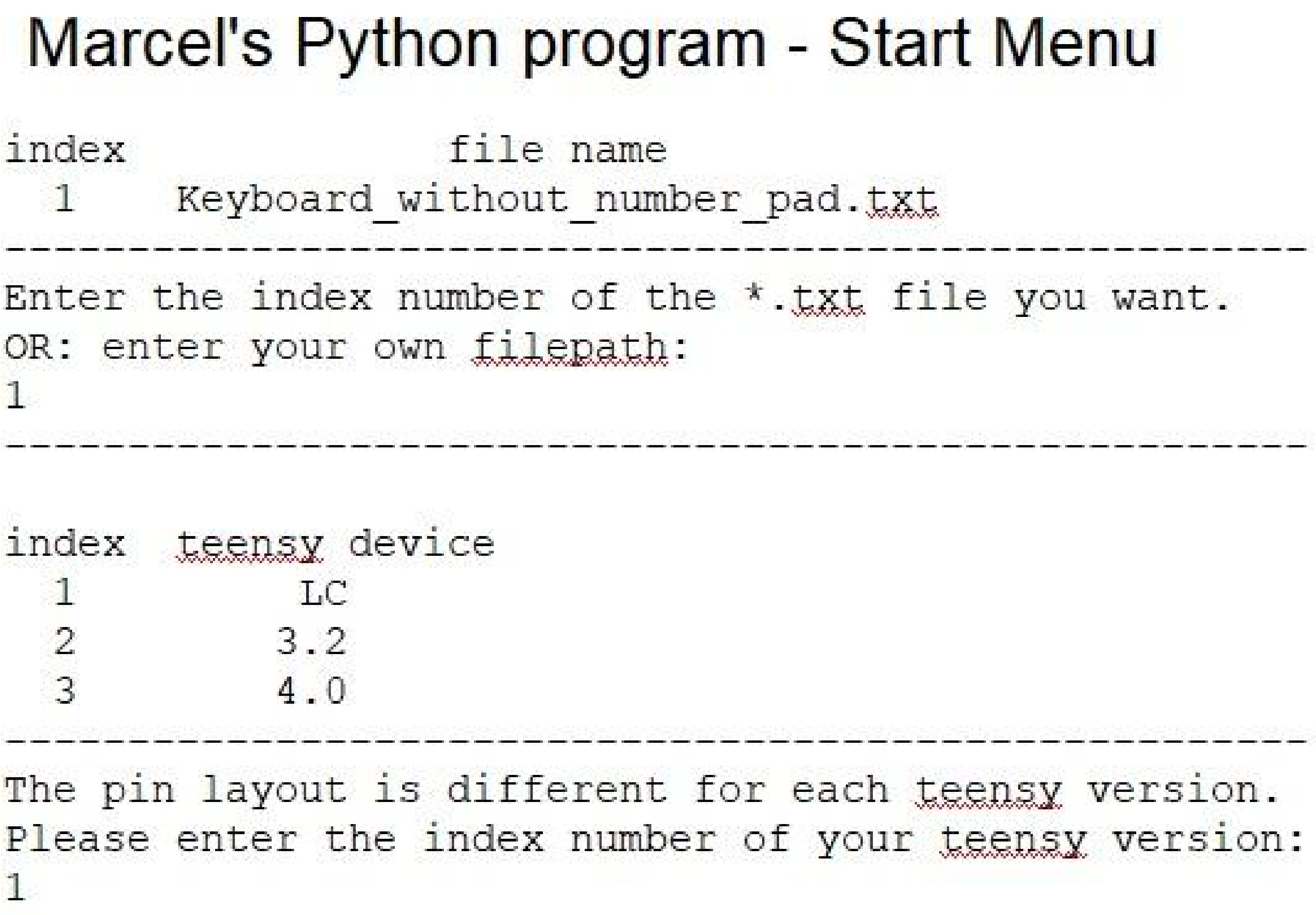
Keyboards with dual FPC cables are the easiest to figure out because the input pins are on the cable with fewer pins and the output pins are on the other cable.



Inputs to the Teensy

Outputs from the Teensy

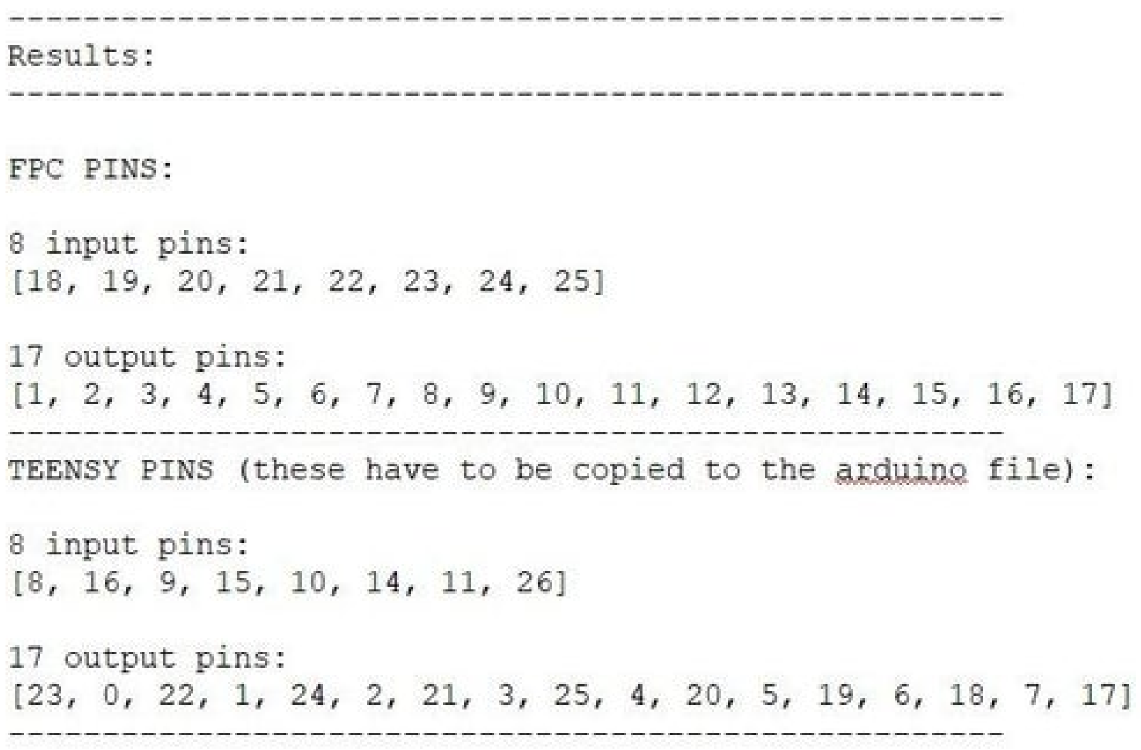
FPC Connector Pin Number	15	18	19	21	22	24	25	26
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
16								
17								
20								
23								



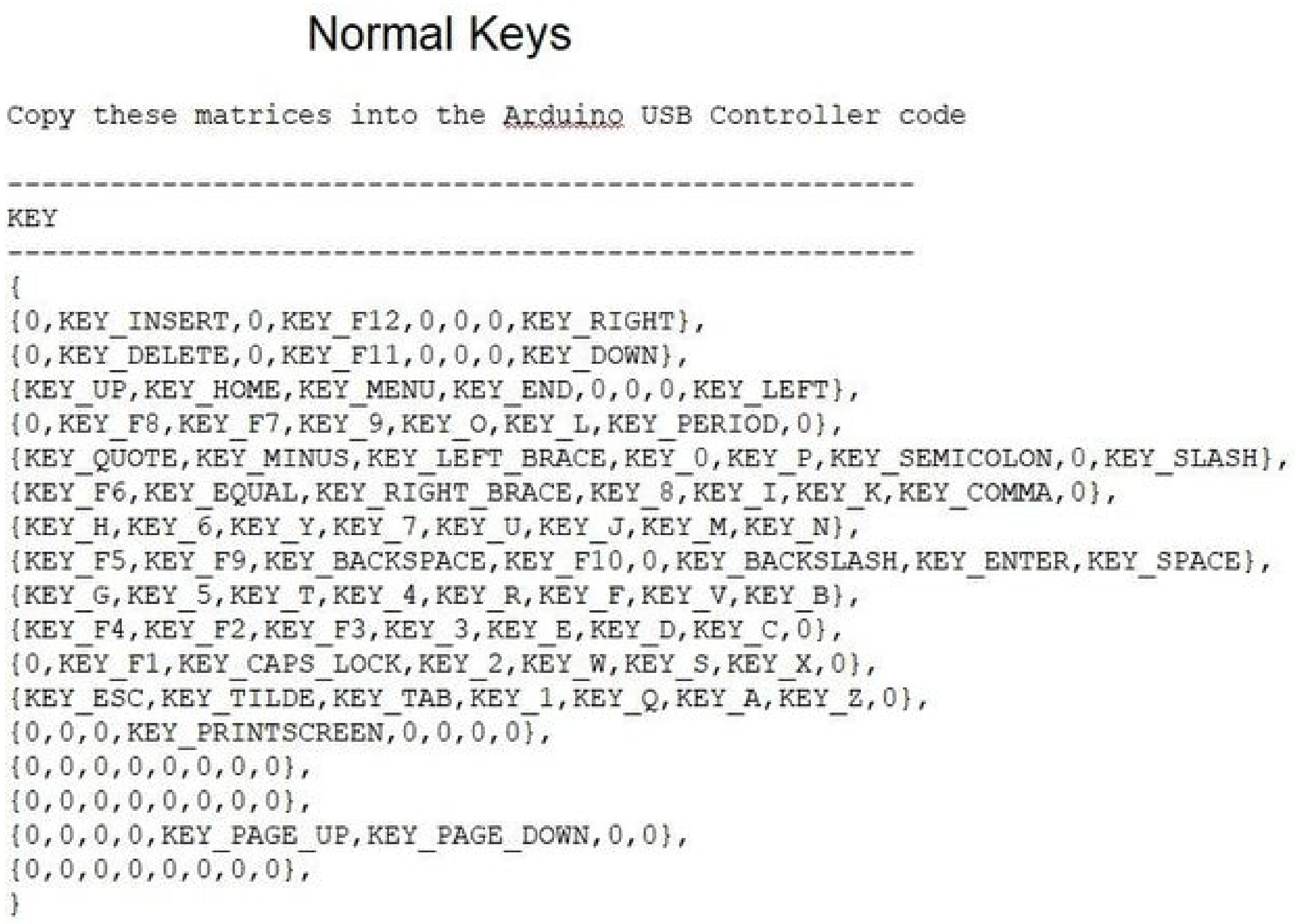
Marcel's Python program - Start Menu

```
index              file name
  1    Keyboard_without_number_pad.txt
-----------------------------------------------------
Enter the index number of the *.txt file you want.
OR: enter your own filepath:
1
-----------------------------------------------------

index  teensy device
  1          LC
  2         3.2
  3         4.0
-----------------------------------------------------
The pin layout is different for each teensy version.
Please enter the index number of your teensy version:
1
```

FPC and Teensy Pins

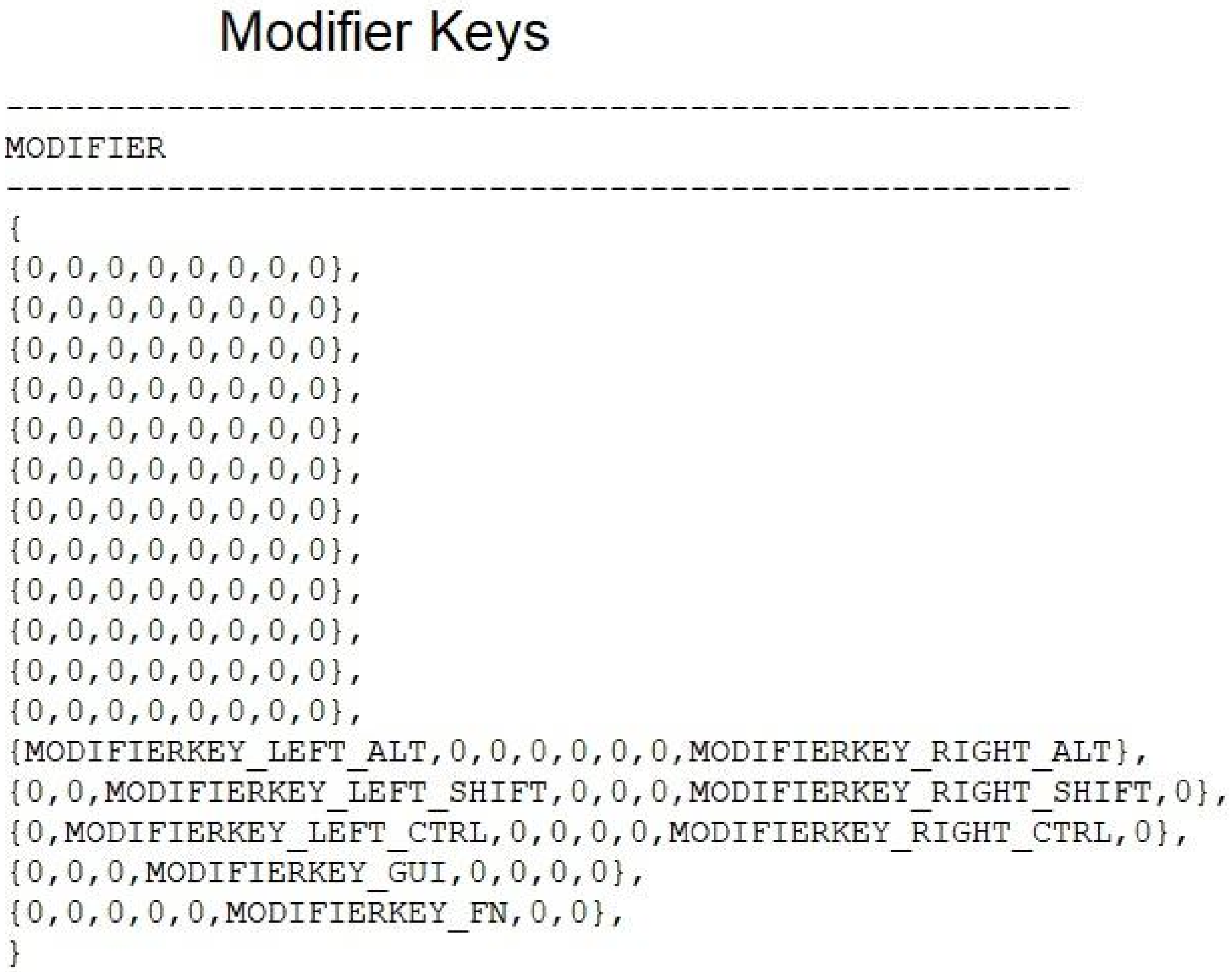


```
-----------------------------------------------------
Results:
-----------------------------------------------------

FPC PINS:

8 input pins:
[18, 19, 20, 21, 22, 23, 24, 25]

17 output pins:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
-----------------------------------------------------
TEENSY PINS (these have to be copied to the arduino file):

8 input pins:
[8, 16, 9, 15, 10, 14, 11, 26]

17 output pins:
[23, 0, 22, 1, 24, 2, 21, 3, 25, 4, 20, 5, 19, 6, 18, 7, 17]
-----------------------------------------------------
```



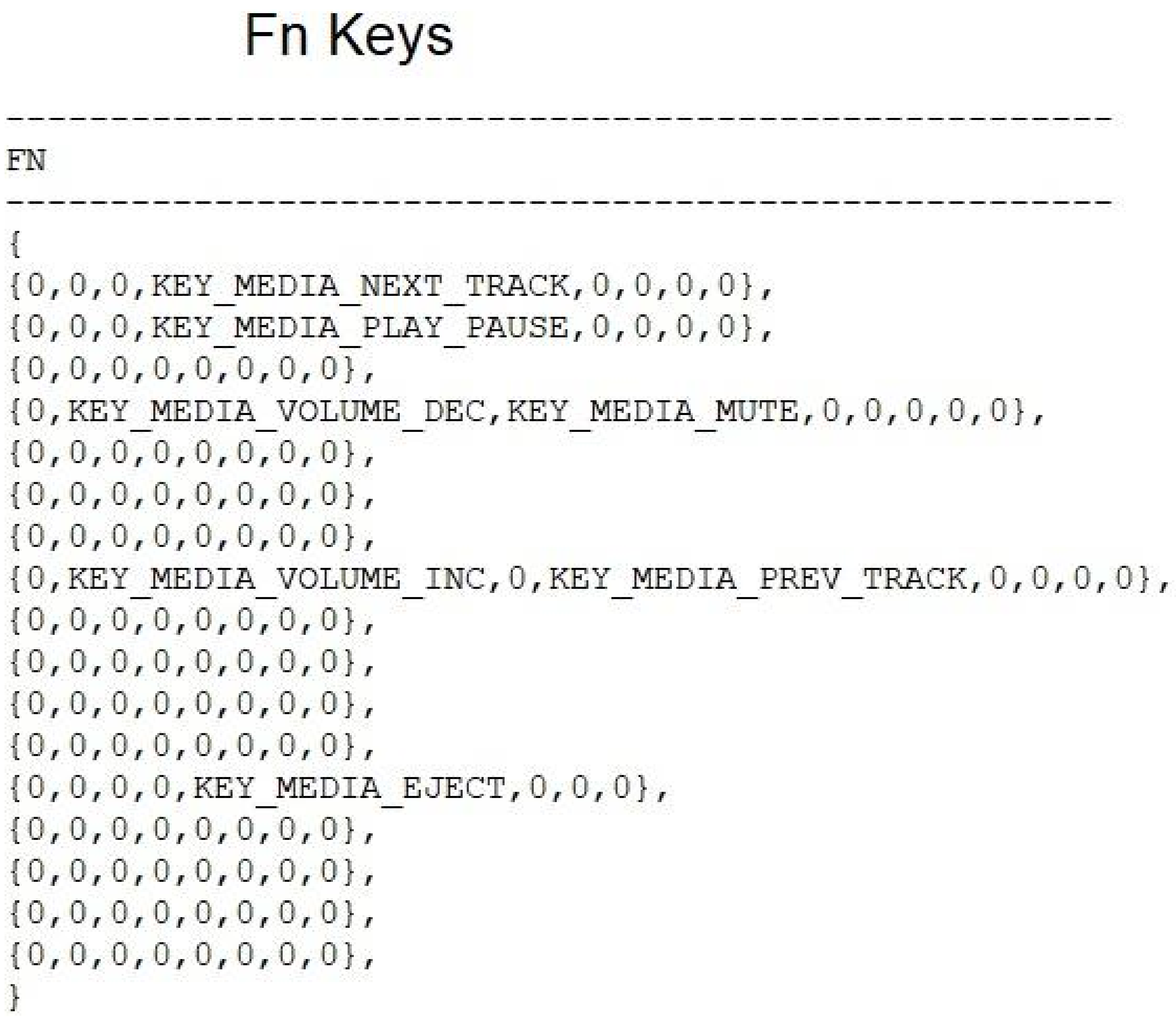
Normal Keys

```
Copy these matrices into the Arduino USB Controller code

----------------------------------------------------------
KEY
----------------------------------------------------------
{
{0,KEY_INSERT,0,KEY_F12,0,0,0,KEY_RIGHT},
{0,KEY_DELETE,0,KEY_F11,0,0,0,KEY_DOWN},
{KEY_UP,KEY_HOME,KEY_MENU,KEY_END,0,0,0,KEY_LEFT},
{0,KEY_F8,KEY_F7,KEY_9,KEY_O,KEY_L,KEY_PERIOD,0},
{KEY_QUOTE,KEY_MINUS,KEY_LEFT_BRACE,KEY_0,KEY_P,KEY_SEMICOLON,0,KEY_SLASH},
{KEY_F6,KEY_EQUAL,KEY_RIGHT_BRACE,KEY_8,KEY_I,KEY_K,KEY_COMMA,0},
{KEY_H,KEY_6,KEY_Y,KEY_7,KEY_U,KEY_J,KEY_M,KEY_N},
{KEY_F5,KEY_F9,KEY_BACKSPACE,KEY_F10,0,KEY_BACKSLASH,KEY_ENTER,KEY_SPACE},
{KEY_G,KEY_5,KEY_T,KEY_4,KEY_R,KEY_F,KEY_V,KEY_B},
{KEY_F4,KEY_F2,KEY_F3,KEY_3,KEY_E,KEY_D,KEY_C,0},
{0,KEY_F1,KEY_CAPS_LOCK,KEY_2,KEY_W,KEY_S,KEY_X,0},
{KEY_ESC,KEY_TILDE,KEY_TAB,KEY_1,KEY_Q,KEY_A,KEY_Z,0},
{0,0,0,KEY_PRINTSCREEN,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,KEY_PAGE_UP,KEY_PAGE_DOWN,0,0},
{0,0,0,0,0,0,0,0},
}
```



Modifier Keys

```
-----------------------------------------------------
MODIFIER
-----------------------------------------------------
{
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{MODIFIERKEY_LEFT_ALT,0,0,0,0,0,0,MODIFIERKEY_RIGHT_ALT},
{0,0,MODIFIERKEY_LEFT_SHIFT,0,0,0,MODIFIERKEY_RIGHT_SHIFT,0},
{0,MODIFIERKEY_LEFT_CTRL,0,0,0,0,MODIFIERKEY_RIGHT_CTRL,0},
{0,0,0,MODIFIERKEY_GUI,0,0,0,0},
{0,0,0,0,0,MODIFIERKEY_FN,0,0},
}
```



Fn Keys

```
---------------------------------------------------------
FN
---------------------------------------------------------
{
{0,0,0,KEY_MEDIA_NEXT_TRACK,0,0,0,0},
{0,0,0,KEY_MEDIA_PLAY_PAUSE,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,KEY_MEDIA_VOLUME_DEC,KEY_MEDIA_MUTE,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,KEY_MEDIA_VOLUME_INC,0,KEY_MEDIA_PREV_TRACK,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,KEY_MEDIA_EJECT,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
}
```

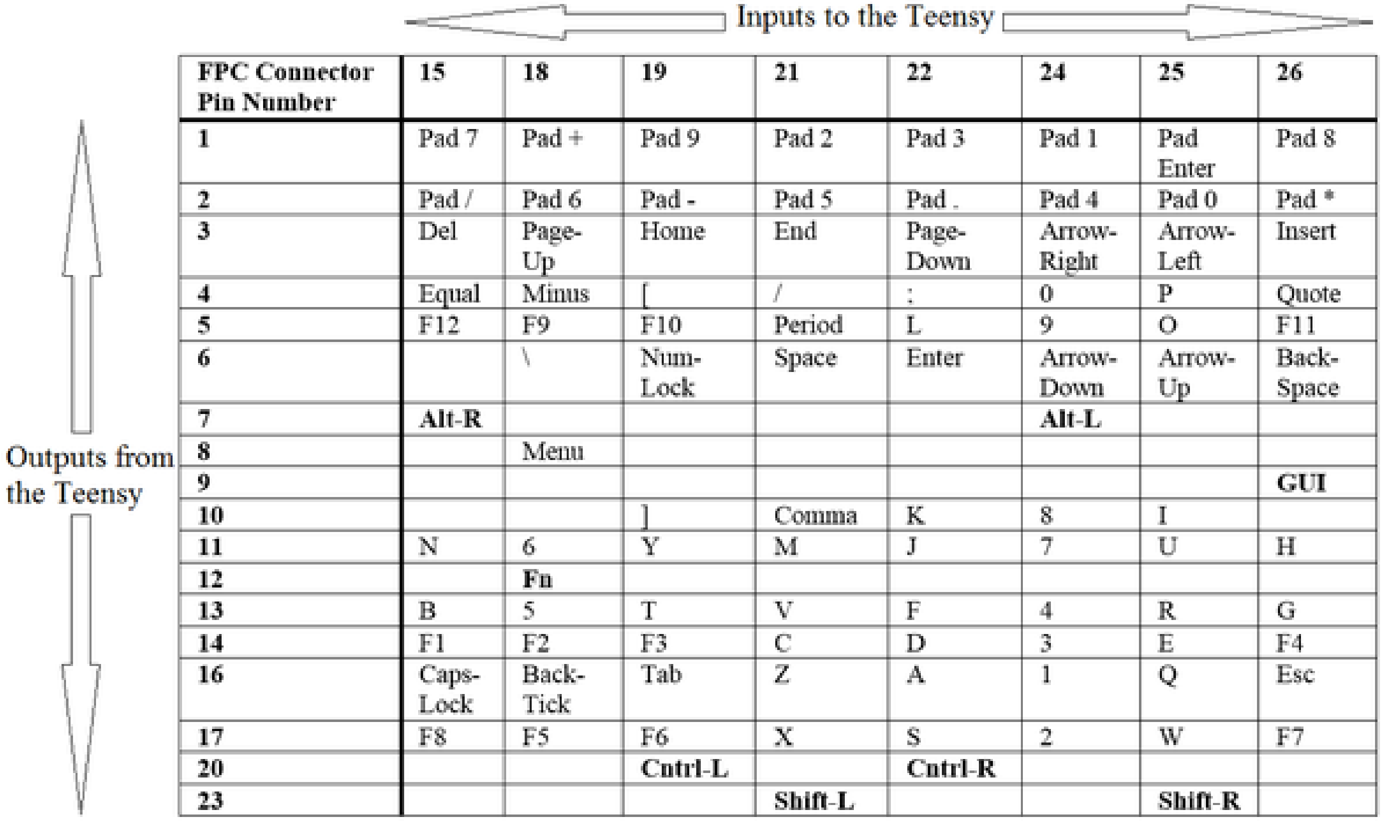
Download

https://www.instructables.com/FI3/UICL/KU18FXP6/FI3UICLKU18FXP6.pdf

Step 13: Fill the Matrix With Keys

To manually fill the matrix, place each key name at the row/column intersection of the pins as shown in the HP DV9000 keyboard example given above. The modifier keys are in bold to make it easy to see that they have a row all to themselves. This keyboard followed the "rules" exactly.

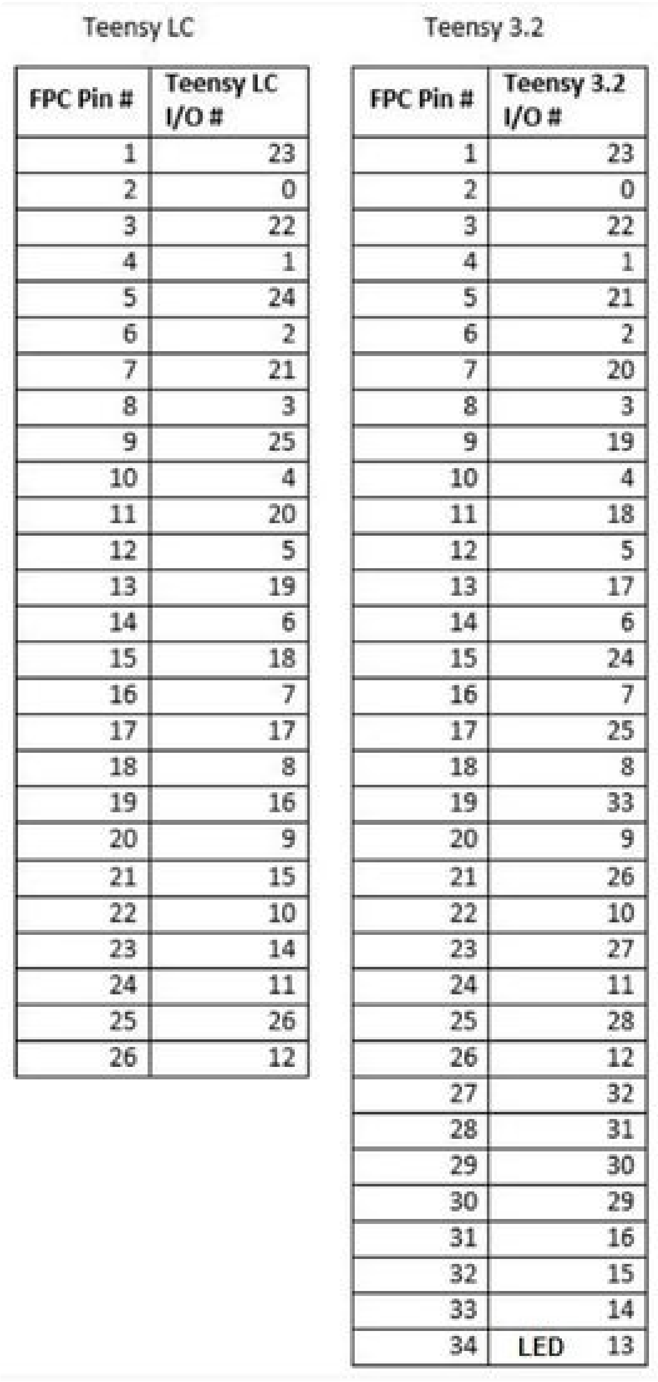
You don't need to fill out the matrix if you're using Marcel's Python 3 program.


Inputs to the Teensy
Outputs from the Teensy

FPC Connector Pin Number	**15**	**18**	**19**	**21**	**22**	**24**	**25**	**26**
1	Pad 7	Pad +	Pad 9	Pad 2	Pad 3	Pad 1	Pad Enter	Pad 8
2	Pad /	Pad 6	Pad -	Pad 5	Pad .	Pad 4	Pad 0	Pad *
3	Del	Page-Up	Home	End	Page-Down	Arrow-Right	Arrow-Left	Insert
4	Equal	Minus	[	/	;	0	P	Quote
5	F12	F9	F10	Period	L	9	O	F11
6		\	Num-Lock	Space	Enter	Arrow-Down	Arrow-Up	Back-Space
7	**Alt-R**					**Alt-L**		
8		Menu						
9								**GUI**
10			]	Comma	K	8	I	
11	N	6	Y	M	J	7	U	H
12		**Fn**						
13	B	5	T	V	F	4	R	G
14	F1	F2	F3	C	D	3	E	F4
16	Caps-Lock	Back-Tick	Tab	Z	A	1	Q	Esc
17	F8	F5	F6	X	S	2	W	F7
20			**Cntrl-L**		**Cntrl-R**			
23				**Shift-L**			**Shift-R**	

Step 14: Translate FPC Pin Numbers to Teensy I/O Numbers

Marcel's Python program automatically translates the FPC pins to the Teensy I/O's. If you are using the manual method, you will need to use the Teensy LC, 3.2, or 4.0 tables shown above. The Teensy ++2.0 uses the manual method. Notice that there is an LED on the Teensy 3.2, 4.0, and ++2.0 that must be removed if you are using that I/O pin.

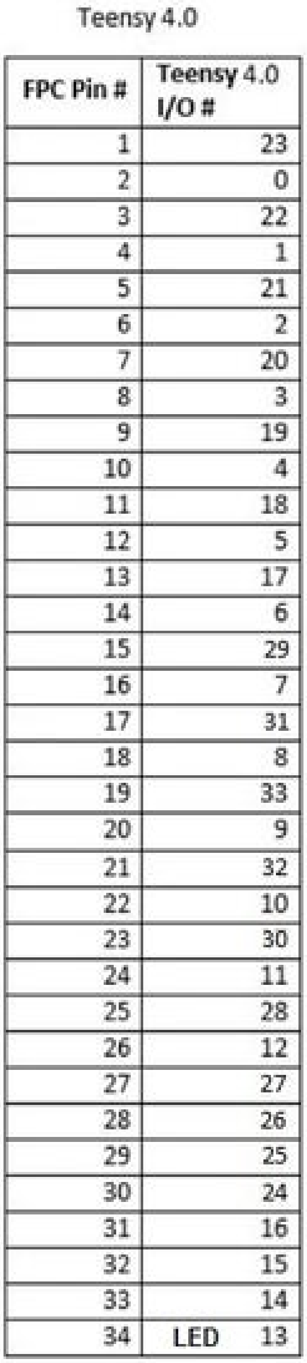


Teensy LC

FPC Pin #	Teensy LC I/O #
1	23
2	0
3	22
4	1
5	24
6	2
7	21
8	3
9	25
10	4
11	20
12	5
13	19
14	6
15	18
16	7
17	17
18	8
19	16
20	9
21	15
22	10
23	14
24	11
25	26
26	12

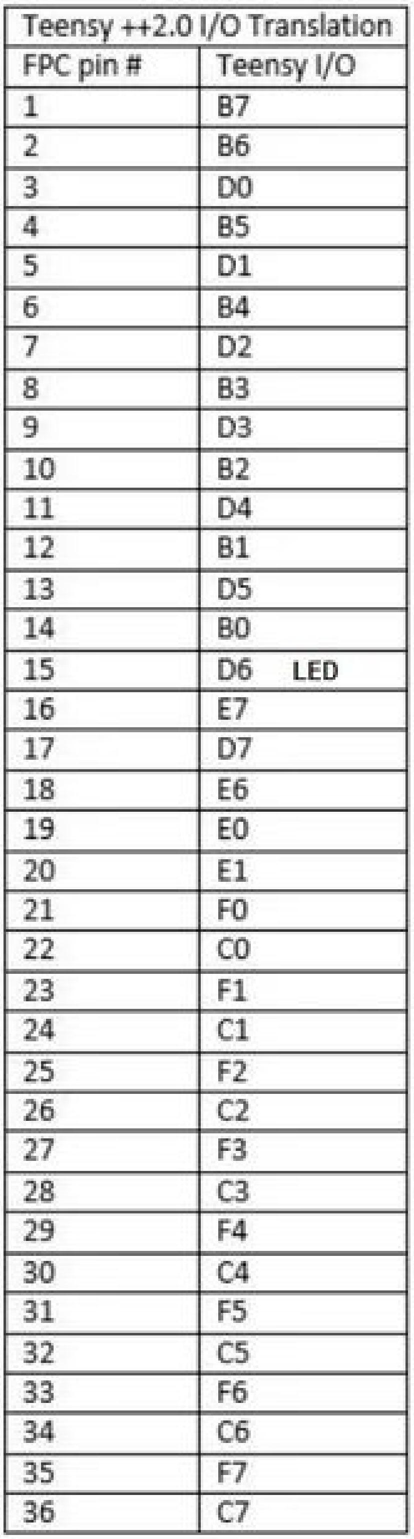
Teensy 3.2

FPC Pin #	Teensy 3.2 I/O #
1	23
2	0
3	22
4	1
5	21
6	2
7	20
8	3
9	19
10	4
11	18
12	5
13	17
14	6
15	24
16	7
17	25
18	8
19	33
20	9
21	26
22	10
23	27
24	11
25	28
26	12
27	32
28	31
29	30
30	29
31	16
32	15
33	14
34	LED 13



Teensy 4.0

FPC Pin #	Teensy 4.0 I/O #
1	23
2	0
3	22
4	1
5	21
6	2
7	20
8	3
9	19
10	4
11	18
12	5
13	17
14	6
15	29
16	7
17	31
18	8
19	33
20	9
21	32
22	10
23	30
24	11
25	28
26	12
27	27
28	26
29	25
30	24
31	16
32	15
33	14
34	LED 13



Teensy ++2.0 I/O Translation	
FPC pin #	Teensy I/O
1	B7
2	B6
3	D0
4	B5
5	D1
6	B4
7	D2
8	B3
9	D3
10	B2
11	D4
12	B1
13	D5
14	B0
15	D6 LED
16	E7
17	D7
18	E6
19	E0
20	E1
21	F0
22	C0
23	F1
24	C1
25	F2
26	C2
27	F3
28	C3
29	F4
30	C4
31	F5
32	C5
33	F6
34	C6
35	F7
36	C7

Step 15: Load a USB Keyboard Routine Into the Teensy

A Deskthority post from "flabbergast" describes using the ChibiOS development environment to configure TMK for ARM based processors like those used on the Teensy. A toolchain such as the GNU ARM Embedded Toolchain is used to compile the Teensy code. You will need to install the ChibiOS development environment per these instructions. The teensy_lc_onekey example details the steps to create a working TMK build. The QMK keyboard routine is based on TMK and it also has ChibiOS support for the Teensy LC and 3.2. QMK offers timestamp based debouncing with various options set when compiling. There is a Complete Newbs Guide for QMK.

Jay Thompson has created a QMK fork that gives all the information for his Teensy 3.2/Lenovo T420 keyboard project. Jay provides his build environment setup and the make instructions so you have an example to modify for your keyboard.

Gordon has made a Hackaday.IO project called "QMK Powered Laptop Keyboard" which details the "gymnastics" of using QMK.

The TMK/QMK keyboard software is very powerful with tons of features but it can be confusing, (at least to me). As an alternative, I wrote an Arduino USB keyboard routine using the Teensyduino “Micro-Manager” functions. There's just 1 file to load using the Arduino IDE and it's only about 385 lines with lots of comments. I'm a hardware guy so expect ugly code but it provides a basic keyboard controller with 6 key rollover that you can modify to suite your needs. Key debouncing is handled using a 30 msec scan rate. The files named "Instructions for modifying the Teensyduino LC code, Teensyduino 3p2 code, Teensyduino 4p0 code, and Teensyduino pp2p0 code" describe the changes you need to make for your matrix. The instructions also detail how to use the results from Marcel's Python program.

Every keyboard listed below has a folder at my repo containing a Teensyduino USB keyboard routine giving you lots of examples. Pick the one that's closest to your keyboard and use its code as your starting point, (they're all based on the same basic routine). The 1525, 2100, and DV9000 folders also have Marcel's completed key list file and the results from his Python program. The D630 folders include a Teensy 4.0 keyboard controller routine in addition to a Teensy 3.2 routine. The Sony Vaio PCG-K25 folder includes a Teensy ++2.0 keyboard controller routine in addition to a Teensy LC routine. If you are using any of the other keyboard routines as a starting point for a Teensy ++2.0, you must change the "int" arrays to "unsigned int" or it will give compilation errors.

- Dell Inspiron 1525 - Keyboard Part Number D9K01
- Dell Latitude 131L - Keyboard Part Number V-0511BIAS1-US
- Dell Latitude X1 - Keyboard Part Number 0M6607
- Dell Latitude D630 - Keyboard (Teensy 3.2 and 4.0 example)
- HP Compaq Presario 2100 - Keyboard Part Number AEKT1TPU011
- HP Compaq Presario V4000 - Keyboard Part Number NSK-H3L01
- HP Pavilion DV9000 - Keyboard Part number AEAT5U00110
- HP ZV6000 - Keyboard with Italian layout
- Sony Vaio PCG-K25 - Keyboard (Teensy LC and ++2.0 example)
- Sony Vaio VPCCW - Keyboard Part Number 148754321
- Sony Vaio VPCEA - Keyboard part number A-1765-621-A
- Sony Vaio VPCEB4 – Keyboard part number A-1766-425-A
- Sony Vaio P - Keyboard part number N860-7885-T001
- Lenovo ThinkPad T61 – Keyboard part number 42T3177
- Toshiba T1200 Keyboard
- Toshiba 2415 Keyboard
- Fujitsu Lifebook 755Tx Keyboard
- MSI GE63 Raider RGB 8RE Keyboard
- Acer Extensa 2509 - Keyboard part number V121702AS4 UI
- Heathkit H89 Keyboard
- Zenith Supersport SX Keyboard
- Alienware 17 p18e001 Keyboard
- Microsoft Surface Comfort Keyboard contributed by Stephen Sweetland

IBM380ED - Brian Chan and I teamed up to convert an IBM 380ED keyboard and trackpoint to USB as shown in this video.

The code, Eagle files, and project description are in a folder at my repo. This project was documented at Hackaday.IO and Hackaday.com. Stephen designed a stand alone case for the 380 that you can 3D print with his files at thingverse. Brian has converted his IBM 380ED into a KVM, documented at Hackaday.IO.

Commodore 64 - Olga modified my code for a Commodore 64 keyboard. The Teensy LC code, key list files and project description are in a folder at my repo.

Compaq 286 - Olga also modified my code for an old Compaq 286 keyboard but he did much more. Now the program will take the pin number results from running the keylist text file so you don't need to create a key matrix. The code and PDF description are in a folder at my repo.

Blackberry Q10 & HP Jornada & Atari Porfolio - T Caschy has modified my Teensy code to work with these keyboards. The code is in a folder at my repo.

Lenovo Y480 - Luigi Caradonna has modified my Teensy LC code to work with a Lenovo Y480 with an Italian layout and backlight control. The code is in a folder at my repo. Luigi also made a stunning marble base for this keyboard. Pictures are in the "I made it" section at the end of this Instructable.

Lenovo X1 Carbon - Maykro has modified my Teensy ++2.0 code to work with a Lenovo X1 Carbon. The PS/2 trackpoint is working in "stream mode" with adjustable speed. The code is at his repo.

Apple eMate 300 - Billy67 has modified my Teensy code to work with an Apple eMate 300 keyboard. The code is in a folder at my repo. Pictures are in the "I made it" section at the end of this Instructable. Check out his very thorough video plus reviews from Cult of Mac and The Register to see how he made a Raspberry Pi laptop from an Apple eMate 300.

Apple iBook G3 Clamshell - Billy67 used my special connector board for this keyboard. See step 18 for the details.

Apple Powerbook models 100, 140 through 180, and 520 - Billy67 used a Teensy 3.2 to talk to these Apple keyboard and trackball units. Pictures are in the "I made it" section at the end of this Instructable. The trackball uses an Apple Desktop Bus (ADB) interface that is controlled by the Teensy after it scans the keyboard matrix. The Teensy code for the Powerbook 100, Powerbook 520, and for Powerbooks 140 thru 180 are at my repo.

GRiD 1550 UK - SimonT192 has modified my Teensy LC code to work with a GRiD 1550 keyboard with a UK layout. This laptop has a rotary encoded "IsoPoint" mouse which has been converted to USB using a Teensy LC. The code for the keyboard and mouse are in a folder at my repo along with a PDF describing the operation. Watch Simon's YouTube video for a very detailed description of his "Aliens Sentry Gun" GRiD laptop build. Hackaday featured his project as well.

GRiD 1550 US - Chris has modified his GRiD 1550 US keyboard to work with a Teensy LC. The code for the keyboard is in a folder at my repo along with a PDF. Chris was also able to get the IsoPoint mouse working.

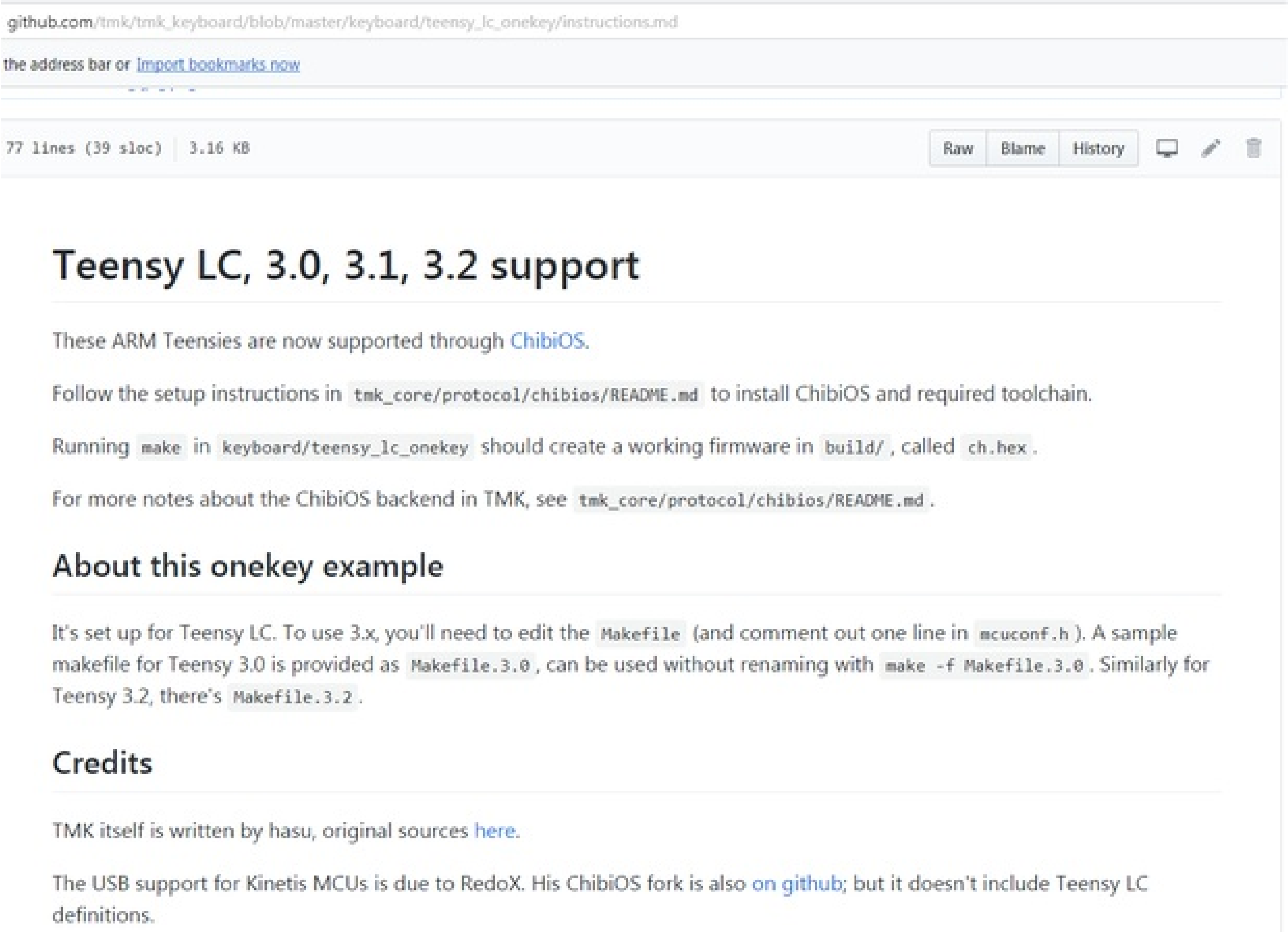
GRiDCase 3 - Lucas and I converted his Bloomberg GRiDCase 3 laptop keyboard to USB. The project documentation and code is at my repo. The financial spreadsheet keys were redefined but some of the unique keys were maintained. It does not have a right shift key. The "CODE" key acts as a Figure Shift which allows access to the symbols on the alphabet keys. Shifted numbers do not send anything but shifted alphabet keys send the upper case letter. Lucas is accustomed to this key layout thanks to his day job of repairing typewriters. Check out his project writeup for a blow by blow description.

IBM Thinkpad T43 - Nat has modified my Teensy code to work with a Thinkpad T43 keyboard and he has created a new Teensy 3.2 connector board using KiCad. The code, layout, and PDF description are in a folder at my repo.

IBM Thinkpad T41 - I created a new Teensy 4.0 connector board for a T41 which will also work on a T42 and T43. In addition to controlling the keyboard and trackpoint, the code also controls the touchpad. The code, circuit boards, and PDF are in a folder at my repo.

Psion MC400 - Zedstarr used my Teensy LC code on the mechanical keyboard from a Psion MC400. His website gives all the details including a video link.

Logitech K120 - I replaced the original controller from this desktop keyboard with a Teensy 4.1 and documented the results at Hackster.


github.com/tmk/tmk_keyboard/blob/master/keyboard/teensy_lc_onekey/instructions.md
the address bar or Import bookmarks now
77 lines (39 sloc) | 3.16 KB
Raw
Blame
History

Teensy LC, 3.0, 3.1, 3.2 support

These ARM Teensies are now supported through ChibiOS.

Follow the setup instructions in `tmk_core/protocol/chibios/README.md` to install ChibiOS and required toolchain.

Running `make` in `keyboard/teensy_lc_onekey` should create a working firmware in `build/`, called `ch.hex`.

For more notes about the ChibiOS backend in TMK, see `tmk_core/protocol/chibios/README.md`.

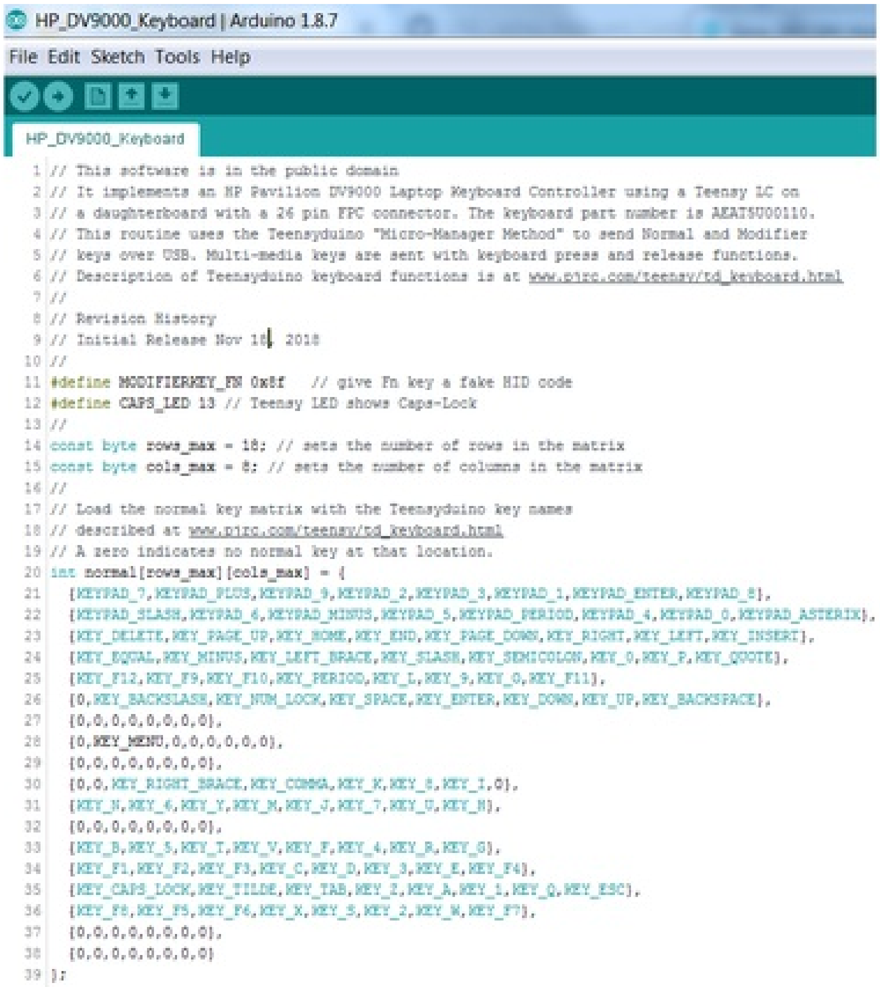
About this onekey example

It's set up for Teensy LC. To use 3.x, you'll need to edit the `Makefile` (and comment out one line in `mcuconf.h`). A sample makefile for Teensy 3.0 is provided as `Makefile.3.0`, can be used without renaming with `make -f Makefile.3.0`. Similarly for Teensy 3.2, there's `Makefile.3.2`.

Credits

TMK itself is written by hasu, original sources here.

The USB support for Kinetis MCUs is due to RedoX. His ChibiOS fork is also on github; but it doesn't include Teensy LC definitions.


HP_DV9000_Keyboard | Arduino 1.8.7
File Edit Sketch Tools Help
HP_DV9000_Keyboard

```
// This software is in the public domain
// It implements an HP Pavilion DV9000 Laptop Keyboard Controller using a Teensy LC on
// a daughterboard with a 26 pin FPC connector. The keyboard part number is AEAT5U00110.
// This routine uses the Teensyduino "Micro-Manager Method" to send Normal and Modifier
// keys over USB. Multi-media keys are sent with keyboard press and release functions.
// Description of Teensyduino keyboard functions is at www.pjrc.com/teensy/td_keyboard.html
//
// Revision History
// Initial Release Nov 18, 2018
//
#define MODIFIERKEY_FN 0x8f   // give Fn key a fake HID code
#define CAPS_LED 13 // Teensy LED shows Caps-Lock
//
const byte rows_max = 18; // sets the number of rows in the matrix
const byte cols_max = 8; // sets the number of columns in the matrix
//
// Load the normal key matrix with the Teensyduino key names
// described at www.pjrc.com/teensy/td_keyboard.html
// A zero indicates no normal key at that location.
int normal[rows_max][cols_max] = {
  {KEYPAD_7,KEYPAD_PLUS,KEYPAD_9,KEYPAD_2,KEYPAD_3,KEYPAD_1,KEYPAD_ENTER,KEYPAD_8},
  {KEYPAD_SLASH,KEYPAD_6,KEYPAD_MINUS,KEYPAD_5,KEYPAD_PERIOD,KEYPAD_4,KEYPAD_0,KEYPAD_ASTERIX},
  {KEY_DELETE,KEY_PAGE_UP,KEY_HOME,KEY_END,KEY_PAGE_DOWN,KEY_RIGHT,KEY_LEFT,KEY_INSERT},
  {KEY_EQUAL,KEY_MINUS,KEY_LEFT_BRACE,KEY_SLASH,KEY_SEMICOLON,KEY_0,KEY_P,KEY_QUOTE},
  {KEY_F12,KEY_F9,KEY_F10,KEY_PERIOD,KEY_L,KEY_9,KEY_O,KEY_F11},
  {0,KEY_BACKSLASH,KEY_NUM_LOCK,KEY_SPACE,KEY_ENTER,KEY_DOWN,KEY_UP,KEY_BACKSPACE},
  {0,0,0,0,0,0,0,0},
  {0,KEY_MENU,0,0,0,0,0,0},
  {0,0,0,0,0,0,0,0},
  {0,0,KEY_RIGHT_BRACE,KEY_COMMA,KEY_K,KEY_8,KEY_I,0},
  {KEY_N,KEY_6,KEY_Y,KEY_M,KEY_J,KEY_7,KEY_U,KEY_H},
  {0,0,0,0,0,0,0,0},
  {KEY_B,KEY_5,KEY_T,KEY_V,KEY_F,KEY_4,KEY_R,KEY_G},
  {KEY_F1,KEY_F2,KEY_F3,KEY_C,KEY_D,KEY_3,KEY_E,KEY_F4},
  {KEY_CAPS_LOCK,KEY_TILDE,KEY_TAB,KEY_Z,KEY_A,KEY_1,KEY_Q,KEY_ESC},
  {KEY_F8,KEY_F5,KEY_F6,KEY_X,KEY_S,KEY_2,KEY_W,KEY_F7},
  {0,0,0,0,0,0,0,0},
  {0,0,0,0,0,0,0,0}
};
```

Download

https://www.instructables.com/FZQ/2VZS/KHJ4PBOH/FZQ2VZSKHJ4PBOH.pdf

Download

https://www.instructables.com/FLM/PCV5/KLTG9BK0/FLMPCV5KLTG9BK0.pdf

Download

https://www.instructables.com/FPR/KE44/KLTG9BKB/FPRKE44KLTG9BKB.pdf

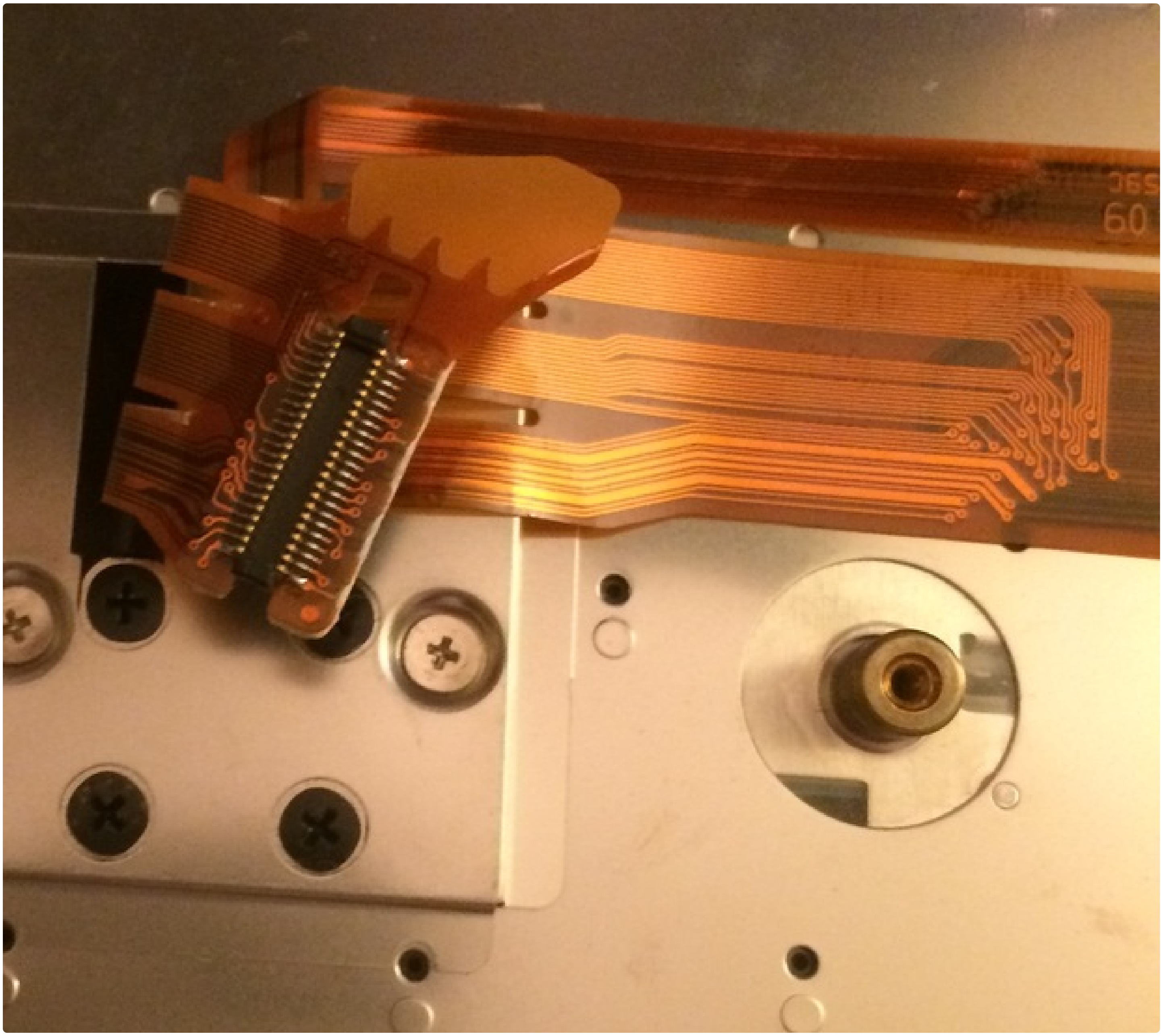
Download

https://www.instructables.com/FX6/CSQ5/KLTG9BKC/FX6CSQ5KLTG9BKC.pdf

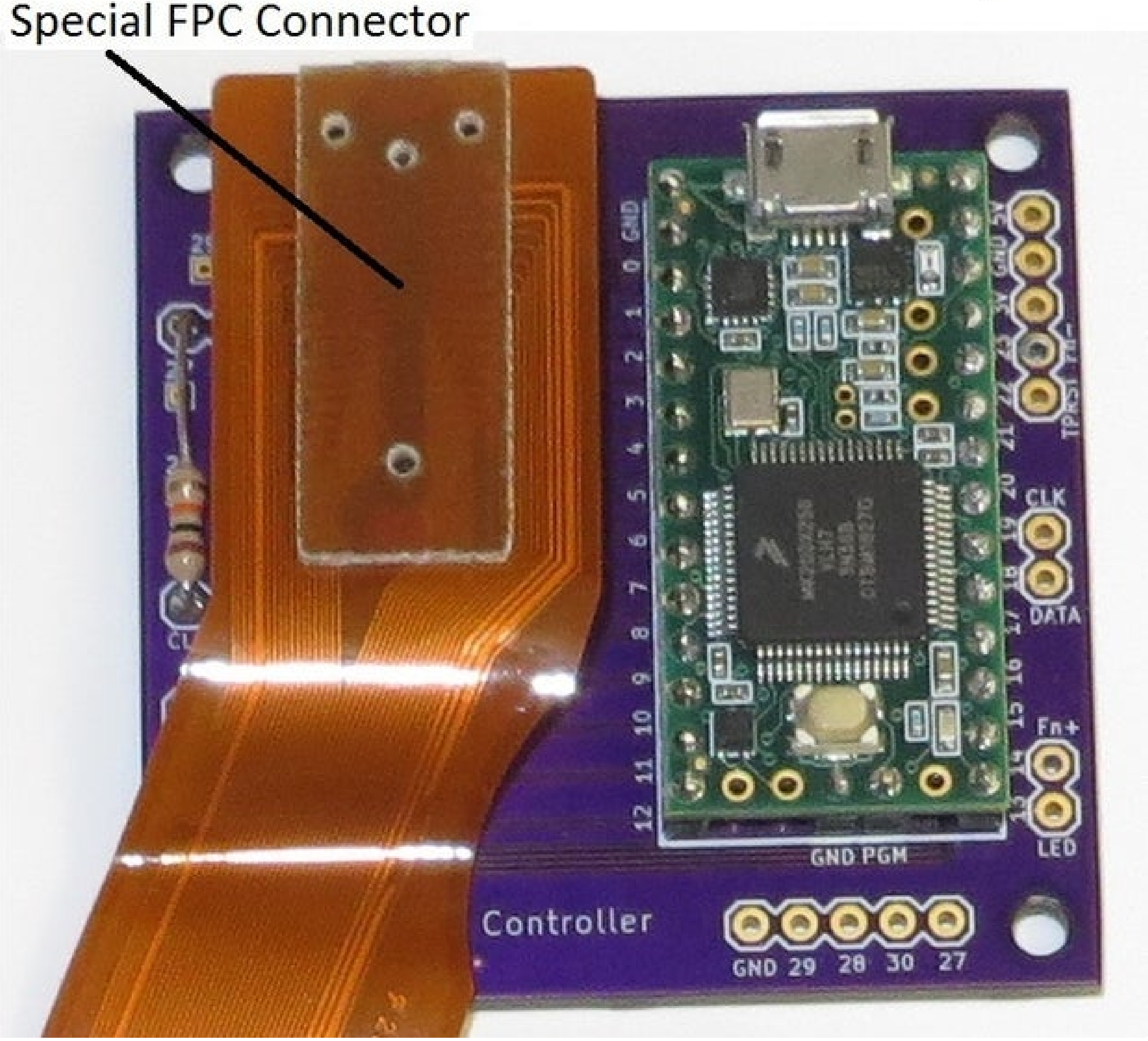
Step 16: Non-Standard FPC Cable Connectors

If your keyboard has a non-standard FPC cable like the Thinkpad and iBook connectors shown above, the task becomes more challenging. If you can't find a mating connector at Aliexpress or any other site, your only alternative is to remove the connector on the laptop motherboard. A common method is to put flux and low melt solder such as ChipQuik SMD removal alloy on all the joints, then use a soldering iron or hot air rework station. You will need to make a board layout that routes the Teensy I/O signals to your keyboard connector. I like to do a preliminary layout on paper first in order to place the parts and route the signals with the fewest via's. It's easy to assign the Teensy I/O pins in software based on whatever pin order makes the layout work best. It's tempting to make the layout next, but do the schematic first so your layout will have air-wires showing you how to route each trace. I didn't make a schematic for the keyboard scanner board because of the confusing front side LC/back side 3.2 routing. The downside of not having a schematic was the lack of any verification that the layout was electrically correct. I had to triple check everything before sending the file out for fab.

Eagle, KiCad, PCBWeb Designer, EasyEDA, and DesignSpark PCB are some of the free layout tools that are available. Sparkfun has excellent tutorials for Eagle schematic and layout. Also look at the Adafruit tutorial on creating parts in Eagle because you'll need to make a package and symbol for your connector. After you get your layout fabricated, you'll need to change the Matrix_Decoder software to work with the new I/O pin-out.



IBM Thinkpad T43 with Teensy 3.2



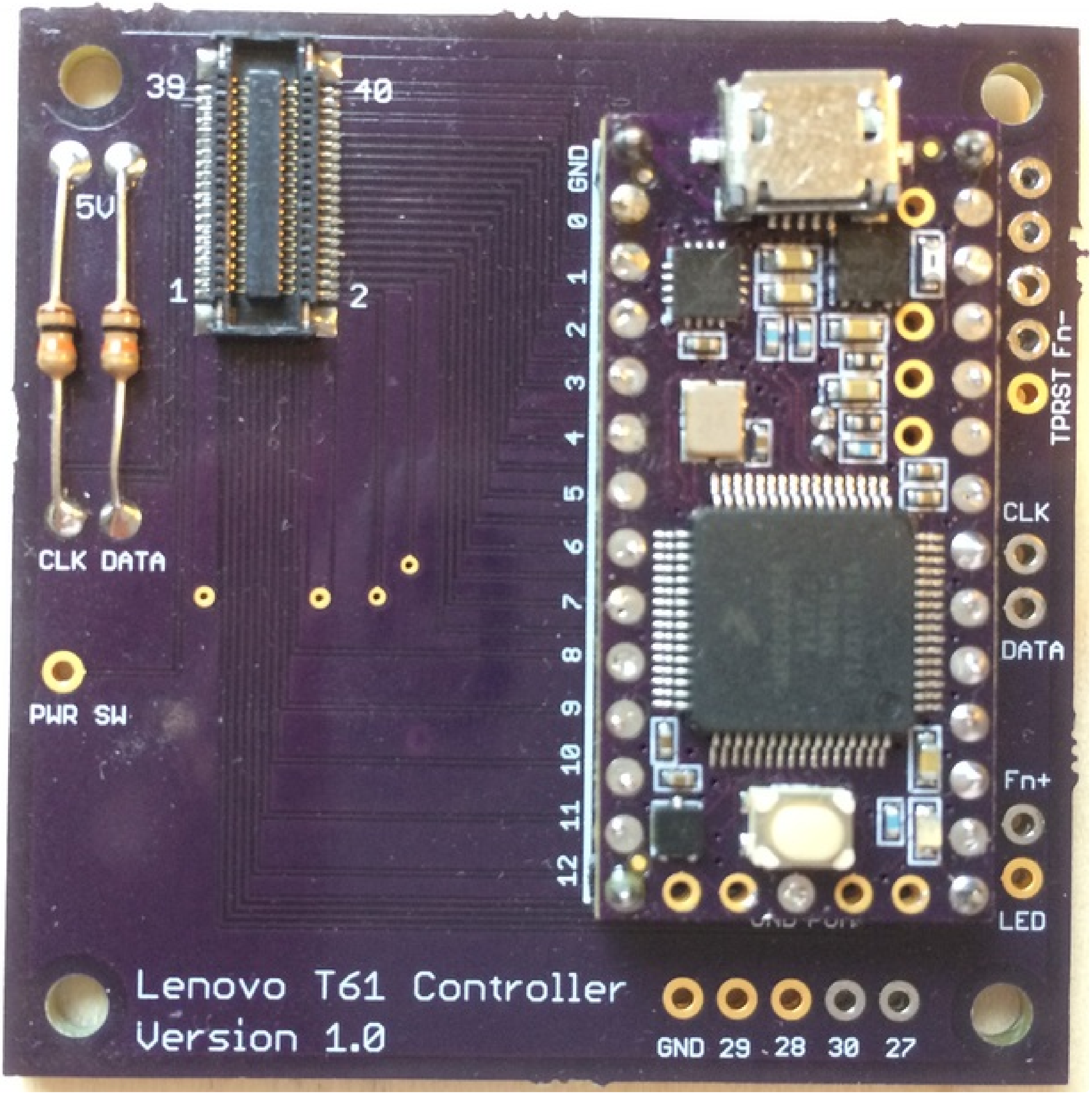
Special FPC Connector

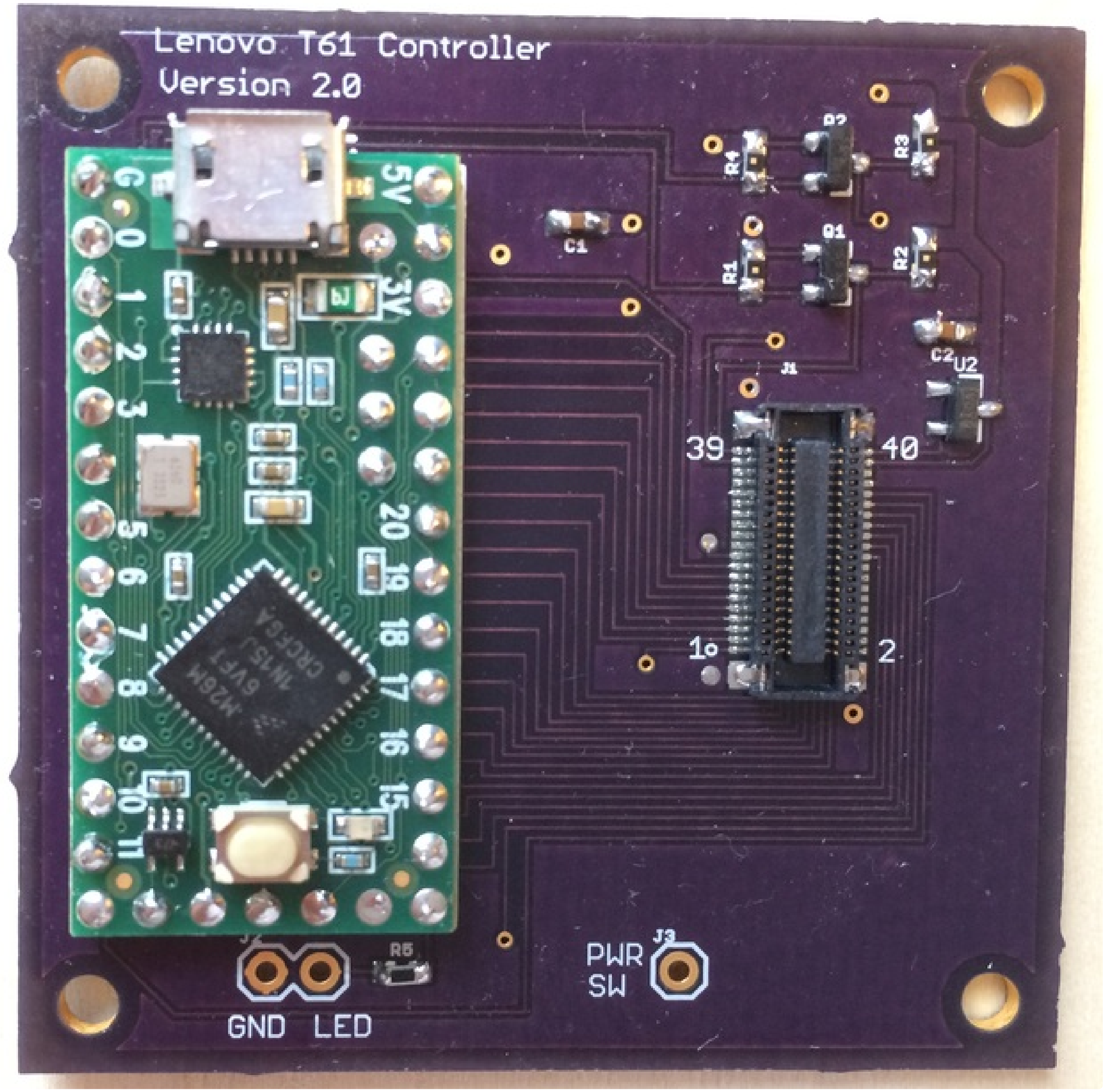
Step 17: Teensy 3.2 and LC Controller for a Lenovo Thinkpad T61 Keyboard

A perfect example of a non-standard FPC cable is the 44 pin connector used on the Lenovo Thinkpad T61 keyboard shown above. There are at least three web sites that detail how to make a USB controller for this Lenovo keyboard. An Instructable from rampadc uses a connector board with some glue logic and wires to an Arduino. A later Instructable from rampadc uses a single board with an MSP430 microcontroller. Mark Furland from Tome uses a connector board with wires to an Arduino. Mark's web site states that a Digikey WM6787CT-ND connector will work with the keyboard FPC cable. It lacks the plastic guides at the ends that guarantee the pins are in alignment so it takes a few insertions to get it positioned correctly. The Digikey connector saved me from having to unsolder the motherboard connector. It was pretty easy to search online and find a schematic for this laptop due to its popularity. Without the schematic or the info from rampadc, I would have been doing a lot of probing with an ohm meter to determine the ground pins and narrow down which pins needed to be scanned for the key matrix. I really like the feel of this keyboard which made it worth the effort to design the Teensy 3.2 circuit board shown above. I modified the Matrix_Decoder scanning software to only scan the 8 input pins and 16 output pins from the matrix. The scanning software produced a connection list that was turned into a key matrix table using the same steps described earlier in this Instructable. The Fn switch has its own two pins on the connector that are scanned separately from the key matrix. The Trackpoint on the keyboard needs PS/2 clock and data signals from the Teensy plus a reset signal when power is applied. The Teensy 3.2 is 5 volt tolerant so it can directly drive these signals. All of the T61 3.2 files and documentation are at my repo including a zip'ed gerber file for fabrication at JLCPCP. Nathaniel has modified my Teensy 3.2 board design for an IBM T43 keyboard which has a larger 40 pin connector. All of his design files can be found in a folder at my Github site. Pictures of his board are at the end of this Instructable in the "I made it" section.

I wanted to build a standalone T61 Keyboard on a wood base but the 3.2 circuit board needed the connector and Teensy re-positioned so the circuit board would be hidden underneath the keyboard. I figured while I'm at it, I should change over to the LC and save some money. The Teensy LC has fewer I/O signals and they are not 5 volt tolerant so I needed to make some design changes. I added a TLV810 to generate a reset for the trackpoint plus a couple BSS138 FETs as level translators for the trackpoint clock and data. To save an I/O pin, I wired the Fn switch into an empty cell in the matrix so it can be scanned with all the other keys. There was one Teensy I/O pin left to drive the Caps Lock LED. The finished LC circuit board is shown above. All of the T61 LC files and documentation can be downloaded from my repo including a zip'ed gerber file for fabrication at JLCPCP. For an even smaller Teensy LC T61 controller board, check out Martin Refseth's Github repository. He has taken my LC board design and shrunk it down to be the same size as the Teensy. A picture of his board is at the end of this Instructable in the "I made it" section.

Lenovo Thinkpad T61 Keyboard with integrated PS/2 Trackpoint

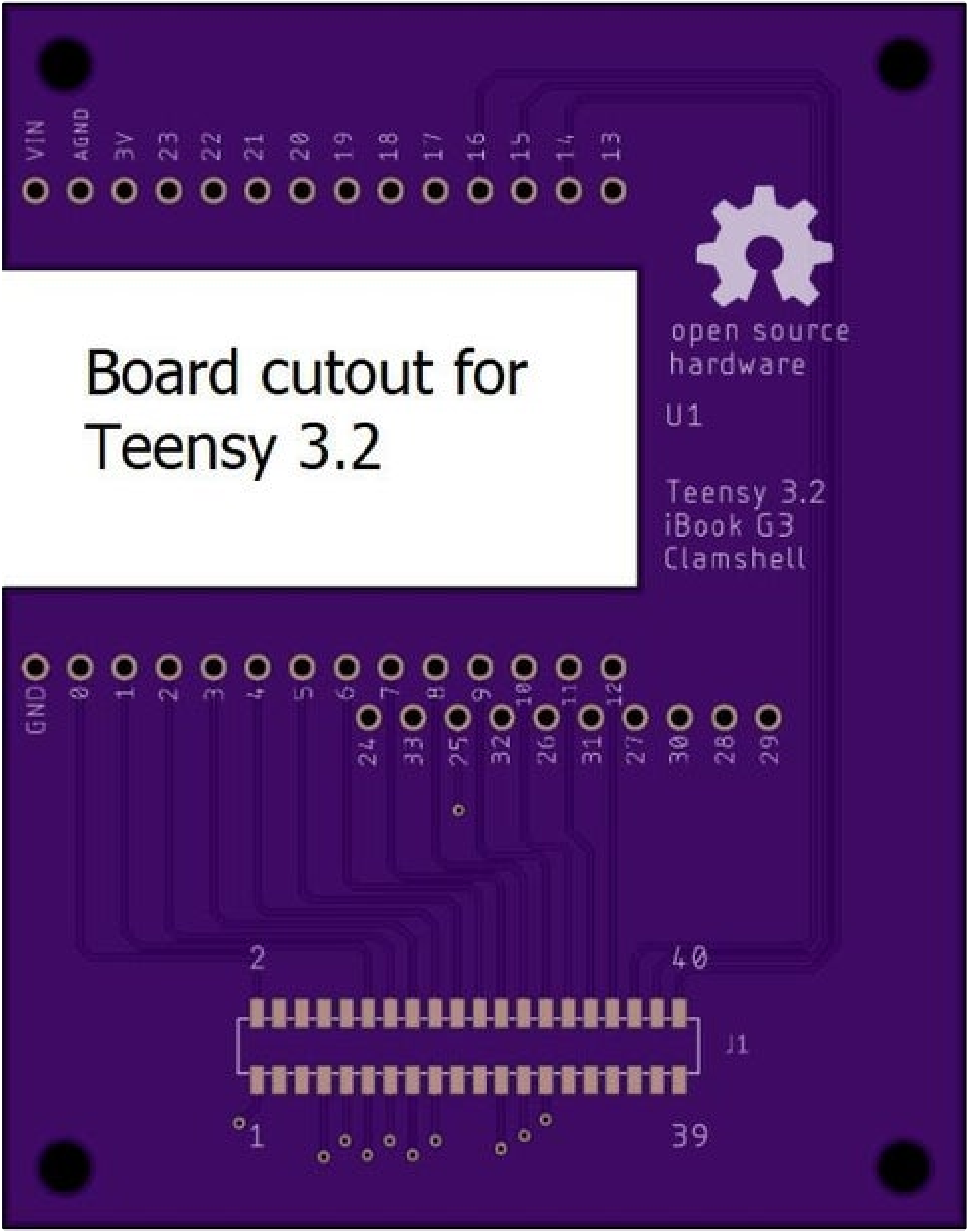

39
40
5V
1
2
CLK DATA
PWR SW
GND
0
1
2
3
4
5
6
7
8
9
10
11
12
TPRST Fn-
CLK
DATA
Fn+
LED
Lenovo T61 Controller
Version 1.0
GND 29 28 30 27

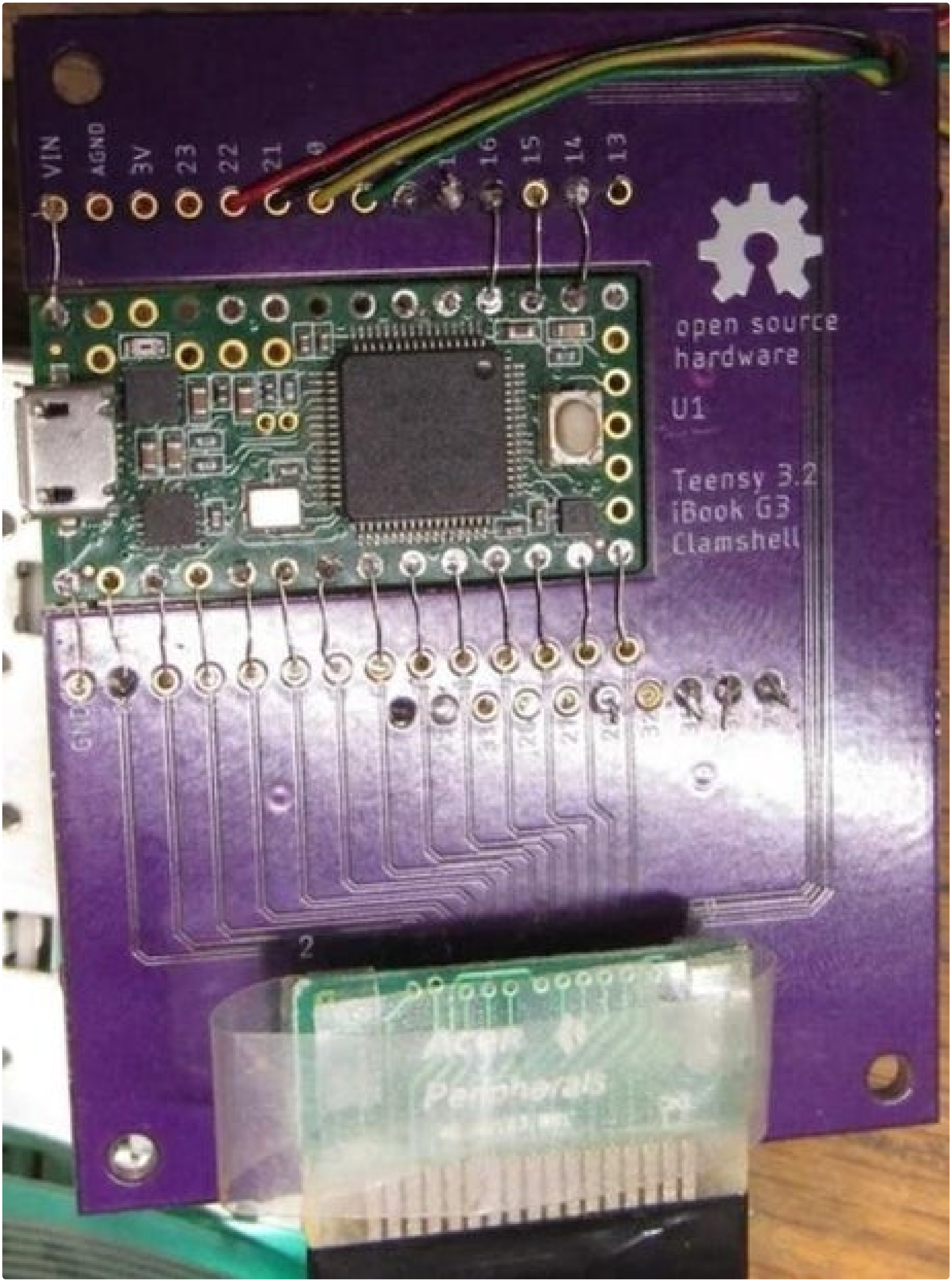


Step 18: Apple IBook G3 Clamshell

"Billy67" has used my code and circuit boards to convert several Apple keyboards to USB. Many are documented in the "I made it" section at the end of this Instructable. One of those keyboards is from an iBook G3 clamshell which has a special 40 pin connector on the end of its FPC cable. Because of height restrictions in the clamshell laptop, the Teensy must sit inside a cutout in the board. Wires or "U" shaped header pins tie the Teensy I/O's to the connector board. Billy did some experimenting and found part number HDR127MET40F-G-V-SM-DR mates up with this keyboard perfectly. To build a prototype, he etched his own circuit board with fine pitch traces drawn freehand with an ink pen (wow, that's hard core). If you're like me and haven't used PCB etchant in a few decades, you're going to want a circuit board file to send to a fab house. For this reason, I created the Eagle layout file "iBookG3Clamshell.brd" for sending to OSH Park and a zip'ed Gerber file for fabrication at JLCPCB. These files are available at my repo. Billy fab'ed the board at OSH Park and assembled it with a Teensy 3.2 and connector (see above). Billy has added power, ground, clock and data wires from the Teensy to a PS/2 touchpad (from a Dell). The keyboard and touchpad Teensy code for the iBook G3 is at my repo.

Apple iBook G3 Clamshell keyboard with 40 pin connector


VIN
AGND
3V
23
22
21
20
19
18
17
16
15
14
13
open source hardware
Board cutout for Teensy 3.2
U1
Teensy 3.2
iBook G3
Clamshell
GND
0
1
2
3
4
5
6
7
8
9
10
11
12
24
33
25
32
26
31
27
30
28
29
2
40
J1
1
39


VIN
AGND
3V
23
22
21
16
15
14
13
open source hardware
U1
Teensy 3.2
iBook G3
Clamshell
GND
2
Peripherals

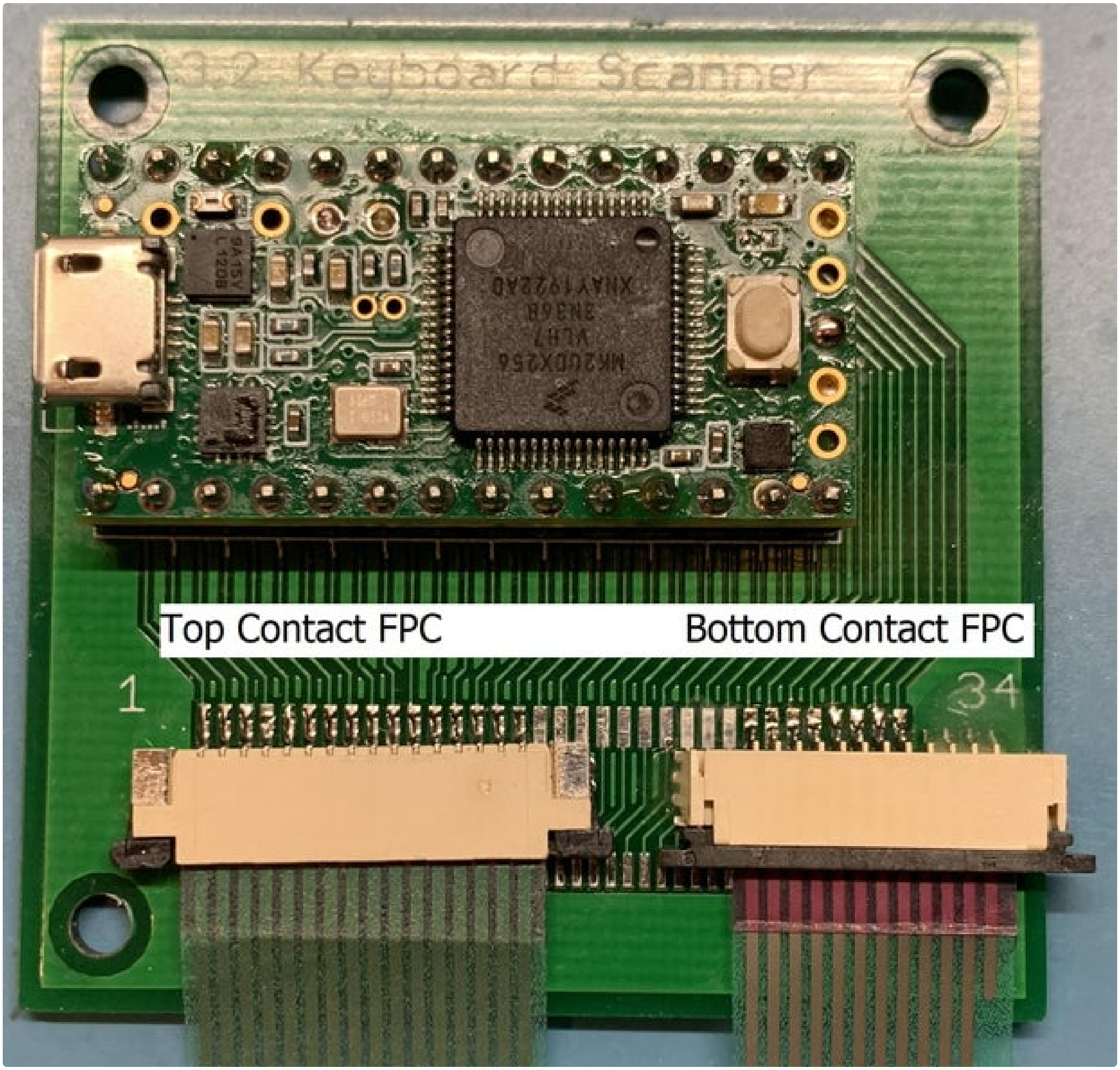
Step 19: Keyboards With Two FPC Cables

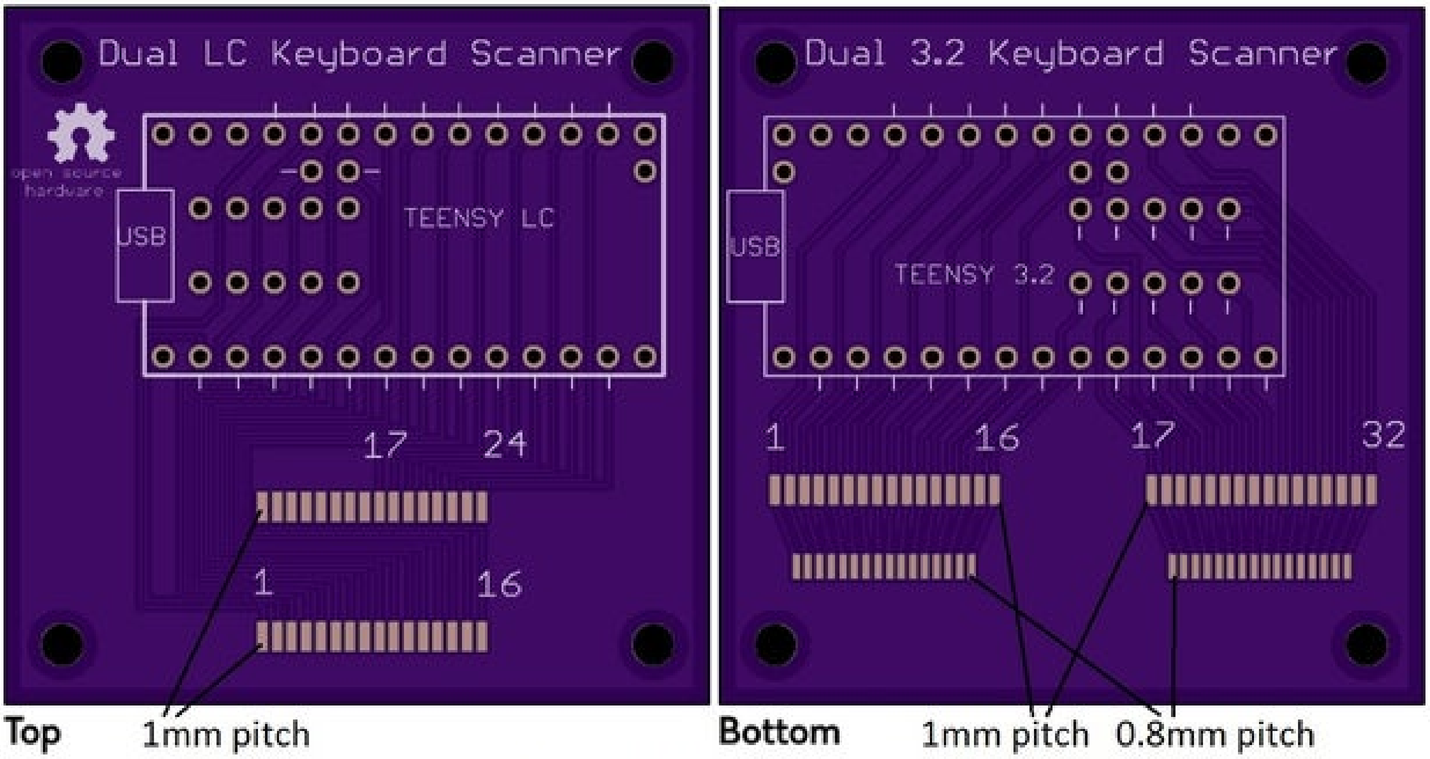
If your keyboard has two FPC cables, you may be able to insert them side by side in one FPC connector or solder two FPC connectors on the board as shown above. It's common for one FPC cable to have top contacts and the other to have bottom contacts. Use some JB-Weld epoxy on the connector legs and unused pins to help secure it to the board. Some other two cable options are:

The Keyboard_Scanner_Dual board shown above uses a Teensy LC on one side for keyboards like the Panasonic Toughbook CF-48 which has 2 separate 1mm pitch FPC cables that are on top of each other (not side by side). The other side of this board is for a Teensy 3.2 with 1mm and 0.8mm pads for FPC connectors placed side by side. This Eagle file can be downloaded below or from my repo. To order the board from JLCPCB, use the gerber zip file at my repo.

If you don't mind soldering a lot of wires, you can use the FPC breakout board shown above for one of the keyboard FPC connectors and put the other keyboard FPC connector and a Teensy on the regular board. Wire the breakout board to the Teensy per the tables on Step 14 so you won't need to modify the software. The front side of the breakout board will accept an FPC connector with up to 18 pins and either a 1mm or 0.8mm pitch. The back side of the board is for a 0.5mm pitch FPC connector. This approach will allow you to place the breakout board in whatever location is needed by the keyboard cables. The FPC_18pin1mm.brd Eagle board file can be downloaded below or from my repo. It only costs $3.75 to order 3 boards from OSH Park. The gerber zip at my repo can be used to order from JLCPCB.

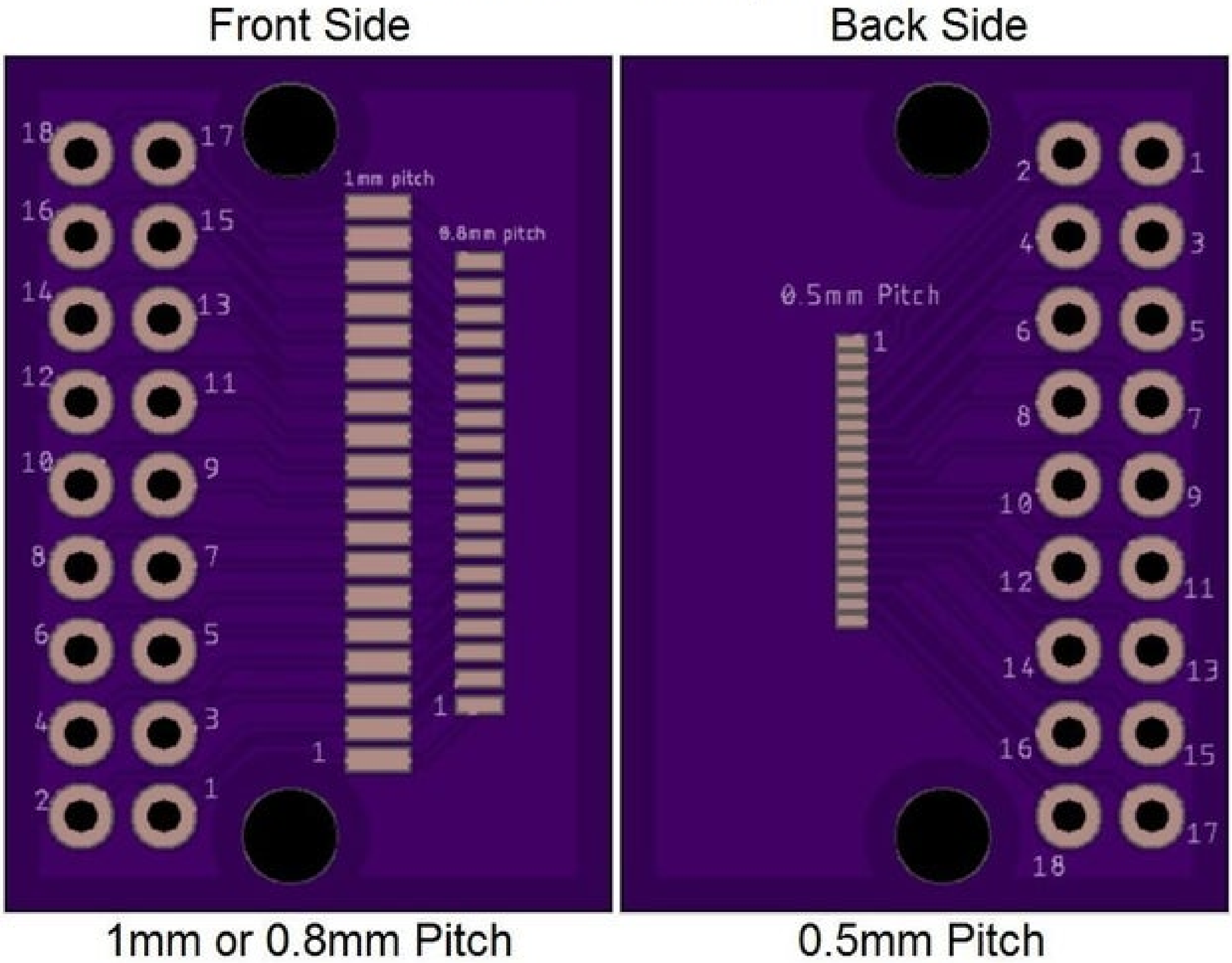
Your final option is to make a new layout like I did for the Lenovo 380 Keyboard shown above. A complete design description and Eagle board/schematic are at my repo. This could serve as a starting point for a new board layout and you can add other circuitry like a trackpoint interface or backlight connector.

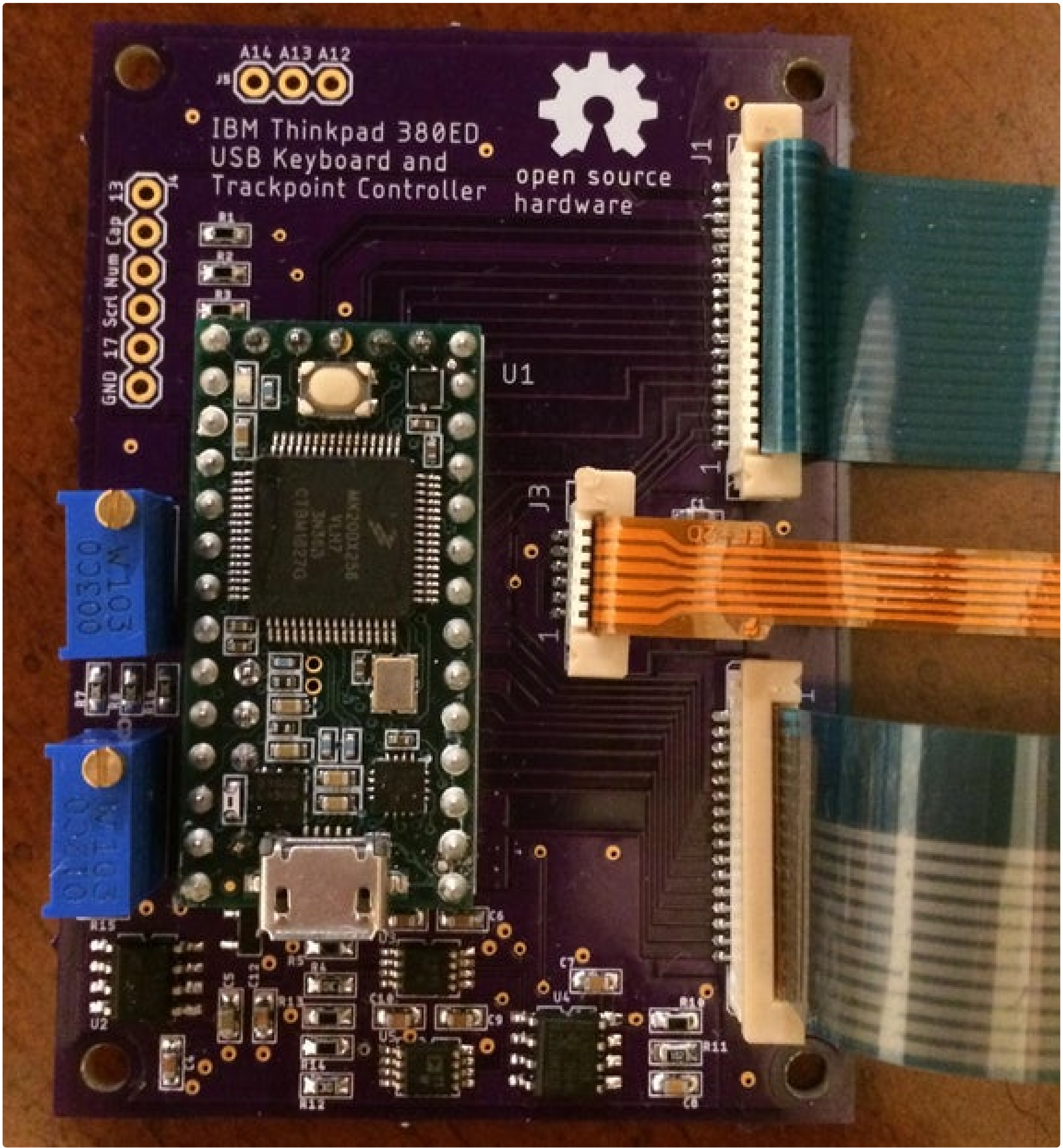

3.2 Keyboard Scanner
Top Contact FPC
Bottom Contact FPC
1
34



Keyboard_Scanner_Dual.brd

FPC Connector with up to 18 pins





Download

https://www.instructables.com/FAA/ZY74/K0R1M4OB/FAAZY74K0R1M4OB.brd

Download

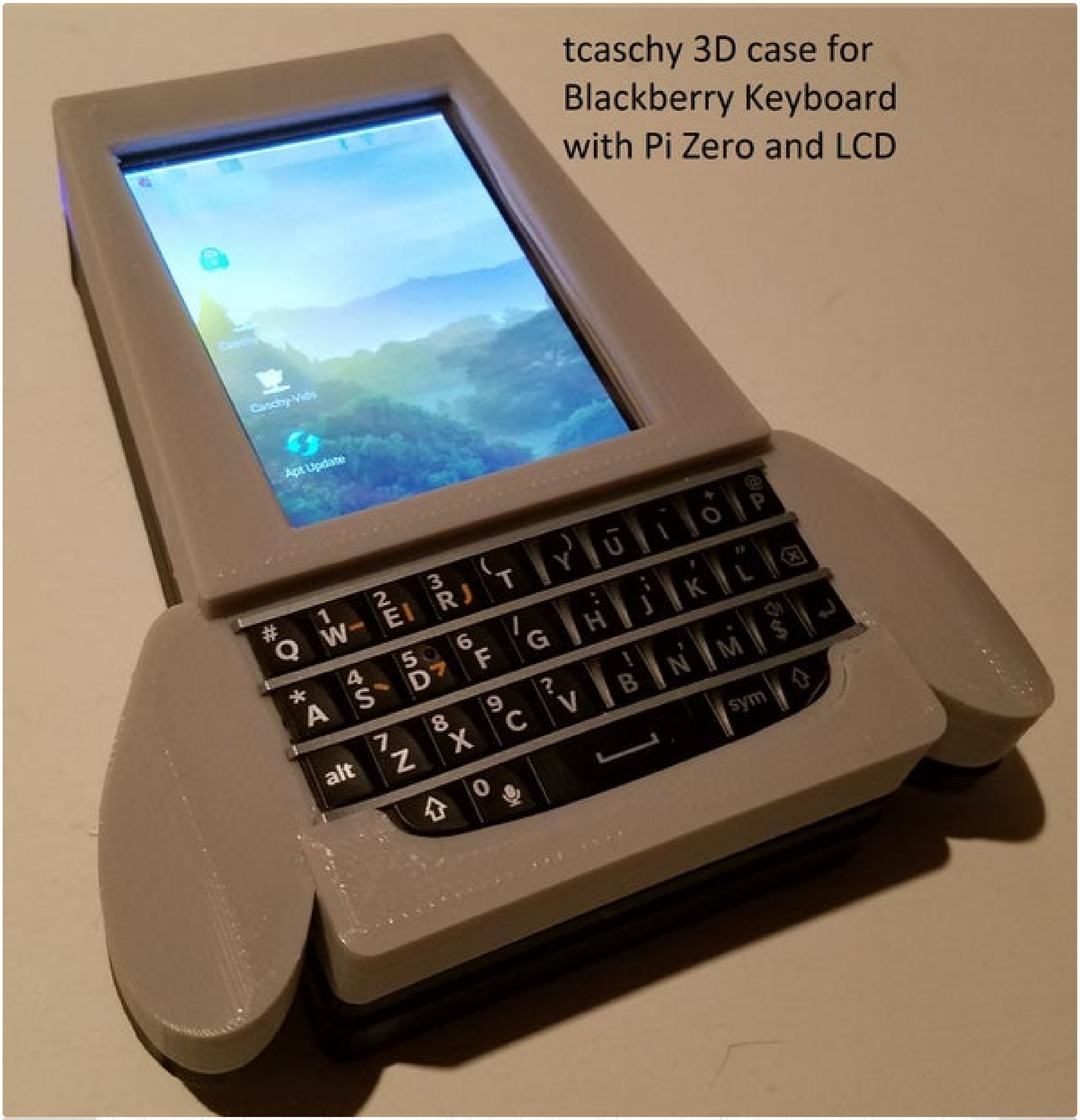
https://www.instructables.com/FF5/LG2W/KFL4659Q/FF5LG2WKFL4659Q.brd

Step 20: Building a Keyboard Base

The easiest way to mount your keyboard is to use the base from the original laptop, (see the Fujitsu case above). If you're not going to use the original laptop case, you can build a custom base out of various materials. Luigi Caradonna has made a stunning marble base using the CNC milling machine from his work. If you are lucky enough to have a 3D printer, you can make a custom case like TheGreenMamba's beautiful X61 Thinkpad keyboard. His 3D print files are at thingiverse . tcaschy has made a few 3D printed cases including the one shown above for a Blackberry keyboard with a Raspberry Pi Zero and LCD. Ben has the files for his T420 3D case at his repo. Harjoc used acrylic to make a cool see-thru base. The Thinkpad T43 keyboard shown above has been mounted on a wood base by "Than the FIT man". All of these projects are documented further in the "I made it" section at the end of this Instructable. I made a Thinkpad T61 keyboard base using 3 sheets of 1/4" plywood. Each sheet had its own cutouts to provide a cavity for the Teensy and wire routing. I've also built wood bases for the Toshiba 2415 keyboards that are documented at my repo. Warren has chiseled a wood base for a Dell Inspiron 1420 that is pictured on the PJRC forum with code at his Github repo. The Fujitsu Lifebook 755Tx keyboard has a thick backside so it can sit on a desk without any case. The picture above shows how I routed the dual FPC cables into a project box that houses the Teensy 3.2. The Fujitsu keyboard project documentation is at my repo. The Logitech K120 keyboard can be re-wired for control by a Teensy as shown in the picture above and documented at Hackster.

Black marquina marble keyboard base by Luigi Caradonna

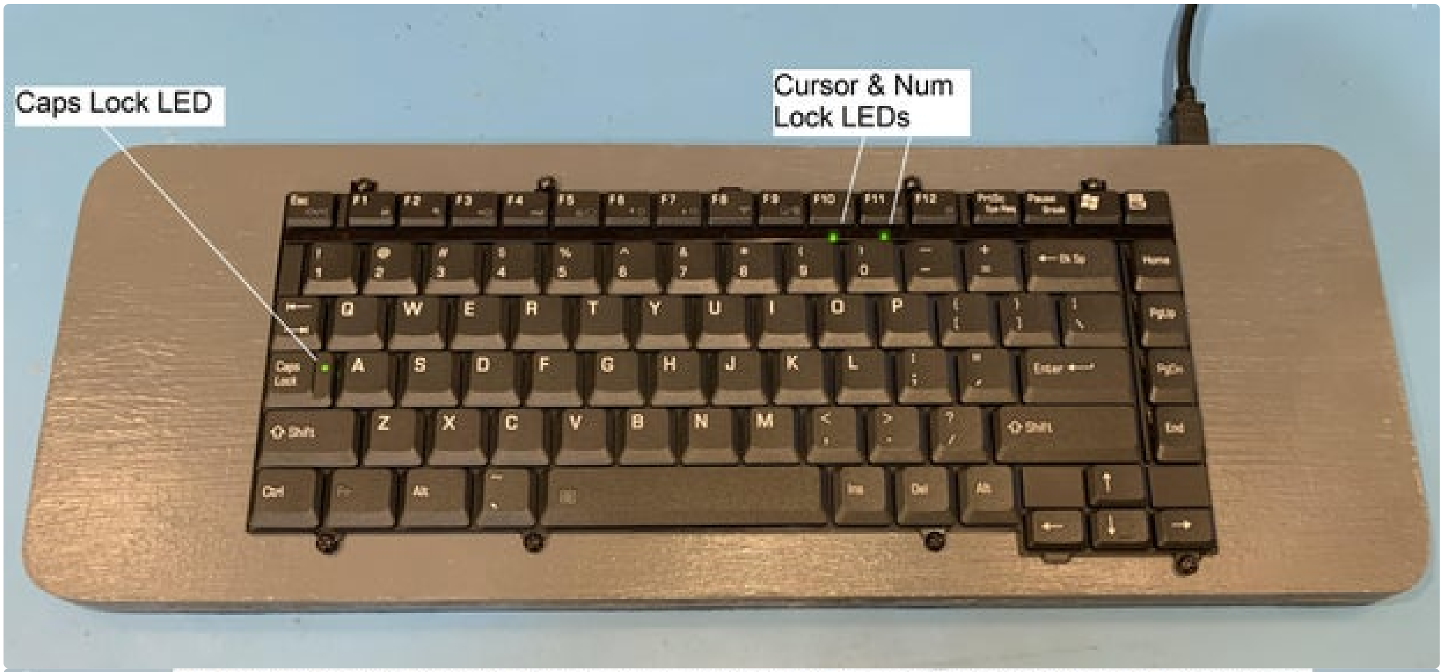
TheGreenMamba
Lenovo Thinkpad X61
3D Printed Case


tcaschy 3D case for Blackberry Keyboard with Pi Zero and LCD

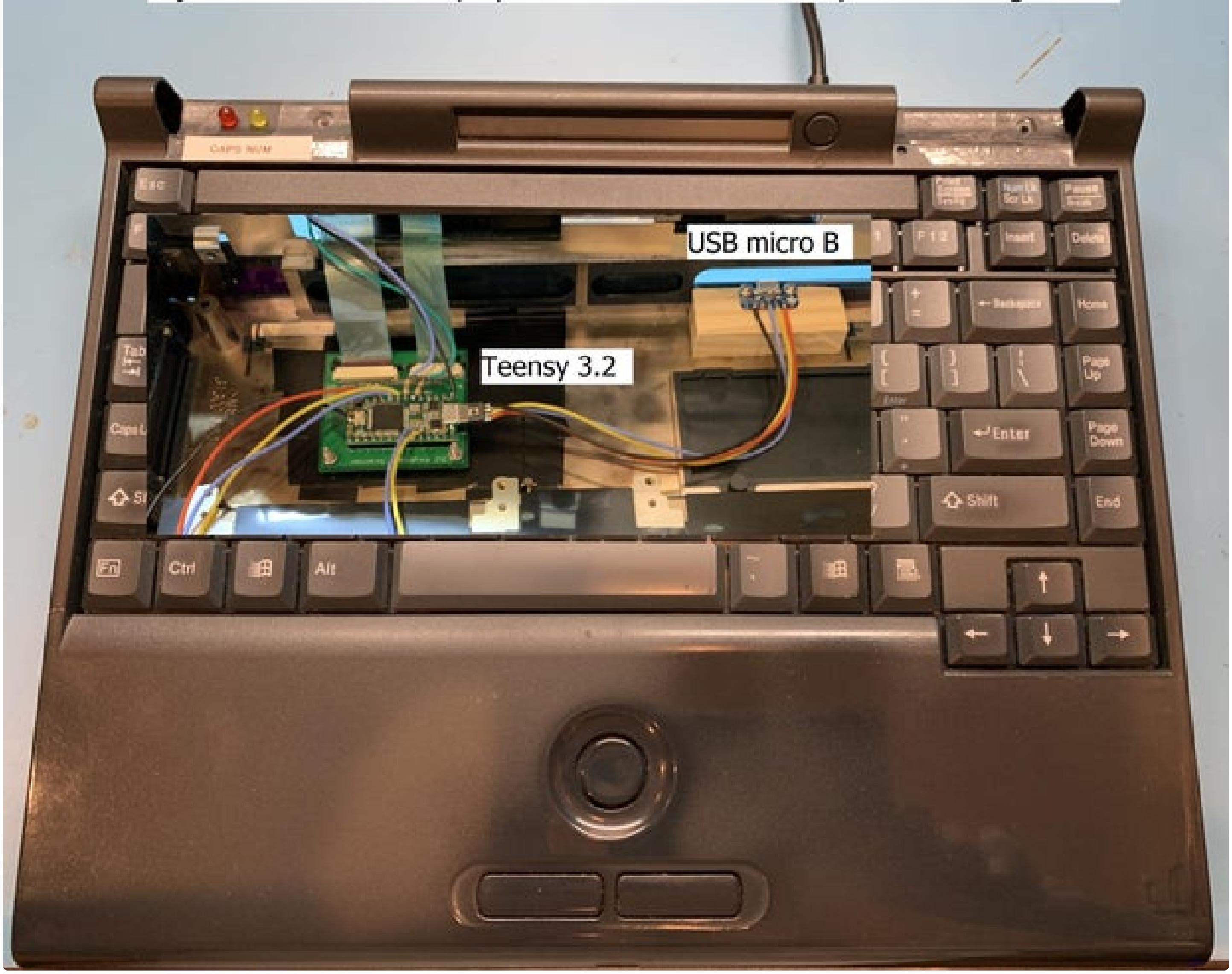
Ben's T420 USB/BLE portable keyboard with 3D printed case

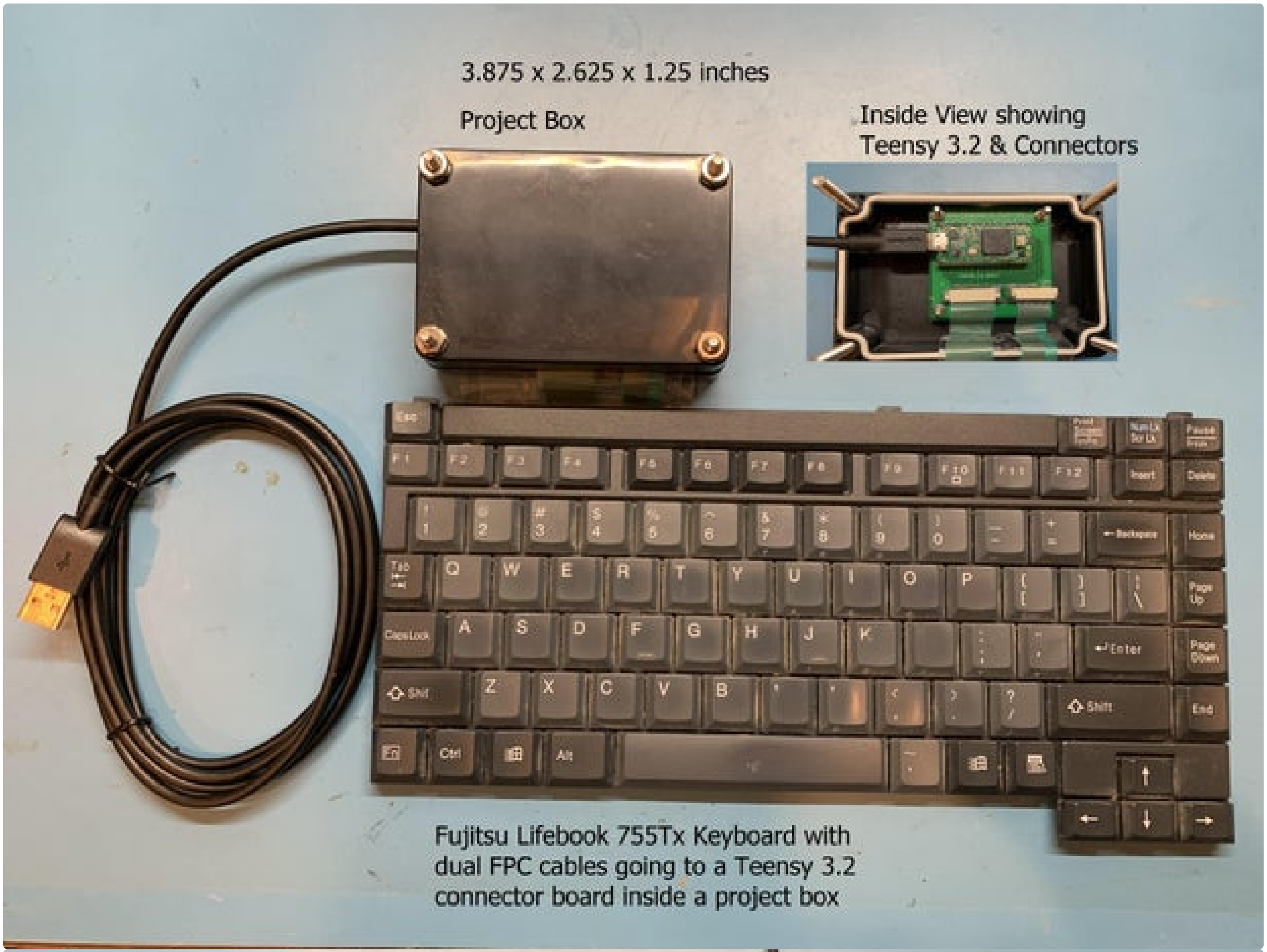
Harjoc Acrylic Base

Thinkpad T43 keyboard base by "Than the FIT man"



Fujitsu Lifebook 755Tx laptop base converted to USB keyboard and Ergo Trac



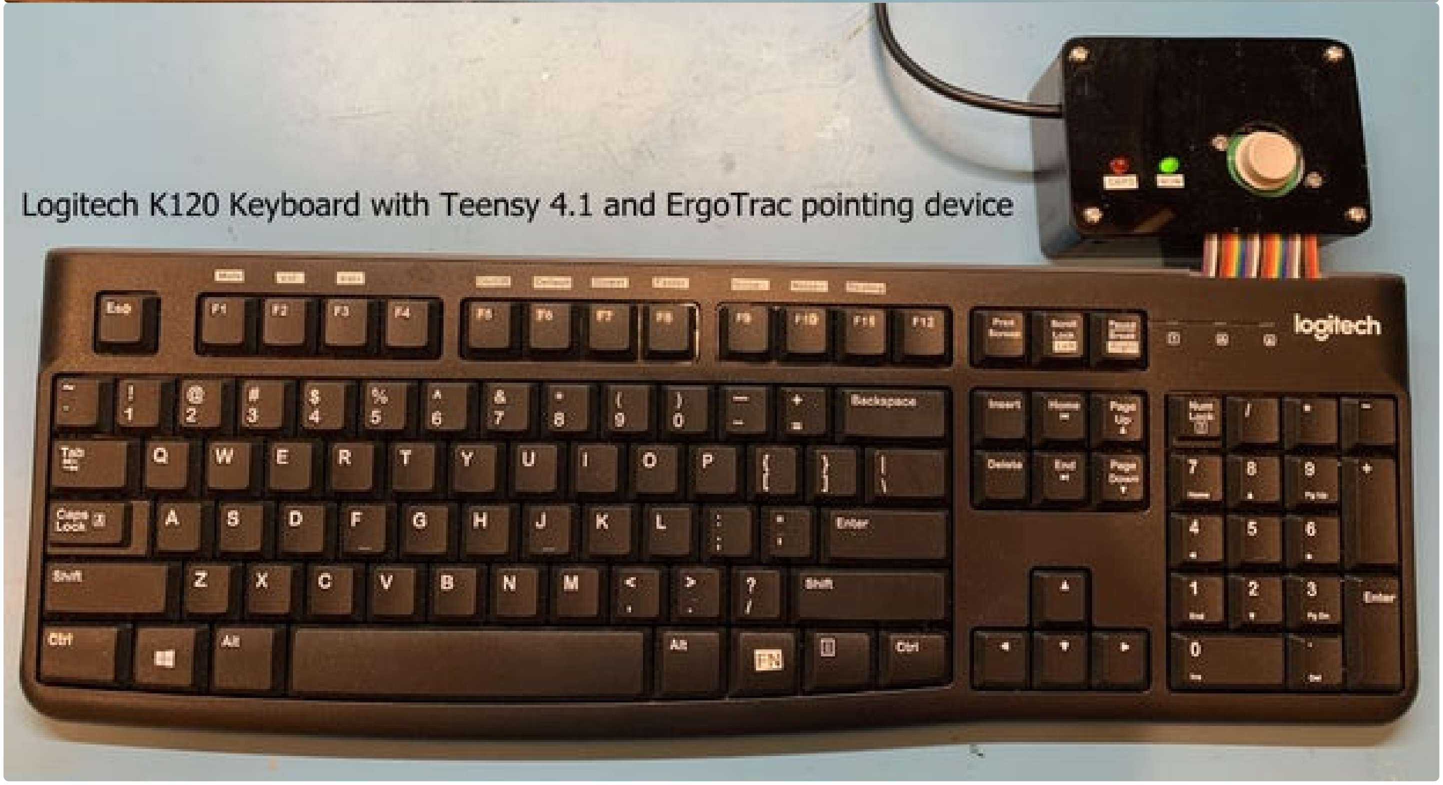


3.875 x 2.625 x 1.25 inches

Project Box

Inside View showing
Teensy 3.2 & Connectors

Fujitsu Lifebook 755Tx Keyboard with
dual FPC cables going to a Teensy 3.2
connector board inside a project box



Logitech K120 Keyboard with Teensy 4.1 and ErgoTrac pointing device

Step 21: Keyboard Backlight & LED Indicators

If your keyboard has a backlight, it can be controlled by the Teensy, as long as you have a spare I/O pin left over. The backlight brightness will be controlled using a pulse width modulated (PWM) signal that drives the gate of an N channel FET (see schematic above). The N-FET will handle the high current needed by the backlight LED. Only certain Teensy I/O pins are PWM capable so you may need to swap an I/O from the keyboard. Swapping I/O's is easy if you use wires instead of header pins to mount the Teensy to the board. The backlight signals from the keyboard usually come out on a small FPC cable that is separate from the row/column keyboard cable. You can glue a small connector on a blank board and solder wires to it but the fine pitch makes it difficult. I have designed two small connector breakout boards that make it easier to hook up the backlight cable. The Eagle board files can be downloaded below or from my repo and cost less than $3 to fabricate at OSH Park. One board will take an FPC connector with up to 10 pins and a 0.5mm pitch like this Molex connector. The other board will take an FPC connector with up to 8 pins and a 1mm pitch like this Molex connector. On the backside of each board are pads for a 1206 style current limit resistor and an MMBF170 N-FET. I've used a ½ watt, 22 ohm resistor but you should experiment to find a value that gives good brightness without excessive current. Set your ohm meter to diode mode so you can determine which pins are the anode and cathode. There are usually multiple pins for each in order to handle the current. Wire the anode pins to 5 volts and the cathode pins to the pad labeled "Drain" (which goes thru the 1206 resistor). The N-FET source is tied to the pad labeled "Ground" and should be jumpered to Teensy Ground. The pad labeled "Gate" should be jumpered to the Teensy I/O pin that will drive the PWM signal. Your code will need to set the PWM frequency. Don't make it too low or it will flicker. To set the PWM on pin 23 to 400Hz, use the following:

```
analogWriteFrequency(23,400);
```

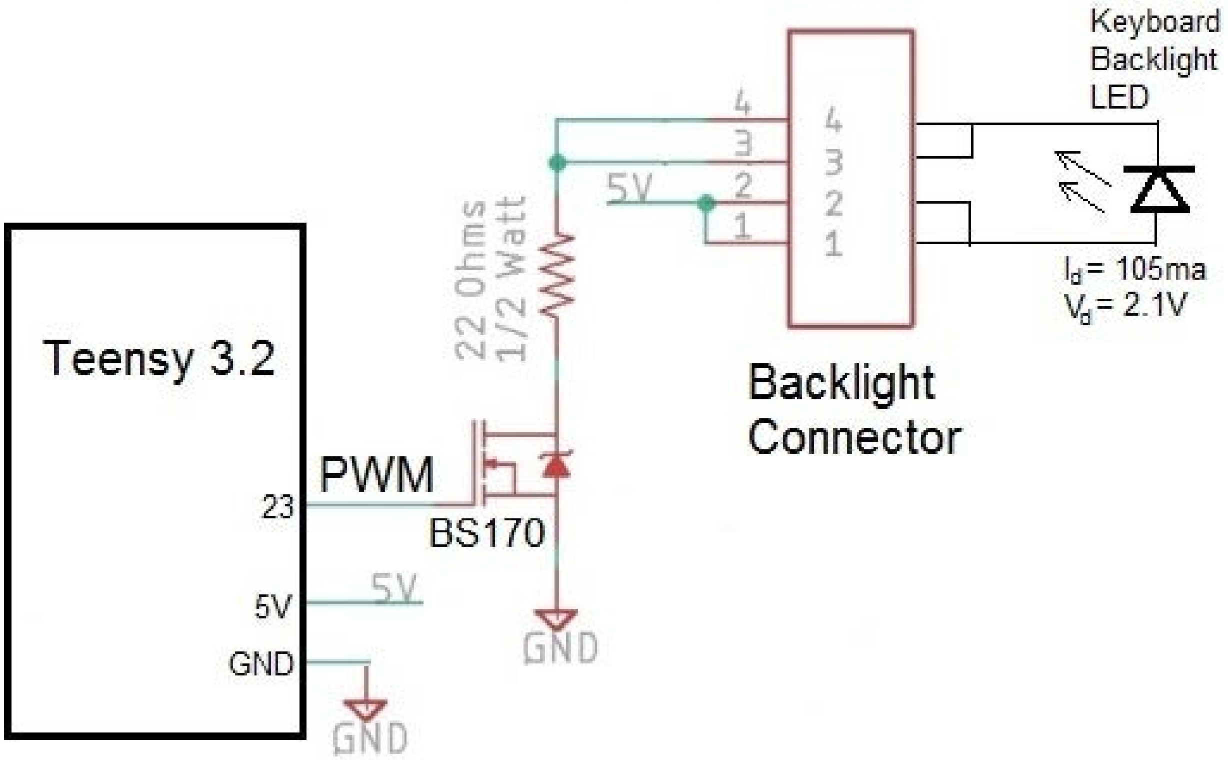
The PWM duty cycle can range from 0 (fully off), to 255 (fully on). To set the PWM on pin 23 to 80%, use the following:

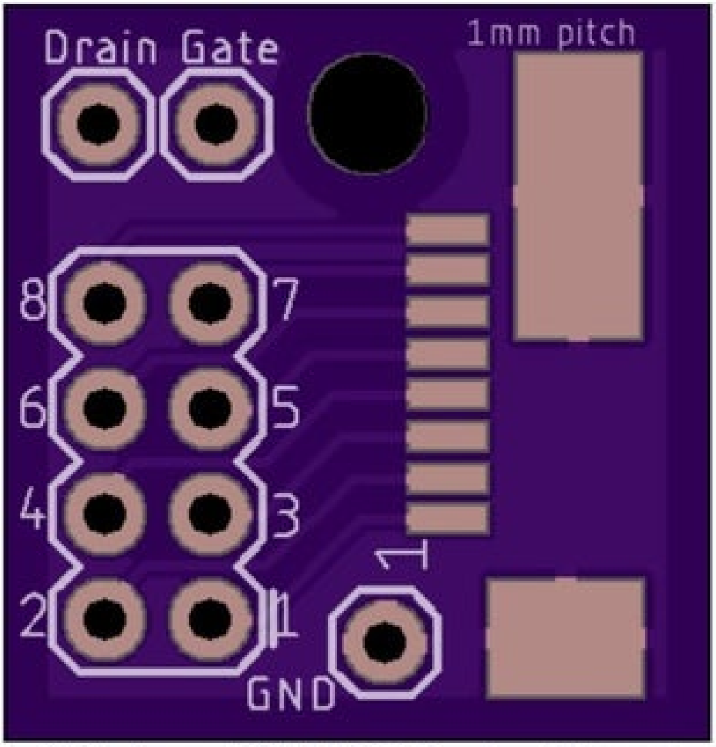
```
analogWrite(23,205); // 205/255 = 80%
```

Use a variable for the PWM duty cycle so it can be modified with an Fn-Function key press.

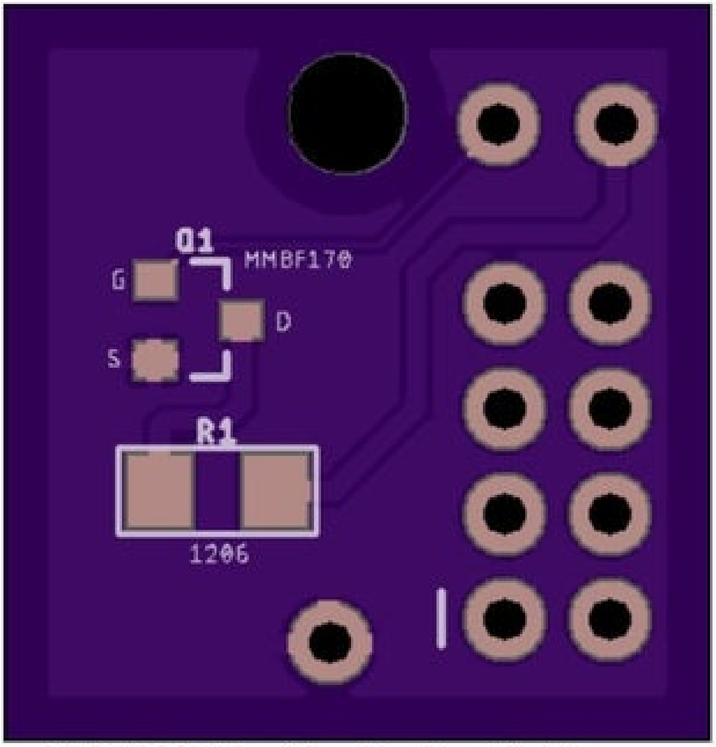
The CAPS, NUM, and Scroll Lock indicator LEDs can be controlled by the Teensy as shown in the schematic above. Look at the Dell D630 folder at my repo for an example. The keyboard matrix shows the 3 LEDs have a common anode and separate cathodes that go to pins on the FPC connector. For your keyboard, determine the keyboard matrix pins first, then the unused pins are the ones you'll probe with your multimeter to find the LEDs. Use the diode setting on your meter to provide enough current to light the LED when you have the correct pins and polarity. You will need a current limit resistor when the Teensy is controlling the LED. It may take some experimenting to determine the current level that gives good brightness. I've had good luck using 715 ohms on several keyboards which used 2ma LEDs. Depending on how the LEDs are wired, the Teeny will either drive the control signal low to turn on the LED or drive the signal high. Look at the bottom of many of the keyboard.ino files to see when the keyboard_leds variable is checked. This byte is read by the Teensy over USB from the host. The low 3 bits in the byte indicate when an LED should be turned on. Num Lock is at bit position 0, Caps Lock is at bit position 1, and Scroll Lock is at bit position 2. I often only care about the Caps lock LED (like on the Dell D630) so the other 2 LEDs are not in the code or wired to the keyboard, especially if I'm running out of Teensy I/O pins. Look at the Toshiba 2415 for an example of all 3 LEDs being controlled. Note it is possible to put a single resistor on the common LED pin instead of 3 separate resistors. The downside to this is the LED brightness will vary depending on how many LEDs are turned on.

Example Keyboard Backlight Control Circuit

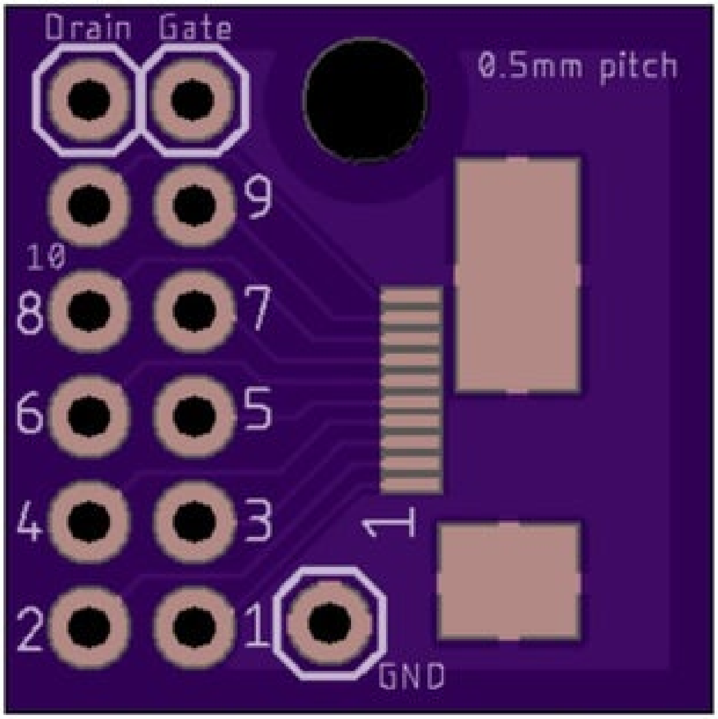




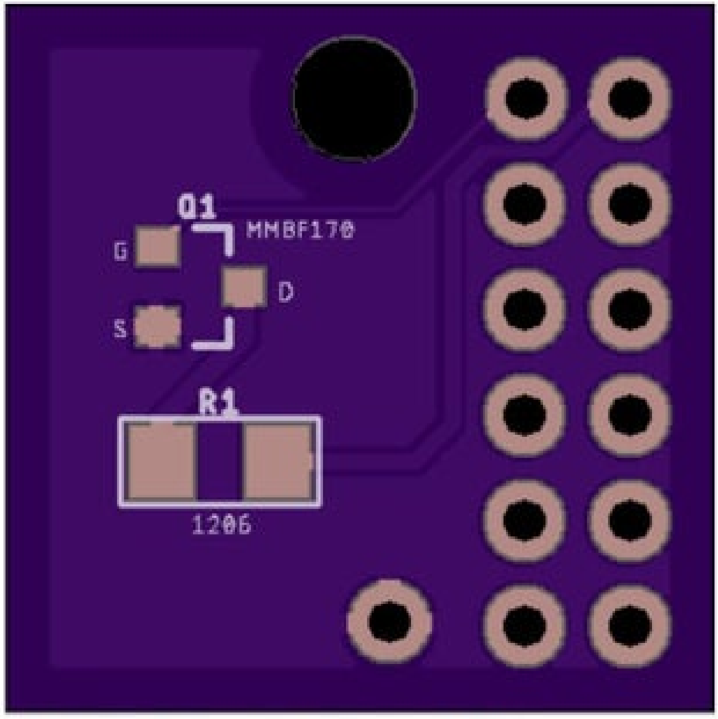
8 Pin 1mm Pitch FPC Breakout Board



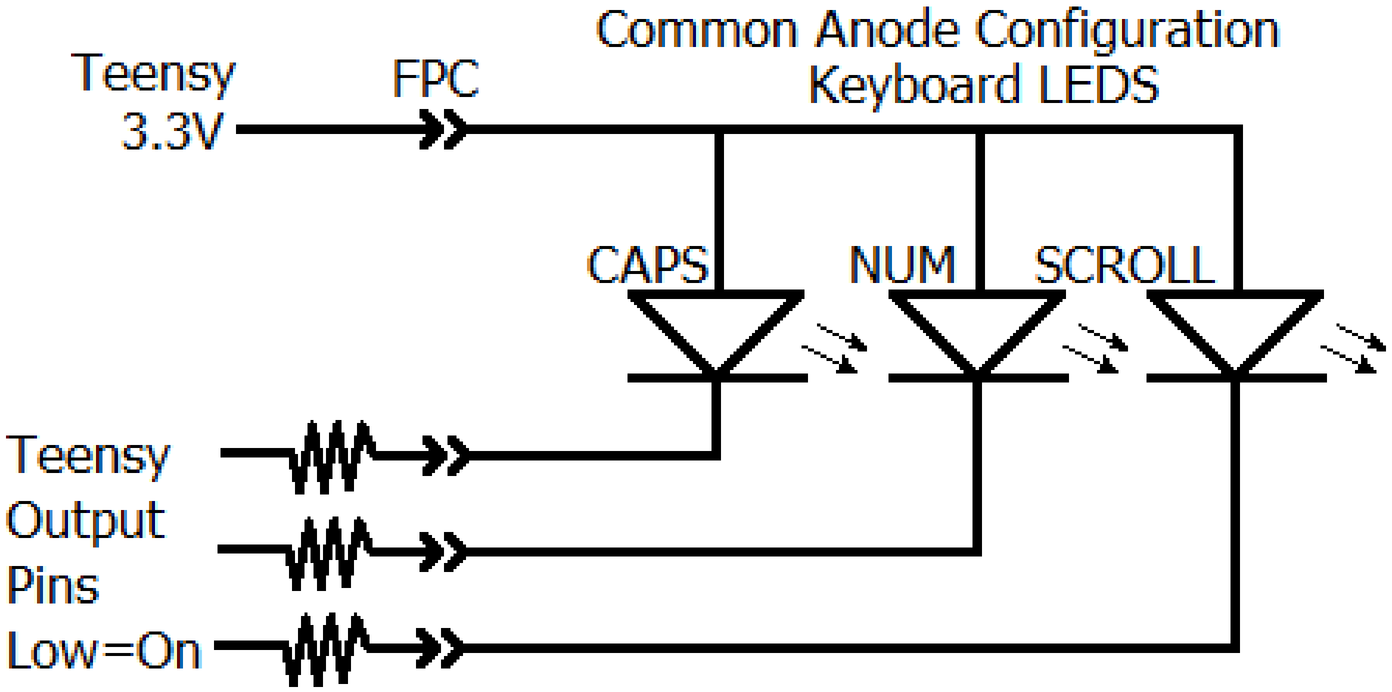
N-FET & Resistor for Backlight

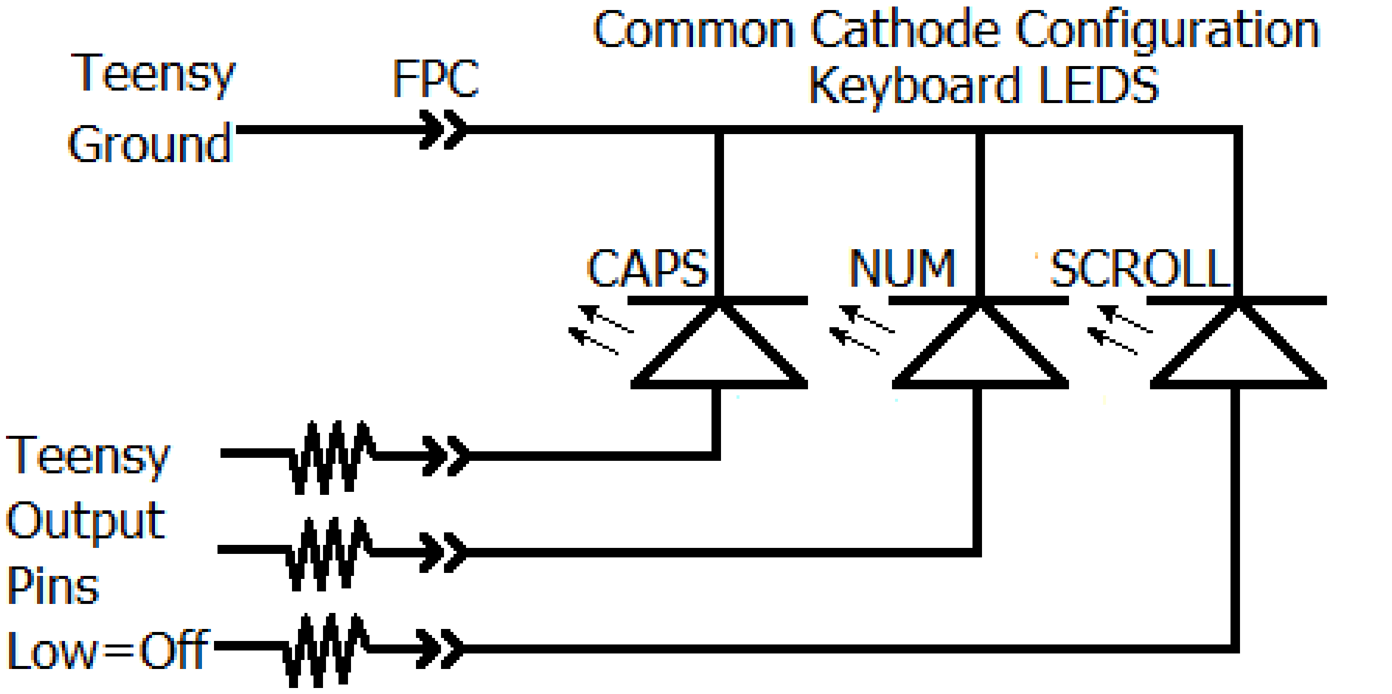


10 Pin 0.5mm Pitch FPC Breakout Board



N-FET & Resistor for Backlight





https://www.instructables.com/FUJ/ADQV/KFGTU13X/FUJADQVKFGTU13X.brd Download

https://www.instructables.com/F4E/4XPN/KFGTU13Y/F4E4XPNKFGTU13Y.brd Download

Step 22: Touchpad/Pointing Stick

The PS/2 signals from a laptop's pad/pointing stick can be converted to USB with an off-the-shelf converter as shown in numerous videos. If you also want to convert a laptop keyboard to USB, then it makes sense to combine these tasks with a Teensy and only use 1 USB port. PJRC has created mouse functions in Teensyduino so the USB coding is easy. I wrote code for the Teensy that bit-bangs two I/O pins to make the clock and data for the PS/2 bus. You can download my PS/2 code below or from my repo. In the Arduino IDE, under "Tools", set the code to the Teensy model you are using and to "Keyboard+Mouse+Joystick". Compile and load the code into the Teensy. The code sets the pad/pointing device to be polled at a regular rate to see if any movement or button presses have occurred. This makes it easy to merge in with the keyboard scanning code, which also repeats at a regular rate. The Thinkpad T61 shown earlier uses this method to merge the keyboard and PS/2 trackpoint code. The Dell Latitude D630 keyboard code below or at my repo has also been merged with the touchpad code. It can be difficult to figure out the clock, data, power, and ground pins if you don't have a schematic so I wrote a step by step procedure to identify the pins. This guide also describes how to use a level translator for the clock and data signals when using a Teensy LC or 4.0. The schematic for the translator is shown above. For a quick and dirty cable solution, the clock, data, power, and ground wires from the Teensy can be soldered directly to pads on the board. If the board has an FPC connector and cable, you can use one of the breakout boards from the previous step to provide access to the 4 signals that go to the Teensy. An example of a breakout board for a Toshiba 2415 touchpad is shown above. The composite keyboard and touchpad code for the Toshiba 2415 is at my repo.

For you "real" coders, the interrupt driven trackpoint routine at Martin Prochnow's Github repo is more efficient than polling. This code has been merged with my USB keyboard code (with many improvements) by Ben for his Thinkpad T420. His original Github repo has his initial code and circuit board but he just couldn't stop there. Now his second repo provides the code, circuit board files, and 3D printer files to make a keyboard that's USB and Bluetooth capable. It has rechargeable batteries in the case along with lots of weights to hold it in place.

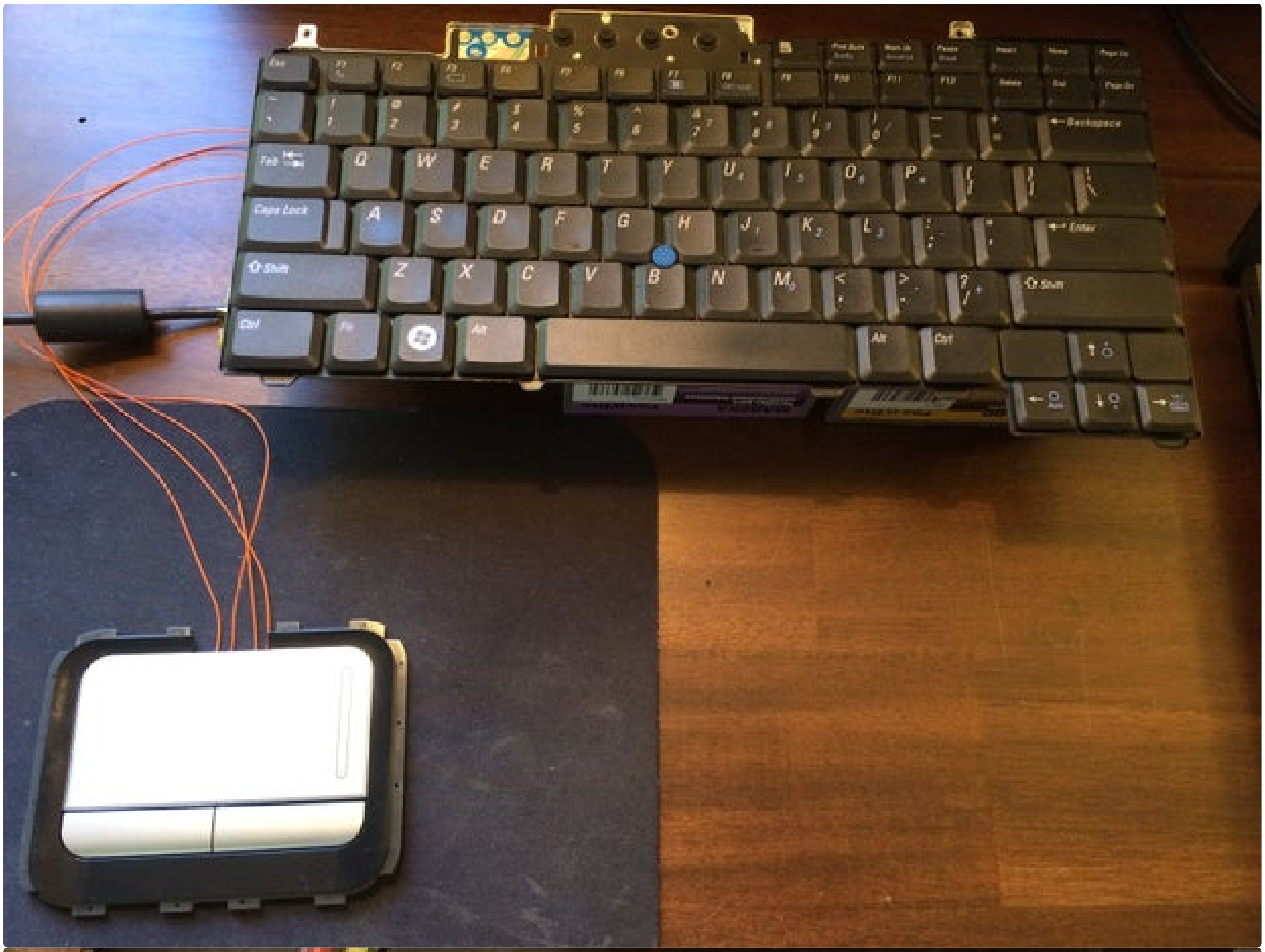
A trackball, touchpad, or mouse that uses the Apple Data Bus can be converted to USB using the Teensy code at my repo. This code was merged with the keyboard scanning circuitry for a Mackintosh Powerbook 140 shown in the "I made it" section at the end of this Instructable.

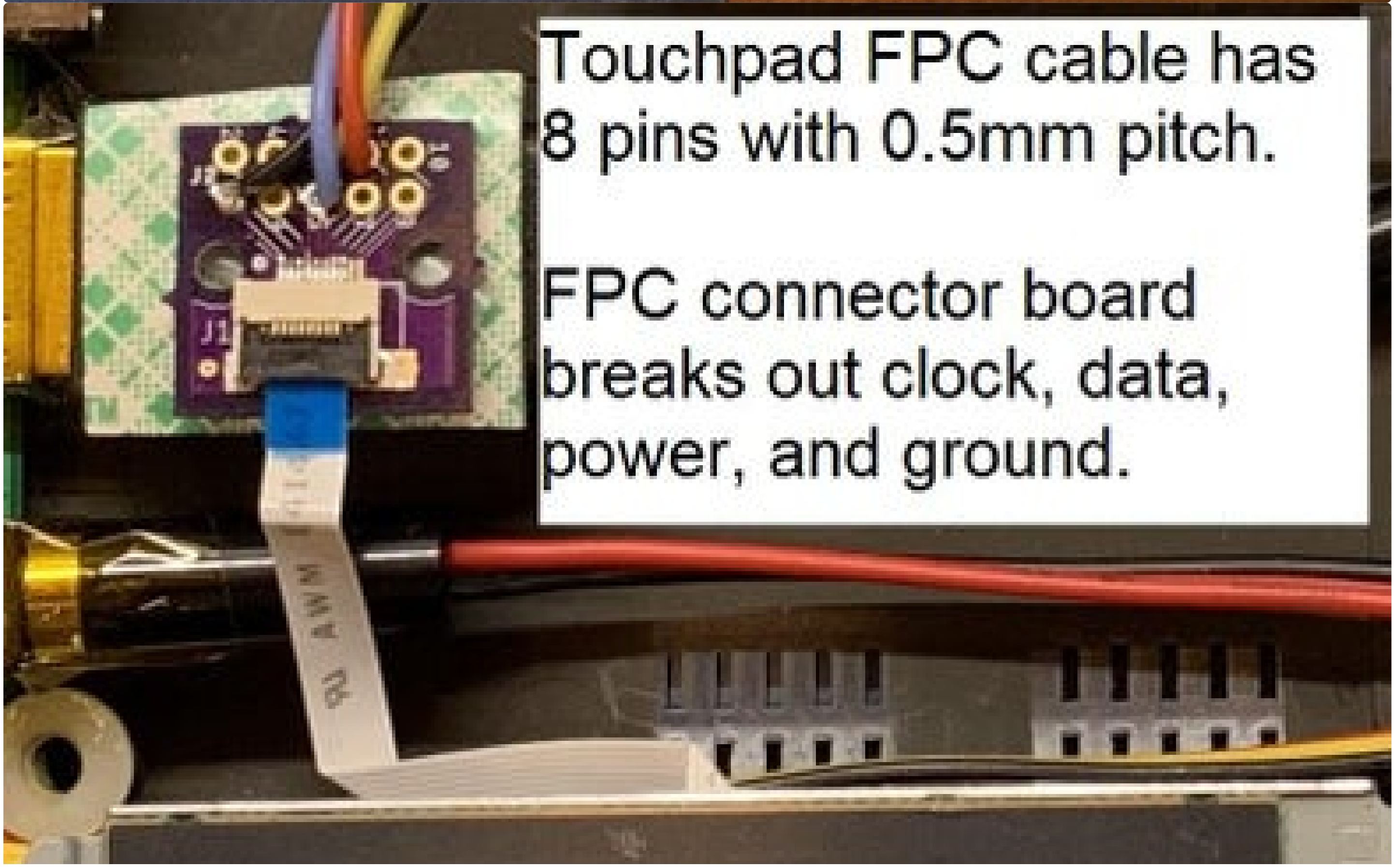
I2C touchpads are becoming more popular in modern laptops. This 2 wire clock and data bus can be directly driven by the I2C pins of a Teensy so no bit-banging is needed. My Hackster.IO touchpad tutorial describes how to determine the touchpad's I2C address and registers using a Raspberry Pi. Example I2C Teensy code for a basic touchpad and for a very complex Azoteq TPS65-201A-S touchpad can be downloaded from my repo and merged into a keyboard scanning routine. The Azoteq touchpad shown above has tons of features described in it's datasheet and it only costs $5.70 at Mouser.

Before pads, nubs, and trackballs became popular, the GRiD 1550 laptop of the 1980's used a metal rod below the space bar called an IsoPoint. The rod can be spun clockwise or counterclockwise for the Y cursor movement. This spinning rod is rotary incremental encoded with 2 signals. The rod will also slide left or right for the X cursor movement which also has two rotary incremental encoded signals. The Teensy code at my repo translates these 4 signals into USB. A complete description of this project is at my repo.

If your trackpoint/pointing stick does not output a serial bus, it may only provide the 4 signals from its strain gauge resistors. The Dell D630 pointing stick has four 4K ohm strain gauge resistors that change resistance enough that the ADC in the Teensy can see the voltage changes above the noise. The Dell D630 pointing stick code and PDF are at my repo. Old Thinkpad trackpoints like the one on the 380 can also be converted to USB but its strain gauge resistors need to be amplified as documented in my Hackaday.IO project.

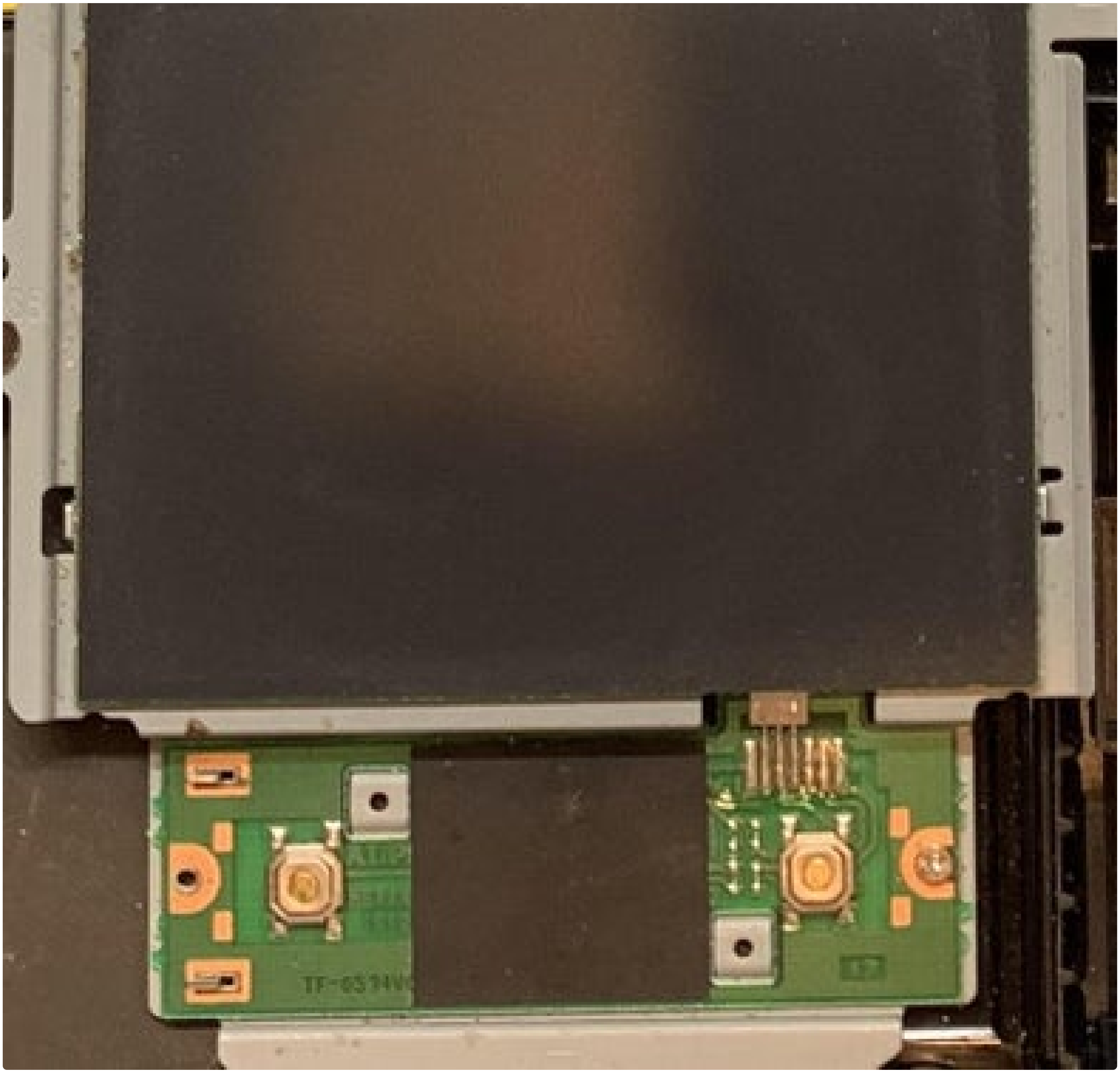
Some Fujitsu laptops have a pointing device called an Ergo Trac instead of a touchpad. The sensor part number is FID-828-100/20 and it uses magnetic field detection technology. The picture above shows the 4 connections that normally go to a controller chip on a CA20290-B40X daughterboard but it can also be driven by a Teensy 3.2. The Teensy code at my repo drives a reference logic pulse to the sensor and reads the resulting pulse voltage with two ADC channels. The code then converts the digital values to USB mouse movements.

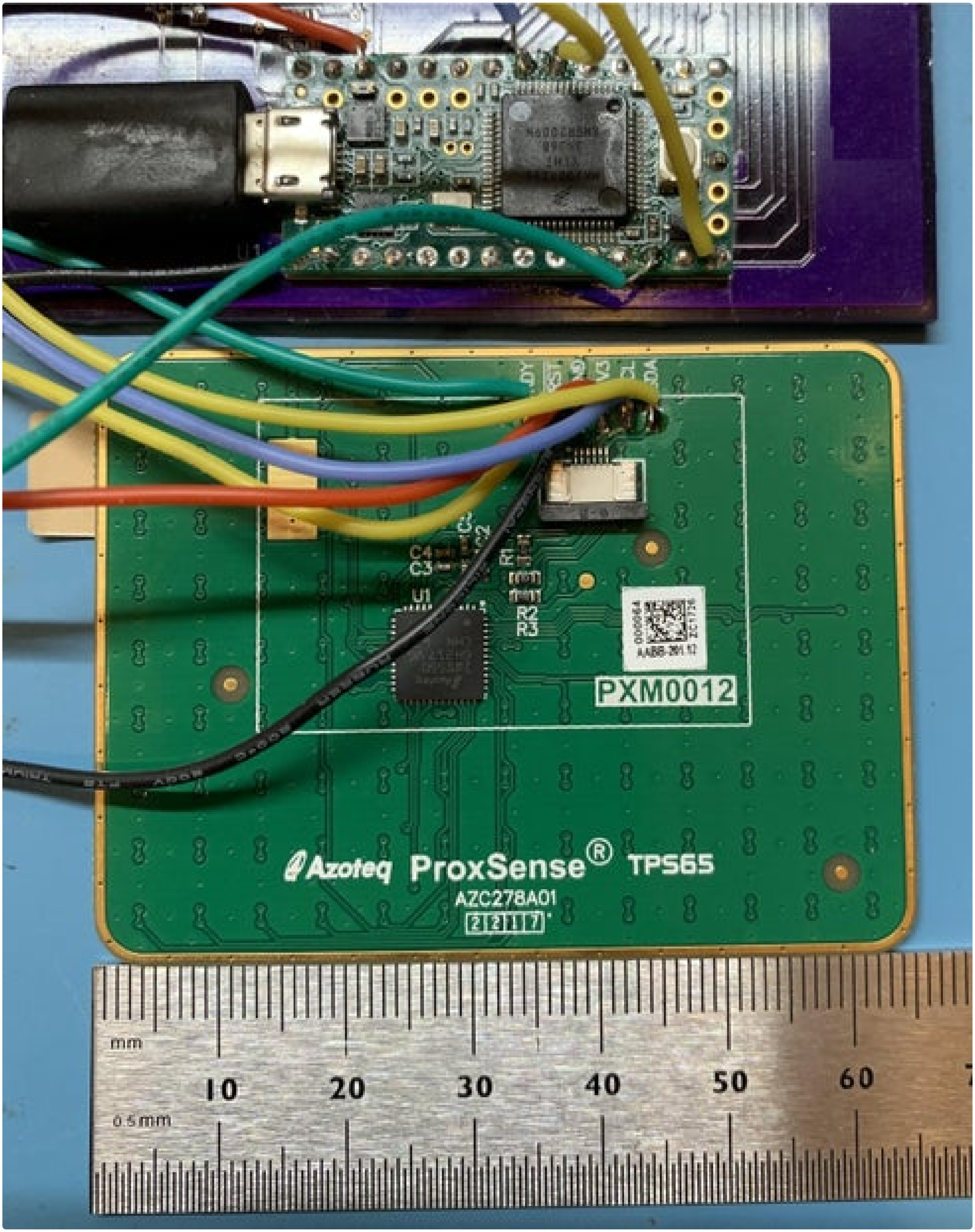


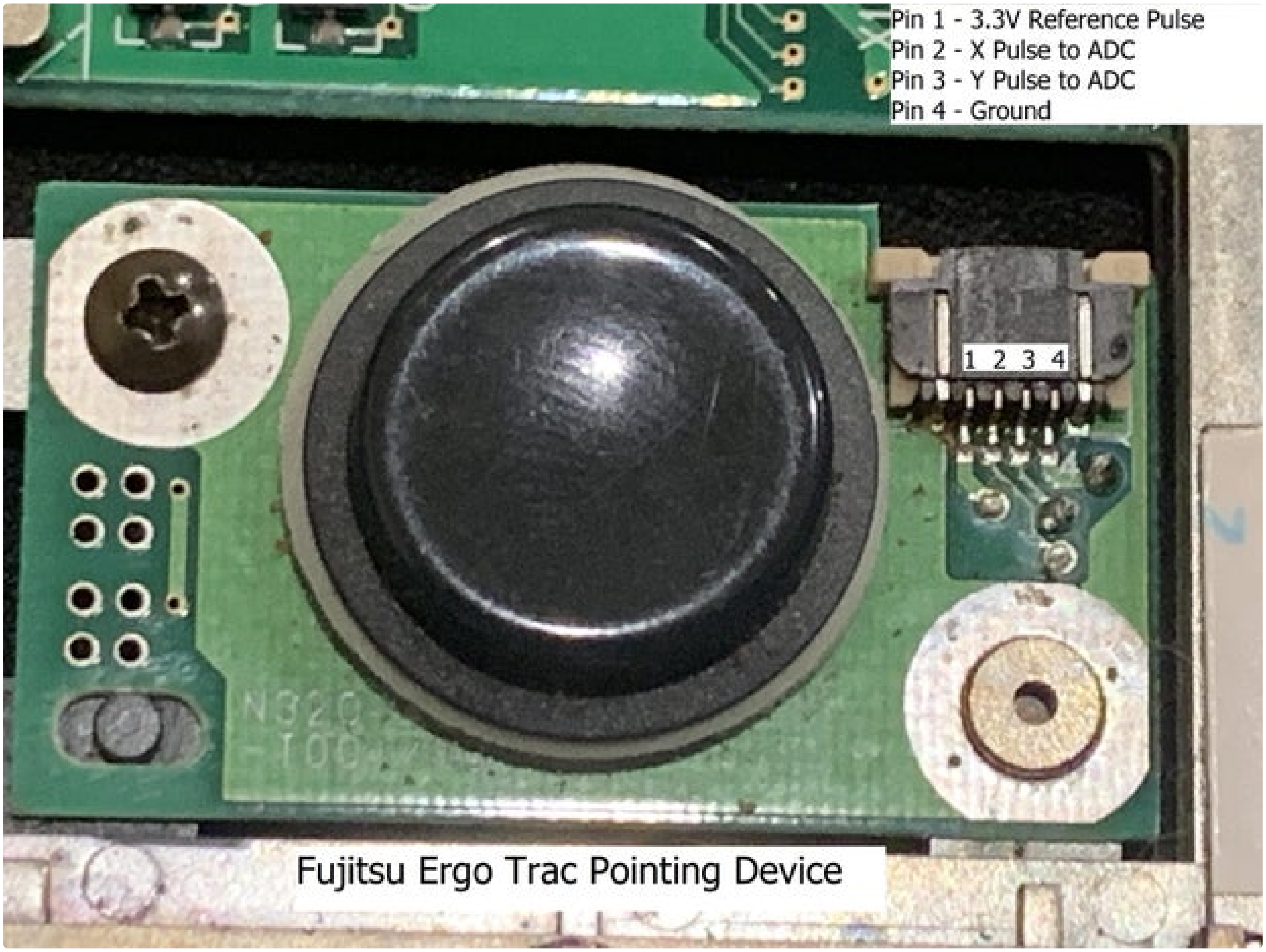


Touchpad FPC cable has 8 pins with 0.5mm pitch.

FPC connector board breaks out clock, data, power, and ground.

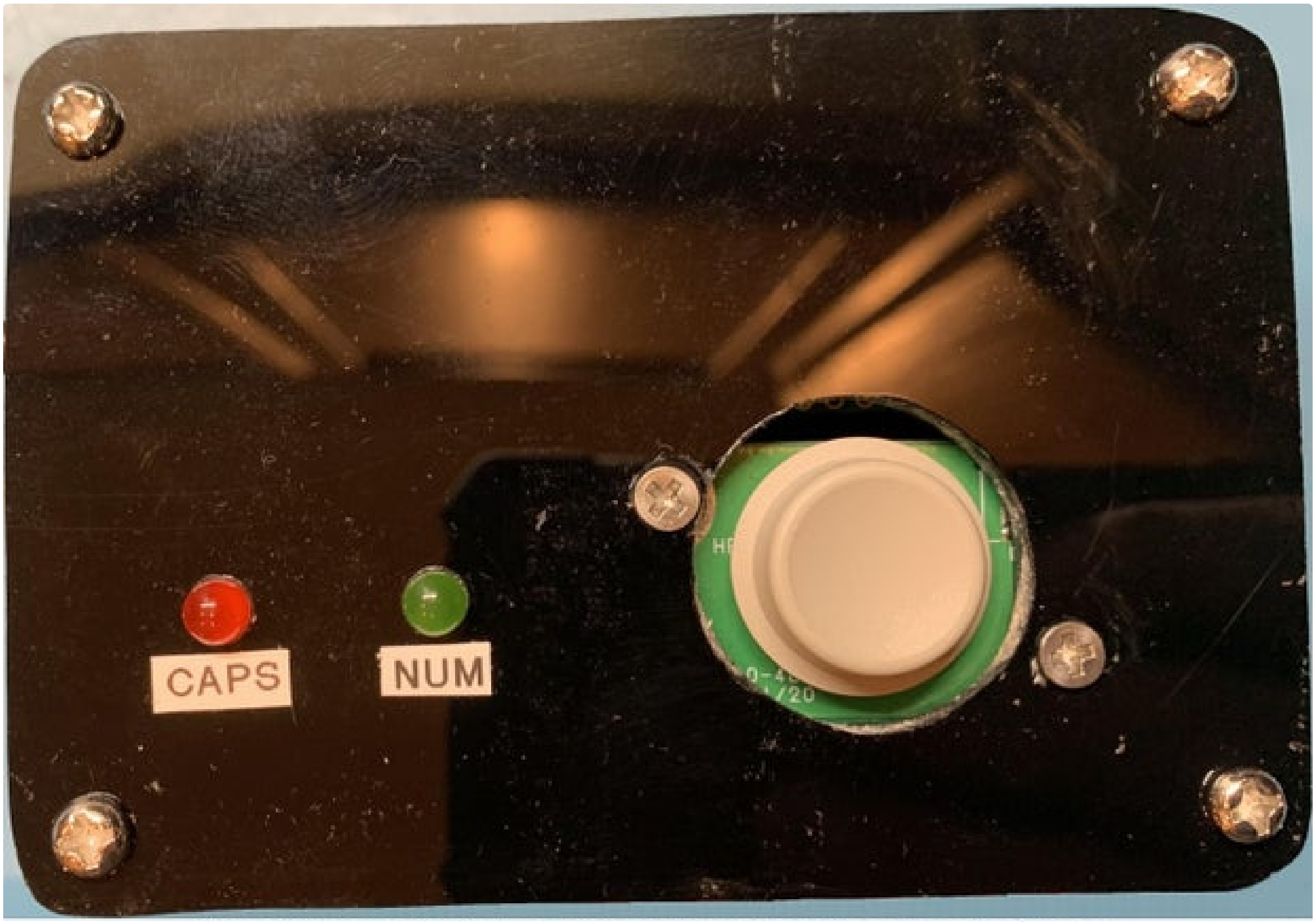



Azoteq ProxSense® TPS65
AZC278A01
PXM0012
U1
C4
C3
R1
R2
R3
mm
10
20
30
40
50
60
0.5mm

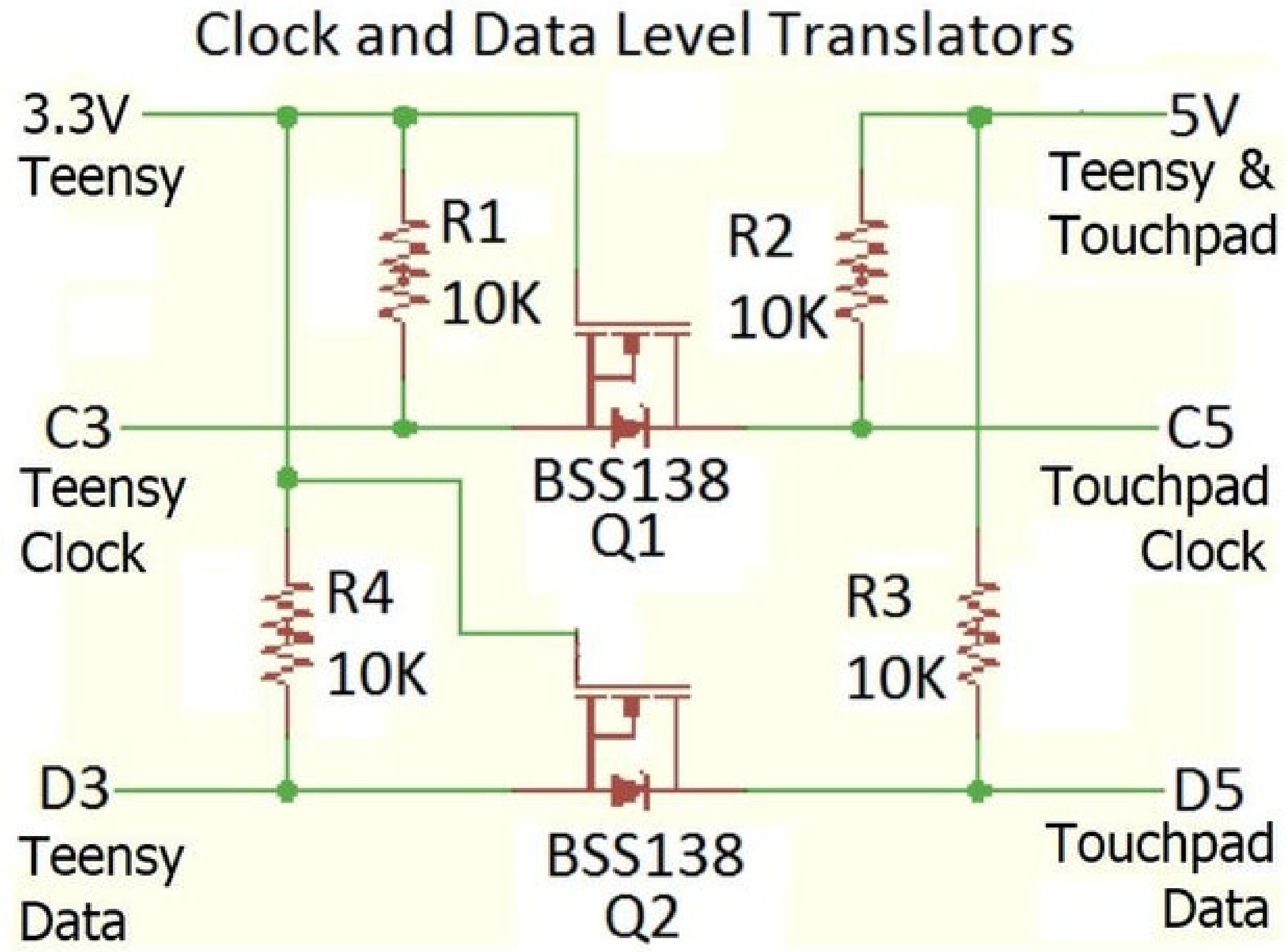


Pin 1 - 3.3V Reference Pulse
Pin 2 - X Pulse to ADC
Pin 3 - Y Pulse to ADC
Pin 4 - Ground

Fujitsu Ergo Trac Pointing Device



ErgoTrac pointing device mounted in a project box



Download

https://www.instructables.com/FS8/0UAI/JY579DKZ/FS80UAIJY579DKZ.ino

Download

https://www.instructables.com/FS8/VY60/KG54AZMB/FS8VY60KG54AZMB.ino

Step 23: Conclusion

The next step after getting the original keyboard and touchpad working is to add a video converter card to drive the LCD and make a keyboard, video, mouse (KVM). A KVM is often used to troubleshoot servers and the one shown above is from a Dell D630 laptop. I also added a Raspberry Pi inside the case to make a stand alone laptop. This build is described in my KVM Instructable. The Toshiba 2415 laptop shown above was converted into a KVM in order to display the HDMI video from a desktop PC as well as send the keyboard & touchpad data over USB. The documentation and code for this build is at my repo.

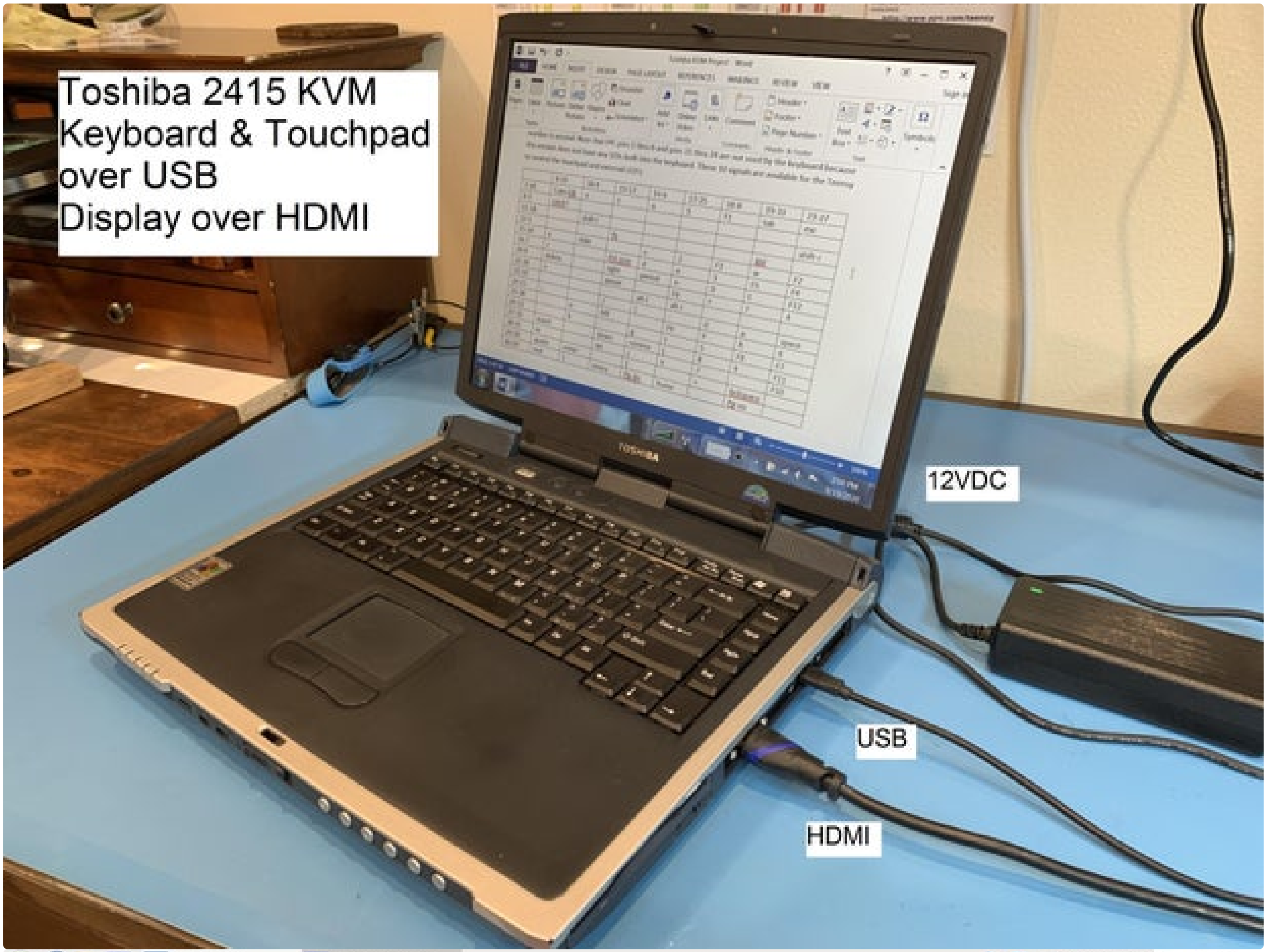
If you really want a challenge, check out my Instructable to make a "Battery Powered Raspberry Pi in a Repurposed Laptop". In addition to describing the Lithium battery charger shown above, there is Pi software to read the battery status registers over an SMBus and display a fuel gauge.

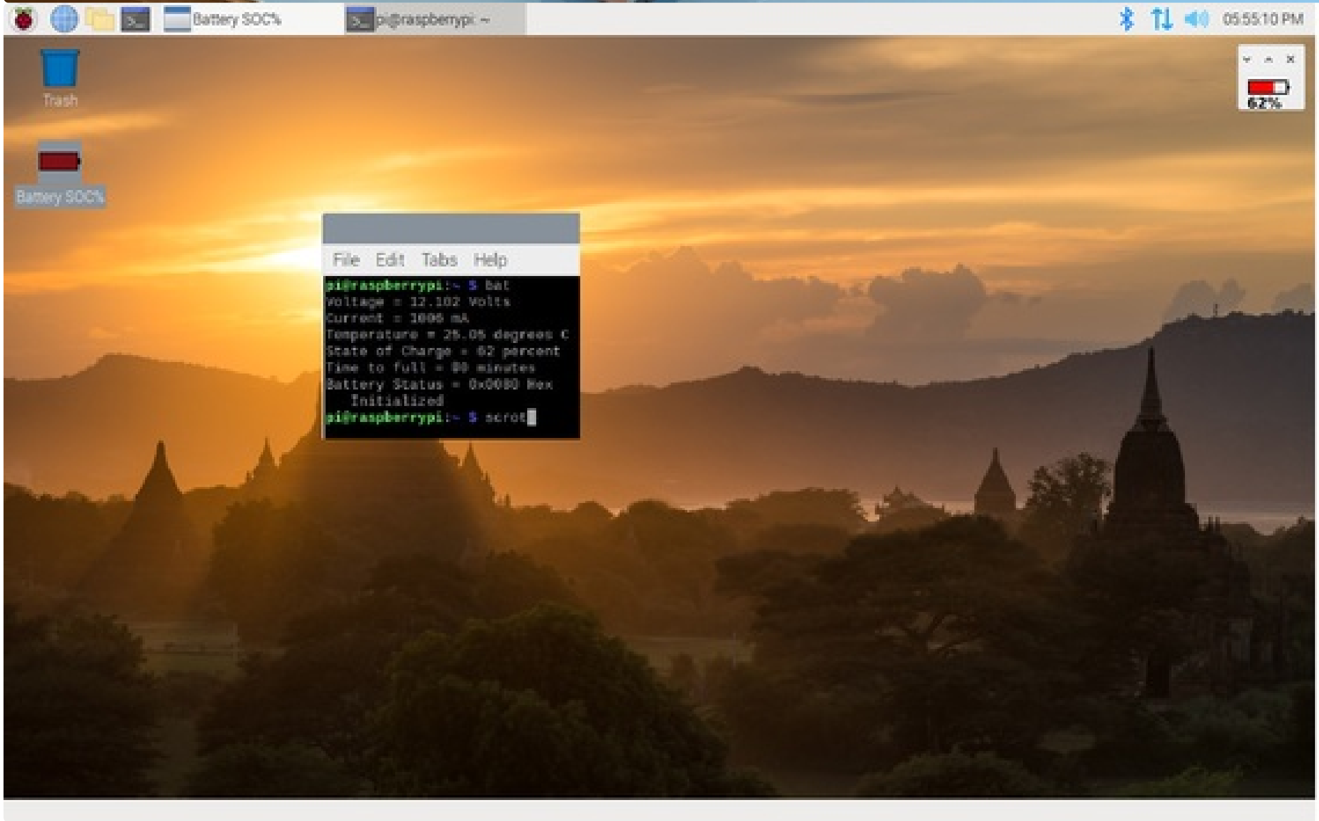
I hope you find this Instructable useful for re-purposing your old laptop. Leave a comment below if you have any questions, comments, or corrections.

Good Luck

Frank Adams

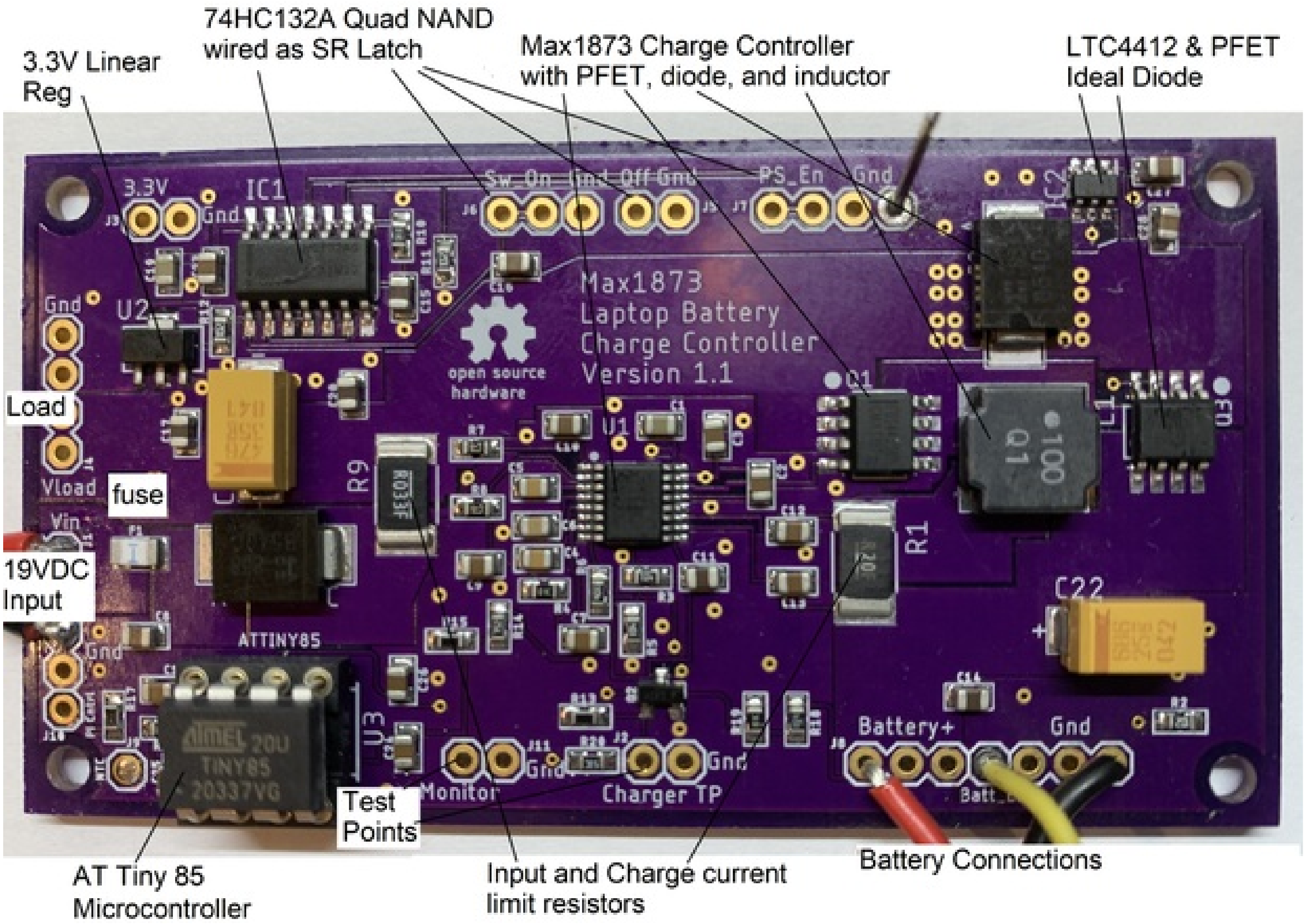
Dell D630 Converted to a KVM and Pi Laptop





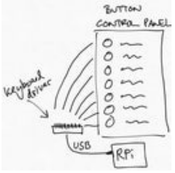
```
pi@raspberrypi:~ $ bat
Voltage = 12.102 Volts
Current = 1006 mA
Temperature = 25.05 degrees C
State of Charge = 62 percent
Time to full = 80 minutes
Battery Status = 0x0080 Hex
   Initialized
pi@raspberrypi:~ $ scrot
```

Small window to display the battery SOC in percent and as a fuel gauge


3.3V Linear Reg
74HC132A Quad NAND wired as SR Latch
Max1873 Charge Controller with PFET, diode, and inductor
LTC4412 & PFET Ideal Diode
Load
fuse
19VDC Input
Max1873 Laptop Battery Charge Controller Version 1.1
open source hardware
Test Points
AT Tiny 85 Microcontroller
Input and Charge current limit resistors
Battery Connections
Battery+
Gnd
Monitor
Charger TP

i Have several points
1- HR8201 work as USB keyboard
2- CH9329 module costumed as USB HID full keyboard mouse
3- Lenovo 300 USB Keyboard integrated IC board allow you to weld pins directly to the board without that the need to scratch the connection pins.

Nice project. Question: I like to use old USB keyboards as microcontroller input, which means soldering wires onto the keyboard PCB which is really tedious and doesn't work very well. Since you work with USB keyboards - would you have any idea of how to do this better? I am using a raspberry pi as a backbone of a "control panel" in my home, basically a larger board of mechanical buttons that each have a home automation function. Using the GPIO's on the raspberry is limited and using a USB keyboard control means access to a lot more inputs.



I assume the old USB keyboard no longer works so you can’t
just connect it to the Pi and code it to watch for a certain key sequence. First

I would take the keyboard apart and find how the switches are routed to the microcontroller. This may be with an FPC cable or with Mylar plastic sheets with metal traces. Neither one can be soldered directly so you’ll need a circuit board. If there is an FPC cable, use my PCB with an FPC connector (or buy one from Aliexpress) so you can use an ohm meter to find out which 2 traces are connected when you push a key. It will take a long time so it would make more sense to follow this
Instructable and install a Teensy with code so it can find out the key connections for you. Then put the matrix into the USB keyboard routine so it can report any key sequence you want. If you try to have the Pi do all the scanning for every key, it will use all its GPIO and still may not have enough, plus you have to write a lot of new software.

If the broken USB keyboard has Mylar plastic sheets, the trace connections are done differently and you’ll need to build a matching circuit board like I did for this Logitech keyboard: https://www.hackster.io/frank-adams/teensy-controls-a-logitech-keyboard-fujitsu-ergotrac-a0e308

The metal traces on the Mylar sheets are very resistive so you’ll need to use high value discrete resistor pullups (not the built in pullups).

Hope this helps

Thanks for the tips. I will look into the solution you are suggesting.

Took a few months, but thanks to Frank I was able to get my GRiDcase 3 laptop up and running with the original keyboard! It runs a core i5 with 8gb of ram and 500gb of ssd storage. Super exciting project, works fabulously!
https://themechanicaltype.blogspot.com/2022/12/i-needed-laptop-1980s-grid-computer.html

Wow Lucas, this project really turned out great. Nice job coming up with a new LCD and getting it all to fit back in the case. I've done a few of these projects myself and it seems like you need a shoe horn to get it all to fit inside. Enjoyed reading your step by step description as well. Congratulations.

i think on custom matrixes you can try another way.

create a Pivot with rows and cols as both Pins and Key Names as Values from Keyboard.....txt, look how you can swap row and col to have more keys only in the rows or Cols.
After hours and weeks,python programming,try and fail, i do it in 10 minutes.
just:
search&replace tabulator to comma
import in Google sheets
make a pivot.

is there any ways to turn this into Bluetooth multi device keyboard? like connecting to some Bluetooth module? any tutorial suggestion?

I started helping Ben with his Thinkpad T420 keyboard project a year ago. He got the keyboard and trackpoint working over USB with no problems. Then he did a bunch of research and code additions that make it also operate over Bluetooth using the Adafruit Bluefruit. His Github has 2 repositories, the first is forked from mine with his initial circuit board. https://github.com/orbitinstasis/USB_Laptop_Keyboa...
The second at https://github.com/orbitinstasis/T420_KB_BLE
has more improvements and includes 3D printer files. It's not documented very well yet other than comments in the code but he may be willing to help explain it better if you send him an "issue" message. Perhaps he will make a video or Instructable if there is enough interest.

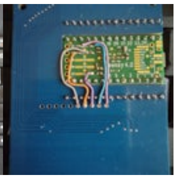
I'm confused about how modifier keys are custom coded. My keyboard requires two modifier keys for a single key, for example you can type an H, shift H for capital, and Figure H for apostrophe. I have zero coding experience and feel a bit out of my depth. I will add that I also intend to use a Teensy 4.1

Keep in mind that because you are programming your keyboard, you don't *have* to do anything. You can program it to work any way you want. The teensy will see what you pressed, then it will translate it, and tell the computer whatever you wan it to.

Take a look at the key commands that the teensy is able to send at https://www.pjrc.com/teensy/td_keyboard.html

When you want an apostrophe, you hold down the "Figure" key and press H ???
I don't know what a Figure key is. Send a picture of your keyboard to: thedalles77@gmail.com or point me to a link so I can better understand how the keys are laid out. What model laptop did this keyboard come from? How many pins and what is the pitch of the FPC cable or does this keyboard have some other connector style? send me a picture of the cable/connector as well. The Teensy 4.1 will certainly have more than enough I/O pins but I don't currently have a circuit board for it that connects to an FPC connector. How do you plan to wire the 4.1 to the keyboard? Are you using this Teensy because of availability? You will need to learn some coding and be willing to experiment because this sounds like a non-standard keyboard. I'm happy to help you but it will take some effort on your part.

I made it for a Lenovo G550 keyboard (spanish layout). I had to "dive" a bit into code to configure spanish layout (one extra key), but at last... I made it!!!
THANK YOU SO MUCH to Frank Adams (thedalles77 in github) and Marcel Hillesheim for the hard working. I'm very happy this proyect succeeded. Of course, it is not worth it from the monetary point of view (in 2022 it is EXTREMELY HARD finding a Teensy 3.2 o 4.0 board, and if you find one, it's as expensive as it was made of gold...) but it's REALLY worth it as a personal challenge.



New images, with a 3D printed shell where I can store the USB-C cable and a USB-C to USB adapter. I'm using the keyboard at the office and it works great!

Wow, you really did a professional job on the 3D case. Congratulations.

Nice Job, this looks great! If you want to share your code so others can use it, I can put it at my repo or link to wherever you keep it. Let me know if you're interested.

Yes, you have a pull request in your repo with the code. My user in git is NitemareReal

I accepted the pull request so your code is now at my repo. Fantastic job figuring out how to make your Spanish keyboard work and providing the information for others to use.
Thanks

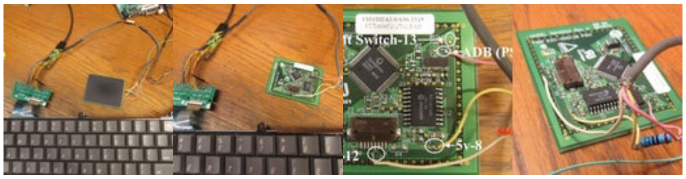
What program are you putting your keyboard matrix in?

I'm sorry, I don't quite understand the question? Teensyduino is what I used to program the matrix into the Teensy.

I see now that you used one of the "Example_Keyboards" ino files, and then edited it, putting your own matrix in.

Following Frank's touchpad tutorial I was able to use an ADB trackpad Model TM1002A from a PowerBook 1400 with a PowerBook 100 keyboard. For the Apple touchpads made by Synaptics, you only use the 5v test point 8, ground test point 12, and PS2/CLK test point 9 (this is the ADB bi-directional line). Since 1990s Apple laptops only had one button, the left switch pin is the only one used, simply wire the switch to test point 13 and ground . There is a 560 ohm pull-up resistor wired between the 5v and ADB signal wire.

The code is the same code Frank has for the Apple ADB trackball, after making all of the solder connections, I simply plugged it into the Teensy 3.2:)



What software did you end up putting your keyboard matrix into?

I built a laptop using a desktop motherboard, and I am now in the process of getting a laptop keyboard to work.

When I use the Matrix_Decoder_4p0.ino on my Teensy 4.1, each pin number is doubled. So if I connect pin 2 to pin 3, the output says 4 6.

I partly fixed the code. I think I will rewrite it without the array.

I got it working. It was a problem where con_pin was used in one some parts of the code, but not other parts of the code. I also added LED blink whenever pins are activated. LED blink can be easily disabled if someone wants to use pin 13. I also changed it to show real pin numbers, so that pin 0 is given as pin 0 instead of pin 1 (obviously the next stage would have to be adjusted for that). I also adjusted the key print out, making it simpler with less code, and also so that it can handle as many as 99 pins.

What's the reason for the con_pin array? I have not removed it yet.

Oh I see what the con_pin array is for. It's for the adapter board that you designed, so that it fits the order that you hooked up the teensy pins to the keyboard cable. I'm not using that, I'm going to be hooking it up with wires.

You wrote:

"Pitch = Total width / (N + 1) where N is the number of pins"

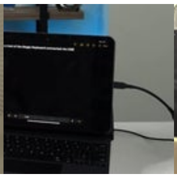
That should be:
"Pitch = Total width / (N - 1) where N is the number of pins"

I just measured 3 FPC keyboard cables and in each case, using N+1 gave me the correct pitch.
28mm wide/(34pins + 1) = 0.8mm pitch
27mm wide/(26pins + 1) = 1.0mm pitch
25mm wide/(24pins + 1) = 1.0mm pitch
It's easier to visualize if you think of a 2 pin, 1mm pitch FPC cable. It will measure 1mm from center of pin 1 to center of pin 2. It will measure 1mm from center of pin 1 to the left edge and 1mm from center of pin 2 to the right edge. The total width of the cable is 3mm. The formula gives: 3mm wide/(2pins +1) = 1.0mm pitch

Yea, I see now.

You did say "measure the width of the entire FPC cable" not "first pin to last pin". :-) I was thinking first pin to last pin. First pin to last pin would be -1, but as you correctly stated, width of cable is +1.

Thank you once again Frank! I was able to use this guide to customize Apple's Magic Keyboard for the iPad Pro, so now it has custom keys and works with other PCs and tablets!



Thanks for sharing all of this Frank (and Marcel!) I brought back to life a vintage(!) 61-key Cherry mechanical keyboard from a 1989 Psion MC400 laptop. Got together all the bits to do this in 2021 but only just got round to it in June 2022.

https://zedstarr.com/2022/06/20/repurposing-an-old...

Video of slow one-handed typing here: https://youtube.com/shorts/tcAmia2SWaY?feature=sha...

Thanks!

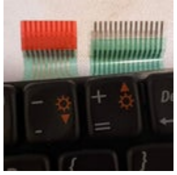
That looks like a nice keyboard. Thanks for sharing the details at your website.

Thanks for the tutorial, Frank! I was able to transform an ancient Compaq LTE 286 laptop (using the original keyboard!) into a usable Raspberry Pi computer. Check it out:
https://dmitrybrant.com/2022/02/20/a-retro-laptop-like-no-other

This looked like an amazing project for an old Acer Aspire One. I took it apart, ordered the ribbon connector, downloaded the teensy pcb file and ordered from China, soldered everything together, loaded the matrix file onto it and pressed each key corresponding to the txt file but now I'm stuck.
I can't get my head around the input/output numbers as there seem to be too many duplicates, or am I overthinking it? Every key press produced two numbers so everything seems to work, I just can't work out what to do next. Has anyone done one of these for an Acer netbook? I feel I've come too far to put it in a drawer and forget about it.

*********Update***********
After a few revisions, I was able to get the keyboard to work.

Hi Frank, I am looking at repurposing a keyboard I like from an old Vadem Clio. I ran across your instructable and this looks like exactly what I need. However my keyboard has two separate cables, so it looks like I can't actually leverage your board design. Any suggestions? I am going to get a ribbon cable and manually wire it I guess.



The 14 pin cable will be the Teensy output rows and the 10 pin cable will go to the Teensy input columns. If the FPC cables have a 1mm pitch, then you can mount the row connector on a regular Teensy LC board. Use a separate 1mm pitch bare board for the 10 pin connector and run wires from it over to the Teensy. Check out step 19 where it says:
"If you don't mind soldering a lot of wires, you can use the FPC breakout board shown above for one of the keyboard FPC connectors and put the other keyboard FPC connector and a Teensy on the regular board. The front side of the breakout board will accept an FPC connector with up to 18 pins and either a 1mm or 0.8mm pitch. "

If your FPC cables have a 1.25 mm pitch, then use two boards from the Toshiba T1200 shown at this link. https://github.com/thedalles77/USB_Laptop_Keyboard...
Mount the Teensy LC on the board for the 14 pin cable and no Teensy on the board for the 10 pin cable but run 10 wires over to the Teensy.

Thanks for the resource. I will see what I can use. As best as I can measure with my calipers the connectors are 1.33mm pitch. Unfortunately I have had no luck finding connectors so far. However my brother is an avionics technician and digging into some of his resources for retro tech, so my fingers are still crossed. I am looking at building a cable connector by hand since he recommends not trying to solder to these ribbon cables directly. I will keep you posted. ▯

Hi Joe, how did your project turn out? I came here to ask basically the same question. I've got a Clio with a bad screen, so I was going to attempt a Raspberry Pi conversion with a 7 inch screen. I really like the Clio's form factor and would like to make it a usable device.

I ended up going a different way because my nephew built me a mechanical keyboard and it fits nicely with the rest of my project. But I still have this on the back burner since I have the extra Clio keyboard. I have a couple of Clio's sitting around right now doing nothing, but I have no extra bandwidth as yet.

I've been asking on other forums about the possibility of designing a keyboard PCB that would fit into the Clio's plastics, and it sounds possible but I would need to do it myself (paying someone could run to the thousands).

I was just thinking about desoldering the ribbon connectors off one of my bad Clio boards and using those to do the wiring for a USB conversion. I'll update here if that works out, since the ribbon pitch appears to be unique.

If you ever want to clear some space and get rid of those Clios, please keep me in mind. I really enjoy using mine as a standalone word processor.

Here are some detailed photos of the keyboard membrane if anyone can use them:
https://imgur.com/a/unuyEH1

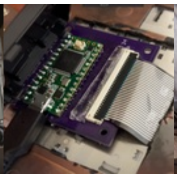
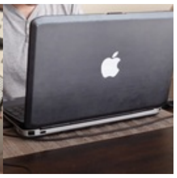
Definitely don't try to solder to the cables directly, it won't work. 1.33mm pitch isn't a standard size but 1.27mm is standard. It's difficult to measure the pitch with calipers so it might be worthwhile to order the 487576-2 and 487576-6 1.27mm pitch connectors from Mouser and see if they fit. Here are the Mouser links:
https://www.mouser.com/ProductDetail/TE-Connectivi...
https://www.mouser.com/ProductDetail/TE-Connectivi...
If they fit, you'll need to glue down both connectors on a blank board, then solder 24 wires from the connectors to a Teensy LC. This would be a good project to learn how to use Eagle layout.
Let me know how it goes. Good Luck

Good morning.
I really like you work here and I want to try to do it myself with a standard 24 pin 1mm pitch keyboard connector and I also have a teensy 3.2 laying around.
My only issue is that I don't understand if I can use the 3.2 for 24 pin since your brd with it has 34 pins and I don't know if works even with 10 less pins (and the LC brd has 24 pins, looking like the only correct one for me).
Can I use it or not?
Thank you in advance and congratulations for your work

Yes you can use a Teensy 3.2 with a 24 pin FPC connector. Solder starting at connector pin 1 and leave pads 25 thru 34 empty. You will need to solder the Teensy on the 3.2 side of the board (not the LC side). Also you will need to solder flying leads or right angle header pins in order to attach the surface mount I/O pads of the Teensy to the board.

Thanks a ton man, have a nice day

Thank you so much for this guide! Thanks to you I was able to build a custom MacBook by putting a MacBook M1 motherboard into a larger Dell Latitude shell!
https://www.youtube.com/watch?v=BLcw1Mx4fic&t=2s&ab_channel=innoiso

This is a great build. Thanks for sending pictures and a link to your video. You should be able to get the original Dell touchpad working using 2 spare IO pins from the

Teensy to send the PS/2 clock and data signals. Step 22 in this Instructable talks about integrating the touchpad with the keyboard to make a composite USB device.

There is more touchpad info at my Hackster site:

https://www.hackster.io/frank-adams/laptop-touchpa...

For access to the volume up and down keys, perhaps the 0.5mm pitch breakout board from step 21 (or something similar) can be used. Then wire these signals over to 2 spare IO pins so the Teensy code can detect when they're pushed and send the volume up or volume down HID codes over USB.

Counting up the Teensy IO pins, there's 29 traces on your keyboard FPC cable (plus pin 1 that is a dead end). Add to that the 2 needed for a touchpad and 2 needed for volume up&down for a total of 33 I/O's. If you need one more input, the 34th Teensy I/O is #13 and can be used as an input if you unsolder the LED.

I like your idea of cabling the spare video input over to the edge of the chassis for an external video source. I also like to run the USB D+, D-, and 5V from the Teensy thru a 3 pole double throw switch so it can also feed a USB connector at the chassis edge for programming the Teensy without opening the case and swapping cables. This is covered in my KVM Instructable: https://www.instructables.com/DIY-Portable-KVM-Cra...

Let me know if you have any questions.

Keep up the good work.

Frank

Thank you so much! The back of the trackpad doesn't have labels on the PINs like in the guides you have, how would I go about figuring out which is which? Do I need to find some schematic or do some multimetering?

A schematic would be great but usually hard to come by. My guide at

https://www.hackster.io/frank-adams/laptop-touchpa...

will describe how to use a multimeter and some trial and error testing with a Teensy to figure out the pins. Send me a picture of the trackpad and any meter results if you get stuck. My email is thedalles77@gmail.com