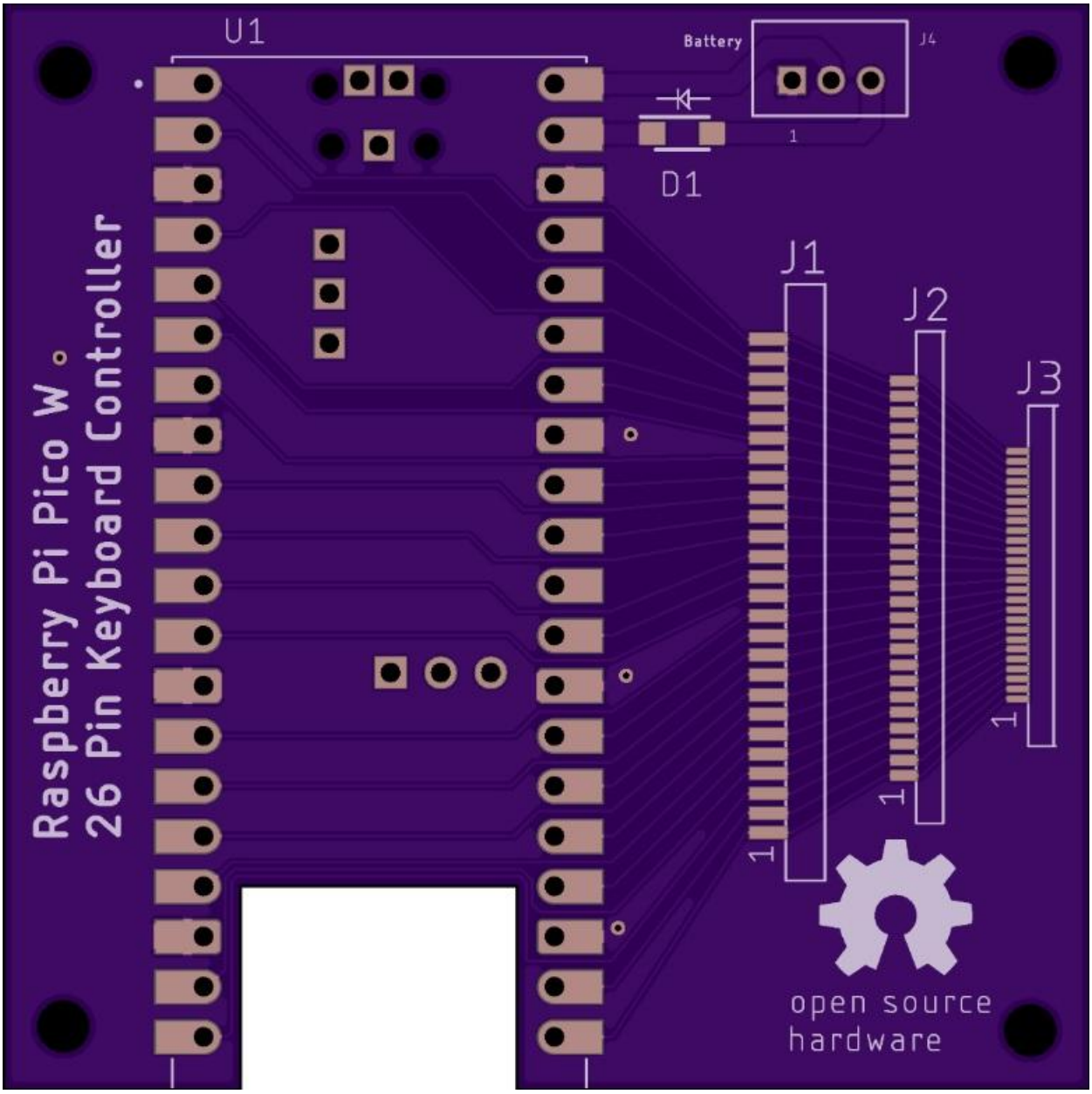
Raspberry Pi Pico W 26 Pin Laptop Keyboard Controller

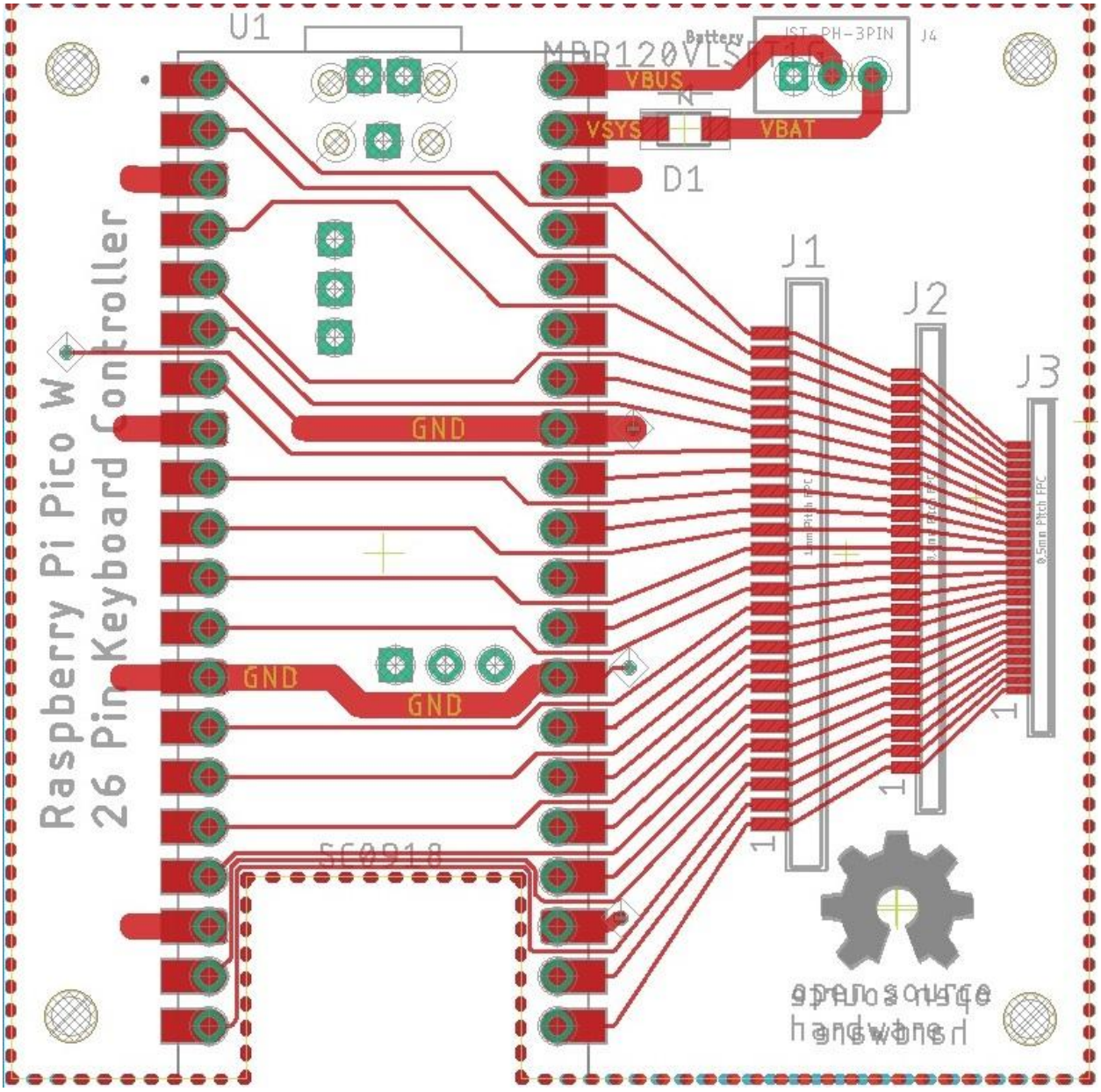
This document will describe how to make a USB and Bluetooth controller for a laptop keyboard with 26 or less FPC pins using a Raspberry Pi Pico W. There are pads to solder either a 1.0mm, 0.8mm, or 0.5mm pitch FPC connector. All associated files are at my Github repository.

The Pico 26 pin keyboard controller board is shown below as depicted by OSHPark.



The cutout in the board gives better reception for the Bluetooth antenna on the Pico. The Pico can be mounted with header pins or soldered directly to the board for a lower profile. If the keyboard will not be used for Bluetooth, it will always have USB power so the Schottky diode (D1) and 3 pin JST connector (J4) are not needed. These components are only used for battery operation with Bluetooth.

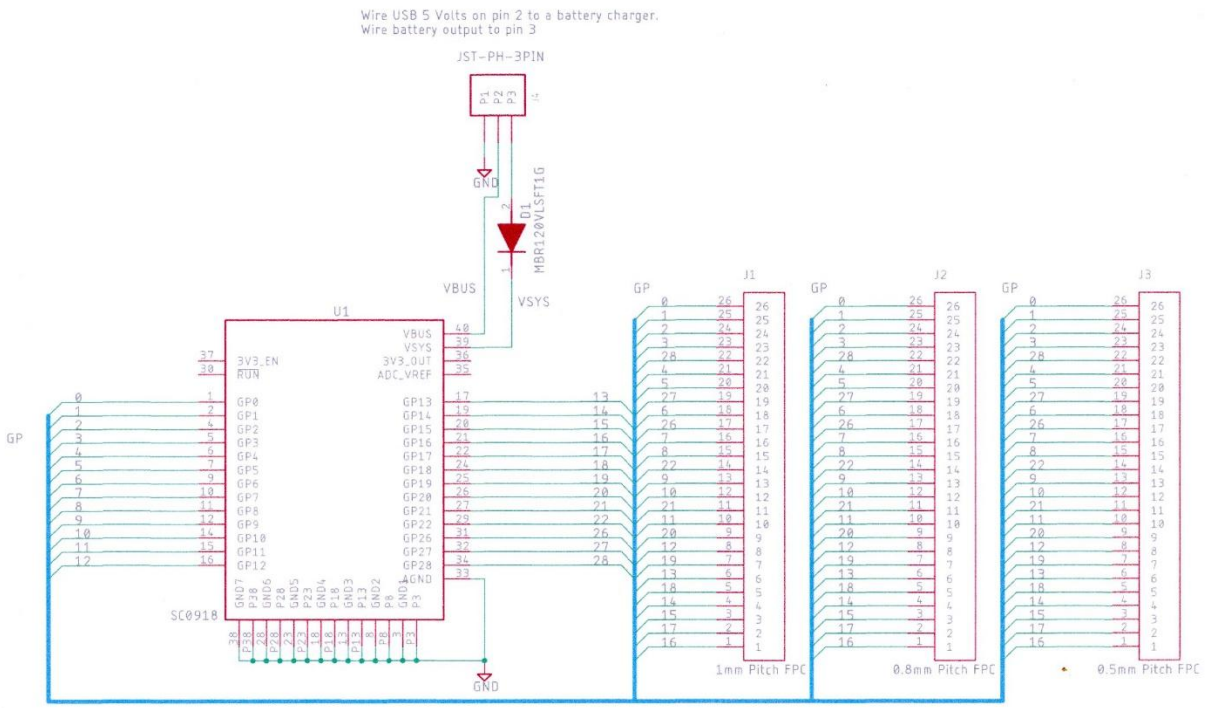
The “Pico_26Pin_Keyboard.brd” Eagle file and “Pico_26Pin_Keyboard.zip” Gerber file at my repo can be fabricated by OSH Park or other fab houses like JLCPCB. The Eagle layout (without area fill) is shown below so you can see the signal traces from the GPIO pins to the 3 FPC connector pads.



The circuit board measures 55mm x 55mm (2.17” x 2.17”).

If you plan to use Bluetooth, the controller will be running from a lithium battery. To charge the battery, plug in a USB cable to the Pico. USB 5 volts on the Pico VBUS pin is provided on J4 pin 2 (this is a 3 pin JST connector). This should be cabled along with ground to a battery charging circuit. The nominal 3.7 volt battery output should be cabled to the JST connector J4 pin 3. This feeds an MBR120VLSFT1G Schottky diode (D1) that “or-ties” the battery voltage to the Pico’s VSYS pin. There is another Schottky diode in the Pico that brings USB power to VSYS when the USB cable is attached.

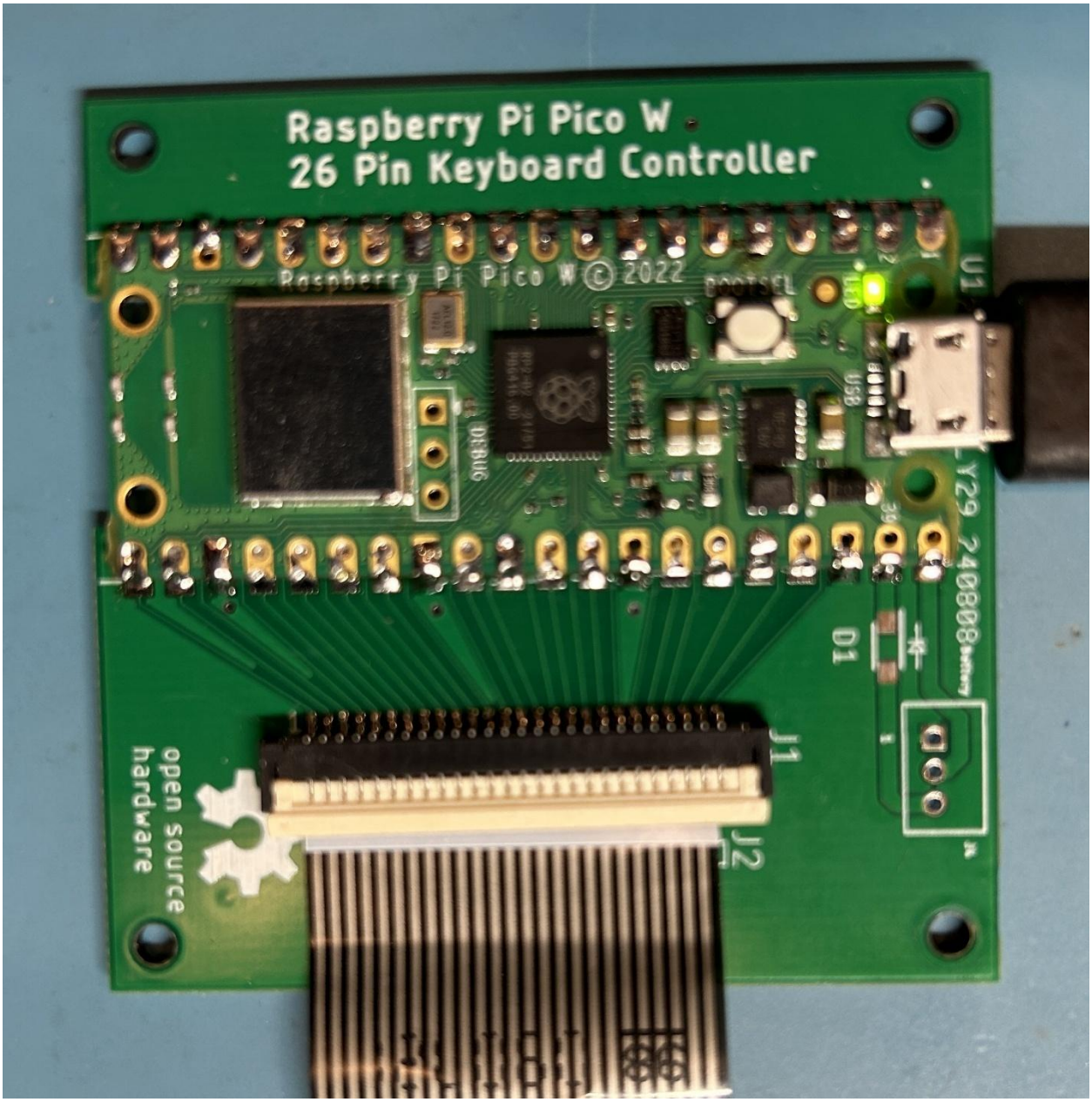
The Eagle schematic “Pico_26Pin_Keyboard.sch” at my repo is shown below.



The keyboard connections to the Pico GP I/O are shown below.

FPC Connector pin number	Pico GP I/O number
1	16
2	17
3	15
4	14
5	18
6	13
7	19
8	12
9	20
10	11
11	21
12	10
13	9
14	22
15	8
16	7
17	26
18	6
19	27
20	5
21	4
22	28
23	3
24	2
25	1
26	0

The assembled circuit board with a Raspberry Pi Pico W and a 26 pin 1mm pitch FPC connector is shown below. The FPC cable from a Dell Inspiron 1545 keyboard is attached. The procedure to make this example keyboard operational over USB will be described on the following pages.



I soldered this Pico directly to the connector board for a low profile but it could also be soldered with header pins. The diode at D1 and the JST connector at J4 are not yet installed for Bluetooth battery operation.

A good starting point for learning how to write Python code for the Pico is at circuitpython.org/. Select the "Get Started" box. The following is a synopsis of the steps that they provide.

To enter into boot loader mode, hold the Pico push button down while plugging in the Pico's USB cable. The new folder will show as a drive named RPI-RPI2.

If the Pico is acting weird or you want to start with a clean slate, copy the flash_nuke.uf2 file (downloaded here) over to the Pico's RPI-PPI2 folder. The drive will blink out and then return all clean.

Python - Start with the circuitpython.org/downloads page, then select your Pico model. Download and copy the latest Adafruit Circuit Python uf2 over to the Pico's RPI-PPI2 folder. The drive will blink out and return as CIRCUITPY with an empty lib folder. The code.py file is a one-line program that prints "Hello world". You will be replacing this file later.

Keyboard library – Go to circuitpython.org/libraries and download the library bundle that matches the version of Python you are using. The bundle includes so many libraries that they can't all fit in the Pico's memory. Only copy the Adafruit_HID folder into the lib folder on the Pico.

As you write and debug your Python code you will need an editor. Adafruit likes the Mu editor and they give links for installation in their tutorial. I prefer Thonny and it can be downloaded at thonny.org/. My descriptions below are based on Thonny.

Download my Matrix_Decoder.py code, then start Thonny. Select File, Open, This Computer and navigate to the Matrix_Decoder.py code. This will bring it up in the editor and you can read thru the comments to get an idea of how it works. On Thonny, select the "Stop" icon to halt any program that is running on the Pico. Next select File, Save As, CircuitPython Device. Overwrite the existing code.py file with your Thonny code. Select the "Run" icon so the Pico will run the code.py program. As you debug your code, you will need to hit the Stop icon in order to save it to the Pico. Be sure to also save your updated code to your computer.

Bring up another editor like Notepad++ and load a blank key_list text file that lists all the possible keyboard keys. Place the cursor to the far right of the first key in the list. This is where the Python program will send the Pico GPIO numbers when you push a key. Then the program will send a down arrow to position the cursor for the next key. The pin connections shown below are for a Dell Inspiron 1545 keyboard (model NSK-D9301) that I used as an example keyboard.

KMK Keycode Name	GPIO#	GPIO#
LCTRL	27	8
RCTRL	8	2
LSHIFT	22	5
RSHIFT	22	2
LALT	9	6
RALT	9	1
LGUI	7	4
FN	26	3
A	10	3

B		20	1
C		11	2
D		11	3
E		28	11
F		20	3
G		20	6
H		19	6
I		28	13
J		19	3
K		13	3
L		14	3
M		19	2
N		19	1
O		28	14
P		28	18
Q		28	10
R		28	20
S		21	3
T		20	5
U		28	19
V		20	2
W		28	21
X		21	2
Y		19	5
Z		10	2
GRAVE	`	27	10
N1	(NUMBER 1)	10	4
N2		21	4
N3		11	4
N4		20	4
N5		27	20
N6		27	19
N7		19	4
N8		13	4
N9		14	4
N0		18	4
MINUS		27	18
EQUAL		27	13

BSPACE	12	5
ESC	10	6
F1	27	21
F2	27	11
F3	11	5
F4	11	6
F5	12	6
F6	13	6
F7	14	5
F8	27	14
F9	27	12
F10	12	4
F11	17	4
F12	16	4
INSERT	27	16
DELETE	27	17
RIGHT (ARROW)	16	1
LEFT	15	1
UP	15	6
DOWN	17	1
APP (MENU)	15	5
SLASH /	18	1
DOT . (PERIOD)	14	2
COMMA ,	13	2
SCOLON ;	18	3
QUOTE '	18	6
ENTER	12	2
LBRC [	18	5
RBRC]	13	5
BSLASH \	12	3
CAPS	21	5
TAB	10	5
SPACE	12	1
HOME	27	15
END	15	4
PGUP	28	7
PGDOWN	7	3
PSCREEN (PRINTSCREEN)	9	4

EJCT (EJECT)	28	9
MPRV (PREVIOUS)	12	4
MNXT (NEXT)	16	4
VOLD (VOL DOWN)	27	14
VOLU (VOL UP)	27	12
MPLY (PLAY/PAUSE)	17	4
MUTE	14	5
BRIU (BRIGHTNESS UP)	12	6
BRID (BRIGHTNESS DOWN)	11	6

Dell Keyboard Connected to the Pico

With the connection data from the key list, you can build a key matrix table like the one shown below. The KMK keycode names are given here.

	GP1	GP2	GP3	GP4	GP5	GP6	GP27	GP28
GP7			PGDOWN	LGUI				PGUP
GP8		RCTRL					LCTRL	
GP9	RALT			PSCREEN		LALT		EJCT
GP10		Z	A	N1	TAB	ESC	GRAVE	Q
GP11		C	D	N3	F3	F4	F2	E
GP12	SPACE	ENTER	BSLASH	F10	BSPACE	F5	F9	
GP13		COMMA	K	N8	RBRC	F6	EQUAL	I
GP14		DOT	L	N9	F7		F8	O
GP15	LEFT			END		UP	HOME	
GP16	RIGHT			F12			INSERT	
GP17	DOWN			F11			DELETE	
GP18	SLASH		SCOLON	N0	LBRC	QUOTE	MINUS	P
GP19	N	M	J	N7	Y	H	N6	U
GP20	B	V	F	N4	T	G	N5	R
GP21		X	S	N2	CAPS		F1	W
GP22		RSHIFT			LSHIFT			
GP26			FN					

The table below is the same as above except the media keys replace the function keys. These are sent when Fn is held down. The table of KMK Media keys can be found here.

	GP1	GP2	GP3	GP4	GP5	GP6	GP27	GP28
GP7			PGDOWN	LGUI				PGUP
GP8		RCTRL					LCTRL	
GP9	RALT			PSCREEN		LALT		EJCT
GP10		Z	A	N1	TAB	ESC	GRAVE	Q
GP11		C	D	N3	F3	BRID	F2	E
GP12	SPACE	ENTER	BSLASH	MPRV	BSPACE	BRIU	VOLU	
GP13		COMMA	K	N8	RBRC	F6	EQUAL	I
GP14		DOT	L	N9	MUTE		VOLD	O
GP15	LEFT			END		UP	HOME	
GP16	RIGHT			MNXT			INSERT	
GP17	DOWN			MPLY			DELETE	
GP18	SLASH		SCOLON	N0	LBRC	QUOTE	MINUS	P
GP19	N	M	J	N7	Y	H	N6	U
GP20	B	V	F	N4	T	G	N5	R
GP21		X	S	N2	CAPS		F1	W
GP22		RSHIFT			LSHIFT			
GP26			FN					

KMK is open-source firmware normally used for mechanical keyboards but it also works for laptop keyboards, as long as you know how the switches are wired (see above tables). KMK uses CircuitPython which is why I also wrote my matrix decoder code in CircuitPython. All of the hard work of scanning key switches and communicating over USB is embedded in KMK. All you need to do is edit my KMK Dell example code with your key matrix information. Getting started with KMK is described at their GitHub repo which I will detail below.

For step 1, you already have CircuitPython installed.

For step 2, click on “get an up-to-date copy of KMK”.

For step 3, unzip the kmk_firmware-main folder. The top folder is kmk_firmware-main. It has many folders and files. Copy the KMK folder and boot.py to the CIRCUITPY drive on the Pico.

For step 4, use my main_Dell1545 KMK keyboard code as a starting point. Load it into Thonny and edit it as follows:

Change keyboard.col_pins to list the GPIO column pins left to right across the top of your matrix.

Change keyboard.row_pins to list the GPIO row pins top to bottom along the side of your matrix.

Change the keyboard.keymap matrix for the base layer and the Fn media layer per the tables you created (see above). Each keycode name (except FN) must be preceded with KC. Put KC.NO if there is no key at a matrix location. After saving the code to your computer, save it to the Pico’s CIRCUITPY drive with the name code.py (this will overwrite your previous code.py that was the matrix decoder).

With the code.py still In Thonny, click the “Stop” icon, then click the “Run” icon. You should now be able to bring up a blank editor like Notepad++ and type on the keyboard to test if all the keys work. If you have any typo’s, they will be reported in Thonny.

For normal USB keyboard operation (without Thonny), you can just plug in the cable. It will bring up a CIRCUITPY drive folder which you can close.