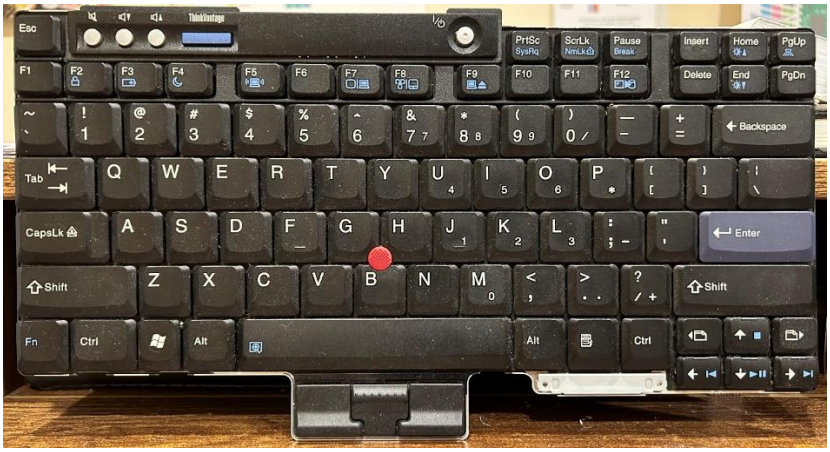
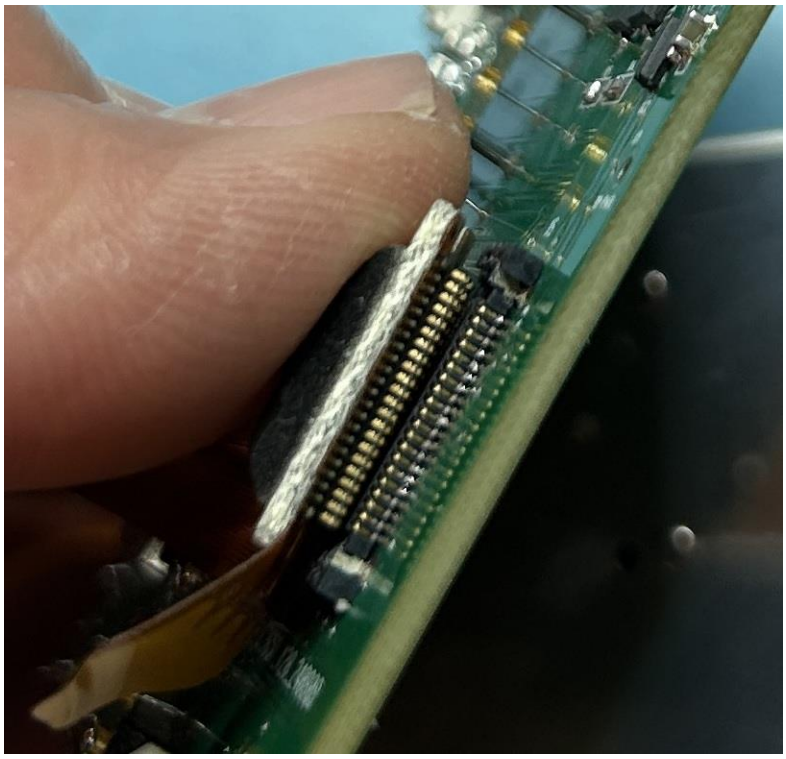
Raspberry Pi Pico W Keyboard Controller - 12/23/2024

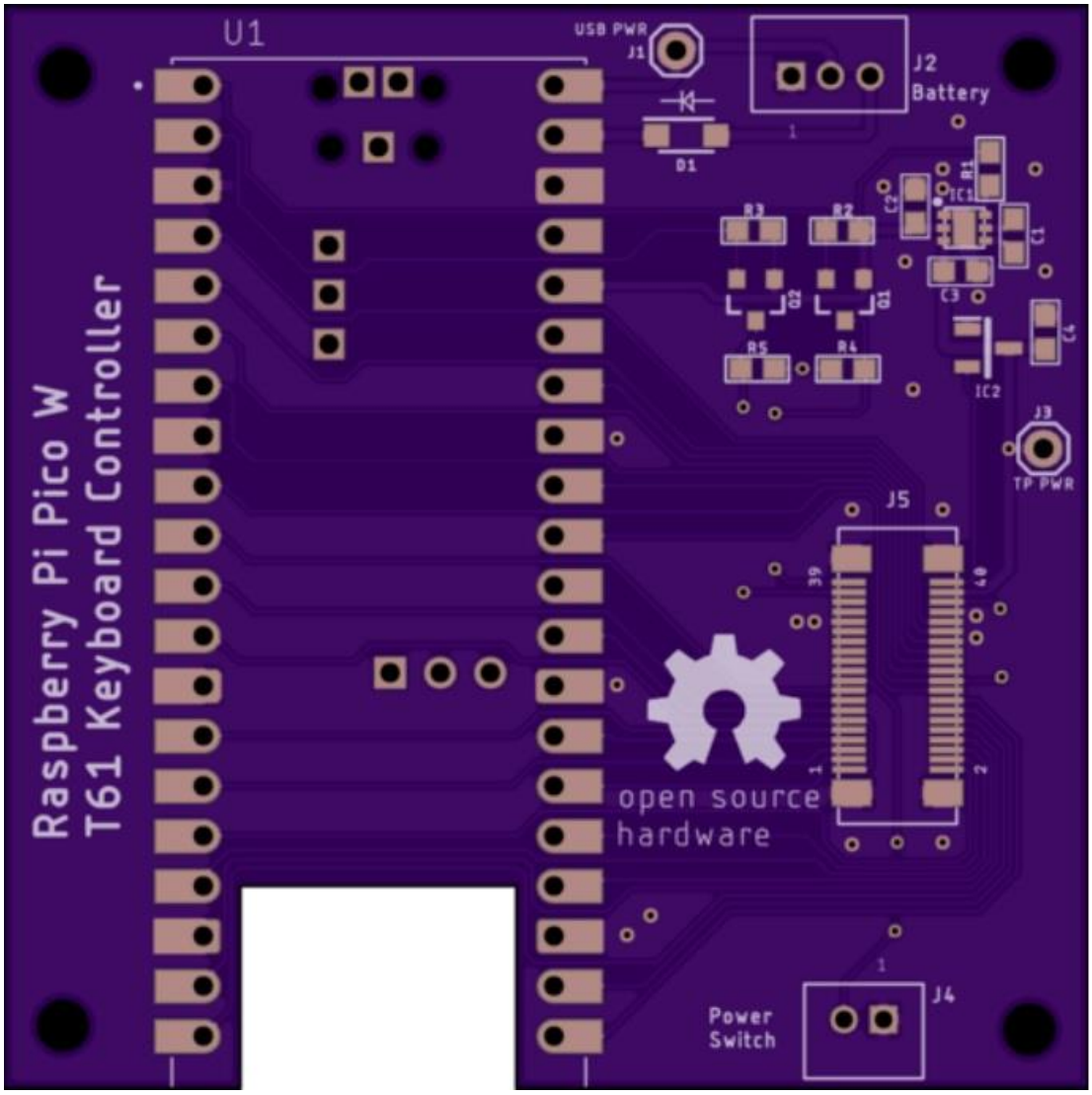
This document will describe a Raspberry Pi Pico W USB and Bluetooth controller for the Thinkpad T60 series keyboards such as the T61 shown below. This is an ongoing project that has been built and tested but needs further software development. I'm providing my design information to help others and perhaps they can help me with the software. All associated files are at my Github repository and I've added this project to step 17 of my Instructable so you can add comments or send me a direct message.



The controller board will also work with 220, 410, 420, 510, and 520 Thinkpad keyboards. These other keyboards look slightly different but the key matrix and connector are the same. The 44 pin connector soldered to the end of the FPC cable has 4 corner pins that are not used when installed in the 40 pin board connector. The board connector part number is AA01B-S040 and is available from Ali-Express. The picture below shows how to manually align the contacts so there is an extra contact on each end.

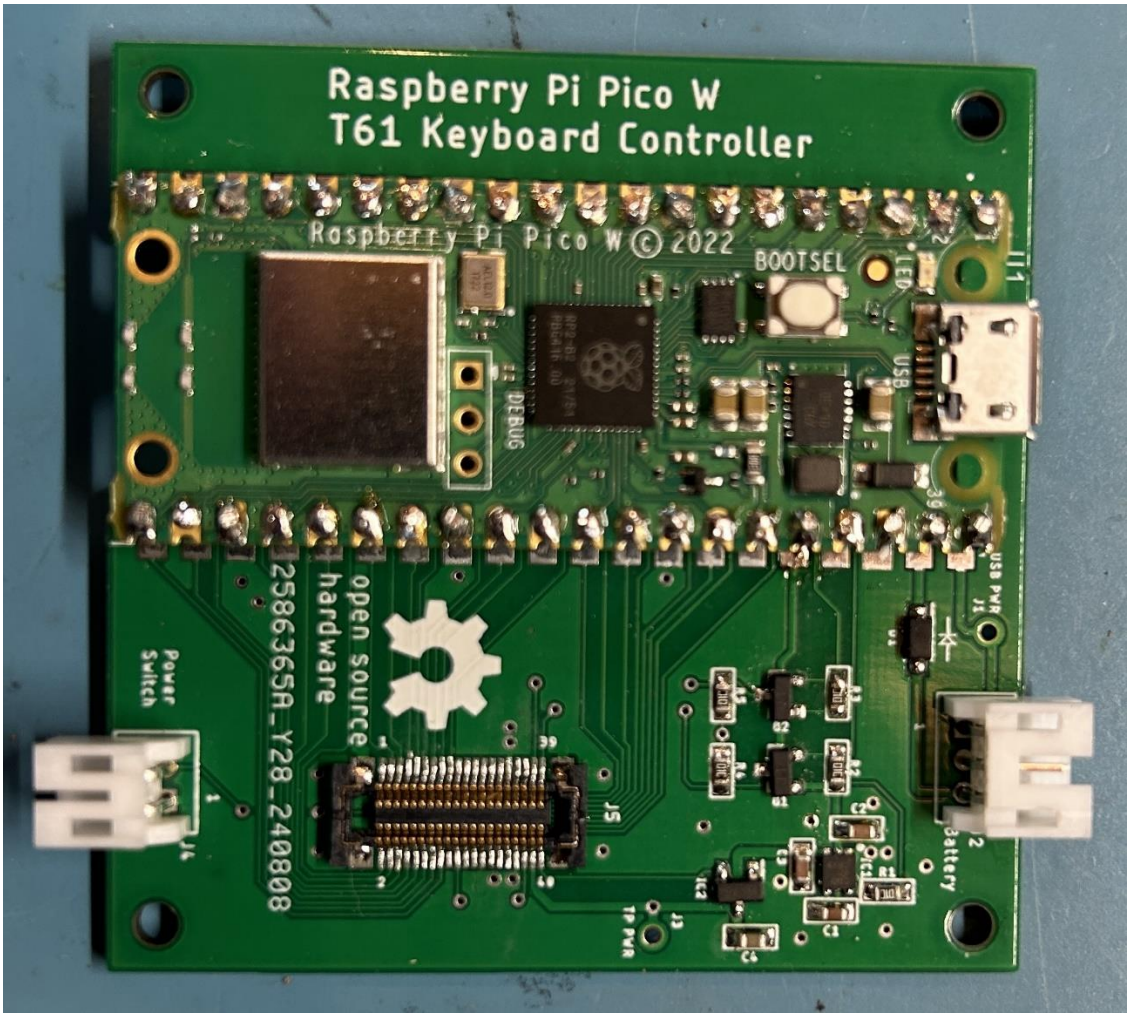


The GPIO pins of the Raspberry Pi Pico W are connected to the 40 pin AA01B-S040 connector with the circuit board shown below. The other surface mount components provide level translation, reset generation, and 5 volt power for the TrackPoint.

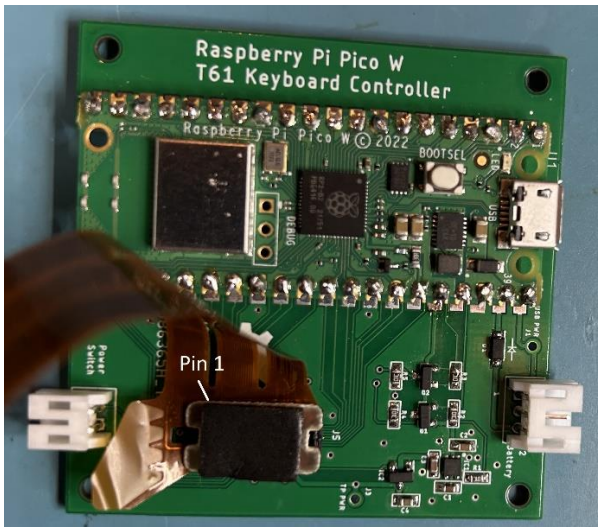


The cutout in the board gives better reception for the Bluetooth antenna on the Pico. The Pico can be mounted with header pins or soldered directly to the board for a lower profile. The Eagle file "Pico_T61_Keyboard.brd" can be sent to OSH Park or Eurocircuits. The Gerber file "Pico_T61_Keyboard.zip" can be sent to any fab house such as JLCPCB or PCBWay. Both layout files are in the Eagle directory at my repo. Hand soldering the keyboard connector with an iron is challenging. I prefer applying solder paste to the surface mount pads using a stencil. You can use a hot plate and heat gun to melt the solder but a reflow oven, such as the converted toaster oven described here, makes the job much easier.

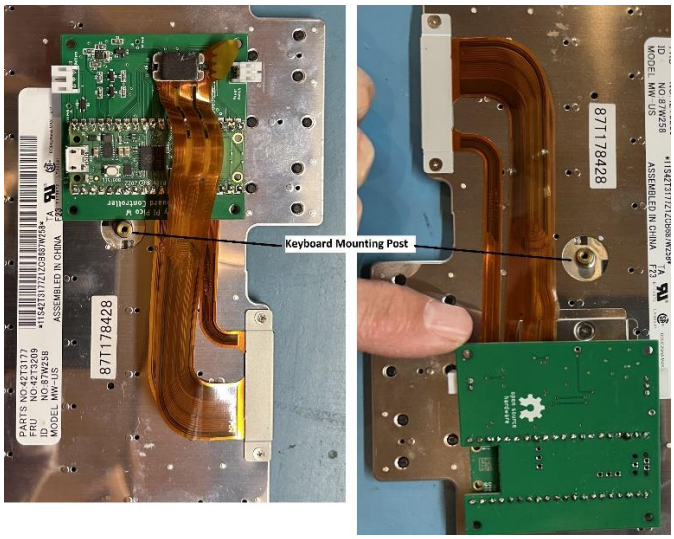
The picture below shows the assembled board. This board was manufactured by JLCPCB.



There is no keying so it's possible to rotate the keyboard connector 180 degrees. See the picture below for the correct orientation of pin 1 on the FPC cable connection.



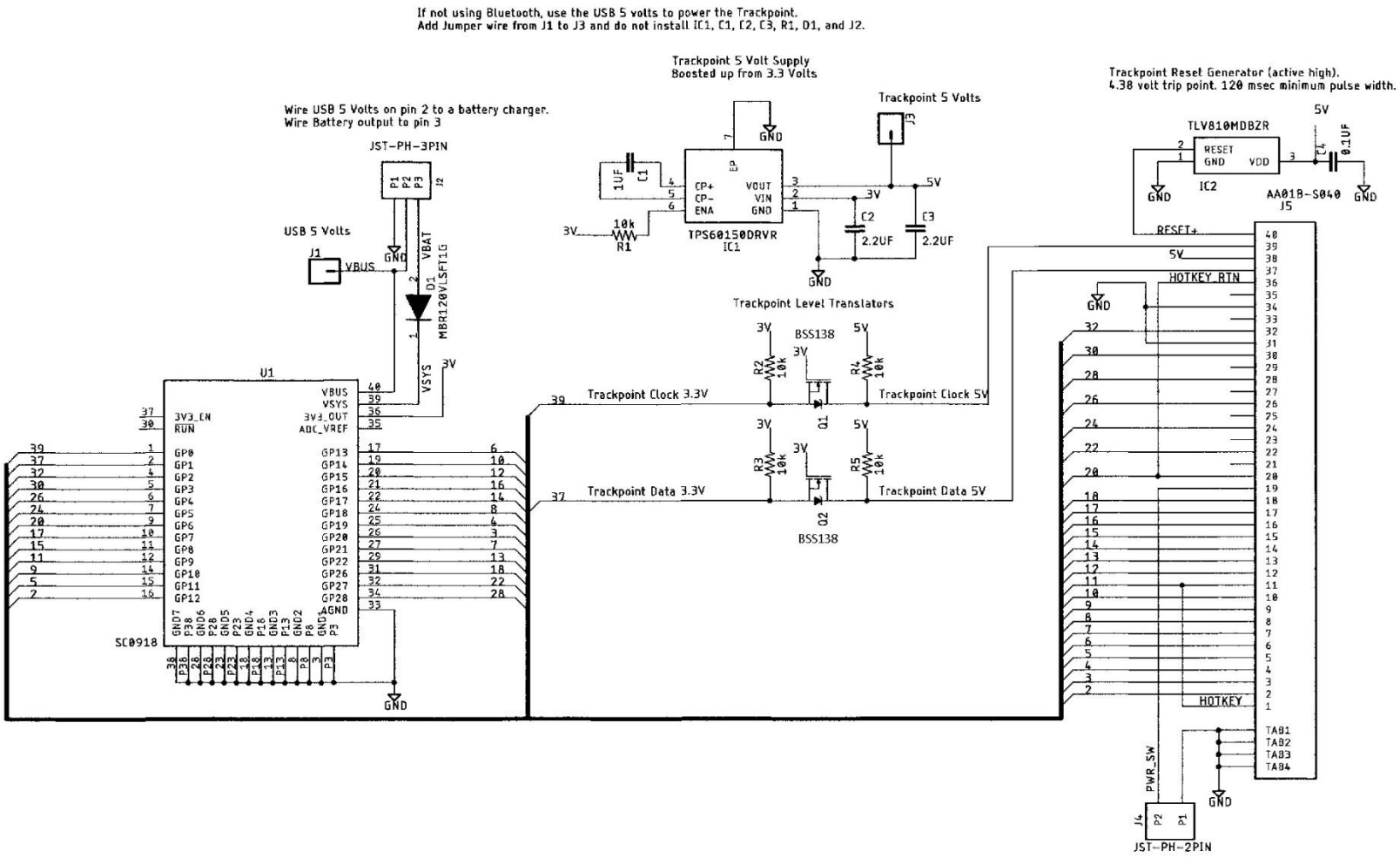
The circuit board can either be mounted facing up or down as shown below. The keyboard mounting post clears the board in both cases.



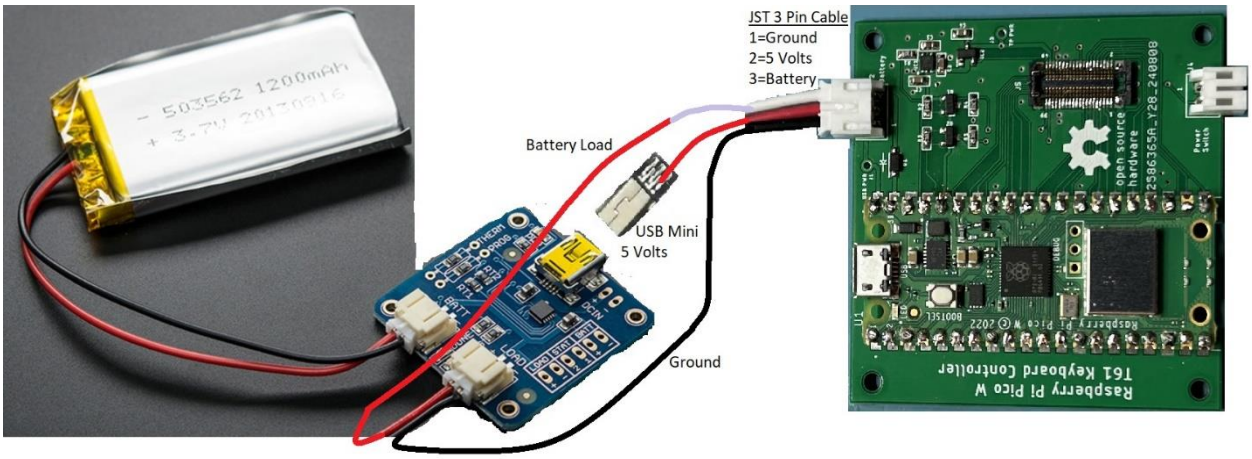
The Eagle schematic “Pico_T61_Keyboard.sch” is at my repo, and shown on the next page. The TrackPoint circuitry in the keyboard is powered with 5 volts and draws about 6ma. The Pico GP I/O’s are not 5 volt tolerant so there are BSS138 N channel FETs labeled Q1 and Q2 that act as level translators for the TrackPoint Clock and Data signals. The TrackPoint reset signal can be created with a resistor and capacitor but to be more reliable, I’ve included a TLV810MDBZR reset generator labeled IC2. It creates a 120msec active high reset pulse at 4.38 volts. A 2 pin JST connector at location J4 can be installed if you have an off board power latch circuit that will be controlled by the keyboard’s “Power” button.

Bluetooth operation implies the board will be running from a lithium battery which ranges from 3 to 4.2 volts. When the battery needs charging, a USB cable is connected to the PICO from a power source. Bluetooth operation can continue while the battery is charging. USB 5 volts is “or tied” with the battery voltage via Schottky diodes. These diodes feed the Pico’s internal 3.3 volt buck/boost regulator. One diode is internal to the Pico and the other is labeled D1 on the board. I’ve routed the Pico’s 3.3 volts to a TPS60150 switched capacitor charge pump labeled IC1 that creates the 5 volts to power the TrackPoint and level translators.

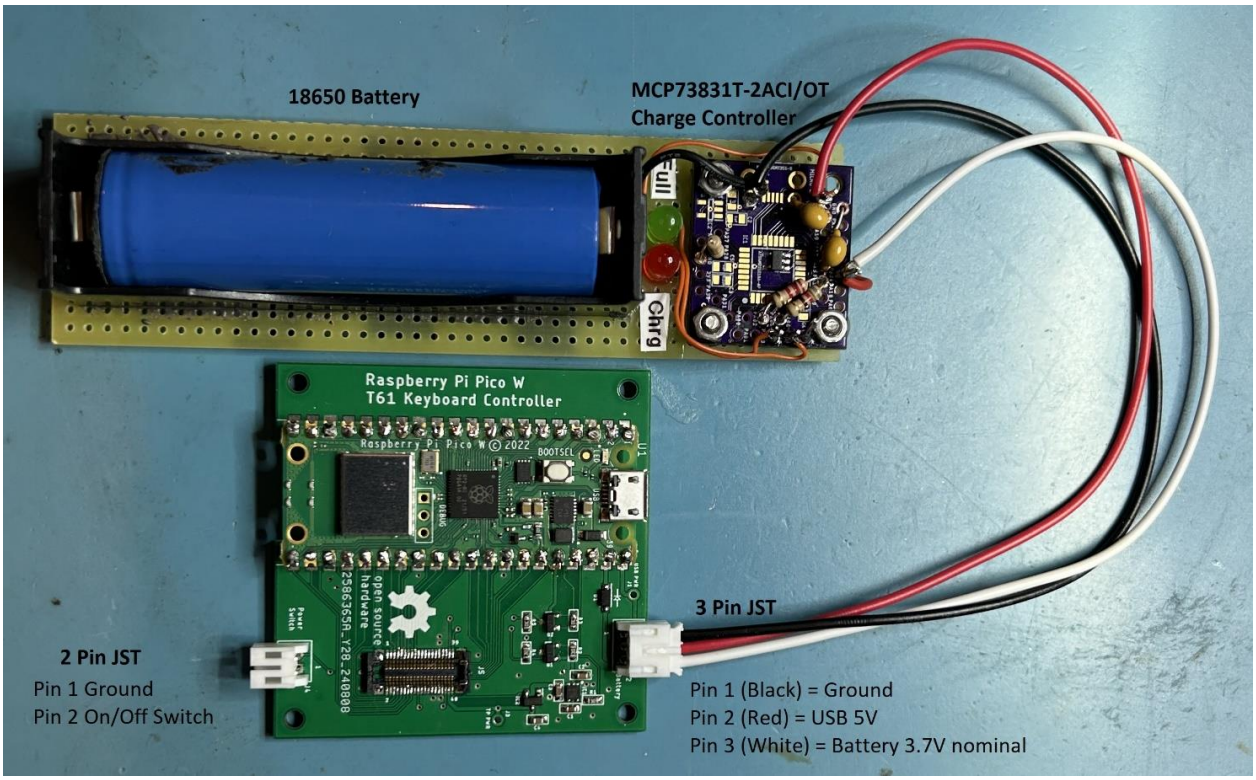
If the keyboard will only be used for USB, solder a jumper wire between test points J1 and J3. This routes USB 5 volts directly to the TrackPoint. Do not install charge pump IC1 or its associated components, R1, C1-C3. Do not install Schottky diode D1 or the 3 pin JST connector at J2. You will still need the PS/2 level translators at Q1, Q2, R2-R5, for the TrackPoint clock and data signals.


If not using Bluetooth, use the USB 5 volts to power the Trackpoint.
Add Jumper wire from J1 to J3 and do not install IC1, C1, C2, C3, R1, D1, and J2.
Trackpoint 5 Volt Supply
Boosted up from 3.3 Volts
Trackpoint 5 Volts
Trackpoint Reset Generator (active high).
4.38 volt trip point. 120 msec minimum pulse width.
Wire USB 5 Volts on pin 2 to a battery charger.
Wire Battery output to pin 3
JST-PH-3PIN
USB 5 Volts
TPS60150DRVR
IC1
TLV810MDBZR
IC2
AA018-S040
J5
Trackpoint Level Translators
BSS138
Trackpoint Clock 3.3V
Trackpoint Clock 5V
Trackpoint Data 3.3V
Trackpoint Data 5V
MBR120VLSFT1G
SC0918
U1
RESET+
HOTKEY_RTN
HOTKEY
PWR_SW
JST-PH-2PIN

The 3 pin JST connector labeled J2 on the board is cabled to the battery and charging circuit. Pin 1 on the JST connector is ground, pin 2 is USB 5 volts, and pin 3 is the battery output. This picture shows the cable connections using Adafruit components. Their charger board needs a USB mini connector to feed in 5 volts.



Instead of buying a lithium battery and charging circuit, I put my own together using an MCP73831T-2ACI/OT controller chip and battery salvaged from a laptop (see below).



Parts list with supplier links for the Assembled Circuit Board is shown below.

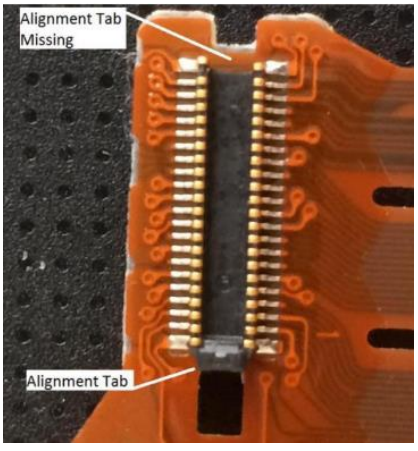
Qty	Part Number	Ref	Description
1	KGM15BR71H104JM	C4	0.1uF 0603 X7R Ceramic Chip Capacitor 50V 10%
5	RC0603JR-0710KL	R1-R5	10K 0603 5% tolerance 1/10W Chip Resistor
1	KGM15AR71C105KT	C1	1 uF 0603 X7R Ceramic Chip Capacitor 16V 10%
2	KGM15BR51A225KM	C2, C3	2.2 uF 0603 X5R Ceramic Chip Capacitor 10V 10%
2	BSS138	Q1, Q2	BSS138 N-Channel MOSFET SOT23-3 50V 230ma
1	AA01B-S040	J5	40 Pin Keyboard Connector
1	S2B-PH-K-S	J4	2 Pin right angle JST Board Connector 2mm
1	S3B-PH-K-S	J2	3 Pin right angle JST Board Connector 2mm
1	MBR120VLSFT1G	D1	Schottky Diode 20V 1A
1	Raspberry Pi Pico W	U1	Raspberry Pi Pico W with or without Header pins
1	TLV810MDBZR	IC2	Reset Generator Active High 4.38V 120ms Push-Pull SOT23-3
1	TPS60150DRVR	IC1	5V 140ma switched capacitor voltage converter WSON-6 package
1	Pico_T61_Keyboard	PWB	Printed Wiring Board manufactured from Eagle file “Pico_T61_Keyboard.brd” or Gerber file “Pico_T61_Keyboard.zip”

The total parts cost is about $70 if using OSHPark or about $55 if using JLCPCB (with slow shipping).

A lithium battery, charging circuit, USB Mini connector, and 3 pin JST cable will cost about $40.

OSH Park sends you 3 boards and JLCPCB sends you 5 boards so build this project with friends to spread out the cost. You will save a few bucks if you order the Pico W without header pins. You can install your own headers or solder the Pico directly to the board. I installed single header pins on the grounds for a robust connection but for all the GPIO pins, I soldered 30 gauge wires so I could reroute signals during debug. I ended up not having any problems and if I build another board, I’ll probably just solder the Pico directly to the board for a low profile.

The board has surface mount pads for the keyboard connector that are extra-long to accept an alternate connector, Digikey part number WM6787CT-ND. This was done just in case the AA01B-S040 is not available (I think there is only one source). I’ve used the WM6787CT-ND connector on previous T61 projects and it works fine. It lacks the plastic guides on each end so you must visually align the pins when mating with the keyboard connector. Even the AA01B-S040 needs help with alignment if the plastic guides on the keyboard connector have broken off (see below). The guides break off if you’re way off when pushing it into the board connector (my bad).



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

Lenovo ThinkPad T61 FPC Connector	Pico GP I/O number	Notes	T61 Schematic Signal Name
1	9	Added to key matrix	HOTKEY
2	12		DRV<4>
3	20		SENSE<5>
4	19		DRV<5>
5	11		SENSE<0>
6	13		DRV<8>
7	21		SENSE<3>
8	18		DRV<6>
9	10		SENSE<2>
10	14		DRV<3>
11	9		SENSE<4>
12	15		DRV<7>
13	22		SENSE<1>
14	17		DRV<2>
15	8		SENSE<6>
16	16		DRV<10>
17	7		SENSE<7>
18	26		DRV<1>
19		PWR SW – J4 Pin 2	PWR SW
20	6		DRV<9>
21	No connect		NC
22	27		DRV<0>
23	No connect		NC
24	5		DRV<11>
25	No connect		KBDID0
26	4		DRV<14>
27	No connect		KBDID1
28	28		DRV<12>
29	No connect		KBDID2
30	3		DRV<15>
31	Pico GND		KBDID RTN
32	2		DRV<13>
33	No connect		NC
34	Pico GND		KBDID RTN
35	No connect		NC
36	6	Added to key matrix	HOTKEY RTN
37	1	Level Translated	TP_DATA
38		5 volts from TPS60150	TP_5V
39	0	Level Translated	TP_CLK
40		TLV810 Reset Generator	TP_RESET

The Pico inputs are the Sense<0> thru <7> columns across the top which must be programmed with pullup resistors. The Pico outputs are the Drive<0> thru <15> rows on the side and are either driven low or floated. Floating a pin is done by making it an input.

Matrix for the Lenovo ThinkPad T61 Keyboard

Pico GP I/O Number	**Sense<0> GP 11**	**Sense<1> GP 22**	**Sense<2> GP 10**	**Sense<3> GP 21**	**Sense<4> GP 9**	**Sense<5> GP 20**	**Sense<6> GP 8**	**Sense<7> GP 7**
Drive<0> GP 27	Back-Tick	1	Q	Tab	A	Esc	Z	
Drive<1> GP 26	F1	2	W	Caps-Lock	S		X	
Drive<2> GP 17	F2	3	E	F3	D	F4	C	
Drive<3> GP 14	5	4	R	T	F	G	V	B
Drive<4> GP 12	6	7	U	Y	J	H	M	N
Drive<5> GP 19	Equal	8	I	Right-Brace	K	F6	Comma	
Drive<6> GP 18	F8	9	O	F7	L		Period	
Drive<7> GP 15	Minus	0	P	Left-Brace	Semi-colon	Quote		Forward-Slash
Drive<8> GP 13	F9	F10		Back-Space	Back-Slash	F5	Enter	Space
Drive<9> GP 6	Insert	F12			**Fn added here**			Arrow-Right
Drive<10> GP 16	Delete	F11	Volume-Up	Volume-Down	Mute	Think-Vantage		Arrow-Down
Drive<11> GP 5	Page-Up	Page-Down	**GUI**		Menu		Page-Left	Page-Right
Drive<12> GP 28	Home	End				Arrow-Up	Pause	Arrow-Left
Drive<13> GP 2		Print-Screen	Scroll-Lock			**Alt-L**		**Alt-R**
Drive<14> GP 4				**Shift-L**			**Shift-R**	
Drive<15> GP 3	**Cntrl-L**						**Cntrl_R**	

The keyboard uses 24 Pico GP I/O's and the TrackPoint uses 2 GP I/O's. There are no spare GP I/O's for the Fn "Hotkey" so it was wired into the matrix at the location shown above.

Software and Testing:

KMK was used to verify the Pico keyboard connections but not the TrackPoint. The KMK Getting Started page describes how to install CircuitPython, the KMK folder, and main.py on the Pico. Several KMK websites and videos describe how to modify main.py for a keypad. I took their code and expanded the matrix for a T61 keyboard. I'm new to KMK and a beginner with Python yet the T61 main.py code at my repo is fully functional. The code switches to a second layer from the main layer when the Fn key is held down. The code could also be modified with a third layer that is activated when Num Lock is selected so the number pad is usable, (I haven't figured this out yet). I've included USB main.py code below but the formatting is easier to read at my repo.

```
import board
from kmk.kmk_keyboard import KMKKeyboard
from kmk.keys import KC
from kmk.scanners import DiodeOrientation
from kmk.extensions.media_keys import MediaKeys
from kmk.modules.layers import Layers
keyboard = KMKKeyboard()
keyboard.col_pins = (board.GP11, board.GP22, board.GP10, board.GP21, board.GP9, board.GP20, board.GP8, board.GP7)
keyboard.row_pins = (board.GP27, board.GP26, board.GP17, board.GP14, board.GP12, board.GP19, board.GP18, board.GP15,
                     board.GP13, board.GP6, board.GP16, board.GP5, board.GP28, board.GP2, board.GP4, board.GP3)
keyboard.diode_orientation = DiodeOrientation.COL2ROW
keyboard.modules.append(Layers())
keyboard.extensions.append(MediaKeys())
FN = KC.MO(1)
keyboard.keymap = [
   [#layer 0: Base Layer
            KC.GRAVE, KC.N1, KC.Q, KC.TAB, KC.A, KC.ESC, KC.Z, KC.NO,
            KC.F1, KC.N2, KC.W, KC.CAPS, KC.S, KC.NO, KC.X, KC.NO,
            KC.F2, KC.N3, KC.E, KC.F3, KC.D, KC.F4, KC.C, KC.NO,
            KC.N5, KC.N4, KC.R, KC.T, KC.F, KC.G, KC.V, KC.B,
            KC.N6, KC.N7, KC.U, KC.Y, KC.J, KC.H, KC.M, KC.N,
            KC.EQUAL, KC.N8, KC.I, KC.RBRC, KC.K, KC.F6, KC.COMMA, KC.NO,
            KC.F8, KC.N9, KC.O, KC.F7, KC.L, KC.NO, KC.DOT, KC.NO,
            KC.MINUS, KC.N0, KC.P, KC.LBRC, KC.SCOLON, KC.QUOTE, KC.NO, KC.SLASH,
            KC.F9, KC.F10, KC.NO, KC.BSPACE, KC.BSLASH, KC.F5, KC.ENTER, KC.SPACE,
            KC.INSERT, KC.F12, KC.NO, KC.NO, FN, KC.NO, KC.NO, KC.RIGHT,
            KC.DELETE, KC.F11, KC.VOLU, KC.VOLD, KC.MUTE, KC.NO, KC.NO, KC.DOWN,
            KC.PGUP, KC.PGDOWN, KC.LGUI, KC.NO, KC.NO, KC.NO, KC.NO, KC.NO,
            KC.HOME, KC.END, KC.NO, KC.NO, KC.NO, KC.UP, KC.PAUSE, KC.LEFT,
            KC.NO, KC.PSCREEN, KC.SLCK, KC.NO, KC.NO, KC.LALT, KC.NO, KC.RALT,
            KC.NO, KC.NO, KC.NO, KC.LSHIFT, KC.NO, KC.NO, KC.RSHIFT, KC.NO,
            KC.LCTRL, KC.NO, KC.NO, KC.NO, KC.NO, KC.NO, KC.RCTRL, KC.NO,
           ],
```

```
[#layer 1: Fn Media Layer
            KC.GRAVE, KC.N1, KC.Q, KC.TAB, KC.A, KC.ESC, KC.Z, KC.NO,
            KC.F1, KC.N2, KC.W, KC.CAPS, KC.S, KC.NO, KC.X, KC.NO,
            KC.F2, KC.N3, KC.E, KC.F3, KC.D, KC.F4, KC.C, KC.NO,
            KC.N5, KC.N4, KC.R, KC.T, KC.F, KC.G, KC.V, KC.B,
            KC.N6, KC.N7, KC.U, KC.Y, KC.J, KC.H, KC.M, KC.N,
            KC.EQUAL, KC.N8, KC.I, KC.RBRC, KC.K, KC.F6, KC.COMMA, KC.NO,
            KC.F8, KC.N9, KC.O, KC.F7, KC.L, KC.NO, KC.DOT, KC.NO,
            KC.MINUS, KC.N0, KC.P, KC.LBRC, KC.SCOLON, KC.QUOTE, KC.NO, KC.SLASH,
            KC.F9, KC.F10, KC.NO, KC.BSPACE, KC.BSLASH, KC.F5, KC.ENTER, KC.SPACE,
            KC.INSERT, KC.F12, KC.NO, KC.NO, FN, KC.NO, KC.NO, KC.MNXT,
            KC.DELETE, KC.F11, KC.VOLU, KC.VOLD, KC.MUTE, KC.NO, KC.NO, KC.MPLY,
            KC.PGUP, KC.PGDOWN, KC.LGUI, KC.NO, KC.NO, KC.NO, KC.NO, KC.NO,
            KC.BRIU, KC.BRID, KC.NO, KC.NO, KC.NO, KC.MSTP, KC.PAUSE, KC.MPRV,
            KC.NO, KC.PSCREEN, KC.NLCK, KC.NO, KC.NO, KC.LALT, KC.NO, KC.RALT,
            KC.NO, KC.NO, KC.NO, KC.LSHIFT, KC.NO, KC.NO, KC.RSHIFT, KC.NO,
            KC.LCTRL, KC.NO, KC.NO, KC.NO, KC.NO, KC.NO, KC.RCTRL, KC.NO,
           ],
]
if __name__ == '__main__':
    keyboard.go()
```

KMK does not appear to support a PS/2 mouse so the TrackPoint was tested separately. First I used a multimeter to verify the charge pump was providing 5 volts to the TrackPoint. I wrote Arduino C code to bit-bang the PS/2 clock and data signals to the TrackPoint. The “Pico_TP_T61.ino” code at my repo translates the TrackPoint PS/2 data to USB to move the cursor and give left and right button pushes. I’ve also used this code to allow the Pico to control a standard PS/2 touchpad, documented here.

Next Steps: Make the keyboard and TrackPoint work at the same time over USB. This might be possible with KMK but I doubt it. I believe QMK can do it but that’s a steep learning curve. The other approach is to write new Arduino C code for the keyboard and combine it with my TrackPoint USB code. The really big next step is to get Bluetooth working for the keyboard and TrackPoint. Stay tuned…..