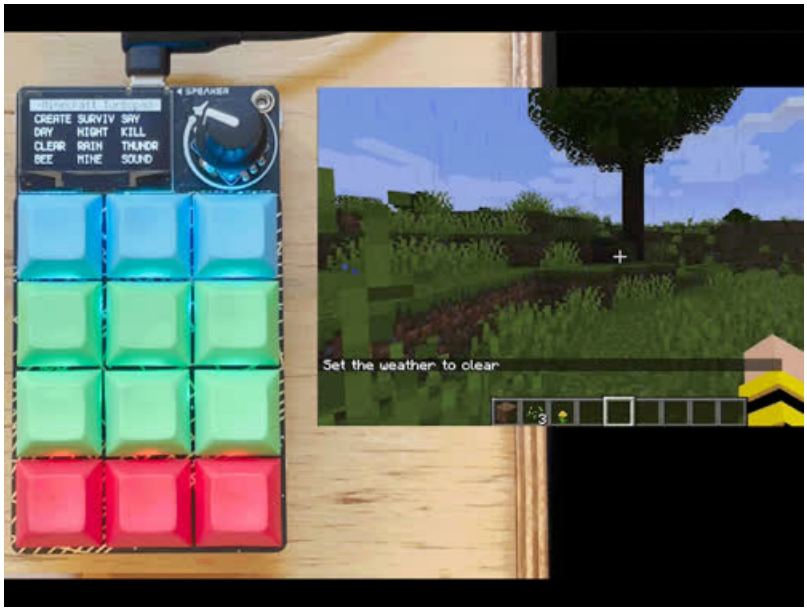
adafruit learning system

Minecraft Turbopad

Created by John Park



Last updated on 2021-08-03 06:56:35 PM EDT

Guide Contents

Overview

Use the Macropad to send any command to Minecraft with just one button press. Program twelve custom keys that can run commands or key combos.

CircuitPython makes it simple to customize your Minecraft Turbopad to run any command you like, even initiate latching mouse and key combos, such as auto strip-mining mode.

Parts

Your browser does not support the video tag.

Adafruit MacroPad RP2040 Starter Kit - 3x4 Keys + Encoder + OLED

Strap yourself in, we're launching in T-minus 10 seconds...Destination? A new Class M planet called MACROPAD! M here stands for Microcontroller because this 3x4 keyboard...

Out of Stock

Out of Stock

-or-

Adafruit MACROPAD RP2040 Bare Bones - 3x4 Keys + Encoder + OLED

Strap yourself in, we're launching in T-minus 10 seconds...Destination? A new Class M planet called MACROPAD! M here, stands for Microcontroller because this 3x4 keyboard...

Out of Stock

Out of Stock

Adafruit MacroPad RP2040 Enclosure + Hardware Add-on Pack

Dress up your Adafruit Macropad with PaintYourDragon's fabulous decorative silkscreen enclosure and hardware kit. You get the two custom PCBs that are cut to act as a protective...

$4.95

In Stock

Add to Cart

Kailh Mechanical Key Switches - Linear Red - 12 Pack

For crafting your very own custom keyboard, these Kailh Red Linear mechanical key switches are deeee-luxe! With smooth actuation and Cherry MX compatibility,...

$7.95

In Stock

Add to Cart

Clear Keycaps for MX Compatible Switches - 12-pack

Here is a 12 pack of Clear DSA keycaps for your next mechanical keyboard or

$6.95

In Stock

Add to Cart

Pink and Purple Woven USB A to USB C Cable - 1 meter long

This cable is not only super-fashionable, with a woven pink and purple Blinka-like pattern, it's also made for USB C for our modernized breakout boards, Feathers, and...

$2.95

In Stock

Add to Cart

CircuitPython

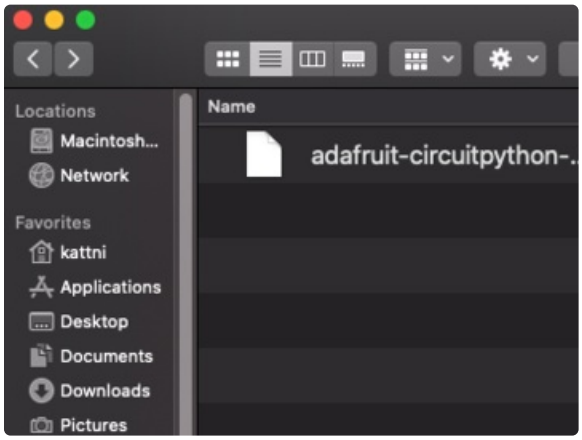
CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the **CIRCUITPY** drive to iterate.

CircuitPython Quickstart

Follow this step-by-step to quickly get CircuitPython running on your board.

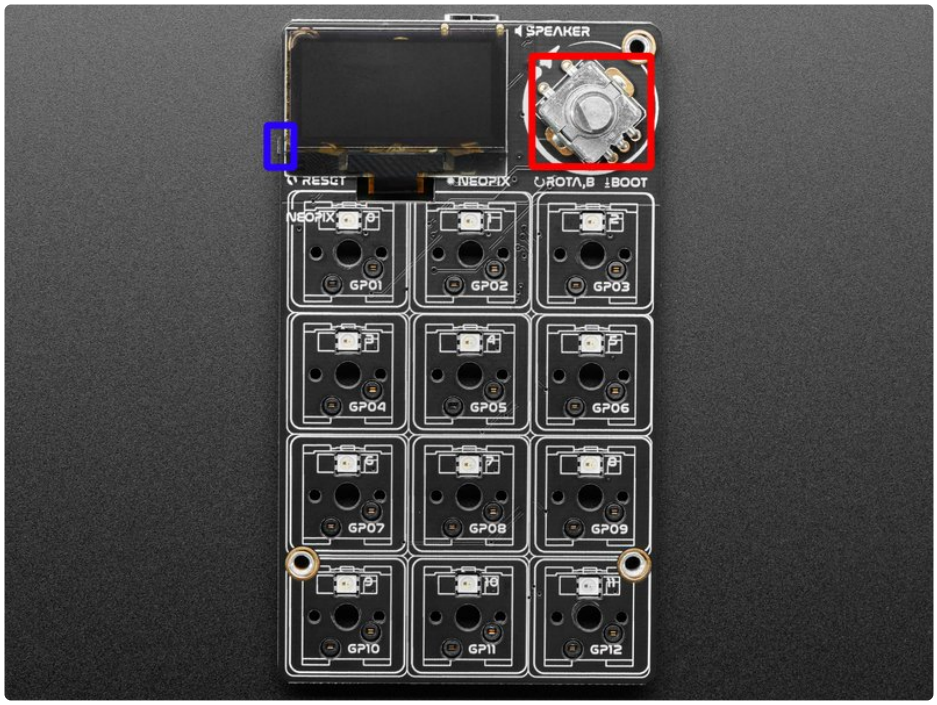
https://adafru.it/TB9

https://adafru.it/TB9



Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.



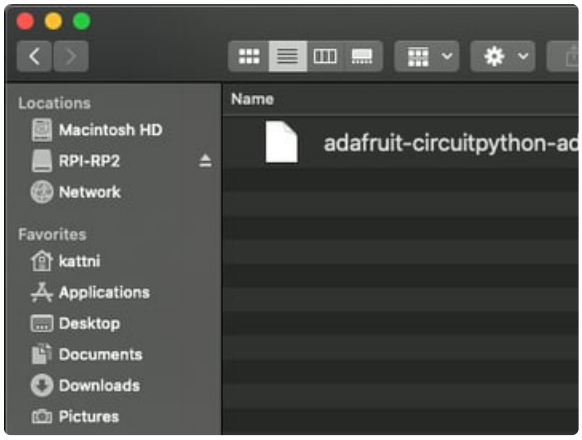
The BOOT button is the button switch in the rotary encoder! To engage the BOOT button, simply press down on the rotary encoder.

To enter the bootloader, hold down the **BOOT/BOOTSEL button** (highlighted in red above), and while continuing to hold it (don't let go!), press and release the **reset button** (highlighted in blue above). **Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!**

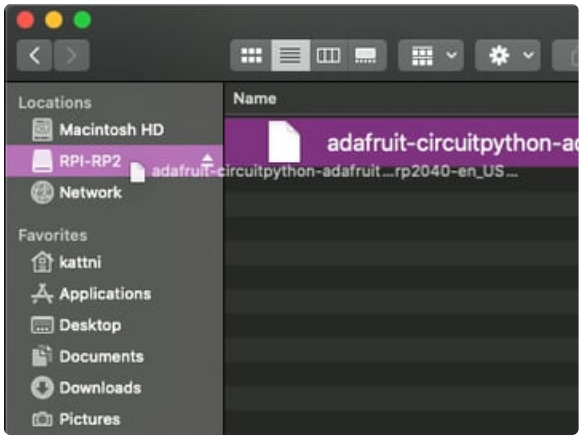
If the drive does not appear, release all the buttons, and then repeat the process above.

You can also start with your board unplugged from USB, press and hold the BOOTSEL button (highlighted in red above), continue to hold it while plugging it into USB, and wait for the drive to appear before releasing the button.

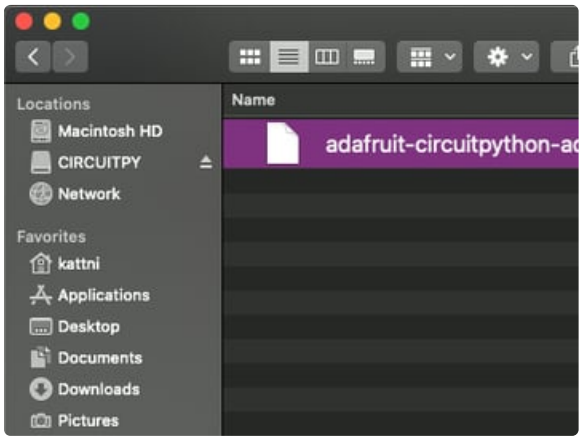
A lot of people end up using charge-only USB cables and it is very frustrating! **Make sure you have a USB cable you know is good for data sync.**



You will see a new disk drive appear called **RPI-RP2**.



Drag the **adafruit_circuitpython_etc.uf2** file to **RPI-RP2.**



The **RPI-RP2** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

That's it, you're done! :)

Safe Mode

You want to edit your **code.py** or modify the files on your **CIRCUITPY** drive, but find that you can't. Perhaps your board has gotten into a state where **CIRCUITPY** is read-only. You may have turned off the **CIRCUITPY** drive altogether. Whatever the reason, safe mode can help.

Safe mode in CircuitPython does not run any user code on startup, and disables auto-reload. This means a few things. First, safe mode *bypasses any code in* **boot.py** (where you can set **CIRCUITPY** read-only or turn it off completely). Second, *it does not run the code in* **code.py**. And finally, *it does not automatically soft-reload when data is written to the* **CIRCUITPY** *drive*.

Therefore, whatever you may have done to put your board in a non-interactive state, safe mode gives you the opportunity to correct it without losing all of the data on the **CIRCUITPY** drive.

Entering Safe Mode in CircuitPython 6.x

This section explains entering safe mode on CircuitPython 6.x.

To enter safe mode when using CircuitPython 6.x, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 700ms. On some boards, the onboard status LED (highlighted in green above) will turn solid yellow during this time. If you press reset during that 700ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

Entering Safe Mode in CircuitPython 7.x

This section explains entering safe mode on CircuitPython 7.x.

To enter safe mode when using CircuitPython 7.x, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 1000ms. On some boards, the onboard status LED (highlighted in green above) will blink yellow during that time. If you press reset during that 1000ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

In Safe Mode

Once you've entered safe mode successfully in CircuitPython 6.x, the LED will pulse yellow.

If you successfully enter safe mode on CircuitPython 7.x, the LED will intermittently blink yellow three times.

If you connect to the serial console, you'll find the following message.

```
Auto-reload is off.
Running in safe mode! Not running saved code.

CircuitPython is in safe mode because you pressed the reset button during boot. Press again to
exit safe mode.

Press any key to enter the REPL. Use CTRL-D to reload.
```

You can now edit the contents of the **CIRCUITPY** drive. Remember, *your code will not run until you press the reset button, or unplug and plug in your board, to get out of safe mode.*

Flash Resetting UF2

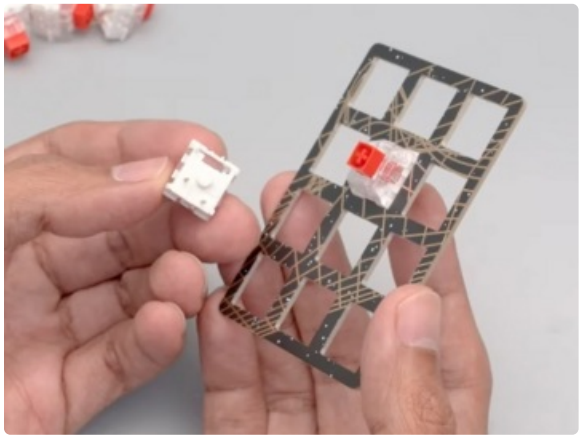
If your board ever gets into a really *weird* state and doesn't even show up as a disk drive when installing CircuitPython, try loading this 'nuke' UF2 which will do a 'deep clean' on your Flash Memory. **You will lose all the files on the board**, but at least you'll be able to revive it! After loading this UF2, follow the steps above to re-install CircuitPython.

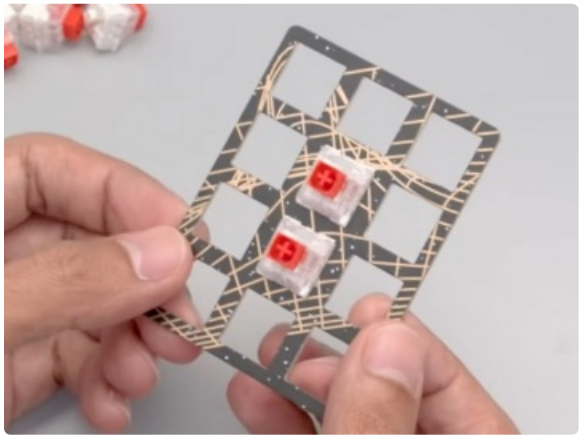
https://adafru.it/RLE

https://adafru.it/RLE

Macropad Assembly

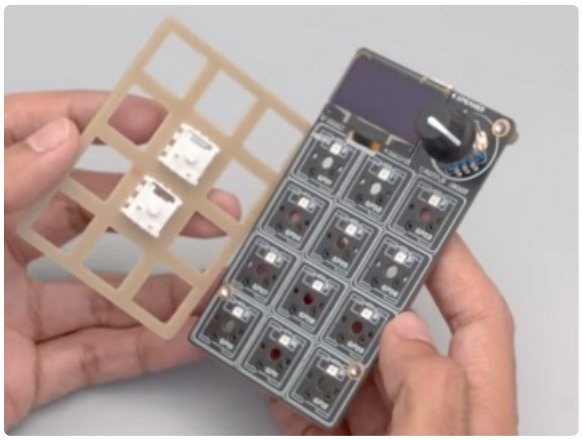
The Macropad features hot-swap sockets for the switches -- gone are the days of having to *commit* to one type of switch and solder it down! Now, you can plug in your Cherry MX red keyswitches, use them for a while, get bored, decide its time to test out some lubed, filmed, re-sprung Invyr Holy Pandas, and swap them just like that!

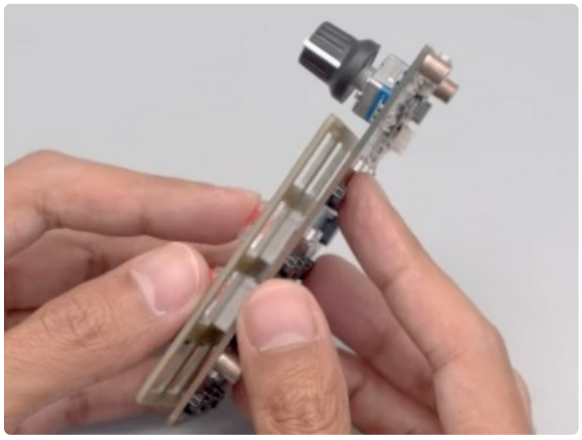




Switches into Plate

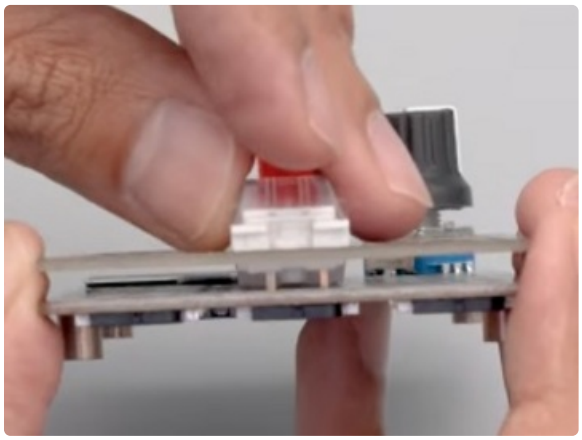
First, insert a couple pf keyswitches through the keyswitch plate. The plate mechanically connects the switches to each other, which lends some nice lateral stability to the keys.

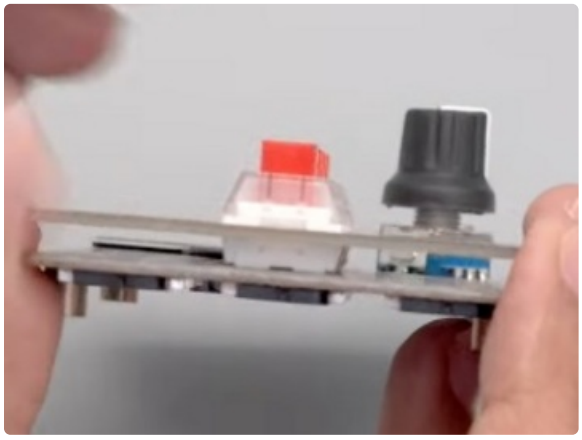




Connect to Board

Carefully press the two switches into the switch sockets, being very careful to align the legs so none bend!



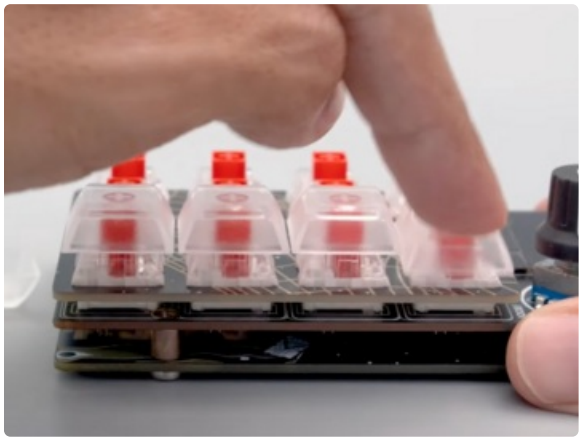


Add Switches

Continue adding switches, being mindful of their orientation.

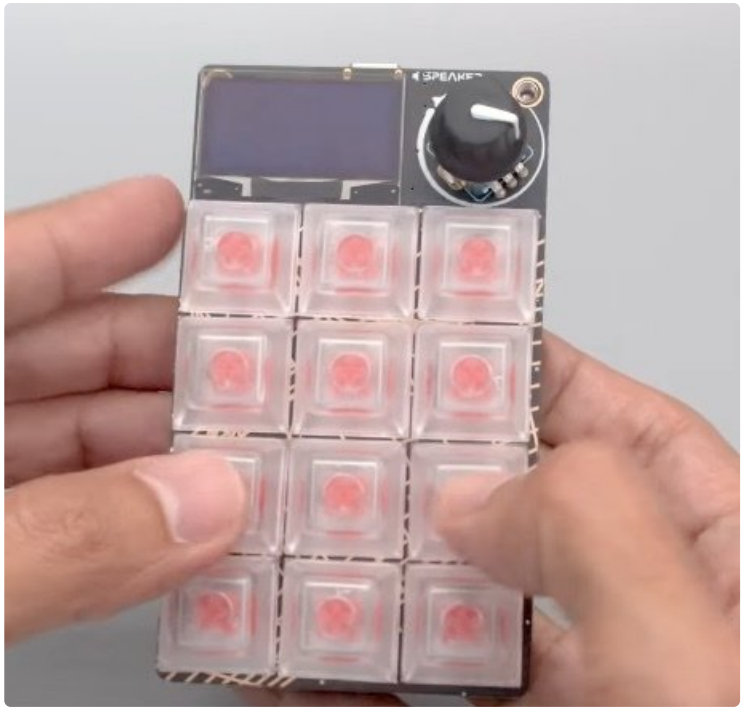
Backplate

You can add the optional backplate using four M3 x 6mm screws.



Keycaps

Now, you can add your keycaps! simply press them onto the keyswitch stems until they are fully seated.



Code and Use the Minecraft Turbopad

Text Editor

Adafruit recommends using the **Mu** editor for editing your CircuitPython code. You can get more info in this guide (https://adafru.it/ANO).

Alternatively, you can use any text editor that saves simple text files.

Download the Project Bundle

Your project will use a specific set of CircuitPython libraries and the **code.py** file, along with a folder full of key configuration files. To get everything you need, click on the **Download Project Bundle** link below, and uncompress the .zip file.

Drag the contents of the uncompressed bundle directory onto your MACROPAD board's **CIRCUITPY** drive, replacing any existing files or directories with the same names, and adding any new ones that are necessary.

```
# SPDX-FileCopyrightText: 2021 John Park for Adafruit Industries
# SPDX-License-Identifier: MIT
# Minecraft Turbopad for Adafruit Macropad RP2040
import time
import displayio
import terminalio
from adafruit_display_text import bitmap_label as label
from adafruit_displayio_layout.layouts.grid_layout import GridLayout
from adafruit_macropad import MacroPad

macropad = MacroPad()

# --- Variable setup for action types
KEEB = 0
MEDIA = 1
MOUSE = 2
KBMOUSE = 3
COMMAND = 4

KEY_MOMENT = 0  # momentary press/release keys
KEY_HOLD = 1  # toggle keys

# --- LED colors
GREEN = 0x00ff00
RED = 0xff0000
MAGENTA = 0xff0033
YELLOW = 0xffdd00
AQUA = 0x00ffff
LATCH_COLOR = YELLOW
# --- Key mappings
```

```
# (<key>): (<color>, <action type>, <key hold>, <keycodes>, <"text">, <enter>)
keymap = {
    (0): (AQUA, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
          "gamemode creative", [macropad.Keycode.ENTER]),
    (1): (AQUA, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
          "gamemode survival", [macropad.Keycode.ENTER]),
    (2): (AQUA, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
          "say Dinner Time!", [macropad.Keycode.ENTER]),

    (3): (AQUA, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
          "time set day", [macropad.Keycode.ENTER]),
    (4): (AQUA, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
          "time set night", [macropad.Keycode.ENTER]),
    (5): (AQUA, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
          "kill", [macropad.Keycode.ENTER]),

    (6): (GREEN, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
          "weather clear", [macropad.Keycode.ENTER]),
    (7): (GREEN, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
          "weather rain", [macropad.Keycode.ENTER]),
    (8): (GREEN, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
          "weather thunder", [macropad.Keycode.ENTER]),

    (9): (RED, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
          "summon minecraft:bee", [macropad.Keycode.ENTER]),
    (10): (RED, KBMOUSE, KEY_HOLD,  [macropad.Keycode.W], [macropad.Mouse.LEFT_BUTTON]),
    (11): (RED, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
           "playsound minecraft:block.bell.use ambient @a ~ ~ ~", [macropad.Keycode.ENTER]),
}


latched = [False] * 12  # list of the latched states, all off to start

last_knob_pos = macropad.encoder  # store knob position state

# --- Pixel setup --- #
macropad.pixels.brightness = 0.1
for i in range(12):
    macropad.pixels[i] = (keymap[i][0])


main_group = displayio.Group()
macropad.display.show(main_group)
title = label.Label(
    y=4,
    font=terminalio.FONT,
    color=0x0,
    text=" -Minecraft Turbopad- ",
    background_color=0xFFFFFF,
)
layout = GridLayout(x=0, y=13, width=128, height=54, grid_size=(3, 4), cell_padding=5)
label_text = [
    "CREATE", "SURVIV", "SAY",
    "DAY", "NIGHT", "KILL",
    "CLEAR", "RAIN", "THUNDR",
    "BEE", "MINE", "SOUND",
]
```

```
labels = []
for j in range(12):
    labels.append(label.Label(terminalio.FONT, text=label_text[j]))

for index in range(12):
    x = index % 3
    y = index // 3
    layout.add_content(labels[index], grid_position=(x, y), cell_size=(1, 1))

main_group.append(title)
main_group.append(layout)

while True:
    key_event = macropad.keys.events.get()  # check for key press or release
    if key_event:
        if key_event.pressed:
            key = key_event.key_number
            labels[key].color = 0x0
            labels[key].background_color = 0xffffff

            if keymap[key][1] == KEEB:
                if keymap[key][2] == KEY_HOLD:
                    macropad.keyboard.press(*keymap[key][3])  # * expands the variable to list
                else:
                    macropad.keyboard.send(*keymap[key][3])

            elif keymap[key][1] == MOUSE:
                macropad.mouse.click(*keymap[key][3])

            elif keymap[key][1] == KBMOUSE:
                if keymap[key][2] == KEY_HOLD:
                    if latched[key] is False:
                        macropad.keyboard.press(*keymap[key][3])
                        time.sleep(0.01)
                        macropad.mouse.press(*keymap[key][4])
                        latched[key] = True
                    else:
                        macropad.keyboard.release(*keymap[key][3])
                        time.sleep(0.01)
                        macropad.mouse.release_all()
                        latched[key] = False

            elif keymap[key][1] == COMMAND:
                macropad.keyboard.send(*keymap[key][3])
                time.sleep(0.1)
                macropad.keyboard_layout.write(keymap[key][4])
                time.sleep(0.1)
                macropad.keyboard.send(*keymap[key][5])
                time.sleep(0.1)
            macropad.pixels[key] = LATCH_COLOR

        if key_event.released:
            key = key_event.key_number
            if keymap[key][1] == KEEB:
                if keymap[key][2] == KEY_HOLD:
                    macropad.keyboard.release(*keymap[key][3])
```

```
            if latched[key] is False:
                macropad.pixels[key] = (keymap[key][0])
                labels[key].color = 0xffffff
                labels[key].background_color = 0x0

    current_knob_position = macropad.encoder

    if macropad.encoder > last_knob_pos:
        macropad.mouse.move(wheel=-1)
        last_knob_pos = current_knob_position

    if macropad.encoder < last_knob_pos:
        macropad.mouse.move(wheel=+1)
        last_knob_pos = current_knob_position
```

Use It

The Minecraft Turbopad acts just like a regular USB keyboard, so you don't need to do anything tricky to enable it -- in fact, just fire up Minecraft and start using it once you're playing! Press a key and the command will run, just like that!

Here's a video of it in action:

Customize the Commands

You can switch out the commands to any that you prefer! Here's a pretty comprehensive list for reference (https://adafru.it/Uc5).

The `keymap` dictionary contains all of the commands, per key. For example:

```
(0): (AQUA, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
      "gamemode creative", [macropad.Keycode.ENTER])
```

That is the first entry, so it it triggered by the first (upper left) key on the Macropad. When pressed it will type out `/gamemode creative`. If you wanted to change this to switch you into adventure mode, the line would look like this:

```
(0): (AQUA, COMMAND, KEY_MOMENT, [macropad.Keycode.FORWARD_SLASH],
      "gamemode adventure", [macropad.Keycode.ENTER])
```

Display Labels

When you customize your key commands, you'll want to update the display labels to match. This code is the list you'll need to change, simply swap in different text -- just keep it short and sweet so it fits on screen!

```
label_text = [
    "CREATE", "SURVIV", "SAY",
    "DAY", "NIGHT", "KILL",
    "CLEAR", "RAIN", "THUNDR",
    "BEE", "MINE", "SOUND",
]
```