Desk Calculator with CircuitPython

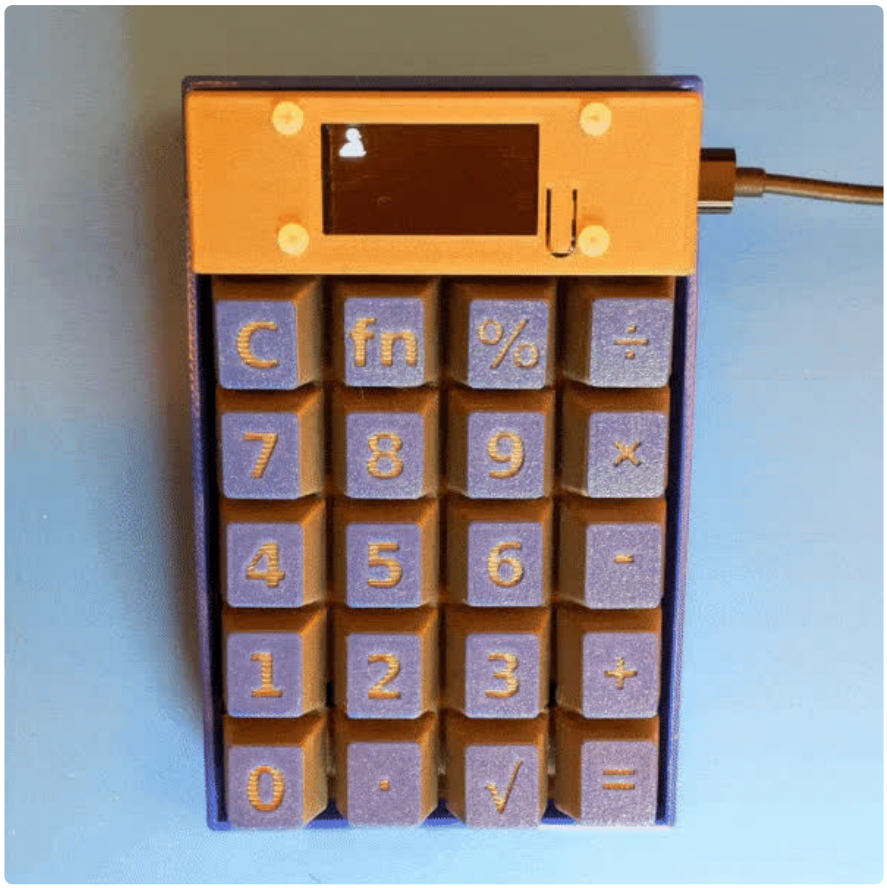
Created by Jeff Epler

https://learn.adafruit.com/desk-calculator-with-circuitpython

Last updated on 2024-06-03 03:26:12 PM EDT

Table of Contents

Overview



Consider the humble calculator. Now just an app that you always have with you on your telephone, an electronic calculator used to be a technological marvel (https://adafru.it/TFA).

In this project, you'll build your own calculator with an Adafruit Feather RP2040, 128x64 OLED display, and 20 keys. Optional 3D printed parts like an enclosure and even the keycaps themselves make for a sharp presentation. Make it your own by customizing the code, or use the hardware for your own MacroPad-inspired project.

Custom Printed Circuit Board

This project needs a custom PCB. You can pick up the KiCad (https://adafru.it/TFB) (design) files or the Gerber (production) files below and upload them to the PCB house of your choice, or you can use this link to order them from OSH Park (https://adafru.it/TFC). You are free to modify the design files and use them in your own designs. (In copyright terms, the design files are licensed as CC0 (https://adafru.it/DSc))

OSH Park's standard turnaround is 9 to 12 days from order to shipment, so plan accordingly.

The board files were designed in KiCad (https://adafru.it/TFB), open source software that is free to download & use.

Gerber Files for ordering PCB

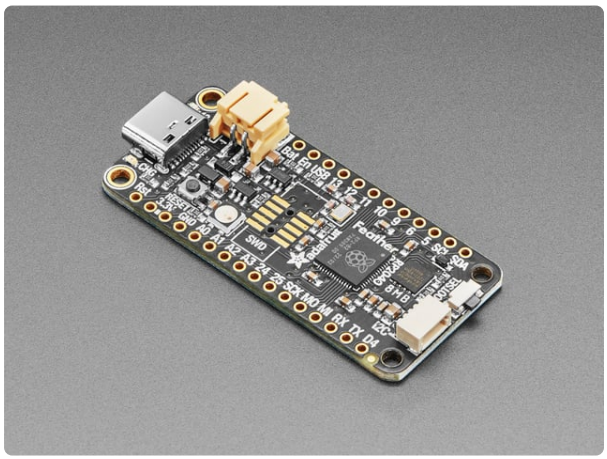
https://adafru.it/TFD

PCB Design Files for KiCad

https://adafru.it/TFE

Required Parts

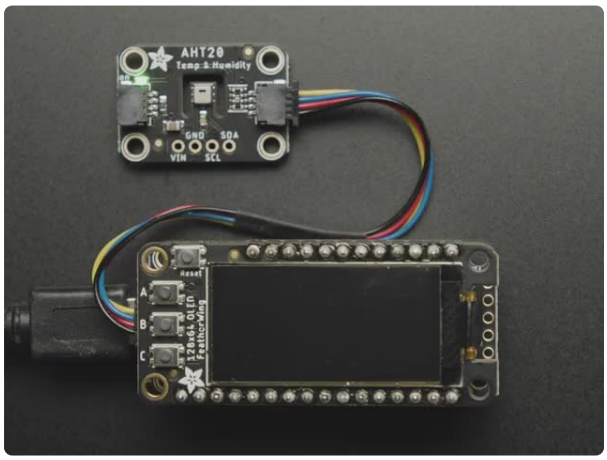
Along with the custom PCB, you'll need basic tools, soldering supplies, and the Adafruit parts listed below.



Adafruit Feather RP2040

A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought "this chip is going to be awesome when we give it the Feather...

https://www.adafruit.com/product/4884



Adafruit FeatherWing OLED - 128x64 OLED Add-on For Feather

A Feather board without ambition is a Feather board without FeatherWings! This is the FeatherWing 128x64 OLED: it adds a gorgeous 128x64 monochrome OLED plus 3...

https://www.adafruit.com/product/4650

White Nylon Machine Screw and Stand-off Set – M2.5 Thread

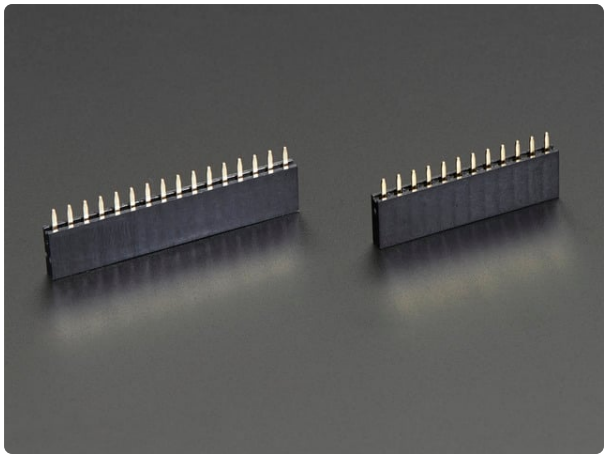
Totaling 420 pieces, this White Nylon M2.5 Screw Set is a must-have smörgåsbord for your workstation. You'll have more than enough...

https://www.adafruit.com/product/3658

Pink and Purple Woven USB A to USB C Cable - 2 meters 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 more.

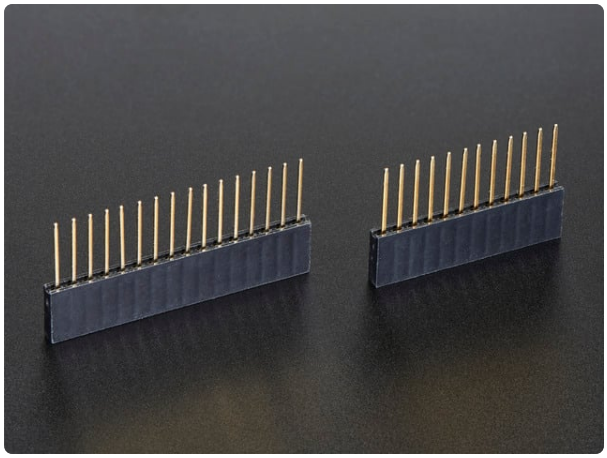
https://www.adafruit.com/product/5044



Header Kit for Feather - 12-pin and 16-pin Female Header Set

These two Female Headers alone are, well, lonely. But pair them with any of our

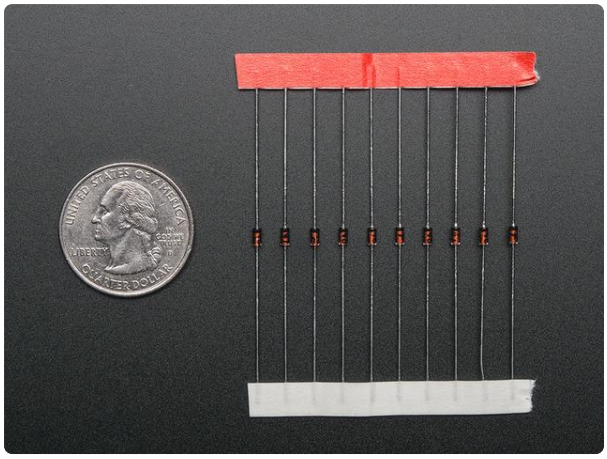
https://www.adafruit.com/product/2886



Stacking Headers for Feather - 12-pin and 16-pin female headers

These two Female Stacking Headers alone are, well, lonely. But pair them with any of our Feather boards and...

https://www.adafruit.com/product/2830

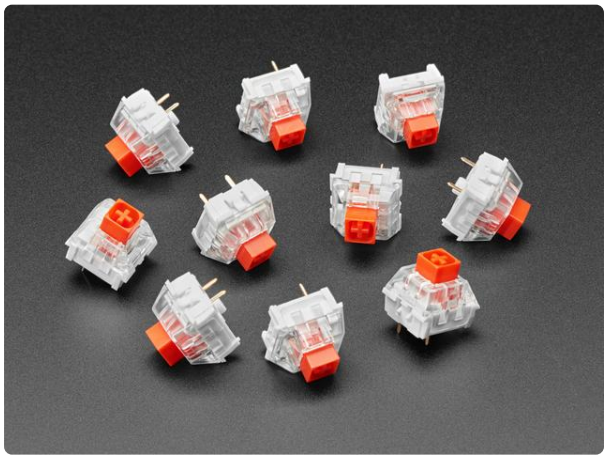


1N4148 Signal Diode - 10 pack

You have some electrons over here, and you want them over there but you don't want the electrons from over there to be able to come over here? That's what a diode is for, these...

https://www.adafruit.com/product/1641

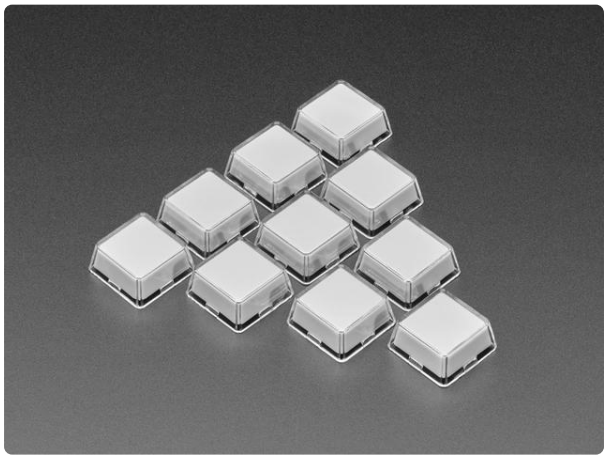
Pick up 2 ten-packs of diodes, you need one for each key!



Kailh Mechanical Key Switches - Linear Red - 10 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,...

https://www.adafruit.com/product/4952



Relegendable Plastic Keycaps for MX Compatible Switches 10 pack

Get ready to customize your keeb with a 10 pack of two-part plastic keycaps for your next mechanical keyboard or

https://www.adafruit.com/product/5039

Little Rubber Bumper Feet - Pack of 4
Keep your electronics from going barefoot, give them little rubber feet! These small sticky bumpers are our favorite accessory for any electronic kit or device. They are sticky, but...
https://www.adafruit.com/product/550

Optional parts

1 x Tapered Heat-Set Inserts for Plastic
Brass, M2.5 x 0.45 mm Thread Size, 3.4 mm Installed Length (Pack of 100)
https://www.mcmaster.com/94180A321/

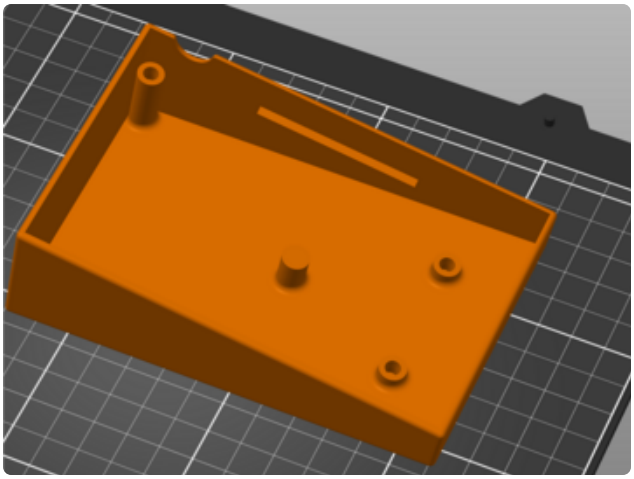
3D Printed Parts

While these parts are all optional, they'll greatly refine the look of your calculator and make it more pleasant to use.

Some of the models were developed with FreeCAD (https://adafru.it/TFI), others with OpenSCAD (https://adafru.it/TFJ). Both packages are open source software packages that are free to download & use.

3D Printed Parts - All Files, All Formats
https://adafru.it/TFK

Base Enclosure



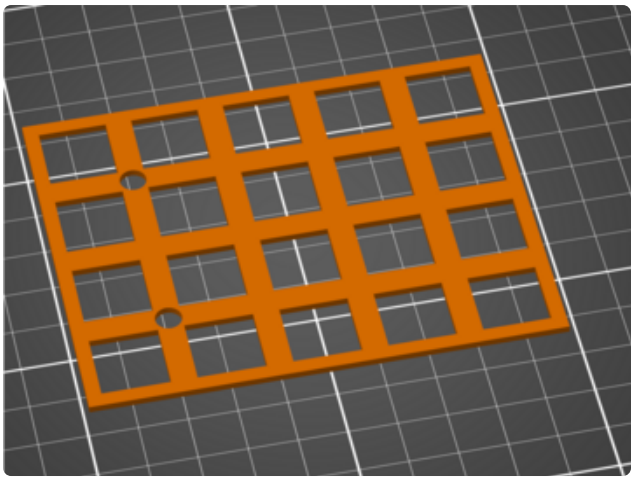
The base enclosure protects your desk or table from being scratched by the solder junctions on the PCB, and prevents you from delivering static shocks to the circuit when holding the calculator in hand. It also holds the keypad at an angle that makes it more comfortable to use. The main alternative to printing the base enclosure is to add adhesive rubber feet directly to the bottom side of the PCB.

You can choose either the "snap-fit" version or the "screw-in" version. In the first case, the PCB is retained by two vee-shaped protrusions from the sides. In the second case, the PCB is retained by M2.5×6 screws using heat-set threaded inserts. (the snap-fit version also includes the holes for the heat-set inserts, but you don't need to use them)

In your slicer, use the "lay flat" operation to place the bottom of the enclosure directly on the print bed. Default slicing settings should suffice. If desired, clean up the shallow-angled top face by lightly sanding after 3D printing. Attach bumper feet to the bottom for anti-skid goodness.

The enclosure was designed in FreeCAD; you can load the **FCStd** file into FreeCAD to make changes to the model. Use the **stl** file for 3D printing.

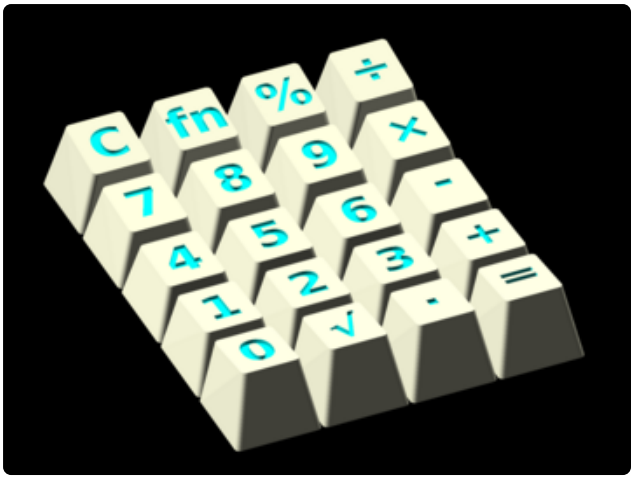
Keyswitch Plate



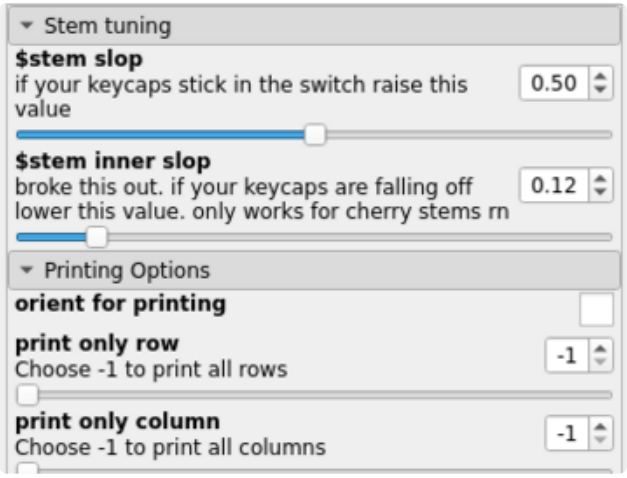
The keyswitch plate helps ensure that the keyswitches are soldered in without any rotation, and provides stability during use; you can get by without it, but it's easy to end up with some of your keys looking wonky. If you print it, remember that you must snap the keyswitches into it **before** soldering the keyswitches to the PCB.

The plate was designed in OpenSCAD; you can load the **scad** file into OpenSCAD to refine the model. Use the **stl** file for 3D printing.

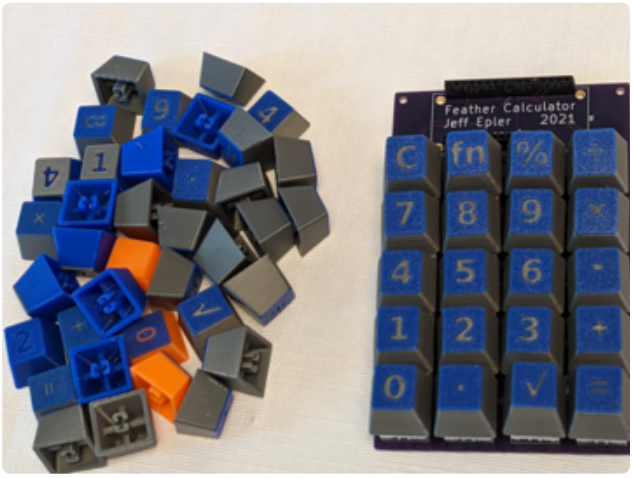
Keycaps



You can print custom keycaps for your calculator. These are based on KeyV2 by @rsheldiii (https://adafru.it/TFL). The values are tuned for the author's Prusa I3 mk3s with an 0.4mm nozzle, PLA filament and 0.2mm "speed" printing profile. Each key takes about 20 minutes and uses about 2g of plastic, so it can be well worth the time to dial in the design for your printer.



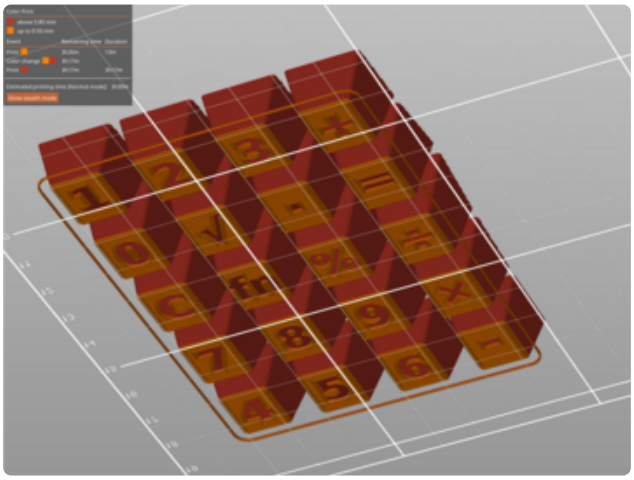
The geometry of the stems must be very precise, so begin by printing a single "central row" switch to check it for fit and function in your keyswitches. If the keyswitch from the provided STL does not fit, use the OpenSCAD Customizer (https://adafru.it/TFM) to increase **$stem_slop** and/or **$stem_inner_slop** until you get a good fit (Start by changing things in 0.5mm increments). If they fall out too readily, decrease **$stem_slop** and/or **$stem_inner_slop** similarly. **$stem_slop** controls the outside of the stem, so increasing it makes the fit against the outside of the "box" of the keycap plunger looser and increasing it makes it tighter. **$stem_inner_slop** controls the inside of the stem, so increasing it leaves a greater gap for the "+" of the keycap plunger and decreasing it leaves a smaller gap.



It can be visually difficult to tell whether the interference is at the inside or the outside; you can modify each number alternately until you get a fit or can tell which one is wrong.

Next, print a whole column of switches and confirm that they all fit; on the author's printer, one set of **slop** values was OK for all rows, but your experience may differ.

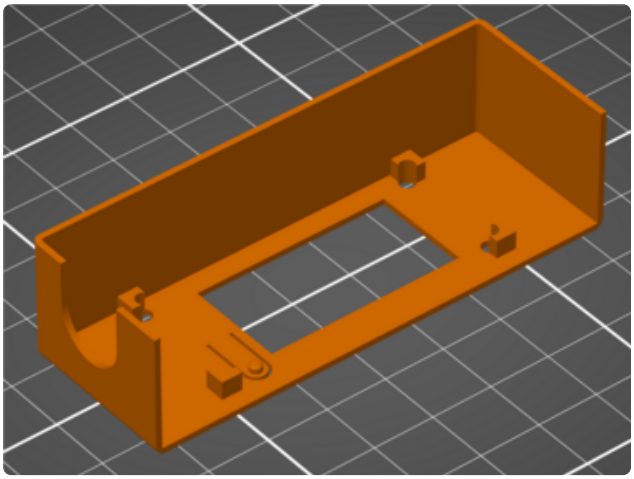
Expect to repeat this process a few times; here is the pile of switches the author discarded as unusable while perfecting the design & fine tuning the **slop** values.



The model should load with the tops of the keys on the build surface. For two-tone keys, configure your slicer to pause at 0.8mm Z-height for a color change. This really makes the legend stand out, and can be done on most 3D printers.

For fit testing, you can print the whole thing in one color, but be sure to choose the color that the stem will ultimately be printed in; the author found that two different spools of PLA required different fine-tuning numbers.

Screen Bezel



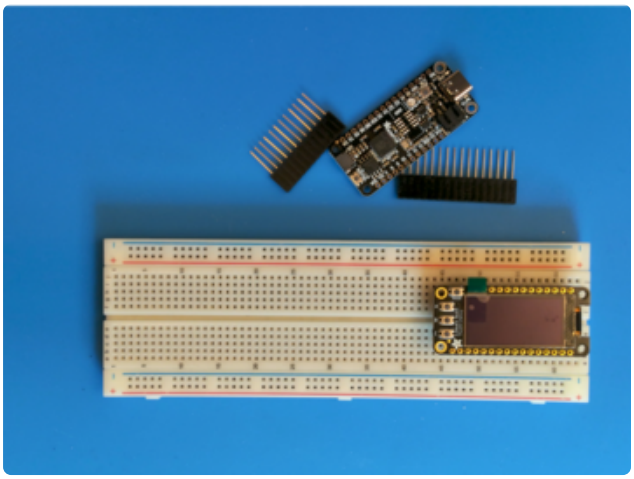
Use your slicer's "lay flat" function to place the top of the bezel directly on the printing surface. Default print settings suffice for this print.

The bezel was designed in FreeCAD; you can load the **FCStd** file into freecad to refine the model. Use the **stl** file for 3D printing.

The little cutout allows the reset switch to be pressed, in the event you need to restart the calculator or access the UF2 bootloader for updates.

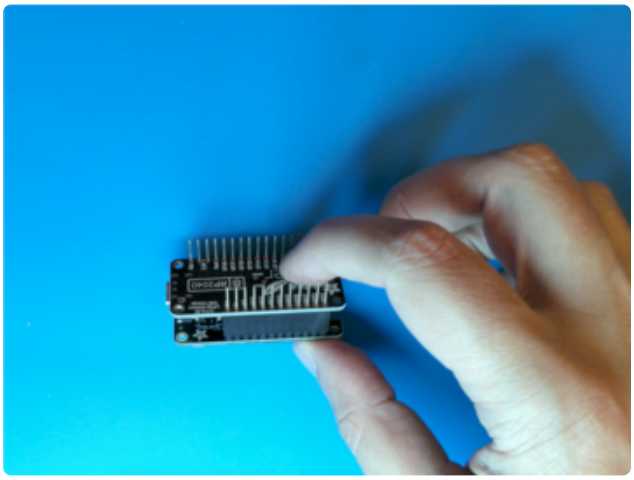
Prepare the Feather & OLED FeatherWing

The stacking style described below is well-matched to the height of the 3D printed keycaps. If you plan to use different keycaps, particularly low-profile keycaps, you should do a test fit of everything to check the height of the keycap vs the height of the screen. If you want a shorter screen height, there are lots of options you can explore with various ways of stacking short headers.



Solder male headers to the OLED

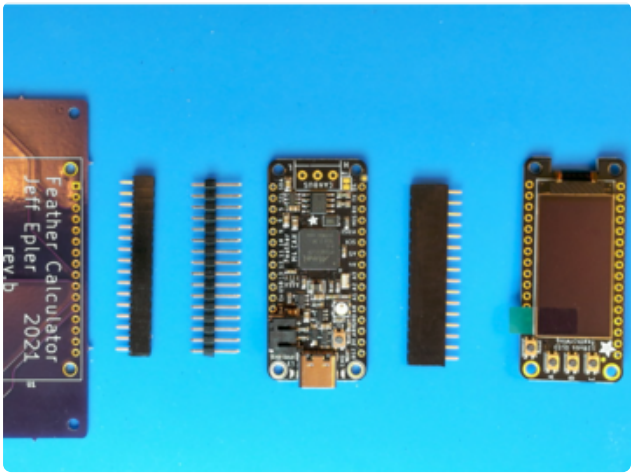
Insert male headers into the OLED from the bottom, then insert it into a solderless breadboard. Solder the male headers from the top side. Double check after soldering just one pad on each header that the headers are straight, not at an angle. Then, finish soldering the rest of the pads.

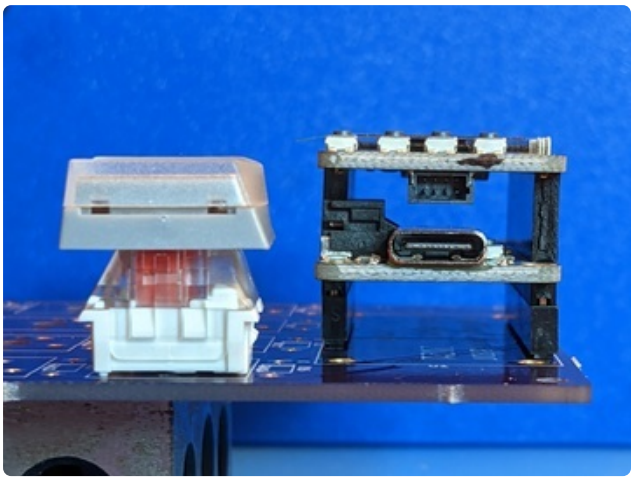


Solder stacking headers to the Feather

Insert the stacking headers into the Feather from the top, then insert the OLED into it. Solder the stacking headers from the bottom side. Double check after soldering just one pad on each header that the headers are straight, not at an angle. Then, finish soldering the rest of the pads.

Alternative Stacking for Shorter Keycaps





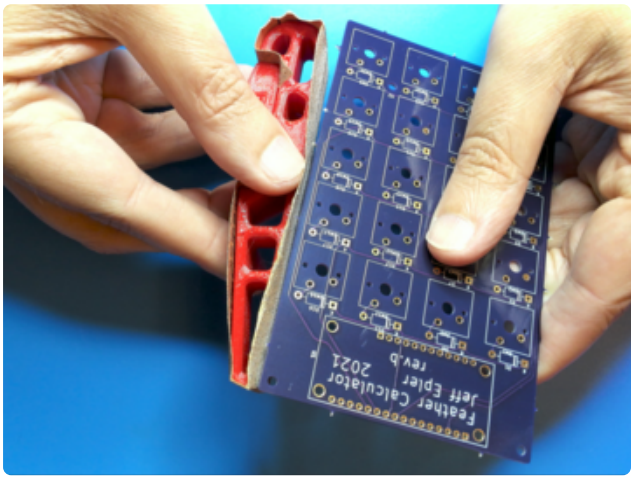
If you're using short keycaps like the relegendable ones, here's one alternative way to stack it up. The short female headers go into the calculator PCB; the male headers' long ends are inserted into the bottom of the Feather microcontroller board and soldered from the top; and (standard) female headers are inserted from the bottom side of that PCB.

To get even shorter, you can desolder the STEMMA QT connector from the OLED and use short female headers on the OLED, or move the position of the microcontroller **below** the main PCB, which would require adding an opening in the main enclosure to access the USB connector.

These modifications probably also require modifications to the display bezel 3D print.

Solder the PCB

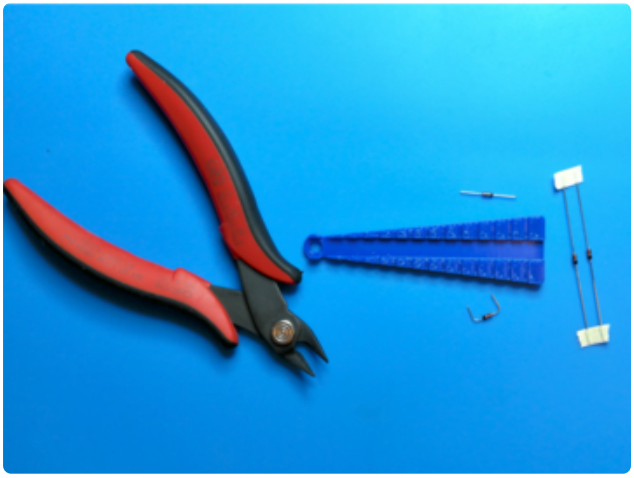
Finish board edges

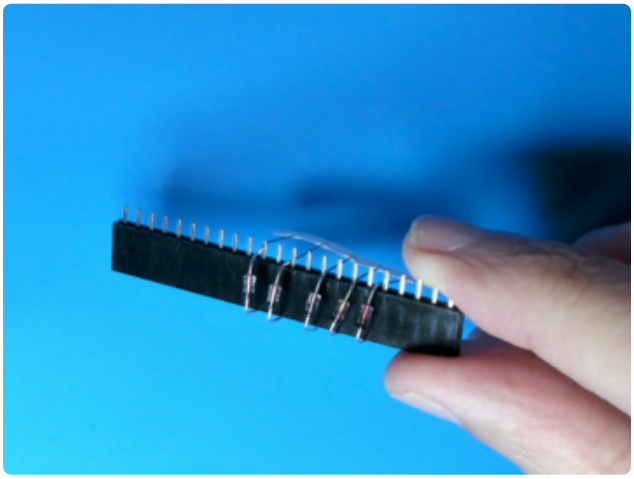


Often, PCBs come with little bits of the FR4 material poking out of the sides. Simply sand these away with some coarse sand-paper.

(This Sanding block design is by eponra on Thingiverse (https://adafru.it/TFG))

Solder the diodes

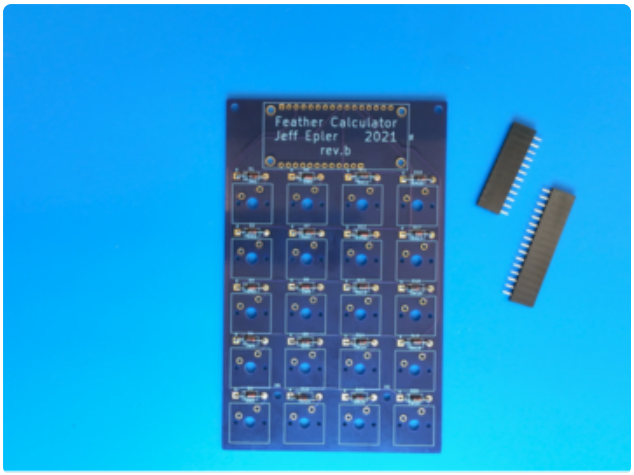




The board has 20 diodes, one for each keyswitch.

First, form the leads of the diodes using the 0.300" position on a lead former. You can use a standard PCB header as an improvised lead former if you don't have one.

(This resistor lead forming tool is by dnewman on thingiverse (https://adafru.it/TFH); the 2nd position on the narrow-short-imperial stl is correct for this project)



Next, Carefully noting the orientation of the diode, insert them in the labeled positions: Match the black strip on the diode body with the strip on the soldermask and insert the dioes into the board. Bend the leads slightly so that the diodes will not fall out of the board.

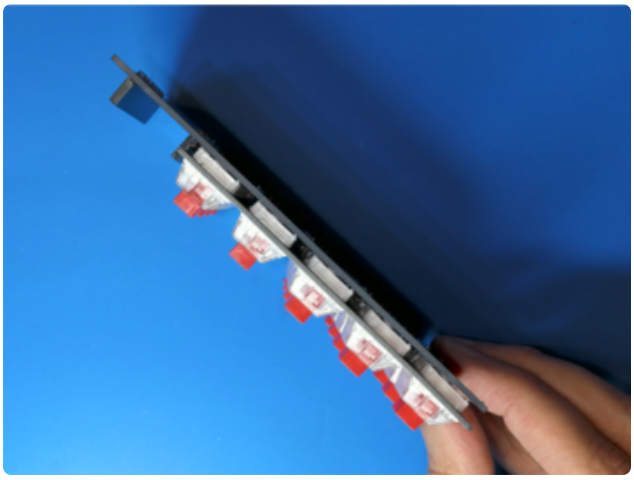
Then, flip the board over and solder the diodes from the back. Finally, use flush cutters to trim away the excess leads.

Solder the headers

Solder two rows of female headers into the two rows of pins at the top of the board. By inserting a Feather board with male headers into the female headers during soldering, you can ensure that the headers are properly aligned, not at a wonky angle. After soldering just one or two positions on each connector, check that they are properly aligned before soldering the rest.

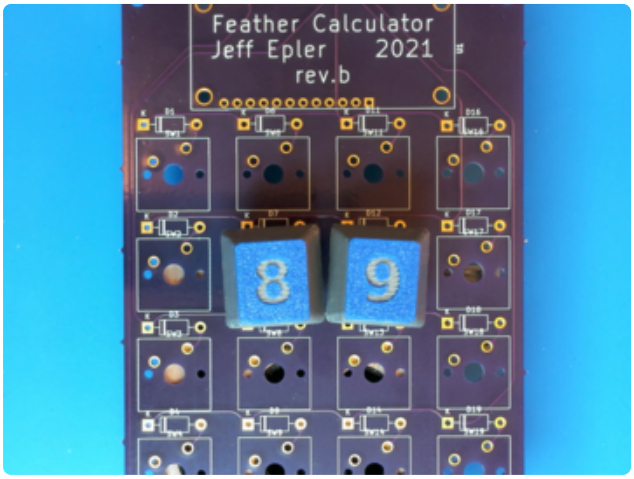
Solder the keyswitches

If you use the 3D printed keyswitch plate, you must snap the switches into the plate first, before soldering the keyswitches to the PCB.



If you are using the 3D printed plate—highly recommended—press the keyswitches into the plate from the back until they click. Carefully note the orientation of each keyswitch; if you will be using the screw-mount case, also note the position of the two screw access holes.

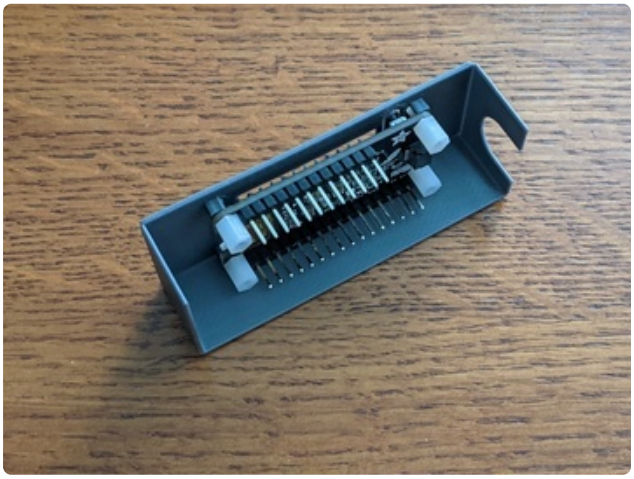
Then, once all the switches are in place in the plate, carefully fit them into the PCB, and solder.



If you are **not** using the 3D printed plate, fit and solder just one or two switches at a time, starting in the corners. Without the plate, the keyswitch can rotate a surprising amount within the holes; center the leads carefully as rotated keyswitches mean rotated keycaps and the esthetic result is not as good. (The effect was deliberately exaggerated for this photo)

Assemble the Calculator

Attach the Bezel to the OLED Display



In the M2.5 fastener set, find 4 each of the longest screw (M2.5×10) and the shortest standoff (M2.5×6 F-F)

Ensuring that the reset switch is positioned below the cutout tab, insert each screw from the front of the bezel, then through the OLED display. Using the stand-off like a nut, tighten the screen in place.

Adjust the rotation of the standoff nuts so that the OLED display can be pressed down onto the headers of the Feather.

If using heat-set inserts

Place the inserts loosely in the holes, with the narrow side down. With a soldering iron heated to 240°C (or the lowest setting available, if it's above 240°C), gently press the insert down into the plastic, keeping the iron in line with the post. **Note that this is not perpendicular to the base!** If you are using Adafruit's heat-set insert rig (https://adafru.it/Fis) one easy way to get the angle right is to print a second enclosure and stack them back to back so that the posts are vertical. Have more questions about using heat-set inserts? Check out our guide about them! (https://adafru.it/TFN)

Place the PCB in the enclosure and secure using 4 M2.5×6 screws.

If using the snap fit case

Place the PCB in the enclosure and gently press down until it moves past the snap fit ridges. You can release the PCB by gently flexing the sides of the enclosure away from each other.

Placing the keycaps

Simply ease the caps onto the stems and check that they operate smoothly. For tricky keys like 9 and 6 you may find that you've swapped their positions or rotated them, which leads to the key profile being inconsistent.

Stack the PCBs

Stack the RP2040 Feather and the OLED FeatherWing on top of the keypad PCB, and insert a USB cable for power and connection to your PC.

Install CircuitPython

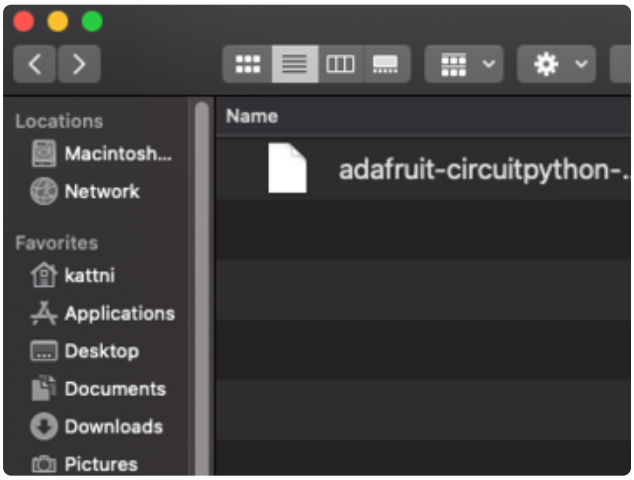
CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files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

CircuitPython Quickstart

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

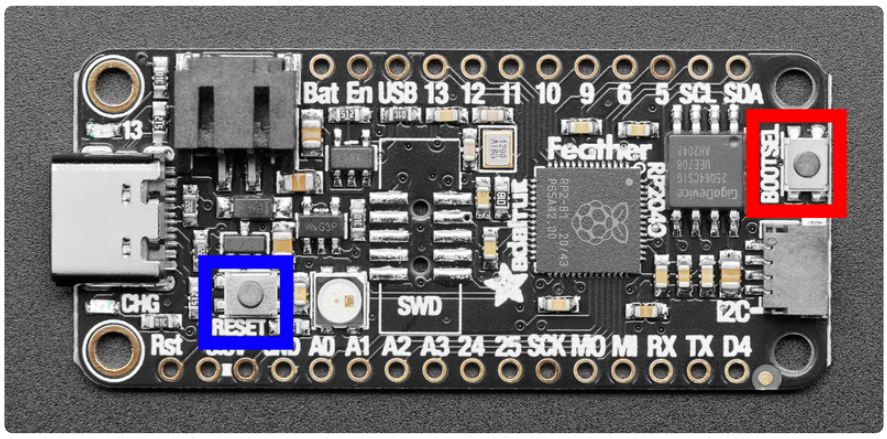
Download the latest version of CircuitPython for this board via circuitpython.org

https://adafru.it/R1D



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

Save it wherever is convenient for you.



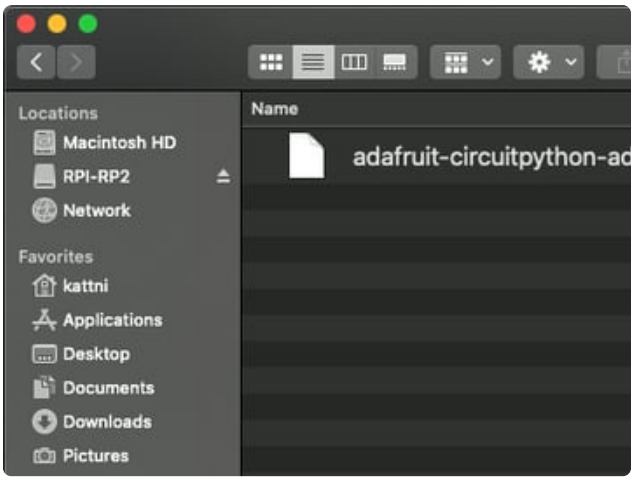
To enter the bootloader, hold down the **BOOT/BOOTSEL button** (highlighted in red above), and while continuing to hold it (don't let go!), press and release the **reset**

button (highlighted in blue above). **Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!**

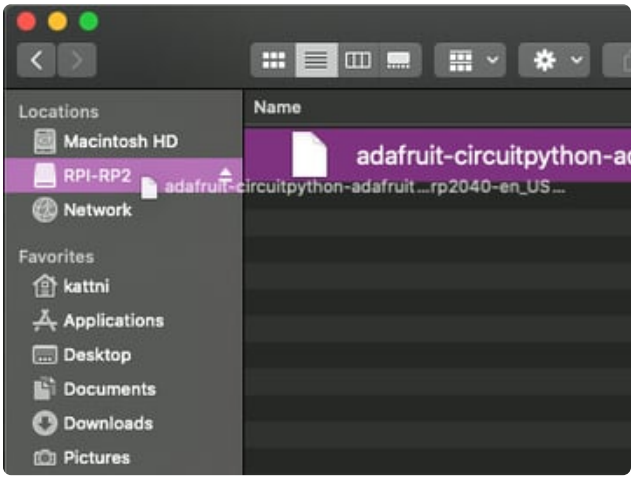
If the drive does not appear, release all the buttons, and then repeat the process above.

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.

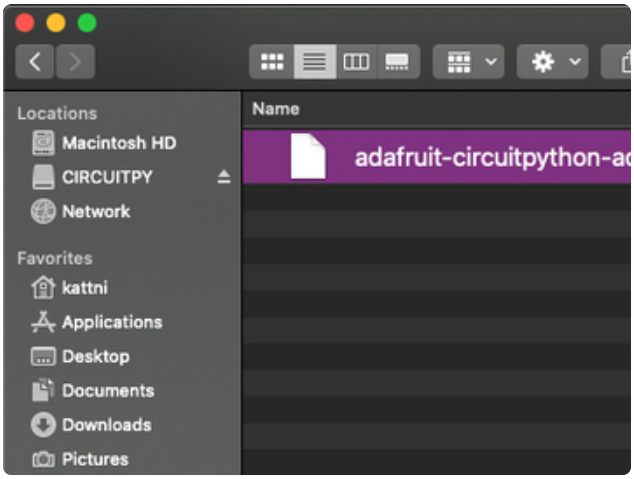
A lot of people end up using charge-only USB cables and it is very frustrating! **Make sure you have a USB cable you know is good for data sync.**



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

To enter safe mode when using CircuitPython, 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

If you successfully enter safe mode on CircuitPython, 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 CIRCUITPY doesn't show up as a disk drive after installing CircuitPython, try loading this 'nuke' UF2 to RPI-RP2. 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.

Download flash erasing "nuke" UF2

https://adafru.it/RLE

Upload the CircuitPython code

Use CircuitPython 7.0.0-alpha.4 or newer for the code in this guide!

Are you new to using CircuitPython? No worries, there is a full getting started guide here (https://adafru.it/cpy-welcome).

Adafruit suggests using the Mu editor to edit your code and have an interactive REPL in CircuitPython. You can learn about Mu and installation in this tutorial (https://adafru.it/ANO).

Download the Project Bundle

Your project will use a specific set of CircuitPython libraries and the **code.py** file. In order to get the libraries 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 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 Jeff Epler for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import board
import displayio
import keypad
import adafruit_displayio_sh1107
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
from adafruit_display_text import label
from adafruit_bitmap_font import bitmap_font

try:
    import usb_hid
except ImportError:
    usb_hid = None

K_SQ = "√"
K_CL = "<clear>"
K_FN = "<fn>"
K_PA = "<paste>"

KEYMAP0 = [
    K_CL, K_FN, '%',  '/',
    '7',  '8',  '9',  '*',
    '4',  '5',  '6',  '-',
    '1',  '2',  '3',  '+',
    '0',  '.',  K_SQ, '='
]

KEYMAP1 = [
    K_CL, None, '', '',
```

```
    '',   '',   '', '',
    '',   '',   '', '',
    '',   '',   '', '',
    '',   '',   '', K_PA,
]

keymaps = {
    0: KEYMAP0,
    1: KEYMAP1,
}

# pylint: disable=redefined-outer-name
def lookup(layer, key_number):
    while layer >= 0:
        key = keymaps[layer][key_number]
        if key is not None:
            return key
        layer -= 1
    return None

displayio.release_displays()
# oled_reset = board.D9

# Use for I2C
i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
display_bus = displayio.I2CDisplay(i2c, device_address=0x3C)

# SH1107 is vertically oriented 64x128
WIDTH = 128
HEIGHT = 64

display = adafruit_displayio_sh1107.SH1107(display_bus, width=WIDTH, height=HEIGHT,
rotation=180)
display.auto_refresh = False

font = bitmap_font.load_font("/digit-16px.pcf")
text_area = label.Label(font, text=" ", line_spacing=0.95)
text_area.y = 8
display.root_group = text_area

N = float

unary = {
    K_SQ: lambda a: a**.5,
}

binary = {
    '+': (lambda a, b: a+b, lambda a, b: a * (1+b/100)),
    '-': (lambda a, b: a-b, lambda a, b: a * (1-b/100)),
    '*': (lambda a, b: a*b, lambda a, b: a * (b/100)),
    '/': (lambda a, b: a/b, lambda a, b: a / (b/100)),
}

class Calculator:
    def __init__(self):
        self._number1 = N("0")
        self._number2 = None
        self.trail = ["Ready."]
        self.entry = ""
        self.op = None
        self.keyboard = None
        self.keyboard_layout = None

    def paste(self, text):
        if self.keyboard is None:
            if usb_hid:
                self.keyboard = Keyboard(usb_hid.devices)
```

```
            self.keyboard_layout = KeyboardLayoutUS(self.keyboard)
        else:
            return

    if self.keyboard_layout is None:
        self.add_trail("No USB")
    else:
        text = str(text)
        self.keyboard_layout.write(text)

        self.add_trail(f"Pasted {text}")


def add_trail(self, msg):
    self.trail = self.trail[-3:] + [str(msg).upper()]

@property
def number1(self):
    return self._number1

@number1.setter
def number1(self, number):
    self._number1 = number
    self.add_trail(number)

@property
def number2(self):
    if self.entry == "":
        if self._number2 is not None:
            return self._number2
        return None
    return N(self.entry)

@number2.setter
def number2(self, number):
    self._number2 = number
    self.entry = ''

def clear(self):
    self.number1 = N("0")

def clear_entry(self):
    self.number2 = None

def key_pressed(self, k): # pylint: disable=too-many-branches
    if k == K_CL:
        if self.entry:
            self.entry = self.entry[:-1]
        elif self.op:
            print("clear op")
            self.op = None
        elif self.number2 is None:
            self.clear()
        else:
            print("clear entry - op = ", self.op)
            self.clear_entry()

    if len(k) == 1 and k in "0123456789":
        self.entry = self.entry + k

    if k == "." and not "." in self.entry:
        if self.entry == "":
            self.entry = "0"
        self.entry = self.entry + k

    if k == K_PA:
        if self.number2 is not None:
            self.paste(self.number2)
        else:
```

```
                self.paste(self.number1)

        if k == "=":
            self.do_binary_op(0)

        if k == "%":
            self.do_binary_op(1)

        op = unary.get(k)
        if op:
            self.do_unary_op(op)

        if k in binary:
            if self.number2 is not None:
                if self.op:
                    self.do_binary_op(0)
                else:
                    self.number1 = self.number2
                    self.clear_entry()
            self.op = k

    def do_unary_op(self, op):
        if self.number2 is not None:
            self.number2 = op(self.number2)
        else:
            self.number1 = op(self.number1)

    def do_binary_op(self, i):
        if self.op and self.number2 is not None:
            op = binary[self.op][i]
            self.op = None
            self.number1 = op(self.number1, self.number2)
            self.clear_entry()

    def show(self):
        rows = [""] * 4
        trail = self.trail
        if len(trail) > 0:
            rows[2] = trail[-1]
        if len(trail) > 1:
            rows[1] = trail[-2]
        if len(trail) > 2:
            rows[0] = trail[-3]

        entry_or_number = self.entry or self.number2
        cursor = ' :' if (self.number2 is None or self.entry != "") else ""
        op = self.op or ''
        op = 'd' if op == '/' else op
        rows[-1] = f"{op}{entry_or_number or ''}{cursor}"
        for r in rows:
            print(r)
        text_area.text = "\n".join(rows)

km=keypad.KeyMatrix(
    row_pins=(board.A2, board.A1, board.A3, board.A0, board.D0),
    column_pins=(board.D25, board.D11, board.D12, board.D24))

calculator=Calculator()
calculator.show()

layer = 0
while True:
    ev = km.events.get()
    if ev:
        key = lookup(layer, ev.key_number)
        if ev.pressed:
            if key == K_FN:
                layer = 1
            try:
```

```
                calculator.key_pressed(key)
            except Exception as e: # pylint: disable=broad-except
                calculator.add_trail(e)
            calculator.show()

        elif ev.released:
            if key == K_FN:
                layer = 0

    else:
        display.refresh()
```

Adapting to other Adafruit Feather boards

The code can be adapted to most Adafruit Feather boards that support CircuitPython, but the Feather RP2040 board names some pins differently than most boards, so you'll need to make modifications. Find the portion of the code that creates the KeyMatrix and change D24/D25 to A4/A5 for most other Feathers:

```
km=keypad.KeyMatrix(
    row_pins=(board.A2, board.A1, board.A3, board.A0, board.D0),
    column_pins=(board.A5, board.D11, board.D12, board.A4))
```

Use your Calculator

Basic functions

Enter a number by typing digits; use the "." key to insert a decimal point.

To perform an operation, press the operation key, enter another number, then press "=". The pending operation is shown on the bottom line. Because the "7-segment"-style font does not have a "/" or "÷" character, "d" is shown to indicate a division operation.

To perform a percentage operation, press the operation key, enter another number, then press "%".

- The operation × **number** % multiplies by a percentage, 20 * 25% is 4.
- The operation + **number** % adds a percentage, 20 + 25% is 24.
- The operation **-** **number** % subtracts a percentage, 20 - 25% is 16.
- The operation ÷ **number** % divides by a percentage, 20 / 25% is 80.

Square Root

Press the √ key to compute the square root of the last number entered or calculated. For example, to calculate the "golden ratio" you can type: **1** + **5** √ / **2** which is equivalent to (1+√5)/2.

Clear and Backspace

The button marked **C** is a combination backspace/clear button. If a number has been entered, press backspace to erase it one digit at a time. If no number is entered (or the number has been erased) press it to clear the pending operation. If no number or operation is entered, press it to set the last number calculated to **0.0**.

Pasting to a Host Computer

Hold **fn** and press = to paste the last number calculated to an attached computer.

Make it your own

The author had so many ideas and not enough time to implement them.

- The **fn** key exists to access just one special feature. There should be more! Do you want to add parentheses? Trig functions? Exponents? Go wild!
- Investigate 2/3/4 line layouts & experiment with different fonts
- Add a mode where it operates as a pure numeric keypad or macro pad
- Install a battery under the PCB so you can go mobile
- Use the **udecimal** (https://adafru.it/TFO) library to upgrade to arbitrary-precision numbers