

# Pico Based Number Pad: Part 1 and 2

By [tinyboatproductions](#) in [CircuitsRaspberry Pi](#)



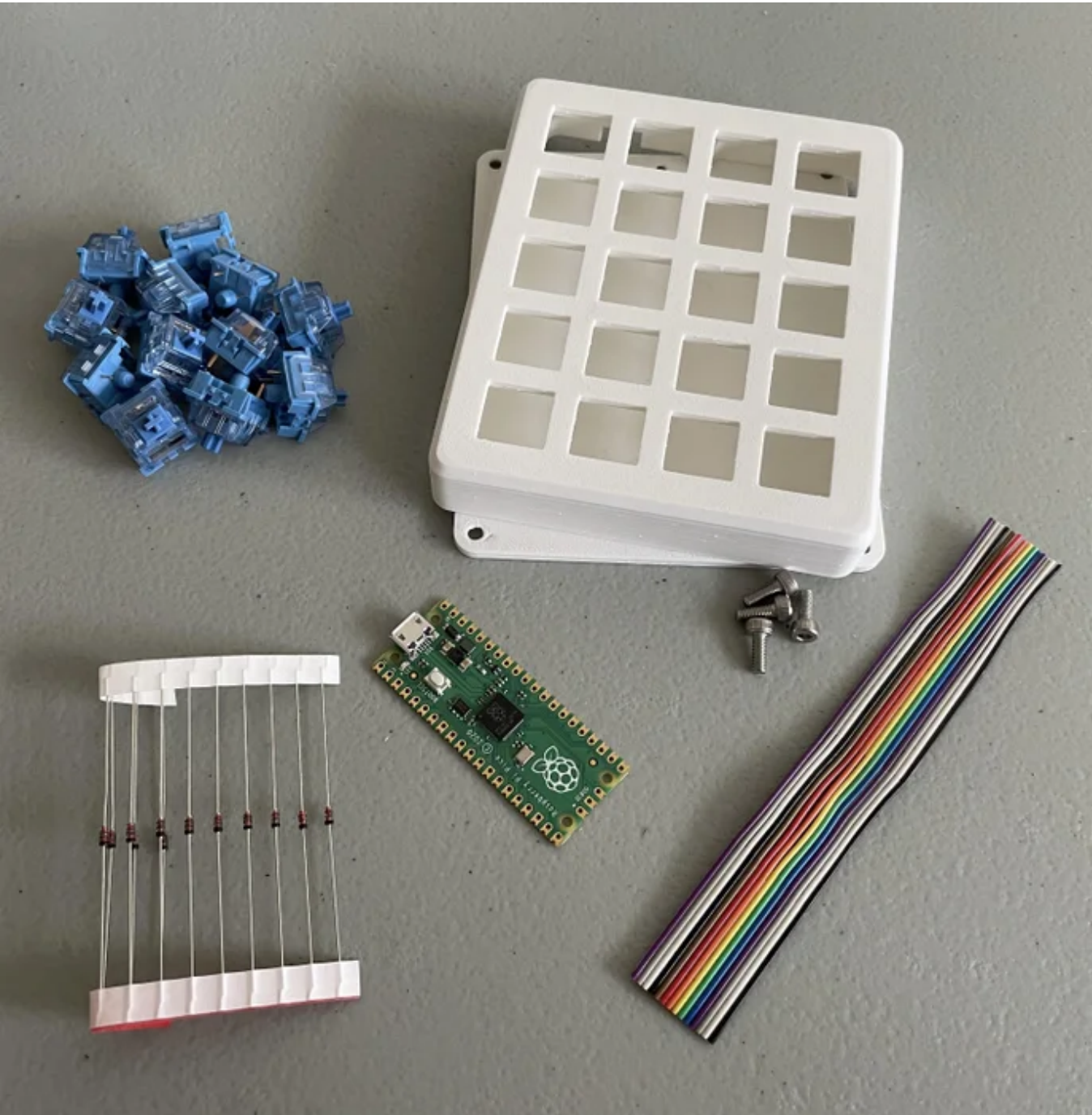
## Introduction: Pico Based Number Pad: Part 1 and 2

To day I am going to build a Raspberry Pi Pico based number pad. I am going to build a simple ortho linear number pad with 20 switches. Then I am going to go through how I installed KMK firmware on it.

KMK can be installed on python based micro controllers. It can be used for things as simple as number pads all the way up to full keyboards.

Edit (Aug, 2022): There was an update to KMK after the I published this. I have updated the code with the correct lines. Note that the screen shots have not changed. (Updated: kmk.matix—> kmk.scanners)

## Supplies



To build this num pad I needed the below supplies. Please keep in mind these will change depending on if what you build is larger or smaller.

None of the links below are affiliate links

## Supplies

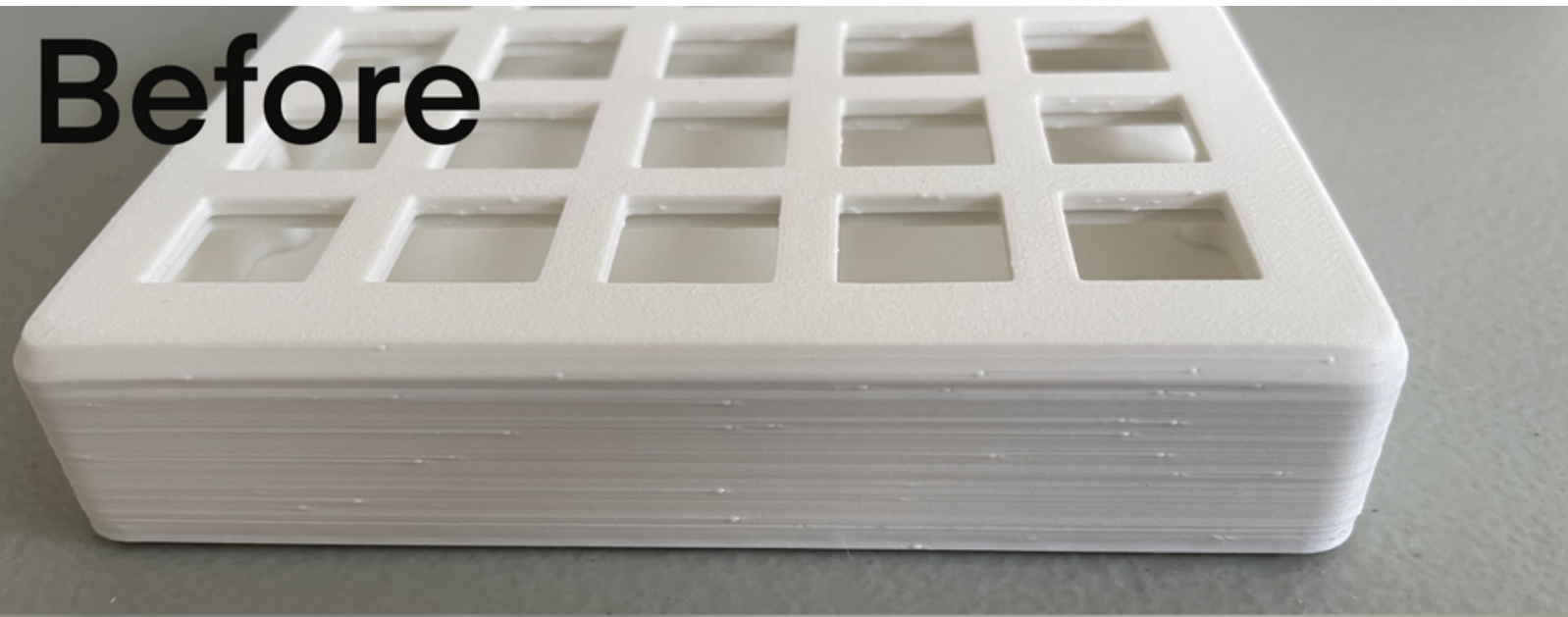
- (1) [Raspberry Pi Pico](#)
- (20) [Key Switches](#)
- (20) [Diodes](#)
- (20) Key Caps
- Wire
- Case and hardware
- Solder

## Tools

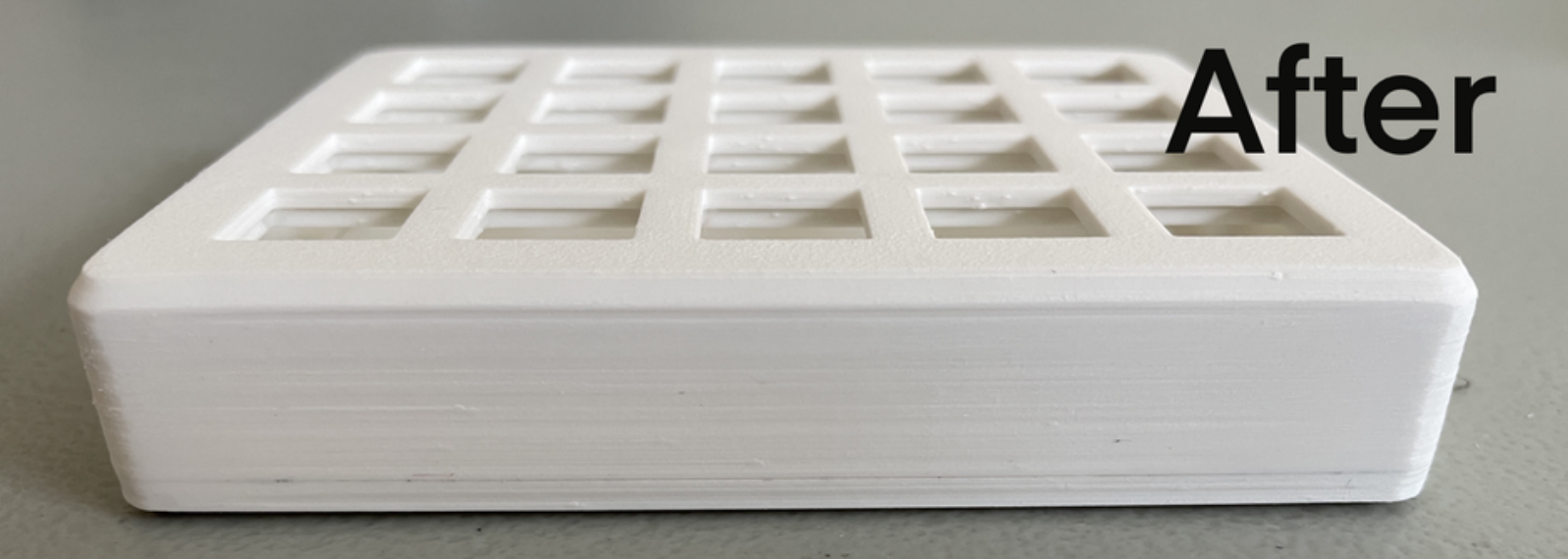
- 3D printer (For the case)
- Wire cutters
- Pliers
- Tweezers
- Sand Paper (optional)
- Soldering Iron
- Computer
- USB cord

## Step 1: Prepare the Case

# Before



# After



I designed and printed this case out of PLA on my Ender 3 Pro.

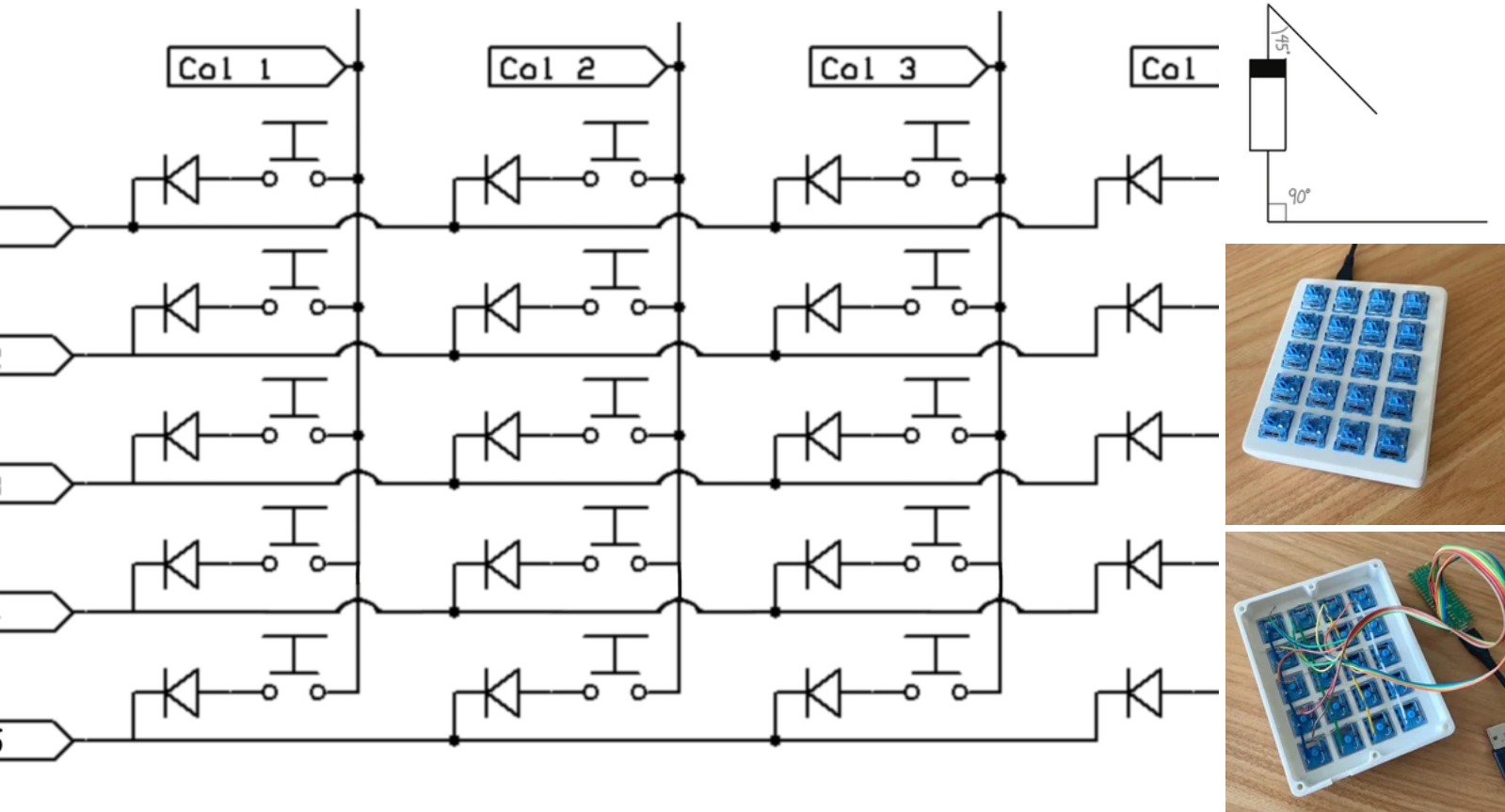
Link: <https://www.printables.com/model/156379-number-pad-case>

The case was designed to use up to 6 M3 screws, but for the num pad I will only be using 4 of them.

The edges came out a bit rough so I screwed the base onto the top and wet sanded the edges to smooth them out. In just a few minutes I was able to get rid of the bumps and gave the case a nice matte finish.



# Step 2: Install the Hardware Components

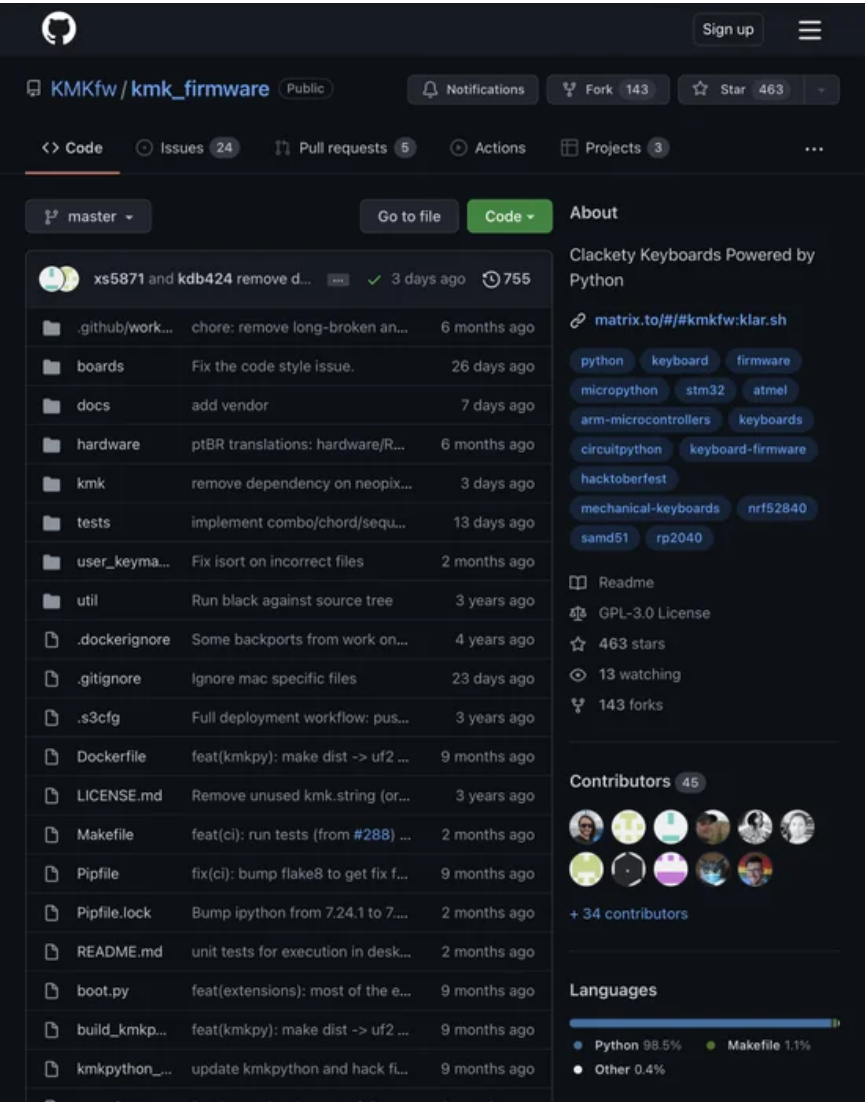


I am going to lump all the hardware into one step. As I mention in the Part 1 video I have some advice here so you can, hopefully have an easier time than me.

I have included the circuit diagram above.

- Install the switches
- Put all the switches in in the same direction. This makes it easier to install the diodes in the next step.
- Bend the diodes
- Bend one leg of the diode over so that it makes a 45 degree angle from the rest of the diode. Set this angle over the pin so the diode runs down the side of the switch
- Bend the other leg at 90 degrees. This can then be used to connect to the rest of the row of diodes
- Take a look at the other diagram in this step for a better idea of what I mean.
- Install the diodes
- Place the diode in place and then solder the diode to the switch first
- Solder the 90 degree leg to the rest of the row of diodes
- Double check the diode orientation every time, the orientation doesn't matter it just matters that they are all the same
- Trim the excess lead off
- Connect the columns
- I used a piece of solid core wire and cut the insulation every 14ish mm and then you can slid the insulation around to make small gaps for the other pin on the switch.
- I also remove about 1" (25mm) of insulation from one end to make enough space to move the insulation around
- When putting the wire in place, I zigzag between the pins to connect to. This helps hold it in place while I solder
- Trim off the excess wire
- Connect the rows and columns to the Pico
- I used some long sections of stranded wire to connect to each row and column
- Then connected them to the Pico. The pin numbers are not super import just keep track of them so they can be used in the code later.

# Step 3: What Is KMK?



Really quickly I am going to touch on what is KMK and why am I using it.

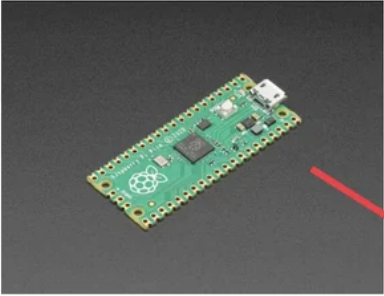
“KMK is a feature-rich and beginner-friendly firmware for computer keyboards written and configured in [CircuitPython](#).”

Here is a link to their [GitHub](#) with more information.

I am using it because I find it much easier to use than QMK, the standard for most keyboards. It can all be done in plain text on any computer and is in python, which I know.

I’m making this instructable and the videos linked because I want more eyes on this project because I think it’s great, and I’m not good enough at python to help build software so I hope someone who does this can help with this project.

# Step 4: Install the Software on the Pico



The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to do the same for microcontrollers with the release of the brand new **Raspberry Pi Pico**. This low-cost microcontroller board features a powerful new chip, the **RP2040**, and all the fixin's to get started with embedded electronics projects at a stress-free price.

The Pico is 0.825" x 2" and can have headers soldered in for use in a breadboard or perfboard, or can be soldered directly onto a PCB with the castellated pads. There's 20 pads on each side, with groups of general purpose input-and-output (GPIO) pins interleaved with plenty of ground pins. All of the GPIO pins are 3.3V logic, and are

CircuitPython 7.2.3

This is the latest **stable** release of CircuitPython that will work with the Pico.

Start here if you are new to CircuitPython.

Release Notes for 7.2.3

ENGLISH (US)

DOWNLOAD .UF2 NOW

Built-in modules available: `_bleio`, `adafruit_bus_device`, `adafruit_pixelbuf`, `aesio`, `alarm`, `analogio`, `atexit`, `audiobusio`, `audiocore`, `audiomixer`, `audiomp3`, `audiopwmio`, `binascii`, `bitbangio`, `bitmaptools`, `bitops`, `board`, `busio`, `countio`, `digitalio`, `displayio`, `errno`, `fontio`, `framebufferio`, `getpass`, `gifio`, `imagecapture`, `json`, `keypad`, `math`, `microcontroller`, `msgpack`, `neopixel_write`, `nvm`, `onewireio`, `os`, `paralldisplay`, `pulseio`, `pwmio`, `qrio`, `rainbowio`, `random`, `re`, `rgbmatrix`, `rotaryio`, `rtc`, `sdcario`, `sharpdisplay`, `storage`, `struct`, `supervisor`, `synthio`, `terminalio`, `time`, `touchio`, `traceback`, `ulab`, `usb_cdc`, `usb_hid`, `usb_midi`, `vectorio`, `watchdog`

91 lines (59 sloc) | 4.63 KB

Getting Started

Life was like a box of chocolates. You never know what you're gonna get.

KMK is a keyboard focused layer that sits on top of CircuitPython. As such, it should work with most boards that support CircuitPython. It is best to use the last stable version (>5.0). Known working and recommended devices can be found [here](#)

TL;DR Quick start guide

To infinity and beyond!

1. Install CircuitPython on your board. With most boards, it should be as easy as drag and dropping the firmware on the drive.

2. Get a copy of KMK from the master branch

3. Unzip it and copy the KMK folder and the boot.py file at the root of the USB drive corresponding to your board (often appearing as CIRCUITPY)

4. Create a new code.py or main.py file in the same root directory (same level as boot.py) with the example content hereunder:

IMPORTANT: adapt the GP0 / GP1 pins to your specific board !

```
print("Starting")

import board

from kmk.kmk_keyboard import KMKKeyboard
from kmk.keys import KC
from kmk.matrix import DiodeOrientation

keyboard = KMKKeyboard()

keyboard.col_pins = (board.GP0,) # try D5 on Feather, keeboard
keyboard.row_pins = (board.GP1,) # try D6 on Feather, keeboard
keyboard.diode_orientation = DiodeOrientation.COL2ROW

keyboard.keymap = [
    [KC.A,]
]

if __name__ == '__main__':
    keyboard.go()
```

With all the hardware in place we need to install some software on the Pico.

This can all be done without installing any other software.

I learned how to do all of this from the KMK GitHub [Getting Started](#) page. There is tons of good information there if you want more information or features.

The first piece of software needed is the most recent [CircuitPython UF2 from Adafruit](#).

Simply download the file

Hold the boot button down on the Pico and plug it into the computer (if it keeps giving you an error, try plugging it in without holding down the boot button)

The Pico should show up as a external storage device

Drag the downloaded UF2 file to the Pico. On its own it should eject and then after a few minutes reconnect with a new name

The next thing is the KMK firmware “install”.

Grab the latest version of KMK from their GitHub (get this link from the getting started page).

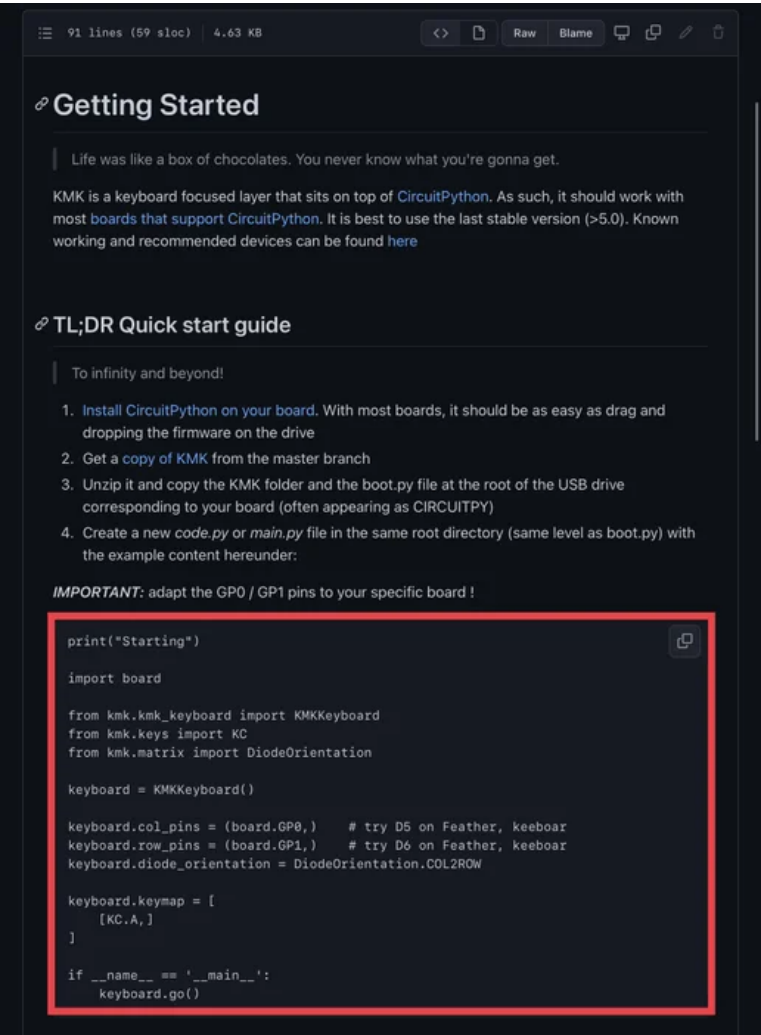
It will download a .zip file. Extract all the files.

In the folder find the KMK folder and the boot.py file.

Copy those items from the folder and pasted them in the Pico

That should be all the setup on the Pico

# Step 5: Add a Code.py



So far the stage has just been set for KMK now I need to tell it what to actually do.

This is done using a code.py file. There should be one on the Pico, if not just create a new file called code.py.

Luckily most of the hardworking has been done already. I just need to copy the example code from the GitHub page and paste it into the code.py file. I edit this file using NotePad++ but this can also be done in NotePad

I need to give this file all the specifics of my num pad.

To do this I first need to give the code the pin numbers for the rows and columns in line 11 and 12.

Next I need to change the diode orientation as the code needs to know how everything is set up. I changed it from COL2ROW to ROW2COL.

Most of the way done now, I just need to add the key map. Or what keys should be where. I referenced [this](#) web page to find all the key codes I can use. I won't go over this in detail but take a look and what I used and make any changes you want.

```
print("Starting")

import board

from kmk.kmk_keyboard import KMKKeyboard
from kmk.keys import KC
from kmk.scanners import DiodeOrientation

keyboard = KMKKeyboard()

keyboard.col_pins = (board.GP0,board.GP1, board.GP2, board.GP3) # try D5 on Feather, keeboar
keyboard.row_pins = (board.GP4, board.GP5, board.GP6, board.GP7, board.GP8) # try D6 on Feather, keeboar
keyboard.diode_orientation = DiodeOrientation.ROW2COL

keyboard.keymap = [
    [KC.A, KC.LSHIFT, KC.TAB, KC.KP_PLUS,
     KC.N7, KC.N8, KC.N9, KC.KP_ASTERISK,
     KC.N4, KC.N5, KC.N6, KC.KP_MINUS,
     KC.N1, KC.N2, KC.N3, KC.KP_SLASH,
     KC.BSPC, KC.N0, KC.KP_DOT, KC.KP_ENTER,
    ]
]

if __name__ == '__main__':
    keyboard.go()
```



# Step 6: Admire Your Work



Great job! That should be everything to get you into trouble.

If you have questions pleas let me know. If you make this or this helps you add a make here, or on the model page.

Thanks for reading!