# Table of Contents

## Overview
- Parts
- Tools

## CircuitPython
- CircuitPython Quickstart
- Safe Mode
- Flash Resetting UF2

## Assembly

## Installing KMK

## Setting up KMK
- Key Mapping

## Word Game Keyboard
Overview

In this guide, you'll learn how to set up your PB Gherkin 30% keyboard to use KMK firmware. You'll use the PB Gherkin, designed by 40percent.club, and the new Adafruit KB2040, to build a keyboard that you'll put KMK firmware on and learn how to remap it and create your own configuration.

What is KMK?

KMK is a feature-rich and beginner-friendly firmware for computer keyboards written and configured in CircuitPython.

What is the Gherkin?

The Gherkin is a fun project to practice soldering and is a fun talking piece to have on display! This Gherkin (PB Gherkin) can have switches oriented in any direction.

The kit comes with:

- Top/Bottom plates, PCB (all black FR4)
- Pro micro
- 30 Diodes
Enough screws and standoffs to hold it together (10 or more standoffs, 20 or more screws)

Things you’ll still need:

- MX style switches
- Keycaps
- Micro USB cable

All credit goes to 40percent.club for creating this amazing keyboard (and many other really fun projects!)
Parts

PB Gherkin Kit ()
The Gherkin is a fun project to practice soldering and is a fun talking piece to have on display! This Gherkin (PB Gherkin) can have switches oriented in any direction.

$25.00

Adafruit KB2040 - RP2040 Kee Boar Driver
A wild Kee Boar appears! It’s a shiny KB2040! An Arduino Pro Micro-shaped board for Keebs with RP2040. (#keeblife 4 evah) A lot of folks like using Adafruit...
https://www.adafruit.com/product/5302

USB Type A to Type C Cable - approx 1 meter / 3 ft long
As technology changes and adapts, so does Adafruit. This USB Type A to Type C cable will help you with the transition to USB C, even if you're still...
https://www.adafruit.com/product/4474

Tools

To complete this project you will need a few tools. You should be able to use any soldering iron, I just linked the one that I used. You also don't technically need the precision flat pliers, but unless you have a different way to bend the diodes, they will save you a lot of time.
A soldering iron such as:

Digital Genuine Hakko FX-888D (936 upgrade)
Known by engineers for making excellent quality tools & soldering irons! This is a genuine Hakko FX-888D with digital temperature control! We worked hard to get...
https://www.adafruit.com/product/1204

Solder:

Solder Wire - 60/40 Rosin Core - 0.5mm/0.02" diameter - 50 grams
If you want to make a kit you'll need some solder. This 0.1 lb (about 50 grams) spool is a good amount, not too much (like 1 lb spools) and not too little (like those little...
https://www.adafruit.com/product/1886

Something that can cut excess through-hole component leads such as these diagonal cutters:

Flush diagonal cutters
These are the best diagonal cutters, large super-comfortable grip to use and have strong nippers for perfect trimming of wires and leads. I've used my pair every day for years.
https://www.adafruit.com/product/152

Some way to bend the diodes such as these pliers:
Hakko Precision Flat Pliers
These Italian-made Hakko pliers are excellent for any precision work. The nose is flattened and fairly short to provide excellent control. The inside of the jaws is also flattened so...
https://www.adafruit.com/product/1368

A small Phillips head screwdriver to assemble the standoffs such as the one included in this kit:

Precision screwdriver set (6 pieces)
Break seals! Void warranties! With this handy screwdriver set at your hands, all hardware you touch will open up and reveal their secret innards. This set contains 6...
https://www.adafruit.com/product/424

CircuitPython
CircuitPython is a derivative of MicroPython 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
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 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.

Download flash erasing "nuke" UF2

Assembly

First, bend and solder all the diodes. Make sure to solder them so that the black band is on the side of the square soldering pad. I used these pliers () to bend them.
As you’re soldering the diodes you may find it easier to go through and cut the legs off the ones you’ve soldered. I’d recommend using these.

Next, insert the headers, long side through the Gherkin PCB, put the KB2040 on it to hold it in place, and solder the headers to the Gherkin PCB. The long side of the headers should be on the same side as the diodes, and the short side as well as the plastic bits should be on the other side. After you’ve soldered them, cut off the excess on the side you soldered so there’s space for the switches.
Now, solder the switches in. Make sure that the side of the plate with the markings on it is facing down. Just about any orientation should work, but some keycaps can be pretty specific, so make sure to test that before soldering all the switches.

Important: Make sure you have already installed CircuitPython using the instructions on the page labeled "CircuitPython". If you solder the KB2040 before doing so, this becomes much more difficult to do. If you already soldered the KB2040 on, you can reach the buttons using a pair of fine-tip curved tweezers.
After that, solder the KB2040 to the headers. Have it face inwards, and make sure to leave enough space so the switches don’t come in contact with it and short anything out (I cut the legs off the two switches above the KB2040 to ensure this wasn’t an issue. You’ll notice the KB2040 is has two extra pins. Don’t connect the pins labeled D+ and D- to anything.
Finally, attach the top plate and bottom plate with the included standoffs. Put standoffs in all the holes except for the two directly next to the USB and make sure that the logo is facing out on the bottom plate. (also disregard me using 3 very different switches in this build, I ran out of other switches). This keyboard has a pretty non-standard layout so I’d suggest looking at the diagrams in the "Setting up KMK" page when installing them.

---

Installing KMK

[Link to Installing KMK]
Setting up KMK

After having installed KMK on the previous page, you're going to want to click 'Download Project Bundle' on the file below. Unzip the file, and copy the lib directory, kb.py and code.py to your CIRCUITPY drive.

```python
# SPDX-FileCopyrightText: 2022 Eva Herrada for Adafruit Industries
# SPDX-License-Identifier: MIT

from kb import KMKKeyboard
from kmk.keys import KC
from kmk.modules.layers import Layers
from kmk.modules.modtap import ModTap

keyboard = KMKKeyboard()

# Designed for PB Gherkin (version without LEDs and where switches can be mounted in 4 different orientations)
modtap = ModTap()
layers_ext = Layers()
keyboard.modules = [layers_ext, modtap]

# Cleaner key names
_______ = KC.TRNS
XXXXXXX = KC.NO
FN1_SPC = KC.LT(1, KC.SPC)
FN2_BSPC = KC.LT(2, KC.BSPC)
FN3_C = KC.LT(3, KC.C)
FN4_V = KC.LT(4, KC.V)
CTL_Z = KC.MT(KC.Z, KC.LCTL)
ALT_X = KC.MT(KC.X, KC.LALT)
ALT_N = KC.MT(KC.N, KC.LALT)
CTL_M = KC.MT(KC.M, KC.LCTL)
SFT_ENT = KC.LSFT(KC.ENT)

keyboard.keymap = [
    [KC.Q,    KC.W,    KC.E,    KC.R,    KC.T,    KC.Y,    KC.U,    KC.I,
     KC.O,    KC.P,    KC.A,    KC.S,    KC.D,    KC.F,    KC.G,    KC.H,
     KC.J,    KC.K,    KC.L,    KC.ESC,
     CTL_Z,   ALT_X,   FN3_C,   FN4_V,   FN2_BSPC, FN1_SPC, KC.B,   ALT_N,
     CTL_M,   SFT_ENT,
    ],
    [KC.N1,   KC.N2,   KC.N3,   KC.N4,   KC.N5,   KC.N6,   KC.N7,   KC.N8,
     KC.N9,   KC.N0,   KC.F1,   KC.F2,   KC.F3,   KC.F4,   KC.F5,   KC.F6,
     KC.F7,   KC.F8,   ______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _________, KC.EXLM, KC.AT, KC.HASH, KC.DLR,
     KC.PERC, KC.CIRC, KC.AMPR, KC.ASTR,
     KC.LPRN, KC.RPRN, KC.F11, KC.F12, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _______, _________, KC.GRV]
```
After you've copied everything over, your CIRCUITPY drive should look something like this.

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

© Adafruit Industries
Page 17 of 23
The key mapping for the Gherkin is quite complicated. Feel free to click on them to enlarge them.
At this point, the keyboard should just work when you hit the keys. Feel free to mess around with the key maps. KMK has some good documentation on this process.
Word Game Keyboard

This variation on the board setup is used for playing word games such as Wordle and the NYT Crossword app on an iOS device. You'll connect them by using a USB-C to USB-A cable and an OTG USB-A to Lightning adapter.
iOS Lightning to USB OTG Cable
Your iOS phone or tablet may not have a USB port on the bottom but that doesn't mean you can't use it to connect USB devices. Secretly known as a 'Camera Connector' or...
https://www.adafruit.com/product/3940

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...
https://www.adafruit.com/product/5153

For Wordle, you have every key you need right on the board. The alphabet keys, Delete, and Enter are the only keys used in Wordle's on-screen keyboard.

Using the layers allows you to play the crossword puzzle, including navigation. For example, on layer three the A key is used for tab by holding the C mod, which is used to move to the next clue.

You can use arrow keys to move from space to space and switch from across to down. These are also on layer three, so a C mod and the R, D, F, G keys act as arrows.

The full layer mapping is shown in the keyboard.keymap in the code.py.
To use this alternate mapping, simply copy and paste the code here onto your Gherkin as code.py.

```python
# Wordle configuration
from kb import KMKKeyboard
from kmk.keys import KC
from kmk.modules.layers import Layers
from kmk.modules.modtap import ModTap

keyboard = KMKKeyboard()

modtap = ModTap()
layers_ext = Layers()
keyboard.modules = [layers_ext, modtap]

# Cleaner key names
    __________ = KC.TRNS
    XXXX = KC.NO
FN1_SPC = KC.LT(1, KC.SPC)
FN2_BSPC = KC.LT(2, KC.BSPC)
FN3_C = KC.LT(3, KC.C)
FN4_V = KC.LT(4, KC.V)
CTL_Z = KC.MT(KC.Z, KC.LCTL)
ALT_X = KC.MT(KC.X, KC.LALT)
ALT_N = KC.MT(KC.N, KC.LALT)
CTL_M = KC.MT(KC.M, KC.LCTL)
SFT_ENT = KC.MT(KC.ENT, KC.LSFT)
GUI_B = KC.MT(KC.B, KC.LGUI)

keyboard.keymap = [
    [KC.Q,    KC.W,    KC.E,    KC.R,    KC.T,     KC.Y,    KC.U,    KC.I,        
     KC.O,    KC.P,    KC.A,    KC.S,    KC.D,    KC.F,    KC.G,    KC.H,        
     KC.J,    KC.K,    KC.L,    KC.ESC,    SFT_ENT, CTL_Z,    ALT_X,    FN3_C,    FN4_V,    GUI_B,    ALT_N,    CTL_M,    FN1_SPC, FN2_BSPC
    ],
    [KC.N1,    KC.N2,    KC.N3,    KC.N4,    KC.N5,    KC.N6,    KC.N7,    KC.N8,        
     KC.N1,    KC.N2,    KC.N3,    KC.N4,    KC.N5,    KC.N6,    KC.N7,    KC.N8,        
    ]
]
```
if __name__ == '__main__':
    keyboard.go()