Lemon Mechanical Keypad

Created by Ruiz Brothers

https://learn.adafruit.com/qtpy-lemon-mechanical-keypad-macropad

Last updated on 2023-08-29 04:38:22 PM EDT
## Table of Contents

### Overview
- DIY USB-HID With a Twist of Lime
- Lemon, Lime and Fruits
- Kailh Mechanical Key Switches
- Custom Keys
- Prerequisite Guides
- Parts

### Circuit Diagram
- Adafruit Library for Fritzing

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

### Code
- Download Project Bundle
- Upload Code, Libraries and Assets
- USB HID Keyboard and Mouse
- Modify Controls

### CAD Files
- CAD Parts List
- CAD Design
- CAD Assembly
- Printing Keycaps
- Design Source Files
- SVG Keycap Decals

### 3D Printing
- Fuzzy Skin Tutorial
- Textured Skin
- Fuzzy Skin
- Textured Key Caps
- Printing Keycaps

### Wiring
- Install Switches to Key Plate
- Installed Switches
- Wires
- Solder Ground Wires
- Soldered Common Ground
- Solder Switches 1-2
- Solder Switches 3-4
- Solder Switches 5-6
- Add Heat Shrink Tubing
- NeoPixel Wiring
- Solder Ground to Jewel
- Solder to QT Py
- Solder Jewel to QT Py
- Solder Switches to QT Py
• Soldered QT Py

Assembly

• Key Plate and Core
• Installing Key Plate
• Installing Top Cover
• Install Cover
• Install Core to Skin
• Installing Core
• Install NeoPixel Jewel and QT Py
• Install Bottom Cover
• Installing Key Caps
• Final Build
Overview

DIY USB-HID With a Twist of Lime

Make a custom USB-HID keypad in the shape of a citrus fruit. This keypad features six mechanical switches for controlling media players like Spotify, iTunes, YouTube, etc. Powered by the Adafruit QT Py RP2040 and CircuitPython.

Inside the case is a NeoPixel Jewel that slowly fades in and out making the keypad illuminate. The QT Py and NeoPixel are housed in a 3D printed case with parts that snap fit together.
Lemon, Lime and Fruits
The case and key caps can be 3D printed in different colors. Choose a color scheme that best matches your preferred flavor of fruit. Could it be an orange, a lemon, a lime or maybe a watermelon? You decide!

Kailh Mechanical Key Switches
Click, click, click... Mechanical key switches offer a tactile feeling and there's a few varieties of switch type. Choose the type that best suits your preference.
Custom Keys
One of the goals of this project was to create a unique layout for a keypad. Originally conceived as a circular keypad, the enclosure evidently turned into a fruit inspired shape. As a result, the key caps are cute little triangular pieces.

Prerequisite Guides

Take a moment to walk through the following guides:

- Adafruit QT Py RP2040
- CircuitPython LED Animations
- NeoPixel Uber Guide
- CircuitPython HID Keyboard and Mouse
Parts

List of parts required for this build:

Adafruit QT Py RP2040 ()
Kailh Mechanical Switches ()
NeoPixel Jewel ()

Adafruit QT Py RP2040
What a cutie pie! Or is it... a QT Py? This diminutive dev board comes with one of our new favorite chip, the RP2040. It's been made famous in the new
https://www.adafruit.com/product/4900

NeoPixel Jewel - 7 x 5050 RGB LED with Integrated Drivers
Be the belle of the ball with the NeoPixel Jewel! We fit seven of our tiny 5050 (5mm x 5mm) smart RGB LEDs onto a beautiful, round PCB with mounting holes and a...
https://www.adafruit.com/product/2226
<table>
<thead>
<tr>
<th>Item Description</th>
<th>Adafruit Product Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kailh Mechanical Key Switches - 10 packs</td>
<td><a href="https://www.adafruit.com/product/4996">https://www.adafruit.com/product/4996</a></td>
</tr>
<tr>
<td>- Cherry MX Compatible</td>
<td></td>
</tr>
<tr>
<td>For crafting your very own custom keyboard, these Kailh</td>
<td></td>
</tr>
<tr>
<td>mechanical key switches are deeee-luxe!Come in a pack of 10</td>
<td></td>
</tr>
<tr>
<td>switches, plenty to make a...</td>
<td></td>
</tr>
<tr>
<td>Silicone Cover Stranded-Core Wire - 50ft 30AWG Black</td>
<td><a href="https://www.adafruit.com/product/3164">https://www.adafruit.com/product/3164</a></td>
</tr>
<tr>
<td>Silicone-sheathing wire is super-flexible and soft, and it's</td>
<td></td>
</tr>
<tr>
<td>also strong! Able to handle up to 200°C and up to 600V, it will</td>
<td></td>
</tr>
<tr>
<td>do when PVC covered wire wimps out. We like this wire...</td>
<td></td>
</tr>
<tr>
<td>Keep your electronics from going barefoot, give them little</td>
<td></td>
</tr>
<tr>
<td>rubber feet! These small sticky bumpers are our favorite</td>
<td></td>
</tr>
<tr>
<td>accessory for any electronic kit or device. They are sticky,</td>
<td></td>
</tr>
<tr>
<td>but...</td>
<td></td>
</tr>
</tbody>
</table>

1 x **Heat Shrink**
Pre-Cut Multi-Colored Heat Shrink Pack Kit

1 x **Solder Wire**
60/40 Rosin Core - 0.5mm/0.02" diameter - 50 grams

1 x **Adjustable 60W Pen-Style Soldering Iron**
120VAC USA Plug - BEST 102C
Circuit Diagram

The diagram below provides a visual reference for wiring of the components. This diagram was created using the software package [Fritzing](https://fritzing.org).
Adafruit Library for Fritzing

Use Adafruit's Fritzing parts library to create circuit diagrams for your projects. Download the library or just grab individual parts. Get the library and parts from GitHub - Adafruit Fritzing Parts.

NeoPixel Jewel to QT Py RP2040

The list of connections for the NeoPixel Jewel and QT Py RP2040.

- DOUT to A0 on QT Py
- GND to GND QT Py
- 5V to 5V QT Py

Switches to QT Py RP2040

The list of connections for the switches and QT Py RP2040. All of the switches share common ground. The last switch in the array connects the ground pin on the NeoPixel Jewel.

- Switch 1 pin 1 to A1 on QT Py
- Switch 2 pin 1 to A2 on QT Py
- Switch 3 pin 1 to A3 on QT Py
- Switch 4 pin 1 to SCK on QT Py
- Switch 5 pin 1 to MI on QT Py
- Switch 6 pin 1 to MO on QT Py
- Switch 1 pin 2 to GND on NeoPixel
- Switch 1-6 pin 2 to common ground

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

---

**Code**

**Download Project Bundle**

Once you've installed the latest version of CircuitPython onto your board, you'll need to grab the code, libraries and any assets included with the project.

Click the download project bundle button below to get the code, libraries and assets all in one!

```python
# SPDX-FileCopyrightText: 2021 Noe Ruiz for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import digitalio
import board
import usb_hid
import neopixel
from adafruit_hid.consumer_control import ConsumerControl
from adafruit_hid.consumer_control_code import ConsumerControlCode
from adafruit_led_animation.animation.pulse import Pulse
from adafruit_led_animation.animation.solid import Solid
from adafruit_led_animation.animation.group import AnimationGroup
from adafruit_led_animation.sequence import AnimationSequence

from adafruit_led_animation.color import WHITE, BLACK

pixel_pin = board.A3
num_pixels = 7

pixels = neopixel.NeoPixel(pixel_pin, num_pixels, brightness=.3, auto_write=False)
internal_pixel = neopixel.NeoPixel(board.NEOPIXEL, 1, brightness=1, auto_write=False)

# The button pins we'll use, each will have an internal pullup

# The keycode sent for each button, will be paired with a control key
buttonkeys = [
    ConsumerControlCode.PLAY_PAUSE,
    ConsumerControlCode.FAST_FORWARD,
    ConsumerControlCode.VOLUME_INCREMENT,
    ConsumerControlCode.MUTE,
    ConsumerControlCode.VOLUME_DECREMENT,
    ConsumerControlCode.REWIND
]

# the keyboard object!
cc = ConsumerControl(usb_hid.devices)
```

©Adafruit Industries
# our array of button objects
buttons = []

# make all pin objects, make them inputs w/pullups
for pin in buttonpins:
    button = digitalio.DigitalInOut(pin)
    button.direction = digitalio.Direction.INPUT
    button.pull = digitalio.Pull.UP
    buttons.append(button)

animations = AnimationSequence(
    AnimationGroup(
        Pulse(pixels, speed=0.05, color=WHITE, period=6),
        Solid(internal_pixel, color=BLACK),
    ),
)

print("Waiting for button presses")

while True:
    # animate neopixel jewel
    animations.animate()
    # check each button
    for button in buttons:
        if not button.value:  # pressed?
            i = buttons.index(button)

            print("Button #%d Pressed" % i)

            while not button.value:
                pass  # wait for it to be released!
                # type the keycode!
                k = buttonkeys[i]  # get the corresp. keycode
                cc.send(k)
                time.sleep(0.01)

Upload Code, Libraries and Assets

Unzip the project bundle and upload the files to the CIRCUITPY drive.

Your CIRCUITPY drive should look like this after you've uploaded the code, libraries and assets.
USB HID Keyboard and Mouse

Check out the guides linked below for more information on using the USB-HID library for CircuitPython.

**HID Keyboard Mouse Guide**

**ReadTheDocs – HID Library**

Modify Controls

The six key switches are assigned the consumer controls for play/pause, previous track, next track, volume up, volume down and mute. Adjust the following code to change the keys control or the order of the arrangement. The `buttonpins` and `buttonkeys` are ordered chronologically.

```python
# The button pins we'll use, each will have an internal pullup

# The keycode sent for each button, will be paired with a control key
buttonkeys = [
    ConsumerControlCode.PLAY_PAUSE,
    ConsumerControlCode.FAST_FORWARD,
    ConsumerControlCode.VOLUME_INCREMENT,
    ConsumerControlCode.MUTE,
    ConsumerControlCode.VOLUME_DECREMENT,
    ConsumerControlCode.REWIND
]
```
CAD Files

CAD Parts List
STL files for 3D printing are oriented to print "as-is" on FDM style machines. Parts are designed to 3D print without any support material. Original design source may be downloaded using the links below.

- Keyplate.stl
- Skin-nofuzzy.stl
- Skin-fuzzy.stl
- Core.stl
- Top-Cover.stl
- Bottom-Feather.stl
- Bottom-QTPy.stl
- Cap.stl
- Cap-alt.stl
- Cap-Set.stl

Download STLs-v2.zip

Download CAD Source

CAD Design
The enclosure is designed to be modular. The parts can be swapped out for different colors or features. The bottom cover can be swapped out, for example, with built-in standoffs for a Feather board.
CAD Assembly
The caps fit over the stems of the switches. The switches are press fitted into a plate. The plate is fitted into the core. The skin is fitted over the core. The bottom cover is press fits under the skin. The QT Py RP2040 and NeoPixel Jewel are fitted into the built-in holders on the bottom cover.

Printing Keycaps
Try printing a single key cap before printing a whole set. For the best quality print, print one key cap at a time. The stem needs very accurate tolerances in order to press fit properly onto the mechanical switch.
Design Source Files
The project assembly was designed in Fusion 360. This can be downloaded in different formats like STEP, STL and more. Electronic components like Adafruit's board, displays, connectors and more can be downloaded from the Adafruit CAD parts GitHub Repo.

SVG Keycap Decals
Optionally use a vinyl cutter to create decals for the key cap. Use the SVG file provided to create a template or cut as is. Use vinyl with an adhesive backing and transfer tape.

Download SVG template

3D Printing

Fuzzy Skin Tutorial

In this tutorial we'll take a look at using the fuzzy skin feature in CURA. I demo the Fuzzy Skin Outside Only feature and talk about how I optimized the 3D model in order to avoid unwanted geometry. I also quickly take a look at slicing a cauldron and display stand.
Textured Skin
The skin features a texture that provides better grip to print surfaces. The look and feel of the texture can be modified by adjusting slice setting in the CURA slicing software.

Fuzzy Skin
The outer skin of the case is 3D printed using the fuzzy skin feature in CURA.

In CURA, under the printing settings pane, search for fuzzy skin. Use the default settings to slice the skin.
Textured Key Caps

The key caps were 3D printed using a textured PEI flexible build plate. The sheet of PEI is powered coated and offers a unique texture on the first layer of 3D printed parts. This texture looks really great on the key caps.
Printing Keycaps

Try printing a single key cap before printing a whole set. For the best quality print, print one key cap at a time. The stem needs very accurate tolerances in order to press fit properly onto the mechanical switch.
Install Switches to Key Plate

The switches are press fitted into the key plate. The switches feature mini clips that keep the body of the switch panel mounted.
Installed Switches
The placement of the switches are symmetrical so the orientation doesn't matter.

Wires
Use short wires to connect the mechanical switches common ground. The switches signal connections will need to be long wires in order to reach the QT Py.

Signal Wires 7x – 15cm long
Ground Wires 5x – 6cm long
Solder Ground Wires
Solder the short wires to the mechanical switches. Solder two wires to a single pin to share common ground.

Soldered Common Ground
Each switch should be soldered with a short wire. The last mechanical switch should use a longer ground wire to reach the QT Py.
Solder Switches 1-2
Connect one of the longer wires to one of the signal pins on a switch. Continue to solder another long wire to the switch next in series. It doesn't matter which order the switches are in.
Solder Switches 3-4
Proceed to solder longer wires to the third and forth switch.
Solder Switches 5-6
Proceed to solder longer wires to the fifth and sixth switch.
Add Heat Shrink Tubing
Add pieces of heat shrink tubing to group the wires together. Apply heat to the tubing to shrink!
NeoPixel Wiring
Use a piece of 3-wire silicone ribbon cable to connect the NeoPixel jewel to the QT PY board.

3-wire ribbon cable – 9.5cm

Solder Ground to Jewel
Since there is only a single ground pin on the QT Py, we'll use the remaining ground pin on the NeoPixel Jewel.

Solder the longer ground wire from the switches to the available ground pin on the NeoPixel Jewel.
Solder to QT Py
The switches and NeoPixel Jewel are ready to connect to the QT Py.

Solder Jewel to QT Py
Connect the three wires from the NeoPixel Jewel to the QT Py board. Solder the 5V wire to the 5V pin, ground wire to the ground pin, and DIN wire to the A0 pin on QT Py.

Solder Switches to QT Py
Solder the six wires from the switches to the QT Py board.

Use pins A1, A2, A3, SCK, MI, MO.
Soldered QT Py
Double check the wiring to ensure the connections are solid.

Assembly

Key Plate and Core
The key plate is press fitted into the core. Start by placing the QT Py and NeoPixel Jewel through the top of the core.
Installing Key Plate
Orient the key plate so the notches are fitted through the rails in the core. Press the key plate through the core until it's flush with the inner ledge.

Installing Top Cover
The top cover is press fitted on top of the core.
Install Cover
Place the top cover over the core and line up the notches with the rails. Firmly press the top cover into the core so it's flush with the surface.

Install Core to Skin
The core is fitted into the skin. Start by inserting the QT Py and NeoPixel Jewel through the skin.
Installing Core
Fit the core through the center of the skin. Firmly press the core through the skin until they're flush.

Install NeoPixel Jewel and QT Py
Press fit the NeoPixel Jewel and QT Py into the built-in holders in the bottom cover.
Install Bottom Cover
Line up the notches in the bottom cover with the rails on the bottom of the skin. Firmly press the bottom cover into the skin.
Installing Key Caps
Start by press fitting the key caps into the stems of each mechanical switches.

Final Build
Congrats on building the lemon key pad! Use a USB-C cable to connect the QT Py to your computer.