No-Solder Visualizer Capacitive Touch Controller

Created by John Park

https://learn.adafruit.com/no-solder-visualizer-capacitive-touch-controller

Last updated on 2023-11-08 01:14:13 PM EST
# Table of Contents

## Overview
- Parts
- Other Parts

## Build the Touch Controller
- QT Py Mount
- Alligator Wires
- Copper Tape
- Overlay
- Graphic Overlay

## CircuitPython
- Set up CircuitPython Quick Start!

## CircuitPython Libraries
- CircuitPython Library Bundle
- Library Install Troubleshooting

## Code the Touch Controller
- Text Editor
- Code
- How It Works
- Use the Touch Control
Use the power of touch to control the show!

Animus () is a beautiful music visualizer for macOS made with Processing. There are a number of keyboard shortcuts to adjust the visualizations, but instead of a computer keyboard, you can use a custom touch controller you design and build yourself. Coded with CircuitPython.

This project uses the MPR121 12-key cap touch sensor STEMMA QT breakout as input for a QT Py, which will act as a USB keyboard device for your computer.

The great thing about this project is there's NO soldering required - you can plug-and-play the capacitive touch board into the QT Py using a simple cable connection - and then use copper tape to make the pads!

You can adapt this project for any OS and any software that uses keyboard shortcuts. So long as it is a USB HID keystroke, combo, or consumer control (e.g., volume) you can build it!
Adafruit MPR121 12-Key Capacitive Touch Sensor Gator Breakout
Add lots of touch sensors to your next microcontroller project with this easy-to-use 12-channel capacitive touch sensor breakout board, starring the MPR121. This chip can handle up to...
https://www.adafruit.com/product/4830

Adafruit QT Py - SAMD21 Dev Board with STEAMMA QT
What a cutie pie! Or is it... a QT Py? This diminutive dev board comes with our favorite lil chip, the SAMD21 (as made famous in our GEMMA M0 and Trinket M0 boards). This time it...
https://www.adafruit.com/product/4600

STEMMA QT / Qwiic JST SH 4-pin Cable - 100mm Long
This 4-wire cable is a little over 100mm / 4" long and fitted with JST-SH female 4-pin connectors on both ends. Compared with the chunkier JST-PH these are 1mm pitch instead of...
https://www.adafruit.com/product/4210
Short Wire Alligator Clip Test Lead (set of 12)
Connect this to that without soldering using these handy mini alligator clip test leads. Approximately 4.5" overall cables with alligator clip on each end, color coded. You get 12...
https://www.adafruit.com/product/1592

Copper Foil Tape with Conductive Adhesive - 6mm x 5 meters long
Copper tape can be an interesting addition to your toolbox. The tape itself is made of thin pure copper so it's extremely flexible and can take on nearly any shape. You can solder...
https://www.adafruit.com/product/3483

Other Parts
You'll also need some cardboard or other surface to affix the controller. I used a 18cm wooden disc from a craft store (shown here with copper tape applied).

Glu Dots or double stick adhesive tape works well to secure the QT Py to the board.

If you want to print a graphic overlay, you can use some spray mount adhesive to secure it.
You can get creative with the form factor of your touch control. Any non-conductive material will work as a base -- here I chose a wooden disc that I had from a craft store -- just make sure it's something to which the copper tape will stick well.

With the alligator clips and copper tape you can form the capacitive touch pads in any layout or pattern you like. Plus, since you don't need direct finger-to-copper contact for the touch to register, you can cover the copper pads with something more attractive, such as fabric, paper with a printed UI design, even a canvas with painted artwork.
QT Py Mount
To begin, mount the QT Py near an edge of the board so that you'll be able to plug in a USB cable for power and HID data.

I used a Glu Dot to affix it.

Plug in the STEMMA QT wire to the QT Py and the MPR121 breakout.
Alligator Wires
Clip the alligator wires to the breakout board's twelve connectors.
Copper Tape
Cut a 2” segment of copper tape and remove the protective paper.

Fold over one end a small amount to give the clip something to bite into a little, then stick the strip to the base, folding a bit of excess around to the other side.

Continue doing this for each connector. Then, you can add some additional tape to help things stay put.

The larger surface area of each pad will give you a larger target for your touch interactions as well.
Overlay

You can use the controller as is, or add a graphic overlay on top of the copper touch pads. You don't need to touch them directly for them to work!

Here's a sample graphic I made if you'd like to use it as a launching off point.
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.

Set up CircuitPython Quick Start!

Follow this quick step-by-step for super-fast Python power :)

If you want to get started with your QT Py, and you have NOT soldered a chip to the back of it, download CircuitPython from the following link:

Download the latest version of CircuitPython for your QT Py from CircuitPython.org

If you have soldered a GD25Q16 SPI flash chip to the bottom of your board, you must use the Haxpress version of CircuitPython for the Adafruit QT Py for the flash to work! If you have NOT soldered a SPI flash chip to your QT Py, do NOT use this download! It will not give you 2MB of flash space without a chip!

Download the latest version of CircuitPython for your QT Py Haxpress from CircuitPython.org

Click the link above and download the latest UF2 file.

Download and save it to your desktop (or wherever is handy).
Plug your QT Py into your computer using a known-good USB cable.

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

Double-click the small RST (reset) button, and you will see the NeoPixel RGB LED turn green. If it turns red, check the USB cable, try another USB port, etc.

If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!

You will see a new disk drive appear called QTPY_BOOT.

Drag the adafruit_circuitpython_etc.uf2 file to QTPY_BOOT.
The red LED will flash. Then, the QTPY_BOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

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

---

### CircuitPython Libraries

The touch controller code requires some CircuitPython libraries to work. Complete the following steps to get your board ready.

### CircuitPython Library Bundle

You'll need to install some libraries on your QT Py.

Carefully follow the steps to find and install these libraries from Adafruit's [CircuitPython Library Bundle](https://circuitpython.org/library-bundle). Our CircuitPython starter guide has a great page on how to install libraries from the bundle.

You'll need to manually install the following libraries by copying the files to the lib folder on your CIRCUITPY drive:

- adafruit_bus_device
- adafruit_mpr121.mpy
- adafruit_hid

### Library Install Troubleshooting

If you get an error indicating you have run out of space while copying files over, please refer to the Prevent & Remove MacOS Hidden Files section of the [Troubleshooting page](https://circuitpython.org/library-bundle) for information on resolving the issue.
Code the Touch Controller

Text Editor

Adafruit recommends using the Mu editor for editing your CircuitPython code. You can get more info in [this guide](https://www.adafruit.com/product/933).

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

Code

Copy the code shown below and paste it into Mu, then save it to your CircuitPY drive with the name `code.py`

```python
# SPDX-FileCopyrightText: 2021 John Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import board
import busio
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode
import usb_hid
import adafruit_mpr121

# Create I2C bus.
i2c = busio.I2C(board.SCL, board.SDA)
# Create MPR121 object.
mpr121 = adafruit_mpr121.MPR121(i2c)
# Note you can optionally change the address of the device:
# mpr121 = adafruit_mpr121.MPR121(i2c, address=0x91)

keylist = [
    Keycode.RIGHT_ARROW,
    Keycode.ONE,
    Keycode.TWO,
    Keycode.THREE,
    Keycode.FOUR,
    Keycode.W,
    Keycode.A,
    Keycode.S,
    Keycode.D,
    Keycode.B,
    Keycode.I,
    Keycode.H,
]

# Loop forever testing each input and sending keystrokes when they're touched.
while True:
    # Loop through all 12 inputs (0-11).
    for i in range(12):
        # Call is_touched and pass it then number of the input. If it's touched
        # it will return True, otherwise it will return False.
        if mpr121[i].value:
            # print("Input {} touched!".format(i))
```

©Adafruit Industries
How It Works

Libraries

First, the libraries are imported, including the USB HID keyboard and adafruit_mpr121 library for the cap touch breakout.

```python
import time
import board
import busio
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode
import usb_hid
import adafruit_mpr121
```

Setup

The i2c bus and sensor objects are created next:

```python
i2c = busio.I2C(board.SCL, board.SDA)
mpr121 = adafruit_mpr121.MPR121(i2c)
```

Keyboard

The keyboard object is created and a list of keycodes that will be sent. You can adjust these to suit other uses if you like.

```python
kbd = Keyboard(usb_hid.devices)
keylist = [
    Keycode.RIGHT_ARROW,
    Keycode.ONE,
    Keycode.TWO,
    Keycode.THREE,
    Keycode.FOUR,
    Keycode.W,
    Keycode.A,
    Keycode.S,
    Keycode.D,
    Keycode.B,
    Keycode.I,
    Keycode.H,
]```
Main Loop

The main loop is super simple! It loops through twelve times, checking each cap touch pad to see if it's being touched.

If a pad is being touched, the corresponding key from the list is sent.

Then, a short pause for debouncing and it's done! This cycle repeats forever and ever.

```python
while True:
  # Loop through all 12 inputs (0-11).
  for i in range(12):
    if mpr121[i].value:
      kbd.send(keylist[i])
      time.sleep(0.15)  # Small delay to keep from spamming output messages.
```

Use the Touch Control

First, download Animus from this link and then install it. For non mac os operating systems, you will need to grab your code from the GitHub page.

With the QT Py plugged into your computer over USB, launch Animus and start some music playing for it to react to.

Notice the UI on the side of the screen? You don't need to mouse and click on those any longer! Try touching your touch controller pads instead!

Now, you can mix the visuals in style!