Deluxe 4x4 NeoPixel Keypad

Created by John Park

https://learn.adafruit.com/deluxe-4x4-neopixel-neokey-keypad

Last updated on 2023-08-29 04:43:28 PM EDT
Table of Contents

Overview  
- Parts

Assemble the Keypad  
- Prep the Case  
- Prep the PCB  
- Snap-apart  
- Wire the NeoKey PCBs and QT Py  
- Solder to the PCB  
- Board Positioning  
- Solder to QT Py  
- Add Switches to Plate and PCB  
- Case Assembly  
- USB Port Extension  
- Back Plate  
- Keycap Time

Neo Keypad 4x4 MIDI Controller  
- Download  
- Install the Firmware  
- Bootloader Mode  
- Play It  
- Custom Code in Arduino  
- Customization
Overview

Create a high-end, custom keypad using NeoKey Ortho Snap-apart PCBs, mechanical keyswitches, keycaps, a QT Py, and an aluminum case.

You can create anything from a USB HID keypad, to MIDI controller, to music clip launcher using this design.

Code it in Arduino on the QT Py M0, or you could opt instead for a QT Py RP2040 and use CircuitPython as well.
Parts

4x4 Key Deluxe Aluminum Keypad Shell Enclosure
For all the DIY keebfans in the house here is a very fancy and sturdy 4x4 Keypad
https://www.adafruit.com/product/5071

NeoKey 5x6 Ortho Snap-Apart Mechanical Key Switches w/ NeoPixel
For folks who want ready-to-go keeb action, we’ve got the lovely Adafruit
Macropad with a 3x4 grid of MX+NeoPixel key switches -...
https://www.adafruit.com/product/5157

Adafruit QT Py - SAMD21 Dev Board with STEMMA 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
DIY USB or HDMI Cable Parts - 10 cm Ribbon Cable
If you're looking to make a custom cable with any two of our USB or HDMI adapters, then you'll need to have these flex cables handy! Making custom cables has never been...
https://www.adafruit.com/product/3560

DIY USB Cable Parts - Straight Micro B Jack
If you love DIY cable-rigging, then these USB adapters are right up your alley! Making custom USB cables has never been easier. Just grab one of our
https://www.adafruit.com/product/4107

DIY USB Cable Parts - Straight Type C Plug
If you love DIY cable-rigging, then these USB adapters are right up your alley! Making custom USB cables has never been easier. Just grab one of our
https://www.adafruit.com/product/4108

Kailh Mechanical Key Switches - Linear Red - 12 Pack
For crafting your very own custom keyboard, these Kailh Red Linear mechanical key switches are deeeeee-luxe! With smooth actuation and Cherry MX compatibility,...
https://www.adafruit.com/product/5122
Assemble the Keypad

Prep the Case

The first step is to prep the case.
Remove Back
Unscrew the four screws holding the plastic back plate and metal mid plate, and pull both out.
Remove Keyswitch Plate
Unscrew the four screws holding in the keyswitch plate and remove it.

Prep the PCB

Next you'll trim the excess NeoKey PCBs away, leaving just the 4x4 matrix needed for the 16-key pad.
Please use eye protection when cutting PCB material to avoid fragments getting in ones eyes.

Snap-apart

Use pliers to carefully snap off the top and bottom mounting strips.

Then, use your hands to carefully wiggle the extra two columns off.

Finally, remove the extra row.

Remember to save these extras somewhere safe, you can use them for another project later!
Wire the NeoKey PCBs and QT Py

Follow the guide and photos below to wire up the four columns, four rows, power, ground, and NeoPixel data line.
Start by wiring the power, ground and columns. I used thin, silicone covered wire, which is nice and flexible. It is thin enough to thread through the PCB holes without first stripping the insulation, which makes it easier to "tighten" things to even lengths all at once.

After that you will cut, strip, and solder.
Thread the wires into the QT Py, then add the NeoPixel data wire and the row wires.
Solder to the PCB

Trim, strip and solder the wires on the PCB now. Then, you'll pull the wires to length on the QT Py side of things.
Board Positioning
Now you can place the QT Py toward the center of the PCB and pull the wires to even lengths with a little bit of slack.
Solder to QT Py
With the wires tightened up, you can now trim, strip, and solder them to the QT Py as shown here.
Add Switches to Plate and PCB
Click your switches into the plate and then into the PCB. Be mindful of the orientation and check that the legs line up with the sockets before pushing switches into place.

Case Assembly
To assemble the case, first drop the switch plate in and screw it in.
USB Port Extension

Use the short DIY ribbon cable and straight USB port and jack to create an easy to use connector. This fits very nicely compared to the QT Py, which will not lay flush at the port end of the case due to the keyswitch sockets.
Adhere
Use an adhesive square or double stick foam tape to adhere the bottom of the USB port extender to the metal mid-plate as shown. This will make the port stable enough for plugging and unplugging the USB cable.

Make sure not wires or ribbon cable are pinched by the plate as you close it.
Back Plate
Screw the back plate into place. Don't crank down too hard on the screws or you may risk cracking the plastic back plate.

Keycap Time

Now you can add your keycaps and move on to coding the Deluxe 4x4 Neo Keypad!
Neu Keypad 4x4 MIDI Controller

In order to make it simple to get up and running with your Keypad with no programming required, we've created a drag-and-drop firmware you can use. You can skip to the section on Custom Code in Arduino if you want to customize it.
Download

First, download the firmware file linked in the button below and save it to your computer hard drive somewhere you'll be able to find it, such as your Downloads folder.

Keypad 4x4 Clocked uf2

Install the Firmware

Plug your QT Py into your computer with a good quality, data capable USB cable. Life is too short to go through the pain of accidentally using a power-only USB cable, so please round up any you own, cut them in half, travel to a distant land, bury them, and dance on their grave.

Your QT Py will start up with whatever code was running on it. Let's change that!

Bootloader Mode

Now, we'll put the QT Py into bootloader mode. In this mode it will appear as a USB drive on your computer and will be ready to receive a new .uf2 firmware file. Double-click the reset button on the top side of the board (you will need to open the case to do this if you previously closed it).

The QTPY_BOOT drive will show up on your computer. Simply drag-and-drop the key 4x4.uf2 file onto it. It will copy the file and then automatically restart, running the code!

Play It

Now you can run the synth software of your choice (or hardware synth with USB MIDI Host capabilities) and pick the QT Py M0 as your active MIDI input.

The code will run at a clock rate of 120 BPM, sending MIDI clock, and when you press the keys you can play an A major chord (plus bonus G#) in two octaves.
Custom Code in Arduino

If you’d like to customize things further, you can edit the code in Arduino. Open your Arduino IDE. If you are new to Arduino, check out this guide: Adafruit Arduino IDE Setup.

You'll also need to make sure you've updated your Board Definitions to the latest version the SAMD boards by heading to Tools -> Board Manager in the Arduino IDE, filter for "Adafruit SAMD" and choose the install or update.

You'll also need to install the Tiny USB library, Keypad library, MIDI library, and NeoPixel library.

```cpp
// SPDX-FileCopyrightText: 2021 John Park for Adafruit Industries
// SPDX-License-Identifier: MIT
// Keypad 4x4 for NeoKey Ortho Snap-apart PCB & QT Py M0
// Sends MIDI NoteOn/Off and Clock
// --works well in VCV Rack for keeping clock timing via MIDI-CV CLK

#include <Adafruit_TinyUSB.h>
#include "Adafruit_Keypad.h"
#include <Adafruit_NeoPixel.h>
#include <MIDI.h>

// User variables
bool latch_mode = false;  // set latch/toggle mode
int BPM = 120;  // set BPM
int MIDI_OUT_CH = 1;  // pick your midi output channel here

uint32_t interval = 60000 / BPM;

Adafruit_USBD_MIDI usb_midi;
MIDICREATEINSTANCE(Adafruit_USBD_MIDI, usb_midi, MIDI);

const byte ROWS = 4; // rows
const byte COLS = 4; // columns

//define the symbols on the buttons of the keypads -- used for Serial output, debugging
char keys[ROWS][COLS] = {
  {'1','2','3','4'},
  {'5','6','7','8'},
  {'A','B','C','D'},
  {'E','F','G','H'}
};

byte rowPins[ROWS] = {A3, A2, A1, A0}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {SCL, A6, A7, A8}; //connect to the column pinouts of the keypad

//define the MIDI notes to send per key
int pads[] = {
  76, 77, 79, 80,
  69, 71, 72, 74,
  64, 65, 67, 68,
  57, 59, 60, 62
};
```
int pads[] = { // corresponds to 1010music blackbox pads for sample launching
  48, 49, 50, 51,
  44, 45, 46, 47,
  40, 41, 42, 43,
  36, 37, 38, 39
};

int current_key = 0; // current key press int
int pixorder[] = { // to convert "snake" order to grid order
  0, 1, 2, 3,
  7, 6, 5, 4,
  8, 9, 10, 11,
  15, 14, 13, 12
};

//initialize an instance of keypad
Adafruit_Keypad customKeypad = Adafruit_Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);

#define NEO_PIN SDA
#define NUMPIXELS ROWS*COLS
Adafruit_NeoPixel pixels(NUMPIXELS, NEO_PIN, NEO_GRB + NEO_KHZ800);

uint32_t red = pixels.Color(100, 0, 0);
uint32_t light_blue = pixels.Color(0, 50, 60);
uint32_t black = pixels.Color(0, 0, 0);

uint32_t priorcolor = pixels.Color(0,0,0); // store state of color before sequence changes it
uint32_t currentcolor = pixels.Color(0,0,0);

int current_pixel = 0; // current pixel
unsigned long previousMillis = 0;

void setup() {
  Serial.begin(9600); // use for debugging
  MIDI.begin(MIDI_OUT_CH); // begin MIDI before the delay to settle
delay(1000);
customKeypad.begin(); // begin keypad
  pixels.begin(); // begin neopixels

  for(int i=0; i<NUMPIXELS+1; i++) { // light up each pixel
    pixels.setPixelColor(pixorder[i], light_blue);
    pixels.setPixelColor(pixorder[i-1], black);
    pixels.show(); // Send the updated pixel colors to the hardware.
    delay(int(interval/4));
  }
}

void loop() {
  customKeypad.tick();
  unsigned long currentMillis = millis();

  while(customKeypad.available()){
    keypadEvent e = customKeypad.read(); // scan the keypad for changes
    //Serial.print((char)e.bit.KEY);

    if(e.bit.EVENT == KEY_JUST_PRESSED){
      current_key = (e.bit.ROW * ROWS) + e.bit.COL ;
      //Serial.println(" pressed");
      //Serial.println(interval);
      currentcolor = pixels.getPixelColor(pixorder[current_key]);
      if (latch_mode == true){
        if (currentcolor==0){
          MIDI.sendNoteOn(pads[current_key], 127, MIDI_OUT_CH);
          pixels.setPixelColor(pixorder[current_key], light_blue);
        }
      }
    }
  }
}
else{
    MIDI.sendNoteOff(pads[current_key], 0, MIDI_OUT_CH);
    pixels.setPixelColor(pixorder[current_key], black);
}
}
else{
    MIDI.sendNoteOn(pads[current_key], 127, MIDI_OUT_CH);
    pixels.setPixelColor(pixorder[current_key], light_blue);
}
pixels.show();
}
else if(e.bit.EVENT == KEY_JUST_RELEASED){
    current_key = (e.bit.ROW * ROWS) + e.bit.COL ;
    //Serial.println(" released");
    if (latch_mode == false){
        MIDI.sendNoteOff(pads[current_key], 0, MIDI_OUT_CH);
        pixels.setPixelColor(pixorder[current_key], black);
        pixels.show();
    }
    // delay(10);
}
//----- Running light
if (currentMillis - previousMillis >= interval) {
    MIDI.sendClock();
    if (current_pixel==0){  // loop around to last pixel for color reset
        pixels.setPixelColor(pixorder[NUMPIXELS-1], priorcolor);
    } else{
        pixels.setPixelColor(pixorder[current_pixel-1], priorcolor);
    }
priorcolor = pixels.getPixelColor(pixorder[current_pixel]);  // grabs the
    // current color of the pixel for later use
    pixels.setPixelColor(pixorder[current_pixel], red);
    pixels.show();
    current_pixel++;
    previousMillis = currentMillis;
} if (current_pixel==NUMPIXELS){
    current_pixel=0;
}

Customization

Latching

The main thing you may want to customize is the behavior of the switches -- do they latch or are the momentary?

Adjust this line from false to true and you will have each key latched as a NoteOn message when you press it. Press it a second time to unlatch it and send NoteOff.

bool latch_mode = false; // set latch/toggle mode
BPM clock

Beyond that you can change the BPM (note PPQN may vary between synths/softsynths so you may need to subdivide to get the desired clock rate).

```cpp
int BPM = 120;  // set BPM
```

MIDI Channel

You can also pick a different MIDI out channel:

```cpp
int MIDI_OUT_CH = 1;
```

Scales

The array called `pads[]` determines which notes are sent. You can adjust this to suit your needs. By default, the Neo Keypad sends two different octaves of A major scales with a bonus G#.

```cpp
int pads[] = {  // two A scales with added G#
    76, 77, 79, 80,
    69, 71, 72, 74,
    64, 65, 67, 68,
    57, 59, 60, 62
};
```

Uncomment this array for a chromatic set of notes that correspond to the pad launching mode of the 1010Music blackbox:

```cpp
/*int pads[] = {  // corresponds to 1010music blackbox pads for sample launching
    48, 49, 50, 51,
    44, 45, 46, 47,
    40, 41, 42, 43,
    36, 37, 38, 39
};*/
```