Your own macro keyboard: four NeoKey keys and the QT Py RP2040

Four mechanical switches with LEDs, an RP2040 board and a breadboard: the computer sees it as a USB keyboard and each key sends a shortcut.

Online guide, always up to date

By **Ada Intelisano** · informaticait.it technical team

IN SHORT

- Each NeoKey has a socket for a mechanical switch and a NeoPixel: S+ goes to the pin, S− to GND, and the LEDs are chained from OUT to IN.
- The two rows of pins on each NeoKey fit into rows c and h of the breadboard; the QT Py controls everything with five pins: A0–A3 and RX.
- In the Arduino IDE you need Earle Philhower's arduino-pico core; the Keyboard library comes with it.
- With `Keyboard.begin(KeyboardLayout_en_US)` the board types correctly on a computer set to the US layout; for another layout you change just that name.

Time: an afternoon | Level: some soldering | Before: guides no. 20, 48 and 70

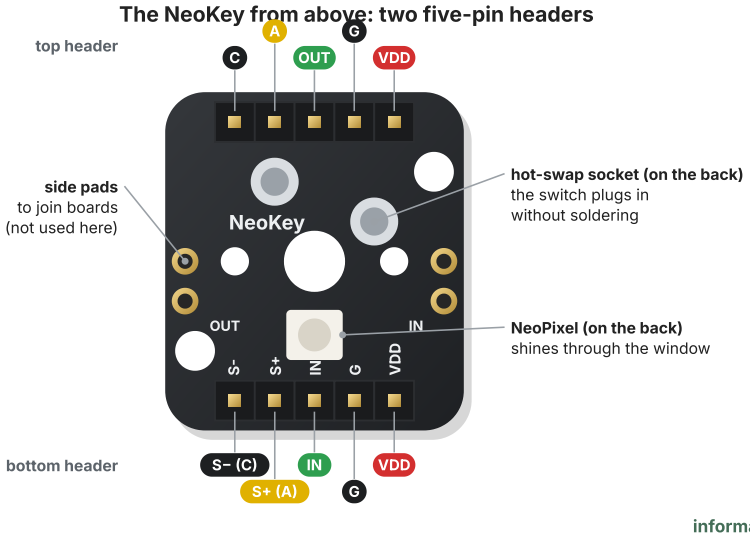
Copy, paste, undo: the same key combinations every day. With a macro keyboard you put them on keys of your own, which you press with one finger and which light up in the color you choose. Here we build one with four keys, using Adafruit **NeoKey** boards, small boards with a socket for a mechanical switch and a NeoPixel LED, and a board the size of a postage stamp, the **QT Py RP2040**. The computer recognizes it as an ordinary keyboard: there is no software to install.

1 The parts

- four Adafruit 4978 **NeoKeys** and a **QT Py RP2040**; each board comes with its own header to solder;
- four MX-compatible **mechanical switches**, for example Kailh White (clicky) or Brown (no click, just a small bump), and four clear **keycaps**, which let the light through. They are not included with the NeoKeys;
- an 830-point **breadboard**, about ten male-to-male jumper wires and a USB-C cable for the computer.

New word: HID
Human Interface Device: the family of USB devices a person uses to control a computer, such as keyboards and mice. The computer recognizes them on its own. The QT Py can present itself as an HID keyboard.

2 A closer look at the NeoKey



- In the middle there is a **hot-swap socket** soldered on the back: the switch plugs in from above and comes out again, with no solder. Without a plate to hold it, it can work loose; if needed, Adafruit suggests a dab of hot glue.
- **S+** and **S–** are the two ends of the switch, also called A (anode) and C (cathode): there is a diode in series, so the direction matters. With the board's pull-up, **S+ goes to the pin** and **S– to GND**.
- **VDD** and **G** power the NeoPixel, from 3 to 5 V. The NeoPixel receives data on **IN** and passes it from **OUT** to the next board, as in a strip (guide no. 48).
- The two rows of pins carry the same signals, except IN (bottom) and OUT (top).

3 Solder the headers

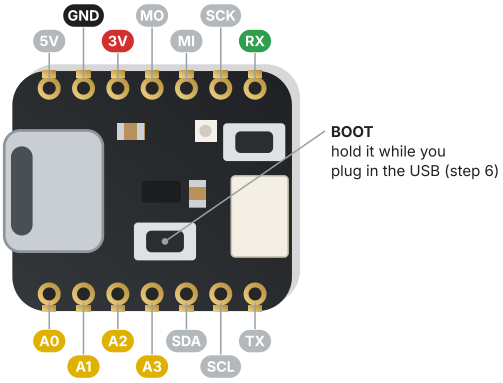
- On each NeoKey solder **both rows** of five pins, with the tips pointing down, on the socket side: Adafruit recommends it because the board stays steadier, and here you need both rows anyway.
- A trick to keep them straight: push the headers into the breadboard in rows c and h, rest the board on top and solder (guide no. 70). Do the same with the QT Py, but in rows d and h: its rows are 15.24 mm apart, that is six steps (our calculation from the board file).
- The NeoKey rows are 17.78 mm apart, that is seven 2.54 mm steps (our calculation from the board file): they straddle the center channel and leave two holes free above (a, b) and two below (i, j).
- Plug the switches into the sockets and put the keycaps aside: they go on at the end.

4 The QT Py RP2040

We use **3V** and **GND** for power, **A0, A1, A2, A3** for the four keys and **RX** for the LED data. Watch the names: in the code A0–A3 are written just like that, but for the RX pin you use the

GPIO number of the RP2040 chip, which is **5** (Adafruit warns that the names printed on the board are not the Arduino ones).

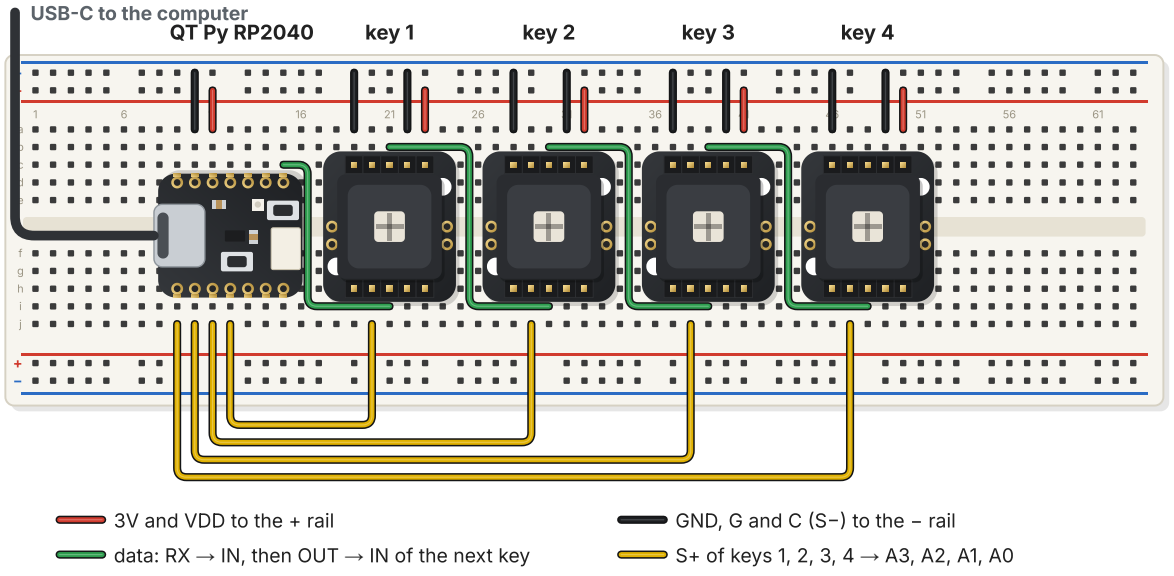
The QT Py RP2040: the pins we use



informaticait.it

5 Put it all on the breadboard

The wiring: QT Py RP2040 and four NeoKeys on the breadboard



informaticait.it

- The QT Py goes on the left, with the USB facing outward and its headers in rows d and h; the NeoKeys go on the right, one every nine columns, with the headers in rows c and h.
- From the top header: **VDD** to the + rail, **G** and **C** (that is, S−) to the − rail. From the QT Py: 3V to +, GND to −.
- The data go along a chain: **RX** → **IN** of key 1, then **OUT** of each key → **IN** of the next one. The jumper wires run down through the gap between one board and the next.
- From the bottom header, **S+** of each key goes to a pin: key 1 → A3, 2 → A2, 3 → A1, 4 → A0.

6 Set up the Arduino IDE

- In **File** → **Preferences**, in the Additional boards manager URLs field, paste `https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json`
- In the Boards Manager install **Raspberry Pi Pico/RP2040/RP2350** by Earle F. Philhower, III and choose the **Adafruit QT Py RP2040** board. In the Tools menu leave **USB Stack** set to Pico SDK: the Keyboard library only works that way.
- In the Library Manager install **Adafruit NeoPixel**. Keyboard, on the other hand, already comes with the board package.
- The first upload: hold down **BOOT** while you plug in the USB cable, choose the **UF2 Board** port and upload. From then on the new serial port is enough.

7 Test the wiring

```
// Wiring test: each key has its own color and turns white when you press it.
// The Serial Monitor prints the key number. Here the board is not a keyboard yet.
#include <Adafruit_NeoPixel.h>

const int LED_PIN = 5;                    // RX pin of the QT Py: goes to IN of key 1
const int KEYS[4] = {A3, A2, A1, A0};     // S+ pins of keys 1, 2, 3 and 4
const uint32_t COLORS[4] = {0x0040FF, 0x00FF40, 0xFF8000, 0xC000FF};

Adafruit_NeoPixel leds(4, LED_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  Serial.begin(115200);
  for (int i = 0; i < 4; i++) {
    pinMode(KEYS[i], INPUT_PULLUP);       // internal pull-up: at rest the pin reads HIGH
  }
  leds.begin();
  leds.setBrightness(30);                 // low light, from 0 to 255
}

void loop() {
  for (int i = 0; i < 4; i++) {
    bool pressed = digitalRead(KEYS[i]) == LOW;    // pressed: the pin goes to GND
    leds.setPixelColor(i, pressed ? 0xFFFFFF : COLORS[i]);
    if (pressed) {
      Serial.print("Key ");
      Serial.println(i + 1);
    }
  }
  leds.show();
  delay(100);
}
```

This first sketch leaves the computer's keyboard alone: it lights the LEDs and prints the key you press on the Serial Monitor (115200 baud). `INPUT_PULLUP` keeps the pin high until the switch pulls it to GND (guide no. 20).

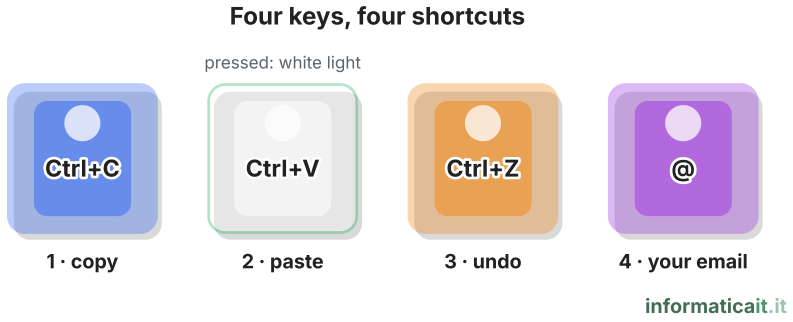
✓ What you should see

The four keys light up blue, green, orange and purple, from the left. The one you press turns white and the Serial Monitor prints “Key 1”, “Key 2”...

If it doesn't work

- No LED lights up: check the wire from RX to IN of key 1 and the + and – rails. Remember that 3V goes to the red rail.
- Only the first keys light up: the chain breaks where the wire from OUT to IN of the next key is missing.
- A key does not respond: S+ and S– are swapped (remember the diode), or the switch is not pushed all the way in.
- No port to upload to: hold down BOOT, plug the USB cable in again and choose UF2 Board.

8 Now it becomes a keyboard




```
// Four-key macro keyboard: QT Py RP2040 and four Adafruit NeoKeys.
// The computer sees it as a USB keyboard and each key sends a shortcut.
#include <Adafruit_NeoPixel.h>
#include <Keyboard.h>

const int LED_PIN = 5;                    // RX pin of the QT Py: goes to IN of key 1
const int KEYS[4] = {A3, A2, A1, A0};     // S+ pins of keys 1, 2, 3 and 4
const uint32_t COLORS[4] = {0x0040FF, 0x00FF40, 0xFF8000, 0xC000FF};

Adafruit_NeoPixel leds(4, LED_PIN, NEO_GRB + NEO_KHZ800);
bool down[4];                             // is the key down?
unsigned long changed[4];                 // when it last changed (ms)

void setup() {
  for (int i = 0; i < 4; i++) {
    pinMode(KEYS[i], INPUT_PULLUP);
  }
  leds.begin();
  leds.setBrightness(30);
  Keyboard.begin(KeyboardLayout_en_US);   // the computer uses the US keyboard layout
}

// Ctrl plus a letter, as if you pressed them yourself with two fingers
void shortcut(char letter) {
```

```
  Keyboard.press(KEY_LEFT_CTRL);          // on the Mac, instead of Ctrl: KEY_LEFT_GUI
  Keyboard.press(letter);
  delay(20);
  Keyboard.releaseAll();
}

void macro(int key) {
  if (key == 0) shortcut('c');            // copy
  if (key == 1) shortcut('v');            // paste
  if (key == 2) shortcut('z');            // undo
  if (key == 3) Keyboard.print("name@example.com");   // write your email here
}

void loop() {
  for (int i = 0; i < 4; i++) {
    bool now = digitalRead(KEYS[i]) == LOW;
    if (now != down[i] && millis() - changed[i] > 30) {  // 30 ms against bounce
      down[i] = now;
      changed[i] = millis();
      if (now) macro(i);                  // only once, when you press it
    }
    leds.setPixelColor(i, down[i] ? 0xFFFFFF : COLORS[i]);
  }
  leds.show();
  delay(5);
}
```

- `Keyboard.begin(KeyboardLayout_en_US)` tells the library that the computer uses the US keyboard layout, which is also the library default: that way the characters typed with `Keyboard.print`, such as the @, come out right. If your computer uses another layout, write its name here: the library also has, for example, `KeyboardLayout_de_DE`, `KeyboardLayout_fr_FR` and `KeyboardLayout_it_IT`.
- `Keyboard.press` presses a key and holds it down, `Keyboard.releaseAll` lets them all go: Ctrl plus a letter makes the shortcut.
- The macro runs only once, when you press the key; the 30 ms wait stops the switch from bouncing.

✓ What you should see
Select some text, press key 1, then key 2: the text is copied and pasted. Key 3 undoes, key 4 types the email address you put in the sketch.

Warning

- When a board acts as a keyboard, it controls the computer: the Arduino documentation recommends always making sure you have control before you use `Keyboard.print`. Here the text is sent only when you press key 4, never on its own in the `loop`.

If it doesn't work

- Compile error on Keyboard: Tools → USB Stack is not set to Pico SDK.
- Wrong characters come out: the computer does not use the US layout. Change the layout in the `Keyboard.begin` line.

9 On the Mac and beyond

- On the Mac the same shortcuts use Cmd: instead of `KEY_LEFT_CTRL` write `KEY_LEFT_GUI`, which on Windows is the key with the logo.
- The library only sends characters from the ASCII table: no accented letters in `Keyboard.print`.
- Finally, put on the keycaps. If the switch comes out with the keycap when you pull it off, a dab of hot glue, as Adafruit suggests, holds it in place.

Try it yourself

- Change the colors in `COLORS` and the brightness.
- Put the save shortcut, Ctrl+S, on key 3.
- Make key 4 type your signature, using `Keyboard.println` to start a new line.

✓ What you learned

- the QT Py RP2040 can present itself to the computer as a USB keyboard;
- a switch with a diode in series must be connected the right way round: S+ to the pin, S− to GND;
- the NeoPixels of the NeoKeys are connected in a chain, OUT of one into IN of the next.

Sources: Adafruit, NeoKey Socket Breakout with NeoPixel for MX and CHOC Key Switches (4978): product page, guide (Pinouts, Arduino) and Eagle file “Adafruit NeoKey Breakout” · Adafruit, QT Py RP2040 (4900): product page, guide “Adafruit QT Py RP2040” (Pinouts, Arduino IDE Setup, Arduino Usage) and Eagle file; product pages of the Kailh 4955 and 4954 switches and of the 4956 and 5068 keycaps · Earle F. Philhower, III, Arduino-Pico 6.3.1 documentation (Installation, USB); adafruit_qtpy variant · Arduino, Keyboard library reference: begin(), press(), releaseAll(), print(), modifiers · Adafruit, “NeoPixel Überguide”, Arduino Library Use; Adafruit NeoPixel library · Microsoft, Keyboard shortcuts in Windows; Apple, Mac keyboard shortcuts · informaticait guides no. 20, 48 and 70