

That's My Type: Handwiring a keyboard

...

By: Cory Palmer

What is a Hand Wired Keyboard?

A hand wired keyboard is the same looking from the outside but on the inside there is just wires put in by hand instead of wire traces printed on a PCB.

Hand wiring allows for true customisation without ordering a specialised PCB.

This allows for you to 3D print your own keyboard case.

Why you want a hand wired keyboard

In many cases and for many people, market grade keyboards available at most retailers are adequate. In some cases however, it is better to customize a keyboard through hand wiring. For my project, I decided to hand wire a keyboard. While this is perfectly fine, I wanted to be able to expand on the possibilities that a keyboard can do. While I was looking for another keyboard to use on my computer I found someone that had put a solenoid (an electromagnet coil) in to a keyboard to have a similar sound to a typewriter.

There is a keyboard that already had support for this solenoid, the ibm beamspring, but this keyboard is old and collectors want it making it cost hundreds of dollars.

Why you want a hand wired keyboard cont.

Wanting to have a cheaper option I found that you can build a keyboard from scratch with the help of a microcontroller. Although there were minimal guides on how to hook up a solenoid up to a keyboard, I found some information on how to hook it up through the [QMK\(Quantum Mechanical Keyboard\) tutorial](#)¹.

First steps

I already picked up a mechanical keyboard from a discount store, although this keyboard worked perfectly fine, I wanted to connect the solenoid to this keyboard later on.

The keyboard I used is the E-Element z88.

Since the controller for this prebuilt keyboard would not natively support the QMK software I had to remove the PCB(printed circuit board) from this keyboard.

My next steps were to gather a microcontroller capable of controlling all the key switches; for this I will be using a teensy 2.0 microcontroller.

You can use a small list of microcontrollers but the teensy 2.0 had just enough pins outs on it to contain the keyboard matrix.

Second steps

We now got most of the shopping done now we get to the boring part of putting together the keyboard matrix.

The keyboard matrix is used such that you can have a 10 x 10 grid of keys and only use 20 pins on a microcontroller instead of 100 (one for each key).

Each key will need a diode connected to it for voltage to give the board exactly what key was pressed. For this I used the 1N4148 switching diode.

Second steps

Before starting to wire everything together you will need to make a plan of how you will wire everything together. I used two websites for help with this since this is my first time wiring a keyboard by hand.

I First used a keyboard layout editor² to make a virtual layout that matched my physical keyboard.

I then was able to port the raw data from the layout editor² over to a keyboard firmware editor³. Although the keyboard firmware editor is able to create the code needed to fully program a keyboard it doesn't have any other features to support external devices such as a solenoid.

Keyboard layout editor home page

[keyboard-layout-editor.com](#) [Preset](#) [Color Swatches](#) [Character Picker](#) [Options](#) [Permalink](#) [Sign in with GitHub](#)

[Add Key](#) [Delete Keys](#) [Undo](#) [Redo](#) [Cut](#) [Copy](#) [Paste](#) [Download](#) [Save](#)

Num Lock	/	*	-
7	8	9	+
Home	↑	PgUp	
4	5	6	
←		→	
1	2	3	Enter
End	↓	PgDn	
0		.	
Ins		Del	

Getting Started with Keyboard-Layout-Editor.com

Keyboard-layout-editor.com is a web application that enables the editing of keyboard-layouts, i.e., the position and appearance of each physical key.

Start by exploring the presets and samples from the menu-bar to give you an idea of the possibilities. Once you are ready to start designing your own keyboard, just load one of the presets and start customizing it! Some tips:

- The selected keys can be modified on the *Properties* tab.
- The *Keyboard Properties* tab lets you edit the keyboard background and keyboard metadata.
- The *Custom Styles* tab lets you write advanced CSS styling rules.
- Don't forget the *Color Swatches* and *Character Picker* menu items! These give you easy access to colors and symbol characters, respectively.
- There are a lot of available keyboard shortcuts; press '?' or 'F1' to see a list.

When you're ready to save your layout, simply 'Sign In' with your [GitHub](#) account and click the Save button. Your layout will be saved in a [GitHub Gist](#).

Have fun!

[Properties](#) [Keyboard Properties](#) [Custom Styles](#) [Raw data](#) [Summary](#) [Tools](#)

Top Legend:

	x		x		x		x

Center Legend:

	x		x		x		x

Bottom Legend:

	x		x		x		x

Front Legend:

	x		x		x		x

Legend Size:

Legend Color: [X](#)

Key Color: [X](#)

Width: / [X](#)

Height: / [X](#)

X: +

Y: +

Rotation: ° ,


Profile / Row:

Switch: Mount N/A Brand N/A Switch N/A

Keyboard layout editor with my keyboard layout

keyboard-layout-editor.com Preset Color Swatches Character Picker Options Permalink Sign in with GitHub

Add Key Delete Keys Undo Redo Cut Copy Paste Download Save



Properties Keyboard Properties Custom Styles </> Raw data Summary Tools

```
1 ["esc",{x:1},"f1","f2","f3","f4",{x:0.75},"f5","f6","f7","f8",{x:0.5},"f9","f10","f11","f12"],
2 [{y:0.25},"~","1","2","3","4","5","6","7","8","9","0","-","=",{w:2},"backspace",{x:0.25},"home"],
3 [{w:1.5},"tab","q","w","e","r","t","y","u","i","o","p","[",""]",{w:1.5},"\\","{x:0.25},"end"],
4 [{w:1.75},"caps\n\n\n\n\n\n\n\nlock","a","s","d","f","g","h","j","k","l",";","'",{w:2.25},"enter",{x:0.25},"PgUp"],
5 [{w:2.25},"LShift","z","x","c","v","b","n","m",".,","/","{w:1.5},"RShift",{x:0.25},"up",{x:0.25},"PgDn"],
6 [{w:1.25},"Rctrl",{w:1.25},"RGUI",{w:1.25},"Lalt",{w:6.25},"space",{w:1.25},"MO_FL",{w:1.25},"Ralt",{x:0.25},"left",{x:0.25},"down",{x:0.25},
  ,"right"]
7
```

Keyboard firmware editor

Keyboard Firmware Builder

Upload Keyboard Firmware Builder configuration

Upload

Or import from keyboard-layout-editor.com

Paste layout here...

Import

Or choose a preset layout

GH60 (ANSI)

GH60 Satan (ANSI)

Alps64 (AEK)

MiniVan (Standard)

MiniVan (Arrow)

1Up RGB Custom PCB (Full)

1Up RGB Custom PCB (Standard)

1Up RGB HSE PCB (Standard)

Keyboard firmware editor with my keyboard layout.

Keyboard Firmware Builder

Board Size Flip ☐

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

0
1
2
3
4
5

WIRING PINS KEYMAP MACROS QUANTUM SETTINGS COMPILE

Change the number of rows and columns in the matrix.

Rows 6
Columns 16

Specify the diode direction.

Column to Row

Second steps

Following handwriting guides provided by the QMK tutorial¹, I personally followed the short lengths of wire guide since I want good insulation between wires and don't care about the looks.

Programing steps

This wouldn't be a computer project without something to code, all we did so far was tear apart a good keyboard and reduced it to a shell with wires and switches.

We need to program the microcontroller, although this is possible with arduino software it would only work on select machines.

QMK is its own open source program to control devices such as a keyboard and be recognized on all computers.

Since it is open source I can make anything in this program without buying special software and there is a community that is very helpful when it comes to problem that you encounter.

Programing steps cont.

I suggest following the QMK tutorial¹ since that is what I used. For the people that want to make their own keyboard but don't know how to code properly can utilize the keyboard firmware editor³ since it streamlines the process through an online Graphical user interface (GUI).

Following the QMK tutorial* You will need three things a plain text editor (I used sublime text), QMK toolbox, and a Unix-like environment.

Programing steps cont.

Follow the steps on the qmk tutorial¹ which recommends msys2 as the unix like environment.

While I followed this guide, having slowerer internet made it difficult to progress fast, I decided to create the concept of making the teensy 2.0 board be recognised as a keyboard.

programming

Qmk releases on external text Files so the coding is straightforward and they have predone examples (for prebuilt keyboards that use qmk software).

Below is the keymap I created for my keyboard, although not completed (no external device to activate) I can always come back to this text file and edit it whenever I want.

keyboard layout.c

```
1  #include "kb.h"
2
3  const uint16_t PROGMEM keymaps[][MATRIX_ROWS][MATRIX_COLS] = {
4
5      KEYMAP(
6          KC_ESC, KC_F1, KC_F2, KC_F3, KC_F4, KC_F5, KC_F6, KC_F7, KC_F8, KC_F9, KC_F10, KC_F11, KC_F12,
7          KC_TILD, KC_1, KC_2, KC_3, KC_4, KC_5, KC_6, KC_7, KC_8, KC_9, KC_0, KC_MINS, KC_EQL, KC_BSPC, KC_HOME,
8          KC_TAB, KC_Q, KC_W, KC_E, KC_R, KC_T, KC_Y, KC_U, KC_I, KC_O, KC_P, KC_LBRC, KC_RBRC, KC_BSLS, KC_END,
9          KC_NO, KC_A, KC_S, KC_D, KC_F, KC_G, KC_H, KC_J, KC_K, KC_L, KC_SCLN, KC_QUOT, KC_ENT, KC_PGUP,
10         KC_LSFT, KC_Z, KC_X, KC_C, KC_V, KC_B, KC_N, KC_M, KC_COMM, KC_DOT, KC_SLSH, KC_RSFT, KC_UP, KC_PGDN,
11         KC_RCTL, KC_RGUI, KC_LALT, KC_SPC, MO(1), KC_RALT, KC_LEFT, KC_DOWN, KC_RIGHT),
12
13 |
14     void matrix_init_user(void) {
15     }
16
17     void matrix_scan_user(void) {
18     }
19
20     bool process_record_user(uint16_t keycode, keyrecord_t *record) {
21         return true;
22     }
23
24     void led_set_user(uint8_t usb_led) {
25
26         if (usb_led & (1 << USB_LED_NUM_LOCK)) {
27
28         } else {
29
30         }
31
32         if (usb_led & (1 << USB_LED_CAPS_LOCK)) {
33
34         } else {
35
36         }
37
38         if (usb_led & (1 << USB_LED_SCROLL_LOCK)) {
39
40         } else {
41
42         }
43
44         if (usb_led & (1 << USB_LED_COMPOSE)) {
45
46         } else {
47
48         }
49
50         if (usb_led & (1 << USB_LED_KANA)) {
51
52         } else {
53
54         }
55
56     }
```

Keyboard layout config.h

```
1 #ifndef CONFIG_H
2 #define CONFIG_H
3
4 #include "config_common.h"
5
6 /* USB Device descriptor parameter */
7 #define VENDOR_ID    0xFEED
8 #define PRODUCT_ID    0x6060
9 #define DEVICE_VER    0x0001
10 #define MANUFACTURER  qmkbuilder
11 #define PRODUCT        keyboard
12 #define DESCRIPTION    Keyboard
13
14 /* key matrix size */
15 #define MATRIX_ROWS 6
16 #define MATRIX_COLS 16
17
18 /* key matrix pins */
19 #define MATRIX_ROW_PINS { B0, B1, B2, B3, B4, B5 }
20 #define MATRIX_COL_PINS { B6, B7, C0, C1, C2, C3, C4, C5, C6, C7, D0, D1, D2, D3, D4, D5 }
21 #define UNUSED_PINS
22
23 /* COL2ROW or ROW2COL */
24 #define DIODE_DIRECTION COL2ROW
25
26 /* number of backlight levels */
27
28 #ifdef BACKLIGHT_PIN
29 #define BACKLIGHT_LEVELS 3
30 #endif
31
32 /* Set 0 if debouncing isn't needed */
33 #define DEBOUNCING_DELAY 5
34
35 /* Mechanical locking support. Use KC_LCAP, KC_LNUM or KC_LSCR instead in keymap */
36 #define LOCKING_SUPPORT_ENABLE
37
38 /* Locking resynchronize hack */
39 #define LOCKING_RESYNC_ENABLE
40
41 /* key combination for command */
42 #define IS_COMMAND() ( \
43     keyboard_report->mods == (MOD_BIT(KC_LSHIFT) | MOD_BIT(KC_RSHIFT)) \
44 )
45
46 /* prevent stuck modifiers */
47 #define PREVENT_STUCK_MODIFIERS
48
49
50 #ifdef RGB_DI_PIN
51 #define RGBLIGHT_ANIMATIONS
52 #define RGBLED_NUM 0
53 #define RGBLIGHT_HUE_STEP 8
54 #define RGBLIGHT_SAT_STEP 8
55 #define RGBLIGHT_VAL_STEP 8
56 #endif
57
58 #endif
```

keyboard.h

```
1  #ifndef KB_H
2  #define KB_H
3
4  #include "quantum.h"
5
6  #define KEYMAP( \
7      K000,      K002, K003, K004, K005,      K007, K008, K009, K010, K011, K012, K013, K014,      \
8      K100, K101, K102, K103, K104, K105, K106, K107, K108, K109, K110, K111, K112,      K114, K115, \
9      K200,      K202, K203, K204, K205, K206, K207, K208, K209, K210, K211, K212, K213, K214, K215, \
10     K300,      K302, K303, K304, K305, K306, K307, K308, K309, K310, K311, K312, K313,      K315, \
11     K401, K402, K403, K404, K405, K406, K407, K408, K409, K410, K411,      K413, K414, K415, \
12     K500, K501,      K503,      K506,      K510, K511,      K513, K514, K515 \
13 ) { \
14     { K000, KC_NO, K002, K003, K004, K005, KC_NO, K007, K008, K009, K010, K011, K012, K013, K014, KC_NO }, \
15     { K100, K101, K102, K103, K104, K105, K106, K107, K108, K109, K110, K111, K112, KC_NO, K114, K115 }, \
16     { K200, KC_NO, K202, K203, K204, K205, K206, K207, K208, K209, K210, K211, K212, K213, K214, K215 }, \
17     { K300, KC_NO, K302, K303, K304, K305, K306, K307, K308, K309, K310, K311, K312, K313, KC_NO, K315 }, \
18     { KC_NO, K401, K402, K403, K404, K405, K406, K407, K408, K409, K410, K411, KC_NO, K413, K414, K415 }, \
19     { K500, K501, KC_NO, K503, KC_NO, KC_NO, K506, KC_NO, KC_NO, KC_NO, K510, K511, KC_NO, K513, K514, K515 } \
20 }
21
22 #endif
```

Programing continued

Through the msys2 application you can compile your keymap to make a .hex file.

To do this we by typing in to the console `$ make <keyboard name>: <layout name>`

After having the .hex file we can now switch to flashing the teensy 2.0.

Flashing the microcontroller

The first thing that you want to do is download the qmk toolbox⁴.

Downloading this program is straightforward and downloads like any other program.

After having it downloaded you will want to know where your hex file is located.

You will select this file location under the local file option.

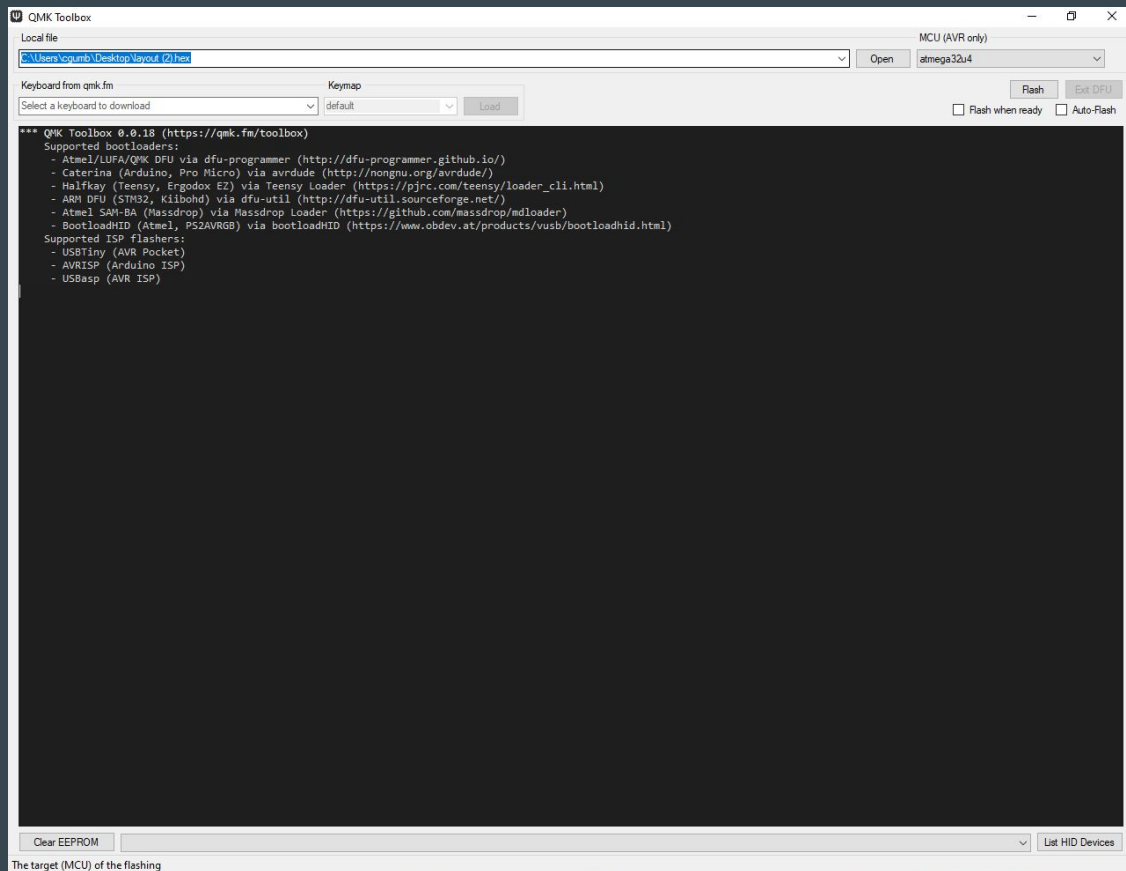
Flashing continued

You will want to select your microcontroller under the MCU dropdown bar.

The teensy is the atmega32u4.

Now that you have the program ready to flash you need to connect the microcontroller to the computer.

QMK Toolbox



Flashing the controller cont.

Most microcontrollers need to be put into a reset mode. The teensy happens to have a physical button that puts it into reset mode, but microcontroller such as the pro micro have to have the reset pin and ground pin shorted out.

Flashing the keyboard

Once you have put the controller into a reset mode you will be able to see the controller show up in the prompt that it was connected.

Last thing you need to do is click the flash button and it will take a little bit to transfer the data.

Wrapping up

Now that the microcontroller has been flashed unplug and replug in to the computer and look under devices to see if your board has been recognized as a keyboard instead of an arduino device.

Although I was not able to complete the hand wiring part of the build I was still able to make the microcontroller be recognised as a keyboard.

Wrapping up

Now you know how to build your own keyboard from the ground up. You can create your own keyboard with a physically different key layout than used currently without the use of computer side software.

There is a handy site called [qmk configurater](#)⁵ to help test your brand new keyboard.

Instead of making your own keyboard from scratch you can buy specific keyboards that use qmk natively.

You might have noticed that the qmk tutorial uses the [clueboard 66%](#) as an example.

QMK configurator to test your keys

If the key
you
pressed
does not
match the
key lit up
you will
have to
change
that in the
code.

The screenshot displays the QMK Configurator API v0.1 web interface. At the top, there's a header with the QMK logo, version information, and status indicators for 'Service' (green) and 'Ready' (green). A 'Settings' link is visible on the right. The main content area shows a keyboard layout with keys labeled with their functions or characters. Above the keyboard, there's a text input field containing the sentence 'Henry Spalding first planted potatoes in Idaho in 1837'. Below the keyboard, there's a section titled 'Keycodes Detected' with a table showing detected keycodes and their corresponding keys. The table has three columns: 'CODE', 'KEY', and 'KEYCODE'. Below the table is a large black rectangular area, likely a placeholder for a video or image.

QMK Configurator API v0.1

Service Ready

< Settings

Henry Spalding first planted potatoes in Idaho in 1837

< BACK RESET

ANSI ISO

Esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 Print Screen Scroll Lock Pause

~ ! @ # \$ % ^ & * () _ = Back Space Insert Home Page Up Num Lock / * -

Tab Q W E R T Y U I O P { } \ Del End Page Down 7 8 9 +

Caps Lock A S D F G H J K L ; ' , Enter 4 5 6

Left Shift Z X C V B N M < > ? / Right Shift Up 1 2 3 Enter

Left Ctrl Left OS Left Alt Space Right Alt Right OS Menu Right Ctrl Left Down Right 0 .

Keycodes Detected

CODE	KEY	KEYCODE
------	-----	---------

Natively supported keyboards

The clueboard 66% is a diy kit that you can purchase, alternatively to the clueboard 66% there is the planck which is a 40% keyboard that offers the same QMK freedom.

Most of the time these keyboards will have a default layout pre installed on it but that doesn't stop people editing their keyboard to be able to do more functions than keys. You can even access these defaults from the keyboard firmware editor³.

Sources

1. <https://beta.docs.qmk.fm/>
2. <http://www.keyboard-layout-editor.com/#/>
3. <https://kbfirmware.com/>
4. https://github.com/qmk/qmk_toolbox
5. <https://config.qmk.fm/#/test>