SMD CHALLENGE: KEYRING EDITION

Development journal

Matthieu Michel
January 27, 2025

1 Table of Contents

1.1 Tables of figures

2 Project overview

2.1 Objective

As a hobbyist electrician, when asked if I solder, I often say, “Yes I solder my boards, and even those little, teeny tiny things you see on Arduinos for example!”, talking about SMD parts of course. And when I stumbled upon the great SMD Challenge by MakersBox on Tindie, I thought to myself “What a great way to show my soldering skills to anyone (and flex a little too)”, until I realized it was a pin and that carrying it around a lot would eventually lead to its demise.

I then had the idea to remake it in a keychain format, I will thus be able to carry it around and show it to everyone, while preserving the components on it, plus it would certainly make for a cool keychain for any hobbyist.

2.2 Scope

The goal of this project is to provide a small, encased PCB forming a keychain, on which the user may reflow the back part and hand solder the front part, which will constitute the main challenge. Upon successful assembly, the five LEDs onboard will flash in pre-programmed sequences from the firmware.

The user will also be able to, by soldering three wires, reprogram the onboard MCU.

2.3 Constraints

The constraints are the following:

- The product shall not exceed 2x2cm ($w \times l$) and 9 mm thickness.
- The BOM shall not exceed €20.
- The product shall have a hole to attach a chain.
- The product should have a protective case to cover components.

2.4 Timeline

This project does not have any time constraints.

3 Planning

3.1 Requirements

3.1.1 Functional requirements

The functional requirements are actions the system does. They specify functions, features or tasks the product must perform. They are often described as clear tasks.

The functional requirements are the following:

- The LEDs on the PCB must be able to be controlled individually.
- The onboard MCU should be able to be reprogrammed.
- The product must be USB-C powered.

3.1.2 Non-functional requirements

Non-functional requirements describe how well the system performs, operates, or adheres to certain constraints. They are often quality attributes or external limitations. They often define standards rather than tasks.

For our project, they are as such:

- The product must not consume no more than 250mA.
- LEDs should not consume no more than 8mA individually.
- The enclosure must withstand minimal scratches.
- The enclosure must be transparent.
- The product must not directly expose the user to lead solder or any other toxic compounds.

3.1.3 Hardware requirements

Hardware requirements are requirements based on needed parts for the project.

We have the following:

- STM32F030F4Px

3.2 Initial design

For the initial design, I thought about a small PCB, with resistors and LEDs of all sizes ranging from 1206 to 01005 (0201 for LEDs) on the front and the STM32, USB-C port and needed circuitry on the back. This way, it leaves the front for the real challenge while the back manages LED control.

Here is the corresponding block diagram:

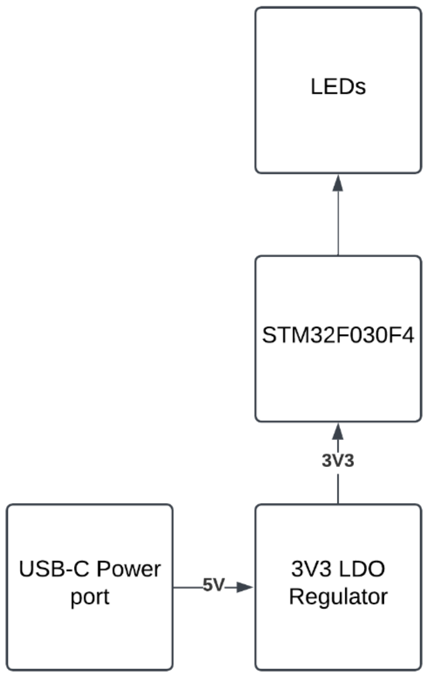

Figure 1 : Block diagram of the project.

Using a full-SMD USB port here will be needed as we don't want the holes ruining the front side. We will also leave three holes, one for the keychain and two for covers that will be screwed on.

Talking about the said cover. After studying all options, including a full case, conformal coating and more, I've settled on a two-parts cover that will sit on the PCB flush with the sides. This way, we will still keep access to the PCB, we will be able to replace the cover when it gets too many scratches, and we only gain thickness, thus making a small size easier to maintain.

3.3 Implementation plan

To develop this project, we will first issue the first prototype which might, if functional, serve as the final product. Once everything has been tested and confirmed working, we will start modeling the cover. If the latter is successful, then the project will be considered complete.

4 Prototypes

4.1 Prototype N°1

4.1.1 Goal

This project being quite simple, the goal of this first prototype is to get everything up and running. It must be able to meet all requirements, except case-related ones.

4.1.2 Design & diagrams

To design it, I first have a USB port to power it, followed by an LDO regulator taking the 5 V and dropping it to 3V3. Do not forget the 5.1 kΩ resistors on the CC pins for USB C-to-C capabilities.

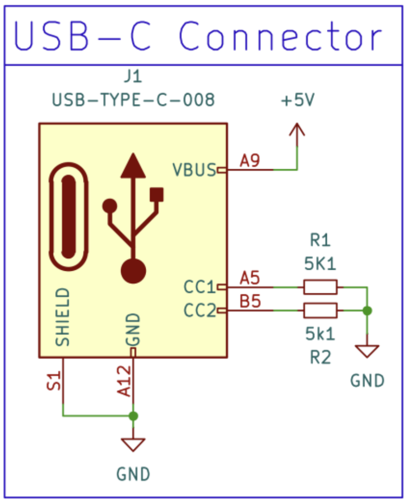

Figure 2: Prototype N°1 – USB-C connector design

The chosen regulator, XC6206P332MR, has a 250 mA current capability, which is more than enough to power our entire system. Two 1 µF capacitors will be attached to its input and output for stability.

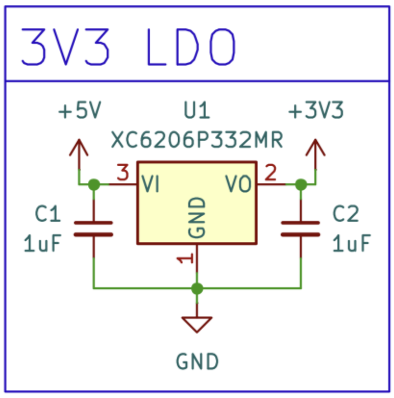

Figure 3: Prototype N°1 – 3V3 LDO Regulator

The 3V3 is used to power a STM32F030F4, chosen for its small package (TSSOP-20) and low cost (< €0.66), making it suitable for our use case. This chip already has a weak pull-up resistor on its NRST pin, no need for an external one, we will just add one on the BOOT0 pin, since no default state is dictated by the microcontroller. Two 100 nF and 10 nF capacitors will be added for decoupling VDD and VDDA.

We will attach to it an 8MHz crystal as HSE. The crystal load capacitance is calculated using the following formula: $C_{load} = \left(C_L - C_{stray}\right) \times 2$ with C_{load} the capacitance required for each capacitor in pF, C_L the load capacitance given in the crystal's datasheet in pF and C_{stray} the parasitic capacitance from the board itself, usually estimated to be around 2 to 5 pF. With $C_L = 10\,pF$ and $C_{stray} = 2\,pF$, we get: $C_{load} = (10 - 2) \times 2 = 16\,pF$ per capacitor. We will choose C0G ceramic capacitors to ensure stable capacitance across a wide temperature range.

We then have the following setup for our microcontroller:

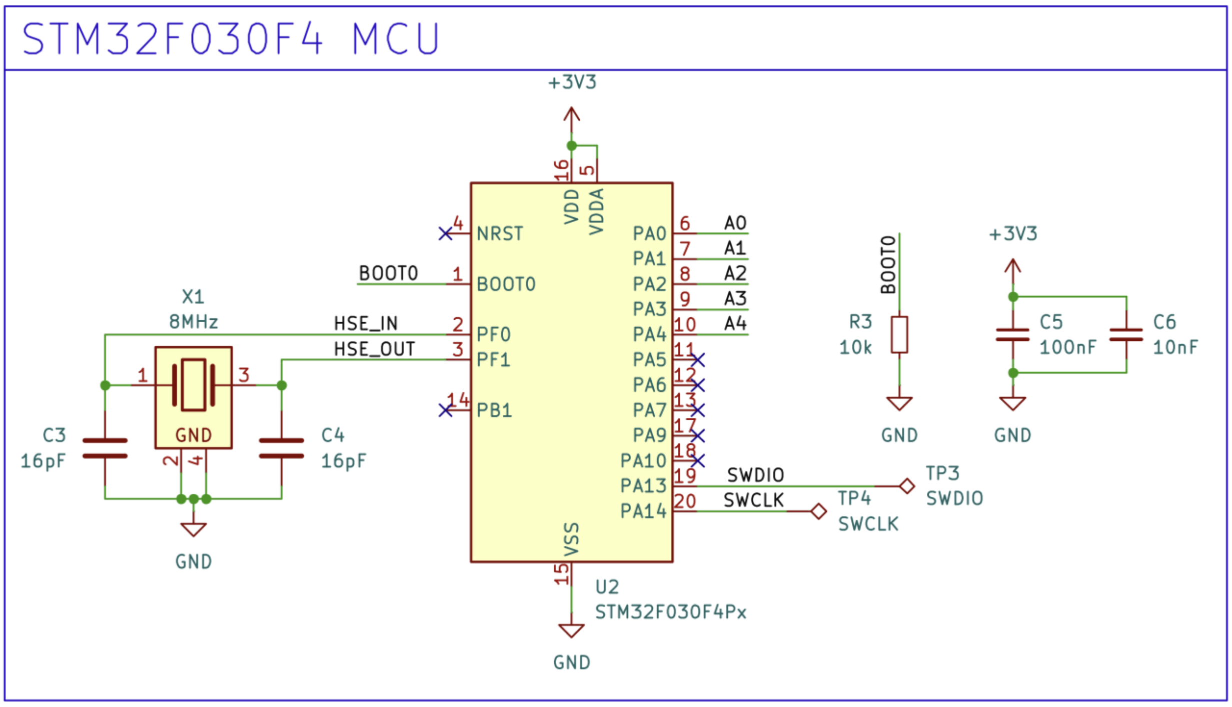

Figure 4: Prototype N°1 – MCU setup.

Finally, we have the resistors and the LEDs, each LED is assigned a series resistor. Only the 01005 resistor is single, only linked to two test points, since 01005 LEDs aren't a thing. LEDs will be colored as red, orange, yellow, ice blue (cyan), blue. This should give us a nice red-to-blue gradient, and it should fit well with the blue solder mask.

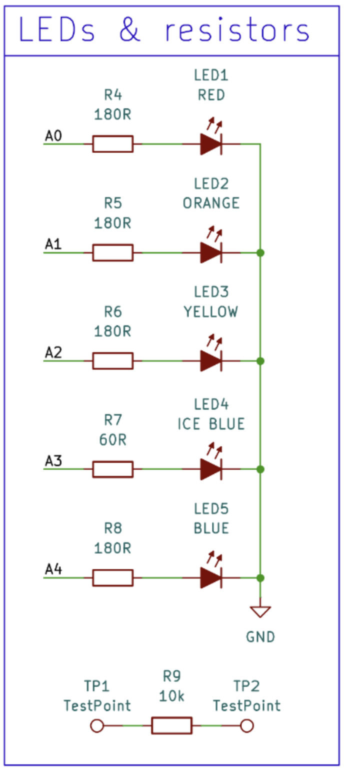

Figure 5: Prototype N°1 – Front LEDs and resistors.

And last, we add three M2 mounting holes for later use.

Full schematic below:

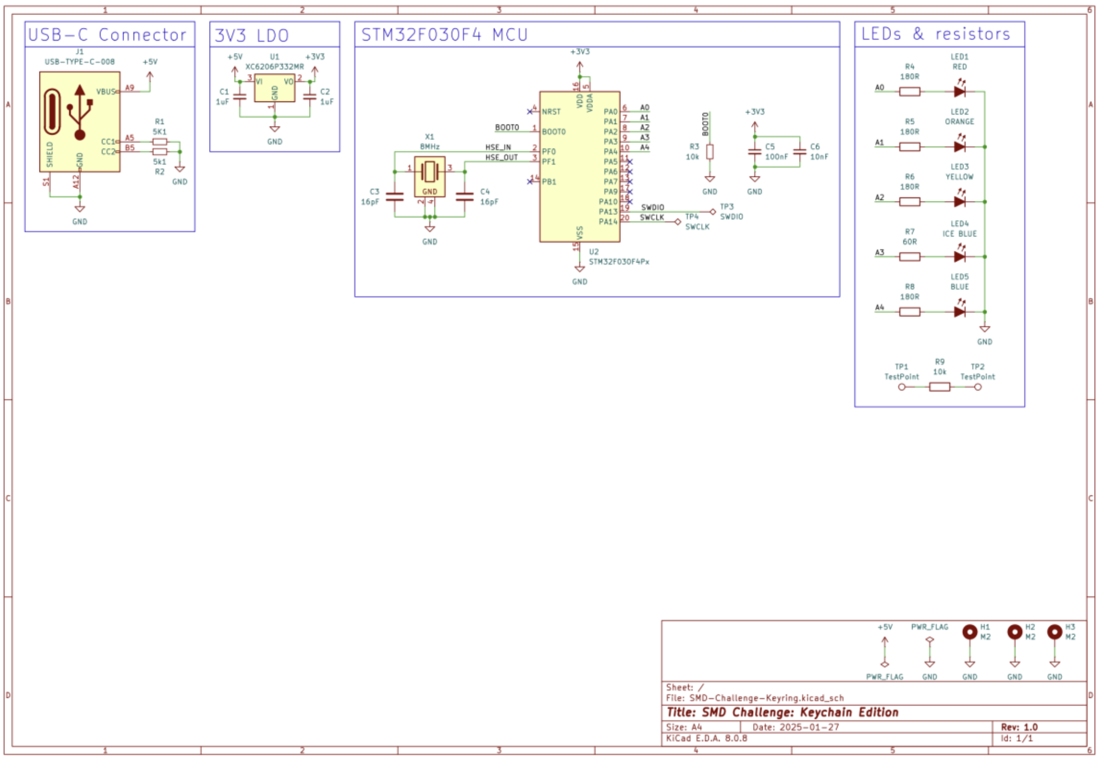

Figure 6: Prototype N°1 – Full schematic.

4.1.3 Assembly notes

I ordered two sets of PCBs, one with blue solder mask and HASL finish, and one green solder mask and ENIG finish.

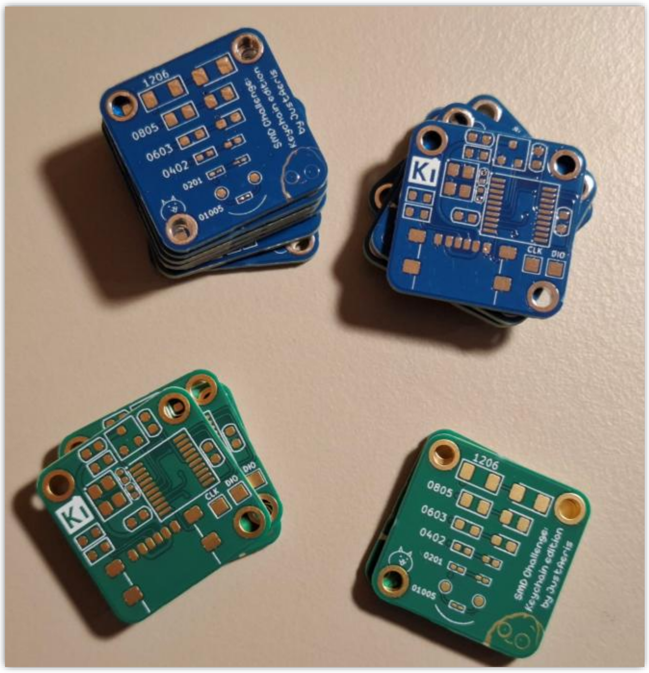

Figure 7: Prototype N°1 – Pictures of the PCBs.

As expected, the assembly of this board was a rather consequent challenge. The back with all the programming circuit was pasted with leaded paste, then reflowed and of course the front was hand soldered.

Here is a picture of the back side:

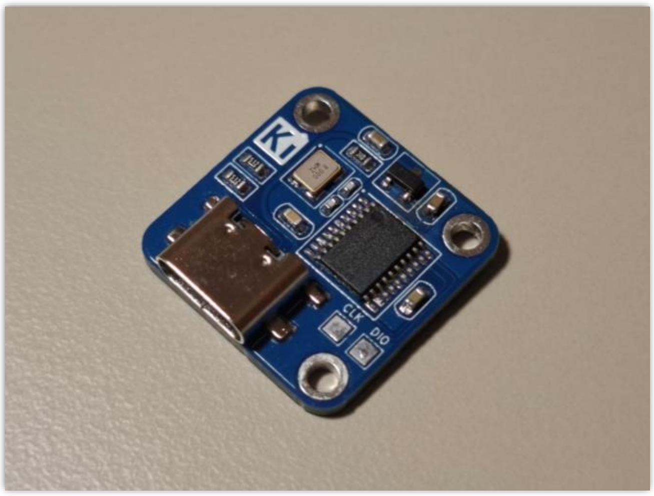

Figure 8: Prototype N°1 – Assembled back side of the PCB.

It was easy for the most part, until 0201. With 0201, I had to make use of greater magnification and patience. The LED was especially tricky, due to having pads underneath itself. But the greatest challenge was definitely the 01005 resistor. I lost half a day on it because I accidentally ripped off one of the pads, the solder couldn't adhere to the resistor. When I switched to a new one, it worked on the second try.

Here is a picture of the complete front assembly:

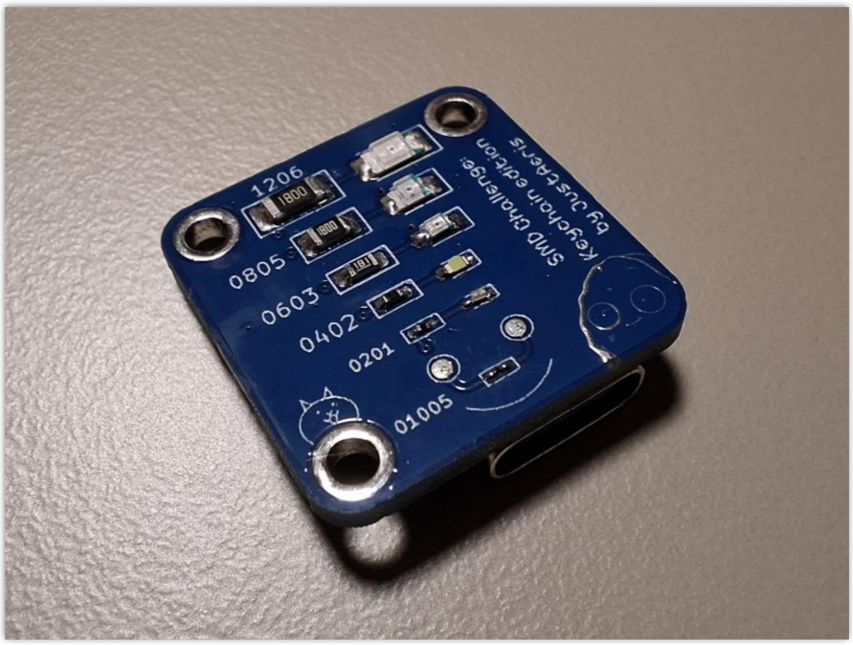

Figure 9: Prototype N°1 – Complete front side assembly.

And here is a picture of the 01005 resistor with my phone's 10x zoom plus a 23x magnification lens in front:

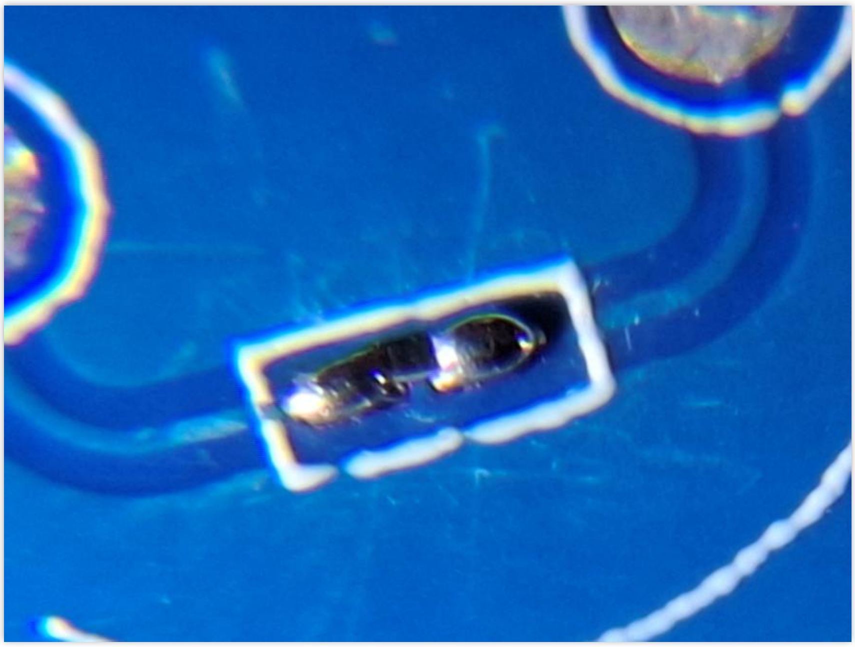

Figure 10: Prototype N°1 – Soldered 01005 resistor.

Soldering is in fact pretty mild when seen this close but I doubt I'd been able to make it better, mainly because this resistor is **0.4 × 0.2 mm wide** (or if you prefer 400 × 200 µm).

4.1.4 Testing

After painfully loosing again multiple hours on the fact that you actually need the ST-Link's VCC to be connected to the board's VCC, I managed to upload some code to the board and get it working. See the video below:

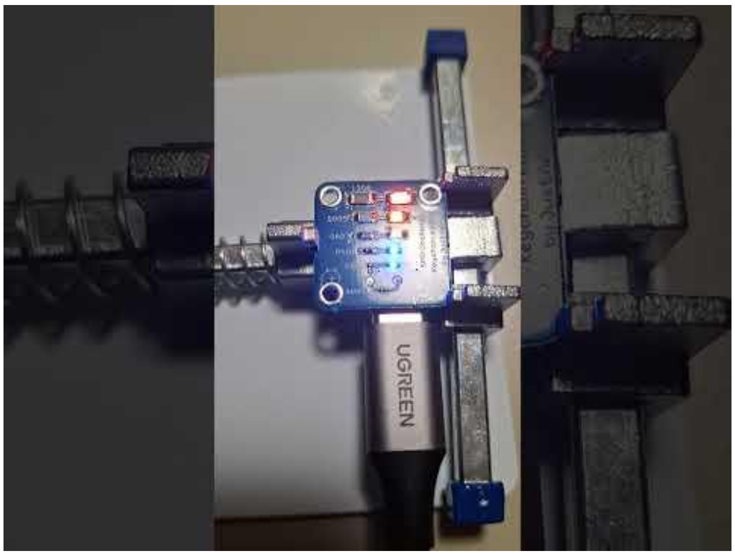

Figure 11: Prototype N°1 – Demo (clickable video).

We can also verify the good connections of out 01005 resistor by probing the two test points with an ohmmeter:

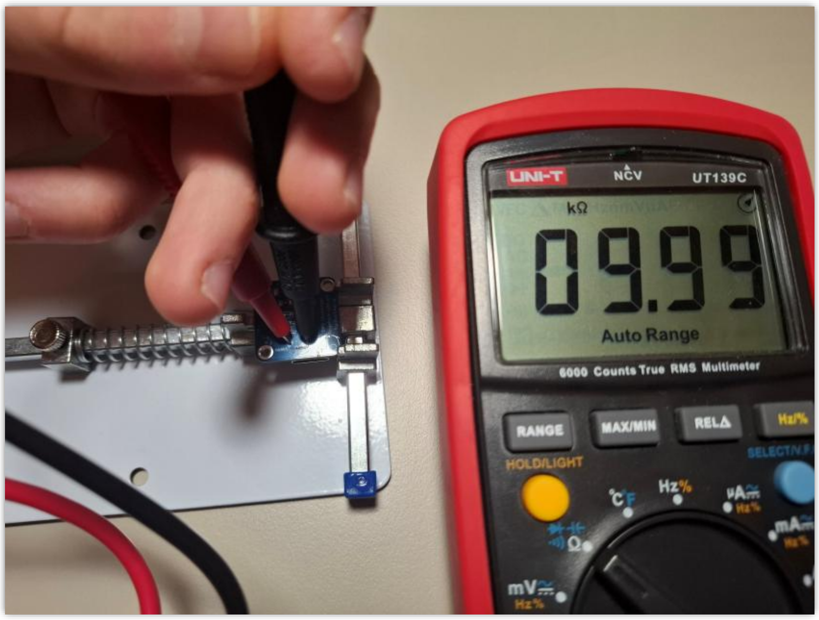

Figure 12: Prototype N°1 – Testing setup for the 01005 resistor.

Showing 10 kΩ means that we successfully made a connection between the two pads with our resistor!

4.1.5 Software implementation

The software-side of things is rather simple. Except for the two blue LEDs. Since they were way to bright, I had to dim them otherwise it'd just singe anyone's eyeballs.

Fixing the blue 0201 LED was pretty easy considering I inadvertently put it on timer 14's PWM channel. Configuring accordingly the timer for 10kHz and setting up a small wrapper function makes this quite easy:

```
#define BLUE_LED_BRIGHTNESS 20
// ...
// Function to turn the blue LED on or off
void Blue_LED_Set(uint8_t state) {
        if (state) {
                // Turn on - set PWM to defined brightness
                __HAL_TIM_SET_COMPARE(&htim14, TIM_CHANNEL_1, BLUE_LED_BRIGHTNESS);
        } else {
                // Turn off - set PWM to 0
                __HAL_TIM_SET_COMPARE(&htim14, TIM_CHANNEL_1, 0);
        }
}
```

Code 1: Prototype N°1 – Blue LED PWM wrapper.

For the 0402 cyan LED, more trickery was required. Since it was just attached to a regular GPIO, no timer whatsoever, I had to build a software PWM implementation using a software micro-delay.

```
#define CYAN_LED_BRIGHTNESS 5
// ...
// Short microsecond delay for software PWM
void Software_PWM_Delay(void) {
        for (volatile uint16_t i = 0; i < 10; i++);
}
// Software PWM for the cyan LED - apply during a delay interval
void Cyan_LED_Dimmed_Delay(uint32_t ms) {
        uint32_t end_time = HAL_GetTick() + ms;

        while (HAL_GetTick() < end_time) {
                // Turn on for CYAN_LED_BRIGHTNESS% of the time
                HAL_GPIO_WritePin(LED_PORT, CYAN_LED_PIN, GPIO_PIN_SET);
                for (uint8_t i = 0; i < CYAN_LED_BRIGHTNESS; i++) {
                        Software_PWM_Delay();
                }

                // Turn off for (100-CYAN_LED_BRIGHTNESS)% of the time
                HAL_GPIO_WritePin(LED_PORT, CYAN_LED_PIN, GPIO_PIN_RESET);
                for (uint8_t i = 0; i < (100 - CYAN_LED_BRIGHTNESS); i++) {
                        Software_PWM_Delay();
                }
        }
}

// Function to control the cyan LED - executes only one cycle
void Cyan_LED_Set(uint8_t state) {
        if (state) {
                HAL_GPIO_WritePin(LED_PORT, CYAN_LED_PIN, GPIO_PIN_SET);
                for (uint8_t i = 0; i < CYAN_LED_BRIGHTNESS; i++) {
                        Software_PWM_Delay();
                }

                HAL_GPIO_WritePin(LED_PORT, CYAN_LED_PIN, GPIO_PIN_RESET);
                for (uint8_t i = 0; i < (100 - CYAN_LED_BRIGHTNESS); i++) {
                        Software_PWM_Delay();
                }
        } else {
                HAL_GPIO_WritePin(LED_PORT, CYAN_LED_PIN, GPIO_PIN_RESET);
        }
}
```

Code 2: Prototype N°1 – Software-side PWM implementation for the cyan LED.

The rest of the code is straightforward, it just consists of GPIO writes. The six following animations were implemented:

- Chase effect (3×)
- Blink all LEDs (5×)
- Alternating pattern, red, yellow and blue vs. orange and cyan (5×)
- Fill up and empty (once)
- Knight rider (3×)
- Flash each color (3× each)

4.1.6 Review

Overall, I'd call this a success. The board wasn't too complex to design in the first place, and I managed to assemble it almost on the first try, although I didn't know it'd require two full afternoons from me to complete the assembly.

4.1.7 Improvement and changes needed

4.1.7.1 Incorrect series resistors

The blue and cyan LEDs are way too bright, wrong values were used in the calculation of their series resistor. **If it were to be assembled again** and for ease of use, the values would need to be changed. We'll choose a forward current of 1 mA, this should be bright enough for our application, especially for blue-tinted LEDs.

For the cyan LED, and using the graph from the datasheet:

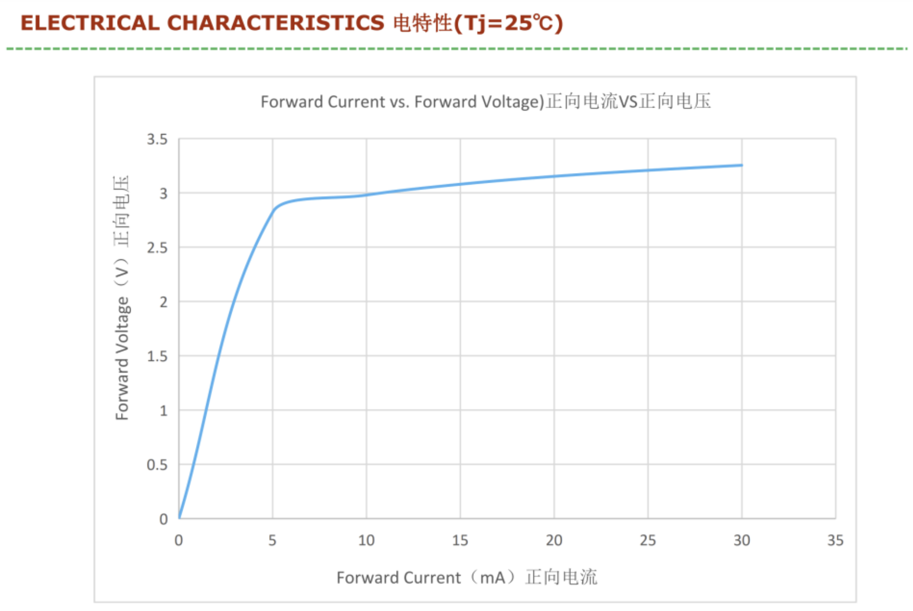

Figure 13: Forward current vs. forward voltage of the cyan LED.

We can estimate the forward voltage at 1 mA to be roughly 2.2 V. Using the following formula:

$$R = \frac{V_{Supply} - V_f}{I_f}$$

$$R_7 = \frac{3.3 - 2.2}{0.001} = \frac{1.1}{0.001} = 1100\ \Omega$$

The use of a standard E24 1.1 kΩ resistor for R7 is possible and should be the right fit.

For the blue LED, the chart is harder to read but we can estimate $V_f = 2.6\,V$ at 1 mA. This also correlates with the minimum forward voltage specified in the datasheet of the same value.

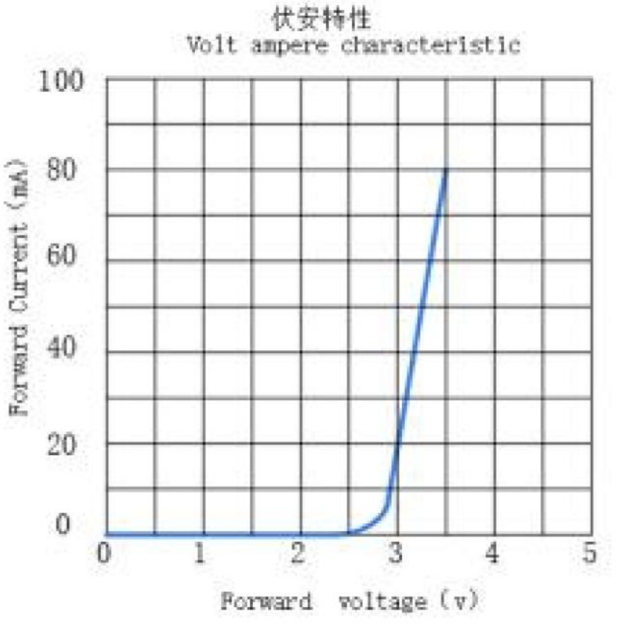

Figure 14: Forward voltage vs. forward current of the blue LED.

We then again calculate the new series resistor value:

$$R_8 = \frac{3.3 - 2.6}{0.001} = 700\ \Omega$$

We can thus replace R8 with a standard E24 680 Ω resistor.

Those modifications should greatly decrease the brightness of the two blue LEDs, as to not have to resort to trickeries to dim them via the code.

4.1.7.2 Missing VCC pad

Since the VCC connection is required to program the chip, we may want to add a VCC pad similar to the SWDIO and SWCLK ones to ease connection between the ST-Link and the board. An additional ground pad wouldn't be too much too, although we still can use one of mounting pads of the USB-C connector, though painful due to the connector sinking the heat away.

5 3D modeling and enclosure design

5.1 Objectives

The goal of the case here is to protect the components of the board from scratches, keys, and external shocks, while keeping said parts visible. The cover shall not apply destructive operations on the circuit, such as glue or other permanent modification, thus making the cover entirely separate and replaceable. The cover should also protect the user from toxic materials such as lead solder, if used in assembly.

5.2 Initial concept

To realize this, I first thought about conformal coating, applying a layer of coating sounded good as it would have kept the product thin, with high transparency. However, conformal coating is oriented towards electrical insulation as well as environmental protection against temperature, humidity, and other factors. It is not made to protect against scratches and shocks.

I then turned myself to resins such as epoxy resin or UV-cured resin. However, resin might get scratched easily, plus it is not convenient for the user to encapsulate the circuit in resin. Finally, resin would be hard to remove without destroying the circuit.

Obviously, the next step was to consider 3D printing a case/cover. First thought was a press-fit case but then, how do you remove a circuit from a press-fit case? You do not. A full case then? Adds thickness on the sides, and it might make the product too thick.

I then looked back on conformal coating and thought about making a cover, flush with the sides, which would function as a coating layer, except it is rigid. Thin enough, it might give superior results. Especially if the nuts and screw heads are made flush with the top and bottom. That is the design we'll settle on.

5.3 Design process

The case consists of two parts sitting on the PCB, being flush with the sides. We'll define the top at the sides with resistors and LEDs and the bottom the side with the programming circuitry.

The top cover consists of a 1.2 mm thick plate and 1.8 mm wide walls, with holes for mounting screws heads. This allows to get the screws flush with the cover as to not have little bumps that might get stuck on other things. Fillets were used on sharp corners to comply with the minimum wall thickness requirement, which is 0.8 mm. Since we're using M2 screws, we have a 2.4 mm hole and a head diameter of 4.2 mm (including clearance). We're using EDDM-M2-L6 screws from JLCMC for their flat head, allowing us to reduce further the thickness of the total product.

We finally have a 2.6 mm thick cover, with 1.4 mm of room for our top parts.

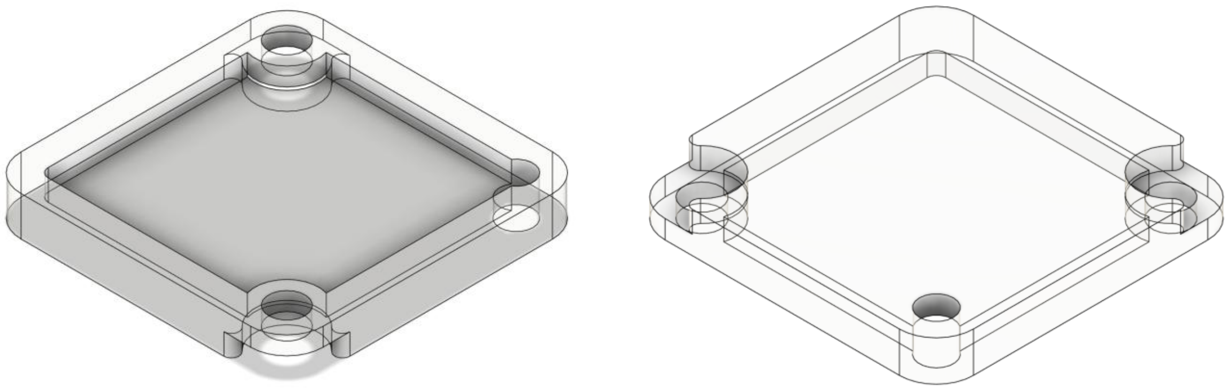

Figure 15: Front and back views of the top cover

The bottom cover was tricker, noticeably due to the USB-C port and several parts being too close to the edges. For the first matter, we simply did a cutout in the main part, allowing the USB-C port to sit flush with the rest of the cover. For the parts, and pads, that were too close to the edges, I did use slopes in the goal to keep the sides filled while allowing parts and solder to fit under the walls. We also made sure to use thin M2 hex nuts, EMLA-S1W-B1L1-M2 from JLCMC, to keep saving on thickness.

Here is the result:

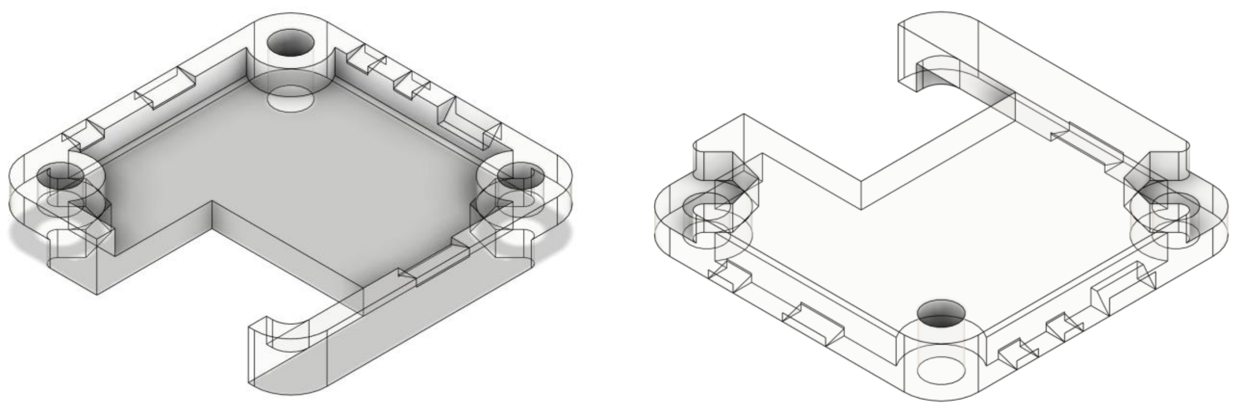

Figure 16: Font and back views of the bottom cover

Those two pieces combined with the board yield a total of 7.5 mm of thickness, keeping our final product rather thin and not too bulky for something that’s going to hang around with your keys.

Since the cover sits flush with the PCB, it keeps the 20 × 20 mm dimensions of the latter.

Here are the assembled 3D models:

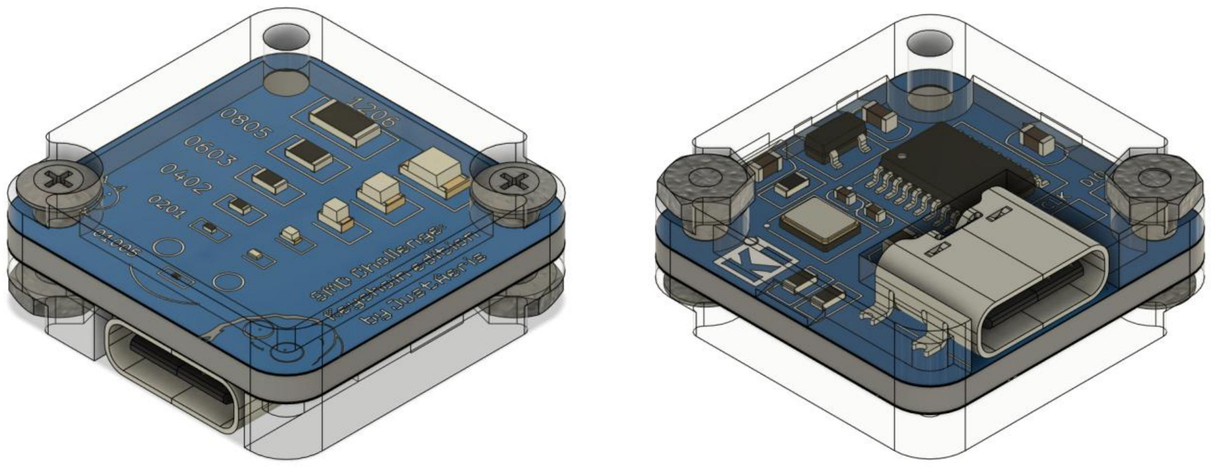

Figure 17: Full 3D view of the product

5.4 Material selection

For this project, as I'm using JLC3DP services to get my parts printed, I'll make use of their 8001 resin which has (citing JLC3DP) "excellent strength and toughness, high precision, and good dimensional stability" and aside from that, it is obviously transparent. I already ordered pieces using this material before and it should be strong enough to achieve what I want.

5.5 3D printing notes

5.5.1 Constraints

For this assembly, all constraints have been respected, they are as follow for SLA (constraints that do not apply to our design will be omitted):

- Min. design size (mm): 5x5x5 OR 10x2x2
- Wall thickness: 0.8 mm
- Model clearance: 0.2 mm
- Holes design: for an aperture of 2.0 mm, the height must be at least 2.0mm
- Printing tolerances:
  - Model tolerances: ± 0.2 mm (within 100mm), ± 0.3 % (above 100mm)
  - Hole tolerances: ± 0.3 mm

5.5.2 Surface finish

For this print, we need a transparent finish, this is why oil spraying finish was selected.

5.6 Assembly fit & review

Perfect fit! It exactly looks like the 3D renders.

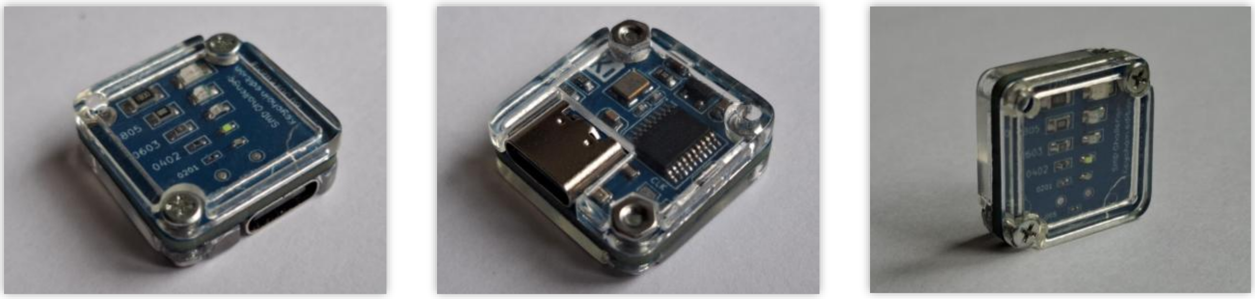

Figure 18: Picture of the case on the board

It only has a major downside which needs investigations: the resin is absolutely not scratch proof. This could be fixed with the use of varnishes like a 2K clear coat, which can form a hard, thin, protective layer on our 3D prints. This is going to be tested.

5.7 Improvements for future cases

Nothing to improve in terms of modeling.

6 Final version

6.1 Final design

The final design successfully meets all requirements: 20 × 20 mm footprint, 7.5 mm total thickness, and under €20 BOM cost. The transparent two-piece enclosure protects components while maintaining visibility, with flush M2 screws preventing snags. All components from 1206 down to 01005 are functional, and the USB-C connector remains accessible.

6.2 What it does

Upon USB-C power connection, the device cycles through six LED animations: chase effect (3×), blink all (5×), alternating warm/cool colors (5×), fill up and empty (1×), knight rider (3×), and individual color flash (3× each). The animations showcase the color gradient from red to blue while demonstrating the different component sizes. Software PWM dimming keeps the blue LEDs at comfortable brightness levels.

This allows us to appreciate the soldering work.

6.3 Performance evaluations

Power consumption is ~39 mA peak (verified by a USB power meter), well below the 250 mA limit. All LEDs function correctly with proper current limiting. Programming works reliably through SWDIO/SWCLK test points, though a VCC pad would improve convenience. The enclosure fits perfectly but scratches easily during handling.

6.4 Challenges and solutions

The 01005 resistor was the biggest challenge as expected - lost half a day when the first pad ripped. Success required 23× magnification and careful temperature control. The blue LEDs were too bright, solved with PWM dimming rather than PCB rework. Forgot the ST-Link VCC connection initially, costing debugging time. The resin case needs better scratch protection.

7 Conclusion

7.1 Reflection

The project successfully created a portable, protected SMD challenge that fulfills its dual purpose: personal achievement and demonstrating soldering skills to others. Hand-soldering the 01005 resistor was extremely satisfying and pushed the limits of conventional techniques. The keychain format makes it practical for daily carry while showcasing the achievement - perfect for flexing those SMD skills! Software PWM solutions worked well but highlighted the need for better LED characterization during design.

7.2 Future of the project

Future improvements include correcting the blue LED resistor values, adding VCC/GND test pads, and investigating scratch-resistant enclosure treatments. An even more brutal version could feature 008004 components (0.25 × 0.125 mm) and expand beyond resistors to include fine-pitch QFP, QFN, or SOP packages for the ultimate hand-soldering challenge. The concept could expand to educational kits or other miniaturized demonstration platforms. This project demonstrates how advanced prototyping is now accessible to individual makers.

8 Appendices

8.1 Datasheets

- STM32F030F4Px

https://www.lcsc.com/datasheet/lcsc_datasheet_1809301214_STMicroelectronics-STM32F030R8T6_C39105.pdf

- 8MHz crystal (TAXM8M4RDBCCT2T)

https://www.lcsc.com/datasheet/lcsc_datasheet_1912111437_Yajingxin-TAXM8M4RDBCCT2T_C400090.pdf

- 3V3 LDO regulator (XC6206P332MR)

https://www.lcsc.com/datasheet/lcsc_datasheet_2407250919_UMW-Youtai-Semiconductor-Co---Ltd--XC6206P332MR_C347376.pdf

- Cyan 0402 Led (F.0402.00018/P2-0402W6YS2-045T-001)

https://www.lcsc.com/datasheet/lcsc_datasheet_2410121645_TUOZHAN-F-0402-00018-P2-0402W6YS2-045T-001_C7496805.pdf

- Blue 0201 LED (XL-0201UBC)

https://www.lcsc.com/datasheet/lcsc_datasheet_2504101957_XINGLIGHT-XL-0201UBC_C3646921.pdf

8.2 Bill of materials

Designator	Footprint	Qty.	Value	LCSC Part #
C1, C2	0603	2	1uF	C15849
C3, C4	0402	2	16pF	C541434
C5	0603	1	100nF	C1591
C6	0603	1	10nF	C1589
J1	USB_C_Receptacle_GCT_USB4135-GF-A_6P_TopMnt_Horizontal	1	USB-TYPE-C-008	C2927028
LED1	1206	1	RED	C965822
LED2	0805	1	ORANGE	C965813
LED3	0603	1	YELLOW	C965802
LED4	0402	1	ICE BLUE	C7496805
LED5	0201	1	BLUE	C3646921
R1, R2	0603	2	5K1	C25905
R3	0603	1	10k	C25804
R4	1206	1	180R	C17924
R5	0805	1	180R	C25270
R6	0603	1	180R	C22828
R7	0402	1	60R	C38941
R8	0201	1	180R	C138168

R9	01005	1	10k	C270305
U1	SOT-23	1	XC6206P332MR	C347376
U2	TSSOP-20	1	STM32F030F4Px	C89040
X1	3225-4Pin	1	8MHz	C400090

8.3 Sources and references

JLCPCB. (2024, december 26). *3D Printing Design Guideline*. Retrieved from JLC3DP: https://jlc3dp.com/help/article/3D-Printing-Design-Guideline

JLCPCB. (n.d.). *Hex nut Standard/Thin/Thickened Coarse/Fine thread*. Retrieved may 4, 2025, from JLCMC: https://jlcmc.com/product/s/E04/EMLA/FA-%E7%B4%A7%E5%9B%BA%E9%9B%B6%E4%BB%B6-%E8%9E%BA%E6%AF%8D

JLCPCB. (n.d.). *Phillips Ultra Thin Head Screw*. Retrieved may 4, 2025, from JLCMC: https://jlcmc.com/product/s/E02/EDDM/FA-%E7%B4%A7%E5%9B%BA%E9%9B%B6%E4%BB%B6-%E8%9E%BA%E9%92%89

MakersBox. (n.d.). *SMD Challenge from MakersBox on Tindie*. Retrieved from Tindie: https://www.tindie.com/products/makersbox/smd-challenge/