mini eLiXiVy Assembly Guide

Written by minibois, August 2021.

Assembly Guide version: *v1.0*
Assembly Guide license: *CC BY-SA 4.0 (See section [10.0])*
Applicable PCB version(s): *rev.1a / rev.1*
PCB/Schematic source location: https://github.com/minibois/eLiXiVy/

Table of Contents

[0] Introduction

This assembly guide shows how to assemble the eLiXiVy, the full source PCB and schematic of which can be found here: https://github.com/minibois/eLiXiVy
This guide will cover the tool/component prerequisites, testing procedures, assembly steps and flashing of the bootloader and firmware.

The instructions in this guide are written for people who are still a bit new to soldering and electronics projects, but do have some experience under their belt.
This assembly guide is written under the CC BY-SA 4.0 license, full details of which can be found in section [8.0].

[0.1] Warning

Assembling the eLiXiVy requires you to solder components in place. Soldering carries with itself some inherent risks, especially so when working with leaded solder.
Following this guide is at your own risk.
Always follow the necessary and recommended safety procedures, such working in a well-ventilated area and not (directly or indirectly through unwashed hands/equipment) ingest (leaded) solder. Wear protective goggles too.

Some of the soldering tasks in this project (particularly the microcontroller) can be very tough, so this is not recommended as a beginner project.
Only attempt this project at your own risk and if you feel confident in your ability and knowledge of soldering.

If you're a novice at soldering, be sure to consult a soldering guide like this one from EEVBlog: https://www.youtube.com/watch?v=J5Sb21qbpEQ
This video covers the basic tools and techniques. This guide will also cover these, but in lesser detail than this video.

[0.2] General tips:

When working with soldering, there are some general tips tips to keep in mind:
- A soldering iron is a hot piece of equipment, handle it with care and never leave it unattended.
- Wash your hands after soldering
- Avoid touching other stuff during soldering, to not spread around any dangerous substances
- Work in a well ventilated area, preferably with a fume-extractor or fan running
- Don't keep your iron on the board for a long time, to avoid heat damage
- Add flux when reheating solder joints
- It's easier to clean flux when it hasn't solidified, so clean up the flux as soon as possible
- There is flux included in the soldertin, cleaning is recommended after soldering in each component

[1] Prerequisites

To assemble the eLiXiVy, certain tools and components are needed, listed below.

[1.1] Tools

- Soldering iron
    - A station - such as the Hakko FX888D or Ksger T12 - is recommended. Cheap tools can make the job much harder
- Multimeter (needs to have continuity mode at the least)
- Sidecutters/flush cutters (if using THT diodes, more on this in section [4.6.3])
- Screwdriver (PH00 size)
- Tweezers
- Anti-static bracelet
- Fan: to blow away fumes from your face
- Safety goggles (solder can splatter, especially when working with solder wick and using side cutters can be dangerous for your eyes)
- To flash a bootloader (possible optional, see [5.1]):
    - an Arduino UNO
    - 6x 2.56mm jumper wires (male to female)

[1.2] Supplies

- Soldertin (60/40 Sn/Pb in sizes 0.5mm and 0.7mm recommended)
- Isopropyl alcohol (IPA) (70-99% recommended)
- Something to apply the IPA with (cotton swab/toothbrush recommended)
- Solderwick
- Flux

[1.3] Components

The components needed to assemble the eLiXiVy are described on the BOM.
Octoparts link: https://octopart.com/bom-tool/allQRgda
.txt link: https://github.com/minibois/eLiXiVy/blob/master/Documents/BOM.txt
Excel (.xlsx) link: https://github.com/minibois/eLiXiVy/blob/master/Documents/BOM.xlsx

[1.3.1] Optional components

As mentioned on the BOM, some components are optional. The optional components are the reset switch and rotary encoder, which will be discussed in section [4.2] and [6.1] respectively.
The BOM also mentions getting either 1N4148 *or* 1N4148W diodes, which will be discussed in section [4.6].

How many diodes/switches/keycaps are needed depends on the layout chosen, which will be discussed in section [3.1].

[2] Inspect/Testing the board

It is important to first know the PCB is in a good condition - before assembly of the board - as it may be difficult to find out later on if the fault is in the components or board itself.

[2.1] Visual Check

Do a visual inspection of the board:
- Does the soldermask look right in all places? Is there any copper showing, where it shouldn't?
- Is all the silkscreen looking good?
- Are all holes (i.e. screw holes and any THT holds, for switches and diodes) plated properly?
- Are all pads in place (i.e. the microcontroller, USB Type C and nearby components)?

[2.2] Electrical Check

Now it's time to grab your multimeter, set it to continuity mode (preferably with the beeper sound) and check if everything is connected properly. The main things to look out for are D+, D- and +5V, the Ground plane, the rows/columns and the rotary encoder's connection.

[2.2.1] Ground Plane

First it's time to start at the easiest to check, which is the Ground plane.
The easiest way to deal with that is by placing your probe on mounting hole H0 (near the USB Type C pad) and with the other probe touching all the other grounded pads and holes. It should beep (or give output on screen) to show it's all connected together.
Make sure to do this on:
- All the MCU's GND pads
- Both USB Type C GND pads and the shield holes.
- ICSP header's GND hole
- All capacitors (one pad should be connected, other not)
- Rotary Encoder's GND hole
- TVS diode GND pad
- Resistors R4, R5 and R6

[2.2.2] General checks

Next check the D+ and D- from the resistors R2 and R3 respectively to the microcontroller's pins 4 and 3 respectively (counting starts at the top-left corner of the MCU and moves clockwise).
After that, check the +5V connection from F1 to U2 and and from U2 to one of the capacitors around the microcontroller.
Now you can also check if the rotary encoder's pins are connected to the microcontroller. The encoder's pins A and B connect to pin 8 and 22 respectively.
Check if the reset switch's pad is connected to the reset pin on the microcontroller (pin 13).

[2.2.3] Row/Column Check

After these general checks are done, it's time to check if the matrix is connected properly.

Also double-check if all shared switches (i.e. the modifiers keys at the bottom right) and cut off holes (i.e. ANSI \) are connected properly.

After checking everything is fine, clean off the board (specifically the pads to solder to) with some IPA and some cotton buds.

[3] Choices to make, general information

The eLiXiVy gives you some choices to make, specifically in what layout you want to use and whether you want to use THT or SMD diodes.

[3.1] The layout

The layout of a (mechanical) keyboard is defined by the sizes of key(cap)s you use and where.
The eLiXiVy's layout support can be described as supporting an ANSI or ISO 65% layout, but as is usually the case, a picture speaks a thousand words:

Made using https://kle-render.herokuapp.com/

This layout can be described as the general 65% layout, with some choices to make, such as whether to use ANSI/ISO, a rotary encoder (or MX-like switch), etc.

Depending on the layout chosen (and whether or not a rotary encoder is used), the eLiXiVy requires between 66 and 69 keyswitches and diodes.
Speaking of diodes..

[3.2] SMD vs THT components

This guide has mentioned surface mount device (SMD) and through-hole technology (THT) a few times.

The difference between these two types of devices is the way they are secured to the PCB.
SMD components adhere to the board with metal pads. This includes parts like the microcontroller and passive components (such as the capacitors)

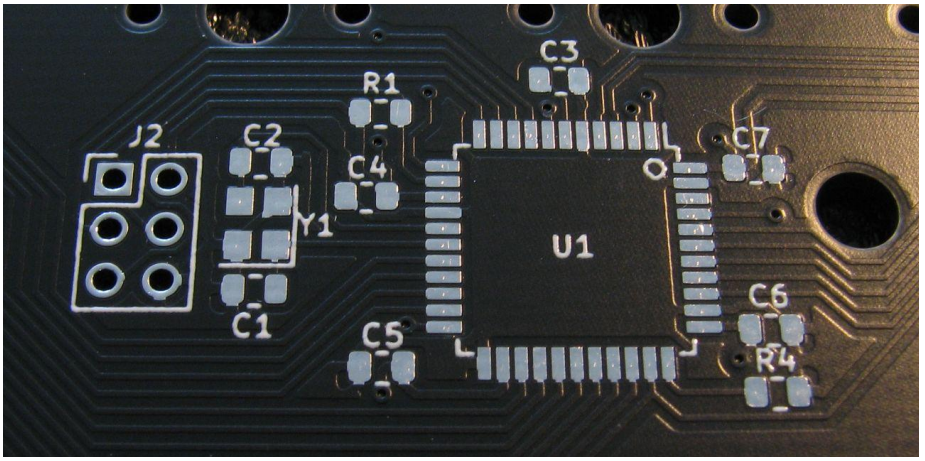
THT components have legs that stick through the board and adhere like that. This includes the switches, header pins and optionally the diodes.

All components on the eLiXiVy fall within these categories.
SMD: capacitors, resistors, ESD TVS diode, microcontroller, fuse, reset switch and crystal.
THT: Keyswitches, rotary encoder and ICSP headers.
SMD + THT: USB Type C.
SMD *or* THT: diodes (it's possible to use either type, see section [4.6]).

[4] Assembly, Part 1

After you've ensured the PCB is in proper working order, it's time to assemble the board.
It is recommended (if not a necessity) to use a multimeter between every step, to ensure no wrong connections are made.
Particularly, it's important to check if there is no connection between the +5V and GND nets. There are many components on the board that involve these two nets (like the microcontroller and all capacitors), so it is good to be mindful about what to check.

The easiest way to check if +5V and GND are connected together (which again, they *shouldn't* be) is by having your multimeter in continuity mode (beeping mode) and touching the probes on the +5V and GND points of the ICSP header.



The ICSP header (J2) has a GND and +5V point. These should not be shorted together, so should be tested often.
rev.1: +5V is in the top left and GND is bottom left.
rev.1a: +5V is in the top right and GND is bottom right

If your multimeter shows continuity (i.e. by beeping) at this point, you know there is a wrong connection made. By checking this often (such as after installing any component) you will ensure you know exactly where the fault may lie.

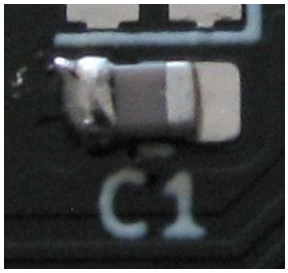
While this guide will call out when your PCB should be tested, that is only after some major steps. You should test for such shorts after every component, so you know for sure where any issue would be.
Now to the main assembly steps. Turn on your soldering iron/station and let it get up to temperature and keep in mind the general safety warnings when soldering.

[4.1] The passive SMD components

The passive components follow four/five basic steps to solder them in place:

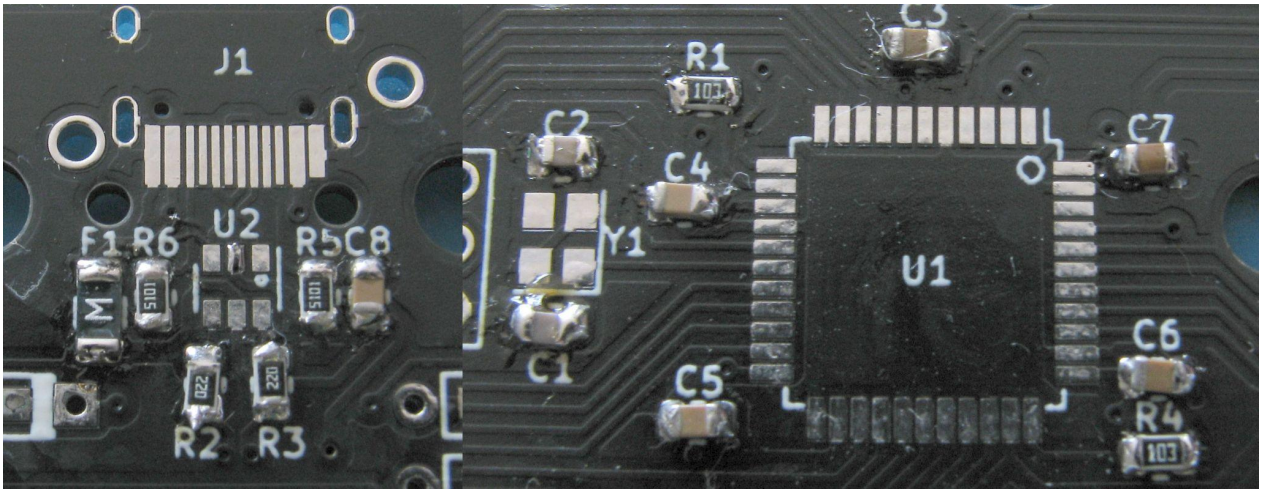
1. Hold your iron to one of the pads and apply some solder to that pad, with your 0.5mm tin

2. Grab the component with your tweezers and bring it to the previously made solderblob on the pad (The capacitors, resistors and fuse do not have a specific orientation)
3. Heat up the solderblob and bring the component into it
    i. The component may not be laying flat on the board, fix this by heating up the solderblob again and pushing the component down with your tweezers



4. Now hold your soldering iron to the other pad of the PCB and component apply some soldertin

These are the basic steps to solder the passive components in place. Be sure to use your multimeter after every component, to know for sure there are no wrong shorts made.
Soldertin also has some flux inside (which is the golden/yellow substance you see on the board above). Be sure to clean it off after installing each component as well. It is much easier to clean at this step than it is at a later time.



The two pad SMD components soldered in place. Not perfect, but it will work fine.

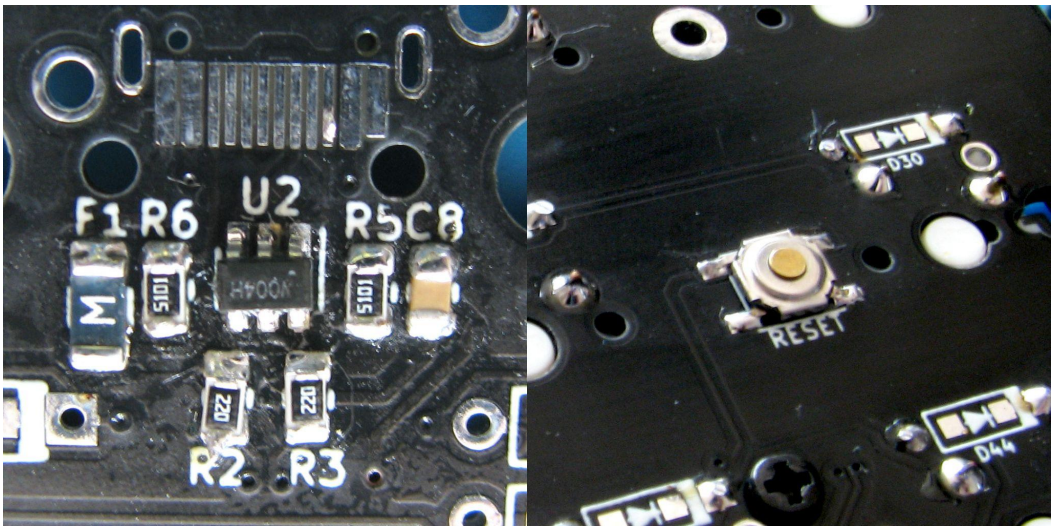
A list of all components involved in this (and any further) step will be in section [8].

[4.2] Multi-pad components (reset switch, ESD TVS diode)

The multi-pad SMD components follow the same basic steps as the passive components, just repeat step 4 for the remaining legs of the component.

Just like the passive components of the previous section, the reset switch does not have a specific orientation, but because it's rectangular shaped, it only fits in two ways.
This switch would be an optional part, as you can also just short reset and ground together (i.e. with tweezers or a paperclip), but it is still recommended, as you will need to trigger the reset when you want to flash a firmware to the board.
The reset switch can still be accessed when the keyboard is put together, as the case has a hole for it.

The TVS diode has a certain orientation, indicated by a small circle on the component and PCB.
This component follows the same steps otherwise, but because of the size and density of legs, it may be needed to apply a bit of flux and heat up the legs after soldering them in place. Just make sure there are no shorts.

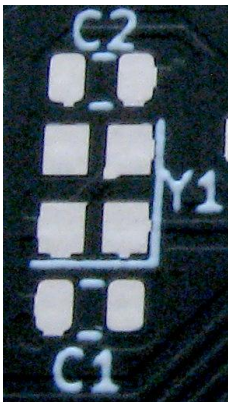


[4.3] Crystal

The difference between the previous SMD components and the crystal, is that the pads on the crystal are on the bottom of the components, so you can't touch the PCB's pad and crystal's pad at the same time.
This makes soldering in the crystal a little harder, as you can't heat the PCB and component at the same time.

The way I've found to work is:
- Tin all the pads with a tiny bit of solder
- Add some flux to all the solderblobs
- If possible, change your iron's tip to a small one
- Try to heat the blobs one by one, if with a small tip try to get under the crystal

Be mindful the crystal has a specific orientation, or rather two specific orientations, as it's a symmetrical one way.
As a rectangular part on a rectangular footprint, it can only fit in two ways, which would both work.

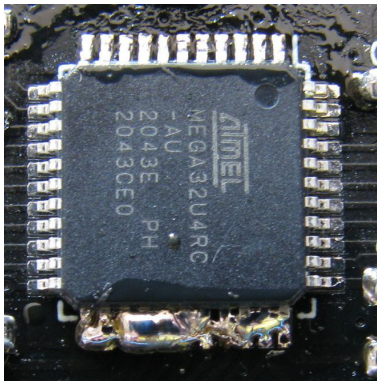


The crystal and its PCB footprint are rectangular, so it will only fit in the two correct orientations

[4.4] Microcontroller

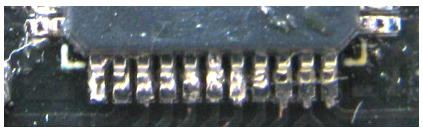
Warning: ensure there are no shorts on +5V on GND right now.
There will likely be shorts between these nets when soldering the microcontroller in place, so if there were issues before, you want to know about those now.
The easiest place to check for shorts is on the ICSP header, as explained in section [4].

The method I've found works well with soldering the microcontroller is the 'drag-solder' method. This involves first tacking down one pin (just like done in section [4.1]), making sure the microcontroller is properly in place (if it's not, heat the pin and set it right in place) and then adding a lot of solder to one side of the microcontroller, like so:



Note the excess of soldertin on the bottom pins

After doing this, you can add some flux on top of this soldertin and go over it with your solderwick, to get a nice thin layer of soldertin, like so:



Note the thin layer of solder on the bottom pins.

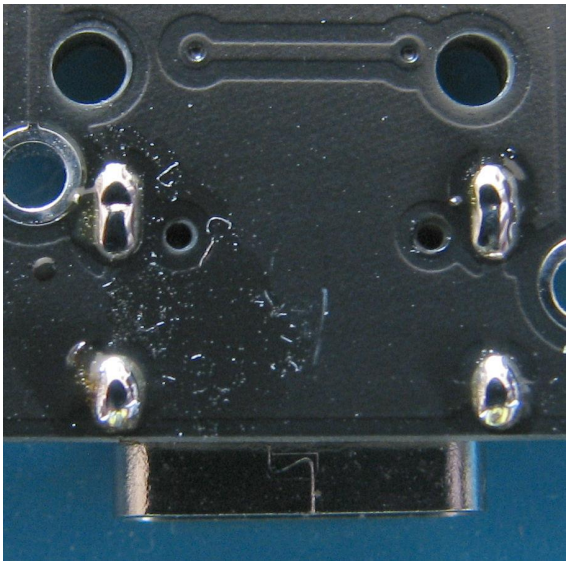
Be sure to test after soldering one side, I'd recommend this :

1. Add a bunch of solder
2. Add flux
3. Wick it away
4. Use multimeter to make sure there is no short between +5V and GND, nor any shorts between neighbouring pins
5. Repeat from step 1 for another side, until all sides are soldered in place

[4.5] USB Type C

The USB Type C connector is the only component in this build that uses SMD *and* THT methods of assembly. It has four legs that protrude through the board (which are there for grounding and stability) and twelve SMD pins, for connectivity.
The recommended way of installing such a connector involves using flux.

First tack down the connector - as explained in [4.1] - and make sure the SMD pins are fully aligned (if not, heat the tacked down pin and move the connector accordingly). Also make sure it's flush with the board and not at an angle (although the two plastic legs should help with the angle).
Once it's right in place, it's time to solder down the THT pins. Turn the board around, heat up the THT hole and add solder until a slightly convex pad forms:

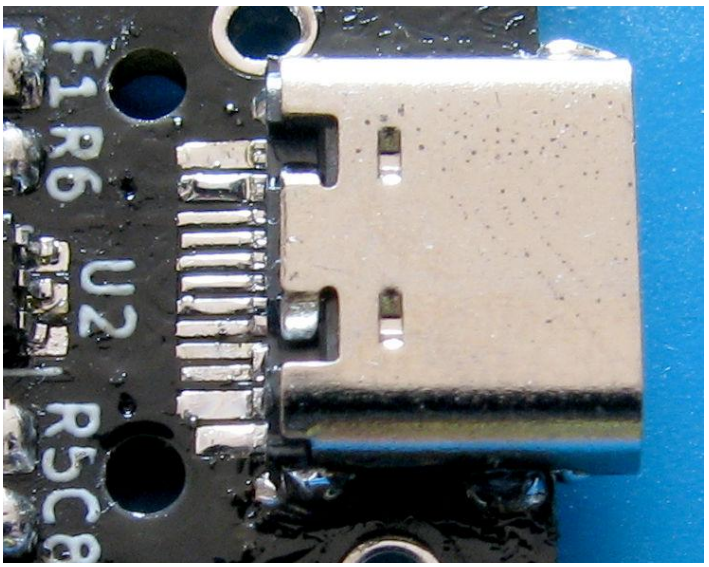


The legs of this connector are not long enough to protrude through the entire board and this connector will likely often get plugged in and out, so it is important to add additional support to the USB Type C connector.

Turn the board around again, so you see the SMD pads of the USB Type C connector. It's now time to add some tin to the THT holes, but on this side.
Touch the iron on the USB Type C connector leg and the THT hole and add in some solder tin.

Once you have added some solder to all four sides, wait a bit for the connector to cool down. Now you can try to plug in a USB Type C cable and try to wiggle it a bit. No movement in the connector? Perfect! If there is some movement still, add some more tin to the places that seem to need it still.

Now it's time to tackle the SMD pads of this USB Type C connector. The recommended strategy is similar to the microcontroller. It's recommended to add a generous bit of flux, and then going over the pins with quite a bit of flux, following with some solderwick to remove any excess.
Just like with the microcontroller, shorts are bound to happen. Check for shorts with your multimeter. To get rid of shorts, it's usually a case of applying some flux and touching your iron on the offending pins, possibly with some wick if there is too much tin on the pads.



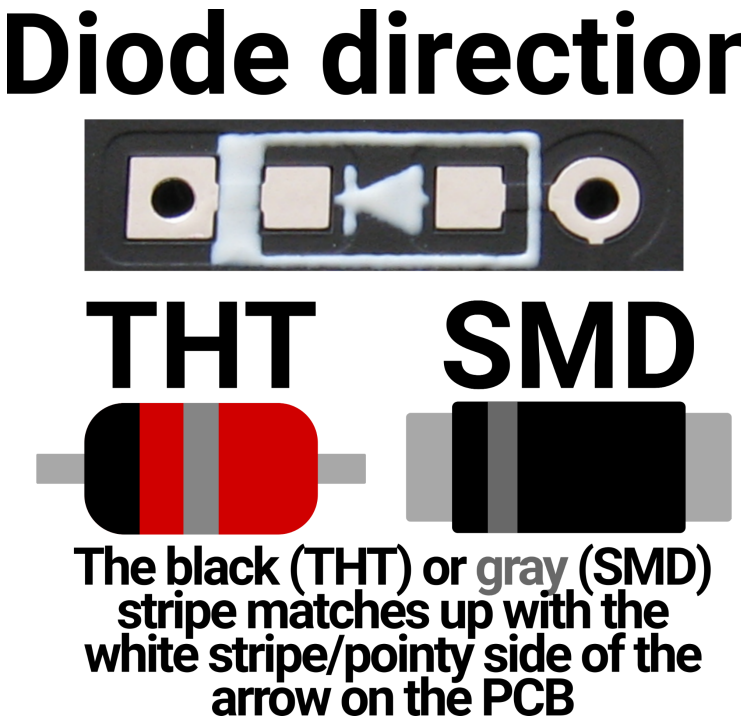
USB Type C connector soldered into the board.
The legs don't make contact with each other and the THT pins have solder on both sides

[4.6] Diodes

With the diodes, a few choices can be made. Specifically if THT or SMD should be used and if they should be installed on the front or the back. It is of course possible to have a mix of both, such as using some THT diodes and some SMD diodes or using some on the front and some on the back, but this guide is written to assume you will use either THT or SMD and either the front or back.

The functionality between the different options will be the same, the decision is mainly up to:
- Aesthetics: SMD and THT diodes look different and having them on the front/back influences how the board looks, which is of course very subjective.
- Ease of install: personally I find THT diodes less of a hassle to install, but your mileage may vary.

As for installation, the main thing to remember is the orientation of the diodes matters.
The diodes have a marking on them, in the form of a stripe. The PCB has a thicker white stripe, the pointy side of the arrow and the square pad that this stripe needs to point into, as follows:



[4.6.1] Not all diodes are needed

Depending on the layout chosen, not all diodes may be needed.
D45: only needed if you use a 1.25u Left Shift + 1u Backslash key setup (i.e. ISO layout)
D65: only needed if you use a 3x 1u modifier (Alt, Fn, Ctrl) in between the spacebar/arrow keys
It doesn't hurt anything to insert them all, these two may just not be needed, depending on the layout chosen.

[4.6.2] SMD Diodes

The way to solder in an SMD diode is the same as described in section [4.1], except you have to keep in mind the orientation of the diode.

[4.6.3] THT Diodes

To install a THT diode, you have to bend both legs so the diode can be inserted in the two holes on the board. Insert the diode - keeping in mind the orientation - and bend the legs out, so the diode stays in place.
You can insert one diode at a time, all at once or a few at once and then solder them in place. Be sure to clean the flux off of it right after.

With the diode inserted into the board, you can touch the soldering iron to the leg of the diode and pad for about half a second, bring in the soldertin (I recommend 0.7mm for this), so a nice concave cone forms around the diode and pad.
Then remove any excess length of the diode, using your sidecutters/flushcutters.

In picture form:

Bend the legs, insert into the board (keeping in mind the orientation), bend the legs to hold it in place, solder the legs, cut the excess

It may prove easier to first install a bunch of diodes (i.e. a full row) and solder all one after another, then cut the legs one after another and then go to the next row.

[4.6.4] Diodes on the front vs. back

You can install the diodes - whether SMD or THT - on both sides of the board.
Due to a design oversight in rev.1a (not present in rev.1), it's not possible to install a diode on the front of the board for diodes D13 and D62 (backspace and spacebar respectively). A diode can only be installed on the backside and special care should be made to cut the leads of THT leads short.
Diodes on the front - or protruding THT legs - can come in contact with the metal stabilizer wire for the backspace/spacebar stabs.

rev.1 does not have such restrictions. All diodes can be installed either on the front or back (regardless of SMD vs. THT use).

[4.7] Clean flux

As mentioned in the general tips, it's important to remove flux off the board after installing each component, because it's much easier to get rid off at that stage. Be sure to clean the board of any leftover flux. While not all flux is like this, some flux is known to be conductive, which could make shorts in the wrong places.

Use a cotton swab and some IPA to get rid of any flux left on the board.

[5] Testing, Bootloader, firmware and more testing

Now that the eLiXiVy is in a minimum viable product state, it's time to check for continuity, load the needed software and test the functionality.
This step is close to being able to get output from your board.

[5.1] Testing

First grab your multimeter and make sure there are absolutely no shorts that shouldn't be there.
Places recommended to test at the very least:
- +5V and GND (on the ICSP header),
- GND and Reset (can also be done on the ICSP header)
- Check around the MCU, to make sure none of the legs on that component are connected together
- All other connections on the ICSP header (none should short together)
- Check everything around the USB Type C connection (such as D- and D+, +5V/VCC and GND, etc.)

[5.2] Burning the bootloader (possibly optional)

These steps may be optional, as certain atmega32u4 chips (such as the AU model) should come with a bootloader included. Some versions though - like the RC - do not come with a bootloader included, in which case this step is required.

It's easy to recognize if your atmega chip has a bootloader. After ensuring there are no shorts anywhere - as described in section [5.1] - you can connect the board to your computer using a USB cable. It will either show up on your computer or not. In the "or not" case, the atmega chip does not have a bootloader and you have to follow the upcoming steps. If it does show up, you can skip these steps and move to section [5.3].

[5.2.1] Tools/Supplies required

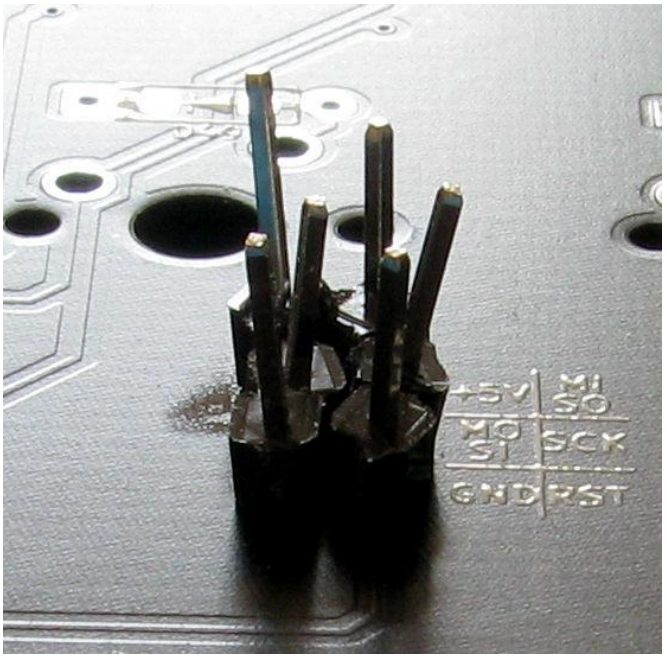
After you have ensured there are no shorts anywhere, it's time to burn a bootloader on the board.
For this step, you need:
- An Arduino UNO to use for the flashing
- Six 2.56mm jumper wires
- A USB cable to connect your Arduino to a PC
- Arduino IDE installed on your PC (https://www.arduino.cc/en/software)
- The USB Type C cable for the eLiXiVy

[5.2.2] Soldering the ICSP header in place

To connect the eLiXiVy to the Arduino UNO, header pins are needed on the board to connect to. These headers together are known as the ICSP header.
These pins can be purchased in different configurations (and cut if needed). While the header on the board is laid out as a 2x3 pin block, it's possible to get 2x3 headers or two 3x1 or six single (1x1) headers.

These header pins have to be removed at a later point (as explained in [6.1]), which is why it is recommended to use single (1x1) pin headers (it's easier to remove those).

Soldering these headers in place is a lot like soldering the THT diodes in place. Prop up the board so the pin is sticking through the hole, bring in your iron and heat up the leg/pad and bring in some solder.



Looks a bit like a spiky mess, but these are just temporary.

[5.2.3] Arduino IDE settings

These steps will be written as if you're using an Arduino UNO as a programmer. It may be possible to use another type of 5v Arduino - such as the Leonardo - but this guide will only cover the use of an UNO.

First connect your Arduino to your PC and open the Arduino IDE

1. Go to *Tools* > *Board* > and set the right board (Arduino UNO)
2. *Tools* > *Programmer* > *Arduino as ISP*
3. *File* > *Examples* > *11.ArduinoISP* > *ArduinoISP*
4. Upload the sketch (press the "right arrow" button, at the top left)

Unplug the Arduino from the computer, but keep the Arduino IDE open.

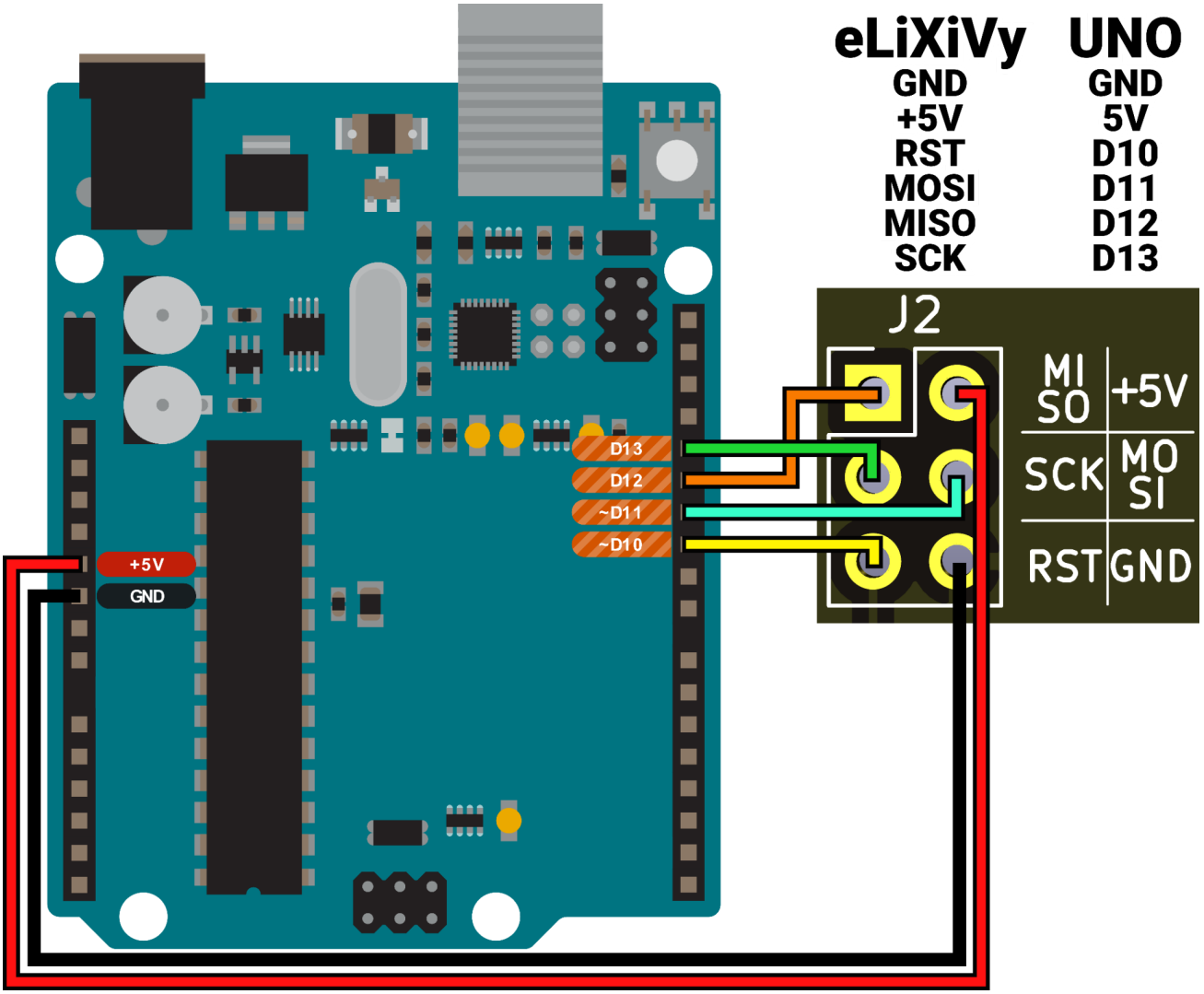
[5.2.4] Wiring it all up

Now it's time to grab the eLiXiVy and the six jumper wires to connect it to the Arduino. This is a case of connecting the ICSP header on the eLiXiVy to the right pins on the Arduino.

NOTE: The ICSP header in rev.1 is the default standard, while in rev.1a it was horizontally mirrored. You can follow the picture or table for PCB rev.1, for PCB rev.1a, follow the table below.

eLiXiVy ICSP	Arduino UNO
GND	GND

+5V	5V
RST	PIN 10
MOSI	PIN 11
MISO	PIN 12
SCK	PIN 13



Arduino image source: https://content.arduino.cc/assets/Pinout-UNOrev3_latest.pdf

[5.2.5] Burning the bootloader

Now that the eLiXiVy is connected to your Arduino, plug the Arduino into your PC with its USB cable, as it's time to set up the Arduino IDE and burn the bootloader.

Go to *Tools* > *Board* > *Arduino Leonardo* (the Arduino Leonardo and eLiXiVy both use the atmega32u4 microcontroller, the Leonardo bootloader can be loaded to the eLiXiVy).
After you have set the Arduino IDE to Arduino Leonardo, go to Tools > Burn Bootloader and wait a small while (it should just take a couple seconds) and that should be it.

Now unplug the Arduino from the USB in your PC, then take out the jumper wires from the eLiXiVy.
You can now plug in the eLiXiVy into your PC, with a USB Type C cable and if all went well it will now show up as an Arduino Leonardo!

[5.3] Getting the firmware and prerequisites

The atmega32u4 chip on the board should now be equipped with a bootloader. Either through the steps described in section [5.2], or it already came with one to begin with. It's now time to flash the firmware on the board, so the eLiXiVy can behave like a keyboard.
The firmware decides the functions of the keyboard, such as what letter/number/action each key corresponds to and what the (optional) rotary encoder has in terms of actions behind it. This firmware-file is presented in a .hex file-type and made with QMK, one of the most popular keyboard firmwares.

This file is flashed on the board using QMK Toolbox, which can be found here:
https://github.com/qmk/qmk_toolbox/releases
How this program is used will be discussed in [5.4], as it is now time to grab the right .hex file for the keyboard configuration.

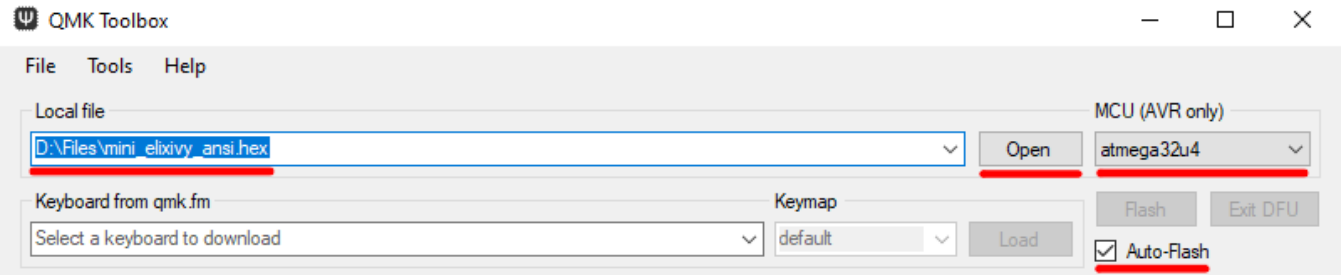
The Github repository includes three .hex files: default, ANSI and ISO.
The default and ANSI files follow the ANSI layout, which is shown below, while the ISO layout switches around the green left Shift for a narrower left Shift and Backslash key and switches the yellow keys (Enter and Backslash) for a differently shaped Enter and backslash.

The ANSI layout also incorporates PgUp, PgDn and Insert on the function layer, on the Printscreen, Delete and Accent Grave keys respectively, while the default and ISO layouts keep it simpler, only using the function layer for the number keys.
For now, grab the desired keymap from the firmware folder (https://github.com/minibois/eLiXiVy/tree/master/Firmware) and move on to section [5.4].

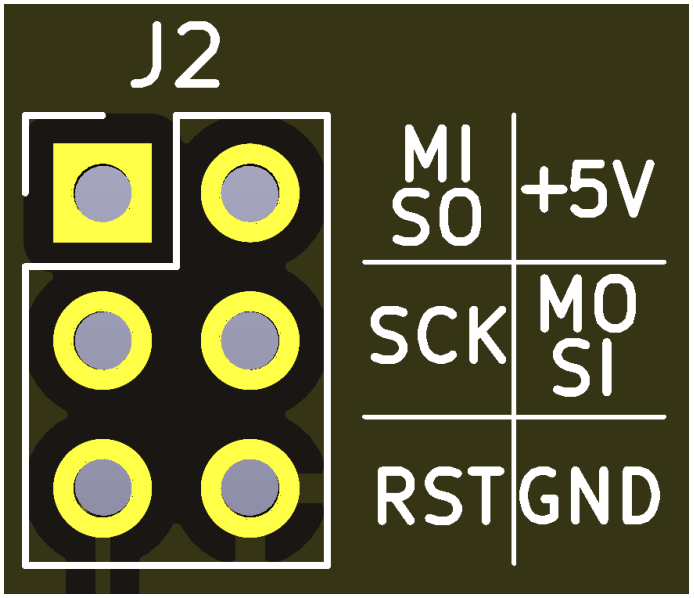
Should you want to apply changes to the layout, check section [6.6] for more information on adapting the keymap to your use.

[5.4] Flashing the firmware

Open QMK Toolbox, installed previously.
Press the “Open” button and locate the previous downloaded .hex file. Set “MCU (AVR only)” to atmega32u4 and tick the box next to “Auto-Flash”. (*It’s also possible to skip the “Auto-Flash” option and just press the “Flash” button after resetting the board, but with the seven or so seconds you get to press “Flash”, I’ve found it easier to just use Auto-Flash.*)



Now it’s time to reset the board, which puts it in the flashing mode.
If you soldered the reset button in place in section [4.2], simply press that button.
If you skipped that step: use something conductive (i.e. tweezers, a paperclip, a jumper wire, etc.) to jump the reset (RST) and GND nets together. This is probably easiest to do on the ICSP header:



If you don’t have a RESET button in place,
simply short the GND and RST pads (bottom two pads).

Once you have pressed the reset button (or shorted the nets together), you will hear the sound your computer plays when a device disconnects and QMK Toolbox will see the board and flash the firmware on it.
The eLiXiVy might connect/disconnect a couple more times (so you might hear the (dis)connect sound some more). This will all only take a couple of seconds.

After this you now have flashed the QMK firmware to the board and it should now show up as an eLiXiVy!

[5.5] Keyswitch testing

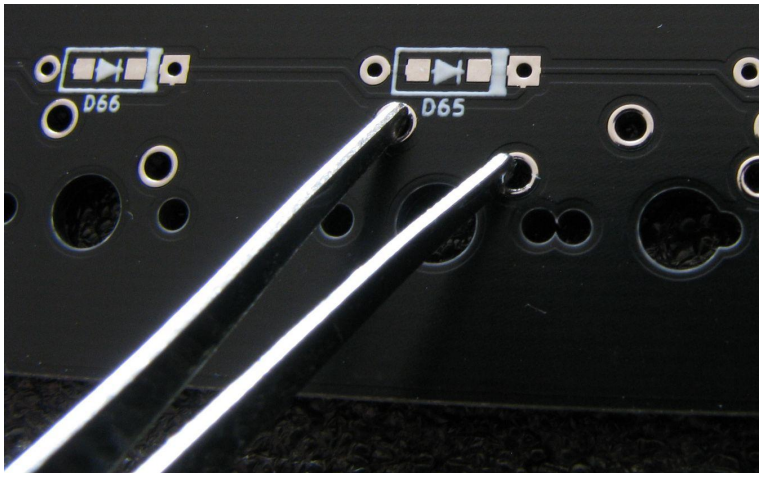
Your eLiXiVy can now interact with your computer as a keyboard, so it's time to test out all the switch pads, before the switches are installed.
The easiest way to do this is using tweezers and a keyboard tester.

First grab your multimeter, set it to continuity mode and touch both probes together. This means electricity runs from one probe to another.
We want to make sure the tweezers you use also allow electricity to run through it. Touch a probe to either end of your tweezers.
Does it beep? If so, these tweezers are fit to use for this following step.
Do they not? These tweezers are not fit for the following purpose.

Find some tweezers that do allow electricity to flow through (something like a paperclip can also be used, tweezers are just more comfortable.

Now that you have ensured you have a tool fit for this step, it's time to test the keyboard functionality.
Be sure to only touch the row/column pins of the keyswitch together, not anything else on the board.
Avoid touching anything else together (i.e. the pads on the USB Type C connector, the ICSP header's pins, the rotary encoder holes, etc.

Go to the QMK configurator keyboard test page found here: https://config.qmk.fm/#/test
Now touch all the keyswitch row/column holes together, like so:



Touch the two pads of each switch together with your tweezers or a pair of pliers

Do this with all switch locations you intend to use and if all went well, all keys in your keymap should now light up on the screen!

[5.6] Troubleshooting

No matter how well everything goes, with so many steps involved, it's of course possible something doesn't work and has to be fixed. Here is a list of common issues that can happen and how to fix them.
This chapter is included before the install stabilizer/keyswitch/rotary encoder step, as the eLiXiVy is still easy to handle without those installed.

[5.6.1] There is a short where they shouldn't be

A short is when two nets connect, that shouldn't.
One very common short that can happen is between +5V and GND, which is because they interact a lot with each other.

When assembling the eLiXiVy, you should use your multimeter to check if there is a short between these two nets. That way, if there is a short, you know exactly what component it has to be in.
One common place where such a short will happen (whether it be between +5V and GND or other adjacent pins) is the microcontroller or USB Type C connector.
These are components with many very well legs.

When such a short does happen in the microcontroller, you should know where it is (at least which of the four sides, if you follow the 'solder one side > test > repeat' approach).
With both the USB Type C connector and atmega32u4 microcontroller, the approach will be the same.
There are different ways of handling such a short, my go to is adding a generous bit of flux on the (possibly) offending microcontroller pins, place your solderwick on it and then you soldering iron, to wick away some solder (and with the flux, the flux should flow away from the soldermask too).

With other shorts and their associated components, it's typically much easier to visually see if there is any solder bridging the connectors.

[5.6.2] The tin doesn't melt when soldering

Either your soldering iron doesn't get hot enough, or it's just not set to a high enough temperature. Or maybe you're dealing with a ground connection, which just takes longer.

A weak iron might have trouble with melting the solder, particularly on the components that connect to ground (ground is a large copper plane on the PCB, so there are a lot of places that can soak in heat).
My go-to temperature is 350°C (662°F), on a pretty good iron. If you can change the temperature on your iron, that would be the target temperature I go for.
Leaded solder also has a lower melting point than unleaded, so perhaps you have gotten unleaded solder (no Pb).

P.S.: make sure you soldering iron is turned on (I totally don't say out of experience)

[5.6.3] The bootloader doesn't install on the eLiXiVy

Make sure the wiring between the eLiXiVy and your Arduino of choice is correctly done.
Check if the crystal is correctly soldered on the board (with its pads being on the bottom it's not possible to check with a multimeter, but just check visually).

[5.6.4] The QMK Firmware doesn't work

The firmware provided on the Github page of this project is verified to work, if the one on the QMK configurator website doesn't work.
If it doesn't work, please open an issue on Github: https://github.com/minibois/eLiXiVy/issues

[5.6.5] One/some/all keyswitch location(s) doesn't work

Usually when a keyswitch location doesn't work, it's an entire row or column that doesn't work.

Identify if it's just a single keyswitch location, or if there is some pattern to it (i.e. a specific row/column).

This is why I recommend in section [2] to make sure to test the PCB, to know if all rows/columns connect together and if they connect well to the pads for the microcontroller.
If you had done these checks, the only thing that could be at fault is the microcontroller not being properly soldered to the pads on the board.

[6] Assembly, Part two

Now that the keyboard works, it's time to add the final components to it.
Unplug the keyboard from the USB port, as more components will need to be installed.

[6.1] Remove ICSP header pins

The ICSP header has served its purpose, in letting you burn the bootloader to the MCU.
It's now time to remove this header, as it would otherwise interfere with the spacebar stabilizer.

The easiest way to do this depends on what sort of headers you have used.
With the recommended singular 1x1 headers, the easiest way to go about this is by propping up the PCB vertically, making sure it's standing pretty sturdy. Hold your iron on the solderblob of one pin and pull lightly on the pin header with a pair of pliers.

With the other pin configurations, you're going to have to use some solderwick and a desoldering pump to get rid of the solder, to remove the pins.

[6.2] (Optional) Rotary Encoder

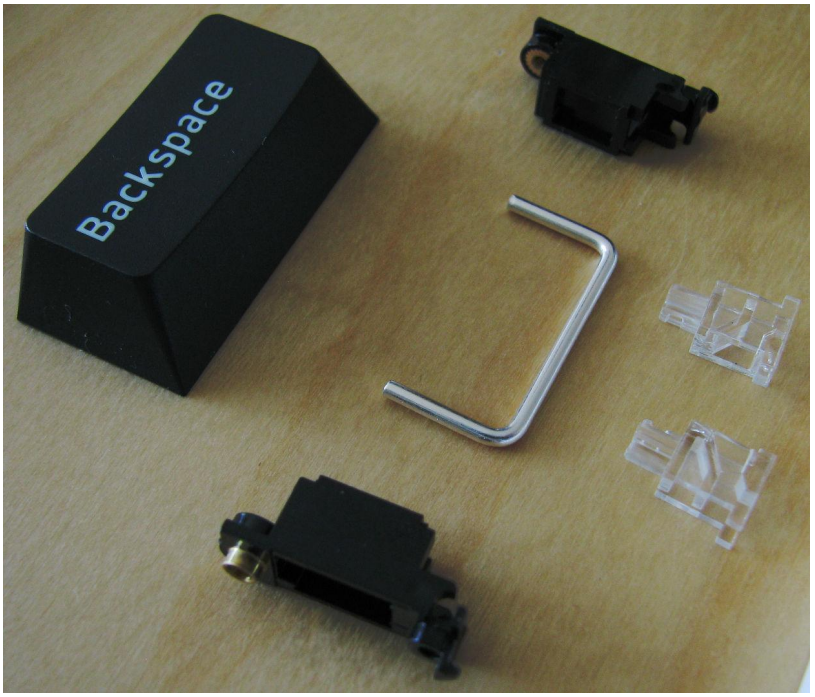
In the top right of the eLiXiVy you can choose to use a Cherry MX(-like) keyswitch, or a rotary encoder (optionally one with a push switch option).
This rotary encoder can be used for a host of different actions (such as volume adjustment, pressing arrows up/down, screen brightness, scrolling through a website, etc.), which makes it a convenient and helpful feature of a keyboard.
Nonetheless, the rotary encoder is an optional feature, you can choose to use or not (in favor of an extra keyswitch).

The rotary encoder installs just like you would expect.
Align the encoder with the holes in the board and set it in place (ensure it's flat on the board, so fully seated in place). Soldering it in place is just like the other THT component(s) installed (such as the ICSP header), by holding your iron to the pad and leg for a second and adding some soldertin.
The two legs on either side of the encoder require more tin than the other legs (and thus more time with the iron), as these are there for the stability of the encoder.

[6.3] Stabilizers

Stabilizers serve to stabilize - as the name suggests - the larger keys in the keyboard.
This includes the spacebar, backspace, enter (whether it's the ANSI or ISO style) and left shift (in a 2.25u/ANSI layout).

A stabilizer consists of five main parts:
- A (metal) wire
- Two stems
- Two housing



The housings hold the stabilizer to the board, the stems move up and down with the key and the wire allows them to move up and down.

Because of the metal-to-plastic and plastic-to-plastic contact between these parts, it's advised to apply lubrication to these parts, which would aid in a fluid typing experience and sound.
There are dozens of lubricants available and even more ways to apply it.
My go-to recommendation on how to apply lubricant would be this video from Taeha Types:
https://youtu.be/usNx1_d0HbQ

When using screw-in stabilizers the side where the screw goes in aligns with the smaller of the holes on the PCB.

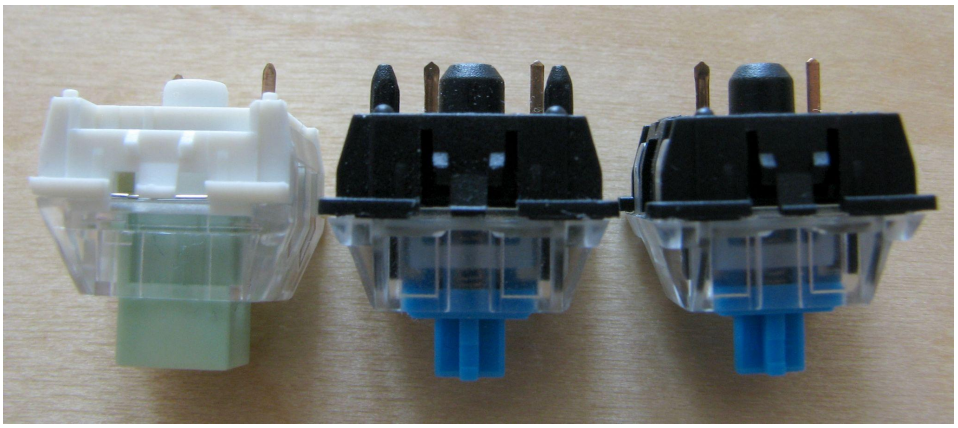
[6.4] Installing the switches and plate

After installing and tuning the stabilizer to your liking, it's finally time to install the switches into the keyboard.
Soldering the switches in place is much like soldering any other THT components in place (like the ICSP header pins or the THT diodes). Just hold your iron on the pad and pin for a second and bring in the solder for a nice concave solder joint.

To ensure the switches, plate and PCB are installed properly together it's important to make sure the switches are flush with the PCB. This involves starting with the corner switches, to ensure it's held tightly together.
Another way to make sure the switches are well in place is by only soldering one leg of the switch in place and after holding the iron on the switch leg and pushing the switch into the PCB to ensure a flush install.

After all the switches are in place, it's time to check all the switches to see if they are straight. Most 1u keys will pose no problems as they are supported on all sides on the plate, but some of the large keys can be a little crooked. It is recommended to install some keycaps on these keys (keys like Space, Shift, Backspace, etc.) and check if they are straight.
Are they not? Hold your iron on the solder joint and adjust how needed.

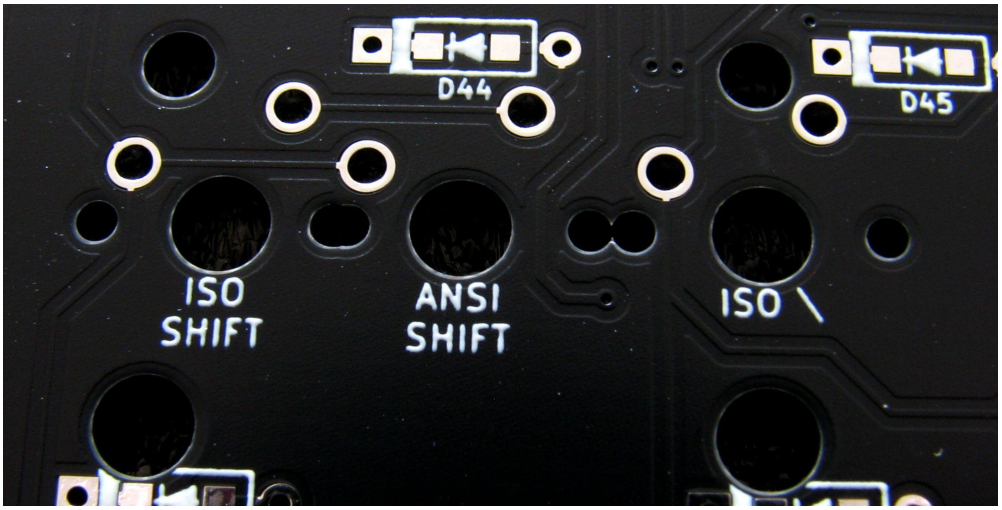
This is something less of an issue with PCB mount/5 pin switches:



3 pin vs. 5 pin vs. 5 pin with its legs cuts off

Note: PCB rev.1a supports 5 pin switches everywhere, except for the key in the top right (when a rotary encoder is not used). PCB rev.1 supports 5 pin switches everywhere, including the top right.

Of course you should keep in mind the layout you've chosen when installing the switches. The PCB has some markings on the silkscreen, to tell you what locations to use when using certain size/variant of keys:



These markings on the PCB will show you what layout the holes are for. Refer to [3.1] for layout options. rev.1a only has the markings on the front, rev.1 has them on the front and back.

After all switches are in place, straight and flush to the board, it's time to solder down the other leg of all the switches. Be sure to double-check you got all pins soldered down once you're done.

[6.5] Case and keycap install

The eLiXiVy is designed to work with KDBFans' 65% low profile case. Before the PCB/plate/switch assembly is installed it's important to make a choice on if the elevation feet will be used or not. These feet elevate the back of the board for an incline. These feet install with the large silver screws, screwed in from the inside of the case.

After deciding on using the elevation feet, it's time to install the small rubber feet on the board. The case comes with four rubber feet for the bottom of the board. Two go on the front side of the back, while two may be installed on the back if the elevation feet are not used.

The case comes with two sets of screws, silver ones and smaller black screws. The silver screws would screw the plate to the case, while the black screws are used to screw the PCB into the case. Using one or the other can make a minimal difference in sound and feel.

If you decided to install the keycaps before installing it in the case, you do need to remove some of the keycaps to reach the screw holes:

The keycaps which have to be removed to reach the screw holes

If you haven't done so already, install the keycaps, which is one of the last steps before enjoying your new keyboard!

[6.6] (Optional) Advanced keymap options

The provided .hex files should provide a solid basis for most people, but should the default, ANSI or ISO layouts not be to your liking you can of course edit these to suit your needs. There are different ways of going about this editing, which will be discussed in order of simplicity:

- Through the QMK Configurator website
- Editing the files locally
- Publishing the files for others to use

Editing through the QMK configurator website requires nothing more than the website itself and the QMK Toolbox - discussed in [5.4] - while the other ways require a bit more setup.

[6.6.1] QMK Configurator

The QMK Configurator is an online website which allows you to configure and compile a keymap to your liking. While it is a quick and easy way to edit the keymap, it doesn't expose all features QMK has to offer. Some features missing are related to the rotary encoder and some advanced settings (such as tap-dance and certain advanced keycodes).
If your main goal is to change around some keys and quickly get a keymap, the QMK Configurator is the perfect way to achieve this.

Open the configurator website and find the eLiXiVy (under the name 'mini_elixivy), or simply open this link: https://config.qmk.fm/#/mini_elixivy/LAYOUT_65_ansi

Here you can click on any keys and change its function to suit your needs. Once you are happy with the result, you can give the keymap a name (at the top) and compile the keymap (top-right). Once the spinning potato is gone, the keymap in .hex format can be downloaded by pressing the “Firmware”-button.

Note: the QMK Configurator only supports one default layout, so the keys on “LAYOUT_65_iso” will all show “N/A”. To quickly import all keycodes to this iso layout, select the LAYOUT_65_iso option, download the mini_elixivy_iso.json found here:
https://raw.githubusercontent.com/minibois/eLiXiVy/master/Firmware/QMK%20Configurator%20files/mini_elixivy_iso.json
Now press the “Import QMK Keymap JSON file” button and select this downloaded .json file, to import all keycodes and edit it like described above.

The online QMK configurator is great, but should you want to share your keymaps with the greater QMK community, change rotary encoder options and more, you should look at locally editing the keymap files. This can be done by cloning the Github repository, making the edits you want and submitting a pull request to include your new keymap (which is optional, you can also just make edits, compile the keymap and upload it to your eLiXiVy).

[6.6.2] Editing the files locally and sharing the keymaps

Should the online QMK configurator not have the right settings for your purpose - or if you want to share your keymaps to the greater QMK community - you can edit and compile the keymaps locally.
The first steps include downloading the repository locally (using your Git client of choice, like Github) and setting up the different dependencies QMK has.

These steps are described as such here: https://beta.docs.qmk.fm/tutorial/newbs
Where the default keyboard can be set to “mini_elixivy” of course.

There are a few things to keep in mind though, if you plan on publishing your keymaps to the main project though. If you are familiar with the Git workflow, these will seem obvious, but it doesn’t hurt to mention it again.
After cloning the repository locally you will want to work in your own branch. After you are happy with your keymap and want to include it in QMK, you can publish your branch (which will make a fork of the project) and then make a pull request, to be reviewed by QMK members and to include your keymap in the full project.
Please read this page thoroughly before making a new pull request, to be sure your changes can be accepted in the QMK repository: https://beta.docs.qmk.fm/developing-qmk/pr_checklist

If you’re unsure about what to do, do not hesitate to contact someone about your questions. It’s possible to contact the QMK community via Discord, Reddit or Github: https://beta.docs.qmk.fm/tutorial/support
Should you run into an issue with the keymap you think should be solved by me, please open an issue on the Github page: https://github.com/minibois/eLiXiVy/issues

[6.7] Enjoy the keyboard

Now that you’ve assembled, tested and perfected your own eLiXiVy, it’s time to enjoy your new keyboard! Be sure to show off your work where-ever you prefer!

I shared it on the LinusTechTips forum, including an informative piece on the anatomy of PCB's and information on why a license should be used (or at the very least considered):
https://linustechtips.com/topic/1366493-elixivy-a-65-mechanical-keyboard-build-log-pcb-anatomy-and-how-i-open-sourced-this-project/

Should there be any questions, concerns of corrections, do not hesitate to open an issue on the Github page for this project: https://github.com/minibois/eLiXiVy/issues

[7] Sources

eLiXiVy source: https://github.com/minibois/eLiXiVy
rev.1 source: https://github.com/minibois/eLiXiVy/releases/tag/rev.1
rev.1a source: https://github.com/minibois/eLiXiVy/releases/tag/rev.1a (alpha version)
eLiXiVy build log:
https://linustechtips.com/topic/1366493-elixivy-a-65-mechanical-keyboard-build-log-pcb-anatomy-and-how-i-open-sourced-this-project/

Assembly guide license: CC BY-SA 4.0, See [10]
PCB and schematic license: https://ohwr.org/cern_ohl_s_v2.pdf

ai03's PCB design guide, which this board was based on:
https://wiki.ai03.com/books/pcb-design/page/pcb-guide-part-1---preparations

Libraries used in the creation of the PCB/schematic:
That-Canadian's KiCad_labs (for the D-SOD123_axial-dual):
https://github.com/That-Canadian/KiCad_Libs
Type-C.Pretty: https://github.com/ai03-2725/Type-C.pretty
MX_Alps_Hybrid: https://github.com/ai03-2725/MX_Alps_Hybrid
random-keyboard-parts (for a reset switch): https://github.com/ai03-2725/random-keyboard-parts.pretty

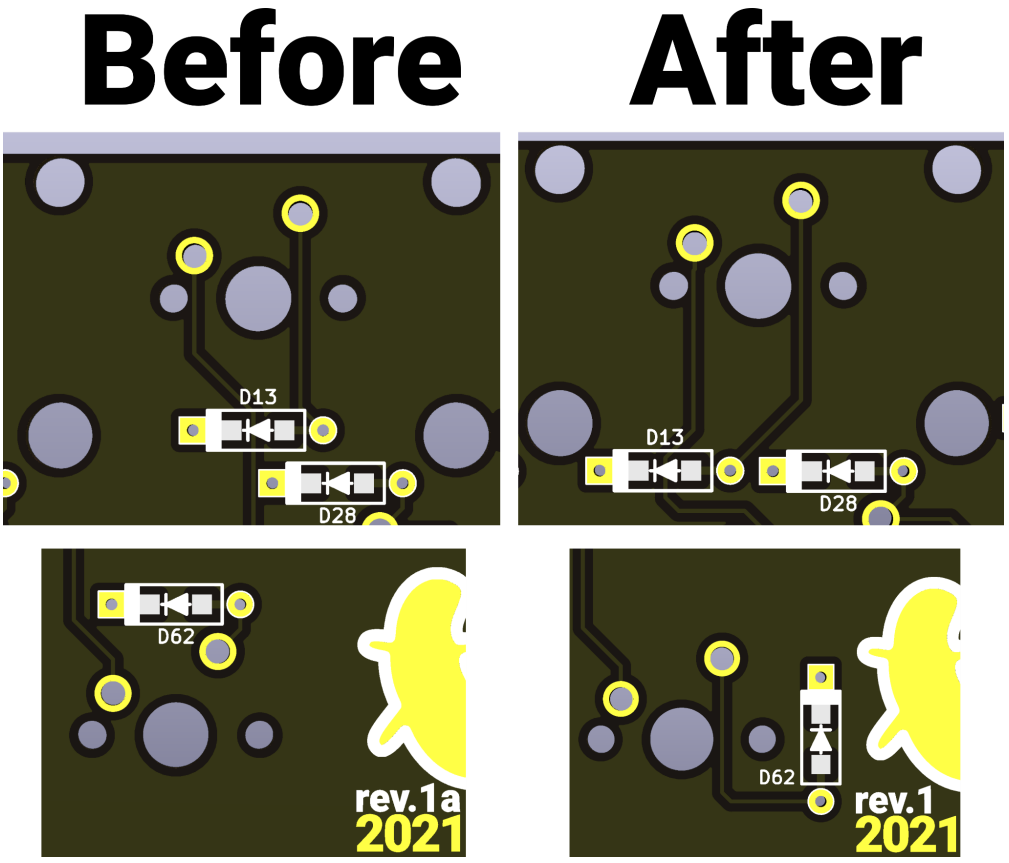
[8] Differences between rev.1a and rev.1

After assembly of rev.1a, some changes were made after reflecting on this build experience.
The differences between the two revisions are minor, but still bare a mention.

[8.1] Diodes

During assembly, it was clear diodes D13 and D62 - backspace and spacebar respectively - were not in a great position.
The metal wire on the stabilizer for these keys would come in contact with the diode when it was installed on the top of the board (whether SMD or THT diode) and when a THT diode was installed on the bottom, the legs had to be cut almost flush to avoid contact between the leg and the stabilizer wire.

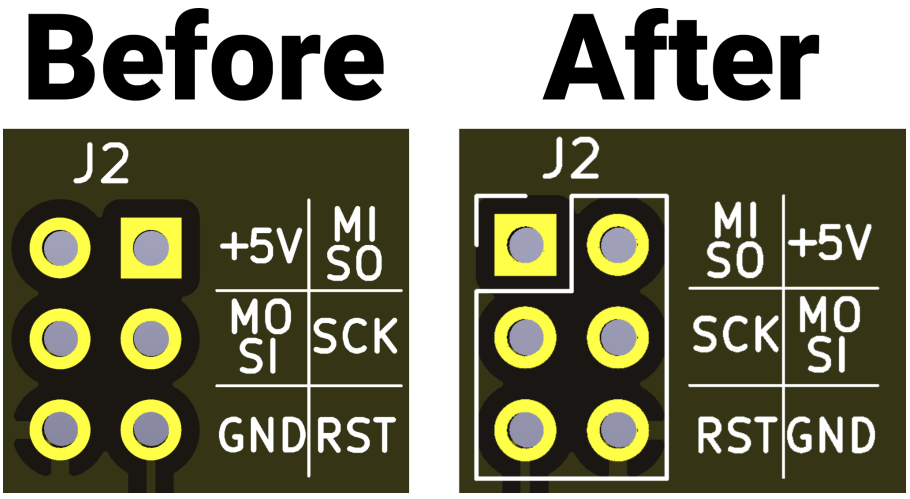
These diodes got a new location on the board. While D13 remained in the same orientation, D62 had to be rotated 90° to accommodate the space available.



Two of the diodes have been moved

[8.2] ICSP header

As explained in [5.4], the ICSP header is used to connect an Arduino to the eLiXiVy, to burn the bootloader to the MCU. rev.1a had the header mirrored horizontally, compared to the default. rev.1 follows the default layout.



Note the horizontal mirror in the labeling

[8.3] Rotary Encoder/switch footprint

While rev.1a supports a switch or a rotary encoder in the top right, it doesn't support 5 pin switches. This is because of an oversight, in removing the drillholes for pin 4/5 of a mechanical switch. A 5 pin switch could still be used, the two extra pins just have to be snipped off.

This is explained further in [6.4].

rev.1 supports 5 pin switches in all locations, including the top right (which is either for a switch or rotary encoder).

[9] FUT: Frequently Used Terms

Microcontroller

MCU: Microcontroller Unit. See it as a microprocessor, with a few more components (such as memory) to make it a full computer. In this project it will refer to the ATmega32U4.
Arduino: The company behind the ATmega MCU powered microcontroller boards, such as the Arduino UNO.
Bootloader: A small piece of the MCU's memory, which allows the user to load firmware via USB. See this as the BIOS of the MCU.
Firmware: A program on the MCU that allows it to function as a device (QMK in this project).
Flashing/burning [the firmware/bootloader]: The process of installing either a firmware or bootloader to the MCU.
ICSP: In-Circuit Serial Programmer [header]. Headers that cables can be connected to, to burn a bootloader to an MCU that is soldered onto a PCB.
I/O: The Input/Output lines of an MCU (used for the keyboard matrix)

Mechanical keyboard components/concepts

Be sure to check out Eschew's introduction to mechanical keyboards, for a lot more in-depth keyboard information: https://linustechtips.com/topic/1214368-an-introduction-to-custom-mechanical-keyboards/
Check out the mini_cardboard build log, for some more detailed information on some of these concepts (particularly a matrix, its diodes and the MCU):
https://linustechtips.com/topic/1328547-mini_cardboard-a-4-keyboard-build-log-and-how-keyboards-work/

Keyswitch: a component that can be pressed down to connect two pieces of metal together, which in turn can be used to detect a keypress
Cherry MX: The largest/most well-known manufacturer within mechanical keyswitches.
Cherry MX-like: Keyswitches with a design based on Cherry MX switches, because of Cherry's expired patent on keyswitches. Includes companies like Kailh, Gateron and much more. PCBs and keycaps typically advertise compatibility with "MX-Like" switches, so indicate it works with all of those.
Matrix: The concept of efficiently using the I/O lines of an MCU, by using rows/columns to detect keypresses.
QMK: One of the most well-known open-source keyboard firmwares.
Plate: Holds the keyswitches in place together with the PCB. Either the PCB or case is mounted to the case. Available in aluminium, brass or polycarbonate which have different feelings. This is an optional part, but a recommended part.
Stabilizer (stabs): Makes sure the larger keys (2u and larger) are properly supported when pressed down. PCB mount stabs are supported.
Keycap: The (usually) plastic covers that sit on the keyswitches. Popular keycap materials include - but

are not limited to - ABS and PBT. Refer to Eschew's introduction to mechanical keyboards for much more info. A keyboard needs different sizes of keycaps, depending on the layout chosen.
Layout: The different sizes of the keycaps decide the layout of the keyboard. The eLiXiVy is a 65% size ANSI or ISO keyboard. Refer to [3.1] for more layout information.
Rotary Encoder: the eLiXiVy supports a rotary encoder, which is a knob that can be turned around, much like a volume wheel. Popular choices on how to use the rotary encoder include - but are not limited to - volume up/down, page up/down, (Shift+)+Alt+Tab and more.

Electronics

PCB: Printed Circuit Board. A board made from fiberglass, with its purpose in a keyboard being to hold the components in place and connecting them together with traces (wires on a PCB)
Ground: The 'return' part of electricity, such as Anti-static discharges.
Diodes: An electrical component that allows power to flow one way, but not another. Used in the keyboard matrix.
Short: When two different signal types on the board are connected together. Can be positive or negative. A short is usually not desired, such as accidentally connecting +5V and GND together.
Sometimes a short is desired. Shorting the RESET and GND pins together is exactly what the reset button does, which is the way the MCU resets.
THT: Through Hole Technology components have long legs that stick through a PCB and get soldered to a plate through hole to the board. These components include the switches, header pins and diodes.
SMT: Surface Mount Technology uses pads on the board to solder SMD (Surface Mount Device) components in place. SMD components can be smaller than THT components and because they don't stick through the board, it's possible to design a PCB with SMD components on both sides.

Soldering

Soldering: The process of joining two components together (like the PCB and component) with soldering tin
Soldering iron: A tool to heat up the two components, to make the soldering tin flow between them
Soldering station: A device that includes a soldering iron, but also has a base station which allows you to change the temperature of the iron.
Solder(tin): a metal (combination) used to melt between the components you want to solder together. Typically consists of Sn (tin) and Pb (lead) in a 60/40 ratio. Lead free options typically include - other than tin - metals like Ag (silver), Cu (copper), etc.
Lead-free/leaded solder: Leaded solder is easier to work with, because of its lower melting temperature, but additional safety precautions need to be taken and it may be more difficult to get.
Flux: A substance used to make soldertin flow better to the pads and flow away from the rest of the PCB.
Desoldering braid/wick: A braid from copper strands with flux in it, used to wick away soldertin. Used in drag-soldering too.
Drag soldering: A popular method to install a chip with a lot of pins, like the MCU. Place a lot of tin over the legs/pads and wick it away. More info in "The Build IV: The First* Board"
IPA: Isopropyl alcohol, a cleaning agent to clean off the flux from the board.
Tacking down: soldering one pad of a component to the board, so it's held in place when soldering in the other pad(s).

Software/Firmware/QMK

QMK: One of the most well-known open-source keyboard firmwares.
Git: A version control system, to make it easy to work together on a project, or just keep track of different versions in general
Github issue: Github is the chosen Git version control platform of this project.
Keymap: An inherent part of the firmware that decides what key outputs what and what the rotary encoder does. This is generated by QMK and can be adjusted to fit everyone's personal needs.

[10] License

[10.1] Changes

Please note dates in the ISO 8601 (YYYY/MM/DD) format. You may include your website/a way to contact you if you wish.

Revision: eLiXiVy_AssemblyGuide_rev1
Date: 2021/08/03
Change(s): Assembly guide created.
Author: minibois
Contact: https://github.com/minibois/

[11] Checklists

Section	Component Reference	Value	Checklist		Component Reference	Value	Checklist
[4.1]	C1	22pF			F1	500mA	
	C2	22pF			R1	10k Ohm	
	C3	1uF			R2	22 Ohm	
	C4	0.1uF			R3	22 Ohm	
	C5	0.1uF			R4	10k Ohm	
	C6	0.1uF			R5	5.1k Ohm	
	C7	10uF			R6	5.1k Ohm	
	C8	0.1uF					
[4.2]	SW1	Switch					
	U2	TVS Diode					
[4.3]	Y1	Crystal					
[4.4]	U1	Atmega32u4 controller					
[4.5]	J1	USB Type C connector					
[4.6.2]	D0 - D68	1N4148W					
[4.6.3]	D0 - D68	1N4148					
[4.7]	J2	2.56mm headers					
[6.1]	RE0	Rotary encoder					