



Uppsala University

---

# PRECISELY CONTROLLED DIY ETCHING MACHINE FOR USAGE AT HOME AND IN SMALL LABS

---

Project Report

for the project course  
of Embedded Control Systems (1RT911)  
carried out in the autumn term 2017 (HT17)

Deadline: 2018-01-12 12:00

**Student**

Nils Weber

(Nilsthorben.Weber.3030@student.uu.se)

**Student**

Maximilian Stiefel

(Maximilian.Stiefel.8233@student.uu.se)

**Supervisor**

Alexander Medvedev

(Alexander.Medvedev@it.uu.se)

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background and Analysis</b>	<b>3</b>
2.1	Market analysis . . . . .	3
2.2	LED Drivers . . . . .	5
2.2.1	Buck Converters . . . . .	5
2.2.2	Linear Regulators . . . . .	7
2.3	Temperature Control . . . . .	7
<b>3</b>	<b>Design</b>	<b>9</b>
3.1	System Architecture and Required Functionality . . . . .	9
3.2	Design Choices . . . . .	10
3.3	Mathematical Model of the Tank . . . . .	11
<b>4</b>	<b>Implementation</b>	<b>13</b>
4.1	UV Light . . . . .	13
4.1.1	Hardware . . . . .	14
4.1.2	Software . . . . .	16
4.1.3	Mechanics . . . . .	17
4.2	Etching Tank . . . . .	17
4.2.1	Hardware . . . . .	18
4.2.2	Software . . . . .	18
4.2.3	Mechanics . . . . .	19
4.3	Operating System . . . . .	21
<b>5</b>	<b>Evaluation</b>	<b>24</b>
<b>6</b>	<b>Conclusion and Future Work</b>	<b>26</b>
<b>A</b>	<b>Schematics Extension Board</b>	<b>IV</b>
<b>B</b>	<b>Technical drawings of PCB-holder</b>	<b>X</b>

# List of Figures

- 1.1 Layered structure of Printed Circuit Boards (PCBs). . . . . 1
- 2.1 Typical buck converter as it can be found in a car. . . . . 5
- 2.2 More complex buck converter for automotive applications with analog control circuit. . . 6
- 2.3 Typical linear regulator circuit. . . . . 7
- 2.4 Waveforms of heating power and temperature of a system with on-off control. . . . . 7
- 2.5 The control system with unity negative feedback and inertia corrector. . . . . 8
- 2.6 The control system with proportional and inertia correctors. . . . . 8
- 3.1 *μC* with user interface, custom LED driver and feedback loop. . . . . 9
- 3.2 *μC* with user interface, heater and feedback loop. . . . . 10
- 4.1 UV light system. . . . . 13
- 4.2 Four layer extension board for the Nucleo-F103RB. . . . . 14
- 4.3 Layered code structure . . . . . 16
- 4.4 Wooden box for the Ultraviolet (UV) exposur unit. . . . . 17
- 4.5 Adapter board for Light Emitting Diode (LED) matrix enables simple plug and play. . . . 17
- 4.6 Lid and floor of the etching tank. . . . . 20
- 4.7 The etching tank. . . . . 20
- 4.8 Structure of the Operating System (OS) . . . . . 21
- B.1 Technical Drawing: Crossbar . . . . . X
- B.2 Technical Drawing: Floor . . . . . X
- B.3 Technical Drawing: Holder . . . . . XI
- B.4 Technical Drawing: Lid . . . . . XI

# List of Tables

- 2.1 Market analysis. . . . . 3
- 3.1 Design choices for the system: Why has each component been chosen? . . . . . 11
- 5.1 Status Quo OS . . . . . 24
- 5.2 Status Quo UV light . . . . . 24
- 5.3 Status Quo etching bath . . . . . 25

# 1 Introduction

At some point in the career an embedded systems engineer needs a quick hardware prototype. One had an idea and wants to try it out. With modern Electronic Computer-Aided Design (ECAD) software the PCB design is ready within a week or even less. Leading electronic component distributors also guarantee delivery within an acceptable timespan of a few days. However, the preferred PCB manufacturer most likely needs a week for the manufacturing and another week for shipping. For this reason, if time is a limited resource, Do It Yourself (DIY) etching in the lab can be an option to consider.

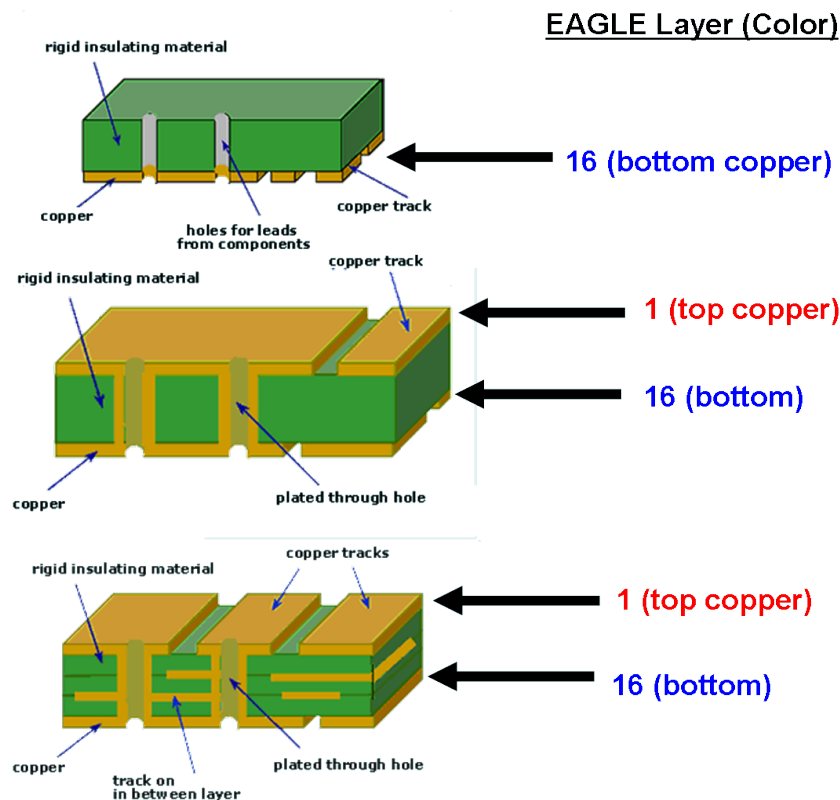


Figure 1.1: Layered structure of PCBs. Top: 1 layer. Middle: 2 layers. Bottom: 4 layers. Source: [online:pcbayers](#)

A PCB basically consists out of copper layers, substrate layers and vias to interconnect different layers (cf. fig. 1.1). There are three different kind of vias in modern electronics:

1. Through-hole vias as one can see in fig. 1.1
2. Blind vias, which only connect an outer layer to one (or multiple) inner layers
3. Buried vias, which connect two copper layers inside

Blind and buried vias can be implemented as micro vias, which means nothing, but that they are very small, because a laser is used to melt two layers together in a tiny ( $< 0.1$  mm) circular area. Conventional DIY etching techniques only support the first option. The other two options are usually nowadays still



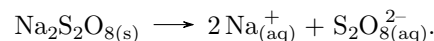
more expensive than normal through-hole vias, which have been around for quite some time already. Without reservations, using a DIY approach, PCBs with a complexity of two layers can be crafted nowadays. The basic idea of DIY PCB crafting is to remove the copper from a blank in certain areas to obtain the desired layout. Distributors nowadays offer blanks with one or two layers and substrate (with a thickness of usually 1.6 mm). Most common is the format 100 mm x 160 mm (*Eurocard*). Taking away the copper can be achieved mechanically and chemically, whereas this work is about the latter.

DIY etching is usually done in either of two ways:

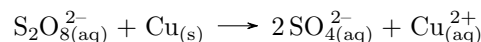
1. The toner transfer method: The layout is printed on glossy or transparent paper and the toner is transferred with heat to the blank copper board (cf. **online:instructtoner**). There are several severe disadvantages with this method e.g. that it is hard to align the masks for a PCB with two layers.
2. The photoresist method: Photoresist is used to cover the copper completely and then a UV lamp in combination with a mask printed on transparency is leveraged to destroy the photoresist, where it is not covered by the black color on the transparency (cf. **online:instructphoto**). The cracked photoresist, which is damaged by UV light can then be washed away with sodium hydroxid (NaOH).

In both cases an etching bath is necessary to etch away the unprotected copper. Popular chemicals to achieve the copper removal are iron(III) chloride ( $\text{FeCl}_3$ ) and sodium persulfate ( $\text{Na}_2\text{S}_2\text{O}_8$ ).

Sodium persulfate gives better results i.e. better or finer contours. Moreover, the sodium persulfate etching liquid is transparent blue unlike the liquid with iron(III) chloride, which is brown and non-transparent. Solving sodium persulfate in water results in



In the etching process elemental copper is oxidated to copper ions, which are then a part of the etching liquid.



Special care needs to be taken of the etching liquid afterwards. Copper ions are toxic for all kind of organisms. The last step of PCB crafting is the drilling. It can be challenging especially if one uses small via sizes (down to 0.2 mm is not uncommon). One also has to find solution or rather keep in mind, that a via is supposed to interconnect different PCB layers, which is hard to achieve creating PCB in a DIY manner as one somehow has to plate the inside of the cylindric via drill hole with copper in order to create a genuine via.

The goal of this project is to develop cheap and easy-to-use etching equipment for the photoresist method. It shall be very easy for engineers to replicate the equipment and also to even tailor it. All the design files shall be open and available on the net.

In principle, what is necessary to start crafting prototypes, is an etching tank and a UV light source. Both are subject to what this project opts to achieve as a result.

The whole project is pushed regularly on *Github*: <https://github.com/m3x1m0m/EmbeddedEtcher>

## 2 Background and Analysis

### 2.1 Market analysis

There are already different commercial and non-commercial solutions on the market to realize small-scale PCB etching. To find possible problems and optimization potential, in order to create a superior product, a small market analysis (see tab. 2.1) has been carried out in the run-up to the development phase of the project.

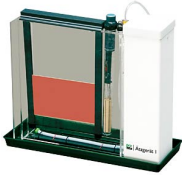



Product	Distributor / Manufacturer	Price	Technical Data
 AETZGERAET 1	Reichelt / PROMA	129.50 EUR (03.01.2018)	Needed Acid Volume: 1.75 l. Max. PCB Size: 235 mm x 170 mm. Heating Power: 100 W. Air Pumping System included.
 500-004 Universal Tank	RS Components / Mega Electronics	234.26 GBP (03.01.2018)	Needed Acid Volume: 5 l. Max. PCB Size: 315 mm x 260 mm. Air Pumping System included
 UV-BELICHTER 1	Reichelt / PROMA	219.7 EUR (03.01.2018)	Timer included (0 - 100 min). Max. PCB Size: 160 mm x 250 mm. 4 x 8 W UV tubes. Aluminium housing.
 UV Exposure Unit	RS Components / Mega Electronics	183.32 GBP (03.01.2018)	Timer included (0 - 6 min). Max. PCB Size: 150 mm x 245 mm. 2 x 8 W UV tubes. Metal housing.

Table 2.1: Market analysis.

Talking about the etching machines, it is a fact, that all of them do have a heater and some sort of pump to accelerate the chemical reaction introduced in chapter 1.

The *500-004 Universal Tank* from *RS Components* is quite big, usually too big for small scale production. For the etching process with sodium persulfate one uses  $220 \text{ g l}^{-1}$  with  $45^\circ\text{C}$  (cf. **online:aetzen**). Using the tank from RS Components one needs 1.1 kg of sodium persulfate, which is neither really practical nor



cheap, in case one only wants to etch a few PCBs to e.g. test a microstrip bandpass filter. Nevertheless, the design looks really robust and justifies somehow the tough price. Also the maximum PCB size is enormous or rather a bit too big for prototyping.

The *AETZGERAET 1* from *Reichelt* has a way more competitive price, than its equivalent from *RS Components*. It also seems to be a bit more bodge. The maximum PCB size is still impressive as well as the heater power.

Presumably, there is not really a sophisticated control mechanism keeping the temperature close to the setpoint. As far as one can derive this from the given technical data, the thermostats of both devices are based on a bimetal.

The UV lights are both functioning with UV fluorescent tubes. Fluorescent tubes are considered hazardous as they contain mercury vapor, which is extremely toxic (cf. **online:tubes**). For this reason there are plans to ban them in the European Union. It also seems, that one pays for the UV light power (number of tubes).

Both UV light exposure machines have a timer to set the exposure time. This is clearly an important and necessary feature for such a machine as the development process parameters define the quality of the final etched result. Hence, the UV light exposure time is an important factor to fine tune the etching result.

Two features neither the *UV Exposure Unit* from *RS Components* nor the *UV-BELICHTER 1* have: It is not possible to adjust the light intensity and the light does only come from one direction at a time. The second shortcoming can introduce problems, since one has to turn on the light, turn it off, turn the PCB around and repeat the exposure process. Worst-case, this might lead to alignment problems (e.g. a mounting hole on one side does not align with a mounting hole on the other side).

The exposure units are both relatively pricey, which might be related to the metal housing.

To summarize one can name some ideas how to rectify shortcomings of the currently available products for prototype PCB etching:

- The etching tank shall not be bigger than 2l, as one needs more etching liquid, which is unnecessary for small-scale production.
- No expensive metal housings.
- It shall be possible, that the user or replicator picks the heater and the compressor for air bubbles flexibly.
- No mercury.
- Double-sided UV light exposure.
- Precisely controlled light intensity.
- Digital and elegant user interface.



## 2.2 LED Drivers

### 2.2.1 Buck Converters

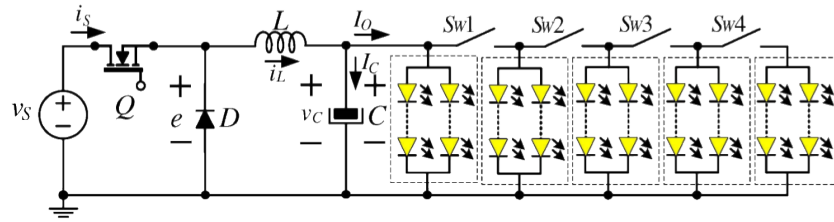


Figure 2.1: Typical buck converter as it can be found in a car. Source: **conference:buck1**

One of the ideas for improvements mentioned in section 2.1 induces the use of LEDs. Therefore, in this section different state-of-the-art LED driving techniques are introduced. To drive one LED or a string of LEDs coupled in series one needs to keep the current constant for achieving constant light intensity. It is a well-known fact, that every single LED differs concerning its electrical characteristics. The input to output transmission behavior i.e.  $I(U)$  differs from LED to LED. This is one challenge one has to master building a LED driver.

A common approach, which is popular in automotive applications is a *buck converter* (cf. **conference:buck1**). *Buck converters* leverage a switching-frequency to convert a high voltage into a lower voltage. In fig. 2.1 one can see, that in a car the current, which is drawn by the source can change rapidly, as for instance a rear light or an indicator is switched on. For this reason, a controller is needed to ensure, that enough current is provided even if rear light and indicator are switched on simultaneously. Buck converters are that popular because of their remarkable efficiency (greater than 90 % is normal).

A buck converter from an electrical point of view switches between two states i.e. transistor  $Q$  is conducting (1) or non-conducting (2):

1. If  $Q$  conducts  $L$  is directly connected to the voltage source.  $C$  and the load is connected to  $L$ . The energy stored in  $L$  increases. The current change results in a voltage across  $L$ , that counteracts the voltage of the source. This of course reduces the voltage over the load.
2. If  $Q$  does not conduct  $D$  becomes conductive, because  $L$  is now a voltage source. The energy stored in the coil provides a current through the load.  $L$  discharges and the voltage across  $L$  drops again. The current through the load is still maintained.  $C$  helps to keep voltage and current through the load stable (close to Direct Current (DC) although a certain ripple is normal).

If the coil is charged and discharged quickly enough one can get into a steady state, where the output is within a certain range. The voltage is ramping up to a maximum and ramping down to a minimum. The output and possible input filters take care of converting this signal to a DC signal.

One can also see the buck converter as a low-pass filter connected to a source with a square wave signal. The low-pass filter has ideally a cut-off frequency very close to 0 Hz. The filter excitation happens with a Pulse-Width Modulation (PWM) signal. One can easily explain how the steering of the current or voltage respectively works looking at the Fourier transformation of a rectangular pulse with a pulse width of  $2t_0$ . The pulse is defined as

$$x(t) = \begin{cases} 1 & -t_0 \leq t \leq t_0 \\ 0 & \text{otherwise} \end{cases} \quad (2.1)$$

This function can be brought into the frequency domain using the most known equation of the heritage of Joseph Fourier.

$$X(\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j\omega t} dt = \int_{-t_0}^{t_0} 1 \cdot e^{-j\omega t} dt \quad (2.2)$$

Using  $\int e^{at} = \frac{1}{a}e^{at}$  leads to

$$X(\omega) = \frac{1}{-j\omega} [e^{-j\omega t_0} - e^{j\omega t_0}] \quad (2.3)$$

As the next step Euler's formula can be applied.

$$X(\omega) = \frac{2\sin(\omega t_0)}{\omega} = \frac{2t_0 \sin(\omega t_0)}{\omega \omega t_0} = 2\text{sinc}(\omega t_0) \quad (2.4)$$

So what one ends up with the sinc function. As one tries to filter out every frequency except of 0 Hz, one can take a close look at  $\omega = 0$  with the help of l'Hôpital's rule.

$$\lim_{\omega \rightarrow 0} X(\omega) = \lim_{\omega \rightarrow 0} \frac{2\sin(\omega t_0)}{\omega} = \lim_{\omega \rightarrow 0} 2t_0 \cos(\omega t_0) = 2t_0 \quad (2.5)$$

Hence, the absolute value of  $X(\omega = 0)$  becomes larger if the pulse length  $t_0$  becomes wider. This is why PWM steering works.

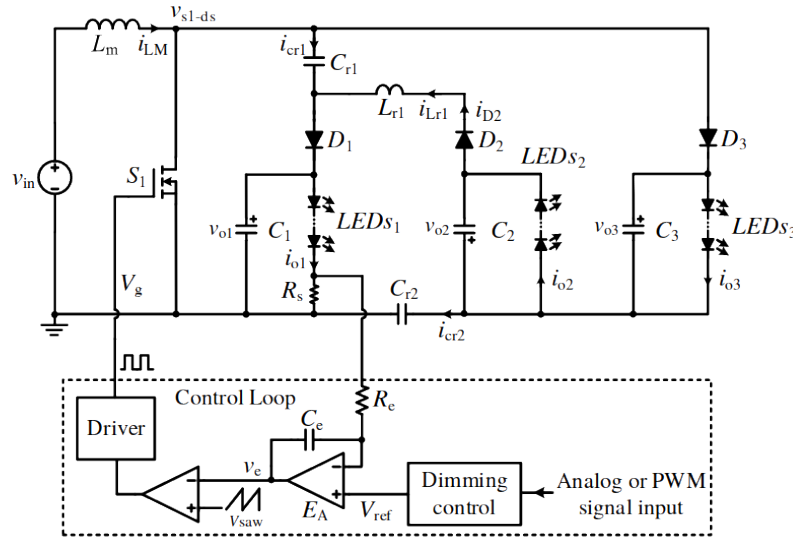


Figure 2.2: More complex buck converter for automotive applications with analog control circuit. Source: **article:buck2**

In fig. 2.2 one can see a more complex draft of a buck converter with a analog controller. This circuit originates from **article:buck2** The first amplifier of the controller is an integrator. It adds memory to the control circuit. It also achieves, that the steering signal is somewhat proportional to the setpoint, which is what one can see analyzing the behavior:

$$V_e = -\frac{V_{ref}}{C_e R_e} \int V_{in} dt + A \quad (2.6)$$

whereas  $A$  is some constant.  $V_{in} = R_s \cdot i_{o1}$  is according to Ohm's law proportional to the current through the load (and a shunt resistor). The output of the first amplifier is fed into the second stage, a comparator. It compares the input with a sawtooth signal, which results in a PWM with a duty cycle depending on the output of the first integrator. As the circuit analysis shows one deals with a proportional-integral controller here.

Despite their tremendously superior efficiency, there are some disadvantages with buck converters like the switching frequency, that can cause Electromagnetic Interference (EMI) problems. Comparing the different possibilities of buck converters one thing also becomes quite clear: Buck converters are complex. Moreover, they always need magnetic devices. Hence, designs become bulkier, less tightly integrated and more expensive if buck converters are used unnecessarily in places where they do not belong. For this reasons, linear regulators still have the right of existence nowadays.

## 2.2.2 Linear Regulators

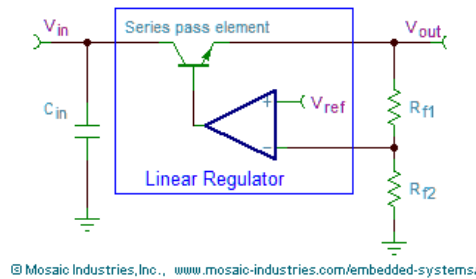


Figure 2.3: Typical linear regulator circuit. Source: [online:linreg1](#)

The other class of LED drivers (power regulators) are linear regulators, which simply adjust their resistance in order to provide a constant current or voltage respectively. Unfortunately, a resistor converts electrical energy into heat, which is why linear regulators suffer from inefficiency, yet, they are very simple and one gets rid of possible EMI problems caused by the switching frequency, that buck converters need. In fig. 2.3 one can see a typical circuit of a linear regulator to provide a fixed voltage for a varying load impedance. This is a pure proportional controller.

## 2.3 Temperature Control

Two approaches have been considered. Resources here can be money and time spent during development or operation.

1. Put more resources in sensors and save resources in the system model.
2. Put more resources in the controller model put save resources for sensors.

The first approach was largely inspired by [article:revLag](#) where the application of the reversed lag principles are discussed for on-off control. A typical on-off control would result in cyclic temperature fluctuations like shown in fig. 2.4.

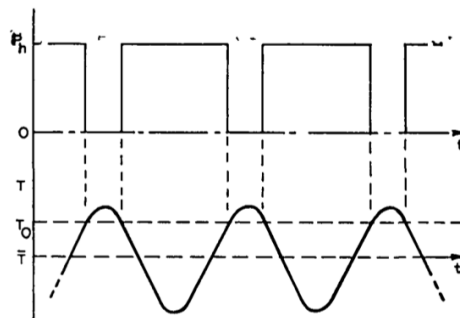


Figure 2.4: Waveforms of heating power and temperature of a system with on-off control. Source: [article:revLag](#)

The paper shows that such a system can be improved by using two control elements which are completely immersed in the fluid of the enclosure. Thus, they are not affected by different errors. One of the two output signals has to have a lagging phase angle, which can be achieved by enclosing it "within a structure which ensures that its mean temperature is that of the fluid, but thermal transmission through the structure introduces a phase lag into the fluctuating component of the temperature of the element" [article:revLag](#) However, the circuitry of both sensors results in a leading phase angle of the enclosed sensor with respect to the directly immersed sensor. Thus, the next temperature can be "foreseen" without the need of a system model. The paper concludes that "the principle of reversed lag can be applied to an on-off temperature-control system with useful but not spectacular reductions in the amplitude of the



cyclic fluctuation in temperature” **article:revLag** Furthermore, two sensors would need to be purchased to acquire essentially the same data point, which spends more resources and thus contradicts the idea of a low-cost DIY solution for small labs or at home.

On-off controllers are known for their energy efficiency of their output systems, compared to amplifiers. That is especially true for high power control signals. Electro-heating systems, like the here considered system, are an example for such a system, which is characterized by large time constants. That lowers the need for high frequencies to keep proper parameters of the controlled signal further. The system’s dynamics can be approximated with the following equation, where,  $k$  is the static gain,  $L$  is the time delay constant and  $T$  is the system time constant **article:onoffPLC**

$$G(s) = \frac{ke^{-sL}}{1 + sT} \quad (2.7)$$

A simple on-off controller leaves an static error, which can be reduced by increase of the controller hysteresis. However, this results in increasing the amplitude and period of the oscillation of the controlled signal. This is the disadvantage of this approach.

The paper **article:onoffPLC** further introduces an inertia corrector, to counteract the described disadvantage. ”The hysteresis  $h$  of the controller is the only parameter that can affect the control quality in the basic control system.” The new system model with inertia correction is shown in fig. 2.5. If the time constant of the inertia correction is much smaller than the time constant of the system, ”then the controller with connected corrector exhibits features of proportional-derivative controller relative to the mean value of controlled signal” **article:onoffPLC**

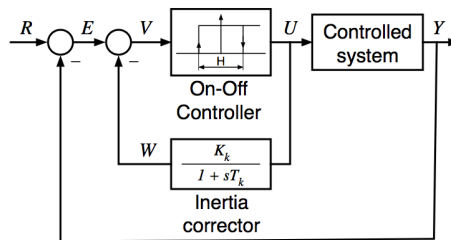


Figure 2.5: The control system with unity negative feedback and inertia corrector. Source: **article:onoffPLC**

However, as the authors of **article:onoffPLC** show, the steady state error can be eliminated by using an additional corrector, instead of an integrator. This proportional corrector rescales the reference value to the range of 0 to 1, before entering the control comparator. The system has to be pre-identified, but the introduction makes the mean value of the controlled signal independent from any changes of the hysteresis **article:onoffPLC**

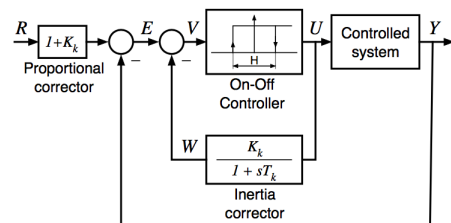


Figure 2.6: The control system with proportional and inertia correctors. Source: **article:onoffPLC**

The paper **article:onoffPLC** concludes that the introduced control system ”significantly reduces an amplitude of oscillation of controlled signal and shortens its period. An additional proportional corrector eliminates an error of the mean value of the controlled signal in the steady state.”

# 3 Design

## 3.1 System Architecture and Required Functionality

First, the system consists out of two machines, the UV light and the etching tank. Of course, these two machines are different and therefore need different designs. However, effort has been put into trying to use as many software and hardware components in both parts simultaneously to reduce the overall workload.

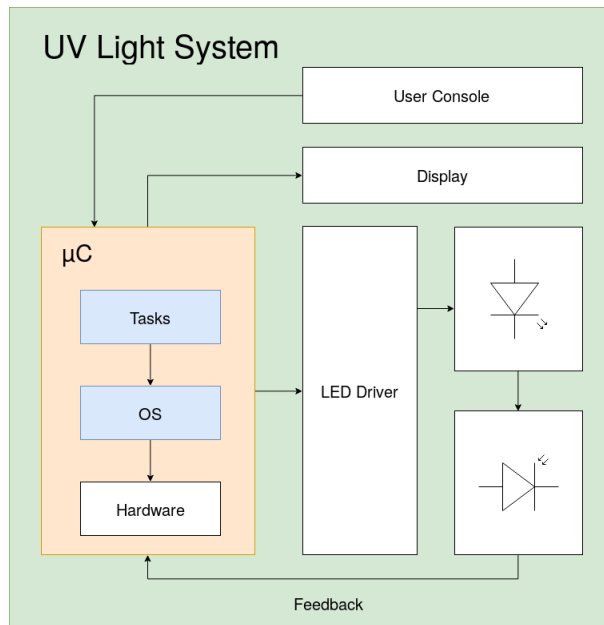


Figure 3.1:  $\mu C$  on the left executing tasks with an OS below. User interface is hooked up to the  $\mu C$ . Custom LED driver is connected to the LEDs.

In fig. 3.1 a rough sketch of the UV light exposure unit is shown. The most important part is probably the LEDs and the driver for it. A feedback from the LEDs is generated with the help of a light sensor. This feedback can be coupled into a controller steering the driver. Besides the LED driver, one needs a microcontroller, that executes the OS and tasks to implement the desired functionality. In addition, some components for communication with the user are necessary, i.e. a console and a display. The user must be able to adjust exposure time and intensity. The intensity is not changeable with devices currently available on the market.

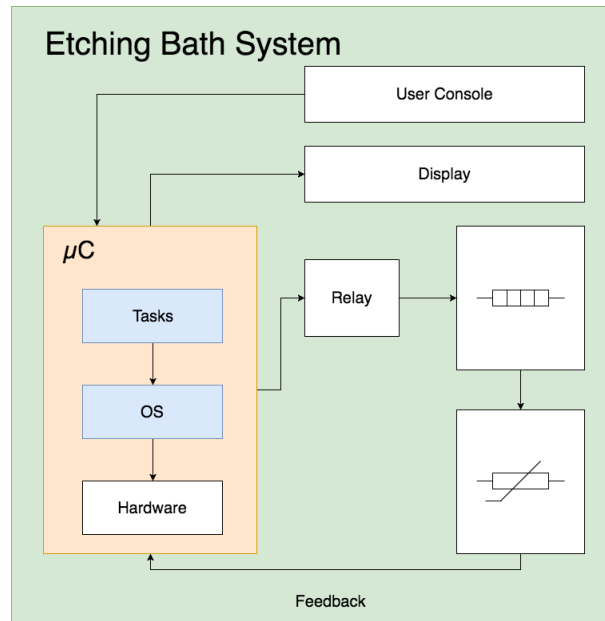


Figure 3.2:  $\mu C$  on the left executing tasks with an OS below. User interface is hooked up to the  $\mu C$ . The heater is connected via a relay.

In fig. 3.2 the system architecture is sketched. Comparing to fig. 3.1 one can see many similarities, which result from the efforts to keep many parts of the system interchangeable. The heater is not very sophisticated, thus the control is realized via a simple, optocoupled relay. An Negative Temperature Coefficient (NTC) resistor is used as a temperature sensor and submersed into the etching bath. Thus, it closes the feedback loop for the control system.

Tab. 3.1 gives a bit more insights dealing with which components have been chosen and why. Also it includes information about the costs of each component.

## 3.2 Design Choices

The following table explains why different components have been chosen to enable the required functionality.



Block	Component(s)	Costs	Justification of the Choice
Microcontroller	STM32 microcontroller family	From 1.30 USD	The $\mu$ Cs are cheap, because they are used everywhere. Open-source Integrated Development Environments (IDEs) are available and have a huge community. Open-source hardware is available as well. Big family tree with all kind of options. Powerful ARM Cortex-M0, Cortex-M3, Cortex-M4, Cortex-M7 32-bit processors. Ultra-low power (L-models) available.
LED Driver	Custom designed linear regulator	0.40 USD per stage (25.60 USD for 64 stages)	One needs less components with a linear regulator (saves space and money). Choosing the voltage supply as close as possible to the voltage drop over the LED or the LED string, the efficiency is not too bad. Control part becomes very simple, yet effective.
Display	Organic LED (OLED) SSD1306	2.50 USD	The display is cheap. Open-source driver software only needs to be ported to the platform. Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I2C) interface available. OLED technology is very elegant in comparison to alternatives.
User Console	Two rotary encoders and one tactile switch	2 USD	Very comfortable interface. One encoder for the exposure time, one for the light intensity and the tactile switch for starting the exposure process.
	One rotary encoder including a push button	< 1 USD	Different parameters of the controller can be changed via the rotary encoder; the button is to switch between changing the parameter value or the parameter.
Light Sensor	TSL2561	1 USD	Very easy to use with I2C interface. Integrated amplifier. Very small. No need to develop a readout circuit with a photodiode.
Temperature Sensor	NTCALUG01T103FL	3.5 USD	Simple NTC in metal package with ring connector. Robust, used in automotive applications. Comes with 150 mm of wire. Detailed support from manufacturer.
Relay with optocoupler	-	< 1 USD	Simple and safe to use via digital out.
Air Temperature Sensor	SHT31	4 USD	Highly accurate air temperature and humidity sensor. Includes I2C-interface.

Table 3.1: Design choices for the system: Why has each component been chosen?

### 3.3 Mathematical Model of the Tank

Using the concept of energy balance to describe the thermodynamical model of the tank yields following equations **online:mathModel**

$$Heat_{in} = Heat_{out} + Heat_{stored}$$

$$q_i = \frac{\theta_t - \theta_e}{R_{ta}} + c \frac{d\theta_t}{dt} \quad (3.1)$$

with  $c$  as heat capacity of water,  $4184 \frac{J}{kgK}$ , and  $R_{ta}$  as temperature resistance between tank and ambient environment.



To further develop a model, different measurements are necessary. Water has a different specific heat capacity in different phases, i.e. as ice or water **online:watercurve**

$$q = m * C_s * \Delta T \quad (3.2)$$

Different measurements to gather data to further build and understand the mathematical system. All measurements are to be done with the original water tank, heater and the temperature sensor.

- Cooling behavior
  - Measure temperature every 10s while cooling down water
  - Start from different temperatures
- Heating behavior
  - Measure temperature every 10s while heating up water
- Determine time constants



# 4 Implementation

## 4.1 UV Light

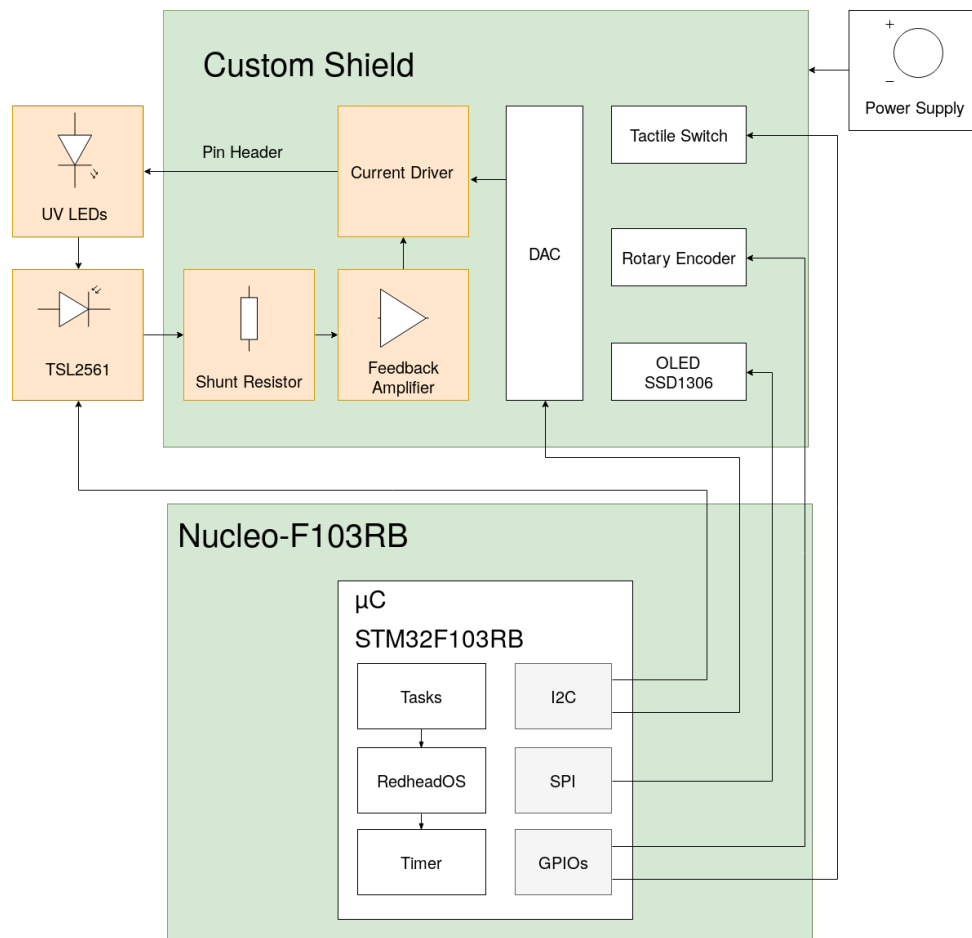


Figure 4.1: UV light system. Top: Custom extension board. Bottom: Nucleo board with custom software.

The decision has been made to use the *Nucleo-F103RB*, a cheap (approx. 25 USD) development board, instead of designing a processor board. This has a couple of advantages:

- No PCB design required to bring the STM32 on a board.
- Directly start programming without waiting until the hardware is ready.
- Debugger and programmer from STM included.
- Time to develop something, that works is shortened.

Instead, a custom extension board for the Nucleo board has been designed (cf. fig. 4.1). Everything has been laid out so the hardware can drive a lot of *industry standard T-1 3/4 5 mm LEDs*. In fact it can



supply 256 of these LEDs. Four LEDs can be put in series. The voltage drop does not exceed 14 V. This has been verified by measurements in the lab with a current of 30 mA. Hence, a power supply of 18 V is sufficient. Although something closer to 14 V would be more efficient. However, it does not matter so much, as long as the linear regulator can dissipate enough power without being destroyed.

The LEDs are meant to be connected with a 64 pin header (low-side) and a 10 pin header to the board. Actually, a rather big housing has been built together to take in the LEDs and the electronics eventually.

A small OS, developed from scratch, runs on the STM32 to make real-time scheduling possible with a small code footprint and give a programmer all the necessary infrastructure to program conveniently (semaphores, queues, printf etc.).

### 4.1.1 Hardware

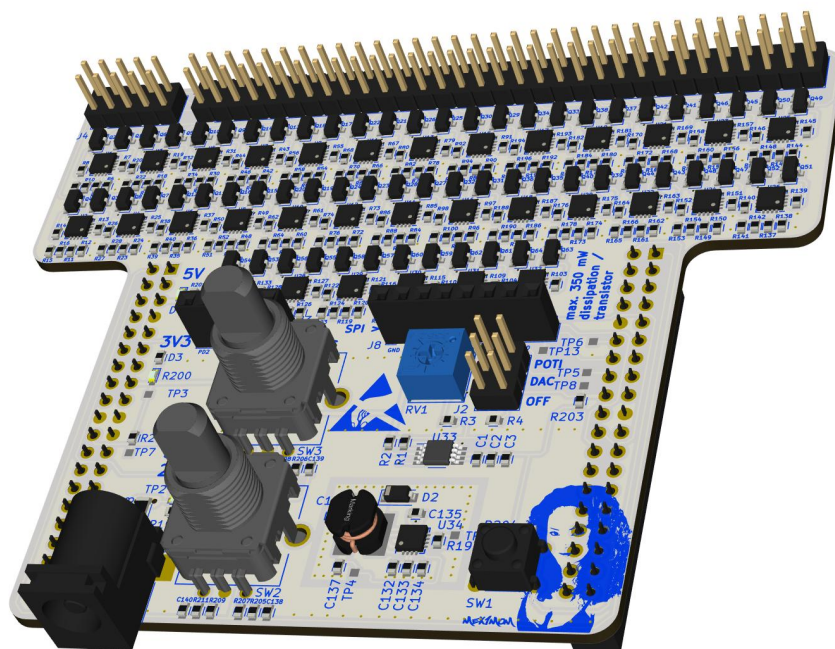


Figure 4.2: Four layer extension board for the Nucleo-F103RB to drive 64 LED channels. All 3D models have been entered into the ECAD program to prevent unintended surprises.

The board shown in fig. 4.2 has been developed and produced in the course of this project. It has four layers, but no micro vias, to, on the one hand get a grip on the complexity, and on the other hand save money (micro vias are still expensive than through-hole vias in production, cf. 1). As an introduction, one can briefly list the features of this sophisticated piece of hardware.

- 64 LED driver stages.
- 6.4 A (0.1 A per stage) and 24 V max. input.
- 0.2 A max. and 350 mW dissipation max. per stage. Stages can be left floating if unused without the threat of damaging the device.
- 10 Bit Digital-to-Analog Converter (DAC) to steer the driver combined with feedback amplification to achieve a tremendously fine resolution.
- Optional potentiometer to set the current.
- Control loop to guarantee the desired current and respond quickly to changing the setpoint.
- 2 rotary encoders and 1 tactile switch.



- I2C header.
- SPI header.

In appendix A an extract of the schematics of the board is available. The following lines shall briefly describe the different circuits. On page 1/37 one can see, that there are four main modules: The interface to the processor board, the LED driver, the power management and the user interface.

On page 2/37 one can see the LED driver part. On the right side, the *DAC101C085* from *Texas Instruments* is placed.  $0\Omega$  resistors give flexibility in setting the bus address. The reference voltage is connected to the supply voltage of  $3.3\text{ V}$ , provided by the processor board. Below that, one can see a potentiometer and a connector, that can be populated with a jumper to set the reference voltage for the driver accordingly. The reference voltage is translated into a current whereas with the resistor configuration so far  $1\text{ V}$  equals  $10\text{ mA}$ . The last connector on the right side is to make the I2C interface 1 of the microcontroller accessible. On the left side there are 32 times 2 driver stages, which equals a total of 64 driver stages. The group forming is as it is, because the deployed quadruple operational amplifier (*LM324*) can be used for 2 stages as it has four operational amplifiers inside. In addition, one can see the big 64 pin header for LED connection.

Page 3/37 reveals the LED driver design as it shows one driver stage, which consists out of one *LM324* (four operational amplifiers) and two *MBT3904* (NPN transistor). The controller works as follows:

1. A current  $I_C$  flows through the transistor into the shunt resistor  $R_S$ , which is connected to the input of a non-inverting amplifier.
2. The shunt resistor  $R_S$  has a small resistance value of  $10\Omega$  and translates the current  $I_C$  into a voltage  $V_S = I_C \cdot R_S$ . This might even work with a smaller shunt resistor.
3. Obviously, the resulting voltage  $V_S$  is very small, which is why there is an amplifier in the feedback loop. Analyzing this operational amplifier configuration one ends up with the following relation between input and output:

$$V_{out} = V_{in} \frac{R_1 + R_2}{R_1} = V_{in} \frac{20\text{ k}\Omega + 180\text{ k}\Omega}{20\text{ k}\Omega} = 10V_{in} \quad (4.1)$$

The size of the shunt resistor and the amplification factor determine the mapping between bits set in the DAC (10 Bits) and the current through the LED. Making the shunt resistor smaller is of course better because of unnecessary heat dissipation, but one has to keep in mind, that the operational amplifier is absolutely low-cost and has a certain offset voltage. Also, the closer one gets to the power supply voltages of the operational amplifier, the more problems of non-linearity occur.

4. In the last step, the voltage output of the first stage is compared with the setpoint, coming either from the DAC or the potentiometer. The output of the second stage is a current  $I_B$  proportional to the difference between the inputs.

$$I_C = I_B \cdot \beta \quad (4.2)$$

is the relation between the base current and the collector current. According to the datasheet  $\beta$  is somewhere between 100 and 300, which is why a small error is neglected, since  $I_C \gg I_B$ : The current into the transistor collector is not the same as the current going out of the emitter. Hence, the current through the LEDs is about less than 1 % smaller than the current the controller is aware of.

The power distribution becomes clear taking a look at page 35/37. Using a barrel jack a lot of standard power supplies can be interfaced with the board. The board design is *IPC 2221* compliant given the parameters one can see in the schematics. A powerful buck converter Integrated Circuit (IC) the *TS30011* steps down the input voltage to  $5\text{ V}$ . The processor board can be powered with external  $5\text{ V}$ , assumed that configured correctly.  $3.3\text{ V}$  can then be obtained from the processor board to supply e.g. the DAC or the OLED screen on the extension board.

Page 36/37 shows the interfacing between processor and extension board. Nothing to really see here.

The small user interface is on the last page 37/37. One connector enables access to the SPI 1 of the microcontroller. Also one finds the tactile switch and two rotary encoders, which are debounced hardware wise.



### 4.1.2 Software

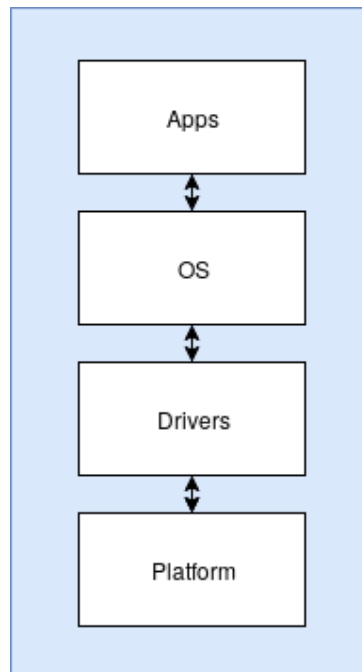


Figure 4.3: Layered code structure

Fig. 4.3 shows how the code ought to be structured in general. This allows to migrate the actual functionality quickly to another platform. However, as it turned out it can be hard sometimes not run into layer breaches i.e. the code one layer is supposed to only communicate with the layer above and the layer below if implemented properly.

Every kind of controller is implemented in the *App layer*. The OS layer provides a Proportional Integral Differential (PID) controller infrastructure. A task is running, which handles the controller for the UV light intensity, that gets the feedback input from the light sensor and steers the DAC.

The DAC driver is implemented in the *Driver layer*, whereas this driver needs to use I2C, which is a hardware component of the *STM32*. I2C is for this reason located in the *Platform layer*. Switching to another microcontroller, one ideally only needs to change the *Platform layer*.

The OLED screen driver belongs into the *Driver layer* accordingly and there are tasks to handle the user input from the rotary encoders and the switch in the *App layer*.

### 4.1.3 Mechanics



Figure 4.4: Wooden box for the UV exposurer unit with 16 x 16 LED matrix.

In cooperation with a carpenter a wooden box has been build, which uses a glass plate from an old scanner (cf. fig. 4.4). A wooden plate is located below the glass. On this wood plate there is a matrix out of 256 LEDs. The distance between glass and LED matrix can be varied with the help of IKEA parts, that are plugged into the sidewalls while simultaneously holding the LED matrix. The cabling, drilling and soldering to create the LED matrix is a nightmare as one can imagine. Each single LED needs to be connected to wires to lengthen the legs of it. Subsequently, every LED is fixed in a hole and fixed with a hot glue gun. Always four LEDs have to be connected together in series. Eventually, every single one of the 64 stages has to be connected somehow to the extension board. For this reason a small adapter PCB has been constructed (cf. fig. 4.5). This board can be installed with wood screws and high- and low-side of the LEDs can be soldered. Following, one just has to plug in the two ribbon cables.

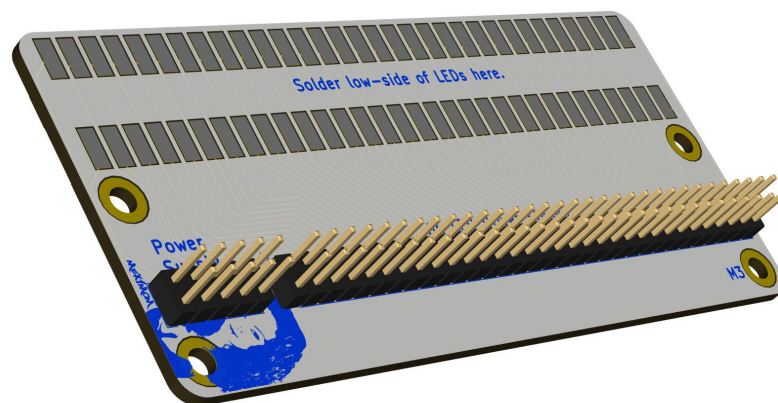


Figure 4.5: Adapter board for LED matrix enables simple plug and play after soldering high- and low-side of the LED strings.

## 4.2 Etching Tank

As briefly outlined in section 4.1, the Nucleo-F103RB has been chosen. During the development of the temperature control the Nucleo-L073RZ has been used for the simple reason of immediate availability. This led to more complications that originally anticipated, as discussed later in chapter 5.



### 4.2.1 Hardware

Less sophisticated is the electric hardware for the temperature control, as it is in early prototyping stage to date and future tests will provide knowledge to chose the smartest path in further developments. As shown earlier in fig. 3.2 the system output is a 50W heater which is controlled via a relay. Its power source is a common power outlet of 230VAC. This relay is optocoupled to a normal digital output GPIO for security reasons. For the user interface a small OLED screen with the SSD1306 driver and I2C interface is used as output, and a rotary encoder is used as input. The screen shows the temperature inside the etching bath, the reference temperature and the ambient temperature. Since no cooling mechanism is installed, the reference temperature cannot be set below the ambient temperature.

### 4.2.2 Software

Core of the system is the control algorithm implemented as described in **article:onoffPLC** The algorithm shown in fig. 2.6 has been implemented in *c* and is briefly described in the following. The names of the variables are chosen according to **article:onoffPLC**

A simple on-off controller is of course the heart of the algorithm. It determines a binary output based upon the corrected control error  $V$  and the hysteresis  $H$ .

```
1  uint8_t onOffController(double iV, double iH) {
2      uint8_t oU = 0;
3
4      if (iV >= (iH/2)) {
5          oU = 1;
6      }
7      else if (iV <= (iH/2)) {
8          oU = 0;
9      }
10
11     return oU;
12 }
```

The above described on-off controller is embedded in the following function, which would be called from the main program or called in a periodic task. At first, the input parameters are re-scaled using the function `u_map()` shown further below. The negative feedback is re-scaled to a ranged from 0 to 1, while the reference value is scaled to 0 to 100. After the re-scaling the algorithm computes the proportional correction and calculates the control error based upon that value. Next, the inertia correction is applied to the current value using the inertia correction value from the previous computation. Now the on-off controller is called with the modified values from proportional and inertia corrections. As a last step before returning the binary output value, the next inertia correction value is computed calling the respective function described below.

```
1  uint8_t controller(uint8_t _nowT, uint8_t _aimT) {
2      double deltaT = 1; // HAS TO BE DETERMINED!!
3      double K_k = 10.0;
4      double T_k = 0.01;
5      double H = 1; // HAS TO BE DETERMINED!!
6
7      double A = (deltaT*K_k)/T_k;
8      double B = 1-(deltaT/T_k);
9
10     static double nW; // keep inertia correction value
11
12
13     // re-scale input parameters
14     double nY = ((double) u_map(_nowT, 0, 100, 0, 100))/100;
15     double nR = (double) u_map(_aimT, 0, 100, 0, 100);
16
17     // proportional correction
18     double nP = nR * (1+K_k); // C = 1 + K_k
19 }
```



```
20 // calculate error
21 double nE = nP - nY;
22
23 // apply inertia correction
24 double nV = nE - nW;
25
26 // call on-off controller
27 uint8_t bU = onOffController(nV, H);
28
29 // call inertia correction
30 nW = inertiaCorrection(bU, nW, A, B);
31
32 return bU;
33 }
```

Following equation is implemented in the inertia correction function.

$$W_{t(i)} = AU_{t(i-1)} + BW_{t(i-1)} \quad (4.3)$$

where

$$A = \frac{\Delta t K_k}{T_k} \quad (4.4)$$

$$B = 1 - \frac{\Delta t}{T_k} \quad (4.5)$$

```
1 double inertiaCorrection(uint8_t fU, double nW, double A, double B) {
2     double UA, WB;
3     UA = ((double)fU)*A;
4     WB = nW*B;
5
6     return (UA+WB); //nW
7 }
```

```
1 uint32_t u_map(uint32_t x, uint32_t in_min, uint32_t in_max,
2               uint32_t out_min, uint32_t out_max) {
3     return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
4 }
```

Other than that, the rotary encoder is implemented to easily adjust the wanted temperature. Both, the realtime and wanted temperature are shown on the display via I2C.

### 4.2.3 Mechanics

The etching tank has been produced of a mix of glass and additive manufactured plastic parts to add functionality and form. The lid and floor are shown in fig. 4.6. Between lid and floor is a tank made of glass, seen in fig. 4.7. Inside this tank, hanging from the lid, is a U-shaped system to hold the PCB. Its size is adjustable to the PCB and maximum 148 mm x 160 mm. The technical drawings are attached in appendix B. On the right side one can also see the heater and in the bottom a thinner black tube. This tube is connected to a compressor and ejects the air bubbles to speed up the removal of copper from the PCB.



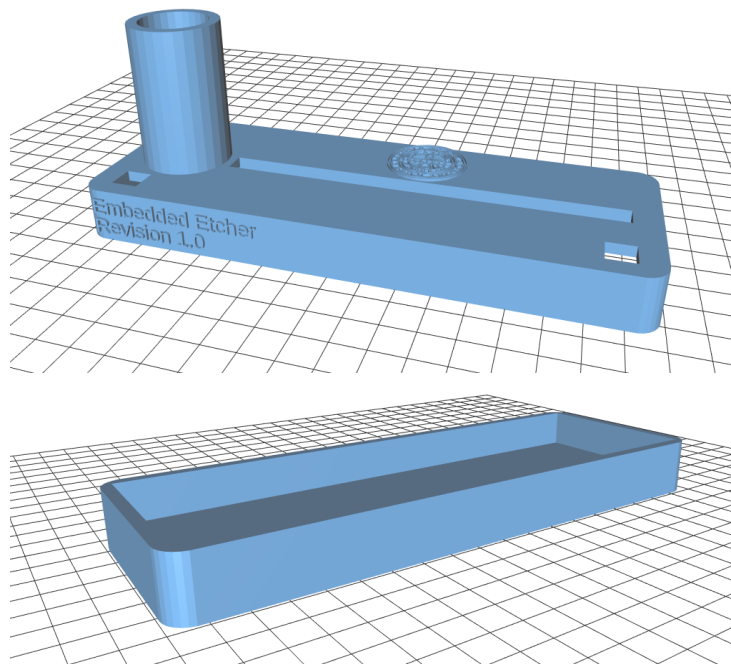


Figure 4.6: Lid and floor of the etching tank.

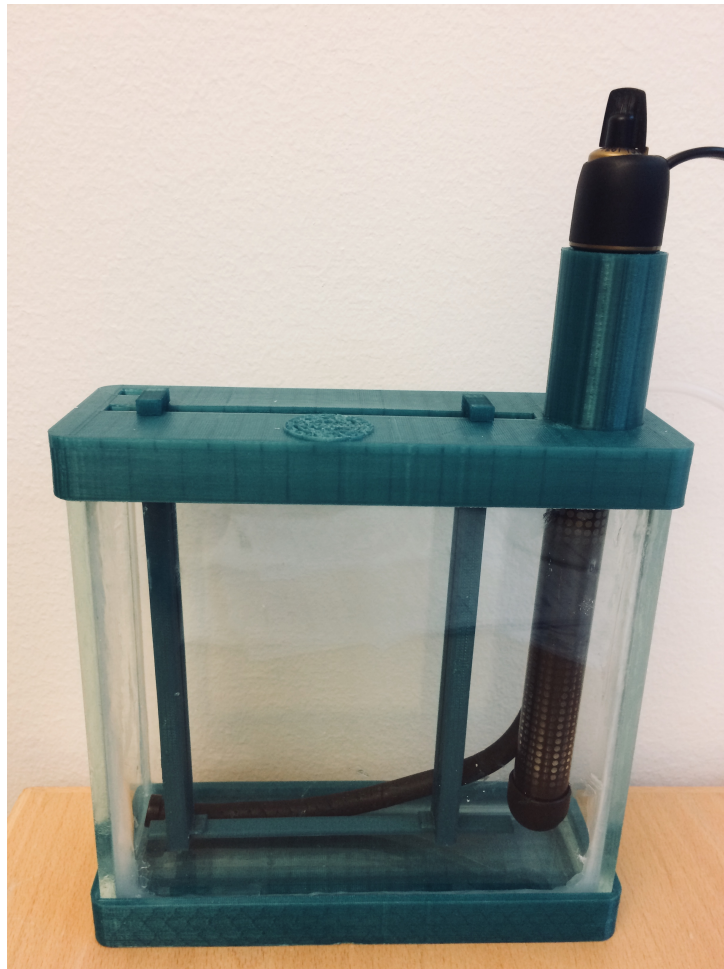


Figure 4.7: The etching tank.





## 4.3 Operating System

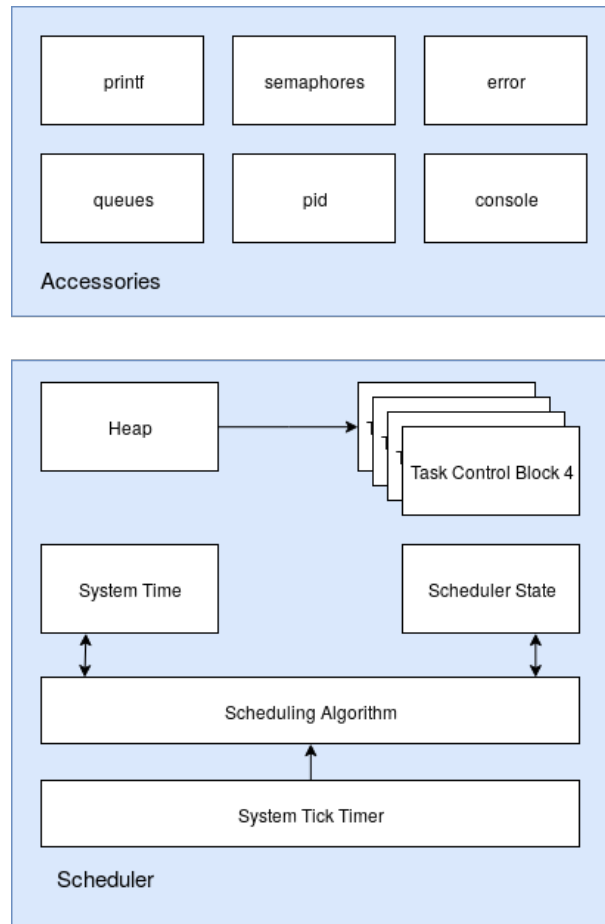


Figure 4.8: Structure of the OS

In the course of the project a small real-time OS has been developed. Fig. 4.8 shows how the OS is structured. The operating system kernel consists out of a pretty simple scheduler, which is hooked up to the system tick timer, that the *Cortex-M3* (in the case of the *STM32F103RB*) provides. Configuring the system tick period is possible. With every system tick the scheduling algorithm is invoked. A state machine is deciding how to handle the system tick. Moreover, the scheduling is based on a binary heap, that contains pointers to the Task Control Blocks (TCBs) of the tasks, which are supposed to be executed. A task control block is created, when a task is spawned by a programmer with. This works as follows:

```
1  /** Spawn a task.
2   *
3   * @param ifnc_ptr Pointer to the task function.
4   * @param itask_name Internal task name.
5   * @param iarguments Enables passing user-defined arguments to the task.
6   * @param ipriority A higher value means a higher priority of the task.
7   * @param oTaskHandle Pointer to TCB.
8   * @retval 1 (task has been spawned) or 0 (FAILED)
9   */
10 uint8_t osTaskCreate(void (*ifnc_ptr)(void*), char* itask_name, void* iarguments,
11                    uint8_t ipriority, const osTCB_t* oTaskHandle);
```

In general the OS interface is frankly speaking inspired by FreeRTOS™. One can also see, that the scheduling is priority based. The task with the highest priority is always executed as soon as the current task finished its job. Priorities can not be changed on run-time. Of course, as with any real-time OS



delay functions are available. `osTaskDelayUntil` is the more important one, as it allows periodic task execution.

```
1 void task3(void* ptr)
2 {
3     static uint32_t wakeup = 0;
4     char* args = (char*)ptr;
5
6     wakeup = osSchedulerGetSysT();
7     DEBUG_MSG("%s here!\n\r", args);
8     osTaskDelayUntil(wakeup, MS_2_TICKS(300));
9 }
```

As shown in the example above the TCB also holds a pointer to data, that can be passed to a task.

Besides creating a easily understandable real-time OS it was intended to provide some smooth infrastructure with it, or rather to equip it with some tools, so that one does not always have to reinvent the wheel, kicking off a new project. Therefore, the OS comes with a growing set of accessories as for instance queues (cf. accessories in fig. 4.8).

```
1 void USART2_IRQHandler(void)
2 {
3     /* USART2 receive buffer contains a char. */
4     if(USART_GetITStatus(USART2, USART_IT_RXNE) == SET)
5     {
6         uint8_t data;
7         data = USART_ReceiveData(USART2) & 0xFF;
8         if(!osEnqueue(&usart_rx_q, (void*)&data))
9             THROW_ERROR(E_USART_RX_BUFFER_OVERFLOW);
10    }
11
12    /* USART2 transmit buffer empty. */
13    if(USART_GetITStatus(USART2, USART_IT_TXE) == SET)
14    {
15        uint8_t data;
16        if(osDequeue(&usart_tx_q, &data))
17            USART_SendData(USART2, data);
18        else
19        {
20            /* Nothing to send. Disable interrupt. */
21            USART_ITConfig(USART2, USART_IT_TXE, DISABLE);
22            tx_overflow = 0;
23        }
24    }
25 }
```

The code example above shows how the error and the queue accessories can be used to implement the Universal Asynchronous/Synchronous Receiver/Transmitter (USART) Interrupt Request (IRQ) handler of the *STM32F103RB* elegantly. The USART status register has to be checked for finding out which event triggered the interrupt. Either the transmit buffer is empty again and needs to be fed or the receive buffer contains a char, which needs to be stored somewhere. Both buffers are only as big as 8 bit, so one needs queues, that can be emptied and filled by concurrent processes. The OS takes care of that. Also, in this example one can see how an error is passed to the according module in case, the receive buffer overflows. The idea is to have a proper log of what is going on in case the system fails (where did it go wrong and when). Also the sink for error logging can be changed (Electrically Erasable Programmable Read-Only Memory (EEPROM) / USART / screen).

Other accessories are

- `printf` - A light version of GNU's Not Unix! (GNU) `printf`.



- semaphores - Different kind of semaphores to get rid of data inconsistency problems that occur with concurrent processes e.g. tasks.
- pid - Infrastructure for PID controll.
- console - A console to command the system and get feedback, similar to a terminal.

The whole *Doxygen* documentation can be found on *Github*.

## 5 Evaluation

The whole project is a strict work in progress, which is not finished yet. Different components have a different maturity.

Tab. 5.2 reflects the status quo of the OS. It can certainly not keep up with modern OS. Nevertheless, it is easily comprehensible, because of its small footprint, and provides a bit another approach as a pure solution for real-time OS for embedded applications with useful accessories in this field.

Component	Working	Problem
Non-preemptive Scheduling	✓	Round-robin scheduling is not guaranteed among tasks having the same priority.
Preemptive Scheduling	✓	
System Tick Configuration	✓	
Error Logging	✓	Only supports one sink (USART) so far.
printf	✓	Only supports the options %d, %f, %c and %s currently.
Queues	✓	Only supports one kind of queue, which is useful for a producer-consumer scenarios.
Semaphores		
PID controller infrastructure		
Console		

Table 5.1: Status Quo OS

The UV light hardware has been simulated with the help of *LTSpice* quite extensively. After gaining support from Yi Wang and convincing the department to pay these companies for their products and services respectively, both PCBs introduced earlier have been manufactured in China by the company *Elecrow* (. Also, the components have been ordered in the USA from the distributor *Digikey* and are available. The boards have not yet been assembled and tested. The planned intend to get rid of mercury, mentioned in chapter 2, has been pursued consequently. It is an absolutely revolutionary feature, that this UV exposure unit does not use fluorescent lamps like the most common type the competitors offer on the market right now. A housing has been built from wood, which is cheaper and rather environmentally friendly. The current mechanics allow single-sided UV light exposure.

Component	Working	Problem
No mercury	✓	
Double-sided UV light exposure		
Precisely controlled light intensity	✓	
Digital and elegant user interface	✓	
Hardware design ready	✓	
Hardware assembled and tested		
No expensive metal housings	✓	
DAC driver		
OLED driver		
Rotary encoder driver		

Table 5.2: Status Quo UV light

The etching bath temperature control is in an earlier stage of development, where success was for a long time hindered by a driver issue with the Analog-to-Digital Converter (ADC) on a similar STM32-chip.



For development the STM32L073RZ was used, for the simple reason of immediate availability. However, the ADC on this specific chip has had problems reading the values properly until now. Without actual data from the temperature sensor the rest of the system could only be implemented but not tested. A simple on-off controller has been tested successfully, the implementation with proportional and inertia correction has not yet been tested successfully.

<b>Component</b>	<b>Working</b>	<b>Problem</b>
Digital and elegant user interface	✓	
Hardware design ready		
Hardware assembled and tested		Heater needs to be "hacked" for control
Additive manufactured housing	✓	PCB holder needs improvement
ADC driver for NTC	✓	
I2C driver	✓	
OLED driver	✓	
Rotary encoder driver	✓	Low-pass filter missing
On-off controller	✓	
On-off controller with corrections		

Table 5.3: Status Quo etching bath

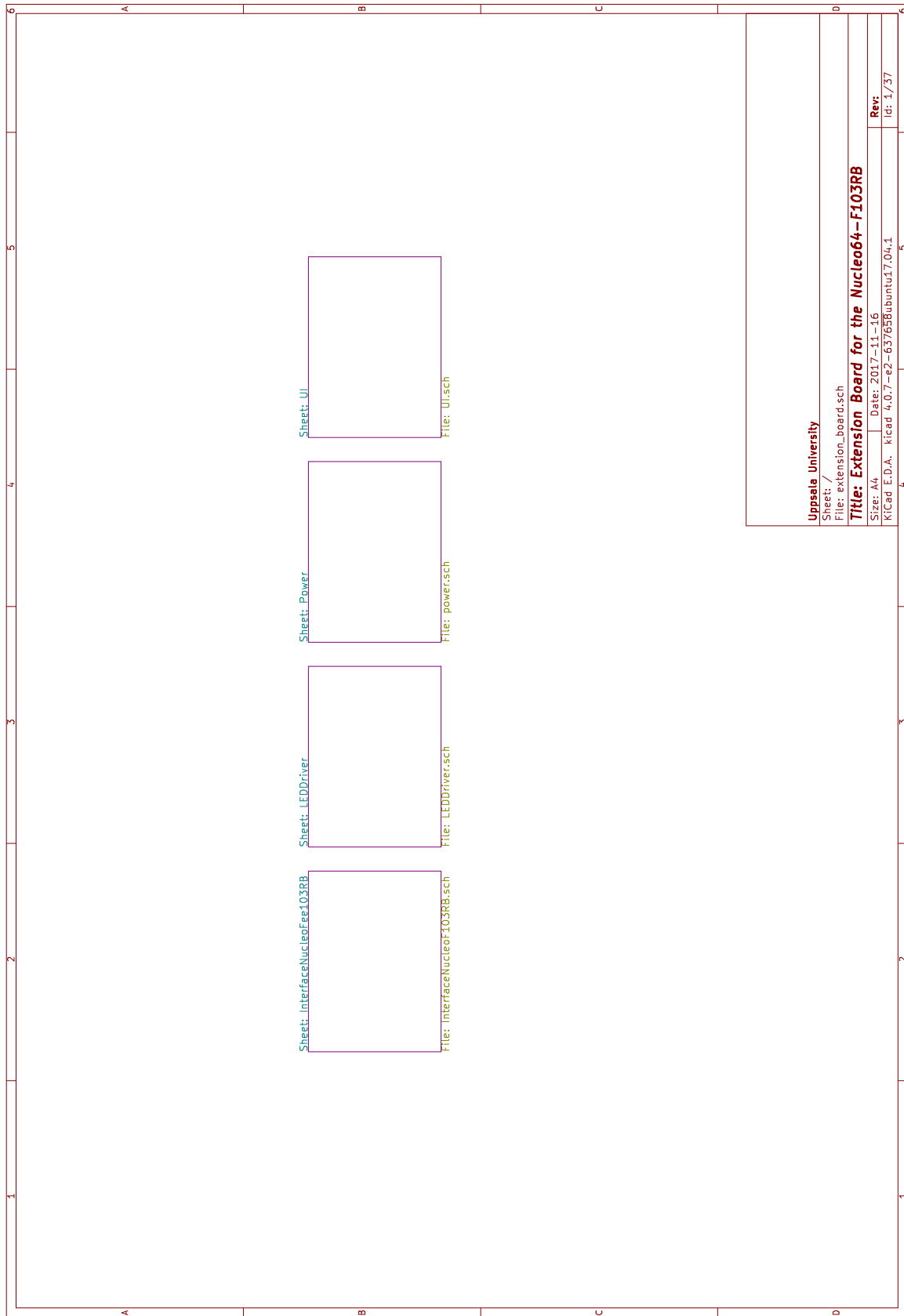
## 6 Conclusion and Future Work

The OS kernel works and the system is ready to be deployed. A couple of accessories as are not implemented at all yet. Others need to be upgraded. The kernel is also subject to improvements. It shall be possible for the programmer to chose whether the scheduling is preemptive or non-preemptive. Also, it shall be possible to assign a stack size to a task and make sure it is not overflowed.

The UV light development put forth a complex four layer extension board for the *Nucleo-F103RB* from *STM*, including a sophisticated LED driver using a linear regulator approach. The circuits on the board have been laid out, simulated and in a next step they will be tested. Moreover, a mechanical design has been sketched and built up. Unfortunately, it only supports single-sided UV light exposure and causes a lot of manual work building it, which is something to be solved in the near future. One idea is to use some Fused Deposition Modeling (FDM) parts combined with two PCBs, 128 LEDs each, to make double-sided exposure and an easy assembly possible. Most of the driver software implementation still has to be done and the whole system has to be integrated.

The temperature control development for the etching bath is ready for testing as soon as the temperature sensor is read correctly from the ADC. These tests will show if the implemented improved on-off control will deliver sufficient results or if the heater has to be controlled with a constant current controller. When that is decided further steps regarding hardware design can be made. Also, the current implementation has to be ported to the OS, which should not entail much work, as it is structured in a fitting way already.

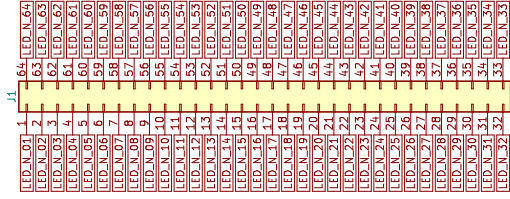
# A Schematics Extension Board



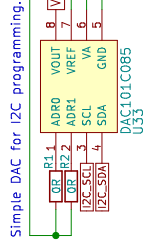
32 x 2 = 64 stages. 1 stage consists out of 4 LEDs.

- Sheet: 2Stages1  
LED\_N.01 | LED\_N.01
- LED\_N.02 | LED\_N.02
- File: 2Stages1.sch
- Sheet: 2Stages2  
LED\_N.03 | LED\_N.03
- LED\_N.04 | LED\_N.04
- File: 2Stages1.sch
- Sheet: 2Stages3  
LED\_N.05 | LED\_N.05
- LED\_N.06 | LED\_N.06
- File: 2Stages1.sch
- Sheet: 2Stages4  
LED\_N.07 | LED\_N.07
- LED\_N.08 | LED\_N.08
- File: 2Stages1.sch
- Sheet: 2Stages5  
LED\_N.09 | LED\_N.09
- LED\_N.10 | LED\_N.10
- File: 2Stages1.sch
- Sheet: 2Stages6  
LED\_N.11 | LED\_N.11
- LED\_N.12 | LED\_N.12
- File: 2Stages1.sch
- Sheet: 2Stages7  
LED\_N.13 | LED\_N.13
- LED\_N.14 | LED\_N.14
- File: 2Stages1.sch
- Sheet: 2Stages8  
LED\_N.15 | LED\_N.15
- LED\_N.16 | LED\_N.16
- File: 2Stages1.sch
- Sheet: 2Stages9  
LED\_N.17 | LED\_N.17
- LED\_N.18 | LED\_N.18
- File: 2Stages1.sch
- Sheet: 2Stages10  
LED\_N.19 | LED\_N.19
- LED\_N.20 | LED\_N.20
- File: 2Stages1.sch
- Sheet: 2Stages11  
LED\_N.21 | LED\_N.21
- LED\_N.22 | LED\_N.22
- File: 2Stages1.sch
- Sheet: 2Stages12  
LED\_N.23 | LED\_N.23
- LED\_N.24 | LED\_N.24
- File: 2Stages1.sch
- Sheet: 2Stages13  
LED\_N.25 | LED\_N.25
- LED\_N.26 | LED\_N.26
- File: 2Stages1.sch
- Sheet: 2Stages14  
LED\_N.27 | LED\_N.27
- LED\_N.28 | LED\_N.28
- File: 2Stages1.sch
- Sheet: 2Stages15  
LED\_N.29 | LED\_N.29
- LED\_N.30 | LED\_N.30
- File: 2Stages1.sch
- Sheet: 2Stages16  
LED\_N.31 | LED\_N.31
- LED\_N.32 | LED\_N.32
- File: 2Stages1.sch

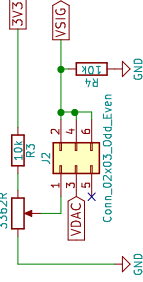
- Sheet: 2Stages32  
LED\_N.63 | LED\_N.63
- LED\_N.64 | LED\_N.64
- File: 2Stages1.sch
- Sheet: 2Stages33  
LED\_N.65 | LED\_N.65
- LED\_N.66 | LED\_N.66
- File: 2Stages1.sch
- Sheet: 2Stages34  
LED\_N.67 | LED\_N.67
- LED\_N.68 | LED\_N.68
- File: 2Stages1.sch
- Sheet: 2Stages35  
LED\_N.69 | LED\_N.69
- LED\_N.70 | LED\_N.70
- File: 2Stages1.sch
- Sheet: 2Stages36  
LED\_N.71 | LED\_N.71
- LED\_N.72 | LED\_N.72
- File: 2Stages1.sch
- Sheet: 2Stages37  
LED\_N.73 | LED\_N.73
- LED\_N.74 | LED\_N.74
- File: 2Stages1.sch
- Sheet: 2Stages38  
LED\_N.75 | LED\_N.75
- LED\_N.76 | LED\_N.76
- File: 2Stages1.sch
- Sheet: 2Stages39  
LED\_N.77 | LED\_N.77
- LED\_N.78 | LED\_N.78
- File: 2Stages1.sch
- Sheet: 2Stages40  
LED\_N.79 | LED\_N.79
- LED\_N.80 | LED\_N.80
- File: 2Stages1.sch
- Sheet: 2Stages41  
LED\_N.81 | LED\_N.81
- LED\_N.82 | LED\_N.82
- File: 2Stages1.sch
- Sheet: 2Stages42  
LED\_N.83 | LED\_N.83
- LED\_N.84 | LED\_N.84
- File: 2Stages1.sch
- Sheet: 2Stages43  
LED\_N.85 | LED\_N.85
- LED\_N.86 | LED\_N.86
- File: 2Stages1.sch
- Sheet: 2Stages44  
LED\_N.87 | LED\_N.87
- LED\_N.88 | LED\_N.88
- File: 2Stages1.sch
- Sheet: 2Stages45  
LED\_N.89 | LED\_N.89
- LED\_N.90 | LED\_N.90
- File: 2Stages1.sch
- Sheet: 2Stages46  
LED\_N.91 | LED\_N.91
- LED\_N.92 | LED\_N.92
- File: 2Stages1.sch
- Sheet: 2Stages47  
LED\_N.93 | LED\_N.93
- LED\_N.94 | LED\_N.94
- File: 2Stages1.sch
- Sheet: 2Stages48  
LED\_N.95 | LED\_N.95
- LED\_N.96 | LED\_N.96
- File: 2Stages1.sch
- Sheet: 2Stages49  
LED\_N.97 | LED\_N.97
- LED\_N.98 | LED\_N.98
- File: 2Stages1.sch
- Sheet: 2Stages50  
LED\_N.99 | LED\_N.99
- LED\_N.100 | LED\_N.100
- File: 2Stages1.sch



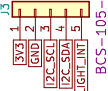
This connector connects the driver to the LEDs.



It shall be possible to switch between a (screw driver) poti and the DAC.

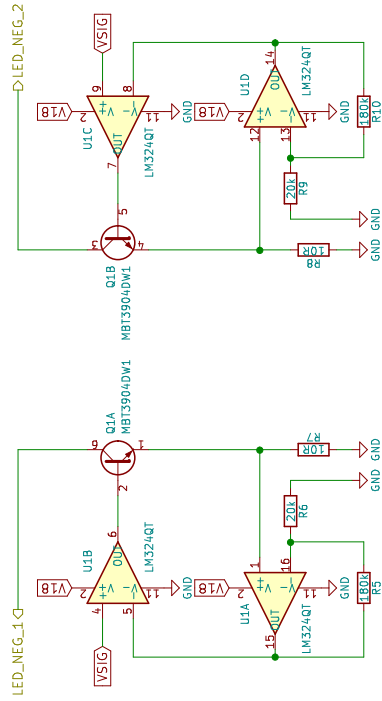


Light sensor TSL2561 for feedback.





Place decoupling capacitors close to IC.



Shunt Resistor Dissipation

$$P = U^2/R = (0.33 \text{ V})^2 / 10 \text{ } \Omega = 108.9 \text{ mW}$$

No problem!

Higher resistor means more stability (less prone to noise).

-> 696.96 W overall dissipation.

Uppsala University  
 Sheet: /LEDDriver/25stages1/  
 File: 25stages1.sch

**Title: UV LED Driver Two Stages**

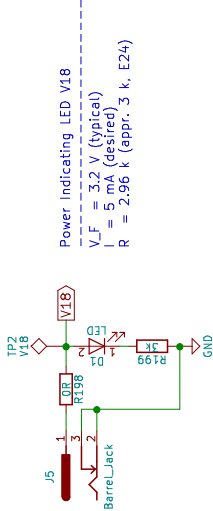
Size: A4  
 Date: 2017-11-13  
 KitCad E.D.A. kicad 4.0.7-e2-637658ubuntu17.04.1

Rev:  
 id: 3/37

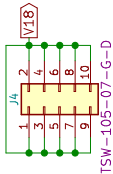
Current Rating (IPC 2221)

Max. Expected Current = 64 \* 0.05 A = 3.2 A  
 Abs. Max. Current (dT = 10 K, W = 1.6 mm) = 3.36 A

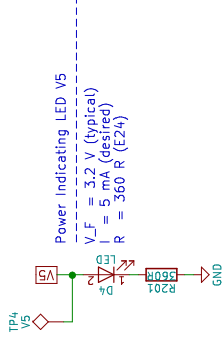
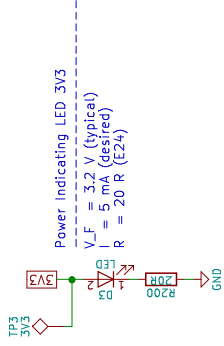
1206 resistor footprint for being able to solder in a switch (housing) in a later revision.



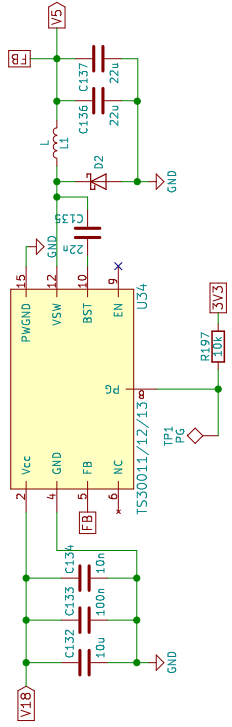
Power connector for extension board (LEDs).



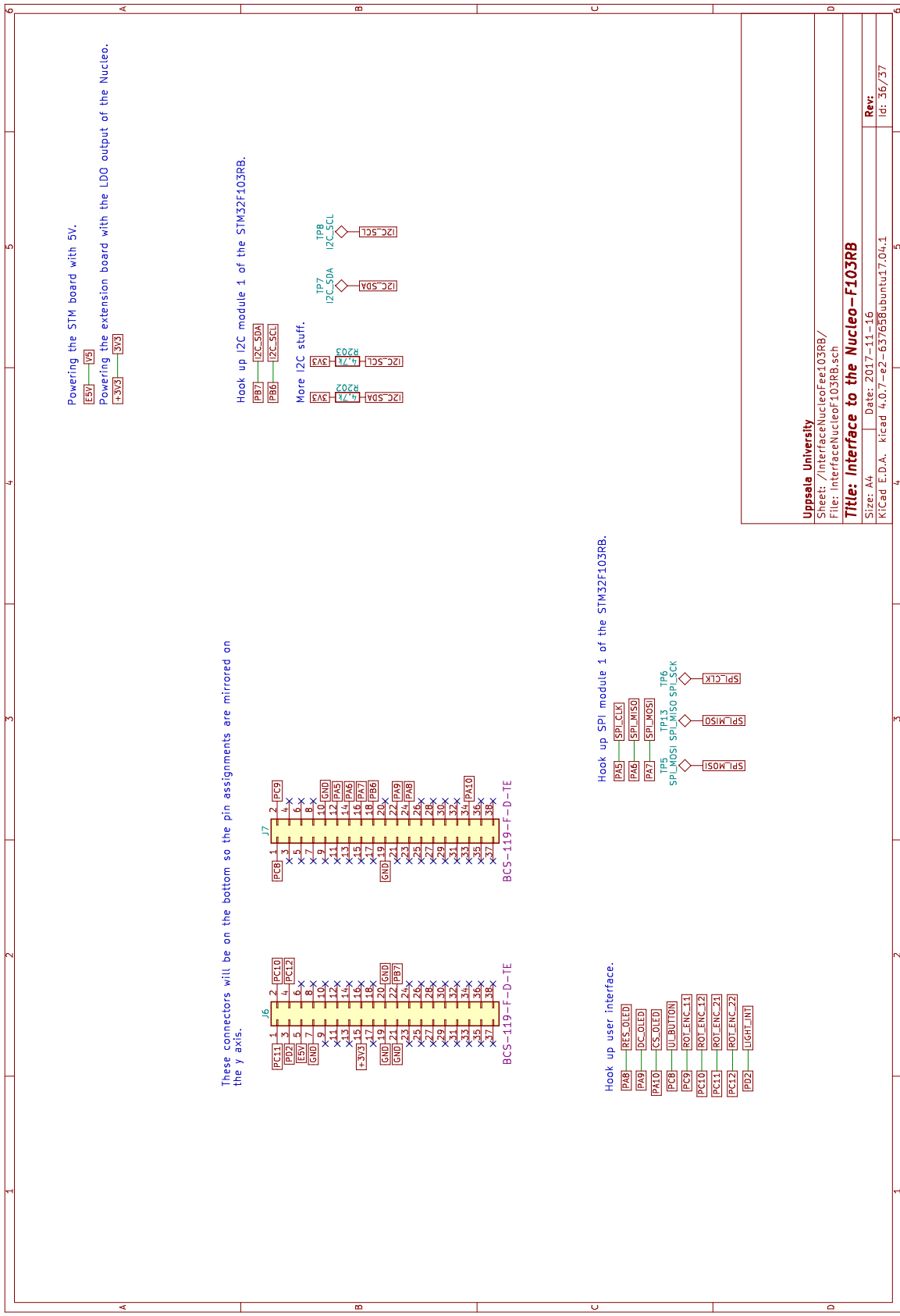
Tell Kicad, that these voltages are externally driven.



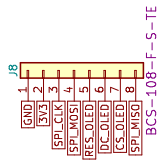
With this step-down converter the STM board is powered.



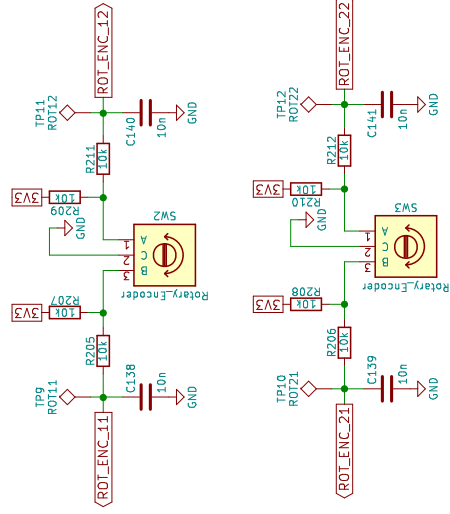
A low ESR is required for the output capacitors.



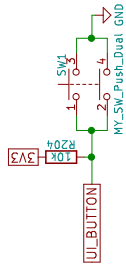
SSD1306 OLED SPI (4 wire) interface.



Rotary encoders with external debouncing mechanism.  
 Debouncing circuit stolen from:  
<https://hifiduino.wordpress.com/2010/10/20/rotaryencoder-hw-sw-no-debounce/>



Tactile switch.



Uppsala University

Sheet: /UI/

File: UI.sch

Title: User Interface

Size: A4 Date: 2017-11-16

KiCad E.D.A. kicad 4.0.7-e2-637658ubuntu17.04.1

Rev:

id: 37/37

# B Technical drawings of PCB-holder

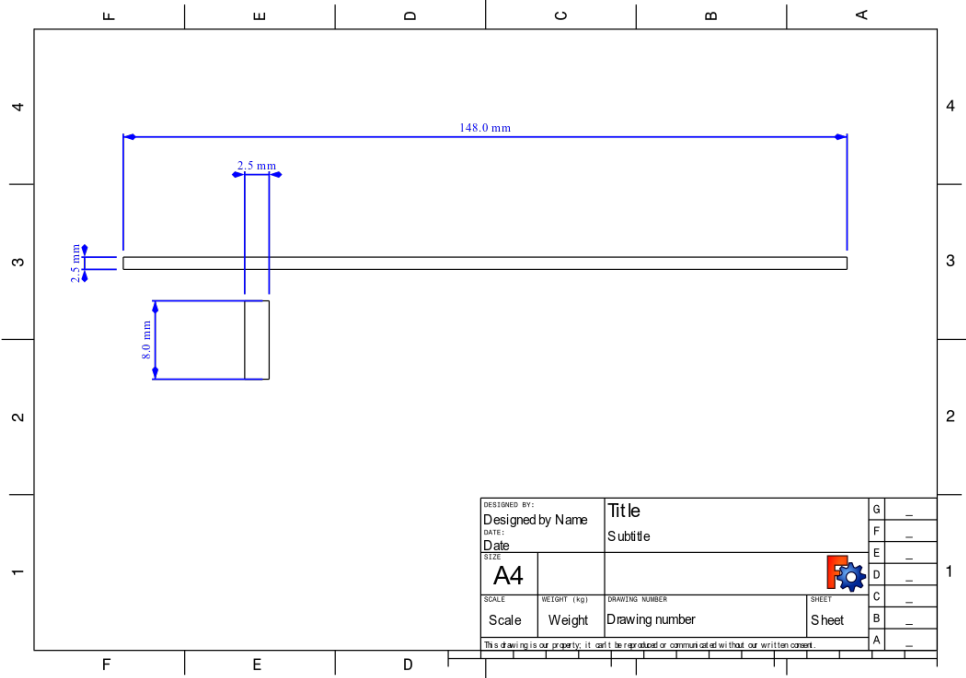


Figure B.1: Technical Drawing: Crossbar

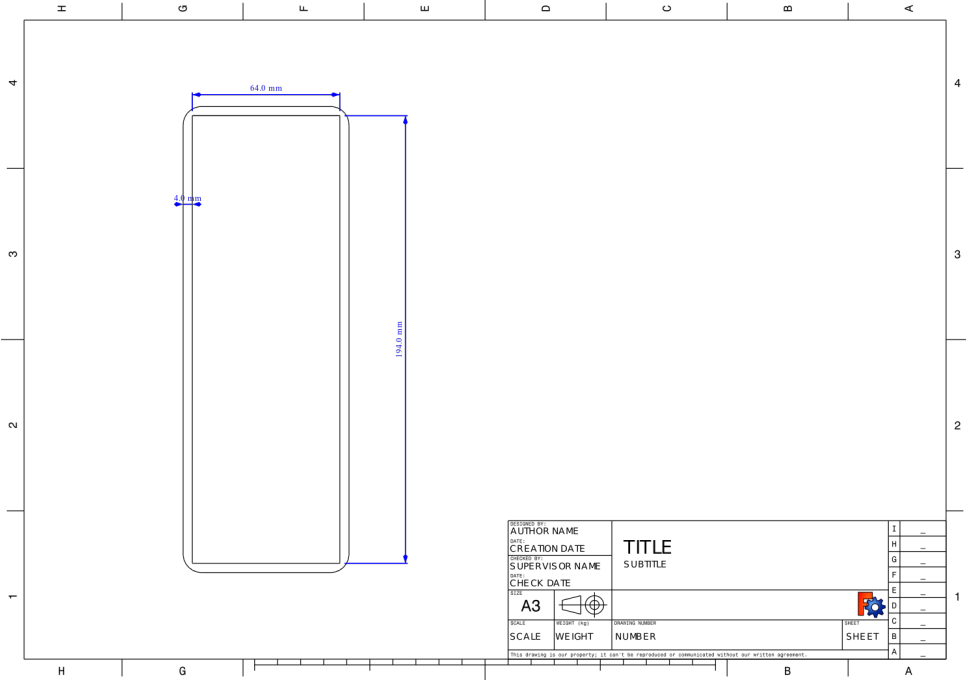


Figure B.2: Technical Drawing: Floor

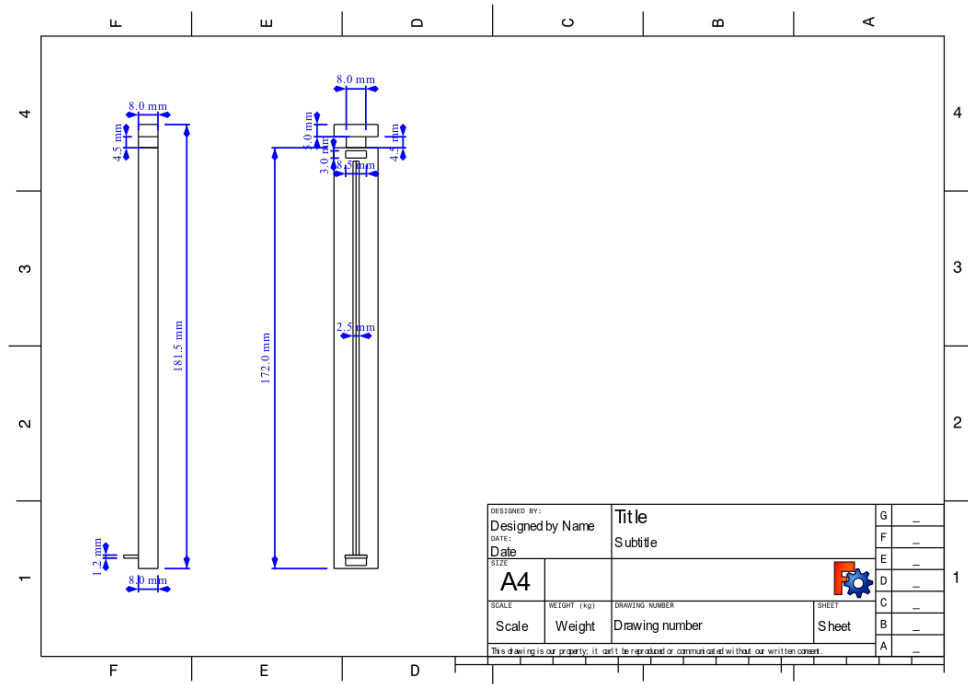


Figure B.3: Technical Drawing: Holder

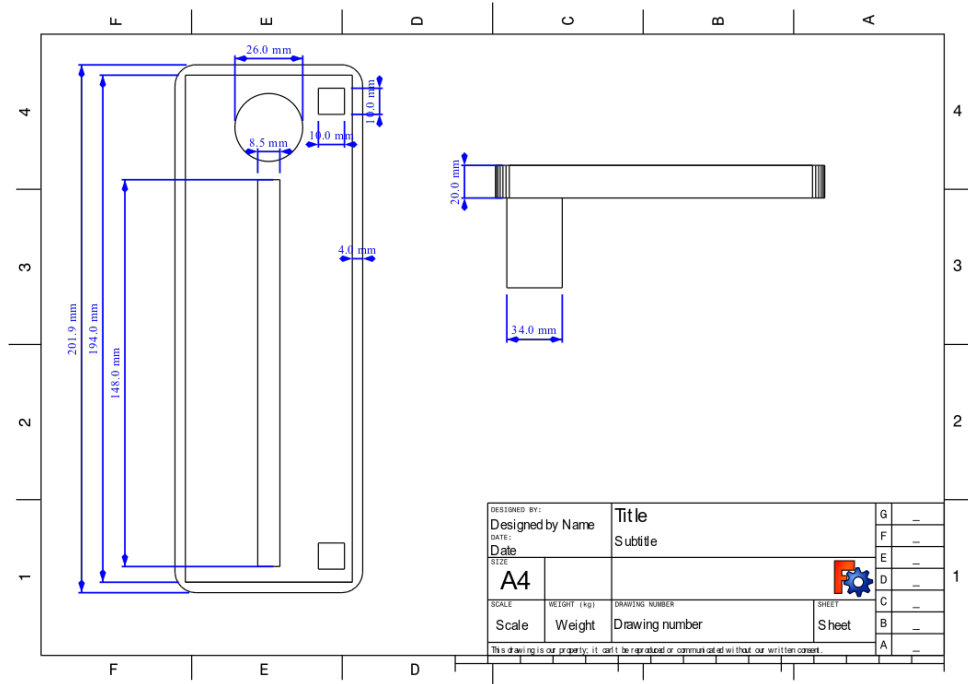


Figure B.4: Technical Drawing: Lid