Höhere Technische Bundeslehr- und Versuchsanstalt Rankweil

Höhere Lehranstalt für Elektronik
Ausbildungsschwerpunkt Technische Informatik
Schuljahr 2013/14

Diplomarbeit

Intermittent Laser Projection System

Projektteam:
David Beinder
Wolfgang Dür
Daniel Mathis
Matthias Übelher

Betreuer:
Prof. Dipl.-Ing. Franz Lauritsch

Rankweil, am 28.05.2014

1 Eidesstattliche Erklärung

Wir versichern an Eides statt, dass wir alle entsprechend gekennzeichneten Teile der vorliegenden Diplomarbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit gekennzeichnet haben. Ferner gestatten wir der Höheren Technischen Lehr- und Versuchsanstalt Rankweil (HTLBuVA Rankweil), die vorliegende Diplomarbeit unter Beachtung insbesondere datenschutzrechtlicher und wettbewerbsrechtlicher Vorschriften für Lehre und Forschung zu benutzen.

Declaration of Oath

We hereby declare by oath that all appropriately labelled parts of our thesis were independently written by ourselves, no other than the indicated sources and tools have been used and that all parts of the thesis which have been borrowed, either literally or in a analogous manner, have been properly cited. Furthermore, we permit the Higher Technical College and Laboratory of Rankweil (Höhere Technische Bundeslehr- und Versuchsanstalt Rankweil) to use the thesis for teaching and research, with due regard to matters of privacy and antitrust regulations.

Rankweil, am _____

David Beinder	Matthias Übelher
Wolfgang Dür	Daniel Mathis

2 Abstract

Deutsch

Das Ziel ist es einen Laserprojektor zu bauen, welcher einen Laserstrahl moduliert und mittels Spiegelablenkung eine Vektorgrafik auf eine Fläche projiziert. Dies muss so schnell erfolgen, dass das menschliche Auge ein stehendes Bild wahrnimmt. Die Spiegel, welche den Laserstrahl ablenken, sollen mit selbstentwickelten Galvanometern gesteuert werden. Um diese in Gegenwart des Erdmagnetfelds und anderen Einflüssen in Position zu halten, wird eine aktive Regelung verwendet.

Eine graphische Oberfläche dient zum Erstellen der Grafiken und zur Steuerung des Geräts. Die Oberfläche ist webbasiert, was den Zugriff mit jedem WLAN-fähigen Gerät ermöglicht. Die Vektordaten werden von der Software so optimiert, dass der Laser möglichst wenig Weg zurücklegen muss. Der dafür notwendige Webserver wird auf einem Microcomputer unter Embedded Linux betrieben.

English

The goal of our project is to build a laser projector, which uses a modulated laser beam and deflection mirrors to draw a vector image onto a surface. The laser dot has to move fast enough in order to fool the human eye into seeing a static image. The mirrors used to deflect the laser beam are mounted on custom-built galvanometers. To keep these in position in the presence of the earth's magnetic field and other influences, an active control loop is used.

The users can control the device and draw vector images using a web-based graphical user interface. This allows every WiFi capable device to access the gadget. The vector image data is optimized by the software to keep the path driven by the laser dot to a minimum. All this runs on a microcomputer running a flavor of embedded linux.

3 Kurzfassung

Deutsch

Das Ziel ist es einen Laserprojektor zu bauen, welcher mit einem modulierten Laserstrahl eine Vektorgrafik auf eine Fläche projiziert. Um mit einem kleinen Punkt ein Bild an die Wand zu zeichnen, muss das Auge überlistet werden. Dazu wird der Laserpunkt so schnell bewegt, dass das Auge den bewegten Punkt nicht wahrnehmen kann und eine stehende Linie sieht.

Da der Laser nicht mit solch einer hohen Geschwindigkeit bewegt werden kann, muss der Laserstrahl selbst abgelenkt werden. Dies wird durch zwei Spiegel auf Galvanometern realisiert. Ein Galvanometer besteht aus einem Magneten auf einer gelagerten Achse auf welcher sich auch ein Spiegel befindet. Die Position des Spiegels wird durch einen kapazitiven Sensor erfasst und mittels eines Elektromagneten und eines Regelkreises kontinuierlich korrigiert.

Die Galvanometer werden durch einen STM32 gesteuert, ein 32Bit Microcontroller der Firma ST-Microelektronik. Der STM32 verfügt über zwei 12-Bit DACs über welche die Regelkreise der Galvanometer angesteuert werden. Die Firmware wird mit der Programmiersprache C in der Programmierumgebung TrueStudio und Eclipse programmiert.

Ein eigens konstruiertes Power-Modul liefert die von uns benötigten Spannungen 3.3V, 3.8V und 5V. Die Spannungen werden von zwei gekauften Netzteilen mit $\pm$12V gespeist. Diese versorgen auch die beiden Galvanometer.

Mithilfe einer graphischen Oberfläche können Benutzer Vektorgrafiken erstellen, die dann an den Server gesendet werden. Als Server dient ein Raspberry Pi, welches das Herzstück dieses Projektes ist und die Kommunikation zwischen den einzelnen Clients und dem Server steuert. Die Software wird vom Raspberry Pi gehostet und über ein WiFi Netzwerk für Tablets und Smartphones verfügbar gemacht. Sobald sich ein Endgerät mit dem Netzwerk verbunden hat, kann die Software per Webbrowser geöffnet werden. Erstellte Vektorgrafiken werden vom Client direkt zum Server gesendet, anschließend verbreitet der Server die Daten an andere verfügbare Clients und optimiert sie so, dass der Laserpunkt möglichst wenig Weg zurücklegen muss. Damit der Laserpunkt die Vektorgrafik darstellen kann, müssen die Koordinaten der Punkte der Zeichnung über eine Formel in einen fixen Wertebereich für die DACs umgewandelt werden.

Bei der Software für die Webanwendung wird die neuartige Programmiersprache NodeJS verwendet. NodeJS besitzt eine asynchrone Architektur. Diese basiert auf der neuen V8 JavaScript Engine und ist dadurch sehr ressourcensparend. Durch die asynchronen Fähigkeiten von NodeJS arbeitet das Programm weiter wenn beispielsweise ein Datenbankzugriff gemacht wird. Da das Programm nicht wartet bis ein solcher Zugriff fertig ist, ist es möglich eine viel schnellere Umgebung zu erschaffen. Das asynchrone Verhalten der Programmiersprache erhöht zwar die Geschwindigkeit, allerdings erhöht sich damit auch die Komplexität der Software.

Da NodeJS auf JavaScript aufbaut, kann sowohl im Frontend als auch im Backend mit JavaScript gearbeitet werden. Zusätzlich ist es mit NodeJS möglich, eine Verbindung zu einem Datenbanksystem aufzubauen. Das verwendete Datenbanksystem heißt MongoDB. MongoDB ist derzeit die verbreitetste NoSQL Datenbank, das heißt sie baut nicht wie übliche SQL Datenbanken auf den relationalen Normen und Tabellen auf, sondern erstellt einfache Tabellen die nicht unbedingt miteinander verknüpft sind. Durch die unabhängigen Tabellen können sowohl große als auch kleine Datenmengen schneller aus der Datenbank geladen bzw. bearbeitet werden, als mit einer üblichen SQL Datenbank.

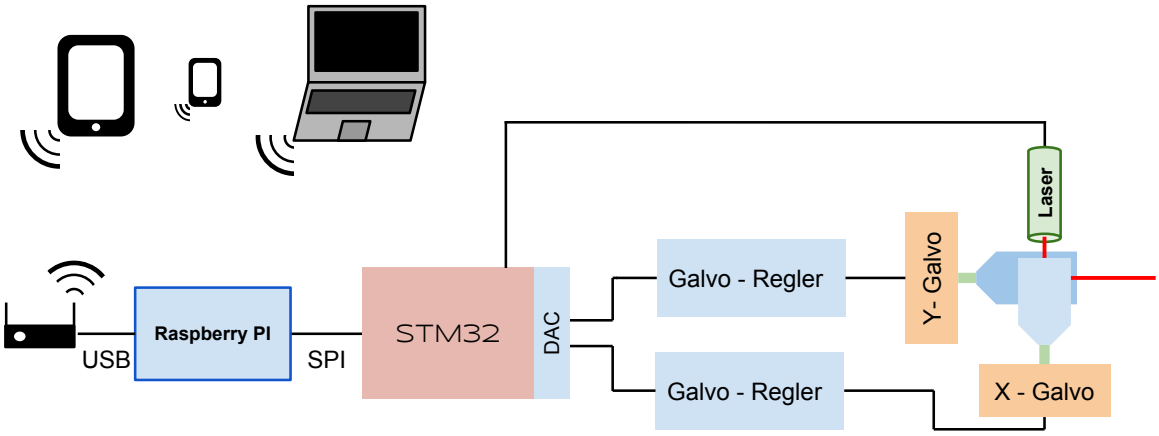


Abbildung 1: Blockschaltbild

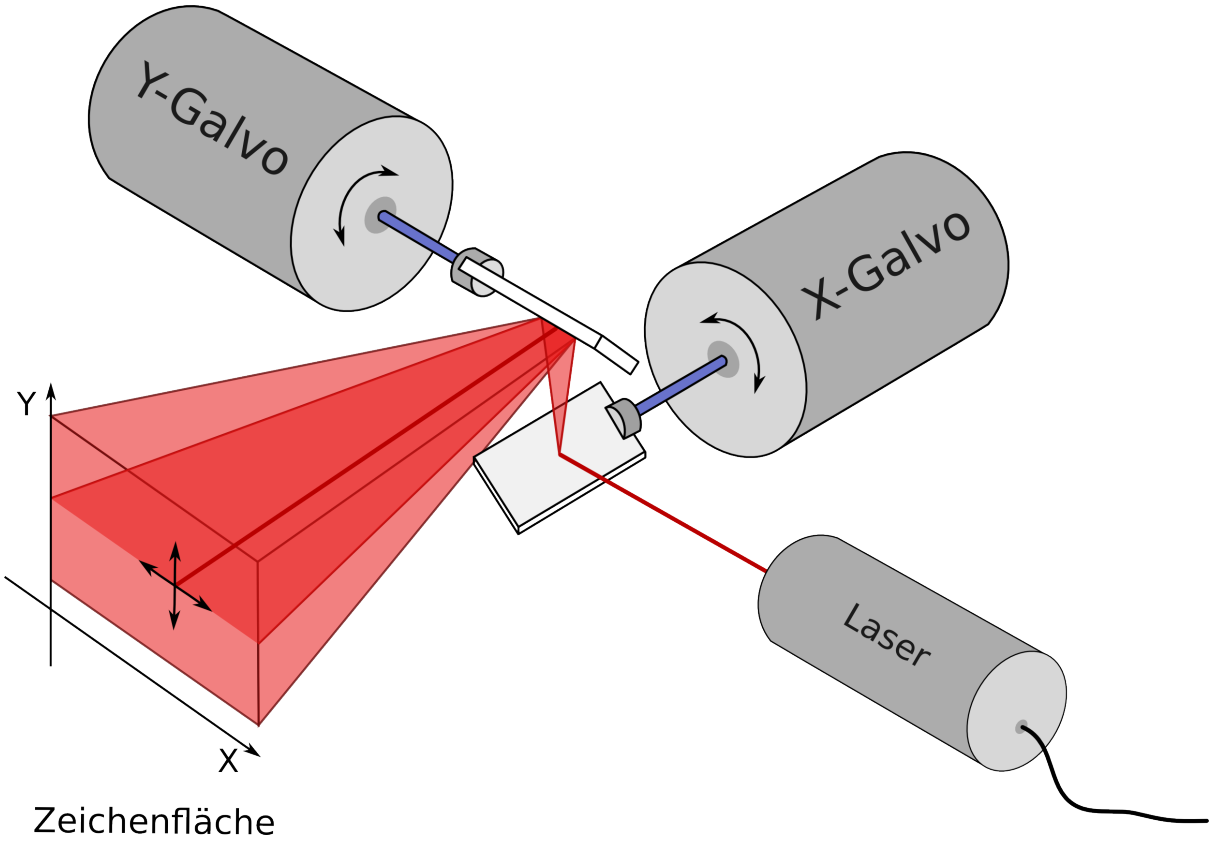


Abbildung 2: Spiegelablenkung

English

The goal of our project is to build a laser projector which uses a modulated laser beam to draw a vector image onto a surface. The laser dot has to move fast enough in order to fool the human eye into seeing a standing line on the projection surface. As the laser cannot be moved this quickly, the laser beam itself has to be deflected. This is accomplished by two deflection mirrors mounted on custom-built galvanometers. A galvanometer consists of a magnet mounted on a axis on bearings with a mirror on one end. The position of the mirror is detected by a capacitive sensor and using a electromagnet and a control loop, the angle of the mirror is constantly adjusted.

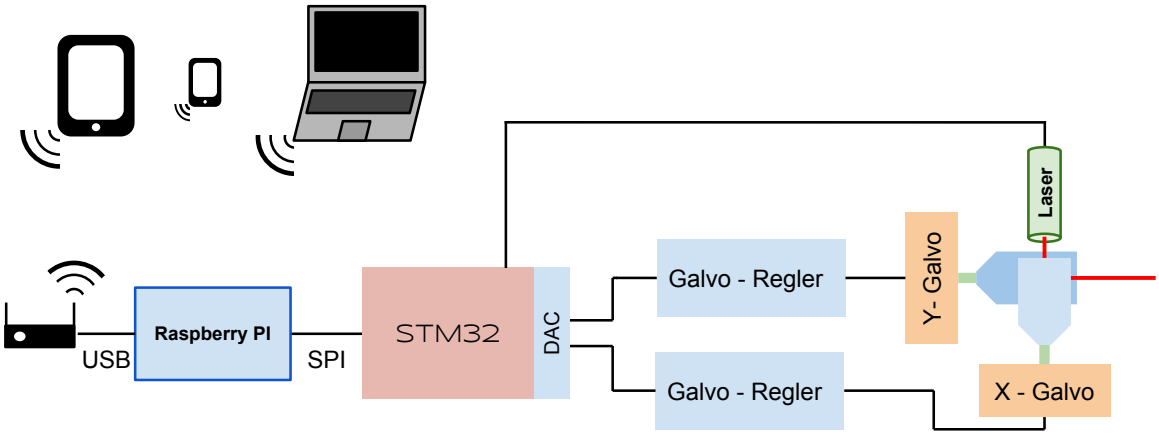
The galvanometers are controlled by a STM32, a 32bit microcontroller of STMicroelectronics. The controller has two integrated 12bit DACs, which provide the input for the control loops of the galvanometers. The firmware for the microcontroller is written in C using both the Atollic TrueStudio IDE and Eclipse.

A custom power module supplies power rails with 3.3V, 3.8V and 5V to the rest of the device. This module is fed by a pair of 12V switched-mode power supplies, whose rails are also used to supply ±12V to the two galvanometers.

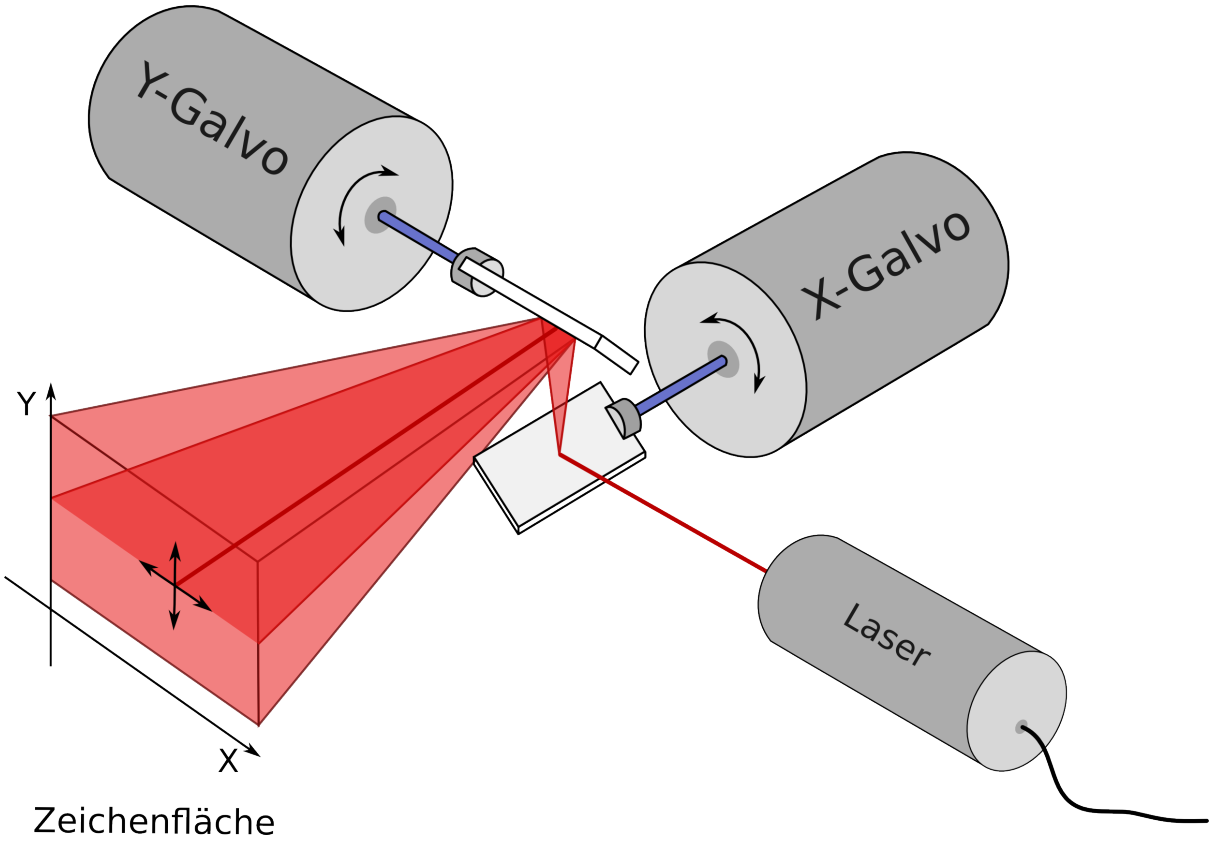
The users can draw vector graphics using a graphical user interface. These pictures are then sent to the server, a Raspberry Pi. It can be called the heart of the device, as it manages the communication between the clients and the server and processes all data. The web interface is hosted on the Raspberry Pi and made public for smartphones and tablets using a integrated WiFi access point. As soon as a client connects to the network, the software can be accessed using a web browser. Everything drawn is sent to the server, which distributes the current picture to all connected clients. It also optimized the path of the laser for minimal distance between the lines. To project the image, the server calculates the coordinates for each point to be sent to the DACs.

For the web interface, the new NodeJS framwork is used. It utilizes a asynchronous architecture and is based on the new V8 JavaScript engine. Because of the asynchronous architecture, the program never stalls when it has to wait for a ressource like a database query. In such a case, the program can fullfill other requests while it waits. This behaviour can greatly speed up the program and enhance the user experience but this comes at the cost of additional complexity.

Because NodeJS is based on JavaScript, both the frontend as well as the backend could be written in the same programming language. Additionally it is possible to synchronize all images with a database system for storage. For this project, the most widely used NoSQL database system was used, MongoDB. A NoSQL database does not use the conventional abstraction with fixed colums and normalised structures, but stores its data in simple tables which are not necessarily connected. Since the tables are smaller and independent, the data can be accessed faster than with a common SQL system.



System Overview



Laser Deflection

Inhaltsverzeichnis

4 Pflichtenheft

4.1 Einleitung

4.1.1 Allgemein

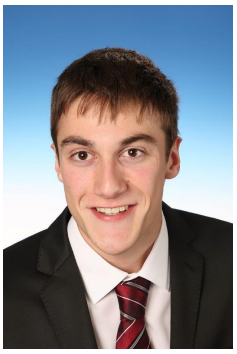
Das Projekt ILPS dient zur Darstellung von Vektorgrafiken mittels Laserstrahl und einer Spiegelablenkung. Um ein stehendes Bild zu erhalten, wird die Trägheit des menschlichen Auges ausgenutzt. Somit kann mit einem schnellbewegenden Punkt eine Illusion eines Bildes erzeugt werden. Damit die gezeichneten Linien deckungsgleich sind wird ein sowohl hochgenauer als auch schneller Ablenkungsmechanismus benötigt. Wir verwenden dafür aktiv geregelte Galvanometer. Die gesamte Steuerung des Gerätes erfolgt über Touchscreen.

4.1.2 Team & Betreuung

4.1.2.1 Projektteam

Das Projektteam besteht aus folgenden vier Mitgliedern:

David Beinder

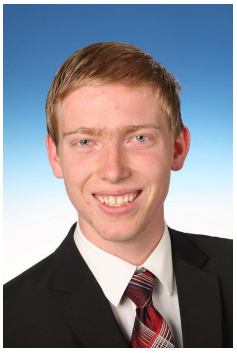


Hardware

Matthias Übelher

Hardware

Wolfgang Dür



Software

Daniel Mathis

Software

4.1.2.2 Betreuung

Dipl. Ing. Franz Lauritsch

Projektbetreuer

Dipl. Ing. (FH) Günther Jehle

Hilfestellung bezüglich Laser

Firma HQLaser

Sponsoring Laser

4.1.3 Zweck & Einsatzgebiet

Professionelle Geräte dieser Art werden in Lasershows zur Projektion von diversen Grafiken und Effekten eingesetzt. Unser Gerät soll bei kleinen Shows und Vorführungen zum Einsatz gelangen. Das Einsatzgebiet beschränkt sich auf abgedunkelte Räume, da mit den an der HTL zur Verfügung stehenden Mitteln und Sicherheitsvorkehrungen nur Laser mit Klasse 2 verwendet werden dürfen.

4.2 Spezifikation

4.2.1 Funktionsweise

Die zentrale Schaltstelle unseres Geräts ist ein Raspberry Pi, ein günstiger Microcomputer auf ARM Basis. Dieser stellt über einen Webservice die Benutzeroberfläche zur Verfügung, verarbeitet die Vektordaten, und stellt die Verbindung zur restlichen Hardware her. Die Web-Oberfläche kann von allen per WLAN verbundenen Geräten verwendet werden, in der Frontplatte ist zusätzlich bereits ein Android-Tablet fix integriert und immer verbunden. Die Ansteuerung der Hardware wird von einem STM32 Microcontroller übernommen. Dieser ist der einzige Teil des Geräts welcher immer aktiv ist - alle anderen Komponenten können von ihm bei Bedarf oder Störfall aktiviert und deaktiviert werden. Auf der Frontplatte ist eine USB-Verbindung zum Raspberry Pi und die Speicherkarte herausgeführt. Auch eine Audiobuchse ist verfügbar um allfällige Videobilder synchron mit Ton zu unterlegen.

4.2.2 Blockschaltbild

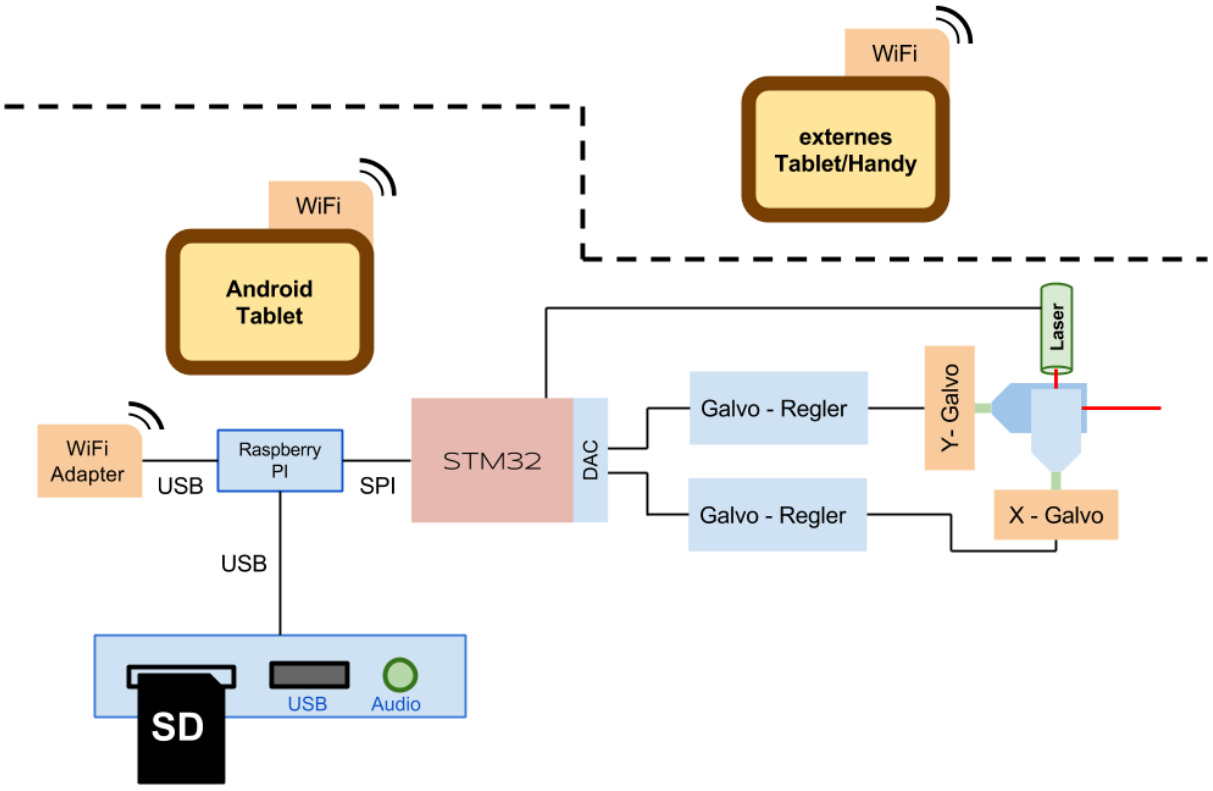


Abbildung 3: Blockschaltbild

4.2.3 Beschreibung der Komponenten

4.2.3.1 Stromversorgung

Die Stromversorgung wird mit Step-Down Reglern realisiert. Die galvanische Trennung wird durch ein gekauftes Netzteil, welches eine symmetrische Versorgung mit ±12V zur Verfügung stellt. Die drei Hilfsspannungen 5V/3.8V/3.3V werden mit einem Schaltregler energieeffizient erzeugt. Da einige Komponenten wie z.B. der Raspberry Pi ihre Stromversorgung nicht eigenständig trennen können, werden alle Spannungen mittels MOSFETs vom Hauptcontroller einzeln geschaltet werden.

Die folgenden Spannungen werden für die einzelnen Komponenten benötigt:

Spannung	Gerät
±12V	Galvanometer
5V	Raspberry Pi USB-Hub
3.8V	Tablet
3.3V	Hauptboard Lasermodul

4.2.3.2 Eingabe - Tablet

Als Haupteingabegerät dient ein preiswertes Android Tablet, welches direkt in das Gehäuse integriert ist. Der Vorteil an einem Tablet ist, dass wir ein relativ großes Touchscreen Display zur Verfügung haben, welches einzeln kaum zu diesem Preis erhältlich wäre. Gleichzeitig verbindet sich das Tablet über WLAN in das lokale Netz des Raspberry Pi. Das Tablet kann somit via Browser auf eine Oberfläche, welche vom Raspberry Pi zur Verfügung gestellt wird, zugreifen und dementsprechend Zeichnungen erstellen, laden und speichern sowie Einstellungen für den Laser vornehmen.

4.2.3.3 Datenverarbeitung - Raspberry Pi

Das Raspberry Pi wurde ursprünglich entwickelt um der sinkenden Anzahl an Informatikstudenten in Oxford entgegenzuwirken. Ziel war es, einen einfachen und günstigen Computer zu bauen, mit welchem Jugendliche experimentieren und das Programmieren erlernen können. Wir verwenden das Raspberry Pi zur Aufbereitung und Abspeicherung der Daten, die anschließend über die SPI Schnittstelle an den STM32 gesendet werden. Die Daten werden in einer dokumentenorientierten Open-Source-Datenbank namens MongoDB abgespeichert.

4.2.3.4 NodeJS

Node.JS ist eine Programmiersprache, welche verwendet wird um eine Webserverapplikation zu schreiben. Diese Applikation ermöglicht den Austausch von Koordinaten, sodass jedes Tablet oder Smartphone automatisch den aktuellen Stand der Zeichnung übermittelt bekommt. Zeitgleich werden diese Daten auch über die SPI Schnittstelle vom Raspberry Pi zu einem DAC Wandler gesendet, welcher anschließend die Spiegel ausrichtet. So kann das gezeichnete Bild in Echtzeit an der Projektionsfläche betrachtet werden.

4.2.3.5 Hauptcontroller

Der Controller ist primär für die Digital-Analog Wandlung verantwortlich. Zusätzlich übernimmt er die Temperaturüberwachung und die Kontrolle der Stromversorgung. Auch die Tasten des Tablets werden vom Controller angesteuert, beispielsweise um das Tablet ein- und auszuschalten.

4.2.3.6 Galvanometer

Um den Laserstrahl gleichzeitig schnell und präzise ablenken zu können, ist eine sehr leichte Mechanik vonnöten. Bei unserem Galvanometer liegt ein diametral magnetisierter Anker frei in einer Luftspule, mit welcher dieser zu einer Drehung angeregt werden kann.

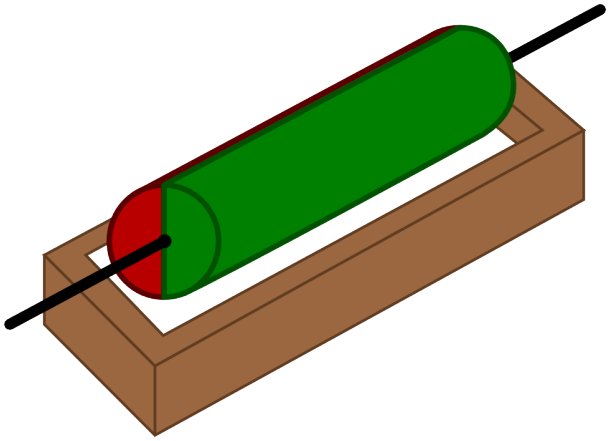


Abbildung 4: magnetisierter Anker

Um Gewicht zu sparen sind die Abmessungen des Magneten nur wenige Millimeter, und er ist auf einer Kohlefaser-Achse montiert. Um eine präzise Steuerung zu erhalten muss der aktuelle Winkel mit einer Regelschleife fortlaufend korrigiert werden. Dieser Winkel wird mit einem kapazitiven Messverfahren in eine kleine Spannung umgesetzt. Der analoge Regler steuert somit den Spulenstrom eines Galvanometers. Dabei muss darauf geachtet werden, dass ein Fehlerfall zeitnah erkannt wird, da im Pulsfall mit Strömen im Amperebereich gearbeitet wird, und die Spulen dieser Belastung nicht dauerhaft standhalten.

4.2.3.7 Konnektivität

Da das Raspberry Pi nicht in ein Gehäuse eingebaut wird, kann kein USB Stick, SD Karte oder Audiosignal direkt verbunden werden, deshalb wird eine Platine mit diesen drei Komponenten erstellt, auf welcher USB und Audio vom Raspberry Pi herangeführt wird und per USB-Hub ein SD-Slot angschlossen wird.

4.2.4 Projektmanagement

4.2.4.1 Teamaufteilung

Alle Aufgabenbereiche wurden den je qualifiziertesten Teammitglieder zugeteilt. Der Terminplan sowie die Zeiterfassung wird online über Cloud-Sevices abgewickelt.

4.2.4.2 geforderte Unterlagen

- Pflichtenheft
- Projektplanung (Projektrecherche)
- Projektdokumentation bestehend aus
  - Aufzeichnung der durchgeführten Arbeiten mit Datum und Zeitaufwand
  - Fertigungsunterlagen
  - Programmcode
  - Erläuterungen (Schnittstellen, etc.)
- Kurzbeschreibung des Projektes in englischer und deutscher Sprache
- Präsentation des Projekts in HTML-Format zwecks Veröffentlichung im Internet auf der HTL-Homepage
- Dokumentation inklusive aller Hilfsprogramme und Dateien (PDF-, etc.) auf Datenträger

4.2.5 Features

4.2.5.1 Must - Have

Hardware

- Stromversorgung
- integriertes Tablet für individuelle Eingabemöglichkeiten
- Galvanometer

Software

- Funktionsfähige NodeJS Webanwendung
- Webserver mit automatisierter Datenweitergabe
- Frontend mit benutzerfreundlicher Oberfläche
- Backend zur Weiterverarbeitung der Koordinaten für die Galvanometer

4.2.5.2 Nice - to - Have

- Abspielen von Videos

- Konvertierung von Bildern und Videos
- Ausfahrbares Tablet

4.2.5.3 Not - to - Have

- diverse Laser für unterschiedliche Farben

4.2.6 Normen

- IEC 60825-1 Sicherheit bei Lasergeräten

4.2.7 Abkürzungen

Abkürzung	**Ausgeschrieben**
SPI	Serial Peripheral Interface
SD - Card	Secure Digital Memory Card
JS	JavaScript
DB	Datenbank
USB	Universal Serial Bus
DAC	Digital Analog Converter

Tabelle 1: Abkürzungsverzeichnis des Pflichtenhefts

4.2.8 Quellen

Thematik	**URL**
Hobbyprojekt - Laserprojektor	elm-chan.org/works/vlp/report_e.html
Raspberry Pi	raspberrypi.org
NodeJS	nodejs.org/
Mongo Datenbank	mongodb.org/

Tabelle 2: Quellenverzeichnis des Pflichtenhefts

5 Projektidee

5.1 Entstehung des Projekts

Für unsere Diplomarbeit wollten wir kein reines Software- bzw. Hardwareprojekt, sondern ein Projekt, in dem beide Komponenten gleichermaßen benötigt werden. Aufgrund dieser Spezifikationen reichte unser Ideenspektrum von einem Roboter über einen Quadrocopter bis zu einer fliegenden Drohne in Form eines Zeppelins. Schließlich entschieden wir uns für einen Laserprojektor. Auf die Idee, einen solchen zu entwickeln, brachte uns ein Hobbyprojekt, welches wir im Internet fanden. Trotz anfänglicher Bedenken vonseiten einiger Betreuungslehrer bezüglich der Realisierbarkeit, konnten wir diese überzeugen, dass unsere Kenntnisse ausreichen und reichten es schließlich ein.

5.2 Ziel des Projekts

Es soll ein Laserprojektor gebaut werden, welcher einen Laserstrahl moduliert und mittels Spiegelablenkung eine Vektorgrafik zeichnet. Mit einer Software soll der Laser dann gesteuert werden und (bewegte) Bilder an die Wand projizieren können. Die Spiegel, die den Laserstrahl ablenken, sollen mit selbstgebauten Galvanometern bewegt werden. Diese werden mit einer geschlossenen Regelschleife auf Position gehalten. Die Ansteuerung der Hardware erfolgt durch einen Mikrocontroller. Als weiterer Teil des Projekts soll eine Software erstellt werden um Vektorbilder zu zeichnen und verarbeiten.

5.3 Herausforderungen

Dieses Projekt ist sehr umfangreich und hat sowohl aus Software- als auch Hardwaresicht großes Erweiterungpotential. Das Projekt bietet Herausforderungen in allen in unserer Ausbildung behandelten Themengebieten, wie beispielsweise:

- *Mechanischer Aufbau* der Galvanometer
- *Elektronische Regelungstechnik,* um Galvanometer zu steuern
- *Digitaltechnik / Prozessortechnik,* zur schnellen Datenverarbeitung
- *Mathematik,* zur Verarbeitung von Vektordaten
- *Programmierung,* sowohl Low-Level für µCs als auch High-Level für die GUI

6 Hardware

6.1 Schema

Um alle geplanten Funktionen realisieren zu können, wurde folgendes Grundkonzept überlegt. Für die Berechnung der Daten wird ein Raspberry Pi benutzt, da dieser leicht zu verwalten ist und auch sehr klein und preisgünstig ist. Die berechneten Daten werden per SPI weiter zu einem μController (STM32) gesendet. Die Daten werden über zwei DACs an analoge Regler angelegt, die zwei selbstgebaute Galvanometer regeln. Um Zugriff auf die Weboberfläche zu ermöglichen, wird ein WLAN Modul als Access Point verwendet. Somit kann von jedem Tablet, Smartphone oder Notebook aus gezeichnet werden. Ebenso wird die Temperatur verschiedener Komponenten gemessen und gemittelt. Dazu wird ein TCN75 Baustein verwendet. Bei Anfrage des Raspberry Pi der Temperatur, wird die gemessene Temperatur per SPI zurückgesendet.

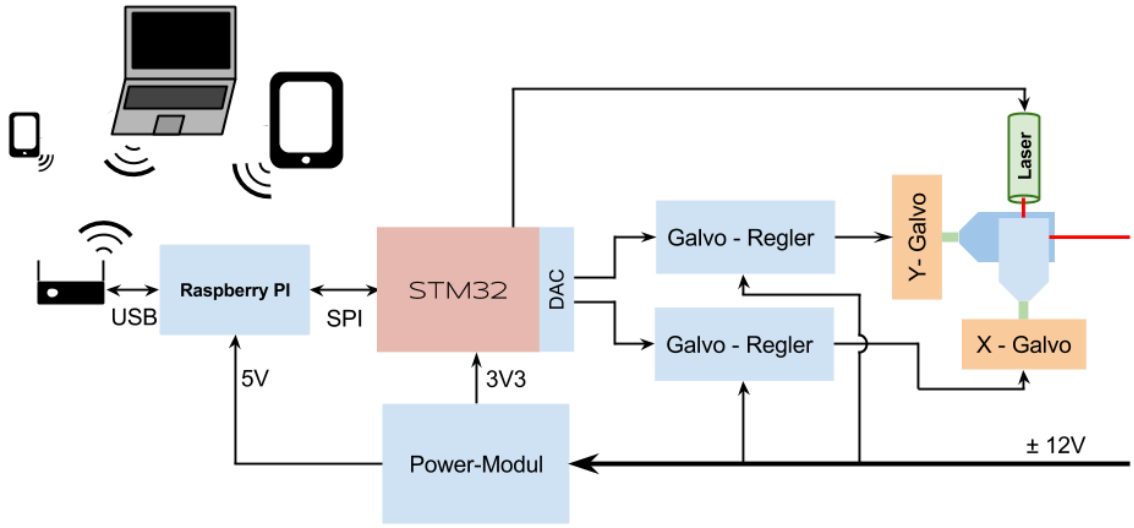


Abbildung 5: ILPS Hardwareschema

Das Gerät wird mit $\pm 12V$ versorgt. Alle zusätzlich benötigten Spannungen werden von einem Power-Modul erzeugt.

6.2 Mainboard

6.2.1 Verwendeter Mikrocontroller

Als μController wird der STM32F103RFT6 verwendet. Dieser Baustein hat 64 Pins und 96KB RAM. Es wäre auch möglich einen externen RAM anzuschließen um eine größere Anzahl von Punkten zu speichern.

STM32 ist eine Mikrocontroller-Familie von ST mit einem 32-Bit ARM Cortex-M0/M3/M4 Kern. Diese Architektur wurde speziell für den Einsatz in Microcontrollern neu entwickelt und löst damit die bisherigen ARM7-basierten Controller weitestgehend ab. Den STM32 gibt es von ST in unzähligen Varianten mit variabler Peripherie und verschiedenen Gehäusegrößen und -formen. Durch die geringe Chipfläche des Cores ist es ST möglich, eine 32 Bit-CPU für weniger als 1€ anzubieten.

6.2.1.1 Peripherie

Der STM32F103RFT6 bietet viel Peripherie, auch wenn nicht alles bei ILPS in Verwendung kommt, stehen folgende Funktionen dem STM32 zur Verfügung.

Peripherie	**STM32F103RFT6**
Flash Speicher	768KB
SRAM	96Kbytes
FSMC	JA
DMA	7
Timer	10
SPI	3
I^2C	2
USART	5
USB	1
CAN	1
SDIO	1
GPIOs	51
12-bit ADC	3
12-bit DAC	2
CPU Frequenz	72MHz
Operating voltage	2.0 to 3.6V
Package	LQFP64

Tabelle 3: Peripherieübersicht des Hauptcontrollers

Beim STM gibt es verschiedene PORT's mit jeweils 16 Pins. Diese werden z.B. PB12 genannt, was gleichbedeutend mit PORT B, PIN 12 ist.

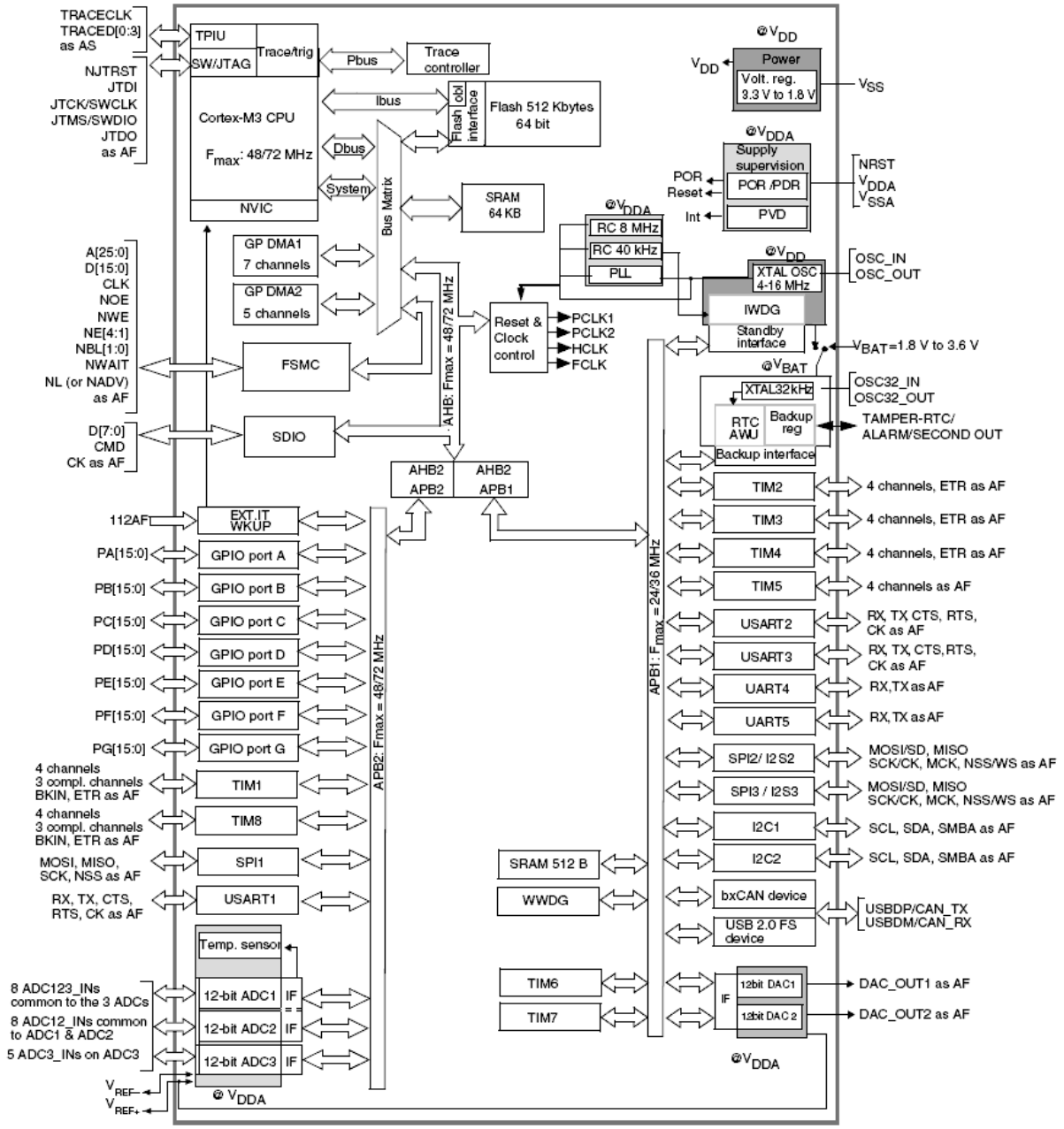


Abbildung 6: STM32-Blockschaltbild

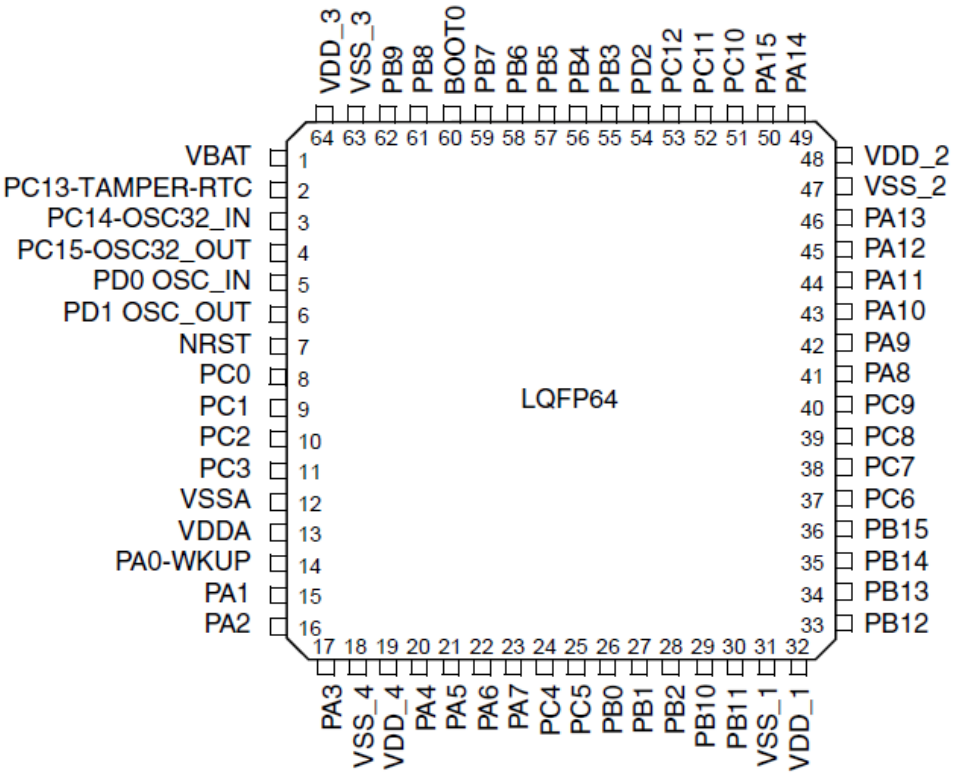


Abbildung 7: STM32F103RFT6-Pinouts

6.2.1.2 Eval-Board

Als Eval-Board wird das STM32VL Discovery Board von STMicroelectronics verwendet. Auf dem Board befindet sich der STM32F100RB sowie der ST-Link Programmer/Debugger mit USB-Schnittstelle.

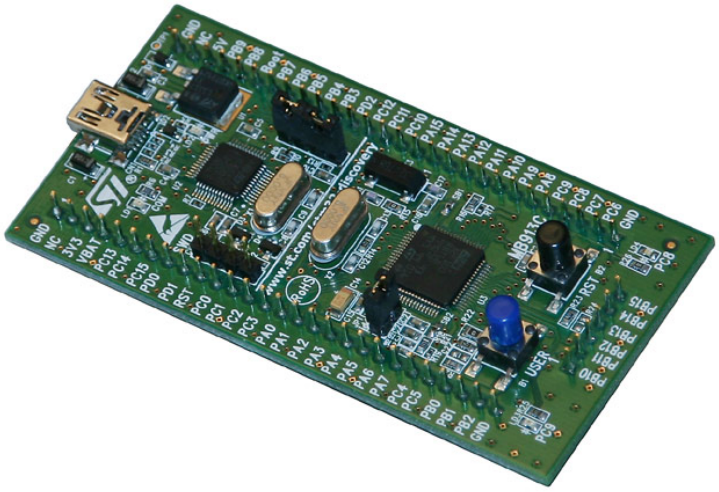


Abbildung 8: STM32-Discovery

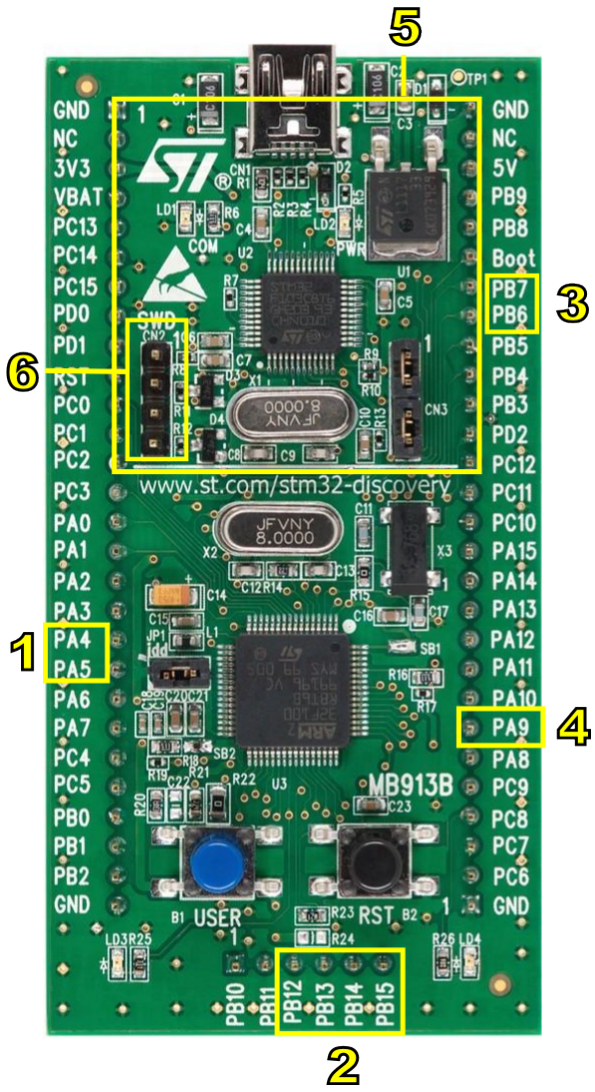


Abbildung 9: STM32-Discovery Beschreibung

Beim Discovery Test Board werden folgende PINs verwendet:

1. DAC
    - PA4 X-DAC
    - PA5 Y-DAC
2. SPI
    - PB12 CS
    - PB13 MISO
    - PB14 MOSI
    - PB15 CLK
3. I^2C
    - PB6 SDA

- PB7 SCL

4. Laser
    - PC9 IO-Port Laser Ein/Aus
5. ST-Link-Programmer
6. SWD-Schnittstelle

6.2.1.3 Programmer & Debugger

Zum Flashen des STM32 wird der ST-Link V2 von STMicroelectronics verwendet. Die Hauptplatine kann über die SWD-Schnittstelle des Eval-Boards programmiert werden. Über diese Schnittstelle ist es auch möglich zu debuggen.

6.2.1.4 Entwicklungsumgebung

Es wird die Entwicklungsumgebung Atollic TrueStudio verwendet. Diese Umgebung baut auf dem bekannteren System Eclipse auf. TrueStudio ist ein bereits fertig konfiguriertes Eclipse mit allen Libraries für den STM32. True Studio ist leicht zu bedienen und sehr übersichtlich.

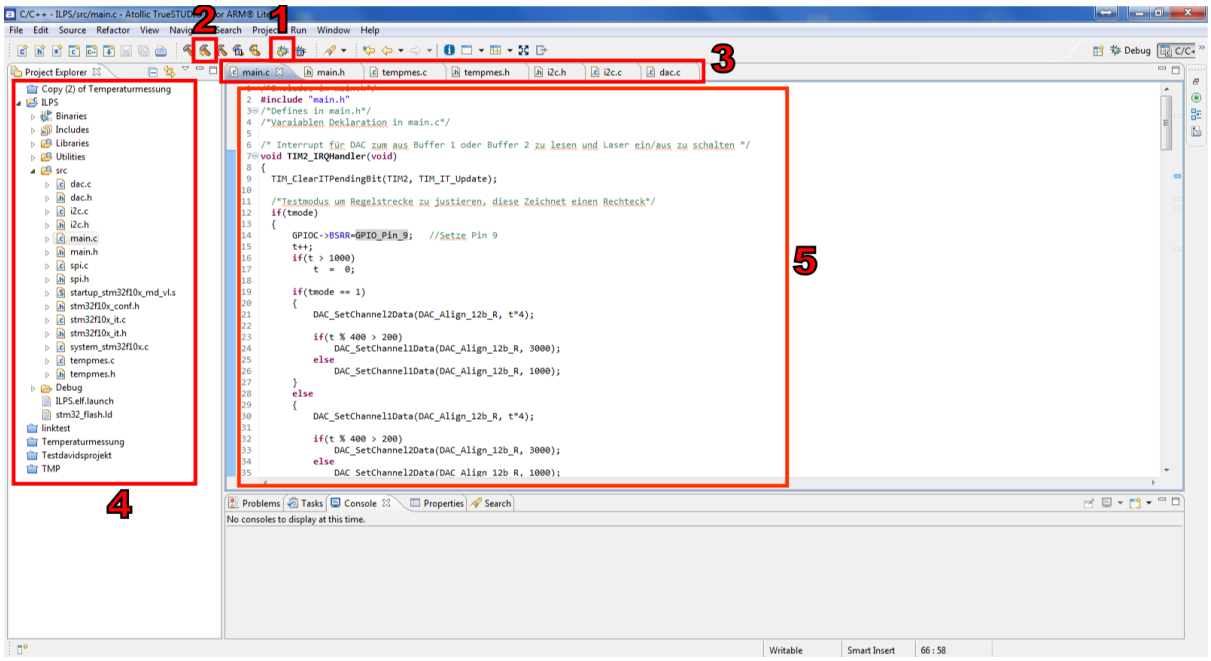


Abbildung 10: Programmierumgebung

Die wichtigsten Programmpunkte sind:

1. Debuggen auf der Hardware
2. Kompilieren
3. Momentan geöffnete Programmfiles
4. zum Projekt gehörende Programmfiles

5. Programmcode

6.2.2 Firmwarekonzept

Anhand der benötigten Funktionen wird folgendes Firmwarekonzept angewendet, welches bei der strukturierten Programmierung hilft. (siehe Abbildung 11)

Die vom Raspberry Pi gesendeten Daten werden per SPI und DMA in einen SPI-Buffer gespeichert. Mit einem programmierten Komunikations-Controller werden die Daten aus dem SPI-Buffer geladen. Dieser erkennt einen Befehl oder Koordinaten. Befehle werden ausgeführt und Koordinaten werden weiter mit einem Writepointer in Buffer 1 gespeichert. Aus Buffer 2 werden die Daten mit einem Interrupt ausgelesen und an die beiden DAC's gesendet, bzw. der Laser Ein/Aus geschalten.

Die Temperatur wird über I^2C eingelesen und überprüft. Bei einer Überhitzung wird eine Systemabschaltung inizialisiert. Bei Anfrage des Raspberry Pi der Temperatur sendet der STM32 die gemittelten, gemessenen Temperaturen per SPI zum Raspberry Pi zurück.

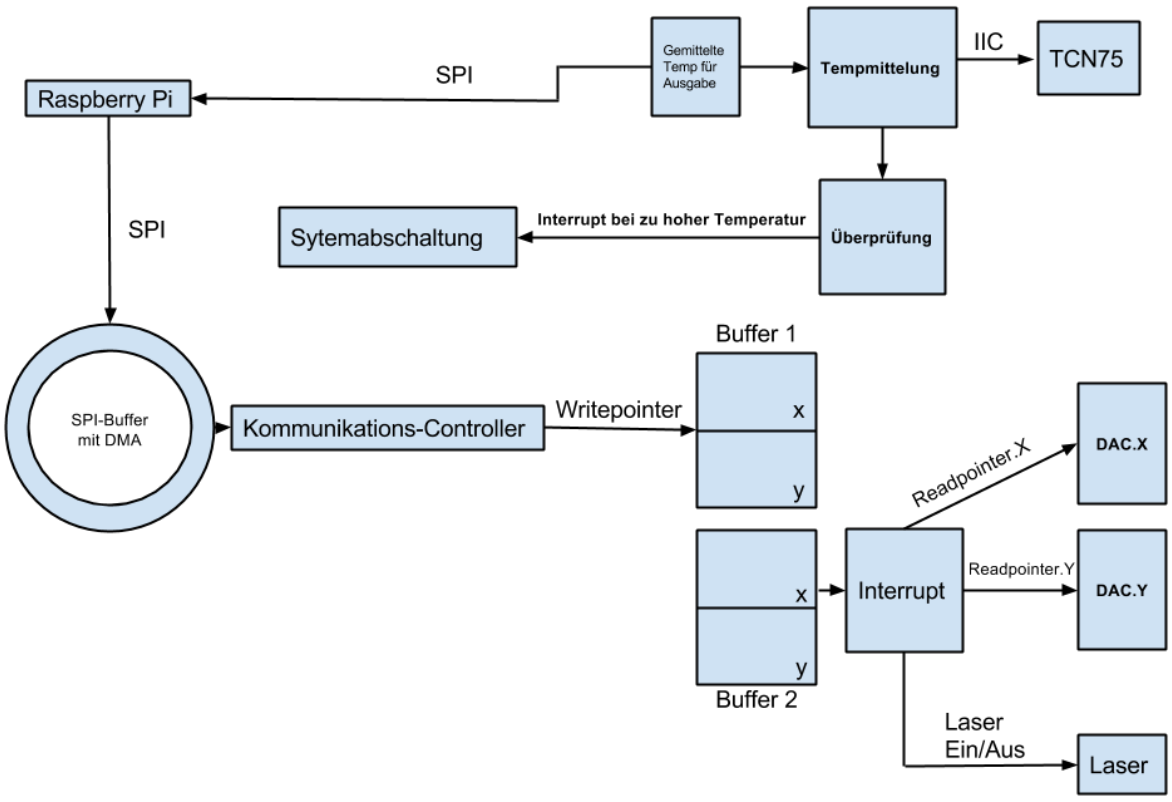


Abbildung 11: STM-Firmwarekonzept Write in Buffer 1

Da ganze Frames gesendet werden, wird ein Double-Buffering System verwendet: Wenn der Schreibvorgang in Buffer 1 beendet ist, wechseln die beiden Pointer. Der Writepointer zeigt auf Buffer 2 während der Readpointer zum Lesen auf Buffer 1 zeigt. Der Datenfluss nach dem Umschaltvorgang ist in Abbildung 12 dargestellt.

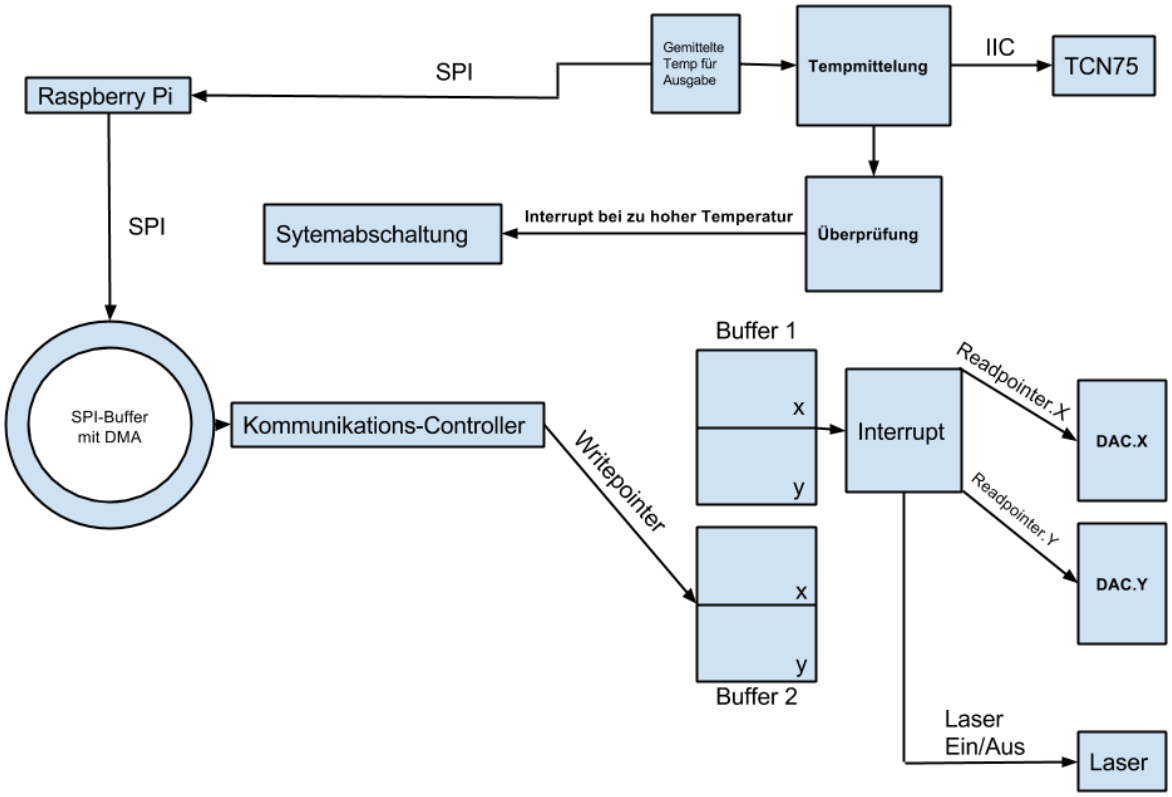


Abbildung 12: STM-Firmwarekonzept Write in Buffer 2

6.3 Firmware

6.3.1 System-Initialisierung

Am Anfang müssen alle benötigten PINs / PORTs / Peripherieelemente und Timer aktiviert werden. Hier werden auch noch einmalig alle Temperaturen eingelesen und der Interrupt für die Temperaturmessung wird aktiviert. Am Ende der System-Initialisierung werden noch alle benötigten Spannungen eingeschaltet.

Listing 1: System-Initialisierung

```
1 SystemInit(); //STM32 Systemeinstellungen laden
2 INTER_Init(); // Interruptaktivierung und Einstellung der Prioritaeten
3 GPIOInit();   //Aktivierung und setzen aller GPIO's
4 i2c_init();   //aktivieren der i^2 Interrupts
5 temp_init();    //einmaliges Lesen der Temperatur und Aktivierung des Interrups↙
    ↳ fuer weiteres Lesen
```

```
6  spi_DMA_init();   //DMA und SPI Aktivierung ebenfalls wie die Uebergabe von SPI↙
       ↳ an DMA
7  dacx_init();    //dacx Aktivierung
8  dacy_init();    //dacy Aktivierung
9  dac_interrupt_init(); //Aktivierung des Interrupt fuer das verwendete ↙
       ↳ Doublebuffering bei der Ausgabe der Laser
10
11 //die Spannungen +12V / -12V / 5V und 3V8 werden eingeschalten
12 PWR_p12V_ON;
13 PWR_m12V_ON;
14 PWR_p5V_ON;
15 PWR_p3V8_ON;
```

6.3.2 Port-Initialisierung

Vor der Verwendung, muss ein PORT initialisiert werden. Zuerst muss das Taktsignal aktiviert und danach der PORT und PIN eingestellt werden. Die Initialisierung funktioniert generell immer auf dieselbe Art.(siehe Listing 2)
Das Taktsignal wird in Zeile eins für das GPIOA aktiviert.
Bei der PIN Initialisierung wird das Verhalten des PINs festgelegt. Je nachdem was der PIN machen soll, wird jeder PIN mit dem dazu benötigten `GPIO_Mode_x` versehen. Bei Ausgängen ist noch ein `GPIO_Speed_xMHz` notwendig. Mit dieser Einstellung kann die Steilheit eines Signals an einem Ausgangspin verändert werden. Je kleiner die Anstiegsgeschwindigkeit, desto weniger Störungen sind an diesem Signal.

Listing 2: Aktivierung des Taktsignals bei PORTA und Initialiesirung Pin4

```
1  RCC_APB1PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE); //Aktivierung von PORTA
2
3  GPIO_InitTypeDef GPIO_InitStructure;
4
5  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN; //Analoger Eingangs PIN ↙
       ↳ Initialisierung
6  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4; //Initialisierung auf PIN4
7  //GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; //nur fuer Ausgaenge ↙
       ↳ relevant
8  GPIO_Init(GPIOA, &GPIO_InitStructure);  //der PIN von PORT GPIOA wird ↙
       ↳ Initialisiert
```

6.3.2.1 GPIO-Mode

- `GPIO_Mode_AIN` Analoger Eingang
- `GPIO_Mode_IN_FLOATING` hochohmiger Eingang
- `GPIO_Mode_IPD` Eingang mit Pull-Down
- `GPIO_Mode_IPU` Eingang mit Pull-Up
- `GPIO_Mode_OUT_OD` Ausgang Open Drain
- `GPIO_Mode_OUT_PP` Ausgang Push-Pull
- `GPIO_Mode_AF_OD` Alternative Funktion Open-Drain
- `GPIO_Mode_AF_PP` Alternative Funktion Push-Pull

Die letzten beiden Ausgangsmodi müssen für Ausgangsperipherien wie USART / SPI / I^2C benutzt werden.

6.3.2.2 GPIO-Speed

- `GPIO_Speed_2MHz`
- `GPIO_Speed_10MHz`
- `GPIO_Speed_50MHz`

Die Steilheit der Flanken ist nur bei den Ausgängen relevant.

6.3.2.3 Verwendete PINs beim STM32F103RFT6

- DAC
  - PA4 X-DAC - PIN20
  - PA5 Y-DAC - PIN21
- SPI
  - PB12 CS - PIN33
  - PB13 MISO - PIN35
  - PB14 MOSI - PIN36
  - PB15 CLK - PIN34
- I^2C
  - PB6 SDA - PIN59
  - PB7 SCL - PIN5
- Laser
  - PC9 IO-Port Laser Ein/Aus - PIN40
- Power Switches
  - PC13 +12V - PIN2
  - PC14 -12V - PIN3
  - PC15 5V - PIN4
  - PD1 3V8 - PIN6
- SWD-Schnittstelle
  - PA13 SWDIO - PIN46
  - PA14 SWCLK - PIN49

6.3.3 Timer

Beim STM32 gibt es zehn verschiedene Timer. Manche sind nur begrenzt nutzbar oder für eine bestimmte Funktion vorgesehen, aber es müssen alle auf die selbe Art initialisiert werden. Bei ILPS werden die Timer TIM2 und TIM7 verwendet. TIM2 ist für das Double-Buffering zuständig und löst den Interrupt für die DAC Ausgabe aus. Der TIM7 wird für die Temperaturmessung verwendet. Der Interrupt von TIM2 hat eine höhere Priorität als TIM7 da dieser wichtiger ist.

Hier ein kleines Beispiel der Initialisierung von TIM2:

Listing 3: TIM2 Initialisierung

```
1 TIM_TimeBase_InitStructure.TIM_ClockDivision = TIM_CKD_DIV1;  //hier wird der ↙
      ↳ Clockdevider eingestellt
2   TIM_TimeBase_InitStructure.TIM_CounterMode = TIM_CounterMode_Up; //der Timer ↙
      ↳ zaehlt aufwaerts
3   TIM_TimeBase_InitStructure.TIM_Period = 600;  //Der Timer zaehlt von 0 bis zu↙
      ↳  diesem Wert
4   TIM_TimeBase_InitStructure.TIM_Prescaler = 8; //teilt den Eingangstakt des ↙
      ↳ Timers herunter
5   TIM_TimeBaseInit(TIM2, &TIM_TimeBase_InitStructure);
6
7   TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE); //Timer Aktivierung
```

6.3.4 Interrupt

Es gibt vier externe und acht interne Interrupts die alle eingestellt und aktiviert werden müssen. Alle werden auf die selbe Art initialisiert und verwendet. Es werden nur die Quellen der Interrupts verändert. Hier ein kleines Beispiel der Initialisierung vom Interrupt von TIM2:

Listing 4: Interrupt Initialisierung von TIM2

```
1 NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
2   NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
3   NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1; //legt die ↙
      ↳ Prioritaet zwischen 15-1 fest, je kleiner der Wert, desto hoeher die ↙
      ↳ Prioritaet
4   NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; //falls Sub-Prioritaeten
5   NVIC_Init(&NVIC_InitStructure);
6
7   TIM_Cmd(TIM2, ENABLE);
```

6.3.5 DMA

Der Direct Memory Access Controller (DMA) ermöglicht es Daten zwischen Peripherie und Speicher zu übertragen, ohne dabei die CPU direkt zu belasten. Der Code hierfür stammt von Diller-Technologies[6]. Die Einstellungen wurden an das Projekt angepasst.

6.3.6 SPI-Empfangsverarbeitung

Für den Empfang der SPI-Daten wird DMA verwendet. Die Daten werden in einen Ringbuffer geschrieben und von dort mit folgender Schleife wieder abgearbeitet. Die Daten werden als 16Bit

Blöcke abgespeichert und jeweils Einzel abgearbeitet. Der erste 16Bit Block ist die Funktion und der zweite Block das Argument. (siehe Kapitel SPI-Übertragung)

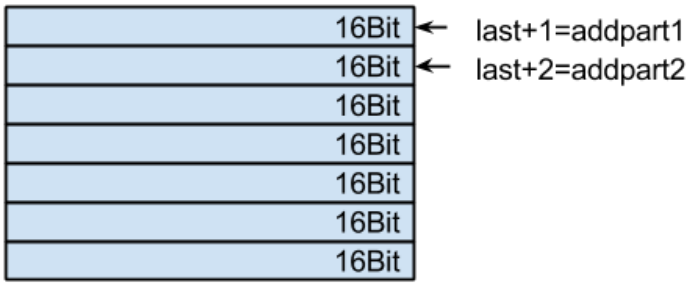


Abbildung 13: SPI Ringbuffer Aufteilung

Es gibt zwei Indexe in dieser Funktion. Der erste Index zeigt auf den Standort in welche Adresse des Ringbuffers gerade geschrieben wird und der zweite Index zeigt auf die Adresse aus der gerade gelesen wird. Dabei muss sichergestellt werden, dass die Abarbeitung den Empfang nicht überholen darf und der Durchgang vom Ringbuffer von der letzten auf die erste Adresse korrekt erfolgt.
Dieses Problem löst folgende for-Schleife (siehe Listing 5 und Abbildung 14). Der aktuelle Zustand der DMA bzw. SPI writer wird im Code in der Variable `akts` abgespeichert und `last` ist Anfangs die gesamte Buffergröße-1 und später ist last immer der zuletzt eingelesene Wert.
Bei der Abarbeitung wird `last+1=addpart1` und `last+2=addpart2` gesetzt. Anfangs wird dadurch die Adresse 1 und Adresse 2 im Ringbuffer abgefragt, beide Adressen müssen kleiner als der aktuelle Zustand sein. Später erhöht sich die Adresse immer um 2 da am Ende eines for Durchgangs immer `last=addpart2` gesetzt wird.

Listing 5: SPI-Empfangsverarbeitung

```
1 akts=128-(DMA1_Channel4->CNDTR);  //Aktueller Zustand
2
3     for(int addrpart1, addrpart2; ((addrpart1 = ((last+1) & SPI_BUFF_MASK)) != ↙
          ↳ akts) && ((addrpart2 = ((last+2) & SPI_BUFF_MASK)) != akts); last = ↙
          ↳ addrpart2)
4     {
5 }
```

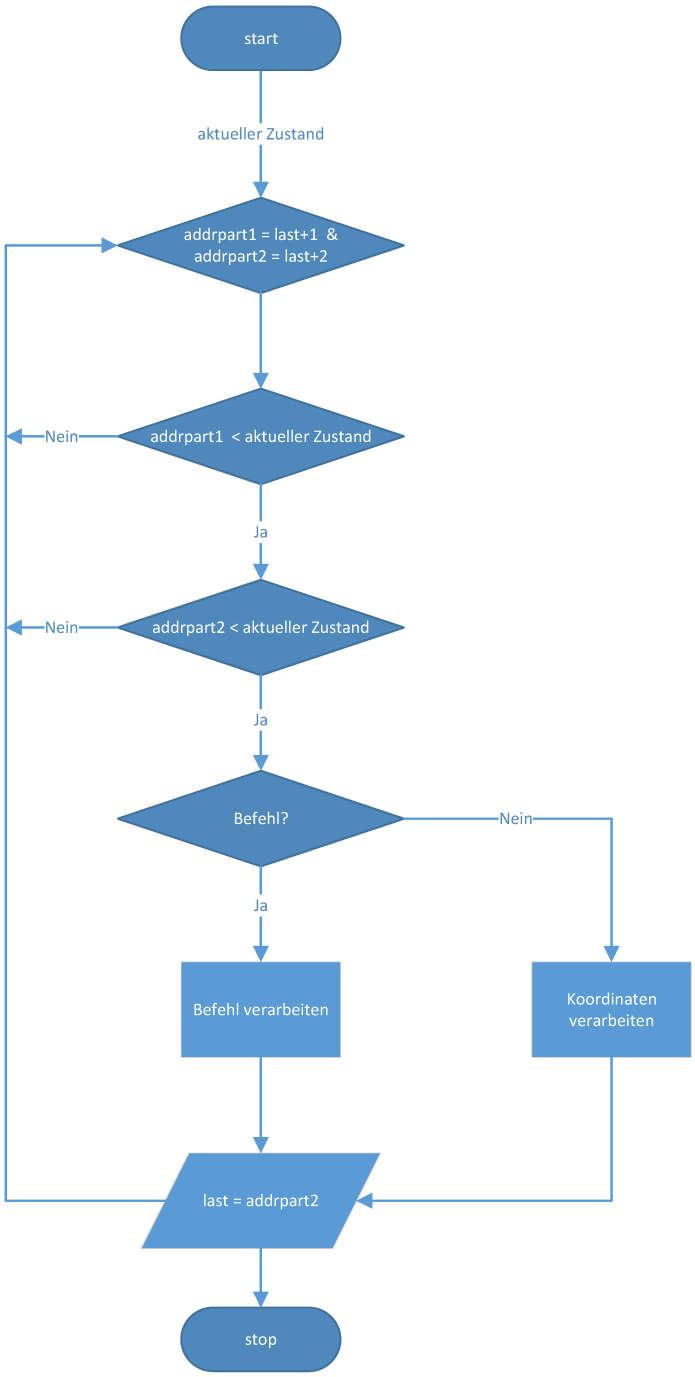


Abbildung 14: for Schleife für die Abarbeitung von SPI-Daten

6.3.7 Befehlsverarbeitung

Wenn in `addpart1` ein Befehl erkannt wird, wird der Befehl mit einem Switch-Statement abgearbeitet. Im Falle eines Befehls wird `addpart1` als Befehl und `addpart2` als Argument des Befehls interpretiert. (Der Aufbau des SPI-Frames ist in Kapitel SPI-Übertragung: spi.js)

Es gibt sechs verschiedene Befehle:

- CMD_NO_OPERATION - kein Befehl, beim Pollen auf Antwort
- CMD_RECEIVE_FRAME - Empfangen eines neuen Frames
- CMD_GET_TEMP - Abfrage einer Temperatur
- CMD_SET_SPEED - stellt die Geschwindigkeit des Lasers ein
- CMD_DELETE_FRAME - schaltet den Laser ab und stellt kein Bild dar
- CMD_SHUTDOWN - schaltet alle Spannungen ab

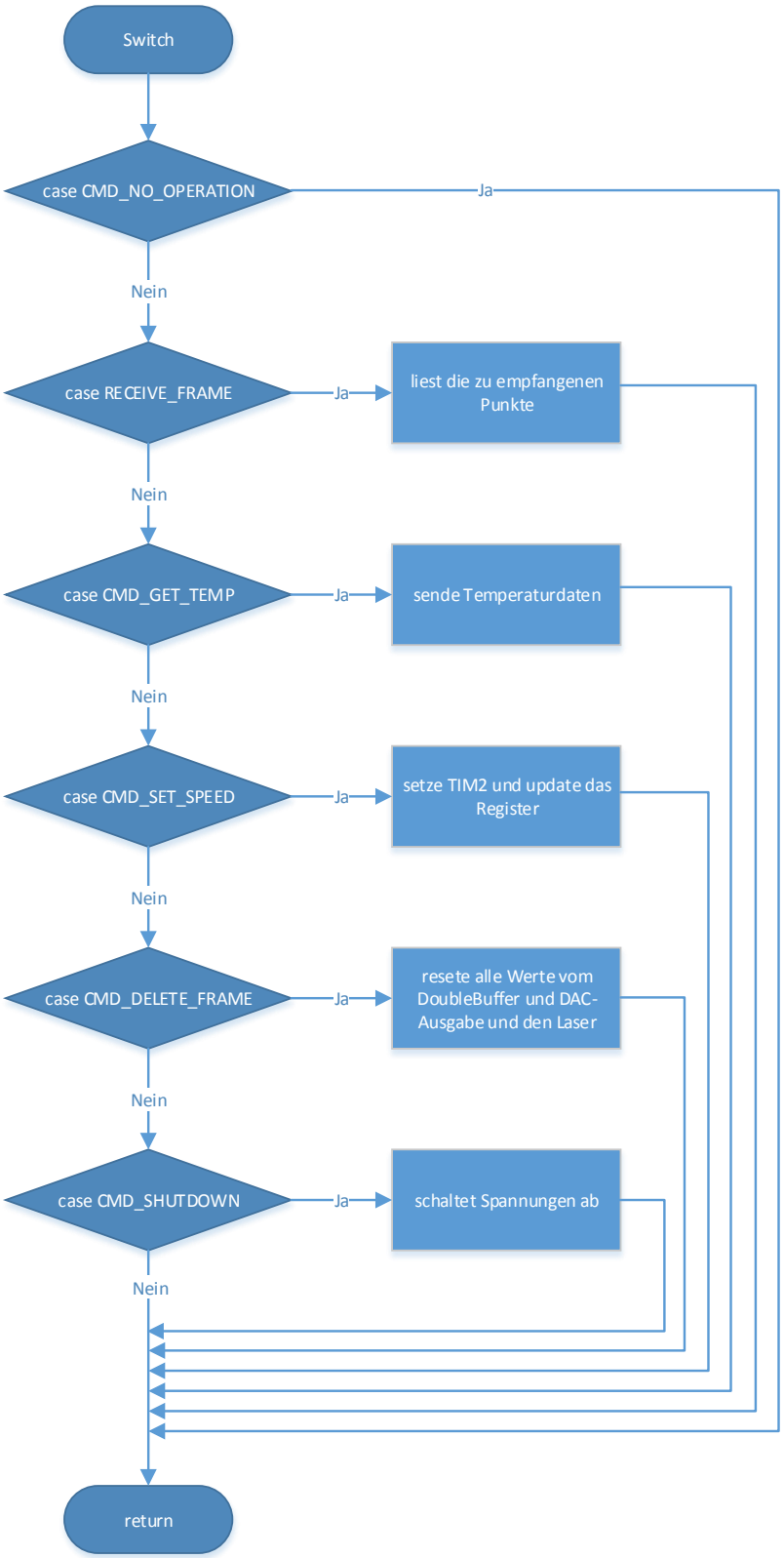


Abbildung 15: Befehlsverarbeitung

Im `CMD_RECEIVE_FRAME` wird das `lastELM` auf den im Argument-1 stehenden Wert gesetzt. Das bedeutet, dass die SPI so viele Punkte, wie der Inhalt vom Argument ist, empfangen muss und an die Double-Buffering Funktion weiterleitet.

Bei einer Anfrage `CMD_GET_TEMP` (Listing 6) wird die Temperatur eines bestimmten Temperatursensors mit der `spi_sendWord()` (siehe Listing 7) Funktion zurückgesendet. Im Argument steht von welchem Temperatursensor die gewünschten Temperaturdaten gesendet werden sollen. Der Funktion wird noch die gemessene Temperatur mitgegeben. Die Temperaturmessung wird durch die Funktion `temp_getSensor()` (Funktion temp_getSensor() siehe Listing 8) von der Mittlung geladen.

Beim Befehl `CMD_SET_SPEED` wird in das TIM2 Register die empfangene Geschwindigkeit geschrieben und anschließend aktualisiert.

Der Befehl `CMD_DELETE_FRAME` deaktiviert den Timer2, löscht alle gespeicherten Daten und stellt die DAC's auf die Grundeinstellung zurück. Ebenso wird der Laser ausgeschaltet und ein einzelner dunkler Punkt im Framebuffer abgelegt.

Als letzter Befehl gibt es noch den `CMD_SHUTDOWN` Befehl. Durch diesen Befehl werden alle Spannungsversorgungen getrennt und die Hardwarekomponenten deaktiviert. Da dieser Befehl nur bei einer Notabschaltung verwendet wird, kann die Hardware nur durch ein manuelles Einschalten mit Spannung versorgt werden.

Listing 6: Befehlsverarbeitung

```
1  switch(SPIRXBuffer[addrpart1] & COM_MASK)
2          {
3            case CMD_NO_OPERATION:
4            break;
5
6            case CMD_RECEIVE_FRAME:
7              lastElm=SPIRXBuffer[addrpart2] - 1;   //gibt die zu empfangenen ↙
                   ↳ Punkte zurueck
8            break;
9
10           case CMD_GET_TEMP:
11             spi_sendWord(temp_getSensor(SPIRXBuffer[addrpart2]));   //↙
                   ↳ uebertrage gemessene Temperaturdaten fuer geforderten Sensor
12           break;
13           case CMD_SET_SPEED:
14             TIM2->ARR = SPIRXBuffer[addrpart2]; //reload Register mit den ↙
                   ↳ Werten vom RX Buffer
15             TIM2->EGR = TIM_PSCReloadMode_Immediate;  //Update Register
16           break;
17           case CMD_DELETE_FRAME:
18             //setzt alle Werte auf Ausgangswert zurueck und zeichnet nichts ↙
                   ↳ mehr
19             TIM_ITConfig(TIM2, TIM_IT_Update, DISABLE);
20             dac_position = 0;
21             lastElmPaint = 1;
22             recvFB[0].X = 0;
23             recvFB[0].Y = 0;
24             GPIOC->BSRR=GPIO_Pin_9; //schaltet Laser aus
25             TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
26           break;
27
28           case CMD_SHUTDOWN: //schaltet alle Spannungen aus
29             PWR_p12V_OFF;
```

```
30              PWR_m12V_OFF;
31              PWR_p5V_OFF;
32              PWR_p3V8_OFF;
33            break;
34          }
```

6.3.8 Temperaturübertragung

Beim Aufrufen der Funktion `spi_send_Word()` wird die Temperatur dem sendenden DMA-SPI-Buffer übergeben. Die Übertragung beinhaltet zwei 32-Bit Werte. Im ersten Teil steht die Anzahl der Bytes, die gelesen werden müssen. Im zweiten Teil stehen die Temperaturdaten. Der Raspberry Pi sollte nur die beiden Datensätze lesen. Falls der Raspberry Pi mehr Datensätze liest, bekommt es ein `0xDEAD`, da dies im vierten Byte steht, spätestens mit diesem wird die Übertragung abgebrochen werden.

Listing 7: Temperatur Übertragung

```
1  void spi_sendWord(uint16_t data)
2  {
3    while(DMA1_Channel5->CNDTR);//fixe Funktion fuer die DMA uebergabe
4    //Aufbau der Uebertragung: 32-Bit fuer die Anzahl der zu lesenden Bytes / 32-↙
         ↳ Bit fuer Daten / DEAD (sollte nicht mehr gelesen werden)
5    //erster 32Bit Block
6    SPITXBuffer[0] = 0;
7    SPITXBuffer[1] = 1;
8
9    //zweiter 21Bit Block
10   SPITXBuffer[2] = 0;
11   SPITXBuffer[3] = data;
12
13   SPITXBuffer[4] = 0xDEAD; //sollte nie vom Raspberry Pi gelesen werden
14   DMA1_Channel5->CCR &= ~DMA_CCR5_EN; //DMA ausschalten
15   DMA1_Channel5->CNDTR = 5; //groesse definieren
16   DMA1_Channel5->CCR |= DMA_CCR5_EN;  //DMA einschalten
17 }
```

6.3.9 Temperaturmessung

6.3.9.1 Erstmalige Temperaturmessung

Bei der Temperaturmessung werden zuerst die Adressen aller Sensoren initialisiert. Nach dem Einschalten des STM32 wird die Temperatur einmalig von allen Sensoren gemessen. (siehe Listing 8 Zeile: 22-31).

6.3.9.2 Temperaturmessung mit Mittelung

Listing 8: Temperaturmessung

```
1  uint16_t avgTemps[SENS_NUM];
2  uint16_t temp_readData(uint16_t addr);
3  void temp_interrupt(void);
4
5  #define SENS_NUM  4
```

```
6  #define SENS_ADDR 0b10010000
7  #define GALVOXTEMP  0b000
8  #define GALVOYTEMP  0b100
9  #define POWERTEMP 0b101
10 #define RASPITEMP 0b010
11 #define EXTRATEMP 0b110
12
13 //erstellt die Temperatursensoradressen
14 uint16_t i2cAddrs[SENS_NUM] =
15 {
16     SENS_ADDR | GALVOXTEMP << 1,
17     SENS_ADDR | GALVOYTEMP << 1,
18     SENS_ADDR | POWERTEMP << 1,
19     SENS_ADDR | RASPITEMP << 1
20 };
21
22 void temp_init(void) //fuer erstmaliges Messen aller Temperaturen beim ↙
       ↳ Einschalten (ohne Mittelung)
23 {
24   i2c_create(I2C1);
25
26   for (int mesp = 0; mesp < SENS_NUM; ++mesp) //in dieser for wird bei allen ↙
       ↳ Sensoren nacheinander die Temperatur abgefragt
27   {
28     avgTemps[mesp] = temp_readData(i2cAddrs[mesp]); //Temperaturabfrage des ↙
         ↳ ausgewaehlten Temperatursensors
29   }
30   temp_interrupt(); //Temperaturinterrupt wird aktiviert fuer zukuenftige ↙
       ↳ Temperaturmessungen
31 }
32
33 //fuehrt die Temperaturmessung durch (siehe http://www.diller-technologies.de/↙
       ↳ stm32.html)
34 uint16_t temp_readData(uint16_t addr)
35 {
36   i2c_readTwoBytes(addr);
37   return i2c_getData();
38 }
39
40 uint16_t temp_getSensor(int num)
41 {
42   return avgTemps[num];
43 }
44
45 //Temperaturmessung gesteuert durch den Temperaturinterrupt "void ↙
       ↳ TIM7_IRQHandler(void)" mit Mittelung
46 void temp_nextMeasurement(void)
47 {
48   uint16_t i2cmeasure =  temp_readData(i2cAddrs[messpunkt]);  //Messung der ↙
       ↳ Temperatur
49
50   avgTemps[messpunkt] = (avgTemps[messpunkt] + i2cmeasure) / 2; //Mittelung der↙
       ↳  Temperatur
51
52   messpunkt++;  //Erhoehung der naechsten Adresse
53   if(messpunkt == SENS_NUM) //falls die Temperaturmessung die Sensoranzahl ↙
       ↳ erreicht wird es auf 0 zurueckgesetzt
54     messpunkt = 0;
55  }
```

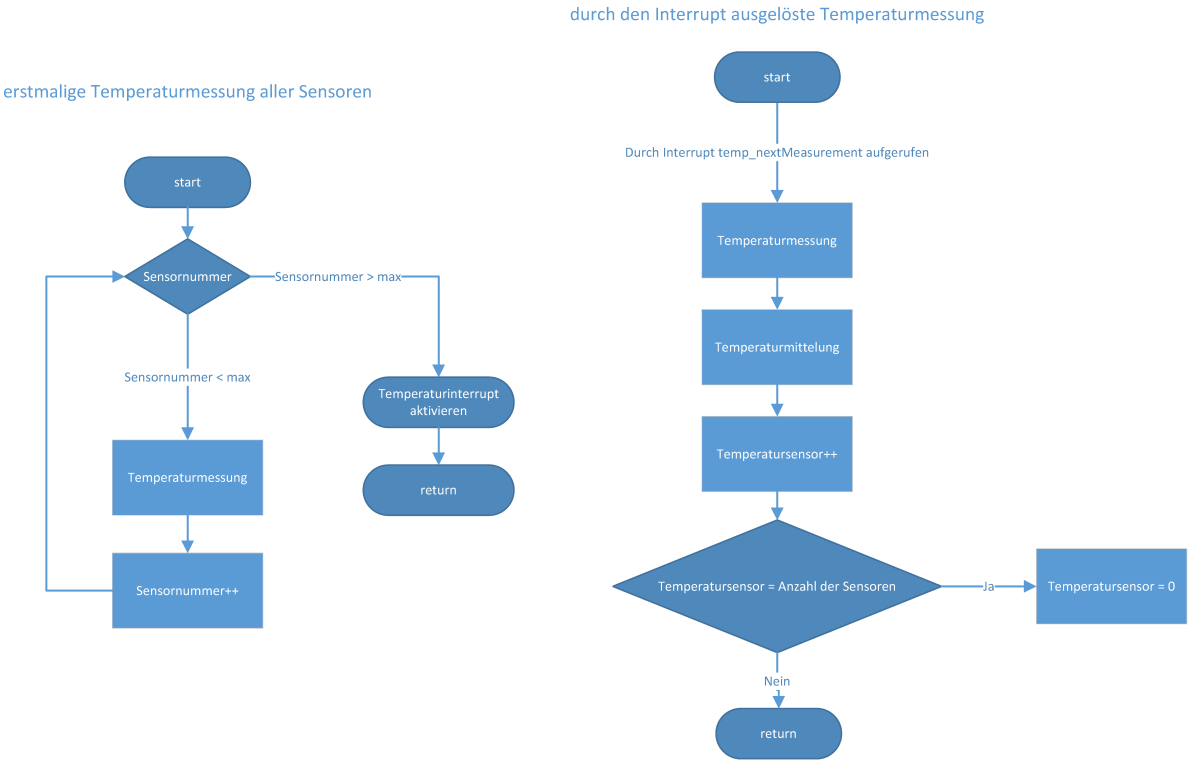


Abbildung 16: Temperaturmessung

Durch einen zeitlich einstellbaren Timerinterrupt wird die Funktion `temp_nextMeasurement()` aufgerufen. In der Funktion wird die Temperaturmessungs-Funktion `temp_readData()` aufgerufen, welche die Temperatur eines Temperatursensors misst. Die gemessene Temperatur wird mit der zuvor gemessenen Temperatur dieses Sensors gemittelt und in der gleichen Variable abgespeichert:
`avgTemps[messpunkt] = (avgTemps[messpunkt] + i2cmeasure) / 2;`
Für den nächsten Interrupt wird die Sensoradresse um eins erhöht und abgespeichert. Am Ende der Funktion ist noch eine Abfrage, ob die momentane Sensoradresse der maximalen Sensoranzahl entspricht. Falls dies zutrifft, wird die Sensornummer auf null zurückgestellt. Für die Temperaturmessung sind noch die zwei Files i2c.c und i2c.h vorhanden. Diese wurde von Diller-Technologies übernommen und an das Projekt angepasst.

6.3.10 Double-Buffering

Listing 9: Double-Buffering

```
1   //Double-Buffering
2         if(writecountFB==lastElm)    //Ueberpruefung des letzten vorhandene ↙
              ↳ Elements und den momentanen writecounter
3         {
4           TIM_ITConfig(TIM2, TIM_IT_Update, DISABLE);
5
6           writecountFB=0;
7           //wechseln der beiden Pointer
8           Point *vertauschung = paintFB;
9           paintFB = recvFB;
10          recvFB = vertauschung;
11          //zuruecksetzten der DAC's
```

```
12            dac_position = 0;
13            lastElmPaint = lastElm;
14
15            TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
16          }
17
18          else
19          {
20          recvFB[writecountFB].X=SPIRXBuffer[addrpart1];
21          recvFB[writecountFB].Y=SPIRXBuffer[addrpart2];
22          writecountFB++; //erhoeht immer den Counter, wird zurueckgesetzt bei ⤢
                ↳ Framewechsel
23          }
```

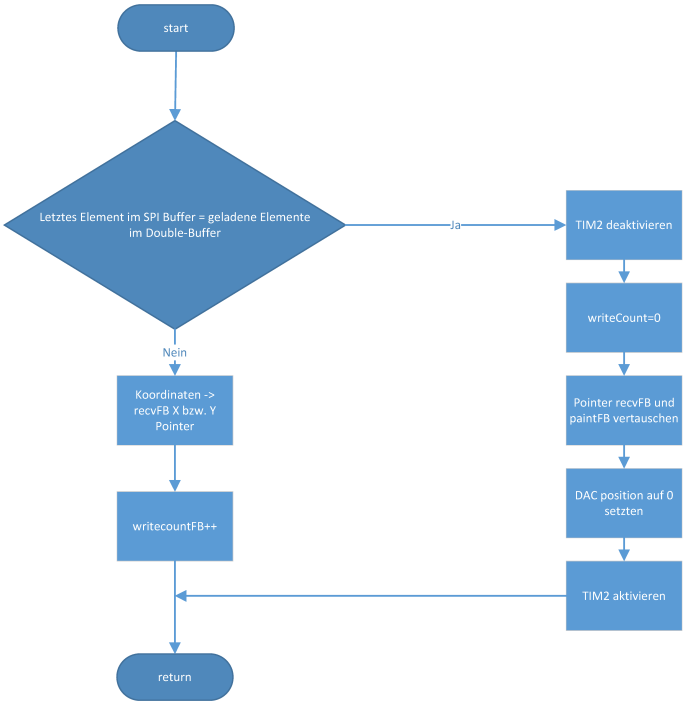


Abbildung 17: Double-Buffering

Das Double-Buffering ist dafür zuständig, dass während die SPI-Übertragung das letzte Frame ausgegeben werden kann. Das bedeutet, dass zwei Buffer und zwei Pointer erstellt wurden. Dieses Prinzip ist in Abbildung 11 und Abbildung 12 gut erkennbar. Über einen Writepointer (`recvFB`) werden die durch die SPI empfangenen Daten in ein Array geschrieben. Währenddessen wird aus einem zweiten Array, zyklisch mittels Timerinterrupt, mit einem ReadPointer gelesen und die Daten an die beiden DAC's gesendet. Wenn das letzte empfangene Element an das Array übertragen wurde, werden die beiden Pointer gewechselt und der Writepointer schreibt nun in das zweite Array und der Paint-Pointer liest die Daten aus dem ersten Array. Dieses Double-Buffering System benötigt doppelt soviel Arbeitsspeicher und wird auch bei Grafikkarten von Computern eingesetzt.

6.3.11 DAC-Interrupt

6.3.11.1 Testmode

Der Testmode wurde von David Beinder programmiert um die Galvanometer Regler anzupassen. Es wird je von einem Galvanometer ein Sägezahn und vom anderen ein Rechteck gezeichnet. So können die Flanken des letzteren direkt beobachtet werden. Um das andere Galvanometer anzupassen werden die Ausgangssignale getauscht. Der Testmode wird durch die Variable `tmode` manuell in der Firmeware aktiviert oder deaktiviert.

Listing 10: Testmodus zur Ausgabe eines Rechtecks

```
1  /*Testmodus um die Regelstrecke zu justieren, dieser zeichnet ein Rechteck*/
2  if(tmode)
3  {
4    GPIOC->BSRR=GPIO_Pin_9; //Setze Pin 9 -> Laser
5    t++;
6    if(t > 1000)
7      t  =  0;
8
9    if(tmode == 1)  /*zeichnet ein Rechteck*/
10   {
11     DAC_SetChannel2Data(DAC_Align_12b_R, t*4);
12
13     if(t % 400 > 200) // t/400->Rest>200
14       DAC_SetChannel1Data(DAC_Align_12b_R, 3000);
15     else
16       DAC_SetChannel1Data(DAC_Align_12b_R, 1000);
17   }
18   else    /*Zeichnen einen Rechteck mit inventierten Kanaelen*/
19   {
20     DAC_SetChannel1Data(DAC_Align_12b_R, t*4);
21
22     if(t % 400 > 200)   // t/400->Rest>200
23       DAC_SetChannel2Data(DAC_Align_12b_R, 3000);
24     else
25       DAC_SetChannel2Data(DAC_Align_12b_R, 1000);
26   }
27
28   return;
29 }
30 /*Ende des Testmodus */
```

6.3.11.2 Ausgabe an DAC's vom Double-Buffer

Wenn der Testmode deaktiviert ist, wird abgefragt, ob die aktuelle `dac_position` im Array, dem letzten Punkt entspricht. Wenn dies der Fall ist, wird die Position auf 0 gesetzt. Falls dies nicht der Fall ist, wird jede X-Koordinate auf das Laser-Bit überprüft. Ist das Laserbit eins, wird der Laser gesetzt. Am Ende werden noch X-Koordinaten dem DAC-Channel2 übergeben und die Y-Koordinaten dem DAC-Channel1. Am Ende wird noch die `dac_position` inkrementiert.

Listing 11: Ausgabe an DAC's

```
1  /*Normalbetrieb zum zeichnen der empfangenen Daten */
2  if(dac_position==lastElmPaint)
3  {
4    //Anfangs bzw. am Ende wird der DAC auf 0 gesetzt
```

```
5       dac_position=0;
6     }
7
8     else
9     {
10      int thisState = (paintFB[dac_position].X & LASER_MASK); //in jeder X-↙
            ↳ Koordinate wird das Laserbit ueberprueft und falls es gesetzt ist der↙
            ↳  Laser gesetzt
11      if(thisState)
12          {
13                SET_LASER; //setze Pin 9 => Laser
14          }
15
16      else
17          {
18                RESET_LASER; //resete Pin 9 => Laser
19          }
20      DAC_SetChannel1Data(DAC_Align_12b_R, paintFB[dac_position].Y & ↙
            ↳ DAC_KORD_MASK);     //Setze Dac Channel1 mit Y-Koordinate
21      DAC_SetChannel2Data(DAC_Align_12b_R, paintFB[dac_position].X & ↙
            ↳ DAC_KORD_MASK);     //Setze Dac Channel1 mit X-Koordinate
22      dac_position++;
23    }
```

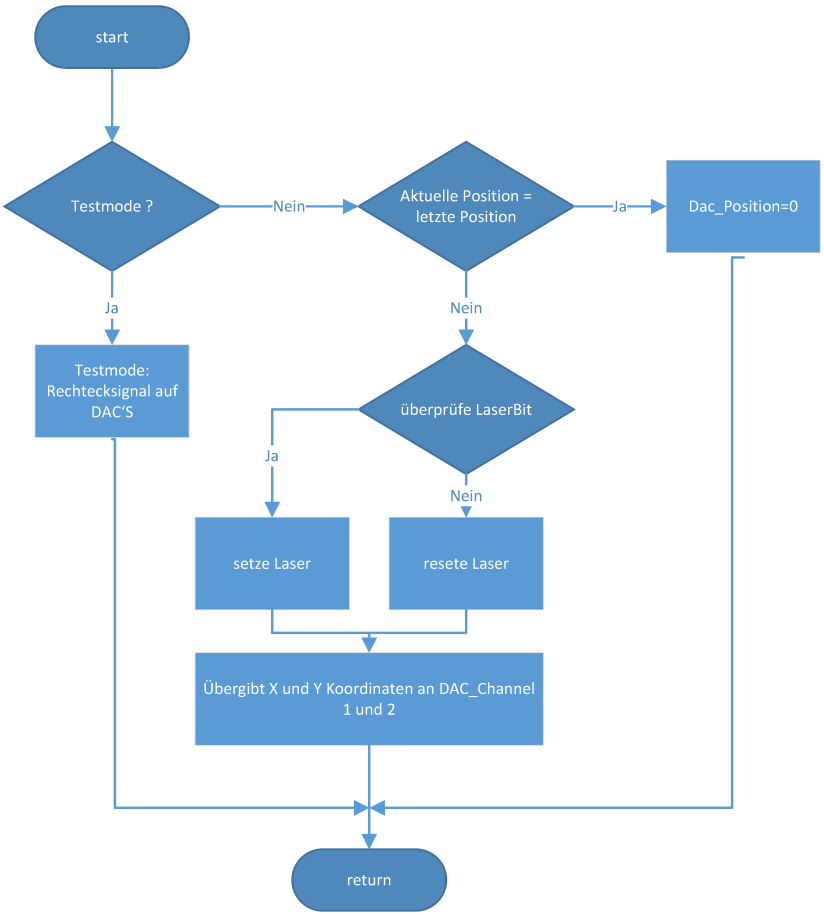


Abbildung 18: Interrupt für Ausgabe an DAC's

6.3.12 Firmware Prescaler Berechnung

Für die Berechnung von Prescaler und Period steht das Programm Timer Calculator von MikroElektronika zur Verfügung. Mit diesem Timer Calculator kann der Baustein STM32F10x und die MCU clock frequency eingestellt werden, ebenso der zu berechnende Timer ausgewählt werden. Am Ende muss noch die Interrupt Zeit eingegeben werden. Aus diesen Daten berechnet das Programm den Prescaler und die Periode.

Der STM32 wird mit 24MHz betrieben und für die Berechnung des Timer 7 wird ein 1s Interrupt verwendet, dieser ist für die Temperaturmessung zuständig. Wichtig dabei ist, dass der Interrupt für die Temperaturmessung nicht den SPI Empfangsinterrupt unterbricht. Aber generell können beide Geschwindigkeiten von der Software beim Raspberry Pi eingestellt werden.

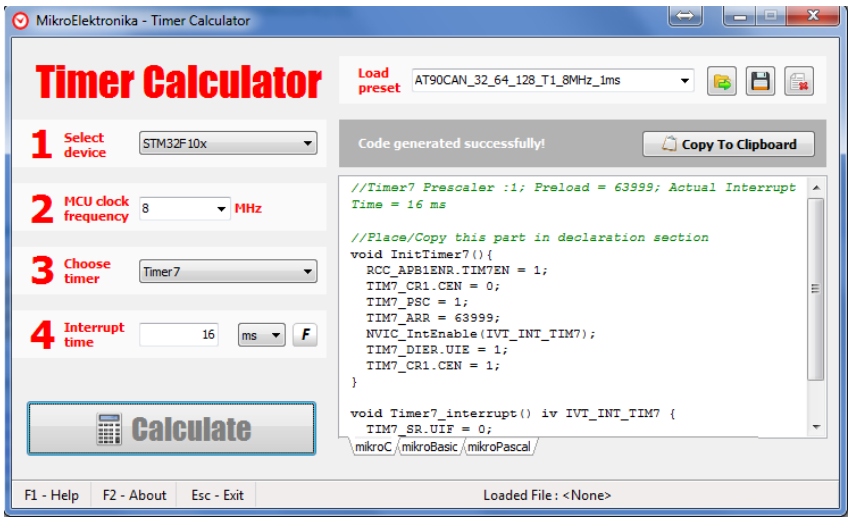


Abbildung 19: TimeCalculator

6.4 Schaltplan

6.4.1 Verbindungen

Für die verschiedenen Verbindungen zwischen allen Platinen und Modulen wurden möglichst die gleichen Stecker verwendet. Die Stecker sind alle mit einem Flachbandkabel angebunden.

6.4.1.1 Flachbandkabel

Für die Verbindung zwischen dem Raspberry Pi und dem Mainboard sowie den beiden Galvanometer Verbindungen und zum Netzteil werden 26-Pol. Wannenstecker verwendet, damit nur ein Stecker für alle Signale und Spannungen verwendet werden kann und nicht mehrere Stecker zur selben Platine gehen.

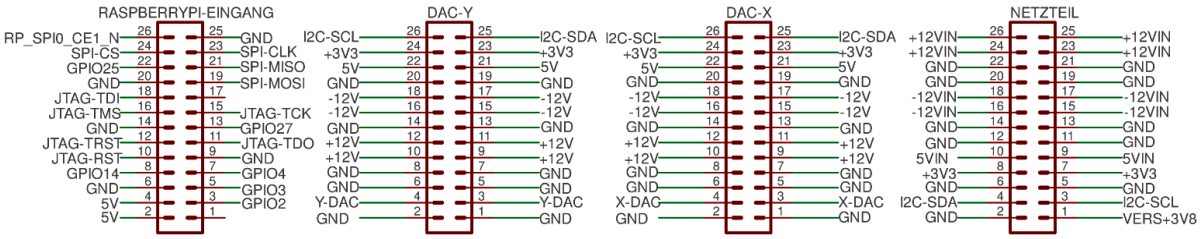


Abbildung 20: Verbindungsstecker zum Raspberry Pi / DAC-X / DAC-Y / Netzteil

Für die Verbindung zum Laser wird ein 6-Pol. Wannenstecker verwendet. Hier ist vorgesehen, dass nicht nur ein modulierbarer Laser angeschlossen werden kann, sondern auch ein Laser, welcher mit einem Transistor geschaltet werden muss (siehe Abbildung 22). Diese Lasersteuerschaltung wird im Normalfall nicht verwendet, da der verwendete Laser modulierbarer ist. Der RASPTEMP Stecker ist für die Temperaturmessung des Raspberry PI zuständig. Für einen externen USB-Anschluss ist ein Anschluss einer USB-Versorgung vorgesehen. Hier werden die benötigten Spannungen übertragen und die I^2C Leitungen für eine mögliche Temperaturüberwachung angeschlossen, aber dies wurde aus zeitlichen Gründen nicht gemacht.

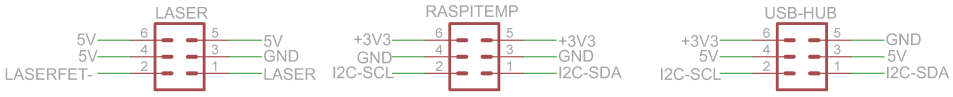


Abbildung 21: Verbindungsstecker zum Laser / Raspberry PI Temperatur / USB-HUB

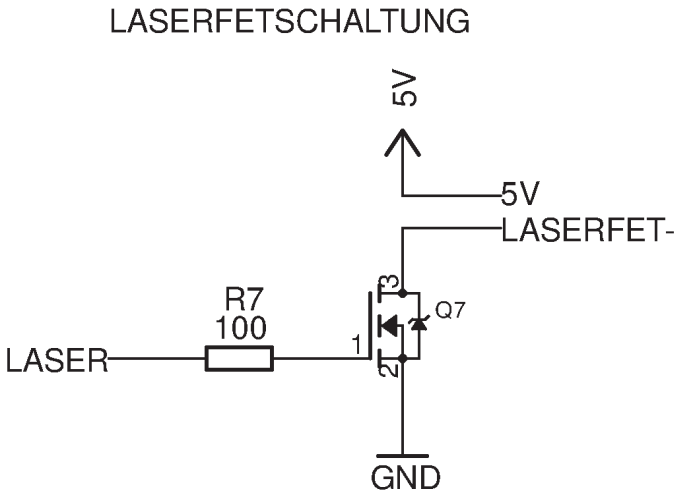


Abbildung 22: Lasersteuerschaltung

6.4.1.2 Leistungsschalter

Um einzelne Spannungen mit dem STM32 zu schalten, werden die Spannungen mit einer MOSFET-Schaltung geschalten (siehe Abbildung 23). Bei einem Absturz des STM32 oder bei einer Übertemperatur werden diese Spannungen abgeschaltet. Die Spannung +3V3 wird nicht geschaltet,

da diese nur den STM32 versorgt. Die Spannung +3V8 wird direkt auf dem Power-Modul geschaltet und es wird nur eine Steuerleitung vom Mainboard zum Power-Modul geführt.

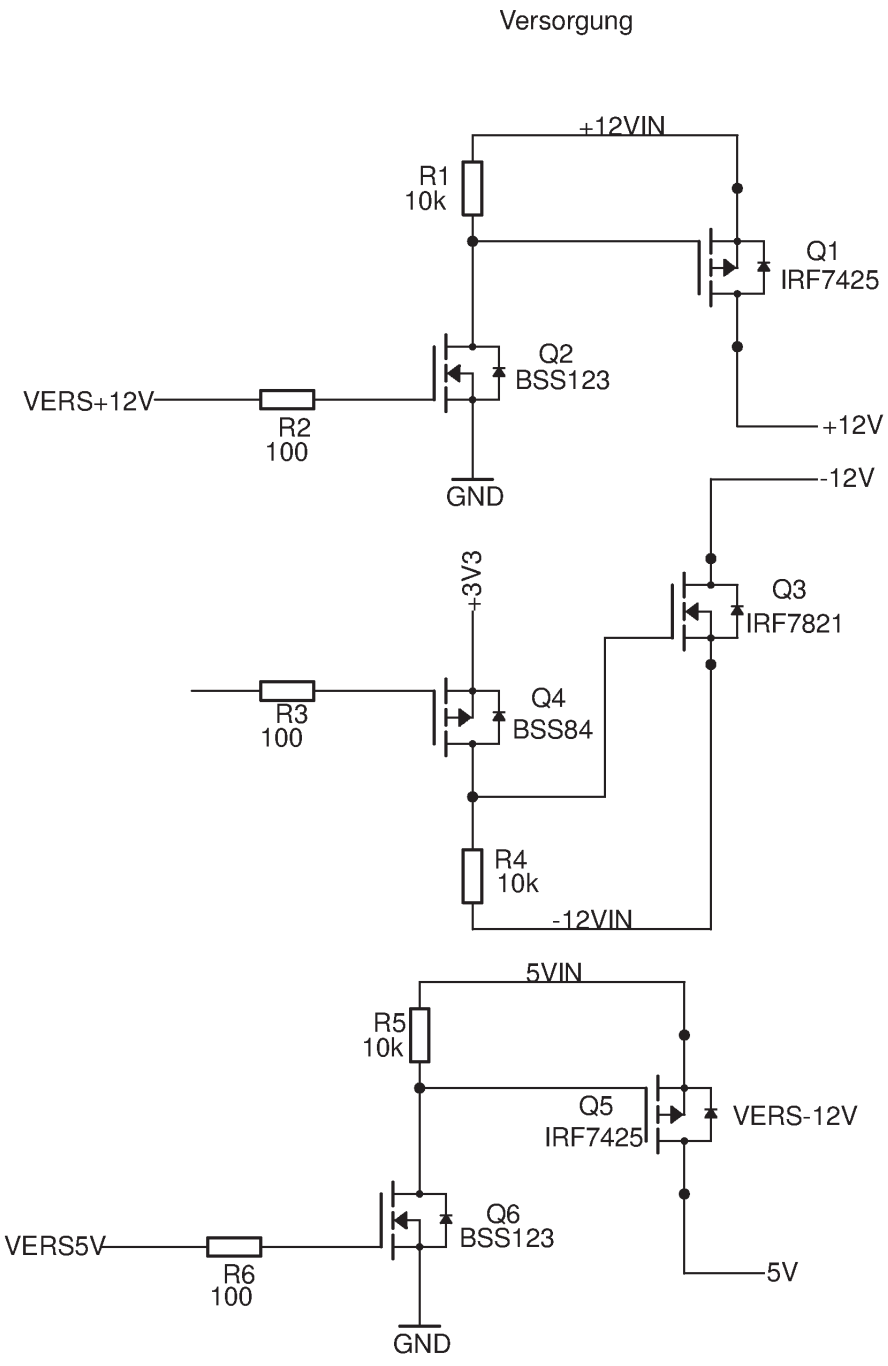


Abbildung 23: Abschaltung der verschiedenen Spannungen

6.4.1.3 Analoge Spannungsversorgung

Für die Analogspannung der DACs , wird eine analoge Filterschaltung mit einem Ferrite Bead und zwei Kondensatoren verwendet (siehe Abbildung 24). Diese Schaltung filtert Störungen aus der +3V3 Versorgung heraus.

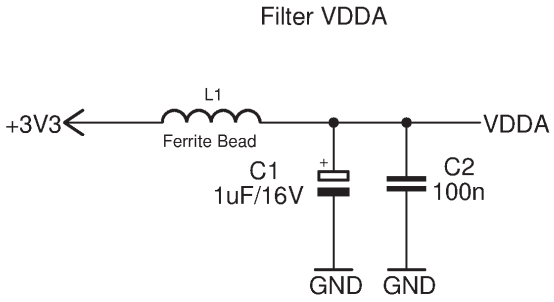


Abbildung 24: VDDA Spannungsversorgung

6.5 Raspberry Pi

6.5.1 Allgemein

Der Raspberry Pi[1] ist ein preisgünstiger Einplatinencomputer, der von der britischen Raspberry Pi Foundation entwickelt wurde.

Er wurde mit dem Ziel entwickelt, jungen Menschen den Erwerb von Programmier- und Hardwarekenntnissen zu erleichtern. Als Prozessor dient ein 700-MHz-ARM11-Prozessor. Es gibt zwei verschiedene Modelle, ein Modell A und ein Modell B. Das Modell B unterscheidet sich vom Modell A durch eine zusätzliche Ethernet- und USB-Schnittstelle. Weiters besitzt er 512MB Arbeitsspeicher anstelle von 256MB.

Als Betriebssystem können verschiedenste Linuxdistributionen, Android oder andere Betriebssysteme verwendet werden, welche die ARM-Architektur unterstützen. Als Speichermedium für das Betriebssystem dient eine SD-Karte.

Versorgungstechnisch benötigt der verwendete Raspberry Pi vom Modell B (in Abbildung 25 dargestellt) eine Spannung von 5V und einen Strom von maximal 700mA. Dies entspricht einer Leistungsaufnahme von 3.5 Watt.

Abbildung 25: Raspberry Pi Modell B

[1] http://www.raspberrypi.org/

6.6 Temperatursensor

6.6.1 Allgemein

Ein Temperatursensor ist für die Sicherheit sehr wichtig. Bei einer Überhitzung muss das ganze System vom μController abgeschaltet werden.

Eine Temperaturmessung wird bei folgenden Punkten durchgeführt:

- Raspberry Pi
- Stromversorgung (LM2576)
- Galvanometer X-Richtung
- Galvanometer Y-Richtung

Für die Temperaturmessung wird ein TCN75 verwendet. Dieser Baustein hat gegenüber einem LM75 oder einem DS1624, welcher standardmäßig an der HTL-Rankweil verwendet wird, einen großen Vorteil, er ist viel preisgünstiger. Der TCN75 kostet bei RS knapp 0,70€und der DS1624 kostet 9,70€.

Der TCN75 ist ein I^2C-Baustein, mit einer Messgenauigkeit von ±0.5°C. Er kann Temperaturen zwischen +125°C und -55°C erfassen. Der Baustein wird in einer SMD SOIC6 Baugröße verwendet und ist daher sehr klein. Seine Adressierung wird mit drei Adressleitungen eingestellt, welche auf Ground oder +3V3 gezogen werden.

Adressen der vier Temperatursensoren:

- 010 Raspberry Pi
- 101 Stromversorgung (LM2576)
- 000 Galvanometer X-Richtung
- 100 Galvanometer Y-Richtung

6.6.2 Schaltplan

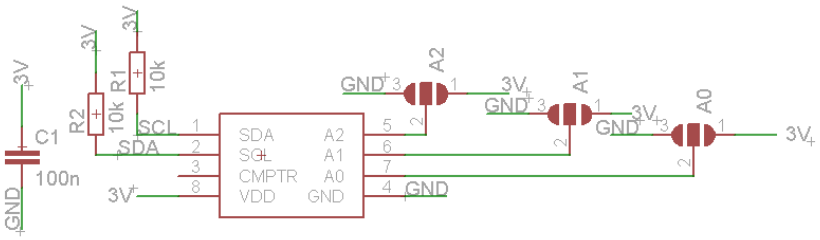


Abbildung 26: Schaltplan Temperatursensor

6.6.3 Layout

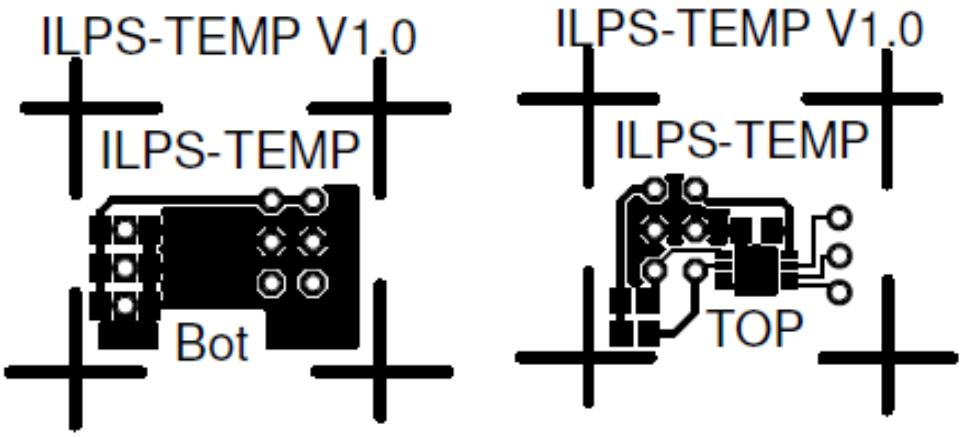


Abbildung 27: Layout Temperatursensor

6.6.4 Funktionsprüfung

Um das geschriebene Programm und das Layout zu testen wurde ein eigenes Programm für die Temperaturmessung eingeführt. Die gemessene Temperatur wird auf ein Display ausgegeben. Das Programm ist auf ein main.c, ein i2c.c und ein i2c.h aufgeteilt.[6] Die I^2C Messung wurde von Diller-Technologies übernommen und nur in der main.c die notwendigen Empfangsfunktionen benutzt und die Bits für Vor- und Nach-Komma sowie das Vorzeichen geschrieben.

Die LCD Ausgabe wurde vom Projekt aus der vierten Klasse übernommen und wird für das Projekt ILPS nicht verwendet. Dieses Programm ist nur ein Testprogramm um die Temperaturmessung zu prüfen.

Listing 12: main.c

```
1  int main(void)
2  {
3    GPIO_INIT();
4    TB_Setup(); //TimeBase Aktivierung und Initialisierung, kopiert aus ↙
       ↳ vorherigen Projekten
5    TB_Wait_ms(100);  //Funktion aus TB (Delay 100ms)
6    LCD_init(); //LCD Initialisierung
7    TB_Wait_ms(100);
8  //LCD Initialisierung
9    LCD_setcursor(0, 1);
10   LCD_string("ILPS"); //Ausgabe an Display ILPS
11   LCD_command(LCD_SET_DISPLAY |
12   LCD_DISPLAY_ON |
13   LCD_CURSOR_OFF |
14   LCD_BLINKING_OFF);
15   //Auslesen des i2c Wertes
16   i2c_create(I2C1);
17   i2c_readTwoBytes(0b10010000);
18   LCD_setcursor(0, 2);
19   i2c_getData();
20   while (1)
21   {
22     i2c_readTwoBytes(0b10010000); //Addresse des auszulesenden Wertes
```

```
23      uint16_t Data =  i2c_getData();
24        if(Data & 0x8000)//Abfrage auf das Vorzeichen wenn das 15te Bit ein 1er ↙
              ↳ ist dann ist es ein Minus-Vorzeichen
25        {
26          Data = (uint16_t)((int16_t)Data * -1);
27          uint8_t vorkomma;
28          uint8_t nachkomma;
29          nachkomma=((Data&0xFF)>>7)*5;
30          vorkomma =(Data>>8);
31          LCD_setcursor(0, 2);
32          printf("Temp: -%d,%d Grad   ", vorkomma, nachkomma); //Ausgabe in ↙
                ↳ zweiter Zeile der Temperatur mit vorkomma und nachkomma
33          TB_Wait_ms(200);
34        }
35        else              //bei positiven Vorzeichen
36        {
37          uint8_t vorkomma;
38          uint8_t nachkomma;
39          nachkomma=((Data&0xFF)>>7)*5;
40          vorkomma =(Data>>8);
41          LCD_setcursor(0, 2);
42          printf("Temp: %d,%d Grad   ", vorkomma, nachkomma);
43          TB_Wait_ms(200);
44        }
45    }
46  }
```

6.6.5 Funktion

In einem Bereich von -50°C bis +50°C, wurde eine Genauigkeit von +/-1°C ermittelt.

In Abbildung 28 sieht man gut, dass die Temperaturmessung gut funktioniert und eine Raumtemperatur von 23°C gemessen wird. Eine Referenzmessung mit einem anderen Temperaturmessgerät ergibt eine Temperatur von 23.1°C.

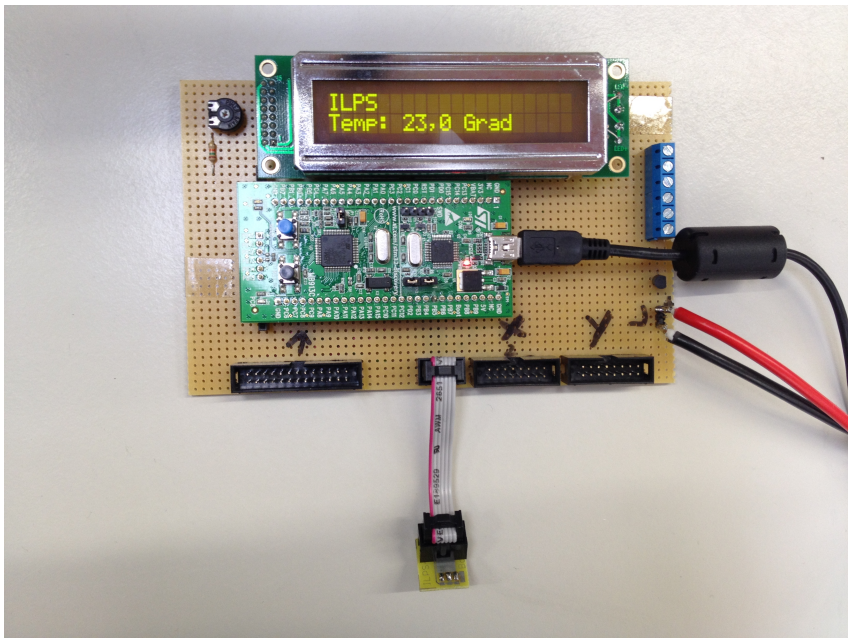


Abbildung 28: Foto des Testaufbaus des Temperatursensors

Ebenso wurde die I^2C-Übertragung am Oziloskop gemessen und hier dargestellt. Bei CH1 ist der SDA-Channel zu sehen, welcher sich immer abhängig der Temperatur ändert. Bei CH2 ist der SCK-Channel zu sehen, welches dem Clock-Signal entspricht.

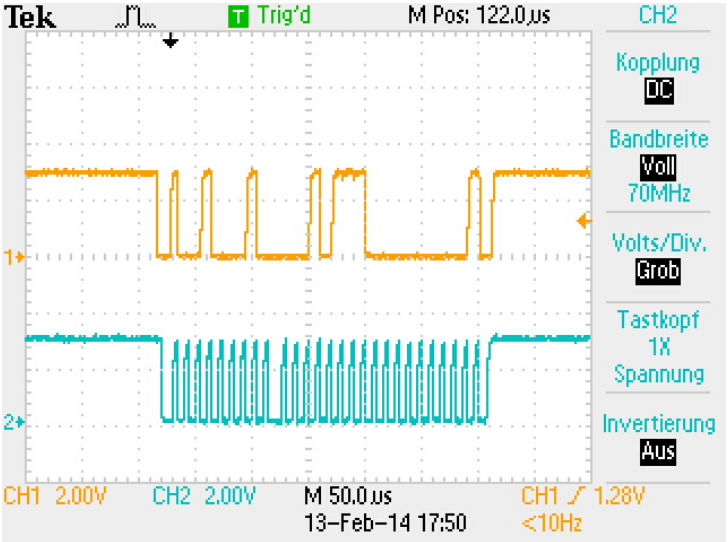


Abbildung 29: I^2C Übertragung

6.7 Galvanometer

6.7.1 Funktionsprinzip

Um den Laserstrahl mit hoher Geschwindigkeit präzise ablenken zu können, sind Schrittmotoren und andere große mechanische Systeme zu träge. Aus diesem Grund wird ein Galvanometer verwendet:

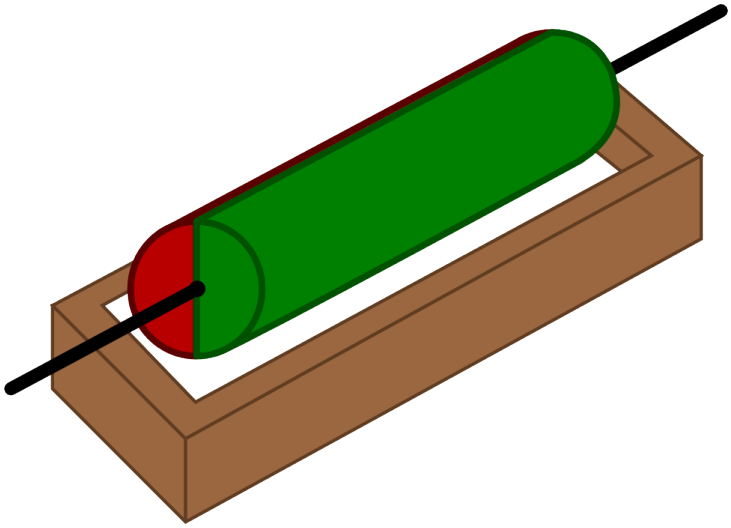


Abbildung 30: Schematischer Aufbau des Galvanometers

In Abbildung 30 ist der prinzipielle Aufbau eines Galvanometers zu sehen. Ein diametral[2] magnetisierter Permanentmagnet ist auf einer drehbaren Achse gelagert. Mittels einem darüber- bzw. darunterliegendem Elektromagneten kann eine Kraft auf den Magneten ausgeübt werden. Die beiden Spulen sind in Serie geschalten und wirken somit in die selbe Richtung.

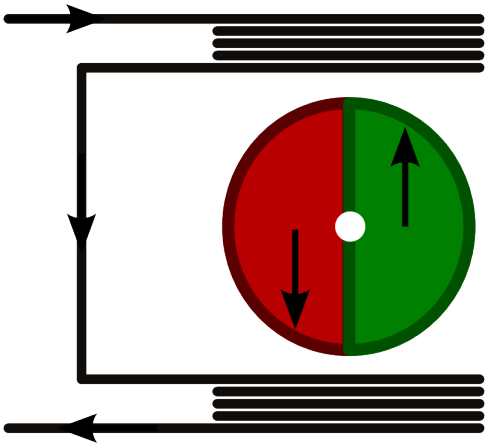


Abbildung 31: Kraftwirkung auf den Magneten

[2]Die Magnetisierung verläuft parallel zum Durchmesser

Aus der Kraft welche in Abbildung 31 dargestellt ist, kann das Drehmoment berechnet werden:

$$M = r \cdot F_{tangential}$$

Dabei ist jedoch zu beachten, dass sich sowohl der Betrag als auch der Winkel des Kraftvektors aufgrund der Drehung des Magneten ändert. So ist in der gezeigten Position das Drehmoment maximal und im 90° Winkel nicht vorhanden. Unter Vernachlässigung einer möglichen Änderung des Betrags des Kraftvektors entsteht ein sinusförmiger Verlauf des Drehmoments in Abhängigkeit des Drehwinkels.

$$M(\varphi) = r \cdot F_{tangential} = r \cdot |F| \cdot cos(\varphi) \tag{1}$$

Durch diese Abhängigkeit ergibt sich die Notwendigkeit, die Ruheposition möglichst optimal auf die oben gezeigte Position einzustellen, da hier das größte Drehmoment bei einer gegebenen Stromstärke erzielt werden kann. Andererseits sollte die 90° Position vermieden werden, da sie nicht geregelt verlassen werden kann. Die einzige Möglichkeit, aus dieser Position in die Ruheposition zu drehen, ist ein Stromstoß in die entgegengesetzte Richtung. Theoretisch würde dies kein Drehmoment erzeugen, doch durch kleine Abweichungen ist es in der Praxis kein Problem, in dieser Position eine Bewegung zu veranlassen. Allerdings ist nicht definiert in welche Richtung die Achse ausbricht. Um das Galvanometer nach dem Einschalten einmalig auszurichten ist dies ausreichend, während des Betriebs hingegen, ist ein solches Verhalten nicht hinnehmbar.

Da der angestrebte Auslenkbereich jedoch nur wenige 10° umfasst, sollte dies unter der Voraussetzung einer Regelung ohne starkes Überschwingen kein Problem darstellen.

6.7.2 Positionsregelung

Die Regelstrecke kann folgendermaßen charakterisiert werden:

- Regelstrecke ohne Ausgleich
- Konstante Störgrößen
  - Erdmagnetfeld
- Dynamische Störgrößen
  - Zweites Galvanometer
  - Metallische oder magnetische Gegenstände in der Umgebung

Aufgrund des integrativen Verhaltens der Strecke ist eine aktive Regelung notwendig. Alternativ könnte die Achse mit einer Feder in ein selbstrückstellendes System und somit in eine Regelstrecke mit Ausgleich umgebaut werden. In diesem Falle wäre auch eine Steuerung ausreichen, um die Position zu halten, es wäre jedoch mit signifikanten Einschränkungen zu rechnen:

- Höherer Stromverbrauch, da immer ein Gegenmoment erzeugt werden muss
- Die Rückstellgeschwindigkeit ist durch die Stärke der Feder bestimmt
  - Eine stärkere Feder ermöglicht höhere Winkelgeschwindigkeiten
  - Eine stärkere Feder erhöht den Stromverbrauch

Dennoch wurde entschieden, eine aktive Regelung zu implementieren, da deren Vorteile überwiegen:

- Strom wird nur für Änderungen benötigt
- Keine zusätzliche Dämpfung der Regelstrecke durch Federn

6.7.3 Positionserkennung

Für die Regelung ist eine genaue und schnelle Positionserkennung nötig. Dabei stehen verschiedene Möglichkeiten zur Auswahl:

Technologie	Vorteile	Nachteile
• Potentiometer	• stabiles Ausgangssignal • sehr einfache Ansteuerung	• Dämpfung der Regelstrecke durch mech. Schleifer • Spiel des Potentiometers kann eine genaue Regelung verhindern
• optische Abtastung (wie bei Kugelmaus)	• berührungslos • schnell	• Nur diskrete Werte möglich
• magnetische Messung	• berührungslos • schnell	• Es sind starke Störungen durch die regelnde Spule zu erwarten • Störungen durch Erdmagnetfeld
• kapazitive Messung	• berührungslos • schnell	• Störungen durch Bewegung der Platten

Tabelle 4: Übersicht über verschiedene Möglichkeiten zur Positionserkennung

Aufgrund dieser Überlegungen fiel die Wahl auf die kapazitive Erfassungsmethode. So kann die Position genau und ausreichend schnell erfasst werden. Jedoch muss beim Layout und Aufbau darauf geachtet werden, möglichst keine zusätzlichen Einstreuungen zu produzieren.

Die gesamte Erfassung erfolgt dabei berührungslos über eine drehbare Platte zwischen zwei fixen Platten. Diese sind mechanisch fixiert, sodass ein Fehler aufgrund einer Bewegung entlang der Achse minimal bleiben sollte. Die an der Achse befestigte Platte sollte auch bei einer Drehung einen fixen Abstand zu den anderen Platten einhalten. Der Aufbau ist in folgender Explosionszeichnung dargestellt:

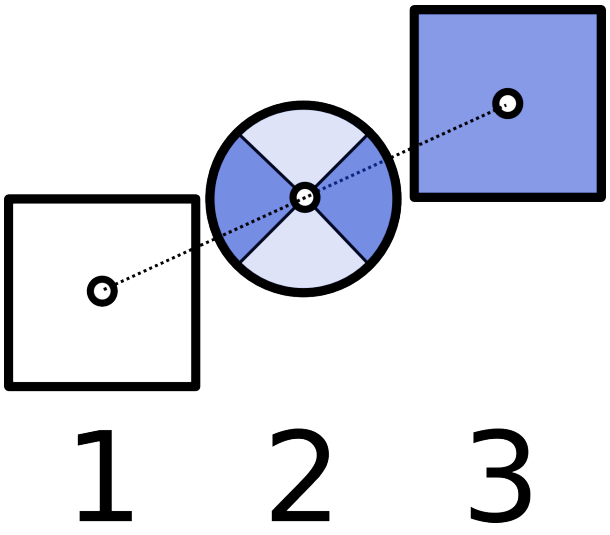


Abbildung 32: Explosionszeichnung von links

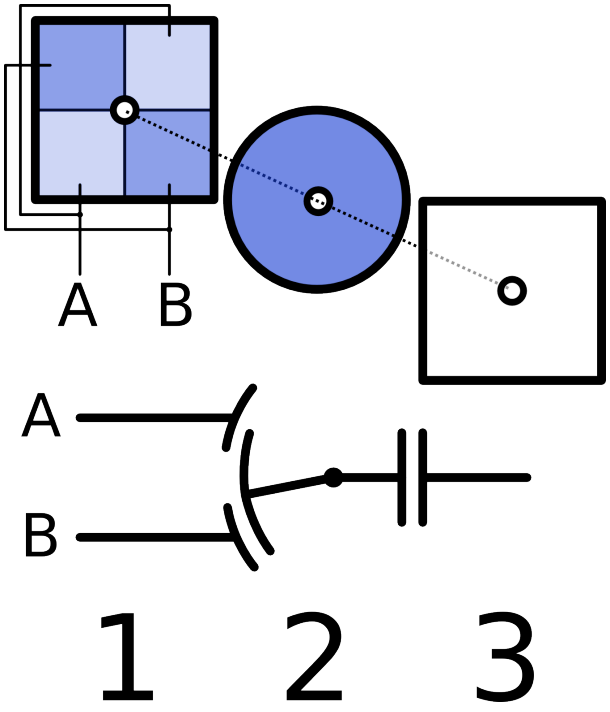


Abbildung 33: Explosionszeichnung von rechts

Nun wird ein hochfrequenter Wechselstrom von einigen MHz durch die Anordnung geleitet und die Differenz der Ströme an den Eingängen A und B (siehe Abbildung 33) ausgewertet.

Der Wechselstrom durch einen Kondensatorzweig kann nach der üblichen Formel berechnet werden:

$$I = U \cdot j\omega C$$

Bei einer konstanten Frequenz ist der Strom proportional zum Drehwinkel, da durch eine Drehung der mittleren Platte die Fläche der Kondensatoren verändert wird:

$$\begin{aligned}
I_A(\varphi) &= U \cdot j\omega \cdot \varepsilon_0\varepsilon_r \cdot \frac{A_A(\varphi)}{d} \\
A_A(\varphi) &= \frac{r^2\pi}{2} \cdot \frac{\varphi + 45°}{90°} \\
I_B(\varphi) &= U \cdot j\omega \cdot \varepsilon_0\varepsilon_r \cdot \frac{A_B(\varphi)}{d} \\
A_B(\varphi) &= \frac{r^2\pi}{2} \cdot \frac{45° - \varphi}{90°} \\
I_{max} &= U \cdot j\omega \cdot \varepsilon_0\varepsilon_r \cdot \frac{r^2\pi}{2} \\
\Delta I(\varphi) &= I_A - I_B = I_{max} \cdot \frac{((\varphi + 45°) - (45° - \varphi))}{90°} = I_{max} \cdot \frac{2\varphi}{90°} \\
\Delta I(\varphi) &\sim \varphi
\end{aligned}$$

Mit folgender Schaltung wird dieser Drehkondensator angesteuert:

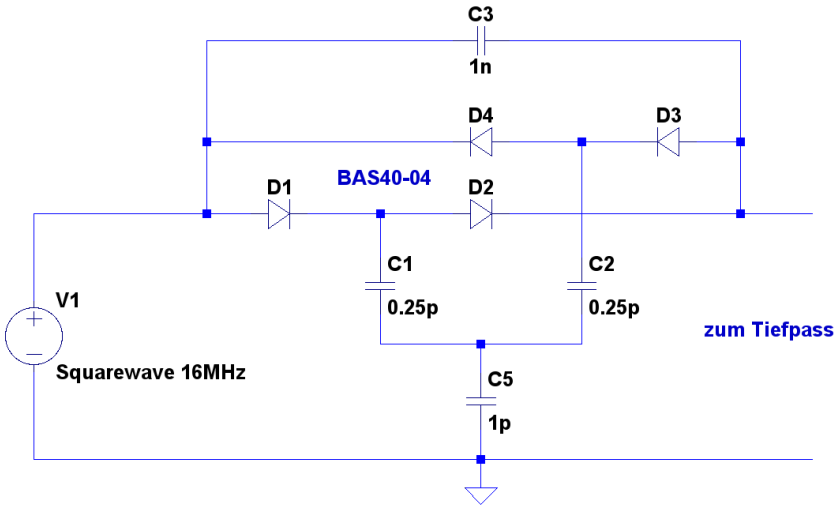


Abbildung 34: Detektorschaltung ohne Tiefpass

In Abbildung 34 ist die gemeinsame Seite des Drehkondensators durch C_5 dargestellt, die getrennten Flügel durch C_1 und C_2. Der Großteil des Stromes fließt durch C_3 und enthält keinen DC-Anteil. Dieser wird durch den Drehkondensator erzeugt. Durch die Diodenpaare (D_1+D_2 und D_3+D_4) kann eine Kondensatorhälfte immer nur in einer Halbwelle Strom abzweigen und so einen DC-Offset aufbauen, wenn sich durch eine Drehung das Verhältnis zwischen C_1 und

C_2 ändert. Es entsteht somit am Ausgang eine Wechselspannung mit einem kleinen Gleichspannungsanteil proportional zum Drehwinkel. Mittels eines Tiefpasses wird der gesamte HF-Anteil über einigen kHz weggefiltert und der verbleibendene Offset wird anschließend verstärkt.

Der Tiefpass setzt sich im wesentlichen aus einem Pi-Filter mit einer Cut-Off Frequenz von 340kHz und einem weiteren Tiefpass zusammen:

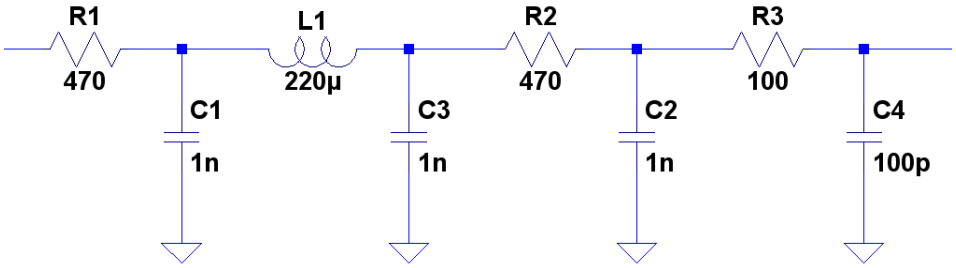


Abbildung 35: Tiefpass für Sensor

Die genauen Grenzfrequenzen sind weitgehend unkritisch, sie dürfen nur nicht zu niederfrequent gewählt werden, um einen negativen Einfluss auf die Regelbarkeit des Systems zu haben. Der abschließende $100pF$ Kondensator wird am Eingang des Verstärkers platziert, um eventuell auftretende Einstreuungen abzublocken. In einer LTspice Simulation wird der Frequenzgang dargestellt:

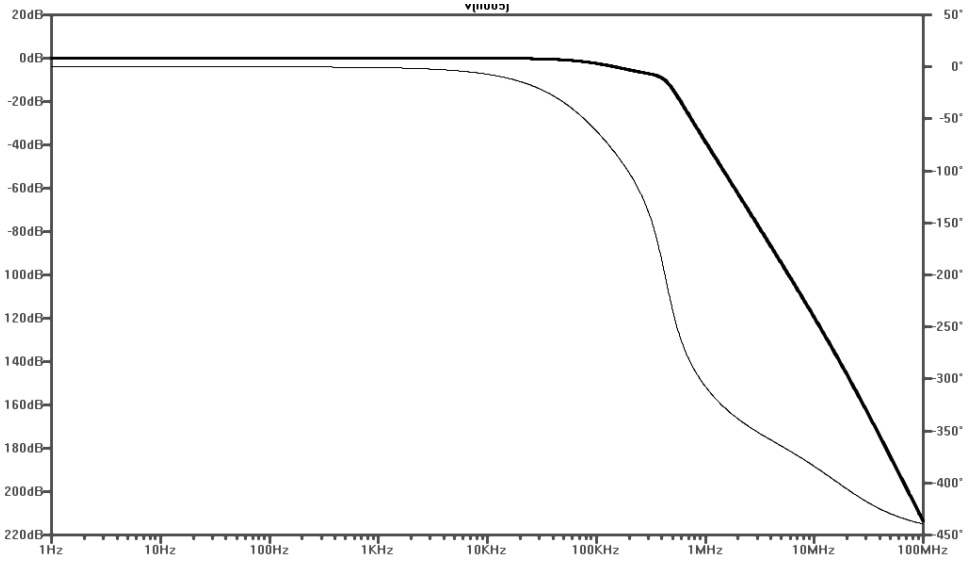


Abbildung 36: Frequenzgang des Filters

Die Bauteilwerte wurden gewählt weil die Bauteile vorhanden waren und um die Schaltung im begrenzten Platz hinter den Galvanometern unterzubringen.

Das resultierende Signal ist nach dem Filter nur wenige mV groß und wird in eine hochohmige Verstärkerschaltung geleitet:

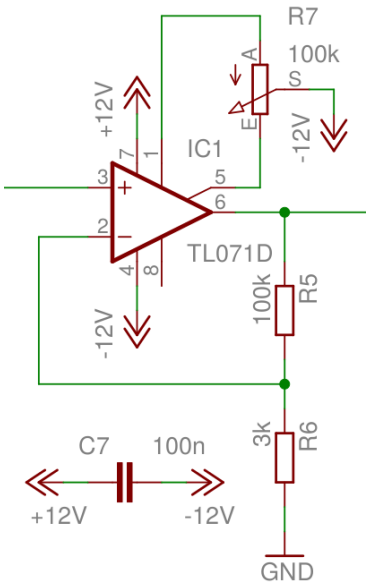


Abbildung 37: Nichtinvertierender Verstärker

Das 30-fach verstärkte Signal wird von hier an den Regler weitergeleitet. Mittels R_7 kann die Zentrierung des Signals - und somit der mittlere Ausgangswinkel eingestellt werden.

Das HF-Signal auf der Sensorplatine wird durch einen Quarzoszillator erzeugt, allerdings muss von diesem zuerst der Gleichanteil entfernt werden:

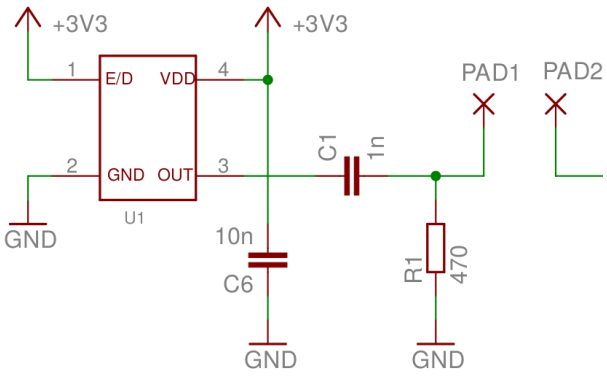


Abbildung 38: HF Erzeugung

Wichtig ist eine gute Versorgung des Quarzoszillators. Bei unserem Aufbau konnten wir ein deutlich rauschärmeres Ergebnis erzielen, indem wir parallel zum $10nF$ Kondensator einen $100nF$ Abblockkondensator einfügten. An die beiden SMD-Pads wird der fehlende Schaltungsteil (siehe Abbildung 34) angeschlossen, bestehend aus den Detektorflächen und 4 Dioden sowie einem Kondensator.

Zum Einsatz kommen Schottky-Dioden, da diese aufgrund ihrer geringen Flußspannung vorteilhaft sind. Bei der Auswahl dieser, wurde insbesondere auf eine geringe Gesamtkapazität C_T sowie eine kurze Reverse Recovery Time t_{rr} geachtet. Bei ILPS kommen pro Sensor je zwei Stück BAS40-04 Dual Schottky Dioden zum Einsatz:

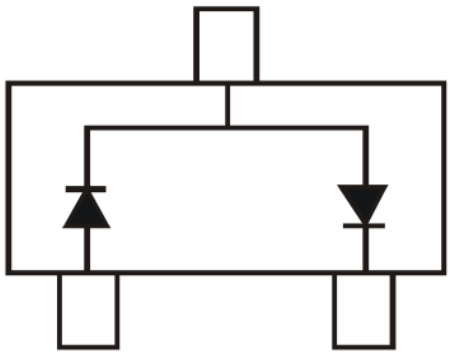


Abbildung 39: BAS40-04 im SOT23 Gehäuse

Diese werden direkt auf der hinteren Kondersatorplatte aufgelötet, um die parasitäre Kapazität zwischen den zwei Kondensatorzweigen zu minimieren. Der gesamte Aufbau wird zwischen Unter- und Oberteil der Galvanometer fixiert:

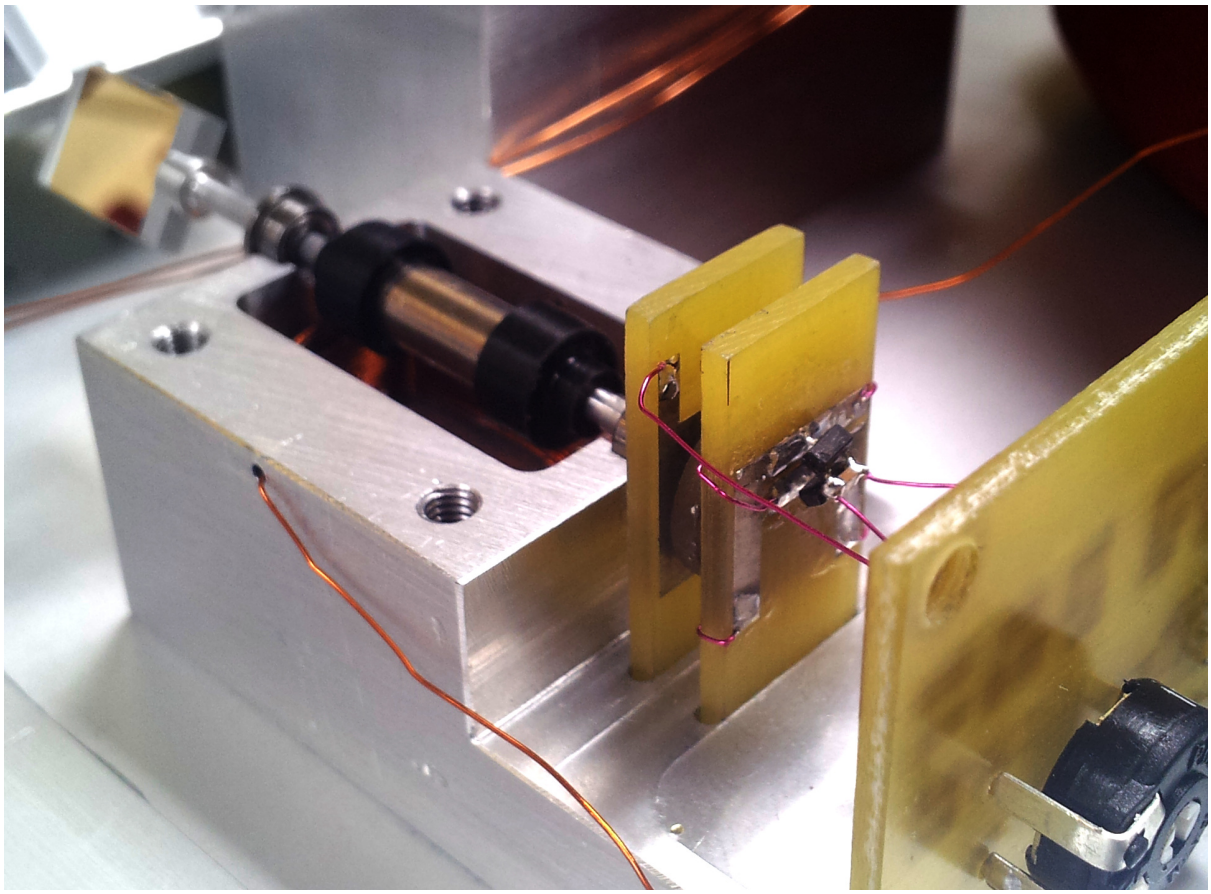


Abbildung 40: Kompletter Aufbau eines Galvanometers ohne Oberteil

6.7.4 Störungen der Sensoren

Durch die hohe Verstärkung und den Einsatz von schnellen Dioden ist die Sensorschaltung empfindlich gegenüber Einstrahlungen. Während der Entwicklung zeigten sich folgende Elemente als Störquellen:

- Eine zweite Sensorschaltung
- WLAN
- Schaltnetzteile

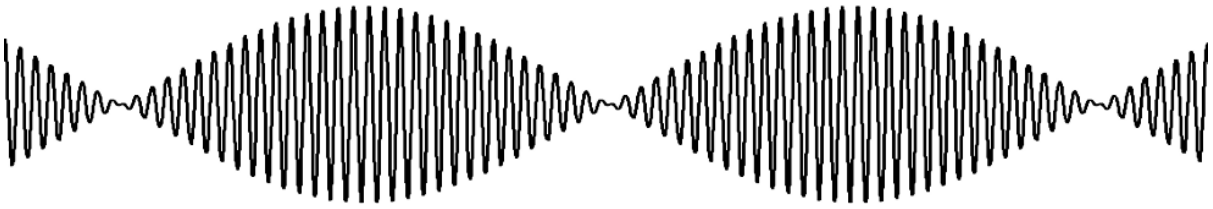


Abbildung 41: Eine Schwebung

Meist handelt es sich bei den Störungen um niederfrequente Schwebungen, entstehend durch die Überlagerung ähnlicher Frequenzen, die von den Dioden gleichgerichtet werden. Wenn die Einstreuungen eine höhere Frequenz aufweisen würden, könnten diese das Tiefpassfilter in der Sensorschaltung nicht passieren, doch so entstehen stark sichtbare Störungen. Sie zeigen sich in einem wobbelnden Bild und durchlaufenden Störungen.

Wenn in beiden Sensoren ein Quarzoszillator mit derselben Frequenz verbaut ist, entstehen besonders niederfrequente Störungen, da die beiden Frequenzen sehr nahe anneinander liegen. Bei einem Aufbau traten so Störungen bei rund 20Hz auf. Auch Oberwellen lassen Schwebungen entstehen, und so können nur wenige Quarze mit handelsübliche Frequenzen (8MHz, 10MHz, 12MHz, 16MHz) gemeinsam verwendet werden. Als Alternative bieten sich Baudratenquarze an, die als Vielfache von 300Hz kaum Schwebungen erzeugen. Bei ILPS wird ein Quarzosziallator mit 16MHz und ein zweiter mit 14.7456MHz verwendet.

Auch das WLAN-Modul erwies sich als Störquelle. Dessen Störungen waren kürzer und traten nur sporadisch an einer durchlaufenden Position auf. Durch eine andere Positionierung und dem Anbringen von Schirmfolie konnten auch diese Störungen entfernt werden.

Die Schaltnetzteile erzeugten die meisten Störungen und da diese mit hoher Leistung abgestrahlt werden, konnten die ursprünglich eingeplanten Netzteile nicht eingesetzt werden. Die Störleistung ist ausreichend stark, dass der reine Betrieb im Leerlauf in unmittelbarer Nähe zu den Sensoren ausreicht, um inakzeptable Bildstörungen zu verursachen. Die Stärke der Störungen unterscheidet sich jedoch stark zwischen unterschiedlichen Modellen von Schaltnetzteilen. Auch der parallele Betrieb von zwei identischen Schaltnetzteilen ist problematisch, da durch fast-identische Frequenzen zusätzlich niederfrequente Schwebungen auftreten können.

Um dies zu beheben, sollten die beiden Schaltnetzteile durch einen Ringkerntrafo mit einfacher Gleichrichterschaltung ersetzt werden. Als kurzfristige Lösung wurden Versorgungsbuchsen am Gehäuse angebracht um das Gerät mit einem Labornetzteil zu versorgen.

6.7.5 Regelstrecke

Ein einfaches Modell der zu regelnden Strecke ohne komplexe Modellierung der Reibung könnte so aussehen:

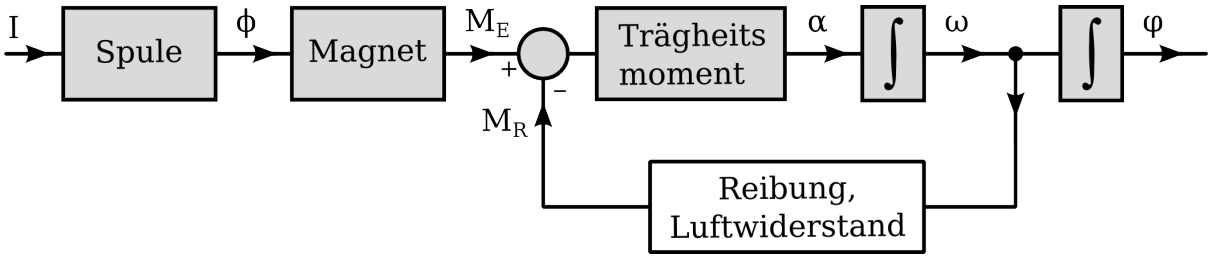


Abbildung 42: Einfaches Modell der Regelstrecke

Die Regelstrecke kann somit als zweifach integrierend bezeichnet werden, und am Regler wird nie ein I-Anteil notwendig sein. Da ohne selbstrückstellendes Verhalten die mittlere Position des Galvanometers abdriften kann und durch den Einfluss des Erdmagnetfeld ist es nur sehr schwer möglich die Übertragungsfunktion der Strecke zu messen. Als Eingangsgröße wird der Strom in die Spule gewählt, als Ausgangsgröße die Ausgangsspannung des Winkelsensors.

Als Alternative bietet sich an, nicht die Regelstrecke selbst zu messen, sondern einen Regelkreis aufzubauen und einen schwachen P-Regler einzusetzen. So wird die Position stabilisiert, und die Übertragungsfunktion wird durch den P-Regler nur in der Amplitudenachse verschoben. Natürlich wird so auch das Verhalten im Bereich von niedrigen Frequenzen verfälscht, doch dieses ist ohnehin problemlos durch einen P-Anteil zu regeln.

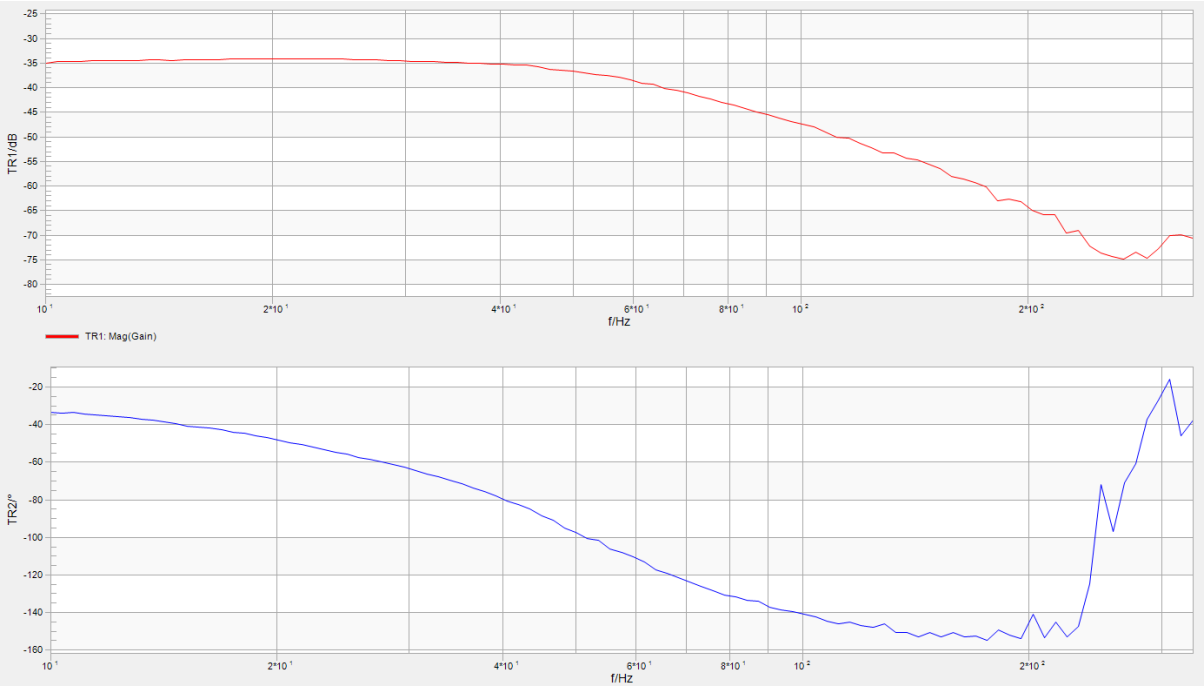


Abbildung 43: Bodediagramm der Stecker mit P-Regler

6.7.6 Regler

Der wichtigste Anteil des Reglers für unser System ist der D-Anteil, ohne einen solchen kann keine ausreichende Änderungsrate erziehlt werden ohne ein Überschwingen zu provozieren. Um den Regler in Position zu halten ist auch ein P-Anteil notwendig, dieser kann allerdings sehr klein gewählt werden. Zusätzlich wurde ein D-Anteil 2. Ordnung verbaut um Überschwinger zu beseitigen.

Alle wichtigen Größen der Regelung (Skalierung, Verhältnis der Anteile, Offset) sind mithilfe von Potentionmetern einstellbar da zum Zeitpunkt der Platinenerstellung die Werte noch nicht klar waren. Zusätzlich hängen die konkret realisierbaren Werte auch stark vom mechanischen Aufbau ab und um ein mit dem gegebenen Aufbau optimales Ergebnis zu erzielen muss der Regler abgeglichen werden.

6.7.6.1 Skalierung und Differenz

Vom Signal der DACs im Bereich $[0.0V - 3.3V]$ muss der Offset entfernt werden und das Signal skaliert werden. So kann auch die Skalierung der Zeichnung in eine Richtung angepasst werden:

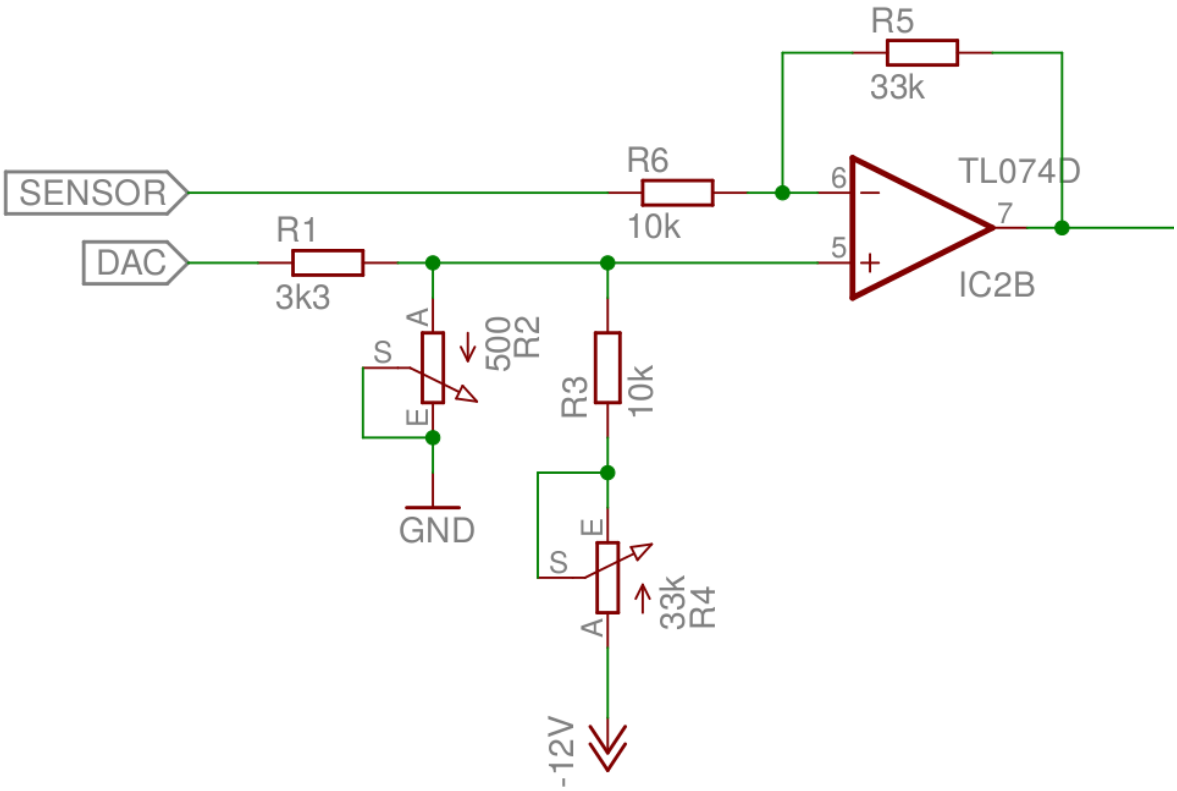


Abbildung 44: Differenzbildung vor dem Regler

Mit dieser Schaltung können Amplitude und Offset beinahe unabhängig eingestellt werden. Da der DAC - Ausgang mit maximal $3.3k\Omega$ belastet werden kann, muss sichergestellt werden, dass der Ausgangsbuffer (=Impedanzwandler) des DACs eingeschalten ist.

6.7.6.2 Ansteuerung der Spule

Hierfür wird ein Leistungs - OP verwendet, der die angelegte Spannung am Eingang über einen Shunt in Strom wandelt:

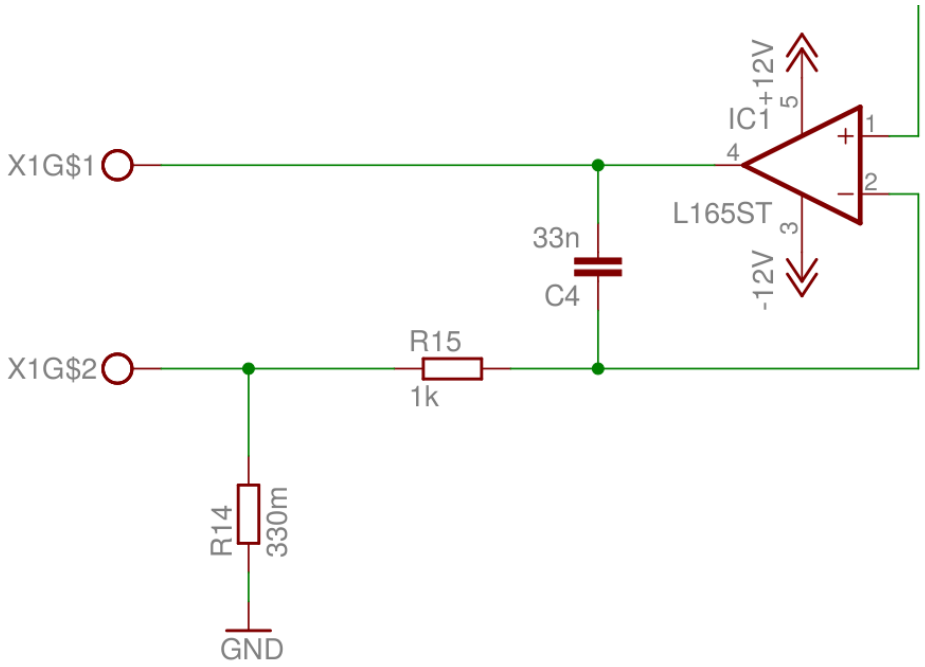


Abbildung 45: Spannungsgesteuerte Stromquelle

Der L165 kann dauerhaft bis zu 3A liefern, bei bis zu $\pm 18V$. Mittel C_4 und R_{15} werden hochfrequente Störungen aus der Rückkopplung entfernt um ein Schwingen zu verhindern.

Um bei schnellen Änderungen Spikes auf der Versorgungsschiene zu verhindern, muss eine größere Kapazität eingesetzt werden um hochfrequente Spitzen zu entfernen. Wenn Ströme im Ampere-Bereich durch die Spule fließen, muss mit einem geringen Einbruch der Versorgungsspannung gerechnet werden, dieser wird jedoch von den Operationsverstärkern gut kompensiert. Es muss nur darauf geachtet werden, dass die Änderung möglichst langsam und ohne Überschwinger vonstatten geht.

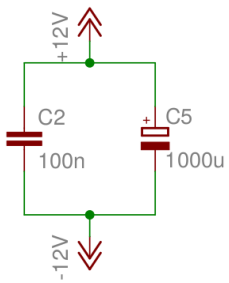


Abbildung 46: Abblockkondensatoren am OP

Der Elektrolytkondensator ist nahe am Leistungs-OP platziert, und wird durch diesen erwärmt, daher sollte ein 105° Typ verwendet werden.

6.7.6.3 P-Anteil

Der Ausgang des Differenzbildners wird entsprechend der Potentiometereinstellung verstärkt:

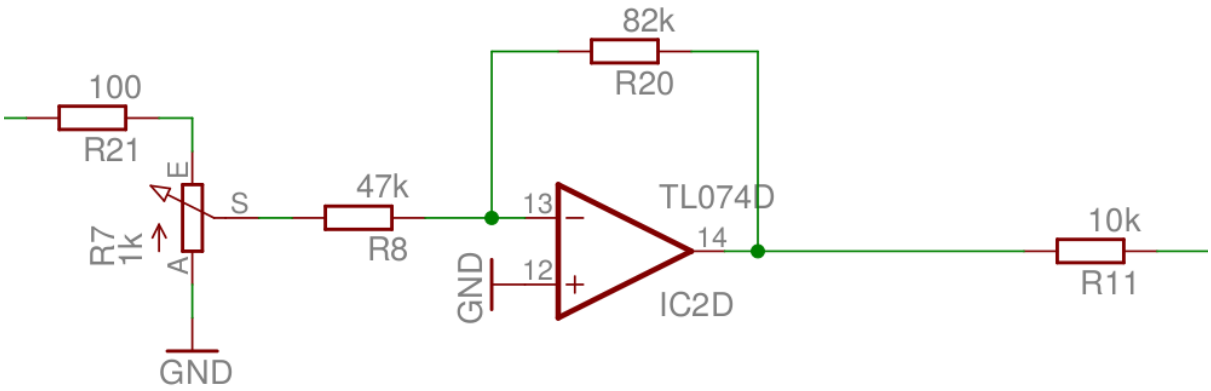


Abbildung 47: P - Anteil

6.7.6.4 D-Anteil

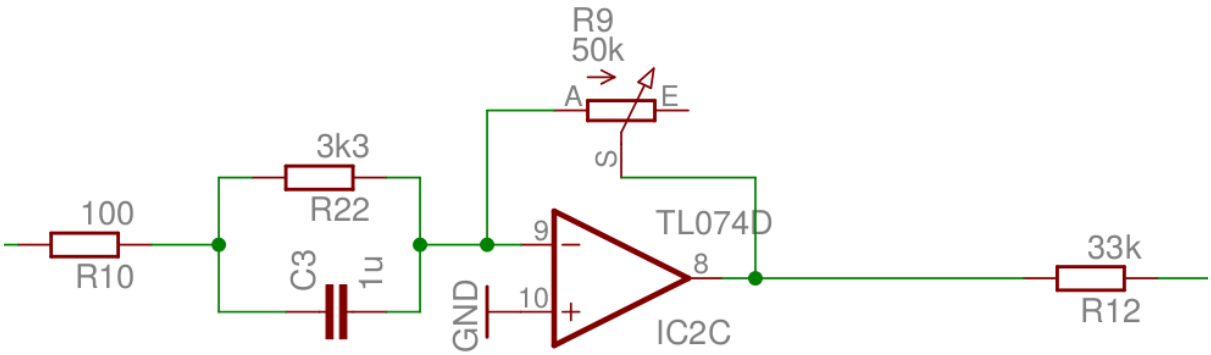


Abbildung 48: D - Anteil

Der D-Anteil wirkt ab rund 50Hz, dies wird durch R_{22} und C_3 eingestellt.

$$f_g = \frac{1}{2\pi R_{22} C_3} = 48Hz$$

Mit R_9 kann die Übertragungsfunktion vertikal verschoben werden, in Verbindung mit dem Summierer kann so die effektive Knickfrequenz verschoben werden.

6.7.6.5 D2-Anteil

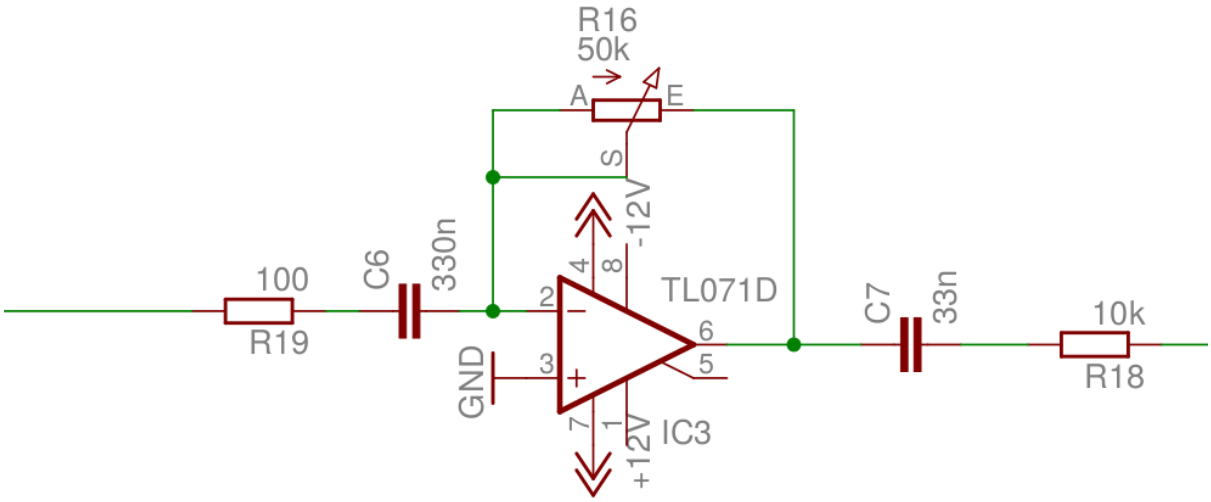


Abbildung 49: D2 - Anteil

Dieser Schaltungsteil bringt +40dB/dek bis zu rund 500Hz wo die Verstärkung in +20dB/dek übergeht, bei 5kHz endet auch diese Steigung.

$$f_{g_1} = \frac{1}{2\pi R_{18} C_7} = 482Hz \qquad f_{g_2} = \frac{1}{2\pi R_{19} C_6} = 4.82kHz$$

6.7.6.6 Summierer

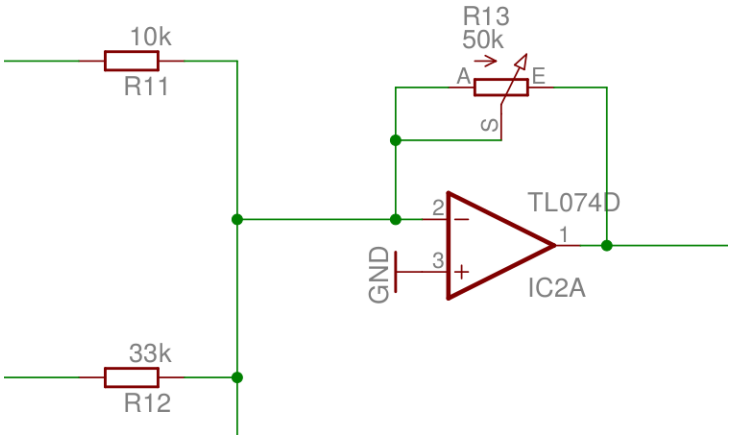


Abbildung 50: Summierer

Schlussendlich werden die einzelnen Anteile summiert, was mathematisch einer Maximum-Funktion entspricht. So können mit den Potentiometern der einzelnen Anteile die Knickfrequenzen verändert werden. Mittels R_{13} kann die Gesamtverstärkung eingestellt werden.

6.7.6.7 Regleranpassung

Der Regler wurde hauptsächlich experimentell angepasst, insbesondere bei den Prototypen wurde fast ausschließlich auf Lochrasterplatinen experimentiert. Für den engültigen Aufbau wurde die Regelstrecke an ein Bode 100 angeschlossen und die optimalen Knickpunkte ermittelt. Um nun den Regler anzupassen, werden Signale für ein Bild eingespeist und mit den Potentiometern für unverfälschte Kurvenform optimiert. So können deutlich bessere Ergebnisse erzielt werden als durch Beobachtung des Sensorsignals oder durch Anpassung anhand des Bodediagramms. Als einfachste Methode erwies sich:

1. Erhöhen des P-Anteils zur Zentrierung im Arbeitspunkt
2. Erhöhen des D2-Anteils knapp unter den Punkt, an welchem Schwingungen hörbar werden
3. Erhöhen des D1-Anteils bis Ecken ohne Überschwingen abgefahren werden
4. Wiederholen von 2. und 3. bis Schwingungen auftreten
5. D1 und D2 leicht verringern um ein stabiles System zu erhalten

Wichtig: Da die Signale addiert werden, handelt es sich dabei vor allem um die Verschiebung der Knickpunkte, die Änderungen der Verstärkung ist zweitrangig. Es zeigte sich, dass ein möglichst flacher Frequenzgang erreicht werden muss um ein graphisch ansehliches Bild zu erhalten.

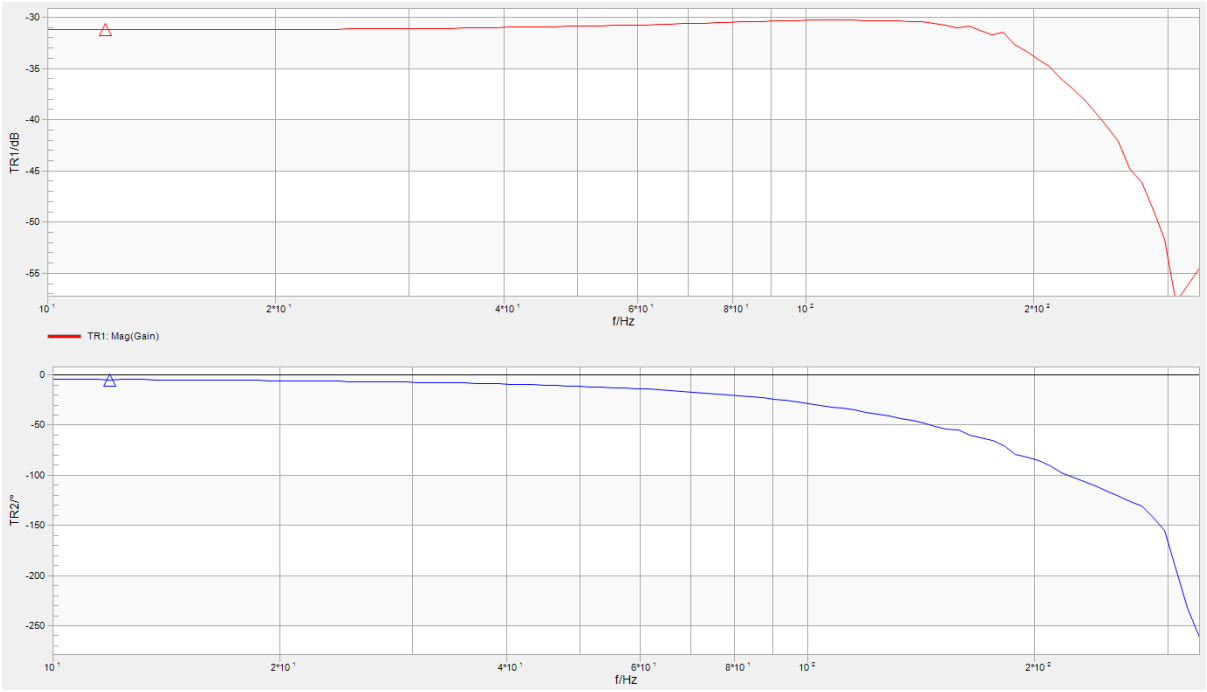


Abbildung 51: Frequenzgang eines angepassten Reglers

So kann ein -180° Punkt von 40Hz für eine P geregelte Strecke auf etwa 330Hz erhöht werden. Bei etwa 350Hz scheint eine mechanische Polstelle vorhanden zu sein und Versuche das Galvanometer schneller zu betreiben erzeugten laute Geräusche und nicht zufriedenstellende Ergebnisse.

6.7.7 Simulation des Drehmoments

Nach ausgiebiger Rechereche im Internet stellte sich heraus, dass das händische Berechnen von Kräften in magnetischen Systemen ohne ferromagnetischen Kern nur mit unverhältnismäßigem Aufwand zu bewerkstelligen ist. Wenn ein Kern vorhanden wäre, wären die benötigten Integrale deutlich einfacher, da ein homogenes Feld innerhalb eines bekannten Werkstoffs angenommen werden kann. Um diese komplexen Felder zu berechnen wird typischerweise auf rechnergestützte Simulation zurückgegriffen. Dabei kommen zwei mögliche Softwarepakete zur Simulation von physikalischen Feldern in Betracht:

- *QuickField* - Tera Analysis QuickField Student Edition[3]
- *FEMM* - Finite Element Method Magnetics[4]

Bei QuickField handelt es sich um ein kommerzielles Produkt, für welches jedoch eine eingeschränkte Student Edition erhältlich ist. FEMM hingegen ist ein Opensource Projekt mit ähnlicher Funktionalität. Die Oberfläche von QuickField ist etwas einfacher und intuitiver zu verwenden, aber in großen Teilen mit FEMM identisch. Der entscheidende Nachteil bei QuickField ist die Einschränkung der Netzdichte in der kostenlosen Student Edition, was sich in einer stark verringerten Genauigkeit äußert:

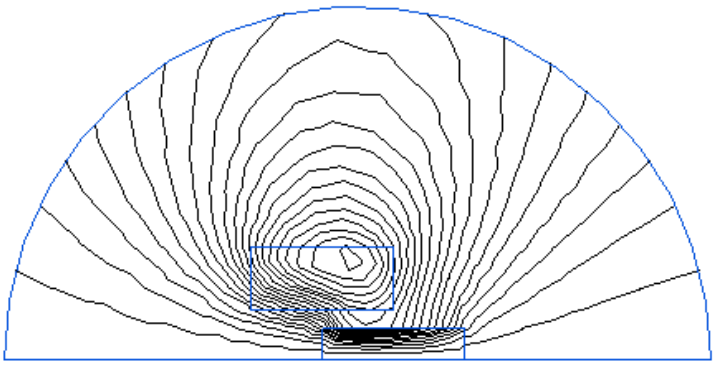


Abbildung 52: Feldlinien berechnet mit QuickField SE

[3]Tera Analysis QuickField, siehe http://quickfield.com
[4]Finite Element Method Magnetics, siehe http://femm.info

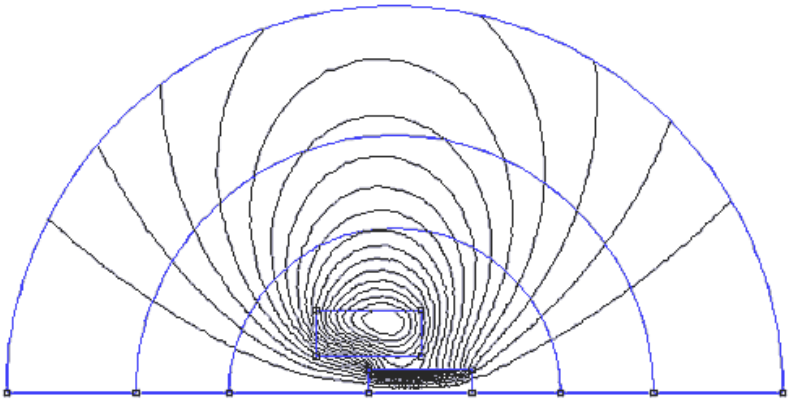


Abbildung 53: Feldlinien berechnet mit FEMM

Aus diesem Grund wurde das Simulationstool FEMM zur Berechnung des Drehmoments, welches auf den Anker wirkt, verwendet. Es kann sowohl in Verbindung mit MATLAB®[5] als mit auch dessen freien Klon GNU Octave[6] genutzt werden. In unserem Falle wurde die Simulation direkt in der Oberfläche des Programms durchgeführt. Um dreidimensionale Modelle zu simulieren muss die Symetrie ausgenutzt werden, und es wird nur der Querschnitt gezeichnet sowie die Höhe des Objekts angegeben. Im Programm wurden die ungefähren Dimensionen und verwendeten Werkstoffe unseres Aufbaus skiziiert:

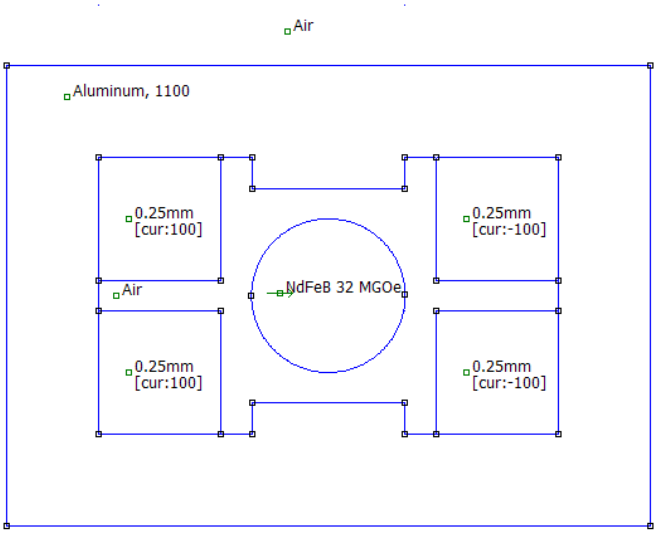


Abbildung 54: 2D - Skizze des Galvanometers

[5]Mathworks MATLAB, siehe http://www.mathworks.de/products/matlab
[6]http://www.gnu.org/software/octave

In Abbildung 54 sind auch bereits die Ströme in den Spulen eingtragen. (gekennzeichnet mit *0.25mm cur: 100*) Dies gibt den verwendeten Draht an (0.25mm Kupfer) und den Strom (100 Windungen x 1 Ampere). Auf der gegenüberliegenden Seite ist der Strom negativ eingezeichnet, der hier anders durch die gezeichnete Fläche tritt. Durch die Finite-Elemente-Methode[1] wird das Berechnungsgebiet in eine beliebig große Anzahl von Elementen unterteilt. Diese Elemente sind endlich klein und lassen sich mit einer endlichen Zahl von Parametern beschreiben. So kann die Lösung der zugrundeliegenden Differentialgleichung angenähert werden. In unserem Falle wird der Berechnungsraum auf ein endliches Gebiet beschränkt und die Fläche in ein Netz aus dreieckigen Elementen unterteilt:

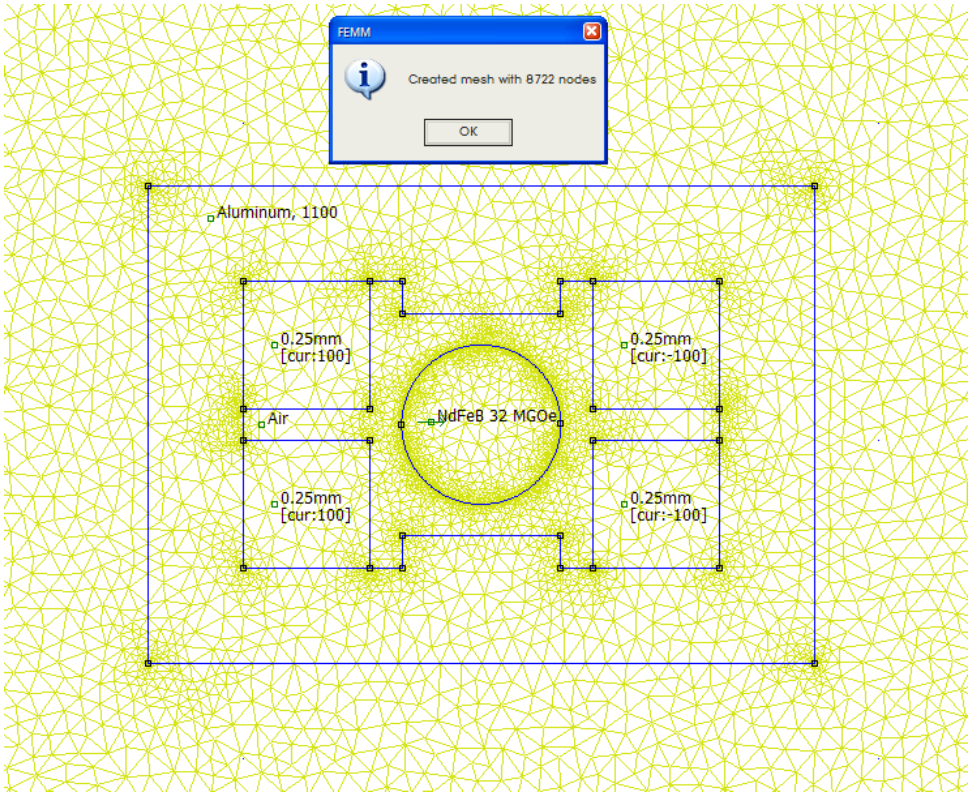


Abbildung 55: Netz zur Näherung

Dabei geht das Programm intelligent vor und unterteilt geometrisch komplexere Gebiete in kleinere Teile um eine bessere Näherung zu erhalten.

Nach Abschluss der Simulation kann der Verlauf der Flussdichte graphisch dargestellt werden:

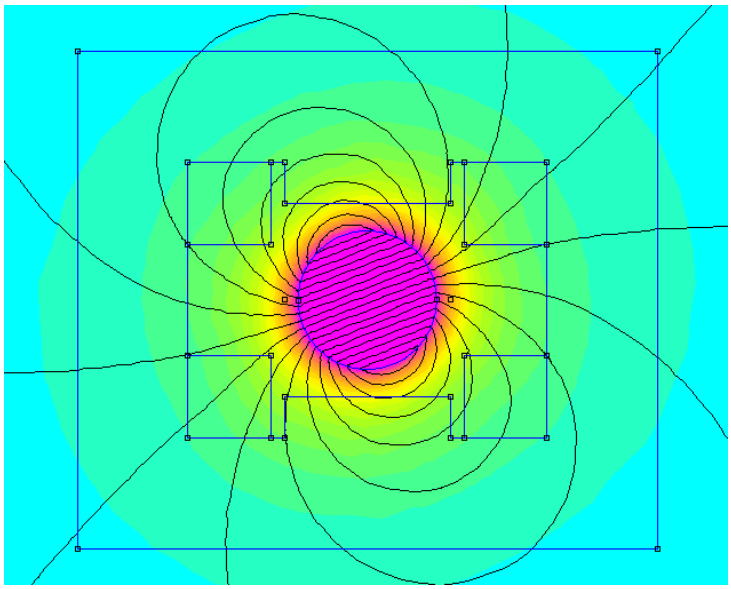


Abbildung 56: graphische Darstellung der magnetichen Flussdichte bei Stromfluss

Um nun das Drehmoment zu berechnen muss die entsprechende Fläche ausgewählt werden und die Operation über das Mathematikmenü gestartet werden:

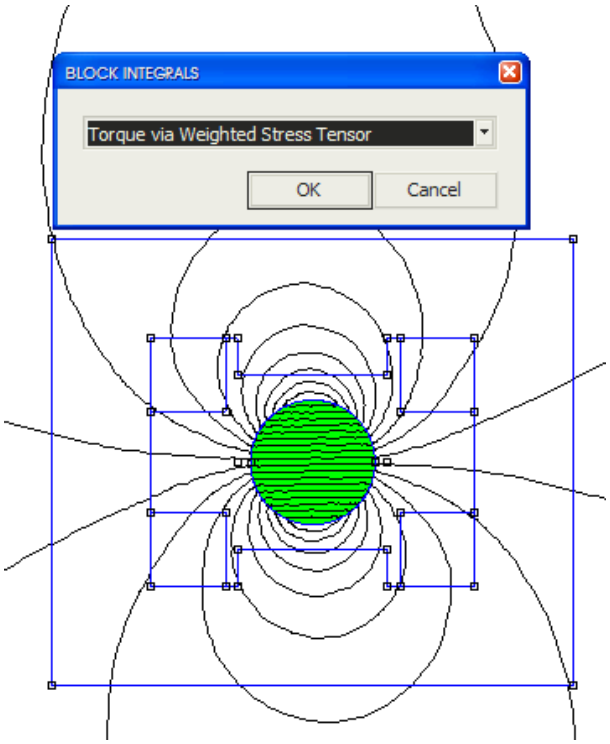


Abbildung 57: Starten der Berechnung

Nach einer kurzen Wartezeit kann das Ergebnis abgelesen werden:

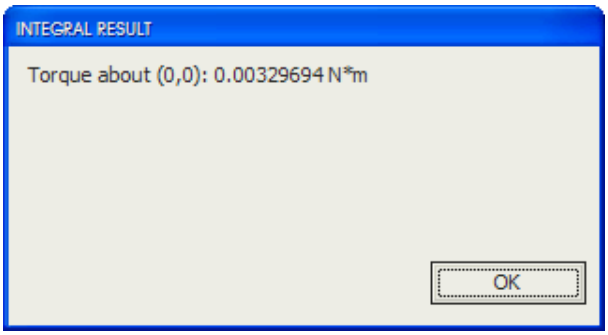


Abbildung 58: Ergebnis der Berechnung

Um nun einen Graphen über den Verlauf des Drehmoments in Abhängigkeit des Drehwinkels zu erhalten wurde ein Lua[7]-Script geschrieben um den Vorgang zu automatisieren:

Listing 13: Automatische Berechnung mit Lua

```
fid = openfile("c:/out.csv", "w")
write(fid, "Angle [deg];Torque [mNm]\n")

for i=-180,180 do
  mi_clearselected()
  mi_selectlabel(0, 0)
  mi_setblockprop("NdFeB 40 MGOe", 1, 0, "", i, 0, 1)
  mi_refreshview()
  mi_analyse()

  mo_reload()
  mo_clearblock()
  mo_seteditmode("area")
  mo_selectblock(0, 0)
  write(fid, i, ";", mo_blockintegral(22)*1000, "\n");
  print(i)
end

closefile(fid)
```

Mit der Laufvariable i wird der Drehwinkel durchgezählt, bei jedem Schritt wird:

- Vorherige Berechnungen gelöscht
- Der Winkel des Magneten geändert
- Die Berechnung durchgeführt
- Die Fläche des Magneten selektiert und das Moment berechnet
- Abspeichern des Punktes in einer *csv* Datei

[7]Eine einfache, aber vielseitige Programmiersprache für Scripting, siehe: http://lua.org

Nach der Darstellung der entstandenen csv - Datei mittels Excel ergibt sich folgende Abhängigkeit:

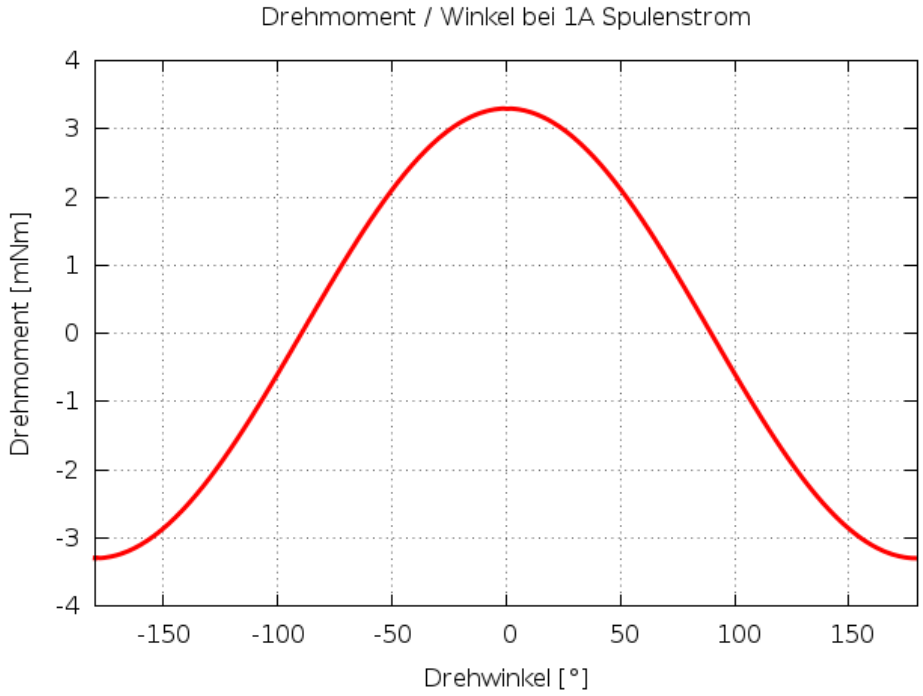


Abbildung 59: Verlauf des Drehmoments in Abhängigkeit der Ankerwinkels

Dies deckt sich sehr gut mit der theoretischen Betrachtung in Gleichung 1. Da der Großteil des mechanischen Aufbaus aus Aluminium hergestellt wurde, sind durch Imperfektionen in den Gehäusen der Galvanometern keine Einflüsse auf das magnetische Verhalten zu erwarten. Dabei muss jedoch darauf geachtet werden, dass auch Verbindungselement wie Schrauben und Achsen aus nicht-magnetischem Material aufgebaut sind.

6.8 Ablenkspiegel

Mit zwei Spiegeln wird eine freie Ablenkung des Laserpunkts auf einer ebenen Fläche erzielt. Dabei wird je ein Spiegel für die X- und Y-Ablenkung benötigt. Der einfallende Laserstrahl wird vom ersten Spiegel nach oben auf einen weiteren Spiegel gelenkt, dabei wird der Austrittswinkel so verändert, dass die gewünschte X-Ablenkung erzielt wird:

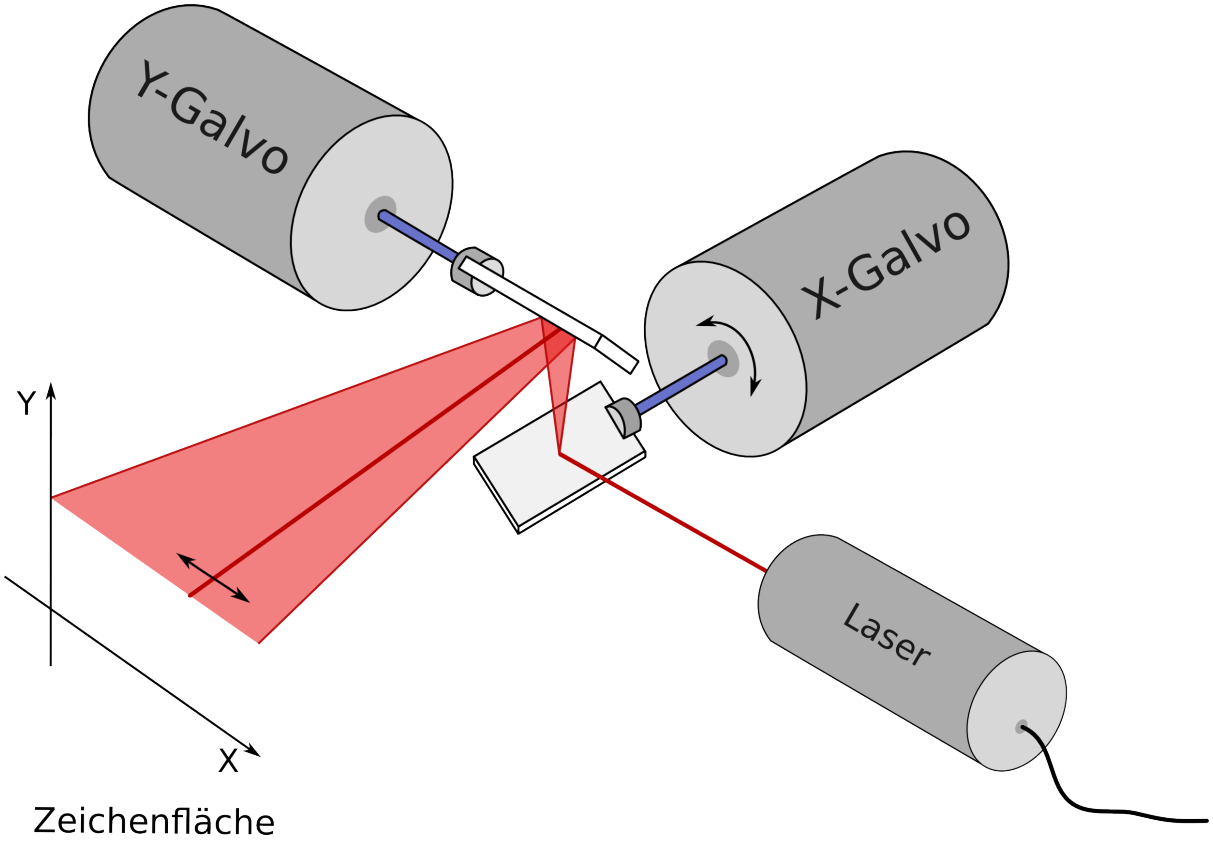


Abbildung 60: Ablenkung in X-Richtung

So kann der Laser auf einer Linie in X-Richtung abgelenkt werden. Dabei muss darauf geachtet werden, dass die Grenzen des oberen Spiegels nicht überschritten werden. Der obere Spiegel lenkt den Laser wieder um 90° und erzeugt zusätzlich die Ablenkung in Y-Richtung:

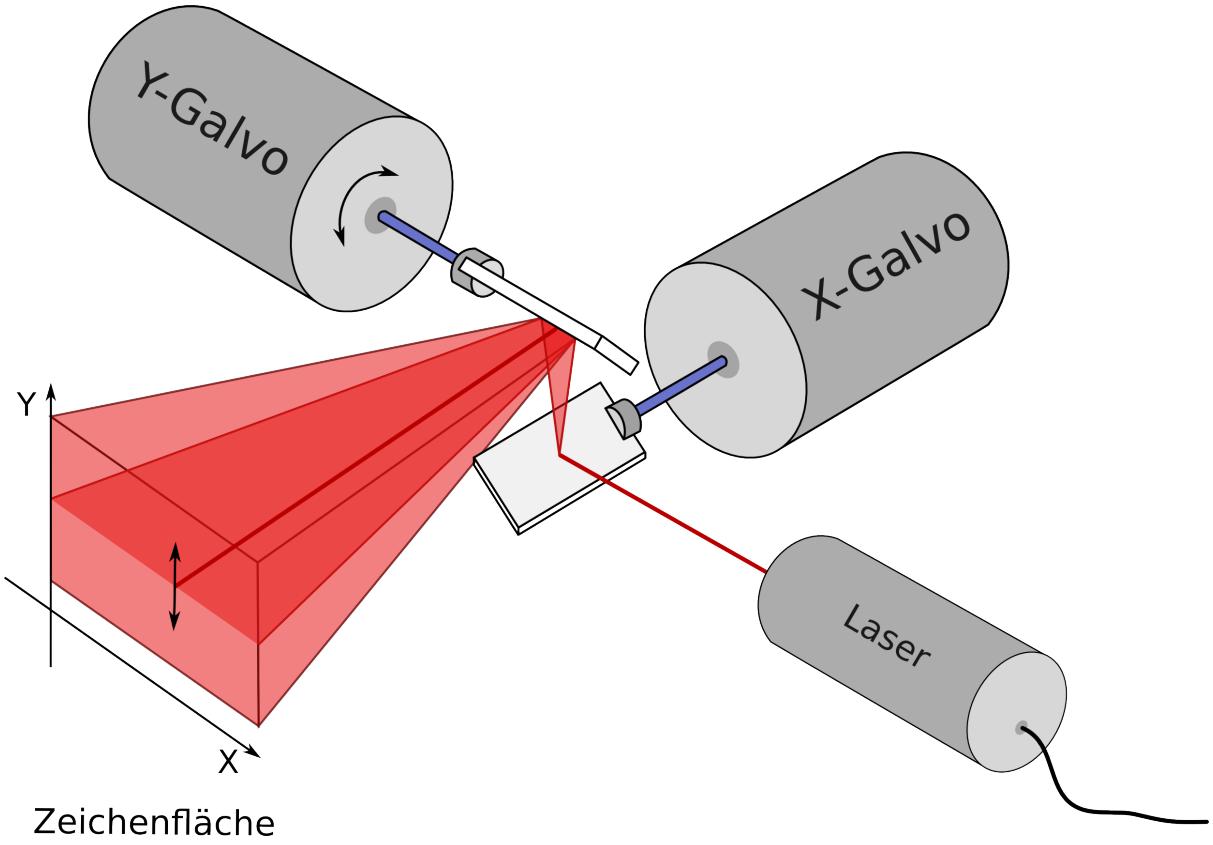


Abbildung 61: Ablenkung in Y-Richtung

Mit beiden Spiegeln kann so der Strahl auf jeden Punkt auf der gesamten Zeichenfläche gerichtet werden.

6.8.1 Projektion der Koordinaten

Dazu müssen die kartesischen Koordinaten der Bilddaten in Winkel für die Galvanometer umgesetzt werden. Der eingestellte Winkel ergibt den doppelten Winkel in der Ablenkung, wie folgende Überlegung zeigt:

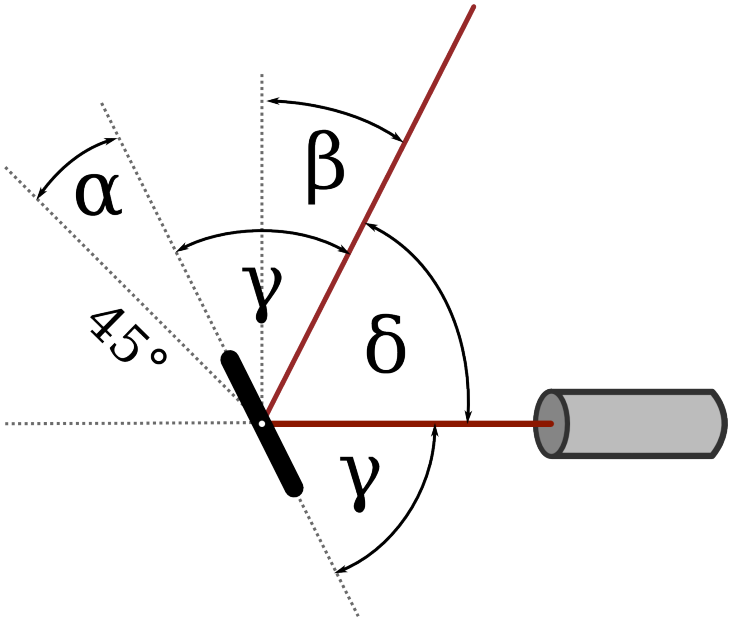


Abbildung 62: Winkel bei der Spiegelablenkung

$$\begin{aligned} \gamma &= 45^\circ + \alpha \\ \delta &= 180^\circ - 2 \cdot \gamma \\ \beta &= 90^\circ - \delta \end{aligned}$$

$$\begin{aligned} \beta &= 90^\circ - (180^\circ - 2 \cdot \gamma) \\ \beta &= -90^\circ + 2 \cdot \gamma = -90^\circ + 2 \cdot (45^\circ + \alpha) \\ \beta &= 2 \cdot \alpha \end{aligned}$$

Abbildung 63 zeigt die Umlenkung des Laserstrahls und die zu berücksichtigenden Winkel:

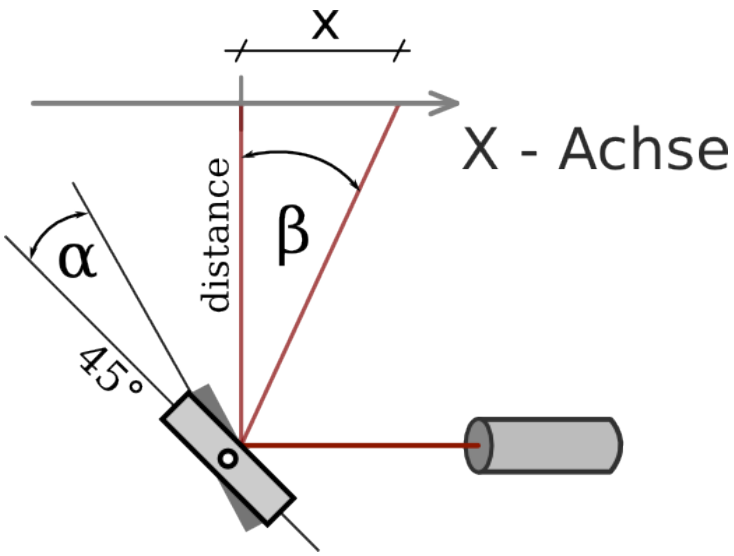


Abbildung 63: Ablenkung auf eine Fläche

Die kartesichen Koordinaten liegen im Bereich $[0.0-1.0]$. Die Werte für den 12 Bit DAC liegen im Bereich $[0-4095]$ und einem Mittenwert von 2047. Um die korrekt entzerrten Werte auszugeben muss der Zusammenhang zwischen Stellwinkel und X - Koordinate berücksichtigt werden. Dieser kann durch folgende Formeln beschrieben werden:

$$
\begin{aligned}
\frac{x}{distance} &= \tan\beta \\
\beta &= \arctan\left(\frac{x}{distance}\right) \\
\frac{x_{max}}{distance} &= scale = \tan(\beta_{max}) \\
\beta &= \arctan\left(\left(2 \cdot x_{digital} - 1\right) \cdot scale\right) \\
\alpha &= \frac{1}{2}\beta \\
\alpha_{digital} &= 2047 \cdot \left(1 + \frac{\alpha}{\alpha_{max}}\right) \\
\alpha_{digital} &= 2047 \cdot \left(1 + \frac{\frac{1}{2} \cdot \arctan\left(\left(2 \cdot x_{digital} - 1\right) \cdot scale\right)}{\frac{1}{2} \cdot \beta_{max}}\right) \\
\alpha_{digital} &= 2047 \cdot \left(1 + \frac{\arctan\left(\left(2 \cdot x_{digital} - 1\right) \cdot scale\right)}{\beta_{max}}\right)
\end{aligned}
$$

Für kleine Werte ist die Verzerrung jedoch sehr gering und so könnte im Falle von Performanceproblemen auf eine ressourcenschonendere lineare Approximation ausgewichen werden.

6.8.2 Fehler durch Spiegelpositionierung

Die Spiegel wurden in der Mitte der Achse positioniert um ein günstigeres Trägheitsmoment für die Achse zu erhalten. Dies hat jedoch zur Folge, dass sich der Auftrittspunkt des Laserstrahls auf den Spiegel minimal auf der Achse des Strahls verschiebt:

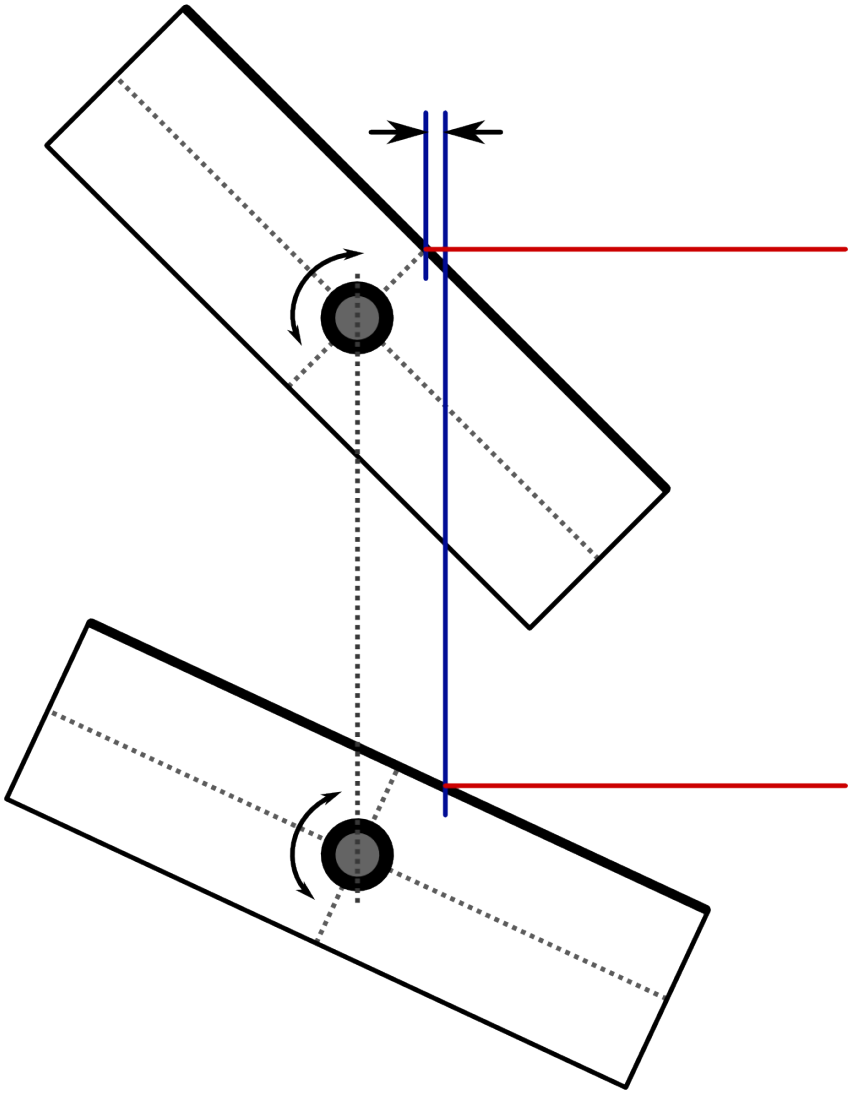


Abbildung 64: Auftrittspunkt des Lasers

In einer Detailansicht zeigt sich, dass sich der Winkel des austretenden Strahls nicht verändert, allerdings ergibt sich eine Verschiebung auf der Zeichenfläche. Diese Verschiebung ist unabhängig vom Abstand der Zeichenfläche zum Gerät:

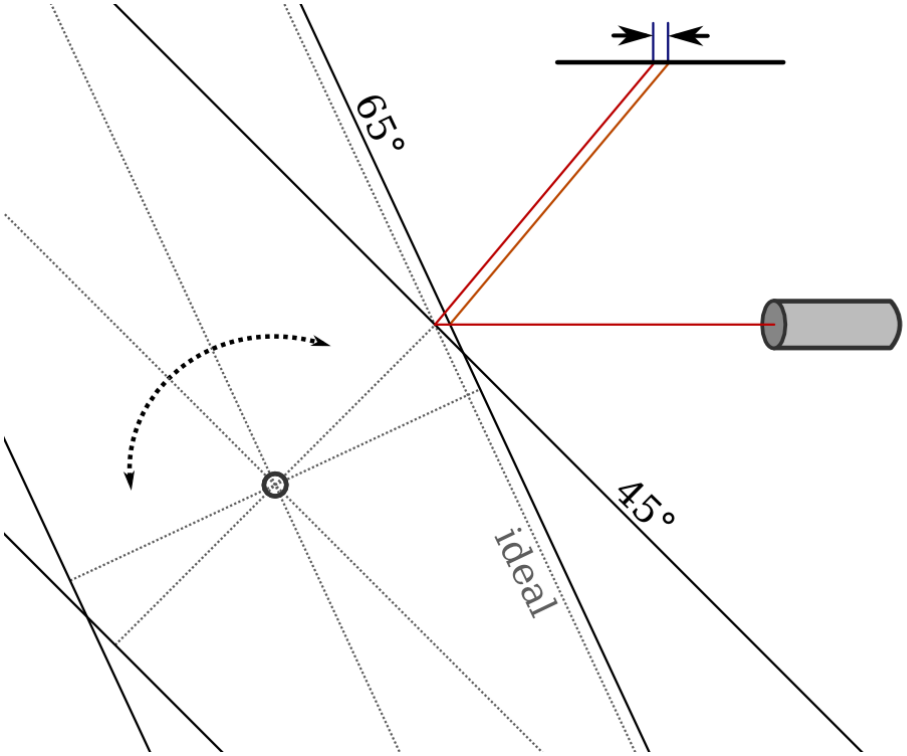


Abbildung 65: Positionsfehler durch Spiegel

Allerdings ist die Verschiebung unsymetrisch und begrenzt besonders in eine Kipprichtung (wie in Abbildung 64 dargestellt) den nutzbaren Winkel. Im folgenden wird der Fehler mathematisch beschrieben.

Die Verschiebung hängt direkt vom Winkel des Spiegels ab, wie in dieser Ansicht dargestellt wird:

d ... Halbe Dicke des Spiegels
α ... Kippwinkel des Spiegels
a ... Fehler

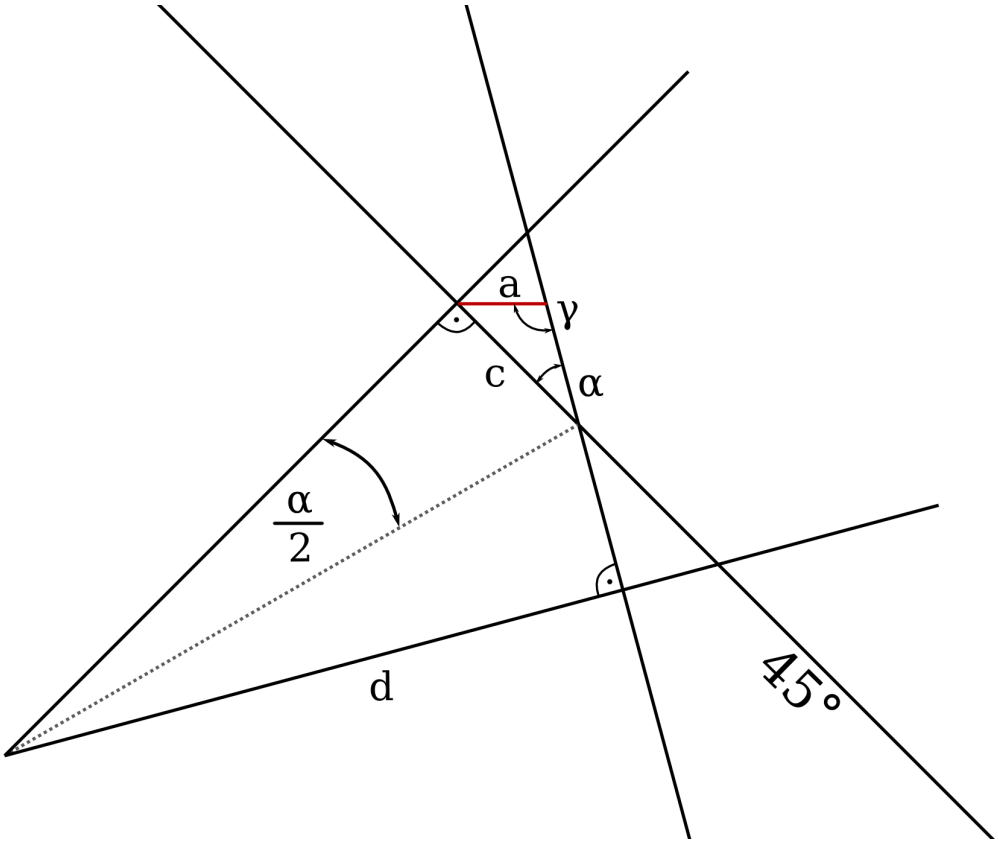


Abbildung 66: Winkel im Detail

$$\begin{aligned} \frac{c}{d} &= \tan\left(\frac{\alpha}{2}\right) \\ \gamma &= 180^\circ - 45^\circ - \alpha = 135^\circ - \alpha \\ \frac{a}{\sin(\alpha)} &= \frac{c}{\sin(\gamma)} \\ a(\alpha) &= d \cdot \frac{\sin(\alpha)}{\sin(135^\circ - \alpha)} \cdot \tan\left(\frac{\alpha}{2}\right) \end{aligned}$$

Es ergibt sich folgende Abweichung in Abhängigkeit des Winkels:

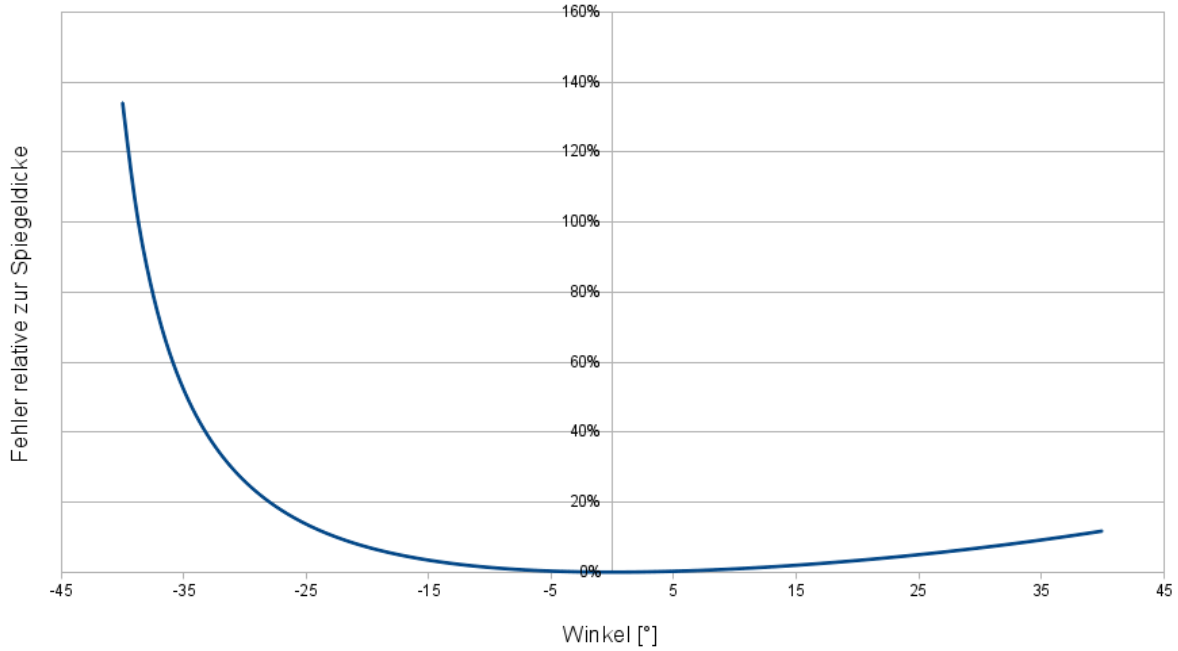


Abbildung 67: Graph der Funktion $a(\alpha)$

Es zeigt sich, dass die Verschiebung zwar korrigiert werden kann, doch der rechentechnische Aufwand, alle Koordinaten in Echtzeit anzupassen recht hoch ist. Da sich nur eine Abweichung im Sub-Millimeterbereich ergibt und da der tatsächlich verwendete Winkelbereich sehr klein ist (ca. $\pm 20°$) nehmen wir diese Verzerrung an den Bildrändern in Kauf. Das Projektziel von ILPS ist die Darstellung von Graphiken für Betrachter in einigem Abstand, eine Korrektur wäre nur erforderlich, wenn der Laserpunkt für Vermessungszwecke oder ähnliches zum Einsatz käme.

6.9 Power-Modul

6.9.1 Funktionsprinzip

Das Power-Modul versorgt alle Komponenten mit den benötigten Spannungen. Es wird mit ±12V versorgt und wandelt mithilfe von Step-Down Wandler die Spannungen in die benötigten Spannungen um.

6.9.2 Abschätzung

Benötigte Spannungen und Ströme:

- ± 12V
  - Galvanometer (1-2A)
- +5V
  - Raspberry Pi (700mA)
  - Externe USB Schnittstelle (500mA)
- +3.8V
  - Tablet (1.5A)
- +3.3V
  - Hauptboard (100mA)
  - Lasermodul (100mA)
  - sonstige Versorgungen (100mA)

6.9.3 Aufteilung

Das Power-Modul besteht aus zwei Teilen. Zum ersten Teil besteht es aus einem gekauften Netzteil von Reichelt, welches ±12V Spannung und einen maximalen Strom von 6A liefert. Der zweite Teil besteht aus einer Step-Down Schaltung, mit der die benötigten Spannungen, 5V, 3.8V und 3.3V, erzeugt werden kann. Für diesen Einsatzzweck wird eine Step-Down Regelung verwendet, da die Verlustleistung einem Linearregler gegenüber einfach viel geringer ist, trotz höherem Schaltungsaufwand.

6.9.4 Planung von Schaltplan und Layout

- Grundschaltung aus Datenblatt von LM2576

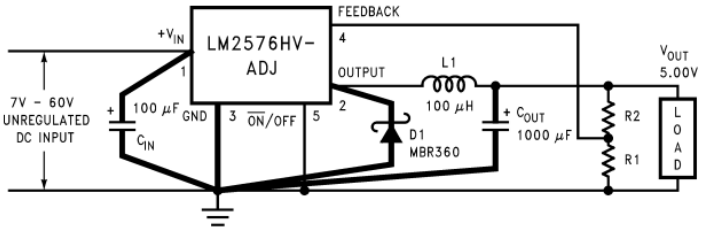


Abbildung 68: Grundschaltung aus Datenblatt von LM2576

- Entwickelte Schaltung für alle benötigten Spannungen

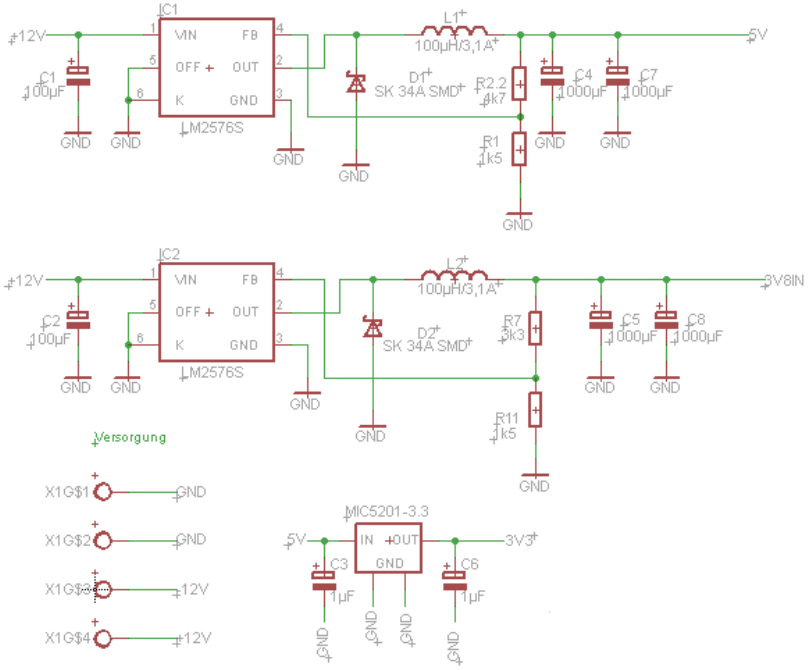


Abbildung 69: Schaltungsteil für die verschiedenen Spannungen

- Power-Modul Verbindung mit dem Mainboard

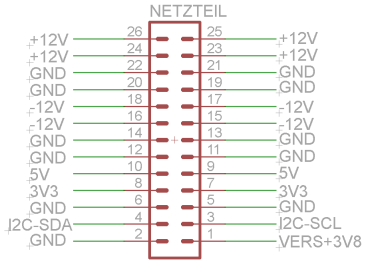


Abbildung 70: Verbindungsstecker Mainboard Power-Modul

Für eine Temperaturüberwachung wird ein Temperatursensor zwischen den beiden LM2576 angebracht. Die Daten werden zurück an den STM32 gesendet. Für die Verbindung des Power-Moduls mit dem Mainboard wird ein 26-poliger Stecker verwendet.

6.9.5 Berechnungen

- Widerstandsberechnung zum Einstellen der Spannung

$$R_2 = R_1 \cdot \left(\frac{V_{\text{OUT}}}{V_{\text{REF}}} - 1 \right)$$

  Deshalb werden folgende Werte für R_2 verwendet:

  – bei 5V: 4.6kΩ

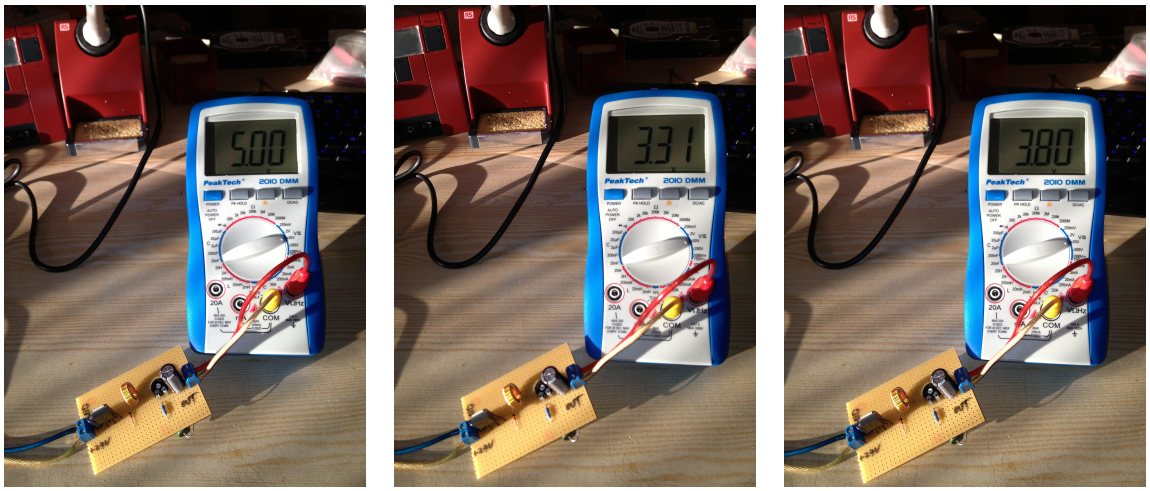
  – bei 3.8V: 2.547kΩ

  – bei 3.3V: 3.13kΩ

- Berechnung der Kühlung des LM2576
  Der LM2576 hat eine Effizienz von circa 80% und so müssen max. 2W Verlustleistung abgeführt werden. Im DPAK-Gehäuse auf der Platine aufgelötet wird mit rund 5cm^2 Kupferfläche eine Wärmeableitung von 37° C/W erziehlt. Da im Mittel deutlich weniger Verlustleistung anfällt und die effektive Kuperfäche größer ist, erwärmt sich der Schaltregler im Betrieb nur auf ca. 45° C.

6.9.6 Testaufbau

Der Testaufbau beweist, dass die Versorgung auf 10mV genau eingestellt werden kann.



(a) 5V (b) 3V3 (c) 3V8

Abbildung 71: Spannungsmessung Testaufbau

Es wurde nur ein Testboard aufgebaut und mithilfe eines Potentiometer kann hier die Spannung eingestellt werden. Somit ist es möglich, genau an die gewollte Spannung zu kommen. Am Eingang der Testplatine werden mithilfe eines Labornetzgerätes 12V gespeist. Das Testboard ist für 2A ausgelegt, deshalb kann mit diesem Board nicht die volle Last verwendet werden. Ein Test ist nur mit aktiver Last möglich. Um eine stabile Ausgangsspannung zu erhalten, muss ein Widerstand angeschlossen werden. Es zeigte sich, das die ursprüngliche Dimensionierung mit ±12V zu hoch angesetzt war, deshalb wird ILPS nun mit ±7V betrieben um die Leistungs-OPs nicht unnötigt zu belasten.

7 Software

7.1 Grundprinzip

Bei der Software wurde sowohl auf der Serverseite als auch auf der Clientseite die Programmiersprache Javascript gewählt. Der Server wird mit NodeJS programmiert. Dies ermöglicht es, dass Daten individuell zwischen Client und Server ausgetauscht werden können. Auf dem Server werden anschließend die Daten in einer Datenbank gespeichert und je nach Bedarf weiterverarbeitet und zum Mainboard weitergesendet. Zusätzlich ermöglicht es NodeJS, dass die von einem Client empfangenen Daten auch direkt vom Server zu allen anderen Clients weitergesendet werden. Somit kann jeder Client in Echtzeit sehen, was ein andere Client gerade zeichnet.

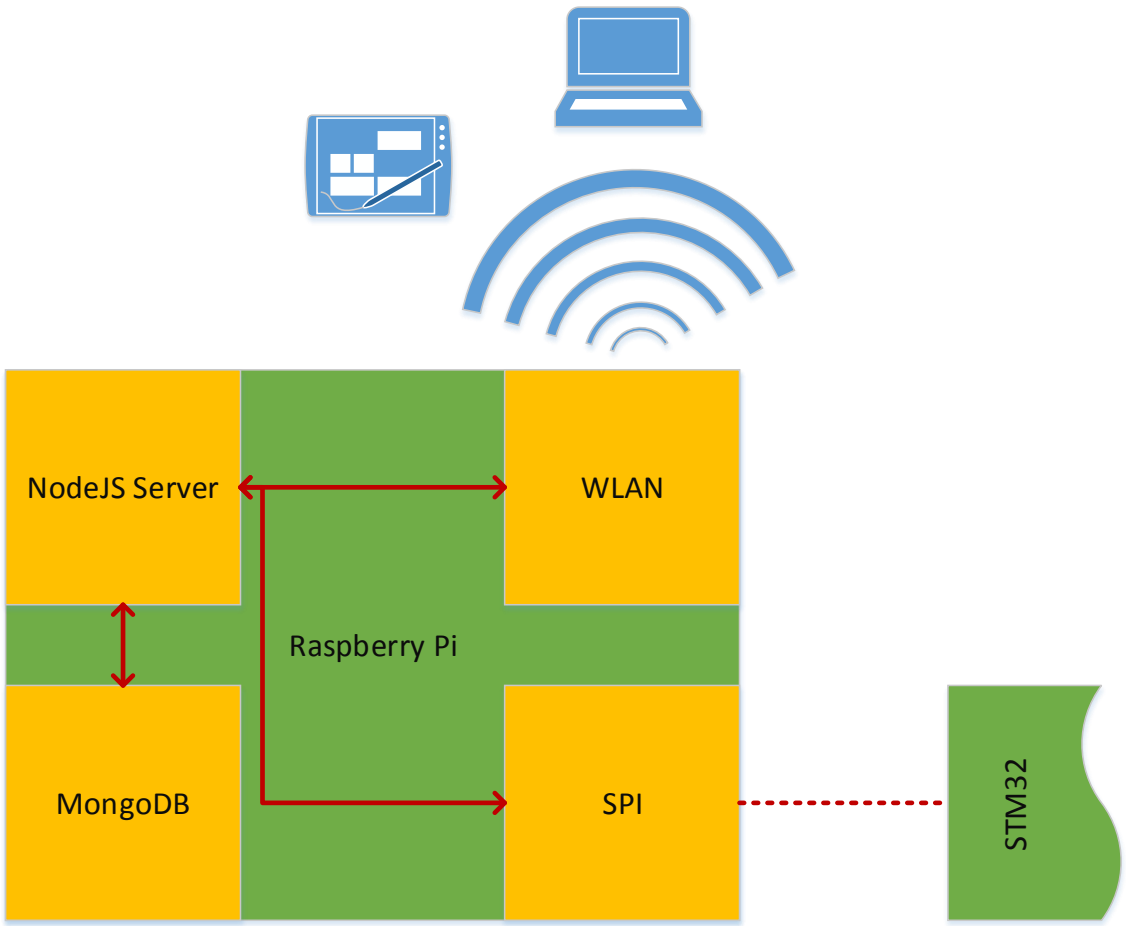


Abbildung 72: Blockschaltbild mit den Softwarekomponenten

7.2 NodeJS und Socket.IO

Bei der Konzeptüberlegung waren folgende Punkte wichtig:

- wenig Ressourcenverbrauch
- asynchrone Datenverarbeitung
- Kommunikation zwischen Server und Client

Durch diese Kriterien fiel die Wahl auf NodeJS[8] mit der Websocketbibliothek Socket.IO. Durch die Architektur von NodeJS ist es möglich eine Software ohne hohe Leistungsanforderungen zu entwickeln. Zusätzlich ist es ein ereignisgesteuertes, nicht blockierendes I/O Modul, welches perfekt für Echtzeitapplikationen mit Netzwerkanwendungen zum Austausch von Daten ist. Die Websocketbibliothek SocketIO ermöglicht eine Echtzeitkommunikation zwischen einzelnen Clients und dem Server.

7.2.1 NodeJS

Basierend auf der JavaScript - Laufzeitumgebung V8[9], welche ursprünglich für den Chrome-Browser entwickelt wurde, bietet NodeJS eine resourcensparende Architektur, die eine besonders große Anzahl gleichzeitig bestehender Netzwerkverbindungen ermöglicht.
Das resultierende NodeJS Programm ist plattformunabhängig, da es von einem beliebigen Gerät mit einem Browser aufgerufen werden kann. Über ein TCP/IP Protokoll ermöglicht NodeJS eine Übertragung von Daten und Befehlen. Diese Kommunikation zwischen Client und Server geschieht wie die anderen Routinen von NodeJS in Echtzeit.
Durch NodeJS kann vom Frontend bis zum Backend die gleiche Scriptsprache und somit auch Programmierumgebung verwendet werden. Das Frontend wird nach dem HTML5 Standard und in der Scriptsprache JavaScript programmiert. Das Backend besteht aus reinem JavaScript.

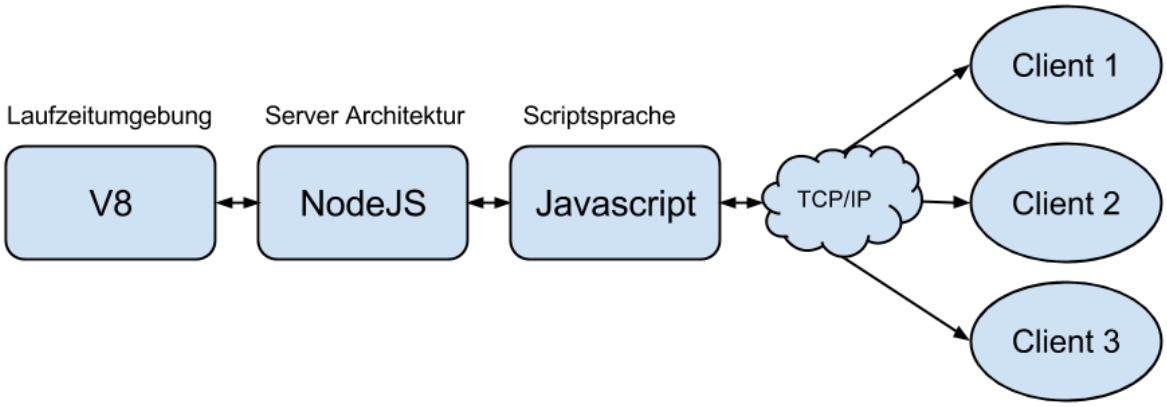


Abbildung 73: NodeJS Schema

In der Grafik (Abbildung 73) ist ersichtlich wie die Architektur aufgebaut ist. Den Kern bildet die Laufzeitumgebung V8, welche mit der Server Architektur NodeJS kommuniziert. NodeJS

[8] http://de.wikipedia.org/wiki/NodeJS
[9] http://code.google.com/p/v8/

wiederum interpretiert die Scriptsprache JavaScript. Das resultierende Programm kommuniziert anschließend durch eine TCP/IP Verbindung mit den Clients.

7.2.2 Ereignisgesteuert

Mit NodeJS ist es möglich ereignisgesteuerte Programme zu entwickeln. Diese Ereignisse können sowohl von außen mit z.B. Benutzereingaben oder Sensorwerten, als auch vom System selbst ausgelöst werden. Somit hat die ereignisgesteuerte Architektur selbst, kaum Kontrolle darüber, wann Daten verarbeitet oder ausgetauscht werden.
Das einfachste Beispiel für ereignisgesteuerte Elemente ist die graphische Oberfläche, sobald der Benutzer auf einen Button klickt, wird ein Ereignis aufgerufen und die dafür zuständigen Codezeilen abgearbeitet.
Für jedes Ereignis gibt es immer einen Ereignis-Produzent (event generator) und einen Ereignis-Träger (event channel). Der Produzent ist derjenige, der den Status eines Objektes überwacht, sobald dieses Objekt den Status ändert wird ein Ereignis erzeugt und dem Träger zugesendet. In unserem Fall ist der Client meistens in der Rolle des Ereignis-Produzenten. Der Ereignis-Träger ist das Medium, welches das Ereignis empfängt und bearbeitet. Bei diesem Projekt ist der Ereignis-Träger im Normalfall der Server, welcher vom Client Ereignisse empfängt und verarbeitet.[7]

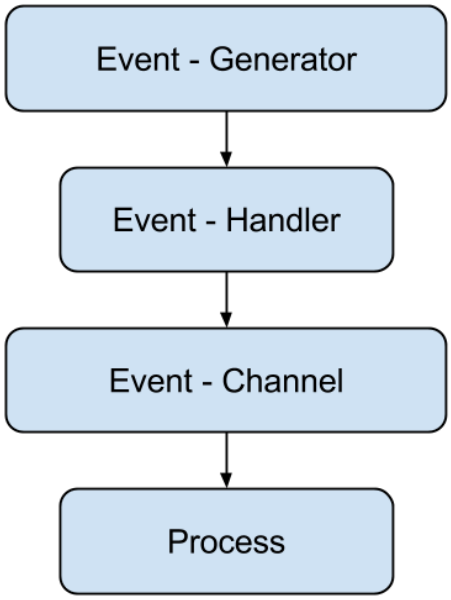


Abbildung 74: Ereignisgesteuerter Ablauf

In Abbildung 74 ist ersichtlich, wie ein ereignisgesteuertes Programm arbeitet. Als Erstes wird vom Ereignis-Produzent ein Ereignis erzeugt. Dieses wird durch den Ereignis-Handler zum Ereignis-Träger weitergeleitet. Das Medium und somit der Ereignis-Träger empfängt das erzeugte Ereignis und ruft einen Prozess auf um die Aktionen des Ereignis auszuführen.

7.2.3 Non-Blocking I/O

NodeJS besteht auch aus einem Non-Blocking oder asynchronen I/O Modul. Non-Blocking I/O eignet sich ausgezeichnet für Anwendungen, die gleichzeitig mehrere hundert TCP-Verbindungen verarbeiten und primär Netzwerk-Funktionalitäten realisieren.[11]
Durch das asynchrone Verhalten, muss die Software nicht auf die I/O Geräte warten, sondern kann währenddessen andere Aufgaben übernehmen. Gleichzeitig ist es ressourcensparend, da das Programm zwischen Ereignissen nichts zu tun hat und somit in einen "Idle" Modus wechselt, in welchem das Programm in einem schlafähnlichen Zustand ist und keine weiteren Ressourcen mehr benötigt, bis das nächste Ereignis auftritt.

7.2.4 SocketIO

SocketIO ist eine JavaScript Bibliothek für Echtzeit-Webapplikationen. Es besteht aus zwei Teilen: einer clientseitigen Bibliothek, welche im Browser verwendet wird und einer serverseitigen Bibliothek für NodeJS. Beide Komponenten haben näherungsweise die gleiche API. Wie auch NodeJS ist SocketIO ereignisgesteuert.
SocketIO verwendet primär das WebSocket[10] Protokoll. Falls das WebSocket Protokoll nicht zur Verfügung steht, kann es auf andere Methoden wie Adobe Flash[11] Sockets, JSONP polling[12] und AJAX long polling[13] zurückgreifen. Für die Standardfunktionen würde die WebSocket Bibliothek reichen. Allerdings ist es mit SocketIO möglich, Multicasts bzw. Broadcasts zu senden. Zusätzlich bietet SocketIO die Möglichkeit individuell Daten für jeden Client abzuspeichern.

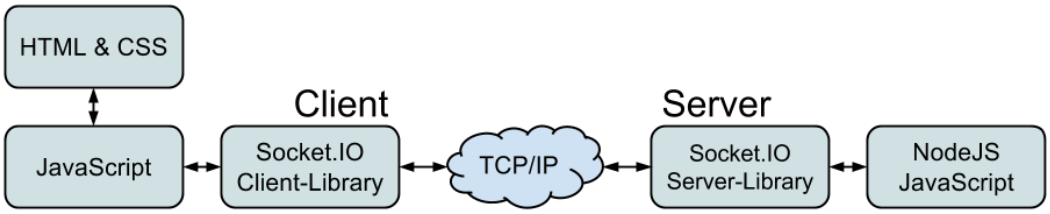


Abbildung 75: Socket.IO Schema

In Abbildung 75 ist ersichtlich wie die 2 Teile von der SocketIO Bibliothek aufgeteilt sind. Der Client liest über die Bibliothek Daten vom Server ein und verwendet diese anschließend im JavaScript für weitere Berechnungen bzw. Ausgaben. Auf der Serverseite übergibt NodeJS die Daten der SocketIO Bibliothek. Diese werden anschließend über die verfügbare Transportmöglichkeit zum Client gesendet.

7.2.4.1 Socket Kommunikation

Wie bereits erwähnt, ist es mit SocketIO möglich, jeden Client individuell anzusprechen. SocketIO speichert für jeden verbundenen Client einen eigenen Socket. Der Socket beinhaltet unter anderem eine unique ID, Datum / Uhrzeit vom Zeitpunkt des Erstellens, IP Adresse und Port

[10] http://en.wikipedia.org/wiki/WebSocket
[11] http://en.wikipedia.org/wiki/Adobe_Flash
[12] http://en.wikipedia.org/wiki/JSONP
[13] http://en.wikipedia.org/wiki/Comet_%28programming%29#Ajax_with_long_polling

des Clients, Heartbeat enabled/disabled, Heartbeat timeout und Heartbeat Intervall. Ein Heartbeat ist ein Mechanismus die periodisch in einem bestimmten Intervall aufgerufen wird und überprüft, ob die Sockets bzw. Clients noch verfügbar sind.

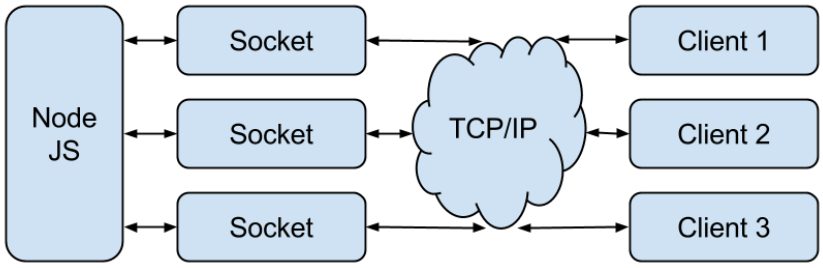


Abbildung 76: Kommunikation Socket<–>Client

In Abbildung 76 ist die Kommunikation zwischen Socket und Client dargestellt. Jeder Client kommuniziert mit dem vom Server zugewiesenen Socket.
Die gezielte Kommunikation mit einem Client ist ressourcensparend und zugleich eine Entlastung für die Übertragungsleitung. Meistens werden nur wenige Clients angesprochen. In solch einem Fall werden die Daten nur gezielt an die Clients versendet, welche sie angefordert haben bzw. benötigen.

7.2.4.2 Installation

Für die Installation von NodeJS wird die aktuelle Version von NodeJS benötigt. Je nach Betriebssystem muss die 32Bit[14] oder die 64Bit[15] Version installiert werden.
Installationsschritte:

1. Ein neues Verzeichnis erstellen und dieses der Pathvariable des Systems hinzufügen.

2. Die aktuelle Version der `node.exe` in dieses Verzeichnis kopieren

3. Die aktuelle Version von npm herunterladen und in diesem Verzeichnis entpacken.

Anschließend kann mit der Konsole ein NodeJS Programm mit dem Befehl `node scriptname.js` gestartet werden.

[14] http://nodejs.org/dist/latest/node.exe
[15] http://nodejs.org/dist/latest/x64/node.exe

7.3 npm Package Manager

npm[16] ist ein offizieller Paketmanager für NodeJS. Er ist zur Gänze in JavaScript programmiert und läuft auf der NodeJS Plattform.
Seit der NodeJS Version 0.6.3 ist npm ein fixer Bestandteil von NodeJS und wird automatisch installiert. Der Paketmanager wird mithilfe einer Konsole bedient. Hauptaufgabe des Paketmanagers ist das Verwalten der Abhängigkeiten einer Applikation. Zusätzlich erlaubt npm dem Benutzer auch das Installieren von NodeJS Modulen, welche in der npm Bibliothek vorhanden sind.[9]
Mithilfe von npm ist es sehr einfach neue Module zu aktualisieren bzw. installieren. Zuerst muss mit der Konsole mithilfe des `cd` Befehls in den Ordner mit der NodeJS Applikation gewechselt werden. Anschließend kann mit `npm install [modulname]` ein neues Modul installiert werden. Um ein bereits vorhandenes Modul zu aktualisieren, muss der Befehl `npm update [modulname]` angewendet werden.

7.3.1 Package.json

Der npm Package Manager ist auch in der Lage, die `package.json` Datei im Stammordner der Applikation zu öffnen. Package.json ist eine Textdatei im JSON Format. Diese Datei enthält alle Abhängigkeiten mit den benötigten Versionen. Dies erleichtert anschließend das Installieren aller Abhängigkeiten enorm.
Der Aufbau einer textttpackage.json Datei ist immer gleich und sieht folgendermaßen aus:

Listing 14: package.json

```
1  {
2      "name":"ILPS",
3      "version":"0.0.1",
4      "private":true,
5      "dependencies":{
6          "socket.io":"0.9.16",
7          "express":"3.3.8",
8          "mongodb":"*",
9          "jquery":"*",
10         "nodetime":"*",
11         "node-static":"*"
12     }
13 }
```

In Zeile 3 wird die derzeitige Version angegeben. Bei kleinen Änderungen wird meistens nur die letzte Zahl erhöht. Bei großen Änderungen, wie komplett neuen Funktionen werden die ersten zwei Zahlen erhöht.
In Zeile 4 wird das Privat-Flag gesetzt, welches verhindert, dass das Paket veröffentlicht wird.
Von Zeile 6 bis 11 werden die dependencies, sprich Abhängigkeiten, mit den benötigten Versionen angegeben.
Damit eine textttpackage.json Datei verwendet werden kann, muss mindestens der Name, die Version und eine Abhängigkeit angegeben werden.
Bei Verwendung einer textttpackage.json Datei reicht das Ausführen des Befehls `npm install`. Dadurch werden alle Abhängigkeiten mit den geforderten Versionen installiert.

[16] https://www.npmjs.org/

7.4 MongoDB

Beim Datenbanksystem ist die Wahl auf eine der derzeit verbreitetsten NoSQL[17] - Datenbank, MongoDB gefallen.
MongoDB ist eine hochperformante, schema-freie, dokumentenorientierte Open Source Datenbank, die in der Programmiersprache C++ geschrieben ist. Da MongoDB dokumentenorientiert ist, ist das Datenbanksystem in der Lage, große Sammlungen von JSON-ähnlichen Dokumenten zu verwalten. Somit können die Daten in komplexen Hierarchien verschaltet werden, bleiben jedoch trotzdem immer abfragbar.[10]
Der ausschlaggebende Punkt für MongoDB war, dass es ein hochperformantes System bietet, welches sehr kurze Zugriffszeiten besitzt und eine schnelle Datenübertragung zwischen Datenbank und Serveranwendung ermöglicht.

7.4.1 NoSQL Datenbank

MongoDB ist neben Cassandra und Couchbase eine der populärsten NoSQL Datenbanken.
SQL Datenbanken leiden im Normalfall an Leistungsproblemen bei datenintensiven Applikationen wie Webseiten mit hohen Lastaufkommen oder Streamingapplikationen. Auf viele kleine oder wenige große Zugriffe können SQL Datenbanken optimiert und effizient eingesetzt werden. Sehr kritisch werden SQL Datenbanken, wenn gleichzeitig große Datenänderungen vorgenommen werden.
Eine NoSQL Datenbank kann dagegen sehr gut mit vielen Schreib-/Leseanfragen umgehen. Auch bei häufigen großen Datenänderungen sind bei einer NoSQL Datenbank kaum Leistungseinbrüche zu erwarten.[8]

7.4.2 Datenbankschema und Speichermanagement

MongoDB ist beim Datenbankschema sehr flexibel. Das heißt, falls man die Datenbank bereits erstellt hat und trotzdem noch zusätzliche Daten abspeichern möchte, die man allerdings noch nicht berücksichtigt hat, kann man ohne großen Aufwand die Datenbank erweitern.
Beim Speichermanagement wird sowohl ein dynamisches als auch ein fixes Verfahren angeboten. Das dynamische Verfahren hat keine fixe Speicherkapazität, sondern wird je nach Bedarf automatisch größer. Dies hat den Vorteil, dass die Datenbank, falls genügend Kapazität vorhanden ist, beliebig groß werden kann und somit keine Softwareeinschränkungen besitzt.
Die fixe Variante des Speichermanagements wird in der Fachsprache auch Capped Collection genannt. Die fixe Variante unterstützt eine größenbeschränkte Dokumentensammlung. Das heißt, eine Datenbank wird mit einer bestimmten Größe und einer Anzahl an Elementen angelegt. Somit kann maximal die vorher bestimmte Kapazität von der Datenbank eingenommen werden. Sobald die spezifizierte Größe erreicht ist, verhält sich die Capped Collection ähnlich wie ein digitaler Ringspeicher.[10]

[17] Not only SQL

7.4.2.1 Schema

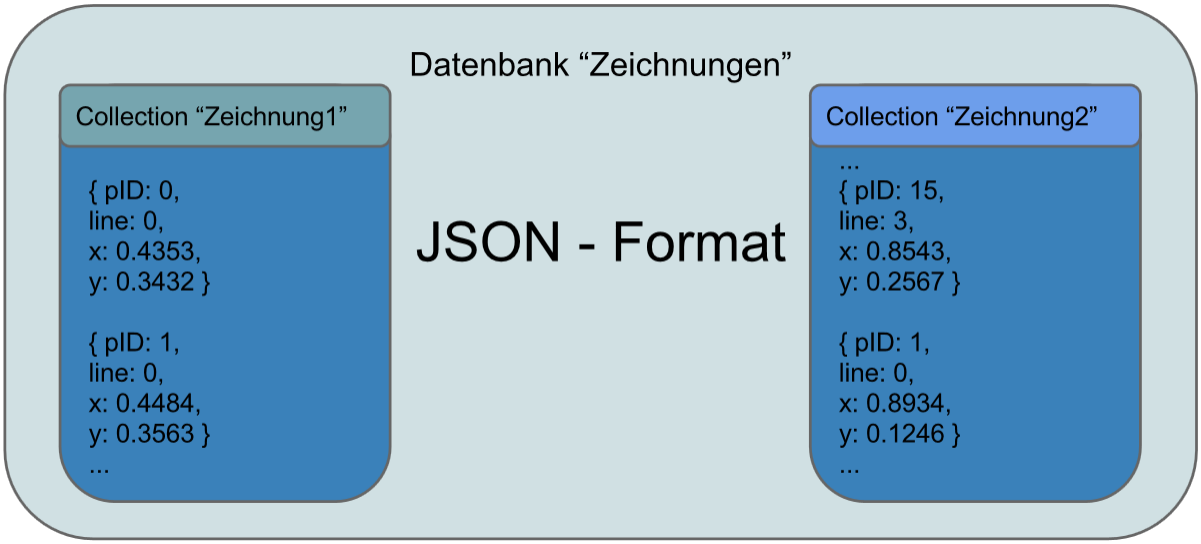


Abbildung 77: MongoDB Schema

Intern speichert MongoDB die Daten im JSON-Format[18]. JSON ist eine Abkürzung für Javascript Object Notation und ist ein Dateiformat von Javascript.
In der Grafik (siehe Abbildung 77) ist ein Beispiel von unserem Datenbankschema zu erkennen. Die Datenbank heißt hier “Zeichnungen“. In dieser Datenbank gibt es verschiedene Collections, welche die Koordinaten für die einzelnen Zeichnungen beinhalten.
In einer Collection werden zum einen die PunkteID und LineID sowie die X und Y Koordinaten abgespeichert. PunkteID und LineID wird verwendet, um nach dem Auslesen der Datenbank die einzelnen Punkte richtig zu sortieren. X und Y Koordinaten werden in einem Fließkommawert zwischen 0 bis 1 gespeichert, dies ermöglicht uns bei der Ausgabe die Zeichnungen je nach Größe der Zeichenfläche zu skalieren.

[18] www.wikipedia.org/wiki/JavaScript_Object_Notation

7.5 Raspberry Pi Konfiguration

7.5.1 Betriebssystem

Als Betriebssystem wird Raspian[19] verwendet. Raspbian ist eine speziell für den Raspberry Pi angepasste Linux-Distribution. Das Raspbian-Repository enthält mehr als 35000 eigens für den Raspberry Pi kompilierte Pakete.

Raspian kann von der offiziellen Seite des Raspberry Pi Projekts heruntergeladen[20] werden. Mit dem Programm Win32 Disk Imager wird das Image-File auf die SD-Karte geschrieben.

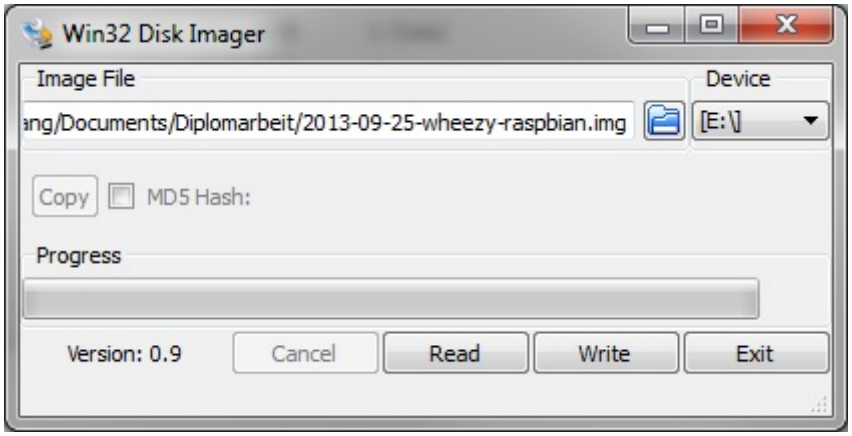


Abbildung 78: Win32 Disk Imager

Raspbian verfügt über eine graphische Benutzeroberfläche, die jedoch in diesem Anwendungsfall nicht benötigt wird. Der Zugriff auf den Raspberry Pi erfolgt über SSH. Mit SSH kann eine verschlüsselte Netzwerkverbindung aufgebaut werden. Für die Verbindung wird das Programm Putty verwendet.

7.5.2 Konfiguration

Der Raspberry Pi bezieht beim Starten, falls es an ein Netzwerk angeschlossen ist, automatisch eine IP Adresse. Dies ist jedoch nicht günstig, da in diesem Fall bei jeden Neustart im Netzwerk überprüft werden muss, welche IP Adresse der Raspberry Pi erhalten hat. Daher wurde der Ethernet-Schnittstelle des Raspberry Pi eine statische IP zugewiesen. Als Gateway wird die IP-Adresse der Ethernetschnittstelle des Computers, von dem auf den Raspberry Pi zugegriffen wird, angegeben. Dabei muss beachtet werden, dass die Ethernetschnittstelle des Computers auch diese IP Adresse hat.

Listing 15: /etc/network/interfaces

```
1 iface eth0 inet static
2 address 192.168.137.2
3 netmask 255.255.255.0
4 gateway 192.168.137.1
```

[19] http://www.raspbian.org/
[20] http://www.raspberrypi.org/downloads

Um die Konfigurationen mit der Konsole von Putty durchführen zu können, muss im Konfigurationsfenster nur die richtige IP eingetragen und mit dem `Connection type` Radio-Button muss SSH ausgewählt werden.

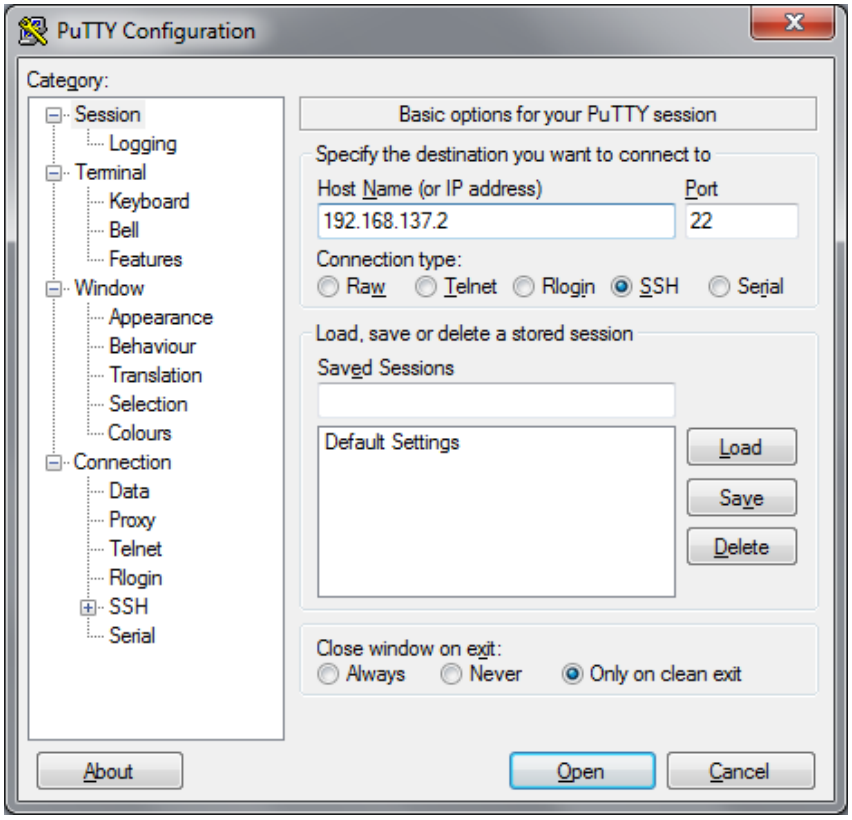


Abbildung 79: Putty Konfigurationsfenster

Um in der Konsole eine Datei öffnen und bearbeiten zu können, wird der Texteditor nano verwendet. Mit diesem lässt sich in der Kommandozeile mit `nano [Dateiname]` die gewünschte Datei öffnen.

7.5.2.1 WLAN Konfiguration

Damit die verschiedenen mobilen Geräte wie Tablets und Smartphones mit dem Raspberry Pi kommunizieren können, wird ein WLAN Access Point benötigt. Da das Raspberry Pi kein WLAN Modul enthält, wird ein einfacher WLAN Stick wie in Abbildung 80 dargestellt ist, verwendet.

Abbildung 80: Edimax WLAN-Stick

Weiters werden ein DHCP Server und ein eigenes WLAN benötigt. Für die Erzeugung des eigenen WLANs wird das Package `hostapd` mit `sudo apt-get install` installiert. Die Konfigurationsdatei von hostapd wurde wie folgt konfiguriert:

Listing 16: /etc/hostapd/hostapd.conf

```
interface=wlan0
driver=rtl871xdrv
ssid=ILPS
channel=1
```

Als erstes wurde das verwendete Interface angegeben, welches in diesem Fall das wlan0 Interface ist, weiters der benötigte Treiber für den WLAN-Stick und anschließend die SSID, welche `ILPS` ist. Die SSID gibt an, wie das WLAN Netzwerk heißt. Eine weitere wichtige Konfiguration ist der `channel`. Beim Channel sind je nach Land aus juristischen Gründen zwischen 11 und 14 verschiedenen Channels erlaubt. Damit es hier zu keinen Problemen kommen kann, wurde Channel 1 gewählt.

Für den DHCP-Server wird das Package `dnsmasq` verwendet. Der Dnsmasq ist ein einfacher DNS- und DHCP-Server für kleine Netzwerke. Im textttdnsmasq.conf File, welches sehr gut kommentiert ist, mussten nur noch folgende Zeilen hinzugefügt werden.

Listing 17: etc/dnsmasq.conf

```
interface=wlan0
dhcp-range=192.168.0.2,192.168.0.255,10min
```

In der ersten Zeile wurde angegeben, auf welchem Interface der DHCP-Server antwortet, anschließend musste nur noch die `dhcp-range` angegeben werden. Bei der DHCP Range wird der Bereich der möglichen IP Adressen definiert und optional kann eine Lease Time angegeben werden, die 10 Minuten beträgt. Da es sich in diesem Anwendungsfall um ein privates Netzwerk handelt, wurde als IP Adresse eine Klasse C Adresse gewählt. Die Klasse C beinhaltet 256 verschiedene private Netze mit jeweils 256 Adressen. Der Netzbereich geht von 192.168.0.0/24 bis 192.168.255.0/24. Die 1. verwendbare IP Adresse 192.168.0.1 wird dem WLAN Interface wlan0 zugeteilt.

Listing 18: etc/network/interfaces

```
iface wlan0 inet static
address 192.168.0.1
netmask 255.255.255.0
```

7.5.2.2 MongoDB Installieren

Für MongoDB gibt es kein fertiges Package für die ARM Architektur. Daher wird eine nonx86 Version [21] von MongoDB heruntergeladen und installiert[2]. Dazu müssen einige Packages installiert werden:

Listing 19: Packages installieren

```
sudo apt-get install build-essential libboost-filesystem-dev libboost-program-↙
    ↳ options-dev libboost-system-dev libboost-thread-dev scons libboost-all-↙
    ↳ dev python-pymongo git
```

Nun kann mit dem kompilieren von MongoDB begonnen werden. Dazu muss in das Verzeichnis, in das MongoDB gespeichert wurde, gewechselt und der Befehl `scons` ausgeführt werden. Es

[21] https://github.com/skrabban/mongo-nonx86

dauert einige Stunden bis der Raspberry Pi den Build Prozess abgeschlossen hat und MongoDB installiert werden kann.

Listing 20: MongoDB installieren

```
$ sudo scons --prefix=/opt/mongo install
```

MongoDB wird in den Pfad texttt/opt/mongo installiert. Auch dies nimmt einige Stunden Zeit in Anspruch.

7.5.2.3 Startup Daemon für MongoDB

Damit MongoDB beim Hochfahren von Raspbian automatisch gestartet wird, müssen einige Ordner mit speziellen Rechten erstellt und im Startupfile die richtigen Pfade angegeben werden. Weiters muss ein neuer Benutzer für MongoDB erstellt werden.

Listing 21: Neuer User erstellen

```
sudo adduser --firstuid 100 --ingroup nogroup --shell /etc/false --disabled-↙
    ↳ password --gecos "" --no-create-home mongodb
```

Für den User `mongodb` wird kein eigenes Home-Directory erstellt. Desweiteren verfügt er über kein Passwort.

Anschließend muss noch mit `mkdir` (make directory) ein Ordner erstellt werden, der die Datenbank enthält.

Listing 22: Ordner für Datenbanken

```
sudo mkdir /home/pi/databases
sudo chown mongodb:nogroup /home/pi/databases
```

Mit dem Befehl `chown` (change owner) wird dem Ordner `databases` der Benutzer `mongodb` aus der Gruppe `nogroup` als Eigentümer zugewiesen.

Dasselbe muss noch für den Ordner gemacht werden, der die Logdatei enthält. Diese hat den Pfad `/var/log/mongodb`.

Nun muss die Konfigurationsdatei und die Startupdatei in den `/etc` Ordner kopiert werden.

Listing 23: Mongodb Startupfile Ausschnitt

```
rm /home/pi/databases/mongod.lock

mongod --repair
# Default defaults.  Can be overridden by the /etc/default/$NAME
NAME=mongodb
CONF=/etc/mongodb.conf
DATA=/home/pi/databases
LOGDIR=/var/log/mongodb
PIDFILE=/var/run/$NAME.pid
LOGFILE=$LOGDIR/$NAME.log  # Server logfile
ENABLE_MONGODB=yes
```

Bei jedem Start muss die vorhandene `mongod.lock` Datei gelöscht werden, da sie ansonsten zu Problemen führen kann.

Listing 24: Rechte für Startupfile

```
1 sudo chmod u+x /etc/init.d/mongod
2 sudo update-rc.d mongod defaults
```

Dem Eigentümer des Startupfiles muss das Recht gegeben werden, dass er die Datei ausführen darf. Dies geschieht mit dem Befehl `chmod u+x`. Wobei das u für User oder auch Eigentümer und das x steht für ausführen (eXecute). Anschließend wird das Startupfile mit update-rc.d als Service registriert.

7.5.2.4 NodeJS installieren

Für NodeJS gibt es ein fertiges Package, das nur noch installiert werden muss. Jedoch ist dieses bereits sehr veraltet und weist einige Unschönheiten auf. Daher muss auch hier die aktuelle Version für den Raspberry Pi heruntergeladen und installiert [3] werden. Die Version für den Raspberry Pi ist auf der Seite mit allen NodeJS Versionen[22] zu finden. Dabei muss nach der aktuellsten Version gesucht werden, die eine `node-version-linux-arm-pi.tar.gz` Datei enthält.

Die Datei wird entpackt und anschließend in den selbst erstellten Ordner `/opt/node` kopiert. Anschließend muss das Verzeichnis zu `$PATH` hinzugefügt werden. `$PATH` ist eine Umgebungsvariable, die den Suchpfad für Befehle enthält.

Listing 25: /etc/profile

```
1 NODE_JS_HOME="/opt/node"
2 PATH="$PATH:$NODE_JS_HOME/bin"
3 export PATH
```

Die obigen Codezeilen müssen in der Profile Datei vor dem `export PATH` Befehl eingefügt werden. Nachdem der Raspberry Pi neugestartet wurde, können bereits NodeJS Programme ausgeführt werden. Um jedoch ein NodeJS Programm direkt beim Starten des Rasperry Pis zu starten, muss ein Startscript geschrieben werden.

7.5.2.5 Startscript für NodeJS

Bei Linux liegen die Start- bzw. Stop-Skripte im Verzeichnis `/etc/init.d`. Die Befehle in diesen Skripten können entweder manuell oder automatisch beim Booten oder Herunterfahren des Systems aufgerufen werden.

Listing 26: /etc/init.d/nodejs.sh

```
1  #!/bin/bash
2
3  NODE=/opt/node/bin/node
4  SERVER_JS_FILE=/home/pi/ILPS-Software/app.js
5  USER=root
6  OUT=/home/pi/nodejs.log
7
8  case "$1" in
9
10 start)
```

[22] http://nodejs.org/dist/

```
11          echo "starting node: $NODE $SERVER_JS_FILE"
12          sudo -u $USER $NODE $SERVER_JS_FILE > $OUT 2>$OUT &
13          ;;
14
15  stop)
16          killall $NODE
17          ;;
18
19  *)
20          echo "usage: $0 (start|stop)"
21  esac
22
23  exit 0
```

In `$1` steht das 1. Argument. Ist dies `start`, so wird alles in `start` ausgeführt. In `start` wird das NodeJS Server als Benutzer `root` ausgeführt und es wird angegeben, dass alle Logmeldungen ins angegebene Logfile gespeichert werden sollen. `2>$OUT &` gibt an, dass auch die Fehlermeldungen in dieses Logfile gespeichert werden sollen. Bei `stop` werden alle Prozesse mit NodeJS geschlossen und falls keines der beiden zutrifft, wird in die Konsole geschrieben, welche Möglichkeiten vorhanden sind.

Listing 27: Zugriffsrecht ändern

```
1  chmod 755 nodejs.sh
2  sudo update-rc.d nodejs.sh defaults
```

Mit `chmod 755` wird die Datei für den Eigentümer ausführbar gemacht. Alle anderen dürfen nur lesen und schreiben. Mit `update-rc.d` registriert das File `nodejs.sh` als Service. Mit diesen Einstellungen startet das NodeJS Programm automatisch beim Hochfahren.

7.5.2.6 DNS-Server und Umleitung

Mithilfe des `dnsmasq` Packages kann auch ein eigener DNS-Server erstellt werden.

Listing 28: DNS Wildcard Configuration hostapd.conf

```
1  address=/#/ip/192.168.0.1
```

Mithilfe einer Wildcard Configuration werden alle DNS-Requests auf die IP-Adresse des Raspberry Pis geleitet. Um im Browser in der Adresszeile `ilps` anzuzeigen wird der lokalen IP in der `hosts` Datei der Domainname zugewiesen.

Listing 29: /etc/hosts

```
1  192.168.0.1     ilps
```

Um den Zugriff auf das graphische Userinterface zu erleichtern wird, egal was im Browser eingegeben wurde, das Userinterface aufgerufen. Dies wurde mit dem Package `nginx`[23] gelöst.

Listing 30: /etc/nginx/sites-enabled/default

```
1  server {
2  server_name _;
3  listen 80;
```

[23] http://nginx.org/

```
4  return 302 http://ilps:8080;
5  }
6
7  server {
8  server_name _;
9  listen 443;
10 return 302 http://ilps:8080;
11 }
```

Der erste Server fängt alle Anfragen, die per HTTP gesendet werden, ab und hört daher auf den Port 80. Die abgefangenen Anfragen werden anschließend auf `http://ilps:8080` umgeleitet. Dasselbe wird mit allen Anfragen gemacht, die durch Port 443 (z.B. HTTPS) erfolgen.

Da iOS Geräte beim Verbinden mit einem Netzwerk versuchen, eine ihnen bekannte Internetseite aufzurufen und da dies bei einer solchen Umleitung nicht funktioniert, gehen sie davon aus, dass es sich um ein Captive Portal[24] handelt, welches einen HTTP-Client in einem Netzwerk auf eine spezielle Webseite umleitet, bevor sich dieser normal in das Internet verbinden kann. Daher erscheint ein Popupfenster um sich zu authentifizieren. Es gibt für jedes OS eigene Seiten, die versucht werden zu öffnen[4]. Damit das OS nun davon ausgeht, dass es sich um kein Captive Portal handelt, wird beim Aufruf dieser Adressen eine HTML-Datei mit besonderem Inhalt zurückgeliefert.[25].

Listing 31: Umgehung von Authentifizierungsfenstern

```
1  server {
2          server_name clients3.google.com www.appleiphonecell.com *.apple.com www↙
               ↳ .itools.info www.ibook.info www.airport.us www.thinkdifferent.us ↙
               ↳ *.apple.com.edgekey.net *.akamaiedge.net gsp1.apple.$
3          listen 80;
4          root /home/pi;
5
6          location /{
7                  try_files $uri /success.html;
8          }
9  }
```

Das Success-File wird nur geladen, wenn die Anfrage durch Port 80 erfolgte.

[24] http://de.wikipedia.org/wiki/Captive_Portal
[25] http://www.apple.com/library/test/success.html

7.6 Bedienungsanleitung - Benutzeroberfläche

Die graphische Oberfläche kann mit einem beliebigen Browser geöffnet werden. Damit ist das komplette System plattformunabhängig und kann sowohl von Android, iOS als auch von Windows geöffnet bzw. gesteuert werden.
Im folgenden ist unter anderem die Rede von sogenannten "Modalfenstern". Ein Modalfenster ist ein einfaches Dialogfenster, welches erscheint, sobald ein Button angeklickt wird.

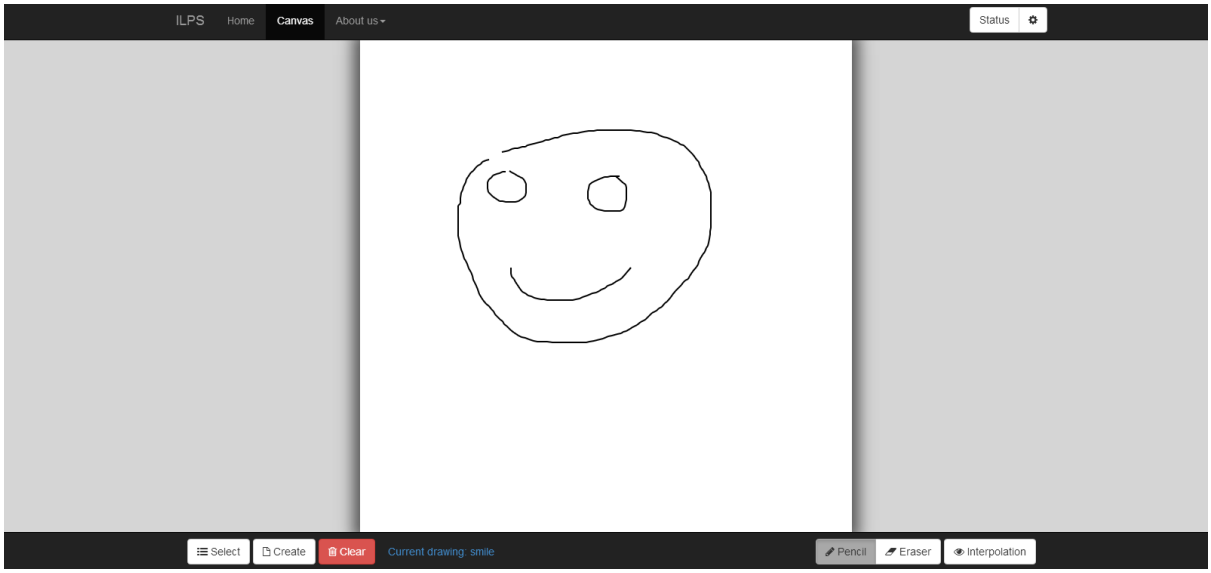


Abbildung 81: graphische Oberfläche - Übersicht

In Abbildung 81 ist die Hauptansicht zu sehen. In dieser Ansicht können Zeichnungen erstellt, geöffnet, gelöscht und bearbeitet werden.

In der rechten unteren Ecke sind folgende 3 Buttons zu finden.

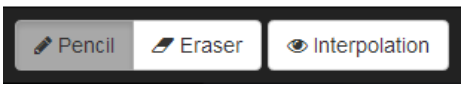


Abbildung 82: Toolbuttons

Die `Pencil`- und `Eraser` Buttons verhalten sich wie Radiobuttons. Das heißt, es kann immer nur einer gedrückt sein, sobald einer angeklickt wird, wechselt der andere in den invertierten Zustand.
Ist der `Pencil` ausgewählt, können auf der Hauptzeichenfläche Linien gezeichnet werden.
Der `Eraser` ist das Gegenteil vom `Pencil`, ist dieser Ausgewählt, so können auf der Zeichenfläche Linien ausradiert bzw. gelöscht werden.

7.6.1 Interpolation

In Abbildung 82 ist der Button zu sehen, welcher die Interpolationslinien auf der Hauptzeichenfläche anzeigt. Interpolationslinien werden rot gezeichnet. Diese Linien zeichnen den Weg ein, der vom Laserpunkt zwischen den Linien zurückgelegt wird.

In Abbildung 83 ist das gleiche Bild wie bei Abbildung 81 zu sehen. Jedoch wurden im folgenden Bild die Interpolationslinien eingezeichnet.

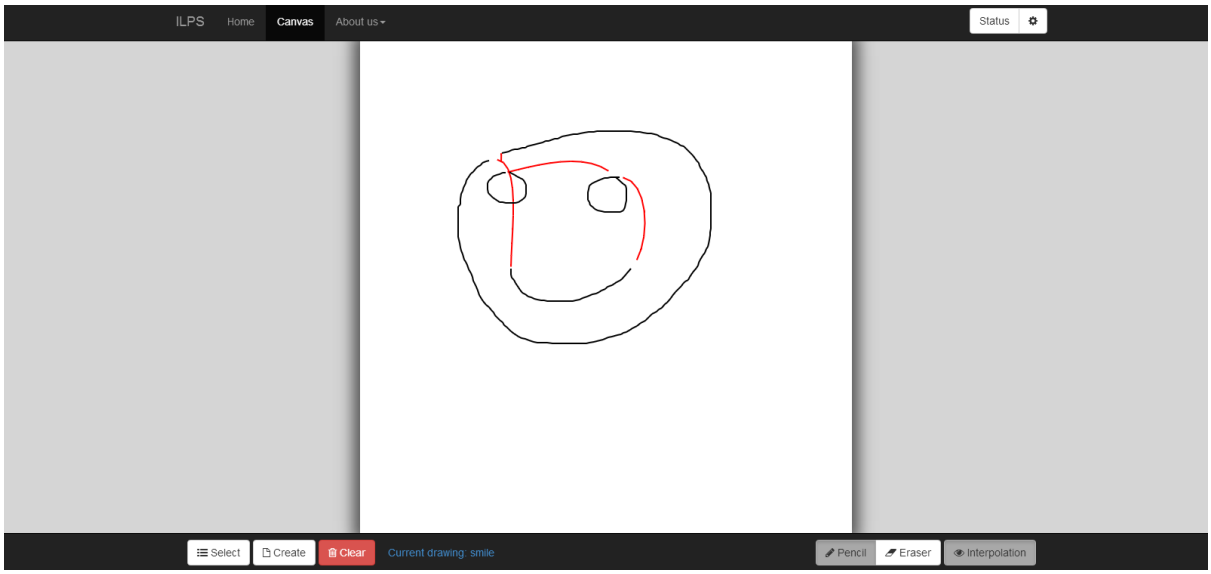


Abbildung 83: Übersicht mit Bezierlinien

Der Laserpunkt fährt zwar all diese Linien ab, jedoch wird bei den roten Linien der Laser deaktiviert. Durch das temporäre Ausschalten des Lasers ist es möglich mehrere nicht verbundene Linien zu zeichnen.

7.6.2 Auswählen/Löschen/Erstellen

Die folgenden Buttons sind in der linken unteren Ecke zu finden.

Abbildung 84: Modalbuttons

Bei einem Klick auf den `Select` oder `Create` Button öffnet sich ein dafür zuständiges Modalfenster. Der `Clear` Button wird verwendet um die aktuelle Zeichenfläche zu löschen.

7.6.2.1 Auswählen

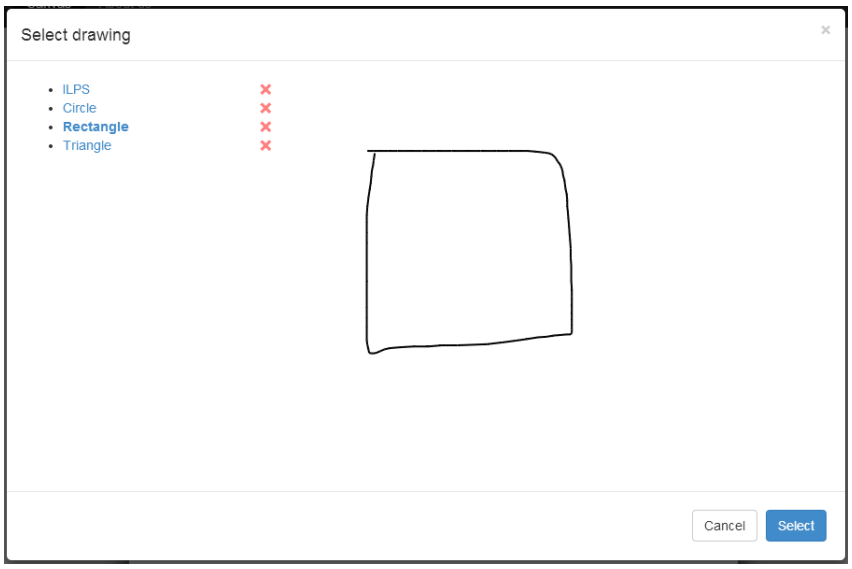


Abbildung 85: Auswählen - Modal

Beim Klick auf den Button `Select` öffnet sich das Modal von Abbildung 85. In diesem sind die bereits bestehenden Zeichnungen zu sehen. Bei einem einfachen Klick auf einen Zeichnungsnamen wird auf der rechten Seite eine Vorschau dieser Zeichnung angezeigt.
Nachdem eine Zeichnung ausgewählt wurde, kann mit einen Klick auf `Select` die Zeichnung ausgewählt und in der Hauptansicht bearbeitet werden. Zeitgleich wird das Bild über den STM32 zu den Galvanometer geschickt und mit dem Laser dargestellt.

7.6.2.2 Löschen

Zusätzlich ist in Abbildung 85 neben jeder Zeichnung ein rotes X zu sehen. Klickt man auf das X erscheint ein neues Modal (Abbildung 86) mit einer Abfrage.

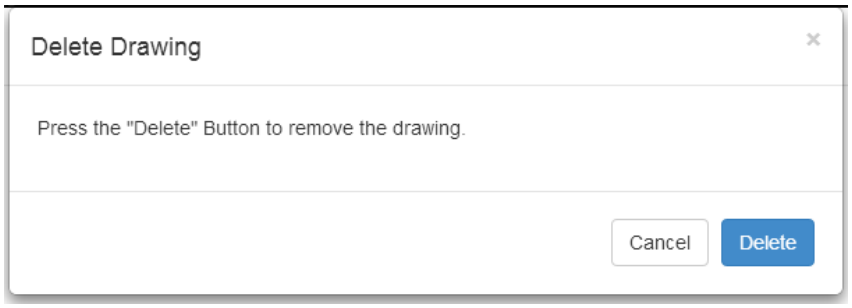


Abbildung 86: Löschen - Modal

Diese Abfrage dient zur Sicherheit, dass Zeichnungen nicht aus Versehen gelöscht werden. Falls die Zeichnung entfernt werden soll, muss dies mit dem `Delete` Button bestätigt werden. Anschließend wird die Zeichnung gelöscht.

7.6.2.3 Erstellen

Abbildung 87: Erstellen - Modal

Das in Abbildung 87 zu sehende Fenster erscheint bei einem Klick auf den `Create` Button. In diesem kleinen Modal kann der Name einer neuen Zeichnung eingegeben werden. Bei ungültigen Namen erscheint eine Fehlermeldung (siehe Abbildung 88), bei gültigem Namen wird die Zeichnung in der Datenbank erstellt. Daraufhin kann die neue Zeichnung in der Hauptansicht bearbeitet werden.

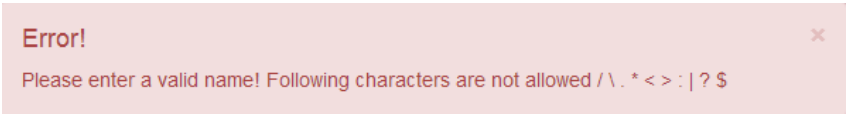


Abbildung 88: Warnung - Modal

Bei Fehleingaben wird ein Popup-Fenster mit einer Fehlermeldung geöffnet, welches Informationen über den Fehler angibt.

7.6.3 Status und Einstellungen

Buttons für das Statusfenster und Einstellungen sind in Abbildung 81 rechts oben zu finden.

Abbildung 89: Status und Einstellung Button

7.6.3.1 Status - Modal

Bei einem Klick auf den `Status` Button öffnet sich folgendes Modal:

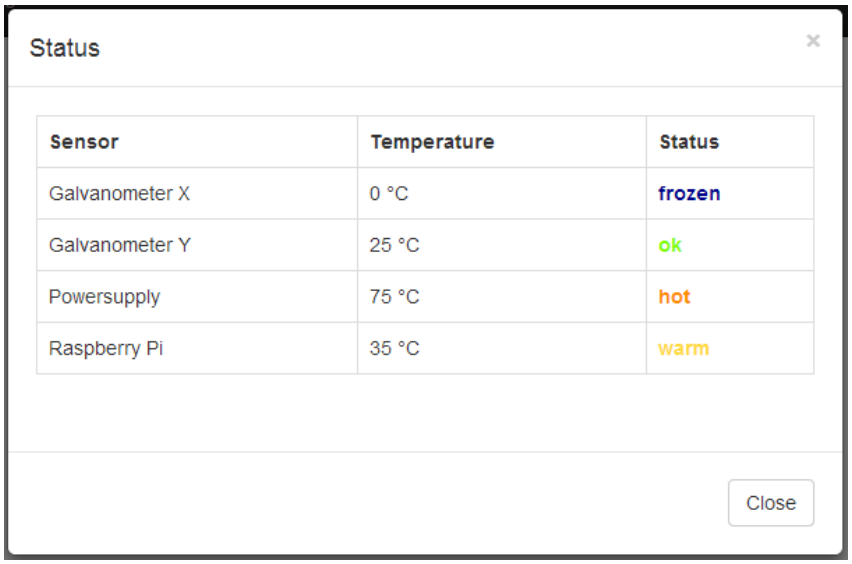


Abbildung 90: Status - Modal

Beim Modal in Abbildung 90 sind nur Beispieltemperaturen beinhaltet. Bei der laufenden Software werden diese Temperaturen direkt von Temperatursensoren abgelesen und live dargestellt.

7.6.3.2 Einstellungen - Modal

Wird auf das Zahnrad geklickt erscheint erst ein Modal mit einer Passwortabfrage, damit nicht jeder Client Änderungen an den Einstellungen vornehmen kann.

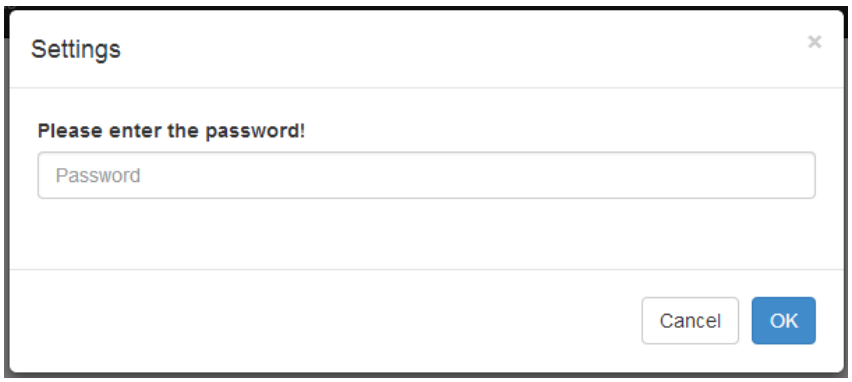


Abbildung 91: Passwort - Modal

Bei richtiger Eingabe des Passwortes wird das Modal geschlossen und es öffnet sich Abbildung 92.

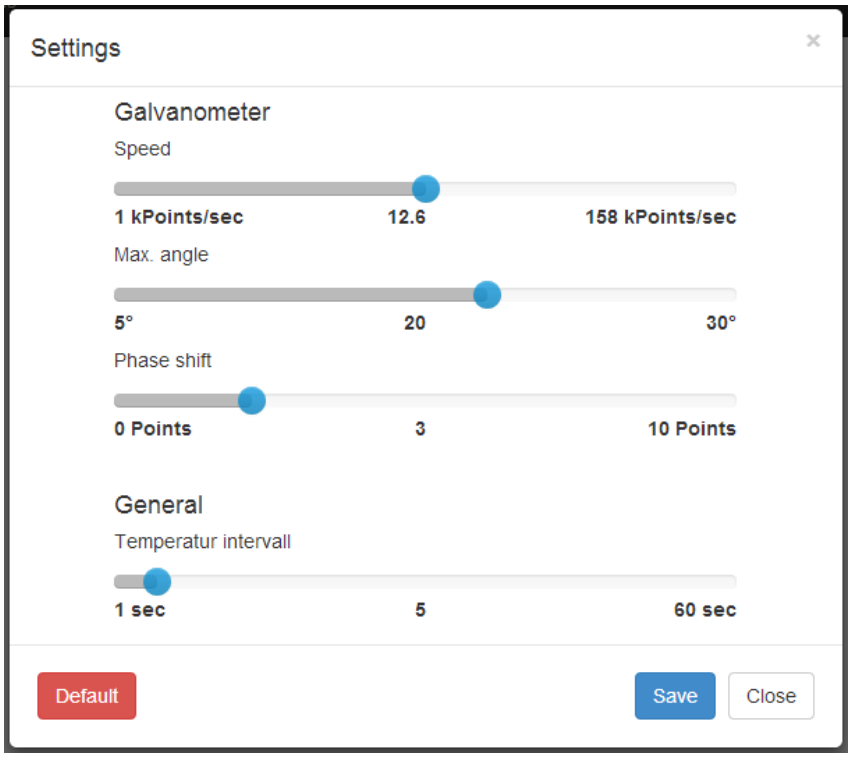


Abbildung 92: Einstellungen - Modal

Hier können nun folgende Einstellungen geändert werden:

- **Galvanometer Geschwindigkeit**
  Die Geschwindigkeit bestimmt, wie viele Punkte pro Sekunde mit dem Laserpunkt abgefahren werden. `kPoints/sec` steht für Kilopunkte (1000 Punkte) pro Sekunde.
- **Galvanometer maximaler Winkel**
  Dieser Slider gibt den Winkel an, der von einem Galvanometer maximal zurückgelegt wird. Die Einstellung dieses Winkels dient zur Korrektur der sphärischen Verzerrung.
- **Phasenverschiebung**
  Desto höher die Geschwindigkeit der Galvanometer ist, desto ungenauer wird der Ein- und Ausschaltzeitpunkt des Lasers. Um dieser Ungenauigkeit entgegen zu wirken, kann mit der Phasenverschiebung eingestellt werden, ab welchem sichtbaren Punkt der Laser aktiviert wird.
- **Zeitintervall der Temperatur**
  Das Zeitintervall gibt an, wie viele Sekunden zwischen den Temperaturmessungen gewartet wird.

Änderungen an den Einstellungen können mit den Schieberegler vorgenommen werden. Diese werden sofort von der Hardware übernommen. Bei einer Betätigung vom `Save` Button werden die derzeitigen Einstellungen in der Datenbank abgespeichert. Einstellungen die abgespeichert wurden, sind nach einem Neustart der Software bzw. Hardware weiterhin gespeichert.
Falls die Einstellungen nicht mithilfe des `Save` Buttons gespeichert werden, werden die Einstellungen nach einem Neustart verworfen.

8 Programmcode

Der Kern des Programmcodes besteht aus verschiedenen JavaScript und HTML Dateien. Die HTML Dateien sind hauptsächlich für die Benutzeroberfläche im Webbrowser zuständig. Zusätzlich werden Bibliotheken, ein Frontend Framework für das Design und JavaScript Dateien eingebunden.
Mit JavaScript wird alles gemacht, was für den normalen Benutzer nicht direkt sichtbar ist. Um mit JavaScript einfacher HTML Objekte manipulieren zu können wird die Bibliothek jQuery verwendet. Eine weitere Bibliothek, welche von der Software verwendet wird, ist Modernizr. Modernizr wird für ein besseres Verhalten der Weboberfläche auf einem Endgerät mit Touchverhalten eingesetzt.

8.1 Standardablauf der Software

Im folgenden Flussdiagramm (Abbildung 93) ist der normale Ablauf zu sehen, welcher eintritt, wenn man eine Zeichnung erstellt oder eine bereits vorhandene lädt.

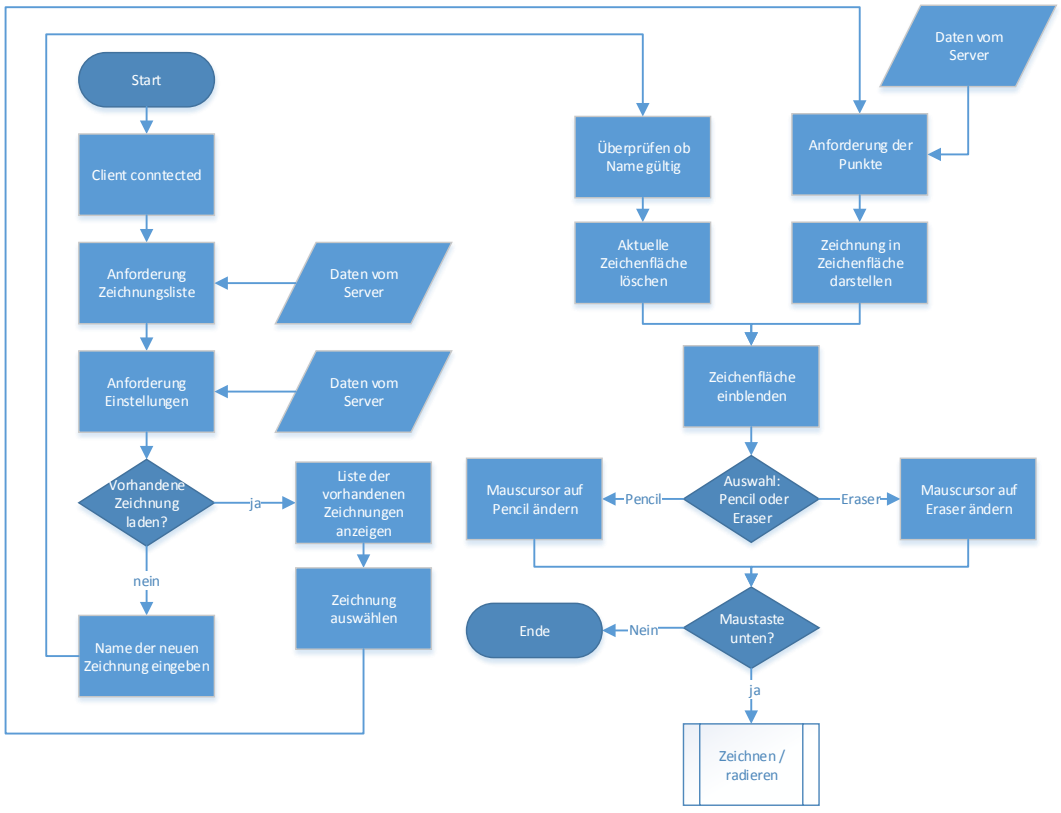


Abbildung 93: Standardablauf der Software

Bei der letzten Abfrage in Abbildung 93 wird geprüft, ob die Maustaste noch unten ist. Solange sie gedrückt ist, wird der Teilprozess von Abbildung 94 aufgerufen.

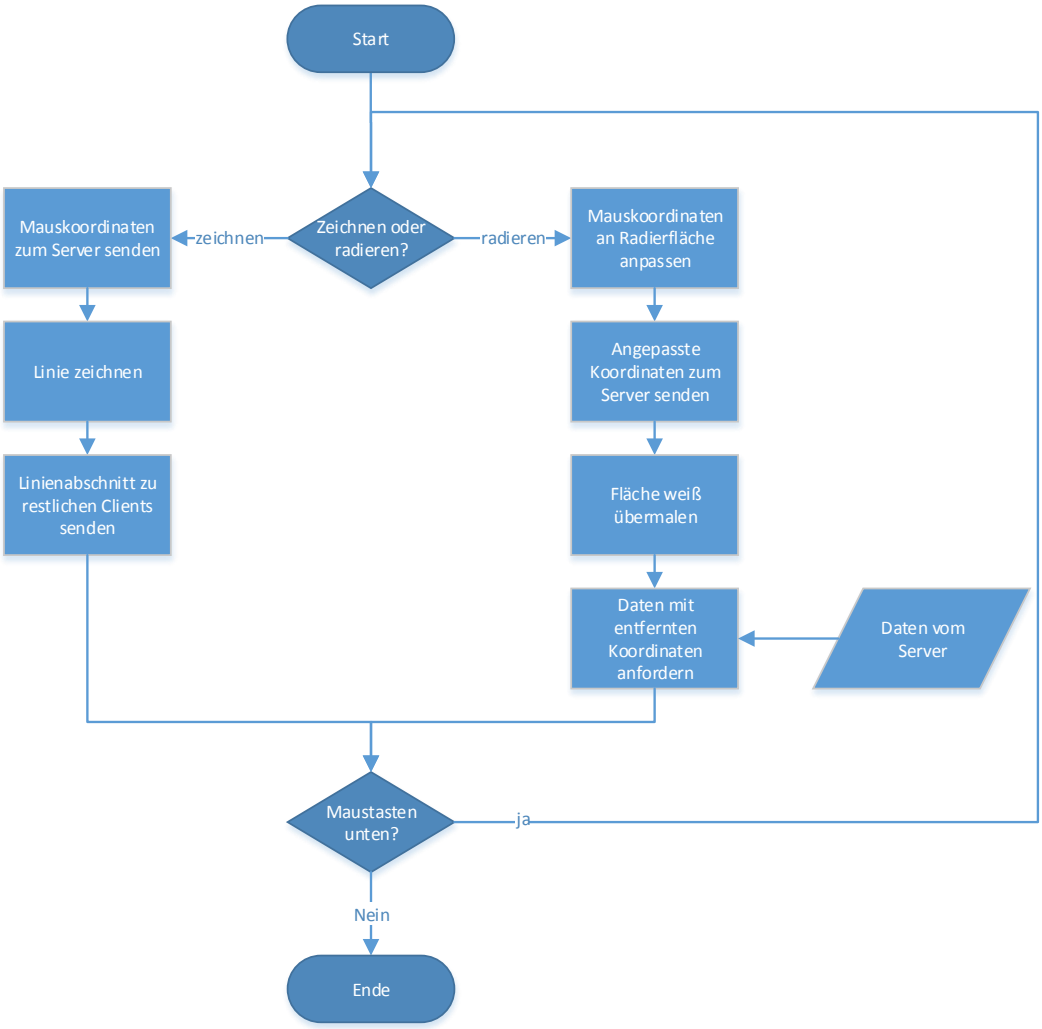


Abbildung 94: Zeichen-/Radiervorgang

In diesem Flussdiagramm wird erläutert, wie sich die Software beim zeichnen bzw. radieren verhält. Falls eine Linie gezeichnet wird, werden die Koordinaten zum Server gesendet. Beim zeichnenden Client selbst wird die Linie direkt in der Zeichenfläche gezeichnet und anschließend die Koordinaten an die restlichen Clients versendet, welche sich im ILPS-Netzwerk befinden. Im Fall eines Radiervorgangs, werden die Mauskoordinaten an die Fläche des Radiergummis angepasst und zum Server gesendet. Wie bereits beim Zeichenvorgang, wird bei dem radierenden Client die Linie für einen flüssigeren Radiervorgang mit weiß übermalt. Daraufhin wird vom Server das neue Array mit allen Punkten und Linien angefordert und neu gezeichnet.

8.2 HTML Codesegmente

Das Frontend-Framework, das beim Design benutzt wird, nennt sich Bootstrap. Bootstrap ist ein modernes Template welches von Twitter entwickelt wurde. Durch die bereits von Bootstrap vorhandenen CSS-Klassen kann man mit wenigen Zeilen Code, ein gut aussehendes Popup Fenster oder eine ganze Navigationsleiste erstellen.

8.2.1 Modal

Folgender HTML Code erstellt ein Popup Fenster, sprich ein Modal, mit einer Textbox und zwei Buttons.

Listing 32: create-modal

```
<!-- Modal -->
<div class="modal fade" id="create-modal" tabindex="-1" role="dialog" aria-
    ↪ hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-
                    ↪ hidden="true">&times;</button>
                <h4 class="modal-title">Create a new database</h4>
            </div>
            <div class="modal-body">
                <form role="form">
                    <div class="form-group">
                        <label for="name">Name:</label>
                        <input type="text" class="form-control" id="name" 
                            ↪ placeholder="Name">
                    </div>
                </form>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="
                    ↪ modal">Cancel</button>
                <button type="button" class="btn btn-primary" id="create">
                    ↪ Create</button>
            </div>
        </div>
        <!-- /.modal-content -->
    </div>
    <!-- /.modal-dialog -->
</div>
<!-- /.modal -->
```

Anschließend sieht das Modal im Webbrowser wie bei Abbildung 95 aus.

Abbildung 95: create-modal

Ein Modal ist immer in 3 Teilen aufgebaut. Der erste Teil ist der Header (Z. 5-8), in diesem steht meist nur eine Überschrift, die angibt, worum es in diesem Modal geht. Der zweite Teil besteht aus dem Body (Z. 9-16), hier sind die Funktionen des Modals eingebaut, wie im Beispiel bei Abbildung 95 besteht der Body aus einem Label (Z. 12) und einer Textbox (Z. 13), in welcher der Name der neuen Zeichnung eingegeben werden kann. Zum Schluss kommt der dritte Teil, der Footer (Z. 17-20). Dieser Teil wird meistens mit Buttons versehen, die das Modal schließen bzw. die eingegebenen Daten abschicken.

8.2.2 Navigationsleiste

Folgender HTML Code erstellt eine Navigationsleiste, die zur Bedienung der graphischen Oberfläche verwendet wird.
Eine Navigationsleiste ist im Gegensatz zu einem Modal nicht immer aus 3 Teilen aufgebaut, sondern besteht je nach Anwendung aus mehreren Gruppen. Diese bestehen aus Elementen wie Radiobuttons, Buttons, Labels oder ähnlichem. Bei Abbildung 96 besteht die linke sowie die rechte Hälfte aus einer eigenen Navbar, die unabhängig voneinander sind. Bei der linken Hälfte (Z. 6-13) werden normale Buttons (Select, Create, Delete) verwendet. In der rechten Hälfte (Z. 14-33) sind 2 Radiobuttons (Pencil und Eraser) sowie ein normaler Button (Interpolation) vorhanden.

Listing 33: navbar

```
1  <div class="navbar-wrapper">
2      <div class="container">
3          <div class="navbar navbar-inverse navbar-fixed-bottom">
4              <div class="container">
5                  <div class="navbar-collapse collapse">
6                      <form class="navbar-form navbar-left form-inline" role="↙
                           ↳ form">
7                          <button id="choose-button" class="btn btn-default"><i ↙
                               ↳ class="fa fa-list-ul"></i> Select</button>
8                          <button id="create-button" class="btn btn-default"><i ↙
                               ↳ class="fa fa-file-o"></i> Create</button>
9                          <button id="delete-button" class="btn btn-danger"><i ↙
                               ↳ class="fa fa-trash-o"></i> Delete
10                         </button>
```

```
11                          <div class="btn btn-link" id="database-name-button">↙
                                ↳ Current Database: <span
12                                  id="database-name">none</span></div>
13                      </form>
14                      <form class="navbar-form navbar-right" id="right" role="↙
                            ↳ form">
15                          <div class="form-group">
16                              <div class="btn-group" data-toggle="buttons-radio">
17                                  <button id="pencil" class="btn btn-default btn-↙
                                        ↳ custom-mode active"><i
18                                          class="fa fa-pencil"></i>
19                                      Pencil
20                                  </button>
21                                  <button id="eraser" class="btn btn-default"><i ↙
                                        ↳ class="fa fa-eraser"></i>
22                                      Eraser
23                                  </button>
24                              </div>
25                          </div>
26                          <div class="form-group">
27                              <div class="form-element" data-toggle="button-radio↙
                                    ↳ ">
28                                  <button id="interpolation" class="btn btn-↙
                                        ↳ default"><i class="fa fa-eye"></i>
29                                      Interpolation
30                                  </button>
31                              </div>
32                          </div>
33                      </form>
34                  </div>
35              </div>
36          </div>
37      </div>
38  </div>
```

Die Navigationsleiste sieht anschließend wie bei Abbildung 96 aus.

Abbildung 96: Navigationsleiste unten

8.3 JavaScript Codesegmente

JavaScript steuert die einzelnen HTML Elemente und ermöglicht es, Objekte direkt zu bearbeiten ohne, dass die Seite neu geladen werden muss. Der größte Teil der Software besteht aus JavaScript. Insgesamt werden sieben JavaScript Dateien verwendet. Drei sind für den Client zuständig und vier für den Server.

8.3.1 Kommunikation Server - Client

Wie bereits bei Abbildung 76 erwähnt, kann in NodeJS der Client mit dem Server über sogenannte Sockets kommunizieren. Im JavaScript Code sieht die Übertragung zwischen Client und Server wie folgt aus.

Listing 34: Kommunikation - Client

```
1 socket.emit('choose sheet', {name:selectedName});
```

In diesem Beispiel sendet der Client mit einem `socket.emit` dem Server einen Datensatz `name:selectedName` und einen string 'choose sheet'. Durch diesen string erkennt der Server, zu welcher Funktion der Datensatz gehört und was er ausführen soll.

Listing 35: Kommunikation - Server

```
1 socket.on('choose sheet', function (data) {
2     function(...){
3         ...
4     };
5 });
```

Sobald der Server eine Anforderung durch `emit` von einem Client bekommt, sucht er nach einen dazugehörigen `socket.on`.
Die Kommunikation ist von beiden Richtungen möglich, sprich der Server sowie der Client können `socket.on` und `socket.emit` verwenden. Zusätzlich gibt es noch verschiedene Varianten von `emit`. Zum einen gibt es das normale `socket.emit`, welches nur dem Server bzw. dem Client, der die Anfrage gestellt hat eine Antwort zurück sendet. Zum anderen ist es möglich mit `io.sockets.emit` alle Clients, oder mit `socket.broadcast.emit` alle, außer den Client, der die Anfrage zum Server gesendet hat, anzusprechen.

8.4 Server

Die Software auf Seiten des Servers besteht aus vier JavaScript Dateien. Jede dieser vier Dateien übernimmt je eine Aufgabe wie z.B. MongoDB, SPI, Temperaturüberwachung oder SocketIO.

8.4.1 Webserver: app.js

Der Einstiegspunkt des Webservers bildet die Datei `app.js`:

Listing 36: app.js

```
1  var app = require('http').createServer(handler),
2      io = require('socket.io').listen(app),
3      static = require('node-static'), // for serving files
4      socketIO = require(__dirname + '/core/socket.io.js').init(io),
5      SPI = require('pi-spi'),
6      spi = require(__dirname + '/core/spi.js').init(SPI);
7
8  //supress debug messages
9  io.set('log level', 1);
10 // This will make all the files in the current folder
11 // accessible from the web
12 var fileServer = new static.Server('./client');
13
14 function handler(request, response) {
15
16     request.addListener('end', function () {
17         fileServer.serve(request, response);
18     });
19
20     request.resume();
21 }
22 //sets the Port for the Webserver
23 app.listen(8080);
```

Im ersten Teil werden die Variablen (`app, io, static,..`) deklariert und mit der zugehörigen Bibliothek verknüpft. In Zeile 1 wird der Webserver erstellt. Ab Zeile 23 wird der Webserver gestartet und hört ab diesem Zeitpunkt auf den Port 8080.
Node-static ist ein weiteres Plugin von NodeJS, welches einen Fileserver erstellt, der den Clients die benötigten JavaScript Dateien übergibt.
`io` und `SPI` binden die zwei Bibliotheken SocketIO (siehe Kapitel 7.2.4 SocketIO) und `pi-spi` (siehe Kapitel 8.4.4 `spi.js`) ein.
`socketIO` und `spi` initialisieren die dazugehörigen JavaScript Dateien mithilfe der zuvor eingebundenen Bibliothek (`io` und `SPI`).

8.4.2 Programmlogik des Server: socket.io.js

In der `socket.io.js` Datei befinden sich alle Hauptfunktionen der Anwendung. Eigentlich würde diese Datei reichen, jedoch wird der Programmcode in mehrere kleinere Dateien ausgelagert, damit die Übersicht nicht verloren geht.

8.4.2.1 Initialisierung

Listing 37: Initialisierung von SocketIO

```
1  //init for the serverconnection
2  exports.init = function (tmp) {
3      io = tmp;
4
5      database.connect();
6      io.sockets.on('connection', connection);
7  };
```

Beim Start der Software wird eine Verbindung mit der Datenbank und zur SocketIO Bibliothek aufgebaut. Anschließend wird für jeden Client, der sich mit der Software verbindet, ein eigener Socket erstellt und der Funktion `connection` übergeben.

Listing 38: connection Function

```
1  var connection = function (socket) {
2      console.log('Client connected');
3      socket.on('...', function (...){
4          ...
5          };
6      )};
```

Falls sich der Client erfolgreich verbindet, wird in der Konsole des Servers eine Meldung mit `Client connected` ausgegeben. In der `connection` Funktion befinden sich alle `socket.on` Funktionen die von einem Client aufgerufen werden können. Da jeder Endnutzer einen eigenen Socket zugewiesen bekommt, kann der Server erkennen, welcher Client Daten anfordert bzw. sendet.

8.4.2.2 Einstellungen abrufen

Listing 39: Einstellungen abrufen

```
1  socket.on('getSettings', function () {
2      database.getSettings(function (error, data) {
3          var settings = {
4              speed:(data[0] ? data[0].speed : 600),
5              angle:(data[0] ? data[0].angle : 20),
6              intervall:(data[0] ? data[0].intervall : 5000)
7          };
8          settings.status = (data[0] ? 'dbIsNotEmpty' : 'dbIsEmpty');
9
10         io.sockets.emit('settings', settings);
11     });
12 });
```

Die Einstellungen für das Raspberry Pi sowie die Galvanometer werden einerseits abgerufen sobald ein neuer Client sich mit der Software verbindet und andererseits sobald auf den `Settings`

Button in der Weboberfläche gedrückt wird.
Nachdem eines dieser Szenarien eingetroffen ist, holt sich der Server mit einem Zugriff auf die Datenbank die aktuellen Einstellungen. Da diese lediglich ein Array mit Werten zurückgibt, werden die Daten überprüft und nur übernommen, wenn die Daten Zahlen sind, mit denen der STM32 sowie das Raspberry Pi arbeiten kann. Falls keine richtigen Werte zurückkommen, werden die Standardwerte übernommen.
Anschließend werden die neuen Einstellungen an alle Clients weiterverbreitet.

8.4.2.3 Einstellungen ändern

Listing 40: Einstellungen speichern

```
1 socket.on('changeSettings', function (settings) {
2     spi.setSpeed(settings.speed);
3     temperatureController.setTempInterval(settings.intervall);
4     phimax = settings.angle * Math.PI / 180;
5     database.saveSettings(settings);
6 });
```

Die Einstellungen werden geändert, sobald auf der Weboberfläche Änderungen an den Einstellungen vorgenommen wurden und auf den `Save` Button gedrückt wird.
Bei neuen Einstellungen wird der neue Maximalwinkel sowie der neue Zeitbereich zwischen Temperaturmessungen direkt auf dem Raspberry Pi übernommen. Die neuen Geschwindigkeitseinstellungen werden per SPI zum STM32 gesendet und dort übernommen.

8.4.2.4 Einstellungen zurücksetzen

Listing 41: Einstellungen zurücksetzen

```
1 socket.on('resetSettings', function () {
2     phimax = 20 * Math.PI / 180;
3     spi.defaultSettings();
4     temperatureController.defaultSettings();
5 });
```

Die Funktion wird aufgerufen, sobald der `Default` Button auf der graphischen Oberfläche betätigt wird.
Auf dem Server sind die Standardwerte definiert und werden sowohl von den Clients als auch von der Hardware direkt übernommen.

8.4.2.5 Zeichnung erstellen

Listing 42: create Funktion für neue Zeichnungen

```
1  //creates a new DB if the client requests it
2  socket.on('create new db', function (data) {
3      database.createNew(data.name);
4      selectedDB = data.name;
5      deletedDB = false;
6      socket.broadcast.emit('change selection', data);
7      io.sockets.emit('sheets', database.allDatabases);
8      lineID = 0;
9      out = [];
10     spiout = [];
11 });
```

Sobald ein Client eine neue Zeichnung erstellt, wird der Name der neuen Zeichnung dem Server übermittelt. Der Server erstellt in der MongoDB eine neue Collection mit dem Namen, welche vom Client übermittelt wurde.
Anschließend wird allen anderen Clients mitgeteilt, dass es eine neue Zeichnung gibt. Weiters wird die neue Zeichnung ausgewählt. Da in einer neuen Zeichnung noch keine Daten vorhanden sein können, werden bestimmte Variablen(`lineID, out, ...`) zurückgesetzt.

8.4.2.6 Zeichnung auswählen

Listing 43: alte Zeichnung laden

```
1  socket.on('choose sheet', function (data) {
2      outDB = [];
3      out = [];
4      spiout = [];
5      lineID = 0;
6      loadDB(data.name);
7      selectedDB = data.name;
8      deletedDB = false;
9      socket.broadcast.emit('change selection', data);
10
11     database.load_drawing(function (error, data) {
12     //reads the coords from the database and sorts them
13         for (var i = 0; i < data.length; i++) {
14             var lID = data[i].line,
15                 pID = data[i].pID
16         [...]
17             outDB[lID][pID] = {
18                 'x':data[i].x,
19                 'y':data[i].y
20             }
21             calculation(data[i].x, data[i].y, lID, pID);
22         }
23         var spioutCache = spiout;
24         for (var i = 0; i < outDB.length; i += 2) {
25             out[i] = outDB[i];
26             if (i >= 2)
27                 bezierBetweenLines();
28         }
29         [...]
30
31         lastFirstBezier();
```

```
32          io.sockets.emit('load old drawing', out);
33          sendCoordsToStm(spiout);
34      });
35  });
```

Auch hier müssen die Variablen auf ihren Standardwert zurückgesetzt werde, damit die Daten der neuen Zeichnung neu aufgenommen werden können.
Sobald ein Client eine neue Zeichnung auswählt, wird mithilfe des `socket.broadcast.emit` allen anderen Clients mitgeteilt, welche Zeichnung ausgewählt wurde.
Anschließend werden die Punkte von der Datenbank geladen. Da MongoDB die Daten nicht in einer bestimmten Reihenfolge herausliest, werden die Koordinaten in einem zweidimensionalen Array nach der `lineID` und `punktID` sortiert. Direkt während dem Sortiervorgang werden die einzelnen Punkte mithilfe einer eigens für diesen Zweck entwickelten Formel in einen Zahlenbereich von 0 bis 4095 gebracht und im Array `spiout` abgespeichert. Der Zahlenbereich von 0 bis 4095 ist wichtig, da dieser Bereich vom DAC benutzt wird.
Bei mehr als 2 vorhandenen Linien wird die `bezierBetweenLines` Funktion aufgerufen, um für den Laser eine möglichst einfache Strecke von Linie zu Linie zu berechnen. Die Bezierstrecken zwischen den gezeichneten Linien werden auch im Array `spiout` abgespeichert.
Zum Schluss wird die lineID des Servers aktualisiert und die Verbindungsstrecke zwischen der letzten und der ersten Linie berechnet. Sobald alle Berechnungen fertig sind, wird das `out` Array zu den Clients und der `spiout` Array zum STM32 gesendet.

8.4.2.7 Zeichnung löschen

Listing 44: Zeichnung löschen

```
1  socket.on('delete', function (data) {
2      deleteDB();
3      deleteFunction(out, data.x1, data.x2, data.y1, data.y2);
4
5      if (data.status == 'stopDeleting') {
6          io.sockets.emit('clear');
7          io.sockets.emit('load old drawing', out);
8          deletedDB = true;
9          sendCoordsToStm(spiout);
10         for (var j = 0; j < out.length; j += 2) {
11             saveCoords(out[j], j);
12         }
13     }
14 });
```

Sobald ein Client ein Löschen der aktuellen Zeichnung anfordert, wird der Befehl zur MongoDB weitergeleitet, damit diese anschließend die Datenbank löscht. Gleichzeitig wird eine Funktion aufgerufen, die alle Variablen zurücksetzt und jedem Client mitteilt, dass die aktuelle Datenbank gelöscht wurde und somit die Zeichenfläche gelöscht werden kann.

8.4.2.8 Beginn einer Linie

Listing 45: startDrawing

```
1  //if the client started drawing save the received data in firstcoords
2  socket.on('startDrawing', function (prev) {
3      drawing = true;
4      socket.lineID = lineID;
5      lineID += 2;
```

```
6       socket.pID = 0;
7       socket.i = 0;
8
9       firstCoords = {
10          'x':prev.x,
11          'y':prev.y,
12          'pID':socket.pID
13      }
14
15      if (coordBuff.length == 0)
16          coordBuff[0] = firstCoords;
17
18      out[socket.lineID] = [];
19
20      out[socket.lineID][socket.pID] = {
21          'x':firstCoords.x,
22          'y':firstCoords.y
23      };
24
25      spiout[socket.lineID] = [];
26      calculation(out[socket.lineID][socket.pID].x, out[socket.lineID][socket.pID↙
            ↳ ].y, socket.lineID, socket.pID);
27
28      pointsOfNewLine = 1;
29
30      socket.pID++;
31      socket.i++;
32  });
```

Wenn ein Client beginnt eine Linie zu zeichnen, werden die ersten Koordinaten zum Server gesendet. Dieser setzt die benötigten Variablen auf ihre Standardwerte zurück und gibt der neuen Linie die nächste freie `lineID` und dem ersten Koordinatenpaar die erste pID.
Daraufhin wird die neue Linie im `out` Array gespeichert. Zusätzlich zum `out` Array wird der Punkt auch im `CoordBuff` an der ersten Stelle (`CoordBuff[0]`) gespeichert. CoordBuff wird später für das Abspeichern einer kompletten Linie in die Datenbank benötigt. Für genaueres über CoordBuff siehe Kapitel 8.4.2.11 Ende einer Linie.
Mithilfe der `calculation` Funktion wird der neue Punkt in einen für den DAC brauchbaren Bereich gewandelt und im `spi` Array abgespeichert.
Zum Schluss wird die `pID` erhöht, damit die nächsten Koordinaten richtig in das Array einsortiert werden.

8.4.2.9 Zeichnen

Listing 46: mousemove

```
1  //saves the received frames
2  socket.on('mousemove', function (data) {
3          if (drawing) {
4              for (var j = 0; j < data.length; j++) {
5                  data[j].pID = socket.pID;
6                  coordBuff[socket.i] = data[j];
7
8                  out[socket.lineID][socket.pID] = {
9                      'x':data[j].x,
10                     'y':data[j].y
11                 };
12                 calculation(out[socket.lineID][socket.pID].x, out[socket.lineID↙
                       ↳ ][socket.pID].y, socket.lineID, socket.pID);
13
14                 socket.i++;
15                 socket.pID++;
16
17                 if ((pointsOfNewLine > 2) && (out.length > 2)) {
18                     bezierBetweenLines();
19                     pointsOfNewLine=0;
20                     io.sockets.emit('load old drawing', out);
21                 }
22                 if (pointsOfNewLine != 0)
23                     pointsOfNewLine++;
24             }
25             coordCount++;
26             if (coordCount >= 5) {
27                 if (out.length > 2) {
28                     coordCount = 0;
29                 }
30                 lastFirstBezier();
31                 //sends the frames all n points to the spi
32                 sendCoordsToStm(spiout);
33             }
34         }
35         else {
36             socket.emit('load old drawing', out);
37         }
38         //broadcasts every coordinate to the other frames for synchronized ↙
               ↳ drawing
39         socket.broadcast.emit('moving', data);
40     }
41 );
```

Die `socket.on('mousemove', ...)` Funktion ruft, wenn die Maus noch nicht losgelassen wurde, also wenn `drawing` zutrifft und nach allen fünf Punkten eine Funktion auf, welche die Punkte speichert. Dabei wird überprüft, ob es sich um eine neue Linie handelt oder nicht. Wenn es sich um eine neue Linie handelt, ist die Variable `pointsOfNewLine` nicht 0. In diesem Fall wird nach dem dritten Punkt die Funktion `bezierBetweenLines` aufgerufen. Nachdem alle Punkte abgespeichert und die `lastFirstBezier` berechnet wurde, werden alle Daten über die SPI an den STM32 gesendet. Dadurch ist es möglich, annähernd live zu zeichnen.

8.4.2.10 Radieren

Listing 47: erase

```
socket.on('delete', database.removeDB);
socket.on('delete', function (data) {
    deleteFunction(out, data.x1, data.x2, data.y1, data.y2);

    if (data.status == 'stopDeleting') {
        io.sockets.emit('clear');
        io.sockets.emit('load old drawing', out);
        deletedDB = true;
        sendCoordsToStm(spiout);
        for (var j = 0; j < out.length; j += 2) {
            saveCoords(out[j], j);
        }
    }
});
```

Sobald ein Client etwas löscht, wird die ganze Datenbank gelöscht. Anschließend werden die zu löschenden Punkte in den Arrays gelöscht. Dies geschieht in der `deleteFunction`. Diese wird weiter unten erklärt. Erst wenn der Client aufhört zu löschen, werden die sichtbaren Linien abgespeichert und auch die Punkte über die SPI gesendet und neu gezeichnet.

Löschfunktion

Listing 48: deleteFunction

```
var deleteFunction = function (data, x1, x2, y1, y2) {
    var datalength = data.length;
    var newLineID = data.length + 1;

    for (var j = 0; j < datalength; j++) {
        if (j > 0)
            data[j - 1] = null;

        if (j % 2 == 1)
            data[j] = null;

        else {
            var newPID = 0;
            var numberOfPoints = data[j].length;
            for (var i = 0; i < numberOfPoints; i++) {
                //delete the point
                if ((data[j][i].x > x1) && (data[j][i].x < x2) && (data[j][i].y
                    ↪  > y1) && (data[j][i].y < y2)) {
                    data[j][i] = null;
                    newPID = 0;
                }

                else {
                    // save the point with a new LineID and a new PID
                    if (newPID == 0) {
                        newLineID = newLineID + 1;
                        data[newLineID] = [];
                        spiout[newLineID] = [];
                    }
                    data[newLineID][newPID] = data[j][i];

```

```
31                      data[newLineID][newPID].pID = newPID;
32                      spiout[newLineID][newPID] = spiout[j][i];
33
34                      newPID++;
35                  }
36              }
37          }
38      }
39
40      var klID = 0;
41      var spioutCache = spiout;
42      out = [];
43      spiout = [];
44      //sort the lines so that there are no lines with less than 2 points
45      for (var i = 0; i < data.length; i++) {
46
47          if (data[i] && data[i] && data[i].length > 1) {
48              out[klID] = data[i];
49              spiout[klID] = spioutCache[i];
50
51              if (klID >= 2)
52                  bezierBetweenLines();
53              klID += 2;
54          }
55      }
56
57      if (out.length == 0)
58          lineID = 0;
59      else
60          lineID = out.length + 1;
61
62      lastFirstBezier();
63  };
```

Die `deleteFunction` überprüft alle Punkte, ob sie im zu löschenden Bereich, bei welchem es sich um ein Rechteck handelt, liegen oder nicht. Dabei werden alle bereits überprüften Linien gelöscht, da die bestehenden Linien nach dem Löschen als neue Linien abgespeichert werden. Anschließend wird das `data`-Array in das `out`-Array geschrieben. Dabei werden jedoch leere Linien bzw. Linien mit weniger als 2 Punkten nicht berücksichtigt. Gleichzeitig werden auch die Bezierkurven neu berechnet.

8.4.2.11 Ende einer Linie

Listing 49: stopDrawing

```
1  //if the client stopped drawing save the last coordinates and send the complete array to the database
2  socket.on('stopDrawing', function (last) {
3      if (drawing) {
4          drawing = false;
5
6          lastCoords = {
7              'x':last.x,
8              'y':last.y,
9              'pID':socket.pID
10         }
11         if (coordBuff[coordBuff.length - 1].x != lastCoords.x &&
12             coordBuff[coordBuff.length - 1].y != lastCoords.y) {
13             coordBuff[coordBuff.length] = lastCoords;
14             out[socket.lineID][lastCoords.pID] = {
15                 'x':lastCoords.x,
16                 'y':lastCoords.y
17             };
18             calculation(out[socket.lineID][socket.pID].x, out[socket.lineID][socket.pID].y, socket.lineID, socket.pID);
19         }
20         if ((coordBuff.length < 2)) {
21             lineID -= 2;
22         }
23         else {
24             var coordArray = new cloneObject(coordBuff);
25             coordArray.length = coordBuff.length;
26             saveCoords(coordArray, socket.lineID);
27
28             if (pointsOfNewLine != 0 && out.length > 2) {
29                 bezierBetweenLines();
30                 pointsOfNewLine = 0;
31             }
32             lastFirstBezier();
33             sendCoordsToStm(spiout);
34         }
35         socket.i = 0;
36         socket.pID = 0;
37         coordBuff.length = 0;
38         io.sockets.emit('clear');
39         io.sockets.emit('load old drawing', out);
40     }
41 });
```

Die `socket.on('stopDrawing', ...)` Funktion wird aufgerufen, wenn ein Client eine Linie fertig gezeichnet hat, das heißt er hat die Maustaste losgelassen.
Falls `drawing = true` ist wird überprüft, ob das letzte Koordinatenpaar bereits im `coordBuff` Array an letzter Stelle steht, falls das Koordinatenpaar nicht vorhanden ist, wird es im `out` Array sowie im `coordBuff` Array abgespeichert. Nun wird mit `calculation` die Berechnung durchgeführt, damit der STM32 die Daten weiter zu den Galvanometer schicken kann.
Wenn die aktuelle Linie mehr als 2 Punkte besitzt, wird der `coordBuff` Array zusammen mit der `socket.lineID` der Datenbank übergeben um die neue Linie zu speichern. Abschließend werden die Bezierlinien berechnet, gebrauchte Variablen zurückgesetzt und jedem Client das neue Bild, sprich das `out` Array übermittelt.

8.4.2.12 DAC-Werte der Koordinaten berechnen

Für den DAC des STM's müssen die Werte im Bereich von 0 bis 4095 liegen. Wie in Kapitel 6.8.1 Projektion der Koordinaten bereits erklärt wurde, treten sphärische Verzerrungen auf, die mit einer Berechnung ausgeglichen werden.

Listing 50: calculation

```
var calculation = function (x, y, lineID, PID) {
    var result = {x:Math.round(2047 * (1 + (Math.atan((2 * x - 1) * scale)) / ↙
        ↳ phimax)),
        y:Math.round(2047 * (1 + (Math.atan((2 * y - 1) * scale)) / phimax))};

    spiout[lineID][PID] = {
        'x':result.x,
        'y':result.y
    };
};
```

Diese Berechnung erfolgt für jeden Punkt. Die berechneten Werte werden in das `spiout`-Array gespeichert, welches an den STM gesendet wird.

8.4.2.13 Koordinaten über SPI senden

Listing 51: sendCoordsToStm

```
var sendCoordsToStm = function (data) {
    var datalength = 0;

    for (var j = 0; j < data.length; j++) {
        datalength += data[j].length;
    }

    if (datalength <= 4000)
        spi.spi_send(data);
    else
        socket.emit('WarnMessage', 'More than 4000 Points!!!');
};
```

Da der STM32 nur einen begrenzten Speicher hat, ist es derzeit nicht möglich, eine Zeichnung mit mehr als 4000 Punkten an den STM32 zu senden. Daher wird beim Erreichen von 4000 Punkten eine Warnung an den gerade zeichnenden Client gesendet.

8.4.2.14 Optimierung

Insbesondere wenn Linien oder Teile einer Linie gelöscht werden, ist es wichtig, dass die sichtbaren Linien so angeordnet werden, dass zwischen diesen ein möglichst kurzer Abstand ist und somit Weg eingespart werden kann.

Für diese Sortierung wurde das Prinzip Nearest Neighbor[26] angewendet. Dabei wird zwischen verschiedenen Punkten ein Punkt als Startpunkt gewählt und von diesem aus der Punkt gesucht, der am nächsten ist. Anschließend wird vom gefundenen Punkt aus der nächste gesucht. Dies wird sooft wiederholt bis alle Punkte einmal ausgewählt wurden.

[26] http://de.wikipedia.org/wiki/Nearest-Neighbor-Heuristik

Jedoch muss berücksichtigt werden, dass es sich bei diesem Prinzip nur um Punkte handelt, die miteinander verbunden werden. Da es sich in dieser Anwendung jedoch um Linien handelt, müssen sowohl der Anfangs- als auch der Endpunkt überprüft und wenn notwendig die Linie umgedreht werden.

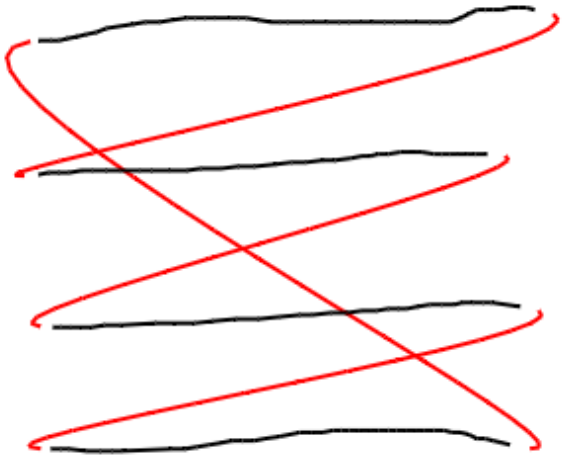


Abbildung 97: Ohne Optimierung

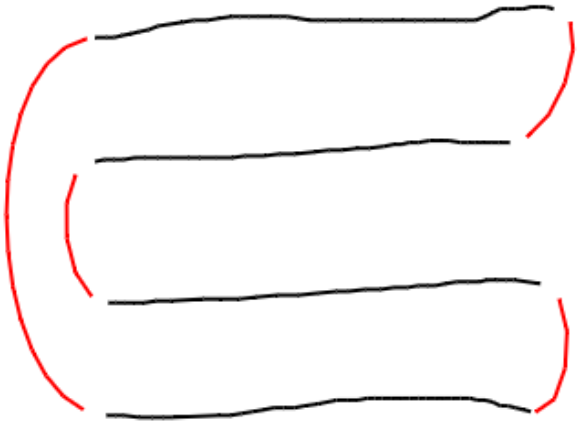


Abbildung 98: Mit Optimierung

Aus den Abbildungen wird ersichtlich, dass durch die Optimierung erheblich an Weg eingespart werden kann.

Listing 52: nearestNeighbour

```
for (var j = 0; j < outCache.length - 2; j += 2) {
        for (var i = 2; i < outCache.length; i += 2) {

            if (outCache[i].length > 1) {

                var distbetweenLines = Math.sqrt(Math.pow(out[j][out[j].length ↙
                    ↳ - 1].x - outCache[i][0].x, 2) +
                    Math.pow(out[j][out[j].length - 1].y - outCache[i][0].y, 2)↙
                        ↳ );
```

```
8
9                   if (distbetweenLines < nextLine[0][0].distance) {
10                      nextLine[0][0].distance = distbetweenLines;
11                      nextLine[0][0].lID = i;
12                      nextLine[0][0].pIDStart = 0;
13                  }
14
15                  distbetweenLines = Math.sqrt(Math.pow(out[j][out[j].length - ↙
                        ↳ 1].x - outCache[i][outCache[i].length - 1].x, 2) +
16                      Math.pow(out[j][out[j].length - 1].y - outCache[i][outCache↙
                            ↳ [i].length - 1].y, 2));
17
18                  if (distbetweenLines < nextLine[0][0].distance) {
19                      nextLine[0][0].distance = distbetweenLines;
20                      nextLine[0][0].lID = i;
21                      nextLine[0][0].pIDStart = outCache[i].length;
22                  }
23              }
24          }
25
26          if (nextLine[0][0].pIDStart != 0) {
27              out[j + 2] = [];
28              spiout[j + 2] = [];
29              for (var a = 0; a < outCache[nextLine[0][0].lID].length; a++) {
30                  out[j + 2][a] = outCache[nextLine[0][0].lID][outCache[nextLine↙
                        ↳ [0][0].lID].length - 1 - a];
31                  out[j + 2][a].pID = a;
32                  spiout[j + 2][a] = spioutCache[nextLine[0][0].lID][spioutCache[↙
                        ↳ nextLine[0][0].lID].length - 1 - a];
33              }
34          }
35
36          else {
37              out[j + 2] = outCache[nextLine[0][0].lID];
38              spiout[j + 2] = spioutCache[nextLine[0][0].lID];
39          }
40
41          outCache[nextLine[0][0].lID] = [];
42          nextLine[0][0].distance = 2;
43  }
```

Der Optimierungsalgorithmus besteht im wesentlichen aus zwei `for`-Schleifen. Mit der ersten wird die zuletzt einsortierte Linie ausgewählt. Anschließend werden mit der zweiten Schleife alle noch verbliebenen Linien durchgegangen und die Linie mit der kürzesten Distanz zur zuletzt einsortierten Linie gesucht. Wurde diese gefunden, wird sie zu den sortierten Linien hinzugefügt und im Array mit den zu sortierenden Linien gelöscht.

8.4.3 Datenbanklogik: mongodb.js

Alle Lese- und Schreibvorgänge mit der Datenbank MongoDB erfolgen in NodeJS asynchron. Das heißt, das NodeJS-Programm läuft während einem Datenbankzugriff weiter. Für MongoDB gibt es, wie bei gewöhnlichen SQL Datenbanken, einige Kommandos[5], um bestimmte Datensätze zu erhalten.

8.4.3.1 Verbindung mit MongoDB-Server

Beim Start des NodeJS-Programms wird eine Verbindung mit der MongoDB aufgebaut. Mit `mongo.connect` wird eine Verbindung mit dem MongoDB-Server aufgebaut, der `localhost` als Server und den Standardport von MongoDB verwendet, welcher 27017 ist. `local` gibt die zu verwendende Datenbank an. In der Datenbank `local` befinden sich die einzelnen als `collection` abgespeicherten Zeichnungen sowie die `settings`, welche sich auch in einer Collection befinden.

Listing 53: MongoDB Verbindung

```
1  //connects with the Database
2  exports.connect = function () {
3      mongo.connect("mongodb://localhost/local", function (err, db) {
4          mongodb = db;
5          settings = mongodb.collection('settings');
6
7          return listAllDatabases();
8      });
9  };
```

Um nun den Clients eine Liste von allen gespeicherten Zeichnungen zu übermitteln, werden die Namen aller Collections in ein Array gespeichert.

Listing 54: Vorhandene Zeichnungen abfragen

```
1  //return a list of alle Collection names
2  var listAllDatabases = function () {
3      mongodb.collections(function (err, collections) {
4          exports.allDatabases = [];
5          for (var i = 0; i < collections.length; i++) {
6              exports.allDatabases[i] = {name: collections[i].collectionName};
7          }
8          if (socketIO.updateDatabases)
9              socketIO.updateDatabases();
10     });
11 };
```

Da diese Funktion auch beim Löschen einer Datenbank aufgerufen wird und die Datenbankliste direkt an alle Clients sendet, wird auch abgefragt, ob zumindest ein Client verbunden ist. Falls ein Client verbunden ist, wird seine Datenbankliste aktualisiert.

8.4.3.2 Erstellen einer Zeichnung

Eine neue Collection kann mit `mongodb.collection(name)` angelegt werden. Die Collection wird jedoch erst beim Speichern von Koordinaten endgültig erstellt.

Listing 55: Koordinaten abspeichern

```
exports.createNew = function (name) {
    exports.allDatabases[exports.allDatabases.length] = {'name': name};
    collection = mongodb.collection(name);
};
```

8.4.3.3 Koordinaten abspeichern

Da das Abspeichern der Koordinaten asynchron erfolgt, werden Callbacks verwendet. Das heißt, wenn die Datenbank einen Punkt abgespeichert hat, wird eine Funktion aufgerufen, in welcher steht, was nach dem Abspeichern geschehen soll.

Listing 56: Koordinaten abspeichern

```
//sends the coordinateS to the savecoordinate function
exports.saveCoords = function (data, lineID, deletedDB) {
    var index = 0,
        callback = function (data) {
            index++;
            if (index < data.length) {
                saveCoord(data[index], callback, lineID);
            }
            else if(deletedDB)
                listAllDatabases();
        },
    //saves each coordinate
        saveCoord = function (coord, callback, lineID) {
            var object = {pID: coord.pID, x: coord.x, y: coord.y, line: lineID↙
                ↳ };
            collection.insert(object, {safe: true}, function (err, result) {
                if (err)
                    console.log('Insert coordinate error!');
                return callback(data);
            });
        };

    saveCoord(data[index], callback, lineID);
};
```

Der `saveCoords`-Funktion kann eine ganze Linie zum Abspeichern übergeben werden. Jedesmal nach dem Abspeichern einer Koordinate wird die Funktion `callback` aufgerufen, welche die Funktion `saveCoord` aufruft. Dies wird wiederholt bis alle Koordinaten abgespeichert wurden.

8.4.3.4 Laden einer Zeichnung

Um eine Datenbank zu laden, muss zuerst die gewünschte Collection ausgewählt werden.

Listing 57: Auswählen der Collection

```
//load an old collection
exports.load = function (name) {
```

```
3       collection = mongodb.collection(name);
4   };
```

Anschließend müssen alle Einträge, die dieser Collection zugeordnet sind, geladen und in ein Array gespeichert werden. Nachdem alle Daten geladen sind, wird der `callback` ausgelöst.

Listing 58: Koordinaten in ein Array Speichern

```
1   //returns all coordinates of a collection
2   exports.load_drawing = function (callBack) {
3       collection.find().toArray(callBack);
4   };
```

8.4.3.5 Löschen einer Zeichnung

Um eine Collection zu löschen, reicht es nicht, wenn nur die Collection gelöscht wird. Es muss auch der Index der Collection, bei dem es sich um den `collectionName` handelt, gelöscht werden.

Listing 59: Auswählen der Collection

```
1   //drops the collection
2   exports.removeDB = function () {
3       collection.drop(function (err, replies) {
4           collection.dropIndex(collection.collectionName, function (err, replies)↙
                ↳  {
5               listAllDatabases();
6           });
7       });
8   };
```

8.4.4 SPI-Übertragung: spi.js

Mit dem Package `pi-spi`[27] aus dem offiziellen Repository für NodeJS kann mit NodeJS auf die SPI-Schnittstelle des Raspberry Pi zugegriffen werden. Alle SPI-Zugriffe erfolgen asynchron. Das heißt, dass das Nodeprogramm weiterläuft, während die Daten über die SPI-Schnittstelle gesendet werden. Der Raspberry Pi übernimmt die Rolle des Masters, während der STM32 der Slave ist.

Das `pi-spi` Package verfügt über eine `transfer`-Funktion, welche gleichzeitig lesen und schreiben kann. Weiters kann die SPI-Schnittstelle initialisiert und die Geschwindigkeit, in welcher die Daten gesendet werden sollen, festgelegt werden.

Listing 60: Initialisierung der SPI

```
1  var SPI,
2      spi,
3      clockspeed = 5e6;
4
5  //Initialisation funktion for the SPI
6  exports.init = function (tmp) {
7      SPI = tmp;
8
9      //spi device 0.0 there is also a second channel (/dev/spidev0.1)
10     spi = SPI.initialize("/dev/spidev0.0");
11     spi.clockSpeed(clockspeed);
12     spi.dataMode(SPI.mode.CPHA | SPI.mode.CPOL);
13 };
```

8.4.4.1 SPI-Übertragungsprotokoll

Für die SPI-Übertragung wird ein eigens erstelltes Übertragungsprotokoll angewendet. So werden immer 32Bit gesendet. Dabei wird zwischen Command und Koordinaten unterschieden.

Command

Mit Commands teilt der Raspberry Pi dem STM32 mit, was er als nächstes machen will. Mit einem Command kann er beispielsweise mitteilen, von welchem Sensor er gerne die Temperatur haben möchte oder, dass er nun Koordinaten senden möchte.

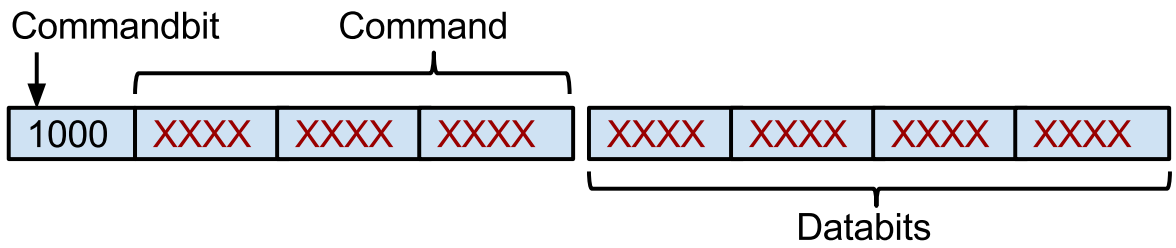


Abbildung 99: Übertragungsprotokoll

[27] https://www.npmjs.org/package/pi-spi

Bei einem Command ist das 31. Bit logisch eins. Werden zum Comamand noch Daten gesendet, wie z.B. die Anzahl der Koordinaten, stehen dafür Bit 0 bis 15 zur Verfügung.

Listing 61: Commands

```
var sendFrame = 0x0001,        //moegliche Argumente: Anzahl der Koordinaten
    getTemperature = 0x0002,       //Sensor
    setSpeed = 0x0003,             //neue Geschwindigkeit
    deleteFrame = 0x0004,          //welches Frame geloescht werden soll
    shutdown = 0x0005;             //welches Geraet ausgeschalten werden soll
```

Die wichtigsten Commands sind sendFrame und getTemperature.

Listing 62: Command über SPI senden

```
var spi_command = function (commandType, commandData, endOfWriting) {

    writingBuffer.writeUInt16BE(command + commandType, 0);
    writingBuffer.writeUInt16BE(commandData, 2);

    spi.transfer(writingBuffer, 4, function (e, d) {
        if (e)
            console.error(e);
    });

    if (endOfWriting)
        spiIsLocked = 0;
};
```

Es gibt auch Commands, die keine weitere Datenübertragung starten, wie zum Beispiel `setSpeed`. In diesem Fall wird die SPI-Schnittstelle nach dem Senden des Commands gleich wieder freigegeben.

Koordinaten

Für die Übertragung von Koordinaten wird das Commandbit auf 0 gesetzt. Das 30. Bit gibt in diesem Fall an, ob der Laser bei diesem Punkt eingeschaltet ist oder nicht. Für die X- und Y-Koordinate werden jeweils 12 Bits verwendet.

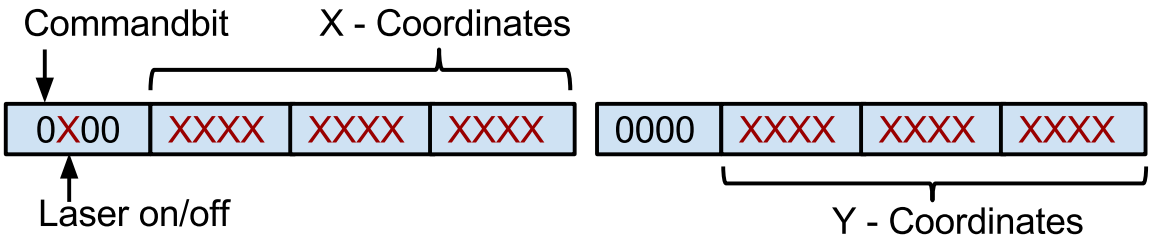


Abbildung 100: Koordinatenprotokoll

Alle Bits, die in Abbildung 100 nicht beschriftet sind, sind noch nicht belegt, jedoch für eventuelle Erweiterungen reserviert.

8.4.4.2 Controller

Die zentrale Steuerung der SPI übernimmt die Controller-Unit. Diese besteht aus einer Routine, die jede Millisekunde aufgerufen wird und überprüft, ob die SPI-Schnittstelle gesperrt ist oder nicht. Ist dies nicht der Fall, wird aus einem Array, in dem alle anstehenden Befehle stehen, der nächste Eintrag entnommen und ausgeführt. Dabei wird das Prinzip des FIFO angewendet.

Listing 63: Controller

```
var controllerBuffer = [];

var spiControllerCallback = function () {
    if (!spiIsLocked) {
        var spiCb = controllerBuffer.shift();
        if (spiCb) {
            spiIsLocked = 1;
            spiCb.func.apply(null, spiCb.arg);
        }
    }
}

var spiController = setInterval(spiControllerCallback, 1);
```

Im `controllerBuffer` werden die aufzurufende Funktion sowie deren Argumente gespeichert. Falls eine Funktion aufgerufen wird, wird die Controllerschleife gesperrt. Das heißt, `spiIsLocked` wird auf 1 gesetzt und erst in der jeweiligen Funktion, die aufgerufen wird, auf 0 gesetzt um die Controllerschleife wieder freizugegeben.

Listing 64: Befehle die in den controllerBuffer gespeichert werden

```
exports.read_temperature = function (device) {
    controllerBuffer.push({ func:read_temperature_controller, arg:[device] });
};

exports.setSpeed = function (speed) {
    console.log('Changed Speed of Laser to ' + speed);
    controllerBuffer.push({ func:spi_command, arg:[setSpeed, speed, ↙
        ↳ spiStopWorking]});
};

exports.defaultSettings = function () {
    console.log('Changed Settings to default!');
    controllerBuffer.push({ func:spi_command, arg:[setSpeed, defaultSpeed, ↙
        ↳ spiStopWorking] });
};

exports.spi_send = function (data) {
    controllerBuffer.push({ func:spi_send_controller, arg:[data] });
};
```

8.4.4.3 Koordinaten senden

Bevor die Daten gesendet werden, wird ein Command gesendet. Dieser gibt an, dass nun Koordinaten gesendet werden und die Anzahl der Koordinaten. Folgendes Beispiel zeigt, wie ein Command für das Senden von Koordinaten aussieht, wenn 100 Koordinaten übertragen werden.

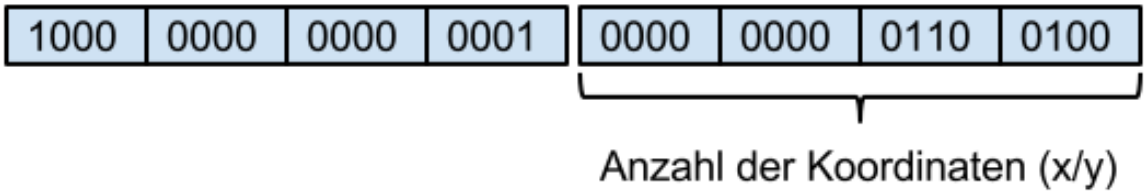


Abbildung 101: Koordinaten Command

Die Anzahl der Koordinaten wird in einer for-Schleife in der `spi_send_controller` berechnet.

Listing 65: Phasenverschiebung

```
var spi_send_controller = function (data) {
    var dataout, laser;
    var datalength = 0;

    for (var j = 0; j < data.length; j++) {
        datalength += data[j].length;
    }

    spi_command(sendFrame, datalength);
    console.log("start sending");

    var paint = true,
        filled = phase,
        lastLine = data.length - 1,
        paintBuf = [],
        actualPaint;

    for (var j = 0; j <= lastLine; j++) {
        max = data[j].length;
        count = max >= phase ? phase : max;

        for (i = 0; i < count; i++) {
            paintBuf.push(paint);
        }
        filled -= count;
        if (filled <= 0) break;

        paint = !paint;
    }
    paint = false;
```

Da durch die Regler eine Phasenverschiebung entsteht, muss für das Ein- und Ausschalten des Lasers auch eine Phasenverschiebung eingebaut werden. Die Phasenverschiebung ist im Settingsmenü einstellbar. Dazu wird zuerst von den ersten Punkten die Information in ein FIFO gespeichert, ob der Laser in diesem Punkt eingeschaltet ist oder nicht. Beim Durchlauf durch die Sendeschleife, wird das zuerst gespeicherte Element des `paintBuf` geholt und dem zu sendenden Element zugewiesen. Die Information über den Laserzustand des zu sendenden Elements wiederum wird in den `paintBuf` gespeichert.

Listing 66: Koordinaten an STM32 senden

```
  for (var j = data.length - 1; j >= 0; j--) {

        for (var i = data[j].length - 1; i >= 0; i--) {
            paintBuf.push(paint)
            actualPaint = paintBuf.shift();

            if (actualPaint)
                laser = laseron;
            else
                laser = laseroff;

            //buffer for the x and y coordinates
            writingBuffer.writeUInt16BE(laser + data[j][i].x, 0);
            writingBuffer.writeUInt16BE(data[j][i].y, 2);
            dataout = writingBuffer;

            //send the data to the stm
             spi.transfer(dataout, 4, function (e, d) {
                if (e)
                    console.error(e);
             });
        }
        paint = !paint;
    }
    console.log(datalength + "Points sent");
    spiIsLocked = 0;
};
```

Am Ende des Sendevorgangs muss die SPI-Schnittstelle freigegeben werden.

8.4.4.4 Temperatur lesen

Die Temperatur wird mithilfe von Callbacks eingelesen. Dies ist nötig, da NodeJS asynchron arbeitet. Daher wird der Funktion `read_from_stm` ein callback mitgegeben, der eine Funktion angibt, die das Programm nach erfolgreichem Lesen von 32Bits abarbeiten soll.
Die `waitForTempData` Funktion wird dabei jedoch höchstens 50 mal aufgerufen, da danach davon ausgegangen werden kann, dass keine Temperatur mehr eingelesen werden kann.

Um die Temperatur zu erhalten, wird ein Command zum Lesen der Temperatur gesendet, der angibt, welcher Sensor ausgelesen werden soll. Der STM32 sendet nach dem Erhalt dieses Commands eine 1 und anschließend die Temperatur.

Listing 67: Temperatur lesen

```
var read_temperature_controller = function (tempSensor) {
    sensor = tempSensor;
    spi_command(getTemperature, tempSensor);
    read_from_stm(waitForTempData);
};

var waitForTempData = function (dataIn) {
    if (dataIn.readUInt32BE(0) == 1) {
        read_from_stm(processTemp);
        spiWaitforTemperatureCounter = 0;
    }
    else {
```

```
13          if (spiWaitforTemperatureCounter == 50) {
14              console.log('No Temperature could be read!!!');
15
16              spiIsLocked = 0;
17              return;
18          }
19          else {
20              spiWaitforTemperatureCounter++;
21              read_from_stm(waitForTempData);
22          }
23      }
24  };
25
26  var processTemp = function (dataIn) {
27      var sensorTemperature = Math.round(10 * dataIn.readUInt32BE(0) / 256) / 10;
28
29      console.log(sensor + ". Temperatur: " + sensorTemperature);
30      temperatureController.getTemperatureBack(sensorTemperature, sensor);
31      spiIsLocked = 0;
32  };
33
34  var read_from_stm = function (callback) {
35       spi.transfer(sendDataWhileReceiving, 4, function (e, data) {
36          if (e)
37              console.error(e);
38
39       return callback(data);
40       });
41  };
```

Wenn eine Temperatur gelesen werden konnte, wird sie im `deviceInformation`-Array, das sich in der Datei temperatureController.js befindet, gespeichert.

8.4.5 Temperaturüberwachung: temperatureController.js

Der Raspberry Pi holt in gewissen Zeitabständen die Temperaturen der einzelnen Sensoren vom STM32 und überprüft, ob die Temperaturen im zugelassenen Bereich sind. Ist dies nicht der Fall, wird eine Warnung an alle verbundenen User gesendet. Auch der Raspberry Pi wird abgeschaltet, falls der Temperatursensor meldet, dass er zu heiß ist.

Listing 68: Temperatur lesen und prüfen

```
exports.controlTemperature = function (connecting) {
    for (var i = 0; i < exports.deviceInformation.length; i++) {
        spi.read_temperature(i);
    }

    for (var i = 0; i < exports.deviceInformation.length; i++) {
        if (exports.deviceInformation[i].temperature > maxwert) {
            if (!exports.deviceInformation[i].warningsent || connecting) {
                if (socketIO.controlTemperatureWarning) {
                    socketIO.controlTemperatureWarning(exports.deviceInformation[i].name);
                    exports.deviceInformation[i].warningsent = true;
                }
                console.log(exports.deviceInformation[i].name);
            }

            if (!exports.deviceInformation[i].warningsent && exports.deviceInformation[i].name == 'Raspberry Pi') {

                exec = require('child_process').exec;

                exec('sudo poweroff',
                    function (error, stdout, stderr) {
                        console.log('stdout: ' + stdout);
                        console.log('stderr: ' + stderr);
                        if (error !== null) {
                            console.log('exec error: ' + error);
                        }
                    });
            }
        }

        else
            exports.deviceInformation[i].warningsent = false;
    }
};

var temperatureControllerInterval = setInterval(exports.controlTemperature, defaultReadTempTime);
```

Unter Settings kann der User den Zeitabstand angeben, in dem diese Überprüfung erfolgen soll. Standardmäßig beträgt dieser eine Sekunde.

Damit nicht jede Sekunde eine Warnmeldung an den User gesendet wird, wird abgespeichert, ob für diesen Sensor bereits eine Warnung gesendet worden ist. Es wird erst wieder eine Warnung an das Gerät geschickt, wenn die Temperatur zwischenzeitlich einmal unter 100°C gewesen ist.

8.4.6 Bezierkurve

Damit der Laser die unsichtbare Strecke zwischen den einzelnen Linien kontrolliert abfährt und keine Überschwinger entstehen, ist es nötig, einige Punkte zwischen diesen zu speichern. Dazu eignen sich Bezierkurven[28] hervorragend. In diesem Anwendungsfall werden kubische Bezierkurven verwendet. Für diese werden vier Punkte benötigt. Dabei werden zwei Punkte miteinander verbunden, während die anderen zwei nur Hilfspunkte sind, um eine interpolierte Linie zwischen den zwei zu verbindenden Punkten zu zeichnen.

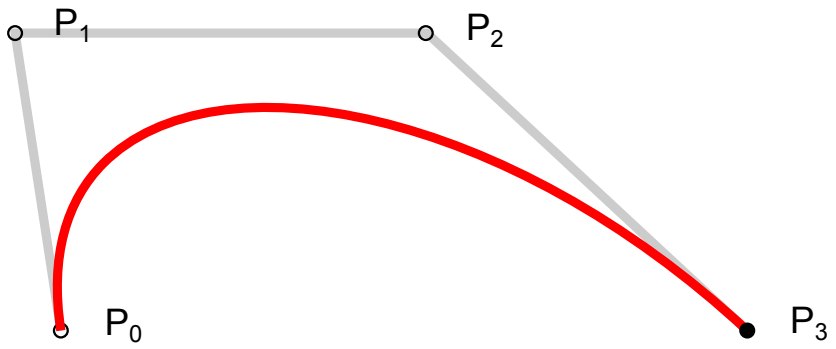


Abbildung 102: Kubische Bezierkurve

Die Punkte P1 und P2 sind die Hilfspunkte. Mit ihnen kann die Krümmung der Bezierkurve beeinflusst werden. Damit die Bezierkurve tangential in die sichtbaren Linien übergeht, werden P1 und P2, wie in Abbildung 103 gezeigt wird, berechnet.

[28] http://de.wikipedia.org/wiki/Bézierkurve

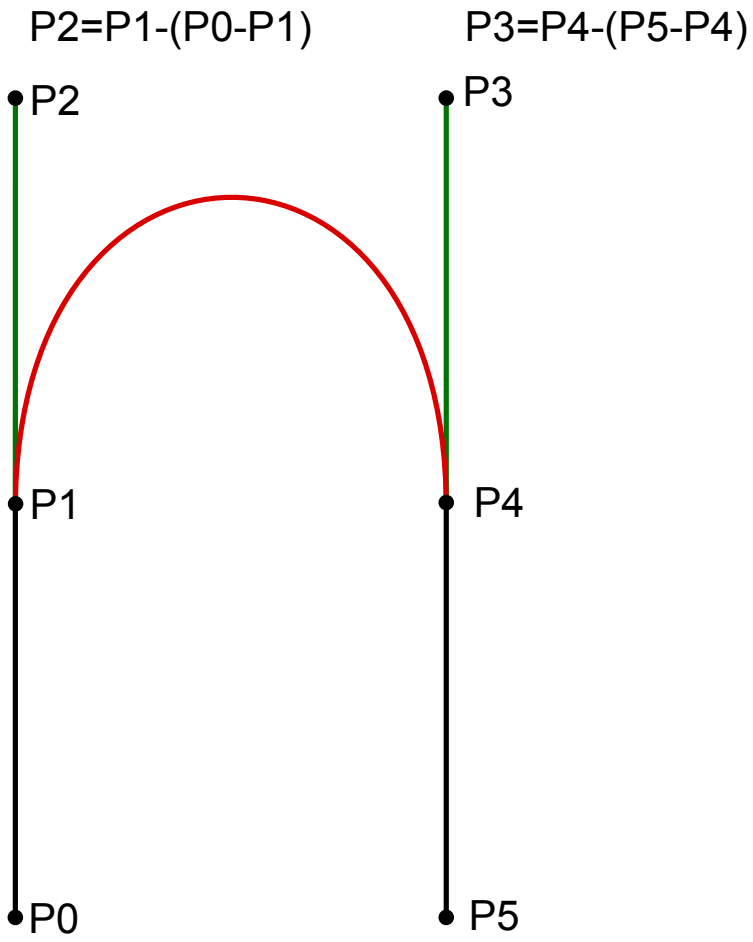


Abbildung 103: Berechnung von P2 und P3

Die Gerade zwischen P0 und P1 ist gleichlang wie die Gerade zwischen P2 und P1. Die Berechnung erfolgt für alle X- und Y-Punkte gleichermaßen. P0 und P5 sind im Normalfall nicht der zweitletzte bzw. der zweite Punkt der Gerade, sondern einige Punkte weiter entfernt von P1 bzw. P4. Dies wurde so gewählt, da ansonsten die Bezierkurve zum Teil fast rechtwinklig zur Gerade beginnt.

Durch die Berechnung von P2 und P3 kann jeder Punkt der Bezierkurve mit folgender Gleichung berechnet werden, wobei t von 0 bis 1 stetig erhöht wird:

$$P(t) = P1 \cdot (-t^3 + 3 \cdot t^2 - 3 \cdot t + 1) + 3 \cdot P2 \cdot t \cdot (t^2 - 2 \cdot t + 1) + 3 \cdot P3 \cdot t^2 \cdot (1 - t) + P4 \cdot t^3$$

Listing 69: Bezierkurve berechnen

```
1 var bezier = function (p1x, p0x, p5x, p4x, p1y, p0y, p5y, p4y) {
2     //calculate p2 and p3
3     var p2x = p1x - (p0x - p1x);
4     var p3x = p4x - (p5x - p4x);
5
```

```
6       var p2y = p1y - (p0y - p1y);
7       var p3y = p4y - (p5y - p4y);
8
9   // calculate the distance between the last and the second to last point
10      var distance1 = Math.sqrt(Math.pow(p1x - p2x, 2) + Math.pow(p1y - p2y, 2)),
11          distance2 = Math.sqrt(Math.pow(p4x - p3x, 2) + Math.pow(p4y - p3y, 2));
12
13      // calculate the distance between the two lines
14      var distbetweenLines = Math.sqrt(Math.pow(p4x - p1x, 2) + Math.pow(p4y - ↙
            ↳ p1y, 2));
15
16      if (distbetweenLines == 0)
17          distbetweenLines = 0.000000001;
18
19      // new calculation of p2 and p3 with vector of the length 1
20      p2x = (p2x - p1x) / distance1 * distbetweenLines / 3 + p1x;
21      p2y = (p2y - p1y) / distance1 * distbetweenLines / 3 + p1y;
22
23      p3x = (p3x - p4x) / distance2 * distbetweenLines / 3 + p4x;
24      p3y = (p3y - p4y) / distance2 * distbetweenLines / 3 + p4y;
25
26      var distbetweenPoints = 0.02;
27      // calculate number of points for Bezierline
28      var dt = 1 / (distbetweenLines / distbetweenPoints);
29
30      var buffer = new Array();
31      var count = 0;
32
33      for (var t = dt; t <= 1; t += dt) {
34          var x = p1x * (Math.pow(-t, 3) + 3 * Math.pow(t, 2) - 3 * t + 1)
35                  + 3 * p2x * t * (Math.pow(t, 2) - 2 * t + 1)
36                  + 3 * p3x * Math.pow(t, 2) * (1 - t) + p4x * Math.pow(t, 3),
37              y = p1y * (Math.pow(-t, 3) + 3 * Math.pow(t, 2) - 3 * t + 1)
38                  + 3 * p2y * t * (Math.pow(t, 2) - 2 * t + 1)
39                  + 3 * p3y * Math.pow(t, 2) * (1 - t) + p4y * Math.pow(t, 3);
40
41          if (y > 1)
42              y = 1;
43
44          if (x > 1)
45              x = 1;
46
47          if (y < 0)
48              y = 0;
49
50          if (x < 0)
51              x = 0;
52
53          buffer[count] = {
54              'x':x,
55              'y':y
56          };
57
58          count++;
59      }
60      return buffer;
61  }
```

Die Anzahl der berechneten Bezierpunkte hängt vom Abstand zwischen den einzelnen Linien ab. Je weiter diese auseinander sind, desto mehr Punkte sollen berechnet werden. Da die Koordinaten

des normierten Zeichenfenster im Bereich von 0 bis 1 sein müssen, muss nach der Berechnung überprüft werden, ob die berechneten Punkte in diesem Bereich liegen. Ist dies nicht der Fall, wird ihnen der nächstliegende Randpunkt zugewiesen.

Da die Punkte P2 und P3 je nach Abstand zwischen den sichtbaren Linien für eine optimale Verbindung unterschiedlich weit von P1 und P4 entfernt sein müssen, werden sie neu berechnet. Dazu wird zuerst der Einheitsvektor zwischen P1 und P2 ermittelt. Dieser wird mit einem Drittel des Abstandes zwischen den Linien multipliziert. Anschließend wird P2 neu berechnet. Dasselbe wird mit P3 auch durchgeführt.

8.4.6.1 Bezier zwischen den Linien

Die Funktion `bezierBetweenLines` berechnet die Bezierkurven zwischen den einzelnen Linien. Es gibt noch eine weitere Funktion deren Namen `FirstLastBezier` lautet. Auf diese wird später eingegangen, denn sie verbindet die 1. mit der letzten Linie aus dem Array.

Die sichtbaren Linien erhalten eine gerade und die unsichtbaren Bezierkurven eine ungerade Zahl als LineID. Für den Punkt P5 wird standardmäßig der 3. Punkt der Linie gewählt. Beim Punkt P0 funktioniert es gleichermaßen, jedoch wird nun der 3. letzte Punkt genommen. Besteht die Linie aus weniger als 3 Punkten wird die Variable `lengthOfFirstLine` bzw. `lengthOfLastLine` angepasst.

Listing 70: Bezierkurve zwischen den Linien

```
1  var bezierBetweenLines = function () {
2      var lengthOfFirstLine = 3;
3      var lengthOfLastLine = 3;
4
5      if ((out[out.length - 1].length) < 3)
6          lengthOfLastLine = out[out.length - 1].length;
7
8      if ((out[out.length - 3].length) < 3)
9          lengthOfFirstLine = out[out.length - 3].length;
10
11     var p4x = out[out.length - 1][0].x;
12     var p5x = out[out.length - 1][lengthOfLastLine - 1].x;
13     var p0x = out[out.length - 3][out[out.length - 3].length - ↙
           ↳ lengthOfFirstLine].x;
14     var p1x = out[out.length - 3][out[out.length - 3].length - 1].x;
15
16     var p4y = out[out.length - 1][0].y;
17     var p5y = out[out.length - 1][lengthOfLastLine - 1].y;
18     var p0y = out[out.length - 3][out[out.length - 3].length - ↙
           ↳ lengthOfFirstLine].y;
19     var p1y = out[out.length - 3][out[out.length - 3].length - 1].y;
20
21     var bez = bezier(p1x, p0x, p5x, p4x, p1y, p0y, p5y, p4y);
22
23     var placeToSave = 0;
24     if (out.length % 2 == 1) {
25         out[out.length - 2] = [];
26         spiout[out.length - 2] = [];
27         placeToSave = 2;
28     }
29     else {
30         out[out.length - 3] = [];
31         spiout[out.length - 3] = [];
```

```
32          placeToSave = 3
33      }
34      saveBezInArrays(bez, placeToSave);
35  }
```

Je nachdem, ob bereits eine Bezierkurve zwischen der 1. und letzten Linie besteht, hat das `out` Array eine gerade Anzahl an Einträgen oder nicht. Falls diese Bezierkurve noch nicht hinzugefügt worden ist, ist die Anzahl der Einträge ungerade und die gerade berechnete Bezierkurve wird an zweitletzter Stelle des `out`- und `spiout`-Arrays gespeichert. Ansonsten wird es an die drittletzte Stelle gespeichert.

8.4.6.2 Bezier zwischen erster und letzter Linie

Die Berechnung für die Bezierkurve zwischen der 1. und letzten Linie funktioniert analog zur `bezierBetweenLines`, nur dass die Kurve an der letzten Stelle der Arrays eingefügt wird.

Listing 71: Bezierkurve zwischen 1. und letzter Linie

```
1  var lastFirstBezier = function () {
2      if ((out[0] == null)) return;
3
4      if ((out.length % 2 == 1) || (out.length == 1)) {
5          out[out.length] = [];
6          spiout[out.length - 1] = [];
7      }
8      else {
9          out[out.length - 1] = [];
10         spiout[out.length - 1] = [];
11     }
12
13     var lengthOfFirstLine = 3;
14     var lengthOfLastLine = 3;
15
16     if ((out[out.length - 2].length - 1) < 3)
17         lengthOfLastLine = out[out.length - 2].length - 1;
18
19     if (out[0].length - 1 < 3)
20         lengthOfFirstLine = out[0].length - 1;
21
22     var p1x = out[out.length - 2][out[out.length - 2].length - 1].x;
23     var p0x = out[out.length - 2][out[out.length - 2].length - 1 - ↙
           ↳ lengthOfLastLine].x;
24     var p5x = out[0][lengthOfFirstLine].x;
25     var p4x = out[0][0].x;
26
27     var p1y = out[out.length - 2][out[out.length - 2].length - 1].y;
28     var p0y = out[out.length - 2][out[out.length - 2].length - 1 - ↙
           ↳ lengthOfLastLine].y;
29     var p5y = out[0][lengthOfFirstLine].y;
30     var p4y = out[0][0].y;
31
32     var bez = bezier(p1x, p0x, p5x, p4x, p1y, p0y, p5y, p4y);
33
34     saveBezInArrays(bez, 1);
35 }
```

Falls es nur eine Linie gibt, werden die beiden Endpunkte miteinander verbunden. Da die Funktion `lastFirstBezier` auch aufgerufen werden kann, wenn keine Linien existieren, wird am

Anfang überprüft, ob eine Linie existiert oder nicht. Wenn keine Linie existiert, wird die Funktion abgebrochen. Die `saveBezInArrays` speichert die Bezierkoordinaten im `out`-Array und führt die Berechnung für das `spiout`-Array durch.

8.4.6.3 Beispiel für Bezierkurve

In folgendem Beispiel, das mit unserem graphischen Userinterface erstellt wurde, ist eine sichtbare (schwarz) und eine interpolierte (rot) Linie ersichtlich.

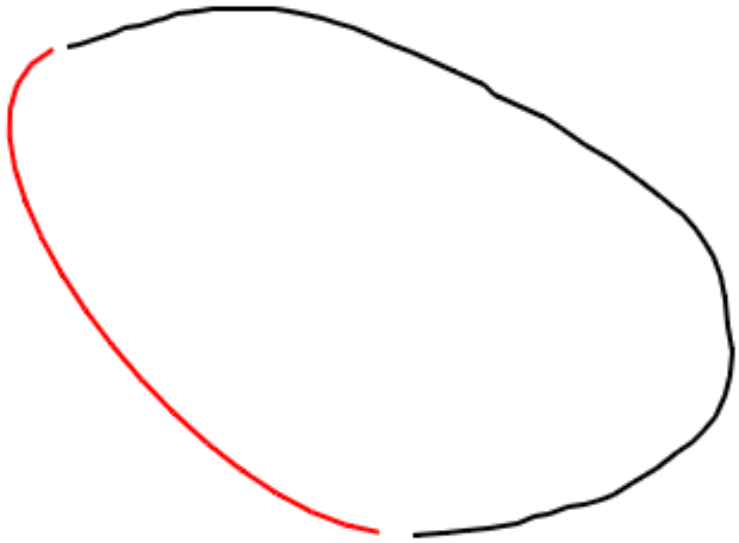


Abbildung 104: Beispiel für Bezierkurve

Diese Bezierkurve wurde mit der `firstLastBezier`-Funktion ermittelt, da es sich hier nur um eine Linie handelt, deren Anfangs- und Endpunkt miteinander verbunden wurde.

8.5 Client

Dem Client werden bei einer Verbindung mit der Software 3 JavaScript Dateien übermittelt. In diesen wird das dynamische HTML der Weboberfläche behandelt. Die 3 Dateien sind ähnlich wie beim Server, eigenen Aufgaben zugewiesen.

8.5.1 Globale Deklarationen: global.js

Wie der Name schon sagt, werden in global.js Variablen und Funktionen deklariert, die global verwendet werden. Die zwei wichtigsten globalen Variablen sind:

Listing 72: global.js - Variablen

```
1 //var url = 'http://localhost:8080',  //Notebook
2 var url = '192.168.0.1:8080',         //Raspberry Pi
3     socket = io.connect(url),
```

Die Variable `url` beinhaltet die Adresse des Servers. Wenn man die Software intern auf einem Rechner laufen lässt, reicht es, wenn man nur `localhost` und den gewünschten Port verwendet. Sobald die Software jedoch in einem Netzwerk funktionieren soll, muss die IP-Adresse des Servers angegeben werden. Das Raspberry Pi hört in diesem Fall auf die Adresse 192.168.0.1 und den Port 8080.
`Socket` wird beim Client für jegliche Art von Kommunikation mit dem Server verwendet. Für eine Übertragung zwischen Client und Server wird beim Client die Adresse des Servers benötigt, der Server braucht allerdings keine Adresse vom Client, da er über den Socket kommuniziert und dieser einzigartig für jeden Client ist.

Listing 73: global.js - Funktionen

```
1 function clearCanvas(canvas) {
2     canvas.width = canvas.width;
3 }
4
5 function clearCanvasprev(canvasprev) {
6     canvasprev.width = canvasprev.width;
7 }
```

Auch bei den Funktionen werden zwei global verwendet. Diese beiden sind bei Listing 73 zu sehen. Sowohl `clearCanvas` als auch `clearCanvasprev` werden nur verwendet, um beide Zeichenflächen zu löschen. Global deklariert wurden diese Funktionen, weil sie sowohl vom gui.js als auch vom script.js verwendet werden.

8.5.2 Graphische Oberfläche: gui.js

Die `gui.js` Datei kümmert sich um fast alles, was direkten Einfluss auf die graphische Oberfläche hat. Das heißt, sobald irgendetwas mit Enter bestätigt wird, oder ein neues Popup - Fenster erscheint, wird Programmlogik von der `gui.js` Datei abgearbeitet.

8.5.2.1 Slider

Die Slider sind Schieberegler, welche bei den Einstellungen zu finden sind.

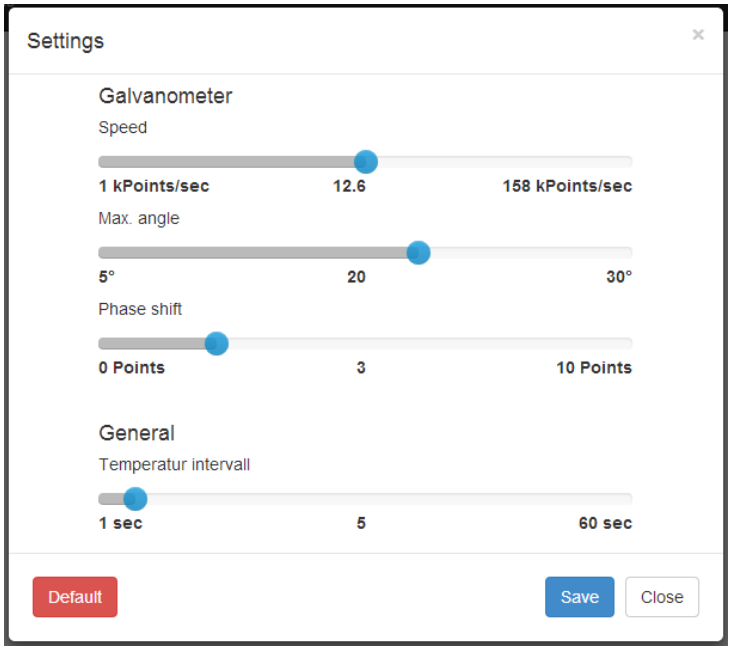


Abbildung 105: Settings - Modal

Damit die Slider funktionieren, müssen sie erst mit folgendem Code initialisiert werden.

Listing 74: Slider - Initialisierung

```
var speedSlider = $('#speed-slider').slider({
        'min': 1,
        'max': 158,
        'step': 1,
        'value': 25,
        'tooltip': 'hide'
    }),
    angleSlider = $('#angle-slider').slider({
        'min': 5,
        'max': 30,
        'step': 1,
        'value': 15,
        'tooltip': 'hide'
    }),
    intervallSlider = $('#intervall-slider').slider({
        'min': 1,
        'max': 60,
        'step': 1,
        'value': 30,
```

```
20          'tooltip': 'hide'
21      }),
22      phaseSlider = $('#phase-slider').slider({
23          'min': 1,
24          'max': 10,
25          'step': 1,
26          'value': 3,
27          'tooltip': 'hide'
28      });
```

Jeder Slider hat mindestens vier Attribute. Jedes Attribut muss direkt bei der Initialisierung angegeben werden. Der Minimal- und Maximalwert wird mit `'min'` und `'max'` eingestellt. `'Step'` gibt die minimale Sprungweite des Sliders an, diese sollte je nach Spannweite gut gewählt werden. Als letztes Attribut wird `'value'` benötigt. Der Value Parameter gibt die Position des Sliders bei der Initialisierung an.
Bei jedem Slider sind optionale Parameter möglich. Bei der Initialisierung wurde nur ein optionaler Parameter `'tooltip': 'hide'` verwendet. Mit diesem Parameter wird verhindert, dass ein Tooltip über dem Slider erscheint.

Wenn vom Server neue Einstellungen empfangen worden sind, müssen die Slider auf die neuen Werte eingestellt werden. Dies wird mit folgendem Code gemacht:

Listing 75: Slider - Wert ändern

```
1  socket.on('settings', function (data) {
2      if (data.status == 'dbIsEmpty') {
3          alert('Warning!', 'Database is empty! Default Values applied!');
4          speedSliderValue = 1.397;
5          angleSliderValue = 20;
6          intervallSliderValue = 5000;
7          phaseSliderValue = 3;
8      }
9      else {
10         speedSliderValue = data.speed;
11         angleSliderValue = data.angle;
12         intervallSliderValue = data.intervall;
13         phaseSliderValue = data.phase;
14     }
15     speedSlider.slider('setValue', speedSliderValue);
16     angleSlider.slider('setValue', angleSliderValue);
17     intervallSlider.slider('setValue', intervallSliderValue / 1000);
18     phaseSlider.slider('setValue', phaseSliderValue);
19     $('#speedValue').text(speedSliderValue);
20     $('#angleValue').text(angleSliderValue);
21     $('#tempValue').text(intervallSliderValue / 1000);
22     $('#phaseValue').text(phaseSliderValue);
23 });
```

Falls vom Server ungültige Daten kommen, bzw. noch keine Werte in der Datenbank gespeichert wurden, werden die Defaultwerte übernommen und gleichzeitig eine Warnung ausgegeben, dass die Datenbank leer ist.
Bei gültigen Daten, können Geschwindigkeit und Winkel direkt übernommen werden, beim Temperaturintervall muss noch mit 1000 dividiert werden, da die Oberfläche mit Sekunden arbeitet und intern mit Millisekunden gerechnet wird. Anschließend werden die dazugehörigen Slider auf die richtige Position gesetzt.

8.5.2.2 On Click Events

On Click Events sind Ereignisse die aufgerufen werden, wenn z.B. auf einen Button oder ein bestimmtes Element geklickt wird. Ereignisse dieser Art wurden bei fast allen Popups bzw. Modalfenster verwendet.
Wie ein On Click Event mit JavaScript erstellt wird, zeigt folgender Codeausschnitt:

Listing 76: Click Event Beispiel

```
// shows the create modal
$('#create-button').on('click', function (e) {
    e.preventDefault();
    $('#create-modal').modal();
});
```

Sobald auf das Objekt mit der ID `'create-button'` geklickt wird, wird die Funktion aufgerufen. In der Funktion wird mit jQuery auf das `'create-modal'` Objekt zugegriffen und dieses mit `.modal()` aufgerufen.

Wie bereits erwähnt, werden die meisten Modalfenster mit einem on Click Event aufgerufen. On Click Events werden jedoch nicht nur für das Öffnen von neuen Fenstern verwendet, sondern auch für das Ändern der Farbe eines Buttons oder des Mauszeigers, wie der folgende Code zeigt.

Listing 77: Bleistift / Radiergummi toggle

```
$('#eraser').on('click', function (e) {
    e.preventDefault();
    if (eraserSelected == false) {
        eraserSelected = true;
        $('#eraser').toggleClass('btn-custom-mode');
        $('#pencil').toggleClass('btn-custom-mode');
    }
});

$('#pencil').on('click', function (e) {
    e.preventDefault();
    if (eraserSelected) {
        eraserSelected = false;
        $('#pencil').toggleClass('btn-custom-mode');
        $('#eraser').toggleClass('btn-custom-mode');
    }
});
```

Bei diesem Codeausschnitt wird ein Click Event für einen Button verwendet, um einen Radiobutton zu simulieren. Die zwei Buttons `#pencil` und `#eraser` sind ähnlich wie Radiobuttons. Sobald einer angeklickt wird, wird der andere wieder weggeklickt. Sprich, es kann nur einer von beiden aktiviert sein.

8.5.2.3 Scrollbalken

Damit das Modal, bei dem man Zeichnungen auswählen kann, nicht zu groß wird, wird ein Scrollbalken benötigt. Den Scrollbalken erstellen wir ähnlich wie die zuvor erwähnten Slider. Durch das jQuery slimScroll Plugin kann mit folgendem Code ein Scrollbalken initialisiert werden.

Listing 78: Scrollbalken

```
$('#databases').slimScroll({
    position: 'left',
    height: '400px',
    railVisible: true,
    alwaysVisible: false,
    disableFadeOut: true
});
```

`#databases` ist das Objekt bei welchem ein Scrollbalken eingefügt werden soll. Mit `.slimScroll()` wird ein neues HTML-Div um das `databases`-Objekt gelegt, welches einen Scrollbalken benötigt.
Für den Scrollbalken gibt es diverse Parameter.
`Position` gibt an, an welcher Stelle sich der Balken befinden soll, in diesem Fall wäre das links.
`Height` gibt die Größe des Balkens an. Im Normalfall, wie im Beispiel auch, werden feste Größenangaben mit Pixel(px) gemacht, es kann jedoch auch die Größe mit Prozentangaben festgelegt werden.
`RailVisible` wird benötigt, um den kompletten Balken anzuzeigen. Wäre diese Variable auf `false`, würde man nur den Scrollbalken selber sehen, jedoch nicht die Leiste, auf der man ihn ziehen kann.
`AlwaysVisible` ist in diesem Fall auf `false`, da der Scrollbalken nicht angezeigt werden soll, wenn man mit dem Mauszeiger außerhalb des Bereichs der Liste ist.
`disableFadeOut` wird auf `true` gesetzt, damit der Scrollbalken langsam ausbleicht.

Nach der Initialisierung sieht der Scrollbalken wie bei Abbildung 106 aus.

Abbildung 106: Scrollbar

8.5.3 Programmlogik: script.js

In der `script.js` Datei ist die eigentliche Programmlogik enthalten. Wenn sich etwas auf die Zeichenfläche oder ähnliches bezieht, wird dies in dieser Datei abgehandelt.

8.5.3.1 Zeichnungen vom Server laden und ausgeben

Im Code bei Listing 79 ist zu sehen, wie die vom Server zum Client gesendeten Namen der Collections verarbeitet werden, um die Namen der verschiedenen Zeichnungen auf der graphischen Oberfläche auszugeben.

Listing 79: Sheets

```
1  //gets all the collection names from the DB
2  socket.on('sheets', function (data) {
3      curDBs = data;
4      //removes all entries and inserts all the collection names except the ↙
           ↳ standard ones
5      databases.children().remove();
6      for (var i = 0; i <= data.length - 1; i++) {
7          if (data[i].name != "local" && data[i].name != "startup_log" && data[i↙
               ↳ ].name != "system.indexes" && data[i].name != "settings") {
8              if (data[i].name == selectValue) {
9                  databases.append('<li><a href="#" style="font-weight: bolder;">↙
                       ↳ ' + data[i].name + '</a></li>');
10             } else {
11                 databases.append('<li><a href="#">' + data[i].name + '</a></li>↙
                       ↳ ');
12             }
13         }
14     }
15 });
```

Sobald der Server die Daten zum Client gesendet hat, wird die Funktion von Listing 79 aufgerufen. Der Code aus Zeile 5 löscht die Einträge der bereits vorhandenen Liste. In Zeile 6 bis 14 wird mit einer `for` Schleife das Datenarray durchlaufen und die Namen der Zeichnungen in eine Liste geladen. Mithilfe von zwei if - Anweisungen (Z. 7 und 8) wird verhindert, dass die `'local'`, `'startup_log'` und `'system.indexes'` Collections als Zeichnungen bezeichnet wird. Diese drei Collections werden von der MongoDB für Fehlerlogs und Indexierungen verwendet. Weiters wird die Collection `'settings'` ausgeschlossen, da diese Collection nur zum Speichern von Einstellungen vom Raspberry Pi und dem STM32 verwendet wird.

8.5.3.2 Zeichnung wechseln

Der folgende Code gibt an, wie sich die Software verhält, wenn ein Client die Collection, sprich die aktuelle Zeichnung wechselt.

Listing 80: change current Collection

```
1  //gets the new database if another client changed it
2  socket.on('change selection', function (data) {
3      // the selected collection becomes bold, and save the collection name
4      selectValue = data.name;
5      selectedName = selectValue;
6      $('#database-name').text(selectValue);
```

```
7
8       databases.children().each(function (key, item) {
9           var $item = $(item);
10          if ($item.text() == selectValue) {
11              $('#databases li > a').css('font-weight', 'normal');
12              $item.css('font-weight', 'bolder');
13          }
14      });
15
16      canvas.fadeIn();
17      clearCanvas(document.getElementById('paper'));
18      clearCanvas(document.getElementById('bezier'));
19  });
```

Wenn ein Client eine andere Zeichnung lädt, wird sofort vom Server aus ein Broadcast gestartet, der allen anderen Clients ausrichtet, auf welche Zeichnung gewechselt wurde.
Anschließend wird die neue aktuelle Zeichnung im dazu entsprechenden Modal fett markiert. Damit keine falschen Linien auf der Zeichenfläche zu sehen sind, wird diese gelöscht.

8.5.3.3 Neue Zeichnung erstellen

Wenn eine neue Zeichnung erstellt wird, muss der Name auf Sonderzeichen oder doppelte Namen überprüft werden. Dies geschieht mit dem Code aus Listing 81.

Listing 81: create Collection

```
1  $('#create').on('click', function () {
2          var unacceptedChars = ["/", "\\", ".", "*", "<", ">", ":", "|", "?", "$↙
               ↳ "], name = dbTextbox.val(), correctName = true;
3
4          if (name.length < 16) {
5              for (var i = 0; i < (curDBs.length > unacceptedChars.length ? ↙
                   ↳ curDBs.length : unacceptedChars.length); i++) {
6                  if (name == '' || name.indexOf(unacceptedChars[i]) != -1) {
7                      alert('Error!', 'Please enter a valid name! Following ↙
                           ↳ characters are not allowed / \\ . * < > : | ? $ ');
8                      correctName = false;
9                      break;
10                 }
11                 else if (curDBs[i] && name == curDBs[i].name) {
12                     alert('Error!', 'Unaccepted database name. "' + name + '" ↙
                           ↳ is already in use by another collection.');
13                     correctName = false;
14                     break;
15                 }
16             }
17             if (correctName) {
18                 selectedName = selectValue = name;
19                 $('#database-name').text(selectValue);
20                 socket.emit('create new db', {name: name});
21                 $('#create-modal').modal('hide');
22                 canvas.fadeIn();
23                 clearCanvas(document.getElementById('paper'));
24                 clearCanvas(document.getElementById('bezier'));
25             }
26         }
27         else
28             alert('Error!', 'Collection name is too long! Not more than 15 ↙
                   ↳ characters allowed.');
29         dbTextbox.val('');
30     }
31 );
```

Als erstes findet die Überprüfung auf die Länge des Strings statt. Falls der Name durch diese Überprüfung kommt, wird er auf alle Sonderzeichen, die in der MongoDB nicht erlaubt sind überprüft. Diese sind im Array `unacceptedChars` gespeichert. Während die Sonderzeichen überprüft werden, wird auch überprüft, ob der Name bereits vorhanden ist. Bei einer Übereinstimmung mit einem Sonderzeichen oder einem gleichen Namen wird sofort eine Fehlermeldung aufgerufen und die Funktion unterbrochen.
Falls der String durch alle Überprüfungen durchkommt, wird der neue Name dem Server übermittelt und eine neue Zeichnung erstellt. Daraufhin wird das `create-modal` geschlossen und die Zeichenfläche eingeblendet.

8.5.3.4 Maustastenposition bzw. Touchgesten Erkennung

In den folgenden Funktionen werden die Koordinaten des Mauszeigers bzw. Fingers ausgelesen und einer Funktion übergeben. Bevor die Koordinaten übergeben werden, werden diese auf den Bereich [0;1] transformiert.

Maustaste unten / Start Touchbewegung

Wenn auf dem Canvas die Maustaste nach unten gedrückt wird, oder auf ein Tablet/Smartphone eine Touchgeste vollzogen wird, ruft das JavaScript sofort eine Funktion auf, die dem Server meldet, dass eine Linie gezeichnet wird.

Listing 82: Beginn Linie

```
1  //on touch(tablet) saves the coordinates and draws or deletes lines
2  canvas.on('touchstart', function (e) {
3      e.preventDefault();
4      canvasbezier.stop();
5      canvasbezier.fadeOut();
6      touch = e.originalEvent.touches[0] || e.originalEvent.changedTouches[0];
7      if (eraserSelected)
8      {
9          erase(touch.pageX / userpanel, touch.pageY / userpanel - offset, 'start↙
               ↳ ');
10     }
11     else
12         startDrawing(touch.pageX / userpanel, touch.pageY / userpanel - offset)↙
               ↳ ;
13 });
14 //on mousedown draw or delete lines
15 canvas.on('mousedown', function (e) {
16     e.preventDefault();
17     canvasbezier.stop();
18     canvasbezier.fadeOut();
19     if (eraserSelected)
20     {
21         erase(e.pageX / userpanel, e.pageY / userpanel - offset, 'start');
22     }
23     else
24         startDrawing(e.pageX / userpanel, e.pageY / userpanel - offset);
25 });
```

Die zwei Funktionen für Touchgeste bzw. `Maustaste unten` sind fast identisch, der einzige Unterschied ist, dass bei der Touchgeste die Koordinaten anders eingelesen werden müssen. Damit die Bezierlinien nicht ständig mitgezeichnet werden, wenn eine Linie gezeichnet wird, wird der Canvas für die Bezierlinien ausgeblendet.
Anschließend wird überprüft ob der Pencil oder Eraser ausgewählt ist und die Daten je nach Auswahl der entsprechenden Funktion übergeben.

Mauszeiger bewegen / Touchbewegung

Nachdem die `Maustaste unten` / `Touch start Funktion` vollendet ist, wird die `moving` Funktion aufgerufen. Diese Funktion zeichnet bzw. radiert bei Bewegungen des Mauszeigers.

Listing 83: Linie zeichnen

```
//save the coordinates and draw lines during mousemoving
doc.on('mousemove', function (e) {
    moving(e.pageX / userpanel, e.pageY / userpanel - offset);
});
//save the coordinates and draw lines during touchmoving
doc.on('touchmove', function (e) {
    e.stopPropagation();
    touch = e.originalEvent.touches[0] || e.originalEvent.changedTouches[0];
    moving(touch.pageX / userpanel, touch.pageY / userpanel - offset);
});
```

Auch hier sind die Funktionen für `Maus bewegen` bzw. `Touchbewegung` bis auf die Koordinatenauslesung identisch.
Die Koordinaten werden wieder der entsprechenden Funktion übergeben.

Maustaste oben / Ende Touchbewegung

Ist eine Linie fertig gezeichnet, geht die Maustaste nach oben und die `Touchbewegung` ist fertig. Somit können dem Server die letzten Koordinaten zugesendet werden.

Listing 84: Ende Linie

```
//on mouseup / mouseleave save the last coordinate and stop drawing/deleting
doc.on('mouseup mouseleave', function (e) {
    canvasbezier.fadeIn();
    if (deleting)
        erase(e.pageX / userpanel, e.pageY / userpanel - offset, 'stop');
    else
        stopDrawing(e.pageX / userpanel, e.pageY / userpanel - offset);
});
//on touchend save the last coordinate and stop drawing/deleting
doc.on('touchend', function (e) {
    touch = e.originalEvent.touches[0] || e.originalEvent.changedTouches[0];
    canvasbezier.fadeIn();
    if (deleting)
        erase(e.pageX / userpanel, e.pageY / userpanel - offset, 'stop');
    else
        stopDrawing(touch.pageX / userpanel, touch.pageY / userpanel - offset);
});
```

Auch hier sind die beiden Funktionen beinahe identisch. In diesen Funktionen werden die Koordinaten des Mauszeigers bzw. Fingers ausgelesen, überprüft ob gerade gezeichnet oder radiert wird und anschließend der entsprechenden Funktion übergeben.

8.5.3.5 Start-/Stopp zeichnen

Folgende Funktionen werden beim Beginn bzw. Ende einer Linie aufgerufen.

Listing 85: Start-/Stopp zeichnen

```
//function for the first coordinates
function startDrawing(x, y) {
    drawing = true;

    prev.x = Number(x.toPrecision(round));
    prev.y = Number(y.toPrecision(round));
    socket.emit('startDrawing', prev);
```

```
8  }
9  //function for the last coordinates
10 function stopDrawing(x, y) {
11     drawing = false;
12
13     last.x = Number(x.toPrecision(round));
14     last.y = Number(y.toPrecision(round));
15     socket.emit('stopDrawing', last);
16     //if the buffer isn't empty send the missing coordinates
17     if (buffer.length > 0) {
18         socket.emit('mousemove', buffer);
19         count = 0;
20         buffer.length = 0;
21     }
22 }
```

Beim `startDrawing` werden die Koordinaten des ersten Punktes auf `round` viele Kommastellen gerundet und in den `prev` Array gespeichert. Das Runden der Koordinaten mindert etwas die Netzwerkbelastung. Daraufhin wird dem Server der `prev` Array übertragen.
Beim `stopDrawing` werden die Koordinaten des letzten Punktes gerundet und dem Server übertragen. Falls im `buffer` (siehe 8.5.3.7 Teil 1 - Zeichnen) noch nicht übertragene Koordinaten vorhanden sind, werden auch diese übertragen und der `buffer` zurückgesetzt.

8.5.3.6 Start-/Stopp löschen

Diese Funktion wird ausgelöst, sobald der Client anfängt bzw. aufhört eine Linie zu zeichnen.

Listing 86: Start-/Stopp löschen

```
1  //coordinates for the delete rectangle
2  function erase(x, y, status) {
3      deletingData.x1 = Number((x - erasersize / userpanel).toPrecision(round));
4      deletingData.y1 = Number((y - erasersize / userpanel).toPrecision(round));
5      deletingData.x2 = Number((x + erasersize / userpanel).toPrecision(round));
6      deletingData.y2 = Number((y + erasersize / userpanel).toPrecision(round));
7
8      if (status == 'start') {
9          deleting = true;
10         prevXdel = Number(x.toPrecision(round));
11         prevYdel = Number(y.toPrecision(round));
12         deleteArea(deletingData.x1 * userpanel, deletingData.y1 * userpanel, ↙
               ↳ deletingData.x2 * userpanel, deletingData.y2 * userpanel);
13         deletingData.status = 'startDeleting';
14     }
15     else {
16         deleting = false;
17         deletingData.status = 'stopDeleting';
18     }
19
20     socket.emit('delete', deletingData);
21 }
```

Wenn die Funktion aufgerufen wird, werden die vier Eckkoordinaten des Radierbereichs ausgerechnet. Daraufhin wird abgefragt, ob es der Anfang oder das Ende eines Radiervorgangs ist.

Anfang des Radiervorgangs

Beim Anfang eines Radiervorgangs wird mit den berechneten Koordinaten ein weißes Quadrat gebildet, welches beim löschenden Client die Linien mit der Hintergrundfarbe übermalt. Die aktuellen Koordinaten werden über das `prevXdel` und `prevYdel` Array gespeichert. Anschließend wird `deletingData.status` auf `'startDeleting'` gesetzt und das komplette `deletingData` Objekt dem Server übergeben.

Ende des Radiervorgangs

Am Ende des Radiervorgangs wird lediglich `deletingData.status` auf `'stopDeleting'` gesetzt und das Objekt `deletingData` dem Server übergeben.

8.5.3.7 Zeichnen

Bei Bewegungen mit gedrückter Maustaste oder Touchgesten wird die `moving` Funktion aufgerufen. Diese ist in zwei Teile aufgeteilt. Der erste Teil wird beim Zeichnen und der zweite beim Löschen abgearbeitet.

Teil 1 - Zeichnen

Listing 87: Moving - Draw

```
if (drawing) {
    if (x > 1 || x < 0 || y < 0 || y > 1) {
        if (y > 1) y = 1;
        if (x > 1) x = 1;
        if (y < 0) y = 0;
        if (x < 0) x = 0;

        stopDrawing(x, y);
    }
    //calculates the distance between two coordinates
    deltax = Math.abs(x - prev.x);
    deltay = Math.abs(y - prev.y);
    distance = Math.sqrt(Math.pow(deltax, 2) + Math.pow(deltay, 2));


    if (distance >= 0.005) {
        buffer[count] = {
            x: Number(x.toPrecision(round)),
            y: Number(y.toPrecision(round)),
            prevX: prev.x,
            prevY: prev.y
        };

        count++;
        if (count >= 5) {
            //sends the buffer to the server
            socket.emit('mousemove', buffer);
            count = 0;
            buffer.length = 0;
        }
        drawLine(prev.x * userpanel, prev.y * userpanel, x * userpanel, y * ↙
            ↳ userpanel);
```

```
32          prev.x = Number(x.toPrecision(round));
33          prev.y = Number(y.toPrecision(round));
34      }
35  }
```

Sobald die `moving` Funktion ausgeführt wird, wird überprüft ob der Abstand zwischen dem neuen und dem vorherigen Punkt größer als 0.5% der Zeichenfläche ist. Bei größerem Abstand wird das neue Koordinatenpaar im `buffer` Array abgespeichert. Sobald das `buffer` Array mehr als 4 Koordinatenpaare beinhaltet wird der Linienteil dem Server übermittelt und anschließend das Array zurückgesetzt.
Während dem Zeichnen wird die Linie beim zeichnenden Client selbst gezeichnet. Zum Schluss wird das aktuelle Koordinatenpaar im `prev` Array abgespeichert. Das `prev` Array wird für den nächsten Linienabschnitt benötigt.

Teil 2 - Löschen

Listing 88: Moving - Delete

```
1  function moving(x, y) {
2      if (deleting) {
3          if (x > 1 || x < 0 || y < 0 || y > 1)
4              return;
5
6          deltax = Math.abs(x - prevXdel);
7          deltay = Math.abs(y - prevYdel);
8          distance = Math.sqrt(Math.pow(deltax, 2) + Math.pow(deltay, 2));
9
10         if (distance >= 0.005) {
11             deletingData.x1 = Number((x - erasersize / userpanel).toPrecision(↙
                   ↳ round));
12             deletingData.y1 = Number((y - erasersize / userpanel).toPrecision(↙
                   ↳ round));
13             deletingData.x2 = Number((x + erasersize / userpanel).toPrecision(↙
                   ↳ round));
14             deletingData.y2 = Number((y + erasersize / userpanel).toPrecision(↙
                   ↳ round));
15             deletingData.status = 'moving';
16
17             deleteArea(deletingData.x1 * userpanel, deletingData.y1 * userpanel↙
                   ↳ , deletingData.x2 * userpanel, deletingData.y2 * userpanel);
18             socket.emit('delete', deletingData);
19
20             prevXdel = x;
21             prevYdel = y;
22         }
23     }
24 }
```

Wie bereits beim Teil 1 wird auch beim Löschen die Überprüfung gemacht, ob der neue Punkt mindestens 0.5% vom vorherigen Punkt entfernt ist. Bei größerer Entfernung werden die vier Koordinaten berechnet, die ein Viereck um den Mauszeiger bilden. Diese zwei Koordinatenpaare werden dem Server übermittelt, damit dieser alle Punkte in diesem Bereich löschen kann. Zur Überprüfung wird auch ein Status mitübertragen der angibt, ob der Client mit dem Radiervorgang bereits fertig ist.
Für eine flüssigere Wirkung des Radierens wird beim Client selber die derzeit bestehende Linie mit der Hintergrundfarbe des Zeichenbereichs übermalt.

Zum Schluss werden wieder die aktuellen Koordinaten im `prevXdel` und `prevYdel` Array gespeichert.

8.5.3.8 On Moving

Um ein flüssiges Bild auf mehreren Clients zu erstellen, wird nach jedem fünften Punkt ein Array zum Server übertragen. Daraufhin macht der Server einen Broadcast und sendet die fünf Koordinatenpaare an alle anderen Clients. Jeder Client kann die Linie selber zeichnen, wodurch eine flüssige Darstellung auf jedem Client gleichzeitig möglich ist. In Listing 89 ist zu sehen wie diese Aktion im Code umgesetzt wird.

Listing 89: on Moving

```
1  //receives single coordinates from the server and draws them
2  socket.on('moving', function (data) {
3      for (var i = 0; i <= data.length - 1; i++) {
4          drawLine(data[i].prevX * userpanel, data[i].prevY * userpanel, data[i].↙
               ↳ x * userpanel, data[i].y * userpanel);
5      }
6  });
```

In der `for` Schleife wird jeder Punkt auf die Größe der Zeichenfläche angepasst. Anschließend werden die Punkte miteinander verbunden.

8.5.3.9 Alte Zeichnung laden

Wird ein neues Objekt ausgewählt, muss der Server allen Clients die neuen Daten senden. Anschließend wird mit folgendem Code das neue Bild auf die Zeichenfläche übertragen.

Listing 90: load old Drawing

```
1  //receives a complete collection of coordinates from the server and draws them ↙
       ↳ with or without the interpolation ones
2  socket.on('load old drawing', function (data) {
3      for (var j = data.length - 1; j >= 0; j--) {
4          for (var i = data[j].length - 2; i >= 0; i--) {
5              if (j % 2 == 0) {
6                  drawLine(data[j][i + 1].x * userpanel, data[j][i + 1].y * ↙
                       ↳ userpanel,
7                      data[j][i].x * userpanel, data[j][i].y * userpanel);
8              }
9              else if (j % 2 == 1 && !deleting) {
10                 if (interpolationSelected)
11                     drawInterpolation(data[j][i + 1].x * userpanel, data[j][i +↙
                           ↳  1].y * userpanel,
12                         data[j][i].x * userpanel, data[j][i].y * userpanel);
13             }
14         }
15     }
16 });
```

Das vom Server zum Client übertragene zweidimensionale Array wird mithilfe von zwei ineinander verschachtelten `for` Schleifen durchlaufen.
Die erste Dimension `data[j]` entspricht einer Linie. Mit der zweiten Dimension werden die einzelnen Punkte der Linie abgefragt. Bei jedem Punkt wird überprüft, ob es sich um eine sichtbare oder eine unsichtbare Bezierlinie handelt. Bezierlinien haben immer eine ungerade `lineID`, die

Überprüfung wird mit %2 (Modulo 2) durchgeführt.
Bei Bezierlinien wird vor dem Zeichnen überprüft, ob sie mitgezeichnet werden sollen oder nicht.

8.5.3.10 Vorschau zeichnen

Sobald im `choose-modal` eine Zeichnung ausgewählt wird, ist eine Vorschau im selben Modal dargestellt (siehe Abbildung 107).

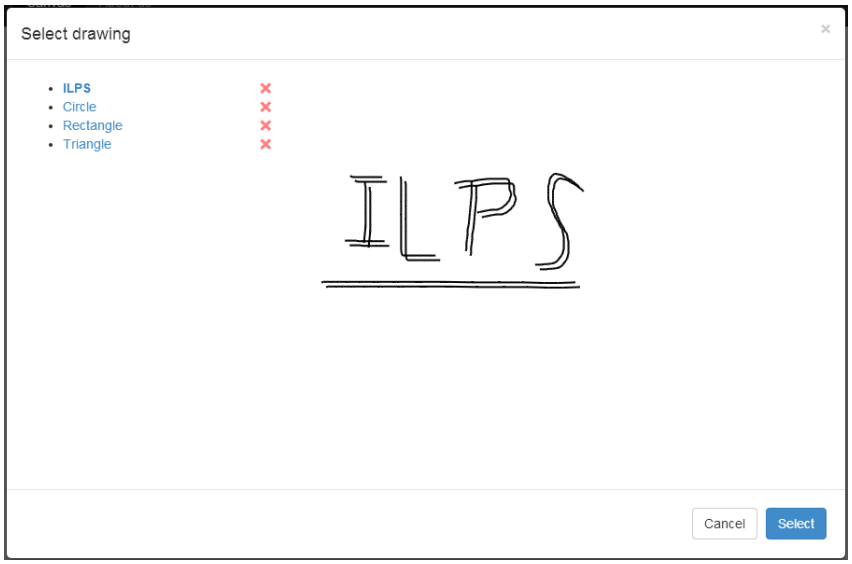


Abbildung 107: Vorschau einer Zeichnung

Listing 91: load preview drawing

```
1  //draws the preview image
2  socket.on('load preview', function (data) {
3      for (var j = data.length - 1; j >= 0; j--) {
4          for (var i = data[j].length - 2; i >= 0; i--) {
5              if (j % 2 == 0) {
6                  drawLine(data[j][i + 1].x * userpanel, data[j][i + 1].y * ↙
                       ↳ userpanel,
7                      data[j][i].x * userpanel, data[j][i].y * userpanel);
8              }
9          }
10     }
11 });
```

Der Code in Listing 91 ähnelt sehr dem von Listing 90. Hier wird dasselbe Verfahren verwendet: der Server schickt ein zweidimensionales Array, das mit zwei verschachtelten `for` Schleifen die Punkte der Linien extrahiert und gezeichnet. Allerdings werden hier die Bezierlinien komplett ignoriert und nur die normalen Linien gezeichnet.

9 Mechanischer Aufbau

9.1 Allgemein

Das Gehäuse und der mechanische Aufbau muss praktisch, gut aussehend, genau und vor allem funktionell sein. Deshalb werden alle Teile mittels CNC-Fräse gefertigt. Für die CNC werden alle benötigten Teile mit Solidworks in 3D gezeichnet. Das verwendete Metall ist Aluminium, da es einfach zu verarbeiten ist und gut aussieht. Für die Gehäusefront wurde Plexiglas verwendet. Wichtig bei der Planung war, dass das Gehäuse nicht Magnetisch ist, den dies würde die Magnete stark beeinflusse.

9.2 Solidworks

Solidworks ist ein 3D-Zeichenprogramm, welches in der HTL-Rankweil für Zeichnungen von mechanischen Aufbauten verwendet wird. Mit diesen 3D-Zeichnungen können die Koordinaten an die CNC-Fräse übergeben werden. Das Programm ist kostenpflichtig und kommt von der Firma Dassault Systems, welche eine Schülerlizenz für die HTL-Rankweil zur Verfügung stellt. Die Lizenz ist etwas eingeschränkt, da es z.B. keine Toolbox gibt.

9.3 Gehäuse

Das Gehäuse soll möglichst klein sein und trotzdem genug Platz für alle Komponenten bieten. Benötigte Schnittstellen und Einstellungsmöglichkeiten müssen erreichbar sein. An der Frontseite und der schrägen Oberseite wird Plexiglas zur Einsicht in das Innere verwendet. Ebenso ist in der Front eine Ausnehmung, damit der Laserstrahl ungehindert und ungeschwächt aus dem Gehäuse strahlen kann. An der Rückseite sind zwei Ausnehmungen ausgefräst, um einen Kaltgerätestecker und einen Ein- bzw. Ausschalter zu befestigen.

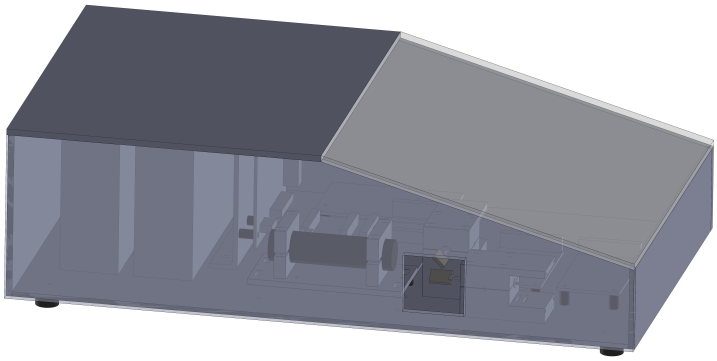


Abbildung 108: Gehäuse zusammengestellt

9.4 Galvanometer

Die Galvanometer werden aus einem Metallblock ausgefräst, damit die Spule in die Ausnehmung eingelegt werden kann. In die Ausnehmung kommt eine Achse, welche als Kernstück einen 15mm langen und 5mm dicken Diametralmagnet besitzt. Der Magnet wird mit Kunststoffhalterungen und 2mm dicken Aluminium-Stäbchen zusammengesetzt. Am vorderen Ende des einen Stäbchens ist eine Halterung für den Spiegel angebracht. Der Deckel des Galvanometergehäuses ist beinahe das selbe wie das Unterteil, er besitzt ebenso eine Spule als Kern. Die Spulen im Kern des Galvanometers dürfen die Mittelachse nicht berühren, allerdings auch nicht zu weit weg sein, damit das magnetische Feld optimal genutzt werden kann. Im Deckel des Galvanometers werden die Schrauben komplett versenkt, deshalb ist die Bohrung im oberen Teil größer. Im hinteren Teil des Galvanometers wird der Positionssensor angebracht. Dieser wird mit vier Schrauben befestigt. Zusätzlich ist mithilfe von zwei kleinen länglichen Ausnehmungen die Kapazitätsmesseinheit angebracht.

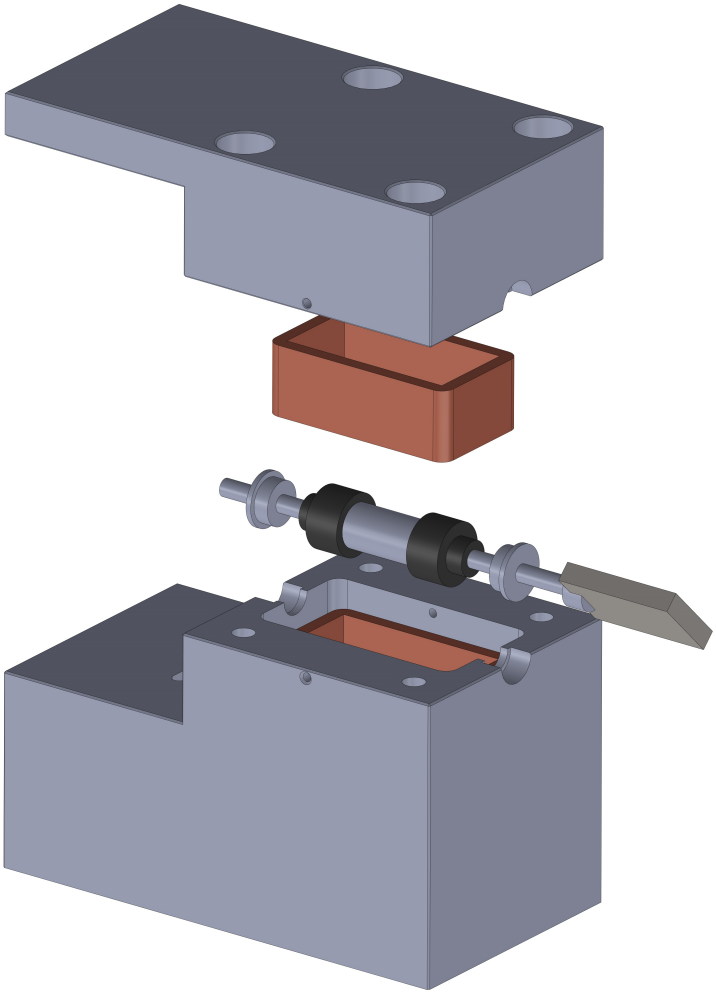


Abbildung 109: Galvanometer-3D-Explosionszeichnung

9.4.1 Magnethalterung

Die Magnethalterung wird aus Kunststoff gefertigt und ist für eine stabile und gute Verbindung zwischen dem Magneten und den beiden Achsen zuständig. Um eine bessere Stabilität zu bekommen, wird die Magnethalterung mit dem Magnet und den Achsen durch Klebstoff verbunden.

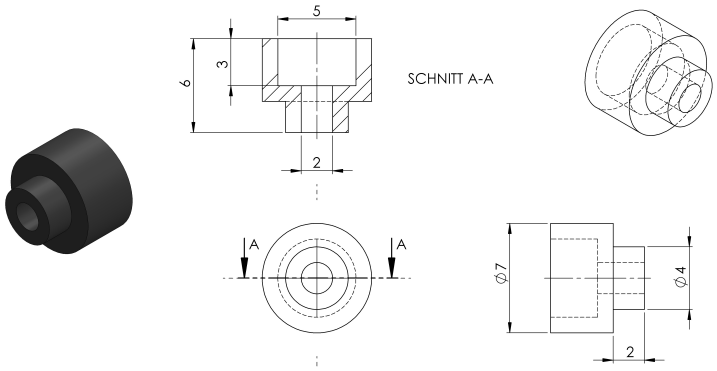
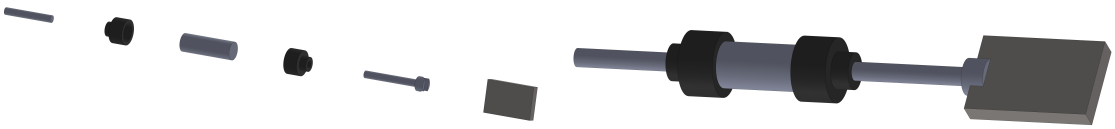


Abbildung 110: Magnethalterung

9.5 Mittelachse

Die Mittelachse ist das Kernstück des Galvanometers. Sie besteht aus sechs Komponenten. In der Mitte der Achse ist der Diametralmagnet. Mit zwei Kunststoffhalterungen wird dieser mit den beiden Trägerachsen verbunden. Am Ende der Trägerachsen ist eine Nute eingefräst, in der der Spiegel befestigt wird.



(a) Explosionszeichnung (b) zusammengebaut

Abbildung 111: Mittelachse

9.5.1 Spiegelhalterung

Der Spiegelhalter bekommt eine Nute in einer der beiden Trägerachsen. In diese Nute wird der Spiegel stabil eingeklebt. Bei unserem Prototyp führte nur das Ankleben der Spiegel an der Achse zu mechanischen Schwingungen. Durch diese Halterung soll dies unterdrückt werden, da der Spiegel noch einmal fixiert wird.

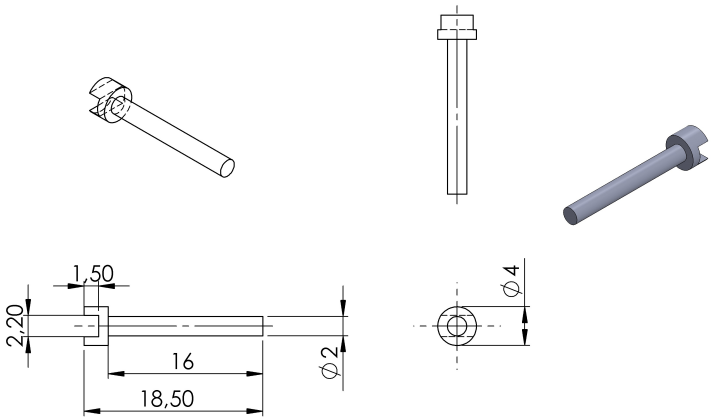


Abbildung 112: Spiegelhalterung

9.6 Galvanometer-Positionierung

Beim Aufbau der Galvanometer ist es sehr wichtig, dass die beiden Spiegel übereinander sind und die beiden Spiegel sich nicht berühren. Ebenso dürfen sich die Galvanometer gegenseitig nicht am Gehäuse berühren. Die Grundeinstellung der Spiegel ist 45°, diese wird durch die Spule in der Mitte gehalten. Der Abstand muss aber groß genug sein, damit sich die Spiegel um 360° bewegen können ohne sich zu berühren. Trotzdem sollte der Abstand zwischen den Spiegel so gering wie möglich gehalten werden.

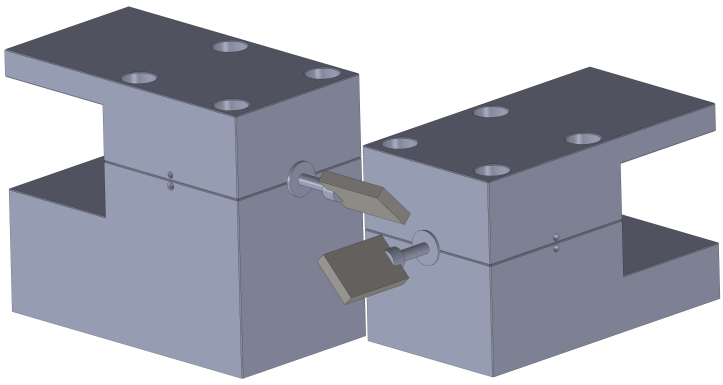
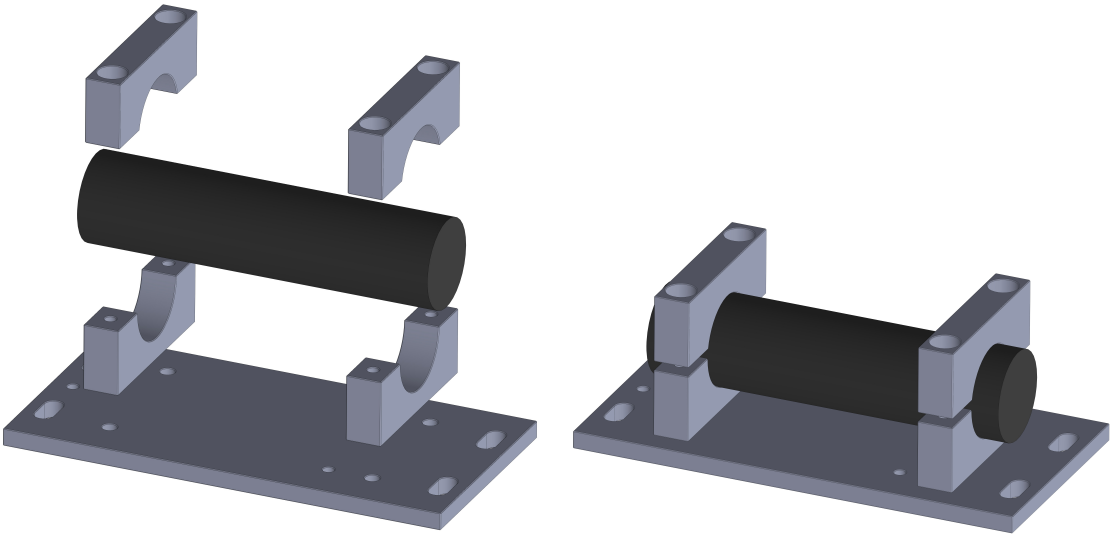


Abbildung 113: Galvanometer-3D-Positionierung

9.7 Laserhalterung

Der Laser wird zwischen zwei Metallhalterungen geklemmt und auf eine höhenverstellbare Platte befestigt. Die höhenverstellbare Platte wird mit vier Gewindestangen an der Grundplatte befestigt und mit einer Dreipunktauflage kann die richtige Höhe eingestellt werden.



(a) Explosionszeichnung (b) zusammengebaut

Abbildung 114: Laserhalterung

10 Management

10.1 Umfeldanalyse

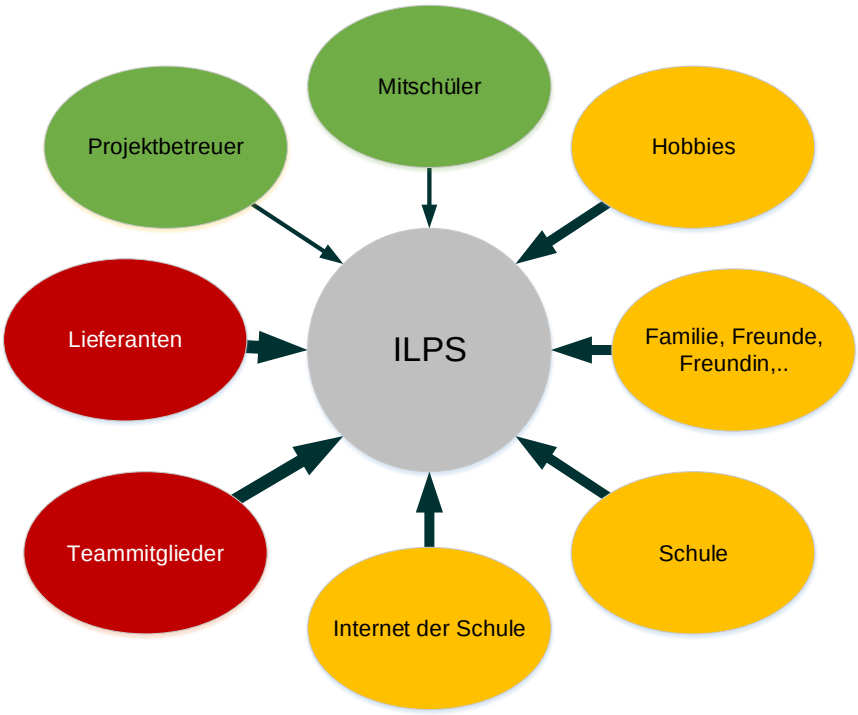


Abbildung 115: Umfeldanalyse

In dieser Grafik werden alle Einflüsse auf das Projekt dargestellt. Dabei werden Einflüsse, die eine größere Gefahr für das Projekt darstellen, mit einer anderen Farbe sowie mit einem dickeren Pfeil dargestellt, als Einflüsse mit geringer Gefahr für das Projekt.

- Grün: geringer Einfluss auf das Projekt
- Orange: mäßiger Einfluss auf das Projekt
- Rot: hoher Einfluss auf das Projekt

Einflussgröße	Art des Einflusses	Risiko 1 (niedrig) bis 10 (hoch)	Maßnahmen, Strategie
Projektbetreuer	Beurteilung, Lenkung, Ergänzungen, Änderungswünsche	1	Fortschritte kontrollieren, Zeitdruck
Mitschüler	Zusammenarbeit im Unterricht, Hilfe durch die Erfahrung anderer	1	Nur im Notfall miteinbeziehen
Hobbies	Ablenkung durch anderweitige zeitintensive Beschäftigungen	3	Nach Abschluss der HTL kann man sich wieder verstärkt den Hobbies widmen
Familie, Freunde, Freundin,..	Verpflichtungen, Anliegen, Ansprüche	5	Die Zeit nach Abschluss der HTL sollte mit der Familie verbracht werden
Schule	Prüfungen, Laborprotokolle, Hausübungen	5	Lerngruppen bilden, um effektiver zu lernen.
Internet der Schule	Laden von Datenblättern, Austausch der Dokumentation und Software nimmt viel Zeit in Anspruch, Versorgung des Raspberry Pi mit Internet nicht möglich	7	Verwendung einer VPN Verbindung
Teammitglieder	Arbeitspakete, Zusammenarbeit, gute Arbeitseinteilung ist wichtig	8	Bei Totalausfall eines Projektmitglieds müssen die Arbeitspakete neu aufgeteilt werden
Lieferanten	Lange Lieferzeiten	9	Liefertermine beachten, früh genug bestellen

10.2 Zeiterfassung

Für die Zeiterfassung sowie der Dokumentation der aktuellen Tätigkeit wurde das System von Fink Zeitsysteme[29] verwendet. Mit diesem System lassen sich einfach neue Buchungen erstellen. Zusätzlich kann man mehrere einzelne Berichte erstellen. Ein weiterer Vorteil ist, da die Buchungen online erstellt werden, dass der Betreuungslehrer immer auf dem aktuellen Stand ist und weiß, woran das Projektteam gerade arbeitet.

10.2.1 Erstellung einer Buchung

Das Erstellen einer Buchung kann grundsätzlich über zwei Arten erfolgen, über ein Webterminal oder über die Hauptseite des Zeiterfassungssystems.

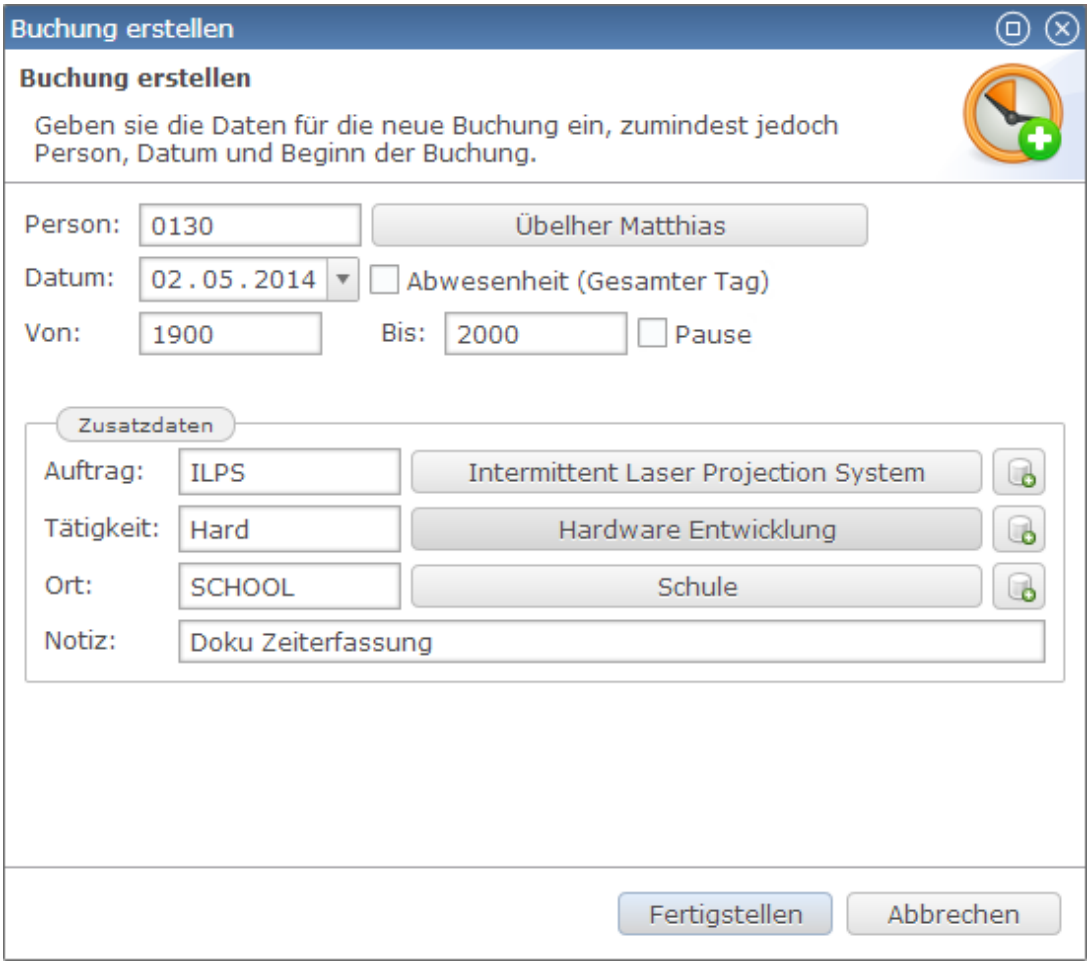


Abbildung 116: Buchung erstellen auf der Hauptseite des Zeiterfassungssystems

[29] http://www.finkzeit.at/

10.2.2 Stundenverteilung je Arbeitsauftrag

In Abbildung 117 ist die Arbeitsaufteilung je Arbeitsauftrag bzw. Arbeitsthema gut ersichtlich. Es wurde gesamt 1607,9 Stunden an dem Projekt gearbeitet.

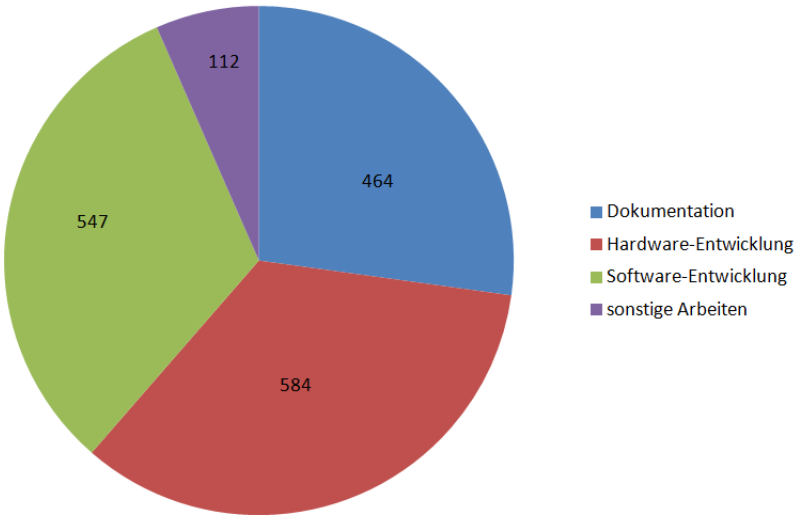


Abbildung 117: Stundenverteilung je Auftrag

Dokumentation	464 Stunden
Hardware-Entwicklung	584 Stunden
Software-Entwicklung	547 Stunden
sonstige Arbeiten	112 Stunden
Gesamt	**1707 Stunden**

Tabelle 5: Stunden je Arbeitsauftrag

10.2.3 Stundenverteilung je Monat

In Abbildung 118 ist die Arbeitsstundenverteilung je Monat der ganzen Diplomarbeitszeit ersichtlich.

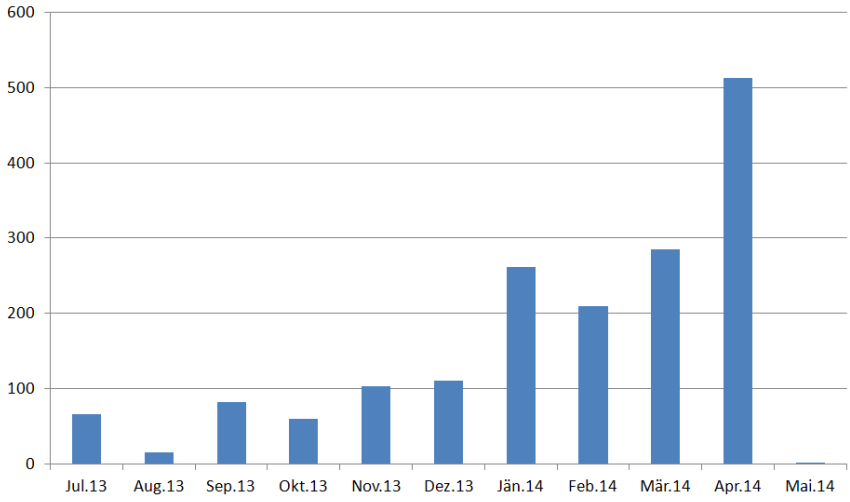


Abbildung 118: Stunden je Monat

Juli 2013	66 Stunden
August 2013	15 Stunden
September 2013	82 Stunden
Oktober 2013	60 Stunden
November 2013	103 Stunden
Dezember 2013	111 Stunden
Januar 2014	261 Stunden
Februar 2014	209 Stunden
März 2014	285 Stunden
April 2014	512 Stunden
Mai 2014	4 Stunden
Gesamt	**1707 Stunden**

Tabelle 6: Monatliche Stunden

10.3 Versionskontrollsystem

Um ein gemeinsames Arbeiten an der Dokumentation sowie an der Software zu ermöglichen, wurde Mercurial verwendet. Mercurial ist ein verteiltes, plattformunabhängiges Versionskontrollsystem. Es wird zum Austausch und Zusammenfügen der Beiträge der Mitarbeiter bei Softwareprojekten eingesetzt. Bei ILPS wurde es zum Abgleich der LaTeX - Dokumentation, Software und der Firmware verwendet. Bei richtiger Anwendung erfolgt die Zusammenführung von Dateien, welche von verschiedenen Mitarbeitern bearbeitet wurden, vollautomatisch. Um den Abgleich durchzuführen, wurde der kostenloses Repository beim Service Bitbucket[30] registriert.

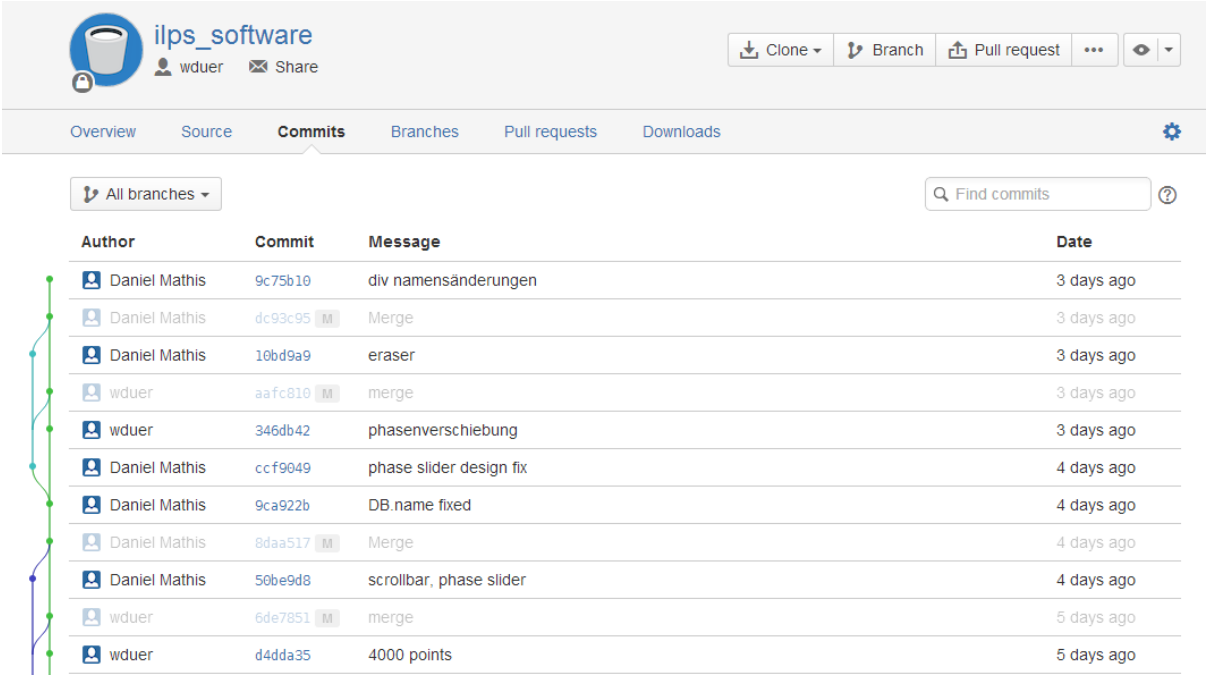


Abbildung 119: Ausschnitt der Commits auf Bitbucket

Zum Erstellen von Commits wurde Sourcetree verwendet. Mit diesem Programm kann man alle Befehle auf einer graphischen Benutzeroberfläche ausführen. Jedoch ist die Funktion `merge` sehr unzuverlässig. Daher ist es sinnvoll, das Zusammenführen von verschiedenen Versionen mit der Console durchzuführen. Unterschiede werden zum größten Teil automatisch erkannt. Unterschiede, die Mercurial nicht selber auflösen kann, müssen von Hand aufgelöst werden. Dafür wird KDiff3 verwendet.

[30] Atlassian Bitbucket, https://bitbucket.org

11 Aktueller Projektstand

Es konnten alle wesentlichen Funktionen umgesetzt werden. So konnte auch der kritischste Teil, die Galvanometer, realisiert werden. Durch Softwaretests mit verschiedenen Testpersonen konnten viele Bugs entfernt werden. Jedoch wurden bis zum Zeitpunkt der Drucklegung noch keine umfangreichen Multi-User Tests durchgeführt.

11.1 Fertige Funktionen

11.1.1 Hardware

Bei der Hardware konnten alle relevanten Ziele erreicht werden. Alle Platinen und auch das Gehäuse sind komplett aufgebaut und funktionsfähig. Zwar wurden bei den geätzen Platinen einige Fehler von Hand behoben, in der abgedruckten Fassung sind diese jedoch schon entfernt.

Ebenso wurden die Galvanometer aufgebaut und sind voll funktionsfähig. Die Geschwindigkeit reicht aus, um einfache Graphiken und Wörter zu zeichen und die Ansteuerung des Lasers funktioniert ohne Probleme.

Die Verbindung von Hauptplatine und Raspberry Pi mittels SPI über ein eigenes Protokoll ist fertiggestellt. Das Protokoll ist erweiterbar entworfen und könnte um komplexere Befehle zur Übertragung von Frames erweitert werden.

Zusätzlich wurde auf der Frontseite eine Leuchtschrift mit dem Projekttitel angebracht. Diese wird durch eine gedimmte LED beleuchtet um nicht von der Laserzeichnung abzulenken.

11.1.2 Software

Es konnten alle Must-Haves implementiert werden. So konnte mit einem NodeJS-Programm vom Zeichnen über das Abspeichern in eine Datenbank bis zum Senden an den STM32 alles realisiert werden.

Mit NodeJS konnte auch eine graphische Weboberfläche erstellt werden, die einfach und leicht zu bedienen ist. Die Weboberfläche ist sowohl mit Touch-Funktionen, als auch mit normalen Maus-Funktionen steuerbar.

Die Asynchronität von NodeJS erschwerte die Programmierung anfänglich sehr. Jedoch konnte im Endeffekt ein Programm mit sehr wenig Ressourcenanforderungen entwickelt werden.

Auch ein WLAN-Accesspoint mit dem Raspberry Pi und einem WLAN-Stick konnte problemlos eingerichtet werden. Das Installieren von MongoDB entpuppte sich als relativ umständlich, konnte jedoch nach einigen Bemühungen doch realisiert werden.

11.2 Ungelöste Funktionen

11.2.1 Hardware

Die Nice-to-Have Ziele, wie ein ausfahrbares Tablet oder der USB-Hub, wurden aus zeitlichen Gründen nicht realisiert.

Das Tablet wurde, wegen mangelnder Leistung nicht verwendet da es die Webanwendung, welche programmiert wurde, nicht ausreichen flüssig ausführen kann. Es zeigte sich, dass der Browser die Anwendung inakzeptabel langsam ausführt. Obwohl der Prozessor und die GPU genügend Leistung erbringen würden, wurde das Android - System nicht genügend an die Hardware angepasst und so können nur wenige Applikation von der Leistung der GPU profitieren. Eine Anfrage an den Hersteller, den Code entsprechend den GPL Lizenzbedingungen zu veröffentlichen, blieb erfolglos.

Aufgrund der starken Störwirkung konnten die beiden eingeplanten Schaltnetzteile im Gerät nicht zum Einsatz kommen. Diese müssten entweder sehr gut geschirmt werden oder ein Netzteil herkömmlicher Bauart zum Einsatz kommen. Dies ließ sich jedoch aus zeitlichen Gründen nicht mehr realisieren.

11.2.2 Software

Im Bereich Software konnten die Nice-to-Have Features, wie das Abspielen von Videos, nicht realisiert werden. Des Weiteren wurde auch kein Programm mehr entwickelt, das Bilder und Videos in unser Format konvertiert.

Weiters müssen noch Multi-User Tests durchgeführt werden, da dies bisher nicht im Fokus stand. Es wurden vor allem Tests mit nur einem Client durchgeführt, da dies bei einer normalen Nutzung meistens der Fall sein wird.

11.3 Endprodukt

Abbildung 120: Fertiges Produkt

In Abbildung 120 ist das fertige Produkt zu sehen. Die Frontplatte besteht aus Plexiglas. Die Vorderseite wurde Schwarz lackiert und mit einem ILPS Schriftzug versehen. Für ein leichteres Lesen ist hinter dem Schriftzug eine kleine LED-Leiste installiert, welche den Schriftzug mit einem schönen blau ausleuchtet.

12 Quellenverzeichnis

[1] Deutsche Wikipedia: Finite-Elemente-Methode http://de.wikipedia.org/wiki/Finite-Elemente-Methode, abgerufen 2013

[2] Englisches Tutorial zur Installation von MongoDB http://c-mobberley.com/wordpress/index.php/2013/10/14/raspberry-pi-mongodb-installation-the-working-guide/, abgerufen 2014

[3] Englisches Tutorial zur Installation von NodeJs http://blog.rueedlinger.ch/2013/03/raspberry-pi-and-nodejs-basic-setup/, abgerufen 2014

[4] Captive Portal und verhindern, dass ein Authentifizierungsfenster geöffnet wird http://blog.tanaza.com/blog/bid/318805/iOS-7-and-captive-portal-a-guide-to-captive-portal-requirements, abgerufen 2014

[5] Kommandoübersicht MongoDB http://wikis.gm.fh-koeln.de/wiki_db/MongoDB/KommandoBersicht, abgerufen 2014

[6] Diller-Technologies STM32 Tutorial http://www.diller-technologies.de/stm32.html, abgerufen 2014

[7] Ereignisgesteuerte Architektur http://de.wikipedia.org/wiki/Ereignisgesteuerte_Architektur

[8] NoSQL Datenbank http://de.wikipedia.org/wiki/NoSQL

[9] Npm Package Manager http://en.wikipedia.org/wiki/Npm_%28software%29

[10] MongoDB Datenbank http://de.wikipedia.org/wiki/MongoDB

[11] Non-Blocking I/O http://www.nosid.org/about-non-blocking-io.html

13 Verwendete Tools

LTspice http://www.linear.com/designtools/software

LTspice ist eine graphische Oberfläche für das SPICE[31] Simulationstool, entwickelt von Linear Technology. Es ermöglicht die Simulation von elektrischen System im Zeit- und Frequenzbereich. Die Simulation erfolgt anhand von hierarchischen Modellen welche selbst erstellt werden oder im Internet gefunden können. Zusätzlich werden viele für gängige Modelle mitgeliefert.

LaTeX http://latex-project.org

LaTeX ist eine Makrosammlung für das Textsatzsystem TeX. Es vereinfacht die Erstellung von strukturierten Dokumenten und erzwingt die Einhaltung typographischer Standards in der Dokumentation.

Mercurial http://mercurial.selenic.com

Mercurial ist ein verteiltes, plattformunabhängiges Versionskontrollsystem. Es wird zum Austausch und Zusammenfügen der Beiträge der Mitarbeiter bei Softwareprojekten eingesetzt. Bei ILPS wurde es zum Ablgeich der LaTeX - Dokumentation und der Firmware verwendet. Bei richtiger Anwendung erfolgt die Zusammenführung von Dateien welche von verschiedenen Mitarbeitern bearbeitet wurden vollautomatisch. Um den Abgleich durchzuführen, wurde der kostenloses Repository beim Service Bitbucket[32] registriert.

Inkscape http://www.inkscape.org

Inkscape ist ein Zeichenprogramm für Vektorgrafiken in vielen Formaten, allen voran das SVG[33] - Format. Für die Dokumentation von ILPS wurde Inkscape zum Zeichnen einiger Illustationen verwendet.

Win32 Disk Imager http://sourceforge.net/projects/win32diskimager

Win32 Disk Imager wurde entwickelt, um Images auf mobile Geräte wie USB-Sticks und SD-Karten zu schreiben und somit bootfähige mobile Geräte erstellen zu können. Es können auch Backups von bootfähigen Speichern erstellt werden.

PUTTY http://www.putty.org

Putty dient dazu, eine Verbindung von einem Rechner zu einem Secure-Shell- bzw. Telnet-Server herzustellen. In der textorientierten Terminalsitzung können direkt Befehle abgesetzt werden, die auf dem fernen System ausgeführt werden.

[31] Simulation Program with Integrated Circuit Emphasis
[32] Atlassian Bitbucket, https://bitbucket.org
[33] http://en.wikipedia.org/wiki/Scalable_Vector_Graphics

Atollic TrueStudio ARM http://atollic.com

TrueStudio ist eine vereinfachte und abgeänderte Version des bekannten Programms Eclipse. TrueStudio ist abgeändert für die Programmierung von STM32 und beinhaltet bereits viele benötigte Funktionen für die Programmierung am STM32.

TimerCalculator http://www.mikroe.com/timer-calculator

Mit dem TimerCalculator werden die Prescaler und Periode der verschiedenen Timer berechnet. Ohne viel Aufwand kann die Frequenz der Mikrocontroller und die Interrupt zeit eingestellt werden und der Calculator gibt den Prescaler und die Period zurück.

SolidWorks 3D 2014 http://solidworks.at

SolidWorks 3D 2014, ist ein 3D Zeichenprogramm mit welchem jedes einzelne Teil in 3D gezeichnet werden kann. Es können auch mehrere einzeln gezeichnete Objekte zu einer Baugruppe zusammengefügt werden und Gesamt dargestellt werden. Dies ist sehr gut zur Planung und diese 3D Zeichnungen können für die CNC direkt gespeist werden und erleichtert den Weg in der Fertigung erheblich.

PhpStorm http://www.jetbrains.com/phpstorm

PhpStorm ist eine Entwicklungsumgebung für die Programmiersprache PHP. Jedoch gibt es ein Node.JS Plugin. Mit diesem kann man auch das NodeJS Programm debuggen. Dadurch ist die Softwareentwicklung wesentlich leichter, als wenn man das NodeJS Programm nur in der Console starten könnte.

SourceTree http://www.sourcetreeapp.com

SourceTree ist eine grafische Benutzeroberfläche zur Bedienung der Versionskontrollsysteme Git und Mercurial. Das Programm von Atlassian, Hersteller von Entwickler- und Projektmanagement-Werkzeugen, zeigt die wichtigsten Befehle in einer einfachen Oberfläche an, sodass für diese Funktionen eine Bedienung von der Kommandozeile aus nicht notwendig ist.

Google Drive https://drive.google.com

Google Drive ist eine von Google angebotene Webanwendung, die ein Netzwerk-Dateisystem für die Synchronisation von Dateien zwischen verschiedenen Rechnern und Google-Benutzern bereitstellt und damit gleichzeitig eine Online-Datenspeicherung ermöglicht. Zusätzlich stellt es Funktionalitäten für Textverarbeitung, Tabellenkalkulation, Erstellung von Bildschirmpräsentationen, Formularen und Zeichnungen zur Verfügung. Drive ermöglicht Anwendern, diese Dokumente gemeinsam mit anderen Nutzern zu bearbeiten, wobei Änderungen in Echtzeit bei allen Beteiligten angezeigt werden.

14 Abkürzungsverzeichnis

Abkürzung	**Bezeichnung**
2D	Zweidimensional
3D	Dreidimensional
AJAX	Asynchronous JavaScript and XML
CNC	Computerized Numerical Control
CSV	Comma Separated Values
DAC	Digital Analog Converter
DB	Datenbank
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
FIFO	First In First Out
FSMC	Flexible Static Memory Controller
GNU	GNU's Not Unix
GUI	Graphical User Interface
HF	Hochfrequenz
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
IIC / I^2C	Inter-Integrated Circuit
IO	Input / Output
IP	Internet Protocol (Address)
JS	JavaScript
JSON	JavaScript Object Notation
JSONP	JSON with Padding
LCD	Liquid Crystal Display
NoSQL	Not Only SQL
OS	Operating System
PHP	PHP Hypertext Preprocessor
RAM	Random Access Memory
SD - Card	Secure Digital Memory Card
SMD	Surface Mount Device
SPI	Serial Peripheral Interface
SQL	Structured Query Language
SSH	Secure Shell
SSID	Service Set Identification
ST	STMicroelectronics
SWD	Serial Wire Debug
TCP	Transmission Control Protocol
USART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
WiFi / WLAN	Wireless LAN

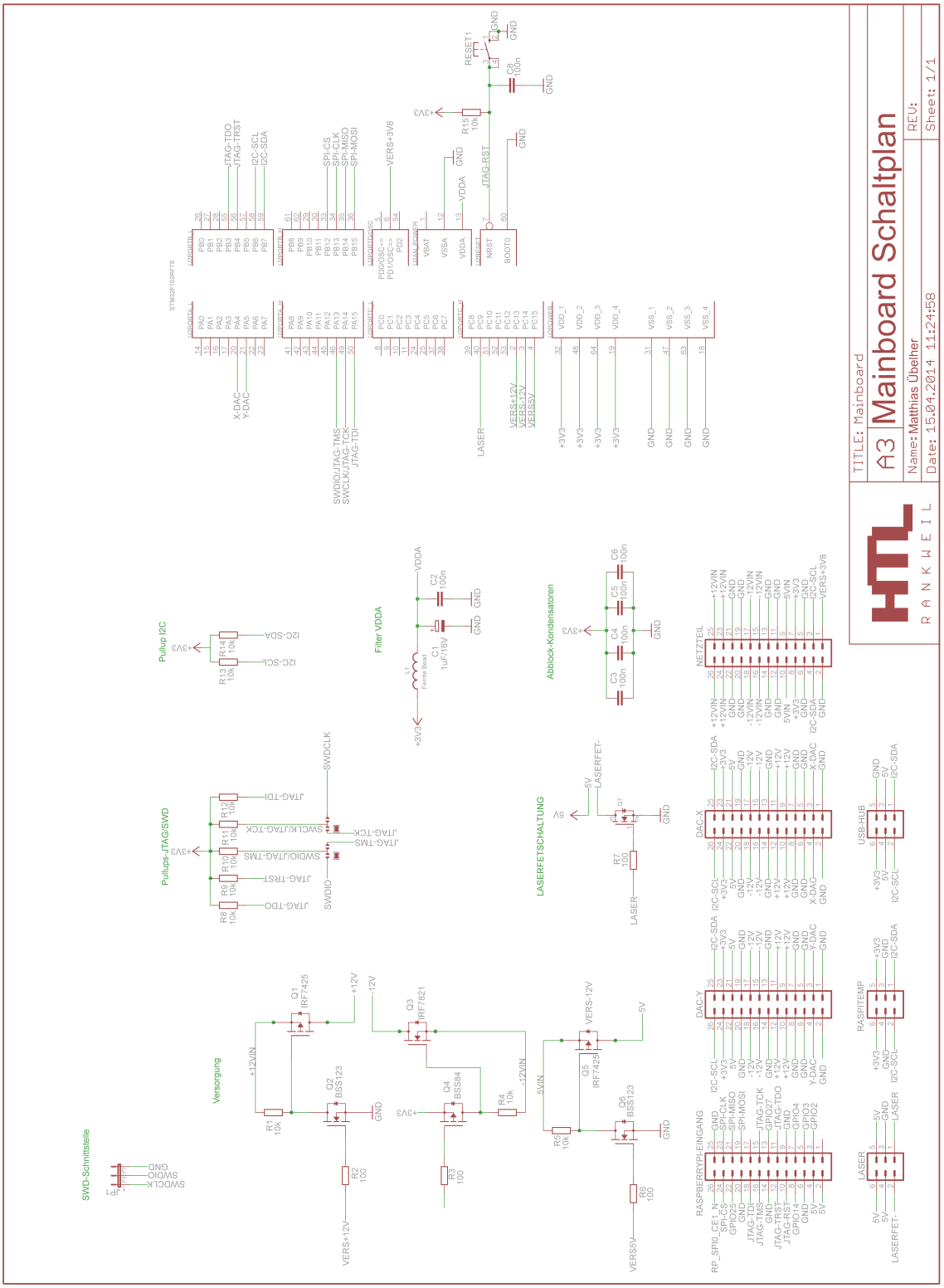
Tabelle 7: Abkürzungsverzeichnis

15 Abbildungsverzeichnis

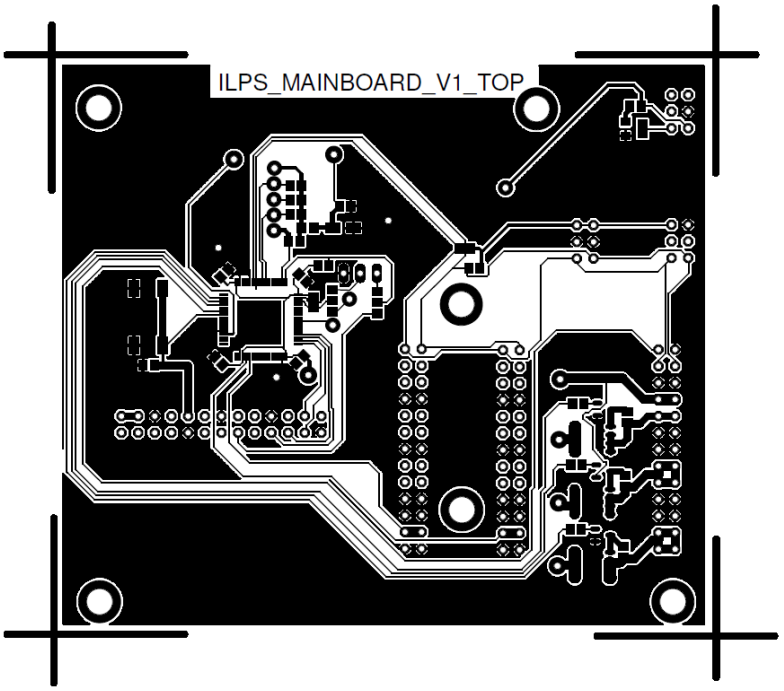
16 Tabellenverzeichnis

17 Anhang

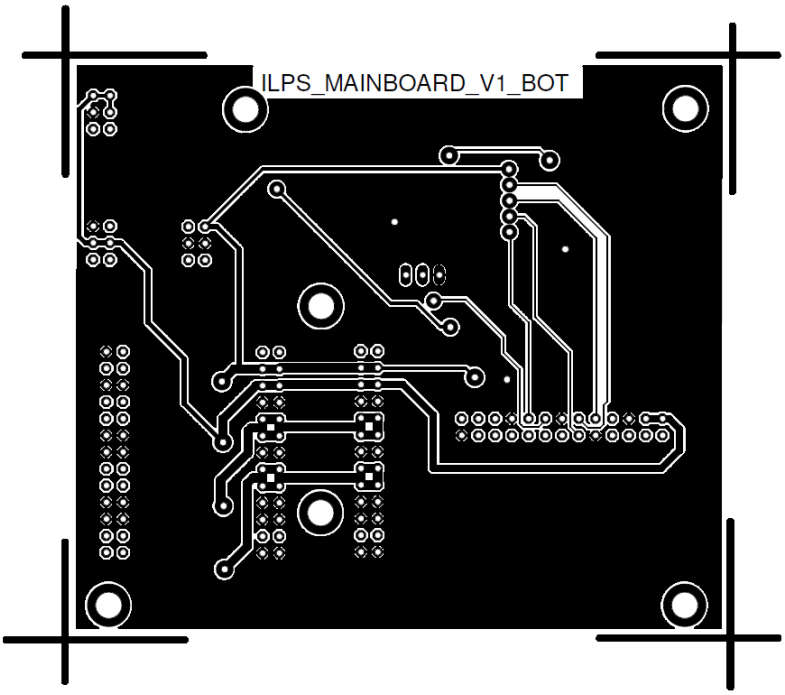
17.1 Mainboard

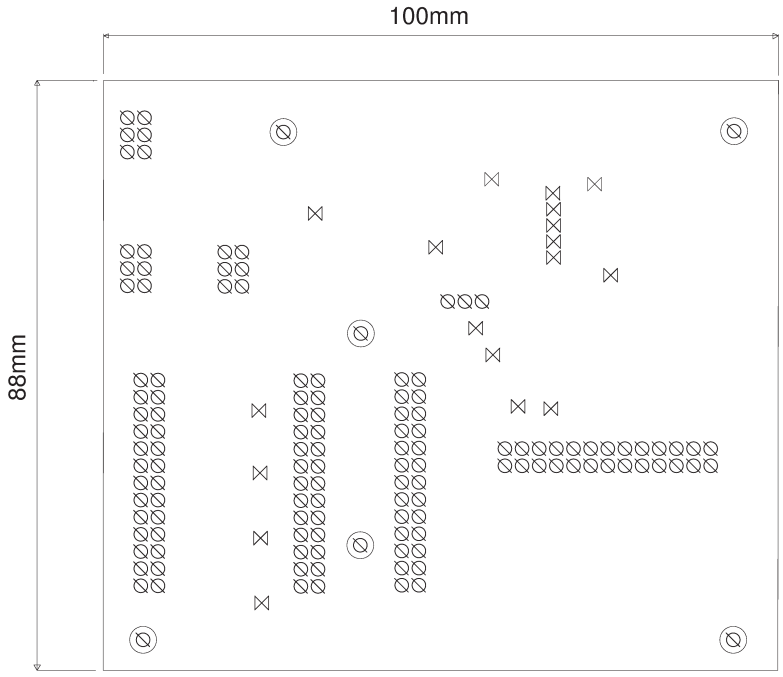


17.1.1 Layout Top



17.1.2 Layout Bottom





Material: FR4 1,5mm 35u Cu
Toleranz: Aussenmass -0,5mm, sonst +/_0,2mm

Bohrplan		
SYM	SIZE	CNT
∅	0.91 mm	125
⋈	1.00 mm	16
∅	4.00 mm	12
	TOTAL	153

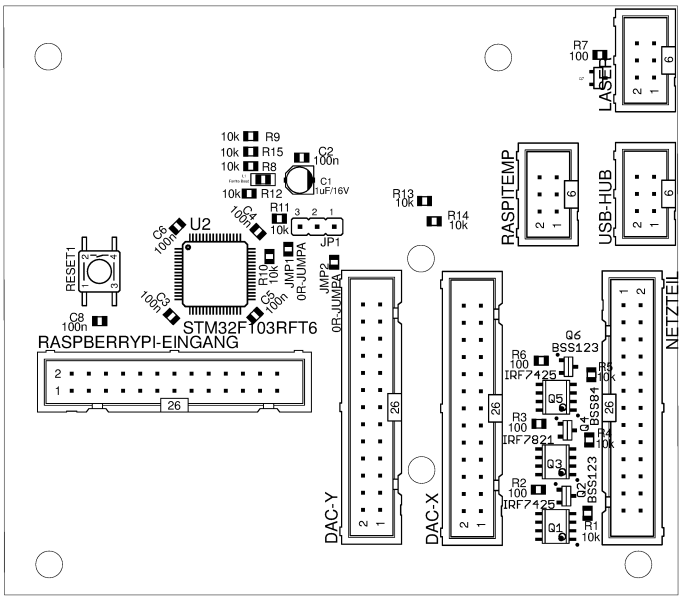
HTL RANKWEIL

TITLE: Mainboard

A4 Bohrplan

Name: Matthias Übelher | REV:

Date: 15.04.2014 17:07:49 | Sheet:



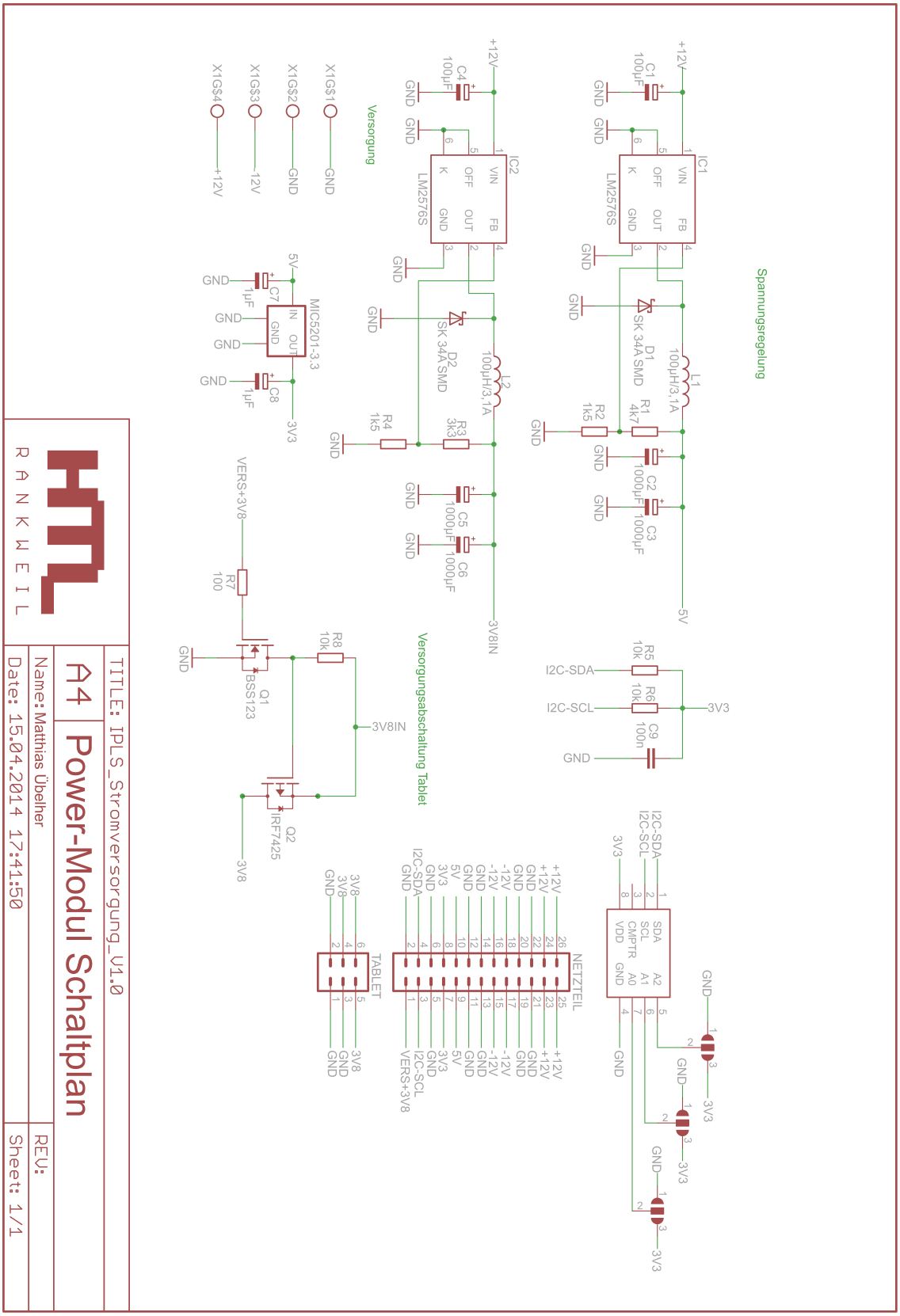
HTL RANKWEIL	TITLE: Mainboard	
	A4 Bestückungsplan	
	Name: Matthias Übelher	REV:
	Date: 29.04.2014 14:02:05	Sheet:

Stückliste - Mainboard

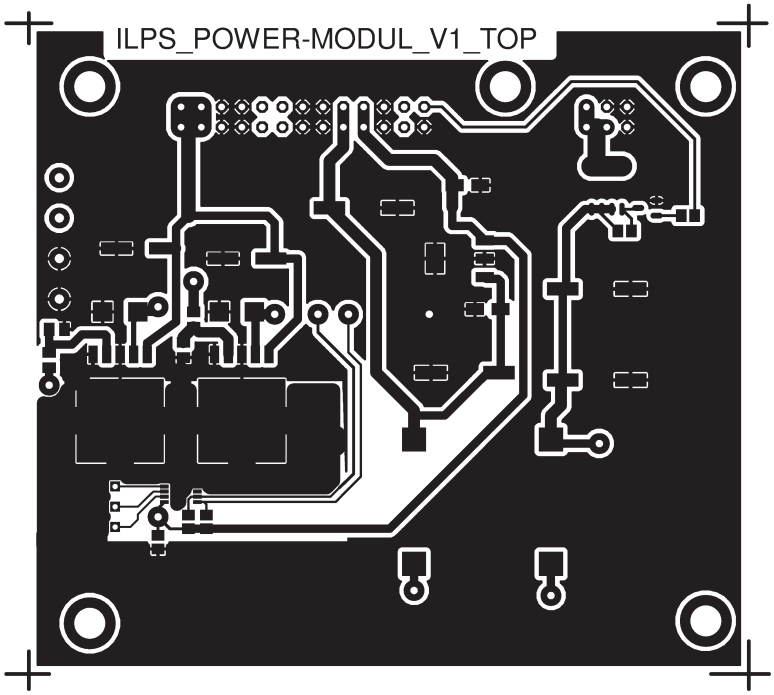
Matthias Übelher ILPS Diplomarbeit 2013/2014

Name	Bezeichnung	Quelle	BestNr	Stück	Stückpreis	Gesamt
R1	0805 10k Ohm Widerstand SMD	Magazin		10	0,01 €	0,10 €
R2	0805 200 Ohm Widerstand SMD	Magazin		2	0,01 €	0,02 €
R3	0805 100 Ohm Widerstand SMD	Magazin		2	0,01 €	0,02 €
R4	0805 10k Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
R5	0805 10k Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
R6	0805 100Ohm Widerstand SMD	Magazin		1	0,01 €	0,01 €
R7	0805 100 Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
R8	0805 10k Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
R9	0805 10k Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
R10	0805 10k Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
R11	0805 10k Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
R12	0805 10k Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
R13	0805 10k Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
R14	0805 10k Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
R15	0805 200 Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
C1	1µF Elko SMD Panasonic	Reichelt	VF 1,0/50 C-B	1	0,15 €	0,15 €
C2	0805 100n Kondensator	Magazin		7	0,01 €	0,07 €
C3	0805 100n Kondensator	Magazin			0,01 €	0,00 €
C4	0805 100n Kondensator	Magazin			0,01 €	0,00 €
C5	0805 100n Kondensator	Magazin			0,01 €	0,00 €
C6	0805 100n Kondensator	Magazin			0,01 €	0,00 €
C7	0805 100n Kondensator	Magazin			0,01 €	0,00 €
C8	0805 100n Kondensator	Magazin			0,01 €	0,00 €
Q1	MOSFET P-Ch 20V 15A, SO-8	Reichelt	IRF 7425	2	0,85 €	1,70 €
Q2	Transistor SMD N-FET SOT-23 100V 0,17A	Reichelt	BSS 123-SMD	2	0,04 €	0,08 €
Q3	MOSFET N-Ch 30V 13,6A, So-8	Reichelt	IRF 7821	1	0,45 €	0,45 €
Q4	Transistor SMD P-FET SOT-23 -60V -0,17A	Reichelt	BSS 84P SMD	1	0,05 €	0,05 €
Q5	MOSFET P-Ch 20V 15A, SO-8	Reichelt	IRF 7425		0,85 €	0,00 €
Q6	Transistor SMD N-FET SOT-23 100V 0,17A	Reichelt	BSS 123-SMD		0,04 €	0,00 €
Q7	IRLML6246 MOSFET	Reichelt	IRLML6246	1	0,14 €	0,14 €
L1	Ferrite Perle 0805	RS	151-5616	1	0,09 €	0,09 €
IC1	STM32F103RFT6	RS	775-2890	1	9,93 €	9,93 €
JP1	3-Pin Jumperstecker	Magazin		1	0,01 €	0,01 €
ST1	26-Pol Wannenbuchse	Magazin		4	0,10 €	0,40 €
ST1	26-Pol Wannenstecker mit Zug und Schubentlastung	Magazin		4	0,10 €	0,40 €
ST2	26-Pol Wannenbuchse	Magazin			0,10 €	0,00 €
ST2	26-Pol Wannenstecker mit Zug und Schubentlastung	Magazin			0,10 €	0,00 €
ST3	26-Pol Wannenbuchse	Magazin			0,10 €	0,00 €
ST3	Schubentlastung	Magazin			0,10 €	0,00 €
ST4	26-Pol Wannenbuchse	Magazin			0,10 €	0,00 €
ST4	Schubentlastung	Magazin			0,10 €	0,00 €
ST5	6-Pol Wannenbuchse	Magazin		3	0,10 €	0,30 €
ST5	Schubentlastung	Magazin		3	0,10 €	0,30 €
ST6	6-Pol Wannenbuchse	Magazin			0,10 €	0,00 €
ST6	Schubentlastung	Magazin			0,10 €	0,00 €
ST7	6-Pol Wannenbuchse	Magazin			0,10 €	0,00 €
ST7	Schubentlastung	Magazin			0,10 €	0,00 €
S1	SMD Taster	Magazin		1	0,01 €	0,01 €
	Kunstoffabstandsbolzen	Magazin		6	0,01 €	0,06 €
	M3x16 Zylinderkopfschrauben	Magazin		6	0,01 €	0,06 €
	Gesamtkosten			**60**		**14,35 €**

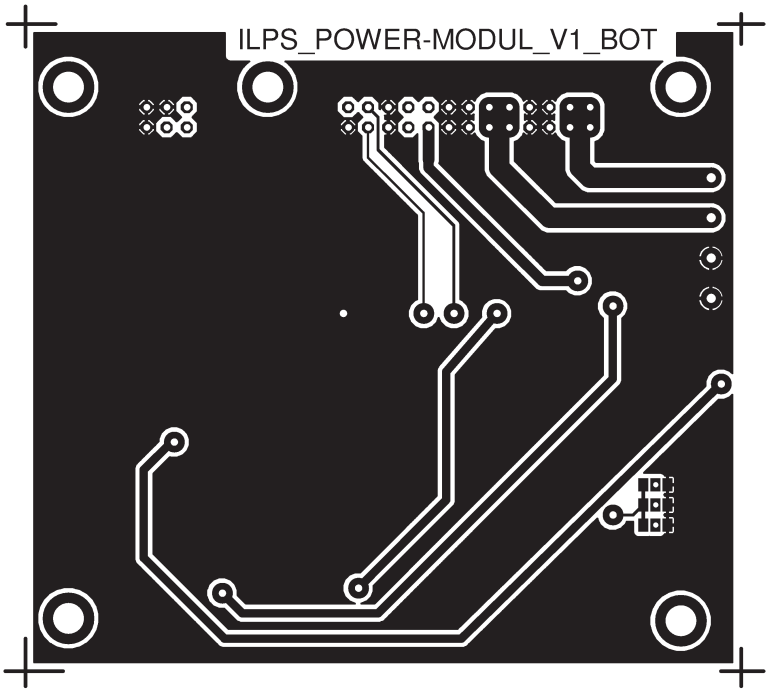
17.2 Power-Modul

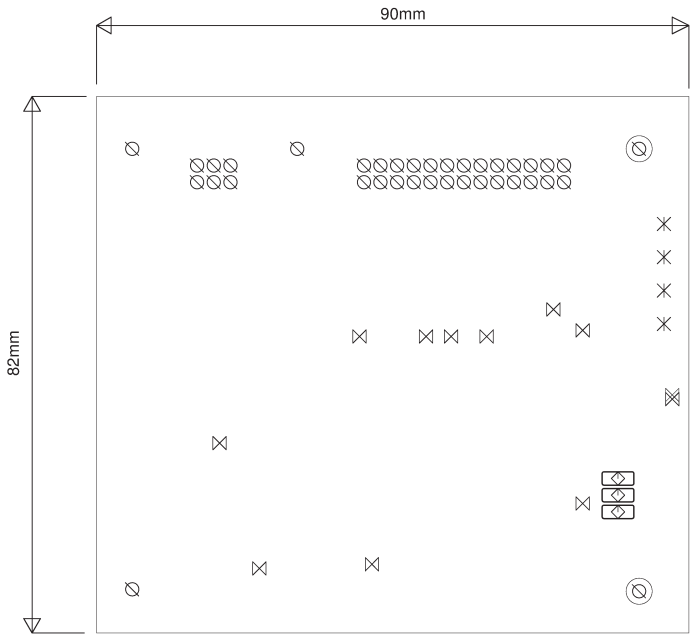


17.2.1 Layout Top



17.2.2 Layout Bottom

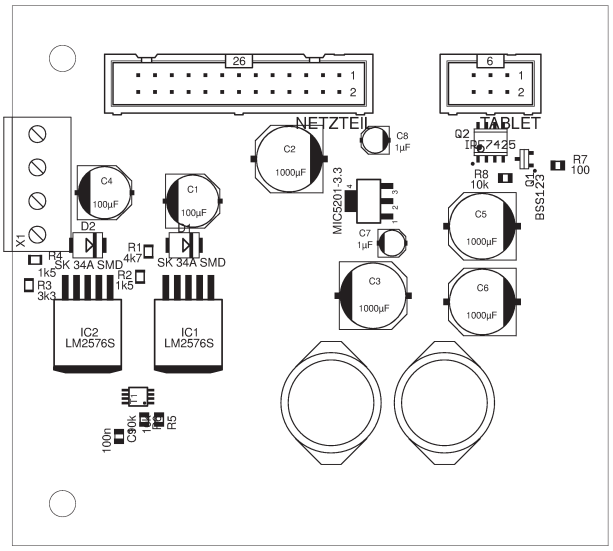




Material: FR4 1,5mm 35u Cu
Toleranz: Aussenmass -0,5mm, sonst +/_0,2mm

Bohrplan		
SYM	SIZE	CNT
◇	0.80 mm	3
∅	0.91 mm	32
⋈	1.00 mm	11
✳	1.30 mm	4
∅	4.00 mm	7
	TOTAL	57

HTL RANKWEIL	TITLE: IPLS_Stromversorgung_V1.0	
	A4 Bohrplan Power-Modul	
	Name: Matthias Übelher	REV:
	Date: 15.04.2014 18:41:59	Sheet:



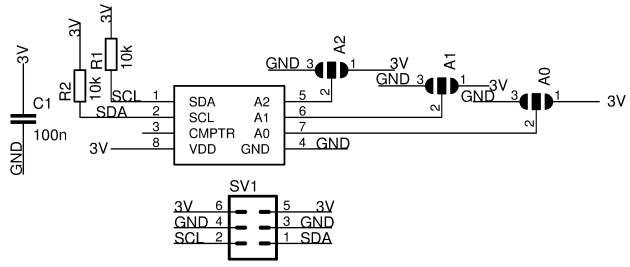
HTL RANKWEIL	TITLE: IPLS_Stromversorgung_V1.0	
	A4 Bestückungsplan Power-Modul	
	Name: Matthias Übelher	REV:
	Date: 15.04.2014 18:37:05	Sheet:

Stückliste - Power-Modul

Matthias Übelher ILPS Diplomarbeit 2013/2014

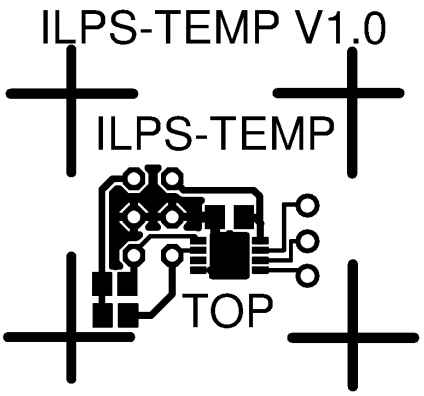
Name	Bezeichnung	Quelle	BestNr	Stück	Stückpreis	Gesamt
R1	0805 4k7 Ohm Widerstand SMD	Magazin		1	0,01 €	0,01 €
R2	0805 1k5 Ohm Widerstand SMD	Magazin		2	0,01 €	0,02 €
R3	0805 3k3 Ohm Widerstand SMD	Magazin		1	0,01 €	0,01 €
R4	0805 1k5 Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
R5	0805 10k Ohm Widerstand SMD	Magazin		3	0,01 €	0,03 €
R6	0805 10k Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
R7	0805 100 Ohm Widerstand SMD	Magazin		1	0,01 €	0,01 €
R8	0805 10k Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
C1	100µF Elko SMD Panasonic E	Reichelt	VF 100/25 K-E	2	0,24 €	0,48 €
C2	1000µF Elko SMD Panasonic G	Reichelt	VF 1000/10 P-G	4	0,58 €	2,32 €
C3	1000µF Elko SMD Panasonic G	Reichelt	VF 1000/10 P-G		0,58 €	0,00 €
C4	100µF Elko SMD Panasonic E	Reichelt	VF 100/25 K-E		0,24 €	0,00 €
C5	1000µF Elko SMD Panasonic G	Reichelt	VF 1000/10 P-G		0,58 €	0,00 €
C6	1000µF Elko SMD Panasonic G	Reichelt	VF 1000/10 P-G		0,58 €	0,00 €
C7	1µF Elko SMD Panasonic B	Reichelt	VF 1,0/50 C-B	2	0,45 €	0,90 €
C8	1µF Elko SMD Panasonic B	Reichelt	VF 1,0/50 C-B		0,45 €	0,00 €
C9	0805 100n Kondensator SMD	Magazin		1	0,01 €	0,01 €
Q1	Transistor SMD N-FET SOT-23 100V 0,17A	Reichelt	BSS 123-SMD	1	0,04 €	0,04 €
Q2	MOSFET P-Ch 20V 15A, SO-8	Reichelt	IRF 7425	1	0,85 €	0,85 €
X1	2-Poliger Platinenanschlusstecker	Magazin		2	0,01 €	0,02 €
IC1	Spannungsregler 3A 1,23-37V 45Vs TO-263-5	Reichelt	LM2576 S-ADJ	2	1,15 €	2,30 €
IC2	Spannungsregler 3A 1,23-37V 45Vs TO-263-5	Reichelt	LM2576 S-ADJ		1,15 €	0,00 €
IC3	LOW Dropout Spannungsregler MIC5201-3V3	Rs	727-4719	1	1,40 €	1,40 €
IC4	TCN75AVUA 12-Bit Temperatursensor	Rs	668-7329	1	0,72 €	0,72 €
L1	Induktivität 100µH/3,1A	Reichelt	L-PISR 100µ	2	0,85 €	1,70 €
L2	Induktivität 100µH/3,1A	Reichelt	L-PISR 100µ		0,85 €	0,00 €
D1	Diode SK 34A SMD Schottky	Reichelt	SK 34A SMD	2	0,18 €	0,36 €
D2	Diode SK 34A SMD Schottky	Reichelt	SK 34A SMD		0,18 €	0,00 €
ST1	26-Pol Wannenbuchse	Magazin		1	0,10 €	0,10 €
ST1	26-Pol Wannenstecker mit Zug und Schubentlastung	Magazin		1	0,10 €	0,10 €
ST2	6-Pol Wannenbuchse	Magazin		1	0,10 €	0,10 €
ST2	6-Pol Wannenstecker mit Zug und Schubentlastung	Magazin		1	0,10 €	0,10 €
	Kunstoffabstandsbolzen	Magazin		5	0,01 €	0,05 €
	M3x16 Zylinderkopfschrauben	Magazin		5	0,01 €	0,05 €
	Gesamtkosten			**43**		**11,68 €**

17.3 Temperatur-Sensor

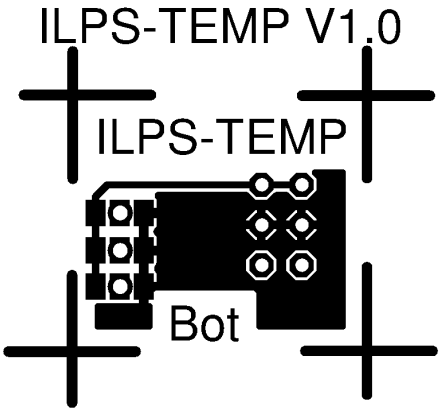


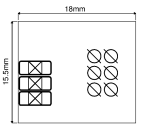
HTL RANKWEIL	TITLE: Temperatursensor	
	A4 Temperatursensor ILPS	
	Name: Matthias Übelher	REV:
	Date: 15.04.2014 20:14:58	Sheet: 1/1

17.3.1 Layout Top



17.3.2 Layout Bottom





Material: FR4 1,5mm 35u Cu
Toleranz: Aussenmass -0,5mm, sonst +/_0,2mm

Bohrplan		
SYM	SIZE	CNT
∅	0.91 mm	6
⋈	1.00 mm	3
	TOTAL	9

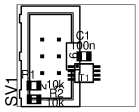
HTL RANKWEIL

TITLE: Temperatursensor

A4 | Bohrplan Temperatursensor

Name: Matthias Übelher | REV:

Date: 15.04.2014 20:26:52 | Sheet:



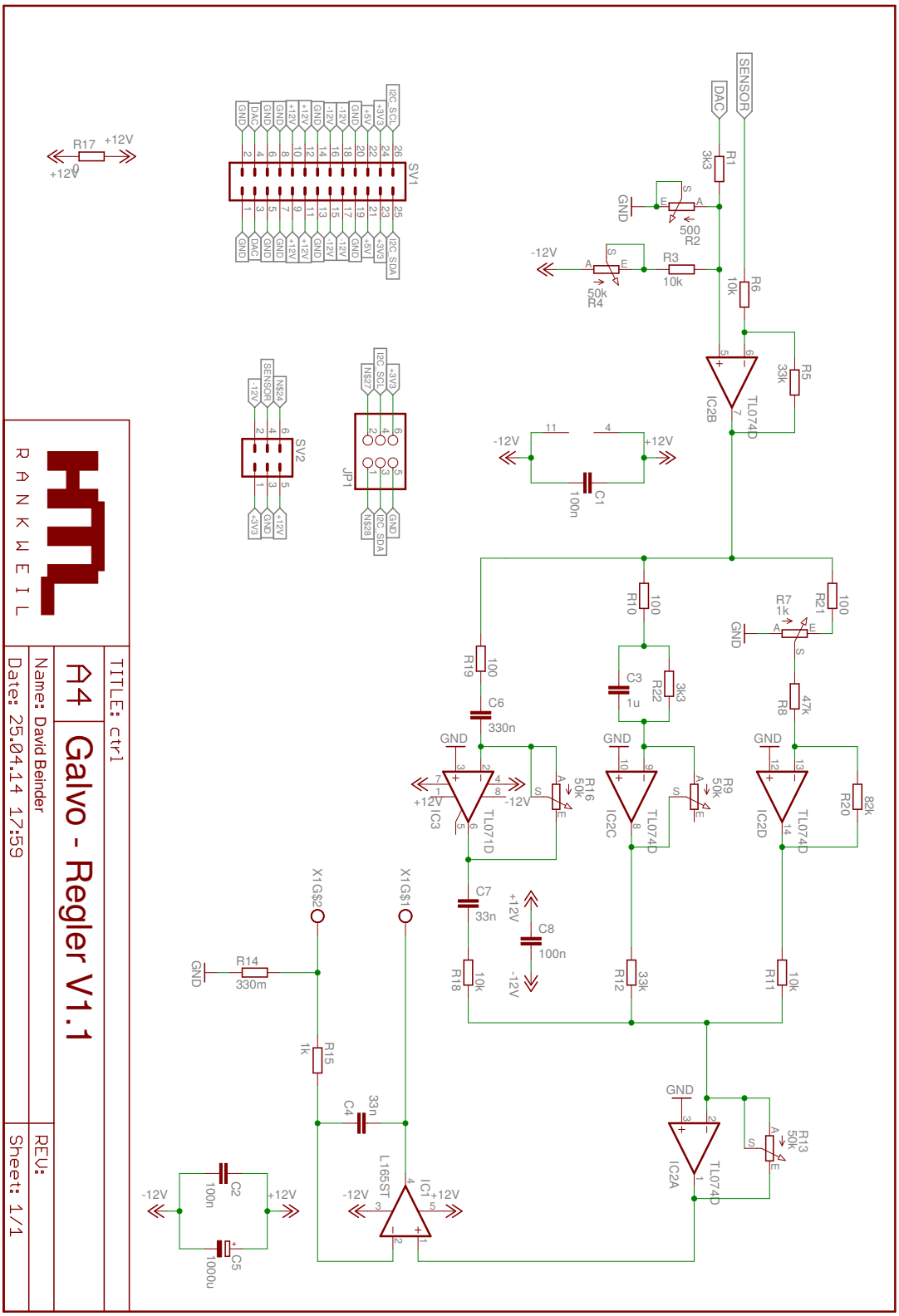
HTL RANKWEIL	TITLE: Temperatursensor	
	A4 Bestückungsplan Temperatursensor	
	Name: Matthias Übelher	REV:
	Date: 15.04.2014 20:19:32	Sheet:

Stückliste - Temperatursensor

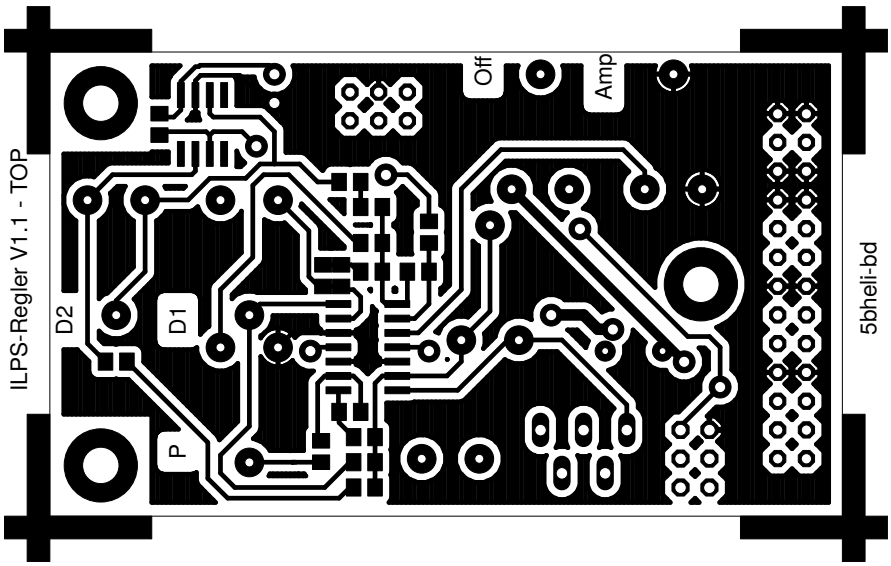
Matthias Übelher ILPS Diplomarbeit 2013/2014

Name	Bezeichnung	Quelle	BestNr	Stück	Stückpreis	Gesamt
R1	0805 10k Ohm Widerstand SMD	Magazin		2	0,01 €	0,02 €
R2	0805 10k Ohm Widerstand SMD	Magazin			0,01 €	0,00 €
C1	0805 100n Kondensator	Magazin		1	0,01 €	0,01 €
SV1	6-Pol Wannenbuchse	Magazin		1	0,10 €	0,10 €
SV1	Schubentlastung	Magazin		1	0,10 €	0,10 €
IC1	TCN75AVUA 12-Bit Temperatursensor	Rs	668-7329	1	0,72 €	0,72 €
	Gesamtkosten			**6**		**0,95 €**

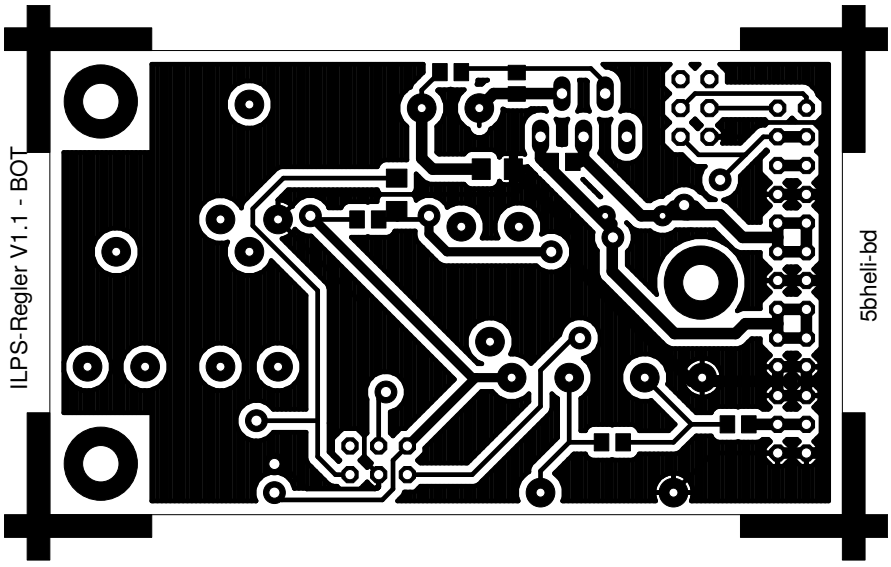
17.4 Galvo-Regler

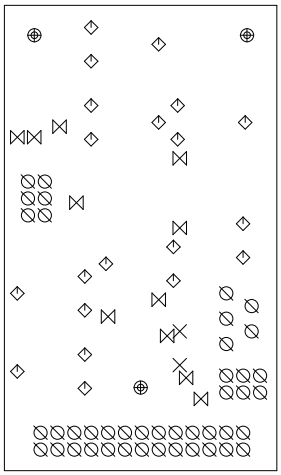


17.4.1 Layout Top



17.4.2 Layout Bottom





Material: FR4 1,5mm 35u Cu
Toleranz: Aussenmass -0,5mm, sonst +/_0,2mm

DRILL PLAN

SYM	SIZE	TOL.	CNT	PLATE
	0.60 mm	+/-0.08 mm	2	Yes
	0.80 mm	+/-0.08 mm	20	Yes
	0.91 mm	+/-0.08 mm	32	Yes
	1.00 mm	+/-0.08 mm	11	Yes
	1.02 mm	+/-0.08 mm	11	Yes
	3.20 mm	+/-0.08 mm	3	Yes
		TOTAL	79	79

HTL RANKWEIL

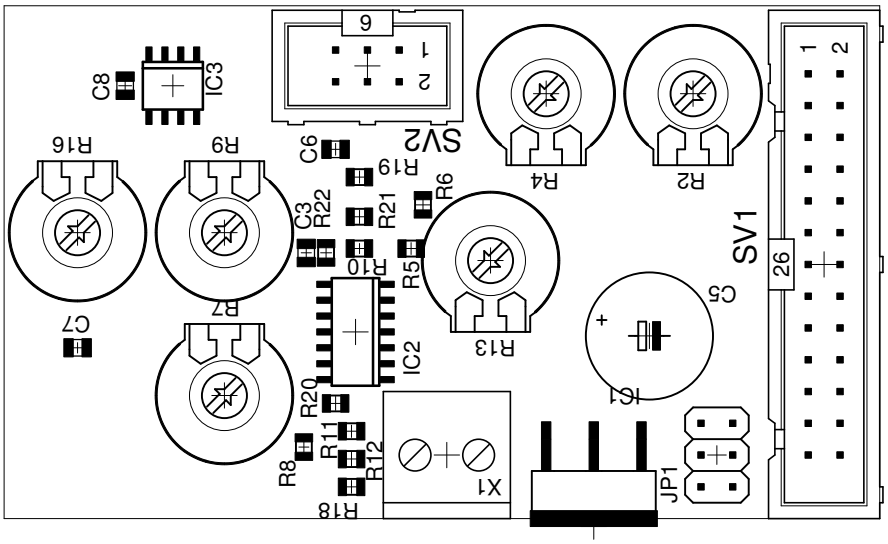
TITLE: ctrl

A4 | Bohrplan Regler V1.1

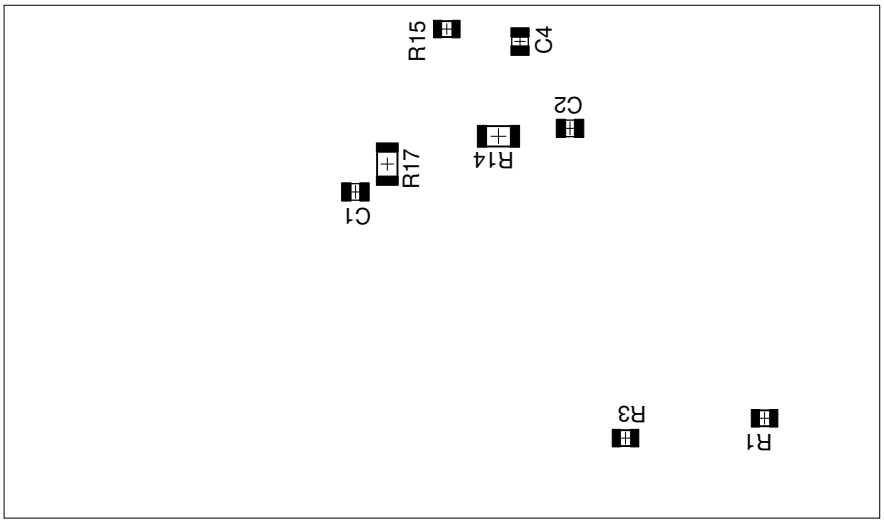
Name: David Beinder | REV:

Date: 29.04.14 14:51 | Sheet:

17.4.3 Bestückungsplan Top



17.4.4 Bestückungsplan Bottom

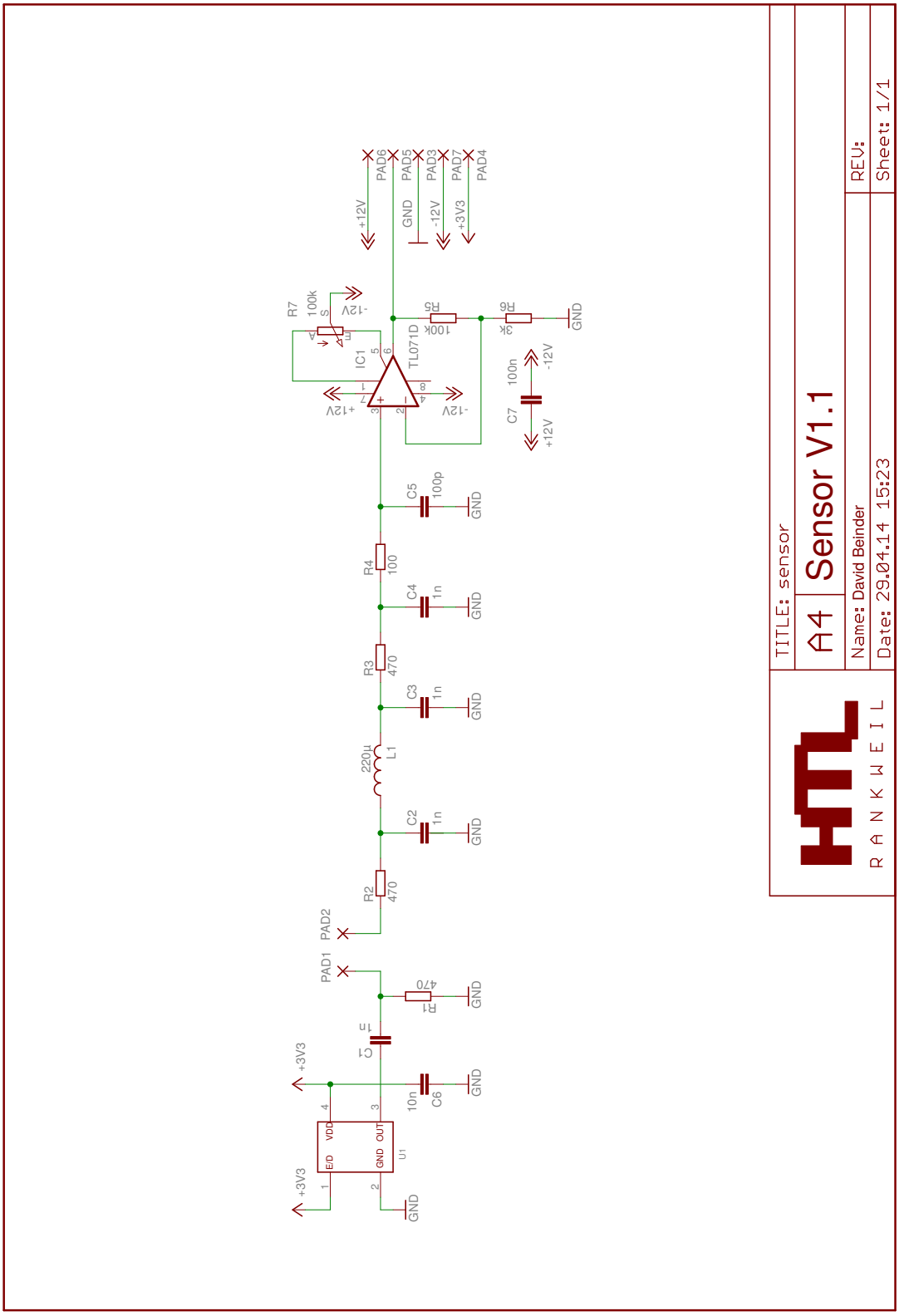


Stückliste - Regler

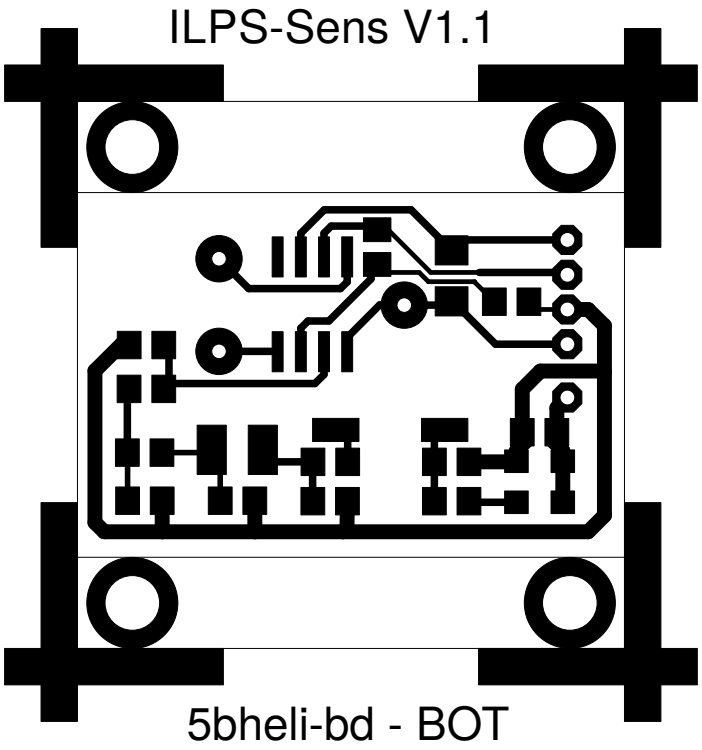
5bHELi ILPS Diplomarbeit 2013/2014

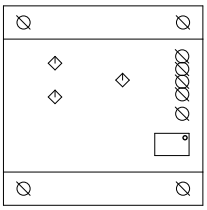
Name	Bezeichnung	Quelle	BestNr	Stück	Stückpreis	Gesamt
R1	0805 1% 3k3	HTL-Magzin		2	0,01 €	0,02 €
R2	Poti 1cm 500R	HTL-Magzin		1	0,10 €	0,10 €
R3	0805 1% 10k	HTL-Magzin		4	0,01 €	0,04 €
R4	Poti 1cm 50k	HTL-Magzin		4	0,10 €	0,40 €
R5	0805 1% 33k	HTL-Magzin		2	0,01 €	0,02 €
R6	0805 1% 10k	HTL-Magzin			0,01 €	
R7	Poti 1cm 1k	HTL-Magzin		1	0,10 €	0,10 €
R8	0805 1% 47k	HTL-Magzin		1	0,01 €	0,01 €
R9	Poti 1cm 50k	HTL-Magzin			0,10 €	
R10	0805 1% 100R	HTL-Magzin		3	0,01 €	0,03 €
R11	0805 1% 10k	HTL-Magzin			0,01 €	
R12	0805 1% 33k	HTL-Magzin			0,01 €	
R13	Poti 1cm 50k	HTL-Magzin			0,10 €	
R14	1206 1% 330mR	HTL-Magzin		1	0,02 €	0,02 €
R15	0805 1% 1k	HTL-Magzin		1	0,01 €	0,01 €
R16	Poti 1cm 50k	HTL-Magzin			0,10 €	
R17	1206 1% 0R	HTL-Magzin		1	0,02 €	0,02 €
R18	0805 1% 10k	HTL-Magzin			0,01 €	
R19	0805 1% 100R	HTL-Magzin			0,01 €	
R20	0805 1% 82k	HTL-Magzin		1	0,01 €	0,01 €
R21	0805 1% 100R	HTL-Magzin			0,01 €	
R22	0805 1% 3k3	HTL-Magzin			0,01 €	
C1	0805 100n	HTL-Magzin		3	0,01 €	0,03 €
C2	0805 100n	HTL-Magzin			0,01 €	
C3	0805 1u	HTL-Magzin		1	0,01 €	0,01 €
C4	0805 33n	HTL-Magzin		2	0,01 €	0,02 €
C5	Elko RM5 105° 1000u	HTL-Magzin		1	0,10 €	0,10 €
C6	0805 330n	HTL-Magzin		1	0,01 €	0,01 €
C7	0805 33n	HTL-Magzin			0,01 €	
C8	0805 100n	HTL-Magzin			0,01 €	
IC1	L165	Reichelt	L165	1	3,53 €	3,53 €
IC2	TL074CD	Reichelt	TL074CD	1	0,34 €	0,34 €
IC3	TL071CD	Reichelt	TL071CD	1	0,31 €	0,31 €
X1	Schraubklemme 2pin	HTL-Magzin		1	0,10 €	0,10 €
SV1	Wannenbuchse 2x13	HTL-Magzin		1	0,10 €	0,10 €
SV2	Wannenbuchse 2x3	HTL-Magzin		1	0,10 €	0,10 €
JP1	Stiftleiste 2x3	HTL-Magzin		1	0,10 €	0,10 €
	Platine 70x41mm zweiseitig	HTL-Magzin		1	1,00 €	1,00 €
	Schraube M3x6	HTL-Magzin		6	0,10 €	0,60 €
	Gesamtkosten					**7,13 €**

17.5 Galvo-Sensor



17.5.1 Layout Bottom





Material: FR4 1,5mm 35u Cu
Toleranz: Aussenmass -0,5mm, sonst +/_0,2mm

DRILL PLAN				
SYM	SIZE	TOL.	CNT	PLATE
◈	0.80 mm	+/-0.08 mm	3	Yes
⊘	0.81 mm	+/-0.08 mm	5	Yes
⊘	3.00 mm	+/-0.08 mm	4	Yes
		TOTAL	12	12

HTL RANKWEIL

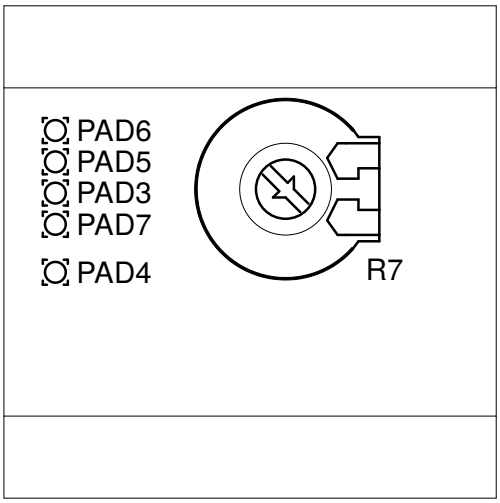
TITLE: sensor

A4 | Bohrplan Sensor V1.1

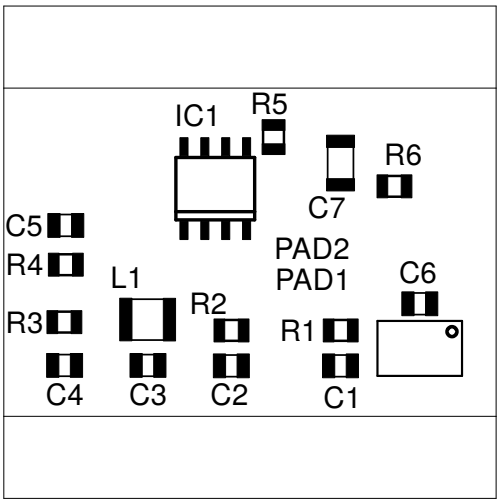
Name: David Beinder | REV:

Date: 29.04.14 15:56 | Sheet:

17.5.2 Bestückungsplan Top



17.5.3 Bestückungsplan Bottom



Stückliste - Sensor

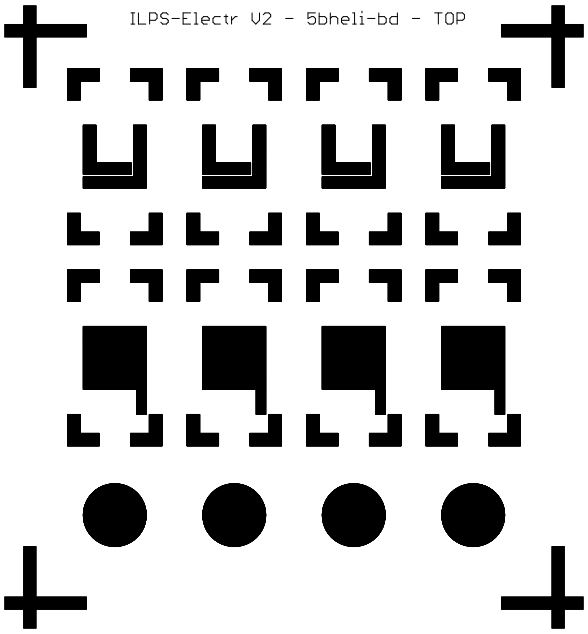
5bHELi ILPS Diplomarbeit 2013/2014

Name	Bezeichnung	Quelle	BestNr	Stück	Stückpreis	Gesamt
R1	0805 1% 470	HTL-Magzin		3	0,01 €	0,03 €
R2	0805 1% 470	HTL-Magzin				
R3	0805 1% 470	HTL-Magzin				
R4	0805 1% 100	HTL-Magzin		1	0,01 €	0,01 €
R5	0805 1% 3k3	HTL-Magzin		1	0,01 €	0,01 €
R6	0805 1% 100k	HTL-Magzin		1	0,01 €	0,01 €
R7	Poti 1mm 100k	HTL-Magzin		1	0,10 €	0,10 €
C1	0805 1n	HTL-Magzin		4	0,01 €	0,04 €
C2	0805 1n	HTL-Magzin				
C3	0805 1n	HTL-Magzin				
C4	0805 1n	HTL-Magzin				
C5	0805 100p	HTL-Magzin		1	0,01 €	0,01 €
C6	0805 10n	HTL-Magzin		1	0,01 €	0,01 €
C7	0805 100n	HTL-Magzin		1	0,01 €	0,01 €
IC1	OpAmp TL071CD	Reichelt	TL071CD	1	0,31 €	0,31 €
U1	FXO-HC536R-16MHz / 14MHz	Farnell	672-1289	1	1,50 €	1,50 €
	Platine 30x30mm einseitig	HTL		1	1,00 €	1,00 €
	Schraube M3x6 Kunstoff	HTL-Magzin		4	0,10 €	0,40 €
	Gesamtkosten					**3,44 €**

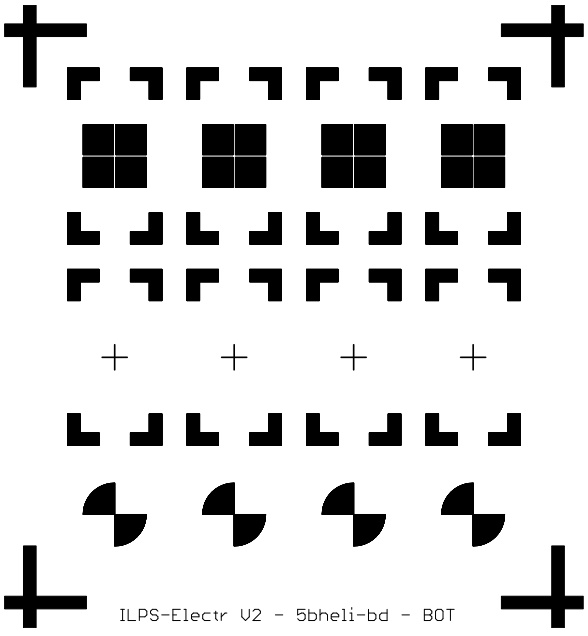
17.6 Galvo-Kondensatorplatinen

Hier wurden sicherheitshalber die doppelte Anzahl der benötigten Platten hergestellt.

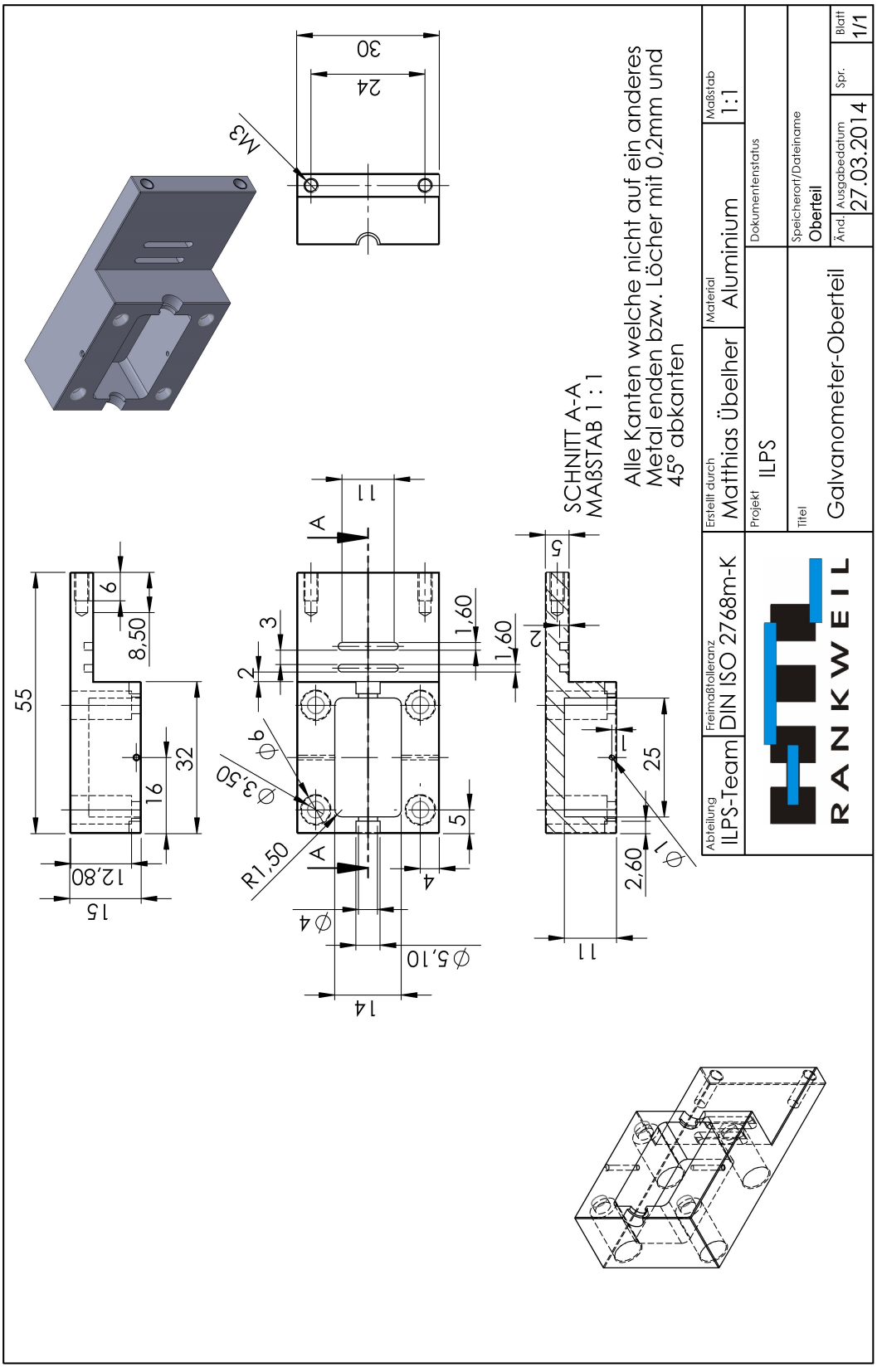
17.6.1 Layout Top

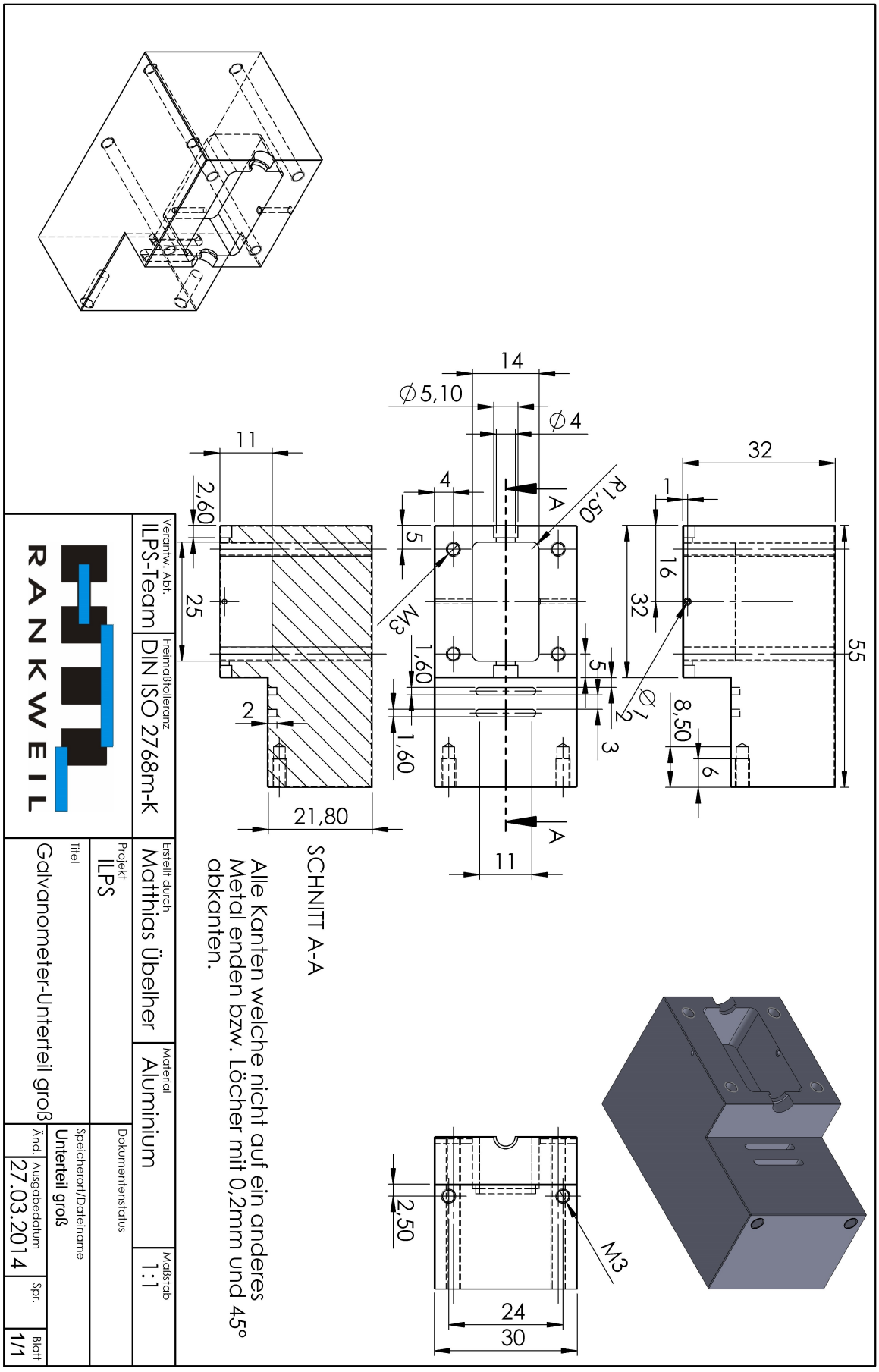


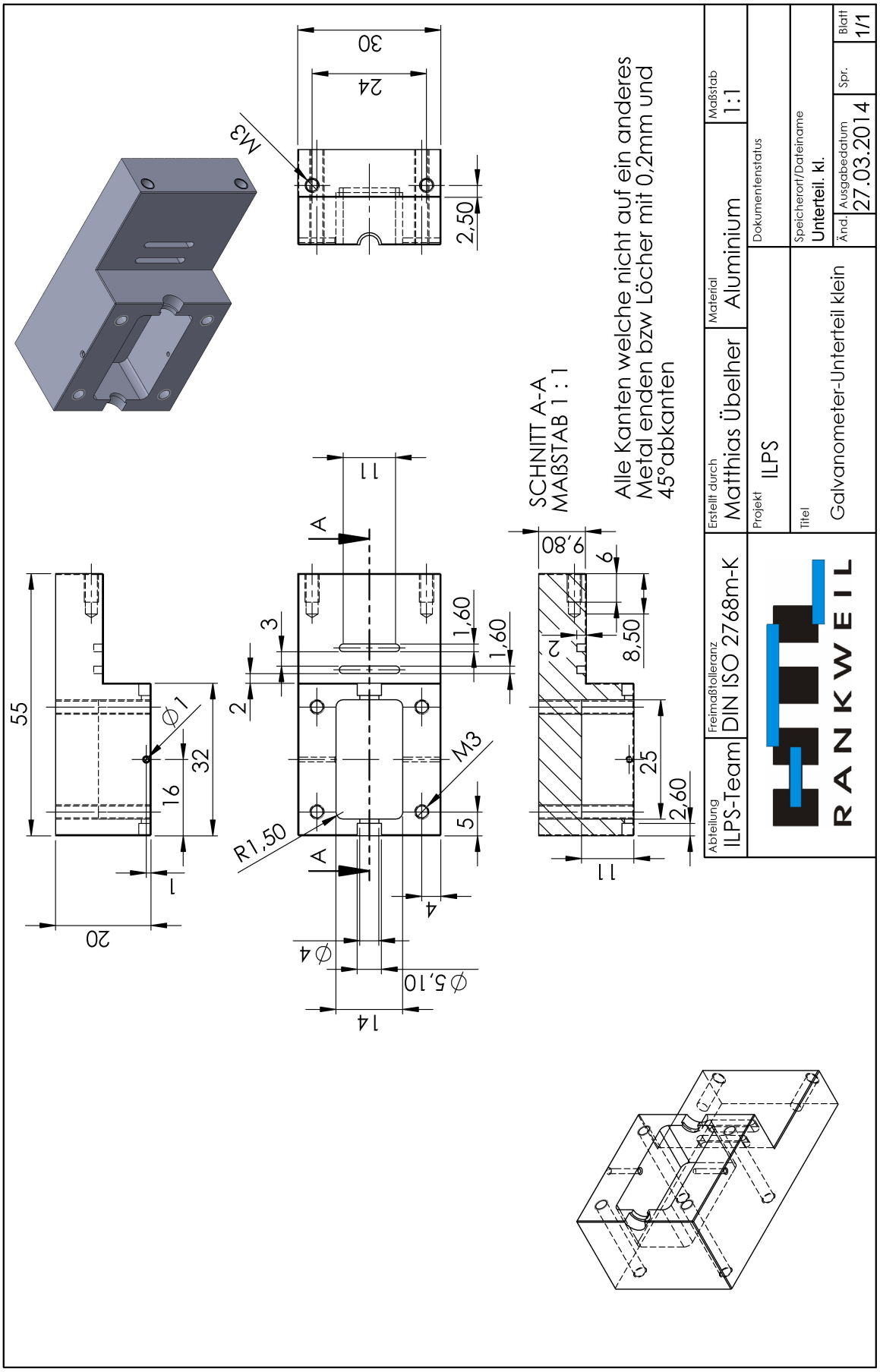
17.6.2 Layout Bottom

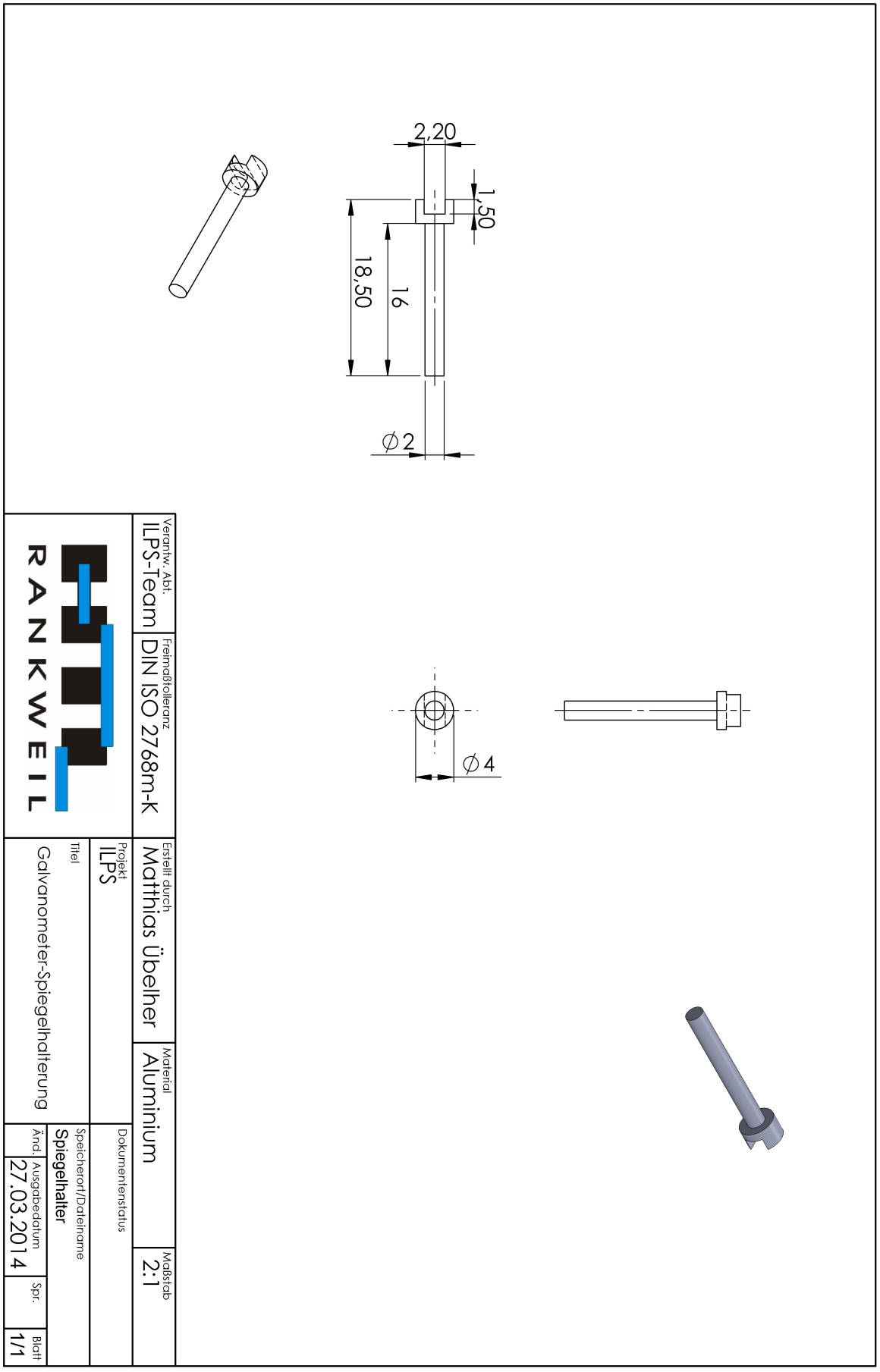


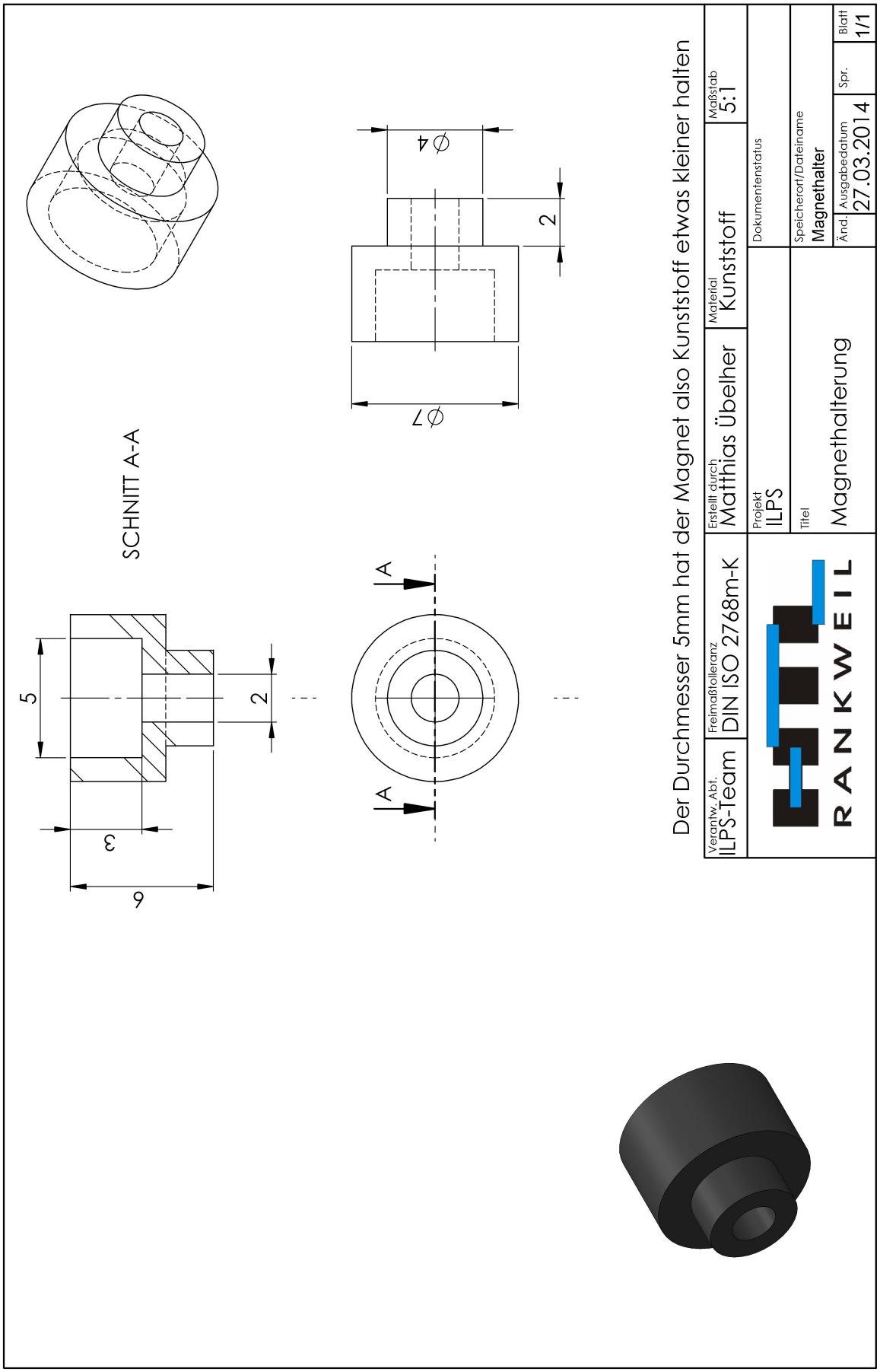
17.7 Konstruktionspläne

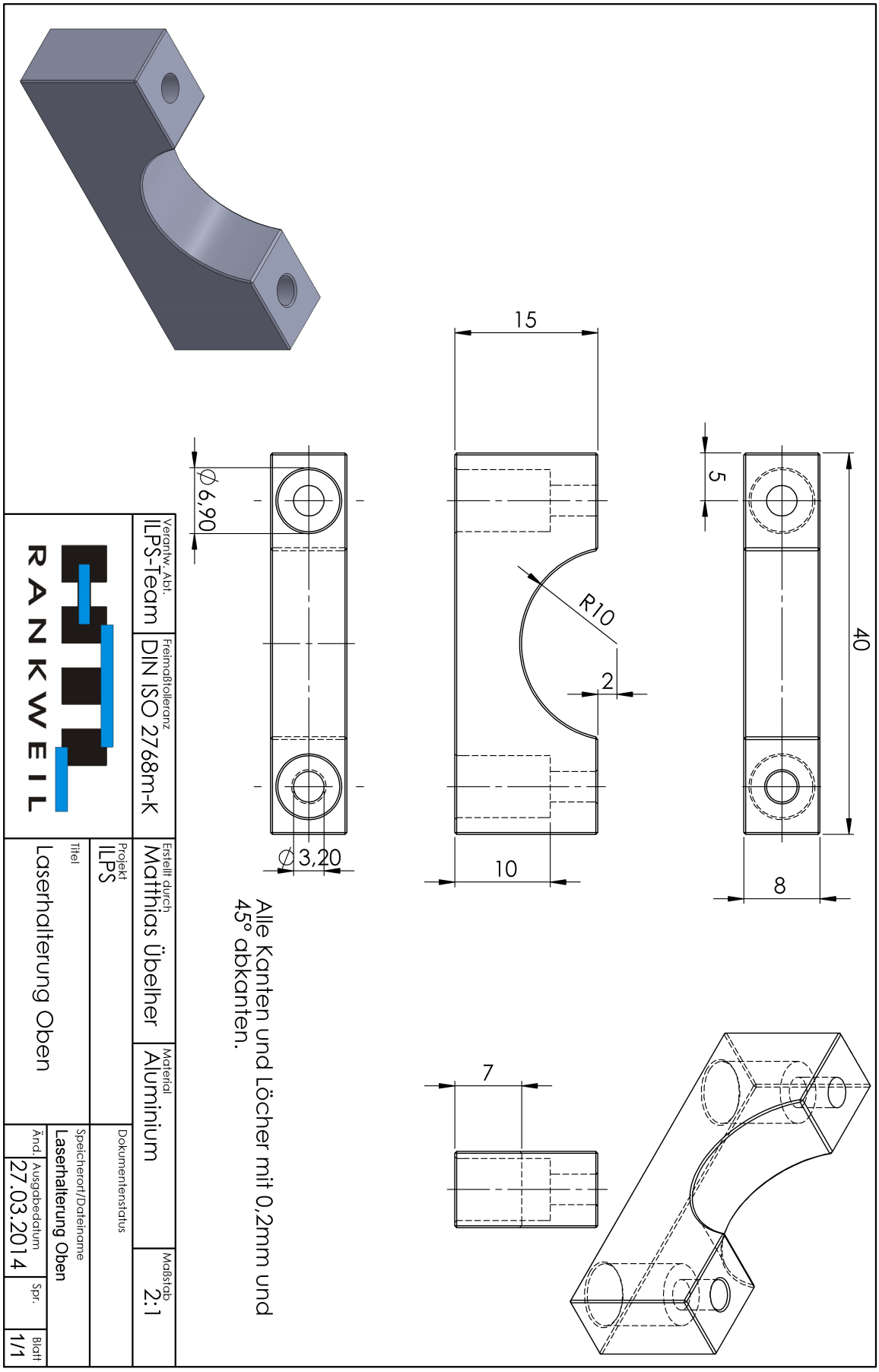


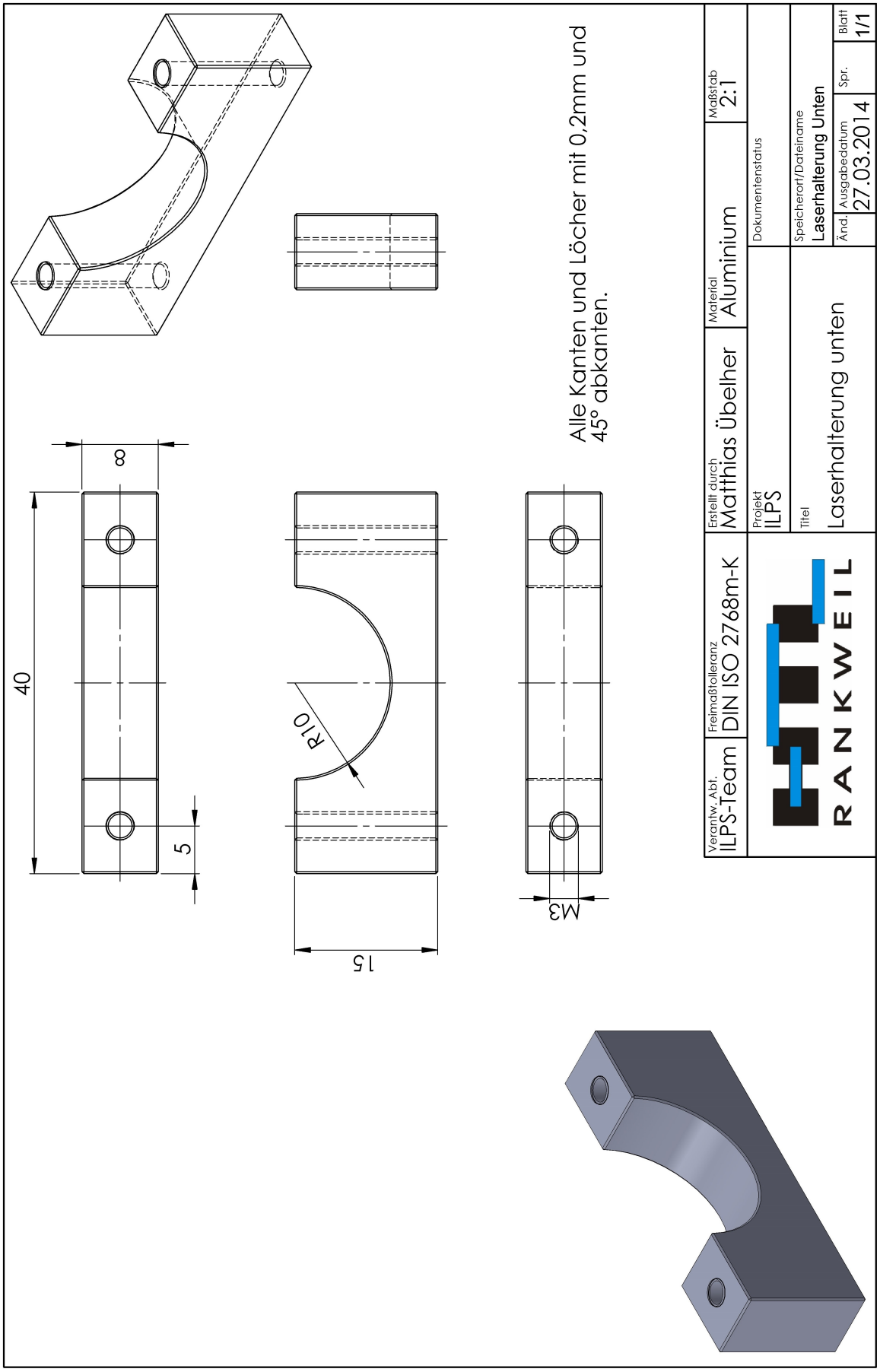

14
∅5,10
∅4
11
2,60
4
A
R1,50
5
25
1,60
2
21,80
11
A
32
1
16
32
55
8,50
6
3
SCHNITT A-A
Alle Kanten welche nicht auf ein anderes Metal enden bzw. Löcher mit 0,2mm und 45° abkanten.
2,50
M3
24
30
RANKWEIL
Verantw. Abt.
ILPS-Team
Freimaßtoleranz
DIN ISO 2768m-K
Erstellt durch
Matthias Übelher
Material
Aluminium
Maßstab
1:1
Projekt
ILPS
Dokumentenstatus
Titel
Galvanometer-Unterteil groß
Änd.
Ausgabedatum
27.03.2014
Spr.
Speicherort/Dateiname
Unterteil groß
Blatt
1/1

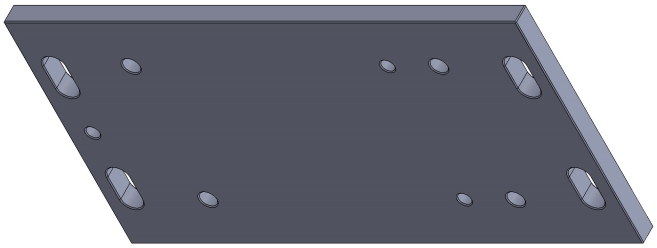

30
24
M3
2,50
SCHNITT A-A
MAßSTAB 1 : 1
Alle Kanten welche nicht auf ein anderes
Metal enden bzw Löcher mit 0,2mm und
45°abkanten
11
A
3
1,60
1,60
2
M3
R1,50
5
4
Ø4
Ø5,10
14
9,80
6
2
8,50
25
2,60
11
55
Ø1
32
16
1
20
Abteilung
ILPS-Team
Freimaßtoleranz
DIN ISO 2768m-K
Erstellt durch
Matthias Übelher
Material
Aluminium
Maßstab
1:1
Projekt
ILPS
Dokumentenstatus
Titel
Galvanometer-Unterteil klein
Speicherort/Dateiname
Unterteil. kl.
Änd.
Ausgabedatum
27.03.2014
Spr.
Blatt
1/1
RANKWEIL

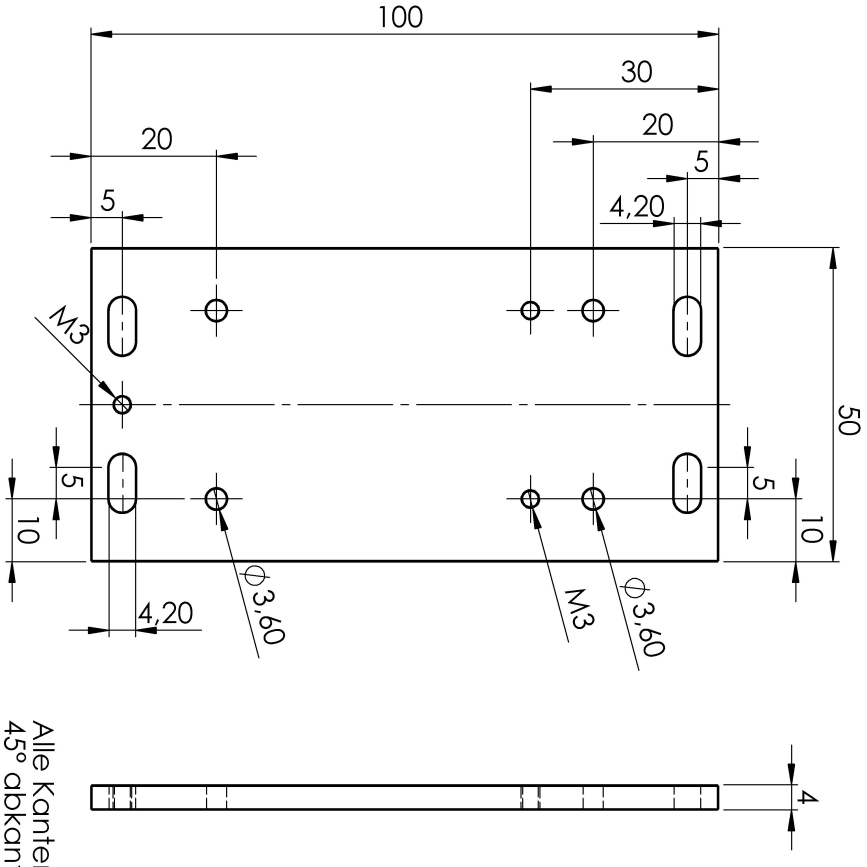

2,20
1,50
18,50
16
∅2
∅4
RANKWEIL
Verantw. Abt.
ILPS-Team
Freimaßtoleranz
DIN ISO 2768m-K
Erstellt durch
Matthias Übelher
Material
Aluminium
Maßstab
2:1
Projekt
ILPS
Dokumentenstatus
Titel
Galvanometer-Spiegelhalterung
Speicherort/Dateiname
Spiegelhalter
Änd.
Ausgabedatum
27.03.2014
Spr.
Blatt
1/1


SCHNITT A-A
5
2
3
6
9
A
A
Ø4
2
Ø7
Der Durchmesser 5mm hat der Magnet also Kunststoff etwas kleiner halten
Verantw. Abt.
ILPS-Team
Freimaßtoleranz
DIN ISO 2768m-K
Erstellt durch
Matthias Übelher
Material
Kunststoff
Maßstab
5:1
Projekt
ILPS
Dokumentenstatus
Titel
Magnethalterung
Speicherort/Dateiname
Magnethalter
Änd.
Ausgabedatum
27.03.2014
Spr.
Blatt
1/1
RANKWEIL


15
5
40
R10
2
10
8
⌀6,90
⌀3,20
7
Alle Kanten und Löcher mit 0,2mm und
45° abkanten.
RANKWEIL
Verantw. Abt.
ILPS-Team
Freimaßtoleranz
DIN ISO 2768m-K
Erstellt durch
Matthias Übelher
Material
Aluminium
Maßstab
2:1
Projekt
ILPS
Dokumentenstatus
Titel
Laserhalterung Oben
Speicherort/Dateiname
Laserhalterung Oben
Änd.
Ausgabedatum
27.03.2014
Spr.
Blatt
1/1

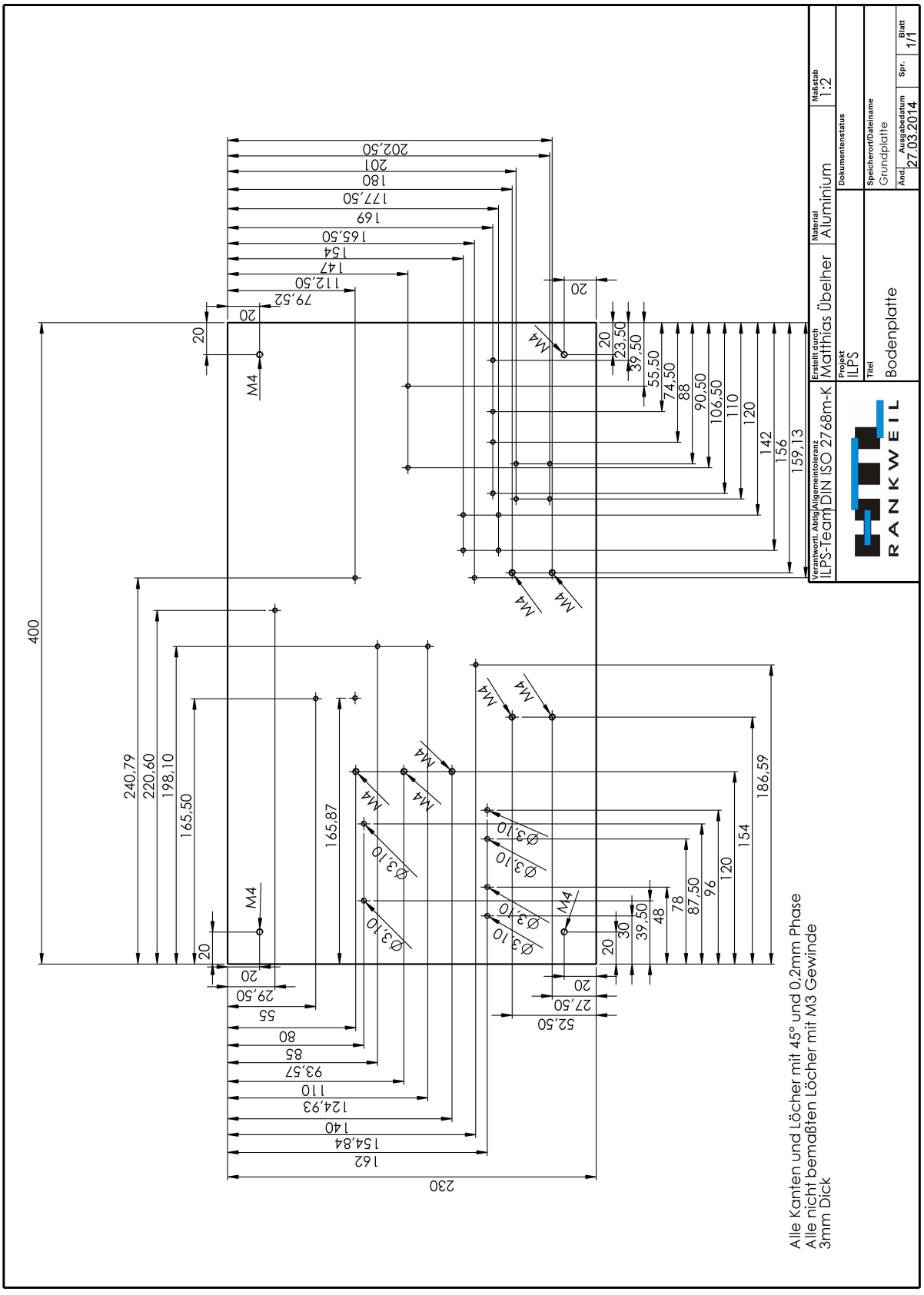

Alle Kanten und Löcher mit 0,2mm und 45° abkanten.
8
40
5
R10
15
M3
Verantw. Abt.
ILPS-Team
Freimaßtoleranz
DIN ISO 2768m-K
Erstellt durch
Matthias Übelher
Material
Aluminium
Maßstab
2:1
Projekt
ILPS
Dokumentenstatus
Titel
Laserhalterung unten
Speicherort/Dateiname
Laserhalterung Unten
Änd.
Ausgabedatum
27.03.2014
Spr.
Blatt
1/1
RANKWEIL

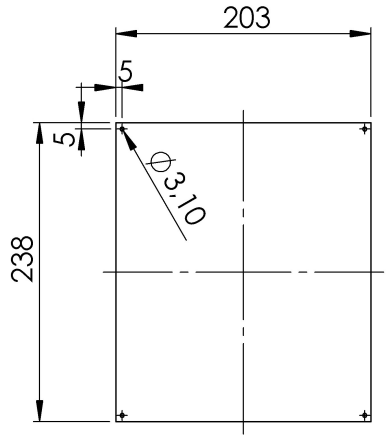




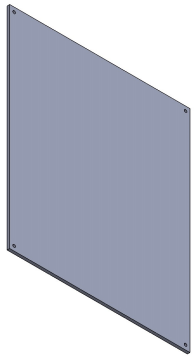
Alle Kanten und Löcher mit 0,2mm und 45° abkanten.

Verantw. Abt. ILPS-Team	Freimaßtoleranz DIN ISO 2768m-K	Erstellt durch Matthias Übelher	Material Aluminium	Maßstab 2:1
RANKWEIL	Projekt ILPS		Dokumentenstatus	
	Titel Laser Befestigungsplatte		Speicherort/Dateiname Laserhalterung Befestigung	
		Änd. Ausgabedatum 27.03.2014	Spr.	Blatt 1/1


Alle Kanten und Löcher mit 45° und 0,2mm Phase
Alle nicht bemaßten Löcher mit M3 Gewinde
3mm Dick
Verantwortl. Abtlg/Allgemeintoleranz
ILPS-Team DIN ISO 2768m-K
Erstellt durch
Matthias Übelher
Material
Aluminium
Maßstab
1:2
Projekt
ILPS
Titel
Bodenplatte
Dokumentenstatus
Speicherort/Dateiname
Grundplatte
And.
Ausgabedatum
27.03.2014
Spr.
Blatt
1/1
RANKWEIL



3mm Dick



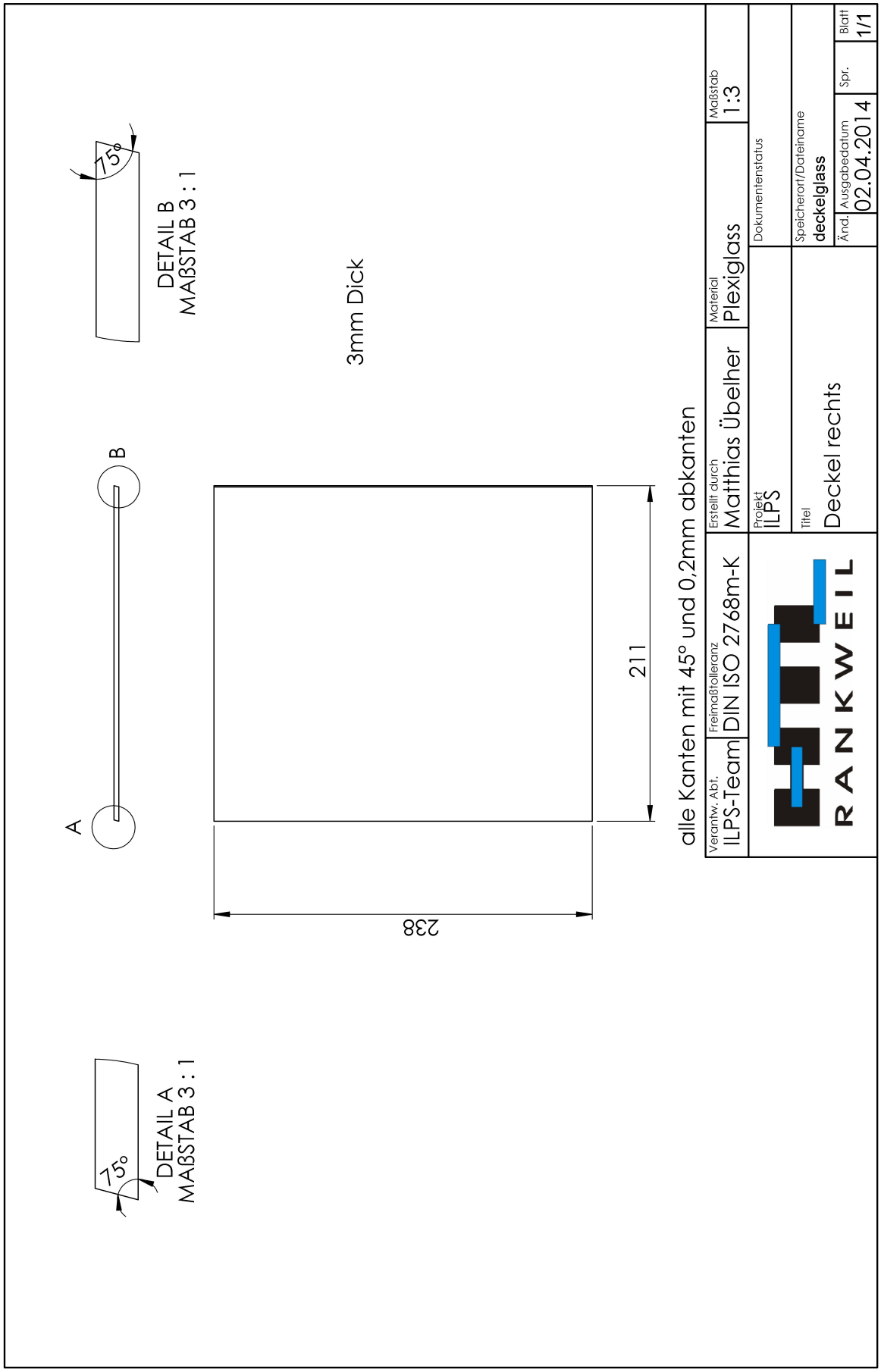
Alle Kanten mit 45° und 0,2mm abkanten, und Löcher für eine M3 Schraube vorsenken

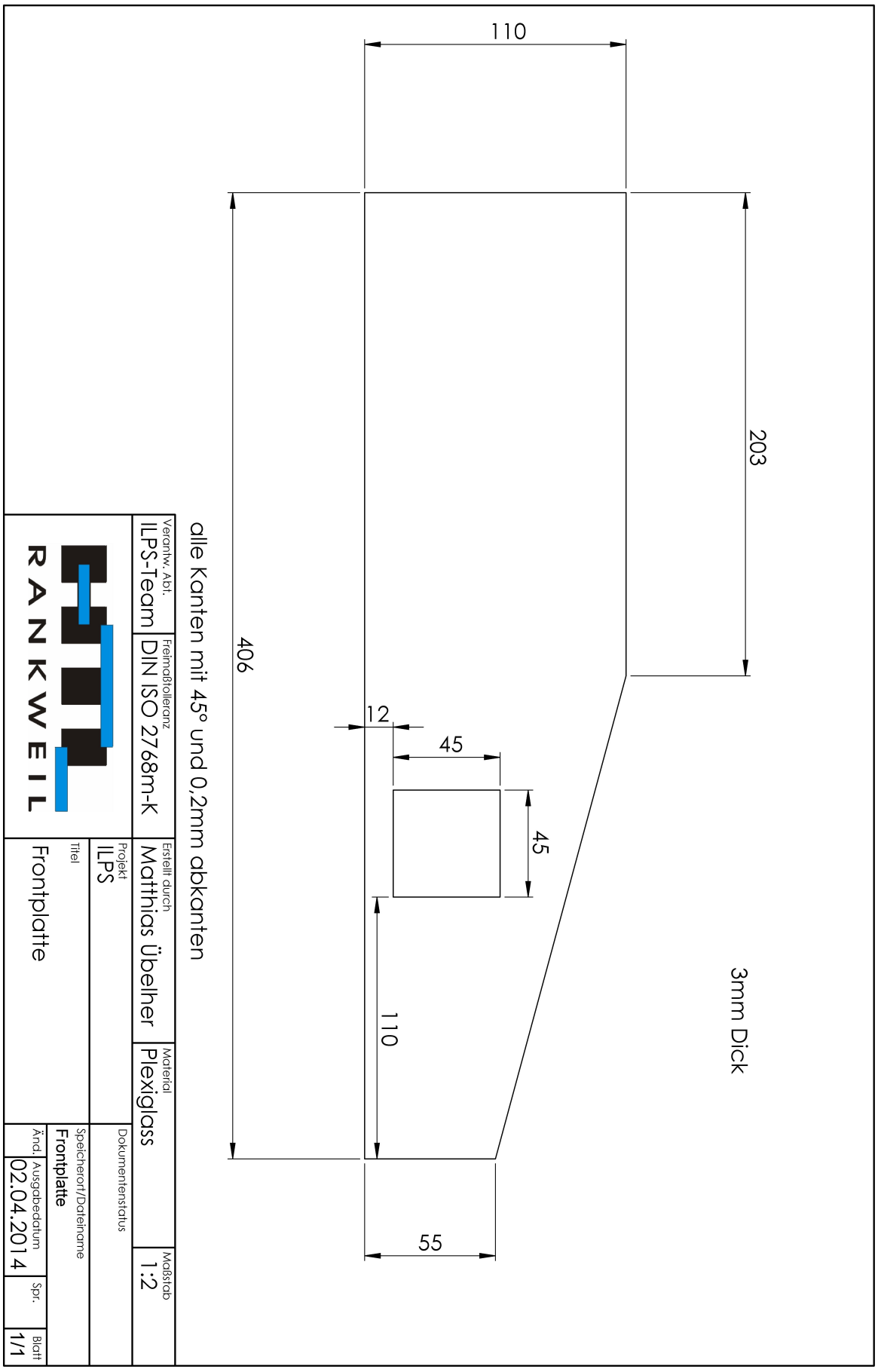
Verantw. Abt.	Freimaßtolleranz	Erstellt durch	Material	Maßstab
ILPS-Team	DIN ISO 2768m-K	Matthias Übelher	Aluminium	1:5

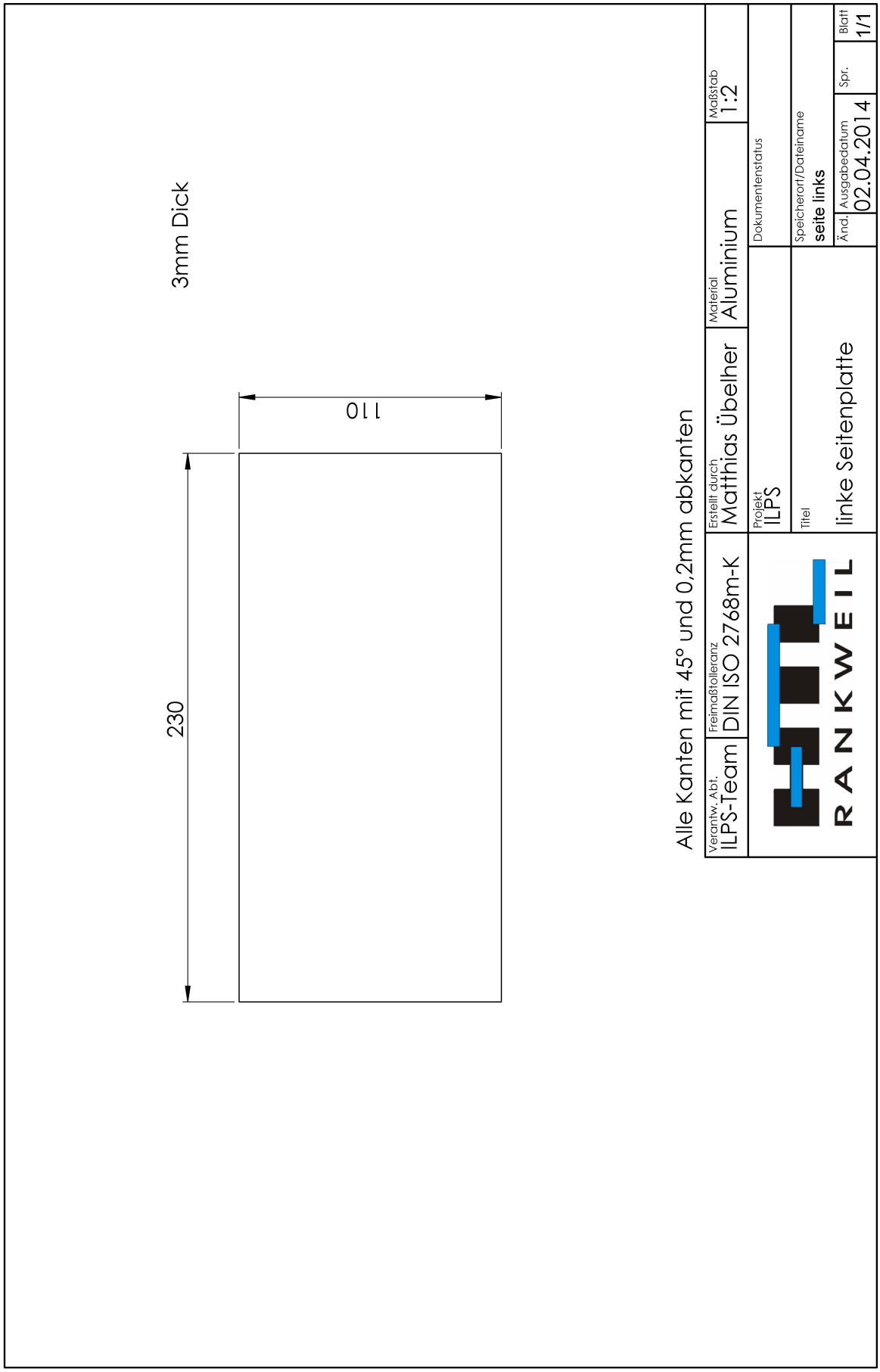
RANKWEIL

Projekt	Dokumentenstatus
ILPS	
Titel	Speicherort/Dateiname
Deckelplatte links	deckplatte

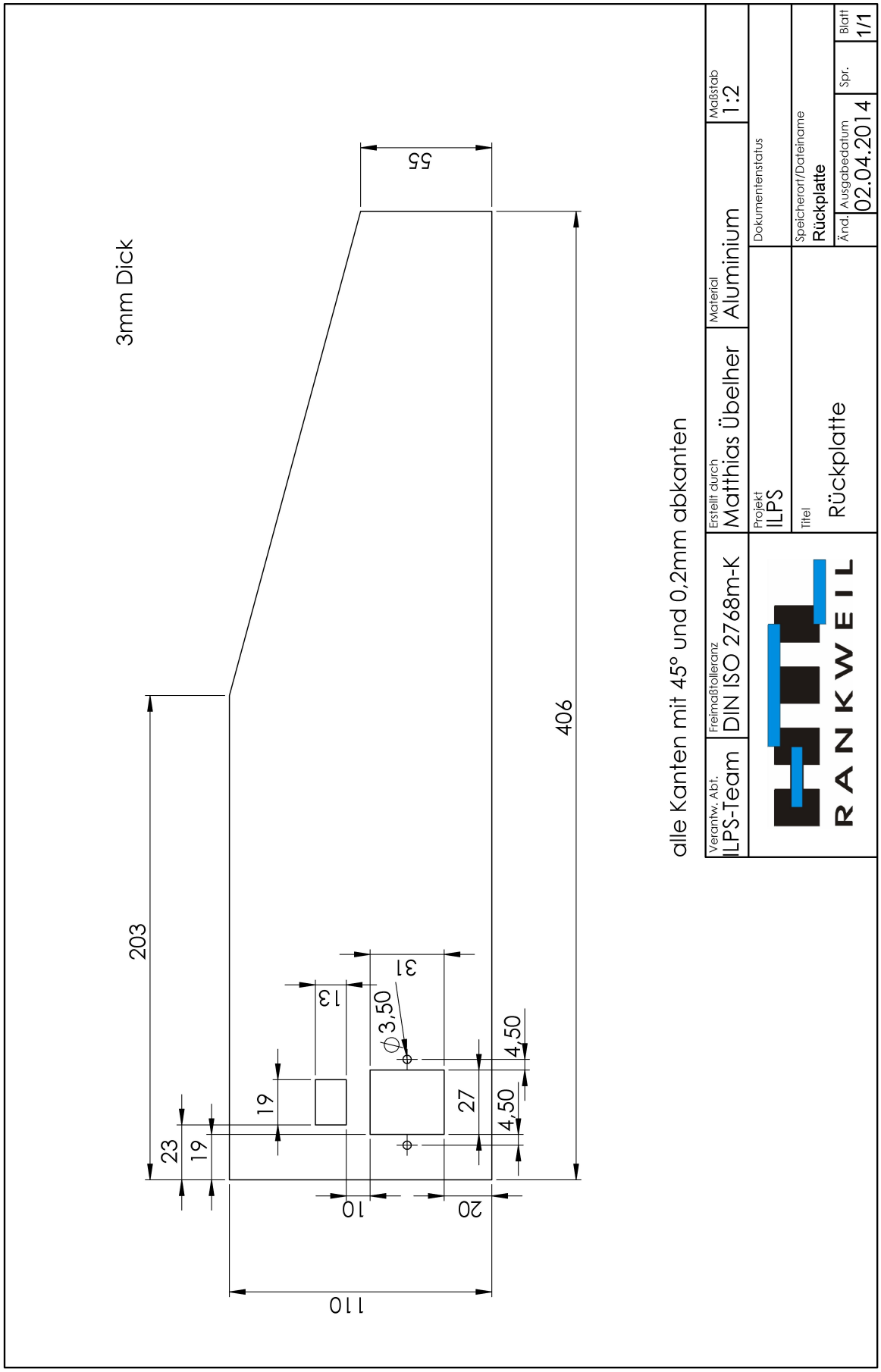
Änd.	Ausgabedatum	Spr.	Blatt
	02.04.2014		1/1

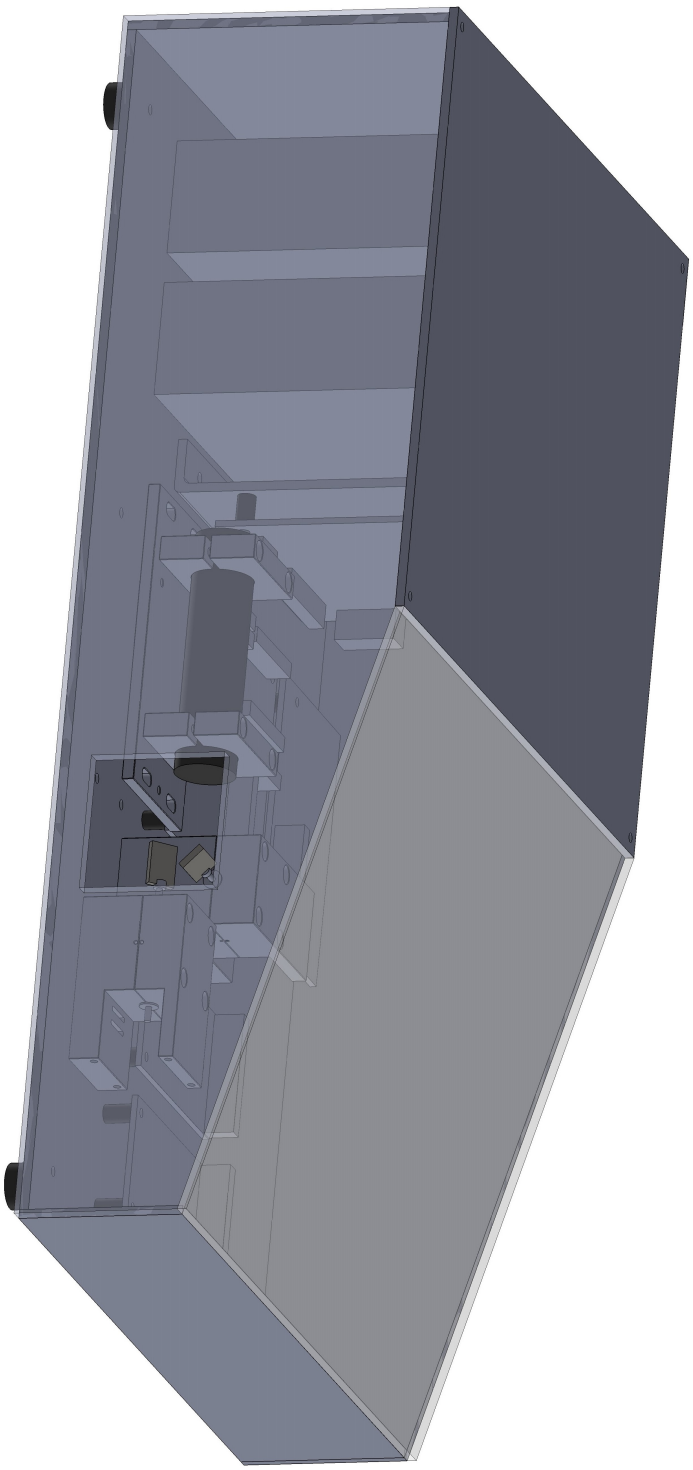

DETAIL B
MAßSTAB 3 : 1
75°
3mm Dick
B
A
238
211
alle Kanten mit 45° und 0,2mm abkanten
DETAIL A
MAßSTAB 3 : 1
75°
Veranlw. Abt.
ILPS-Team
Freimaßtoleranz
DIN ISO 2768m-K
Erstellt durch
Matthias Übelher
Material
Plexiglass
Maßstab
1:3
Projekt
ILPS
Dokumentenstatus
Titel
Deckel rechts
Speicherort/Dateiname
deckelglass
Änd.
Ausgabedatum
02.04.2014
Spr.
Blatt
1/1
RANKWEIL


110
203
406
12
45
45
110
55
3mm Dick
alle Kanten mit 45° und 0,2mm abkanten
Verantw. Abt.
ILPS-Team
Freimaßtoleranz
DIN ISO 2768m-K
Erstellt durch
Matthias Übelher
Material
Plexiglass
Maßstab
1:2
RANKWEIL
Projekt
ILPS
Dokumentenstatus
Titel
Frontplatte
Speicherort/Dateiname
Frontplatte
Änd.
Ausgabedatum
02.04.2014
Spr.
Blatt
1/1


3mm Dick
110
230
Alle Kanten mit 45° und 0,2mm abkanten
Verantw. Abt.
ILPS-Team
Freimaßtoleranz
DIN ISO 2768m-K
Erstellt durch
Matthias Übelher
Material
Aluminium
Maßstab
1:2
Projekt
ILPS
Dokumentenstatus
Titel
linke Seitenplatte
Speicherort/Dateiname
seite links
Änd.
Ausgabedatum
02.04.2014
Spr.
Blatt
1/1
RANKWEIL

230
55
3mm Dick
alle Kanten mit 45° und 0,2mm abkanten
Verantw. Abt.
ILPS-Team
Freimaßtoleranz
DIN ISO 2768m-K
Erstellt durch
Matthias Übelher
Material
Aluminium
Maßstab
1:2
RANKWEIL
Projekt
ILPS
Dokumentenstatus
Titel
rechte Seitenplatte
Speicherort/Dateiname
rechte platte
Änd.
Ausgabedatum
02.04.2014
Spr.
Blatt
1/1


3mm Dick
55
406
203
110
10
20
23
19
19
13
31
Ø3,50
4,50
27
4,50
alle Kanten mit 45° und 0,2mm abkanten
Verantw. Abt.
ILPS-Team
Freimaßtoleranz
DIN ISO 2768m-K
Erstellt durch
Matthias Übelher
Material
Aluminium
Maßstab
1:2
Projekt
ILPS
Dokumentenstatus
Titel
Rückplatte
Speicherort/Dateiname
Rückplatte
Änd.
Ausgabedatum
02.04.2014
Spr.
Blatt
1/1
RANKWEIL



Verantw. Abt.	Freimaßtoleranz	Erstellt durch	Material	Maßstab
ILPS-Team	DIN ISO 2768m-K	Matthias Übelher	Aluminium	1:2
		Projekt	Dokumentenstatus	
		ILPS		
		Titel	Speicherort/Dateiname	
RANKWEIL		Gesamtgehäuse ausgestattet	Grundplatte Zusammengestellt	
		Änd. Ausgabedatum	Spr.	Blatt
		15.04.2014		1/1

17.8 Stückliste Gesamt

Stückliste - Gesamt

Matthias Übelher ILPS Diplomarbeit 2013/2014

Name	Bezeichnung	Quelle	BestNr	Stück	Stückpreis	Gesamt
	Stückliste-Mainboard			1	14,35 €	14,35 €
	Stückliste-PowerModul			1	11,68 €	11,68 €
	Stückliste-Temperatursensor			3	0,95 €	2,85 €
	Stückliste-Galvanometer-Sensor			2	3,44 €	6,88 €
	Stückliste-Galvanometer-Regler			2	7,13 €	14,26 €
	M4x5 Linsenkopfschrauben	Magazin		4	0,01 €	0,04 €
	M4x6 Linsenkopfschrauben	Magazin		3	0,01 €	0,03 €
	M4x20 Linsenkopfschrauben	Magazin		4	0,01 €	0,04 €
	M3x5 Linsenkopfschrauben	Magazin		6	0,01 €	0,06 €
	M3x6 Linsenkopfschrauben	Magazin		3	0,01 €	0,03 €
	M3x16 Linsenkopfschrauben	Magazin		22	0,01 €	0,22 €
	M3x6 Kunststoffschrauben	Magazin		20	0,01 €	0,20 €
	M4 Mutter	Magazin		4	0,01 €	0,04 €
	26-Pol Flachbandkabel in Meter	Magazin		2	1,50 €	3,00 €
	6-Pol Flachbandkabel in Meter	Magazin		2	1,30 €	2,60 €
	230V Kaltgerätestecker Einbaustecker	Magazin		1	1,50 €	1,50 €
	230V 8A Fuse Träge	Magazin		1	0,10 €	0,10 €
	230V Kippschalter	Magazin		1	0,20 €	0,20 €
	Kabelschuh Flach 1,5	Magazin		5	0,01 €	0,05 €
	Kabelschuh 1,5 Rund 4Durchmesser	Magazin		13	0,01 €	0,13 €
	Raspberry Pi B 512MB	Rs	756-8308	1	27,00 €	27,00 €
	SD Card 10GB Class 10	Pollin	722-577	1	8,99 €	8,99 €
	USB-WLAN Adapter	Reichelt	EDIMAX EW-7811UN	1	7,00 €	7,00 €
	Laser Greendot Modulierbar	HighQLaser		1	0,00 €	0,00 €
	Schaltnetzteil, 72W, 6A, 12V	Reichelt	SNT RS 75 12	2	17,75 €	35,50 €
	Mainboard Platine	Schnetzer		1	4,00 €	4,00 €
	PowerModul Platine	Schnetzer		1	4,00 €	4,00 €
	Temperatursensor Platine	Schnetzer		3	1,00 €	3,00 €
	Galvanometersensor Platine	Schnetzer		2	2,00 €	4,00 €
	Galvanometerregler Platine	Schnetzer		2	3,00 €	6,00 €
	Galvanometerkondensator Platine	Schnetzer		1	2,00 €	2,00 €
	Dual Schottky	Reichelt	BAS 40-04 SMD	4	0,01 €	0,04 €
	0805 1n Kondensator	Magazin		2	0,01 €	0,02 €
	Galvanometer Oberteil	Salzgeber		2	0,01 €	0,02 €
	Galvanometer Unterteil groß	Salzgeber		1	0,01 €	0,01 €
	Galvanometer Unterteil klein	Salzgeber		1	0,01 €	0,01 €
	Galvanometer Spiegelhalterung	Salzgeber		2	0,01 €	0,02 €
	Galvanometer Magnethalterung	Salzgeber		4	0,01 €	0,04 €
	Laserhalterung oben	Salzgeber		2	0,01 €	0,02 €
	Laserhalterung unten	Salzgeber		2	0,01 €	0,02 €
	Laserhalterung Befestigungsplatte	Salzgeber		1	0,01 €	0,01 €
	Bodenplatte	Salzgeber		1	0,01 €	0,01 €
	Deckelplatte links	Salzgeber		1	0,01 €	0,01 €
	Deckelplatte rechts	Salzgeber		1	0,01 €	0,01 €
	Frontplatte	Salzgeber		1	0,01 €	0,01 €
	Seitenplatte links	Salzgeber		1	0,01 €	0,01 €
	Seitenplatte rechts	Salzgeber		1	0,01 €	0,01 €
	Rückplatte	Salzgeber		1	0,01 €	0,01 €
	Zweikomponentenkleber	Hornbach		1	9,99 €	9,99 €
	Dreiecksleiste Aluminium 15mmx15mm	Hornbach		4	2,15 €	8,60 €
	Litzenkabel schwarz 1,5mm^2 in m	Magazin		1	0,10 €	0,10 €
	Litzenkabel blau 1,5mm^2 in m	Magazin		1	0,10 €	0,10 €
	Litzenkabel grüngelb 1,5mm^2 in m	Magazin		1	0,10 €	0,10 €
	Gesamtkosten			**149**		**178,62 €**

18 Danksagung

Wir möchten und bei folgenden Personen und Unternehmen für ihre Unterstützung bedanken:

- *HighQLaser GmbH*, für die materielle Unterstützung
- *DI Franz Lauritsch*, Projektbetreuer seitens der HTL Rankweil
- *DI Christoph Büsel*, für die Unterstützung beim Auffinden von HF-Einwirkungen
- *DI Christoph Stüttler*, für die Unterstützung beim Aufbau von Regelstrecken
- *Hubert Salzgeber*, für die Unterstützung beim mechanischen Aufbau der Galvanometer
- *HTL Rankweil*, für die zur Verfügung gestellten Räumlichkeiten in der unterrichtsfreien Zeit.