



Fachbereich VII: Elektrotechnik - Mechatronik - Optometrie

Studiengang: Master Mechatronik

Labor: Fertigungsverfahren der Mechatronik

Bericht zur Projektaufgabe der Module AMS und MiMs

Thema:

„Konstruktion und Aufbau eines STL-3D-Druckers“

Bearbeitungszeitraum: 07.10.2019 bis 13.02.2020 (Abgabe)

Verantwortliche Lehrkraft: Prof. Dr.-Ing. Nicolas Lewkowicz
Prof. Dr.-Ing. Peter Gober
Nils Fußwinkel M. Eng.

Betreuer: Dipl.-Ing. B. Leuschner
D. Uckert B. Eng.
T. Sohr

Erstellt von: Gruppe C

Sperling, Florian	Mat-Nr. 897453
Stender, Dominik	Mat-Nr. 897981
Eggert, David	Mat-Nr. 806823
Dhiab, Sami	Mat-Nr. 871321
Eistel, Patrick	Mat-Nr. 889743

Aufgabenstellung

Projektziel ist es, ein Gerät zum additiven Fertigen auf der Basis von Stereolithografie als Labormuster aufzubauen und erfolgreich zu testen. Das Gerät soll vorgegebene Objekte mit möglichst hoher Genauigkeit herstellen können. Dieses Jahr liegt der Fokus auf einer besonders hohen Genauigkeit im Aufbau der Werkstücke. Der Drucker soll mit einer Druckgeschwindigkeit von mindestens 1 cm³ pro Stunde drucken können. Eine Aufstellung von Anforderungen ist in der Tabelle 1 dargestellt.

Tabelle 1: Anforderungsliste zum 3D-Drucker

Pos	Kd	Beschreibung					
		Funktion					
T1		Aufbau von Formen unter Nutzung des additiven Fertigungsverfahrens Stereolithographie					
T2		Aufbau mathem. beschriebener Geometrien oder Einlesen der Werkstückgeometrie aus dem CAD					
			Parametername	Wert		Einheit	Bemerkung
				min	typ	max	
		Bauraum					
1		Größte mögliche Werkstückmaße	x_max	2	2,5	3	cm
2			y_max	2	2,5	3	
3			z_max	2	3,5	5	
		Auflösung, Genauigkeit, Geschwindigkeit					
4		Auflösung in x,y,z	Delta_x,y,z	0,05	0,1	0,2	mm
5		Versatz zwischen Druckebenen	Delta_xx, Delta_yy	0,025	0,05	0,1	mm
6		Voxelgröße	v_Voxel	1	10	100	pl
7		Produktionsgeschwindigkeit	dV_polymer/dt	1	5	10	cm ³ /h
		Werkstoff der Werkstücke					
8		Photopolymer	Name_Werkstoff	Dehnstium			Eigene Herstellung
9		Wellenlänge des Photoinitiators	lambda_Werkstoff	405			passend zum Dehnstium
		Gehäuse und Sicherheit					
10		Betrieb einer Festspannungsversorgung	U_cc	9	12	15	V
11		Laserquelle über Türschalter gesichert		unabdingbar			
12		Gehäuse verhindert Austritt des Laserstrahls		unabdingbar			auch im teilmontierten Zustand
13		Galvanische Trennung zum Computer		unabdingbar			
		Bedienung					
14		Hauptschalter+PowerStatusLed					
15		Statusanzeigen über LED oder Display					
16		Bedienungstasten, Start, Pause, Reset					
17		Geometriefestlegung durch mathematische Formel im Programm		unabdingbar			Mindestanforderung Hohlzylinder mit Fensterausschnitt
		Geometriefestlegung durch Einlesen Geometrie CAD von PC			optional		Datenimport aus Creo
		Materialkostenbudget			200	€	inclusive der Kosten gestellter oder verbrauchter vorhandener Materialien

Um die Projektabwicklung in der Lehrveranstaltung zu vereinfachen und die Vergleichbarkeit zwischen den Projekten herzustellen, werden manche Komponenten und Funktionsprinzipien vorgegeben. Die Komponentenvorgaben beziehen sich vor allem auf den Mikrocontroller, Sensor- und Treiberbauelemente und die Laserdioden. Das Lasermodul, welches mit dem Treibermodul zusammen vorgegeben ist, muss am Rahmen der Anlage befestigt sein. Im Gegensatz zu den vorangegangenen

Semestern soll dieses Semester in der y- und z-Achse ein Schrittmotor vorgesehen und als x-Achse hingegen, die schneller betrieben werden soll als die anderen Achsen, ein Masse-Federschwinger eingesetzt werden, der elektromagnetisch- oder elektrodynamisch angeregt werden soll. Die Auslenkung des Schwingers soll nahe den Endlagen durch jeweils eine zweikanalige Durchlicht-Lichtschranke aufgenommen werden. Der Standardbauteilekatalog wird als Excelliste geführt und während des Projektes erweitert. Ziel ist es, Bauteile für alle gleich zu bestellen und damit den Bestellvorgang zu vereinfachen.

Inhalt

1 Entwurf	1
1.1 Regelungstechnik.....	2
1.1.1 Bedienungskonzept und Druckablauf	2
1.1.2 Grundlegender Regelkreis und seine Größen	3
1.1.3 Bedarfsanalyse Filter und Beobachter.....	5
1.2 Mechanik.....	7
1.2.1 Konzepte und Auswahl.....	7
1.2.2 Konzept y-Achse	8
1.2.3 Konzept z-Achse	9
1.2.4 Konzept x-Achse	9
1.2.5 Aufbau Konzept.....	11
1.2.6 Fertigungsgerechte Zeichnung der x-Achse, y- Achse und z-Achse	12
1.3 Sensorik	12
1.3.1 Bewertung und Auswahl der Messverfahren	12
1.3.2 Beschreibung der ausgewählten Messtechnik	14
1.3.3 Auslegung Messtechnik in y- und z-Achse	15
1.3.4 Genauigkeitsberechnung.....	16
1.3.5 Zeitliche und geometrische Auflösung.....	18
1.4 Aktorik.....	20
1.4.1 Anforderungen Aktoren der y- und z-Achse	20
1.4.2 Aktorauslegung.....	21
1.4.3 Auslegung magnetischer Kreis	21
1.4.4 Auswahl Aktorgröße	22
1.5 Elektronik	23
1.5.1 Genutzte Peripheriemodule am Mikrocontroller.....	23
1.5.2 Funktionsgruppen und ihre Bauteile	25
1.5.3 Beschreibung des detaillierten Schaltplans	26

1.5.4 Bestimmung Leiterbahnbreite.....	35
1.5.5 Beschreibung Layout.....	36
1.5.6 Vorversuche zu ausgewählten Funktionsgruppen.....	39
1.5.7 LT-Spice Simulationen zu kritischen Schaltungselementen	39
1.6 Software.....	40
1.6.1 Funktionale Beschreibung der Treiber.....	40
1.6.2 Funktionale Beschreibung der Interrupts und Timer.....	42
1.6.3 Funktionale Beschreibung des Main-Programms	44
1.6.4 Test- und Inbetriebnahmekonzept.....	44
2 Detailkonstruktion	46
2.1 Regelungstechnik.....	46
2.1.1 Vorüberlegungen zu den Regelparameter.....	46
2.1.2 Erste Inbetriebnahme der Baugruppen.....	48
2.2 Mechanik.....	48
2.2.1 Detailkonstruktion aller Teile unter Berücksichtigung der Fertigbarkeit	48
2.2.2 Beschreibung relevanter Toleranzketten	53
2.3 Sensorik	54
2.3.1 Testaufbauten zur Positionsmessung.....	54
2.3.2 Auswertung der Testmessungen	55
2.3.3 Fehleranalyse	55
2.4 Aktorik	56
2.4.1 Testaufbauten zu den Aktoren.....	56
2.4.2 Funktionstest PWM bzw. Schrittmusterausgabe	59
2.4.3 Auswertung der Tests.....	60
2.5 Elektronik	60
2.5.1 Beschreibung Layout.....	61
2.5.2 Bauteilliste im Anhang	62
2.5.3 Inbetriebnahmeplanung	62

2.6 Test Treibersoftware	65
2.6.1 Test von notwendigen Funktionen vom STM32 Nucleo Board	65
2.6.2 Verbindung Display zu Board über I2C	66
2.6.3 Ausgabe auf eine Konsole auf dem Bildschirm durch „Printf“:.....	68
2.6.4 ADC Single Channel Test mit Thermistor und NPN Transistor:.....	69
2.6.5 Funktionstest der Lichtschranke durch „Input Capture“ mit Timer	72
2.6.6 Implementierung Digitale Filter in Software:	77
2.6.7 Schrittmotor-Treiber Softwaretest.....	81
2.6.8 H-Bücke Software und Funktion Test.....	84
2.6.9 Sinusförmige Anregung - Funktionstest.....	86
2.6.10 Sinuserzeugung mittels Software:	89
3 Inbetriebnahme und Systemtest	96
3.1 Software	96
3.1.1 Deklaration der Variablen	96
3.1.2 Erstellung Algorithmen	98
3.1.3 Vorstellung verwendete Funktionen	115
3.1.4 Main-Programm und State-Machine.....	122
3.2 Regelungstechnik.....	123
3.2.1 Regelverhalten der Amplitudenregelung	124
3.2.3 Filterbedarf und Notwendigkeit von Beobachtern	126
3.3 Inbetriebnahme/ Test Hardware	126
3.3.1 Beschreibung des Aufbaus zum Ausrichten des Lasers.....	127
3.3.2 Abbilden einer Linie mit der Laserdiode	128
3.3.3 Y-Achse und z-Achse	129
3.4 Bedienungsanleitung.....	129
4 Testdruckergebnisse	130
4.1 Beschreibung und Bewertung der Druckergebnisse	131
4.2 Nicht umgesetzte Abhilfemaßnahmen	132

5 Zusammenfassung	133
5.1 Projekterfolge	133
5.2 Wesentliche technische Herausforderungen und ihre Lösungen	134
5.3 Bewertung als Semesterprojekt.....	135
5.4 Ausblick	137
6 Verzeichnis	138
6.1 Abbildungsverzeichnis	138
6.2 Tabellenverzeichnis	143
6.3 Formelverzeichnis	144
7 Quellen	145
8 Anlagen	147
Bauteilliste.....	I
Schaltpläne	II
Bedienungsanleitung.....	IX
Technische Zeichnungen	XII

1 Entwurf

In der folgenden Abbildung 1 wird ein grober Entwurf des SLA 3D-Druckers vorgestellt. Hierbei wird das organische Photopolymer auf 50 bis 60 °C erhitzt, um das Aushärten mithilfe eines Lasers zu beschleunigen. Die z-Achse des Druckers (Auf- und Ab-Bewegung) wird über einen Schrittmotor mit Gewindestange realisiert. Hierdurch ist es möglich, die oberste Schicht Photopolymer durch ein genaues Ansteuern des Druckbetts zu belichten. Dadurch härtet das Photopolymer an dieser Stelle aus. Nachdem eine Schicht dementsprechend belichtet wurde, kann das Druckbett um eine weitere Druckschicht nach unten verfahren werden. Dieser Vorgang wird wiederholt, bis das Druckobjekt fertig gedruckt wurde.

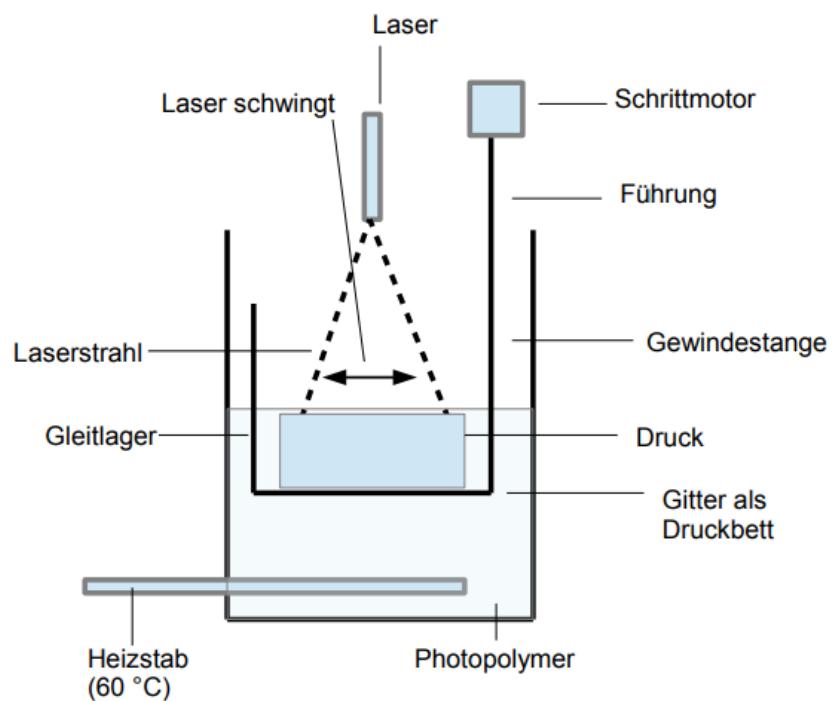


Abbildung 1: Entwurf 3D-Drucker

Anhand dieser Prozedur soll das Druckobjekt Schicht für Schicht belichtet werden, um das Druckobjekt anzufertigen. Diese Skizze stellt dabei einen sehr groben, ungefähren Aufbau dar und soll das Druckprinzip verdeutlichen.

1.1 Regelungstechnik

Es wird die Regelungstechnik des 3D-Druckers erläutert. Dabei werden die Regelungen und das Bedienkonzept bestimmt.

1.1.1 Bedienungskonzept und Druckablauf

Zunächst wird ausdrücklich vor einer Benutzung des 3D-Druckers ohne entsprechende Schutzmaßnahmen, wie eine Laserschutzbrille (für blaue Wellenlängen) gewarnt. Der Betrieb des Prototyps beim Druckvorgang ist sehr gefährlich und Schutzmaßnahmen müssen jederzeit eingehalten werden.

Der Benutzer steckt den 3D-Drucker in einem 12 Volt Netzgerät ein. Anschließend muss dieser kontrollieren, ob sich ausreichend Photopolymer im Behälter des 3D-Druckers für den nächsten Druckvorgang befindet. Des Weiteren wird ein Taster für eine manuelle Abbewegung des Druckbetts vorhanden sein. Hierdurch kann der Benutzer das Druckbett auf eine optimale Belichtungsposition bewegen. Diese optimale Position schwankt jeweils von der Höhe des Photopolymers im Becher.

Anschließend wird die Tür des Druckers geschlossen, um die enthaltene Sperrfunktion (kein Einschalten und Laser spannungsfrei) des Druckers zu deaktivieren. Diese Sperrung wird über Soft- und Hardware realisiert, indem der als Schließer ausgelegte Endschalter in der Tür die Spannungsversorgung des Lasers unterbricht. Nach Schließen der Tür kann der Benutzer anschließend mit einem Startknopf den Druckvorgang beginnen. Am angehängten Display kann dann die Frequenz des Schwingers, die restliche Druckdauer und die Anzahl der bereits gedruckten Schichten abgelesen werden.

Dabei wird zu Beginn des Drucks die Heizung für das organische Photopolymer aktiviert. Diese wird das Polymer auf 50 bis 60 °C vorheizen. Nach Erreichen der richtigen Temperatur startet anschließend der eigentliche Druckprozess und das Druckmodell wird gedruckt.

Nach Beendigung des Druckvorgangs wird der Drucker automatisch in die Endlagen der Achsen fahren, um die Entnahme des Druckobjekts zu vereinfachen.

1.1.2 Grundlegender Regelkreis und seine Größen

Die grundlegenden Regelkreise des 3D-Druckers werden vorgestellt.

Regelkreis der Heizung

Es wird der Regelkreis der Heiztemperatur des Photopolymers mit dessen Größen in der Abbildung 2 vorgestellt.

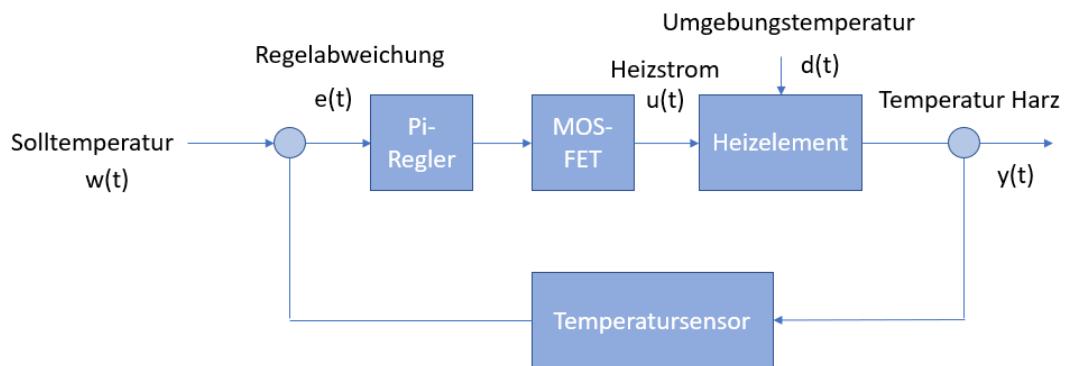


Abbildung 2: Regelkreis Heiztemperatur

Durch den Mikrocontroller wird eine definierte Solltemperatur des Photopolymers festgelegt. Diese Information des Soll-Wertes wird mit dem gemessenen Istwert des Temperatursensors subtrahiert und die Regelabweichung e an den PI-Regler gesendet. Durch die Regelabweichung bzw. Temperaturdifferenz erzeugt der PI-Regler ein High-Signal, um einen MOSFET im Heizstromkreis anzusteuern und somit einen entsprechenden Heizstrom durch das Heizelement zu erzielen. Störeinflüsse, wie z.B. die Umgebungstemperatur, werden durch die Regelung korrigiert. Der Regelkreis wird dabei als sehr träge eingeschätzt, mit wenig Schwankung und geringen Größenänderungen. Durch die Regelung wird die Temperatur des Photopolymers durch Hinzufügen von Wärmeenergie auf einem konstanten Niveau gehalten.

Regelkreise der schwingenden x-Achse:

Für die x-Achse als resonantes System wird eine Regelung für eine gleichmäßige Schwingung und Frequenz benötigt. Zur Realisierung wird eine Amplitudenregelung mit eventuell zusätzlicher PLL (Phasenregelschleife) ausgewählt: Diese regeln die zugeführte Energie in das System, einerseits über die Amplitude der anzuregenden Schwingung (voraussichtlich anregende Sinus-Schwingung) sowie über eine definierten Phasendifferenz zwischen elektrischer Spannung und mechanischer Schwingung.

Regelung der Frequenz:

Die Regelung der Taktfrequenz wird in Abbildung 3 dargestellt.

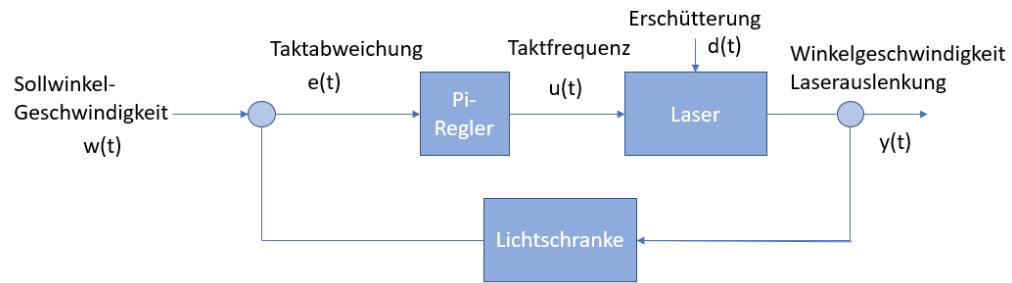


Abbildung 3: Regelkreis Taktfrequenz x-Achse

Der Mikrocontroller gibt eine entsprechende Soll-Winkelgeschwindigkeit für den Schwinger vor. Diese Information wird mit der gemessenen Ist-Winkelgeschwindigkeit durch die Lichtschranke an den Mikrocontroller gesendet und verglichen. Hieraus wird die Abweichung zwischen den Phasen (elektrisch und mechanisch) ermittelt und die Taktfrequenz entsprechend angepasst. Störeinflüsse am Laser, wie z.B. Erschütterungen oder Dämpfung, werden somit korrigiert. Der Laser erreicht somit durch die angepasste Taktfrequenz eine exakte Winkelgeschwindigkeit bzw. Auslenkung.

Regelung der Amplitude:

In der Abbildung 4 wird der Regelkreis zum Sollverhältnis der Dunkelzeit zur Periodendauer vorgestellt.

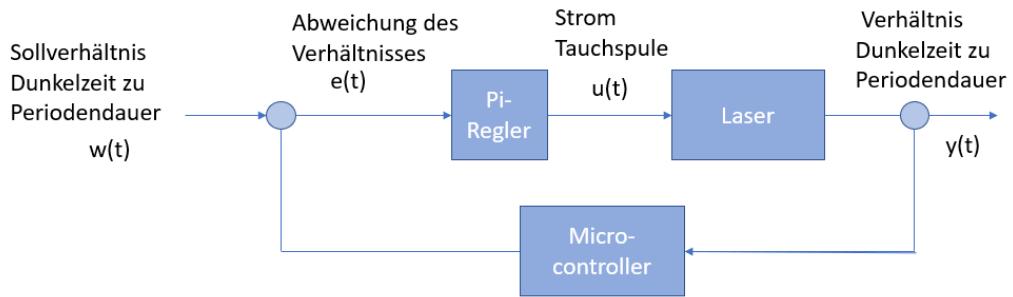


Abbildung 4: Regelkreis Verhältnis Dunkelzeit zu Periodendauer

Durch den Mikrocontroller wird ein entsprechendes Verhältnis von Dunkelzeit zu Hellzeit und damit eine Information zur Periodendauer vorgegeben. Diese Information wird mit dem gemessenen Ist-Verhältnis durch die Lichtschranke gemessen, verglichen und anschließend an den PI-Regler gesendet. Die Abweichung vom Soll- und Ist-Verhältnis wird durch den Strom der Tauchspule (Amplitude der Schwingung) entsprechend geregelt, sodass der Laser im definierten Sollverhältnis schwingt.

1.1.3 Bedarfsanalyse Filter und Beobachter

Es werden Filter und Beobachter untersucht und bestimmt.

Bedarfsanalyse für die Regelung der y- und z-Achse:

Die Verstellung der y- und der z-Achse erfolgt über einen Schrittmotorantrieb, welcher mit einer definierten Steigung und einem bestimmten Winkel betrieben wird. Die Höhenänderung ist somit errechenbar und eine Regelung wird nicht benötigt. Somit werden in diesen beiden Achsen auch keine Filter verwendet.

Bedarfsanalyse für die Regelung der x-Achse:

Da die Baugruppe der x-Achse auf einem resonanten System beruht, sind korrekte Einstellungen und Regelungen der Sinusfrequenz und der Amplitude für die Lasersteuerung notwendig. Da angrenzende Bauteile die Spannungssignale verfälschen können, muss eventuell eine Filterung vorgesehen werden. Dies wird sich in späteren Tests (Kontrolle Ausgangssignal Lichtschranke) herausstellen. Falls ein Filter benötigt wird, eignet sich ein FIR-Filter (Finite Impulse Response Filter), welcher keine Rückkopplungen besitzt und einen Ausgangswert des Signals nur unter Verwendung von endlich vielen Eingangswerten berechnet und daher immer stabil bleibt.

Bedarfsanalyse für die Regelung der Heizung:

Ein Temperatursensor nimmt die Messung der Temperatur des Harzes vor. Temperaturschwankungen geschehen jedoch in diesem Fall sehr langsam. Eine kritische Auswirkung durch die Störquellen auf der Platine ist durch die träge Änderung der Temperatur unwahrscheinlich und ein Filterbedarf ist nicht relevant.

1.2 Mechanik

Es wird der geplante mechanische Aufbau des Druckers vorgestellt.

1.2.1 Konzepte und Auswahl

Die Konstruktion des 3D-Druckers erfolgt mit dem Programm Creo Parametric 5.0. Der 3D-Drucker besteht dabei aus mehreren Baugruppen, die in der Tabelle 2 aufgelistet und beschrieben sind.

Tabelle 2: Auflistung und Beschreibung der mechanischen Baugruppen

Baugruppe	Werkstoff	Beschreibung
Gestell	Aluminiumprofil (20 x 20 mm)	An das Gestell werden die verschiedenen Baugruppen montiert. Dabei soll es eine hohe Steifigkeit besitzen.
x-Achse (Feder-Masse-Schwinger)	Aluminium, (alternativ: Polyamid, Photopolymer)	Diese Baugruppe hält, bzw. schwingt mit dem Laser in vorgegebener Frequenz. Das Material Aluminium für den Schwinger kann aktuell noch nicht festgelegt werden, da die Benutzung einer Fräsmaschine nicht gesichert ist. Alternativ wird der Schwinger in Photopolymer gedruckt
y-Achse	Polyamid, Aluminium, Gewindestange (Messing)	Hält Schrittmotor der y-Achse, überträgt die Rotationsbewegung des Motors in eine Translationsbewegung der y-Achse
z-Achse	Polyamid, Aluminium	Hält Schrittmotor der z-Achse, überträgt die Rotationsbewegung des Motors in eine Translationsbewegung der z-Achse

1.2.2 Konzept y-Achse

In der nachfolgenden Abbildung 5 ist der Aufbau der y-Achse dargestellt. Eine Gewindestange wird mit dem Schrittmotor mithilfe einer Kupplung verbunden. Die Drehung der Gewindestange wird durch eine Gewindeglocke in dem Verbindungsstück realisiert. Durch das Verbinden der Gewindeglocke mit dem Führungssystem wird die Bewegung der y-Achse sichergestellt. Außerdem kann durch einen Endschalter die Endlage bestimmt werden.

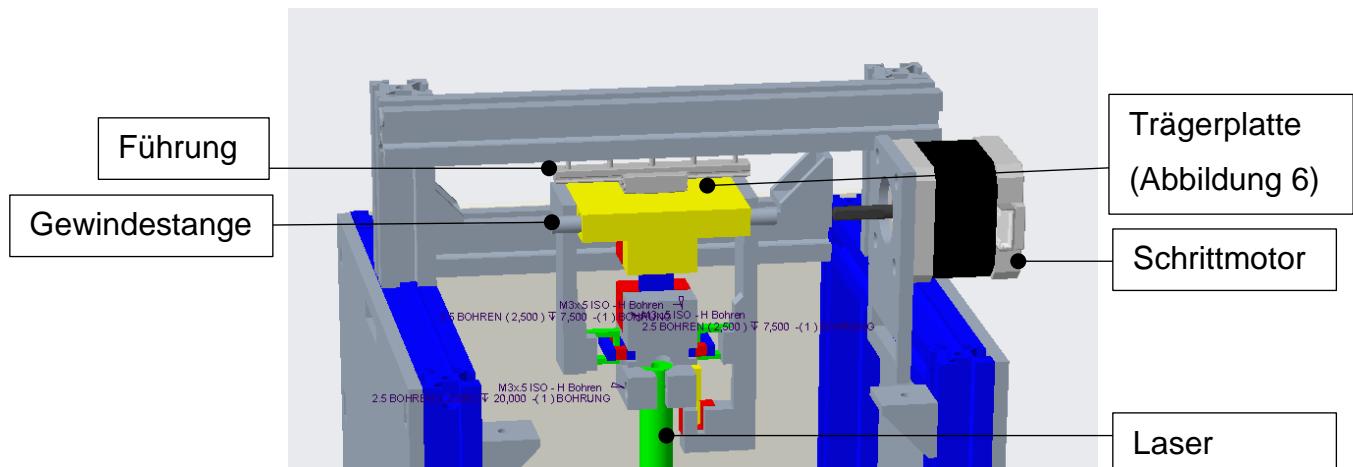


Abbildung 5: Konzept y-Achse

Die Führung der y-Achse wird dabei über eine Schiene mit Führung erreicht. Diese wird im weiteren Verlauf vorgestellt. An dieser Führung wird die Trägerplatte (Abbildung 6) über eine Gewindestange angetrieben und positioniert. Die Trägerplatte selbst hält anschließend den Swinger mit der Lasereinheit.

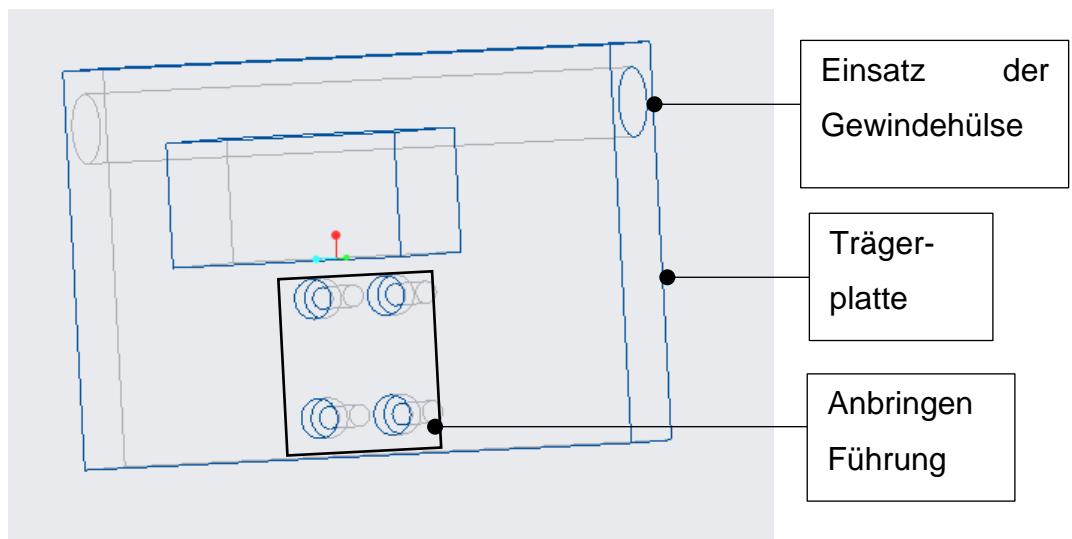


Abbildung 6: Trägerplatte y-Achse

1.2.3 Konzept z-Achse

In der nachfolgenden Abbildung 7 ist das Konzept der z-Achse zu sehen. Das Prinzip ist grundsätzlich dasselbe wie in der y-Achse, nur dass statt einer Gewindehülse eine Sechskantmutter im Verbindungsstück enthalten ist.

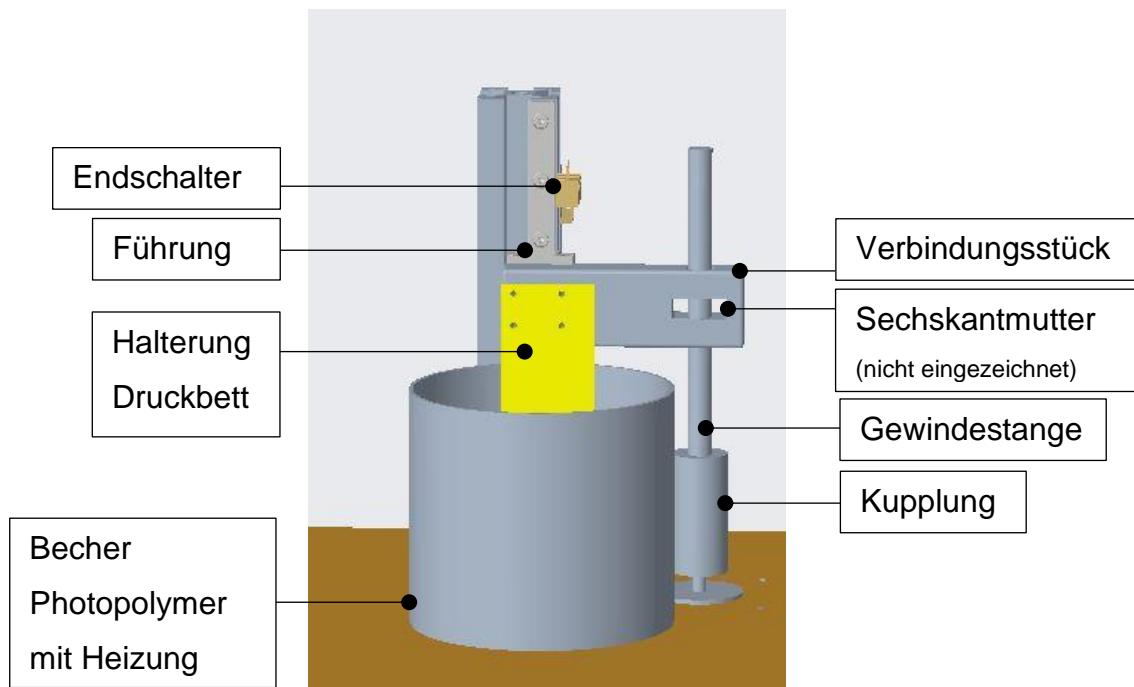


Abbildung 7: Darstellung z-Achse

1.2.4 Konzept x-Achse

In der Abbildung 8 wird der konzeptionelle Aufbau der x- bzw. schwingenden Achse dargestellt. Die Nummern werden anschließend im weiteren Text erläutert.

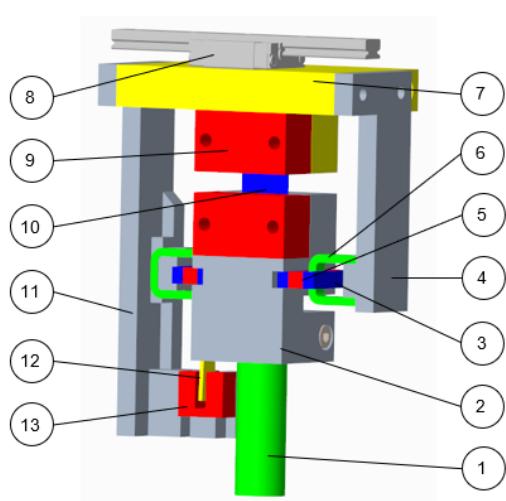


Tabelle 3: Beschreibung Swinger x-Achse

1	Laserhülse
2	Laserhalterung
3	(Quader-)Magnet
4	Halterung Spule
5	Abstandshalter
6	Spule
7	Trägerplatte
8	Linearführung
9	Klemmelement
10	Blattfeder
11	Halterung Spule und Lichtschranke
12	Blende
13	Lichtschranke

Abbildung 8: Aufbau der x-Achse (Swinger)

Eine Trägerplatte (7) ist das Bindeglied zwischen dem oszillierenden Swinger, den Halterungen (4,11) für Spulen und Lichtschranke, der Linearführung (8) und des Antriebs der y-Achse. An ihr wird durch ein Klemmelement (9) eine Blattfeder (10) befestigt, die gleichzeitig als Lager und Federkomponente der x-Achse dient. An der Feder wiederum ist durch einen weiteres Klemmelement die Laserhalterung (2) befestigt. Diese klemmt die Laserhülse (1) und nimmt die Magnetenspakete auf. Die Magnetenspakete bestehen aus je zwei Quadermagneten (3) und einem Abstandshalter (5) aus Kunststoff. Die Magnete sind so zueinander ausgerichtet das sich in einem Luftspalt dazwischen ein nahezu homogenes Magnetfeld ausbildet, in dem sich senkrecht dazu eine Spule (6) befindet, die bei Bestromung eine Kraft auf den Swinger ausübt und diesen antreibt. Die Spulen sind dabei fest mit den Halterungen verbunden. Einer der Halter (11) trägt zusätzlich die Lichtschranke (13), die über eine an der Laserhalterung befestigte Blende (12) zur Schwingungsmessung dient.

Die einzelnen Komponenten sind nach einer möglichst integralen Bauweise konzipiert, um die Bauteilanzahl und damit den Bauaufwand gering zu halten. Trägerplatte, Laserhalterung und Klemmelemente werden falls möglich aus Aluminium gefräst, um eine hohe Präzision zu gewährleisten. Falls die Fräsmaschine nicht verwendet werden kann, werden diese Teile mithilfe eines FDM oder SLA-Druckers gedruckt. Die Halter für Spulen und Lichtschranke können ebenfalls kostengünstige FDM-Druckteile sein.

Die schwingenden Elemente wurden möglichst symmetrisch aufgebaut um eine reine Sinusschwingung, ohne Überlagerung und Verzerrung, zu gewährleisten. Die Anregung der Schwingung soll konstant, sinusförmig und exakt synchron bei Resonanz des Swingers, um etwa 8 bis 9 Hertz liegen. Diese Schwingung von 8 Hertz werden mindestens benötigt, um die vorgegebene maximale Druckzeit zu erreichen.

1.2.5 Aufbau Konzept

Es werden zwei verschiedene Konzepte ausgearbeitet, da noch unklar ist, wie praktikabel die Verbindung der y- und z-Achse mithilfe je einer Gewindestangen abläuft. Aus diesem Grund wurde ein Konzept 1 mit Gewindestangen (für y- und z-Achse) sowie ein anderes Konzept (Konzept 2) mit einer Gewindestange und einem Riemen (y-Achse mit Riemen, z-Achse mit Gewindestange) konzipiert. Das Konzept 1 wird in der folgenden Abbildung 9 dargestellt.

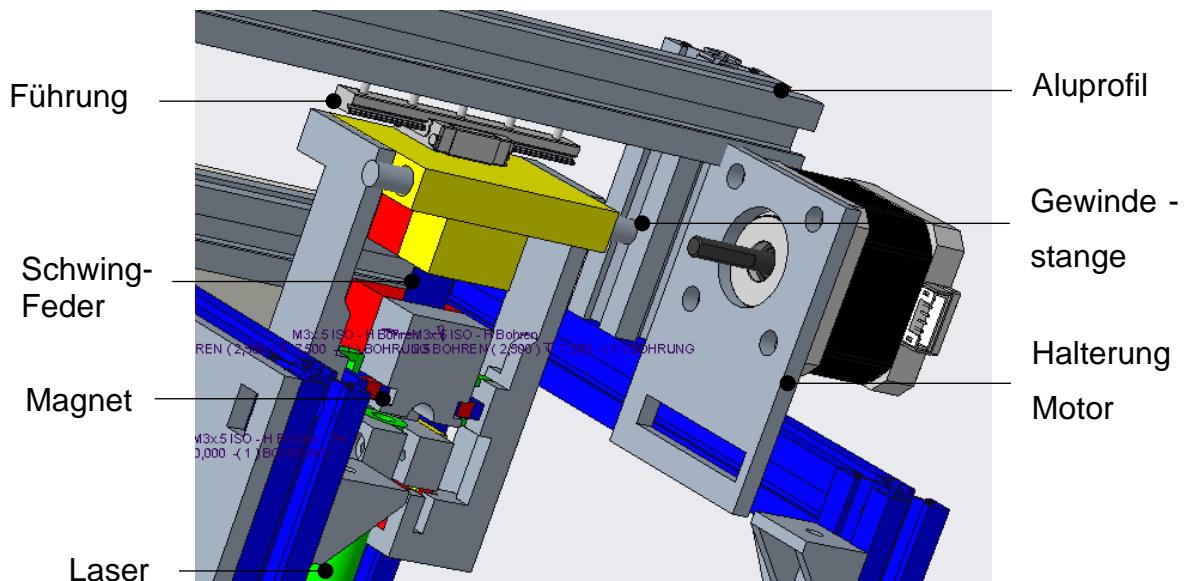


Abbildung 9: Konzept 1 mit zwei Gewindestangen

In der folgenden Abbildung 10 wird das Konzept 2 mit einem Riemenantrieb an der y-Achse und einer Gewindestange an der z-Achse dargestellt.

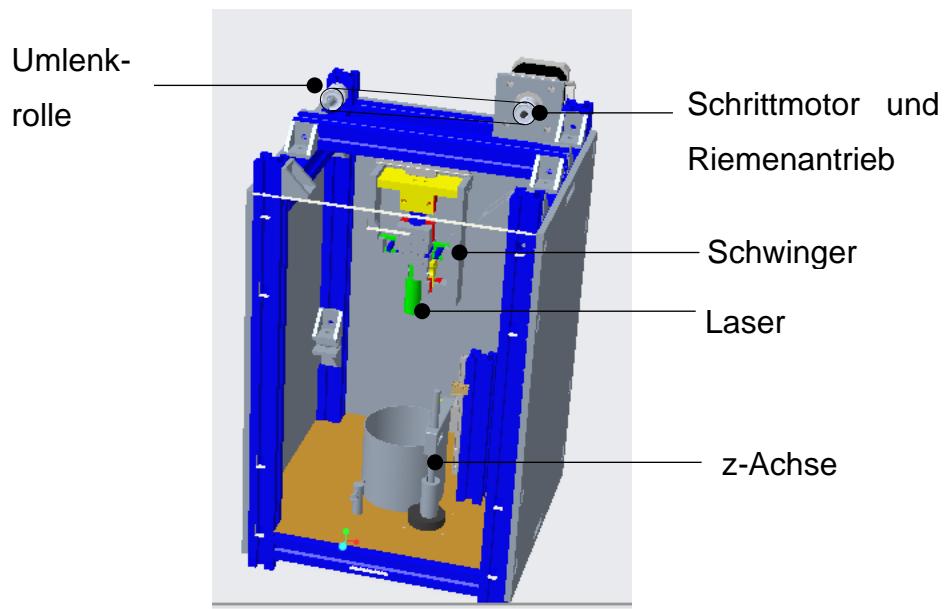


Abbildung 10: Konzept 2: 3D-Drucker mit Riemenantrieb und Gewindestange

1.2.6 Fertigungsgerechte Zeichnung der x-Achse, y- Achse und z-Achse

Die fertigungsgerechten Zeichnungen der x-, y- und z-Achse sind in den Anlagen eingefügt.

1.3 Sensorik

Es wird die verwendete Sensorik und das Messverfahren vorgestellt.

1.3.1 Bewertung und Auswahl der Messverfahren

Während des Druckvorgangs werden verschiedene Messungen durchgeführt. Dazu zählen die Messung der Position, Frequenz und Verhältnis (Hell- und Dunkel) des Feder-Masse-Schwingers. Außerdem werden Endlagensensoren in der y- und z-Achse sowie die Messung der Temperatur des Photopolymers durchgeführt. Es gelten besondere Anforderungen an die Lichtschranke, die das Signal so zeitnah wie möglich übertragen muss.

Die eigentliche Bestimmung der Positionen der y- und z-Achse wird nur beim Start des 3D-Druckers anhand von Endschaltern durchgeführt. Nach Anfahren der Endschalter und einmaliger Betätigung kennt der Mikrocontroller zu jeder Zeit die Position der y- und z-Achse (von Fehlern wie Schrittverlusten abgesehen).

In der folgenden Abbildung 11 wird grob das Messprinzip und der Aufbau des Messvorhabens dargestellt. Hierzu wird eine Lichtschranke verwendet (siehe auch Abbildung 15), die einen Durchgang des Masse-Feder-Schwingers registriert.

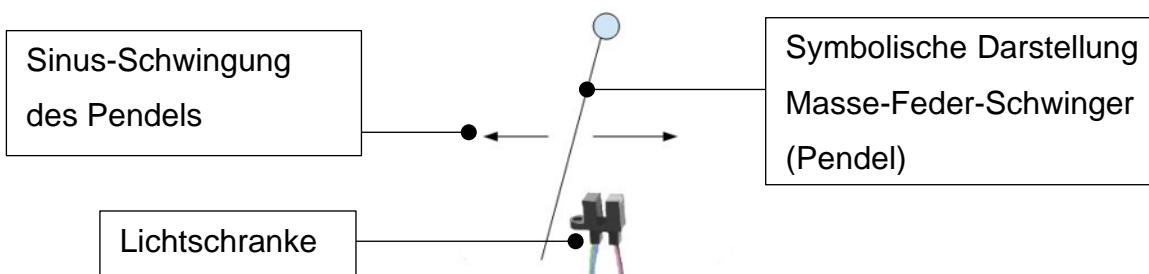


Abbildung 11: Aufbau und Position der Lichtschranke (Lichtschranke aus [1])

In der Abbildung 12 wird anschließend das angestrebte Messprinzip und Messverfahren des Feder-Masse-Schwingers mit den erwünschten Ergebnissen und Signaleingängen vorgestellt. Dabei schwingt der Masse-Feder-Schwinger (Pendel) in

einer Sinus-Schwingung. Pro Periode des Pendels soll es hierbei vier Messpunkte geben. In der Abbildung 12 ist die zu erwartende Sinus-Schwingung des Schwingers dargestellt.

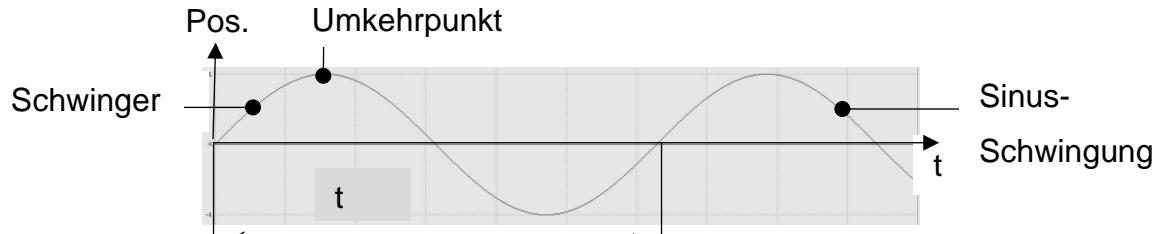


Abbildung 12: Darstellung Bewegungsablauf Schwinger

Anschließend werden die Messpunkte der Lichtschranke in der Schwingung dargestellt (siehe Abbildung 13).

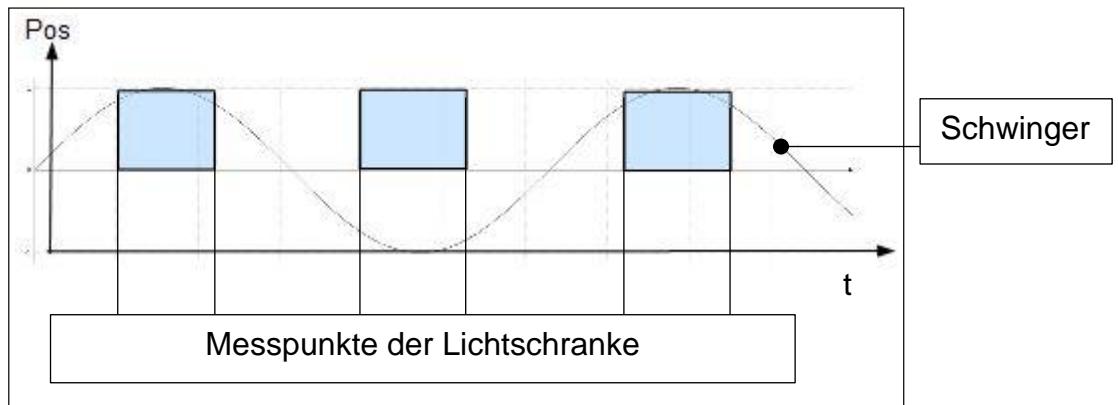


Abbildung 13: Messpunkte der Lichtschranken

Nach Aufnahme der Messwerte werden die Ergebnisse an den Mikrocontroller übertragen. Dieser wird das Signal anschließend weiterbearbeiten und die Aktorik dementsprechend steuern bzw. regeln.

Für die Signale der Endschalter wird ein einfaches High-Signal (wenn als Schließer ausgelegt) erwartet. Das Signal wird in der Abbildung 14 dargestellt.

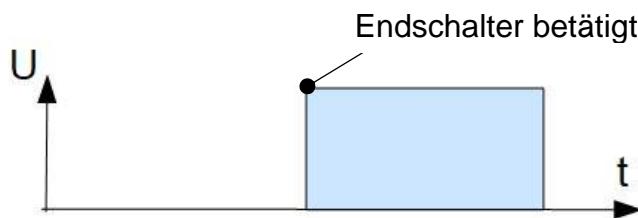


Abbildung 14: Signal, wenn Endschalter als Schließer ausgelegt

1.3.2 Beschreibung der ausgewählten Messtechnik

Für die Messung des Schwingverhaltens der x-Achse wird die Lichtschranke OPB 460 (siehe Abbildung 15) der Firma TT Electronics verwendet. Diese besitzt 5 Anschlüsse, welche im angehängten Datenblatt [1] beschrieben werden.



Abbildung 15: Verwendete Lichtschranke [1]

Die wichtigsten Betriebsbedingungen der Lichtschranke werden in der Abbildung 16 gezeigt. Dazu zählt der Spannungsabfall an der LED sowie der durchschnittliche Stromfluss durch die LED.

Electrical Characteristics ($T_A = 25^\circ C$ unless otherwise noted)						
SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
Input Diode						
V_F	Forward Voltage	-	-	1.7	V	$I_F = 20 \text{ mA}, T_A = 25^\circ C$
I_R	Reverse Current	-	-	100	μA	$V_R = 2 \text{ V}, T_A = 25^\circ C$

Abbildung 16: Gewöhnliche Betriebsbedingungen der Lichtschranke [1]

Zur Bestätigung des Starts eines Druckvorgangs wird ein Miniatur-Taster verwendet. Dieser ist in der Abbildung 17 gezeigt. Der Taster kann bis zu 50 mA bei 24 Volt DC führen.



Abbildung 17: Verwendeter Taster [2]

Des Weiteren wird ein Endschalter beim Start des Druckers für den Betrieb von y- und z-Achse einmalig benötigt. Beide Achsen fahren nacheinander in die zuvor definierte Endlage und werden dadurch auf den Endschalter fahren. Sobald dieser betätigt wird, kennt der Mikrocontroller die aktuelle Position des Schrittmotors und stoppt ein weiteres Verfahren in diese Richtung. Der Endschalter wird in Abbildung 18 vorgestellt.

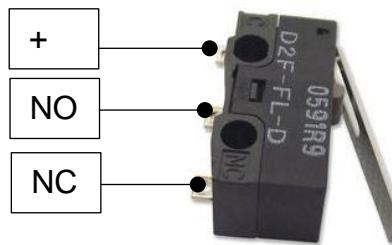


Abbildung 18: Endschalter der y- und z-Achse [3]

„NC“ steht hierbei für die Beschaltungsart „normal geschlossen“ und bedeutet, dass der Endschalter bei Betätigung unterbricht, während er bei „NO“ „normal geöffnet“ den Stromkreis schließt.

Außerdem wird ein Temperatursensor für die Messung der Temperatur des Polymers verwendet. Dazu ist ein NTC (Kaltleiter) mit $100\text{ k}\Omega$ geplant. Da zur aktuellen Zeit noch keine Bauteile bestellt sind, kann noch keine Aussage zum genauen Typ getroffen werden.

1.3.3 Auslegung Messtechnik in y- und z-Achse

In der y- und z-Achse werden die zuvor gezeigten Endschalter zum Start des Druckers benötigt, um die Position der Schrittmotoren einmalig festzustellen. Eine weitere Messung ist in diesen beiden Achsen nicht notwendig, da der Mikrocontroller nach dem einmaligen Anfahren der Endschalter zu jeder Zeit die Positionen der y- und z-Achse kennt (von Fehlern wie Schrittfehler abgesehen). Da Schrittmotoren keine Positionssensoren benötigen, sondern lediglich der Mikrocontroller die verfahrenen

Schritte addiert bzw. subtrahiert (je nach Richtung) wird kein zusätzlicher Sensor benötigt.

1.3.4 Genauigkeitsberechnung

Im Folgenden wird die Ungenauigkeit, bzw. die Toleranzen der einzelnen Bauelemente benannt und berechnet. Dazu werden zunächst die beiden Führungen SZB8-100 (mit 100 mm Länge) und SZB8-70 (mit 70 mm Länge) betrachtet. Die Führung wird in der Abbildung 19 dargestellt.



Abbildung 19: Verwendete Führung Misumi SEBZ8 -70 [4]

Ein Screenshot des Datenblatts zur Toleranz der Parallelität der beiden Führungen ist in der Abbildung 20 zu sehen.

Rail Length (mm)		Miniature			Linear Guides for Medium/Heavy Load		
Over	or Less	Precision Grade	High Grade	Standard Grade	High Grade	Interchangeable	Standard Grade
	50	2	3	13	7	6	7
50	80	2	3	13	7	6	7
80	125	3	7	15	7	6.5	7

Abbildung 20: Präzision der Misumi-Führungen SZB8-100 und SZB8-70 [4]

Aus den gegebenen Werten von 13 µm Toleranz bei der 70 mm langen Schiene sowie der 15 µm Toleranz der 100 mm langen Schiene wird im weiteren die maximale Gesamtabweichung eines Druckobjekts anhand der Toleranzen der Führungen berechnet (siehe Abbildung 21). Dabei wird die Führung SEBZ8-70 für die z-Achse und die Führung SZB8-100 für die y-Achse verwendet.

	Führung SEBZ8-70	Führung SEBZ8-100	
Druckbereich d	$d_{70} := 30 \text{ mm}$	Druckbereich d	$d_{100} := 30 \text{ mm}$
Länge Schiene	$l_{70} := 70 \text{ mm}$	Länge Schiene	$l_{100} := 70 \text{ mm}$
Abweichung	$F_{70} := 13 \mu\text{m}$	Abweichung	$F_{100} := 15 \mu\text{m}$
Toleranz	$\frac{d_{70} \cdot F_{70}}{l_{70}} = 5.571 \mu\text{m}$	$\frac{d_{100} \cdot F_{100}}{l_{100}} = 6.429 \mu\text{m}$	

Abbildung 21: Berechnung der möglichen Abweichung

Außerdem werden die kleinsten Winkel zum Verfahren der Schrittmotoren berechnet. Dabei werden die Eingänge MS1 und MS2 der Schrittmotorentreiber mit der Masse verbunden. Die Schrittmotoren besitzen eine maximale Schrittzahl von 200 Schritten pro Umdrehung. Das bedeutet, dass 200 Schritte einer ganzen Umdrehung der Motorwelle entsprechen. Teilt man diese 200 Schritte durch 360° erhält man den kleinsten verfahrbaren Schrittewinkel. Die Rechnung wird in der Abbildung 22 durchgeführt.

Umdrehung U	$U := 360^\circ$
Schritte	$S := 200$
kleinster Schrittewinkel	$\frac{U}{S} = 1.8^\circ$

Abbildung 22: kleinster möglicher Schrittewinkel der Schrittmotoren

Für die Messung und Aufnahme der Messdaten von Analogspannungen (Potentiometer und Thermometer analog) muss ein ADC (Analog-Digital-Converter) geschalten werden. Dieser ändert den Analogwert der Spannung in einen Digitalwert. Ein wichtiges Kriterium ist hierbei die Auflösung des ADC. In der Abbildung 23 werden die einzelnen Auflösungen berechnet. In dieser Rechnung werden die Genauigkeit eines 12-, 16- und 32-Bit ADCs berechnet, um zu prüfen, ob die internen ADCs des Mikrocontrollers ausreichend sind.

ADC	Anzahl Werte	Spannung	Auflösung
12 Bit	$Bit_{12} := 4096$	$U := 3.3 \text{ V}$	$\frac{U}{Bit_{12}} = (8.057 \cdot 10^{-4}) \text{ V}$ pro Bit
16 Bit	$Bit_{16} := 65535$	$U := 3.3 \text{ V}$	$\frac{U}{Bit_{16}} = (5.035 \cdot 10^{-5}) \text{ V}$ pro Bit
32 Bit	$Bit_{32} := 4294967295$	$U := 3.3 \text{ V}$	$\frac{U}{Bit_{32}} = (7.683 \cdot 10^{-10}) \text{ V}$ pro Bit

Abbildung 23: Genauigkeit von 12-, 16- und 32-Bit ADCs

Die Genauigkeit des 12-Bit ADC ist mehr als ausreichend und kann verwendet werden. Diese können direkt in der Software des Nucleo-Boards eingestellt werden.

1.3.5 Zeitliche und geometrische Auflösung

Mit den Angaben des Schrittmotors lässt sich die Positioniergenauigkeit der Schrittmotoren in der y- und z-Achse bestimmen. Bei der Konzeptvariante 1 wird an beiden Achsen jeweils eine Kupplung verbunden, an welcher eine M6-Gewindestange montiert wird. Dabei besitzt eine M6-Gewindestange eine Steigung von 1 mm pro Umdrehung. Die Berechnung der minimalen Höhendifferenz wird in der Abbildung 24 gezeigt und beträgt 5 μm .

Steigung M6-Gewinde	$S_t := 1 \text{ mm}$
Schritte	$S := 200$
Genauigkeit G	$G := \frac{S_t}{S} = 5 \mu\text{m}$

Abbildung 24: Auflösung Schrittmotor y- und z-Achse

Die zeitliche und geometrische Auflösung der y- und z-Achse ist deshalb in der Abbildung 25 dargestellt.

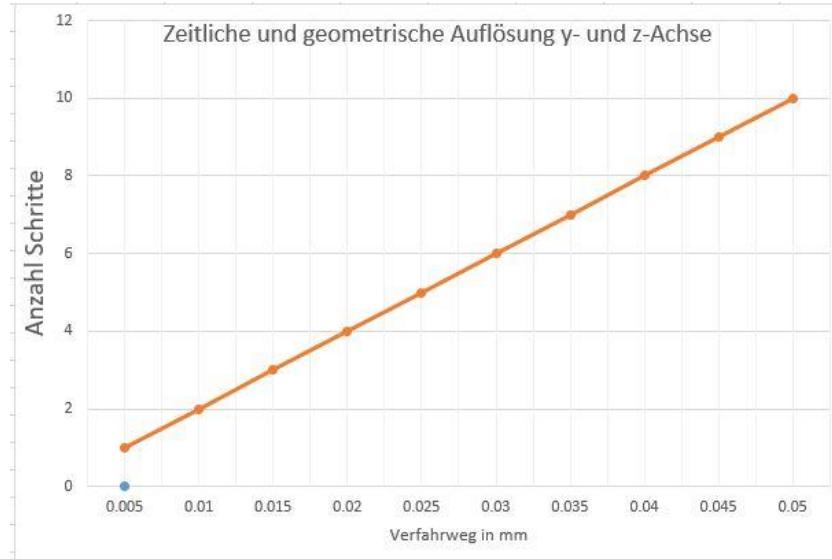


Abbildung 25: Zeitliche und geometrische Auflösung der y- und z-Achse

Zusätzlich wird die zeitliche und geometrische Auflösung der x-Achse in Abbildung 26 vorgestellt. Diese wurde bereits im Kapitel Messtechnik teilweise erläutert

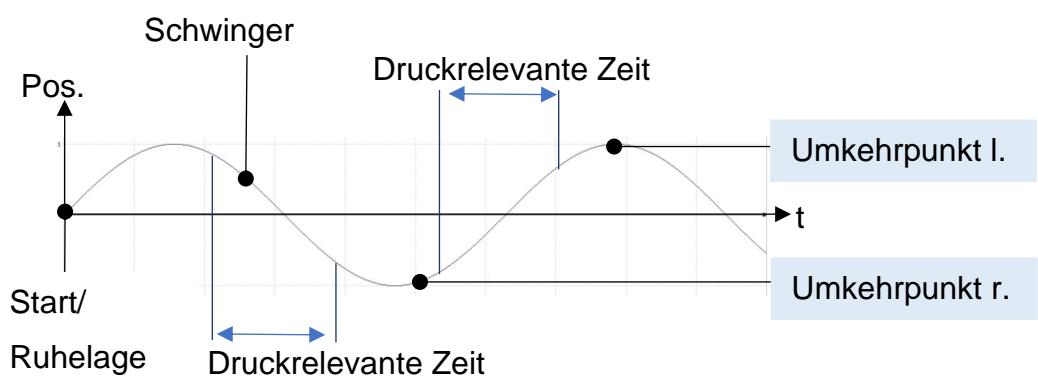


Abbildung 26: Zeitliche und geometrische Auflösung der x-Achse

1.4 Aktorik

An den Aktor der x-Achse werden einige weitere Bedingungen gestellt. Das Prinzip zur Bewegung der x-Achse basiert auf dem Effekt der Lorentzkraft. Diese wird in der Abbildung 27 gezeigt. Dabei wird ein stromdurchflossener Leiter im Magnetfeld abgelenkt.

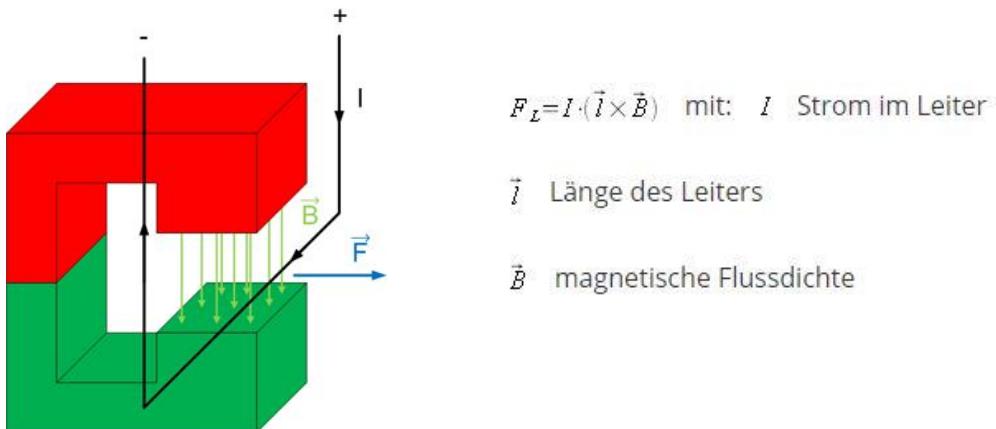


Abbildung 27: Darstellung und Formel Lorentzkraft [5]

Des Weiteren muss der Aktor der x-Achse in der Lage sein, im Betrieb einen zu druckenden Zylinder mit einem Durchmesser von 30 mm zu erzeugen. Außerdem soll er einen Schwinger in einem Takt von 8 bis 10 Hertz antreiben. Der Aktor muss dabei so zu regeln sein, dass es keine großen Schwankungen um die angestrebte Frequenz gibt, da sonst Fehler im Druckobjekt entstehen könnten. Eine weitere Anforderung an den Aktor der x-Achse ist das zu erzeugende Magnetfeld, welches möglichst homogen und linear aufgebaut sein muss um keine Torsion im Schwinger zu erzeugen.

1.4.1 Anforderungen Aktoren der y- und z-Achse

Im Folgenden werden die Anforderungen an die Aktoren der y- und z-Achse aufgelistet. Die y-Achse sowie die z-Achse benötigen außer einem Schrittmotor mit ausreichend Drehmoment und einem kleinen Schrittewinkel (nicht über $1,8^\circ$) keine weiteren zu erfüllenden Anforderungen.

1.4.2 Aktorauslegung

In der Abbildung 28 wird der Aktor für die Erzeugung der Schwingung der x-Achse vorgestellt. Dabei werden das Prinzip sowie die Bauform der zu verwendenden Elektromagneten gezeigt. Der Aufbau des Aktors soll dabei wie in Abbildung 28 aufgebaut werden. Er besteht dabei aus 2 Magneten, die durch einen Abstandhalter aus Kunststoff getrennt werden. In der Mitte zwischen den beiden Magneten wird die Spule zur Erzeugung der Schwingung platziert.

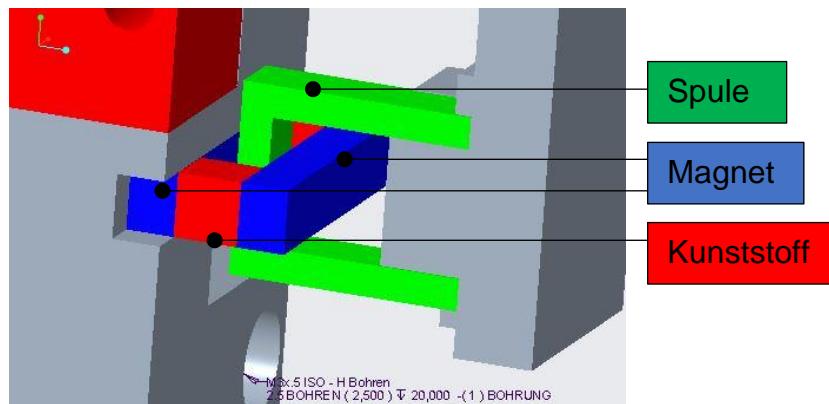


Abbildung 28: Aufbau Aktor x-Achse

1.4.3 Auslegung magnetischer Kreis

In der folgenden Abbildung 29 und Abbildung 30 wird der magnetische Kreis dargestellt. Dabei werden pro Tauchspule 2 Magnete verwendet, die entgegengesetzt angeordnet sind. Das bedeutet, dass der Nordpol eines Magneten auf den Südpol des anderen Magneten zeigt.

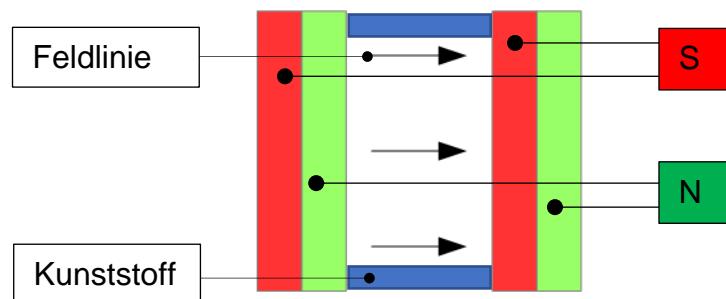


Abbildung 29: Prinzip Aktoraufbau

Beide Magnete werden dabei von einem nicht magnetisierbaren Material getrennt, um die Linearität möglichst konstant zu halten. Der Abstand dabei muss gleichmäßig sein,

Im nächsten Bild wird anschließend ein Leiter (eine Windung der Spule) symbolisch dargestellt. Durch das Magnetfeld der Permanentmagneten (konstant) und der Stromrichtung (variabel) kann dadurch die Lorentzkraft F_L in beide Richtungen umgepolt werden.

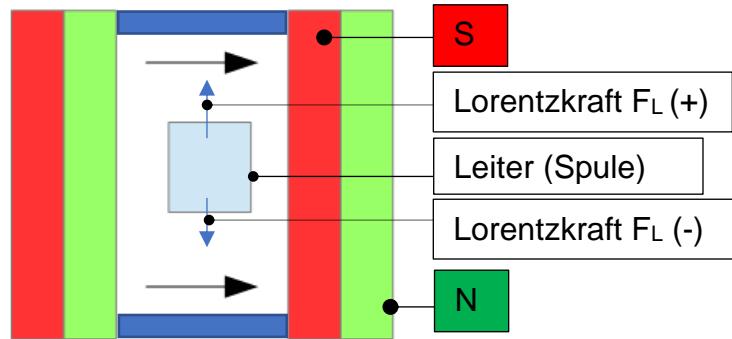


Abbildung 30: Leiter im Magnetfeld

Im praktischen Betrieb wird je nachdem in welche Richtung der Feder-Masse-Schwinger schwingen soll, mittels der H-Brücke die Stromrichtung so angepasst, dass die Lorentzkraft in der gewünschten Richtung entsteht und den Aktor bzw. die Achse in der richtigen Richtung schwingen lässt.

1.4.4 Auswahl Aktorgröße

Da aktuell noch mehrere Stahlbänder (Verwendung als Feder des Schwingers) zur Auswahl stehen sowie der konstruierte Schwinger noch nicht aus einem bestimmten Material gefertigt ist, kann noch kein Gewicht der Achse ermittelt werden, was wiederum eine genaue Berechnung der benötigten Lorentzkraft unsicher macht. Aus diesem Grund wird an dieser Stelle ein vermuteter, recht wahrscheinlicher, Kraftwert berechnet. Hierzu wird ein Strom von 0,5 bis 2 Ampere, ein Magnet aus Neodym (N40-Magnet) mit 0,7 Tesla und eine Leiterlänge im Magnet von 5 mm angenommen. Bei einer angenommenen Anzahl von 30 Windungen ergibt sich eine mögliche zu verwendende Kraft von 0,5 N (bei 0,5 Ampere) bis hin zu einer Kraft von 2,1 N (bei 2 Ampere), wie in Abbildung 31 dargestellt.

Länge Leiter	$l := 0.05 \text{ m}$	$l := 0.05 \text{ m}$	$l := 0.05 \text{ m}$
magn. Flussdichte	$B := 0.7 \text{ T}$	0.7 T	0.7 T
Strom	$I_{0.5} := 0.5 \text{ A}$	$I_1 := 1 \text{ A}$	$I_2 := 2 \text{ A}$
Windungen	$n := 30$	$n := 30$	$n := 30$
Lorentzkraft	$Fl_{0.5} := B \cdot l \cdot I_{0.5} \cdot n = 0.525 \text{ N}$	$Fl_1 := B \cdot l \cdot I_1 \cdot n = 1.05 \text{ N}$	$Fl_2 := B \cdot l \cdot I_2 \cdot n = 2.1 \text{ N}$

Variabler Strom
 zur Einstellung
 der Lorentzkraft

Abbildung 31: Auslegung Elektromagnete

Der berechnete Wert stellt aktuell nur eine Annahme dar und bietet eine mögliche einstellbare Lorentzkraft mithilfe des Stroms. Diese Werte müssen im praktischen Versuch überprüft werden. Da unterschiedliche Stahlbänder mit unterschiedlichem Schwingverhalten vorhanden sind, muss ein gemeinsames Paar ermittelt werden. Zur Einstellung einer geeigneten Lorentzkraft bietet sich die Stromstärke als leicht einzustellende Variable an. Über den sogenannten kdc-Wert der H-Brücke (mehr in Kapitel 2, Elektronik), kann der Strom in einer großen Bandbreite begrenzt werden.

1.5 Elektronik

In diesem Kapitel wird die Elektronik, das Nucleo-Board und die Platine vorgestellt. Die gesamten Bauteile mit allen benötigten elektronischen Bauelementen befinden sich im Anhang.

1.5.1 Genutzte Peripheriemodule am Mikrocontroller

An das verwendete Nucleo-Board STM32F303RET6 können verschiedene Peripheriemodule mithilfe der Platine verbunden werden. Im Folgenden werden die wichtigsten unterschiedlichen Pin-Varianten vorgestellt.

GPIOs

GPIOs sind Pins mit dem Ziel, Signale aufzunehmen oder auszugeben. Ein Beispiel hierfür ist der Türsensor im 3D-Drucker. Der daran angeschlossene Pin wird ein 0- oder 1-Signal des Sensors erhalten und auswerten.

Timer

Pins können eine Timer-Funktion besitzen, um periodische Abläufe wie eine PWM (Pulsweitenmodulation) zu realisieren. Dabei können Eingänge ausgelesen oder Ausgänge periodisch angesteuert werden.

In diesem Projekt wird das PWM-Signal für die Ansteuerung der H-Brücke der x-Achse verwendet.

EXTIs

EXTIs (External Interrupts) sind hauptsächlich GPIOs mit externer Unterbrechung des Hauptprogramms bzw. Main-Programms, um eine wichtige zuvor definierte Funktion aufzurufen und das aktuelle Programm zu unterbrechen.

Beispielweise wird ein EXTI für den Sensor am Türschalter programmiert, um das Programm beim unabsichtlichen Öffnen der Tür aufgrund der gefährlichen Laserstrahlung, zusätzlich zur mechanischen Sicherung, sofort zu unterbrechen.

ADCs

ADC sind Analog-Digital-Konverter und werden verwendet, um analoge Spannungswerte in digitale Werte zu konvertieren. Dies wird benötigt, da der Mikrocontroller selbst keine analogen Werte verarbeiten kann. Alle analogen Messungen müssen hierbei mit einem ADC konvertiert werden. Diese werden für Potentiometer und die Temperaturregelung benötigt.

I²C

I²C ist ein synchroner serieller Datenbus für die Kommunikation zwischen ICs und dient im Projekt der Übertragung von Daten an das Display.

SWD

SWD ist eine 2-Pin-Schnittstelle, die dem Nutzer einen Zugriff auf das System „Memory und Debug-Register“ ermöglicht. SWD befindet sich auf dem Nucleo Board und nicht auf der Platine.

USART

USART steht für Universal Synchronous/ Asynchronous Receiver Transmitter und dient zur Datenübertragung zwischen µC und dem Rechner. USART befindet sich auf dem Nucleo Board und nicht auf der Platine.

1.5.2 Funktionsgruppen und ihre Bauteile

Die einzelnen Funktionsgruppen werden nacheinander aufgelistet.

Ansteuerung x-Achse

Die x-Achse wird zur Bewegung des Lasers verwendet. Die x-Achse (bzw. schnelle Achse) schwingt mit dem Laser in einer Sinus-Schwingung, welche von einer Tauchspule angetrieben wird, die wiederum durch eine H-Brücke angesteuert wird.

Ansteuerung y-Achse

Die Ansteuerung der y-Achse (Achse mit Schrittmotor) dient zur Bewegung des Lasers in y-Achsrichtung. Diese wird von einem Schrittmotor mit geeignetem Schrittmotortreiber durch den Mikrocontroller angesteuert.

Ansteuerung der z-Achse

Die z-Achse sorgt für die Auf- und Ab-Bewegung des Druckbetts im organischen Photopolymer. Nach jeder fertig ausgehärteten Schicht fährt der Objektträger weiter nach unten und führt die Bewegungen in x- und y-Achse erneut aus. Die z-Achse wird ebenfalls von einem Schrittmotor angesteuert.

Laser Ansteuerung

Der Laser wird mithilfe eines High-Signals vom Mikrocontroller angesteuert. Dabei wird ein Lasertreiber zwischen Laser und Mikrocontroller geschalten. Der Lasertreiber ist leistungsfähiger (mehr Strom als μC) und schaltet dementsprechend den Laser ein und aus.

Heizung Ansteuerung

Die Temperatur des Polymers darf bis maximal 60 °C aufgeheizt werden und muss während des Druckvorgangs konstant bleiben, um eine gleichmäßige Aushärtung des Polymers zu erreichen. Es wird ein Temperaturwert um 55°C angepeilt. Die Ansteuerung der Polymertemperatur erfolgt über einen MOSFET.

1.5.3 Beschreibung des detaillierten Schaltplans

Zunächst wird eine Übersicht angefertigt. Die Spannungsversorgungen unterteilen sich dabei in die Spannungen 3,3; 5 und 12 Volt. Des Weiteren werden die Bauelemente in vier Gruppen eingeteilt, die im Folgenden vorgestellt werden:

Sensorik

Sensoren messen physikalische Effekte und geben diese in Form von elektrischen Signalen an den Mikrokontroller weiter. Lichtschranke benötigt dabei 3,3 und 12 Volt.

Ansteuerung

Nucleo-Board, das alle Signale sammelt, verarbeitet und entsprechende Aktoren ansteuert. Das Nucleo-Board kann entweder mit 3,3, 5 oder 12 Volt betrieben werden.

Leistungselektronik

Alle Aktoren und deren Treiber mit vergleichsweise hohen Strömen. Die Spannungswerte liegen hierbei zwischen 5 und 12V.

Bauelemente wie Schrittmotoren und Tauchspule, die den Strom aus den entsprechenden Treibern in eine gewünschte Bewegung umwandeln.

In der folgenden Abbildung 32 wird dieser Zusammenhang in einem Schaubild sehr vereinfacht dargestellt. Dabei werden die vier benannten Gruppen nach den Eigenschaften und der benötigten Spannung sortiert.

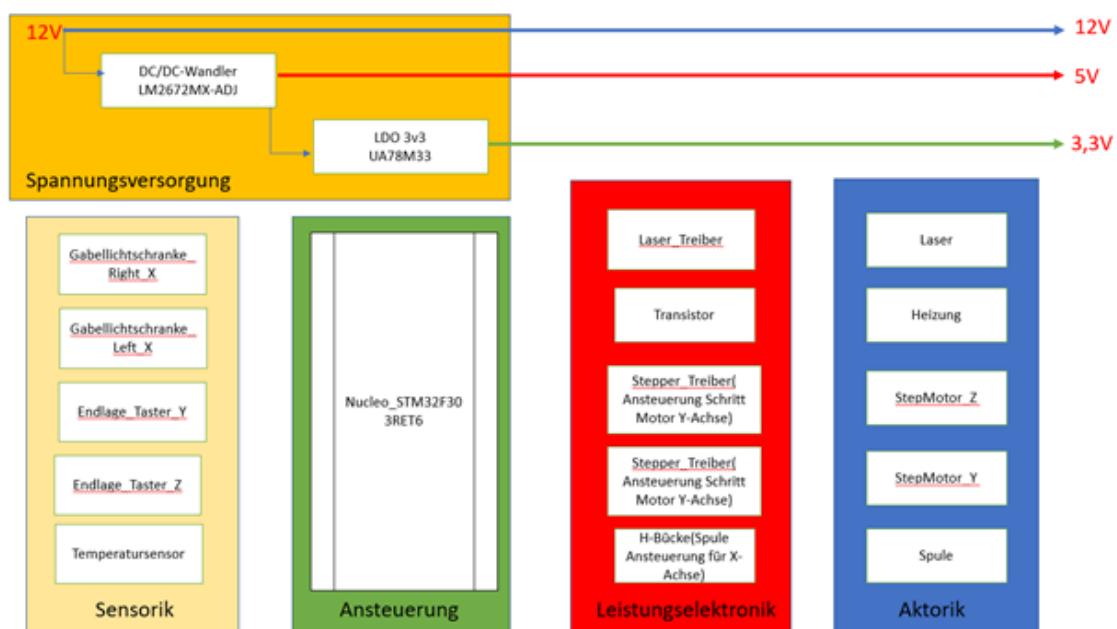


Abbildung 32: Darstellung Baugruppen und Spannungsversorgung

Prüfung auf Stromstärke der unterschiedlichen Stromkreise

In der folgenden Tabelle 4 befinden sich die Bauteile mit deren benötigtem Spannungs- und Strombedarf. Die Liste wird zum Erstellen des Stromlaufplans verwendet und soll nur einen ungefähren Bedarf ermitteln, um die Auswahl geeigneter Spannungswandler und Leiterbahnbreiten zu validieren. Es geht dabei lediglich darum, die verfügbaren Spannungswandler nicht zu überlasten bzw. die Leiterbahnbreiten nicht zu schmal (Erhitzungsgefahr) zu konstruieren.

Tabelle 4: Spannungs- und Strombedarf der Bauteile

Komponenten	Spannung	Strom (angenommene Werte)
Laser-Modul	12 V	0,2 A
Tauchspule	12 V	2 A
Schrittmotor y-Achse	12 V	0,8 A
Schrittmotor z-Achse	12 V	0,8 A
Heizung	12 V	1 A
H-Brücke (Steuerung)	5 V	13 mA
Nucleo_STM32F303	5 V	250 mA
Gabellichtschranke rechts	3,3 V	20 mA
Gabellichtschranke links	3,3 V	20 mA
Endlage Mikroschalter Y	5 V	10 mA
Endlage Mikroschalter Z	5 V	10 mA
Türschalter	5 V	10 mA
I ² C -Display	3,3 V	15 mA
Temperatursensor	3,3 V	20 mA
Potentiometer	3,3 V	10 mA
Potentiometer	3,3 V	10 mA

Daraus kann die entsprechende Summe des Stromverbrauchs und die Leistung ermittelt werden (s. Tabelle 5).

Tabelle 5: Stromverbrauchsumme und Leistung

Spannung	Stromverbrauch Summe	Leistung U*I
12 V	4,8 A	57,6 W
5 V	293 mA	1,465 W
3,3 V	95 mA	0,3135 W

Spannungsversorgung

Es wird der Schaltregler LM2672MX-ADJ/NOPB verwendet. Der LM2672MX-ADJ/NOPB liefert eine Ausgangsspannung von 5 Volt und einen maximalen Ausgangstrom von 1 Ampere bei einem Wirkungsgrad von 90 %. Der Schaltregler liefert eine justierbare Ausgangsspannung, die durch einen Spannungsteiler festgelegt wird. Es wird eine 5 Volt Festspannung benötigt. Aus diesem Grund wird die zu berechnende Formel in Abbildung 33 dargestellt.

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

where

- $V_{REF} = 1.21 \text{ V}$

Abbildung 33: Schaltregler Berechnungsformel (aus [6])

Laut Hersteller soll R1 hierbei zwischen 240Ω und $1,5 \text{ k}\Omega$ ausgewählt werden.

Wenn $R1 = 1\text{k}\Omega$ beträgt lässt sich für R2 in Abbildung 34 folgendes ausrechnen:

$$\begin{aligned} R_1 &:= 1 \text{ k}\Omega & V_{out} &:= 5 \text{ V} & V_{ref} &:= 1.21 \text{ V} \\ R_2 &:= R_1 \cdot \left(\frac{V_{out}}{V_{ref}} - 1 \right) & & & &= 3.132 \text{ k}\Omega \end{aligned}$$

Abbildung 34: Berechnung Schaltregler Widerstandswert R2

Aus diesem Grund wird in der E12-Widerstandstabelle nach einem geeigneten Widerstand gesucht. Dabei fällt die mögliche Wahl zwischen 2,7 und 3,3 kΩ. In diesem Fall wurde aufgrund des geringeren Abstands der Widerstandswert 3,3 kΩ gewählt. Der Linearregler UA78M33 hat eine feste Ausgangsspannung von 3,3 Volt und einem maximalen Strom von 500 mA (s. Tabelle 6). Damit ist dieser für den 3D-Drucker ausreichend.

Tabelle 6: Linearregler Modell UA78M33 (siehe [7])

Linearregler UA78M33	
Möglicher Strom	500 mA
Benötigter Strom	155 mA

Der verwendete Schaltregler ist eine bekannte Störquelle (EMV-Effekte) und muss daher weit entfernt von störempfindlichen Bauteilen angebracht werden und dessen „Power Ground“ (hier: PGND bezeichnet) von anderen, speziell analogen Bauteilen (hier: AGND) getrennt werden. Die Elektrolyt-Kondensatoren an dem Ein- und Ausgang des Schaltreglers dienen zur Stabilisierung der Spannung. Die Folien-Kondensatoren hingegen sorgen für eine schnelle Stromaufnahme beim hochfrequenten Umschalten. Die Spule ist ein Energiespeicher und verlangsamt den Strom. Die Schottky Diode wird als eine Freilauf-Diode verwendet und führt etwaige Spannungsspitzen auch bei hohen Frequenzen ab. Die Leitungen auf der Platine haben eine hohe parasitäre Induktivität und müssen so kurz wie möglich gehalten werden. Bei der elektromagnetischen Verträglichkeit muss jede Leiterbahn wie eine störende Antenne betrachtet werden. In der Abbildung 35 wird der Stromlaufplan der Spannungsversorgungen mit 3,3, 5 und 12 Volt gezeigt. Die Stromlaufpläne sind im Anhang als technische Zeichnung vergrößert dargestellt.

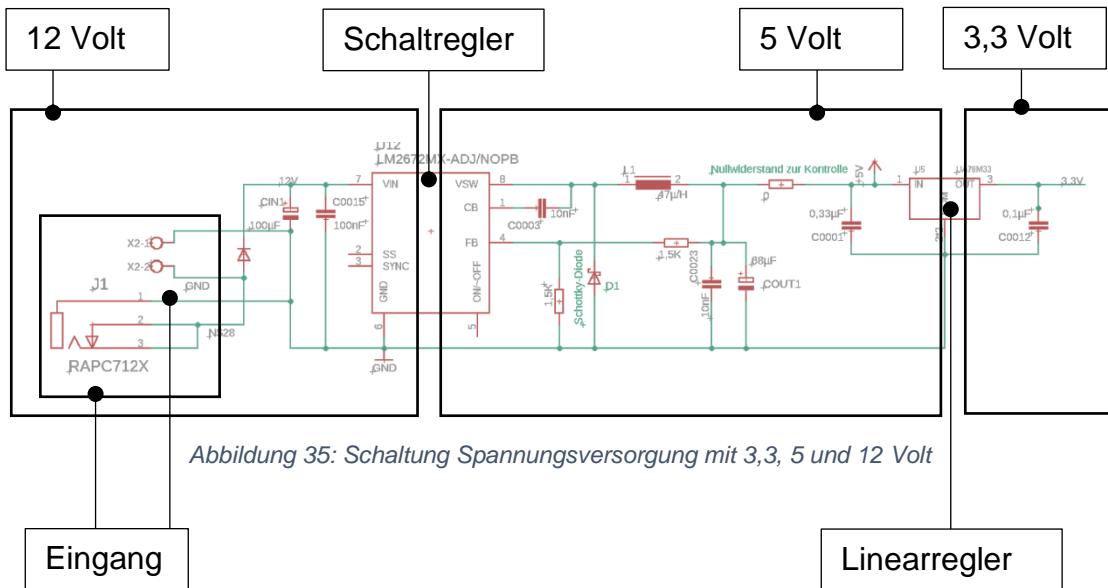


Abbildung 35: Schaltung Spannungsversorgung mit 3,3, 5 und 12 Volt

Schrittmotorentreiber der y- und z-Achse

Es wird je ein Schrittmotor für die Bewegung der y- und z-Achse verwendet. Dabei wird für jeden Schrittmotor ein Schrittmotortreiber benötigt. Die Auswahl der Schrittmotoren und Schrittmotortreiber ist dabei das bereits vorgegebene Modell A4982. Der A4982 wird in diesem Projekt mit 5 Pins an dem Mikrocontroller verbunden. Dabei wird an jedem Pin ein Schutzwiderstand von $1\text{ k}\Omega$ bis $3,3\text{ k}\Omega$ vorgesehen.

Die R_{Sense} -Pins des Motortreibers werden an sehr niederohmige Widerstände (siehe Abbildung 36) verbunden und sollen den maximalen Ausgangsstrom der Schrittmotortreiber bestimmen, um die Schrittmotoren nicht zu überlasten. Da in diesen Widerständen ein hoher Strom fließen wird, können diese nicht in der sonst verwendeten Bauform 0603 eingesetzt werden. Die Berechnung der R_{Sense} -Widerstände wird in Abbildung 36 gezeigt. Dabei ergibt sich ein Wert von $0,516\ \Omega$.

$$V_{REF} := 3.3 \text{ V} \quad I_{max} := 0.8 \text{ A}$$

$$R_s := \frac{V_{REF}}{8 \cdot I_{max}} \quad R_s = 0.516 \text{ } \Omega$$

Abbildung 36: Berechnung R_{Sense} -Pins

Die Widerstände können aufgrund ihrer Baugröße nicht seriell auf die Platine eingesetzt werden. Aus diesem Grund werden lediglich 2 Widerstände pro Schrittmotor mit $200\text{ m}\Omega$ verwendet.

Als weitere Beschränkung des Stroms wird ein Potentiometer am Eingang V_{ref} des Schrittmotortreibers angebracht, um den Strom am Ausgang des Schrittmotortreibers zusätzlich definieren zu können. In der Abbildung 37 wird der Stromlaufplan der Schrittmotortreiber gezeigt.

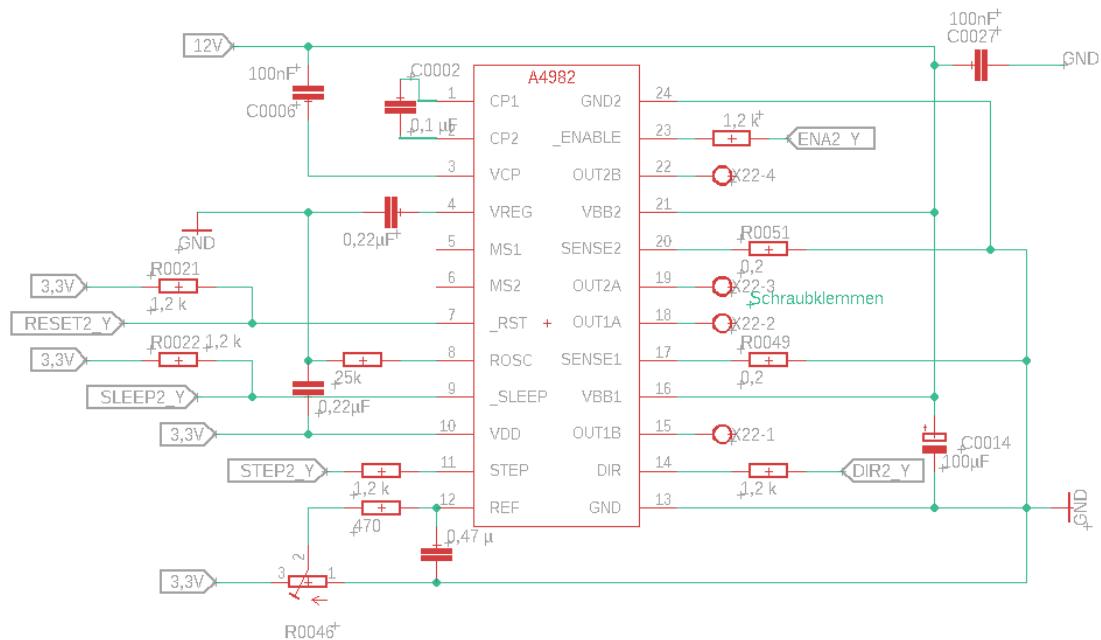


Abbildung 37: Stromkreis Schrittmotortreiber

H-Brücke für x-Achse

Die Tauchspule zur Erzeugung der Schwingung der x-Achse wird mit einer H-Brücke angetrieben, die wiederum mit einem PWM-Signal des Mikrocontrollers angesteuert wird. Als H-Brücke ist das Modell IFX9201SG Modell vorgegeben.

Der Aufbau der Schaltung erfolgt dabei nach dem Datenblatt. Dabei sind 4 Pins der H-Brücke mit Schutzwiderständen mit dem Mikrocontroller verbunden. Die Kondensatoren an der Spannungsversorgung (VS-Pin) werden Ladungspumpe genannt und nach der Empfehlung des Herstellers eingesetzt. Zusätzlich werden die 33 Nanofarad Kondensatoren an den Outputs 1 und 2 eingesetzt. Der Stromlaufplan der H-Brücke wird hierbei in der Abbildung 38 dargestellt.

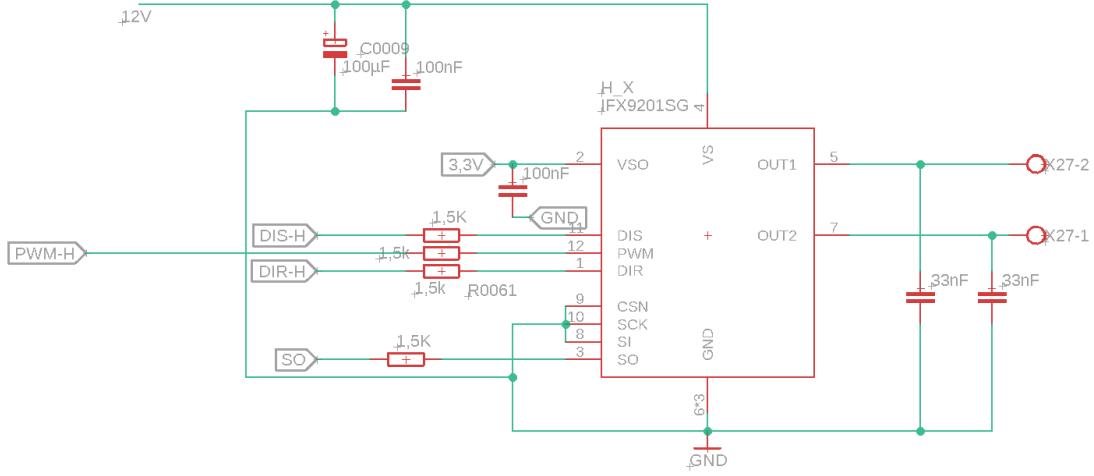


Abbildung 38: Ansteuerung der H-Brücke

Lichtschranke

Eine Lichtschranke ist ein optoelektronischer Sensor, der bei einer Unterbrechung des Lichtstrahls ein entsprechendes Datensignal ausgibt.

Nach Datenblatt müssen vor der LED der Lichtschranke Vorwiderstände angebracht werden, um den maximalen Strom durch die LED zu begrenzen. Die LED verursacht hierbei einen Spannungsabfall von bis zu 1,7 V und benötigt einen Strom von 20 mA. Maximal darf die LED einen Strom von bis zu 40 mA führen. Für die Berechnung des Vorwiderstands wird folgende Rechnung in Formel I ausgeführt.

$$R_D = \frac{3,3 \text{ V} - 1,7 \text{ V}}{20 \text{ mA}} = 80 \Omega \quad \text{Formel I}$$

Da zu diesem Augenblick noch nicht sicher festgelegt ist, dass der Drucker nur mit einer Lichtschranke funktioniert, wird auf der Platine ein Slot für eine weitere Lichtschranke angefertigt. Dies dient lediglich als Hilfe, falls bei diesem System eine Änderung benötigt wird.

Der Schaltplan der beiden Lichtschranken wird in Abbildung 39 gezeigt. Das Ausgangssignal der Lichtschranke (Signal „Lichtschranke_Rechts_X“ und „LichtschrankeLinks_X“) muss auf 3,3 Volt reduziert werden, um die Elektronik des

Nucleo-Boards nicht zu zerstören. Aus diesem Grund wird ein Spannungsteiler aus dem internen Widerstand ($6,8\text{ k}\Omega$) und einem Widerstand mit $3,9\text{ k}\Omega$ gewählt.

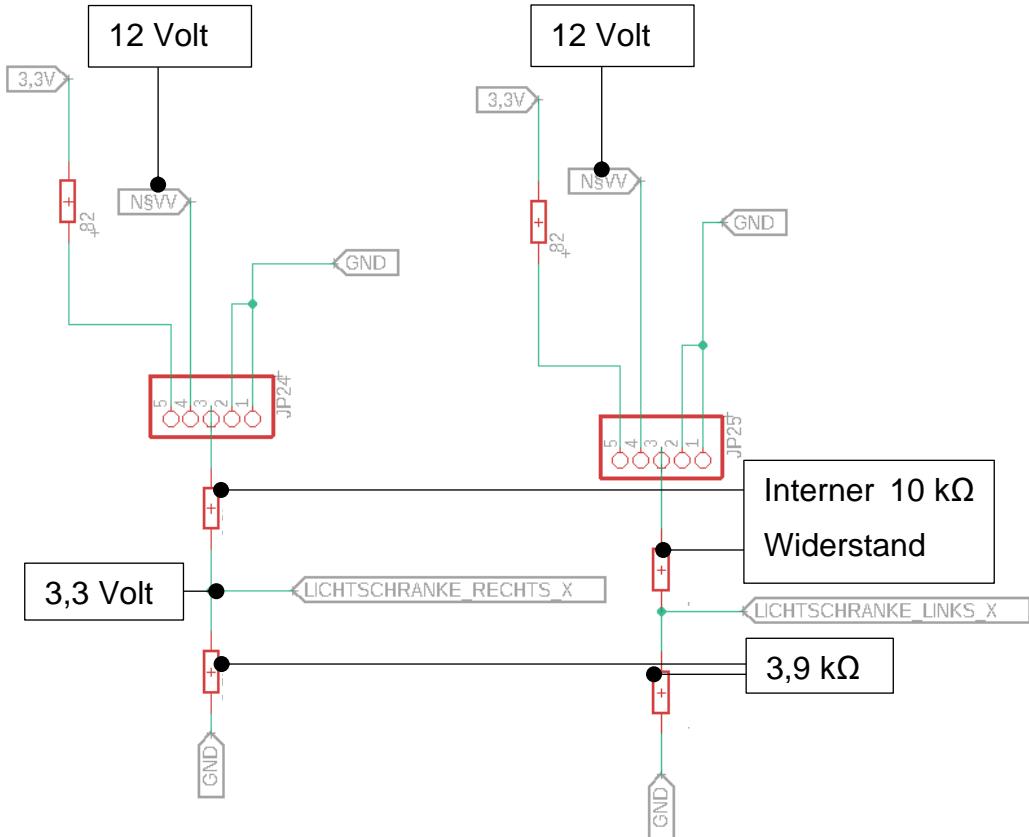


Abbildung 39: Stromkreis der Lichtschranken

Heizungsteuerung

Für die Heizungsteuerung wird ein MOSFET eingeplant.

Der MOSFET steuert je nach anliegender Spannung am „Gate-Anschluss“, d.h. bei nicht angelegter Spannung am Gate isoliert der MOSFET die Verbindung zwischen Drain und Source. Dann verhält sich der MOSFET wie ein offener Schalter und lässt keinen Strom fließen. Bei anliegender Spannung wird der MOSFET leitend. Es wird das Modell TSM320N03CX ausgewählt. Dieser kann bis maximal 30 Volt zwischen Drain und Source trennen und bei guter Kühlung einen maximalen Strom von 5,5 Ampere führen.

Der TSM320N03CX ist nicht in der Eagle Library zu finden, deswegen wird ein MOSFET in Eagle mit der gleichen Gehäuseform SOT23 verbaut (siehe Abbildung 40).

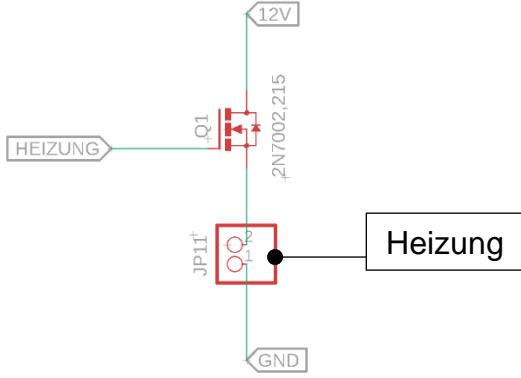


Abbildung 40: Spannungsversorgung Heizstromkreis

Zur Validierung, dass keine extra Kühlkörper für den MOSFET gebraucht werden, muss die folgende Berechnung in Abbildung 41 durchgeführt werden.

$I := 1 \text{ A}$	$R_{DSon} := 32 \cdot 10^{-3} \Omega$	R_{DSon} : Widerstand zwischen Drain und Source
		I: Strom der Heizung
$P := R_{DSon} \cdot I^2$	$P = 0.032 \text{ W}$	P: Heat Dissipation
$T_{jmax} := 150 \text{ }^{\circ}\text{C}$	$T_A := 25 \text{ }^{\circ}\text{C}$	$R_{\theta JA} := 124 \text{ }^{\circ}\text{C} \frac{1}{W}$
$P_D := \frac{T_{jmax} - T_A}{R_{\theta JA}} = 0.315 \text{ W}$		T_{jmax} : max Junction Temperature
		T_A : Ambient Temperature
		$R_{\theta JA}$: Junction to Ambient Thermal Resistance
		P_D : Power Dissipation
+		
$P < P_D$	True	Der MOSFET braucht keine Kühlkörper

Abbildung 41: Berechnen der Erwärmung des MOSFETs

Spannungsversorgung Laser (mit Sicherheitseinrichtung)

Der Lasertreiber wird mit einem Pin_Set Befehl vom Mikrocontroller auf High und Low gesetzt. Der Lasertreiber wurde bereits von vorherigen Gruppen getestet und benötigt eine Spannung von 12 Volt. Die Spannungsversorgung des Lasermoduls ist wie in Abbildung 42 dargestellt, in Reihe mit dem Endschalter der Tür verschalten, um eine Hardware Abschaltung des Lasers bei Öffnen der Tür zu ermöglichen.

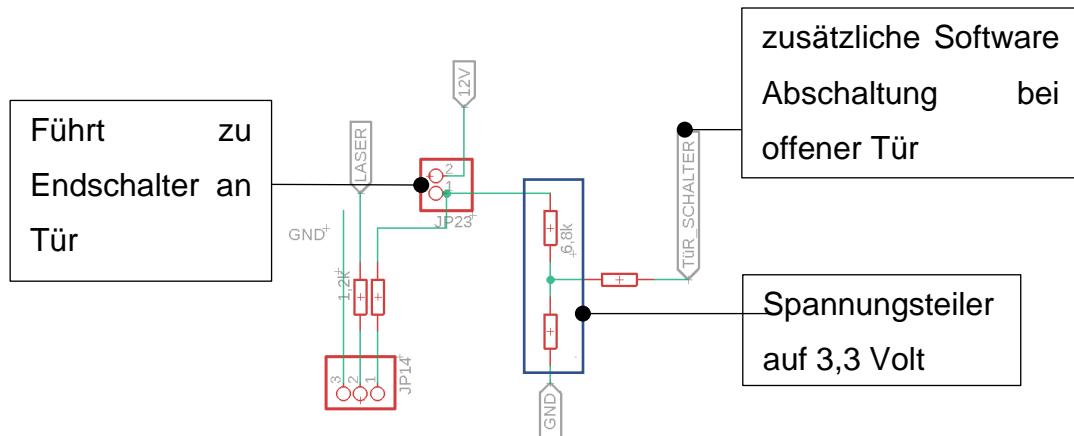


Abbildung 42: Spannungsversorgung Stromkreis Laser

NucleoF303RE

Die Zuweisung von Pinbelegungen (siehe Abbildung 43) wird im Kapitel Software vorgenommen, da die Pins nach ihren möglichen Eigenschaften zugewiesen werden.

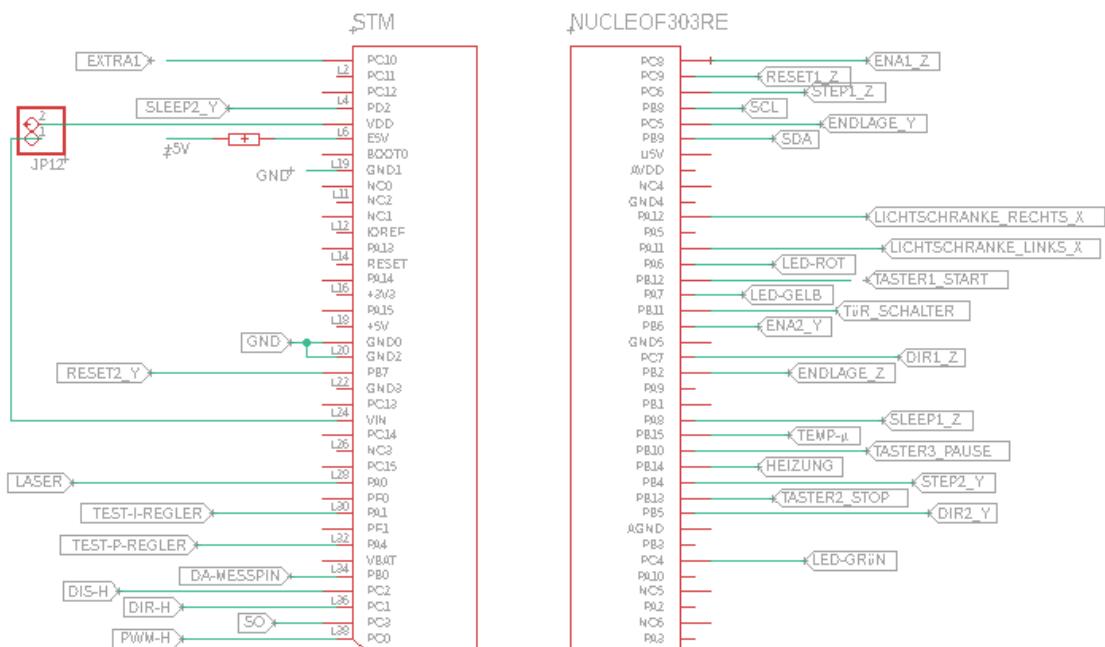


Abbildung 43: Nucleo Schalt- und Kontaktplan

1.5.4 Bestimmung Leiterbahnbreite

Es wird die Leiterbahnenbreite anhand der fließenden Stromstärke bestimmt. Angenommen wird eine Raumtemperatur von 25 °C und einer möglichst geringen Erwärmung der Leiterplatte auf nicht über 50 °C.

Nach dem Labor Design Rule Check des FVM-Labors (Regeln des Mechatronik-Labors) darf die minimale Leiterbahnhöhe (Thickness DRC) nicht unter 0,035 mm und die minimale Breite 0,3 mm betragen. Die Leiterbahnbreiten wurden mit folgendem Muster, wie in Abbildung 44, überprüft. Dabei geben die Ergebnisse nur den Mindestwert an, der bei Möglichkeit höher gewählt werden kann.

Inputs		Internal Layers		External Layers in Air	
Strom	Current: 0.5 A	Required Trace Width: = 0.1722550733 mm	Required Trace Width: = 0.06621528225 mm	Resistance: = 0.000000000 Ω	Resistance: = 0.000000000 Ω
Thickness: 0.035 mm	Voltage Drop: = 0.000000000 V	Voltage Drop: = 0.000000000 V	Ambient Temperature: 25 °C	Power Loss: = 0.000000000 W	Power Loss: = 0.000000000 W
Temperature Rise: 25 °C	Power Loss: = 0.000000000 W	Trace Length: Enter Length... in			

Abbildung 44: Leiterbahnbreite [8]

Dabei ist auch zu beachten, dass die Durchmesser der Bohrungen an das Werkzeug der Fertigung im Labor angepasst werden. Als Durchmesser eines Vias (Durchkontaktierung beider Layer) wird eine Bohrung von 0,4 mm ausgewählt.

1.5.5 Beschreibung Layout

Das Layout wird in Abbildung 45 vorgestellt. Die blauen Linien stehen hierbei für den Layer Bottom, also für die Unterseite der Leiterplatte. Die roten Leiterbahnen stehen für den Layer Top und zeigen die Oberseite der Leiterbahnen.

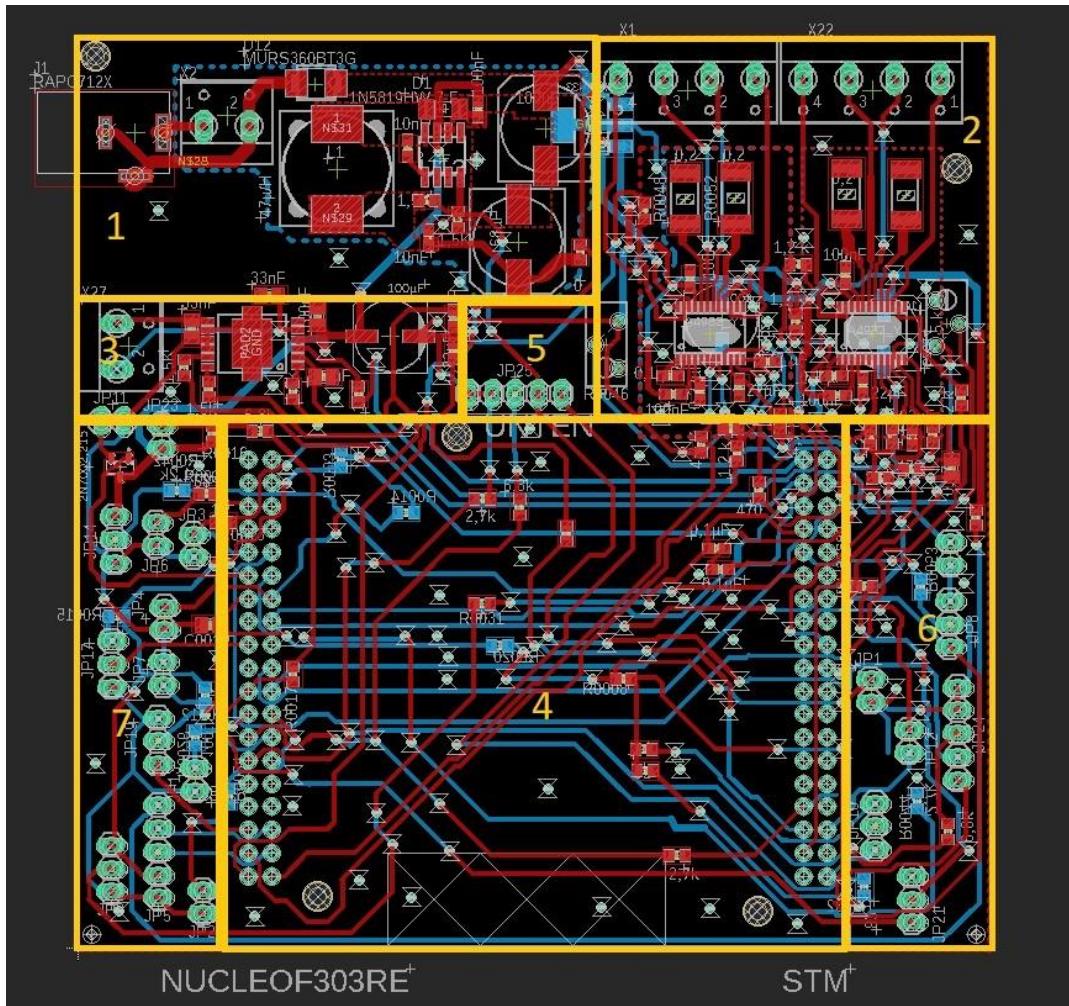


Abbildung 45: Platine Layout

1 Spannungsversorgung:

Das Layout für die Spannungsversorgung, den Schaltregler und dem LDO wird nach Anweisung des Datenblatts gefertigt. Dabei müssen auf die Leiterbahnen der Kondensatoren und der Diode des Schaltreglers besonders geachtet werden.

Es ist außerdem auf die Stromstärke und die daraus resultierende Leiterbahnbreite zu achten. Die Spule muss sehr nah an den Schaltregler platziert werden, da sie ein „störendes“ induktives Bauteil ist. Die Spannungsversorgung ist absichtlich an der oberen Ecke der Platine platziert, um eine möglichst gute Isolierung zu der Gruppe Sensorik zu erstellen. Die Spannungsversorgung ist außerdem eher nah an der Gruppe Aktorik, damit die Ströme der Aktoren nicht über die gesamte Platine zurückfließen müssen.

2 Stepper Treiber:

Die Stepper-Treiber für die Motoren der y- und z-Achse gehören zur Gruppe Aktorik und sind nah an der Spannungsquelle platziert, um die Leitung dorthin kurz zu halten. Die Anschlüsse für beide Motoren sind dabei Schraubklemmen. Das Layout zur Platzierung der Stepper-Treiber ICs wurde nach der Empfehlung des Herstellers angefertigt.

3 H-Brücke:

Die H-Brücke gehört ebenfalls zur Gruppe Aktorik und ist nah an der Spannungsquelle platziert, um die Leitung kurz zu halten. Die Anschlüsse sind ebenfalls Schraubklemmen.

4 Nucleo:

Das Nucleo-Board wird auf die Oberseite der Platine mithilfe der Buchsenleisten aufgesteckt. Durch Platzmangel der Platine (Maße 100 x 100 mm) werden verschiedene Bauteile unter dem Board angebracht. Das Nucleo Board ist die steuerbare Logik der Elektronik und ein recht sensibles Element, weswegen es weit von der Spannungsversorgung und der Aktorik platziert wurde.

5, 6 und 7 Stiftleisten:

Für Anschlüsse, die keine hohen Ströme führen, werden die Stiftleisten verwendet. Diese sind am Rand der Platine montiert, um eine einfache Handhabung zu ermöglichen, ohne andere Bauteile zu beschädigen. Folgende Elemente werden via Stiftleiste an das Nucleo kontaktiert:

- Lichtschranken
- Temperatur-Sensor
- Bedientaster
- Endlagetaster
- Potentiometer zum Testen des P- und I-Reglers
- LEDs Statusanzeige (grün, rot, gelb)
- Heizung MOSFET
- Mess-Pins

1.5.6 Vorversuche zu ausgewählten Funktionsgruppen

Die Platine wurde zunächst im Labor gefertigt, als auch später extern bestellt, da der Lötstopplack in der Beuth Hochschule aktuell nicht aufgetragen werden konnte und der Lötprozess deshalb nicht erfolgreich war. Die bestellten Platinen werden anschließend für den Lötvorgang verwendet. Die Vorversuche zu den einzelnen Funktionsgruppen werden im weiteren Verlauf ergänzt. In den folgenden Abbildung 46 und Abbildung 47 sind die beiden Ebenen Top und Bottom dargestellt.

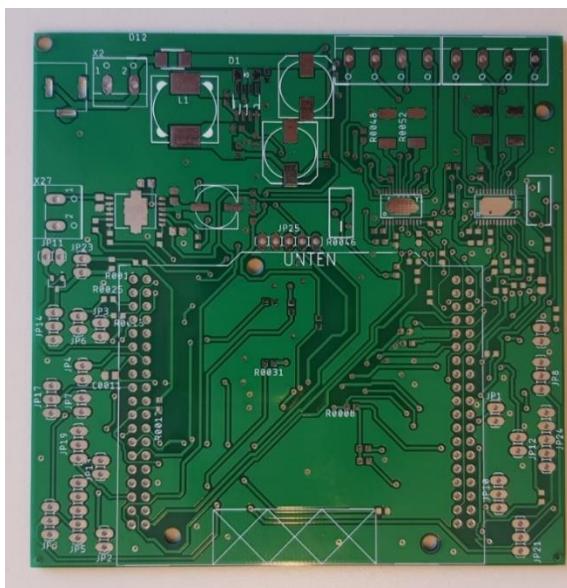


Abbildung 46: Top Layer Platine

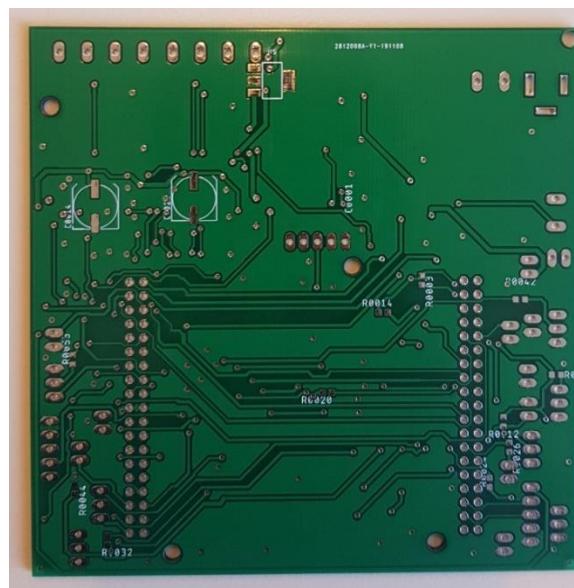


Abbildung 47: Bottom Layer Platine

1.5.7 LT-Spice Simulationen zu kritischen Schaltungselementen

LTS spice ist eine Freeware von der Firma „Linear Technology Corporation“, die die Simulation von analogen Schaltungen ermöglicht. Die Software basiert auf SPICE, welches für "Simulation Program with Integrated Circuit Emphasis" steht. Seit der Version 8.4 von Eagle kann keine Datei mehr an LTS spice exportiert werden, da Eagle ein neues eigenes Modul installiert hat.

Das in Eagle integrierte Simulationsprogramm ist allerdings noch in der Entwicklungsphase und hat keine vollständigen Bibliotheken für die hier verwendeten Bauteile. Daher wurde auf eine SPICE Simulation verzichtet.

1.6 Software

Im Folgenden wird die Software für Treiber, H-Brücke und andere Bauelemente vorgestellt

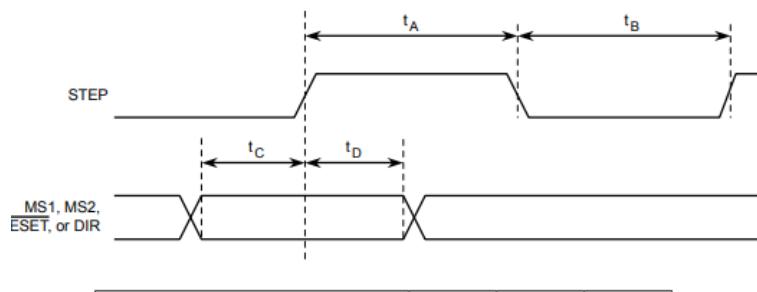
1.6.1 Funktionale Beschreibung der Treiber

Die einzelnen Baugruppen werden für die Programmierung dargestellt.

Schrittmotor Treiber A4982

Nach Datenblatt hat der Treiber A4982 fünf zu verbindende Logik-Anschlüsse, die in diesem Projekt verwendet werden. Diese werden im folgenden kurz erläutert.

STEP: Dieser Pin erhält ein PWM Signal vom µC und dreht den Motor pro Signal um einen Schritt weiter. Die kleinste für den Betrieb benötigte PWM hat eine Periodendauer von 2 µs und 50% Dutycycle, d.h. das Signal ist mindestens 1 µs auf High und anschließend 1 µs auf Low. Aus der nachstehenden Abbildung 48 „Step Sequencing Settings“ kann der Verlauf der zeitlichen Abhängigkeit betrachtet werden.



Time Duration	Symbol	Typ.	Unit
STEP minimum, HIGH pulse width	t _A	1	µs
STEP minimum, LOW pulse width	t _B	1	µs
Setup time, input change to STEP	t _C	200	ns
Hold time, input change to STEP	t _D	200	ns

Abbildung 48: Logic Interface Timing Diagramm [9]

Dabei werden in der Tabelle 7 die restlichen 4 Pins des Schrittmotortreibers kurz erläutert.

Tabelle 7: Beschreibung Pin-Eingänge Schrittmotortreiber A4982

Pin	Beschreibung
DIRECTION	bestimmt die Drehrichtung des Schrittmotors
ENABLE	kann Ausgänge komplett Ein- und Ausschalten
RESET	setzt den Schrittmotortreiber zurück
SLEEP	stellt den Motor im Schlafmodus bzw. minimalen Verbrauchsmodus. Beim High-Signal normaler Zustand, bei Low-Signal werden alle Funktionen deaktiviert.

H-Brücke IFX9201SG

Nach Datenblatt hat die H-Brücke IFX9201SG in diesem Projekt 4 Logik Pins, die an den Mikrocontroller angeschlossen werden müssen. Diese werden im Folgenden erläutert und in Tabelle 8 aufgelistet.

DIS: Disable, deaktiviert alle Funktionen, wenn der Eingang auf High gesetzt wird.

DIR: Direction, bestimmt die Richtung des Stroms an den Ausgängen, Wechsel durch Umschalten des Direction-Pins zwischen einem 1- und 0-Signal.

PWM: Empfängt das PWM Signal aus dem µC und schaltet die internen Mosfets dem Eingang entsprechend.

Tabelle 8: Beschreibung Pins der H-Brücke [10]

Pin	Beschreibung
DIRECTION	Bestimmt Drehrichtung
DISABLE	kann Ausgänge komplett Ein- und Ausschalten
PWM	Schaltet Mosfets entsprechend PWM-Signal

Anhand der Tabelle 9 „Output Truth Table“ kann die Stromrichtung an den Ausgängen ermittelt werden. Die notwendige logische Ansteuerung zum Aktivieren der H-Brücke wird in Tabelle 9 dargestellt.

Tabelle 9: Output Truth Table [10]

DIS	PWM	DIR	OUT1	OUT2	Comment
1	X	X	Z	Z	disabled, outputs tristate
0	1	1	H	L	forward / clockwise
0	1	0	L	H	reverse / counterclockwise
0	0	1	H	Z	freewheeling in HS (forward)
0	0	0	Z	H	freewheeling in HS (reverse)

1.6.2 Funktionale Beschreibung der Interrupts und Timer

Eine ISR (Interrupt Service Routine) ist eine Funktion, die das laufende Main-Programm unterbricht, um eine zuvor bestimmte Funktion sofort aufzurufen.

Das geschieht über ein Ereignis, das beispielsweise unregelmäßig, bzw. extern ist, wie zum Beispiel durch Druck auf einen Taster oder das Abnehmen der Deckel des Druckers. Im 3D-Drucker werden dann EXTI (externe Interrupts) für die Taster und Türschalter verwendet. Als internes Interrupt kann ebenso ein Timer verwendet werden. Der Timer zählt dann, bis der ARR-Wert (Auto-Reload-Register) den zuvor definierten Zielwert erreicht hat, dann tritt eine ISR auf und unterbricht das Main-Programm. Anschließend wird eine zuvor definierte Funktion ausgeführt. Danach wird das Main-Programm an der gestoppten Stelle weiter ausgeführt.

Um ein paralleles Auslösen verschiedener Interrupts zu verhindern, müssen die verschiedenen ISRs mit einer Priorität belegt werden. Die höchste Priorität (Nucleo mit Wert 0) findet zuerst statt, dann folgen die anderen ISR mit niedriger Priorität in chronologischer Reihenfolge. Eine spezielle Pinbelegung für die Zuweisung der Prioritäten ist nicht erforderlich.

Erzeugung PWM-Signal

Ein PWM Signal hat zwei interne technische Parameter, den Duty Cycle (DC) und der Frequenz. Ein Duty Cycle ist das zeitliche Verhältnis zwischen High-Signal einer Periode zur gesamten Dauer einer Periode. Die Frequenz des PWM-Signals hängt von der Länge der Periodendauer ab. Bei einem schnellen Wechsel zwischen ON/ OFF und einem entsprechenden Duty Cycle kann das Ausgangssignal als niedrigere Gleichspannung betrachtet werden. Um beispielweise eine Gleichspannung mit 2 Volt zu simulieren, kann das Signal aus der PWM mit 5 Volt (High) und 0 Volt (Low) mit einem Duty Cycle von 40% eingestellt werden. Das Ausgangssignal berechnet sich dann mit 5 Volt mal 40 % (Duty Cycle) auf eine 2 Volt Ausgangsspannung.

Ein Timer kann verwendet werden, um ein PWM Signal zu generieren und den Duty Cycle zu definieren. Dabei gilt der sogenannte Wert CCRy. Dieser ist ein „Zwischenwert“, zwischen 0 und dem maximalen Zählerwert ARR. Die Formel zur Berechnung ist in der Formel II zu sehen

$$TIM_{xARR} = \frac{f_{CLK} * T}{(TIM_{xPSC} + 1)} - 1 \quad \text{Formel II}$$

Solange der interne Zähler kleiner als der Wert CCRy ist, ist der entsprechende Ausgangs-Pin der PWM auf 1 (high), beim Überschreiten dieses Wertes wird am PWM-Pin ein 0-Signal eingestellt.

Im 3D-Drucker wird eine PWM durch Timer bei den folgenden Komponenten erzeugt:

- H-Bücke
- 2 Schrittmotorentreiber
- Heizung (MOSFET)

a. Erzeugung eines Impulses

Mit einem ähnlichen Prinzip wie der PWM kann stattdessen auch nur ein einmaliger Impuls an einem Pin-Ausgang ausgegeben werden. Dieser ist dann einmalig und nicht periodisch. Der Timer wird dabei durch einen Trigger gestartet und hält nach einmaligem Durchlauf an.

Dabei ist die Konfiguration im Programm CubeMX anders einzurichten, d.h. der Channel bleibt inaktiv, bis der Zähler den Wert zum Auslösen des Impulses erreicht hat. Der Impuls-Modus wird im 3D-Drucker für den Laser verwendet.

b. Input Capture-Modus

Ein Timer hat zusätzlich ein Capture-Modus im Eingangs-Pin, wodurch er eine steigende oder fallende Flanke am Eingang erfassst und anschließend ein Interrupt, bzw. einen Zählvorgang startet. Das kann verwendet werden, um eine Frequenz oder

Pulsbreite ausmessen zu können. Dies wird für die Lichtschranke verwendet, um die Frequenz des Signals zu messen und die Zeiten der Schwingungen zu ermitteln.

1.6.3 Funktionale Beschreibung des Main-Programms

In der Programmiersprache C beginnt die Programmausführung mit der Hauptfunktion „Main ()“. Jedes C-Programm muss deshalb eine „Main ()“ -Funktion enthalten. Diese Hauptfunktion kann beliebige Anweisungen ausführen. Dabei werden diese Anweisungen nacheinander chronologisch in der Reihenfolge ausgeführt. Nach Bearbeiten der Einstellung im Programm CubeMX wird einen C-Code mit der jeweiligen ausgeführten Toolchain (bei uns STM32Workbench) generiert.

Die „Main.c“ enthält dabei die vorgestellte Architektur:

#include: Die Main Funktion holt den Zugriff auf die inkludierten Bibliotheken bzw. Header Dateien, die für bestimmte Funktionen, Klassen oder Variablen notwendig sind.

- #define: definiert einen Wert oder eine Funktion.
- typedef: definiert einen neuen Typ für entsprechende Variablen und kann den Code abkürzen.
- extern: Funktionen oder Variablen, die aus einer anderen Quell-Datei aufgerufen werden.
- Private Variablen: sind Variablen, die vom Nutzer erstellt und privat zu betrachten sind.
- Private Funktion Prototypen: sind Funktionen, die nach dem „Main-Programm“ geschrieben sind.
- int main ()
definiert die Main-Funktion im gesamten Code.
- while (1){
wiederholt einen Vorgang nach Anzahl vorgegebener Wiederholungen. Die 1 in Klammer steht hierbei für eine unendliche Wiederholung}

1.6.4 Test- und Inbetriebnahmekonzept

Zum Test bzw. zur Kontrolle der getätigten Ergebnisse wird eine modulare Vorgangsart erstellt. Dabei bedeutet modular, dass jedes Bauteil (wie Lichtschranke, H-Brücke) zunächst einzeln getestet wird, um die korrekte Funktion und Verschaltung der Elektronik zu überprüfen. Dabei wird nach dem Muster in Tabelle 10 vorgegangen.

Jeder Arbeitsschritt muss erfolgreich verlaufen sein, um zum nächsten Bauteil zu wechseln. In der 3. Spalte „Prüfen“ wird angegeben, wie der Kontrollvorgang im weiteren Verlauf getestet werden soll.

Tabelle 10: Planung Inbetriebnahmekonzept

Baugruppe	Test	Prüfen
Licht-schranken	LED bestromt?	Mit Handykamera die LED im Betrieb betrachten, um Infrarotlicht zu sehen
	Unterbrechung registriert?	Messen der Spannung mit Multimeter
Schritt-motoren	Können beide Schrittmotoren angesteuert werden?	Verfahren der Schrittmotoren um 10 Schritte
	Drehrichtung ändern möglich?	Ändern des Direction-Pins in Programmcode
	Erhitzen sich Motoren bzw. Treiber zu stark?	Prüfung mit Laserthermometer
H-Brücke	Werden beide Ausgänge bestromt?	Anschließen zweier LEDs an Ausgänge der H-Brücke, Wechsel in 0,5 Sekunden Takt
Magnet und Aktor	Schwingung des Aktors (und Resonanzfrequenz) vorhanden?	Test der Schwingung mithilfe eines Funktionsgenerators

2 Detailkonstruktion

In diesem Kapitel werden die angedachten Planvorstellungen realisiert und Details des 3D-Druckers ausgearbeitet. Dazu zählt das Fertigen der konstruierten Teile, das Testen der Sensoren und Aktoren und die erste Programmierung des Codes für jede einzelne Baugruppe.

2.1 Regelungstechnik

Es werden die drei geplanten Regelungen des 3D-Druckers im Folgenden beschrieben. Dazu zählen die Heizungsregelung, die das Photopolymer auf bis zu 60 °C halten soll, die Einhaltung des Phasenversatzes zwischen Anregungswelle und dem mechanischen Schwinger sowie die Größe der Amplitude des Schwingers, um so viel Kraft mit dem Elektromagnet in den Schwingvorgang zu geben, dass eine gleichmäßige Schwingung mit im Druckbereich weitestgehend linearer Geschwindigkeit entsteht.

2.1.1 Vorüberlegungen zu den Regelparametern

Um das Photopolymer für den 3D-Druckbetrieb besser nutzbar zu machen (zum schnelleren Aushärten), muss dieses auf eine Temperatur von 60 °C aufgeheizt werden. Da keine großen Temperaturschwankungen aufgrund der Trägheit des Regelkreises zu erwarten sind, erscheint ein Zweipunktregler als ausreichend. Die Funktion der Temperaturmessung soll hierbei in einem Abstand von 10 Sekunden überprüft werden. Der Regelkreis der Temperatur ist in der Abbildung 49 zu sehen.

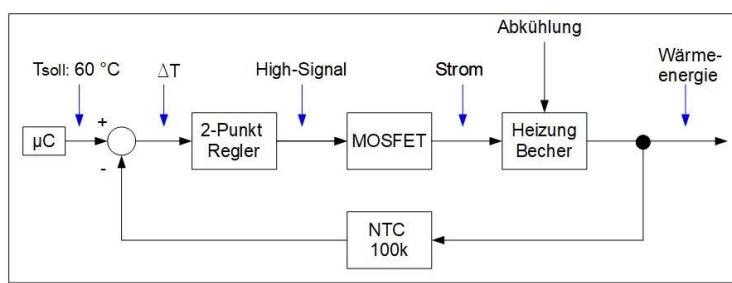


Abbildung 49: Regelkreis Temperatur

Im weiteren Verlauf wird der Regelkreis der Amplitudenregelung ausgearbeitet. Die Vorüberlegungen zum Amplitudenregler werden aus der allgemeinen Formel eines PI-Reglers erstellt und die 5 Regelparameter in der Tabelle 11 benannt.

Tabelle 11: Regelgrößen in PI-Amplitudenregelung

Regelparameter	Beschreibung	Erklärung
Sollgröße	Verhältnis der Zeiten T_{Hell} zu T_{Dunkel}	Ziel: Verhältnis T_{Hell} zu T_{Dunkel}
Regelabweichung	ΔV Verhältnis Hell- zu Dunkelzeit	Abweichung zwischen Soll- und Istwert Verhältnis
Stellgröße	kdc-Wert: Eine Art Duty-Cycle der H-Brücke, der Wert kdc gibt an, für wie viel Prozent der Zeit die H-Brücke bestromt wird.	kdc muss auf die entsprechenden Gegebenheiten angepasst und der Wert für kdc_{\min} und kdc_{\max} muss festgelegt werden.
Störgröße	Dämpfung durch Leitungen, Schräger Stand des Druckers	-
Regelgröße	Zeit T_{Hell} , aber dadurch auch die Frequenz	8 bis 9 Hz

Dazu wird als Sollwert das Verhältnis zwischen T_{Hell} und T_{Dunkel} ausgewählt. Durch die Signale der Lichtschranke kann anschließend die Regeldifferenz e zwischen dem Sollwert des Verhältnisses und dem tatsächlichen Ist-Wert bestimmt werden. Anschließend wird Fehlersumme esum, sowie Verstärkungsfaktoren kp und ki festgelegt und je nach Regelbedarf der Faktor kdc (On-Time H-Brücke) nachgeregelt.

Als Störgröße des Regelkreises (s. Abbildung 50) wird die Dämpfung durch Leitungen des Lasers erkannt. Diese haben einen großen Einfluss auf den gesamten Regelkreis.

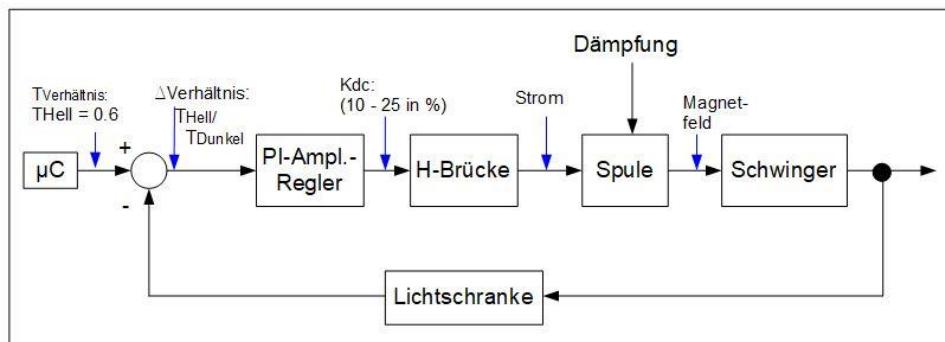


Abbildung 50: Überlegung Amplitudenregelung

2.1.2 Erste Inbetriebnahme der Baugruppen

Die weitere Planung zur Inbetriebnahme ist in Tabelle 12 dargestellt.

Tabelle 12: Erste Inbetriebnahme der Baugruppen

Fachbereich	Aufgabe	Beschreibung
Elektronik	Löten Platine	Platine bestücken und löten
	Test Spannungsversorgung	Kontrolle auf 5 und 3.3 V in Ordnung
Mechanik	Schwinger fertigen	Drucken/ Fräßen Schwinger
	Montage Schwinger	Montieren des Schwingers
	Spule anfertigen/ anbringen	Spulen wickeln und testen
Software	Vorbereitung Testprogramme für jede Baugruppe	Vorbereitung simpler und einfacher Programme zum ersten Test der Funktion
Ziel	Jede Baugruppe wurde einzeln positiv getestet und funktioniert. Debuggen von Elektronik und Mechanik beendet.	

2.2 Mechanik

Es wird die gesamte Mechanik detailliert ausgearbeitet und aufgelistet. Die Einzelteilzeichnungen befinden sich hierbei im Anhang.

2.2.1 Detailkonstruktion aller Teile unter Berücksichtigung der Fertigbarkeit

Im Folgenden werden die konstruierten Elemente so ausgewählt, dass sie im Mechatronik-Fertigungslabor der Beuth fertigbar sind. Deshalb wird versucht, vorhandene Materialien effektiv zu nutzen, um den Material- und Arbeitsbedarf zu verbessern. Die folgende Abbildung 51 zeigt die fertige CAD-Darstellung von dem konstruierten 3D-Drucker ohne Deckel.

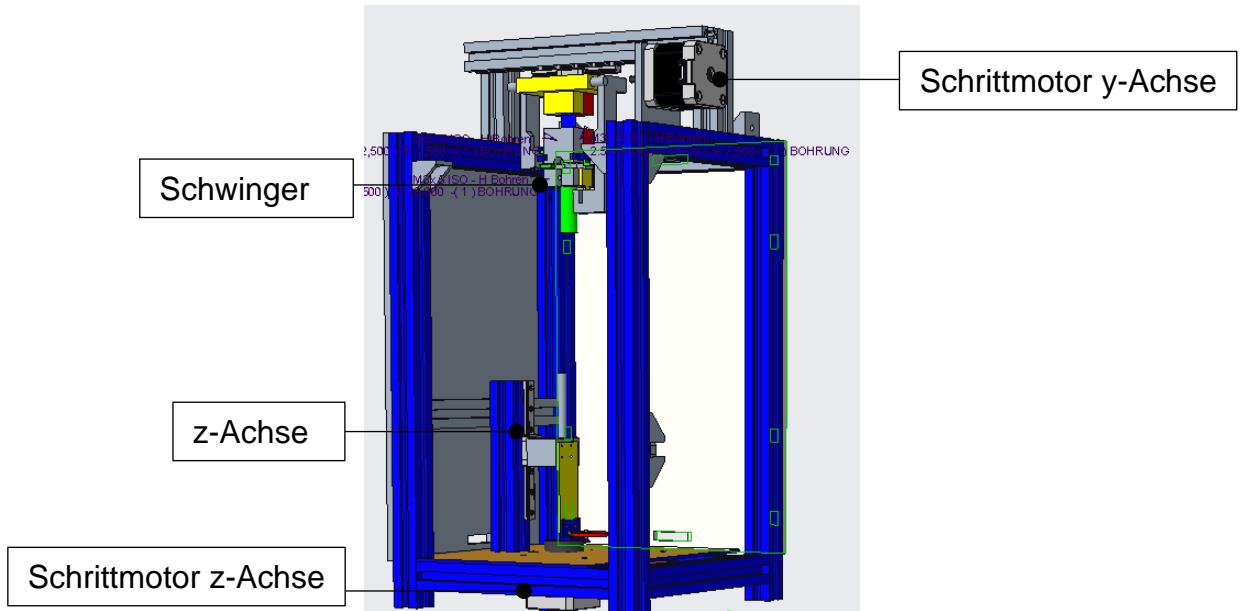


Abbildung 51: CAD Darstellung des 3D - Drucker

Es wird die Konzeptvariante 1 mit den zwei Gewindestangen für y- und z-Achse ausgewählt. Ein Antrieb über eine Gewindestange ist in diesem Fall aufgrund des Bauraums einfacher realisierbar als der Antrieb über einen Riemen. Der Abstand vom Laserkopf zum Druckbett wird später so eingestellt, dass die Entfernung ca. 150 mm beträgt. In der Abbildung 52 wird der Aufbau des Schwingers sowie anschließend Realisierung dargestellt. Der Swinger wird aus Photopolymer mithilfe eines 3D-Druckers gedruckt (siehe Abbildung 53), da Materialien wie Aluminium im Labor nicht praktikabel und zeitnah fertigbar sind.

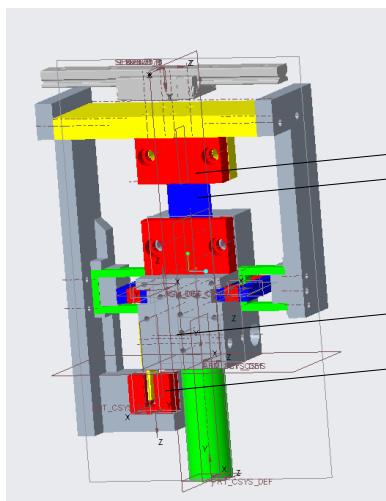


Abbildung 52: Darstellung Swinger

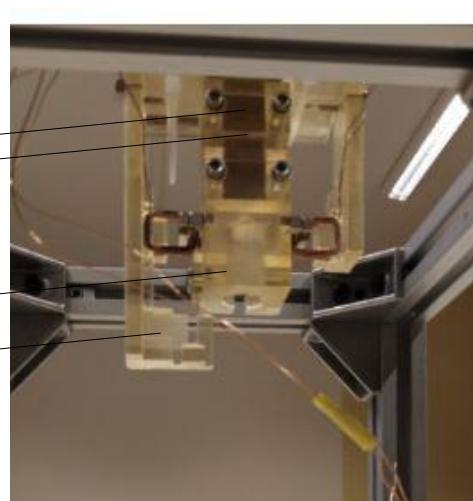


Abbildung 53: Swinger in Aufbauphase

Zudem wird die Blende des Schwingers, wie in Abbildung 55 dargestellt, angefertigt. Die Blende hat dabei einen besonderen Zweck, da sie die Größe des Druckbereichs festlegen wird. Das Prinzip wird in der Abbildung 54 dargestellt. Die Blende wird so konzipiert, dass bei einem Eintreten in der Lichtschranke das Programm zur Erstellung des Kreises aktiviert werden soll. Je größer/ breiter die Blende gefertigt wird, desto größer könnte der potenzielle Druckraum praktikabel gestaltet werden.

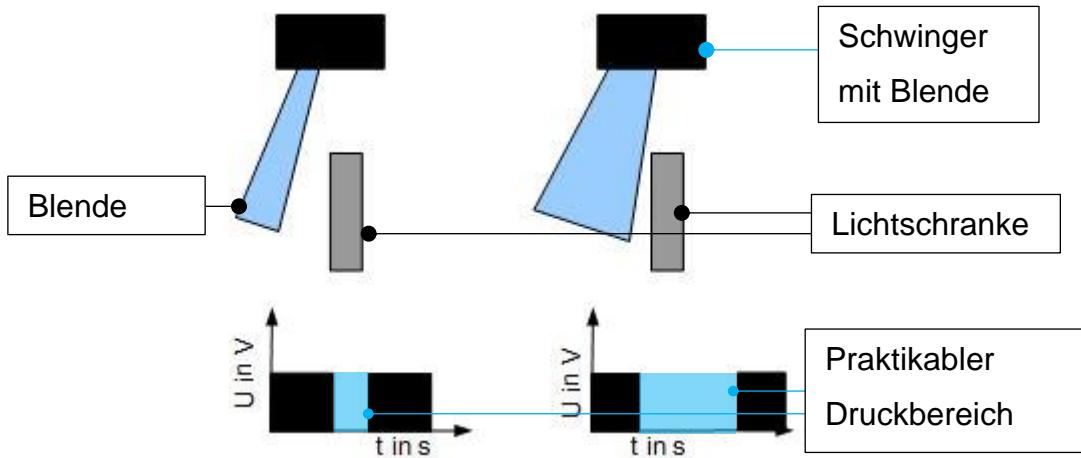


Abbildung 54: Festlegen einer geeigneten Blende

Durch die Breite der Blende ist es möglich, den Druckbereich an verschiedenen Stellen festzulegen. Wird (wie hier) davon ausgegangen, dass der Druck bei einer steigenden Flanke beginnt, so kann der Bauraum dadurch vergrößert werden, in dem das 1-Signal der Lichtschranke früher auftritt und später wieder austritt. Anschließend werden die Spulen erstellt und der Schwinger montiert (siehe Abbildung 55).

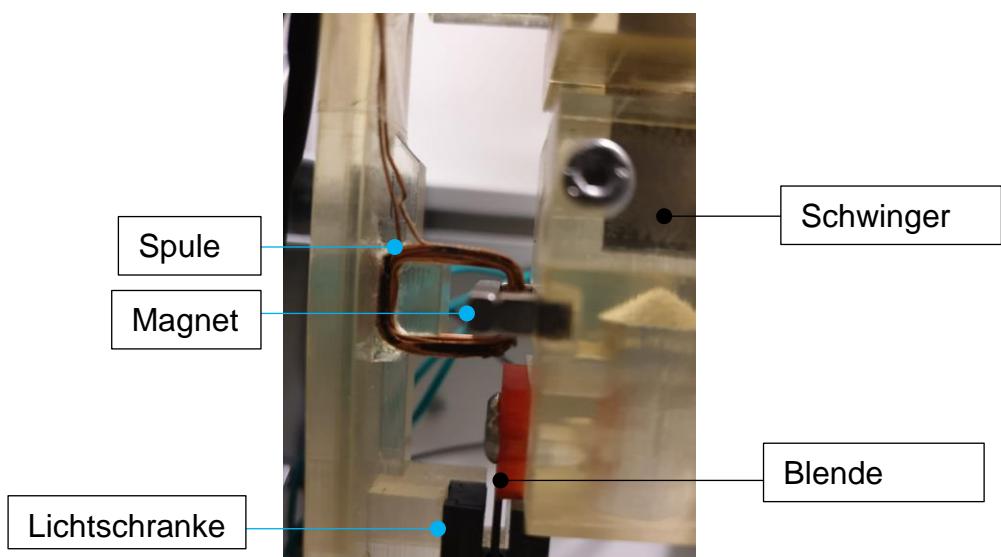


Abbildung 55: Prinzip Antrieb Schwinger und Lichtschranke

In Abbildung 56 wird die geplante z-Achse mit Becher und Antrieb dargestellt. Diese wird ebenfalls angefertigt.

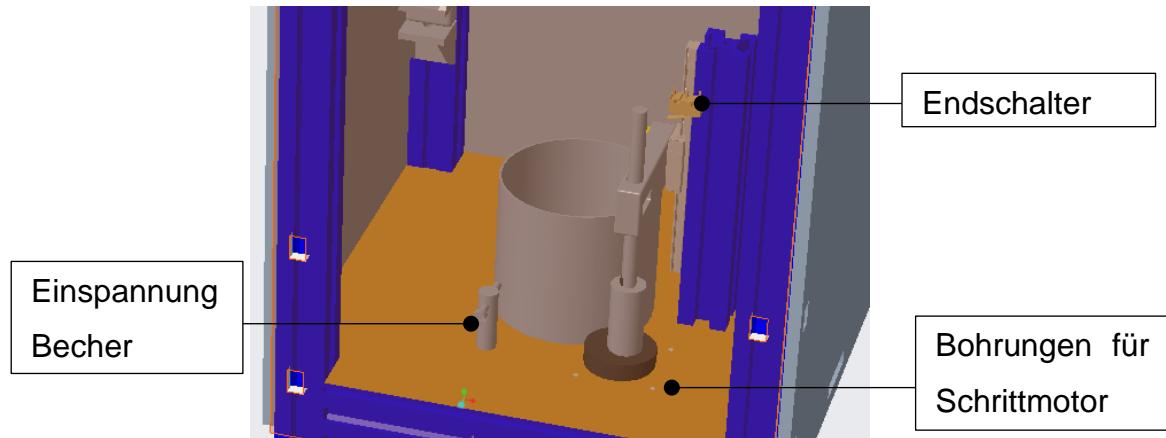


Abbildung 56: Einspannung Druckbecher und z-Achse

Die gefertigte z-Achse (mit neuem nachträglich geändertem Druckbett) ist in der Abbildung 57 dargestellt.

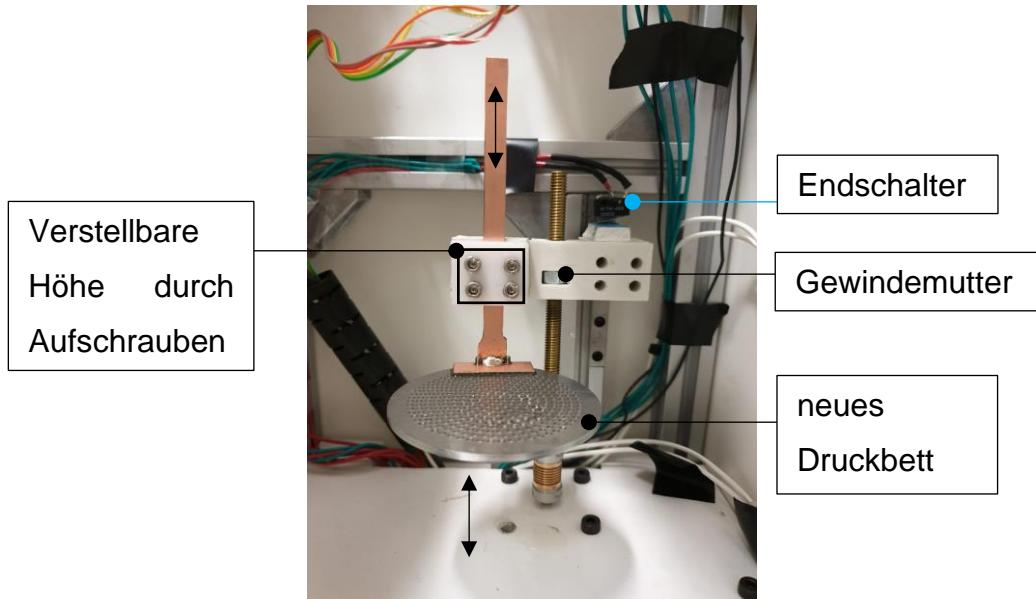


Abbildung 57: Gefertigte z-Achse

Durch die geklemmte Einspannung ist eine Verschiebung des Druckbetts in z-Richtung (Hoch und runter) durch Lösen der Schrauben möglich, um den Becher mit Polymer entsprechend einstellen zu können. Im Druckbett wurden Löcher vorgesehen, sodass das Polymer beim Eintauchen auf die Oberfläche des Druckbetts gedrückt wird.

Der für die Belichtung verwendete Laser ist für den menschlichen Körper (speziell die Augen) gefährlich und somit darf das Laserlicht nicht aus dem 3D-Drucker austreten. Hierfür muss ein entsprechendes Gehäuse mit einer verschließbaren Tür angebracht

werden. An der Tür wird ein Türschalter (Endschalter) verbaut, sodass zu jedem Zeitpunkt sichergestellt werden kann, dass die Tür verschlossen ist. Das hierfür angefertigte Gehäuse ist in der Abbildung 58 dargestellt.

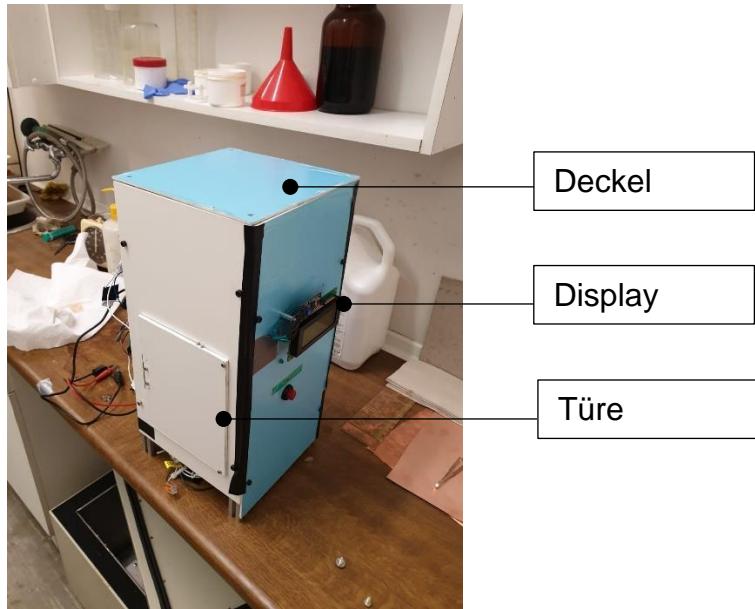


Abbildung 58: 3D-Drucker mit Gehäuse

Die Gehäuseplatten und der Gehäusedeckel bestehen aus einem Verbundwerkstoff mit Kunststoffanteil und wurden entsprechend mit den Profilstangen verschraubt. Es wurde darauf geachtet, dass das Gehäuse lichtdicht verschlossen ist und der Laser somit auf keinen Fall ausdringen kann. Die mit einem Scharnier eingebaute Tür ist in der Abbildung 59 erkennbar.

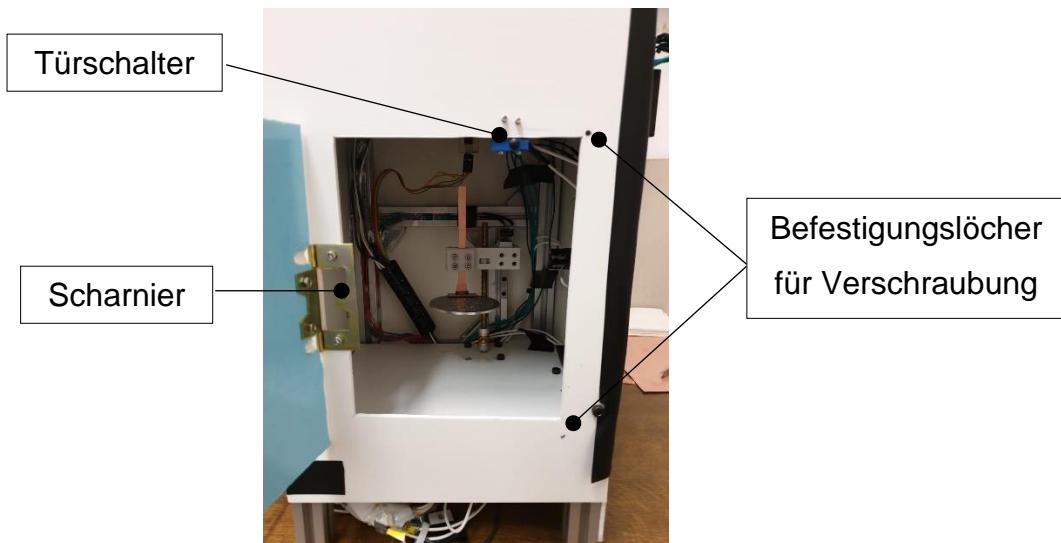


Abbildung 59: Tür mit Türschalter

Beim Schließen der Tür wird der Türschalter betätigt und der Mikrocontroller erhält ein entsprechendes Signal. Durch die Befestigungslöcher wird sichergestellt, dass die Tür während des gesamten Druckprozesses geschlossen bleibt, da bei nicht geschlossener Tür der Druckprozess sofort abgebrochen wird (Soft- und Hardwareabschaltung des Lasers).

2.2.2 Beschreibung relevanter Toleranzketten

Die wichtigsten Toleranzen sind bereits in der Abbildung 20, Abbildung 21 und Abbildung 22 dargestellt. In folgender Tabelle 13 werden weitere zu berücksichtigende Toleranzen aufgelistet, die möglicherweise einen Einfluss auf das Druckergebnis nehmen könnten.

Tabelle 13: Relevante Toleranzen

Achse	Baugruppe	Toleranz	Beschreibung
y-Achse	Schrittmotor	Toleranz durch Mindest-Schrittgröße	kleinste verfahrbare Länge des Schrittmotors ist 5 µm
	Linearführung	Abweichungen Geraadlinigkeit	max. 6 µm (bei 100 mm Schiene)
	Gewindestange, Kupplung	Gewindestange (Abweichung der Geraadlinigkeit); Kupplung federt (dadurch Spiel),	Gewindestange mit Abweichung der Geraadlinigkeit (ähnlich Führung); Motor mit federnder Kupplung an Gewindestange verbunden, dadurch Spiel bei Start eines Schrittes (speziell bei Hin- und Rückfahrt der y-Koordinate)
z-Achse	Führung	Geraadlinigkeit	Max. 5 µm bei 70 mm Schiene
	Schrittmotor	Toleranz durch Mindest-Schrittgröße	kleinste verfahrbare Länge des Schrittmotors ist 5 µm
	Gewindestange, Kupplung	Gewindestange (Abweichung der Geraadlinigkeit)	Gewindestange mit Abweichung der Geraadlinigkeit (ähnlich Führung); Motor mit federnder

		Geradlinigkeit); Kupplung federt (dadurch Spiel),	Kupplung an Gewindestange verbunden. Da im Druck die z-Achse nur in eine Richtung bewegt wird, ist es nicht so ausschlaggebend wie in der y-Achse
x-Achse	Schwinger mit Sollfrequenz	Frequenz ändert sich sehr leicht	Frequenz bleibt nicht komplett konstant auf einem sehr genauen Wert, sondern schwingt leicht um die Eigenfrequenz.
	Schwinger in Sinus-Schwingung	Wellenform	Der Laser wird im annähernd linearen Teil der Sinuswelle verwendet. Dennoch ist dieser Bereich nicht exakt linear, wodurch Abweichungen festzustellen sind
	Lichtschranke	Offset	Die Lichtschranke kann nicht so exakt ausgerichtet werden. Aus diesem Grund muss mit Offset gearbeitet werden

2.3 Sensorik

Im Folgenden werden die Testaufbauten der Sensorik vorgestellt.

2.3.1 Testaufbauten zur Positionsmessung

Es werden Tests zur Funktionsprüfung der Lichtschranke und Endschalter durchgeführt. Die Positionsmessung der y- und z-Achse erfolgt mit dem Endschalter der jeweiligen Achse. In diesem 3D-Drucker werden die Endschalter als Öffner verwendet. Das heißt, dass ein nicht betätigter Endschalter ein 1-Signal und ein betätigter Schalter ein 0-Signal am Pin ausgibt. Dies wird aus Sicherheitsgründen verwendet, da ein defekter als Schließer ausgelegter Endschalter, ein Sicherheitsrisiko

darstellt (da der µC sonst erst ein 1-Signal erwartet, wenn der Endschalter betätigt ist). Bei einem Defekt würde der Schrittmotor so lange rotieren, bis er festklemmt.

Die Lichtschranke wird zunächst mithilfe eines Entwicklerboards getestet. Dieser Aufbau wird in der folgenden Abbildung 60 dargestellt. Es wird die Lichtschranke gemäß dem Stromlaufplan verbunden. Die Leuchtdiode der Lichtschranke wird mit einem 82 Ohm Widerstand an 3,3 Volt Spannung und an GND verbunden, während die Transistorschaltung der Lichtschranke mit der 12 Volt Spannung, GND und dem Signal-Pin verbunden wird.

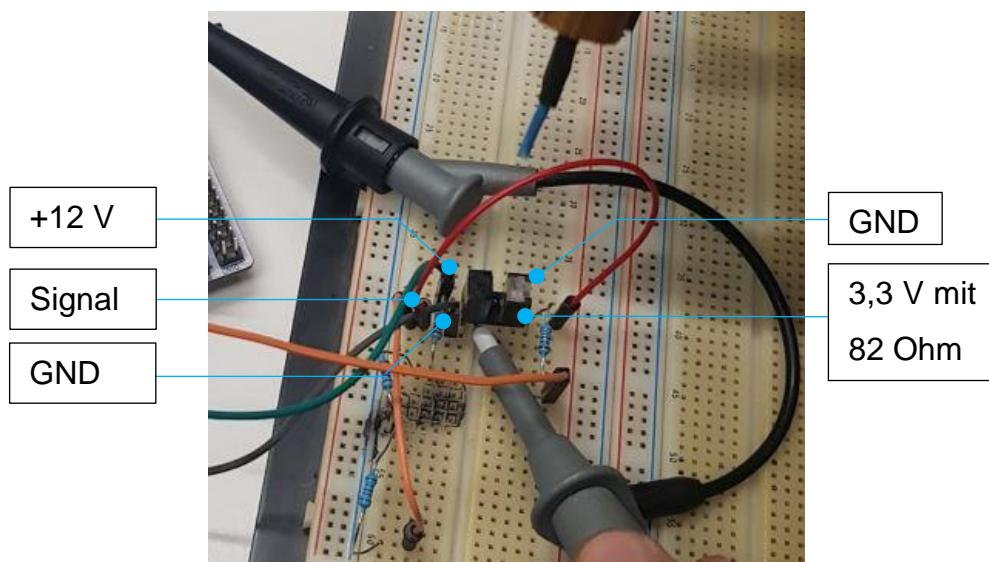


Abbildung 60: Test der Lichtschranke

2.3.2 Auswertung der Testmessungen

Die Lichtschranke funktioniert erwartungsgemäß. Bei nicht eingelegtem Gegenstand zwischen den Gabeln wird am Ausgang der Lichtschranke der Wert 0 Volt gemessen. Beim Unterbrechen des Lichtstrahls wird ein Wert von ca. 2,5 Volt gemessen. Die Lichtschranke enthält zudem einen internen 10 kΩ Widerstand.

2.3.3 Fehleranalyse

Die Lichtschranke funktioniert fehlerfrei. Lediglich zu Beginn des Tests der Lichtschranke kam es zu Fehlern, da der interne Widerstand der Lichtschranke nicht berücksichtigt wurde und die anliegende Spannung am µC nicht verwendet werden konnte.

2.4 Aktorik

Es folgt der Testaufbau der Aktorik. Es wird darauf hingewiesen, dass die eigentliche Programmierung der Aktoren und Funktionen (wie Schrittmotor und Swinger) erst in Kapitel 2.6 endgültig getestet werden.

2.4.1 Testaufbauten zu den Aktoren

Im Folgenden werden Aktoren des 3D-Druckers überprüft.

Test: Swinger mit Funktionsgenerator

In diesem Abschnitt wird der Swinger mithilfe eines Funktionsgenerators geprüft. Zur Prüfung des Schwingers muss ein Gewicht von ca. 17 Gramm eingesetzt werden, um das Gewicht der Laserhülse zu simulieren.

Für den Funktionsgenerator wird die sogenannte Sweep-Funktion des Funktionsgenerators verwendet, bei welcher der Funktionsgenerator ein eingestelltes Frequenzband von 5 bis 15 Hertz durchlaufen soll. Als Beobachter muss daraufhin die Resonanzfrequenz beobachtet werden. Diese ist annähernd erreicht, wenn der Swinger beginnt zu schwingen. Anschließend muss die Frequenz sehr fein eingestellt werden (max. 0,01 Hz Schritte). Außerdem muss bei einer Schwingung darauf geachtet werden, Hebungen des Schwingers durch gezieltes Einstellen der Frequenz weitestgehend zu beseitigen. In der Abbildung 61 ist ein Ausschnitt des Testdurchlaufs zu sehen.

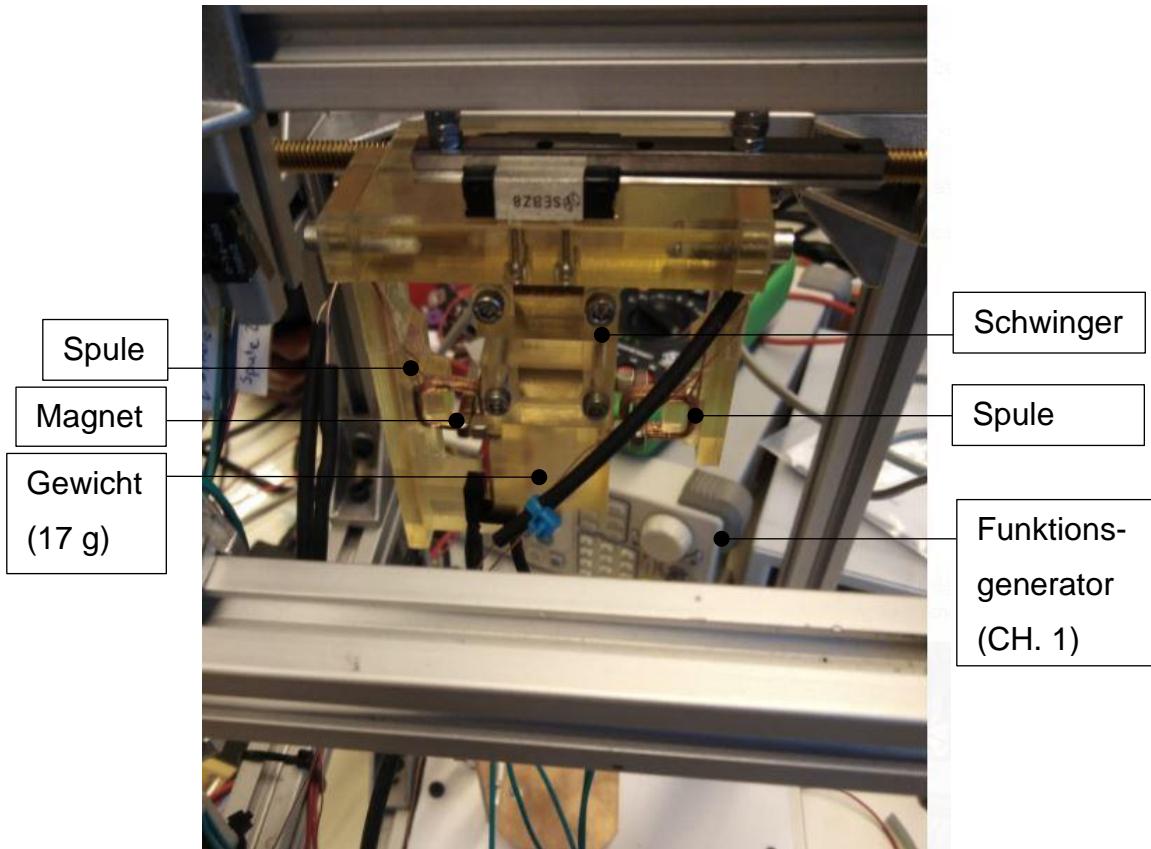


Abbildung 61: Funktionsgenerator mit Schwinger

Test: H-Brücke Funktion

Im nächsten Schritt werden die Ausgänge der H-Brücke getestet. Dabei werden LEDs an je einen Ausgang der H-Brücke geschalten und die H-Brücke als Vollbrücke eingesetzt. Dies soll zeigen, ob die H-Brücke einwandfrei funktioniert und ordnungsgemäß verlötet wurde. Der Test wird in der Abbildung 62 gezeigt und eine schematische Darstellung des Kontaktplans ist in der Abbildung 63 dargestellt.

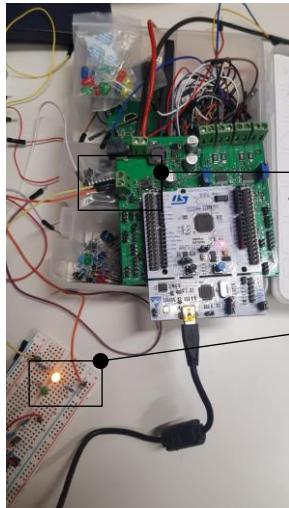


Abbildung 62: Kontrolle H-Brücke

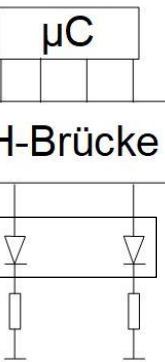


Abbildung 63: Test H-Brücke mit LEDs

Das LEDs blitzen abwechselnd nacheinander (entsprechend dem DIR-Signal). Somit ist der erste Test erfolgreich. Der Code zur Vorbereitung wird im Kapitel Software vorgestellt.

Dann werden die beiden Schrittmotoren getestet. Da diese baugleich sind, wird die Verdrahtung der Motoren gleich angeordnet. Der Kontaktplan für den Test ist in Abbildung 64 dargestellt.

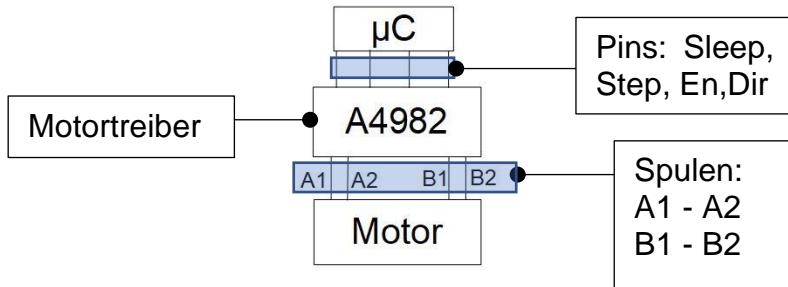


Abbildung 64: Kontaktplan

In der Abbildung 65 wird dieser praktische Aufbau abgebildet. Das Ziel ist den Schrittmotor eine Umdrehung drehen zu lassen. Die Programmierung der Schrittmotoren erfolgt ebenfalls in dem Kapitel Programmierung in 2.6. Anbei wird der praktische Aufbau dargestellt.

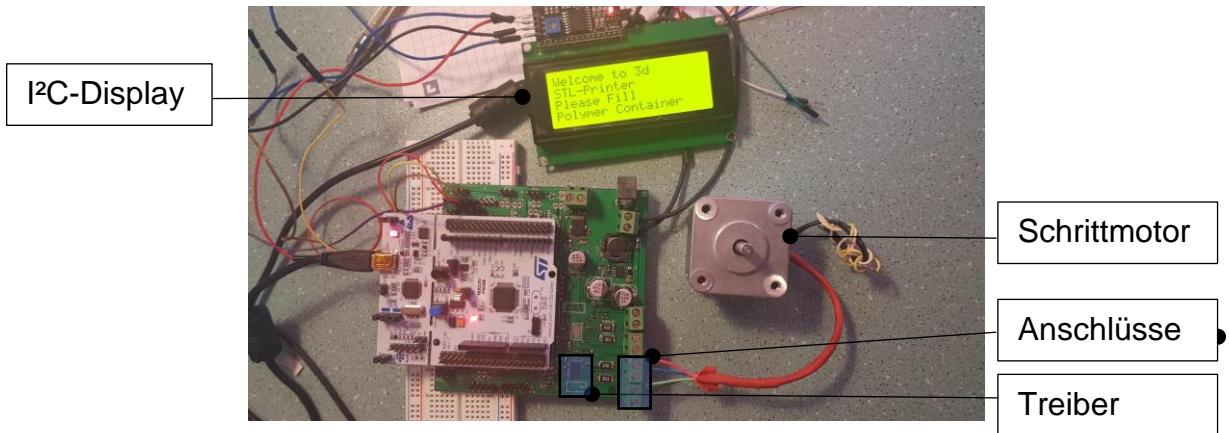


Abbildung 65: Funktionstest Schrittmotor

Der Test ergab ein erfolgreiches Umdrehen der Motorwelle.

2.4.2 Funktionstest PWM bzw. Schrittmusterausgabe

Die Programmierung einer PWM bzw. die Schrittmusterausgabe wird im Kapitel 2.6 detaillierter erläutert. In der Abbildung 66 wird gezeigt, wie die PWM visualisiert wird. Dabei wird eine PWM mit einer Frequenz (PWM = 5 Hz, 50 % DC) erzeugt und das Signal mit LEDs visualisiert. Es ist ein leichtes Pulsieren der LEDs ersichtlich.

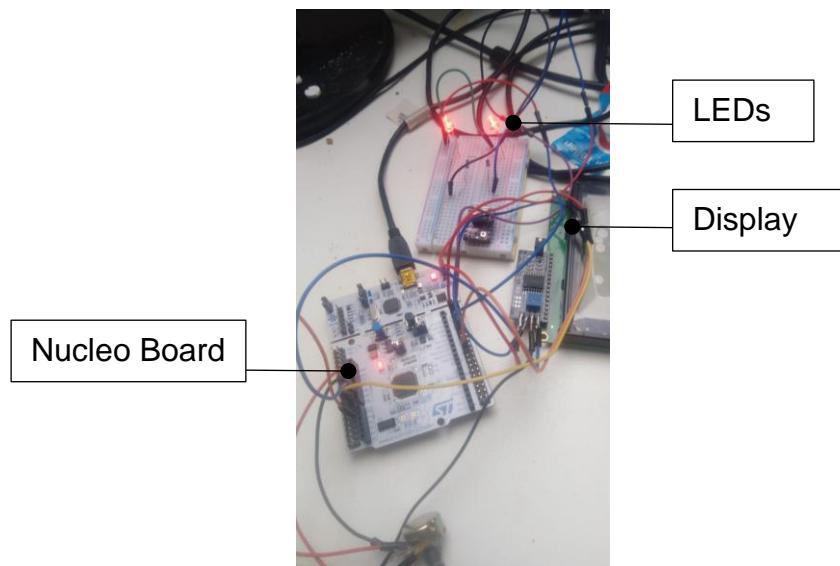


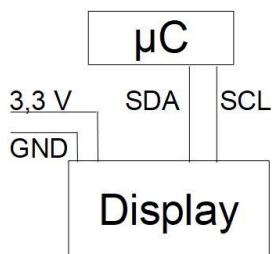
Abbildung 66: PWM-Test

2.4.3 Auswertung der Tests

Die Tests der Spulen mit Funktionsgenerator sowie der Test der Motoren verlief erfolgreich und kann programmiert werden. Nach der Programmierung kann mit der H-Brücke dann der Schwinger betrieben werden und die Schwingung kontrolliert werden.

2.5 Elektronik

Es wird die weitere Elektronik und Elektrik angebracht. Im nächsten Schritt wird zur weiteren Verwendung ein I²C-Display (20x4) mit dem Board verbunden. Dies wird bereits an dieser Stelle frühzeitig in die Steuerung miteingebunden, um beispielsweise Variablen und Messwerte direkt am Display ablesen zu können. Der Kontaktplan wird in der Abbildung 67 gezeigt.



Das Display wird über den I²C-Slot 1 des Nucleo-Boards mit dem SDA- und SCL-Kontakt des Displays verbunden.

Abbildung 67: Kontaktplan Display

Das Display soll verschiedene zu messende Variablen (wie beispielsweise Frequenz) anzeigen und beim Debuggen des Programms helfen (siehe Abbildung 68). Der Programmcode wird im Kapitel 2.6 dargestellt.



Abbildung 68: Implementierung I²C-Display

2.5.1 Beschreibung Layout

Das Layout wurde wie in der Abbildung 45 dargestellt gefertigt und gelötet. Das gelötete Board ist in der Abbildung 69 dargestellt. Es werden die verschiedenen Spannungen (3,3; 5 und 12 Volt) getestet. Die Messpunkte der Spannungen werden in der Abbildung als P3,3 (für 3,3 Volt); P5 (5 V) und P12 (12 V) bezeichnet.

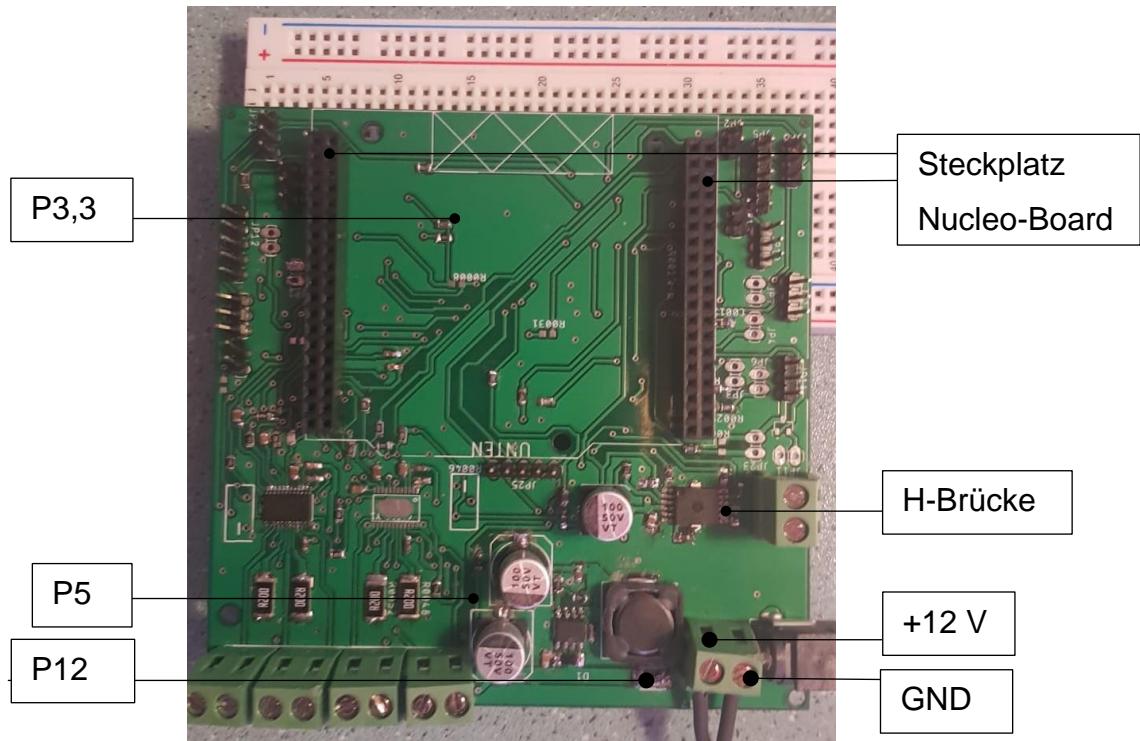
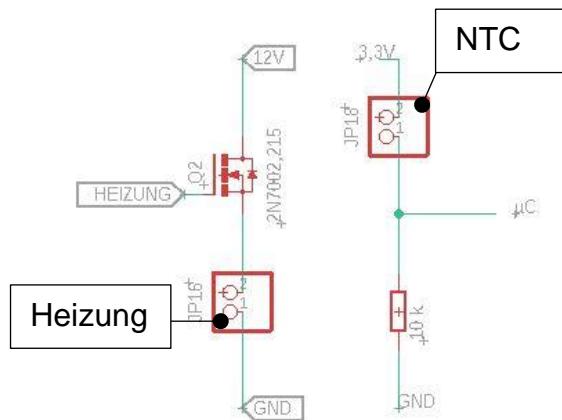


Abbildung 69: Verlötetes Board bereit zum Test

Temperatur- und Heizstromkreis



Zusätzlich wird der Heizstromkreis (mit extra Temperaturmessung) festgelegt (siehe Abbildung 70). Die Heizung wird über einen Transistor (durch den μ C) durchgeschalten und die Heizung an Spannung gelegt. Der Temperaturmesskreis besteht aus einem NTC und einem $10\text{ k}\Omega$ Spannungsteiler.

Abbildung 70: Links: Stromkreis Heizung und rechts: Temperaturmessung

2.5.2 Bauteilliste im Anhang

Die Bauteilliste befindet sich im Anhang.

2.5.3 Inbetriebnahmeplanung

Die Inbetriebnahme der einzelnen Komponenten erfolgte bzw. erfolgt nacheinander. Das bedeutet, dass jeweils eine Baugruppe komplett getestet, geprüft und in Betrieb genommen wird.

Wie bereits erwähnt, sind Funktion der Lichtschranke, Endschalter und Schwinger (mit Frequenzgenerator) nacheinander getestet worden, um einen Fehler auf der Platine, bzw. einen Fehler beim Verlöten ausschließen zu können.

Eine dementsprechende Inbetriebnahme der restlichen Baugruppen, bzw. deren Zusammenspiel wird anfangs im Kapitel 2.6 durchgeführt, in welchem der Programmcode für alle Bauelemente beschrieben wird. Anschließend wird in Kapitel 3 die Eingliederung der Baugruppen und die Erstellung des Main-Programms mit Interrupts und Unterfunktionen und die Gesamtfunktion durchgeführt. Die weitere Inbetriebnahmeplanung (siehe Tabelle 14) sieht dabei folgende Baugruppen mit Beschreibung und deren Aufgaben vor.

Tabelle 14: Inbetriebnahmeplanung

Pos.	Baugruppe	Beschreibung	Aufgaben
1	Schritt-motoren	Festlegen Richtung Referenzfahrt	Richtung für Fahrt zum Endschalter bestimmen;(Test erst nach verdrahten Endschalter)
		Motorstrom einstellen	Einstellen des Motorstroms (genügend Drehmoment, wenig Erhitzung)
		Schrittzahl definieren	Bestimmen der Anzahl von Schritten für jeweils eine y- und z-Koordinate
2	Elektrik	Elektrik installieren	Dazu zählen Spulen an H-Brücke anschließen, Platine befestigen, Endschalter und Motoren anschließen
3	Schwinger	Programm/ Ablauf zur Erfassung für Frequenz bzw. T _{Hell}	Es wird ein Programm/ Ablauf angefertigt, dass mithilfe der Lichtschranke die Periodendauer

		und T_{Dunkel} schreiben	(bzw. Frequenz) einer Schwingung angeben kann
		Kontrolle des Programms durch Funktionsgenerator	Es wird der Ausgang des Funktionsgenerators an den vorgesehenen Signal-Pin des µC (zur Erfassung der Lichtschranke) angeschlossen. Kontrolle der Frequenz
		Lichtschranke einkleben	Lichtschranke wird so mittig wie möglich eingeklebt.
		Laserhülse einsetzen und f_{soll} erkennen	Es wird das Gewicht des späteren Lasers benötigt, um die ungefähre Eigenfrequenz zu ermitteln
4	H-Brücke	Anregungsart festlegen	Gewünschtes Verfahren: Sinusanregung. Programm dafür in Kapitel 2.6 Programmierung erstellen
		Kontrolle des Programms	Kontrolle des Programms am Ausgang der H-Brücke. Dabei ist kein „typischer“ Sinus zu erwarten, sondern veränderter DC-Wert
5	H-Brücke und Schwinger	Amplitudenregelung	Amplitudenregelung programmieren
		falls benötigt: PLL	PLL wird nicht fest vorgesehen, sondern wenn die Frequenz nicht gut geregelt werden kann
Meilenstein 1: An dieser Stelle sind die wichtigsten Teile des Druckers aufgebaut, der rote Laser ist eingesetzt und die Regelung ist vorhanden. Das Vervollständigen der Regelung wird als Meilenstein deklariert. Ohne diesen Meilenstein kann nicht weiter am roten Kreis gearbeitet werden.			
6	Roter Kreis	Es wird versucht, einen roten Kreis auf dem Druckbett darzustellen	Schreiben eines Ablaufs und Programmcodes zum Erstellen des roten Kreises

		Prüfen roter Kreis	Die dargestellten Punkte werden mit Hilfe der Langzeitbelichtung geprüft
7	Roter Kreis und Höhe	Es wird die z-Achse in den roten Kreis mit eingebunden	Es werden weitere Schichten (z-Achse) programmiert. Nach jeder gelaserten Schicht fährt die z-Achse eine Schicht hinab und lasert weiter
Meilenstein 2: Der rote Kreis kann projiziert und die einzelnen Schichthöhen verfahren werden. Programm zum Eintauchen des Druckbetts, Anfahren der Endschalter, etc. funktioniert.			
8	Mechanik Vorbereit. für Drucktest	Restliche Mechanik anbringen	Es wird die restliche benötigte Mechanik, wie Deckel und Gehäuse angebracht sowie der rote Laser entfernt
9	Probe-druck	Ersten Proberuck anfertigen	Es wird der blaue Laser eingebaut, die Sollfrequenz ermittelt, eingestellt, die Dämpfung geprüft, verkabelt etc. Vorbereitungen für Druck sind erledigt
Meilenstein 3: Erstes Druckteil angefertigt			
10	Druck-qualität	Kontrolle Druckqualität	Überprüfung der Belichtungszeit pro Voxel, der Höhe, der Qualität und Form des Drucks
11	Opti-mierung	Mögliche Fehlerquellen beseitigen	Prüfen auf Rundheit, Maßhaltigkeit, mögliche Anpassungen in Programm und Hardware
12	Kreis-fenster	Hohlraum Zylinder in	Es wird der Zylinder mit Kreisfenster entworfen und mit dem roten Laser überprüft.
Der Zylinder mit Kreisfenster wurde gedruckt und ist in einer zufriedenstellenden Qualität			

2.6 Test Treibersoftware

In diesem Kapitel wird die Software zur Ausarbeitung eines Main-Programms erläutert.

2.6.1 Test von notwendigen Funktionen vom STM32 Nucleo Board

In diesem Kapitel werden die Programme zu den jeweiligen Baugruppen beschrieben.

Anhand der dargestellten Pinbelegung in Abbildung 71 und Abbildung 72 können die entsprechenden Pins verbunden werden.

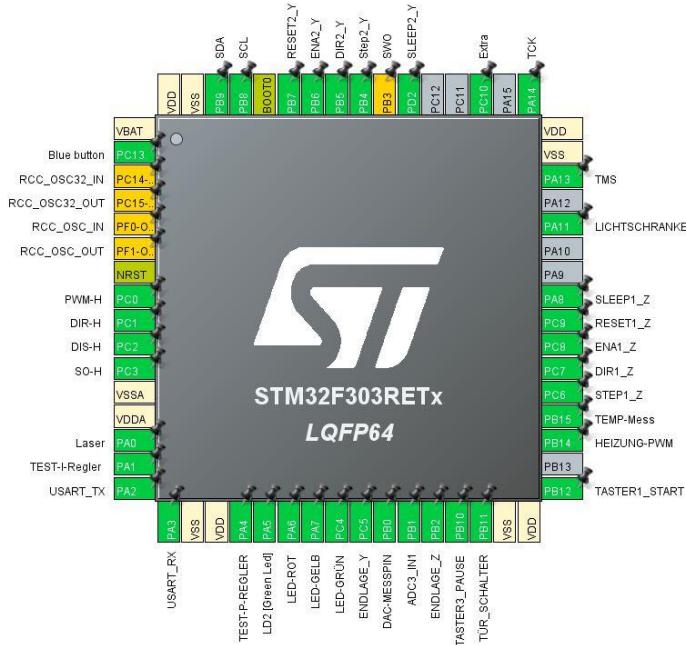


Abbildung 71: STM32 Cube MX Pinbelegung

Pinbelegung NUCLEO-F303RE

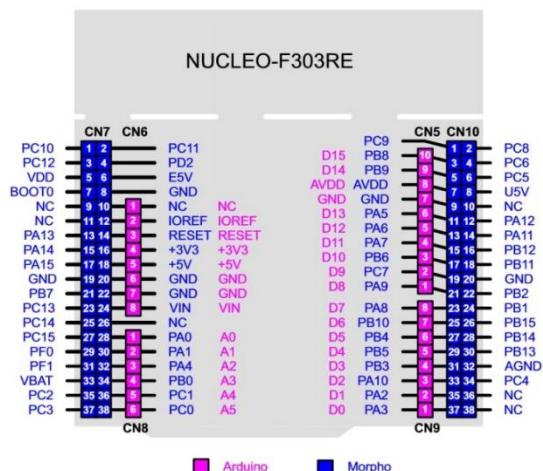


Abbildung 72: STM32 Hardware Pinbelegung [11]

2.6.2 Verbindung Display zu Board über I2C

Die I²C-Verbindung für das Display hat zwei Pins, SDA und SCL. Die beiden Pins SDA und SCL entsprechen hierbei den Pins PB9 (SDA) und PB8 (SCL) am Nucleo-Board. Ein Teil des Codes zur Ansteuerung des Displays wird im Weiteren dargestellt. Zunächst wird die I²C-Adresse des Displays ermittelt.

Die Displayadresse kann mit einer „for“ Schleife ermittelt werden:

```
//1. Scan the I2C addresses
for(uint8_t i=0; i<255; i++)
{
    if(HAL_I2C_IsDeviceReady(&myI2Chandle, i, 1, 10) == HAL_OK)
    {
        HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_12);
        break;
    }
}
```

Nachdem die Adresse ermittelt wurde, kann nun eine Bibliothek von einem ähnlichen Display im Format „20x04“ von der Website „Github“ gesucht und heruntergeladen werden. Dann muss die Bibliothek ins Projekt eingebunden und angepasst werden. Mit dem Befehl „HAL_I2C_Master_Transmit (&hi2c1, SLAVE_ADDRESS_LCD,(uint8_t *) data_t, 4, 100);“ kann die Übertragung via I²C erfolgen. Das ist in der Abbildung 73 zu sehen.

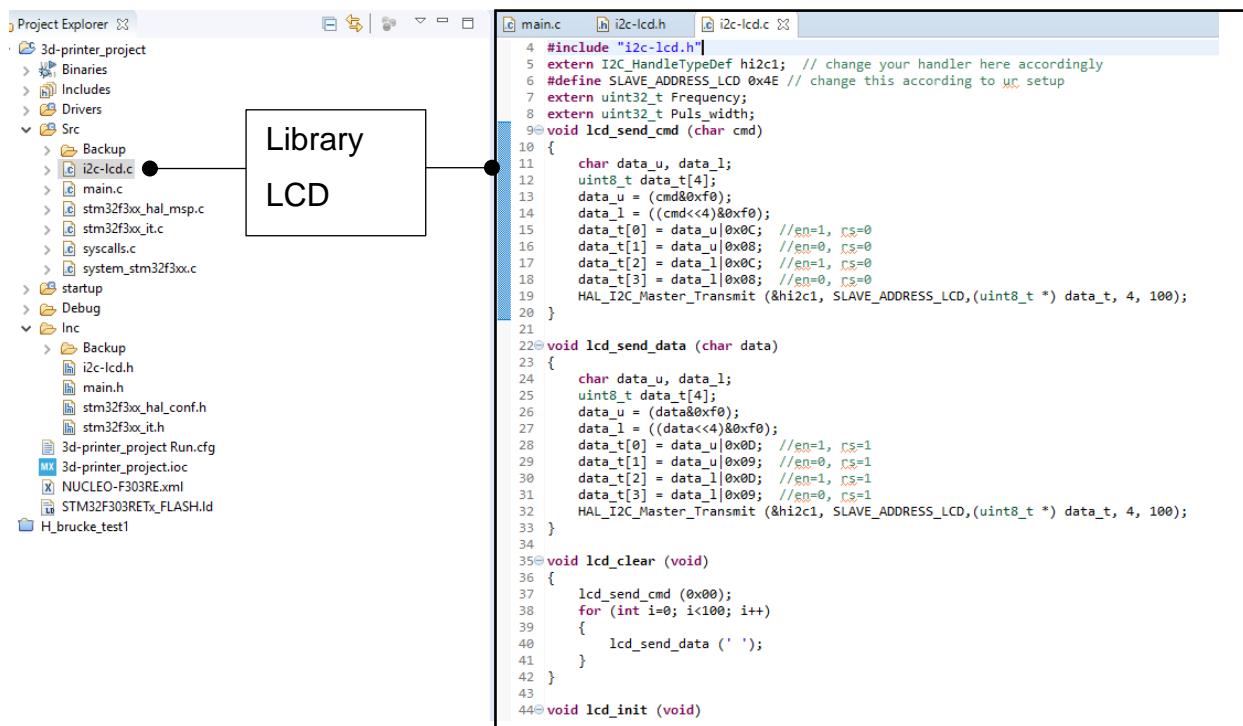
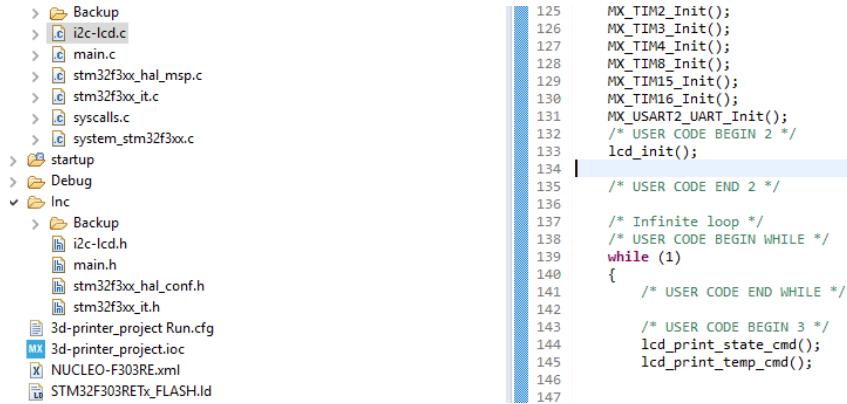


Abbildung 73: I²C_LCD (siehe auch [12])

Zuerst wird die Header-Datei „i2c-lcd.h“ im Hauptprogramm main.c inkludiert. Dann kann die Funktion in der main.c aufgerufen werden. Mit dem Befehl „lcd_init ();“ wird das LCD initialisiert. Dies wird in der Abbildung 74 gezeigt.



```

> Backup
> i2c-lcd.c
> main.c
> stm32f3xx_hal_msp.c
> stm32f3xx_it.c
> syscalls.c
> system_stm32f3xx.c
> startup
> Debug
> Inc
> Backup
  i2c-lcd.h
  main.h
  stm32f3xx_hal_conf.h
  stm32f3xx_it.h
  3d-printer_project.Run.cfg
MX 3d-printer_project.ioc
X NUCLEO-F303RE.xml
STM32F303RETx_FLASH.Id

```

```

125     MX_TIM2_Init();
126     MX_TIM3_Init();
127     MX_TIM4_Init();
128     MX_TIM8_Init();
129     MX_TIM15_Init();
130     MX_TIM16_Init();
131     MX_USART2_UART_Init();
132     /* USER CODE BEGIN 2 */
133     lcd_init();
134   | /* USER CODE END 2 */
135
136   /* Infinite loop */
137   /* USER CODE BEGIN WHILE */
138   while (1)
139   {
140     /* USER CODE END WHILE */
141
142     /* USER CODE BEGIN 3 */
143     lcd_print_state_cmd();
144     lcd_print_temp_cmd();
145
146
147

```

Abbildung 74: i2c _im Main-Programm

Mit dem Aufruf der beiden Funktionen in der „While“-Schleife des Main-Programms sind folgende Resultate in Abbildung 75 (Startbildschirm für Einsticken des Druckers) sowie in Abbildung 76 (Menüauswahl) ersichtlich.

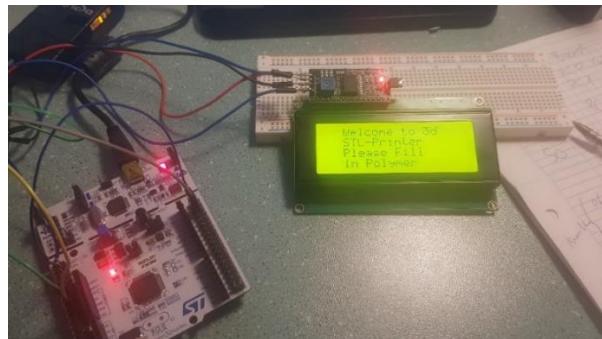


Abbildung 75: LCD-Ausgabe: Startbildschirm

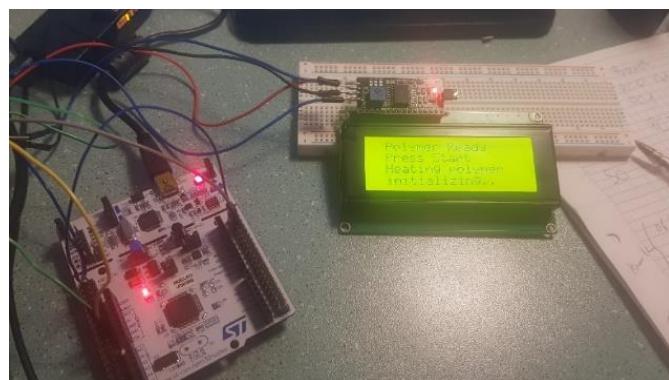


Abbildung 76: LCD-Ausgabe Menüauswahl

2.6.3 Ausgabe auf eine Konsole auf dem Bildschirm durch „Printf“:

Zum Ausgeben von Ereignissen, Auswerten oder auch zum Debuggen kann zudem eine Konsole bzw. ein Serieller Monitor und eine Arduino Konsole verwendet werden. Um Überschreibungen von Printf über UART zu ermöglichen, muss zuerst eine Bibliothek erstellt werden. Durch folgende Funktion (siehe auch Abbildung 77) ist eine serielle Übertragung möglich:

- Baudrate 9600 Bit/s, 8bit polarity
- „HAL_UART_Transmit(&huart2, ptr, len, HAL_MAX_DELAY);“

```
main.c | print_scan_usart2.h |
2⊕ * print_scan_usart2.h
7
8 #include "stm32f3xx_hal.h"
9 extern UART_HandleTypeDef huart2;
10
11 #ifndef PRINT_SCAN_USART2_H_
12 #define PRINT_SCAN_USART2_H_
13
14 int _write(int32_t file, uint8_t *ptr, int32_t len)
15 {
16     HAL_UART_Transmit(&huart2, ptr, len, HAL_MAX_DELAY);
17     return len;
18 }
19 int _read(int32_t file, uint8_t *ptr, int32_t len)
20 {
21     HAL_UART_Transmit(&huart2, ptr, len, HAL_MAX_DELAY);
22     return len;
23 }
```

Abbildung 77: Printf_Bibliothek

Die Funktion scanf zum Übertragen von User-Eingaben ist hier zusätzlich eingefügt, aber für das Projekt nicht notwendig. Die Ausgabe von Float- und Long-Variablen ist in der Einstellung zu ändern. Nach Einbindung der Bibliothek in die Main.c kann folgendes Resultat in der Abbildung 78 und Abbildung 79 gesehen werden.

```
133 /* USER CODE BEGIN 2 */
134 printf("Welcome to 3d\nSTL-Printer\nPlease Fill\nin Polymer\n");
135
136 /* USER CODE END 2 */
137
138 /* Infinite loop */
139 /* USER CODE BEGIN WHILE */
140 while (1)
141 {
142     /* USER CODE END WHILE */
```

Abbildung 78:: Uart mit Main-Funktion und Funktion printf

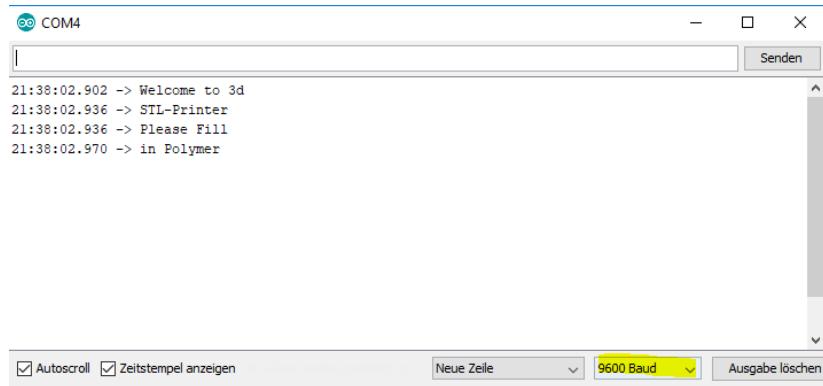


Abbildung 79: *Printf_Ausgabe*

2.6.4 ADC Single Channel Test mit Thermistor und NPN Transistor:

Der ADC (Verwendung bei der Temperaturmessung) wird mit einem Interrupt konfiguriert, sodass nach 10 Sekunden ein Interrupt ausgelöst und die Temperatur abgefragt wird. Wenn die Temperatur unter der Solltemperatur liegt, wird ein Signal an den Transistor gesendet und die im Test verwendete LED leuchtet. (Im Test wurde statt einem Transistor eine LED am Gate-Pin des Transistors verwendet). Dieser Test wurde mit privaten Bauteilen realisiert und lässt sich auf das analoge Prinzip zu der eigentlichen Heizung übertragen. In der Abbildung 80 ist die fertige Schaltung erkennbar.

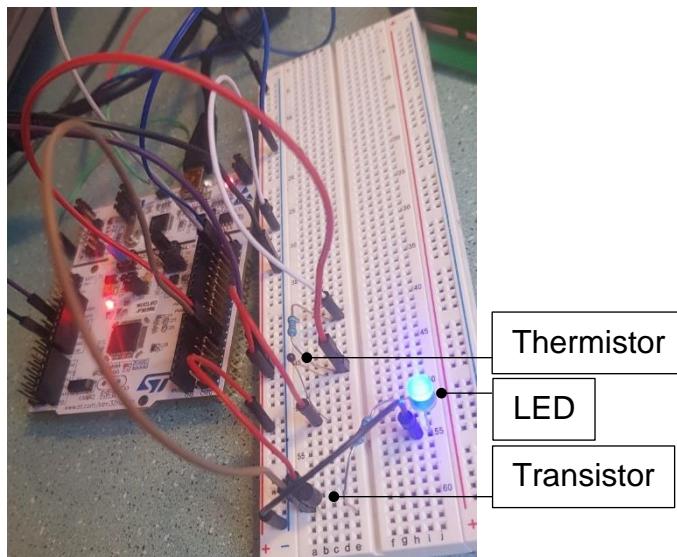


Abbildung 80 Schaltung Thermistor

Die Ausgabe der Messwerte erfolgt über das LCD (siehe Abbildung 81), Arduino Konsole und anhand des Programms STM-Studio.



Abbildung 81: Ausgabe der aktuellen Temperatur am Display

```

COM4
14:24:32.440 -> Temperature=24.11
14:24:42.410 -> Temperature=24.02
14:24:52.420 -> Temperature=24.07

```

Abbildung 82: Ausgabe der Temperatur an Konsole

Die Abbildung 82 zeigt, dass die Ausgabe alle 10 Sekunden stattfindet und der Timer entsprechend nach Planung programmiert wurde. Als nächstes ist die Ausgabe der Temperatur in STM-Studio dargestellt (siehe Abbildung 83).

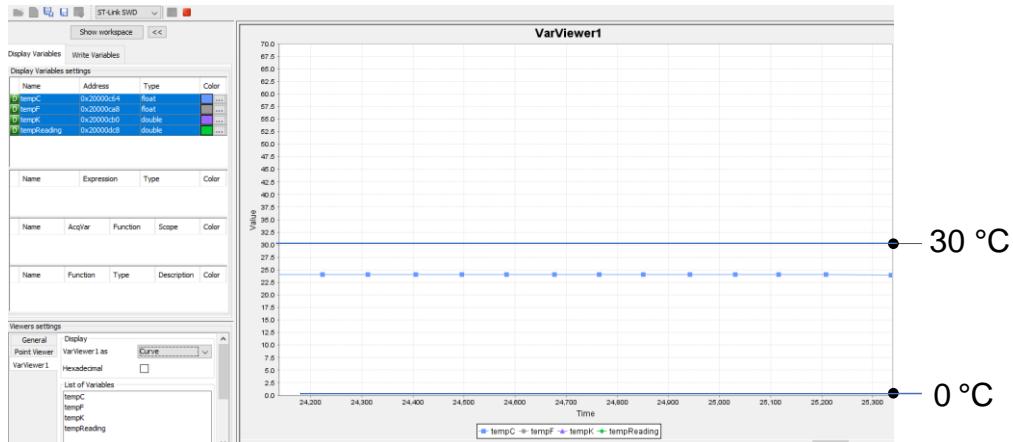


Abbildung 83: Ausgabe Temperatur STM-Studio

Die im Test verwendete LED leuchtet, da die Temperatur unter 60°C ist und die Steuerung den Transistor bestromen würde. Nun wird die Temperatur am NTC durch ein Wärmemittel über 60 °C gebracht, um zu prüfen, ob das Programm richtig

funktioniert, d.h. ob die LED dann nicht mehr leuchtet. Dies ist in Abbildung 84, Abbildung 85 und Abbildung 86 dargestellt.

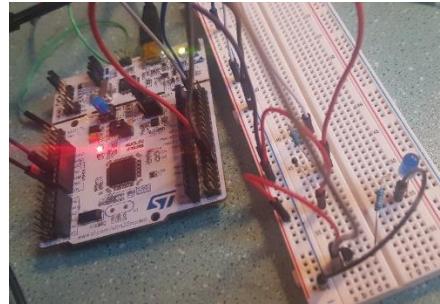


Abbildung 84: Ausgabe Temperatur über 60°C

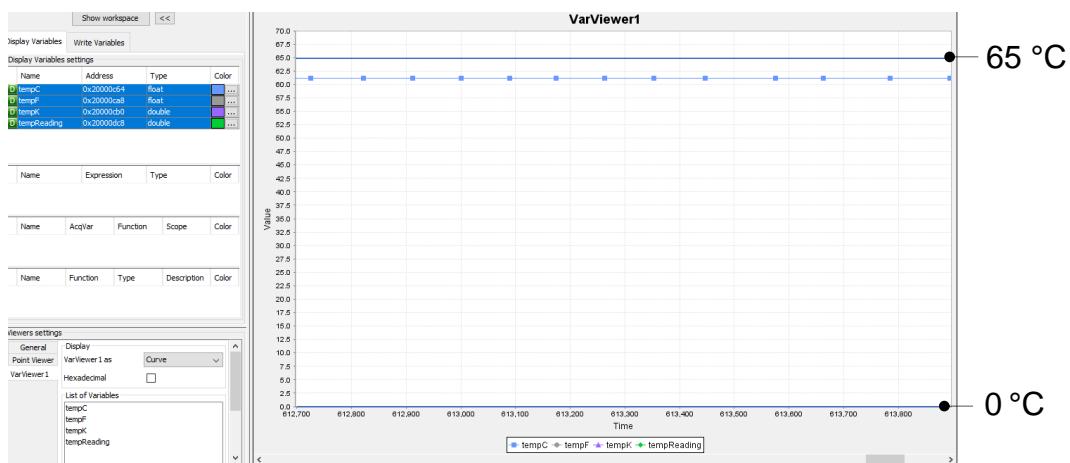


Abbildung 85: Ausgabe Temperatur über 60 °C STM Studio

```
COM4
14:48:42.279 -> Temperature=63.75
14:48:52.260 -> Temperature=62.32
14:49:02.273 -> Temperature=57.29
```

Abbildung 86: Ausgabe Temperatur über 60 °C Konsole

Die oben gezeigten Bilder beweisen, dass das Programm einwandfrei funktioniert. Mit dem gleichen Prinzip kann die Heizung geregelt werden. Der Code zur Regelung der Heizung ist in der Abbildung 87 dargestellt.

```

/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *ht)
{
    if (ht == &htim7) {
        HAL_ADC_PollForConversion(&hadc4, 1); // poll for conversion
        tempReading = HAL_ADC_GetValue(&hadc4); // get the adc value
        tempK = log(10000.0 * ((4096.0 / tempReading - 1))); //Specification of thermistor Modell
        tempK = 1 / (0.001129148 + (0.000234125 + (0.000000876741 * tempK * tempK)) * tempK);
        tempC = tempK - 273.15;
        tempF = (tempC * 9.0) / 5.0 + 32.0;
        printf("\nTemperature=%f", tempC); //Konsole Ausgabe
        Temperature_print(); //LCD Ausgabe
        if (tempC<60){ •
            HAL_TIM_PWM_Start(&htim15, TIM_CHANNEL_1);
            HAL_TIM_SET_COMPARE(&htim15, TIM_CHANNEL_1, 200); // 100% duty cycle, voll Spannung
        }
        else{ •
            HAL_TIM_SET_COMPARE(&htim15, TIM_CHANNEL_1, 0); // 0% duty cycle, 0 spannung
            HAL_TIM_PWM_Stop(&htim15, TIM_CHANNEL_1);
        }
    }
}

```

Formel zur Berechnung

Heizung an

Heizung aus

Abbildung 87: Programm Temperatur (Link aus [13])

Für das Projekt kann diese Art von Thermistor verwendet werden. Es muss dabei das Programm auf die Spezifikation vom verwendeten Modell angepasst werden. Im nächsten Schritt wird das Programm über eine Bibliothek aufgerufen, um die Lesbarkeit und Optimierung in der Main.c zu ermöglichen.

Es wird der im µC verwendete ADC des Temperatursensors mit dem Befehl „*Polloffconversion*“ aufgerufen. „Polling“ steht für blockierende Funktionen. Schneller wäre innerhalb des ADC ein Interrupt auszulösen mit dem folgenden Befehl:

„*void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef *hadc)*“.

Es gibt zudem eine weitere mögliche Variante. Durch den DMA „*Direct Memory Access*“ können viele Daten gespeichert werden, ohne die CPU auszubremsen. DMA ist aber in unserem Projekt nicht von großer Bedeutung, da keine großen Mengen an Daten gleichzeitig übertragen werden müssen. Zur Filterung der Messwerte könnte falls notwendig entweder ein FIR- oder IIR-Filter angewendet werden. Dieser ist aktuell aber nicht vorgesehen. Die Regelung der Polymer-Temperatur des 3D-Druckers ist aufgrund der hohen Trägheit der Flüssigkeit mit einem Zweipunkt-Regler voreingestellt.

2.6.5 Funktionstest der Lichtschranke durch „Input Capture“ mit Timer

Die Lichtschranke hat 5 Anschlüsse; zwei GND, ein 3.3 und ein 12 Volt sowie ein Signal-Pin. Um das Ausgangssignal zu testen, wird die Lichtschranke in der Abbildung 88 (wie in Abbildung 60) verschaltet.

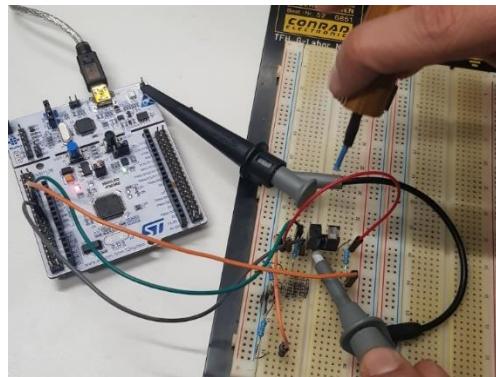


Abbildung 88: Schaltung Lichtschranke

Ein Spannungsteiler (intern 10 kOhm) und extern 3,9 kOhm sichert den Übergang von 12 Volt des Signalausgangs zum maximal 3,3 Volt Eingangssignal des µC. Ein Vorwiderstand schützt zudem die Leuchtdiode vor zu hohen Strömen. Das Funktionsprinzip einer Lichtschranke erklärt sich dadurch, dass im normalen Zustand bei keiner Unterbrechung des Lichtsignals ein High-Signal am Signal-Pin gemessen wird. Sobald eine Unterbrechung des Lichtübergangs stattfindet, wird ein Low-Signal gemessen. Dies wird in der Abbildung 89 verglichen.

Links ist dabei ein High-Signal zu sehen (Zustand: kein Objekt in Lichtschranke). Rechts hingegen ist das Low-Signal zu sehen, also ist die Lichtschranke unterbrochen.

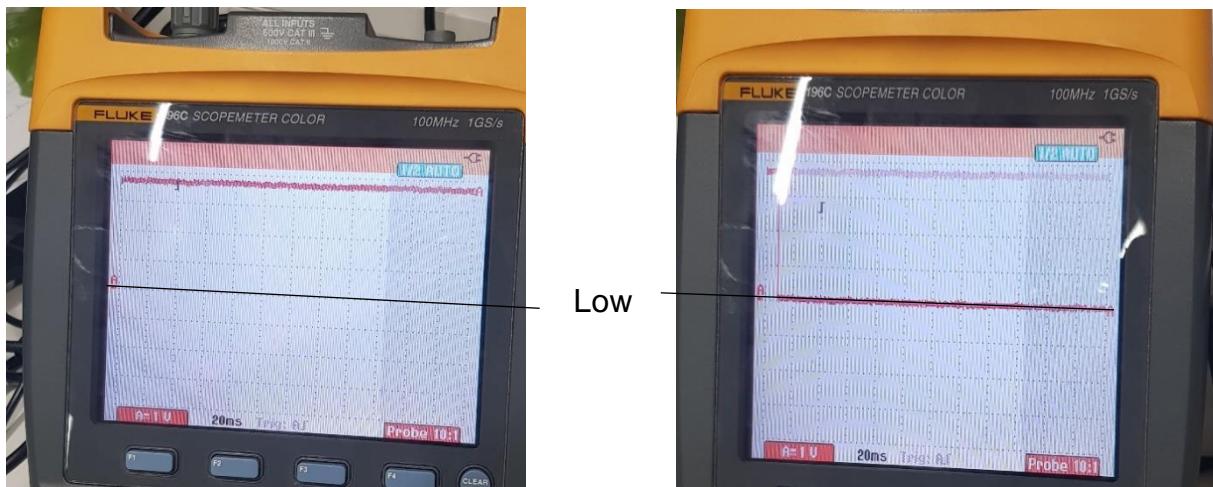


Abbildung 89: Links High-Signal und rechts Low-Signal der Lichtschranke

Nach dem elektronischen Test der Lichtschranke kann das Programm mittels STM32 ausgeführt werden. Hier wird ein Timer mit einem maximalen ARR Wert (0xffff) ausgewählt, um die Capture des Channel 1 und 2 zu speichern und die beiden

anschließend zu vergleichen. Hier ist zu beachten, dass die messbare Frequenz vom Prescaler-Wert abhängig ist. Die F_{clk} bzw. die Clock Frequenz ist: $\frac{72 \cdot 10^6 \text{Hz}}{720} = 10^5 \text{Hz}$

Die kleinste Frequenz ist durch die Division von 10^5 Hz durch 0xffff und ergibt 1,52 Hz. Da unser System wie zuvor berechnet mit ca. 8,3 Hz schwingt, kann die Einstellung für die Input Capture verwendet werden. Mit dem Befehl „HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_1);“ startet die Input Capture den Channel 1 mit einem Interrupt und mit „HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_2);“ startet die Input Capture den Channel 2 mit einem Interrupt.

Innerhalb der Callback Funktion des Input Capture sieht der Code folgendermaßen aus (siehe Abbildung 90).

```
/* USER CODE BEGIN 4 */
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1) // if interrupt source is channel 1
    {
        if (Is_First_Captured==0) // is the first value captured ?
        {
            IC_Value1 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1); // capture the first value
            Is_First_Captured =1; // set the first value captured as true
        }

        else if (Is_First_Captured==1) // if the first is captured
        {
            IC_Value2 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1); // capture second value

            if (IC_Value2 > IC_Value1)
            {
                Period = ((IC_Value2-IC_Value1)*(0.00001)); // calculate the Period
            }

            else if (IC_Value2 < IC_Value1)
            {
                Period = (((0xffff-IC_Value1)+IC_Value2) +1)*(0.00001));
            }

            else
            {
                Error_Handler();
            }
        Frequency = (1/Period);
    }
}

```

Abbildung 90: Input Capture Callback Funktion (siehe auch [14])

Zuerst werden die Captures (Werte) im Falling-Mode jeweils in Variable IC_Value1 und IC_Value2 gespeichert. Die ganze Periode ist dann die Differenz zwischen zwei nacheinander ablaufenden Falling Edge Capture Values. Da der Timer mit 10^5 Hz zählt, muss die Differenz mit 0.00001 multipliziert werden, um die Periode in Sekunden anzuzeigen. Falls der Timer während des Zählens überläuft, kann es sein, dass der IC_Value_2 einen kleineren Wert als IC_Value1 hat. Deswegen schreibt man oft die Form $((0xffff - IC_Value1) + IC_Value2 + 1)$, sodass die Periode immer positiv bleibt.

Die Frequenz ist somit die Inverse der Periode. Gleiches gilt für die Rising Edge. Im Folgenden wird der Code für die Periode 2, die Pulsbreite und den Duty Cycle dargestellt (siehe Abbildung 91).

```

else if (Is_First_Captured_2==1) // if the first is captured
{
    IC_Value2_2 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_2); // capture second value

    if (IC_Value2_2 > IC_Value1_2)
    {
        Period_2 = ((IC_Value2_2-IC_Value1_2)*(0.00001)); // calculate the Period
    }

    else if (IC_Value2_2 < IC_Value1_2)
    {
        Period_2 = (((0xffff-IC_Value1_2)+IC_Value2_2) +1)*(0.00001));
    }

    else
    {
        Error_Handler();
    }
    Frequency_2 = (1/Period_2);
    //Frequency = HAL_RCC_GetPCLK2Freq()/(Period*htim4.Init.Prescaler); // calculate frequency
    Is_First_Captured_2 = 0; // reset the first captured
}

if (IC_Value2 > IC_Value1 && IC_Value2_2 > IC_Value1_2)
{
    Pulswidth = (IC_Value2_2 - IC_Value2)*(0.00001);
    Duty_Cycle = (Pulswidth/((Period+Period_2)*0.5))*100;
}

```

Abbildung 91: Periode, Pulsbreite und Duty Cycle

Die Pulsbreite ist die Differenz zwischen einer fallenden und einer steigenden Flanke; Um den Duty Cycle in % zu zeigen, multipliziert man die Gleichung mit 100. Um den Code einfach zu testen, schließt man eine Verbindung zwischen dem Input Capture Pin und einer Signalquelle (PWM Pin, Wave Generator) mit einer bekannten Frequenz. Die eigene verwendete Frequenz von ca. 8.3 Hz wird mittels PWM-Pin und Wave Generator generiert und anhand der Input Capture gelesen. Dies wird durchgeführt, um die Richtigkeit der Messung zu validieren Das Ergebnis ist in Abbildung 92 und Abbildung 93 dargestellt und ist ideal.

VarViewer1			
Variable Name	Address/Expression	Read Value	
Frequency	0x2000046c	8.299444	
Frequency_2	0x20000480	8.300133	
IC_Value1	0x20000460	38034.0	
IC_Value1_2	0x20000474	56107.0	
IC_Value2	0x20000464	50083.0	
IC_Value2_2	0x20000478	44058.0	
Period	0x20000468	0.12049	
Period_2	0x2000047c	0.12048	
Pulswidth	0x20000484	0.06025	
Duty_Cycle	0x20000498	50.006726	

Abbildung 92: Ergebnis Input Capture mit PWM-Generierung

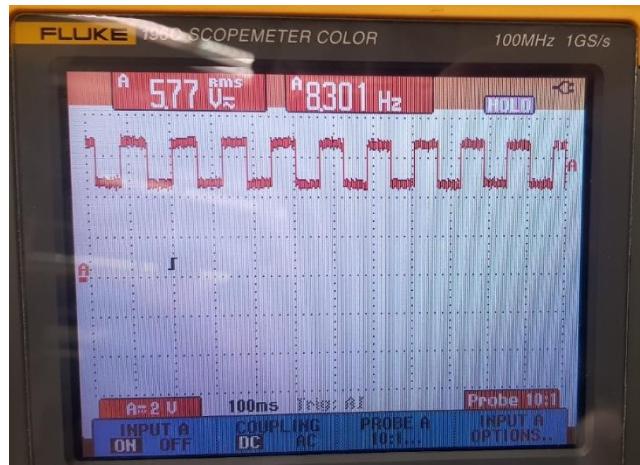


Abbildung 93: Ausgabe der erzeugten Frequenz am Oszilloskop

In der Abbildung 94 und Abbildung 95 ist das Ergebnis mit einer langsameren Schwingung durch die Lichtschranke zu sehen.

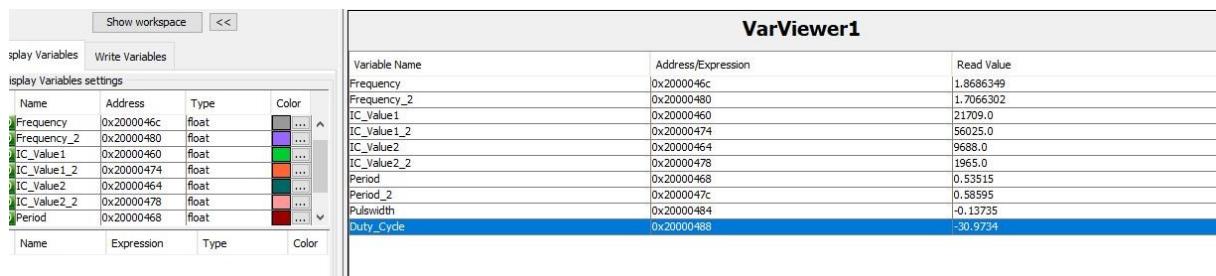


Abbildung 94: Frequenz-Messung bei einer beliebigen Schwingung mit Werten

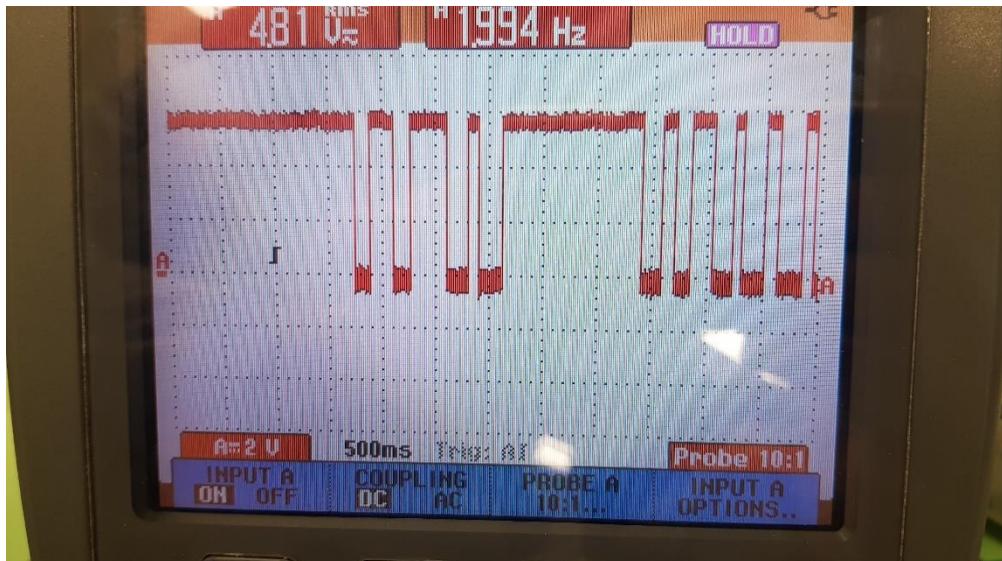
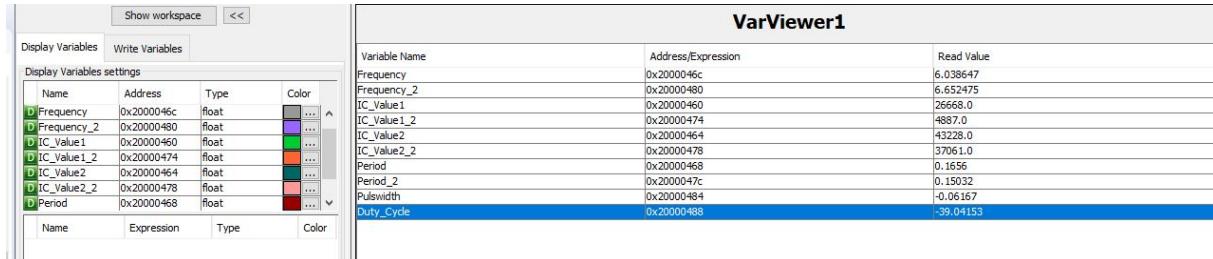


Abbildung 95: Darstellung Frequenz am Oszilloskop bei langsamer Schwingung

Bei einer beliebig schnelleren Schwingung ergibt sich eine höhere Frequenz. Die Abbildung 96 und die Abbildung 97 zeigen das Ergebnis einer schnelleren Schwingung.



The screenshot shows the VarViewer1 software interface. On the left, there is a tree view of variables under 'Display Variables' and a table for 'Display Variables settings'. The table has columns for Name, Address, Type, and Color. On the right, there is a table titled 'VarViewer1' with columns for Variable Name, Address/Expression, and Read Value. The table lists several variables related to Frequency and Period.

Variable Name	Address/Expression	Read Value
Frequency	0x2000046c	6.038647
Frequency_2	0x20000480	6.652475
IC_Value1	0x20000460	26668.0
IC_Value1_2	0x20000474	4887.0
IC_Value2	0x20000464	43228.0
IC_Value2_2	0x20000478	37061.0
Period	0x20000468	0.1656
Period_2	0x2000047c	0.15032
PulseWidth	0x20000484	-0.06167
Duty_Cycle	0x20000488	-39.04153

Abbildung 96: Frequenz schnellere Schwingung

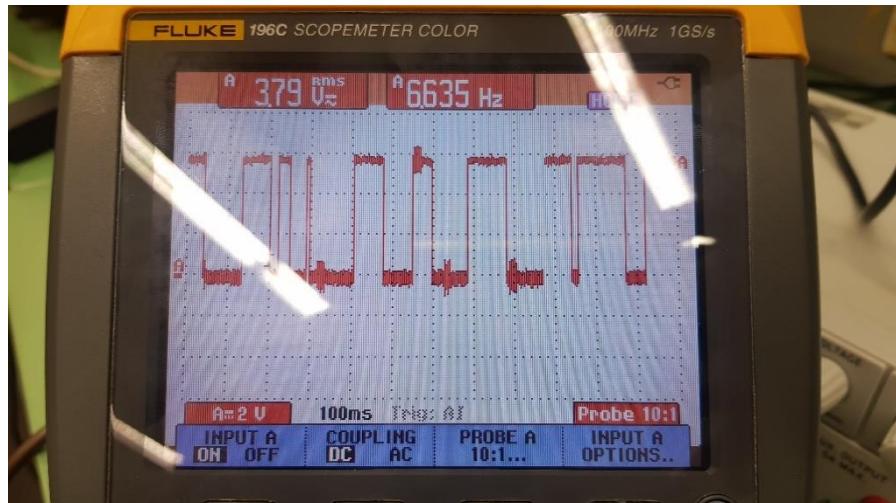


Abbildung 97: Getestete schnellere Schwingung am Oszilloskop

Nach dem ersten Versuch funktioniert die Lichtschranke einwandfrei. Es wird davon ausgegangen, dass das Eingangssignal quadratisch ist.

Im weiteren Verlauf wird die Frequenz nach einer sinusförmigen Anregung mit demselben untersucht (siehe Kapitel 2.6.9).

2.6.6 Implementierung Digitale Filter in Software:

Ein digitaler Filter filtert die Eingänge analoger Messwerte aus dem ADC, bestimmte Frequenzen oder auch binäre Inputs.

Es wird der IIR-Filter implementiert und auf Funktion getestet. Der IIR Low-Pass Filter ist schnell berechnet und verhält sich wie ein analoger Tiefpass Filter. Der IIR Filter ist programmietechnisch der einfachste Filter. Der Code stellt sich in der Abbildung 98 dar.

```

10 /*
2 * IIR_1_Order.c
3 *
4 * Created on: 09.12.2019
5 * Author: Home
6 */
7 **** IIR Filter Program*****
8 #include "IIR_1_Order.h"
9
10 float old_value=0.0f;
11
12 float coeffs[2] = { // Koeffizienten des Filters (werte aus der Vorlesung entnommen)
13     0.1f, // new value weight:Je höher, desto schneller reagiert die Ausgabe,
14     //aber es kommt auch mehr Rauschen durch.
15     0.9f // old value weight:Je höher, desto stärker werden Störungen entfernt,
16     //desto langsamer ändert sich aber auch die Ausgabe des Filters
17 };
18
19
20
210 float Filter_IIR_1_Order (float new_value){ // IIR Filter 1 order function
22
23     // Berechnung y = a0 * x(k-0) + a1 * x(k-1)
24     return old_value = coeffs[0] * new_value + coeffs[1] * old_value;
25 }
26
27
280 /*Quelle: [atwillys.de] Digitale Filter in C für Embedded-Anwendungen
29 * http://atwillys.de/content/cc/digital-filters-in-c/
30 */
31

```

Abbildung 98: IIR Filter Implementierung (Link aus [15])

Im Gegensatz zum FIR ist der IIR Filter stabiler. Jedoch ist er komplizierter und braucht mehr Speicher, da er den Mittelwert über die letzten Eingabewerte bildet. In der Abbildung 99 wird das einfachste Programm für eine FIR-Implementierung dargestellt.

```

10 /*
2 * Filter_3_Order.c
3 * Created on: 08.12.2019
4 * Author: Sami
5 */
6 #include "Filter_3_Order.h"
7 // Filter coefficient is 1(no Weight)
8 float FIR[3];
9 int FIR_zaeher = 0;
10 float Sum_Fir = 0;
11 float Div_Fir = 0;
12 float Filter_FIR_3_Order (float FIR_Value) // FIR Filter 3 order function
13 {
14     for (FIR_zaeher = 0; FIR_zaeher <2; FIR_zaeher++) // for schleife für 3 Werte
15     {
16         FIR[FIR_zaeher]= FIR_Value;           // auffüllen des FIR Array
17         Sum_Fir = (Sum_Fir+FIR[FIR_zaeher]); // summieren alle werte des Array dann dividieren durch 3
18         Div_Fir = Sum_Fir/3;                // Resultat
19     }
20     FIR_zaeher = 0;
21     Sum_Fir = 0;
22
23     return(Div_Fir); // Rückgabe der Wert des filters
24 }
25

```

Abbildung 99: FIR Filter erste Lösung Implementierung

Die erste Lösung ist eine „for“ Schleife, um ein Array auszufüllen und die Summe der Arraywerte durch die Anzahl der Werte zu dividieren. Eine optimierte Lösung ist durch einen Ring Buffer zu erreichen. Diese Methode nutzt ein Pointer, um Datenspeicher zu minimieren.

In der Abbildung 100 ist das Programm für einen FIR Filter der 5.Ordnung mittels Ring Buffer dargestellt.

```

11// This Fir 5 order Filter use the Ring Buffer Method to minimize data storage when moving average
12// Coef =1; no Weight
13
14#include "FIR_5Order_RingBuffer.h"
15
16float filter_buffer[5];           //Buffer Filter 5 werte
17float filter_sum = 0;             // summe der werte buffer
18float *filter_position;          //zeiger auf die position in buffer
19
20float FIR_5_RingBuffer(float new_value){}
21
22filter_sum = 0;                 //initialise summe
23filter_position = filter_buffer;//initialise zeiger anfangs array
24for(int i=0; i<5; i++)
25{
26    filter_buffer[i] = 0;        //alle speicher auf 0 setzen
27
28    filter_sum -= *filter_position; // alt wert von der summe substrahieren
29    *filter_position = new_value; // old value mit new überschreiben
30    filter_sum += new_value;     //summe mit neuer wert inkrementieren
31    //check ob ring buffer voll ist
32    if(++filter_position >= filter_buffer + 5)
33    {
34        filter_position = filter_buffer; // setzt zeiger wieder auf erste wert speicher
35    }
36
37
38    return filter_sum *0.2; // return das resultat der Summe/5 oder Summe*0.2
39}
40/* Quelle : [atwillys.de] Digitale Filter in C für Embedded-Anwendungen
41 * http://atwillys.de/content/cc/digital-filters-in-c/
42 */

```

Abbildung 100: FIR Filter Ring Buffer Lösung (Link aus [15])

Die Filterkoeffizienten sind 1, da keine Gewichtung der Werte verwendet wird. Die Anzahl und Werte der Koeffizienten bestimmen, welche Aufgabe der Filter durchzuführen hat (wie Tief-, Hoch- und Bandpass).

Eine Float-Division (Teilen einer Kommazahl) dauert dabei länger als eine Integer-Division. Bei einer Integer-Division können teilweise die Bits einfach nach rechts verschoben (Bit-Shifting) werden, wenn die Puffergröße 2 Bit (oder Potenz von 2, wie 2^2 entspricht). Zum Beispiel wäre ein Filter 8. Ordnung = 2^3 , das heißt, man kann, statt durch 8 zu dividieren, den zu verarbeitenden Integer-Wert um 3 Bits nach rechts verschieben. Dies hat den Vorteil, deutlich weniger Rechenzeit als eine reguläre Division zu benötigen. Speziell bei zeitintensiven Aufgaben kann dies große Vorteile bringen.

Die Nutzung eines digitalen Filters ist in diesem Projekt jedoch aufgrund nicht-anfälliger analoger Technik vermutlich irrelevant, jedoch wurde das Thema aufgrund eines möglichen späteren Verbesserungspotentials näher betrachtet.

2.6.7 Schrittmotor-Treiber Softwaretest

Die y- und z-Achse wurden mit dem gleichem Stepper Treiber angesteuert. Damit gilt das gleiche Prinzip für beide Motoren mit einem gleichen Programmcode für beide Achsen. Zunächst wird das Programm für den Stepper der y-Achse dargestellt (siehe Abbildung 101).

```
2④ * Schrittmotor_Y.c⑤
7 #include "stm32f3xx_hal.h"
8
9 /* Schrittmotor_Y_Achse */
10 void Stepper_Y_Achse(void){
11 // Initialisierung
12
13 HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, GPIO_PIN_SET);           // Reset (inverted) PIN
14 HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, GPIO_PIN_SET);           // Sleep (inverted) PIN
15 HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_RESET);         // Enable (inverted) PIN
16
17 // Rotation Schleife
18 HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5,SET);                      // Direction Pin (+sense rotation)
19
20     for(int i=0; i<200; i++)                                // 200 Schritte = 360°
21     {
22         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET); // STEP PIN High
23         HAL_Delay(500); // 1ms
24         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_RESET); // STEP PIN Low
25         HAL_Delay(500);
26         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET); // LD2 on
27     }
28     HAL_Delay(2000);
29
30 HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5,RESET);                    // Direction Pin (-sense rotation)
31
32     for(int j=0; j < 200; j++)
33     {
34         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET);
35         HAL_Delay(500);
36         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_RESET);
37         HAL_Delay(500);
38         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
39     }
40
41     HAL_Delay(2000);
42 }
```

Abbildung 101: Stepper-Treiber der y-Achse

Anschließend erfolgt das Programm für den Schrittmotortreiber in der z-Achse (siehe Abbildung 102).

```

2④ * SchrittMotor_Z.c
7 #include "stm32f3xx_hal.h"
8
9
10 /* Schrittmotor_Z_Achse */
11
12⑤ void Stepper_Z_Achse(void){
13
14 // Initialisierung
15
16 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_SET);           // Reset (inverted) PIN
17 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);           // Sleep (inverted) PIN
18 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);         // Enable (inverted) PIN
19
20 // Rotation Schleife
21 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_SET);           // Direction Pin (+sense rotation)
22
23 for(int i=0; i<200; i++)          // 200 schritte = 360°
24 {
25     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET); // STEP PIN High : HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_6);
26     HAL_Delay(500); //1ms
27     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET); // STEP PIN Low
28     HAL_Delay(500);
29     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET); // LD2 on
30 }
31 HAL_Delay(2000);
32
33 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_RESET);           // Direction Pin (-sense rotation)
34
35 for(int j=0; j < 200; j++)
36 {
37     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET); // kann auch mit Toggle pin: HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_6);
38     HAL_Delay(500);
39     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET);
40     HAL_Delay(500);
41     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
42 }
43
44 HAL_Delay(2000);

```

Abbildung 102: Stepper-Treiber der z-Achse

Laut Datenblatt müssen der Reset- und Sleep-Pin auf High und der Enable-Pin auf Low gesetzt werden, um den Stepper-Treiber zu initialisieren. Um die Rotationsrichtung zu bestimmen, muss der Pin-Direction entweder auf 0 oder 1 umgesetzt werden.

Innerhalb der „for“ Schleife bestimmt man die maximale Schrittzahl. Dann kann mit Umschalten des Pins Step eine PWM erstellt werden. Ein maximales Delay laut Datenblatt ist 2 ms zwischen High und Low. Eine LED (LD2) dient im Beispiel nur zur Anzeige, dass das Programm entsprechend ordnungsgemäß läuft. Optional wäre eine PWM direkt durch einen Timer zu erzeugen, mit einer Periode von 2 ms und einem Duty Cycle von 50%. In der Abbildung 103 befindet sich der Code für den PWM-Befehl.

```

// PWM function(optional)

    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_SET);           // Direction Pin (+sense rotation)

    for(int i=0;i<200;i++)
    {
        // STEP PWM Pin 2ms 50% duty; siehe einstellung for Timer
        HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1);
    }
    //HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_1);           // Stop PWM

    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_RESET);           // Direction Pin (-sense rotation)

    for(int j=0;j<200;j++)
    {
        HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1);           // STEP PWM Pin 2ms 50% duty
    }
    //HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_1);
}

```

Abbildung 103: PWM Optional Stepper

Nach erfolgreichem Test wird das Ergebnis auf dem Bildschirm des Oszilloskops (siehe Abbildung 104) betrachtet.

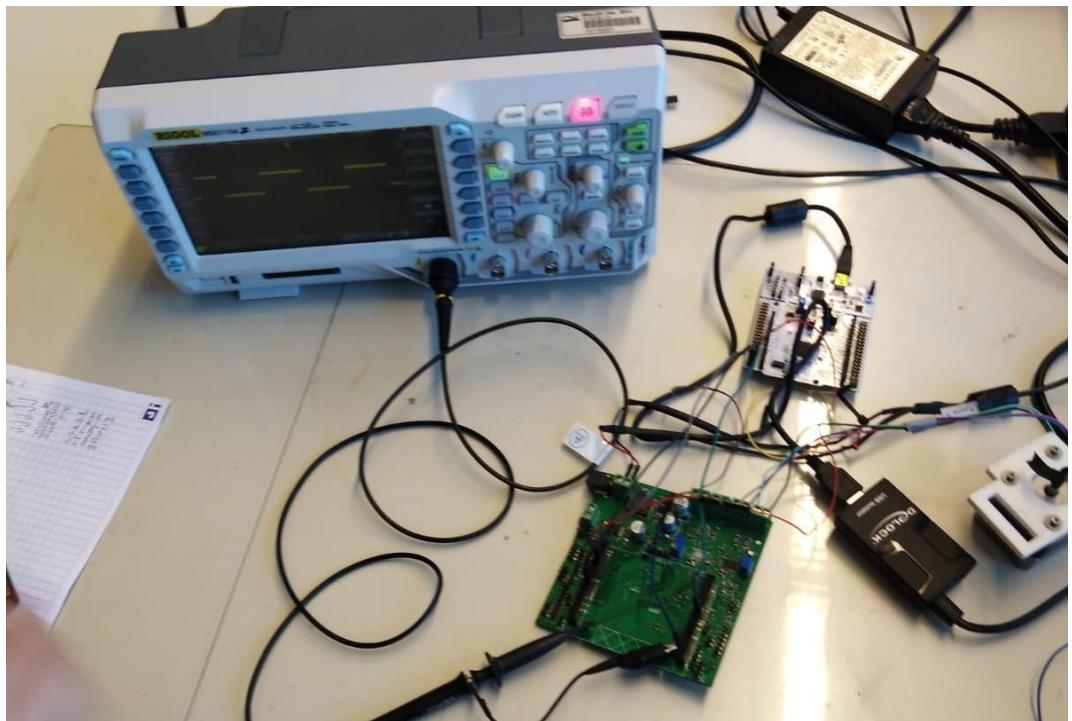


Abbildung 104: PWM Stepper Test

Das Ergebnis einer Drehung mit 10 Schritten mit einer 1 Sekunden-Periode sieht dabei folgendermaßen wie in der Abbildung 105 aus.

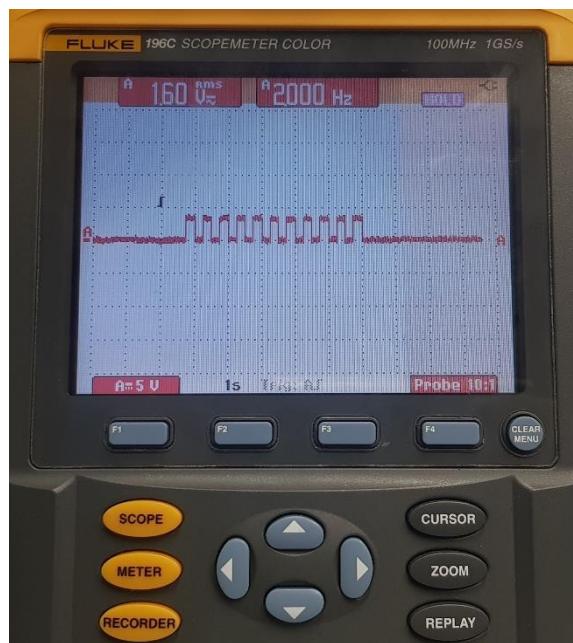


Abbildung 105: Schrittmotor 10 Schritte Test

2.6.8 H-Bücke Software und Funktion Test

Nach erfolgreichem Test der Treiber kann nun die H-Brücke getestet werden. Die H-Brücke hat dabei 4 anzuschließende Anschlüsse:

- PWM: Eingang PWM Signal
- DIR: Richtung des Stroms ändern; 0 oder 1 (siehe Kapitel 1 H-Brücke Tabelle)
- DIS: Inverse von Enable und muss auf 0 gesetzt werden
- SO: Serial Output Pin für das Debuggen

Der Basistest beginnt mit einer einfachen LED.

Hier wird eine PWM mit einer 20 KHz Frequenz an die H-Brücke gesendet und die Richtung bleibt gleich. In der Abbildung 106 wird der Code dargestellt.

```
/* USER CODE BEGIN 2 */  
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, RESET); // DIS PIN  
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, SET); //DIR PIN  
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1); //PWM PIN
```

Abbildung 106: H-Brücke Test Code

In der Abbildung 107 ist erkennbar, dass die LED leuchtet. Die gelben und orangenen Kabel kommen aus der Buchse der H-Brücke und sind mit der LED und den Vorwiderständen verbunden.

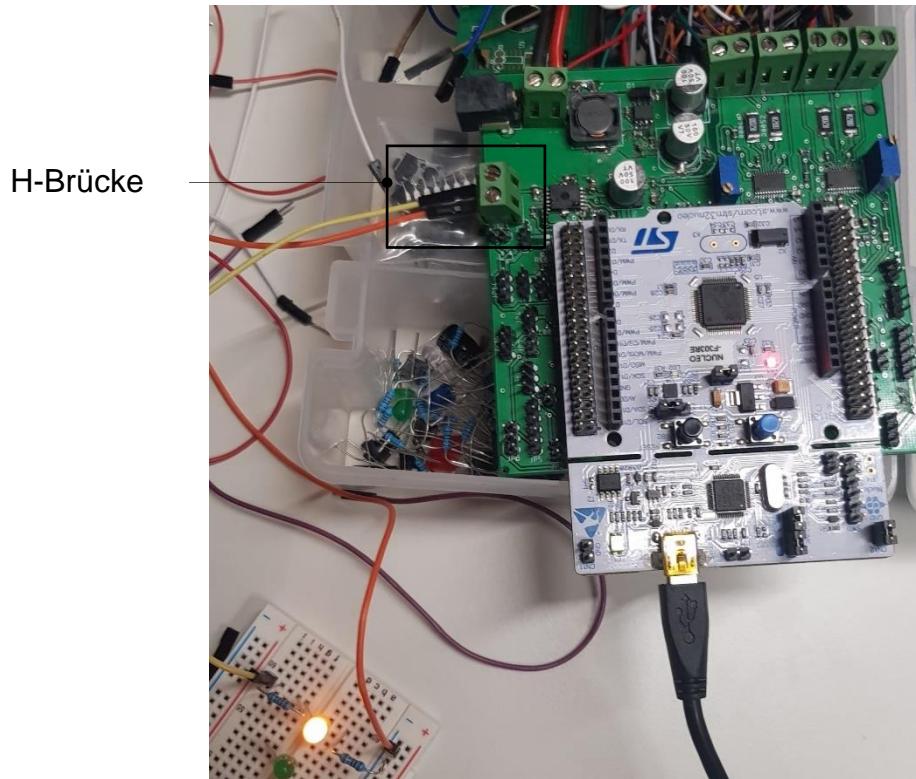


Abbildung 107: H-Brücke Programm-Test mit LEDs

Zur Prüfung der richtigen Signalausgabe der H-Brücke muss direkt an der Buchse auf der Platine gemessen werden. In der Regel gibt ein Pin eine PWM aus, während der andere Pin auf high liegt und umgekehrt bei umgeschalteter Stromrichtung. Die Ergebnisse erfüllen die Erwartungen (siehe Abbildung 108 und Abbildung 109).

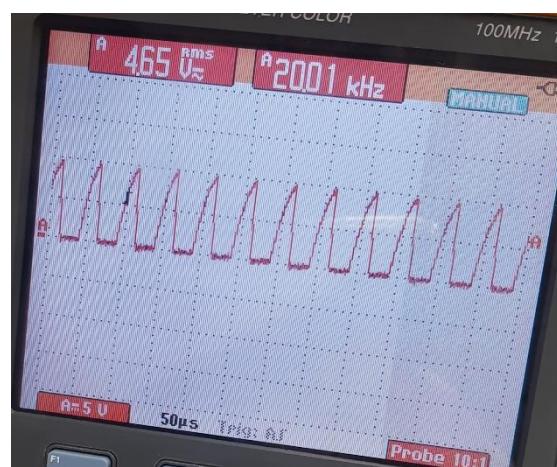


Abbildung 108: PWM H-Brücke Test

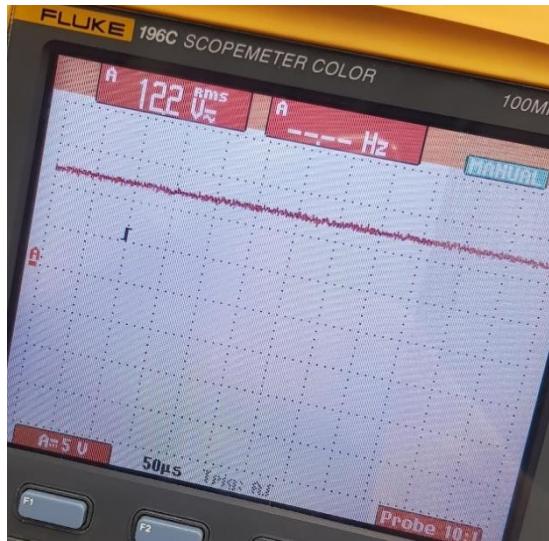


Abbildung 109: High Signal H-Brücke Test

Nach dem Anschließen der H-Brücke mit dem System wurden folgende Ergebnisse festgestellt:

- Der Schwinger kann mit einer PWM direkt angeregt werden. Jedoch muss die Stromrichtung umgeschaltet werden, um eine stabile Schwingung zu erhalten. Diese Methode nennt sich Impuls-Anregungs-Methode, da die Methode beim richtigen Zeitpunkt einen Impuls gibt und die Stromrichtung ändert. Diese Methode ist bei uns unerwünscht, da unser System sehr träge ist.
- Die zweite Methode umfasst die Sinusanregung: Diese Methode wird anfangs mit einem Sinusgenerator getestet und zeigt gute Ergebnisse. Dadurch erkennt man die Resonanzfrequenz unseres Systems in zwei Fällen: 8.3 Hz mit Hülse und 15.8 Hz ohne Hülse (Gewicht oder Laserhalter). Im nächsten Schritt wird diese Methode genauer untersucht und mit Experimenten verifiziert.

2.6.9 Sinusförmige Anregung - Funktionstest

Hiermit werden verschiedene Methoden untersucht, die eine sinusförmige Anregung ausgeben.

Zuerst wird die Anregung durch ein „Tonemitter“ (Programm auf einem Computer, das die Frequenz vom Ausgangssignal des Audio-Ausgangs übernimmt und einen Verstärker dann verstärkt, welcher im Gegensatz zum Sinusgenerator eine hohe

Amplitude besitzt und somit mehr Strom ausgeben kann, damit unser System eine ausreichende Schwingung simulieren kann) erzeugt (siehe Abbildung 110).

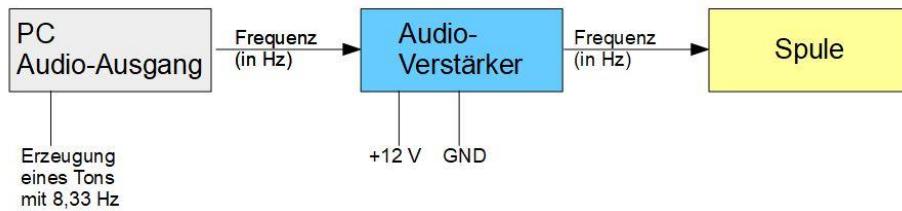


Abbildung 110: Testaufbau mit Audio-Verstärker

Das Ergebnis des ersten Experiments wird in der Abbildung 111 dargestellt. Der Vergleich der Frequenzen wurde gleichzeitig vom Oszilloskop und der Lichtschranke erfasst.

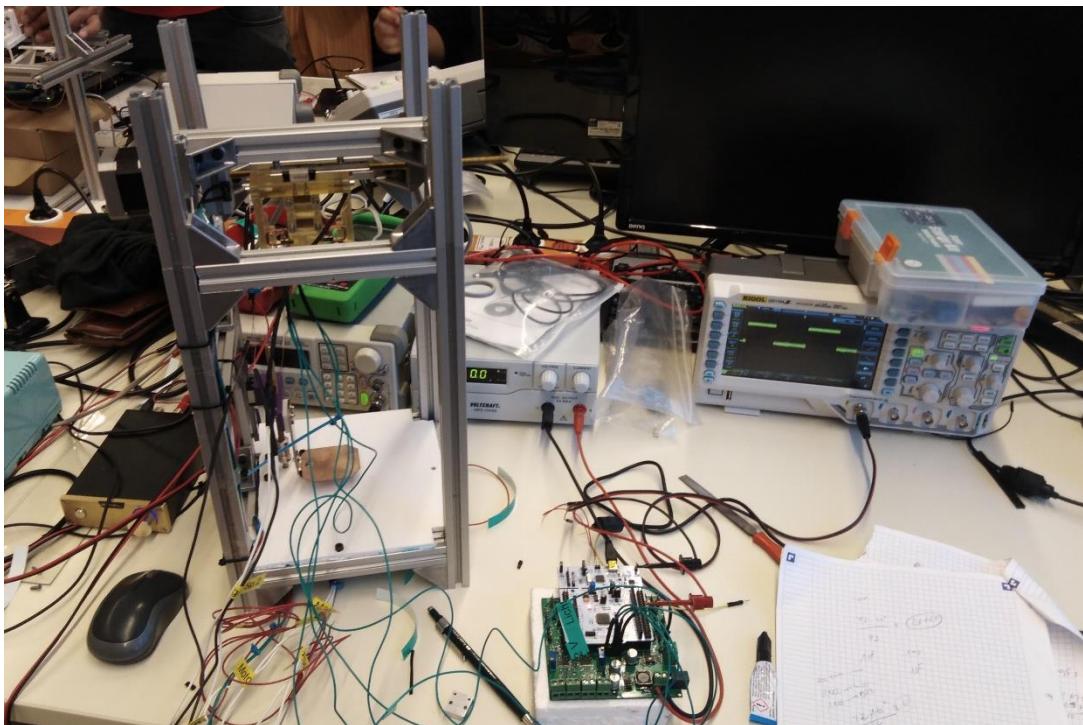


Abbildung 111: Sinuserzeugung mittels Tonemitter

Ab einer bestimmten Amplitudenhöhe des Sinussignals (das mit dem Ausgabe-Pin der Lichtschranke verbunden ist) triggert es den µC über den Input Capture Modus. Hier ist hier anzumerken, dass die Periode des Sinus-Signals nicht gleich dem Rechtecksignal entspricht. D.h. um die Frequenz des Sinussignals zu ermitteln, müssen die Phase Delay und Amplitude geregelt werden. Dies wird in der Abbildung 112 grafisch dargestellt.

Dabei ist zu sehen, dass die Phasenverschiebung beachtet werden muss.

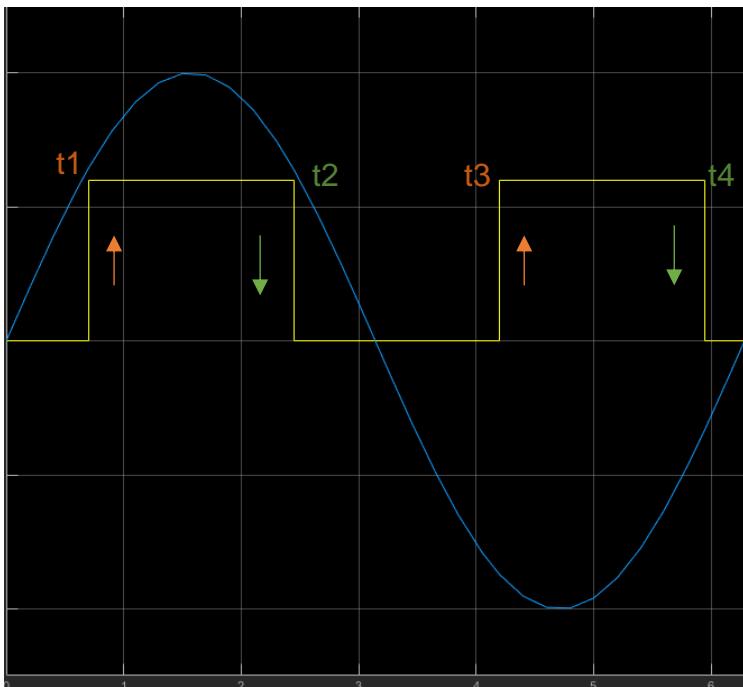


Abbildung 112: Matlab Darstellung Sinus- und Rechtecksignal

Das Programm für die Lichtschranke berechnet die Frequenz des Sinus wie folgt:

In der Abbildung 112 sind 2 Pfeile orange gefärbt und stellen die steigende Flanke dar, während die grünen Pfeile die fallende Flanke darstellen. Die Zeitvariablen {t1 bis t4} stellen dar, wann die Ereignisse geschehen sind.

$$\text{Folgende Formel ergibt sich: } t_3 - t_1 = t_4 - t_2 = \frac{\text{Periode des Sinus}}{2}$$

Um die Periode des Sinus zu berechnen multipliziert man die Differenz der Zeiten mit 2. Dann berechnet man die Inverse, um die Frequenz auszurechnen. Diese Methode zeigt das gewünschte Resultat. Die Resonanzfrequenz ist hierbei 15,8 Hz, da die Laserhülse nicht vorrätig war und somit eine andere resultierende Eigenfrequenz aufgrund des fehlenden zusätzlichen Gewichts entstand. Das ist für diesen Test jedoch nicht relevant. Das Ergebnis von der Lichtschranke ist in der Abbildung 113 zu sehen.

VarViewer1		
Variable Name	Address/Expression	Read Value
Frequency	0x20000468	15.06932
Frequency_2	0x2000047c	15.827794
IC_Value1	0x2000045c	43113.0
IC_Value1_2	0x20000470	38251.0
IC_Value2	0x20000460	39952.0
IC_Value2_2	0x20000474	41410.0
Period	0x20000464	0.06636
Period_2	0x20000478	0.06318
Pulsewidth	0x20000480	0.02916
Duty_Cycle	0x20000484	45.020844

Abbildung 113: Ergebnis erster Versuch Sinusfrequenz mit der Lichtschranke

Im nächsten Versuch wird ein Sinus nicht aus einem Tonemitter, sondern mittels Software erzeugt.

2.6.10 Sinuserzeugung mittels Software:

Ein Sinussignal ist ein analoges Signal, welches eine positive und eine negative Amplitude enthält. Um ein Sinus zu erzeugen, sind drei Methoden möglich. In allen Methoden ist ein Sinus „Lookup Table“ zu definieren, in dem die Werte vom Sinus gespeichert sind.

- Die erste Methode erzeugt ein Sinussignal anhand eines PWM-Signals, der verschiedene Duty Cycle nach der Zeit ausgibt und somit ergibt sich ein Positiv-Sinus.

Hier ist anzumerken, dass der µC keine negative Spannung ausgibt.

Um die negative Welle vom Sinus zu erzeugen, muss die Richtung an der H-Brücke umgeschaltet werden.

Der Vorteil dieser Methode besteht darin, dass der gleiche Timer für die PWM-Erzeugung und die Integration der neuen Werte in den Duty Cycle verwendet werden kann.

Eine Darstellung dieser Methode befindet sich in der Abbildung 114.

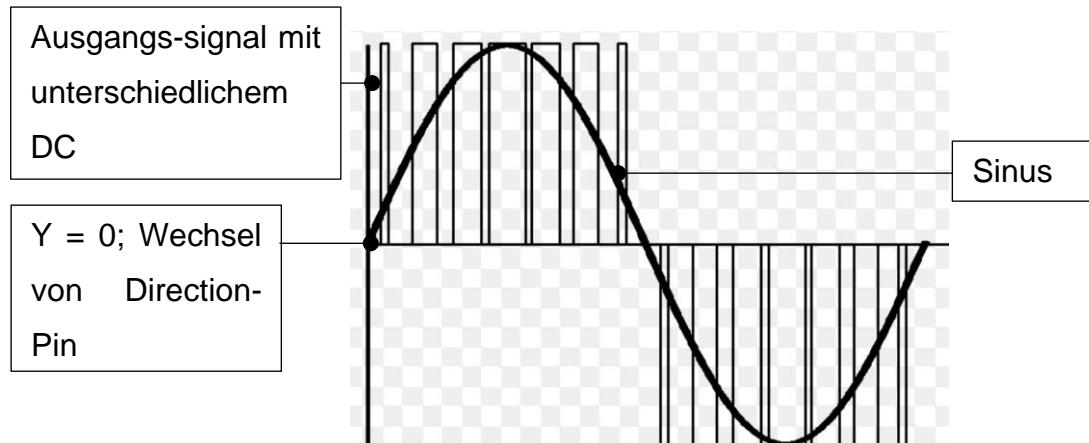


Abbildung 114: Erzeugung Sinus mit PWM [16]

Der Code für diese Methode ist im Folgenden in Einzelschritten dargestellt:

1. Deklaration der Variablen festlegen (siehe Abbildung 115). Dazu wird die Anzahl der Einzelwerte des Sinus festgelegt (aus wie vielen einzelnen Werten die Sinuskurve simuliert werden soll). Des Weiteren wird die Frequenz der PWM zur Ansteuerung der H-Brücke festgelegt.

$\frac{1}{2}$ Sinuswelle besteht aus [128] Werten

```

int i;
float t;
float Freq = 10.0f; • Ausgangsfrequenz
float Fpwm = 1280.0f; • PWM an H-Brücke
float Tab[128]; // Fpwm/Freq
float PI = 3.141f;
uint8_t dir;
/* USER CODE END PV */

```

Abbildung 115: Sinuswelle-Simulation über Wechsel des DC erzeugen

2. Anschließend werden die Werte in der Sinus-Tabelle (im Folgenden auch „Sinus-table“ genannt) wie in Abbildung 116 dargestellt, gespeichert. Dann wird der Pin DIS (Disable-Pin“ auf 0 gesetzt und die PWM zur H-Brücke mit dem Befehl „HAL_TIM_PWM_Start_IT“ gestartet. Der Printf-Befehl hilft bei der Überprüfung, ob die Werte richtig gerechnet sind.

```

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, GPIO_PIN_RESET); // Disable Pin auf 0
for(i=0;i<128;i++)
{
    Tab[i] = sinf((i*2*PI)/128.0f); // werte in die tabelle speichern
    printf("i=%d\n wert tab%f\n",i,Tab[i]);
}

HAL_TIM_PWM_Start_IT(&htim1, TIM_CHANNEL_1); // PWM_Hbrucke start mit IT

```

Abbildung 116: Erzeugung Sinus Tabelle

3. Callback Funktion, um neue Werte einzulesen (siehe Abbildung 117):

```

/* USER CODE BEGIN 4 */
void HAL_TIM_PWM_PulseFinishedCallback(TIM_HandleTypeDef *htim)
{
    if (htim == &htim1) {

        static unsigned int i;

        for(i=0;i<128;i++)
        {
            t = TIM1->ARR*Tab[i];
            if (t>0)
            {
                HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, GPIO_PIN_SET); // Pin Dir SET
                __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, t);
                printf("t=%f\n",t);

            }
            else
            {
                HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, GPIO_PIN_RESET); // Pin Dir RESET
                __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, -t);
                printf("t=%f\n",t);

            }
            dir=HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_1);
        }
    }
}

```

Abbildung 117: Sinus Callback Funktion

Beim Testen des Codes wird die PWM mit dem Oszilloskop beobachtet. Das Resultat zeigt die gewünschte Änderung des Tastgrades über eine lange Ein- und Ausschaltzeit. Ändert man die Zeitbasis, erkennt man die Annäherung an ein Dreiecksignal und somit die Ausbildung des Sinussignals. Die Sinuskurve kann durch einen externen Tiefpassfilter auf dem Oszilloskop betrachtet werden. Als zusätzliche Alternative wird die Sinus-Kurve anhand des Programms „Varviewer“ durch die Variable „t“ dargestellt (siehe Abbildung 118).

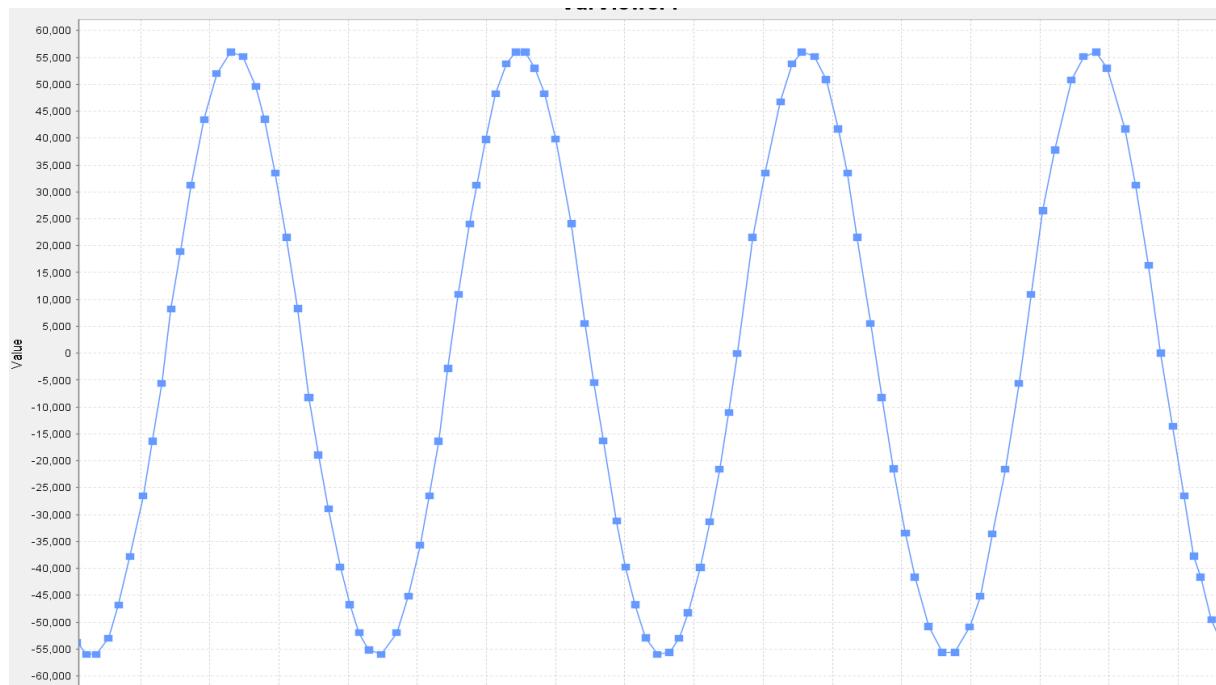


Abbildung 118: Sinus Varviewer

Die Abbildung 119 zeigt, wie die Richtungssignale der H-Brücke bei positiven und negativen Sinuswellen zwischen 1 und 0 wechseln.

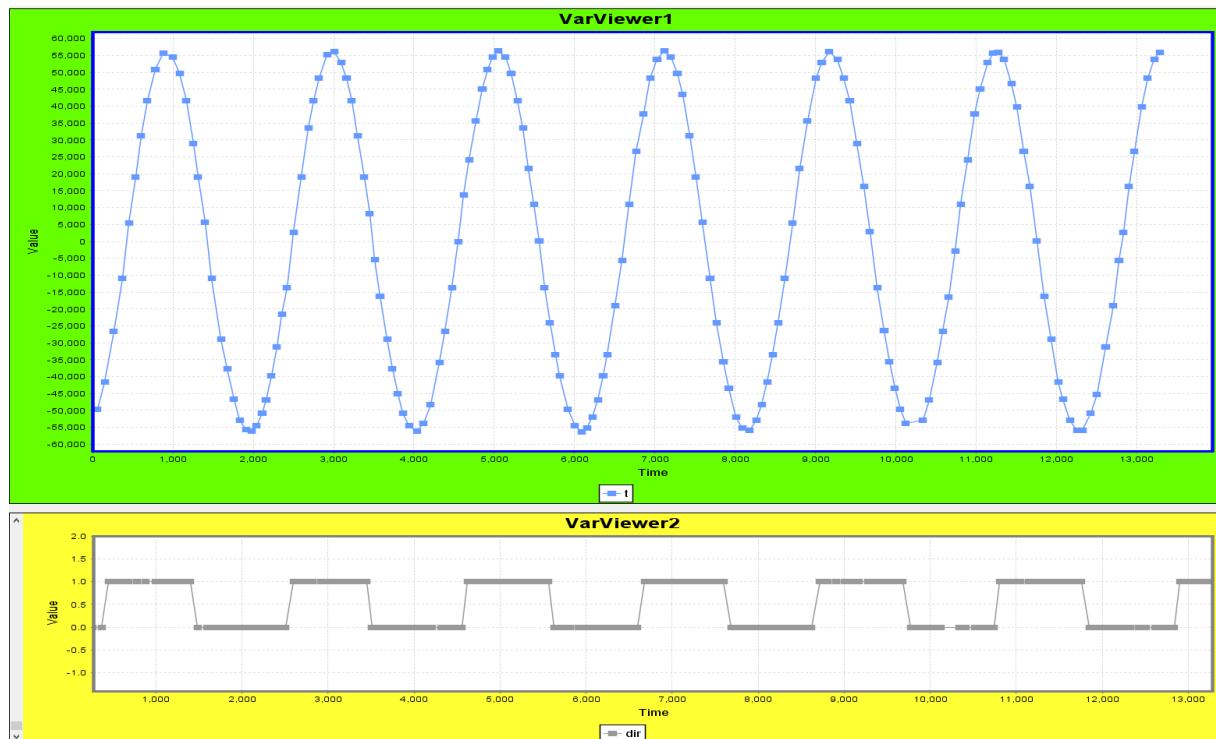


Abbildung 119: Sinus Varviewer mit DIR

- Die zweite Methode besitzt ein ähnliches Prinzip wie die Methode 1. Jedoch wird ein zweiter Timer verwendet, der bei jedem Interrupt den Duty Cycle ändert. Diese Methode ergibt das gleiche Ergebnis wie die erste. Daher ist die weitere Dokumentation nicht weiter relevant.
- Methode 3: Sinus mittels Digital Analog Converter (DAC):

Diese Methode verwendet einen DAC, um digitale Signale in analoge umzuwandeln. Die Funktionsweise dieser Methode findet sich in folgender Formel zu sehen:

$$y_{\text{SineDigital}}(x) = \left(\sin\left(x \cdot \frac{2\pi}{n_s}\right) + 1 \right) \left(\frac{(0xFF + 1)}{2} \right)$$

Abbildung 120: Formel Sinus DAC (siehe auch [17])

n_s : Anzahl Samples bzw. wie viele Punkte verwendet werden, um den Sinus zu erzeugen

0xffff: ist die Auflösung von DAC und entspricht 4095 bzw. 12 Bits (2^{12})-1.

Da der Sinus Werte zwischen -1 und 1 ergibt, muss in der Formel +1 stehen. Ein Timer triggert den DAC, um den DMA für die Übertragung von Daten aus der Sinus Tabelle im Hintergrund ohne CPU zu unterbrechen.

Die Frequenz des Sinus ist mit folgender Formel in nächster Abbildung zu berechnen:

$$f_{\text{Sinewave}} = \frac{f_{\text{TimerTRGO}}}{n_s}$$

Abbildung 121: Frequenz Sinuswelle DAC (siehe auch [17])

Nun wird der theoretische Teil in einem praktischen Versuch erforscht.
Ein erster Versuch wird mit 128 Samples gestartet und erzeugt ein Sinus mit 120 KHz.

In der main.c sieht der Code folgendermaßen aus (siehe Abbildung 122).

```
for (i=0;i<128;i++)
{
    sinus_Tab[i] = ((sin((i*2*PI)/128) + 1)*(4096/2));
}

HAL_TIM_Base_Start(&htim2);

HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1, sinus_Tab, 128, DAC_ALIGN_12B_R);
```

Abbildung 122 Sinuswelle DAC Code

Zuerst wird die Sinus-Tabelle ausgefüllt. Dann wird der Timer und der DAC im DMA gestartet.

Die Funktion hat folgende Parameter:

```
HAL_DAC_Start_DMA(DAC_HandleTypeDef* hdac, uint32_t Channel, uint32_t* pData, uint32_t Length, uint32_t Alignment)
```

DAC_HandleTypeDef* hdac: &hdac1
uint32_t Channel: DAC_CHANNEL_1
uint32_t* pData: sinus_Tab
uint32_t Length: 128
uint32_t Alignment: DAC_ALIGN_12B_R

Nach Ausführung des Programms wird folgendes Ergebnis ermittelt (siehe folgende Abbildung 123).



Abbildung 123: Sinus Frequenz 120 KHz DAC

Die Dreieckswellenform kann direkt mit dem folgenden Befehl (siehe Abbildung 124) angezeigt werden:

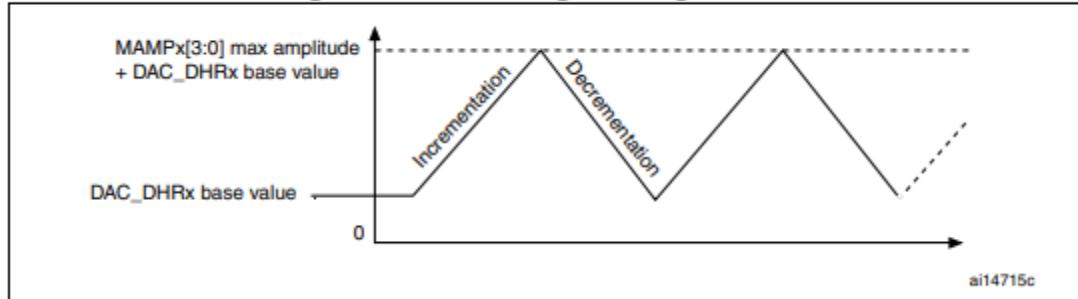
```
HAL_DACEx_TriangleWaveGenerate(&hdac1, DAC_CHANNEL_1, DAC_TRIANGLEAMPLITUDE_2047);
```

Abbildung 124 Dreieckswelle Code

Die maximale Amplitude ist 4095 bei 3,3 V und 2047 bei 1.65 V.

Anhand der Abbildung 125 wurde die Frequenz in diesem Fall folgendermaßen berechnet:

Figure 120. DAC triangle wave generation



$T = \text{Amplitude of the Waveform in terms of DAC Counts} \times 2 \times \text{Trigger Frequency}$

Abbildung 125: DAC Dreieckform (siehe auch [17])

$$F = \frac{100\,000}{2047 \cdot 2} = 24,4 \text{ Hz} \text{ mit Frequenz, Trigger } = 100000 \text{ Hz, Amplitude } = 2047.$$

Die Werte sind richtig ausgegeben.

Das Ergebnis wird auf dem Oszilloskop dargestellt (siehe Abbildung 126).

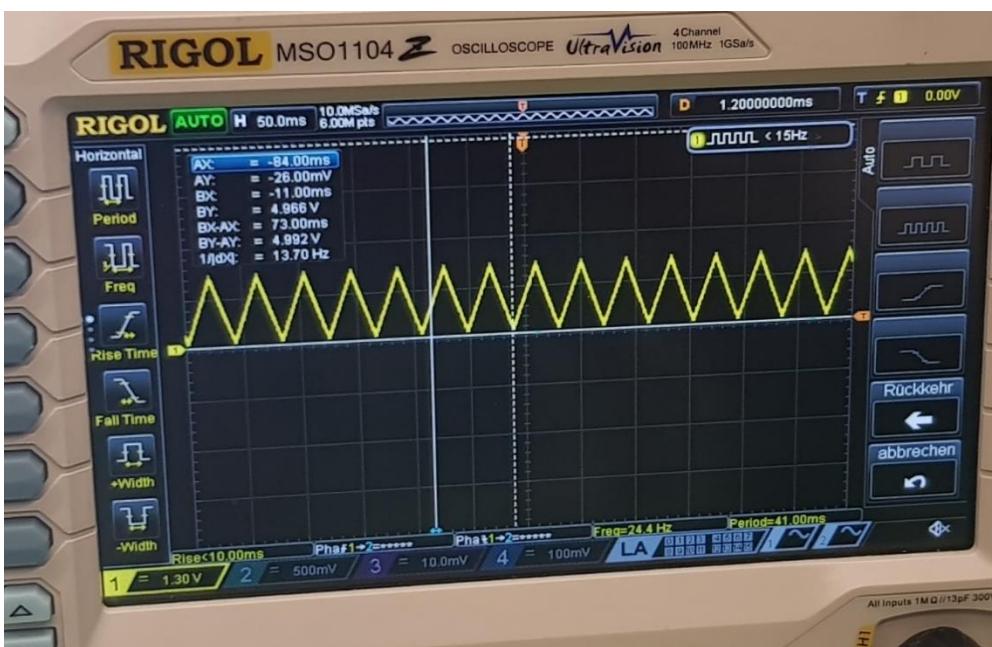


Abbildung 126: Dreieckswelle DAC

3 Inbetriebnahme und Systemtest

In diesem Kapitel erfolgen die Inbetriebnahme und ein Systemtest zu der verwendeten Hard- und Software.

3.1 Software

Es wird die Software des Programms vorgestellt. In diesem Kapitel erfolgt die Inbetriebnahme und Erstellung der Funktionen und des Main-Programms des 3D-Druckers. Dazu werden die einzelnen Variablen erstellt, Ablaufpläne angefertigt, die Funktionen geschrieben und Interrupts definiert.

3.1.1 Deklaration der Variablen

Es werden die Variablen für die Algorithmen und die Funktionen deklariert (siehe Abbildung 127).

```
*****Variablen Sinus Tabelle*****
float pi = 3.1416f;                                // Variable pi
float sinus [128];                                  // Array für Sinustabelle
float ZeitProSchwingung;                            //
float remainTime;                                  //

*****Zeit variablen Lichtschranke*****
volatile float t_1 = 0;                             // Zeitpunkt erste fallende Flanke
volatile float t_2 = 0;                             // Zeitpunkt zweite steigende Flanke
volatile float t_3 = 0;                             // Zeitpunkt zweite fallende Flanke
volatile float t_4 = 0;                             // Zeitpunkt dritte steigende Flanke
volatile float hell_1 = 0;                           // |
volatile float hell_2 = 0;                           // |
volatile float dunkel_1 = 0;                         // |
volatile float dunkel_2 = 0;                         // |
volatile float dunkel_1_90P = 0;                     // Dunkel_1 bereich mit offset
volatile float dunkel_2_90P = 0;                     // Dunkel_2 bereich mit offset

*****Variablen Amplitude Regler*****
volatile float frequenz_soll = 8.34f;                // Sollfrequenz --> muss für beide Laser jeweils angepasst werden
volatile float frequenz_gemessen = 0;                 // gemessene Frequenz
volatile float kdc = 0.25f;                           // Vorfaktor für duty cycle in Sinustabelle
volatile float Toleranz_Regler = 0.5f;                // toleranz faktor pi regler

volatile float e = 0;                                 // aktuelle Regelabweichung pro Iteration
volatile float esum = 0;                             // Aufsummierung Regelabweichung
volatile float kp = 0.8;                            // P-Anteil des Reglers
volatile float ki = 0.06f;                           // I-Anteil des Reglers
volatile float kdc_max = 0.25f;                      // Maximaler Vorfaktor von 0,445 (45%)
volatile float kdc_min = 0.1;                         // Minimaler kdc-Wert des Programms
volatile float faktor_psc = 327272.72f;             // Faktor für psc zum Bestimmen der gemessenen Frequenz
volatile float verhaeltnis_soll= 0.6f;               // Führungsgroße
volatile float verhaeltnis;                          // Verhältnis Hell zu Dunkel
```

Abbildung 127: Variablen Dekl. von Sinus, Lichtschranke (Frequenz) und Regler

Die jeweiligen Variablen sind mit Kommentaren beschriftet. Diese Kommentare geben an, zu welcher Funktion oder Algorithmus die Variablen zählen und welche Aufgabe sie erfüllen. In der Abbildung 128 werden weitere Variablen vorgestellt, die Zustände und Flags angeben.

```
*****Zustände und Flags*****
volatile int zustand = 0;                                // Zustands variable für case-Funktion der Lichtschranke
volatile int flag_schr_z =0;                             // Für was sind diese flags?
volatile _Bool flag_schr_y = 0;                          // flag für interrupt um die y achse zu bewegen
volatile _Bool ref_y = 0;                                 // Referenz der y-Achse
volatile _Bool ref_z = 0;                                 // Referenz der z-Achse
volatile _Bool ist_geschwungen = 0;                      // Flag für wie viele schwingung sind gemacht
volatile int Anzahl_Schwingung = 0;                      // Zähler fur anzahl der schwingung, parameter für regeleung
volatile int Anzahl_Ebenen = 0;                           // Endschalter an Türe geschlossen?
volatile _Bool Doorstate =0;                            // Damit Taster nicht zufällig Wert 1 enthält
volatile _Bool Start_Taster_gedruckt = 0;
volatile _Bool Init = 0;
volatile _Bool Start = 0;
volatile _Bool Drucken = 0;
volatile _Bool Ende = 0;
volatile _Bool Schalter_geschlossen = 0;
```

Abbildung 128: Deklaration der Variablen (Zustände und Flags)

Zuletzt folgen noch die Variablen für unterschiedliche Zähler, für die Kreisberechnung sowie für die Temperaturmessung (siehe Abbildung 129).

```
*****Zähler variablen*****
volatile int32_t zaehler_schr_y = 0;                     //zähler für schritt der y achse
volatile int32_t zaehler = 0;                            // Variable für das Abschreiben der Sinustabelle
volatile int32_t zaehler_X = 0;                          //zahler für 250 abtasten der x achse / Ich glaub das sind jetzt 280 oder so?
volatile int32_t zaehler_Y = 0;                          //zahler für 250 abtasten der y achse
volatile int32_t Ebenerunter = 0;

*****Variablen kreis Berechnung*****
volatile float xm = 124.5;                             // Mittelpunkt x
volatile float ym = 124.5;                             // Mittelpunkt y
volatile float dx;                                    // aktueller x-Achsen Abstand von aktuellem Punkt zu Mittelpunkt
volatile float dy;                                    // aktueller y-Achsen Abstand von aktuellem Punkt zu Mittelpunkt
volatile float dxy;                                  // Abstand zum Mittelpunkt (Ist dann der spätere Betrag aus a2 und b2)
float X_Achse[250];                                   // Array für x_Achse mit 250 Abfragen
//uint8_t ZweiD_Array[250][250] = {0};                  // test array 50*50 (nicht verwendet)

*****Temperatur Variablen*****
double tempReading;
float tempF;
double tempK;
float tempC;
```

Abbildung 129: Deklaration der Variablen für Zähler, Kreis und Temperatur

3.1.2 Erstellung Algorithmen

Im Folgenden werden die verwendeten Algorithmen in der Tabelle 15 aufgelistet.

Tabelle 15: Erstellung der Algorithmen

Algorithmen	Beschreibung
Sinuserzeugung	Wird verwendet, um eine simulierte Sinusschwingung durch die H-brücke zu erzeugen
Kreiserzeugung	Wird benötigt, um ein Kreis abzuzeichnen. Berechnung mit Satz des Pythagoras
Zeiterfassung	Misst 4 unterschiedliche Zeitpunkte pro Periode des Schwingers. Berechnet dann Frequenz
Amplitudenregler	PI-Regler, um Schwinger zu regeln
Temperaturregler	Verwendeter Regler, um Temperatur des Polymers während des Drucks zu regeln

Sinuserzeugung

Im Folgenden wird der Ablauf des Programms zur Sinuserzeugung der Anrege-Schwingung abgebildet. Dabei muss das Programm so erzeugt werden, dass die anzusteuernde H-Brücke eine Sinusschwingung in der Tauchspule simuliert. Zum Ablauf muss zunächst eine Sinustabelle im Programm deklariert werden, welche durch eine for-Schleife mit Array generiert wird (siehe Abbildung 130).

```
void generate_sinus()
{
    for(int i=0; i<128; i++)           //sinus tabelle auffüllen für 128 werte
    {
        sinus [i] = sinf(pi*i/128);
    }
}
```

Abbildung 130: Hinterlegung der Sinustabelle im Programm

Dabei werden 128 Werte (pro Halbwelle) für den zu erzeugenden Sinus verwendet. Diese Werte werden dann genutzt, um beim Ablauf eines Timers ein Interrupt auszulösen und den benötigten Wert entsprechend der aktuellen Sinus-Position abzufragen. Im folgenden PAP ist der weitere Ablauf dargestellt (siehe Abbildung 131).

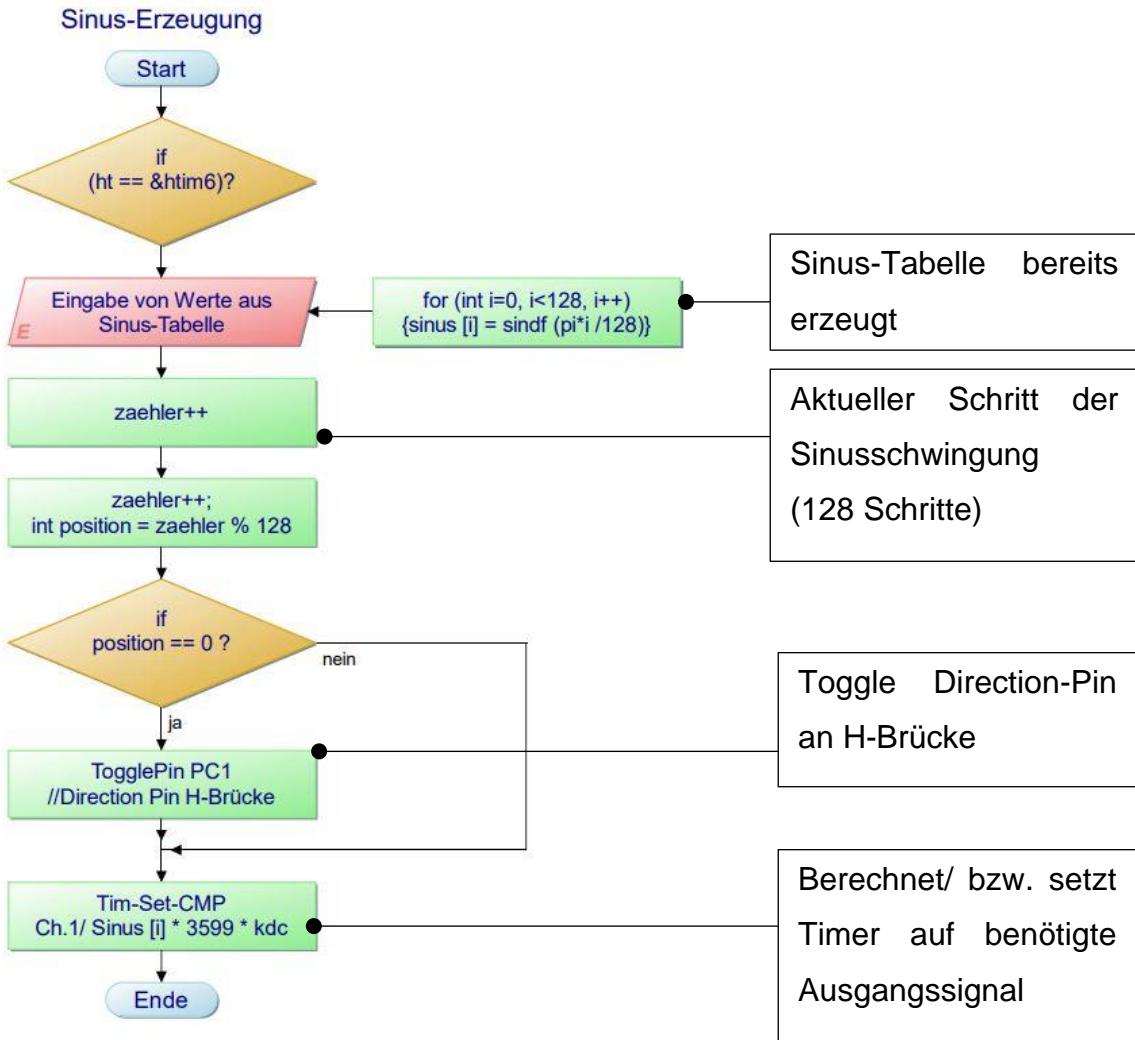


Abbildung 131: PAP zur Sinuserzeugung

Bei Auslösen des Interrupts wird der Zähler der Sinus-Position um 1 hochgesetzt und ein entsprechend äquivalentes Sinussignal wird ausgegeben. Erreicht der Zähler den Wert 128, wird die Polarität der H-Brücke geändert. In der Abbildung 132 wird der Programmcode zur Erzeugung dargestellt.

```

void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *ht)
{
    if (ht == &htim6)
    {
        zaehler++;
        int position = zaehler%128;
        if (position==0)
        {
            HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_1);
        }
    }

    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, sinus[position]*3599*kdc); //setze neu duty cycle wert in CCR mit ARR=3599
    //zaehler = 0;
}

```

Abbildung 132: Interrupt zur Sinuserzeugung

Kreiserzeugung

Es folgt der Programmablauf der Kreiserzeugung. Dabei soll ein roter Kreis (wie in Abbildung 133) auf das Druckbett projiziert werden.

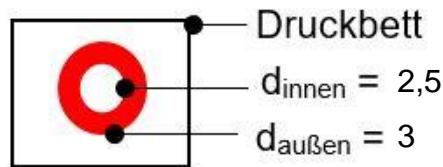


Abbildung 133: Ziel: Roter Kreis mit folgenden Maßen

Die verwendeten Variablen für das Kreisprogramm sind in Kap. 3.1.1 deklariert. Die Punkte x_m und y_m zeigen den Mittelpunkt des Kreises (Position 0/0).

Im Folgenden wird das Prinzip zur Darstellung des Kreises gezeigt. Dazu wird in Abbildung 134 und Abbildung 135 eine Prinzip-Skizze dargelegt. Es wird ein ausgedachtes Koordinatensystem über den Kreis gelegt und die Mittelpunkte x_m und y_m als Ursprung (0/0) festgelegt.

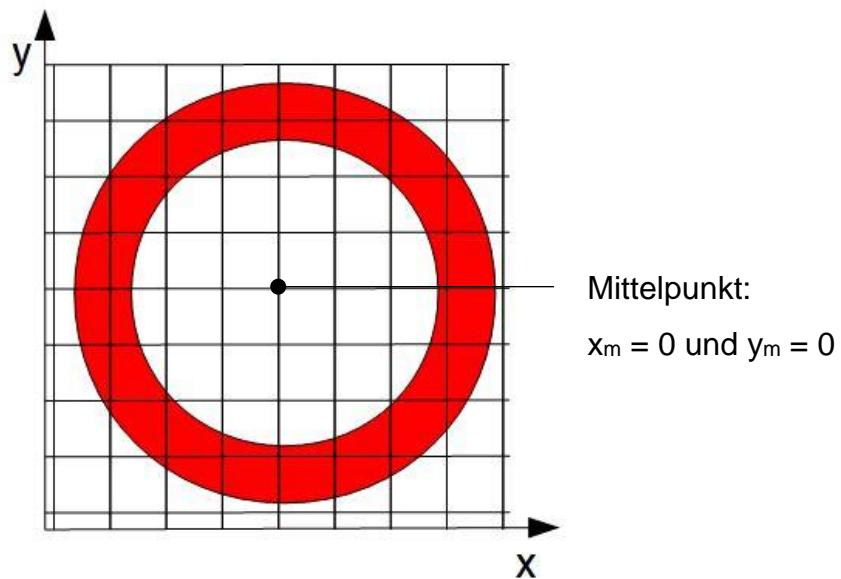


Abbildung 134: Symbolische Darstellung der Kreisberechnung

Mit Hilfe dieses Koordinatensystems wird nun der Abstand jedes Kästchens zum Mittelpunkt mithilfe des Satz des Pythagoras' berechnet. Dies wird in Abbildung 135 anhand dreier farbiger Kästchen dargestellt. Von jedem einzelnen Kästchen wird im Anschluss der Abstand dx (Abstand aktuelle x-Koordinate zu Mittelpunkt x_m) sowie Abstand dy (Abstand aktuelle y-Koordinate zu Mittelpunkt y_m) berechnet.

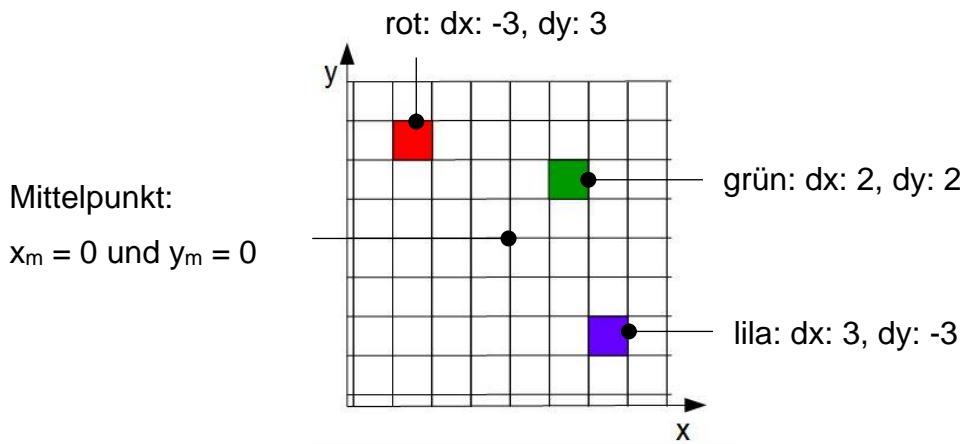


Abbildung 135: Rechnung Abstand Mittelpunkt

Im Anschluss wird aus der Prinzipdarstellung der Abstand dxy (mit Pythagoras) berechnet. Die Formel ist dabei in der Tabelle 16 angegeben.

Tabelle 16: Berechnung Abstand dxy mit Pythagoras

Farbe	dx	dy	$d_{xy} = \sqrt{dx^2 + dy^2}$
Rot	3	3	~4.24
Dann	3	3	~4.24
Grün	2	2	~2.82

Durch die Berechnung des Abstands dxy zum Ursprung kann nun ein Mindest- sowie Maximalabstand vom Ursprung festgelegt werden, indem der Laser eingeschalten werden soll.

Im weiteren Verlauf wird nun das Raster nach den entsprechenden Gegebenheiten angepasst. Dazu wird das Raster in x-Richtung auf 280 Felder und in y-Richtung auf 250 Felder festgesetzt. Das Prinzip bleibt dabei gleich, es wird weiterhin der Abstand jedes Feldes zum Mittelpunkt berechnet. Dadurch ist es möglich, einen Kreisring darstellen zu können. Es müssen lediglich Abstandsbereiche vom Mittelpunkt angegeben werden, in denen der Laser an ist. Dies wird im Projekt folgendermaßen realisiert. Dabei wird angegeben, dass der Laser angeschaltet werden soll, wenn der Abstand über 74 Einheiten und gleichzeitig unter 126 Einheiten liegt (siehe Abbildung 136).

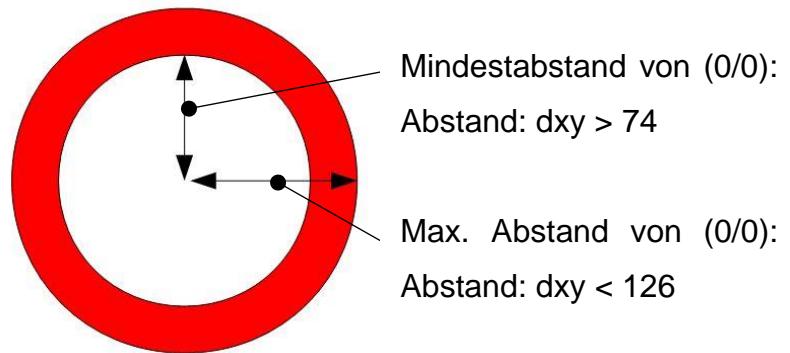


Abbildung 136: Prinzip Berechnung Kreis Programm

Nach Auswahl des Verfahrens zur Erzeugung des Kreises wird anschließend der PAP für den Kreis entwickelt (siehe Abbildung 137). Zum Start des Kreisprogramms wird ebenfalls ein Interrupt verwendet. Dabei löst der Eintritt in die Lichtschranke einen Interrupt aus und startet den Timer3. Dieser wird pro durchlaufenes Feld aktiv und aktualisiert die x-Koordinate. Zu Beginn des PAPs wird abgefragt, ob bereits mehr als 20 Schwingungen geschwungen sind. Dieser Wert ist beliebig und soll den Einschwing- und Regelvorgang des Schwingers ohne Auswirkungen auf den zu lasernden Kreis ermöglichen.

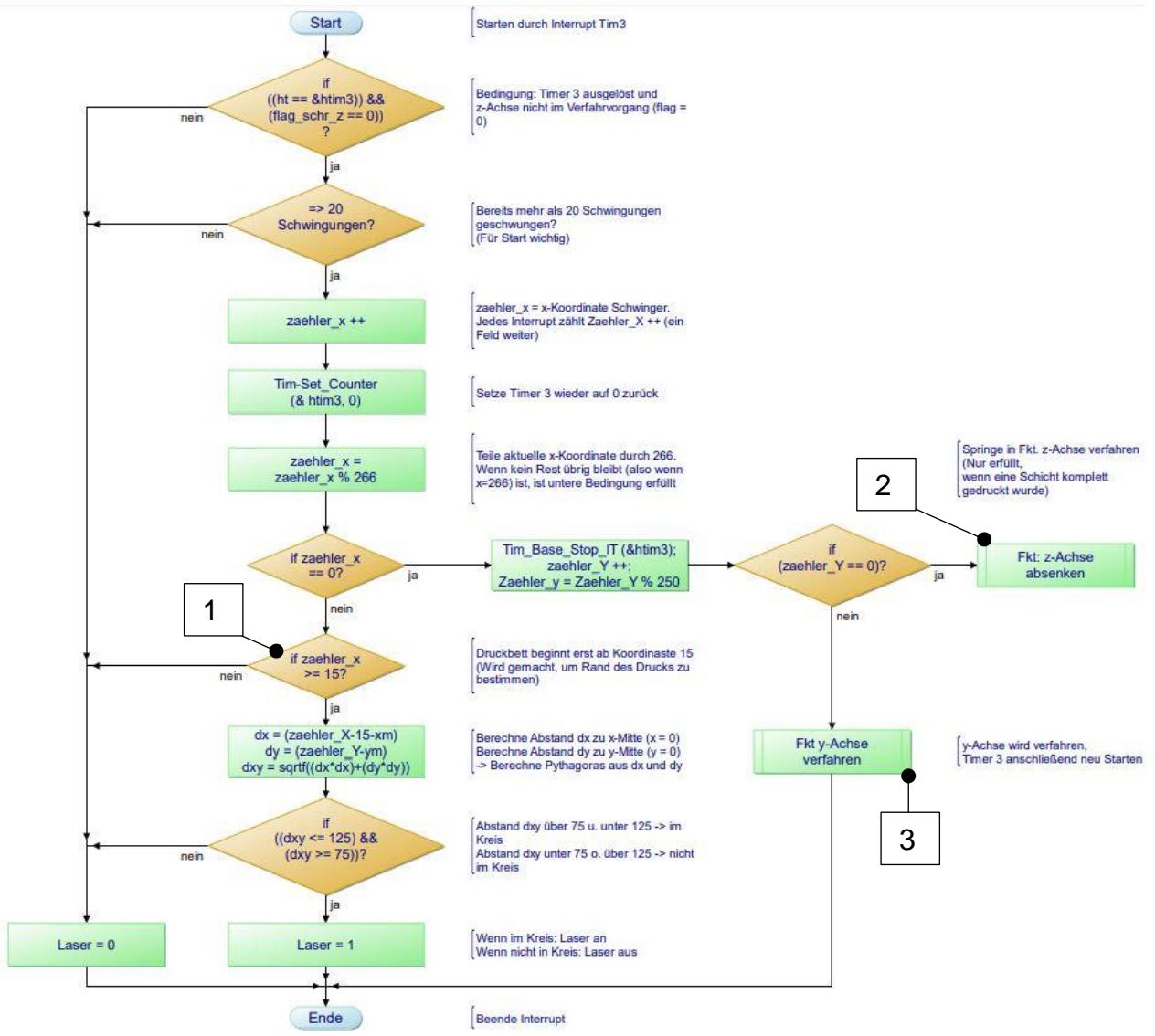


Abbildung 137: Ablaufplan Kreisprogramm

In der Pos. 1 (Bedingung: if zaehler_x >= 15) wird geprüft, ob der Schwinger sich auf der x-Koordinate bereits 15 Felder vom Rand entfernt hat. Dies wird hier als Offset verwendet und soll ein Lasern außerhalb des Druckbetts verhindern. Die Pos. 2 (Funktion z-Achse absenken) wird aufgerufen, wenn in einer Schicht alle x- und y-Koordinaten abgefahren sind und eine neue Druckschicht angefahren werden soll. Die Funktion aus Pos. 3 (y-Achse verfahren) wird hingegen verwendet, wenn in einer Schwingung alle x-Koordinaten abgefahren sind und sich die y-Achse eine Schicht verschieben soll.

Zuletzt wird der Programmcode zum Interrupt der Belichtung in der Abbildung 138 vorgestellt.

```
*****Belichtung_interrupt*****
if ((ht == &htim3) &&(flag_schr_z == 0)) //&&
{
    if (Anzahl_Schwingung >= 20)           //wenn Timer 3 auslöst (und z nicht in bewegvorgang ist) dann anfang belichten
    {
        zaehler_X++;                      //Sind bereits mehr als 20 Schwingungen durchgeführt?
        zaehler_X = zaehler_X%266;          // Setze zaehler_X um eins nach oben: zaehler_X ist aktuelle x-Koordinate
        HAL_TIM_SET_COUNTER(&htim3,0);       // Teile aktuelle x-Koordinate durch 266.
        if (zaehler_X==0)                   // Setze Counter 3 zurück
            // Wenn "zaehler_X = zaehler_X%266;" gleich 0 ist
            // (also Schwinger in Schwingung alle x-Koordinaten durchlaufen hat), dann {...}
        {
            HAL_TIM_Base_Stop_IT (&htim3); // Dann stoppe TIM_3 per Interrupt
            zaehler_Y++;                  // Wenn die Schwingung alle x-Koordinaten in einer Schwingung durch hat,
            zaehler_Y = zaehler_Y%250;      // dann setze y-Koordinate um +1 höher
            if (zaehler_Y==0)              // Kontrolliere ob in y-Koordinate schon alle 250 Koordinaten durchlaufen sind
                // Ablauf: Teile aktuellen Wert durch 250 und verwende den Rest %
                // Wenn aktuelle y-Koordinate durch /250 keinen Rest ergibt (also wenn wir auf y = 250 stehen)
            {
                flag_schr_z = 1;           // hier: Ebene absenken oder für kreis. "flag_schr_z" bedeutet, dass als nächstes der z-Motor verfahren wird
                // und so lange keine Belichtung stattfinden kann.
                // dann z-Achse_ebene_sinken();
            }
        }
        if (zaehler_X >= 15)             // Wenn 15 Felder auf x-Koordinate durchlaufen sind (also 15 Interrupts ausgelöst haben), dann:
        {
            dx = (zaehler_X-15-xm);     // Berechne den Abstand (dx) der aktuellen x-Position (zaehler_X) zum Mittelpunkt (xm)
            dy = (zaehler_Y-ym);          // und ziehe dann 15 davon ab. Sinn: Druckfeld eingrenzen, sodass die ersten 15 Interrupts keinen Laser auslösen
            dxy = sqrtf((dx*dx)+(dy*dy));
        }
        if ((dxy <= 125)&&(dxy >= 75))
        {
            Laser_switch_state(1);
        }
        else
        {
            Laser_switch_state(0);
        }
    }
}
```

Abbildung 138: Kreis-Programm in Software

Algorithmus Frequenzmessung

Es werden zunächst die Grundlagen der Frequenzmessung erläutert. Die Zeitabstände werden im weiteren als T0 bis T4 bezeichnet und in der Abbildung 139 dargestellt. Der Anfang einer neuen Periode wird hierbei als T4 (wenn die aktuelle Periode berechnet werden soll) oder als T0 bezeichnet (wenn es um die Berechnung der neuen Periode geht).

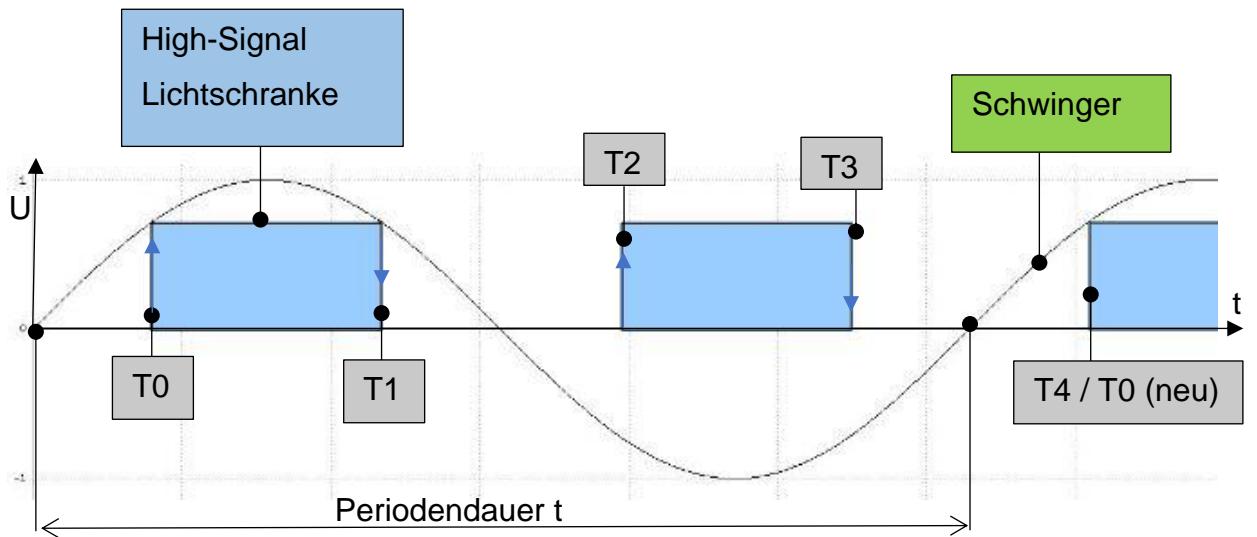


Abbildung 139: Darstellung Periodendauer und Lichtschranke

In der Abbildung 140 wird die Hell- und Dunkelzeit des Systems definiert. Hellzeit bedeutet, dass die Lichtschranke nicht unterbrochen wird, sondern der Lichtstrahl vom Empfänger registriert wird. Dunkelzeit hingegen steht für eine Lichtschranke, die unterbrochen ist.

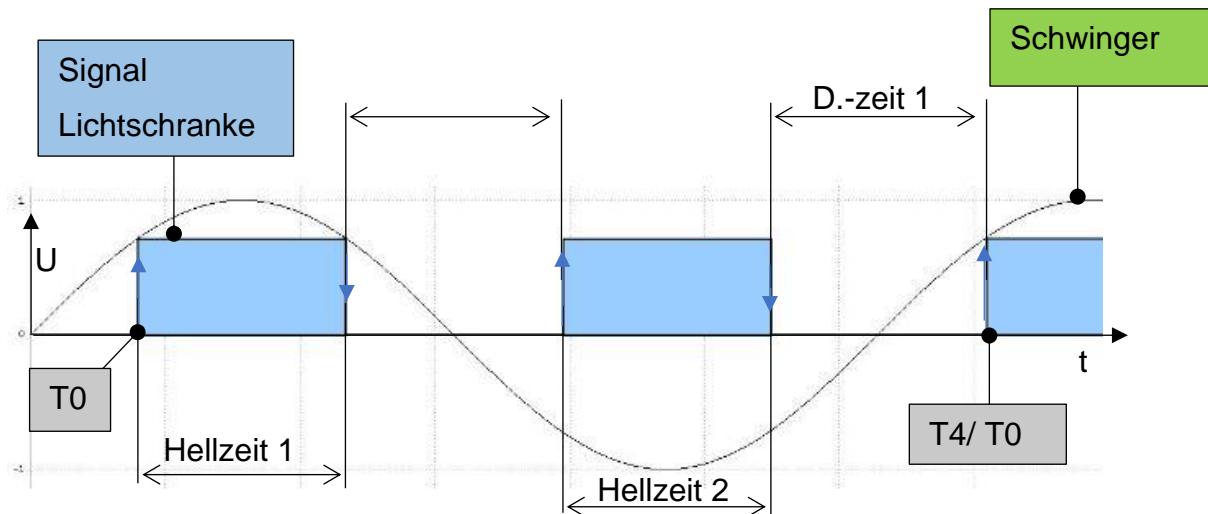


Abbildung 140: Beschreibung Hell- und Dunkelzeit

In der Tabelle 17 wird dann aufgelistet, mit welchem Channel die einzelnen Signale registriert werden und wie zu verfahren ist. Dabei wird zwischen steigenden und fallenden Flanken des Ausgangssignals unterschieden.

Tabelle 17: Beschreibung der Abbildung 140

Schritt	Signalform der Flanke; (Channel)	Beschreibung
T0	steigend (CH1)	Austritt der Blende aus Lichtschranke
T1	fallend (CH2)	Blende kommt vom Umkehrpunkt zurück und tritt in Lichtschranke ein
T2	steigend (CH1)	Austritt der Blende aus Lichtschranke
T3	fallend (CH2)	Blende kommt vom Umkehrpunkt zurück und tritt in Lichtschranke ein
T4/ T0	steigend (CH1)	Austritt der Blende aus Lichtschranke

Anhand der skizzierten Zeitpunkte können nun Gleichungen aufgestellt werden, die die einzelnen Zeitpunkte logisch miteinander verknüpfen und eine Berechnung für das Programm ergeben. Die Tabelle 18 gibt diese Berechnungen an

Tabelle 18: Berechnung der Zeitpunkte für Frequenzberechnung

Wert	Variable	Berechnung	Beschreibung
Dunkelzeit	Dunkelzeit 1	$T4 - T3$	Dunkelzeit 1 zwischen Zeitpunkt T4 abzüglich T3
	Dunkelzeit 1_90P	$\frac{T4 - T3}{280}$	Die Dunkelzeit 1 wird für Koordinaten in 280 Felder eingeteilt. Berechnet 1 „Feld“
	Dunkelzeit 2	$T2 - T1$	Dunkelzeit 2 zwischen Zeitpunkt T2 abzüglich T1
	Dunkelzeit 2_90P	$\frac{T2 - T1}{280}$	Die Dunkelzeit 2 wird für Koordinaten in 280 Felder eingeteilt. Berechnet 1 „Feld“
Hellzeit	Hellzeit 1	$T1$	Die Hellzeit 1 berechnet sich aus T1 (-T0)

	Hellzeit 2	T3 - T2	Die Hellzeit 2 berechnet sich aus T3 abzüglich T2
Verhältnis	Verhältnis $\frac{Hell1 + Hell2}{Dunkel1 + Dunkel2}$	$\frac{T1 + (T3 - T2)}{(T2 - T1) + (T4 - T3)}$	Verhältnis aus Hell- zu Dunkelzeit
Frequenz	gemessen	$\frac{PSC}{T4}$	Gibt Frequenz des Schwingers an

Anschließend kann der PAP zur Berechnung der Frequenz ermittelt werden.

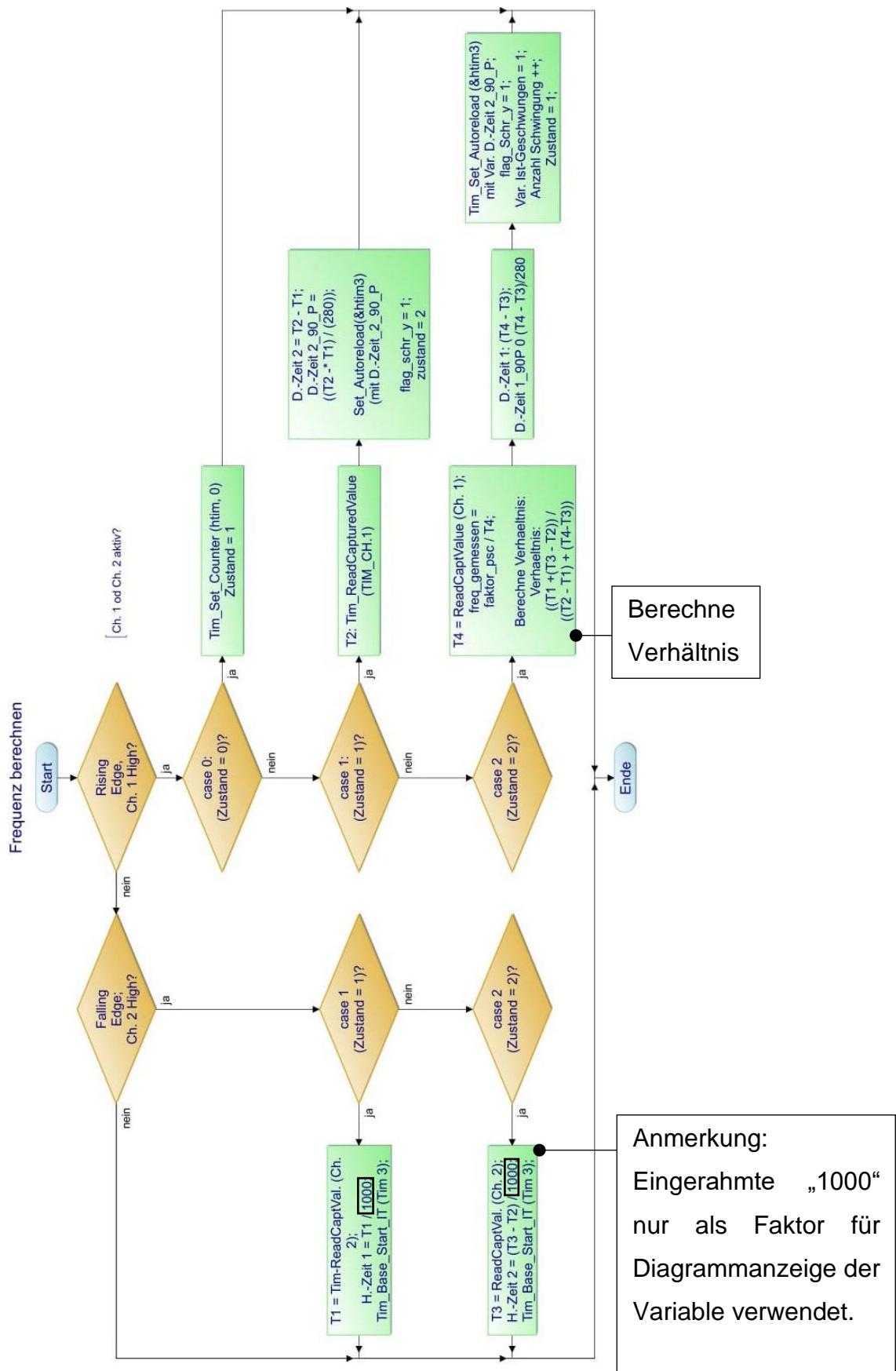


Abbildung 141: Programmablaufplan Frequenzmessung

In der Abbildung 142 wird der Programmcode zum Auslösen des Interrupts mit einer steigenden Flanke (Rising Edge) dargestellt.

```

void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)           //Input Capture interrupt für lichtschranke
{
    if(htim==&htim4)                                         //wenn der timer 4 ist dann
    {
        //*****Rising_edge_channel*****
        if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)          // if interrupt source is channel 1
        {
            switch(zustand)                                     // Capture Zustande für Rising Edge
            {
                case 0:
                    __HAL_TIM_SET_COUNTER(htim,0);                 //first Capture rising
                    zustand=1;                                    //stell counter auf 0 und zahl hoch
                    break;                                       //Übergang zum nächsten zustand
                case 1:
                    t_2 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1); //second capture rising
                    dunkel_2 =(t_2-t_1);                         // berechne dunkel zeit 2
                    dunkel_2_90P = (t_2-t_1)/280;                  // berechne dunkel zeit 2 mit offset
                    __HAL_TIM_SET_AUTORELOAD(&htim3, (uint16_t) dunkel_1_90P); //setzten des ARRwert timer 3 als zeit für abtasten
                    flag_schr_y = 1;                            //flag für die bewegung der schrittmotor der y achse
                    zustand =2;                                //transision auf zustand 2
                    break;
                case 2:                                         // case third capture rising edge
                    t_4 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1); //read third capture
                    frequenz_gemessen = faktor_psc/t_4;             // berechne die frequenz aus der dritte capture
                    verhaeltnis = ((t_1+(t_3-t_2))/((t_2-t_1)+(t_4-t_3))); //berechne aus der zeiten der verhaltnis
                    dunkel_1 = (t_4-t_3);                         //berechne die erste dunkle seite
                    dunkel_1_90P = (t_4-t_3)/280;                 //berechne daraus die hell zeit 1
                    __HAL_TIM_SET_AUTORELOAD(&htim3, (uint16_t) dunkel_2_90P); //stell die wert der dunkel im timer 3 zur belichten
                    flag_schr_y = 1;                            // flag für bewegung y_achse
                    ist_geschwungen = 1;                        //setze ist geschwungen auf 1
                    ++Anzahl_Schwingung;
            }
        }
    }
}

```

Abbildung 142: Interrupt Bedingung und Rising-Edge Erkennung

Anschließend ist das Interrupt für eine fallende Flanke (Falling Edge) abgebildet (siehe Abbildung 143). Mit diesen beiden Programmstücken kann die Frequenz gemessen, angezeigt und für den Regler verwendet werden.

```

if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_2)// if interrupt source is channel 2
{
    switch(zustand)                                     // Capture Zustande für falling Edge
    {
        case 1:
            t_1 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_2); //first Capture falling
            hell_1 = t_1/1000;                                 //read first capture falling
            HAL_TIM_Base_Start_IT (&htim3);                   //berechne daraus die hell zeit 1
                                                               //timer 3 zum belichten starten

            break;
        case 2:                                         //second Capture falling
            t_3 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_2); //read second capture
            hell_2 = (t_3-t_2)/1000;                          //berechne die zweite hell seite
            //flag_schr_y = 1;                                //stell das flag für bewegung schrittmotor
            HAL_TIM_Base_Start_IT (&htim3);                  //start timer 3 zur belichten mit interrupt

            break;
    }
}

```

Abbildung 143: Falling Edge Erkennung

Temperaturregler Algorithmus

Im Folgenden wird der Code der Temperaturregelung in Abbildung 144 dargestellt. Im weiteren Verlauf des Berichts wurde dabei ein Re-Design des Heizungsreglers durchgeführt. Dieser war ursprünglich als 2-Punkt-Regler geplant und wird aufgrund besserer Regeleigenschaften verbessert, indem ab 50 °C eine geringere Heizleistung (jetzt 3-Punkt-Regler) verwendet wird. Dies soll ein Schwanken der Temperatur besser verhindern als ein 2-Punkt-Regler. Es werden ein ADC-Wandler und ein geeigneter Temperatursensor benötigt. Timer 7 startet alle 10 Sekunden eine Abfrage.

```
*****Heizung Regelung*****  
if (ht == &htim7) // wenn timer is timer dann  
{  
    void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDefDef* hadc)  
    {  
        if (hadc == &hadc4)  
        {  
            //HAL_ADC_PollForConversion(&hadc4, 1); // poll for conversion  
            //tempReading = HAL_ADC_GetValue(&hadc4); // get the adc value  
            tempReading = adc_buf[1]; //lese die werte aus dma buffer  
            tempK = log(10000.0 * ((4096.0 / tempReading - 1))); //Specification of thermistor Modell  
            tempK = 1 / (0.001129148 + (0.000234125 + (0.000000876741 * tempK * tempK ))* tempK );  
            tempC = tempK - 273.15;  
            tempF = (tempC * 9.0)/ 5.0 + 32.0;  
            printf("\nTemperature=%2f\n",Filter_IIR_1_Order(tempC)); //Konsole Ausgabe  
            //Temperature_print(); //LCD Ausgabe  
            if ((tempC>50) && (tempC<60))  
            {  
                HAL_TIM_PWM_Start(&htim15, TIM_CHANNEL_1); //PWM für Heizung Start  
                __HAL_TIM_SET_COMPARE(&htim15, TIM_CHANNEL_1, 100); // 50% duty cycle, langsam heizen  
            }  
            if (tempC>20 && tempC<50){  
                HAL_TIM_PWM_Start(&htim15, TIM_CHANNEL_1);  
                __HAL_TIM_SET_COMPARE(&htim15, TIM_CHANNEL_1, 200); // 100% duty cycle, schnell Heizen  
            }  
            if (tempC>=60)  
            {  
                __HAL_TIM_SET_COMPARE(&htim15, TIM_CHANNEL_1, 0); // 0% duty cycle, 0 spannung  
                HAL_TIM_PWM_Stop(&htim15, TIM_CHANNEL_1);  
                Polymerready = 1;  
            }  
    }  
}
```

Abbildung 144: Programmierung Regelung Heizung

Die dargestellte Temperaturregelung wird aufgrund von Verbesserungsmaßnahmen zudem nicht mehr mit dem „Polling-Befehl“ arbeiten, da Polling die CPU ausbremst. Speziell bei vielen verwendeten Interrupts wie in diesem Projekt ist Polling sehr zum Nachteil der Performance. In diesem Programm wird der Messwert mit dem Befehl „ConvCpltCallback“ ausgelesen und der jeweilige Messwert direkt im DMA Buffer gespeichert, wird dann von dort für die Berechnung verwendet. Bei unter 50°C wird der Transistor voll durchgesteuert, bei 50 bis 60 °C wird der DC der PWM um die Hälfte reduziert und bei 60 °C komplett ausgeschalten, Hierbei setzt sich der flag „Polymerready“ auf High um den Drucker zu starten.

Amplitudenregler Algorithmus

Für den Einsatz der H-Brücke muss ein geeigneter kdc-Wert (kdc = On-Time der H-Brücke) bestimmt werden. Der Versuch wird bei der H-Brücke mit unterschiedlichen Parametern durchgeführt und das Ergebnis beobachtet.

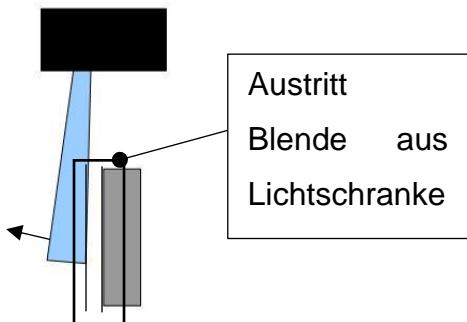


Abbildung 145: Benötigte Aktorkraft

Im Test wird dabei geprüft, ob der Schwinger genügend Energie zugeführt bekommt, dass die Blende aus der Lichtschranke hinausschwingt (siehe Abbildung 145). Dies ist der wichtigste entscheidende Punkt zur Einstellung der Aktorkraft, da die Blende an beiden Enden der Lichtschranke austreten muss, um entsprechende High- Bzw. Low-Signale zu erhalten.

Die Aktorkraft selbst wird dabei nicht mit Messgeräten geprüft, da weniger die eigentliche Kraft, als mehr das Ergebnis ermittelt werden muss. Dabei ergeben sich in der Tabelle 19 verschiedene Beobachtungen mit unterschiedlichen Parametern.

Tabelle 19: Anpassung Parameter H-Brücke

Aktor	Anregungs-frequenz (PWM)	Parameter: kdc (On-Time H-Brücke)	Beobachtung
H-Brücke	20.480 Hz	0.4	Gutes Schwingverhalten, aber H-Brücke erhitzt sich und schaltet ab
		0.35	
		0.3	
		0.15	Schwingverhalten schlecht
	10.240 Hz	0.4	Störendes Geräusch, H-Brücke schaltet sich ab (wegen Wärme)
		0.3	
	5.120 Hz	0.3	sehr Störendes Geräusch
	2.560 Hz	0.25	Guter Wert, ausreichende Energiezuführung, keine Überhitzung der H-Brücke
		0.2	

Die Tests ergeben einen bestimmten Bereich, in dem der Schwinger ausreichend Energie zugeführt bekommt, um mit der Blende aus der Lichtschranke hinaus zu

schwingen, während die H-Brücke dabei nicht überhitzt wird. Dieses Problem mit Überhitzung trat zuvor im Versuch ständig auf. Wurde versucht, die störenden PWM-Geräusche der H-Brücke zu umgehen (z.B. mit 20 kHz Ansteuerung) entstand in der H-Brücke zu viel Wärmeenergie durch die schnellen Wechsel, sodass deren Sicherheitseinrichtung nach einer Minute aktiviert wurde und sie sich ausgeschalten hat. Als die Frequenz etwas tiefer gestellt wurde (5 bis 10 kHz) ergaben sich sehr störende und laute Geräusche. Diese wurden für einen Dauerbetrieb als sehr störend empfunden.

Der Versuch ergab außerdem, dass mit niedrigerer PWM-Frequenz deutlich höhere kdc-Werte eingestellt werden konnten, ohne die H-Brücke zu stark zu erhitzen. Am Ende des Versuchs wurde dabei ein Kompromiss zwischen einem mäßigen kdc-Wert (0,1 bis 0.25) und einer eher niedrigeren PWM-Frequenz (2048 Hz) ausgewählt.

Die Implementierung des Reglers wird nach der Form des PI-Reglers angefertigt. Zunächst wurden die benötigten Variablen implementiert. Diese sind in Abbildung 146 aufgelistet.

```

volatile float frequenz_soll = 8.34f;           // Sollfrequenz |
volatile float frequenz_gemessen = 0;           // gemessene Frequenz
volatile float kdc = 0.25f;                      // Vorfaktor für duty cycle in Sinustabelle
volatile float Toleranz_Regler = 0.5f;           // toleranz faktor pi regler

volatile float e = 0;                            // aktuelle Regelabweichung pro Iteration
volatile float esum = 0;                         // Aufsummierung Regelabweichung
volatile float kp = 0.8;                         // P-Anteil des Reglers
volatile float ki = 0.06f;                        // I-Anteil des Reglers
volatile float kdc_max = 0.25f;                  // Maximaler Vorfaktor von 0,445 (45%)
volatile float kdc_min = 0.1;                     //Minimaler kdc-Wert des Programms
volatile float faktor_psc = 327272.72f;          // Faktor für psc zum Bestimmen der gemessenen Frequenz
volatile float verhaeltnis_soll= 0.6f;            // Führungsgroße
volatile float verhaeltnis;                      // Verhältnis Hell zu Dunkel

```

Abbildung 146: Initialisieren der Variablen

Zur übersichtlicheren Gestaltung wird ein PAP zum PI-Regler angefertigt (siehe Abbildung 147).

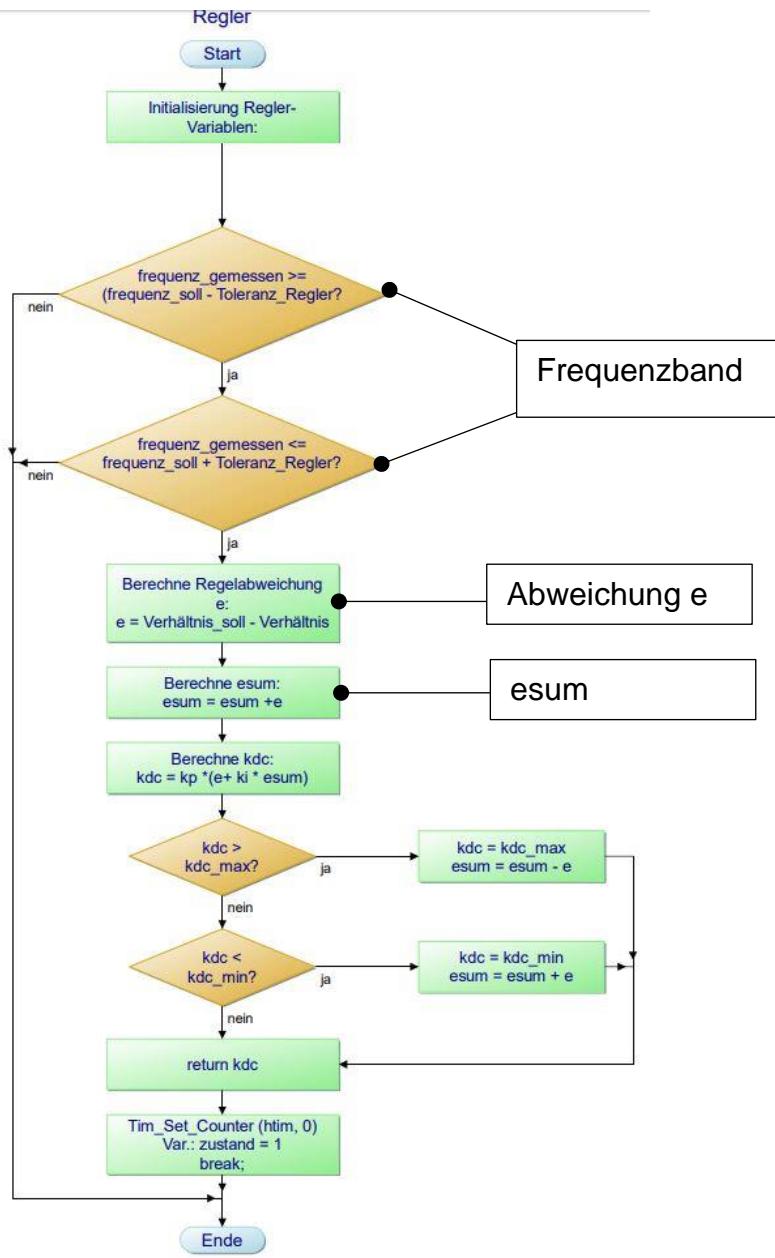


Abbildung 147: Planung Programmablauf der Regelung

Der Programmcode des PI-Reglers ist in der folgenden Abbildung 148 dargestellt. Im Anschluss an die Abbildung werden die einzelnen Zeilen genauer erläutert.

```

void Amplitude_Regler()
{
    if ((frequenz_gemessen >= (frequenz_soll - Toleranz_Regler)) && ((frequenz_gemessen<=frequenz_soll + Toleranz_Regler)))
    {
        e = (verhaeltnis_soll - verhaeltnis);                                // Regelabweichung e zwischen Soll- und Ist-Wert
        esum = esum + e;                                                       // Berechnen Summe der Gesamtfehler esum
        kdc = kp*(e+ki*esum);                                                 // Berechne kdc anhand kp und ki * esum

        if (kdc > kdc_max)                                                    // Wenn kdc größer als kdcmax:
        {
            kdc = kdc_max;                                                     // Kdc auf das Maximum begrenzen
            esum = esum - e;                                                   // Regelabweichung e von Gesamtfehler esum abziehen
        }
        if (kdc < kdc_min)                                                    // Wenn kdc ist kleiner als sein minimum wert:
        {
            kdc = kdc_min;                                                     // kdc nimmt den kleinsten Wert an
            esum = esum + e;                                                   // Regelabweichung e zu Gesamtfehler esum addieren
        }
    }
}

```

Abbildung 148: Programmcode PI-Amplitudenregelung

Die Zeile des eingerahmten Programmcodes

```

„if      ((frequenz_gemessen      >=      (frequenz_soll      -      Toleranz_Regler))      &&
((frequenz_gemessen<=frequenz_soll + Toleranz_Regler)))“

```

bestimmt, dass sich der Schwinger in einem gewissen Bereich befinden muss, bevor der Regler aktiv wird. In diesem Fall sind es +/- 0.5 Hz um die Sollfrequenz von 8,34 Hz. Die Sollfrequenz ändert sich je nach Laser und Standort, während der Toleranzwert immer um +/- 0.5 Hz um die Sollfrequenz bestimmt ist. Ist der Schwinger im Toleranzbereich des PI-Reglers, wird die Regelabweichung e ermittelt aus dem Soll-Verhältnis und dem Ist-Verhältnis, welches im Programm der Frequenzmessung ermittelt, berechnet und schließlich aufsummiert wird. Daraus wird zusammen mit dem P- und I-Anteil des Reglers der Vorfaktor für den Duty-Cycle der H-Brücke (kdc) ermittelt. Sollte der kdc-Wert über den Maximalwert von 0,25 liegen, nimmt der kdc-Wert diesen Maximalwert an und der Fehler wird negativ. Bei Unterschreitung des Minimalwertes für kdc (0,1) wird der kdc-Wert auf den Minimalwert gesetzt.

Für den Regler wird zudem ein Programm zum Überwachen der Reglervariablen angelegt. Diese Übersicht ist in Abbildung 149 aufgelistet. Dabei wird ersichtlich, dass die Zeiten T0 bis T4 recht stabil bleiben. In der gezeigten Skizze muss beachtet werden, dass die Zeit T_{Hell} nicht maßstäblich eingezeichnet ist. Diese wurde durch einen Faktor 1000 geteilt, um sie zusammen mit den anderen Werten betrachten zu können, da die Balken von T_{Dunkel} nicht mit den anderen Werten betrachtet werden müssen.

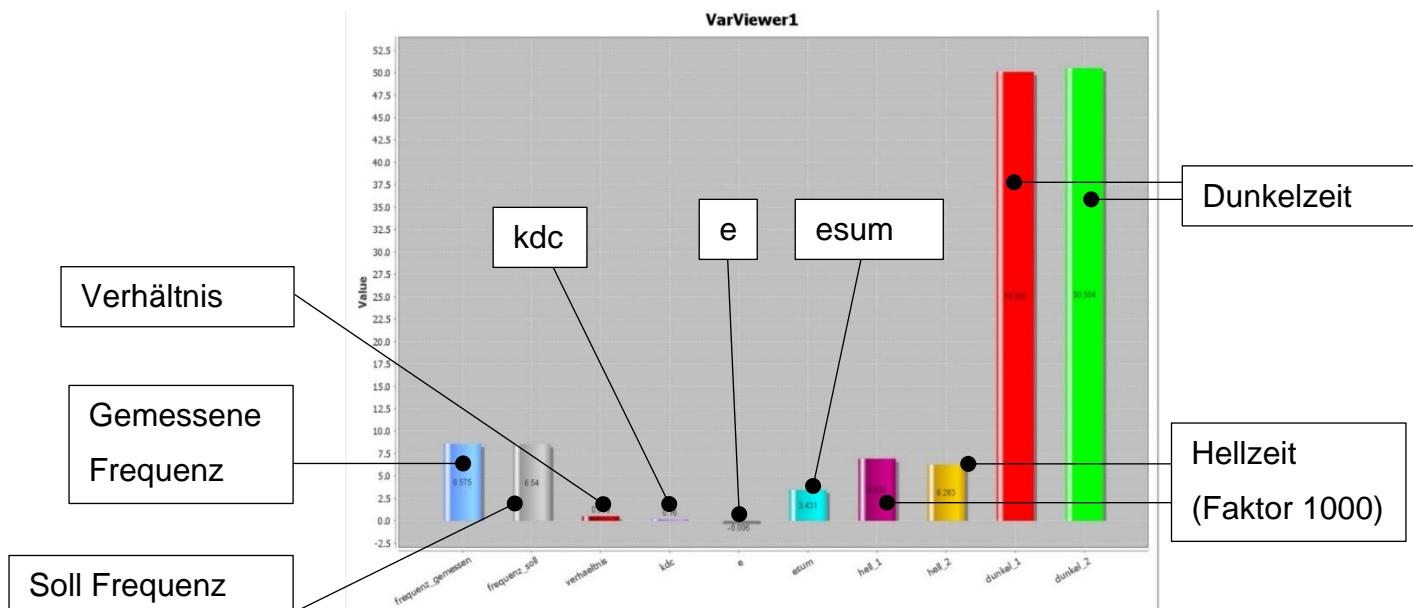


Abbildung 149: Anzeige STM-Studio mit Regelparametern

3.1.3 Vorstellung verwendete Funktionen

Es werden wichtige Funktionen des Druckprogramms vorgestellt, die für den Druck benötigt werden. Dazu werden in Tabelle 20 alle Funktionen aufgelistet.

Tabelle 20: Verschiedene Funktionen für Druckvorgang

Funktion	Beschreibung
delay_us()	Erzeugt Delay in Mikrosekunden
generate_sinus()	Generiert Sinus-Tabelle
Kreis_berechnen()	Kreisberechnung
home_y_achse()	Referenzfahrt y-Achse
home_z_achse()	Referenzfahrt z-Achse
Schrittmotor_y_achse_bewegen()	Verfahren y-Achse während Druck
Z_Achse_ebene_sinken()	Verfahren z-Achse während Druck
Lichtschranke_timer_start()	Start init Lichtschranke-Treiber
Laser_switch_state()	Toggle Laser
schwinger_start()	Starte Schwinger
schwinger_stop()	Stoppe Schwinger
Amplitude_regler()	Regelung der Amplitude
adjust_frequency()	Bestimmt Sollfrequenz beim Start
CalculateRemainTime()	Berechnet verbleibende Druckzeit
init_Z_achse()	Initialisiere z-Achse
Init_Drucker()	Initialisiere Drucker
Icd_init();	Initialisiere Drucker
Icd_clear();	„Clear“ Display
Zebene_runter()	Feinjustierung Höhe Druckbett
DebounceButton()	Taster, bzw. Schalter entprellen
Play_music()	„Welcome music“ (Imperial March)
state_machine()	Ablauf Druckerfunktion
Filter_IIR_1_Order()	Filtert Signal anhand IIR 1. Ordnung
FIR_5_RingBuffer()	Filtert Signal anhand FIR Ring-Buffer
Icd_print_state_cmd()	Display: Willkommensanzeige
Icd_print_temp_cmd()	Display Menüauswahl
Doorgetstate()	Ermittelt Zustand der Tür
Start_Taster_getstate()	Ermittelt Zustand Starttaster
Temperature_print()	Temperaturausgabe
frequency_print()	Displayausgabe aktuelle Frequenz
getAnzahlEbenen()	Displayausgabe noch zu druckende Ebenen
Show_temp_Freq_Time_Layer()	Display verbleibende Zeit, Ebene, Freq., Temperatur

Resonanzfrequenz zum Druckstart einlesen

Als eine weitere Vereinfachung wird ein Einlese-Programm zur Bestimmung der jeweiligen Resonanzfrequenz verwendet (siehe Abbildung 150). Der Benutzer hat dabei über ein Potentiometer die Möglichkeit, die Resonanzfrequenz zwischen 8 und 9 Hertz zu variieren und die Frequenz mit dem besten Schwingverhalten auszuwählen. Der Wert des Potentiometers wird über den Befehl „*PollforConversion*“ ermittelt. Zu dieser Zeit wird nicht gedruckt, weshalb Polling-Befehle keine negativen Auswirkungen haben,

```
*****Adjust_resonance_frequency_at_Begin*****
/*Briefing: use *Potentiometer to adjust the resonance at the beginning
 * ADC Values could varies from 0 to 4095. As factor is the Log of the value given + 1.0f to avoid log infinity
 * 0.00008 is chosen based on estimation that from 0 to 4095 will be approximately distributed like [8..9] graduation
 * it means when the *Potti turned to max it reaches 9 and then decrease by turning in other side until it reaches 8hz.
 */
volatile float adjust_frequency(ADC_HandleTypeDefDef *hadc2, float *frequenz_soll)
{
    volatile float Frequenz_Akt = *frequenz_soll;          // Aktuelle Frequenz soll Frequenz_Soll werden anfangs mit 8.0f
    HAL_ADC_PollForConversion(hadc2, 1); // poll for conversion: Hol ADC-Wert von Potti
    // get the adc value // Erzeuge aus dem ADC-Wert einen Faktor
    volatile float Factor = logf(HAL_ADC_GetValue(hadc2)+1.0f)*0.00008f;
    if ((Frequenz_Akt>8.0) && (Frequenz_Akt<9.0))//Wenn aktuelle Frequenz im Bereich 8 und 9, ikrement um factor umdrehung
    {
        Frequenz_Akt = Frequenz_Akt + Factor;// dann Frequenz_Akt = Frequenz_Akt + Factor;
    }
    else if (Frequenz_Akt>9.0) //Wenn aktuelle Frequenz größer als 9.0 Hz, dann Frequenz_Akt = Frequenz_Akt - Factor;
    {
        Frequenz_Akt = Frequenz_Akt - Factor;
    }
    else if (Frequenz_Akt<8.0) //Wenn aktuelle Frequenz kleiner als 8.0 Hz
    {
        Frequenz_Akt = *frequenz_soll;
    }
    *frequenz_soll = Frequenz_Akt;
    return *frequenz_soll;
}
//Abru function: adjust_frequency(&hadc2, &frequenz_soll);
```

Abbildung 150: Programm zum Justieren der Resonanzfrequenz

Berechnung der restlichen Druckzeit

Mit diesem Programmcode (siehe Abbildung 151) kann die restliche Druckzeit (Remaining Time) berechnet werden. Diese Druckzeit wird berechnet, indem 250 Schwingungen pro Ebene als eine Schicht gezählt werden und die Zeit zum Verfahren der z-Achse eingeplant wird. Anschließend kann die restliche Druckzeit in Minuten angezeigt werden. Zuvor wird in diesem Programm festgelegt, wie viel Ebenen zu drucken sind, um die Druckzeit zu berechnen.

```

/*********************Calculate_Remaining_Time*****/
//function to calculate the remaining time and show on display
float CalculateRemainTime()
{
    //periode der schwingung in s mal 1000 ist in ms
    float ZeitProSchwingung = 1000/frequenz_soll;
    float Delay_z = 1372; //2000 + 980*1.4f:delay time for z axis einstellung pro ebene:
    //1.4ms pro schritt mal 980 schritte + 2000ms delay fur polymer abtropfen
    float Delay_y = 40; // delay time for y: 20schritte and 2ms pro schritt
    int soll_Anzahl_Ebenen = 50; //50 ebene geplante
    int soll_Anzahl_zeilen = 250; // 250 zeilen pro ebene
    int soll_Anzahl_Schwingung = 12520;
    //20 + 250*50: 20 Einschwingungen,250 schwingung pro ebene * anzahl geplante ebenen
    float soll_druckzeit = soll_Anzahl_Schwingung*ZeitProSchwingung+Delay_y*soll_Anzahl_zeilen+Delay_z*soll_Anzahl_Ebenen;
    //gesamte zeit zum drucken
    float ist_druckzeit =Anzahl_Schwingung*ZeitProSchwingung+Delay_y*zahler_Y+Delay_z*Anzahl_Ebenen;
    //zeit vergangen beim druck
    remainTime = (soll_druckzeit - ist_druckzeit)/60000;
    //verbleibende zeit aktualisiert durch 1000 in s durch 60 in Minuten
    float remaintime_prozent = (remainTime/soll_druckzeit)*100;
    return remainTime;
}
//this function can be replaced by using timer that counts at the beginning and calculate the numbers of overflow

```

Abbildung 151: Funktion zur Berechnung der Druckzeit

Referenzfahrt y

Eine Referenzfahrt der y-Achse wird benötigt, um die exakte y-Position des Schwingers zum Start des Drucks zu bestimmen bzw. einzustellen. Hierfür wird zunächst eine Initialisierung nach Datenblatt vorgenommen (siehe Abbildung 152).

```

void home_y_achse()
{
    // Initialisierung Y_achse
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, GPIO_PIN_SET);                                // Reset (inverted) PIN
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_SET);                                // Sleep (inverted) PIN
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_RESET);                             // Enable (inverted) PIN
}

```

Abbildung 152: Programmcode Initialisierung y-Achse

Für die Referenzfahrt wird zunächst die Drehrichtung des Schrittmotors eingestellt und der Swinger entsprechend verfahren, bis der Endschalter erreicht wird und ein Signal an den µC ausgibt. Nach der Signalausgabe wird die Drehrichtung umgekehrt und der Swinger verfährt 400 Schritte zurück. Der Programmcode für diese Anwendung ist in der Abbildung 153 dargestellt.

```

// Referenziert Y-Achse
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_RESET);
do
{
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET); // Direction Pin Reset (-sense rotation)
    delay_us(700); // Mache Anweisung, während Pin Endschalter (PC5) gleich 1 ist (da Öffner)
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_RESET); // STEP PIN High
    delay_us(700); // STEP PIN Low
}
while (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_5) == 1); // wenn gpio pin ist pin endlage y schalter betatigt

// Dann fahre wieder von Endschalter weg zu Startposition für Druck
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5,SET); // Dazu wird Direction Pin umgedreht (auf set)
for(int i=0; i<400; i++)
{
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET); // Rotation Schleife Y_achse 500 Schritte = 10 mm
    delay_us(700); // STEP PIN High
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_RESET); // STEP PIN Low
    delay_us(700);
}
ref_y=1; // Variable stellt dar, dass Referenzfahrt abgeschlossen wurde
}

```

Abbildung 153: Programmcode Referenzfahrt y-Achse

Verfahren der Y-Achse während Druckvorgang

Im Folgenden wird das Programm zum Verfahren des Schwingers in y-Achsrichtung während des Druckvorgangs vorgestellt. Zu Beginn eines Drucks ist ein Einschwingvorgang von 20 Schwingungen eingeplant, bevor der eigentliche Zählvorgang startet (siehe Abbildung 154).

```

void Schrittmotor_y_achse_bewegen()
//Ablauf: Wenn Flag gesetzt und bereits über 20 Schwingungen erledigt, dann verfahren y-Achse um 20 Schritte
{
    if ((flag_schr_y == 1) && (Anzahl_Schwingung >= 20)) // wenn flag schrittmotor y-Achse = 1 und Schwinger i
    {
        zaehler_schr_y++; // Zählt Koordinate der y-Achse

        for(int i=0; i<20; i++) // 20 Schritte fahren
        {
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET); // STEP PIN High
            delay_us(1000);
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_RESET); // STEP PIN Low
            delay_us(1000);
        }

        //Funktion: Wenn y-Koordinate den Wert 250 erreicht hat, soll die Drehrichtung des Schrittmotors umgedreht werden
        int position_schr_y = zaehler_schr_y%250; // Man teilt das aktuelle Feld immer durch 250 (zb. 5
        if (position_schr_y==0) // Wenn y-Koordinate = 250, wird die Drehrichtung des
        {
            HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_5); // Motor umgedreht
        }
        flag_schr_y = 0; // Flag wird wieder auf 0 gesetzt, dadurch kann sich
    }
}

```

Abbildung 154: Programmcode Verfahren der Y-Achse

Nach dem Einschwingvorgang (20 Schwingungen) wird der Zähler für die y-Koordinate nach jeder Reihe mithochgezählt. Für jede neue Position verfährt der Swinger 20 Schritte in y-Richtung. Sobald der Zähler der y-Achse den Wert 250 erreicht hat, wird die Drehrichtung des Motors umgekehrt, die z-Achse verfahren (Aufruf in anderem Programm) und der Zähler beginnt wieder bei null.

Referenzfahrt z-Achse

Die Referenzfahrt der z-Achse wird benötigt, um die exakte Position des Druckbetts zum Start des Drucks zu bestimmen bzw. einzustellen. Da die z-Achse den gleichen Schrittmotor mit entsprechenden Schrittmotortreiber wie die y-Achse verwendet, erfolgt die Referenzfahrt auf die gleiche Weise (siehe Abbildung 155)

```
// Initialisierung z-Achse
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_SET);           // Reset (inverted) PIN
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);           // Sleep (inverted) PIN
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);         // Enable (inverted) PIN

// Rotation Schleife: z-Achse ganz unten und dann bis end schalter
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_RESET);          // Direction Pin (+sense rotation) nach oben
do
{
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET);        // do this while condition ist true
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET);       // STEP PIN High
    delay_us(700);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET);       // STEP PIN Low
    delay_us(700);
} while (HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_2) == 0);           // wenn gpio pin ist pin endlage z schalter betätigt
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7,GPIO_PIN_SET);             // Direction Pin set (-sense rotation) nach unten
for(int i=0; i<1500; i++)                                     // Rotation Schleife Z_achse 20 Schritte sind 0,1 mm
{
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET);        // STEP PIN High
    delay_us(700);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET);       // STEP PIN Low
    delay_us(700);
}
ref_z=1;                                                       // Referenzfahrt
}
```

Abbildung 155: Programmcode Referenzfahrt z-Achse

Nach Erreichen des Endschalters verfährt das Druckbett 1500 Schritte nach unten und bleibt in dieser Position, bis der Benutzer vor dem Start das Druckbett feinjustiert (siehe Funktion „Justierung Höhe bei Start“).

Justierung Höhe bei Start

Das Druckbett muss im späteren Betrieb auf die Höhe der Polymer-Füllung des Druckbechers angepasst werden. Aus diesem Grund ist es notwendig, das Druckbett nach Einlegen des Polymer-Bechers manuell verfahren zu können. In diesem Projekt wird hierfür der „Blue Push Button“ des Nucleo-Boards verwendet. Nach der Referenzfahrt der z-Achse soll das Druckbett bei jedem Tastendruck um eine Schicht nach unten fahren (20 Schritte/ 0,1 mm). Der Code wird in der Abbildung 156 vorgestellt.

```

void Zebene_runter() // z-Achse soll eine Schicht nach unten
{
    if ((Ebenerunter==1)&&(ref_z==1))
    {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7,GPIO_PIN_SET); // Direction Pin set (-sense rotation) unten

        for(int i=0; i<20; i++) // Rotation Schleife Z_achse 20 Schritte 0,1 mm
        {
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET); // STEP PIN High
            delay_us(700);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET); // STEP PIN Low
            delay_us(700);
        }

        Ebenerunter =0;
        //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
    }
}

```

Abbildung 156: Funktion Druckbett manuell absinken

Verfahren der z-Achse während Druckvorgang

Es wird das Verfahren der z-Achse während eines Drucks vorgestellt. Das besondere Verhalten bei dieser Achse ist das vollständige Eintauchen des Druckbetts in das Polymer. Dies wird benötigt, um die gesamte Oberfläche des Druckbetts mit weiterem Polymer zu benetzen. Anschließend soll das Druckbett wieder nach oben fahren und das überschüssige Polymer abtropfen lassen. Der Programmcode wird in der Abbildung 157 vorgestellt.

```

int Z_Achse_ebene_sinken()
{
    if (flag_schr_z == 1)
    {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7,GPIO_PIN_SET); // Direction Pin set (-sense rotation) nach unten

        for(int i=0; i<500; i++) //Rotation Schleife Z_achse 20schritte 0,1 mm (Fahr 500 Schritte nach unten)
        {
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET); // STEP PIN High
            delay_us(700);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET); // STEP PIN Low
            delay_us(700);
        }
        //Druckbett fährt hier also 500 Schritte runter in das Polymer

        // Fahr Druckbett wieder hoch auf Druckposition
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7,GPIO_PIN_RESET); // Drehrichtung wieder umkehren (nach oben)
        for(int i=0; i<480; i++) // Fährt zuvor 500 Schritte runter und jetzt wieder 480 Schritte nach oben
        {
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET); // STEP PIN High
            delay_us(700);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET); // STEP PIN Low
            delay_us(700);
        }
        //Warte 2 sekunden und lass Polymer abtropfen
        HAL_Delay(2000);

        ++Anzahl_Ebenen; // Zählt die z-Koordinate um +1
    }
    flag_schr_z = 0;
}

return Anzahl_Ebenen;

```

Abbildung 157: Programmcode zum Verfahren der z-Achse in Polymer

Sobald die z-Achse auf die nächste untere Schicht verfahren soll, wird das Druckbett um 500 Schritte (2,5 mm) in das Polymer eingetaucht. Anschließend fährt das Druckbett wieder 480 Schritte (2,4 mm) nach oben. Dort angekommen wird ein Delay

von 2 Sekunden durchgeführt. Dies wird dazu verwendet, um das überschüssige Polymer auf dem Druckbett abtropfen zu lassen.

3.1.4 Main-Programm und State-Machine

Die State Machine stellt einen endlichen Zustandsautomaten dar und beinhaltet die Realisierung eines Steuerungskonzepts mit einer Reihe von Zuständen, um einen fehlerfreien Betriebsablauf zu gewährleisten. Der Programmcode befindet sich in der main-Funktion und die verschiedenen Etappen des Druckens sind in der Abbildung 158 dargestellt.

```
/* Private typedef -----
/* USER CODE BEGIN PTD */
typedef enum { Start, Init, Drucken, Ende} States_t;
/* USER CODE END PTD */
```

Abbildung 158: Definition enumrate states

Zunächst wird bei geöffneter Tür eine Referenzfahrt durchgeführt, beim Betätigen des blauen Push-Buttons des Nucleo fährt das Druckbett eine Ebene herunter

```
/* USER CODE BEGIN 3 */
*****User_State_Machine*****
*****State Machine Function*****
```

```
void state_machine()
{
    States_t State_Manager = Start;

    switch(State_Manager)
    {

        case Start:
            while (Doorgetstate() == 0)          // Erfüllt, wenn Tür noch offen ist
            {
                Laser_switch_state(0);         // Mach Laser aus
                if (ref_z == 0)                // und noch keine Referenzfahrt erledigt ist
                {
                    home_z_achse();           // Dann home mal die z-Achse
                }
                Zebene_runter();             // Fahr eine Schicht runter pro druck auf taster
                lcd_print_state_cmd();      //Display function for start drucker
            }

            State_Manager = Init;
            break;
    }
}
```

Abbildung 159: Status Zustand Tür offen

Wenn die Tür geschlossen und der Start-Taster gedrückt wird, werden sämtliche Druckerfunktionen initialisiert (siehe Abbildung 160). Nach der Initialisierung verfahren die y- und die z-Achse und der Druck beginnt. Nach dem Druckvorgang und dem

Erreichen von 250 Ebenen wird das Schwingen des Schwingers eingestellt und der Druck ist somit beendet. Während des Drucks sind alle Prozesse auf dem Display sichtbar, um den aktuellen Stand zu verfolgen. Nach Beendigung des Drucks und mit offener Türe kann der Vorgang von vorn durchgeführt werden.

```
// nachdem z-Achse ajustiert ist und Tür z geschlossen wird und start taster gedrückt ist
while ((DoorGetstate() == 1) && (Start_Taster_getstate() == 1))
{
    case Init:
        lcd_clear(); //Clear display
        lcd_print_temp_cmd(); //function display to initialise and heat polymer
        Init_Drucker(); // ruf function init drucker
        lcd_clear(); //Clear display
        Temperature_print(); // show current tempature
        if (Polymerready == 1) // check if polymer 60 grad erreicht hat
            State_Manager = Drucken; //switch zum drucken
        break;
    case Drucken:
        lcd_clear(); //Clear display
        Schrittmotor_y_achse_bewegen(); // y verfahren
        Z_Achse_ebene_sinken(); // eben sinken
        Show_temp_Freq_Time_Layer(); //show all parameter drucker temperatur, frequenz, remain time and ebene
        State_Manager = Ende;
        break;
    case Ende:
        if (Anzahl_Ebenen == 250) // Wenn gedruckt und Anzahl Ebenen 250 erreicht hat, dann
        {
            schwinger_stop(); // Mach Schwinger aus
            lcd_clear(); //Clear display
            lcd_print_End_cmd(); //show Fertig display
        }
        State_Manager = Start; //spring wieder in case start
        break;
    }
}
```

Abbildung 160: Ablauf bei Schließen der Türe und Drücken Start-Taster

3.2 Regelungstechnik

Mit Hilfe der Amplitudenregelelung wird die Einhaltung einer möglichst gleichbleibenden Frequenz erreicht, um damit einen konstanteren Druckprozess zu ermöglichen. Die wichtigsten Größen des Regelkreises sind in der Tabelle 21 dargestellt. Im Folgenden wird geprüft, inwieweit der implementierte Regler die Eigenfrequenz einhalten kann.

Tabelle 21: Amplitudenregelung Parameter

Größe aus Regelkreis	Parameter in 3D-Drucker
Stellgröße	Vorfaktor kdc an H-Brücke
Regelgröße	Amplitude der Sinusschwingung
Sollgröße	Verhältnis Hell- zu Dunkelzeit
Regelabweichung	Differenz aus Soll- und Ist-Verhältnis

Eine PLL (Phase-Lock-Loop, Phasenregelschleife) wird oft verwendet, wenn Systeme eine gleichmäßige Phasendifferenz benötigen. In diesem Projekt würde die PLL den Phasenabstand zwischen der anregenden Sinusschwingung der H-Brücke und dem mechanischen Schwinger auf einer definierten Phasendifferenz festlegen. Eine PLL wird in diesem System aktuell nicht verwendet, da der Amplitudenregler eine ausreichend gleichmäßige Regelung ermöglicht.

3.2.1 Regelverhalten der Amplitudenregelung

Es wird eine Messung zur Genauigkeit der Regelgüte aufgestellt. Dazu wird gemessen, welche Frequenzen der Schwinger innerhalb von 10 Schwingungen annimmt. Die Messung wurde dabei durchgeführt, indem am Display (siehe Abbildung 161) der aktuelle Frequenzwert angezeigt und anschließend notiert wurde.

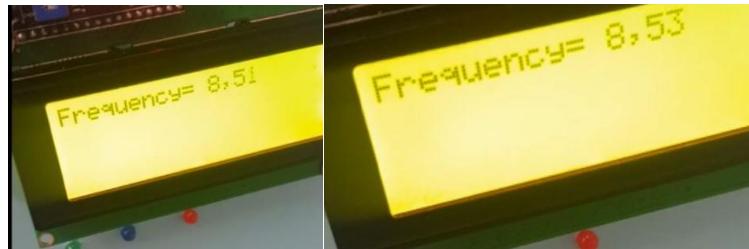


Abbildung 161:: Ablesen der aktuellen Ist-Frequenz

Die notierten Frequenzwerte werden anschließend in einem Diagramm abgebildet. Dieses ist in der Abbildung 162 dargestellt. Bei diesem Durchlauf sollte die Sollfrequenz genau 8,52 Hertz betragen. Dabei ist ersichtlich, dass der Regler leicht um die Eigenfrequenz schwankt. Diese Schwankungen sind allerdings recht minimal und werden nicht als Problem betrachtet.

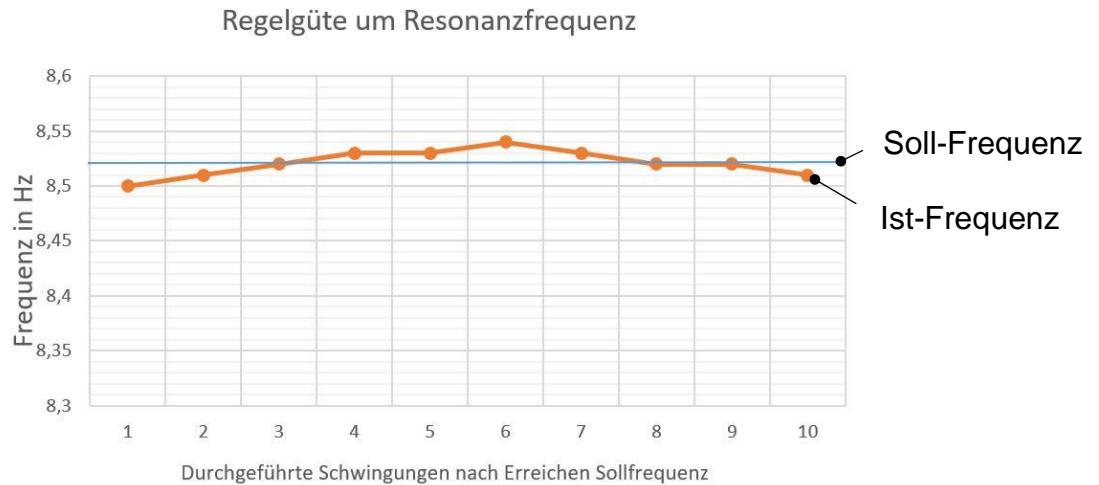


Abbildung 162: Regelgüte um Resonanzfrequenz

Zudem werden die einzelnen Variablen des Reglers im Programm STM-Studio beobachtet. Dabei können, wie in Abbildung 163 dargestellt und in Tabelle 22 beschrieben, die einzelnen Werte der Variablen angezeigt werden. In der folgenden Abbildung werden aus Gründen der besseren Übersichtlichkeit (da andere Größenordnung) die Zeiten der Dunkelzeit nicht miteingefügt. Dabei ist zu beobachten, wie der Regler jeweils nachregelt und sich die Fehlersumme des I-Anteils verändert.

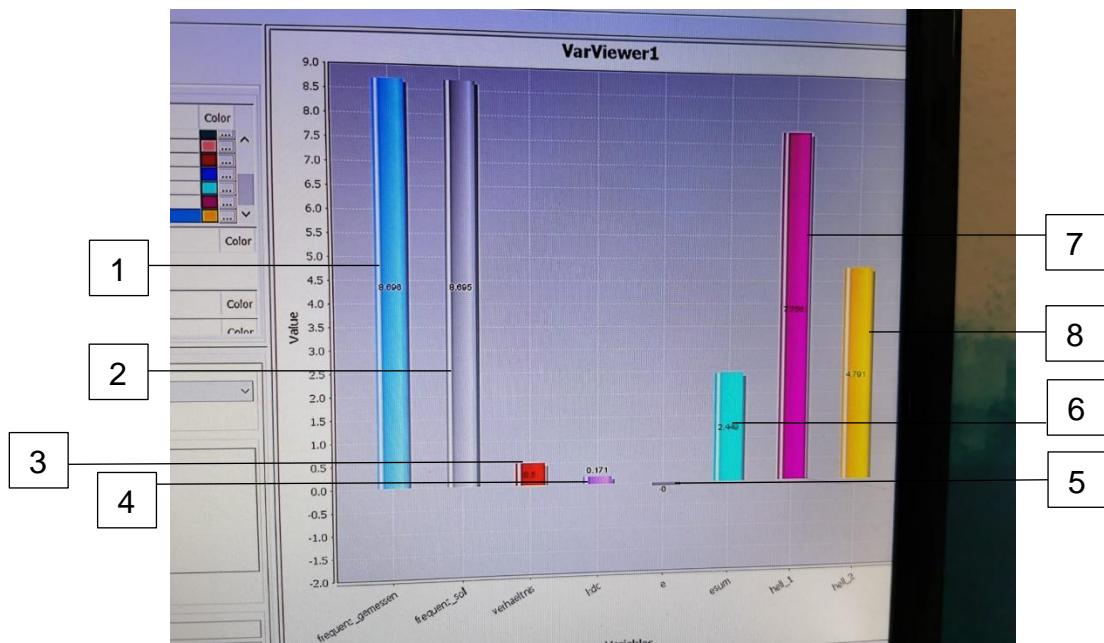


Abbildung 163: Messwertaufnahme Schwinger

Die Beschreibung der Positionsnummern ist in der folgenden Tabelle dargestellt.

Tabelle 22: Beschreibung der Werte aus Abbildung 163 Fehler! Verweisquelle konnte nicht gefunden werden. Abbildung 163 Fehler! Verweisquelle konnte nicht gefunden werden.

Pos.	Bezeichnung	Beschreibung
1	Ist-Wert Frequenz	Ist die gemessen Frequenz aktuell.
2	Soll-Wert Frequenz	Ist die im Programm eingestellte Frequenz.
3	Ist-Verhältnis Hell/ Dunkel	Dort wird angezeigt wie sich das Verhältnis zu Hell und Dunkel verhält.
4	Stellwert kdc	Der Stellwert reguliert nach, sodass die Frequenz gleichbleibt.
5	Regelabweichung e	Bei Regelabweichung wird der Wert angezeigt wie hoch aktuell die Abweichung beträgt.
6	Summe Regelabweichung	Die einzelnen Regelabweichungen werden aufsummiert. Regelabweichung zwischen Soll- und Istwert
7	Ist-Hellzeit 1	Soll das gleiche Verhältnis haben wie Hellzeit 2.
8	Ist-Hellzeit 2	Soll das gleiche Verhältnis haben wie Hellzeit 1.

3.2.3 Filterbedarf und Notwendigkeit von Beobachtern

Es werden keine zu verwendenden Filter oder Beobachter vorgesehen. In den bisherigen Tests gab es keine Probleme aufgrund ungefilterter Signale.

3.3 Inbetriebnahme/ Test Hardware

Die Mechanik, Sensorik und Aktorik wird für die Inbetriebnahme vorbereitet. Hierzu wird zunächst eine Laserhülse mit einem Gewicht von 17 Gramm angefertigt. In der Abbildung 164 ist der Einbau des simulierten Lasers mit Gewicht dargestellt. Diese Hülse wird verwendet, wenn der rote Laser nicht zur Verfügung steht.

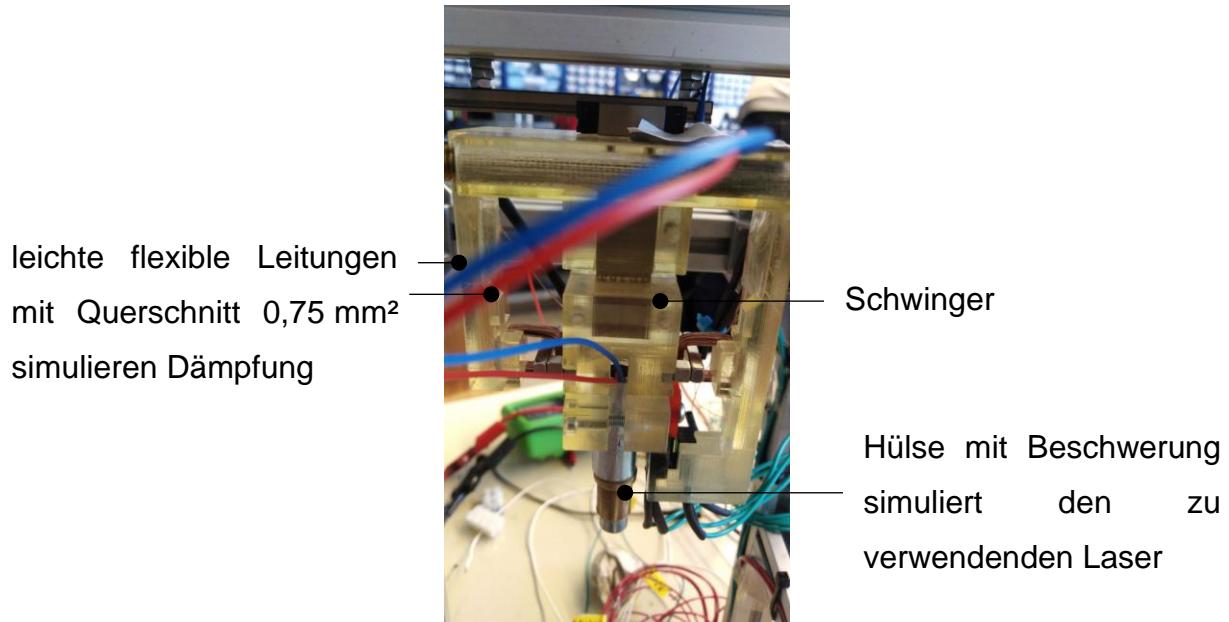


Abbildung 164: Einbau Laserdiode mit Gewicht

In der Tabelle 23 wird die Eigenfrequenz des Schwingers mit einem simulierten Gewicht von 17 Gramm ermittelt. Die Ergebnisse sind in der Tabelle 23 dargestellt.

Tabelle 23: Tabelle Eigenfrequenz und Gewicht

Objekt	Gewicht	Eigenfrequenz
Laserdiode mit Hülse	17 Gramm	8,23 Hz
Ohne Laserdiode und ohne Hülse	7 Gramm	15,6 Hz

3.3.1 Beschreibung des Aufbaus zum Ausrichten des Lasers

Nachdem der rote Laser verfügbar ist, kann die zuvor verwendete Hülse wieder ausgebaut und der Laser eingebaut werden. Erneut wird dabei die Resonanzfrequenz bestimmt und justiert. Das Gewicht des Lasers beträgt 17,7 Gramm und weicht damit leicht von dem vorherigen Versuch ab. Es wird eine neue Eigenfrequenz von 8,50 Hz ermittelt.

Zum Ausrichten des Lasers wird dieser an ein 3,3 Volt und ein GND-Pin geschalten. Daraufhin beginnt der Laser zu leuchten. Dies ist in folgender Abbildung symbolisiert.

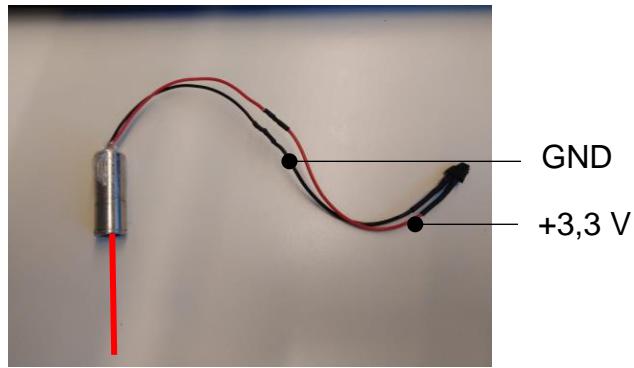


Abbildung 165: LED-Laserdiode mit 3,3 Volt

Anschließend wird der Laser in die Konstruktion eingebaut und ausgerichtet. Der Laserabstand, also die Höhe, bzw. Entfernung des Lasers zum Druckfeld (siehe Abbildung 166) soll einen Abstand von ca. 15 cm haben. In der nachfolgenden Abbildung ist dieses Schema dargestellt.

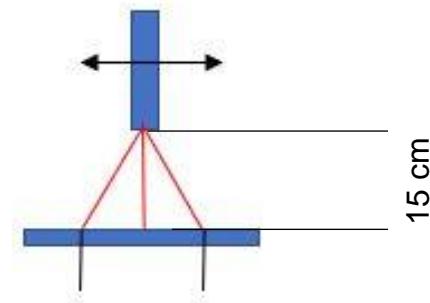


Abbildung 166: Ausrichten des Lasers

3.3.2 Abbilden einer Linie mit der Laserdiode

Die Abbildung 167 zeigt die erste erzeugte Linie auf dem Druckbett mit dem roten Laser. In dieser Phase wird getestet, ob die einzelnen Komponenten einen ausreichend großen Strich erzeugen können und dabei das Druckbett nicht verlassen. Dazu wird die Laserdiode dauerhaft eingeschalten und der Schwingvorgang gestartet. Das Prinzip ist in der Abbildung 168 dargestellt.

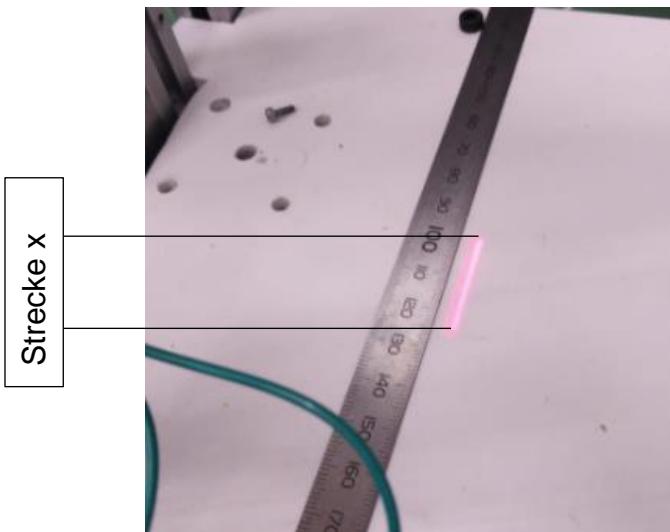


Abbildung 167: Messen der maximalen Druckfläche

Prinzip: Streckenmessung

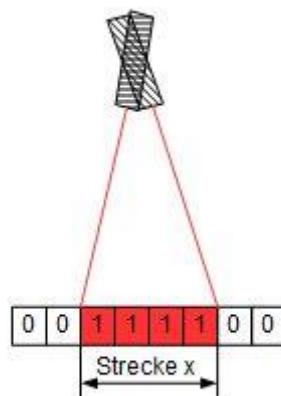


Abbildung 168: Prinzip Messung

3.3.3 Y-Achse und z-Achse

Es soll die y- und z-Achse des Druckers geprüft werden. Die Positioniergenauigkeit der y-Achse kann versuchsweise über eine eingespannte Messuhr kontrolliert werden. Dabei wird eine Messuhr an das Gehäuse geschraubt und der Schrittmotor um 200 Schritte in Richtung Messuhr verfahren. Anschließend wird kontrolliert, inwieweit der theoretische Messwert von dem realen Messwert abweicht. Theoretisch, bei absoluter Präzision des Schrittmotors und Geradlinigkeit der Führungen sollte ein Messwert von 1 Millimeter gemessen werden. Aufgrund von Zeitmangel konnte dieser Test nicht durchgeführt werden.

Zudem wurden für die Bewegung von y- und z-Achse keine Rampen zur Bewegung der Motoren verwendet. Diese könnten in einer weiteren Optimierungsphase implementiert werden.

3.4 Bedienungsanleitung

Die fertige Bedienungsanleitung befindet sich im Anhang.

4 Testdruckergebnisse

In der folgenden Abbildung ist der erste Versuch eines vorläufigen Kreises mit einem roten Laser dargestellt. Zusätzlich zu sehen ist, dass der gewünschte rote Kreis realisiert wurde. Mit dieser Erkenntnis konnte der nächste Schritt eingeleitet werden, um den richtigen Druck zu erzeugen.

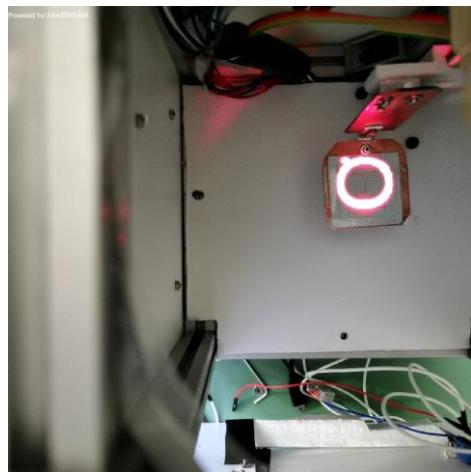


Abbildung 169: Langzeitbelichtung von Druck

In der Abbildung 170 ist der erste Testdruck mit dem 3D-Drucker mit blauem Laser erfolgt. Die Form eines Kreises ist zu erkennen. Außerdem hat der Kreis einen Innen- sowie einen Außendurchmesser. Zu diesem Zeitpunkt fehlt allerdings noch das Fenster.

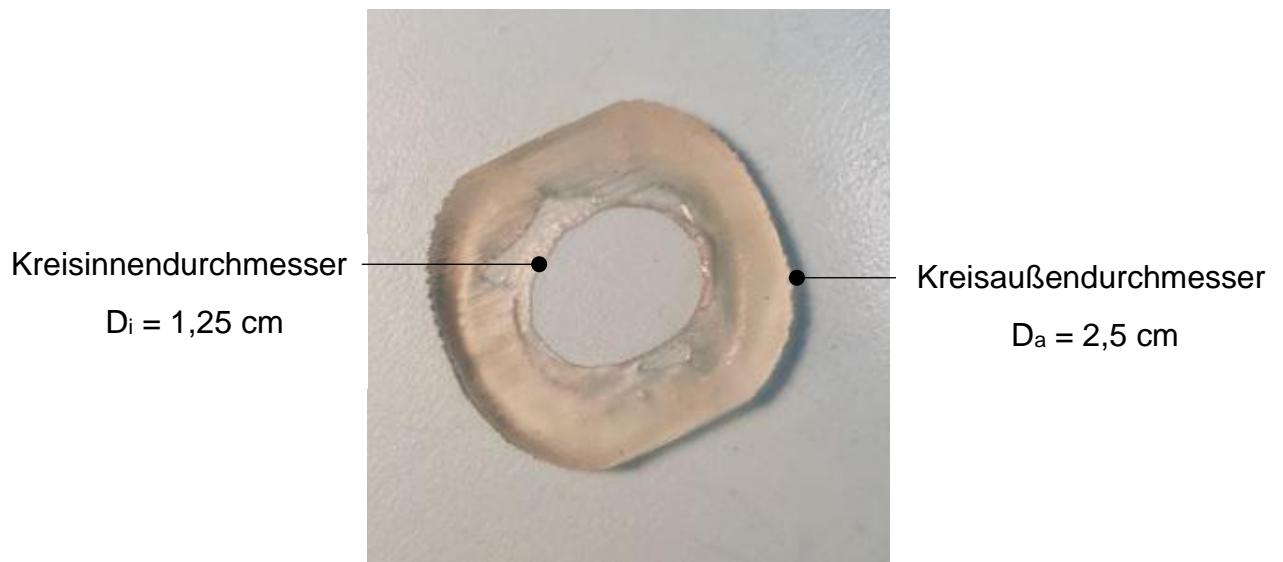


Abbildung 170: Erstes Druckteil

In der nachfolgenden Abbildung 171 wurde ein Druck durchgeführt, um einen Kreis mit einem Fenster zu realisieren. Bei diesem Druck kam es leider nicht zu dem gewünschten Ergebnis. In dem Druck wurde zwar ein Fenster erzeugt und anfangs auch ein Kreis, doch später kam es zum Fehler. Das Ergebnis dieses Druckes ist nachfolgend zu sehen.

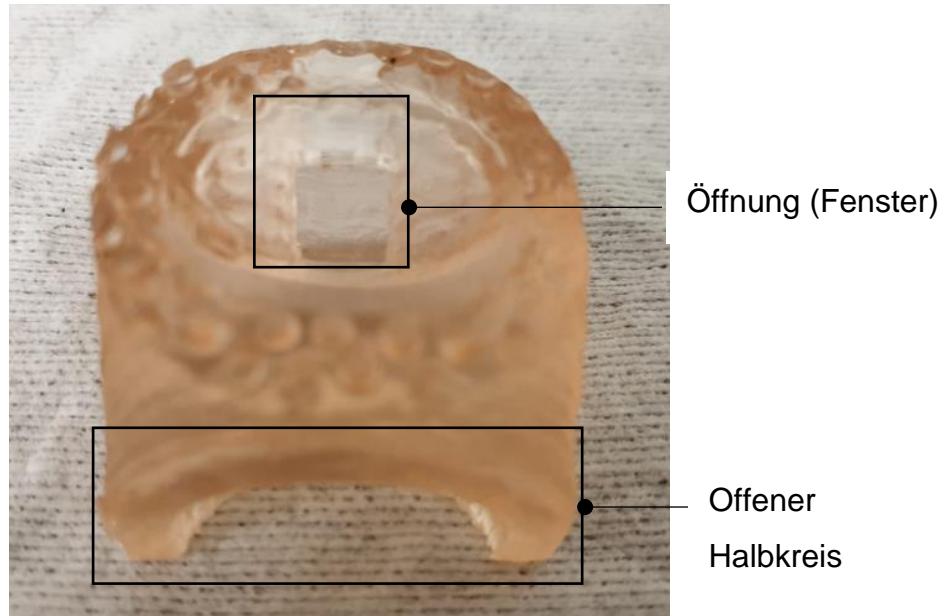


Abbildung 171: Druckversuch mit Fenster

4.1 Beschreibung und Bewertung der Druckergebnisse

Beim ersten Testdruck bestand das Ziel lediglich in einem Druck eines Kreisrings mit einem Außendurchmesser von 2,5 cm und einem Innendurchmesser von 1,25 cm. Da diese Erwartung erfüllt wurde, ist der erste Testdruck als Erfolg anzusehen (siehe Abbildung 170). Es fällt jedoch eine leichte Ovalität auf, welche Optimierungspotential aufweist.

Der zweite Testdruck sollte einen vollen Hohlzylinder mit einem Fenster erzeugen. Leider wurde dieser Körper nicht zufriedenstellend gedruckt, da sich nach ca. 15 Minuten der Programmablauf änderte und eine ungewünschte Form entstand (siehe Abbildung 171). Nach einem zunächst gedruckten Kreis entstanden zwei entgegengesetzte Halbkreise. Ein Fenster wurde im Druck vorgesehen, jedoch durch die unerwünschte Form nicht an der richtigen Stelle. Da der erste Testdruck reibungslos verlief, ist die Ursache im veränderten Softwarecode durch Zufügung des Fensters zu vermuten.

4.2 Nicht umgesetzte Abhilfemaßnahmen

Aufgrund des zweiten Druckergebnisses muss der Programmcode zur Erzeugung des Fensters überarbeitet werden. Leider wurde dies aufgrund von Zeitmangel nicht umgesetzt und ein weiterer Testdruck war so nicht möglich.

5 Zusammenfassung

Im Folgenden wird eine komplette Zusammenfassung dargestellt, die sowohl technische Herausforderungen und daraus resultierende Erfolge als auch Schwierigkeiten und Einschätzungen zum Projekt sowie einen entsprechenden Ausblick beinhaltet.

5.1 Projekterfolge

Zum Ende des Projekts konnten eine Vielzahl von Funktionen nachgewiesen werden, um erfolgreich einen effizienten Druck nachweisen zu können. Hierzu zählen die Funktionalität der Lichtschranke, das positionsgenaue Verfahren der y- und z-Achse, der gesamte mechanische Aufbau des Schwingers mit der Auslegung der Spulen, die entsprechende Sinuserzeugung als Kraftanregung, das Einschwingverhalten und die dazugehörige Amplitudenregelung für die Sinusschwingung sowie die Entwicklung eines Algorithmus für den Druck des Kreises. Die Gesamtheit dieser Funktionalitäten führte schließlich zum ersten Testdruck, welcher jedoch lediglich einen Kreisring ohne Fenster beinhaltete.

Zunächst wurde der Drucker mit einer Spannungsquelle verbunden und der Becher mit flüssigem Polymer wurde unter dem Druckbett positioniert bei entsprechender Justierung der Z-Achse. Nach dem Start des Druckvorgangs durch den Start-Taster referenzierten sich die Y- und die Z-Achse. Der Swinger begann seinen Einschwingprozess und setzte seine Soll-Frequenz von ca. 8,34 Hz fest. In diesem Toleranzbereich startete der Amplitudenregler die Regelung und hielt diese Frequenz konstant aufrecht. Nach dem Einschwingprozess (20 Schwingungen) startete der Laser und belichtete die Voxel entsprechend des Programmablaufs bei gleichzeitiger Positionsänderung der Y- und Z-Achse, um einen Kreisring zu drucken. Das Verfahren des Druckbetts an der Z-Achse resultierte in einer Abwärtsbewegung, um das Polymer durch die Löcher des Druckbetts auf die Oberfläche zu generieren und eine Aufwärtsbewegung, um die angestrebte Position zu erreichen. Da alle Funktionen fehlerfrei verliefen, wurde ein Kreisring gedruckt und der erste Testdruck ist als Erfolg zu verzeichnen. Lediglich die ovale Form birgt Optimierungspotential.

Der zweite Testdruck erfolgte nach der Einbettung eines Programmcodes für das Fenster. Der Ablauf ähnelte dem ersten Testdruck. Alle Funktionen verliefen fehlerfrei,

bis ab einen bestimmten Zeitpunkt der Laser keinen Kreis mehr abfuhr, sondern zwei entgegengesetzte Halbkreise generierte. Das Fenster wurde eingebettet, jedoch gleicht die Form keinem Zylinder mehr. Da alle Funktionen sowohl hier als auch beim ersten Testdruck reibungslos abliefen, wurde der Fehler in der veränderten Software vermutet. Eine Optimierung diesbezüglich wurde jedoch aus Zeitgründen nicht mehr durchgeführt.

5.2 Wesentliche technische Herausforderungen und ihre Lösungen

Die größten technischen Herausforderungen in diesem Projekt waren der mechanische Aufbau des Schwingers mit der Auslegung der Feder und der Spulen, die Erstellung eines funktionierenden Layouts für die Leiterplatte, die Erzeugung der Kraftanregung für den Swinger, die Regelung des Schwingverhaltens und die Erstellung des Algorithmus für den Kreisdruck.

Beim mechanischen Aufbau des Schwingers wurde zunächst die Entscheidung getroffen nur eine Lichtschranke für das System zu verwenden. Die Ausrichtung der Magnete musste so gewählt werden, dass sich ein nahezu homogenes Magnetfeld senkrecht zur Spule bildet und diese bei Bestromung eine Kraft auf den Swinger einleitet. Zusätzlich musste die Lichtschranke in diesen Swinger integriert und die benötigte Feder für den Schwingvorgang ausgelegt und erprobt werden.

Die Erstellung des Layouts für die Leiterplatte stellte einen Meilenstein in den Anfangsphasen des Projektes dar. Die Wichtigkeit dieses Meilensteins wurde zum Anfang deutlich gemacht und nahm einige Zeit in Anspruch, da das Layout wiederholt überarbeitet bzw. komplett neu gemacht werden musste, da die Funktion sonst möglicherweise nicht gegeben sei. Nach mehrmaligem Überarbeiten wurde das Platinenlayout schlussendlich für gut befunden und konnte entsprechend gefertigt werden.

Für die Kraftanregung des Schwingers stellten die Stoßanregung zum richtigen Zeitpunkt oder eine Sinusanregung über den gesamten Bereich eine Option dar. Da eine Stoßanregung die Erfüllung der Druckgeschwindigkeit gemäß Anforderungsliste nicht erreichen konnte, wurde zunächst eine Sinustabelle erstellt, welche später in das Softwareprogramm eingepflegt und erprobt wurde, um eine kontinuierliche Sinusanregung einzubetten.

Eine weitere Herausforderung bestand in der Regelung des Schwingverhaltens, da zunächst nicht feststand, ob ein Amplitudenregler bzw. eine Phasenregelschleife (PLL) oder sogar beide verwendet werden müssen. Es wurde angenommen, dass ein Amplitudenregler möglicherweise ausreichend ist, um den Schwinger auf eine konstante Frequenz zu halten. Die Herausforderung bestand hierbei in der Ermittlung des richtigen Proportional- und Integralanteils des Amplitudenreglers. Hierfür waren mehrere Tests notwendig, da der Schwinger aufgrund falscher PI-Werte aus dem Takt kam und den Schwingvorgang beendete. Nach Ermittlung der richten PI-Werte konnte das gewünschte Verhältnis von Dunkelzeit zu Periodendauer geregelt werden und die Frequenz des Schwingers blieb somit konstant, wodurch eine PLL nicht notwendig war.

Die letzte Schwierigkeit beinhaltete die Erstellung des Algorithmus für einen Kreisdruck. Die Idee bestand zunächst darin die zu druckende Form in ein 250x250 Raster einzuteilen und den Mittelpunkt der x- und y-Koordinate im Ursprung festzulegen. Da der Innen- und Außenradius des Kreises sowie die y-Koordinate durch die Schrittweite des Schrittmotors bekannt sind, sollte die x-Koordinate durch den Satz des Pythagoras errechnet werden und später entschieden werden, ob diese Koordinate belichtet wird oder nicht. Für jede x-Koordinate sollte somit ein Interrupt ausgelöst und abgefragt werden, ob die Koordinate den Wert 1 für „belichten“ oder den Wert 0 für „nicht belichten“ enthält. Diese Vorgehensweise wurde jedoch aufgrund des hohen Aufwands und der schwierigen Umsetzbarkeit verworfen. Die neue Lösung ist jedoch nicht gänzlich anders als die vorige Lösung. Die Ursprungskoordinaten für x und y finden weiterhin Anwendung, auf welche ein imaginäres Koordinatensystem gelegt wird. Durch dieses Koordinatensystem wird nun für jedes Kästchen der Abstand dxy zum Mittelpunkt, entsprechend der x- und der y-Koordinate mithilfe des Satzes von Pythagoras berechnet. Dadurch kann ein Minimal- und Maximalabstand bestimmt werden, in dem der Laser jeweils eingeschaltet sein soll. Um keine Verluste aufgrund eines verspäteten Signals der Lichtschranke zu generieren, wurde das Muster auf 280 (x-Koordinate) zu 250 (Y-Koordinate) erweitert.

5.3 Bewertung als Semesterprojekt

Das Semesterprojekt „Stereolithographie-3D-Drucker“ ist wohl das mit Abstand komplexeste und fordernste Projekt in der gesamten Studienlaufbahn. Es verbindet

die Teildisziplinen Mechanik, Elektronik und Informatik zu einem kompletten Projekt und jede Teildisziplin erfordert hohe Kenntnisse, umfangreiches Engagement und eine zielstrebige Arbeitsweise. Schon zu Beginn wurde festgestellt, dass hohe Anforderungen herrschen und bestimmte Zusammenhänge erst gründlich erschlossen werden müssen, um eine hinreichende Bearbeitung liefern zu können.

Die Layouterstellung für die Leiterplatte benötigte zu Beginn des Projektes eine schnelle Bearbeitung, welche sich als schwierig herausstelle, da kein Gruppenmitglied ausreichende Kompetenzen mit der Software „Eagle“ aufwies. Mit der Zeit stellte sich ebenfalls heraus, dass die Gruppenmitglieder unterschiedliche Ziele im Hinblick auf dieses Projekt besitzen. Zusätzlich sind fünf unterschiedliche Köpfe nicht unbedingt immer innovativer und schneller, da unterschiedliche Sichtweisen und Charakteristika von fünf Personen zwangsläufig unter Einwirkung von Stress zu Spannungen innerhalb der Gruppe führen. Das Schlüsselwort zum Erfolg eines Projektes ist Kommunikation und diese war in der Projektgruppe leider nicht ausreichend vorhanden. Dadurch arbeiteten teilweise mehrere Personen an der gleichen Aufgabe bzw. setzten bestimmte Aufgaben anders um als erwartet, da vorige Arbeiten den anderen Gruppenmitgliedern nicht ausreichend vermittelt wurden.

Das Projekt im Allgemeinen vermittelt viele Kompetenzen auf den unterschiedlichen Gebieten der Mechatronik und ist somit als notwendig und wichtig in diesem Studium anzusehen. Aufgrund der umfangreichen Aufgabe und den komplexen Zusammenhängen war es jedoch leider an manchen Stellen nötig Mindestanforderungen zu erfüllen, anstatt lange ingenieurtechnische Arbeitsschritte zu verwenden, um ein „perfektes Produkt“ zu entwickeln. Es stellte sich schon Zufriedenheit ein, wenn der Drucker „irgendwie“ funktioniert, da Stress und mangelnde Zeit stets ein Begleiter waren.

Die technischen Herausforderungen gepaart mit den menschlichen Komplikationen brachten sicherlich jedes einzelne Gruppenmitglied an seine Grenzen. Jedoch wurden auch Kompetenzen auf dem Gebiet der Mechatronik erweitert und sowohl technische als auch menschliche Komplikationen sind in der Industrie ebenfalls nicht zu vermeiden. Somit dient das Projekt als gute Vorbereitung für das Berufsleben.

Zudem wurde das Projekt trotz aller Widrigkeiten erfolgreich beendet, da der Drucker einen zufriedenstellenden Kreisring drucken konnte und die vorgegebenen Funktionen

erfüllte. Das Projekt ist somit trotz aller Schwierigkeiten als Erfolg zu bezeichnen und hat jedes Gruppenmitglied in vielfacher Hinsicht vorangebracht.

5.4 Ausblick

Aufgrund der Komplexität birgt dieses Projekt eine Menge Spielraum für Optimierung. Die resonante X-Achse sorgt bereits für einen schnellen Druckablauf, jedoch ist die maximale Druckgeschwindigkeit noch nicht erreicht. Es wurde bereits eine Sinusanregung verwendet, welche deutlich schneller ist als die Stoßanregung und Softwareoptimierungen auf dem hochleistungsfähigem Microcontroller stellten weitere Verbesserungen an. Möglicherweise könnte die Masse des Schwingers leicht verringert bzw. generell der Swinger leicht umkonstruiert werden, um geringfügig schnellere Geschwindigkeiten zu erreichen.

Zudem wurde in diesem Projekt lediglich ein Kreisring gedruckt. Die anfängliche Idee für den Kreisdruck könnte weiterhin betrachtet und ausgearbeitet werden, um beliebige Formen zu drucken, indem das erstellte Raster in die Software integriert wird und dadurch Variantenvielfalt möglich ist.

Hochwertige 3D-Drucker erleichtern Produktionsprozesse für die Industrie erheblich und Optimierungen auf diesem Gebiet sind zukunftsträchtig und zudem aktuelle Forschungsthemen, welche in den nächsten Jahren nicht weniger Relevanz aufweisen werden.

6 Verzeichnis

6.1 Abbildungsverzeichnis

Abbildung 1: Entwurf 3D-Drucker	1
Abbildung 2: Regelkreis Heiztemperatur	3
Abbildung 3: Regelkreis Taktfrequenz x-Achse	4
Abbildung 4: Regelkreis Verhältnis Dunkelzeit zu Periodendauer	5
Abbildung 5: Konzept y-Achse.....	8
Abbildung 6: Trägerplatte y-Achse	8
Abbildung 7:Darstellung z-Achse.....	9
Abbildung 8: Aufbau der x-Achse (Schwinger)	9
Abbildung 9: Konzept 1 mit zwei Gewindestangen.....	11
Abbildung 10: Konzept 2: 3D-Drucker mit Riemenantrieb und Gewindestange.....	11
Abbildung 11: Aufbau und Position der Lichtschranke (Lichtschranke aus [1])	12
Abbildung 12: Darstellung Bewegungsablauf Schwinger	13
Abbildung 13: Messpunkte der Lichtschranken	13
Abbildung 14: Signal, wenn Endschalter als Schließer ausgelegt	14
Abbildung 15: Verwendete Lichtschranke [1].....	14
Abbildung 16: Gewöhnliche Betriebsbedingungen der Lichtschranke [1]	14
Abbildung 17: Verwendeter Taster [2]	15
Abbildung 18: Endschalter der y- und z-Achse [3].....	15
Abbildung 19: Verwendete Führung Misumi SEBZ8 -70 [4].....	16
Abbildung 20: Präzision der Misumi-Führungen SZB8-100 und SZB8-70 [4]	16
Abbildung 21: Berechnung der möglichen Abweichung	17
Abbildung 22: kleinster möglicher Schrittwinkel der Schrittmotoren	17
Abbildung 23: Genauigkeit von 12-, 16- und 32-Bit ADCs.....	18
Abbildung 24: Auflösung Schrittmotor y- und z-Achse.....	18
Abbildung 25: Zeitliche und geometrische Auflösung der y- und z-Achse	19
Abbildung 26: Zeitliche und geometrische Auflösung der x-Achse	19
Abbildung 27: Darstellung und Formel Lorentzkraft [5].....	20
Abbildung 28: Aufbau Aktor x-Achse	21
Abbildung 29: Prinzip Aktoraufbau	21
Abbildung 30: Leiter im Magnetfeld	22
Abbildung 31: Auslegung Elektromagnete.....	23
Abbildung 32: Darstellung Baugruppen und Spannungsversorgung	26

Abbildung 33: Schaltregler Berechnungsformel (aus [6]).....	28
Abbildung 34: Berechnung Schaltregler Widerstandswert R2	28
Abbildung 35: Schaltung Spannungsversorgung mit 3,3, 5 und 12 Volt	30
Abbildung 36: Berechnung R_{Sense} -Pins	30
Abbildung 37: Stromkreis Schrittmotortreiber	31
Abbildung 38: Ansteuerung der H-Brücke	32
Abbildung 39: Stromkreis der Lichtschranken	33
Abbildung 40: Spannungsversorgung Heizstromkreis	34
Abbildung 41: Berechnen der Erwärmung des MOSFETs.....	34
Abbildung 42: Spannungsversorgung Stromkreis Laser.....	35
Abbildung 43: Nucleo Schalt- und Kontaktplan.....	35
Abbildung 44: Leiterbahnbreite [8].....	36
Abbildung 45: Platine Layout	37
Abbildung 46 Top Layer Platine.....	39
Abbildung 47 Bottom Layer Platine	39
Abbildung 48: Logic Interface Timing Diagramm [9]	40
Abbildung 49: Regelkreis Temperatur	46
Abbildung 50: Überlegung Amplitudenregelung	47
Abbildung 51: CAD Darstellung des 3D - Drucker	49
Abbildung 52: Darstellung Schwinger	49
Abbildung 53: Schwinger in Aufbauphase	49
Abbildung 54: Festlegen einer geeigneten Blende	50
Abbildung 55: Prinzip Antrieb Schwinger und Lichtschranke.....	50
Abbildung 56: Einspannung Druckbecher und z-Achse.....	51
Abbildung 57: Gefertigte z-Achse	51
Abbildung 58: 3D-Drucker mit Gehäuse	52
Abbildung 59: Tür mit Türschalter.....	52
Abbildung 60: Test der Lichtschranke	55
Abbildung 61: Funktionsgenerator mit Schwinger	57
Abbildung 62: Kontrolle H-Brücke.....	58
Abbildung 63: Test H-Brücke mit LEDs	58
Abbildung 64: Kontaktplan.....	58
Abbildung 65: Funktionstest Schrittmotor	59
Abbildung 66: PWM-Test.....	59

Abbildung 67: Kontaktplan Display	60
Abbildung 68: Implementierung I ² C-Display	60
Abbildung 69: Verlötetes Board bereit zum Test	61
Abbildung 70: Links: Stromkreis Heizung und rechts: Temperaturmessung	61
Abbildung 71: STM32 Cube MX Pinbelegung	65
Abbildung 72: STM32 Hardware Pinbelegung [11].....	65
Abbildung 73: I ² C_LCD (siehe auch [12])	66
Abbildung 74: i2c_im Main-Programm.....	67
Abbildung 75: LCD-Ausgabe: Startbildschirm.....	67
Abbildung 76: LCD-Ausgabe Menüauswahl	67
Abbildung 77: Printf_Bibliothek.....	68
Abbildung 78:: Uart mit Main-Funktion und Funktion printf.....	68
Abbildung 79: Printf_Ausgabe	69
Abbildung 80 Schaltung Thermistor.....	69
Abbildung 81: Ausgabe der aktuellen Temperatur am Display	70
Abbildung 82: Ausgabe der Temperatur an Konsole	70
Abbildung 83: Ausgabe Temperatur STM-Studio	70
Abbildung 84: Ausgabe Temperatur über 60°C	71
Abbildung 85: Ausgabe Temperatur über 60 °C STM Studio	71
Abbildung 86: Ausgabe Temperatur über 60 °C Konsole	71
Abbildung 87: Programm Temperatur (Link aus [13])	72
Abbildung 88: Schaltung Lichtschranke.....	73
Abbildung 89: Links High-Signal und rechts Low-Signal der Lichtschranke	73
Abbildung 90: Input Capture Callback Funktion (siehe auch [14])	74
Abbildung 91: Periode, Pulsbreite und Duty Cycle	75
Abbildung 92: Ergebnis Input Capture mit PWM-Generierung	75
Abbildung 93: Ausgabe der erzeugten Frequenz am Oszilloskop	76
Abbildung 94: Frequenz-Messung bei einer beliebigen Schwingung mit Werten	76
Abbildung 95: Darstellung Frequenz am Oszilloskop bei langsamer Schwingung ...	76
Abbildung 96: Frequenz schnellere Schwingung.....	77
Abbildung 97: Getestete schnellere Schwingung am Oszilloskop	77
Abbildung 98: IIR Filter Implementierung (Link aus [15])	78
Abbildung 99: FIR Filter erste Lösung Implementierung.....	79
Abbildung 100: FIR Filter Ring Buffer Lösung (Link aus [15])	80

Abbildung 101: Stepper-Treiber der y-Achse.....	81
Abbildung 102: Stepper-Treiber der z-Achse.....	82
Abbildung 103: PWM Optional Stepper	82
Abbildung 104: PWM Stepper Test	83
Abbildung 105: Schrittmotor 10 Schritte Test	83
Abbildung 106: H-Brücke Test Code	84
Abbildung 107: H-Brücke Programm-Test mit LEDs	85
Abbildung 108: PWM H-Brücke Test	85
Abbildung 109: High Signal H-Brücke Test.....	86
Abbildung 110: Testaufbau mit Audio-Verstärker	87
Abbildung 111: Sinuserzeugung mittels Tonemitter	87
Abbildung 112: Matlab Darstellung Sinus- und Rechtecksignal.....	88
Abbildung 113: Ergebnis erster Versuch Sinusfrequenz mit der Lichtschranke.....	89
Abbildung 114: Erzeugung Sinus mit PWM [16]	90
Abbildung 115: Sinuswelle-Simulation über Wechsel des DC erzeugen	90
Abbildung 116: Erzeugung Sinus Tabelle.....	91
Abbildung 117: Sinus Callback Funktion	91
Abbildung 118: Sinus Varviewer.....	92
Abbildung 119: Sinus Varviewer mit DIR	92
Abbildung 120: Formel Sinus DAC (siehe auch [17])	93
Abbildung 121: Frequenz Sinuswelle DAC (siehe auch [17])	93
Abbildung 122 Sinuswelle DAC Code	94
Abbildung 123: Sinus Frequenz 120 KHz DAC	94
Abbildung 124 Dreieckswelle Code	95
Abbildung 125: DAC Dreieckform (siehe auch [17])	95
Abbildung 126: Dreieckswelle DAC	95
Abbildung 127: Variablendecl. von Sinus, Lichtschranke (Frequenz) und Regler	96
Abbildung 128: Deklaration der Variablen (Zustände und Flags)	97
Abbildung 129: Deklaration der Variablen für Zähler, Kreis und Temperatur.....	97
Abbildung 130: Hinterlegung der Sinustabelle im Programm	98
Abbildung 131: PAP zur Sinuserzeugung.....	99
Abbildung 132: Interrupt zur Sinuserzeugung.....	99
Abbildung 133: Ziel: Roter Kreis mit folgenden Maßen	100
Abbildung 134: Symboldarstellung Kreisberechnung	100

Abbildung 135: Rechnung Abstand Mittelpunkt	101
Abbildung 136: Prinzip Berechnung Kreis Programm.....	102
Abbildung 137: Ablaufplan Kreisprogramm	103
Abbildung 138: Kreis-Programm in Software.....	104
Abbildung 139: Darstellung Periodendauer und Lichtschranke	105
Abbildung 140: Beschreibung Hell- und Dunkelzeit.....	105
Abbildung 141: Programmablaufplan Frequenzmessung	108
Abbildung 142: Interrupt Bedingung und Rising-Edge Erkennung.....	109
Abbildung 143: Falling Edge Erkennung.....	109
Abbildung 144: Programmierung Regelung Heizung.....	110
Abbildung 145: Benötigte Aktorkraft	111
Abbildung 146: Initialisieren der Variablen.....	112
Abbildung 147: Planung Programmablauf der Regelung.....	113
Abbildung 148: Programmcode PI-Amplitudenregelung.....	114
Abbildung 149: Anzeige STM-Studio mit Regelparametern.....	115
Abbildung 150: Programm zum Justieren der Resonanzfrequenz.....	117
Abbildung 151: Funktion zur Berechnung der Druckzeit.....	118
Abbildung 152: Programmcode Initialisierung y-Achse	118
Abbildung 153: Programmcode Referenzfahrt y-Achse.....	119
Abbildung 154: Programmcode Verfahren der Y-Achse	119
Abbildung 155: Programmcode Referenzfahrt z-Achse.....	120
Abbildung 156: Funktion Druckbett manuell absinken.....	121
Abbildung 157: Programmcode zum Verfahren der z-Achse in Polymer.....	121
Abbildung 158: Definition enumrate states	122
Abbildung 159: Status Zustand Tür offen	122
Abbildung 160: Ablauf bei Schließen der Türe und Drücken Start-Taster	123
Abbildung 161:: Ablesen der aktuellen Ist-Frequenz	124
Abbildung 162: Regelgüte um Resonanzfrequenz	125
Abbildung 163: Messwertaufnahme Swinger.....	125
Abbildung 164: Einbau Laserdiode mit Gewicht	127
Abbildung 165: LED-Laserdiode mit 3,3 Volt.....	128
Abbildung 166: Ausrichten des Lasers	128
Abbildung 167: Messen der maximalen Druckfläche	129
Abbildung 168: Prinzip Messung	129

Abbildung 169: Langzeitbelichtung von Druck.....	130
Abbildung 170: Erstes Druckteil.....	130
Abbildung 171: Druckversuch mit Fenster	131

6.2 Tabellenverzeichnis

Tabelle 1: Anforderungsliste zum 3D-Drucker	2
Tabelle 2: Auflistung und Beschreibung der mechanischen Baugruppen.....	7
Tabelle 3: Beschreibung Schwinger x-Achse	9
Tabelle 4: Spannungs- und Strombedarf der Bauteile	27
Tabelle 5: Stromverbrauchsumme und Leistung	28
Tabelle 6: Linearregler Modell UA78M33 (siehe [7])	29
Tabelle 7: Beschreibung Pin-Eingänge Schrittmotortreiber A4982.....	41
Tabelle 8: Beschreibung Pins der H-Brücke [10]	41
Tabelle 9: Output Truth Table [10].....	42
Tabelle 10: Planung Inbetriebnahmekonzept	45
Tabelle 11: Regelgrößen in PI-Amplitudenregelung	47
Tabelle 12: Erste Inbetriebnahme der Baugruppen	48
Tabelle 13: Relevante Toleranzen	53
Tabelle 14: Inbetriebnahmeplanung	62
Tabelle 15: Erstellung der Algorithmen.....	98
Tabelle 16: Berechnung Abstand dxy mit Pythagoras	101
Tabelle 17: Beschreibung der Abbildung 140	106
Tabelle 18: Berechnung der Zeitpunkte für Frequenzberechnung.....	106
Tabelle 19: Anpassung Parameter H-Brücke	111
Tabelle 20: Verschiedene Funktionen für Druckvorgang	116
Tabelle 21: Amplitudenregelung Parameter	123
Tabelle 22: Beschreibung der Werte aus Abbildung 163.Abbildung 163	126
Tabelle 23: Tabelle Eigenfrequenz und Gewicht	127

6.3 Formelverzeichnis

Formel I	32
Formel II	43

7 Quellen

Literaturverzeichnis

- [1] RS COMPONENTS GMBH: *Optek Gabel-Lichtschranke OPB460T11, Puffer, Open-Collector mit 10 K Pull-Up-Widerstand Ausgang, Schraubmontage, 5 Pin.* URL <https://de.rs-online.com/web/p/gabel-lichtschranke/4550874/> – Überprüfungsdatum 2020-02-10
- [2] CONRAD ELECTRONIC SE: *TE Connectivity.* URL <https://asset.conrad.com/media10/add/160267/c1/-/en/000701749DS01/datenblatt-701749-te-connectivity-1825910-2-drucktaster-24-vdc-005-a-1-x-ausein-tastend-1-st.pdf> – Überprüfungsdatum 2020-02-10
- [3] PREMIER FARNELL LIMITED: *Mikroschalter, Miniatur, Hebel,.* URL http://www.farnell.com/datasheets/2340369.pdf?_ga=2.180138412.945511001.1581529073-489411319.1581529073 – Überprüfungsdatum 2020-02-10
- [4] MISUMI CORPORATION: *Structure and Precision of Linear Guides.* URL https://us.misumi-ec.com/pdf/fa/2012/p1_0469.pdf – Überprüfungsdatum 2020-02-06
- [5] ELEKTRO-ARCHIV.DE: *LORENTZKRAFT.* URL <http://www.elektro-archiv.de/archiv/l/lorentzkraft/> – Überprüfungsdatum 2020-02-10
- [6] MOUSER ELECTRONICS, Inc: *LM2672 SIMPLE SWITCHER.* URL <http://www.ti.com/lit/ds/symlink/lm2672.pdf> – Überprüfungsdatum 2019-11-20
- [7] FARNELL.COM: *UA78M33CDCY.* URL <https://de.farnell.com/texas-instruments/ua78m33cdcy/v-reg-2vdo-0-5a-3-3v-4sot223/dp/2296031> – Überprüfungsdatum 2019-11-11
- [8] DIGI-KEY ELECTRONICS: *PCB Trace Width Calculator.* URL <https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-pcb-trace-width> – Überprüfungsdatum 2020-02-06
- [9] ALLEGRO MICROSYSTEMS, L. L.C.: *A4982 : DMOS Microstepping Driver with Translator And Overcurrent Protection.* URL <https://www.allegromicro.com-/media/Files/Datasheets/A4982-Datasheet.ashx>. – Aktualisierungsdatum: 2014-05-07 – Überprüfungsdatum 2020-02-06

- [10] INFINEON TECHNOLOGIES AG: *IFX9201SG : 6 A H-Bridge with SPI*. URL
https://www.infineon.com/dgdl/Infineon-IFX9201SG-DS-v01_01%20EN.pdf?fileId=5546d4624cb7f111014d2e8916795dea. –
Aktualisierungsdatum: 2015-02-15 – Überprüfungsdatum 2020-02-06
- [11] DOC.RIOT-OS.ORG/: *STM32 Nucleo-F303RE*. URL https://doc.riot-os.org/group__boards__nucleo-f303re.html – Überprüfungsdatum 2020-10-20
- [12] CONTROLLERSTECH.COM: *LCD displays*. URL <https://controllerstech.com/lcd-20x4-using-i2c-with-stm32/> – Überprüfungsdatum 2019-12-20
- [13] ELEGOO INC: *Elegoo Einsteiger Kit für UNO*. URL
<file:///C:/Users/domin/Downloads/Elegoo%20Super%20Starter%20Kit%20for%20UNO%20V1.0.18.12.24-Deutsch.pdf> – Überprüfungsdatum 2020-02-10
- [14] CONTROLLERSTECH.COM: *Input Capture*. URL <https://controllerstech.com/how-to-use-input-capture-in-stm32/> – Überprüfungsdatum 2019-12-20
- [15] WILHELM, Stefan: *Digitale Filter in C für Embedded-Anwendungen : IIR Tiefpass-Beispiel*. URL <http://atwillys.de/content/cc/digital-filters-in-c/> – Überprüfungsdatum 2020-02-10
- [16] THOMAS S-: <https://www.mikrocontroller.net/topic/173340>. URL
<https://www.mikrocontroller.net/topic/173340> – Überprüfungsdatum 2020-02-06
- [17] STMICROELECTRONICS: *AN3126 Application note : Audio and waveform generation using the DAC in STM32 microcontrollers*. URL
https://www.st.com/content/ccc/resource/technical/document/application_note/05/fb/41/91/39/02/4d/1e/CD00259245.pdf/files/CD00259245.pdf/jcr:content/translations/en.CD00259245.pdf. – Aktualisierungsdatum: 2017-05-11 –
Überprüfungsdatum 2020-02-10

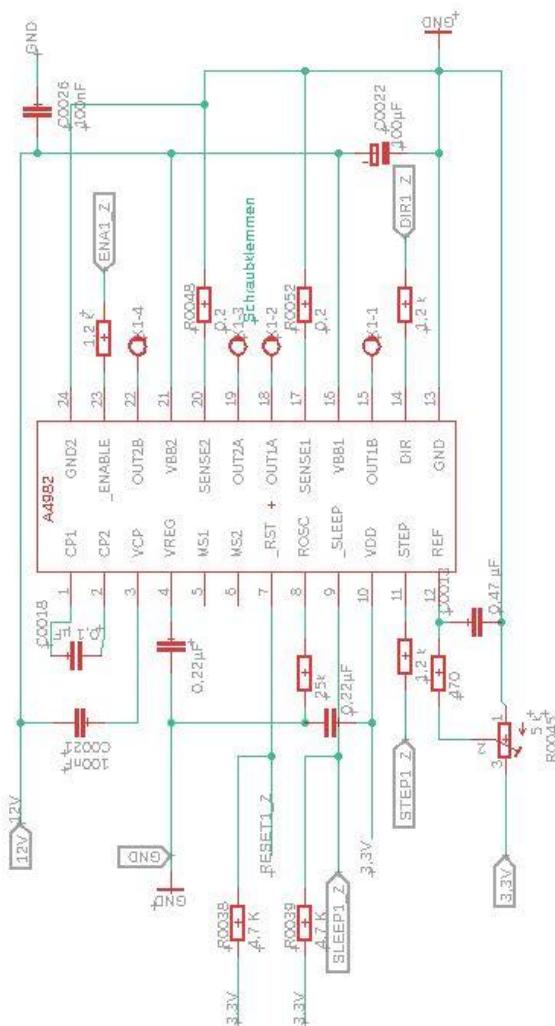
8 Anlagen

Bauteilliste	I
Schaltpläne	II
Bedienungsanleitung	IX
Technische Zeichnungen	XII

Bauteilliste

Qty	Value	Device	Package	Parts
1		C-EUC0805	C0805	C0011
11		PINHD-1X2	1X02	JP1, JP2, JP3, JP
7		PINHD-1X3	1X03	JP8, JP9, JP10, J
3		PINHD-1X5	1X05	JP5, JP24, JP25
9		R-EU_R0603	R0603	R0008, R0014, R00
1		R-TRIMM3296Y	RTRIMM3296Y	R0046
2		SCHRAUBKLEMME_2X	W237-102	X2, X27
2		SCHRAUBKLEMME_4X	W237-4	X1, X22
1	0	R-EU_R0603	R0603	TEST_R
2	0,1 µF	C-EUC0603	C0603	C0002, C0018
3	0,1µF	C-EUC0603	C0603	C0012, C0024, C00
4	0,2	R-EU_R2512	R2512	R0048, R0049, R00
4	0,22µF	C-EUC0603	C0603	C0004, C0005, C00
1	0,33µF	C-EUC0603	C0603	C0001
1	0,47 µ	C-EUC0805	C0805	C0010
1	0,47 µF	C-EUC0603	C0603	C0013
3	1 k	R-EU_R0603	R0603	R0012, R0015, R00
8	1,2 k	R-EU_R0603	R0603	R0009, R0010, R00
1	1,2k	R-EU_R0603	R0603	R0042
4	1,5K	R-EU_R0603	R0603	R0001, R0063, TES
2	1,5k	R-EU_R0603	R0603	R0061, R0062
1	100	R-EU_R0603	R0603	R0026
5	100nF	C-EUC0603	C0603	C0006, C0015, C00
2	100nF	C-EUC0805	C0805	C0016, C0041
1	100µF	CPOL-EU140CLH-1010	140CLH-1010	CIN1
3	100µF	CPOL-EUUD-8X10	UD-8X10_NICHICON	C0009, C0014, C00
2	10nF	C-EUC0603	C0603	C0003, C0023
1	1N5819HW-7-F	1N5819HW-7-F	SOD-123_MINI-SMA	D1
2	1k	R-EU_R0603	R0603	R0003, R0013
3	2,7k	R-EU_R0603	R0603	R0004, R0028, R00
2	25k	R-EU_R0603	R0603	R0006, R0019
1	2N7002,215	2N7002,215	SOT23-3	Q1
2	33nF	C-EUC0805	C0805	C0007, C0008
2	4,7 K	R-EU_R0603	R0603	R0038, R0039
2	470	R-EU_R0603	R0603	R0023, R0037
1	47µ/H	SRR1240-680M	INDUCTOR	L1
1	5 k	R-TRIMM3296Y	RTRIMM3296Y	R0045
3	6,8k	R-EU_R0603	R0603	R0002, R0005, R00
2	68	R-EU_R0603	R0603	R0024, R0025
1	68µF	CPOL-EU140CLH-1010	140CLH-1010	COUT1
2	82	R-EU_R0603	R0603	R0029, R0032
2	A4982_STEPDRIVER	A4982_STEPDRIVER	TSSOP24	A4982_Y, A4982_Y1
1	IFX9201SG	IFX9201SG	BSOP-12	H_X
1	LM2672MX-ADJ/NOPB	LM2672MX-ADJ/NOPB	SOIC-8	U12
1	MURS360BT3G	MURS360BT3G	SMB	D12
1	NUCLEO303RE	NUCLEO303RE	DIL40	STM
1	RAPC712X	RAPC712X	SWC_RAPC712X	J1
1	UA78M33	UA78M33	SOT-223	U5
1	frei lassen	R-EU_R0603	R0603	R0018

Schaltpläne



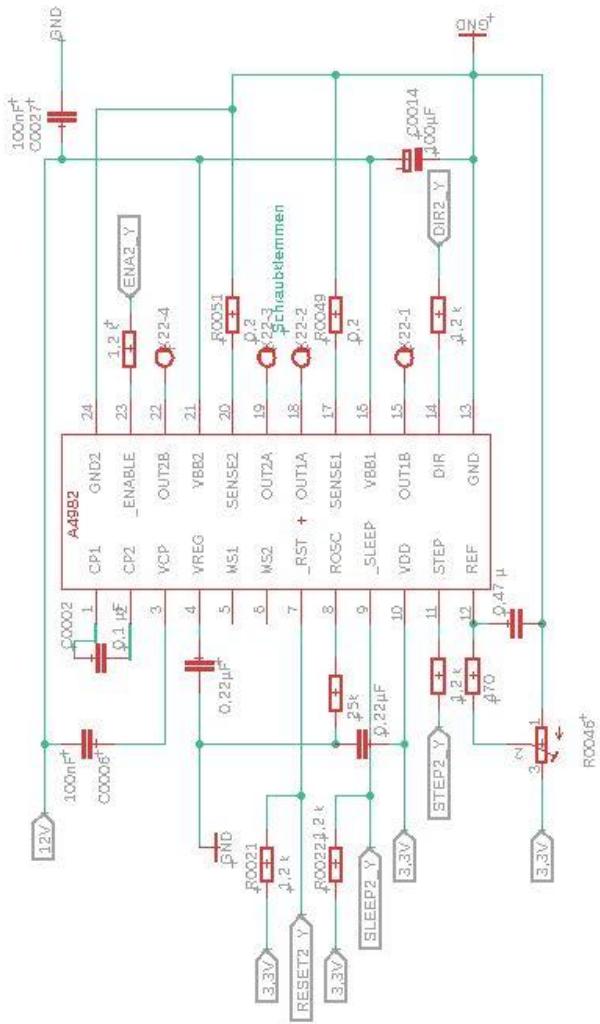
Projekt 3D-Drucker Gruppe C^t

TITLE: Schrittmotortreiber z-Achse

Document Number:

REV:

Date: 30.11.2019 Sheet: 1/7



Projekt 3D-Drucker Gruppe C

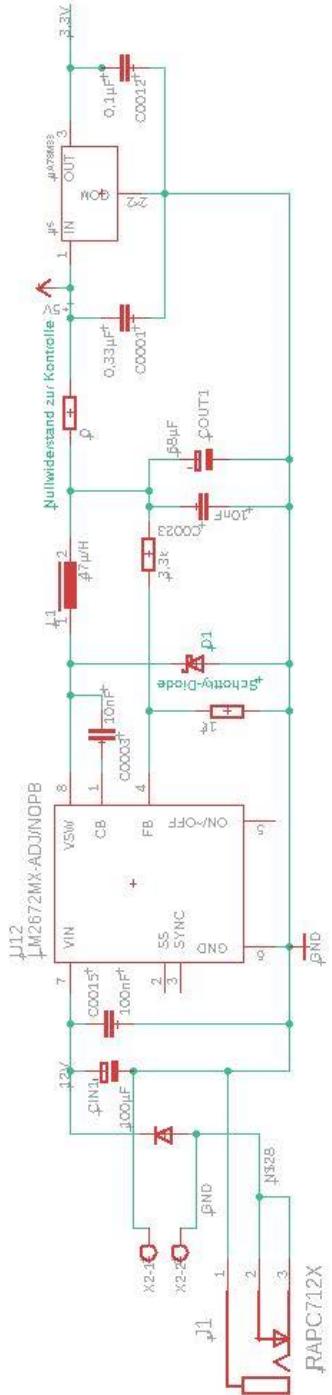
TITLE

Document Number:

REV.

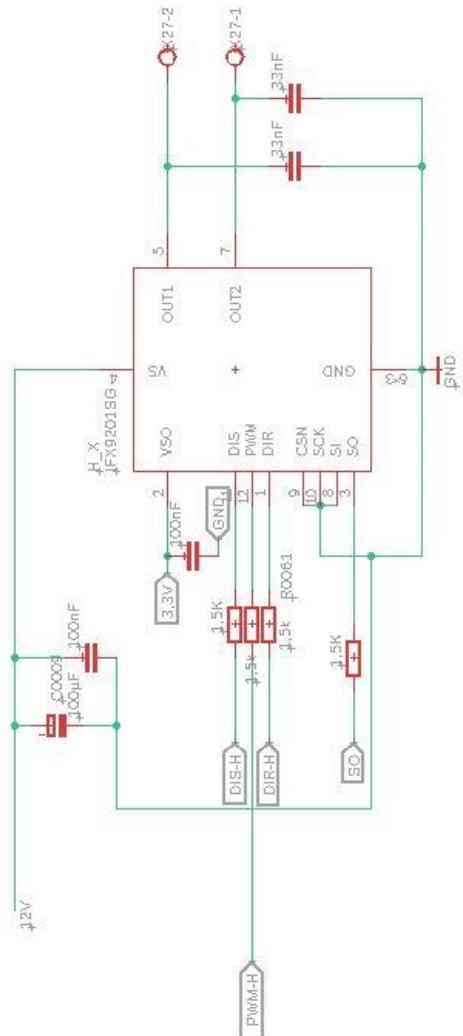
Date: 30.11.2019

217



Projekt 3D-Drucker Gruppe C
TITLE: Spannungsversorgung

Document Number:		REV:
Date: 30.11.2019	Sheet: 2/7	

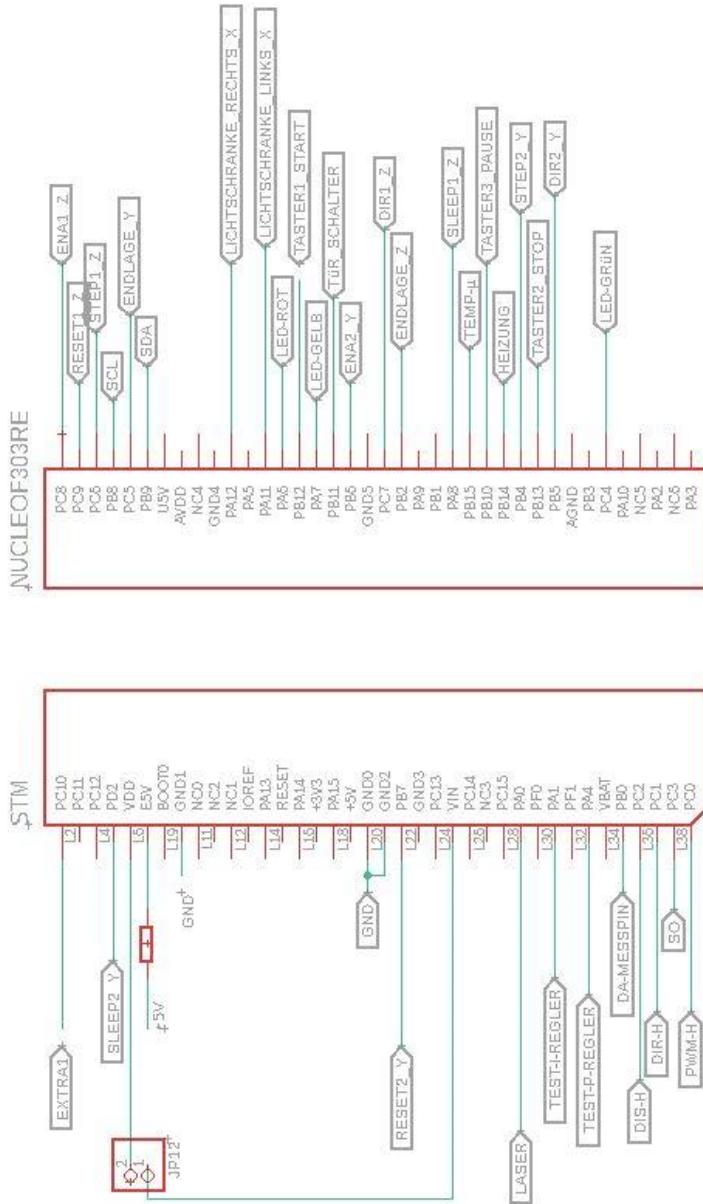


Projekt 3D-Drucker Gruppe C*

TITLE: H-Brücke

Document Number:

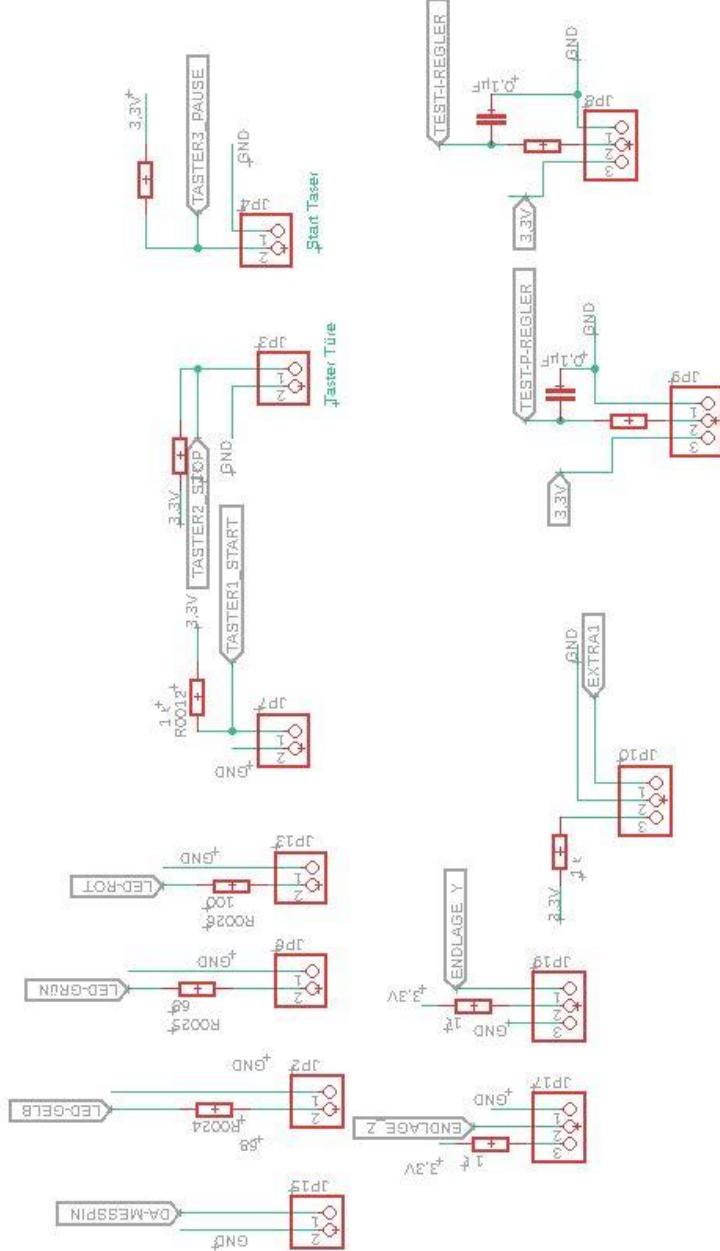
Date: 30.11.2019 Sheet: 4/7 REV:

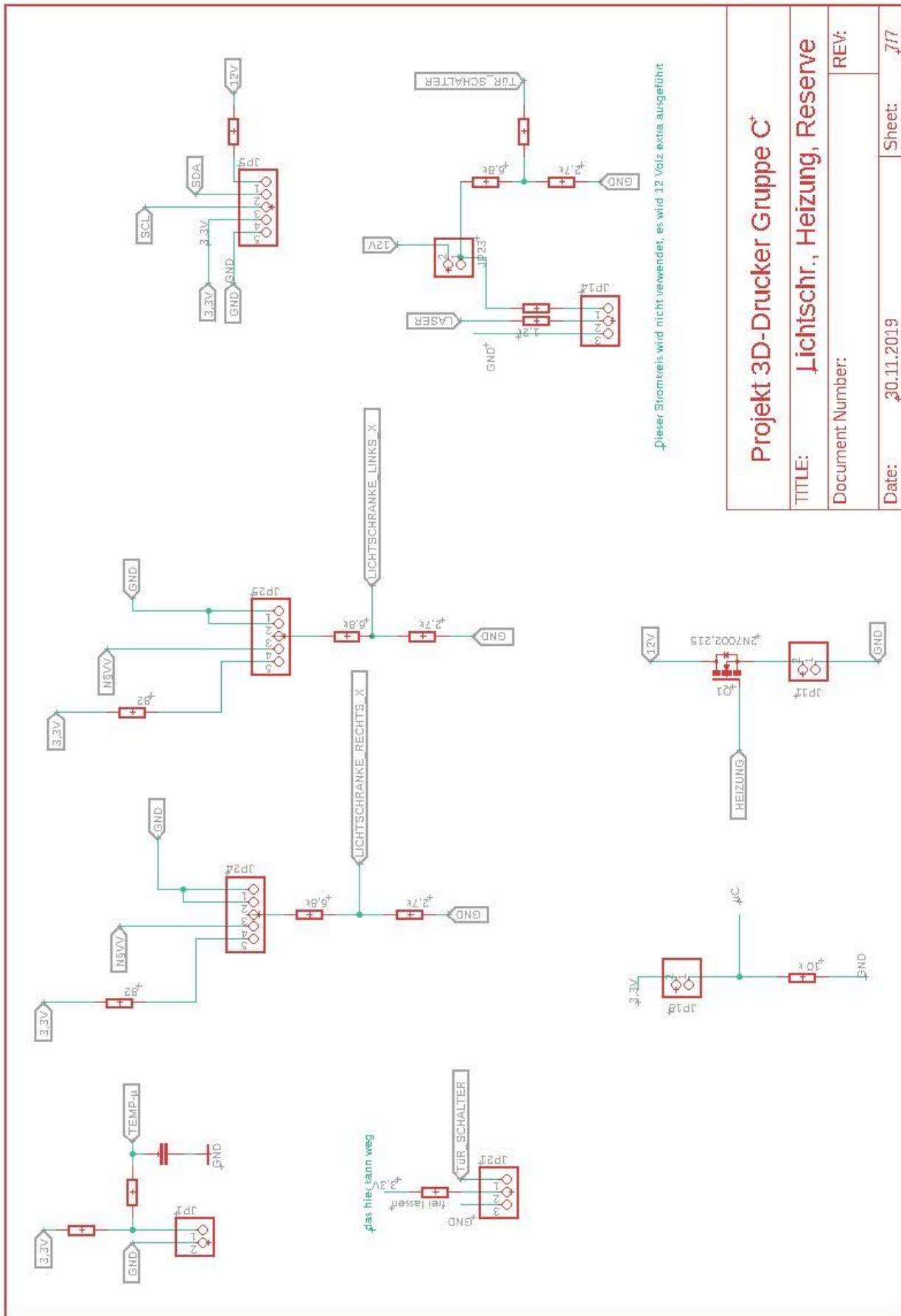


Projekt 3D-Drucker Gruppe C	
TITLE:	Kontakiplan Nucleo-Board
Document Number:	REV:
Date:	30.11.2019
Sheet:	5/7

Projekt 3D-Drucker Gruppe C

TITLE: Potti, Endschalter, Taster, LED+	
Document Number:	REV:
Date: 30.11.2019	Sheet: 6/7

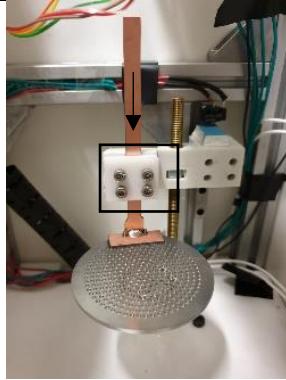
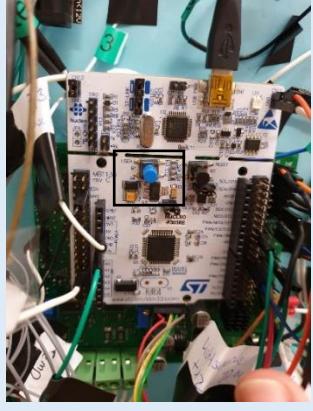




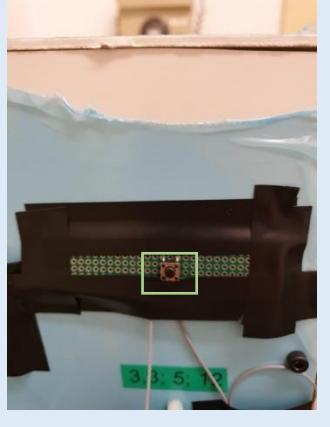
Bedienungsanleitung

Im Folgenden wird die Bedienungsanleitung für den Drucker vorgestellt.

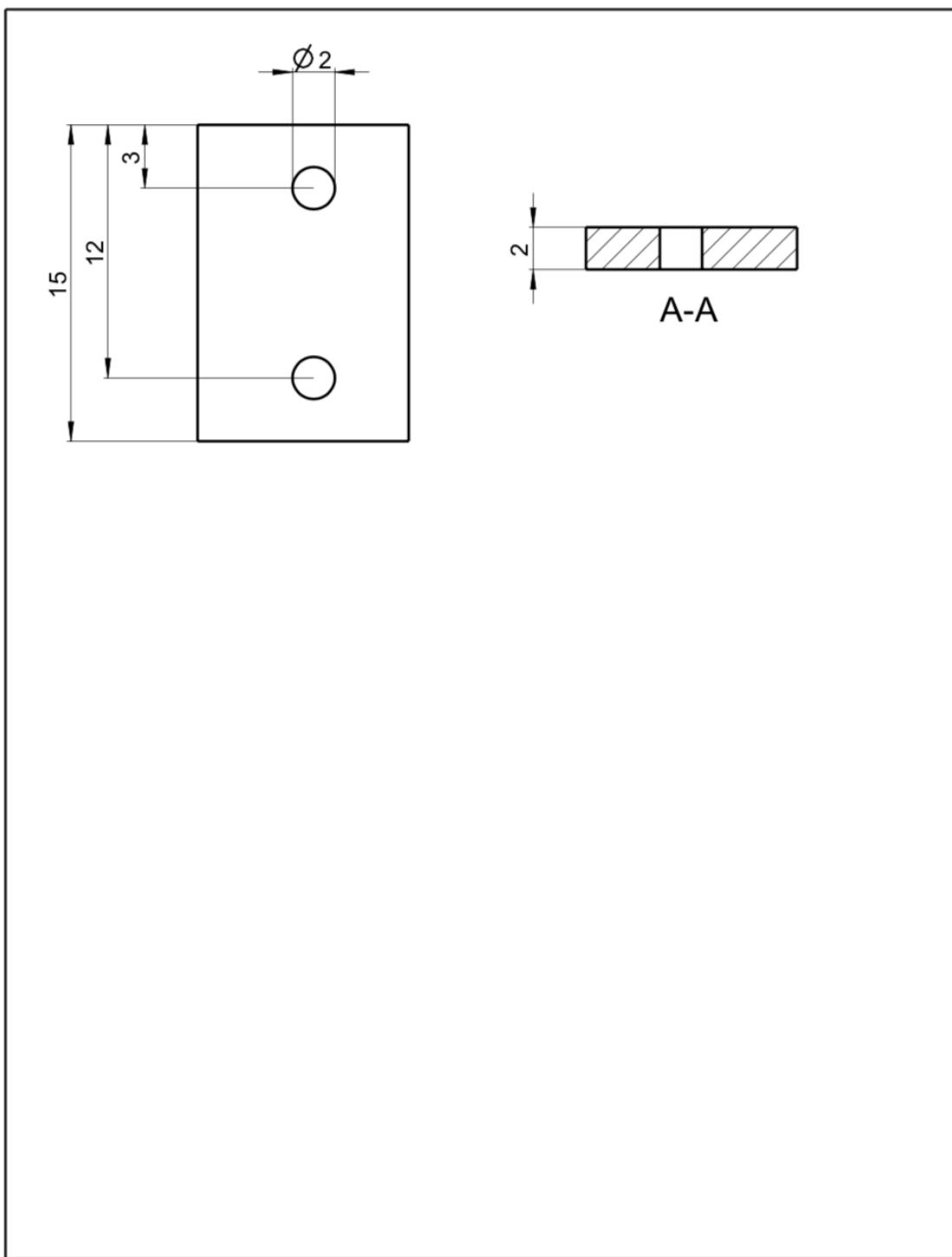
Schritt	Beschreibung	Abbildung	Gefahrenhinweis
Schritt 1	Mit Spannung (+12 V) verbinden		
Schritt 2	Türe aufmachen mit kleiner Sichtkontrolle		Laser Einschalten nicht möglich; Motor z-Achse bewegt sich
Schritt 3:	Der Drucker fährt anschließend die z-Achse zum Referenzpunkt am oberen Ende der Achse		Laser Einschalten nicht möglich
Schritt 4:	Nun wird das erwärmte Polymer in den Drucker gestellt. Sollte der Platz zwischen Druckbett und Becher zum Einstellen nicht ausreichen, kann das Druckbett mit seiner Halterung durch Aufschrauben der markierten Schrauben noch weiter in der Höhe variiert werden		Laser Einschalten nicht möglich

Schritt 5:	(Nur falls Druckbett-Schrauben geöffnet wurden) Festschrauben des Druckbetts und soweit ablassen bis Druckbett ganz nah über dem Polymer ist		
Schritt 6:	Mit Hilfe des blauen Tasters auf dem Nucleo Board kann das Druckbett anschließend feinjustiert werden. Dabei verfährt das Druckbett pro Knopfdruck um einen minimalen Weg nach oben.		
Schritt 7:	<p>Bis hierher: Alles ist eingesteckt, Druckbett ist in richtiger Höhe im Polymer. <u>Ab hier:</u> Schutzbrille verwenden, durch Tätigen des nächsten Schritts wird der Laser an Spannung geschalten.</p>		
Schritt 8:	Tür wird geschlossen und mit zwei Schrauben fixiert		

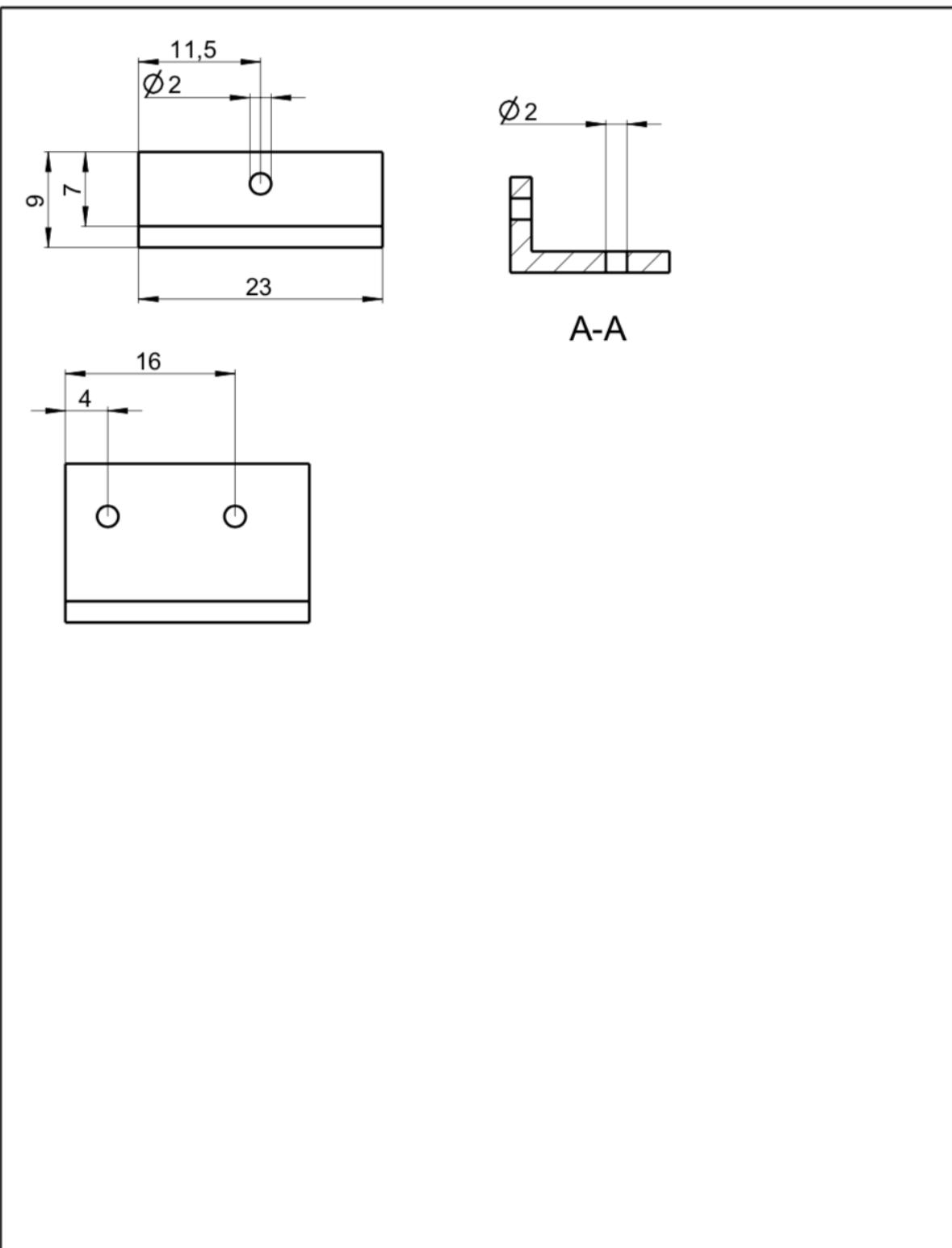
X

Schritt 9:	Beim Schließen der Türe wird dieser Endschalter geschlossen und schaltet den Laser an Spannung		
Schritt 10:	Start-Taster drücken		Programmablauf beginnt, besondere Vorsicht geboten
Schritt 11:	Benötigte Druckzeit kann anschließend am Display abgelesen werden		
Ab jetzt: Raum verlassen, Schutzbrille erst außerhalb des Raumes absetzen			
Schritt 12:	Zeit abwarten, bis Druck vorüber		
Schritt 13:	Fertigen Druck entnehmen		

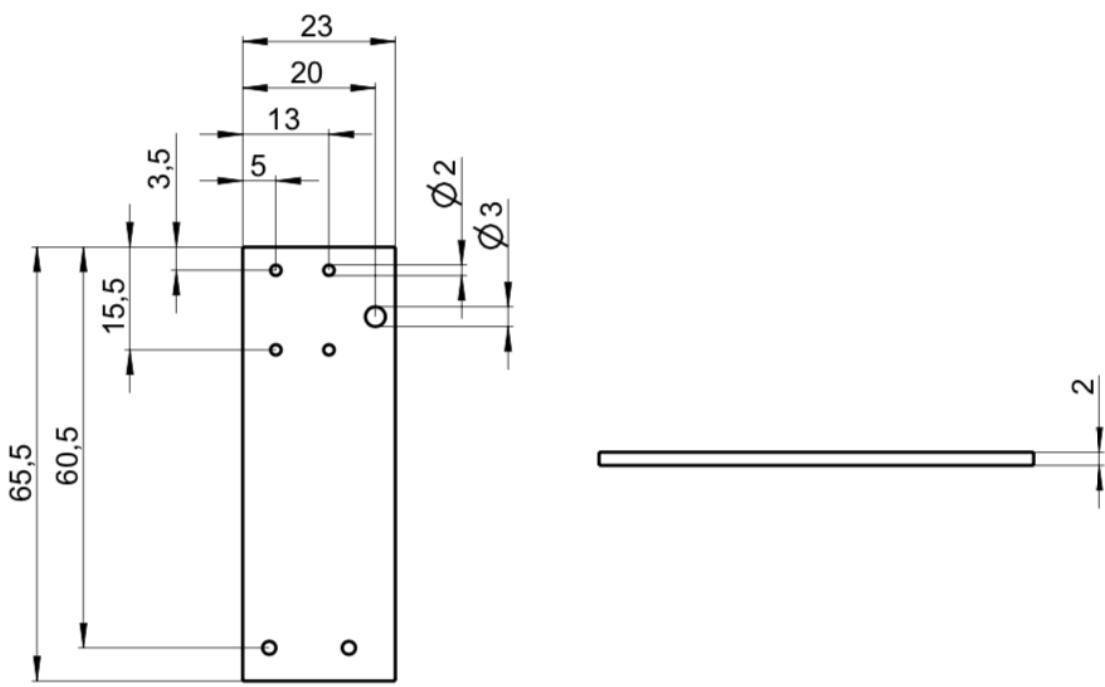
Technische Zeichnungen



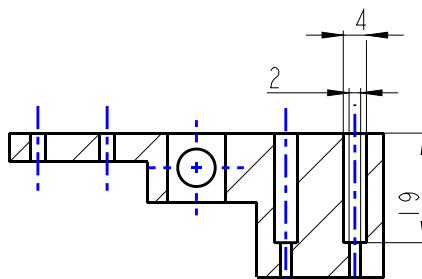
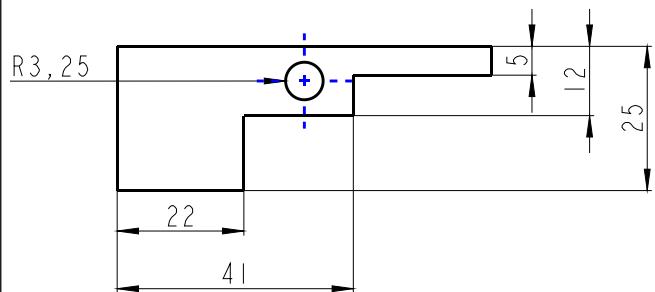
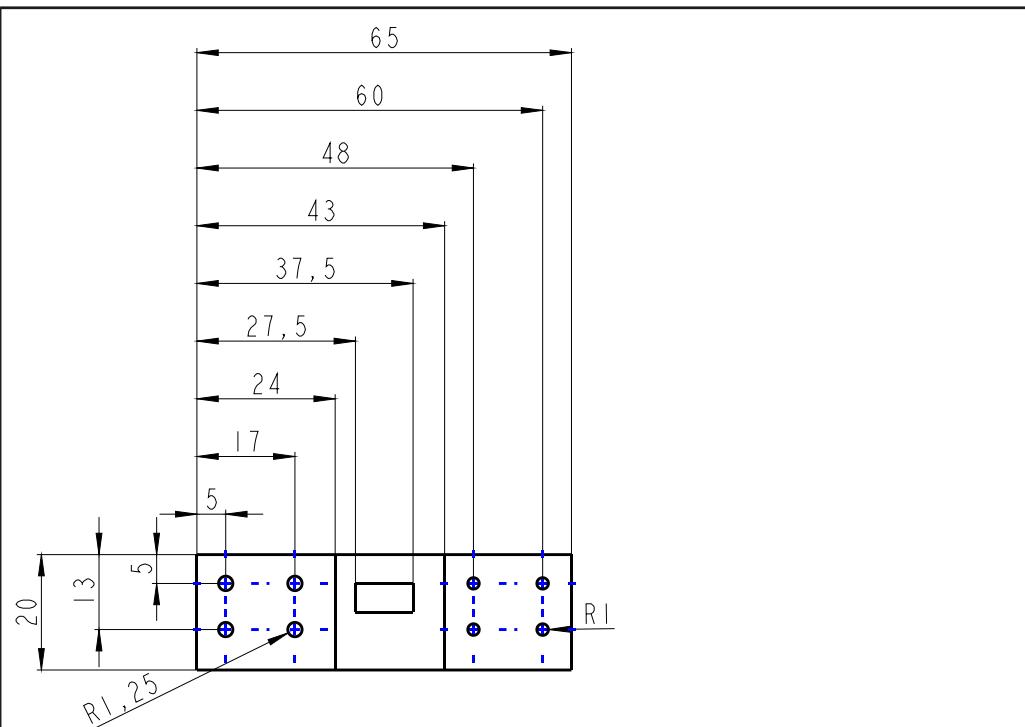
Allgemeintoleranzen DIN ISO 2768-m	Werkstückkanten DIN ISO 13715	Werkstoff, Halzeug	Maßstab 1:1	
Verantwortliche Abt. FB VII	Technische Referenz	Dokumentenart	Modellname VERBINDUNG_HALTER_FLUESSIK	
Matr.-Nr.	Erstellt durch:	Titel, Zusätzlicher Titel		Zeichnungs-Nr.
	Genehmigt von:			Änd. Ausgabedatum Spr. Blatt
				Nov-14-19 de



Allgemeintoleranzen DIN ISO 2768-m	Werkstückkanten DIN ISO 13715	Werkstoff, Halbzeug	Maßstab 1:1	
Verantwortliche Abt. FB VII	Technische Referenz	Dokumentenart	Modellname <u>VERBINDUNG_HALTUNGERUNG</u>	
Matr.-Nr.	Erstellt durch:	Titel, Zusätzlicher Titel		Zeichnungs-Nr.
	Genehmigt von:			Änd. Ausgabedatum Spr. Blatt
				Nov-14-19 de

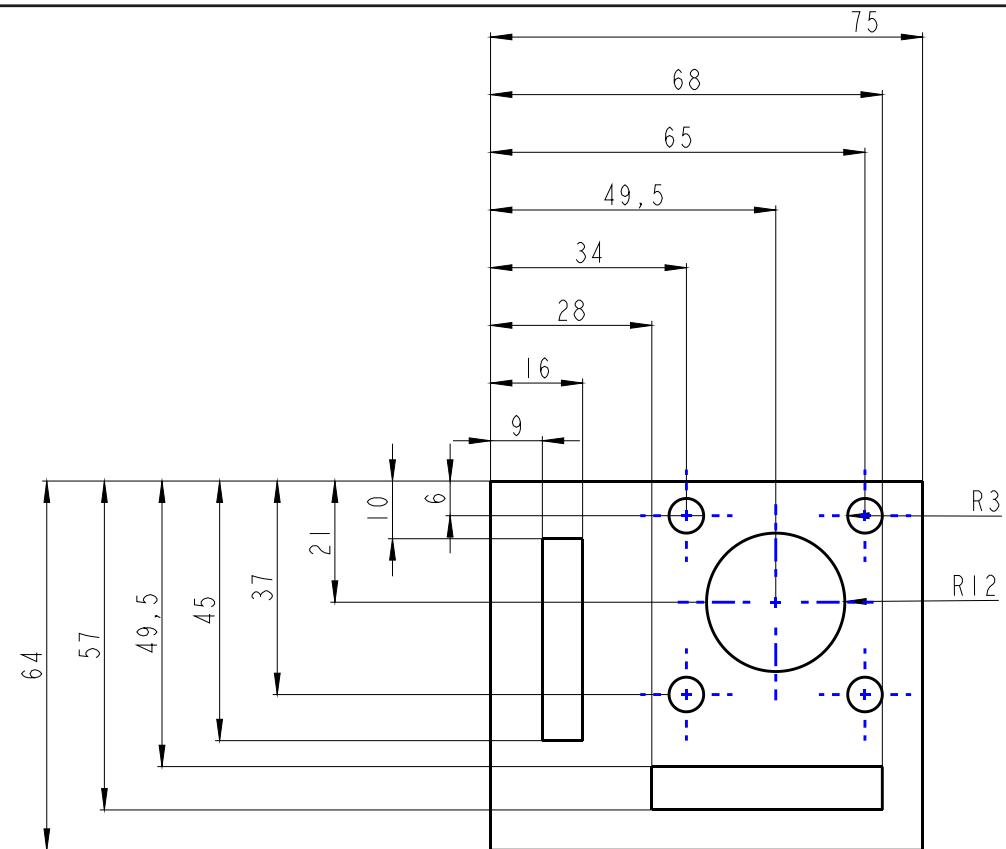


Allgemeintoleranzen DIN ISO 2768-m	Werkstückkanten DIN ISO 13715	Werkstoff, Halbzeug	Maßstab 1:1	
Verantwortliche Abt. FB VII	Technische Referenz	Dokumentenart	Modellname HALTUNG_FUER_DIE_FLUESSIG	
Matr.-Nr.	Erstellt durch:	Titel, Zusätzlicher Titel		Zeichnungs-Nr.
	Genehmigt von:			Änd. Ausgabedatum Spr. Blatt
			Nov-14-19	de

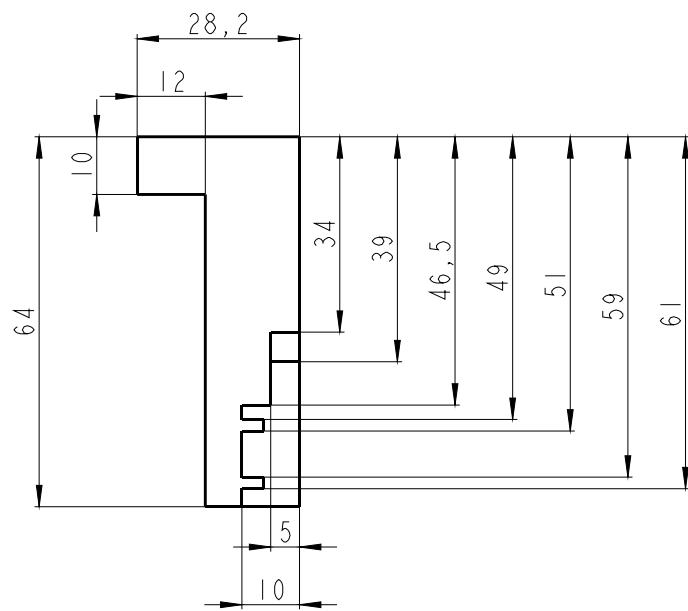


A - A

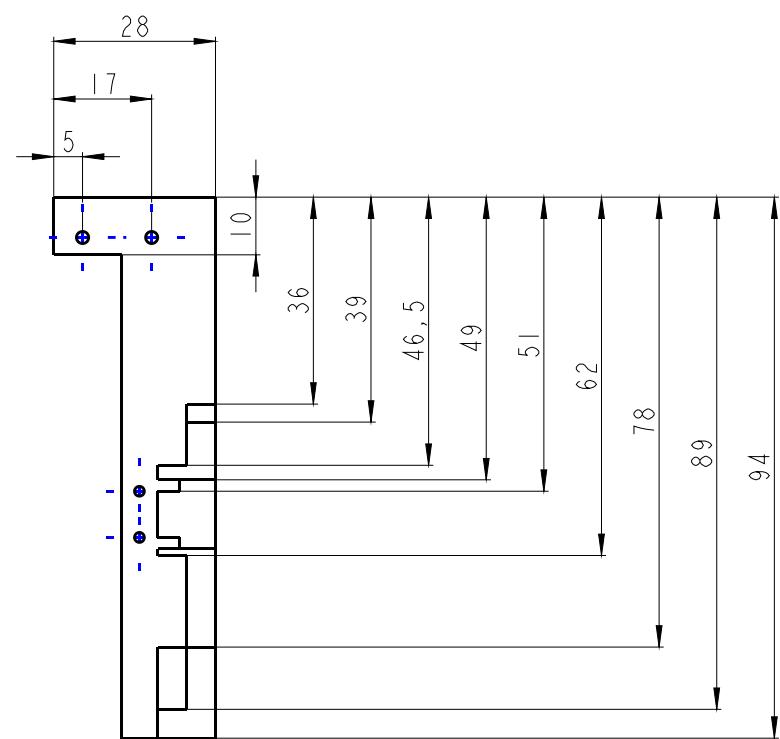
Allgemeintoleranzen DIN ISO 2768-m	Werkstückkonturen DIN ISO 13715	Werkstoff, Halbzeug	Maßstab 1:1	
Verantwortliche Abt. FB VII	Technische Referenz	Dokumentenart	Modellname VERBINDUNG_AKTUELL	
Matr.-Nr. Gruppe C	Erstellt durch:	Titel, Zusätzlicher Titel Verbindung	Zeichnungs-Nr. 20-0000-06	
	Genehmigt von:		And.	Ausgabedatum Feb-12-20
BEUTH HOCHSCHULE FÜR TECHNIK BERLIN University Applied Sciences			Spr.	Blatt de



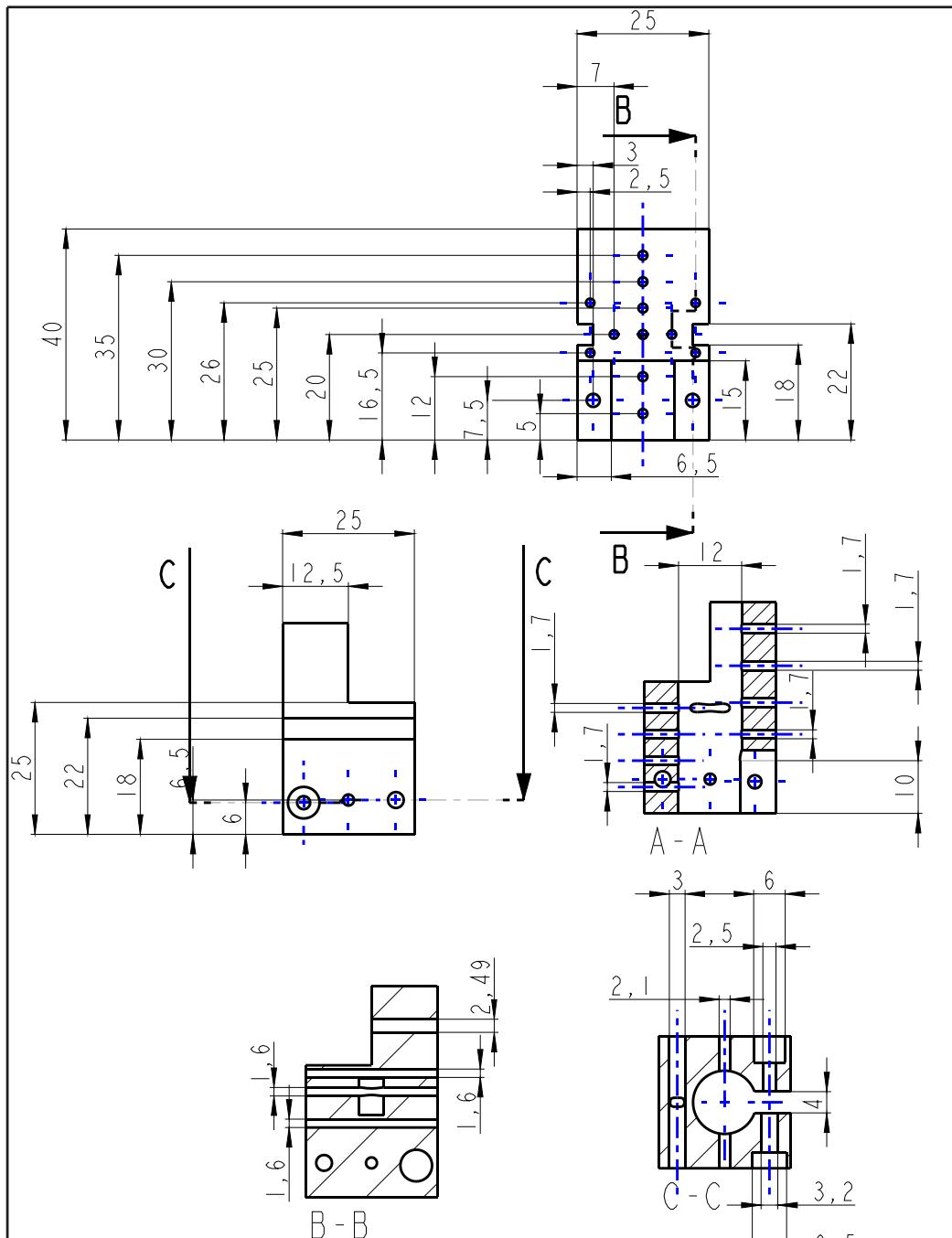
Allgemeintoleranzen DIN ISO 2768-m	Werkstückkonten DIN ISO 13715	Werkstoff, Halbzeug	Maßstab 1:1	
Verantwortliche Abt. FB VII	Technische Referenz	Dokumentenart	Modellname PRT0001	
Matr.-Nr. Gruppe C	Erstellt durch:	Titel, Zusätzlicher Titel	Zeichnungs-Nr. 20-0000-07	
	Genehmigt von:		And.	Ausgabedatum Feb-12-20



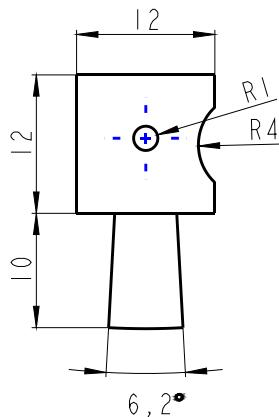
Allgemeintoleranzen DIN ISO 2768-m	Werkstückkonten DIN ISO 13715	Werkstoff, Halbzeug PA	Maßstab 1:1	
Verantwortliche Abt. FB VII	Technische Referenz	Dokumentenart	Modellname HALTER_SP_2	
Matr.-Nr. Gruppe C	Erstellt durch:	Titel, Zusätzlicher Titel Halterung	Zeichnungs-Nr. 20-0000-01	
	Genehmigt von:		And. Ausgabedatum Feb-11-20	Spr. Blatt de



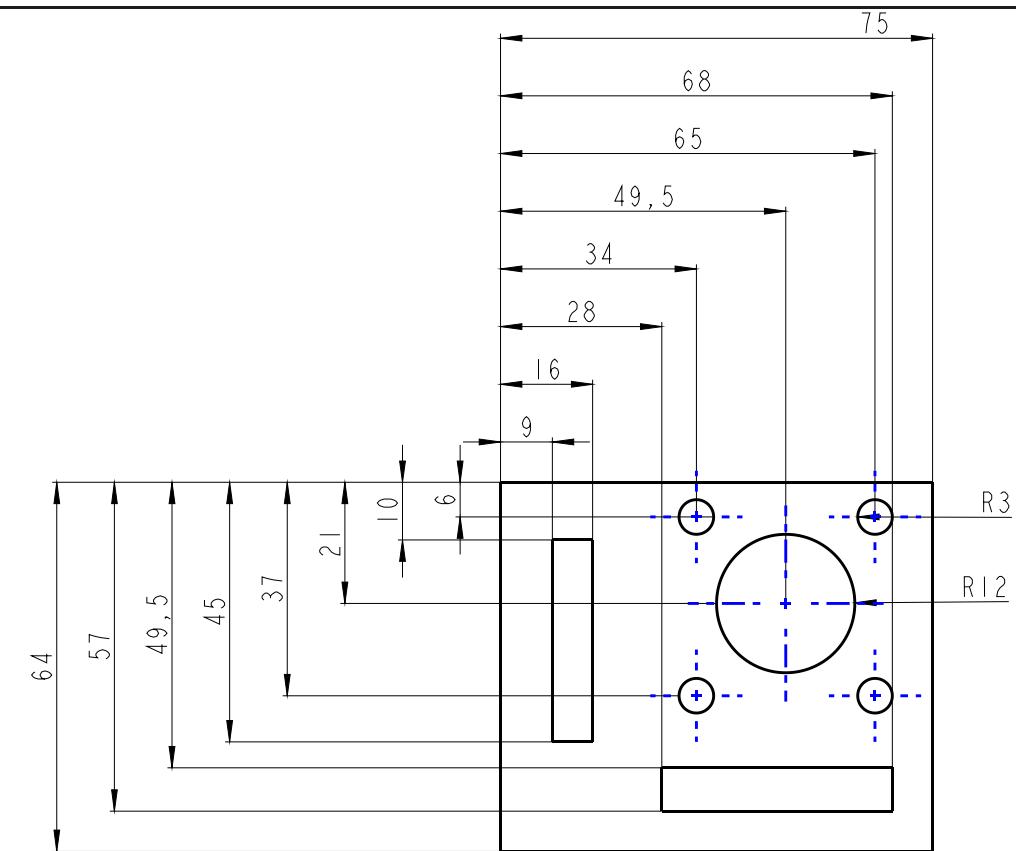
Allgemeintoleranzen DIN ISO 2768-m	Werkstückkonturen DIN ISO 13715	Werkstoff, Halbzeug PA	Maßstab 1:1	
Verantwortliche Abt. FB VII	Technische Referenz	Dokumentenart	Modellname HALTER_LS_SP	
Matr.-Nr. Gruppe C	Erstellt durch:	Titel, Zusätzlicher Titel	Zeichnungs-Nr. 20-0000-03	
	Genehmigt von:		And.	Ausgabedatum Feb-11-20



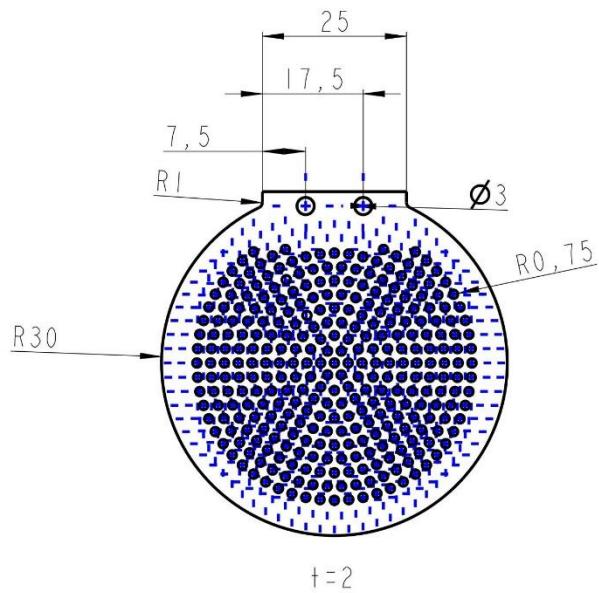
Allgemeintoleranzen DIN ISO 2768-m	Werkstückkonten DIN ISO 13715		Maßstab 1:1
Verantwortliche Abt. FB VII	Technische Referenz	Dokumentenart	Modellname LASER_HALTER
Matr.-Nr. Gruppe C	Erstellt durch: xx	Titel, Zusätzlicher Titel Laserhalterung	Zeichnungs-Nr. 20-0000-01g
	Genehmigt von:	And.	Ausgabedatum Feb-11-20
BEUTH HOCHSCHULE FÜR TECHNIK BERLIN University Applied Sciences		Spr.	Blatt de



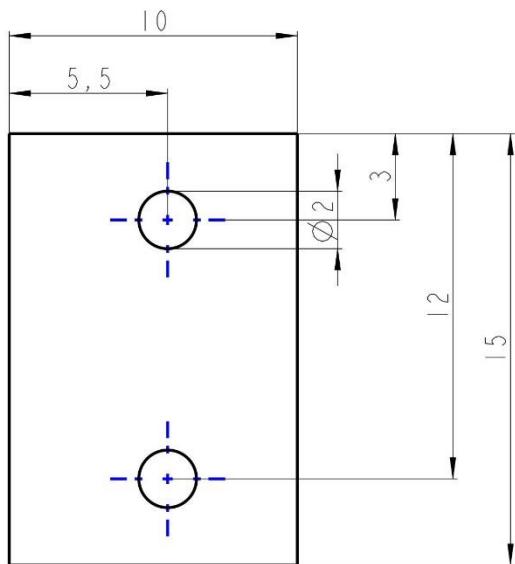
Allgemeintoleranzen DIN ISO 2768-m	Werkstückkonturen DIN ISO 13715	Werkstoff, Halbzeug PA	Maßstab 1:1	
Verantwortliche Abt. FB VII	Technische Referenz	Dokumentenart	Modellname BLENDE	
Matr.-Nr. Gruppe C	Erstellt durch:	Titel, Zusätzlicher Titel	Zeichnungs-Nr. 20-0000-04	
	Genehmigt von:		And. Ausgabedatum Feb-11-20	Spr. Blatt de



Allgemeintoleranzen DIN ISO 2768-m	Werkstückkonturen DIN ISO 13715	Werkstoff, Halbzeug	Maßstab 1:1			
Verantwortliche Abt. FB VII	Technische Referenz	Dokumentenart	Modellname PRT0001			
Matr.-Nr. Gruppe C	Erstellt durch:	Titel, Zusätzlicher Titel	Zeichnungs-Nr. 20-0000-07			
 BEUTH HOCHSCHULE FÜR TECHNIK BERLIN University of Applied Sciences	Genehmigt von:		Änd.	Ausgabedatum Feb-12-20	Spr.	Blatt de



Allgemeintoleranzen DIN ISO 2768-m	Werkstückkanten DIN ISO 13715	Werkstoff, Halbzeug AL	Maßstab 1 : 1		
Verantwortliche Abt. FB VII	Technische Referenz	Dokumentenart	Modellname Z_PLATTFORM_F_FLO		
Matr.-Nr. Gruppe C	Erstellt durch:	Title, Zusätzlicher Titel Druckfeld	Zeichnungs-Nr. 20-0000-05		
	Genehmigt von:		Änd.	Ausgabedatum Feb-12-20	Spr. Blatt de



Allgemeintoleranzen DIN ISO 2768-m	Werkstückkanten DIN ISO 13715	Werkstoff, Halbzeug	Maßstab 5:1		
Verantwortliche Abt. FB VII	Technische Referenz	Dokumentenart	Modellname VERBINDUNG_HALTER_FLUSS		
Matr.-Nr. Gruppe C	Erstellt durch:	Title, Zusätzlicher Titel	Zeichnungs-Nr.		
	Genehmigt von:		Änd.	Ausgabedatum Feb-11-20	Spr. Blatt de