

Service- und Entwicklerdokumentation

Demonstrator für einen Schrittmotor

Autor:	Maik Theilmann
Matrikel-Nr.:	7021861
Autor:	Jannik Grönke
Matrikel-Nr.:	7022006
Autor:	Enno Stein
Matrikel-Nr.:	7021960
Autor:	Lars Hanneken
Matrikel-Nr.:	7021910
Autor:	Jan ter Veen
Matrikel-Nr.:	7022071
Studiengang:	Maschinenbau und Design
Erstprüfer:	Prof. Dr. Elmar Wings
Abgabedatum:	22. Juni 2024

Hochschule Emden/Leer · Fachbereich Technik · Abteilung Maschinenbau
Constantiaplatz 4 · 26723 Emden · <http://www.hs-emden-leer.de>

Inhaltsverzeichnis

Inhaltsverzeichnis	i
Abbildungsverzeichnis	v
Tabellenverzeichnis	vii
Liste des Listings	ix
Acronyms	xi
1 Projektbeschreibung	1
1.1 Kurzbeschreibung des Demonstrators	1
1.2 Herausforderungen	1
1.3 Lösungsansatz	1
1.4 Anwendungsbereich des Demonstrators	4
1.5 Besondere Herausforderungen bei den Quellen	4
2 Hardwarebeschreibung des Arduino	7
2.1 Aufbau des Arduinos	7
2.2 Integrierte Sensorik	8
2.2.1 9-Achs-IMU für die Bewegungserkennung (LSM9DS1)	8
2.2.2 Näherungs-, Umgebungslicht-, Farb- und Gestensen-	
sor (APDS9960)	8
2.2.3 Barometrischer Drucksensor (LPS22HB)	9
2.2.4 Digitales Mikrofon (MP34DT05)	10
2.2.5 DC-DC-Wandler (MPM3610)	10
2.3 Beschreibung der Schnittstellen	10
2.3.1 I ² C	10
2.3.2 USB	12
2.3.3 Bluetooth®5	12
2.3.4 Weitere Kommunikationsschnittstellen	12
2.3.5 Digitale Ein- und Ausgangspins	13
2.3.6 Analoge Eingangspins	13
2.3.7 Weitere Pins	13
2.3.8 LED-Lampen	14
2.4 Hinweis Arduino 33 BLE Sense Lite	14
2.5 Bezugsquellen	14

3 Schrittmotor	15
3.1 Beschreibung eines Schrittmotors	15
3.1.1 Aufbau und Funktionsweise eines Schrittmotors	15
3.1.2 Schrittmotor Bauformen	17
3.1.3 Betriebsarten unipolar und bipolar	18
3.1.4 Mikroschrittverfahren	18
3.2 Schrittverluste verhindern	19
3.2.1 Auswahl des Schrittmotors	19
3.2.2 Betriebsart	19
3.2.3 Externe Ereignisse	22
3.3 Beschreibung des verwendeten Schrittmotors	22
4 Weitere Hardware	25
4.1 OLED-Display	25
4.2 Signalleuchte	26
4.3 Spannungswandler	26
4.3.1 Netzteil SNT RD 50A	26
4.3.2 Spannungswandler ASM1117	26
4.4 Linearführung	27
4.5 Drehwinkel-Encoder	27
4.6 Schrittmotorsteuerung	27
5 Beschreibung der Software IDE	29
5.1 Installation der Arduino IDE	29
5.2 Beschreibung der Entwicklungsumgebung	29
5.3 Erste Schritte in der Entwicklungsumgebung	30
5.3.1 Auswahl des Mikrocontrollers	30
5.3.2 Bibliotheken einbinden	31
5.4 Programmierung	34
5.4.1 header	34
5.4.2 setup()	34
5.4.3 loop()	34
5.5 Erster Programmtest	34
5.6 Erster Programmtest	34
6 Beschreibung des Programms auf dem Arduino	37
6.1 Aufgabe des Programms	37
6.2 Erklärung des Codes	39
6.3 Bibliotheken	45
6.3.1 Accelstepper	45
6.3.2 Wire	46
6.3.3 Adafruit GFX	46
6.3.4 Adafruit SSD1306	46
6.3.5 Encoder	47

Inhaltsverzeichnis	iii
7 Konstruktion	49
7.1 Rahmenbedingungen	49
7.2 Arbeiten mit SolidWorks 2023	49
7.3 Gehäusekonstruktion	49
7.4 Anbauteile	57
7.5 3D-Druck mit PLA	62
8 Testdurchläufe	65
8.1 Arduino Nano 33 BLE Sense Lite	65
8.2 Schaltnetzteil	65
8.3 Spannungsregler AMS1117	65
8.4 Nema 17 Schrittmotor mit Schrittmotorsteuerung A4988	66
8.5 OLED-Display	66
8.6 SMD-LED	75
8.7 Drucktaster und Mikroschalter	76
8.8 Drehwinkel-Encoder	78
8.9 Mittlere Geschwindigkeit des Schlittens	79
9 Offene Punkte	81
9.1 Misslungene Messung der Beschleunigung des Schlittens	81
9.2 Misslungene Ansteuerung der SMD-LED	83
9.3 Verbesserungen	84
9.4 Erweiterung	84
9.5 Anwendung in der Lehre	84
Literaturverzeichnis	85
Anhang	89
10 Schaltplan	91
11 Materialliste	93
Index	99

Abbildungsverzeichnis

1.1	Erste Konzeptskizze (Eigenaufnahme)	2
1.2	Zweite Konzeptskizze (Eigenaufnahme)	3
1.3	CAD-Modell (Eigenaufnahme)	3
1.4	Mindmap	4
2.1	Arduino Nano 33 BLE Sense (Vereinfachte Darstellung) . .	7
2.2	Arduino Nano Pinbelegung [Ard24a]	11
3.1	Prinzipieller Aufbau Schrittmotor (2-phasic) [Hag21] . .	16
3.2	Start-Stopp Frequenz [Fau20]	20
3.3	Trapezförmiges Geschwindigkeitsprofil [Fau20]	20
3.4	Drehmoment-Diagramm für den Schrittmotor 42BYG015 [Glo]	23
5.1	Hauptoberfläche in der Arduino IDE (Eigenaufnahme) .	30
5.2	Installation des Boards (Eigenaufnahme)	31
5.3	Herunterladen von Bibliotheken (Eigenaufnahme) . . .	33
6.1	Flussdiagramm des Programms	38
7.1	Baugruppe Gehäuse (Eigenaufnahme)	50
7.2	Bodenplatte (Eigenaufnahme)	51
7.3	Vorderplatte (Eigenaufnahme)	52
7.4	Seitenplatte Links (Eigenaufnahme)	53
7.5	Hinterplatte (Eigenaufnahme)	54
7.6	Seitenplatte Rechts (Eigenaufnahme)	55
7.7	Deckelplatte (Eigenaufnahme)	56
7.8	Halterung für den Motor (Eigenaufnahme)	58
7.9	Halterung für die Welle (Eigenaufnahme)	59
7.10	Transportgriff (Eigenaufnahme)	60
7.11	Drehknopf (Eigenaufnahme)	61
7.12	Anzeiger (Eigenaufnahme)	62
7.13	AnyKubic 2 Kobra Neo (Eigenaufnahme)	63
9.1	Montage des Arduino Nano 33 Ble Sense Rev2 auf dem Schlitten	82
9.2	Testaufbau Messung Beschleunigung	83

Tabellenverzeichnis

2.1	Arduino Nano 33 BLE Sense [Ard24a]	7
3.1	Ursachen und Lösungen: Motor läuft nicht an [Fau20]	20
3.2	Ursachen und Lösungen: Motor beendet die Beschleunigungsrampe nicht [Fau20]	21
3.3	Ursachen und Lösungen: Motor beschleunigt bis zur Enddrehzahl und bleibt stehen, sobald eine konstante Drehzahl erreicht ist. [Fau20]	21
7.1	Bohrungen im Gehäuse	57
7.2	Bohrungen der Anbauteile	62
8.1	Ergebnisse Messung der Zeit eines Weges der einzelnen Stufen	79
8.2	Mittlere Geschwindigkeit und Durchschnittswert in SI-Einheiten	79
8.3	Durchschnittswerte umgerechnet in mm/s	80
11.1	Materialliste	98

Liste des Listings

5.1	Blink.py	35
6.1	Einbindung der Bibliotheken	39
6.2	define-Anweisungen	40
6.3	globale Variablen	41
6.4	void setup	42
6.5	void loop	43
6.6	void updateLEDs	44
6.7	void updateDisplay	44
6.8	bool findReference	44
6.9	void handleInterrupt	45
8.1	Testprogramm für den Schrittmotor	66
8.2	Testprogramm für das OLED-Display	74
8.3	Testprogramm für die LED	75
8.4	Testprogramm für die Taster	76
8.5	Testprogramm für die Taster mit Debounce	77
8.6	Testprogramm für den Encoder	78

Abkürzungen

AAR Automatic Address Recognition

ADC Analog to Digital Converter

AES Advanced Encryption Standard

BLE Bluetooth Low Energie

CAD Computer Aided Design

CCM Counter with CBC-MAC

DMA Direkt Memory Access

ECB Electronic Codebook

EMI Electromagnetic Interference

FPU Floating Point Unit

hy Hybrid

I²C Inter-Integrated Circuit

IC Inter Circuit

IMU Inertialmesseinheit

LED Light Emitting Diode

Micro USB Micro Universal Serial Bus

NFC Near Field Communication

OLED Organic Light Emitting Diode

PDM Pulsdichtenmodulation

SCL Serial Clock

SDA Serial Data

SMD Surface Mount Device

SoC System on Chip

SPI Serial Peripheral Interface

UV-Filter Ultraviolet-Filter

1 Projektbeschreibung

1.1 Kurzbeschreibung des Demonstrators

Ein Riemen wird von einem Schrittmotor angetrieben. An diesem Riemen ist ein Schlitten befestigt, der durch die Bewegung des Schrittmotors auf einer Linearführung bewegt wird. Am Gehäuse befindet sich ein Drehschalter, mit dem eine von zehn Bewegungsstufen ausgewählt werden kann. Nach der Auswahl kann das Programm mit der grünen Start-/Stop-Taste gestartet werden. Die Status-LED leuchtet rot, sobald das Gerät betriebsbereit ist. Für die Referenzfahrt ist am Schrittmotorhalter ein Endschalter angebracht, damit der Schrittmotor auf eine definierte Position referenzieren kann. Es gibt zehn verschiedene Bewegungsstufen. Die Bewegungsstufen unterscheiden sich in der Geschwindigkeit. Mit jeder Stufe erhöht sich die Geschwindigkeit.

1.2 Herausforderungen

Das Projekt kann in drei verschiedene Teilprojekte unterteilt werden, die zur Erfüllung der Aufgabe führen. Das erste ist der Aufbau des Demonstrators und die Auswahl der dafür notwendigen Hardwarekomponenten. Die Vorgabe war, den Aufbau nicht zu komplex zu gestalten und sich auf die wesentlichen Teile zu konzentrieren. Des Weiteren ist die Arbeit mit der Arduino IDE und die Programmierung des Arduino ein weiterer Teil. Der letzte Punkt ist der Umgang mit den elektronischen Bauteilen. Der Umgang mit den elektronischen Bauteilen muss sehr vorsichtig erfolgen, damit diese nicht durch den elektrischen Strom beschädigt werden.

1.3 Lösungsansatz

Durch die Erarbeitung der Herausforderungen können Lösungsansätze entwickelt werden. Im Vorfeld wurde eine erste Konzeptskizze erstellt, aus der Ideen entstanden. Im ersten Konzept sollte der Schrittmotor über einen Riementrieb eine Plattform axial bewegen, erkennbar in Abbildung 1.1. Dabei sollte der Verfahrweg der Plattform über einen Abstandssensor mit einem vorgegebenen Abstand zu einem Objekt geregelt werden. Bewegt sich das Objekt auf den Abstandssensor zu oder von diesem weg, so bewegt

sich die Plattform um einen definierten Abstand mit. Alle vorhandenen Bauteile sollen an einem Aluminiumprofil befestigt werden.

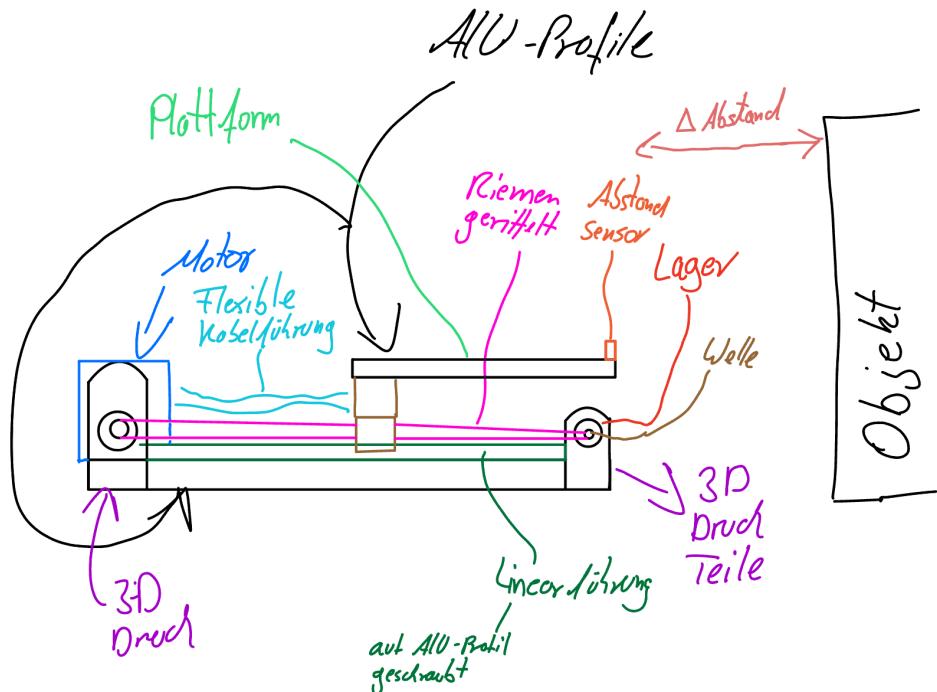


Abbildung 1.1: Erste Konzeptskizze (Eigenaufnahme)

Aufgrund der hohen Komplexität wurde beim zweiten Konzept, wie in Abbildung 1.2 dargestellt, auf die Plattform inklusive Abstandssensor verzichtet. Stattdessen ist auf dem Schlitten der Linearführung ein Zeiger und auf dem Aluminiumprofil ein Lineal integriert. Der Schrittmotor soll über verschiedene Stufen unterschiedliche Bewegungsprofile demonstrieren. Insbesondere soll die Positioniergenauigkeit mit dem Zeiger dargestellt werden. Des Weiteren wurde ein Netzteil integriert, das sowohl den Motor als auch den Arduino mit Spannung versorgt. Weiterhin sind ein Schalter zum Ein- und Ausschalten des Demonstrators, ein Stufenschalter zur Auswahl der Stufen sowie ein Taster zum Starten des Demonstratorablaufs integriert.

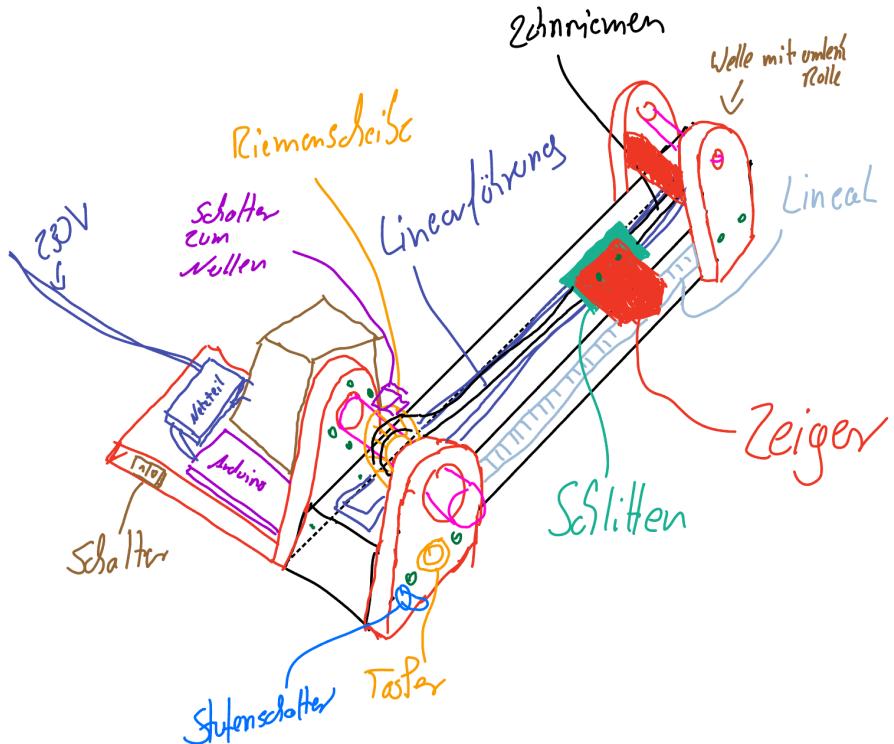


Abbildung 1.2: Zweite Konzeptskizze (Eigenaufnahme)

Das zweite Konzept gefiel und wurde weiter ausgearbeitet, so dass mit dem Handbuch begonnen werden konnte (siehe Handbuch Demonstrator für einen Schrittmotor). In diesem Handbuch sollten vor allem die einzelnen Funktionen und die Bedienung aus Kundensicht geklärt werden. Parallel dazu konnte ein Computer Aided Design (CAD)-Modell erstellt werden, zu sehen in Abbildung 1.3. Dazu konnte im Team eine MindMap erstellt werden, die alle möglichen Inhalte und zu erledigenden Aufgaben zusammenfasst, zu sehen in Abbildung 1.4. Im Vergleich zum zweiten Konzept wurde die Hardware aus Transportgründen auf das Aluminiumprofil gelegt und erhielt zum Schutz ein Gehäuse. Außerdem konnte eine Materialliste erstellt werden.

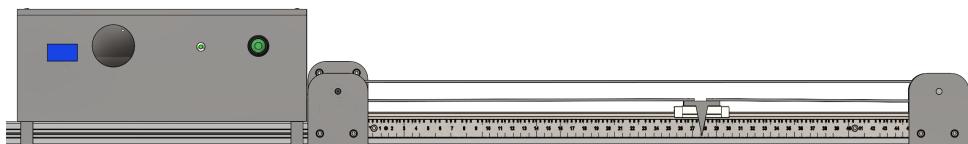


Abbildung 1.3: CAD-Modell (Eigenaufnahme)

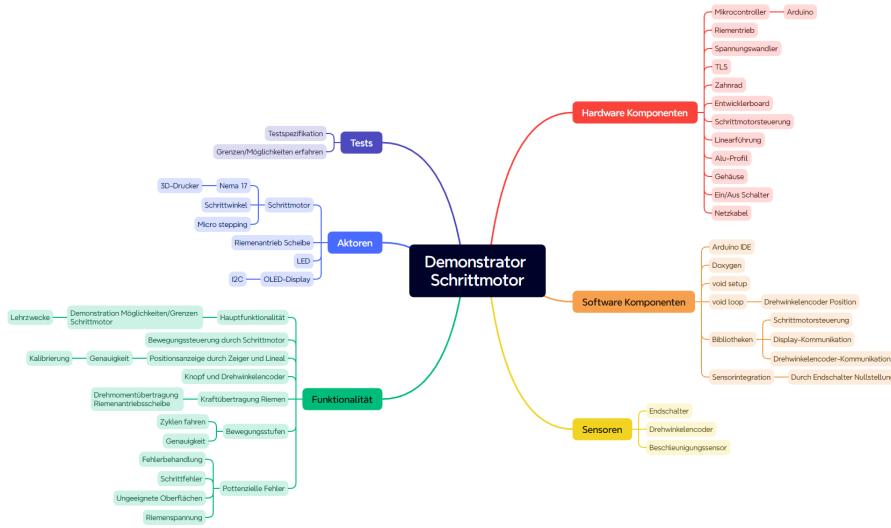


Abbildung 1.4: Mindmap

1.4 Anwendungsbereich des Demonstrators

Die grundsätzliche Aufgabe des Demonstrators ist es, die Funktionsweise eines Schrittmotors zu veranschaulichen. Der Demonstrator soll zur Veranschaulichung und als ergänzendes Lehrmittel dienen. So können Studierende die Prinzipien der Schrittmotorsteuerung einschließlich Schrittauflösung und Positionierung praxisnah erlernen. Darüber hinaus können die Studierenden den Demonstrator nutzen, um praktische Erfahrungen zu sammeln, z.B. durch Programmierübungen. Darüber hinaus könnte ein solcher Demonstrator als Qualitätskontrolle dienen, um die Leistungsfähigkeit und Genauigkeit von Schrittmotoren zu überprüfen und sicherzustellen, dass diese den geforderten Spezifikationen entsprechen. Ein weiteres Anwendungsgebiet könnten Fachmessen sein, um mit einem solchen Demonstrator Bewegungsabläufe und Positioniergenauigkeit zu demonstrieren.

1.5 Besondere Herausforderungen bei den Quellen

Die größte Herausforderung bei der Beschaffung von Literaturquellen ist die Sprachbarriere, wenn relevante Quellen nur in Englisch oder anderen Sprachen verfügbar sind. Diese Barriere kann das Verständnis und die Interpretation der Quellen erheblich erschweren. Dies führt zu erhöhtem

Aufwand, da Quellen häufig übersetzt werden müssen. Online-Tools wie Deepl (www.deepl.com) können hier Abhilfe schaffen. Ein weiteres Problem ist die mangelnde Verfügbarkeit von Datenblättern, die von den Herstellern trotz mehrfacher Nachfrage nicht zur Verfügung gestellt wurden. Datenblätter sind für technische Projekte unerlässlich, da sie wichtige Informationen über die Funktionsweise und die Einsatzmöglichkeiten liefern. Ein weiteres Problem ist das Fehlen des Veröffentlichungsdatums oder die Veralterung der Quellen. Dies erschwert eine genaue Dokumentation, da der aktuelle Stand der Technik nicht genau nachvollzogen werden kann.

2 Hardwarebeschreibung des Arduino

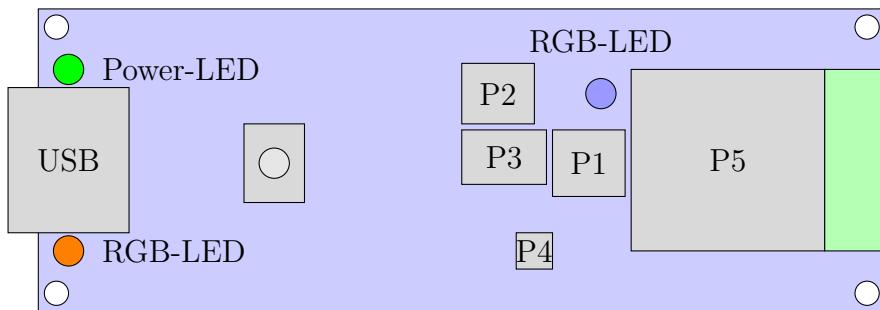


Abbildung 2.1: Arduino Nano 33 BLE Sense (Vereinfachte Darstellung)

Pos.Nr.	Modul/Sensor
P1	Mikrophon
P2	9-Achs-IMU
P3	Näherungs-, Umgebungslicht-, Farb- und Gestensor
P4	Drucksensor
P5	Bluetooth-Modul

Tabelle 2.1: Arduino Nano 33 BLE Sense [Ard24a]

2.1 Aufbau des Arduinos

Der Arduino Nano 33 Bluetooth Low Energie (BLE) Sense Lite ist ein Mikrocontroller basierend auf dem Nordic nRF52840-System on Chip (SoC). Trotz seiner kompakten Bauweise mit einer Größe von 45 x 18 mm verfügt es über mehrere verschiedene integrierte Sensoren, Aktoren und Anschlusschnittstellen (siehe Abbildung 2.1). Darüber hinaus verfügt der Arduino-Prozessor über 1 MB Flash-Speicher und 256 KB RAM. Bei dem Prozessor handelt es sich um einen Arm®Cortex-M4F Prozessorkern, welcher mit einer Taktrate von 64 MHz arbeitet und für Mikrocontroller Anwendungen optimiert ist. Dieser bietet eine gute Leistung bei geringen Stromverbrauch und ist auch für Echtzeit-Verarbeitungsaufgaben geeignet. Außerdem besitzt der Prozessor eine Floating Point Unit (FPU), die eine effiziente Abarbeitung von Datenverarbeitungsoperationen ermöglicht.

Mit Hilfe einer integrierten Steckverbindung (engl. „header“) kann der Mikrocontroller direkt auf ein Board gesteckt werden. [Arm20] [Ard24a]

2.2 Integrierte Sensorik

Der Arduino Nano 33 BLE-Sense Lite verfügt über eine Vielzahl von integrierten Sensoren und Aktoren, mit denen verschiedene Umgebungseigenschaften erfasst werden können. Dazu gehören die in den folgenden Abschnitten beschriebenen Sensoren und Aktoren.

2.2.1 9-Achs-IMU für die Bewegungserkennung (LSM9DS1)

Bei der Inertialmesseinheit LSM9DS1 handelt es sich um ein System-in-Package, d.h. es sind mehrere elektronische Komponenten bzw. Chips auf kleinstem Raum in einem Gehäuse kombiniert.[LD12] Die Inertialmessseinheit (IMU) verfügt über einen 3D-Linearbeschleunigungsmesser, ein 3D-Gyroskop und einen 3D-Magnetometer. Darüber hinaus verfügt das System über eine serielle Inter-Integrated Circuit (I^2C)-Busschnittstelle, die einen Standard und einen Fast Mode (100/400 kHz) bereitstellt, zudem eine serielle SPI-Standardschnittstelle. Der Sensor verfügt über einen linearen Beschleunigungsmesser (Accelerometer) mit wählbarer Skala von $\pm 2/\pm 4/\pm 8/\pm 16$ g, der die lineare Beschleunigung in drei Achsen (x, y, z) misst und die Erfassung von Geschwindigkeits- und Positionsänderungen ermöglicht. Das Magnetfeld ist mit einer wählbaren Skala von $\pm 4/\pm 8/\pm 12/\pm 16$ Gs ausgestattet, damit misst es das magnetische Feld in den drei Achsen und ermöglicht die Bestimmung der Ausrichtung relativ zur Erdmagnetfeldrichtung. Das 3D-Gyroskop ist mit einem wählbaren Skalenendwert von: $\pm 125/\pm 250/\pm 500/\pm 1000/\pm 2000 \frac{\text{Grad}}{\text{s}}$ ausgestattet und ist für die Erfassung der Winkelgeschwindigkeit bzw. Drehbewegung um die drei Achsen zuständig.[STM15][Ard24a] Mögliche Anwendungsbereiche sind zum Beispiel eine Bewegungssteuerungen (Drohnensteuerung, Robotik und industrielle Automatisierung), Schwingungsüberwachung und -kompensation, Antennen, Plattformen, optische Bild- und Objektivstabilisierung.

2.2.2 Näherungs-, Umgebungslicht-, Farb- und Gestensor (APDS9960)

Hierbei handelt sich um einen sehr vielseitigen Sensor. Er dient zur Gestenerkennung, Farberkennung, Abstandsmessung und Umgebungslichtmessung. Die Gestenerkennung nutzt vier gerichtete Fotodioden, um reflektierte Infrarot-Energie (von der integrierten Light Emitting Diode

(LED)) zu erfassen, um physikalische Bewegungsinformationen (d.h. Geschwindigkeit, Richtung und Entfernung) in digitale Informationen zu übersetzen. Die Näherungserkennung ermöglicht die Messung der Entfernung durch die Erkennung der reflektierten Infrarot-Energie, (von der integrierten LED) mit Hilfe einer Fotodiode. Erkennungseignisse sind interruptgesteuert und treten immer dann auf, wenn das Näherungsergebnis die oberen und/oder unteren Schwellenwerte überschreitet. Der Sensor verfügt außerdem über Offset-Einstellregister zur Kompensation des System-Offsets, der durch unerwünschte Infrarot-Energierflexionen am Sensor verursacht wird. Die Intensität der Infrarot-LED wird werkseitig eingestellt, so dass eine Kalibrierung der Endgeräte aufgrund von Bauteilschwankungen nicht erforderlich ist. Die Farb- und Umgebungslichterkennung liefert Daten zur roten, grünen, blauen Farbspektrum und Daten zum klaren Lichtintensität. Jeder der Kanäle R, G, B, C verfügt über einen Ultravioletter-Filter (UV-Filter)- und Infrarot-Sperrfilter und einen speziellen Datenkonverter, der gleichzeitig 16-Bit-Daten erzeugt. Diese Architektur ermöglicht Anwendungen eine genaue Messung des Umgebungslichts zu messen und die Farbe zu erkennen, was es den Geräten wiederum ermöglicht, die Farbtemperatur zu berechnen und die Hintergrundbeleuchtung des Displays dementsprechend anzupassen.[Ava15][Ard24a]

2.2.3 Barometrischer Drucksensor (LPS22HB)

Der LPS22HB ist ein sehr kompakter Absolutdrucksensor, der auf dem piezoresistiven Prinzip basiert. Er arbeitet als Barometer und verfügt über einen digitalen Ausgang. Zudem ist in diesem Drucksensor ein Temperatursensor integriert, mit welchem Druckmessungen zusätzlich kompensiert werden können. Das Gerät besteht aus einem Sensorelement und einer Inter Circuit (IC)-Schnittstelle, die eine Kommunikation zwischen der Sensoreinheit und der Anwendung über I²C oder Serial Peripheral Interface (SPI) ermöglicht. Das Sensorelement erfasst den absoluten Druck und besteht aus einer speziell gefertigten, hängenden Membran. Der LPS22HB ist in einem vollständig vergossenem Land-Grid-Array-Gehäuse untergebracht, das kleine Löcher aufweist. Durch diese Öffnung gelangt der externe Druck auf das Sensorelement. Der Betrieb des Sensors ist über einen Temperaturbereich von -40 °C bis +85 °C gewährleistet. [STM17][Ard24a] Anwendungsbereiche für diesen Sensor sind zum Beispiel: Wetterstationen, Höhenmesser, Luftdrucküberwachung in industriellen Prozessen oder tragbare smarte Geräte, wie Sportuhren oder Smartphones.

2.2.4 Digitales Mikrophon (MP34DT05)

Das MP34DT05-A ist ein kompaktes, stromsparendes, omnidirektionales, digitales Mikrofon mit einem kapazitiven Sensorelement und einer IC-Schnittstelle. Das Sensorelement, das in der Lage ist, akustische Wellen zu detektieren, wird mit einem speziellen Silizium-Mikrobearbeitungsverfahren für die Herstellung von Audiosensoren hergestellt. Die IC-Schnittstelle wird in einem speziellen Halbleiterherstellungsverfahren (CMOS-Verfahren[GW22][Ber20]) hergestellt, das die Entwicklung einer speziellen Schaltung ermöglicht, die ein digitales Signal im Pulsdichtenmodulation (PDM)-Format extern bereitstellen kann. Das Mikrofon hat einen Signal-Rausch-Abstand von 64 dB und eine Empfindlichkeit von -26 dBFS ± 3 dB. Sein maximaler Schalldruckpegel liegt bei 122,5 dB SPL. Der MP34DT05-A ist in einem Surface Mount Device (SMD)-kompatiblen, Electromagnetic Interference (EMI)-geschirmten Top-Port-Gehäuse erhältlich und arbeitet zuverlässig über einen erweiterten Temperaturbereich von -40 °C bis +85 °C. Abmessungen des Gehäuses HCLGA-4 LD sind $3 \times 4 \times 1$ (in mm). Anwendung findet das Modul Beispielsweise in mobilen Endgeräten, im Bereich der Spracherkennung sowie in Laptops und Notebooks.[STM21][Ard24a]

2.2.5 DC-DC-Wandler (MPM3610)

Das MPM3610-Modul ist ein integriert Gleichspannungs- zu Gleichspannungs-Wandler. Dieser kann Eingangsspannungen von bis zu 21 V verarbeiten mit einem Wirkungsgrad von mindestens 65% bei minimaler Last. Bei einer Eingangsspannung erhöht sich der Wirkungsgrad auf 85%. [Ard24a]

2.3 Beschreibung der Schnittstellen

Der Arduino Nano 33 BLE-Sense Lite bietet eine Vielzahl von Schnittstellen (siehe Abbildung 2.2), die das Board mit anderen Komponenten und Geräten einfach verbinden lassen. In diesem Abschnitt sind einige Details zu den wichtigsten Schnittstellen des Boards erläutert.

2.3.1 I²C

Der I²C-Bus ermöglicht die geräteinterne Kommunikation von Mikrocontrollern mit Sensoren, externen Bildschirmen, Echtzeituhren und vielen anderen Bausteinen. Das Kommunikationsprotokoll basiert auf dem Master-Slave-Prinzip. Der Master initiiert und beendet die Kommunikation, stellt den Takt zur Synchronisation bereit und löst Kommunikationsprobleme. Jeder Slave hat eine eindeutige Adresse, mit einer Länge von 7

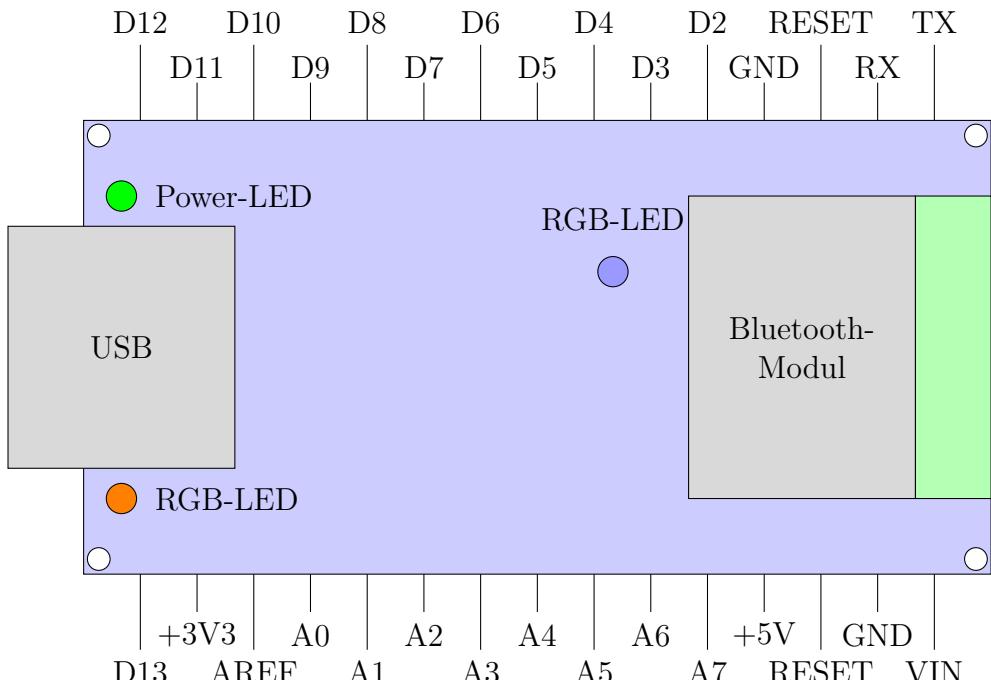


Abbildung 2.2: Arduino Nano Pinbelegung [Ard24a]

oder 10 Bit und ermöglicht die Adressierung von bis zu 128 bzw. 1024 unterschiedlichen Slaves in dem selben Bus. Geräte mit unterschiedlichen Adresslängen können im selben Bus koexistieren. Bevor der Master Daten überträgt oder empfängt, muss er einen Slave mit einer vorher vereinbarten Adresse ansprechen.[MS21][STM15][Ber20] Sind ihre Adressen in Übereinstimmung kann im nächsten Schritt ein einzelnes Bit gesendet werden, mit dem dann festgelegt wird, ob der Master Daten an dem Slave übertragen (Binär:1) oder auslesen möchte (Binär:0). Dieser Datenaustausch wird danach bestätigt, sodass weitere Daten ausgetauscht werden können. Zur Beendigung des Datenaustausches wird ein Stop-Signal gesendet.[GW22] In der minimalen Konfiguration werden Master und Slave über die bidirektionale Busleitungen Serial Data (SDA) und Serial Clock (SCL) verbunden, die über Pull-up-Widerstände an die Versorgungsspannung angeschlossen sind. Weitere Geräte können durch Verbindung ihrer SDA- und SCL-Anschlüsse mit den entsprechenden Busleitungen an den Bus angeschlossen werden.[MS21] In diesem Projekt dient der Arduino als Master und ein zusätzlich angeschlossenes Organic Light Emitting Diode (OLED)-Bildschirm als Slave.

2.3.2 USB

Das Board kann über einen Micro Universal Serial Bus (Micro USB) mit einem Computer verbunden werden, um es zu programmieren oder Daten zu übertragen. Die Datenübertragungsrate beträgt dabei 12 Mbit/s. Außerdem kann der Arduino über diesen Micro USB-Port Strom beziehen.

2.3.3 Bluetooth®5

Die Bluetooth-Verbindung kann als drahtloser Kommunikationsweg eingesetzt werden. Das Bluetooth-Protokoll hat eine Übertragungsrate von 2 Megabit pro Sekunde (Mbps) und eine Sendeleistung von +8 Dezibel Milliwatt (dBm). Die Empfindlichkeit beträgt dabei -95 dBm. Des weiteren verbraucht diese Verbindung im Sendebetrieb 4,8 mA und 4,6 mA im Empfangsbetrieb. Das Bluetooth-Modul ist kompatibel mit mehreren Protokollen, unter anderem mit dem *Thread-Protokoll* und dem *Zigbee-Protokoll*.[Ard24a] [Nor24a]

2.3.4 Weitere Kommunikationsschnittstellen

[Ard24a]

- **NFC-A-Tag:** Near Field Communication (NFC) ist eine zusätzliche Funktion zur drahtlosen Kommunikation über kurze Distanzen. Zudem besitzt der NFC-A-Tag die Funktionen, sich in einen Bereitschaftsmodus versetzen zu lassen, dass durch ein NFC-fähiges Gerät dann initiiert werden kann. Außerdem unterstützt es *touch-to-pair*, diese Funktion ermöglicht eine Kopplung mit anderen NFC-fähigen Geräten durch Berührung.
- **Arm CryptoCell CC310 Security Subsystem:** Für die Durchführung kryptografischer Operationen und Sicherheitsaufgaben. [Nor24c]
- **QSPI/SPI/TWI/I²S/PDM/QDEC:** Verschiedene weitere serielle Kommunikationsschnittstellen, die für den Datenaustausch verwendet werden können.
- **EasyDMA:** Direkt Memory Access (DMA) ist für die Übertragung von Daten zwischen verschiedenen Speicherbereichen, ohne dabei die CPU zu belasten.[GW22]
- **Analog to Digital Converter (ADC):** Wandelt analoge Eingangsignale in digitale Daten um. Der Wandler hat eine Auflösung von 12 Bit und eine maximale Abtastrate von 200 Kilosamples pro Sekunde (ksps).

- **128-Bit-AES/ECB/CCM/AAR-Co-Prozessor:** Co-Prozessor für kryptografische Operationen, der auf dem Advanced Encryption Standard (AES) basiert. Dieser unterstützt verschiedene Betriebsmodi wie Electronic Codebook (ECB), Counter with CBC-MAC (CCM) und Automatic Address Recognition (AAR). [Nor24b]
- **Quad-SPI-Schnittstelle 32 MHz:** SPI-Schnittstelle, die eine maximale Taktrate von 32 MHz unterstützt. Quad-SPI ermöglicht es, Daten schneller als die herkömmliches SPI zu übertragen, indem es vier Datenleitungen verwendet.[Nor23]

2.3.5 Digitale Ein- und Ausgangspins

Das Board verfügt über 14 digitale Ein- und Ausgangspins. Die digitalen Pins können nur zwei Zustände, nach dem Binär-System lesen: wenn ein Spannungssignal vorliegt und wenn kein Signal vorhanden ist (0 oder 1). Einige der Pins sind zudem zur Pulsweitenmodulation fähig (D3, D5, D6, D9, D10). Außerdem sind die digitalen Pins D11 und D12 als Master-Output-Slave-Input (MOSI) und als Master-Input-Slave-Output (MISO), in einer Serial-Peripheral-Interface (SPI) Kommunikation einsetzbar.[Ard24a]

2.3.6 Analoge Eingangspins

Die Platine hat zusätzlich 8 analoge Eingangspins (A0-A7), die wiederum als Analog-Digital-Wandler (ADC) verwendet werden können. Außerdem sind diese Pins als digitale Ein-/Ausgangspins konfigurierbar. Der Pin A0 kann zudem als Digital-Analog-Wandler (DAC) konfiguriert werden. Die beiden Pins A4 und A5 können außerdem für die I2C-Kommunikation verwendet werden. Dabei fungiert A4 als Datenleitung (SDA), während A5 als Takteleitung (SCL) fungiert.[Ard24a]

2.3.7 Weitere Pins

- **+3,3 V:** Erzeugt interne Stromquelle im Gerät und wird als Referenzspannung verwendet.
- **VIN:** Stromversorgung
- **5V:** Gibt 5V an die externen Komponenten ab.
- **RST-Pin:** Dient zum Zurücksetzen des Arduinos.
- **AREF-Pin:** Liefert die Spannungsreferenz, die der Mikrocontroller zur Zeit verwendet.
[Ard24a]

2.3.8 LED-Lampen

Im Arduino selbst sind 3 LED's verbaut, die auch alle programmiert werden können. Diese sind vor allem für die Überprüfung der Sensorik oder Softwareprogrammen nützlich. Zu den LED-Lampen gehören:

- Programmierbare Power-LED (grün): Zeigt an, dass das Arduino-Board eingeschaltet ist.
- Programmierbare LED (orange)
- Programmierbare RGB-LED
[Ard24a]

2.4 Hinweis Arduino 33 BLE Sense Lite

Der Arduino Nano 33 BLE Sense Lite ist eine komprimierte Variante vom ursprünglichen Arduino Nano 33 BLE Sense, welcher zusätzlich noch über einen Temperatur- und Feuchtigkeitssensor verfügt. Der Lite hat stattdessen einen Drucksensor integriert, über welchem auch die Temperatur gemessen werden kann, jedoch nicht die Feuchtigkeit.[PA22]

2.5 Bezugsquellen

Als Bezugsquellen dienten vor allem Datenblätter der Hersteller. Die meisten Informationen konnten dem Datenblatt des Arduino entnommen werden, jedoch ist dazu anzumerken, dass sich dieses Datenblatt auf den Arduino 33 BLE Sense bezieht und nicht auf den Arduino 33 BLE Sense *Lite*. Speziell zum *Lite* gibt es jedoch kein Datenblatt, so wurde das Datenblatt vom Arduino 33 BLE Sense herangezogen. Dies stellt sonst kein Problem dar, da sich, wie in 2.4 beschrieben, nur um eine komprimierte Version des Arduino 33 BLE Sense handelt.

3 Schrittmotor

In diesem Kapitel folgt eine grundsätzliche Beschreibung eines Schrittmotors. Darauf folgt die Erläuterungen zum Aufbau und Funktionsweise eines Schrittmotors sowie die Aufzählung verschiedener Grundbauarten. Nachfolgend werden Ursachen und Lösungen zum Verhindern von Schrittfehlern und der verwendete Schrittmotor genauer erläutert.

3.1 Beschreibung eines Schrittmotors

Ein Schrittmotor ist ein Elektromotor, der sich für präzise Positionierungsaufgaben eignet. Im Gegenteil zu anderen Elektromotoren wird bei einem Schrittmotor keine Positionsmeßung oder Positionsregelung benötigt. Andere mechanische Antriebssysteme benötigen einen geschlossenen Regelkreis und mechanische Bremsen um Drehzahl und Position einzuhalten. Der Motor führt durch die Rotation des Rotors diskrete Schritte aus, wobei jeder Steuerimpuls eine Verschiebung um einen konstanten Winkel bewirkt. Mit diesem Motor ist eine erreichbare Positioniergenauigkeit von $0,1^\circ$ möglich. Diese Art von Elektromotor wird beispielsweise in Druckern oder Scanner, aber auch im Kraftfahrzeugbereich verwendet. Im Kraftfahrzeugbereich werden die Schrittmotoren zur Spiegelverstellung sowie der Sitzverstellung verwendet. Das maximal erreichbare Drehmoment eines Schrittmotors liegt bei zwei Newtonmeter und die maximal erreichbare Drehzahl bei ca. 2000 Umdrehung pro Minute. Der Vorteil eines Schrittmotors ist die Wartungsfreiheit, da der Rotor keine Wicklungen hat. [Bab23][Hag21][Ber18][SK21]

3.1.1 Aufbau und Funktionsweise eines Schrittmotors

Der Schrittmotor besteht aus einem Stator, einem Rotor ohne Wicklungen und einer Steuerelektronik. Diese Steuerelektronik setzt sich zusammen aus einer Treiberstufe und der eigentlichen Steuerung. Bei dem Stator handelt es sich um den feststehenden äußeren Teil und bei dem Rotor um den beweglichen inneren Teil, wie in Abbildung 3.1 zu erkennen ist. Im Stator sind Spulen verbaut, die von einem Strom durchflossen werden. Hierdurch entsteht ein magnetisches Feld. Da der Rotor magnetisch ist, folgt er dem Magnetfeld des Stators. Soll eine Bewegung hervorgerufen werden, werden einzelne Wicklungsstränge ein- aus- oder umgeschaltet.

Durch diesen Vorgang wird ein rotierendes Magnetfeld erzeugt. Das erzeugte Magnetfeld zieht den Rotor an. Der Rotor wird bei jedem Puls (Takt) um einen Winkelschritt weitergeschaltet. Die Anzahl der Polpaare im Stator geben die Anzahl der Schritte vor. Die Drehzahl und die Drehrichtung hängt von der Reihenfolge und der Häufigkeit der Stromimpulse ab. Es gibt drei verschiedene Betriebsarten, die abhängig von der Genauigkeit und der Drehzahl sind. Im Vollschrittbetrieb werden alle Polpaare bestromt. In dem Halbschrittbetrieb wird die Schrittzahl des Motors verdoppelt, wodurch sich die Positionsauflösung im Gegensatz zum Vollschrittbetrieb verdoppelt. Allerdings wird in diesem Betrieb das Drehmoment reduziert. In dem Mikroschrittbetrieb bewegt sich der Rotor in sehr kleinen Schritten. Hierdurch wird eine hohe Positioniergenauigkeit und ein ruhiger Lauf erreicht, da die Ströme und das Drehmoment in kleineren Schritten verändert werden. Bei einer zu hohen Schrittzahl kann ein Schrittverlust entstehen. Bei einem Schrittverlust überspringt der Motor einzelne Winkelschritte und landet in einer vorherigen oder nächsten Position gleicher Phase. Durch die offene Steuerkette werden die Verluste nicht erkannt.[Hag21][Ber18][SK21]

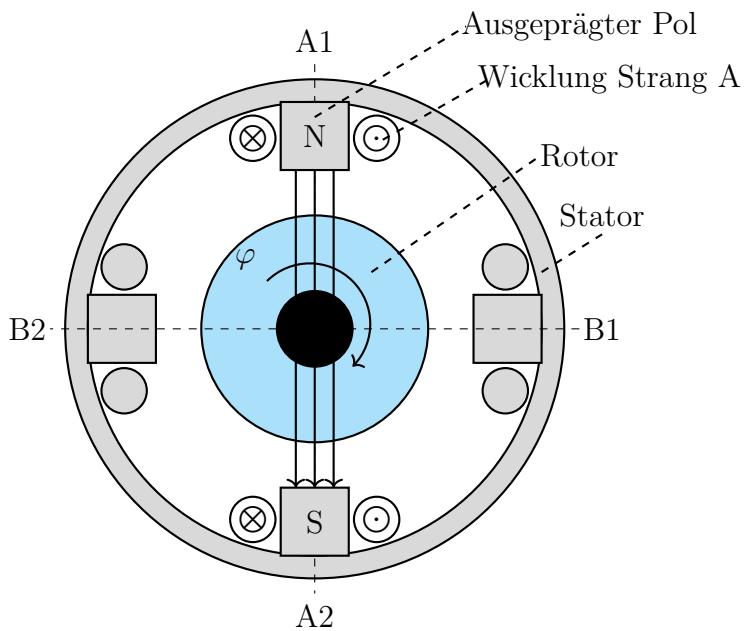


Abbildung 3.1: Prinzipieller Aufbau Schrittmotor (2-phasisig) [Hag21]

3.1.2 Schrittmotor Bauformen

Wie bei anderen Elektromotoren gibt es auch bei dem Schrittmotor verschiedene Grundbauarten.

- Permanentmagneterreger-Schrittmotor (PM-Schrittmotor)
- Reluktanzschrittmotor (VR-Schrittmotor)
- Hybridschrittmotor (Hybrid (hy)-Schrittmotor)

Der **permanentmagnetische Schrittmotor** hat einen Permanentmagneten in dem Rotor verbaut. Hierbei stellt sich der permanentmagnetische Rotor immer so, dass der Nordpol des Rotors dem Nordpol des Statorfeldes gegenüber liegt und der Südpol des Rotors dem Südpol des Statorfeldes. In dieser Ausrichtung ziehen sich die Pole gegenseitig an. Die Drehrichtung des Rotors hängt von der Fließrichtung des Stromes ab. Der permanentmagnetische Schrittmotor entwickelt im ausgeschalteten Zustand ein Drehmoment zur Selbsthaltung. Dies ist aufgrund des permanentmagnetischen Rotors möglich. Bei diesem Drehmoment handelt es sich um das höchste Drehmoment, das auf die Welle des Motors übertragen werden kann, ohne dass diese sich in eine rotierende Bewegung versetzt. Zu dieser Art von Schrittmotoren gehören beispielsweise der Klauenpol-Schrittmotor und der Scheibenmagnet-Schrittmotor. Bei der zweiten Bauart handelt es sich um den **Reluktanzschrittmotor**. Bei dieser Bauart besteht der Rotor aus einem weichmagnetischen Material und besitzt eine gezahnte Form. So lange der Schrittmotor von keinem Strom durchflossen wird, entsteht kein Magnetfeld. Sobald der Motor in Betrieb genommen wird, entsteht ein magnetischer Fluss innerhalb des Rotors. Wird nun eine Wicklung erregt, wird der nächste Zahn des Rotors angezogen. Dadurch, dass die Rotorzähne ungleich der Polteilung sind, kann das System unendlich lange fortgesetzt werden. Die Anzahl der Schritte und die Genauigkeit des Reluktanzschrittmotors ist abhängig von der Anzahl der Zähne auf dem Rotor. Aus technischer Sicht sind mit dieser Bauart Schrittwinkel unter 1° möglich. Damit die Drehrichtung verändert werden kann, sind mindestens zwei Strangwicklungen nötig. Bei den **Hybridschrittmotoren**, auch bekannt als hy-Schrittmotoren handelt es sich um eine Kombination aus dem Reluktanzschrittmotor und dem Permanentmagneterreger-Schrittmotor. Durch diese Kombination aus den beiden Schrittmotoren werden die Vorteile aus der kleinen Schrittweite, dem hohen Drehmoment und dem Selbsthaltemoment genutzt. Bei dem hy-Schrittmotor besteht der Rotor aus zwei um eine halbe Zahnteilung versetzten weichmagnetischen Polrädern, die eine zahnförmige Form haben. Bei den beiden Polrädern bildet das eine Polrad den Nordpol und das zweite Polrad den Südpol. Zwischen den beiden Polrädern befindet sich ein Permanentmagnet. Anders als bei anderen Schrittmotoren

wird der Rotor bei dieser Bauart axial magnetisiert. Damit ein kleiner Schrittinkel möglich ist, haben die Statorpole ebenfalls eine zahnförmige Form. Die Ausrichtung des Rotors ist abhängig von der Stromrichtung und wird durch den minimalen Widerstand bestimmt, der sich aus dem Stromfluss durch die einzelnen Stränge ergibt. Wird ein besonders kleiner Schrittinkel benötigt, kann dies durch Erhöhung der Zähnezahl erreicht werden. [SK21] [Hag21] [Bab23]

3.1.3 Betriebsarten unipolar und bipolar

Neben den oben bereits genannten Betriebsarten, kann außerdem zwischen dem Unipolarbetrieb und dem Bipolarbetrieb unterschieden werden. Der große Unterschied zwischen den beiden Betriebsarten besteht darin, dass in dem Unipolarbetrieb der Strom in eine Richtung fließt. Bei dem Bipolarbetrieb hingegen fließt der Strom in beide Richtungen. Dies ist möglich, da jeder Wicklungsstrang über eine Vollbrücke gespeist wird. Ein weiterer Unterschied besteht in der Schaltung der Zweige, durch die ein Gleichstrom fließt. In dem Unipolarbetrieb werden die beiden Zweige in Reihe geschaltet. Jeder Wicklungsstrang wird mit zwei Drähten parallel gewickelt. Sind die beiden Wicklungsstränge in dem Bipolarbetrieb parallel gewickelt, müssen die Zweige parallelgeschaltet werden. Im Bipolarbetrieb kann ein höherer Wirkungsgrad erzielt werden, wo hingegen der Unipolarbetrieb eine deutlich einfachere Schaltung aufweist. [SK21]

3.1.4 Mikroschrittverfahren

Mit dem Mikroschrittverfahren können Zwischenschritte und Mikroschritte erzeugt werden. Es bietet die Fähigkeit, Geräusche und Vibrationen zu minimieren sowie die Auflösung der Umdrehung zu erhöhen. Um Zwischenschritte zu erreichen, müssen beide Wicklungen eines Schrittmotors gleichzeitig mit Strom versorgt werden. Mit dem Sinus/Cosinus-Mikroschrittverfahren können die Mikroschrittverfahren realisiert werden, indem der Strom in jeder Phase des Motors sinusförmig variiert. Durch die Anwendung sinusförmiger Ströme auf die Motorwicklungen wird eine feinere und gleichmäßige Bewegung erzielt, was zur einer präziseren Steuerung und reduzierten mechanischen Vibrationen führt.[Mar24]

Ein Mikroschritt-Treiber unterteilt die Motorschrittinkel in vier oder mehr Teil. Es handelt sich um einen Controller, der die Methode der Stromreglung verwendet, um die Mikroschritte zu erzeugen. Der Controller bietet außerdem den Vorteil einer erhöhten Flexibilität bei der Integration der Strombegrenzung und der Anpassung der Antriebscharakteristik als Reaktion auf Änderungen der Systemdynamik. Die Anzahl der Unterteilungen ist einstellbar, was eine präzise Steuerung und Anpassung an spezifische Anforderungen ermöglicht.[Bab23]

3.2 Schrittverluste verhindern

Um mögliche Schrittverlusten einzugrenzen und oder sie zu verhindern, wird in diesem Kapitel beschrieben, wie die Ursachen für Schrittverluste oder Stillstand methodisch zu ermitteln sind. [Fau20]

3.2.1 Auswahl des Schrittmotors

Zunächst muss ein passender Motor für die Anwendung gewählt werden. Dabei sollten folgende grundlegende Regeln erfüllt sein:

- Motorauswahl durch Höchstwerte für Drehmoment und Drehzahl (Worst-Case-Szenario)
- Verwendung eines Sicherheitsaufschlag von 30 % auf die Drehmoment-Drehzahl-Kennlinie(Kippmoment)
- Sicherstellen, dass externe Ereignisse die Anwendung nicht blockieren können

Sind die geforderten Drehmomente bei den jeweiligen Drehzahlen, den Motorspezifikation entsprechend, dann sind keine Probleme zu erwarten. Ist der Motor zu schwach gewählt und die Anwendung fordert mehr Leistung als der Motor abgeben kann, so bleibt der Motor stehen. Der nächste Schritt ist die Durchführung von Testdurchläufen. Es soll im Betrieb überprüft werden, ob Schrittverluste auftreten. Schrittmotoren verlieren konstruktionsbedingt nicht nur einen einzigen Schritt. Bei geringen Drehzahlen verliert der Motor ein Vielfaches von vier Schritten.[Fau20]

3.2.2 Betriebsart

In diesem Abschnitt werden je nach Betriebsart mögliche Ursachen erläutert, falls der Schrittmotor bei den Tests versagt.

Start-Stopp-Betrieb

Der Motor ist mit der Last fest verbunden und wird mit konstanter Drehzahl betrieben. Innerhalb des ersten Schrittes muss der Motor auf die vorgegebene Frequenz beschleunigen wie in Abbildung 3.2 zu sehen ist.[Fau20]

Fehlerbild: Motor läuft nicht an (siehe Ursachen und Lösungen aus Tabelle 3.1)

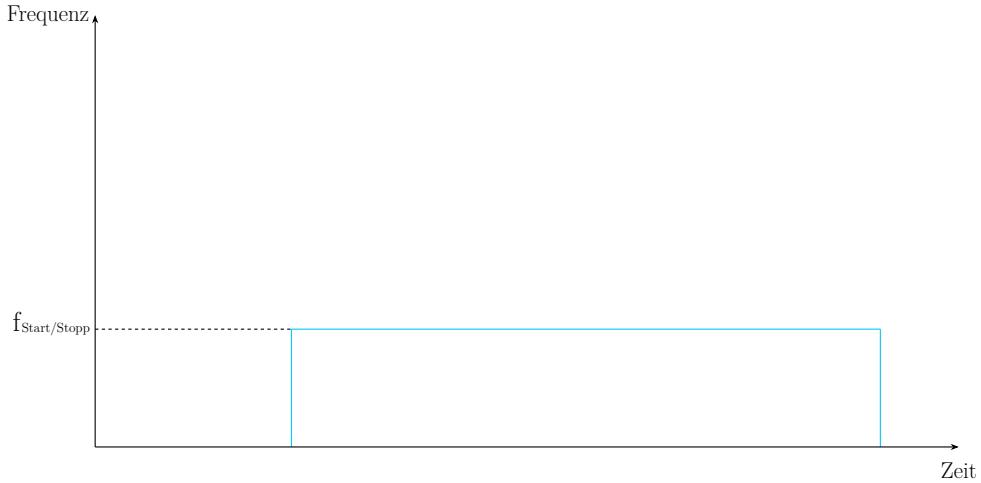


Abbildung 3.2: Start-Stopp Frequenz [Fau20]

Ursachen	Lösungen
Last zu Hoch	Falscher Motor, größeren Motor wählen
Frequenz zu hoch	Frequenz reduzieren
Pendelt der Motor von Links nach Rechts	Es könnte eine Phase unterbrochen oder nicht angeschlossen sein, dies muss repariert werden
Phasenstrom passt nicht	Phasenstrom erhöhen

Tabelle 3.1: Ursachen und Lösungen: Motor läuft nicht an [Fau20]

Beschleunigung und Rampenprofil (Trapezförmig)

Der Motor kann mit einer im Controller vorgegebenen Beschleunigungsrate bis auf die Maximalfrequenz beschleunigen wie in Abbildung 3.3 zu sehen ist.[Fau20]

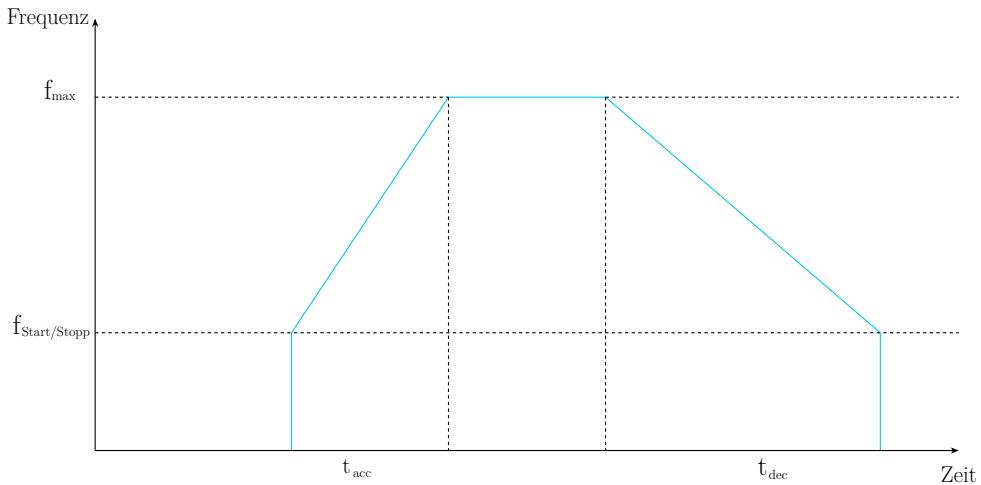


Abbildung 3.3: Trapezförmiges Geschwindigkeitsprofil [Fau20]

Fehlerbild: Motor beendet die Beschleunigungsrampe nicht (siehe Ursachen und Lösungen aus Tabelle 3.2)

Ursachen	Lösungen
Motor bleibt bei der Resonanzfrequenz hängen	<ul style="list-style-type: none"> Beschleunigung erhöhen, um die Resonanzfrequenz schneller zu durchlaufen Start-Stopp Frequenz über dem Resonanzpunkt wählen Halbschritt- oder Mikroschrittbetrieb verwenden Mechanische Dämpfung vorsehen
Falsche Einstellung von Versorgungsspannung oder Strom zu gering	<ul style="list-style-type: none"> Spannung oder Strom erhöhen Motor mit geringerer Impedanz testen Stromregelung verwenden
Maximaldrehzahl zu hoch	<ul style="list-style-type: none"> Maximaldrehzahl reduzieren Beschleunigungsrampe abflachen
Schlechte Vorgabe der Beschleunigungsrampe durch Elektronik	<ul style="list-style-type: none"> Anderen Controller verwenden

Tabelle 3.2: Ursachen und Lösungen: Motor beendet die Beschleunigungsrampe nicht [Fau20]

Fehlerbild: Motor beschleunigt bis zur Enddrehzahl und bleibt stehen, sobald eine konstante Drehzahl erreicht ist (siehe Ursachen und Lösungen aus Tabelle 3.3)

Ursachen	Lösungen
Motor wird an Leistungsgrenze betrieben und bleibt stehen aufgrund zu hoher Beschleunigung	<ul style="list-style-type: none"> Ruckeln verringern durch geringere Beschleunigungsrate oder durch unterschiedliche Beschleunigungsrampen, erst steil dann flacher Drehmoment erhöhen Motor im Mikroschrittbetrieb betreiben Mechanische Dämpfung vorsehen

Tabelle 3.3: Ursachen und Lösungen: Motor beschleunigt bis zur Enddrehzahl und bleibt stehen, sobald eine konstante Drehzahl erreicht ist. [Fau20]

3.2.3 Externe Ereignisse

Lastrückkopplung

„Manchmal wird der vom Motor angetriebene Mechanismus/Last während der Bewegung „aufgezogen“ und gibt diese Energie wieder an den Motor zurück, wenn die Ströme ausgeschaltet werden. Der Mechanismus könnte z.B. ein Untersetzungsgetriebe sein.“ [Fau20] Wird diese Energie an den Motor zurück geleitet, kann es passieren, dass der Motor sich um einen Winkel verdreht, der mehr als einem Schritt entspricht. Dabei kann es passieren, dass der Motor nicht ausreichend Drehmoment entwickelt und nicht oder erst nach 4 Vollschritten anläuft. [Fau20]

Lösungen:

- Die Kommutierung so programmieren, dass Wert und Polarität vor dem Abschalten gespeichert und beim Wiedereinschalten verwenden werden kann.
- Nicht vollständig den Strom abschalten, sondern bei Motorstillstand einen reduzierten *Stand-By* Strom aufrechterhalten

Erhöhung der Nutzlast mit der Zeit

„Manchmal läuft der Motor für eine lange Zeit störungsfrei und viel später treten die ersten Schrittverluste auf. In diesem Fall ist es sehr wahrscheinlich, dass die Last, die der Motor „sieht“, sich geändert hat. Das kann auf Verschleiß der Motorlager oder ein externes Ereignis zurückzuführen sein.“ [Fau20]

Lösungen:

- Prüfen ob ein externes Ereignis durch Veränderung des Mechanismus vorliegt.
- Prüfen ob Lagerverschleiß vorhanden ist. Verwendung von Kugellagern erhöhen die Lebensdauer des Motors.
- Verwendung von Schmiermittel um Reibung zu verhindern.

3.3 Beschreibung des verwendeten Schrittmotors

Für das Automatisierungsprojekt wird ein Nema 17 Schrittmotor der Firma Creality3D verwendet. Dieser Schrittmotor wird im 3D-Druck sowie in CNC-Maschinen eingesetzt. Der Motor arbeitet im Bipolarbetrieb und es sind zwei Spulen verbaut. Er hat einen Schrittewinkel von $1,8^\circ$ und benötigt 200 Schritte für eine volle Umdrehung. Die Wellenlänge

beträgt 20 mm und der Wellendurchmesser 5 mm. Der ausgewählte Schrittmotor arbeitet mit einer Nennspannung von 12 Volt. Der Motor ist nicht für große Drehmomente ausgelegt, sondern für schnelle und präzise Bewegungen. Dies wird qualitativ in der Abbildung 3.4 für den Schrittmotor GM42BYG015 dargestellt. Hier ist zu erkennen, dass der Schrittmotor ein Drehmoment von ca. 0,8 kg·cm erreicht. Auf der x-Achse wird die Geschwindigkeit in Pulses per Second (PPS) angegeben, d.h. die Anzahl der Steuerimpulse pro Sekunde. Der Schrittmotor erzielt aber eine gute Positioniergenauigkeit durch den kleinen Schrittewinkel und die Microstepp-Funktion. Des weiteren arbeitet der Motor mit 0,84 Amper pro Phase. Weitere Vorteile des Schrittmotors sind die kompakte Bauform mit den Abmessung $42 \times 42 \times 34$ mm sowie die geringe Masse mit 220 Gramm. Betrieben werden kann der Schrittmotor in einer Umgebungstemperatur von -10 °C bis +50 °C.[Glo]

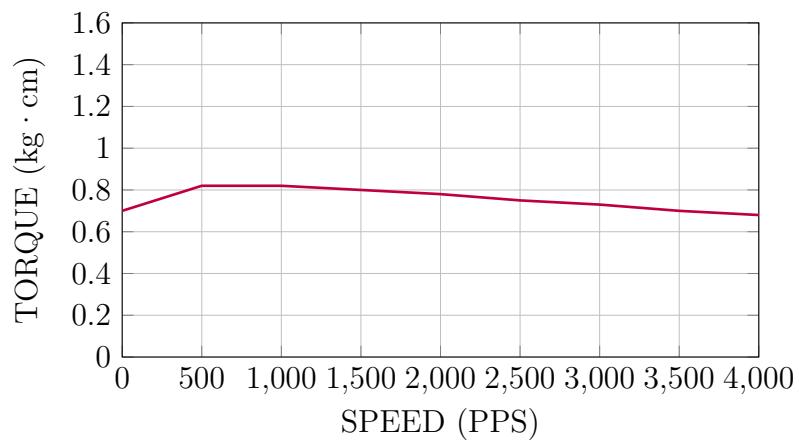


Abbildung 3.4: Drehmoment-Diagramm für den Schrittmotor 42BYG015
[Glo]

4 Weitere Hardware

In den folgenden Kapiteln werden die wichtigsten zusätzlich benötigten Hardwarekomponenten aufgelistet und erläutert. Dazu zählen Aktoren zur Darstellung des Betriebszustandes, Schalter und weitere Komponenten, die für den Aufbau des Schrittmotor-Demonstrators nötig waren.

4.1 OLED-Display

Zur Feststellung des aktuellen Betriebszustandes und zur Ausgabe der eingestellten Bewegungsstufe des Schlittens dient ein 1,3 Zoll OLED-Display (*Organic Light Emitting Diode*). Bei OLED-Displays sind jeweils drei OLEDs für einen Bildpunkt zuständig und benötigen keine Hintergrundbeleuchtung oder LC-Zellen, wodurch ein scharfer Kontrast möglich ist. OLED-Displays zeichnen sich durch die Reaktionsschnelligkeit aus. Die Helligkeit eines Pixels wechselt in weniger als einer Mikrosekunde.[Die19] Insgesamt ist das Modul $36\text{ mm} \times 34\text{ mm} \times 36\text{ mm}$ groß mit einer Bildschirmdiagonalen von 1,3 Zoll. Das Display besteht aus 128×64 weißen OLED-Bildpunkten. Durch die I²C-Kommunikation kann das Display mit dem Arduino verbunden werden. Zur Stromversorgung wird 3,3 V Gleichstromspannung, bei einem Stromverbrauch von weniger als 11 mA, benötigt. Mithilfe von 4 Steckbrückenkabel ist es mit dem Arduino verbunden, um es mit der nötigen Spannung zu versorgen und die I²C-Verbindung herzustellen (Verdrahtung detailliert in Kapitel 10 Schaltplan). Die zur I²C-Kommunikation nötige Adresse ist 0x3F. Zusätzliche Spezifikationen des Displays sind:

- Pixelgröße: $0,21\text{ mm} \times 0,21\text{ mm}$
- Pixelabstand: $0,23\text{ mm} \times 0,23\text{ mm}$
- Anzeigemodus: Passive Matrix
- Pixelfarbe: Blau

[AD24]

4.2 Signalleuchte

Zusätzlich zum OLED-Display ist eine SMD-LED-Signalleuchte zur Zustandserkennung verbaut. Sie soll Störungen und Fehler erkenntlich machen. SMD steht für *Surface Mount Device* und bedeutet, dass die Signalleuchte für Oberflächenmontagen konzipiert ist. Die Leuchte ist dabei mit einem einfachen Stecksystem am Gehäuse befestigt. Dazu wird sie in einer 8 mm Montagebohrung des Gehäuses gesteckt und über die 10 mm Gehäuseblende der Signalleuchte fixiert. Die Leuchte hat eine 3 mm im Durchmesser große LED und emittiert ein rotes Licht. Die Betriebsspannung liegt zwischen 2 bis 2,4 V und der Betriebsstrom bei 20 mA. Angeschlossen wird sie über vier Litzen am Arduino (siehe 10 Schaltplan). [Men24]

4.3 Spannungswandler

Für den Aufbau des Demonstrators sind mehrere Spannungswandler benötigt. Folgende Wandler wurden in der Schaltung verwendet.

4.3.1 Netzteil SNT RD 50A

Dieser Wechselspannung-/Gleichspannungswandler (AC/DC-Wandler) wandelt die 230 V Wechselspannung des Netzanschlusses in 12 V und 5 V Gleichspannung um. Weitere Spezifikationen des Wandlers sind:

- **Bauteilabmessung** ($L \times B \times H$): 99 mm × 97 mm × 36 mm
- **Ausgangsstrom**: 2 A
- **Leistung**: 54 W
- **Wirkungsgrad**: 79 %

[Mea19]

4.3.2 Spannungswandler ASM1117

Dieser Spannungswandler wandelt die 5 V des Netzteils SNT RD 50A in jeweils 3,3 V. Die 3,3 V werden für den Arduino, Schrittmotorsteuerung, OLED-Display und dem Drehwinkel-Encoder verwendet. Weitere Technische Daten des Wandlers sind:

- **Bauteilabmessung** ($L \times B \times H$): 40 mm × 40 mm × 20 mm
- **Ausgangsstrom**: 800 mA

[Adc24]

4.4 Linearführung

Zur Demonstration einer Linearbewegung wird eine kompakte Linearführung verwendet. Die Führung ist 450 mm lang und 12 mm breit. Diese Linearführung wird vor allem in Fused Deposition Modeling (FDM) -Drucker verwendet, wo es auf hohe Präzision bei niedrigen Toleranzen ankommt. Daher ist diese Führung auch für dieses Projekt gut geeignet, bei dem es nicht darum geht, große Lasten zu bewegen, sondern die Schrittauflösung möglichst genau auf die Millimeterskala zu übertragen.[RB24]

4.5 Drehwinkel-Encoder

Der Demonstrator soll mehrere Programme fahren können, deswegen wurde ein Drehwinkel-Encoder der Steuerung hinzugefügt. Dieser wird über drei Pins am Arduino angeschlossen und über zwei weitere Pins wird er mit Spannung versorgt. Durch drehen des Drehschalters werden nacheinander drei Kontakte geschlossen oder geöffnet (ein Kontakt ist immer geschlossen). Dieser dadurch entstehende Signalfluss, bestehend aus zwei um 90 Grad versetzte Sinus bzw. Cosinusschwingungen werden ausgewertet. Daraus wird bestimmt, in welche Richtung (im oder gegen Uhrzeigersinn) und wie weit (inkrementell) gedreht wurde. Mithilfe dieser Logik kann durch ein Menü eine Bewegungsstufe ausgewählt werden, die der Schrittmotor fahren soll.[Bas16] Bei einer Drehung im Uhrzeigersinn wird im Menü eine Bewegungsstufe höher und bei einer Drehung gegen den Uhrzeigersinn eine Bewegungsstufe niedriger ausgewählt. Zusätzlich zum Drehwinkel-Encoder ist auch noch ein Schaltfunktion im Bauteil selbst integriert. Durch eindrücken des Encoders wird ein Taster betätigt, durch denn der eingestellte Wert bestätigt und an den Arduino zur weiteren Verarbeitung weitergegeben wird. Zur Besseren Handhabung des Drehwinkel-Encoders wurde noch ein Drehgriff angefertigt und auf dem Drehgeber montiert. Weitere Details:

- **Abmessungen ($b \times l \times h$):** $18 \text{ mm} \times 31 \text{ mm} \times 30 \text{ mm}$
- **Betriebsspannung:** 3,3 - 5 V [Sim19]

4.6 Schrittmotorsteuerung

Für eine einfachere Bedienung des Schrittmotors wurde ein Schrittmotortreiber mit integriertem Übersetzer verbaut. Bei einer Leistung von bis zu 35 V und $\pm 2 \text{ A}$ ist der DEBO DRV A4988 für den Betrieb von bipolaren Schrittmotoren ausgelegt. Auf dem Treiber ist ein Stromregler mit fester Ausschaltzeit integriert, bei dem zwischen einem langsamen und einem

gemischten Abschaltmodus gewählt werden kann. Durch eine Impulseingabe wird der Motor um einen Mikroschritt angetrieben. Der Vorteil der Schrittmotorsteuerung liegt darin, dass keine Phasensequenztabellen, Hochfrequenz-Steuerleitungen oder komplexe Schnittstellen programmiert werden müssen. Im Schrittbetrieb wählt der A4988 automatisch den Abklingmodus, langsam oder gemischt. Im gemischten Modus wird im Abklingvorgang zwischen dem schnellen und langsamen Modus gewechselt. Dabei führt der gemischte Modus zu einer Verringerung der hörbaren Motorgeräusche, einer höheren Schrittgenauigkeit und einer geringeren Verlustleistung. Weitere Details:

- **Abmessungen** ($b \times l \times h$): $5 \text{ mm} \times 5 \text{ mm} \times 0,9 \text{ mm}$
- **Steuerspannung:** 3,3 - 5 V
- **Maximale Ausgangskapazität:** bis 35 V und $\mp 2 \text{ A}$
- **Schutzschaltungen:** Thermische Abschaltung, Schutz vor Masseschluss, Schutz vor kurzgeschlossener Last, Überkreuzungsstromschutz [All22]

5 Beschreibung der Software IDE

5.1 Installation der Arduino IDE

Die Programmierung von Mikrocontrollern, wie dem verwendeten Arduino Nano 33 BLE Sense Lite, erfolgt unter Verwendung der Software Arduino IDE in der Version 2.3.2. Als Alternative kann auch die Entwicklungsumgebung Qt verwendet werden, welche die Programmierung in der Programmiersprache C++ ermöglicht. Vorteil der Arduino IDE besteht in ihren Funktionen, welche das Einbinden der Software auf dem Mikrocontroller vereinfachen. Die Software Arduino IDE kann über die offizielle Webseite von Arduino bezogen werden, welche unter der folgenden URL erreichbar ist: www.arduino.cc/software erreichbar ist. Auf dieser Plattform stehen diverse Versionen, für unterschiedliche Betriebssysteme, wie Windows, Linux und Mac OS X, sowie in verschiedenen Dateiformaten, zur Auswahl bereit [Ard24b]. Die Installation der Anwendung erfolgt durch die Ausführung der heruntergeladenen Datei sowie die Befolgung der Installationsanweisungen. Im Anschluss ist eine korrekte Konfiguration des verwendeten Entwicklungsbretts sowie die Installation entsprechender Bibliotheken erforderlich.

5.2 Beschreibung der Entwicklungsumgebung

Nach dem Start der installierten Anwendung öffnet sich die Hauptoberfläche, erkennbar in Abbildung 5.1

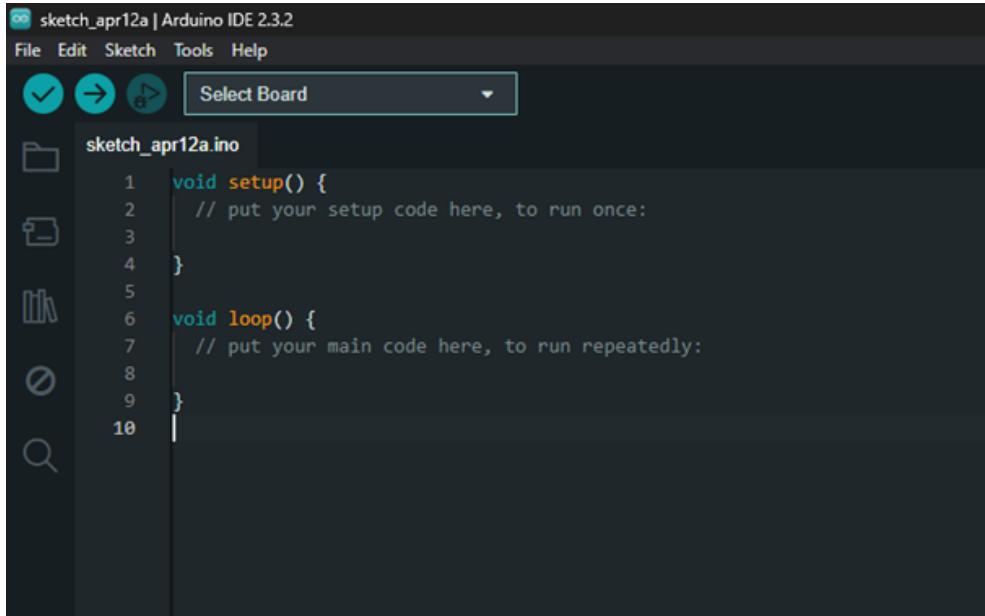


Abbildung 5.1: Hauptoberfläche in der Arduino IDE (Eigenaufnahme)

Die Hauptoberfläche ist mit einer Menüleiste ausgestattet, die verschiedene Menüs wie *Datei*, *Bearbeiten* und *Sketch* enthält. Diese Menüs ermöglichen das Bearbeiten und Öffnen von Sketches, also Programmen in der IDE, das Kompilieren und Hochladen von Code sowie das Verwalten von Bibliotheken. Zudem ist in der Menüleiste ein Hilfe-Menü integriert, das bei Fragen und Problemen Unterstützung bietet. Die Symbolleiste auf der linken Seite der Hauptoberfläche bietet Schaltflächen für häufig genutzte Funktionen wie das Verwalten von Sketches, Boards und Bibliotheken. Im Zentrum der Hauptoberfläche befindet sich ein Code-Editor, welcher die Bearbeitung der Sketches ermöglicht. Nach dem ersten Kompilieren eines Sketches erscheint darunter ein Ausgabefenster, welches den Kompilierungs- und Hochladeprozess sowie eventuelle Fehler während der Kompilierung anzeigt. Dadurch können Probleme im Code schnell identifiziert werden.

5.3 Erste Schritte in der Entwicklungsumgebung

5.3.1 Auswahl des Mikrocontrollers

In der Entwicklungsumgebung können Sketche geöffnet und geschrieben werden. Um den Sketch kompilieren zu können, ist es erforderlich das korrekte Board auszuwählen. Dafür muss zunächst das entsprechende

Board installiert werden. Dies erfolgt über den *Boards-Manager*, erkennbar in Abbildung 5.2.

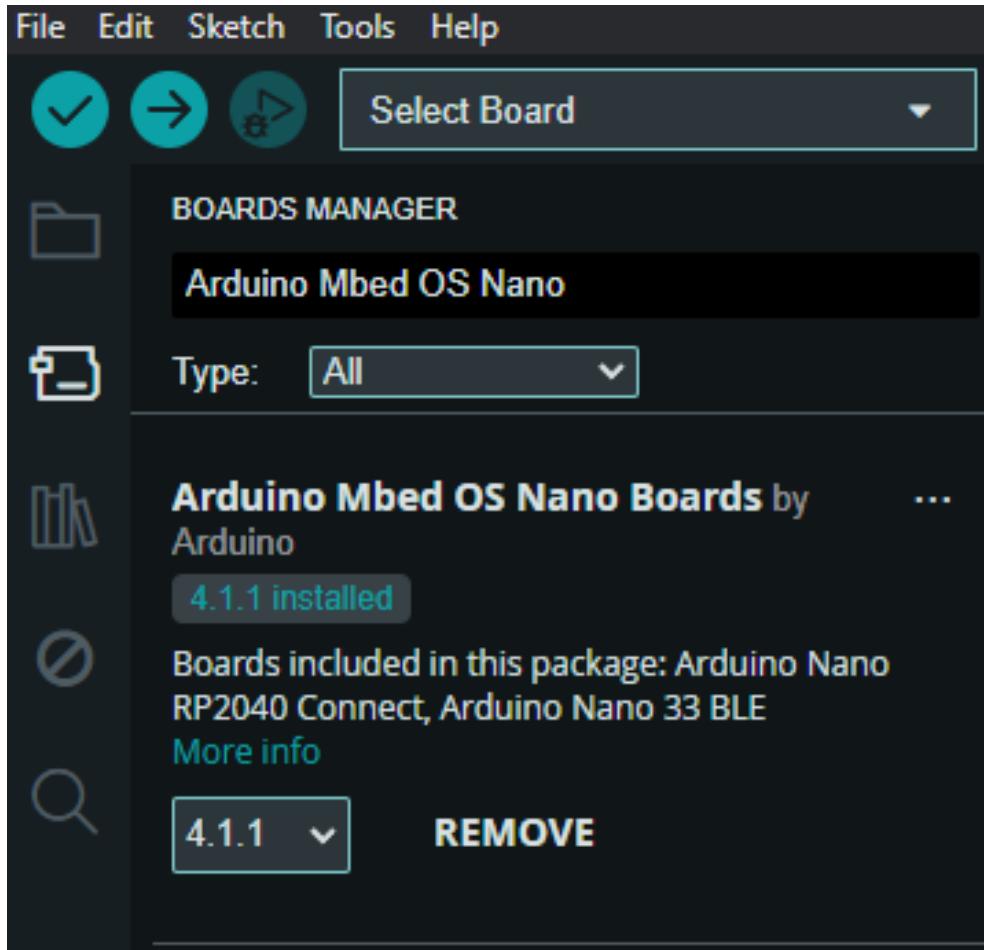


Abbildung 5.2: Installation des Boards (Eigenaufnahme)

Im *Boards Manager* steht die Datei Arduino Mbed OS Nano in der aktuellen Version 4.1.1 zum heruntergeladen und installieren bereit. Im Anschluss kann unter *Select Board* das Board Arduino BLE Sense 33 ausgewählt werden [Ard24d]. Alternativ kann das Board über ein USB-Kabel angeschlossen werden. Unter *Select Board* wird bereits das korrekte Board und der entsprechende COM, also in diesem Fall der USB-Port, angeboten. Nach der Auswahl sind die Installationsanweisungen zu befolgen, um das Board zu installieren.

5.3.2 Bibliotheken einbinden

Von Arduino gibt es bereits viele offiziell unterstützte Bibliotheken. Um diese in einem Sketch nutzen zu können, ist zunächst eine Installation er-

forderlich. Dazu kann über den *Library Manager* die benötigte Bibliothek heruntergeladen und installiert werden, erkennbar in Abbildung 5.3. Die Suchfunktion hilft dabei, die korrekte Bibliothek zu finden [Ard24c].

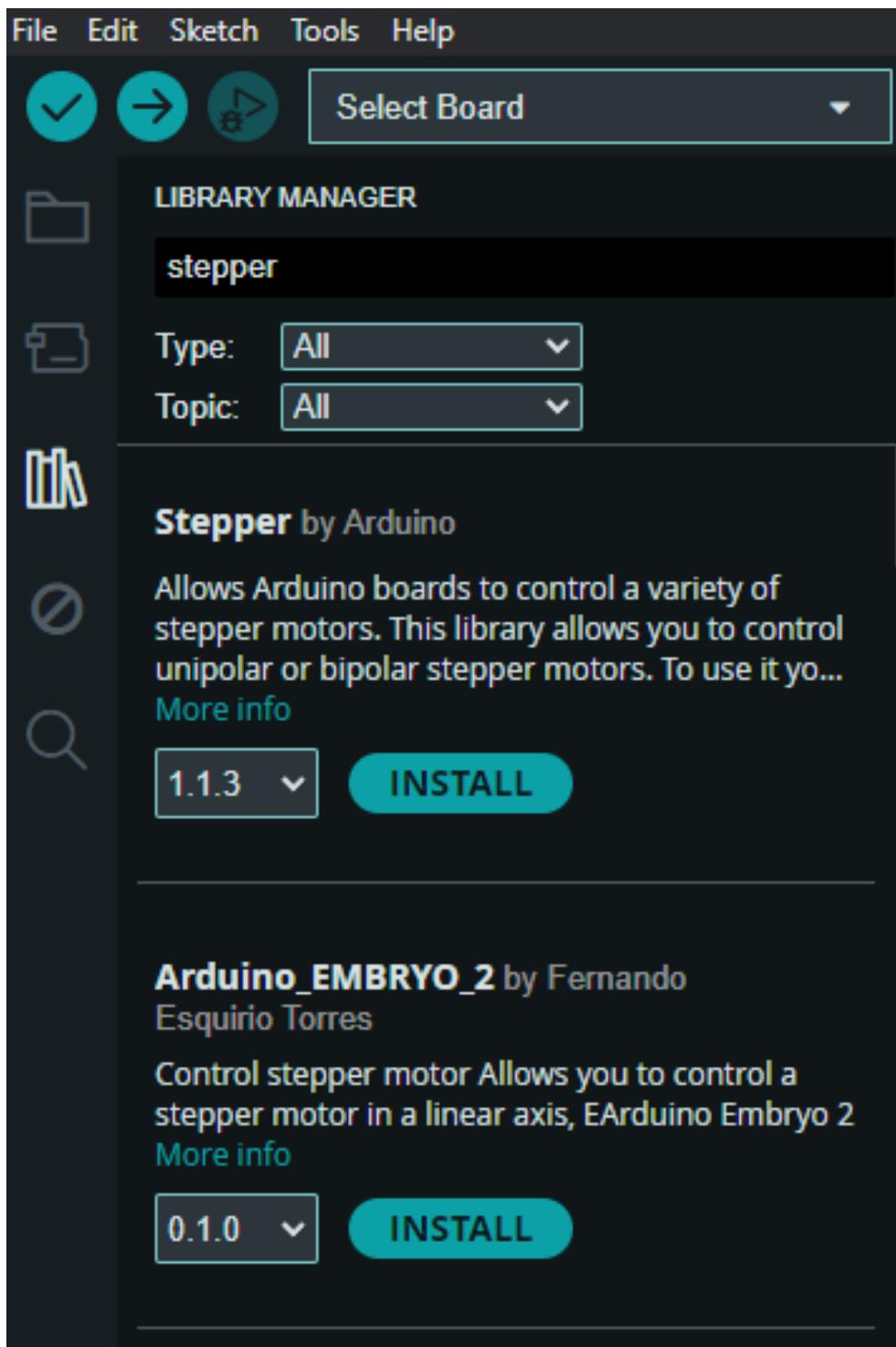


Abbildung 5.3: Herunterladen von Bibliotheken (Eigenaufnahme)

Nachdem die Bibliothek installiert wurde, kann sie im header in den Sketch eingebunden werden. Bei Verwendung von Bibliotheken, die nicht

im *Library Manager* zu finden sind, besteht die Möglichkeit diese über *Sketch->Include Libraries->Add .ZIP Library...* einzubinden.

5.4 Programmierung

Bei einem neuen Sketch sind bereits die Funktionen *void setup()* und *void loop()* hinterlegt.

5.4.1 header

Im *header* werden die erforderlichen Bibliotheken hinterlegt und initialisiert. Gleichzeitig können hier auch Adressen von Peripheriegeräten hinterlegt und erste wichtige Variablen definiert werden.

5.4.2 setup()

Die *void setup()*-Funktion wird bei jedem Neustart oder Reset einmal ausgeführt [Ard24f]. In diesem Beispiel wird die serielle Kommunikation gestartet, aber auch verwendete Pins initialisiert und zugewiesen.

5.4.3 loop()

In der *void loop()*-Funktion findet der größte Teil des Programms statt. Der Inhalt dieser Funktion wird dauerhaft wiederholt, bis entweder kein Strom mehr an dem Arduino anliegt, oder der Reset-Knopf gedrückt wird, wodurch zunächst erst die *void setup()*-Funktion wieder gestartet wird [Ard24e].

5.5 Erster Programmtest

Um die Funktion eines Systems zu testen, wird oft zunächst ein sehr simples Programm oder eine sehr grundlegende Funktion getestet. In diesem Fall kann hier über *Datei -> Beispiele -> 01.Basics -> Blink* ein Sketch geöffnet werden, in dem die auf dem Arduino Nano 33 BLE Sense Lite aufgebrachte LED in einem fest gelegten Takt blinkt, wodurch die korrekte Auswahl des Boards und die Übertragung des Sketches auf den Mikrocontroller getestet werden können.

5.6 Erster Programmtest

Um die Funktionalität eines Systems zu testen, wird in der Regel zunächst ein simples Programm oder eine grundlegende Funktion getestet. In diesem

```
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
}
```

Listing 5.1: Blink.py

Fall kann hier über *Datei -> Beispiele -> 01.Basics -> Blink* ein Sketch geöffnet werden, in dem die auf dem Arduino Nano 33 BLE Sense Lite aufgebrachte LED in einem fest gelegten Takt blinkt. Auf diese Weise kann die korrekte Auswahl des Boards und die Übertragung des Sketches auf den Mikrocontroller getestet werden. 5.1

In der *void setup()*-Funktion wird in diesem Beispiel über den Ausdruck *pinMode(LED_BUILTIN, OUTPUT)* die auf dem Mikrocontroller verbaute LED in dem Sketch aufgerufen und eingebunden. In *void loop()* wird durch den Befehl *digitalWrite()* die LED eingeschaltet, indem eine Spannung an sie angelegt wird. Nach einer kurzen Verzögerung durch den gleichen Befehl wird die LED dann wieder ausgeschaltet, indem die anliegende Spannung gesenkt wird. Die Verzögerung kann mit dem *delay()*-Befehl eingestellt werden, indem der Wert in der Klammer angepasst wird. Dabei wird der Wert in Millisekunden angegeben.[Ard14]

6 Beschreibung des Programms auf dem Arduino

6.1 Aufgabe des Programms

Das Programm steuert einen Schrittmotor und zeigt Informationen auf einem OLED-Display an. Es nutzt einen Drehwinkel-Encoder, um verschiedene Betriebsstufen des Motors auszuwählen.

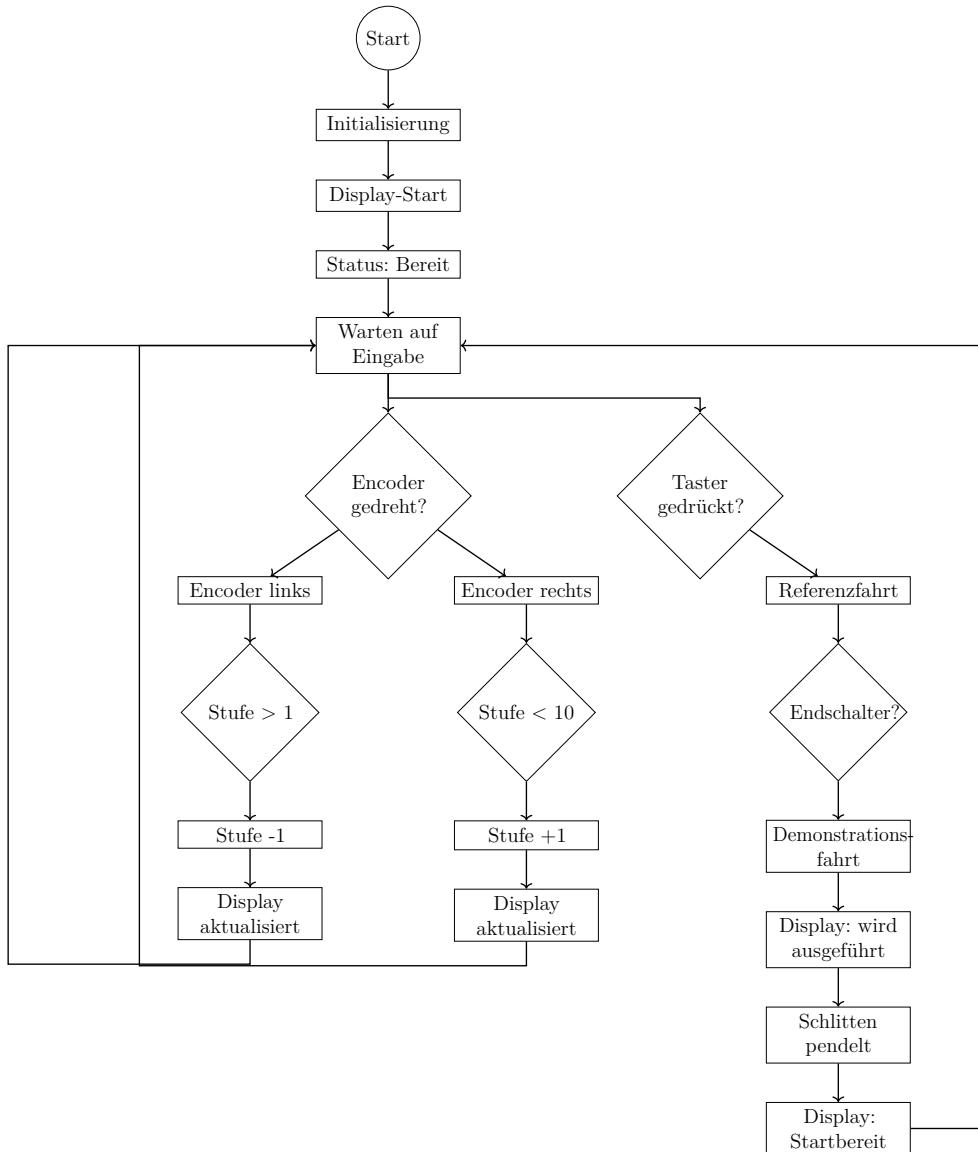


Abbildung 6.1: Flussdiagramm des Programms

Beim Start des Programms werden alle Komponenten, einschließlich des Schrittmotors, des Displays, der LEDs und des Encoders, initialisiert. Nach einer kurzen Startanimation auf dem Display wechselt das System in den Bereitschaftsmodus. Der Benutzer kann den Drehwinkel-Encoder drehen, um eine der zehn verfügbaren Stufen auszuwählen. Jede Stufe hat spezifische Geschwindigkeitswerte für den Schrittmotor. Die aktuell ausgewählte Stufe wird auf dem OLED-Display angezeigt, sodass der Benutzer immer über die gewählte Einstellung informiert ist. Ein Taster ermöglicht

```
#include "AccelStepper.h"
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Encoder.h>
```

Listing 6.1: Einbindung der Bibliotheken

es dem Benutzer, den Startprozess des Schrittmotors zu initiieren. Nach dem Drücken des Tasters führt der Schrittmotor Bewegungen aus, die der ausgewählten Stufe entsprechen. Diese Bewegungen beinhalten das Fahren zu einem festgelegten Punkt und das anschließende Zurückfahren zur Ausgangsposition. Zusammengefasst ermöglicht das Programm dem Benutzer, durch Drehen des Encoders verschiedene Betriebsstufen auszuwählen und den Schrittmotor zu steuern. Die Stufenanzeige auf dem Display und die LED-Statusanzeigen bieten dabei eine klare und leicht verständliche Rückmeldung über den aktuellen Zustand des Systems. Der Ablauf des Programms ist in Abbildung 6.1 dargestellt.

6.2 Erklärung des Codes

Das Programm zur Steuerung des Demonstrators wurde zunächst aus Teilen der Testprogramme für jede Hardware-Komponente basierend auf der gewünschten Funktion zusammengefügt. Nachdem die gewünschte Funktionalität erreicht war, wurde der Code mit Hilfe des Large-Language-Models Chat GPT in der Version 4.0 des US-Unternehmens OpenAI Inc. bezüglich der Lesbarkeit und Vereinheitlichung Benennung von Variablen. Zuerst werden die notwendigen Bibliotheken eingebunden, die für die Steuerung des Schrittmotors, das Display und den Encoder erforderlich sind. 6.1

Die define-Anweisungen legen Konstanten fest, wie die Art des Motorinterfaces, die Abmessungen des Displays und die Pin-Nummern für den Stepper-Motor, die LEDs, den Taster, den Endschalter und den Encoder. 6.2

Nun werden die Objekte für den Stepper-Motor, das Display und den Encoder erstellt.

```
AccelStepper stepper(MOTOR_INTERFACE_TYPE, PIN_STEP_STEP, PIN_STEP_DIR);
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
Encoder encoder(PIN_DT, PIN_CLK);
```

Die globalen Variablen speichern den Status des Tasters, die Zeit der letzten Zustandsänderung, die aktuelle Stufe sowie die Geschwindigkeiten und Beschleunigungen für jede Stufe. 6.3

```
#define MOTOR_INTERFACE_TYPE 1
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C

#define PIN_STEP_DIR 4
#define PIN_STEP_STEP 5
#define PIN_LED_RED 6
#define PIN_LED_GREEN 3
#define PIN_LED_BLUE 9
#define PIN_BUTTON 11
#define PIN_ENDSTOP 12
#define PIN_CLK 9
#define PIN_DT 2
#define PIN_SW 3
```

Listing 6.2: define-Anweisungen

In der setup-Funktion werden die Startwerte für die maximale Geschwindigkeit und Beschleunigung des Schrittmotors gesetzt, die Pins für LEDs, Taster und Endschalter initialisiert, die serielle Kommunikation gestartet und das Display initialisiert. Ein Interrupt für den Encoder-Taster wird festgelegt. Das Display zeigt eine Startanimation, und die LEDs werden auf ihre Startzustände gesetzt. 6.4

Die loop-Funktion wird kontinuierlich ausgeführt. Sie liest die Position des Encoders aus und zeigt bei einer Änderung die entsprechende Stufe auf dem Display an. Der Zustand des Tasters wird abgefragt und bei Betätigung die Referenzfahrt des Motors gestartet. Nach der Referenzfahrt führt der Motor Bewegungen gemäß den eingestellten Stufen und Geschwindigkeiten aus. 6.5

Die updateLEDs-Funktion aktualisiert den Zustand der LEDs basierend auf den übergebenen Parametern. 6.6

Die updateDisplay-Funktion aktualisiert das Display mit den übergebenen Nachrichten. 6.7

Die findReference-Funktion führt die Referenzfahrt aus und überprüft den Endschalter. Wenn der Endschalter gedrückt ist, stoppt der Motor und die Funktion gibt true zurück. 6.8

Die handleInterrupt-Funktion wird aufgerufen, wenn der Encoder-Taster gedrückt wird. Sie stoppt den Motor und aktualisiert das Display mit einer Abbruchnachricht. 6.9

```

int buttonStatus = HIGH;
int lastButtonStatus = HIGH;
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;
int endstopStatus = HIGH;
int lastEndstopStatus = HIGH;
unsigned long lastEndstopDebounceTime = 0;
int stage = 0;
int speeds [] = {1000,
                  500,
                  750,
                  1000,
                  1250,
                  1500,
                  1750,
                  2000,
                  2250,
                  2500,
                  1750};
int accelerations [] = {1000,
                        6000,
                        6000,
                        6000,
                        6000,
                        6000,
                        6000,
                        6000,
                        6000,
                        6000,
                        6000,
                        6000,
                        6000,
                        6000,
                        6000};
long oldPosition = -999;
const int safeSteps = 70;
const int trackLength = 1750;
const char* stageMessages [] =
{"Keine\u2022Stufe\u2022eingestellt",
 "Stufe\u20221",
 "Stufe\u20222",
 "Stufe\u20223",
 "Stufe\u20224",
 "Stufe\u20225",
 "Stufe\u20226",
 "Stufe\u20227",
 "Stufe\u20228",
 "Stufe\u20229",
 "Stufe\u202210"};
const char* messages [] =
{"Start ...",
 "Startbereit",
 "Programm\u2022wird\u2022ausgefuehrt ...",
 "Vorgang\u2022abgebrochen,\u2022das\u2022Geraet
 \u2022muss\u2022neu\u2022gestartet\u2022werden",
 ""};

```

```
void setup() {
    stepper.setMaxSpeed(1000);
    stepper.setAcceleration(1000);
    pinMode(PIN_LED_RED, OUTPUT);
    pinMode(PIN_LED_GREEN, OUTPUT);
    pinMode(PIN_LED_BLUE, OUTPUT);
    pinMode(PIN_BUTTON, INPUT_PULLUP);
    pinMode(PIN_ENDSTOP, INPUT_PULLUP);

    Serial.begin(9600);
    if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
        Serial.println(F("SSD1306 allocation failed"));
        for (;;) {
    }

    attachInterrupt(digitalPinToInterrupt(PIN_SW), handleInterrupt,
                    FALLING);

    display.display();
    delay(2000);
    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    updateDisplay(messages[0], messages[4]);
    delay(1000);
    updateLEDs(false, true, false);
}
```

Listing 6.4: void setup

```

void loop() {
    long newPosition = encoder.read();
    if (newPosition != oldPosition) {
        stage = constrain(newPosition / 4, 1, 10);
        updateDisplay(messages[1], stageMessages[stage]);
        oldPosition = newPosition;
    }

    int buttonReading = digitalRead(PIN_BUTTON);
    if (buttonReading != lastButtonStatus) {
        lastDebounceTime = millis();
    }

    if ((millis() - lastDebounceTime) > debounceDelay) {
        if (buttonReading != buttonStatus) {
            buttonStatus = buttonReading;
            if (buttonStatus == LOW) {
                stepper.setAcceleration(accelerations);
                while (!findReference()) {
                    stepper.setSpeed(-200);
                    stepper.runSpeed();
                }
            }

            int currentStage = stage;
            updateDisplay(messages[2], stageMessages[stage]);
            updateLEDs(false, false, true);
            for (int i = 0; i < 2; i++) {
                if (i == 0) {
                    stepper.setMaxSpeed(slowSpeed);
                    stepper.setAcceleration(slowAcceleration);
                    stepper.move(trackLength);
                    while (stepper.distanceToGo() > trackLength) {
                        stepper.run();
                    }
                } else {
                    stepper.setMaxSpeed(fastSpeed);
                    stepper.setAcceleration(fastAcceleration);
                    stepper.move(trackLength);
                    while (stepper.distanceToGo() > trackLength) {
                        stepper.run();
                    }
                }
            }
            stepper.move(-trackLength);
            while (stepper.distanceToGo() > trackLength) {
                stepper.run();
            }
        }
        updateDisplay(messages[1], stageMessages[stage]);
    }
}
lastButtonStatus = buttonReading;

```

```
void updateLEDs(bool red, bool green, bool blue) {
    digitalWrite(PIN_LED_RED, red ? HIGH : LOW);
    digitalWrite(PIN_LED_GREEN, green ? HIGH : LOW);
    digitalWrite(PIN_LED_BLUE, blue ? HIGH : LOW);
}
```

Listing 6.6: void updateLEDs

```
void updateDisplay(const char* line1, const char* line2) {
    display.clearDisplay();
    display.setCursor(0, 0);
    display.println(F(line1));
    if (line2 != "") {
        display.println(F(line2));
    }
    display.display();
}
```

Listing 6.7: void updateDisplay

```
bool findReference() {
    int endstopReading = digitalRead(PIN_ENDSTOP);
    if (endstopReading != lastEndstopStatus) {
        lastEndstopDebounceTime = millis();
    }

    if ((millis() - lastEndstopDebounceTime) > debounceDelay) {
        if (endstopReading != endstopStatus) {
            endstopStatus = endstopReading;
            if (endstopStatus == LOW) {
                stepper.stop();
                return true;
            }
        }
    }
    lastEndstopStatus = endstopReading;
    return false;
}
```

Listing 6.8: bool findReference

```
void handleInterrupt() {
    stepper.stop();
    updateDisplay(messages[3], messages[4]);
}
```

Listing 6.9: void handleInterrupt

6.3 Bibliotheken

Für die Erstellung des Programms werden Bibliotheken verwendet, um die Programmierung zu vereinfachen.

6.3.1 Accelstepper

Die Arduino AccelStepper Bibliothek bietet eine objektorientierte Schnittstelle für die Steuerung von Schrittmotoren und Motortreibern über 2, 3 oder 4 Pins. Diese Bibliothek stellt eine wesentliche Verbesserung gegenüber der standardmäßigen Arduino Stepper Bibliothek dar und bietet zahlreiche erweiterte Funktionen, die für anspruchsvollere Anwendungen erforderlich sind. Sie wird in diesem Projekt in der Version 1.64 verwendet. Zu den herausragenden Merkmalen der AccelStepper Bibliothek gehört die Unterstützung von Beschleunigungs- und Verzögerungsprozessen. Dies ermöglicht eine feinere Steuerung der Motorbewegungen, was insbesondere bei präzisen Anwendungen von Vorteil ist. Darüber hinaus unterstützt die Bibliothek die gleichzeitige Steuerung mehrerer Schrittmotoren, wobei jeder Motor unabhängig voneinander gesteuert werden kann. Dies ist besonders nützlich in Anwendungen, die eine synchronisierte Bewegung mehrerer Motoren erfordern. Ein weiterer Vorteil der AccelStepper Bibliothek ist, dass die meisten API-Funktionen nicht blockieren oder verzögern, es sei denn, dies ist ausdrücklich angegeben. Dies trägt dazu bei, dass der Hauptprogrammfluss nicht unterbrochen wird, was die Effizienz und Reaktionsfähigkeit des Systems verbessert. Die Bibliothek unterstützt sowohl 2-, 3- als auch 4-Draht-Schrittmotoren sowie 3- und 4-Draht-Halbschrittmotoren. Darüber hinaus können alternative Schrittverfahren verwendet werden, um beispielsweise den AFMotor zu unterstützen. Die Theorie hinter der AccelStepper Bibliothek basiert auf den Geschwindigkeitsberechnungen, die in dem Artikel „Generate stepper-motor speed profiles in real time“ von David Austin beschrieben sind. Die Bibliothek verwendet Schritte pro Sekunde, anstelle von Radianten pro Sekunde, da der Schrittwinkel des Motors nicht bekannt ist. Ein anfängliches Schrittintervall wird basierend auf der gewünschten Beschleunigung berechnet, und bei nachfolgenden Schritten werden kürzere Schrittintervalle basierend auf dem vorherigen Schritt berechnet, bis die maximale Geschwindigkeit er-

reicht ist. Die AccelStepper Bibliothek wird unter einer freien GPL-Lizenz angeboten, was bedeutet, dass sie kostenlos genutzt und verändert werden kann, solange der Quellcode offengelegt wird. Für kommerzielle Anwendungen, bei denen der Quellcode nicht offengelegt werden soll, steht eine kommerzielle Lizenz zur Verfügung. Zusammenfassend bietet die Arduino AccelStepper Bibliothek eine leistungsstarke und flexible Lösung für die Steuerung von Schrittmotoren. Mit ihrer Unterstützung für Beschleunigung und Verzögerung, der gleichzeitigen Steuerung mehrerer Motoren und ihrer nicht blockierenden API ist sie eine ausgezeichnete Wahl für anspruchsvolle Anwendungen.[Mik24]

6.3.2 Wire

Die Wire Bibliothek ermöglicht die Kommunikation mit I2C-Geräten auf allen Arduino-Plattformen und ist ein weit verbreitetes Protokoll zum Lesen und Senden von Daten zu und von externen I2C-Komponenten. Die I2C-Pins variieren je nach Hardware-Design und Architektur der verschiedenen Arduino-Boards, wobei die Standard-I2C-Pins für Nano Boards die Pins A4 (SDA) und A5 (SDL) sind.

6.3.3 Adafruit GFX

Die Adafruit GFX Bibliothek ist eine Kern-Grafikbibliothek, die grundlegende Grafikprimitive wie Punkte, Linien und Kreise für Adafruit-Displays bereitstellt. Sie ist unerlässlich für die Darstellung von Grafiken auf verschiedenen Hardware-Displays und erfordert für jedes Display eine spezifische Hardware-Bibliothek zur Verwaltung der niedrigeren Funktionen. Zusätzlich unterstützt sie verschiedene Bitmap-Schriftarten und bietet Werkzeuge wie Image2Code zur Konvertierung von BMP-Dateien in Code-Arrays für die Anzeige. Die Bibliothek legt besonderen Wert auf die Rückwärtskompatibilität mit bestehenden Arduino-Sketches und bietet umfassende Möglichkeiten zur Optimierung und Anpassung von Grafik- und Textdarstellungen auf den Displays. Sie wird in diesem Projekt in der Version 1.11.9 verwendet.[Ada24a]

6.3.4 Adafruit SSD1306

Die Adafruit SSD1306 Bibliothek ist speziell für monochrome OLED-Displays entwickelt, die den SSD1306 Treiber verwenden. Diese Displays kommunizieren über I2C oder SPI und erfordern 2 bis 5 Pins für die Verbindung. Die Bibliothek bietet grundlegende Funktionen zur Steuerung der Displays wie das Initialisieren der Kommunikation, das Zeichnen von Grafikelementen wie Linien und Kreisen sowie das Anzeigen von Text. Sie ermöglicht die einfache Anpassung der Displaygröße über den Konstruktor

und unterstützt sowohl SPI-Transaktionen als auch die Spezifikation der SPI-Bitrate ab Arduino 1.6 oder höher. Die Installation erfolgt idealerweise über den Arduino IDE Bibliotheksmanager oder durch manuelles Herunterladen und Entpacken des Quellcodes von GitHub. Sie wird in diesem Projekt in der Version 2.5.10 verwendet.[Ada24b]

6.3.5 Encoder

Die Encoder Library ermöglicht das Zählen von Impulsen aus bestimmten Signalen, die häufig von Drehknöpfen, Motor- oder Wellensensoren sowie anderen Positionsgebern stammen. Diese Bibliothek ist für Arduino und Teensy Boards optimiert und bietet verschiedene Zählmodi, darunter 4X counting für präzise Positionserfassung. Die Verwendung erfolgt durch die Initialisierung eines Encoder-Objekts mit zwei Pins, wobei der erste Pin idealerweise über Interruptfähigkeit verfügt für optimale Leistung. Die Bibliothek unterstützt sowohl I2C- als auch SPI-Schnittstellen und bietet verschiedene Hardware- und Softwareoptionen zur Anpassung an die Anforderungen des Benutzers. Sie wird in diesem Projekt in der Version 1.4.4 verwendet.[Pau24]

7 Konstruktion

In diesem Kapitel wird die Konstruktion des Demonstrator-Schrittmotors beschrieben. Zunächst werden die Rahmenbedingungen erläutert, anschließend werden das Gehäuse und die Anbauteile des Aluprofils besprochen. Abschließend wird die gesamte Baugruppe betrachtet.

7.1 Rahmenbedingungen

Das Gehäuse sowie sämtliche Anbauteile sollen mittels additiver Fertigung aus Polylactic Acid (PLA) hergestellt werden. In und am Gehäuse werden sämtliche Komponenten zur Bedienung und Steuerung des Demonstrators untergebracht. Die Konstruktion soll eine komfortable Handhabung ermöglichen und sicherstellen, dass die Komponenten, wie beispielsweise das Tiny Machine Learning Shield mit dem aufgesteckten Arduino, fest sitzen.

7.2 Arbeiten mit SolidWorks 2023

Zur Herstellung des Gehäuses muss ein 3D-Modell mittels Computer Aided Design (CAD) erstellt werden. Hierfür wurde die Software SolidWorks 2023 vom französischen Software-Entwicklungsunternehmen Dassault Systèmes (DS) verwendet. „*SOLIDWORKS ist eine professionelle und leistungsstarke CAD-Software, die vor allem im Maschinenbau viel genutzt wird.*“ [Web24, S.225] Für die Additive Fertigung muss das CAD-Modell als STL-Datei gespeichert werden.

7.3 Gehäusekonstruktion

Zu Beginn wurden die Abmessungen der einzelnen Komponenten ermittelt, um die Dimensionen des Gehäuses festzulegen. Das Gehäuse, ersichtlich in Abbildung 7.1, besteht aus sechs Einzelteilen, die mit Innensechskantschrauben DIN 912 M3 verbunden werden.

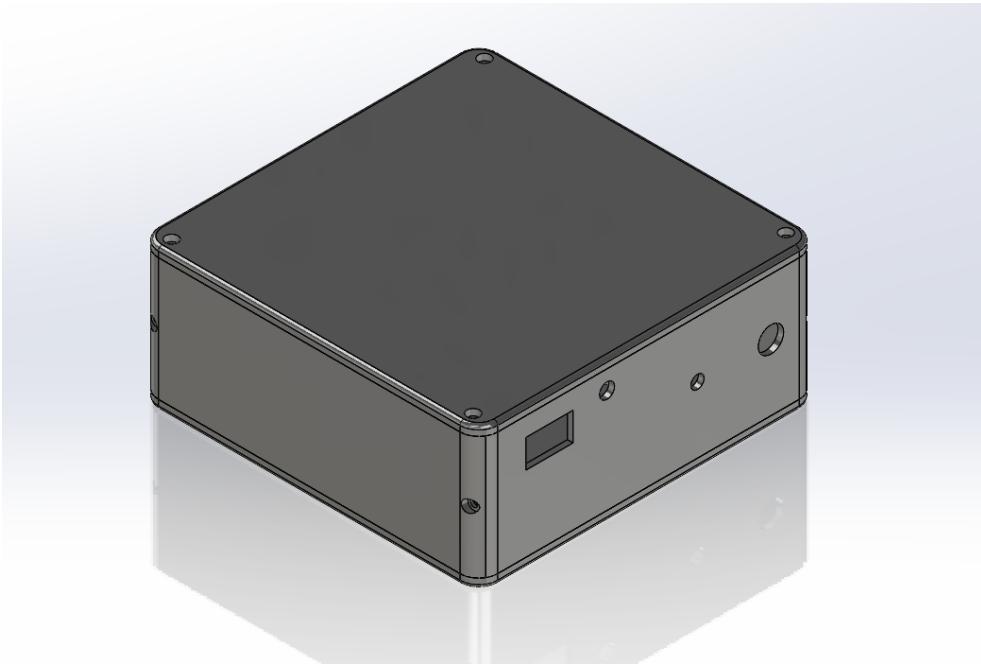


Abbildung 7.1: Baugruppe Gehäuse (Eigenaufnahme)

Die Basis der Baugruppe ist die Bodenplatte mit einer Länge von 180 mm, einer Breite von 180 mm und einer Dicke von 7 mm, wie in Abbildung 7.2 dargestellt. An der Bodenplatte werden die anderen Teile des Gehäuses, das Schaltnetzteil, das Tiny Machine Learning Shield, das Aluminiumprofil, die Schrittmotorsteuerung und die Halterung für den Spannungswandler befestigt. Die Ecken sind mit einem Radius von 10 mm abgerundet. Die Bohrungen 1 bis 6 sind in Tabelle 7.1 aufgeführt.

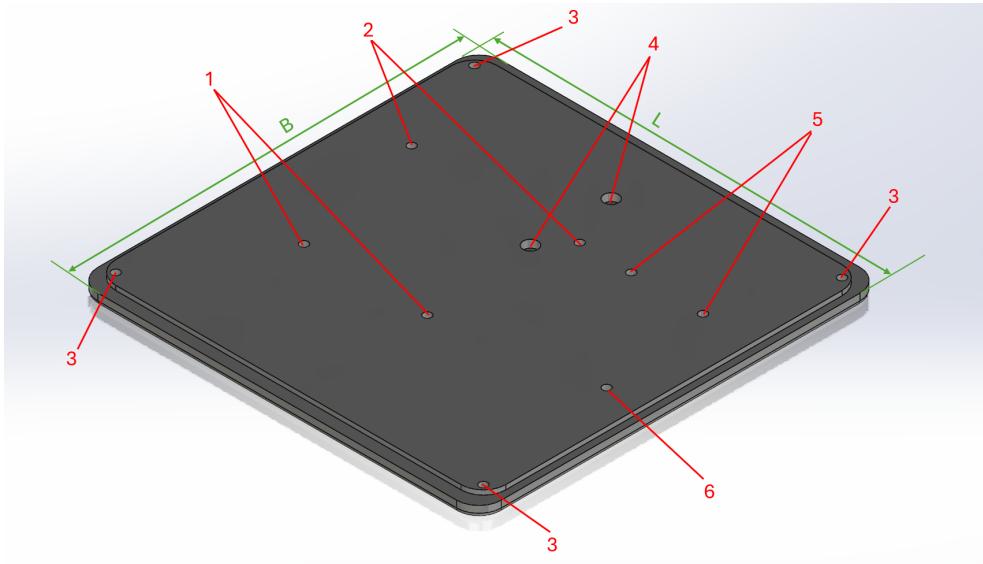


Abbildung 7.2: Bodenplatte (Eigenaufnahme)

An der Bodenplatte wird die Vorderplatte befestigt. Sie hat eine Breite von 170 mm und eine Höhe von 65 mm, wie in Abbildung 7.3 dargestellt. Das OLED-Display wird von innen eingesetzt und mit DIN 912 M3×6 Schrauben verschraubt (siehe Markierung a $25 \times 15 \text{ mm}$ in Abb.7.3). Die LED wird von innen eingesetzt und mittels Presspassung fixiert (siehe Markierung b $\varnothing 8 \text{ mm}$ in Abb.7.3). Der Drehwinkel-Encoder wird von innen durchgeführt und mittels einer P4-Mutter befestigt (siehe Markierung c $\varnothing 7 \text{ mm}$ in Abb.7.3). Der Drucktaster wird von außen durchgeführt und mittels Kontermutter befestigt (s. Markierung a $\varnothing 13,5 \text{ mm}$ in Abb.7.3). Die Bohrungen 7 bis 9 sind der Tabelle 7.1 aufgeführt.

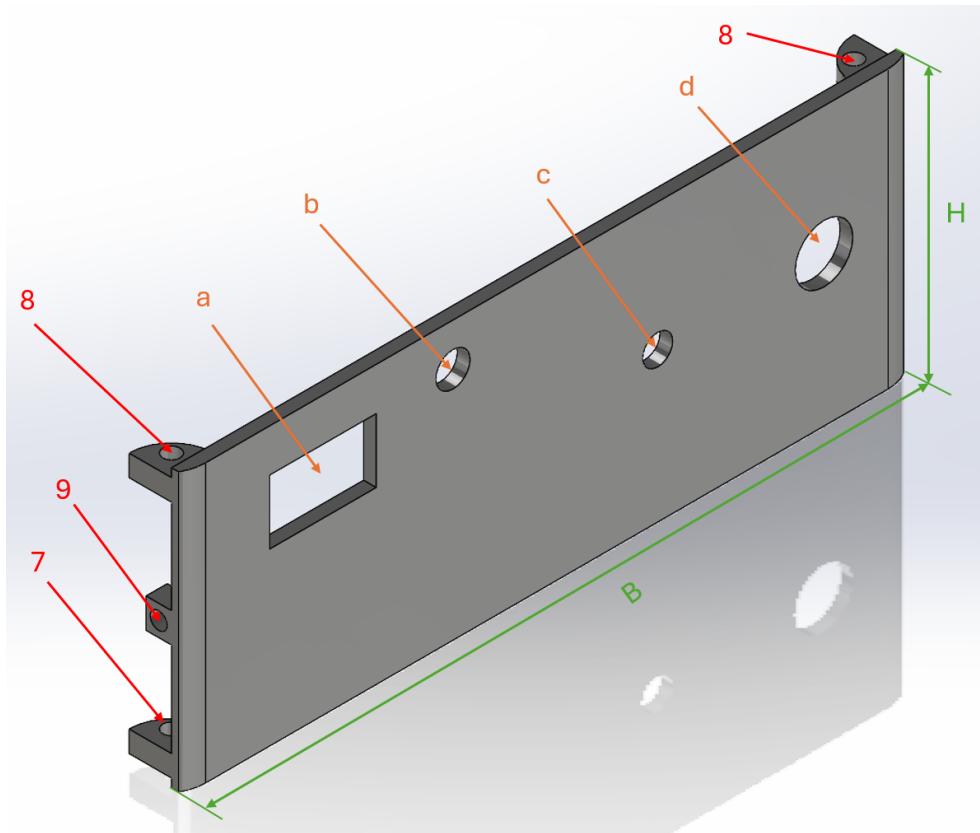


Abbildung 7.3: Vorderplatte (Eigenaufnahme)

Die Seitenplatte Links wird an der Vorder- und Hinterplatte befestigt. Die Seitenplatte Links hat eine Länge von 177,32 mm, eine Höhe von 65 mm und eine Dicke von 5 mm, erkennbar in Abbildung 7.4. Die Ecken sind mit einem Radius von 10 mm abgerundet. Die Bohrungen 10 und 11 sind der Tabelle 7.1 aufgeführt.

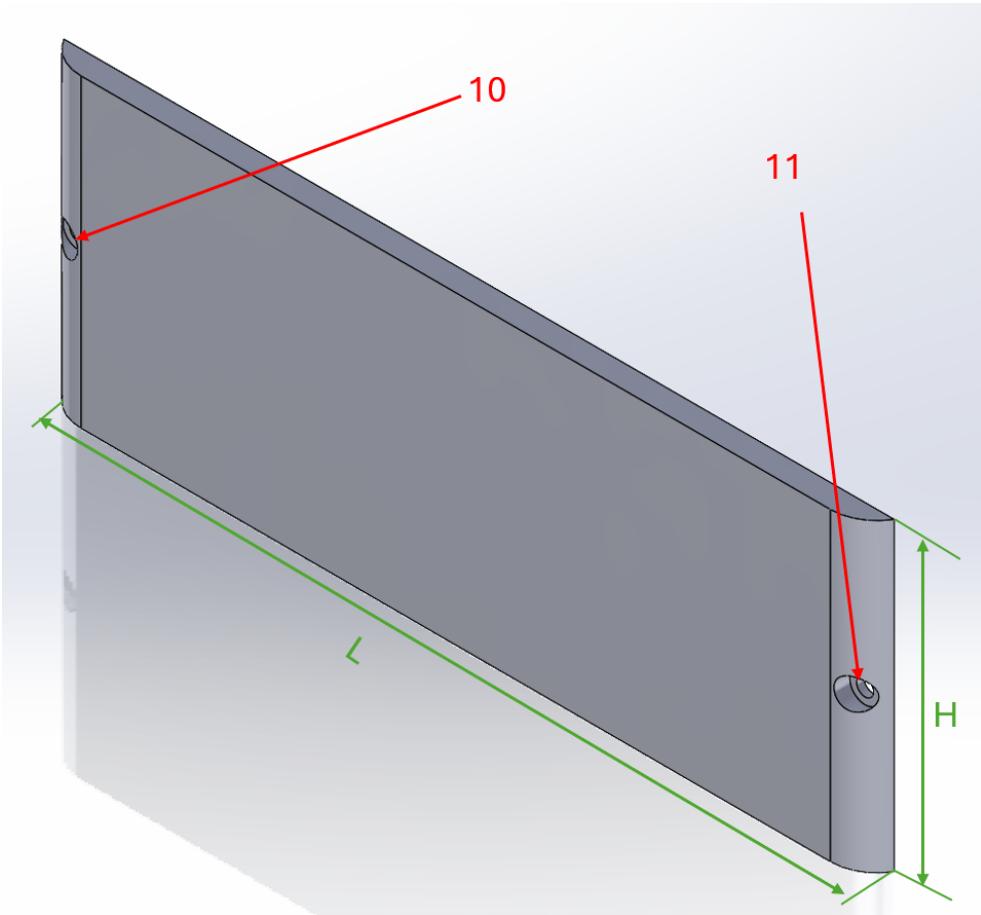


Abbildung 7.4: Seitenplatte Links (Eigenaufnahme)

Die Hinterplatte wird an der Bodenplatte gefügt. Die Hinterplatte hat eine Breite B von 170 mm und eine Höhe H von 65 mm, erkennbar in Abbildung 7.5. Der Kaltgerätestecker mit Artillerie-Wippschalter wird an der Außenseite der Rückplatte montiert (siehe Markierung e 30,5 × 47 mm in Abb.7.5). Die Bohrungen 12 bis 15 sind der Tabelle 7.1 aufgeführt.

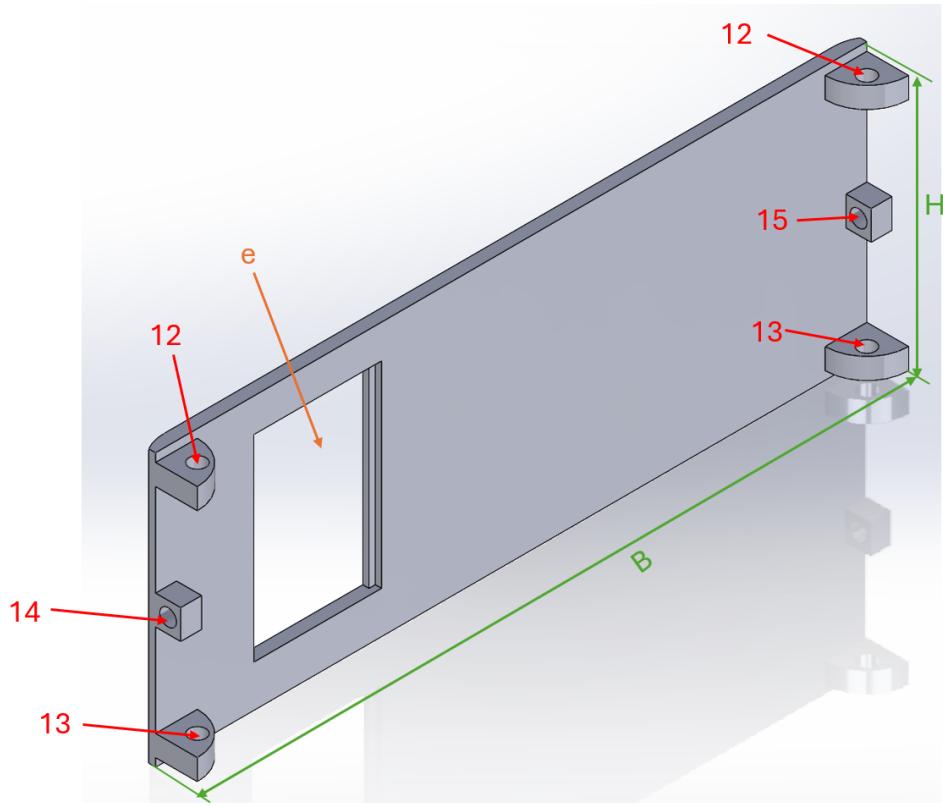


Abbildung 7.5: Hinterplatte (Eigenaufnahme)

Die Seitenplatte Rechts wird an der Vorder- und Hinterplatte befestigt. Die Seitenplatte Rechts hat eine Länge von 177,32 mm, eine Höhe von 65 mm und eine Dicke von 5 mm, erkennbar in Abbildung 7.4. Die Ecken sind mit einem Radius von 10 mm abgerundet. Für die Durchführung der Leitungen vom Schrittmotor und des Microschalters wurde eine Aussparung konstruiert (siehe Markierung f Ø 13 mm in Abb.7.6). Die Bohrungen 16 und 17 sind der Tabelle 7.1 aufgeführt.

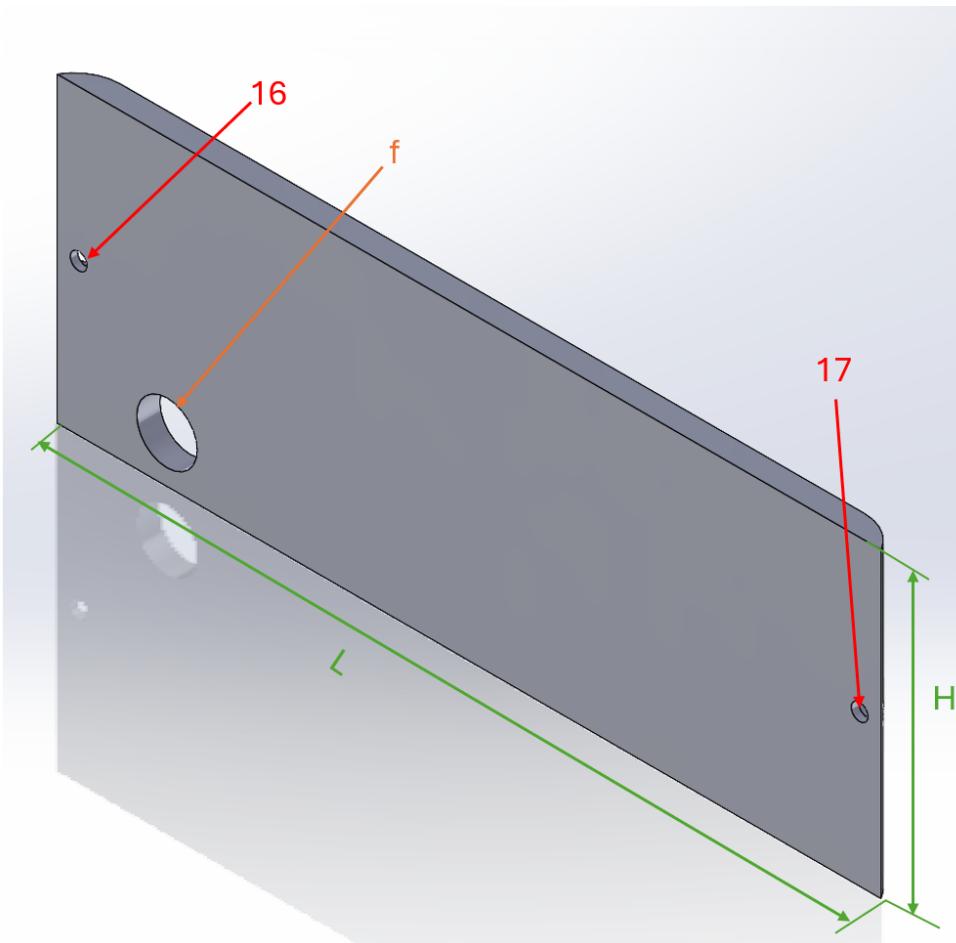


Abbildung 7.6: Seitenplatte Rechts (Eigenaufnahme)

Die Deckelplatte wird an der Vorder- und Hinterplatte befestigt. Die Deckelplatte hat ein Länge L von 180 mm und einer Breite B von 180 mm und einer Dicke von 7 mm, erkennbar in Abbildung 7.7. Die Ecken sind mit einem Radius von 10 mm abgerundet. Die Bohrungen 18 sind der Tabelle 7.1 aufgeführt.

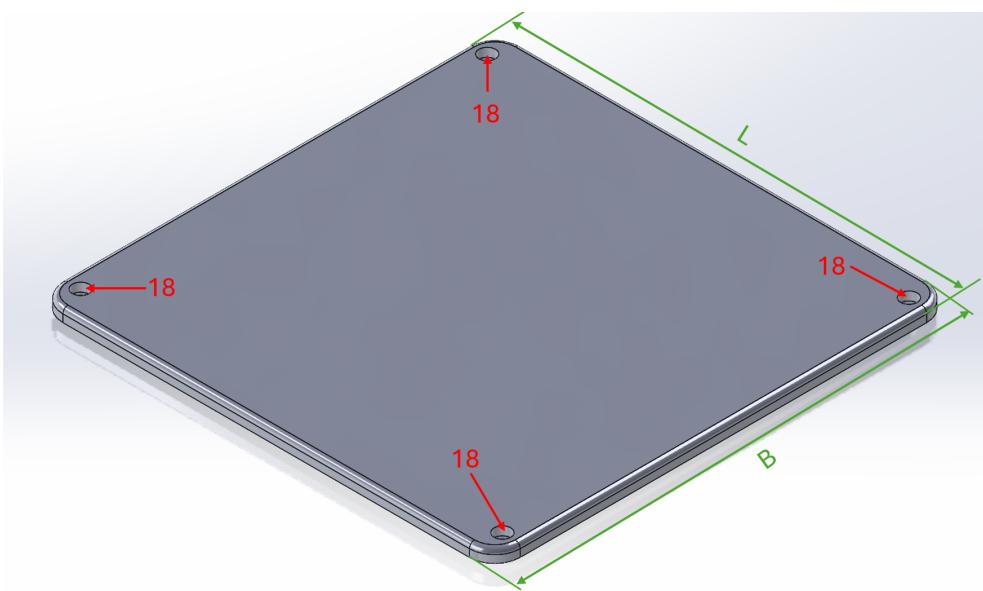


Abbildung 7.7: Deckelplatte (Eigenaufnahme)

Nr.	\varnothing	Beschreibung
1	DIN912 M3	Befestigung Schaltnetzteil an Bodenplatte
2	DIN912 M3	Befestigung Tiny Maschine Learning Schield an Bodenplatte
3	DIN912 M3	Fügen der Gehäuseteile an Bodenplatte
4	DIN912 M3	Befestigung Bodenplatte an Aluprofil
5	DIN912 M3	Befestigung Schrittmotorsteuerung an Bodenplatte
6	DIN912 M3	Befestigung des Halters für Spannungswandler an Bodenplatte
7	4 mm	Bohrung für M3-Einpressmutter zur Befestigung Vorderplatte an Bodenplatte
8	4 mm	Bohrung für M3-Einpressmutter zur Befestigung Vorderplatte an Deckelplatte
9	4 mm	Bohrung für M3-Einpressmutter zur Befestigung Vorderplatte an Seitenplatte Links
10	DIN912 M3	Befestigung der Seitenplatte Links an Hinterplatte
11	DIN912 M3	Befestigung der Seitenplatte Links an Vorderplatte
12	4 mm	Bohrung für M3-Einpressmutter zur Befestigung Hinterplatte an Deckelplatte
13	4 mm	Bohrung für M3-Einpressmutter zur Befestigung Hinterplatte an Bodenplatte
14	4 mm	Bohrung für M3-Einpressmutter zur Befestigung Hinterplatte an Seitenplatte Links
15	4 mm	Bohrung für M3-Einpressmutter zur Befestigung Hinterplatte an Seitenplatte Rechts
16	DIN912 M3	Befestigung der Seitenplatte Rechts an Hinterplatte
17	DIN912 M3	Befestigung der Seitenplatte Rechts an Vorderplatte
18	DIN912 M3	Fügen der Gehäuseteile an Deckelplatte

Tabelle 7.1: Bohrungen im Gehäuse

7.4 Anbauteile

Nachfolgend werden die Anbauteile in der Konstruktion beschrieben. In der Abbildung 7.8 zu erkennen, ist die Halterung für den Motor. Die Halterung hat ein Breite B von 50 mm und eine Höhe H von 70 mm und eine Dicke von 5 mm, erkennbar in Abbildung 7.8. Sie enthält eine Aussparung für den Motor (siehe Markierung g Motor \varnothing 22,5 mm, Welle \varnothing 6 mm in Abb.7.6). Die Halterung wird an dem Aluprofil montiert. Die Bohrungen 20 und 21 sind der Tabelle 7.2 aufgeführt.

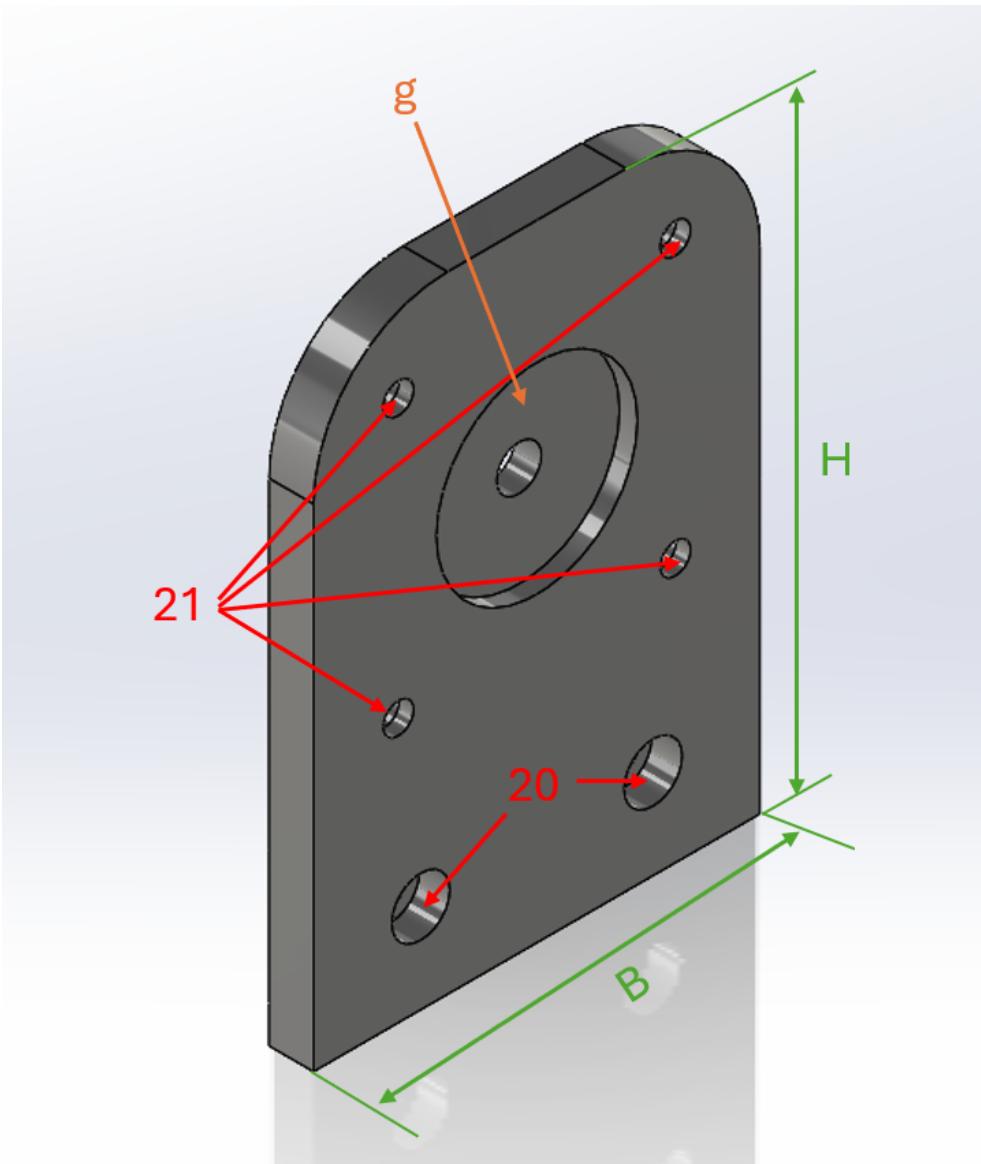


Abbildung 7.8: Halterung für den Motor (Eigenaufnahme)

Die Halterung für die Welle hat eine Breite B von 50 mm und eine Höhe H von 70 mm und eine Dicke von 5 mm , erkennbar in Abbildung 7.9. Die Halterung besitzt eine Aussparung für die Welle (siehe Markierung h Ø 5 mm in Abb.7.9). Die Halterung wird an dem Aluprofil montiert. Die Bohrungen 22 sind der Tabelle 7.2 aufgeführt.

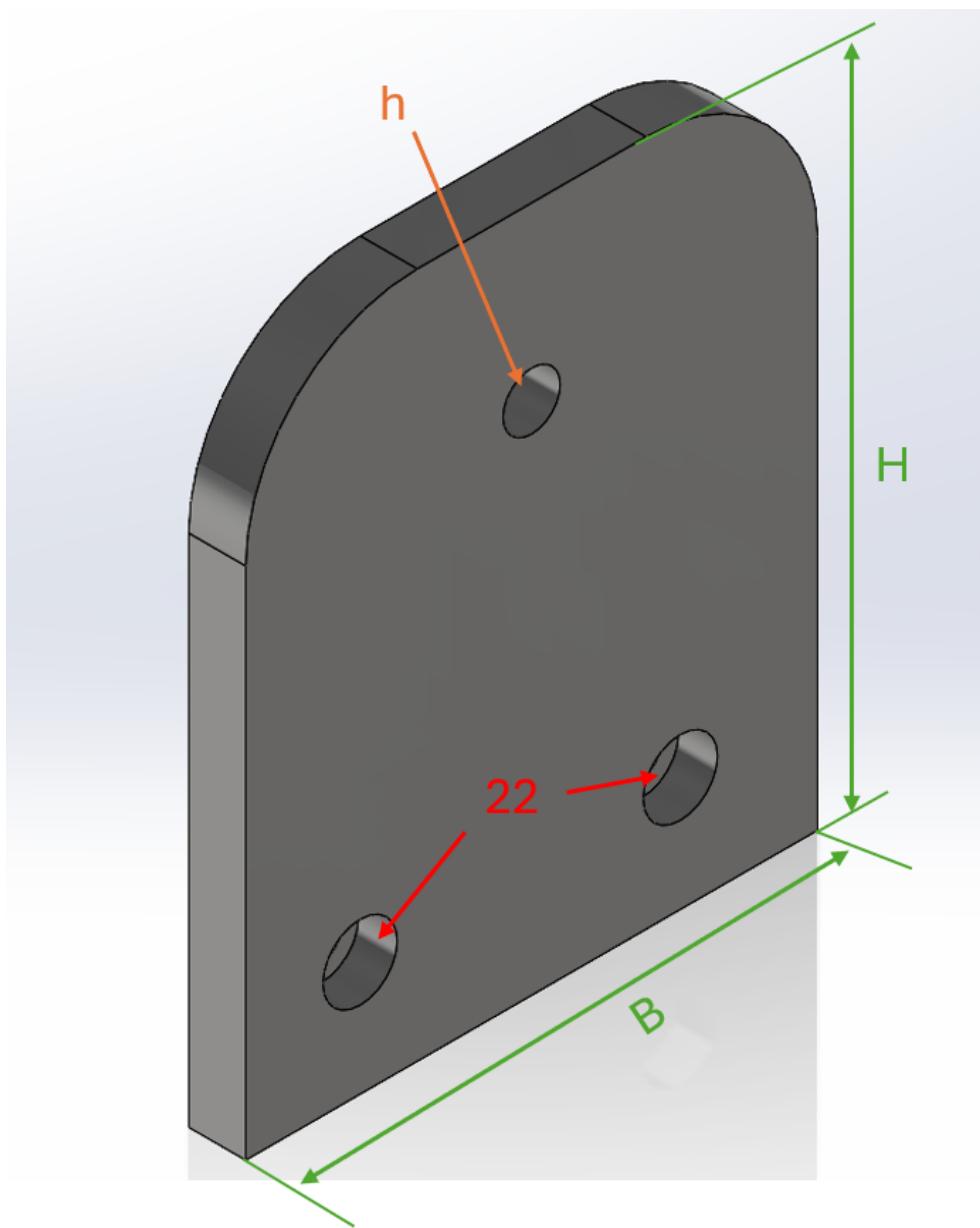


Abbildung 7.9: Halterung für die Welle (Eigenaufnahme)

Der Griff zum Transportieren hat eine Breite B von 170 mm und eine Höhe H von 60 mm bei einer Dicke von 20 mm, die innere Spannweite Li beträgt 100 mm mit einer Höhe Hi von 50 mm , erkennbar in Abbildung 7.10. Die Bohrungen 23 sind der Tabelle 7.2 aufgeführt.

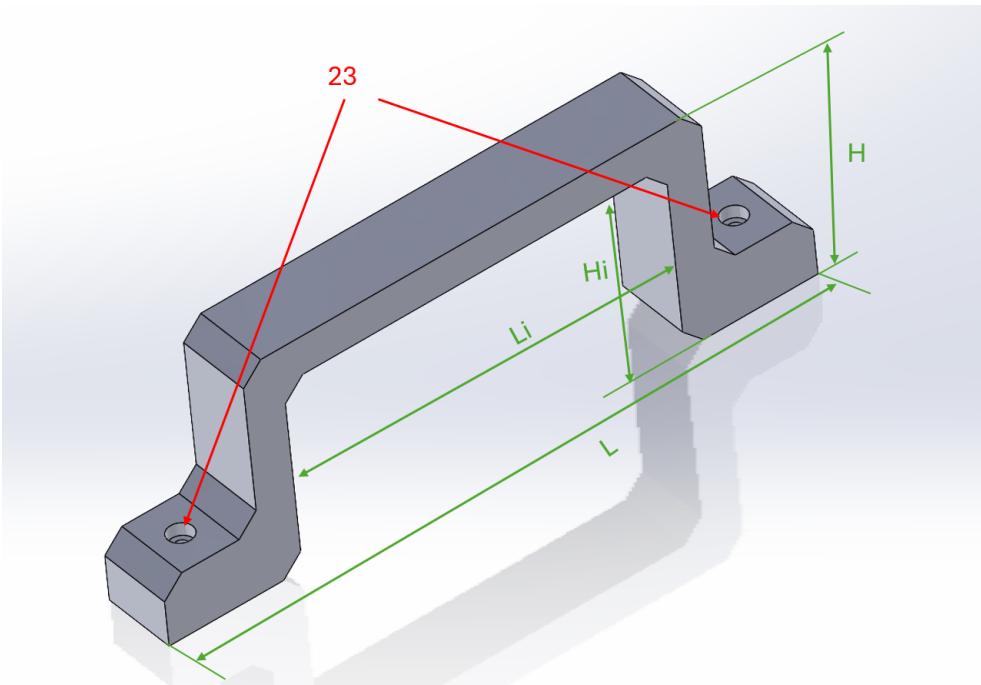


Abbildung 7.10: Transportgriff (Eigenaufnahme)

Der Drehknopf wird mittels einer DIN912 M3 Schraube auf dem Drehwinkel-Encoder befestigt. Der Drehknopf hat einen Durchmesser D von 32 mm und einer Länge von 20 mm. Der Drehknopf zeigt eine zahnförmige Kontur, erkennbar in Abbildung 7.11. Die Bohrungen 24 sind der Tabelle 7.2 aufgeführt.

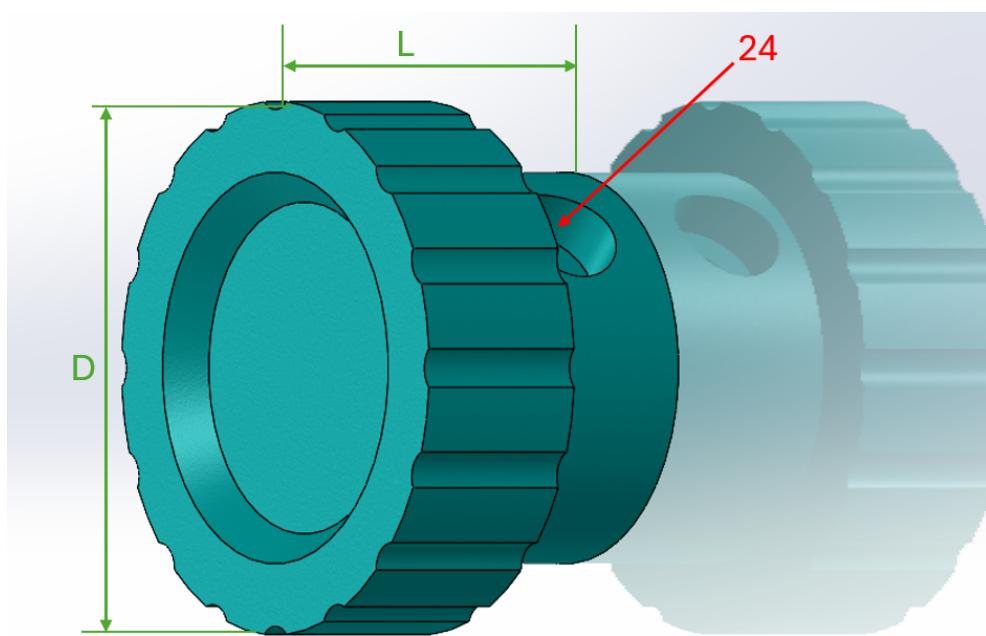


Abbildung 7.11: Drehknopf (Eigenaufnahme)

Der Anzeiger hat eine Breite B von 40 mm und eine Länge L von 50 mm, erkennbar in Abbildung 7.12. Der Pfeil (siehe Markierung i in Abb.7.12) auf dem Anzeiger hat eine Tiefe von 35 mm. Die Bohrungen 25 und 26 sind der Tabelle 7.2 aufgeführt.

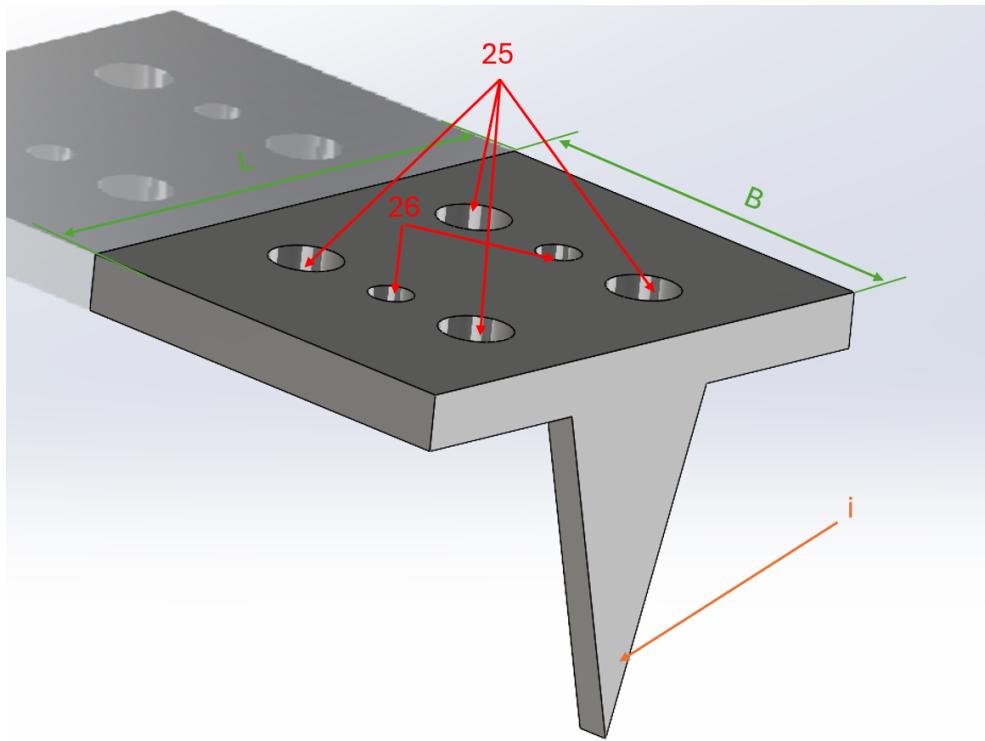


Abbildung 7.12: Anzeiger (Eigenaufnahme)

Nr.	\varnothing	Beschreibung
20	DIN912 M3	Befestigung Motor an Flanke Halterung für Motor
21	DIN912 M3	Befestigung Flanke Halterung für Motor an Aluprofil
22	DIN912 M3	Befestigung Flanke an Aluprofil
23	DIN912 M3	Befestigung Griff an Aluprofil
24	DIN912 M3	Befestigung Drehknopf mit Drehwinkel-Encoder
25	DIN912 M3	Befestigung Anzeiger an Schlitten
26	2,8 mm	Bohrung für M3-Einpressmutter zur Befestigung Riemen an Anzeiger

Tabelle 7.2: Bohrungen der Anbauteile

7.5 3D-Druck mit PLA

Die Einzelteile des Gehäuses sowie die Anbauteile wurden mit dem Anycubic Kobra 2 Neo gefertigt, wahrnehmbar in 7.13. Als Fertigungsmaterial wurde PLA gewählt. Das additve Verfahren eignet sich als kostengünstiges und schnelles Verfahren für den Prototypen-Bau. Für den Druck aller

Einzelteile wurde c.a 0,5 kg Filament verbraucht. Der Preis für einen kg liegt bei ungefähr 20 €. Der längste Druck dauerte ca. 2,5 h.

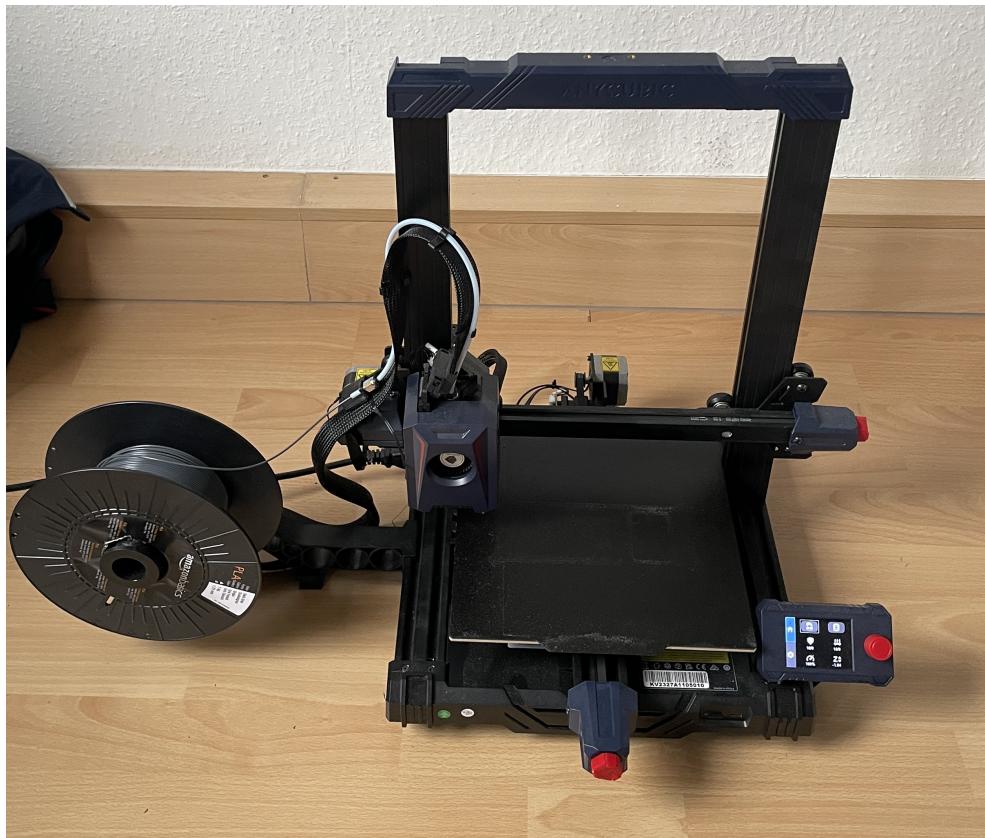


Abbildung 7.13: AnyKubic 2 Kobra Neo (Eigenaufnahme)

8 Testdurchläufe

Ein fehlerfreier Ablauf kann nie vollständig gewährleistet werden, es können stets zufällige Fehler auftreten. Um die Funktionalität und Zuverlässigkeit des Systems zu gewährleisten, sind regelmäßige Testdurchläufe notwendig. Eine erste Überprüfung findet mit der Sichtprüfung der Hardware auf Schäden statt. Für die softwareseitige Überprüfung können Testprogramme nach Einschalten durchlaufen werden.

8.1 Arduino Nano 33 BLE Sense Lite

Für die Überprüfung des Arduinos, werden Beispielsketches von der Arduino IDE zur Verfügung gestellt. Unter dem Pfad File/Examples/01.Basics können Beispielsketches ausgesucht werden. Mit dem Beispielprogramm Blink wird die Built In LED angesprochen. Bei richtiger Funktion des Arduinos sollte die LED für eine Sekunde eingeschaltet und für eine Sekunde ausgeschaltet werden. Dieses wiederholt sich fortlaufend. Dadurch ist die Ansteuerung des Arduinos sichergestellt. Der Code dieser Anwendung wird detaillierter im Kapitel 5.5 erläutert.

8.2 Schaltnetzteil

Die Überprüfung des Schaltnetzteils **muss** von einer Elektrofachkraft durchgeführt werden. Ist das Schaltnetzteil ordnungsgemäß angeschlossen, leuchtet eine Signalleuchte Grün auf dem Schaltnetzteil. Um eine einwandfreie Nutzung zu gewährleisten müssen noch die Ausgangsspannungen des Netzteils mit einem Voltmeter überprüft werden. Liegen die Ausgangsspannungen jeweils bei 12 V und 5 V ist eine einwandfreie Nutzung gewährleistet.

8.3 Spannungsregler AMS1117

Ist der Spannungswandler ordnungsgemäß angeschlossen, leuchtet eine Signalleuchte Rot auf dem AMS1117. Um eine einwandfreie Nutzung zu gewährleisten muss noch die Ausgangsspannung des Spannungswandlers mit einem Voltmeter überprüft werden. Liegt die Ausgangsspannung bei 3,3 V ist eine einwandfreie Nutzung gewährleistet.

8.4 Nema 17 Schrittmotor mit Schrittmotorsteuerung A4988

Die korrekte Verbindung der Pins sowie der Schrittmotorsteuerung wird durch das folgende Testprogramm 8.1 getestet. Hierbei wird der Schrittmotor in einer fest gelegten Geschwindigkeit rotiert. [Mik24]:

```
#include "AccelStepper.h"

#define dirPin 0
#define stepPin 8
#define motorInterfaceType 1

AccelStepper stepper = AccelStepper(motorInterfaceType, stepP

void setup() {
    stepper.setMaxSpeed(1000);
    stepper.setAcceleration(1000);
}

void loop() {
    stepper.setSpeed(500);
    stepper.runSpeed();
}
```

Listing 8.1: Testprogramm für den Schrittmotor

Zunächst hat die Ansteuerung des Motors nicht wie geplant funktioniert, statt zu rotieren, vibrierte der Motor lediglich. Nach der Überprüfung der Innenwiderstände am Motor und dem Abgleich mit dem Anschlussplan der Schrittmotorsteuerung wurde deutlich, dass die Zuordnung der Pins am Steckkabel nicht korrekt waren. Durch die Umpolung des Steckkabels konnte die korrekte Ansteuerung des Schrittmotors hergestellt und bestätigt werden.

8.5 OLED-Display

Um die Funktion und korrekte Ansteuerung des OLED-Displays zu testen, wird ein Testprogramm (Listing 8.2) verwendet, welches nacheinander verschiedene Abbildungen und Animationen auf dem Display darstellt und abspielt [Rui16]:

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, 

#define NUMFLAKES 10

#define LOGO_HEIGHT 16
#define LOGO_WIDTH 16
static const unsigned char PROGMEM logo_bmp [] =
{ 0b00000000, 0b11000000,
  0b00000001, 0b11000000,
  0b00000001, 0b11000000,
  0b00000011, 0b11100000,
  0b11110011, 0b11100000,
  0b11111110, 0b11111000,
  0b01111110, 0b11111111,
  0b00110011, 0b10011111,
  0b00011111, 0b11111100,
  0b00001101, 0b01110000,
  0b00011011, 0b10100000,
  0b00111111, 0b11100000,
  0b00111111, 0b11110000,
  0b01111100, 0b11110000,
  0b01110000, 0b01110000,
  0b00000000, 0b00110000 };

void setup() {
  Serial.begin(9600);

  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS))
    Serial.println(F("SSD1306 allocation failed"));
  for (;;) {
}

  display.display();
  delay(2000);

  display.clearDisplay();

  display.drawPixel(10, 10, SSD1306_WHITE);
  display.display();
  delay(2000);
```

```
    testdrawline();
    testdrawrect();
    testfillrect();
    testdrawcircle();
    testfillcircle();
    testdrawroundrect();
    testfillroundrect();
    testdrawtriangle();
    testfilltriangle();
    testdrawchar();
    testdrawstyles();
    testscrolltext();
    testdrawbitmap();

    display.invertDisplay(true);
    delay(1000);
    display.invertDisplay(false);
    delay(1000);

    testanimate(logo_bmp, LOGO_WIDTH, LOGO_HEIGHT);
}

void loop() {
}

void testdrawline() {
    int16_t i;

    display.clearDisplay();

    for (i=0; i<display.width(); i+=4) {
        display.drawLine(0, 0, i, display.height()-1,
                        display.display());
        delay(1);
    }
    for (i=0; i<display.height(); i+=4) {
        display.drawLine(0, 0, display.width()-1, i,
                        display.display());
        delay(1);
    }
    delay(250);

    display.clearDisplay();

    for (i=0; i<display.width(); i+=4) {
        display.drawLine(0, display.height()-1, i, 0,
                        display.display());
        delay(1);
    }
```

```
    for ( i=display.height()-1; i>=0; i-=4) {
        display.drawLine(0, display.height()-1, display.display());
        delay(1);
    }
    delay(250);

    display.clearDisplay();

    for ( i=display.width()-1; i>=0; i-=4) {
        display.drawLine(display.width()-1, display.display());
        delay(1);
    }
    for ( i=display.height()-1; i>=0; i-=4) {
        display.drawLine(display.width()-1, display.display());
        delay(1);
    }
    delay(250);

    display.clearDisplay();

    for ( i=0; i<display.height(); i+=4) {
        display.drawLine(display.width()-1, 0, 0, i,
        display.display());
        delay(1);
    }
    for ( i=0; i<display.width(); i+=4) {
        display.drawLine(display.width()-1, 0, i, display.display());
        delay(1);
    }

    delay(2000);
}
```

```
void testdrawrect() {
    display.clearDisplay();

    for(int16_t i=0; i<display.height()/2; i+=2) {
        display.drawRect(i, i, display.width()-2*i, 2);
        display.display();
        delay(1);
    }

    delay(2000);
}

void testfillrect() {
    display.clearDisplay();

    for(int16_t i=0; i<display.height()/2; i+=3) {
        display.fillRect(i, i, display.width()-i*2, 3);
        display.display();
        delay(1);
    }

    delay(2000);
}

void testdrawcircle() {
    display.clearDisplay();

    for(int16_t i=0; i<max(display.width(), display.height()); i+=2) {
        display.drawCircle(display.width()/2, display.height()/2, i);
        display.display();
        delay(1);
    }

    delay(2000);
}

void testfillcircle() {
    display.clearDisplay();

    for(int16_t i=max(display.width(), display.height())/2; i>0; i-=2) {
        display.fillCircle(display.width() / 2, display.height() / 2, i);
        display.display();
        delay(1);
    }

    delay(2000);
}
```

```
void testdrawroundrect() {
    display.clearDisplay();

    for(int16_t i=0; i<display.height()/2-2; i+=2) {
        display.drawRoundRect(i, i, display.width()-2,
            display.height()/4, SSD1306_WHITE);
        display.display();
        delay(1);
    }

    delay(2000);
}

void testfillroundrect() {
    display.clearDisplay();

    for(int16_t i=0; i<display.height()/2-2; i+=2) {
        display.fillRoundRect(i, i, display.width()-2,
            display.height()/4, SSD1306_INVERSE);
        display.display();
        delay(1);
    }

    delay(2000);
}

void testdrawtriangle() {
    display.clearDisplay();

    for(int16_t i=0; i<max(display.width(), display.height());
        display.drawTriangle(
            display.width()/2
            , display.height()/2-i,
            display.width()/2-i, display.height()/2+i,
            display.width()/2+i, display.height()/2+i, SSD1306_WHITE);
        display.display();
        delay(1);
    }

    delay(2000);
}

void testfilltriangle() {
    display.clearDisplay();
```

```
    for (int16_t i=max(display.width(), display.height()) / 2  
        display.fillTriangle(  
        display.width() / 2  
, display.height() / 2 - i,  
        display.width() / 2 - i, display.height() / 2 + i,  
        display.width() / 2 + i, display.height() / 2 + i, SS  
        display.display();  
        delay(1);  
    }  
  
    delay(2000);  
}  
  
void testdrawchar() {  
    display.clearDisplay();  
  
    display.setTextSize(1);  
    display.setTextColor(SSD1306_WHITE);  
    display.setCursor(0, 0);  
    display.cp437(true);  
  
    for (int16_t i=0; i < 256; i++) {  
        if (i == '\n') display.write('↵');  
        else  
display.write(i);  
    }  
  
    display.display();  
    delay(2000);  
}  
  
void testdrawstyles() {  
    display.clearDisplay();  
  
    display.setTextSize(1);  
    display.setTextColor(SSD1306_WHITE);  
    display.setCursor(0, 0);  
    display.println(F("Hello, world!"));  
  
    display.setTextColor(SSD1306_BLACK, SSD1306_WHITE);  
    display.println(3.141592);  
  
    display.setTextSize(2);  
    display.setTextColor(SSD1306_WHITE);  
    display.print(F("0x")); display.println(0xDEADBEEF, H)  
  
    display.display();  
    delay(2000);  
}
```

```
void testscrolltext() {
    display.clearDisplay();

    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(10, 0);
    display.println(F(" scroll "));
    display.display();
    delay(100);

    display.startscrollright(0x00, 0x0F);
    delay(2000);
    display.stopscroll();
    delay(1000);
    display.startscrollleft(0x00, 0x0F);
    delay(2000);
    display.stopscroll();
    delay(1000);
    display.startscrolldiagright(0x00, 0x07);
    delay(2000);
    display.startscrolldiagleft(0x00, 0x07);
    delay(2000);
    display.stopscroll();
    delay(1000);
}

void testdrawbitmap() {
    display.clearDisplay();

    display.drawBitmap(
        (display.width()
        - LOGO_WIDTH) / 2,
        (display.height() - LOGO_HEIGHT) / 2,
        logo_bmp, LOGO_WIDTH, LOGO_HEIGHT, 1);
    display.display();
    delay(1000);
}

#define XPOS 0
#define YPOS 1
#define DELTAY 2

void testanimate(const uint8_t *bitmap, uint8_t w, uint8_t h)
    int8_t f, icons[NUMFLAKES][3];
```

```

for ( f=0; f< NUMFLAKES; f++) {
    icons [ f ] [XPOS]
= random(1 - LOGO_WIDTH, display . width ());
    icons [ f ] [YPOS]
= -LOGO_HEIGHT;
    icons [ f ] [DELTAY] = random(1, 6);
    Serial . print(F("x:\u00d7"));
    Serial . print(icons [ f ] [XPOS], DEC);
    Serial . print(F("\u00d7y:\u00d7"));
    Serial . print(icons [ f ] [YPOS], DEC);
    Serial . print(F("\u00d7dy:\u00d7"));
    Serial . println(icons [ f ] [DELTAY], DEC);
}

for (;;) {
    display . clearDisplay ();

    for ( f=0; f< NUMFLAKES; f++) {
        display . drawBitmap(icons [ f ] [XPOS], ic
    }

    display . display ();
    delay (200);

    for ( f=0; f< NUMFLAKES; f++) {
        icons [ f ] [YPOS] += icons [ f ] [DELTAY];
        if ( icons [ f ] [YPOS] >= display . height (
            icons [ f ] [XPOS]
= random(1 - LOGO_WIDTH, display . width ());
            icons [ f ] [YPOS]
= -LOGO_HEIGHT;
            icons [ f ] [DELTAY] = random(1,
        }
    }
}

```

Listing 8.2: Testprogramm für das OLED-Display

Mit Hilfe dieses Beispielprogrammes 8.2 konnte die Funktion des OLED-Displays einwandfrei bestätigt werden.

8.6 SMD-LED

Analog zum Blink-Beispiel-Sketch für den Test der Funktion des Mikrocontrollers (verweis Kap Beispielprogramm) wird für den Test der Funktion und der Ansteuerung der LED ein Testprogramm 8.3 genutzt, bei dem die LED in Abständen von wenigen Sekunden zwischen der Darstellung der verschiedenen Farben (Rot, Grün und Blau) wechselt.

```
int Led_Rot = 3;
int Led_Gruen = 2;
int Led_Blaue = 5;

void setup ()
{
    pinMode (Led_Rot, OUTPUT);
    pinMode (Led_Gruen, OUTPUT);
    pinMode (Led_Blaue, OUTPUT);
}

void loop ()
{
    digitalWrite (Led_Rot, HIGH);
    digitalWrite (Led_Gruen, LOW);
    digitalWrite (Led_Blaue, LOW);
    delay (1000);

    digitalWrite (Led_Rot, LOW);
    digitalWrite (Led_Gruen, HIGH);
    digitalWrite (Led_Blaue, LOW);
    delay (1000);

    digitalWrite (Led_Rot, LOW);
    digitalWrite (Led_Gruen, LOW);
    digitalWrite (Led_Blaue, HIGH);
    delay (1000);
}
```

Listing 8.3: Testprogramm für die LED

Dadurch konnte die Funktion der LED und die korrekte Ansteuerung der LED bestätigt werden.

8.7 Drucktaster und Mikroschalter

Die Funktion des Drucktasters und des Mikroschalters können auf die gleiche Art und Weise mithilfe des gleichen Testprogramms geprüft werden. Bei dem Testprogramm wird zunächst die Farbe der LED geändert, wenn der Taster oder Schalter betätigt ist, sobald der Taster oder Schalter nicht mehr betätigt wird, kehrt die LED in ihren Ursprungszustand zurück 8.4.

```

int Taster = 11;
int StatusTaster = 0;

void setup ()
{
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(Taster, INPUT);
    Serial.begin(9600);
}

void loop ()
{
    StatusTaster = digitalRead(Taster);

    if (StatusTaster == HIGH)
    {
        digitalWrite(LED_R, HIGH);
        digitalWrite(LED_G, LOW);
        digitalWrite(LED_B, LOW);
        delay(2000);

        digitalWrite(LED_R, LOW);
        digitalWrite(LED_G, LOW);
        digitalWrite(LED_B, HIGH);
    }
}

```

Listing 8.4: Testprogramm für die Taster

Sowohl der Drucktaster als auch der Mikroschalter konnten durch diesen Test ihre Funktion bestätigen. Die Umsetzung dieser Auswertung im Code für den Demonstrator erwies sich als problematisch, da der Zustand der beiden Schalter wurde nicht zuverlässig geprüft wurde. Dies konnte durch die Implementierung eines Debounce-Delays [SM.2022] im Code gelöst werden 8.5.

```
int Taster = 3;
int StatusTaster = HIGH;
int letzterStatusTaster = HIGH;
unsigned long letzteZeit = 0;
unsigned long debounceDelay = 50;

void setup()
{
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(Taster, INPUT_PULLUP);
    Serial.begin(9600);
}

void loop()
{
    int reading = digitalRead(Taster);

    if (reading != letzterStatusTaster) {
        letzteZeit = millis();
    }

    if ((millis() - letzteZeit) > debounceDelay) {
        if (reading != StatusTaster) {
            StatusTaster = reading;

            if (StatusTaster == LOW) {
                digitalWrite(LED_R, HIGH);
                digitalWrite(LED_G, LOW);
                digitalWrite(LED_B, LOW);
                delay(2000);

                digitalWrite(LED_R, LOW);
                digitalWrite(LED_G, LOW);
                digitalWrite(LED_B, HIGH);
            }
        }
    }
    letzterStatusTaster = reading;
}
```

Listing 8.5: Testprogramm für die Taster mit Debounce

8.8 Drehwinkel-Encoder

Um die Funktion und korrekte Ansteuerung des Drehwinkel-Encoders zu testen, wird das Testprogramm 8.6 verwendet. Das Programm wertet die Drehung des Encoders aus und gibt den neuen Zustand im seriellen Monitor der Arduino IDE aus [Fun24].

```
#include <Encoder.h>

const int CLK = 9;
const int DT = 2;
const int SW = 3;
long altePosition = -999;

Encoder meinEncoder(DT, CLK);

void setup()
{
    Serial.begin(9600);
    pinMode(SW, INPUT);
}

void loop()
{
    long neuePosition = meinEncoder.read();

    if (neuePosition != altePosition)
    {
        altePosition = neuePosition;
        Serial.println(neuePosition);
    }

    int StatusTaster = digitalRead(SW);
    if (StatusTaster == 0) {
        Serial.println("Switch betätigt");
        Serial.println("Switch betätigte");
    }
}
```

Listing 8.6: Testprogramm für den Encoder

8.9 Mittlere Geschwindigkeit des Schlittens

Zur Ermittlung der mittlere Geschwindigkeit des Schlittens, ist eine Division der zurückgelegten Strecke durch die dafür benötigte Zeit erforderlich. Für die Tests wurde der Demonstrator aufgebaut und die Stufen jeweils fünfmal gefahren. Die Zeit, die der Schlitten vom Wendepunkt bis zum Wendepunkt benötigt, wurde in der jeweiligen Stufe gestoppt. Die Strecke beträgt 0,35 m. Die Zeitmessung erfolgte mit einer handelsüblichen Stoppuhr. Die Ergebnisse sind in der Tabelle 8.1 aufgeführt.

Stufe	Ergebnis Versuch 1 [s]	Ergebnis Versuch 2 [s]	Ergebnis Versuch 3 [s]	Ergebnis Versuch 4 [s]	Ergebnis Versuch 5 [s]
1	3,54	3,61	3,62	3,63	3,58
2	2,43	2,35	2,42	3,37	2,32
3	1,95	1,83	1,86	1,97	1,89
4	1,43	1,58	1,62	1,64	1,89
5	1,39	1,38	1,48	1,51	1,31
6	1,27	1,24	1,23	1,25	1,21
7	1,16	1,12	1,17	1,11	1,13
8	0,97	1,05	1,08	1,05	1,09
9	1,06	0,96	1,01	1,01	1,03
10	0,96	0,96	0,94	1,00	0,98

Tabelle 8.1: Ergebnisse Messung der Zeit eines Weges der einzelnen Stufen

Mit den Ergebnisse aus der Tabelle 8.1 kann die mittlere Geschwindigkeit errechnet werden. Die Strecke dividiert mit der Zeit ergibt die mittlere Geschwindigkeit, erkenntlich in Tabelle 8.2. Dabei ist zu beachten dass die Ergebnisse in Meter pro Sekunde dargestellt sind.

Stufe	Ergebnis Versuch 1 [m/s]	Ergebnis Versuch 2 [m/s]	Ergebnis Versuch 3 [m/s]	Ergebnis Versuch 4 [m/s]	Ergebnis Versuch 5 [m/s]	\bar{O} [m/s]
1	0,0988	0,0969	0,0966	0,0964	0,0977	0,0973
2	0,1440	0,1489	0,1446	0,1476	0,1508	0,1472
3	0,1794	0,1912	0,1881	0,1776	0,1851	0,1843
4	0,2447	0,2215	0,2160	0,2134	0,2201	0,2231
5	0,2517	0,2536	0,2364	0,2317	0,2671	0,2481
6	0,2755	0,2822	0,2845	0,2800	0,2892	0,2823
7	0,3017	0,3125	0,2991	0,3465	0,3097	0,3076
8	0,3608	0,3333	0,3240	0,3333	0,3211	0,3211
9	0,3301	0,3645	0,3465	0,3465	0,3398	0,3455
10	0,3645	0,3645	0,3723	0,3500	0,3571	0,3617

Tabelle 8.2: Mittlere Geschwindigkeit und Durchschnittswert in SI-Einheiten

Da die Darstellung für in Meter pro Sekunde für Schrittmotoren unüblich sind, wurden diese mit in Millimeter pro Sekunde umgerechnet, erkenntlich in Tabelle 8.3.

Stufe	\varnothing [m/s]	\varnothing [mm/s]
1	0,0973	97,33
2	0,1472	147,2
3	0,1843	184,3
4	0,2231	223,1
5	0,2481	248,1
6	0,2823	282,3
7	0,3076	307,6
8	0,3345	334,5
9	0,3455	345,5
10	0,3617	361,7

Tabelle 8.3: Durchschnittswerte umgerechnet in mm/s

9 Offene Punkte

Dieses Projekt befindet sich noch in der Entwicklungsphase. Bei dem Demonstrator handelt es sich um einen Prototyp und es bietet noch Verbesserungs- und Erweiterungspotenzial. In diesem Fall findet ausschließlich ein Ausblick zu Lern- und Demonstrationszwecken statt.

9.1 Misslungene Messung der Beschleunigung des Schlittens

Zunächst sollte die Beschleunigung gemessen werden. Um die Beschleunigung des Schlittens zu messen, muss auf den Schlitten ein Beschleunigungssensor integriert werden. Dafür wurde ein Arduino Nano 33 BLE Sense Rev2 verwendet, der die gleiche 9-Achs-IMU (LSM9DS1) verwendet wie die im Arduino Nano 33 BLE Sense Lite. Die 9-Achs-IMU wird in Kapitel 2.2.1 genauer erläutert. Für die Befestigung des Arduinos wurde eine Konstruktion aus einem älteren Projekt verwendet, dabei handelt es sich um ein kleines Gehäuse indem der Arduino reingelegt wird und mit einer DIN 912 M3 Schraube an dem Schlitten befestigt wurde, zu sehen in Abbildung 9.1.

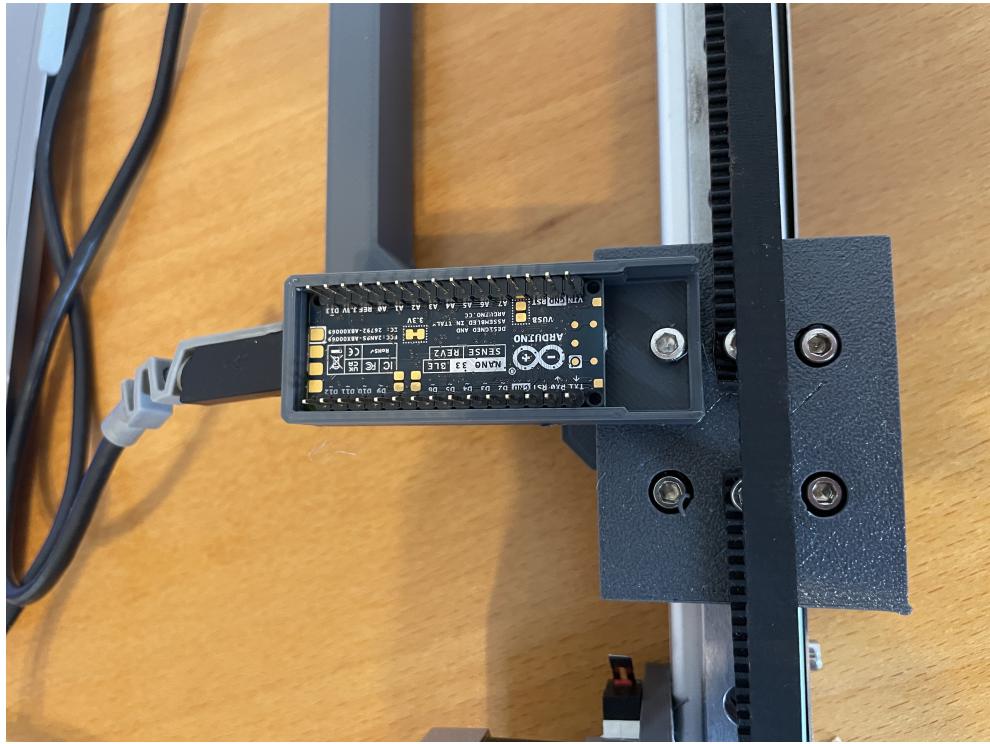


Abbildung 9.1: Montage des Arduino Nano 33 Ble Sense Rev2 auf dem Schlitten

Den notwendige Code für den Arduino lieferte die IDE mithilfe der Bibliothek ArduinoLSM9DS1 unter dem Pfad File/Examples/ArduinoLSM9DS1/SimpleAccelerometer. Der Code wurde lediglich an das Ausgeben der relevanten Achse und die Umrechnung in m/s^2 angepasst. Der Messaufbau wird in Abbildung 9.2 dargestellt.

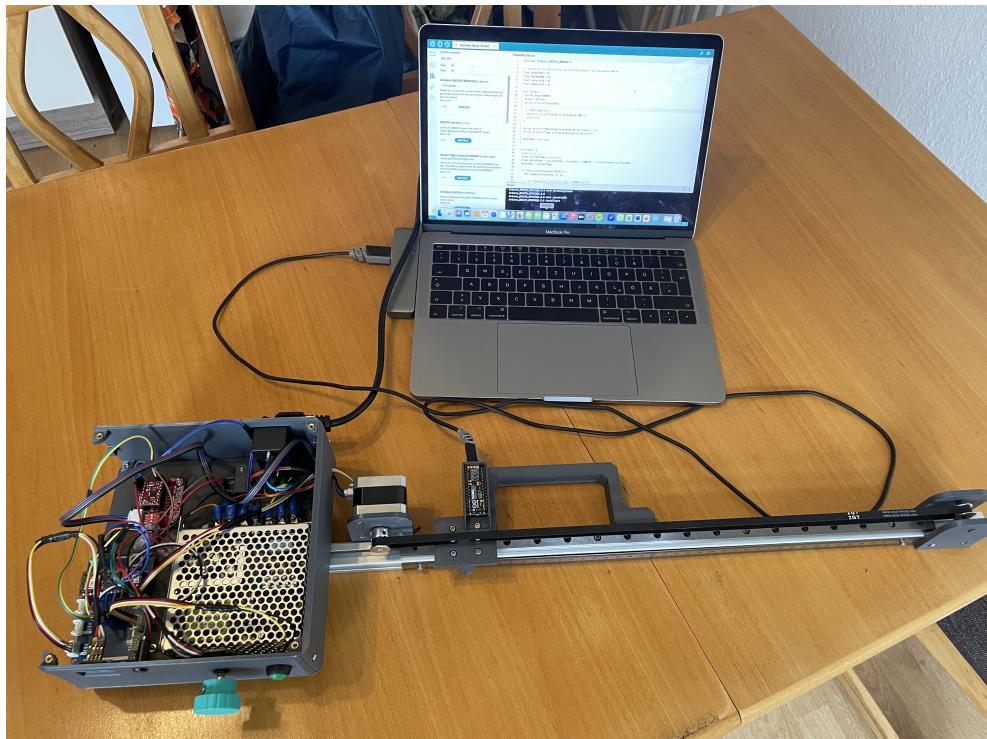


Abbildung 9.2: Testaufbau Messung Beschleunigung

Bei der Auswertung jedoch fiel auf, dass obwohl verschiedene eingestellte und auch ersichtliche Beschleunigungswerte oft der identische Wert angezeigt wurde. Es wurde vermutet, dass durch die Vibrationen und die Eigenfrequenz des Systems zu Ausreißern kommt. Zunächst wurde das Gehäuse demontiert, was jedoch keine Abhilfe brachte. Der nächste Schritt war einen Filter gegen Ausreißer in den Code einzupflegen, allerdings auch ohne Erfolg. Aus zeitlichen Gründen wurden die Bewegungsprofile in jeder Stufe auf die gleiche Beschleunigung angepasst. Unterschieden wird nur in den Geschwindigkeiten, sodass bei den Tests ausschließlich die Messung der mittleren Geschwindigkeit durchgeführt werden konnten.

9.2 Misslungene Ansteuerung der SMD-LED

Die Funktion der SMD-LED wurde im Test nachgewiesen und sie gab alle geforderten Farben aus. Die LED sollte die drei verschiedenen Demonstrator-Status anzeigen (Grün = Bereit für den Demonstrierablauf, Gelb = Referenzfahrt, Blau = Demonstrierablauf). Aufgrund mangelnder Digitalanschlüsse konnte nur die Farbe Rot ausgegeben werden, die nun den Status anzeigt, dass das System allgemein bereit ist.

9.3 Verbesserungen

Am offensichtlichsten ist die Überarbeitung des Gehäuses. Durch die Vibration des Motors wird häufig in die Eigenfrequenzen des Gehäuses gefahren, sodass eine starke Geräuschkulisse entsteht. Ein Spritzguss-Gehäuse mit verbesserter Konstruktion mit zum Beispiel die Integration von elastischen Gehäusefüßen könnten diesen Mangel beheben. Außerdem könnten weitere Einzelteile mit anderen Fertigungstechniken hergestellt werden, um bessere Qualitäten der Bauteile zu erreichen.

Softwaretechnisch könnte ein verbesserter Code zu kürzeren Rechenzeiten des Prozessors und einem stabileren Lauf führen. Außerdem könnten die Bewegungsstufen besser und genau definiert, werden um noch bessere Ergebnisse zu erzielen.

9.4 Erweiterung

Die einfachste Erweiterung ist die Integration von neuen Bewegungsstufen per Software. Darüber hinaus könnte per Drehknopf eine manuelle Eingabe einer Geschwindigkeit und Beschleunigung oder das Erreichen eines genauen Punktes hinzugefügt werden. Das Integrieren eines Beschleunigungssensors auf dem Schlitten könnte zu genaueren Stufen führen indem das System durch den Sensor und einen Regler geregelt werden könnte. Des Weiteren könnten Differenzen zwischen Eingabe und tatsächlichen Werten am OLED-Display angezeigt werden.

9.5 Anwendung in der Lehre

Der Demonstrator kann durch die kompakte Bauweise mobil mitgenommen und schnell aufgebaut werden. So kann der Demonstrator Studierenden praktische Erfahrungen in verschiedenen Fachbereichen vermitteln. Die Fachrichtungen sind Automatisierungstechnik, Elektrotechnik, Steuerungstechnik, Maschinenelemente und viele mehr. Durch das Durchfahren der Stufen werden reale Situationen von Schrittmotoren simuliert. Dies kann Einblicke in Anforderungen und Abläufe in der Industrie geben.

Literatur

- [AD24] AZ-Delivery. *1,3 Zoll OLED Display Datenblatt*. Hrsg. von AZ-Delivery. 2024. URL: <https://www.az-delivery.de/products/1-3zoll-i2c-oled-display>.
- [Ada24a] Adafruit Industries. *Adafruit GFX Library*. 2024.
- [Ada24b] Adafruit Industries. *Adafruit SSD1306 Library*. 2024.
- [Adc24] Adcanced Monolithic System. *AMS1117 ADMOS / Alldatasheet: 800mA LOW DROPOUT VOLTAGE REGULATOR*. 2024. URL: <http://www.advanced-monolithic.com/pdf/ds1117.pdf>.
- [All22] Allegro. *A4988-Datasheet*. 2022. URL: <https://www.allegromicro.com>.
- [Ard14] Arduino. *Blink: Turn an LED on and off every second*. Hrsg. von Arduino. 2014. URL: <https://docs.arduino.cc/built-in-examples/basics/Blink/>.
- [Ard22] Arduino. *Debounce on a Pushbutton*. 2022.
- [Ard24a] Arduino. *ABX00031-Datasheet*. 2024. URL: <https://docs.arduino.cc/resources/datasheets/ABX00031-datasheet.pdf>.
- [Ard24b] Arduino. *Downloads: Arduino IDE 2.3.2*. Hrsg. von Arduino. <https://www.arduino.cc/en/software>, 2024. URL: <https://www.arduino.cc/en/software>.
- [Ard24c] Arduino. *Installing Libraries*. Hrsg. von Arduino. 2024. URL: <https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-installing-a-library/>.
- [Ard24d] Arduino. *Installing a Board Package in the IDE 2*. Hrsg. von Arduino. 2024. URL: https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-board-manager/?queryID=145da8e8c0ca68927b79659df14079a5&_gl=1*1c27moy*_ga*MTQ0NDAYMjU0Ni4xNzEyMjI4NjMx*_ga_NEXN8H46L5*MTcxMjkzNTM5NS4xMi4xLjE3MTI5MzYxNTguMC4wLjE4MTYyNDUyNw.*_fplc*R3haa2kwYmJ5b0owSXBZQmNtNDElMkYySTNJZEZjVDdueVdoViUyQkg1Nn
..

- [Ard24e] Arduino. *loop()*. Hrsg. von Arduino. 2024. URL: <https://www.arduino.cc/reference/en/language/structure/sketch/loop/>.
- [Ard24f] Arduino. *setup()*. Hrsg. von Arduino. 2024. URL: <https://www.arduino.cc/reference/en/language/structure/sketch/setup/>.
- [Arm20] Arm. *Arm-Cortex-M4-Processor-Datasheet*. 2020. URL: <https://developer.arm.com/documentation/102832/latest/>.
- [Ava15] Avago Technologies. *Datasheet - APDS-9960 - Digital Proximity, Ambient Light, RGB and Gesture Sensor*. 2015. URL: https://cdn.sparkfun.com/assets/learn_tutorials/3/2/1/Avago-APDS-9960-datasheet.pdf.
- [Bab23] G. Babiel. *Elektrische Antriebe in der Fahrzeugtechnik: Lehr- und Arbeitsbuch*. 5. Auflage. Wiesbaden und Heidelberg: Springer Vieweg, 2023. ISBN: 978-3-658-40585-4. DOI: [10.1007/978-3-658-40586-1](https://doi.org/10.1007/978-3-658-40586-1).
- [Bas16] S. Basler. *Encoder und Motor-Feedback-Systeme*. Wiesbaden: Springer Fachmedien Wiesbaden, 2016. ISBN: 978-3-658-12843-2. DOI: [10.1007/978-3-658-12844-9](https://doi.org/10.1007/978-3-658-12844-9). URL: <https://link.springer.com/book/10.1007/978-3-658-12844-9>.
- [Ber18] H. Bernstein. *Elektrotechnik/Elektronik für Maschinenbauer: Einfach und praxisgerecht*. 3., überarbeitete Auflage. Lehrbuch. Wiesbaden und Heidelberg: Springer Vieweg, 2018. ISBN: 978-3-658-20837-0. DOI: [10.1007/978-3-658-20838-7](https://doi.org/10.1007/978-3-658-20838-7).
- [Ber20] H. Bernstein. *Mikrocontroller: Grundlagen der Hard- und Software der Mikrocontroller ATTiny2313, ATTiny26 und ATmega32*. 2., aktualisierte und erweiterte Auflage. Lehrbuch. Wiesbaden und Heidelberg: Springer Vieweg, 2020. ISBN: 978-3-658-30067-8.
- [Die19] Dieter Stotz. *Computergestützte Audio- und Videotechnik: Multimediatechnik in der Anwendung*. 3. Aufl. Berlin, Heidelberg: Springer Vieweg, 2019. ISBN: 978-3-662-58872-7. URL: <https://link.springer.com/book/10.1007/978-3-662-58873-4>.
- [Fau20] Faulhaber Drive Systems. *FAULHABER Tutorial: Schrittverluste verhindern bei Schrittmotoren*. Hrsg. von DR. FRITZ FAULHABER GMBH & CO. KG. Schönaich · Deutschland, 2020. URL: <https://www.faulhaber.com/de/know-how/tutorials/schrittmotoren-tutorial-schrittverluste-verhindern-bei-schrittmotoren/>.
- [Fun24] Funduino. *Der Rotary Encoder KY-040*. 2024.

- [GW22] W. Gehrke und M. Winzker. *Digitaltechnik: Grundlagen, VHDL, FPGAs, Mikrocontroller*. 8. Auflage. Berlin und Heidelberg: Springer Vieweg, 2022. ISBN: 9783662639535. URL: <https://link.springer.com/book/10.1007/978-3-662-63954-2>.
- [Glo] Global Electric Motor Solution LLC. *NEMA 17, 1.8° 42mm GM42BYG Stepper Motors*. Hrsg. von Global Electric Motor Solution LLC. URL: <https://gemsmotor.com/stepper/nema17-stepper-motor.pdf>.
- [Hag21] R. Hagl. *Elektrische Antriebstechnik*. 3., überarbeitete und erweiterte Auflage. München: Hanser, 2021. ISBN: 978-3-446-46572-5. DOI: [10.3139/9783446468214](https://doi.org/10.3139/9783446468214).
- [LD12] J. Lienig und M. Dietrich, Hrsg. *Entwurf integrierter 3D-Systeme der Elektronik*. Berlin und Heidelberg: Springer Vieweg, 2012. ISBN: 978-3-642-30572-6. DOI: [10.1007/978-3-642-30572-6](https://doi.org/10.1007/978-3-642-30572-6).
- [MS21] A. Meroth und P. Sora. *Sensornetzwerke in Theorie und Praxis*. Wiesbaden: Springer Fachmedien Wiesbaden, 2021. ISBN: 978-3-658-31708-9. DOI: [10.1007/978-3-658-31709-6](https://doi.org/10.1007/978-3-658-31709-6). URL: <https://link.springer.com/book/10.1007/978-3-658-31709-6>.
- [Mar24] Marc McComb. *Micro-stepping for Stepper Motors*. Hrsg. von Microchip Technology. 2024.
- [Mea19] Meanwell. *RD-50-20190729.cdr*. 2019. URL: https://cdn-reichelt.de/documents/datenblatt/D500/RD-50_DS-EN.PDF.
- [Men24] Mentor. *SIGNALLEUCHTENSMD-LED-Datenblatt*. 2024. URL: <https://www.mentor.de.bauelemente/product/2660-8301>.
- [Mik24] Mike McCauley. *AccelStepper Library*. 2024.
- [Nor23] Nordic Semiconductor. *nRF5340 Product Specification: QSPI — Quad serial peripheral interface*. 2023. URL: <https://infocenter.nordicsemi.com>.
- [Nor24a] Nordic Semiconductor. *nRF52840 Product Specification 2*. 2024. URL: <https://infocenter.nordicsemi.com>.
- [Nor24b] Nordic Semiconductor. *nRF52840 Product Specification Memory*. 2024. URL: <https://infocenter.nordicsemi.com>.
- [Nor24c] Nordic Semiconductor. *nRF9161 Product Specification: Cryptocell-ARM TrustZone CryptoCell 310*. 2024. URL: <https://infocenter.nordicsemi.com>.

- [PA22] Petr Filipi und Arduino Tech Support Team. *Difference_between_A33BLESense_and_SenseLite: A Difference between A N 33 BLE Sense vs. Sense Lite*. 2022. URL: <https://forum.arduino.cc/t/a-difference-between-a-n-33-ble-sense-vs-sense-lite/1030305>.
- [Pau24] Paul Stoffregen. *Encoder Library*. Hrsg. von PJRC. 2024. URL: https://www.pjrc.com/teensy/td_libs_Encoder.html.
- [RB24] Roboter-Bausatz. *Linearführung MGN12H 450mm*. Hrsg. von Roboter-Bausatz. 2024. URL: <https://www.roboter-bausatz.de/p/linearfuehrung-mgn12h-450mm>.
- [Rui16] Rui Santos. *Guide for I2C OLED Display with Arduino*. randomnerdtutorials.com, 2016.
- [SK21] D. Schröder und R. Kennel. *Elektrische Antriebe – Grundlagen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021. ISBN: 978-3-662-63100-3. DOI: [10.1007/978-3-662-63101-0](https://doi.org/10.1007/978-3-662-63101-0).
- [STM15] STMICROELECTRONICS. *Datasheet - LSM9DS1- iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer*. Hrsg. von STMICROELECTRONICS. 2015. URL: <https://www.st.com/en/mems-and-sensors/lsm9ds1.html>.
- [STM17] STMICROELECTRONICS. *Datasheet - LPS22HB-MEMS nano pressure sensor: 260-1260 hPa absolute digital output barometer*. Hrsg. von STMICROELECTRONICS. 2017. URL: <https://www.st.com/en/mems-and-sensors/lps22hb.html>.
- [STM21] STMICROELECTRONICS. *Datasheet - MP34DT05-A - MEMS audio sensor omnidirectional digital microphone*. Hrsg. von STMICROELECTRONICS. 2021.
- [Sim19] Simac Electronics GmbH. *COM-KY040RE-Datenblatt: Drehencoder mit Tasterfunktion*. 2019. URL: www. joy-it.net.
- [Web24] I. Weber. *VBA für Office-Automatisierung und Digitalisierung*. Wiesbaden: Springer Fachmedien Wiesbaden, 2024. ISBN: 978-3-658-42716-0. DOI: [10.1007/978-3-658-42717-7](https://doi.org/10.1007/978-3-658-42717-7).

Anhang

10 Schaltplan

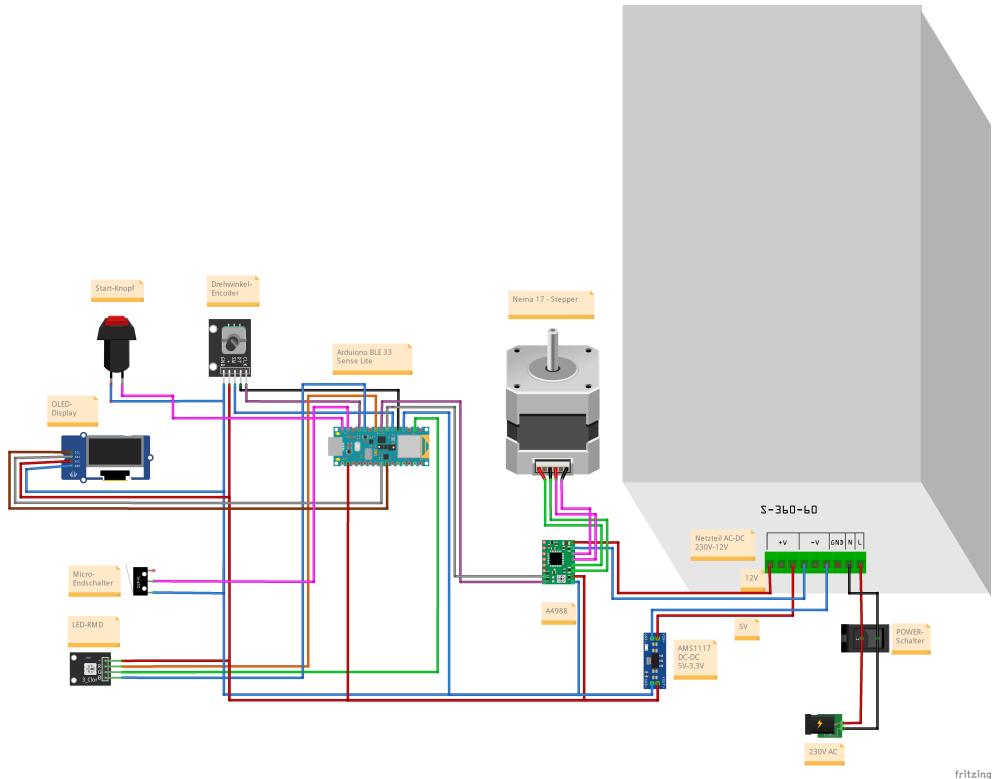
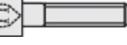


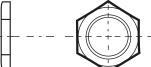
Abbildung 10.1: Schaltplan

11 Materialliste

Pos.	Stk.	Bezeichnung	Artikel-Nr.	Preis[€]	Abbildung	Bestelladresse
1	1	Arduino Lern-Kit: - Arduino Nano Sense BLE 33 BLE Lite - USB-A-Micro Verbindungska-bel - Klemmboard	ARD KIT TINYML	52,40		www.reichelt.de
2	1	CR-10 Nema 17 Schrittmotor 34 mm 42-34	RBS12536	14,25		www.roboter-bausatz.de
3	1	1 m Aluprofil 20 × 20 I-Typ Nut 5	RBS12105	11,81		www.roboter-bausatz.de
4	1	Schaltnetzteil, geschlossen, 50 W, 5/12 V, 6 A	SNT RD 50A	17,10		www.reichelt.de
5	1	Mini Spannungsregler Step-Down Modul AMS1117 3,3 VV AMS1117	RBS10505	0,65		www.roboter-bausatz.de
6	1	GT2 Riemenscheibe 20 Zähne 5 mm Bohrung für 6 mm Riemen	RBS10277	1,10		www.roboter-bausatz.de
7	1	GT2 Riemenscheibe 20 Zähne 5 mm Bohrung mit Dual Kugellager	RBS12569	1,85		www.roboter-bausatz.de
8	0,5	yourDroid High Speed PLA Grau 1.75mm 1 kg	RBS16371	16,95		www.roboter-bausatz.de

Pos.	Stk.	Bezeichnung	Artikel-Nr.	Preis[€]	Abbildung	Bestelladresse
9	1	Linearführung MGN12H 450 mm	RBS12916	29,21		www.roboterb ausatz.de
10	1	yourDroid GT2 Zahnriemen offen; 6 mm; 1 m faserverstärkt	RBS10161	2,25		www.roboterb ausatz.de
11	17	M3 Hämmermutter T-Schlitz Nut 6	RBS11229	0,29		www.roboterb ausatz.de
12	14	Zylinderschrauben M3 6 mm Innensechskant DIN 912 Edelstahl	888735	0,18		www.conrad.de
13	6	Zylinderschrauben M3 8 mm Innensechskant DIN 912 Edelstahl	827278	0,18		www.conrad.de
14	21	Zylinderschrauben M3 10 mm Innen-sechskant DIN 912 Edelstahl	888738	0,10		www.conrad.de
15	7	Zylinderschrauben M3 14 mm Innen-sechskant DIN 912 Edelstahl A2 100 St.	1061821	0,18		www.conrad.de
16	7	Sechskantmuttern M3 DIN 934 Stahl verzinkt	888718	0,02		www.conrad.de
17	2	Senkschrauben M2.5 5 mm Kreuzschlitz Phillips DIN 965 Stahl verzinkt	889754	0,06		www.conrad.de

Pos.	Stk.	Bezeichnung	Artikel-Nr.	Preis[€]	Abbildung	Bestelladresse
18	1	BMI Lineal 963050030 Maßstab 0,5 m Edelstahl	823983 - 62	9,49		www.conrad.de
19	1	65 Jumper Wire Kabel im Set	RBS10023	1,55		www.roboter-bausatz.de
20	8	40 Pin Dupont / Jumper Kabel Buchse-Buchse 20 cm	RBS101136	0,98		www.roboter-bausatz.de
21	5	40 Pin Jumper Kabel Buchse- Stecker 20 cm	RBS10038	1,15		www.roboter-bausatz.de
22	2	Arduino - Grove Universal-Kabel, 4-Pin, 20 cm	GRV CA-BLE4PIN20	3,50		www.reichelt.de
23	1	Schrittmotor-kabel 4 Pin auf 6 Pin HX2.54 50 cm	RBS10966	0,98		www.reichelt.de
24	9	Ring-Kerbschuhe, für M4, blau	VT RK-B-4	0,13		www.reichelt.de
25	3	Flachstecker-hülse, Breite: 4,7 5mm, blau	VT FSH-B-4,75	0,10		www.reichelt.de
26	1	Anschlussleitung H07RN-F, 3 x 2,5 mm, 10 m	H07RN F325 10M	31,40		www.reichelt.de
27	1	Netzkabel, Schutzkontakt-stecker gew, 1,8 m, schw, C13 gew	NKSK 200 SW GEW	3,00		www.reichelt.de

Pos.	Stk.	Bezeichnung	Artikel-Nr.	Preis[€]	Abbildung	Bestelladresse
28	2	Hebelklemme 3-polig Ø 0,2 mm - 2,5 mm 10 Stück	4650057	4,99		www.obi.de
29	1	Artillery Siedewinder X2 Wippschalter AC-01	RBS15072	4,19		www.roboterb ausatz.de
30	1	Entwicklerboard Drehwinkel-Encoder	KY-040	2,20		www.reichelt.de
31	1	Kontermutter für Potentiometerknöpfe	P4-Mutter	0,22		www.reichelt.de
32	1	Drucktaster Button Schalter Grün 12 mm; 250 V; 1 A	RBS13869	0,85		www.roboterb ausatz.de
33	1	Mikroschalter / Miniatur Endschalter	RBS 10856	0,49		www.roboterb ausatz.de
34	1	Entwicklerboard Schrittmotorsteuerung, A4988	DEBO DRV A4988	5,80		www.reichelt.de
35	1	Signallleuchte, SMD-LED, ø3 mm, Kunststoff	MEN 2660.8301	4,60		www.reichelt.de
36	1	0,96 Zoll OLED Display I2C/IIC/TWI 128x64 Pixel für Arduino	RBS10350	4,05		www.roboterb ausatz.de

Pos.	Stk.	Bezeichnung	Artikel-Nr.	Preis[€]	Abbildung	Bestelladresse
37	14	Einpressmuttern $M3 \times 4 \times 4$ 100 Stück für Voron 2.4	RBS15553	3,14		www.roboterbausatz.de
38	0,1	Sunlu PLA Filament Grass Green 1.75mm 1kg	RBS16624	17,95		www.roboterbausatz.de

Tabelle 11.1: Materialliste

Index

- Inertialmesseinheit
 - siehe* IMU, xi, 8
- AAR, xi, 13
- ADC, xi, 12
- Advanced Encryption Standard
 - siehe* AES, xi, 13
- AES, xi, 13
- Analog to Digital Converter
 - siehe* ADC, xi, 12
- Automatic Address Recognition
 - siehe* AAR, xi, 13
- BLE, xi, 7, 8, 10
- Bluetooth Low Energie
 - siehe* BLE, xi, 7
- CCI, xi, 3, 49
- CCM, xi, 13
- Computer Aided Design
 - siehe* CCI, xi, 3
- Counter with CBC-MAC
 - siehe* CCM, xi, 13
- Direkt Memory Access
 - siehe* DMA, xi, 12
- DMA, xi, 12
- HDMI, xi, 17
- Hybrid
 - siehe* HDMI, xi, 17
- I²C, xi, 8–10, 25
- IC, xi, 9, 10
- IMU, xi, 8
- Inter Circuit
 - siehe* IC, xi, 9
- Inter-Integrated Circuit
- LED, xi, 9
- Light Emitting Diode
 - siehe* LED, xi, 8
- Micro Universal Serial Bus
 - siehe* Micro USB, xi, 12
- Micro USB, xi, xii, 9, 12
- Near Field Communication
 - siehe* NFC, xi, 12
- NFC, xi, 12
- OLED, xi, 11
- Organic Light Emitting Diode
 - siehe* OLED, xi, 11
- PDM, xi, 10
- Pulsdichtenmodulation
 - siehe* PDM, xi, 10
- SCL, xi, 11
- SDA, xi, 11
- Serial Clock
 - siehe* SCL, xi, 11
- Serial Data
 - siehe* SDA, xi, 11
- Serial Peripheral Interface
 - siehe* SPI, xii, 9
- SMD, xi, 10
- SoC, xii, 7
- SPI, xii, 9, 13
- Surface Mount Device
 - siehe* SMD, xi, 10
- System on Chip
 - siehe* SoC, xii, 7
- Ultraviolet-Filter
 - siehe* Micro USB, xii, 9