

DIPLOMARBEIT

Test Station for CubeSat STS1



Ausgeführt im Schuljahr 2023/2024 von:

Sophia Hagen

Betreuer

5BHEL Dipl.-Ing. Bischof Gerold

Julius Scherrer

5BHEL

Constantin Zumtobel

5BHEL

Sebastian Bellai

5AHEL

Rankweil, am 05.04.2024

Abgabevermerk:

DA original, am 05.04.2024

.....

Dipl.-Ing. Bischof Gerold

DA digital, am 05.04.2024

.....

Dipl.-Ing. Bischof Gerold

1 Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Rankweil, am 05.04.2024

.....
Sophia Hagen

.....
Julius Scherrer

.....
Constantin Zumtobel

.....
Sebastian Bellai

Inhaltsverzeichnis

1 Eidesstattliche Erklärung	2
2 Diplomarbeit Dokumentation	9
3 Zusammenfassung	9
4 Diploma Thesis Documentation	10
5 Abstract	10
6 Danksagung	11
7 Pflichtenheft	12
7.1 Einleitung	12
7.1.1 Projektname	12
7.2 Motivation	12
7.2.1 Ausgangslage	12
7.2.2 Zielsetzung	12
7.2.3 Ziele <i>Must-Have</i>	14
7.2.4 Optionale Ziele <i>Nice-to-Have</i>	14
7.3 Meilensteine	14
7.4 Team	15
7.4.1 Teammitglieder	15
7.4.2 Betreuungslehrer	17
8 Abkürzungen	18
9 Hardware	20
9.1 Gehäuse	20
9.1.1 Holzplatte	22
9.1.2 PVC-Platten	24
9.1.3 Tür	24
9.1.4 Dichtungsrahmen	26
9.1.5 Schließer	28
9.2 Gyroskop	30
9.2.1 Motor	32
9.2.2 Motor-Treiber	33
9.2.3 Netzteil	34

9.2.4	Relais	35
9.3	Leiterplatte	36
9.3.1	Schaltplan	36
9.3.2	PCB	39
9.4	Sensoren in der Testkammer	42
9.4.1	Temperatursensor	43
9.4.2	UV-Sensor	44
9.4.3	Magnetsensor	46
9.4.4	Drucksensor	47
9.5	Sensoren auf dem Satellit	49
9.5.1	EDU	49
9.5.2	Sensorik des EDUs	50
9.5.3	Temperatursensor	50
9.5.4	Magnetfeldsensor/Gyroskop	51
9.5.5	Beschleunigungssensor	51
9.5.6	UV-Sensor	51
9.5.7	Dosimeter	51
9.6	UV-Lampe	53
9.7	Magnetfeld	55
9.8	Pneumatik	56
9.8.1	Ventil	57
9.8.2	Vakuum Ejektor	58
9.8.3	Druckluft Kugel Vibrator	60
9.8.4	Rüttelplatte	61
9.9	LED-Streifen	63
9.10	CubeSat	65
9.11	Halterung	68
9.11.1	Boden	68
9.11.2	Deckel	70
9.11.3	3D-Druck der Halterung	71
9.12	Lüftung	73
9.12.1	Pinbelegung des BC547	73
9.12.2	Aufbau Lüfter	74
9.12.3	Befestigung	75
9.13	Kühlung	76
9.13.1	Verkabelung des Kühlgerät	76
9.13.2	Verdrahtung der Taster	78

9.13.3 Aufbau mittels Streifenrasterplatine	78
9.13.4 Befestigung des Kühlgeräts an der Teststation	80
9.14 Sicherheit	81
9.14.1 Notausschalter	81
9.14.2 Schlüsselschalter	82
9.14.3 Endschalter	82
10 Software	84
10.1 Tasterabfrage	84
10.2 Testprogramm	85
10.2.1 GUI	85
10.2.2 Button	87
10.2.3 Temperatursensor	87
10.2.4 UV-Sensor	88
10.2.5 Drucksensor	89
10.2.6 UV-Lampe	90
10.2.7 Motor	91
10.2.8 Netzteil	91
10.2.9 LED-Streifen	92
10.2.10 Schlüsselschalter	92
10.2.11 Notausschalter	93
10.2.12 Endschalter	93
10.2.13 Kühlung	93
10.2.14 Lüfter	95
10.3 Verbindungsmöglichkeit	96
10.3.1 UDP	96
10.3.2 TCP	97
10.3.3 Serielle Verbindungsmöglichkeit	97
10.4 Übertragungsmethode	99
10.4.1 Senderoutine	99
10.4.2 Programm SensorOut.py	100
10.5 Empfangsroutine	102
10.5.1 Schreiben der Daten in eine CSV-Datei	104
10.5.2 CSV-Datei Programm	105
10.5.3 Sensorik mittels I ² C-Interface lesen	110
10.6 Bibliothek „STS1_Sensors.py“	110
10.7 Auslesen der Sensorik mittels „STS1_Sensors.py“	111

11 Steuerung	113
11.1 Raspberry Pi	113
11.2 Grafische Oberfläche Raspberry Pi	114
11.3 Remote-Zugriff	115
11.4 Fast API	117
11.4.1 Erstellen einer Fast API	117
11.4.2 Temperatursensor	120
11.4.3 UV-Sensor	120
11.4.4 Drucksensor	121
11.4.5 Schalten zwischen High und Low	121
11.4.6 Motor und Endschalter	122
11.5 Webapplikation	123
11.5.1 Wie funktioniert Dash	123
11.5.2 Callbacks	123
11.5.3 HTML-Objekte in Dash	124
11.5.4 Das Programm „dashboard.py“	125
11.5.5 Das HTML-Layout	126
11.5.6 Die Funktion ‚generate_figure‘	127
11.5.7 Die Callbacks der Website	129
11.5.8 Die Funktion ‚update_args‘	130
11.5.9 Design der Webapplikation	131
11.5.10 Steuerung von Aktoren	132
11.5.11 MQTT	132
11.5.12 Fast API	133
12 Testen	134
12.1 Testroutine	134
12.1.1 Rotationstest	134
12.1.2 Vibrationstest	134
12.1.3 Test im Vakuum	135
12.1.4 Einsatz von Sensoren	135
12.1.5 UV-Lampe	135
12.1.6 Fazit der Tests	136
12.2 Messungen	137
12.2.1 Erklärung der Daten	137
12.2.2 Temperatur	137
12.2.3 Messung des Magnetfeldes	137
12.2.4 Messung von Beschleunigung	138

12.2.5	Messung der UV-Strahlung	138
12.2.6	Messung der Luftfeuchtigkeit	138
12.2.7	Messung des Luftdrucks	139
12.2.8	Messung der Luftqualität	139
12.2.9	Messergebnisse der Sensoren	139
12.2.10	Messergebnis Temperatur	139
12.2.11	Messergebnis Luftfeuchtigkeit	141
12.2.12	Messergebnis Luftdruck	141
12.2.13	Messergebnis des UV-Sensors	143
12.2.14	Messergebnis der Geschwindigkeit und des Gyroskops	144
13	Dokumentationsaufteilung	146
13.1	Sophia Hagen	146
13.2	Julius Scherrer	146
13.3	Constantin Zumtobel	147
13.4	Sebastian Bellai	148
14	Projektmanagement	149
14.1	Projektmanagementtool	149
14.1.1	GanttProject	149
14.1.2	GitHub	149
14.2	Einteilung des Projektes	151
14.3	Arbeitsaufteilung	151
14.4	Arbeitszeiten	151
15	Probleme	153
16	Verzeichnisse	154
17	Anhang	165
17.1	Entwicklungsumgebung für Hardware	165
17.1.1	Solidworks ¹	165
17.1.2	Eagle 7.7.0	165
17.2	Entwicklungsumgebung für Software	166
17.2.1	Thonny	166
17.2.2	Visual Studio Code ²	166

¹ SOLIDWORKS.

² Visual Studio Code - Code Editing. Redefined.

17.2.3 Plotly Dash	167
17.2.4 TeraTerm	167
17.3 3D-Druck	167
17.4 Datenblätter	168

2 Diplomarbeit Dokumentation

Verfasser/in:

Sophia Hagen , Julius Scherrer , Constantin Zumtobel , Sebastian Bellai

Klassen/Schuljahr: 5AHEL und 5BHEL, Schuljahr 2023/2024

Thema der Diplomarbeit: Test Station for CubeSat STS1

Kooperationspartner: TU Wien Space Team,³ Omicron electronics GmbH,⁴ Bachmann electronic GmbH,⁵ Hefel Technik GmbH,⁶ Tschabrun Hermann Gesellschaft m.b.H.,⁷ Robotunits GmbH⁸

Aufgabenstellung:

Es soll eine Teststation entwickelt werden, die die Zuverlässigkeit des CubeSats testet. Der CubeSat ist ein kleiner Satellit der vom SpaceTeam der TU-Wien entwickelt wurde.

3 Zusammenfassung

Das Spaceteam der TU-Wien benötigt eine Teststation, um die Zuverlässigkeit des CubeSat zu überprüfen. Dafür soll eine Kammer gebaut werden, in der verschiedene Tests mit dem Satelliten durchgeführt werden können. Die Kammer soll bei einer Temperatur von 0°C bis 30°C betrieben werden können. In der Kammer befinden sich verschiedene Sensoren, um einen Vergleich zwischen den Sensoren auf dem CubeSat und in der Teststation herzustellen. Durch diesen Ist-Soll Vergleich können Fehler auf dem CubeSat erkannt werden und fehlerhafte Bauteile ausgetauscht werden. Um diesen Ist-Soll-Vergleich zu veranschaulichen und um alle Komponenten anzusteuern, soll eine benutzerfreundliche Webapplikation entwickelt werden. Mithilfe eines Gyroskops wird der CubeSat in x-, y- und z-Richtung rotiert. Mit den Sensoren auf dem Satellit können die Auswirkungen dieser Rotation erfasst werden und mithilfe der Webapplikation dargestellt werden.

³ Homepage.

⁴ OMICRON.

⁵ Bachmann electronic GmbH - Bachmann electronic GmbH.

⁶ Hefel Technik auf Facebook.

⁷ Tschabrun Holz- und Baustoffe in Rankweil, Bludenz-Bürs und Innsbruck | Tschabrun Holz & Baustoffe - Starke Produkte. Starke Beratung.

⁸ Robotunits – Der Automatisierungsbaukasten.

4 Diploma Thesis Documentation

Author(s):

Sophia Hagen , Julius Scherrer , Constantin Zumtobel , Sebastian Bellai

Class/ Academic year: 5BHEL and 5AHEL, 2023/2024 Diplo Thesis Topic: Test Station for CubeSat STS1

Cooperation Partners: TU Wien Space Team,⁹ Omicron electronics GmbH,¹⁰ Bachmann electronic GmbH,¹¹ Hefel Technik GmbH,¹² Tschabrun Hermann Gesellschaft m.b.H.,¹³ Robotunits GmbH¹⁴

Assignment of Tasks:

A test station is to be developed that tests the reliability of the CubeSat. The CubeSat is a small satellite developed by the SpaceTeam at TU Vienna.

5 Abstract

The TU Vienna space team needs a test station to check the reliability of the CubeSat. For this purpose, a chamber is to be built in which various tests can be carried out with the satellite. The chamber should be able to operate at a temperature of 0°C to 30°C. There are various sensors in the chamber to make a comparison between the sensors on the CubeSat and in the test station. Through this actual-target comparison, errors on the CubeSat can be identified and faulty components can be replaced. In order to illustrate this actual-target comparison and to control all components, a user-friendly web application should be developed. Using a gyroscope, the CubeSat is rotated in the x, y and z directions. The sensors on the satellite can be used to record the effects of this rotation and display them using the web application.

⁹ Homepage.

¹⁰ OMICRON.

¹¹ Bachmann electronic GmbH - Bachmann electronic GmbH.

¹² Hefel Technik auf Facebook.

¹³ Tschabrun Holz- und Baustoffe in Rankweil, Bludenz-Bürs und Innsbruck | Tschabrun Holz & Baustoffe - Starke Produkte. Starke Beratung.

¹⁴ Robotunits – Der Automatisierungsbaukasten.

6 Danksagung

Wir danken all unseren Sponsoren für die Unterstützung, ohne Sie wäre die Umsetzung der Diplomarbeit nicht möglich gewesen.

- Omicron electronics GmbH¹⁵
- Bachmann electronic GmbH¹⁶
- Hefel Technik GmbH¹⁷
- Tschabrun Hermann Gesellschaft m.b.H.¹⁸
- Robotunits GmbH¹⁹

Wir möchten auch dem Space Team²⁰ der TU-Wien danken, im speziellen bei Herrn Behrle, die uns in all diesen Monaten beraten haben.

Wir möchten unserem Betreuungslehrer DI Bischof Gerold für die Hilfe bei der Umsetzung bedanken.

Ebenfalls möchte sich jedes Teammitglied bei seiner/ihrer Familie bedanken. Diese mussten in den letzten Monaten auf ihr Kind verzichten und einige Stimmungsschwankungen ertragen.

Herzlichen Dank!

¹⁵ OMICRON.

¹⁶ Bachmann electronic GmbH - Bachmann electronic GmbH.

¹⁷ Hefel Technik auf Facebook.

¹⁸ Tschabrun Holz- und Baustoffe in Rankweil, Bludenz-Bürs und Innsbruck | Tschabrun Holz & Baustoffe - Starke Produkte. Starke Beratung.

¹⁹ Robotunits – Der Automatisierungsbaukasten.

²⁰ Homepage.

7 Pflichtenheft

7.1 Einleitung

7.1.1 Projektname

Der Projektnname lautet "Test Station for CubeSat STS1". Der Name wurde von der TU-Wien vorgegeben. Für die Diplomarbeit wurde ein Logo erstellt.



Abbildung 1: Logo

7.2 Motivation

7.2.1 Ausgangslage

Die TU-Wien hat der HTL-Rankweil verschiedenen Ideen für eine Diplomarbeit vorgestellt. Das Space-Team hat ihre Anforderungen aufgelistet. Angelehnt an diese Anforderungen wurde ein Konzept entwickelt. Dieses wurde in einem Zeitraum vom September 2023 bis März 2024 umgesetzt.

7.2.2 Zielsetzung

Mit der Teststation soll die Zuverlässigkeit der Payload und des CubeSat getestet werden. Die Teststation soll so gebaut werden, dass Umwelteinflüsse möglichst gering gehalten werden und die Station gut transportierbar ist. In der Teststation sollen Vorrichtungen errichtet werden, um die Zuverlässigkeit des Satelliten testen zu können. Zu diesen Vorrichtungen zählen folgende Dinge:

- ein Gyroskop, um den Satellit auf der x-Achse, y-Achse und z-Achse zu rotieren.
- eine UV-Lampe, um die Auswirkungen von UV-Strahlen darzustellen
- ein Kühlgerät, um die Teststation im Bereich von 0°C bis 30°C zu betreiben.
- ein Temperatursensor, um die Temperatur in der Teststation zu messen.
- eine Vakuum-Ejektor, um den Satellit in einem Vakuum zu betreiben.
- ein Dauermagnet, um ein Magnetfeld zu erzeugen.
- eine Rüttelplatte.

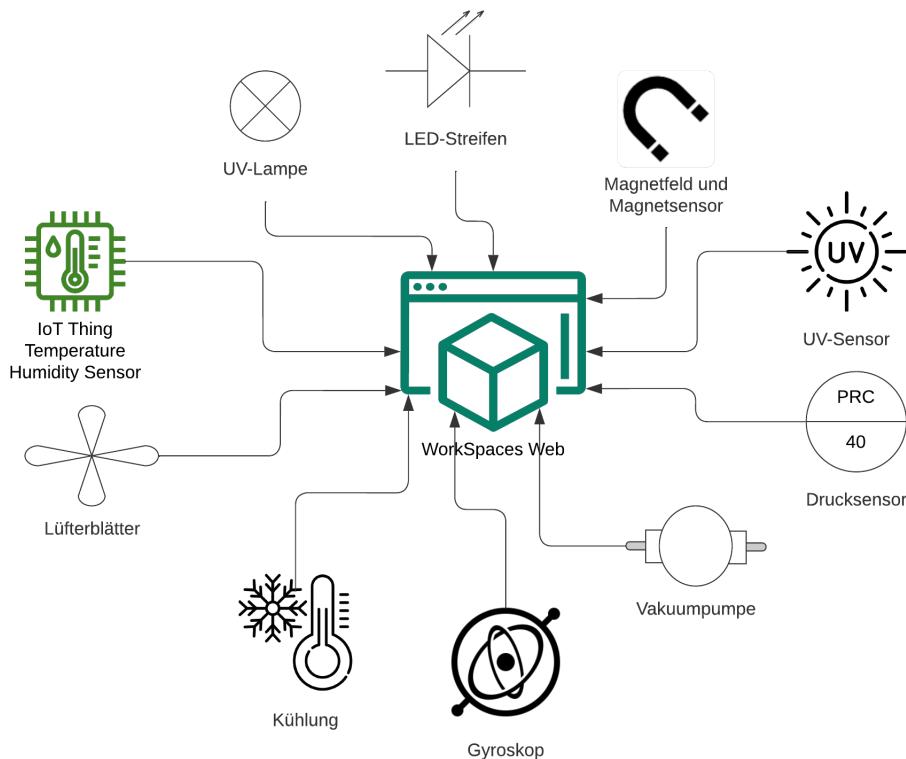


Abbildung 2: Blockschaltbild

Das Setup, mit dem die Teststation gesteuert wird, soll einfach zu bedienen sein. Die Datenauslesung und die Ansteuerung der Geräte sollen in der selben Web-Applikation erfolgen.

7.2.3 Ziele Must-Have

- Rotation des Satellit
- Messen der folgenden Auswirkungen auf den CubeSat
 - Magnetfeld
 - Temperatur
 - UV-Strahlung
- Datenauslesung
- Benutzerfreundliche Bedienoberfläche

7.2.4 Optionale Ziele Nice-to-Have

- Eine Vorrichtung, um ein Vakuum zu erzeugen.
- Auswirkungen von Vibrationen auf den CubeSat

7.3 Meilensteine

Meilenstein	zu erledigende Arbeit	bis
Gehäuse	Konstruktion und Aufbau des Gehäuses für die Teststation	06.11.2023
Testkammer	Sensoren einbauen Gyroskop aufbauen Vorrichtung für Vakuum Kühlung UV-Lampe	10.02.2024
Daten ein- und auslesen	Programm für Ein- und Auslesen der Sensoren verschiedene Sendemethoden testen	12.02.2024
Limits Testen	Kritische Werte testen	01.03.2024
Abgabe Diplomarbeit	Dokumentation abgeben	20.03.2024

Tabelle 1: Meilensteine

7.4 Team

7.4.1 Teammitglieder



Sophia Hagen
5BHEL

Abbildung 3: Sophia Hagen



Julius Scherrer
5BHEL

Abbildung 4: Julius Scherrer



Constantin Zumtobel
5BHEL

Abbildung 5: Constantin Zumtobel



Sebastian Bellai
5AHEL

Abbildung 6: Sebastian Bellai

7.4.2 Betreuungslehrer



Dipl. Ing. Gerold Bischof
gerold.bischof(at)htl-rankweil.at

Abbildung 7: Diplo. Ing. Gerold Bischof

8 Abkürzungen

TU	Technische Universität
PCB	Printed Circuit Board
UV	Ultraviolett
I2C	Inter Integrated Circuit
GUI	graphical user interface
VS	Visual Studio
CAD	Computer-Aided Design
3D	dreidimensional
SSH	Secure Shell
RDP	Remote Desktop Protocol
VNC	Virtual Network Computing
API	Application Programming Interface
GPIO	General Purpose Input Output
hPa	HektoPascal
m	Meter
mm	Millimeter
bzw	beziehungsweise
ADC	analog-to-digital converter
DA	Diplomarbeit
IoT	Internet of Things
ADC	analog-to-digital converter
TCP	Transmission Control Protocol
ACK	Acknowledgement BIT
SYN	Synchronize Sequence Numbers

UDP	User Datagram Protocol
USB	Universal Serial Bus
RX	Receive Pin
TX	Transmit Pin
UART	Universal Asynchronous Receiver Transmitter
CSV	Comma Separated Values
MQTT	Message Queuing Telemetry Transport
CM3+	Compute Module 3+
EDU	Educational Module
UVA	Ultravioletstrahlung des Typ A
UVB	Ultraviolettstrahlung des Typ B
cm	Zentimeter

9 Hardware

9.1 Gehäuse

Damit die Tests in einer hochwertigen Umgebung durchgeführt werden können, ist ein professionelles Gestell der Grundstein für eine ausgezeichnete Teststation. Dadurch der Korpus stabil, leicht, einfach transportierbar und optisch bewundernswert sein sollte, wurden uns von der Robotunits GmbH 40x40 Alu Profile²¹ zu Verfügung gestellt. Diese Profile sind universell einsetzbar, erfüllen unsere Ansprüche und können so eine perfekte Basis für unsere Teststation bilden.

Die Profile werden mit einem Gewinde Stirnseitig geliefert. An den Gewinde werden sogenannten 90° Verbinder²² montiert, denn so kann man die verschiedenen Teile einfach zusammenschrauben.



(a) Profil



(b) Verbinder

Abbildung 8: Profil und Verbinder

²¹ Alu Profile 40 x 40 bis 80 x 80 | Robotunits.

²² Verbinder 40x40 | Robotunits Verbindungstechnik Raster 40.

Das Gestell wird in drei Bereiche aufgeteilt:

- Bedienfläche
 - Die oberste Fläche wird als Arbeitsfläche für den Anwender verwendet. Auf dieser Fläche kann ein Bildschirm oder ein Laptop platziert werden, und von dort aus die Tests steuern und die Daten auszulesen.
- Testkammer
 - In der mittleren Ebene wird eine Kammer gebaut, in der die Tests durchgeführt werden können. Dieser Bereich bekommt Wände aus PVC Platten und kann mit einer Tür verschlossen werden. Darin wird ein Gyroskop, Sensoren, eine Lampe und sonstige Hardwareteile platziert.
- Steuereinheit
 - In dem untersten Bereich werden die Steuerungsgeräte platziert wie zum Beispiel der Raspberry, Netzteile, Treiber und sonstige Ansteuerungsgeräte. Dieser Bereich wird mit Plexiglas als Wände verschlossen um von Staub und sonstigen Einwirkungen zu schützen.

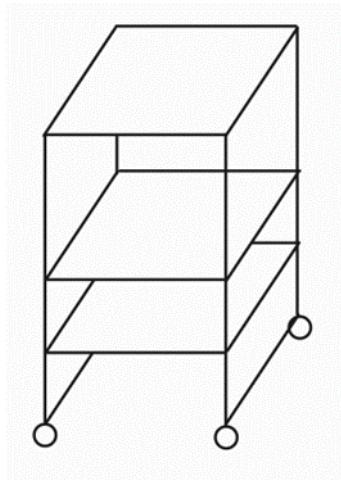


Abbildung 9: Skizze Gestell

Maße der Profile:

Die Maße wurden so gewählt, dass das Gestell eine Grundfläche von 78x78cm hat. Diese Fläche ist groß genug, um einen 27 Zoll Monitor einfach zu platzieren. Die Vertikalen Profile haben eine Länge von 100cm, denn so wird mit den Rollen eine Gesamthöhe von 110cm erreicht. Diese Höhe gewährleistet eine gesunde und komfortable Arbeitshöhe.

Das Gewicht vom Gesamtgestell aus den Aluprofilen, Rollen und Verbindungen beträgt 22kg.

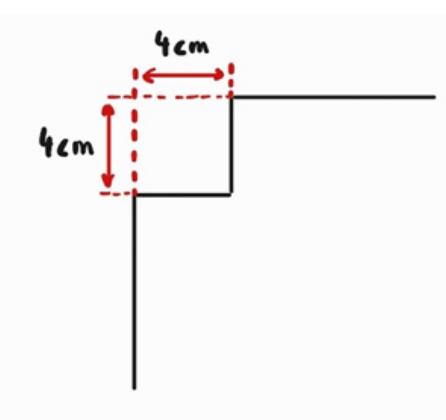
Für das Gestell sind folgende Teile nötig:

Bezeichnung	Stückzahl
70cm Aluminiumprofil 40x40mm	14
100cm Aluminiumprofil 40x40mm	4
Stirnverbinder 40x40mm	28
Einschwenkmutter M8	30
Rollen mit Rückenloch fixierbar	2
Rollen mit Rückenloch	2
Abdeckkappe 40x40mm	4

Tabelle 2: Stückliste Gestell

9.1.1 Holzplatte

Die verschiedenen Bereiche der Teststation werden mittels 3 Schicht Holzplatten, die eine Stärke von 18mm haben, getrennt. Diese Platten haben die Vorteile, dass sie sehr formstabil und langlebig sind. Außerdem weisen sie dank drei Schichten Massivholz eine hohe Biegefestigkeit. Dankensweise wurden uns die Platten von der Firma Tschabrun Herman Gesellschaft m.b.H. gesponsert. Da die Grundfläche vom Gestell die Fläche von 78x78cm hat, wird aus optischen Gründen die Arbeitsfläche mit den Maßen von 82x82cm gewählt. So steht die Platte auf jeder Seite 2cm über das Alugestell. Damit die Platten in der mittleren und unteren Ebene bündig mit dem Gestell sind, haben diese ein Maß von 78x78cm. Bei diesen Platten müssen die Ecken ausgesägt werden, damit sie formschlüssig mit dem Gestell sind. Diese Aussparungen sind 4x4cm groß, genauso groß wie die Aluprofile.



(a) Aussparung Maße



(b) Aussparung umgesetzt

Abbildung 10: Aussparung

Damit die Wände befestigt werden können, werden in eine Holzplatte Nuten gefräst. So können die Platten nicht mehr verschoben werden und können fest verklebt werden. Die Nut ist 5mm tief und 8mm breit. Auf der Vorderseite, wird ein Stahlrahmen befestigt, der auch in eine Nut eingelassen wird, diese Nut ist jedoch 4mm breit und 5mm tief.

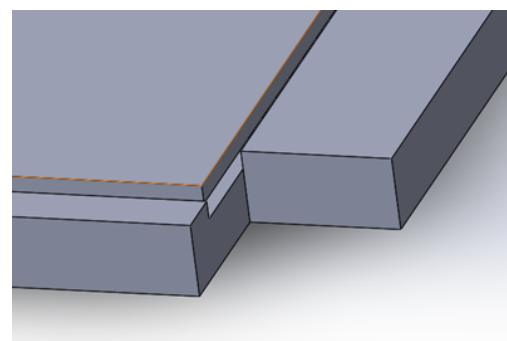
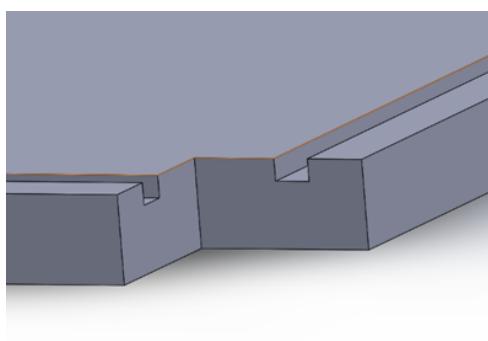


Abbildung 11: Nut Vorderseite

Jede Holzplatte wird mit M8 Schrauben und Nutsteine angeschraubt. Dazu werden 8mm Löcher in die Holzplatten gebohrt und anschließen werden die die Löcher mit einem Senkbohrer gebohrt, dadurch können die Schrauben bündig mit der Platte verschraubt werden.

9.1.2 PVC-Platten

In der mittleren Ebene werden Wände montiert, damit die Kammer luft- und staubdicht ist. Dazu eignen sich Hart PVC-Platten mit einer Dicke von 8mm. Die Platten sind elektrisch gut isoliert, haben eine große Beständigkeit gegen Chemikalien, sind hitzebeständig bis 60°C und außerdem sorgen sie für ein einzigartiges Design unserer Teststation. Die Platten werden mit einem Pattex Montagekleber verklebt, denn dieser Kleber ist speziell geeignet für PVC-Materialien und überzeugt mit einem sehr starken Halt.

Bezeichnung	Stückzahl
Hart PVC Schwarz 80x56cm	3
Pattex Montagekleber	2

Tabelle 3: Stückliste PVC-Platte

Die PVC-Platten werden in die Nut der Aluprofile gesteckt. Da oben die Querstangen mit Verbindungen ausgestattet sind, behindern diese die PVC-Platten. Aus diesem Grund muss jeweils in den Ecken eine 1x2cm Aussparung entfernt werden.

9.1.3 Tür

Es soll möglich sein, die Kammer zu verschließen, um äußerliche Einwirkungen zu verhindern. Zusätzlich sollte die Kammer luftdicht sein, denn weder warme oder kalte Luft sollte entfliehen können. Da auch das Gyroskop akustisch etwas laut ist, dient die Tür auch als Geräusch Verminderung. Der Rahmen für die Tür wird aus Holzleisten gefertigt, die eine Stärke von 2cm und eine Breite von 4cm haben.

Bezeichnung	Stückzahl
Holzleiste 4x2x70cm	2
Holzleiste 4x2x56cm	2
Holzleiste 0.2x54x68cm	1
Edelstahlgriff Lochabstand 128mm	1
M3 Schraube mm	2
M3 Mutter	2
Silikon Kleber	1

Tabelle 4: Stückliste PVC-Platte

Die Holzleisten werden auf Gehrung verbunden, denn durch diese Methode vergrößert sich die Kontaktfläche und die Stabilität wird erhöht. Plexiglas ist ein transparentes Kunststoffmaterial, das als Ersatz für Glas verwendet wird. Es ist leichter, günstiger und bruchsicherer als Glas. Das Plexiglas hat eine Stärke von 2mm und wird in eine Nut im Rahmen eingelassen, um eine einfache Befestigung zu gewährleisten.

Zusammenbau: Um sicherzustellen, dass das Plexiglas während des Zusammenbauens und Verleimens ordnungsgemäß in den Rahmen eingesetzt wurde, waren folgende Schritte erforderlich:

1. Vorbereitung der Leisten

- Zunächst wurde eine lange Leiste mit der Nut nach unten auf die Arbeitsfläche (Fläche X) gelegt.
- Eine kurze Leiste mit der Nut nach unten wurde bündig an der rechten Seite der langen Leiste platziert. Dieser Schritt wurde auch auf der linken Seite der langen Leiste mit einer weiteren kurzen Leiste wiederholt.

2. Fixierung mit Klebeband:

- Die drei Leisten wurden mit Klebeband zusammengehalten, um zu verhindern, dass sie beim Verleimen verrutschen.

3. Vorbereitung für den Leimauftrag:

- Die drei Leisten wurden zusammen um 180° gedreht. Holzleim wurde großzügig auf die Oberfläche aufgetragen und gleichmäßig verteilt.

4. Einsetzen des Plexiglasses:

- Das Plexiglas wurde vorsichtig in die Nut der langen Leiste eingelegt. Hierbei konnte eine zweite Person von Vorteil sein, um das Plexiglas zu halten, während die anderen beiden Leisten um 90° angehoben wurden. Dabei wurde darauf geachtet, dass das Plexiglas auch in den kurzen Leisten eingespannt war.

5. Abschluss des Rahmens:

- Abschließend wurde die verbleibende zweite lange Leiste auf die kurzen Seiten und das Plexiglas gelegt.

6. Fixierung und Trocknung:

- Schraubzwingen wurden verwendet, um den Rahmen fest zusammenzuspannen und den Leim trocknen zu lassen. Durch diese Schritte wurde sichergestellt, dass das Plexiglas ordnungsgemäß in den Rahmen eingesetzt wurde und während des Verleimens nicht verrutschte. Mit einem Silikonkleber wird das Plexiglas mit der Nut verklebt und verdichtet. Nachdem das Silikon ausgetrocknet war, war die Tür dicht. Als Türgriff wird ein Edelstahlgriff mit einem Lochabstand von 13cm verwendet. Der Griff kann dank einer soliden Verschraubung sicher und stabil angebracht werden, was eine zuverlässige Nutzung gewährleistet. Die Tür wird mit einfachen Scharnieren an dem Gestell befestigt.

9.1.4 Dichtungsrahmen

Damit man die Tür dicht verschließen kann und diese einen Anschlag hat, braucht es einen Rahmen. Der Rahmen wird aus 4mm dickem und 35mm breiten Flachstangen erstellt. Auf diesen Rahmen kommt ein Dichtungsband und die Tür wird dann mit sogenannten Kniehebelspanner an den Rahmen gedrückt. So ist es möglich, die Kammer Luft und Staubdicht zu verschließen. Der Rahmen wird an den Seiten an das Gestell geklebt und unten sowie oben in eine Nut eingelassen.

Bezeichnung	Stückzahl
Flachstange 35x580mm	2
Flachstange 20x664mm	1
Flachstange 35x664mm	1
Schaumisolierung 3m	1

Tabelle 5: Stückliste Dichtungsrahmen

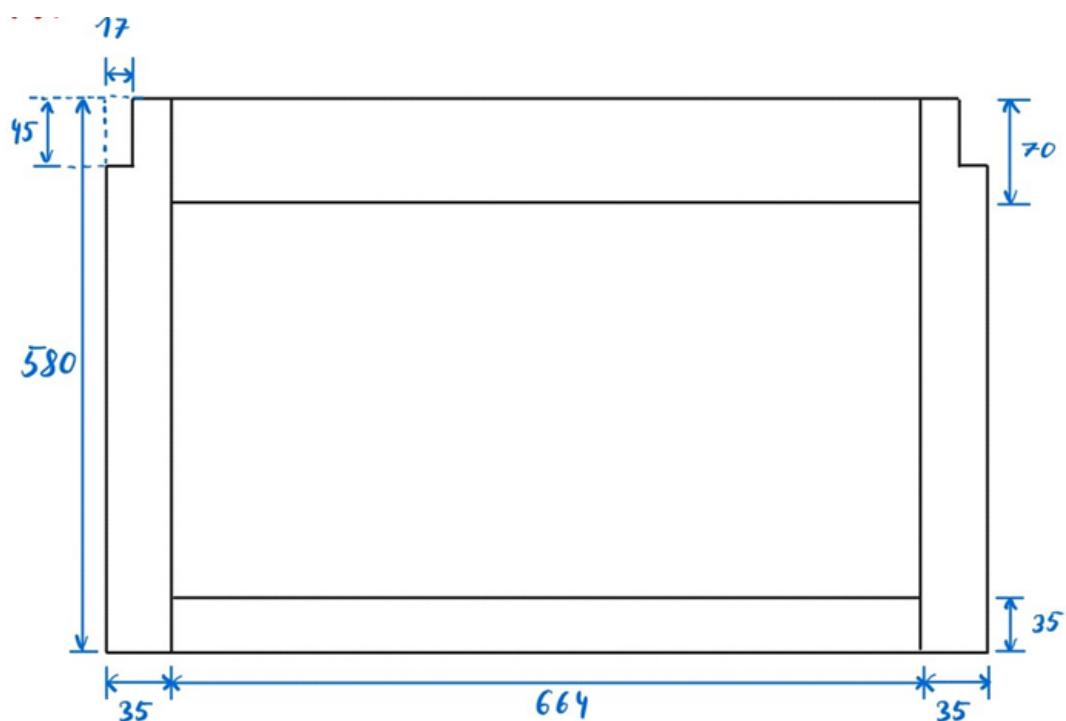


Abbildung 12: Skizze Dichtungsrahmen in mm

Die Flachstangen werden zusammengeschweißt und dann mit einem Tiefschwarz Lackspray lackiert. Das Dichtungsband wird doppelt mit einem Abstand dazwischen auf den Rahmen geklebt.

Das Schaumisolierbar hat eine hervorragende Dicht- und Verformungsbeständigkeit, ist für Temperaturen von -50°C bis 150°C geeignet, hat eine gute Dämpfung sowie eine gute Wärmeisolierung.



Abbildung 13: Bild Rahmen

9.1.5 Schließer

Damit die Kammer fest verschlossen werden kann, wird die Tür mit sogenannten Kniehebelspanner an den Rahmen gepresst. Die Kniehebelspanner können nicht direkt am Gestell montiert werden, da ihr Arm zu lang ist und dadurch die Tür nicht mehr geöffnet werden könnte.

Aus diesem Grund wurde aus einem Reststück Aluprofil ein Adapter gebaut und mit einem 3D Drucker ein Winkel gedruckt. In das Loch vom Aluprofil wurde ein Gewinde geschnitten, denn so kann man den Winkel anschrauben.

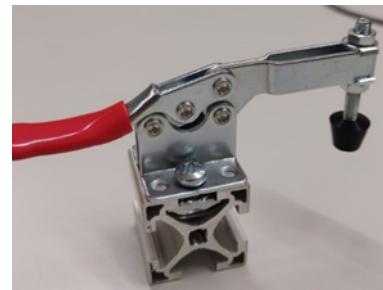
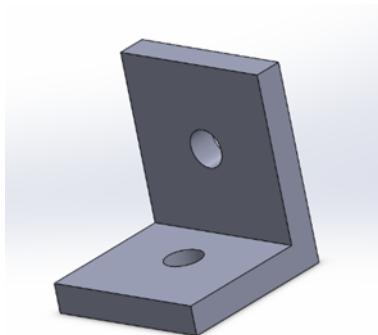
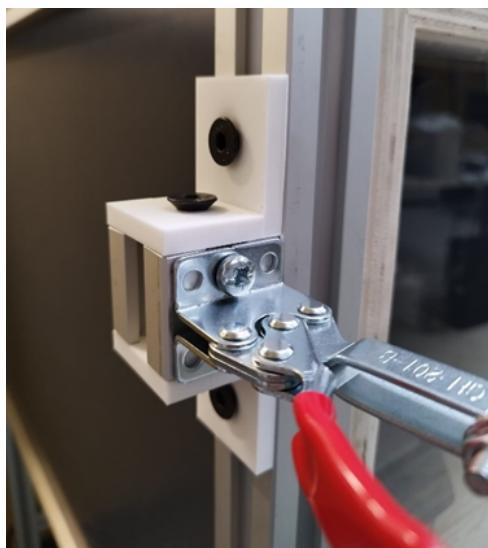


Abbildung 14: Schließer



Damit der Winkel genau auf das Aluprofil passt, hat dieser eine Grundfläche von 4x4cm. Die Rückseite wird an dem Gestell mit einer M8 Schraube fixiert.

Abbildung 15: Winkel



(a) geöffnet



(b) geschlossen

Abbildung 16: Zustand Schließer

9.2 Gyroskop

In der Teststation sollen Tests durchgeführt werden, bei denen sich der Satellit rotiert. Der Satellit sollte sich in alle Richtungen drehen können, denn nur so können die Gyro Sensoren komplett und genau getestet werden. Da sich der Satellit später im All willkürlich rotiert, bringt eine konstante Rotation um nur eine Achse nichts. Aus diesem Grund wird ein sogenanntes Gyroskop oder auch Kreiselinstrument verwendet. Ein Gyroskop kann ein Objekt gleichzeitig in alle Richtungen (x,y,z) drehen. Es besteht aus 3 Ringen, der äußerste Ring wird mit einem Stepper Motor um die X-Achse in Bewegung gesetzt, der Mittlere Ring dreht sich dann mit dem Eigengewicht um die Y-Achse und der innerste Ring ist in unserer Anwendung der Satellit.

Um die Ringe in einer Höhe von 30cm zu halten, braucht es 2 Halterungen. Diese Halterungen werden aus Flachstangen, die eine Stärke von 8mm und einer Breite von 20mm haben, geschweißt. Dazu sind folgende Flachstange notwendig.

Bezeichnung	Stückzahl
Flachstange 5cm	4
Flachstange 20cm	2
Flachstange 25cm	4
Flachstange 10cm	2
Flachstange 30cm	2

Tabelle 6: Stückliste Halterung Gyroskop



Abbildung 17: Halterung Gyroskop

Komplette Zusammenschaltung Gyroskop

Das Relais steuert das Netzteil, welches die Spannung für den Treiber liefert. Der Treiber bekommt Signale vom Raspberry Pi und steuert mit denen die Wicklungen vom Motor.

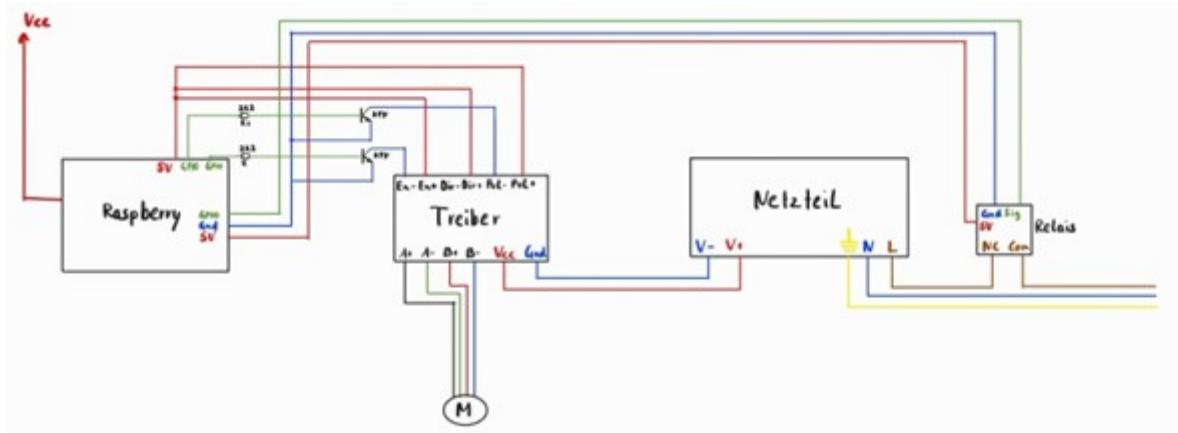


Abbildung 18: Zusammenschaltung Gyroskop

9.2.1 Motor

Als Motor wurde der STEPPERONLINE Nema 17 Schrittmotor²³ gewählt, der speziell für 3D-Drucker und CNC-Reprap-Maschinen konzipiert ist. Mit einem Drehmoment von 59 Ncm, einer Betriebsspannung von 24V, einem Strom von 2A und einem Schrittewinkel von 1.8 Grad pro Schritt gewährleistet er präzise Bewegungssteuerung. Das 4-Draht-Design und das beiliegende 1 Meter lange Kabel mit Verbinder ermöglichen eine einfache Integration in unsere Teststation. Dieser Schrittmotor ist optimal für DIY-Projekte und Anwendungen mit hohen Anforderungen an Präzision und Steuerung von Bewegungen.



Abbildung 19: Schrittmotor

Befestigung am Gyroskop:

Zuerst war geplant, den Ring im Gyroskop über eine Antriebsstange zu betreiben, dazu wäre ein 90° Umlenkung mit Zahnrädern notwendig gewesen. Da die Getriebe Stange über 30cm lang ist und nicht präzise zentriert ist, ist es schwierig, ein leicht gängiger Antrieb zu erhalten. Aus diesem Grund wird der Motor direkt an dem Linker Steher befestigt.



Abbildung 20: Motor am Gyroskop

²³ STEPPERONLINE Schrittmotor Nema 17 Stepper Motor 59Ncm 2A 1.8deg 2 Phase 4-Draht Stepping Motor w/1m Kabel & Verbinder für 3D Drucker/CNC Reprap : Amazon.de: Gewerbe, Industrie & Wissenschaft.

5 Volt Steuerung:

Die GPIO-Pins vom Raspberry Pi haben eine Ausgangsspannung von 3.3V. Für den Motor Treiber TB6600 wird aber eine Signalspannung von 5V benötigt. Mit einem NPN-Transistor und einem 2k2 Widerstand wird eine Ausgangsspannung von 5V erzeugt. Diese Zusammenschaltung wird auch Emitter Schaltung bezeichnet. Die Basis des bipolaren NPN-Transistors wird über einen Vorwiderstand mit dem Ausgangssignal eines GPIO-Pins vom Raspberry verbunden, während der Emitter mit dem 5V Ausgang vom Raspberry verbunden ist. Am Emitter ist zu gleich das Ausgangssignal mit nun 5V. Dieser wird mit dem Eingang vom Treiber verbunden.

9.2.2 Motor-Treiber

Da man den Motor nicht direkt mit einem Raspberry steuern kann, braucht es einen Motor treiber. Der Twotrees DM542 5.6A Schrittmotortreiber²⁴ ist ein leistungsstarker Treiber für 2-Phasen-Schrittmotoren, wie unser Motor, der verwendet wird. Hier ist eine kurze technische Produktbeschreibung:

Eigenschaften:

- Leistung: Der Treiber unterstützt Motoren mit bis zu 5.6A Stromversorgung.
- Betriebsspannung: Geeignet für Gleichstromquellen im Bereich von 18-48V.
- Schrittauflösung: Präzise Mikroschrittsteuerung für genaue Positionierung.
- Phasen: Entwickelt für 2-Phasen-Schrittmotoren.
- Peak-Strom: Bietet einen Spitzenstrom von bis zu 4.2A für verbesserte Motorleistung.

Schutz und Zuverlässigkeit:

- Überstromschutz: Der Treiber ist mit einem Überstromschutz ausgestattet, der den Motor vor Schäden durch zu hohe Ströme schützt.
- Wärmeschutz: Integrierter Schutzmechanismus gegen Überhitzung für eine zuverlässige Langzeitnutzung. So können wir die Tests lang genug durchführen.

Der Twotrees DM542 5.6A Schrittmotortreiber bietet eine leistungsstarke und zuverlässige Lösung für die Steuerung von 2-Phasen-Schrittmotoren in anspruchsvollen Anwendungen. Mit Funktionen wie Überstromschutz, Wärmeschutz und Mikroschrittsteuerung ist er eine geeignete Wahl für unsere Gyroskop Ansteuerung.

²⁴ Twotrees DM542 5.6A Schrittmotortreiber Stepper Treiber, Dm542 2 Phasen Schrittmotortreiber Dc 18-48v 57/86 Stepper Motor Driver Peak 4.2a : Amazon.de: Gewerbe, Industrie & Wissenschaft.

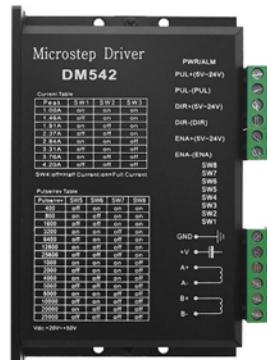


Abbildung 21: Motor-Treiber

9.2.3 Netzteil

Der Motortreiber muss man mit mindestens 9V DC versorgen. Diese Spannung wird mit einem Netzteil erzeugt. Das MeanWell²⁵ LRS-100-24 ist ein industrielles Netzteil mit 108W Leistung, einer Ausgangsspannung von 24V und 4,5A Ausgangsstrom. Es ist für den zuverlässigen Einsatz in industriellen Anwendungen wie Steuerungen und LED-Beleuchtungen konzipiert. Mit integriertem Überlast- und Kurzschlusschutz bietet es eine effiziente und sichere Stromversorgungslösung. Seine robuste Bauweise und Kompaktheit machen es ideal für anspruchsvolle Umgebungen mit begrenztem Platz.



Abbildung 22: Netzteil

²⁵ MeanWell LRS-100-24 108W 24V 4,5A Industrielles Netzteil : Amazon.de: Computer & Zubehör.

9.2.4 Relais

Mit einem Relais kann man mit kleinen Spannungen große Spannungen schalten. Ein Relais²⁷ besteht aus einer Spule und einem beweglichen Kontakt. Wenn Strom durch die Spule fließt, erzeugt sie ein Magnetfeld, das den Kontakt beeinflusst. Dadurch ändert sich der Zustand des Schaltkontakte, und ein elektrischer Pfad wird geöffnet oder geschlossen. Relais dienen dazu, mit einem schwachen Steuersignal einen starken elektrischen Stromkreis zu steuern oder zu schalten. Relais werden in unserer DA verwendet, um mit dem Raspberry Netzspannungen zu schalten. Für unsere Anwendung brauchen wir das GRV RELAY SPDT30, dieses ist ein hochwertiges einpoliges Zweiwege-Relais mit hoher



Abbildung 23: Relais²⁶

Schaltleistung. Es ist für Betriebsspannungen von 4,75 bis 5,25 V und Schaltströme von bis zu 30 A bei 250 V AC oder 30 V DC ausgelegt. Das Relais ist kompatibel mit den Schnittstellen vom Raspberry Pi, verfügt über eine schnelle Einschaltzeit von maximal 15ms und eine Betriebstemperatur von -25 bis +75°C. Eingesetzt wird es in der Teststation für die Steuerung des Netzteils und für die UV Lampe.

²⁷ Team (webmaster@reichelt.de), GRV RELAY SPDT30 - Arduino - Relais SPDT, 30 A, SLA-05VDC-SLC.

9.3 Leiterplatte

In dieser Anwendung ist die Leiterplatte ein Raspberry Shield. Ein Shield sorgt dafür das alle Komponenten, die angesteuert werden sollen, übersichtlich an den Raspberry Pi angeschlossen werden. Es kann aber auch sein, dass das Shield eine Erweiterung für den Raspberry Pi ist. In diesem Fall wurde das Shield verwendet, um die Komponenten mit dem Raspberry Pi zu verbinden.

PCB und der Schaltplan wurden in Eagle 7.0.0 entworfen.

9.3.1 Schaltplan

Das Shield wird über den PRT-14017²⁸ Header mit dem Raspberry Pi verbunden

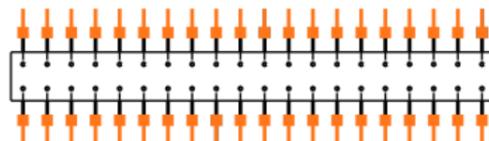


Abbildung 24: Schaltplansymbol PRT-14017

Um die verschiedenen Komponenten anzuschließen, werden Schraubklemmen verwendet. Es werden fünf 4-Polige Schraubklemmen²⁹ und elf 3-Polige Schraubklemmen³⁰ benötigt.

Nachdem die Schraubklemmen und der Header platziert wurden, werden die Pole der Schraubklemmen mit dem Header verbunden. Die Zuordnung ist in der Tabelle 7 zu sehen.

²⁸ PRT-14017 footprint & symbol by SparkFun Electronics | SnapMagic Search.

²⁹ Team (webmaster@reichelt.de), CTB0502-4 - Lötbare Schraubklemme - 4-pol, RM 5 mm, 90°.

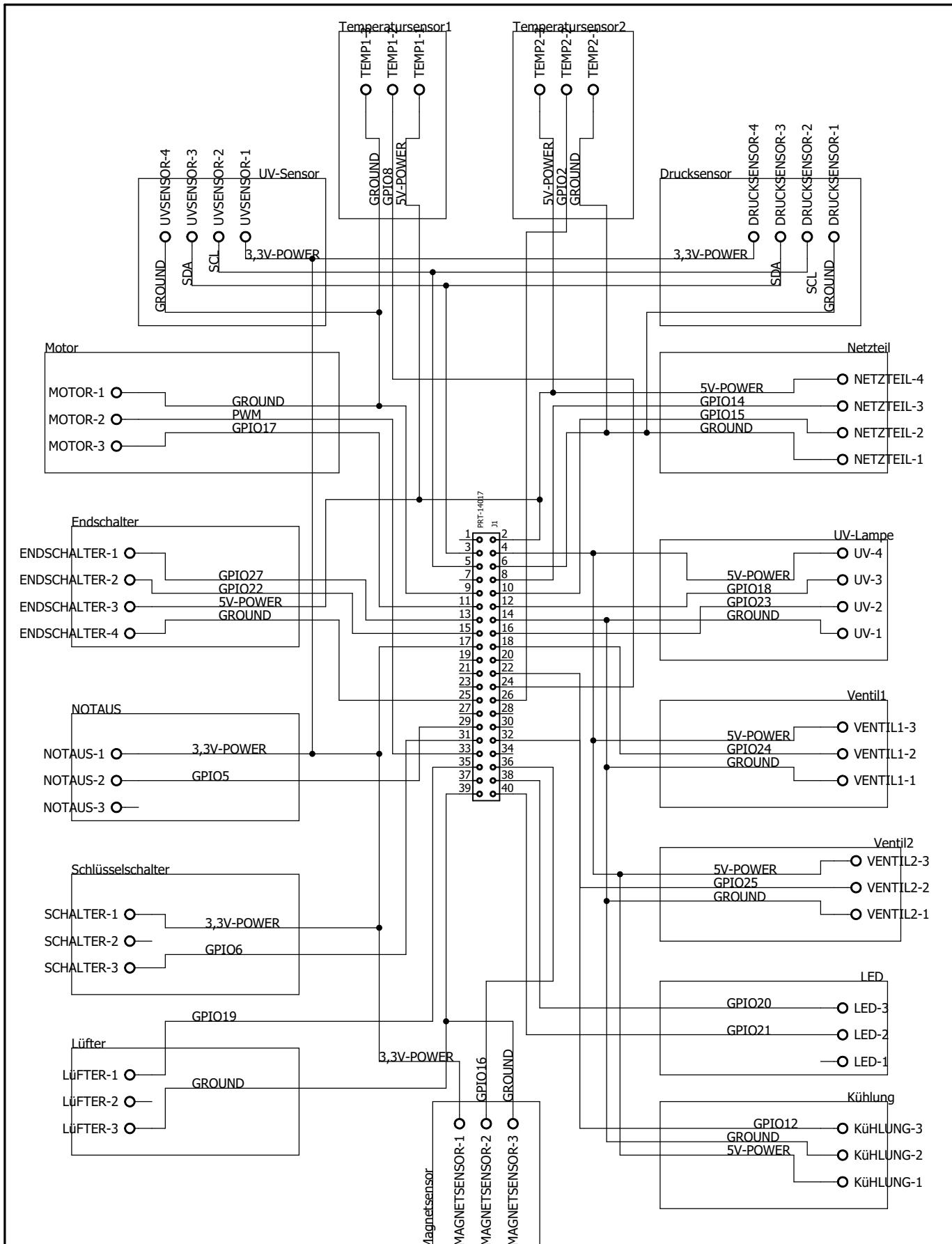
³⁰ 237-103.

Bezeichnung	GPIO-Pin	GPIO-Pin	Versorgung
Lüfter	GPIO19	-	-
Kühlung	GPIO12	-	5V
LED-Streifen	GPIO20	GPIO21	-
Schlüsselschalter	GPIO06	-	3.3V
Ventil2	GPIO25	-	5V
Notaus	GPIO05	-	3.3V
Ventil1	GPIO24	-	5V
UV-Lampe	GPIO18	GPIO23	5V
Endschalter	GPIO27	GPIO22	5V
Motor	GPIO17	GPIO13 PWM	-
Netzteil	GPIO14	GPIO15	5V
Temperatursensor1	GPIO07	-	5V
Temperatursensor2	GPIO08	-	5V
Magnetsensor	GPIO16	-	3.3V
Drucksensor	GPIO03 SDA	GPIO05 SCL	5V
UV-Sensor	GPIO03 SDA	GPIO05 SCL	5V

Tabelle 7: Pinbelegung Raspberry Pi Shield

Der Drucksensor und der UV-Sensor sind beide über die I2C-Schnittstelle mit dem Raspberry Shield verbunden.

Information über die I2C-Schnittstelle sind unter [ti.com/lit/an/slva704/slva704.pdf?ts=1712280198725&ref_url=https%253A%252F%252Fwww.google.com%252F](https://www.google.com/search?q=ti.com/lit/an/slva704/slva704.pdf?ts=1712280198725&ref_url=https%253A%252F%252Fwww.google.com%252F) zu finden.



 HTL Rankweil	TITLE: RaspyBoard		Teststation TU Wien
	A4	Schaltplan	
Name:	Sophia Hagen	REV:	
Date:	16.02.2024 09:26	Sheet:	1/1

9.3.2 PCB

Die Platine hat eine Größe von $95mm \times 65mm$. Geplant war, das Board in der Größe des Raspberry Pi zu entwerfen. Der Raspberry Pi 4B besitzt 2 USB-Buchsen und einen LAN-Anschluss. Diese sind jedoch zu hoch, um darüber eine Platine zu platzieren die mit dem Raspberry Pi verbunden werden soll. Aus diesem Grund wurde das Board in die Länge gezogen. Die 16 Schraubklemmen werden einzeln beschriftet. Beispielsweise hat die Schraubklemme für den UV-Sensor die Beschriftung UV-Sensor. Zur Befestigung werden 4 Bohrungen benötigt. Diese haben einen Durchmesser von $3.5mm$.

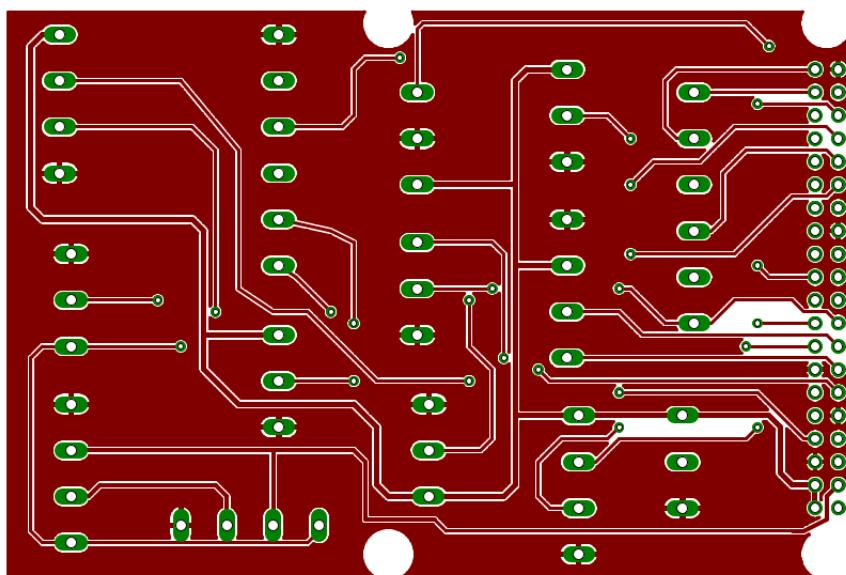


Abbildung 25: Layout Top

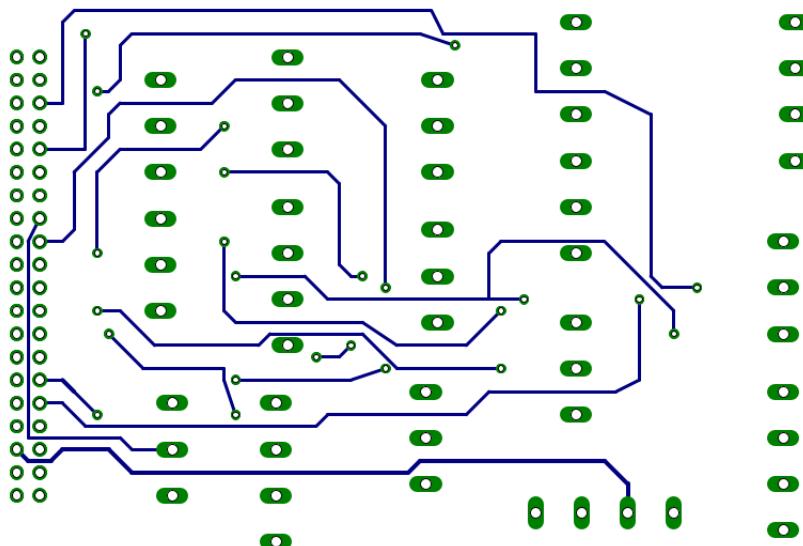


Abbildung 26: Layout Bottom

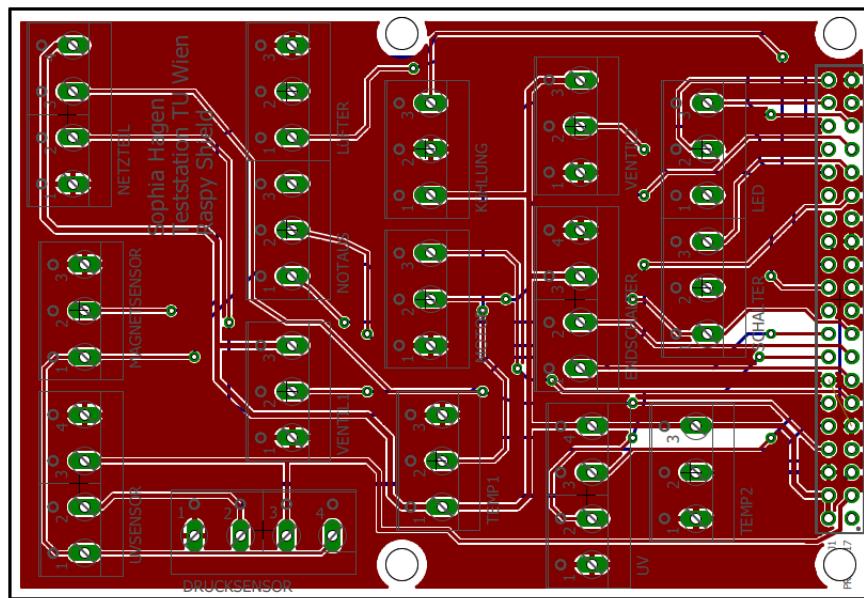


Abbildung 27: PCB mit Polygon

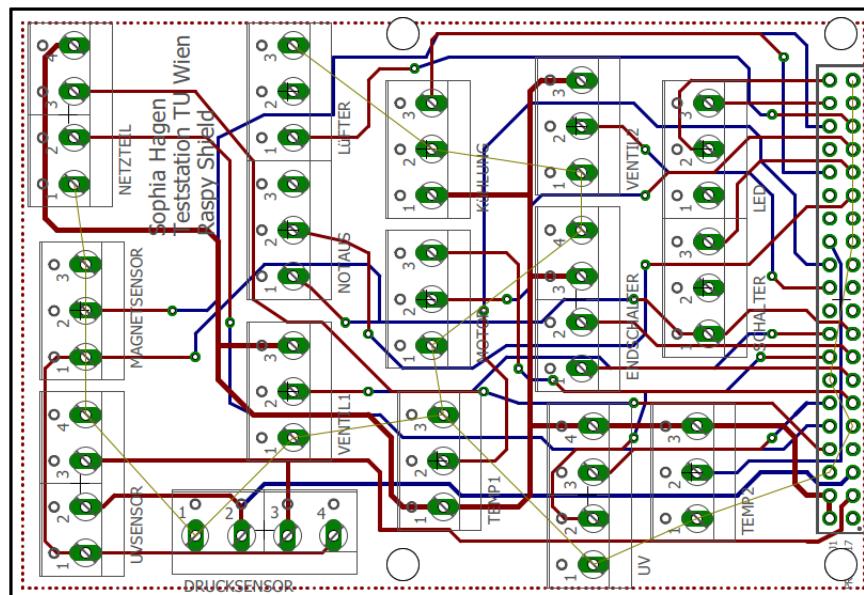


Abbildung 28: PCB ohne Polygon

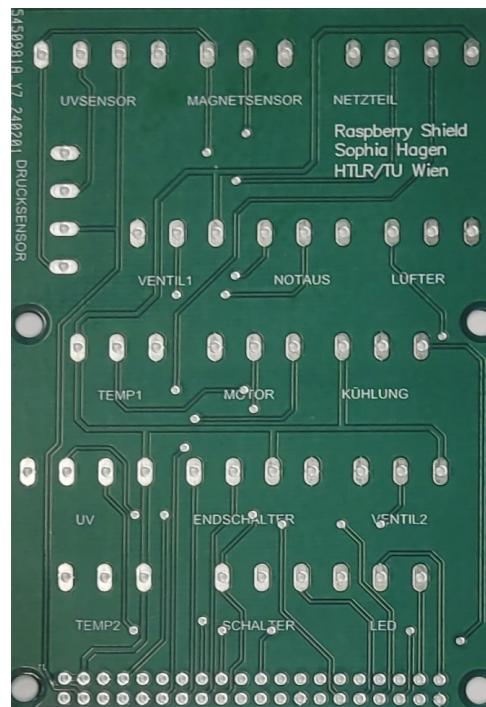


Abbildung 29: Unbestückte Platine

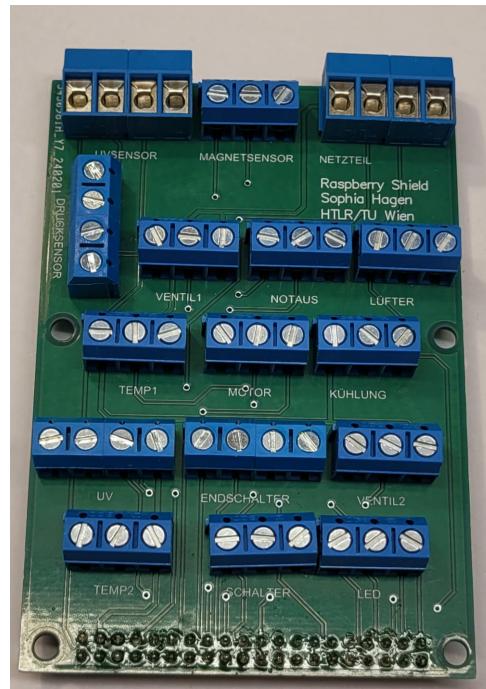


Abbildung 30: Bestückte Platine

9.4 Sensoren in der Testkammer

Die verschiedenen Sensoren werden in der Testkammer platziert, damit einen Vergleich zwischen den Sensorwerten in der Kammer und auf dem CubeSat dargestellt werden kann. Dieser Vergleicht ist eine Überprüfung, ob alle Sensoren auf dem CubeSat noch funktionieren und richtige Werte ausgeben. Bei den verwendeten Sensoren musste darauf geachtet werden, dass sie bei einer Temperatur von 0°C bis 30°C betrieben werden können.

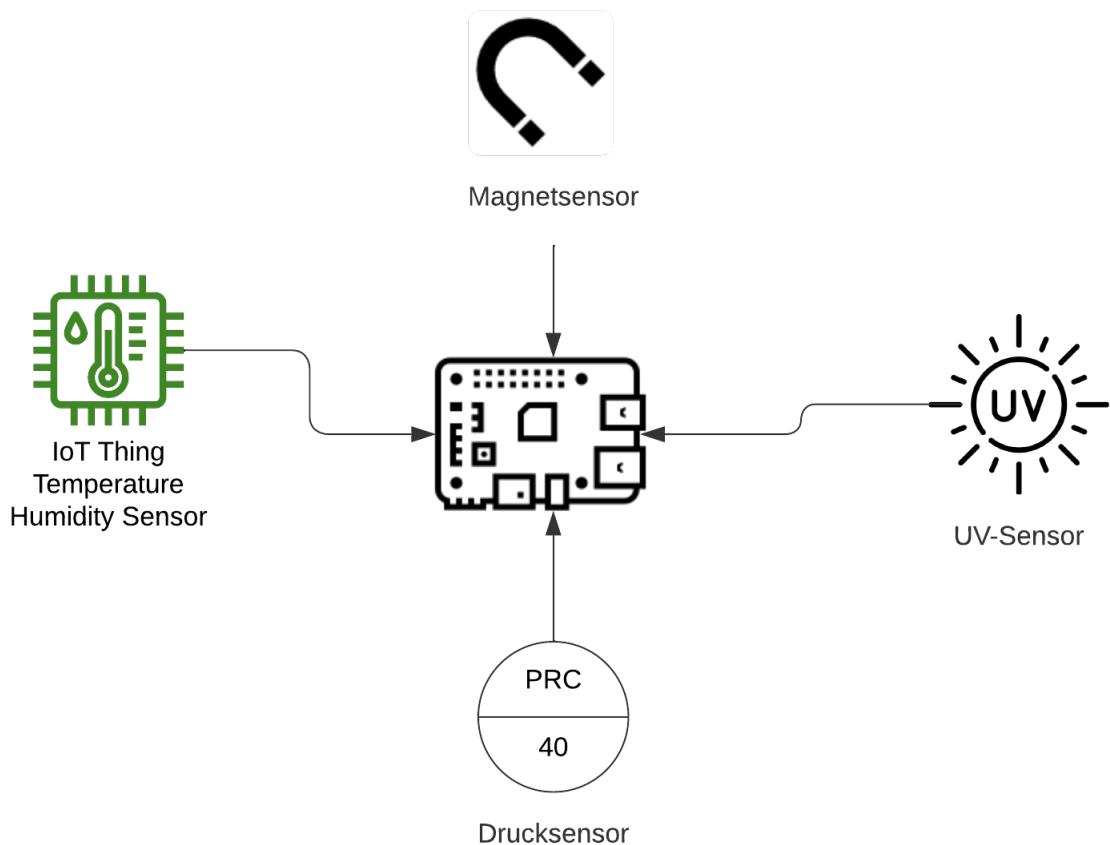


Abbildung 31: Blockschaltbild Sensoren

9.4.1 Temperatursensor

In der Testkammer befinden sich zwei Temperatursensoren mit integriertem Feuchtigkeits-sensoren. Diese messen die Temperatur sowie die Feuchtigkeit in der Kammer. Dadurch können die Sensorwerte der Kammer und des CubeSat verglichen werden.

Für diese Anwendung wird der Temperatursensor **AM2302**³¹ verwendet. Der Anschluss an den Raspberry Pi erfolgt durch die PCB. Der Sensor wird mit 3 Pins an die PCB angeschlos-sen.

VDD	5 Volt
GND	Ground
Data	GPIO7 und GPIO8

Tabelle 8: Pinbelegung Temperatursensor

Um beide Sensoren in der Testkammer zu montieren, wurde ein Gehäuse in SolidWorks ent-worfen. Danach wurden die Teile mit dem 3D-Drucker ausgedruckt.

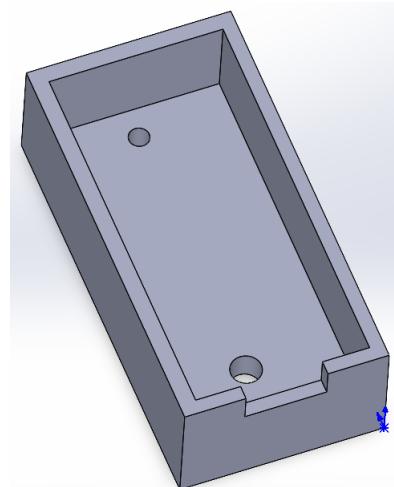


Abbildung 32: Gehäuse Temperatursensor

³¹ Industries, AM2302 (*wired DHT22*) temperature-humidity sensor.

Um den Temperatursensor zu verwenden, wird eine Bibliothek benötigt. Die Bibliothek wird von Adafruit bereitgestellt. Um diese zu installieren wird folgender Befehl auf dem Raspberry Pi ausgeführt:

```
pip3 install adafruit-circuitpython-dht
```

Durch die Bibliothek³² wird das Auslesen des Sensors vereinfacht. Mit wenigen Befehlen kann der Sensor die Temperatur und die Feuchtigkeit messen.

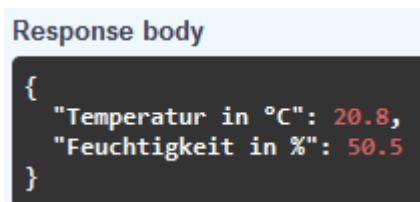


Abbildung 33: Temperatursensorausgabe

Die Messung mit dem Temperatursensor wurden in einem normalen Zimmer mit Raumtemperatur durchgeführt. Die ermittelten Daten für Temperatur und Feuchtigkeit werden in der oberen Abbildung abgebildet.

Die Programmierung für das Testprogramm befindet sich im Kapitel 10.2.3, und der Programm Abschnitt für die API im Kapitel 11.4.2

9.4.2 UV-Sensor

Der CubeSat besitzt einen UV-Sensor, aus diesem Grund wurde auch in die Testkammer einer eingebaut. Es wurde der UV-Sensor LTR390³³ verwendet. Dieser verfügt über einen integrierten ADC, somit muss kein Analog-Digital-Wandler dazwischengeschaltet werden und kann direkt über I2C an den Raspberry Pi angeschlossen werden.

VCC	5 Volt
GND	Ground
SDA	GPIO3
SCL	GPIO5
INT	Interruptausgang

Tabelle 9: Pinbelegung UV-Sensor

³² [adafruit/Adafruit_CircuitPython_DHT](#).

³³ Adafruit LTR390 UV Light Sensor 300 to 350nm STEMMA QT Qwiic 4831 - E, 6,95 €.

Das Licht der UV-Lampe9.6 hat eine Wellenlänge von 385nm bis 400nm. Daher muss der UV-Sensor auch für diesen Bereich ausgelegt sein. Der Messbereich des LTR390 ist zwischen 200nm und 400nm. Der Sensor misst das Umgebungslicht und den UV-Index. Das Umgebungslicht wird in Lux angegeben. Ein niedriger Lux-Wert bedeutet das der Sensor sich in einer dunklen Umgebung befindet. Ein hoher Wert bedeutet eine helle Umgebung. Der UV-Index wird in verschiedene Kategorien eingeteilt.



Abbildung 34: UV-Sensor

UV-Index	Kategorie
0-2	Niedrig
3-5	Mäßig
6-7	Hoch
8-10	sehr Hoch
11+	Extrem

Tabelle 10: UV-Index

Response body

```
[  
  "UV-Index:",  
  2.379130434782609,  
  "Umgebungslicht in Lux:",  
  65535  
]
```

(a) indirektes Sonnenlicht

Response body

```
[  
  "UV-Index:",  
  0,  
  "Umgebungslicht in Lux:",  
  89  
]
```

(b) kein Sonnenlicht

Abbildung 35: Messungen mit UV-Sensor

Die Messungen der oberen zwei Abbildungen wurden in einem Zimmer durchgeführt. Der Sensor in Abbildung (a) wurde für die Messung dem Sonnenlicht durch ein Fenster ausgesetzt. Aus diesem Grund ist der UV-Index eher niedrig. Der LTR390 kann Umgebungslicht von 0 Lux bis zu 65535 Lux messen. In Abbildung (a) ist zu erkennen, dass der maximale Wert bei der Messung des Umgebungslichtes erreicht wurde. Daraus kann geschlussfolgert werden, dass sich der Sensor in einer hellen Umgebung befunden hat. In Abbildung (b) wurde der Sensor vom Sonnenlicht entfernt, somit haben keine UV-Strahlen den Sensor getroffen.

Dies kann auch am gemessenen UV-Index erkannt werden. Es wurde ein Umgebungslicht von 89 Lux gemessen. Der UV-Sensor war also in einer eher dunklen Umgebung.

Für den LTR390 UV-Sensor wird die folgende Bibliothek³⁴ verwendet:

```
pip install adafruit-circuitpython-ltr390
```

Das Testprogramm für den UV-Sensor ist im Kapitel 10.2.4 und der Codeabschnitt für die API im Kapitel 11.4.3

9.4.3 Magnetsensor

Der Magnetsensor misst das Magnetfeld in seiner Umgebung. Der KY-024 Linearer, magnetischer Hall-Sensor,³⁵ ist für den Dauerbetrieb geeignet und ist in einem Temperaturintervall von -150°C bis 150°C stabil.

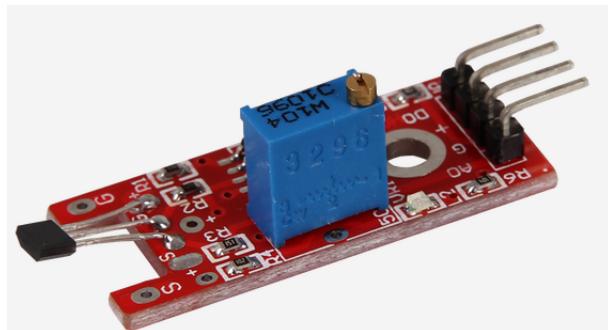


Abbildung 36: KY-024 Magnetsensor

Der Sensor besitzt zwei LEDs.

- LED 1: zeigt an, ob der Sensor mit Strom versorgt wird.
- LED 2: zeigt an, ob der Sensor ein Magnetfeld erkannt hat.

Durch ein Drehpotentiometer kann die Empfindlichkeit des Sensors eingestellt werden. Über den digitalen Ausgang des Sensors wird ein Signal ausgegeben, sobald ein Magnetfeld erkannt wird.

³⁴ Adafruit LTR390 UV Light Sensor 300 to 350nm STEMMA QT Qwiic 4831 - E, 6,95 €.

³⁵ ARD SEN HALL3: Arduino - Hall-Magnet-Sensor, linear bei reichelt kaufen.

Der KY-024 ist ein analoger Sensor. Da der Raspberry Pi keinen integrierten ADC besitzt, muss einer gekauft werden, damit der Sensor über den Raspberry Pi angesteuert und gelesen werden kann. Als ADC eignet sich hierfür der 16-Bit ADC KY-053.³⁶

Zusammenschaltung von Raspberry Pi , Magnetsensor und ADC

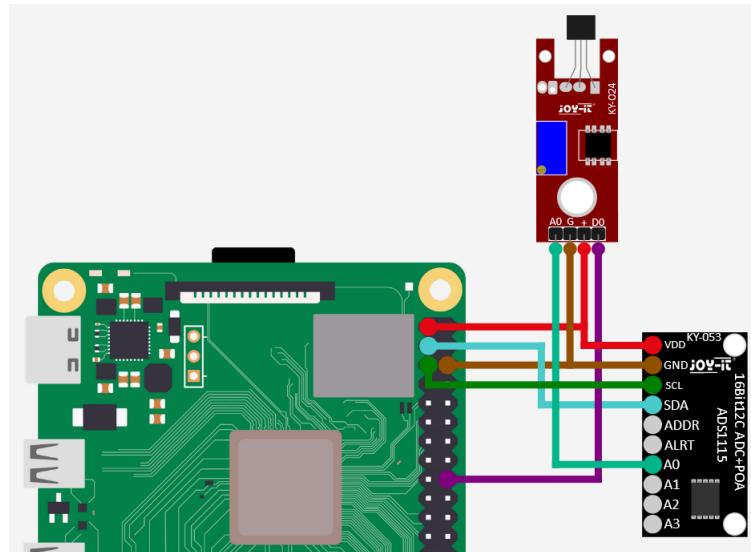


Abbildung 37: Zusammenschaltung Magnetsensor³⁷

9.4.4 Drucksensor

In der Teststation gibt es eine Vorrichtung, um ein Vakuum zu erzeugen. Durch dieses Vakuum kann der integrierte barometrischer Drucksensor auf dem CubeSat getestet werden. Der Drucksensor in der Kammer wird dazu verwendet, einen Vergleich zwischen Testkammer und Satellit herzustellen.

Es wurde der Drucksensor BMP180³⁸ verwendet. Er befindet sich in der Testkammer, hat jedoch keinen fixen Platz. Der Sensor kann je nachdem, ob ein Vakuum erzeugt wird oder nicht, platziert werden. Der BMP180 hat 4 Pins die über die PCB angeschlossen und mit dem Raspberry Pi verbunden werden.

Um den Sensor verwenden zu können, muss auch hier eine Bibliothek heruntergeladen werden. Diese wird von Adafruit bereitgestellt. Mit folgendem Befehl kann die Bibliothek³⁹ heruntergeladen werden.

³⁶ RPI ADC: Raspberry Pi - Analog/Digital Konverter bei reichelt kaufen.

³⁸ AZDelivery GY-68 BMP180 Barometrischer Luftdruck und Temperatur Sensor kompatibel mit Arduino und Raspberry Pi inklusive E-Book! : Amazon.de: Gewerbe, Industrie & Wissenschaft.

³⁹ circuitpython-bmp180.

Vin	5 Volt
GND	Ground
SDA	GPIO3
SCL	GPIO5

Tabelle 11: Pinbelegung Drucksensor

```
pip install circuitpython-bmp180
```

Die Suche nach einer geeigneten Bibliothek war nicht einfach, da die meisten Repository archiviert wurden. In einigen Dokumenten wird auch angegeben, dass der Drucksensor BMP180 gar nicht mehr hergestellt wird.

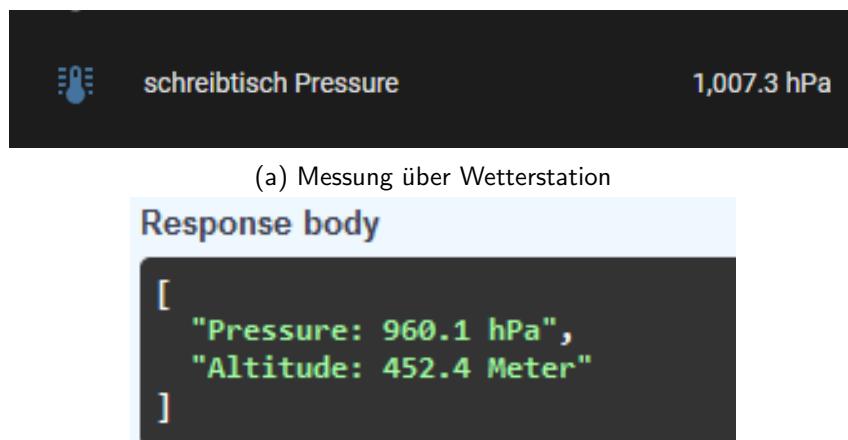


Abbildung 38: Druckmessungen

Die oben angeführten Druckmessungen, wurden beide in Wolfurt durchgeführt. Der Druck in Abbildung (a), wurde mit einer Netatmo Wetterstation⁴⁰ gemessen. In Abbildung (b), wurde der BMP180 Drucksensor verwendet, um den Druck zu ermitteln. Die Werte stimmen einigermaßen überein. Auch die Meereshöhe kann mit dem BMP180 angezeigt werden. Wolfurt liegt auf einer Meereshöhe von 434 m. Der Sensor zeigt eine Meereshöhe von 452 Metern an. Da dieser Wert nicht genau mit dem tatsächlichen Wert übereinstimmt, kann eine Kalibrierung durchgeführt werden. Dies erfolgt durch eine Anpassung an der Berechnung im Code.

Der Codeabschnitt für das Testprogramm befindet sich im Kapitel 10.2.5. Das Programm für die API ist im Kapitel 11.4.4.

⁴⁰ Smart Home Weather Station.

9.5 Sensoren auf dem Satellit

9.5.1 EDU

Die Programmierung des EDU (Educational Module), welches sich im Cubesat der TU-Wien befindet, dient als eine Plattform, welche für Schülerinnen und Schüler zur Verfügung steht, um über ihre Software verschiedene Experimente durchzuführen. Das Modul besteht aus einem Raspberry PI, verschiedenen Sensoren und zwei Kameras (siehe Abb.), welche beide genutzt werden können. Die Ansteuerung der Sensoren funktioniert über den I2C-Bus.

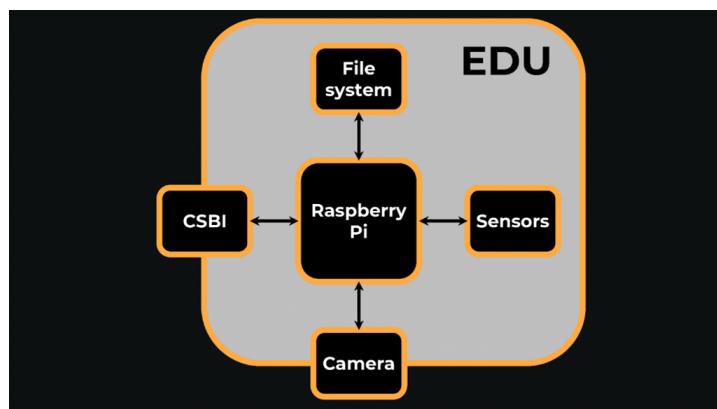


Abbildung 39: Blobschaltbild des EDUs

Das System kann von jeder Person zuhause getestet werden. Dazu dient der EDU-Hat, eine Platine, welche die Sensorik des EDUs enthält und sich einfach auf einen Raspberry PI stecken lässt.

9.5.2 Sensorik des EDUs

Auf dem EDU sind folgende Sensoren zu finden:

- Temperatursensor
- Magnetfeldsenso
- Beschleunigungssensor
- Gyroskop
- GNSS-Empfänger
- UV-Sensoren
- Dosimeter

Der verwendete EDU-HAT (Simulation des EDUs für Selbstexperimente auf dem Boden) besitzt jedoch nur einen Temperatursensor, einen Beschleunigungssensor, einen Magnetfeldsensor, einen UV-Sensor und einen Gas Sensor. Die Montage einer Raspberry PI Kamera ist jedoch auch möglich.

9.5.3 Temperatursensor

Sowohl auf dem EDU-Hat als auch auf dem EDU selbst befinden sich 2 Sensoren, welche Temperaturen messen können.

Der Sensor TMP112⁴¹ ist ein rein digitaler Temperatursensor, welcher im Bereich von -40°C bis 125°C akkurat ($\pm 0,5^\circ\text{C}$ bei 0° bis 65°C und $\pm 1^\circ\text{C}$ bei -40 bis 125°C) Temperaturen messen kann. Die Betriebsspannung des Sensors liegt bei 1.4V-3.3V bei einem maximalen Strom von 10 μA .

Der Gas Sensor BME688⁴² dient als Feuchtigkeitssensor, Gassensor und als guter Temperatursensor. Auch ist die Messung des Luftdrucks möglich. Die Betriebsspannung des Sensors liegt zwischen 1.71V-3.6V, während der Sensor je nach Betriebsmodus zwischen 2,1 μA und bis zu 0,9mA Strom verbraucht. Der Arbeitsbereich des Sensors liegt zwischen -40°C bis 85°C für die Temperatur, 0-100°C für Luftfeuchtigkeit und Gaserkennung und 300-1100 hPa für den Luftdruck.

⁴¹ TMP112-Q1 Datenblatt, Produktinformationen und Support | TI.com.

⁴² BME 688: KI-Kombo-Sensor, Luftdruck/Luftfeuchtigkeit/Temp./Gas bei reichelt kaufen.

9.5.4 Magnetfeldsensor/Gyroskop

Der BMM150⁴³ ist ein dreiachsiger geomagnetischer Sensor, welcher sich für die Messung von Magnetfeldern und deren Einwirkungen eignet. Der Arbeitsbereich des Sensors liegt zwischen -40°C bis +85°C bei einer Betriebsspannung von 1.62V-3.6V. Auch der Stromverbrauch ist bei 170µA minimal. Die Rückgabe des Sensors besteht aus X, Y und Z Werten, welche die Stärke des Magnetfeldes in den verschiedenen Richtungen darstellen.

9.5.5 Beschleunigungssensor

Der Beschleunigungssensor ADXL345⁴⁴ dient als dynamischer Sensor, indem er Bewegung und Schock in G-Kräften zurückgibt. Der wichtigere Messbereich des ADXL345 Sensors ist die statische Messung, welche die Gravitation auf den verschiedenen Achsen zurückgibt. Aus dieser Information heraus kann der Winkel des Sensors ausgerechnet werden und somit als Gyroskop Sensor verwendet werden.

Der Stromverbrauch des Sensors liegt bei 40µA bei einer Betriebsspannung von 2.0V bis 3.6V. Der Sensor hat eine Auflösung von 13-Bit und misst in den Bereichen von ±16g, was umgerechnet rund ±156,91 m/s² entspricht.

Die Rückgabewerte des Beschleunigungsmessers entsprechen den G-Kräften an den X, Y- und Z-Achsen.

9.5.6 UV-Sensor

Der UV-Sensor GUVA_C32 dient zur Messung von UV-Strahlen welche sich im Orbit, bzw. auf dem Boden befinden. UV-Strahlungen werden in Stärkegraden⁴⁵ als UV-Index (1-14 UV) angegeben. Die Betriebsspannung des Sensors liegt zwischen 2.6V bis 3.6V bei einem Stromverbrauch von 100µA.

9.5.7 Dosimeter

Der Dosimeter ist einer der Sensoren, welche auf der EDU-Hat nicht vorhanden sind. Seine Aufgabe ist es ionisierte Strahlung zu empfangen. Strahlungen dieser Art sind hauptsächlich in höheren Elevationen verbreitet (siehe Abb. 40).

⁴³ BMM150 | Bosch Sensortec Bewegungssensormodul 3-Achsen SMD I2C / SPI Digital | RS.

⁴⁴ ADXL345 - Acceleration Sensor.

⁴⁵ Cosmic Rays Exposure.

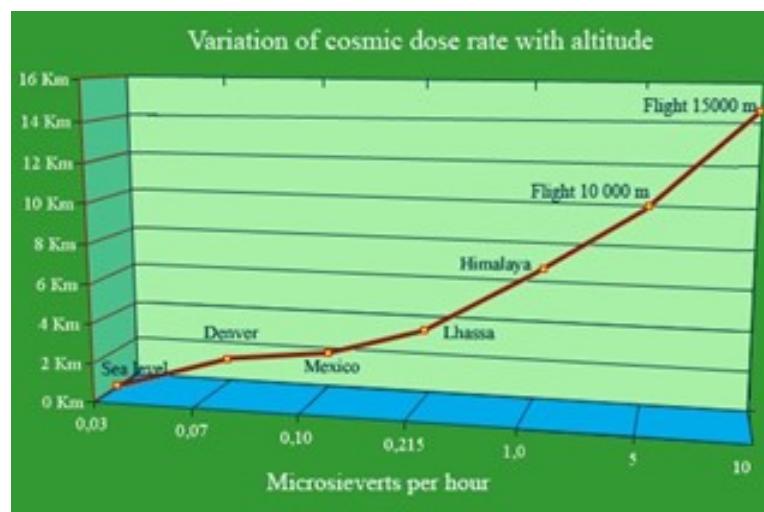


Abbildung 40: Korrelation Dosimeter

Abbildung 40: Zu sehen ist die Korrelation zwischen der Elevation und des Mikrosieverts pro Stunde. Mikrosieverts beschreibt hierbei das stochastische Gesundheitsrisiko, welches durch ionisierte Strahlung verursacht wird.

9.6 UV-Lampe

Da der Satellit später näher an der Sonne ist und somit höherer UV-Strahlung ausgesetzt ist, wird sich in der Testkammer eine UV-Lampe befinden. Mit dieser UV-Lampe sollen die Effekte der ultravioletten Strahlung auf den Satelliten getestet werden.

Es gibt verschiedene Arten von UV-Strahlung:

- UV-A-Strahlung (Wellenlänge von 320 bis 400 nm)
- UV-B-Strahlung (Wellenlänge von 280 bis 320 nm)
- UV-C-Strahlung (Wellenlänge von 100 bis 280 nm)

UV-A-Strahlung ist der größte Teil der UV-Strahlung, die die Erde erreicht. Sie kann zu Hautalterung und Hautkrebs führen. Die UV-B-Strahlung wird fast vollständig durch die Ozonschicht absorbiert. Wenn diese Strahlen doch auf die Erdoberfläche treffen, kann es zu Sonnenbrand und Hautkrebs führen. Die UV-C-Strahlung wird vollständig absorbiert und trifft nicht auf die Erdoberfläche. Sie kann künstlich mit Hilfe einer UV-Lampe erzeugt werden. Diese Strahlung wird verwendet zur Desinfektion und Sterilisation.

Die UV-Lampe⁴⁶ in der Testkammer hat eine Wellenlänge von 385 bis 400nm. Diese Art der UV-Strahlung zählt zu der UV-A Klasse und ist am wenigsten schädlich für Menschen. Da der Satellit später in 500 km Höhe sein wird, wäre eventuell eine UV-Lampe der Klasse UV-C besser geeignet. Jedoch ist die Arbeit mit UV-C-Strahlung nicht ungefährlich und kann zu Verbrennungen am Auge und auf der Haut führen.



Abbildung 41: UV-Lampe⁴⁷

⁴⁶ MEEKBOS UV Schwarzlicht Strahler,60W LED Schwarzlichtlampe,385-400NM IP66 Wasserdicht UV Flutlicht mit Stecker,UV Fluter für Black light Party,GLOW,Fluoreszierende Malei,Körperbemalung,Halloween : Amazon.de: Beleuchtung .

Die UV-Lampe wird nicht direkt vom Raspberry Pi gesteuert. Zwischen dem Raspberry Pi und der Lampe ist ein Relais. Dies ermöglicht es, mit einer Spannung von 5V vom Raspberry Pi die 230V Netzspannung zu schalten.

VDD	5 Volt
GND	Ground
Datenleitung (NC)	GPIO18
Signal (SIG)	GPIO23

Tabelle 12: Pinbelegung UV-Lampe

Die Programmierung für das Testprogramm befindet sich im Kapitel 10.2.6, und der Programm Abschnitt für die API im Kapitel 11.4.5

9.7 Magnetfeld

Der CubeSat befindet sich später in einer Umlaufbahn der Magnetosphäre, deshalb soll der CubeSat in der Testkammer einem Magnetfeld ausgesetzt werden.

Die Magnetosphäre ist eine Region die die Erde umschließt, um die Oberfläche von Sonnenwinden zu schützen. Diese Region besitzt ein Magnetfeld. Das Magnetfeld der Magnetosphäre wird durch den Erdkern erzeugt. Die Grenze für die Magnetosphäre ist unbekannt, da diese Region weit ins Weltall reicht.

Es wurden Neodym Dauermagnete⁴⁸ eingesetzt, um das Magnetfeld zu erzeugen.

Ein Neodym Magnete erzeugt ein Magnetfeld im Bereich von einigen Hundert bis mehreren Tausend Gauss. Das Magnetfeld der Erde ist sehr schwach. Es besitzt eine Größe von 0,7 Gauss.

Diese Magnete werden in der Kammer befestigt. Die ursprüngliche Idee war, die Magnete am Gyroskop zu befestigen. Da diese Magneten jedoch ein sehr starkes Magnetfeld erzeugen, werden sie weiter weg platziert.



Abbildung 42: Dauermagnet⁴⁹

⁴⁸ OCEUMAOA Neodym Magnete mit Loch 30kg Magnet Stark,Magnet zum Anschrauben mit Schrauben für Werkstatteinrichtung Zubehör 25mm 4 Stück : Amazon.de: Küche, Haushalt & Wohnen.

9.8 Pneumatik

Um ein Vakuum zu erzeugen und um den Satelliten rütteln zu lassen, werden Pneumatik Teile benötigt. Pneumatik ist ein Bereich der Technik, der Druckluft nutzt, um mechanische Arbeit einfach zu verrichten. Es bietet eine vielseitige und sichere Methode für verschiedene Anwendungen wie Automatisierung, Maschinenbau und Fahrzeugtechnik. Durch die Verwendung von pneumatischen Aktuatoren wie Zylindern und Ventilen können Bewegungen gesteuert und automatisiert werden. Insgesamt ermöglicht Pneumatik eine effiziente und zuverlässige mechanische Leistung. Für die Teststation werden folgende Pneumatik Teile benötigt:

Bezeichnung	Stückzahl
Ventil 24 VDC	1
Ventil 24 VDC, Druck justierbar	1
Vakuum Ejektor	1
Druckluft Kugel Vibrator	1

Tabelle 13: Stückliste Pneumatik

Mit den beiden Ventil können wir die Luftzufuhr zu dem Vakuum Ejektor und dem Druckluft Kugel Vibrator steuern. Mit dem Vibrator können wir eine Rüttelplatte verwirklichen und können so den Satelliten auf Vibrationen testen. Die Pneumatik Teile wurden uns von der Firma Hefel Technik GmbH gesponsert, die spezialisiert auf Pneumatik und Automatisierungstechnik sind.

Komplette Zusammenschaltung Pneumatik



Abbildung 43: Zusammenschaltung Pneumatik

Das rechte Ventil ist für die Rüttelplatte und beim rechten ist direkt der Vakuum Ejektor angeschlossen.

9.8.1 Ventil

Das VY-83ELB00-T⁵⁰ ist ein pneumatisches Ventil, das häufig in der Industrie eingesetzt wird, um den Durchfluss von Druckluft zu steuern. Mit diesem Ventil steuern wir die Rüttelplatte und den Vakuum Ejektor. So ein Ventil besteht aus einem elektromagnetischen Spulenmechanismus, der einen beweglichen Ventilspulenkerne betätigt, um den Luftstrom zu öffnen oder zu schließen. Das VY-Solenoidventil bietet eine zuverlässige und präzise Steuerung des Luftstroms, was es ideal für Anwendungen in der Automatisierung, Maschinenbau und anderen industriellen Bereichen macht.

⁵⁰ Datenblatt VY-83ELB00-T.



Abbildung 44: Ventil

Steuern kann man das Ventil, in dem man eine Spannung anschließt. Werden 24V eingespeist, öffnet das Ventil und die Luft kann durchströmen. Sobald die Spannung entzogen wird, schließt sich das Ventil wieder. Die Spannungszufuhr wird mit einem Relais gesteuert.

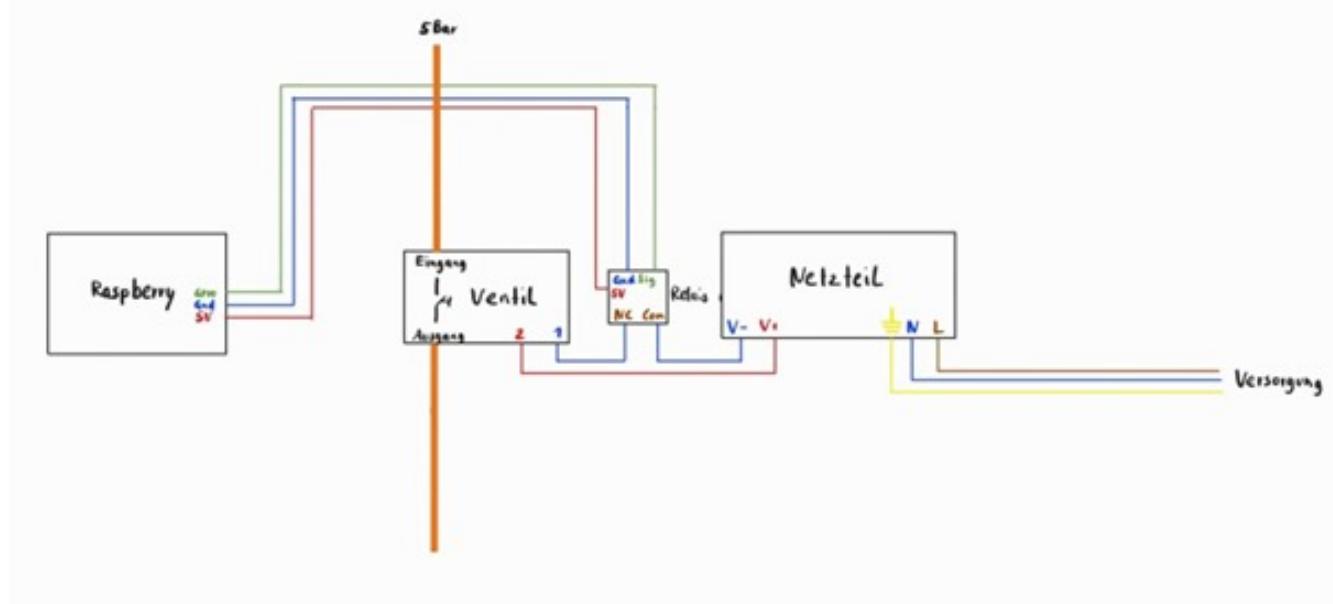


Abbildung 45: Zusammenschaltung Ventil

9.8.2 Vakuum Ejektor

Da wir den Satelliten in einem Vakuum testen müssen, benötigen wir ein Vakuum Erzeuger. Ein Vakuum ist ein physikalischer Zustand, der durch das Fehlen von Materie in einem bestimmten Bereich gekennzeichnet ist. Es ist ein Raum, der frei von Gasen, Flüssigkeiten oder Feststoffen ist. In einem Vakuum gibt es keinen atmosphärischen Druck, da kein Medium

vorhanden ist, das diesen Druck ausüben könnte. Da der Weltraum nahezu leer von Materialien ist und deshalb ein Vakuum ist, wird der Satellit auch auf diese Einflüsse getestet.

Ein Vakuum-Ejektor⁵¹ erzeugt ein Vakuum durch Absaugen von Luft oder Gas aus einem geschlossenen Raum. Dies geschieht durch Hochdruckluft, die durch eine Düse strömt und einen Unterdruck erzeugt. Vakuum-Ejektoren sind effizient, zuverlässig und erfordern wenig Wartung. Mit dem Ventil kann der Vakuum Ejektor gesteuert werden.

Wir haben uns für den Typ CV⁵² entschieden, da dieser haltbar und langlebig und langlebig ist, dieser Typ ist der beliebteste Ejektor. Mit diesem kann ein Vakuumwert von - 0,58 oder - 0,92 Bar erreicht werden.

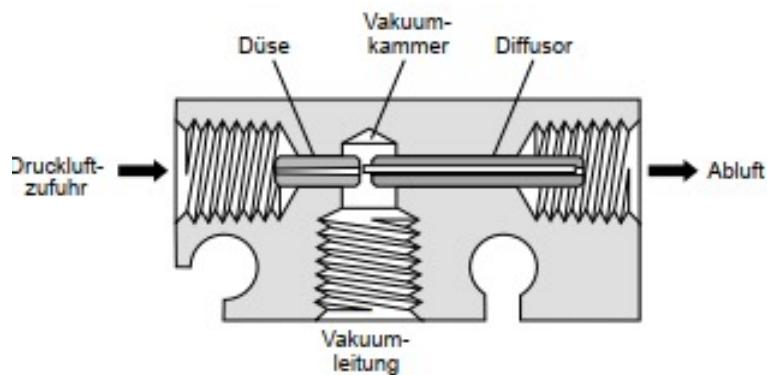


Abbildung 46: Skizze Vakuum-Ejektoren⁵³

Der Satellit wird dann in eine Vakuum Glocke gegeben. Eine Vakuumglocke ist eine abgedichtete Glaskuppel, die in unserem Fall über den Satelliten platziert wird, um ein Vakuum zu erzeugen. Sie schützt den Satelliten vor äußeren Einflüssen wie Luft, Staub oder Feuchtigkeit und wird oft in Labors und Werkstätten verwendet, um eine kontrollierte Atmosphäre zu schaffen.

⁵¹ Datenblatt Vakuum Ejektor.

⁵² Datenblatt Vakuum Ejektor.

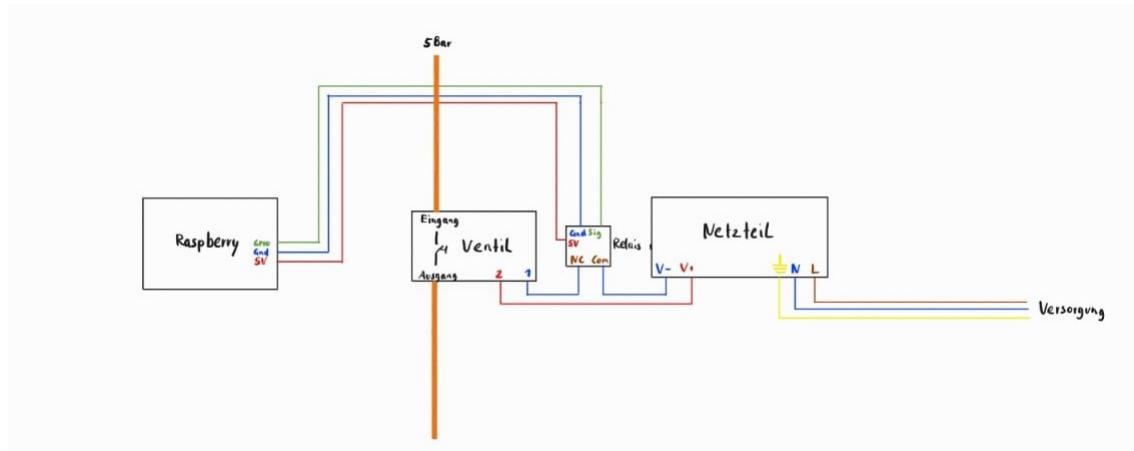


Abbildung 47: Ejektor Zusammenschaltung

9.8.3 Druckluft Kugel Vibrator

Ein pneumatischer Kugelvibrator ist ein Gerät, das verwendet wird, um Vibrationen zu erzeugen, indem Druckluft durch einen Hohlraum geleitet wird, der eine Kugel enthält. Hier ist eine Beschreibung, wie ein pneumatischer Kugelvibrator funktioniert: Ein pneumatischer Kugelvibrator besteht aus einem Gehäuse, das eine oder mehrere Kugeln enthält, die sich frei bewegen können. Das Gehäuse hat Ein- und Auslassöffnungen, durch die Druckluft ein- und austreten kann. Die Zufuhr wird mit einem Ventil gesteuert.

Druckluft wird durch die Einlassöffnung in das Gehäuse geleitet, der Druck wird mit dem Ventil gesteuert. Diese Luftströmung erzeugt eine Druckdifferenz im Inneren des Gehäuses, die die Kugel oder Masse dazu bringt, sich zu bewegen.

Wenn die Druckluft in das Gehäuse strömt, dreht sich die Kugel im Inneren. Diese Bewegung erzeugt Vibrationen, die auf das umgebende Material oder die Maschine übertragen werden.

Die Intensität der Vibration kann durch die Steuerung des Luftdrucks und der Luftströmung gesteuert werden. Durch Anpassen dieser Parameter kann die Vibrationsstärke des Kugelvibrators angepasst werden.



Abbildung 48: Druckluft Kugel Vibrator⁵⁴

9.8.4 Rüttelplatte

Um den Druckluft Kugel Vibrator zu verwenden, muss eine Plattform gebaut werden, die beweglich ist, aber trotzdem fix verschraubbar ist. Dazu sind sogenannte Schwingungsdämpfer nötig. Diese bestehen aus Gummi und können mit M8 Schrauben befestigt werden. Auf die Plattform wird der Satellit platziert und kann durchgeschüttelt werden. Die Rüttelplatte wird an den Dämpfungselementen mit einem Rest-Aluprofile verschraubt.

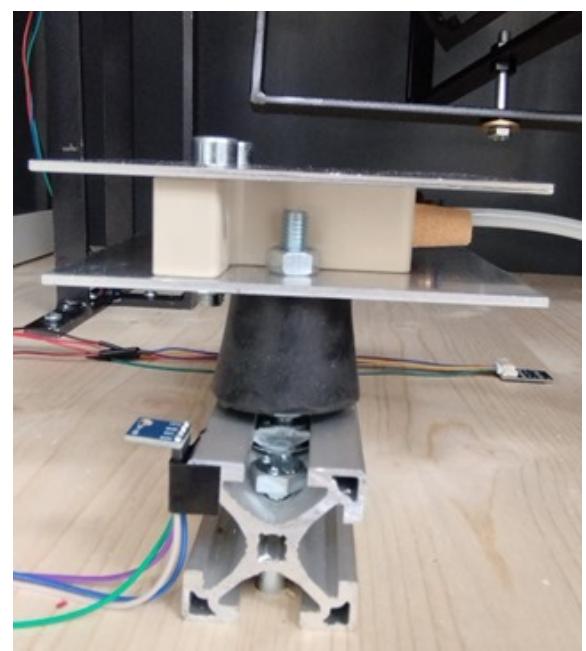
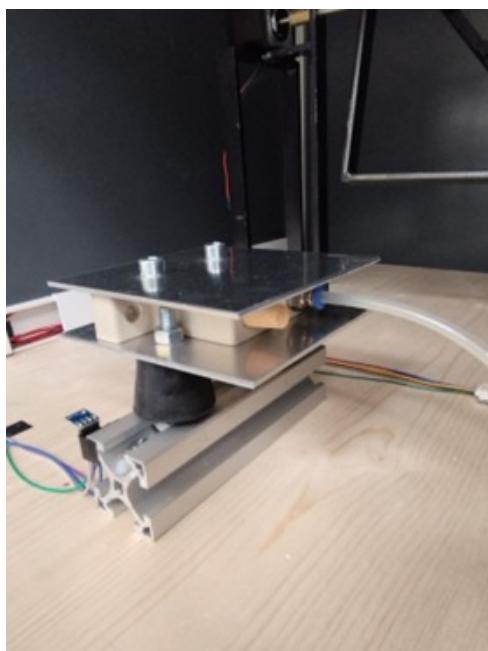


Abbildung 49: Rüttelplatte

Da die Schrauben von den Schwingungsdämpfern zu kurz sind, werden sie an einer Aluplatte befestigt. Diese Platte ist mit dem Vibrator verschraubt. Eine zweite Aluplatte ist notwendig, um den Satelliten drauf zu stellen. Die Schrauben werden mit Schraubkleber beschmiert, da sie sich sonst bei ständigen Vibrationen lösen.

9.9 LED-Streifen

Da die PVC-Platten sehr dunkel sind und nur durch die Tür Licht in die Kammer kommt, ist eine Beleuchtung notwendig. Dies wird mit einem einfachen LED-Streifen umgesetzt. Der gewählte LED-Streifen wird mit 24V versorgt und kann dank dem IC WS2811 mit dem Raspberry gesteuert werden. Es kann ein warmes oder kaltes Licht erzeugt werden.

Schaltplan:

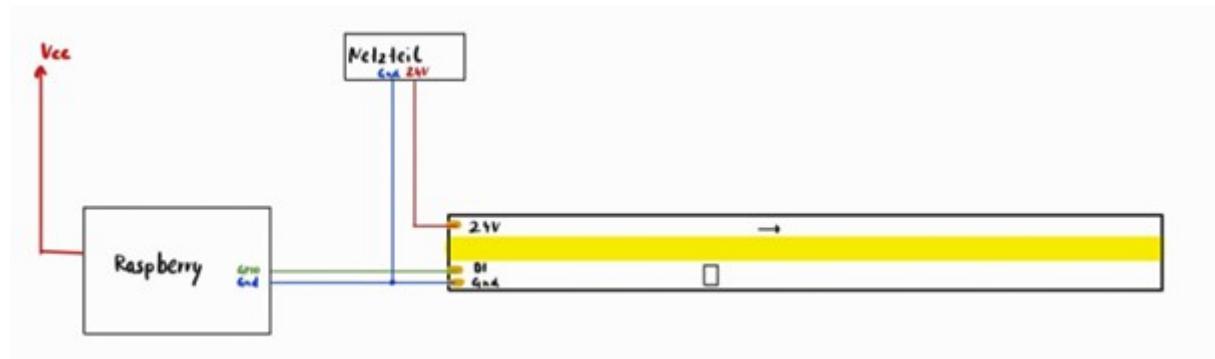


Abbildung 50: Schaltplan LED

Der LED-Streifen wird mit demselben Netzteil als der Motor versorgt. Dieses liefert genau 24V. DI auf dem LED-Streifen steht für Daten In und wird über einen GPIO-Pin gesteuert. GND geht jeweils zum Raspberry Pi und zum Netzteil.

Damit auch die Ecken sauber an den Deckel der Kammer geklebt werden kann, wurde der LED-Streifen auseinandergeschnitten, um 90° gedreht und wieder mit Kabel verbunden. Da der UV-Strahler an dem Decker auf einem Aluprofil montiert ist, muss das Kabel über das Profil gehen, um eine durchgängige Verbindung zu erhalten.

Bei der Verdrahtung ist zu beachten, dass die Pfeile auf dem Streifen berücksichtigt werden.

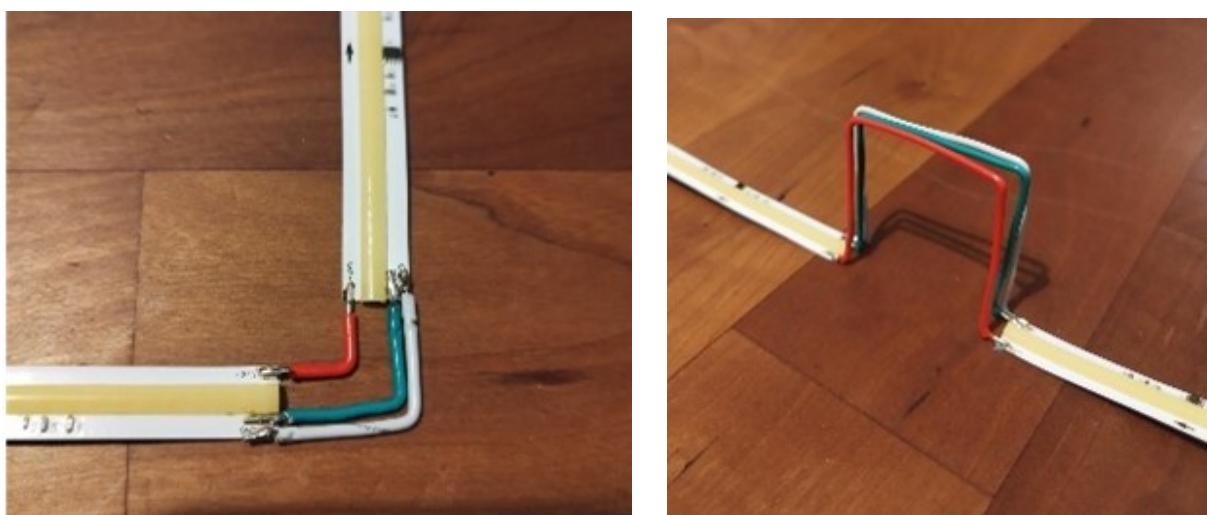


Abbildung 51: LED Verbindung

9.10 CubeSat

Der CubeSat, initiiert vom TU Wien Space Team, ist eine Bildungsmission, die im August 2020 gestartet wurde. Es handelt sich um einen CubeSat, der entwickelt wurde, um Schülern in Österreich die Möglichkeit zu bieten, eigene Software auf ihm auszuführen. Der Satellit enthält eine Bildungsnutzlast mit einem Raspberry Pi, Sensoren und Kameras, die von Schülern über Python programmiert werden können. Ziel ist es, durch den Betrieb dieses CubeSats und einer eigenen Bodenstation das Interesse und Wissen über Raumfahrttechnologien zu fördern.

Das detaillierte Modell des CubeSats im Dateiformat für SolidWorks habe ich von der Technischen Universität Wien erhalten. Dieses hochpräzise Modell stellt eine wesentliche Ressource für die Diplomarbeit dar. Das bereitgestellte CubeSat-Modell zeichnet sich durch seine Detailgenauigkeit aus.

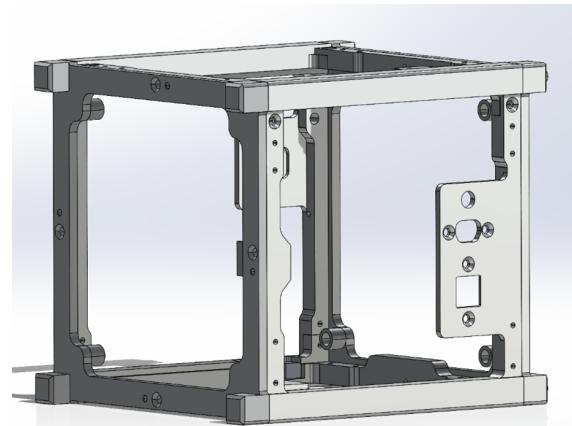


Abbildung 52: 3D-Modell des CubeSats

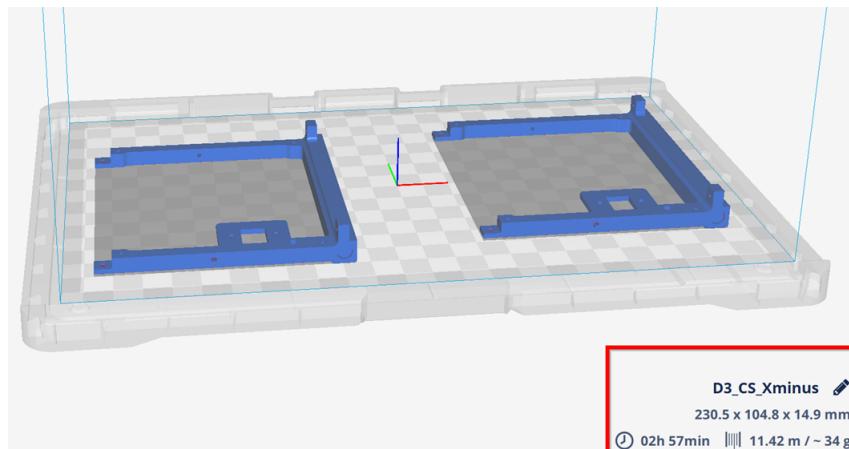


Abbildung 53: Druckvorbereitung von zwei Teilen des CubeSats

Wie man dem Bild entnehmen kann, sieht man auch, wie lange der Druckvorgang dauert, wie viele Meter Filament benötigt werden und wie viel der Druck wiegt.

Da sechs Seiten mit dem 3D-Drucker gedruckt werden müssen und maximal zwei Teile gleichzeitig gedruckt werden können, muss man den Vorgang mit den jeweiligen Teilen noch zweimal wiederholen.

Nachdem alle Teile des CubeSats gedruckt worden sind, muss der CubeSat noch zusammengeschraubt werden.



Abbildung 54: Ausgedrucktes Modell des CubeSats

Um den Raspberry Pi sicher an einer Metallplatte zu befestigen, wurden vier Löcher in die Platte gebohrt. Für die Montage kamen spezielle Kunststoffschrauben, Standbolzen und Muttern zum Einsatz, um Kurzschlüsse zu vermeiden.

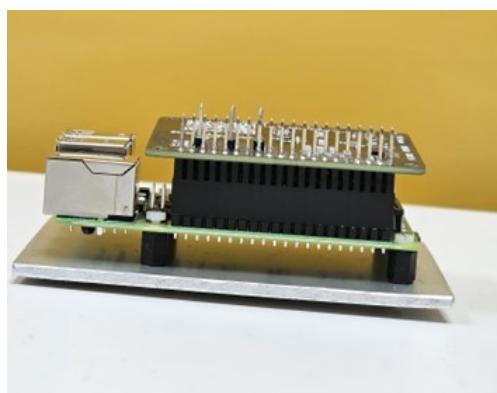


Abbildung 55: Raspberry Pi an der Metallplatte befestigt

Der Raspberry PI wurde aufgrund von Stabilitätsgründen mit Abstandbolzen auf einer Metallplatte montiert und aus Isolationsgründen wurden Abstandsbolzen aus Kunststoff verwendet. Durch diese sorgfältige Vorgehensweise konnte ich sicherstellen, dass der Raspberry Pi optimal für seinen Einsatz im CubeSat vorbereitet ist, ohne die Sicherheit und Zuverlässigkeit des Systems zu gefährden.

Bezeichnung	Stückzahl
Kunststoffschrauben (M2,5)	4
Kunststoffstandbolzen	4
Kunststoffmuttern	4

Tabelle 14: Stückliste CubeSat

Um eine optimale Haftung zu erzielen, wurde die Metallplatte mit dem Epoxidkleber sorgfältig am CubeSat angebracht. Bei diesem Vorgang kommt eine Schraubzwinge zum Einsatz, ein entscheidendes Werkzeug, um Druck auszuüben und so die Platte während des Aushärteprozesses des Klebers fest an ihrem Platz zu halten.

Die Schraubzwinge gewährleistet, dass der Kleber gleichmäßig verteilt wird und keine Luftblasen oder Unregelmäßigkeiten die Verbindung schwächen können. Diese Methode garantiert eine hohe Zuverlässigkeit, Stabilität und Halt für die späteren Rotationstests.



Abbildung 56: Fertiges CubeSat-Modell

Abbildung zeigt den CubeSat mit der erfolgreich angepassten Aluminiumplatte mit dem Raspberry Pi.

9.11 Halterung

Das Konzept für die Halterung des CubeSats zielt darauf ab, eine effektive Lösung zu finden, um den CubeSat am Gyroskop zu befestigen. Dabei gilt es zu beachten, dass möglichst alle Seiten des CubeSats offen und zugänglich bleiben. Die Halterung soll an zwei Punkten mit dem Gyroskop befestigt werden.

Der Kern dieses Konzepts ist ein Käfigsystem, das aus zwei Hauptteilen besteht, nämlich einem Boden und einem Deckel. Dieses Design ist speziell darauf ausgelegt, den CubeSat vollständig zu umschließen und durch Schrauben sicher zu fixieren. Es gibt dann somit am Boden und am Deckel jeweils einen Verbindungspunkt zum Gyroskop.

9.11.1 Boden

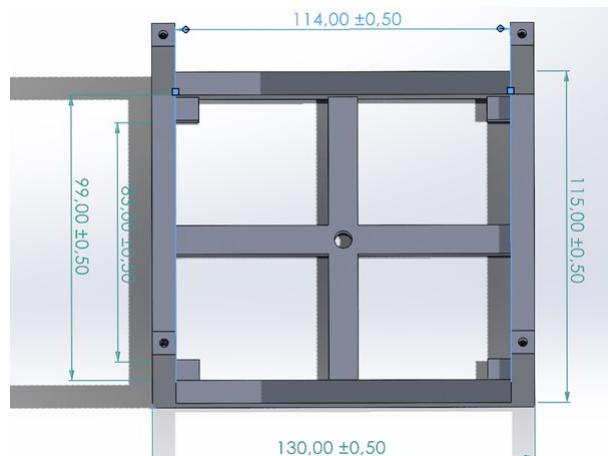


Abbildung 57: Boden

Das Bild zeigt eine technische Zeichnung einer rechteckigen Grundfläche mit den Maßen 130 mm in der Länge, 115 mm in der Breite und einer Materialstärke von 8 mm. Vier Pfosten erstrecken sich von den Ecken der Grundfläche, wobei die Pfostenlänge 77 mm beträgt.

In jedem Pfosten befindet sich eine Gewindebohrung, die eine Tiefe von 15 mm aufweist und für Schrauben mit einem Durchmesser von 4 mm vorgesehen ist. Im Mittelpunkt der Grundfläche ist ein Loch eingezeichnet, das als Befestigungspunkt für ein Gyroskop dient.

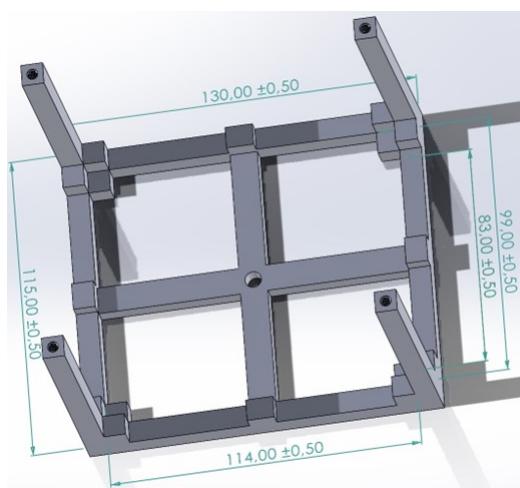


Abbildung 59: fertiger Boden

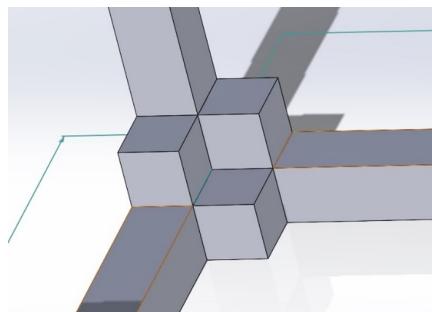


Abbildung 58: Zusätzliche Halterung

Jede Ecke der Struktur wird verstärkt, indem an jeder Ecke zwei zusätzliche Stützen angebracht werden. Diese Stützen dienen dazu, den CubeSat sicher und fest in der Struktur zu verankern, indem sie zusätzlichen Halt bieten und verhindern, dass sich der Satellit innerhalb des Trägerrahmens bewegt.

Um einen noch besseren Halt des CubeSats sicherzustellen, habe ich weitere Stützen jeweils in der Mitte der Grundfläche angebracht. Diese spannen den CubeSat noch besser ein und bieten dadurch eine noch bessere Stabilität.

9.11.2 Deckel

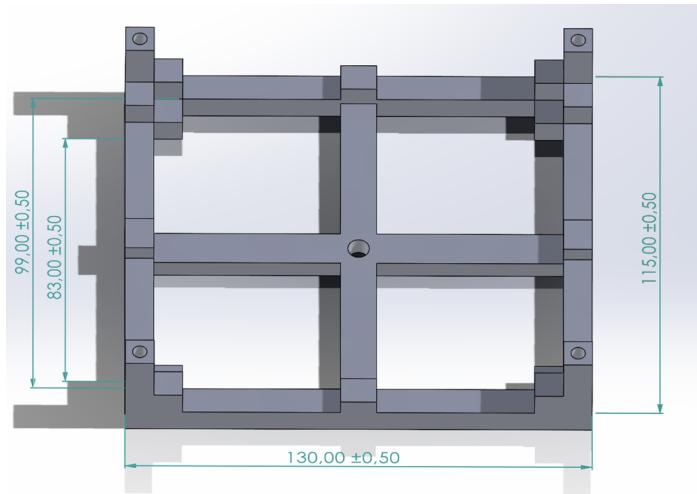


Abbildung 60: Deckel

Das Bild zeigt die technische Zeichnung einer rechteckigen Grundfläche, welche in seinen Grundmaßen und Konstruktionsmerkmalen dem Bodenteil ähnelt.

Die Abmessungen des Rechtecks betragen 130 mm in der Länge und 115 mm in der Breite, mit einer Toleranz von $\pm 0,50$ mm, und die Materialdicke ist ebenfalls gleichbleibend. Auch der Deckel hat im Mittelpunkt ein Loch, welches als Verbindung zum Gyroskop dient.

Der wesentliche Unterschied zwischen dem Deckel und dem Boden des Geräts besteht in der Länge der Pfosten, die beim Deckel auf 23 mm reduziert wurde, um eine Verbindung mit dem Boden mittels 4 cm langen Schrauben zu ermöglichen. Diese Anpassung sorgt für eine sichere Befestigung zwischen beiden Teilen.

Ein weiteres wichtiges Detail ist die Art der Löcher in den Pfosten. Während die Pfosten des Bodens Gewindebohrungen mit einer Tiefe von 15 mm aufweisen, die für Schraubverbindungen vorgesehen sind, sind die Pfosten des Deckels mit durchgehenden Löchern versehen.

Diese durchgehenden Löcher erlauben es, Befestigungselemente komplett durch die Pfosten hindurchzuführen, was eine durchgängige Verbindung zwischen dem Deckel und dem Boden aufweist.

9.11.3 3D-Druck der Halterung

Um den Boden und den Deckel drucken zu können, müssen die STL-Dateien in Dremel Digilab 3D Slicer wie folgt verarbeitet werden:

Als erstes wird das 3D-Modell (üblicherweise im .stl-Format) importiert. Danach wird das Modell auf der virtuellen Druckplatte platziert und nach Bedarf angepasst. Anschließend wird das Material ausgewählt und die Druckqualität sowie die Füllungsdichte festgelegt.

Bei Bedarf können auch Stützstrukturen hinzugefügt werden und Raft und Brim für eine bessere Haftung auf der Druckplatte genutzt werden. Mit Klicken auf „Slice“ wird eine Gcode-Datei erzeugt, welche dann auf den 3D-Drucker übertragen und gedruckt werden kann.

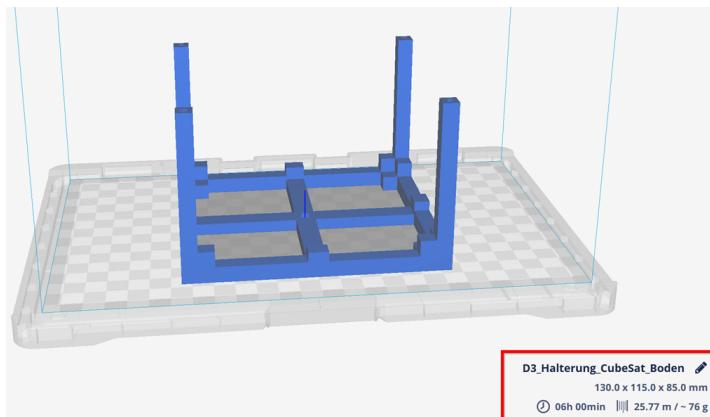


Abbildung 61: Druckvorbereitung für den Boden

Abbildung zeigt, wie lange der Druckvorgang geht, wie viele Meter Filament benötigt wird und wie viel der Druck wiegt. Dieser Vorgang muss für den Deckel wiederholt werden.

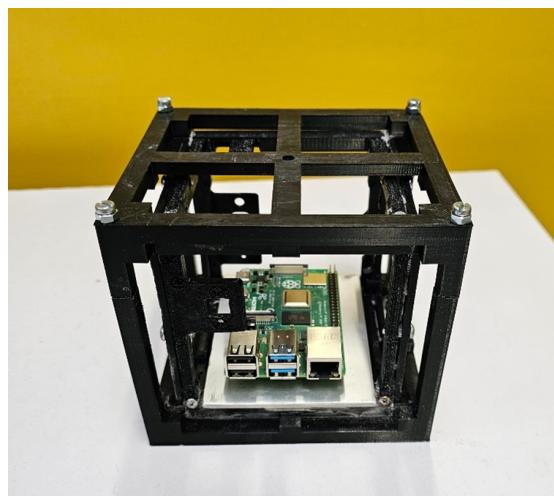


Abbildung 62: CubeSat in der Halterung

Der CubeSat ist sicher im Käfig verankert. Der Käfig selbst besteht aus einer Struktur, die mit Schrauben und Muttern an den Ecken zusammengehalten wird. Die Konstruktion ist stabil und robust, was für eine gute Haltbarkeit spricht und somit bestens für die Rotations- tests im Gyroskop geeignet ist.

Bezeichnung	Stückzahl
Schrauben	4
Muttern	4

Tabelle 15: Stückliste Halterung

9.12 Lüftung

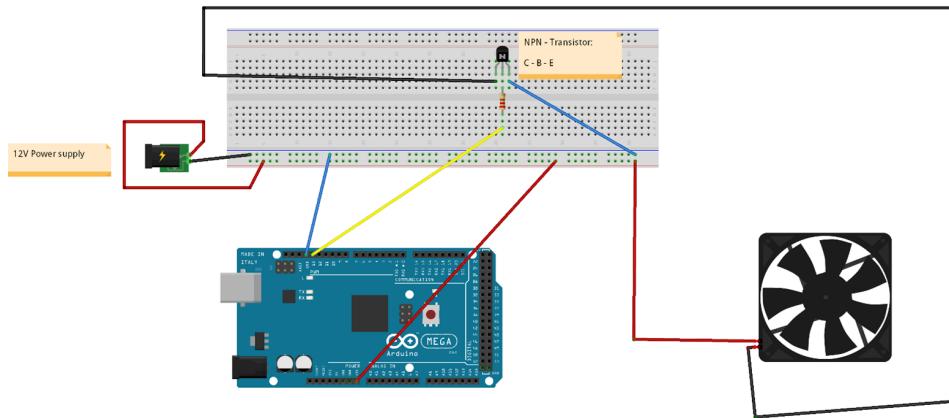


Abbildung 63: Arduino Lüftungssteuerung (Verkabelung)⁵⁵

Das Bild zeigt ein elektronisches Schaltbild, in dem ein Arduino⁵⁶ Board verwendet wird, um einen Lüfter zu steuern. Den Arduino habe ich nur verwendet, um die Funktion zu testen.

Der Arduino ist mit einer 12V-Stromversorgung verbunden, welche für die Versorgung des Lüfters verwendet wird. Da der Arduino mit einer Versorgungsspannung von 5V arbeitet habe ich einen NPN-Transistor verwendet, der als Schalter dient und von einem digitalen Pin des Arduinos gesteuert werden kann.

Darüber hinaus verwende ich in der Schaltung auch einen 220 Ohm Widerstand, der den Strom begrenzt und von einem digitalen Pin des Arduino bzw. des Raspberry Pi zur Basis des Transistors fließt. Das schützt den Mikrocontroller Pin vor Überlastung.

Die Schaltung wurde im ersten Schritt auf dem Steckboard aufgebaut und mit dem Arduino getestet. Danach wurde diese Schaltung auf einer Streifenrasterplatine aufgebaut und wieder getestet.

9.12.1 Pinbelegung des BC547

Pin	Name	Symbol
1	Kollektor	C
2	Basis	B
3	Emitter	E

Tabelle 16: Stückliste Pneumatik

⁵⁶ Arduino - Home.

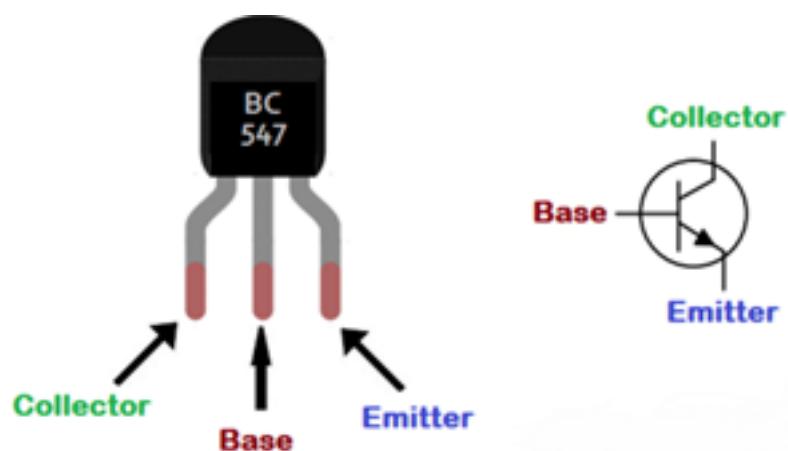


Abbildung 64: Pinbelegung des BC547-Transistor⁵⁷

Abbildung 60 zeigt die symbolische Pinbelegung des Transistors (BC547) als Bauteil und als Schaltsymbol. Der Transistor hat drei Anschlüsse: Basis (B) oder Steueranschluss, Kollektor (C) oder Ausgangsanschluss und Emitter (E) oder Masseanschluss. Die Basis ist der Steueranschluss, über den der Fluss der Elektronen zwischen Emitter und Kollektor gesteuert wird.

9.12.2 Aufbau Lüfter

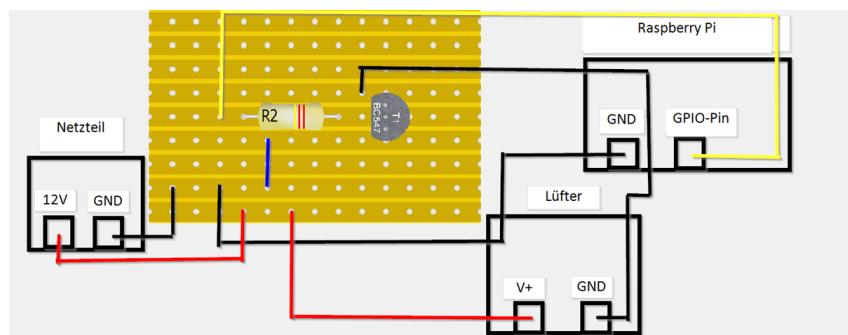


Abbildung 65: Aufbau Streifenrasterplatine (Lüftungssteuerung)

Die Abbildung zeigt den Aufbau für die Lüftungssteuerung mittels Streifenrasterplatine.

9.12.3 Befestigung

Das Plexiglas hat die Maße 72cm x 22,5cm und der Lüfter hat die Maße 12cm x 12cm.

Der Lüfter muss in der Mitte des Plexiglasses befestigt werden. Für die mechanische Befestigung des Lüfters am Plexiglas sind vier Löcher vorgesehen.

Danach wird eine Bohrung im Mittelpunkt gemacht, damit man später den Durchmesser des Lüfters ausschneiden kann.



Abbildung 66: Lüfter am Plexiglas

9.13 Kühlung

In der Testkammer kann es sehr warm werden, weshalb eine Regelung der Temperatur wichtig ist. Diese erfolgt durch den Einsatz eines Kühlgeräts. Die Suche nach einem passenden Kühlgerät gestaltete sich jedoch schwierig, da es spezifische Anforderungen erfüllen musste: Es war notwendig, dass das Gerät über analoge Knöpfe verfügt. Der Grund dafür liegt in der Notwendigkeit, dass wir es zunächst über einen Arduino und später über einen Raspberry Pi steuern können.

9.13.1 Verkabelung des Kühlgerät

Abbildung zeigt das Kühlgerät mit den analog zu bedienenden Knöpfen.

Das Gerät ist auch ohne die Zugabe von Wasser als Ventilator nutzbar.

Es gibt zwei verschiedene Geschwindigkeitsstufen.

Das in Abbildung 63 gezeigte Kühlgerät konnte durch einige Modifikationen von einem Raspberry Pi bzw. einem Arduino angesteuert werden.



Abbildung 67: Kühlgerät mit analogen Knöpfen⁵⁸

Der Luftkühler:

- Der Luftkühler verfügt über einen Ventilator, der warme Luft ansaugt und sie im Inneren des Gerätes durch einen mit Wasser befeuchteten Filter befördert. Dadurch wird die warme Luft effizient abgekühlt und tritt wieder aus dem Gerät aus. Eine angenehm kühle Brise entsteht.

Wassertank:

- Der blaue Wassertank ist entnehmbar. Man kann den Wassertank am Griff aus dem Gerät entnehmen und somit sehr einfach mit frischem Wasser nachfüllen. Der Wassertank hat ein Volumen von 0,75 Litern.

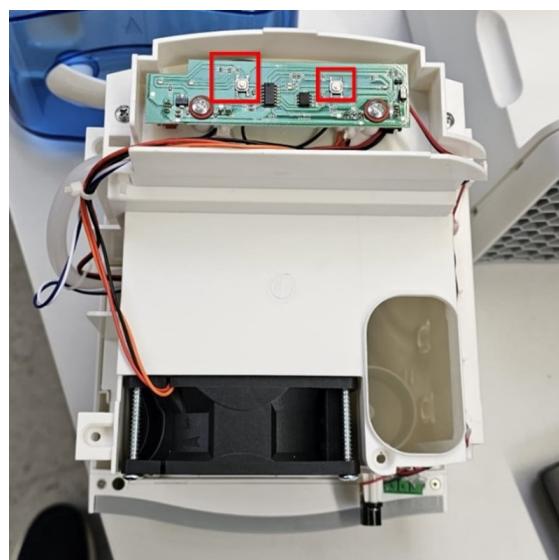
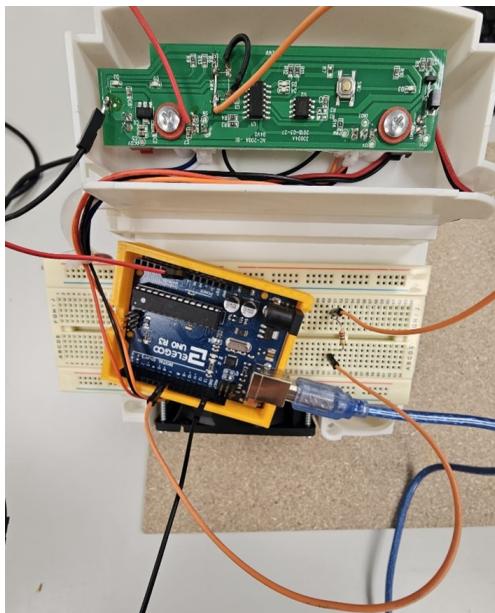


Abbildung 68: Auseinandergebautes Kühlgerät

Abbildung zeigt die zwei Taster für die Steuerung des Kühlgerätes. Der linke Taster steuert die 2 Stufen des Kühlgerätes und der rechte Taster dient zur Befeuchtung des Filters.

9.13.2 Verdrahtung der Taster



Der Taster wird nun verdrahtet. Jeweils eine Verbindung besteht vom Anschluss 1 zum Anschluss 2 und von Anschluss 3 zum Anschluss 4. Vom Anschluss 2 führt eine Verbindung über den Widerstand zum Mikrocontroller.

Der gleiche Vorgang muss auch für den 2. Schalter durchgeführt werden.

Abbildung 69: Verkabeltes Kühlgerät
für Lüftungsstufen

Die rote Leitung ist Vin, diese Leitung ist allerdings nur kurz zu Testzwecken verwendet worden. Da der Arduino per USB mit Strom versorgt wird, ist die Vin Leitung nicht notwendig.

9.13.3 Aufbau mittels Streifenrasterplatine

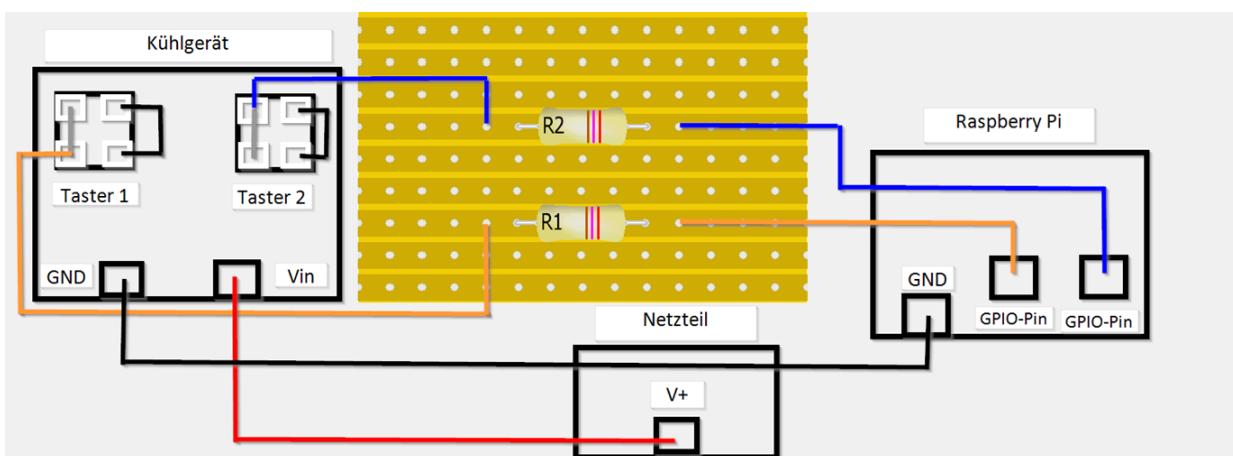


Abbildung 70: Aufbau Streifenrasterplatine (Kühlgerät)

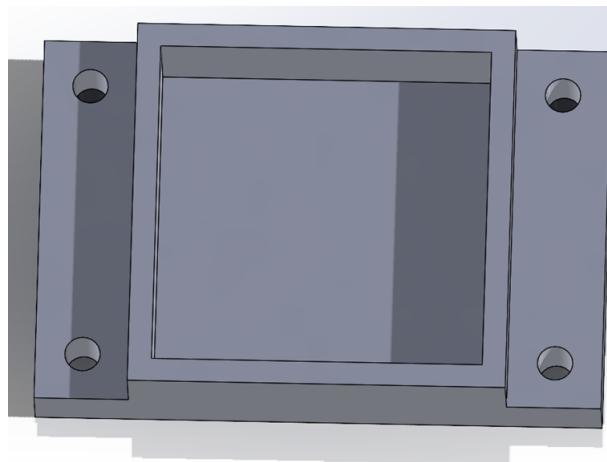


Abbildung 71: Halterung der Streifenrasterplatine

Die Grundfläche der Halterung besteht aus einem 3,2 cm x 3,2 cm großem Quadrat, wie die Abbildung 22 zeigt. Damit die Halterung am Kühlgerät abgeschraubt werden kann, falls dies notwendig ist, habe ich die vier Löcher seitlich an der Halterung gemacht.

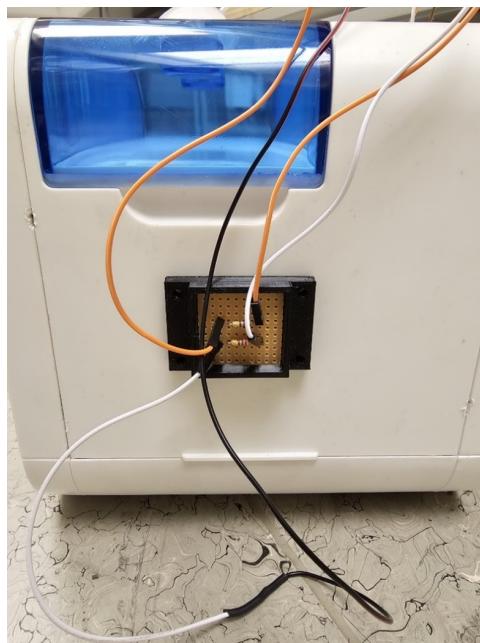


Abbildung 72: Anbringen der Streifenrasterplatine am Kühlgerät

Für die Streifenrasterplatine wurde eine Halterung mit dem 3D-Drucker gefertigt. Diese wurde mittels doppelseitigem Klebeband. Am Kühlgerät angebracht. Die Halterung wurde nicht wie ursprünglich geplant angeschraubt, da das Gehäuse des Kühlgeräts nicht beschädigt werden soll.

9.13.4 Befestigung des Kühlgeräts an der Teststation

Damit das Kühlgerät die Testkammer kühlen kann, muss an der Rückseite der Testkammer ein Loch eingesägt werden. Die Maße des Loches in der PVC-Platte müssen genau mit denen vom Kühlgerät übereinstimmen, damit das Kühlgerät fest darin verankert wird.



(a) Loch in der PVC-Platte



(b) Befestigung des Kühlgeräts auf der Rückseite

Abbildung 73: Kühlung in der Teststation

9.14 Sicherheit

9.14.1 Notausschalter

Ein Notausschalter sorgt dafür, dass das Gyroskop im Notfall sofort ausgeschalten werden kann. Um den Notaus zu implementieren gibt es 2 verschiedene Möglichkeiten, eine Softwarelösung und eine Hardwarelösung. Beide Möglichkeiten haben ihre Vor- und Nachteile und diese werden genauer beschrieben.

Bei der Hardwarelösung wird durch Betätigen des Schalters die Spannungsversorgung vom Netzteil unterbrochen und der Motor stoppt. Die Implementation wäre sehr einfach, doch bei der Hardwarelösung ist das Problem, wenn der Notaus wieder entriegelt wird, der Motor sofort wieder zu drehen beginnt.

Bei der Softwarelösung wird der Notausschalter mit dem Raspberry Pi verbunden und bei Betätigen des Tasters erkennt der Raspberry ein Low Input an einem GPIO-Pin und schaltet den Motor aus. Diese Version ist aber nicht zuverlässig genug, denn wenn der Raspberry eine Störung hat oder das Input Signal nicht erkannt wird, dreht sich das Gyroskop weiter und stellt somit eine Gefahr für den Satelliten dar. Der Vorteil dieser Variante ist, dass programmiert werden kann, dass der Motor nach Entriegelung des Notaus nicht automatisch wieder gestartet wird.

Da beide Lösungen wichtige Funktionen haben, werden die beiden Varianten miteinander verbunden. Der Notausschalter wird zwischen die Spannungsversorgung verbunden und mit einem Relais, das an den Raspberry Pi geschlossen wird, wird der Zustand geprüft. Wird der Notausschalter gedrückt, muss mit einem Software Button das Relais wieder geöffnet werden. Durch diese Implementierung wird ein automatisches Starten nach Entriegelung des Notausschalters verhindert. Dies nennt man quittieren, was so viel bedeutet wie bestätigen. Damit man im User Interface um quittieren erinnert wird, blinkt der Quittierungsbutton ROT.

9.14.2 Schlüsselschalter

Damit nicht jeder die Berechtigung hat, die Teststation in Betrieb zu nehmen, wird ein Schlüsselschalter eingebaut. Wird mit dem Schlüssel der Schalter gedreht, hat man den vollen Zugriff auf die Steuerung mit dem Raspberry Pi.

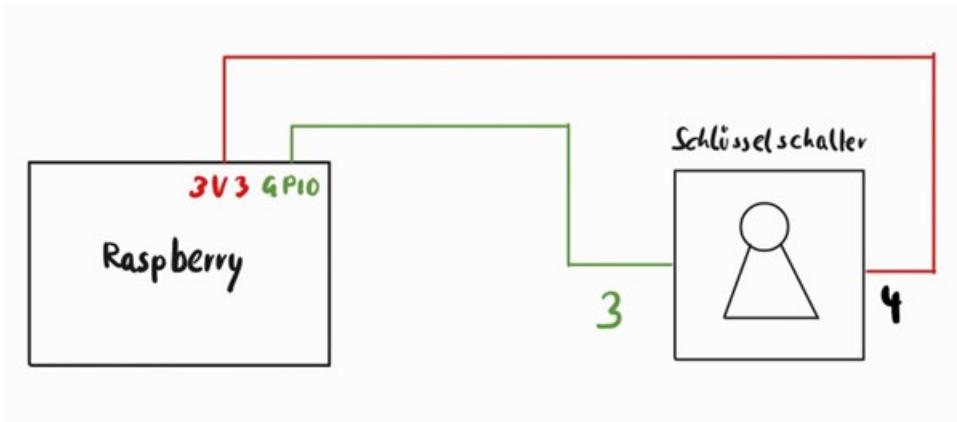


Abbildung 74: Schlüsselschalter

9.14.3 Endschalter

Endschalter sind Schalter, die Endposition eines mechanischen Systems erfassen. Sie kommen in verschiedenen Anwendungen wie Maschinenbau, Automatisierungstechnik und Robotik zum Einsatz, um das Erreichen eines bestimmten Punktes zu erkennen. Ihre Verwendung trägt zur Sicherheit und Effizienz von Maschinen und Systemen bei, indem sie das Erreichen der Endpunkte von Bewegungen oder Prozessen signalisieren.

Verkabelung und Befestigung: Im Rahmen gibt es zwei kleine Aussparungen, in denen die Endschalter platziert werden. Wird nun die Tür geschlossen, werden die Schalter betätigt.

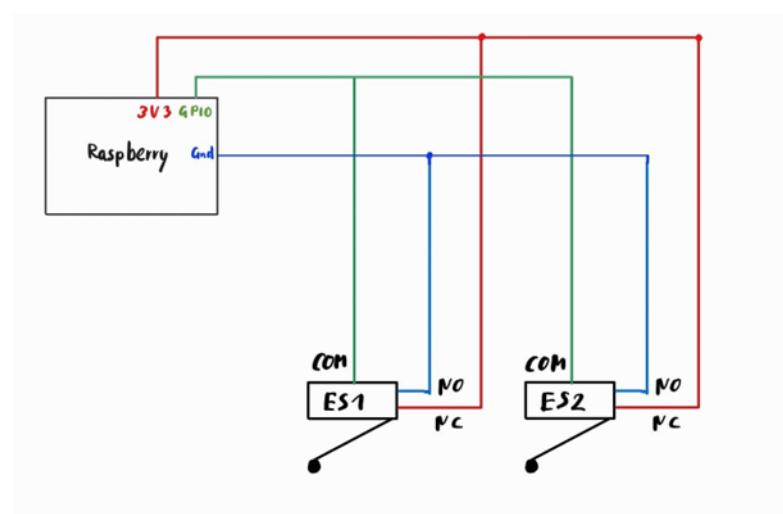


Abbildung 75: Zusammenschaltung Endschalter

Wenn die Teststation während einem Test bewegt wird, könnten durch die Erschütterungen die Endschalter elektrische Störungen erkennen. Um dies zu verhindern, müssen die Endschalter mindestens eine Sekunde lang das jeweilige Signal erkennen.

10 Software

10.1 Tasterabfrage

Die Tasterabfrage wird in verschiedenen Programmabschnitten benötigt. Je nachdem wie oft ein Button gedrückt wird, soll ein anderer Programmteil abarbeitet werden. Dieser Ablauf wird benutzt um zum Beispiel eine Lampe ein und auszuschalten. Um die Tasterabfrage zu realisieren, werden boolsche Variablen verwendet.

Zu Beginn des Programms wird eine boolsche Variable festgelegt, diese wird auf *False* gesetzt. Damit die Variable auch im Unterprogramm verwendet werden kann, muss sie zum Beispiel mit **global** *boolUV* übergeben werden. In der nächsten Zeile wird der Zustand der Variable gewechselt. Da diese am Anfang auf *False* gesetzt ist, wird sie jetzt auf *True* gesetzt. Jedes Mal wenn der Button gedrückt wird, durchläuft das Programm diesen Abschnitt. Durch den Wechsel wird dafür gesorgt, dass nach jedem Drücken einer der beiden Codeteile abarbeitet wird. Wenn die boolsche Variable *True* ist, wird der Code der unter dem Abschnitt *True* steht abarbeitet. Das selbe geschieht, wenn die Variable auf *False* gesetzt ist.

```
#globale boolsche Variable
global boolUV

#Zustandswechsel
boolUV = not boolUV

if boolUV == True :
    #Code um UV-Lampe einzuschalten

else:
    #Code um Lampe auszuschalten
```

Abbildung 76: Beispielcode Tasterabfrage

10.2 Testprogramm

Die Steuerung der Hardware Module wie Motor, Netzteil, UV-Lampe und die Auslesung vom Notaus oder der Sensoren werden mit einer GUI verwirklicht. Diese GUI dient aber nur für Entwicklungszwecke, da diese später von einer API ersetzt wird. Die GUI kann über das LX-Terminal vom Raspberry gestartet werden.

10.2.1 GUI

Um eine GUI zu erstellen und die Ein und Ausgänge vom Raspberry Pi zu steuern, werden 2 Bibliotheken benötigt.

Tkinter: Um eine grafische Benutzeroberfläche zu erstellen, wird die Python-Bibliothek „tkinter“ benötigt. Mit dieser Bibliothek können Fenster, Schaltflächen, Eingabefelder und andere GUI-Elemente erstellt werden.

RPi.GPIO: Die Python Bibliothek RPi.GPIO ermöglicht, mit den Ein und Ausgängen vom Raspberry Pi zu kommunizieren. In unserer Anwendung werden Sensoren eingelesen und externe Geräte gesteuert.

Installieren: Damit man die Bibliotheken verwenden kann, müssen sie auf dem Raspberry Pi installiert werden. Normalerweise ist Tkinter bereits vorinstalliert, doch bei unserem Raspberry Pi nicht. Folgenden Befehlen müssen im Terminal ausgeführt werden um die Bibliotheken zu erhalten.

```
Sudo apt-get update
Sudo apt-get install python3 -tk
Sudo apt-get install python3-pip
Sudo pip install Rpi.GPIO
```

Einbinde der Bibliotheken:

```
from tkinter import *
import RPi.GPIO as GPIO
```

Abbildung 77: GUI-Bibliothek einbinden

Fenster erstellen: Dieser Code erstellt ein Hauptfenster für eine GUI-Anwendung mit dem Titel "Controll Unit", einem weißen Hintergrund und einem Frame namens rightFrame, das als Container für weitere GUI-Elemente innerhalb des Hauptfensters dient. Die Höhe dieses Fensters ist 800 Pixel und eine Breite von 800 Pixel.

```
#Fenster Erstellen
root = Tk()
root.wm_title("Controll Unit")
root.config(background = "#FFFFFF")

rightFrame = Frame(root, width=500, height =800)
rightFrame.grid(row=100, column =200, padx=0, pady=3)
```

Abbildung 78: Beispielcode Fenster erstellen

Ein weiteres Fenster wird erstellt, um darauf alle Buttons zu platzieren. Dieses ist innerhalb des „rightFrame“ und kann so die GUI zwischen Ein und Ausgabe trennen. Für die Sensordaten wird ein weiteres Fenster erstellt.

```
#GUI-Fenster Erstellen
buttonFrame = Frame(rightFrame)
buttonFrame.grid(row=0, column=0, padx=50,pady=3)
```

Abbildung 79: Beispielcode GUI-Fenster erstellen

10.2.2 Button

Fürs Ein- und Ausschalten von Geräten werden Buttons verwendet. Buttons können in der GUI gedrückt werden und die gewünschte Funktion wird gestartet.

```

#Button erstellen:
btmot = Button(buttonFrame, text="Exit", bg = "#FF0000",
width=15, command=exitProgram)
btmot.grid(row=50, column=1, padx=0,pady=5)

#Button Funktion:
#Wird der Button gedrückt, wird folgende
#Funktion gestartet:
def btmot():
    #Funktion...

```

Abbildung 80: Beispielcode Button

10.2.3 Temperatursensor

Die Temperatursensoren wird nicht über ein Button gesteuert, damit die aktuelle Temperatur immer wieder ausgegeben wird. Die empfangenen Daten werden nach dem Starten des Programms, automatisch auf der GUI angezeigt. Durch die zuvor installierte Bibliothek, werden nur wenige Zeilen an Code benötigt, um Sensordaten auszugeben.

```

#Einbinden der Bibliothek im Programm
import adafruit_dht

#Definition des Sensors und festlegen des Pins
dhtDevice1 = adafruit_dht.DHT22(board.D2)

#Funktion temperature einer Variable zuweisen
temp_c = dhtDevice1.temperature

#Funktion humidity einer Variable zuweisen
humidity = dhtDevice1.humidity

```

Abbildung 81: Beispielcode Temperatursensor

10.2.4 UV-Sensor

Wie beim Temperatursensor, wird auch hier die Ausgabe direkt nach dem Starten des Programms erfolgen. Es muss die zuvor installierte Bibliothek eingebunden werden. Danach kann auch hier das Programm in wenigen Schritten programmiert werden.

```
#Einbinden der Bibliothek
import adafruit_ltr390

#I2C-Schnittstelle initialisieren
i2c = board.I2C()

#Definition des Sensors und Übergabe des I2C-Schnittstelle
ltr = adafruit_ltr390.LTR390(i2c)

while True:
    #Ausgabe der gemessenen Werte

    print("UV-Index:", ltr.uvi,
          "Umgebungslicht in Lux:", ltr.light)
```

Abbildung 82: Beispielcode UV-Sensor

10.2.5 Drucksensor

Der Drucksensor wird im Testprogramm verwendet, um die Richtigkeit des Sensors zu überprüfen. Die Werte des Drucksensors können sehr einfach überprüft werden. Indem im Internet nach der Höhe von Rankweil gesucht wird und mit dem gemessenen Wert verglichen werden.

```
#Einbinden der Bibliothek
import bmp180

i2c = board.I2C()
bmp = bmp180.BMP180(i2c)
bmp.sea_level_pressure = 1013.25
return{"Druck: " + str(bmp.pressure) + " hPa",
        "Meereshöhe: " + str(bmp.altitude) + " Meter"}
```

Abbildung 83: Beispielcode Drucksensor

10.2.6 UV-Lampe

Damit getestet werden kann, ob und wie gut die Lampe funktioniert, wurde ein Code zur Ansteuerung für das Relais erstellt. Mithilfe des Button und der Tasterabfrage wird die Lampe ein- und ausgeschaltet.

```
#boolUV als globale Variable deklarieren
global boolUV
#wechseln des bool Zustandes
boolUV = not boolUV

if boolUV == True :
    #Pin auf High setzen (Lampe wird eingeschalten)
    GPIO.output(PinUV,GPIO.HIGH)
    #Text auf Button ändern
    btuv["text"]= "UV-Lampe aus"
    #Lampe ein wird ausgegeben
    print("Lampe ein")

else:
    #Pin auf Low setzen (Lampe wird ausgeschalten)
    GPIO.output(PinUV,GPIO.LOW)
    #Text auf Button ändern
    btuv["text"]= "UV-Lampe ein"
    #Lampe aus wird ausgegeben
    print("Lampe aus")
```

Abbildung 84: Beispielcode UV-Lampe

10.2.7 Motor

Der Motor wird mit einem Button ein und ausgeschalten. Jedes Mal, wenn der Button betätigt wird, sollte der Motor ein oder ausgeschalten werden. Die Buttonabfrage, wurde schon bei der UV-Lampe ausführlich erklärt.

Da die Masse, die bewegt werden muss, relativ schwer ist, wird der Motor als Rampe programmiert. Der Motor braucht dann ca. 3 Sekunden, bis er die Endgeschwindigkeit erreicht. Mit dieser Programmierung braucht der Motor nicht so viel Kraft, um das Gyroskop in Bewegung zu bringen und zusätzlich wird eine ruckartige Bewegung verhindert. Beim Ausschalten wird dasselbe Prinzip angewendet, denn sonst würde es die Wicklungen vom Motor negativ in Anspruch nehmen und die Lebensdauer verkürzt.

#Initialisierung:

```
PUL= 17
EN=13
GPIO.setmode(GPIO.BCM)
GPIO.setup(PUL, GPIO.OUT)
GPIO.setup(EN, GPIO.OUT)
```

#PWM Signal:

```
for i in range(10000000):
    GPIO.output(PUL,GPIO.HIGH)
    time.sleep(0.0005)
    GPIO.output(PUL,GPIO.LOW)
    time.sleep(0.0005)
```

Abbildung 85: Beispielcode UV-Lampe

10.2.8 Netzteil

Das Netzteil kann mit einer Button Betätigung ein und ausgeschalten werden. Dazu wird ein Relais verwendet. Die Funktion des Buttons und der Tasterabfrage 10.1 wurde schon erklärt.

10.2.9 LED-Streifen

Um die LED's zu programmieren wird die Bibliothek „import neopixel“ benötigt. Diese kann in der Console installiert werden.

Mit dieser Bibliothek kann man die LED's einfach steuern.

Folgender Coder verwirklicht das: Mit der „brightness“ und der „fill“ können die LEDs spezifisch gesteuert werden.

```
global boolLED
print(boolLED)
boolLED = not boolLED
print(boolLED)
pixels = neopixel.NeoPixel(board.D21, 40, brightness =5)
pixels.fill((0,0,0))
if boolLED == True :
    pixels.fill((0,255,0))
    btled["text"]= "LEDs aus"
    print(boolLED)
else:
    pixels.fill((0,0,0))
    btled["text"]= "LEDs ein"
    print(boolLED)
```

Abbildung 86: Beispielcode LED-Streifen

10.2.10 Schlüsselschalter

Hier wird einfach nur geprüft, ob der Schalter auf High oder Low ist.

```
SS=6
GPIO.setmode(GPIO.BCM)
GPIO.setup(SS, GPIO.OUT)

if(GPIO.input(SS) == GPIO.LOW):
    continue
else:
```

Abbildung 87: Beispielcode Schlüsselschalter

10.2.11 Notausschalter

Beim Notaus wird derselbe Code wie beim Schlüsselschalter verwendet. Es wird geprüft, ob der Schalter auf High oder Low ist. Die Reaktionszeit, bis das Gyroskop ausgeschalten wird, ist unter 0,5 Sekunden und somit akzeptabel.

10.2.12 Endschalter

Um den Schalterzustand zu erkennen, wird geprüft, ob der Schalter auf High oder Low ist. Sind beide Schalter auf Lauf, ist die Tür geschlossen und der Motor kann gestartet werden.

```

#Initialisierung
END1= 27
END2=22
GPIO.setmode(GPIO.BCM)
GPIO.setup(END1, GPIO.OUT)
GPIO.setup(END2, GPIO.OUT)

#Abfrage
while(TRUE):
    #Tasterabfrage
    #Motoransteuerung
    if(GPIO.input(END1) == GPIO.LOW) and (GPIO.input(END2) == GPIO.LOW):
        continue
    else:
        break

```

Abbildung 88: Beispielcode Endschalter

10.2.13 Kühlung

Der angepasste Code für den Raspberry Pi dient dazu, den Lüfter eines Kühlgerätes in zwei Stufen zu steuern und gleichzeitig einen Filter zu befeuchten. Dabei wird die Tasterabfrage 10.1 genutzt, um zwischen den beiden Zuständen der Kühlung (Ein/Aus) umzuschalten und die Intensität der Kühlung (Stufe 1 oder Stufe 2) sowie die Befeuchtung des Filters zu kontrollieren.

```
#boolUV als globale Variable deklarieren
global boolKühlung
#wechseln des bool Zustandes
boolKühlung = not boolKühlung

if boolKühlung == True :
    #Pin auf High setzen (Kühlung wird eingeschalten)
    GPIO.output(KuehlungPin, GPIO.HIGH)
    #Text auf Button ändern
    btuv["text"]= "Kühlung Stufe 2"
    #Filter Pin auf High setzen
    GPIO.output(filterPin, GPIO.HIGH)
    #Ausgabe in der Konsole
    print("Kühlung eingeschalten - Stufe 1")

else:
    #Pin auf Low setzen (Kühlung kurz ausgeschalten)
    GPIO.output(KuehlungPin, GPIO.LOW)
    #Pausiert das Programm kurz für 0.1 Sekunden.
    time.sleep(0.1)
    #Pin auf High setzen (Kühlung wird eingeschalten)
    GPIO.output(KuehlungPin, GPIO.HIGH)
    #Filter Pin auf High setzen
    GPIO.output(filterPin, GPIO.HIGH)
    #Text auf Button ändern
    btuv["text"]= "Kühlung Stufe 1"
    #Ausgabe in der Konsole
    print("Kühlung eingeschalten - Stufe 2")

    #Kühlung Pin und Filter Pin auf Low setzen
    GPIO.output(KuehlungPin, GPIO.LOW)
    GPIO.output(filterPin, GPIO.LOW)

    #Ausgabe in der Konsole
    print("Kühlung ausgeschalten")
```

Abbildung 89: Beispielcode Kühlung

10.2.14 Lüfter

Dieser Code steuert das Ein- und Ausschalten des Lüfters über einen GPIO-Pin des Raspberry Pi, mithilfe der Tasterabfrage 10.1.

```

#boolUV als globale Variable deklarieren
global boolLüfter
#wechseln des bool Zustandes
boolLüfter = not boolLüfter

if boolLüfter == True :
    #Pin auf High setzen (Lüfter wird eingeschalten)
    GPIO.output(lueftungPin, GPIO.HIGH)
    #Text auf Button ändern
    btuv["text"]= "Lüfter aus"
    #Lüfter ein wird ausgegeben
    print("Lüftung eingeschalten")

else:
    #Pin auf Low setzen (Lüfter wird ausgeschalten)
    GPIO.output(lueftungPin, GPIO.LOW)
    #Text auf Button ändern
    btuv["text"]= "Lüfter ein"
    #Lüfter aus wird ausgegeben
    print("Lüftung ausgeschalten")

```

Abbildung 90: Beispielcode Lüfter

10.3 Verbindmöglichkeit

In Bezug auf die Teststation sollen alle Daten der Sensorik extern über ein drahtloses Verfahren auf einem Laptop/PC empfangen werden, da eine serielle Verbindung durch das Gyroskop in der Teststation viele Komplikationen mit sich bringt. Die Möglichkeiten für solch einen drahtlosen Empfang sind TCP, UDP oder Bluetooth.

Da der CM3+ im Cubesat kein WLAN/Bluetooth-Modul aufweist, muss ein externes Modul über die USB2.0 Schnittstelle am Raspberry PI angeschlossen werden. Ein sogenannter „Dongle“ (USB/Wifi-Bluetooth Modul) wird hierbei verwendet. Da keine normale USB-Buchse am CM3+ angebracht ist (siehe Abb. 88), muss der Dongle mit dem CM3+ durch eine andere Möglichkeit verbunden werden. Das Erstellen einer Buchse für die Verbindung des Dongles wäre hier von Vorteil.



Abbildung 91: Bild des CM3+ von Raspberry PI

10.3.1 UDP

Da für die Simulation des Cubesats ein Raspberry PI 4 benutzt wurde, konnte zuerst das WIFI-Modul auf dem Einplatinencomputer verwendet werden. Für die Kommunikationsmethode wurde dafür das UDP-Protokoll genutzt, welches eine schnelle Datenübertragung sicherstellt. UDP⁵⁹ dient als minimales Netzwerkprotokoll, das sich für das Versenden von Datenpaketen über das Internet eignet. UDP verwendet keine Handshakes, um eine Verbindung zwischen zwei Endgeräten herzustellen. Dies bedeutet, dass Daten ohne vorherige Kommunikation gesendet werden. Das Protokoll führt zu einer schnelleren Übertragung als TCP, ist jedoch beim Datentransfer unzuverlässiger.

⁵⁹ User Datagram Protocol.

10.3.2 TCP

TCP⁶⁰ (transmission control protocol) bietet auch eine gute Übertragungsmöglichkeit, da es mit der Hilfe von dem „Handshake-Vereinbarung“ (siehe Abb. 89) eine fehlerfreie Übertragung des Pakets garantiert. Durch die längere Sendezeit des TCP-Protokolls eignet sich das Protokoll für eine zeitkritische Aufnahme von Daten, wie bei der Teststation vorhanden, nicht.

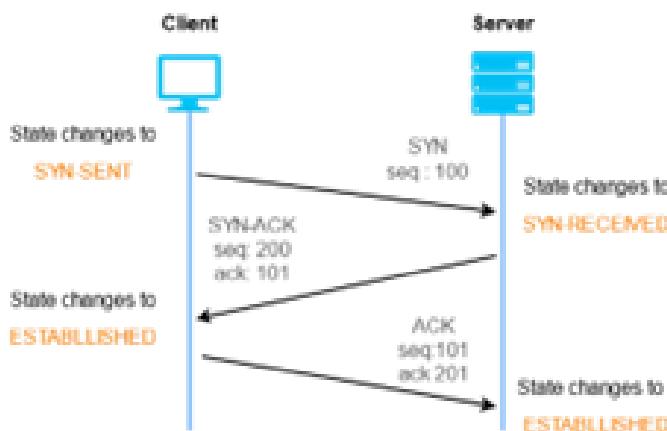


Abbildung 92: Sendeverfahren des TCP-Protokolls⁶¹

Bei dem „Dreifach Handshake Verfahren“, werden SYN- und ACK-Pakete zwischen zwei Endgeräten ausgetauscht. Der Austausch beginnt mit einem SYN-Paket des Clients, gefolgt von einem SYN-ACK-Paket des Servers. Anschließend wird noch ein ACK-Paket vom Client zum Server geschickt. Durch diese Reihenfolge wird sichergestellt, dass Datenpakete in der richtigen Reihenfolge und ohne Fehler übertragen werden. Auch ermöglicht das Protokoll eine Flusskontrolle, um möglichen Datenstau zu vermeiden.

10.3.3 Serielle Verbindungsmöglichkeit

Beginn wurde eine Verbindung über die Schnittstelle UART probiert. Hierbei wird der Raspberry PI über die RX und TX-Pins mit einem Serial/USB Adapter mit einem PC verbunden.

Damit solch eine Verbindung funktioniert müssen die Jumper-Kabel so angeschlossen werden, dass der RX-Pin des Senders mit dem TX-Pin des Empfängers und der TX-Pin des Senders mit dem RX-Pin des Empfängers verbunden ist.

Anschließend lässt sich über die PySerial Bibliothek eine serielle Verbindung erstellen. Für die Datenaufnahme am PC können verschiedene Programme wie TerraTerm oder RealTerm

⁶⁰ Transmission Control Protocol.

verwendet werden. Auch kann ein eigenes Programm geschrieben werden, welches den Prozess kontrollierbarer macht.

Für die Testung der Verbindung wurde ein kleines Programm auf dem Raspberry PI entwickelt, welches Daten über die serielle Schnittstelle sendet und vom Computer über TerraTerm empfangen wird. Für solch ein Skript muss zuerst der serielle Port des Raspberry Pis geöffnet werden. Dies kann über das Interface Menu in der Raspberry PI Config unter „sudo raspi-config“ eingestellt werden. Der Name des Ports und alle anderen offenen Ports lassen sich unter „/dev \$ ls“ anzeigen. Nun kann mit der Python Library „PySerial“ die Verbindung aufgebaut werden:

- Zuerst muss ein Serial-Objekt erstellt werden: `ser = serial.Serial('/dev/ttyS0', 9600)`

```
#Serial Port tty0 wird geöffnet mit einer Baud Rate von 9600
```

- Für das Testen der seriellen Verbindung wurden nur Strings zum Laptop geschickt. Dies funktioniert über:

```
while True:  
    ser.write(b'Test\n') #Sendet den String Test  
    time.sleep(1) # Wartet eine Sekunde
```

- Damit das Programm nicht endlos läuft, wurde noch ein Keyboard-Interrupt eingebaut. Dies stoppt das Programm, sobald die Tastenkombination strg+c gedrückt wird.

```
except KeyboardInterrupt:  
    ser.close()  
    # Schließt die serielle Verbindung,  
    # sobald das Programm unterbrochen wird.
```

Sobald das Skript gestartet wird, fängt der Raspberry PI an, Daten an den Laptop zu senden, wo sie anschließend von TerraTerm aufgenommen werden.

10.4 Übertragungsmethode

Nach weiterer Überlegung und Analyse wurde UDP gegenüber TCP oder einer seriellen Verbindung bevorzugt. Dies liegt daran, dass UDP weniger Overhead hat und daher schneller ist. Es bietet auch eine große Flexibilität, da es keine Verbindung herstellen oder aufrechterhalten muss, was zu einer verbesserten Leistung führen kann.

Eine Übertragung von Paketen in UDP, unter bestimmten Umständen kann auch sicherheitsrelevant sein. Insbesondere wenn es um die Integrität, Verfügbarkeit und Vertraulichkeit der Daten geht. Es ist jedoch möglich, diese Sicherheitsrisiken zu mindern, indem man zusätzliche Sicherheitsmaßnahmen und Protokolle implementiert, die für den jeweiligen Anwendungsfall geeignet sind.

Um eine Verbindung zwischen dem Raspberry PI und dem Computer herzustellen, müssen sich beide Endgeräte im gleichen Netz befinden. Auch sollten beide Systeme die IP-Adresse des anderen im Voraus wissen.

10.4.1 Senderoutine

Das Python Skript, welches sich auf dem Raspberry PI befindet, ist für das Senden der Sensorsdaten zuständig, welche über I²C vom EDU-HAT abgelesen werden. Für die I²C Kommunikation mit der Sensorik wird eine Bibliothek der TU-Wien namens STS1_Sensors.py verwendet. Wie die Auslesung über I²C im Programm funktioniert, wird im Kapitel 10.5.3 noch genauer behandelt.

Um die UDP-Verbindung herzustellen, wird die „sockets“ Bibliothek von Python benötigt. Sie erlaubt es, Daten über das Internet-Protokoll zu senden und zu empfangen. Sockets sind Endpunkte einer bidirektionalen Kommunikationslinie zwischen zwei Programmen, die auf dem Netzwerk laufen.

Die „time“ Bibliothek ist auch ein wichtiger Bestandteil dieses Programms. Sie ermöglicht das Anhalten des Programms für eine bestimmte Anzahl von Sekunden.

Eine weitere Bibliothek, die für dieses Programm nützlich ist, ist die „json“ Library. Speziell die Funktion „json.dumps()“, welche es ermöglicht um ein Python-Objekt (in unserem Fall ein String) in eine JSON-Zeichenkette umzuwandeln. JSON ist leicht lesbar und einfach zu verstehen, was uns beim Schreiben der CSV-Datei deutlich hilft.

10.4.2 Programm SensorOut.py

Dies ist das Programm für die Senderoutine wie in Kapitel 10.4.1 genannt. In diesem Kapitel wird nur das UDP-Sendeverfahren genau besprochen. In Kapitel 10.3.1 werden genauere Informationen über den Vorgang der Datenauslesung dargestellt.

Um die UDP-Verbindung zu erstellen, muss der Raspberry PI wissen, an wen er die Pakete schicken muss. Die Zieladresse und Portnummer sind als Variable angegeben, damit ein Benutzer sie leicht verändern kann.

- `udp_ip = '172.20.10.3'` #Zieladresse: Kann verändert werden
- `udp_port = 50687` #Portnummer: Kann verändert werden

Es ist wichtig zu wissen, dass die Zieladresse immer die Adresse sein muss, bei welcher die Daten empfangen werden sollen. Die Portnummer muss bei Sender und Empfänger gleich gewählt werden.

Ansonsten werden keine Daten am Empfänger ankommen. Da ohne ein Interrupt die Arbeitsschleife des Systems endlos laufen würde, muss eine BOOL Variable erstellt werden, welche abfragt, ob ein Interrupt eingetroffen ist.

```
# Flag Controlling für den Loop
running = True
try:
    while running:
        # Rest des Codes
```

Man kann sehen, dass während das BOOL „running“ den Zustand „True“ besitzt, die Arbeitsschleife ausgeführt wird. Sobald jedoch ein KeyboardInterrupt auftritt, wird die Schleife gestoppt und ist nur durch einen Neustart des Programms wieder ausführbar.

```
except KeyboardInterrupt:
    # Programm stoppt sobald strg+c gedrückt wird
    running = False #Running flag auf False
    print("Loop stopped by user.")
    #Meldung, dass das Programm abgebrochen wurde
```

Running ist nun auf FALSE und entspricht nicht mehr der Bedingung, welche oben genannt wurde.

Die UDP-Senderoutine befindet sich in der Hauptschleife des Programms. Zuerst wird über einen Konstruktor ein neues Socket-Objekt erzeugt. Dieses erstellt einen neuen Socket für die Verwendung mit UDP.

```
# UDP-Socket erstellen und öffnen
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

“socket.AF_INET” ist dabei die Adressfamilie, über welche das UDP-Paket verschickt wird. In diesem Beispiel wird das Paket über IPv4 verschickt. Der Begriff „socket.SOCK_DGRAM“ gibt an, dass die Versendungsmethode UDP sein soll.

Anschließend können die Sensordaten über den Socket verschickt werden.

```
# Daten über UDP-Socket schicken
sock.sendto(data_string.encode(), (udp_ip, udp_port))
```

Die Methode “sock.sendto” versendet den Daten String zu der oben ausgemachten Ziel IP-Adresse über den auch ausgemachten Port

Zum Schluss schließt sich der Socket und es wird eine beliebige Anzahl von Sekunden gewartet. Die Zeit kann bei einer oben angelegten Variable („t_sleep“), je nach Schnelligkeit der Messung, umgeändert werden.

Die Variable t__sleep:
Konfiguration der Wartezeit
t_sleep = 5 #Wartezeit: Kann verändert werden

Schließung des Sockets und Verwendung der Wartezeit:
sock.close() #Schließen des Sockets nach dem Senden
#Warten für t_sleep Sekunden
time.sleep(t_sleep)

Dies ist der Ablauf der Senderoutine auf dem Raspberry PI. Das Programm ist eine Endlosschleife, welche sich nur durch das Ausschalten des Raspberry PI's oder durch die Verwendung des eingebauten Keyboard Interraps (strg+c) unterbrechen lässt.

Das gleiche Programm wurde für das Senden für die Sensordaten der Teststation umgeschrieben. Die Senderoutine ist gleich aufgebaut wie oben erklärt mit dem Unterschied, dass weniger Daten gesendet werden.

10.5 Empfangsroutine

Das zweite Modul, was für eine erfolgreiche UDP-Verbindung verwendet werden muss, ist ein Skript namens „UDPSniff.py“ auf dem Computer, welches die gesendeten Daten empfängt und in eine CSV-Datei einschreibt.

Der Empfang funktioniert in etwa gleich wie das Senden von UDP-Paket. Wie im genannten Skript (siehe: Kap 10.4.2) wird für das Empfangen die „Sockets“ Bibliothek von Python verwendet.

Wie schon zuvor wird die IP-Adresse und der UDP-Port außerhalb der Arbeitsschleife angegeben. Auch wird der Socket geöffnet und die Adresse sowie Port dem Socket zugewiesen:

```
# UDP setup
udp_ip = "0.0.0.0"
#Jede IP-Adresse auf dem Port zuhören
udp_port = 50687
#Port muss gleich wie auf dem Raspi sein
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
#Code zeile wurde in Kap 2.4.1 erklärt
sock.bind((udp_ip, udp_port))
#IP und Port dem Socket zuweisen
```

In diesem Beispiel wird erlaubt, dass für jede IP-Adresse auf dem Port 50687 gehört wird. Falls mehrere IP-Adressen schon über den gegebenen Port kommunizieren, wäre es vorteilhaft die IP-Adresse auf die IP-Adresse des Senders umzuändern. Wie in Kap: 10.4.2 schon genannt ist es wichtig, dass der UDP-Port des Senders und Empfängers identisch ist, da sonst keine Daten empfangen werden können.

Die Daten des Programms werden in der Arbeitsschleife des Programms empfangen. Die Funktion dafür lautet:

```

data, addr = sock.recvfrom(1024)
#Warten auf Date -> Buffer size 1024
print(f"Data received from {addr}")
#Anzeigen von welcher IP-Adresse das Paket
#kommt

```

Die erste Zeile des Code Abschnitts wartet auf ein einkommendes UDP-Paket. Die maximale Größe dieser Daten ist durch die Buffer-Size auf 1024 Bits, begrenzt. Sobald ein Paket auf den angegebenen Port und von der angegebenen IP-Adresse eintrifft, werden die Daten sowie die Sende-Adresse abgespeichert. Für die Prüfung der Korrektheit wird angegeben, von welcher IP-Adresse des Pakets versendet wurde.

Anschließend werden die Daten separiert und in eine CSV-Datei eingespeichert. Dies wird in einem anderen Kapitel (Kapitel 10.5.1) noch genauer erklärt.

Um das Programm zu beenden, wurde ein Keyboard Interrupt eingebaut. Da bei UDP möglicherweise Fehler beim Senden entstehen können, wurden auch verschiedene „Exceptions“ eingebaut. Exceptions reagieren auf Ausnahmen und Fehler, die während der Ausführung des Codes auftreten können.

```

except KeyboardInterrupt:
    print("\nScript terminated by user.")
except Exception as e:
    print(f"An error occurred: {e}")
finally:
    sock.close()
    print("Socket closed.")

```

Die Code Zeile „except Exception as e:“ reagiert auf die Fehler, die im Code auftreten können. Anschließend wird der Fehler auf die Variable gespeichert und im Terminal dargestellt.

Mit der Anwendung von „finally:“ wird der Socket automatisch nach dem Vorkommen eines Fehlers geschlossen, um einen Netzwerkstau am Port zu vermeiden.

Nachdem die Daten angekommen sind, werden sie noch im Terminal als Kontrolle dargestellt:

```
Data received from ('192.168.345.241', 52047)
#Angabe von wo die Daten gesendet wurden
Decoded Data: {'Temperature (TMP112)': 23.9375,
'Temperature (BME688)': 24.32558551326565,
'Humidity': 41.17989757653827,
'Pressure': 96206.06435710512,
'Gas Resistance': 71688.60263231589,
'Acceleration X': -253,
'Acceleration Y': 4,
'Acceleration Z': -2,
'Gyroscope X': -0.9902152641878669,
'Gyroscope Y': 0.015655577299412915,
'Gyroscope Z': -0.007827788649706457,
'UVA': 13, 'UVA Index': 0}

#Darstellung der dekodierten Daten
```

10.5.1 Schreiben der Daten in eine CSV-Datei

Nachdem die Daten am Empfänger angekommen sind, werden sie in eine CSV-Datei geschrieben. Eine CSV-Datei ist eine Textdatei, die Daten in Form einer Tabelle speichert. Jede Zeile in der CSV-Datei entspricht einer Zeile in einer Tabelle. Die einzelnen Werte (Zellen) in dieser Zeile sind durch Kommas getrennt (Abb. 90-91). CSV eignet sich gut für den Datenaustausch zwischen Programmen. Dies ist besonders vorteilhaft, da hier separate Programme für das Einlesen und das Visualisieren von Daten verwendet werden.

Timestamp	Temperature (TMP112)	Temperature (BME688)	Humidity	Pressure	Gas Resistance	Acceleration X	Acceleration Y	Acceleration Z	Gyroscope X	Gyroscope Y	Gyroscope Z	UVA	UVIndex	
2024-01-20 14:37:01	29.875	30.9545688	25.07442062	97348.6048	8988.609665	-4	0	253	-0.01665677	0	0.990215264	25	0	
2024-01-20 14:37:11	29.875	30.9589157	23.99162062	97347.52125	17922.05965	5	2	246	0.019593472	0.0179473581	0	0.979473581	25	0
2024-01-20 14:37:3	29.875	30.9930544	24.5444327	97345.67117	17992.06834	177	25	178	0.092759295	0.074363992	0.72407045	530	0	
2024-01-20 14:37:4	29.875	30.97011334	23.7649997	97343.85259	17992.06834	181	17	180	0.7094414873	0.079454098	0.69667319	564	0	
2024-01-20 15:07:3	30.375	30.68037724	22.48688534	97344.05544	8946.044171	0	-3	255	0	-0.011741683	0.998043053	24	0	
2024-01-20 15:24:4	29.25	30.27248569	28.32777682	97350.04983	8946.044171	0	-1	251	0.03913894	-0.007827789	0.982387476	20	0	
2024-01-20 15:25:0	29.125	30.01574705	28.62167551	97354.96177	8946.044171	1	0	253	0.03913894	0	0.98836137	20	0	
2024-01-20 15:25:1	29.25	29.84448437	27.80163879	97358.26319	8951.075329	5	-2	254	0.019593472	-0.007827789	0.984129159	20	0	
2024-01-20 15:25:11	29.1875	29.70417142	27.64656244	97360.38873	8950.609865	2	0	252	0.007827789	0	0.98836137	20	0	
2024-01-20 15:25:2	29.875	29.85063135	27.50007782	97362.50337	17922.15965	0	-1	251	0	-0.003913894	0.982387476	20	0	
2024-01-20 15:28:0	29.125	29.70417142	27.64656244	97364.62337	17922.15965	1	-3	251	0.007827789	-0.011741683	0.982387476	21	0	
2024-01-20 16:28:0	30.1875	29.853655	24.541629568	97367.83205	8946.044171	4	-2	251	0.019593472	-0.007827789	0.982387476	22	0	
2024-01-20 16:28:1	30.1875	30.10214481	24.93091521	97369.51938	17992.06834	0	-2	251	0	-0.007827789	0.982387476	22	0	
2024-01-20 16:28:2	30.1875	30.20936962	25.30444958	97370.30544	17922.15965	15	7	257	0.05709415	0.02736725	1.050570941	22	0	
2024-01-20 16:28:2	30.25	30.29101818	26.69617783	97371.76365	17937.21973	3	-2	251	0.011741683	-0.015665577	0.982387476	22	0	
2024-01-20 16:28:3	30.375	30.34570492	26.31215055	97372.38918	17992.06834	9	-3	250	0.03225549	-0.011741683	0.978473581	22	0	
2024-01-20 16:28:3	30.4375	30.39724423	26.29880733	97374.10582	17992.06834	4	0	251	0.01565577	0	0.982387476	22	0	
2024-01-20 16:28:4	30.4375	30.44123567	26.19854968	97374.46764	17992.06834	-254	-3	252	-0.007827789	-0.011741683	0.98836137	22	0	
2024-01-20 16:28:4	30.5	30.48900193	26.02340619	97374.88974	17992.06834	7	-3	247	0.02736725	-0.011741683	0.966731898	22	0	
2024-01-20 16:28:5	30.5625	30.52671054	25.91070097	97374.62031	17937.21973	50	-6	250	0.039138943	-0.02483306	0.978473581	22	0	
2024-01-20 16:29:0	30.5625	30.57227638	25.81444116	97375.04915	17992.06834	2	-1	251	0.011741683	-0.007827789	0.982387476	22	0	
2024-01-20 16:29:0	30.6875	30.51815538	25.17034917	97375.52621	17992.06834	6	-3	248	0.02483306	-0.011741683	0.97045793	22	0	
2024-01-20 16:29:1	30.5625	30.65038004	25.34862466	97375.70167	17907.10488	3	-3	251	0.011741683	-0.007827789	0.982387476	22	0	

Abbildung 93: Tabellenform

Timestamp	Temperature (TMP112)	Temperature (BME688)	Humidity	Pressure	Gas_Resistance	Acceleration_X	Acceleration_Y	Acceleration_Z	Gyroscope_X	Gyroscope_Y	Gyroscope_Z	UVA	UVA_Index
2024-03-05 16:03:21,28,-1875,29,586804665900096,27,918712643142367,96051,14691914218,8916,132627472834,9,8,252,0,03522504892367946,0,03131115459882583,0,98638013698630136,58,													
2024-03-05 16:03:26,28,-1875,29,5937180131344187,27,9581210878896,96050,98392054089,8916,132627472834,2,2,252,0,007827788640706457,0,01174168297455986,0,98638013698630136,57,													
2024-03-05 16:03:31,28,-1875,29,617286250278084,27,729775321361554,96051,98501718478,8916,132627472834,3,4,254,0,019569471624266144,0,03131115459882583,0,99412191585227281,64,													
2024-03-05 16:03:37,28,-25,29,6499675488985114,27,6899459943156,96052,45947160144,8916,132627472834,4,5,252,0,01565577299412915,0,015659471624266144,0,98638013698630136,69,													
2024-03-05 16:03:42,28,-1125,29,68264885940666,27,6871137838023825,96051,2848635267,8886,420438767009,0,3,241,0,01174168297455986,0,9432485122856281,66,													
2024-03-05 16:03:48,28,-375,29,79651136310484,27,6680519410560195,96054,3375522518,8886,420438767009,0,2,252,0,01131115459882583,0,007827788649706457,0,9388436528175731,71,													
2024-03-05 16:03:53,28,-375,29,73575601425477,27,649164100451305,96053,7468838761,8889,429995503037,3,13,251,0,01274168297455986,0,0508806362230951974,0,923874755381604,72,													
2024-03-05 16:03:59,28,-4375,29,7561811851367883,27,608341369189315,96053,73507142309,8901,25173852579,2,10,255,0,007827788649706457,0,0913804324851229,0,9988436528175733,88,													
2024-03-05 16:04:04,28,-375,29,78869286185817,27,5711750823794,96054,35118493241,8886,420438767009,2,9,253,0,0508806362230951974,0,015659471624266144,0,9902152641878469,69,													
2024-03-05 16:04:10,28,-4375,29,8889748805159867,27,517088886577766,96051,44420886994,8901,25173852573,7,11,255,0,0273972402739726,0,043952817573385516,0,9988436528175733,89,													
2024-03-05 16:04:15,28,-4375,29,82562971177992,27,82116382737476,96052,67444282295,8886,420438767009,6,3,250,0,0273972402739726,0,039138043248513229,0,00195694716242661,79,													
2024-03-05 16:04:21,28,-375,29,853283201286184,27,9247713977351,96052,54645852563,8908,685968819596,7,7,252,0,0273972402739726,0,0273972402739726,0,98638013698630136,76,													
2024-03-05 16:04:26,28,-5,29,88531609697115,27,7199994789026,96052,920880042815,17772,840877534618,7,7,246,0,0273972402739726,0,0273972402739726,0,9628188039138943,80,													
2024-03-05 16:04:32,28,-5,29,87309864871836,27,51829957447863,96053,606875479867,17772,8408775346018,8,6,253,0,0273972402739726,0,01565577299412915,0,9902152641878669,73,													

Abbildung 94: Darstellungsweise der CSV-Datei

10.5.2 CSV-Datei Programm

Das Programm, welches für das Schreiben der CSV-Datei zuständig ist, ist dasselbe wie jenes, das für das Empfangen des UDP-Pakets verantwortlich ist („UDPSniff.py“). Somit werden alle empfangenen Daten gleich in die CSV-Datei eingeschrieben. Dafür muss zuerst eine Datei erstellt und ihr Pfad angegeben werden:

```
# Pfad der CSV-Datei
csv_file_path='C:\\\\Users\\\\Constantin\\\\Documents
\\\\5BHEL\\\\Diplomarbeit\\\\DashboardSTS1
\\\\udp_data.csv'
```

Wo genau die CSV-Datei liegt, ist egal, solange der genaue Pfad angegeben wird.

Anschließend werden die Kopfzeilen („Headers“) definiert. Diese müssen identisch zu den Titeln der empfangenen Daten sein, da das Programm, je nach Titel, die Werte zu dem angegebenen Header sortiert.

```
# Definition der Headers
headers =
["Timestamp", "Temperature (TMP112)",
"Temperature (BME688)",
"Humidity",
"Pressure",
"Gas Resistance",
"Acceleration X",
"Acceleration Y",
"Acceleration Z",
"Gyroscope X",
"Gyroscope Y",
"Gyroscope Z",
"UVA", "UVA Index"]
```

Nicht in dem UDP-Paket vorhanden ist der Header: „Timestamp“. Dieser wird benötigt, um die Zeit zu erfassen, zu der die Daten empfangen wurden. Der Header wird manuell hinzugefügt, um die Sensordaten bei der Visualisierung zeitlich einzuordnen und um zu analysieren, wie sich die Sensorwerte im Laufe der Zeit verändern.

Anschließend wird in einem „try“-Block, welcher für das Abfragen von Ausnahmezuständen zuständig ist, die CSV-Datei geöffnet.

```
#Öffnen der CSV-Datei im Modus a
with open(csv_file_path, mode='a', newline='') as file:
    # Überprüft, ob die Datei leer ist
    file_empty = file.tell() == 0
    #Erstellen eines CSV writer Objekts
```

Die erste Zeile dieses Codeabschnitts öffnet die angegebene Datei, welche sich am Pfad „csv_file_path“ befindet. Der Modus bestimmt, wie die Daten eingelesen werden sollen. Der Modus „a“ gibt an, dass neu einkommende Daten an das Ende der Datei angehängt werden, ohne schon eingeschriebene Inhalte zu überschreiben. Der Modus „a“ kann aber auch verwendet werden, um eine noch nicht existierende Datei zu erstellen.

Der Parameter „newline=‘ ‘“ wird verwendet, um sicherzustellen, dass die Zeilenumbrüche

korrekt gehandhabt werden. Dies ist besonders wichtig, falls die Datei zwischen mehreren Betriebssystemen geteilt wird. Das oben geöffnete Dateiobjekt `'open ()'` wird der Variable `'file'` zugewiesen, die innerhalb des „with“ Blocks verwendet wird.

`'file.tell()'` gibt die aktuelle Position in der CSV-Datei zurück. Hier wird durch, `„file.tell() == 0“` abgefragt, ob die CSV Datei leer ist.

Anschließend wird ein. `'csv.writer()'` Objekt erstellt, das für das Schreiben von Daten in das `'file'-Objekt` verwendet wird.

```
#Erstellen eines CSV writer Objekts
writer = csv.writer(file)
```

Da ein leeres File keine Kopfzeilen besitzt wird, falls die Datei leer ist, alles Headers eingeschrieben.

```
if file_empty:
#Falls die CSV Datei noch leer ist
writer.writerow(headers)
#Headers als in erste Reihe einfügen
```

Dafür wird zuerst abgefragt, ob die oben definierte BOOL `'True'` ist. Falls dies der Fall ist, werden alle Header, welche zuvor beschrieben wurden, in die erste Reihe der CSV-Datei geschrieben.

In der Arbeitsschleife werden die empfangenen Daten dekodiert und in die CSV-Datei geschrieben:

```
try:  
    #Daten in einen String konvertieren  
    decoded_data = data.decode()  
    #Daten für die Verarbeitung  
    in ein JSON-Objekt umwandeln  
    sensor_data = json.loads(decoded_data)  
    #Ausgabe der dekodierten Daten  
    print(f"Decoded Data: {sensor_data}")  
    #Erzeugung eines Zeitstempels  
    current_time =  
        datetime.now().strftime("%Y-%m-%d %H:%M:%S")  
  
    # Für jeden Zeitstempel werden die Sensordaten eingefügt  
    writer.writerow([current_time] +  
        [sensor_data.get(key, '') for key  
         in headers[1:]])  
    #Buffer der Datei löschen,  
    damit Daten sofort geschrieben werden  
    file.flush()
```

Da der Code potenziell eine Ausnahme (Exception) werfen könnte, beginnt die Arbeitsschleife mit einem ‚try‘ Block. Falls während der Ausführung des Blocks eine Ausnahme auftritt, kann das Programm gleich zu der gegebenen Exception springen.

Um die Daten wieder in ein String Format zu konvertieren, werden die empfangenen Daten mit der Methode ‚.decode()‘ dekodiert und auf die Variable ‚decoded_data‘ abgespeichert.

Anschließend werden die dekodierten Daten für das Einschreiben in die Datei in ein „JSON-Objekt“ umgewandelt und als die Variable ‚sensor_data‘ gespeichert. Obwohl dies schon im Programm der Senderoutine (siehe: Kap. 10.4.2) gemacht wurde, wurde der String für das Senden über UDP in Byte-Daten umgewandelt. Es ist trotzdem wichtig, dass der String ‚decoded_data‘ im JSON-Format geschrieben ist, da die Operation ‚json.loads()‘ voraussetzt, dass dieser eine gültige JSON-Zeichenkette ist.

Auf die Variable „current_time“ wird durch die Funktion ‚datetime.now()‘ die aktuelle Zeit in einen String abgespeichert. Auch wird das Format der Zeit angegeben (Jahr-Monat-Tag Stunde: Minute:Sekunde).

In dem nächsten Abschnitt wird mit der Funktion, „writer.writerow()“ eine Zeile der CSV-Datei geschrieben. Zuerst wird die aktuelle Zeit in die Zeile geschrieben. Anschließend werden die Werte von ‚sensor_data‘ geschrieben, welche den Titeln der Kopfzeilen entsprechen.

Falls ein bestimmter Wert nicht mit dem Header übereinstimmt, wird ein leerer String eingefügt. Der Codeabschnitt, ‚headers [1:]‘ deutet darauf hin, dass das erste Element in ‚Headers‘ übersprungen wird, da diese Spalte für die Zeit reserviert ist.

Um einen möglichen Datenstau bei schnellerem Senden und Empfangen zu vermeiden, wird durch die Funktion ‚file.flush()‘, das System, gezwungen, alle gepufferten Daten sofort in die CSV-Datei zu schreiben.

Da das Programm durch die Implementierung von JSON-Dekodierung anfällig auf Ausnahmen (Exception) sein kann, werden die möglichen Fehler am Schluss des Codes noch abgefangen und aufgeschrieben. Dies wurde auch für generelle Ausnahmen, außerhalb der JSON-Dekodierung, gemacht.

```
except json.JSONDecodeError as e:  
    #Apprüfung von Fehlern der JSON-Dekodierung  
    print(f"JSON Decode Error: {e}")  
    #Ausgabe des Fehlers  
except Exception as e:  
    #Apprüfung für allgemeine Fehler  
    print(f"Unexpected error: {e}")  
    #Ausgabe der allgemeinen Fehler
```

Durch dieses Programm können nun alle empfangenen Daten in eine CSV-Datei in bestimmter Reihenfolge abgespeichert werden und später für die Visualisierung und Analyse der Daten verwendet werden.

10.5.3 Sensorik mittels I²C-Interface lesen

Alle Sensorik, welche sich auf dem EDU befindet, wird mit I²C angesteuert und gelesen. Dies ermöglicht eine effiziente Kommunikation und Datenübertragung zwischen den Sensoren auf dem HAT und dem Raspberry PI. Nach der Realisation verschiedener Python Skripten, die zur Auslesung von den Werten der Sensorik dienten, wurde eine Bibliothek der TU-Wien, Namens „STS1_Sensors.py“ verwendet, was diesen Prozess vereinfachte.

Eine allgemeine Information ist unter „ti.com/lit/an/slva704/slva704.pdf?ts=1712280198725&ref_url=https%253A%252F%252Fwww.google.com%252F“ zu finden.

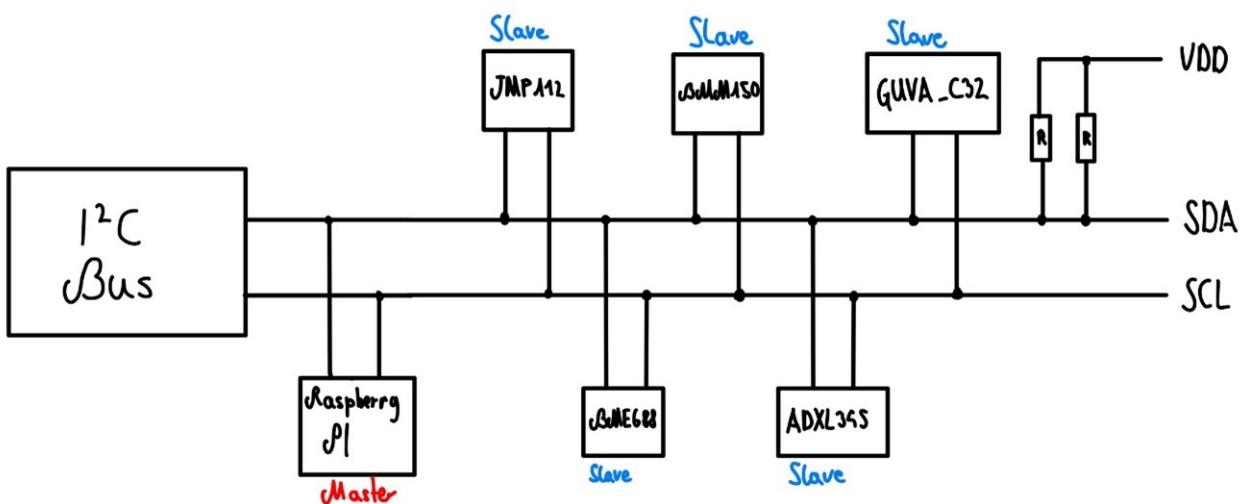


Abbildung 95: Aufbau des I²C-Systems auf dem Raspberry PI

10.6 Bibliothek „STS1_Sensors.py“

Die Bibliothek „STS1_Sensors.py“ dient zur Interaktion zwischen dem Raspberry PI und der Sensorik über den I²C-Bus. Das Python Programm verfügt über mehrere Klassen, um mit den Sensoren zu interagieren und um Daten zu lesen.

- Die Klasse „SensorsData“ speichert alle Daten, die von den Sensoren gelesen werden. Darunter sind die Temperatur, Luftfeuchtigkeit, Luftdruck, Gaswiderstand, Beschleunigung, Magnetfeldstärke und UV-Strahlung (siehe Kap.: 9.5.1). Sie ist die wichtigste Klasse für das einfache Auslesen der Sensoren.
- Die Klasse „Sensors“ ist für die Initialisierung und Deaktivierung der Sensoren verantwortlich. Auch für das Einrichten der Sensoren und das Sammeln von Daten wird sie benötigt.
- In der Bibliothek besitzt auch jeder Sensor eine eigene spezifische Klasse. Diese kann

für unterschiedliche Initialisierungsmethoden eines bestimmten Sensors verwendet werden, um Konfigurationsoptionen einzurichten oder um einen Sensor gezielt zu deaktivieren.

Da für die Teststation alle Werte der Sensorik überprüft werden müssen, eignet sich das generelle Auslesen aller Sensordaten mehr als das Auslesen eines einzelnen Sensors. Aus diesem Grund wird die Klasse „SensorData“ verwendet.

10.7 Auslesen der Sensorik mittels „STS1_Sensors.py“

Das Auslesen der Sensorik mittels der Bibliothek „STS_Sensors.py“ wird in dem Programm „SensorOut.py“ realisiert. Dazu müssen die Bibliothek der Sensoren sowie auch die Bibliothek für die I2C-Verbindung („smbus2“) in das Programm eingebunden werden.

Um die Verbindung zu ermöglichen, muss der I2C-Bus zuerst initialisiert und anschließend eine Instanz der Sensorklasse kreiert werden.

```
# I2C Bus initialisieren
bus = smbus2.SMBus(1)

# Instanz der Sensorklasse erstellen
sensors = Sensors(bus)
```

Anschließend können mit der Funktion ‚sensors.setup()‘ alle Sensoren eingestellt und vorbereitet werden. Über die Funktion ‚Sensors.getData()‘ werden alle Sensoren nach ihren Werten abgefragt. Die Klasse ‚getData()‘ beschäftigt sich mit dem Sammeln der Daten von allen Sensoren. Diese werden alle als ‚Output‘ Objekt abgespeichert.

Dieses Objekt kann mit ‚data = sensors.output‘ nun als Variable ‚Data‘ in der Hauptschleife des Programms abgespeichert werden.

Anschließend werden alle Daten in ein Dictionary namens ‚sensor_data‘ eingefügt und strukturiert dargestellt. Um dies zu ermöglichen, wird jedem Wert eine Beschreibung gegeben.

#Dictionary für die Sensorwerte erstellen

```
sensor_data = { "Temperature (TMP112)": data.tempTMP,
                 "Temperature (BME688)": data.tempBME,
                 "Humidity": data.hum,
                 "Pressure": data.press,
                 "Gas Resistance": data.gasRes,
                 "Acceleration X": data.accX,
                 "Acceleration Y": data.accY,
                 "Acceleration Z": data.accZ,
                 "Gyroscope X": data.gX,
                 "Gyroscope Y": data.gY,
                 "Gyroscope Z": data.gZ,
                 "UVA": data.uva,
                 "UVA Index": data.uvaI }
```

Anschließend werden die Daten mittels UDP an die Zieladresse (Computer) zur Visualisierung verschickt (siehe: Kap.: 10.4.1).

Ein etwa gleiches Verfahren wurde auch auf für die Sensorik der Teststation entwickelt, damit die Soll- und Istwerte des Cubesats bzw. der Teststation verglichen werden können.

In keinem der Datenblätter für die Sensoren wurde "Clock Stretching" erwähnt. "Clock Stretching"⁶² ist eine Funktion des I²C-Protokolls, die es einem Slaven ermöglicht, die Taktleitung auf "Low" zu halten, sodass der schnellere Master warten muss.

⁶² I²C Slave Mode.

11 Steuerung

11.1 Raspberry Pi

Der Raspberry Pi 4⁶³ ist ein leistungsstarker Einplatinencomputer, der über einen Quad-Core ARM Cortex-A72 Prozessor mit Geschwindigkeiten von bis zu 1,5 GHz und bis zu 8 GB LPDDR4 RAM verfügt. Der Raspberry Pi 4 bietet verbesserte Grafikfähigkeiten dank eines Broadcom VideoCore VI Grafikprozessors und unterstützt 4K-Video bei 60 Bildern pro Sekunde über HDMI. Mit Gigabit-Ethernet, WLAN, Bluetooth, USB 3.0 und USB-C-Stromversorgung bietet er eine Vielzahl von Konnektivitätsoptionen. Der Raspberry Pi 4 verfügt auch über GPIO-Pins für die Interaktion mit elektronischen Komponenten und Sensoren. Dank seiner Kompatibilität mit verschiedenen Betriebssystemen und der Unterstützung einer engagierten Community eignet sich der Raspberry Pi 4 für eine Vielzahl von Anwendungen, von Heimautomatisierung bis hin zur Entwicklung von IoT-Geräten und als preiswerte Desktop-Alternative.

Der Raspberry Pi ist perfekt für unsere DA geeignet, da er sowohl über Ein- als auch Ausgänge verfügt und sich ideal für die Steuerung über eine Website eignet. Die GPIO-Pins ermöglichen es uns, mit verschiedenen Sensoren, Aktoren und anderen externen Geräten zu interagieren, während gleichzeitig die Möglichkeit besteht, die Steuerung und Überwachung über eine Webanwendung zu realisieren. Diese Kombination aus physischer Einbindung und webbasierter Steuerung bietet die Flexibilität und Leistungsfähigkeit, die wir für die Umsetzung unserer DA benötigen.



Abbildung 96: Raspberry Pi

⁶³ Raspberry Pi 4 Modell B 4 GB RAM, Cortex-A72 64 Bit Wi-Fi Bluetooth: Amazon.de: Computer & Zubehör.

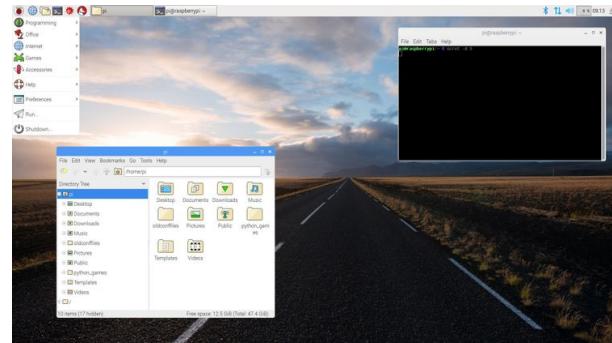
11.2 Grafische Oberfläche Raspberry Pi

Der Raspberry Pi kann nicht nur mit Befehlen über die Befehlszeile bedient werden, sondern auch über eine grafische Oberfläche. Die GUI vereinfacht zum Beispiel die Implementation von Daten, ermöglicht einen Remote-Zugriff und kann bei der Entwicklung von Anwendungen hilfreich sein.

```
pi@raspberrypi: ~
login as: pi
pi@10.0.0.106's password:
Linux raspberrypi 3.12.34+ #1 PREEMPT Sun Dec 7 22:39:06 CET 2014 armv6l
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*-copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Feb  3 08:26:14 2015 from 10.0.0.38
pi@raspberrypi ~ $ sudo su
root@raspberrypi:/home/pi#
```

(a) Befehlszeile



(b) grafische Oberfläche

Abbildung 97: Befehlszeile⁶⁴ und grafische Oberfläche⁶⁵

Die grafische Oberfläche kann mit wenigen Befehlen installiert werden.

Als erstens wird der X-Server installiert. Dieser Server ist die Grundlage um grafische Oberflächen zeichnen zu können.

```
sudo apt-get install --no-install-recommends xserver-xorg xinit
```

Nachdem der X-Server installiert wurde, kann die Desktopumgebung heruntergeladen werden. Es gibt viele verschiedene Umgebungen. Die Pixel Umgebung ist eine Desktop-Umgebung und wurde speziell auf den Raspberry Pi angepasst und eignet sich somit sehr gut für diese Anwendung.

```
sudo apt install raspberrypi-ui-mods
```

Bevor der Raspberry Pi neu gestartet wird, kann in der Konfiguration noch eingestellt werden, ob der Raspberry Pi vor dem Starten nach einem Passwort fragen soll oder nicht.

Mit dem unten angegebenen Befehl wird die Konfiguration⁶⁶ geöffnet.

⁶⁶ *Raspberry Pi*.

```
sudo raspi-config
```

Nachdem diese Wahl getroffen wurde, kann der Raspberry Pi neu gestartet werden und beim nächsten Starten wird die grafische Oberfläche sichtbar.

11.3 Remote-Zugriff

Um auf den Raspberry Pi zuzugreifen, auch wenn dieser nicht am selben Standort ist, wird ein Remote-Zugriff eingerichtet. Durch einen Remote-Zugriff, können Benutzer auf entfernte Geräte zugreifen. Es gibt eine Vielzahl von Protokollen die einen Remote-Zugriff ermöglichen. Die am häufigsten verwendeten Protokolle sind:

- Remote Desktop Protocol (RDP)
- Virtual Network Computing (VNC)
- Secure Shell (SSH)

Das Secure Shell Protokoll auch SSH genannt, sorgt für eine sichere Verbindung zwischen zwei Geräten. Das Protokoll baut eine verschlüsselte Verbindung auf und ist ebenfalls dafür zuständig, dass Daten die an den Empfänger gesendet werden, nicht manipuliert werden. Um eine SSH Verbindung aufzubauen benötigt man die Anmelde Daten und eine Schnittstelle. SSH bietet eine Kennwortauthentifizierung mit hoher Sicherheit. Es können aber auch andere Authentifizierungsmöglichkeiten verwendet werden. Die Schnittstelle kann das Terminal aber auch integrierte Optionen in VS-Code sein.

In VS-Code gibt es eine Erweiterung um einen Remote-Zugriff zu ermöglichen. Die Erweiterung **Remote-SSH** muss installiert werden, da diese nicht Standardmäßig in VS-Code vorhanden ist. Der **Remote Explorer** zeigt an, mit welchem Gerät der Laptop schon verbunden war oder mit welchen Geräten eine Verbindung aufgebaut werden kann.

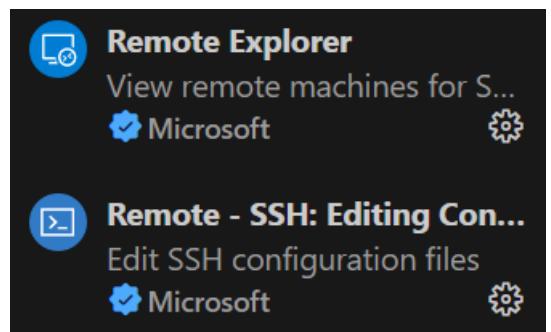


Abbildung 98: Erweiterungen in VS-Code

Im Remote Explorer kann ein neues Gerät hinzugefügt werden.

- mit dem New Remote wird ein Gerät hinzugefügt.
- Danach wird der Name des Raspberry Pi und die IP-Adresse angegeben. Zum Beispiel:

jusch@192.168.10.34

- Nach diesem Schritt wird das Passwort abgefragt.
- Wenn diese Schritte erfolgreich waren, wird eine SSH-Verbindung aufgebaut.

11.4 Fast API

Die Fast API⁶⁷ ist eine der am häufigsten verwendeten Frameworks für die Entwicklung von Webanwendungen in der Programmiersprache Python. Frameworks sind Ansammlungen von Bibliotheken, Code und anderen Entwicklungstools, sie sollen die Entwicklung von Softwareanwendungen erleichtern. Eine Fast API kann ab einer Python Version von 3.6 erstellt werden. Die API ist eine Schnittstelle die es verschiedenen Anwendungen ermöglicht miteinander zu kommunizieren und Daten auszutauschen.

Das vorher erstellte Testprogramm wird umgeschrieben und zur API hinzugefügt. Später wird die API zu einem HTML Skript hinzugefügt. Durch das HTML Skript erhalten wir eine Webapplikation. Die API ist somit die Schnittstelle zwischen dem Python Code und dem HTML Skript.

11.4.1 Erstellen einer Fast API

Zuerst wird eine virtuelle Umgebung in VS-Code erstellt. Die Befehle werden alle im Terminal eingegeben.

```
python -m venv env
```

Die virtuelle Umgebung mit dem Name `env` wurde erstellt, jedoch ist diese noch nicht aktiviert. Mit dem unten angeführten Befehl wird die Umgebung aktiviert.

```
source env/bin/activate
```

Nachdem die virtuelle Umgebung aktiviert ist, kann die Fast API installiert werden.

```
pip install fastapi
```

Die API funktioniert nicht ohne lokalen Webserver. Hierfür eignet sich der Server Uvicorn. Uvicorn⁶⁸ ist ein schneller Webserver für Python Entwicklungen.

```
pip install uvicorn
```

Um den API-Server zu starten wird der folgende Befehl in das Terminal eingegeben.

⁶⁷ Ramírez, *FastAPI*.

⁶⁸ *Uvicorn*.

```
uvicorn main:app --reload
```

In VS-Code kann ein neues Programm erstellt werden. Wichtig ist das die Fast API eingebunden wird und eine Instanz festgelegt wird.

```
from fastapi import FastAPI  
app = FastAPI()
```

Es gibt verschiedene Dekoratoren die einer Funktion zugewiesen werden können. Ein Dekorator ist eine Funktion die einem bestimmten Teil eines Codes, eine Funktionalität zuweist. Mit dem Dekorator GET können Inhalte generiert und zurückgegeben werden. Durch den Dekorator POST, können verschiedene Funktionen aufgerufen und Daten verarbeitet werden. GET und POST⁶⁹ sind die wichtigsten HTTP-Requests. Diese Requests können ganz einfach eingebunden werden:

```
app@get("/url")  
app@post("/url")
```

Abbildung 99: API-Dekoratoren

- app, ist die vorher definierte Instanz
- @get, ist der Dekorator
- ("URL"), URL wird angegeben.

Als URL wird in diesem Fall nur ein Wort genommen. Mit der URL /docs wird die Swagger UI⁷⁰ angezeigt. Auf der Swagger UI werden die vorher definierten GET- und POST-Requests dargestellt.

⁶⁹ GET vs. POST – die beiden wichtigsten HTTP-Requests im Vergleich.

⁷⁰ Features - FastAPI.

FastAPI

0.1.0 OAS 3.1

[/openapi.json](#)

default

GET	/uv	Uv
GET	/temp1	Temp1
GET	/temp2	Temp2
GET	/druck	Druck
POST	/uvlampe	Uvlampe
POST	/netzteil	Netzteil
POST	/led	Ledstreifen
POST	/motor	Motor

Abbildung 100: Beispiel Swagger UI

Die Daten des Temperatursensors werden entweder in der Swagger UI ausgegeben oder unter der URL /temp1

11.4.2 Temperatursensor

```
@app.get("/temp1")
def temp1():
    dhtDevice1 = adafruit_dht.DHT22(board.D7)
    temp_c = dhtDevice1.temperature
    humidity = dhtDevice1.humidity
    return{"Temperatur":temp_c , "Feuchtigkeit": humidity}
```

Abbildung 101: API-Programm Temperatursensor

Damit der Code funktioniert muss die vorher heruntergeladene Bibliothek eingebunden werden.

```
import adafruit_dht
```

In der Funktion *temp1* wird dem Sensor einen GPIO-Pin zugewiesen. Danach werden die Temperatur sowie die Feuchtigkeit durch eine vorgefertigte Funktion aus der Bibliothek abgefragt und ausgegeben.

11.4.3 UV-Sensor

```
@app.get("/uv")
def uv():
    i2c = board.I2C()
    ltr = adafruit_ltr390.LTR390(i2c)

    while True:
        return("UV-Index:", ltr.uvi,
              "Umgebungslicht in Lux:", ltr.light)
        time.sleep(1.0)
```

Abbildung 102: API-Programm UV-Sensor

11.4.4 Drucksensor

Der Drucksensor gibt Daten über die I2C-Schnittstelle zurück. Diese Schnittstelle wird in der Funktion initialisiert. Damit die Messungen bzw. die Umrechnungen zu einem sinnvollen Ergebnis führen, muss der mittlere Luftdruck bei Meereshöhe angegeben werden. Dieser beträgt 1013.25 hPa. Der Druck und die Meereshöhe können durch Funktionsaufrufe abgefragt werden.

```

@app.get("/druck")
def druck():
    i2c = board.I2C()
    bmp = bmp180.BMP180(i2c)
    bmp.sea_level_pressure = 1013.25
    return{"Druck": f"{bmp.pressure:.1f} hPa",
           "Meereshöhe": f"{bmp.altitude:.1f} Meter"}

```

Abbildung 103: API-Programm Drucksensor

Um Bibliothek circuitpython-bmp180 verwenden zu können, muss diese im Programm eingebunden werden. Mit der unten angeführten Codezeile wird die Bibliothek zum Programm hinzugefügt.

```
import bmp180
```

11.4.5 Schalten zwischen High und Low

Es gibt einige Komponente, die eine ähnlichen Code besitzen. Diese wurden in diesem Kapitel zusammengefasst und anhand eines Beispiels erklärt. Für diesen Teil des Codes wird keine Bibliothek benötigt, es wird nur die Tasterabfrage aus dem Kapitel 10.1 verwendet. Der unten stehende Code zieht einen Pin auf High oder auf Low. Das sorgt dafür das die Komponente ein- und ausgeschalten werden könne. Dieser Code gilt für folgende Komponente:

- UV-Lampe
- Netzteil
- Kühlung
- Lüftung

Der Code wird anhand der UV-Lampe erklärt. Wenn der Code ausgeführt wird, wird in der Swagger UI etwas zurückgegeben. Entweder UV-Lampe eingeschaltet und UV-Lampe ausgeschaltet.

```
@app.post("/uvlampe")
def uvlampe():
    global boolUV
    boolUV = not boolUV
    if boolUV == True :
        GPIO.output(PinUV,GPIO.HIGH)
        return{"UV-Lampe eingeschaltet"}
    else:
        GPIO.output(PinUV,GPIO.LOW)
        return{"UV-Lampe ausgeschaltet"}
```

Abbildung 104: API-Programm UV-Lampe

Der Code kann ganz einfach auf andere Komponente umgeschrieben werden. Die Variablen müssen nur umbenannt werden und der gewünschte Satz der ausgegeben werden soll, abgeändert werden.

Response body

```
[ "UV-Lampe eingeschalten" ]
```

(a) UV-Lampe eingeschaltet

Response body

```
[ "UV-Lampe ausgeschalten" ]
```

(b) UV-Lampe ausgeschaltet

Abbildung 105: API-Ausgabe UV-Lampe

11.4.6 Motor und Endschalter

Das Programm sieht genau gleich aus wie im Testprogramm (Kapitel 10.2.7). Zu beachten ist, dass ein Thread verwendet werden muss. Da die API den Code nur einmal ausführt. Wird jedoch ein Thread verwendet, bearbeitet er die `in range`Schleife.

11.5 Webapplikation

Dieses Kapitel beschäftigt sich mit der Visualisierung der Daten über Plotly Dash.

Um die Daten in der Applikation darzustellen, wurde zuerst eine Verbindung mittels UDP (siehe Kap. 10.5) erstellt. Somit konnte eine kabellose Verbindung zu einem Laptop hergestellt werden, welcher anschließend alle Sensorsdaten in einer CSV-Datei einspeichert (siehe Kap. 10.5.1). Auf dem Laptop wurde Plotly Dash verwendet, um alle Daten darzustellen. Dash eignet sich für solch eine Aufgabe, da über die Python Library einfach dynamische Graphen generiert werden können.

Durch die Möglichkeit, über Dash, Diagramme zu vergrößern und Zeitintervalle leicht einzustellen, können die Daten auch einfacher analysiert werden.

Für die Visualisierung der Daten werden Liniendiagramme verwendet, die die Änderung der Sensorwerte über die Zeit einfach darstellen. Um die Genauigkeit der Datenüberwachung zu verbessern, werden für verschiedene Sensoren auch Vergleichssensoren verwendet, um mögliche Fehler auf dem CubeSat zu erkennen.

11.5.1 Wie funktioniert Dash

Dash-Anwendungen⁷¹ bestehen aus verschiedenen HTML-Elementen, wie Bildern, Texten und Tabellen sowie speziellen Dash-Komponenten für interaktive Graphen und andere Visualisierungen.

Auch bietet Dash durch die Verwendung von „Callbacks“ und Funktionen die Möglichkeit interaktive Elemente darzustellen.

11.5.2 Callbacks

Sogenannte „Callbacks“ sind im Grunde genommen Funktionen, die automatisch ausgeführt werden, um auf bestimmte Aktionen des Benutzers zu reagieren, wie z.B. Klicks auf einen Button, Eingaben in ein Textfeld oder Änderungen in einem Dropdown-Menü.

Die Grundidee hinter den „Callbacks“ in Dash besteht darin, dass sie eine Verbindung zwischen den Eingabeelementen (Input), mit denen der Benutzer interagiert, und den Ausgabeelementen (Output), die aktualisiert werden sollen, herstellen.

Die „Callback-Funktion“ wird jedes Mal aufgerufen, wenn ein Input-Event ausgelöst wird. Diese Funktion führt den Code aus, der notwendig ist, um basierend auf den Eingabewerten

⁷¹ *Dash Documentation & User Guide | Plotly*.

neue Ausgabewerte zu berechnen.

Der Dekorator ‚@app.callback‘ wird in einem Dash Programm verwendet, um ein „Callback“ zu definieren. Anschließend folgt die Definition von Input- und Output-Parameter der Funktion.

```
@app.callback(
    Output('ausgabe', 'children'), #Ziel
    [Input('texteingabe', 'value')]) #Quelle
)
```

Abbildung 106: Beispiel eines Callbacks

Dash eignet sich am besten bei der Verwendung einer weiteren Python Bibliothek namens „Plotly“. Diese Bibliothek wird für die Darstellung von interaktiven Graphen in Python verwendet und ist auch das Grundgerüst der Dash Bibliothek.

11.5.3 HTML-Objekte in Dash

In Dash werden HTML-Objekte durch die Verwendung der ‚dash.html_components‘-Bibliothek eingebunden. Somit wird ermöglicht, HTML-Elemente als Python-Objekte zu erstellen und gleich wie in einer HTML-Datei zu manipulieren.

Jedes HTML-Objekt muss Teil des vordefinierten ‚Layouts‘ sein. Ein HTML-Element kann durch ‚html.‘, mit dem Namen des anschließenden Elements erstellt werden. Somit kann ein Titel, welcher in HTML mit dem Tag ‚`jh1`‘ erstellt wird, in Python mit der Codezeile ‚`html.H1`‘ erstellt werden.

Jede HTML-Komponente in Dash hat eine Reihe von Attributen, die auch den normalen HTML-Attributen entsprechen. Somit werden zum Beispiel für CSS – Klassen die Attribute ‚className‘ verwendet. Beispiel für eine HTML-Struktur in einem Python-Dash-Programm:

```
app.layout = html.Div(children=[  
    html.H1(children='Test Program'),  
    html.Div(children=''  
        Beispiel einer HTML-Struktur in Python  
    '''),  
])
```

Abbildung 107: Beispiel HTML-Struktur

11.5.4 Das Programm „dashboard.py“

Das Programm „dashboard.py“ ist für die Erstellung des interaktiven Web-Dashboards der Teststation zuständig. Das Programm liest die Sensordaten der CSV-Datei ein und stellt sie in Graphen dar. Auch wird das Dashboard periodisch aktualisiert, was für die Anzeige von live aktualisierten Daten nützlich ist.

Zu Beginn werden alle notwendigen Bibliotheken importiert. Darunter die Bibliotheken ‚Dash‘ und ‚plotly.express‘, da sie für das Web-Dashboard und für die Datenvisualisierung verwendet werden müssen. ‚Flask‘ wird als Webserver verwendet und ‚pandas‘ ist für die Datenmanipulation und Datenanalyse notwendig.

Zuerst wird der Pfad der CSV-Datei definiert, da sie die empfangenen zu visualisierenden, Daten enthält.

```
csv_file_path=' ...\\udp_data.csv' #Pfad der CSV-Datei
```

Anschließend muss der ‚Flask‘-Server und die Dash-App initialisiert werden. Dafür muss eine Instanz für die beiden Objekte erstellt werden.

```
server = Flask(__name__) # Instanz des Flask-Servers
app = dash.Dash(__name__, server=server, suppress_callback_exceptions=True)
#Instanz der Dash App
```

‘suppress_callback_exceptions = True’ erlaubt es, Callbacks für nicht sofort verfügbare IDs zu definieren. Dies ist nützlich, da viele Komponenten der Webapplikation dynamisch generiert werden.

11.5.5 Das HTML-Layout

Anschließend wird über das App Layout, die HTML-Struktur der Dash-Applikation gestaltet.

```
dcc.Interval(id='update-interval', interval=5000, n_intervals=0),  
# Update every 5 seconds  
html.Div([  
    html.Div([#Switch for Power Adapter  
        html.Div('Power Adapter', className='switch-text'),  
        dcc.Checklist(  
            id='switch-poweradapter',  
            options=[{'label': '', 'value': 'PA'}],  
            value=[],  
            inputClassName='switch-input',  
            labelClassName='switch-label'  
        )  
    ], className='switch-wrapper'),
```

Abbildung 108: Beispiel HTML-Layout

In dem Codeabschnitt lassen sich zwei verschiedene Arten von Komponenten der HTML-Struktur erkennen. Dash-Core-Komponenten (,dcc.') sind Komponente der HTML-Struktur, welche exklusiv nur mit der Dash-Library verwendbar sind. Alle ,html.'-Objekte sind normale HTML-Komponente.

Die regelmäßige Aktualisierung der Benutzeroberfläche wird auch in diesem Layout gesteuert. Durch die Codezeile ,dcc.Interval()' wird die Webseite jede 5s aktualisiert, um hinzugekommene Daten dazustellen.

Über das Objekt ,dcc.DatePickerRange()' kann eingestellt werden in welchem Zeitabschnitt die Daten angezeigt werden sollen.

```
dcc.DatePickerRange(  
    id='time-range-selector',  
    start_date_placeholder_text="Start Date",  
    end_date_placeholder_text="End Date",  
    calendar_orientation='horizontal',  
) ,
```

Abbildung 109: Beispiel DatePickerRange

Damit nach der Aktualisierung des Programmes nicht das aktuell eingegebene Zeitintervall verloren geht, wird über die „.ddc“-Komponente „dcc.store(id = ‚stored-time-range‘)“ das aktuelle Zeitintervall auf Clientseite, also auf der Seite des Benutzers, gespeichert.

Des Weiteren verfügt das HTML – Layout über verschiedene Schalter, welche zur Steuerung der Aktorik verwendet werden.

```
html.Div([
    #Switch for UV-Lamp
    html.Div('UV-Lamp', className='switch-text'),
    dcc.Checklist(
        id='switch-uvlamp',
        options=[{'label': '', 'value': 'U'}],
        value=[],
        inputClassName='switch-input',
        labelClassName='switch-label'
    )
], className='switch-wrapper'),
```

Für das Designen der HTML-Oberfläche wird jeder Komponente im Layout eine „id“ (Identifikation) hinzugefügt., da diese Identifikation für die CSS-Datei (Stylesheet) verwendet wird. Auch wird jedem Schalter eine „value“ (Wert) beigefügt. So kann abgefragt werden, ob der Knopf gedrückt wurde oder nicht.

11.5.6 Die Funktion „generate_figure“

Die Funktion dient zur Generierung der Graphen und beschreibt auch das allgemeine Designthema der Graphen, da für manche „.dcc“-Objekte keine CSS-Stylesheet verwendet werden kann.

```
def generate_figure(dataframe, x_column, y_column, title):
```

Die Funktion nimmt insgesamt vier Parameter auf:

- ,Dataframe': Das DataFrame enthält die Daten, die geplottet werden sollen
- x_column': Der Name der Spalte im DataFrame, die auf der X-Achse geplottet werden soll
- ,y_column': Der Name der Spalte bzw. eine Liste von Spaltennamen, die auf der Y-Achse geplottet werden sollen
- ,title': Der Titel des Graphen

In der Funktion wird zuerst überprüft, ob ,y_column' eine Liste ist. Falls dies der Fall ist, ist es möglich, Plots zu erstellen, die mehrere Linien generieren.

```
if isinstance(y_column, list):  
    # Check if multiple y_columns are provided for multiple traces  
    fig = px.line(dataframe, x=x_column, y=y_column, title=title)  
else:  
    fig = px.line(dataframe, x=x_column, y=y_column, title=title,  
    labels={y_column: y_column})
```

Falls 'y_column' eine Liste ist, wird ,px.line' mit der X-Spalte und den Y-Spalten als Liste aufgerufen, um mehrere Linien zu plotten.

Wenn ,y_column' kein Listentyp ist, wird ein einzelner Linienplot erstellt, wobei ,y_column' als Y-Achsenwert verwendet wird.

Anschließend wird das Layout des Graphen mit der Methode ,update_layout' aktualisiert. Hierbei werden verschiedene Eigenschaften der Figur angepasst, um ein dunkles Designthema zu erreichen, und um die Lesbarkeit des Graphen zu verbessern.

```
fig.update_layout(  
    plot_bgcolor='#002533', # Dunkler Hintergrund innerhalb  
    paper_bgcolor='#00171f', # Dunkler Hintergrund außerhalb  
    font_color='white', # Weißer Text für bessere Erkennbarkeit  
    title_font_color='white', # Titel Farbe
```

11.5.7 Die Callbacks der Website

Die Callbacks (siehe: Kap.:11.5.2) werden verwendet, um das Dash-Programm mit interaktiven Komponenten zu gestalten. Für das Programm wurden zwei Callback-Funktionen benötigt.

Das erste Callback ist für die Aktualisierung des gespeicherten Zeitbereichs zuständig. Aktualisiert werden dabei die Daten des ‚stored-time-range‘-Objekts, welche auf den vom Benutzer ausgewählten Start- und Enddatumsangaben basieren.

```
# Callback um die Graphen, basierend auf die Zeit, zu aktualisieren
@app.callback(
    Output('stored-time-range', 'data'), #Ausgang gespeichert
    [Input('time-range-selector', 'start_date'), #Eingang: Startdatum
     Input('time-range-selector', 'end_date')], #Ausgang: Enddatum
    prevent_initial_call=True #nicht beim ersten Mal laden
)
```

,prevent_initial_call = True‘ verhindert, dass das Callback beim ersten Laden der Seite aufgerufen wird, da sonst unnötig Initialisierungen stattfinden können oder ein leerer Aufruf stattfindet. Anschließend werden die Start- und Enddaten als Werte in der Funktion ‚update_stored_time_range(start_date, end_date)‘ gespeichert.

```
def update_stored_time_range(start_date, end_date):
    return {'start_date': start_date, 'end_date': end_date}
```

Ziel des zweiten Callbacks ist es, die Eigenschaften des ‚graphs-container‘- Elements zu aktualisieren, was bedeutet, dass alle Graphen basierend auf den aktuellen Daten und Einstellungen neu gerendert werden.

```
@app.callback(  
    #Ausgang: alle Eigenschaften des 'graphs-container'  
    Output('graphs-container', 'children'),  
    [Input('update-interval', 'n_intervals')], #Eingang: Timer  
    [State('stored-time-range', 'data'), #States der Schalter  
     State('switch-poweradapter', 'value'),  
     State('switch-fans', 'value'),  
     State('switch-motor', 'value'),  
     State('switch-cooler', 'value'),  
     State('switch-leds', 'value'),  
     State('switch-uvlamp', 'value')])
```

Der Eingang wird dafür durch den ‚update-interval‘-Timer gesteuert, welcher im Layout (Kap.: 11.5.5) definiert wurde. Die verschiedenen Werte der Schalter und des Anfangs- und Enddatums, welche als Zustand im Callback gespeichert wurden, bleiben bei einer Aktualisierung unverändert.

11.5.8 Die Funktion ‚update_args‘

In der Funktion ‚update_graphs‘ wird eine Liste von Graphen basierend auf einen gefilterten Datensatz generiert, der durch einen ausgewählten Zeitbereich bestimmt wird. Um dies zu verwirklichen, wird die ‚generate_figure‘-Funktion in der Funktion verwendet.

```
#Funktion für die Aktualisierung  
def update_graphs(n_intervals, stored_time_range, *args):
```

Die Funktion nimmt insgesamt 3 Parameter auf:

- n_intervals: Dieser Parameter beschreibt die Anzahl der Intervalle, seit der letzten Aktualisierung.
- stored_time_range: Ist das oben definierte Dictionary, welches den ausgewählten Zeitbereich abspeichert.
- *args: Ist eine Variable, die beschreibt das zusätzliche Argumente in der Funktion benutzt werden können, da die Funktion etwa 8 Parameter einlesen muss aber nur 2 („n_intervals“ & „stored_time_range“) wirklich benutzt werden.

In der Funktion wird zuerst überprüft, ob ein Zeitbereich ausgewählt wurde. Falls dies nicht der Fall ist, wird eine leere Liste zurückgegeben, was heißt, dass keine Daten oder Graphen angezeigt werden. Es könnten Callback Fehler vorkommen, falls dies nicht der Fall ist.

Anschließend wird über die Library Pandas die CSV-Datei, welche im oberen Teil des Codes angegeben wurde, ausgelesen. Pandas wird auch für die Umwandlung der Zeit, welche in der CSV-Datei vorhanden ist, in ein DateTime-Format verwendet.

```
df = pd.read_csv(csv_file_path)
df['Timestamp'] = pd.to_datetime(df['Timestamp'])
```

Basierend auf dem ausgewählten Zeitbereich wird der DataFrame so gefiltert, dass nur Daten zwischen ,start_date‘ und ,end_date‘ behalten werden.

```
#Filterung der Daten basierend auf dem ausgewählten Zeitpunkt
if start_date and end_date:
    filtered_df = df[(df['Timestamp'] >= start_date) &
                      (df['Timestamp'] <= end_date)]
else:
    filtered_df = df
```

Nun werden die benötigen Graphen generiert. Für manche Graphen wird dafür auch mehr als nur eine Datenreihe verwendet.

```
dcc.Graph(figure=generate_figure(filtered_df, 'Timestamp',
['Gyroscope X', 'Gyroscope Y', 'Gyroscope Z'], 'Gyroscope')),
```

Die Funktion ‘generate_figure’, wird benutzt, um aus der entsprechenden Datenreihe eine Figur, bzw. einen Graphen zu erstellen. Dabei werden der gefilterte DataFrame, der Name der ,Timestamp‘, der Name der Datenspalte und der Titel für den Graphen übergeben.

11.5.9 Design der Webapplikation

Die Webapplikation wurde mithilfe von eines CSS-Skript gestaltet und sollte ein dunkles Aussehen haben. Dafür wurden die Farben Schwarz, Weiß, Blau und Grau verwendet.

11.5.10 Steuerung von Aktoren

Um eine Verbindung vom Laptop zum Raspberry PI zu erstellen, um somit alle Aktoren der Teststation zu steuern, muss eine einfache und schnelle Verbindung zwischen den beiden Endgeräten hergestellt werden.

11.5.11 MQTT

Das MQTT-Protokoll⁷² basiert auf einem Publish-Subscribe Modell. In diesem Modell gibt es zwei verschiedene Teilnehmer:

- Der MQTT-Client ist eine Anwendung oder ein Gerät, das MQTT-Nachrichten über das Netzwerk (lokal oder global) publiziert (sendet) oder abonniert (empfängt). Sie sind auch für das Erstellen der Verbindung zum Client zuständig.
- Ein MQTT-Broker verwaltet die Nachrichtenübertragung zwischen den MQTT-Clients. Der Broker empfängt alle Nachrichten, die von den Clients publiziert werden und filtert diese Nachrichten basierend auf den Themen, zu denen sich andere Clients abonniert haben. Anschließend leitet er diese Nachrichten entsprechend weiter.

Die Korrelation zwischen Client, Broker und Subscriber ist in Abb. 110 veranschaulicht.

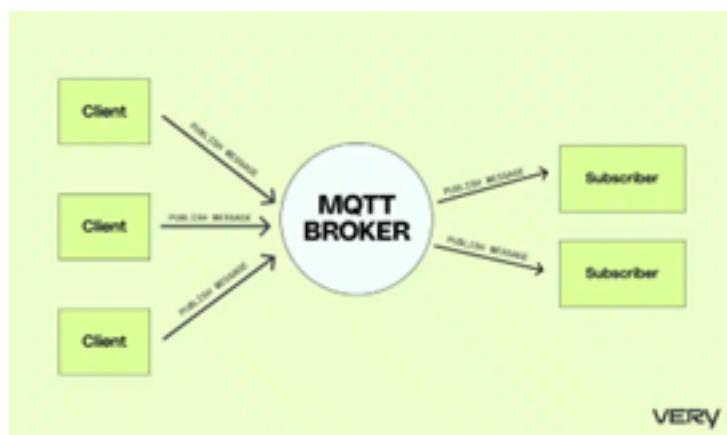


Abbildung 110: Korrelation⁷³

In der Abbildung ist der MQTT-Broker ein reiner Vermittler. Bei der Verbindung zwischen Laptop und Raspberry PI ist jedoch der Einplatinen-Computer sowohl der MQTT-Broker als auch der Subscriber. Der Client ist der Laptop/Computer, welcher Nachrichten über ein

⁷² Mai, *MQTT für Dummies*.

bestimmtes Teilgebiet (Topic) veröffentlicht und weiterschickt. Diese Routine dient der Ansteuerung von Aktoren auf dem Raspberry PI, der Benutzer kann den Zustand der verschiedenen Komponenten von dem Laptop aus über ein Dashboard steuern.

11.5.12 Fast API

Schlussendlich wurde anstatt des MQTT-Protokolls die Fast API verwendet (siehe: Kap: 11.4)

Die Fast API bietet einen leistungsstarken und flexiblen Weg, Aktoren über eine Webanwendung zu steuern. Im Vergleich zu MQTT kann die Nutzung einer API in Bezug auf Netzwerkeffizienz und native Unterstützung von IoT-Geräten eingeschränkt sein. Die API eignet sich für Situationen, in denen komplexe Anfragen zur dynamischen Generierung von Antworten erforderlich sind. MQTT wird hingegen vor allem in Umgebungen eingesetzt, in denen die Effizienz der Nachrichtenübermittlung und der geringe Ressourcenverbrauch im Vordergrund stehen.

12 Testen

12.1 Testroutine

12.1.1 Rotationstest

Der CubeSat, ein kleiner würfelförmiger Satellit, kann in einem Gyroskop befestigt werden, um die Stabilität und die Reaktionsfähigkeit des Satelliten auf Rotationsbewegungen in X-, Y-, und Z-Richtung zu testen.

Diese Rotationstests sind von wesentlicher Bedeutung, da sie es ermöglichen zu überprüfen, inwiefern der Satellit fähig ist, den komplexen Belastungen durch Rotation sowie anderen Bewegungen, denen er im Weltraum ausgesetzt ist, standzuhalten. Der Weltraum stellt eine äußerst anspruchsvolle Umgebung dar, in der Satelliten dynamischen Bewegungen, die insbesondere während des Manövrierens oder der Ausrichtung des Satelliten auftreten. Ein Schaden durch unkontrollierte oder nicht korrekt abgefangene Rotationsbewegungen könnte die Funktionstüchtigkeit des Satelliten erheblich beeinträchtigen und somit seine Mission gefährden. Durch die sorgfältige Durchführung dieser Tests vor dem Einsatz des Satelliten im Orbit lässt sich sicherstellen, dass dieser während seiner gesamten Einsatzzeit zuverlässig funktioniert und seine vorgesehenen Aufgaben erfüllen kann.

12.1.2 Vibrationstest

Der Vibrationstest ein unverzichtbarer Schritt, der dazu dient, die Widerstandsfähigkeit und Robustheit des Satelliten gegenüber den extremen Bedingungen während des Raketenstarts zu bewerten. Für diesen Test wird der CubeSat auf einer speziell konstruierten Rüttelplatte befestigt, die in der Lage ist, die intensiven Vibrationen zu simulieren, denen der Satellit beim Start ausgesetzt ist.

Diese Tests sind kritisch, da sie dazu beitragen, eventuelle Schwachstellen in der Struktur des CubeSats oder in seinen einzelnen Komponenten zu identifizieren, bevor er seine Mission im Weltraum antritt. Während des Vibrationstests werden nicht nur die strukturelle Integrität und die mechanische Belastbarkeit des CubeSats auf die Probe gestellt, sondern auch die Zuverlässigkeit der Sensoren auf dem CubeSat und anderer kritischer Systeme.

Die Erkenntnisse aus diesen Tests sind entscheidend, um festzustellen, wie gut der Satellit und seine Sensoren in der Lage sind, den Startbedingungen standzuhalten, ohne ihre Funktionsfähigkeit zu verlieren. Sollten die Tests Schwachstellen aufdecken, müssen notwendige Anpassungen vorgenommen werden. Diese können von der Verstärkung struktureller Komponenten bis hin zur Überarbeitung der Montage oder des Designs der Sensoren reichen, um

die Missionstauglichkeit des CubeSats sicherzustellen. Letztendlich zielen Vibrationstests darauf ab, das Risiko eines Ausfalls während des kritischen Startvorgangs zu minimieren und die Erfolgschancen der Mission zu maximieren.

12.1.3 Test im Vakuum

Ein weiterer Test umfasst die Kombination von Vibrationstests mit der Simulation des Vakuums des Weltraums. Dies wird erreicht, indem der CubeSat auf einer Rüttelplatte platziert und zusätzlich eine Vakuumglocke verwendet wird. Diese Konfiguration ermöglicht es, gleichzeitig die mechanischen Belastungen des Starts und die Bedingungen des Weltraumvakuums zu simulieren.

Die Simulation des Weltraumvakuums ist kritisch, da sie Aufschluss darüber gibt, wie der Satellit und seine integrierten Systeme unter dem Einfluss eines nahezu vollständigen Vakuums operieren. Im Weltraum herrscht ein Druckniveau, das nahezu bei null liegt, was bedeutet, dass alle Materialien und Komponenten, die in der irdischen Atmosphäre entwickelt und getestet wurden, sich dort anders verhalten können. Insbesondere wird untersucht, wie gut die Materialien und elektronischen Komponenten des Satelliten diese Bedingungen tolerieren, ohne ihre Integrität oder Funktionalität zu verlieren.

Ein zentrales Anliegen dieser Tests ist die Überprüfung auf mögliche Lecks, die im Vakuum schneller auftreten können als unter irdischen Bedingungen.

12.1.4 Einsatz von Sensoren

Die Durchführung von Tests unter Einsatz von Sensoren, sowohl auf dem CubeSat selbst als auch in der Testumgebung, spielt eine entscheidende Rolle bei der Vorbereitung des Satelliten auf seinen Einsatz im Weltraum. Diese ermöglichen es, eine Vielzahl von Daten zu erfassen, die für die Bewertung der Leistungsfähigkeit und Zuverlässigkeit des CubeSats unter verschiedenen Simulationen bzw. Tests unerlässlich sind.

12.1.5 UV-Lampe

Um die anspruchsvollen und vielfältigen Bedingungen des Weltraums möglichst realitätsnah zu simulieren, ist es unerlässlich, neben Vakuum und Vibration auch die Auswirkungen der Sonnenstrahlung sowie die extremen Temperaturschwankungen, denen ein Satellit im Orbit ausgesetzt sein wird, zu testen. Die Verwendung einer UV-Lampe ermöglicht es, die intensive ultraviolette Strahlung der Sonne nachzubilden, der der CubeSat im Weltraum kontinuierlich ausgesetzt sein wird. Diese Simulation ist von entscheidender Bedeutung, da UV-Strahlung das Potenzial hat, elektronische Systeme zu beeinträchtigen und somit die Leistung und Lebensdauer des Satelliten signifikant zu reduzieren.

Zur Simulation der extremen Temperaturbedingungen, die im Weltraum herrschen, kommt ein Kühlgerät zum Einsatz. Im Weltraum durchläuft ein Satellit drastische Temperaturschwankungen.

Er ist hohen Temperaturen ausgesetzt, wenn er direkt der Sonne zugewandt ist, und erlebt gleichzeitig niedrige Temperaturen, wenn er sich im Schatten befindet. Diese Temperaturschwankungen können eine signifikante Belastung für die Struktur des Satelliten, seine Systeme und die an Bord befindlichen Instrumente darstellen. Materialien können sich ausdehnen oder zusammenziehen, was die strukturelle Integrität beeinträchtigen und zu Funktionsstörungen führen kann. Elektronische Komponenten sind besonders anfällig für Temperaturänderungen, die ihre Leistung beeinträchtigen oder sogar zum vollständigen Ausfall führen können.

Die Durchführung dieser Tests unter kontrollierten Bedingungen auf der Erde ist entscheidend, um sicherzustellen, dass der CubeSat und seine Systeme den extremen Bedingungen im Weltraum standhalten können.

12.1.6 Fazit der Tests

Die sorgfältige Durchführung dieser vielfältigen Tests ist von fundamentaler Bedeutung für die Vorbereitung des CubeSats auf seine Mission im Weltraum. Indem sie eine breite Palette an Umweltbedingungen simulieren, die im Weltraum vorherrschen, tragen diese Tests entscheidend dazu bei, die Zuverlässigkeit, Leistungsfähigkeit und Missionstauglichkeit des CubeSats zu gewährleisten.

12.2 Messungen

12.2.1 Erklärung der Daten

Um die Daten darzustellen und gegebenenfalls zu analysieren, müssen alle Daten zuerst fest definiert werden. Dazu sollte der Nutzen der Daten erklärt und alle Einheiten klargemacht werden.

12.2.2 Temperatur

Auf dem EDU-Hat gibt es zwei Sensoren, welche akkurat die Temperatur des Raumes messen können. Der erste Sensor (TMP112) ist ein reiner Temperaturmesser, welcher die Temperatur in Grad Celsius zurückgibt. Auch der Gassensor (BME688) gibt die Temperatur akkurat in Grad Celsius an.

Das Messen der Temperatur ist für mögliche Experimente im All sehr wichtig. Eine Messung der Temperaturveränderung des CubeSats, um die Auswirkungen der Sonneneinstrahlung zu studieren, wäre hier ein gutes Experiment.

Weitere Informationen über die Sensoren können in Kapitel 9.5.3 gefunden werden.

12.2.3 Messung des Magnetfeldes

Der Magnetfeldsensor (BMM150 siehe Kap.: 9.5.4) gibt seine Werte in einem kartesischen Koordinatensystem aus. Alle X, Y und Z Werte werden als μT (Micro-Teslas) zurückgegeben. Tesla ist die standardisierte Einheit für die magnetische Flussdichte. Das Magnetfeld der Erde besitzt am Äquator etwa $30 \mu\text{T}$,⁷⁴ während ein durchschnittlicher Magnet etwa 1T-1,5T besitzt.

Der Magnetfeldsensor dient zur Messung der Rotation des Satellites. Auch können mögliche magnetische Störungen vom Sensor, wie zum Beispiel magnetische Stürme oder das Magnetfeld der Erde, aufgenommen werden.

Signifikante Messergebnisse können möglicherweise durch den Einfluss eines externen Magneten in der Teststation gemessen werden.

⁷⁴ Erdmagnetfeld – Wikipedia.

12.2.4 Messung von Beschleunigung

Wie in Kapitel 9.5.5 schon erwähnt, werden die Werte des Beschleunigungssensors (ADXL345) in G-Kräften angegeben. Ein G entspricht der Erdanziehungskraft, während zum Beispiel 6G der sechsfachen Erdanziehungskraft entsprechen.

G-Kräfte können sich jedoch nicht nur auf die Gravitationskraft der Erde beziehen. Die Größe kann auch für die Beschleunigung eines Objekts verwendet werden.

Obwohl in der Teststation nur gyrokopische Effekte auf den Satelliten wirken, können eventuell mögliche Erschütterungen durch eine eingebaute Rüttelplatte gemessen werden. Sonstige Bewegungen müssen außerhalb der Teststation simuliert werden.

12.2.5 Messung der UV-Strahlung

Der UV-Sensor (GUVA_C32) welcher für die Messung einkommender UV-Strahlung verwendet wird, gibt bei der Auslesung zwei wichtige Werte zurück.

Der Wert UVA steht für Ultraviolettrstrahlung vom Typ A. UVA-Strahlen haben eine Wellenlänge von 320-400 Nanometer.

Der UV-Index ist ein internationaler Index für die Messung der Stärke von UV-Strahlen. Der UV-Index berücksichtigt sowohl UVA-, als auch UVB-Strahlen und wird in einem Index von 0 bis 11+ skaliert, wobei 0 die niedrigste Stärke darstellt.

UVB-Strahlen stehen für Ultraviolettrstrahlung vom Typ B. Diese Strahlen haben eine Wellenlänge von 280-320 Nanometer und sind hauptverantwortlich bei der Entstehung von Sonnenbrand und Hautkrebs.

Der Sensor ist besonders wichtig, um die Sonneneinstrahlung im Weltraum, wo die Strahlungsintensität um einiges höher ist als auf der Erdoberfläche, zu messen.

12.2.6 Messung der Luftfeuchtigkeit

Für die Messung der Luftfeuchtigkeit ist der Gassensor (BME688) zuständig. Die Luftfeuchtigkeit ist der Anteil des gasförmigen Wassers, welcher sich in einem Raum, bzw. in der Atmosphäre, befindet. Dieser Anteil wird in Prozent angegeben und korreliert mit der Temperatur.

Der Feuchtigkeitssensor kann verwendet werden, um die Menge des Wasserdampfes in der oberen Atmosphäre zu messen. Somit kann der Sensor für verschiedene meteorologische Versuche verwendet werden.

12.2.7 Messung des Luftdrucks

Der Luftdruck wird ebenfalls über den BME688 gemessen. Der Luftdruck ist definiert als das Gewicht der Luft in der Erdatmosphäre. Je nach Höhe des Messvorgangs kann sich dieser Wert verändern. Der Luftdruck nimmt mit zunehmender Höhe des Satelliten ab. Der Luftdruck wird in der Einheit des Drucks (Pa - Pascal) angegeben. Bei der Testung des Systems sollte der Luftdruck bei einer Höhe von 461m (Höhe Rankweil) und bei einer Raumtemperatur von 25°C etwa 96080Pa sein.

Solch ein Sensor kann im Sensor verwendet werden, um Höheninformationen zu ermitteln. Auch können Änderungen des Luftdrucks wichtig für die Vorhersage von anstehenden Wetterveränderungen sein.

12.2.8 Messung der Luftqualität

Der Gassensor (BME688) kann auch die Luftqualität in einem bestimmten Raum, bzw. Gebiet messen. Dazu wird der Wert der Gasbeständigkeit genutzt und in Ohm angegeben. Je höher die Gasbeständigkeit in einem Raum ist, desto besser ist die allgemeine Luftqualität.

Somit kann der Sensor auch für die Überprüfung der Carbon Emissionen von bestimmten Gebieten genutzt werden.

12.2.9 Messergebnisse der Sensoren

Nachdem die Sensordaten erfolgreich über die Webanwendung (siehe Kapitel: 11.5) ausgegeben und in einer visuell ansprechenden Weise dargestellt wurden, besteht die Möglichkeit, die Sensorik des EDUs eingehend zu prüfen. Diese Prüfung zielt darauf ab, die Funktionalität der Sensorik unter variierenden Umweltbedingungen zu testen.

12.2.10 Messergebnis Temperatur

Beide Temperatursensoren messen die Temperatur bei einer Raumtemperatur von 28°C. Nach der Belüftung des Raumes, in dem sich der EDU befindet, wird erwartet, dass die von den beiden Temperatursensoren erfasste Temperatur sinkt. Dies ist eine direkte Folge der Abkühlung, die durch den Luftaustausch während des Lüftens verursacht wird.

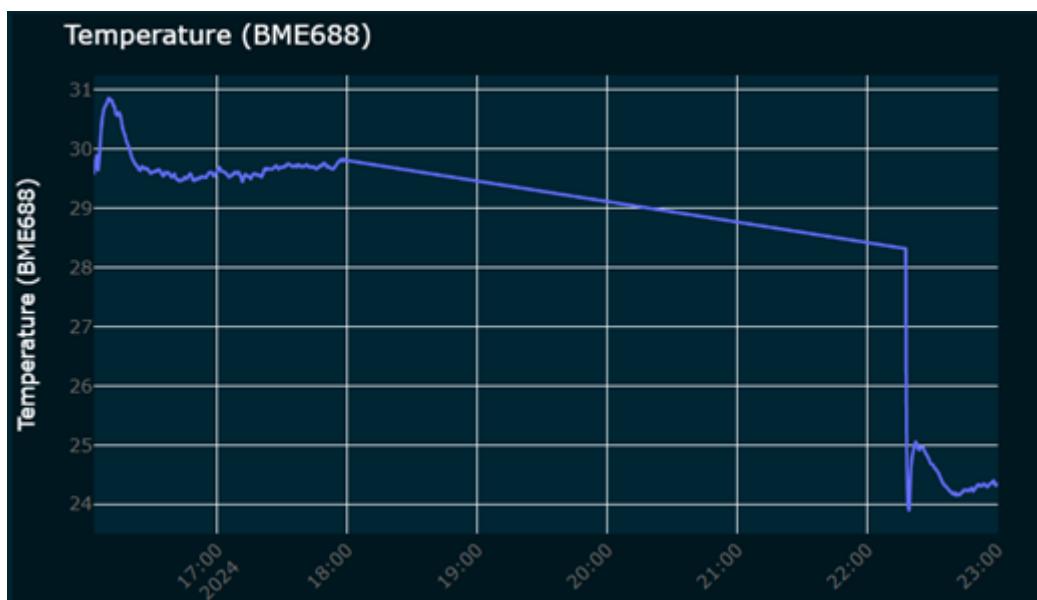


Abbildung 111: Messung Temperatur

Zu sehen ist der Temperaturunterschied des Sensors BME688 (Kap.:9.5.3). Links im Diagramm ist die Raumtemperatur zu sehen. Anschließend wird das Fenster des Raumes geöffnet und die Temperatur sinkt. Die linear verlaufende Linie, die zwischen den zwei Temperaturunterschieden zu sehen ist, kennzeichnet, dass in diesem Zeitraum keine Datenaufnahme stattgefunden hat.

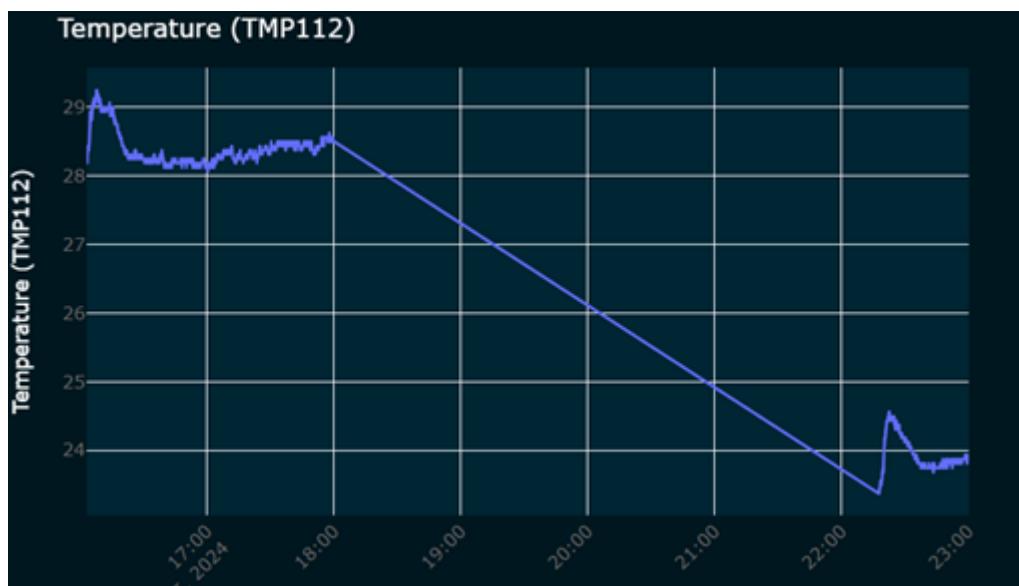


Abbildung 112: Messung Temperatur

Auch der Temperatursensor TMP112 zeigt das gleiche Verhalten wie der obere Sensor. Jedoch sind die Messwerte des reinen Temperatursensors um ca. $1,5^{\circ}\text{C}$ kälter als die Temperaturwerte des Gassensors (BME688).

12.2.11 Messergebnis Luftfeuchtigkeit

Die Luftfeuchtigkeit kann durch die Verwendung eines Luftbefeuchters, welcher mit Hilfe von Wasserdampf die Luft feuchter macht, beeinflusst werden.

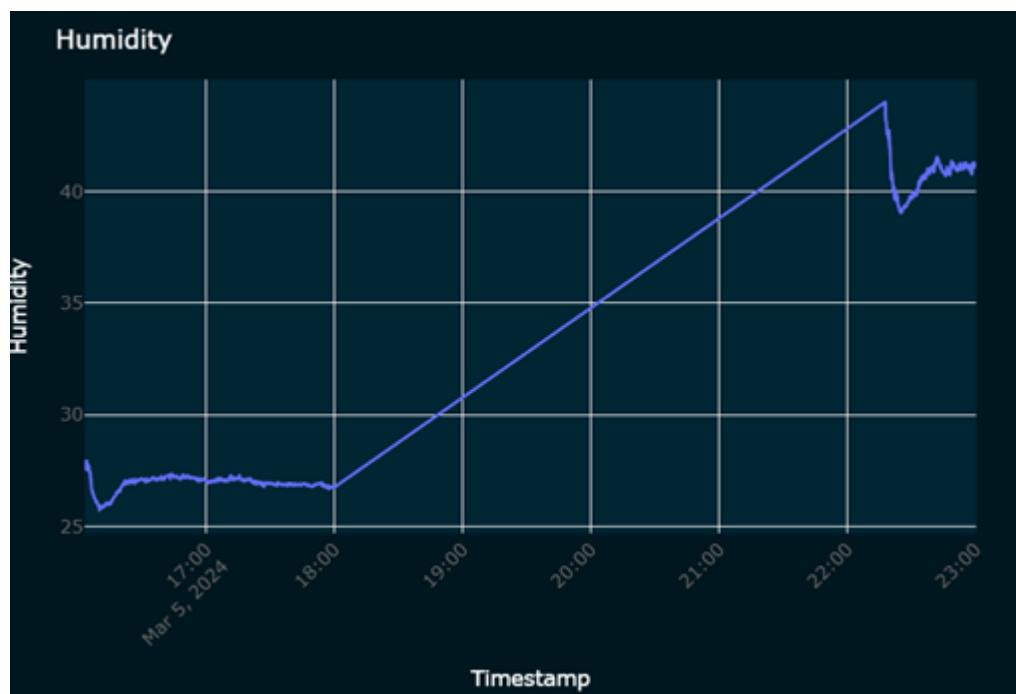


Abbildung 113: Messung Feuchtigkeit

Hier ist die Veränderung der Luftfeuchtigkeit über die Zeit zu sehen. Zu Beginn der Datenaufnahme wurde die normale Luftfeuchtigkeit des Raumes ohne äußere Einflüsse gemessen (Wert = 20%). Anschließend wurde der Luftbefeuchter eingeschaltet, was die Luftfeuchtigkeit des Raumes auf 40% erhöhte.

12.2.12 Messergebnis Luftdruck

Der Luftdruck eines Raumes kann verändert werden, indem die Temperatur des Raumes verändert wird. Umso höher die Temperatur eines Raumes ist, desto größer ist der Luftdruck des Raumes. Somit kann der Luftdruck auf die gleiche Weise wie die Temperatur beeinflusst werden.

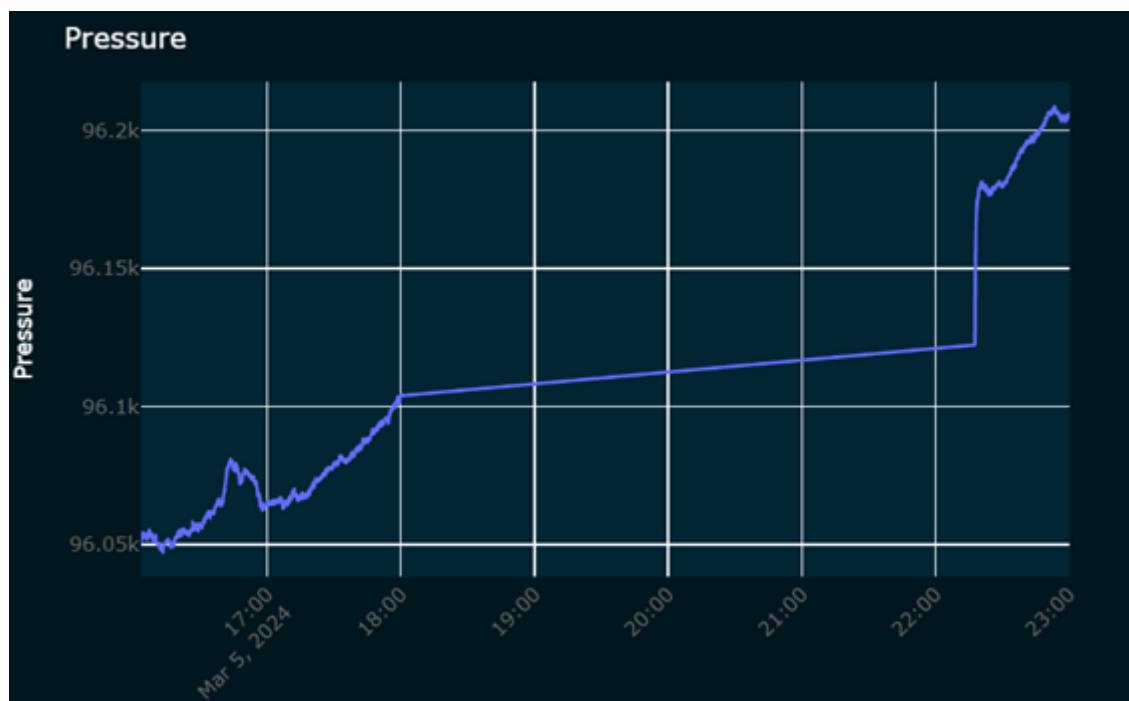


Abbildung 114: Messung Druck

Oben zu sehen ist die Veränderung des Luftdrucks über die Zeit. Die Korrelation des Luftdrucks und der Temperatur ist dabei gut zu sehen.

Der Luftdruck kann mit folgender Formel berechnet werden:

$$P_{abs} = P_{rel} \times \left(\frac{T_0 - 0.0065 \times H}{T_0} \right)^{5.255}$$

Abbildung 115: Formel Druck

Dabei ist:

- P_{abs} der[H] absolute Luftdruck
- P_{rel} der relative Luftdruck
- H die Höhe in Metern über dem Meeresspiegel
- T_0 die Standardtemperatur auf Meereshöhe in Kelvin

Die Messung wurde in Klaus, mit einer Höhe von etwa 507 m. ü. A. (Meter über Adria) durchgeführt. Als relativer Luftdruck wird der Luftdruck am Meeresspiegel verwendet (1013.25 hPa). Anschließend kann der Luftdruck berechnet werden:

$$P_{abs} := P_{rel} \cdot \left(\frac{T_0 - 0.0065 \text{ Alt}}{T_0} \right)^{5.255} = 955.575 \text{ hPa} \quad \dots \text{Luftdruck bei } 24^\circ\text{C}$$

$$P_{abs2} := P_{rel} \cdot \left(\frac{T_{02} - 0.0065 \text{ Alt}}{T_{02}} \right)^{5.255} = 956.323 \text{ hPa} \quad \dots \text{Luftdruck bei } 28^\circ\text{C}$$

Abbildung 116: Druckberechnung

Der berechnete Wert und der gemessene Wert weichen um ca. 5 hPa ab. Der Grund dafür ist, dass die Luftfeuchtigkeit, welche für die Messung künstlich erhöht wurde, in der Rechnung nicht berücksichtigt wurde.

12.2.13 Messergebnis des UV-Sensors

Eine Veränderung der UV-Strahlung lässt sich durch langes Messen (ca. ein Tag) gut beobachten. Vor allem lässt sich die Veränderung der UVA-Strahlung (siehe: Kap.: 12.2.5) erkennen. Die Messung des UV-Index gestaltet sich zu dieser Jahreszeit (stand: 5.03.2024) jedoch schwierig.

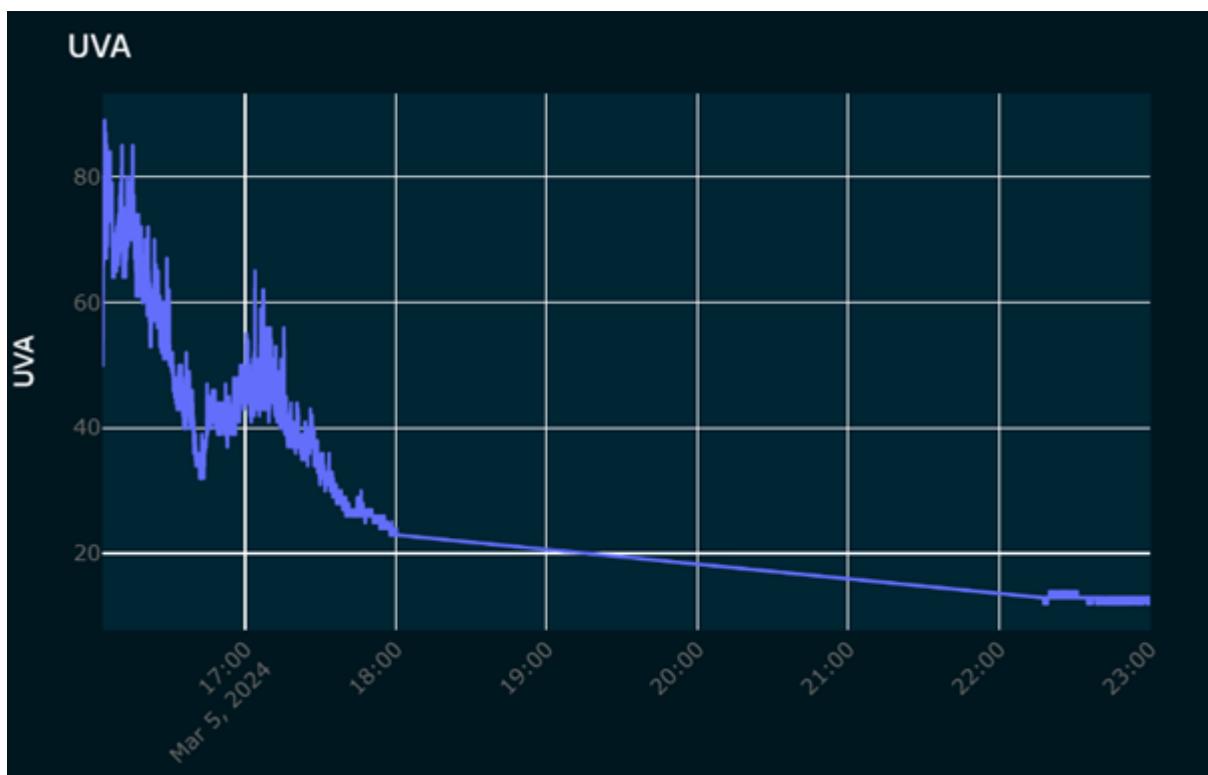


Abbildung 117: Messung UV

Zu sehen ist die Veränderung der UVA-Strahlung über eine Zeit von 7 Stunden. Die Sonne ging zu dieser Zeit des Jahres um etwa 18:30 Uhr unter, was auch bedeutet, dass die Sonnenstrahlung schwächer wird.

12.2.14 Messergebnis der Geschwindigkeit und des Gyroskops

Sowohl das Gyroskop als auch der Geschwindigkeitssensor können auf die gleiche Weise getestet werden. Durch Drehung des Sensorbretts oder des CubeSats kann eine Bewegung durch die Schwerkraft simuliert werden.

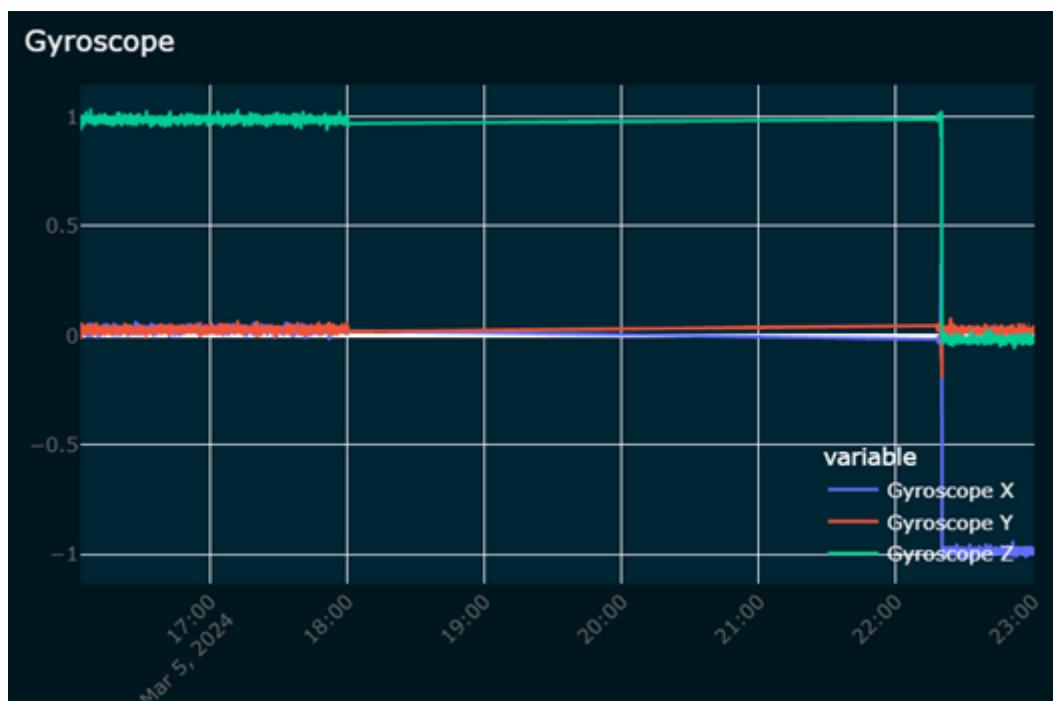


Abbildung 118: Messung Gyroskop

Hier sehen Sie den gyroskopischen, magnetischen Sensor BMM150. Es ist zu erkennen, dass für die gyroskopische Messung drei Achsen verwendet werden. Zunächst ist der Wert Z auf eins, was auf die Orientierung des Sensors in Richtung der Z-Achse hinweist. Anschließend um etwa 22:30 Uhr wurde der CubeSat auf seine X-Achse gestellt.

13 Dokumentationsaufteilung

Unten wird die Aufteilung der Dokumentation veranschaulicht. Im Kapitel 14.3 Arbeitsaufteilung ist zu erkennen welches Teammitglied für was verantwortlich war. Dementsprechend musste jedes Teammitglied diesen Teil dokumentieren.

13.1 Sophia Hagen

- Kapitel 7 Pflichtenheft
- Hardware
 - Kapitel 17.1.2 Eagle 7.7.0
 - Kapitel 9.4 Sensoren in der Teststation
 - Kapitel 9.6 UV-Lampe
 - Kapitel 9.7 Magnetfeld
- Software
 - Kapitel 17.2.2 Visual Studio Code
 - Kapitel 10.1 Tasterabfrage
 - Testprogramm
 - * Kapitel 10.2.3 Temperatursensor
 - * Kapitel 10.2.4 UV-Sensor
 - * Kapitel 10.2.5 Drucksensor
 - * Kapitel 10.2.6 UV-Lampe
- Steuerung
 - Kapitel 11.2 Grafische Oberfläche Raspberry Pi
 - Kapitel 11.3 Remote Zugriff
 - Kapitel 11.4 Fast API
- Kapitel 14 Projektmanagement

13.2 Julius Scherrer

- Hardware
 - Kapitel 17.1.1 Solidworks
 - Kapitel 9.1 Gehäuse
 - Kapitel 9.2 Gyroskop
 - Kapitel 9.2.4 Relais
 - Kapitel 9.8 Pneumatik
 - Kapitel 9.9 LED-Streifen
 - Kapitel 9.14 Sicherheit

- Software
 - Kapitel 17.2.1 Thonny
 - Testprogramm
 - * Kapitel 10.2.1 GUI
 - * Kapitel 10.2.2 Button
 - * Kapitel 10.2.7 Motor
 - * Kapitel 10.2.11 Notaus
 - * Kapitel 10.2.9 LED-Streifen
 - * Kapitel 10.2.12 Endschalter
- Steuerung
 - Kapitel 11.1 Raspberry Pi

13.3 Constantin Zumtobel

- Hardware
 - Kapitel 9.5 Sensoren auf dem CubeSat
- Software
 - Kapitel 17.2.3 Plotly Dash
 - Kapitel 17.2.4 TeraTerm
 - Kapitel 10.3 Verbindungsmöglichkeit
 - Kapitel 10.4 Übertragungsmethode
 - Kapitel 10.5 Empfangsroutine
 - Kapitel 10.6 Bibliothek „STS1_Sensors.py“
 - Kapitel 10.7 Auslesen der Sensorik mittels „STS1_Sensors.py“
- Steuerung
 - Kapitel 11.5 Webapplikation
- Testen
 - Kapitel 12.2 Messungen

13.4 Sebastian Bellai

- Hardware
 - Kapitel 9.10 CubeSat
 - Kapitel 9.11 Halterung
 - Kapitel 9.12 Lüfter
 - Kapitel 9.13 Kühlung
- Software
 - Testprogramm
 - * Kapitel 10.2.13 Kühlung
 - * Kapitel 10.2.14 Lüfter
- Testen
 - Kapitel 12.1 Testroutine

14 Projektmanagement

Sophia Hagen hat die Rolle der Projektleiterin übernommen. Jedoch konnte jeder bei der Arbeitsaufteilung sowie bei der zeitliche Einteilung mitbestimmen.

14.1 Projektmanagementtool

Für das Projektmanagement wurden zwei Programme verwendet. Diese Programme wurden genutzt, um die Arbeit im Blick zu behalten und die Unterlagen für alle zugänglich zu machen.

14.1.1 GanttProject

GanttProject⁷⁵ wurde verwendet, um einen Projektplan zu erstellen. Es wurden verschiedene Arbeitspakete definiert, die zu einem bestimmten Datum abgeschlossen werden müssen. In GanttProject können verschiedene Teammitglieder den Arbeitspaketen zugewiesen werden. So entsteht das Ressourcendiagramm. Im Ressourcendiagramm wird die Auslastung einer Person oder einer Ressource über eine Zeitspanne dargestellt. Die erstellte File ist in der GitHub Repository.

14.1.2 GitHub

Auf GitHub wurde eine Repository für die Diplomarbeit erstellt. So hat jedes Teammitglied Zugriff auf die verschiedenen Files. In der Repository sind vier Ordner zu finden.

- firmware - Testprogrammfile
- api - API-File
- Projektmanagment - Gantt-Projekt
- Dokumentation

⁷⁵ GanttProject: Free Project Management App.

Die Repository kann erreicht werden, indem der unten angeführte QR-Code gescannt wird.



Abbildung 119: QR-Code GitHub Repository

14.2 Einteilung des Projektes

Die Diplomarbeit kann in vier verschiedene Teile aufgeteilt werden:

- Hardware
- Software
- Steuerung
- Testen

14.3 Arbeitsaufteilung

Sophia Hagen	Julius Scherrer	Constantin Zumtobel	Sebastian Bellai
Projektleitung	Hardware	Hardware	Hardware
Hardware	Software	Software	Software
Software		Steuerung	Testen
Steuerung		Testen	

14.4 Arbeitszeiten

Die Arbeitszeiten wurden in Excel dokumentiert. Es wurde ein Kalender geführt, indem die Anzahl der Stunden und eine Beschreibung eingetragen wurden. In den unten abgebildeten Abbildungen werden die Arbeitsstunden aufgelistet. Es werden folgende Sachen mit den Abbildungen abgedeckt:

- Arbeitsstunden pro Monat
- Gesamte Arbeitsstunden
- Anzahl der Stunden Zuhause und in der Schule

Name	September	Oktober	November	Dezember	Januar	Februar	März
Sophia	12	21	8	20	28.5	76.5	54
Julius	16	40	24	32	24	23	31,5
Constantin	12	15	15	12	23	24	64
Sebastian	16	26	34	25	18	31	11

Tabelle 17: Stunden pro Monat

Name	Schule	Zuhause
Sophia	80	140
Julius	88	102,5
Constantin	58	107
Sebastian	65	98

Tabelle 18: Stunden Zuhause und in der Schule

Name	Stunden	Prozentanteil
Sophia	220	30%
Julius	190,5	26%
Constantin	165	22%
Sebastian	163	22%
Gesamt	738,5	100%

Tabelle 19: Gesamte Stunden

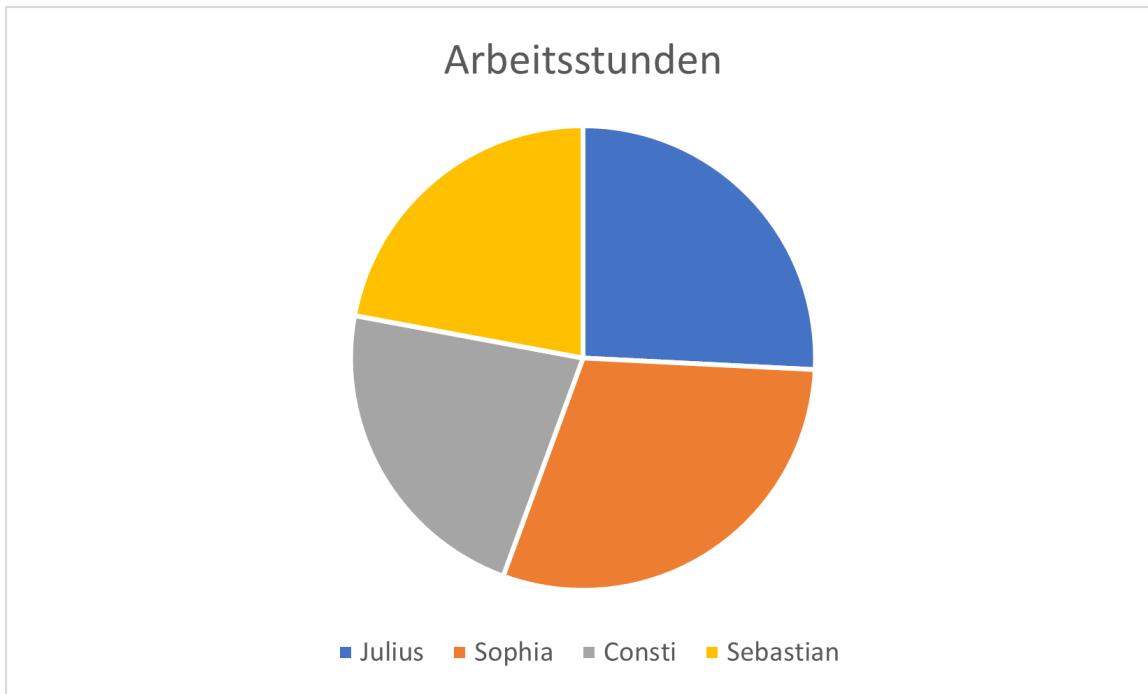


Abbildung 120: Aufteilung Arbeitsstunden

15 Probleme

In Projekten treten oft unvorhersehbare Probleme und Herausforderungen auf, die den geplanten Ablauf stören können. Trotz sorgfältiger Planung und Vorbereitung können externe Faktoren oder Änderungen im Projektumfeld auftreten, die zusätzliche Anpassungen erfordern. Die Fähigkeit, flexibel zu reagieren, Probleme proaktiv anzugehen und kreative Lösungen zu finden, ist entscheidend, um den Projekterfolg sicherzustellen. Durch eine offene Kommunikation im Team, klare Priorisierung von Aufgaben und die Fähigkeit, sich an veränderte Umstände anzupassen, können Projekte trotz unvorhersehbarer Hindernisse erfolgreich abgeschlossen werden.

Probleme während unserer Diplomarbeit:

- 3V3 Ausgang von Raspberry Pi
 - Nicht gewusst, Motor Treiber braucht 5V, Motor nicht funktioniert

Problem unter Kapitel 9.2.1 5V Steuerung gelöst.

- Der Magnetsensor befindet sich nicht in der Testkammer, da der ADC nicht liefert werden konnte.

16 Verzeichnisse

Abbildungsverzeichnis

1	Logo	12
2	Blockschaltbild	13
3	Sophia Hagen	15
4	Julius Scherrer	15
5	Constantin Zumtobel	16
6	Sebastian Bellai	16
7	Diplo. Ing. Gerold Bischof	17
8	Profil und Verbinder	20
9	Skizze Gestell	21
10	Aussparung	23
11	Nut Vorderseite	23
12	Skizze Dichtungsrahmen in mm	27
13	Bild Rahmen	27
14	Schließer	28
15	Winkel	28
16	Zustand Schließer	29
17	Halterung Gyroskop	31
18	Zusammenschaltung Gyroskop	31
19	Schrittmotor	32
20	Motor am Gyroskop	32
21	Motor-Treiber	34
22	Netzteil	34
23	Relais ⁷⁶	35
24	Schaltplansymbol PRT-14017	36
25	Layout Top	39
26	Layout Bottom	39
27	PCB mit Polygon	40
28	PCB ohne Polygon	40
29	Unbestückte Platine	41
30	Bestückte Platine	41

⁷⁶ Team (webmaster@reichelt.de), *GRV RELAY SPDT30 - Arduino - Relais SPDT, 30 A, SLA-05VDC-SL-C.*

31	Blockschaltbild Sensoren	42
32	Gehäuse Temperatursensor	43
33	Temperatursensorausgabe	44
34	UV-Sensor	45
35	Messungen mit UV-Sensor	45
36	KY-024 Magnetsensor	46
37	Zusammenschaltung Magnetsensor ⁷⁷	47
38	Druckmessungen	48
39	Blogschaltbild des EDUs	49
40	Korrelation Dosimeter	52
41	UV-Lampe ⁷⁸	53
42	Dauermagnet ⁷⁹	55
43	Zusammenschaltung Pneumatik	57
44	Ventil	58
45	Zusammenschaltung Ventil	58
46	Skizze Vakuum-Ejektoren ⁸⁰	59
47	Ejektor Zusammenschaltung	60
48	Druckluft Kugel Vibrator ⁸¹	61
49	Rüttelplatte	61
50	Schaltplan LED	63
51	LED Verbindung	64
52	3D-Modell des CubeSats	65
53	Druckvorbereitung von zwei Teilen des CubeSats	65
54	Ausgedrucktes Modell des CubeSats	66
55	Raspberry Pi an der Metallplatte befestigt	66
56	Fertiges CubeSat-Modell	67
57	Boden	68
59	fertiger Boden	69
58	Zusätzliche Halterung	69
60	Deckel	70
61	Druckvorbereitung für den Boden	71
62	CubeSat in der Halterung	72

⁷⁷ KY-024 Linearer, magnetischer Hall-Sensor - SensorKit.

⁷⁸ <https://m.media-amazon.com/images/I/617AufdHaoL.AC SL1500.jpg>.

⁷⁹ 61QHrBwsDuL.SL1500.jpg (JPEG-Grafik, 1500 × 1500 Pixel) - Skaliert (37%).

⁸⁰ Datenblatt Vakuum Ejektor.

⁸¹ Pneumatischer Vibrator 245-4000 N 7500-34100 Mal/min Balltyp Filtersieb.

63	Arduino Lüftungssteuerung (Verkabelung) ⁸²	73
64	Pinbelegung des BC547-Transistor ⁸³	74
65	Aufbau Streifenrasterplatine (Lüftungssteuerung)	74
66	Lüfter am Plexiglas	75
67	Kühlgerät mit analogen Knöpfen ⁸⁴	76
68	Auseinandergebautes Kühlgerät	77
69	Verkabeltes Kühlgerät für Lüftungsstufen	78
70	Aufbau Streifenrasterplatine (Kühlgerät)	78
71	Halterung der Streifenrasterplatine	79
72	Anbringen der Streifenrasterplatine am Kühlgerät	79
73	Kühlung in der Teststation	80
74	Schlüsselschalter	82
75	Zusenschaltung Endschalter	83
76	Beispielcode Tasterabfrage	84
77	GUI-Bibliothek einbinden	86
78	Beispielcode Fenster erstellen	86
79	Beispielcode GUI-Fenster erstellen	86
80	Beispielcode Button	87
81	Beispielcode Temperatursensor	87
82	Beispielcode UV-Sensor	88
83	Beispielcode Drucksensor	89
84	Beispielcode UV-Lampe	90
85	Beispielcode UV-Lampe	91
86	Beispielcode LED-Streifen	92
87	Beispielcode Schlüsselschalter	92
88	Beispielcode Endschalter	93
89	Beispielcode Kühlung	94
90	Beispielcode Lüfter	95
91	Bild des CM3+ von Raspberry PI	96
92	Sendeverfahren des TCP-Protokolls ⁸⁵	97
93	Tabellenform	104
94	Darstellungsweise der CSV-Datei	105

⁸² SchaltungLüfter.

⁸³ BC547Bild.

⁸⁴ Emerio Verdunstungskühler, Luftkühler, für eine angenehme kühle Brise, Ventilator Funktion, 2 Geschwindigkeitseinstellungen, AC-116964, weiß/blau : Amazon.de: Küche, Haushalt & Wohnen.

⁸⁵ 1*I_E_SkdCYUrY3HyQ5U59WZg.webp (WEBP-Grafik, 551 × 351 Pixel).

95	Aufbau des I2C-Systems auf dem Raspberry Pi	110
96	Raspberry Pi	113
97	Befehlszeile ⁸⁶ und grafische Oberfläche ⁸⁷	114
98	Erweiterungen in VS-Code	115
99	API-Dekoratoren	118
100	Beispiel Swagger UI	119
101	API-Programm Temperatursensor	120
102	API-Programm UV-Sensor	120
103	API-Programm Drucksensor	121
104	API-Programm UV-Lampe	122
105	API-Ausgabe UV-Lampe	122
106	Beispiel eines Callbacks	124
107	Beispiel HTML-Struktur	124
108	Beispiel HTML-Layout	126
109	Beispiel DatePickerRange	126
110	Korrelation ⁸⁸	132
111	Messung Temperatur	140
112	Messung Temperatur	140
113	Messung Feuchtigkeit	141
114	Messung Druck	142
115	Formel Druck	142
116	Druckberechnung	143
117	Messung UV	144
118	Messung Gyroskop	145
119	QR-Code GitHub Repository	150
120	Aufteilung Arbeitsstunden	152
121	Logo SolidWorks ⁸⁹	165
122	Logo Eagle	165
123	Logo Thonny ⁹⁰	166
124	Logo VS-Code ⁹¹	166

⁸⁶ *lcd-sudo-su.png* (PNG-Grafik, 672 × 423 Pixel).

⁸⁷ *pixel.jpg* (JPEG-Grafik, 750 × 422 Pixel).

⁸⁸ *MQTT-architecture.jpg.webp* (WEBP-Grafik, 800 × 475 Pixel).

⁸⁹ *solidworks-connect_2.jpg* (JPEG-Grafik, 1518 × 644 Pixel) - Skaliert (84%).

⁹⁰ *thonny-Thonny-icon.png* (WEBP-Grafik, 224 × 224 Pixel).

⁹¹ *Visual_Studio_Code_1.35_icon.svg.png* (PNG-Grafik, 768 × 768 Pixel) - Skaliert (73%).

Tabellenverzeichnis

1	Meilensteine	14
2	Stückliste Gestell	22
3	Stückliste PVC-Platte	24
4	Stückliste PVC-Platte	24
5	Stückliste Dichtungsrahmen	26
6	Stückliste Halterung Gyroskop	30
7	Pinbelegung Raspberry Pi Shield	37
8	Pinbelegung Temperatursensor	43
9	Pinbelegung UV-Sensor	44
10	UV-Index	45
11	Pinbelegung Drucksensor	48
12	Pinbelegung UV-Lampe	54
13	Stückliste Pneumatik	56
14	Stückliste CubeSat	67
15	Stückliste Halterung	72
16	Stückliste Pneumatik	73
17	Stunden pro Monat	151
18	Stunden Zuhause und in der Schule	152
19	Gesamte Stunden	152

Literatur

- [1] *1*IE_SkdCYUrY3HyQ5U59WZg.webp* (WEBP-Grafik, 551 × 351 Pixel). URL: https://miro.medium.com/v2/resize:fit:828/format:webp/1*IE_SkdCYUrY3HyQ5U59WZg.png (besucht am 20.03.2024).
- [2] 237-103. de-at. URL: <https://www.digikey.at/de/products/detail/wago-corporation/237-103/15539605> (besucht am 15.02.2024).
- [3] *61QHrBwsDuL._SL1500_.jpg* (JPEG-Grafik, 1500 × 1500 Pixel) - Skaliert (37%). URL: https://m.media-amazon.com/images/I/61QHrBwsDuL._SL1500_.jpg (besucht am 22.02.2024).
- [4] Adafruit LTR390 UV Light Sensor 300 to 350nm STEMMA QT Qwiic 4831 - E, 6,95 €. de. URL: <https://eckstein-shop.de/Adafruit-LTR390-UV-Light-Sensor-300-to-350nm-STEMMA-QT-Qwiic> (besucht am 24.02.2024).

- [5] *adafruit/Adafruit_CircuitPython_DHT*. original-date: 2017-09-18T19:18:29Z. Jan. 2024. URL: https://github.com/adafruit/Adafruit_CircuitPython_DHT (besucht am 15.02.2024).
- [6] *ADXL345 - Acceleration Sensor*. de. URL: <https://electronics.semaf.at/ADXL345-Acceleration-Sensor> (besucht am 20.03.2024).
- [7] *Alu Profile 40 x 40 bis 80 x 80 | Robotunits*. URL: https://robotunits.com/produkt-kategorie/profil-und-verbindungstechnik/profiltechnik-raster-40/?_gl=1*fndd4f*_up*MQ..&gclid=CjwKCAiAqNSsBhAvEiwAn_tmxSP6zdLT_F1BTP2o31HqzjZf70Rmw6HcQwWkvvrNG_26JvM0b1CGJBoCeVMQAvD_BwE (besucht am 04.03.2024).
- [8] *ARD SEN HALL3: Arduino - Hall-Magnet-Sensor, linear bei reichelt kaufen*. URL: https://www.reichelt.at/at/de/shopat/produkt/arduino_-hall-magnet-sensor_linear-282517?PROVID=2807&gad_source=1&gclid=CjwKCAjw7-SvBhB6EiwAwYdCabZ701-FbjLjSRpjVmrhAa5qN7-wLTtEN4oU8bHXUSue7ac6Dx15tBoCSloQAvD_BwE&q=%2Fat%2Fde%2Fshopat%2Fat%2Fde%2Farduino-hall-magnet-sensor-linear-ard-sen-hall3-p282517.html (besucht am 19.03.2024).
- [9] *Arduino - Home*. URL: <https://www.arduino.cc/> (besucht am 20.03.2024).
- [10] *Autodesk Eagle*. URL: <http://eagle.autodesk.com/eagle/software-versions/1> (besucht am 28.02.2024).
- [11] *AZDelivery GY-68 BMP180 Barometrischer Luftdruck und Temperatur Sensor kompatibel mit Arduino und Raspberry Pi inklusive E-Book!* : Amazon.de: Gewerbe, Industrie & Wissenschaft. URL: https://www.amazon.de/dp/B07D8S617X?ref=ppx_yo2ov_dt_b_product_details&th=1 (besucht am 22.02.2024).
- [12] *Bachmann electronic GmbH - Bachmann electronic GmbH*. URL: <https://www.bachmann.info/de> (besucht am 22.02.2024).
- [13] *BME 688: KI-Kombo-Sensor, Luftdruck/Luftfeuchtigkeit/Temp./Gas bei reichelt kaufen*. URL: https://www.reichelt.at/at/de/shopat/produkt/ki-kombo-sensor_luftdruck_luftfeuchtigkeit_temp_gas-306044?q=%2Fat%2Fde%2Fshopat%2Fki-kombo-sensor-luftdruck-luftfeuchtigkeit-temp-gas-bme-688-p306044.html (besucht am 20.03.2024).
- [14] *BMM150 | Bosch Sensortec Bewegungssensormodul 3-Achsen SMD I2C / SPI Digital | RS*. URL: https://at.rs-online.com/web/p/bewegungssensor-ics/2457371?cm_mmc=AT-PPC-DS3A--google--DSA_AT_DE_Halbleiter_Index--Bewegungssensor+ICs%7C+Products--DYNAMIC+SEARCH+ADS&matchtype=&dsat1654329450134&gad_source=1&gclid=CjwKCAjwkuqvBhAQEiwA65XxQP9VN1biIntS7zBuzc5rY

NugztQt1XhkPJmtFZY61_vwQ6QHg3xoCVtYQAvD_BwE&gclsrc=aw.ds (besucht am 20.03.2024).

- [15] *circuitpython-bmp180: CircuitPython driver for the bmp180 sensor.* URL: https://github.com/jposada2020/CircuitPython_bmp180 (besucht am 24.02.2024).
- [16] *Cosmic Rays Exposure.* EN. URL: https://laradioactivite.com/in_daily_life/dose_cosmic (besucht am 20.03.2024).
- [17] *Dash Documentation & User Guide | Plotly.* URL: <https://dash.plotly.com/> (besucht am 19.03.2024).
- [18] *Datenblatt Vakuum Ejektor.* URL: https://www.parker.com/literature/Pneumatics%20Division%20Europe/PDE-Documents/Vacuum%20Generators_Technical%20Catalogue-DE.pdf.
- [19] *Datenblatt VY-83ELB00-T.* URL: https://www.airwork-pneumatic.com/wp-content/uploads/2016/11/VY_solenoid_valves_2021.pdf.
- [20] *Emerio Verdunstungskühler, Luftkühler, für eine angenehme kühle Brise, Ventilator Funktion, 2 Geschwindigkeitseinstellungen, AC-116964, weiß/blau : Amazon.de: Küche, Haushalt & Wohnen.* URL: <https://www.amazon.de/Emerio-Verdunstungsk%C3%BChler-Luftk%C3%BChler-Ventilator-Geschwindigkeitseinstellungen/dp/B0893QFYNP> (besucht am 20.03.2024).
- [21] *Erdmagnetfeld – Wikipedia.* URL: <https://de.wikipedia.org/wiki/Erdmagnetfeld#:~:text=Am> (besucht am 20.03.2024).
- [22] *Features - FastAPI.* en. URL: <https://fastapi.tiangolo.com/features/> (besucht am 24.02.2024).
- [23] *GanttProject: Free Project Management App.* URL: <https://www.ganttpoint.biz/> (besucht am 04.03.2024).
- [24] *GET vs. POST – die beiden wichtigsten HTTP-Requests im Vergleich.* de-AT. Apr. 2020. URL: <https://www.ionos.at/digitalguide/websites/web-entwicklung/get-vs-post/> (besucht am 24.02.2024).
- [25] *Hefel Technik auf Facebook.* de-AT. URL: <https://www.hefel-technik.com/> (besucht am 22.02.2024).
- [26] *Homepage.* de-AT. URL: <https://spaceteam.at/> (besucht am 22.02.2024).
- [27] *Adafruit Industries. AM2302 (wired DHT22) temperature-humidity sensor.* en-US. URL: <https://www.adafruit.com/product/393> (besucht am 14.02.2024).

- [28] *I²C Slave Mode*. URL: <https://onlinedocs.microchip.com/pr/GUID-CA6B1F8F-E09D-4780-A52A-CBBF61902464-en-US-2/index.html?GUID-5CCAB0DB-28BD-4095-B2E2-2F3CF0FC6966> (besucht am 05.04.2024).
- [29] *KY-024 Linearer, magnetischer Hall-Sensor - SensorKit*. URL: <https://sensorkit. joy-it.net/de/sensors/ky-024> (besucht am 19.03.2024).
- [30] *lcd-sudo-su.png (PNG-Grafik, 672 × 423 Pixel)*. URL: <https://www.circuitbasics.com/wp-content/uploads/2015/02/lcd-sudo-su.png> (besucht am 15.02.2024).
- [31] Marc Mai. *MQTT für Dummies*. de-DE. März 2016. URL: <https://blog.doubleslash.de/mqtt-fuer-dummies> (besucht am 19.03.2024).
- [32] *MeanWell LRS-100-24 108W 24V 4,5A Industrielles Netzteil : Amazon.de: Computer & Zubehör*. URL: https://www.amazon.de/dp/B06XWR8RGJ?ref=ppx_yo2ov_dt_b_product_details&th=1 (besucht am 04.03.2024).
- [33] *MEEKBOS UV Schwarzlicht Strahler,60W LED Schwarzlichtlampe,385-400NM IP66 Wasserdicht UV Flutlicht mit Stecker,UV Fluter für Black light Party,GLOW,Fluoreszierende Malerei,Körperbemalung,Halloween : Amazon.de: Beleuchtung*. URL: https://www.amazon.de/dp/B09Y34KSR5?psc=1&ref=ppx_yo2ov_dt_b_product_details (besucht am 18.02.2024).
- [34] *MQTT-architecture.jpg.webp (WEBP-Grafik, 800 × 475 Pixel)*. URL: <https://www.verytechnology.com/wp-content/uploads/2023/06/MQTT-architecture.jpg.webp> (besucht am 19.03.2024).
- [35] *OCEUMAOA Neodym Magnete mit Loch 30kg Magnete Stark,Magnet zum Anschrauben mit Schrauben für Werkstatteinrichtung Zubehör 25mm 4 Stück : Amazon.de: Küche, Haushalt & Wohnen*. URL: https://www.amazon.de/dp/B0C61QL9LD/ref=sspa_dk_detail_2?pd_rd_i=B0C61QL9LD&pd_rd_w=JicHo&content_id=amzn1.sym.ae2317a0-2175-4285-af64-66539858231f&pf_rd_p=ae2317a0-2175-4285-af64-66539858231f&pf_rd_r=G92WGQ0R5PSNDJP713&pd_rd_wg=9E3cK&pd_rd_r=3d5d3a39-475d-4c21-85cd-c5ad2e74ff9f&s=industrial&sp_csd=d21kZ2V0TmFtZT1zcF9kZXRhawWw&th=1 (besucht am 22.02.2024).
- [36] *OMICRON*.
- [37] *pixel.jpg (JPEG-Grafik, 750 × 422 Pixel)*. URL: <https://i.ds.at/GMLBkw/rs:fill:750:0/plain/2016/09/29/pixel.jpg> (besucht am 15.02.2024).
- [38] *Pneumatischer Vibrator 245-4000 N 7500-34100 Mal/min Balltyp Filtersieb*. de-de. URL: <https://www.ebay.de/itm/176052899010> (besucht am 10.03.2024).

- [39] *PRT-14017 footprint & symbol by SparkFun Electronics | SnapMagic Search.* URL: <https://www.snapeda.com/parts/PRT-14017/SparkFun%20Electronics/view-part/> (besucht am 15.02.2024).
- [40] Sebastián Ramírez. *FastAPI.* original-date: 2018-12-08T08:21:47Z. Feb. 2024. URL: <https://github.com/tiangolo/fastapi> (besucht am 18.02.2024).
- [41] *Raspberry Pi 4 Modell B 4 GB RAM, Cortex-A72 64 Bit Wi-Fi Bluetooth: Amazon.de: Computer & Zubehör.* URL: https://www.amazon.de/gp/product/B0C7KXMP7W/ref=ppx_yo_dt_b_asin_title_o07_s00?ie=UTF8&psc=1 (besucht am 04.03.2024).
- [42] *Raspberry Pi: Erste Schritte bei der Konfiguration (Grundkonfiguration).* URL: <https://www.elektronik-kompendium.de/sites/raspberry-pi/1906291.htm> (besucht am 15.02.2024).
- [43] *Robotunits – Der Automatisierungsbaukasten. de_DE.* URL: <https://robotunits.com/> (besucht am 22.02.2024).
- [44] *RPI ADC: Raspberry Pi - Analog/Digital Konverter bei reichelt kaufen.* URL: https://www.reichelt.at/at/de/shopat/produkt/raspberry_pi_-_analog_digital_konverter-226847?PROVID=2807&gad_source=1&gclid=CjwKCAjw7-SvBhB6EiwAwYdCAAdMjdMf7eZKzJNCL_2VYPVofDJnrXHpxryPIag0MNFIP9oVM3vPepRoCYOEQAvD-BwE&q=%2Fat%2Fde%2Fshopat%2Fat%2Fde%2Fraspberry-pi-analog-digital-konverter-rpi-adc-p226847.html (besucht am 19.03.2024).
- [45] *Smart Home Weather Station. en-EU.* URL: <https://www.netatmo.com/en-eu/smart-weather-station> (besucht am 24.02.2024).
- [46] *SOLIDWORKS.* URL: <https://www.solidworks.com/de> (besucht am 15.02.2024).
- [47] *solidworks-connect_2.jpg (JPEG-Grafik, 1518 × 644 Pixel) - Skaliert (84%).* URL: https://www.solidworks.com/sites/default/files/inline-images/solidworks-connect_2.jpg (besucht am 15.02.2024).
- [48] *STEPPERONLINE Schrittmotor Nema 17 Stepper Motor 59Ncm 2A 1.8deg 2 Phase 4-Draht Stepping Motor w/1m Kabel & Verbinder für 3D Drucker/CNC Reprap : Amazon.de: Gewerbe, Industrie & Wissenschaft.* URL: https://www.amazon.de/dp/B00PNEQKC0?psc=1&ref=ppx_yo2ov_dt_b_product_details (besucht am 04.03.2024).
- [49] reichelt elektronik GmbH Internet Team (webmaster@reichelt.de). *CTB0502-4 - Lötbare Schraubklemme - 4-pol, RM 5 mm, 90°.* de. URL: <https://www.reichelt.at/at/de/loetbare-schraubklemme-4-pol-rm-5-mm-90--ctb0502-4-p292693.html> (besucht am 15.02.2024).

- [50] reichelt elektronik GmbH Internet Team (webmaster@reichelt.de). *GRV RELAY SPDT30 - Arduino - Relais SPDT, 30 A, SLA-05VDC-SL-C.* de. URL: <https://www.reichelt.at/at/de/arduino-relais-spdt-30-a-sla-05vdc-sl-c-grv-relay-spdt30-p243033.html> (besucht am 19.02.2024).
- [51] *Tera Term.* en. Nov. 2023. URL: <https://sourceforge.net/projects/teraterm/> (besucht am 19.03.2024).
- [52] *Thonny, Python IDE for beginners.* URL: <https://thonny.org/> (besucht am 19.02.2024).
- [53] *thonny-Thonny-icon.png (WEBP-Grafik, 224 × 224 Pixel).* URL: https://images.sftcdn.net/images/t_app-icon-m/p/b7c6837f-d3d8-48a2-985a-aae1988e1ac3/524712048/thonny-Thonny-icon.png (besucht am 19.02.2024).
- [54] *TMP112-Q1 Datenblatt, Produktinformationen und Support | TI.com.* URL: <https://www.ti.com/product/de-de/TMP112-Q1> (besucht am 20.03.2024).
- [55] *Transmission Control Protocol.* de. Page Version ID: 243014776. März 2024. URL: https://de.wikipedia.org/w/index.php?title=Transmission_Control_Protocol&oldid=243014776#Daten%C3%BCbertragung (besucht am 20.03.2024).
- [56] *Tschabrun Holz- und Baustoffe in Rankweil, Bludenz-Bürs und Innsbruck | Tschabrun Holz & Baustoffe - Starke Produkte. Starke Beratung.* de. URL: <https://www.tschabrun.at/> (besucht am 22.02.2024).
- [57] *Twotrees DM542 5.6A Schrittmotortreiber Stepper Treiber, Dm542 2 Phasen Schrittmotortreiber Dc 18-48v 57/86 Stepper Motor Driver Peak 4.2a : Amazon.de: Gewerbe, Industrie & Wissenschaft.* URL: https://www.amazon.de/dp/B093LFXT5S?ref=ppx_yo2ov_dt_b_product_details&th=1 (besucht am 04.03.2024).
- [58] *User Datagram Protocol.* de. Page Version ID: 242735887. März 2024. URL: https://de.wikipedia.org/w/index.php?title=User_Datagram_Protocol&oldid=242735887 (besucht am 20.03.2024).
- [59] *Uvicorn.* URL: <https://www.uvicorn.org/> (besucht am 18.02.2024).
- [60] *Verbinder 40x40 | Robotunits Verbindungstechnik Raster 40.* URL: https://robotunits.com/shop/profil-und-verbindungstechnik/verbindungstechnik-raster-40/verbinder-40x40/?_gl=1*1aqg5as*_up*MQ..&gclid=CjwKCAiAqNSsBhAvEiwAn_tmxSP6zdLT_F1BTP2o31HqzjZf70Rmw6HcQwWkvvrNG_26JvM0b1CGJBoCeVMQAvD_BwE (besucht am 04.03.2024).
- [61] *Visual Studio Code - Code Editing. Redefined.* en. URL: <https://code.visualstudio.com/> (besucht am 23.02.2024).

- [62] *Visual_Studio_Code_1.35_icon.svg.png* (PNG-Grafik, 768 × 768 Pixel) - Skaliert (73%).
URL: https://upload.wikimedia.org/wikipedia/commons/thumb/9/9a/Visual_Studio_Code_1.35_icon.svg/768px-Visual_Studio_Code_1.35_icon.svg.png (besucht am 23.02.2024).

17 Anhang

17.1 Entwicklungsumgebung für Hardware

17.1.1 Solidworks⁹²

SOLIDWORKS ist eine professionelle 3D-CAD-Software (Computer-Aided Design), die es Ingenieuren und Designern ermöglicht, hoch-präzise Modelle und Zeichnungen von Produkten zu erstellen. Die Software bietet eine breite Palette von Funktionen, darunter Modellierung, Baugruppenkonstruktion, Zeichnungserstellung, Simulation, Datenmanagement und vieles mehr. Mit SOLIDWORKS können Anwender komplizierte Bauteile und Baugruppen modellieren, um realistische virtuelle Darstellungen von Produkten zu erstellen. Die Software bietet leistungsstarke Werkzeuge für die Erstellung von Skizzen, die Extrusion von Formen, das Hinzufügen von Details wie Bohrungen und Gewinden sowie die Erstellung komplexer Oberflächen. Darüber hinaus ermöglicht SOLIDWORKS die Erstellung von Baugruppen, in denen mehrere Teile miteinander verbunden sind. Benutzer können Teile zu Baugruppen zusammenfügen, Bewegungen simulieren und Kollisionen erkennen, um sicherzustellen, dass das Endprodukt korrekt funktioniert.



Abbildung 121: Logo SolidWorks⁹³

17.1.2 Eagle 7.7.0

EAGLE⁹⁴ ist die Abkürzung für Einfach Anzuwendender Grafischer Layout-Editor. Die Software besteht aus mehreren Komponenten, Layout-Editor, Schaltplan-Editor, Autorouter und Bauteil-Bibliothek. Mit diesen Komponenten können Schaltpläne gezeichnet werden und danach das Board erstellt werden. Eagle besitzt eine große Bibliothek an Bauteilen. Es gibt jedoch viele Bauteile, die vom Internet heruntergeladen werden können und zur Bibliothek hinzugefügt werden können. Bauteile können auch selbst gezeichnet werden.



Abbildung 122: Logo Eagle

⁹² SOLIDWORKS.

⁹⁴ Autodesk Eagle.

17.2 Entwicklungsumgebung für Software

17.2.1 Thonny

Thonny⁹⁶ ist eine benutzerfreundliche und einfache Python-Entwicklungsumgebung (IDE), ideal für unser Testprogramm. Es bietet einen einfachen Code-Editor mit Syntaxhervorhebung und Autovervollständigung, einen interaktiven Modus zum schrittweisen Ausführen von Code, Debugging-Funktionen, einfache Paketinstallation und Integration mit Mikrocontroller-Plattformen wie dem Raspberry Pi .



Abbildung 123: Logo
Thonny⁹⁵

17.2.2 Visual Studio Code⁹⁷

Visual Studio Code, kurz auch VS Code, ist ein Code-Editor. Dieser Editor ist plattformübergreifend für die Betriebssysteme Windows, macOS und Linux verfügbar. VS-Code wird hauptsächlich von einem Team in der Schweiz entwickelt und wird als offenes Projekt auf GitHub verfügbar. Jeden Monat erscheint eine neue Version und es ist das am stärksten unterstützte Projekt auf GitHub. VS-Code ist geeignet für die Programmierung in vielen verschiedenen Programmiersprachen. Beispiele für Programmiersprachen sind, Batch, C++, Python und viele mehr. Durch Plug-ins kann VS-Code erweitert werden.



Abbildung 124:
Logo VS-Code⁹⁸

⁹⁶ Thonny, Python IDE for beginners.

⁹⁷ Visual Studio Code - Code Editing. Redefined.

17.2.3 Plotly Dash

Plotly Dash⁹⁹ ist ein Open-Source-Framework für interaktive Webanwendungen in Python. Es erleichtert Entwicklern das Erstellen von Webanwendungen, ohne dass Kenntnisse in HTML oder CSS notwendig sind. Die Apps sind komplett in Python geschrieben. Plotly.js wird für Datenvisualisierung genutzt, was viele interaktive Diagramme ermöglicht. Eine Dash-App besteht aus zwei Teilen:

Das Layout bestimmt das Aussehen der App und ordnet die visuellen Elemente an. Es wird mit vordefinierten Dash-Komponenten erstellt, einschließlich HTML-Elementen und Plotly-Diagrammen.

Callbacks machen die App interaktiv. Sie werden aktiviert, wenn Benutzer mit der App interagieren, z.B. einen Button klicken oder eine Auswahl aus einem Dropdown-Menü treffen. Diese Funktionen nehmen Benutzereingaben entgegen, führen Berechnungen oder Datenabfragen durch und aktualisieren die Anzeigeelemente mit den neuen Daten.

17.2.4 TeraTerm

TeraTerm¹⁰⁰ ist ein Open-Source-Emulationstool für Microsoft Windows. Es ist ein Terminal-Emulator, der die Kommunikation mit entfernten Systemen über verschiedene Protokolle ermöglicht - Telnet, SSH (Secure Shell) und serielle Verbindungen. TeraTerm unterstützt auch Makro-gesteuerte Logik, was es für automatisierte Aufgaben und Tests nützlich macht.

17.3 3D-Druck

Um die einzelnen Teile drucken zu können, müssen die von der Technischen Universität Wien zur Verfügung gestellten STL-Dateien in Dremel DigiLab 3D Slicer wie folgt verarbeitet werden:

Als erstes sollte 3D-Modell (üblicherweise im .stl-Format) importiert werden. Danach wird das Modell auf der virtuellen Druckplatte platziert und nach Bedarf angepasst. Anschließend wird das Material ausgewählt und die Druckqualität sowie die Füllungsdichte festgelegt.

Bei Bedarf können Stützstrukturen hinzugefügt werden und Raft und Brim für eine bessere Haftung auf der Druckplatte genutzt werden. Mit Klicken auf „Slice“ wird eine Gcode-Datei erzeugt, welche dann auf den 3D-Drucker übertragen und gedruckt werden kann.

⁹⁹ *Dash Documentation & User Guide | Plotly*.

¹⁰⁰ *Tera Term*.

17.4 Datenblätter

Datenblatt ADXL345:

<https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf>

Datenblatt BME688:

https://www.mouser.at/datasheet/2/783/bst_bme688_f1000-2307034.pdf

Datenblatt BMM150:

https://www.mouser.at/datasheet/2/783/BST_BMM150_DS001-1509615.pdf

Datenblatt GUVA-C32SM:

<https://www.digikey.at/en/htmldatasheets/production/3642461/0/0/1/guva-c32sm>

Datenblatt TMP112:

<https://www.ti.com/document-viewer/TMP112/datasheet>

Datenblatt Taster:

https://www.lcsc.com/datasheet/lcsc_datasheet_1912111437_SHOU-HAN-TS5215A-160gf-C412369.pdf

Datenblatt BMP180:

<https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf>

Datenblatt AM2302:

<https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf>

Datenblatt LTR390:

https://optoelectronics.liteon.com/upload/download/DS86-2015-0004/LTR-390UV_Final_%20DS_V1%201.pdf

Datenblatt KY-024:

https://cdn.shopify.com/s/files/1/1509/1638/files/Hall_Sensor_Modul_Datenblatt.pdf?11496986819545999115

Datenblatt 16-Bit ADC KY-053:

https://joy-it.net/files/files/Produkte/COM-KY053ADC/COM-KY053ADS_Datenblatt_2022-11-25.pdf