

Dokumentation – Masterprojekt 1

Entwurf eines vernetzten, teleoperierten 1:5 Fahrzeugs

Masterstudiengang Elektrotechnik mobiler Systeme



Projektbetreuer:

Prof. Dr. Christian Birkner

Tobias Kern

Projektbearbeiter:

Hirschmann, Tom

Barabadi, Arash

Alfarhan, Akram

Hrabos, Albin

Mohammadi, Awat

Rawat, Shubham

Joelle Christelle, Pouadeu

Yaghoubi, Shahrzad

Bearbeitungszeitraum: Wintersemester 2023/2024

Inhaltsverzeichnis

1	Einleitung.....	5
1.1	Motivation und Ziel der Arbeit	5
1.2	Verteilung der Gruppen.....	5
1.3	Aufteilung des praktischen Teils der Projektarbeit	6
1.3.1	Hardware-Team.....	6
1.3.2	Software-Team	7
2	Hardware-Dokumentation	8
2.1	Ermittlung der technischen Daten der Komponenten	8
2.1.1	Motor.....	8
2.1.2	Umrichter	8
2.1.3	Batterie	10
2.1.4	Jetson.....	10
2.1.5	Servo-Motor	12
2.1.6	Kamera.....	13
2.1.7	IMU	14
2.1.8	DC-/DC-Wandler.....	14
2.1.9	Lüfter	15
2.1.10	Positionsgeber	15
2.1.11	Antenne	16
2.1.12	Zusammenfassung der technischen Daten	17
2.3	Erarbeitung des Schaltplans	18
2.3.1	Versorgung der Komponenten	18
2.3.2	Bestimmung der zu verwendenden Schnittstellen	19
2.3.3	Erstellung des Schaltplans	20
2.4	Festlegung der vorläufigen Leitungsverläufe	21
2.4.1	Festlegung der Rahmenbedingungen	21
2.4.2	Bestimmung der Leitungsverläufe	22
2.5	Ermittlung der groben Leitungslängen.....	26
2.6	Berechnung und Auswahl der Kabelquerschnitte.....	27
2.7	Auswahl geeigneter Leitungen	30
2.7.1	Bestimmung des Leitermaterials.....	31
2.7.2	Bestimmung des Isolationsmaterials.....	31
2.7.3	Festlegung der Farbcodierung der Isolation	31

2.8	Auswahl der Steckverbindungen.....	32
2.9	Bestimmung der endgültigen Leiterverläufe und Steckerpositionen	33
2.9.1	Grundplatte	33
2.9.2	obere Platte (Unterseite).....	34
2.10	Durchführung der Verkabelung.....	35
2.11	Mögliche Verbesserungen an der Verkabelung	36
3	Software-Dokumentation (Albin Hrabos).....	37
3.1	Ubuntu 20.04 (Arash Barabadi).....	37
3.2	ROS2 (Arash Barabadi)	37
3.3	Fahrzeug- und Fahrerseite (Albin Hrabos)	37
3.4	ROS2-Begriffe (Arash Barabadi)	38
3.4.1	Workspace	38
3.4.2	Package (Paket)	38
3.4.3	Erstellung von ROS2-Workspace	38
3.4.4	Erstellung von ROS2-Paket(Package)	38
3.4.5	Node	40
3.4.7	Kommunikationsfunktionen	41
3.4.8	Launch-System	41
3.5	ROS2-Architektur (Arash Barabadi).....	42
3.6	Mesh Netzwerk Konfiguration (Albin Hrabos)	48
3.7	Treiber von W-Lan Modul (Arash Barabadi).....	50
3.8	Bildübertragung von Kamera (Albin Hrabos)	51
3.8.1	RealSense Viewer	51
3.8.2	Kamera Node.....	52
3.9	Kommunikation und Befehleingabe (Albin Hrabos).....	52
3.10	Start von Kamera Node auf dem Jetson (Arash Barabadi).....	53
3.11	Automatisierung von Befehlen (Albin Hrabos)	54
3.12	Verzögerung des Systems (Albin Hrabos).....	54
3.12.1	Video-Verzögerung im Mesh Netzwerk	54
3.12.2	Gesamtverzögerung	56
3.13	Bild Komprimierung (Albin Hrabos).....	56
3.14	Start der Teleoperation (Albin Hrabos)	57
3.15	Rückkehren des Lenkrades zur Mittelposition (Albin Hrabos)	58
3.16	Andere Befehle (Albin Hrabos).....	59
3.17	VESC –Tool (Shubham Rawat)	59

3.18	IMU-Sensor (Arash Barabadi)	61
3.18.1	Einführung des IMU-Sensors in ROS2 (Arash Barabadi).....	62
3.18.2	Teil 7 von ROS2-Architektur (Arash Barabadi)	63
3.18.4	Erklärung des Python-Nodes (Arash Barabadi)	64
3.19	Launch-Datei (Arash Barabadi).....	66
4	Abbildungsverzeichnis.....	70
5	Tabellenverzeichnis	71
6	Literaturverzeichnis.....	72

1 Einleitung

1.1 Motivation und Ziel der Arbeit

Verfasser des Abschnitttextes: Joelle (80%), Tom (10%), Akram (10%)

Bearbeiter des Abschnitts: Joelle (40%), Tom (30%), Akram (30%)

Die Automobilindustrie befindet sich in einer großen Transformation, bedingt durch steigende CO₂-Emissionen, Erderwärmung, begrenzte fossile Brennstoffe und strengere politische Regulierungen. Die Elektromobilität bietet Lösungen für diese Herausforderungen, mit Schlagworten wie Vernetzung, Funktionalität und autonomes Fahren. Deshalb soll im Rahmen dieses Projekts ein vernetztes und teleoperiertes Modellfahrzeug entworfen werden, welches funktional und effizient ist. Diese Maßnahmen sollen Funktionalität, Zuverlässigkeit und Produktqualität des Fahrzeugs verbessern. Als erste werden wir ein detailliertes Konzept für ein vernetztes, teleoperiertes Fahrzeug im Maßstab 1:5 entwickeln, wobei unser Hauptaugenmerk auf die Integration elektronischer Komponenten gelegt wird. Die Verkabelung wird präzise und sicher gestaltet, unter Einhaltung aller relevanten elektrischen Sicherheitsstandards. Das Robot Operating System 2 (ROS2) wird als Framework eingesetzt, um die Schnittstelle zu implementieren, die für die Teleoperation mit Bildübertragung verantwortlich ist. Als Fahrplan für die Arbeit wurden folgende Meilensteine festgelegt: In der ersten Phase werden die Aktoren und Sensoren gemäß den technischen Spezifikationen und Anforderungen des Projekts installiert. Diese Sensoren und Aktoren werden dann in Betrieb genommen. Eine erfolgreiche Kommunikation mit dem Steuerungssystem wird als Akzeptanzkriterium für die Fortsetzung festgelegt. Im Laufe der Entwicklung werden teamübergreifende Tests von Hardware und Software durchgeführt, um Fehler schnell zu identifizieren und zu beheben.

1.2 Verteilung der Gruppen

Verfasser des Abschnitttextes: Joelle (50%), Tom (20%), Akram (30%)

Bearbeiter des Abschnitts: Joelle (33%), Tom (33%), Akram (33%)

Die Gruppenverteilung dieses Projekts teilt sich wie bereits genannt in das Hardware- und Software-Team auf.

Das **Hardware-Team** ist verantwortlich für die Entwicklung und Umsetzung der Hardwareverkabelung, Integration von Sensoren, Aktoren und Steuerungssystemen sowie gründliche Tests zur Sicherstellung reibungsloser Kommunikation für das Projekt und besteht aus den Mitgliedern:

- Tom Hirschmann
- Akram Alfarhan
- Awat Mohammadi
- Shahrzad Yaghoubi,
- Joelle Christelle Pouadeu

Das **Software-Team** ist verantwortlich für die Implementierung der Software in ein Gesamtpaket, auf dem die Teleoperation fungiert. Es besteht aus den Mitgliedern:

- Albin Hrabos
- Arash Barabadi
- Shubham Rawat

Abbildung 1 zeigt die genaue Gruppenverteilung als Diagramm.

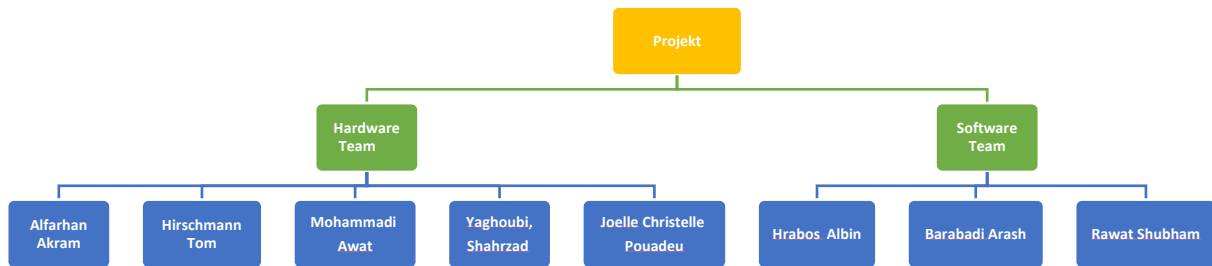


Abbildung 1: Gruppenverteilung des Projekts

Abbildung 2 ist die Grundlage für den Schaltplan, anhand welchem sichtbar wird, welche Komponenten miteinander kommunizieren. Generell gibt es zwei Seiten, die miteinander kommunizieren, die Fahrer- und Fahrzeugseite. Die rechte Seite bezieht sich auf das Auto selbst und die darauf installierte Software. Die Kommunikation zwischen der linken und der rechten Seite erfolgt über Mesh-Netzwerk.

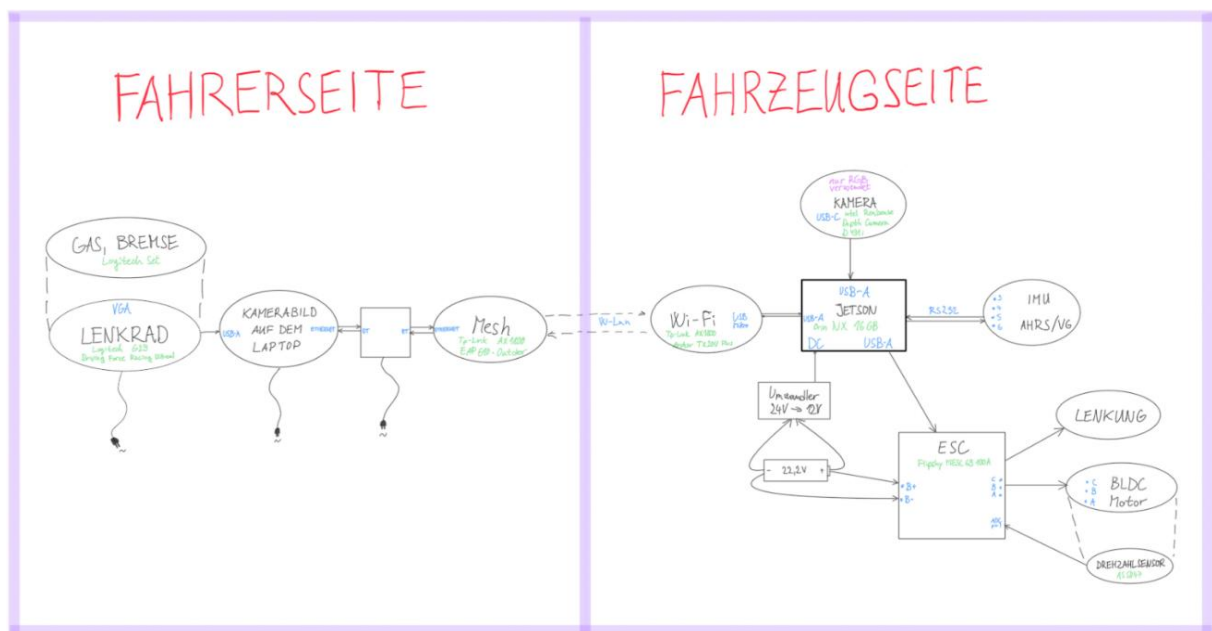


Abbildung 2: Strukturbild des Fahrzeugs
Autor der Abbildung: Albin Hrabos

1.3 Aufteilung des praktischen Teils der Projektarbeit

Die Aufteilung der praktischen Tätigkeit im Labor sieht wie folgt aus:

1.3.1 Hardware-Team

- | | |
|-----------------------------|-------|
| • Akram Alfharan | 29,0% |
| • Tom Hirschmann | 34,0% |
| • Awat Mohammadi | 14,0% |
| • Shahrzad Yaghoubi | 8,5% |
| • Joelle Christelle Pouadeu | 14,5% |

1.3.2 Software-Team

- Albin Hrabos 33%
- Arash Barabadi 33%
- Shubham Rawat 33%

2 Hardware-Dokumentation

2.1 Ermittlung der technischen Daten der Komponenten

2.1.1 Motor

Verfasser des Abschnitttextes: Shahrzad (50%), Akram (30%), Tom (20%)

Bearbeiter des Abschnitts: Tom (55%), Akram (20%), Shahrzad (25%)

Der "TT5692 Top Tuning Brushless Motor 1090 kV" ist ein leistungsstarker Motor, der u.a. in Modellautos verwendet wird. Er ist speziell für den Einsatz in ferngesteuerten Fahrzeugen entwickelt worden. Eine wichtige Tatsache ist, dass Brushless Motoren im Vergleich zu herkömmlichen, bürstenbehafteten Motoren eine höhere Leistung und Effizienz bieten. Sie haben auch eine längere Lebensdauer, da keine Bürsten verschleifen. [1]

Der hier verwendete Antrieb weist folgende Spezifikationen auf:

- Max. Strom: 195 A
- Max. Spannung: 27 V
- Max. Leistung: 5.200 W
- Anschlusskabel: 8AWG
- 4-poliger Rotor mit hochwertigen Magneten
- Stator aus superdünnem Siliziumstahl
- CNC-Aluminiumgehäuse mit Kühlrippen
- Hochtemperaturfeste Kupferwicklung
- Drei Phasen (U, V, W)



Abbildung 3: Motor TT5692 Top Tuning Brushless Motor
Quelle: [1]

2.1.2 Umrichter

Verfasser des Abschnitttextes: Joelle (75%), Akram (25%)

Bearbeiter des Abschnitts: Joelle (20%), Akram (60%), Tom (20%)

Ein Umrichter beschreibt eine elektrische Schaltung, die die Eigenschaft hat, eine zugeführte Wechselstrom-, Drehstrom- oder Wechselspannung in eine andere Wechselstrom-, Drehstrom- bzw. Wechselspannung umzuwandeln. Im Falle einer Spannung wird die neue Spannung sich von der ersten Spannung über ihre Höhe, ihre Frequenz oder ihre Phasenzahl unterscheiden. Bei Wechselstrom oder Drehstrom verursacht der Umrichter eine Änderung der Frequenz und/oder der Amplitude. Ein

Umrichter besteht normalerweise aus zwei Komponenten: einem Gleichrichter und einem Wechselrichter. Ein Umrichter ist auch als Frequenzumrichter oder AC/AC-Konverter bekannt.

Der FlipSky FSESC 6.9 setzt einen neuen Standard für die Leistung von Wechselrichtern im Bereich der Elektrofahrzeuge. Mit seinen fortschrittlichen Funktionen und seiner Zuverlässigkeit ist er die perfekte Wahl für unser Projekt.

Die Firmware ist stets auf dem neuesten Stand, um optimale Leistung sicherzustellen. Die Strombelastbarkeit ist bemerkenswert, mit einem Dauer Strom von 100A und einem Peak Strom von 200A.

In Bezug auf die Spannung unterstützt der Umrichter einen Betriebsspannungsbereich von 14V-60V (Zellen: 4-13S), wobei er sicher für den Einsatz von 4S bis 12S ist, wobei Spannungsspitzen 60V nicht überschreiten dürfen.

Das integrierte BEC (Battery Eliminator Circuit) liefert eine zuverlässige 5V-Ausgangsspannung bei 1A für die Stromversorgung elektronischer Komponenten.

Mit einer maximalen ERPM von 150.000 ermöglicht der FSESC 6.9 eine präzise Steuerung und Anpassung. Die verschiedenen Schnittstellenanschlüsse, darunter USB, CAN und UART, bieten vielfältige Steuerungsmöglichkeiten.

Unterstützt werden verschiedene Sensoren wie ABI, HALL, AS5047 und AS5048A. Eingangssignale können über PPM, ADC, NRF, UART, SPI, IIC erfolgen.

Verschiedene Betriebsmodi, darunter DC, BLDC und FOC (sinusförmig), machen den Umrichter äußerst vielseitig. Die Regenerationsfähigkeit ermöglicht die Rückgewinnung von Energie während des Bremsvorgangs.

Der FSESC 6.9 ist programmierbar und kann individuell konfiguriert werden. Motor- und Leitungsdetails umfassen eine 10AWG-Motorleitung und ein 8AWG-Stromkabel.

Die Abmessungen betragen 75,4 mm x 63,7 mm x 31,1 mm, einschließlich Kühlkörper. Insgesamt präsentiert sich der FlipSky FSESC 6.9 als eine leistungsstarke und flexible Lösung für eine Vielzahl von elektrischen Antriebsanwendungen. [2] [3]



Abbildung 4: Umrichter FlipSky FSESC 6.9
Quelle: [2]

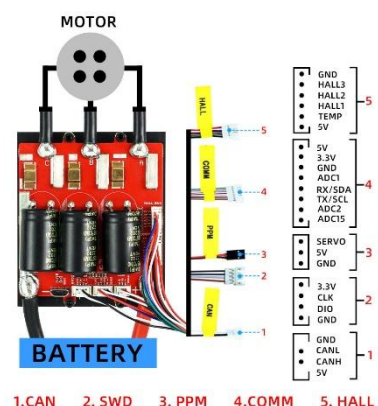


Abbildung 5: Pinverteilung des Umrichters
Quelle: [3]

2.1.3 Batterie

Verfasser des Abschnitttextes: Joelle (100%),
Bearbeiter des Abschnitts: Joelle (20%), Akram (60%), Tom (20%)

Die Batterie Top Tuning TT1539/650 weist typischerweise folgende Eigenschaften auf:

Sie verfügt über einen niedrigen Innenwiderstand bei hoher MPV. Die Energiedichte ist hoch, trotz des geringen Gewichts. Die Batterie weist gute Eigenschaften in Bezug auf ihre Lade- und Entladezyklen auf, bis zu 200 Zyklen nach Behandlung. Die Impulsbelastbarkeit liegt auf einem hohen Niveau. Unter normalen Bedingungen beträgt der Ladestrom 2C, kann jedoch in Ausnahmefällen auf 5C erhöht werden. [4]



Abbildung 6: Batterie TopTuning TT1530/650

Quelle: [4]

Die Spezifikationen der Batterie sind wie folgt: Spannung: 6s/ 22,2 V, Kapazität mAh: 5000, Gehäuse:

Soft, Kabel mm²: 10AWG/5,3, Länge +/- mm: 150 mm, Breite +/- 1,5 mm: 47mm, Höhe +/- 1,5 mm: 48mm, Gewicht +/- 10g: 742g, Dauerstrom: 60C(220A).

2.1.4 Jetson

Verfasser des Abschnitttextes: Akram (100%)
Bearbeiter des Abschnitts: Akram (50%), Tom (50%)

Das NVIDIA® Jetson Orin Nano™ Developer Kit ermöglicht die Entwicklung von KI-gesteuerten Robotern, intelligenten Drohnen und intelligenten Kameras auf Basis der Jetson Orin Nano-Serie. [5]

Die herausragenden Merkmale des Nvidia Jetson Orin Nano Developer Kits sind:

Hervorragende KI-Leistung: Dieses Kit bietet eine beeindruckende KI-Verarbeitung mit einer Leistung von bis zu 40 TOPS bei gleichzeitig niedrigem Energieverbrauch und geringer Latenz.

Handflächengroßes Smart-Edge-Gerät: Mit seinen kompakten Abmessungen von 100 x 79 x 21 mm präsentiert sich das Kit als handflächengroßes Smart-Edge-Gerät, das leistungsfähige KI-Verarbeitung in einem kleinen Formfaktor bietet.

Erweiterbar durch zahlreiche Ein- und Ausgänge:

USB 3.2 Gen 2-Anschlüsse: Das Gerät verfügt über 4 USB 3.2 Gen 2-Anschlüsse, die schnelle Datenübertragungsraten ermöglichen.

DisplayPort 1.2: Mit einem DisplayPort 1.2-Anschluss können Sie hochauflösende Displays anschließen und qualitativ hochwertige visuelle Inhalte genießen.

M.2 Key E mit WiFi-Modul: Die M.2 Key E-Schnittstelle bietet die Möglichkeit zur Integration eines WiFi-Moduls, was drahtlose Konnektivität ermöglicht.

M.2 Key M für SSD: Es sind zwei M.2 Key M-Anschlüsse für SSDs vorhanden, um die Speicherkapazität zu erweitern und die Datenübertragungsgeschwindigkeiten zu optimieren.

MIPI CSI: Zwei MIPI CSI-Anschlüsse sind verfügbar, was die Integration von Kameras erleichtert und hochwertige Bild- und Videodaten ermöglicht.

Gigabit Ethernet: Das Gerät ist mit Gigabit Ethernet ausgestattet, um schnelle und zuverlässige kabelgebundene Netzwerkverbindungen zu ermöglichen.

40-pin GPIO: Die 40-pin GPIO-Schnittstelle bietet eine Vielzahl von Ein- und Ausgangsmöglichkeiten für die Anbindung an verschiedene externe Geräte und Komponenten. [5] [6]

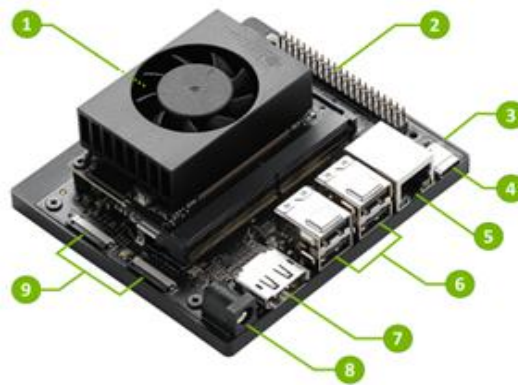


Abbildung 7: NVIDIA® Jetson Orin Nano™ Developer Kit mit nummerierten Pins (Legende darunter)

Quelle: [5]

Pinverteilung:

- | | |
|---------------------------------------|---------------------------------------|
| 1. microSD card slot for main storage | 2. 40-pin expansion header |
| 3. Power indicator LED | 4. USB-C port for data only |
| 5. Gigabit Ethernet port | 6. USB 3.2 Type A ports (x4) |
| 7. DisplayPort connector | 8. DC Barrel jack for 19V power input |
| 9. MIPI CSI camera connectors | |

Das folgende Pinbelegungsdiagramm des Jetson Nano Developer Kits zeigt die verschiedenen Verwendungsmöglichkeiten der Pins:

Sysfs	Name	Pin	Pin	Name	Sysfs
	3.3V DC	1	2	5V DC	
	I2C_2_SDA	3	4	5V DC	
	I2C_2_SCL	5	6	GND	
gpio216	AUDIO_MCLK	7	8	UART_2_TX	
	GND	9	10	UART_2_RX	
gpio50	UART_2_RTS	11	12	I2S_4_CLK	gpio79
gpio14	SPI_2_SCK	13	14	GND	
gpio194	LCD_TE	15	16	SPI_2_CS1	gpio232
	3.3V DC	17	18	SPI_2_CS0	gpio15
gpio16	SPI_1_MOSI	19	20	GND	
gpio17	SPI_1_MISO	21	22	SPI_2_MISO	gpio13
gpio18	SPI_1_SCK	23	24	SPI_2_CS0	gpio19
	GND	25	26	SPI_2_CS1	gpio20
	IC2_1_SDA	27	28	I2C_1_SCL	
gpio149	CAM_AF_EN	29	30	GND	
gpio200	GPIO_PZO	31	32	LCD_BL_PWM	gpio168
gpio38	GPIO_PE6	33	34	GND	
gpio76	I2S_4_LRCK	35	36	UART_2_CTS	gpio51
gpio12	SPI_2_MOSI	37	38	I2S_4_SDIN	gpio77
	GND	39	40	I2S_4_SDOUT	gpio78

Abbildung 8: Pinverteilung Jetson

Quelle: [6]

2.1.5 Servo-Motor

Verfasser des Abschnitttextes: Awat (75%), Tom (25%)

Bearbeiter des Abschnitts: Awat (15%), Akram (40%), Tom (45%)

Als Servomotor werden spezielle Elektromotoren bezeichnet, die die Kontrolle der Winkelposition ihrer Motorwelle sowie der Drehgeschwindigkeit und Beschleunigung erlauben. Sie bestehen aus einem Elektromotor, der zusätzlich mit einem Sensor zur Positionsbestimmung ausgestattet ist.

Ein Servomotor arbeitet als Teil eines geschlossenen Regelkreissystems, das ein Drehmoment und eine Geschwindigkeit liefert, die von einem Servoregler vorgegeben werden, der ein Rückführsystem zum Schließen des Regelkreises nutzt.

Der Servo-Aufbau eines Antriebssystems besteht im Wesentlichen aus Stator und Rotor bzw. der Kombination aus Motorwicklung und Motorwelle. Angetrieben wird der Motor durch elektrische Energie, die die Wicklung durchströmt und so ein elektromagnetisches Feld erzeugt, das die Welle in Rotation versetzt. [7]



Abbildung 9: Servo K-Power DM4000
Quelle: [7]

Spezifikationen:

- Modell: K-Power DM4000
- Betriebsspannung: 6 Volt / 7,4 / 8,4 Volt
- Strom: 0,5 A
- Drehmoment: (Ncm): 420/500/540
- Geschwindigkeit: (sec/60°): 0,155/0,13/0,11

2.1.6 Kamera

Verfasser des Abschnitttextes: Awat (90%), Tom (10%)
Bearbeiter des Abschnitts: Awat (100%)

Die Intel® RealSense™ Depth Camera D435i ist eine fortschrittliche Tiefenkamera, die entwickelt wurde, um die Welt um uns herum dreidimensional zu erfassen. Sie verwendet eine Kombination aus Infrarot- und RGB-Sensoren, um hochpräzise Tiefeninformationen zu liefern und so eine detaillierte Darstellung der Umgebung zu ermöglichen. Ein Beispiel für die Anwendung der D435i wäre die Erstellung von 3D-Modellen von Objekten oder Räumen. Indem sie die Tiefeninformationen erfasst und mit den Farbinformationen kombiniert, kann die Kamera präzise Modelle erstellen, die in verschiedenen Bereichen wie Architektur, Medizin oder Robotik eingesetzt werden können.

Eine Statistik, die verdeutlicht, wie fortschrittlich die Technologie der D435i ist, ist ihre Fähigkeit, eine Tiefenauflösung von bis zu 1280x720 Pixeln bei 30 Bildern pro Sekunde zu erreichen. Dies ermöglicht eine äußerst präzise Erfassung der Umgebung und eine genaue Erstellung von 3D-Modellen. [8]

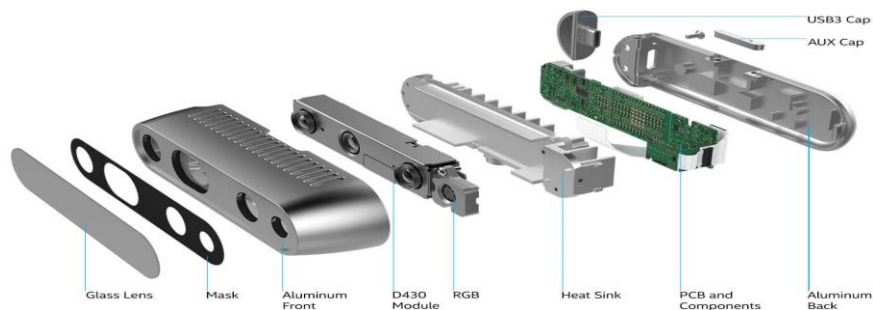


Abbildung 10: Kamera Intel® RealSense™ Depth Camera D435i
Quelle: [8]

Spezifikationen:

- Bis zu 1280 x 720 aktive Stereo-Tiefenauflösung
- Bis zu 1920x1080 RGB-Auflösung
- Tiefendiagonales Sichtfeld über 90°
- Duale Global-Shutter-Sensoren für Tiefenstreaming mit bis zu 90 FPS
- Reichweite 0,2 m bis über 10 m (variiert je nach Lichtverhältnissen)

2.1.7 IMU

Verfasser des Abschnitttextes: Joelle (90%), Tom (10%)

Bearbeiter des Abschnitts: Joelle (35%), Akram (35%), Tom (30%)

Das ACEINNA MTLT305D () ist ein dynamisches Neigungssensormodul mit CAN 2.0- (J1939) und RS232-Schnittstellen für vielseitige Konnektivität. Es integriert Beschleunigungsmesser, Gyroskope und einen Temperatursensor mit hochentwickelten Kalibrierungs- und Korrekturalgorithmen. Dadurch ermöglicht es präzise 3D-Beschleunigungsmessungen, 3D-Geschwindigkeitsmessungen sowie Nick- und Rollwinkelmessungen in dynamischen Fahrzeuganwendungen. Durch Nutzung fortschrittlicher Sensorfusionstechnologien wie erweiterte Kalman-Filterung und Kalibrierungsalgorithmen erreicht es eine Neigungsgenauigkeit von 0,5 Grad und eine Beschleunigungsgenauigkeit von weniger als 0,1% der Gewichtskraft unter verschiedenen dynamischen Bedingungen. Das Modul unterstützt einen breiten Versorgungseingangsbereich von 4,9 V bis 32 V, was es ideal für den Einsatz in 12-V- und 24-V-Fahrzeugplattformen macht. [9]



Abbildung 11: IMU
ACEINNA MTLT305D
Quelle: [9]



Abbildung 12: Pinverteilung IMU
Quelle: [9]

2.1.8 DC-/DC-Wandler

Verfasser des Abschnitttextes: Joelle (100%)

Bearbeiter des Abschnitts: Joelle (40%), Akram (20%), Tom (40%)

Der DC/DC (Abbildung 12) 8-40 zu 12V 20A Spannungswandler Converter (EAN: 4262375690064) ist ein Gleichspannungswandler, der Gleichspannungen von 8 – 40 V zu 12 V umwandelt. Der Wandler verfügt über sehr interessante Eigenschaften: Er ist durch sein Gehäuse sehr robust und kann starken Vibrationen standhalten. Der Wandler kann über verschiedene Schnittstellen angeschlossen werden, wie beispielsweise USB. Das Gerät muss nicht aktiv gekühlt werden und arbeitet außerdem sehr leise. Neben Anwendungsgebieten wie Photovoltaik-Anlagen ist dieser Wandler auch in Fahrzeugen oder allgemein in Antriebssystemen zu finden, wo verschiedene Spannungsbereiche (hoch, niedrig, negativ) benötigt werden. [10]



Abbildung 13: DC-/DC-Wandler Bauer Electronics DC-DC 8-40V-12V - 20A
Quelle: [10]

2.1.9 Lüfter

Verfasser des Abschnitttextes: Shahrzad (50%), Tom (50%)

Bearbeiter des Abschnitts: Tom (100%)

Der Lüfter JD6025B8HH hat die Aufgabe, elektronische Geräte zu kühlen und vor Überhitzung zu schützen, im Falle des Projekts, einen Elektromotor. Sie können die warme Luft aus dem Inneren des Geräts abtransportieren und kühlere Luft ansaugen, um die Temperatur auf einem optimalen Niveau zu halten. [11]

Spezifikationen:

- Spannung: 8,4 V
- Strom: 0,5 A
- Leistung: 4,6 W
- Drehzahl: 8500 rpm



Abbildung 14: Lüfter Jiabofan JD6025B8HH
Quelle: [11]

2.1.10 Positionsgeber

Verfasser des Abschnitttextes: Shahrzad (75%), Tom (25%)

Bearbeiter des Abschnitts: Shahrzad (10%), Tom (90%)

Die AMS AS5x47P-Motorplatine ist eine einfache Leiterplatte, die an BLDC- oder Schrittmotoren zur Positionsbestimmung angeschlossen werden kann. Es ermöglicht eine einfache und schnelle

Evaluierung. Die Steckverbinder JP1 ermöglicht die Auswahl zwischen 5-V- oder 3,3-V-Betrieb. Der Sensor und alle erforderlichen externen Komponenten sind bereits auf die Leiterplatte gelötet. [12]



Abbildung 15: Positionsgber AMS Osram AS5X47P-TS_EK_MB

Quelle: [12]

2.1.11 Antenne

Verfasser des Abschnitttextes: Awat (90%), Tom (10%)

Bearbeiter des Abschnitts: Akram (100%)

Der TP-Link AX1800 ist ein WLAN-Router, der den neuen Wi-Fi 6 Standard unterstützt. Wi-Fi 6 ist die neueste Generation von drahtlosen Netzwerken und bietet eine verbesserte Geschwindigkeit, Kapazität und Effizienz im Vergleich zu früheren Standards.

Eine überprüfbare Tatsache ist, dass Wi-Fi 6 die Übertragungsgeschwindigkeiten im Vergleich zu älteren Standards erheblich verbessert. Es kann eine maximale Geschwindigkeit von bis zu 1,8 Gbit/s bieten, was mehr als das Doppelte der Geschwindigkeit von Wi-Fi 5 (dem vorherigen Standard) ist. [13]

Spezifikationen:

- Dimension: 84 × 156.3 × 19.2 mm
- Antenne: 2× High-Gain Dual-Band Antennas
- WLAN-Standards: IEEE 802.11a/b/g/n/ac/ax
- Sendeleistung: 5 GHz: 19dBm (FCC) /16dBm(CE) (EIRP)
2.4 GHz : 19dBm(FCC) / 14.5dBm(CE) (EIRP)
- WLAN-Modi: Infrastructure mode
- WLAN-Sicherheit: WEP, WPA/WPA2/WPA3, WPA-PSK/WPA2-PSK



Abbildung 16: Antenne TP-Link AX1800

Quelle: [13]

2.1.12 Zusammenfassung der technischen Daten

Verfasser des Abschnitttextes: Tom (100%)

Bearbeiter des Abschnitts: Akram (20%), Tom (20%), Shahrzad (20%), Joelle (20%), Awat (20%)

Die Daten über die Betriebsströme und -spannungen, Ausgabespannungen, als auch die verfügbaren Schnittstellen werden in der folgenden Tabelle 1 gesammelt.

Tabelle 1: Technische Daten der Komponenten

Komponente	Typ:	Spannung-Input [V]	Strom [A]	Spannung-Output [V]	Schnittstellen
Motor	Surpass Hobby 5692 Brushless Motor 1090/980/730KV Shaft 8mm for 1/5 RC Car RTR	max. 27	max. 197	-	PWM (U, V, W)
ESC (Umrichter)	Flipsky FSESC 6.9	14 - 60	100 (dauerhaft), 200 (Peak)	5	PWM, USB, CAN, HALL
Batterie	Top Tuning TT1530/650	22,2	max. 200	22,2	+ und -
Jetson	Jetson Orin NX AI	9 - 20	1,5	5	USB, LAN, GPIO, PWM, CAN, HDMI, RS232
Servo	K-Power DM4000 Servo	6,0 - 8,4	0,5	-	PWM
Lüfter	JIABOFAN JD6025B8HH	8,4	0,5	-	+ und -
Kamera	Intel® RealSense™ Depth Camera D435i	5	0,7	-	USB
IMU	ACEINNA MTLT305D	4,9 - 32,0	0,5		CAN, RS232
Positionsgeber	ams OSRAM AS5X47P-TS_EK_MB	5	max. 0,5	-	Digital In-/Output
Antenne	TP-link AX1800	5	0,7	-	USB
DC-/DC-Wandler	Bauer Electronics DC-DC 8-40V-12V - 20A	8 - 40	20	12	+ und -

2.3 Erarbeitung des Schaltplans

Verfasser des Abschnittstextes: Tom (100%)

Bearbeiter des Abschnitts: Tom (80%), Akram (20%)

Nachdem zu allen am Projekt beteiligten Komponenten die technischen Daten um Spannung, Strom, Schnittstellen und Pinverteilung herausgearbeitet wurden, kann der Schaltplan für die Verkabelung des Modellfahrzeugs erstellt werden. Dieser ist erforderlich, um den genauen, theoretischen Aufbau der Verkabelung des Modellautos zu bestimmen. D.h. es soll ein detaillierter Überblick entstehen, welche Einzelverbindungen zwischen den Komponenten erforderlich sind, mit welchem Strom und Spannung eine Komponente versorgt werden muss und mit welcher Schnittstelle die Kommunikation zwischen zwei Geräten ablaufen soll. Zudem ist darüber herauslesbar, mit welchem Anschluss eine Leitung an eine Komponente verbunden ist.

2.3.1 Versorgung der Komponenten

Hierzu werden zuerst die Betriebsspannungen aus Tabelle 1 überprüft. Dabei ist ersichtlich, dass Motor, Umrichter und DC-/DC-Wandler direkt mit der Batterie versorgt werden können. Denn alle drei Komponenten können im Bereich der Batteriespannung von 22,2 V betrieben werden. Alle anderen Geräte liegen jedoch deutlich unter der Batteriespannung. Folglich sind die meisten Komponenten über einen DC-/DC-Wandler zu betreiben, welcher zu den vorgegebenen Komponenten zählt. Da die Betriebsspannung des Jetsons im Bereich der Ausgangsspannung des DC-/DC-Wandlers liegt, soll dieser darüber betrieben werden. Die anderen Komponenten liegen jedoch unterhalb der Ausgangsspannung des Wandlers. Ein Betrieb mit zu hoher Spannung würde zur Beschädigung der angeschlossenen Geräte führen. Demzufolge ist es unumgänglich, den Servo, die IMU, den Positionsgeber und den Lüfter anders zu versorgen.

Durchleuchtet man die Tabelle 1, fällt auf, dass der Umrichter und der Jetson jeweils Spannungsausgänge mit 5 V aufweisen. Diese können für die IMU und den Positionsgeber angewandt werden. So wird die IMU über den Jetson betrieben, der Positionsgeber über den Umrichter. Servo und Lüfter liegen jedoch über den 5 V. Deshalb muss ein Kompromiss getroffen werden. Da die Leistungen beider Geräte sehr gering sind, sollen sie mit geringerer Spannung über den Umrichter betrieben werden. Die geringere Leistungsfähigkeit in Form von geringerer Drehzahl wurde durch den Projektbetreuer abgesegnet.

Weiter ist die Spannungsversorgung des Motors, Umrichters und DC-/DC-Wandlers festzulegen. Würde man nur eine Batterie verwenden, könnte man alle drei Komponenten problemlos daran anschließen, da deren Betriebsspannungen und -ströme sich im Bereich des Akkumulators befinden. Da jedoch zwei gleiche Batterien Anwendung finden sollen, muss eine Verschaltung ausgewählt werden. Möglich ist eine Parallel- oder Reihenschaltung.

Bei einer Parallelschaltung würde die Spannung bei 22,2 V verharren, während sich der max. abgebbare Strom der Batterien auf bis zu 400 A verdoppelt. Umgekehrt verhält sich die Reihenschaltung, d.h. es würde eine Spannung von 44,4 V anliegen und ein Strom von bis zu 200 A fließen. Da der Motor nur mit max. 27 V betrieben werden darf, wird auf eine Reihenschaltung verzichtet und die Parallelschaltung beider Batterien in Erwägung gezogen. Grund dafür ist, dass die Spannung identisch bleibt.

Um nun die angeschlossenen Geräte, erst recht den über den Umrichter angeschlossenen Jetson (sehr teuer), im Fehlerfall, z.B. durch einen Kurzschluss, nicht durch zu hohe Ströme zu beschädigen, wird in die Plusleitung der Batterie (nach der Parallelschaltung) eine Sicherung verbaut. Weiter soll diese

auslösen, falls der Motor bzw. Umrichter fehlerhaft (z.B. durch Softwarefehler) zu lange im Überlastbereich (über 100 bis zu 197 A) oder darüber hinaus betrieben werden, um eine zu starke Erhitzung und/oder Beschädigung von Motor und Umrichter zu vermeiden. Gleichzeitig werden die Leitungen vor Überlast geschützt.

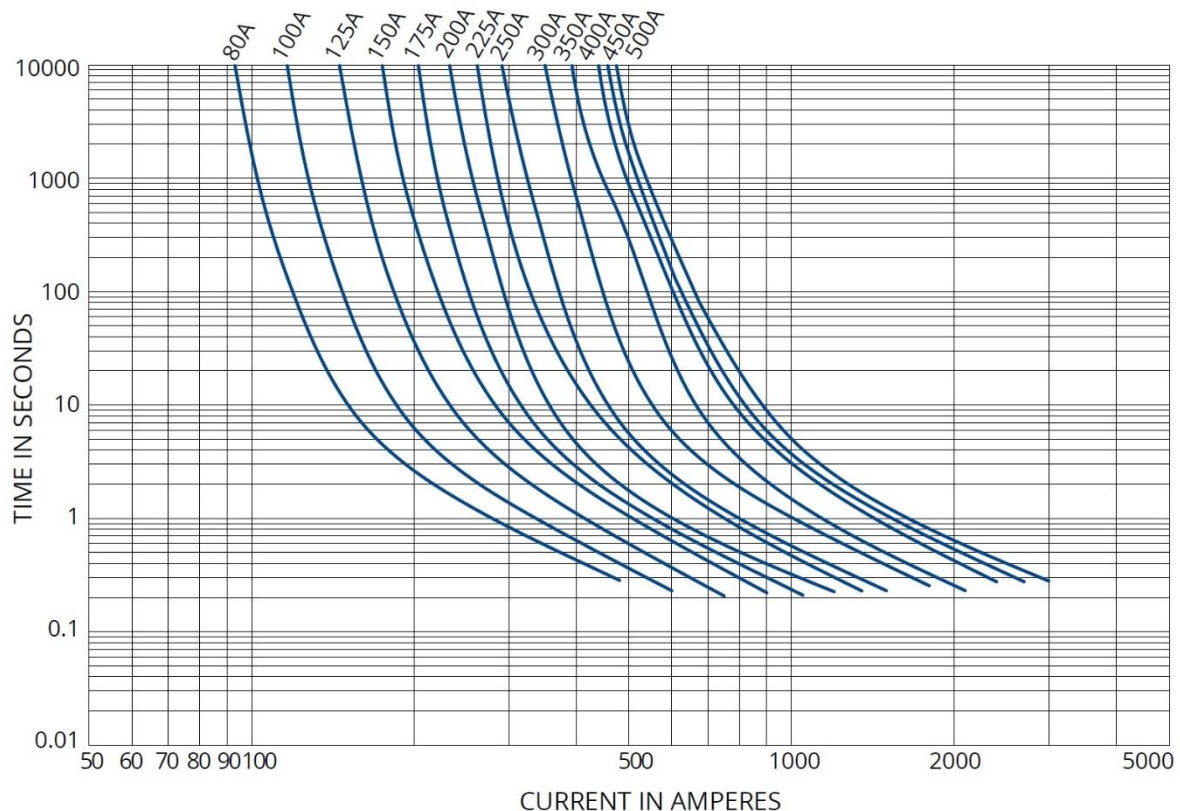


Abbildung 17: Sicherungskennlinien
Quelle: [14]

Dazu wird eine Sicherung mit einem Auslösestrom von rund 125 – 150 A in Erwägung gezogen. Da eine 150 A Sicherung bereits vorhanden ist, wird die dazugehörige Sicherungskennlinie herangezogen und überprüft, *siehe* Abbildung 17. Zum Vorteil sind dort auch kleinere und größere Modelle eingezeichnet.

Wie man erkennen kann, ist mit der 150 A Sicherung ein Volllastbetrieb (bis knapp 200 A) ca. 400 bis 500 Sekunden möglich. Beim 125 A Modell sind es 30 bis 40 Sekunden, bei 175 A wären es nahezu 10.000 Sekunden. Nach kurzer Rücksprache mit dem Projektbetreuer soll die vorliegende 150 A Sicherung angewandt werden, da Volllastfahrten für mehrere Minuten durchaus angedacht und die Auslösezeiten für höhere Ströme akzeptabel sind. Die 175 A Lösung ist hingegen zu gefährlich, da diese für den Anwendungszweck des Modellautos sehr hohe Ströme für zu lange Zeiten zulässt.

Die anderen grundlegenden Spezifikationen, wie der zulässige Temperaturbereich, passen zum Einsatzzweck des Modellfahrzeugs. Grundsätzlich handelt es sich dabei auch um eine Automotive-Sicherung. Das Datenblatt der Sicherung ist im bereitgestellten Verzeichnis zu finden (Dateiname: Datenblatt Sicherung PEC.pdf). [14] [15]

2.3.2 Bestimmung der zu verwendenden Schnittstellen

Nun sind die Schnittstellen zu wählen, d.h. die Methode, wie zwei Komponenten miteinander kommunizieren (z.B. Sensor mit Jetson oder Umrichter mit Motor). Hierzu wird anhand der Spalte „Schnittstellen“ der *Tabelle 1* überprüft, welche Schnittstellen eine Komponente überhaupt besitzt.

Anschließend schaut man mithilfe des Strukturbilds, welche beiden Komponenten eine Kommunikationsverbindung aufweisen müssen und wählt daraufhin die Schnittstelle, welche **beide** Geräte besitzen. Grundsätzlich haben die Sensoren nur eine (z.B. USB), wodurch die Auswahl sehr leicht ist.

Jedoch gibt es auch Komponenten, die mehrere Schnittstellen haben (IMU, Jetson und Umrichter). Folgerichtig muss daraus eine zur Verwendung ausgewählt werden. Dies geschieht in Absprache mit dem Software-Team, indem beide Seiten offenlegen, welche Schnittstelle für sie einfacher umzusetzen ist. Hierbei konnte man sich in allen drei Fällen sofort einigen, da sich die Gründe der Hardware- (keine bis wenig Zusatzhardware und wenig zusätzlicher Leitungsaufwand) und der Software-Gruppe (einfachere Implementierung) überschneiden. So kommunizieren Jetson und Umrichter über USB, die IMU via RS232.

2.3.3 Erstellung des Schaltplans

Nachdem die Vorarbeiten für den Schaltplan erledigt sind, kann dieser nun, unter Berücksichtigung der vorher erarbeiteten Ergebnisse, erstellt werden. Als Basis hierzu dient das Strukturbild (Abbildung 2). Allerdings werden nun für den Schaltplan zwischen allen Komponenten, die miteinander kommunizieren, die Einzelleitungen aufgetragen und die in 2 bestimmten Pinverteilungen berücksichtigt. Dass bedeutet beispielsweise, eine Spannungsversorgung wird nun durch zwei Leitungen dargestellt (+ und -) oder die RS232-Schnittstelle mit RS232 RX und RS232 TX.

Der Schaltplan kann der *Abbildung 18* entnommen werden. Dieser wurde mit der Software Wondershare EdrawMax erstellt. Die Einzeldatei kann auch hier dem beigefügten Verzeichnis entnommen werden (Dateiname: Schaltplan.eddx).

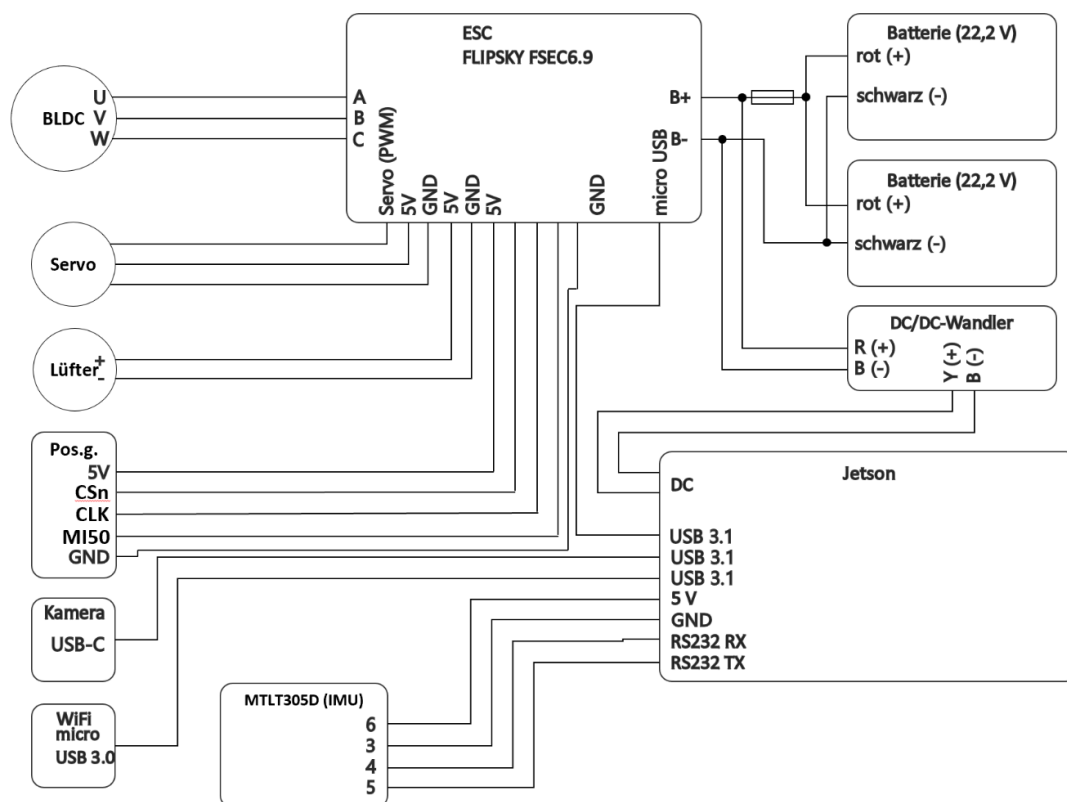


Abbildung 18: Schaltplan des Modellfahrzeugs
Quelle: Eigene Darstellung

2.4 Festlegung der vorläufigen Leitungsverläufe

Verfasser des Abschnitttextes: Tom (85%), Akram (15%)

Bearbeiter des Abschnitts: Tom (67,5%), Akram (27,5%), Awat (5%)

Um im folgenden Abschnitt die Leitungslängen und die Querschnitte der Kabel zu ermitteln, sind jetzt die Leitungsverläufe, d.h. die Positionen einer einzelnen Leitung im Fahrzeug zu bestimmen. Dazu werden im folgenden Unterabschnitt zuerst einige Rahmenbedingungen festgelegt.

2.4.1 Festlegung der Rahmenbedingungen

Hierbei ist besonders darauf zu achten, möglichst viele der Leitungen auf einer großen Strecke in einen oder mehrere Kabelbäume zusammenzufassen, dass bedeutet, mehrere Leitungen zu einem Strang zu bündeln. Dies soll auch möglichst bei vom Hauptkabelbaum abzweigenden Leitungen zu Komponenten realisiert werden, sodass z.B. dem Jetson nicht von mehreren Seiten Leitungen zugeführt werden, sondern ein kleiner Strang (Nebenstrang), ausgehend vom Hauptkabelbaum, dahinführt und sich erst kurz vor der Komponente zu den Anschlüssen in Einzelleitungen aufteilt. Abbildung 19 veranschaulicht dieses Kriterium.

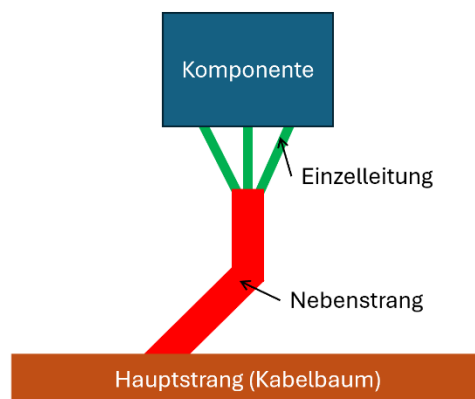


Abbildung 19: Aufzweigung eines Kabelbaums

Quelle: Eigene Darstellung

Dieser Schritt ist notwendig, weil das aufzubauende Modellfahrzeug möglichst stark dem eines realen Fahrzeugs ähneln soll. Auch hier wird in großem Umfang auf Kabelbäume gesetzt. In der folgenden Abbildung 20 ist ein KFZ-Kabelbaum zu erkennen.



Abbildung 20: Kabelbaum eines PKW

Quelle: [16]

Anderweitig kann durch Kabelbäume der Platzbedarf reduziert werden, sie sorgen für eine bessere Übersichtlichkeit im Modellfahrzeug (sonst viele verstreute Leitungen) und bieten eine einfachere Befestigung. Zusätzlich erleichtert ein dicker Kabelbündel die Führung der Leitungen, verbessert die Reparierbarkeit und minimiert die Suchzeit nach Fehlern. Nicht zu vergessen ist die höhere Robustheit gegen äußere Einflüsse, wie Vibrationen, mechanische Stöße oder Umwelteinflüsse, welche gerade sehr dünne, einzelne Leitungen betreffen. Zudem werden weniger Befestigungsobjekte benötigt, was zusätzlich, im geringen Maß, das Gewicht und die Kosten reduziert.

Deshalb ist auch darauf zu achten, dass der Baum keinerlei Stecker und Lötverbindungen beinhaltet (müssen außerhalb platziert werden). Sonst würde der Kabelbaum gegen die eben aufgestellten Bedingungen verstoßen, indem er aufgrund der Steckerabmaße nicht mehr platzsparend zu verlegen, sowie einfach zu befestigen wäre und die Übersichtlichkeit einschränken würde.

Ergänzend ist auf möglichst symmetrische Leitungsverläufe zu achten. Zudem sollen Überkreuzungen einzelner Leitungen, erst recht an Kabelbäumen, möglichst vermieden werden. Dieser Schritt ist notwendig, um einerseits die Übersichtlichkeit zu gewähren, andererseits um EMV-Störungen (elektromagnetische Verträglichkeit), gerade bei Kreuzungen von Signal- und Versorgungs-/Motorleitungen, zu verhindern. Erst recht durch die hohen Ströme und Frequenzen in den Motorkabeln besteht durchaus die Möglichkeit, dass die elektromagnetischen Wechselfelder Störströme in den Sensorleitungen induzieren. Hierdurch kann es zu unerwünschten Fehlfunktionen wie z.B. einer nicht funktionierenden, falsch reagierenden oder automatisch einsetzenden Lenkung kommen. Kabelkreuzungen sind allerdings nie ganz zu vermeiden.

Aus der EMV-Thematik kann auch abgeleitet werden, dass mehrere Kabelbäume erforderlich werden. So gibt es im Modellfahrzeug mehr oder weniger zwei grundlegende Spannungsniveaus. Einmal für die Motorleitungen (wegen hohen Frequenzen aus Pulsweitenmodulation besonders kritisch in Hinblick auf EMV-Störungen bei Signalleitungen) und Versorgungsleitungen von Umrücker und DC-/DC-Wandler mit rund 22,2 V und sehr hohen Strömen bis zu 197 A. Andererseits die Signal- und Versorgungsleitungen (meist in einem Kabel kombiniert) von Lüfter, Servo, Sensoren etc. mit Spannungen im Bereich von 3,3 bis 9 V und Strömen von einigen mA bis zu 2 A. Folglich wird es einen Kabelbaum für hohe Spannungen und einen für niedrigere Spannungen geben. Ob eine Abschirmung der 22,2 V Leitungen, gerade der drei Motorleitungen, nötig ist, soll nach einen ausführlichen Funktionstest des Fahrzeugs geklärt und ggf. nachgerüstet werden.

Jedenfalls wären für die Umsetzung entweder geschirmte Leitungen erforderlich oder die Leitungen bzw. Kabelbäume müssten mit einem leitenden Gegenstand, z.B. Alufolie, eingewickelt werden. Im Falle einer Einhüllung mit einem leitenden Material wäre für eine bessere Abschirmung zudem ein Masseverbindung dahin sinnvoll.

In Sachen Kabelverlängerungen bzw. Anschluss von Komponenten dürfen zudem keine Lötverbindungen angewandt werden. Einerseits, um im Fehlerfall (z.B. falsche Ansteuerung) schnell eine Komponente abtrennen zu können, gerade für die allgemeine Stromversorgung (Batterien) sinnvoll, andererseits um defekte Leitungen schneller austauschen zu können. Zudem besteht bei Lötverbindungen die Gefahr, dass sie durch zu hohe Leitungstemperaturen wiederaufschmelzen oder durch Vibrationen bei der Fahrt brechen. [15] [16]

2.4.2 Bestimmung der Leitungsverläufe

Für einen ersten Entwurf der Leitungsverläufe wird auf die CAD-Daten des Autos zurückgegriffen, welche im Rahmen eines Masterprojekts eines anderen Studenten erstellt wurden und angewandt werden dürfen. Hierbei ist zuerst wichtig, freie Plätze für die Kabelbäume, später auch die

Einzelleitungen und Stecker zu finden. Denn die genauen Positionen der Komponenten wurden durch das genannte Masterprojekt bereits vergeben. So gibt es eine Grundplatte, auf welcher sich die Batterien, Motor, Servo, als auch die gesamte Aufhängung befinden und eine zweite Platte, welche 15 cm höher montiert ist und beidseitig genutzt wird. Allerdings sind die meisten Komponenten daran auf der Unterseite montiert (u.a. Jetson und Umrichter).

Beide Platten werden im Folgenden von der Vogelperspektive betrachtet und freie Räume zwischen den Komponenten gesucht. Abbildung 21 zeigt dabei die Vogelperspektive der Grundplatte. Hierbei sind die freien Flächen rötlich gekennzeichnet.

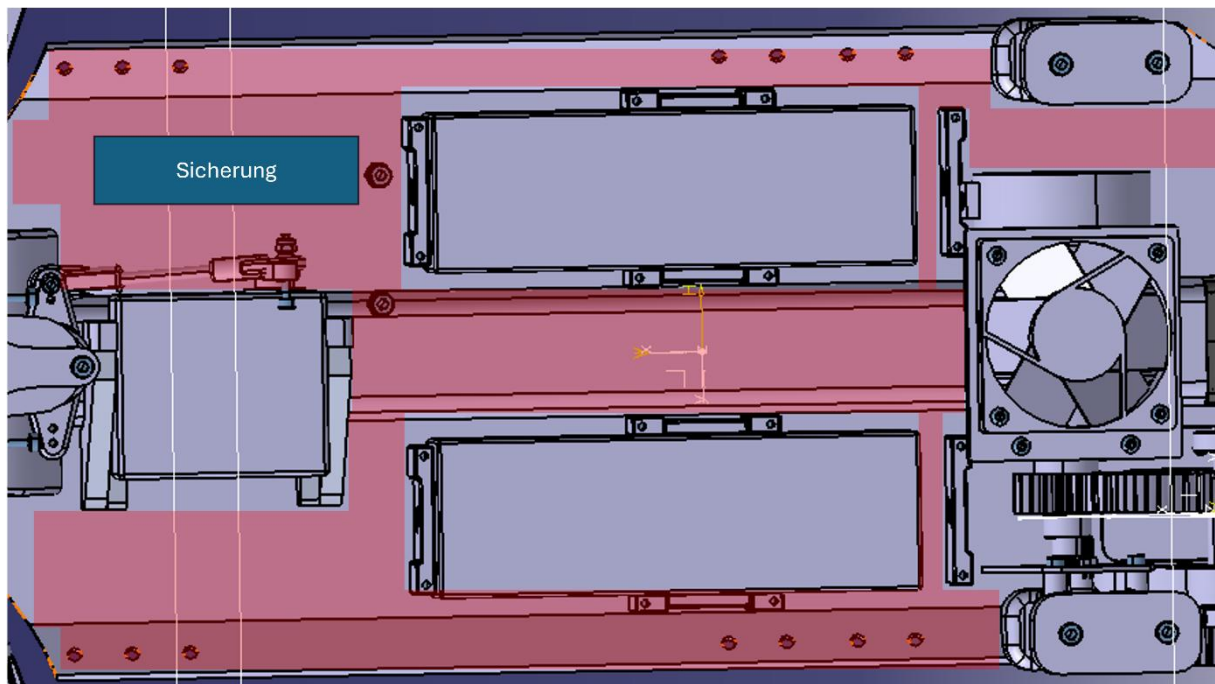


Abbildung 21: Vogelperspektive der Grundplatte mit freien Flächen
Quelle: Eigene Darstellung

Abbildung 22 stellt die Draufsicht auf die Unterseite der oberen Platte dar, freie Flächen ebenfalls in rot markiert.

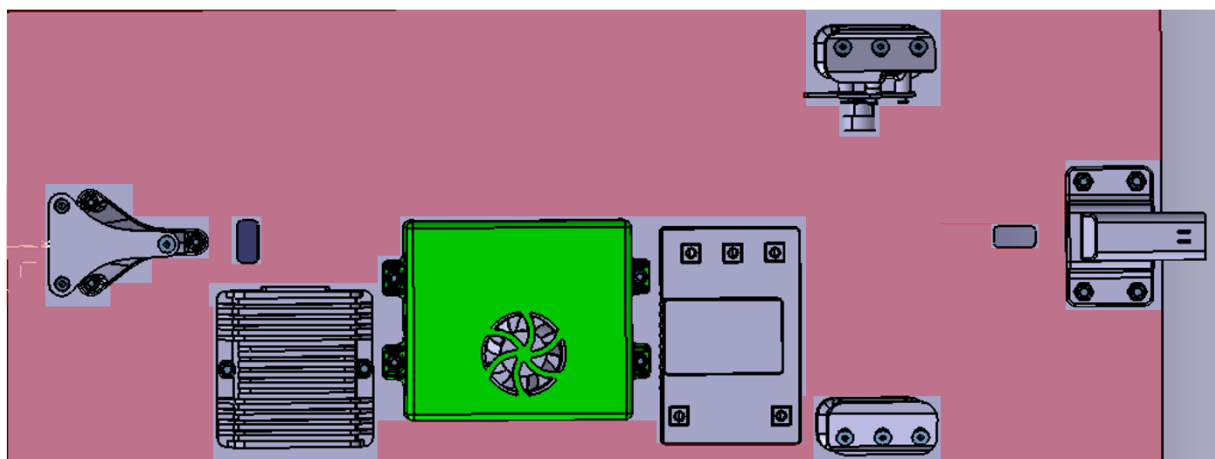


Abbildung 22: Vogelperspektive der Unterseite der oberen Platte mit freien Flächen
Quelle: Eigene Darstellung

1.1.1.1 Grundplatte

Wie man in Abbildung 21 erkennen kann, ist auf der Grundplatte fast über die gesamte Fahrzeuglänge in der Mitte ein Freiraum (auf der Verkleidung des Antriebsriemens). Dieser Freiraum soll künftig als Position für die unteren beiden Kabelbäume dienen. Da alle Komponenten von unten mit dem Umrichter, Jetson oder DC-/DC-Wandler verbunden werden (liegen auf der Unterseite der oberen Platte), ist eine Führung der Kabel nach oben erforderlich. In aufgeräumter Form ist dies nur über eine der beiden hinteren Säulen möglich. Jedoch ist die linke bereits mit dem Positionssensor verbaut, die rechte aber frei, womit an dieser eine Leitungsführung vorstellbar ist. Vorteilhaft ist der ebenfalls freie Platz zwischen rechter Säule und dem mittigen Freiraum, weshalb so künftig die Kabelführung verlaufen soll. Dankenswerterweise wird an der rechten Säule eine Halterung zur Befestigung der Kabelbäume angebracht.

Da die Position der Sicherung bereits vorgegeben ist, gilt es nun, die Leitungsführung der Batterien dahin zu bestimmen. Beide sollen weiterhin mittels des angelöteten EC8-Steckers an- und abgesteckt werden können (dementsprechend ist die Leitungslänge der ausgehenden Batterieleitungen nicht veränderbar). Folgendermaßen ist jeweils ein Adapterstück zwischen Sicherung und einer Batterie erforderlich, welches die EC8-Buchse besitzt. Somit muss zweimal eine EC8-Steckverbindung (min. 5 cm Länge) untergebracht werden, welche nicht in der Mitte im Kabelbaum liegen darf und auch in der Nähe der Sicherung keinen Platz findet. Diese Steckverbindung soll deshalb jeweils im Freiraum an den Außenseiten der Batterie, seitlich hochkant, untergebracht werden, wodurch die Batterie mit Kabelausgang in Richtung Fahrzeugheck zeigt. Die Leitung wird also um die Batterie herum nach vorne geführt. Bei der linken Batterie kommt zudem noch ein schräger Verlauf auf die rechte Seite hinüber hinzu.

Aufgrund größerer Querschnitte der Leitungen (erwartet wegen hoher Ströme), folglich eher unflexibel zu verlegen, und der Kabelschuhe, welche zum Anschluss an die Sicherung bzw. zum Masseknotenpunkt benötigt werden, müssen zudem größere Schleifen miteingeplant werden. Infolgedessen führt die Plusleitung der rechten Batterie in einem Bogen um die Batterie herum, bis sie in der Sicherung endet. Ähnlich ist es bei der Masseleitung der Batterie, da diese nur von vorne an den Massepunkt angebunden werden kann.

Der Kabelbaum mit der höheren Spannung entspricht folglich am hinteren Ausgang der Sicherung (um weniger Leitungen zu kreuzen) bzw. dem Masseknoten, läuft schräg zur Fahrzeugmitte, dort entlang und wird im hinteren Teil zur Säule abgeführt. Er besteht jeweils aus einer Plus- und Minus-Leitung für die Verbindung zum Umrichter und zum DC-/DC-Wandler.

Wie man an der Abbildung 21 erkennen kann, entspringen die Leitungen des Motors seitlich, ungefähr auf der Höhe der rechten Batterie. Diese müssen zum Umrichter geführt werden. Infolgedessen sind die Leitungen auch zur rechten Säule zu führen und in den 22,2 V Kabelbaum zu integrieren. Da allerdings jede der drei Leitungen eine Steckverbindung benötigt, die zudem schnell erreichbar sein muss, ist eine besondere Führung zu suchen.

Anhand des 3D-Modells ist ein Freiraum unter dem Antrieb erkennbar, weshalb die Leitungen mit einer Schleife unter diesem hindurchführen sollen. Die Steckverbindungen folgen anschließend direkt dem Ende des Antriebs, dort wo die Leitungen unterhalb des Antriebs heraustreten. Nach den Steckverbindungen werden die drei Leitungen zur Führung zum Umrichter in den 22,2 V Kabelbaum integriert.

Der Signalkabelbaum entspringt im Freiraum links neben dem Servo und führt anschließend im gleichen Verlauf neben dem anderen Baum, kleiner Abstand wird eingehalten, zur Säule. Auf Höhe des Antriebs wird das einzelne Servokabel um dass des Lüfters und des Positionsgebers ergänzt.

Zur Verdeutlichung der Leitungs- und Kabelbaumpositionen auf der Grundplatte folgt Abbildung 23 mit farbiger Codierung.

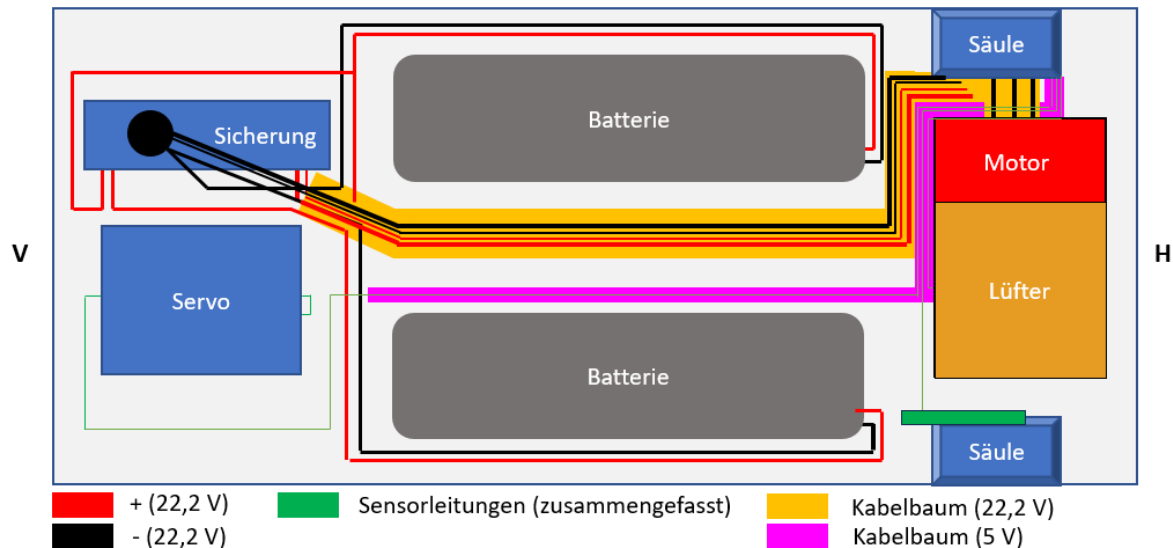


Abbildung 23: Skizze der Grundplatte mit den vorläufigen Leitungs- und Kabelbaumverläufen
Quelle: Eigene Darstellung

1.1.1.2 obere Platte (Unterseite)

Die beiden Kabelbäume führen von der Grundplatte über die hintere, rechte Säule nach oben auf die Unterseite der oberen Platte. Dort sollen beide Kabelbäume parallel neben dem Umrichter in Richtung Plattenmitte verlaufen, da dies der einzige freie Platz ist. Vorher müssen jedoch alle Leitungen, die zum Umrichter führen sollen (außer Motoranschlusskabel) ausgelagert werden (aufgrund der Anschlusspositionen des Umrichters). Dass bedeutet Batteriekabel, ebenso die Leitungen von Lüfter und Servo. Der Signalkabelbaum wird wenig später jedoch um die Antennen- und IMU-Leitungen ergänzt.

Die beiden Kabelbäume sollen weiter in Richtung Mitte der Platte und anschließend, nach einer möglichst rechtwinkligen Kurve in Längsrichtung der Platte verlaufen. Wie man in Abbildung 22 erkennen kann, ragen der Jetson und der Umrichter in die Plattenmitte hinein. Zusätzlich muss man bedenken, dass die Anschlüsse nochmals ein paar Zentimeter in Anspruch nehmen. Deshalb werden die Kabelbäume nicht mittig auf der Platte verlaufen, sondern mehrere Zentimeter nach links versetzt. Dies ist kein Problem, denn die linke Plattenseite ist freie Fläche.

Allerdings darf der Kabelbaum nicht zu weit links in Richtung Plattenende verlegt werden, da dort Quetschverbinder positioniert werden. Dieser Ort hat den Grund, da die Kabel des Umrichters sehr kurzgehalten sind und nur an dieser Stelle verlängert werden können. Folgendermaßen müssen sie für den Verbindungsanschluss aus dem 22,2 V Kabelbaum ausgelagert werden, um nicht gegen die vorhin aufgestellten Kabelbaumvorgaben zu verstoßen.

Aus diesem Grund wird der 22,2 V Kabelbaum auf Höhe des Umrichters enden. Auch, weil die Motorenkabel nun in Richtung Umrichter zu ihren Anschlüssen abgeführt werden und somit keine weitere Leitung in diesem Baum mehr enthalten ist.

Im Gegensatz dazu soll der Signalkabelbaum fast bis zum vorderen Ende der Platte in der waagerechten nach links weitergeführt werden. Denn so gut wie alle Leitungen führen zum Jetson.

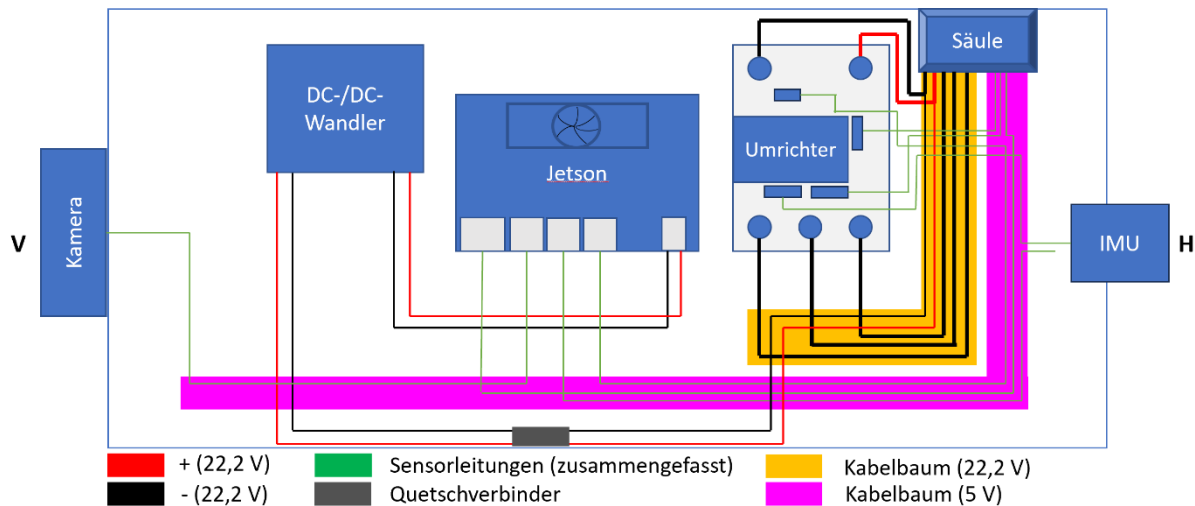


Abbildung 24: Skizze der oberen Platte (Unterseite) mit den vorläufigen Leitungs- und Kabelbaumverläufen

Quelle: Eigene Darstellung

Analog zur Grundplatte Abbildung 23 sind die genauen Verläufe von Leitungen und Kabelbäumen der Abbildung 24 zu entnehmen. Hierbei muss klargestellt werden, dass die Fläche aus Gründen der Übersichtlichkeit gespiegelt ist. Das bedeutet, alles abgebildete befindet sich in Wirklichkeit unterhalb der Platte. Für die eigentliche Vogelperspektive wäre die Säule etc. nach unten gespiegelt, anstatt wie hier dargestellt oben.

1.1.1.3 Obere Platte (Oberseite)

Die Oberseite der oberen Platte wird sehr wenige Leitungen führen, weshalb hier keine Kabelbäume realisierbar sind. So ist die Kamera an der Front der Platte fixiert, weshalb die Leitung hier nur auf kürzestem Weg bis zum vorderen Durchgangsloch der Platte verlaufen muss. Ähnlich sieht es am hinteren Ende der Platte aus. Hier sollen die Leitung der IMU und der Antenne zum hinteren Durchgangsloch der Platte führen.

2.5 Ermittlung der groben Leitungslängen

Verfasser des Abschnittes: Tom (100%)

Bearbeiter des Abschnitts: Tom (100%)

Jetzt geht es darum, die groben, benötigten Leitungslängen zu erfassen, da evtl. benötigte Leitungen aufgrund interner Richtlinien der Hochschule dringend bestellt werden müssen. Außerdem werden die Leitungslängen für die anschließende Querschnittsberechnung benötigt.

Hierzu wird die *Messen-Funktion* in Catia angewandt und die Leitungslängen zwischen zwei Komponenten, wie sie im vorherigen Abschnitt festgelegt wurden, ausgemessen. Die resultierenden Längen werden in der *Tabelle 2* festgehalten und mit der darin anliegenden Spannung, den fließenden

Dauer- bzw. Peakströmen, sowie den im anschließenden Kapitel errechneten Kabelquerschnitten gegenübergestellt.

2.6 Berechnung und Auswahl der Kabelquerschnitte

Verfasser des Abschnitttextes: Tom (100%)

Bearbeiter des Abschnitts: Tom (80%), Joelle (15%), Awat (2,5%), Shahrzad (2,5%)

Für die Berechnung der Kabelquerschnitte wird die Formel (1) herangezogen, indem zweimal der Dauerstrom I mit der Leitungslänge L multipliziert und anschließend durch die spezifische Leitfähigkeit des Leiters κ mal den Spannungsabfall U_D dividiert wird. Die Dauerströme und Leitungslängen sind jeweils der *Tabelle 2* zu entnehmen. Für die Leitfähigkeit wird 58 S/m eingesetzt, der Wert für Kupfer, da Kupferkabel (Erläuterung folgt in 2.7.12.7.1). eingesetzt werden sollen. Für den Spannungsabfall werden 0,1 V angesetzt. Dieser Wert resultiert aus Normtabellen für Automobile aus der Energiespeichervorlesung, jedoch leicht reduziert, da die Leitungslängen im Modellfahrzeug wesentlich kürzer ausgeführt sind als in einem PKW oder LKW.

$$A = \frac{2 \cdot I \cdot L}{\kappa \cdot U_D} \quad (1)$$

Die berechneten Querschnitte für jede Leiterstrecke finden sich in *Tabelle 2* (Spalte Querschnitt berechnet). Diese ist auch dem beigefügten Verzeichnis zu entnehmen (Dateiname: Kabeleigenschaften.pdf).

Tabelle 2: Parameter der einzelnen Leitungen

Verbindung:	Typ:	gemessene Länge (eine Leitung):	Spannung:	Dauerstrom:	Peakstrom:	Querschnitt berechnet:
Umrichter - Motor (Strang U)	1- polig	45 cm	22,2 V	100 A	197 A	15,52 mm ²
Umrichter - Motor (Strang V)	1- polig	47 cm	22,2 V	100 A	197 A	16,21 mm ²
Umrichter - Motor (Strang W)	1- polig	49 cm	22,2 V	100 A	197 A	16,90 mm ²
Umrichter (+) - Sicherung	1- polig	55 cm	22,2 V	100 A	197 A	18,97 mm ²
Umrichter (-) - Knotenpunkt (-)	1- polig	55 cm	22,2 V	100 A	197 A	18,97 mm ²
Batterie 1 (+) - Sicherung	1- polig	20 cm	22,2 V	50 A	98,5 A	3,45 mm ²
Batterie 1 (-) - Knotenpunkt (-)	1- polig	20 cm	22,2 V	50 A	98,5 A	3,45 mm ²
Batterie 2 (+) - Sicherung	1- polig	25 cm	22,2 V	50 A	98,5 A	4,31 mm ²
Batterie 2 (-) - Knotenpunkt (-)	1- polig	25 cm	22,2 V	50 A	98,5 A	4,31 mm ²
Sicherung - DC/DC- Wandler (+)	1- polig	85 cm	22,2 V	1,5 A	2 A	0,44 mm ²
Knotenpunkt (-) - DC/DC-Wandler (-)	1- polig	85 cm	22,2 V	1,5 A	2 A	0,44 mm ²
Umrichter - Servo	3- polig	75 cm	5 V	0,5 A	-	0,13 mm ²
Umrichter - Positionsgeber	5- polig	55 cm	5 V	0,5 A	-	0,28 mm ²

Umrichter - Lüfter:	2-polig	40 cm	5 V	0,5 A	-	0,07 mm ²
Jetson - IMU	4-polig	59 cm	5 V	0,5 A	-	0,1 mm ²
Jetson - DC/DC-Wandler	2-polig	21 cm	12 V	1,5 A	-	0,11 mm ²
Jetson - DC/DC-Wandler (+)	1-polig	-	12 V	1,5 A	-	0,11 mm ²
Jetson - DC/DC-Wandler (-)	1-polig	-	12 V	1,5 A	-	0,11 mm ²
Jetson - Kamera	USB	21 cm	5 V	0,7 A	-	-
Jetson - Antenne	USB	38 cm	5 V	0,7 A	-	-
Jetson - Umrichter	USB	31 cm	5 V	0,7 A	-	-

Die berechneten Werte werden nun auf den nächstgrößeren, verfügbaren Querschnitt gerundet. Denn nicht für jeden Querschnitt in mm²-Schritten werden Leitungen angeboten. Lediglich bei sehr kleinen Abweichungen wird abgerundet. Hierbei wird sich an die Automobilnorm ISO 6722-3 gehalten, welche die verfügbaren Querschnitte für elektrische Kupferleitungen vorgibt, da bei diesem Modellfahrzeug möglichst Automobilstandards anzuwenden sind.

Anschließend gilt es, die berechneten Querschnitte auf Plausibilität zu überprüfen, indem die Strombelastbarkeit bei maximalem Dauer- und Peakstrom, also die Erwärmung der Leitungen bei diesen Strömen, kontrolliert wird. Auch die bereits verbauten Leitungsquerschnitte (z.B. Motorkabel – fest mit dem Antrieb verlötet) sollen zur Beurteilung miteinbezogen werden. Hierzu wird das Diagramm der Tabelle 2 angewandt, welches angibt, wie stark sich eine Leitung bei bestimmtem Querschnitt bei bestimmtem Strom erwärmt. Dass bedeutet, für jede Leitung wird der gerundete Querschnitt herangezogen, die dazugehörige Kurve ausgesucht und danach geprüft, wie stark sich die Leitung bei maximalen Dauerstrom bzw. Peakstrom erhitzt. Ganz wichtig ist hier, zusätzlich die Raumtemperatur miteinzubeziehen, also rund 25 °C. Generell sollte die Erwärmung eigentlich nicht mehr als 100 °C betragen, da die maximale Kabeltemperatur meist bei 125 °C liegt.

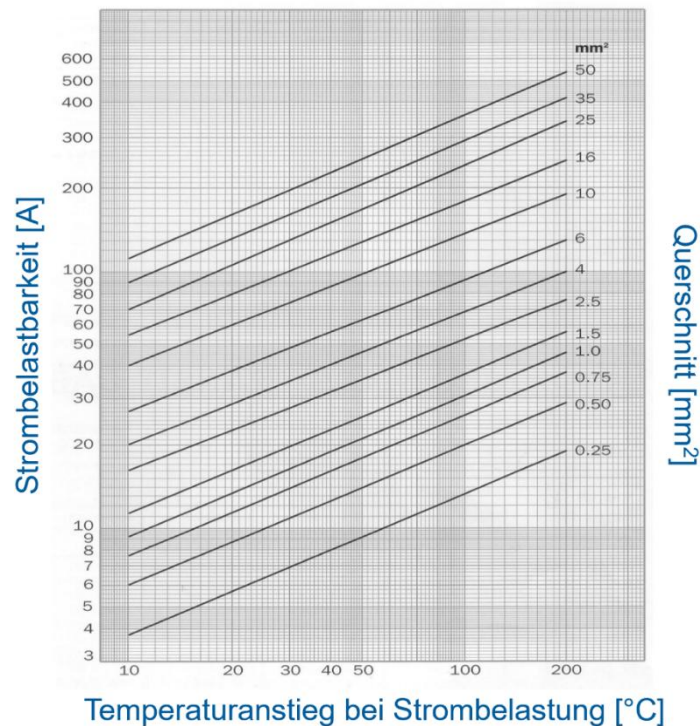


Abbildung 25: Diagramm mit der thermischen Belastbarkeit von elektrischen Leitungen

Quelle: [17]

Für die Verbindung Batterie zu Umrichter wurde ein Querschnitt von 25 mm² berechnet. Schaut man sich die Erwärmung bei Dauerstrom (100 A) an, steigt die Temperatur der Leitung um rund 17 °C, bei Peakstrom um ca. 45 °C. Dies ist ziemlich wenig. Aufgrund der hohen Masse, Abmaße und Kosten solcher Leitungen, gerade durch die erforderlichen Längen, wird der nächstkleinere Querschnitt herangezogen. Hier liegt die Erwärmung im Dauerbetrieb bei 30 °C, im Überlastbereich bei 115 °C, was addiert mit der Raumtemperatur etwas über der Grenze liegt. Nach genauer Überlegung und Abklärung mit dem Projektbetreuer soll dennoch die 16 mm² Leitung verbaut werden. Zum einen aufgrund der Kabellänge von knapp 50 cm (Erwärmung benötigt länger – größerer Wärmekörper), der Abmaße und der besseren Flexibilität, da das Fahrzeug einen geringen Platzbedarf aufweist und bereits dieses Kabel darin schwierig zu verlegen ist, zum anderen, weil ein längerfristiger Betrieb mit knapp 200 A eher unwahrscheinlich ist, gerade wegen der eingebauten Getriebeübersetzung.

Bei den Motorleitungen konnte ein Querschnitt von gerundet 16 mm² taxiert werden, verbaut sind aktuell 8,37 mm² (8AWG). Dieser Querschnitt ist generell auch nötig, siehe vorheriger Absatz für den Umrichter (gleiche Stromstärken). Allerdings muss man betonen, dass es sich bei diesem Antrieb um einen Drehstrommotor mit Sternschaltung handelt. Folglich teilt sich der Strom, welcher aus der Batterie bezogen wird, gleichmäßig auf die drei Phasen auf, wenn auch zeitlich verschoben und mit wechselnden Vorzeichen, also 33,3 A im Dauerbetrieb, ca. 65 A bei Überlast. Dementsprechend wären 16 mm² völlig überdimensioniert (kann dem Diagramm aufgrund der niedrigen Erwärmung nicht entnommen werden). Einen akzeptablen Temperaturanstieg bei 65 A bietet die 6 mm² Leitung (55 °C). Jedoch werden die bereits vorhandenen Motorleitungen (kaum austauschbar) weiterverwendet, um eine einheitliche Verkabelung sicherzustellen.

Im Folgenden müssen die Batterieleitungen zum Sicherungs- bzw. Masseknotenpunkt ausgewählt werden. Nach der Berechnung können 4 bzw. 6 mm² genannt werden (je nach Strecke). Überprüft man die Dauer- und Maximalströme (50 A und 98,5 A), so erhält man eine Kabelerwärmung von 33 °C bzw. 120 °C für die 6 mm² Leitung. Die 4 mm² Möglichkeit wird erst gar nicht auf Plausibilität geprüft, da die

Erwärmung im Überlastbetrieb aufgrund des Ergebnisses der 6 mm² Leitung viel zu hoch wäre. Folgendermaßen sind die Verbindungsleitungen zwischen den beiden Batterien und dem Masseknotenpunkt bzw. der Sicherung jeweils als 6 mm² Leitung auszuführen. Da die Batteriekabel fest mit dem Batteriepack verbunden sind, gilt dies nur für die Verlängerungen, wobei die batterieeigenen Leitungen denselben Querschnitt besitzen.

Anschließend wird der Querschnitt der beiden Versorgungsleitungen des DC-/DC-Wandlers, ausgehend von der Sicherung bzw. des Masseknotenpunktes, bestimmt. Hier wurden rechnerisch 0,5 mm² ermittelt. Belastet werden beide Kabel mit 1,5 (Dauerbetrieb) bzw. 2 A (Peak). Da die Werte niedrig sind, führen sie zu keiner nennenswerten Erwärmung der Leitung und sind folglich auch nicht in Abbildung 25 eingezeichnet. Schaut man sich das Diagramm an, würde theoretisch sogar die 0,25 mm² Leitung ausreichen. Aufgrund von möglichem zusätzlichem Verbraucher, die zukünftig über den DC-/DC-Wandler betrieben werden könnten, der Kabelschuhe, welche für den Anschluss an die Sicherung bzw. den Masseknotenpunkt notwendig sind (kleinere Leitungen nicht fest vercrimpbar) und der nicht möglichen Steckverbindung mit den vom Wandler ausgehenden Leitungen (ca. 3,31 mm²), werden 2,5 mm² Leitungen angewandt.

Dasselbe Bild zeichnet sich bei der Verbindung zwischen Jetson und dem Wandler ab, welches die einzige an den Wandler angeschlossene Komponente ist. Dementsprechend fließt dort derselbe Strom. Der berechnete Querschnitt liegt hier sogar bei nur 0,25 mm². Folglich gibt es auch dort keinen nennenswerten Temperaturanstieg, womit dieser Querschnitt ausreichen würde. Aufgrund des vom DC-/DC-Wandler ausgehenden Kabel, mit einem Querschnitt von 2,08 mm² (14AWG), welches minimal verlängert werden muss, gibt es allerdings das Problem, dass Stecker oder Steckerverbinder entweder nur für das etwas dickere oder das sehr dünne Kabel verfügbar sind. Ebenso kann am DC-Stecker für den Jetson keine 0,25 mm² Leitung befestigt werden. Deshalb wird der Querschnitt für die benötigte Leitung auf 0,75 mm² angehoben, wodurch beide Probleme behoben werden können. Demnach wären nun auch deutlich höhere Ströme über diese Leitung möglich.

Als nächstes wird der Leitungsquerschnitt für die Strecke Servo zu Umrichter ausgewählt. Durch die rechnerische Bestimmung wurden 0,25 mm² ermittelt. Selbiges gilt für die Leitung vom Umrichter zum Lüfter, sowie zum Lüfter und zum Positionsgeber. Da hier die fließenden Ströme jeweils maximal 0,5 A betragen, ist kein Temperaturanstieg aus Abbildung 25 entnehmbar, weshalb sich auf die Berechnungen verlassen wird und 0,25 mm² Leitungen Anwendung finden sollen. Die ausgehenden Leitungen an den Komponenten bestätigen die Wahl, da sie dieselben Querschnitte aufweisen.

Für die Leitungen von der IMU zum Jetson soll ebenfalls der berechnete Querschnitt von 0,25 mm² umgesetzt werden. Der Grund liegt im kaum vorhandenen Temperaturanstieg wie bei den im vorherigen Absatz behandelten Leitungen.

Schließlich bleiben nur noch die USB-Leitungen übrig. Bei diesen ist kein Querschnitt zu bestimmen.
[15] [17] [18] [19]

2.7 Auswahl geeigneter Leitungen

Verfasser des Abschnitttextes: Tom (95%), Akram (5%)

Bearbeiter des Abschnitts: Tom (75%), Akram (25%)

Nachdem die Kabelquerschnitte bestimmt sind, gilt es nun, die passenden Leitungen für das Modellfahrzeug auszuwählen. Hierbei ist es wichtig, das passende Leiter- und Isolationsmaterial zu wählen, sowie eine farbliche Codierung der Isolation herzustellen. Grundsätzlich sind zudem Automobileitungen zu verwenden, da diese eine höhere Robustheit und einen großflächigen

Temperaturbereich besitzen. Außerdem ist es wichtig, bereits bestehende Leitungen möglichst weiterzuführen, d.h. zu verlängern.

2.7.1 Bestimmung des Leitermaterials

Als Leitermaterial wird Kupfer herangezogen, weil es die zweithöchste Leitfähigkeit nach Silber besitzt (58 S/m anstatt 62 S/m), aber deutlich billiger ist, was bei einem Studentenprojekt keine untergeordnete Rolle spielt. Noch günstiger wäre Aluminium. Darauf wird aber verzichtet, da es einerseits eine niedrigere Leitfähigkeit als Kupfer besitzt (37,7 S/m), andererseits eine geringere Flexibilität und Zugfestigkeit aufweist, was bei diesem Modellfahrzeug nicht unwesentlich ist. Denn aufgrund des teils geringen Platzbedarfs werden die Leitungen teils mit größeren Biegungen verlegt. Hinzu kommt die einfachere Isolation von Kupferleitungen. [15] [19] [20] [21]

2.7.2 Bestimmung des Isolationsmaterials

Für das Material der Isolation soll grundsätzlich PVC genutzt werden. Es ist der Standard bei Fahrzeugleitungen, weshalb Kabel vom Typ FLY oder FLRY eingesetzt werden. Dies sind Standardleitungen in der Automobilindustrie (Fahrzeugleitungen) und weisen eine hohe Robustheit, als auch Temperaturbeständigkeit auf (-40 – +105 °C).

Generell sollen alle Leitungen mit geringer Strombelastung, d.h. die Sensor- und Versorgungsleitungen (von und zum DC-/DC-Wandler) als FLRY ausgeführt werden, d.h. mit reduzierter Isolierung, aufgrund der geringen Strombelastung. Dadurch steigt die Flexibilität, das Gewicht und der Platzbedarf sinkt.

Die Kabel mit sehr hoher Strombelastung (Batterie zu Umrichter) werden als FLY ausgeführt, also mit stärkerer Isolation als bei FLRY, wodurch auch die Erwärmung im Überlastbetrieb weniger stark ausfällt, sowie ein besserer Isolationsschutz gegeben ist.

Eine Ausnahme stellen die Motorleitungen dar. Da die bereits bestehenden Leitungen verlängert werden sollen, ändert sich auch das Isolationsmaterial, nämlich zu Silikon. Dies hat jedoch die entscheidenden Vorteile, dass es deutlich flexibler ist, somit besser zu verlegen, als auch eine höhere Temperaturbeständigkeit aufweist (-60 – +200 °C).

Solch eine Isolation hätte sich auch für die 16 mm² Leitung angeboten. Jedoch war diese vom Typ FLY bereits vorhanden und hätte gleichzeitig hohe Kosten verursacht. [15] [21] [22]

2.7.3 Festlegung der Farbcodierung der Isolation

Generell sollen die Farben der Isolation möglichst standardisiert sein, d.h. an die bekannte Farbgebung, sodass auch Außenstehende die wichtigsten Leitungen schnell erkennen. Folgendermaßen werden Masseleitungen immer in schwarz ausgeführt, Plus-Leitungen in Rot. Da die Output-Plus-Leitung des Umrichters in Gelb gehalten ist, soll somit auch die Verlängerung dieser Leitung aus Gründen der Übersichtlichkeit in Gelb ausgeführt werden.

Die Signalleitungen der Sensoren hingegen werden in ihrer bestehenden Farbe, zwecks der Übersichtlichkeit, weitergeführt. Bei eigens zu verkabelnden Sensoren (z.B. die IMU oder der Positionsgeber) werden die Farben der Isolation, welche bei anderen Sensoren für die Signalleitungen angewandt wurden, ebenfalls genutzt. Diese sind weiß, orange, gelb, grün und blau.

2.8 Auswahl der Steckverbindungen

Verfasser des Abschnitttextes: Tom (90%), Akram (10%)

Bearbeiter des Abschnitts: Tom (70%), Akram (30%)

Zur Auswahl der Steckverbindungen müssen zuerst Anforderungen erarbeitet werden, um im Anschluss einen passenden Stecker aussuchen zu können. So ist es zuerst einmal wichtig, dass der Stecker die erforderlichen Spannungen (bis 22,2 V) und Ströme (bis ca. 200 A) unterstützt. Des Weiteren muss die Steckverbindung aufgrund der Außenverwendung des Modellfahrzeugs gegen allerweil Umwelteinflüsse resistent sein. Dass bedeutet, Schutz gegen Eindringen von Wasser, Feuchtigkeit, Fremdkörpern, anderen Flüssigkeiten, wie auch einem Temperaturbereich von (-20 bis 120 °C). Hinzukommen Aspekte wie hohe Robustheit, geringer Platzbedarf, Verpolenschutz, eine Verriegelung vor selbstständigem Öffnen der Steckverbindung, sowie die Unterstützung von häufigem An- und Abstecken. Nicht zu vergessen sind geringe Übergangswiderstände der stromführenden Leiter und eine hohe elektrische Isolationsfestigkeit zwischen den Leitern untereinander und zwischen Leiter und Umgebung. Generell sollten zudem so wenig unterschiedliche Steckertypen wie möglich verbaut sein.

In erster Linie bietet sich der AMP-Stecker an Abbildung 26. Er ist in der Automobilindustrie etabliert und erfüllt die eben genannten Kriterien. Einziger Nachteil ist die Strombelastbarkeit von bis zu 14 A, womit er nicht für die Verbindung der Batterien, des Umrichters und des Motors geeignet ist. Dies ist allerdings nicht schlimm, den aufgrund des großen Querschnitts der Leitungen könnte man diesen Stecker sowieso nicht daran anbringen (max. bis 1,5 mm² möglich). Deshalb soll nur für die Anwendungen mit Spannungen von 5 V, also die Sensorleitungen, der AMP-Stecker genutzt werden. Je nach Anzahl der Leitungen drei- oder fünfpolig.



Abbildung 26: AMP-Stecker und -Buchse

Quelle: [23]

Für die Batterien, Motor und Umrichter ist jedoch ein anderer Stecker erforderlich. Hier wird der EC8-Stecker Abbildung 27 angewandt. Auch dieser unterstützt die oben genannten Anforderungen, hat aber eine Strombelastbarkeit von bis zu 180 A und die Möglichkeit, 16 mm² Leitungen aufzunehmen. [15] [23] [24]



Abbildung 27: EC8-Stecker
Quelle: [24]

2.9 Bestimmung der endgültigen Leitungsverläufe und Steckerpositionen

Verfasser des Abschnitttextes: Tom (90%), Akram (10%)

Bearbeiter des Abschnitts: Tom (70%), Akram (30%)

Da in 0 kaum Steckerpositionen festgelegt wurden, auch weil diese zum damaligen Zeitpunkt noch nicht ausgewählt waren, sind nun einige Anpassungen an der Leitungs- bzw. Kabelbaumpositionierung von Nöten. Dies ist auch durch anfangs nicht erdachte, teils sehr große Leiterquerschnitte bedingt. Zur Unterstützung wurde hierzu ein 1:1 Ausdruck der 2D-Zeichnung des Fahrzeugs durch die Projektbetreuer ausgehändigt, auf welchem die einzelnen Leitungen eingezeichnet und anschließend direkt für den Einbau abgemessen werden können.

Schaut man sich nochmals die Abbildung 23 und Abbildung 24 an, so erkennt man, dass bisher nur Quetschverbinder eingezeichnet und positioniert wurden. Diese Position und der Kabelverlauf der dahin führenden Leitungen sollen so beibehalten werden. Allerdings sind nun noch einige, hinzugekommene Stecker anzuordnen.

2.9.1 Grundplatte

So gibt es an jeder Batterie einen EC8-Stecker, um diese u.a. für das Laden vom Fahrzeug zu trennen. Dieser Stecker ist rund 5 cm lang. Da die bereits eingepplanten Leitungsverläufe von den Batterien zur Sicherung bzw. zum Masseknotenpunkt aus Platzgründen nicht anders angeordnet werden können und die Länge der ausgehenden Batterieleitungen nicht veränderlich sind, soll der Stecker längs jeweils zwischen dem Akku und Plattenende angeordnet werden.

Weiter ist die ausgehende Leitung des Servos zu kurz. Infolgedessen muss diese verlängert werden, wodurch möglichst in der Nähe des Servos ein Stecker erforderlich wird. Leider verlässt die Leitung des Servos diesen auf der rechten Seite in unmittelbarer Nähe zu den Kabelbäumen, weshalb dort kein Stecker platziert werden darf. Schaut man sich hierzu nochmals das CAD-Modell an, erkennt man, dass der Servo mit Haltern etwas über der Grundplatte positioniert ist, sichtbar in der folgenden Abbildung 28. Darunter ist jedoch ein Freiraum.

Somit soll die Leitung nun in einem Bogen, unterhalb des Servos durchgeleitet und von vorne in den AMP-Stecker eingeführt werden, der neben dem Servo platziert wird. Von diesem Stecker aus wird die Leitung dann nach altem Verlauf in Richtung Signalkabelbaum verlaufen.

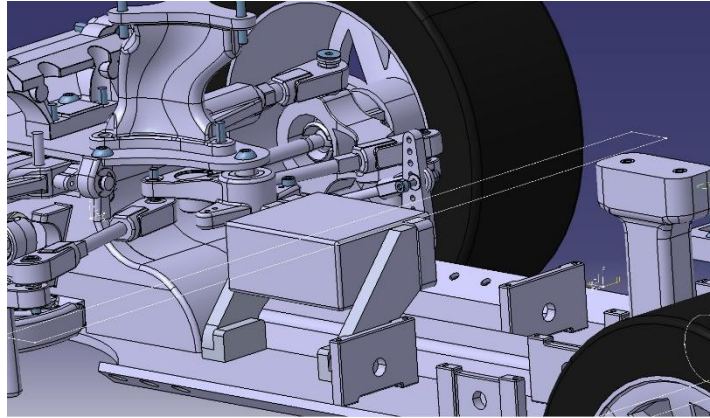


Abbildung 28: Ausschnitt des CAD-Modells im Bereich des Servos

Quelle: Eigene Darstellung

Weiter besteht das Problem, dass die beiden Kabelbäume nicht zwischen Batterie und Motor angeordnet werden können. Einerseits, weil der Platz durch die enormen Kabelquerschnitte nicht ausreichen würde, auf der anderen Seite sind bei diesen Leitungen aufgrund ihrer schlechten Flexibilität die eingezeichneten Abbiegungen nicht umsetzbar. Allerdings konnte über das CAD-Modell ein ausreichend großer Freiraum unter dem Motor identifiziert werden (dargestellt in Abbildung 29). Somit werden nun beide Kabelbäume, mit gegenseitigem Abstand (wegen der EMV-Thematik), unter dem Motor in Richtung Säule hindurchgeführt.

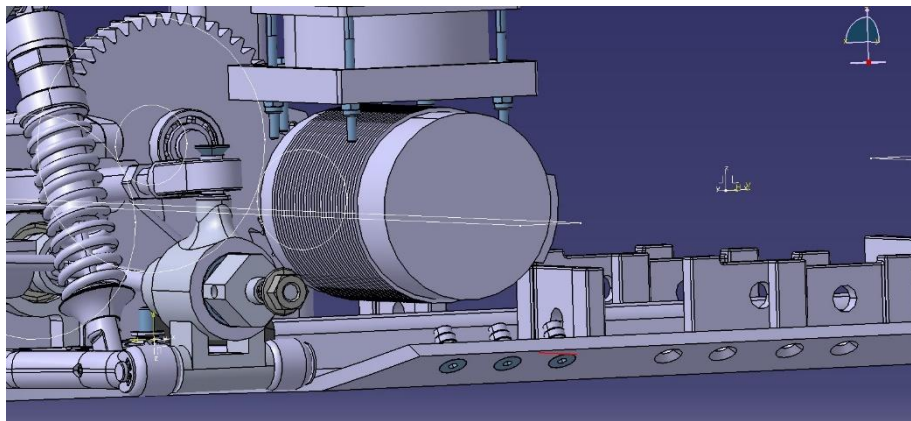


Abbildung 29: Ausschnitt des CAD-Modells im Bereich des Motors

Quelle: Eigene Darstellung

Auch die Leitung des Lüfters muss verlängert werden. Da dieser auf dem Motor sitzt, ist der Platz für einen Stecker gering. Deshalb wird dieser freihängend neben dem Antrieb angeordnet, sodass die Leitung, welche zum Umrichter führt, direkt darunter in den Signalkabelbaum integriert werden kann.

2.9.2 obere Platte (Unterseite)

Auch auf der oberen Platte sind mehrere Stecker zu platzieren. Denn Lüfter, Servo und der Positionsgeber müssen mit speziellen Steckern am Umrichter angeschlossen werden. Diese Stecker sind jedoch immer an sehr kurze Leitungen gebunden, weshalb sie nicht direkt mit der vom Sensor kommenden Leitung verbunden werden können. Infolgedessen sind nochmals drei AMP-Stecker erforderlich von Nöten, die aber aufgrund der eben genannten kurzen Leitungen in unmittelbarer Nähe zum Umrichter zu platzieren sind. Aus diesem Gründen müssen auch die Kabelbaumverläufe neu positioniert werden.

Diese sollen zukünftig nicht mehr senkrecht von der Säule zur Plattenmitte führen, sondern einen bogenhaften Verlauf, in unmittelbarer Nähe zum hinteren Durchgangsloch, nehmen. Grund dafür ist, dass zwei der Stecker direkt neben dem Umrichter angeordnet werden müssen. Der dritte wird etwas nach rechts versetzt, da dazwischen die Kabelbäume verlaufen sollen. Diese hätten sonst keinen Platz mehr, wegen dem eben beschriebenen Loch.

Die genannten Positionen und Verläufe sind der folgenden Abbildung 30 zu entnehmen, welche die 2D-Zeichnung des Fahrzeugs enthält. Darin sind die Kabelbaum- und Steckerpositionen zur besseren Veranschaulichung farblich markiert.

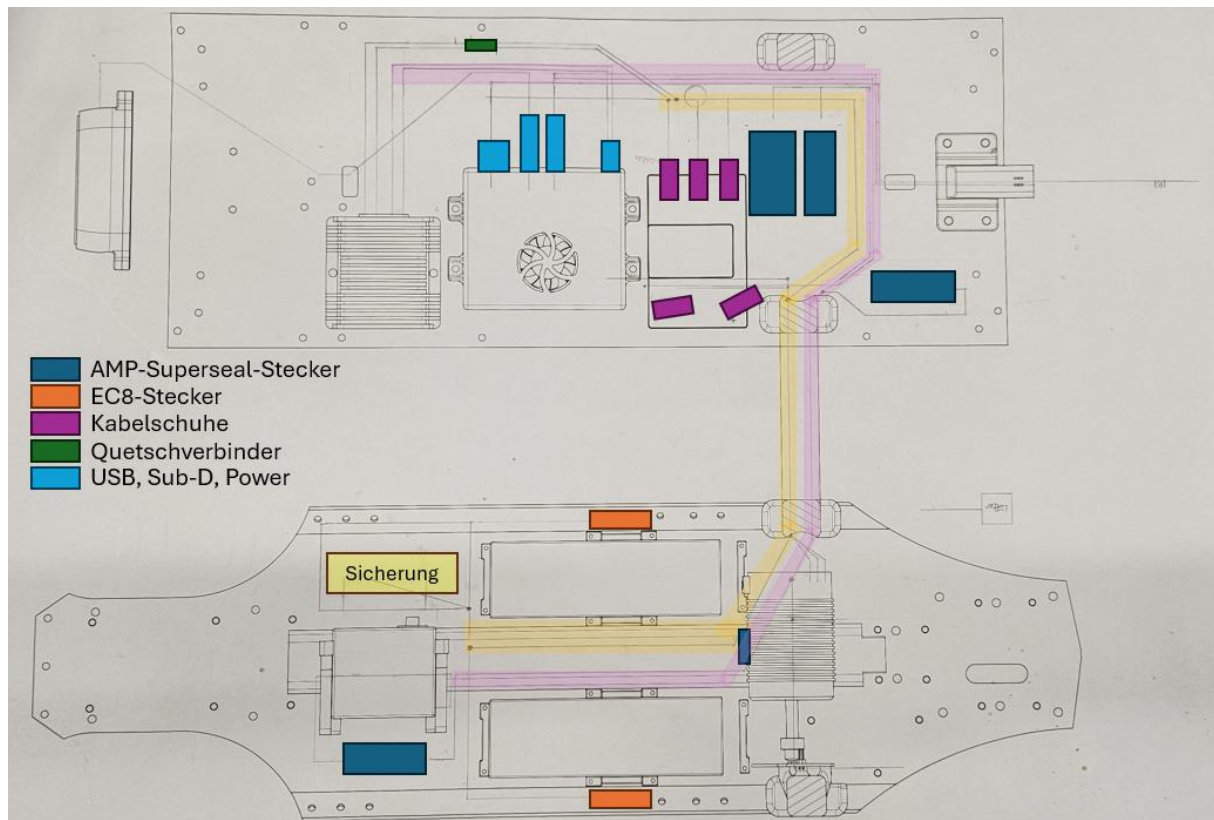


Abbildung 30: 2D-Skizze des Fahrzeugs mit farblich eingezeichneten Steckern und Leitungen
Quelle: Eigene Darstellung

2.10 Durchführung der Verkabelung

Verfasser des Abschnitttextes: Akram (90%), Tom (10%)

Bearbeiter des Abschnitts: Akram (37,5%), Tom (37,5%), Joelle (10%), Awat (15%)

1. Zielsetzung: Unser Ziel ist es, eine präzise und sichere Verkabelung für unser teleoperiertes Fahrzeug zu realisieren, um sowohl die Funktionalität als auch die Sicherheit des Fahrzeugs zu gewährleisten.

2. Abmessen mit Plan: Die erste Phase unseres systematischen Ansatzes besteht darin, die erforderlichen Kabellängen anhand des detaillierten Fahrzeugplans abzumessen. Die Präzision dieses Schrittes ist entscheidend, um Überschuss zu vermeiden und sicherzustellen, dass die Kabel genau passen und effizient angeordnet werden können.

3. Stecker anschließen: Nachdem die Kabellängen abgemessen wurden, schreiten wir zur Verbindung der Stecker fort. Hierbei beachten wir sorgfältig die elektrischen Standards, um sicherzustellen, dass jede Verbindung korrekt und fest ist, was potenzielle Risiken minimiert.

4. Funktionstest: Bevor die Kabel endgültig positioniert werden, führen wir einen gründlichen Funktionstest durch. Dieser Schritt ist entscheidend, um mögliche Probleme frühzeitig zu erkennen und zu beheben, und gewährleistet, dass alle Systeme einwandfrei funktionieren.

5. Positionierung: Nach erfolgreichen Tests positionieren wir die Kabel im Fahrzeug. Dabei legen wir besonderen Wert darauf, dass sie gut organisiert sind und die Bewegungsfreiheit der mechanischen Teile nicht einschränken. Eine ordentliche Kabelverlegung ist essenziell für die Langlebigkeit und Zuverlässigkeit unseres Fahrzeugs.

6. Klebeflächen: Abschließend sichern wir die Kabel mit Klebeflächen und Kabelbindern im Fahrzeug. Dies verhindert, dass sich Kabel während der Fahrt lösen oder beschädigt werden, und trägt entscheidend zur allgemeinen Sicherheit des Fahrzeugs bei.

2.11 Mögliche Verbesserungen an der Verkabelung

Verfasser des Abschnitttextes: Akram (80%), Tom (20%)

Bearbeiter des Abschnitts: Akram (80%), Tom (20%)

Platzbedarf-Optimierung durch große Querschnitte

Aufgrund von großen Kabelquerschnitten und nicht eingeplanten Trennsteckern, insbesondere bei der Batterie und dem Motor, kam es gerade im Bereich der Säule zu leichten Platzproblemen. Durch besseren Einbezug in die theoretische Planung und bessere Absprache hätte dies besser gelöst werden können.

Optimierung der allgemeinen Kabelführung

Bezüglich der allgemeinen Kabelführung stellen wir fest, dass die aktuelle Anordnung Verbesserungspotenzial bietet. Unordentliche Kabelwege können Wartungsarbeiten erschweren und Fehlerdiagnosen verlangsamen. Wir planen daher, ein systematisches Kabelführungskonzept zu implementieren, das Kabel klare Beschriftungen umfasst. Dies erleichtert die Identifizierung und Wartung der Kabel erheblich.

Bessere Verknüpfung zwischen theoretischer und praktischer Planung

Wir haben bemerkt, dass es eine Lücke zwischen der theoretisch geplanten Kabelverlegung und der praktischen Umsetzung gibt, besonders bei Kabelabzweigungen und -kurven. Um dies zu verbessern, denken wir, dass die verstärkte Nutzung auf früheren relevanten Ergebnissen von 3D-Modellierung und Simulationstools ermöglichen es uns, Kabelwege präziser zu planen und potenzielle Probleme im Voraus zu identifizieren.

3 Software-Dokumentation (Albin Hrabos)

Folgende Kapitel sind ein Leitfaden für die Installation von der Software für das teleoperierte Fahrzeug und Beschreibung der Versuche im Laufe der Softwareintegration. Am Ende werden die Software-Pakete den für das Projekt verantwortlichen Labormitarbeiter zur Verfügung gestellt. Sie sind auch unter folgenden Links zu finden:

Erster Teil auf dem Jetson-Zweig:

https://github.com/Arash-Barabadi/Teleoperated_Auto/tree/Teleop_Jetson

Zweiter Teil auf dem Remote-Treiber-Zweig:

https://github.com/Arash-Barabadi/Teleoperated_Auto/tree/Teleop_Remote

3.1 Ubuntu 20.04 (Arash Barabadi)

Zuerst wird ein Rechner mit **Ubuntu 20.04.x** benötigt. Dieses Betriebssystem wird gewählt, weil es für gute Kompatibilität mit **ROS2** sorgt.

3.2 ROS2 (Arash Barabadi)

Für unsere Anwendung muss sowohl auf dem Rechner als auch auf dem Jetson **ROS2 Roxy** [25] installiert werden. In diesem Betriebssystem werden die Nodes erstellt die für Informationsaustausch zwischen der **Fahrzeug- und Fahrerseite** sorgen.

ROS2 wird auf den Jetson genauso wie auf den Laptop anhand dieser Anleitung installiert:

[Anleitung ROS2](#) [26]

Diese Installation auf Jetson erfolgt durch GVNC-Viewer durch den Rechner und die Schritte entsprechen der Installation von ROS2 wie für einen normalen Rechner. Mehr dazu kann man in dem Kapitel **Kommunikation und Befehlsgebung** erfahren.

3.3 Fahrzeug- und Fahrerseite (Albin Hrabos)

Die Software der Teleoperierte Fahrt basiert auf zwei Seiten, die miteinander kommunizieren. Die Fahrer- und Fahrzeugseite. Die Fahrzeug- und Fahrerseiten sind auch in dem Strukturbild (Abbildung 2) zu sehen.

Die zentrale Schnittstelle von Fahrerseite ist ein Rechner mit dem Betriebssystem Ubuntu 20.04 und installiertem ROS2 Foxy. In dem ROS2 Umfeld läuft auch die ganze Kommunikation der Fahrzeugkomponente mit dem Jetson, der die zentrale Schnittstelle der Fahrzeugseite ist. Auf dem Jetson laufen auch Nodes über die die Fahrzeugdaten auf die Fahrerseite übertragen werden können. Diese Datenübertragung passiert in einem Mesh Netzwerk. Aus der Fahrerseite werden Befehle wie z.B. Starten der Teleoperation Nodes durchgeführt oder die Lenk- und Beschleunigungsbefehle aus dem Lenkrad und Pedal. Außerdem wird das Bild von Fahrzeugkamera auf dem Laptop über ein Kamera Node gezeigt.

3.4 ROS2-Begriffe (Arash Barabadi)

3.4.1 Workspace

Ein Workspace in ROS2 ist ein Verzeichnis, das alle Pakete und deren Abhängigkeiten für die Entwicklung von Robotersoftware enthält. In einem Workspace können mehrere ROS2-Pakete gleichzeitig bearbeitet und kompiliert werden.

3.4.2 Package (Paket)

Ein Package in ROS2 ist eine grundlegende Organisationseinheit für Roboter- oder Autosoftware, die verschiedene Dateien und Ressourcen enthält, um eine bestimmte Funktion oder Aufgabe auszuführen. Es kann sich um eine Sammlung von Bibliotheken, ausführbaren Dateien, Konfigurationsdateien, Skripten, Testdateien und anderen Ressourcen handeln, die für die Entwicklung und Ausführung von Robotik Anwendungen erforderlich sind.

3.4.3 Erstellung von ROS2-Workspace

Ein Workspace wird eingerichtet und dort wird ein Paket bereitgestellt.

```
mkdir -p ~/projekt1_ws/src  
cd projekt1_ws  
colcon build
```

Wenn der Workspace erstellt wird, enthält er vier verschiedene Ordner,

1-build

2-install: Sie enthält viele Dateien, aber im Moment sind zwei davon für uns wichtig, nämlich "setup.bash" & "local_setup.bash".

"local_setup.bash" wird einfach nur den Workspace projekt1_ws (Overlay-Workspace) sourcen. Wenn ich das Skript "local_setup.bash" verwende, kann ich alles verwenden, was ich im Arbeitsbereich von projekt1_ws erstellt habe.

"setup.bash" wird das projekt1_ws (overlay) und die globale ROS2-Installation (underlay workspace) verwenden, daher sollte der Einfachheit halber der folgende Befehl in die .bashrc-Datei geschrieben werden, um die "setup.bash" zu verwenden.

```
source ~/projekt1_ws/install/setup.bash
```

3-log

4-src (das mit dem Befehl "mkdir -p ~/projekt1_ws/src" erstellt wurde)

3.4.4 Erstellung von ROS2-Paket(Package)

Um einen ROS2-Knoten zu erstellen, wird ein Paket benötigt. Pakete ermöglichen es dem Benutzer, den Code in wiederverwendbare Blöcke zu implementieren. Jedes Paket ist eine unabhängige Einheit. Zum Beispiel können wir ein Paket haben, um die Kamera zu handhaben, ein anderes Paket ist für das Lenkrad unseres Autos. Es sollte beachtet werden, dass alle Pakete im src-Ordner erzeugt werden müssen.

1. Navigiert wird zum src-verzeichnis der ROS-Pakete.

```
cd ~/projekt1_ws/src
```

2. Ein Paket wird einfach durch Eingabe erstellt: `ros2 pkg create "Paketname" "Pakettyp" "Abhängigkeiten"`

"Paketname": Es wird einfach ein Paketname ausgewählt.

"Pakettyp": Es wird ein Argument angegeben, um anzugeben, welcher Art das Paket sein soll. In ROS2 gibt es einen Unterschied zwischen einem Python-Paket und einem C++-Paket.

"Abhängigkeiten": Abhängigkeiten sind einfach die Pakete, von denen dieses neue Paket abhängt.

```
ros2 pkg create imu_neu --build-type ament_python --dependencies rclpy
```

#imu_neu: Name von Paket des IMU-Sensors

3.4.4.1 Übersicht über den Inhalt des generierten src-Ordners

1-imu_neu Ordner: Dieser Ordner hat immer den gleichen Namen wie das Paket. Alle Python-Knoten werden in diesem Ordner abgelegt. Er enthält bereits die Datei "init.py". Sie braucht im Moment nicht geändert zu werden.

2-resource folder

3-test folder

4-package.xml: Jeder Paket-Ordner hat eine package.xml-Datei, die im Wesentlichen zwei Unterabschnitte enthält. Die erste enthält Informationen über das Paket wie Name, Version, Beschreibung, E-Mail und Lizenz, die je nach dem, was der Entwickler veröffentlichen möchte, angepasst werden können. Der zweite Abschnitt bezieht sich auf die Abhängigkeiten (Bibliotheken), die von dem Paket wie rclpy verwendet werden. Wenn der Benutzer neue Abhängigkeiten hinzufügen möchte, sollte dies ebenfalls hier eingetragen werden. Am Ende der Datei ist der Typ des Pakets zu sehen, der hier ament_python ist.

5-setup.cfg

6-setup.py

Danach wird zur Workspace-Adresse zurückgegangen und erneut "colcon build" eingegeben, um die Erstellung des neuen Pakets "imu_neu" zu bestätigen.

```
colcon build
```

```
Starting >>> imu_neu
```

```
Finished <<< imu_neu [1.23s]
```

Summary: 1 package finished [3.09s]

Oder wenn es viele Pakete gibt, kann der folgende Befehl speziell für ein Paket geschrieben werden, wodurch die Kompilierzeit kürzer sein sollte.

```
colcon build --packages-select imu_neu
```

Starting >>> imu_neu

Finished <<< imu_neu [1.26s]

Summary: 1 package finished [3.13s]

Jetzt ist das Python-Paket bereit, jeden Python-Knoten zu hosten.

Der Prozess der Erstellung eines Musterknoten wird im IMU-Sensor-Teil erklärt.

3.4.5 Node

Ein Node in ROS 2 ist eine grundlegende Softwareeinheit, die Berechnungen durchführt und mit anderen Nodes kommuniziert, indem sie Nachrichten über das ROS2-Kommunikationsnetzwerk austauscht. Jeder Node kann Nachrichten veröffentlichen, um sie anderen Nodes mitzuteilen, oder Nachrichten abonnieren, um Informationen von anderen Nodes zu empfangen. Dies ermöglicht es Nodes, Informationen aus der Umgebung zu erfassen, zu verarbeiten und darauf zu reagieren.

Ein Node wird typischerweise in einer Programmiersprache wie C++ oder Python geschrieben und verwendet ROS 2-Bibliotheken wie rclcpp oder rclpy, um mit dem ROS 2-System zu interagieren. Jeder Node hat einen eindeutigen Namen und kann eine oder mehrere Funktionen ausführen, z. B. das Steuern eines Roboters, das Verarbeiten von Sensordaten oder das Ausführen von Algorithmen zur Pfadplanung.

Nodes kommunizieren miteinander über Topics, die als Kanäle für den Nachrichtenaustausch dienen. Ein Node kann Nachrichten an ein Topic veröffentlichen, und andere Nodes können dieses Topic abonnieren, um die Nachrichten zu empfangen. Dadurch entsteht ein verteiltes System von Nodes, die miteinander kommunizieren und zusammenarbeiten, um komplexe Aufgaben auszuführen.

Zusätzlich zur Nachrichtenkommunikation können Nodes auch auf den Parameter-Server zugreifen, um Konfigurationsparameter abzurufen oder zu setzen. Dies ermöglicht es Nodes, ihre Konfiguration zur Laufzeit anzupassen, ohne den Quellcode ändern zu müssen.

Ein Node durchläuft einen definierten Lebenszyklus, der Initialisierung, Ausführung und Beendigung umfasst. ROS 2 stellt Mechanismen zur Verfügung, um den Lebenszyklus eines Nodes zu verwalten und sicherzustellen, dass er ordnungsgemäß gestartet und beendet wird.

Wie man einen Node Schritt für Schritt schreibt, wird im Teil IMU-Sensor ausführlich erklärt.

3.4.7 Kommunikationsfunktionen

Die zwei wichtigsten Kommunikationsfunktionen in ROS2 sind Topics und Services. Topics werden für Datenströme verwendet und Services für eine Client/Server-Interaktion.

3.4.7.1 Topic

Um eine allgemeine Definition zu haben, ist ein Topic ein bestimmter Bus, über den Nodes Nachrichten austauschen.

Der Name des Topics muss mit einem Buchstaben beginnen und kann dann Buchstaben, Zahlen, Unterstriche und Schrägstriche enthalten. Ein Topic wird oft dann verwendet, wenn Sie einen Datenstrom benötigen. Einige Knoten (Nodes) können in einem Topic veröffentlichen und einige Knoten können ein Topic abonnieren.

Es gibt keine Antwort von einem Subscriber zu einem Publisher, mit anderen Worten, die Daten gehen nur in eine Richtung. Das bedeutet, dass der Datenstrom unidirektional ist.

Der Publisher und der Subscriber sind voneinander anonym. Der Publisher weiß nur, dass er ein Topic veröffentlicht und der Subscriber weiß nur, dass er ein Topic abonniert, mehr nicht.

Ein Topic hat einen Nachrichtentyp. Alle Publisher und Subscriber des Topics müssen denselben Nachrichtentyp verwenden, der dem Topic entspricht.

Die rclpy-Bibliothek in Python verfügt ebenfalls über die Topic-Funktionalität. Der Subscriber/Publisher kann also in Python direkt innerhalb eines Nodes geschrieben werden.

Es ist zu erwähnen, dass ein Node viele Publisher und Subscriber aus vielen verschiedenen Themenbereichen enthalten kann.

3.4.8 Launch-System

Ein Launch-Datei ist eine XML-Datei, die verwendet wird, um die Ausführung von ROS2-Knoten zu orchestrieren und deren Parameter zu konfigurieren. Diese Dateien dienen dazu, mehrere Knoten auf einmal zu starten und ihre Konfiguration zu vereinfachen. Das ist genau der Grund, warum die Launch-Datei in unserem Auto verwendet wurde.

Die Verwendung einer Launch-Datei für die Ausführung eines Teleoperationsautos bietet zahlreiche Vorteile. Sie ermöglicht eine effiziente und zuverlässige Start- und Konfigurationsverwaltung durch das gleichzeitige Starten mehrerer Knoten und die zentrale Spezifikation von Parametern und Kommunikationseinstellungen. Die Wiederverwendbarkeit der Launch-Datei erleichtert die Anpassung des Systems an verschiedene Szenarien und Umgebungen, während die Modularität die Entwicklung, Wartung und Erweiterbarkeit des Teleoperationsautos unterstützt. Die Vermeidung von Konfigurationsfehlern wird durch die präzise Spezifikation in der Launch-Datei erleichtert, was zu einer verbesserten Entwicklungs- und Betriebseffizienz führt.

wie die Launch-Datei geschrieben wird, wird im Teil Launch Datei erklärt.

Quelle: <https://docs.ros.org/en/foxy/>

3.5 ROS2-Architektur (Arash Barabadi)

Nun, da die grundlegenden Konzepte von ROS definiert worden sind. Sie sollen jetzt in unserem Auto in die Praxis umgesetzt werden. Wie aus dem folgenden Diagramm ersichtlich ist, wird das rqt-Diagramm des gesamten Fahrzeugs zunächst gezeigt, um einen Überblick darüber zu geben, was im Projekt in Bezug auf die Definitionen in ROS2 getan wurde.

Node-Darstellung mit **Ellipse**.

Topic-Darstellung mit **Rechteck**.

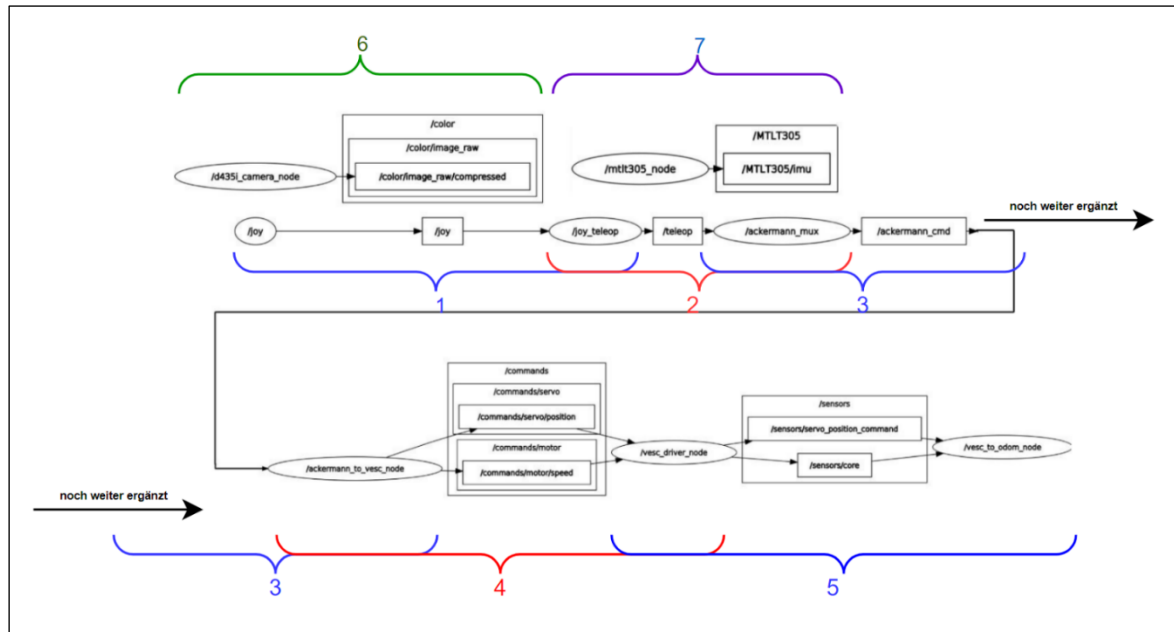


Abbildung 31: Die ROS2_Architektur

Der Prozess der Befehlsübertragung vom Lenkrad zu den Motoren wird wie folgt ablaufen.

1:

Publisher	/joy	Das Dienstprogramm für die Teleoperation über die Tastatur spielt eine Rolle als Befehlsgeber
Topic	/joy	Dient als Schnittstelle für ROS-Nodes, um Eingaben von einem Joystick oder Gamecontroller zu empfangen und so das Verhalten des Autos auf der Grundlage der Steuereingaben des Benutzers zu steuern.
Subscriber	/joy_teleop	Ein Node zur Umsetzung von Joystick-Steuerungen in Auto-Teleoperationsbefehle.

Tabelle 3: Die Datenverbindung zwischen dem Joystick und dem vordefinierten Teleoperationsknoten

Topic: **/joy**

Das Topic /joy ist ein Standard-ROS 2-Topic, das für die Veröffentlichung von Joystick-Eingabedaten verwendet wird. Dieses Topic veröffentlicht normalerweise Nachrichten des Typs **sensor_msgs/Joy**.

Subscriber: **joy_teleop.py** code

Dieser Code definiert einen ROS2-Knoten namens "JoyTeleop", der die Steuerung eines Autos über einen Joystick ermöglicht. Hier ist eine Erklärung des Codes:

Importe: Importiert benötigte ROS2-Module wie rclpy für die ROS2-Kommunikation und sensor_msgs.msg für Nachrichten wie Joystick-Ereignisse.

Ausnahmen: Definiert eine benutzerdefinierte Ausnahme JoyTeleopException, die für Fehler im JoyTeleop-Programm verwendet wird.

Hilfsfunktionen: Enthält Hilfsfunktionen wie get_interface_type und set_member, um Nachrichtentypen zu verarbeiten und ihre Mitglieder zu setzen.

JoyTeleopCommand-Klasse: Die Basis für alle Joystick-Teleop-Befehle. Es enthält Methoden zum Aktualisieren des Befehlsstatus basierend auf Joystick-Ereignissen.

JoyTeleopTopicCommand-Klasse: Eine Unterklasse von JoyTeleopCommand, die Befehle über ROS2-Themen veröffentlicht.

JoyTeleopServiceCommand-Klasse: Eine Unterklasse von JoyTeleopCommand, die Befehle über ROS2-Dienste aufruft.

JoyTeleopActionCommand-Klasse: Eine Unterklasse von JoyTeleopCommand, die Befehle über ROS2-Aktionen ausführt.

JoyTeleop-Klasse: Der Hauptknoten, der JoyTeleop initialisiert, Befehle aus der Konfiguration lädt und Joystick-Nachrichten empfängt.

Hauptfunktion: Initialisiert den ROS2-Knoten und führt ihn aus.

Dieser Knoten (Node) liest Konfigurationsparameter, um die gewünschten Joystick-Befehle zu laden, und abonniert dann auf das ROS2-Thema (Topic) **/joy**, um Joystick-Ereignisse zu empfangen und entsprechend zu reagieren. Generell bietet dieser Code einen Rahmen für die Teleoperation eines Autos mit einem Joystick in einer ROS2-Umgebung und ermöglicht die Anpassung des Teleoperationsverhaltens durch Konfigurationsparameter.

2:

Publisher	/joy_teleop	
Topic	/teleop	überträgt die lesbaren Daten für den Node ackermann_mux.
Subscriber	/ackermann_mux	basierend auf twist_mux Ackermann-Multiplexer mit Unterstützung für <u>ackermann_msgs/AckermannDriveStamped</u> Topics und <u>std_msgs/Bool</u> Sperren mit Prioritäten.

Tabelle 4: Die Datenverbindung zwischen Teleoperationsknoten und Ackermann-Modell

3:

Publisher	/ackermann_mux	
Topic	/ackermann_cmd	Der Kanal, über den Ackermann-Steuerbefehle gesendet werden.

Subscriber	/ackermann_to_vesc_node	Bei diesem Code handelt es sich um einen ROS2-Node, der Nachrichten des Typs 'ackermann_msgs::msg::AckermannDriveStamped' empfängt, die Ackermann-Antriebsbefehle darstellen, und sie in für VESC (Vedder Electronic Speed Controller) verständliche Befehle umwandelt.
------------	-------------------------	--

Tabelle 5: Die Datenverbindung zwischen Ackermann-Modell und VESC-bezogenem Knoten

Topic: **/ackermann_cmd**

Der **/ackermann_cmd** Topic ist ein Kommunikationskanal, über den Ackermann-Steuerbefehle gesendet werden. In ROS 2 ermöglicht er anderen Knoten oder Komponenten das Veröffentlichen von Nachrichten des Typs **ackermann_msgs::msg::AckermannDriveStamped**, die in der Regel Informationen wie die gewünschte Geschwindigkeit und Lenkwinkel für ein Fahrzeug mit Ackermann-Lenkung enthalten. Der Node **/ackermann_to_vesc_node** abonniert dieses Topic, um diese Steuerbefehle zu empfangen, zu verarbeiten und in Befehle umzuwandeln, die von VESC-Motorsteuerungen verstanden werden, um die Motorgeschwindigkeit und den Lenkwinkel des Fahrzeugs weiter zu steuern.

Subscriber: **ackermann_to_vesc_node.cpp** code

- **Namespace und Verwendung von Deklarationen:** Der Code ist im Namespace **'vesc_ackermann'** definiert und enthält Nutzung von Deklarationen, um die Verwendung von Nachrichtentypen und Platzhaltern zu vereinfachen.
- **Konstruktor:** Der Konstruktor initialisiert den Node und ruft die Konvertierungsparameter vom ROS2-Parameterserver ab. Er erstellt außerdem Publisher für VESC-Befehle (**commands/motor/speed** und **commands/servo/position**) und abonniert das Topic **/ackermann_cmd**.
- **ackermannCmdCallback** Funktion: Diese Funktion wird immer dann aufgerufen, wenn eine Nachricht zum Topic **/ackermann_cmd** empfangen wird. Sie berechnet die elektrische VESC-Drehzahl (Geschwindigkeit) und den Lenkwinkel (Servoposition) anhand der empfangenen Ackermann-Fahrbefehle. Anschließend veröffentlicht sie diese Befehle in den entsprechenden Topics, wenn sich der Node noch in einem gültigen Zustand befindet (**rclcpp::ok()**).
- **Registrierung des Nodes:** Der Node wird mit **RCLCPP_COMPONENTS_REGISTER_NODE** als Komponente registriert, so dass er dynamisch geladen und innerhalb eines ROS2-Systems verwendet werden kann.

4:

Publisher	/ackermann_to_vesc_node	
Topic	/commands/servo/position	- dient zum Empfang von Befehlen zur Einstellung der Servoposition.
	/commands/motor/speed	- dient zum Empfang von Befehlen zur Einstellung der Motordrehzahl.

Subscriber	/vesc_driver_node	Dieser Node ist ein Treiber für VESC. Der Knoten abonniert verschiedene Befehlstypen und veröffentlicht Telemetriedaten des VESC sowie IMU-Daten.
------------	-------------------	---

Tabelle 6: Die Datenverbindung zwischen dem VESC-bezogenen Knoten und dem VESC-Treiberknoten

Topic: **/commands/servo/position**

Dieses Topic ist ein Abonnement-Topic für den Empfang von Befehlen zur Einstellung der Position des an den VESC angeschlossenen Motors. Die in diesem Topic erwarteten Nachrichten sind vom Typ `std_msgs::msg::Float64`.

Topic: **/commands/motor/speed**

Dieses Topic ist ein Abonnement-Topic für den Empfang von Befehlen zur Einstellung der Drehzahl des an den VESC angeschlossenen Motors. Die für dieses Topic erwarteten Nachrichten sind vom Typ `std_msgs::msg::Float64`.

Subscriber: **/vesc_driver_node**

Bei diesem Code handelt es sich um einen ROS2-Node, der als Treiber für einen VESC-Motorcontroller (Vedder Electronic Speed Controller) dient. Er abonniert verschiedene ROS2-Topics, um Befehle zur Steuerung des VESC zu erhalten, und veröffentlicht Telemetriedaten vom VESC.

Die wichtigsten Komponenten und Funktionen des oben genannten Codes sind folgende;

(Enthält & Namensraum) Includes & Namespace: Der Code enthält die erforderlichen Header für ROS2-Nachrichten, VESC-bezogene Nachrichten und andere Standard-C++-Bibliotheken. Der Code ist Teil des Namespace `vesc_driver`.

Konstruktor: Der Konstruktor von `VescDriver` initialisiert den Node mit dem Namen `"vesc_driver"` und richtet verschiedene Parameter und Grenzwerte für die Steuerung des VESC ein.

Initialisierung: Der Konstruktor initialisiert das VESC-Objekt, stellt eine Verbindung zur seriellen Schnittstelle her und erstellt Publisher für Telemetriedaten (Status, IMU) und einen Subscriber für den Empfang von Motor- und Servo-Befehlen.

Timer-Rückruf: Eine Timer-Callback-Funktion (`timerCallback`) wird in regelmäßigen Abständen (alle 20ms) aufgerufen. Sie verwaltet die Zustandsmaschine des Treibers und fragt den VESC nach Telemetriedaten ab.

VESC-Paket Rückrufe: Die Funktion `vescPacketCallback` wird aufgerufen, wenn ein neues VESC-Paket empfangen wird. Sie behandelt verschiedene Arten von Paketen, wie `"Values"`, `"FWVersion"` und `"ImuData"`. Je nach Pakettyp veröffentlicht sie die relevanten Informationen in ROS2-Topics.

Befehlsrückrufe (Command Callbacks): Der Nodes abonniert verschiedene Topics für Motor- und Servo-Befehle (*duty_cycle*, *current*, *brake*, *speed*, *position*, *servo*). Callback-Funktionen für diese Abonnements (*dutyCycleCallback*, *currentCallback*, etc.) senden entsprechende Befehle an den VESC.

Befehlsbegrenzung: Die Klasse *CommandLimit*(*CommandLimit*) wird verwendet, um die empfangenen Befehlswerte innerhalb bestimmter Bereiche zu begrenzen und zu beschneiden.

Telemetrie-Veröffentlichung: Die vom VESC empfangenen Telemetriedaten werden in ROS2-Nachrichten verpackt und in Topics veröffentlicht (*sensors/core*, *sensors/imu*, *sensors/imu/raw*, etc.).

Treiber-Zustandsmaschine: Der Treiber hat einen einfachen Zustandsautomaten mit zwei Modes: *MODE_INITIALIZING* und *MODE_OPERATING*. Während der Initialisierung wird die VESC-Firmware-Version abgefragt, und nach der Initialisierung geht er in den Betriebsmodus über, in dem er kontinuierlich Telemetriedaten abfragt.

Fehlerbehandlung: Die Funktion *vescErrorCallback* wird im Falle von Fehlern, die von der VESC-Schnittstelle gemeldet werden, aufgerufen. Sie loggt die Fehlermeldung.

Parameter-Verarbeitung: Der Node ruft Parameter wie die Adresse der seriellen Schnittstelle und die Befehlsgrenzen vom ROS2-Parameterserver ab.

Node Registrierung: Das Makro *RCLCPP_COMPONENTS_REGISTER_NODE* registriert den *VescDriver* als ROS2-Node.

5:

Publisher	/vesc_driver_node	
Topic	/sensors/servo_position_command /sensors/core	Sendet Befehle zur Steuerung der Position oder des Winkels eines Servomotors. Übermittelt VESC-Zustandsinformationen an andere ROS-Nodes im System.
Subscriber	/vesc_to_odom_node	abonniert Nachrichten von zwei Topics, verarbeitet sie entsprechend und veröffentlicht die resultierenden Odometriedaten.

Tabelle 7: Die Datenverbindung zwischen VESC-Treiberknoten und VESC_to_odom-Knoten.

Topic: **sensors/servo_position_command**

Das Thema "sensors/servo_position_command" wird verwendet, um Befehle zur Steuerung der Servoposition zu veröffentlichen. Der Servomotor wird im Auto zur Steuerung von Lenkmechanismen verwendet, die eine präzise Winkelpositionierung erfordern.

Die zu diesem Topic veröffentlichten Nachrichten sind vom Typ Float64, einem Standard-ROS-Nachrichtentyp, der eine Gleitkommazahl darstellt. Die Float64-Nachricht enthält den gewünschten Positions- oder Winkelbefehl für den Servomotor.

Zusammenfassend lässt sich sagen, dass das Topic "sensors/servo_position_command" verwendet wird, um Befehle zur Steuerung der Position oder des Winkels eines Servomotors zu senden, so dass externe Nodes die gewünschte Position für die Betätigung eines Servomotors angeben können.

Topic: **/sensors/core**

Das Topic "sensors/core" wird für die Veröffentlichung von VESC-Zustandsmeldungen verwendet. VESC wird häufig in der Robotik und anderen Anwendungen zur Steuerung von Motoren, insbesondere in Elektrofahrzeugen, verwendet.

Die zu diesem Topic veröffentlichten Nachrichten sind vom Typ VescStateStamped, einem benutzerdefinierten Nachrichtentyp, der im Paket vesc_msgs definiert ist.

Diese Nachricht enthält wahrscheinlich Informationen über den aktuellen Zustand des VESC, wie Motordrehzahl, Motorstrom, Spannung, Temperatur usw.

Subscriber: **/vesc_to_odom_node**

Dieser Node stellt einen ROS2-Subscriber Node dar, der zwei Topics abonniert: "sensors/core" und "sensors/servo_position_command". Hier ist eine Erklärung des Codes:

Header-Dateien einbeziehen (Include): Der Code enthält verschiedene Header-Dateien, die für die Verwendung von ROS2-Nachrichten und anderen Funktionen benötigt werden.

Namespace-Deklaration: Der Code ist im Namespace vesc_ackermann organisiert.

Klassendefinition: Es wird eine Klasse VescToOdom definiert, die von rclcpp::Node erbt. Diese Klasse repräsentiert den Node, der die Abonnements verwaltet und die Callback-Funktionen für eingehende Nachrichten bereitstellt.

Konstruktor: Im Konstruktor werden verschiedene Parameter initialisiert, darunter Rahmen, Flaggen und Initialwerte für Position und Geschwindigkeit.

Initialisierung der Abonnements: Der Konstruktor enthält Logik zum Erstellen von Abonnements für die Topics "sensors/core" und "sensors/servo_position_command". Diese Abonnements sind mit entsprechenden Callback-Funktionen verbunden, die aufgerufen werden, wenn neue Nachrichten auf den Themen empfangen werden.

Callback-Funktionen:

- **vescStateCallback:** Diese Funktion wird aufgerufen, wenn eine Nachricht auf dem Topic "sensors/core" empfangen wird. Sie verarbeitet den empfangenen Zustand und berechnet die Odometrie basierend auf dem aktuellen Zustand und der vergangenen Odometrie.

- **servoCmdCallback:** Diese Funktion wird aufgerufen, wenn eine Nachricht auf dem Topic "sensors/servo_position_command" empfangen wird. Sie speichert den empfangenen Servo-Befehl für die spätere Verwendung.

Registrierung des Nodes: Am Ende wird der Knoten für die Ausführung in einem ROS2-System registriert.

6: & 7:

Die Teile 6 und 7, bei denen es sich um die Kamera bzw. die IMU-Sensor-Verarbeitungseinheit handelt, werden in separaten Abschnitten behandelt, die ihren Namen tragen.

3.6 Mesh Netzwerk Konfiguration (Albin Hrabos)

Der Mesh Netzwerk besteht aus dem Mesh Router **EAP610-Outdoor**. Dieses Gerät wird mit der Handy Anwendung **Omada** konfiguriert. Während der Konfiguration muss der Router sowohl mit Stromquelle als auch mit Internet per Ethernet Kabel verbunden werden. Der Name des Netzwerks wird "projekt1" genannt. Nach der Konfiguration braucht der Mesh Netzwerk keine Internetverbindung.

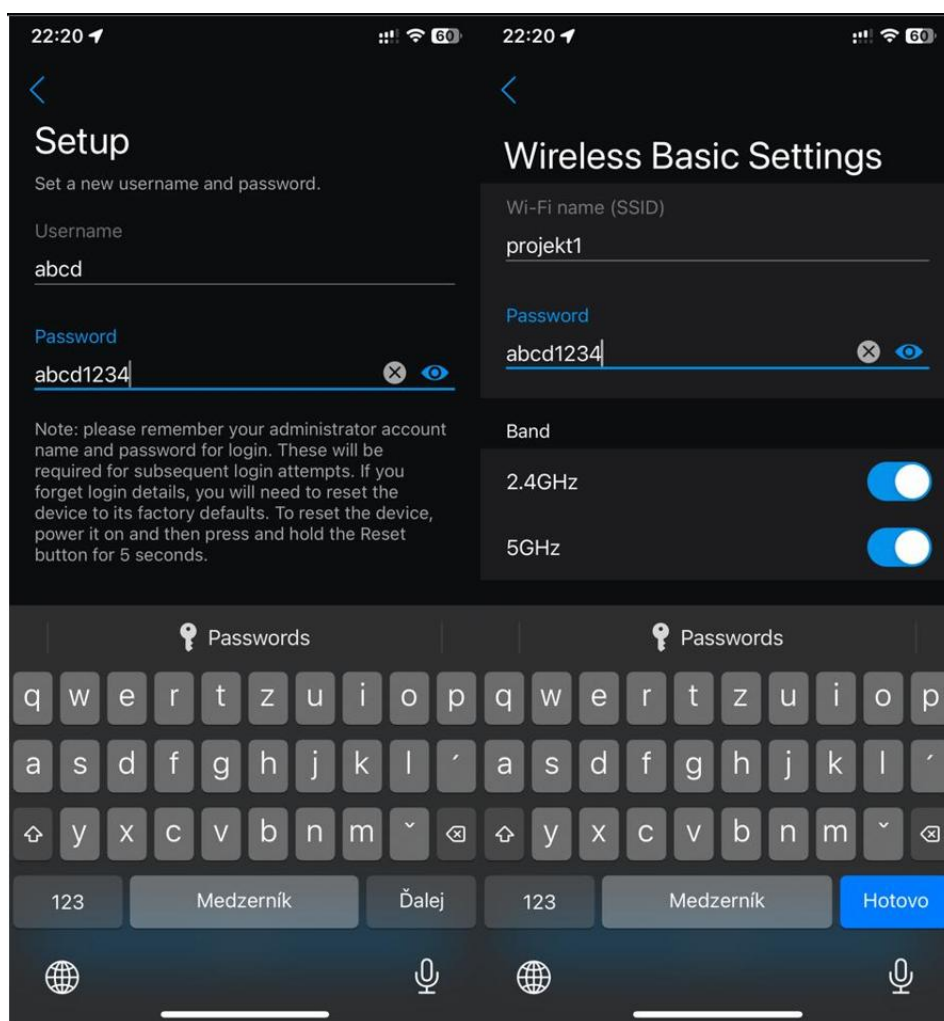


Abbildung 32: Setup von Mesh Router in der Anwendung Omada

für Jetson wird Adress und Netmask manuell, wie unten eingestellt, bevor es mit dem “projekt1” Netzwerk verbunden wird.

The screenshot shows the 'projekt1' network configuration window in Ubuntu. The 'IPv4' tab is selected. Under 'IPv4 Method', 'Manual' is chosen. The 'Addresses' table has one entry with IP '192.168.0.201' and Netmask '255.255.255.0'. The 'DNS' and 'Routes' sections are both set to 'Automatic'.

Address	Netmask	Gateway
192.168.0.201	255.255.255.0	

Address	Netmask	Gateway	Metric
---------	---------	---------	--------

Abbildung 33: Einstellung der Netzwerkverbindung im Ubuntu

Um beste Verbindung gewährleisten zu können ist es wichtig, dass der Rechner (Laptop) per Ethernet-Kabel zu dem Mesh Router verbunden wird. Der Rechner sollte die gleichen Netzwerk Einstellungen wie oben haben nur statt 201 eine andere Zahl haben z.B 202, falls er sowohl per Kabel als auch per W-Lan zu dem Netzwerk projekt1 verbunden ist.

Die Verbindung des Jetsons mit dem Mesh Netzwerk kann man in dem Tp-Link Konfiguration Umfeld überprüfen. Die Adresse lautet: <http://192.168.0.254/> Die Anmeldedaten sind oben in Handy Screenshots zu sehen. In folgendem Bild ist die “Client List”, also die Überprüfung ob Jetson mit dem Mesh Netzwerk verbunden ist:

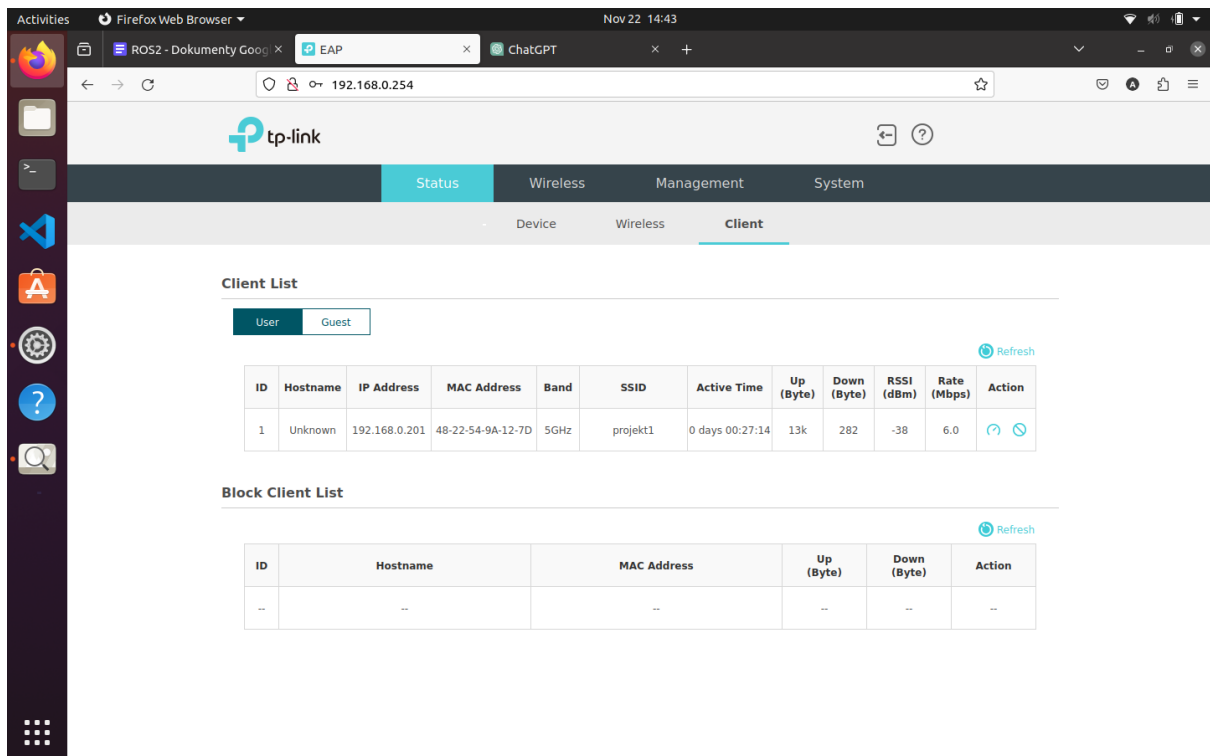


Abbildung 34: Client List von Mesh Router

3.7 Treiber von W-Lan Modul (Arash Barabadi)

Für die Kommunikation über Mesh Netzwerk wird ein W-Lan Modul auf das Fahrzeug per Kabel angeschlossen, das eine bessere Reichweite dank zwei verstellbaren Antennen bietet. Treiber von dem W-Lan Modul Tp-Link AX1800 Archer TX20U Plus wird anhand folgender Anleitung auf Jetson installiert:

[Anleitung für W-Lan Modul Treiber Installation](#) [27]

```
sudo apt-get update
sudo apt-get install make gcc linux-headers-$(uname -r) build-essential git
```

```
cd rtl8852au
make
sudo make install
```

```
cd rtl8852au
git pull
make
sudo make install
```

```
# Add module to dkms tree
sudo dkms add.
```

```
# Build
sudo dkms build rtl8852au -v 1.15.0.1

# Install
sudo dkms install rtl8852au -v 1.15.0.1

# Check installation
modinfo 8852au

# Load driver
modprobe 8852au
```

Quelle: [27]

Anschließend wird das Modul in den W-Lan Einstellungen des Jetsons angezeigt und kann an das Mesh Netzwerk mit dem Name projekt1 angeschlossen werden.

3.8 Bildübertragung von Kamera (Albin Hrabos)

Um das Kamerabild über ROS2 übertragen zu können wird das Package mit ROS2 Node für Kamera auf dem Jetson installiert. Die Vorgehensweise ist für Laptop und Jetson identisch.

3.8.1 RealSense Viewer

Zuerst wird Intel Realsense Viewer installiert. Mit dieser Software kann man die Kamerafunktionsfähigkeit überprüfen. Dies kann man auch über direkte Kabelverbindung zwischen Kamera und Laptop testen, indem man den RealSense Viewer auf den Laptop installiert.

[Librealsense Installation](#) [28]

falls curl nicht installiert dann erst folgenden Schritt durchführen:

```
sudo apt-get install curl
```

Weitere Schritte:

```
sudo mkdir -p /etc/apt/keyrings
curl -sSf https://librealsense.intel.com/Debian/librealsense.pgp |
sudo tee /etc/apt/keyrings/librealsense.pgp > /dev/null

echo "deb [signed-by=/etc/apt/keyrings/librealsense.pgp]
https://librealsense.intel.com/Debian/apt-repo `lsb_release -cs` main"
| \
sudo tee /etc/apt/sources.list.d/librealsense.list
sudo apt-get update
```

```
sudo apt-get install librealsense2-dkms
```

```
sudo apt-get install librealsense2-utils
```

RealSense Viewer starten:

```
realsense-viewer
```

Quelle: [28]

3.8.2 Kamera Node

Dieser Test wird immer noch über Kabelverbindung mit dem Laptop durchgeführt später wird der Kamera Node analog auf den Jetson installiert. Erst wird Kamera Node wie folgt installiert:

[Realsense ROS2](#) [29]

Folgende Befehle sind für die Kamera Node Installation:

```
sudo apt install ros-foxy-librealsense2*  
sudo apt install ros-foxy-realsense2-*
```

Danach kann man schon den Kamera Node starten:

```
ros2 run realsense2_camera realsense2_camera_node
```

Nachdem kann in einem neuen Fenster rviz2 gestartet werden:

```
rviz2
```

Den Node mit folgendem Befehl stoppen:

```
Ctrl + C
```

Quelle: [29]

3.9 Kommunikation und Befehleingabe (Albin Hrabos)

Die Kommunikation und entsprechende Befehleingabe ist auf 2 Wege möglich:

1. GVNC-Viewer

Diese Option ermöglicht eine Benutzeroberfläche wie bei einem Rechner. Das Laptopbildschirm wird zu Bildschirm von Jetson.

Installation von GVNC-Viewer:

```
sudo apt-get install gvncviewer
```

GVNC-Viewer starten:

```
gvncviewer 192.168.55.1
```

2. SSH (Secure Shell) Session

Zweite Möglichkeit ist sogenannte Secure Shell Session und die Befehleingabe ist rein über Terminal Fenster.

Secure Shell Session starten:

```
ssh 192.168.0.201 -l nvidia
```

falls die Verbindung nicht funktioniert, dann nochmal Strom einstecken.
(Jetson Passwort: `abcd1234`)

(Herunterfahren von Jetson durch SSH:

```
sudo shutdown now)
```

3.10 Start von Kamera Node auf dem Jetson (Arash Barabadi)

Jetson startet nach dem Anschließen ans Stromnetz automatisch und es sind keine Anmeldedaten benötigt. Mit dieser Einstellung kann erreicht werden, dass nach dem Start keine Kabelverbindung zw. Laptop und Jetson benötigt wird. Zusätzlich wird der Jetson mit dem Mesh Netzwerk projekt1 automatisch verbunden. Es wird eine SSH erstellt, um Einstellungsänderungen über Terminal im Jetson vorzunehmen. Dafür wird folgender Befehl durchgeführt:

```
ssh 192.168.0.201 -l nvidia
```

Danach wird nur der Befehl für Jetson für den Kamera-Node-Start eingetippt:

```
ros2 run realsense2_camera realsense2_camera_node
```

Auf dem Laptop wird nun rviz2 gestartet:

```
rviz2
```

Diese Vorgehensweise ist eine Vereinfachung. Die vollständige Vorgehensweise besteht aus mehreren Schritten, die aber in der "Bashrc" Datei integriert wurden, d.h. die werden nach dem Start des Geräts

automatisch durchgeführt. Diese Automatisierung wird im nächsten Kapitel [Automatisierung von Befehlen](#) erklärt. Eine vollständige Vorgehensweise sieht wie folgt aus:

Befehle Jetson:

```
export ROS_DOMAIN_ID=34
source /opt/ros/foxy/setup.bash
ros2 run realsense2_camera realsense2_camera_node
```

Befehle Laptop:

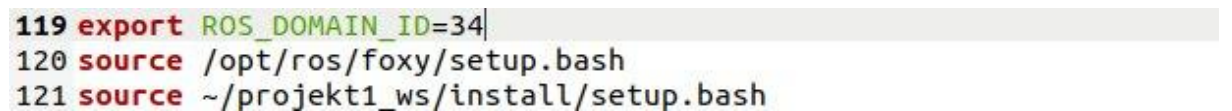
```
export ROS_DOMAIN_ID=34
source /opt/ros/foxy/setup.bash
rviz2
```

weitere nützliche Befehle - Info Kamerabild:

```
ros2 topic echo /color/camera_info
ros2 topic Hz /color/image_raw
```

3.11 Automatisierung von Befehlen (Albin Hrabos)

Um manche Schritte sich sparen zu können und die Startvorgänge zu verkürzen werden in Bashrc Datei Zeilen 119 bis 121 hinzugefügt. Folglich kann bei Neustart der ROS2 Startbefehl und die Domäne Einstellung ausgelassen werden.



```
119 export ROS_DOMAIN_ID=34
120 source /opt/ros/foxy/setup.bash
121 source ~/projekt1_ws/install/setup.bash
```

Abbildung 35: Bashrc Automatisierung Zeilen

3.12 Verzögerung des Systems (Albin Hrabos)

Im Rahmen des Projekts wird die Verzögerung des Systems beurteilt. Einfluss auf das Gesamtsystem haben sowohl Hardware als auch Softwarekomponenten und auch der Fernfahrer. Erst wird die Verzögerung der Videoübertragung gemessen.

3.12.1 Video-Verzögerung im Mesh Netzwerk

Es wird gemessen, wie schnell das reale Bild auf dem Bildschirm des Laptops übertragen wird. Es wird z.B. Licht einer Ampel simuliert. Für gute Sichtbarkeit des "Ampellichts" bei jeder Kameraauflösung würde das Blitzlicht von Handy als Ampellicht verwendet. Die Videoverzögerung wurde mit iPad Kamera gemessen. Die Versuchsaufbau war folgend: die iPad Kamera hat gleichzeitig das reale und auf dem Laptopbildschirm im rviz2 dargestellte Lichtimpuls aufgenommen (siehe Bild unten). Aus dieser Videoaufnahme könnte die Verzögerung berechnet werden. Die Abweichung dieser Messung kann ca. 0,02 s betragen, da die Videoaufnahme von iPad 60 fps ist.

Tabelle 8: Messdaten der Verzögerung

Mesh Ausrichtung	Auflösung	Ausrichtung W-Lan Modul	Verzögerung	Bildqualität
horizontal	1280 x 720p 30fps	schräg seitlich	0,30 s \pm 0,02 s	raw (rviz2)
horizontal	640 x 480p 30fps	schräg seitlich	0,17 s \pm 0,02 s	raw (rviz2)
horizontal	640 x 480p 30fps	vertikal seitlich	0,20 s \pm 0,02 s	raw (rviz2)
horizontal	640 x 480p 30fps	vertikal Vorderseite	0,13 s \pm 0,02 s	raw (rviz2)
horizontal	640 x 480p 30fps	horizontal seitlich	0,16 s \pm 0,02 s	raw (rviz2)
vertikal	640 x 480p 30fps	schräg seitlich	0,18 s \pm 0,02 s	raw (rviz2)
horizontal	640 x 480p 30fps	vertikal Vorderseite	0,13 s \pm 0,02 s	komprimiert (rqt)

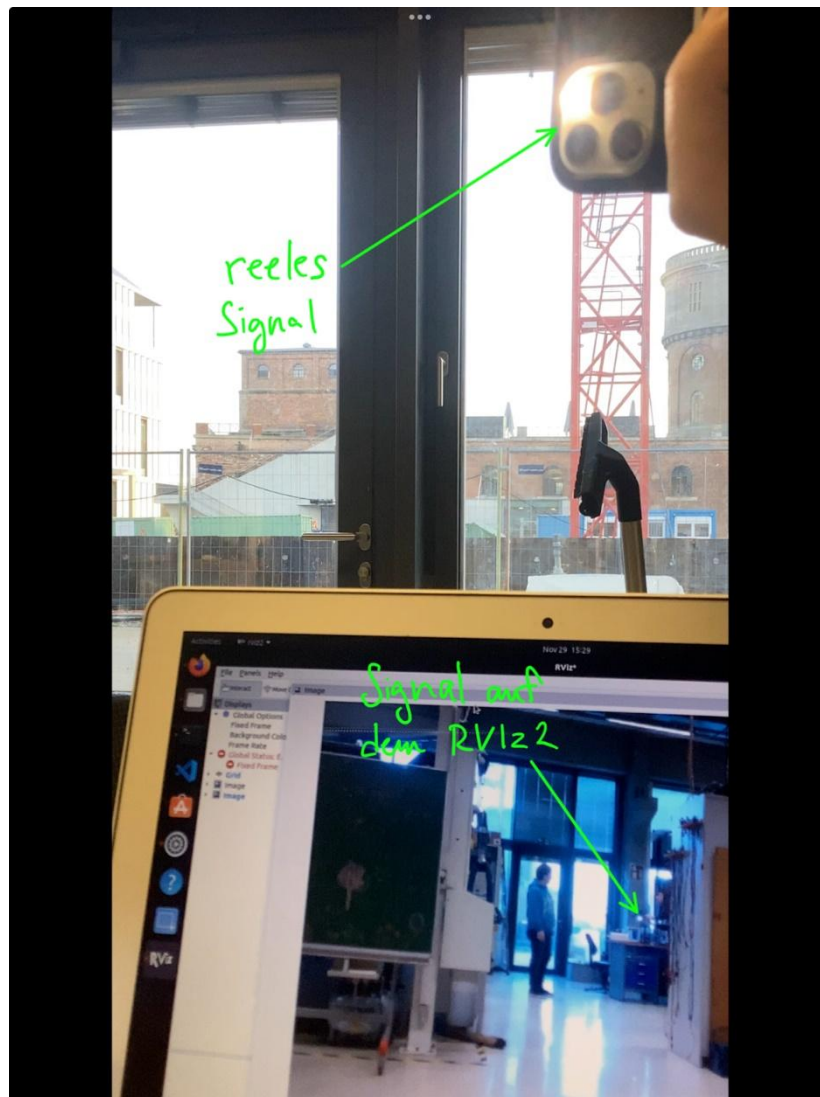


Abbildung 36: Versuchsaufbau der Messung von der Verzögerung auf ca. 10 m

3.12.2 Gesamtverzögerung

Die Gesamtverzögerung wurde während der teleoperierten Fahrt subjektiv beurteilt. Bei dem Test konnte man während einer stabilen Verbindung mit dem Mesh Netzwerk kein Aufhängen des Kamera Bildes oder der Teleoperation - Vorgabe von Pedal und Lenkrad nicht beobachten und auch keine von Menschen merkbare Verzögerung.

3.13 Bild Komprimierung (Albin Hrabos)

Um die Geschwindigkeit der Bildübertragung und dementsprechend die gemessene Verzögerung zu reduzieren und auf dem geringeren Wert stabil halten zu können, wurde ein Package auf dem Jetson installiert, der die Bildinformation komprimiert. Die entsprechenden Schritte sind unter folgendem Link beschrieben:

[Compression Package \[30\]](#)

Die Befehle, die für ROS2 Foxy angepasst sind, sehen folgendermaßen aus:

Schritt 1 - die "compressed topics" zu erlauben:

```
sudo apt install ros-foxy-image-transport  
(sudo apt get update)
```

Schritt 2 - compressed-image-transport package wird installiert:

```
sudo apt install ros-foxy-compressed-image-transport  
(sudo apt get update)
```

Quelle: [30]

Nach dieser Anpassung wird das komprimierte Bild nicht mehr in dem rviz2 Fenster angezeigt, sondern in rqt Umfeld. Man startet es einfach mit dem Befehl:

```
rqt
```

Nach diesem Befehl wird rqt Fenster aufgemacht, wo man das komprimierte Bild mit wenigen Klicks aufrufen kann:

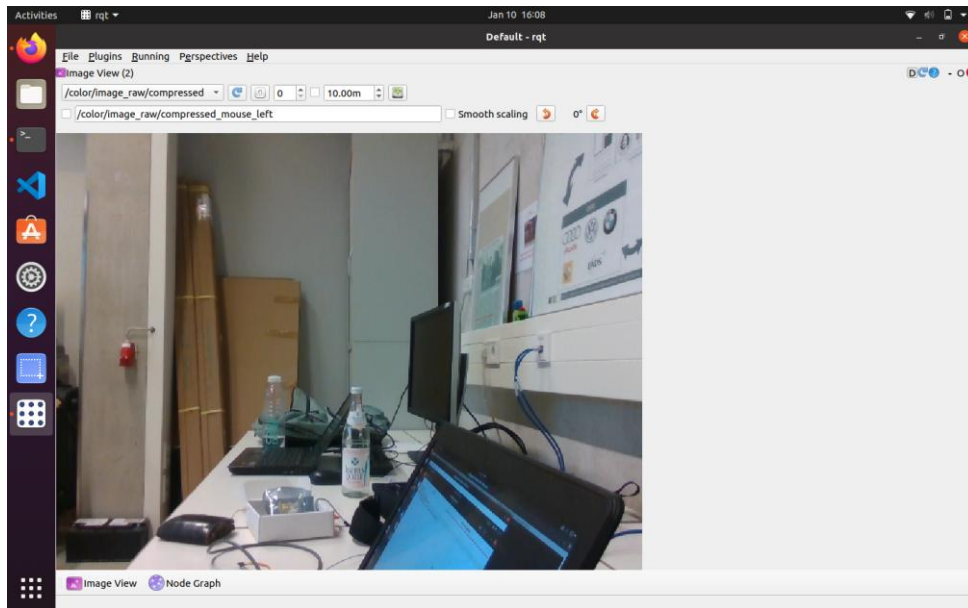


Abbildung 37: Komprimiertes Bild aus der Kamera

Im Vergleich zu originalem Bild können keine qualitativen Unterschiede festgestellt werden:

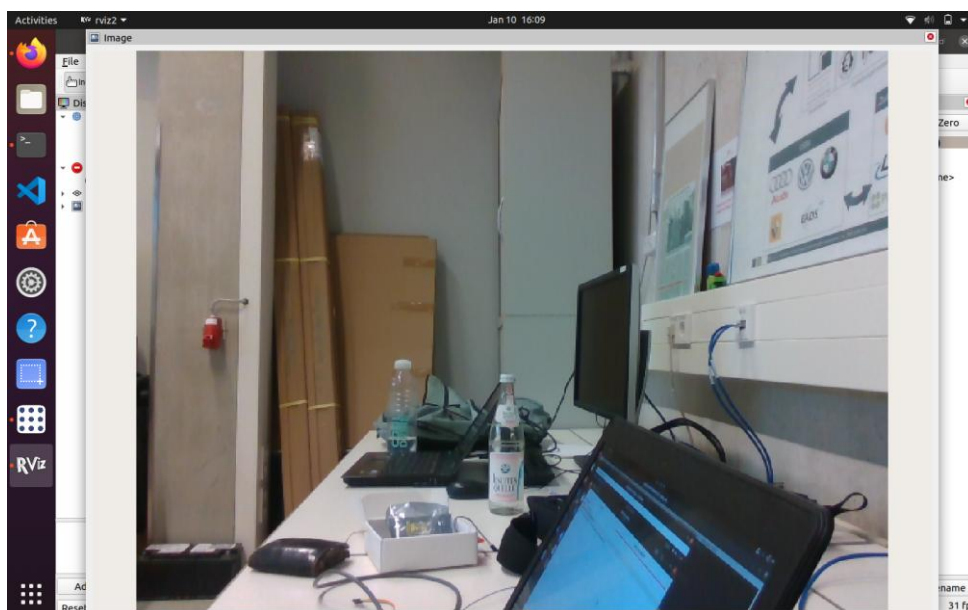


Abbildung 38: Das Originalbild aus der Kamera

Was aussagekräftiger für das komprimierte Bild spricht, ist der Datenfluss der von 26 MBit/s auf ein Zehntel, d.h. 2,6 MBit/s gesunken ist.

3.14 Start der Teleoperation (Albin Hrabos)

Um die Teleoperation möglichst einfach zu starten, wird sowohl auf dem Jetson als auch auf dem Laptop jeweils nur ein Befehl, mit dem Launch Packages geöffnet werden, verwendet. Diese Launch Packages führen dann das Starten allen ROS2 Nodes auf der Fahrzeug- bzw. Fahrerseite automatisch durch. Den Vorgang aus dem Kapitel [Start von Kamera auf dem Jetson](#) muss auch nicht mehr wiederholt werden.

Befehl Jetson:

```
ros2 launch f1tenth_stack bringup_launch.py
```

Befehl Laptop:

```
ros2 launch f1tenth_stack bringup_remote_teleop_launch.py
```

Letztendlich wird das rqt Fenster mit dem Kamera Bild auf dem Laptop aufgemacht:

```
rqt
```

3.15 Rückkehren des Lenkrades zur Mittelposition (Albin Hrabos)

Damit das Fahrgefühl realistischer sein kann, kann diese Funktion optional installiert werden. Nach dem Einlenken wird das Lenkrad zurück zur Mittelposition kehren. Es ist in diesem Fall wichtig, dass die Geometrie der Radaufhängung und die Lenkung kalibriert sind. Um diese Funktion zu aktivieren, die unten angepassten Befehle anwenden.

[ros-g29-force-feedback](#) [31]

Falls schon Kamera Node installiert wurde, muss man diesen erweiterten Befehl verwenden statt in der Anleitung um die **Fehlermeldung zu vermeiden**:

```
colcon build
colcon build --packages-skip realsense2_camera

colcon build --symlink-install
colcon build --symlink-install --packages-skip realsense2_camera
```

Quelle: [31]

Node starten

in untenstehendem Befehl ist Ordner "ros_g29_force_feedback" genannt. Diese Ordner heißt aber "ros-g29-force-feedback" **Man muss in dem Namen des Ordners die "-" mit "_" ersetzen, um den Node starten zu können.**

```
source ros2_ws/install/setup.bash
ros2 run ros_g29_force_feedback g29_force_feedback --ros-args - -
params-file ros2_ws/src/ros_g29_force_feedback/config/g29.yaml
```

Publisher

neues Fenster öffnen und wieder folgenden Befehl verwenden:

```
source ros2_ws/install/setup.bash
ros2 topic pub /ff_target ros_g29_force_feedback/msg/ForceFeedback
"{header: {stamp: {sec: 0, nanosec: 0}, frame_id: ''}, position: 0.3,
torque: 0.5}"
```

Quelle: [31]

3.16 Andere Befehle (Albin Hrabos)

```
ctrl + c Aktion stoppen
ls list all files/folders
cd <file name> open file
cd ../ zurück zum oberen Ordner
python3 --version check Python version
lsusb list von verknüpften USB-Geräten
source /opt/ros/foxy/setup.bash ROS2 starten
ros2 node list
ros2 topic list
ros2 node info <node_name>
colcon build nachdem die Codes geändert werden
rqt_graph Baum von den Nodes anzeigen
```

Kopieren von dem Package von Jetson auf den Laptop:

```
scp -r nvidia@192.168.0.201:/home/nvidia/projekt1_ws /home/albin/
```

3.17 VESC –Tool (Shubham Rawat)

Hier wird erklärt, über VESC-Tool und wie kann es mit ESC-Hardware konfigurieren. Das VESC-Tool ist eine Software zur Konfiguration von ESC-Hardware und das Hochladen der neuesten Firmware. Mit VESC können verschiedene Parameter wie z.b. Motor- und Steuerungseinstellungen konfigurieren.

Schritte zur Konfiguration

1. VESC-Tool herunterladen und installieren: Sicherstellen, dass die neueste Version des VESC-Tools auf Rechner installiert ist. Es Kann von der offiziellen VESC-Projekt-Website herunterladen.

Link zum Herunterladen: <https://vesc-project.com/node/17>

2. Das VESC mit Rechner verbinden: VESC ist mit Rechner über ein USB-Kabel verbindet.

3. System einschalten und VESC-Tool öffnen: Das VESC-Tool auf Rechner starten.

4. VESC-Tool mit dem ESC-Hardware verbinden: Im VESC-Tool auf die **Connect Button** klicken, um eine Verbindung zwischen dem VESC und dem Computer herzustellen.

5. Den gewünschten Motor auswählen: Die Abbildung zeigt, dass zwei Motortypen, ein Servo- und ein Gleichstrommotor, verbunden sind. Der gewünschte Motor ist ausgewählt.

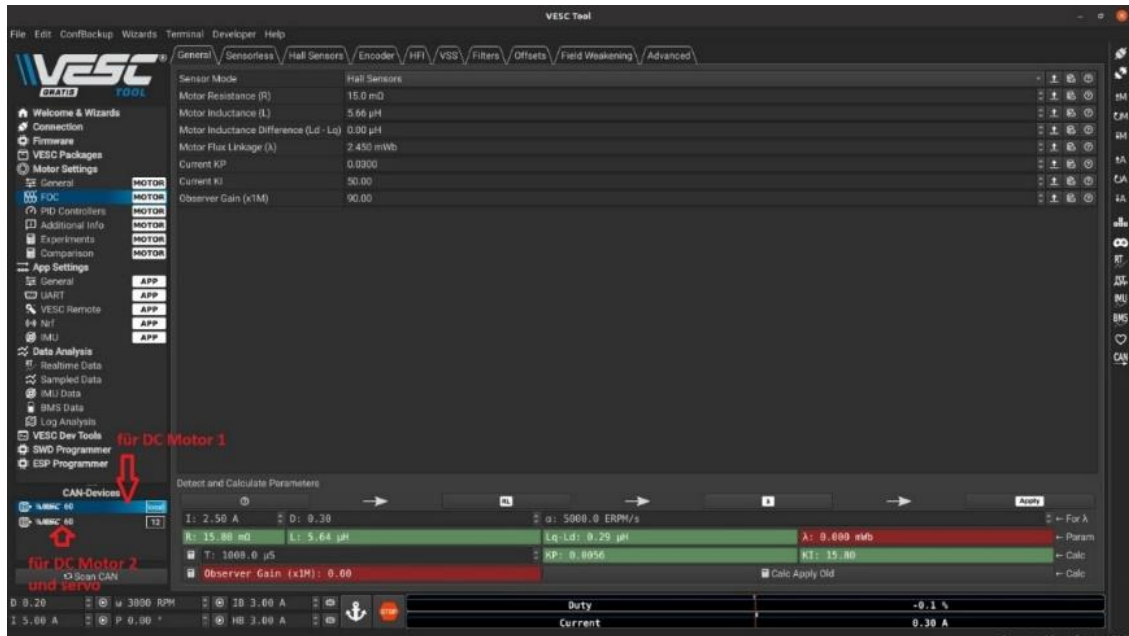


Abbildung 39: Auswahl des Motors

6. Write Konfiguration: Nach der Motorauswahl müssen wir einige Einstellungen ändern. „Enable Servo Output“ auf **True** ändern. Wenn die Parameter eingestellt haben, auf die „Write App Configuration“ klicken, um die Änderungen im VESC zu speichern.

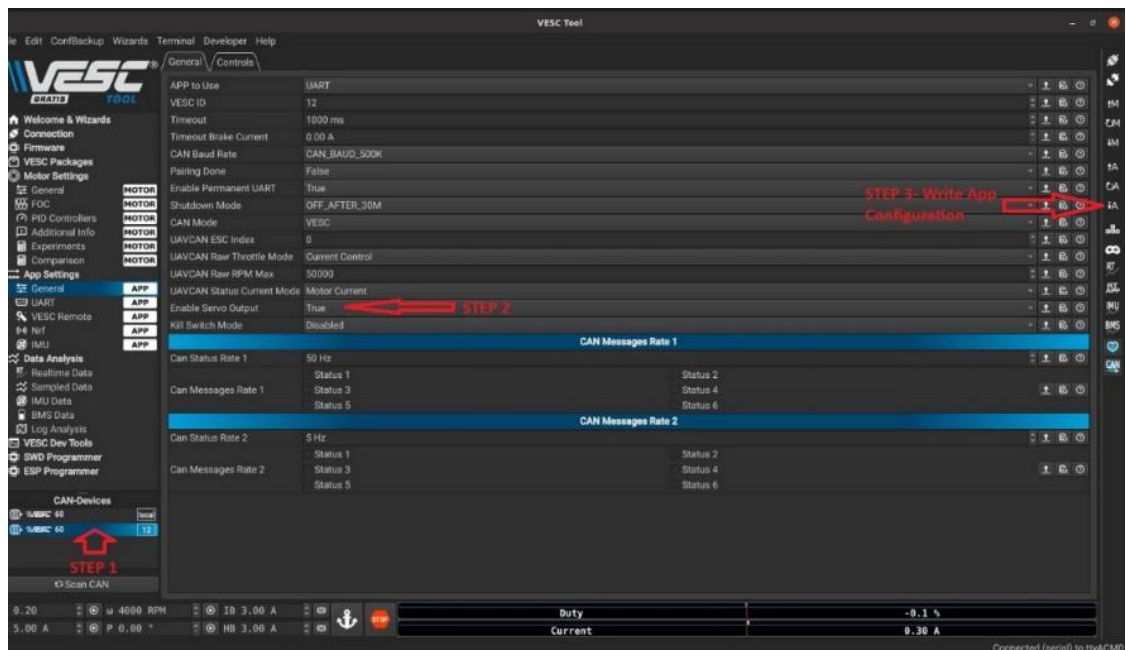


Abbildung 40: „Enable Servo Output“ aktivieren

7. Parameter ändern, um die Geschwindigkeit des Motors zu ändern: Motoren auf sichere Weise testen, um sicherzustellen, dass die neuen Parameter wie erwartet funktionieren. Und auf die Beschleunigung und die Gesamtleistung achten.

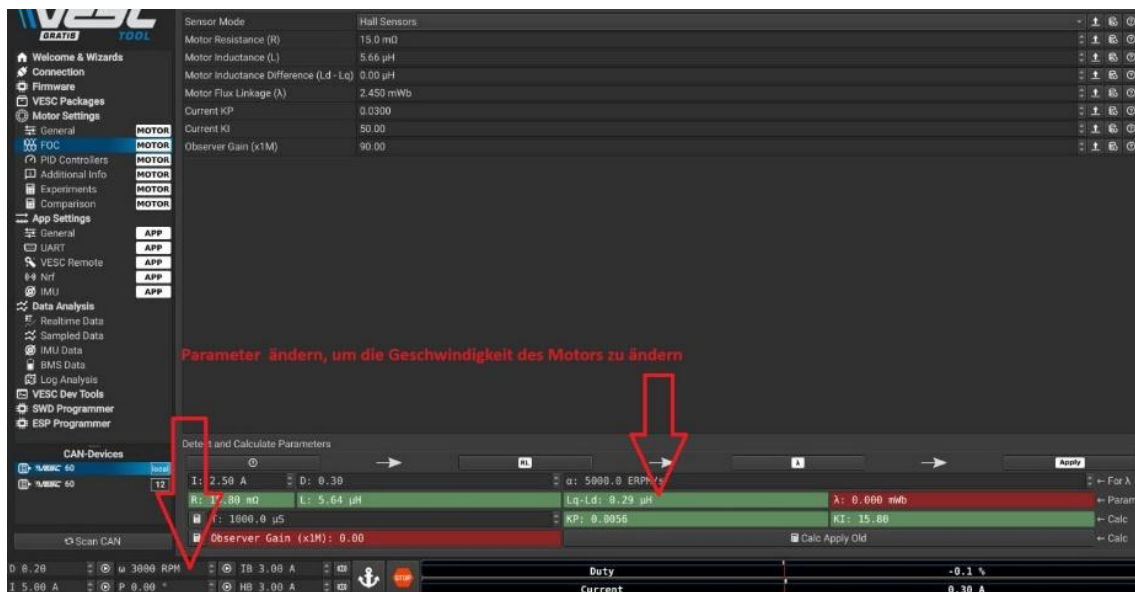


Abbildung 41: Änderung der Motordrehzahl

3.18 IMU-Sensor (Arash Barabadi)

Die MTLT305D-Serie ist eine Familie von dynamischen Neigungs- und Beschleunigungssensoren. Sie liefern dynamische Nicken und Rollen, lineare 3D-Beschleunigung und geschätzte 3D-Messdaten (Nicken und Rollen sind Bias-korrigiert, Gieren nicht).

Die folgende Abbildung zeigt das Hardware-Blockdiagramm der MTLT305D-Serie. Das Herzstück der MTLT305D-Serie ist ein robuste 6-DOF (6-Freiheitsgrad) MEMS (Micro-electromechanical Systems) -Trägheitssensor -gruppe, die allen Mitgliedern der MTLT305D-Serie. Der 6-DOF-MEMS-Trägheitssensor-Cluster umfasst drei Achsen der MEMS-Winkel. Diese Sensoren basieren auf einer robusten, felderproben bewährten Silizium-Mikrobearbeitungstechnologie.

Jeder Sensor innerhalb des Clusters wird im Rahmen des ACEINNAs automatisierter Herstellungsprozess kalibriert. Die kompensierten Sensorfehler sind Temperaturverzerrung, Skalierungsfaktor, Nichtlinearität und Ausrichtungsfehler durch einen proprietären Algorithmus aus Daten, die während der Herstellung gesammelt werden. Die Verzerrungen des Beschleunigungssensors und des Kreiselensors verschieben sich über die Temperatur (-40 °C bis +85 °C) werden mit Hilfe kalibrierter Wärmekammern und Präzisionskurventabellen kompensiert und überprüft.

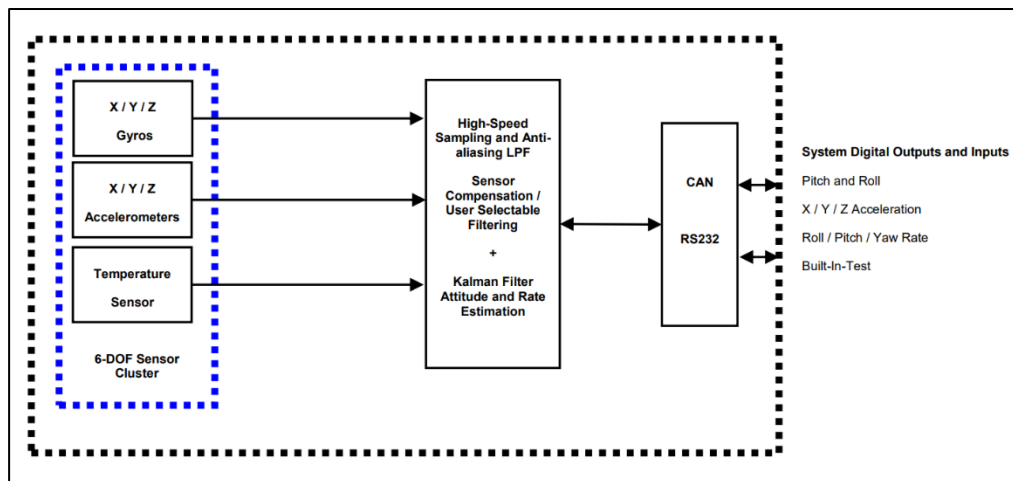


Abbildung 42: Hardware-Blockdiagramm

MTLT305D User's Manual

[Crossnet \(windows.net\)](http://crossnet.windows.net)

3.18.1 Einführung des IMU-Sensors in ROS2 (Arash Barabadi)

Hier wird die Definition des IMU-Sensors auf dem Jetson-Board betrachtet und wie sein Node im Kontext von ROS2 definiert werden soll. Um den IMU-Sensor im Betriebssystem des Roboters (ROS2) einzusetzen, sollte der Sensor als ein Node definiert werden, der das Signal an den menschlichen Fahrer auf RVIZ2 sendet.

In dem bereitgestellten Verzeichnis, im Bereich "Teleop_Jetson", es gibt ein Package mit dem Namen "mtlt305_ros", das unter der folgenden Adresse verfügbar ist: "Teleoperated_Auto/f1tenth_system/imu_neu/mtlt305_ros"

Dieses Paket enthält alle Dateien zur Erstellung und Implementierung der IMU-Software auf der ROS2-Plattform. Wie aus der folgenden Abbildung ersichtlich ist, die Python-Datei MTLT305_node.py ist ein Publisher-Node. Aber der Node empfängt zuerst die Daten vom IMU-Hardware-Sensor über den Port **"/dev/ttyTHS0"** des Jetson-Boards, verarbeitet sie und sendet sie schließlich als Publisher-Node über das Topic **"/MTLT305/imu"** als Signal, das vom RVIZ2-Node empfangen wird.

Rviz2, eine Nachbearbeitungssoftware, die Signale verarbeitet und auf dem Bildschirm anzeigt, spielt als Subscriber-Node.

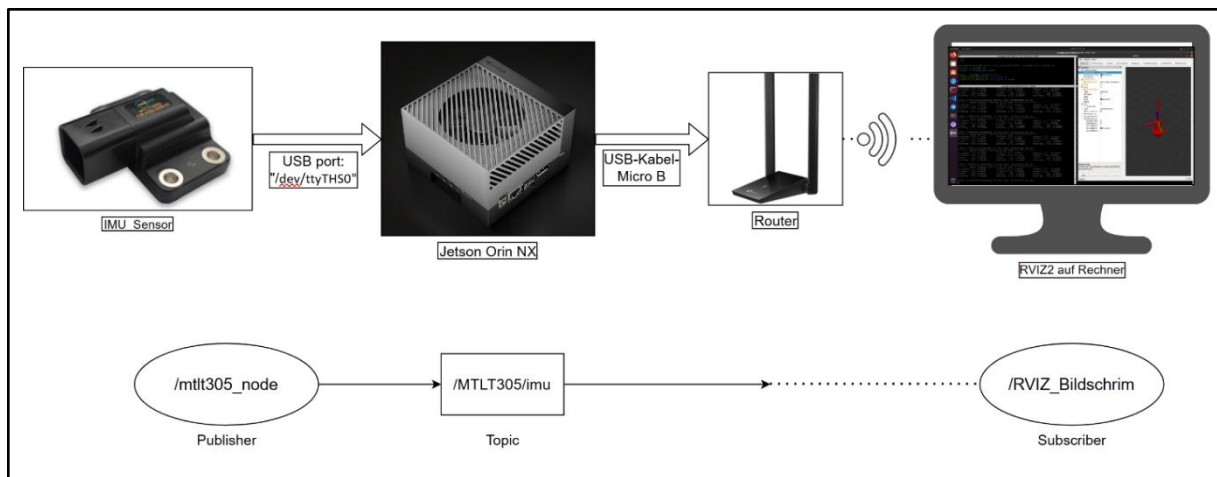


Abbildung 43: IMU-Sensor Node zu RVIZ

3.18.2 Teil 7 von ROS2-Architektur (Arash Barabadi)

Publisher	/mtlt305_node	Dieser Node liest kontinuierlich Daten vom MTLT305-IMU-Sensor, erstellt ROS-2-Imu-Nachrichten und veröffentlicht sie im Topic "/MTLT305/imu". Er überwacht auch den Verbindungsstatus und beendet sich, wenn die Verbindung unterbrochen wird.
Topic	/MTLT305/imu	Der Kanal, über den die Daten, wie z. B. Informationen über Orientierung, Winkelgeschwindigkeit und lineare Beschleunigung, übertragen werden
Subscriber	/RVIZ_Bildschirm	RVIZ 2 visualisiert IMU-Daten, die über das Thema "/MTLT305/imu" empfangen wurden, und liefert grafische Darstellungen der Orientierung, Winkelgeschwindigkeit und linearen Beschleunigung des Roboters sowie der Temperatur für Analyse und Fehlersuche.

Tabelle 9: IMU-Sensor als Publisher zu RVIZ als Subscriber über /MTLT305/imu Topic

3.18.4 Erklärung des Python-Nodes (Arash Barabadi)

Hier wird erklärt, wie der IMU Sensor Node funktioniert. Es ist in Python geschrieben. Der grüne Text sind Kommentare.

Importe: Die erforderlichen Bibliotheken und Module werden importiert. Dazu gehören systembezogene Bibliotheken (sys, os), zeitbezogene (time), datenverarbeitende (numpy, scipy), ROS 2 bezogene (rclpy, Node) und Nachrichtentyp (Imu) aus dem sensor_msgs Paket. Zusätzlich importiert es die Klasse MTLT305 aus einem benutzerdefinierten vorprogrammierten Modul.

```
import sys
import os
import time
import struct
import numpy as np
from scipy.spatial.transform import Rotation
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import Imu
from mtl305_ros.MTLT305 import MTLT305
```

Klasse von mtl305_node: Diese Klasse erbt von der Klasse Node, die von rclpy bereitgestellt wird. Sie stellt den ROS2-Node dar, der für die Veröffentlichung der IMU-Daten des MTLT305-Sensors verantwortlich ist.

```
class mtl305_node(Node):
```

Initialisierung: In der init-Methode wird der Node mit dem Namen "mtlt305_node" initialisiert. Es wird eine Instanz der Klasse MTLT305 erstellt, die über einen seriellen Anschluss mit dem Sensor verbunden wird. Sie stellt die Kommunikation mit dem Sensor her, indem sie dessen ID und Version abfragt.

```
    def __init__(self):
        super().__init__("mtlt305_node")
```

port="/dev/ttyTHS0": Dieser Parameter gibt den seriellen Anschluss an, über den die Kommunikation mit dem MTLT305-Sensor erfolgen soll. In diesem Fall wird er auf "/dev/ttyTHS0" gesetzt, was typischerweise ein serieller Anschluss auf einer Jetson-Plattform ist.

baudrate=57600: Dieser Parameter gibt die Baudrate für die serielle Kommunikation mit dem Sensor an. Die Baudrate bestimmt die Geschwindigkeit, mit der die Daten zwischen dem Node und dem Sensor übertragen werden. In diesem Fall ist sie auf 57600 Bits pro Sekunde eingestellt.

```
        self.mtl305 = MTLT305(port="/dev/ttyTHS0", baudrate=57600)

        if self.mtl305.port is None:
            self.get_logger().fatal("MTLT305 serial-port not valid")
```

```

        exit()
        return

    self.mtl305.get_packet_request(packet="ID")
    while not self.mtl305.received_id:
        time.sleep(0.05)
        self.mtl305.read_packet_response()
        pass
    self.mtl305.get_packet_request(packet="VR")
    while not self.mtl305.received_vr:
        time.sleep(0.05)
        self.mtl305.read_packet_response()
        Pass

    self.get_logger().info(f"Connected to MTL305: \nPort: \t
{self.mtl305.port.name}\nID: \t\t {self.mtl305.id}\nVersion: \t
{self.mtl305.VR}")

```

Erstellung eines Publishers: Es wird ein Publisher erstellt, um IMU-Daten im Topic "/MTLT305/imu" zu veröffentlichen. Als Nachrichtentyp wird "Imu" angegeben, und die Größe der Warteschlange wird auf 10 festgelegt.

```

self._publisher = self.create_publisher(Imu, "/MTLT305/imu", 10)

```

Timer und Watchdog: Es werden zwei Zeitgeber erstellt - _timer und _watchdog. Der _timer wird verwendet, um regelmäßig Daten vom Sensor zu lesen und sie zu veröffentlichen. Der _watchdog-Zeitgeber wird zur Überwachung des Verbindungsstatus mit dem Sensor verwendet. Wenn die Verbindung unterbrochen wird, beendet sich der Node.

```

self._timer = self.create_timer(1e-3, self.timer_callback)
self._watchdog = self.create_timer(4, self.watchdog_callback)

```

timer_callback Methode: Diese Methode wird in regelmäßigen Abständen durch den _timer aufgerufen. Sie liest IMU-Daten vom Sensor und erstellt aus den empfangenen Daten eine IMU-Nachricht. Die Nachricht enthält Orientierungs-, Winkelgeschwindigkeits- und lineare Beschleunigungsdaten.

```

def timer_callback(self):
    self.mtl305.read_packet_response()
    if self.mtl305.received_ang2:
        self._watchdog.reset()
        msg = Imu()
        msg.header.stamp = self.get_clock().now().to_msg()
        msg.header.frame_id = 'mtlt305'
        rot = Rotation.from_euler("xyz", [self.mtl305.a2.rollAngle,
self.mtl305.a2.pitchAngle, self.mtl305.a2.yawAngleTrue], degrees=True)
        [x, y, z, w] = rot.as_quat()
        msg.orientation.x = x

```

```

msg.orientation.y = y
msg.orientation.z = z
msg.orientation.w = w
msg.orientation_covariance = list(np.zeros((9,), dtype=np.float64))
msg.angular_velocity.x = self.mtl305.a2.rollRate
msg.angular_velocity.y = self.mtl305.a2.pitchRate
msg.angular_velocity.z = self.mtl305.a2.yawRate
msg.angular_velocity_covariance =
list(np.zeros((9,), dtype=np.float64))
msg.linear_acceleration.x = self.mtl305.a2.x_accel
msg.linear_acceleration.y = self.mtl305.a2.y_accel
msg.linear_acceleration.z = self.mtl305.a2.z_accel
msg.linear_acceleration_covariance =
list(np.zeros((9,), dtype=np.float64))
self._publisher.publish(msg)
self.get_logger().info("transmitted IMU-msg")

```

watchdog_callback Methode: Diese Methode wird durch den _watchdog-Timer aufgerufen. Sie protokolliert eine fatale Fehlermeldung, die anzeigt, dass die Verbindung mit dem Sensor verloren gegangen ist, und beendet den Node.

```

def watchdog_callback(self):
    self.get_logger().fatal("Connection Lost")
    exit()

```

main-Funktion: Die main-Funktion initialisiert das ROS 2-System, erstellt eine Instanz der Klasse mtl305_node und startet die ROS 2-Ereignisschleife mit rclpy.spin(). Schließlich fährt sie das ROS 2-System herunter, wenn die Ereignisschleife beendet ist.

```

def main(args=None):
    rclpy.init(args=args)
    node = mtl305_node()
    rclpy.spin(node)
    rclpy.shutdown()

```

Ausführung: Die Main-Funktion wird aufgerufen, wenn das Skript direkt ausgeführt wird (nicht als Modul importiert), wodurch der Node gestartet wird.

```

if __name__ == "__main__":
    main()

```

3.19 Launch-Datei (Arash Barabadi)

Im GitHub-Verzeichnis auf dem Jetson-Zweig ist die Datei bringup_launch.py über die folgende Adresse zugänglich.

“Teleoperated_Auto/f1tenth_system/f1tenth_stack/launch/bringup_launch.py”

Diese Datei wurde folgendermaßen geschrieben;

Importe und Funktionsdefinition:

```
from launch import LaunchDescription
from launch_ros.actions import Node
from launch.substitutions import Command
from launch.substitutions import LaunchConfiguration
from launch.actions import DeclareLaunchArgument
from launch.actions import IncludeLaunchDescription
from launch.actions import ExecuteProcess
from launch_xml.launch_description_sources import XMLLaunchDescriptionSource
from ament_index_python.packages import get_package_share_directory
import os
```

Erklärung: In diesem Schritt werden die erforderlichen Python-Module und -Funktionen importiert. Diese Module und Funktionen sind erforderlich, um die Startdatei zu konfigurieren.

Festlegen der Konfigurationspfade:

```
def generate_launch_description():
    joy_teleop_config = os.path.join(
        get_package_share_directory('f1tenth_stack'),
        'config',
        'joy_teleop.yaml'
    )
    vesc_config = os.path.join(
        get_package_share_directory('f1tenth_stack'),
        'config',
        'vesc.yaml'
    )
    sensors_config = os.path.join(
        get_package_share_directory('f1tenth_stack'),
        'config',
        'sensors.yaml'
    )
    mux_config = os.path.join(
        get_package_share_directory('f1tenth_stack'),
        'config',
        'mux.yaml'
    )
```

Erklärung: Pfade zu den Konfigurationsdateien für die Joystick-Steuerung, VESC, Sensoren und den MUX werden festgelegt. Diese Pfade werden mithilfe der Funktion `get_package_share_directory` ermittelt, um das entsprechende Konfigurationsverzeichnis des Pakets zu finden.

Deklaration der Startargumente:

```

joy_la = DeclareLaunchArgument(
    'joy_config',
    default_value=joy_teleop_config,
    description='Descriptions for joy and joy_teleop configs')
vesc_la = DeclareLaunchArgument(
    'vesc_config',
    default_value=vesc_config,
    description='Descriptions for vesc configs')
sensors_la = DeclareLaunchArgument(
    'sensors_config',
    default_value=sensors_config,
    description='Descriptions for sensor configs')
mux_la = DeclareLaunchArgument(
    'mux_config',
    default_value=mux_config,
    description='Descriptions for ackermann mux configs')

```

Erklärung: Startargumente werden für die Konfigurationen deklariert, einschließlich Standardwerten und Beschreibungen. Diese Argumente werden später verwendet, um Parameter für die Knoten festzulegen.

Initialisierung der Startdatei:

```
ld = LaunchDescription([joy_la, vesc_la, sensors_la, mux_la])
```

Erklärung: Eine Instanz von **LaunchDescription** wird erstellt, um die gesamte Startdatei zu repräsentieren. Die Deklarationen der Startargumente werden dieser Instanz hinzugefügt.

Konfiguration der Knoten:

```

joy_node = Node(
    package='joy',
    executable='joy_node',
    name='joy',
    parameters=[LaunchConfiguration('joy_config')]
)
joy_teleop_node = Node(
    package='joy_teleop',
    executable='joy_teleop',
    name='joy_teleop',
    parameters=[LaunchConfiguration('joy_config')]
)
ackermann_to_vesc_node = Node(
    package='vesc_ackermann',
    executable='ackermann_to_vesc_node',
    name='ackermann_to_vesc_node',
    parameters=[LaunchConfiguration('vesc_config')]
)

```

```

vesc_to_odom_node = Node(
    package='vesc_ackermann',
    executable='vesc_to_odom_node',
    name='vesc_to_odom_node',
    parameters=[LaunchConfiguration('vesc_config')]

```

Weitere Knoten werden ähnlich konfiguriert...

Erklärung: Verschiedene Knoten werden konfiguriert, einschließlich Joystick-Knoten, Joystick-Teleoperationsknoten, VESC-zu-Odometrie-Knoten usw. Jeder Knoten wird mit seinem Paketnamen, ausführbaren Namen, Knotennamen und Parametern spezifiziert, die aus den Startargumenten gelesen werden.

Zusätzliche Aktionen

```

record_prc = ExecuteProcess(
    cmd = ["ros2", "bag", "record", "-a"],
    output="screen"
)

```

Erklärung: Zusätzliche Aktionen werden definiert, wie das Starten eines Prozesses zum Aufzeichnen von Daten mit 'ros2 bag record -a'. Diese Aktion wird derzeit nicht zur Startbeschreibung hinzugefügt, kann aber später bei Bedarf hinzugefügt werden.

Abschluss

Aktionen und Knoten werden der Startbeschreibung hinzugefügt

```

ld.add_action(ackermann_to_vesc_node)
ld.add_action(vesc_to_odom_node)
ld.add_action(vesc_driver_node)
ld.add_action(camera_node)
ld.add_action(mtl305_node)

```

Erklärung:

Die konfigurierten Aktionen und Knoten werden der 'LaunchDescription' hinzugefügt.

Dies schließt die Konfiguration der Startdatei ab, und die 'LaunchDescription' wird zurückgegeben, um die gesamte Konfiguration der Startdatei abzuschließen und die Ausführung zu starten.

4 Abbildungsverzeichnis

Abbildung 1: Gruppenverteilung des Projekts	6
Abbildung 2: Strukturbild des Fahrzeugs	6
Abbildung 3: Motor TT5692 Top Tuning Brushless Motor	8
Abbildung 4: Umrichter FlipSky FSESC 6.9	9
Abbildung 5: Pinverteilung des Umrichters.....	9
Abbildung 6: Batterie TopTuning TT1530/650	10
Abbildung 7: NVIDIA® Jetson Orin Nano™ Developer Kit mit nummerierten Pins (Legende darunter).....	11
Abbildung 8: Pinverteilung Jetson.....	12
Abbildung 9: Servo K-Power DM4000	13
Abbildung 10: Kamera Intel® RealSense™ Depth Camera D435i	13
Abbildung 11: IMU ACEINNA MTLT305D	14
Abbildung 12: Pinverteilung IMU	14
Abbildung 13: DC-/DC-Wandler Bauer Electronics DC-DC 8-40V-12V - 20A.....	15
Abbildung 14: Lüfter Jiabofan JD6025B8HH.....	15
Abbildung 15: Positionsgber AMS Osram AS5X47P-TS_EK_MB.....	16
Abbildung 16: Antenne TP-Link AX1800.....	16
Abbildung 17: Sicherungskennlinien	19
Abbildung 18: Schaltplan des Modellfahrzeugs	20
Abbildung 19: Aufzweigung eines Kabelbaums	21
Abbildung 20: Kabelbaum eines PKW	21
Abbildung 21: Vogelperspektive der Grundplatte mit freien Flächen	23
Abbildung 22: Vogelperspektive der Unterseite der oberen Platte mit freien Flächen	23
Abbildung 23: Skizze der Grundplatte mit den vorläufigen Leitungs- und Kabelbaumverläufen.....	25
Abbildung 24: Skizze der oberen Platte (Unterseite) mit den vorläufigen Leitungs- und Kabelbaumverläufen	26
Abbildung 25: Diagramm mit der thermischen Belastbarkeit von elektrischen Leitungen	29
Abbildung 26: AMP-Stecker und -Buchse.....	32
Abbildung 27: EC8-Stecker	33
Abbildung 28: Ausschnitt des CAD-Modells im Bereich des Servos.....	34
Abbildung 29: Ausschnitt des CAD-Modells im Bereich des Motors.....	34
Abbildung 30: 2D-Skizze des Fahrzeugs mit farblich eingezeichneten Steckern und Leitungen	35
Abbildung 31: Die ROS2_Architektur	42
Abbildung 32: Setup von Mesh Router in der Anwendung Omada.....	48
Abbildung 33: Einstellung der Netzwerkverbindung im Ubuntu	49
Abbildung 34: Client List von Mesh Router	50
Abbildung 35: Bashrc Automatisierung Zeilen	54
Abbildung 36: Versuchsaufbau der Messung von der Verzögerung auf ca. 10 m	55
Abbildung 37: Komprimiertes Bild aus der Kamera	57
Abbildung 38: Das Originalbild aus der Kamera	57
Abbildung 39: Auswahl des Motors	60
Abbildung 40: „Enable Servo Output“ aktivieren.....	60
Abbildung 41: Änderung der Motordrehzahl	61
Abbildung 42: Hardware-Blockdiagramm	62
Abbildung 43: IMU-Sensor Node zu RVIZ.....	63

5 Tabellenverzeichnis

Tabelle 1: Technische Daten der Komponenten	17
Tabelle 2: Parameter der einzelnen Leitungen	27
Tabelle 3: Die Datenverbindung zwischen dem Joystick und dem vordefinierten Teleoperationsknoten	42
Tabelle 4: Die Datenverbindung zwischen Teleoperationsknoten und Ackermann-Modell	43
Tabelle 5: Die Datenverbindung zwischen Ackermann-Modell und VESC-bezogenem Knoten	44
Tabelle 6: Die Datenverbindung zwischen dem VESC-bezogenen Knoten und dem VESC-Treiberknoten	45
Tabelle 7: Die Datenverbindung zwischen VESC-Treiberknoten und VESC_to_odom-Knoten.....	46
Tabelle 8: Messdaten der Verzögerung	55
Tabelle 9: IMU-Sensor als Publisher zu RVIZ als Subscriber über /MTLT305/imu Topic	63

6 Literaturverzeichnis

- [1] RC-Car-Shop Hobbythek Roswitha Sturm e.K., „TT5692 Top Tuning Brushless Motor 1090 kV,“ o.D.. [Online]. Available: https://rc-car-online.de/de/products/tt_5692-tt5692-top-tuning-brushless-motor-1090-kv.html. [Zugriff am 01 Februar 2024].
- [2] Alibaba Group, „Elektro Speed Controller FLIPSKY FSESC 6,9 Plus Basis auf vesc 6,6 Mit Power-Taste für Elektrische Skateboard Roller Ebike Roboter,“ o.D.. [Online]. Available: <https://de.aliexpress.com/item/1005005042795947.html>. [Zugriff am 01 Februar 2024].
- [3] „Flipsky-fsesc6-9,“ [Online]. Available: <https://flipsky.net/products/flipsky-fsesc6-9-100a-base-on-vesc6-6-with-aluminum-anodized-heat-sink>.
- [4] RC-Car-Shop Hobbythek Roswitha Sturm e.K., „TT1530/650 Top Tuning 5000 mAh LiPo Akku 6S, 22,2V Premiumqualität 60C,“ o.D.. [Online]. Available: https://rc-car-online.de/de/products/tt_tt1530_650-tt1530-650-top-tuning-5000-mah-lipo-akku-6s-22-2v-premiumqualitaet-60c.html. [Zugriff am 01 Februar 2024].
- [5] „Jetson Orin Nano Developer Kit Getting Started Guide,“ [Online]. Available: <https://developer.nvidia.com/embedded/learn/get-started-jetson-orin-nano-devkit#intro>.
- [6] „GPIO Support on Jetson Nano Developer Kit,“ [Online]. Available: <https://maker.pro/nvidia-jetson/tutorial/how-to-use-gpio-pins-on-jetson-nano-developer-kit>.
- [7] RC-Car-Shop Hobbythek Roswitha Sturm e.K., „K-Power DM4000 Servo 54kg-cm/0,11 s/6 Volt bis 8,4 Volt,“ [Online]. Available: https://rc-car-online.de/de/products/kpower_dm4000-k-power-dm4000-servo-54kg-cm-0-11-s-6-volt-bis-8-4-volt.html. [Zugriff am 02 Februar 2024].
- [8] Intel Corporation, „Intel Realsense Depth Camera D435i,“ 02 Februar 2019. [Online]. Available: <https://www.intelrealsense.com/depth-camera-d435i/>. [Zugriff am 02 Februar 2024].
- [9] Aceinna, „MTLT305 Series Dynamic Tilt Sensor / IMU Module,“ [Online]. Available: https://www.mouser.de/datasheet/2/940/6020_3305_01_F__MTLT305x-2939142.pdf. [Zugriff am 01 02 2024].
- [10] Bauer United, „DC DC 8-40V zu 12V 20A,“ o.D.. [Online]. Available: <https://bauer-united.com/products/dc-dc-8-40v-zu-12v-20a-spannungswandler>. [Zugriff am 01 Februar 2024].
- [11] Jiabofan, o.D.. [Online]. Available: http://www.jiabofan.com/products_content-4823982.html. [Zugriff am 01 Februar 2024].
- [12] RS Components GmbH, „ams OSRAM AS5X47P AS5X47P-TS_EK_MB Entwicklungskit, Positionssensor für AS5X47P,“ o.D.. [Online]. Available: <https://de.rs-online.com/web/p/entwicklungstools-sensorik/2329700>. [Zugriff am 01 Februar 2024].
- [13] Amazon.com, Inc., „TP-Link Archer TX20U Plus WLAN Stick Für PC, WiFi 6 AX1800 Dual Band, USB 3.0, MU-MIMO, Antennen mit hoher Verstärkung, WPA3-Verschlüsselung, Kompatibel mit

- Windows 11/10, Schwarz, único," o.D.. [Online]. Available: https://www.amazon.de/TP-Link-Archer-TX20U-Plus-WPA3-Verschl%C3%BCsselung/dp/B09X3FTL7F/ref=asc_df_B09X3FTL7F/?tag=googshopde-21&linkCode=df0&hvadid=604587227256&hvpos=&hvnetw=g&hvrnd=4493445290583695010&hvpone=&hvpstwo=&hvmqmt=&hvdev=c&hvdvcmdl=&hvlocint=. [Zugriff am 01 Februar 2024].
- [14] Pacific Engineering Corporation, „Automotive Fuses,“ 09 Mai 2019. [Online]. Available: <https://www.pecj.co.jp/en/fuse/slow/sbfw-k.html>. [Zugriff am 01 Februar 2023].
- [15] H.-G. Prof. Dr. Schweiger, „Elektromechanik,“ Ingolstadt, 2023.
- [16] H. Schmidt, „Das Ende des Kabelbaums: Die elektronischen Komponenten moderner Autos werden zunehmend über kabellose Netzwerke gesteuert,“ 13 Juli 2022. [Online]. Available: <https://www.nzz.ch/mobilitaet/kabelbaeume-fuer-verbrenner-sterben-aus-ld.1693239>. [Zugriff am 01 Februar 2024].
- [17] Fritz Diel GmbH & Co. KG, „Strombelastbarkeit für Litzen in Luft,“ Dortmund, 2009.
- [18] Conrad Electronic SE, „Kabelquerschnitt berechnen » Kabel & Leitungen richtig dimensionieren,“ 16 November 2022. [Online]. Available: <https://www.conrad.de/de/ratgeber/handwerk/kabelquerschnitt-berechnen.html>. [Zugriff am 01 Februar 2024].
- [19] Wikimedia Foundation, „Elektrische Leitfähigkeit,“ 11 September 2023. [Online]. Available: https://www.chemie.de/lexikon/Elektrische_Leitf%C3%A4higkeit.html. [Zugriff am 01 Februar 2024].
- [20] Incore Cables B.V, „Die Vorteile von Aluminiumkabeln,“ 25 Juni 2015. [Online]. Available: <https://www.incore-cables.com/die-vorteile-von-aluminiumkabeln/?lang=de>. [Zugriff am 01 Februar 2024].
- [21] SAB Bröckskes GmbH & Co. KG, „Eigenschaften von Isolier- und Mantelwerkstoffen,“ o.D. 2024. [Online]. Available: <https://www.sab-kabel.de/kabel-konfektion-temperaturmesstechnik/technische-daten/kabel-leitungen/eigenschaften-von-isolier-und-mantel-werkstoffen.html>. [Zugriff am 01 Februar 2024].
- [22] Zerocom GmbH, „Silikonkabel hoch flexibel in AWG 4 6 8 10 12 14 16 18 20 22 Rot Schwarz Kupfer RC,“ 17 Januar 2024. [Online]. Available: <https://www.rcgraf.com/produkt/silikonkabel-hoch-flexibel-in-awg-4-6-8-10-12-14-16-18-20-22-rot-schwarz-kupfer-rc/>. [Zugriff am 01 Februar 2024].
- [23] Recambo e.K., „AMP Superseal Stecker Set 2-polig mit Kabel 1,5mm² Auto KZF Boot LKW Roller Motorrad,“ o.D.. [Online]. Available: <https://www.recambo.de/AMP-Superseal-Stecker-Set-2-polig-mit-Kabel-15mm-Auto-KZF-Boot-LKW-Roller-Motorrad>. [Zugriff am 01 Februar 2024].
- [24] Amazon Europe Core S.à r.l., „EC8-Batterieanschluss, EC8-Stecker, weiblich, gute elektrische Leitfähigkeit für Elektrofahrzeuge,“ o.D.. [Online]. Available: <https://www.amazon.de/EC8->

Batterieanschluss-EC8-Stecker-elektrische-Leitfähigkeit-Elektrofahrzeuge/dp/B0BJ9M2JHM. [Zugriff am 01 Februar 2024].

- [25] S. Macenski, T. Foote, B. Gerkey, C. Lalancette und W. & Woodall, „Robot Operating System 2: Design, architecture, and uses in the wild,“ *Science Robotics*, Bd. 66, 2022.
- [26] Open Robotics, „ROS2 Dokumentation: Foxy: Installation,“ [Online]. Available: <https://docs.ros.org/en/foxy/Installation.html>. [Zugriff am 15 November 2023].
- [27] B. Iwfinger, „GitHub: rtl8852au,“ [Online]. Available: <https://github.com/Iwfinger/rtl8852au>. [Zugriff am 6 Dezember 2023].
- [28] IntelRealSense, „GitHub: librealsense,“ [Online]. Available: https://github.com/IntelRealSense/librealsense/blob/development/doc/distribution_linux.md. [Zugriff am 06 12 2023].
- [29] IntelRealSense, „GitHub: realsense-ros,“ [Online]. Available: <https://github.com/IntelRealSense/realsense-ros/tree/ros2-development#installation>. [Zugriff am 6 12 2023].
- [30] IntelRealSense, „IntelRealSense: ROS2,“ [Online]. Available: <https://dev.intelrealsense.com/docs/ros2-wrapper#compression-packages>. [Zugriff am 13 12 2023].
- [31] K. Atsushi, „GitHub: ros-g29-force-feedback,“ [Online]. Available: <https://github.com/kuriatsu/ros-g29-force-feedback>. [Zugriff am 16 11 2023].