

M 2.7 HiL-Verfahren (HILV)

C. Stubbemann, M. Meiners

2024-11-08

Inhaltsverzeichnis

Unsyllabus	1
1. Einleitung	3
1.1. Systems Engineering	3
1.1.1. Requirements Based Engineering	3
1.1.2. Model-Based Systems Engineering	3
1.1.3. In-the-Loop Methodik (MBAT)	3
I. Projekte	5
2. Gimbal	7
2.1. Theoretische Grundlagen	7
2.2. Model-Based Systems Engineering	8
2.3. Cameo - No Magic	9
2.4. 3D Druck des Gimbal	9
2.4.1. CAD Modellierung	12
2.4.2. Erster Druck	12
2.4.3. Zweiter Druck	14
2.4.4. Finale Anpassungen	14
2.5. Zusammengebauter Gimbal	14
2.6. Elektronik des Gimbal	14
2.6.1. Arduino Nano	16
2.6.2. Arduino Code Beschreibung	16
2.6.3. Aufgetretene Probleme und Lösungen in Nano	17
2.6.4. ESP8266 NodeMCU	19
2.6.5. Der Funktionierte Teil	20
2.6.6. ESP8266 Steuerung des Servos	20
2.6.7. ESP8266 Daten von der MPU6050 lesen	21
2.6.8. Der nicht Funktionierte Teil	22
2.6.9. Bibliotheks-Kompatibilität	22
2.6.10. Entwicklung des Interruppt PINs	22
2.6.11. Verbesserung der Schaltung:	23
2.6.12. Die Lösungen waren wie folgt:	24
2.7. Überprüfung der Spannungskonstanz und Stabilität eines Buck-Konverters	37
2.7.1. Testkonzept	37
2.7.2. Zustand des Geräts	37

Inhaltsverzeichnis

2.7.3.	Durchführung	37
2.7.4.	Testanforderungen	38
2.7.5.	Zustand des Konzepts	38
2.7.6.	Ergebnisvergleich mit Anforderung	38
2.8.	Überprüfung der Ausgangsspannungserzeugung des Buck-Konverters	38
2.8.1.	Testkonzept	39
2.8.2.	Zustand des Geräts	39
2.8.3.	Testanforderung	39
2.8.4.	Ergebnisvergleich mit Anforderung	39
II.	Anhang	41
3.	No Magic MBSE Software	43

Abbildungsverzeichnis

2.1. Schaltplan	15
---------------------------	----

Tabellenverzeichnis

Unsyllabus

Name	Beschreibung
Kursus	M 2.7 HiL-Verfahren (HILV)
Semester	WiSe2024/25
Dozent	B.Eng. C. Stubbemann (SCOPE), Prof. Dr.-Ing. M. Meiners (HSB)
Seminar	Fr., 13:30 Uhr - 15:00 Uhr
Raum	E 509

1. Einleitung

1.1. Systems Engineering

- Validierung oder Validation (von lateinisch validus “kräftig, wirksam, fest”)
 - Validierung (Chipentwurf), Vergleich des Chipdesigns mit der vorgegebenen Spezifikation
 - Validierung (Informatik), Nachweisführung, dass ein System die Praxisanforderungen erfüllt
- Verifizierung oder Verifikation (von lateinisch veritas ‚Wahrheit‘ und facere ‚machen‘) ist der Nachweis, dass ein vermuteter oder behaupteter Sachverhalt wahr ist

1.1.1. Requirements Based Engineering

1.1.2. Model-Based Systems Engineering

1.1.3. In-the-Loop Methodik (MBAT)

Teil I.

Projekte

2. Gimbal

In diesem Projekt geht es um die Konzeption und den Bau eines Gimbal, einer selbst stabilisierenden Plattform, die mithilfe von Sensoren und Motoren stabilisiert wird, um eine präzise und ruhige Kameraaufnahme zu ermöglichen. Im Gegensatz zu anderen Gimbalen, die oft als fertige Produkte erhältlich sind, werden wir uns in diesem Projekt nicht nur auf die Montage der Bauteile beschränken, sondern uns auch mit der Programmierung und Entwicklung der elektronischen Komponenten befassen. Dazu kommen Aspekte der 3D-Modellierung mit CAD, die es ermöglichen ein 3D-Modell zu erstellen, um die einzelnen Bauteile im Hause zu drucken. Dadurch werden wir in der Lage sein, den Gimbal nach unseren individuellen Anforderungen und Bedürfnissen zu gestalten und anzupassen. Das Projekt basiert auf einem vorherigen Projekt von [HowToMechatronics](#), das Ziel ist es auf Grundlage dieses Projektes den Gimbal weiterzuentwickeln und eigene Erfahrungen zu sammeln.

2.1. Theoretische Grundlagen

Für die Theorie des Gimbal-Projekts werden wir im Folgenden einen grundlegenden Leitfaden zur Entwicklung der Theorie festlegen.

- **Verständnis des Gimbalkonzepts** Zunächst sollte jeder Projektbeteiligte das Konzept und die Funktionsweise eines Gimbal nachvollziehen und verstehen. Was genau macht die Vorrichtung? Welche Bestandteile hat das Gimbal? Wie viele Achsen hat das Gimbal?
- **Festlegung des Ziels** Das Ziel des Projekt muss klar definiert sein. Was soll am Ende dabei rauskommen? Was für ein Anwendungsszenario ist im Sinn? Somit kann der Umfang des Projekts festgelegt werden.
- **Identifizierung der Verwendeten Hardware** Es muss eine Auswahl für jede Hardwarekomponente getroffen werden. Welche Hardwarekomponente bietet welche Vorteile? z.B. ESP im Vergleich zu Arduino. (Hier befinden wir uns Aktuell)
- **Entwurf der Steuereinheit** Entwickelt einen Steuerungsalgorithmus, der die Bewegungen der Motoren basierend auf den Sensordaten steuert.
- **Implementierung der HIL-Simulation** HIL ermöglicht uns die Simulation und Überprüfung des Steuerungsalgoritmus in einer realistischen Umgebung, bevor wir ihn auf der tatsächlichen Hardware testet.

2. Gimbal

- **Durchführung von Tests und Optmierung** Es sollten umfangreiche Tests durchgeführt werden, um die Leistung eures Gimbal-Systems zu bewerten. Außerdem sollte Überprüft werden ob das Gimbal die Plattform stabil hält, unerwünschte Bewegungen minimiert und auf Benutzereingaben reagiert. Falls nötig sollte der Steueralgorithmus basierend auf den Testergebnissen optimiert werden.
- **Dokumentation und Präsentation** Zuletzt werden die Ergebnisse dokumentiert, zusammengefasst und präsentiert.

2.2. Model-Based Systems Engineering

Das modellbasierte System-Engineering (MBSE) ist eine formalisierte Methodik, die zur Unterstützung der Anforderungen, des Entwurfs, der Analyse, der Verifizierung und der Validierung im Zusammenhang mit der Entwicklung komplexer Systeme eingesetzt wird. Im Gegensatz zum dokumentenzentrierten Engineering stellt MBSE Modelle in den Mittelpunkt der Systementwicklung.

MBSE bringt drei Konzepte zusammen: Modell, Systemdenken und Systemtechnik:

Ein Modell: ist eine vereinfachte Version von etwas - eine grafische, mathematische oder physikalische Darstellung, die die Realität abstrahiert, um eine gewisse Komplexität zu eliminieren. Diese Definition impliziert Formalität oder Regeln bei der Vereinfachung, Darstellung oder Abstraktion. Um ein System zu modellieren, muss ein Systemarchitekt das System mit weniger Details darstellen, so dass seine Struktur und sein Verhalten erkennbar und seine Komplexität beherrschbar ist. Mit anderen Worten: Modelle sollten das System ausreichend darstellen, und das System sollte die Modelle bestätigen.

Systemdenken: ist eine Art, ein zu betrachtendes System nicht als autarke Einheit, sondern als Teil eines größeren Systems zu sehen. Systemdenken ist nicht dasselbe wie das systematische Befolgen guter Pläne, das Sammeln von Statistiken oder methodisches Vorgehen. Der Systemingenieur beobachtet das System aus der Ferne, erforscht seine Grenzen, seinen Kontext und seinen Lebenszyklus, notiert sein Verhalten und identifiziert Muster. Diese Methode kann dem Ingenieur helfen, Probleme zu erkennen (z. B. fehlende Interaktion, ein fehlender Schritt in einem Prozess, doppelter Aufwand, verpasste Möglichkeiten zur Automatisierung) und die Komplexität eines Systems zu bewältigen. Obwohl Systemingenieure das System zu Beginn zerlegen und analysieren müssen - Teile identifizieren und Verbindungen zwischen ihnen beschreiben -, fügen sie die Teile später wieder zu einem kohärenten Ganzen zusammen. Teile sind nicht nur mit anderen Teilen verbunden, sie sind voneinander abhängig, um richtig zu funktionieren. Beim Systemdenken wird diese Vernetzung betont. Das Verhalten des Systems ergibt sich aus den Aktivitäten der einzelnen Teile des Systems. Wenn der Systemtechniker die Verbindungen des Systems beobachtet, erkennt er Rückkopplungsschleifen und Kausalitätsmuster, die auf den ersten Blick vielleicht nicht sichtbar sind. Systemdenken kann dazu beitragen, Probleme deutlicher zu erkennen und leichter zu identifizieren, das System ins Gleichgewicht zu bringen und die Komplexität des Systems zu bewältigen.

Systems Engineering ist ein transdisziplinärer und integrativer Ansatz, der die erfolgreiche Realisierung, Nutzung und Ausmusterung von technischen Systemen unter Verwendung von Systemprinzipien und -konzepten sowie wissenschaftlichen, technologischen und Managementmethoden ermöglicht. Es vereint eine Reihe von Techniken, um sicherzustellen, dass alle Anforderungen durch das entworfene

2.3. Cameo - No Magic

System erfüllt werden. Es konzentriert sich auf die Architektur, Implementierung, Integration, Analyse und Verwaltung eines Systems während seines Lebenszyklus. Dabei werden auch Software, Hardware, Personal, Prozesse und verfahrenstechnische Aspekte des Systems berücksichtigt.

Wenn MBSE richtig durchgeführt wird, führt dies zu einer allgemeinen Verringerung der Entwicklungsrisiken.

2.3. Cameo - No Magic

Das Hauptmodell des SE-Demonstrators besteht aus drei Hauptpaketen, die in verschiedene Unterpakete unterteilt werden können. Diese drei Hauptpakete sind die “Functional Architecture” (Funktionale Architektur), die “Mission & Stakeholder Analysis” (Analyse der Mission und Interessenvertreter) und die “System Architecture” (Systemarchitektur).

Die Functionale Architecture lässt sich in zwei Unterpakete unterteilen, nämlich die Funktionale Analyse und die Systemfunktionen. Die Funktionale Analyse betrachtet alle erforderlichen funktionalen Schritte, um eine normale Operation mit dem Gimbal auszuführen, wie zum Beispiel die Anpassung der Bewegung, das Setzen eines Stabilisierungsmodus und die Überprüfung beim Start des Gimbals, ob dieser ordnungsgemäß versorgt wird.

Die Systemfunktionen lassen sich in High-Level-Verhalten und Sub-Level-Verhalten aufteilen. Das High-Level-Verhalten umfasst die normale Operation des Gimbals, bei der er wie in der Funktionalen Architektur beschrieben den Stabilisierungsmodus auswählt und anhand der Gimbal-Bewegung eine Anpassung vornimmt, um die Plattform entgegen der Bewegung zu stabilisieren. Ein weiteres High-Level-Verhalten ist der Startup Check, der auch in der Funktionalen Analyse beschrieben wird.

Das Sub-Level-Verhalten spiegelt die einzelnen Schritte des High-Level-Verhaltens wider. Im Falle der normalen Operation gibt es drei Sub-Level-Verhalten. Eines davon ist das Setzen des Stabilisierungsmodus und die erforderlichen Schritte, die beim Auswahl des Stabilisierungsmodus eingeleitet werden. Ein anderes Sub-Level-Verhalten ist beispielsweise die Erfassung der Gimbal-Bewegung und die erforderlichen Schritte dafür. Das letzte Sub-Level-Verhalten ist die eigentliche Stabilisierung, also die Kompensation der Eingangsbewegung und die darauf basierende Anpassung der Position.

Beim Startup Check werden vier Sub-Level-Verhalten dargestellt. In einem Verhalten wird der Batteriestatus überprüft, in einem anderen wird überprüft, ob eine externe Spannungsquelle angeschlossen ist. Für beide Fälle gibt es ein Sub-Level-Verhalten, das den Gimbal mit Strom versorgt, je nachdem, welcher der beiden Fälle vorliegt.

2.4. 3D Druck des Gimbal

Zur Erstellung der selbst stabilisierenden Platform, werden die verschiedenen Bauteile des Gimbals mittels 3D Druck erstellt. Diese umfassen das Gehäuse des Microcontrollers, die Platform für die Kamera, sowie die Halter für die Servo-Motoren.

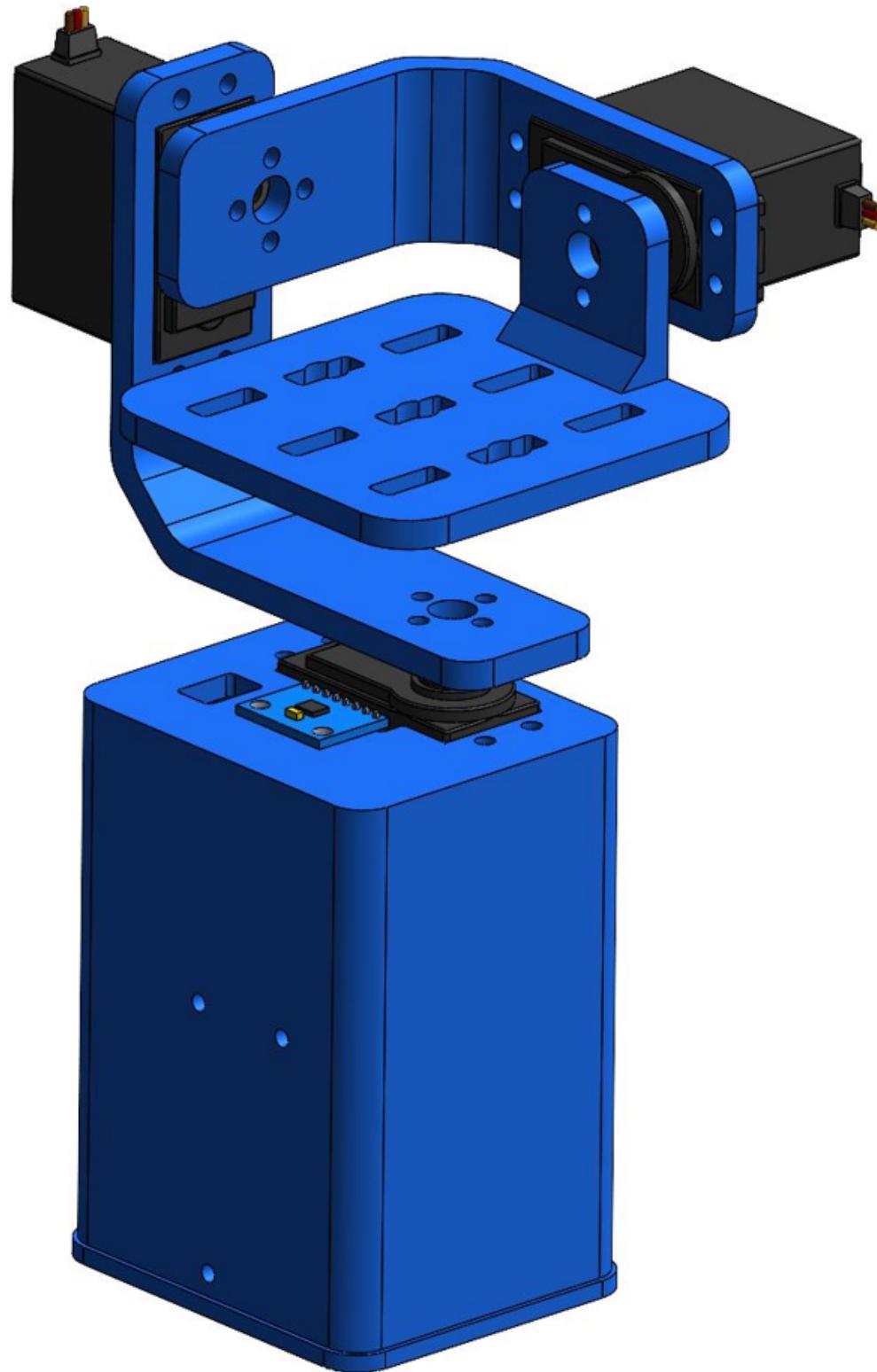
2. Gimbal

Um einen erfolgreichen 3D-Druck zu gewährleisten, werden die hierfür vorgesehenen 3D-Druckdateien in fünf Grundstrukturen unterteilt. Die Aufteilung basiert auf der Kameraplatzform, den drei Servomotoren - die jeweils eine Achse der Stabilisierung steuern - und dem Verschlussdeckel für das Gehäuse des Microcontrollers.

Dadurch erhält man eine STL-Datei als erste Grundstruktur (*Base 01*) für den Servomotor, der die Yaw-Drehung steuert, also die Drehung um die z-Achse als Referenzsystem. Eine weitere Grundstruktur (*Base 02*) ist für den Servomotor, der die Roll-Drehung steuert, was sich auf die x-Achse bezieht. Die dritte Grundstruktur (*Base 03*) ist für den Pitch-Servomotor, der eine Anpassung in Bezug auf die y-Achse vornimmt. Es ist wichtig zu beachten, dass die Grundstruktur für die Yaw-Anpassung auch als Gehäuse für den Microcontroller dient.

Die folgende Abbildung zeigt die zusammengesetzte Grundstruktur des Gimbal, einschließlich der Servomotoren.

2.4. 3D Druck des Gimbal



#fig-gimball1

2. Gimbal

2.4.1. CAD Modellierung

Mit Fusion 360 und Inventor kann man diese 3D-Entwürfe erstellen. Die Programme bieten außerdem die Möglichkeit in Teams zusammenzuarbeiten, Daten zu verwalten und Simulationen auszuführen, um dadurch Konstruktionen im Vorfeld zu validieren.

In unserem Projekt verwenden wir Inventor als CAD-Programm, jedoch besteht die Möglichkeit, das bevorzugte CAD-Programm zur Bearbeitung zu nutzen. Die entsprechenden Inventor-Dateien befinden sich im Ordner “*3d_druck_gimbal*” und können nach Belieben in andere Formate umgewandelt werden.

Autodesk Fusion 360, sowie **Autodesk Inventor** besitzen eine *kostenlose* “Campus” Version, welche Kostenlos für Lehrende und Studenten ist.

Sie unterstützen folgende 3D Druck Formate:

STL(Binär)	STL(ASCII)
3MF	OBJ

und sind erhältlich über folgenden Link: [Fusion-Download/Inventor-Download](#)

2.4.2. Erster Druck

Vor der CAD-Modellierung wurde zunächst ein initialer Testdruck durchgeführt, um zu überprüfen, ob die vorhandenen STL-Dateien für unseren Drucker geeignet sind. Je nach verwendetem Drucker können spezifische Anpassungen erforderlich sein.

Im Folgenden wird der erste Druck basierend auf dem Modell “Base 02” dargestellt. Dieses Beispiel verdeutlicht die Bedeutung eines ersten Testdrucks, da die gedruckte Struktur nicht ideal ist. Es traten Probleme mit der Stabilität aufgrund der gewählten Dichteinstellungen sowie mit den Stützstrukturen auf, die verwendet wurden, um das Einkrachen des Drucks in den Löchern zu verhindern. Das verwendete Stützmaterial war zu fest und konnte nicht ordnungsgemäß entfernt werden.

2.4. 3D Druck des Gimbal



#fig-

2. Gimbal

gimbal

Diese Probleme wurden dokumentiert, und ein zweiter Druck wurde gestartet, um den Gimbal vollständig zusammenzubauen und anschließend erneut zu bewerten, welche Anpassungen erforderlich sind.

2.4.3. Zweiter Druck

Der zweite Druck war erfolgreich, da verschiedene Faktoren wie das Stützmaterial und die Druckdichte angepasst wurden. Zudem wurde der Druck über einen längeren Zeitraum durchgeführt, um die Stabilität sicherzustellen. Es wurden stabile Bauteile hergestellt, die verwendet werden können, um den Gimbal vollständig zusammenzubauen.

2.4.4. Finale Anpassungen

Nachdem der Gimbal zusammengebaut war, stellte sich heraus, dass weitere Anpassungen in der CAD-Modellierung erforderlich sind. Wie auf der folgenden Abbildung zu sehen ist, passten nicht alle Kabel aufgrund der Kabelführung. Dariüber hinaus traten geringfügige Abweichungen bei den Schraubenlöchern auf, die angepasst werden mussten, um eine feste Verbindung der Schrauben zu gewährleisten.

Bevor diese Anpassungen gedruckt wurden, wurde der erste Prototyp evaluiert. Dabei wurde festgestellt, dass es sinnvoll ist, eine Bananenbuchse in den Gimbal zu integrieren. Diese Buchsen ermöglichen die Anwendung einer konstanten Spannung am Gimbal, was es ermöglicht, den Gimbal in Zukunft mit einem HIL-fähigen System zu verbinden und weitere Tests durchzuführen.

Die finalen Anpassungen werden in Autodesk Inventor umgesetzt. Die aktualisierten STL-Dateien sind im Ordner “3d_druck_gimbal/Inventor_files/Finaler_Druck” zu finden.

2.5. Zusammengebauter Gimbal

2.6. Elektronik des Gimbal

Um die Elektronik des Gimbals zu implementieren, d.h. der Servomotor lenkt in die entgegengesetzte Richtung aus, wenn sich die Position des Objekts ändert, um eine Position des Objekts zu erreichen, die sich nicht mit dem Gimbal verändert.

In diesem Abschnitt werden verschiedene Chiptypen (Arduino Nano und ESP8266) und drei-Achsen-Positionssensor (MPU650) zur Steuerung der Servomotoren verwendet. Der Steuercode wird in die verschiedenen Chiptypen geschrieben und der Sensor wird über Kabel mit den Chips verbunden. Um den ordnungsgemäßen Betrieb des Sensors zu gewährleisten, wird die Spannung der Batterie über den Buckconverter so eingestellt, dass an beiden Enden des Sensors die richtige Spannung anliegt.

Die wichtigsten Schritte sind wie folgt:

- Auswahl der Chips: Für die Steuerung des Servomotors werden verschiedene Chiptypen verwendet. Zu diesen Chips gehören der Arduino Nano und der ESP8266, die beide über zahlreiche Funktionen verfügen und gut kompatibel sind.
 - Sensoranschluss: Der MPU650-Sensor wird über ein Kabel mit dem Chip verbunden. Die Hauptfunktion des Sensors ist es, genaue 3-Achsen-Lage- und Bewegungsdaten für die präzise Steuerung des Servomotors zu liefern.
 - Leistungsanpassung: Um den korrekten Betrieb des Sensors sicherzustellen, wird die Batteriespannung mit einem Buckconverter angepasst. Dadurch wird sichergestellt, dass der Sensor innerhalb des geeigneten Spannungsbereichs arbeitet, um genaue Daten zu liefern.
 - Schreiben des Steuercodes: Der für die Steuerung des Servomotors erforderliche Code wird in die verschiedenen Chiptypen geschrieben. Durch das Schreiben des entsprechenden Codes können die Position und die Bewegung des Servomotors auf der Grundlage der vom Sensor gelieferten Daten gesteuert werden.
 - Motoranschluss: Schließlich wird der Servomotor über ein Kabel an den Chip angeschlossen. Dadurch kann der Chip Steuersignale an den Motor senden, was eine präzise Steuerung des Servomotors ermöglicht.

Der spezifische Schaltplan sieht wie folgt aus:

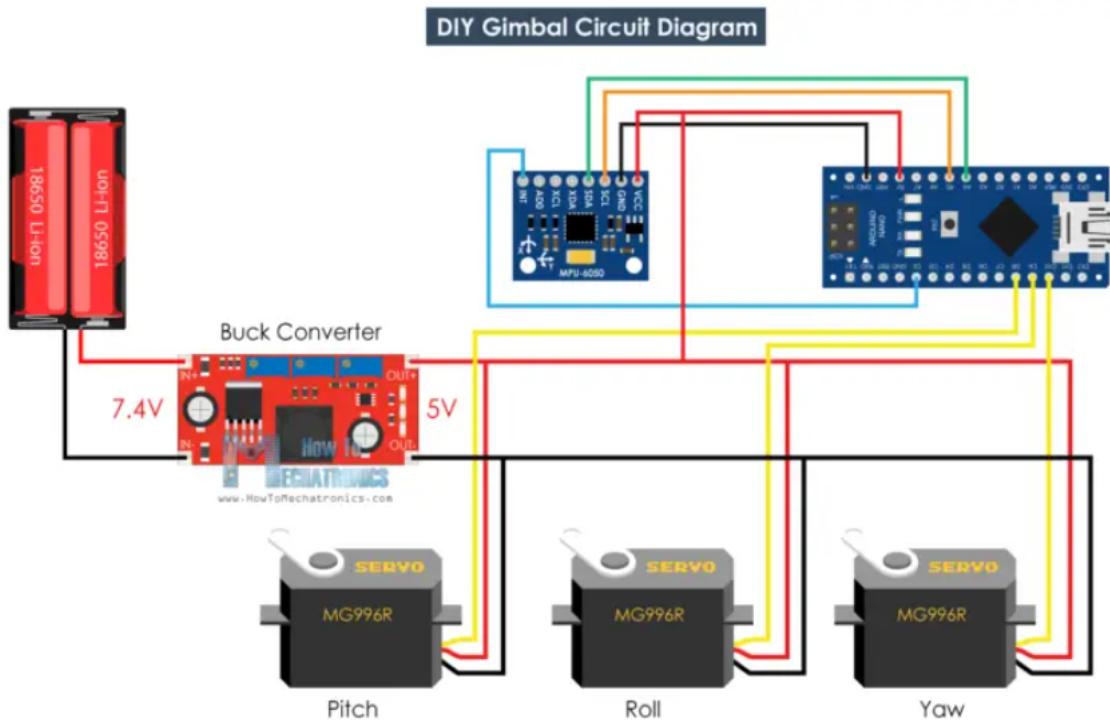


Abbildung 2.1.: Schaltplan

2. Gimbal

2.6.1. Arduino Nano

Der Arduino Nano ist ein Open-Source-Breadboard-Mikrocontroller-Board, das auf dem Microchip ATmega328P-Mikrocontroller basiert. Der Arduino Nano kann mit der integrierten Entwicklungsumgebung (IDE) der Arduino-Software programmiert werden, die für alle Arduino-Boards gleich ist und sowohl online als auch offline läuft.

2.6.2. Arduino Code Beschreibung

Um die Gimbal-Funktionalität mit dem Nano-Chip zu implementieren, muss der Code kompiliert und in die Arduino-Software hochgeladen werden. Der Code ist im Ordner `arduino_gimbal` zu finden. Bevor dieser Code verwendet werden kann, ist es wichtig, die grundlegende Funktionalität des Codes zu verstehen. Während des gesamten Prozesses muss der Code Sensordaten vom MPU6050-Sensor lesen und diese Daten zur Steuerung des Servos verwenden. Es werden bestimmte Pins definiert und die Servos an diese Pins angeschlossen.

1. Der Interrupt-Pin wird auf Pin 2 festgelegt.
2. Der Servo `servo0` wird an Pin 10 angeschlossen.
3. Der Servo `servo1` wird an Pin 9 angeschlossen.
4. Der Servo `servo2` wird an Pin 8 angeschlossen.

```
#define OUTPUT_READABLE_YAWPITCHROLL
#define INTERRUPT_PIN 2
  servo0.attach(10);
  servo1.attach(9);
  servo2.attach(8);
```

Die Aufgabe eines Interrupts ist es, dafür zu sorgen, dass der Prozessor schnell auf wichtige Ereignisse reagiert. Wenn ein bestimmtes Signal erkannt wird, unterbricht ein Interrupt die Arbeit des Prozessors und führt einen Code aus, der auf den externen Stimulus reagiert, der dem Arduino zugeführt wird. Das bedeutet, der Gimbal kann dann den Interrupt behandeln, die entsprechenden Servos steuern und die gewünschte, präzise, schnelle und reaktionsfähige Aktion ausführen.

Dann werden aus den Sensordaten die Quaternion und der Gravitationsvektor berechnet. Die Yaw-, Pitch- und Roll-Werte werden dann in Grad umgerechnet. Diese Werte werden dann zur Steuerung des Servos eingesetzt.

```
mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
```

Wenn j kleiner oder gleich 300 ist, wird der aktuelle Wert der Yaw-Achse (ypr[0]) in der Variable correct gespeichert. Dadurch wird der aktuelle Yaw-Wert erfasst, solange j kleiner oder gleich 300 ist. Dieser Wert beginnt nicht bei 0 wie die Werte für Pitch und Roll, sondern ist immer ein Zufalls Wert.

```
if (j <= 300) {
    correct = ypr[0];
    j++;
}
```

Nach 300 Messungen wird der Yaw-Wert (ypr[0]) durch Subtraktion des zuvor gespeicherten Referenzwerts angepasst. Dadurch wird der Yaw-winkel auf 0 Grad gesetzt. Die Werte für Yaw, Pitch und Roll von -90 bis +90 Grad werden dann auf Werte von 0 bis 180 für das Antriebsservo abgebildet. Mit der Schreibfunktion werden diese Werte schließlich als Steuersignale an das Servo gesendet.

```
else {
    ypr[0] = ypr[0] - correct;

    int servo0Value = map(ypr[0], -90, 90, 0, 180);
    int servo1Value = map(ypr[1], -90, 90, 0, 180);
    int servo2Value = map(ypr[2], -90, 90, 180, 0);

    servo0.write(servo0Value);
    servo1.write(servo1Value);
    servo2.write(servo2Value);
}
```

2.6.3. Aufgetretene Probleme und Lösungen in Nano

2.6.3.1. Problem 1:

Beim Kompilieren des Codes für Arduino, der auf dem MPU6050_DMP6-Beispiel in Jeff Rowbergs i2cdevlib-Bibliothek basiert, müssen die I2Cdev- und MPU6050-Bibliotheken in den entsprechenden Pfaden installiert werden.

2.6.3.2. Lösung 1:

Um das Problem zu beheben, werden folgende Schritte durchgeführt:

Der Datei-Explorer wird auf Ihrem Computer geöffnet und zum Ordner navigiert, in dem die Arduino-Bibliotheken installiert sind. Der Pfad wurde unter C:\Users\cxp\AppData\Local\Arduino15\libraries gefunden.

Die erforderlichen Bibliotheken werden heruntergeladen:

2. Gimbal

Die "I2Cdev.h"-Bibliothek kann von folgendem Link heruntergeladen werden:

Die "MPU6050_6Axis_MotionApps20.h"-Bibliothek kann von folgendem Link heruntergeladen werden: Die heruntergeladenen ZIP-Archive werden extrahiert und der Inhalt jeder Bibliothek wird in einen separaten Unterordner in Ihrem Arduino-Bibliotheksverzeichnis kopiert. Dabei können die Unterordner "I2Cdev" und "MPU6050" verwendet werden.

Die Arduino-IDE wird gestartet (sofern bereits geöffnet) oder erneut geöffnet. Nun kann der Code, der auf dem MPU6050_DMP6-Beispiel basiert, kompiliert und ausgeführt werden, ohne dass der Compiler Fehlermeldungen aufgrund fehlender Bibliotheken anzeigt.

2.6.3.3. Problem 2:

Das Problem besteht darin, dass der Arduino-Code nicht auf den ATmega328p-Chip hochgeladen werden kann. Dies kann verschiedene Ursachen haben, aber eine mögliche Lösung besteht darin, den Bootloader-Typ in der Arduino-Software richtig einzustellen.

2.6.3.4. Lösung 2:

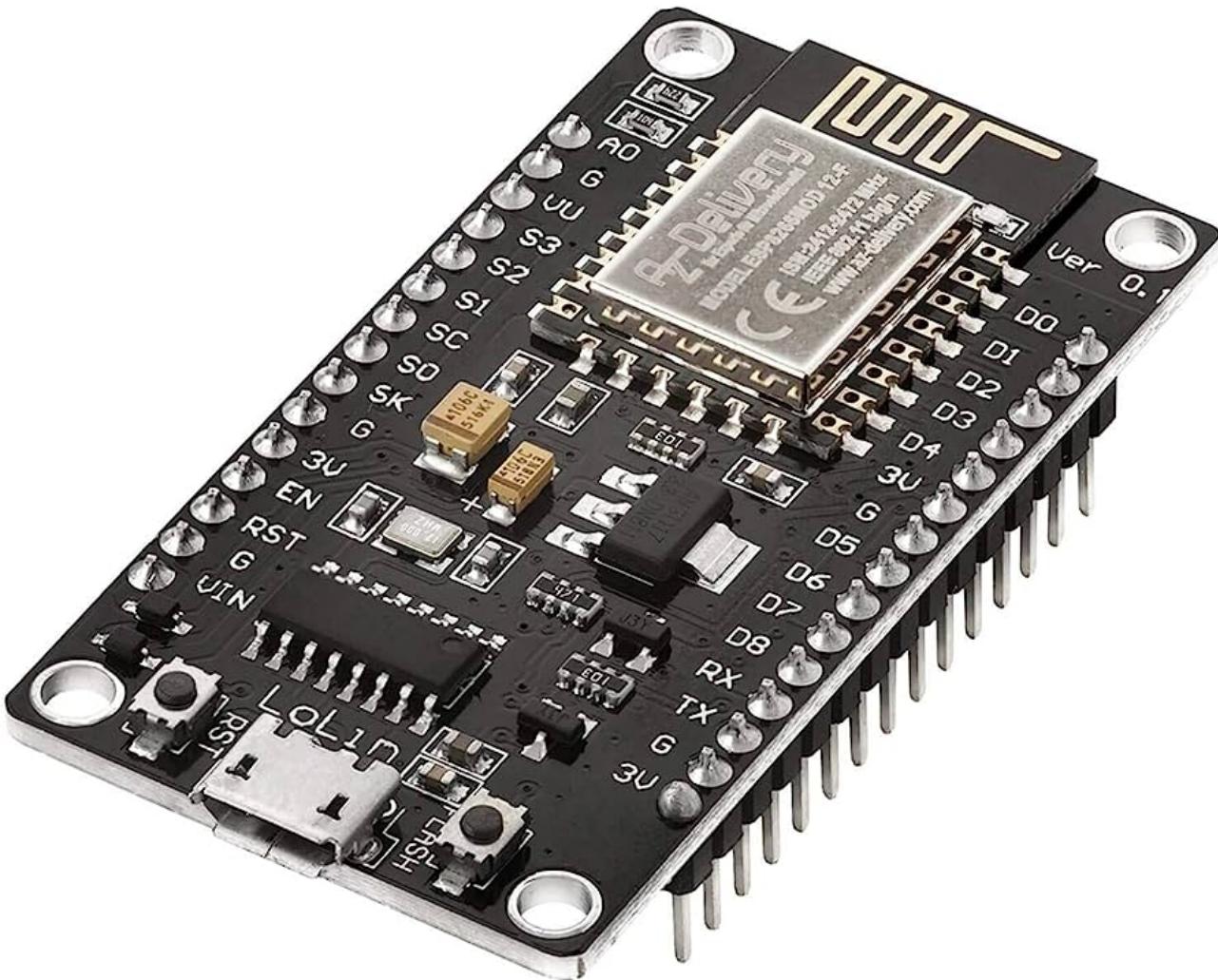
1. Öffne die Arduino-Software auf deinem Computer.
2. Gehe zum Menü "Tools" (Werkzeuge) und wähle "Board" (Platine) aus.
3. Wähle aus der Liste der verfügbaren Boards die Option "Arduino Uno" oder das entsprechende Board aus, das den ATmega328p-Chip verwendet.
4. Gehe erneut zum Menü "Tools" (Werkzeuge) und wähle "Processor" (Prozessor) aus.
5. Wähle im Dropdown-Menü die Option "ATmega328p (old Bootloader)" aus.
6. Stelle sicher, dass auch die richtige Porteinstellung ausgewählt ist, um eine Verbindung zum Arduino-Board herzustellen.
7. Klicke auf die Schaltfläche "Upload" (Hochladen), um den Code auf den ATmega328p-Chip hochzuladen.

Durch die Auswahl des ATmega328p-Bootloaders in der Arduino-Software wird sichergestellt, dass der Code korrekt mit dem Chip kommuniziert und erfolgreich hochgeladen wird. Beachte, dass es verschiedene Bootloader-Typen für den ATmega328p-Chip geben kann, daher ist es wichtig, den richtigen auszuwählen.

Hoffentlich löst diese Vorgehensweise dein Problem und du kannst den Code erfolgreich auf den ATmega328p-Chip hochladen.

2.6.4. ESP8266 NodeMCU

Der ESP8266 ist ein beliebtes und weit verbreitetes Wi-Fi-Modul, das von Espressif Systems entwickelt wurde. Es wurde für Embedded-Anwendungen und Internet of Things (IoT)-Projekte entwickelt. Das Modul integriert eine Mikrocontroller-Einheit (MCU) mit Wi-Fi-Fähigkeit, so dass es sich mit Wi-Fi-Netzwerken verbinden und mit anderen Geräten über das Internet kommunizieren kann.



Hier sind einige der wichtigsten Merkmale des ESP8266-Moduls:

1-Wi-Fi-Konnektivität: Das ESP8266-Modul verfügt über eine integrierte Wi-Fi-Konnektivität, die es ihm ermöglicht, sich mit Wi-Fi-Netzwerken zu verbinden und auf das Internet zuzugreifen. Es unterstützt verschiedene Sicherheitsprotokolle, einschließlich WEP, WPA und WPA2.

2-Mikrocontroller-Einheit (MCU): Das Modul enthält eine 32-Bit-MCU vom Typ Tensilica Xtensa LX106, die mit einer Taktrate von 80 MHz arbeitet. Sie verfügt über GPIO-Pins (General Purpose Input/Output) zur Steuerung externer Geräte und unterstützt verschiedene Schnittstellen wie I2C, SPI, UART und andere.

2. Gimbal

3-Programmierung: Der ESP8266 kann mit verschiedenen Programmiersprachen und Entwicklungsumgebungen programmiert werden. Die beliebteste Wahl ist die Arduino IDE, die eine einfache und einsteigerfreundliche Umgebung zum Schreiben und Hochladen von Code auf das Modul bietet. Darüber hinaus gibt es alternative Frameworks und Sprachen wie Micropython und Espressif's natives ESP-IDF (ESP8266 IoT Development Framework).

4-Geringer Stromverbrauch: Das Modul ist so konzipiert, dass es mit geringem Stromverbrauch arbeitet und sich daher für batteriebetriebene Anwendungen eignet. Es unterstützt Schlafmodi und kann so konfiguriert werden, dass es regelmäßig oder als Reaktion auf externe Auslöser aufwacht.

5-Speicher: Das ESP8266-Modul ist in der Regel mit verschiedenen Speicheroptionen erhältlich, darunter 512 KB oder 4 MB Flash-Speicher für die Speicherung von Programmcode und Daten.

6-Gemeinschaft und Ökosystem: Der ESP8266 hat aufgrund seiner geringen Kosten, der einfachen Handhabung und der umfangreichen Unterstützung durch die Community eine große Popularität erlangt. Es gibt eine große Online-Community, die Ressourcen, Tutorials, Bibliotheken und Beispiele zur Verfügung stellt, um Anwendern den Einstieg in ihre Projekte zu erleichtern.

7-Anwendungen: Der ESP8266 wird häufig in verschiedenen IoT-Anwendungen eingesetzt, z. B. in der Heimautomatisierung, in Sensornetzwerken, intelligenten Geräten und industriellen Überwachungssystemen. Seine Erschwinglichkeit, Vielseitigkeit und einfache Integration machen ihn zu einer beliebten Wahl für Bastler, Macher und Profis gleichermaßen.

2.6.5. Der Funktionierte Teil

Wir haben versucht, mit einigen Funktionen dieses Codes zu experimentieren, um die Funktionalität des ESP8266 zu überprüfen. Zum Beispiel haben wir versucht, das Servo direkt über den ESP8266 anzusteuern, um sicherzustellen, dass es ordnungsgemäß funktioniert. Außerdem haben wir versucht, die Daten vom MPU6050 mithilfe des ESP8266 über I2C zu lesen und auf dem Display anzuzeigen, um die Funktionalität des MPU6050 und die Datenkommunikation zwischen dem ESP8266 und dem MPU6050 zu überprüfen. Da dieser Code nicht direkt auf den ESP8266 angewendet werden kann, haben wir diese Anpassungen vorgenommen, um die verschiedenen Komponenten zu testen.

2.6.6. ESP8266 Steuerung des Servos

Die Bibliothek `Servo.h` wird eingebunden, um die Servo-Funktionalität nutzen zu können. In der `Setup()` werden die Servomotoren initialisiert und den entsprechenden digitalen Pins des Arduino-Boards zugeordnet. In diesem Fall sind die Servos an den Pins 12, 13 und 14 angeschlossen.

In `loop()` wird wiederholt ausgeführt und steuert die Servos. Es werden das `loop` verwendet, um die Servos in eine Richtung zu drehen und dann wieder zurückzudrehen.

```

for (position = 0; position < 180; position++)
{
    servo0.write(position);
    servo1.write(position);
    servo2.write(position);
    delay(waitTime);
}

```

Es werden das loop verwendet, um die Servos wieder zurückzudrehen.

```

for (position = 180; position >= 1; position--)
{
    servo0.write(position);
    servo1.write(position);
    servo2.write(position);
    delay(waitTime);
}

```

Finden Sie Code hier:[Servo-Code](#)

2.6.7. ESP8266 Daten von der MPU6050 lesen

Der entsprechende Code wurde in unsere Dateien hochgeladen. Link:[MPU6050_Display-Code](#). Dieser Code ist eine einfache OLED-Demonstration für die Beschleunigungsmesserwerte des Adafruit MPU6050-Sensors. Es werden die Bibliotheken "Adafruit_MPU6050.h", "Adafruit_SSD1306.h" und "Adafruit_Sensor.h" verwendet.

Zu Beginn des Codes werden die benötigten Bibliotheken und Objekte initialisiert. Der MPU6050 und das Display werden konfiguriert. Die Verbindung mit dem MPU6050 wird überprüft und das Display initialisiert.

Die `setup()` initialisiert die serielle Kommunikation, überprüft die Initialisierung des MPU6050 und initialisiert das Display. Es wird außerdem die Textgröße, die Textfarbe und die Displayrotation festgelegt.

Die `loop()` wird wiederholt ausgeführt. In jeder Schleife werden die Beschleunigungsmesser- und Gyroskopdaten vom MPU6050 gelesen. Diese Daten werden sowohl über die serielle Schnittstelle als auch auf dem Display angezeigt. Die Anzeige auf dem Display wird gelöscht und die Beschleunigungsmesser- und Gyroskopwerte werden aktualisiert. Danach werden sie auf dem Display angezeigt. Es gibt auch eine kurze Verzögerung am Ende jeder Schleife.

Das Tutorial finden Sie hier: [MPU6050_Display-Code](#)

2. Gimbal

2.6.8. Der nicht Funktionierte Teil

DIY Arduino Gimbal - Self-Stabilizing Plaftorm Code von Arduino IDE kompiliert, kann erfolgreich in die ESP8266 Chip zu schreiben, aber das Servo nicht mit der Drehung des MPU6050 zu bewegen, und nicht die Funktion des Gimbal erreichen.

Es kann mehrere Gründe geben, warum die Funktion des Gimbals nicht erreicht wird. Der Code basiert auf den Bibliotheken "I2Cdev" und "MPU6050_6Axis_MotionApps20", die möglicherweise nicht mit dem ESP8266 kompatibel sind. Es müssen Bibliotheken verwendet werden, die mit dem ESP8266 kompatibel sind und den Code entsprechend anpassen. Der ESP8266 kann im Vergleich zu Arduino-Boards begrenzte Ressourcen haben. Daher muss es sichergestellt, dass der Code und die Bibliotheken, die die verwendet werden, für die ESP8266-Plattform optimiert sind.

2.6.9. Bibliotheks-Kompatibilität

Die MPU6050-Bibliothek, die speziell für den ESP8266 entwickelt wurde, zu suchen und zu verwenden. Dadurch wird sichergestellt, dass die Bibliothek mit dem ESP8266 kompatibel ist.

Zum Beispiel erhältlich über folgenden link: [ESP8266 MPU6050 library](#)

Einige der darin enthaltenen Beispiele können als Referenz verwendet werden, um die darin enthaltenen Funktionen und Funktionen entsprechend zu modifizieren, damit sie mit dem MPU6050-Sensor kommunizieren und Daten verarbeiten können.

Der ESP8266 liest Daten, indem er über das I2C-Protokoll Befehle an den MPU6050 sendet. Der MPU6050 speichert seine internen Beschleunigungs- und Gyroskopdaten in seinen FIFO-Puffern, die der ESP8266 dann über I2C auslesen kann. Diese Daten können vom ESP8266 verarbeitet und zur Steuerung der Bewegung des Gimbals verwendet werden. Daher ist eine I2C-Bibliothek, die für den ESP8266 angepasst werden kann, sehr wichtig. Eine solche Bibliothek ist beispielsweise die <Adafruit_MP6050.h>, die im [MPU6050_Display-Code] verwendet wird.

2.6.10. Entwicklung des Interrupt PINs

Das Design des Interrupt-Pins und die Verwendung der Interrupt-Handler-Funktion ermöglichen es dem ESP8266, die Verfügbarkeit der MPU6050-Daten rechtzeitig zu erfassen und sie sofort zu verarbeiten, wenn sie bereit sind. Dadurch wird eine effizientere und Echtzeit-Datenverarbeitung ermöglicht.

Aber Einstellung Interrupt-Pins für die ESP8266 ist nicht das gleiche wie für den Arduino Nano.

In `setup()` kann der Interrupt-Pin mit der Funktion `pinMode()` auf den Eingangsmodus konfiguriert werden.

```
pinMode(2, INPUT);
```

Eine Interrupt-Callback-Funktion.

Diese Funktion wird aufgerufen, wenn eine Interrupt ausgelöst wird, und wird für die Behandlung von Unterbrechungseignissen verwendet. In diesem Code wird zum Beispiel eine Funktion `dmpDataReady()` definiert. Diese Funktion erkennt, ob neue Daten für den MPU6050 verfügbar sind.

```
volatile bool mpuInterrupt = false;      // indicates whether MPU interrupt pin has gone high
void dmpDataReady() {
    mpuInterrupt = true;
}
```

In `setup()` wird die Funktion `attachInterrupt()` benutzt, um die Interrupt-Callback-Funktion(`dmpDataReady()`) mit einem Interrupt-Pin zu verknüpfen. Zum Beispiel wird der Interrupt-Pin GPIO2 (entspricht dem Digital Pin 2 des Arduino) mit der Callback-Funktion `handleInterrupt()` verknüpft, ausgelöst durch eine steigende Flanke.

```
attachInterrupt(digitalPinToInterrupt(2), dmpDataReady, RISING);
```

Wenn die steigende Flanke des InterruptPins der MPU6050 ausgelöst wird, wird die Funktion `dmpDataReady()` aufgerufen. In dieser Funktion wird die Variable `mpuInterrupt` auf `true` gesetzt, was anzeigt, dass neue Daten verfügbar sind.

Das Endergebnis der Schaltung ist, dass sich ihr Wirkungsgrad um einen sehr großen Prozentsatz verbessert hat. Nach mehreren Versuchen in verschiedenen Formaten ist die Zahl der Fehler oder Unterbrechungen völlig zurückgegangen und fast nicht mehr vorhanden. Darüber hinaus ist der Aus- und Einbau der Schaltung für die Einstellung einfacher und übersichtlicher geworden.

2.6.11. Verbesserung der Schaltung:

Am Anfang haben wir die traditionelle Arbeit gemacht, wie sie in dem uns vorgelegten Geschäftsmodell und der Referenz für das Projekt angegeben war.

Aber nach einigen Experimenten an der elektrischen Schaltung fanden wir mehrere Probleme, die wir lösen mussten. Diese Probleme bestehen aus mehreren Punkten:

1. die Instabilität der elektrischen Spannung, die von dem im Stromkreis vorgesehenen Tiefsetzsteller kommt, wie in der Referenz angegeben, was zu einer Schwäche des zu den Motoren fließenden Stroms und damit zu einer Schwäche der Motoren führte, was zu einer Verzögerung des Stroms führte Reaktion der Motoren und damit eine Verzögerung der Reaktionsgeschwindigkeit, was zu einem Mangel am gesamten Projektziel, nämlich der Aufrechterhaltung von Stabilität und Gleichgewicht, und der Reaktionsgeschwindigkeit führte

2. Gimbal

2. hat uns die große Anzahl von sich überlappenden Kabeln bei der Durchführung von Leistungstests und deren Modifizierung sehr gestört. Die vielen sich überlappenden Kabeln mit ähnlichen Farben verursachten Fehler in der Verbindung. Oft musste man die Kabeln von vorne bis hinten demontieren und wieder einbauen, um den richtigen Anschluss zu erreichen, außerdem waren die Kabeln teilweise zu kurz, was dazu führte, dass sie bei der Bewegung des Motors abgeschnitten wurden.
3. die Schwäche des Stromkreises, die sich aus der häufigen Trennung der Kabeln ergibt, weil sie schwach und oberflächlich verbunden waren, was zu wiederholten Dislokationen führte und damit zur Schwäche des Stromkreises und der Notwendigkeit, die dislozierten Drähte zu demontieren und erneut zu installieren.
4. war die Anordnung der Kabeln und die Reihenfolge ihres Anschlusses an den Arduino unklar, weil die Drähte oder der Anschlussstift an den Mechanismus nicht benannt wurden und der Spannungseingang und der Masseeingang nicht bekannt waren.

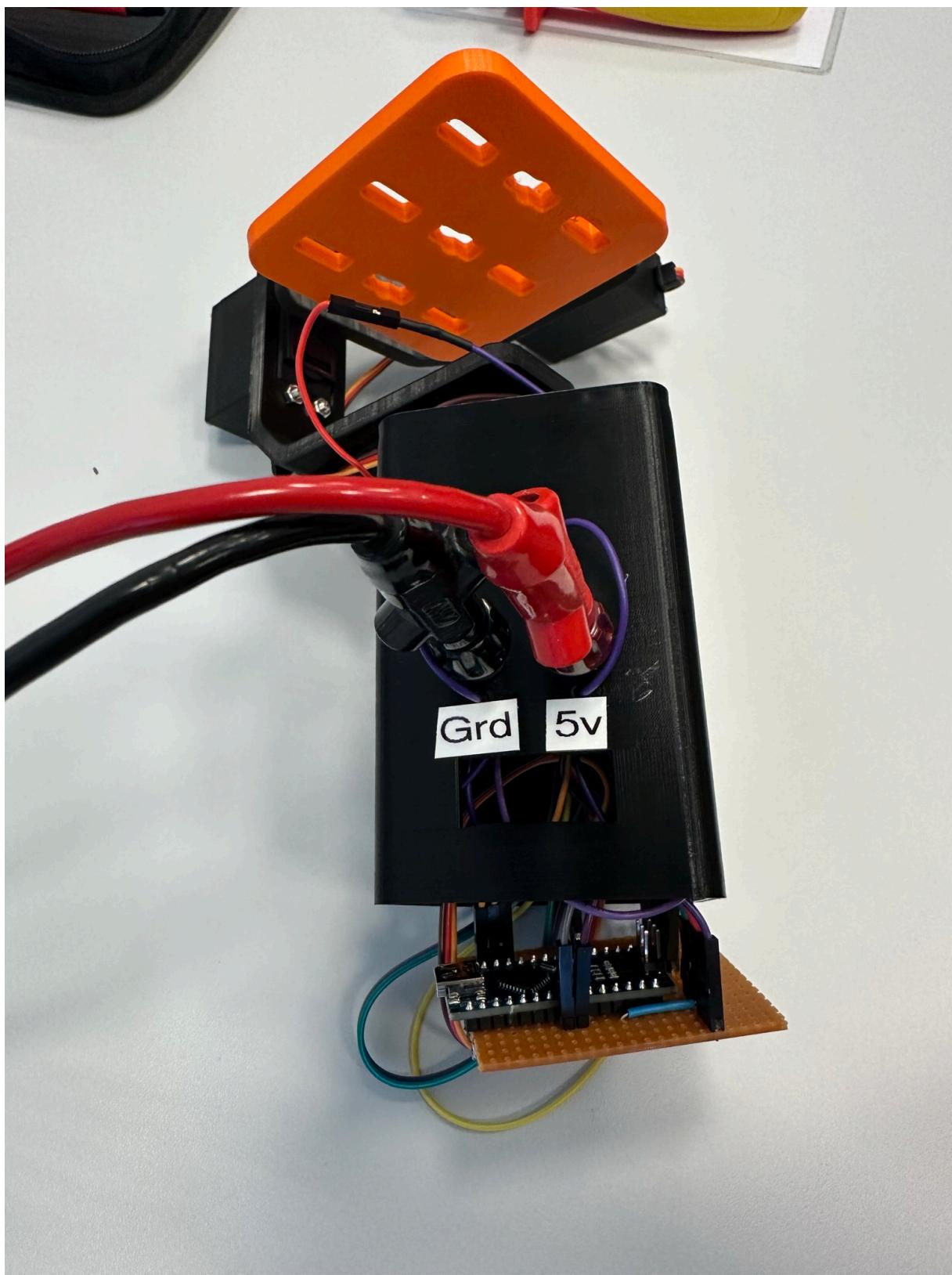
Deshalb mussten wir mehrere Änderungen an der elektrischen Schaltung vornehmen, um die Effizienz der Schaltung zu erhöhen und so diese Probleme und Fehler zu vermeiden, die beim Testen und Experimentieren zu Fehlfunktionen führen.

2.6.12. Die Lösungen waren wie folgt:

Wir haben einen wesentlichen Teil des Stromkreises geändert, nämlich den Verzicht auf den Abwärtswandler, der die für den 5-Volt-Stromkreis erforderliche Spannung liefert.

Also haben wir ihn ersetzt, indem wir die entsprechende elektrische Spannung direkt von der Stromversorgung über zwei Kabeln bezogen haben, und wir haben die Eingänge benannt. Spannungseingang und Ground ,Um es einfacher zu machen und nicht zu verwechseln.

2.6. Elektronik des Gimbal

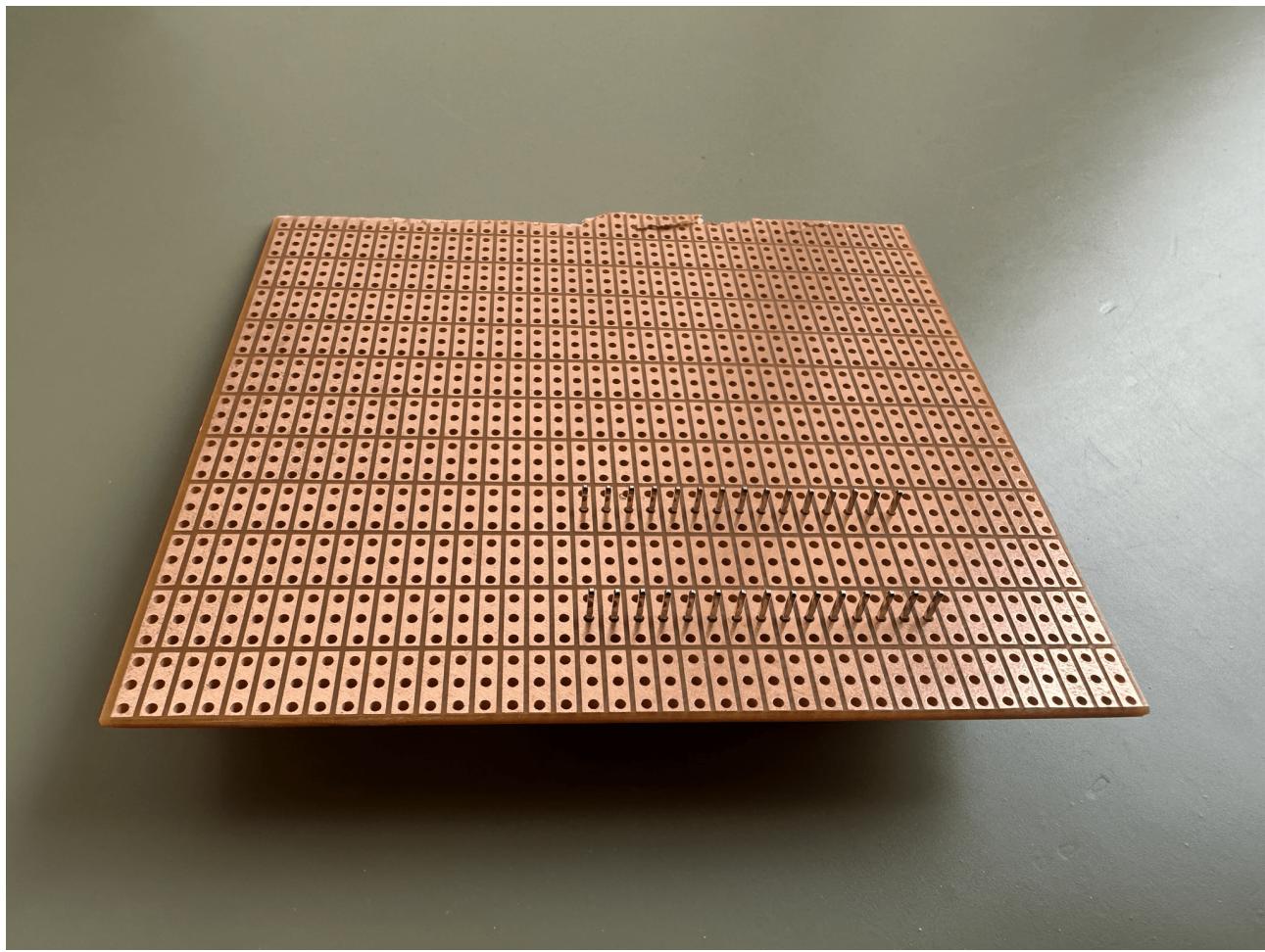


2. Gimbal

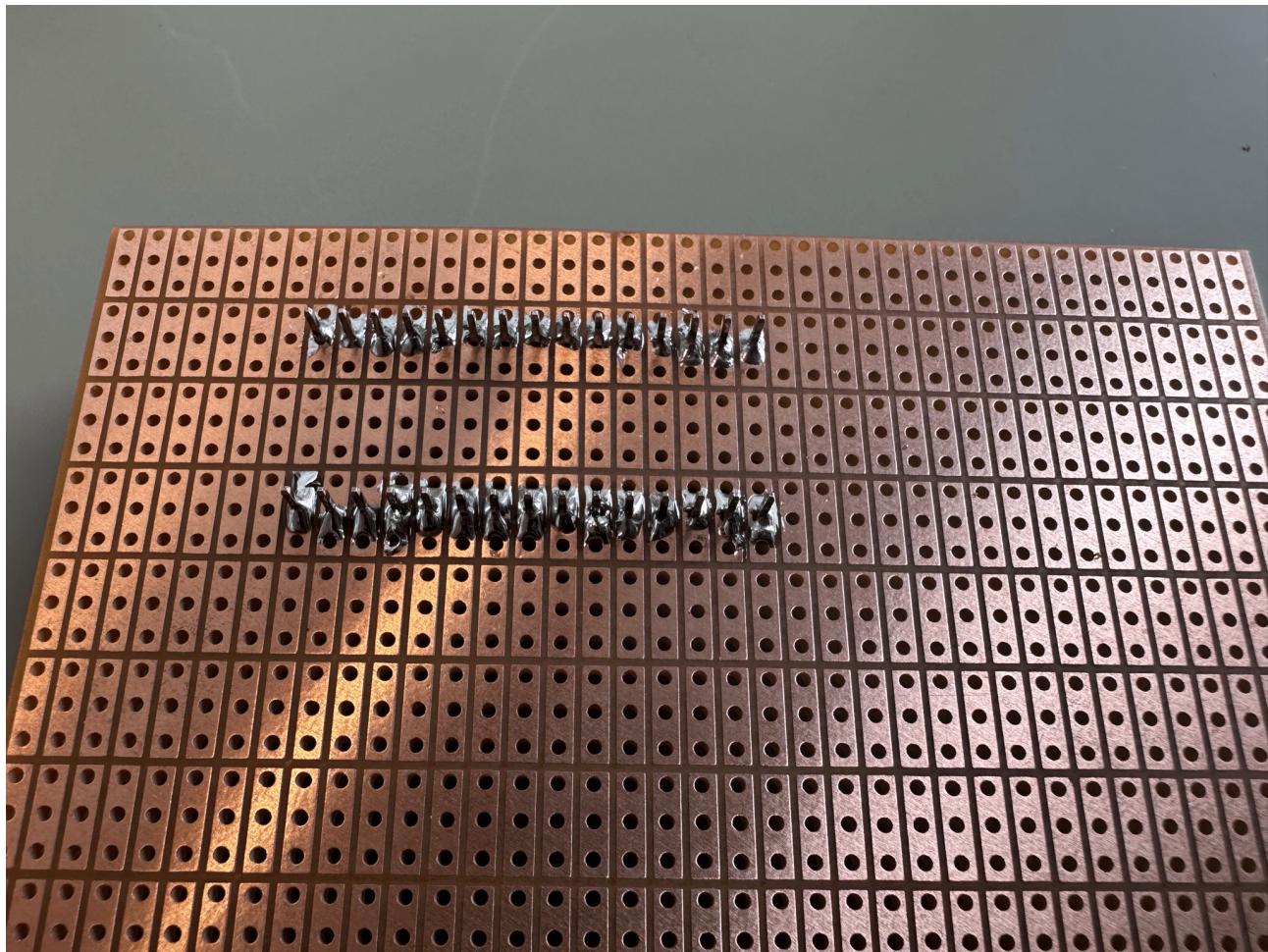


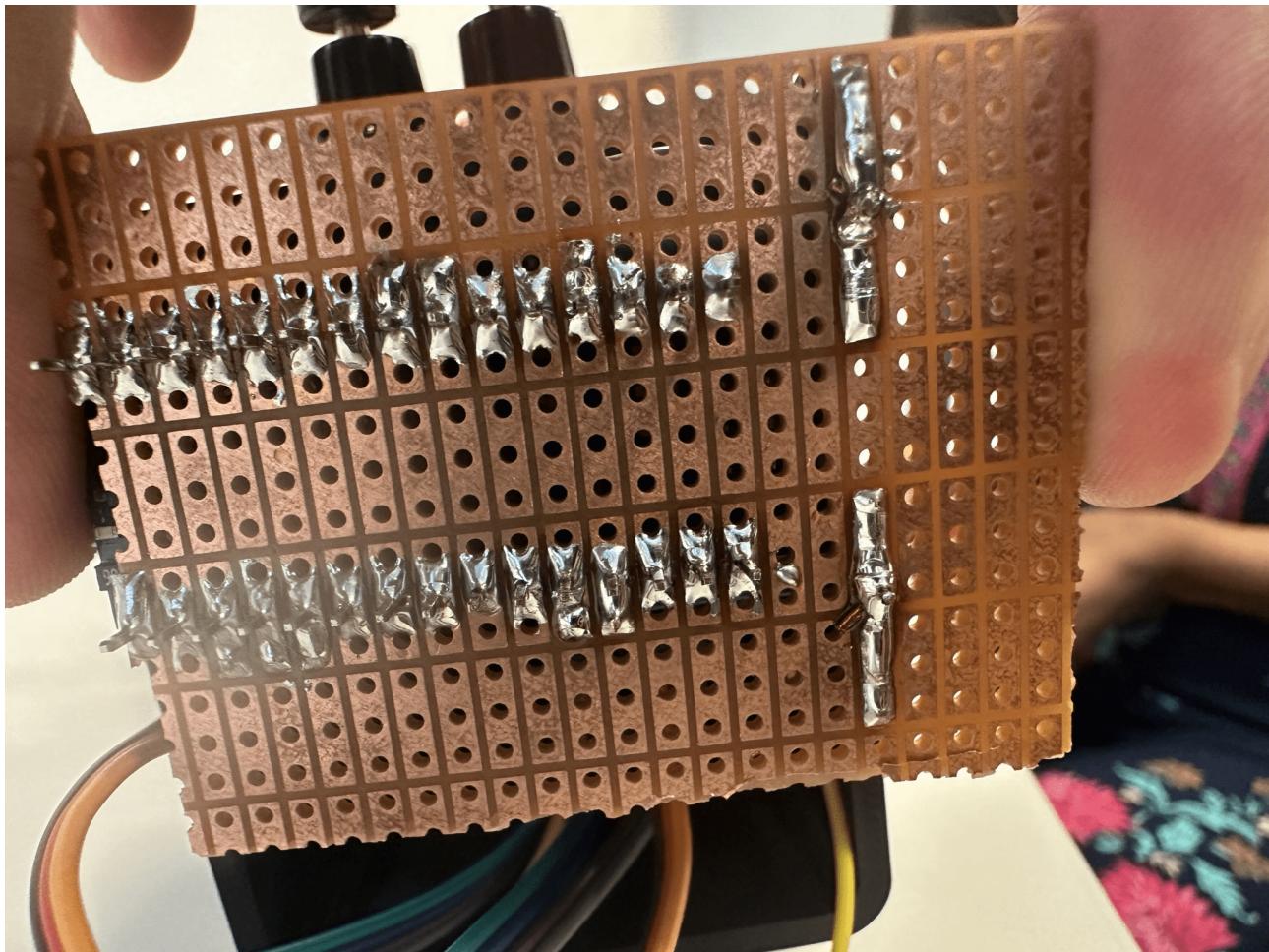
2.6. Elektronik des Gimbal

Die andere Änderung betraf die Form der Schaltung, anstatt die Kabeln direkt mit dem Arduino zu verbinden Wir zogen es vor, eine elektrische Platine zu erstellen, indem wir eine geeignete Platine auswählten, den Arduino darauf installierten, ihn anlötzten und ihn dann modifizierten, um ihn an die letzte Form anzupassen,



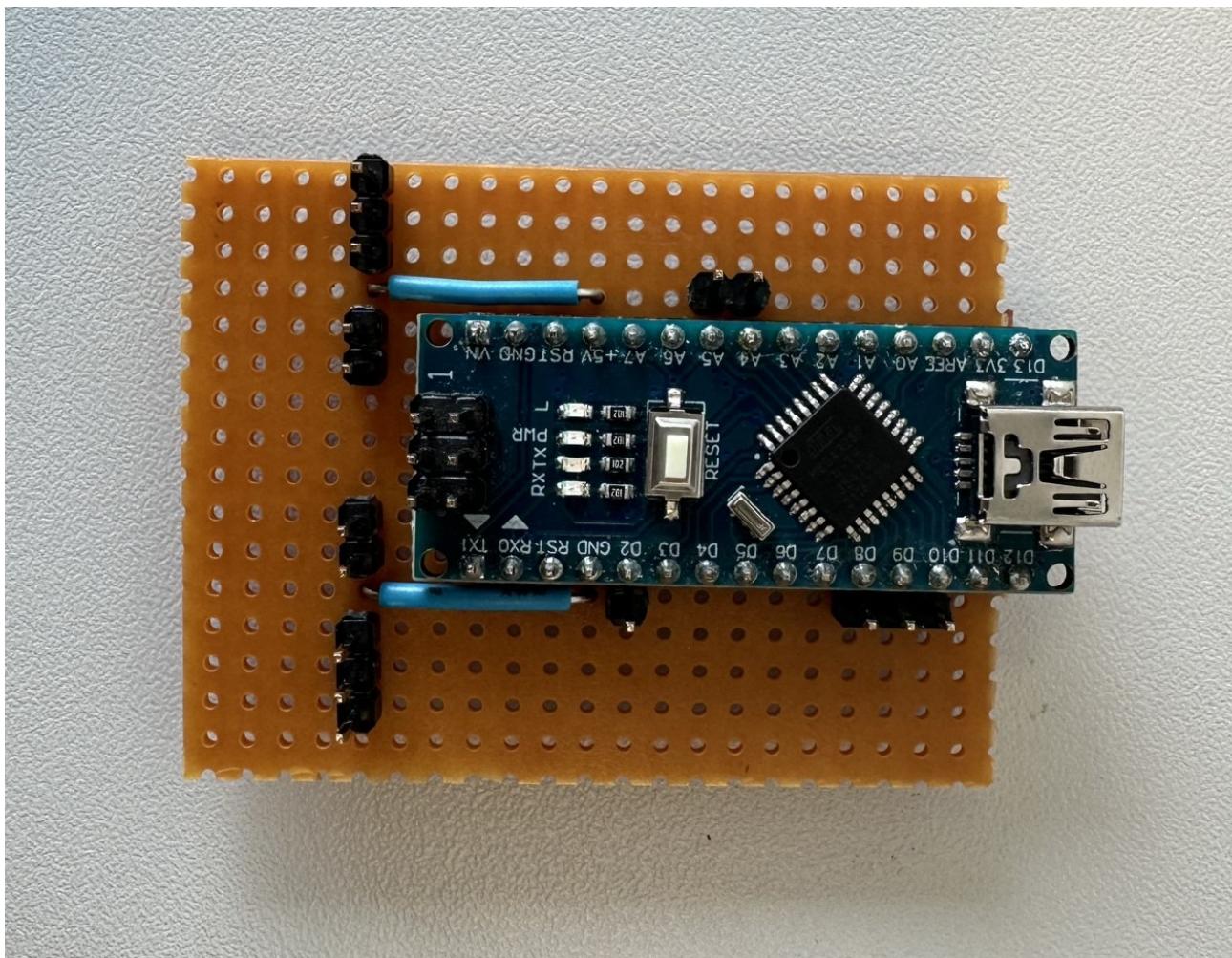
2. Gimbal



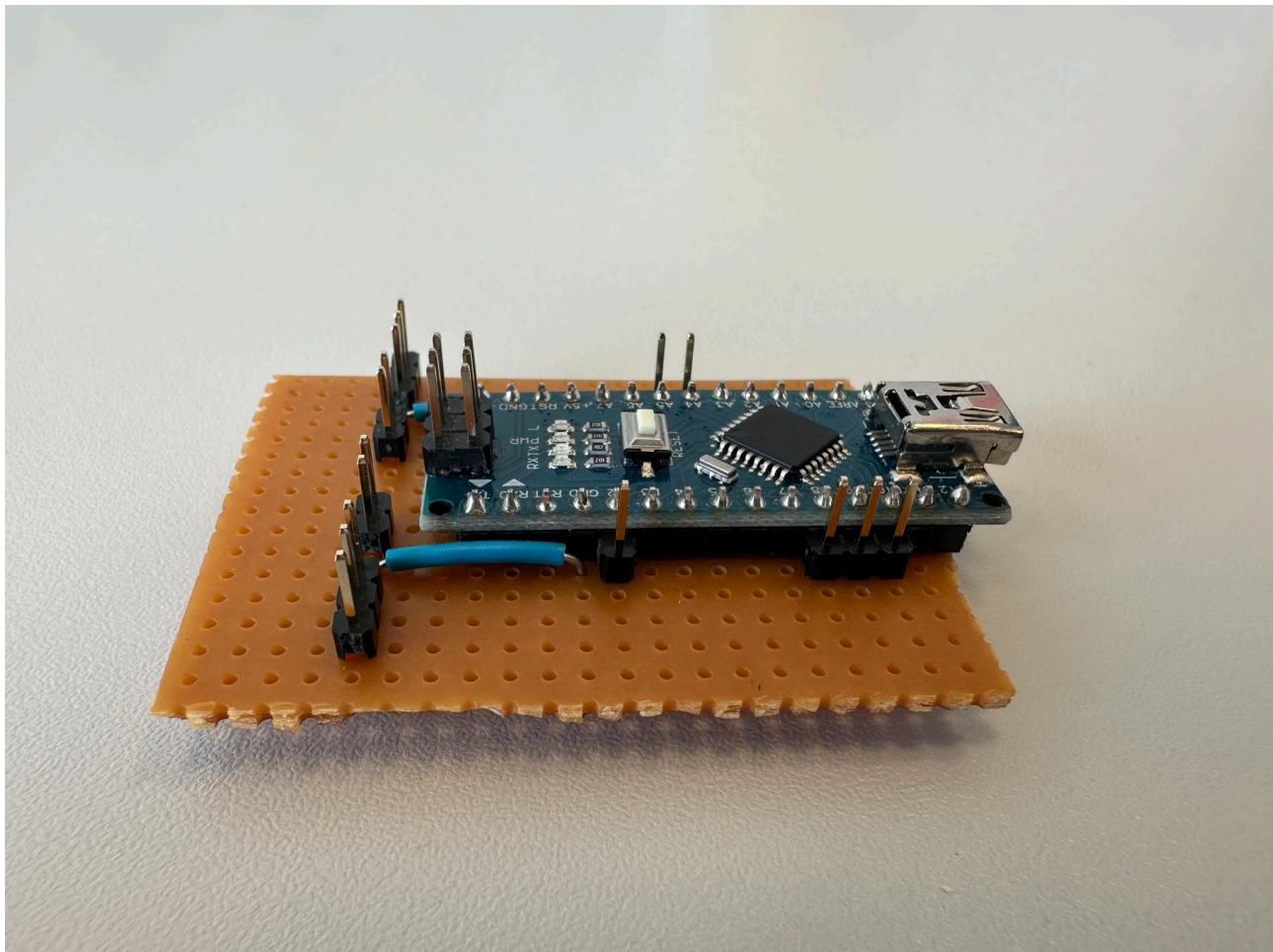


indem wir externe Pins installierten und anschweißten und gemeinsame Pins für die Spannung und gemeinsame Pins für die Ground herstellten, deren Anzahl fünf für Spannung und fünf für Ground beträgt. Wir haben dann die Kabeln installiert und mehrere Male experimentiert, nachdem wir die Form der Spannung und die Form des Stromkreises geändert hatten.

2. Gimbal

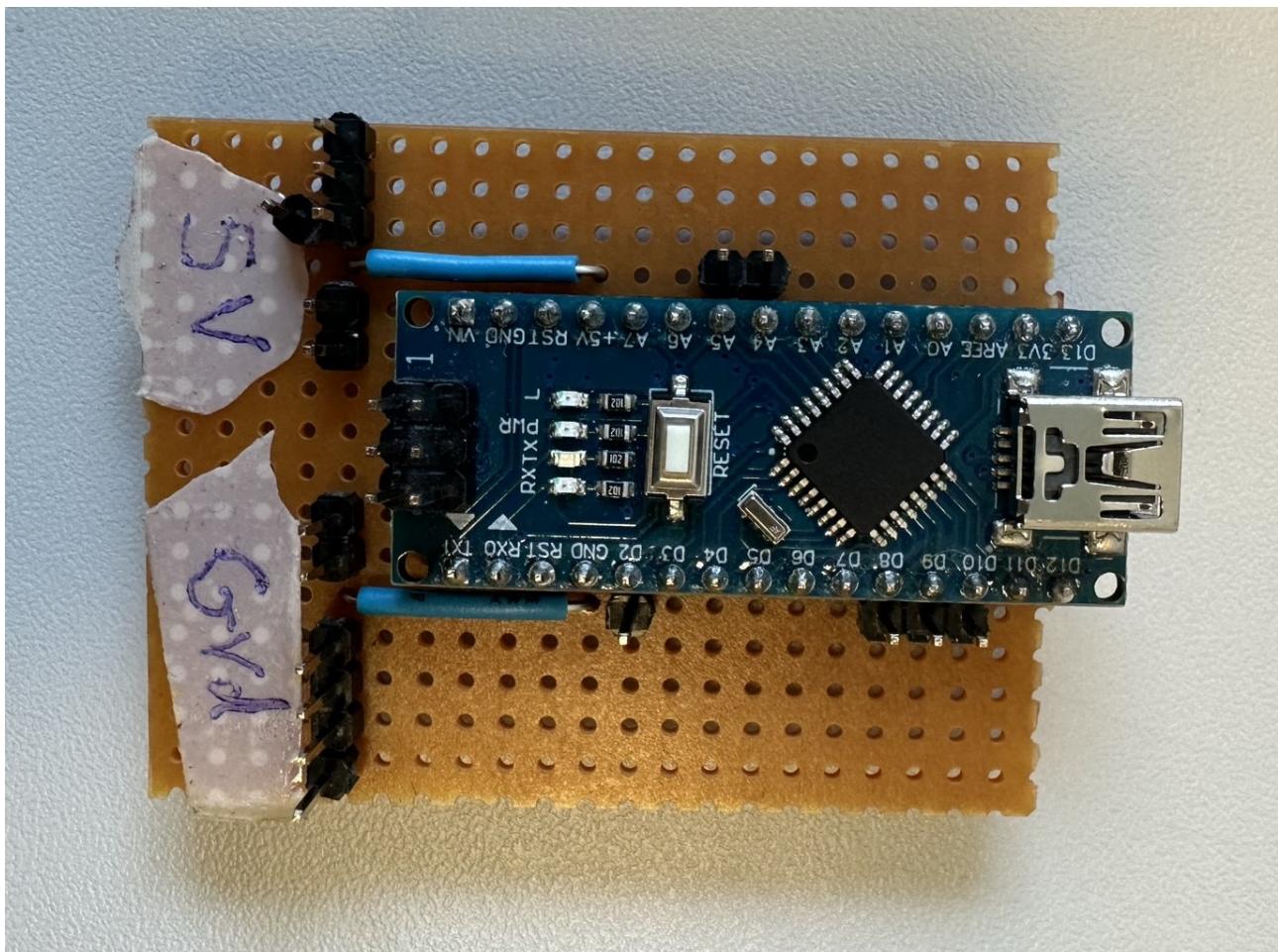


2.6. Elektronik des Gimbal



Anschließend haben wir die Eingänge benannt, um sie leichter identifizieren zu können, nämlich die Spannungseingänge und die Groundeingänge

2. Gimbal

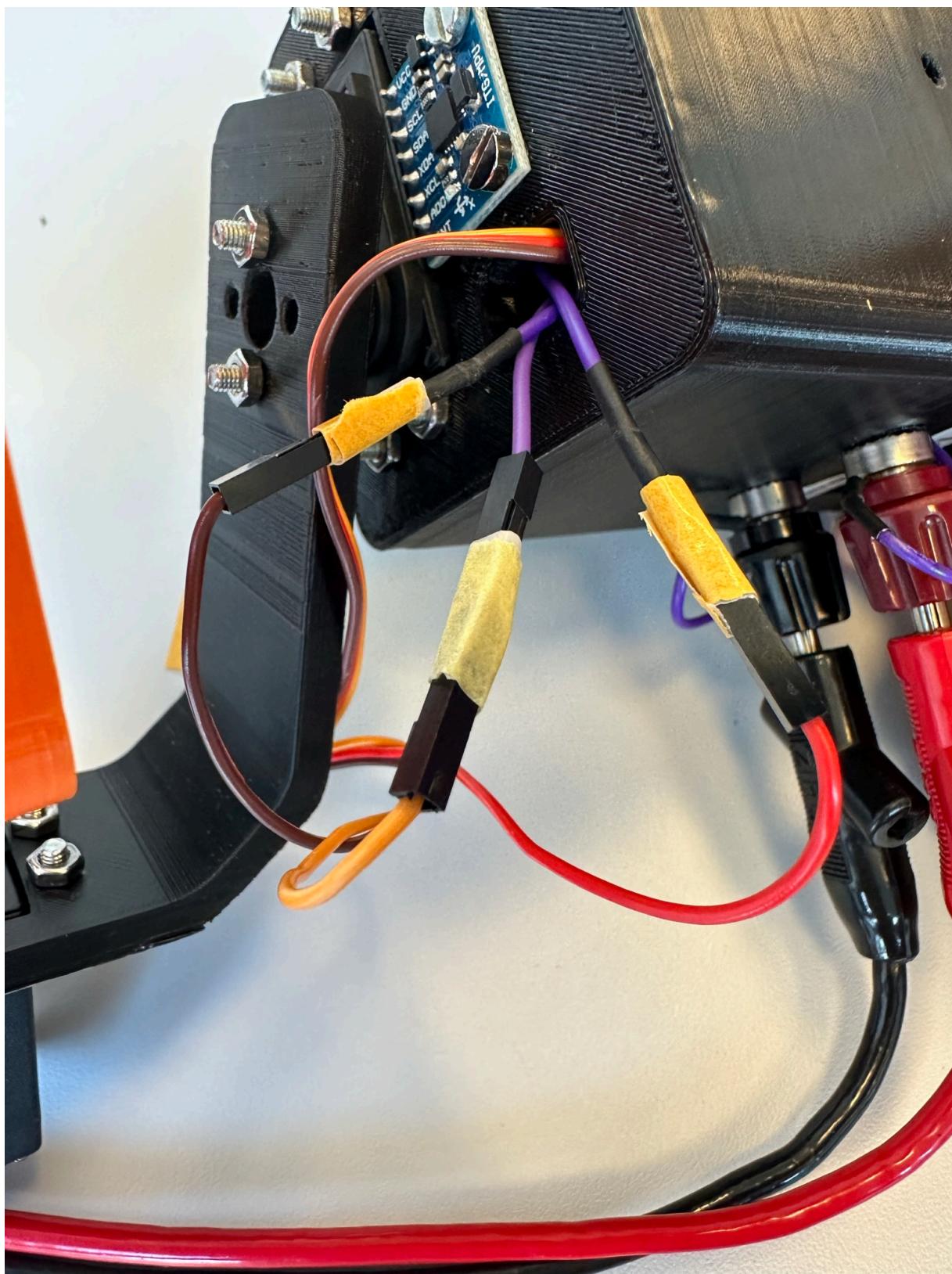


Die Ergebnisse waren gut mit wenigen Fehlern, aber wir haben auch festgestellt, dass die Anzahl der Kabel immer noch zu hoch und nicht ausreichend angeordnet ist

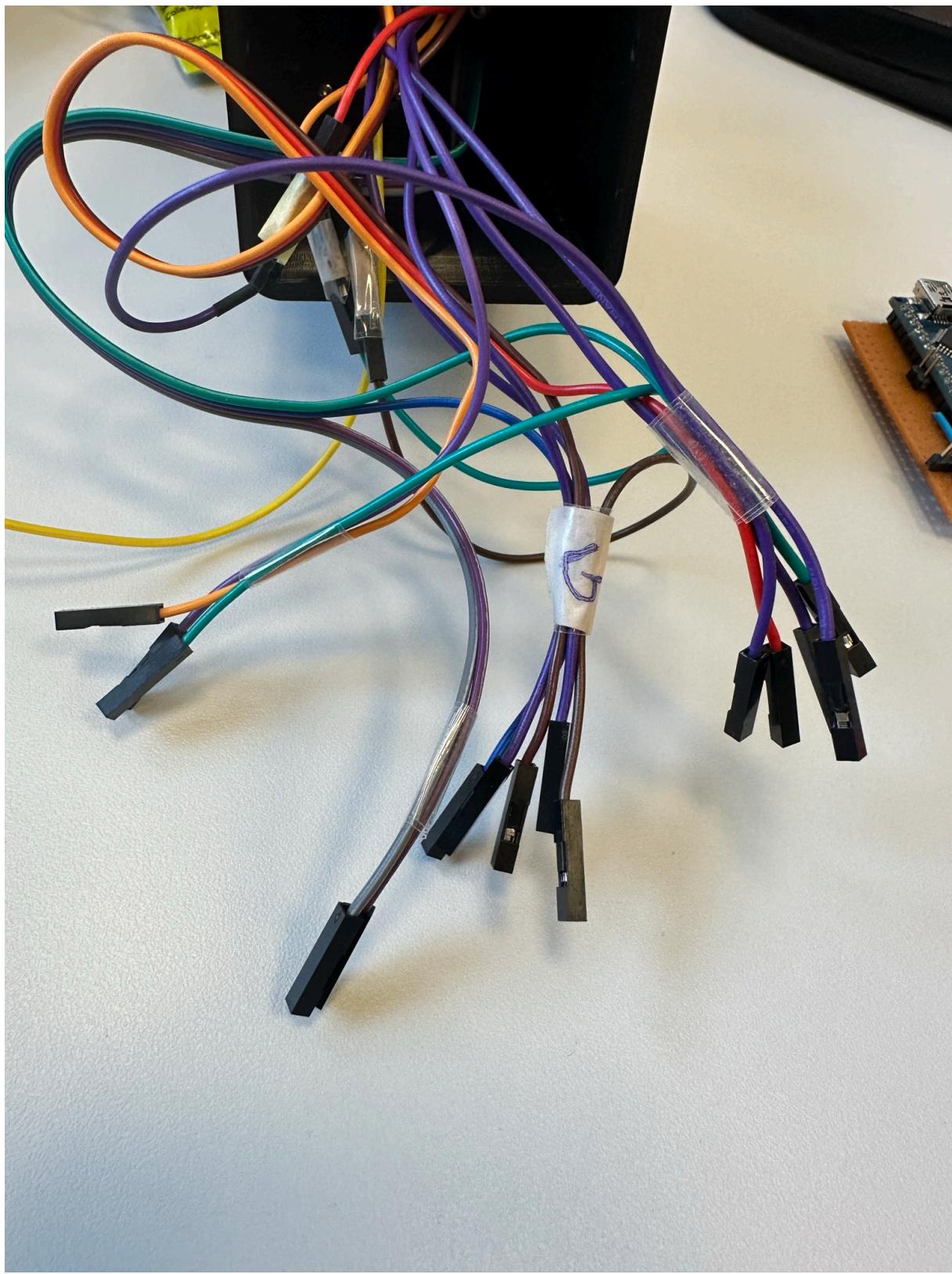
Also nahmen wir einige weitere Änderungen vor, von denen die erste darin bestand, alle Spannungskabeln zu sammeln und sie alle einzukleben und zu benennen, um den Ein- und Ausbau bei Bedarf, Änderungen oder Tests zu erleichtern.

Wir haben auch die GroundKabeln gesammelt, geklebt und benannt, um die Installation und Demonstration bei Bedarf zu erleichtern.

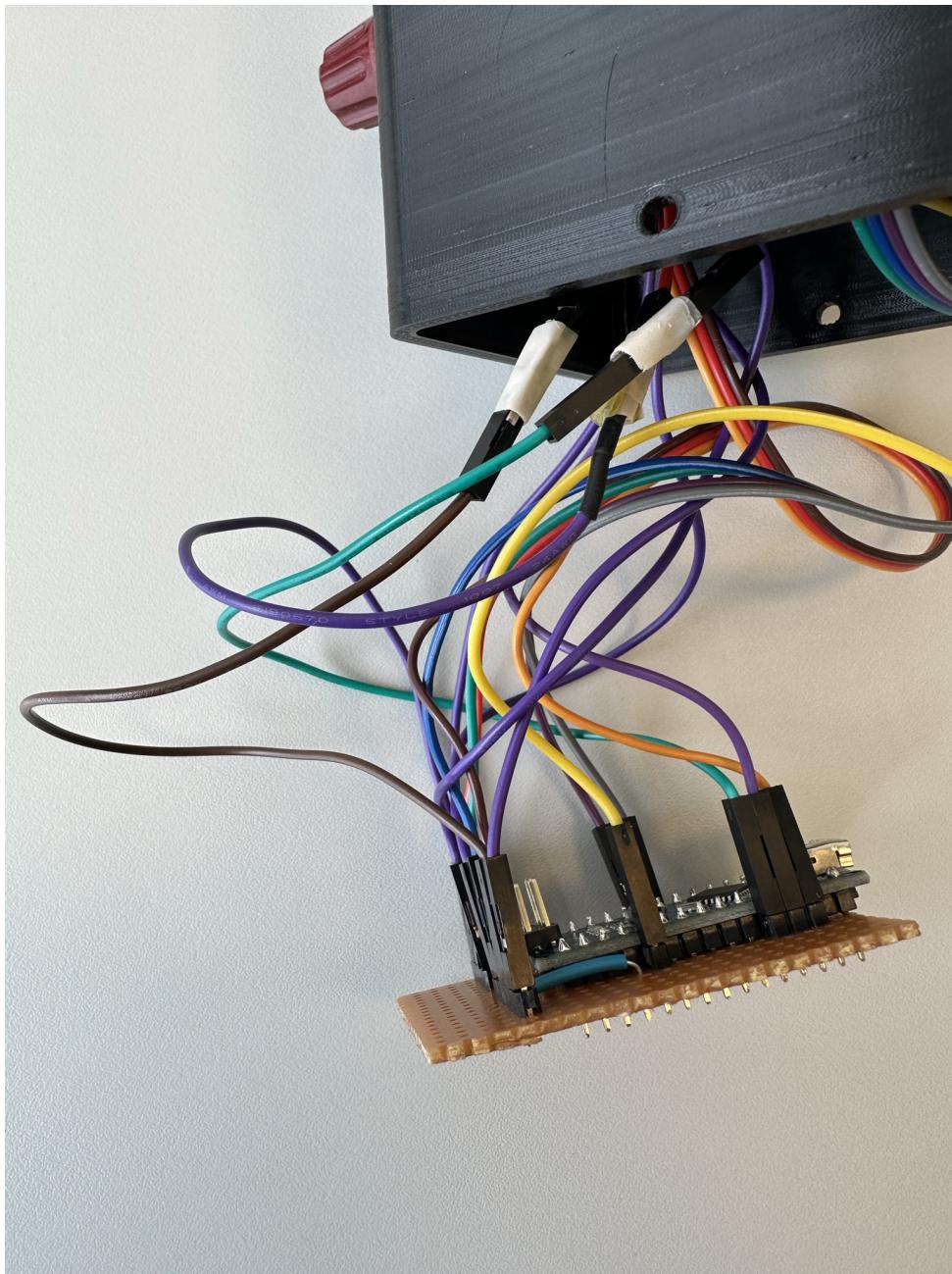
2.6. Elektronik des Gimbal



2. Gimbal

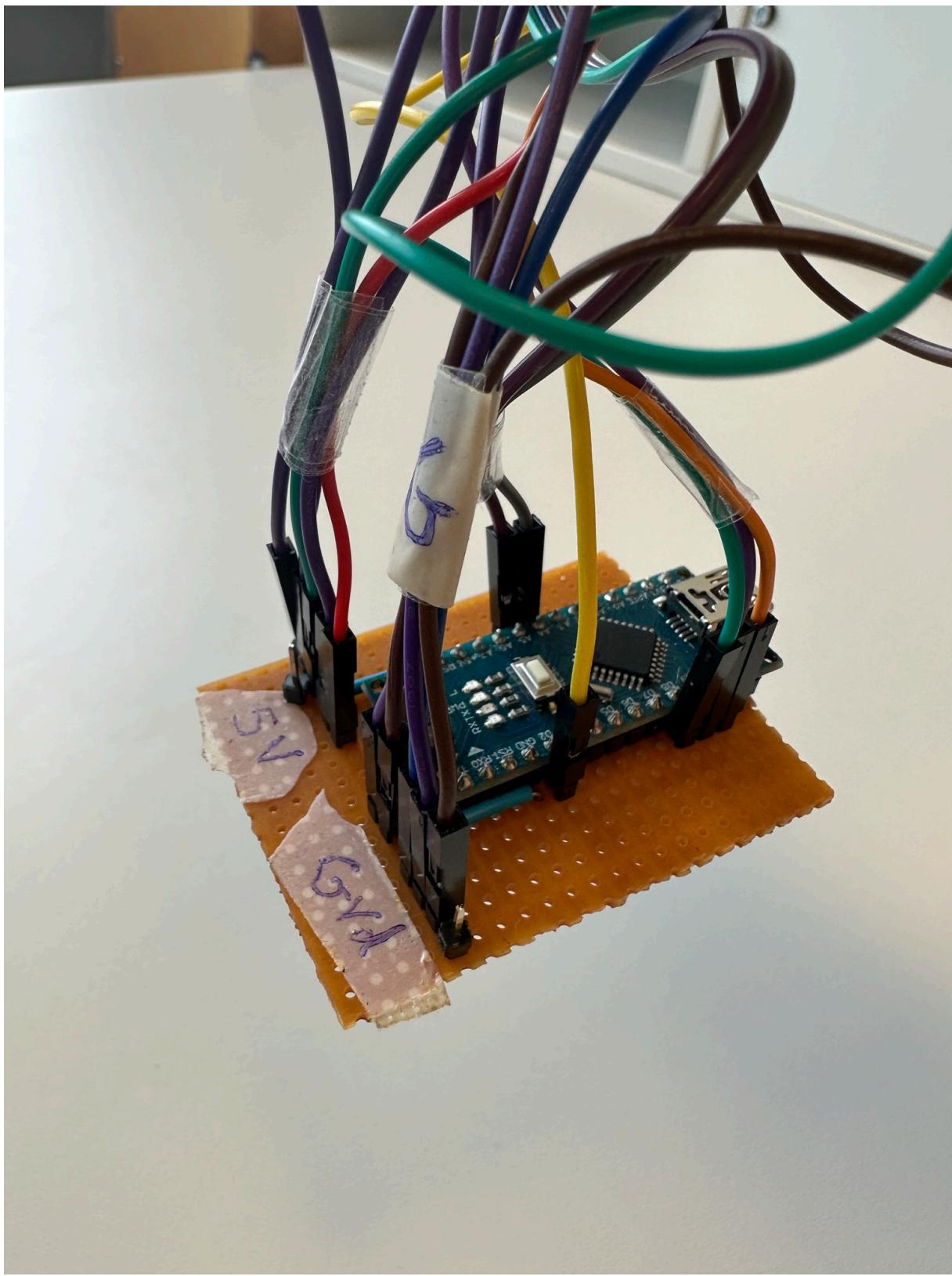


2.6. Elektronik des Gimbal



Wir klebten die Kabeln, die wir mit anderen Kabeln verlängerten, zusammen, um zu verhindern, dass sie bei Experimenten oder im Betrieb verrutschten oder demontiert wurden, um die Stabilität des Stromkreises zu erhalten.

2. Gimbal



2.7. Überprüfung der Spannungskonstanz und Stabilität eines Buck-Konverters

Das Endergebnis der Schaltung ist, dass sich ihr Wirkungsgrad um einen sehr großen Prozentsatz verbessert hat.

Nach mehreren Versuchen in verschiedenen Formaten ist die Zahl der Fehler oder Unterbrechungen völlig zurückgegangen und fast nicht mehr vorhanden.

Darüber hinaus ist der Aus- und Einbau der Schaltung für die Einstellung einfacher und übersichtlicher geworden.,

2.7. Überprüfung der Spannungskonstanz und Stabilität eines Buck-Konverters

In diesem Test wird das Konzept eines Buck-Konverters auf seine Fähigkeit zur Aufrechterhaltung einer konstanten Ausgangsspannung überprüft. Insbesondere wird untersucht, ob die Spannungsquelle konstant bleibt und ob am Ausgang des Buck-Konverters das Eingangssignal im weiteren Verlauf auf 5V über eine bestimmte Zeit (5 Minuten) konstant bleibt oder ob sich die Spannung aufgrund von Temperatureinflüssen stark verändert. Eine Toleranz von $\pm 5\%$ wird als akzeptabel angesehen.

2.7.1. Testkonzept

Das Testkonzept besteht darin, die Spannungsquelle an den Eingang des Buck-Konverters anzuschließen und den Eingangs- sowie den Ausgangsspannungspegel mit einem Oszilloskop zu überwachen. Dabei wird die Konstanz der Spannungsquelle überprüft und gemessen, ob die Ausgangsspannung des Buck-Konverters über einen Zeitraum von 5 Minuten konstant bei 5V bleibt.

2.7.2. Zustand des Geräts

Es liegt Versorgungsspannung mit 5V an, Anschalter ist gedrückt.

2.7.3. Durchführung

1. Verbinden Sie die Spannungsquelle mit dem Eingang des Buck-Konverters.
2. Schließen Sie den Eingangs- und Ausgangsanschluss des Buck-Konverters an ein Oszilloskop an.
3. Überprüfen Sie, ob die Spannungsquelle eine konstante Spannung liefert.
4. Nehmen Sie Messungen der Ausgangsspannung des Buck-Konverters bei einer bestimmten Eingangsspannung vor.
5. Überwachen Sie die Ausgangsspannung über einen Zeitraum von 5 Minuten und stellen Sie sicher, dass sie konstant bei 5V bleibt.
6. Berücksichtigen Sie eine Toleranz von $\pm 5\%$ und dokumentieren Sie jegliche Abweichungen von diesem Wert

2. Gimbal

2.7.4. Testanforderungen

1. Überprüfung, ob die Spannungsquelle konstant bleibt.
2. Messung der Ausgangsspannung des Buck-Konverters bei einer bestimmten Eingangsspannung.
3. Überprüfung, ob die Ausgangsspannung über einen Zeitraum von 5 Minuten konstant bei 5V bleibt, unter Berücksichtigung einer Toleranz von +-5%.

2.7.5. Zustand des Konzepts

Das System ist an.

Durchführung:

1. Verbinden Sie die Spannungsquelle mit dem Eingang des Buck-Konverters.
2. Schließen Sie den Eingangs- und Ausgangsanschluss des Buck-Konverters an ein Oszilloskop an.
3. Überprüfen Sie, ob die Spannungsquelle eine konstante Spannung liefert.
4. Nehmen Sie Messungen der Ausgangsspannung des Buck-Konverters bei einer bestimmten Eingangsspannung vor.
5. Überwachen Sie die Ausgangsspannung über einen Zeitraum von 5 Minuten und stellen Sie sicher, dass sie konstant bei 5V bleibt.
6. Berücksichtigen Sie eine Toleranz von +-5% und dokumentieren Sie jegliche Abweichungen von diesem Wert.

2.7.6. Ergebnisvergleich mit Anforderung

1. Wenn die Spannungsquelle eine konstante Spannung liefert und die Ausgangsspannung des Buck-Konverters bei 5V bleibt, liegt das Ergebnis im Einklang mit den Anforderungen.
2. Falls die Ausgangsspannung des Buck-Konverters innerhalb der Toleranz von +-5% schwankt, ist das Ergebnis ebenfalls akzeptabel.
3. Jegliche Abweichungen außerhalb der definierten Toleranz würden bedeuten, dass das beschriebene Testkonzept des Buck-Konverters die Testanforderungen nicht erfüllt.

2.8. Überprüfung der Ausgangsspannungsproduktion des Buck-Konverters

In diesem Test wird das Konzept 1 eines Buck-Konverters durchgeführt, um die Fähigkeit zur Erzeugung einer Ausgangsspannung zu bestimmen. Das Hauptziel des Tests besteht darin zu überprüfen, ob der Buck-Konverter eine Ausgangsspannung erzeugt.

2.8. Überprüfung der Ausgangsspannungserzeugung des Buck-Konverters

2.8.1. Testkonzept

Das Testkonzept besteht darin, eine Konstantspannungsquelle (Batterie und Laboratory DC power supply) am Eingang des Buck-Konverters anzuschließen und mithilfe eines Multimeters den Ausgangsspannung zu messen. Es wird überprüft, ob der Buck-Konverter bei verschiedenen Eingangsspannungen eine Ausgangsspannung liefert. Die Messergebnisse des Multimeters werden analysiert, um sicherzustellen, dass der Buck-Konverter unabhängig von den variierenden Eingangsspannungen eine Ausgangsspannung aufweist.

2.8.2. Zustand des Geräts

es liegt versorgungsspannung mit 5V an, Anschalter ist gedrückt.

2.8.3. Testanforderung

1. Überprüfung der Ausgangsspannung des Buck-Konverters bei Erhalt einer Eingangsspannung
2. Stellen Sie sicher, dass der Buck-Konverter ordnungsgemäß angeschlossen und betriebsbereit ist.
3. Legen Sie eine bestimmte Eingangsspannung fest. Schalten Sie die Eingangsspannungsquelle ein, um die festgelegte Eingangsspannung bereitzustellen.
4. Überprüfen Sie visuell, ob eine Ausgangsspannung an den Ausgangsanschlüssen des Buck-Konverters vorhanden ist.
5. Verwenden Sie ein Messinstrument zur Spannungsüberwachung, um die genaue Ausgangsspannung zu messen und sicherzustellen, dass sie den erwarteten Werten entspricht.
6. Wiederholen Sie den Test mit verschiedenen Eingangsspannungen, um sicherzustellen, dass der Buck-Konverter bei verschiedenen Eingangsspannungspegeln eine Ausgangsspannungen liefert.

2.8.4. Ergebnisvergleich mit Anforderung

1. Wenn das Multimeter eine Spannung anzeigt (einige Volt), entspricht das Ergebnis den Anforderungen.
2. Wenn das Multimeter keine Spannung anzeigt oder die Spannung sehr gering ist (einige Millivolt oder weniger), entspricht das Ergebnis nicht den Anforderungen.

Teil II.

Anhang

3. No Magic MBSE Software

