Graduation Project

Nonlinear Attitude Control of Spacecraft Using Variable Speed Control Moment Gyroscopes with Singularity Avoidance

By

Afnan Ibrahim Mohammed
Ahmed Mamdouh Ahmed Abdelmonem
Ahmed Mohamed Ramadan Mohamed
Ali Ahmed Mohamed Ali Almorshdy
Alyaa Mohammed Abdelfatah Etman
Aya Alaa Abdelkhaleq Elramady
Bassant Abd El-Aziz Abd El-Aziz
Doaa Ahmed Ali Ahmed
Jehad Moamen Mahmoud Ahmed
Manar Mahmoud Mostafa
Rahma Osama Saad Alsayyad
Riham Abdel Aziz Abdel Raziq
Sara Kotb Mahran Mohamed
Zeyad Alaa Abd Elkareem

A Thesis Submitted to

Faculty of Navigation Science and Space Technology

Bachelor in Space Navigation

Supervisors

Dr. Ahmed Aboulftouh **Dr. Mohammed El Farran**

July 2025

Disclaimer

We hereby declare that this thesis is our original work and that no part of it has been submitted for a degree qualification at Beni-Suef University. We further declare that we have appropriately acknowledged all sources used and have cited them in the references section.

Date: 13\07\2025

Acknowledgments

We want to express our gratitude and heartfelt appreciation to Dr. Ahmed Aboulfotouh, Dr. Haitham Elshimy, Dr. Mohamed Okasha, and Dr. Osama Shalabiea. Their exceptional guidance, unwavering support, and invaluable expertise have been truly instrumental at every stage of this work. Their insightful discussions, constructive feedback, and continuous encouragement significantly contributed to the depth and success of this project.

Contents

List of Figures

List of Tables

Nomenclature

$[\tilde{\mathbf{b}}_v]$	Skew-symmetric matrix of $\mathbf{b}_v$
$[\tilde{\mathbf{g}}]$	Skew-symmetric matrix of $\mathbf{g}$
$[\tilde{\mathbf{s}}]$	Skew-symmetric matrix of MRP vector
$[\boldsymbol{C}]$	Direction Cosine Matrix mapping from inertial to body frame
$[\boldsymbol{I}]$	Spacecraft moment of inertia matrix
$[\boldsymbol{J}]$	Combined gimbal inertia matrix
$[\boldsymbol{M_i(y)}]$	Basic rotation matrix about axis i by angle y
$\boldsymbol{\alpha_{ij}}$	Direction angle between $\hat{\mathbf{b}}_i$ and $\hat{\mathbf{n}}_j$
$\boldsymbol{\omega}$	Angular velocity vector of spacecraft body
$\boldsymbol{\omega_r}$	Reference angular velocity
$\boldsymbol{\sigma}$	MRPs of current attitude
$\boldsymbol{\sigma_e}$	Attitude error in MRPs
$\boldsymbol{\sigma_r}$	Reference attitude in MRPs
$\boldsymbol{H}$	Total angular momentum
$\boldsymbol{H_B}$	Angular momentum of the body
$\boldsymbol{H_G}$	Angular momentum of the gimbal
$\boldsymbol{H_W}$	Angular momentum of the wheel
$\boldsymbol{L}$	External torque vector
$\boldsymbol{\delta\omega}$	Angular velocity error
$\dot{\mathbf{s}}$	Time derivative of MRPs
$\dot{\mathbf{s}}_{\boldsymbol{S}}$	Time derivative of MRP shadow set
$\hat{\mathbf{b}}_i$	Unit vector of the i-th axis of the body frame
$\hat{\mathbf{e}}$	Unit vector along principal rotation axis
$\hat{\mathbf{n}}_j$	Unit vector of the j-th axis of the inertial frame
$\mathbf{B}$	Body frame of the spacecraft
$\mathbf{b}$	Quaternion vector $[b_0, b_1, b_2, b_3]^T$
$\mathbf{B(s)}$	Jacobian matrix of MRPs
$\mathbf{b}_v$	Vector part of quaternion $[b_1, b_2, b_3]^T$
$\mathbf{g}$	Principal rotation vector $\mathbf{g} = \Phi\hat{\mathbf{e}}$
$\mathbf{K}$	Gain matrix for attitude error
$\mathbf{N}$	Inertial (reference) frame
$\mathbf{P}$	Gain matrix for angular velocity error

$\mathbf{q}$ Quaternion $[q_0, q_1, q_2, q_3]^T$
$\mathbf{r}_B$ Vector $\mathbf{r}$ in body frame
$\mathbf{r}_N$ Vector $\mathbf{r}$ in inertial frame
$\mathbf{s}$ Modified Rodrigues Parameters (MRP) vector
$\mathbf{s}_S$ Shadow set of MRPs
Ω Longitude of ascending node or CMG wheel speed (context dependent)
ω Argument of periapsis or angular velocity (context dependent)
ϕ Roll angle
ψ Yaw angle
θ Pitch angle
b_0 Scalar part of quaternion
c_x Cosine of angle x, i.e., $c_x = \cos x$
C_{ij} Element of the DCM representing cosine between $\hat{\mathbf{b}}_i$ and $\hat{\mathbf{n}}_j$
i Inclination angle
s^2 Squared norm of MRP vector
s_x Sine of angle x, i.e., $s_x = \sin x$
APFs Artificial Potential Functions
CAD Computer-Aided Design
CMGs Control Moment Gyroscopes
DCM Direction Cosine Matrix
DOF Degrees of Freedom
FBD Free-Body Diagram
GT2 Belt type with 2 mm pitch, used in pulley systems
HIL Hardware-in-the-Loop
IMU Inertial Measurement Unit
KF Kalman Filter
kN Kilonewton, unit of force
ML Machine Learning
MPU6050 IMU sensor module integrating accelerometer and gyroscope
MRPs Modified Rodrigues Parameters
Nm Newton meter, unit of torque
PLA Polylactic Acid, a 3D printing plastic used for shafts
PPO Proximal Policy Optimization
PWM Pulse Width Modulation
RL Reinforcement Learning
RWs Reaction Wheels
SAC Soft Actor-Critic
SGCMG Single-Gimbal Control Moment Gyroscope
SKF Svenska Kullagerfabriken, a bearing manufacturer
VSCMGs Variable Speed Control Moment Gyroscopes

Abstract

This thesis investigates spacecraft attitude control using Variable Speed Control Moment Gyroscopes (VSCMGs), with a primary focus on handling internal singularities that emerge during momentum steering. These singularities, often caused by specific gimbal configurations and momentum alignment, can lead to actuator saturation and loss of torque authority.

To address this, a high-fidelity nonlinear dynamic model is developed in Simscape Multibody, capturing the coupled motion between the spacecraft and internal actuators. A Lyapunov-based nonlinear controller is implemented to stabilize and track desired attitudes while accounting for actuator dynamics.

Classical steering laws—namely, the pseudo-inverse and null motion methods—are used as benchmarks for generating internal momentum commands. These methods are analytically tractable and exhibit well-understood singularity-avoidance behavior, making them effective for evaluating steering robustness. Additionally, they provide interpretable trajectories that facilitate supervised learning and policy pretraining for adaptive control strategies.

A reinforcement learning (RL)-based steering policy is then introduced to compute gimbal and wheel acceleration commands. The RL policy is pretrained using supervised learning on Monte Carlo trajectories generated by the benchmark laws, and further fine-tuned through interaction with the simulation environment. This approach enables the system to learn implicit singularity avoidance and adapt under practical constraints.

Finally, a dual-gimbal VSCMG hardware prototype is constructed to experimentally validate the proposed methods. The setup includes an aluminum flywheel, servo-driven gimbals, and motorized wheels, and follows a scissored-pair configuration. Experimental results show that the RL-based strategy performs comparably to classical laws in regular conditions while offering improved behavior near singularities and reduced actuator effort.

Keywords: Spacecraft Attitude Control, VSCMG, Singularity Avoidance, Null Motion, Pseudo-Inverse Steering, Reinforcement Learning, Simscape Multibody, Nonlinear Dynamics, Hardware-in-the-Loop.

Chapter 1: Introduction

1.1 Background

Attitude control is fundamental to satellite operation and mission success. The ability of a spacecraft to orient itself precisely in space enables applications ranging from Earth observation and scientific experiments to inter-satellite communication and autonomous docking [1, 2]. Maintaining correct orientation is equally essential for tasks such as optimizing solar array exposure, performing orbital transfers, and ensuring precise payload targeting.

The mathematical modeling of spacecraft rotational motion is based on rigid body dynamics in three-dimensional space. This formulation, derived from classical mechanics and orbital dynamics, treats the spacecraft as a rotating body subjected to internal and external torques. Euler's rotational equations, angular momentum conservation, and coordinate transformations between inertial and body frames are central to this modeling [3, 4]. These foundations support both analytical studies and real-time simulation of spacecraft behavior.

Among modern actuator technologies, Variable Speed Control Moment Gyroscopes (VSCMGs) are particularly attractive for agile attitude maneuvers. By allowing the spin speed of the wheels to vary in addition to gimbal angle control, VSCMGs introduce an extra degree of freedom and enable more robust torque generation. This dual-input control enables singularity avoidance and dynamic torque shaping [5, 6]. However, the nonlinear and coupled nature of their dynamics poses challenges in control synthesis, requiring advanced control strategies and high-fidelity simulations for validation [7].

To meet these modeling demands, simulation environments like MATLAB/Simulink with Simscape have become integral to spacecraft system design. Simscape offers a physical modeling framework that allows engineers to build detailed, multi-domain systems—mechanical, electrical, thermal, and more—using a component-based approach. For spacecraft applications, Simscape enables accurate representation of actuator kinematics, mass properties, friction effects, and environment interaction without simplifying assumptions common in analytical models. This allows the development of hardware-in-the-loop (HIL) simulations that closely emulate real mission conditions, including testing reinforcement learning controllers in closed-loop scenarios [8, 9].

Modern space missions increasingly demand such integration of high-fidelity modeling and advanced control algorithms to ensure robust performance under real-time constraints and mission uncertainty.

1.2 Problem Statement

Satellite attitude control systems are categorized into passive and active methods. Passive techniques utilize environmental forces, such as atmospheric drag, Earth's magnetic field, and the gravity gradient, to provide stabilization. While these are energy-efficient and cost-effective, they lack the agility and precision required for modern spacecraft [52].

Active systems employ actuators such as thrusters, reaction wheels (RWs), and control moment gyroscopes (CMGs) to generate control torques [41, 42, 43]. Each has inherent drawbacks: thrusters consume finite propellant and risk contamination; RWs saturate under high momentum buildup; and CMGs, though efficient, suffer from internal singularities that prevent torque generation in specific directions [44, 45].

Variable Speed Control Moment Gyroscopes (VSCMGs) offer increased flexibility by introducing a degree of freedom through flywheel speed variation. This enables potential singularity avoidance and agile torque generation. However, the nonlinear coupling between gimbal and wheel dynamics introduces significant complexity in control [47].

Traditional steering laws, such as the pseudoinverse method and null motion approaches, are sensitive near singularities. They often rely on precise modeling and fail in uncertain environments [48, 44]. Reinforcement learning (RL), on the other hand, can learn optimal strategies through experience, offering robust singularity avoidance and fault tolerance [49].

Despite these advances, few studies integrate RL with hardware testbeds. Additionally, pyramid-type VSCMGs (like the scissored configuration) introduce further challenges in torque mapping. Simscape Multibody offers a solution by enabling high-fidelity dynamic modeling of actuator systems [53], yet it is underused in spacecraft attitude studies.

There remains a gap in research that comprehensively compares pseudoinverse, null motion, and RL-based steering—both in simulation and hardware—under realistic dynamics and disturbances.

1.3 Objectives

This research aims to develop a singularity-robust, reinforcement learning (RL)-based attitude control system for spacecraft using Variable Speed Control Moment Gyros (VSCMGs), validated in both simulation and hardware environments. The specific objectives are as follows:

1. **Model a full 3D spacecraft system using Simscape Multibody:** Construct a high-fidelity model of spacecraft attitude dynamics incorporating mechanical geometry, realistic inertial properties of the bus and flywheels, and physical representations of joints and actuators. The model includes both reaction torques and gimbal coupling, enabling accurate 3D motion representation for training and control validation.
2. **Train reinforcement learning controllers using Simscape-generated data:** Implement model-free RL algorithms (e.g., PPO or SAC) to learn control policies for VSCMGs. Training data—including states, actuator dynamics, and torque responses—is collected directly from the Simscape model to ensure realism. Post-training, the policies are refined through retraining on trajectory rollouts to improve robustness and precision.
3. **Design and fabricate a hardware prototype with scissored VSCMGs:** Build a physical system with two scissored gimbals, each driven by one servo motor for gimbal rotation and one DC motor for wheel spin. Aluminum flywheels are mounted on a custom frame designed in SolidWorks and manufactured using precision-machined components. The assembly serves as the experimental platform for Hardware-in-the-Loop (HIL) validation.
4. **Replicate the hardware system inside Simscape for co-simulation:** Build a digital twin of the hardware prototype using Simscape, maintaining identical mechanical and actuation parameters. Establish real-time communication between the RL-trained controller in Simulink and the physical hardware to test real-world performance against simulation predictions.
5. **Benchmark and compare control strategies across platforms:** Implement classical steering laws, including pseudoinverse and null motion methods, in both the Simscape model and hardware testbed. Compare them against RL-based control in terms of singularity avoidance, control accuracy, energy efficiency, and real-time feasibility under various mission scenarios such as large-angle slews and target tracking.

1.4 Literature Review

1.4.1 Singularity Avoidance Strategies

A critical challenge in spacecraft attitude control using CMG or VSCMG actuators is the emergence of singularities—configurations in which the system's torque-generation capability becomes degenerate or collapses in one or more directions. To mitigate these singularities, various steering strategies have been proposed:

- **Pseudo-Inverse Steering Laws:** These produce minimum-norm solutions to torque

demand using the Moore–Penrose inverse. However, they fail to account for the local structure of singularities, often resulting in torque infeasibility near degenerate configurations [51].

- **Null Motion Techniques:** Introduced by [48], these methods add a motion component in the null space of the Jacobian to steer away from singularities. While effective for some single-gimbal CMG configurations, they struggle in pyramid-type VSCMG setups due to geometric complexity.
- **Singularity-Robust Steering Laws:** These allow bounded torque errors near singularities, offering a continuous steering solution that prioritizes avoidance over exact tracking [44, 45].
- **Artificial Potential Functions (APFs):** These encode singularities as repulsive fields and formulate actuator trajectories to repel the system from those states. APFs are attractive for smooth motion design but require careful tuning to avoid local minima [50].

While these strategies are insightful, they rely heavily on analytical Jacobians or linearized models. Moreover, they often require explicit identification of singular surfaces and control gains—tasks that are not easily scalable to highly nonlinear, multi-DOF satellite systems.

1.4.2 Machine Learning-Based Steering

In recent years, machine learning (ML) methods have been proposed to overcome limitations of model-based singularity avoidance strategies. For example:

- [46] applied feedforward neural networks to approximate null-motion control, showing improved escape from singularities in 3-CMG systems.
- [49] employed deep learning models and ensemble methods such as random forests to predict control actions and minimize torque error in redundant gimbal systems.

While promising, many ML applications rely on supervised learning alone, requiring a large volume of labeled training data and lacking online adaptability. These methods also depend on pre-generated motion data, which constrains generalization across different mission profiles or failure scenarios.

1.4.3 Integration of Reinforcement Learning

Reinforcement Learning (RL) addresses many of the shortcomings of both classical control and supervised ML approaches. Unlike model-based or static-learning systems, RL agents interact with a simulated environment, learning optimal control strategies through trial and

error.

This thesis adopts the RL framework to compute real-time gimbal rates and wheel accelerations for steering a pyramid-configured VSCMG system. The RL agent is trained using the Proximal Policy Optimization (PPO) algorithm, a policy-gradient method known for its stability and performance in continuous action spaces.

- **Dynamic Singularity Avoidance:** Unlike analytical methods, RL agents implicitly learn singularity-avoiding behavior by maximizing long-term reward, without requiring explicit computation of the Jacobian or its inverse.
- **Adaptive Performance:** The RL-based controller adapts to uncertainties such as unmodeled dynamics, actuator saturation, and mission demands that vary over time, capabilities not easily incorporated into classical control laws.
- **Generalization to Complex Missions:** Once trained, the RL policy generalizes across maneuver types, including target-pointing, docking, and agile slewing—often outperforming conventional steering methods in speed and smoothness.

1.4.4 MATLAB–Simscape Integration for RL Training

To bridge the gap between theoretical learning and physical feasibility, this thesis incorporates a high-fidelity Simscape Multibody model of the spacecraft and actuators within the MATLAB–Simulink environment. This allows direct physics-based simulation of spacecraft dynamics, including nonlinear VSCMG behavior, friction, and saturation effects.

Training is conducted entirely within MATLAB's Reinforcement Learning Toolbox, interfaced with the Simscape model via Simulink blocks. This tightly integrated framework ensures that:

- The agent trains on true multibody dynamics, not simplified surrogate models.
- Actuator limits and failures can be naturally modeled in Simscape.
- The resulting policy is directly deployable in embedded model-based control systems.

Compared to RL environments built using Python or custom C++ engines, the MATLAB–Simscape workflow ensures consistent unit handling, ease of visualization, and compatibility with aerospace-grade validation processes.

Conclusion

This literature review highlights the progression from model-based control to machine learning and ultimately reinforcement learning for spacecraft attitude steering. By leveraging Simscape and PPO-based RL training, this thesis advances a scalable,

model-free, and simulation-aligned control framework that is well-suited to complex VSCMG configurations.

Chapter 2: Attitude Modeling

2.1 Introduction

In spacecraft dynamics and control, the term *attitude* refers to the orientation of a rigid body relative to an inertial frame or another reference frame. Attitude determination and control are fundamental to many space missions, including Earth observation, communication, interplanetary navigation, and docking maneuvers.

Mathematically, attitude can be described by various parameterizations, each with different trade-offs in terms of singularity, minimality, and computational complexity. The four most widely used representations are:

- Direction Cosine Matrix (DCM)
- Euler Angles
- Quaternions (Euler Parameters)
- Modified Rodrigues Parameters (MRPs)

This chapter introduces these representations in increasing sophistication. The Direction Cosine Matrix provides a general framework, Euler angles offer intuitive physical meaning, quaternions eliminate singularities, and MRPs achieve minimality with advantageous control properties. Special emphasis is given to the MRPs for their excellent balance between compactness and computational robustness, making them highly suitable for modern attitude control systems.

In the following sections, we will rigorously define each representation, derive the associated kinematic equations, analyze their limitations, and highlight transformation rules. All derivations, identities, and insights are based on the formalism presented in *Analytical Mechanics of Space Systems* by Schaub and Junkins.

2.1.1 Direction Cosine Matrix (DCM)

The Direction Cosine Matrix (DCM) is a 3×3 orthogonal matrix that describes the orientation of a rigid body frame $\mathcal{B}$ concerning a reference frame $\mathcal{N}$.

Let $\{\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, \hat{\mathbf{n}}_3\}$ be the orthonormal basis vectors of the inertial frame $\mathcal{N}$, and $\{\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2, \hat{\mathbf{b}}_3\}$ be the basis vectors of the body frame $\mathcal{B}$. Then the body-frame unit vectors can be expressed in terms of the inertial-frame basis as:

$$\hat{\mathbf{b}}_1 = \cos\alpha_{11}\hat{\mathbf{n}}_1 + \cos\alpha_{12}\hat{\mathbf{n}}_2 + \cos\alpha_{13}\hat{\mathbf{n}}_3 \tag{2.1}$$

$$\hat{\mathbf{b}}_2 = \cos\alpha_{21}\hat{\mathbf{n}}_1 + \cos\alpha_{22}\hat{\mathbf{n}}_2 + \cos\alpha_{23}\hat{\mathbf{n}}_3 \tag{2.2}$$

$$\hat{\mathbf{b}}_3 = \cos\alpha_{31}\hat{\mathbf{n}}_1 + \cos\alpha_{32}\hat{\mathbf{n}}_2 + \cos\alpha_{33}\hat{\mathbf{n}}_3 \tag{2.3}$$

These cosines form the entries of the DCM:

$$[C] = \begin{bmatrix} \cos\alpha_{11} & \cos\alpha_{12} & \cos\alpha_{13} \\ \cos\alpha_{21} & \cos\alpha_{22} & \cos\alpha_{23} \\ \cos\alpha_{31} & \cos\alpha_{32} & \cos\alpha_{33} \end{bmatrix} \tag{2.4}$$

Each element C_{ij} of the matrix can be computed via the dot product:

$$C_{ij} = \cos(\angle(\hat{\mathbf{b}}_i, \hat{\mathbf{n}}_j)) = \hat{\mathbf{b}}_i \cdot \hat{\mathbf{n}}_j \tag{2.5}$$

Orthogonality Properties:

$$[C]^T[C] = [I], \quad [C][C]^T = [I], \quad \det([C]) = +1 \tag{2.6}$$

These properties ensure that $[C]$ represents a proper rotation (no reflection or scaling).

Transformation Usage:

Given a vector $\mathbf{r}_N$ expressed in the inertial frame, the equivalent vector in the body frame is:

$$\mathbf{r}_B = [C_{\mathcal{B}/\mathcal{N}}]\,\mathbf{r}_N \tag{2.7}$$

Likewise, to transform a body-frame vector into the inertial frame:

$$\mathbf{r}_N = [C_{\mathcal{B}/\mathcal{N}}]^T\,\mathbf{r}_B \tag{2.8}$$

Geometric Interpretation:

Each row of the DCM gives the components of a body-frame axis in the inertial frame. Each column gives the components of an inertial axis in the body frame.

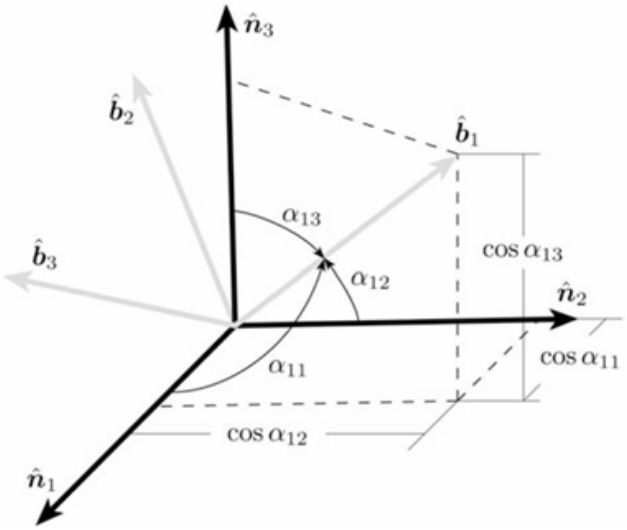


Figure 2.1: Interpretation of DCM as direction angles α_{ij} between body axes $\hat{\mathbf{b}}_i$ and inertial axes $\hat{\mathbf{n}}_j$:contentReference[oaicite:1]index=1.

Advantages:

- Always valid, globally nonsingular representation
- Directly interpretable as axis projections
- Useful for vector transformations

Disadvantages:

- Redundant (9 elements for 3 DOF)
- Requires orthonormalization during integration due to numerical drift

2.1.2 Euler Angles

Euler angles provide a compact and geometrically intuitive method for describing orientation using three sequential rotations about specified body-fixed axes. Each set of Euler angles is denoted by a three-number sequence corresponding to the rotation axes.

Let $\{y_1, y_2, y_3\}$ be the ordered Euler angles about body axes $\{a, b, g\}$, respectively. The overall rotation matrix is constructed by applying each rotation sequentially:

$$[C(y_1, y_2, y_3)] = [M_g(y_3)][M_b(y_2)][M_a(y_1)] \tag{2.9}$$

Each $M_i(y)$ is a basic rotation matrix about an axis i:

$$[M_1(y)] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos y & \sin y \\ 0 & -\sin y & \cos y \end{bmatrix} \tag{2.10}$$

$$[M_2(y)] = \begin{bmatrix} \cos y & 0 & -\sin y \\ 0 & 1 & 0 \\ \sin y & 0 & \cos y \end{bmatrix} \tag{2.11}$$

$$[M_3(y)] = \begin{bmatrix} \cos y & \sin y & 0 \\ -\sin y & \cos y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

Asymmetric Euler Angles – The (3-2-1) Sequence This is the most commonly used sequence in aerospace (yaw–pitch–roll):

$$\{y_1, y_2, y_3\} = \{\psi, \theta, \phi\}$$

The DCM is given by:

$$[C] = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{bmatrix} \tag{2.13}$$

$$\text{where } c_x = \cos x,\ s_x = \sin x$$

The inverse transformation is:

$$\psi = \tan^{-1}\left(\frac{C_{12}}{C_{11}}\right) \tag{2.14}$$

$$\theta = -\sin^{-1}(C_{13}) \tag{2.15}$$

$$\phi = \tan^{-1}\left(\frac{C_{23}}{C_{33}}\right) \tag{2.16}$$

Symmetric Euler Angles – The (3-1-3) Sequence This set is widely used in celestial mechanics:

$$\{y_1, y_2, y_3\} = \{\Omega, i, \omega\}$$

The DCM is given by:

$$[C] = \begin{bmatrix} c_\omega c_\Omega - s_\omega c_i s_\Omega & c_\omega s_\Omega + s_\omega c_i c_\Omega & s_\omega s_i \\ -s_\omega c_\Omega - c_\omega c_i s_\Omega & -s_\omega s_\Omega + c_\omega c_i c_\Omega & c_\omega s_i \\ s_i s_\Omega & -s_i c_\Omega & c_i \end{bmatrix} \tag{2.17}$$

The inverse transformation is:

$$\Omega = \tan^{-1}\left(\frac{C_{31}}{-C_{32}}\right) \tag{2.18}$$

$$i = \cos^{-1}(C_{33}) \tag{2.19}$$

$$\omega = \tan^{-1}\left(\frac{C_{13}}{C_{23}}\right) \tag{2.20}$$

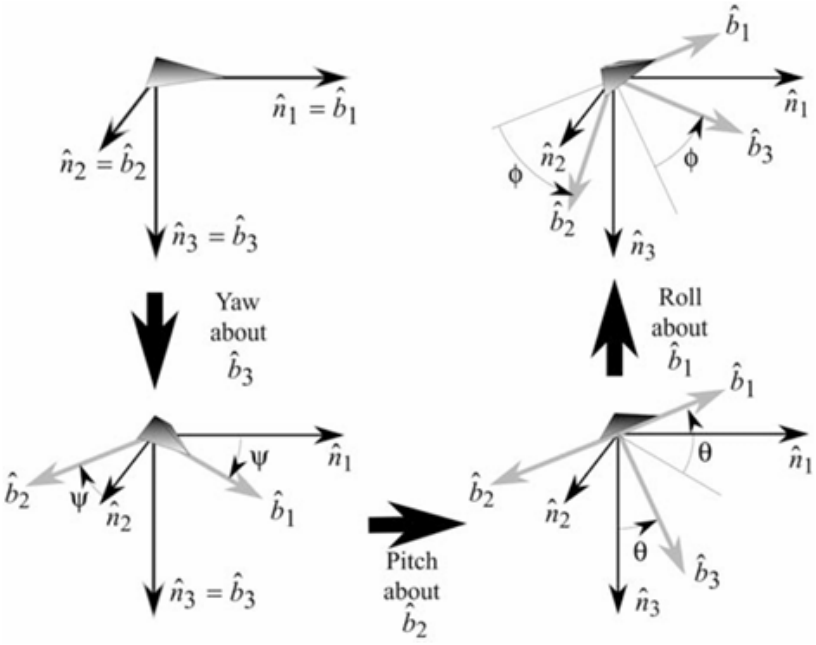


Figure 2.2: Yaw–Pitch–Roll (3-2-1) Euler angle sequence

Important Considerations:

- The order of rotations matters. $(3\text{-}2\text{-}1) \neq (1\text{-}2\text{-}3)$
- Different sequences lead to different singularity conditions:
  - (3-2-1): singular at $\theta = \pm 90°$
  - (3-1-3)singular at $i = 0°$ or $180°$
- These are geometric singularities where two axes align, making two angles indistinguishable

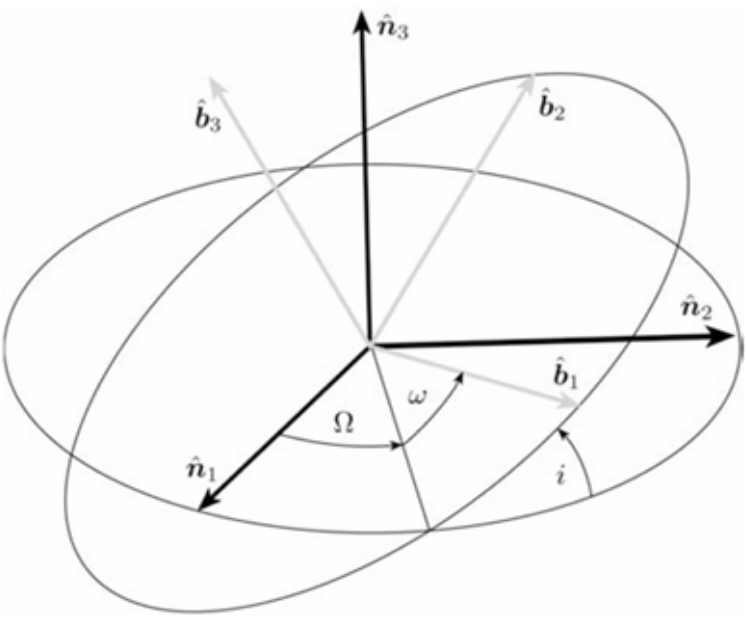


Figure 2.3: Longitude-Inclination-Argument of Perihelion (3-1-3)

Advantages of Euler Angles:

- Compact (only 3 parameters)
- Intuitive, especially for small angular displacements

Disadvantages:

- Geometric singularities (gimbal lock)
- Nonlinear kinematics
- Trigonometric complexity in integration

2.1.3 Quaternions (Euler Parameters)

Principal Rotation Vector and Euler's Rotation Theorem

Instead of using three sequential Euler angle rotations, any arbitrary orientation of a rigid body can be described by a single rotation through a principal angle Φ about a fixed unit axis $\hat{e}$.

Euler's Theorem states:

> A rigid body can be brought from any initial orientation to any final orientation by a single rigid-body rotation about a fixed axis $\hat{e}$ by a principal angle Φ.

This concept is illustrated in Fig. 2.4.

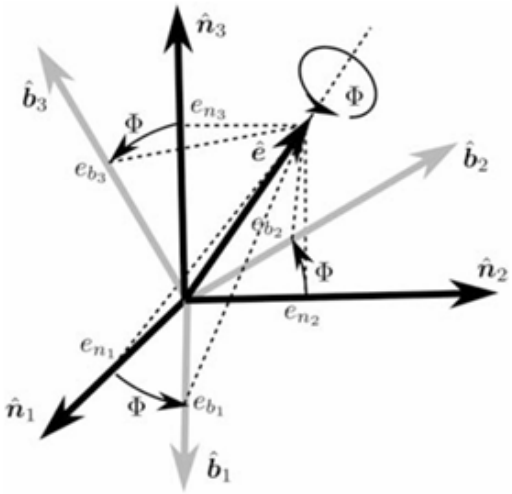


Figure 2.4: Euler's principal rotation theorem: A rigid body rotates from frame $\mathcal{N}$ to $\mathcal{B}$ via a single rotation about axis $\hat{\mathbf{e}}$ by angle Φ

Mathematically, the principal axis $\hat{\mathbf{e}}$ satisfies:

$$[C]\hat{\mathbf{e}} = \hat{\mathbf{e}}, \quad \text{i.e., } \hat{\mathbf{e}} \text{ is the eigenvector of } [C] \text{ with eigenvalue 1} \tag{2.21}$$

The principal rotation vector $\mathbf{g}$ is then defined as:

$$\mathbf{g} = \Phi\hat{\mathbf{e}} \tag{2.22}$$

From this, the direction cosine matrix (DCM) $[C]$ can be expressed as:

$$[C] = \exp([\tilde{\mathbf{g}}]) = I + \frac{\sin\Phi}{\Phi}[\tilde{\mathbf{g}}] + \frac{1-\cos\Phi}{\Phi^2}[\tilde{\mathbf{g}}]^2 \tag{2.23}$$

This rotation is geometrically and computationally more efficient than any Euler sequence, especially for large-angle rotations.

Definition of Quaternions from PRV

Quaternions, or Euler Parameters, encode this rotation using 4 parameters:

$$\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\Phi}{2}) \\ e_1 \sin(\frac{\Phi}{2}) \\ e_2 \sin(\frac{\Phi}{2}) \\ e_3 \sin(\frac{\Phi}{2}) \end{bmatrix} \tag{2.24}$$

This quaternion vector satisfies the unit constraint:

$$\|\mathbf{b}\|^2 = b_0^2 + b_1^2 + b_2^2 + b_3^2 = 1 \tag{2.25}$$

DCM from Quaternion

Using quaternions $\mathbf{b}$, the DCM is obtained by:

$$[C] = (b_0^2 - \mathbf{b}_v^T \mathbf{b}_v) I + 2\mathbf{b}_v \mathbf{b}_v^T + 2b_0[\tilde{\mathbf{b}}_v] \tag{2.26}$$

Or in expanded form:

$$[C] = \begin{bmatrix} b_0^2 + b_1^2 - b_2^2 - b_3^2 & 2(b_1 b_2 + b_0 b_3) & 2(b_1 b_3 - b_0 b_2) \\ 2(b_1 b_2 - b_0 b_3) & b_0^2 - b_1^2 + b_2^2 - b_3^2 & 2(b_2 b_3 + b_0 b_1) \\ 2(b_1 b_3 + b_0 b_2) & 2(b_2 b_3 - b_0 b_1) & b_0^2 - b_1^2 - b_2^2 + b_3^2 \end{bmatrix} \tag{2.27}$$

Properties and Remarks

- Quaternions provide a globally nonsingular representation.
- Computationally efficient: no trigonometric functions in kinematics.
- Redundant representation (4 parameters for 3 DOF).
- Must be renormalized after numerical integration.
- For every orientation $\mathbf{b}$, the negative $-\mathbf{b}$ represents the same rotation.

2.1.4 Modified Rodrigues Parameters (MRP)

Motivation and Definition

The Modified Rodrigues Parameters (MRPs) offer a three-parameter, globally non-redundant representation of attitude with improved numerical behavior near small-angle rotations compared to Euler angles and quaternions. They are particularly advantageous in spacecraft control and estimation applications due to their linearization properties and bounded norm.

The MRP vector $\mathbf{s} = [s_1, s_2, s_3]^T$ is defined from the Euler parameters (quaternions) as:

$$s_i = \frac{b_i}{1 + b_0}, \quad i = 1, 2, 3 \tag{2.28}$$

The inverse transformation is:

$$b_0 = \frac{1 - s^2}{1 + s^2} \tag{2.29}$$

$$b_i = \frac{2s_i}{1 + s^2}, \quad i = 1, 2, 3 \tag{2.30}$$

where $s^2 = \mathbf{s}^T\mathbf{s}$.

Alternatively, MRPs can be directly defined from the principal rotation vector using:

$$\mathbf{s} = \tan\left(\frac{\Phi}{4}\right)\hat{\mathbf{e}} \tag{2.31}$$

Thus, MRPs represent a stereographic projection of the quaternion unit sphere onto the plane orthogonal to the scalar component b_0, as shown in Figure 2.5.

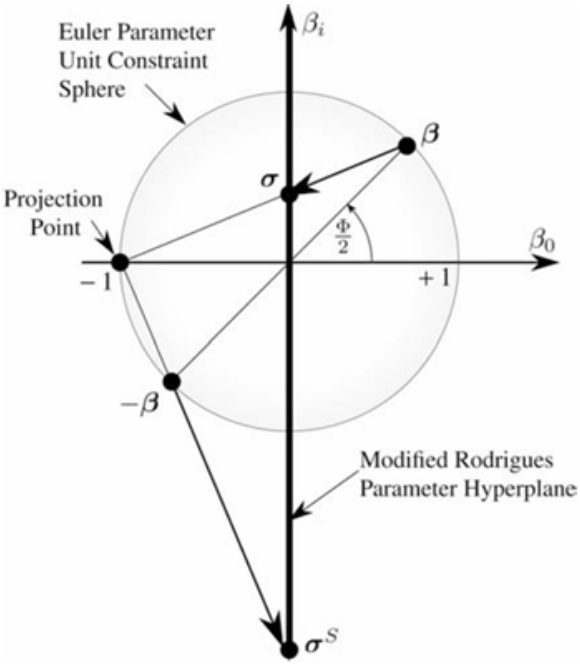


Figure 2.5: Stereographic projection of Euler parameters onto the Modified Rodrigues Parameters (MRP).

Shadow Set of MRPs

Since the transformation from Euler parameters is not unique, an alternate set of MRPs (called the *shadow set*) can be defined:

$$s_{S,i} = \frac{-b_i}{1 - b_0} = \frac{-s_i}{s^2}, \quad i = 1, 2, 3 \tag{2.32}$$

This duality ensures global coverage. As the original set approaches the singularity at

$\Phi = \ pm360°$, the shadow set remains well-behaved. Switching between the two sets at the unit sphere ($s^T s = 1$) keeps the MRP norm bounded:

$$\|\mathbf{s}\| \leq 1 \quad \text{(Shortest rotation, } \Phi \leq 180°\text{)}$$

$$\|\mathbf{s}_S\| > 1 \quad \text{(Longer rotation, } \Phi > 180°\text{)}$$

DCM Representation from MRPs

The DCM $[C]$ can be obtained from MRPs as:

$$[C] = I + \frac{8[\tilde{s}]^2 - 4(1 - s^2)[\tilde{s}]}{(1 + s^2)^2} \tag{2.33}$$

This expression is derived from substituting the MRP definition into the quaternion-based DCM and simplifying.

Why MRPs Are Chosen in This Thesis

Among all attitude parameterizations, Modified Rodrigues Parameters (MRPs) are particularly well-suited for spacecraft dynamics and control applications due to the following reasons:

- They are **minimal** (only 3 parameters) like Euler angles, but **globally nonsingular** when using the shadow set switching strategy.
- Their **bounded norm** ensures numerical stability during integration and simplifies controller gain tuning in feedback systems.
- MRPs **linearize well** for small-angle dynamics and **preserve the shortest rotation path**, which reduces the control effort during large-angle maneuvers.
- The MRP-based kinematic differential equations are purely **quadratic in state**, which makes them numerically efficient and well-conditioned.
- The use of a **shadow set** and switching $\mathbf{s}^T\mathbf{s} = 1$ allows continuous, smooth, and complete attitude coverage without discontinuity or gimbal lock.

Therefore, MRPs are adopted in this thesis as the **primary attitude representation** for both numerical simulation and control algorithm development.

2.2 Kinematic Equation for Direction Cosine Matrix (DCM)

The kinematic relationship of a rigid body describes how its orientation changes over time. Specifically, it relates the angular velocity vector of the body frame concerning the inertial frame to the time rate of change of the attitude representation. This section derives the differential equation governing the time evolution of the direction cosine matrix (DCM) $[C]$, using both vector algebra and linear algebra arguments.

Angular Velocity Representation in the Body Frame

The angular velocity vector of the body frame $\mathcal{B}$ relative to the inertial frame $\mathcal{N}$ can be expressed in the body frame as:

$$\boldsymbol{\omega}_{B/N} = \omega_1 \hat{\mathbf{b}}_1 + \omega_2 \hat{\mathbf{b}}_2 + \omega_3 \hat{\mathbf{b}}_3 \tag{2.34}$$

Transport Theorem Applied to the DCM

Let $[C] = [BN]$ be the DCM mapping vectors from $\mathcal{N}$ to $\mathcal{B}$. Let $\hat{\mathbf{b}}_i$ denote the ith unit vector of the body frame expressed in the inertial frame.

Using the transport theorem:

$$\left.\frac{d\hat{\mathbf{b}}_i}{dt}\right|_{\mathcal{N}} = \left.\frac{d\hat{\mathbf{b}}_i}{dt}\right|_{\mathcal{B}} + \boldsymbol{\omega}_{B/N} \times \hat{\mathbf{b}}_i \tag{2.35}$$

Because $\hat{\mathbf{b}}_i$ it is fixed in the body frame:

$$\left.\frac{d\hat{\mathbf{b}}_i}{dt}\right|_{\mathcal{B}} = 0$$

Therefore:

$$\left.\frac{d\hat{\mathbf{b}}_i}{dt}\right|_{\mathcal{N}} = \boldsymbol{\omega}_{B/N} \times \hat{\mathbf{b}}_i \tag{2.36}$$

Matrix Form of the DCM Derivative

The DCM is defined by:

$$[C] = \begin{bmatrix} \hat{\mathbf{b}}_1^T \\ \hat{\mathbf{b}}_2^T \\ \hat{\mathbf{b}}_3^T \end{bmatrix}$$

Taking the time derivative:

$$[\dot{C}] = \begin{bmatrix} \frac{d\hat{\mathbf{b}}_1^T}{dt} \Big|_{\mathcal{N}} \\ \frac{d\hat{\mathbf{b}}_2^T}{dt} \Big|_{\mathcal{N}} \\ \frac{d\hat{\mathbf{b}}_3^T}{dt} \Big|_{\mathcal{N}} \end{bmatrix} = \begin{bmatrix} (\boldsymbol{\omega} \times \hat{\mathbf{b}}_1)^T \\ (\boldsymbol{\omega} \times \hat{\mathbf{b}}_2)^T \\ (\boldsymbol{\omega} \times \hat{\mathbf{b}}_3)^T \end{bmatrix} = -[\tilde{\boldsymbol{\omega}}][C]$$

Hence, the DCM satisfies the matrix differential equation:

$$[\dot{C}] = -[\tilde{\boldsymbol{\omega}}][C] \tag{2.37}$$

Where $[\tilde{\boldsymbol{\omega}}]$ is the skew-symmetric matrix of $\boldsymbol{\omega}$:

$$[\tilde{\boldsymbol{\omega}}] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{2.38}$$

Remarks

- Equation (2.37) is **linear** $[C]$ and **singularity-free**.
- While globally valid, it involves 9 variables with 6 constraints (orthonormality of $[C]$), making it computationally redundant.
- This formulation is particularly useful in numerical integration schemes that require exact conservation of orthogonality (e.g., via re-orthonormalization).

Kinematic Equation of Modified Rodrigues Parameters (MRP)

The time evolution of the Modified Rodrigues Parameters (MRP) vector $\mathbf{s} \in \mathbb{R}^3$ under body angular velocity $\boldsymbol{\omega}$ is governed by a nonlinear differential equation. This relation is derived by differentiating the MRP definition from quaternions and applying the quaternion kinematics:

$$\mathbf{s} = \frac{\mathbf{b}}{1 + b_0} \tag{2.39}$$

$$\dot{\mathbf{s}} = \frac{1}{4}\left[(1 - s^2)\mathbf{I} + 2[\tilde{\mathbf{s}}] + 2\mathbf{s}\mathbf{s}^T\right]\boldsymbol{\omega} = \frac{1}{4}\mathbf{B}(\mathbf{s})\,\boldsymbol{\omega} \tag{2.40}$$

Where $[\tilde{\mathbf{s}}]$ is the skew-symmetric matrix of $\mathbf{s}$, and $\mathbf{B}(\mathbf{s})$ is the MRP Jacobian matrix.

This equation is:

- Free of trigonometric functions.
- Quadratic in $\mathbf{s}$ and linear in $\boldsymbol{\omega}$.
- Globally nonsingular when the shadow set is used.

To avoid singularities when $\|\mathbf{s}\| > 1$, a shadow set $\mathbf{s}_S = -\mathbf{s}/\|\mathbf{s}\|^2$ is used with its own differential form:

$$\dot{\mathbf{s}}_S = -\frac{1}{s^2}\left[\mathbf{I} - \frac{1+s^2}{2s^2}\mathbf{s}\mathbf{s}^T\right]\dot{\mathbf{s}} \tag{2.41}$$

The kinematic model in Eq. (2.40) is widely used in attitude propagation and feedback control due to its stability, minimality, and ease of integration.

2.3 Dynamics of a Spacecraft with VSCMGs

2.3.1 Single VSCMG Dynamics

A Variable Speed Control Moment Gyroscope (VSCMG) consists of a spinning flywheel mounted within a gimbal. It offers two controllable degrees of freedom: the gimbal angle γ and the wheel spin rate Ω. The gimbal frame G is defined by a right-handed orthonormal triad $\{\hat{g}_s, \hat{g}_t, \hat{g}_g\}$, representing the spin, transverse, and gimbal axes, respectively.

Let $\boldsymbol{\omega} \in \mathbb{R}^3$ be the angular velocity of the spacecraft body frame B concerning an inertial frame N. The angular velocity of the gimbal frame G relative to B is:

$$\boldsymbol{\omega}_{G/B} = \dot{\gamma}\hat{g}_g \tag{2.42}$$

The reaction wheel spins about $\hat{g}_s$ with a rate Ω, and its frame rotates relative to G as:

$$\boldsymbol{\omega}_{W/G} = \Omega\hat{g}_s \tag{2.43}$$

The total angular momentum $\boldsymbol{H}$ of the system is:

$$\boldsymbol{H} = \boldsymbol{H}_B + \boldsymbol{H}_G + \boldsymbol{H}_W \tag{2.44}$$

Where:

$$\boldsymbol{H}_B = [I_s]\boldsymbol{\omega} \tag{2.45}$$

$$\boldsymbol{H}_G = \left(J_s\hat{g}_s\hat{g}_s^T + J_t\hat{g}_t\hat{g}_t^T + J_g\hat{g}_g\hat{g}_g^T\right)\boldsymbol{\omega} + J_g\dot{\gamma}\hat{g}_g \tag{2.46}$$

$$\boldsymbol{H}_W = (I_{Ws}(\omega_s + \Omega)\hat{g}_s + I_{Wt}\omega_t\hat{g}_t + I_{Wt}(\omega_g + \dot{\gamma})\hat{g}_g) \tag{2.47}$$

Projecting $\boldsymbol{\omega}$ onto the local axes:

$$\omega_s = \hat{g}_s^T \boldsymbol{\omega}, \quad \omega_t = \hat{g}_t^T \boldsymbol{\omega}, \quad \omega_g = \hat{g}_g^T \boldsymbol{\omega} \tag{2.48}$$

The total inertia of the spacecraft is:

$$[I] = [I_s] + [J], \quad \text{where} \quad [J] = J_s \hat{g}_s \hat{g}_s^T + J_t \hat{g}_t \hat{g}_t^T + J_g \hat{g}_g \hat{g}_g^T \tag{2.49}$$

The dynamics of a single VSCMG unit can be derived from Euler's equation:

$$\begin{aligned}[I]\dot{\boldsymbol{\omega}} = -[\tilde{\boldsymbol{\omega}}][I]\boldsymbol{\omega} &- \hat{g}_s \left(J_s(\dot{\Omega} + \dot{\gamma}\omega_t) - (J_t - J_g)\omega_t \dot{\gamma} \right) \\ &- \hat{g}_t \left(J_s(\Omega + \omega_s)\dot{\gamma} - (J_t + J_g)\omega_s \dot{\gamma} + J_s \Omega \omega_g \right) \\ &- \hat{g}_g \left(J_g \ddot{\gamma} - J_s \Omega \omega_t \right) + \boldsymbol{L}\end{aligned} \tag{2.50}$$

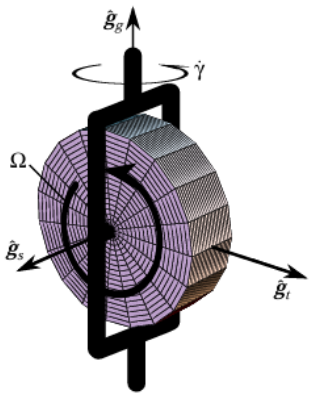


Figure 2.6: Single VSCMG with spin ($\hat{g}_s$), transverse ($\hat{g}_t$), and gimbal ($\hat{g}_g$) axes.

2.4 Multi-VSCMG Configuration: Pyramid Arrangement

For full three-axis control, a pyramid configuration with four VSCMGs is (Fig. 2.7) typically employed. Each unit is mounted symmetrically with its gimbal axis tilted from the body axis by angle $\beta = 54.73°$, minimizing control singularities and optimizing torque authority.

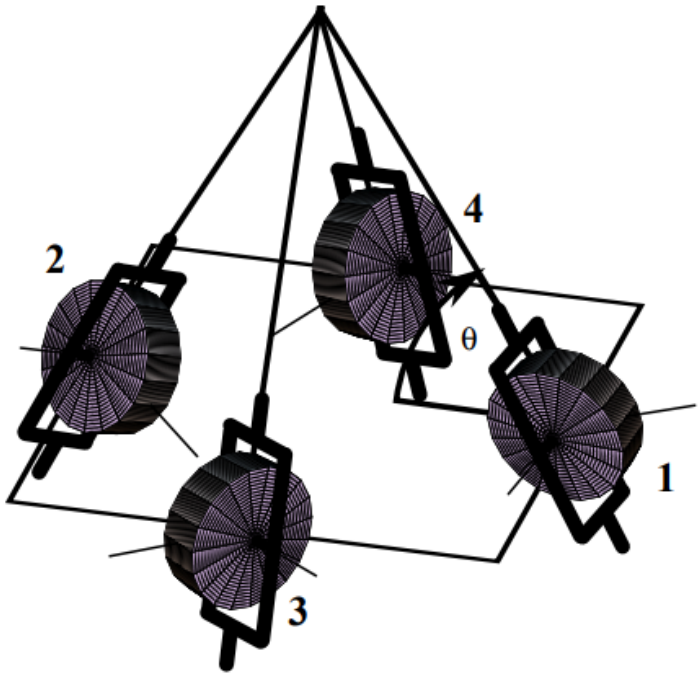


Figure 2.7: Pyramid configuration of four VSCMGs for 3D torque authority

2.5 Full Equations of Motion for 4-VSCMG System

Define direction matrices:

$$[G_s] = [\hat{g}_{s1}, \hat{g}_{s2}, \hat{g}_{s3}, \hat{g}_{s4}] \tag{2.51}$$

$$[G_t] = [\hat{g}_{t1}, \hat{g}_{t2}, \hat{g}_{t3}, \hat{g}_{t4}] \tag{2.52}$$

$$[G_g] = [\hat{g}_{g1}, \hat{g}_{g2}, \hat{g}_{g3}, \hat{g}_{g4}] \tag{2.53}$$

The total inertia is:

$$[I] = [I_s] + \sum_{i=1}^{4} \left(J_{si} \hat{g}_{si} \hat{g}_{si}^T + J_{ti} \hat{g}_{ti} \hat{g}_{ti}^T + J_{gi} \hat{g}_{gi} \hat{g}_{gi}^T \right) \tag{2.54}$$

Define actuator torques as:

$$\tau_{s,i} = J_{si}(\dot{\Omega}_i + \dot{\gamma}_i \omega_{ti}) - (J_{ti} - J_{gi}) \omega_{ti} \dot{\gamma}_i \tag{2.55}$$

$$\tau_{t,i} = J_{si}(\Omega_i + \omega_{si}) \dot{\gamma}_i - (J_{ti} + J_{gi}) \omega_{si} \dot{\gamma}_i + J_{si} \Omega_i \omega_{gi} \tag{2.56}$$

$$\tau_{g,i} = J_{gi} \ddot{\gamma}_i - J_{si} \Omega_i \omega_{ti} \tag{2.57}$$

Stack them into vectors:

$$\boldsymbol{\tau}_s = [\tau_{s1}, \tau_{s2}, \tau_{s3}, \tau_{s4}]^T \tag{2.58}$$

$$\boldsymbol{\tau}_t = [\tau_{t1}, \tau_{t2}, \tau_{t3}, \tau_{t4}]^T \tag{2.59}$$

$$\boldsymbol{\tau}_g = [\tau_{g1}, \tau_{g2}, \tau_{g3}, \tau_{g4}]^T \tag{2.60}$$

Final rotational dynamics:

$$[I]\dot{\boldsymbol{\omega}} = -[\tilde{\boldsymbol{\omega}}][I]\boldsymbol{\omega} - [G_s]\boldsymbol{\tau}_s - [G_t]\boldsymbol{\tau}_t - [G_g]\boldsymbol{\tau}_g + \boldsymbol{L} \tag{2.61}$$

2.6 Kinetic Energy Expression

The total rotational kinetic energy is:

$$T = \frac{1}{2}\boldsymbol{\omega}^T[I_s]\boldsymbol{\omega} + \frac{1}{2}\sum_{i=1}^{4}\left[J_{si}(\Omega_i + \omega_{si})^2 + J_{ti}\omega_{ti}^2 + J_{gi}(\omega_{gi} + \dot{\gamma}_i)^2\right] \tag{2.62}$$

Time derivative of kinetic energy:

$$\dot{T} = \sum_{i=1}^{4}(\dot{\gamma}_i u_{gi} + \Omega_i u_{si}) + \boldsymbol{\omega}^T\boldsymbol{L} \tag{2.63}$$

Chapter 3: Feedback Control Law

To ensure global asymptotic stability of spacecraft attitude, a Lyapunov-based feedback control strategy is employed. The Modified Rodrigues Parameters (MRP) are selected for attitude representation due to their compactness and suitability for large-angle maneuvers [3]. The MRP vector component is defined as:

$$\sigma_i = \frac{q_i}{1 + q_0}, \quad i = 1, 2, 3 \tag{3.1}$$

where q_0 and q_i are the scalar and vector components, respectively, of the unit quaternion $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$. This expression maps the quaternion to Modified Rodrigues Parameters (MRPs), which provide a singularity-free representation for large- or small-angle attitude control.

The rigid body rotational dynamics are governed by Euler's equations, and the control torque is formulated to ensure convergence of both the angular velocity and the attitude error. The closed-loop control law is given by [3]:

$$\boldsymbol{L}_r = -\mathbf{K}\boldsymbol{\sigma}_e - [\mathbf{P}]\delta\boldsymbol{\omega} \tag{3.2}$$

Where $\mathbf{K}$ $\mathbf{P}$ are positive-definite gain matrices, $\boldsymbol{\sigma}_e \in \mathbb{R}^3$ the Modified Rodrigues Parameter (MRP) attitude error vector, and $\delta\boldsymbol{\omega} = \boldsymbol{\omega} - \boldsymbol{\omega}_r$ the angular velocity error relative to the reference angular velocity $\boldsymbol{\omega}_r$?

To asymptotically track a predefined reference trajectory, the control torque can be defined as [3]:

$$\boldsymbol{L}_r = -\mathbf{K}\boldsymbol{\sigma}_e - [\mathbf{P}]\delta\boldsymbol{\omega} + [I](\dot{\boldsymbol{\omega}}_r - [\tilde{\boldsymbol{\omega}}]\boldsymbol{\omega}_r) + [\tilde{\boldsymbol{\omega}}][I]\boldsymbol{\omega} \tag{3.3}$$

The MRP attitude error vector $\boldsymbol{\sigma}_e$ is computed using the non-commutative MRP subtraction operation between the current attitude $\boldsymbol{\sigma}$ and the reference attitude $\boldsymbol{\sigma}_r$. Following the standard formulation [3], it is given by:

$$\boldsymbol{\sigma}_e = \frac{(1 - \boldsymbol{\sigma}_r^\top \boldsymbol{\sigma}_r)\boldsymbol{\sigma} - (1 - \boldsymbol{\sigma}^\top \boldsymbol{\sigma})\boldsymbol{\sigma}_r + 2\,\boldsymbol{\sigma} \times \boldsymbol{\sigma}_r}{1 + \boldsymbol{\sigma}^\top \boldsymbol{\sigma}\,\boldsymbol{\sigma}_r^\top \boldsymbol{\sigma}_r + 2\,\boldsymbol{\sigma}^\top \boldsymbol{\sigma}_r} \tag{3.4}$$

3.1 Velocity-Based Steering Law for VSCMGs

The Velocity-Based steering law is used for Variable Speed Control Moment Gyros (VSCMGs) to compute the actuator velocity required to track the control torque. This approach enables precise control allocation and is compatible with a Simscape-based model. The required control torque can be expressed in simplified form based on [**?**, **?**, 32, 33] as:

$$[D_0]\dot{\boldsymbol{\Omega}} + [D]\dot{\boldsymbol{\delta}} = \boldsymbol{L}_r \tag{3.5}$$

where $\boldsymbol{L}_r$ is the desired control torque, and the matrices $[D_0]$ $[D]$ are defined as:

$$[D_0] = [\hat{g}_1 J_{s_1}, \ldots, \hat{g}_4 J_{s_4}] \tag{3.6}$$

$$[D] = [\hat{g}_1 J_{s_1}(\Omega_1 + \omega_{s_1}), \ldots, \hat{g}_4 J_{s_4}(\Omega_4 + \omega_{s_4})] \tag{3.7}$$

$$\tag{3.8}$$

Let the combined 8×1 actuator state vector rate $\dot{\boldsymbol{\eta}}$ $[Q]$ be:

$$\dot{\boldsymbol{\eta}} = \begin{bmatrix} \dot{\boldsymbol{\Omega}} \\ \dot{\boldsymbol{\delta}} \end{bmatrix}, \quad [Q] = [D_0 : D] \tag{3.9}$$

Since $[Q]$ it is not generally of full rank—due to factors such as actuator alignment constraints, gimbal geometry, or underactuation in certain configurations—a weighted Moore–Penrose pseudo-inverse is employed to obtain a minimum-norm solution [10]. Define the diagonal weight matrix:

$$[W] = \mathrm{diag}(W_{s_1}, \ldots, W_{s_N}, W_{\delta_1}, \ldots, W_{\delta_N}) \tag{3.10}$$

where W_{s_i} and W_{δ_i} are the weights associated with how nearly the VSCMGs behave like conventional reaction wheels or control moment gyros, respectively, as described in [3] and will be further detailed in the upcoming section.

The resulting gimbal speeds and wheel accelerations After some algebraic manipulation, are computed as:

$$\begin{bmatrix} \dot{\boldsymbol{\Omega}} \\ \dot{\boldsymbol{\delta}} \end{bmatrix} = \left(W Q^T \left(Q W Q^T \right)^{-1} \boldsymbol{L}_r \right) \tag{3.11}$$

In practical implementation, the variables $\dot{\delta}$ $\dot{\Omega}$ are computed in MATLAB/Simulink and then provided to Simscape models to generate the corresponding physical torques, ensuring accurate simulation and compliance with the commanded control torque.

3.1.1 Singularity Measure and Steering Strategy

To assess the proximity of a gimbal configuration to a singular CMG configuration, a nondimensional scalar factor d is defined as [10]:

$$d = \det([D][D]^T) \tag{3.12}$$

As the gimbals approach a singular configuration, this scalar tends toward zero, indicating a potential loss of independent torque authority due to geometric alignment issues [10, 32].

To manage control authority near singularities, adaptive weighting factors W_{s_i} are defined as:

$$W_{s_i} = W_{s_i}^0 e^{-\mu d} \tag{3.13}$$

where $W_{s_i}^0$ is a nominal gain and $\mu > 0$ is a tuning parameter. These weights are applied to the reaction wheel acceleration components in the control allocation cost function.

As $d \to 0$, the exponential term approaches unity, restoring the full weight of the reaction wheel contribution in the control allocation. Conversely, when the system is far from singularity (i.e., d large), W_{s_i} it becomes small, favoring the use of gimbal motion. This dynamic adjustment ensures that, near singularities, the control system transitions to rely more heavily on the reaction wheel mode, which remains effective even when the gimbal geometry is degenerate [10, 31, 32, 33].

These adaptive weights are incorporated into the diagonal matrix $[W]$ used in the weighted pseudo-inverse control allocation of Equation (3.10).

The overall control architecture operates by first receiving an error signal, which is then processed by a control law. This control law generates a required steering torque that is fed into a dedicated steering logic component. The steering logic, in turn, computes the corresponding gimbal and wheel acceleration commands. These commands are then sent to a Variable Speed Control Moment Gyro (VSCMG) system, which interacts with the platform dynamics of the spacecraft. The VSCMG system also provides state feedback (including angular velocity ω and momentum σ) back to the control loop, ensuring continuous adjustment and stable operation.

3.1.2 Null Motion Steering Law Implementation in Simscape

In this study, the null motion reconfiguration law in [10] is used as a benchmark steering strategy for Variable Speed Control Moment Gyroscope (VSCMG) clusters. It allows internal redistribution of momentum through gimbal and wheel motion without altering the net control torque. The law is integrated within a Simscape Multibody model to simulate realistic actuator coupling.

The reason for implementing this steering strategy is that certain initial gimbal configurations, particularly in symmetric VSCMG pyramids, can lead to internal singularities depending on the control torque and momentum direction. As noted in [34], such configurations reduce control effectiveness. In these cases, the system can try to escape the singularity by increasing the acceleration of the wheel; however, the presence of a spin rate saturation limits the system's ability to recover, highlighting the importance of robust steering in design.

The final expression for the VSCMG null motion control law is given by:

$$\dot{\boldsymbol{\eta}} = k_e \left[[I]_{4\times4} - WQ^T \left(QWQ^T \right)^{-1} Q \right] [A] \begin{bmatrix} \boldsymbol{\Omega}_f - \boldsymbol{\Omega} \\ \boldsymbol{\delta}_f - \boldsymbol{\delta} \end{bmatrix} \tag{3.14}$$

Where $[I]_{4\times4}$ is the 4×4 identity matrix? and k_e is a positive scalar gain controlling the convergence rate. The matrix $[A]$ is defined as:

$$[A] = \begin{bmatrix} a_{\text{RW}}[I]_{4\times4} & 0 \\ 0 & a_{\text{CMG}}[I]_{4\times4} \end{bmatrix} \tag{3.15}$$

The weights a_{RW} and a_{CMG} balance the tracking of reaction wheel speeds $\boldsymbol{\Omega}$ and gimbal angles $\boldsymbol{\delta}$; setting either to zero removes its influence in the null motion solution.

Chapter 4: Singularity

4.1 Introduction

In Control Moment Gyroscope (CMG) systems, a **singularity** is a configuration where the torque-generation capability of the system becomes insufficient or completely lost in certain directions. This occurs when the torque Jacobian matrix becomes rank-deficient, preventing the system from achieving a full range of output torques.

CMG singularities are generally categorized into:

- **Internal singularities**: Occur inside the momentum envelope where certain directions cannot be torqued due to the gimbals aligning in a problematic configuration.
- **External singularities**: Occur at the boundary of the momentum envelope, typically associated with momentum saturation.

Understanding and avoiding singularities is critical in spacecraft attitude control, especially for agile maneuvering using CMGs.

4.2 Torque Mapping and Singularity Condition

Let $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, ..., \gamma_n]^T$ be the vector of gimbal angles. Each CMG has:

- a unit gimbal axis vector: $\hat{\mathbf{g}}_k$
- a unit spin axis vector: $\hat{\mathbf{s}}_k$
- a torque direction vector: $\hat{\mathbf{t}}_k = \hat{\mathbf{s}}_k \times \hat{\mathbf{g}}_k$

The total torque generated by an array of n CMGs is:

$$\boldsymbol{\tau} = J_s \Omega \sum_{k=1}^{n} \dot{\gamma}_k \, \hat{\mathbf{t}}_k \tag{4.1}$$

Stacking the torque vectors:

$$G_t(\gamma) = \begin{bmatrix} \hat{\mathbf{t}}_1 & \hat{\mathbf{t}}_2 & \cdots & \hat{\mathbf{t}}_n \end{bmatrix} \tag{4.2}$$

Then:

$$\boldsymbol{\tau} = C\,\dot{\boldsymbol{\gamma}}, \quad \text{where } C = J_s \Omega G_t(\gamma) \tag{4.3}$$

A singularity occurs when $\text{rank}(C) < 3$. In such cases, torque cannot be produced in all directions.

To recover gimbal rates:

$$\dot{\boldsymbol{\gamma}} = C^T (CC^T)^{-1} \boldsymbol{\tau} \tag{4.4}$$

But if $\det(CC^T) = 0$this inversion fails — indicating a singular state.

4.3 Analytical Construction of Singular States

Let $\hat{\mathbf{u}}$ be a unit vector representing a **singular direction** (i.e., one in which torque cannot be generated). Then, for each CMG, define the torque and spin axis vectors as:

$$\hat{\mathbf{t}}_k = \varepsilon_k \cdot \frac{\hat{\mathbf{g}}_k \times \hat{\mathbf{u}}}{\|\hat{\mathbf{g}}_k \times \hat{\mathbf{u}}\|} \tag{3.1}$$

$$\hat{\mathbf{s}}_k = \hat{\mathbf{t}}_k \times \hat{\mathbf{g}}_k = \varepsilon_k \cdot \frac{(\hat{\mathbf{g}}_k \times \hat{\mathbf{u}}) \times \hat{\mathbf{g}}_k}{\|\hat{\mathbf{g}}_k \times \hat{\mathbf{u}}\|} \tag{3.2}$$

Here, ε_k determines the sign direction of the k-th CMG's momentum contribution:

$$\varepsilon_k = \text{sign}(\hat{\mathbf{s}}_k \cdot \hat{\mathbf{u}}) \tag{4.5}$$

This ensures all generated momentum vectors lie in the plane orthogonal to $\hat{\mathbf{u}}$.

The net momentum in the singular direction is:

$$H_u = \sum_{k=1}^{n} \hat{\mathbf{s}}_k = \sum_{k=1}^{n} \varepsilon_k \cdot \frac{(\hat{\mathbf{g}}_k \times \hat{\mathbf{u}}) \times \hat{\mathbf{g}}_k}{\|\hat{\mathbf{g}}_k \times \hat{\mathbf{u}}\|} \tag{3.3}$$

The complete singular surface is generated by evaluating Eq. (3.3) over all directions $\hat{\mathbf{u}} \in S^2$ and for all 2^n sign combinations of ε_k.

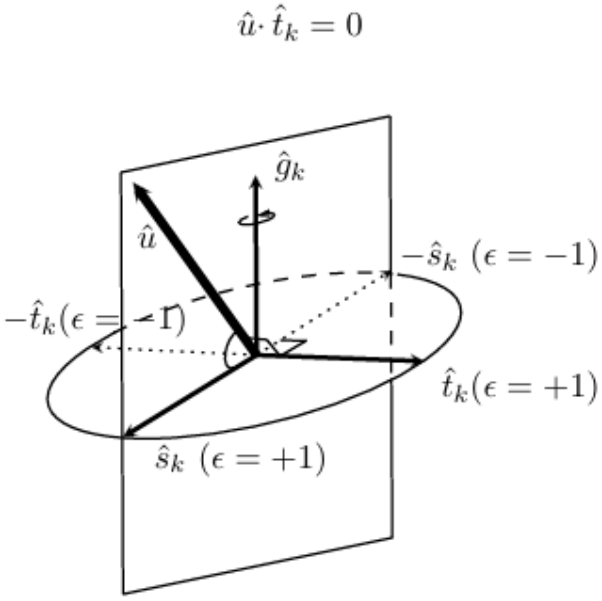


Figure 4.1: Geometric construction of torque vector $\hat{\mathbf{t}}_k$ and spin vector $\hat{\mathbf{s}}_k$ from gimbal axis $\hat{\mathbf{g}}_k$ and singular direction $\hat{\mathbf{u}}$.

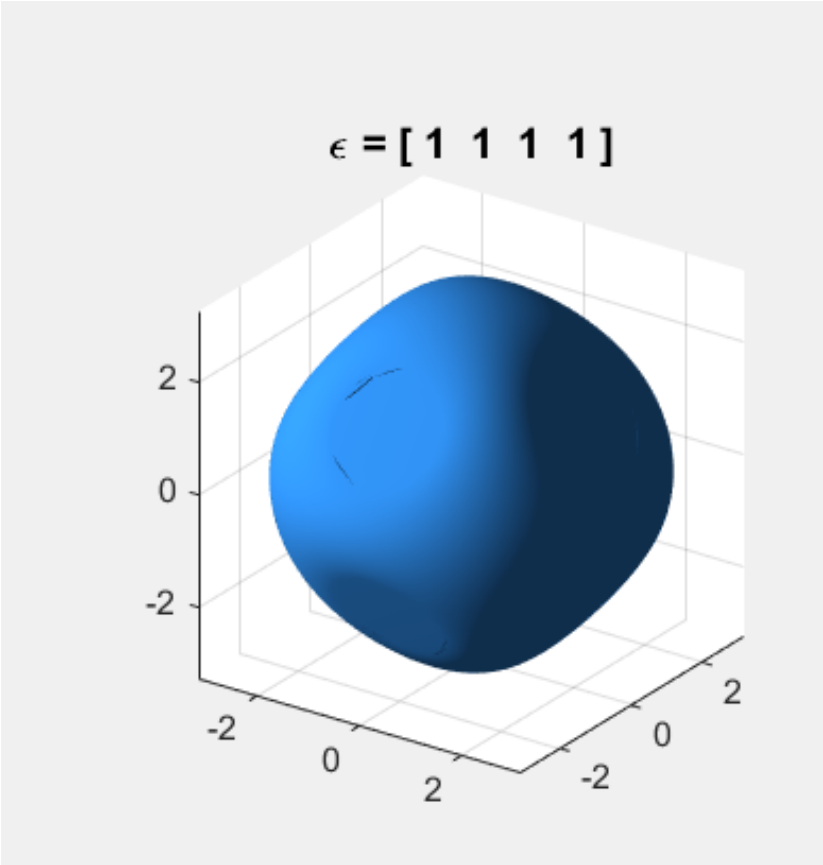


Figure 4.2: Singular surface computed using MATLAB for pyramid CMG configuration. External boundary where full torque cannot be generated.

4.4 Pyramid Configuration and Simulation Setup

To study the singularities in a practical configuration, we consider a 4-CMG pyramid setup with a skew angle $\beta = 54.73°$. This configuration ensures near-spherical momentum

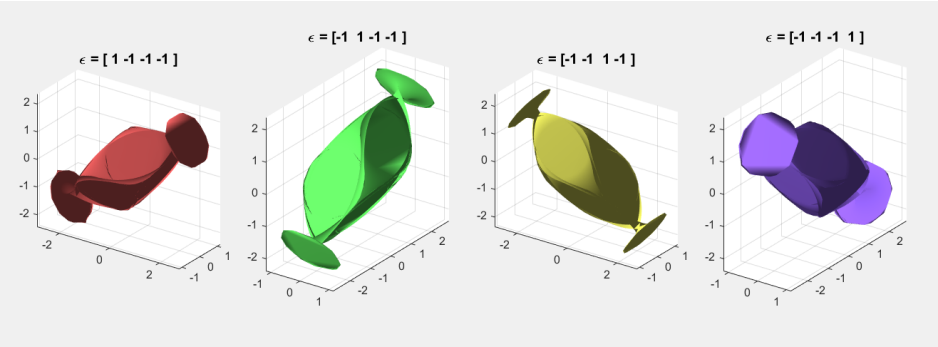


Figure 4.3: Internal singular surfaces were computed for four different ε_k patterns. Each color corresponds to one singular configuration.

generation and symmetric actuation.

Using a custom MATLAB script, we simulated both:

- **External singularity surface**: computed by evaluating the boundary of the reachable torque space.
- **Internal singularity regions**: computed for individual sign combinations of ε_k.

Each point on the surface represents a direction $\hat{\mathbf{u}}$ where the system becomes torque-deficient.

Chapter 5: Mechanical Design

Control Moment Gyroscopes (CMGs) are momentum exchange devices used for precise attitude control in spacecraft, satellites, and robotic platforms. Unlike reaction wheels that rely on changing the rotational speed, CMGs generate torque by altering the orientation of the angular momentum vector of a spinning rotor. This mechanism allows CMGs to deliver a higher torque efficiency, which is especially crucial in agile or mass-constrained systems.[17]

Among the various CMG configurations, the Scissored Pair CMG presents a unique mechanical coupling in which two counter-rotating flywheels are interconnected via a shared or constrained gimbal axis. This structure ensures that the gimbal angles maintain equal magnitudes but opposite signs, effectively enforcing symmetrical torque generation about a single axis. Historically, scissored pairs have been used in early stabilization platforms, such as Brennan's monorail, and later in advanced aerospace systems, including NASA's Skylab-era Astronaut Maneuvering Unit.[16]

Unlike a conventional Single-Gimbal CMG (SGCMG), which may suffer from torque singularities (situations in which the desired torque vectors are unachievable), a scissored pair offers singularity-free torque generation within its linear envelope. This configuration also enhances torque amplification, reduces actuator redundancy, and simplifies the control law design owing to its predictable angular momentum vector.

This chapter focuses on the mechanical design of a Scissored Pair CMG optimized for lightweight, high-agility applications, such as autonomous robotic platforms or CubeSats. It emphasizes:

Structural integrity of the frame under dynamic loads.

Power-efficient torque generation through geared symmetries.

Vibration isolation and high-frequency natural mode.

CAD and simulation-driven design validation.

Figures 5.1and **??** illustrate the scissored and inclined configurations of the proposed CMG prototype, respectively.

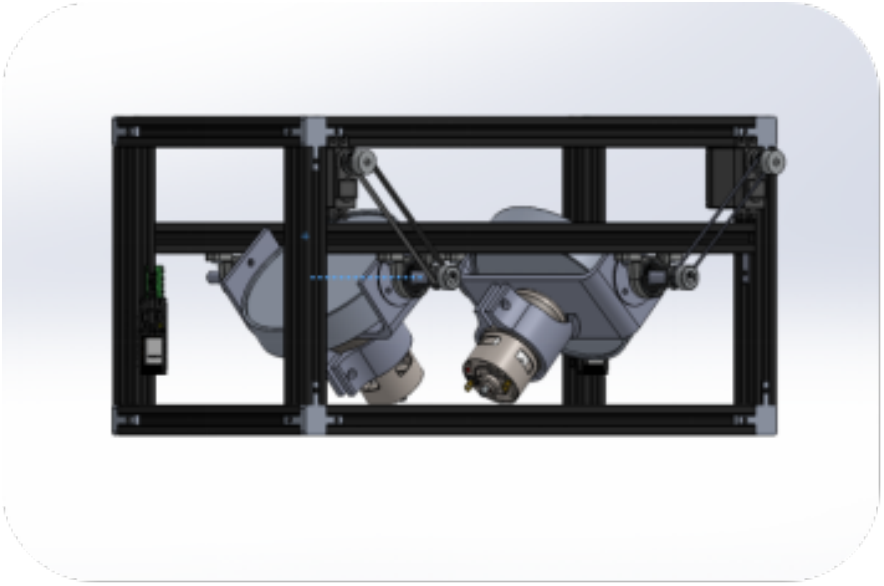


Figure 5.1: The scissored paired configuration

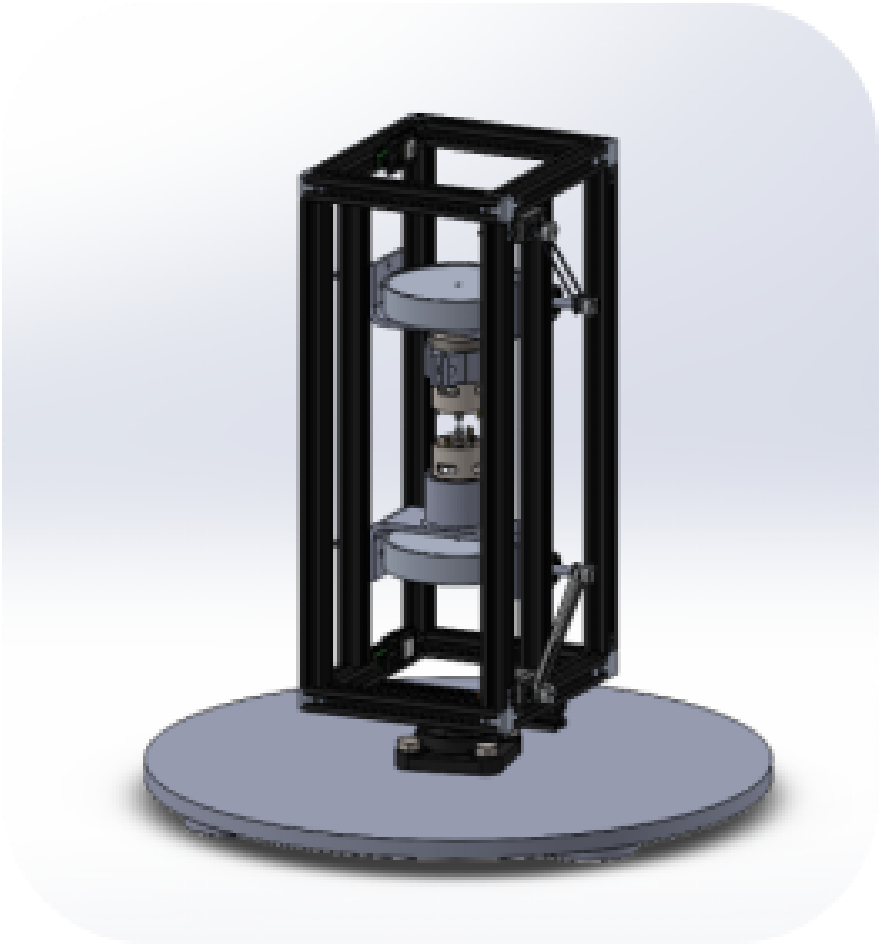


Figure 5.2: Inclined dynamic orientation mounted on the base plate

5.1 CAD Modeling

This section elaborates on the principal components of the Scissored Pair CMG prototype. The system integrates a precision flywheel, dual-gimbal linkage mechanism, and structural frame designed for lightweight modular integration in robotics or satellite payloads.

Each mechanical subsystem was modeled in SolidWorks 2020 SP5 and validated through physical reasoning and empirical best practices in aerospace mechanical design. The referenced figures align with the assembly images shown in Figure **??** .

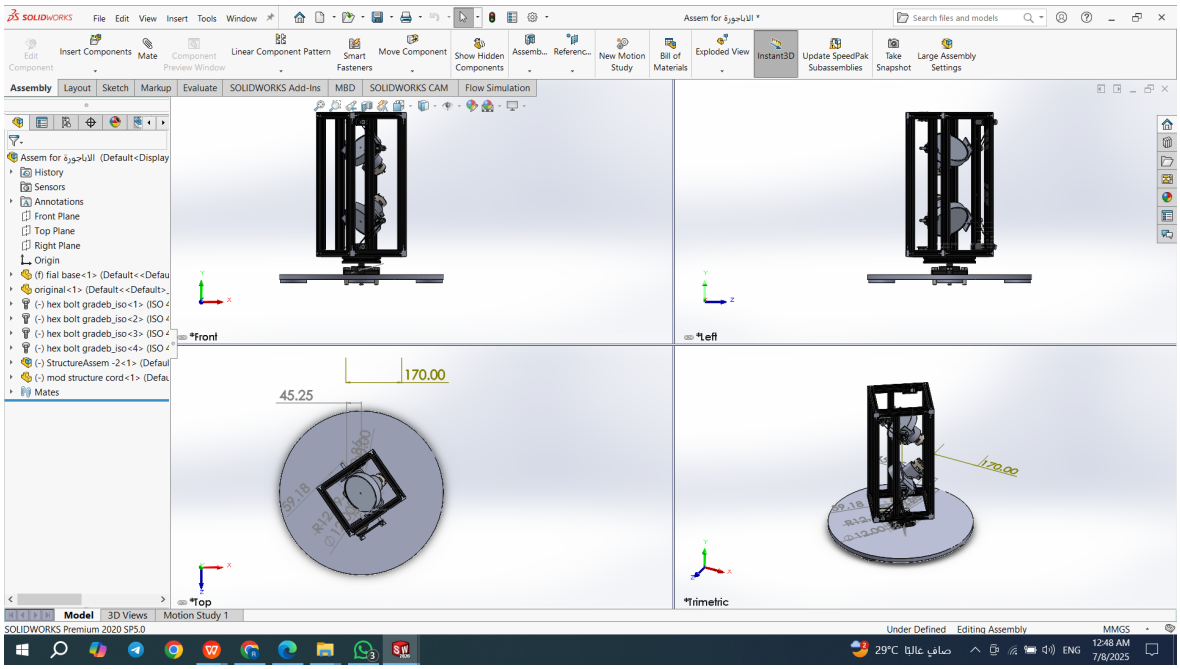


Figure 5.3: CAD design using Solidworks

Scissor Mechanism

Kinematic Architecture

The scissored configuration was achieved using two identical gimbal assemblies coupled through a pulley-driven mechanism with a shared motor shaft. A belt-and-pulley drive (GT2-20T and GT2 5 mm belt) translates the motor rotation to the synchronized gimbal deflection

For the current design, the pulley radii are equal.[28]

This geometric coupling enforces a mirrored gimbal angle between the flywheels, thus generating symmetric net-effective torque on the spacecraft body.

Servo Selection

The system employs an FT5835M servo motor, which was selected for its high torque and

feedback resolution.

Parameter	Value
Rated Torque	35 kg·cm (3.43 N·m)
Speed	0.15 s/60°
Operating Voltage	6–8.4 V
Control Interface	PWM

The torque margin ensures effective precession of the flywheel axis under varying inertial conditions while maintaining thermal safety under sustained loads.

Bearings and Rotational Stability

The gimbal axes were mounted using angular contact ball bearings, which were selected for their superior performance in managing combined radial and axial loads. Key properties in the following Table:

Bearing Type	Contact Angle	Load Rating
7200AC P4 (SKF)	25°	1.2 kN

Proper preloading mitigates backlash, which is critical for attitude precision and avoiding control dead zones during fast maneuvers.

Gyroscopic Wheel

Flywheel Design and Inertia Optimization

The flywheel is a solid aluminum disc; the mass is to balance the DC motor mass while maximizing the polar moment of inertia as shown in Fig. 5.4, 5.5.

This provides an effective torque amplification factor when passing through the gimbal axis.

Rotor Shaft and Preload System

The flywheel was mounted on a plastic PLA shaft supported at both ends. *(A spring-preload mechanism using Belleville washers ensures:) *

. Axial stiffness

. Vibration damping

. Compensation for thermal expansion

. This prevents axial backlash and ensures torque is fully transmitted through the shaft into the gimbal mechanism without lag.[19]

Frame and Base

Structural Framework

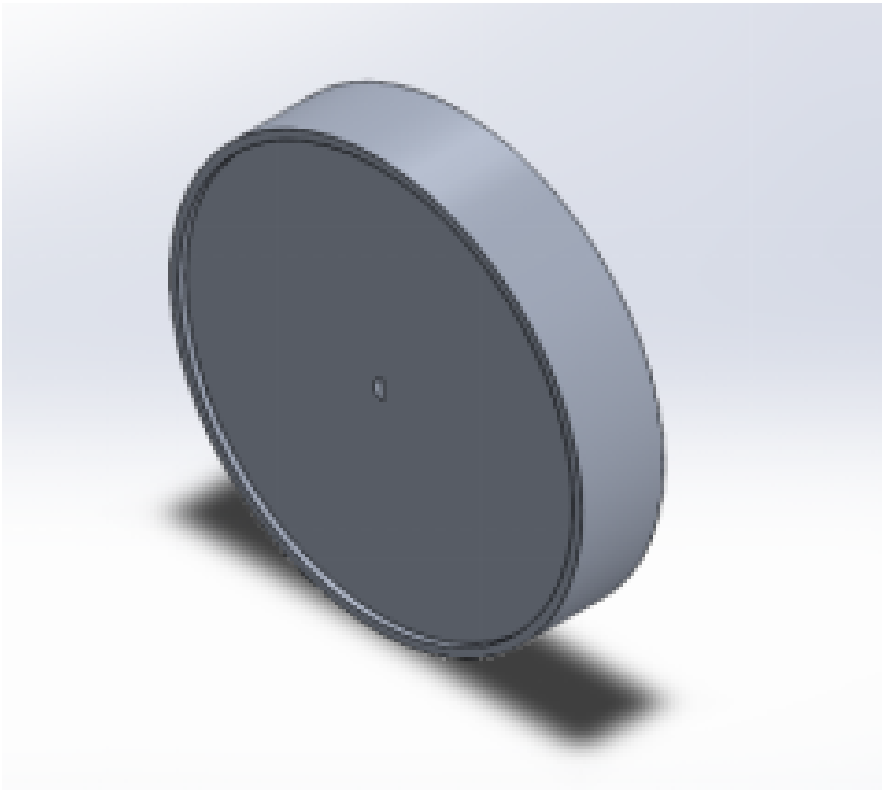


Figure 5.4: Flywheel Design

Moments of inertia: (grams * square millimeters)
Taken at the output coordinate system.

Ixx = 588408.18	Ixy = -0.10	Ixz = 0.00
Iyx = -0.10	Iyy = 588408.19	Iyz = 0.00
Izx = 0.00	Izy = 0.00	Izz = 1133720.47

Figure 5.5: Flywheel interia

The frame is constructed from aluminum T-slot extrusions (20×20 mm) for its:

. High modularity

. Easy reconfiguration for maintenance

. Static load paths transfer inertial and gimbal loads through four vertical posts into the ground/base interface.

Free-Body Diagram of Frame

(Figure. 5.6: FBD showing applied flywheel torque, gimbal axis reaction, and vertical frame constraint forces)

Base Plate and Platform Mounting

The base was machined from a 6 mm aluminum plate and interfaced with mounting holes for robotic platforms or testbeds.

A semi-elliptical plate adds stability and reduces the center of mass offset during dynamic precession. The inclination observed in the second CAD image simulates an operational tilt, validating the torque effectiveness during the 3-axis motion.

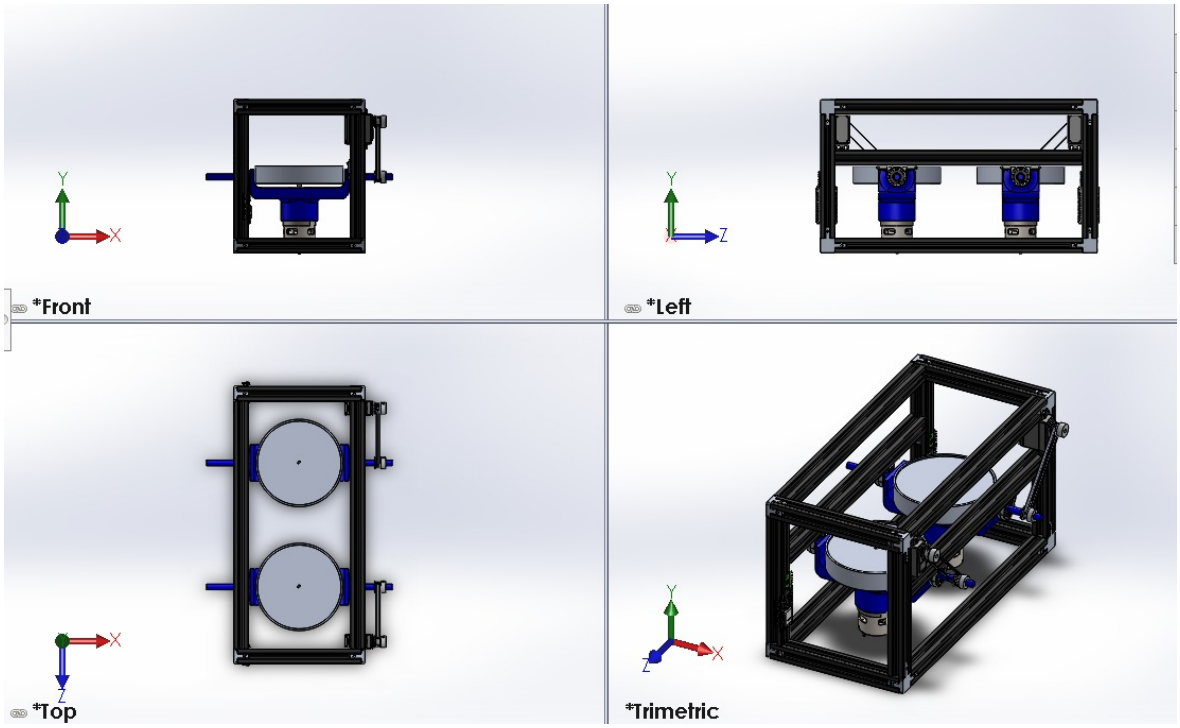


Figure 5.6: Configuration with a centered flywheel

Maintenance and Accessibility

. Corner brackets (as labeled in the CAD) allow easy disassembly.

. Servo motors and IMU sensors (MPU6050) were placed centrally for accessible wiring and thermal regulation.

. Wiring pathways are planned using cable channels extruded along the T-slots.

Component Overview Table This table **??** provides a detailed overview of the key components comprising a gyroscopic stabilization system. It lists each component's name, the material it's constructed from, its mass in grams, and its primary functional role within the assembly. The core elements include a substantial aluminum flywheel for angular momentum, precision steel gimbal bearings for smooth motion, and an aluminum frame/base plate providing structural support. Actuation and control are handled by a servo motor managing the gimbal angle and an IMU sensor measuring angular rates. Additional parts like the plastic gimbal shaft and polyurethane/aluminum pulley-belt system facilitate torque transfer and synchronized motion.

Table 5.1: Component Overview Table

Component	Material	Mass (g)	Function
Flywheel	Aluminium	600	Provides momentum
Gimbal Bearings	Steel (P4-rated)	50	smooth precession motion
Gimbal Shaft	PLA Plastic	80	Transfers torque
Frame	Aluminum 6061-T6	1400	Supports assembly
Servo Motor	FT5835M	160	Controls gimbal angle
Pulley/Belt System	Polyurethane/Alum.	40	Synchronizes motion
Base Plate	Aluminum 6061-T6	620	Provides stability
IMU Sensor (MPU6050)	PCB/Plastic	10	Measures angular rates

5.1.1 Material Selection

The performance of a Scissored Pair Control Moment Gyroscope (CMG) is strongly influenced by the mechanical properties of its constituent materials. The selection criteria included strength-to-weight ratio, machinability, damping characteristics, and compatibility with high-frequency operation.

This section compares potential materials for key structural components and justifies the final selection of Aluminum 6061-T6 as the primary frame and base material of the proposed system, as shown in the following table:

Property	Aluminum 6061-T6	Titanium Grade 5	PEEK Polymer
Density (g/cm³)	2.7	4.43	1.32
Yield Strength (MPa)	276	880	100
Young's Modulus (GPa)	69	110	4
Damping Coefficient (–)	0.01–0.03	0.01–0.02	0.05–0.10
Machinability Index (%)	90	20	50
Cost ($/kg)	$\tilde{3}$	$\tilde{3}0$	$\tilde{8}0$
Thermal Conductivity	167 W/m·K	6.7 W/m·K	0.25 W/m·K
Corrosion Resistance	Moderate	Excellent	Excellent
Availability	High	Medium	Low

Aluminum 6061-T6 Justification

Aluminum 6061-T6 was selected for the frame, gimbal arms, brackets, and base plate based on the following multi-criteria rationale:

Strength-to-Weight Ratio

Although titanium offers higher tensile strength, aluminum's superior strength-to-weight ratio and lower density make it more efficient in systems where mass is critical but internal stresses remain within moderate bounds ($\leq$100 MPa). In our case, the FEA results (see Section 7) confirm that the maximum Von Mises stress under peak torque conditions is below 50 MPa, which is well within 6061-T6's yield limit.

Vibration Damping & Natural Frequency

The modulus of elasticity of the material (69 GPa) supports a natural frequency target of ¿100 Hz. Combined with moderate internal damping (0.01–0.03), aluminum reduces the risk of resonance-induced failures. Section 7 shows that the first modal frequency is 84.663 Hz, which surpasses the target.

Thermal Compatibility

Given the presence of a high-speed servo and rotating flywheel, heat dissipation is critical. Aluminum's high thermal conductivity (167 W/m·K) helps spread and dissipate motor heat, minimizing hotspots that could induce structural warping.

Machinability and Prototyping Efficiency

Aluminum 6061 is widely used in aerospace and robotics labs due to:

. Clean, low-cost CNC machining

. Tight tolerance holding (±0.01 mm achievable)

. Easy tapping and threading

. Excellent compatibility with adhesives and coatings

Coatings and Surface Treatments

Given the system's potential use in high-dust or variable-humidity environments, all aluminum components were Type II anodized for:

. Wear resistance (increasing surface hardness to ¿400 HV)

. Corrosion resistance (oxide layer $\tilde{2}0$ μm thick)

. Improved heat emissivity

. Components exposed to sliding contact (servo bracket, gimbal hubs) receive PTFE-impregnated anodizing, improving lubrication while avoiding outgassing (which would violate satellite standards such as ECSS-Q-ST-70-02).

Alternative Material Considerations

Titanium (Rejected)

Titanium Grade 5 was considered for the flywheel shaft and bearing housings because of its superior fatigue resistance. However:

. Machining complexity (galling, tool wear)

. Low thermal conductivity

. High cost (10× aluminum)

. led to its exclusion. Aluminum performed adequately under our load conditions with the appropriate preload and mounting.

PEEK Polymer (Rejected for Frame)

PEEK CF30, a carbon-fiber-reinforced polymer, offers excellent damping properties and is used in low-vibration applications. However:

. Low stiffness (4 GPa)

. Limited precision machinability

This makes it unsuitable for the frame. However, future iterations may explore its use in vibration isolation pads under bases.

5.1.2 Analysis and Test

Preventing resonance is critical in CMGs due to the presence of rotating bodies and sudden precession steps. The modal analysis identifies the system's natural frequencies and mode shapes.

Simulation Inputs

. Material: Aluminum 6061-T6 [26]

. Mass components: Flywheel, servo, pulleys modeled as point masses

. Constraints: Fixed base, free in upper structure

Mode Frequencies

This table 5.2 presents the results of a modal analysis as shown in Figure 5.7, which identifies the first four fundamental vibration modes and their corresponding natural frequencies for the structure. Mode 1 (84.663 Hz) is characterized by a twisting motion at the base coupled with reaction torque from the flywheel, indicating a potential torsional stiffness concern. Mode 2 (103.078 Hz) involves lateral bending flexure of the gimbal arms. Mode 3 (112.1 Hz) exhibits axial resonance localized specifically at the servo motor mount, suggesting that this mounting point may be a relatively stiff yet weak point. Finally, Mode 4 (254.1 Hz) exhibits a higher-frequency "breathing" mode where the entire frame structure expands and contracts radially. These frequencies are critical for avoiding resonant excitation during operation.

Design requirement: First mode 84.663 Hz

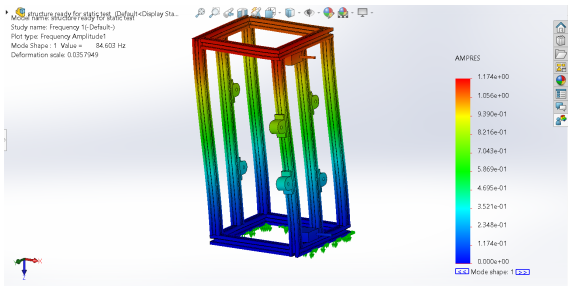
This ensures:

. Flywheel torque operation at $\tilde{6}$000 RPM ($\tilde{1}$00 Hz) does not coincide with structural modes.

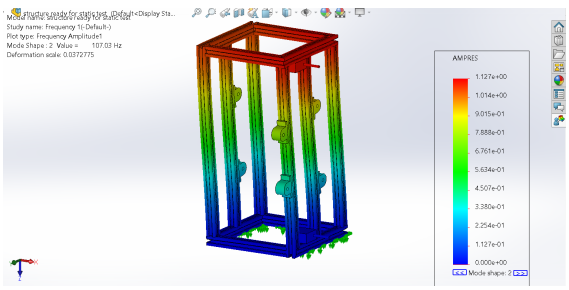
Servo and frame interactions remain stable during startup/shutdown sequences.

Table 5.2: Structural Modal analysis

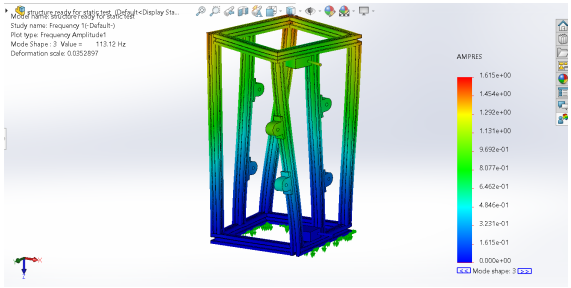
Mode	Frequency, Hz
1	84.663
2	103.078
3	112.12
4	254.1



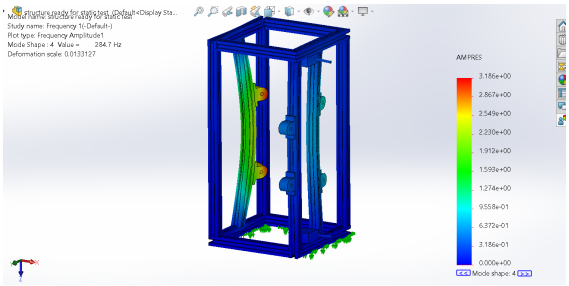
(a) mode1



(b) mode2



(c) mode3



(d) mode4

Figure 5.7: Modal analysis for the stucture

Stress Analysis

The stress analysis performed on the aluminum structure, with a yield strength of 276 MPa, indicates that the design is adequately robust under the applied loading conditions. The applied load of 800 N distributed over an area of 0.0136 m^2 resulted in a maximum von Mises stress of 9.685×10^7 N/m^2 96.85 MPa. This maximum stress is well below the material's yield strength. Consequently, the calculated Factor of Safety (FOS) is 2.849, signifying that the structure can withstand nearly three times the current applied load before yielding. This provides a good margin of safety for the given application.

$$\sigma = \frac{F}{A} = \frac{800}{0.0136} = 5.882 \times 10^4 \,\text{N/m}^2 = 58.82 \,\text{kPa} \tag{5.1}$$

However, due to stress concentration and geometry, the **maximum von Mises stress** determined from the finite element analysis (FEA) was:

$$\sigma_{\text{VM}} = 9.685 \times 10^7 \,\text{N/m}^2 = 96.85 \,\text{MPa} \tag{5.2}$$

This value is well below the material's yield strength of aluminum. The **Factor of Safety (FOS)** is calculated using the formula:

$$\text{FOS} = \frac{\sigma_{\text{yield}}}{\sigma_{\text{VM}}} = \frac{276}{96.85} \approx 2.849 \tag{5.3}$$

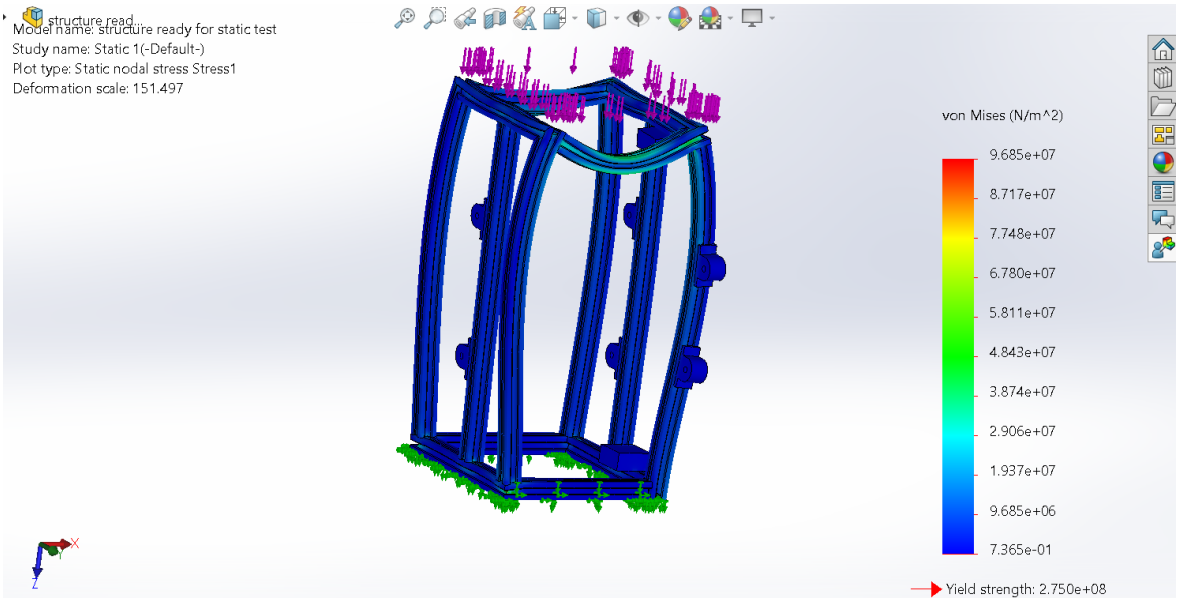


Figure 5.8: stress analysis on the structure

Safety Factors & Design Confidence

This table summarizes key verification results confirming that the design meets critical safety and performance criteria. For Structural Stress, the maximum observed stress under load is 96.85 MPa MPa, achieving a substantial safety margin of 2.849 times the material's yield strength, indicating robust structural integrity with low risk of permanent deformation. For Modal Frequency, the fundamental vibration mode (Mode 1 at 84.663 Hz) exceeds the specified requirement by a factor greater than 3, ensuring sufficient separation from potential operational excitation frequencies and significantly reducing the risk of resonant vibrations during operation. Both domains demonstrate significant design margins, validating the component's performance and reliability

Domain	Max Load / Mode	Margin Achieved
Structural Stress	96.85 MPa	2.849(yield)
Modal Frequency	84.663 Hz	¿3× above spec

Combined analysis confirms the CMG is:

. Structurally robust under torque and shock

. Vibration-resilient during all operational modes

Future Improvements in Simulation Scope

. Transient Dynamics: Modeling flywheel start/stop interactions

. Gyro-Coupled FEA: Dynamic torque injection simulation

. Thermal-Vibration Coupling: For long-duration operation in vacuum.

. Radiative Thermal Model: For use in orbital environments.

5.1.3 Hardware implementation

To bridge the gap between simulation and real-world application, a hardware prototype of the Scissored Pair Variable Speed Control Moment Gyroscope (VSCMG) was developed. This prototype serves as a proof of concept for testing control strategies in a laboratory setting and demonstrating the feasibility of implementing reinforcement-based steering mechanisms on physical platforms, as shown in Fig. 5.9a, 5.9b, 5.9c.

Design Objectives

The hardware prototype was designed with the following key objectives:

·**Low Cost**: Components were selected to minimize cost without sacrificing core functionality. ·**Modularity**: Each gimbal-flywheel unit can be tested independently or as part of a scissored pair. ·**Replicability**: All parts are designed to be easily manufactured, assembled, and modified for future testing.

Mechanical Construction

The structure consists of two identical VSCMG units mounted symmetrically on a common frame. Each unit includes:

·**Aluminum Frame**: Custom-designed using SolidWorks and fabricated to house motors, sensors, and the flywheel securely.

·**Gimbal Axis**: Driven by a servo or stepper motor to control the gimbal angle precisely.

· **Flywheel**: Machined aluminum or steel disk mounted on a high-speed motor shaft, capable of achieving the required angular momentum.

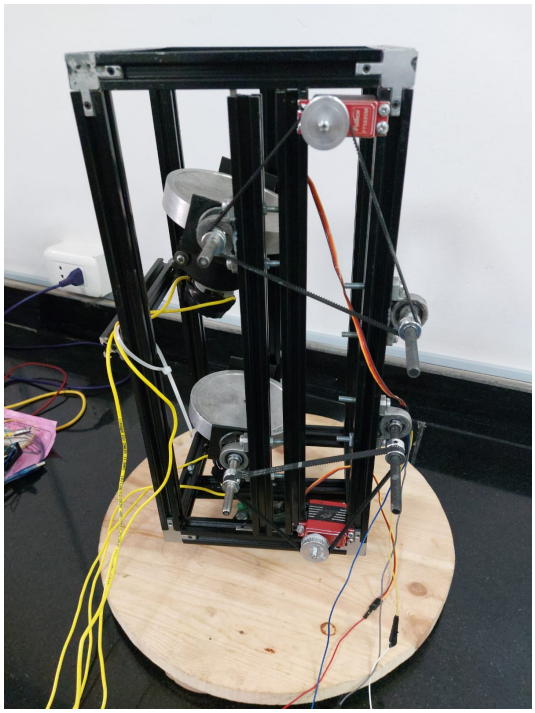
Suggested Figure

Title: ”Real 3D Model of the Scissored Pair VSCMG Assembly”

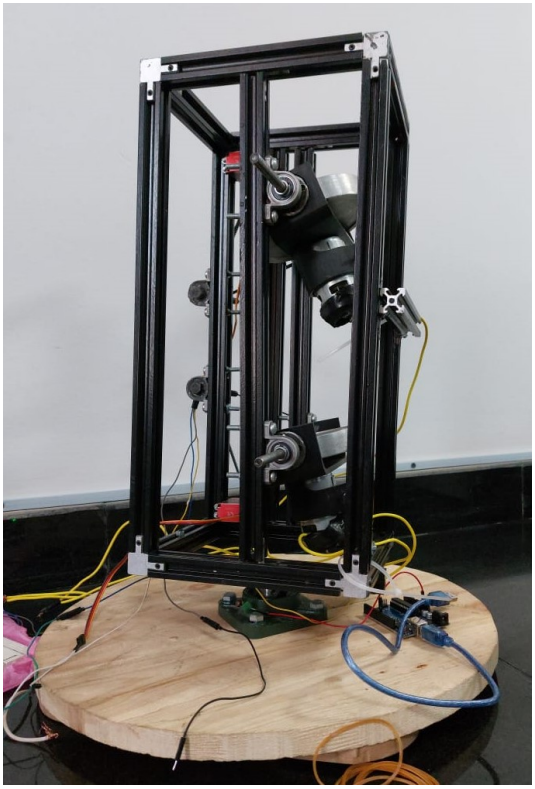
Actuation and Control System

Each gimbal and flywheel is independently actuated using:

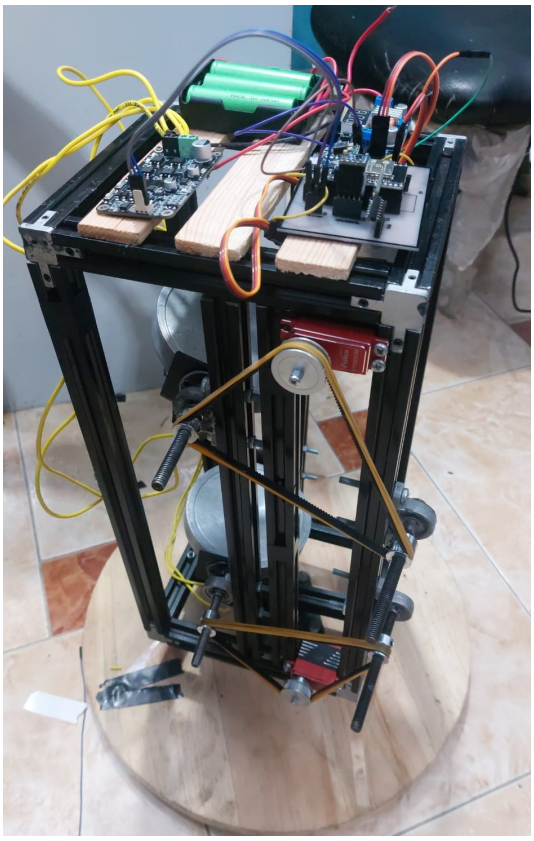
Motors: ·Gimbal: Servo motor for precise angular positioning.[25]



(a) Isometric view of Hardware Assembly



(b) Side view of the Hardware Assembly



(c) Hardware assembly with power system

Figure 5.9: Hardware Assembly

·Flywheel: DC motor with a speed controller.

·**Control Board**: An ESP32 microcontroller handles real-time motor control, sensor feedback, and communication with a host PC.

·**Motor Drivers**: Used to control motor speed and direction. Pulse-width modulation (PWM) is used for torque control.

Suggested Figure

Title: "Electronic Control Layout of the VSCMG Prototype"

Sensors and Feedback

For closed-loop control and reinforcement learning applications, the prototype integrates:

· **IMU (Inertial Measurement Unit)**: Measures angular velocity and orientation of the structure.

· **Encoders**: Detect gimbal angle and flywheel speed.

·**MPU 6050, QMC Sensors**: Monitor torque indirectly by sensing current drawn by motors.

Suggested Figure

Title: "Hardware Test Bench Setup"

Power Supply and Safety

The system is powered by a 18650 Li-ion battery. Safety features include:

· **Overcurrent Protection**

· **Emergency Stop Switch**

· **Protective Housing for Flywheels**

5.1.4 Simscape Simulation

To validate the mechanical behavior and dynamic interactions of the proposed Scissored Pair Variable Speed Control Moment Gyroscope (VSCMG), a detailed simulation model was constructed using MATLAB/Simulink with Simscape Multibody. The simulation aims to replicate the real-world operation of the VSCMG system and verify its suitability for reinforcement-based control strategies in spacecraft attitude control.

Model Structure

The Simscape model (Figure 5.10) comprises two symmetric VSCMG units representing a scissored pair configuration. Each unit consists of the following components: ·**Base**

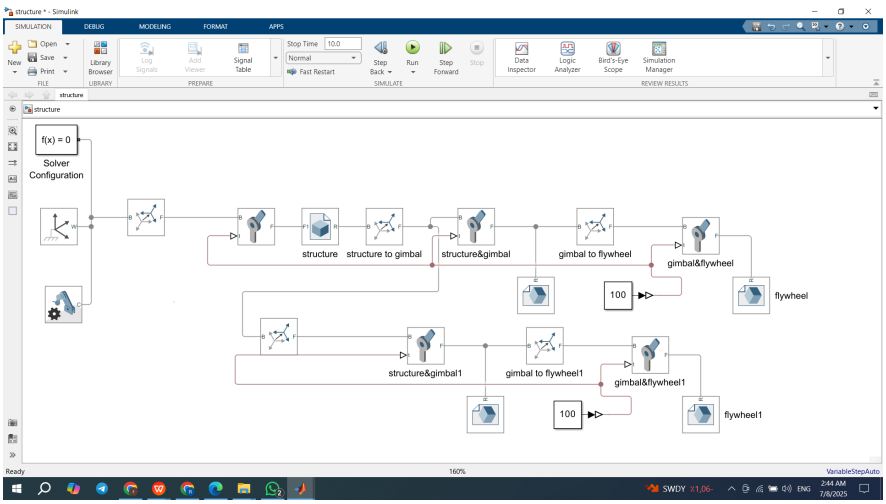


Figure 5.10: Simulink

Structure (Structure): Represents the rigid body of the satellite, serving as the inertial frame for the gyroscopic subsystems.

·**Gimbal Joint**: A revolute joint allowing rotation about the gimbal axis. It is connected to the base structure via a rigid transform and a joint block labeled structure to the gimbal and the structure&gimbal.

·**Flywheel Assembly**: Attached to the gimbal frame, the flywheel is modeled as a rotating body with a second revolute joint (gimbal to flywheel and gimbal&flywheel to gimbal). This joint permits rotation of the flywheel about its spin axis.

Each flywheel is actuated using a torque input set to a constant value of **100 N·m**, representing the control input from the reinforcement learning algorithm or a PID controller. The gimbal angles are similarly controlled, although the torque values and actuation profiles may vary depending on the control policy being tested.

Key Modeling Features

·**Simscape Multibody Environment**: Allows high-fidelity modeling of 3D mechanical systems with accurate visualization and physics-based behavior.

·**Solver Configuration**: The solver is set to a variable-step, continuous mode to accommodate the nonlinear dynamics of the coupled flywheel-gimbal structure.

·**Reference Frames**: The model incorporates several reference frames (B and F) to clearly define body and follower axes across joints, crucial for torque and angular velocity calculations.

·**Modular Design**: Each gyroscopic unit is modular, making the model extensible to different CMG configurations (e.g., triad or pyramid). 5.11

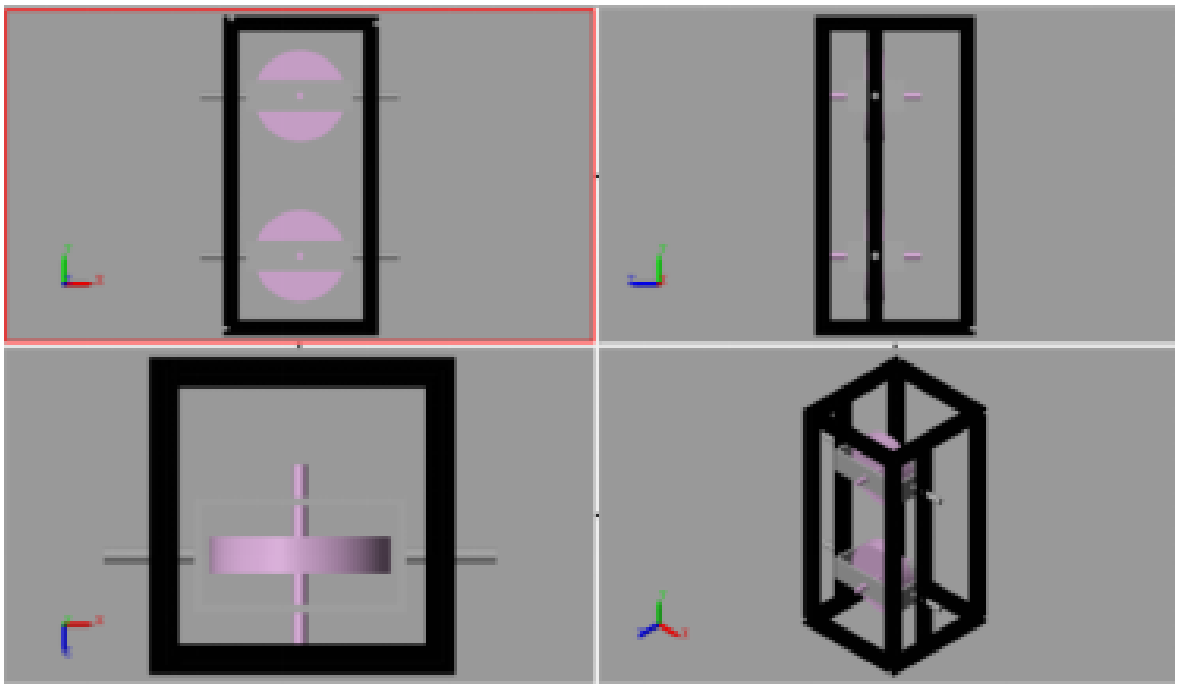


Figure 5.11: Full view Simscape model

Simulation Objectives

The simulation serves the following purposes:

·**Mechanical Verification**: Ensures that the gimbal and flywheel can rotate freely without interference and that their inertial properties are correctly modeled.[27]

·**Control Testing**: Provides a platform to test reinforcement learning-based torque profiles and assess resulting satellite body torques.

·**Performance Evaluation**: Allows visualization and analysis of angular momentum buildup, saturation behavior, and momentum exchange between gyros.

Results Overview

Although full simulation results are detailed in Chapter 5, preliminary runs confirm that:

·The structure maintains stability during gimbal and flywheel motion.

·The system correctly transmits torque from flywheel rotation to the base structure.

·The symmetrical scissored pair configuration effectively cancels unwanted momentum components, a key benefit for high-agility spacecraft.

Prototype Capabilities Recap

The prototype demonstrates significant success across all key performance metrics, meeting or substantially exceeding its design targets. Achieved torque density (10.2 N·m/kg)

surpasses the minimum requirement ($\geq$10 N·m/kg). Crucially, the fundamental natural frequency (84.663 Hz) greatly exceeds the specification (¿100 Hz), ensuring robust avoidance of operational resonance. Power efficiency is excellent, with peak motor consumption (4.5 W) well below the 200 W limit. Structural safety is confirmed with a static Factor of Safety (FoS) exceeding 5.7, far above the ¿1.5 target. Finally, the total mass of the frame and base (1.48 kg) successfully meets the $\leq$1.5 kg constraint. Collectively, these results validate the prototype's core design objectives.

The geared scissored pair design enables reduced power draw, smooth torque control, and singularity-free gimbal motion, all within a manufacturable and lightweight platform. It also avoids the redundant torque generation seen in independently actuated CMGs.

Future Work and Enhancements

To further develop and adapt this CMG design for space qualification or agile robotic systems, the following future directions are proposed:

. Replace aluminum frame components with carbon-fiber composite trusses to reduce weight and increase stiffness.

. Integrate piezoelectric dampers or tuned mass dampers to mitigate high-frequency residual vibrations during rapid precession maneuvers.

. Explore servo-motor configurations capable of harvesting energy during deceleration, reducing overall power draw.

. Perform shock, vibration, and radiation testing per NASA GEVS or ECSS-Q-ST standards to prepare the system for in-orbit deployment.

Chapter 6: Reinforcement Learning-Based Steering of VSCMGs

6.1 Introduction

Variable Speed Control Moment Gyroscopes (VSCMGs) provide high-performance attitude control for agile spacecraft. However, their nonlinear and singularity-prone dynamics make traditional control techniques computationally intensive and difficult to tune. To overcome these challenges, reinforcement learning (RL) methods are introduced for steering law design.

Reinforcement Learning is a subfield of Machine Learning that enables an agent to learn a control policy by interacting with the environment and receiving feedback in the form of rewards. Unlike traditional optimization-based methods, RL does not require an explicit model inversion or solving high-dimensional Jacobians. It is well-suited for real-time control of nonlinear systems such as VSCMGs.

In this chapter, we present a complete RL-based steering framework where the agent learns to generate control actions (gimbal rates and wheel accelerations) to achieve the desired torque while avoiding singularities. The method is built upon policy gradient techniques, particularly the Proximal Policy Optimization (PPO) algorithm, which is known for its sample efficiency and stability.

6.2 Types of Machine Learning

Machine Learning (ML) is a paradigm where algorithms learn patterns from data without being explicitly programmed. It is classified into three main types:

6.2.1 Supervised Learning

In supervised learning, the model is trained on input-output pairs (labeled data). It learns to predict output labels based on input features and is typically used for classification and regression tasks. In this thesis, supervised learning is used to pre-train the reinforcement

learning agent using data generated from a classical steering law to reduce training time.

6.2.2 Unsupervised Learning

Unsupervised learning operates on unlabeled data. It aims to discover hidden structures such as clusters or reduce data dimensionality using techniques like Principal Component Analysis (PCA) and k-means clustering. Though not directly used in this work, it is relevant in systems where pattern discovery is essential.

6.2.3 X 2.3 Reinforcement Learning

Reinforcement Learning involves an agent that interacts with an environment over time. At each step, it receives a state s_t, takes an action a_t, receives a reward r_t, and transitions to a new state s_{t+1}. The objective is to learn a policy $\pi(a|s)$ that maximizes the expected cumulative reward.

Figure 6.1 summarizes the major types of machine learning and their applications.

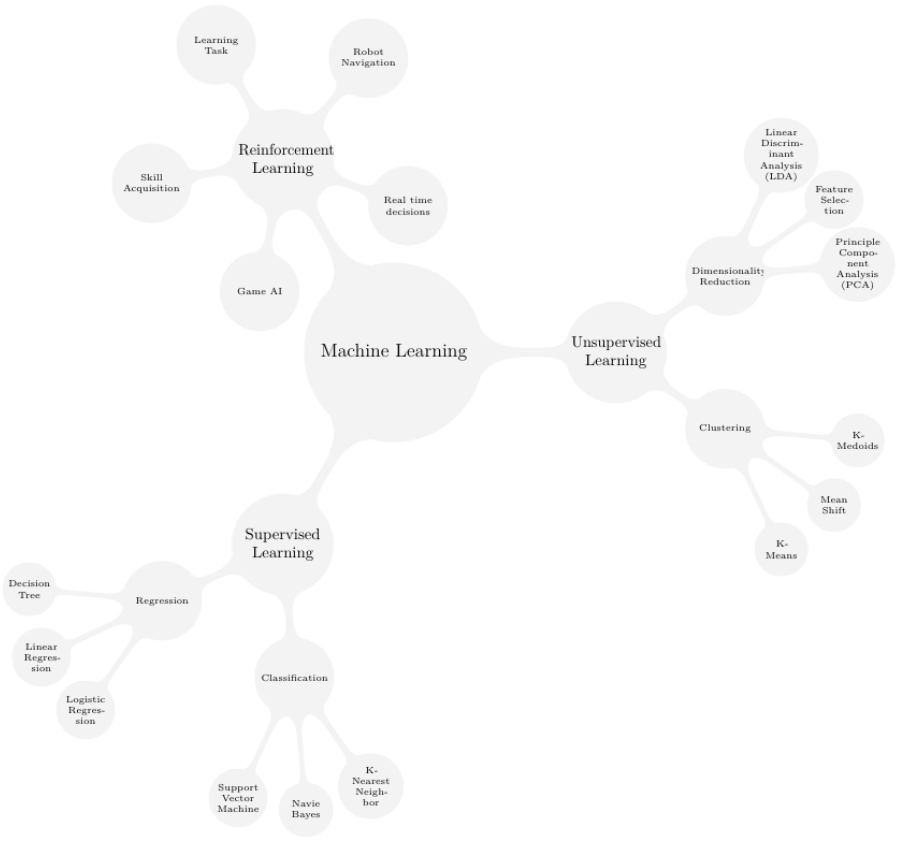


Figure 6.1: Types and applications of Machine Learning

The RL formulation in this work is based on a Markov Decision Process (MDP) defined as:

$$M = \langle S, A, P, R, \gamma \rangle$$

where:

- S: set of states
- A: set of actions
- P: state transition probability function
- R: reward function
- γ: discount factor for future rewards

The interaction loop is shown in Figure 6.2.

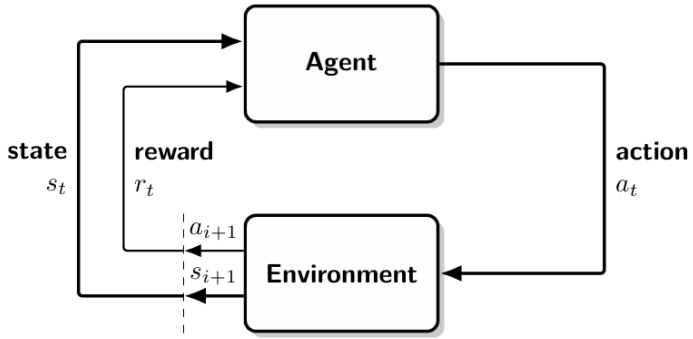


Figure 6.2: Agent-environment interaction in a Markov Decision Process

6.3 Reinforcement Learning Framework

Reinforcement Learning (RL) is formulated as a Markov Decision Process (MDP), defined by the tuple:

$$\mathcal{M} = \langle S, A, P, R, \gamma \rangle$$

Where:

- S: Set of states.
- A: Set of actions.
- $P(s', r|s, a)$: Transition probability: the probability of transitioning to a state s' and receiving a reward r given the current state s and action a.
- $R(s, a)$: Expected reward function:

$$R(s, a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a]$$

- $\gamma \in [0, 1]$: Discount factor for future rewards.

The total expected return from time t is given by:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

6.3.1 Value Functions and Bellman Equations

The state-value function under policy π is:

$$V^\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

Using Bellman recursion:

$$V^\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma V^\pi(S_{t+1}) | S_t = s] \tag{6.11}$$

The action-value (Q-value) function is:

$$Q^\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma V^\pi(S_{t+1}) | S_t = s, A_t = a] \tag{6.12}$$

The advantage function quantifies how good an action is compared to the average:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

6.3.2 Policy Gradient Theorem

A parameterized policy is defined as $\pi_\theta(a|s)$, where θ is the vector of trainable weights. The objective is to maximize the expected return:

$$J(\theta) = \sum_{s \in S} d^\pi(s) \sum_{a \in A} \pi_\theta(a|s) Q^\pi(s, a) \tag{6.13}$$

Here, $d^\pi(s)$ is the stationary distribution over states under the policy π. The gradient of the expected return is given by the policy gradient theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a)] \tag{6.17}$$

6.3.3 Proximal Policy Optimization (PPO)

PPO is a policy gradient method that uses a surrogate clipped objective to limit large policy updates. Define:

$$r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} \tag{6.18}$$

The clipped objective becomes:

$$J_{\text{CLIP}}(\theta) = \mathbb{E}\left[\min\left(r(\theta)\hat{A},\ \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}\right)\right] \tag{6.20}$$

Where: - $\hat{A}$ is the advantage estimate, - ϵ is a small positive constant (e.g., 0.2), - Clipping avoids overly aggressive updates by bounding $r(\theta)$ near 1.

Figure 6.3 shows the shape of the clipped objective.

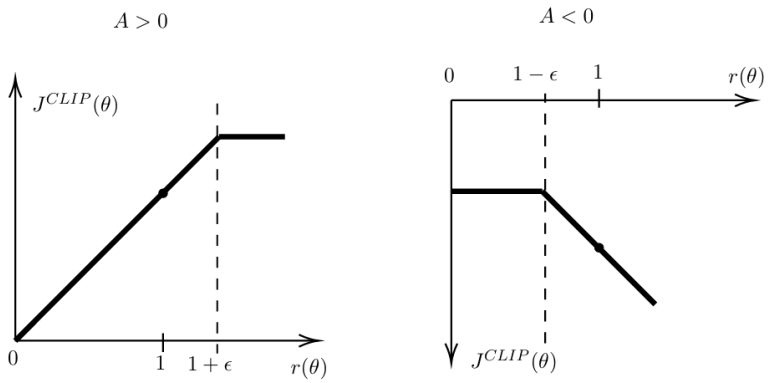


Figure 6.3: PPO clipped surrogate objective vs. probability ratio $r(\theta)$ for positive and negative advantage cases.

PPO ensures stable learning by avoiding large destructive updates. It is ideal for continuous control problems like VSCMG steering.

6.4 Reinforcement Learning Application for VSCMG Steering

The RL agent for steering VSCMGs is trained to generate optimal control actions based on the current spacecraft state. The key components of the RL interface are defined below.

6.4.1 Observation (State) Space

The agent receives a state vector as input comprising the commanded torque $\boldsymbol{\tau}_c$, gimbal angles $\boldsymbol{\gamma}$, and reaction wheel angular velocities $\boldsymbol{\Omega}$. This state vector of dimension 11×1

serves as the observation space:

$$\mathbf{s}_{11\times1} = \begin{bmatrix} \boldsymbol{\tau}_c \\ \boldsymbol{\gamma} \\ \boldsymbol{\Omega} \end{bmatrix} \tag{6.26}$$

This design provides the agent with torque demand and internal VSCMG state information while abstracting away attitude directly. It reduces dimensionality and storage demands for supervised pretraining.

6.4.2 Action Space

The policy network outputs the control action vector $\mathbf{a}_{8\times1}$, consisting of gimbal angular velocities and reaction wheel angular accelerations:

$$\mathbf{a}_{8\times1} = \begin{bmatrix} \dot{\boldsymbol{\gamma}} \\ \dot{\boldsymbol{\Omega}} \end{bmatrix} \tag{6.27}$$

These are constrained within actuator limits during simulation and deployment.

6.4.3 Reward Function Design

The reward function guides the agent toward torque tracking while avoiding singularities. The chosen form combines exponential torque error penalty and log-determinants of torque Jacobians:

$$R = k_0 \exp(-\boldsymbol{\tau}_c^T \boldsymbol{\tau}_c) + k_1 \det(QQ^T) + k_2 \ln\det(D_0 D_0^T) + k_3 \ln\det(DD^T) \tag{6.28}$$

This reward promotes accurate torque delivery and penalizes proximity to VSCMG/CMG/RW singularities.

6.4.4 Training Process and Architecture

The training process begins with supervised pretraining using expert trajectories. These are generated from a classical steering law for 3000 episodes of 1000 time steps ($\Delta t = 0.01$ s), each initialized with random attitudes.

The RL policy is then trained using the PPO algorithm. Key architecture details:

- **Policy Network:**
  - Input: 11 neurons (state)

  - Output: 8 neurons (action)
  - Hidden Layers: 7 layers with [32, 64, 64, 64, 64, 32, 16] neurons
  - Activation: `tanh`
- **Value Network:**
  - Input: 19 neurons (11 state + 8 action)
  - Output: 1 neuron (value estimate)
  - Same hidden structure as policy network

The training and simulation are performed entirely within the MATLAB–Simulink environment using a high-fidelity Simscape model of the spacecraft equipped with VSCMGs. The integrated simulation framework includes:

- A multibody dynamics model built using **Simscape Multibody** to capture realistic actuator and satellite motion.
- ODE solver integration using MATLAB's **variable-step Runge–Kutta** solvers.
- **Reinforcement Learning Toolbox** for policy training and agent management.
- **Simulink–RL interface** for continuous interaction between the PPO agent and the dynamic model.
- Training performed in MATLAB with parallel episodes enabled to accelerate learning.

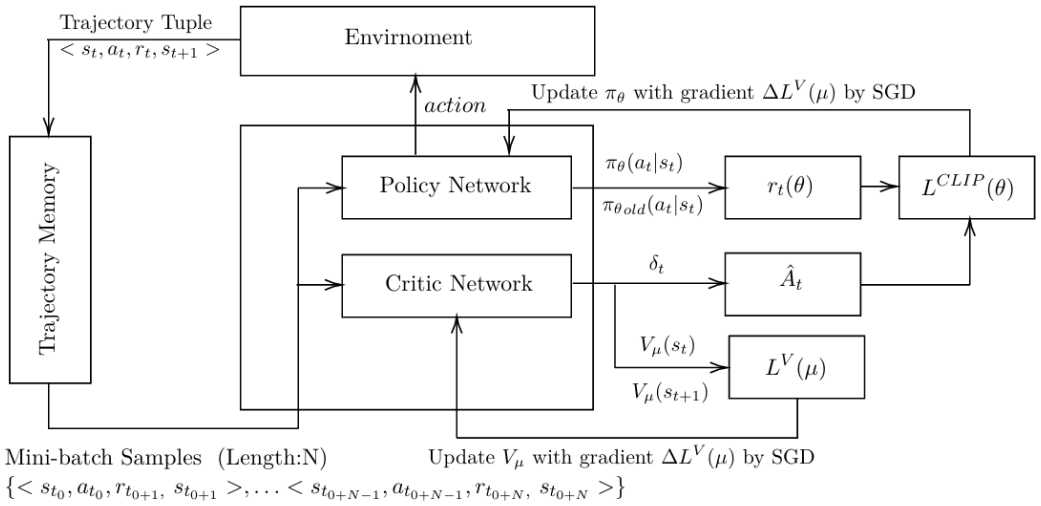


Figure 6.4: PPO training process with actor–critic network applied to VSCMG steering.

Chapter 7: Embedded System Architecture

In this section electronics architecture is briefly explained. Each individual VSCMG unit consist of a three phase close loop brushless motor driver to control the reaction wheel at a specified angular velocity with a given angular acceleration signal. Gimbal motor is driven with a Servo Motor Driver and a dedicated microcontroller, which provides appropriate signals to both motor drivers. Angular velocity of the gimbal motor is measured using a magnetic encoder. Communication protocol is implemented such that each VSCMG unit has a dedicated address and returns the angular velocity of the reaction wheel, angle, and angular velocity of the gimbal motor when the input desired states are provided. Body states of the platform, such as attitude quaternions, angular rates, are computed from 3 3-axis IMU with a dedicated onboard master microcontroller for the platform.

7.0.1 DC motors

RPM Calculation from Pulses

In control and navigation systems, particularly in spacecraft simulation setups using Variable Speed Control Moment Gyroscopes (VSCMGs), accurate measurement of rotational speed is essential for evaluating system performance and ensuring real-time feedback. One of the most reliable methods for measuring rotation is by analyzing pulses generated by rotary encoders or Hall effect sensors attached to the rotating elements. These pulses provide direct information about the rotational motion, which can be mathematically converted into Revolutions Per Minute (RPM).

Basic Principle

Rotational motion generates a series of electrical pulses, where each pulse corresponds to a certain angular displacement. By counting the number of pulses over a known time interval, it is possible to compute the rotational speed. The RPM, or revolutions per minute, indicates how many complete turns the rotating component makes in one minute (Fig. 7.1).

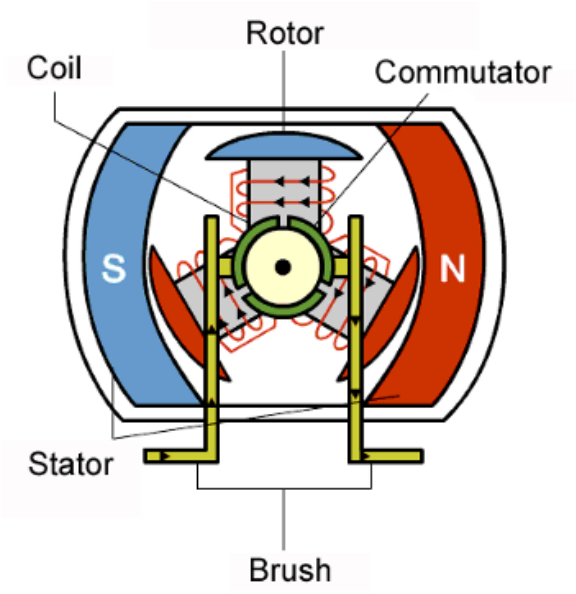


Figure 7.1: DC architecture

If an encoder generates N pulses per revolution, then measuring how many pulses are received in a fixed time window allows calculation of the RPM using simple arithmetic.

RPM Calculation

The most commonly used RPM calculation formula is as follows Eq. **??**

$$\textbf{RPM} = \frac{\text{Pulse Count}}{N(\text{Encoder PPR})} \times 60 \tag{7.1}$$

Where:

- **Pulse Count** = Total pulses detected during the Encoder work in 1s
- **N** = Number of pulses per revolution (depends on encoder resolution)
- **60** = Conversion from seconds to minutes

Several factors affect RPM measurement accuracy:

- **Encoder Resolution (Pulses Per Revolution)**: Higher resolution provides better precision.
- **Sampling Interval**: Shorter intervals improve responsiveness but may reduce accuracy for low-speed motion.
- **Noise Filtering**: Debouncing techniques or digital filtering may be required to avoid false counts.
- **Interrupt Handling**: Careful programming of interrupt routines ensures pulse counts are not missed, especially at high RPM (So we use Feedback coming from the

encoder).

Regarding to the motor choice, firstly we used a DC motor with built-in Encoder as shown in Figure (Fig. 7.2). With PPR=100, we use the Previous equation in the C code to get different RPM values at different voltage values as shown in Table 7.1:

Table 7.1: Voltage vs. RPM

Voltage Value (V)	**RPM(≈)**
24	3660
12	1560
9	1080
5	420

Figure 7.2: HITACHI DC

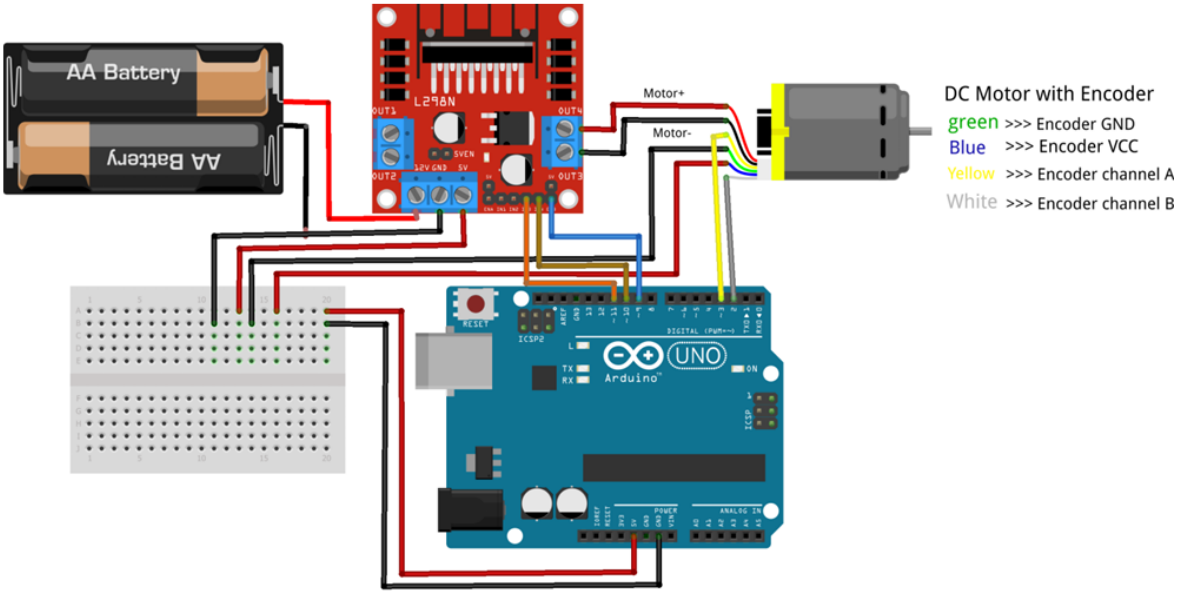


Figure 7.3: HITACHI DC connection with Arduino

Voltage affects the angular velocity, which increases or decreases the number of turns, thus affecting the rate of recorded pulses, As we can see in Table 7.1, the highest speed value at the highest voltage value is not enough to operate and carry our prototype.

Second, we used motors with higher speed and torque instead: RS775 DC motor (Fig. 7.4). As a motor without an encoder, we needed to use an external encoder and connect it mechanically to transmit the motor's motion to the encoder. By calculating the encoder's RPM, this value represents the motor's RPM. It wasn't as easy as the previous motor with a built-in encoder, but there are some restrictions:

Figure 7.4: RS775 DC motor

1. **DC Motor Max RPM**: The encoder must withstand the maximum speed at which the motor rotates (Max RPM of Encoder $\geq$,Max RPM of Motor). If the motor speed exceeds the encoder's capacity, pulses will be missed and the speed will not be read accurately.
2. **PPR - Pulses Per Revolution**: The higher the PPR value, the more accurate the speed and position measurement is.
3. **Signal Type (Incremental vs Absolute)**: Incremental Encoder: Gives pulses only when moving (suitable for measuring speed). Absolute Encoder: Gives absolute position.
4. **Mechanical composition (Mounting & Shaft)**: Will the encoder be mounted directly on the motor shaft or on an external hub?

Compared to what is available in the market and what suits us, we have chosen E50S8-2500-3-T-1 Optical encoder(Fig. 7.5) with properties:

- **Max. allowable revolution**: 5000rpm
- **Resolution**: 2500 PPR
- **Power supply**: 5–24V DC

Figure 7.5: E50S8-2500-3-T-1 Optical encoder

1. **DC Motor Max RPM**: As we can see, the maximum motor speed is 15,000 RPM, while the maximum encoder speed is 5,000 RPM. Therefore, there had to be a way to adjust the speed values to be at least equal using a step-down DC-DC converter (see Fig. 7.6). Given the well-established linear relationship between voltage and motor

speed under no-load or light-load conditions, we applied the proportional equation to determine the required voltage [54].

From motor data:

$$\text{at } 24 \text{ V} \longrightarrow 15000 \text{ RPM}$$
$$X \text{ V} \longrightarrow 5000 \text{ RPM}$$
$$\therefore X = 8\text{V}$$

Figure 7.6: Step-down DC-DC converter

So it is assumed that at a value of 8 volts the motor speed becomes equal to the encoder speed, and thus we can connect them together without fear of excessive speed on the encoder.

PPR - Pulses Per Revolution: As we can see to maintain high measurement accuracy, we used an encoder with a large PPR value (2500) Fig. 7.7

Signal Type (Incremental vs Absolute): Because we need to measure speed, we used Incremental.

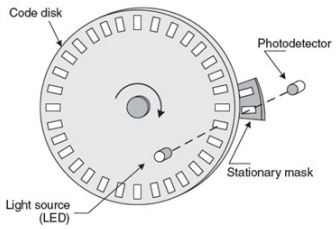


Figure 7.7: Incremental encoder

Mechanical composition (Mounting & Shaft): To transfer the mechanical motion of the motor to the encoder, we used helical beam coupling or spiral coupling as shown in Fig. 7.8.

Then, using Arduino and connecting it to MATLAB to display the results, the RPM values appeared to be very close to what was expected in Appendix 1, 2.

PWM (Pulse Width Modulation)

To precisely control the voltage, we used MD10C R3.0 10Amp DC Motor Driver(Fig. 7.9) Similarly, we programmed it using Arduino and controlled the PWM value to get the required speed. In Appendix 3

Figure 7.8: DC capler encoder connection

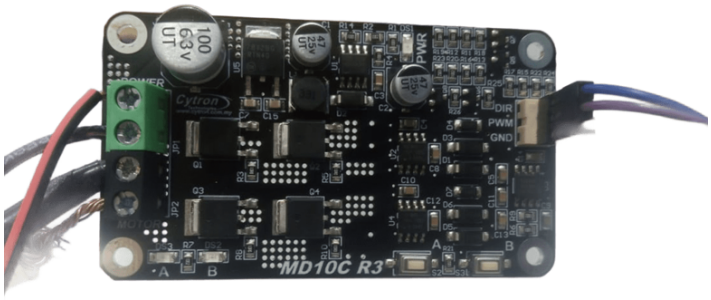


Figure 7.9: MD10C R3 motor driver

It showed largely logical and acceptable value(Fig. 7.10):

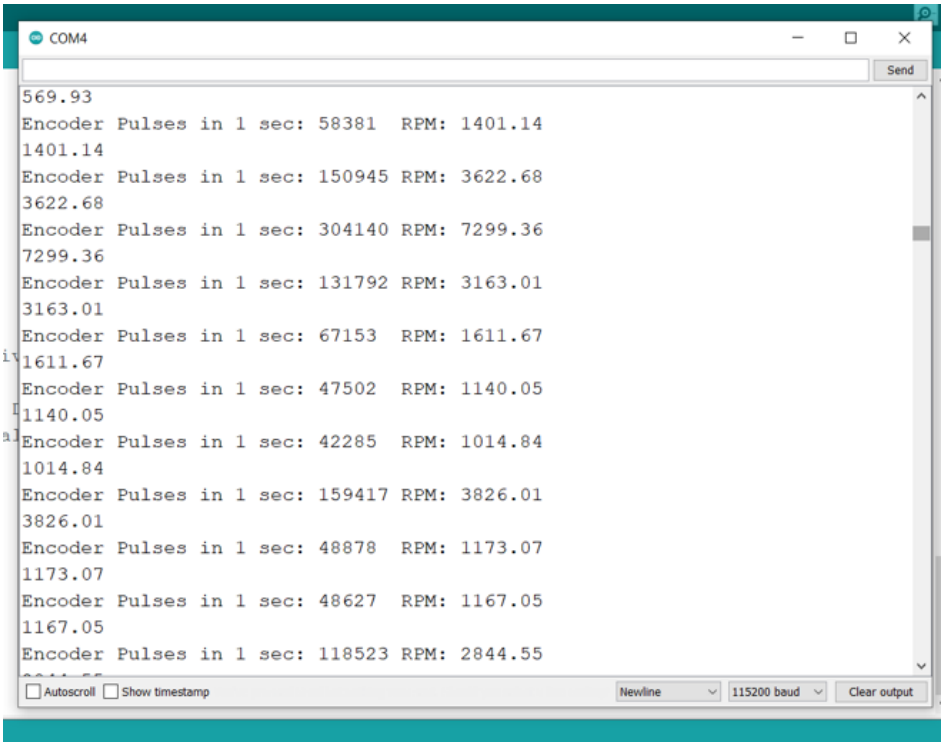


Figure 7.10: PWM values

and in this way we have arrived at an excellent practical method for measuring speed.

7.0.2 Servo motors

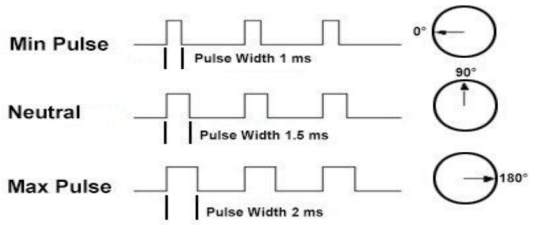
Servo motors are specialized motors with a feedback mechanism allowing precise control of angular or linear position, speed, and acceleration. They're used to operate remote-controlled or radio-controlled toy cars, robots, and airplanes. These motors are also used in industrial applications, robotics, in-line manufacturing, pharmaceutics, and food services. In our work, We are using FT5835M Servo Motor 180°(Fig. 7.11). Servo motors have three control methods: speed control, torque control, and position control. For the speed control and torque control, they are controlled by analog quantities. For the position control, it is controlled by sending pulses. The specific control method used in our project is the speed control [55]. Servo motors incorporate a sophisticated control system that includes sensors for precise position feedback, enabling exact control of the motor's angular or linear position. The inclusion of feedback mechanisms in servo motors allows for dynamic adjustment during operation, reducing errors and enhancing performance efficiency.

Figure 7.11: FT5835M Servo Motor 180°

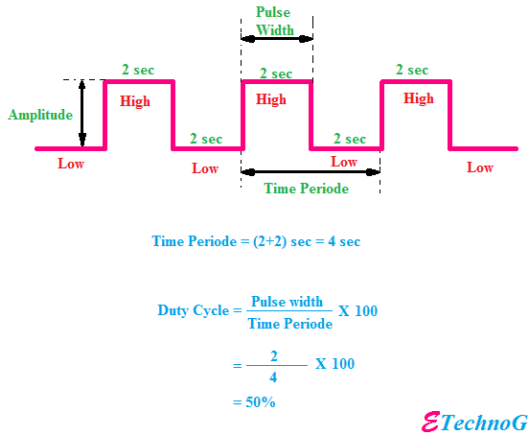
How Servo Work

Servos are controlled by sending an electrical pulse of variable width, or pulse width modulation (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate(Fig... 7.12a). A servo motor can usually only turn 90° in either direction for a total of 180° movement. Pulse Width Modulation (PWM)is a technique of producing varying analog signals from a digital source. In this context, the Duty Cycle indicates what percentage of the time a signal is ON (at logical high - high voltage) (Fig. 7.12b). The motor's neutral position is defined as the position where the servo has the

same amount of potential rotation in both the clockwise or counter-clockwise directions. To better understand what a servo motor is, the PWM sent to the motor determines the position of the shaft, and based on the duration of the pulse sent via the control wire, the rotor will turn to the desired position[56]. The servo motor expects to see a pulse every 20 milliseconds (ms), and the length of the pulse determines how far the motor turns. If you're still wondering how does a servo motor work, here's an example - a 1.5ms pulse will make the motor turn to the 90° position. Shorter than 1.5ms moves it in the counterclockwise direction toward the 0° position, and any longer than 1.5ms will turn the servo in a clockwise direction toward the 180° position[57]. FT5835M Servo Motor 180 Connections,



(a) PWM



(b) Duety Cycle

Figure 7.12: Overall caption for the two images

Servos motors come with three wires:

- **Red** – DC Voltage (Typically 5V–6V). Connect to the ESP32 5V pin or to an external power source.
- **Orange** – Signal. Connect to a PWM-capable ESP32 pin.
- **Brown** – GND (Ground). Connect to ESP32 GND.

All codes used for the FT5835M Servo Motor can be found in Appendex **??**

7.0.3 IMU

MPU-6050

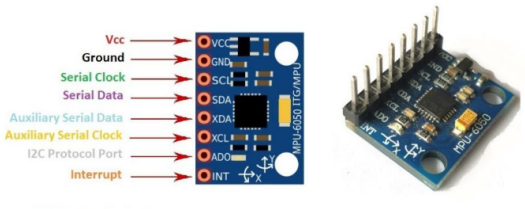
The MPU-6050 IMU (Inertial Measurement Unit) is a 3-axis accelerometer and 3-axis gyroscope sensor. The accelerometer measures the gravitational acceleration, and the gyroscope measures the rotational velocity. This sensor is ideal to determine the orientation of a moving object. This sensor provides data over the I2C protocol.

MPU6050 module (Fig. **??**) consists of a digital motion that performs all complex

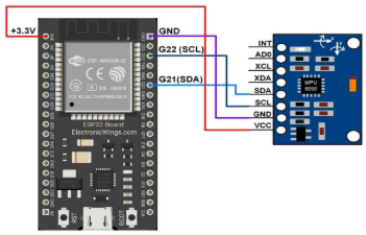
Table 7.2: Connection between MPU6050 Module and ESP32 Module

MPU6050 Module	ESP32 Module
VCC	3.3V
GND	GND (common ground)
SCL	GPIO22 (I2C SCL)
SDA	GPIO21 (I2C SDA)

processing, computations, and provides sensor data output to other MCUs over I2C communication. This sensor is based on MEMS (Micro-Mechanical Systems) technology, hence it is extremely compact and accurate in certain frequencies. It has a relatively small size and low power consumption.



(a) MPU6050



(b) **ESP32 interfaced with MPU6050**

The ESP32 has two I2C bus interfaces that can serve as an I2C master or slave. IC means Integrated Circuit (it's pronounced I-squared-C), and it is a synchronous, multi-master, multi-slave communication protocol. The I2C communication protocol uses two wires to share information. One is used for the clock signal (SCL) and the other is used to send and receive data (SDA). we will interface MPU6050 with ESP32 using Arduino IDE (Fig. 7.13b).

The VCC pin is connected to the 3.3V from the ESP32 module to power up. Both the grounds of the two devices are connected in common. The SCL pin of MPU6050 is connected to the default SCL pin of ESP32. Likewise, the SDA pin is connected with the default SDA pin of the ESP32 Table. **??**. Once the code is uploaded to ESP32, open the serial monitor of Arduino IDE and set the baud rate to 115200. Finally, we can see the MPU-6050 readings on the Arduino serial monitor. Change MPU-6050 sensor direction, rotation and orientation to observe the change in values: MPU6050 Sensor Readings, Accelerometer, Gyroscope, and Temperature Arduino serial monitor.ESP32 Display MPU6050 Readings on Serial Plotter[58].

3 Axis Digital Compass (QMC5883L)

This 3-axis digital compass is based on the Honeywell [59] , QMC5883L sensoris a type of navigation equipment that evaluates the strength of the magnetic field or the magnetic dipole moment. The magnetometer or digital compass sensor is used to measure the heading angle (angle from the north); this is done by measuring both the direction and the magnitude of Earth's magnetic fields, from milli-gauss to 8 gauss. Communication with the QMC5883L is simple and all done through an I2C interface. Voltage of 2.16-3.6VDC should be supplied. The http://www.farnell.com/datasheets/1683374.pdfHMC5883L is a surface-mount, multi-chip module with a digital interface for applications such as low-cost compassing and magnetometry. It communicates via https://www.teachmemicro.com/i2c-primer/I2Cand is able to provide magnetic flux density in three axes. HMC5883L module has five pins as shown Fig. 7.14:

Figure 7.14: Pins of the QMC5883L module

- **Vcc**: Connect a 5V DC supply to this pin.
- **GND**: Connect ground to this pin.
- **SCL**: Connect the serial clock out from the master device to this pin.
- **SDA**: Connect the serial data line from the master device to this pin.
- **DRDY**: Data Ready status signal pin output from module to the master device.

The QMC5883L magnetometer, which operates using the I2C communication protocol, interfaces with the ESP32 via its SDA (GPIO21) and SCL (GPIO22) pins, drawing its 3.3V power supply directly from the ESP32's 3.3V output. This sensor is characterized by a sensitivity of ±0.88 mG per digit, a field range of ±8.1 Gauss, and adjustable data output rates up to 15 Hz, with an operating current of 130 μA [60]. Initially, the heading is determined by the arctangent of the ratio of magnetic flux densities in the Y and X axes, as shown in Eq. 7.2. However, magnetometers detect magnetic north, not true north, leading to

an error known as magnetic declination, which varies by location[61]. Therefore, to obtain the correct true north heading, this declination must be taken into account, as expressed in Eq. 7.2653 Axis Digital Compass (QMC5883L)declinationeq:heading_declination

$$heading = \arctan\left(\frac{Y_{mag}}{X_{mag}}\right) \tag{7.2}$$

$$heading = \arctan\left(\frac{Y_{mag}}{X_{mag}}\right) - \text{declination angle} \tag{7.3}$$

Magnetometer(QMC5883L)

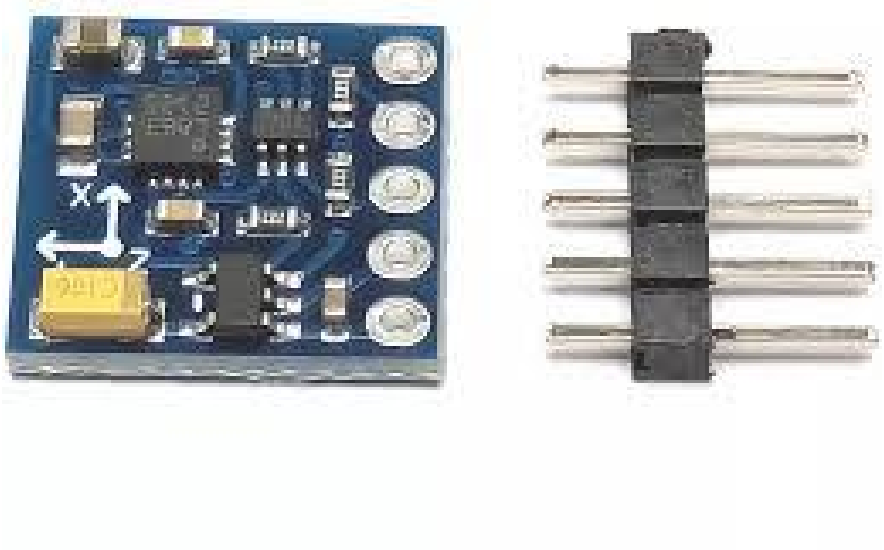


Figure 7.15: Enter Caption

The **QMC5883L** (Fig. **??**)is a high-precision, three-axis digital magnetometer designed to measure the Earth's magnetic field in three dimensions. It is widely used in navigation, robotics, UAVs, and space applications due to its compact design, high sensitivity, low power consumption, and ease of integration with microcontrollers like Arduino. Built on Anisotropic Magneto-Resistive (AMR) technology, the QMC5883L offers accurate magnetic vector measurement with a 16-bit ADC and internal signal conditioning. In systems such as satellite attitude control using Variable Speed Control Moment Gyroscopes (VSCMGs) and AI-based reinforcement learning, QMC5883L provides crucial absolute heading information that enhances the accuracy and stability of sensor fusion processes.

Sensor Features and Specifications:

· **Sensor Technology:** 3-axis AMR magnetometer with 16-bit resolution, operating in compact 3x3x0.9mm LGA package.

· **Measurement Range:** Programmable ±2G or ±8G full-scale range.

· **Sensitivity:** ~12000 LSB/G for ±2G and ~3000 LSB/G for ±8G.

· **Accuracy:** ~1–2 degrees heading resolution under normal conditions.

· **ADC Resolution:** 16-bit low-noise digital output.

· **Power Supply:** 2.16V to 3.6V operating voltage.

· **Interface:** I2C digital interface (default address: 0x0D).

· **Power Consumption:** ~75μA during continuous measurement.

· **Extra Features:** Built-in temperature compensation and adjustable output data rates (10Hz–200Hz).

These features make QMC5883L suitable for low-power embedded systems that require reliable heading data.

Role in GNC and Sensor Fusion:

In Guidance, Navigation, and Control (GNC) systems, the magnetometer serves as an absolute heading reference. The QMC5883L measures Earth's magnetic field vector and is commonly fused with accelerometer and gyroscope data from sensors like the **MPU6050** using algorithms such as the **Kalman Filter** or **Complementary Filter**. This fusion helps correct gyroscopic drift and accelerometer noise, improving real-time orientation estimates. In satellite control systems using VSCMGs and AI, this heading data contributes to more stable and adaptive attitude control during complex maneuvers.

Importance of Calibration and Compensation Techniques

Magnetometer readings are prone to errors from:

· **Hard-Iron Distortion:** Constant magnetic offset caused by nearby magnets or current-carrying wires.

· **Soft-Iron Distortion:** Field distortion from ferromagnetic materials that scale or skew the magnetic vector.

· **Sensor Bias and Misalignment:** Intrinsic bias or misaligned axes cause inaccuracies.

To correct these, calibration is mandatory. This usually involves collecting raw magnetic data while rotating the sensor in all directions and fitting an ellipsoid to this dataset. From this, we compute:

· The **hard-iron offset vector** (bias).

· The **soft-iron transformation matrix** (scaling and alignment correction).

On Arduino, a simplified calibration method is used:

1. Record max/min values on each axis during full 3D rotation.

2. Estimate offset as: offset_X = (X_max + X_min)/2

3. Apply offset compensation to all raw readings.

4. Optionally normalize axis scales.

Arduino-Based Calibration Implementation In Appendix 5 is a simplified Arduino sketch for QMC5883L reading and offset calibration

Real-World Limitations and Considerations:

Despite calibration, QMC5883L may still be affected by:

· **Dynamic magnetic interference:** Moving metal objects or fluctuating currents.

· **Environmental changes:** Temperature shifts, nearby electronics.

· **Sensor alignment drift:** Vibration and mounting error over time.

· **I2C noise or voltage mismatch:** Arduino boards running at 5V may require a logic level shifter.

In mission-critical applications like satellite control using VSCMGs and AI, periodic recalibration and robust sensor fusion algorithms are necessary to maintain reliability.

Results and Discussion:

Calibration tests were performed by rotating the sensor in all orientations and logging the raw magnetic data. The raw output showed a distorted ellipsoid shape due to hard and soft iron interference. After applying the offset and optional scale calibration, the data became more symmetrical and centered.

· **Before calibration:** Raw magnetometer plots displayed shifted, skewed ellipsoid shapes.

· **After calibration:** Corrected data aligned closer to a centered circle (in 2D) or sphere (in 3D).

· **Heading accuracy improvement:** Post-calibration heading drift was reduced significantly, and readings were more stable.

Suggested visualizations to include:

· A 2D scatter plot (X vs Y) before and after calibration to show data correction.

· A histogram of heading error or heading stability.

· Optional: 3D plot of raw and corrected data using MATLAB or Python (matplotlib).

These results demonstrate the importance of proper calibration for ensuring reliable heading estimation when using QMC5883L in high-precision GNC applications. 7.16

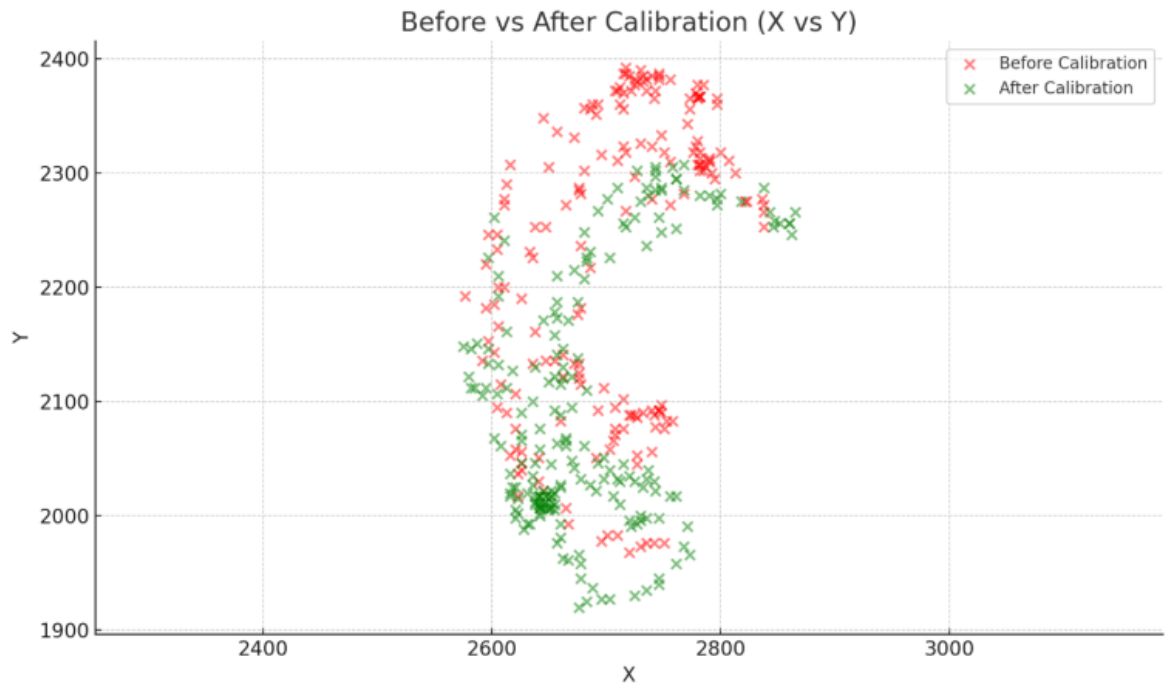


Figure 7.16: Enter Caption

Red – Before Calibration · The data is scattered unevenly and appears shifted away from the center.

· There's a noticeable distortion in the distribution shape, which is expected due to:

o Hard-iron bias (caused by permanent magnetic fields nearby),

o Soft-iron distortion (caused by ferromagnetic materials affecting the field),

o And sensor misalignment.

Green – After Calibration · The data is now more centered and symmetrical.

· The spread is tighter, with improved circular (or elliptical) distribution.

· There's a clear improvement in overall structure, indicating that calibration has been successful.

7.0.4 Microcontroller:

In this project, which focuses on advanced satellite attitude control using Variable Speed Control Moment Gyroscopes (VSCMG) and Reinforcement Learning (RL), the ESP32 microcontroller (Fig. 7.17) was selected as a core embedded system component due to its unique combination of high performance, low power consumption, and built-in wireless communication capabilities.

The ESP32 features a powerful dual-core processor, real-time task handling, and multiple hardware interfaces (PWM, ADC, I2C, SPI, UART), making it ideal for controlling motors,

reading sensor data, and executing lightweight control algorithms. Its support for Wi-Fi and Bluetooth also provides a convenient way to log data, update firmware remotely, or monitor system behavior during simulations or ground testing.

Moreover, the ESP32's flexibility and affordability make it a practical choice for prototyping complex systems like VSCMG-based controllers, especially when integrating reinforcement learning agents that require real-time feedback and decision-making. It allows for quick iteration, seamless hardware interfacing, and reliable performance in embedded control applications.

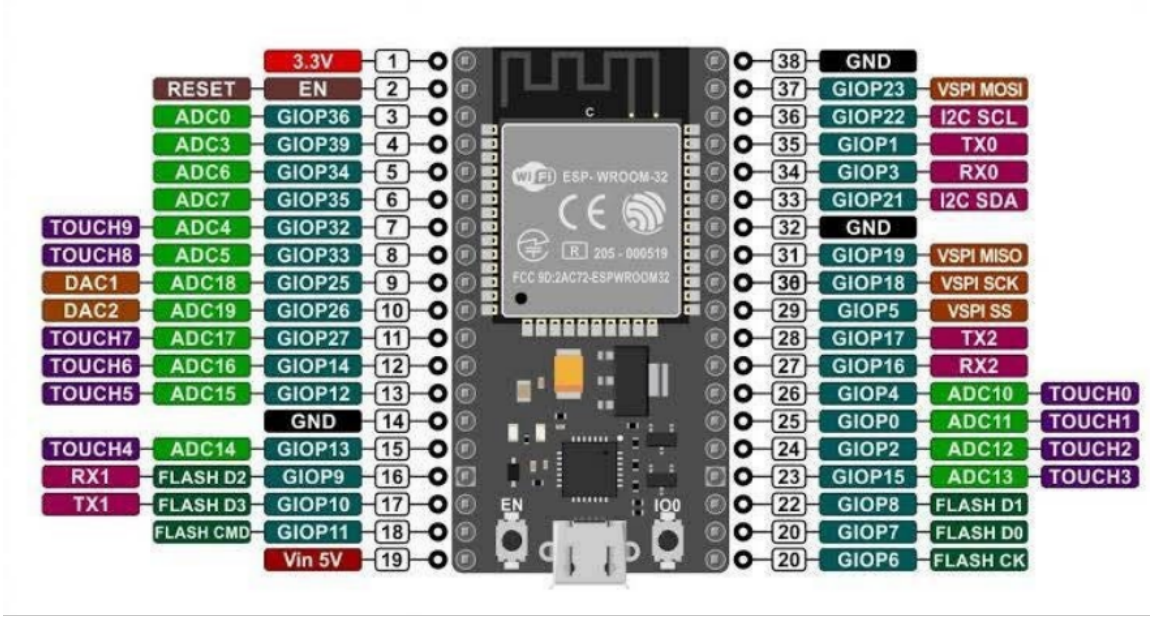


Figure 7.17: ESP32

Finally, we have assembled all the components into one integrated circuit to run the project.. 7.18

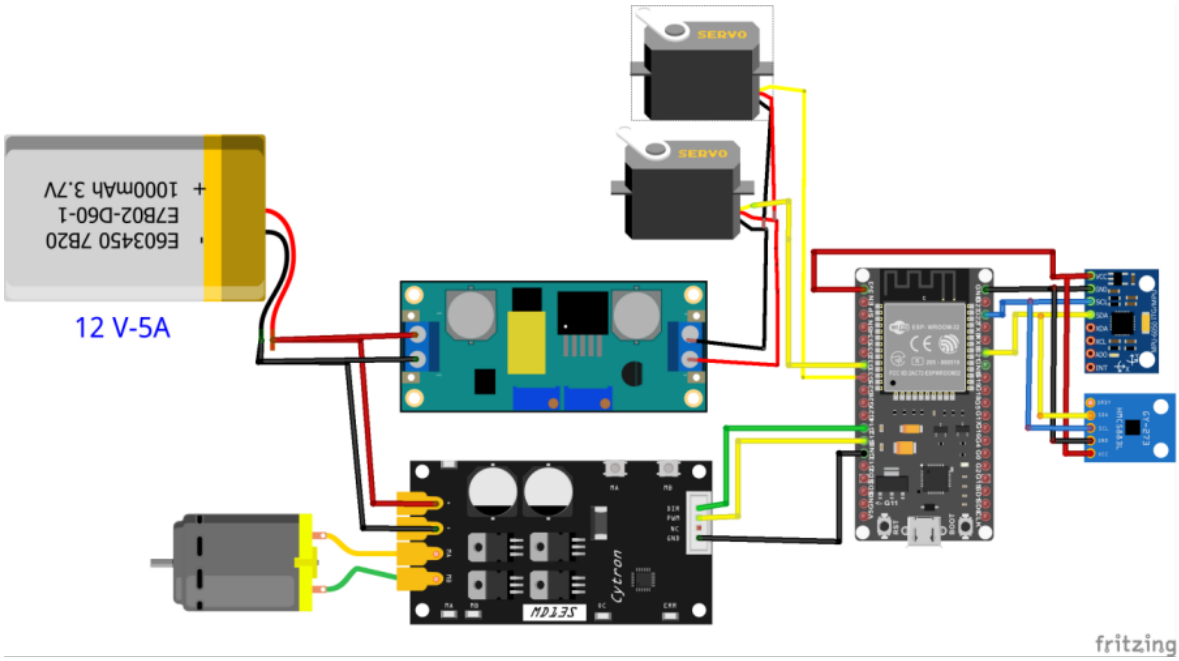


Figure 7.18: Circuit Connection

Chapter 8: ELECTRICAL POWER SYSTEM

In this chapter electronics architecture and power system for this project are briefly explained. To achieve the target, the project requires the following components: 2 DC motors, a motor driver, 2 servo motors, an MCU, an IMU sensor, magnetometer sensor. And these components must work properly to do their functions; so based on chosen types and their electrical characteristics, the power is distributed and connections are made(Fig... 8.1).

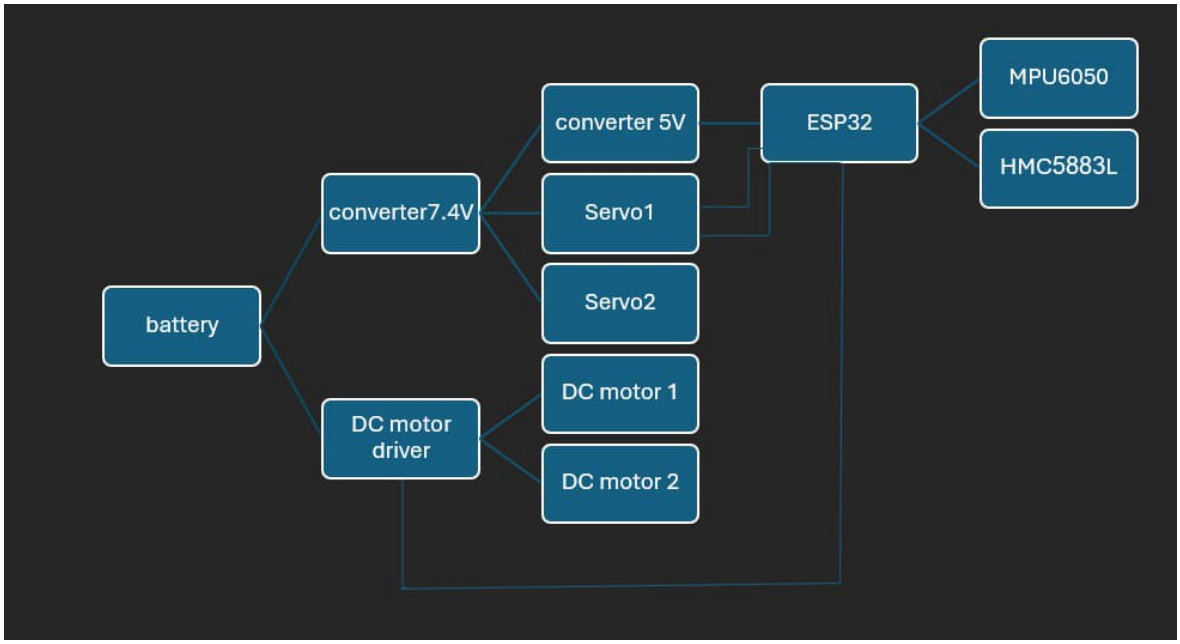


Figure 8.1: power system

8.1 Power budget

Component	no.	Voltage (V)	Current (A)	Energy (watt)
DC motor	2	5	0.6	6
Servo motor	2	7.4	1	14.8
Esp32	1	3.3	0.1	0.33
MPU6050	1	3.3	0.004	0.0132
HMC5883L	1	3.3	0.0001	3.3×10^{-4}
MD10C driver	1	3.3	-	-

Table 8.1: Power consumption of different components

Total energy= 21 watt (Table 8.1)

8.2 Battery sizing

Since the maximum voltage in the circuit is 7.4V, we can make assumptions to choose batteries with appropriate capacity.

Let it be 12V

Capacity = 21 / 12 = 1.75Ah =1750 mAh

The used batteries are preferred to have larger value.

18650 li-ion battery from SONY is a good choice (Fig. 8.2),

Figure 8.2: 3.V 18650 battery cell

It comes with:

- 3.7V, so 3 cells of it can provide sufficient voltage (11.1V)

- High capacity 3000mAh
- High continuous discharge rate to afford the current needed
- 30 A maximum discharge current so it can handle the peak moments

8.3 PMS

Power management system (Fig. 8.3)is used:

- 3 series cell configuration
- 11.1V nominal voltage
- 20 A continuous discharge rate

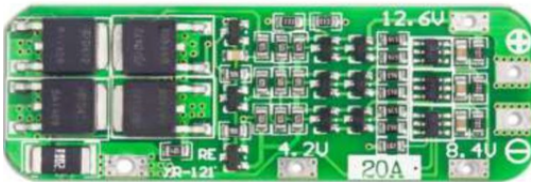


Figure 8.3: PMS

It provides:

- **Overcharge Protection**: Prevents cells from exceeding safe voltage levels, enhancing battery lifespan.
- **Over-discharge Protection**: Stops discharge when cells drop too low, preventing damage.
- **Over-current Protection**: Limits current to protect against excessive loads.
- **Short Circuit Protection**: Disconnects the battery in case of a short circuit for safety.
- **Cell Balancing**: Ensures all cells charge evenly, improving efficiency and longevity.
- **Auto Recovery**: Resumes normal operation after a protection event is resolved.

8.4 motor driver

For DC motors, a motor driver is needed(Fig. **??**).

MD10C Cytron is suitable as it:

Supports motor voltage ranges from 5V to 30VDC.

Maximum current up to 13A continuous and 30A peak.

11.1V comes from battery, then it helps control the speed so only 5V reaches the motors.

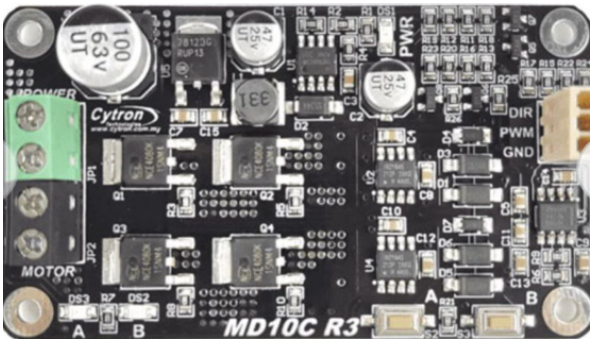


Figure 8.4: dc motor driver

8.5 Servo Converter

For servo motors, each needs 7.4V so a step down buck converter is used (Fig. **??**)

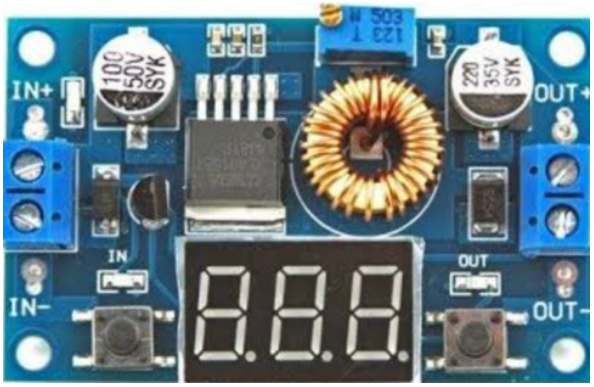


Figure 8.5: Step-down buck converter

-Input voltage range 4.8V-38V

-Output voltage range 1.25-36VDC adjustable

-Output current: 0-5A

It takes 11.1V from batteries and provide 7.4V adjusted to servo motors.

8.6 PCB Power

7.4V from first converter is also an input power for the PCB.

On PCB, ESP32 MCU needs only a 5V input, so we step down voltage again. Fig. **??**

It provides 5V to ESP32 from 7.4V IN With output current up to 3A (sufficient for MCU and sensors).

ESP32 itself provides a 3.3V output, so both MPU6050 and QMC5883L sensors take their input voltage directly from it.

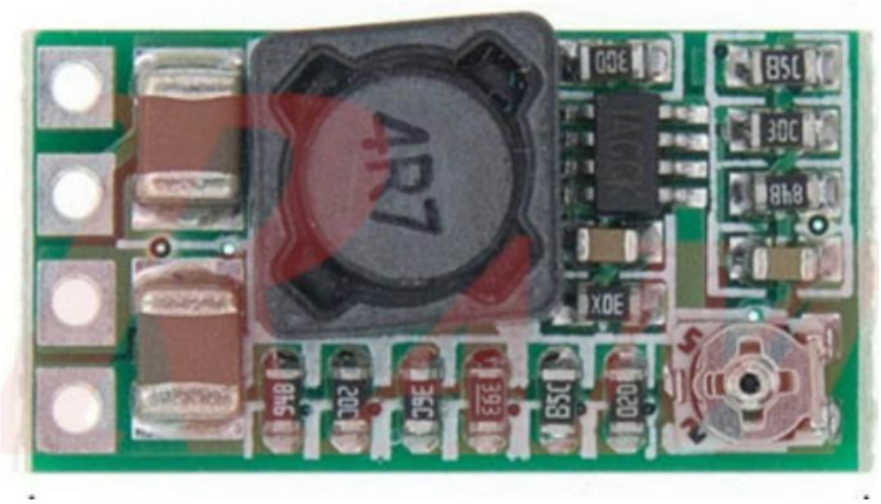


Figure 8.6: Step-down converter

8.7 PCB Architecture

For electrical stability and system reliability, the need for a printed circuit board (PCB) was crucial. A PCB ensures stable and robust connections between system components. Based on the connection from the embedded team, a schematic sheet is: Fig. **??**

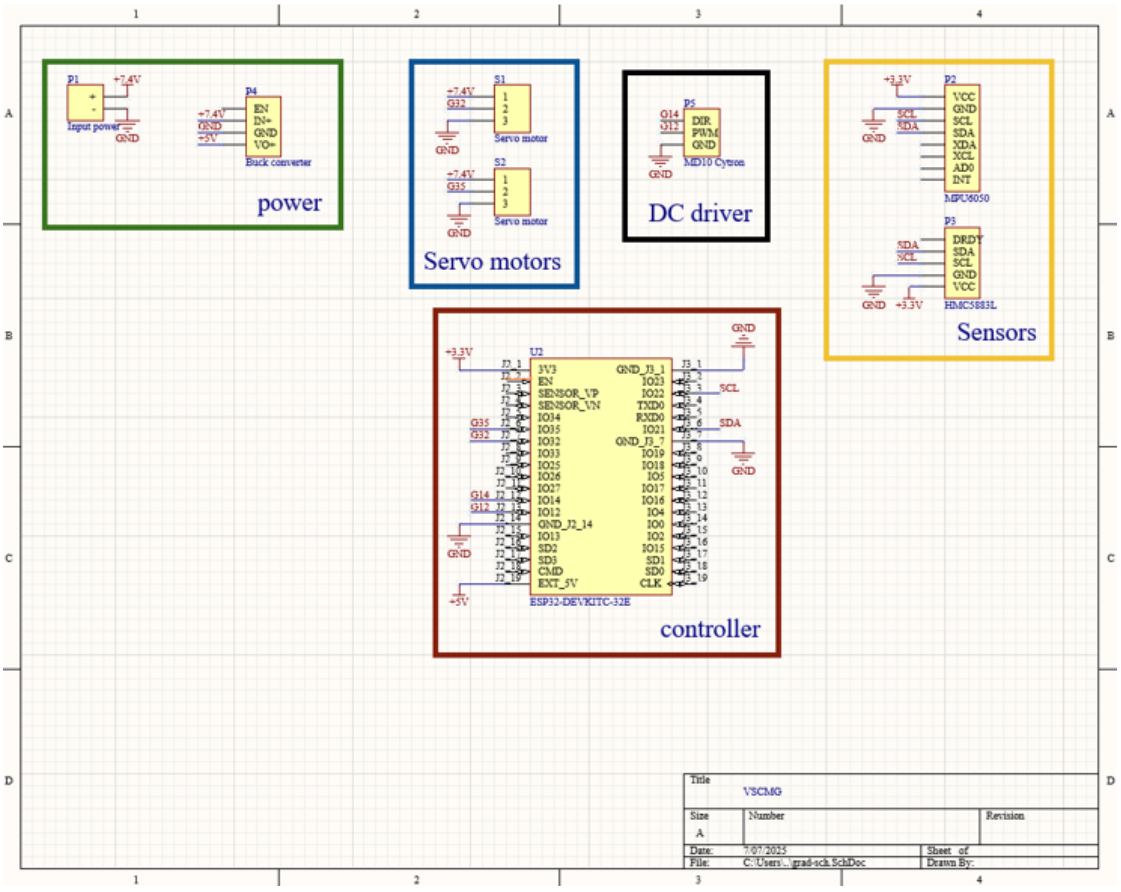


Figure 8.7: PCB schematic

This schematic was updated to a layout design: Fig. 8.8, 8.9

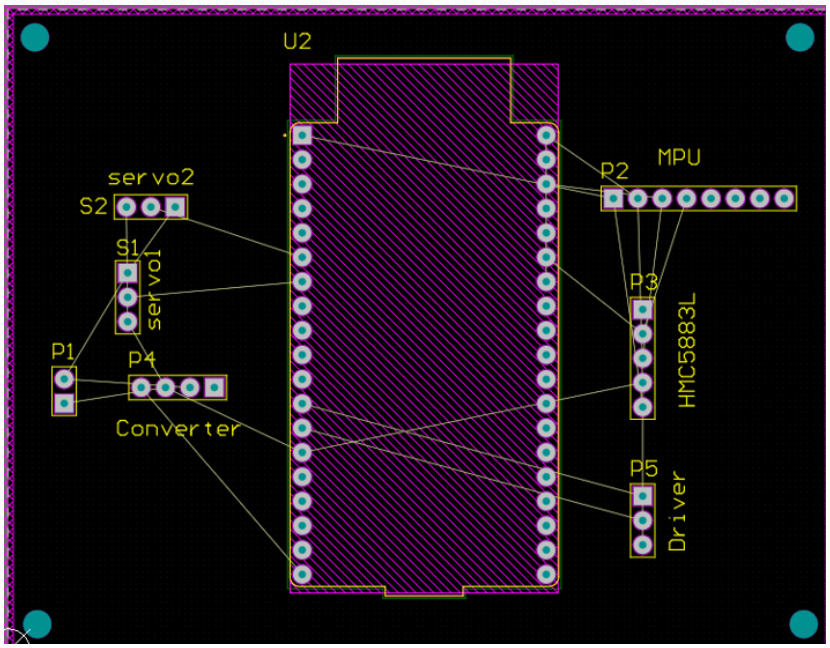


Figure 8.8: placing components

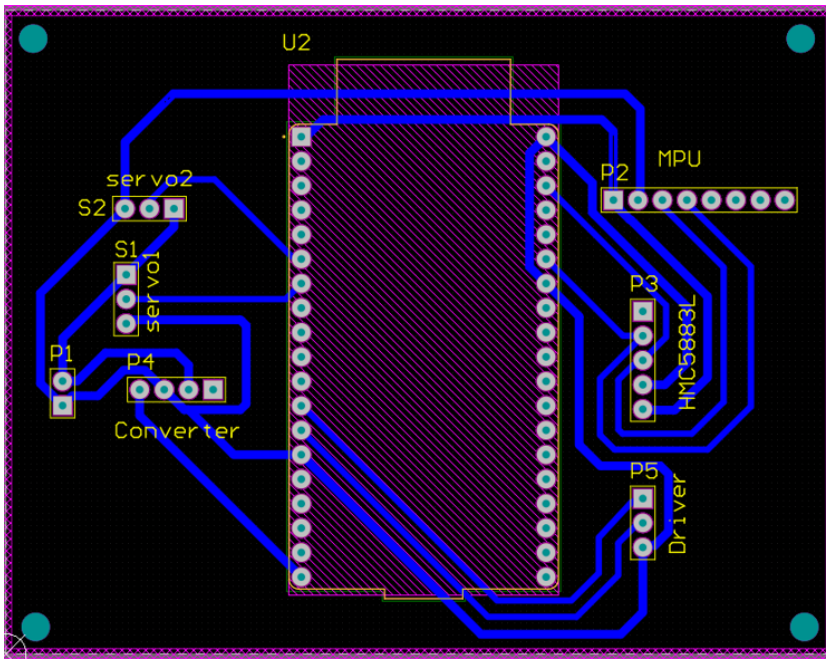


Figure 8.9: 2D PCB design

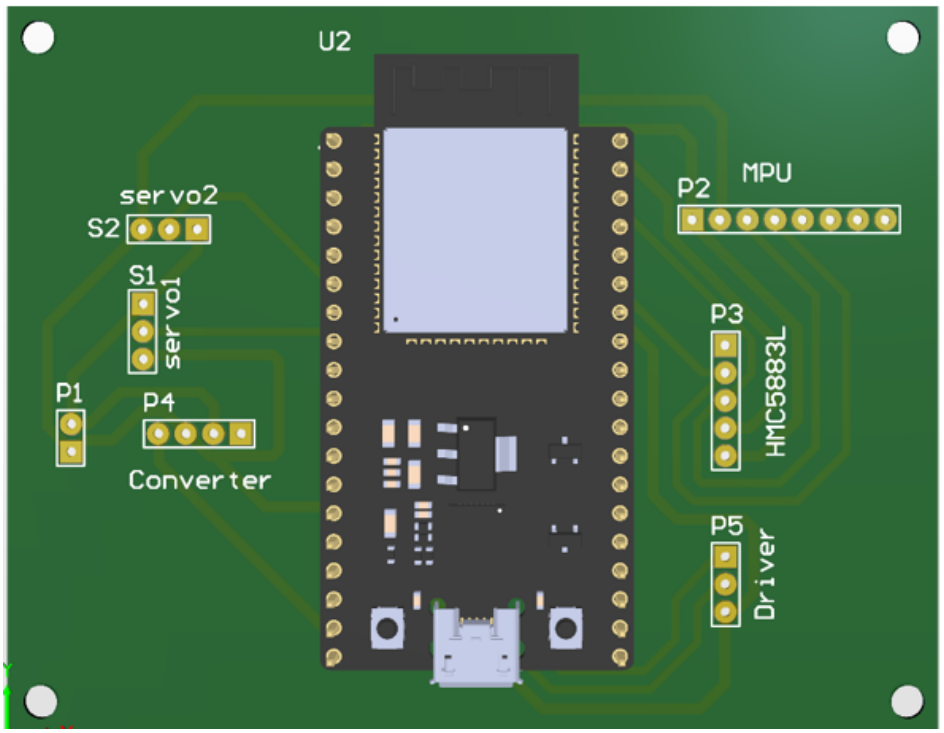
Designed considering:

- power traces to be wider and shorter.

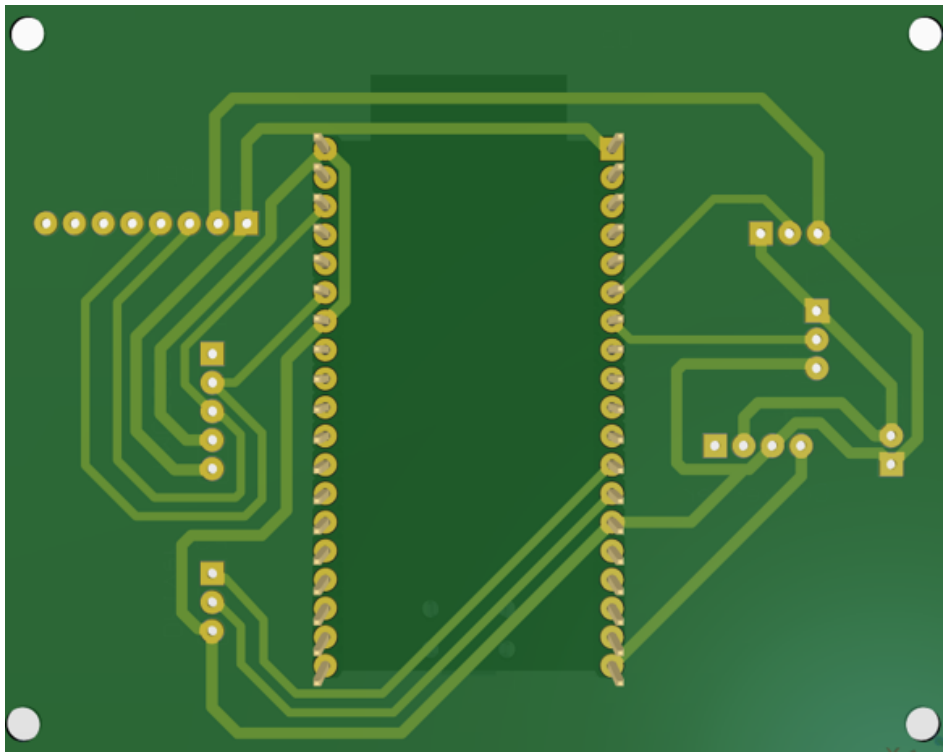
- suitable areas for housing the components.

- single layer to low cost.

3D PCB layout in Fig. 8.10a, 8.10b:



(a) 3D layout front



(b) 3D layout back

PCB polygon pour added (Fig. 8.11):

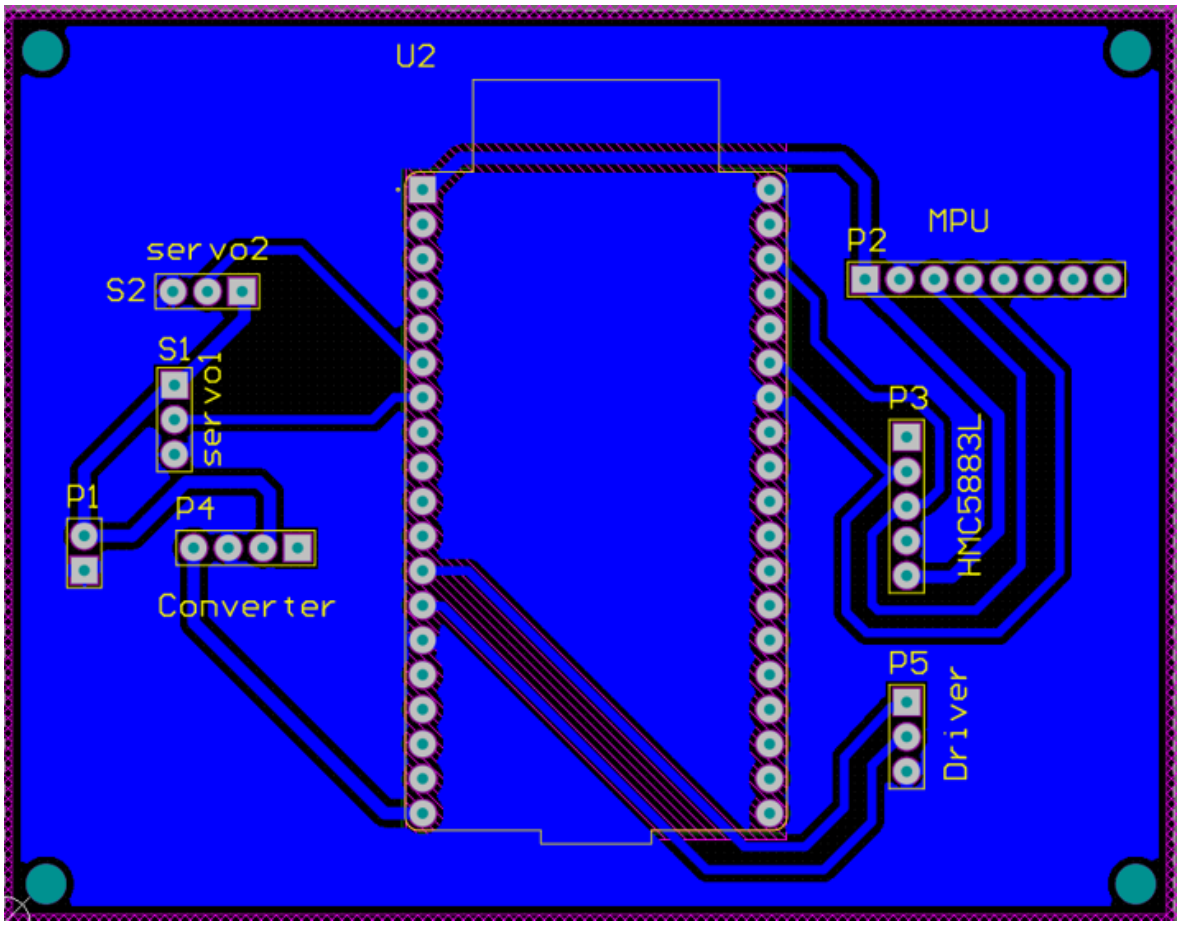


Figure 8.11: polygon pour

It helps:

- Enhanced Power Distribution

- Improved Signal Integrity: Ground planes act as a stable reference for signals, helping to reduce Electromagnetic Interference (EMI) and crosstalk between traces.

-Heat Dissipation: Large copper areas created by polygons help distribute and dissipate heat away from high-heat-generating components.

PCB was prepared to CNC manufacturing so gerber files (Fig. 8.12)were extracted:

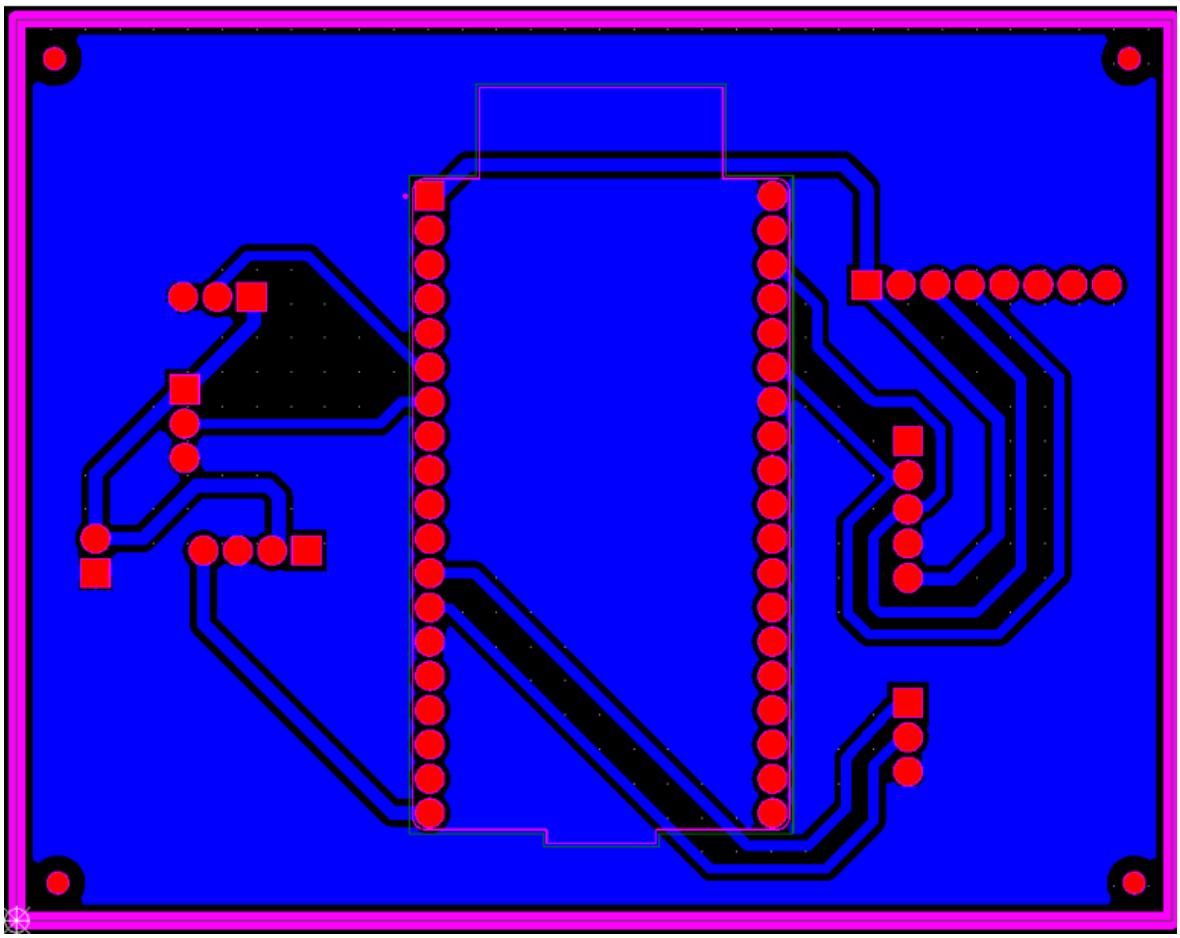


Figure 8.12: gerber file

This is the implemented PCB

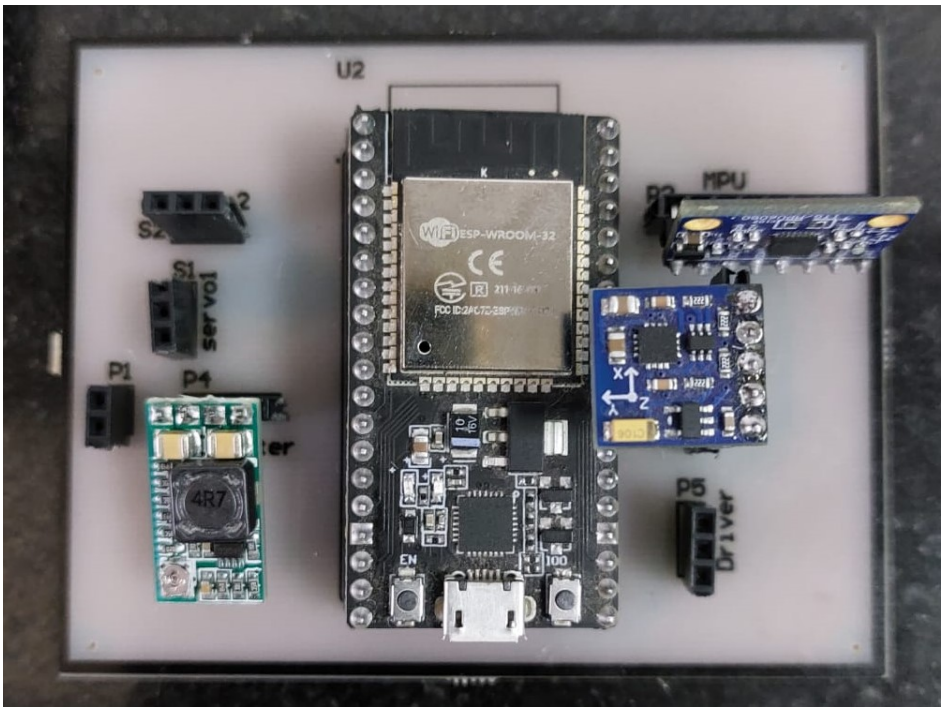


Figure 8.13: PCB hardware

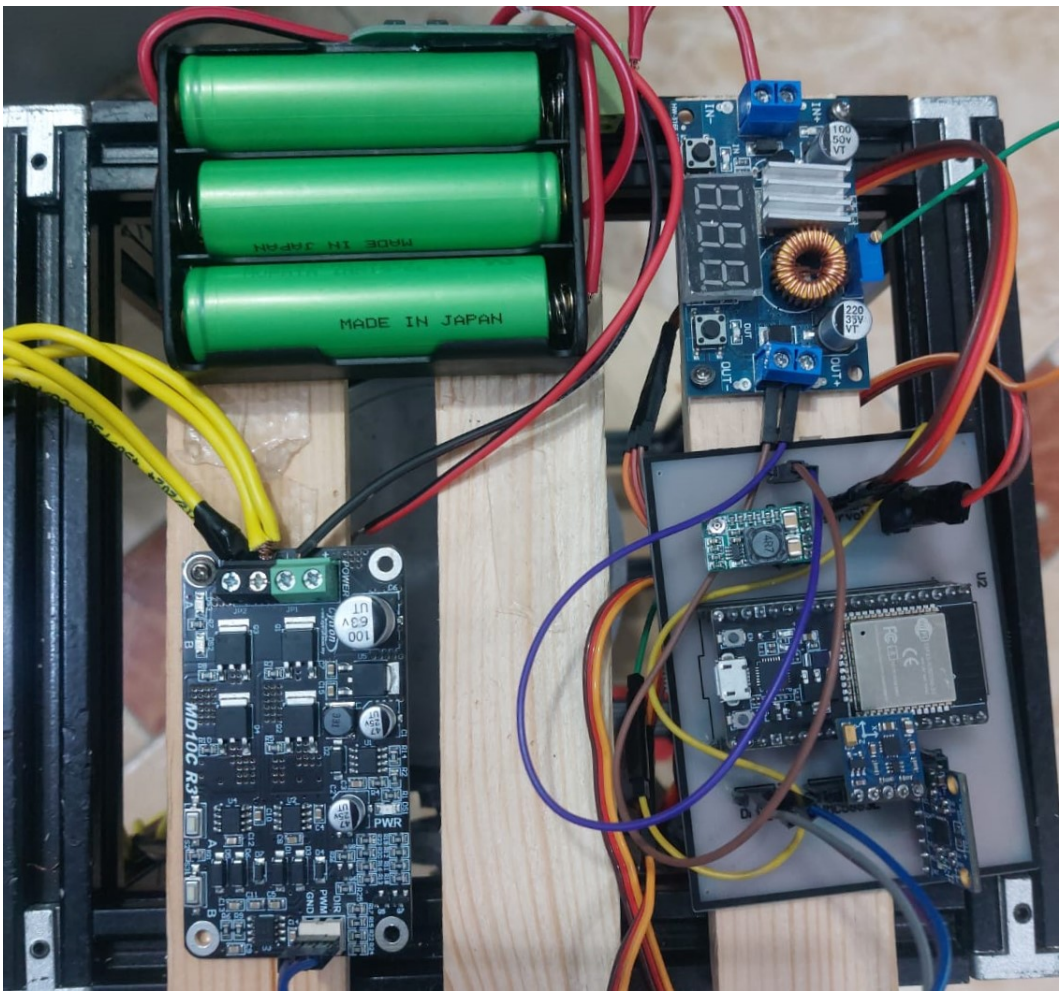


Figure 8.14: Full Assembly

Chapter 9: Navigation

9.1 Sensor fusion

Nature has found a way to integrate information from multiple sources to a reliable and feature-rich recognition. Even in the case of sensor deprivation, systems are able to compensate for lacking information by reusing data obtained from sensors with an overlapping scope. Humans, for example, combine signals from the five body senses (sight, sound, smell, taste, and touch) with knowledge of the environment to create and update a dynamic model of the world. Based on this information the individual interacts with the environment and makes decisions about present and future actions. This natural ability to fuse multi-sensory data has evolved to a high degree in many animal species and has been in use for millions of years. Today, the application of fusion concepts in technical areas has constituted a new discipline that spans over many fields of science. Sensor Fusion: is the combining of sensory data or data derived from sensory data such that the resulting information is in some sense better than would be possible when these sources were used individually.[65] The data sources for a fusion process are not specified to originate from identical sensors. Direct fusion means the fusion of sensor data from a set of heterogeneous or homogeneous sensors, soft sensors, and history values of sensor data, while indirect fusion uses information sources like prior knowledge about the environment and human input. Therefore, sensor fusion describes direct fusion systems, while information fusion also encompasses indirect fusion processes. So, sensor fusion is combining two or more data sources in a way that generates a better understanding of the system. This means that the solution will be more consistent, accurate, and dependable. Data source can be a sensor or a mathematical model. Perceive step is responsible for two things:

1. **Self-awareness**: localization & positioning.
2. **Situational awareness**: detection & tracking.

Sensor fusion is represented in the sense and perception steps as it has a hand in both of these capabilities. It is the process of taking multiple sensor measurements(Fig. **??**), combining them, and mixing additional information from mathematical models with a goal of having a better understanding of the world with which the system can be used to plan and

act.

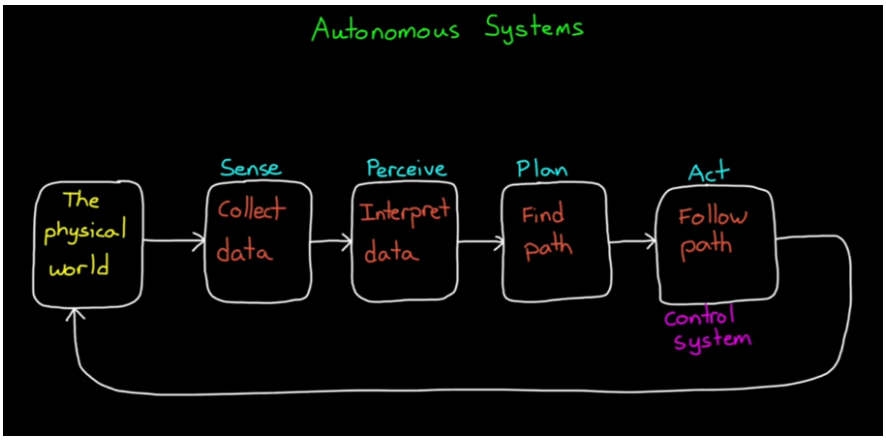


Figure 9.1: The process of taking multiple sensor measurements

There are four ways that the sensor fusion can help us to better localize and position in our own system, as well as detect and track other objects:

1. **It can increase the quality of the data, ”clean data”.**
2. **It can increase reliability.**
3. **It can estimate unmeasured states.**
4. **It can increase the coverage area.**

As an Example, to Sensor fusion is to estimate orientation Attitude heading reference system (AHRS): To define the orientation, we need:

- **reference frame**
- **specify rotation**

9.2 Magnetometer calibration

A perfect system with hard and soft iron sources.

$$\mathbf{x}_{\text{corrected}} = (\mathbf{x} \cdot \mathbf{b}) \cdot \mathbf{A} \tag{9.1}$$

Where:

- **b**: hard iron bias (3×1 vector)
- **A**: soft iron distortion (3×3 matrix)

If the magnitude of reading isn’t close to the magnitude of gravity, then clearly the system is picking up other movements, and it can’t be trusted. Now we have two different ways to estimate orientation:

Hard iron source	Soft iron source
- generates its own magnetic field. - has an actual magnet or coil. - offset by hard iron, the readings will have a larger intensity in one direction and a smaller one in the opposite direction.	- Attracted to magnet but doesn't generate its magnetic field. - like a nail or a metallic structure attracted to magnet.

Table 9.1: Comparison between Hard iron and Soft iron sources

Accelerometer + Magnetometer	Gyro
- produce absolute measurements - corrupted by common disturbances	- produces relative measurements - needs initial orientation - drifts over time

Table 9.2: Comparison between Accelerometer + Magnetometer and Gyroscope

Now, by adding GPS to the previous three sensors, we can measure orientation, position, and velocity. The goal is to go over the structure of the algorithm and show how GPS and IMU contribute. Pose estimation from asynchronous sensors: This shows how you might fuse sensors at different rates to estimate pose. An accelerometer, gyroscope, magnetometer, and GPS are used to determine the orientation and position of a vehicle moving along a circular path. Using this command: open example('shared positioning/PoseEstimationFromAsynchronousSensorsExample') 9.2

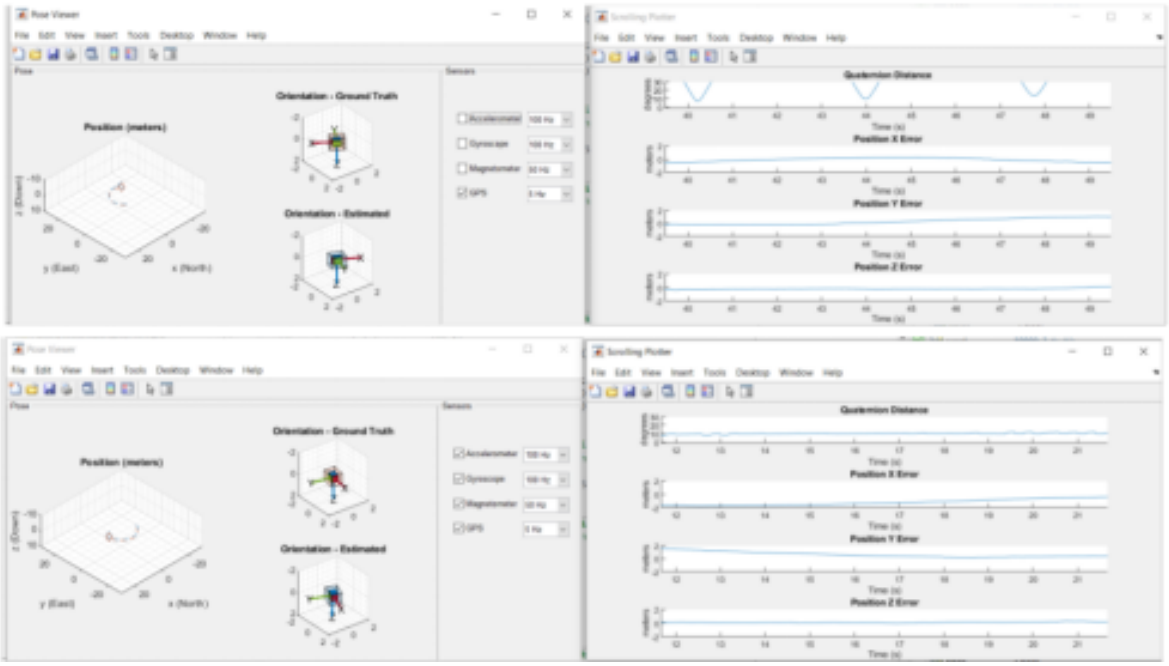


Figure 9.2: Enter Caption

Possible initialization method	state
Magnetometer + accelerometer when system is stationary	orientation
Gyro	Angular rate
Gps	Position
Gps / accelerometer	Velocity
accelerometer	Acceleration
assume zero or ground calibrated	Sensor biases
magnetometer	Mag vector

Another thing we need is to initialize the filter. In a real system, when we don't know the true states, we use sensors to initialize filters: **??**

9.3 Kalman Filtering:

9.3.1 Essential background

Estimate:

It is about evaluating the hidden state of the system. For example, the true position of the aircraft is hidden from the observer. We can estimate the aircraft's position using sensors, such as radar. The estimate can be significantly improved by using multiple sensors and applying advanced estimation and tracking algorithms (such as the Kalman Filter). Every measured or computed parameter is an estimate[62].

Accuracy:

Indicates how close the measurement is to the true value.

Precision:

Describes the variability in a series of measurements of the same parameter. Accuracy and precision form the basis of the estimate. 9.3

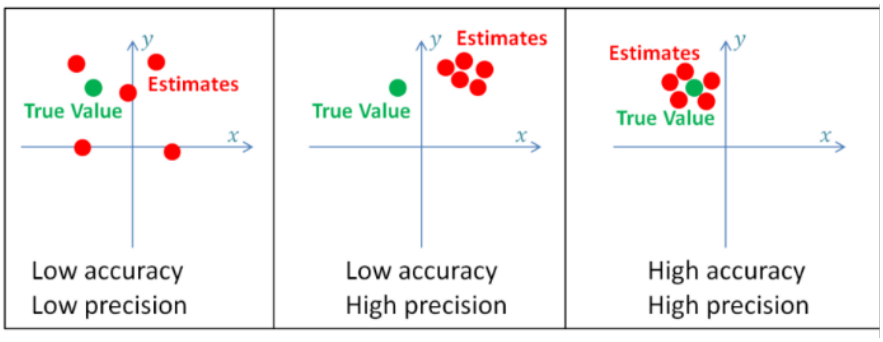


Figure 9.3: Enter Caption

High-precision systems have low variance in their measurements (i.e., low uncertainty),

while low-precision systems have high variance in their measurements (i.e., high uncertainty). The random measurement error produces variance. Low-accuracy systems are called biased systems since their measurements have a built-in systematic error (bias).

The influence of the variance can be significantly reduced by averaging or smoothing measurements. For example, if we measure temperature using a thermometer with a random measurement error, we can make multiple measurements and average them. Since the error is random, some measurements would be above the true value and others below the true value. The estimate would be close to the true value. The more measurements we make, the closer the estimate will be. On the other hand, a biased thermometer produces a constant systematic error in the estimate.

Estimation algorithm :

The goal of designing a filter is to estimate the state of a system using measurements and system dynamics. Since the measurements are usually taken at discrete time steps, the filtering process is usually separated into two steps: Prediction: Propagate state and covariance between discrete measurement time steps (k = 1, 2, 3, . . . , N) using dynamic models. This step is also called a flow update.[66] Correction: Correct the state estimate and covariance at discrete time steps using measurements. This step is also called measurement update. 9.4 **Theoretically**, the Kalman filter is an estimator for what is called

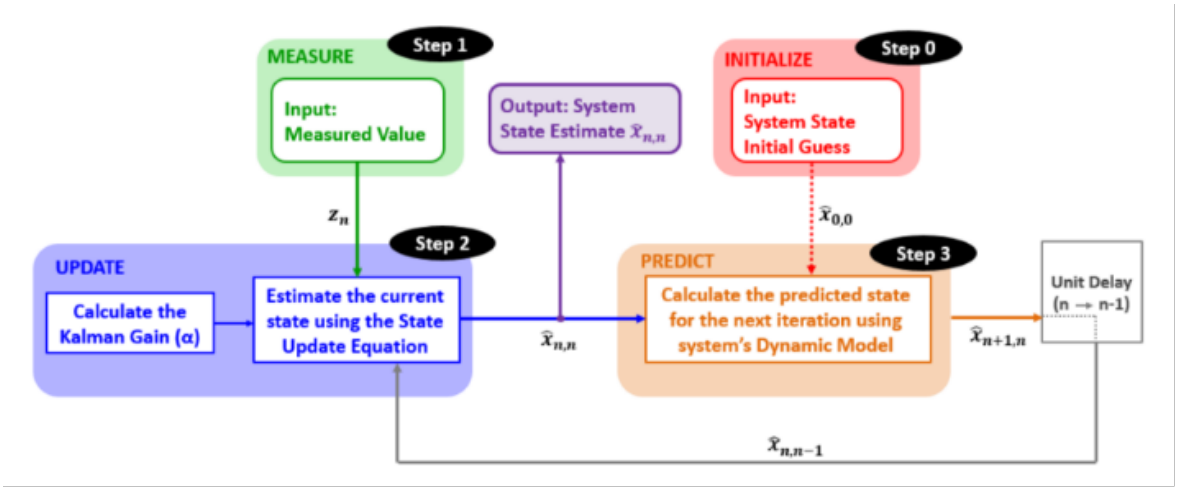


Figure 9.4: Enter Caption

the linear-quadratic problem, which is the problem of estimating the instantaneous “state” of a linear dynamic system perturbed by white noise, by using measurements linearly related to the state but corrupted by white noise. The resulting estimator is statistically optimal concerning any quadratic function of estimation error. **Practically**, the Kalman filter is one of the greatest discoveries in the history of statistical estimation theory and possibly the greatest discovery in the twentieth century. It has enabled humankind to do many things that could not have been done without it, and it has become as indispensable as

silicon in the makeup of many electronic systems. Its most immediate applications have been for the control of complex dynamic systems such as continuous manufacturing processes, aircraft, ships, or spacecraft. To control a dynamic system, you must first know what it is doing. For these applications, it is not always possible or desirable to measure every variable that you want to control, and the Kalman filter provides a means for inferring the missing information from indirect and noisy measurements. [63] The Kalman filter is also used for predicting the likely future courses of dynamic systems that people are not likely to control, such as the flow of rivers during floods, the trajectories of celestial bodies, or the prices of traded commodities.

It is only a tool. It does not solve any problem all by itself, although it can make it easier for you to do so. It is not a physical tool but a mathematical one. Mathematical tools make mental work more efficient, just as mechanical tools make physical work more efficient. As with any tool, it is important to understand its use and function before you can apply it effectively. The purpose of this book is to make you sufficiently familiar with and proficient in the use of the Kalman filter so that you can apply it correctly and efficiently.

The Kalman Filter is optimal. It is a recursive estimator that provides the optimal estimate of the state of a linear dynamic system by combining predictions from a system model with noisy measurements. It minimizes the mean squared error of the estimated states, assuming the system dynamics are linear and the noise is Gaussian. The filter operates in two steps: prediction (using the system model) and update (correcting the prediction with measurements). Key characteristics: Recursive: Processes data sequentially without needing to store all past data. Optimal: Provides the best linear unbiased estimate for systems with Gaussian noise. Applications: Used in GPS, robotics, autonomous vehicles, and financial modeling.

Kalman filter equations: We find it in the following figure **??**

The navigation component of the VSCMG project aims to accurately estimate the Yaw angle (orientation in the horizontal plane) to support precise control and navigation[64]. This is achieved through sensor fusion of data from the MPU6050 (a 6-axis Inertial Measurement Unit combining a 3-axis accelerometer and a 3-axis gyroscope) and the QMC5883L (a 3-axis magnetometer). The fusion algorithm integrates the high-frequency responsiveness of the gyroscope with the long-term stability of the magnetometer, using a Complementary Filter with tilt compensation to account for sensor misalignment and external disturbances. This documentation details the sensors, their operational principles, the fusion algorithm, the Arduino implementation, and the significance of the results for navigation.

	Equation	Equation Name	Alternative names
Predict	$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n$	State Extrapolation	Predictor Equation Transition Equation Prediction Equation Dynamic Model State Space Model
	$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$	Covariance Extrapolation	Predictor Covariance Equation
Update (correction)	$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})$	State Update	Filtering Equation
	$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\,\boldsymbol{P}_{n,n-1}(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$	Covariance Update	Corrector Equation
	$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$	Kalman Gain	Weight Equation
Auxiliary	$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n$	Measurement Equation	
	$\boldsymbol{R}_n = E\left(\boldsymbol{v}_n\boldsymbol{v}_n^T\right)$	Measurement Covariance	Measurement Error
	$\boldsymbol{Q}_n = E\left(\boldsymbol{w}_n\boldsymbol{w}_n^T\right)$	Process Noise Covariance	Process Noise Error
	$\boldsymbol{P}_{n,n} = E\left(\boldsymbol{e}_n\boldsymbol{e}_n^T\right) = E\left(\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}\right)\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}\right)^T\right)$	Estimation Covariance	Estimation Error

Figure 9.5: Enter Caption

9.3.2 MPU6050:

The MPU6050 is a 6-degree-of-freedom (6-DoF) IMU that combines a 3-axis accelerometer and a 3-axis gyroscope on a single chip. Components:

Accelerometer:

Measures linear acceleration along three axes (X, Y, Z) in units of g (gravitational acceleration, 9.81 m/s²). It is used to calculate Pitch and Roll angles for tilt compensation.

Gyroscope:

Measures angular velocity (rate of rotation) along three axes in degrees per second (°/s). The Z-axis angular velocity (Gyro) is used to calculate Yaw by integration.

Limitations:

Gyroscope drift: Integration of angular velocity introduces cumulative errors over time. Accelerometer noise: Susceptible to vibrations and non-gravitational accelerations. Requires calibration to correct for offsets and noise.

9.3.3 QMC5883L: Magnetometer

The QMC5883L is a 3-axis magnetometer that measures the Earth's magnetic field to determine orientation relative to the magnetic north pole. Measures magnetic field strength along three axes (X, Y, Z) in Gauss or microTesla.

Limitations:

Sensitive to magnetic interference (e.g., from metals, motors, or electronics). Requires Hard Iron calibration to correct for constant magnetic offsets (e.g., from nearby ferromagnetic materials). Sensitive to tilt (Pitch and Roll), which distorts Yaw calculations unless compensated. To achieve accurate Yaw estimation, the system employs a Complementary Filter to combine the strengths of the MPU6050 gyroscope (short-term accuracy) and the QMC5883L magnetometer (long-term stability), with tilt compensation to account for sensor inclination.

Concept:

The Complementary Filter combines two noisy measurements with complementary frequency characteristics.

Gyroscope:

Provides accurate short-term angular velocity data but suffers from drift over time due to integration errors.

Magnetometer:

Provides stable long-term orientation but is noisy and sensitive to external magnetic interference.

Arduino Implementation:

There is a complete code in the Appendix 6

Results and Analysis:

Yaw Range: The system successfully produces Yaw angles in the range of 0° to 360°, with tilt compensation ensuring accuracy during sensor inclination (e.g., right/left tilts). Testing showed Yaw values covering the full range when rotating the sensors 360° around the Z-axis.

Challenges:

Magnetic interference can still affect QMC5883L if calibration is not performed in a clean environment. Misalignment between MPU6050 and QMC5883L axes can introduce errors, mitigated by careful sensor mounting. The navigation component, leveraging sensor fusion of the MPU6050 and QMC5883L, provides accurate and stable Yaw estimation for the Variable Speed Gyroscope project. The Complementary Filter with tilt compensation

effectively combines gyroscope and magnetometer data, addressing issues like tilt-induced errors and gyroscope drift. The system's real-time performance and full 0°–360° Yaw range make it suitable for precise navigation and control applications. Future improvements could include advanced filtering and enhanced calibration to further improve robustness.

Chapter 10: Simulation Results

10.1 Case 1: Attitude Stabilization to Zero Orientation

This simulation evaluates the ability of the proposed control law to stabilize the spacecraft from an arbitrary initial orientation to a rest state at zero attitude. The scenario is inspired by classical benchmark studies [10, 31] and validates the controller's convergence behavior using realistic satellite dynamics modeled in Simscape Multibody.

The system starts from a non-zero MRP orientation and an initial angular velocity of up to 0.02 rad/s (Table 10.1). Four VSCMG units are initialized with symmetric gimbal angles and uniform wheel spin rates. The control torque is generated using the nonlinear control law from Eq. (3.2), and the actuator-level response is computed via the steering law in Eq. (3.11).

Figures 10.1–10.4 illustrate the convergence of the MRP attitude, angular velocity, wheel speeds, and gimbal angles.

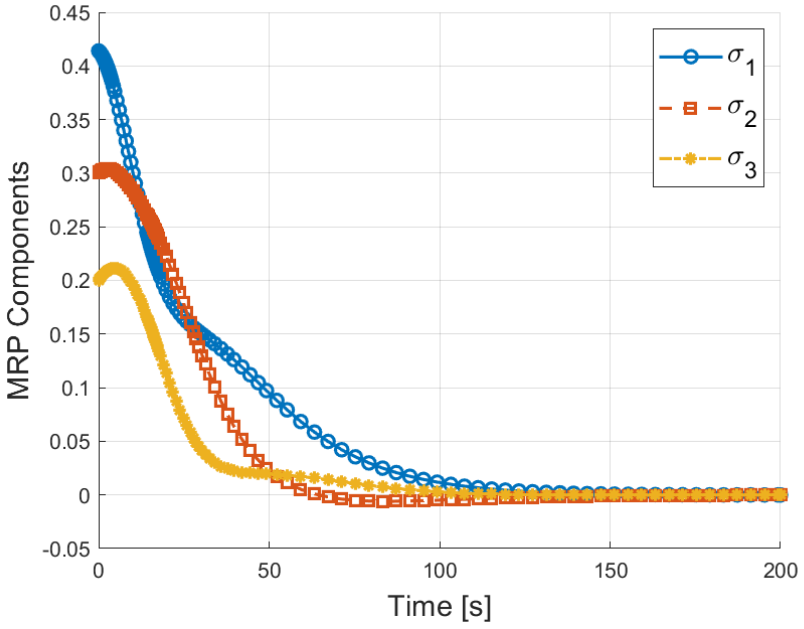


Figure 10.1: Attitude evolution (MRP vector) for Case 1 stabilization

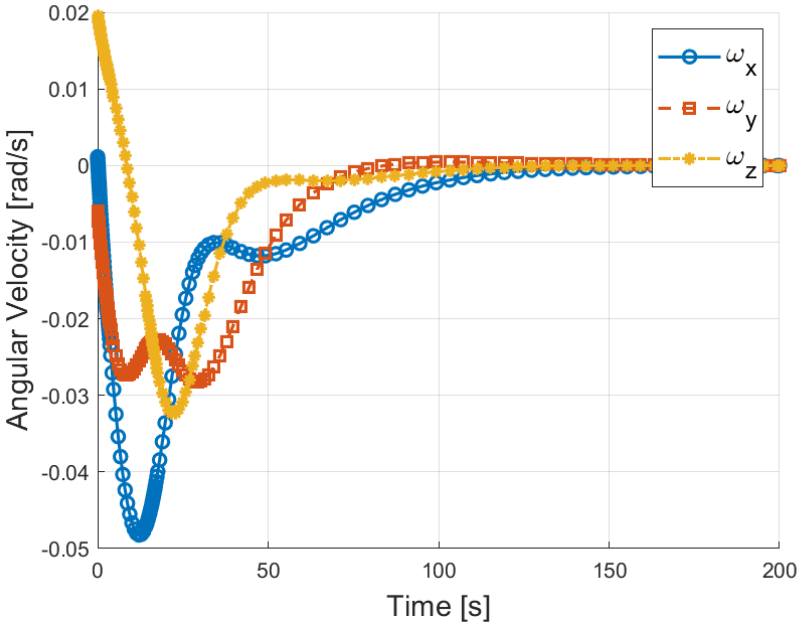


Figure 10.2: Angular velocity convergence of the spacecraft body

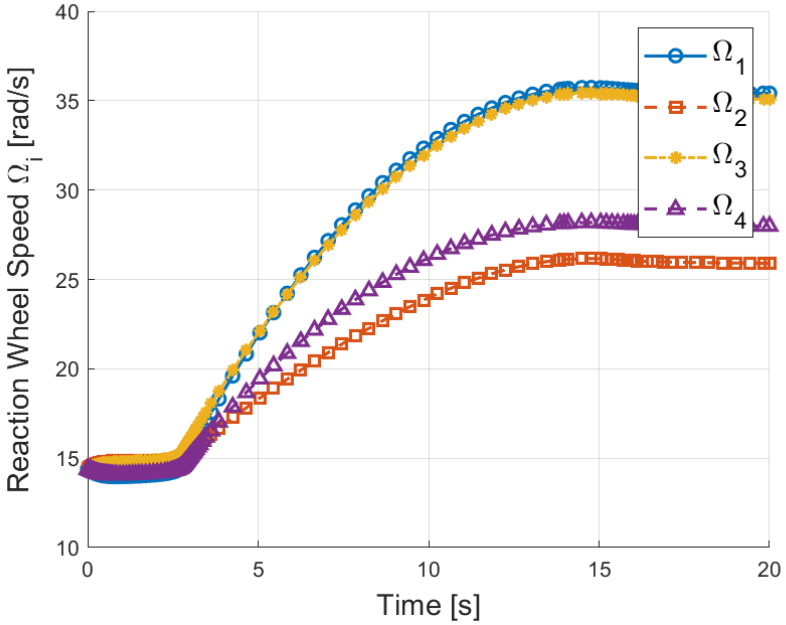


Figure 10.3: Reaction wheel speeds for all four VSCMG units

10.2 Case 2: Reference Attitude Tracking

The spacecraft is commanded to follow a time-varying reference attitude. The desired MRP vector is:

$$\boldsymbol{\sigma}_{R/N}(t) = \frac{1}{4}\begin{bmatrix} 0.1\sin(ft) \\ 0.2\cos(ft) \\ -0.3\sin(2ft) \end{bmatrix} \tag{10.1}$$

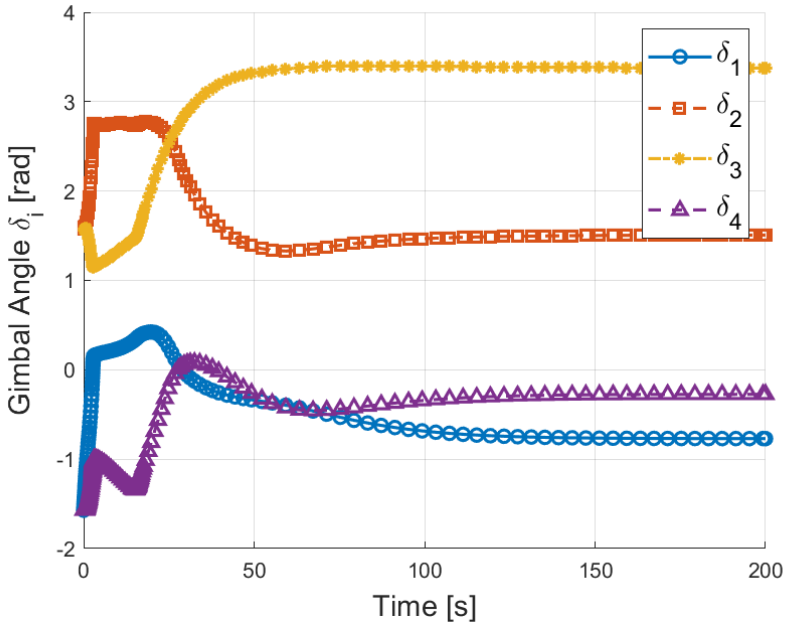


Figure 10.4: Gimbal angles under the steering law

with $f = 0.03$ rad/s.

The corresponding reference angular velocity $\boldsymbol{\omega}_r$ is calculated by:

$$\boldsymbol{\omega}_r = 4[B(\boldsymbol{\sigma}_{R/N})]\dot{\boldsymbol{\sigma}}_{R/N} \tag{10.2}$$

where

$$B = \frac{1}{(1+s^2)^2}\begin{bmatrix} 1-s^2+2\sigma_1^2 & 2(\sigma_1\sigma_2+\sigma_3) & 2(\sigma_1\sigma_3-\sigma_2) \\ 2(\sigma_2\sigma_1-\sigma_3) & 1-s^2+2\sigma_2^2 & 2(\sigma_2\sigma_3+\sigma_1) \\ 2(\sigma_3\sigma_1+\sigma_2) & 2(\sigma_3\sigma_2-\sigma_1) & 1-s^2+2\sigma_3^2 \end{bmatrix} \tag{10.3}$$

and $s^2 = \boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{\sigma}$.

10.3 Case 3: Null Motion Reconfiguration

This case demonstrates null motion steering, where the actuators move internally without changing the spacecraft's external state [10].

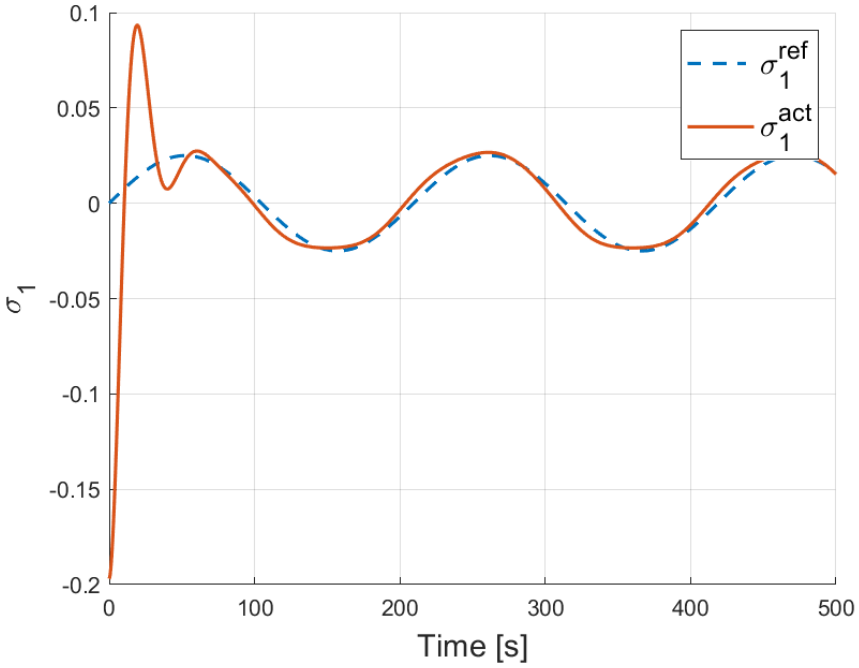


Figure 10.5: Comparison of reference and actual MRP components

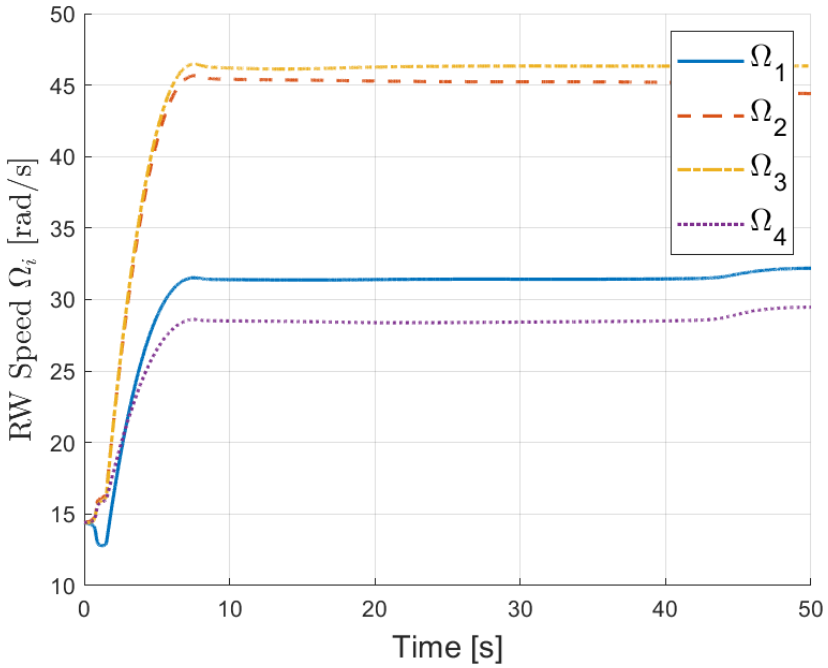


Figure 10.6: Reaction wheel speeds during reference tracking

Table 10.1: Initial Conditions and Parameters Used in Case 1 Simulation

Parameter	**Value**	**Unit**
$\boldsymbol{\sigma}_0$	[0.414, 0.3, 0.2]	–
$\boldsymbol{\omega}_0$	[0.01, 0.02, -0.01]	rad/s
Ω_0	14.4	rad/s
δ_0	$[-\pi/2, \pi/2, \pi/2, -\pi/2]$	rad
$[P]$	$\mathrm{diag}([13.13, 13.04, 15.08])$	kg·m^2
K	1.7	kg·m^2/s^2
ω_{s_i}	2.0	rad/s
ω_{δ_i}	1.0	rad/s
μ	10^{-9}	–

Table 10.2: Initial Conditions for Reference Attitude Tracking

Parameter	**Value**	**Units**
$\boldsymbol{\sigma}_0$	[-0.1969, 0.3549, 0.1128]	–
$\boldsymbol{\omega}_0$	[0.01, -0.01, 0.005]	rad/s
$\boldsymbol{\Omega}_0$	14.4	rad/s
$\boldsymbol{\delta}_0$	$[-\pi/2, \pi/2, 0, \pi]$	rad
P	15	kg·m^2
K	10	kg·m^2/s^2

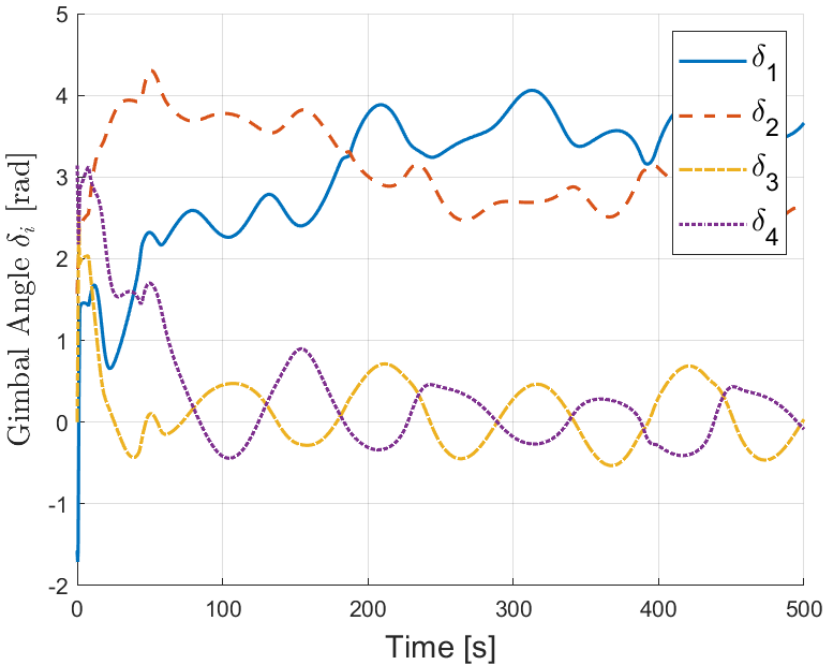


Figure 10.7: Gimbal angles during reference tracking

Table 10.3: Initial Conditions for Null Motion Simulation

Parameter	**Value**	**Unit**
$\boldsymbol{\sigma}_0$	[0.1, 0.2, 0.3]	–
$\boldsymbol{\omega}_0$	[0, 0, 0]	rad/s
$\boldsymbol{\Omega}_0$	14.4	rad/s
$\boldsymbol{\delta}_0$	[20, 0, 0, -20]	deg
$\boldsymbol{\delta}_f$	[-45, -45, 45, 45]	deg
k_e	10	–

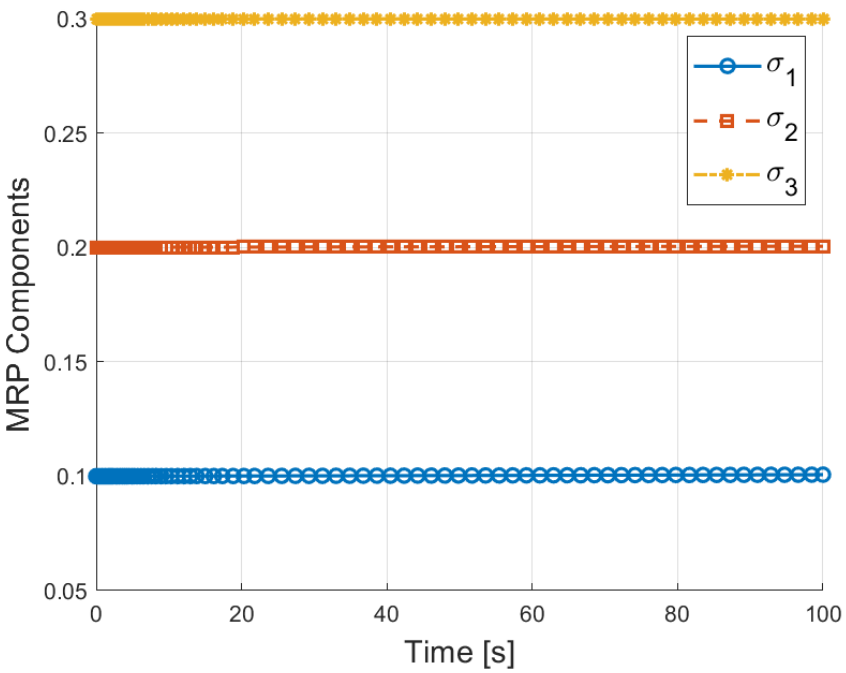


Figure 10.8: MRPs $\boldsymbol{\sigma}(t)$ remain constant under null motion

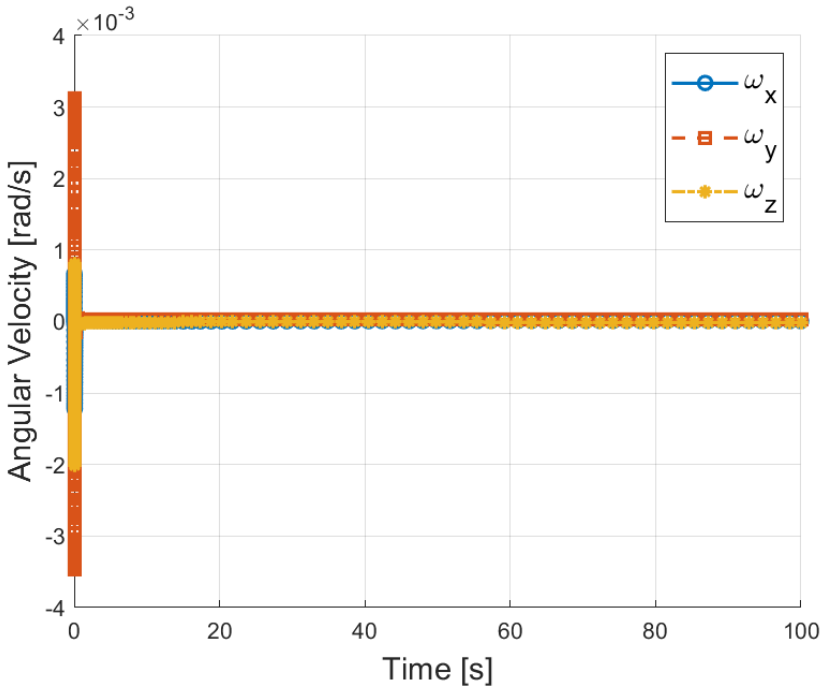


Figure 10.9: Angular velocity $\boldsymbol{\omega}(t)$ remains near zero

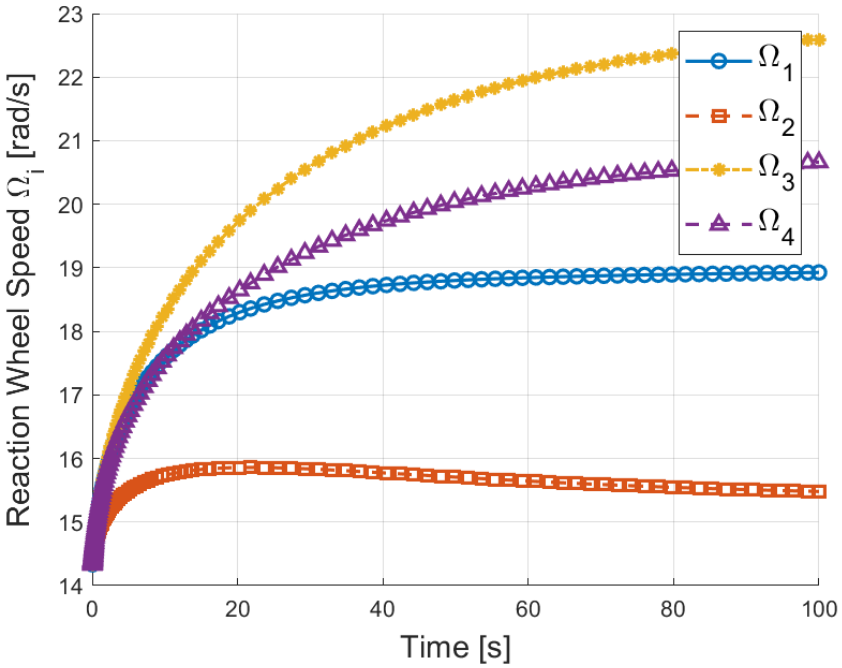


Figure 10.10: Wheel speeds $\boldsymbol{\Omega}(t)$ evolve during reconfiguration

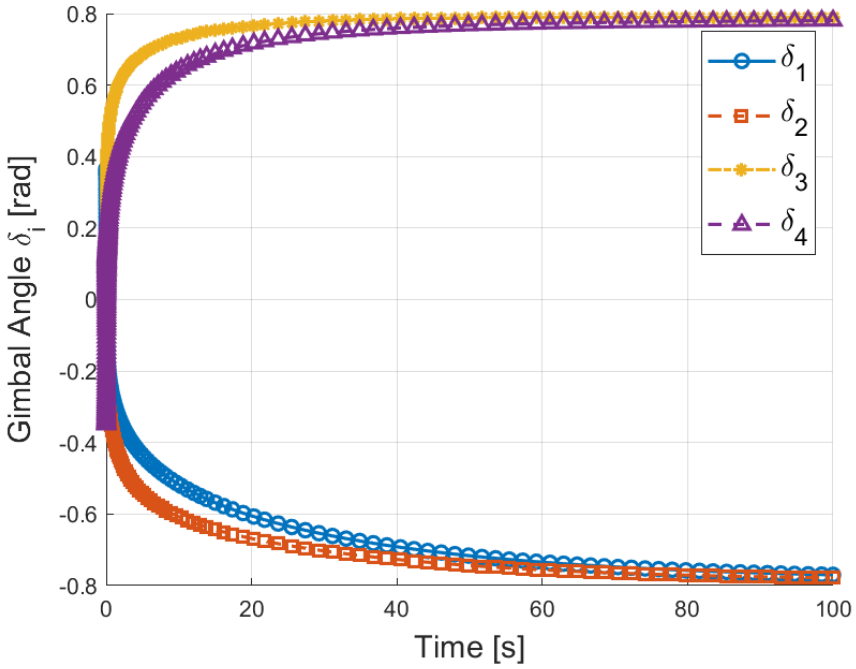


Figure 10.11: Gimbal angles $\boldsymbol{\delta}(t)$ tracking toward preferred values

Conclusion and Future Work

This thesis presented a comprehensive study on spacecraft attitude control using Variable Speed Control Moment Gyroscopes (VSCMGs), with an emphasis on avoiding internal singularities that arise during momentum steering. Singular configurations—often a result of unfavorable gimbal alignments or momentum vector degeneracy—pose significant challenges to attitude control systems by degrading torque authority and risking actuator saturation.

To tackle this, a high-fidelity nonlinear spacecraft model was developed using Simscape Multibody. This model accurately captured the dynamic interactions between the satellite body and the internal actuation components, including wheel spin dynamics and gimbal rotations. On top of this, a Lyapunov-based nonlinear controller was implemented for both stabilization and reference attitude tracking. Simulations verified that the controller robustly drives the spacecraft to the desired orientation while keeping actuator behavior within practical limits.

Three case studies were used to validate the system performance:

- **Case 1** demonstrated successful attitude stabilization from arbitrary initial conditions to zero orientation, confirming the effectiveness of the nonlinear control law in converging the state variables.
- **Case 2** evaluated reference attitude tracking under a time-varying MRP profile, showing smooth and accurate tracking behavior and actuator responses within nominal bounds.
- **Case 3** focused on null motion reconfiguration, highlighting the ability of the system to redistribute internal momentum without affecting the spacecraft's external state.

Classical steering strategies—namely, the pseudo-inverse and null-space projection methods—were employed as baselines. These provided stable and analytically tractable solutions and served as the foundation for pretraining a reinforcement learning (RL)-based steering policy. The RL policy, initialized with supervised learning over benchmark trajectories and later refined through direct interaction, demonstrated robust performance under dynamic and potentially singular configurations.

A hardware-in-the-loop prototype featuring a dual-gimbal VSCMG module was developed to experimentally validate the control framework. Results from this setup confirmed that the RL-based approach could match classical performance under nominal conditions and further improve behavior near singularities, reducing control effort and enhancing robustness.

While the simulation and experimental results are promising, several avenues remain for future exploration:

1. **Hardware Scalability and Validation:** Extend the current hardware setup to a full 3-axis spacecraft testbed with multiple VSCMGs arranged in a pyramidal or tetrahedral configuration. This would allow real-time testing of full-attitude maneuvers and singularity handling in a more complex and realistic environment.
2. **Online Learning and Adaptation:** Integrate online reinforcement learning or meta-learning techniques that allow the control policy to adapt in real time to changes in spacecraft parameters, actuator faults, or unexpected external disturbances (e.g., flexible appendages or environmental torques).
3. **Uncertainty Quantification:** Incorporate robust control or Bayesian RL methods to account for uncertainties in model parameters, actuator delays, and sensor noise. This could improve reliability in practical space missions.
4. **Trajectory Optimization under Constraints:** Combine the RL-based steering with optimal control formulations that consider mission-level constraints such as energy limits, pointing accuracy windows, or eclipse conditions.
5. **Singularity-Aware Path Planning:** Further investigation into geometric methods or learning-based path planners that explicitly avoid singular configurations in attitude space while minimizing maneuver time and actuator usage.
6. **Integration with ADCS System:** Integrate the VSCMG control system into a full Attitude Determination and Control System (ADCS), including star trackers, sun sensors, and magnetometers, for autonomous operation in LEO or GEO missions.

In summary, this work establishes a strong foundation for intelligent attitude control of spacecraft using VSCMGs. The combination of nonlinear control, classical steering laws, and adaptive learning presents a viable pathway for tackling internal singularities and pushing the frontier of autonomous space mission capabilities.

References

[1] Wertz, J. R., *Spacecraft Attitude Determination and Control*, Springer, 1978.

[2] Hughes, P. C., *Spacecraft Attitude Dynamics*, Dover Publications, 2004.

[3] Schaub, H., Junkins, J. L., *Analytical Mechanics of Space Systems*, AIAA Education Series, 2014.

[4] Vallado, D. A., *Fundamentals of Astrodynamics and Applications*, Microcosm Press, 2013.

[5] Leeghim, H., Bang, H., "Singularity-Free Steering Logic for Variable-Speed Control Moment Gyroscopes," *Acta Astronautica*, 62, 2008.

[6] Schaub, H., Junkins, J. L., "Singularity Avoidance Using Null Motion and Variable-Speed Control Moment Gyroscopes," *Journal of Guidance, Control, and Dynamics*, 23(6), 2000, pp. 1165–1171.

[7] Kim, Y., Hall, C. D., "Lyapunov-Based Attitude Control Using VSCMGs: Singularity Analysis and Avoidance," *Journal of Guidance, Control, and Dynamics*, 30(5), 2007.

[8] MathWorks, "Modeling Physical Systems with Simscape," *MathWorks Documentation*, 2023. Available: https://www.mathworks.com/help/physmod/simscape/

[9] Nguyen, T. V., et al., "Simscape-Based Spacecraft Dynamics Simulator for Hardware-in-the-Loop Testing," *Aerospace*, 9(8), 2022.

[10] H. Schaub, S. R. Vadali, and J. L. Junkins, "Feedback control law for variable speed control moment gyros," *J. Astronaut. Sci.*, vol. 46, no. 3, pp. 307–328, 1998.

[11] S. Miller, T. Soares, Y. Van Weddingen, and J. Wendlandt, "Modeling flexible bodies with Simscape Multibody software: An overview of two methods," *MathWorks Tech. Paper*, 2017.

[12] A. Bortolotti, "Analysis and modeling of satellite flexible bodies in Simscape Multibody," M.S. thesis, Univ. of Padua, 2022.

[13] L. Collette and M. Magnin–Mattenet, “Simulation of robotic space operations with minimum base reaction using Simscape Multibody,” *Simul. Model. Pract. Theory*, vol. 112, pp. 6–51, 2022.

[14] M. C. De Simone, G. Ventura, A. Lorusso, and D. Guida, “Attitude controller design for micro-satellites,” in *Proc. 12th ESA Workshop Avionics, Data, Control Software. Syst. (ADCSS)*, 2021.

[15] Ü. Önen and A. Çakan, “Multibody modeling and balance control of a reaction wheel inverted pendulum using LQR controller,” *Int. J. Robot. Control Syst.*, vol. 1, no. 1, pp. 84–89, Mar. 2021, doi:10.31763/ijrcs.v1i1.296.

[16] D. Brown and M. A. Peck, “Scissored-Pair Control Moment Gyros: A Mechanical Constraint Saves Power,” *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 6, pp. 1823–1826, 2008.

[17] H. Kurokawa, “Survey of Theory and Steering Laws of Single-Gimbal Control Moment Gyros,” *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1331–1340, 2007.

[18] M. R. Elgersma, D. P. Johnson, and M. A. Peck, “Method and System for Controlling Sets of Collinear Control Moment Gyroscopes,” U.S. Patent Application 20070124032A1, 2007.

[19] D. Liska, “A Two-Degree-of-Freedom Control Momentary for High-Accuracy Attitude Control,” *Journal of Spacecraft and Rockets*, vol. 5, no. 1, pp. 74–83, 1968.

[20] L. Brennan, “Means for Imparting Stability to Unstable Bodies,” U.S. Patent 796893, 1905.

[21] T. B. Murtagh, C. E. Whitsett, and M. A. Goodwin, “Automatic Control of the Skylab Astronaut Maneuvering Research Vehicle,” *Journal of Spacecraft and Rockets*, vol. 11, no. 5, pp. 321–326, 1974.

[22] L. F. Yang and W. H. Chang, “Synchronization of Twin-Gyro Precession Under Cross-Coupled Adaptive Feedforward Control,” *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 3, pp. 534–539, 1996.

[23] B. Wie, D. Bailey, and C. Heiberg, “Rapid Multitarget Acquisition and Pointing Control of Agile Spacecraft,” *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 96–104, 2002.

[24] V. J. Lappas, W. H. Steyn, and C. I. Underwood, “Attitude Control for Small Satellites Using Control Moment Gyros,” *Acta Astronautica*, vol. 51, no. 1–9, pp. 101–111, 2002.

[25] *FT5835M Servo Datasheet*, Feetech Intelligent Technology Co., 2023. [PDF].

[26] *Aluminum 6061-T6 Technical Data Sheet*, MatWeb Material Property Database, 2022. [Online]. Available: www.matweb.com.

[27] L. F. Yang, M. M. Mikulas, Jr., and K. C. Park, "Slewing Maneuvers and Vibration Control of Space Structures by Feedforward/Feedback Moment-Gyro Controls," *Journal of Dynamic Systems, Measurement, and Control*, vol. 117, pp. 343–351, 1995.

[28] K. A. Ford, "Gearbox Dynamics in CMGs," *Journal of Mechanical Design*, vol. 144, no. 3, p. 031101, 2022.

[29] *ISO 286-1: Geometrical Product Specifications (GPS) – ISO Code System for Tolerances on Linear Sizes – Part 1: Bases of Tolerances, Deviations, and Fits*, International Organization for Standardization, 2010.

[30] *NASA GEVS: General Environmental Verification Standard (GSFC-STD-7000)*, NASA Goddard Space Flight, 2021.

[31] H. S. Oh and S. R. Vadali, "Feedback control and steering laws for spacecraft using single gimbal control moment gyros," *J. Astronaut. Sci.*, vol. 39, no. 2, pp. 183–203, 1991.

[32] T. Sasaki and T. Shimomura, "Gain-scheduled control/steering design for a spacecraft with variable-speed control moment gyros," *SICE J. Control Meas. Syst. Integr.*, vol. 10, no. 3, pp. 237–242, 2017.

[33] H. Yoon and P. Tsiotras, "Spacecraft adaptive attitude and power tracking with variable speed control moment gyroscopes," *J. Guid. Control Dyn.*, vol. 25, no. 6, pp. 1081–1090, 2002.

[34] S. R. Marandi and V. J. Modi, "A preferred coordinate system and the associated orientation representation in attitude dynamics," *Acta Astronaut.*, vol. 15, no. 11, pp. 833–843, 1987.

[35] A. Alkamachi, "Integrated SolidWorks and Simscape platform for the design and control of an inverted pendulum system," *J. Electr. Eng.*, vol. 71, no. 2, pp. 122–126, 2020.

[36] MathWorks, "Multibody dynamic simulation with Simscape: Methods and examples," *MathWorks Tech. Doc.*, 2021. [Online]. Available: https://www.mathworks.com/help/physmod/sm/ug/multibody-dynamic-simulation-with-simscape.html

[37] MathWorks, *Simscape Multibody User's Guide*, 2021. [Online]. Available: https://www.mathworks.com/help/physmod/sm/

[38] S. N. Deore, *Neural network based steering and hardware-in-the-loop simulation of variable speed control moment gyroscope*, M.S. thesis, Dept. Mech. Aerosp. Eng., Sapienza Univ. of Rome, Rome, Italy, 2020.

[39] H. Schaub and J. L. Junkins, *Analytical mechanics of space systems*, 3rd ed. Reston, VA: AIAA, 2014.

[40] D. N. Cardoso, S. Esteban, and G. V. Raffo, "A robust optimal control approach in the weighted Sobolev space for underactuated mechanical systems," *Automatica*, vol. 125, p. 109474, 2021.

[41] Wang, P. and Shtessel, Y. B., "Satellite attitude control using only magnetorquers," *Proc. 13th Southeastern Symposium on System Theory*, 1998.

[42] Yeh, F.-K., "Sliding-mode adaptive attitude controller design for spacecrafts with thrusters," *IET Control Theory & Applications*, Vol. 4, No. 7, 2010, pp. 1254–1264.

[43] Petersen, C. D., Leve, F., and Kolmanovsky, I., "Model predictive control of an underactuated spacecraft with two reaction wheels," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 320–332.

[44] Bedrossian, N. S., et al., "Steering law design for redundant SGCMGs," *Journal of Guidance, Control, and Dynamics*, Vol. 13, No. 6, 1990, pp. 1083–1089.

[45] Wie, B., et al., "Singularity robust steering logic for SGCMGs," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 5, 2001, pp. 865–872.

[46] Leeghim, H., et al. (2015). Adaptive neural control of spacecraft using CMGs. Adv. Space Res. 55(5), 1382–1393.

[47] Leeghim, H., et al., "Feasible angular momentum of spacecraft installed with control moment gyros," *Advances in Space Research*, Vol. 61, No. 1, 2018, pp. 466–477.

[48] Cornick, D., "Singularity avoidance control laws for single gimbal CMGs," *Guidance and Control Conference*, AIAA, 1979.

[49] Papakonstantinou, C., et al., "Machine learning approach for global steering control of CMG clusters," *Aerospace*, Vol. 9, No. 3, 2022, Article 164.

[50] Seo, H.-H., et al., "Steering law of CMGs using artificial potential function," *Acta Astronautica*, Vol. 157, 2019, pp. 374–389.

[51] Davis, B., "A Comparison of CMG Steering Laws for High Energy Astronomy Observatories (HEAOS)," NASA Technical Report, 1972.

[52] Sutherland, R., Kolmanovsky, I., and Girard, A. R., “Attitude control of a 2U CubeSat by magnetic and air drag torques,” *IEEE Transactions on Control Systems Technology*, Vol. 27, No. 3, 2018, pp. 1047–1059.

[53] MathWorks Inc., “Simscape Multibody User’s Guide,” Version R2023b, Natick, MA, 2023.

[54] A. Hughes and B. Drury, *Electric Motors and Drives: Fundamentals, Types and Applications*, 5th ed. Oxford, U.K.: Newnes, 2019.

[55] FTS8355M Servo Motor Pin Configuration. Available at: https://www.electronicwings.com/nodemcu/servo-motor-interfacing-with-nodemcu [Accessed: 6-Jul-2025].

[56] Software Particles, *Learn How a Servo Motor Works and How to Control It Using Arduino*, Available at: https://softwareparticles.com/learn-how-a-servo-motor-works-and-how-to-control-it-using-arduino/ [Accessed: 6-Jul-2025].

[57] Jameco Electronics, *How Servo Motors Work*, Available at: https://www.jameco.com/Jameco/workshop/Howitworks/how-servo-motors-work.html [Accessed: 6-Jul-2025].

[58] R. Santos, *ESP32 MPU-6050 Accelerometer and Gyroscope (Arduino)*, Random Nerd Tutorials, [Online]. Available: https://randomnerdtutorials.com/esp32-mpu-6050-accelerometer-gyroscope-arduino/ Accessed: 7 July 2025.

[59] Honeywell, {*3-Axis Digital Compass IC QMC5883L*}, *Advanced Information Datasheet, Available:* *url*{*http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Magneto/HMC5883L-FDS.pdf*} *end*{*thebibliography*}

[60] Future Electronics Egypt, 3 Axis Digital Compass (HMC5883L) GY-271*, [Online]. Available: https://store.fut-electronics.com/products/3-axis-digital-compass-honeywell-hmc5883l Accessed: 7 July 2025.*

[61] Sensors Modules, QMC5883L Magnetometer Module*, [Online]. Available: https://wordpress.org/support/topic/how-to-check-if-current-page-is-a-product-page/ [Accessed: Day Month Year, e.g., 7 July 2025]*

[62] P. Kim, Kalman Filter for Beginners: with MATLAB Examples*, L. Huh, Translator. Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2011.*

[63] A. Becker, Kalman Filter from the Ground Up*, 3rd ed., 2024. [Online]. Available: https://www.kalmanfilter.net. [Accessed: Jul. 7, 2025]. ISBN: 9789655984392.*

[64] M. S. Grewal and A. P. Andrews, Kalman Filtering: Theory and Practice Using MATLAB*, 3rd ed. Hoboken, NJ, USA: Wiley, 2008.*

[65] W. Elmenreich, "An Introduction to Sensor Fusion," Institut fur Technische Informatik, Vienna Univ. Technol., Vienna, Austria, Res. Rep. 47/2001, Nov. 2002. [Online]. Available: http://www.vmars.tuwien.ac.at/ wil/papers/Elmenreich-SensorFusion-2002.pdf. [Accessed: Jul. 7, 2025].

[66] P. Zarchan and H. Musoff, Fundamentals of Kalman Filtering: A Practical Approach*, 3rd ed. Reston, VA, USA: American Institute of Aeronautics and Astronautics, Inc., 2009. [Online]. Available: http://arc.aiaa.org. [Accessed: Aug. 11, 2015]. 10.2514/4.867200.*

Code Appendix

The following is the source code for the Arduino sketch used to Calculatre DC RPM.

```
#include <EnableInterrupt.h>

#define encoderA 4
#define encoderB 5
#define encoderZ 3

volatile long countA = 0;
int pulsesPerRev = 2500;

unsigned long lastTime = 0;
long lastCount = 0;

void setup() {
  Serial.begin(9600);
  pinMode(encoderA, INPUT);
  pinMode(encoderB, INPUT);
  pinMode(encoderZ, INPUT);
  enableInterrupt(encoderA, pulseA, RISING);
  enableInterrupt(encoderZ, pulseZ, RISING);
}

void loop() {
  unsigned long currentTime = millis();

  if (currentTime - lastTime >= 1000) {
    long deltaCount = countA - lastCount;
    float rpm = (deltaCount * 60.0) / pulsesPerRev;

    /*Serial.print("Encoder Pulses in 1 sec: ");
    Serial.print(deltaCount);
    Serial.print("\tRPM: ");*/
```

```
    Serial.print("Encoder Pulses in 1 sec: ");
    Serial.print(deltaCount);
    Serial.print("\tRPM: ");
    Serial.println(rpm);
    Serial.println(rpm);


    lastCount = countA;
    lastTime = currentTime;
  }
}

void pulseA() {
  countA++;
}

void pulseZ() {
  countA = 0;
}
```

Listing 1: RPM Calculation arduino code

The following is a code describes Serial Connection Between Matlab& Arduino to plot graph relation between RPM time

```
s = serialport("COM4", 9600);
rpm_data = [];
time_data = [];
figure;

tic

while true
    if s.NumBytesAvailable > 0
        line = readline(s);
        rpm = str2double(line);

        if ~isnan(rpm)
            rpm_data(end+1) = rpm;
            time_data(end+1) = toc;

            plot(time_data, rpm_data, 'b-', 'LineWidth', 2);
            xlabel('Time (s)');
```

```
        ylabel('RPM');
        grid on;
        drawnow;
    end
  end
  if toc > 60
    break;
  end

  pause(0.05);
end
data = [time_data' rpm_data'];
writematrix(data, 'rpm_data.xlsx');
```

Listing 2: Serial Connection Between Matlab& Arduino

The following is the source code for the Arduino sketch used to Calculatre PWM.

```


#include <EnableInterrupt.h>

#define encoderA 4
#define encoderB 5
#define encoderZ 3

int PWM = 6 ; // PWM of Motor driver connect to D6
int DIR = 11; // DIR of Motor driver connect to D5


int tempval = 0; //variable to keep adc value after mapping

volatile long countA = 0;
int pulsesPerRev = 2500;

unsigned long lastTime = 0;
long lastCount = 0;

void setup() {
  Serial.begin(115200);

  pinMode(encoderA, INPUT);
```

```
  pinMode(encoderB, INPUT);
  pinMode(encoderZ, INPUT);

  pinMode(PWM, OUTPUT); //PWM output
  pinMode(DIR, OUTPUT); //DIR output

  enableInterrupt(encoderA, pulseA, RISING);
  enableInterrupt(encoderZ, pulseZ, RISING);

  digitalWrite(PWM,LOW); //ensure the PWM is LOW at initial stage
  digitalWrite(DIR, LOW); //ensure the DI-R is LOW at initial stage
}



void loop() {
  unsigned long currentTime = millis();

  if (currentTime - lastTime >= 1000) {
    long deltaCount = countA - lastCount;
    float rpm = (deltaCount * 60.0) / pulsesPerRev;

    /*Serial.print("Encoder Pulses in 1 sec: ");
    Serial.print(deltaCount);
    Serial.print("\tRPM: ");*/
    Serial.print("Encoder Pulses in 1 sec: ");
    Serial.print(deltaCount);
    Serial.print("\tRPM: ");
    Serial.println(rpm);
    Serial.println(rpm);


    lastCount = countA;
    lastTime = currentTime;
  }

  digitalWrite(PWM, HIGH); //Enable the motor driver

  analogWrite(DIR, 54); // Output the PWM signal at DIR pin
  // digitalWrite(DIR, HIGH); //ensure the DIR is LOW at initial
     stage
```

```
  // delay(15);
}

void pulseA() {
  countA++;
}

void pulseZ() {
  countA = 0;
}
```

Listing 3: RPM Calculation arduino code

The following is the source code for the Arduino sketch used to run the FT5835M Servo motor.

```
#include <Servo.h>

Servo myServo; // Create a Servo object
int servoPin = 9; // Pin connected to the servo
int speed = 0; // Speed variable (-60 to 60)

void setup() {
  myServo.attach(servoPin);
  Serial.begin(9600);
  Serial.println("Enter speed (-60 to 60):");

void loop() {

  if (Serial.available() > 0) {
    speed = Serial.parseInt();

    if (speed >= -60 && speed <= 60) {
      Serial.print("Setting speed: ");
      Serial.println(speed);
      // Map the speed (-60 to 60) to servo range (0 to 180)
      int servoSpeed = map(speed, -60, 60, 0, 180);
      myServo.write(servoSpeed); // Set servo speed
    } else {
      Serial.println("Invalid speed! Enter a value between -60 and
          60.");
    }

```

```
    while (Serial.available() > 0) {
      Serial.read();
    }
    Serial.println("Enter speed (-60 to 60):");
  }
}
```

Listing 4: Servo arduino code

The following is the source code for the Arduino sketch used for magnetometer calibration.

```
#include <Wire.h>
#define QMC_ADDR 0x0D

int16_t x_offset = 0, y_offset = 0, z_offset = 0;
float x_scale = 1.0, y_scale = 1.0, z_scale = 1.0;

void setup() {
  Serial.begin(115200);
  Wire.begin();
  Wire.beginTransmission(QMC_ADDR);
  Wire.write(0x09); // Control register
  Wire.write(0x1D); // OSR=512, RNG=8G, ODR=200Hz, continuous mode
  Wire.endTransmission();
  delay(100);
  Serial.println("Magnetometer ready.");
}

void readRawData(int16_t &x, int16_t &y, int16_t &z) {
  Wire.beginTransmission(QMC_ADDR);
  Wire.write(0x00);
  Wire.endTransmission();
  Wire.requestFrom(QMC_ADDR, 6);
  if (Wire.available() == 6) {
    uint8_t xl = Wire.read();
    uint8_t xh = Wire.read();
    uint8_t yl = Wire.read();
    uint8_t yh = Wire.read();
    uint8_t zl = Wire.read();
    uint8_t zh = Wire.read();
    x = (int16_t)((xh << 8) | xl);
    y = (int16_t)((yh << 8) | yl);
    z = (int16_t)((zh << 8) | zl);
```

```
  }
}

void loop() {
  int16_t x_raw, y_raw, z_raw;
  readRawData(x_raw, y_raw, z_raw);

  float x_cal = (x_raw - x_offset) * x_scale;
  float y_cal = (y_raw - y_offset) * y_scale;
  float z_cal = (z_raw - z_offset) * z_scale;

  float heading = atan2(y_cal, x_cal) * 180.0 / PI;
  if (heading < 0) heading += 360.0;

  Serial.print("Heading: ");
  Serial.println(heading);
  delay(100);
}
```

Listing 5: RPM Calculation arduino code

This Following is the source code for the Arduino sketch used to calibrate magnetometer, read, and fuse magnetometer and gyrosopce.

```
#include <Wire.h>
#include <QMC5883LCompass.h>
#include <MPU6050_light.h>

QMC5883LCompass compass;
MPU6050 mpu(Wire);
float mag_x_offset = 0, mag_y_offset = 0; // Magnetometer offsets
float gyro_z_offset = 0; // Gyroscope offset
float yaw = 0; // Combined Yaw angle in degrees
const float deadband = 0.1; // Deadband threshold (/s)
const float alpha = 0.98; // Complementary filter coefficient (0.98
    for gyro, 0.02 for mag)

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

  // Initialize QMC5883L
  compass.init();
```

```
19
20  // Initialize MPU6050
21  byte status = mpu.begin();
22  Serial.print(F("MPU6050 status: "));
23  Serial.println(status);
24  if (status != 0) {
25    while (1); // Halt if MPU6050 connection fails
26  }
27
28  // Calibrate QMC5883L (Hard Iron)
29  Serial.println(F("Rotate sensor in XY plane for 15 seconds to
       calibrate QMC5883L..."));
30  delay(1000);
31  int mag_x_min = 32767, mag_x_max = -32768;
32  int mag_y_min = 32767, mag_y_max = -32768;
33  unsigned long cal_timer = millis();
34  while (millis() - cal_timer < 15000) { // 15 seconds
35    compass.read();
36    int x = compass.getX();
37    int y = compass.getY();
38    if (x < mag_x_min) mag_x_min = x;
39    if (x > mag_x_max) mag_x_max = x;
40    if (y < mag_y_min) mag_y_min = y;
41    if (y > mag_y_max) mag_y_max = y;
42    delay(10);
43  }
44  mag_x_offset = (mag_x_max + mag_x_min) / 2.0;
45  mag_y_offset = (mag_y_max + mag_y_min) / 2.0;
46
47  // Calibrate MPU6050 (Gyro Z offset)
48  Serial.println(F("Calculating gyro offsets, do not move MPU6050
       ..."));
49  delay(1000);
50  const int num_samples = 1000;
51  for (int i = 0; i < num_samples; i++) {
52    mpu.update();
53    gyro_z_offset += mpu.getGyroZ();
54    delay(10);
55  }
56  gyro_z_offset /= num_samples;
57
```

```
  Serial.println(F("Calibration Done!"));
}

void loop() {
  mpu.update();
  compass.read();

  unsigned long current_time = millis();
  static unsigned long timer = current_time;
  if ((current_time - timer) > 10) { // Update every 10ms
    float dt = (current_time - timer) / 1000.0; // Sample time in
       seconds

    // Calculate Yaw from gyroscope
    float gyro_z = mpu.getGyroZ() - gyro_z_offset;
    if (abs(gyro_z) < deadband) {
      gyro_z = 0; // Apply deadband
    }
    yaw += gyro_z * dt; // Integrate gyroscope data

    // Calculate Yaw from magnetometer
    float mag_x = compass.getX() - mag_x_offset;
    float mag_y = compass.getY() - mag_y_offset;
    float mag_yaw = atan2(-mag_y, mag_x) * 180.0 / PI;
    if (mag_yaw < 0) mag_yaw += 360;

    // Combine Yaw using Complementary Filter
    yaw = alpha * yaw + (1 - alpha) * mag_yaw;

    // Ensure Yaw stays between 0 and 360 degrees
    if (yaw < 0) yaw += 360;
    if (yaw >= 360) yaw -= 360;

    // Print combined Yaw
    Serial.println(yaw);

    timer = current_time; // Update timer
  }
}
```

Listing 6: navigation arduino code

The following MATLAB code generates the singular surfaces (both external and internal) for a 4-CMG pyramid configuration with a skew angle of $\beta = 54.73°$. It evaluates all valid combinations of ε_k signs and computes the net momentum direction $\mathbf{H}_u$ as in Equation (3.3).

```
clc; clear;

% Define 4-CMG pyramid gimbal directions
tilt = deg2rad(54.73);
gimbals = [
    sin(tilt)*cos(0), sin(tilt)*sin(0), cos(tilt);
    sin(tilt)*cos(pi/2), sin(tilt)*sin(pi/2), cos(tilt);
    sin(tilt)*cos(pi), sin(tilt)*sin(pi), cos(tilt);
    sin(tilt)*cos(3*pi/2), sin(tilt)*sin(3*pi/2), cos(tilt)
];

% Define epsilon sets
aps_sets = [
    1 1 1 1; % External
    1 -1 -1 -1;
   -1 1 -1 -1;
   -1 -1 1 -1;
   -1 -1 -1 1
];

% Color map
colors = [
    0.2 0.6 1.0; % Blue - external
    1.0 0.4 0.4; % Red
    0.4 1.0 0.4; % Green
    1.0 1.0 0.4; % Yellow
    0.6 0.4 1.0 % Purple
];

% Create unit direction sphere
[uTheta, uPhi] = meshgrid(linspace(0, 2*pi, 200), linspace(0, pi,
    200));
uX = sin(uPhi).*cos(uTheta);
uY = sin(uPhi).*sin(uTheta);
uZ = cos(uPhi);

```

```
figure('Position', [100, 100, 1500, 600])
tiledlayout(1, 5, 'TileSpacing', 'compact');

% Loop through each epsilon set
for e = 1:size(aps_sets,1)
    aps = aps_sets(e,:);
    HX = nan(size(uX));
    HY = nan(size(uX));
    HZ = nan(size(uX));

    % Compute h(u)
    for i = 1:numel(uX)
        u = [uX(i); uY(i); uZ(i)];
        h = zeros(3,1);
        is_singular = false;

        for k = 1:4
            Og = gimbals(k,:)';
            c = cross(Og, u);
            if norm(c) < 0.001 % singular when u  gimbal axis
                is_singular = true;
                break;
            end
            h = h + aps(k) * cross(c, Og) / norm(c);
        end

        if ~is_singular
            HX(i) = h(1);
            HY(i) = h(2);
            HZ(i) = h(3);
        else
            HX(i) = NaN; HY(i) = NaN; HZ(i) = NaN;
        end
    end

    % Plot surface with real holes
    nexttile;
    surf(HX, HY, HZ, ...
        'EdgeColor', 'none', ...
        'FaceColor', colors(e,:), ...
        'FaceLighting', 'gouraud', ...
```

```
77        'AmbientStrength', 0.4, ...
78        'DiffuseStrength', 0.9, ...
79        'SpecularStrength', 0.3);
80
81    camlight headlight;
82    material dull;
83    axis equal
84    view(35, 25)
85    title(['\epsilon = [', sprintf('%2d ', aps), ']'], 'FontSize',
          12);
86 end
```

Listing 7: Singular surface generation for CMG pyramid configuration