Single Axis Reaction Wheel Configuration for CubeSat Attitude Control System

written by Karthik Mukund, Allen Devaraj

under the guidance of Dr. Priyadarshnam Hari

July 16, 2024

Abstract

The Attitude Determination and Control Subsystem (ADCS) is crucial for the vital operations of satellites, tasked with determining the satellite's orientation relative to a reference body in space and achieving the required attitude within a limited time frame. Various sensors, including Sun sensors, Earth sensors, and star sensors, are utilized for attitude determination, while control algorithms and actuators such as magnetorquers and reaction wheels are employed to achieve the desired attitude. These actuators are electrically excited, with the torques generated controlled by adjusting the power supplied to them, guided by control algorithms based on the current attitude and angular velocities. To test and validate the responses of these control strategies, sensors, and actuators, testbeds such as air-bearing systems are used to emulate near-space conditions by eliminating gravitational friction.

FloatSat, or Floating Satellite, is a hardware system designed to emulate an original satellite, incorporating various subsystems. The current version includes a reaction wheel for 1-axis control as part of the ADCS, consisting of a motor and flywheel assembly. This system aims to obtain the rotational velocity and displacement of the FloatSat along a single (vertical) axis through a motor speed control loop. Additionally, the project encompasses testing and validating magnetorquers for Nanosats across three axes. Magnetorquers, which are solenoid coils, generate torque when an electric current creates a magnetic moment that interacts with the Earth's magnetic field, subsequently applying torque to the satellite body. This abstract summarizes the design, implementation, and validation of these control strategies within the FloatSat system.

I. Introduction

A. CubeSats are small, standardized satellites that have gained popularity for space research and technology demonstrations due to their low cost and rapid development cycle. An essential subsystem in a CubeSat is the Attitude Determination and Control Subsystem (ADCS), which is responsible for controlling the satellite's orientation in space. Reaction wheels are a common method used in ADCS for attitude control providing precise torque to change the satellite's orientation.

B. This project aims to develop a mono-wheel reaction wheel system for a CubeSat using an Arduino Uno microcontroller. The system integrates various components, including a Pololu motor with an encoder an LSM9DS0 IMU sensor, and a Pololu motor driver to achieve precise control over the CubeSat's orientation. The primary objective is to implement a control algorithm that adjusts the motor speed gradually to achieve and maintain a desired gyroscope Z-axis rate, ensuring smooth and accurate attitude control.

II. Literature Review

A. *Reaction Wheel*

Reaction wheels are crucial components in spacecraft for attitude control, allowing precise orientation adjustments without the need for external thrusters.

Reaction wheels are essentially flywheels that store angular momentum. They are mounted on electric motors and can spin at varying speeds. When the motor accelerates or decelerates the wheel, the spacecraft experiences a counter-rotation due to the conservation of angular momentum. This principle is based on Newton's third law: for every action, there is an equal and opposite reaction.

B. *Reference Frames*

Reference frames can be specified by the location of their origin and the orientation of their coordinate axes.

1) Spacecraft Body Frame
   a. Origin: Specified point in the spacecraft.
   b. Orientation: Orientation of a sufficiently rigid navigation base (the subsystem of the spacecraft including the most critical attitude sensors and payload instruments).
   c. Components in the spacecraft will generally shift due to large forces experienced during launch and thermal deformations while in orbit.
   d. Purpose of Attitude Determination and Control: To ascertain and control the orientation of the navigation base relative to some external reference frame.

2) Inertial Reference Frame
   a. Definition: A frame in which Newton's laws of motion are valid.
   b. Any non-accelerating frame (constant velocity and not rotating) with respect to an inertial frame is also inertial.
   c. "Inertial frames are any reference frames that move at a constant velocity and without rotation relative to frames in which the universe appears spherically symmetric." – Weinberg

3) Celestial Reference Frames
   a. Definition: Axes fixed relative to distant fixed stars.

4) International Celestial Reference Frame (ICRF):
   a. Standard for celestial reference frames.
   b. Axes fixed with respect to the positions of several hundred distant extragalactic sources of radio waves.
   c. Z-axis: Aligned with Earth's north pole.
   d. X-axis: Aligned with the vernal equinox in the direction of the Sun's position relative to Earth on the first day of spring.
   e. Origin: Center of mass of the solar system.

5) Geocentric Inertial Frame (GCI)
   a. Definition: An approximate inertial frame.
   b. Origin: Center of mass of Earth.
   c. The frame has linear acceleration because of Earth's circular orbit around the Sun (but this is unimportant for attitude analysis).
   d. Axes are aligned with the mean north pole and mean vernal equinox at some epoch.
   e. Denoted by the triad $\{i_1, i_2, i_3\}$.

6) Earth-Centered / Earth-Fixed Frame (ECEF)
   a. Denoted by $\{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$.
   b. Similar to the GCI frame.
   c. $\varepsilon_3 = i_3$.
   d. ε_1 points in the direction of Earth's prime meridian.
   e. ε_2 completes the right-hand rule.
   f. The ECEF frame rotates with the Earth (unlike the GCI frame).
   g. Rotation angle: Known as the Greenwich Mean Sidereal Time (GMST) angle and is denoted by θ_{GMST} (sidereal – of or relating to the distant stars, i.e., constellations or fixed stars).

h. To determine the ECEF position vector from geodetic coordinates (λ - latitude, φ - longitude, h - height), we first compute the distance between the z-axis and the normal to the ellipsoid.

7) Local Vertical / Local Horizontal Frame (LVLH)

a. Referenced to the spacecraft's orbit.
b. Useful especially for Earth-pointing spacecraft.
c. Identified by the subscript O.
d. z-axis (O_3): Points along the nadir vector, directly towards the center of the Earth from the spacecraft.
e. y-axis (O_2): Points along the negative orbit normal, in the direction opposite to the spacecraft's orbital angular velocity vector.
f. x-axis (O_1): Completes the right-handed triad.
g. Specific Angular Momentum (h_I)

$$h_I = r_I \cdot v_I$$

h. Flight Path Angle (γ)

$$\gamma = \cos^{-1}\left(\frac{h}{r \cdot v}\right)$$

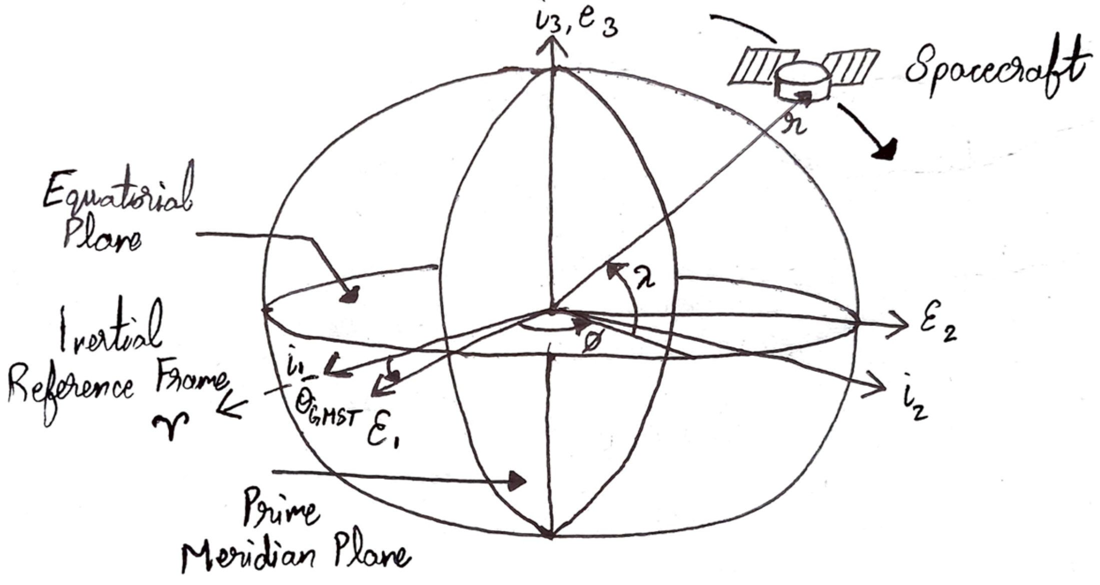


Fig. Definitions of various reference frames

$$\left.\begin{aligned} o_{3I} &= -r_I / \|r_I\| = -g_3 r_I \\ o_{2I} &= -(r_I \times v_I) / \|r_I \times v_I\| = -g_2 (r_I \times v_I) \\ o_{\cancel{3}1I} &= o_{2I} \times o_{3I} = g_2 g_3 (r_I \times v_I) \times r_I \\ &= g_2 g_3 \left[\|r_I\|^2 v_I - (r_I \cdot v_I) r_I \right] \end{aligned}\right\} \text{representation of } o_{3I},\ o_{2I},\ \&\ o_{1I} \text{ vectors in an intertial frame } I.$$

C. *Transformations*

1) Transformations can be classified into passive interpretation (alias sense) or active interpretation (alibi sense).

   a. Passive Interpretation: Unrotated vector, rotated reference frame.
   b. Active Interpretation: Rotated vector, unrotated reference frame.

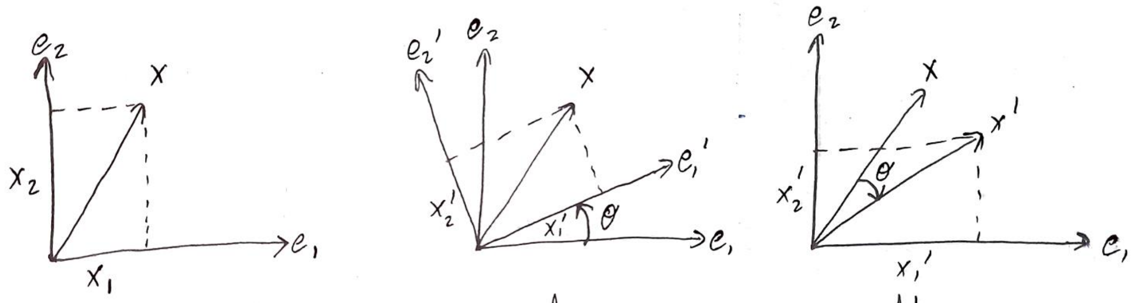


fig. components

fig. alias sense interpretation of transformation

fig. alibi sense interpretation of transformation

2) Direction Cosine Matrices (DCMs)

   a. DCMs are special orthogonal/orthonormal matrices.
   b. An Attitude matrix or Rotation matrix is a DCM which belongs to the set of matrices called SO(3), i.e., Special Orthonormal 3x3 matrix.

$$x_F = D_{FF'} \times x_{F'}$$

$$D_{FF'} = \begin{bmatrix} e_1 \cdot e_1' & e_1 \cdot e_2' & --- & e_1 \cdot e_n' \\ e_2 \cdot e_1' & e_2 \cdot e_2' & --- & e_2 \cdot e_n' \\ \vdots & \vdots & \ddots & \vdots \\ e_n \cdot e_1' & e_n \cdot e_2' & --- & e_n \cdot e_n' \end{bmatrix}$$

D. *Euler's Theorem and Euler Vectors*

Euler's vector and Euler's theorem are fundamental concepts in the field of spacecraft attitude control. They provide a mathematical framework for describing and controlling the orientation of a spacecraft.

1) *Euler's Theorem*

Euler's theorem states that any rotation of a rigid body in three-dimensional space can be described as a single rotation about a fixed axis. This axis is known as the Euler axis, and the angle of rotation about this axis is the Euler angle.

2) *Euler's Vector*

Euler's vector (or rotation vector) is a compact representation of the orientation of a rigid body. It combines the Euler axis and the Euler angle into a single vector. If ***e*** is the unit vector along the Euler axis and θ is the Euler angle, then the Euler vector ***v*** is given by

$$v = \theta e$$

E. *Requirements for ADCS in CubeSats*

1) *Mission Pointing Direction*

a. The CubeSat must maintain a specific orientation relative to an inertial reference frame or a celestial body to fulfill its mission objectives. The mission pointing direction will be determined based on the requirements of onboard instruments such as solar arrays, high gain antennas, and optical instruments.

2) *Pointing Accuracy*

a. Knowledge Accuracy (Sensors): The sensors used for attitude determination must provide accurate measurements of angular velocity, acceleration, and magnetic field strength. The sensor's accuracy directly affects the knowledge accuracy, which is the CubeSat's ability to know its orientation precisely.
b. Control Accuracy (Actuators): The control accuracy depends on the performance of actuators like reaction wheels, thrusters, and motor drivers. Accurate control of the actuators ensures precise adjustments to the CubeSat's orientation.

3) *Translational and Rotational Requirements*

a. Movement along the X, Y, and Z axes should be minimal to prevent interference with attitude control. The CubeSat must maintain specific angular rates and angles within defined durations and frequencies to ensure mission success.

4) *Select Attitude Control System:* Based on the pointing requirements, thrust vector, control authority, maneuver slew rate, and solar panel area, the CubeSat can adopt various attitude control systems.

a. Spin Stabilize: The spacecraft spins around its principal axis, creating gyroscopic stability due to the conservation of angular momentum. Thrusters correct spin rate changes due to parallel disturbance torques and correct perpendicular disturbance causing precession.
b. Dual Spin: The main spacecraft body spins to provide gyroscopic stability. A despin platform, such as an instrument or antenna, remains stationary or rotates independently for precise pointing accuracy.
c. 3-Axis Stabilize: Active stabilization using gyros and periodic updates via star and horizon scanning. Attitude errors are corrected with reaction wheels and thrusters, allowing control of all three axes (roll, pitch, yaw).
d. Momentum Bias: Uses a momentum wheel to provide inertial stiffness to two axes while controlling wheel speed manages the third axis.
e. Gravity Gradient: Uses Earth's gravitational gradient to passively stabilize the spacecraft's long axis with the gravity vector.

5) *Internal and External Disturbance Torques*

a. Drag Torque: Interaction with the atmosphere in Low Earth Orbit (LEO) causes torque if the Center of Pressure (COP) does not coincide with the Center of Mass (COM).
b. Solar Torque: Pressure from solar radiation (solar winds) exerts force on solar panels and body panels, causing torque.
c. Gravity Gradient Torque: Long spacecrafts experience gravity forces from upper extremities aligning themselves with Earth's radius vector.

d. Magnetic Torque: Interaction between Earth’s magnetic field and the magnetic dipole moment of the spacecraft generates torque.
e. Spacecraft-Generated Torque: Internal mechanisms like reaction wheels and control moment gyroscopes generate torque for attitude control.

6) *Hardware Elements*

a. Microcontroller: Central processing unit executing control algorithms and managing data.
b. IMU Sensor: Provides measurements of angular velocity, acceleration, and magnetic field strength.
c. Motor and Encoder: Drives the reaction wheel and provides feedback on motor position and speed.
d. Motor Driver: Interfaces between the microcontroller and the motor, providing the necessary power and control signals.
e. Bluetooth Module: Facilitates wireless communication for remote control and data logging.
f. Voltage Regulator: Ensures a stable power supply for the CubeSat’s components.
g. Battery: Primary energy storage device providing power to the CubeSat.

7) *Control Law and Determination Methods*

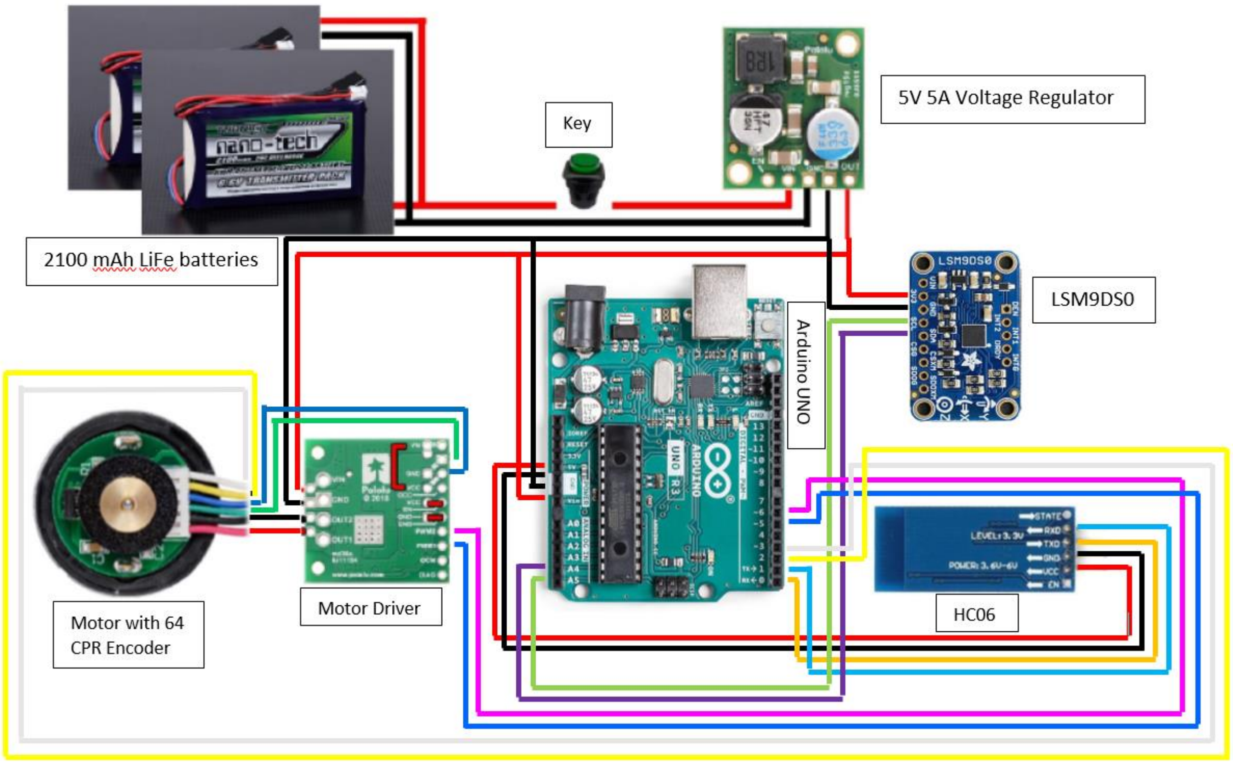
a. Proportional Control: Adjusts the motor speed gradually to achieve the desired gyroscope Z-axis rate.
b. Sensor Fusion: Combines data from multiple sensors for accurate attitude determination.
c. PID Control: Implement PID (Proportional-Integral-Derivative) control for more precise attitude adjustments.

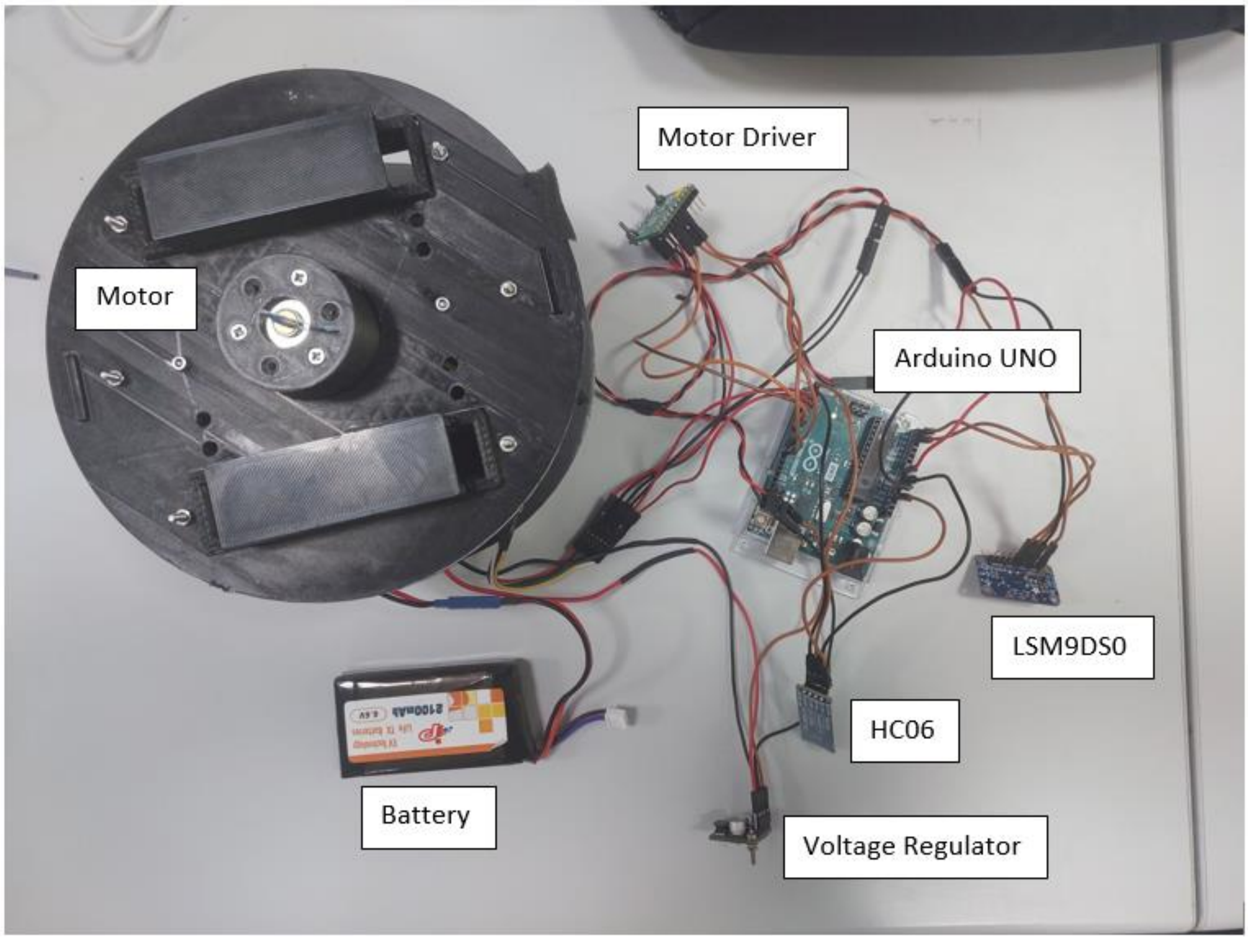
8) *Trade and Selection*

a. Proportional Control: Adjusts the motor speed gradually to achieve the desired gyroscope Z-axis rate.
b. Sensor Fusion: Combines data from multiple sensors for accurate attitude determination.
c. PID Control: Implement PID (Proportional-Integral-Derivative) control for more precise attitude adjustments.

III. Hardware Setup

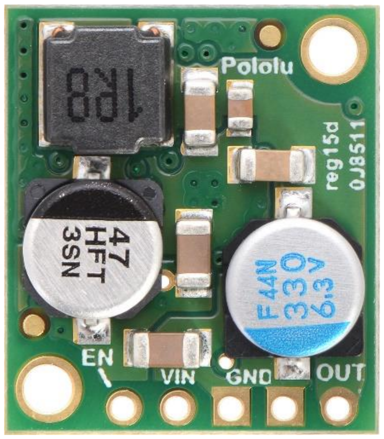
A. Schematic Layout





B. Components

1) EPS (Electrical Power Subsystem)
   a. Voltage Regulator: Pololu 5V, 5A Step-Down Voltage Regulator D24V50F5



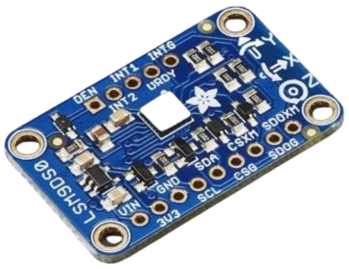
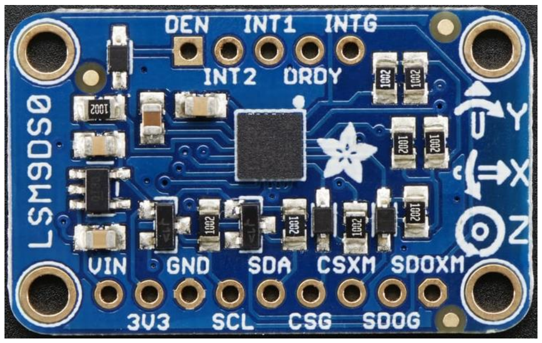
A voltage regulator is a crucial component in the power subsystem of a CubeSat. It ensures that the voltage supplied to various components is consistent and within their operating limits. The Pololu 5V, 5A Step-Down Voltage Regulator D24V50F5 is a high-efficiency switching regulator that converts higher input voltages (up to 38V) to a stable 5V output. It is capable of supplying up to 5A of current, making it suitable for powering multiple components in the CubeSat.

   b. Battery: 2100mAh LiFe Battery 6.6V

TURNIGY power systems
nano-tech
2100mAh 20C DISCHARGE
HIGH-DISCHARGE-LIFEPO4-BATTERY
6.6V TRANSMITTER PACK
Please read safety warning & usage guidelines on reverse side

The battery is the primary energy storage device in the CubeSat's EPS. A 2100mAh LiFe (Lithium Iron Phosphate) battery with a 6.6V rating provides a stable and safe power source. LiFe batteries are preferred for their safety, longer cycle life, and thermal stability compared to other lithium-ion batteries. The battery's capacity (2100mAh) indicates the amount of charge it can store, which is critical for ensuring that the CubeSat can operate through periods of eclipse or low sunlight.

2) ADCS (Attitude Determination and Control System) Subsystem
   a. IMU Sensor: LSM9DS0

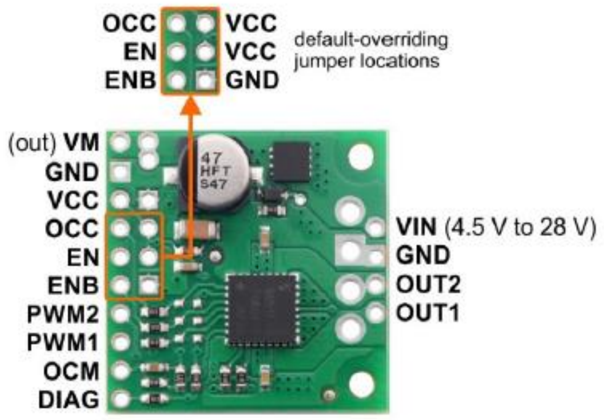



The Inertial Measurement Unit (IMU) sensor, such as the LSM9DS0, is used for attitude determination in the ADCS of a CubeSat. The LSM9DS0 integrates a 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer. It provides precise measurements of acceleration, angular velocity, and magnetic field strength, which are essential for determining the orientation and movement of the CubeSat. This data is used by the control algorithms to adjust the CubeSat's attitude to the desired orientation.

b. Motor: Pololu 12V Motor with 64 CPR Encoder for 37D mm Metal Gear motor (item 4750)

A DC motor with an encoder is used in the reaction wheel or control moment gyroscope of the ADCS to control the CubeSat's orientation. The Pololu 12V motor with a 64 CPR (Counts Per Revolution) encoder provides feedback on the motor's position and speed. This feedback is crucial for precise control of the motor, allowing the ADCS to make accurate adjustments to the CubeSat's attitude.

c. Motor Driver: Pololu TB9051FTG Single Channel Brushed DC Motor Driver

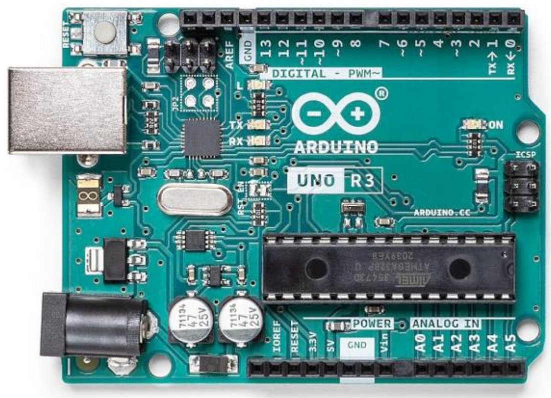


The motor driver interfaces between the microcontroller and the motor, providing the necessary power and control signals to drive the motor. The Pololu TB9051FTG is a robust and efficient motor driver capable of delivering high currents and voltages required by the motor. It includes features such as over-current protection, under-voltage lockout, and fault diagnostics, ensuring reliable operation of the motor in the CubeSat's ADCS.

d. Reaction Wheel

3) CDH (Command and Data Handling) Subsystem

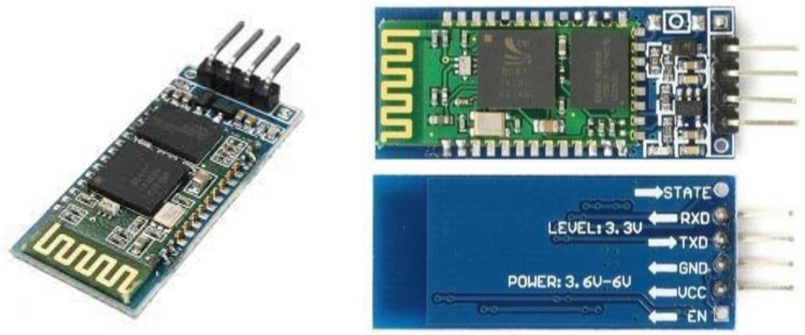
a. Microcontroller: Arduino Uno



The Arduino Uno is a widely used microcontroller board based on the ATmega328P. In the CDH subsystem, the Arduino Uno acts as the central processing unit, executing control algorithms, processing sensor data, and managing communication between different subsystems. It provides various interfaces (digital I/O, analog inputs, UART, I2C, SPI) for connecting sensors, actuators, and communication modules. Its ease of programming and extensive community support make it an ideal choice for rapid development and prototyping of CubeSat control systems.

4) Communications Subsystem

a. Bluetooth Module: HC-06



The HC-06 Bluetooth module is used for wireless communication between the CubeSat and a ground station or other devices. It operates in the 2.4GHz ISM band and provides a simple serial interface for data transmission. In the COMMS subsystem, the HC-06 allows for remote control, data logging, and diagnostics of the CubeSat during testing and operation. Its ease of integration and reliable wireless communication make it a valuable component for the CubeSat's communication needs.

C. Pin Connections for All Components

1) 5V Regulator

OUT1	VIN (Motor Driver)
	VIN (Arduino UNO)
	VIN (LSM9DS0)
GND	GND (Motor Driver)
	GND (Arduino UNO)
	GND (LSM9DS0)
VIN	Battery +ve Terminal
GND	Battery –ve Terminal

2) Motor with 64 CPR Encoder

White (Encoder Channel A)	D3 (Arduino UNO)
Yellow (Encoder Channel B)	D2 (Arduino UNO)
Blue (Encoder VCC)	VIN (Motor Driver)
Green (Encoder GND)	GND (Motor Driver)
Black (Motor Terminal)	OUT2 (Motor Driver)
Red (Motor Terminal)	OUT1 (Motor Driver)

3) Bluetooth Module (HC06)

VCC	5V (Arduino UNO)
GND	GND (Arduino UNO)
RXD	TX – D1 (Arduino UNO)
TXD	RX – D0 (Arduino UNO)

4) IMU sensor (LSM9DS0)

VIN	VIN (5V regulator)
GND	GND (5V regulator)
SDA	A4 (Arduino UNO)
SCL	A5 (Arduino UNO)

5) Microcontroller (Arduino UNO)

VIN	OUT1 (5V Regulator)
GND	GND (5V Regulator), GND (HC06)
5V	VCC (HC06)
A4	SDA (LSM9DS0)
A5	SCL (LSM9DS0)
D0 – RX	TXD (HC06)
D1 – TX	RXD (HC06)
D2	Encoder Channel B
D3	Encoder Channel A
D5	PWM1 (Motor Driver)
D6	PWM2 (Motor Driver)

6) Motor Driver

VIN	OUT1 (5V Regulator)
GND	GND (5V Regulator)
OUT1	Red (Motor Terminal)
OUT2	Black (Motor Terminal)
GND	Green (Encoder GND)
VIN	Blue (Encoder VCC)
PWM1	D5
PWM2	D6

D. Test Plan

The test plan outlines the specific tests you will perform to verify the functionality and performance of your system.

E. Objective

The purpose of this testing is to verify the functionality and performance of the FloatSat's ADCS system, including sensor accuracy, motor control, and communication reliability.

F. Test Cases

1) Test Case 1: Sensor Initialization and Calibration
    a. Objective: Verify that the LSM9DS0 IMU sensor initializes correctly and provides accurate readings.
    b. Procedure:
        - Power on the system.
        - Check serial output for successful initialization messages.
        - Compare sensor readings (accelerometer, gyroscope, magnetometer) against known reference values or a secondary measurement tool.
    c. Expected Results: Sensor readings should be within acceptable error margins compared to reference values.
2) Test Case 2: Motor Encoder Feedback
    a. Objective: Ensure the motor encoder provides accurate RPM feedback.

b. Procedure:
   - Set the motor to a known PWM value.
   - Measure the RPM using the encoder.
   - Compare the encoder RPM reading with a reference tachometer.

c. Expected Results: Encoder RPM should closely match the reference tachometer reading.

3) Test Case 3: Motor Control Algorithm

a. Objective: Validate the proportional control algorithm's ability to adjust motor speed gradually to achieve the desired Gyro Z value.

b. Procedure:
   - Set a target Gyro Z value via Bluetooth.
   - Monitor the motor speed adjustment and Gyro Z value.
   - Ensure motor speed increases or decreases gradually and stabilizes at the target value.

c. Expected Results: Motor speed should adjust smoothly, and Gyro Z should reach and maintain the target value.

4) Test Case 4: Communication Reliability

a. Objective: Test the reliability of Bluetooth communication.

b. Procedure:
   - Send commands via Bluetooth to change motor speed.
   - Log any communication errors or delays.

c. Expected Results: Commands should be received and executed promptly without errors.

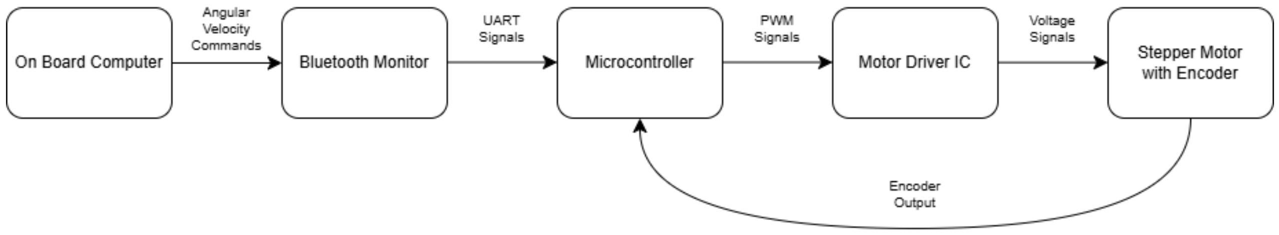
5) Test Case 5: System Integration

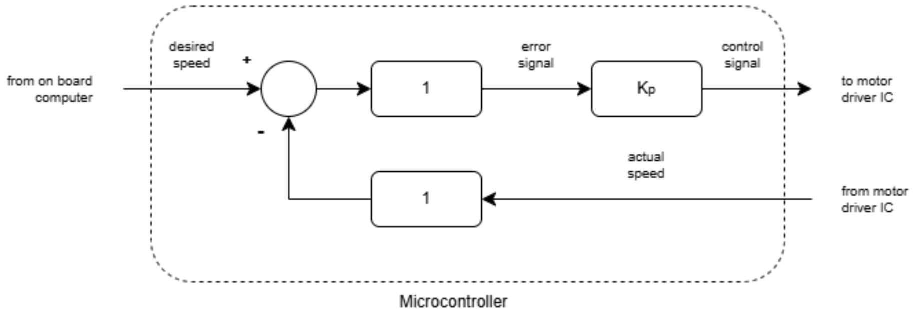
a. Objective: Integrate sensors and verify actuation of motor via Bluetooth using the control algorithm

b. Procedure:
   - Send commands via Bluetooth to change to target Gyro Z value by taking in current orientation from IMU sensor

c. Expected Results: Proper execution of the project mission

G. Control Loop



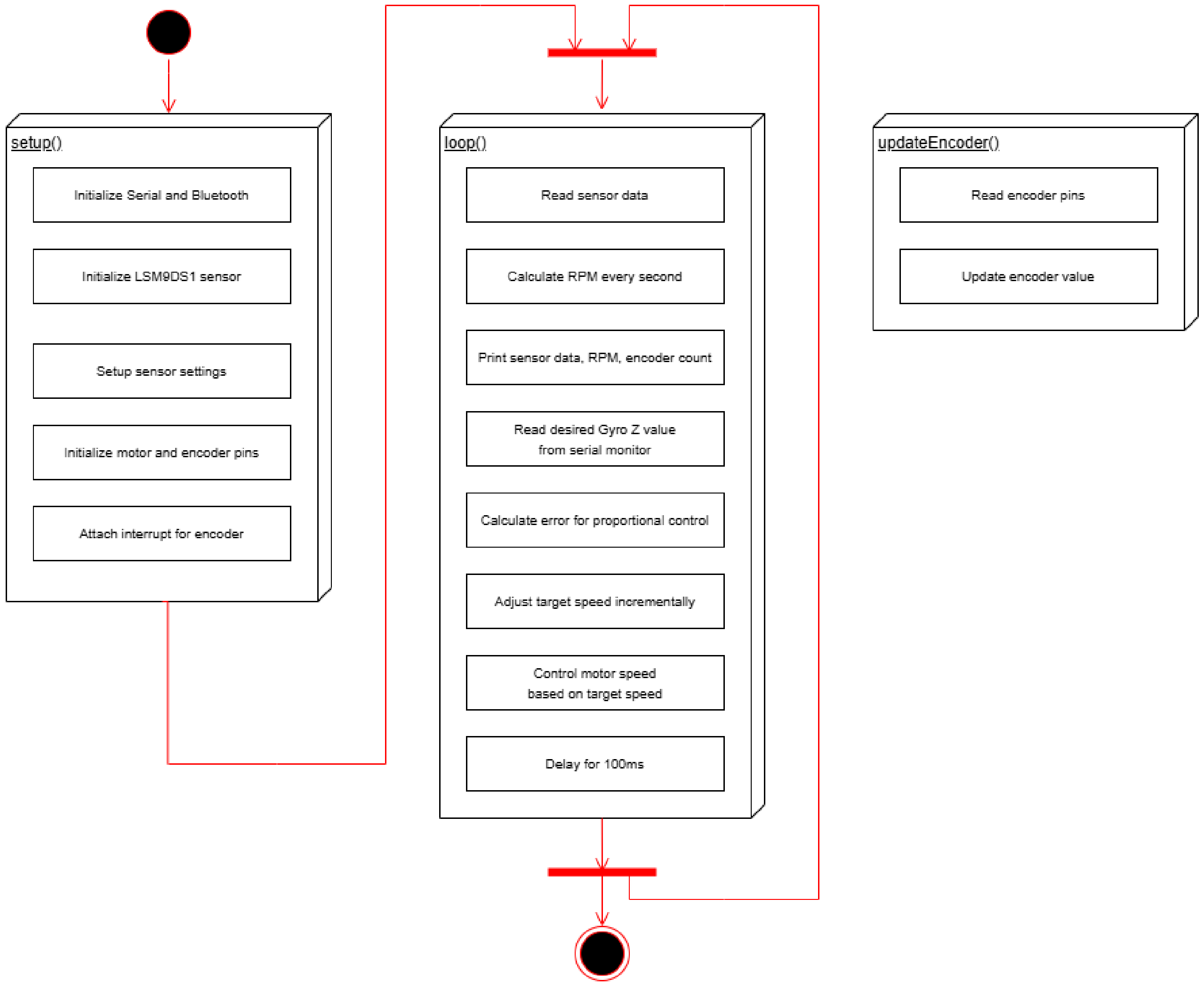


IV. Software Setup

A. *Software Configuration*

1) *Arduino IDE:*
   - Version 2.3.2
2) *Libraries:*
   - Wire
   - SPI
   - Adafruit_LSM9DS1 (v2.2.1)
   - Adafruit_Sensor
   - SoftwareSerial
3) *Bluetooth Communication Medium:*
   - Android Bluetooth Serial app was used to send and receive UART data through the HC06 module

B. *Algorithm (UML)*


setup()
Initialize Serial and Bluetooth
Initialize LSM9DS1 sensor
Setup sensor settings
Initialize motor and encoder pins
Attach interrupt for encoder
loop()
Read sensor data
Calculate RPM every second
Print sensor data, RPM, encoder count
Read desired Gyro Z value from serial monitor
Calculate error for proportional control
Adjust target speed incrementally
Control motor speed based on target speed
Delay for 100ms
updateEncoder()
Read encoder pins
Update encoder value

C. Code

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_LSM9DS1.h>
#include <Adafruit_Sensor.h>
#include <SoftwareSerial.h>

// LSM9DS1 pins
#define LSM9DS1_SCK A5
#define LSM9DS1_MOSI A4

Adafruit_LSM9DS1 lsm = Adafruit_LSM9DS1();

SoftwareSerial BTSerial(0, 1); // RX, TX

// Motor and Encoder pins
#define motorA 5
#define motorB 6
int encoderPin1 = 2;
int encoderPin2 = 3;

volatile int lastEncoded = 0;
volatile long encoderValue = 0;
volatile long correctEncoderValue = 0;
volatile long lastencoderValue = 0;
int lastMSB = 0;
int lastLSB = 0;
int targetSpeed = 0;

unsigned long lastTime = 0; // Variable to store the last time
the RPM was calculated
unsigned long lastTime1 = 0; // Variable to store the last
time data was printed in the serial monitor
float rpm = 0; // Variable to store the calculated RPM
const int CPR = 64; // Counts per revolution of the encoder

float targetGyroZ = 0; // Target Gyro Z value in deg/s
float currentGyroZ = 0; // Current Gyro Z value in deg/s
float currentAccelZ = 0;
float Kp = 0.5; // Proportional gain for the controller

float error = 0; // Variable to sstore the calculated error
int adjustment = 0; // Variable to store the calculated
adjustment to targetSpeed
int adjustmentStep = 10; // Variable to control max abs value
of adjustment

void setup()
{
        Serial.begin(115200);
        while (!Serial)
        {
                delay(1); // wait for Serial to connect
        }
        BTSerial.begin(115200);
        while (!BTSerial)
        {
                delay(1); // wait for Serial to connect
        }
        Wire.begin();
        if (!lsm.begin())
        {
                Serial.println("Unable to initialize the
LSM9DS1 9DOF");
                while (1);
        }
        Serial.println("Found LSM9DS1 9DOF");

        // Initialize sensor settings
        setupSensor();

        // Initialize motor and encoder pins
        pinMode(encoderPin1, INPUT);
        pinMode(encoderPin2, INPUT);
        digitalWrite(encoderPin1, HIGH); // Turn pull-up
resistor on
        digitalWrite(encoderPin2, HIGH); // Turn pull-up
resistor on
        attachInterrupt(digitalPinToInterrupt(encoderPin1),
updateEncoder, CHANGE);
        attachInterrupt(digitalPinToInterrupt(encoderPin2),
updateEncoder, CHANGE);

        pinMode(motorA, OUTPUT);
        pinMode(motorB, OUTPUT);
}

void loop()
{
        // Read sensor data
        sensors_event_t accel, mag, gyro, temp;
        lsm.getEvent(&accel, &mag, &gyro, &temp);
        currentGyroZ = gyro.gyro.z * 57.29578; // Convert
to deg/s
        currentAccelZ = accel.acceleration.z;

        // Calculate RPM every second
        correctEncoderValue = encoderValue / 4;
        if (millis() - lastTime >= 1000)
        {
                noInterrupts(); // Disable interrupts while
calculating RPM
                rpm = (abs(correctEncoderValue) /
(float)CPR) * 60.0;
                encoderValue = 0;
                correctEncoderValue = 0; // Reset the
counter
                lastTime = millis(); // Update the last time
                interrupts(); // Re-enable interrupts
        }

        if (millis() - lastTime1 >= 1000)
        {
                // Print sensor data
                Serial.print("Accel Z: ");
Serial.print(currentAccelZ); Serial.print(" m/s^2");
```

```
			Serial.print("Gyro Z: ");
Serial.print(currentGyroZ); Serial.println(" deg/s");

			// Print RPM and encoder count to serial
monitor
			Serial.print("Encoder Count: ");
			Serial.println(correctEncoderValue);
			Serial.print("RPM: ");
			Serial.println(rpm);
			Serial.println();
			lastTime1 = millis(); // Update the last
time
		}

		// Read desired Gyro Z value from serial monitor
		if (Serial.available())
		{
			targetGyroZ = Serial.parseFloat();
			Serial.print("Target Gyro Z = ");
			Serial.print(targetGyroZ);
			Serial.println(" deg/s");
			Serial.println();
		}

		// Proportional control to adjust motor speed
		error = targetGyroZ - currentGyroZ;
		adjustment = (int)(Kp * error);

		// Constrain adjustment to prevent sudden changes
in speed
		adjustment = constrain(adjustment, -
adjustmentStep, adjustmentStep);

		targetSpeed += adjustment;

		// Constrain targetSpeed to valid PWM range
		targetSpeed = constrain(targetSpeed, -255, 255);

		Serial.print("Target Speed = ");
		Serial.println(targetSpeed);

		// Control motor speed based on targetSpeed
		if(targetSpeed >= 0)
		{
			analogWrite(motorA, targetSpeed);
			digitalWrite(motorB, HIGH);
		}
		else
		{
			analogWrite(motorA, -targetSpeed);
			digitalWrite(motorB, LOW);
		}

		delay(100);
}

void setupSensor()
{
		// Set the accelerometer range
		lsm.setupAccel(lsm.LSM9DS1_ACCELRANGE_2
G, lsm.LSM9DS1_ACCELDATARATE_10HZ);

		// Set the magnetometer sensitivity
		lsm.setupMag(lsm.LSM9DS1_MAGGAIN_4GAU
SS);

		// Setup the gyroscope
		lsm.setupGyro(lsm.LSM9DS1_GYROSCALE_245
DPS);
}

void updateEncoder()
{
		int MSB = digitalRead(encoderPin1); // MSB =
most significant bit
		int LSB = digitalRead(encoderPin2); // LSB = least
significant bit

		int encoded = (MSB << 1) | LSB; // Converting the
2 pin value to a single number
		int sum = (lastEncoded << 2) | encoded; // Adding
it to the previous encoded value

		if (sum == 0b1101 || sum == 0b0100 || sum ==
0b0010 || sum == 0b1011) encoderValue++;
		if (sum == 0b1110 || sum == 0b0111 || sum ==
0b0001 || sum == 0b1000) encoderValue--;

		correctEncoderValue += (sum == 0b1101 || sum ==
0b0100 || sum == 0b0010 || sum == 0b1011) ? 1 : -1;

		if (correctEncoderValue >= CPR ||
correctEncoderValue <= -CPR)
		{
			correctEncoderValue = 0; // Reset after a
full rotation
		}

		lastEncoded = encoded; // Store this value for next
time
}
```

V. Working Principle

A. Key Components:

1) *Flywheel:* Stores rotational energy.

2) *Electric Motor:* Drives the flywheel.

3) *Control System:* Manages the speed and direction of the flywheel to achieve the desired orientation.

B. Mathematical Models:

The dynamics of a reaction wheel system can be described using the principles of rotational motion and the conservation of angular momentum.

1) *Angular Momentum Conservation*

The total angular momentum *H* of the spacecraft and the reaction wheel system is conserved. It can be expressed as:

$$H = I_s \omega_s + I_w \omega_w$$

where:

- I_s is the moment of inertia of the spacecraft.
- $\boldsymbol{\omega}_s$ is the angular velocity of the spacecraft.
- I_w is the moment of inertia of the reaction wheel.
- $\boldsymbol{\omega}_w$ is the angular velocity of the reaction wheel.

2) *Torque and Angular Acceleration*

The torque *T* applied by the reaction wheel motor changes the angular velocity of the wheel and, consequently, the spacecraft. The relationship is given by:

$$T = I_w \frac{d\omega_w}{dt}$$

This torque also induces an equal and opposite reaction on the spacecraft:

$$T = -I_s \frac{d\omega_s}{dt}$$

3) *Control Law*

To achieve a desired orientation, a control law is implemented. One common approach is the Proportional-Derivative (PD) control, which adjusts the wheel speed based on the error in orientation and its rate of change:

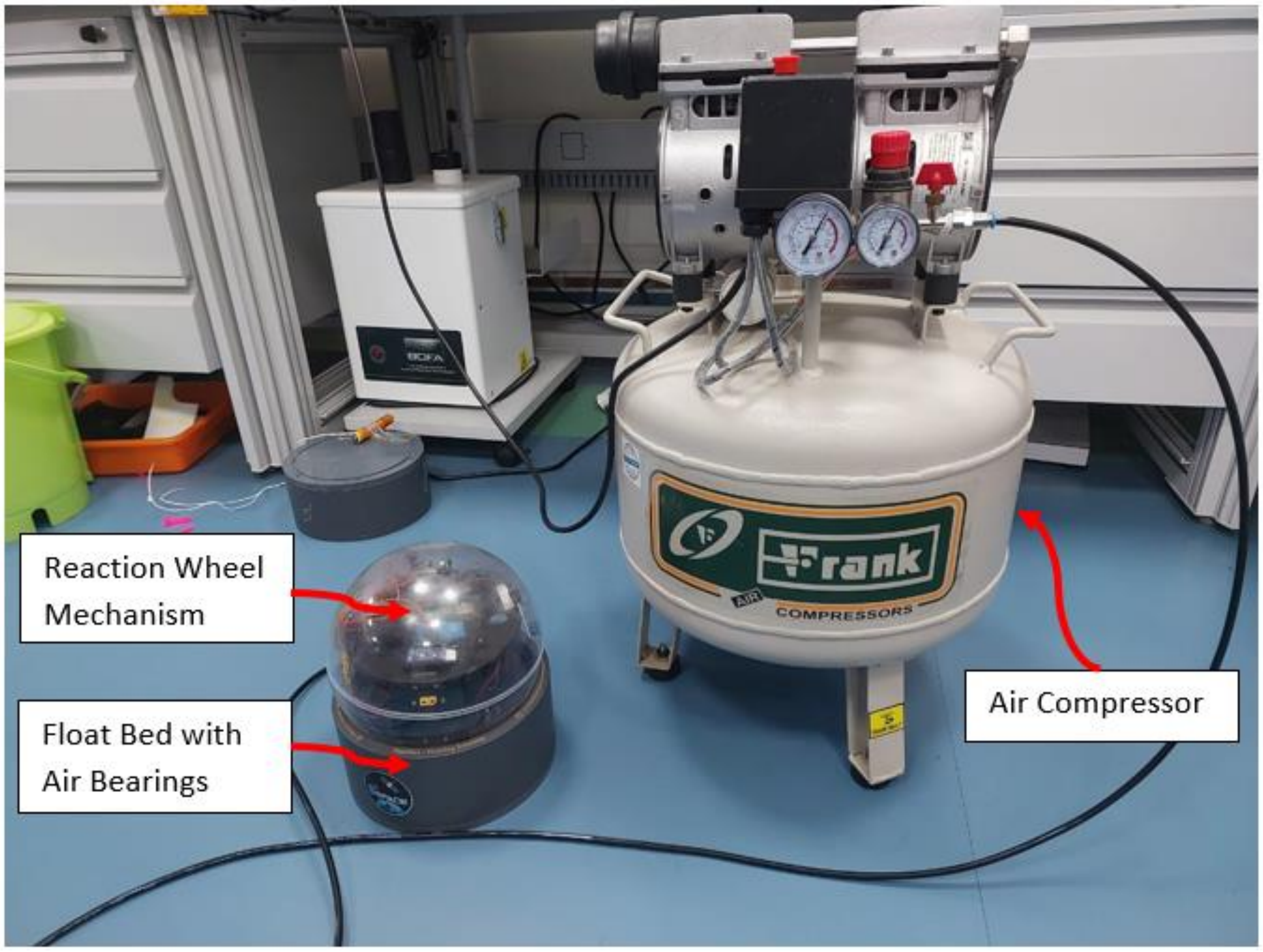
$$T = -K_p e - K_d \frac{de}{dt}$$

where:

- *e* is the orientation error.
- K_p and K_d are the proportional and derivative gains, respectively.

VI. Testing And Results

A. Experimental Setup



The above setup exhibited a constant rate of rotation in both directions, when controlled using external signals.

1) The first case is where the module is maintaining its 0 deg/s state and not rotating
2) The second case is where the module is made to spin in the anticlockwise direction and maintain that rotational rate
3) The third case is where the module is made to spin in the clockwise direction and maintain that rotational rate

VII. Future Scope

The current project establishes a foundational ADCS system for a CubeSat using a mono-wheel reaction wheel. Future work can expand and enhance this system in several ways.

A. *Multi-Axis Control:*

Extend the system to include multiple reaction wheels, enabling control over all three rotational axes (pitch, yaw, and roll) for complete attitude control.

B. *Advanced Control Algorithms:*

Implement more sophisticated control algorithms such as PID (Proportional-Integral-Derivative) control or model predictive control to improve the precision and responsiveness of the system.

C. *Integration with Other Sensors:*

Incorporate additional sensors such as sun sensors, star trackers, or magnetometers to enhance the attitude determination accuracy and robustness.

D. *Miniaturization and Optimization:*

Optimize the hardware design for weight and power consumption, ensuring the system is suitable for integration into a fully functional CubeSat.

E. *Environmental Testing:*

Conduct extensive testing in simulated space environments, including thermal vacuum testing and vibration testing, to ensure the system's reliability and performance under space conditions.

F. *Deployment and In-Orbit Testing:*

Collaborate with space agencies or commercial launch providers to deploy the CubeSat in orbit and validate the ADCS performance in real-world space conditions.

References

[1] A. R. Sergio Montenegro, “Course on floatsat,” Ph.D. dissertation, Julius-Maximilians- University Wuerzburg, Germany, 2021.

[2] I. Virgala and M. Kelemen, “Experimental friction identification of a dc motor,” International journal of mechanics and applications, vol. 3, no. 1, pp. 26–30, 2013.

[3] I. Virgala, P. Frankovsky, and M. Kenderova, “Friction effect analysis of a dc motor,” American Journal of Mechanical Engineering, vol. 1, no. 1, pp. 1–5, 2013.

[4] Polulu-Electronics, https://www.pololu.com/product/2997

[5] ST-Microelectronics, https://cdn-shop.adafruit.com/datasheets/LSM9DS0.pdf

[6] Polulu-Electronics, https://www.pololu.com/product/4750/resources.

[7] Fundamentals of Spacecraft Attitude Determination and Control - F. Landis Markley & John L. Crassidis

[8] Elements of Spacecraft Design – Charles D. Brown

[9] Spacecraft Dynamics and Control – Michael J. Rycroft & Robert F. Stengel

[10] An Educational Platform for Testing and Evaluating Satellite Control Algorithms in a Real-Time and Frictionless Environment 30 December 2022