

Delfi-Twin ADCS

AE4897-25: Space Engineering Practical
Group B



Delfi-Twin ADCS

by

Group B

Student Name	Student Number
Maurizio Pavel	5797144
Fatih Gök	6387276
Abhishek Avadhanam	6435548
Tjalling Huson	5109698

Project Tutor: J. Bouwmeester
Faculty: Faculty of Aerospace Engineering, Delft

Cover: Canadarm 2 Robotic Arm Grapples SpaceX Dragon by NASA
under CC BY-NC 2.0 (Modified)
Style: TU Delft Report Style, with modifications by Daan Zwaneveld

Contents

Nomenclature	v
Review	vi
1 Introduction	1
2 Test Article	2
2.1 Design	2
2.1.1 Test article	2
2.1.2 Test jig	6
2.1.3 Horizon simulant	9
2.1.4 Software design	12
2.1.5 Data Pipeline	16
2.2 Configuration	18
2.2.1 Detumbling Test Setup	19
2.2.2 Nadir Pointing Test Setup	20
2.3 Test set-up	21
2.3.1 Overview	21
3 Requirements	24
3.1 Definition	24
3.2 Verification by Analysis	27
3.2.1 Requirement ACT-1	27
3.2.2 Requirement ACT-1.1	27
3.2.3 Requirement ACT-1.2	27
3.2.4 Requirement ACT-3*	27
3.3 Verification by Review of Design	28
3.3.1 Requirement MTQ-01	28
3.3.2 Requirement MTQ-02	28
3.4 Verification by Test	28
3.4.1 Requirement PWR-1	28
3.4.2 Requirement PWR-4	29
3.4.3 Requirement PWR-6	29
3.4.4 Requirement REC-1	29
3.4.5 Requirement REC-3	29
3.4.6 Requirement SND-4	29
3.4.7 Requirement ACT-1	30
3.4.8 Requirement ACT-1.1	30
3.4.9 Requirement ACT-1.2	30
3.4.10 Requirement ACT-3*	30
3.4.11 Requirement ACT-4	30
3.4.12 Requirement EHS-02	31
3.5 Compliance matrix	31
4 Hypothesis formulation and assessment	33
4.1 Formulation	34
4.2 Evaluation	36
4.3 Assessment	37
5 Experiment Design	38
5.1 Experiment definition	38

5.2	Measurables	40
5.2.1	Experiment variables: Test-case Detumbling	40
5.2.2	Experiment variables: Test-case Nadir Pointing	40
5.3	Sample size	41
5.4	Uncertainty quantification	41
5.5	Uncertainty evaluation	42
6	Data Acquisition and Processing	43
6.1	Acquisition	43
6.1.1	Architectural Philosophy	44
6.1.2	Acquisition Chain	45
6.2	Processing	46
6.2.1	Event Reconstruction	46
6.2.2	Dataset Reconstruction	47
6.2.3	Validation	49
6.2.4	Reference Time Origins	50
6.2.5	Rotation Convention	50
6.2.6	Rotation Matrices	50
6.2.7	Angular Velocity Vector	51
6.2.8	Magnetic Reference Generation	51
6.2.9	Outlier Removal	52
6.2.10	Constant Inertial Field	53
6.2.11	Body-Frame Reference	53
6.2.12	Reference Magnetic-Field Derivative	53
6.2.13	Residuals	54
6.2.14	Magnetic Torque Physics	54
6.2.15	Predicted Angular Momentum Change	55
6.2.16	Reference Angular Momentum Change	55
6.2.17	Angular Momentum Error	57
6.2.18	Coordinate Frames	57
6.2.19	Angular Diagnostics	58
6.2.20	Two Independent Validation Layers	58
6.2.21	Statistics	59
6.2.22	Mathematical Definitions	60
6.2.23	Export Structure	60
6.2.24	Plot Generation	61
6.2.25	Overall Processing Chain	61
6.3	Nadir pointing tests	63
6.3.1	Nadir processing	63
7	Test Plan	65
7.1	High-Level Test Plan	65
7.2	Test Plan	67
7.2.1	Preparation	68
7.2.2	Powering up	69
7.2.3	Set up	69
7.2.4	Functionality Test	70
7.2.5	Testing	70
7.2.6	Finishing up	71
7.3	Personnel required	71
7.4	Safety aspects	72
8	Test Results	75
8.1	Dry Run	75
8.1.1	Horizon Simulant Results	75
8.1.2	Magnetic Field Reconstruction Validation	78
8.1.3	Angular Momentum Reconstruction Validation	79
8.1.4	Dry-Run Uncertainty Summary	80

8.2	Final Test Campaign	82
8.2.1	Nadir-Pointing Results	82
8.3	Requirement Verification	87
9	Conclusion	88
9.1	Recap of our goals and results	88
9.2	Future Recommendations	89
9.2.1	System Engineering Work	89
9.2.2	HW-SW Interfaces	89
9.2.3	Production and Manufacturing	90
9.2.4	EHS	90
9.3	Future studies	90
9.3.1	EHS	90
9.3.2	Magnetorquers	91
9.3.3	Stepper motor	91
	References	92
A	Figures	93
B	Requirements	94
C	Software development steps	98
D	Framework Parameters and Conventions	101
D.1	Communication Parameters	101
D.2	Event-Frame Structure	101
D.3	Outlier Parameters	101
D.4	Angular Diagnostic Quantities	101
E	Plot and Export Implementation	102
E.1	File Naming	102
E.2	Export Directory Structure	102
E.3	Plot Parameters	102
E.4	Disabled Figures	103
F	Magnetometer Alignment Calibration	104
F.1	Calibration Philosophy	104
F.2	Alignment Direction Cosine Matrix	104
F.3	External-Fixed-Frame Reconstruction	104
F.4	Constant Inertial Field Estimation	105
F.5	Forward Prediction	105
F.6	Residual Generation	105
F.7	Cost Function	105
F.8	Numerical Optimization	105
F.9	Calibration Dataset Selection	105
F.10	Exported Calibration Products	105
F.11	Angular-Diagnostic Numerical Conventions	106
G	Estimator Evaluation Framework	107
G.1	Framework Architecture	107
G.2	Dataset Loading	107
G.3	Estimator Execution	108
G.3.1	Causality Convention	108
G.4	Performance Metrics	108
G.4.1	Vector Error Metrics	108
G.4.2	Angular Error Metrics	108
G.4.3	Magnetic-Field Derivative Error	108
G.4.4	Derivative Smoothness	108
G.5	Campaign Aggregation	109
G.6	Outputs	109

G.6.1	Dataset Outputs	109
G.6.2	Campaign Outputs	109
G.6.3	Estimator Ranking	109
G.7	Figures	110
G.7.1	Dataset Figures	110
G.7.2	Campaign Figures	110
G.7.3	Angular-Velocity Statistics	110
G.8	Pareto Analysis	110
G.9	Overall Workflow	110
G.10	Framework Outputs	111
H	Estimator Implementations	112
H.1	Estimator Architecture	112
H.2	Standard Interface	112
H.3	Causality Assumptions	112
H.4	Allowed Information	113
H.5	Variable-Time Derivative	113
H.6	One-Step Prediction	113
H.7	Finite-Difference Estimator	114
H.8	Exponential Moving Average Estimator	114
H.9	Predictive Exponential Moving Average Estimator	114
H.10	Alpha-Beta Filter	115
H.11	Future Extensions	116
I	Parameter-Tuning Framework	117
I.1	Framework Architecture	117
I.2	Parameter Search	117
I.3	Candidate Parameters	117
I.4	Exponential Moving Average Parameters	118
I.5	Predictive Estimator Parameters	118
I.6	Alpha-Beta Parameters	118
I.7	Campaign Evaluation	118
I.8	Performance Metrics	119
I.9	Parameter Ranking	119
I.10	Tuning Figures	119
I.11	Exported Outputs	119
I.12	Overall Workflow	119

Nomenclature

Symbols

Symbol	Definition	Unit
B	Magnetic field strength	[T]
C	Electrical capacitance	[F]
I	Electric current	[A]
P	Power	[W]
R	Resistance	[Ω]
sr	Steradian	[sr]
t	Time	[s]
T	Temperature	[$^{\circ}\text{C}$]
U	Voltage	[V]
ϵ	Emissivity	[-]
τ	Torque	[Nm]
ω	Rotational rate	[$^{\circ}\text{s}^{-1}$]

Review

This section provides a list of reviewers for each section of this report.

Section	Reviewer
Introduction (Section 1)	Jasper Bouwmeester
Test Article (Section 2)	Jasper Bouwmeester
Requirements (Section 3)	Mehmet Sevket Uludag
Hypothesis formulation and assessment (Section 4)	Rudolf Saathof
Experiment Design (Section 5)	Rudolf Saathof
Data Acquisition (Section 6)	Stefano Speretta
Test Plan (Section 7)	Jyoti Botchu
Test Results (Section 8)	Jasper Bouwmeester
Conclusions (Section 9)	Jasper Bouwmeester

Table 2: Reviewers

1

Introduction

The Delfi Program is TU Delft's initiative to develop and demonstrate novel space technologies. As part of this program, the twin satellites of the Delfi-Twin project are planned to become the fourth project to join their predecessors in space. Specifically, the mission aims to deploy and operate two identical satellites as a technology demonstrator, with the goal of achieving formation flying using differential drag, while also investigating the poorly understood spontaneous E-layer in support of laser communication between ground stations and spacecraft. By providing students with the opportunity to develop their engineering skills in a real mission setting, this document captures the work concerning the testing and verification of the Delfi-Twin Attitude Determination and Control System subsystem.

As part of the Space Engineering Practical course (AE4897-26), this subsystem has been under development for two consecutive years, each covering a period of two quarters. During that time, two student teams, ours included, have worked on the subsystem. Each team first contributed to the design and development phase during the Space Engineering Detailed Design Project, followed by verification and testing in the subsequent quarter. Improving upon the work of the previous year, we focused on advancing subsystem-level integration by verifying the functionality of the implemented software. In particular, this concerned the algorithms used to detumble the spacecraft after release into orbit and to orient its instrument towards Earth during operation. Additionally, thorough work has been invested in integrating the hardware and software interfaces with one another, in order to achieve a data pipeline that takes in real-world data and provides appropriate commands for spacecraft actuation.

In this document, the entire process regarding the testing and verification of 3 specific requirements can be found, which were written during detailed design and derived from high-level requirements into technical, verifiable requirements. In logical order, the design of the test article and test jig are first shown. This is followed by the software design, after which the experimental configuration and test setup are presented and elaborated on. Specifically, the documentation concerns a modular, convertible test setup with which a rotating breadboard model of the spacecraft is used to verify detumbling and nadir pointing which were developed during detailed design (AE4904). The setup captures real-world sensor data from Earth Horizon Sensors, represented by IR sensors, and uses magnetometer measurements of the changing local magnetic field to support the verification of the spacecraft's attitude determination and control functionality. Other, relevant requirements are also listed and explained in relation to the experimental design, including what data is to be acquired and processed further on. Finally, the test results can be found in the final quarter of the document, together with the conclusions that can be drawn based on these results.

2

Test Article

In this project, the breadboard model (BBM) is the article being tested in two configurations: one enabling the rotation of the BBM around its long axis, while the other being for the short axis. To this end, the setup in this chapter is also configurable, featuring various rates of rotation and orientation using a stepper motor, which is remotely run with a Moteino from a laptop using Arduino IDE. Additionally, the test article contains all Attitude Determination and Control System (ADCS) components and its associated Data Acquisition (DAQ) architecture to enable subsystem-level testing which concerns spacecraft attitude control.

In this chapter, we will first present the design of the test article and the test jig in section 2.1. It contains the hardware and software design, and an overview of all featured components and their relations in the planned experiment. Continuing with section 2.2, the two test configurations are presented, including a detailed assembly guide for both configurations concerning the test article and the test jig. Finally, the test setup is described in section 2.3, covering the operational overview of the full system during testing.

2.1. Design

This section delves into the design of the test article and test-jig in more detail, providing a model for the test article, a detailed electrical schematic, it's associated DAQ architecture, and the test-jig assembly.

2.1.1. Test article

Due to the extensive development work completed during detailed design, the practical phase was limited to testing and verifying the functional requirements of the ADCS. Performance verification was excluded due to time constraints. The scope was therefore defined as follows:

- Verify only the functional behavior of the ADCS.
- Integrate the intended subsystem components, including 1x Nucleo-L476RG, 4x MLX90640 IR sensors, 1x RM3100 magnetometer, and 3x DRV883x H-bridge drivers.
- Include the 3x magnetorquer (MTQ) actuators representatively to assess operational impact.
- Use a Moteino-USB board for wireless DAQ and switching modes of operation.
- Test one axis at a time to avoid the complexity of full 3D rotation.
- Use an MPU6050 gyroscope to compare measured rotation rates and angles.
- Store timestamped data on the STM32's 1 MB flash memory for post-processing.

With these in mind, it was decided to build a single breadboard model which allows the integration of the various components on a subsystem level to simulate spacecraft behavior in its operational environment. In Figure 2.1, all test article components can be found, with the pin connection tables Table 2.1 and Table 2.2 for the Nucleo and Moteino respectively. Furthermore, the final design of the BBM can be found in Figure 2.2, which shows the final assembly and the location of each component, and by what they are held together by.

Displayed below, the full electrical component inventory of the ADCS can be found. Each entry is labeled with a letter and number:

- **N1 NUCLEO-L476RG:**

This is the centerpiece of the ADCS; an evaluation board for the STM32L476RG microcontroller. It is connected to the voltage regulator (C1) which supplies 3.3 V through the corresponding 3V3 pin. A table of all of its pin connections is given in Table 2.1.

- **B1 Battery:**

For supplying 6.6[V] the test article during testing, a small, accessible battery (2S LiFePo from HobbyKing[3]) was used. It is important to emphasize that it is not meant as flight hardware.

- **C1 Voltage regulator:**

Since B1 supplies 6.6V, and all components of the ADCS were designed to work with 3.3V, a buck converter is needed. An inexpensive option that fulfills the voltage and current requirements is the 3.3V 1.5A DC-DC converter from Tinytronics[15].

- **M1-M4 MLX90640 IR Sensors (4x):**

These are the sensors used for finding the horizon of the Earth and subsequently finding the Nadir direction. They are connected to N1 via two separate Inter-Integrated Circuit (I2C) connections, I2C1 (M3, M4) and I2C2 (M1, M2).

- **G1 RM3100 Magnetometer:**

This is the 3-axis magnetometer that is used to help find tumbling states and yaw direction of the velocity vector. Although the RM3100 supports both I2C and Serial Peripheral Interface (SPI), it was designed for I2C, which is a simpler protocol to implement. For this reason, it is connected via I2C1.

- **D1-D3 DRV8838 DC motor drivers (3x):**

These are drivers typically used for DC motors, in this setup, their H-bridges are used to switch the direction of the MTQs. In this configuration, they are connected using three pins: nSLEEP (not sleep), PH (phase) and EN (enable). Since only slow switching of these lines is expected, they will be connected to GPIO ports on the Nucleo [13][12].

- **MTQ-X,-Y,-Z Magnetorquers (3x):**

Here, all the MTQs are aircoils and are connected to a driver each. While the MTQ-Z is a representative engineering model, the MTQ-X and MTQ-Y are only electrical placeholders for their Printed Circuit Board (PCB) variant. These are present to observe their impact during operation (e.g. power consumption). Overall, they are present to characterize the behavior of the system and to check functionality.

- **P1 MPU6050 Accel/Gyro:**

The MPU6050 is a common choice for an accelerometer/gyroscope set. It was planned to be used to independently measure the rotational rate of the setup in detumbling mode and the actuation angle in pointing mode. However, due to time constraints it was not utilized in the final test article. Nevertheless, it is recommended to integrate it in next year's project. It is inexpensively available on Tinytronics[16].

- **T1 Moteino:**

For the test setup, the Moteino emulates the On-Board Computer (OBC) by sending commands to, and receiving system data from the ADCS microcontroller. It is connected to N1 through I2C3 in master mode and serves three main purposes. First, it emulates the OBC by sending commands to, and receiving system data from, the ADCS microcontroller. Second, it independently measures the magnetorquer voltages to verify the STM32 outputs. It also communicates with P1, the MPU6050, to obtain independent angle and rotation-rate measurements. Finally, it wirelessly relays the collected data in real time to another Moteino connected to a laptop, enabling faster data collection and processing. The pinout of T1 is shown in Table 2.2.

On the next page, the full electrical layout of these components can be found. Each connection is based on the I2C communication protocol. Importantly, the IR sensors and magnetometer are connected to the STM32 through three I2C channels while the moteino is placed on the same channel with the STM32 and MPU6050 as the master for both. Finally, each driver connects to the Moteino through analog pins.

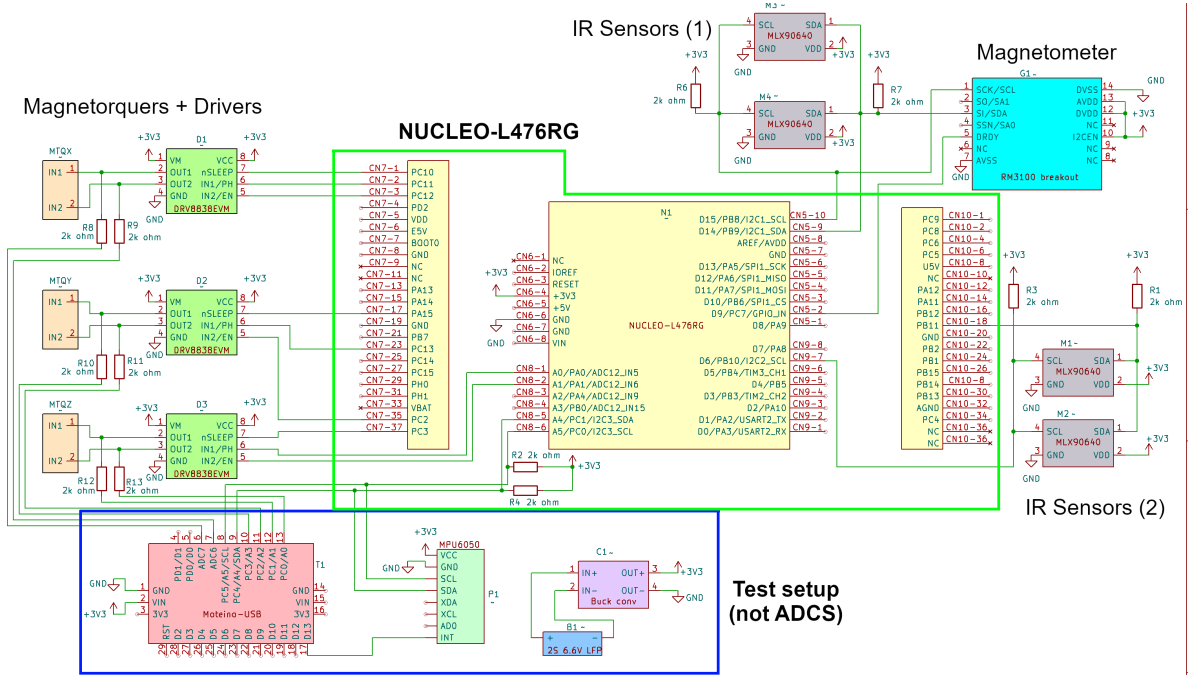


Figure 2.1: Electrical schematic of the test article, with the test setup (DAQ), including the Moteino and test battery, in the blue rectangle

Pin	Function	Connects to
PC10	GPIO_OUT	D1 nSLEEP
PC11	GPIO_OUT	D1 PH
PC12	GPIO_OUT	D1 EN
PA15	GPIO_OUT	D2 nSLEEP
PC13	GPIO_OUT	D2 PH
PC2	GPIO_OUT	D2 EN
PC3	GPIO_OUT	D3 nSLEEP
PA0	GPIO_OUT	D3 PH
PA1	GPIO_OUT	D3 EN
PC0	I2C3_SCL	T1 SCL
PC1	I2C3_SDA	T1 SDA
PC7	GPIO_IN	G1 DRDY
PB8	I2C1_SCL	G1, M3, M4 SCL
PB9	I2C1_SDA	G1, M3, M4 SDA
PB10	I2C2_SCL	M1, M2 SCL
PB11	I2C2_SDA	M1, M2 SDA

Table 2.1: Pin connection table for the Nucleo-L476RG in the BBM.

Pin	Function	Connects to
ADC7	ADC	D1 OUT1
ADC6	ADC	D1 OUT2
PC5	I2C3_SCL	N1 PC1, P1 SCL
PC4	I2C3_SDA	N1 PC0, P1 SDA
A3	ADC	D2 OUT1
A2	ADC	D2 OUT2
A1	ADC	D3 OUT1
A0	ADC	D3 OUT2
D13	GPIO_IN	P1 INT

Table 2.2: Pin connection table for the Moteino-USB in the test setup.

During the BBM's design, the model's philosophy was to produce a platform that could be tempered with iteratively through the course of the quarter. Additionally, whatever was put together, should be able to be taken apart with relative ease considering a high level of uncertainty in various tasks. In response, a modular design emerged which featured rods that skew through different modules in a tight fit, while they hold the components in place in the interior of the structure (see Figure 2.2) facing a rotation rate up to $180[^\circ/s]$. Additionally, as detumbling and nadir-pointing was to be tested, two configurations were needed which led in a model that can be attached to the test-jig to rotate the short- and long-axis of the test article. To provide ease of access to the cables, the wiring is managed externally. Lastly, all magnetorquer cores were halved for the printing process to be put together later.

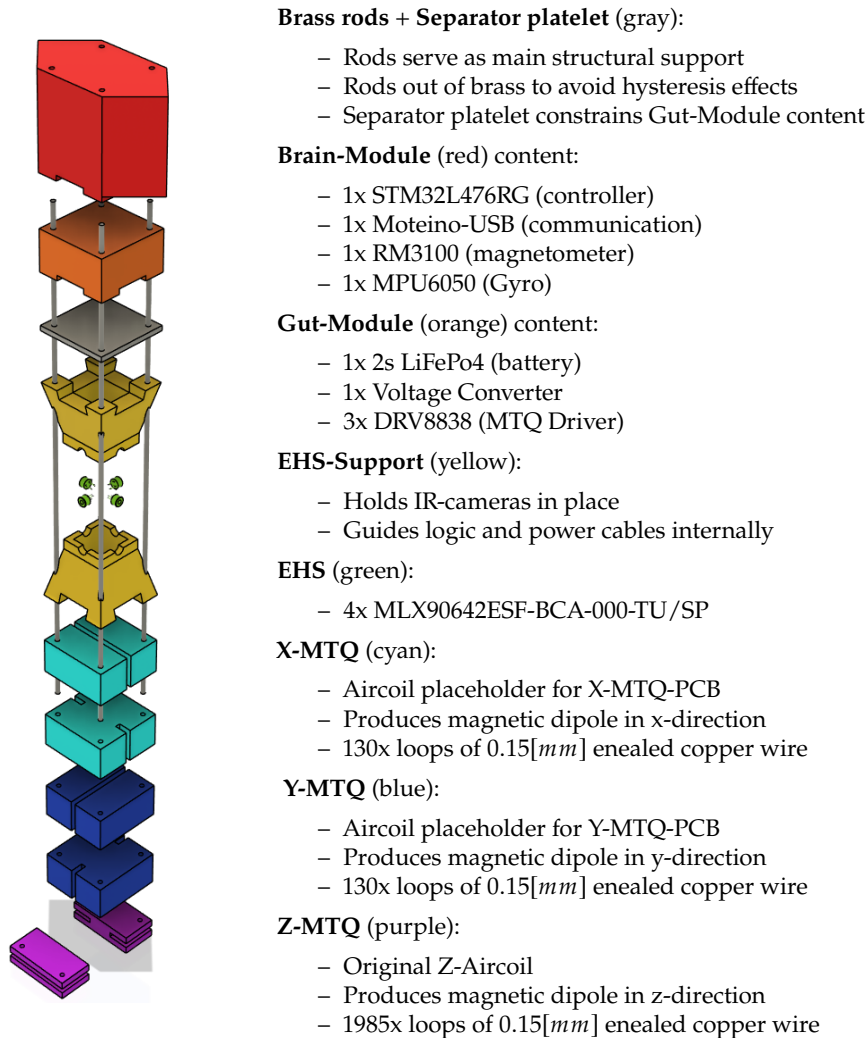


Figure 2.2: Render of the breadboard model and its component ordering.

For testing and verifying the functionality of the detumbling and pointing algorithms, it was required to extract real sensor data. To be able to build a data pipeline that pours the necessary values onto an excel sheet for evaluation, various digital, as well as physical interfaces needed to be tested and verified. For this purpose, the BBM also served to verify the integration of the various components on a subsystem level. With this, we could simulate the spacecraft behavior and its operational environment with a test jig and an horizon simulant, elaborated in the next sections.

2.1.2. Test jig

For the intended verification efforts regarding the functionality of the ADCS, the BBM had to be supported by a test jig that constrained five degrees of freedom, allowing rotation about only one axis. For detumbling tests, the test article must rotate about its short axis, whereas nadir-pointing tests require rotation about its long axis. In addition to supporting the BBM, the test jig was designed to hold a Nema 17 stepper motor,[14] which enables both constant rotation and precise orientation of the model's x-axis. In this setup, the stepper motor was controlled by a Moteino and driven by a DRV8825 driver [11] from tinytronics. To protect the driver from current spikes from the Tenma 72-2645 Programmable DC Power Supply, a 100 μF capacitor was used.

Test Setup Electrical Layout

To achieve the necessary operation during testing, the following electrical connections need to be established. This also dictates the design of the test-jig. Here, only the Nema17 needs to be placed in a specific manner to drive the rotation of the test article. All other components can be placed adjacent to the testing setup, with a (electrical) breadboard if necessary.

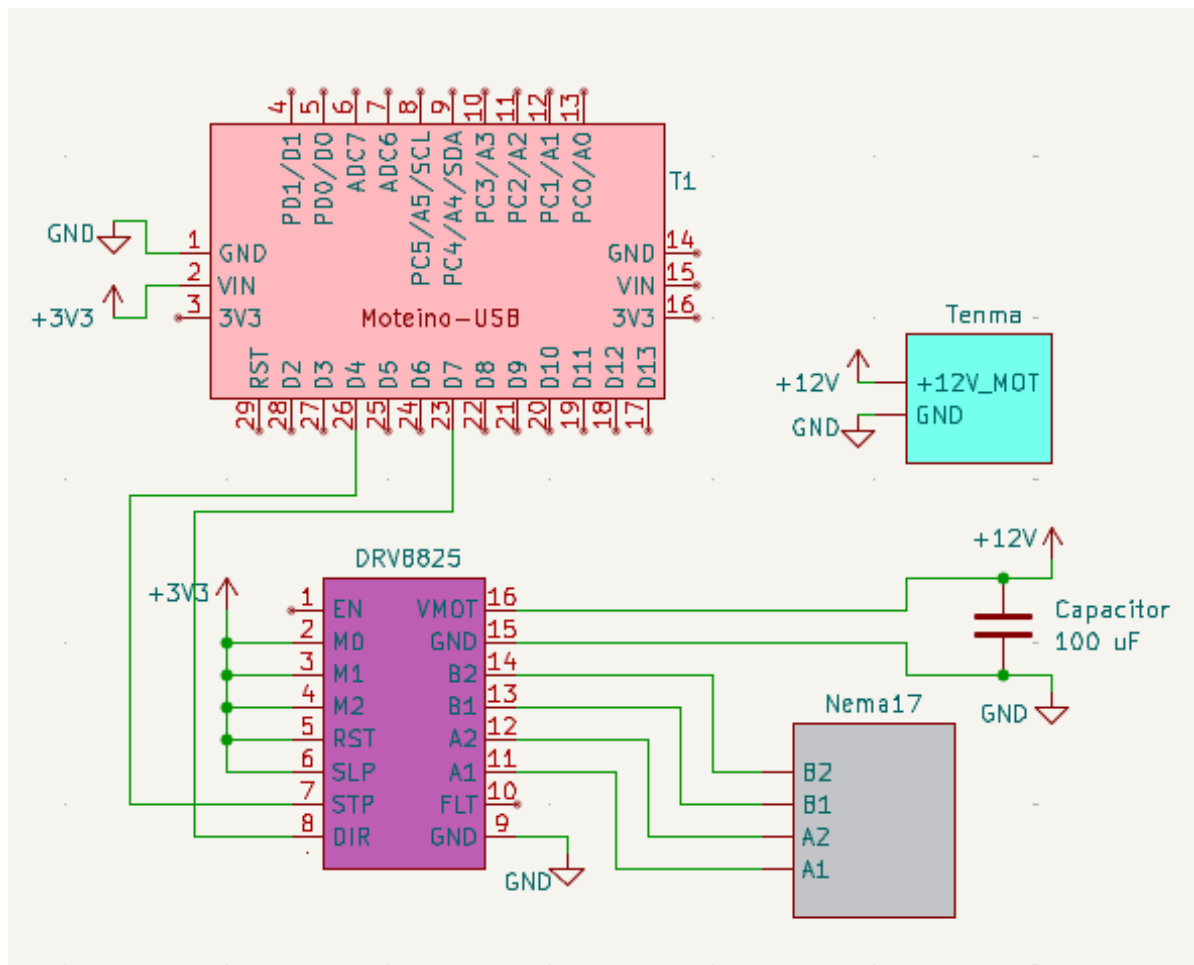


Figure 2.3: Electrical Layout of the Test Jig to run and control a stepper motor

Test Setup Model Design

To support the test article, and enable the test jig’s operation, 3x support stands, 1x stepper stand (Figure 2.4a), 2x Stepper-Rod Connector pieces and 4x hollow aluminum rods of 50 cm length are in use. While the supporting 3D printed structures holds the breadboard in place regardless of configuration, for each testing mode, the BBM needs to be reconfigured; replacing the connector test jig rods from one type to the other. Here, for each configuration, each final rod is glued together with their adjacent parts as displayed in Figure 2.4c and Figure 2.4d. Most importantly, the stepper motor is led into the connector fit as in Figure 2.4b without any permanent connections. The weight of the stepper motor in this case is high enough to rotate the test article around its axis of rotation, while the Nema17 sits in it’s support stand.

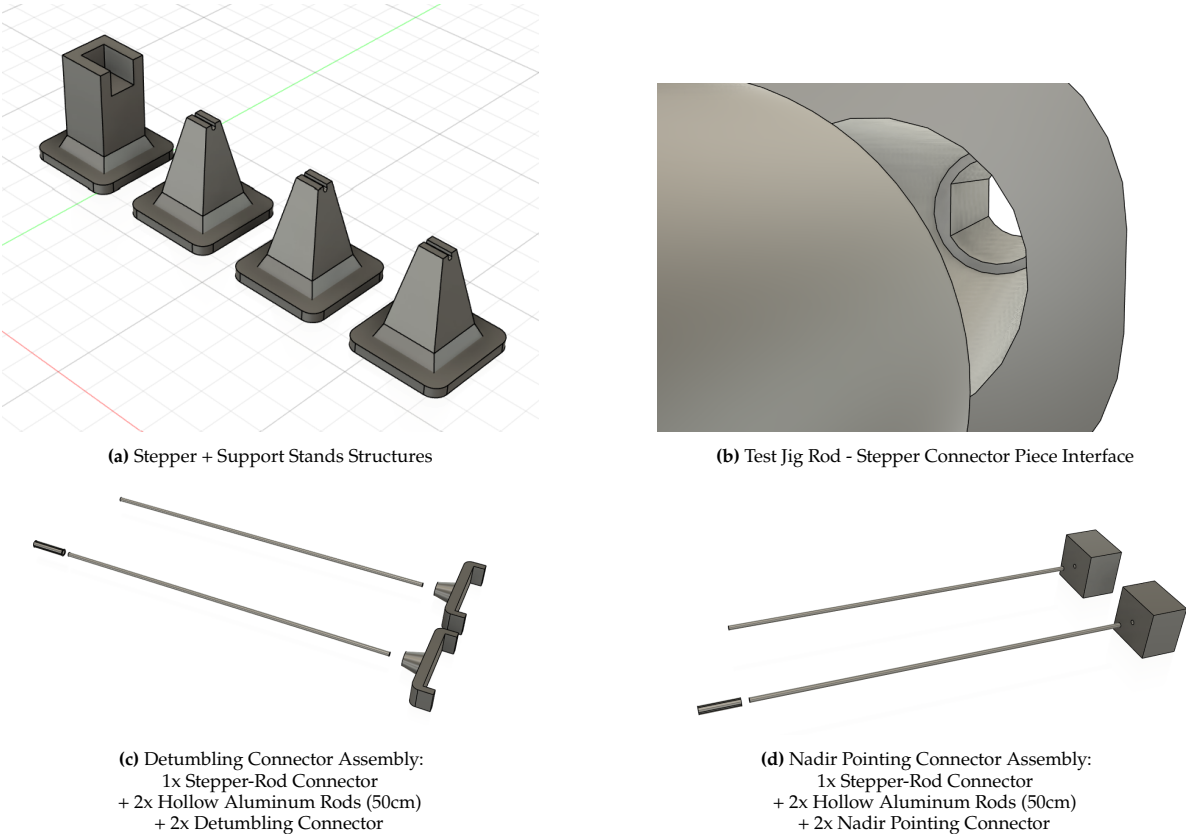


Figure 2.4: Overview of the support and connector assemblies used in the test-jig assembly.

The following tables contain all the required parts to assemble the test article.

Component	Specification	Quantity
Microcontroller	STM32L476RG	1
Magnetometer	RM3100	1
Voltage Regulator	From Tinytronics	1
MPU	MPU6050	1
Communication devices	Moteino-USB	2
Stepper motor	28byj-48	1
Motor driver	ULN2003	1
Lab power supply	Tenma 72-2645	1
Battery	2s LiFePo4	1
USB cables	Type A/B	3
Wiring cables	Dupont jumper cables	>30
Resistors	NIKKOHRNP-10SAR500JZ03-1	18
Multimeter	FLUKE175	1

Table 2.3: List of all electrical components needed for the dry run

Component	Description	Quantity
Jig support	Support for the rotating axis of the test model	4
Gut-Module		1
Brain-Module		1
Brass rods	Non conductive rods to connect all the casings	4
Aluminum rod	Non conductive rods to connect test model to stepper motor	2

Table 2.4: List of all structural components needed for the dry run

2.1.3. Horizon simulant

In the Nadir Pointing Tests a horizon simulant will be present as the attitude determination is mainly governed by the readings of the earth horizon sensor system consisting of a IR sensor on each lateral face and its algorithm.

Follow the legacy of a Space Engineering Practical (SEP) 2025 group[10], the horizon simulant is constructed from an aluminum sheet. Selected primarily for its high thermal conductivity, it will be heated to uniformly distribute the heat across its surface. Furthermore, aluminum is non-ferromagnetic, ensuring that the material does not interact with the magnetorquers, thereby preventing magnetic interference with magnetometer measurements.

The front of the sheet was coated with a high-emissivity IR paint of well-documented properties. Although designed to withstand extreme temperatures, the paint is expected to reach a maximum of 48°C during this test, yielding an emissivity of 0.92[8].

The backside features 18 $0.5\ \Omega$ resistors connected in series and secured using thermal paste and tape. While embedding the resistors directly into the plate would have been a more elegant solution, ensuring a larger contact surface area and lower thermal losses, time constraints necessitated this rudimentary approach. These heating elements are distributed across the plate, with a higher concentration near the two edges to ensure the simulated horizon remains clearly distinguishable for the sensors. It is shown in the figure below:

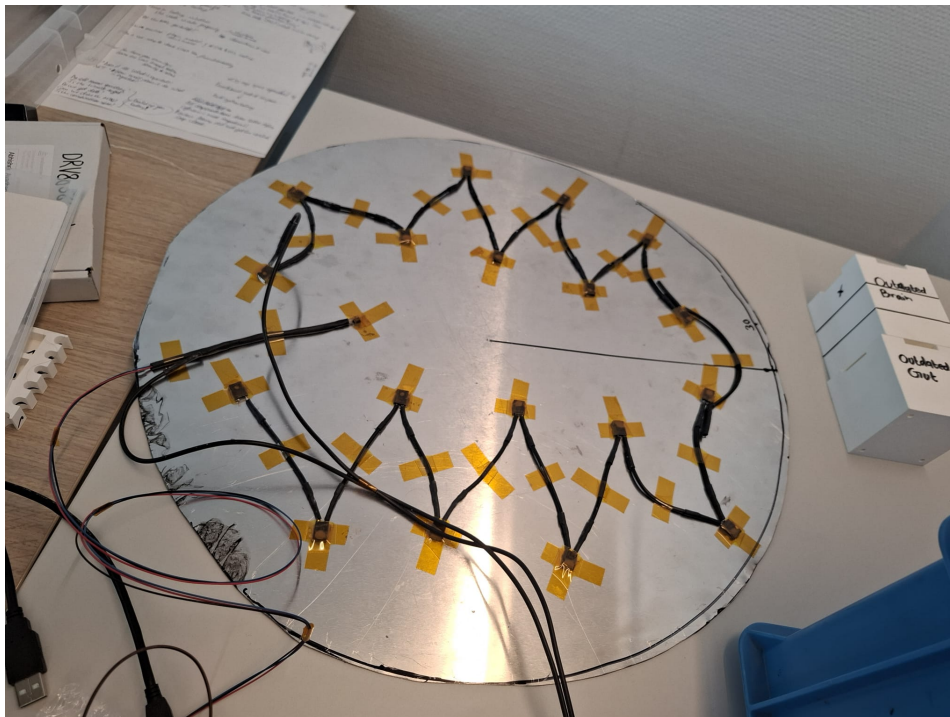


Figure 2.5: Backside of the horizon simulant with 18 resistors in parallel and a temperature sensor attached.

The setup utilizes NIKKOHMRNP-10SAR500JZ03-1 resistors, which are rated for 20 W and were previously validated by last year's SEP team[10]. They are wired using H07V-K cables for robust thermal insulation. The system connects to an Arduino via an RS232 interface, utilizing a MAX3232 chip to handle TTL to RS232 conversion. Finally, the Arduino runs a PID loop, developed by the previous group, which regulates the temperature to the desired setpoint, rounded to three decimal places. The only change that was implemented is that the start/stop button is now a spacebar input from the laptop. The electrical diagram is provided below where this distinction is still separated. Under that is the photo of the Moteino and its outputs and inputs for reference.

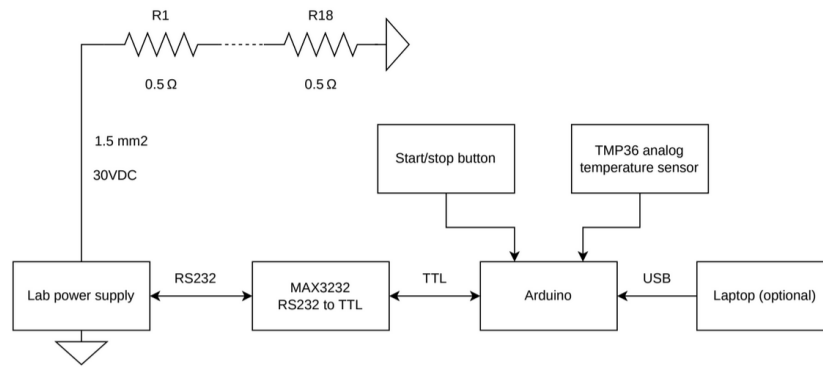


Figure 2.6: Electrical diagram of the setup used to heat the model earth. **Note:** the start/stop button is a digital spacebar input from the laptop.

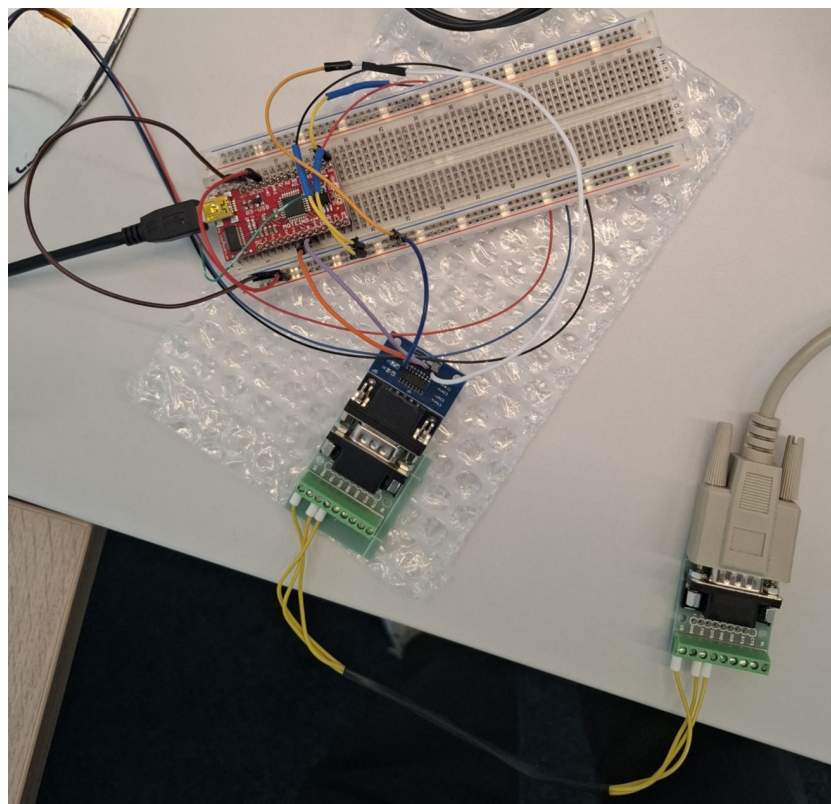


Figure 2.7: Photo of Moteino connected via USB to laptop, via TTL to RS232 to lab power supply and jumper cables to analog temperature sensor.

Distance determination

The sheet was cut into a circle as the 2D projection of a sphere, the Earth, is a flat circle. The radius was determined by simplifying the problem to a 2D plane as shown below, it is reiterated here from last year's SEP[10].

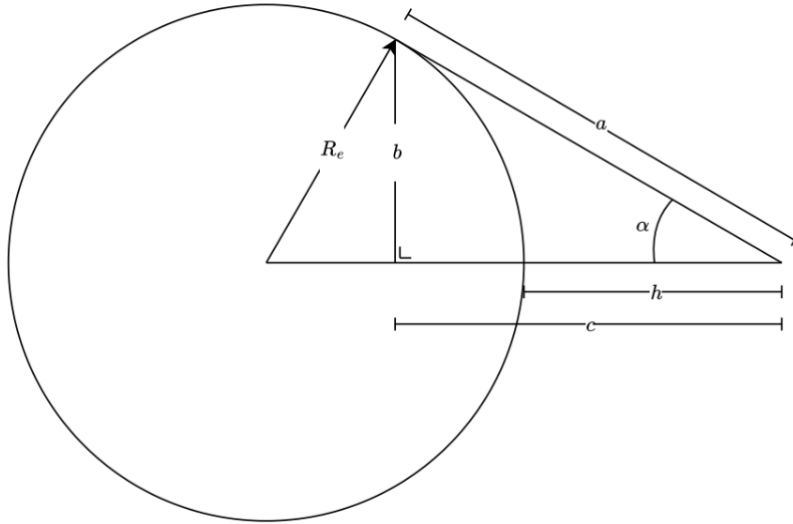


Figure 2.8: Satellite-Earth Model

The most likely altitude the satellite will orbit at is 450 km, as provided by our tutor J. Bouwmeester. With the altitude and Earth's radius, $R_e = 6378.137$ km, it is possible to derive all the other values seen in Figure 2.8. All the variables in the scaled down horizon simulant set-up are denoted with a c in their subscript.

As the dimensions of the aluminium sheet were $1000 \text{ mm} \times 500 \text{ mm} \times 1 \text{ mm}$, the maximum radius that was achievable was 250 mm. From this the scale factor could be determined, SF , via Equation 2.1.

$$SF = \frac{c_s}{c} = \frac{b_s}{b} \quad (2.1)$$

For this the distance between satellite (sensor) and the 2D projection of the Earth needs to be determined. Using Figure 2.8 the following relations are obtained:

$$c = a \cos(\alpha) \quad (2.2)$$

Where

$$a = (R_e + h) \cos(\alpha) \quad (2.3)$$

and

$$\alpha = \arcsin\left(\frac{R_e}{R_e + h}\right) \quad (2.4)$$

and

$$b = a \sin(\alpha) \quad (2.5)$$

So $b = 870.3433015$ km and $SF = 1.09787269 \times 10^{-7}$. Lastly, the distance between the sensor and the horizon simulant, c_s , can be determined either via the scale factor or the relation below:

$$b_s = c_s \tan\left[\arcsin\left(\frac{R_e}{R_e + h}\right)\right] = b \cdot SF = 0.095552 \text{ m} = 9.56 \text{ cm} \quad (2.6)$$

There is, however, an inaccuracy when scaling down the scenario. The relative distance between the sensors are not able to shrink down to the appropriate distances as dividing a distance of 2 cm by SF would result in the sensors occupying the same space. It was chosen to keep these distances as they would be on the Delfi-Twin to check the robustness of sensor switching. However, this causes a 2 cm difference between the furthest and nearest distances from the sensor to the simulant. This results in

significant perceived altitude differences due to the massive SF. To mitigate this, the largest possible radius was chosen, 250 mm, so that the largest possible distance between the sensor and simulant can be achieved, 95.52 mm.

For this reason, the c_s chosen was 1 cm less than the actual c_s for 450 km altitude which results in some readings occurring at 85.52 mm and some at 105.52 mm. These distances, when converted back to the real-life scenario results in perceived altitudes of 363 km and 545 km respectively. This, will likely warp the estimated pitch and roll estimates thus a correction has to be made in the data analysis.

2.1.4. Software design

This subsection will give a brief explanation of the implemented software design, its architecture, and recommendations for future developers.

Introduction

The software has to run on the existing hardware: an STM32. Its main function is to take inputs from the two types of sensors and transform them into magnetorquer outputs. In the meantime, it should be able to provide data to a laptop for analysis and performance characterization.

This report mostly concerns itself with the practical implementation of the software design that was already presented in the previous report, the Space Engineering Detailed Design Project (SEDDP)[9]. For more information on the design of the software, the reader is directed to read that report.

Interfaces and responsibilities

The software is defined by what it has to do and what devices it must talk with. The following are the devices it must interface with:

- One RM3100 magnetometer
- Four MLX90642 Infrared (IR) sensors
- The three DRV8837 drivers
- One Moteino

The tasks it has to do with each peripheral are:

- RM3100: The software has to be able to read data from the sensor's registers and store it into its memory. The RM3100 has the ability to work with either I2C or SPI.
- MLX90642: The software has to be able to read data from each of the four sensors. This data has to be then be processed. The sensors work only on I2C.
- DRV8837: The software has to be able to send binary signals to three of the chip's pins in order to control it effectively. This is not high frequency and can be done through GPIO.
- Moteino: The software has to be able to send telemetry to the Arduino-based microcontroller. This includes sensor readings, timestamps, and driver outputs. The Moteino can also act as a simulated OBC which is gathering telemetry and could be sending commands.

The main responsibilities that the software must accomplish are:

- Store RM3100 readings and convert them into a useful input for the B-dot controller.
- Implement the B-dot controller.
- Store MLX90642 readings and convert them into a useful input for the Nadir pointing controller.
- Implement the Nadir pointing controller.
- Provide telemetry to a channel that the Moteino can access.
- Convert the controllers' outputs into signals for the drivers.

Some of these tasks are to be done in sequence, whereas others are to be done in parallel. The logical sequence of events is Sensing, Applying the control algorithms, Driving the MTQs. The Sensing event can have all sensors working in parallel. In order to be able to provide telemetry at any point, data logging must be done in parallel with everything else.

Architecture

The significant amount of parallelization within the tasks of the software led to an investigation into the feasibility of using a Real-Time Operating System (RTOS) for the ADCS. Such a system would allow multiple tasks to happen effectively at the same time, foregoing the need to implement synchronization within one task.

Trial runs showed that an implementation of FreeRTOS was not only possible but also practical. While it cannot entirely remove the issue of timing, the software is able to do multiple things at once with less difficulty. Furthermore, it provided a great opportunity to learn and practice more advanced programming.

The software was mainly divided into two groups: drivers and tasks. The drivers are responsible for communication with the peripherals. They are also responsible for doing some data pre-processing. For example the MLX90642 driver is split between the MLX90642 driver itself and the horizon process which takes raw data frames from the sensors and processes them into pitch and roll readings. The second group is the tasks. They are mainly responsible for data processing and output generation. Because of the deterministic nature of data processing, one FreeRTOS Task is responsible for the data processing, whereas another Task generates the driver outputs.

The architecture of the software can be seen in Figure 2.9. This diagram shows the relationship between tasks and how data is in general allowed to flow. It also shows some of the constraints that the tasks have between each other to ensure correct timing.

As described in the SEDDP report[9], the tasks are orchestrated to work as one loop. The equivalence is shown in Figure 2.10. Each loop is divided into two halves: the first half deals with sensing, processing and the preparation of MTQ actuation. The second half produces the actuation that generates the required torque for the satellite.

Implementation

RM3100 Driver The RM3100 Driver is contained in the `RM3100.h` file. The file itself is a C port of the `magneto.hpp` driver written by the 2024 *Detumbling the PQ* SEP Group [1]. The main function within this driver is `pollAndReadMagField()`, which uses some basic read/write functions to obtain raw data from the RM3100. It then converts the raw data into floating-point numbers and sends them to its output.

Note that the RM3100 Driver contains a data type called `Magnetometer3axis`. In the original C++ code, this served as an easy way to store multiple readings and their time stamps. However, this function was not used in the new code and the data type is vestigial, along with most of its associated functions.

MLX90642 Driver The MLX90642 driver is a near-copy of the publicly-available driver code from Melexis[7]. The only difference is the implementation of the I2C read/write functions `MLX90642_I2CRead()` and `MLX90642_I2CWrite()` within `MLX90642_I2C_Driver.c`.

Horizon processor The Horizon processor consists of a multitude of files grouped together in the `Drivers/MLX90642` folder. The driver uses the MLX90642 Driver for obtaining the frame data and uses its own functions to process it into pitch and roll readings. It does so by converting the images into a mask which it then compares with a pre-flashed lookup table. More information about the processing is available in the SEDDP[9].¹

Moteino Driver The Moteino Driver is mainly concerned about the maintenance and delivery of the I2C Register to the I2C3 channel. It declares the memory mapping of the register, redefines some I2C3 Hardware Abstraction Layer (HAL) functions, and provides some utility functions to work with the register.

RM3100Reader The RM3100Reader task is the FreeRTOS task which makes use of the RM3100 Driver. After creating a new `Magnetometer3axis` instance and setting up the RM3100, it waits until it can acquire the magnetic Mutually-Exclusive (mutex). Once it is acquired, the task then takes three magnetometer

¹Changelog: implemented Direct Memory Access (DMA); various sanity checks to improve performance; added 4 sensors; set threshold temperature to 38 degrees; increased grace window; consensus check instead of weighting.

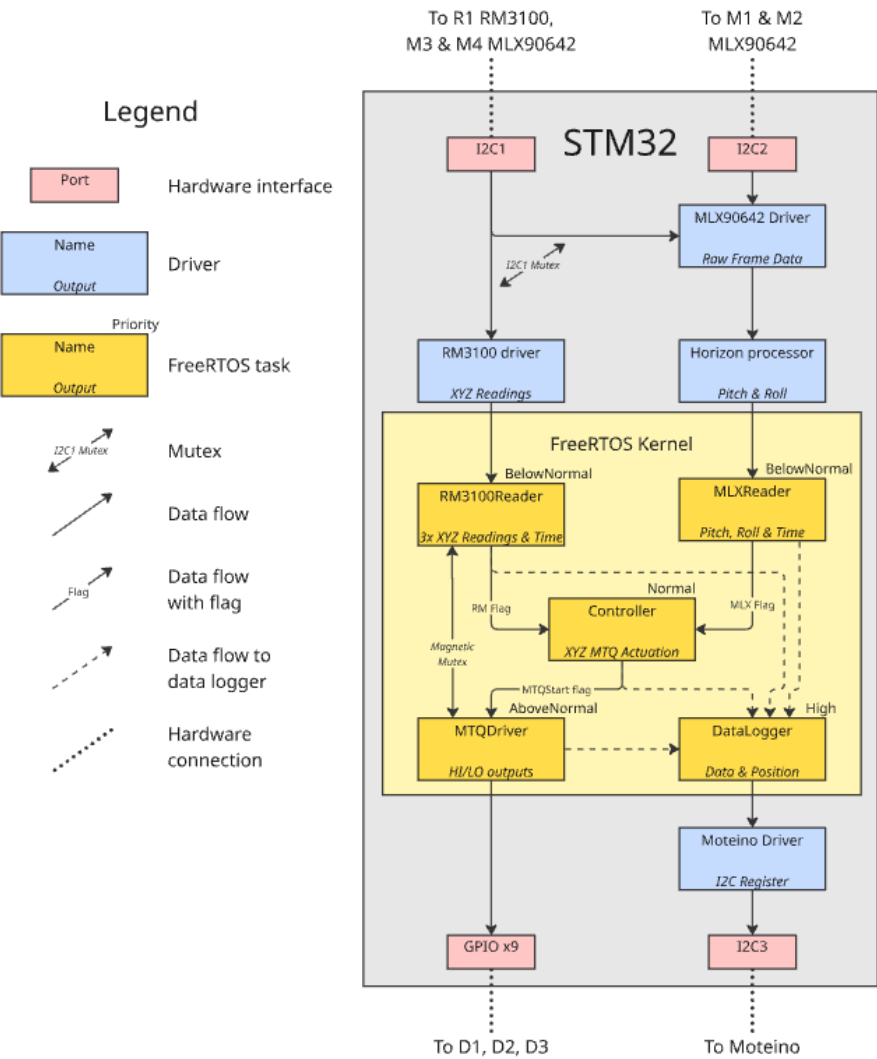


Figure 2.9: Software architecture

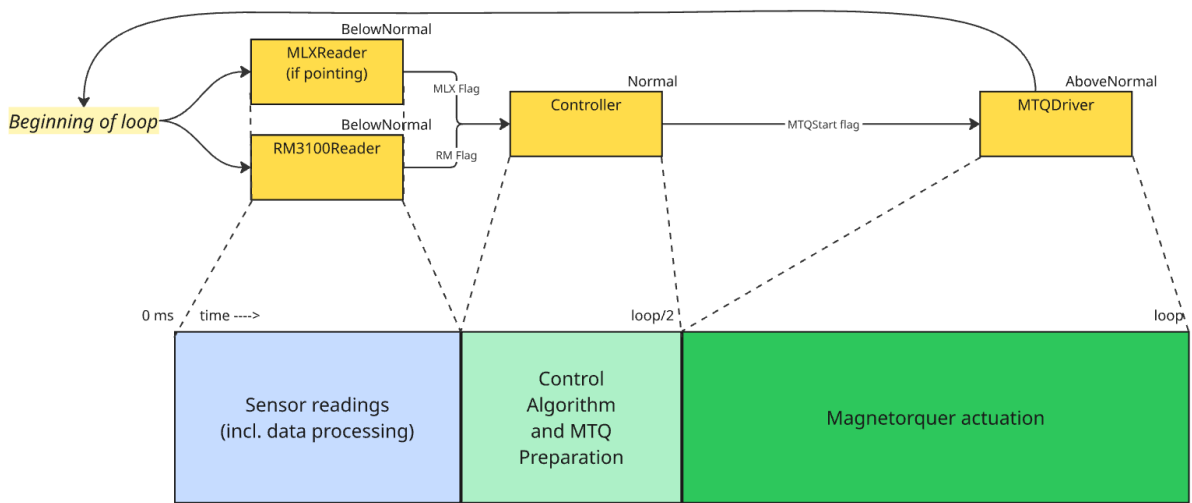


Figure 2.10: Timing schematic of the software architecture and how it implements the loop timing from the SEDDP[9]

readings spaced out by a duration depending on the current operational mode of the system. Each reading receives a timestamp and is sent to the Controller task. Finally, the RM3100Reader sets its corresponding bit of the `ControllerStart` event flag to true, signaling to the Controller task that the magnetometer readings are ready to be processed.

MLXReader The MLXReader task is the FreeRTOS task that makes use of the Horizon processor driver. After initializing the IR sensors it waits for its turn in the task loop. Once per loop it calls the Horizon processor to generate a roll and pitch reading, to which it attaches a timestamp and sends it to the Controller task. At the end it sets its corresponding bit of the `ControllerStart` event flag to true.

Controller The Controller task is the FreeRTOS task that takes the readings from the RM3100Reader and MLXReader and applies the control algorithm developed in the SEDDP[9]. The initialization part of this task contains most of the configuration options for the flight software, including the control gains. Upon receiving both flags from RM3100Reader and MLXReader, it first processes the three magnetometer readings to its own body frame before applying a misalignment correction with a directional-cosine matrix. Next, the magnetic field array is filtered using an exponential moving average applied to the B-dot. The task next calculates the angular velocity based on the filtered magnetic field array, and decides its operational mode based on the angular velocity and a threshold.

If the operational mode is Pointing, it then performs the PD control algorithm using pitch and roll readings. The output of the control algorithm is a magnetic dipole that is stored for later. Next, the B-dot algorithm is also applied. The output of this algorithm is also a magnetic dipole, which is then combined with the PD algorithm's magnetic dipole to generate a total dipole. This total dipole, if it too large, is scaled down to the saturation dipole of the magnetorquers. Finally, it converts the dipole into actuation times for each MTQ scaled by half the loop length, and sends these to the MTQDriver task.

MTQDriver The MTQDriver is the FreeRTOS task that is responsible for creating the right digital signals for the DRV8837 drivers in order to generate the right magnetic dipole through the MTQs. After acquiring the magnetic mutex, it takes the actuation times calculated by the Controller task and centers them around the middle of its actuation period. This ensures that the change in angular momentum is as close as possible to the commanded one, which assumes instantaneous angular momentum changes. The task then executes the commanded MTQ actuation. At the end it releases the magnetic mutex and ends the loop.

DataLogger The DataLogger is the FreeRTOS task that handles data going to the I2C Register. With its high priority, it runs every time a task attempts to log a value. It sends the float byte by byte to the I2C Register in its designated place.

Unit testing

Unit tests were performed on code during development to verify functionality. As a result, software development was approached in steps. Each step represented a minor "milestone", either in the software itself or in the data pipeline. A full list of the documented unit tests can be found in Appendix C.

Unit tests were chosen in the direction of achieving functionality by the simplest path. First, communication with sensors was addressed, and then data communication between the different elements of the pipeline was established.

Important to mention is that due to time constraints not all code was unit tested.

Recommendations

As shown in section 8.2, the failure of the system to achieve its performance goals was, in part, due to errors in the code. This stems from two main aspects. First, the code was never fully verified line by line. It is likely that there are more errors hidden within the code. Second, the code was rushed to achieve functionality for the test, disregarding some of the system's less vital functions like checking its own health. Both aspects can be boiled down to time constraints. As a result, the foremost recommendation for future development is a thorough verification of the code. If feasible, implementation of the less essential functions can also be done.

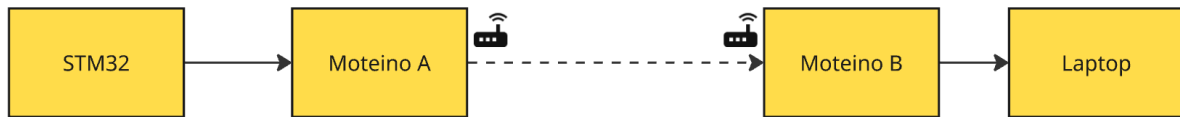


Figure 2.11: Simplified overview of the data pipeline

The below is a list of recommendations for future software development:

- A reassessment of the architecture of the software is advised if the future software developers are not familiar with FreeRTOS. While FreeRTOS has brought benefits, the loop-architecture of the software has minimized them. As it stands, an iterative main superloop might have performed the same as the RTOS orchestra currently implemented.
- Given that the RM3100 Driver is a direct C++ to C port from a previous year's implementation, a significant amount of code is vestigial. The current code can be vastly simplified to only a few functions, most importantly reading, writing, updating gains, polling and processing.
- The system currently has only two operational modes implemented: pointing and detumbling. Safe mode and hibernate are implemented partially or not at all. As it currently stands, they are ignored everywhere.
- A health checker function can be implemented to assess the connection status of the system.
- Not much attention was given to speed optimization. This has resulted in the code being very large and the loop always running over its intended length. For example, in pointing, while the loop length was intended to be 960 ms, the calculation step alone takes more than 1000 ms. This gives the MTQDriver no chance to send signals to the DRV8837s, instead immediately going to the next loop.
- Fail-safes have not been implemented in the software. The above issue, for example, could result in permanent loss of ADCS functionality. The implementation of fail-safes and other guards should be considered leading up to the implementation of flight worthy software.
- There are numerous to-do items left in the code to guide future developers about what is currently known to lack.
- The only way for the system to communicate with the outside world at the moment is a one-way channel with the Moteino where it simply sends its entire I2C Register from a starting position when requested. While this worked fine for the testing of the current report, the system should be able to take data and instructions from the Moteino too. In the future, this functionality can be ported to a (simulated) OBC.

2.1.5. Data Pipeline

Since the STM32 is meant to be placed on a rotating test article, data has to be transferred without a physical wired connection. Arduino clones with RF modules called Moteinos[6] were used for this purpose due to them being readily available. Two Moteinos were used: Moteino A was directly connected to the STM32 via its I2C3 connection, and Moteino B was connected to a laptop via its serial port. The two Moteinos communicated via RF through data packets. Figure 2.11 provides a simple overview of the data pipeline.

The codes of all devices are available in the zip file which should be attached to this report.

STM32

The pipeline begins with the STM32. In order for the STM32 to be able to send data over the I2C3 channel, a byte array called `I2C_REGISTER` was created in its global memory. Each number to be transmitted across was a float, which is 4 bytes. In order to split each number into bytes and store it accordingly, the following data type was used:

```
typedef union {
    float floatValue;
    uint8_t bin[sizeof(float)];
};
```

```
} floatOrBytes_t;
```

To make sure that each float stayed in its designated index within the array, a memory map was created, defining where each piece of telemetry goes.

The HAL I2C callback functions were also altered. When the STM32 hears its own address on I2C3, in receiving mode it stores what it receives, which for this project was simply the address in `I2C\_REGISTER` to start broadcasting from. In transmitting mode, it sends the entire `I2C\_REGISTER` from that requested address.

Moteino A

Moteino A is responsible for requesting and storing data from the STM32. Since the most difficult part of the pipeline was judged to be the RF communication, the final Moteino A (also called Sender) code was simply an expansion over an existing example code for sending RF packets. Only two main modifications were done to achieve functionality.

First, I2C communication was implemented to be able to request and store data from the STM32. Since the Arduino Wire library has a maximum buffer size of 32 bytes, the Moteino has to re-request the data from the STM32 each loop from where it lost track. This results in slightly inefficient communication, but the Moteino is able to receive all the data from the STM32's `I2C\_REGISTER` without loss of integrity.

Second, multiple RF packets are sent every loop. First, a header packet is sent containing how many subsequent data packets to expect. Then, a corresponding number of data packets is sent. This was done to avoid the hardware limitation imposed by the RFM69 module, which sets a buffer limit of 61 bytes.

With these modifications, the Moteino A is able to request, store, and relay the entire contents of the `I2C\_REGISTER` array from the STM32. Its loop takes ~300 ms to complete, which means that the Moteino A requests data from the STM32 and relays it to Moteino B every 300 ms. A side effect of this architecture is that in Pointing mode the STM32 does not have a chance to update its register due to its slower loop length and as a result the Moteino A sends three or four identical packets in series.

Moteino B

Similarly to Moteino A, Moteino B's (also called Receiver) code was based on an existing example code from LowPowerLab's library. Many of the modifications mentioned above were ported to Moteino B. However, unlike Moteino A, Moteino B's code does not contain any use of Wire, since it simply receives the RF packets, unpacks them, and sends them over the serial monitor for the laptop to process.

Laptop

Once the laptop receives Moteino B's packets over the serial monitor, a Python script was written to save each new line of telemetry to a Comma-Separated Values (CSV) file.

2.2. Configuration

For this project, there exists one test article that can be assembled in two configurations. The first for detumbling, where the long axis of the model is perpendicular to the axis of rotation to study system-behavior during detumbling, and for the second the long axis is collinear to the axis of rotation to simulate system-behavior during pointing. Both configurations are subject to multi-stage tests over time.

Table 2.5: Procured items

Item	Quantity	ID	Role
Nucleo-L476RG	2	N1, N2	N1: BBM, N2: SW development
RM3100 magnetometer	2	G1, G2	G1: BBM, G2: SW development
MLX90640 IR sensor (wide field)	5	M1–M5	M1–M4: BBM, M5: backup
DRV8838 motor driver	3	D1–D3	BBM
LiFePo4 battery (2S)	2	B1, B2	B1: BBM, B2: spare
Moteino-USB	3	T1–T3	T1: BBM, T2: receiver, T3: spare
Voltage regulator (3.3V)	1	C1	BBM
MPU6050	1	P1	BBM
NEMA-17 stepper motor	1	SM	Test jig
JST-XH 4P to JST-PH 6P cable	1	–	BBM wiring
100 μ F capacitors	5	–	Stepper driver protection
Dupont wires (10cm)	10	–	BBM wiring
Enameled copper wire (0.15mm)	500m	–	MTQ winding
Aluminium sheet	1	–	Horizon simulant
Breadboards	3	–	SW development
Instant glue	1	–	Assembly

Table 2.6: Manufactured items

Item	Quantity	ID	Role
Brass rods (50cm)	4	ROD1–ROD4	BBM structural support
Aluminium tubes (50cm)	2	ALU1, ALU2	Test jig support
Test jig stands (15cm)	3	SUP1–SUP3	Test jig support
Stepper motor stand (15cm)	1	SS	Test jig support
X/Y-MTQ aircoil (130 loops)	2	MTQ-X, MTQ-Y	Electrical representation
Z-MTQ aircoil (1985 loops)	1	MTQ-Z	Electrical representation
EHS holder	2	EHS-H1, EHS-H2	BBM structural support
Modular model lid	1	LID	BBM structural support
Gut-Module	1	GUT	BBM structural support
Brain-Module	1	BRN	BBM structural support
Nadir stepper-model connector	1	NSM	Test jig, nadir configuration
Nadir model-support connector	1	NMS	Test jig, nadir configuration
Detumbling stepper-model connector	1	DSM	Test jig, detumbling configuration
Detumbling model-support connector	1	DMS	Test jig, detumbling configuration

During the production process, Makerlab’s 3D printing facility at TU Delft was utilized. Importantly, mandatory training was required to gain access to the printers which usually needs to be booked a month in advance. Using generic, white PLA, all modules, connector pieces, adapters and supports were printed by Prusa MK3S+ printers with a 0.4 mm tip. For the infill, a gyroid structure with 10%

infill was chosen for its good strength and light weight. Finally, all halves for the magnetorquers were glued together into single modules with industrial glue.

2.2.1. Detumbling Test Setup

To set up a test for detumbling, two adjacent tables of equal height are required. While one table features one support and stepper stand (upon which the stepper motor rests), the other features two support stands. Connecting the breadboard model with the test-jig rods as in figure 2.7.a., the test article is placed upon the supports, and then led to connect the stepper motor with the adapter end of the test-jig rod as in figure 2.7.b.. Completing this step, the model should be suspending over the gap from the ground between the tables to allow free rotation as in figure 2.8.. No glue or tape is required as all necessary constraints hold the setup in place on their own. Lastly, to ensure the center of mass is located on the axis of rotation, the brain- and gut-module can be shifted along the model rods.

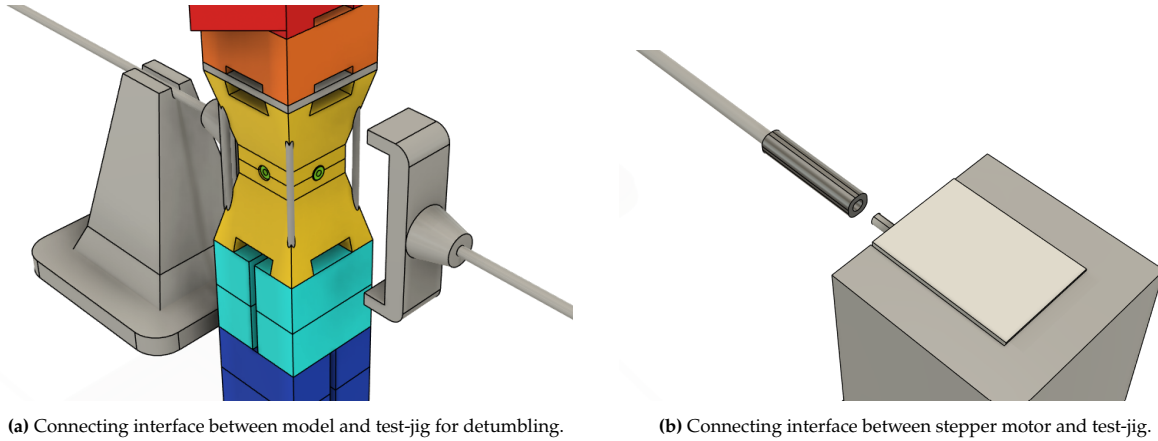


Figure 2.12: Mechanical connection interfaces used in the test-jig setup.

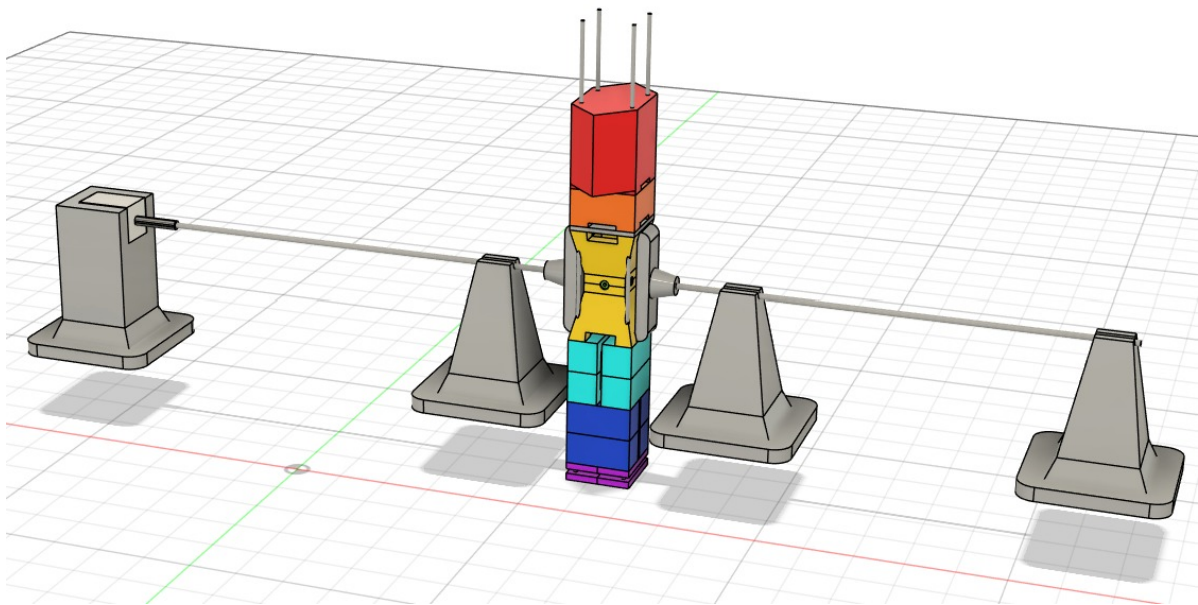


Figure 2.13: Breadboard model and test jig in the detumbling setup using a stepper motor (white). The available rod space above the Brain-/Gut-Module allows the center of mass to be positioned on the axis of rotation.

2.2.2. Nadir Pointing Test Setup

To set up an experiment for nadir pointing, the brass rods of the model are connected at both ends with centered adapter pieces as in Figure 2.9.. Following the same procedure as in the section to for the detumbling test setup, the stepper motor is connected to the test-jig rods. Finally, a horizon simulant is placed underneath the breadboard model for experimentation on nadir pointing, working with the four EHS.

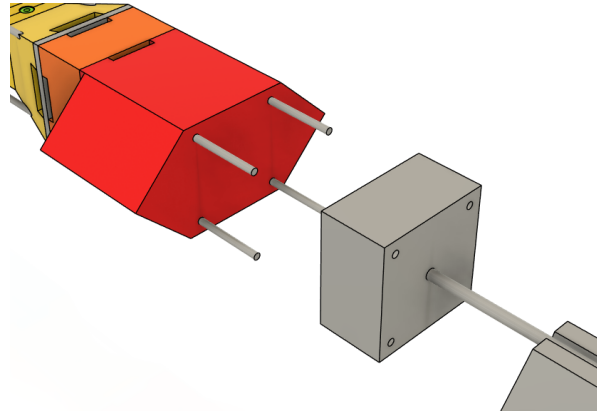


Figure 2.14: Connecting interface between model and test-jig for nadir pointing.

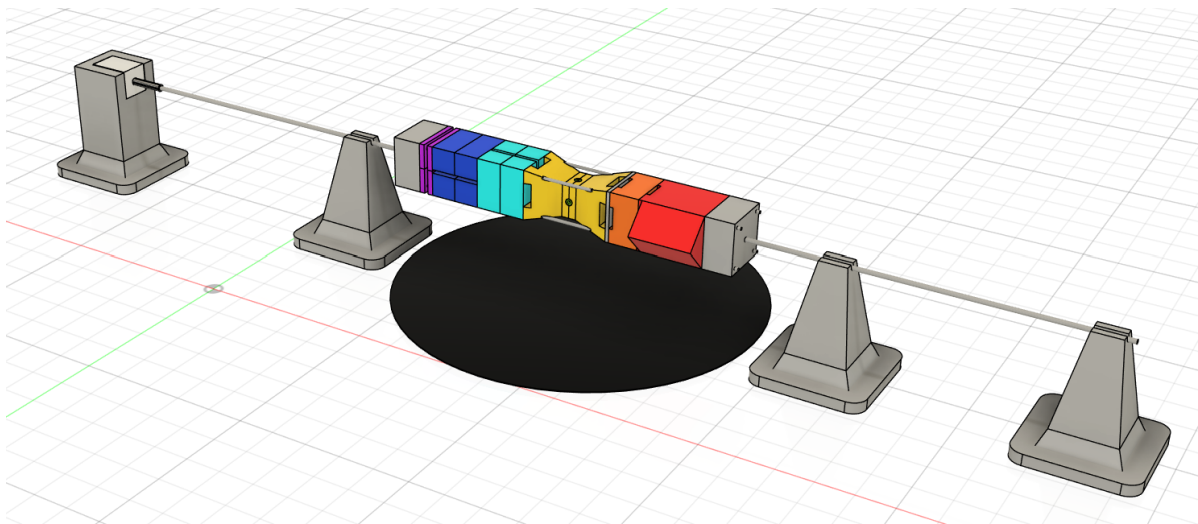


Figure 2.15: Breadboard model and test jig in the nadir-pointing setup using a stepper motor (white) with the horizon simulant placed underneath (black).

The Brain-Module is placed opposite the stepper motor to reduce magnetic interference with the magnetometer.

2.3. Test set-up

The purpose of the test setup is to verify the functional behavior of the ADCS BBM under controlled and repeatable conditions. The setup provides known rotational motion and a constant infrared horizon reference, allowing the implemented ADCS algorithms to be fed with real world sensor data to put out real-time actuation commands.

Two types of tests are performed with this setup:

- **De-tumbling tests:** for each run, the BBM is rotated at a constant, pre-fixed angular velocity using a stepper motor. Each data point is sampled continuously to gain insight on the magnetometer readings, to get an estimate on the rate of tumbling and read the commanded torque for detumbling.
- **Pointing tests:** the BBM is set to a pre-defined orientation using a servo motor while observing a fixed infrared horizon reference. At each data point, the stepper motor rotates by a pre-defined angle, then holds the test article in place to receive attitude estimates based on sensor output.

All relevant measurements, including sensor readings, reference motion data, and control outputs, are transmitted to a laptop and stored for post-processing and evaluation.

Table 2.7: Main elements of the experimental test setup

Element	Function
Breadboard model	ADCS hardware under test including the microcontroller, sensors, and magnetorquer driver electronics.
Test jig	Mechanical structure used to mount and rotate the breadboard model about a single body axis.
Reference measurement system	MPU6050 inertial measurement unit mounted on the breadboard model providing independent measurements of angular rate and orientation.
Horizon simulant	Heated plate providing an infrared reference for the Earth Horizon Sensors, attached using a circular railing to allow for simulated changes in pitch.
Data acquisition system (DAQ)	Moteino communication module transmitting sensor data (including the MPU-6050) and control outputs to the laptop.
Laptop	Monitors system health during the experiment and records outputs from DAQ via another Moteino connected with USB.
Motion generation system	Stepper motor and servo motor used to impose controlled angular velocity or fixed attitude on the breadboard model. Controlled by the laptop via its Moteino.
Power supply	While the DAQ is powered by the same LiFePo4 battery that the BBM uses, the motion generation system will be powered by a DC power supply connected to the mains.

2.3.1. Overview

The experimental setup is based on a BBM of the ADCS, mounted on a motorized test jig that allows rotation about one body-fixed axis at a time. For detumbling, the stepper motor rotates the BBM at a predefined constant angular velocity, while for nadir pointing, it holds the BBM at a defined angle relative to Nadir for Earth Horizon Sensor (EHS) measurements. A heated plate provides a fixed infrared horizon reference for the EHS during nadir pointing. As discussed in section 2.2, the single-axis setup keeps the experiment deterministic and avoids full three-axis motion. The stepper motor is placed away from the BBM to reduce its influence on magnetometer (MGM) measurements, while a reference MPU6050 (discarded during testing due to lack of time) provides independent attitude and angular-rate measurements with respect to gravity. During testing, angle estimates, reference motion data, control outputs, and timestamps are sent wirelessly to a laptop through the Moteino module and used to

evaluate the ADCS response under the imposed test conditions.

Test Article Attitude Simulator

By the use of a stepper motor, both test article attitude is simulated; predefined constant rotation around one long/short axis and the predefined orientation of the BBM with respect to a reference point. For each experiment, the sensor inputs and command outputs are evaluated by the DAQ architecture. Each test can be reconfigured into the other type as explained in section 2.2. This makes it possible to test the same ADCS hardware under different imposed motion conditions without changing the breadboard model itself.

Test Article Backup Attitude

To draw conclusions on the accuracy of the ADCS attitude determination algorithm, an additional gyroscope is present on the test article for comparison. The gyroscope in use concerns a MPU6050 inertial measurement unit to provide independent measurements concerning the BBM's attitude. In other words, it measures the angular velocity and orientation with respect to gravity. However, this was later discarded during the set up of the experiment due to a lack of time. A rough reference by the stepper motor was used instead.

Test Article Nadir Pointing Reference

For the pointing experiments, a heated, circular plate, representing the Earth Horizon, is placed in the middle of the test jig directly underneath the BBM. It provides a fixed infrared reference for the EHS and allows the EHS algorithm's functionality to be tested. The infrared contrast between the heated plate and the ambient room is used to approximate the contrast between Earth's infrared radiation and the space medium's coldness. This allows the nadir-pointing estimation under controlled conditions.

Final Experimental Setup

With all of these discussed, the two configurations were set up as shown in Figure 2.16 and Figure 2.17a. It must be noted that during production of the breadboard model, the 3D-printed parts were printed too tight and had to be reprinted. Having underestimated the volume of the cables and the size of the STM32's evaluation board, some electronic parts had to be moved outside of the model, such as the magnetometer, which was taped to the edge of the gut module's exterior for easy reference. However, the breadboard model served well for its intended purpose, which allowed the sliding of the modules to re-flash the evaluation board when changing test modes.

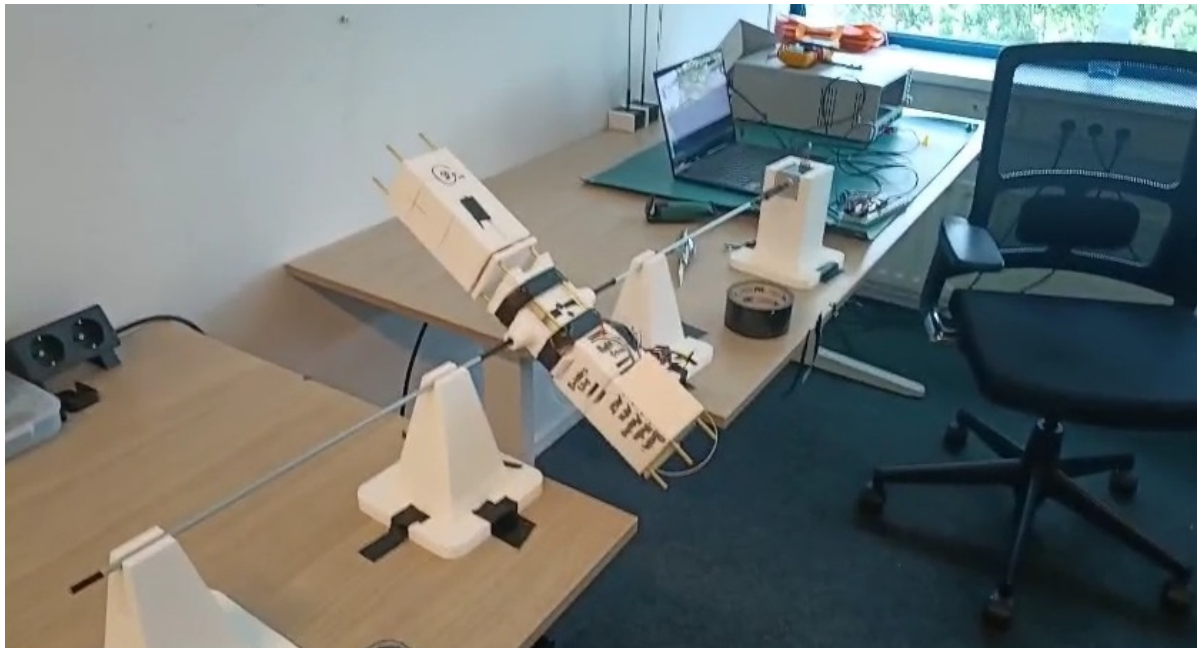
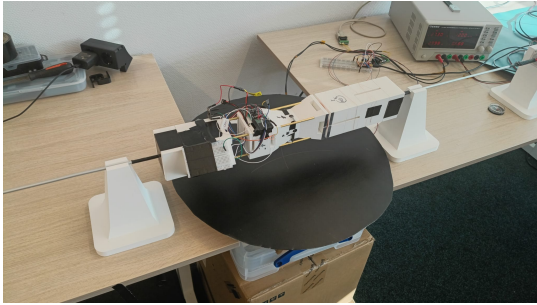
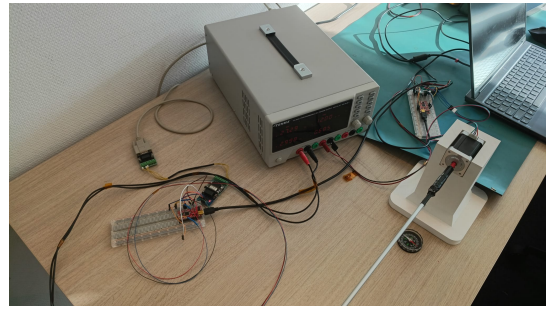


Figure 2.16: Breadboard model in detumbling configuration, with the Nema17 connected to the breadboard by aluminum rods, held in place by the supports stands, controlled by a moteino over a laptop that runs Arduino IDE.



(a) Breadboard model in the nadir pointing configuration with the horizon simulant placed underneath (black), held up by two support stands. Brain-module pointing away from the stepper motor (right-hand-side of picture).



(b) Stepper motor Nema17 sitting on the stepper support, connected to the breadboard module by an aluminum rod. Tenma power supply (box) connected to both, the stepper motor and the horizon simulant, with the necessary electronics placed on two breadboards each, each communicating with one laptop for control.

Figure 2.17: Nadir Pointing Test Configuration

DAQ Architecture and Control Interface

DAQ is performed using a Moteino microcontroller acting as a telemetry interface between the BBM and a laptop. Here the Moteino is connected to the STM32 and retrieves internal ADCS telemetry through an I²C interface. Depending on the test case, this includes magnetometer measurements from the RM3100 and infrared horizon sensor measurements from the MLX90640 sensors. However, given the large size of the full images from the IR sensors, they may be instead stored onto the STM32's flash chip and recovered later for postprocessing, with the Moteino only transmitting calculated angles from those images. Additionally, the Moteino samples the analog output signals of the DRV883x driver boards in order to determine the magnetorquer dipole command generated by the ADCS software. To evaluate these results, the MPU is connected to the Moteino to provide independent attitude measurements with respect to gravity for comparison. Finally, all measurements are transmitted wirelessly to a second Moteino connected to a laptop, where the data is recorded for analysis.

Logged Outputs during Detumbling:

- RM3100 magnetometer measurements, used together with the platform angular rate to compute the expected $\dot{B}$ value,
- Test article angular rate and orientation from the MPU6050, used as reference motion data for evaluating the imposed rotation,
- Analog outputs corresponding to the commanded magnetorquer dipole, used to evaluate the generated $B\text{-dot}$ control command,
- Timestamps, used to synchronize the magnetometer, motion, and control-output data.

Logged Outputs during Nadir Pointing:

- Calculated orientation from MLX90640 infrared horizon sensor images, used to evaluate the nadir estimation,
- Test article angular rate and orientation from the MPU6050, used as reference motion data for comparing the estimated pointing direction,
- Analog outputs corresponding to the commanded magnetorquer dipole, used to evaluate the resulting pointing-control command,
- Timestamps, used to synchronize the horizon-sensor, motion, and control-output data.

3

Requirements

In this chapter, the requirements are defined according to ECSS[2], listed in section 3.1, and elaborated on concerning the respective means of verification in section 3.2, section 3.3 and section 3.4. At the end, all requirements are checked with a compliance matrix in the final section 3.5.

3.1. Definition

Here, the initial requirements were derived from the previous stage of the project[9] which were based on the customer's needs. Therefore, to maintain consistency the same ID's were used which are divided into 6 branches and defined as follows: ENV (Environment), PWR (Power), REC (Receive), SND (Send), ACT (Actuate), and CON (Constraint). Naturally, as the project progressed, certain requirements were edited or deleted. What remained was a list of requirements found in Appendix B, some of which can still be negotiated if deemed to difficult to verify.

Due to budget constraints, all testing was conducted on one single BBM found in section 2.1. In addition to the limited availability in time, we only covered a small portion of the requirements to focus on verifying the functionality of detumbling and nadir-pointing. Identifying a group of relevant requirements for this scope, we have provided a rational for each.

Furthermore, three requirements have been altered to better fit the aim of this project:

- *ACT-1 has been altered to ACT-1**
as the original requirement can only be validated via performance testing of the MTQs. Due to the involved, extremely low magnetic torques, a Helmholtz cage was required for testing whose set up was considered too time intensive, and therefore outside the scope of this project. In more detail, the torque strengths are similar to that of the magnetic field strength of the Earth and are easily drowned out by, for example, an elevator moving.
- *ACT-3 has been altered to ACT-3**
with the same argumentation as above, but with regards to nadir pointing.
- *MGM-2 has been altered to MGM-2**
to account for the fact the magnetometer main function is to de-tumble. Thus, instead of the measuring the magnetic field strength, the change of magnetic field is needed.

Table 3.1: Requirements Definition

ID	Requirement	Rationale	Parent	Test Method
PWR-1	The system shall nominally operate using a voltage of 3.3 V	The bus voltage is unregulated between 2.2 V and 4.5 V and will be converted to 3.3 V	-	Testing
PWR-4	The system shall enter hibernate mode upon every boot-up	ADCS is a slave to the OBC, thus a command from the OBC is needed to switch to nominal operations	-	Testing
PWR-6	The system shall consume less than 1 Watt in nominal operation, with a peak consumption of 2 Watts.	Since the satellite has not more than 4 Watts available, the system is limited to only a smaller allocated amount of the power budget.	-	Testing
REC-1	The system shall recognize and ignore faulty commands from the OBC by use of its internal Cyclic Redundancy Check (CRC)	The microcontroller already has a build-in error detection method but this must also be validated to arrive at a flight-ready system	-	Testing
REC-3	The system's operational mode shall switch from <i>de-tumbling</i> when above $0.5^\circ/\text{s}$ to <i>Nadir pointing</i> when below this threshold and vice versa within one ADCS loop	The ADCS is slave to the OBC but should function independently and provide the OBC with data whenever instructed to. The threshold of $0.5^\circ/\text{s}$ was given by the customer	-	Testing
SND-4	The system shall send all stored system data to the OBC on request	The OBC needs to know the status of the ADCS. This refers to state variables, measurements, and estimations.	-	Testing
ACT-1	The system shall de-tumble the satellite by reducing its rotational rate to at most $0.5^\circ/\text{s}$ from $180^\circ/\text{s}$ in less than 72 hours	De-tumbling is one of the ADCS's main tasks. The cut-off value was set by the customer. To specify performance, maximum initial spin rate is required to be constrained. Rate of Delfi-C3 was just above $5^\circ/\text{s}$ [4] but edge cases have to be examined as rates as high as $60^\circ/\text{s}$ have been observed.	-	Analysis & Testing
ACT-1.1	The system shall command the MTQ drivers to provide a current to the MTQs based on the output of the de-tumbling algorithm	This allows the satellite to de-tumble as the total energy of the system is lowered. It was reformulated as only breadboard testing can be performed.	ACT-1	Testing
ACT-1.2	The system shall make use of a de-tumbling algorithm that determines the magnetic dipoles needed to reduce the kinetic energy of the system by utilizing three consecutive magnetometer readings.	This is the computation needed for the de-tumbling algorithm to function as it utilizes the change in magnetic field	ACT-1	Analysis & Testing
ACT-2	The system shall point the +z axis of the satellite at the velocity vector within 10° at $1-\sigma$ confidence	Key function to keep prograde orientation when pointing to Nadir. Expected to be helped by the aerodynamic shape of the hinged solar panel configuration	-	Analysis
ACT-3	The system shall point the $\pm x$ or $\pm y$ axis with the Nadir within 30° at $1-\sigma$ confidence utilizing the Earth Horizon sensing subsystem for roll rates lower than $0.5^\circ/\text{s}$ when pitch is between $\pm 45^\circ$	Main function for nadir pointing. Requires more than 45 degree accuracy to discern which face is pointed at Earth.	-	Analysis
ACT-3*	The system shall command the MTQ drivers to provide a current, for angular rates lower than 0.5° , to the MTQs which aligns the x-axis of the satellite within $\pm 30^\circ$ of the Nadir vector with a $1-\sigma$ confidence	The system must be able to point the satellite in the Nadir direction within at least $\pm 30^\circ$ as to fulfill ACT-3, it is reformulated to have a verifiable requirement for the breadboard model.	ACT-3	Analysis & Testing

Continued on next page

Table 3.1 – continued from previous page

ID	Requirement	Rationale	Parent	Test Method
ACT-4	The Earth Horizon detection algorithm shall not classify simulated solar illumination as an horizon for all orientations in the simulation except when the simulated solar illumination coincides with actual simulated horizon	The solar illumination is a source of infrared that is to be ignored for the nadir vector determination	-	Testing
EHS-02	The Earth Horizon detection algorithm shall provide pitch/roll estimates with an error $\leq 10^\circ$ of the actual value.	Ensures compliance with ACT-2 and ACT-3. Required for Nadir pointing	ACT-2/3	Testing
MTQ-01	The MTQs shall actuate for at most 50% each control loop cycle.	In accordance with PWR-06, ensuring average power stays below 1 Watt.	-	Review of Design
MTQ-02	Combined peak consumption of all MTQs shall be no more than 1.8 Watts.	Allocates the largest portion of the power budget to the primary actuators.	-	Review of Design

3.2. Verification by Analysis

Analysis is defined in [2], clause 5.2.2.3.a as "Verification by analysis shall consist of performing theoretical or empirical evaluation using techniques agreed with the Customer". Almost all these requirements were also deemed to be the leading requirements for the complete design project. So it was ensured in the detailed design phase that all these requirements would be satisfied.

For ACT-1 and ACT-3, the flight-level performance is verified by analysis and simulation updated with measured breadboard parameters. Meanwhile, breadboard tests verify controller outputs, sensor behavior, and actuator-command generation. It is not implied that the breadboard directly verifies full orbital de-tumbling or nadir pointing.

3.2.1. Requirement ACT-1

To reiterate the requirement: *The system shall de-tumble the satellite by reducing its rotational rate to at most $0.5^\circ/\text{s}$*

The preliminary simulations in the SEDDP stage of the design have shown that the subsystem complies with this requirement. However, that is all virtual with estimated values so an iteration is required for better verification.

The proposed test will measure the output of the drivers that is supposed to go to the MTQs. Furthermore, a unit test for the MTQs will also give the dipole moment and torque produced. These values will be compared together with the previously estimated performance characteristics of the air-coil and PCB printed MTQs.

The newly found parameters can be implemented into the simulations via Monte Carlo to then verify the requirement.

3.2.2. Requirement ACT-1.1

To reiterate the requirement: *The system shall command the MTQ drivers to provide a current to the MTQs based on the output of the de-tumbling algorithm*

This child requirement of ACT-1 will be verified using the same method as its parent requirement as is described in subsection 3.2.1.

3.2.3. Requirement ACT-1.2

To reiterate the requirement: *The system shall make use of a de-tumbling algorithm that determines the magnetic dipoles needed to reduce the kinetic energy of the system by utilizing three consecutive magnetometer readings.*

This child requirement of ACT-1 will be verified using the same method as its parent requirement as is described in subsection 3.2.1.

3.2.4. Requirement ACT-3*

To reiterate the requirement: *The system shall command the MTQ drivers to provide a current, for angular rates lower than $0.5^\circ/\text{s}$, to the MTQs so that magnetic dipoles are created consistent with the EHS algorithm which aligns the x-axis of the satellite within $\pm 30^\circ$ of the Nadir vector with a $1-\sigma$ confidence*

The simulation provides the simulated attitude of the satellite with the implemented EHS system. The simulation should prove that once a sufficient roll rate of $0.5^\circ/\text{s}$ or less is achieved the Nadir pointing system keeps the positive x-axis of the satellite within 30° of the Nadir vector with $1-\sigma$ confidence i.e. $> 68.27\%$ of the rest of the simulated time. This is partial verification as estimated accuracy, noise and bias are used.

There will be a second final verification once testing has been performed of the breadboard model. The actual accuracy, noise and bias will be implemented and the simulation is to show that the nadir pointing can still be achieved within the given boundaries with a $1-\sigma$ confidence to verify this requirement.

3.3. Verification by Review of Design

3.3.1. Requirement MTQ-01

To reiterate the requirement: *The MTQs shall actuate for at most 50% each control loop cycle.*

It can be proven that this requirement is satisfied by the showcase of the design of the two main modes that the main loop consists of de-tumbling and pointing. There are shown in Figure 3.1 and Figure 3.2 respectively. These timings are hardcoded and flashed into the microcontroller as to keep the system completely deterministic. So it is not possible for the MTQ to keep on firing for longer than the assigned time slot. It is possible for the MTQ to actuate for less than the assigned time if determined, such is why the requirement is formulated to include "at most".

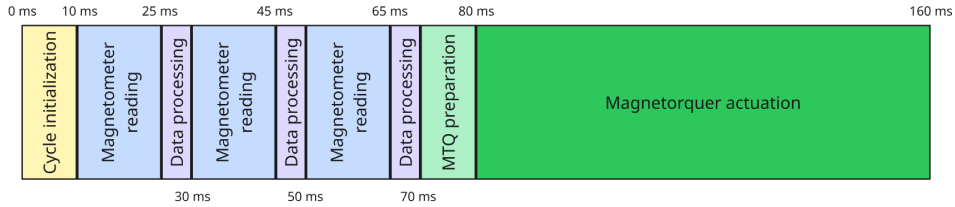


Figure 3.1: Illustration of the timing of the main loop in De-tumbling Mode

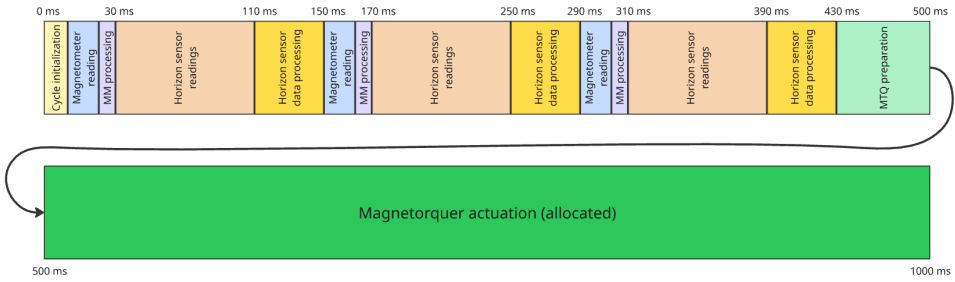


Figure 3.2: Main loop timing in Pointing mode

3.3.2. Requirement MTQ-02

To reiterate the requirement: *Combined peak consumption of all MTQs shall be no more than 1.8 W.*

The power delivery is handled by the EPS of the Delfi-Twin, which provides a nominal voltage of 3.3 V. Each MTQ coil is manufactured with a minimum resistance of 19.41 Ω. Therefore, the maximum peak current drawn by a single coil under worst-case conditions is calculated using Ohm's law:

$$I_{\text{coil}} = \frac{U}{R_{\text{coil}}} = \frac{3.3 \text{ V}}{19.41 \Omega} \approx 0.17 \text{ A} \quad (3.1)$$

When all three MTQs are simultaneously operating at peak capacity, the total maximum power consumption is:

$$P = U \cdot I_{\text{total}} = U \cdot I_{\text{coil}} \cdot 3 = 3.3 \text{ V} \cdot 0.17 \text{ A} \cdot 3 = 1.65 \text{ W} \quad (3.2)$$

Since the maximum potential power usage of 1.65 W remains below the 1.8 W threshold, the requirement is satisfied.

3.4. Verification by Test

3.4.1. Requirement PWR-1

To reiterate the requirement: *The system shall nominally operate using a voltage of 3.3 V*

This integration test is performed by supplying the system with the baseline voltage of 3.3 V. A multimeter is connected in the circuit to verify the correct voltage. If the system boots-up and performs its nominal functions, the requirements is met. In between unit tests and final tests, this requirement is expected to be passively performed upwards of fifty times.

3.4.2. Requirement PWR-4

To reiterate the requirement: *The system shall enter hibernate mode upon every boot-up*

The whole system works as a slave to the OBC. Thus, it remains in a passive state until it receives the heartbeat. This requirement is met once it is proven that when supplied with 3.3 V the system only starts communicating once it receives a mock command from a laptop pretending to be the OBC.

3.4.3. Requirement PWR-6

To reiterate the requirement: *The system shall consume less than 1 W in nominal operation, with a peak consumption of 2 W*

During the experiments and unit tests a multimeter is included in the circuit so the power consumption can be read out. The peak consumption which is a maximum expected value (MEV) happens during de-tumbling is 0.99 W derived from our current best estimate (CBE). Thus, a margin of 61% is taken as a conservative approach.

3.4.4. Requirement REC-1

To reiterate the requirement: *The system shall recognize faulty commands from the OBC by use of its internal Cyclic Redundancy Check (CRC)*

A laptop will be connected to the system that will pose as the OBC and send a command that should get flagged by the CRC. If the system rejects the command the requirement is considered to be met. This requirement was implemented to mitigate the risk of bit switches causing freezes or errors in the operations of the ADCS. The faulty command packet is

3.4.5. Requirement REC-3

To reiterate the requirement: *The system's operational mode shall switch from de-tumbling when above $0.5^\circ/\text{s}$ to Nadir pointing when below this threshold and vice versa within one ADCS loop*

This requirement arose from the constraint that the ADCS is developed independently from the other systems and no definite list of included subsystems is present. Thus, relying on a GNSS subsystem for orbital positioning data is not possible. Furthermore, to ensure integration with the least amount of alterations to the flight software of the Delfi-Twin the modes, de-tumbling & nadir pointing, will switch automatically.

This is tested on the breadboard model via a stepper motor that will cross the rotational threshold velocity so from $1^\circ/\text{s}$ to $0.1^\circ/\text{s}$ in 20 seconds. It should be observed from the output of the microcontroller that from the 11th to 12th second the mode switches.

3.4.6. Requirement SND-4

To reiterate the requirement: *The system shall send all stored system data to the OBC on request*

This requirement is to be verified together with the other integration requirements with the set-up given in subsection 3.4.4. The laptop that acts as the OBC and after it has sent a data request, it awaits the data package. The requirement is met once this package is received. The experiment is repeated whilst different processes are running on the microcontroller to test the Real-Time Operating System capabilities.

The telemetry package should consist of two parts. The first is the health status of the system, this includes the currents measured via its Analog-to-Digital Converter (ADC) and sensors/parts registered as active. The second part is the attitude information. This state estimation consists of the pitch and roll estimates.

3.4.7. Requirement ACT-1

To reiterate the requirement: *The system shall de-tumble the satellite by reducing its rotational rate to at most $0.5^\circ/\text{s}$ from $180^\circ/\text{s}$ in less than 72 hours*

This is the main requirement regarding the de-tumbling capabilities of the ADCS. It can be mostly

3.4.8. Requirement ACT-1.1

To reiterate the requirement: *The system shall command the MTQ drivers to provide a current to the MTQs based on the output of the de-tumbling algorithm*

The ideal output is known for the de-tumble scenario's as a constant angular velocity will be used.

3.4.9. Requirement ACT-1.2

To reiterate the requirement: *The system shall make use of a de-tumbling algorithm that determines the magnetic dipoles needed to reduce the kinetic energy of the system by utilizing three concurrent magnetometer readings*

3.4.10. Requirement ACT-3*

To reiterate the requirement: *The system shall command the MTQ drivers to provide a current, for angular rates lower than $0.5^\circ/\text{s}$, to the MTQs so that magnetic dipoles are created consistent with the EHS algorithm which aligns the x-axis of the satellite within $\pm 30^\circ$ of the Nadir vector with a $1-\sigma$ confidence*

The simulation should have proven that for more than 68.27% of the time the positive x-axis of the satellite is pointed in the right direction. Now it has to proven that the actual hardware also provides the required outputs to achieve this. The STM32 should command the drivers of the MTQ to fire the MTQs with a certain amount of current. This current is measured by implementing a Moteino and resistor in the circuit. The conditions and attitude of the breadboard model are known and thus the expected correct output as well. These will be compared to the measured values to characterize the accuracy, noise and bias for the breadboard system. This will then be used to refine the simulated EHS system to verify the $1-\sigma$ confidence to fully verify this requirement.

3.4.11. Requirement ACT-4

To reiterate the requirement: *The Earth horizon detection algorithm shall not classify solar illumination as an horizon of the Earth for all parts of the orbit except when the solar illumination coincides with the actual horizon*

This requirement was met via the implementation of a flood fill algorithm. When the pixels are checked to be above the threshold temperature a secondary algorithm is counting each pixel that surpasses the threshold. It checks whether its neighbors temperature is also above the threshold and if it is, it adds it to the 'blob'. It keeps track of all the clumps of pixels that exceed the threshold and when it has finished checking all the pixels it chooses the blob with the highest amount of pixels in it. This will guarantee that the Earth will always be chosen as the Sun would, at most, occupy 4 pixels, if positioned perfectly in the middle of them due to its tiny solid angle of 6.794×10^{-5} sr. It must be noted that optical blur via point spread function is possible so the surrounding pixels could give a faint temperature reading that is higher than the coldness that would be observed of deep space. This would still not affect the algorithm as blobs consisting of 20 pixels or less are not considered, due to the large inaccuracies occurring if the Earth is barely in view. It was proven in SEDDP that there will always be at least one sensor with a view of Earth of at least 20 pixels when the z-axis is roughly aligned with the pro-grade motion[9].

This test will be done by including a small heated object, as this will then emit in the infrared band that the MLX90642s use i.e. $8\text{ }\mu\text{m}$ to $14\text{ }\mu\text{m}$. It should be so small that it subtends 0.533° of the MLX90642's view, and has an intensity of 188.45 W m^{-2} . This will be too small for the scaled down distances that the tests work with, this is not a problem as the flood fill algorithm ignores all but the largest blob. Thus, a larger heated source that is omitted proves that a smaller source would also be omitted. It will be included in a ten static tests and a few dynamic tests. The masks will be read out and checked for the inclusion of only one blob.

It should not show up on the masks created from which the algorithm will estimate the nadir vector, the requirement is considered to be met when the heated object is never included in the mask as long as there is at least two pixels between the horizon simulant and the laser pointer. The laser pointer test will

be performed for five cases where one of the five is the edge cases where overlap is expected to occur. The requirement is considered a pass when only the overlapped case shows the inclusion of the heated object.

3.4.12. Requirement EHS-02

To reiterate the requirement: *The Earth Horizon detection algorithm shall provide pitch/roll estimates with an error $\leq 10^\circ$ of the actual value.*

This is tested via static and dynamic tests

3.5. Compliance matrix

This section checks if all the requirements have been passed or failed. It must be mentioned that quite a lot of requirements were not able to be tested due to severe time constraints resulting from the limited size of the group having to tackle the interfacing of several SEP projects e.g. EHS system and MTQs. These are expected to be passed but can not be substantiated to a "pass" without proper testing thus still remain in their "pending" status.

Some requirements were passed via unit tested like the nominal operations using a voltage of 3.3 V and sun-in-view mitigation.

The simulation initially made during SEDDP was improved to show the detumbling. This has been shown and communicated with our supervisor.

¹the simulation passed the requirement for all scenario's except 600 km altitude with a solar panel angle of 180°

ID	Requirement Description	Method	Reference	Outcome/Status
PWR-1	The system shall nominally operate using a voltage of 3.3 V	Testing	Appendix C	Pass
PWR-4	The system shall enter hibernate mode upon every boot-up	Testing	TBR/section 9.3	Fail
PWR-6	The system shall consume less than 1 W in nominal operation, with a peak consumption of 2 W	Testing	TBR/section 9.3	Pending
REC-1	The system shall recognize and ignore faulty commands from the OBC by use of its internal Cyclic Redundancy Check	Testing	TBR/section 9.3	Pending
REC-3	The system's operational mode shall switch from <i>de-tumbling</i> when above $0.5^\circ/\text{s}$ to <i>Nadir pointing</i> when below this threshold and vice versa within one ADCS loop	Testing	Appendix C	Pass
SND-4	The system shall send all stored system data to the OBC on request	Testing	TBR/section 9.3	Pending
ACT-1	The system shall de-tumble the satellite by reducing its rotational rate to at most $0.5^\circ/\text{s}$ from $180^\circ/\text{s}$ in less than 72 hours	Analysis	SEDDP[9]	Pass
		Testing	??	Fail
ACT-1.1	The system shall command the MTQ drivers to provide a current to the MTQs based on the output of the de-tumbling algorithm	Testing	Appendix C	Pass
ACT-1.2	The system shall make use of a de-tumbling algorithm that determines the magnetic dipoles needed to reduce the kinetic energy of the system by utilizing three concurrent magnetometer readings	Analysis	SEDDP[9]	Pass
		Testing	??	Fail
ACT-2	The system shall point the +z axis of the satellite at the velocity vector within 10° at $1-\sigma$ confidence	Analysis	chapter 8	Pending
ACT-3	The system shall point the +x axis with the Nadir vector within 30° at $1-\sigma$ confidence utilizing the EHS for roll rates below $0.5^\circ/\text{s}$ when pitch is between $\pm 45^\circ$	Analysis	SEDDP[9]	Pass ¹
ACT-3*	The system shall command the MTQ drivers to provide a current, for angular rates lower than $0.5^\circ/\text{s}$, to the MTQs which aligns the x-axis of the satellite within $\pm 30^\circ$ of the Nadir vector with a $1-\sigma$ confidence	Analysis	SEDDP[9]	Pass ¹
		Testing	chapter 3	Fail
ACT-4	The Earth Horizon detection algorithm shall not classify simulated solar illumination as an horizon for all orientations in the simulation except when the simulated solar illumination coincides with actual simulated horizon	Testing	subsection 8.1.1	Pass
EHS-02	The Earth Horizon detection algorithm shall provide pitch/roll estimates with an error $\leq 10^\circ$ of the actual value	Testing	subsection 8.2.1	Fail
MTQ-01	The MTQs shall actuate for at most 50% each control loop cycle	Review of Design	subsection 3.3.1 & SEDDP[9]	Pass
MTQ-02	Combined peak consumption of all MTQs shall be no more than 1.8 W	Review of Design	subsection 3.3.2 & SEDDP[9]	Pending

Table 3.2: Requirement Compliance Matrix

4

Hypothesis formulation and assessment

As the verification of the detumbling and nadir pointing algorithms are the main focus of the SEP, they need to be tested experimentally. Based on the previous re-evaluation in chapter 3, we have identified 3 critical requirements that will prove overall functionality if tested. In this chapter, we delve into the formulation of the respective hypotheses, and the means to quantify the outcome of the two separate, but similar, experiments.

Treating the algorithms as deterministic, we only consider its inputs and outputs for verification. In this case, the inputs concern the rate of angular rotation, or fixed rotation, imposed on the test article, while the outputs are computed values from magnetometer and earth horizon sensor readings. Expecting inaccurate measurements due to noise, physical, and digital model limitations, we aim to demonstrate a robust algorithm that actuates reliably regardless.

Here, the formulated hypotheses have to adhere to five criteria to ensure they can be verified:

- **Testability:** Can an experiment be designed to principally reject the null hypothesis?
- **Parsimony:** Is it the hypothesis with the fewest assumptions and not overly complex?
- **Scope:** Does the hypothesis properly encompass the scenario it tries to test?
- **Fruitfulness:** Can the hypothesis explain phenomena and help build future research?
- **Conservativeness:** Does the hypothesis minimize the false positives (Type 1 errors)?

4.1. Formulation

Hypothesis Formulation - De-tumbling Experiment

To verify de-tumbling functionality, the following requirement needs to be met:

- **ACT-1.2:** *"The system shall make use of a de-tumbling algorithm that determines the magnetic dipoles needed to reduce the kinetic energy of the system by utilizing three consecutive magnetometer readings."*

Null-Hypothesis (requirement not met):

Given a constant angular velocity, the sensor measurements provide values to compute a magnetic dipole vector such that the predicted angular momentum change vector has an angle ($\theta_{\Delta H, \omega}$) equal to, or lower than 90° with respect to the angular velocity vector of the system.

$$H_{0,ACT-1.2} : \mu_{\theta_{\Delta H, \omega}} - \sigma_{\theta_{\Delta H, \omega}} \leq 90^\circ$$

Alternative Hypothesis (requirement met):

Given a constant angular velocity, the sensor measurements provide values to compute a magnetic dipole vector such that the predicted angular momentum change vector has an angle ($\theta_{\Delta H, \omega}$) larger than 90° with respect to the angular velocity vector of the system.

$$H_{1,ACT-1.2} : \mu_{\theta_{\Delta H, \omega}} - \sigma_{\theta_{\Delta H, \omega}} > 90^\circ$$

Rationale: Detumbling requires the predicted angular momentum change to oppose the angular velocity vector, since this reduces the rotational kinetic energy of the BBM. This is satisfied when the angle $\theta_{\Delta H, \omega}$ between the predicted angular momentum change vector and the angular velocity vector exceeds 90° . The output quantity is therefore $\theta_{\Delta H, \omega}$ [°]. The metric $\mu_{\theta_{\Delta H, \omega}} - \sigma_{\theta_{\Delta H, \omega}}$ is used as a conservative test value, since the mean angle must remain above 90° even after accounting for one standard deviation of spread. This defines a one-sided mean-deviation test.

Hypothesis Formulation - Nadir Pointing Experiment

- **EHS-02:** *"The Earth Horizon detection algorithm shall provide pitch/roll estimates with an error $\leq 10^\circ$ of the actual value."*

Null-Hypothesis (requirement not met):

The 95th percentile of either the absolute pitch or roll estimation error is equal to, or exceeds 10° .

$$H_{0,EHS-02} : P_{95}(e_{roll}) \geq 10^\circ \vee P_{95}(e_{pitch}) \geq 10^\circ$$

Alternative Hypothesis (requirement met):

The 95th percentiles of both the absolute pitch and roll estimation errors are less than 10° .

$$H_{1,EHS-02} : P_{95}(e_{roll}) < 10^\circ \wedge P_{95}(e_{pitch}) < 10^\circ$$

Rationale: The output quantities are defined as the absolute angular estimation errors e_{roll} [°] and e_{pitch} [°], where $e_\theta(t) = |\theta_{estimated}(t) - \theta_{actual}(t)|$. The criterion is evaluated separately for pitch and roll. The 95th percentile was chosen to ensure that at least 95% of the estimated pitch and roll values remain within the specified 10° accuracy bound. This requirement warrants a one-sided percentile test.

- **ACT-3*:** *"The system shall align the +x-axis of the satellite within $\pm 30^\circ$ of the Nadir vector with a 1- σ confidence at angular rates equal to or lower than $0.5^\circ/\text{s}$."*

Decomposition: The requirement is decomposed into two daughter requirements, both of which must hold for angular rates equal to or lower than $0.5^\circ/\text{s}$. ACT-3* is considered verified only if both alternative hypotheses are accepted.

- **ACT-3.1*:** *"The angular momentum change generated by the controller shall oppose the roll-error direction in order to reduce the roll-angle deviation from nadir, for angular rates equal to or lower than $0.5^\circ/\text{s}$."*

Control Direction

Null-Hypothesis:

For angular rates equal to or lower than $0.5^\circ/\text{s}$, the mean angle reduced by one standard deviation between the reference angular momentum change vector and the roll-error reference axis is equal to, or lower than 90° .

$$H_{0,ACT-3.1*} : \mu_{\theta_c} - \sigma_{\theta_c} \leq 90^\circ$$

Alternative Hypothesis:

For angular rates equal to or lower than $0.5^\circ/\text{s}$, the mean angle reduced by one standard deviation between the reference angular momentum change vector and the roll-error reference axis exceeds 90° .

$$H_{1,ACT-3.1*} : \mu_{\theta_c} - \sigma_{\theta_c} > 90^\circ$$

Rationale: The control direction is evaluated using the angle θ_c [$^\circ$] between the reference angular momentum change vector $\Delta \mathbf{H}_{ref}$ and the roll-error reference axis $\hat{\mathbf{a}}_{err}$. The roll-error reference axis is defined as the direction associated with the measured roll-angle deviation from nadir. A valid corrective action requires $\Delta \mathbf{H}_{ref}$ to oppose this error direction, corresponding to $\theta_c > 90^\circ$. The metric $\mu_{\theta_c} - \sigma_{\theta_c}$ is used as a conservative test value, since the mean direction must remain above 90° even after accounting for one standard deviation of spread. This defines a one-sided mean-deviation test.

- **ACT-3.2*:** *"The magnitude of the commanded correction shall increase with increasing roll-angle deviation from nadir, for angular rates equal to or lower than $0.5^\circ/\text{s}$."*

Control Magnitude

Exclusion of ACT-3.2*: During the formulation of this chapter, it was decided that ACT-3.2* is no longer relevant within the scope of this project. The objective of the experimental verification is to assess the functional correctness of the detumbling and nadir-pointing algorithms, not to evaluate their control performance. For the nadir-pointing case, the functional requirement is that the controller commands a correction in the direction that reduces the roll-angle deviation from nadir. This is captured by ACT-3.1*. In contrast, ACT-3.2* concerns whether the commanded correction magnitude scales with the size of the pointing error. This affects performance aspects such as response strength, convergence rate, and settling behaviour, but it is not required to prove that the nadir-pointing function is present. Therefore, ACT-3.2* is treated as out of scope for the hypothesis assessment, while ACT-3.1* is retained to verify the functional correctness of the commanded control direction.

Hypothesis evaluation

Criterion	$H_{ACT-1.2}$	H_{EHS-02}	$H_{ACT-3.1^*}$
Testability	The angle $\theta_{\Delta H, \omega}$ can be computed from the predicted angular momentum change and angular velocity vectors, then tested against 90° .	Pitch and roll errors can be computed from estimated and actual angles, then tested against the 10° limit.	The angle θ_c can be computed between $\Delta \mathbf{H}_{ref}$ and the roll-error reference axis, then tested against 90° .
Parsimony	The hypothesis uses one angular metric to assess whether the control action opposes the imposed rotation.	The hypothesis only uses the absolute pitch and roll estimation errors.	The hypothesis uses one angular metric to assess whether the control action opposes the roll-error direction.
Scope	The hypothesis applies to the detumbling rotation cases tested on the BBM.	The hypothesis applies to both pitch and roll estimation during the nadir-pointing test.	The hypothesis applies to the functional control direction of the nadir-pointing algorithm.
Fruitfulness	The result shows whether the detumbling algorithm generates angular momentum changes in the correct direction.	The result shows whether the EHS accuracy is sufficient for nadir-pointing control.	The result shows whether the controller commands corrections in the desired direction.
Conservativeness	Using $\mu - \sigma$ requires the mean direction to remain above 90° even after accounting for measurement spread.	Using P_{95} requires most pitch and roll estimates to remain within the allowed error bound.	Using $\mu - \sigma$ requires the mean correction direction to remain above 90° even after accounting for measurement spread.

Table 4.1: Evaluation of each retained hypothesis against the five criteria.**4.2. Evaluation**

The retained hypotheses are evaluated using the metrics defined in section 4.1. Since the algorithms are treated as deterministic, the assessment is based on measured output quantities and predefined threshold criteria rather than a separate stochastic model. Therefore, no distribution is assumed beyond the measured mean, standard deviation, or percentile required by each hypothesis.

For ACT-1.2, the output quantity is the angle $\theta_{\Delta H, \omega}$ [°] between the predicted angular momentum change vector and the angular velocity vector. The distribution is quantified using the sample mean $\mu_{\theta_{\Delta H, \omega}}$ and standard deviation $\sigma_{\theta_{\Delta H, \omega}}$. The test metric is $\mu_{\theta_{\Delta H, \omega}} - \sigma_{\theta_{\Delta H, \omega}}$, which is compared against the 90° threshold. This introduces a one-standard-deviation margin and reduces the risk of falsely accepting a weak detumbling response.

For EHS-02, the output quantities are the absolute roll and pitch estimation errors, e_{roll} [°] and e_{pitch} [°]. The requirement is evaluated using the 95th percentile criterion, meaning that at least 95% of the pitch and roll estimates must remain within the 10° error bound. The null hypothesis is rejected only if both pitch and roll satisfy this condition.

For ACT-3.1*, the output quantity is the angle θ_c [°] between the reference angular momentum change vector and the roll-error reference axis. The distribution is quantified using the sample mean μ_{θ_c} and standard deviation σ_{θ_c} . The test metric is $\mu_{\theta_c} - \sigma_{\theta_c}$, which is compared against the 90° threshold. This verifies whether the commanded control direction opposes the roll-error direction with a conservative one-standard-deviation margin.

No separate p-value is used for these tests. The hypotheses are assessed as engineering verification tests with fixed acceptance criteria. A false positive is limited by using conservative metrics, namely $\mu - \sigma$ for direction tests and P_{95} for attitude-estimation accuracy.

4.3. Assessment

The assessment is performed by comparing the measured test metrics against the acceptance criteria defined in section 4.1. Since the hypotheses are evaluated using threshold-based verification metrics, no separate p-values are reported. The null hypothesis is rejected only if the measured metric satisfies the corresponding acceptance condition.

For ACT-1.2, the measured mean angle between ΔH_{pred} and $\hat{\omega}$ is 15.5° , with an averaged standard deviation of 19.5° . The conservative test metric is therefore

$$\mu_{\theta_{\Delta H, \omega}} - \sigma_{\theta_{\Delta H, \omega}} = 15.5^\circ - 19.5^\circ = -4.0^\circ.$$

Since this value does not exceed 90° , the null hypothesis is not rejected. Therefore, ACT-1.2 is not verified by the test results.

For EHS-02, the roll estimate remains within the 10° requirement for 53.77 % of the evaluated samples, or 56.75 % after degeneracy correction. The pitch estimate remains within the 10° requirement for 57.79 % of the evaluated samples, or 80.16 % after degeneracy correction. Since neither pitch nor roll reaches the required 95 % threshold, the null hypothesis is not rejected. Therefore, EHS-02 is not verified by the test results.

For ACT-3.1*, the required values of μ_{θ_c} and σ_{θ_c} are not reported in the available test results. Therefore, the metric $\mu_{\theta_c} - \sigma_{\theta_c}$ cannot be evaluated. The null hypothesis is consequently not rejected, and ACT-3.1* remains not assessed based on the available data.

5

Experiment Design

This chapter defines the experiments used to evaluate the hypotheses formulated in section 4.1. The objective is not only to verify requirements in general, but to test whether the implemented algorithms perform their intended functional role on the BBM. For the detumbling algorithm, this means verifying whether the controller computes a magnetic dipole command that produces a predicted angular momentum change opposing the imposed angular velocity. For the nadir-pointing algorithm, this means verifying whether the EHS provides pitch and roll estimates within the required accuracy and whether the controller commands a correction direction that reduces the roll-angle deviation from nadir.

Two test cases are used. The first test case evaluates detumbling using imposed constant rotation rates. The second test case evaluates nadir pointing using a discrete attitude sweep with respect to an artificial horizon. Together, these tests provide the data needed to assess ACT-1.2, EHS-02, and ACT-3.1*. ACT-3.2* is not included as a separate experimental hypothesis, since the correction magnitude response is treated as a performance-related property rather than a functional verification condition.

5.1. Experiment definition

The experiments are designed as functional verification tests. They do not reproduce the full orbital environment. Instead, they evaluate whether the implemented algorithms respond correctly to the measured sensor inputs available to the controller during laboratory testing.

The difference between the magnetic environment in the laboratory and in space is treated as a limitation of the test setup. In orbit, the magnetic field changes with position, attitude, and time. In the laboratory, the BBM is exposed to the local Earth magnetic field and to local magnetic disturbances from the test setup. Therefore, the experiment does not directly verify in-orbit magnetic control performance. Instead, the local measured magnetic field is treated as the input environment for the controller. The algorithm output is then evaluated relative to this measured field.

Magnetic-environment limitation: The laboratory magnetic field is not equivalent to the orbital magnetic environment. Since the purpose of this experiment is functional verification, the local measured magnetic field is used as the controller input environment. The experiment therefore verifies whether the algorithm computes the correct command direction under the tested magnetic conditions. Flight-level performance, such as the actual detumbling time or nadir-pointing settling behaviour in orbit, remains outside the scope of this test and must be verified by analysis or simulation.

Test-case Detumbling

The objective of the detumbling test is to verify whether the detumbling algorithm computes a magnetic dipole command that would reduce the rotational kinetic energy of the BBM. This test is linked to ACT-1.2, since the requirement concerns the ability of the detumbling algorithm to determine magnetic dipoles that reduce the kinetic energy of the system.

The test article is rotated at constant angular rates using the stepper-motor setup. The magnetometers measure the local magnetic field and provide these measurements to the controller. From these values, the controller computes the magnetic-field derivative and generates a magnetic dipole command during each main-loop cycle.

The primary output metric is the angle $\theta_{\Delta H, \omega}$ between the predicted angular momentum change vector and the imposed angular velocity vector. This metric is used to assess directional correctness, not general accuracy. A command is directionally correct when the predicted angular momentum change opposes the imposed angular velocity, corresponding to $\theta_{\Delta H, \omega} > 90^\circ$.

The following angular rates are included in the detumbling sweep:

$$\omega \in \{1, 2, 5, 10, 20, 30, 60, 90, 120, 180\}^\circ \text{ s}^{-1}.$$

During each run, the measured magnetic field, controller time stamps, computed magnetic-field derivative, commanded magnetic dipole components, and predicted angular momentum change are recorded or reconstructed. Where available, the MTQ driver current and actuation time are used as secondary checks to confirm that the command is physically reasonable. These values are not used as the main hypothesis metric, but they help show that the controller does not only produce a correct direction with an unrealistic or zero actuation command.

Test-case Nadir Pointing

The objective of the nadir-pointing test is to verify the functional chain from horizon sensing to commanded correction direction. This test is linked to EHS-02 and ACT-3.1*. EHS-02 is assessed using the pitch and roll estimates produced by the EHS, while ACT-3.1* is assessed using the direction of the commanded correction.

The BBM is rotated with respect to an artificial horizon in discrete attitude increments. Starting from an initial orientation, the spacecraft is rotated by 5° at a rate of 1° s^{-1} . Once the target attitude is reached, the rotation is stopped and the spacecraft is held at that orientation long enough to collect sufficient sensor data for a stable attitude estimate. After the measurement interval, the spacecraft is again rotated by 5° and held stationary. This process is repeated until a full 360° rotation has been completed.

The resulting data set therefore consists of a sequence of discrete attitude points spaced by 5° , rather than a continuously rotating motion. This approach allows the EHS performance to be evaluated under quasi-static conditions while still covering the complete nadir-pointing geometry, including small-angle regions, large-angle regions, sensor hand-off regions, and possible roll/pitch degeneracy.

During the test, the EHS estimates pitch and roll, while the magnetometers provide magnetic-field data to the controller. The controller then computes the magnetic dipole command associated with the nadir-pointing correction. For EHS-02, the measured outputs are the pitch and roll estimation errors with respect to the reference attitude imposed by the rotation setup. For ACT-3.1*, the measured output is the direction metric θ_c , which compares the reference angular momentum change vector with the roll-error reference axis. This evaluates whether the commanded correction direction opposes the roll-error direction.

ACT-3.2* is not evaluated as a separate hypothesis, since the magnitude of the commanded correction is a performance metric rather than a required condition for proving that the nadir-pointing function is present.

5.2. Measurables

This section defines the variables recorded during the two experiments. For each test case, the variables are listed together with their role in the experiment and their use in the final hypothesis assessment. Controller data refers to values logged directly by the onboard controller. Reference data refers to imposed values from the test setup, such as motor commands, artificial-horizon geometry, or known reference angles. Derived data refers to quantities computed during post-processing from the logged measurements. The numerical measurement uncertainties and their propagation to the hypothesis metrics are treated separately in section 5.4.

5.2.1. Experiment variables: Test-case Detumbling

The detumbling test uses the imposed angular velocity as the main independent variable. The measured magnetic field and controller output are then used to assess whether the commanded magnetic dipole would generate an angular momentum change opposing the imposed rotation.

Variable	Type	Use
ω [$^{\circ} \text{s}^{-1}$]	Reference input	Imposed by the stepper motor. Tested over 1, 2, 5, 10, 20, 30, 60, 90, 120, 180 $^{\circ} \text{s}^{-1}$.
Rotation axis	Reference input	Fixed by the mechanical setup for each run. Defines the direction of the imposed angular velocity vector.
t [s]	Controller data	Used to synchronize magnetometer data, B-dot computation, and command generation.
B [μT]	Controller data	Measured by the RM3100 magnetometer and used as input for the detumbling algorithm.
$\dot{B}$ [T s^{-1}]	Derived data	Computed from consecutive magnetic-field measurements and used to generate the magnetic dipole command.
m_{cmd} [A m^2]	Controller output	Logged or reconstructed from the controller output. Used to determine the commanded magnetic torque.
ΔH_{pred} [N m s]	Derived data	Computed from the magnetic dipole command and the measured magnetic field.
$\theta_{\Delta H, \omega}$ [$^{\circ}$]	Assessment metric	Angle between ΔH_{pred} and ω . Used for ACT-1.2 with the condition $\theta_{\Delta H, \omega} > 90^{\circ}$.

Table 5.1: Measured and derived variables for the detumbling test case.

5.2.2. Experiment variables: Test-case Nadir Pointing

The nadir-pointing test uses the imposed attitude with respect to the artificial horizon as the main independent variable. The BBM is rotated through a full circle in discrete 5° increments at 1°s^{-1} . The EHS output is compared against the reference attitude, while the controller command is used to evaluate the correction direction.

Variable	Type	Use
ϕ_{ref} [°]	Reference input	Reference roll angle imposed by the stepper-motor setup. The BBM is rotated through 360° in 5° increments.
θ_{ref} [°]	Reference input	Reference pitch angle defined by the artificial-horizon geometry. Used as the pitch reference for EHS-02.
d [m]	Controlled variable	Fixed artificial-horizon distance. Defines the apparent horizon geometry seen by the IR sensors.
t [s]	Controller data	Used to synchronize EHS, magnetometer, and controller-command data.
ϕ_{EHS} [°]	Controller output	Estimated roll angle from the EHS. Compared with ϕ_{ref} .
θ_{EHS} [°]	Controller output	Estimated pitch angle from the EHS. Compared with θ_{ref} .
e_{roll} [°]	Derived data	Roll error computed from ϕ_{EHS} and ϕ_{ref} . Used for EHS-02.
e_{pitch} [°]	Derived data	Pitch error computed from θ_{EHS} and θ_{ref} . Used for EHS-02.
B [μT]	Controller data	Measured by the magnetometer and used during nadir-pointing command generation.
m_{cmd} [A m ²]	Controller output	Logged or reconstructed from the controller output. Used to evaluate the commanded correction direction.
θ_c [°]	Assessment metric	Direction metric used for ACT-3.1* to check whether the commanded correction opposes the roll-error direction.

Table 5.2: Measured and derived variables for the nadir-pointing test case.

5.3. Sample size

The sample size is defined by the number of valid logged samples or actuation events remaining after synchronization and filtering. Since the retained hypotheses use mean, standard deviation, and percentile metrics, the sample size must be large enough to characterize the corresponding output distributions.

For ACT-1.2, the final detumbling evaluation uses 7523 actuation events. These events are used to compute the mean and standard deviation of the direction metric $\theta_{\Delta H, \omega}$ over the tested rotation cases. The acceptance condition is then applied to the conservative metric $\mu_{\theta_{\Delta H, \omega}} - \sigma_{\theta_{\Delta H, \omega}}$.

For EHS-02, the nadir-pointing evaluation uses the logged pitch and roll estimates obtained during a full 360° attitude sweep. The spacecraft is rotated in discrete 5° increments at 1 ° s⁻¹, after which it is held stationary to collect sufficient measurements for attitude estimation. The data are therefore grouped according to these discrete attitude points rather than being sampled during continuous rotation. The uploaded nadir-pointing data set contains 4538 logged samples across 100 CSV files. These samples are used to determine whether at least 95

For ACT-3.1*, the same nadir-pointing command data are used to evaluate the control-direction metric θ_c . The nadir-pointing data set contains up to 4538 logged samples before filtering. Only samples with valid attitude, magnetic-field, and command data are retained for the final ACT-3.1* assessment. The final number of valid samples after filtering is reported together with the test result.

The test sequence is not randomized. The detumbling rates and nadir-pointing attitude points are swept in a predefined order to keep the mechanical setup repeatable and to simplify data synchronization. Time-dependent effects, such as sensor settling, hand-off behaviour, and timing drift, are therefore treated as part of the uncertainty evaluation rather than through randomization.

5.4. Uncertainty quantification

This section aims to quantify how errors propagate through the whole test set-up to the results that are used to validate the requirements.

The error propagations of the measurables stated in section 5.2 are considered and reiterated below.

- **Rotational rates** (ω_i): Measured via the RM3100 magnetometer by the change in magnetic field. The initial uncertainty (σ_ω) is derived from the sensor's inherent noise density, zero-rate

- bias and fluctuating external field.
- Magnetic dipole
 - Calculated B-dot
 - **Pitch angle** (ϕ): Measured by the EHS, the uncertainty (σ_ϕ) derives from the initial resting position not being level, the horizon simulant not being a perfect representation of a scaled down Earth and the inaccurate mounting of the IR sensors in their housing
 - **Roll angle** (θ): Uncertainty (σ_θ) is derived from the inaccuracy of the stepper motor due to it having a finite amount of teeth, the horizon simulant not being a perfect representation of a scaled down Earth and the inaccurate mounting of the IR sensors in their housing
 - **Distance to simulant** (d): Uncertainty (σ_d) is derived from the geometric misalignment that shift during the rotation of the test article

Assuming random independent uncertainties, the general error propagation formula can be applied:

$$\sigma_f = \sqrt{\sum \left(\frac{\partial f}{\partial x_i} \right)^2 \cdot \sigma_{x_i}^2} \quad (5.1)$$

5.5. Uncertainty evaluation

This section compares the measurements acquired during phase D3 against the theoretical uncertainties considered during phase D2 to assess overall systemic reliability and requirement compatibility.

Comparison of Expected vs Measured Uncertainty

During final testing it was observed that the systemic uncertainty exceeded the boundaries anticipated during D2, mostly due to inept functioning of the underlying algorithms. When the system did function properly the baseline errors were 3.85° for roll and 4.23° for pitch. Consequently, requirement **EHS-02** was not reliably able to be met. Both pitch and roll only satisfied the $\leq 10^\circ$ of the true attitude requirement just over half of the tested time.

However, as briefly stated, this was mainly due to algorithmic error and degeneracy in attitude states. Both the sensor mounting inaccuracies, sensor distance deviation and horizon simulant leveling did not prove to cause significant error in the measurements.

The magnetic background noise present in the high-rise building of Aerospace Engineering is shown in ??.

6

Data Acquisition and Processing

This chapter explains how experimental data is collected and processed during testing. First, section 6.1 provides an overview of the whole DAQ architecture by the means of a block diagram, describing each element, and explaining how data flows within the pipeline. Secondly, to make sure that the quality of the data is valid for evaluation, a calibration strategy is presented. Finally, section 6.2 delves into data processing for the sake of evaluation.

6.1. Acquisition

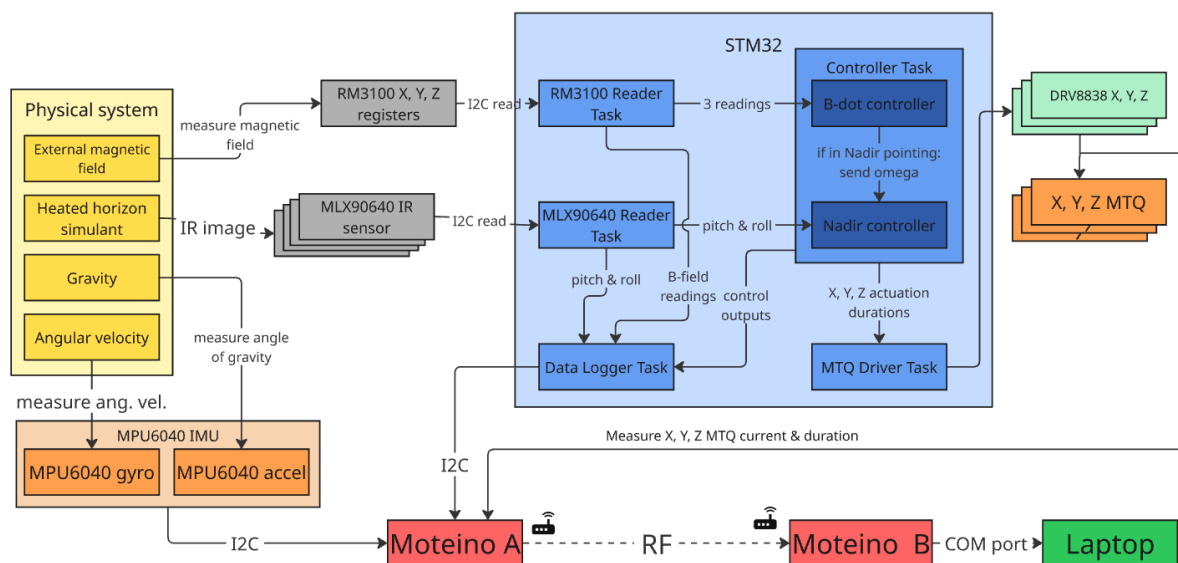


Figure 6.1: Block diagram of data acquisition

In this diagram, the overview of the DAQ architecture can be found. First, analog values are read through the two ADCS sensor types, the magnetometer and the IR sensors, as well as a gyroscope to provide independent readings on test article attitude. All readings are then processed by the STM32 in its internal tasks, including a data logger that sends values directly to Moteino A, which is also able to read the currents through each MTQ, providing a way to externally verify the STM32's outputs. Finally, Moteino A's measurements are sent wirelessly to a second Moteino B, directly connected to a laptop, where all outputs are saved via a serial monitor.

6.1.1. Architectural Philosophy

Acquisition Subsystem

The framework is divided into two independent subsystems.

The acquisition subsystem is responsible only for transporting telemetry and storing measurements.

The acquisition chain is

```
STM32
|
v
I2C
|
v
Sender Moteino
|
v
RFM69 wireless link
|
v
Receiver Moteino
|
v
USB serial
|
v
Python logger
|
v
Raw CSV logs
```

No filtering, estimation, plotting or scientific analysis is performed during acquisition.

Offline Analysis Subsystem

All scientific processing is performed offline.

The offline analysis chain is

```
CSV logs
|
v
Event reconstruction
|
v
Dataset reconstruction
|
v
Validation
|
v
Reference generation
|
v
Residual analysis
|
v
Delta H analysis
|
v
```

```

Statistics
|
v
Plot generation
|
v
Campaign summary

```

6.1.2. Acquisition Chain

Telemetry originates inside the STM32 and propagates through the communication chain until it is recorded as a CSV file.

STM32

The STM32 executes the ADCS algorithms and generates telemetry.

Telemetry is stored inside a fixed-length 96-byte buffer.

Sender Moteino

The sender Moteino receives telemetry from the STM32 through I2C and transmits the reconstructed telemetry through the RFM69 wireless link.

No filtering or scientific processing is performed.

Receiver Moteino

The receiver Moteino collects the transmitted packets, reconstructs the original telemetry and outputs the recovered data through USB serial.

No numerical processing is performed.

The below is a list detailing each block and it's interactions with others in the DAQ chain.

- **Physical system:** This represents the physical test article that is under test. It rotates with an angular velocity set by a stepper motor; it is under the effect of Earth's gravity and magnetic field; and it features a heated horizon simulant at a manually-set angle.
- **RM3100:** The RM3100 measures the magnetic field in its local space. To note is that the RM3100 cannot distinguish between internal and external sources, for which reason care is taken to exclude RM3100 readings while the MTQs are being driven.
- **MLX90640:** The MLX90640s take 24x32 IR images of their surroundings, in this case the heated horizon simulant. They are stored in frames.
- **RM3100 Reader Task:** This is a task that runs inside the STM32 FreeRTOS kernel, and its main job is to read the RM3100 at predesignated times. It collects 3 readings in sequence, together with a timestamp of each reading. It sends these readings to the data logger and the Controller task.
- **MLX90640 Reader Task:** This FreeRTOS task reads the frames from each MLX90640 sensor. It then processes the readings using a temperature mask, and finds the pitch and roll angle through several lookup tables. It then sends these two values to the data logger and the Controller.
- **Controller Task:** This is the main control task of the software. It takes the three readings from the RM3100 Reader Task and, in the Nadir pointing mode, the pitch & roll angles from the MLX90640 Reader Task. Then it runs a control algorithm on these values to produce requested magnetic dipoles from each MTQ. Because the MTQ control method is bang-bang, it then transforms the requested dipoles into a fraction of the allocated actuation time proportional to the fraction of the maximum dipole that each MTQ can provide.
- **MTQ Driver Task:** This task takes the actuation durations from the Controller Task and commands the D1/D2/D3 EN and PH pins accordingly for the second half of the control loop (the allocated actuation time).

- **MPU6050:** The MPU6050 was chosen as a widely used and simple-to-implement gyro and accel combination, used in this project as a way to provide values to externally verify the outputs of the STM32.
- **Data Logger Task:** This FreeRTOS task receives inputs from the other tasks within the FreeRTOS kernel and stores inputs into a designated I2C register in the STM32's memory. Since the STM32 is placed in slave mode with respect to Moteino A, this I2C register is accessed on any I2C read request, and the STM32 will write the entire register on the I2C channel. The I2C register includes values such as last measured xyz magnetic field, calculated angular velocity, an internal timestamp, and MLX90640-calculated pitch and roll.
- **DRV8838 X, Y, Z:** These boards are driven by the MTQ Driver task. They convert PH and EN pins to OUT1 and OUT2 outputs that the MTQs are connected to. At the OUT1 pin of each DRV8838 there is a 1 ohm resistor which, combined with two ADC pins from Moteino A, can help the Moteino measure current.
- **Moteino A:** The Moteino is a small wireless-compatible Arduino board developed by LowPowerLab for IoT applications. In this project, the Moteino physically connected to the STM32 and lying inside the test article, named Moteino A, will collect readings from the MPU6050 and the DRV8838 X,Y,Z outputs. In addition, it will also read the values from the STM32's Data Logger I2C Register. Moteino A stores these values into a struct, and sends it through its RF module in 60-byte packets to Moteino B.
- **Moteino B:** This Moteino board is the RF receiver of the data from Moteino A. It communicates directly with the laptop through a COM serial port.
- **Laptop:** Through its COM serial port, the laptop will collect and save all received data, to be processed later.

Although the current data flows from the STM32 to a laptop over Moteino A and Moteino B-Laptop, it is possible to send commands from the laptop to the STM32. This command flow has not yet been implemented.

In terms of calibration, only the MPU6050 requires calibration. The RM3100 and MLX90640 prepackaged with calibration built-in. Calibration is necessary on the MPU6050 due to floating offsets, and is performed on the MPU6050 using a commonly found implementation from the internet, verified in isolation before integration with the rest of the setup.

6.2. Processing

6.2.1. Event Reconstruction

The telemetry stream contains multiple asynchronous subsystems.

Individual event timestamps are not guaranteed to be generated in chronological order.

Consequently, all events are first reconstructed and only afterwards sorted into a common timeline.

Event Construction

Each telemetry frame generates

- three magnetometer events,
- zero or one MLX event,
- one prediction event.

The timestamp of the first event defines the event origin

$$t = t_{\text{raw}} - t_0. \quad (6.1)$$

All subsequent timestamps are expressed relative to this origin.

Magnetometer Events

Each telemetry frame contains three magnetic measurements

BX1 BY1 BZ1 MM1_TIME

BX2 BY2 BZ2 MM2_TIME

BX3 BY3 BZ3 MM3_TIME

These measurements become three independent magnetic events.

MLX Event

The MLX event contains

ROLL

PITCH

Repeated MLX timestamps are ignored.

Only previously unseen orientation measurements are inserted into the event sequence.

This prevents duplicated attitude measurements from appearing in the reconstructed timeline.

Prediction Event

Prediction events contain

ACT_MS_X

ACT_MS_Y

ACT_MS_Z

PREDICT_BX

PREDICT_BY

PREDICT_BZ

These events represent individual controller iterations.

Chronological Sorting

After all events have been created, the complete event list is sorted by timestamp.

This produces a single ordered timeline

$$t_1 < t_2 < \dots < t_N \quad (6.2)$$

which forms the basis for all subsequent calculations.

6.2.2. Dataset Reconstruction

The framework represents several asynchronous subsystems on a common chronological timeline.

Not every quantity exists at every timestamp.

Missing quantities are therefore represented by

$$\text{NaN}. \quad (6.3)$$

This preserves the original event timing and avoids artificial interpolation between unrelated measurements.

Sparse Arrays

After chronological sorting, every quantity is mapped onto the common event timeline.

For events that do not contain a particular quantity, the corresponding array element is assigned

$$\text{NaN}. \quad (6.4)$$

For example,

timestamp

$$[t_1, t_2, t_3, t_4]$$

Bx

$$[10, \text{NaN}, 11, \text{NaN}]$$

ROLL

$$[\text{NaN}, 5, \text{NaN}, 7]$$

ACT_MS_X

$$[\text{NaN}, \text{NaN}, \text{NaN}, 40]$$

Thus all quantities preserve their asynchronous nature while remaining associated with a common chronological representation.

Dataset Object

All reconstructed quantities are stored inside a Dataset object which acts as the central representation used throughout the framework.

All subsequent calculations operate exclusively on this Dataset representation.

The reconstructed quantities are grouped into three categories.

Event Arrays

Event arrays contain quantities associated with the common event timeline, including

- timestamp,
- magnetic-field components,
- orientation measurements.

Prediction-Loop Arrays

Prediction-loop arrays contain quantities associated with controller iterations, including

- prediction timestamps,
- actuator durations,
- controller-predicted magnetic fields.

Metadata

Metadata describes the experimental conditions associated with the dataset, including

- dataset identity,
- rotation axis,
- angular velocity.

Derived Representations

Several representations are derived from the Dataset object.

These representations provide the views required by subsequent calculations.

Event Timeline

The complete event timeline preserves the chronological order of all measurements.

It is used as the common timeline for subsequent calculations.

Magnetic Sample Sequence

The magnetic sample sequence is derived from the complete event timeline by selecting timestamps containing magnetic measurements.

It is used for

- reference generation,
- outlier removal.

Prediction-Loop Sequence

The framework extracts prediction quantities from the Dataset object into dedicated arrays.

Duplicate prediction timestamps are removed and only the first occurrence is retained.

Consequently,

$$\text{PREDICT_TIME}_i \neq \text{PREDICT_TIME}_j, \quad i \neq j. \quad (6.5)$$

The prediction-loop sequence is used exclusively for ΔH calculations.

The prediction-loop sequence is independent of the magnetic sample sequence and the complete event timeline.

6.2.3. Validation

Validation ensures that the reconstructed dataset is internally self-consistent before reference quantities are generated.

Only validated datasets proceed to subsequent processing stages.

Validation Checks

Datasets are checked for

- minimum sample count,
- array length consistency,
- timestamp monotonicity.

Sample Count

The number of event samples is

$$N = \text{len}(\text{timestamp}). \quad (6.6)$$

The minimum acceptable number of samples is specified by `minimum_valid_samples`.

Datasets satisfying

$$N < \text{minimum_valid_samples} \quad (6.7)$$

are marked invalid.

Timestamp Consistency

Timestamp differences are

$$\Delta t_i = t_{i+1} - t_i. \quad (6.8)$$

Negative values indicate that chronological ordering has been violated.

Datasets containing negative timestamp jumps are marked invalid.

Dataset Rejection

Datasets failing validation are excluded from all subsequent processing.

No reference quantities, statistics or plots are generated from invalid datasets.

6.2.4. Reference Time Origins

Two time origins exist.

$$t = 0$$

Magnetic Origin

The timestamp of the first magnetic sample defines

$$t = 0$$

The magnetic origin is used for

- derotation,
- reference generation,
- reference magnetic-field derivative calculation,
- ΔH calculation.

6.2.5. Rotation Convention

Positive rotation axes are

X

Y

Z

Negative axes

-X

-Y

-Z

are treated as rotations about the corresponding positive axes with negative rotation angle.

6.2.6. Rotation Matrices**Rotation About X**

$$R_X(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \quad (6.9)$$

Rotation About Y

$$R_Y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad (6.10)$$

Rotation About Z

$$R_Z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.11)$$

6.2.7. Angular Velocity Vector**Rotation About X**

$$\omega = \begin{bmatrix} \omega \\ 0 \\ 0 \end{bmatrix} \quad (6.12)$$

Rotation About Y

$$\omega = \begin{bmatrix} 0 \\ \omega \\ 0 \end{bmatrix} \quad (6.13)$$

Rotation About Z

$$\omega = \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} \quad (6.14)$$

6.2.8. Magnetic Reference Generation

Magnetic reference generation reconstructs the magnetic field that would be observed under the assumption that the external field remains constant during the experiment.

The reconstruction pipeline is

```

B_raw
|
v
Derotation
|
v
B_I,est
|
v
Outlier removal
|
v
B_I,clean
|
v
Constant inertial field
|
v
Body-frame reference
|
v
Reference magnetic-field derivative
|
v
Residuals

```

Magnetic Time Origin

Magnetic quantities are referenced to the first magnetic sample.

The magnetic time coordinate is

$$t_{\text{mag}} = t - t_{0,\text{mag}}. \quad (6.15)$$

Magnetic measurements are denoted

$$B_{\text{raw}}. \quad (6.16)$$

Derotation

Derotation transforms body-frame measurements into the external fixed frame.

The body-frame measurements are transformed according to

$$B_{I,\text{est}} = R^T(\omega t) B_{\text{raw}}. \quad (6.17)$$

If the external field is constant, the rotating body-frame measurements collapse into an approximately constant inertial vector.

6.2.9. Outlier Removal

Outlier removal is performed after derotation.

Componentwise Operation

Outlier detection is performed independently on the three magnetic-field components

$$B_x, \quad B_y, \quad B_z. \quad (6.18)$$

Vector magnitude is not used directly.

Therefore vector direction is not explicitly preserved.

Outlier Criterion

Samples are classified using componentwise mean and standard deviation statistics.

The outlier criterion uses a fixed threshold of one standard deviation.

Samples satisfying

$$|x - \mu| > \sigma \quad (6.19)$$

are classified as outliers.

Interpolation

The framework replaces detected outliers using linear interpolation.

The cleaning procedure is

- single-pass,
- non-iterative,
- sample-count preserving.

6.2.10. Constant Inertial Field

Constant-Field Assumption

Under the constant-field assumption,

$$B_I(t) = B_{I,\text{const}} \quad (6.20)$$

and the temporal variation observed in the body frame is attributed entirely to rigid-body rotation.

Constant Field Estimate

The constant inertial field estimate is obtained by applying the mean operation to the cleaned inertial measurements.

$$B_{I,\text{const}} = \text{mean}(B_{I,\text{clean}}) \quad (6.21)$$

Inertial Field Dispersion

The spread of the cleaned inertial measurements about the constant inertial field estimate provides an indication of the quality of the reconstruction.

Small dispersion indicates successful reconstruction of a nearly constant external field.

6.2.11. Body-Frame Reference

The body-frame reference reconstructs the magnetic field corresponding to the estimated constant inertial field.

The constant inertial field is rotated back into the body frame:

$$B_{\text{ref}}(t) = R(\omega t) B_{I,\text{const}}. \quad (6.22)$$

A reference value is generated for every event.

Reference Evaluation Timeline

The body-frame reference is generated over the complete event timeline rather than only at magnetic sample times.

Consequently,

$$B_{\text{ref}}(t) \quad (6.23)$$

is defined over the complete event timeline.

This allows all quantities in the dataset to share a common time axis.

6.2.12. Reference Magnetic-Field Derivative

The analytical magnetic-field derivative is obtained directly from rigid-body kinematics.

Rigid-Body Origin of the Reference Magnetic-Field Derivative

Assuming the external magnetic field is constant in the external fixed frame,

$$\dot{B}_I = 0 \quad (6.24)$$

the apparent variation observed in the rotating body frame arises solely from spacecraft angular velocity.

The transport theorem gives

$$\left(\frac{dB}{dt}\right)_B = \left(\frac{dB}{dt}\right)_I - \omega \times B \quad (6.25)$$

Since

$$\left(\frac{dB}{dt}\right)_I = 0 \quad (6.26)$$

the body-frame magnetic-field derivative becomes

$$\dot{B} = -\omega \times B. \quad (6.27)$$

Advantages of the Analytical Formulation

The analytical formulation is independent of the sampling interval and avoids finite-difference noise amplification.

6.2.13. Residuals

Residuals quantify the disagreement between measured and reconstructed body-frame magnetic fields.

They are defined by

$$\text{Residual} = B_{\text{raw}} - B_{\text{ref}} \quad (6.28)$$

Residual values are computed only at magnetic measurement events.

6.2.14. Magnetic Torque Physics

Instantaneous Torque

Magnetic torque describes the interaction between the magnetic dipole moment and the ambient magnetic field.

The instantaneous torque is

$$\tau = m \times B. \quad (6.29)$$

Angular Momentum Evolution

Angular momentum evolves according to

$$\frac{dH}{dt} = \tau. \quad (6.30)$$

Integral Form

Integrating over the pulse duration gives

$$\Delta H = \int \tau dt = \int m \times B(t) dt. \quad (6.31)$$

For a constant dipole moment,

$$\Delta H = m \times \int B(t) dt. \quad (6.32)$$

This relation forms the basis of the reference model.

6.2.15. Predicted Angular Momentum Change

The predicted model assumes that the magnetic field remains constant during each actuator pulse.

Pulse Durations

Actuator durations are converted from milliseconds to seconds.

For each axis

$$\text{sign}_i = \text{sign}(T_i) \quad (6.33)$$

and

$$\Delta t_i = |T_i|. \quad (6.34)$$

Dipole Direction

Pulse polarity determines the direction of the magnetic dipole.

The magnetic dipole vector becomes

$$m = \begin{bmatrix} \text{sign}_x m_{x,\max} \\ \text{sign}_y m_{y,\max} \\ \text{sign}_z m_{z,\max} \end{bmatrix}. \quad (6.35)$$

Moment Accumulation

Moment accumulation is

$$M_i = m_i \Delta t_i. \quad (6.36)$$

Predicted Angular Momentum Increment

The contribution from each actuator axis is

$$\Delta H_i = M_i \times B. \quad (6.37)$$

The total predicted angular momentum increment is

$$\Delta H_{\text{pred}} = \sum_{i=x,y,z} \Delta H_i. \quad (6.38)$$

6.2.16. Reference Angular Momentum Change

In contrast to the previously described predicted model, the reference model retains the magnetic-field variation caused by spacecraft rotation.

Reference Model

The field variation caused by spacecraft rotation is retained.

Consequently

$$\Delta H_{\text{ref}} = m \times \int B(t) dt. \quad (6.39)$$

The magnetic field is therefore allowed to vary throughout the actuator pulse.

Pulse Windows

Each actuator pulse possesses its own duration

$$\Delta t_i = |T_i|. \quad (6.40)$$

The pulse is centred on the prediction time

$$t_p. \quad (6.41)$$

The integration interval becomes

$$[t_1, t_2] = \left[t_p - \frac{\Delta t_i}{2}, t_p + \frac{\Delta t_i}{2} \right]. \quad (6.42)$$

Independent Pulse Windows

Consequently

$$I_{B,x}, \quad I_{B,y}, \quad I_{B,z} \quad (6.43)$$

are generally evaluated over different time intervals.

The reference angular momentum increment is obtained by summing the individual actuator contributions rather than by integrating over a single common pulse duration.

Analytical Integrated Magnetic Field

The rotating magnetic field is integrated analytically.

Each actuator axis possesses its own integrated magnetic field quantity

$$I_{B,i} = \int_{t_1}^{t_2} B(t) dt. \quad (6.44)$$

because the pulse windows generally differ between axes.

For zero angular velocity,

$$I_B = B_{I,\text{const}} (t_2 - t_1). \quad (6.45)$$

The exact analytical expressions are used for rotations about X, Y and Z.

These expressions provide the exact integral of the rotating field over the pulse duration.

Reference Angular Momentum Increment

The reference increment is

$$\Delta H_{\text{ref}} = \sum_{i=x,y,z} m_i \times I_{B,i}. \quad (6.46)$$

Zero Angular Velocity

When

$$\omega = 0 \quad (6.47)$$

the field does not rotate.

Therefore

$$\Delta H_{\text{ref}} = \Delta H_{\text{pred}} \quad (6.48)$$

apart from differences caused by the controller-provided predicted field.

6.2.17. Angular Momentum Error**Definition**

Angular momentum error quantifies the discrepancy between the predicted and reference angular momentum increments.

It is defined by

$$\Delta H_{\text{error}} = \Delta H_{\text{pred}} - \Delta H_{\text{ref}}. \quad (6.49)$$

6.2.18. Coordinate Frames**External Fixed Frame Definition**

The external fixed frame is defined by the orientation of the body frame at the first magnetic sample.

Consequently

$$t = t_{0,\text{mag}} \quad (6.50)$$

corresponds to

$$\theta = 0. \quad (6.51)$$

Magnetic Quantities

Magnetic quantities are analysed in the body frame.

Quantity	Frame
B_{raw}	Body frame
B_{ref}	Body frame
Reference magnetic-field derivative	Body frame
Residual	Body frame

Inertial Magnetic Quantities

Quantity	Frame
$B_{I,\text{est}}$	External fixed frame
$B_{I,\text{clean}}$	External fixed frame
$B_{I,\text{const}}$	External fixed frame

Angular Momentum Quantities

Angular momentum quantities are initially evaluated in the body frame.

For a prediction event occurring at

$$t_p \quad (6.52)$$

the rotation angle is

$$\theta = \omega t_p. \quad (6.53)$$

The quantities are transformed according to

$$\Delta H_I = R^T(\theta) \Delta H_{\text{body}}. \quad (6.54)$$

Consequently,

Quantity	Frame
ΔH_{pred}	External fixed frame
ΔH_{ref}	External fixed frame
ΔH_{error}	External fixed frame

6.2.19. Angular Diagnostics

Angular diagnostics characterize the geometric relationships between angular momentum increments, the rotation axis and the external magnetic field.

Rotation-Axis Unit Vector

The rotation-axis unit vector

$$\hat{\omega} \quad (6.55)$$

is defined according to the prescribed rotation axis.

For rotations about the X, Y and Z axes, respectively,

$$\hat{\omega} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{\omega} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \hat{\omega} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \quad (6.56)$$

These unit vectors define the rotation-axis direction for the angular diagnostics.

Diagnostic Statistics

For every angular diagnostic quantity, the framework computes

$$\mu(\theta) \quad (6.57)$$

and

$$\sigma(\theta). \quad (6.58)$$

These statistics are available at both dataset and campaign levels.

6.2.20. Two Independent Validation Layers

The framework contains two independent validation layers.

Magnetic Field Reconstruction Layer

The first layer evaluates the reconstruction of the magnetic field.

```

Magnetic reconstruction
|
v
Reference quantities
|
v
Residual diagnostics

```

Angular Momentum Validation Layer

The second layer evaluates the accuracy of the angular momentum prediction.

```

Prediction loops
|
v
Delta Hpred
|
v
Delta Href
|
v
Delta Herror
|
v
Magnitude diagnostics
|
v
Angular diagnostic quantities

```

6.2.21. Statistics

Statistics summarize the quantities generated throughout reconstruction and validation.

Both dataset-level and campaign-level summaries are produced.

Dataset Statistics

Dataset statistics summarize the quantities generated during reconstruction and validation for an individual experiment.

Campaign Statistics

Campaign statistics aggregate information from all experiments and provide campaign-level descriptions of the quantities generated during reconstruction and validation.

Statistical Categories

The framework distinguishes between component, magnitude and directional statistics.

Component statistics characterize individual vector components, magnitude statistics characterize vector magnitudes, and directional statistics characterize angular diagnostic quantities.

In general,

$$|\mu(v)| \neq \mu(|v|). \quad (6.59)$$

Consequently component and magnitude statistics describe different properties of vector quantities.

These statistical categories are applied to residual quantities, inertial magnetic quantities, angular-momentum quantities and angular diagnostic quantities.

6.2.22. Mathematical Definitions

Vector Magnitude

For a vector

$$v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \quad (6.60)$$

its magnitude is

$$|v| = \sqrt{v_x^2 + v_y^2 + v_z^2}. \quad (6.61)$$

Mean

For a sequence

$$x_1, x_2, \dots, x_N \quad (6.62)$$

the mean is

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i. \quad (6.63)$$

Standard Deviation

The standard deviation is

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}. \quad (6.64)$$

Angle Between Two Vectors

For vectors

$$a \quad (6.65)$$

and

$$b \quad (6.66)$$

the angle between them is

$$\theta(a, b) = \cos^{-1} \left(\frac{a \cdot b}{|a||b|} \right) \quad (6.67)$$

where

$$a \cdot b = a_x b_x + a_y b_y + a_z b_z. \quad (6.68)$$

Angles are expressed in degrees.

6.2.23. Export Structure

The framework produces three output layers.

Reference Dataset

A reference dataset is exported for every experiment.

All quantities are represented on a common timeline.

The exported dataset contains the complete reconstruction and validation quantities associated with the corresponding dataset.

These quantities include

- measured quantities,
- reconstructed quantities,
- inertial magnetic quantities,
- reference magnetic-field derivative quantities,
- residual quantities,
- angular-momentum quantities,
- angular diagnostic quantities.

Dataset Statistics

Each experiment produces a statistics table summarizing the quantities generated during reconstruction and validation.

Campaign Summary

Campaign summaries aggregate information from all experiments and provide campaign-level descriptions of the quantities generated during reconstruction and validation.

6.2.24. Plot Generation

The framework generates plots after all valid datasets have been processed.

Global limits are determined using the complete set of valid datasets to ensure consistent visualization throughout the campaign.

Dataset Figures

Dataset figures provide visual representations of the principal reconstructed quantities.

Campaign Comparisons

Common axis limits permit direct comparisons between experiments and operating conditions.

Histogram Diagnostics

Histogram diagnostics provide a statistical view of directional agreement between predicted and reference angular momentum increments.

6.2.25. Overall Processing Chain**End-to-End Pipeline**

```

STM32
|
v
I2C
|
v
Sender Moteino
|
v
RFM69 wireless link
|
v

```

```
Receiver Moteino
|
v
USB serial
|
v
Python logger
|
v
CSV logs
|
v
Event reconstruction
|
v
Dataset reconstruction
|
v
Validation
|
v
Magnetic sample sequence
|
v
Derotation
|
v
Outlier removal
|
v
Constant inertial field
|
v
Body-frame reference
|
v
Reference magnetic-field derivative
|
v
Residuals
|
v
Prediction loop sequence
|
v
Delta Hpred
|
v
Pulse windows
|
v
Analytical field integration
|
v
Delta Href
|
```

```

v
Delta Herror
|
v
Angular diagnostics
|
v
Dataset statistics
|
v
Plot generation
|
v
Campaign summary

```

6.3. Nadir pointing tests

The output of the earth horizon system, its estimated pitch and roll, was captured on the microcontroller and send via the on-board Moteino. The Moteino wirelessly transmits the data to the second Moteino that was connected to a laptop and stored in a CSV file.

Three separate tests were run, the first run had intervals of 10° and an intermittent rotation speed of $40^\circ/\text{s}$. This threw off the algorithm as it is expected to function in very low rotation regimes. Furthermore, the intervals were less accurate due to the stepper motor having 400 teeth each spanning 0.9° , thus it is desirable to have intervals that are divisible by 9.

Hence why, the second test had intervals of 18° and a slower speed of $6^\circ/\text{s}$ between intervals. However, due to the rotational speed still being twelve times that of the nadir pointing regime another final test was ran.

The third test had only a rotation rate between intervals of $1^\circ/\text{s}$ and the intervals themselves were every 9° . The rotation rate and intervals were not further reduced due to time constraints. This test resulted in 40 steps, each with over ten data points. Only this data was processed due to it being the most true-to-life compared to the other tests. All the raw data are stored in CSV files and included in the final SEP package for future studies. All the separate CSV files of the third test are combined into one large excel file for creating the graphs displaying the findings.

6.3.1. Nadir processing

The first processing step concerned itself with the stepper motor's default rotation. It was in the negative Z-axis. So instead of going from 0° to 360° it went from 0° to 351° and down.

Thereafter, the initial offset of the test article to the horizon simulant had to be accounted for. The horizon simulant rested upon stacked plastic and cardboard boxes which made the simulant not sitting level. The rest position, which should have corresponded to zero pitch and roll, resulted in readings of 14° roll and 7° pitch. Thus, all the true reference measurements were shifted to account for this offset. This data reference set is called "true corrected ...".

The recorded time also had to be corrected as the STM32, when powered on, starts incrementing its timestep. Thus, after having ran two tests and waiting in between the tests for the horizon simulant to have achieved a stable temperature, the first data point had a recorded time of 4063 seconds. This was solved by subtracting 4063 from each recorded time step.

The data was also filtered for identical time steps and their readings. This happened due to the data rate of the Moteino being faster than the STM32's data creation. Furthermore, the earth horizon sensing algorithm has a safe guard build in when no sensors are trusted, so have not met the threshold of five conformation frames. When this occurs, it transmits its last known attitude. However, a bug in the main operating software transmits 0° pitch and roll when the amount of active sensors drop to zero. This can be seen happening a couple of times throughout the test in Figure 6.2. These instances have been removed from the final test results as the hypothesis

regarding requirement **EHS-02** is only concerned about the functioning of the EHS algorithm.

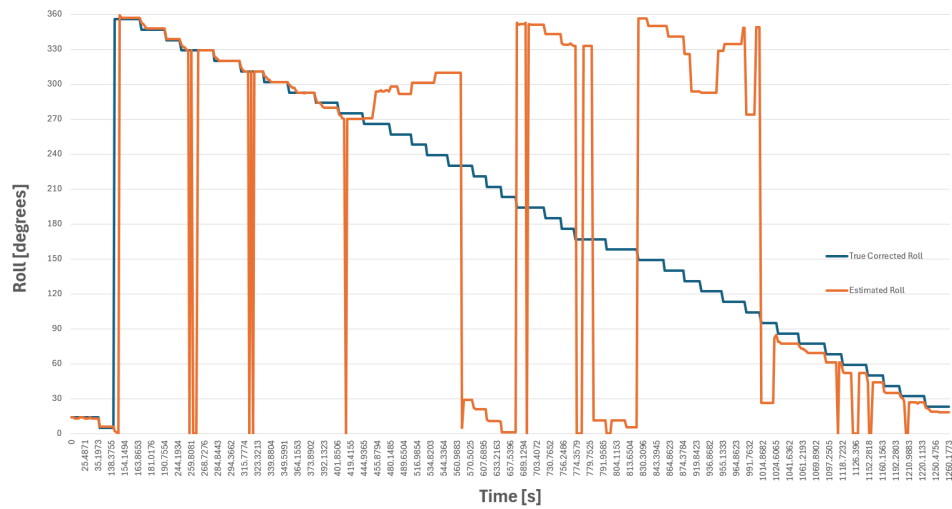


Figure 6.2: Estimated roll (orange) vs true corrected roll (blue) over 360° with zeroing bugs present

7

Test Plan

This chapter presents the overall test plan for the experimental verification of the ADCS breadboard model. The plan outlines the activities required to prepare, execute, and analyse the de-tumbling and nadir pointing experiments described in previous chapters.

The testing process follows the course structure and is divided into three phases:

- **D1 – Verification Specification:** definition of the experiment concept, the verification strategy, and the high-level test plan.
- **D2 – Test Preparation:** construction and validation of the experimental setup, definition of detailed procedures, and execution of a dry-run test.
- **D3 – Final Test:** execution of the final experiments, analysis of the results, and verification of the requirements.

In the present phase (D1), only the high-level structure of the test activities and the expected timeline are defined. Detailed experimental procedures, hardware connections, personnel allocation, and safety measures will be developed during the D2 phase once the experimental setup has been finalized and validated.

7.1. High-Level Test Plan

This section provides a high-level overview of the planned testing activities and milestones for the project. The objective is to outline the sequence of activities required to prepare and execute the experimental verification of the ADCS algorithms implemented on the breadboard model.

The verification focuses on two experiments, listed below. In both, the MTQ commands given by the STM32 are measured to evaluate the outputs of the algorithms.

- de-tumbling verification using the B-dot control algorithm.
- Nadir pointing verification using the Earth Horizon Sensors and magnetometer measurements.

Both experiments will be conducted using the experimental setup described in Chapter 2, where the breadboard model is mounted on a motorized test jig capable of generating controlled angular motion or fixed orientations. Each test case will be executed for approximately 60 seconds (detumbling) or 10 seconds (pointing) in order to obtain a sufficiently large number of samples for statistical evaluation.

The high-level testing activities are summarized below:

- Definition of the experimental verification concept and requirements (D1 phase).
- Finalization of the breadboard model design and test jig configuration.
- Development of the experimental procedures and measurement strategy.

- Unit testing of system components.
- Full integration and assembly the test setup.
- Execution of dry-run experiments to validate the measurement process.
- Execution of the final de-tumbling and nadir pointing experiments.
- Processing and analysis of the recorded data.
- Verification of the requirements and assessment of the experimental hypothesis.

Table 7.1 summarizes the planned activities and expected deliverables for each project phase.

Table 7.1: High-level test plan and milestones

Phase	Main Activities	Deliverables
D1	Definition of verification approach, experiment concept, and measurement strategy. Identification of the experimental setup and definition of the high-level test plan.	Verification specification report and initial test plan.
D2	Assembly and validation of the experimental setup, calibration of sensors, development of detailed test procedures, and execution of dry-run tests to verify measurement capability.	Detailed test procedures, validated setup, and dry-run test results.
D3	Execution of the final de-tumbling and nadir pointing experiments, data processing and analysis, and verification of requirements based on the experimental results.	Final test results, requirement verification, and project conclusions.

Figure 7.1 illustrates the planned timeline of the experimental activities across the three project phases.

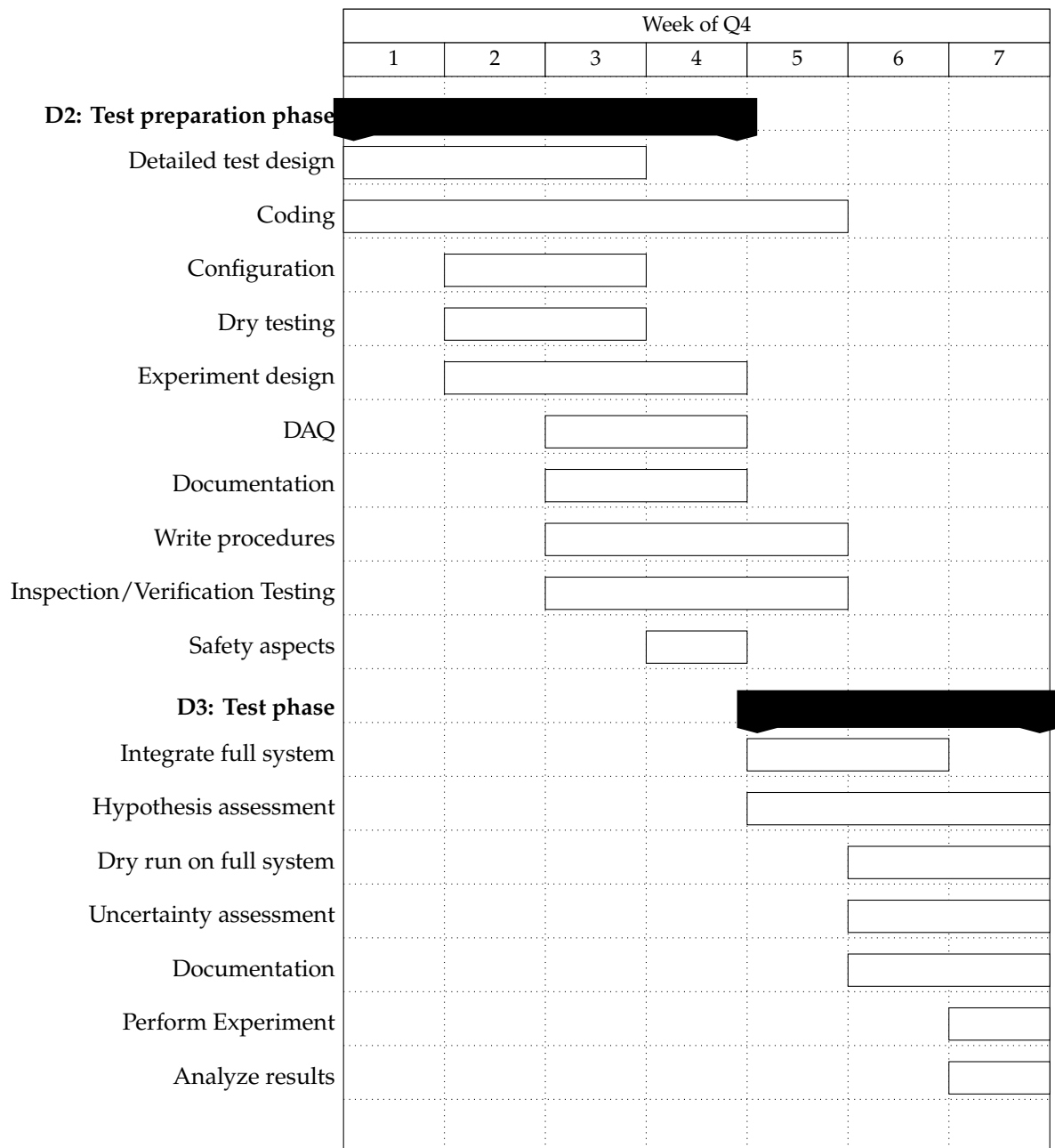


Figure 7.1: High-level timeline of the experimental test campaign. Black lines indicate project phase.

7.2. Test Plan

The test plan is configured in four different procedures; preparation, powering up, testing, finishing up.

The first procedure describes the preparation i.e. the acquisition of components and thereafter the assembly.

Component	Specification	Quantity
Microcontroller	STM32L476RG	1
Magnetometer	RM3100	1
MTQ Drivers	DRV8838	3
MTQs	Manufactured by us	3
Voltage Regulator	From Tinytronics	1
MPU	MPU6050	1
Communication devices	Moteino-USB	2
Stepper motor	NEMA 17	1
Motor driver	DRV8825	1
IR sensors	MLX90642ESF-BCA-000-TU/SP	4
Lab power supply	Tenma 72-2645	1
Heating elements	RNP-10SAR500JZ03	18
Temperature sensor	TPM36	1
Battery	2s LiFePo4	1
USB cables	Type A/B	5
Jumper wires	Dupont F & M	50
Resistors	2 Ω resistors	4
Multimeter	N/A	1

Table 7.2: List of all electrical components needed for the test

Component	Description	Quantity
Jig support	Support for the rotating axis of the test model	4
Gut-Module casing		1
Brain-Module		1
EHS support	Slender casing to house the 4 IR sensors at the correct position & orientation from one-another	2
Brass rods	Non conductive rods to connect all the casings	4
Aluminum rod	Non conductive rods to connect test model to stepper motor	2
Aluminum coated circular plate	horizon simulant	1
Horizon simulant support	Small stands where the horizon simulant can lay atop off	4
MTQs (X, Y)	Identical magnetorquers, each oriented towards X- or Y-direction	2
MTQs (Z)	Magnetorquer oriented in the Z-direction	1

Table 7.3: List of all structural components needed for the test

7.2.1. Preparation

1. Acquire all the items required
 - (a) Acquire all electrical components mentioned in Table 7.2
 - (b) Acquire all structural items mentioned in Table 7.3
 - (c) Acquire laptop
2. Wire all electrical components of the test model as defined in Figure 2.1 and shown in Figure 7.2 **except for the power supply!**
3. Place electronics in their structural casings

4. Construct test model by connecting the casings in order via the brass rods as shown in Figure 2.2
5. Assemble supporting test jig as described in subsection 2.1.2.
6. Place stepper motor on its support
7. Connect the motor driver, to stepper motor
8. Connect aluminum rods to the test model
9. Connect the aluminum rod to the stepper motor
10. Verify functionality with slight rotation of stepper motor
11. Construct horizon simulant
 - (a) Attach all heating elements onto the backside of the aluminum circular plate as described in subsection 2.1.3
 - (b) Wire all heating elements in parallel
 - (c) Attach temperature sensor onto the backside of the aluminum plate
 - (d) Connect the temperature sensor and connected heating elements to the arduino
 - (e) Place the horizon simulant onto its supports

7.2.2. Powering up

1. Powering on the test model
 - (a) Use a multimeter to ensure non-zero resistance over each wire of the converter
 - i. If resistance is zero, investigate short-circuit possibilities before proceeding
 - ii. Else proceed
 - (b) Connect Moteino B to laptop
 - (c) Connect battery to all peripheral components in the test model as shown in Figure 7.2
 - (d) Check that the data pipeline is active and sending data
2. Test horizon simulant
 - (a) Wire Arduino to the computer
 - (b) Wire heating elements to the lab power supply
 - (c) Run `arduino_thermal_control.ino` to check proper functionality
 - i. Press space-bar to start heating process
 - ii. Confirm that 30 V and 3.33 A is flowing from the lab power supply into the system
 - iii. **If this is not the case, a connection is missing, press space-bar to stop the electricity flow and recheck all the cable connections**
 - iv. Monitor temperatures shown in serial monitor of the Arduino IDE
 - v. Show the validity of the PID controller by confirming that the temperature stabilizes at 44°C
 - vi. Press space-bar to stop heating process and cut-off power from heating system
 - vii. Wait until the temperatures shown are below 30°C to safely disassemble system

7.2.3. Set up

1. Powering on the test model
 - (a) Use a multimeter to ensure non-zero resistance over each wire of the converter
 - i. If resistance is zero, investigate short-circuit possibilities before proceeding
 - ii. Else proceed
 - (b) Connect Moteino B to laptop

- (c) Connect battery to all peripheral components in the test model as shown in Figure 7.2
- (d) Run `communication_check.py` to validate communication between Moteino A & B
- (e) Run `Component_check.py` to validate communication between all components and the microcontroller
2. Calibrate sensors (omitted due to time constraint)
 - (a) Arrest movement of test model
 - (b) Run `MPU6050_Calibration.py` to calibrate the MPU

7.2.4. Functionality Test

1. Set stepper motor rotation speed to $0.5^\circ/\text{s}$ via the laptop script, `arduino_motion_control.ino`
2. Run `arduino_motion_control.ino` script
3. Record Moteino B's packets of variables specified in ?? into CSV file
4. Stop `arduino_motion_control.ino` script
5. Check electrical parts for faults

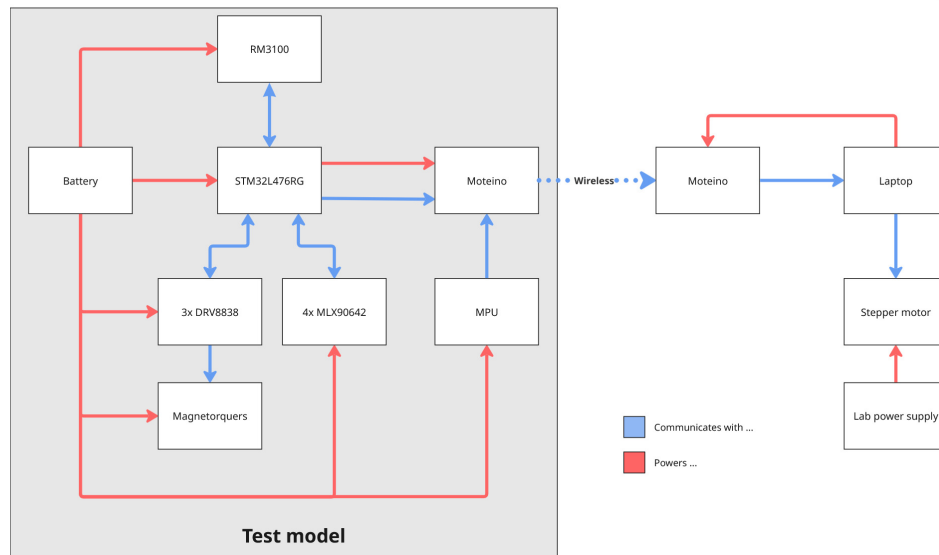


Figure 7.2: Communication and wiring diagram of the test set-up

7.2.5. Testing

1. X°/s De-tumble Test
 - (a) Set stepper motor rotation speed to X°/s via the laptop script, `arduino_motion_control.ino`
 - (b) Run `arduino_motion_control.ino` script
 - (c) Wait 1 minute for test model to reach correct rotational rate
 - (d) Record Moteino B's packets of variables specified in ?? into CSV file
 - (e) Stop `arduino_motion_control.ino` script
 - (f) Wait 1 minute for the test model to wind down
 - (g) Check electrical parts for faults
 - (h) Repeat for other angular velocities if desired
2. $0.5^\circ/\text{s}$ Nadir pointing Test
 - (a) Reorient the setup to rotate the test article around its Z axis.

- (b) Install horizon simulant under set-up under test jig.
- (c) Use `arduino_thermal_control.ino` to command the horizon simulant to the right temperature by pressing space-bar.
- (d) Wait until horizon simulant reaches desired temperature & monitor temperatures.
- (e) Manually rotate the test jig to zero roll, zero pitch.
- (f) Use `arduino_motion_control.ino` to lock the stepper motor in place.
- (g) Let article rest for 10 seconds.
- (h) Record data for 10 seconds and store in unique CSV file.
- (i) Rotate stepper motor with a desired velocity to a specific incremented roll angle.
- (j) Repeat from step (g) until a full rotation is completed.
- (k) Stop `arduino_thermal_control.ino` script by use of space-bar.
- (l) Stop `arduino_motion_control.ino` script
- (m) Wait 1 minute for horizon simulant to cool off
- (n) Remove horizon simulant from test jig
- (o) Check electrical parts for faults

7.2.6. Finishing up

1. Disconnect battery to all test model components
2. Disconnect lab power supply from horizon simulant
3. Ensure all data from Moteino B has been saved in a CSV file
4. Disconnect Moteino B from laptop
5. Disconnect test model from stepper motor
6. Disconnect all the wiring of horizon simulant & test model
7. Store components into their respective casings for future use

7.3. Personnel required

For the execution of the test, multiple people are needed to perform according to plan. The below list explains who does what:

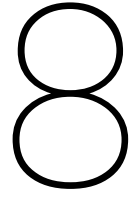
- **Data collector:** The data collector is the person that operates the laptop connected to the receiving Moteino. This person is also responsible for operating the stepper motor and sending commands to the ADCS, if applicable. Their main role is to ensure that the laptop receives and stores all relevant data, as well as supervising the telemetry and system health electronically.
- **Electronics Technician:** The electronics technician is responsible for checking wiring before and after assembly. This person is also tasked with adjusting the software of the ADCS, if needed. The electronics technician will work together with the hardware technician to ensure a complete and safe test setup.
- **Hardware technician:** The hardware technician is responsible for the hardware integration and supervision of the test article. They are to direct assembly and inspect for faults. The hardware technician will work together with the electronics technician to ensure a complete and safe test setup.
- **Test leader:** The test conductor sets the pace for the experiment. They are to follow pre-written procedures and watch out for unsafe situations. They are the final decision maker if need arises, and the other personnel comply to their directions.

To note is that, although the Test leader is mainly responsible for overall test safety, every member is to bear in mind that personal safety is their responsibility too.

Event	Risk	Consequence	Prob.	Sev.	Sore	Mitigation	PM Prob.	PM Sev.	PM Score
Project is on a tight timeline	Rushed electronics assembly	Wiring error, component damage	5	4	20	(1) Schedule at least 3 free days for testing, even taking into account all tasks. This is to leave enough time margin to allow testing to be performed safely. (2) Allow one experienced and one inexperienced team member to check system wiring against pre-designed wiring diagram before testing	2	2	4
High speed de-tumbling testing	Something touches the spinning article (hand, object, etc)	Component damage	2	4	8	Have all participants stand at least 1.5m away from the test article while testing. Ensure no debris or nearby objects are capable of falling onto or interacting with the test article	1	3	3
High speed de-tumbling testing around x/y	Unsecured components fall out of the test article	Component damage, risk to bystanders	3	5	15	Apply rubber bands or other suitable means of preventing modules from sliding off the brass rods. Ensure that all internal components are secured either with tape or via packing. Rattle the test article lightly to check for loose parts, secure if any.	1	3	3
High speed de-tumbling testing around x/y	Unbalanced article	Test article starts jumping out of its guides, falls off and gets damaged/hits someone	3	5	15	Secure down the test jig stands via weight or adhesive. Use guards on top of the Alu rod rail to stop model from potentially flying. Measure article CG before the test and add weight to balance it.	1	2	2
Testing for a long time	Personnel get tired	Human error; anything might happen, from missing procedures to handling mistakes	4	4	16	Schedule breaks. If more than one person still feels too tired, suspending testing until everyone is well rested.	2	2	4

Continued on next page

Event	Risk	Consequence	Prob.	Sev.	Score	Mitigation	PM Prob.	PM Sev.	PM Score
Stepper motor is used for sudden changes in speed	Possible physical or current shock	Possible physical shock that induces damage or topples test stands; possible current shock that damages driver	3	5	15	Slowly increase stepper motor speed via small increments. Use capacitors to smoothen the current jumps	2	2	4
Batteries and power supplies are connected to custom electronics	Wiring error	Shortcircuit. Component damage, or personnel electric shock	5	5	25	Double check wiring before plugging in any power sources. Check resistance with a multimeter and connect power source only if resistance is higher enough for safe current, given voltage (50 mA). Else, investigate source of low resistance until resolved or proven safe beyond doubt	2	2	4
Personnel are hungry, thirsty, or need caffeine	Food, coffee or other substances might be brought into the lab as "it's fine, I'll be careful"	Personnel not actually careful. Possible spillage, damaging not just test article but also tables, carpets etc	4	5	20	Do not allow food or drinks into the lab under any circumstances. Eat and drink outside	1	1	1



Test Results

This chapter presents the results obtained during the dry-run and final test campaigns. The dry-run campaign was used to validate the test setup, reconstruction framework, and horizon-sensing pipeline prior to final testing. The final campaign evaluates the detumbling and nadir-pointing functionality using the measured quantities defined in Section ?? . Requirement-level assessments are presented for the applicable verification cases.

8.1. Dry Run

The dry-run campaign was conducted prior to final testing to validate the acquisition chain, reconstruction framework, and horizon-sensing functionality. The results of these validation activities are presented in the following subsections. Compared to the planned experiment, the dry run was conducted without the flight-representative Earth Horizon Sensor hardware and with a readily available stepper motor and driver that are suboptimal in both torque and angular accuracy.

The objective of the dry-run campaign was not requirement verification but validation of the acquisition chain, reconstruction framework, and horizon-sensing methodology. Requirement-level verification is performed exclusively using the final test campaign results presented in Section 8.2.

8.1.1. Horizon Simulant Results

Several dry-run tests were performed to validate the horizon-sensing pipeline. This section focuses on the static and dynamic tests used to verify horizon-mask generation, sensor communication, and attitude-estimation functionality.

The data stream was read out from the STM32 via a special script. It was only used for unit testing as the needed data throughput of four camera feeds is very high. An instance of this type of test is shown in Figure 8.1. It can be seen that the system has clear communication with all four cameras and that the masks are correctly created.

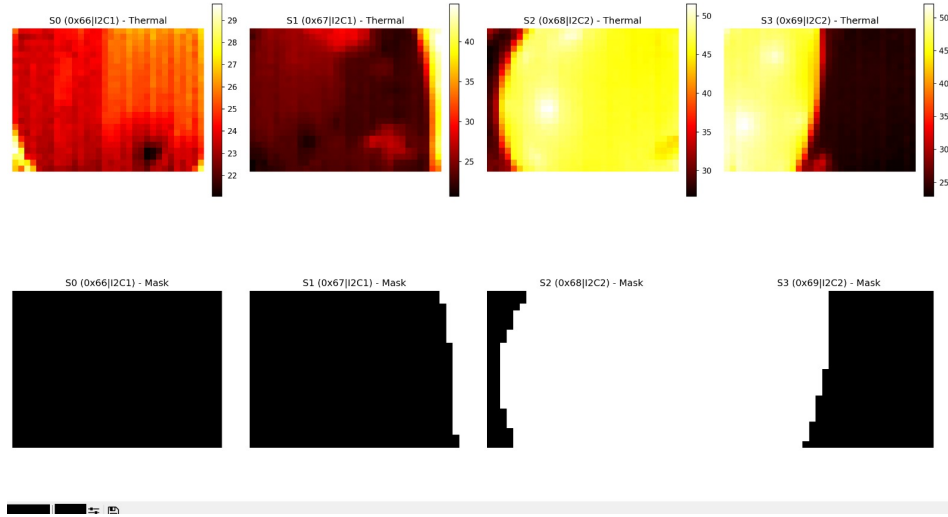


Figure 8.1: Camera feed and corresponding masks of all four sensors

After confirming the sensor data stream, a different approach was used for further testing by reading out the amount of valid sensors, the estimated attitude state of each sensor, and the final weighted estimation. A snippet of such a CSV file is shown in Figure 8.2; here the switch from no trusted sensors and the confirmation frames incrementing can be seen.

After five frames, the three sensors achieve active status and can weigh in on the final attitude estimation of the whole system. However, to get an even more intuitive sense of the satellite's orientation, the output is simulated using `image_generation_V4`. This is shown in Figure 8.3.

Timestamp	System Pitch	System Roll	Active ID	S0_Pitch	S0_Roll	S0_Confirms	S0_Grace	S1_Pitch	S1_Roll	S1_Confirms	S1_Grace	S2_Pitch	S2_Roll	S2_Confirms	S2_Grace	S3_Pitch	S3_Roll	S3_Confirms	S3_Grace	
30:33.5	3.3	15.22	-1	180	96	5	4	0	0	0	0	6	3	284	5	4	15	290	5	4
30:33.7	3.3	15.22	-1	180	96	5	5	0	0	0	0	6	3	284	5	5	170	296	5	5
30:34.0	3.3	15.22	-1	180	96	0	6	0	0	0	0	6	0	287	0	6	2	291	0	6
30:34.2	3.3	15.22	-1	180	96	1	0	0	0	0	0	6	0	287	1	0	2	291	1	0
30:34.5	3.3	15.22	-1	180	96	2	0	0	0	0	0	6	0	287	2	0	2	291	2	0
30:34.7	3.3	15.22	-1	180	96	3	0	0	0	0	0	6	0	287	3	0	2	291	3	0
30:35.0	3.3	15.22	-1	180	96	4	0	0	0	0	0	6	0	287	4	0	2	292	4	0
30:35.2	1.52	292	2	180	98	5	0	0	0	0	0	6	1	290	5	0	2	294	5	0
30:35.5	1.2	292.6	3	180	101	5	1	0	0	0	0	6	359	291	6	0	3	294	6	0
30:35.7	357.01	294	3	182	101	5	2	0	0	0	0	6	349	294	7	0	2	294	7	0
30:36.0	0.64	292.59	3	180	101	5	3	0	0	0	0	6	359	291	8	0	2	294	8	0
30:36.2	0.64	292.59	3	180	101	5	4	0	0	0	0	6	359	291	9	0	2	294	9	0
30:36.5	0.66	292.61	3	180	101	5	5	0	0	0	0	6	359	291	10	0	2	294	10	0
30:36.7	357.75	295.67	3	180	101	0	6	0	0	0	0	6	351	295	11	0	1	296	11	0
30:37.0	358.72	297	3	180	102	0	6	0	0	0	0	6	353	297	12	0	0	297	12	0
30:37.2	2.19	297.83	3	180	104	0	6	0	0	0	0	6	353	297	13	0	4	298	13	0
30:37.5	2.09	297.82	3	182	104	0	6	0	0	0	0	6	353	297	14	0	4	298	14	0
30:37.7	0.71	298.7	3	182	104	0	6	0	0	0	0	6	353	297	15	0	2	299	15	0
30:38.0	3.99	299.91	3	182	107	0	6	0	0	0	0	6	26	299	16	0	2	300	16	0
30:38.2	3.79	299.92	3	182	107	0	6	0	0	0	0	6	26	299	17	0	2	300	17	0
30:38.5	4.59	300.85	3	182	107	0	6	0	0	0	0	6	26	299	18	0	3	301	18	0
30:38.7	3.64	303.85	3	178	107	0	6	0	0	0	0	6	26	299	19	0	3	304	19	0
30:39.0	3.32	303.92	3	180	110	0	6	0	0	0	0	6	26	299	20	0	3	304	20	0
30:39.2	3.32	303.92	3	180	110	0	6	0	0	0	0	6	26	299	21	0	3	304	21	0
30:39.5	4.28	304.91	3	180	110	0	6	0	0	0	0	6	26	299	22	0	4	305	22	0
30:39.7	2	306	3	179	112	0	6	0	0	0	0	6	0	0	22	1	2	306	23	0
30:39.9	2	307	3	181	113	0	6	0	0	0	0	6	0	0	22	2	2	307	24	0
30:40.2	2	307	3	181	113	0	6	0	0	0	0	6	0	0	22	3	2	307	25	0
30:40.4	4	307	3	181	113	0	6	0	0	0	0	6	0	0	22	4	4	307	26	0
30:40.7	5	308	3	181	113	0	6	0	0	0	0	6	0	0	22	5	5	308	27	0
30:40.9	4	308	3	181	115	0	6	0	0	0	0	6	0	0	0	6	4	308	28	0
30:41.2	4	309	3	181	115	0	6	0	0	0	0	6	0	0	0	6	4	309	29	0
30:41.5	3	310	3	180	117	0	6	0	0	0	0	6	0	0	0	6	3	310	30	0
30:41.7	4	311	3	180	117	0	6	0	0	0	0	6	0	0	0	6	4	311	31	0
30:42.0	4	311	3	179	118	0	6	0	0	0	0	6	0	0	0	6	4	311	32	0
30:42.2	4	311	3	179	118	0	6	0	0	0	0	6	0	0	0	6	4	311	33	0
30:42.5	3	311	3	179	118	0	6	0	0	0	0	6	0	0	0	6	3	311	34	0
30:42.7	4	312	3	181	119	0	6	0	0	0	0	6	0	0	0	6	4	312	35	0
30:43.0	4	312	3	181	119	0	6	0	0	0	0	6	0	0	0	6	4	312	36	0

Figure 8.2: Read out of all individual sensors, their status and the combined final estimated attitude

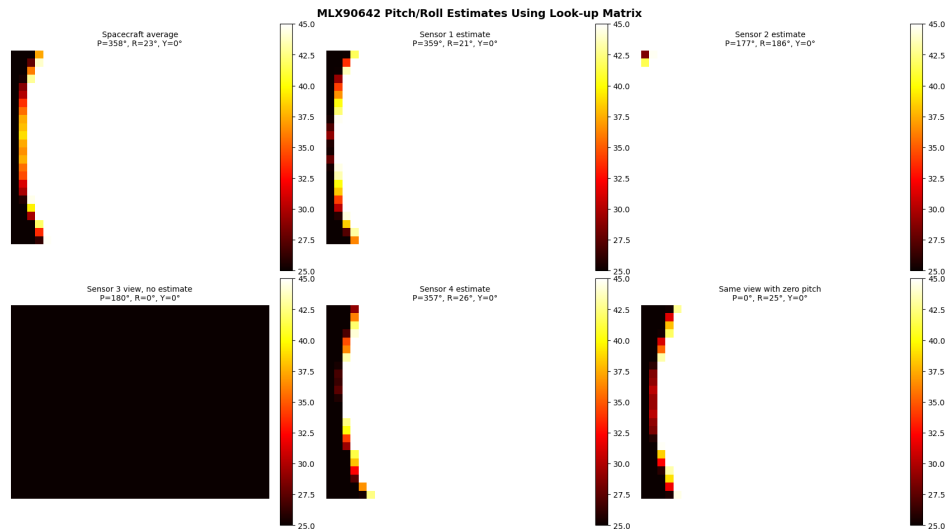


Figure 8.3: Weighted averaged attitude due to multiple valid sensor readings with reference

In Figure 8.3 a problem can already be identified. Sensor 2 estimates a pitch of 177° and roll of 186° ; however, that could just as well be 6° roll and pitch. This brings us to the degeneracy present in attitude states. A clear example is given in Figure 8.4. A rotation of 180° is according to the sensor identical to no rotation. Furthermore, $+80^\circ$ and $+270^\circ$ (i.e., -80°) also have similar horizon views. This can trip up the sensor and is seen to be a problem in the final results as well. It mainly arises at 180° roll and 0° pitch as this also perfectly corresponds to 0° roll and 180° pitch. This was tried to be mitigated by use of a history check consisting of the 3D unit vector of the previous attitude. This was deemed not strong enough, and a moving average of the last three states is proposed but was not able to be tested due to time constraints.

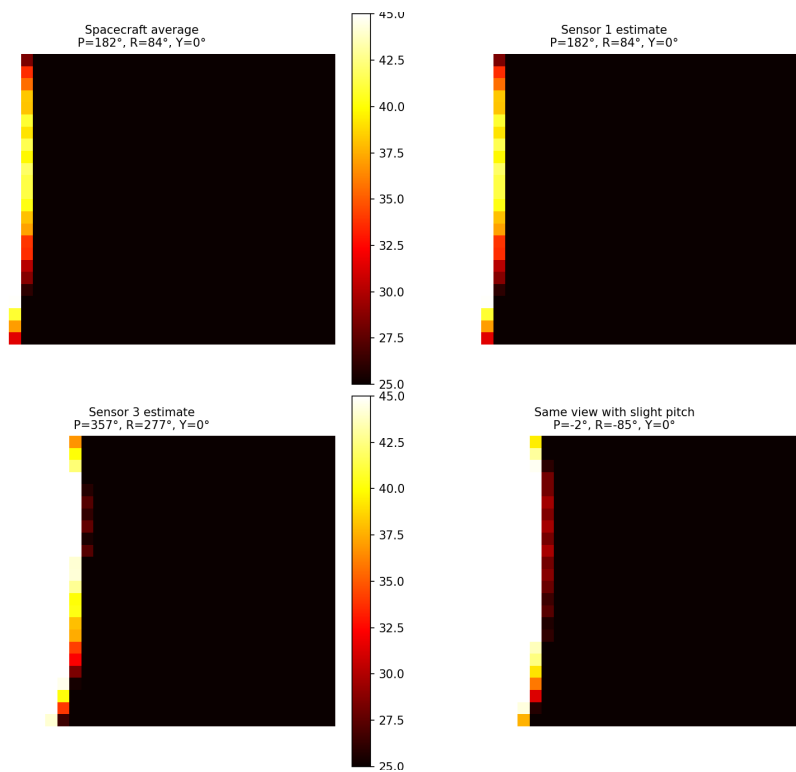


Figure 8.4: Degeneracy of roll and pitch states

Lastly, due to the limited tests able to be performed no hypothesis was written for requirement **ACT-4** regarding the sun in view scenario. However, this was still tested in the dry run tests. The team was not able to find an appropriate laser, thus a rudimentary hot object was used: a mug filled with hot water. This created a large visible IR signature, which is way larger than the star radian angle that the Sun would have. Then a static attitude was used and the hot object was brought into view of the sensors.

The result can be seen in Figure 8.5a and Figure 8.5b. The flood fill algorithm works as expected: it does not take the hot object's pixels into consideration as it is the smaller blob in the picture when it is detached from the horizon. However, when the hot object coincides with the horizon, the algorithm considers it part of the horizon and assimilates it into the horizon mask. This would only occur for 7.5 seconds of the roughly 90 minutes of orbital time, so it would not cause a major error[10].

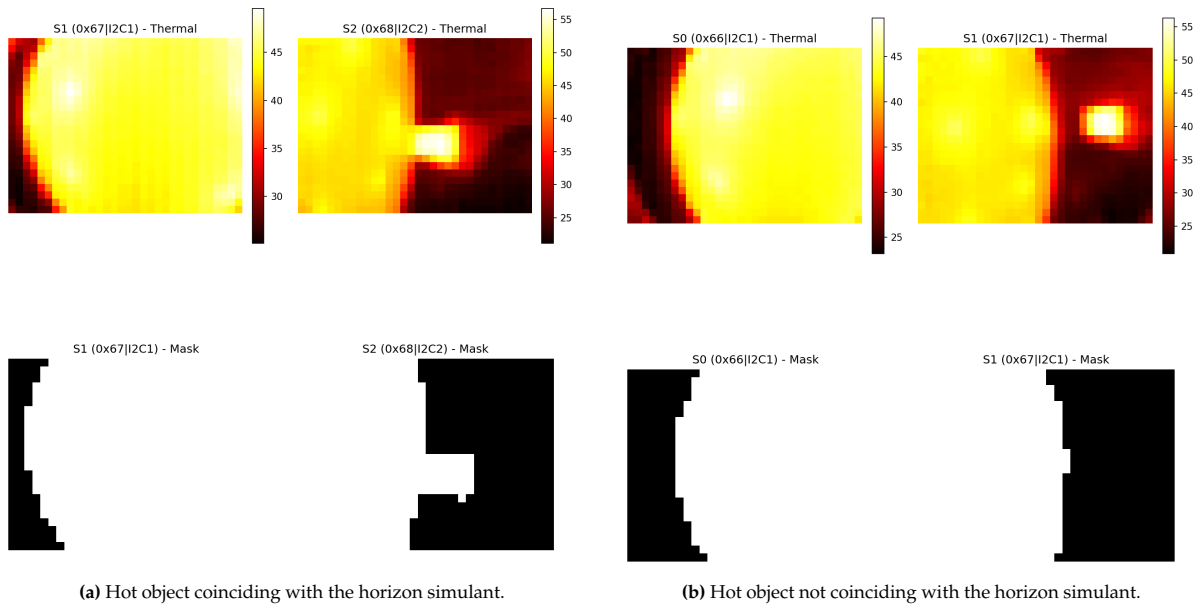


Figure 8.5: Sensor outputs and masks under different hot object conditions.

The dry-run tests, therefore, confirmed the expected behavior of the horizon-extraction algorithm in the presence of a strong infrared source. Additional Sun-sensing functionality could potentially improve attitude-estimation robustness, but such functionality was beyond the scope of the present project.

Dry-Run Observations

- the horizon simulant was not perfectly flat, introducing a small geometric bias in the apparent horizon position;
- the distance between the cameras and the simulant varied slightly between test sessions, affecting the apparent horizon geometry;
- the substituted stepper motor and driver were not representative of the final test hardware and are suboptimal in both torque and angular accuracy;

These limitations should be considered when interpreting the final test results.

8.1.2. Magnetic Field Reconstruction Validation

The dry-run campaign was used to characterize the accuracy of the magnetic-field reconstruction framework used throughout the remainder of this chapter. For each run, the measured body-frame magnetic field was compared against the reconstructed reference field using the residual metric defined in Chapter 6.

In addition to the residual analysis, the reconstructed fixed-frame magnetic-field magnitude was monitored throughout the campaign. Across all tested angular velocities, the reconstructed field magnitude remained relatively constant, with mean values between 41.9 μT and 43.4 μT . The corresponding standard deviation ranged from 0.73 μT to 1.26 μT , indicating that the reconstructed fixed-frame field remained stable during testing.

The magnetic-field reconstruction residuals discussed in this section are summarized in the accompanying analysis. No dedicated summary table is included because the residual statistics are used only to establish reconstruction uncertainty bounds for the subsequent angular-momentum validation.

Table 8.1: ACT-1.2 assessment using the reconstructed reference angular-momentum increment

ω [$^\circ \text{s}^{-1}$]	μ_θ [deg]	σ_θ [deg]	$\mu_\theta - \sigma_\theta$ [deg]
1	67.29	62.97	4.32
2	84.30	68.46	15.84
5	81.13	66.60	14.52
10	51.68	55.60	-3.92
20	27.64	36.40	-8.76
30	28.30	41.63	-13.34
60	7.52	2.93	4.58
90	6.99	1.17	5.82
120	7.33	1.30	6.03
180	7.51	1.67	5.83

For imposed angular velocities up to 10 $^\circ \text{s}^{-1}$, the mean residual remained approximately constant at 1.7 μT . Above this range, larger residuals were observed, reaching 3.55 μT at 20 $^\circ \text{s}^{-1}$, 8.39 μT at 30 $^\circ \text{s}^{-1}$, and a maximum of 12.59 μT at 120 $^\circ \text{s}^{-1}$. The corresponding residual standard deviations increased from approximately 0.6 μT at low angular velocity to 6.90 μT at 120 $^\circ \text{s}^{-1}$.

The datasets corresponding to angular velocities of 90 $^\circ \text{s}^{-1}$, 120 $^\circ \text{s}^{-1}$, and 180 $^\circ \text{s}^{-1}$ contain approximately 200 prediction events, whereas the lower-angular-velocity datasets contain approximately 1000 events. The resulting statistics are therefore based on different dataset durations and sample counts.

The residual statistics discussed above define the uncertainty bounds associated with the magnetic-field reconstruction framework. Since the reconstructed magnetic field is used directly in the angular-momentum reconstruction, these uncertainties propagate into the quantities presented in subsection 8.1.3. Consequently, the residual levels reported in this section should be considered when interpreting the detumbling and nadir-pointing results presented later in this chapter.

8.1.3. Angular Momentum Reconstruction Validation

The dry-run campaign was also used to characterize the uncertainty associated with the angular-momentum reconstruction framework. Two formulations were evaluated: the predicted angular-momentum increment, ΔH_{pred} , and the reference angular-momentum increment, ΔH_{ref} . The predicted formulation is derived directly from the commanded actuation and reconstructed magnetic field, whereas the reference formulation incorporates the measured body motion throughout the actuation interval.

Agreement between both formulations was evaluated using the magnitude ratio

$$\frac{|\Delta H_{\text{pred}}|}{|\Delta H_{\text{ref}}|}$$

and the angular separation between the two vectors. The resulting validation metrics are summarized in Table 8.2.

Table 8.2: Angular-momentum reconstruction validation metrics

ω [$^{\circ}$ s $^{-1}$]	n	$ \overline{\Delta H_{\text{pred}}} $	$ \overline{\Delta H_{\text{ref}}} $	$ \overline{\Delta H_{\text{err}}} $	$ \Delta H_{\text{pred}} / \Delta H_{\text{ref}} $	$\theta_{\Delta H}$ [deg]
1	1085	0.0178	0.0048	0.0163	3.71	67.27
2	916	0.0028	0.0010	0.0029	2.73	85.08
5	916	0.0067	0.0020	0.0066	3.30	81.61
10	990	0.0063	0.0013	0.0057	4.68	54.01
20	1001	0.0057	0.0013	0.0046	4.32	27.46
30	1001	0.0048	0.0015	0.0035	3.23	27.47
60	1003	0.0032	0.0018	0.0014	1.78	5.71
90	200	0.0658	0.0514	0.0159	1.28	4.05
120	209	0.0624	0.0538	0.0115	1.16	4.47
180	202	0.0353	0.0406	0.0059	0.87	3.59

Substantial disagreement between the predicted and reference formulations is observed at low angular velocity. For imposed angular velocities between 1 and 10 $^{\circ}$ s $^{-1}$, the mean angular separation ranges from 54.0 $^{\circ}$ to 85.1 $^{\circ}$, while the predicted angular-momentum magnitude exceeds the reference magnitude by factors between 2.73 and 4.68. Large standard deviations of the directional error (55.8 $^{\circ}$ to 64.5 $^{\circ}$) indicate considerable variability within these datasets.

Although the low-angular-velocity datasets exhibit substantial variability, the agreement between both formulations improves markedly for angular velocities above approximately 20 $^{\circ}$ s $^{-1}$. The mean angular separation decreases from 67.3 $^{\circ}$ –85.1 $^{\circ}$ for angular velocities between 1 and 5 $^{\circ}$ s $^{-1}$ to less than 6 $^{\circ}$ for angular velocities of 60 $^{\circ}$ s $^{-1}$ and above. Over the same range, the mean vector error magnitude decreases from approximately 0.0163 to 0.0059, while the magnitude ratio reduces from values between 2.73 and 4.68 at low angular velocity to 0.87 at 180 $^{\circ}$ s $^{-1}$. The corresponding mean magnitudes of ΔH_{pred} and ΔH_{ref} also become increasingly similar as angular velocity increases.

Both formulations were additionally compared against the imposed rotation axis. The mean angle between ΔH_{pred} and the rotation axis decreases from 21.6 $^{\circ}$ –38.6 $^{\circ}$ at low angular velocity to approximately 4 $^{\circ}$ –5 $^{\circ}$ for angular velocities of 60 $^{\circ}$ s $^{-1}$ and above. A similar trend is observed for ΔH_{ref} , whose mean angle relative to the imposed rotation axis decreases from approximately 67 $^{\circ}$ –84 $^{\circ}$ at low angular velocity to approximately 7 $^{\circ}$ at higher angular velocity. These quantities are reported as diagnostic metrics only and are not used directly for requirement verification.

Throughout the complete campaign, both formulations remained nearly perpendicular to the reconstructed magnetic field. The mean angle between ΔH_{ref} and the magnetic field remained effectively equal to 90 $^{\circ}$ for all datasets, with deviations below 0.01 $^{\circ}$. The corresponding angle for ΔH_{pred} ranged from approximately 86.4 $^{\circ}$ to 88.9 $^{\circ}$. This is consistent with the expected relationship between magnetic torque and the local magnetic field.

The differences between ΔH_{pred} and ΔH_{ref} therefore provide a quantitative estimate of the uncertainty associated with the angular-momentum reconstruction framework. This uncertainty propagates directly into the detumbling and nadir-pointing analyses presented later in this chapter and should be considered when interpreting requirement-level results based on reconstructed angular-momentum increments.

8.1.4. Dry-Run Uncertainty Summary

The dry-run campaign quantified the uncertainty associated with the reconstruction framework used throughout the final test campaign. For angular velocities up to 10 $^{\circ}$ s $^{-1}$, magnetic-field reconstruction residuals remained below approximately 2 μ T. At higher angular velocities, larger residuals were observed, reaching a maximum mean value of 12.6 μ T at 120 $^{\circ}$ s $^{-1}$. Similarly, the agreement between ΔH_{pred} and ΔH_{ref} improved with increasing angular velocity, with mean angular separations reducing from more than 50 $^{\circ}$ at low angular velocity to less than 6 $^{\circ}$ for angular velocities above 60 $^{\circ}$ s $^{-1}$. The uncertainty levels quantified in this section and in Table 8.2 define the confidence bounds within which the detumbling and nadir-pointing results presented in the

remainder of this chapter should be interpreted. The corresponding mean vector-error magnitude between ΔH_{pred} and ΔH_{ref} decreased from approximately 0.0163 at 1° s^{-1} to 0.0059 at 180° s^{-1} .

8.2. Final Test Campaign

ACT-1.2 Assessment

ACT-1.2 was evaluated using the reconstructed reference angular-momentum increment, ΔH_{ref} , which represents the best available estimate of the angular-momentum change generated during each actuation event. The hypothesis assumes that a valid detumbling action produces an angular-momentum change vector that opposes the angular velocity vector. Since all detumbling tests were performed using a controlled rotation about the stepper reference axis, the angle between ΔH_{ref} and the imposed rotation axis was used as the evaluation metric. Under the imposed test conditions, this angle is equivalent to the angle $\theta_{\Delta H, \omega}$ defined in section 4.1.

The acceptance criterion is

$$\mu_{\theta} - \sigma_{\theta} > 90^{\circ}$$

where μ_{θ} and σ_{θ} are the mean and standard deviation of the angular separation between ΔH_{ref} and the imposed rotation axis.

Table 8.3: ACT-1.2 assessment using the reconstructed reference angular-momentum increment

ω [$^{\circ} \text{s}^{-1}$]	μ_{θ} [deg]	σ_{θ} [deg]	$\mu_{\theta} - \sigma_{\theta}$ [deg]
1	67.29	62.97	4.32
2	84.30	68.46	15.84
5	81.13	66.60	14.52
10	51.68	55.60	-3.92
20	27.64	36.40	-8.76
30	28.30	41.63	-13.34
60	7.52	2.93	4.58
90	6.99	1.17	5.82
120	7.33	1.30	6.03
180	7.51	1.67	5.83

None of the tested angular-velocity cases satisfied the acceptance criterion. The maximum observed value of $\mu_{\theta} - \sigma_{\theta}$ was approximately 15.8° at 2°s^{-1} , which remains substantially below the required threshold of 90° .

For angular velocities between 60°s^{-1} and $180^{\circ} \text{s}^{-1}$, the mean angular separation between ΔH_{ref} and the imposed rotation axis remained approximately 7° . The reconstructed angular-momentum change vector therefore appeared nearly parallel to the imposed angular velocity direction rather than opposing it as expected for detumbling.

This behaviour is consistent with a possible sign inversion in the implemented B-dot control law, causing the generated magnetic torque to act with the rotation rather than against it. While the present results do not directly identify the source of the behaviour, the measured angular-momentum changes are not consistent with the expected detumbling response.

The reconstruction framework also exhibits substantially greater variability at low angular velocity than at high angular velocity. As the imposed rotation rate decreases, the magnetic-field variation associated with the rotation becomes smaller and environmental disturbances and measurement noise constitute a larger fraction of the measured signal. Consequently, the highest-angular-velocity datasets provide the most consistent reconstruction results.

Consequently, the null hypothesis cannot be rejected and ACT-1.2 is not verified by the experimental results.

8.2.1. Nadir-Pointing Results

The nadir-pointing scenario was evaluated using both the Earth Horizon Sensor attitude-estimation subsystem and the commanded magnetic-control response generated by the ADCS software. The

first part of this section assesses the accuracy of the horizon-sensing pipeline by comparing the estimated attitude against the imposed reference attitude. The second part evaluates the resulting control response against the hypothesis and acceptance criteria defined in Chapter 4.1.

Horizon Sensing Results

The final cleaned and corrected results of the pitch and roll estimates and the differences of these values compared to the true references values are shown below.

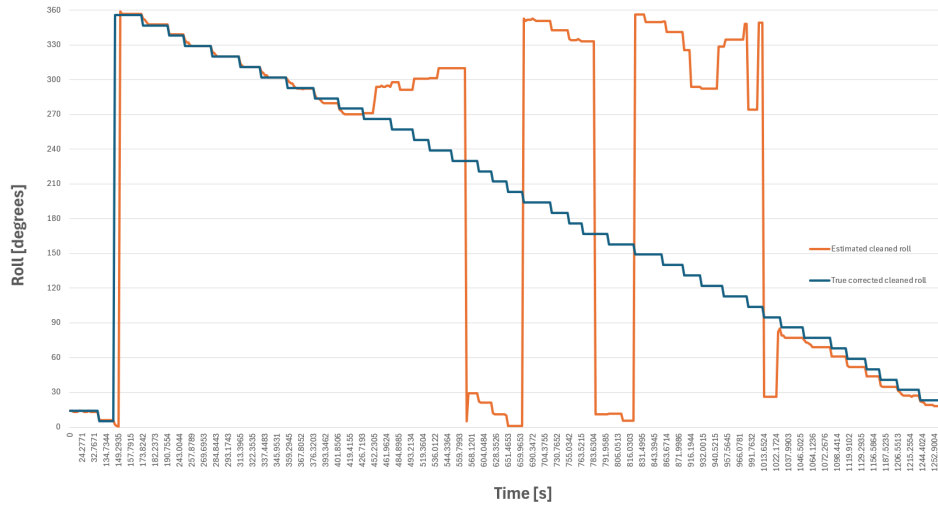


Figure 8.6: Cleaned estimated roll (orange) vs true corrected roll (blue) over 360°

As can be seen from Figure 8.6, the estimated attitude follows the true attitude very well up until 270 degrees. Why this phenomenon occurs remains partially unresolved. One possible explanation is that there is an implementation error in the way the algorithm handles the angle offset of the other sensors i.e. when the nadir sensor with zero offset (0x66) hands off the estimated attitude to the other sensors, it trips up. Sensor 0x66 would be the most trusted sensor and as expected at less than 270° the roll estimation's error becomes very large. Another observation that is consistent with this explanation is that the algorithm does pull itself together at 90° and below i.e. when sensor 0x66 regains view of the horizon simulant.

This does, unfortunately, not explain the whole phenomenon as the initial position of the nadir sensor is nadir and only has the simulant in view. The first attitude estimations are thus made by sensors 0x67 and 0x69. Only after the first couple of steps sensor 0x66 would have valid readings. Furthermore, the algorithm has a consensus system in place that takes the attitude state of multiple sensors that output roughly the same pitch and roll. This is however, superseded by trusting the sensor that has an amount of pixels in its mask that is closest to half of the pixels of the sensor. This system ensured that the sensor with the minimum amount of lens distortion is the most trusted.

The absolute difference between the estimated roll and true roll is showcased in Figure 8.7 with the red dashed line representing the 10° difference threshold. The moment the algorithm estimates begins to differ from the true attitude is clearly noticeable. The horizontal axis is still set to time as multiple readings took place on the same roll value. The system does need some time to stabilize to the appropriate attitude as can be seen by the small peaks every time the roll is proceeded.

Linking the results back to the requirement it can be obtained that 53.77% of the time the difference $\leq 10^\circ$. This is a far stretch from the 95 percentile that was set. When evaluating the assumption that the system is just flipped 180° the result creeps up to 56.75% of the time when the difference is $\leq 10^\circ$ or $\geq 170^\circ$.

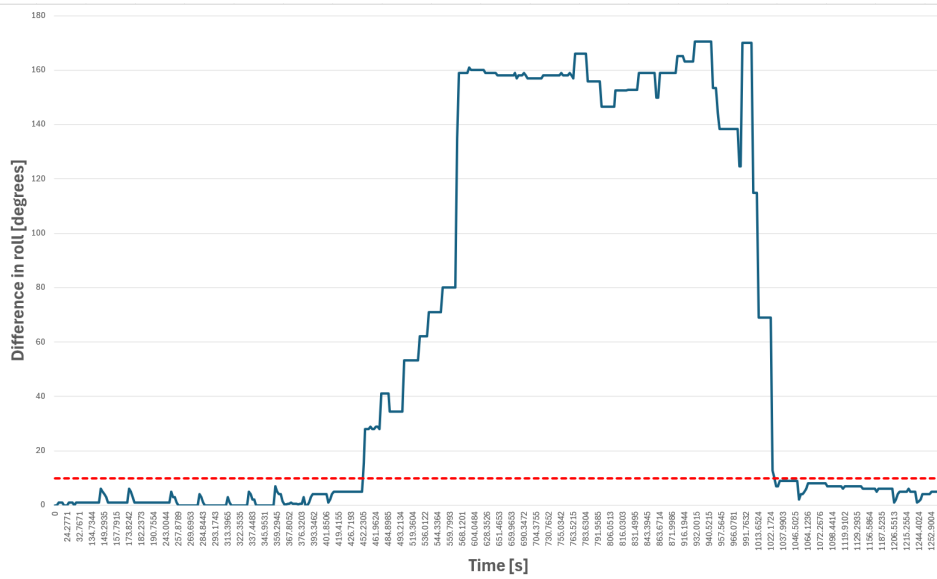


Figure 8.7: Cleaned absolute difference in estimated roll and true corrected roll over 360° with the maximum allowed deviation of 10° (dashed red)

When evaluating the performance of the pitch estimate, as seen in Figure 8.8, an improvement can be observed. The pitch estimation trips up at the same time as the roll estimation. However, after some large errors it stabilizes around 180° between roughly 150° to 210° . Meanwhile, the error that is observed in the roll during this time is semi-stable and bouncing between $\pm 160^\circ$ of the true roll. This should not have occurred as the algorithm, also briefly mentioned in subsection 8.1.1, calculates the combined 3D vector of the weighted trusted sensor readings and compares it to its previous reading via a dot product. If the result is less than zero, so a rotation over 90° , the signs are flipped. This was implemented to try to oppose the errors due to the degeneracy of attitude states. Furthermore, a consensus algorithm is implemented that compares the individual sensor estimates and if they fall within 10° of the last attitude state their weight is doubled. However, since the attitude estimation still jump to large incorrect values it indicates that all sensors briefly lose their active trusted sensor status and need to lock on from scratch. This can likely be able to be further circumvented by allowing for more grace frames but due to time constraint this could not be tested.

Once again, to check the results against the hypothesis and underlying requirement it was calculated that 57.94% of the time the pitch estimate is $\leq 10^\circ$ of the true attitude. As mentioned, the degeneracy of the attitude states causes the estimated attitude to be at 180° . If these estimations are also included the system jumps to being correct for 80.16% of the time; $\leq 10^\circ$ or $\geq 170^\circ$.

It can be concluded that the requirement is not met. All the recommendations for future studies that were not implemented due to the time constraint of this team are gathered are written in section 9.3. The current team expects that with some more grace frames and fixing the geometry offset bug the 95th percentile can be easily reached. As when the system is working nominally the error is 3.852° for roll and 4.228° for pitch. The error in roll is mainly caused by the consistent wrong estimation in the second part at the end of the full rotation, 6.203° , as seen in Figure 8.6 where the estimate is always below the true roll. If a continuous lock from the start is achieved it is speculated that the error can be brought down to 1.502° in roll as this was the average error before the system tripped up.

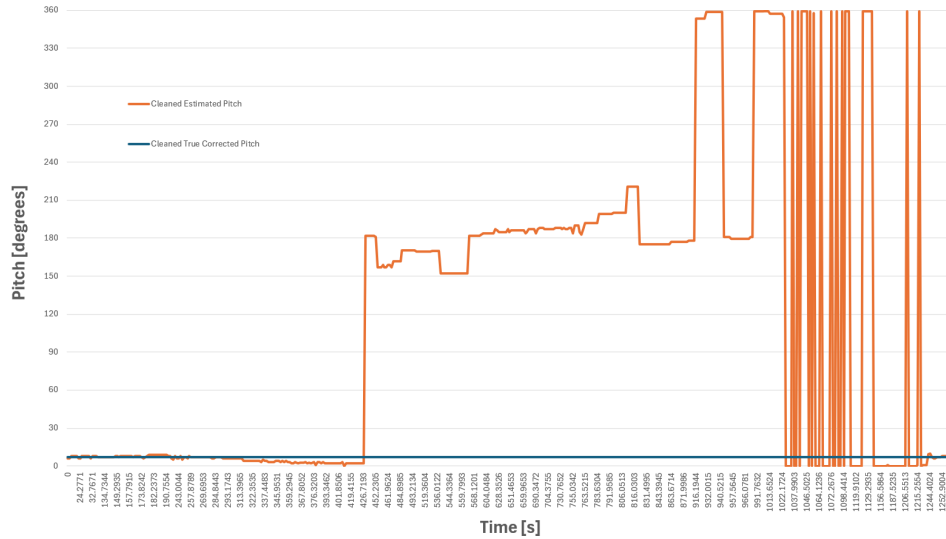


Figure 8.8: Cleaned estimated pitch (orange) vs true corrected pitch (blue) over 360°

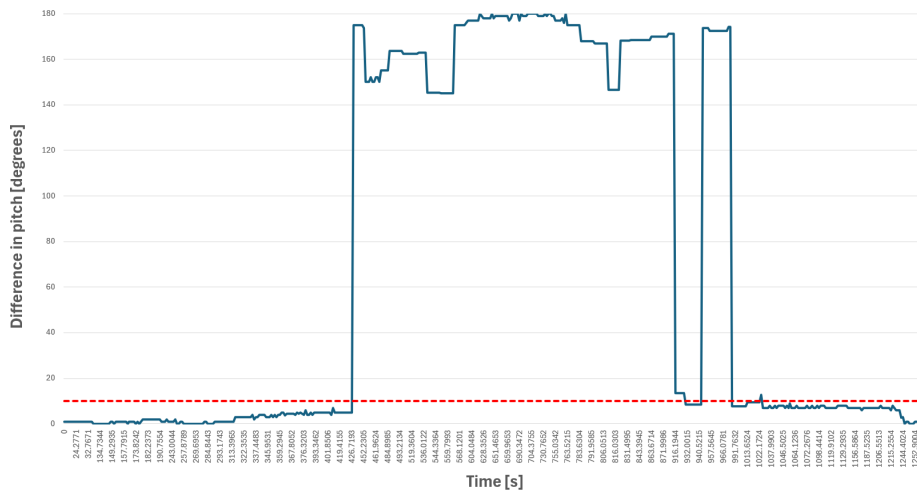


Figure 8.9: Cleaned absolute difference in estimated pitch and true corrected pitch over 360° with the maximum allowed deviation of 10° (dashed red)

Roll Performance

The absolute difference between the estimated and reference roll is shown in Figure 8.7. Requirement EHS-02 specifies that the attitude error shall remain below 10° for at least 95

Table 8.4: Roll estimation performance against requirement EHS-02

Criterion	Result
Time with $ e_{\text{roll}} \leq 10^\circ$	53.77%
Time with $ e_{\text{roll}} \leq 10^\circ$ or $\geq 170^\circ$	56.75%

The estimator remains within the required error bound for 53.77

When the estimator maintains a stable horizon lock, the mean roll error is approximately 3.85°. The dominant contribution to the overall error originates from the degraded estimation region

following sensor hand-off, where a persistent bias relative to the reference attitude is observed.

ACT-3.1* Assessment

ACT-3.1* was evaluated using the reference angular-momentum increment, ΔH_{ref} . During the nadir-pointing campaign, imposed roll-angle deviations were generated about the stepper reference axis. Consequently, the angle between ΔH_{ref} and the stepper reference axis was used as the test metric for evaluating the directional correctness of the reconstructed angular-momentum change.

The acceptance criterion defined in Chapter 4.1 is

$$\mu_{\theta_c} - \sigma_{\theta_c} > 90^\circ$$

where μ_{θ_c} and σ_{θ_c} are respectively the mean and standard deviation of the angular separation between ΔH_{ref} and the roll-error reference direction.

Table 8.5: ACT-3.1* assessment using the reference angular-momentum increment

Roll-angle deviation [deg]	μ_{θ_c} [deg]	σ_{θ_c} [deg]	$\mu_{\theta_c} - \sigma_{\theta_c}$ [deg]
0	79.36	21.85	57.51
18	87.16	11.39	75.77
27	82.12	18.66	63.46
45	89.45	15.24	74.22
54	94.65	5.13	89.52
63	84.92	11.78	73.13
72	89.64	1.49	88.16
81	89.39	6.46	82.93
90	78.86	23.14	55.72
99	83.72	11.25	72.47
108	69.19	31.58	37.61
117	84.37	7.97	76.40
135	81.50	25.89	55.61
144	88.06	18.48	69.58
180	94.21	7.40	86.80
216	95.83	9.44	86.39
225	91.10	7.03	84.08
252	91.11	1.71	89.40
270	93.61	6.84	86.77
315	91.58	8.70	82.87
360	90.50	7.07	83.43

None of the evaluated roll-angle cases satisfied the acceptance criterion. The largest observed value of $\mu_{\theta_c} - \sigma_{\theta_c}$ was approximately 89.5° , obtained for the 252° roll-angle case. Although several cases approached the required threshold, no test case exceeded the required value of 90° .

Consequently, the directional criterion defined by ACT-3.1* was not satisfied by any of the evaluated roll-angle cases. Under the software configuration used during testing, the null hypothesis cannot be rejected and ACT-3.1* is not verified by the experimental results.

Subsequent investigation of the flight software identified an implementation error in the nadir-pointing control algorithm that resulted in a zero commanded control torque during the nadir-pointing test campaign. As a result, the reconstructed angular-momentum increments presented here do not represent the intended nadir-pointing control response. Instead, the measured ΔH_{ref} values primarily reflect environmental disturbances, sensor noise, residual magnetic effects, and reconstruction uncertainty.

The ACT-3.1* assessment therefore characterises the behaviour of the software configuration tested during the campaign rather than the behaviour expected from the corrected nadir-pointing controller. Consequently, the results are insufficient to draw conclusions regarding the directional performance of the intended nadir-pointing control law.

The preceding sections present the measured test results and their interpretation. The requirement verification presented below is based solely on the experimentally obtained quantities and associated reconstruction uncertainties discussed throughout this chapter.

8.3. Requirement Verification

Table 8.6 summarizes the verification status of the requirements evaluated during the final test campaign.

Table 8.6: Requirement verification summary

Requirement	Verification metric	Status
ACT-1.2	$\mu_\theta - \sigma_\theta > 90^\circ$	Not verified
EHS-02 (Roll)	95% of measurements satisfy $ e_{\text{roll}} \leq 10^\circ$	Not verified
EHS-02 (Pitch)	95% of measurements satisfy $ e_{\text{pitch}} \leq 10^\circ$	Not verified
ACT-3.1*	$\mu_{\theta_c} - \sigma_{\theta_c} > 90^\circ$	Not verified

Requirement EHS-02 was not verified for either roll or pitch estimation. The attitude-estimation errors remained within the required $\pm 10^\circ$ bound for only 53.77% and 57.94% of the measurements for roll and pitch respectively, which is substantially below the required 95% threshold.

Requirement ACT-1.2 was not verified because the reconstructed reference angular-momentum increment did not consistently satisfy the required directional criterion. Likewise, requirement ACT-3.1* was not verified, as none of the evaluated roll-angle cases satisfied the acceptance criterion of $\mu_{\theta_c} - \sigma_{\theta_c} > 90^\circ$. Subsequent investigation identified an implementation error in the nadir-pointing controller that resulted in a zero commanded control torque during the test campaign. Consequently, the ACT-3.1* results represent the behaviour of the implemented software configuration tested during the campaign.

9

Conclusion

9.1. Recap of our goals and results

The primary goals of this Space Engineering Practical (SEP) were to advance the subsystem-level integration of the Delfi-Twin Attitude Determination and Control System (ADCS) and verify its implemented software functionalities. Specifically, the project aimed to practically test the algorithms responsible for de-tumbling the spacecraft and nadir pointing using four IR sensors and a magnetometer. A secondary goal was to integrate the hardware and software interfaces to create a reliable data pipeline capable of capturing telemetry and outputting actuation commands.

Ultimately, almost all of the target functional and performance requirements were not met during the final test campaign, primarily due to severe time constraints that limited final test capabilities. However, a robust and highly capable testing framework has been successfully established, providing an excellent foundation for next year's SEP team.

Overview of Results

- **Testing Framework and Integration:** A configurable breadboard model, a motorized single-axis test jig, and a heated horizon simulant were successfully constructed and validated. Furthermore, the wireless data acquisition (DAQ) pipeline, which utilizes Moteino modules to relay telemetry from the STM32's FreeRTOS environment to a laptop, was successfully established and unit-tested Appendix C.
- **Detumbling Verification:** The functional verification of the de-tumbling algorithm (Requirement ACT-1.2) was not achieved section 8.2. Test results showed that the commanded magnetic dipoles did not consistently produce a predicted angular momentum change that opposed the system's angular velocity. Significant discrepancies between the predicted and reference angular momentum increments were observed, particularly at lower angular velocities.
- **Nadir Pointing (EHS):** The Earth Horizon Sensor (EHS) system failed to meet the required accuracy threshold of $\leq 10^\circ$ for pitch and roll estimates (Requirement EHS-02) subsection 8.2.1]. The algorithm only achieved the target accuracy roughly 53% to 58% of the time]. This failure was largely attributed to algorithmic bugs in the sensor hand-off logic and unresolved degeneracy in the attitude states.
- **Nadir Pointing Control Response:** The directional verification of the nadir-pointing control response (Requirement ACT-3.1*) was not achieved. During the final nadir-pointing campaign, the tested software configuration produced zero commanded control torque. Therefore, the reconstructed angular-momentum increments did not represent the intended control response, and the requirement could not be verified from the available test data.
- **Sun-in-View Mitigation:** A notable success was the validation of the system's ability to ignore false infrared sources (Requirement ACT-4). Dry-run testing confirmed that the

implemented flood-fill algorithm functioned exactly as expected, successfully classifying and ignoring simulated solar illumination rather than tracking it as the Earth's horizon unless coinciding with the actual horizon.

Overall, the team was really happy to have been able to work on a project on subsystem level. However, the interfacing between all the components that themselves were projects that a whole SEP group worked on proved very time consuming. Furthermore, the components were not yet plug and play e.g. the EHS algorithm was only written in Python and lacked any multi-sensor capabilities. The robustness and speed also needed a major overhaul.

Furthermore, as the STM32 is written in C and no C or C++ coding was given in the bachelor and sparingly in the master courses it was difficult for a large part of the team to majorly contribute to this side of the project.

The main lesson learned is that subsystem-level verification strongly depends on the maturity of the test article and the interfaces before the final test campaign starts. In this project, a large part of the available effort was required to bring the BBM, EHS software, STM32 implementation, magnetometer logging, command-output reconstruction, and DAQ chain to a working integrated state. This reduced the time available for repeated testing, uncertainty treatment, and statistical verification. For future projects, the verification approach, data-processing chain, and uncertainty evaluation should therefore be prepared and validated as early as possible, rather than being treated mainly as final-stage activities.

The main aspect that went well was the development of a reusable subsystem-level test framework. The BBM, motorized test jig, horizon simulant, wireless DAQ chain, and post-processing framework provide a useful basis for future ADCS verification work. The dry-run tests also showed that individual parts of the system, especially the horizon-mask generation and infrared-source rejection, could operate as intended under laboratory conditions.

The main aspect that did not go well was the late transition from system integration to verification testing. The ADCS components were not yet plug-and-play, and several interfaces required additional implementation and debugging before meaningful end-to-end testing could be performed. This limited the number of final test iterations and the confidence with which the requirements could be assessed. As a result, the project should mainly be seen as establishing and partially validating an integrated test framework, rather than as a completed verification of the detumbling and nadir-pointing functionality.

9.2. Future Recommendations

The team recommends the carry through this project for next year's SEP as it was a blast to be able to contribute on such a high level but strongly advises a team of at least five, preferably six or seven. The possibility of having room 8.06 at our disposal was also very welcome.

9.2.1. System Engineering Work

Take some time at the beginning to sit with your supervisor to define your requirements. This will hopefully save some time with reformulation, which will inevitably happen but mitigating this as much as possible is preferred. Clearly differentiate between what your test article can verify/validate and what the actual system does.

9.2.2. HW-SW Interfaces

It is recommended to immediately work with **STM32CubeIDE** for writing scripts for the microcontroller. Flashing, building and debugging create the least amount of headache as the IDE is tailored for the STM32 platform.

However **first** configure the STM32 using **STM32CubeMX** to define the pins, enable the communication protocols like I2C1, I2C2, USART. Enable Direct Memory Access (DMA) for the USART, set data speed (Fast mode) for the I2C connections etc. Then press "GENERATE CODE" to receive the scaffolding in which you can code using the IDE. External scripts are not changed when generating new code/implementing changes in CubeMX, furthermore all the code in between "USER CODE

BEGIN" and "USER CODE ENDS" are also left untouched. Thus, make sure to always code inside these boundaries otherwise you risk losing all your code.

This should be logical but just to remind the reader, build up slowly. Perform unit test after unit test preferably with lots of print statements. Only problem you may encounter is that 'printf' statements may not work due to their transfer being slow so instead opt to use packages with headers (AA BB CC DD) for faster data transfer when testing larger parts of the system e.g. the camera feed and masks of all 4 IR sensors.

9.2.3. Production and Manufacturing

Future SEP students that work with hardware are strongly advised to book a training slot to be introduced to makerlab's 3D printing facility on campus at the earliest convenience, as most printing introduction slots are booked a month in advance. Tickets for the introduction at the makerslab can be obtained here at <https://tickets.tudelftsciencecentre.nl/en/makerlab/tickets> (charged).

It also recommended to include the MPU6050 in the test article, due to time constraints this was omitted this year. However, it is a useful baseline to test against.

9.2.4. EHS

Future SEP students are advised to first take some time to understand the algorithm. Starting with `HorizonDetector.cpp` as it is the heart of the system and works for a single sensor. Then `CombiningSensors.cpp` regarding the start-up and interfacing with the STM32. Lastly, `SwitchingLogic.cpp` concerns the hand off between sensors and likely contains the geometry bug. Some other less important files are `SensorGeometry.cpp`, containing the lens distortion values, `ConversionTable.cpp`, containing the look-up table, `HorizonBridge.cpp`, containing the wrapper that allows the C++ files to communicate with the C structure of the STM32. The header files (`.h/.hpp`) mainly contain the definitions declared in the source files (`.c/.cpp`). Here `HorizonTypes.hpp` has the main output variables and is useful when performing (unit)tests.

9.3. Future studies

9.3.1. EHS

The following topics/points are yet to be tackled or investigated but are believed to improve performance:

- Investigate the impact of the sensor distance deviation from horizon simulant. As described in Figure 2.1.3, the distance between the test article and the size of the horizon simulant both scaled down evenly but the distance between the sensor did not change. This caused the perceived altitude of the sensors to fluctuate between 363 km and 545 km. It must be said that this did not seem to impact the attitude estimation much but it still has to be tackled one way or the other, maybe prove that the difference of the curve in the horizon is negligible due to the coarseness of the sensor.
- The Sun in view testing lacked the proper star radian angle of the heat source, it was too large at the dry tests done. This was due to the team not having an appropriate heat source, so a mug of boiling water was used. However, it did still show the working of the flood fill algorithm. So this point may be redundant.
- Set the amount of grace frames higher to see if more stability is achieved.
- Use a moving average of the last 3 known attitude states instead of only the last observed measurement to check the current measurement against. This would further mitigate large attitude estimation jumps.
- Configure the system to use have Sun sensing capabilities, the Earth horizon sensors would also function as Sun sensors. This is further worked out in Nikolas Knöll's thesis[5].
- Investigate how to mitigate the error stemming from the degeneracy of attitude states, for sensors 0° pitch and 180° roll is identical to 180° pitch and 0° roll.

- Drill the resistors to the horizon simulant to have a more effective heat transfer

9.3.2. Magnetorquers

Due to insufficient time, the performance testing of magnetorquers was not performed. The magnetorquers were sized and wound according to their supposed strength as if it were PCB-integrated magnetorquers. Thus, next year can use the assembled MTQs of this project.

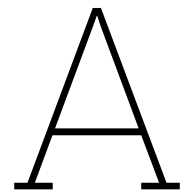
9.3.3. Stepper motor

The stepper motor's tooth span 0.9° , this can be reduced via micro-stepping. However, it resulted in a jerky jittering motion which was not perfect for the small rotation rates that nadir pointing scenario's are working with. Thus it is advised to include a gear ratio inbetween the stepper motor and main rod of the test article.

Furthermore, the ideal scenario would be hardware-on-the-loop testing by starting with an initial rotation. Reading out the supposed counter torque commanded by the STM32 and subtracting this from the stepper motor's rotation to see the actual reduction of the system's angular rate.

References

- [1] Siddharth Dixit, Simon Van Hulle, and Valerie Veldhuizen. *Detumbling the PQ: Prototyping and testing the Magnetic Attitude Determination and Control for DelfiPQ*. Tech. rep. Delft University of Technology, 2024.
- [2] ECSS. *ECSS-E-ST-10-02C Rev.1 – Verification*. 2018. URL: <https://ecss.nl/standard/ecss-e-st-10-02c-rev-1-verification-1-february-2018/> (visited on 02/01/2018).
- [3] HobbyKing. *Turnigy Nano-Tech 850mAh 2S 12C LiFePo4*. 2026. URL: https://hobbyking.com/en_us/turnigy-nano-tech-850mah-2s-12c-lifepo4-transmitter-receiver-pack-w-jr-s-type-connector.html?__store=en_us (visited on 04/04/2026).
- [4] T. Jansen et al. *PERFORMANCE OF THE FIRST FLIGHT EXPERIMENT WITH DEDICATED SPACE CIGS CELLS ONBOARD THE DELFI-C3 NANOSATELLITE*. Tech. rep. Delft University of Technology, 2010.
- [5] Nikolas Knöll. “Optical Attitude Estimation System with Low-Resolution Infrared Imaging”. MA thesis. Delft University of Technology, 2026.
- [6] LowPowerLab. *All about Moteino | LowPowerLab*. URL: <https://lowpowerlab.com/guide/moteino/> (visited on 06/19/2026).
- [7] Melexis. *melexis/mlx90642-library: Library for the MLX90642 Infrared array*. 2017. URL: <https://github.com/melexis/mlx90642-library?tab=readme-ov-file> (visited on 06/19/2026).
- [8] Lab IR Paints. *Thermographic paint for high temperature applications*. URL: <https://paints.labir.cz/en/paints/herp-ht/> (visited on 05/28/2026).
- [9] M. Pavel et al. *Delfi-Twin ADCS*. Tech. rep. Delft University of Technology, 2026.
- [10] J. G. Sampedro et al. *Earth Sensor for PocketQube*. Tech. rep. Delft University of Technology, 2025.
- [11] Texas Instruments. *DRV8825 Stepper Motor Controller IC*. Tech. rep. SLVSA73F. Originally published April 2010. Texas Instruments, July 2014. URL: <https://www.tinytronics.nl/en/mechanics-and-actuators/motor-controllers-and-drivers/stepper-motor-controllers-and-drivers/drv8825-motor-driver-module> (visited on 06/21/2026).
- [12] Texas Instruments. *DRV8838EVM User’s Guide*. Tech. rep. SLVUA43. Texas Instruments, Feb. 2014.
- [13] Texas Instruments. *DRV883x Low-Voltage H-Bridge Driver*. Tech. rep. SLVSB44F. Originally published June 2012. Texas Instruments, Apr. 2021.
- [14] TinyTronics. *17HS8401 Stepper Motor Datasheet*. NEMA17 stepper motor, 0.49 N·m, 1.68 A, JST-PH compatible connector. 2022. URL: <https://www.tinytronics.nl/en/mechanics-and-actuators/motors/stepper-motors/stepper-motor-0.49n.m-1.68a-nema17-jst-ph-compatible-connector> (visited on 06/21/2026).
- [15] TinyTronics. *DC-DC Step-Down Buck Converter 1.5A - 3.3V output - DCDCSTEPDOWN3.3V*. 2026. URL: [https://www.tinytronics.nl/nl/power/spanningsconverters/buck-\(step-down\)-converters/dc-dc-step-down-buck-converter-1.5a-3.3v-output](https://www.tinytronics.nl/nl/power/spanningsconverters/buck-(step-down)-converters/dc-dc-step-down-buck-converter-1.5a-3.3v-output) (visited on 04/04/2026).
- [16] TinyTronics. *MPU-6050 Accelerometer en Gyroscope 3-axis Module 3.3V-5V - MPU6050*. 2026. URL: <https://www.tinytronics.nl/nl/sensoren/acceleratie-rotatie/mpu-6050-accelerometer-en-gyroscope-3-axis-module-3.3v-5v> (visited on 04/04/2026).



Figures

These are the uncleaned tables that still contain the data points where the output was zero for pitch and roll.

Estimated roll vs true corrected roll over 360°

Difference in estimated pitch and true corrected pitch over 360°

Estimated pitch vs true corrected pitch over 360°

Difference in estimated roll and true corrected roll over 360°

Ideal scenario's

B

Requirements

Table B.1: Requirements Definition

ID	Requirement	Rationale	Test Method
ENV-2	System components shall either be shielded or withstand direct contact with atomic oxygen.	Atomic oxygen has the potential to noticeably deteriorate component structures which negatively impact performance.	Inspection
ENV-3	The system shall operate between -10°C and 40°C .	Eclipsing leads to thermal cycling. The values used here are common values used for small satellites.	Inspection
ENV-4	The system shall withstand the mechanical loads as specified in the Falcon 9 user guide.	Delfi-Twin will be launched on a Falcon 9. This requirement will likely be addressed after the design process onto PCB design.	Analysis
ENV-7	The components of the system shall be rated to withstand Low Earth Orbit ionizing radiation for at least 5 years.	The system will have to operate in the low Earth orbit radiation environment for its entire mission lifetime.	Inspection
PWR-6	The system shall consume not more than 1 Watt in normal operation, with a peak consumption of 2 Watts.	Since the satellite has not more than 4 Watts available, the system is limited to a small chunk of the total power budget.	Testing
PWR-1	The system shall operate using a voltage between 2.2 V and 4.5 V.	The bus voltage is unregulated between 2.2 V and 4.5 V.	Testing
PWR-7	The system shall draw a peak of no more than 0.5 A.	Since the whole satellite might be getting 1 A or 1.5 A, we cannot take a big chunk of that.	Testing
PWR-3	The system shall remain physically undamaged if power supply is cut off.	To prepare for unexpected safe mode initiations.	Testing
PWR-4	The system shall enter hibernate mode upon every boot-up.	The OBC decides what the ADCS should be doing. It waits for a command from the OBC before doing anything.	Testing

Continued on next page

Table B.1 – continued from previous page

ID	Requirement	Rationale	Test Method
REC-1	The system shall recognize and ignore faulty commands from the OBC.	It is important that the ADCS behaves in an expected manner. It may only execute commands that it clearly understands.	Testing
REC-3	The system's operational mode shall be set by command from OBC.	The OBC decides what the ADCS is doing, which is determined by its operational mode.	Testing
REC-4	The system shall receive positional data from the OBC.	Based on ground communications, we can correct our estimations with what's observed on the ground.	Testing
REC-6	The system shall change its Nadir-pointing axis on command from the OBC.	Being able to arbitrarily change the roll of the spacecraft opens up opportunities for research.	Testing
SND-4	The system shall send all stored system data to the OBC on request.	By sending all stored system data, the OBC is given full insight into the health and operation of the ADCS.	Testing
ACT-1	The system shall de-tumble the satellite by reducing its rotational rate to at most 0.5° s^{-1} .	De-tumbling is one of the ADCS's main tasks. The cut-off value is an educated guess.	Simulation
ACT-1.1	The system shall be able to de-tumble from at least 180° s^{-1} .	To fully specify system performance, maximum initial spin rate is required to be constrained.	Simulation
ACT-1.2	The system shall de-tumble the satellite in no more than a week, preferably 3 days.	Defining system performance requires a time constraint that balances power and mission time.	Simulation
ACT-2	The system shall point the +z axis of the satellite at the velocity vector within 10° at 1 sigma confidence.	Key function to point keep a prograde orientation when pointing to Nadir.	Simulation
ACT-2.1	The system shall provide x- and y-axis torques at least 2 times stronger than disturbance torques in the respective axes at worst case scenario.	Safety factor enforced by the need to overcome the disturbance torques on the satellite.	Test/Analysis
ACT-3	The system shall point the $\pm x$ or $\pm y$ axis of the satellite with the Nadir within 30° at 1 sigma confidence.	Requires more than 45 degree accuracy to discern which face is pointed at Earth.	Simulation
ACT-3.1	The system shall provide a z-axis torque at least 2 times stronger than disturbance torques in that axis at worst case scenario.	Safety factor enforced by the need to overcome the disturbance torques on the satellite.	Test/Analysis
ACT-3.2	The system shall be able to turn 180 degrees in 12 hours around the z axis.	Required for orientation towards Nadir. Constrains the performance of the MTQ.	Simulation

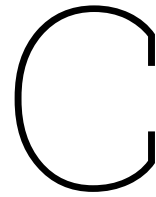
Continued on next page

Table B.1 – continued from previous page

ID	Requirement	Rationale	Test Method
ACT-4	The system shall ignore the presence of the Sun.	The system should only care about the Earth's horizon; Sun and Moon are noise.	Analysis
ACT-5	The magnetometer shall perform its readings after the magnetorquer's magnetic field is less than 10% of the external field.	MM's readings might be influenced by the MTQ if not enough time is given.	Analysis
CON-1	The system shall occupy an indivisible volume not greater than 45 x 45 x 15 mm, except the 80 x 40 mm side panels which may be used.	Volume constraint set to pursue development independently from the rest of the team.	Integration
CON-3	The system shall weigh less than 200 g.	The satellite must weigh less than 750 g, and the ADCS should not take a big portion.	Engineering
CON-4	The system shall be able to operate anywhere on the satellite.	Since the layout is not known, the ADCS needs to be designed for flexible placement.	Analysis
EHS-02	The Earth Horizon Sensors shall provide a pitch and roll measurement with an accuracy of at least 10°.	Ensures the Earth horizon can be found to a sufficiently high accuracy for ACT-2/3.	Analysis
EHS-03	Each Earth Horizon Sensor shall take no longer than 20 ms to perform a full reading.	Required for real-time predictability. The MLX90640 typically takes 19 ms.	Testing
MTQ-01	Each control loop, the MTQs shall actuate for at most 50% of the cycle.	Ensures average power stays below 1 Watt in accordance with PWR-06.	Review
MTQ-02	The combined peak consumption of all magnetorquers at once shall be no more than 1.8 W.	Allocates the largest portion of the power budget to the MTQs.	Testing
MTQ-03	The MTQs shall be able to provide a torque quantum each control loop to a step of 5% of the maximum torque in that loop.	Provide possibility to modulate strength of the MTQs each cycle via time modulation.	Testing
MTQ-04	The MTQs shall have a combined rise fall time of no more than 10% of the minimum torquing duration.	Ensures 90% of cycle is at nominal torque for predictable kicks.	Testing
MTQ-05	Each MTQ shall produce a magnetic torque of at least $2.2 \cdot 10^{-7}$ Nm.	Required to perform de-tumbling from 180 deg/s to 0.5 deg/s in 3 days.	Analysis
MTQ-06	The x and y magnetorquers shall produce a magnetic torque of at least $4 \cdot 10^{-7}$ Nm.	Necessary to reorient the satellite according to a new Nadir direction within 12 hours.	Analysis
MTQ-07	Each MTQ shall produce a magnetic torque of at least $9 \cdot 10^{-7}$ Nm.	Required to overcome aerodynamic torque at worst case scenario at 300 km.	Analysis
Continued on next page			

Table B.1 – continued from previous page

ID	Requirement	Rationale	Test Method
MTQ-08	All MTQs operating simultaneously shall consume no more than 0.5 A in peak loading.	Ensures compliance with total current consumption limits from PWR-7.	Analysis
MTQ-09	Each MTQ shall operate at a fixed voltage of 3.3 V.	All components will run on a fixed supply through a voltage regulator.	Analysis
MGM-01	The magnetometer shall take no more than 15 ms to provide magnetic field data to the STM32.	Ensures real-time loop timing; standard readings take 7 ms.	Testing
MGM-02	The magnetometer shall measure the Earth's magnetic field with a total uncertainty not exceeding 0.2 %.	Allows the ADCS to effectively apply its b-dot algorithm to de-tumble.	Testing
MGM-03	The magnetometer shall measure magnetic fields in the range of $\pm 100 \mu\text{T}$.	Required range to capture LEO magnetic field variations.	Testing
MCU-01	The PCB thickness shall be no less than 1.3 mm and no more than 1.8 mm.	Constrained for MLX90640 sensors to slide onto the PCB and be soldered.	Inspection



Software development steps

In order to design the software, the following unit tests have to be performed first to gain enough experience:

- Change address of MLX sensor.
 - * Why: We'll need to have 4 sensors on one I2C channel, so we need to have custom addresses for each.
 - * How: Written a C++ file for the STM32 that erases and then rewrites the bits that define the address from 0x33 to 0x34. Script has built in messages for when it has completed a step i.e. "Current address is 0x00" or "Erase failed! Register is 0x33. DO NOT POWER CYCLE!". This step is crucial as the sensor can get bricked when power cycled with an invalid address, extra time is also allocated for each step. As it was written in C++ whilst main code for STM32 is in C a wrapper was made so it gets executed.
 - * Outcome: First tried it with the MLX90640 and it worked, this provided confidence to do it with the MLX90642s.
- Communicate with multiple MLX sensors on the same channel.
 - * Why: We'll need to have 2 sensors on one I2C channel, as the amount of I2C channels is three where one is reserved for the magnetometer.
 - * How: Connect 2 MLX sensors on one channel, and then get data from each of them. Visualize the images to test that each sensor is read properly.
 - * Outcome: First rewrote the address, then connected them and assigned the proper addresses in the code. Got a visual and their respective masks. Task completed.
- Get good data from RM3100.
 - * Why: Before writing any algorithm with the MM, we first have to make sure that we're getting data, and that the data is not garbage.
 - * How: get the STM32 to talk to the RM3100, and send data back through serial. Then experiment with different magnetic items to sanity-check the measurements.
 - * Outcome: Getting this to work required multiple days due to hard-to-spot hardware misimplementation (the MISO pin was used in I2C to set the address of the peripheral; it was left intentionally not wired up so it would stay 0, but apparently it's open drain, causing I2C address problems; connecting MISO to GND and SS to VIN fixed all issues). The library from the previous year's SEP group was used, but because it was written in C++, it had to be converted into C first. Memory management through stack size allocation and value-to-pointer conversion were some of the issues that appeared. In the end, the RM3100 was able to give fast data (10Hz), with the DRDY pin taking about 3 ms to signal successful data collection. Sanity checks on the computed angular velocity

indicate good alignment on y-axis, possible coupling between x and z. Recommend further unit tests after more critical steps are finished.

- Establish communications with Moteino.
 - * Why: Before we can attempt to wirelessly transmit from one Moteino to another, we first have to establish that the laptop can talk to each Moteino (saying "Hello world" might not be as easy as plugging in and pressing a button).
 - * How: Look up online tutorials for Hello World programs on the Moteino and make sure that the Moteino can execute basic programs like printing, loops and GPIO.
 - * Outcome: "Hello world" program required the installation of FTDI VCP drivers. After installation, the Moteino behaves like a normal Arduino device and prints "Hello world" to the serial bus.
- Test ADC functionality on Moteino.
 - * Why: The Moteino has to be able to accurately read voltages in order for our test to work, so it is necessary to check that the Moteino can actually do that.
 - * How: Hook up Moteino ADC pins to the programmable power supply, and use resistors to give different known voltages, and measure them with a multimeter and the Moteino. Then compare
 - * Outcome: Power supply set voltage did not match with multimeter's readings, so the power supply was not used. Connected 3V3 to A0 and through a 4.5k ohm resistor to GND. A0 read 5V, matching multimeter reading. Next, connected 4.6k ohm and 2.1k ohm resistors between 3V3 and GND, 1.58 V, multimeter 1.64 V. Multimeter also overestimated power supply's voltage, instrument induced effects might be involved.
- Test wireless communication between two Moteinos.
 - * Why: Wireless communication is very important for our tests. We need to ensure that the Moteinos can actually perform it reliably.
 - * How: Program one Moteino to broadcast some known sequence of numbers. Use another Moteino to listen to those and send what it receives to the laptop via serial communication
 - * Outcome: Used existing example code to successfully send a data packet from one Moteino to another. Data packet comes in the form of a struct whose values can be changed at any time. Packet size and shape can be changed.
- Test STM32-Moteino communication through I2C
 - * Why: In the final test, the only way to get live data out of the STM32 is by sending it through the Moteino. We have to test that the two can communicate with each other.
 - * How: Connect StM32 and Moteino through I2C3. With Moteino in master mode, get the STM32 to send it some dummy package, which is then send through serial to the laptop.
 - * Outcome: Moteino successfully sent a known sequence of numbers to the STM32, which then sent them via serial to the laptop. After further development of the driver, The STM32 was able to successfully send a known sequence of numbers to the Moteino on the Moteino's request.
- Test STM32-Moteino-Moteino wireless communication.
 - * Why: In the final test, live data from the STM32 has to be sent through the onboard Moteino to another Moteino wirelessly, and then through serial to a laptop. This test should conclude that we can get live data from the STM32 while it is spinning in the test article.
 - * How: Connect Moteino A to Nucleo board, and have the STM32 send a dummy package to Moteino A. Moteino A then sends this dummy package wirelessly to Moteino B, connected to a laptop. Moteino B sends what it receives via serial to the laptop.

- * Outcome: Some extra functions were developed to support transmitting float values over I2C. Putting together Moteino wireless communication and STM32-RM3100 data transfer, it was possible to create a data packet that a Moteino could receive from meters away, indicating the angular velocity of the RM3100 in 3 axes.
- Test DRV8837 output by itself.
 - * Why: To make sure that the drivers work as expected, we first test if they give the right output as a function of their input.
 - * How: Connect one driver to the Nucleo according to the schematic, then send each row of the truth table from the DRV8837 documentation one by one and measure the output.
 - * Outcome: STM32 was programmed to give alternating PH and EN pins. OUT1 and OUT2 behaved as expected.
- Test DRV8837 with a dummy MTQ.
 - * Why: It might be straightforward, but if the H bridge doesn't have flyback diodes, we need to get our own, implement them into our design and test them.
 - * How: Connect DRV883x OUT1 and OUT2 pins to X-torquer. Place the X-torquer near the RM3100 and check if the measured magnetic field changes according to STM32 input.
 - * Outcome: X-torquer was connected to the DRV8837 and placed next to the RM3100. The STM32 gave the same alternating EN and PH inputs as in the last unit test and showed the measured B field from the magnetometer. The B field changed as expected, even when rotating the X torquer manually.
- Test Moteino ADC on MTQ output.
 - * Why: As a final unit test, the Moteino has to be proven to be able to read the current that goes through the magnetorquer. This is data that will be used in the final experiment to tell if the ADCS is behaving within requirements.
 - * How: Disconnect the X torquer terminal from the OUT1 pin of the DRV8837 and reconnect with a 1 Ohm resistor in between. Connect two Moteino ADC pins, one at each side of the resistor, and measure the voltage difference. Divide by 1 Ohm to get current.
 - * Outcome: Setup was modified as described above. In one direction, the Moteino read -0.07 A, slightly above expected. In the other, the Moteino read 1.6 A, much higher than expected. The 1 Ohm resistor heated to very high temperature, correlating with a high amperage. Power was cut, and no damage to the system was sustained. Probable cause was a wiring error, but it showed that the Moteino can measure current. In a separate setup with a single 75 Ohm resistor, the Moteino measured the exact predicted current, which serves as verification of the current measurement function. More caution will be used in the future when wiring the magnetorquers.

–

D

Framework Parameters and Conventions

The present appendix summarizes the fixed framework parameters and numerical conventions used throughout the data-processing framework.

D.1. Communication Parameters

`PACKET_ARRAY_SIZE = 12`

defines the number of floating-point values contained in each payload packet.

D.2. Event-Frame Structure

Each telemetry frame contains three magnetometer measurements

`(BX1, BY1, BZ1, MM1_TIME)`

`(BX2, BY2, BZ2, MM2_TIME)`

`(BX3, BY3, BZ3, MM3_TIME)`

These definitions determine the magnetic event construction procedure.

D.3. Outlier Parameters

`OUTLIER_SIGMA_THRESHOLD = 1`

defines the threshold used during outlier detection.

Samples whose distance from the component mean exceeds one standard deviation are classified as outliers.

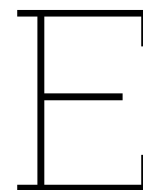
D.4. Angular Diagnostic Quantities

The angular diagnostic quantities are available for diagnostic analysis.

The quantity

$$\theta(\Delta H_{\text{error}}, \hat{\omega}) \tag{D.1}$$

is retained for future analysis and is not currently included in the figures.



Plot and Export Implementation

The present appendix summarizes the file naming conventions, export hierarchy and plotting conventions used throughout the framework.

E.1. File Naming

Log files follow

YYYYMMDD_AXIS_OMEGA_RUN.csv

For example

20260619_X_180_06.csv

contains

- rotation axis = X,
- angular velocity = 180 deg/s,
- run number = 06.

The rotation axis and angular velocity are treated as known experimental inputs.

E.2. Export Directory Structure

Each experiment contains

dataset

|-- reference.csv

|-- noise_statistics.csv

'-- plots

results

|-- campaign_noise_summary.csv

'-- datasets

 '-- dataset_name

 |-- reference.csv

 |-- noise_statistics.csv

 '-- plots

E.3. Plot Parameters

Component quantities use symmetric limits

$$[-v_{\max}, v_{\max}] \quad (\text{E.1})$$

while magnitude quantities use

$$[0, v_{\max}]. \quad (\text{E.2})$$

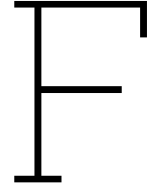
Global plot limits are determined from finite values over all valid datasets.

A small margin is added to the extrema before the final plotting limits are applied.

Thirty histogram bins are currently used.

E.4. Disabled Figures

Several previously used vector-error and angular-error figures are retained in the source code but are currently disabled.



Magnetometer Alignment Calibration

The present appendix summarizes the calibration procedure used to estimate the orientation offset between the magnetometer frame and the body frame.

F.1. Calibration Philosophy

The calibration procedure estimates the orientation offset between the magnetometer frame and the body frame.

The objective is to determine the alignment that produces the most nearly constant magnetic field after transformation into the external fixed frame.

F.2. Alignment Direction Cosine Matrix

A direction cosine matrix

$$C_{BM} \tag{F.1}$$

maps magnetometer-frame quantities into the body frame.

The matrix is parameterized by three small rotation angles

$$\delta = [\delta_x, \delta_y, \delta_z]^T. \tag{F.2}$$

The alignment matrix is obtained from the corresponding rotation vector.

F.3. External-Fixed-Frame Reconstruction

For each magnetic measurement

$$B_{\text{mag}}, \tag{F.3}$$

the body-frame field is computed as

$$B_{\text{body}} = C_{BM} B_{\text{mag}}. \tag{F.4}$$

The body-frame measurement is transformed into the external fixed frame using

$$B_{I,est} = R^T B_{body}. \quad (F.5)$$

where

$$R \quad (F.6)$$

denotes the rotation matrix corresponding to the prescribed rotation axis and angular velocity.

F.4. Constant Inertial Field Estimation

The reconstructed inertial measurements are cleaned using the same outlier-removal procedure employed by the reference framework.

The constant inertial field estimate is obtained by averaging the cleaned inertial measurements.

Datasets producing undefined estimates are rejected.

F.5. Forward Prediction

The estimated constant inertial field is transformed back into the body frame and subsequently into the magnetometer frame.

Predicted magnetometer measurements are generated for every magnetic sample.

F.6. Residual Generation

Residual vectors are computed from the difference between the measured and predicted magnetometer quantities.

The squared Euclidean norm of each residual vector is evaluated.

F.7. Cost Function

The objective function is defined as the mean squared residual magnitude accumulated over all datasets.

F.8. Numerical Optimization

The alignment angles are determined by minimizing the cost function.

Optimization is performed using the L-BFGS-B algorithm.

Bounds are imposed on each alignment angle to restrict the search region.

F.9. Calibration Dataset Selection

Only datasets corresponding to

$$|\omega| = 30 \text{ deg/s} \quad (F.7)$$

and rotations about the X and Z axes are used for calibration.

Datasets containing insufficient magnetic samples are excluded.

F.10. Exported Calibration Products

The calibration procedure exports

- the alignment angles,
- the body-to-magnetometer direction cosine matrix,

- the inverse direction cosine matrix.

Additional determinant and orthogonality checks are performed to confirm that the resulting matrix remains a valid direction cosine matrix.

F.11. Angular-Diagnostic Numerical Conventions

Before evaluating inverse cosine functions, the cosine argument is clipped to

$$[-1, 1]. \quad (\text{F.8})$$

If either vector possesses zero magnitude,

$$\text{NaN} \quad (\text{F.9})$$

is returned.

These conventions prevent numerical singularities during angular calculations.



Estimator Evaluation Framework

The present appendix summarizes the framework used to evaluate estimator performance using reference datasets.

G.1. Framework Architecture

The estimator evaluation framework performs the following sequence:

```
reference.csv
|
v
dataset loading
|
v
estimator execution
|
v
metric evaluation
|
v
campaign aggregation
|
v
exported outputs
|
v
visualization
|
v
Pareto analysis
```

G.2. Dataset Loading

Reference datasets are reconstructed from the exported reference.csv files.

Only rows corresponding to valid magnetic measurements are retained.

For each dataset, the framework reconstructs

- timestamps,
- measured magnetic field,
- reference magnetic field,

- reference magnetic-field derivative,

together with the corresponding experimental conditions.

G.3. Estimator Execution

Enabled estimators are instantiated from the configuration.

Each estimator operates independently on the same dataset.

The resulting magnetic-field predictions and derivative estimates are collected into estimator-specific result structures.

G.3.1. Causality Convention

The estimator framework follows a causal prediction convention.

Historical information up to sample

$$i \tag{G.1}$$

is used to generate predictions at

$$i + 1. \tag{G.2}$$

Consequently, prediction-horizon handling remains internal to each estimator and no additional shifting operations are required elsewhere in the framework.

G.4. Performance Metrics

Estimator performance is characterized using

- vector-error metrics,
- angular-error metrics,
- magnetic-field derivative error,
- derivative smoothness.

G.4.1. Vector Error Metrics

Mean, root-mean-square and maximum vector errors are evaluated using the difference between the predicted and reference magnetic fields.

G.4.2. Angular Error Metrics

Mean, root-mean-square and maximum angular errors characterize directional disagreement between the predicted and reference magnetic fields.

Samples containing zero-magnitude vectors are excluded.

G.4.3. Magnetic-Field Derivative Error

Derivative accuracy is evaluated through the root-mean-square error between the estimated and reference magnetic-field derivative quantities.

G.4.4. Derivative Smoothness

Smoothness is evaluated from the variation of successive derivative estimates.

Small values indicate smoother derivative behaviour.

G.5. Campaign Aggregation

Performance metrics obtained from individual datasets are combined into a campaign summary.

Each entry contains

- dataset identity,
- rotation axis,
- angular velocity,
- estimator identity,
- metric values.

The campaign summary provides the common representation used by subsequent campaign outputs.

G.6. Outputs

G.6.1. Dataset Outputs

For every dataset, estimator-specific outputs are exported.

These outputs contain

- measured magnetic-field quantities,
- reference magnetic-field quantities,
- predicted magnetic-field quantities,
- reference magnetic-field derivative quantities,
- estimated magnetic-field derivative quantities.

Both component quantities and vector magnitudes are included.

G.6.2. Campaign Outputs

Campaign-level exports combine the results from all datasets.

Generated outputs include

- estimator results,
- prediction tables,
- derivative prediction tables,
- campaign statistics,
- estimator rankings,
- spreadsheet summaries.

Campaign statistics are obtained by averaging the metrics associated with each estimator over all datasets.

G.6.3. Estimator Ranking

Estimators are ranked according to

1. root-mean-square magnetic-field derivative error,
2. derivative smoothness,
3. root-mean-square vector error.

This ordering prioritizes derivative quality while retaining information regarding prediction accuracy.

G.7. Figures

G.7.1. Dataset Figures

Each dataset generates comparison figures describing

- reference magnetic-field quantities,
- predicted magnetic-field quantities,
- reference magnetic-field derivative quantities,
- estimated magnetic-field derivative quantities.

G.7.2. Campaign Figures

Campaign-level figures characterize the variation of estimator performance with angular velocity.

Performance trends are generated for

- root-mean-square vector error,
- root-mean-square angular error,
- root-mean-square derivative error,
- derivative smoothness.

G.7.3. Angular-Velocity Statistics

Average metric values are additionally grouped according to angular velocity.

These statistics are retained for subsequent analyses.

G.8. Pareto Analysis

Average derivative accuracy and smoothness are combined into a Pareto representation.

Each estimator is represented by a single point whose coordinates correspond to

- mean root-mean-square derivative error,
- mean derivative smoothness.

This representation illustrates the trade-off between derivative accuracy and smoothness.

G.9. Overall Workflow

The estimator evaluation framework processes all available reference datasets sequentially.

For each dataset, the following sequence is performed

```

reference dataset
|
v
estimator execution
|
v
metric evaluation
|
v
dataset outputs
|
v
dataset figures
|
v
prediction tables

```

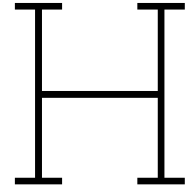
After all datasets have been processed, campaign-level aggregation is performed.

The resulting campaign information is used to generate

- campaign tables,
- campaign figures,
- Pareto representations.

G.10. Framework Outputs

Together, the generated outputs provide both dataset-level and campaign-level descriptions of estimator performance.



Estimator Implementations

The present appendix summarizes the internal estimator architectures, assumptions and update mechanisms used by the estimator framework.

H.1. Estimator Architecture

Estimator implementations are separated from the evaluation framework.

Estimator modules own

- estimator states,
- recursive update equations,
- derivative estimation,
- one-step prediction.

They do not perform

- file input and output,
- metric evaluation,
- plotting,
- campaign aggregation,
- reference generation.

H.2. Standard Interface

All estimators receive the dataset representation and return a common result structure containing

- predicted magnetic field,
- estimated magnetic-field derivative,
- estimator identity.

This interface enables all estimators to be evaluated by the same validation framework.

H.3. Causality Assumptions

All estimators are causal.

At timestamp

i

(H.1)

only measurements and estimator states available up to that instant may be used.

Future measurements are not accessible.

Consequently, estimators operate according to

history

|

v

state update

|

v

Magnetic-field derivative estimate

|

v

one-step prediction

Actual timestamps are used when computing prediction horizons.

No fixed sampling interval is assumed.

H.4. Allowed Information

Estimators may use

- timestamps,
- measured magnetic-field components,
- internally generated histories,
- internal estimator states.

Estimators do not use

- rotation axis,
- angular velocity,
- reference trajectories,
- future measurements.

These quantities belong exclusively to the offline validation framework.

H.5. Variable-Time Derivative

Magnetic-field derivatives are evaluated using the actual timestamp differences

$$\Delta t_i = t_i - t_{i-1}. \quad (\text{H.2})$$

No constant sample-rate assumption is imposed.

Samples satisfying

$$\Delta t \leq 0 \quad (\text{H.3})$$

do not contribute to derivative updates.

H.6. One-Step Prediction

Predictions are generated according to

$$B_{\text{pred}}(i) = B_{\text{state}}(i-1) + \dot{B}(i-1)\Delta t. \quad (\text{H.4})$$

Initialization uses the first available measurement.

Consequently

$$B_{\text{pred}}[i] \quad (\text{H.5})$$

corresponds to the prediction associated with timestamp

$$t_i. \quad (\text{H.6})$$

H.7. Finite-Difference Estimator

The finite-difference estimator provides the baseline model.

Measured field

|

v

Finite difference

|

v

Magnetic-field derivative estimate

|

v

One-step prediction

No filtering is applied.

H.8. Exponential Moving Average Estimator

The exponential moving average estimator first smooths the measured magnetic field before estimating the derivative.

Measured field

|

v

Exponential moving average

|

v

Finite difference

|

v

Magnetic-field derivative estimate

|

v

One-step prediction

Parameter

$$\alpha \quad (\text{H.7})$$

controls the amount of smoothing applied to the measurements.

H.9. Predictive Exponential Moving Average Estimator

The predictive exponential moving average estimator maintains estimates of both the magnetic field and its derivative.

Prediction

|

v

Measurement update

|

v

Derivative update

|

v

Next prediction

The magnetic-field estimate satisfies

$$B_{\text{est}}(i) = (1 - \alpha)B_{\text{pred}}(i) + \alpha B_{\text{meas}}(i). \quad (\text{H.8})$$

The derivative estimate satisfies

$$\dot{B}_{\text{est}}(i) = (1 - \beta)\dot{B}_{\text{fd}}(i) + \beta\dot{B}_{\text{est}}(i - 1). \quad (\text{H.9})$$

Parameter

$$\alpha \quad (\text{H.10})$$

controls the balance between prediction and measurement information.

Parameter

$$\beta \quad (\text{H.11})$$

controls derivative smoothing.

H.10. Alpha-Beta Filter

The alpha-beta filter maintains estimates of the magnetic field and its derivative.

Prediction

|

v

Innovation

|

v

State update

|

v

Derivative update

The innovation vector is

$$r(i) = B_{\text{meas}}(i) - B_{\text{pred}}(i). \quad (\text{H.12})$$

The state update becomes

$$B_{\text{est}}(i) = B_{\text{pred}}(i) + \alpha r(i). \quad (\text{H.13})$$

The derivative update becomes

$$\dot{B}_{\text{est}}(i) = \dot{B}_{\text{est}}(i-1) + \beta \frac{r(i)}{\Delta t}. \quad (\text{H.14})$$

H.11. Future Extensions

Additional estimators may be incorporated while preserving the common interface.

Placeholders currently exist for

- regression estimators,
- polynomial estimators,
- Savitzky-Golay estimators,
- custom estimators.

Parameter-Tuning Framework

The present appendix summarizes the framework used to evaluate estimator parameters over the complete campaign and determine suitable configurations.

I.1. Framework Architecture

```

Estimator
|
v
Parameter grid
|
v
Campaign evaluation
|
v
Metric computation
|
v
Campaign aggregation
|
v
Ranking
|
v
Tuning figures
|
v
Best parameters

```

The framework evaluates estimator parameters over the complete campaign using the reference datasets generated previously.

I.2. Parameter Search

Parameter tuning is performed through exhaustive grid evaluation.

Each parameter combination is evaluated independently over all available datasets.

I.3. Candidate Parameters

The tuning framework currently considers

- exponential moving average estimators,

- predictive exponential moving average estimators,
- alpha-beta filters.

I.4. Exponential Moving Average Parameters

The exponential moving average estimator is characterized by

$$\alpha. \tag{I.1}$$

The parameter grid consists of

0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

I.5. Predictive Estimator Parameters

The predictive exponential moving average estimator is characterized by

$$\alpha \tag{I.2}$$

and

$$\beta. \tag{I.3}$$

The parameter grids are

alpha: 0.1 ... 0.9
beta : 0.75 ... 0.95

All parameter combinations are evaluated.

I.6. Alpha-Beta Parameters

The alpha-beta filter is characterized by

$$\alpha \tag{I.4}$$

and

$$\beta. \tag{I.5}$$

The same parameter grids are used.

I.7. Campaign Evaluation

For each parameter combination, the estimator is executed on every reference dataset.

The resulting metric values are aggregated to produce campaign-level statistics.

I.8. Performance Metrics

Parameter combinations are evaluated using

- root-mean-square vector error,
- root-mean-square angular error,
- root-mean-square magnetic-field derivative error,
- derivative smoothness.

I.9. Parameter Ranking

Parameter combinations are ranked according to

1. root-mean-square magnetic-field derivative error,
2. derivative smoothness,
3. root-mean-square vector error.

I.10. Tuning Figures

For every parameter combination, comparison figures are generated for

- magnetic-field quantities,
- magnetic-field derivatives.

I.11. Exported Outputs

The parameter-tuning framework generates

- tuning results,
- parameter rankings,
- comparison figures,
- best-parameter tables.

I.12. Overall Workflow

```
reference datasets
|
v
parameter grid
|
v
estimator execution
|
v
metric computation
|
v
campaign aggregation
|
v
ranking
|
v
tuning figures
|
v
best parameters
```

The resulting outputs are used to select suitable estimator configurations for subsequent analyses.