NTNU

Kunnskap for en bedre verden

Experts in Team

TTK4852 - Small satellites

SpinSat

Authors:
Henrik Berg-Johansen, Erik Haatuft, Jonas Heen Storeheier, Mads Mølbach,
Mori Adrian Rosland, Simon Sandvik Lee

April, 2025

Contents

List of Figures

List of Tables

List of Acronyms

EIT Experts in Teamwork

NTNU Norwegian University of Science and Technology (Norges teknisk-naturvitenskapelige universitet)

ADCS Attitude control and determination system

IMU Inertial measurement unit

RC Remote Controlled

RTOS Real Time Operating System

API Application Programming Interface

SDK Software Development Kit

DDM Device Driver Model

GAP Generic Access Profile

GATT Generic Attribute Profile

BLE Bluetooth Low Energy

I2C Inter-Integrated Circuit

SPI Serial Peripheral Interface Bus

IoT Internet of Things

PCB Printed Circuit Board

BOM Bill Of Materials

CW Clockwise

CCW Counter-clockwise

DC Direct Current

AC Alternating Current

BEC Battery Elimination Circuit

PID Proportional-integral-derivative

BLDC Brushless direct current

Abstract

SpinSat is an educational 2U CubeSat model designed to demonstrate the physics of reaction wheels for satellite attitude control. It was developed as an interdisciplinary project in the course Experts In Teamwork (TTK4852) at NTNU. The project demanded close collaboration across mechanical design, electronic systems, software development, and motor integration, underscoring the role of teamwork in complex engineering projects. The team designed and 3D-printed the frame, developed a custom PCB with an nRF52840 MCU and LSM9DS1 IMU, implemented software using C and the Zephyr RTOS for state management and BLE communication, and integrated a BLDC motor drive system. Key subsystems were tested individually using a development kit, and the final demonstration successfully showed satellite body rotation via the reaction wheel, achieving the core educational objective. However, difficulties in flashing the custom PCB prevented full system integration. Despite this, SpinSat provided valuable insights into satellite subsystems and collaborative engineering, underscored the complexities of RTOS adoption and hardware-software integration, and laid a foundation for future work to increase its educational value.

1 Introduction

This project is a part of the evaluation *TTK4852 - Small Satellites.*

Experts in Teamwork (EIT) is a course designed to foster interdisciplinary collaboration among students from various fields of study. The primary objective is to simulate a real-world working environment, where team members may not know each other beforehand and bring diverse experiences to the table. In this setting, we had to learn to work together effectively, using our differences as strengths to achieve a common goal.

Our project took place within the "Small Satellites" village, one of several technical programs offered through EIT. This village centers on the design, development, and application of miniature satellites, giving students the freedom to explore a topic of their choice. With approximately 94% of all satellites now classified as small and increasingly favored for their cost-effectiveness in commercial applications, this field is highly relevant to the future of space exploration and utilization [17].

To ensure that everyone on the team could contribute meaningfully, we made deliberate use of the exercises provided by the learning assistants during the first three village days. Some of these activities were carried out collaboratively on a shared Miro board, where we discussed important themes such as individual strengths, backgrounds, brainstorming ideas, and comparing different approaches. Through this process, we not only gained a better understanding of each other's perspectives but also discovered that our team had a strong practical and technical orientation. This insight played an important role in shaping and supporting our chosen topic.

Building on this understanding, most of the team members expressed a clear preference for creating something tangible rather than focusing solely on theoretical aspects. After several discussions, we narrowed our options down to four possible projects: Monitoring of the northern and southern poles, Yappsat (a "speaking" satellite that reports its surroundings), designing our own satellite, and SpinSat (a satellite to demonstrate rotation). We decided that monitoring the poles would be less suited to our team's strengths, so we focused on creating a satellite. This led us to two final concepts: SpinSat and Yappsat. Since Yappsat was more complex, we chose SpinSat, as it allowed everyone to contribute and seemed more feasible to complete within the course time-frame. The expertise of our team, including printed circuit board (PCB) design, low-level programming, systems design, embedded electronics, physics, and networking, was well matched to the requirements of this project.

1.1 Goal of our project

The main objective of our project, SpinSat, was to build a physical satellite model that could demonstrate how satellites rotate in space. Although the model is not space grade, it resembles a small, transparent 2U CubeSat, allowing us to showcase internal components typically found in real small satellites. To make the demonstration both educational and relevant to real satellite operations, our goal was to implement key features such as wireless control of the satellite, allowing it to rotate by a specified number of degrees and maintain its orientation. Elaborating on the main goal of demonstrating how a satellite rotates in space, we wanted to achieve the following sub-goals:

- Develop and implement an attitude hold control mode. In this mode, the satellite is not spinning and maintains its attitude, responding to external disturbances in an appropriate manner.
- Develop and implement a maneuver mode. Here, the satellite executes pre-programmed maneuvers, such as rotating a specific angle (clockwise or counter-clockwise) about its axis.
- Develop and implement a free control mode. In this mode the satellite is controlled directly by an operator using a control interface, such as a joystick or a smartphone.

Although full implementation of this functionality was not achieved, it remained a central fo-

cus throughout the project. This hands-on approach aligns directly with the village's focus on small satellite design and application, providing a practical and accessible platform to explore core principles of satellite dynamics, embedded hardware, software systems and communication.

Because our project was highly practical, most of our time in the course was spent on implementation rather than purely theoretical analysis. However, this paper also includes some theoretical background to explain how the system works.

2 Theory

To design, build, and operate a functional model of a rotating satellite, a solid understanding of several theoretical concepts is essential. This section presents the key theoretical foundations that guided our technical decisions and shaped the development of SpinSat. While the project itself was primarily practical, the performance and functionality of our system depended on principles from satellite dynamics, electromechanics, motor drives, and control systems.

2.1 Attitude determination and control system (ADCS)

For many missions in space it is necessary to be able to know and control the orientation of the satellite (its attitude). For example, the deployment of a small satellite usually results in an unwanted rotational motion, which needs to be stopped. This is called detumbling. Another example is an imaging satellite, which should be able to orient itself such that its camera is pointed towards the desired target.

It is not a trivial task to change the orientation of a satellite in space. Firstly, one needs to determine the actual attitude with respect to a reference coordinate system. This can be accomplished using a variety of sensors, but so-called inertial sensors like accelerometers, gyroscopes and magnetometers are often used. These sensors are typically gathered in an inertial measurement unit (IMU).

When the current attitude is determined, a rotation towards the desired attitude can be initiated. Hence a torque must be applied to the satellite. There exists several methods of generating this torque. For example, an electromagnet installed on the satellite will interact with the Earths magnetic field, with which it tends to align. The dipole moment of the electromagnet, and hence the magnitude of the torque acting on the satellite, can be controlled by adjusting the electric current through the coil. This is the working principle behind magnetorquers [9].

Another method is to install a reaction wheels on the satellite. These are wheels which freely rotate around their axis, and hence carry angular momentum. By adjusting their rotation, using for instance a DC motor, the satellite will start to rotate in the opposite direction due to the conservation of a physical quantity called angular momentum [23]. A simple quantitative explanation of this effect is provided in the next section.

2.2 Attitude control using reaction wheels

In this section we introduce the basic physics of rotation and attitude control using reaction wheels. In rotational dynamics, two important physical quantities are torque τ and angular momentum L. Formally they are vector quantities, defined as follows:

$$\vec{\tau} = \vec{r} \times \vec{F}, \qquad \vec{L} = \vec{r} \times \vec{p} \tag{1}$$

where $\vec{r}$ is the position vector, $\vec{p}$ is the linear momentum and $\vec{F}$ is the force [23]. Here, $\times$ denotes the vector cross product. In the following, we will assume that any rotation occurs about a single, fixed axis. We can therefore drop the vector symbols and distinguish opposing vectors by a minus sign. A fundamental result from classical mechanics is that

$$\frac{dL}{dt} = \tau^{(\text{ext})}, \tag{2}$$

where $\tau^{(\text{ext})}$ now denotes the *net* torque acting on the body due to external sources [23]. This result follows from the definitions in (1) and Newton's laws of motion. The quantity dL/dt is essentially the rate of change of angular momentum with respect to time. From this we can immediately see that if $\tau^{(\text{ext})} = 0$, the angular momentum will remain constant in time. We have thus showed the important result: if the net external torque acting on a rigid body is zero, its angular momentum is conserved.

Until now this is rather abstract, but it turns out that the angular velocity ω of the body is closely related to its angular momentum. For a rigid body (an extended body which does not change its shape) it can be shown that the angular momentum and angular velocity are related by

$$L = I\omega \tag{3}$$

provided that the mass is symmetrically distributed about the axis of rotation. Here, I is the moment of inertia of the body. It generally depends on the geometry of the body, more precisely its mass and how it is distributed [23]. Dynamically it plays a role similar to the mass in the sense that the larger it is, the harder it will be to rotate the body.

Consider now a system consisting of a single reaction wheel with moment of inertia I_w mounted on a satellite with moment of inertia I_s. For simplicity, suppose the wheel and satellite is free to rotate about only a single axis. The angular momenta of the satellite L_s and reaction wheel L_w are given by

$$L_s = I_s\omega_s \quad \text{and} \quad L_w = I_w\omega_w, \tag{4}$$

where ω_s and ω_w are the angular velocities of the satellite and reaction wheel, respectively. In the absence of external torques acting on the system, the total angular momentum $L \equiv L_s + L_w$ is conserved. Suppose the angular velocity of the reaction wheel is increased with $\Delta\omega_w$. By conservation of total angular momentum, $\Delta L = 0$, and hence $\Delta L_s = -\Delta L_w$. Thus the angular velocity of the satellite changes by

$$\Delta\omega_s = -\frac{I_w}{I_s}\Delta\omega_w. \tag{5}$$

This shows that the satellite gains an angular velocity directed opposite to that of the reaction wheel, and the proportionality factor I_w/I_s determines its magnitude. This provides control of rotational motion about one axis. Of course, to provide full three-axis control, a minimum of three reaction wheels is required [9].

2.3 Motor Drive

Understanding how to generate and control rotation is essential for satellite attitude control systems. In this section, we provide a theoretical overview of the key components used to drive SpinSat's reaction wheel. We first introduce brushless direct current (BLDC) motors, explaining their basic working principles and characteristics. We then describe the role of the electronic speed controller (ESC) in managing motor operation, followed by a discussion of important motor parameters such as torque constants and speed ratings.

2.3.1 BLDC motors

There are several types of electric motors. When a DC source is available, such as a battery, the most natural choice is a DC motor. There are multiple types of DC motors, but a detailed discussion is beyond the scope of this work. However, two of the alternatives are brushed motors and BLDCs. The advantages of BLDCs compared to brushed motors are their high speed and acceleration capabilities, greater efficiency, lower electrical noise, and reduced torque ripple. Brushed motors also use physical brushes for commutation, which BLDCs do not possess. This leads to higher lifetime for BLDCs compared to brushed motors. However, their disadvantage is that they are more costly than brushed motors, and that their control is more complicated [13].

Power and torque are fundamental physical concepts for understanding the operation of BLDC motors. These motors are widely used in satellite systems due to their ability to provide precise and efficient rotational control.

The mechanical power output P in motors can be written as

$$P = T\omega, \tag{6}$$

where T is torque generated by the motor and ω is its angular velocity [7]. It is important to note that this calculated power is at the motor shaft, and differs from the electrical power that is supplied to the motor, as a certain amount of energy is always lost during conversion processes [7].

The rotor in a BLDC motor is a permanent magnet which creates a magnetic field. Furthermore, the stator/armature circuit consists of coils designed to carry alternating current (AC) and create a rotating magnetic field. This setup can be seen in Figure 1.

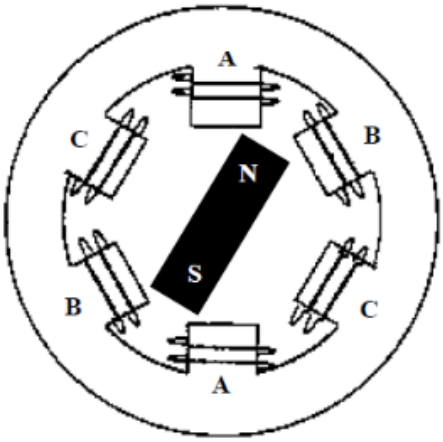


Figure 1: BLDC setup. Figure taken from [3, p. 2].

The back emf is the voltage generated by the motor as it rotates. It opposes the voltage that is applied to the motor, as per Lenz law [7]. This opposing voltage causes lower current at higher speeds. The magnitude of the back emf can be written as

$$E = 2NlrB\omega, \tag{7}$$

where N is the number of turns per phase, l is the length of the rotor, r is the internal radius of the rotor and B is the magnet flux density [3, p. 4]. Furthermore, the torque equation is given by

$$T_{\mathrm{e}} = \frac{1}{2}i^2\frac{dL}{d\theta} - \frac{1}{2}B^2\frac{dR}{d\theta} + \frac{4N}{\pi}Brl\pi i, \tag{8}$$

where ω is the angular velocity, i is the phase current, L is the phase inductance, θ is the electrical rotor position and R is the phase resistance [3, p. 4]. The purpose of presenting (7) and (8) is to illustrate that the back emf is directly proportional to the motor speed, and torque production is nearly directly proportional to the phase current. In simpler terms, applying high voltage yields high speed, while producing high torque draws high current.

2.3.2 ESC

By coordinating the phases in the correct order, the stator creates a rotating magnetic field. The magnet in the rotor will rotate and follow this magnetic field [3]. In order to do this, the system must have an ESC. This subchapter will very briefly show how an ESC connects to a BLDC motor.

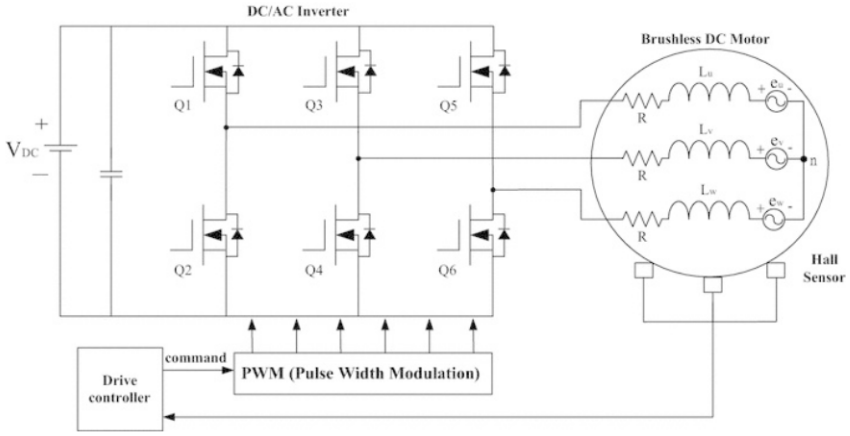


Figure 2: Circuit of BLDC with ESC. Figure taken from [14].

Figure 2 shows a BLDC motor with a driver. There is a DC voltage input, and this is transformed into AC by an inverter with six transistors. On the right, one can observe the equivalent circuit of the BLDC, with resistance, inductance, and back emf [14]. It is important to note that this is just an example of an ESC. Elaborating on the exact structure of the ESC used in this project is beyond the scope of this work.

2.3.3 Motor Parameters

Motors often have some parameters described in the datasheet, which are important for predicting the motor's performance in different applications. In this subsection, only the most basic ones will be introduced and discussed. Simply put, these parameters describe how torque and speed relate to current and voltage.

The so-called velocity constant K_v represents how many RPMs the motor will produce per volt of back emf. Mathematically, this can be written as

$$K_{v,\mathrm{RPM}} = \frac{\omega_{\mathrm{RPM}}}{V_{\mathrm{bemf}}}, \tag{9}$$

where ω_{RPM} is the angular velocity (given in RPM) of the motor and V_{bemf} is the back emf[11]. In SI units (where angular velocity is measured in rad/s) this becomes

$$K_{v,\mathrm{SI}} = \frac{\omega}{V_{\mathrm{bemf}}} = K_{v,\mathrm{RPM}} \cdot \frac{2\pi}{60}\frac{\mathrm{rad/s}}{\mathrm{RPM}}, \tag{10}$$

where ω is angular velocity, now given in rad/s. Furthermore, the torque constant K_t is defined as

$$K_{\mathrm{t}} = \frac{T_{\mathrm{m}}}{I_{\mathrm{a}}}, \tag{11}$$

where T_m is the produced torque and I_a is the phase current[11]. This means that K_t indicates how much torque the motor produces per amp of phase current in the armature. It turns out that K_t and K_v are inversely proportional. If one increases, the other decreases [11], as in

$$K_{v,\mathrm{SI}} = \frac{1}{K_{\mathrm{t}}}. \tag{12}$$

To get an idea of what the torque load and current draw will be, it can be useful to model the load. According to [16, p. 38], a fan can by modeled as the following

$$T_L = k\omega^2, \tag{13}$$

where T_{L} is the torque load and k is a constant. One can see from (13) that the torque load increases with the square of the angular velocity in this model. It is therefore reasonable to assume that the reaction wheel can be modeled in a similar manner, although it is important to note that this is an assumption.

2.4 Control Theory

A Proportional-Integral-Derivative (PID) controller is a widely used control algorithm that continuously calculates an error value as the difference between a desired setpoint and a measured process variable. The controller then applies a correction based on three terms: proportional, integral, and derivative.

The proportional term produces a correction that is proportional to the current error, meaning that a larger error results in a stronger corrective action. The integral term accounts for the accumulation of past errors over time, helping to eliminate steady-state offsets that may persist if only proportional control is used. The derivative term predicts future error behavior based on its rate of change, providing a damping effect that improves system stability and reduces overshoot.

Together, these three components enable the PID controller to react to current, past, and anticipated future errors, resulting in a balanced and responsive control strategy. PID controllers are valued for their simplicity, robustness, and effectiveness, and are applied in a wide range of fields including robotics, industrial automation, and spacecraft control [5].

2.5 Batteries

Lithium Polymer batteries are often used in RC (remote controlled) planes, cars and boats. This is because the high current discharge rating and energy capacity they possess relative to their size[12].

However, these batteries also have disadvantages. The lifespan of LiPo batteries are approximately 300 - 500 cycles, while the comparable LiFe batteries can last up to 2000 - 3000 cycles. LiFe batteries also have better thermal performance than LiPo batteries, meaning that they can opeate sufficiently in a higher temperature range. In addition, the chemistry is more sensitive compared to Li-on and LiFe batteries. In short, LiPo batteries can be a fire hazard if not handled and stored properly [12].

There are several parameters that describe a battery. Some of the most important are mentioned below [24].

- **Cell Count**, or S, describes how many battery cells are in series in a battery system. The higher the amount of batteries in series, the higher the output voltage will be. A cell is the basic building block of a battery. One LiPo cell will have a voltage of 4.2V when it's fully charged.
- **Capacity** of a battery is typically measured in mAh, even though it can be measured in Wh as well. 1000 mAh means that the battery can discharge 1A for one hour.
- **Discharge Rating** is a parameter that describes how fast the battery can be discharged without damaging the battery. For example, a 500 mAh battery with a 10C rating can safely discharge $500\text{mAh} \cdot 10\text{C} = 5\text{A}$.

3 Design and implementation

This section presents the planned design and implementation of our cube satellite.

3.1 System Overview

Figure 3 shows a block diagram of the implemented design and provides a brief overview of how the subsystems interact with each other.

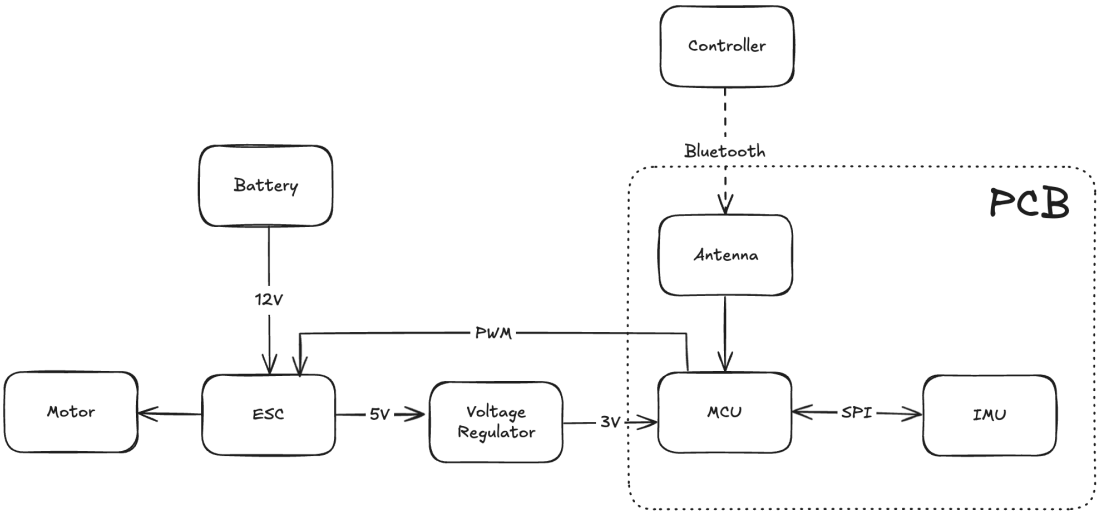


Figure 3: Block diagram of the complete system of SpinSat.

The microcontroller unit (MCU) serves as the central processing unit of the satellite, managing all peripheral components, including the inertial measurement unit (IMU), antenna, and electronic speed controller (ESC). It receives commands from a Bluetooth Low Energy (BLE) enabled controller, such as a smartphone, and executes predefined maneuvers accordingly. The ESC processes a pulse-width modulated signal from the MCU to control the motor's reaction wheel speed, setting the satellite's orientation. All components from figure 3 are mounted on the satellite's structural body, except for the external controller.

A 12V external battery powers the motor, ESC, and MCU. The ESC includes a standard feature that provides a 5V output on one of its pins. This 5V output is routed through a voltage regulator to step it down to the MCU's operating voltage of 3V.

The MCU, antenna, and IMU are integrated into the satellite's custom-printed circuit board (PCB). This PCB is specifically designed to interconnect the satellite's electronic components efficiently. Communication between the MCU and IMU is for example done through a serial peripheral interface bus (SPI).

The following chapters explore the implementation of each subsystem, detailing the hardware architecture, software control strategies, and integration methods. The results of these implementations are presented in Section 4.

3.2 Mechanical and structural components

3.2.1 3D - model

To manufacture a satellite body, a 3D-model was drawn in Onshape CAD[20]. This modeling was important to ensure that all components fit in our satellite. After this, the model was printed with a Prusa MK4S 3D - printer at Omega Verksted in NTNU Trondheim. Its dimensions are 20 cm x 10 cm x 10 cm. A schematic of the satellite body is shown in Figure 4, and the exact dimensions of both the body and the flywheel are shown in Figure 26.

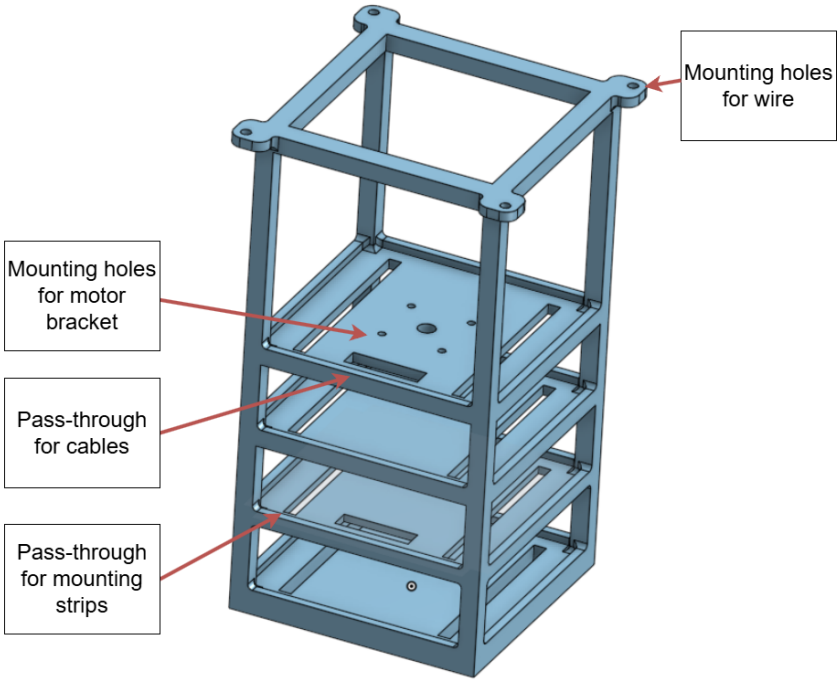


Figure 4: Body of Satellite

Furthermore, the flywheel, which can be viewed in Figure 5, is also 3D - printed. The dimensions are 9 cm x 6 mm. We chose to maximize its diameter, while still being able to fit it into the satellite body. The reasoning for this was to make sure the wheel had enough moment of inertia to be able to rotate the satellite. In addition, it has mounting holes for extra bolts and screws in case we needed to further tune the moment of inertia. These were not used in practice.

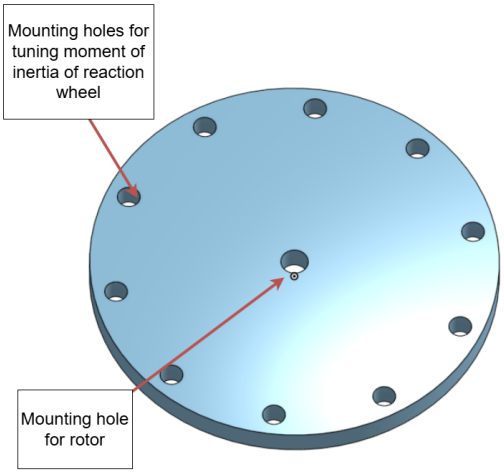


Figure 5: Flywheel

3.3 Software

Hardware alone is not enough, we also need software to conduct the hardware and ensure the satellite behaves as intended. To achieve this, we developed our own low-level software, which allowed us to efficiently manage memory, process sensor data, control the motors, and handle communication between the satellite and its controller. This sounds great, but software development can be challenging, especially when you're unsure where to start, how to design the overall system, which

frameworks/tools/programming languages to use or not use and how to collaborate efficiently as a team. These choices are crucial, as they directly impact both the development process and the final product.

3.3.1 Programming language

For this task, we needed to work closely with hardware under tight resource constraints. Given these requirements, the team selected C as our primary programming language. C is both simple and powerful, serving as the backbone of much of today's technology. As Bjarne Stroustrup describes, "C is close to the hardware, but not tied to any specific hardware." [22, p. 4]. C is widely used in embedded systems around the world and is supported by extensive documentation. Other languages like C++, Zig and Rust were also mentioned as possible candidates, but in the end the team's familiarity and experience with C won. By choosing C, we were able to optimize for both development speed and code performance, without the need to learn a new language.

3.3.2 Operating System

But how would our C code actually run on the satellite? The software team spent considerable time weighing two main options: working directly with a bare-metal solution or using a real-time operating system (RTOS). Bare-metal programming would allow us to avoid third-party frameworks and stick to less complex and familiar tools, potentially speeding up development since we'd only write the code we needed. However, this approach could become challenging as we'd have to implement everything from scratch, and compatibility issues with our custom electronics could make troubleshooting more difficult. On the other hand, using an RTOS like Zephyr, initially seemed excessive for our relatively simple system. However, Mads and Simon had prior experience with Zephyr from their work at Orbit NTNU, and the RTOS offered several advantages. This included device tree abstractions, easier project structure and execution, integrated debugging/monitoring tools, and helpful SDKs. It was also both open source and has been used in space grade software before.

3.3.3 Architecture

At the start we had to plan how we would implement all of our criteria for the project in software. Firstly we thought about how the satellite would perform different actions when needed, and we ended up with a simple state machine shown with states shown below and in figure 6. These states would be access through a BLE signal.

- OFF: The satellite is powered off.
- $INIT$: The satellite is booting up.
- $CONNECTING$: The satellite is waiting for a GAP connection with a Central (this is described in more detail later).
- $IDLE$: The satellite is listening for actions to perform.
- $SPIN_{FORWARD}$: The satellite is spinning clockwise.
- $SPIN_{REVERSE}$: The satellite is spinning counter clockwise.
- $HOLD$: The satellite tries to hold the rotation it is in even when acted on by an external force.

This worked great, but there were still some uncertainties. As mentioned earlier, we wanted to support a variety of maneuvers, but creating separate states for each possible spin could quickly become redundant, since many would be similar in implementation. To optimize this, we decided

to also divide the BLE signal into events, which are described in the list 3.3.3. This meant that the signal that the satellite would receive, would contain the state, event and data. For example, if the satellite receives a command to spin 90 degrees, the relevant signal would specify this in a byte array, and the system would execute the maneuver accordingly. How the signal received and split into the byte array is described in a later section.

- $DEFAULT$: Default behavior, if $SPIN_{FORWARD}$ or $SPIN_{REVERSE}$ spin with data as percentage power.
- $DEGREES$: Spin to a specific degree if $SPIN_{FORWARD}$ or $SPIN_{REVERSE}$. With data as data as degrees of spin.
- NO_STOP: Spin until interrupted if $SPIN_{FORWARD}$ or $SPIN_{REVERSE}$.

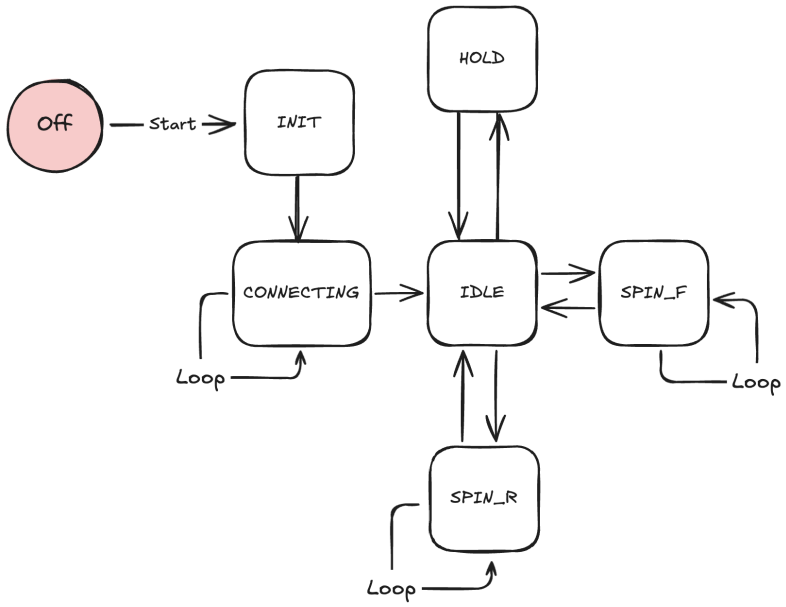


Figure 6: Envisioning the state machine

The functionality for each of these states was implemented in separate modules, each accessible through its own API ("Application Programming Interface"). An API provides a clear separation of logic, allowing us to abstract away details and simplify interactions between different layers of the software. In our design, the main state machine acts as the central conductor, requesting tasks from the various modules and managing and sending the data around to the correct destinations. Our overall software architecture is illustrated in Figure 7. The modules consisted of the Bluetooth API, Inertial Measurement Unit API, Motor/PID API and the Storage API. It is important to note that when bluetooth is mentioned, we are talking about BLE ("Bluetooth Low Energy"). The implementation of each of these modules are described more in detail later in 3.3.4.

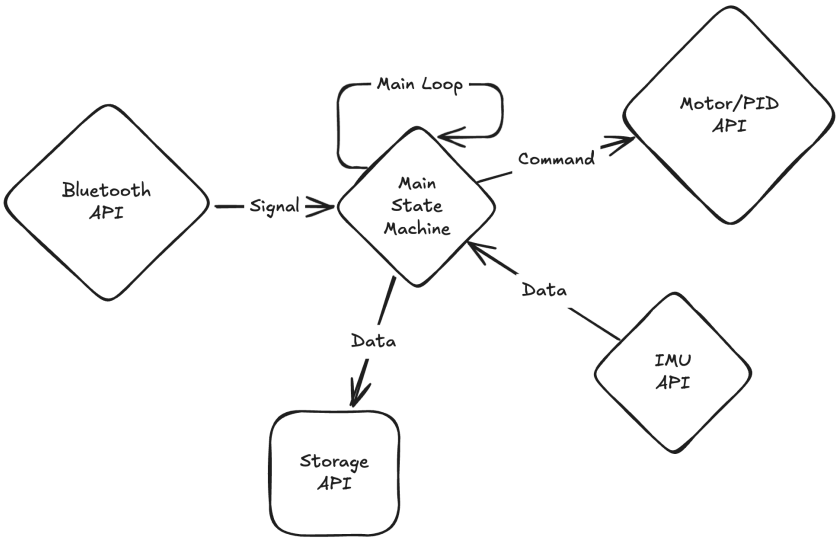


Figure 7: The main communication between different modules in software.

Lets use the example of commanding the satellite to spin 90 degrees clockwise again, which is illustrated in figure 8. Assuming the system is running, connected to the controller, and operating under ideal conditions, the process begins with the controller sending a command to the satellite's main state machine. The state machine receives the signal, stores the relevant data, such as the 90-degree rotation and the direction—and determines which state to transition into. In this case, it switches to $SPIN_{FORWARD}$ and recognizes a $DEGREES$. The state machine then processes the 90-degree value and initiates the spin by calling the motor control function, which uses a PID controller. This function interacts with the IMU API to retrieve gyroscope data, allowing the system to monitor the rotation in real time, and with the Motor API to drive the motor in the correct direction until the satellite has completed the 90-degree turn. The state machine now switches itself to $IDLE$.

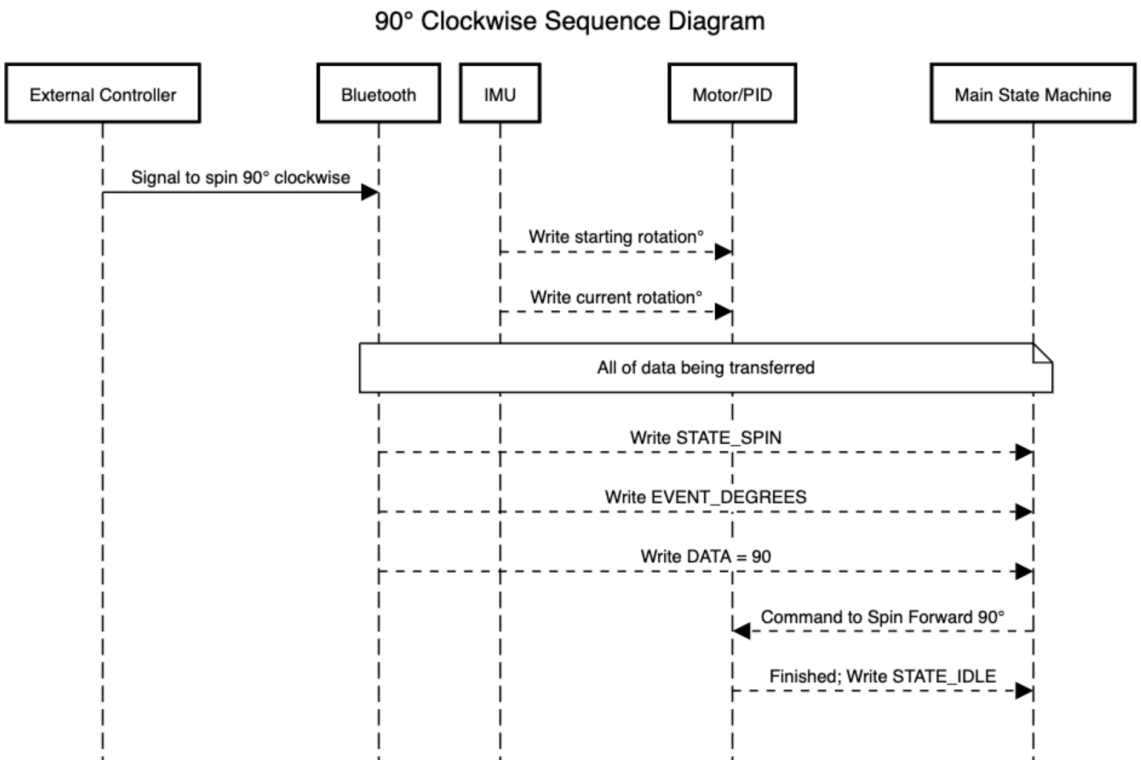


Figure 8: The main flow for the 90 degree example.

3.3.4 Dependencies

To implement these APIs, we developed several custom libraries and utilized existing drivers provided by Zephyr, all while keeping our overall architecture in mind. There were configurations and a directory structure we had to do follow in Zephyr to enable these (Appendix B). To create these configuration flags, we needed to define KConfig and CMakeList.txt files for the build system as described in the README file of the project (Appendix B.6).

Drivers
For the drivers, we chose to use some of Zephyr's built-in abstractions, including the DDM ("Device Driver Model") and Devicetree. This approach allowed us to configure the necessary parameters in a configuration file (Appendix B.1) and easily access the devices we needed. The main device driver required was for the LSM9DS1 IMU sensor, which Zephyr's driver library supports over both SPI and I2C protocols. To select the appropriate protocol, we configured our board overlay file (Appendix B.3). With these abstractions in place, integrating the sensor was straightforward: we simply included the relevant Zephyr sensor header and used the $DEVICE\_DT\_GET\_ANY$ macro to access the sensor in our code (Appendix B.4).

Libraries
For the libraries, we created a separate one for each main API: Bluetooth, Motor/PID, and IMU. All of these are configured in the same file as the drivers (Appendix B.1). While the implementation details vary depending on the use case, each library follows a consistent approach of utilizing both heap and static memory. For each main component, such as the state machine, we defined a corresponding struct and provided functions to mutate/use it. Further documentation and implementation details can be found in Appendix B.6. It is worth noting that the Motor/PID API utilizes Zephyr's PWM configuration to control the motor, the IMU uses the LSM9DS1 sensor driver and the main state machine uses GPIO configurations to interface with the hardware LEDs.

3.3.5 From Sensors to Software

As mentioned earlier, there are two main protocols for communication between hardware components: I2C and SPI. For our PCB, we chose to use SPI, as it offers higher data rates and lower communication overhead compared to I2C[1]. This is easily configurable like mentioned in the driver section 3.3.4.

3.3.6 From Controller to Satellite

Lastly, the controller needed a way to communicate with the satellite. As mentioned earlier, we chose BLE for this purpose. This is because it is straightforward to configure in Zephyr and well suited to our project's requirements. BLE is commonly used in IoT devices because it offers lower power consumption compared to classic Bluetooth albeit reduced data throughput, which was sufficient for our needs. For the GAP ("Generic Access Profile") architecture, we wanted connection-oriented communication. This made the most sense because we wanted to establish a connection between the controller and the satellite. With this the satellite acts as the peripheral, advertising its presence and accepting connections, while the controller functions as the central, scanning for and connecting to the satellite's advertisements.

For data representation and exchange between devices, we looked into GATT ("Generic Attribute Profile"), which organizes all the readable and writable attributes shared between the controller and the satellite. For example, the Bluetooth signal used to write states is defined as shown in Appendix B.5. GATT is a standard way of exchanging information, especially between LE devices like ours. It was also quite simple to implement in code, as seen in the Appendix. The only attribute we needed to configure was a write to the state machine. The signal would therefore just be a byte array of 16 bits that gets processed to the correct information. Lets use the example from figure 8 one more time. We first send the byte array signal hexadecimal $0x0e5a$. It gets divided it into two 8 bit integers, 0x0e and 0x5a. From here we can extract the event and state from the first integer, and the data from the second. Since 0x0e is 15, our event would be $DEGREES$ (event 1) and state would be $SPIN_{FORWARD}$ (state 5) (both are 0 indexed from 3.3.3). The second integer 0x5a means 90 in data. The rest of the states, events and data work the same.

3.4 Printed Circuit Board

The printed circuit board (PCB) serves to package and wire the satellite's electronics, which are essential for controlling the satellite's orientation and communicating with the host controller. To achieve this, the PCB is composed of multiple modules, each addressing a specific functional requirement.

Designing a PCB is a complex process. Consequently, many companies offer pre-built circuit boards or *development kits* that can perform many of the functions that a custom PCB would. These pre-built solutions were certainly an option when building our satellite. However, we chose to design our own PCB to expand our skill set and gain new knowledge, which was one of the primary reasons for choosing this satellite project.

3.4.1 Design process

PCBs are designed by first creating a *circuit schematic*. The schematic specifies the components that make up the PCB, how they are connected, and ensures that all components are wired correctly. Once the schematic is complete, a list of the required components is compiled in the *bill of materials* (BOM)[18].

Next is the design of the physical layout and wiring of the PCB, a phase known as *place and*

[1] I2C is typically more suitable for systems with numerous peripherals, as it is not constrained by the number of available chip select lines, improving scalability.

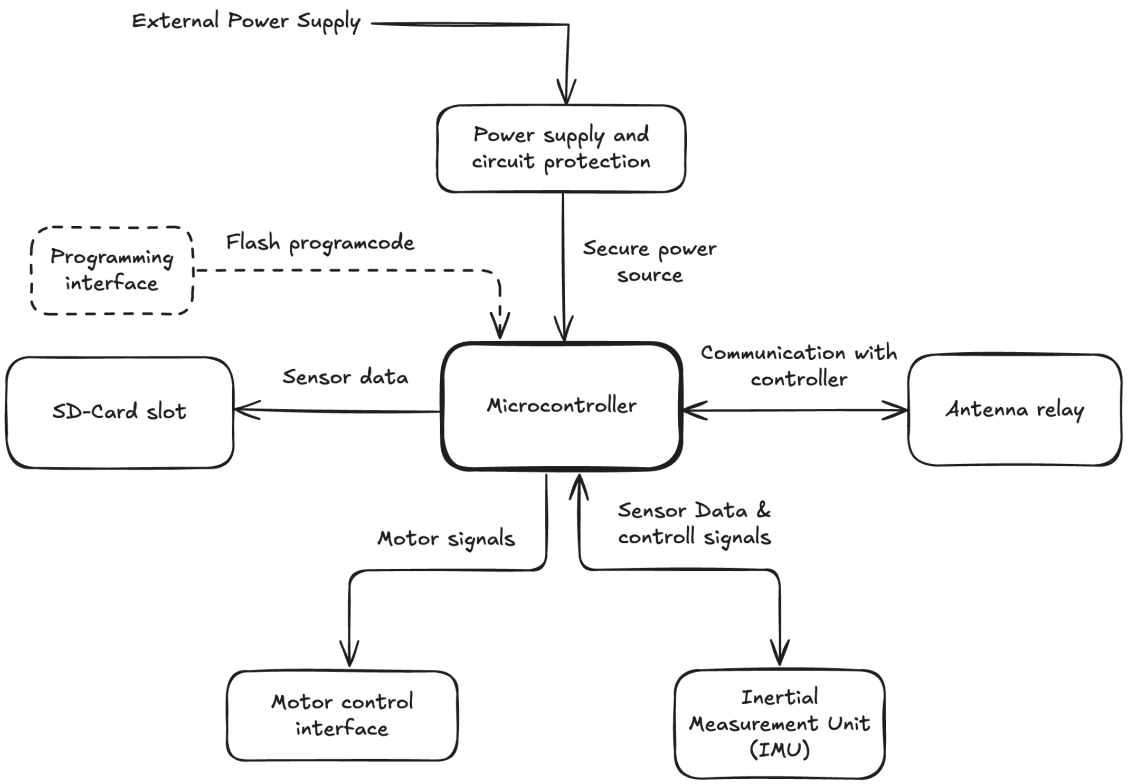


Figure 9: Block diagram of the electric circuits on the PCB. The microcontroller is the primary module, and the rest sub-modules. The programming interface (dotted block), is a one-time used interface program the microcontroller.

route. During this stage, the layers of the PCB are defined. A typical PCB consists of four layers: two signal layers, one ground layer, and one power layer. Once the layers are established, the components listed in the BOM are placed on the board and their connections are routed according to the schematic. After the process of placing and routing, the PCB is ready for manufacturing.

During the manufacturing of the PCB, the components listed in the BOM are also ordered. Once both the manufactured PCB and the components are received, the final step is to solder the components onto the board. After this, the PCB is complete and ready for testing.

3.4.2 Circuit schematic

The PCB consists of one primary module and five sub-modules. Figure 9 provides an overview of the modules and how they are connected.

The microcontroller unit (MCU) executes the software described in Section 3.3, controlling all submodules and communicating with the controller via the antenna relay.

The power supply and circuit protection module converts the external power supply to a safe, stable voltage for the MCU and peripherals. This module filters electrical noise from the power line, prevents reverse-polarity issues, and protects electronics from short-circuiting. As outlined in Sections 2.2, the MCU requires data on the satellite's acceleration and position to determine its orientation and changes thereof. The inertial measurement unit (IMU) captures this sensor data and transmits it to the MCU[2].

To log data captured by the MCU, the printed circuit board (PCB) includes an SD card slot,

[2]The LSM9DS1 IMU features an accelerometer, gyroscope, and magnetometer, capturing data across all spatial dimensions. The MCU processes this data to derive the required spatial and temporal parameters for controlling spacecraft attitude.

enabling the storage of operational records.

The satellite's motor, responsible for maintaining or adjusting orientation, is controlled by an electronic speed controller (ESC). The MCU interfaces with the ESC through a dedicated motor control interface.

The MCU operates based on its programmed instructions. To program the MCU, pin headers are provided for flashing program code into its memory.

Figure 17 in Appendix A shows the PCB's circuit schematic, implementing the block diagram in Figure 9.

The voltage regulator incorporates a low-dropout regulator to step down 5V to 3V, the operating voltage for the nRF52840, the selected MCU. It also includes a reverse-polarity protection circuit to prevent current from flowing in the wrong direction, which could damage the circuit.

Capacitors on signal and power lines filter noise to ensure stable transmission. The underlying circuit theory is beyond the scope of this report.

The capacitors and inductors in the antenna relay, shown at the top of Figure 17, form the impedance matching network. This network optimizes power efficiency during radio wave transmission. The component values are derived from the MCU manufacturer's reference design, as recommended[21].

Table 1 presents the complete bill of materials (BOM) for the PCB.

Table 1: Bill of Materials for the PCB

Reference	**Value**	**Component**
C1,C2	0.1uF	Capacitor
C3	1.0pF	Capacitor
C4	1.2pF	Capacitor
C5,C6,C7,C9,C11,C13,C14	100nF	Capacitor
C8	10uF	Capacitor
C10	10nF	Capacitor
C12	1uF	Capacitor
D2	PWR_LED	Diode
D3	STATUS1	Diode
D4	STATUS0	Diode
J1	ESC	Connector
J2	Conn_01x06_Pin	Connector
J3	GSD090012SEU	Connector
J4	Conn_Coaxial_Small	Connector
L1	4.7nH	Inductor
L2	2.2nH	Inductor
Q1	FDN340P	Transistor
Q2	TPS73250DCQR	Regulator
R1	49kR	Resistor
R2	31kR	Resistor
R3,R7	10kR	Resistor
R4,R5,R6	330R	Resistor
U1	nRF52840-QFXX	Microcontroller
U3	LSM9DS1	Sensor

The most critical component in the bill of materials (BOM) is the microcontroller unit (MCU). Selecting the appropriate MCU is vital, as it defines the satellite's supported functionalities. Given our satellite's requirement for Bluetooth connectivity, choosing an MCU with integrated Bluetooth simplifies the design significantly.

For this purpose, we selected the nRF52840 Bluetooth MCU from Nordic Semiconductor. This MCU meets our memory and performance needs for running the software and includes a built-in Bluetooth module, requiring only an antenna for wireless connectivity[3]. Additionally, our team had access to a development kit featuring a nRF52 series chip (Appendix B.8), enabling parallel software and PCB development, making it an ideal choice.

The BOM also includes three status LEDs (D2–D4) to visually indicate the satellite's power status and operational state. These are particularly useful during development to verify that the code is functioning correctly.

3.4.3 Place and route

Figure 18 in Appendix A shows the layout and routing of the schematic shown in Figure 17. The PCB measures 89.5 mm in length and 45.0 mm in width. In Figure 18, red traces represent the top signal layer, while blue traces indicate the bottom signal layer (underside of the board). To securely attach the PCB to the satellite's structural body, the board includes four mounting holes, one at each corner.

To minimize signal noise in the antenna relay (located at the top of Figure 18), it is positioned far from other circuitry. This increased separation reduces crosstalk between the antenna line and other transmission lines, enhancing the signal strength transmitted by the antenna. This is critical, as the antenna is highly sensitive to disturbances[10].

3.4.4 3D model and soldered PCB

Figure 10 depicts the 3D model of the PCB after component placement and routing. The silver pads visible in the image are soldering points for the PCB's components.

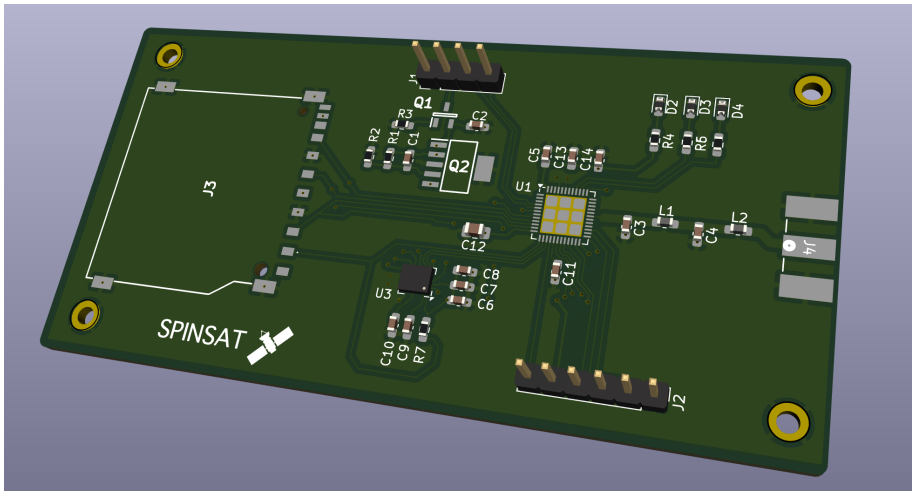


Figure 10: 3D model of the finished PCB.

The PCB was manufactured by PCBWay using lead-free hot air solder leveling (HASL). The HASL finish facilitates easier soldering of components[4]. Figure 11 shows the completed PCB with all components soldered in place.

[3]The nRF52840 features a 64 MHz Arm Cortex-M4 processor, 1 MB Flash, 256 KB RAM, and a 2.4 GHz transceiver for robust connectivity.

[4]PCBWay provides a brief explanation of HASL: https://www.pcbway.com/blog/technology/7_surface_finish_in_PCBWay_.html

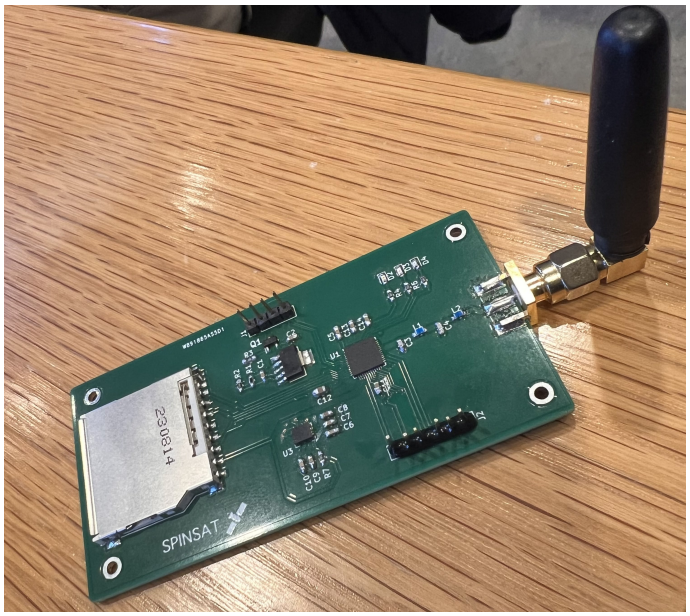


Figure 11: The final PCB with all parts soldered.

Due to the use of very small components, hand-soldering was not feasible. Instead, the components were soldered using a reflow oven, also known as a solder oven. This method is ideal for small surface-mount components, as it eliminates the need for manual dexterity[5].

3.5 Motor Drive

3.5.1 DC motor

Due to several of the advantages discussed in section 2.3.1, we chose to go with a BLDC motor for this project. Unlike brushed motors, BLDC motors have no physical brushes that can wear out, significantly increasing operational lifetime and reducing maintenance requirements, which is very good for satellite applications. Furthermore, BLDC motors offer precise control of speed and torque, which is important for tasks such as attitude control, where fine adjustments are often necessary.

For our BLDC motor, we chose the "PROPDRIVE v2 2830 1000KV Brushless Outrunner Motor", which can be viewed in Figure 12. When starting this process, we wanted to be as flexible as possible, and therefore chose a motor that could cover most of our needs. Its high power rating and relatively high torque provide flexibility for implementing large satellites and flywheels. This particular motor model is also quite inexpensive, which aligned well with our limited budget. Simplifying the mounting of the motor to the satellite body was also important, and this motor had a bracket which made this trivial.

[5]More about reflow ovens: https://en.wikipedia.org/wiki/Reflow_oven

Figure 12: PROPDRIVE v2 2830 1000KV Brushless Outrunner Motor
Figure taken from [19].

The motor has a maximum power of 370 W and 12 electromagnetic poles [19]. Furthermore, the motor has a velocity constant $K_v = 1000$ RPM/V. One can see from this parameter, that for every volt of back emf over a phase, the motor will produce 1000 RPM at no load. In SI units, where one uses rad/s, the velocity constant is $K_{v,\text{SI}} = 104.72 \frac{\text{rad/s}}{\text{V}}$. The corresponding torque constant, K_t, can be computed using (12):

$$K_t = \frac{1}{K_{v,\text{SI}}} = \frac{1}{104.72 \frac{\text{rad/s}}{\text{V}}} = 0.00955 \text{ Nm/A}.$$

This means that for every amp in a phase of the motor, the motor will output 0.00955 Nm of torque. The values of K_t and K_v will be used further in Chapter 4.5 to estimate the voltage and current in the motor.

3.5.2 ESC

For our electronic speed controller, we have selected "Aerostar 50A RVS G2 32bit (2-4S) Electronic Speed Controller w/Reverse Function & 4A 5 6V SBEC", which can be seen in Figure 13. As mentioned in 2.3.2, the motor needs an ESC to function. Applying a fixed voltage, as done with brushed motors, is not sufficient. Therefore, this ESC is essential for our project.

Figure 13: Aerostar 50A RVS G2
. Figure taken from [8].

This ESC has a reverse function, which is essential when the aim is to rotate the satellite in both the clockwise (CW) and counterclockwise (CCW) directions.

In addition, the ESC has the following protection mechanisms:

- **Overheat protection**: When the temperature exceeds 100°C, the power output is reduced.
- **Overload protection**: The ESC will cut power if the load suddenly increases[2].

These protection mechanisms allows the satellite to operate safely, while minimizing risk of fire or damaging the motor. The datasheet does not state exactly what the power ratings must be for the overload protection to trip.

The control strategy is not specified in the datasheet. Therefore it is difficult to say anything about the AC voltages and currents without making assumptions about strategy, sampling time and switching frequency. It is important to note that this driver is a bit of a black box, unless one attempts to "hack" it, for instance by reading or flashing firmware.

Furthermore, it uses MOSFETs to convert DC to AC [2].

The ESC also contains a battery elimination circuit (BEC), which eliminates a need for an extra power supply to the microcontroller or PCB. This supplies 5 V.

BLDCs and ESCs are quite complicated, and a deep analysis of their working method is beyond the scope for this paper.

3.5.3 Battery

For the the power supply, a "3s 1000mAh - 30C - Gens Ace Soaring G-Tech XT60" LiPo battery was chosen. This can be seen in Figure 14.

It was chosen because of its high power to weight ratio, offering high energy output while maintaining a low mass of only 89 grams. In addition, the XT60 connectors fit perfectly with the Aerostar ESC, ensuring a secure connection.

Figure 14: 3s 1000mAh - 30C - Gens Ace Soaring G-Tech
. Figure taken from [6].

Furthermore, it consists of three LiOn cells conntected in series, that supplies a voltage of $4.2\text{V} \cdot 3\text{cells} = 12.6\text{V}$ when it is fully charged.

As can be seen by the specification, the battery can provide a maximum current of $1000mAh \cdot 30C = 30A$[6].

4 Results

This section outlines the results from the SpinSat demonstration on the final day of the EiT course. It evaluates the satellites functioning and rotational behavior according to the goals presented in

Section 1.1.

4.1 SpinSat

Figure 15 shows the final build of SpinSat. The top level houses the engine and reaction wheel, the second level contains the ESC, and the bottom two levels hold the battery and microcontroller. All components fit neatly into their respective compartments, with wiring routed easily through the pass-through holes in the satellite's body.

Figure 15: Close-up view of SpinSat.

To demonstrate attitude control, we designed a plexiglass case with a top-mounted hinge to suspend the satellite in the air. The hinge allows the satellite to rotate freely around the vertical axis, while the plexiglass provides protection against any parts that may detach and fly off during rotation. The satellite is fastened to the hinge using a wire threaded through the four mounting holes at the top of its body. Figure 16 shows the complete test setup.

Figure 16: SpinSat mounted in its case, ready for testing.

4.2 Demonstration of Satellite Rotation

SpinSat successfully demonstrated the use of reaction wheels for spacecraft attitude control on the final day of the EiT presentation. A demonstration can also be viewed in this YouTube video [1]. Using an Arduino Uno to control the BLDC motor, we showcased how the torque from the reaction wheel induced a corresponding rotation in the satellite body, confirming the theory on reaction wheels outlined in Section 2.2[6]. While formal rotation tests were not conducted, the live demonstration clearly illustrated the physical impact of the reaction wheel on the satellite's orientation, effectively mimicking the rotational behavior of a real satellite in a microgravity environment[7]. Despite facing some challenges, which will be discussed later, the project successfully conveyed the practical application of reaction wheels in satellite attitude control. In an educational context, this demonstration serves as an effective tool for visualizing physics concepts, such as the conservation of momentum.

[6]The reason for using an Arduino Uno instead of the custom PCB on presentation day is explained in Section 4.4.

[7]As formulated by NASA: "Microgravity is the condition in which people or objects appear to be weightless". Source: https://www.nasa.gov/learning-resources/for-kids-and-students/what-is-microgravity-grades-5-8/.

4.3 Connectivity

One of the primary objectives of the project was to enable remote control of the satellite. Using the Nordic Semiconductor development kit (Appendix B.8), we successfully established a wireless connection between a controller interface and the satellite.

Throughout development, we primarily relied on the nRF Connect Mobile application for iOS (Appendix B.8), which acted as a host to scan for Bluetooth devices such as SpinSat. Once connected, we were able to interact with SpinSat's embedded software, allowing us to change the satellite's states. By sending a specific byte array signal, the satellite's state was altered, as explained in detail in section 3.3.6.

Although we were unable to complete testing with the final custom PCB due to hardware flashing issues, the successful demonstration on the development kit confirmed the functionality of the underlying software.

4.4 PCB

The testing phase for the printed circuit board (PCB) yielded a mix of promising outcomes and challenges that warrant further investigation. Upon applying power to the PCB, the board successfully powered on, indicating that the fundamental power delivery system, including the voltage regulation and distribution network, functioned as intended. All visible indicators, such as power LEDs, highlighted correctly, and no immediate signs of electrical faults, such as short circuits or overheating components, were observed. This confirmed that the PCB's core hardware infrastructure, including its power management subsystems, was operational at a basic level.

However, a critical issue arose during attempts to flash the program code onto the PCB's microcontroller. Despite multiple efforts using serial-wire programming and debugging error codes from the PCB, the flashing process consistently failed. This prevented us from loading the necessary software to evaluate the PCB's full functionality. The inability to program the microcontroller halted further testing of the software-hardware integration, leaving several subsystems untested in their operational context.

Preliminary analysis suggests that most of the PCB's subsystems—such as the communication interfaces, sensor modules, and peripheral circuits—are likely functional, based on their successful response to power and initial continuity checks. However, without successful programming, these subsystems could not be tested under real-world conditions or with the intended software. Potential causes for the programming failure include issues with the microcontroller's bootloader, incorrect configuration of the programming interface, or a hardware fault in the programming circuitry, such as a misconfigured or damaged serial communication line from souldering.

In summary, while the PCB demonstrated basic functionality by powering on successfully, the inability to flash program code represents a significant obstacle. The power delivery and core hardware appear reliable, but the programming failure prevented comprehensive testing of the board's capabilities. Further diagnostic efforts will focus on resolving the flashing issue to enable complete evaluation and optimization of the PCB's performance.

4.5 Motor and Battery Test

To ensure that the current through the battery was operating within safe limits, a test was conducted. The reaction wheel was attached to the motor, the battery was attached to the ESC, and an Ardiuno Uno was used to generate the PWM - signals.

To perform this test, a WM150 Wattmeter was used.

Table 2: Power, Current, Duty Cycle at different PWM levels

PWM (μs)	1070	1300	1400
Duty Cycle	0.07	0.30	0.40
Peak Current (A)	0.2	1.2	2.4
Peak Power (W)	2	13	26
Steady State Current (A)	0.1	0.7	1.1
Steady State Power (W)	1	7	12
Battery Voltage (V)	11.3	11.3	11.3

As can be seen in Table 2, the currents and power are below what the system can deliver. It is important to note that these measured quantities are DC - values on the battery - side, and not AC - values in the motor. However, these measurements can still give some insight.

When the duty cycle increases, the average voltage supplied to the motor increases, resulting in a higher motor speed. Although the speed increase was not measured quantitatively, an increase was visually observed.

It can be observed that the current through the battery is increasing with speed. This is expected, as equation (13) indicates that current increases with speed in such a model.

This test shows that the motor, ESC and battery are overdimensioned for the current application. However, it also shows that the current and power drawn remain well within the safety margin. In addition, the power system can also be used for larger satellites and reaction wheels.

An informal measurement of the satellites angular velocity was also made in order to compute the angular velocity of the reaction wheel, and hence of the DC-motor. By the robust technique of visually counting the number of revolutions in a given time interval, the angular velocity was estimated to be

$$\omega_s \approx 360^\circ/\text{s} = 60 \text{ RPM (clockwise direction)},$$
$$\omega_s \approx 288^\circ/\text{s} = 48 \text{ RPM (counter-clockwise direction)}.$$

To apply (5), the moments of inertia I_s and I_w of the satellite and reaction wheel, respectively, must also be known. Determining these values to a high degree of accuracy is beyond the scope of this project, so we will simply use the values from Onshape CAD, as an estimate. These values are shown in Figure 27 and Figure 28, located in Appendix C. They are

$$I_s \approx 2.0 \cdot 10^{-3} \text{ kg m}^2,$$
$$I_w \approx 9.9 \cdot 10^{-5} \text{ kg m}^2.$$

Inserting these values into (5), the angular velocity of the reaction wheel comes out to be

$$\omega_w \approx 7.3 \cdot 10^{3^\circ}/\text{s} \approx 1217 \text{ RPM (clockwise direction)},$$
$$\omega_w \approx 5.8 \cdot 10^{3^\circ}/\text{s} \approx 970 \text{ RPM (counter-clockwise direction)}.$$

Although these values are just estimates, they do show that in our design the reaction wheel gains a significant angular velocity compared to the satellite.

Based on these estimates, the back EMF can be approximated using (9), which yields

$$V_{\text{bemf,cw}} = \frac{1217 \text{ RPM}}{1000\text{RPM/V}} = 1.21 \text{ V}.$$

By using (6), one can also estimate the torque output of the motor during steady state. This is given by

$$T_{\text{cw}} = \frac{1 \text{ W}}{127.4 \text{ rad/s}} = 7.84 \text{ mNm}.$$

It is also possible to estimate the phase currents by using (8) and the results found in subchapter 3.5.1. This yields the result

$$I_{\mathrm{a,cw}} = \frac{T_{\mathrm{m}}}{K_{\mathrm{t}}} = \frac{7.84\mathrm{mNm}}{0.00955\ \mathrm{Nm/A}} = 0.82\mathrm{A}.$$

This calculation shows that the phase current is below the rated current of 28 A, as stated in [19].

5 Discussion

5.1 Project Achievements and Challenges

Although we were ultimately unable to complete the full implementation of SpinSat, a significant amount of work was accomplished throughout the project. Both the hardware and software components were developed, and individual subsystems were successfully tested using a development kit.

However, due to PCB-related issues and limited available time, we were unable to finalize the project. The primary issue encountered was the inability to flash the software onto the MCU on the housemade PCB, preventing thorough testing and verification of the system. Despite these setbacks, we demonstrated key aspects of SpinSat's intended functionality. By using an Arduino Uno for the demonstration, we successfully showcased how a satellite can rotate in space using reaction wheels, achieving one of the main goals outlined in the introduction.

Due to issues with the PCB, we also postponed the implementation of the Storage API, as shown in the software architecture diagram (Figure 7). Without a functioning development kit, we were unable to test this functionality, and it was not as critical as other parts of the software. The Storage API was primarily intended for storing results and running simulations.

5.2 Reflections on Design Choices

Several key design decisions had a significant impact on the project's progress and outcome, both positive and negative.

One early technical decision was to base the system on the real-time operating system (RTOS) Zephyr. Zephyr offered appealing advantages, including built-in drivers for Bluetooth, PWM control, and sensor communication protocols. Additionally, it provided an integrated build and flash system, which, in theory, would simplify the development process and accelerate system integration by reducing the need for manual setup. However, in practice, adopting Zephyr introduced considerable complexity. Building and flashing the firmware often proved difficult, with vague and uninformative error messages — such as "error: unknown error" — appearing during the build process. These build issues made diagnosing problems very difficult and consumed a significant amount of project time. Furthermore the motor control PWM driver provided by Zephyr did not function correctly, forcing us to rework much of the already implemented motor code in Zephyr.

While Zephyr's automated build and flash system was attractive in theory, it proved unreliable and contributed significantly to the technical difficulties we encountered during development. In hindsight, programming the MCU using a bare-metal approach would likely have simplified development, provided better control, and reduced the need for extensive debugging.

From a hardware perspective, the PCB design lacked sufficient consideration for in-circuit debugging and flashing. The PCB was not equipped with appropriate debug headers or test points, which would have allowed easier troubleshooting and programming. As a result, when flashing issues arose, diagnosing and solving them became significantly more difficult. Including proper programming and debugging interfaces in the PCB design would have been essential for troubleshooting and would have significantly accelerated getting the system operational.

Another major challenge stemmed from a miscommunication between the hardware and software teams. The hardware team assumed that the software team would handle the programming setup for the MCU, while the software team assumed that the hardware team would design the PCB to be flashable(Plug and play). This misunderstanding meant that, by the time the system needed to be programmed, no clear flashing procedure existed, and neither side had prepared the necessary setup. We attempted to flash the MCU indirectly by using a development kit as an external programmer, but unfortunately, this method was unsuccessful. This experience highlights the critical importance of establishing clear responsibility boundaries and ensuring cross-team communication for essential system functions.

Despite the challenges encountered, it is important to emphasize that a substantial amount of work has been successfully completed. The hardware platform, core software structure, and subsystem functionalities are all in place and have been thoroughly developed and tested individually. Although final integration and full system validation were not achieved within the project timeframe, the majority of the groundwork has been laid. If future teams choose to continue this project, they will be able to build upon a strong foundation, focusing primarily on resolving a few specific hardware issues and completing system integration, rather than having to start from scratch. In this sense, SpinSat provides a robust and well-prepared starting point for further development.

In the following chapter, we will discuss the specific technical limitations identified during the project and outline potential directions for future development.

6 Societal Benefit, Limitations and Future Research

The SpinSat project successfully demonstrated the fundamental principles of satellite rotation, achieving the primary goal set at the project's outset. This section begins by outlining the current societal benefits of SpinSat before moving on to discuss its limitations and opportunities for future research. By reflecting on these aspects, we aim to show how SpinSat can evolve into an even more realistic, robust, and educational platform, ultimately strengthening its value as a learning tool.

6.1 Societal Benefit

Taking into account all the problems we faced during development and final implementation, the SpinSat project still provides an excellent educational platform for understanding how hardware and software work together to form an integrated functional system. By combining practical elements such as motor control, PCB design and soldering, 3D-printing, and applying physical principles, students gain valuable hands-on experience with both space orientation and electrical systems.

Making our implementation public and showcasing the system to the rest of the village served as an important demonstration of key concepts, including how electronic components interact, how software governs physical behavior, and how these systems are applied to solve real-world engineering challenges. As a learning tool, SpinSat offers a tangible and engaging introduction to interdisciplinary engineering, bridging the gap between theoretical studies and practical applications.

Additionally, the project provided a real-world example of how interdisciplinary collaboration is not only beneficial but absolutely necessary to design and build complex systems such as small satellites. Already in the early stages, it became clear that significant progress could not be made without leveraging the unique knowledge and experience of each group member. By choosing a highly practical and technical project, it became more obvious how each team member could contribute meaningfully, which further motivated participation and knowledge-sharing across disciplines.

Beyond education, projects like SpinSat have broader societal benefits by contributing to technological literacy in a growing and critical field. As described in the course description, with the increasing importance of satellite technology in communication, earth observation, navigation, and environmental monitoring, developing future engineers and scientists who understand the core principles behind satellite systems is essential for societal progress [[17]].

SpinSat also visually demonstrates one of the fundamental concepts in both classical and modern physics, namely conservation of angular momentum. Understanding how this concept is applied in a real-world situation such as spacecraft attitude control, makes it clear that a seemingly abstract physical concept has a wide area of applicability. Current and future students in the physical sciences may find motivation in seeing the equations they derive on the blackboard come to life, gaining a deeper appreciation for how theoretical knowledge can lead to real technological achievements.

Finally, by completing EiT, students completing the SpinSat project are encouraged to to develop

the innovation and problem-solving skills that are essential for their future careers. Facing real-world engineering constraints, such as cost, time, and component selection, fosters the kind of creative and resilient mindset needed to address future challenges in every technology sectors. Thus, SpinSat not only advances education but also helps to prepare engineering graduates to contribute meaningfully to society's technological and scientific needs.

6.2 Limitations and Future Research

6.2.1 Choice of Components

Suppliers of parts have lead time on their components. Therefore it was important to purchase the needed components as fast as possible, to ensure that the parts were available for testing. Because of this, there was limited time to make all necessary calculations for motor and drive system before deciding on which parts to purchase. As briefly mentioned in the results section, the chosen DC motor and battery were more than powerful enough for this project. Furthermore, the angular velocity of the satellite in a real mission does not necessarily not to be quite as large as what was achieved in this project. This indicates that there remains work to be done regarding optimization of the physical size and weight of the chosen components, which we did not have time for in this project.

6.2.2 Testing Environment

Although our satellite successfully demonstrated rotation about a single axis, its behavior on Earth is not necessarily representative of how it would behave in space. For example, we were unable to avoid several sources of external torque which would not be present in the vacuum of space, like air resistance. Also, the satellite was suspended from a thread. As the satellite was spinning, the thread would twist causing a slight torque in the opposite direction. While these factors did not negatively impact the results of this project, there is a chance they would have if more sophisticated control techniques were used, as initially planned.

Also, the undesirable effects of an asymmetrical mass distribution, which could be problematic in space, are harder to observe when the satellite is suspended in the Earth's gravity field. The effects of an asymmetrical mass distribution were not discussed the theory section. It can be shown that it may lead to a wobble about the rotational axis [23]. Although the satellite was assembled with symmetry in mind, we certainly can not say that this was actually accomplished to a high degree of accuracy. A slight wobble was actually observed in testing, indicating an asymmetrical distribution of mass.

6.2.3 Implementation of Precise Attitude Control

It would also be beneficial to integrate our PCB to the rest of the system. Using the Arduino for demonstration purposes was not optimal since we were not able to make use of all the software developed in Zephyr. As a further result of this, we were not able to implement all the maneuvers and components required for a satellite to autonomously navigate in space. While SpinSat successfully demonstrated basic rotation, the full range of attitude control such as holding a specific attitude and performing pre-programmed maneuvers remains to be implemented.

An important area for future development is the integration of a PID controller to manage the satellite's rotation and stabilization more precisely. This would enable the satellite to automatically reach and hold a desired orientation, correcting for any drift or external disturbance. Testing such a system in a designated, low-friction environment would provide a more realistic evaluation of its performance.

Another valuable improvement would be the integration of a controller capable of two-way communication with the satellite, ideally operating over a realistic radio frequency band between 300 MHz

and 40 GHz, which is commonly used in spacecraft communication systems [15]. Implementing this would allow for remote control and maneuvering of the satellite over realistic distances, closely mimicking actual space operations. Additionally, receiving data from the satellite in real-time would enable more precise maneuvering by providing feedback on its position, orientation, and system status. This feedback could then be used to further refine control algorithms and improve the overall accuracy and responsiveness of the system.

6.2.4 Energy Optimization

The inclusion of onboard sensors and data transmission would not only enhance control, but also support energy optimization. For small satellites in LEO, which typically have a lifespan of around 7–10 years [4], efficient energy use is critical. By using internal data to manage power consumption dynamically, it is possible to extend operational life and improve system reliability. Developing an energy-aware control strategy could therefore be a major step toward creating a functional and sustainable small satellite system.

6.2.5 Fault Detection and Error Handling

Feedback data could be used to implement fault detection and diagnostic mechanisms, making the satellite more robust and reflective of real satellite systems. By continuously monitoring parameters such as angular velocity, motor current, temperature, and system response times, it becomes possible to identify anomalies that may indicate malfunctions or performance degradation. For example, a sudden drop in motor speed without a corresponding command could suggest a mechanical issue or power fault, while deviations in expected sensor readings could point to calibration drift or hardware failure. Integrating such fault detection systems would not only improve the reliability of the satellite but also enable autonomous error handling, where the satellite can take corrective actions such as resetting a subsystem or switching to a backup mode without manual intervention. This kind of self-awareness is a key feature of modern space technology and would be a valuable addition in future iterations of the project.

7 Conclusion

SpinSat aimed to demonstrate satellite attitude control using reaction wheels in a wirelessly commanded, 2U CubeSat-style model. We designed and 3D-printed the frame, selected and partially assembled core hardware—including a BLDC motor, ESC, battery, and a custom PCB with an nRF52840 MCU and IMU—and developed a Zephyr RTOS-based software architecture for state management and BLE communication, which ran successfully on a development kit. Although the custom PCB powered on and validated its power systems, flashing issues prevented programming, hindering full system integration. Motor and battery tests confirmed safe operation under load, proving the concept of reaction-wheel-driven rotation, even without a fully integrated control loop.

Despite integration challenges, SpinSat successfully demonstrated reaction wheel mechanics for spacecraft control and provided valuable experience in building electronic systems and interdisciplinary collaboration. While the final system awaits resolution of specific hardware issues, SpinSat serves as a strong foundation for future development, aiming to implement more advanced control and fulfill its potential as an educational platform for space systems engineering.

7.1 Personal Application of Professional Knowledge

In this section, we elaborate on how each group member has applied their professional knowledge and expertise to contribute to the project's success. By leveraging our interdisciplinary strengths, we have ensured a comprehensive approach to the project, with each member bringing unique perspectives and skills to the table. The following subsections detail the individual contributions

of Henrik, Erik, Jonas, Mads, Mori, and Simon, emphasizing their specific roles, professional competencies, and the impact of their work on the project's overall outcomes.

7.1.1 Henrik

As a Cyber Security student, I was able to apply my knowledge of programming and system behavior throughout the project. My previous experience from hands-on courses proved particularly valuable during both the design and build phase of the satellite. In addition, since this course emphasized teamwork and collaborative problem-solving, I was able to draw on the team and practical skills I developed during my military service. These skills were particularly useful during the development of the satellite mount used in the SpinSat demonstration.

Although I was somewhat disappointed that we were unable to get the custom PCB fully operational, which prevented me from applying my skills in communication technology, the SpinSat project still provided valuable learning opportunities. Working within a team with diverse technical backgrounds allowed me to broaden my knowledge, particularly in the areas of embedded systems and low-level programming.

7.1.2 Erik

As a physics student, this project served as an excellent way for me to apply and solidify my knowledge of classical physics. In addition, I was introduced to several engineering disciplines which rely on physics concepts I have studied from a more theoretical perspective in the past. This has opened my eyes to the wider world of applied physics.

Specifically, I was able to contribute to the development of SpinSat by sharing my understanding of fundamental concepts in physics, such as conservation of angular momentum, which are essential in understanding how a satellite maneuvers in space. This particularly suited my deep interest in how conservation laws arise as consequences of the underlying symmetries in nature. In addition, I was heavily involved in the discussions pertaining to the physical dimensions of the satellite-flywheel system, identifying the most important physical parameters which determine the performance of the system, and how they are related through conservation laws. Also, my knowledge was used to identify which physical quantities are of interest when controlling a spinning satellite. This influenced the teams decision-making regarding which sensors were necessary to carry onboard SpinSat.

To me, working on this project demonstrated the importance of being able to communicate your ideas in a language that everybody can understand. Being surrounded by engineering students, all of them excellent in their respective fields, I was impressed by their ability to explain advanced concepts, of which I have no knowledge, in simple ways. This motivated me to improve my own communication abilities, because I realized how important they are when working in an interdisciplinary team.

I am grateful for having had the opportunity to participate in this valuable learning experience.

7.1.3 Jonas

During the course process, I have implemented knowledge from both my electrical engineering degree at NTNU Gjøvik and NTNU Trondheim, as well as my career experience.

For designing the 3D model in Onshape, I utilized skills that I acquired during an electrical engineering job in Oslo. Having this competency proved quite valuable in this project.

Furthermore, I applied knowledge from my education during my motor testing using the Arduino Uno. Even though my specialization is power engineering, which does not directly focus on microcontrollers, relevant subjects like ELE2131 at NTNU Gjøvik provided me with a solid foundation to understand the basics of embedded systems.

My degree also helped me to select the motor, ESC and battery for the system. In addition, it enabled me to measure and estimate the voltages and currents in the battery and BLDC. The subject TET4120 played an important role in helping me understand the electrical behavior of our motor drive.

7.1.4 Mads

As a student with a background in Electrical Engineering at NTNU (ELSYS), specializing in embedded systems, I have developed a deep understanding of how electrical systems are designed and implemented. In addition to my academic training, I have gained significant experience through university projects, personal initiatives, and my professional work, all of which proved highly beneficial to this project.

Specifically, I was able to contribute to the development of SpinSat by leveraging my broad experience across all subsystems. This gave me a solid overview of the entire project, and because I had prior exposure to most of the components we were working on, I could help the team avoid common pitfalls and make better design decisions.

I also contributed significantly through my experience in low-level programming—gained from my involvement in Orbit NTNU and from working as an embedded engineer at Texas Instruments. This background was particularly valuable when developing the various APIs required for the project and during the setup and configuration of the Zephyr RTOS.

Throughout the project, I have learned a great deal. My C programming skills have improved significantly, and I've gained a deeper understanding of how to structure and set up projects in Zephyr, and in general. Additionally, I've spent countless hours debugging both hardware and software issues—an experience that, while sometimes frustrating, taught me a lot about problem-solving, persistence, and systems thinking.

7.1.5 Mori

With my background in electrical engineering at NTNU (ELSYS), I have a foundational understanding of how electronic systems are designed. Through personal and university projects, I have gained practical skills in translating ideas into solutions. These projects involved programming microcontrollers, analyzing antenna arrays, and designing electrical circuits. This experience proved particularly valuable when designing and building SpinSat.

During the brainstorming phase for SpinSat, I drew on my prior experience to propose design solutions and avoid common pitfalls when creating electronic systems. This allowed me to contribute to the team's ideation process.

After brainstorming, my primary responsibility was designing the PCB, a task suited to my skill set, as I had designed PCBs previously. My experience enabled me to work autonomously on the PCB, freeing up my teammates to focus on their own tasks. While the work itself was autonomous, previous experience aided me in communicating important design decisions effectively, such as selecting the most suitable power solutions for the PCB.

The complexity of the PCB pushed me to learn new concepts to ensure the system's functionality. I also gained insights into the importance of clear communication. For instance, when designing the programming interface, later discussions revealed that the team was not fully aligned on its implementation, highlighting areas for improved coordination.

In summary, this project allowed me to leverage my experience in circuit design while further developing my abilities. I gained valuable knowledge in RF design, noise prevention, programming interfaces, PCB design best practices, and cross-team communication. I believe this collaborative experience will benefit me in future professional endeavors.

7.1.6 Simon

As a computer science student at NTNU, I have developed a solid understanding of both electronic systems and software, including how to build secure, maintainable, and scalable solutions. I have acquired my theoretical and practical leadership and programming skills through university coursework, reading books, pursuing personal interests, working on side projects, part time jobs, and participating in student organizations such as Orbit NTNU. Many of these experiences involved software development, like my work at Orbit NTNU were especially relevant, as I contributed to low-level software for a satellite. This background proved valuable for our project.

After our initial brainstorming, I was given the responsibility of designing and setting up the software for our satellite. My previous experience provided a strong foundation for understanding how the software architecture should be structured, and enabled me to clearly illustrate and explain the design to the rest of the team. This background also helped me facilitate effective collaboration as we worked toward our software goals. I was able to contribute significantly to key decisions, such as the choice of programming language and operating system because of my familiarity with these technologies.

Throughout the project, I learned a lot, from improving my C programming skills to properly building, compiling and flashing a Zephyr repository from scratch. I gained experience in efficiently debugging embedded systems and troubleshooting hardware issues, often encountering and solving non-trivial problems. The project also challenged me to design software that is both easy to use and adaptable to change. At the start of the project I hoped to deepen my knowledge of Zephyr RTOS, C, and electronic components, and I feel I have achieved this. Additionally, I gained new insights into motors, BLE, and the physics behind reaction wheels, areas in which I previously knew almost nothing. Overall, this project has been a valuable learning experience and I know it will be useful in the future.

References

[1] EiT Group 2. *SpinSat Demo*. Video demonstration made by Group 2 in EiT. Recorded at Make NTNU, Gløshaugen. URL: https://www.youtube.com/shorts/fwadkRE5Dh8 (visited on Apr. 30, 2025).

[2] Aerostar. *Aesrostar G2/G2 HV ESC User Manual*. URL: https://shorturl.at/AECjX (visited on Apr. 22, 2025).

[3] Bilal Akin, Manish Bhardwaj, and Jon Warriner. "Trapezoidal Control of BLDC Motors Using Hall Effect Sensors". In: (2011).

[4] American University, School of International Service. *Small Satellites and International Security*. Sist hentet 28. april 2025. 2024. URL: https://www.american.edu/sis/centers/security-technology/small-satellites-and-international-security.cfm.

[5] Karl Johan Åström and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[6] Elefun. *3s 1000mAh - 30C - Gens Ace Soaring G-Tech XT60*. URL: https://shorturl.at/zzGY9 (visited on Apr. 22, 2025).

[7] Roger Freedman and Hugh Young. *University Physics: Fifteenth Edition*. Pearson, 2018.

[8] Hobbyking. *Aerostar 50A RVS G2 32bit (2 4S) Electronic Speed Controller w/Reverse Function & 4A 5 6V SBEC Image*. URL: https://shorturl.at/JxXRR (visited on Apr. 22, 2025).

[9] F. Landis Markley and John L. Crassidis. *Fundamentals of Spacecraft Attitude Control and Determination*. Springer, 2014.

[10] MCA. *Addressing Noise and Interference in Distributed Antenna Systems (DAS)*. URL: https://callmc.com/noise-interference-distributed-antenna-system-das/.

[11] Mickey McClure. "A Simplified Approach to dc Motor Modeling for Dynamic Stability Analysis". In: (2000).

[12] Ninad Mehendale. "Investigating the Battery Life Issues in Unmanned Aerial Vehicles: An Analysis of Challenges and Proposed Solutions". In: (2021).

[13] Pete Millett. "Brushless vs. Brushed DC Motors:When and Why to Choose One Over the Other". In: (2022).

[14] Seng-Chi Chen & Cih-Huei Shih Ming-Shyan Wang. "Speed control of brushless DC motor by adaptive network-based fuzzy inference". In: (2016).

[15] NASA. *State of the Art: Small Spacecraft Communications*. Sist hentet 28. april 2025. 2024. URL: https://www.nasa.gov/smallsat-institute/sst-soa/soa-communications/.

[16] Roy Nilsen. *TET4120: Electric Drives*. NTNU, 2024.

[17] NTNU. *Eksperter i team - Småsatellitter*. Sist hentet 28. april 2025. 2024. URL: https://www.ntnu.no/eit/ttk4852.

[18] Zachariah Peterson. "The PCB Design Process Methodology and Application". In: (2020).

[19] Propdrive. *PROPDRIVE v2 2830 1000KV Datasheet*. URL: https://shorturl.at/IVr27 (visited on Apr. 22, 2025).

[20] PTC. *Onshape*. https://www.onshape.com/en/. Online CAD. 2025.

[21] Nordic Semiconductor. "nRF52840 - Product Specification v1.11". In: (2024).

[22] Bjarne Stroustrup. *The C++ Programming Language*. 4th. Addison-Wesley, 2013.

[23] John R. Taylor. *Classical Mechanics*. University Science Books, 2004.

[24] MIT Electric Vehicle Team. "A Guide to Understanding Battery Specifications". In: (2008).

A PCB

A.1 PCB Circuit Schematic

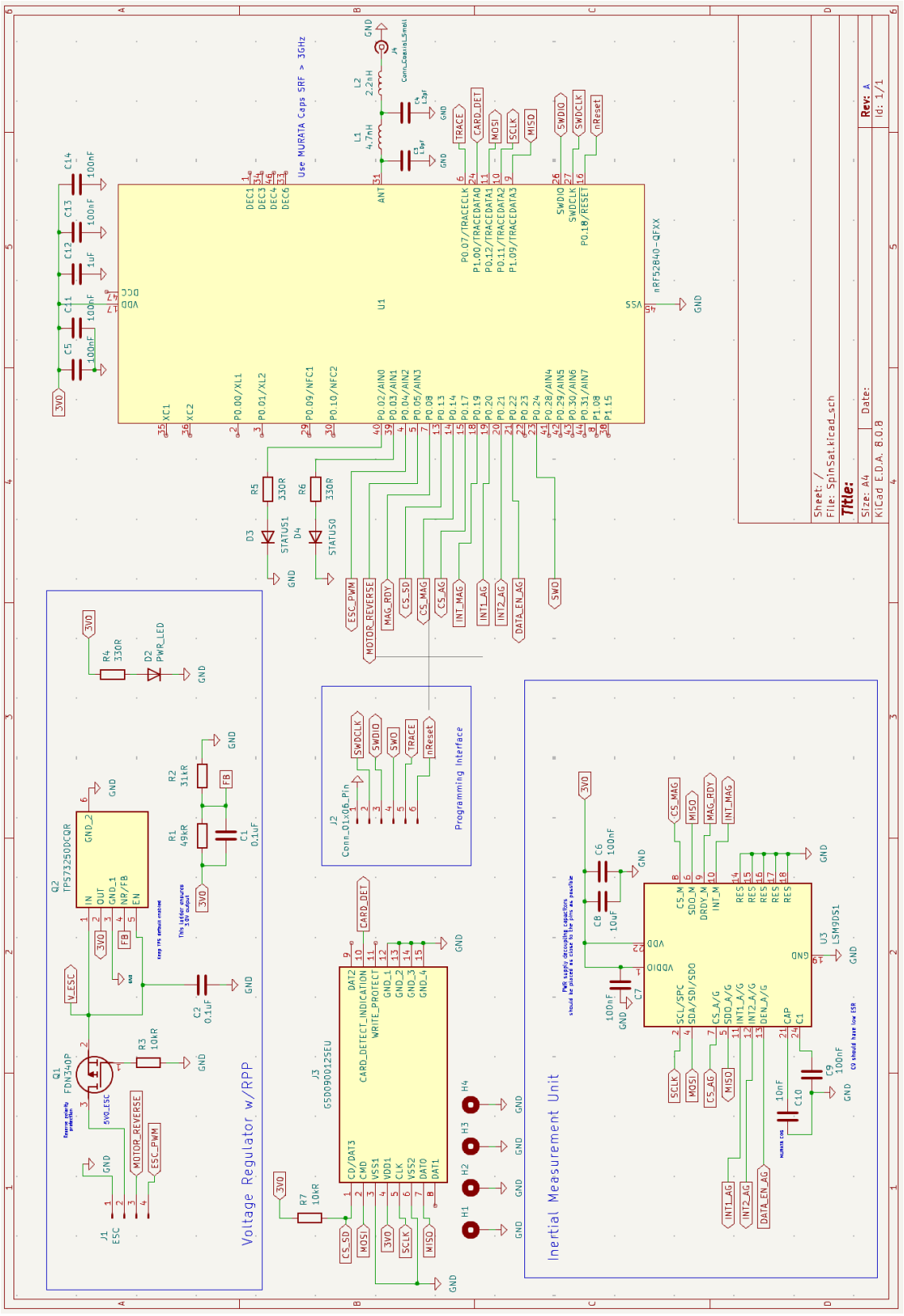


Figure 17: Circuit schematic of the PCB.

A.2 PCB Layout and Routing

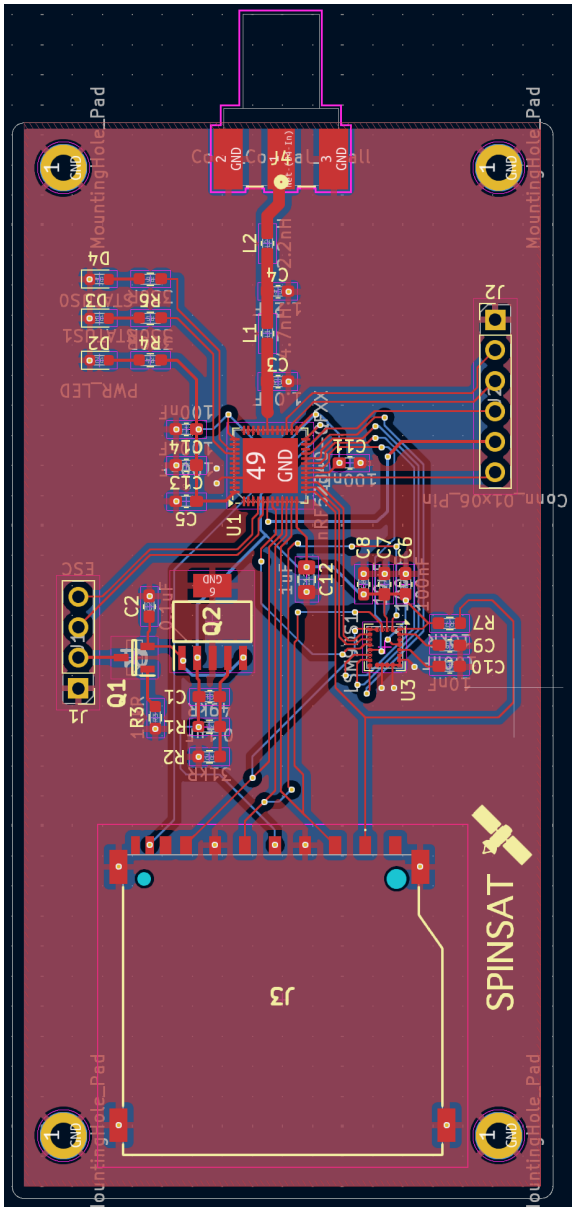


Figure 18: Final layout and routing of the PCB parts.

B Software

B.1 Project Config File

```
# Copyright (c) 2021 Nordic Semiconductor ASA
# SPDX-License-Identifier: Apache-2.0
#
# This file contains selected Kconfig options for the application.

# Logging
CONFIG_LOG=y
CONFIG_APP_LOG_LEVEL_DBG=y

# Custom Libraries
CONFIG_BLUETOOTH=y
CONFIG_STATE_MACHINE=y
CONFIG_IMU=y
CONFIG_MOTOR=y

CONFIG_IMU_GYRO_ODR=119
CONFIG_IMU_ACCEL_ODR=119
CONFIG_IMU_MAG_ODR=10
CONFIG_IMU_CALIBRATION_SAMPLES=100

# Bluetooth LE
CONFIG_BT=y

CONFIG_BT_PERIPHERAL=y
CONFIG_BT_DEVICE_NAME="Spinsat"

# I/O
CONFIG_PWM=y
CONFIG_GPIO=y

# Sensors
CONFIG_SENSOR=y
CONFIG_LSM9DS1=y

# Bus
CONFIG_I2C=y

# Power off
CONFIG_POWEROFF=y

# Stack & Heap
CONFIG_SYSTEM_WORKQUEUE_STACK_SIZE=2048
CONFIG_MAIN_STACK_SIZE=2048
CONFIG_HEAP_MEM_POOL_SIZE=8192

# USB
CONFIG_USB_DEVICE_STACK=y
CONFIG_USB_DEVICE_INITIALIZE_AT_BOOT=y

# New Lib
CONFIG_NEWLIB_LIBC=y
```

Figure 19: Project config file - prj.conf

B.2 Project File Structure

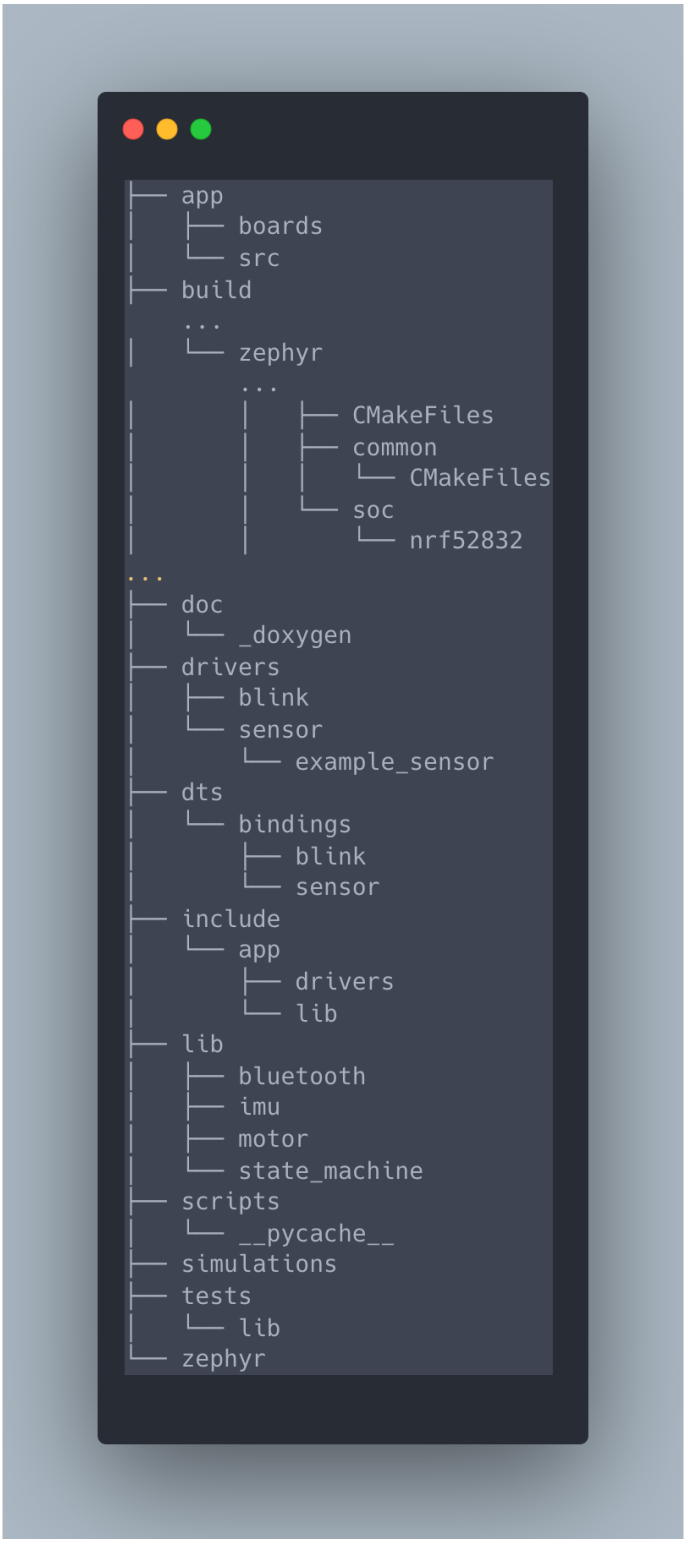


Figure 20: Folder structure of project

B.3 Board Overlay File

```
/ {
    servo: servo {
        compatible = "pwm-servo";
        pwms = <&pwm1 0 PWM_MSEC(20) PWM_POLARITY_NORMAL>;
        min-pulse = <PWM_USEC(1000)>;
        max-pulse = <PWM_USEC(2000)>;
    };
};

&pwm1 {
    status = "okay";
    pinctrl-0 = <&pwm1_custom_motor>;
    pinctrl-1 = <&pwm1_csleep_motor>;
    pinctrl-names = "default", "sleep";
};

&pinctrl {
    pwm1_custom_motor: pwm1_custom_motor {
        group1 {
            psels = <NRF_PSEL(PWM_OUT0, 0, 3)>;
            nordic,invert;
        };
    };

    pwm1_csleep_motor: pwm1_csleep_motor {
        group1 {
            psels = <NRF_PSEL(PWM_OUT0, 0, 3)>;
            low-power-enable;
        };
    };
};

&spi0 {
    status = "okay";
    clock-frequency = <100000>;
    lsm9ds1: lsm9ds1@6b {
        compatible = "st,lsm9ds1";
        reg = <0x6b>;
        label = "LSM9DS1";
    };
};
```

Figure 21: Board overlay configuration file

B.4 Accessing Drivers in Code

```
...

// Example of Servo Motor

#define SERVO_MOTOR_FORWARD DT_NODELABEL(servo)

static const struct pwm_dt_spec servo_f =
PWM_DT_SPEC_GET(SERVO_MOTOR_FORWARD);
// Example of LSM9DS1 Sensor

const struct device *dev = DEVICE_DT_GET_ANY(st_lsm9ds1);
```

Figure 22: Driver access in code

B.5 GATT Bluetooth Definition

```c
#define DEVICE_NAME CONFIG_BT_DEVICE_NAME
#define DEVICE_NAME_LEN (sizeof(DEVICE_NAME) - 1)

#define BT_UUID_SPINSAT_VAL BT_UUID_128_ENCODE(0x00001545, 0x1212, 0xefde, 0x1523, 0x785feabcd123)

#define BT_UUID_SPINSAT_STATE_VAL                                              \
  BT_UUID_128_ENCODE(0x00001546, 0x1212, 0xefde, 0x1523, 0x785afeabcd124)

#define BT_UUID_SPINSAT BT_UUID_DECLARE_128(BT_UUID_SPINSAT_VAL)
#define BT_UUID_SPINSAT_STATE BT_UUID_DECLARE_128(BT_UUID_SPINSAT_STATE_VAL)

/**
 * @brief GATT service definition for the SPINSAT service.
 *
 * This service contains a characteristic for the state machine.
 */
BT_GATT_SERVICE_DEFINE(spinsat_svc, BT_GATT_PRIMARY_SERVICE(BT_UUID_SPINSAT),
                       BT_GATT_CHARACTERISTIC(BT_UUID_SPINSAT_STATE,
                                              BT_GATT_CHRC_WRITE,
                                              BT_GATT_PERM_WRITE, NULL,
                                              write_state, NULL));



static ssize_t write_state(struct bt_conn *conn,
                           const struct bt_gatt_attr *attr, const void *buf,
                           uint16_t len, uint16_t offset, uint8_t flags) {
  LOG_INF("Attribute write, handle: %u, conn: %p", attr->handle, (void *)conn);

  if (!spinsat_callbacks.state_machine_write) {
    LOG_ERR("No callback registered for state machine write");
    return -ENOTSUP;
  }

  // NOTE: NRF52 uses little endian architecture...
  // This means that the first byte is the least significant byte
  uint16_t *val = ((uint16_t *)buf);

  if (len < 2 || len > 2) {
    LOG_ERR("Invalid length: %d, max 2 bytes", len);
    return -EINVAL;
  }

  uint8_t state = (uint8_t)(*val & 0xFF);
  uint8_t data = (uint8_t)((*val >> 8) & 0xFF);

  spinsat_callbacks.state_machine_write(state, data);

  return 0;
}
```

Figure 23: GATT bluetooth definition in code

B.6 Implementation Details

The implementation can be seen on GitHub: https://github.com/sandviklee/spinsat

For specific libraries, look into the ”lib” directory.

B.7 nRF Connect

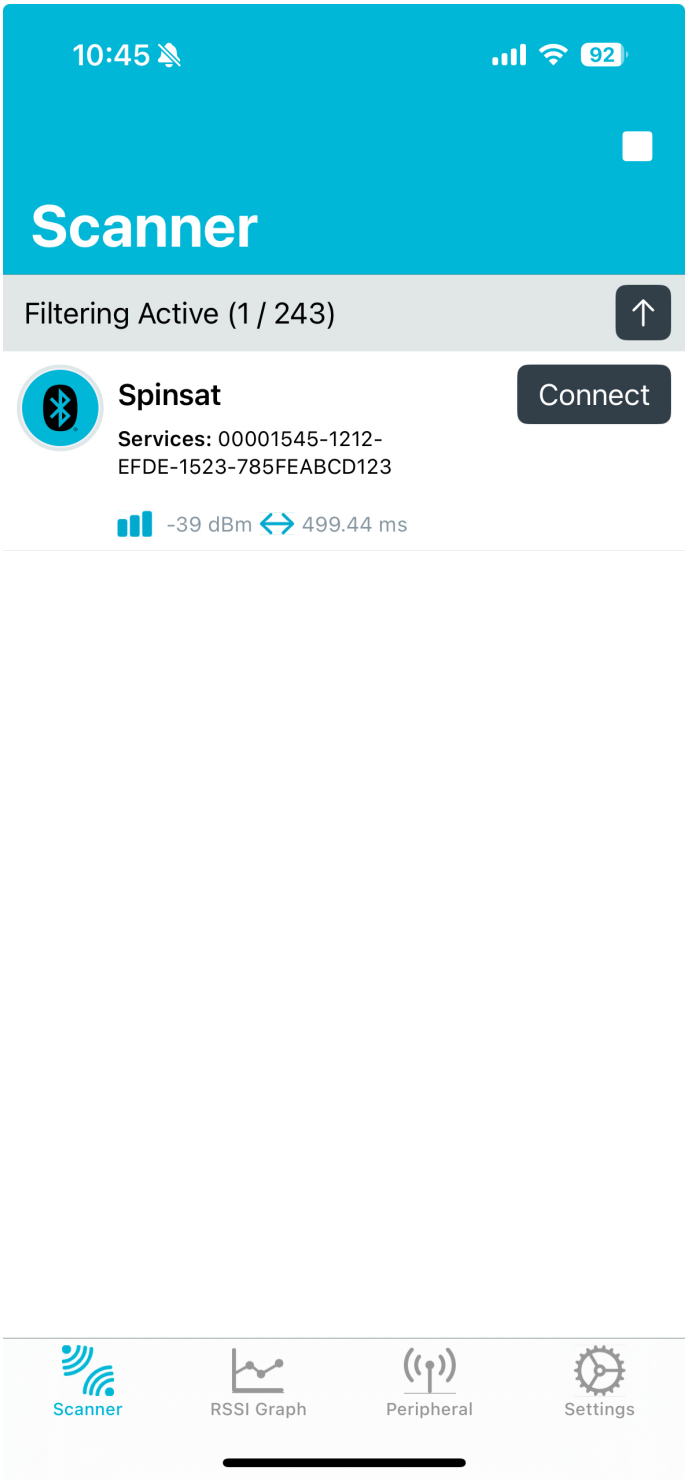


Figure 24: nRF Connect Mobile Application on IOS

B.8 nrf52832 Dev kit

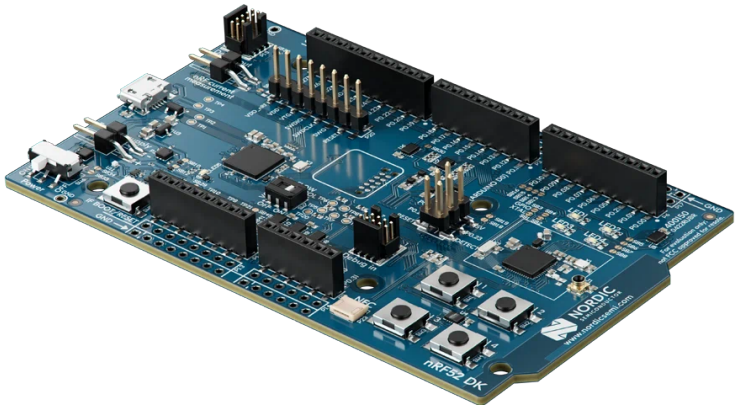


Figure 25: nrf52832 Development kit used

C Satellite Body and Flywheel

C.1 Onshape Schematics

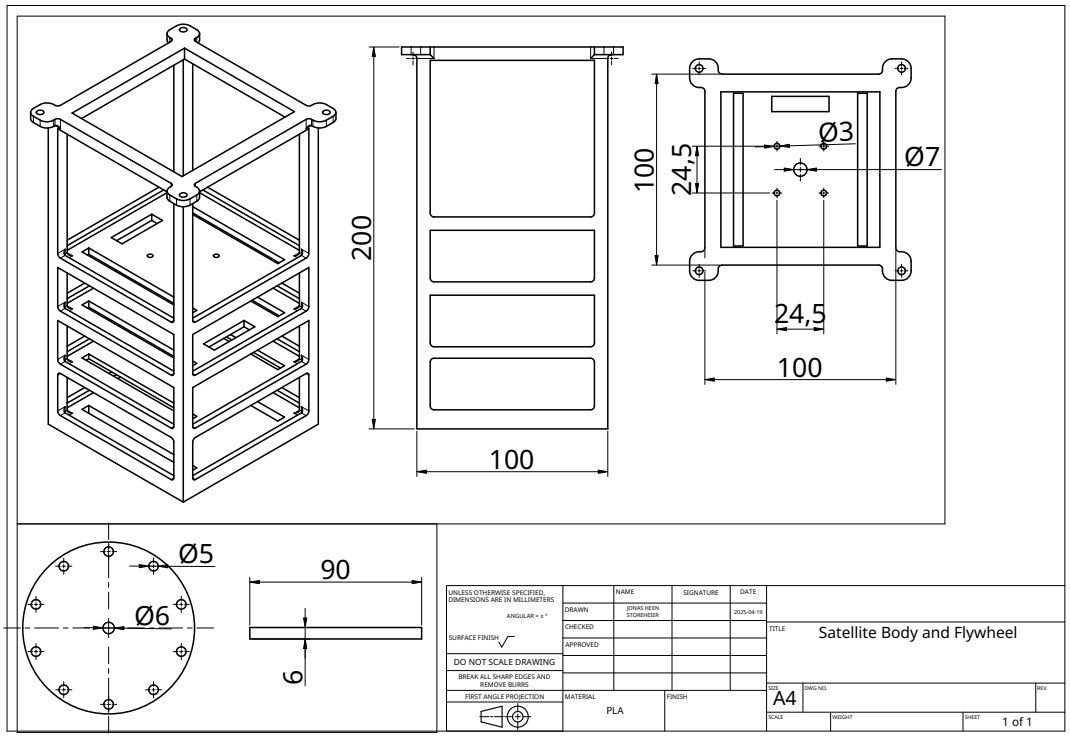


Figure 26: Satellite Body and Flywheel Schematics

C.2 Onshape Calculations

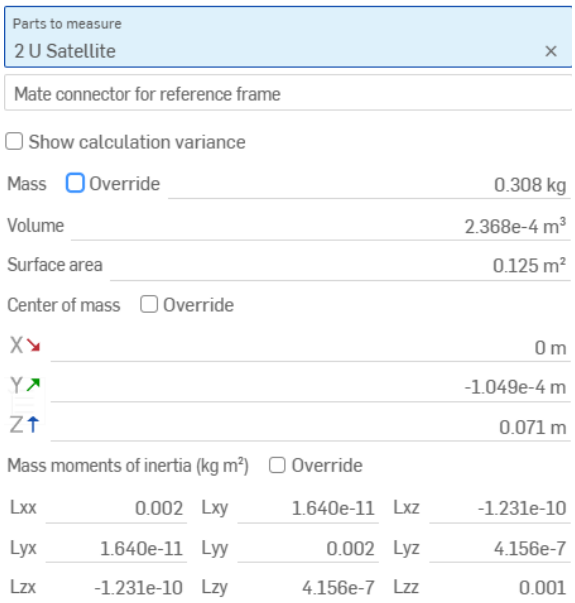


Figure 27: Satellite Body: Moment of Inertia in Onshape

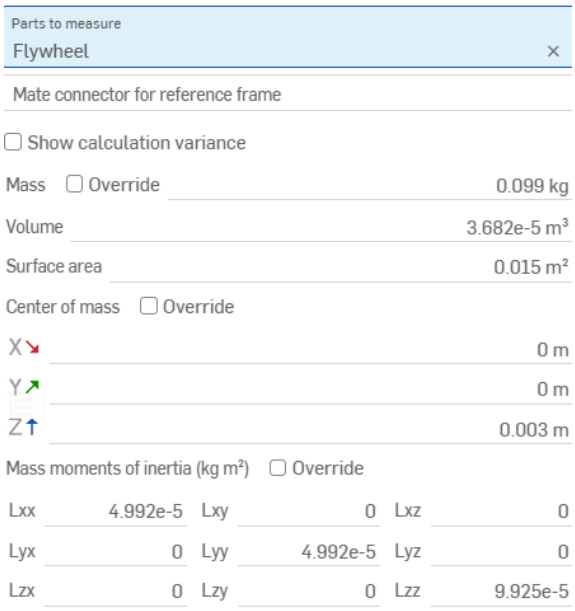
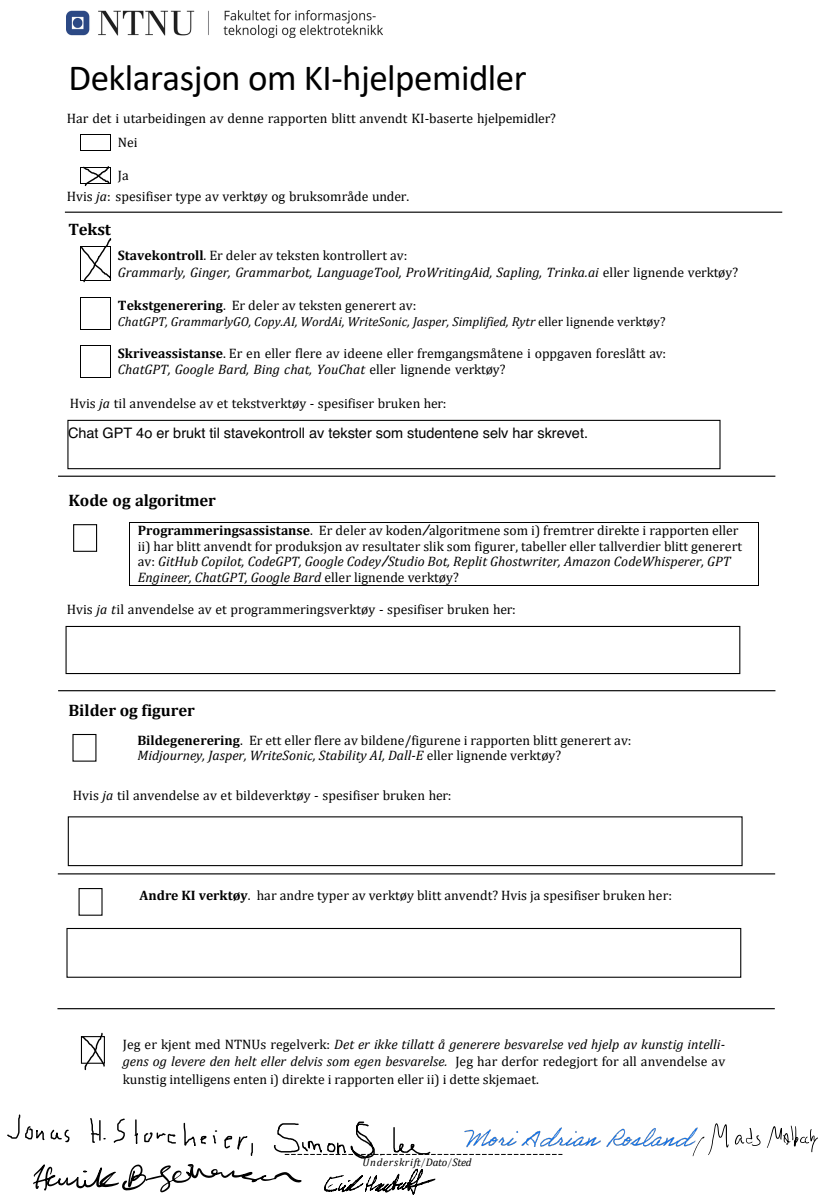


Figure 28: Satellite Flywheel: Moment of Inertia in Onshape

D AI Declaration



NTNU | Fakultet for informasjons-teknologi og elektroteknikk

Deklarasjon om KI-hjelpemidler

Har det i utarbeidingen av denne rapporten blitt anvendt KI-baserte hjelpemidler?

☐ Nei

☒ Ja

Hvis *ja*: spesifiser type av verktøy og bruksområde under.

Tekst

☒ **Stavekontroll**. Er deler av teksten kontrollert av:
Grammarly, Ginger, Grammarbot, LanguageTool, ProWritingAid, Sapling, Trinka.ai eller lignende verktøy?

☐ **Tekstgenerering**. Er deler av teksten generert av:
ChatGPT, GrammarlyGO, Copy.AI, WordAi, WriteSonic, Jasper, Simplified, Rytr eller lignende verktøy?

☐ **Skriveassistanse**. Er en eller flere av ideene eller fremgangsmåtene i oppgaven foreslått av:
ChatGPT, Google Bard, Bing chat, YouChat eller lignende verktøy?

Hvis *ja* til anvendelse av et tekstverktøy - spesifiser bruken her:

Chat GPT 4o er brukt til stavekontroll av tekster som studentene selv har skrevet.

Kode og algoritmer

☐ **Programmeringsassistanse**. Er deler av koden/algoritmene som i) fremtrer direkte i rapporten eller ii) har blitt anvendt for produksjon av resultater slik som figurer, tabeller eller tallverdier blitt generert av: *GitHub Copilot, CodeGPT, Google Codey/Studio Bot, Replit Ghostwriter, Amazon CodeWhisperer, GPT Engineer, ChatGPT, Google Bard* eller lignende verktøy?

Hvis *ja t*il anvendelse av et programmeringsverktøy - spesifiser bruken her:

Bilder og figurer

☐ **Bildegenerering**. Er ett eller flere av bildene/figurene i rapporten blitt generert av:
Midjourney, Jasper, WriteSonic, Stability AI, Dall-E eller lignende verktøy?

Hvis *ja* til anvendelse av et bildeverktøy - spesifiser bruken her:

☐ **Andre KI verktøy**. har andre typer av verktøy blitt anvendt? Hvis ja spesifiser bruken her:

☒ Jeg er kjent med NTNUs regelverk: *Det er ikke tillatt å generere besvarelse ved hjelp av kunstig intelligens og levere den helt eller delvis som egen besvarelse.* Jeg har derfor redegjort for all anvendelse av kunstig intelligens enten i) direkte i rapporten eller ii) i dette skjemaet.

Jonas H. Storcheier, Simon S. Lee, Mori Adrian Rosland, Mads Mølbak, Henrik B. Sebastian, Emil Hasbakk

Underskrift/Dato/Sted

Figure 29: AI Declaration