NTNU – Trondheim
Norwegian University of
Science and Technology

Developing an ADCS Prototype for NTNU Test Satellite

Øyvind Rein

Master of Science in Cybernetics
Submission date: June 2014
Supervisor: Jan Tommy Gravdahl, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Assignment text

The assignment will be about further development of an *Attitude Determination and Control System.* It will be a continuation of the specialization project that was written during the fall of 2013. The long-term goal is to make a functioning ADCS that can be integrated with NUTS (Norwegian University Test Satellite). Previous master students have done a lot of theoretical work with both control and estimation algorithms, which provides a foundation for building a prototype of the ADCS. That way, the theory can be tested in real life. The main topic of this assignment is to build and test an ADCS prototype. Specific tasks:

- Complete and manufacture the ADCS Module, implement drivers and test hardware
- Assemble an ADCS Prototype, which includes:
    - ADCS Module
    - Magnetorquers (Coils)
    - A structure for mounting
    - Placement of photodiodes
    - Battery driven power source
    - Wireless communication
- Create a Test Platform that can be used to test different control and estimation algorithms.
- Use the Test Platform to test the B-dot detumbling algorithm.

Start date: 07.01.2014
Due date: 01.07.2014

Abstract

NUTS is a student satellite program at NTNU with a goal of launching a cube satellite into space. Students contribute through projects, master theses or voluntary work. The satellite will shoot pictures of earth with a camera that will be fixed to the satellite's body.In order to accurately point the camera towards the desired target an Attitude Determination and Control System is required. In this thesis, a prototype of the ADCS has been developed along with a platform for testing software and control algorithms. Further, the b-dot algorithm has successfully been tested using the testbed developed. B-dot is a detumbling algorithm that will be used to stop the satellite's initial spin after it is ejected from its pod.

Sammendrag

NUTS er et studentsatellite prosjekt på NTNU med mål om å skyte en kubesatelitt opp i verdensrommet. Studenter har bidrat gjennom prosjektoppgaver, masteroppgaver eller frivillig arbeid. Satellitten vil bruke et fastmontert kamera til å ta bilde av jorda. Av den grunn er det nødvendig med et system som kan endre retningen til satelliten slik at kamera kan ta blide av et ønsket mål. Et slikt system kalles på fagspråket for *Attitude Determination and Control System*, eller ADCS. I denne oppgaven er det utviklet en prototype av ADCS systemet i lag med en platform for å teste maskinvare, programvare og kontrollalgotitmer. Til slutt ble en nedspinnings algoritme, *B-dot*, testet i dette oppsettet med suksess. Algoritmens funksjon er å stoppe uønsket spinn som kan oppstå etter utløsning fra bæreraketten.

Preface

This Master Thesis was written during the spring of 2014 at the Norwegian University of Technology and Science, Department of Engineering Cybernetics. It has been a part of the student satellite program NUTS (Norwegian University Test Satellite). This thesis concludes my Master's degree in Engineering Cybernetics, Robot Control.

I would like to thank my supervisor Jan Tommy Gravdahl for all the guidance and follow-up throughout the semester. I would also like to thank the whole NUTS team for all the help and support I have gotten, and I hope my contribution makes up for it. I wish everyone who have been, or will be, involved with the project good luck. I am certain all the hard work will be awarded with a successful launch. Finally, I would like to thank my parents for the all the support, and for always keeping their doors open, giving me place where I could escape the studies and recharge my batteries.

List of Contributions

This thesis and semester project

- Design ADCS Module in Altium. Selecting components etc.
- Implemented and tested drivers for the ADCS Module
- Design and assemble an ADCS Prototype and Test Platform
- Implemented and tested B-dot detumbling algorithm.

Previous Theses

- Kalman filter for attitude determination of student satellite [25]
- Development, Implementation and Testing of Two Attitude Estimation Methods for Cube Satellites [16]
- Design and Implementation of Attitude Control for 3-axes Magnetic Coil Stabilization of a Spacecraft [26]
- Optimal attitude control of a double cubesat using magnetorquers [14]
- Satellite Attitude Control System [1]
- Development and Comparison of Estimation Methods for Attitude Determination [23]
- Design of Attitude Control System of a Double CubeSat [5]

Papers

- Development on the EQUEST method for attitude determination [24]
- Implementation of three axis attitude determination and control system for a double cubesat [13]
- A comparison of attitude determination methods: theory and experiments [17]
- Attitude Determination for the Student Satellite nCubeII: Kalman Filter [11]
- Attitude control for the Norwegian student satellite nCube [21]
- Three axis Attitude Determination and Control System for a picosatellite: Design and implementation [10]
- IAA-CU-13-11-06 Using independent combinations of CubeSat solar panels as sun sensors for separating inbound sunlight information [20]
- Magnetic attitude control for satellites [9]
- Explicit model predictive control of a satellite with magnetic torquers [18]

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 NUTS

1.1.1 Overview

NTNU Tests Satellite (NUTS) is a student project with a goal of designing, building and launching a double cube satellite (figure 1.1). The project was initiated in 2010 and is part of the Norwegian student satellite program (ANSAT) that is driven by the Norwegian Center for Space-related Education (NAROM). NUTS is one of three satellites in this program. The University of Oslo is developing their cube satellite CubeSTAR, while Høyskolen i Narvik had an unsuccessful launch of their satellite, HiNCube, during fall 2013. NUTS will have a camera as the primary payload. It was originally intended to be an infrared camera, which turned out to be too expensive. The current design uses a visible light CMOS camera with a much lower price tag. The following list show all the mission goals [4].

- Deliver a tested satellite according to mission specifications
- Transmit a beacon signal receivable for radio amateurs
- Confirm successful de-tumbling
- Establish two-way communication and receive full telemetry
- Test CMOS camera
- Initiate CMOS camera pointing
- Initiate CMOS camera sequence

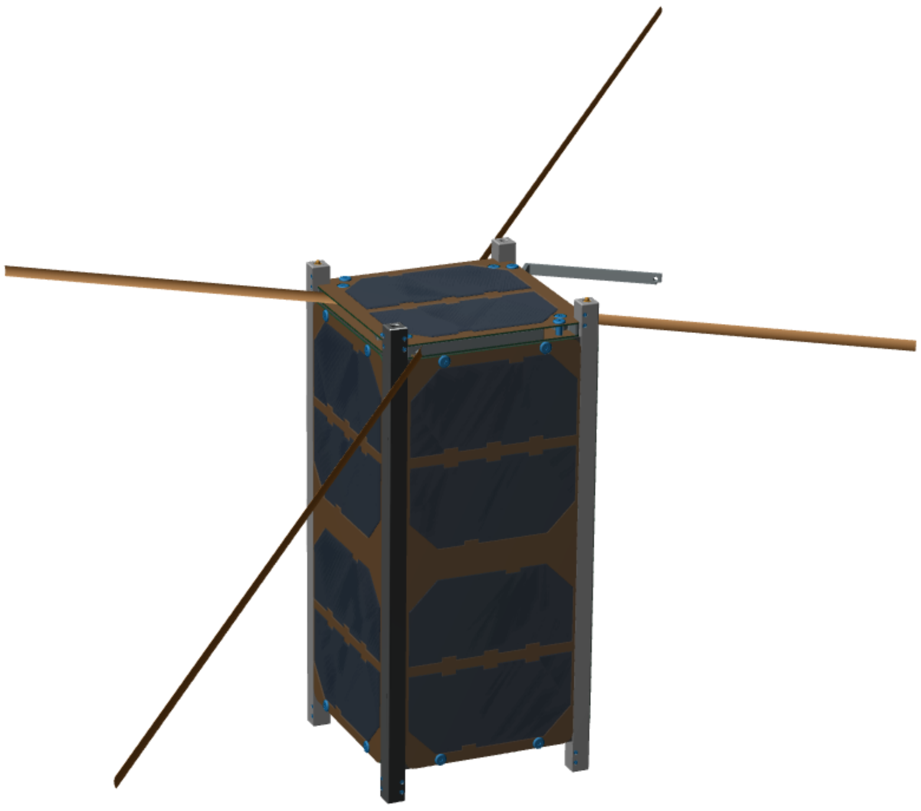


Figure 1.1: CAD model of NUTS, December 2013

- Receive a valid series of images

1.1.2 Mechanical Design

The supporting structure is made of carbon fiber composite material. Aluminum is more commonly used for cube satellites but it weighs more. It is beneficial to keep the weight down since satellite launches are paid by the kilos. The cube satellite standard gives a restriction to the maximum weight. Hence, a lighter mechanical structure gives room for more scientific payload. An inner frame in plastic will support all the electronics with tracks so that circuit boards can be slid in place.

1.1.3 Electrical System

Solar panels are used to generate power that is stored in a set of batteries. From there the energy is available to all the sub system. One of the features of NUTS is the backplane, which provide a communication bus, and distributes power to the subsystems. This is different from other cube satellites that stack the circuit boards on top of each other. The benefit of this approach is that the modules can be connected

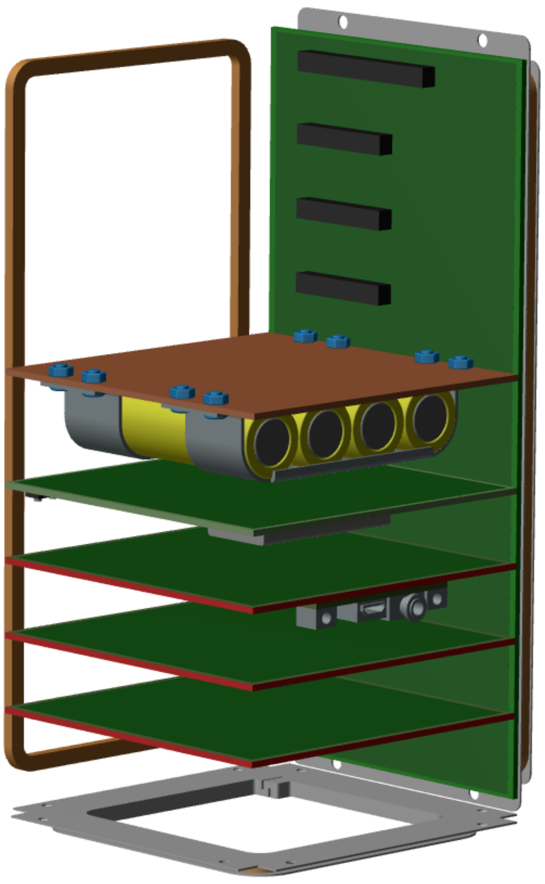


Figure 1.2: The inside of NUTS

and disconnected individually without interfering with other modules. Figure 1.2 is an illustration showing the Backplane and some modules. The following list names all the modules.

- On-board Computer (OBC)
- OWL Radio
- AFD7021 Radio
- Attitude Determination and Control System (ADCS)
- CMOS camera module
- Electrical Power Supply (EPS)
- Battery Module

1.2 Attitude Determination and Control System

Most satellites are required to point instruments towards some target. It can be either a camera, a directional antenna or a scientific instrument. To perform such a maneuver it is necessary for the satellite to know how it is oriented relative to its desired target. In addition, a mechanism for creating torque is needed. The complete system is referred to as an Attitude Determination and Control System, or ADCS for short. Attitude Determination is accomplished using several sensors and a filter to estimate the heading. The available sensors on NUTS are a three-axis magnetometer, a three-axis angular rate gyroscope, and photodiodes for triangulation of the sunlight(referred to as the Sun Sensor in this thesis). To create torque NUTS uses magnetorquers. They are essentially just coils that interact with Earth's magnetic field. The three coils are mounted perpendicularly so that a magnetic moment can be created in any direction. Different control algorithms can be used depending on the wanted behavior. Four useful control strategies where thought of after a discussion with the team.

Detumbling Reduces unwanted spin the satellite and can be very useful right after the satellite has been deployed.

Spin-stabilized pointing The camera is fixed in a celestial direction with a spin about its axis. The spin will help stabilize when one or more references measurements are unavailable. For instance when the sun is behind the earth and the photodiodes receives no sunlight.

Earth fixed pointing The camera point toward Earth's center. Useful for Earth observation.

Target fixed pointing The camera point toward a fixed location on Earth e.g. Trondheim. Can be used when shooting image time series.

1.3 ADCS Prototype and Test Platform

This is the main topic in this thesis. Previous master students have thoroughly investigated several estimation and control algorithms for NUTS during the past few years. EQUEST, which is an attitude estimation algorithm, have been implemented on hardware and tested [16]. For simplicity, an accelerometer was used instead of a sun sensor. The tests did show good results, but it is still uncertain how the algorithm will behave with actual sun sensor data. As for the control algorithms, all the testing have been conducted with Simulink in MATLAB. Simulation has its limitations, despite good results. It is difficult to foresee what will happen in real life. Sample discretization, inaccurate timing, low processing power, non-Gaussian noise and many other factors may impair the performance. For these reasons, the next logical step was to make a prototype of the ADCS as well as a test platform. This also gives the opportunity to test if the implementations themselves are correct before the satellite is launched. The prototype presented in this thesis has been developed such that it can be considered a revision of the final ADCS design. It is beneficial that the software that has been tested can be used for the actual satellite. The process of making a prototype was initiated this fall as part of a semester project [22].

1.4 Previous Work

A lot of work has been done on the ADCS since the NUTS project was initiated in 2010. Most have been theoretical work, but experiments with hardware have been done as well. This section will only mention the work that is most relevant to this thesis and future work. A complete list of contributions can be found in the beginning of this thesis.

Three estimation methods have been investigated; Extended QUaternion ESTimator (EQUEST), Extended Kalman Filter (EKF) and Nonlinear Observer [16] [23]. They all have their pros and cons, but in [22] EQUEST it is argued to be favorable. In [16], both the EQUEST and the EKF was implemented and tested on hardware. Previous master students have investigated different control strategies. [1], [5], [26] and [14]. The B-dot algorithm will almost certainly be used for detumbling, while simulations of the Non-linear control looks promising for attitude reference control [5]. In [26] it was investigated if a simple H-bridge circuit could drive a magnetorquer. In [22], the first revision of the ADCS Module was designed.

Chapter 2

Background Material

The following sections are quoted from [22]: 2.1, 2.2, 2.4, 2.5

2.1 ADCS Module

The ADCS module is the circuit board that is to be slid into the Backplane. The module will be connected to one of the slave slots. A system outline is shown in figure 2.1, and the components are listed here:

- Microcontroller (ATxmega256A3U)
- Gyroscope (L3G4200D)
- Compass Sensor (HMC5983)
- Sun Sensor (6 analog amplification circuits)
- Connectors:

  Backplane Slave Connector

  JTAG

  Bluetooth

  Coil Connector

  Sun Sensor Connector

The OBC (On-Board Computer) will provide services for the ADCS-module as starting/stopping, (re)programming, and clock synchronization. Reprogramming is an important feature if a bit flip should happen in the microcontroller's program memory.

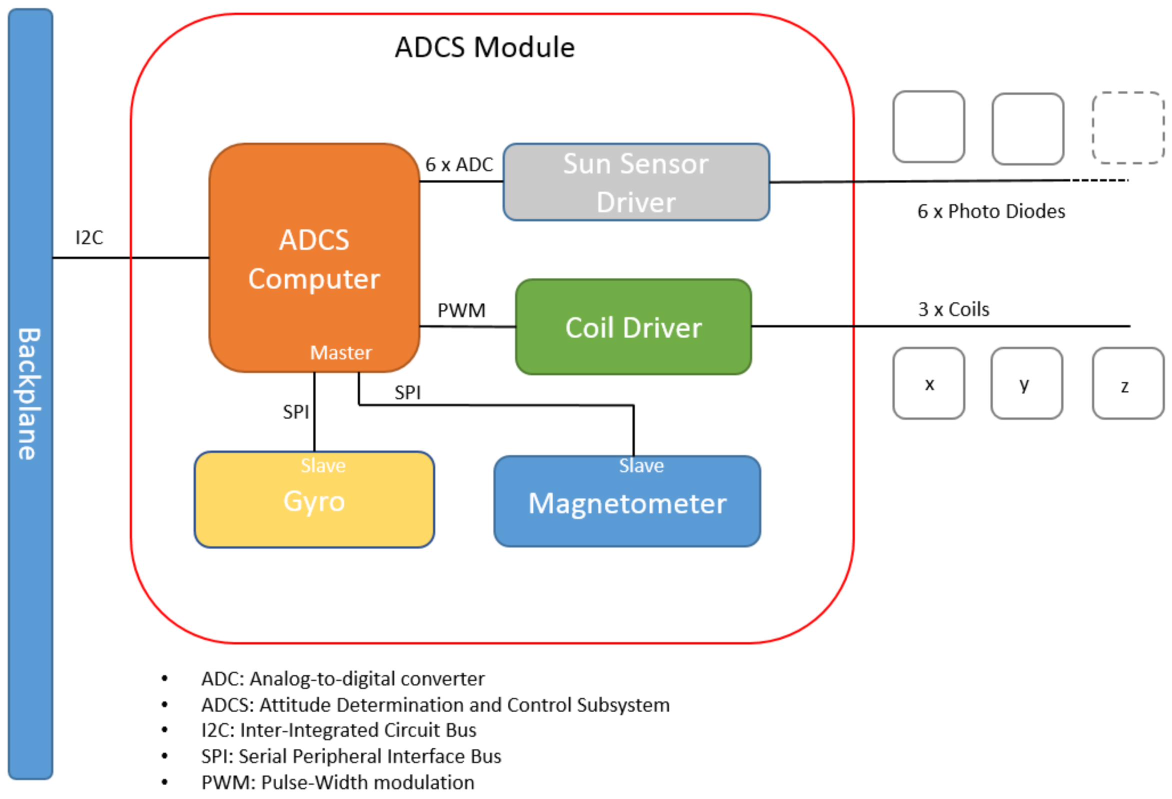

ADCS Module
Backplane
I2C
ADCS Computer
Master
6 x ADC
Sun Sensor Driver
6 x Photo Diodes
PWM
Coil Driver
3 x Coils
x
y
z
SPI
SPI
Slave
Gyro
Slave
Magnetometer
ADC: Analog-to-digital converter
ADCS: Attitude Determination and Control Subsystem
I2C: Inter-Integrated Circuit Bus
SPI: Serial Peripheral Interface Bus
PWM: Pulse-Width modulation

Figure 2.1: ADCS hardware modules

Previous revisions of the EPS did provide both 3.3 volts and 5 volts, but now it only has 3.3 volts available. This had to be taken into account when components were selected.

2.1.1 Module Layout

The module itself measures 89.5 mm $\times$90.0 mm. The sensitive sensor components were placed on the left side of the board while the Bluetooth Header and the Coil Driver were placed to the right. This was done in order to minimize potential disturbances caused by electromagnetic interferences. The routing was done manually for the same reasons. Figure 2.2 shows a 2D view of the module, while figure 2.3 shows a 3D model.

2.2 B-dot Controller

After being deployed, the satellite will have an initial spin. The first control phase is to stop this rotation and is called detumbling. The B-dot algorithm does exactly this, and simulations shows promising results. The name B-dot is referring to the time derivative of the magnetic field vector, $\dot{\boldsymbol{B}}$. The algorithm simply tries to slow down the initial rotation using $\dot{\boldsymbol{B}}$ as feedback to the coil output. See equation 2.1. The great

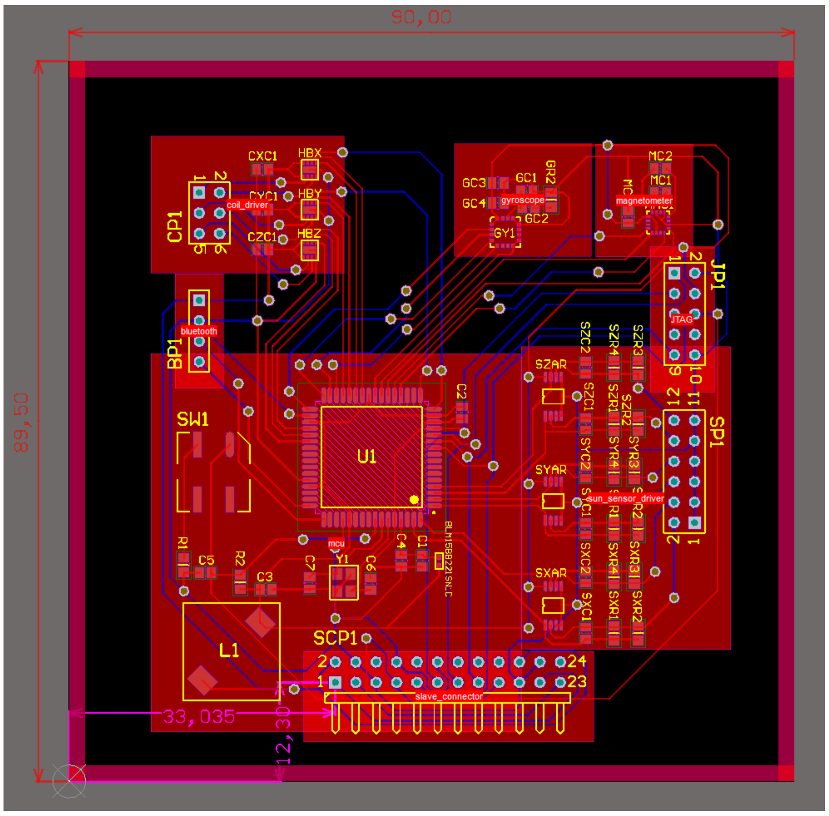


Figure 2.2: 2D view of the layout

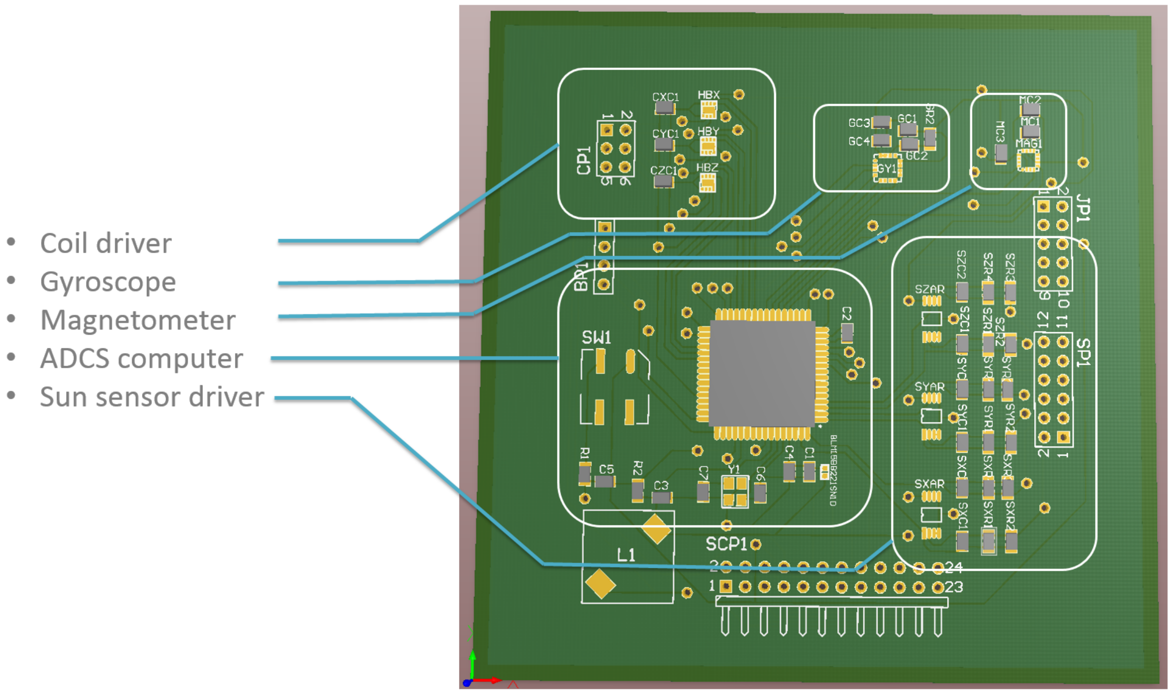


Figure 2.3: 3D view of the layout. Showing the different components

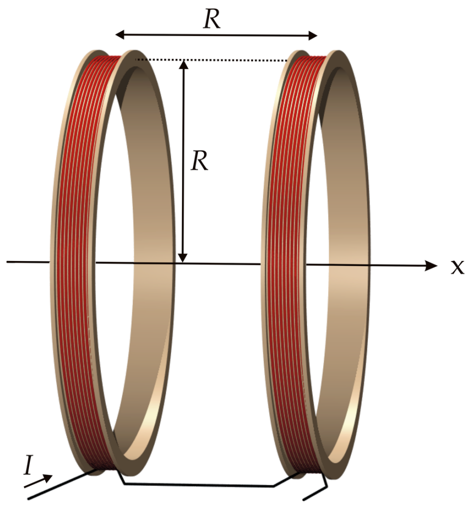


Figure 2.4: Illustration of a Helmholtz Coil [12]

thing about this method is that it only rely on data from the magnetometer. Given sufficient amount of time, it can even detumble with only one working coil.

$$\boldsymbol{m}^b = -k\dot{\boldsymbol{B}}^b \quad , \qquad k > 0 \tag{2.1}$$

where

k is the control gain

$\boldsymbol{m}^b$ is the magnetic control output moment

$\dot{\boldsymbol{B}}^b$ is the time derivative of the magnetic field vector.

2.3 Helmholtz Coil

A Helmholtz Coil can be used to create an approximately uniform magnetic field and consists of two parallel coils. The radius of the two coils is equal to the distance between them, as illustrated in figure 2.4.

2.4 Sun vector model

This is a short description of the Sun vector model developed in [19]. The idea is that the output from two solar panels can be used to calculate where the inbound light is coming from. The sunlight will have different angle of incidence, α, for each solar panel, and can be calculated as a function of the output voltage. Each angle

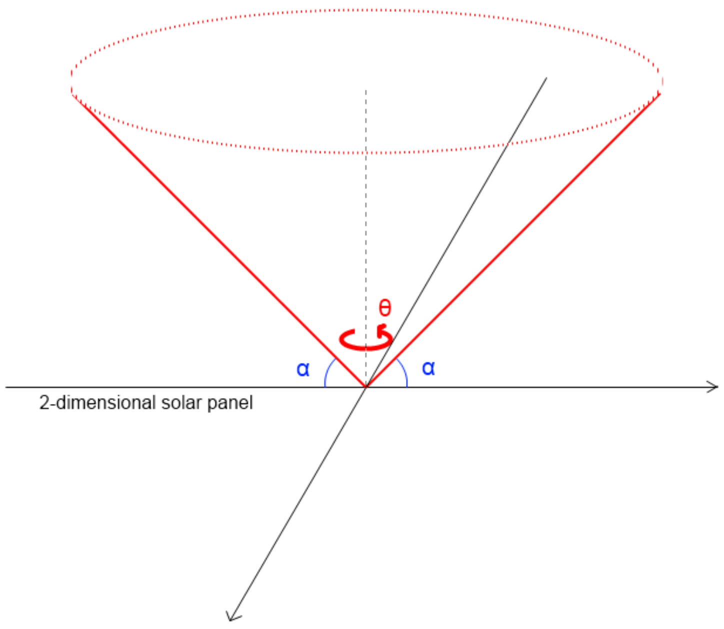


Figure 2.5: 3D vectors inbound on 2D solar panel

can be represented as a cone with centerline perpendicular to the solar panel's surface. A sun vector can be calculated as the intersection between the two cones. The sun vector model that has been developed does not calculate a vector, but an angle θ. See equation 2.2 and figure 2.5.

$$\Theta_x = \sin^{-1}\left(\pm\sqrt{\frac{\tan^2(\alpha_x)+1}{\tan^2(\alpha_x)(\tan^2(\alpha_y)+1)}}\right) \tag{2.2}$$

When θ_x and θ_y are know, it is straightforward to find the corresponding sun vector. It is worth mentioning that sunlight reflected from Earth will influence the measurements. This reflection is called the *albedo effect* and must be compensated for to get accurate results.

2.5 I/O

An embedded systems must have a way to interact with the environment. It would be useless otherwise. The ADCS module has to be able to collect information from sensors, to control the attitude and to communicate with the On Board Computer. All of this is done through input and output signals.

Analog-to-Digital Converter (ADC) ADC's are common in embedded systems. Whenever a physical value has to be digitalized, an ADC is used. The most important properties of an ADC are sampling rate and bit resolution. Higher resolution gives a better approximation of the analog signal. It is good practice, if possible, to apply a gain to the analog signal such that the entire range of the ADC is utilized. This does of course require some knowledge about the values that are going to be measured. The data is useless if the ADC is saturated and may even harm the hardware.

Pulse-width Modulation (PWM) PWM is a technique often used to control the speed of dc-motors, but it can be used for other applications as well. It basically is a switch that is turned on and off at high speeds, with adjustable intervals, such that the average output can be controlled. The PWM *frequency* is how often a pulse is triggered, and the *duty-cycle* is how long the pulse is. The duty-cycle is given in percentage of the frequency (0 % - 100 %). This is useful method when a relatively slow system is to be controlled. Microcontrollers often have many output pins that can generate PWM signals.

Serial Peripheral Interface Bus (SPI) This is a four wire digital interface that is used for communication between modules in hardware. The four wires are called SCLK, MOSI, MISO and SS, and their functions are serial clock, master output, master input, and slave select, respectively. SPI is a very defined standard and is easy to setup.

Chapter 3

ADCS Module

This chapter will focus on testing the hardware on the ADCS Module. More information about the actual design can be found in [22].

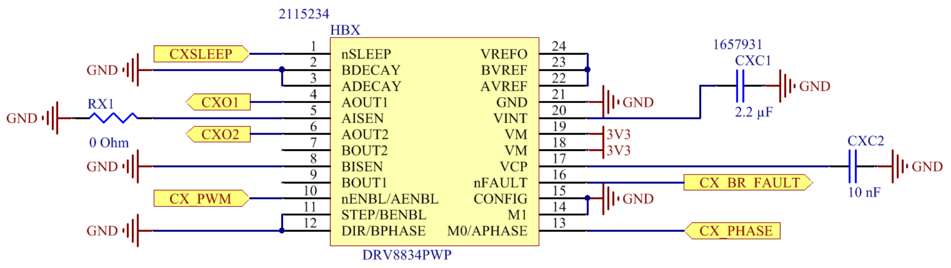
3.1 Modifications

The previous design had three H-bridges that turned out to be difficult to solder because of their small size. This lead to a new revision of the PCB with another set of H-bridges (DRV8834PWP) that was easier to solder. New schematics and layout can be seen in figure 3.1. In addition, they have a fault status pin that might come handy. Instead of making the new modified circuit board in house, ten pieces were ordered from China.

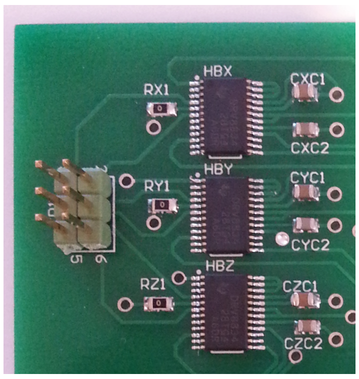
3.2 Sun sensor and Gain Resistor

As mentioned in section 2.1, the sun sensor consists of six analog op-amp circuits that amplify the signal coming from the photodiodes. The amplification gain is selected through a resistor. Figure 3.2 show a simplified schematic of one op-amp circuit where R_{gain} is the gain resistor, $i_d(\Phi_v)$ is the current generated by the photodiode as a function of the luminous flux, and V_{out} is the output voltage that will be measured by an Analog-to-Digital Converter. C_f is just for stabilizing which is good practice when dealing with op-amps. By following the golden rules of an ideal op-amp a relation between input ($i_d(\Phi_v)$) and output (V_{out}) was found:

$$V_{out} = R_{gain} \cdot i_d(\Phi_v) \tag{3.1}$$



(a) H-bridges Circuit Schematics



(b) H-bridges layout (DRV8834PWP)

Figure 3.1

It was decided to find a suitable R_{gain} experimentally because $i_d(\Phi_v)$ is unknown.

The circuit in figure 3.2 was realized using a breadboard, a general purpose op-amp, and several different valued resistors to choose from. An Xplained A1 Evaluation Kit was used to read the analog V_out. It has the same microcontroller as the ADCS Module. The test setup can be seen in figure 3.3b. The experiment was conducted on a clear sunny day (figure 3.3b). The ADC readings were monitored while moving the photodiode around, trying to find the peak output. It occurs when the photodiode is pointed straight towards the Sun. The ADC has a resolution of 12 bit, which means that it has a digital range from 0 to 4095. This corresponds to a voltage range from 0 to 1 volt. The output voltage can in other words not overshoot 1 volt at maximum. When selecting a resistor it had to be taken into consideration that the solar radiation is much lower on Earth's surface than in space because of the atmosphere. Especially during the winter at high latitudes. This experiment was conducted 20th February, 63° deg latitude (Trondheim), at 12:40. At that time the Sun's radiation was estimated to be 0.55 kW/m^2 [15]. In comparison the radiation in orbit is 1.343 kW/m^2. With does assumptions in mind a resistor of 200 Ω seemed to qualify. During the experiment it achieved a maximum ADC output of 1407 (0.34 volt), see figure 3.3a. In orbit the

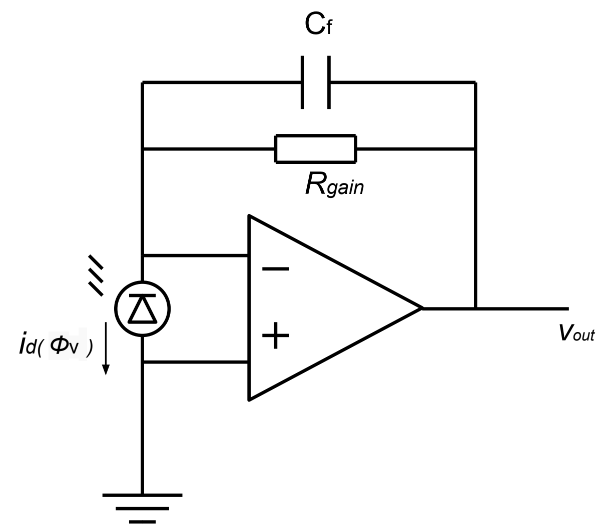


Figure 3.2: Photodiode amplification circuit

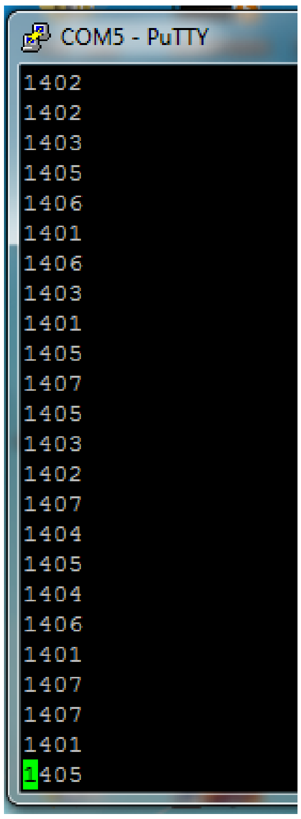
radiation is estimated to be 2.46 times stronger, thus the ADC output would be about 3450 (0.85 volt), assuming there is a linear relation between radiation strength and output.

Before NUTS is launched, it must be known that the Analog-to-Digital Converter will not saturate when the photodiodes are exposed to the Sun. The conclusion in this little experiment has too many uncertainties and cannot be taken for granted, but it may serve as guideline for future testing.

3.3 Hardware testing

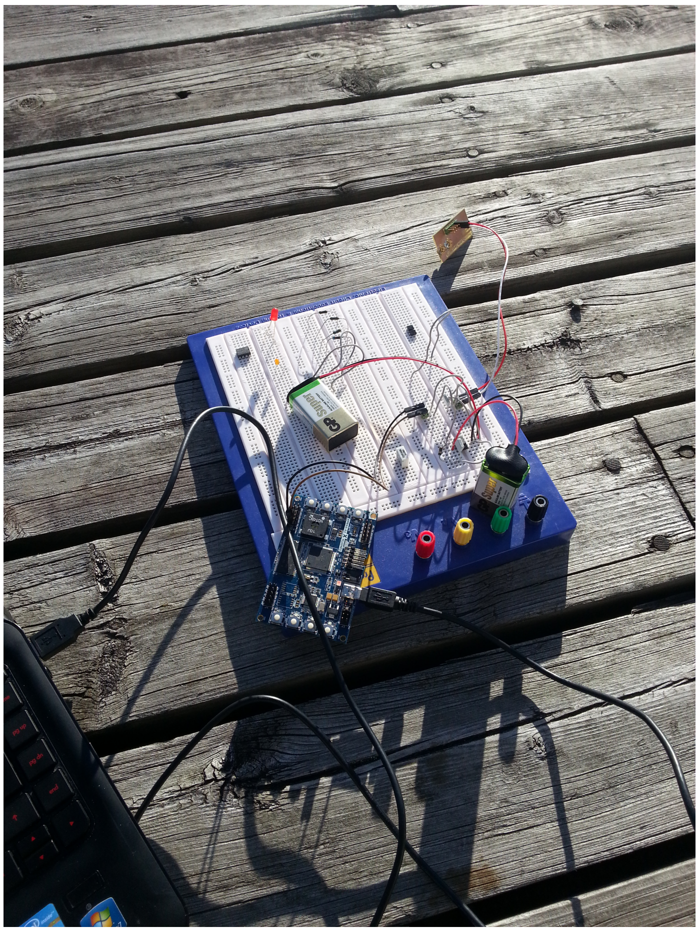
After the soldering was completed, the hardware had to be tested. At that time, there was many uncertainties because it was the first time the design had been realized. The easiest way to test the different components was with software. The drivers from chapter 5 was used for this purpose. The initial impression was good. A DC power source was connected to the backplane connector on pin 2 (3V3) and pin 4 (GND). Then an AVR Dragon development tool was connected to the JTAG connector on the ADCS module. The ATxmega256A3U microcontroller was immediately recognized and ready for programming.

The magnetorquer output connector was tested with a multimeter. It was supposed to behave according to the truth table 5.1 in chapter 5, which it did. Then the magnetorquers were connected, and the magnetic field was successfully detected by the compass sensor of a smartphone.



```
1402
1402
1403
1405
1406
1401
1406
1403
1401
1405
1407
1405
1403
1402
1407
1404
1405
1404
1406
1401
1407
1407
1401
1405
```

(a) ADC measurement, showing peak output. $R_{gain} = 200\Omega$



(b) Bread board, Op-amp circuit, Photodiode, and XMEGA-A1 Xplained

Figure 3.3: Finding a correct gain for the Sun Senor circuit while using the Sun as reference

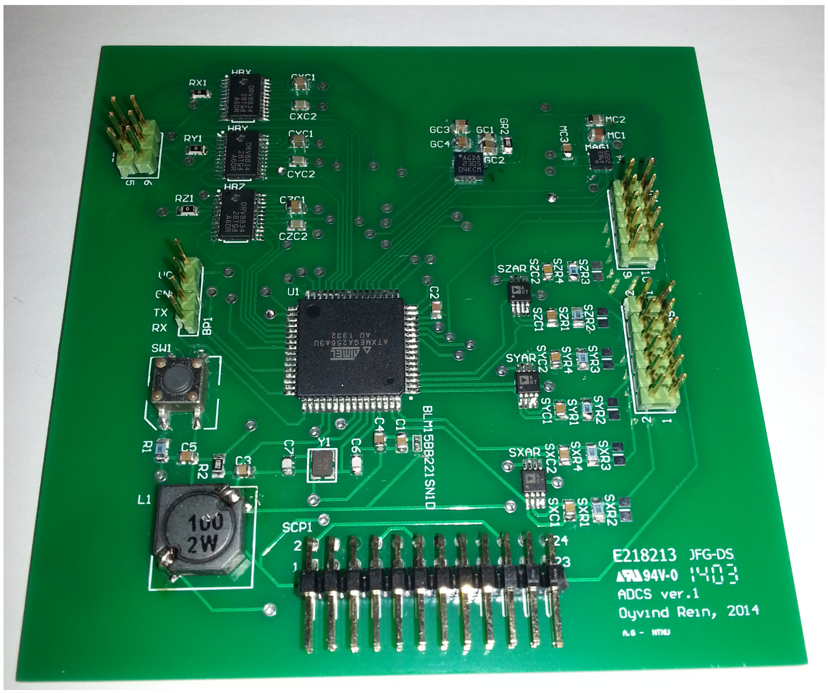
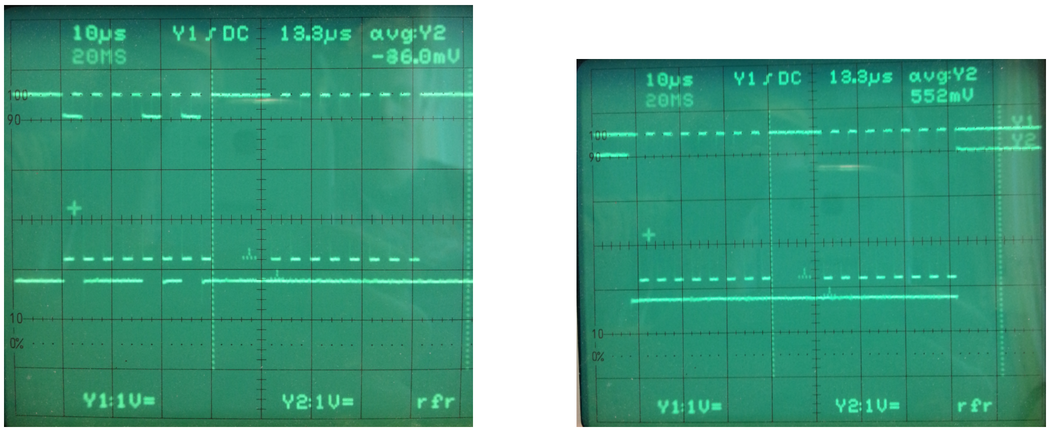


Figure 3.4: Finished ADCS Module



(a) Clock and MOIS. Sending 0x8A (b) Clock and MISO. No response

Figure 3.5: Investigating the SPI interface between the microcontroller and the magnetometer. It seems that the correct message is sent, but the magnetometer does not respond. The correct response to 0x8A (Identification Register A) is 0x48, or 'H' in ASCII

The sensors were tested simply by sending measurement data over the serial interface. The six photodiodes were connected and the module was moved around. Both the gyroscope and the sun sensor responded to the movement, but the magnetometer kept quiet. A lot of time was invested to troubleshoot the issue. The initial thought was that it was a problem with the soldering because the gyroscope and the magnetometer have an almost identical driver since they both use an SPI interface. After a closer investigation it turned out to be a design fault in the layout. Solder mask was missing around the component's pads which makes it difficult to solder. The solder mask is suppose to prevent solder from floating between the pads during soldering. Because of this, it was decided to solder the magnetometer once more. It did not work that time either, which led the suspicion went back to the software.

The next step in the investigation was to measure the digital signals from the microcontroller, and the possible response from the magnetometer, using an oscilloscope. The measurements can be seen in figure 3.5. In the attempt, it was sent a query to Identification Register A, but the magnetometer did not respond. The expected response was the ASCII symbol 'H'. Thus, the software was working after all.

Another possible reason for the lifeless chip could be wrong schematics and layout. However, after looking several times through the datasheet, no deviations from the suggested design could be found. Then it was decided that a simple evaluation circuit should be made in order to confirm that both the software and the layout was indeed correct. The circuit board in figure 3.6 was made so that it could be inserted into a

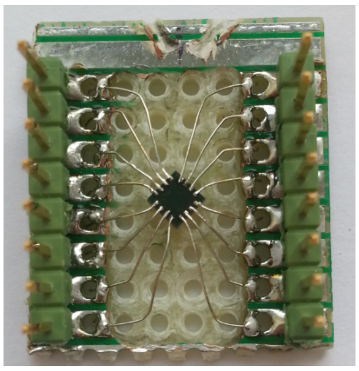


Figure 3.6: Evaluation board of the HMC5983 magnetometer

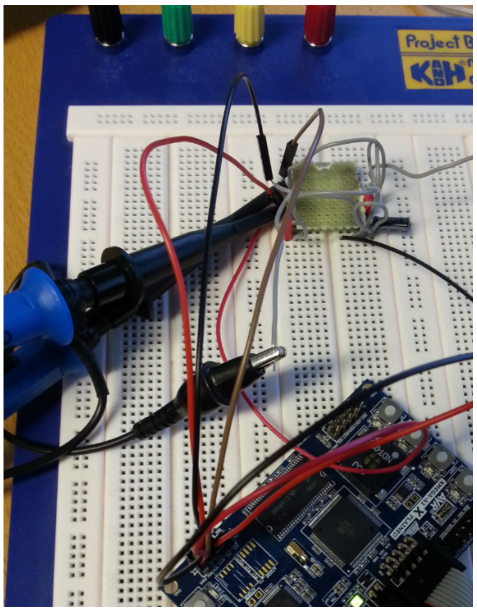


Figure 3.7: Breadboard for evaluation of the HMC5983 magnetometer

breadboard (see figure 3.7). An XMEGA-A1 Xplained evaluation board was used to run the software. It has the same microcontroller as the ADCS Module. Finally, the magnetometer responded, and the oscilloscope confirmed as well. The response can be seen in figure 3.8. Again, this led to the conclusion that the soldering on the ADCS Module was bad, and after soldering the magnetometer a third time it worked like a charm.

With the magnetometer working properly, the ADCS Module was fully functional. Figure 3.9 demonstrates that how the sensors react to random movement in sunlight. The data was obtained after the ADCS Prototype (Chapter 4) was put together.

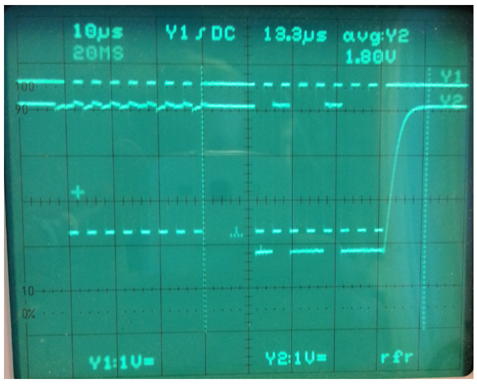


Figure 3.8: Correct SPI response from the magnetometer. 0x48, or 'H' in ASCII

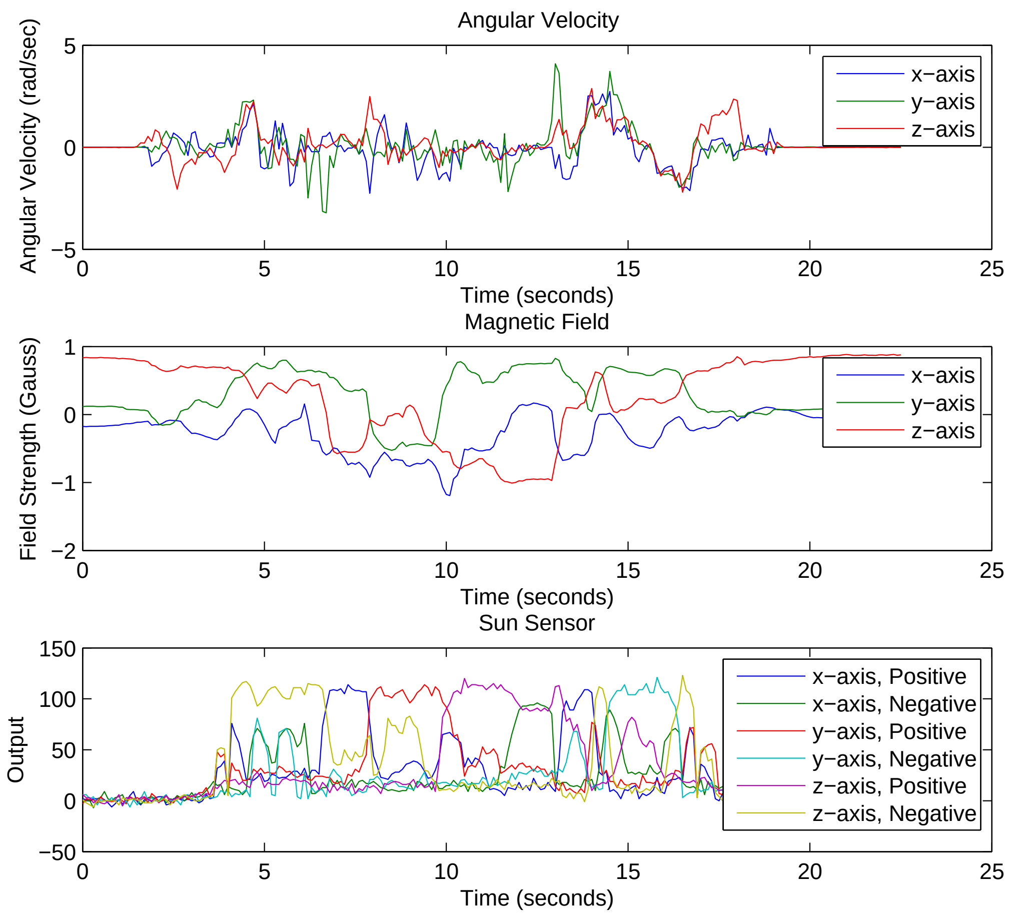


Figure 3.9: Moving the Prototype randomly in the sunshine

Chapter 4

ADCS Prototype

This chapter will show the different mechanical parts the prototype is made of and how it is assembled.

4.1 Inner frame

The inner frame is the mechanical structure that everything is mounted to. It is almost identical to the frame that will be used for the actual satellite. The only difference is the material used and how it is manufactured. The frame consists of six 3D printed parts, two square end pieces and four struts. The parts are screwed together creating a rectangular box. There are four holes on each side for mounting the coils, while tracks in the struts can support several modules (circuit boards).

4.2 Magnetorquer Design

NUTS will have two rectangular magnetorquer covering the sides and one smaller squared magnetorquer in the negative z plane. Further, they will be referred to as the X, Y and Z coil. A design was suggested in [22]. Since the z-coil covers a smaller area than the other coils, a different wire gauge was selected so that it could still produce the same amount of torque. Due to a calculation error, the physical size of the coils had to be expanded a bit. The new and altered dimensions of the coil frames give the design constraints seen in table 4.1. The minimum number of turns is decided by the maximum current the magnetic wire can withstand, given that it will be loaded with 3.3 volts. The maximum number of turns is a theoretical value of how much wire that can be wounded onto the coil frames. The calculations assume the wire is stacked

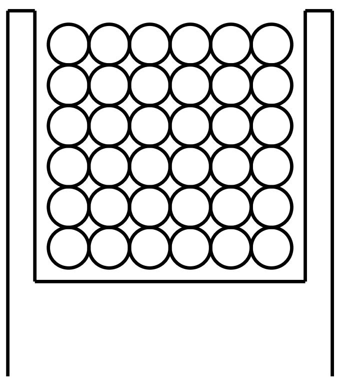


Figure 4.1: Cross Section of a magnetorquer showing theoretical stacking of the magnetic wire

as shown in figure 4.1. It is good to have a margin between minimum and maximum because it is difficult to align the wires perfectly. The torque of the coils depends solely on area, voltage and wire gauge. More diffuse and partially unconstrained factors like weight, current consumption and inductance plays a role when selecting the number of turns. That is why the desired number of turns are based on mere assumptions. The magnetorquers have four holes for mounting.

Table 4.1: Magnetorquer Design Constraints

Magnetorquer	X and Y	Z
Min turns	136	226
Max turns	199	254
Width	3 mm	3 mm
Inner Diameter (long side)	172 mm	67 mm
Inner Diameter (short side)	72 mm	67 mm
Outer Diameter (long side)	182 mm	83 mm
Outer Diameter (short side)	82 mm	83 mm

4.2.1 Production

The frames were designed in NX in cooperation with Anders Håland, who was responsible for NUTS's mechanical design. The Department of Cybernetics have a 3D printer that was used for manufacturing. A much earlier design of the frames was printed a while back where the finished parts got a permanent twist because the material was too thin (figure 4.2). That is why it was decided to add some thickness to the printed parts, which does not play a big difference for the prototype. The final design will ultimately be manufactured in another way and with a different material that does not twist.

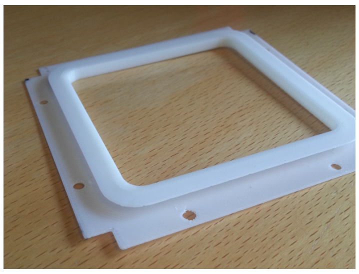


Figure 4.2: An image of the old coil frame. The material is too thin

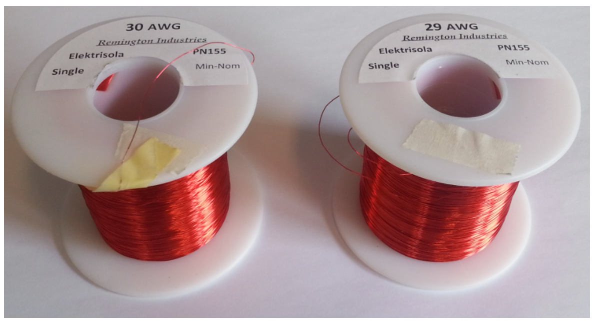


Figure 4.3: AWG-29 and AWG-30 magnetic wire. Ordered for Amazon

The magnetic wires were ordered from Amazon (figure 4.3), and the specifications can be found in table 4.2. Department of Electrical Power Engineering's workshop helped with the actual winding. They have both the experience and the right equipment for these kind of tasks. Manual feeding was used during the process, which resulted in a suboptimal stacking of the wires. This lead to fewer revolutions than the theoretical maximum, but still well within the minimum limit. Initially the idea was to glue two header pins directly to the frame where the magnetic wires exits. It seemed like a good and clean solution at first, but it proved difficult to solder the wire to the pins in that configuration. Instead, thicker wires with a header contact in one end were soldered directly to the magnetic wires. This seems like a better solution anyway because it leaves the option to choose any connector later. The flight model of NUTS will definitely not used header contacts. It is important that the coils are treated with care because the wires, and the insulation of the wires, are very fragile. An image of the coils can be seen in figure 4.4.

4.2.2 Testing

The resistance of the magnetorquers can be a good indicator to see if they work properly. The value should be fairly close to the theoretical value. Too low resistance can mean that the insulation is damaged and causes a short circuit. Too high resistance can happen if the soldering is bad. It is not straightforward to measure the resis-

Table 4.2: Magnetic wire specifications

Wire	AWG-30	AWG-29
Maximum Temperature	155 $°C$	155 $°C$
Diameter	0.274 mm	0.307 mm
Copper diameter	0.255 mm	0.286 mm
Insulation thickness	0.0097 mm	0.0107 mm
Insulation	Solderable Polyurethane w/ Polymide Overcoat	Solderable Polyurethane w/ Polymide Overcoat

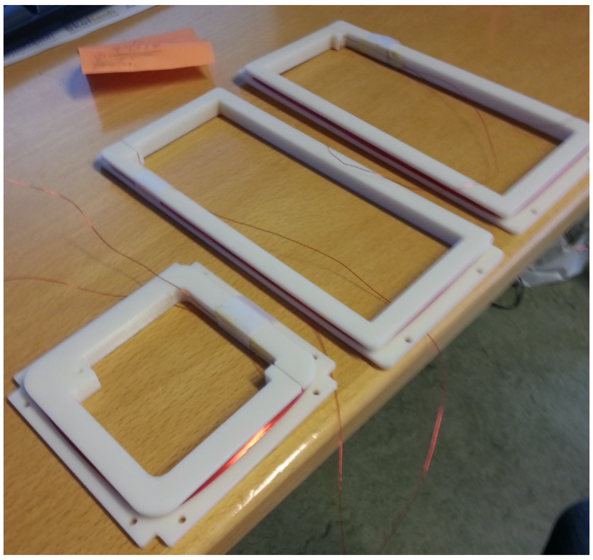


Figure 4.4: The three Magnetorquers before the Header cable was soldered on

tance using a multimeter because the coils are sensitive to the electromagnetic noise that surrounds us. It is also uncertain how a multimeter handles current transients caused by inductance. The solution is show in figure 4.5. First, a voltage source with a fixed output impedance was set to a known voltage. Then each of the magnetorquers were connected and the voltage drop were measured. The resistance can be found by inserting the voltage into equation 4.1.

$$R_c = \frac{v_c R_i}{1 - v_c} \tag{4.1}$$

where R_c is the coil resistance, R_i is the signal source's output impedance, and v_c is the measured voltage over the coil. The second unknown parameter of the magnetorquers is the inductance. It is complicated to calculate inductance of a rectangular squared multi-layered coil, and there is actually no exact solution to the problem. Finding the inductance experimentally is simpler. It can be done by replacing the DC source in figure4.5 with a signal generator, and use an oscilloscope instead of a multimeter.

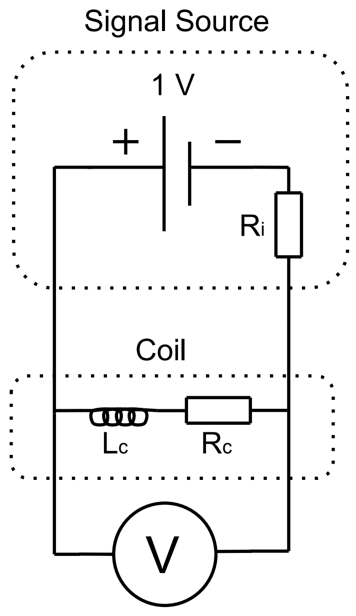


Figure 4.5: Circuit for measuring coil resistance. From top to bottom: Signal source with known impedance, Coil with resistance and inductance, Voltmeter

The following steps can be used [7]: First apply a high frequency sine voltage with a known amplitude to the coil. 20 kHz was used in this case. Then gradually reduce the frequency until the amplitude is exactly halved. This is the cut-off frequency of the RL-circuit (including the output impedance) and can be inserted into equation 4.5 to find the inductance. The results can be found in table 4.3 together with the calculated time constant. The results show that the theoretical resistance is not far from the measured resistance.

$$\frac{V_{scope}}{V_{gen}} = \frac{j\omega L + R_{gen}}{j\omega L + R_{gen} + R} = G \tag{4.2}$$

$$|G| = \frac{\sqrt{\omega^2 L^2 + R^2}}{\sqrt{\omega^2 L^2 + (R_{gen} + R)^2}} = \frac{1}{2} \tag{4.3}$$

$$\omega = 2\pi f \tag{4.4}$$

$$L = \frac{1}{f}\sqrt{\frac{(R_{gen} + R)^2 - 4R^2}{12\pi^2}} \tag{4.5}$$

$$\tau = \frac{L}{R} \tag{4.6}$$

Table 4.3: Magnetorquers Specifications

Parameter	X and Y-coil	Z-coil
Windings	155	238
R theoretical	26.7 Ω	19.2 Ω
R measured	27.2 Ω	18.5 Ω
f	590 Hz	580 Hz
L	8.53 mH	9.13 mH
τ	313 μs	494 μs

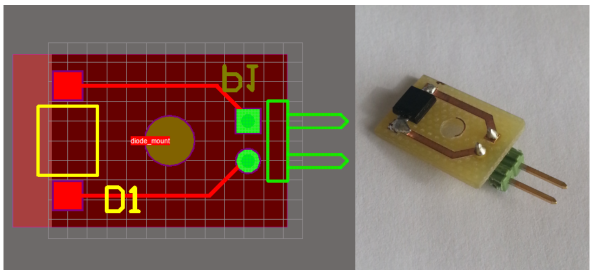


Figure 4.6: Design of photodiode PCB next to finished product

4.3 Photodiodes

Six small PCB's with a photodiode and a 2X header connector were made (see figure 4.6). The holes in the middle were made for easier mounting. One of these was placed on each side of the satellite. Three were fastened with screws, while the rest were attached using rubber band. It is important that nothing will cast shadows over the photodiode's surface.

4.4 Power supply

The power supply must be able to deliver enough energy. The magnetorquers alone will draw 421 ma at 3.3 volts. How much the rest of the circuit need is unknown and depends sampling frequency and such. The power supply used is a simple circuit board with one 3.3 volts and one 5 volt linear regulator. It can deliver up to 1 ampere which is more than enough. A standard 9-volt battery is used as power source. This power supply was originally intended for the NUTS Balloon campaign 2014.

4.5 Wiring

Special cables was made for connecting the three coils and the six photodiodes. The Bluetooth module is connected using four single wires, while the power supply is connected with a standard 2X header cable.

4.6 Assembly

The frame and the magnetorquers was easily screwed together. The ADCS module was inserted into one of the middle tracks of the frame. It was a very tight fit so no extra tightening was needed to hold it in place. Four out of the six photodiode PCBs was screwed to the frame, while the rest was attached using rubber band. Both the power supply and the battery was also mounted to the frame with rubber band. Figure 4.7, 4.8, 4.9 and 4.10 show the assembled ADCS Prototype.

4.7 Test Platform

The Earth's magnetic field is too weak for the ADCS Prototype to produce enough torque in order rotate in a non-frictionless environment. A strong magnetic field is therefore required in order to do some testing. The Helmholtz coil made in[3] could be used as a test bed for the ADCS Prototype. Just like in the EiT project, the prototype can hang from a thread supported by a magnet bearing, thus rotating freely about one axis. More about the Test Platform in chapter 6. Figure 4.11 shows and image of the Helmholtz coil.

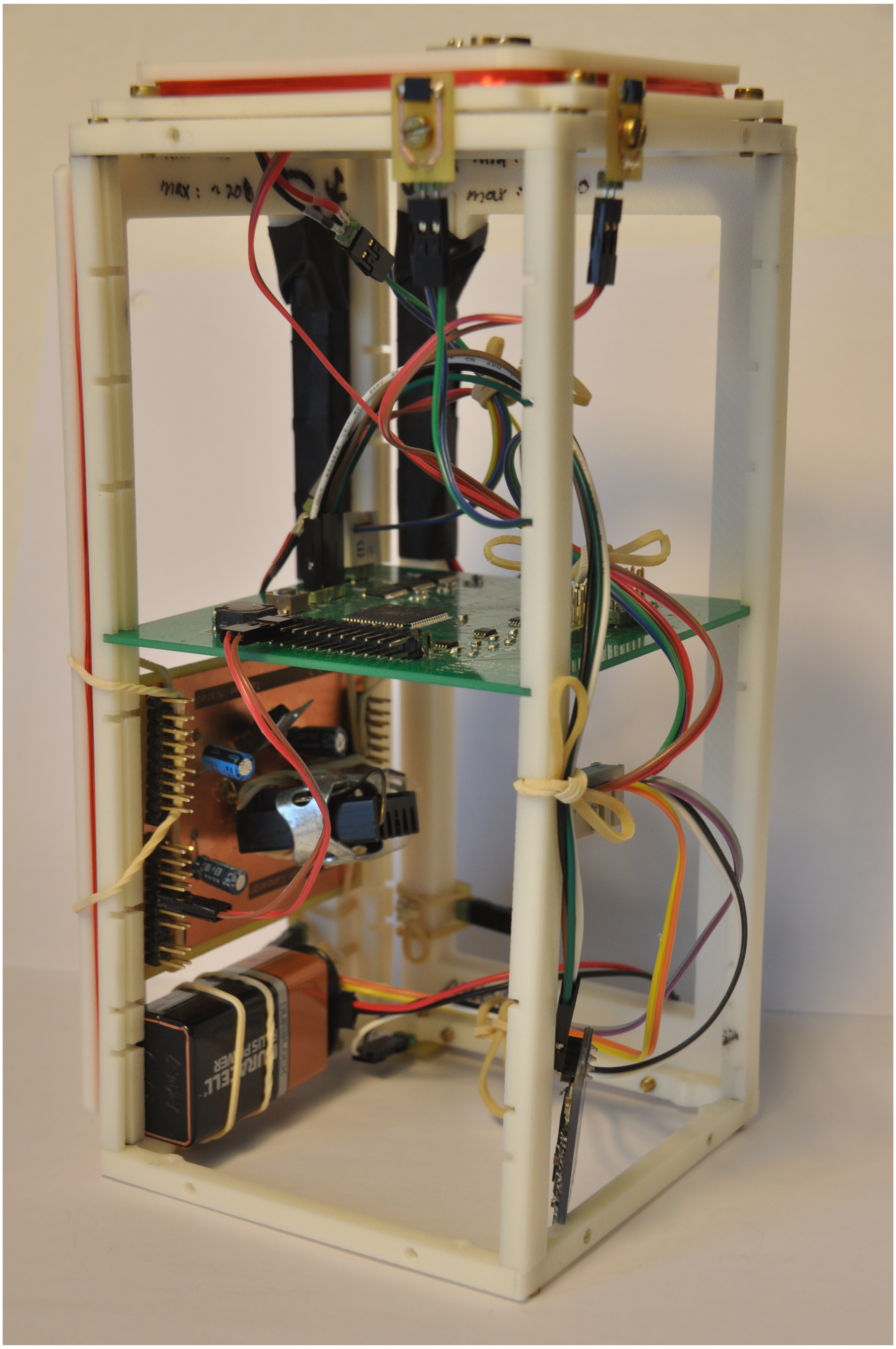


Figure 4.7: Finished ADCS Prototype

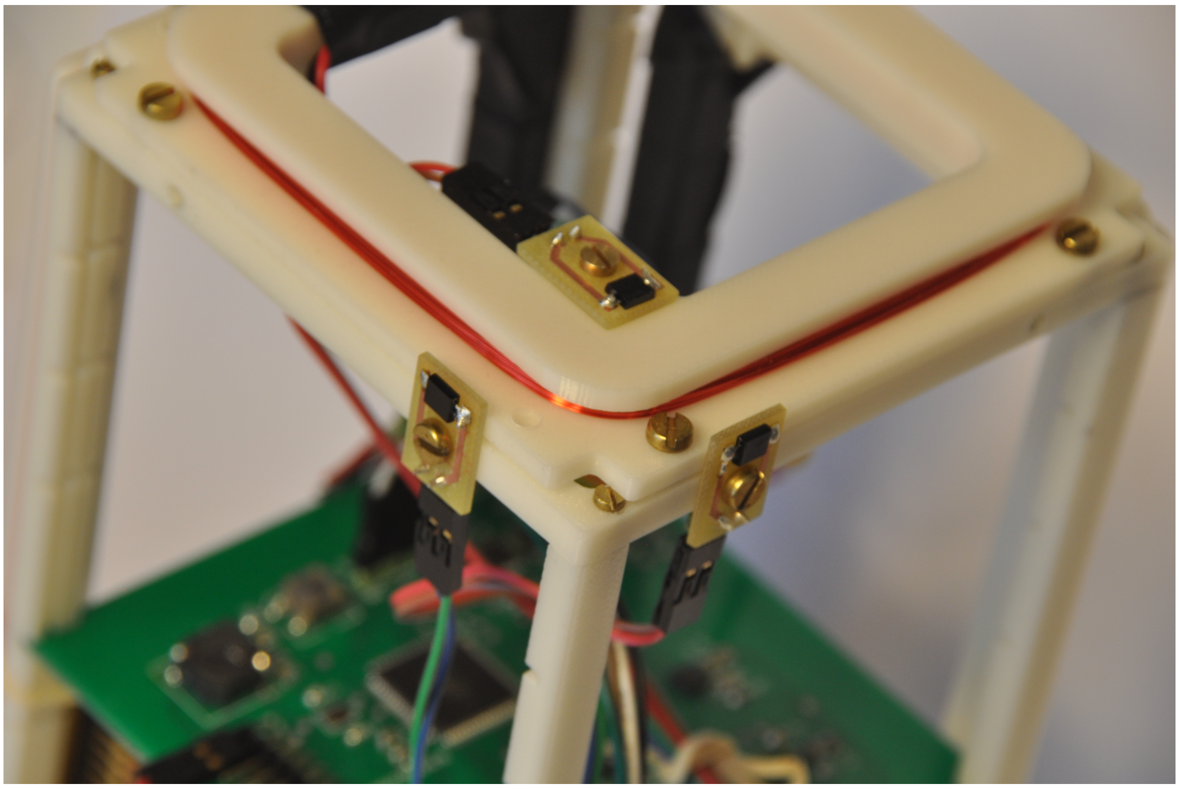


Figure 4.8: Finished ADCS Prototype. Showing three of the six photodiodes

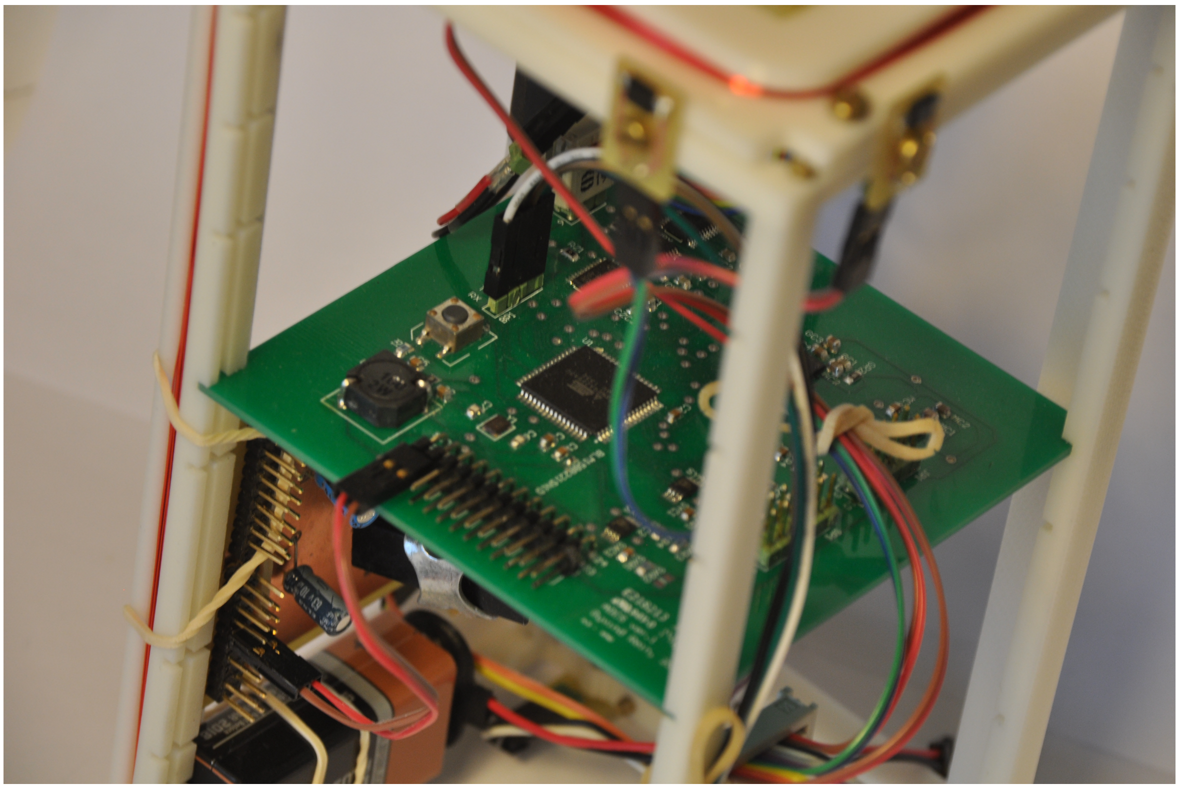


Figure 4.9: Finished ADCS Prototype. The ADCS Module is slided in place

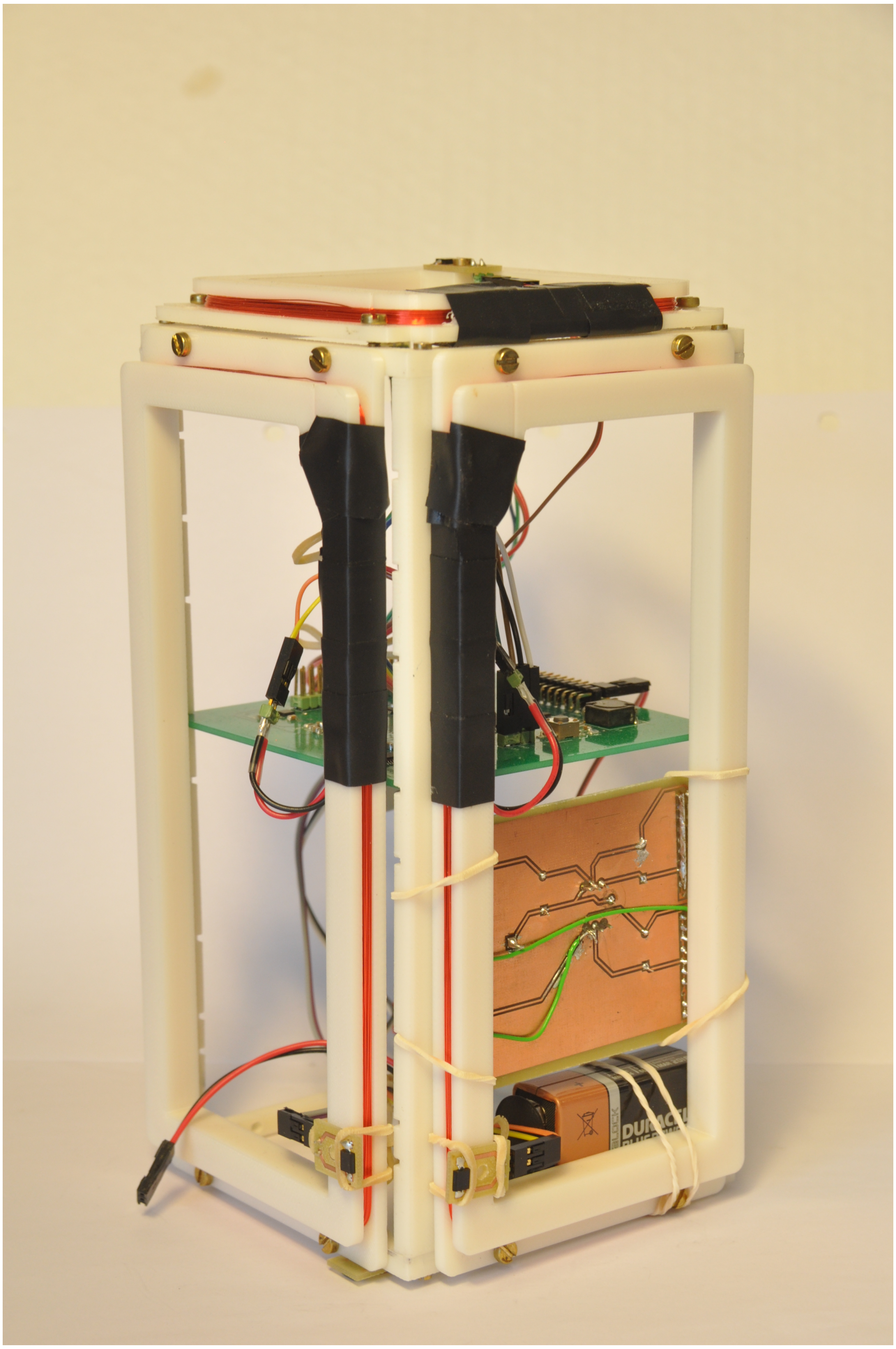


Figure 4.10: Finished ADCS Prototype. Other angle

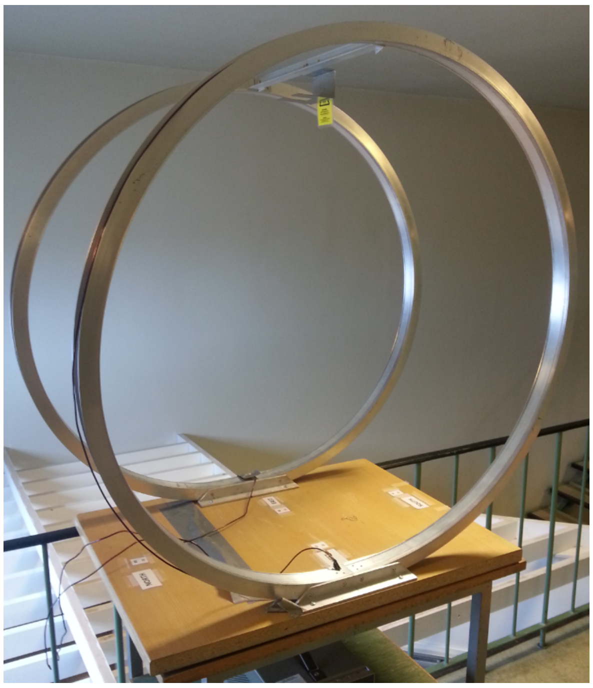


Figure 4.11: The Helmholtz Coil made in [3]. Can be used to test the ADCS Prototype

Chapter 5

Software

5.1 Drivers

The purpose of a driver is to abstract the underlying hardware. It should be simple and intuitive to use without any detailed knowledge of the physical components. Drivers for the different hardware components are separated into several c-files.

5.1.1 Magnetometer - HMC5983

The magnetometer communicates with the ATxmega256A3U through a SPI interface. It support a transfer speed of up to 8 MHz, but 2 MHz is used in order to be well within the margins. There are many options available for this chip. Sample Averaging lets you select the number of samples averaged per measurement output. Since low pass filtering can be done in software, this option seems unnecessary at this point and will only introduce a delay. This is because the maximum sampling frequency is 220 Hz, which is fairly slow to begin with. Sample averaging can be set to 1, 2, 4 or 8 samples. One of the best features of this sensor is the Temperature Compensation. With this turned on, the internal temperature sensor will be measured at each magnetic measurement and enable automatic compensation of sensitivity over temperature. Very useful for space applications in other words. It is also possible to adjust the Gain. This is how sensitive the sensor is. The prototype will hang in a strong magnetic field so that the magnetorquers can create a sufficient amount of torque. The maximum strength of the magnetic field is determined by when the magnetometer measurements become saturated. The gain is set to 230 which is the lowest option and will give the highest measurement range (± 8.1 Ga). This option should be set to 1090 (which is default) for

```
int magnetometer_init(){
        PORTC_DIR &= ~ 0b01000000;
        PORTC_DIRSET |= 0xB0; // SPI and control pins
        PORTC_OUTSET |= 0x10;

        SPIC_CTRL &= ~SPI_DORD_bm; // MSB first
        SPIC_CTRL |= SPI_MASTER_bm; // Master mode
        SPIC_CTRL |= SPI_MODE_3_gc; // SPI mode 3, Leading edge is
            falling. Samples are read on rising edge.
        SPIC_CTRL |= SPI_PRESCALER_DIV16_gc; // 2 MHz
        SPIC_CTRL |= SPI_ENABLE_bm; // SPI enable

        mag_set(MAG_REGA, (MAG_REGA_data_avg_0 | MAG_REGA_temp)); //
            No sample averaging, Temperature Compensation enable.
        mag_set(MAG_REGB, MAG_REGB_gain_230); // Gain is set to 230
        return 1;
}
```

Figure 5.1: Magnetometer (HMC5983) initialization and settings

the final flight model of NUTS in order to get a high measurement resolution. Figure 5.1 show how the SPI is enabled on PORT C, and how the magnetometer is configured using **mag_set(uint8_t reg, uint8_t byte)**. The most useful functions can be seen in the description below.

magnetometer_init() Initializes the magnetometer. SPI on PORTC is enabled and the correct options are set.

magnetometer_poll(int16_t data[**3**]**)** New x, y and z are collected and written to **data** in that same order.

magnetometer_temp(int16_t *temp) Measures the temperature, convert it to Celsius and write it to **temp**. The conversion formula can be found in the datasheet.

5.1.2 Gyroscope - L3G4200D

The gyroscope uses a SPI interface just like the magnetometer, which is why the driver is very similar to the one in figure 5.1. The differences are the registers use, the settings, and which SPI interface they use. The gyroscope is connected to PORTD on the microcontroller. Continuous Update is a setting that will automatically update the MSB and LSB registers whether or not they have been read. This function ensures that the values are fresh. There is also a setting for adjusting the resolution. The resolution is set by the maximum angular velocity (degree per second) the sensor can measure. 250 dps is chosen because it is the lowest option available, which gives the best accuracy. The output data rate is set to 100 Hz. It is uncertain how accurate the sensor is and how much drift is has. Some sort of temperature compensation might be necessary and should be implemented in the driver. The gyro has a built-in temperature sensor that can be utilized for that purpose. Below is a list of the most important functions in the driver.

void gyro_init() Initializes the gyroscope. SPI on PORTD is enabled and the correct settings are set.

void gyro_poll(int16_t data[3]) Captures the newest data from x y and z axis and write them to **data**.

void gyro_temp(int16_t *data) The gyroscope has an internal temperature sensor that can be read with this command. The value is written to *data*.

5.1.3 Sun sensor

The sun sensor consist of six photodiodes and six op-amp circuits. The analog output from each of these circuits are connected to pin 1-6 on PORT A. Two Analog-to-Digital Converters are available on the ATxmega256A3U, but together with the ADC Multiplexer Control, only one is required. There are several settings to choose from for the ADC. Single ended input with no gain is the correct setting for this setup. A voltage reference source, *VREF*, must also be selected. The most accurate source available is an internal bandgap voltage at 1 volt. Using this source gives consistent results independent of VCC. More details and settings are available in the microcontroller's datasheet[2]. The code snippet in figure 5.2 show how the MUX is used. **ADCA.CH0.MUXCTRL** is the register that sets the Multiplexer to a certain pin. **SS_XP** is the pin number of the photodiode that point towards **P**ositive **X**-axis, and so on. The analog signals are measured one by one, starting with **SS_XP** and ending with **SS_ZN**. The function **read_adc()** initiates a reading and returns the result. See figure 5.3 for details. The returned value is a function of the measured voltage and is given by the following equation:

$$result = \frac{(VINP + \Delta V)}{VREF} \cdot (TOP + 1) \qquad (5.1)$$

where *VINP* is the input voltage, *VREF* is the internal reference voltage, and *TOP* is the ADC bit resolution (4095 for 12 bit). $\Delta V = VREF \cdot 0.05$ is a fixed offset that can

```c
int sun_sensor_poll(int16_t data[6]){
        ADCA.CH0.MUXCTRL = SS_XP; // Set multiplexer
        data[0] = read_adc(); // Read signal

        ADCA.CH0.MUXCTRL = SS_XN;
        data[1] = read_adc();

        ADCA.CH0.MUXCTRL = SS_YP;
        data[2] = read_adc();
...
```

Figure 5.2: Analog-to-Digital Converter MUX Control

```c
uint16_t read_adc(){
        ADCA.CH0.CTRL |= ADC_CH_START_bm; // Start conversation
        while(!ADCA.CH0.INTFLAGS); // Wait for flag
        ADCA.CH0.INTFLAGS = 0x0f; // Clear Flag
        return ADCA.CH0RES; // Returns the result
}
```

Figure 5.3: Using the Analog-to-Digital Converter

be neglected with calibration. This means that 1 volt will give a result of 4095, while 0 volt returns 0. For now, the sun sensor driver only produce raw sensor data. Future work will involve implementing an algorithm that outputs the sun vector. The method in [19] can be used, or something similar. Next comes a description of the most useful functions created so far.

sun_sensor_init()

Enables the 12 bit ADC with single ended input, no gain, and bandgap voltage reference. Production test calibration is loaded as well.

sun_sensor_poll(int16_t data[6])

Measures the voltage of the six sun sensor circuits and inserts the values into **data**, starting with positive x-axis and ending with negative z-axis. Equation 5.1 describes the relation between a measured voltage and a returned value.

5.1.4 Coil driver

The h-bridges that control the magnetorquers have three control signals, PHASE, PWM, and SLEEP, and one status signal, BR_FAULT, which is connected to the microcontroller. These signals can be found in figure 3.1a from the ADCS Module chapter. PWM and PHASE are used operate the magnetorquers. High PWM turns them on while high and low PHASE sets the direction. Table 5.1 is a truth table that show input versus output for the x-axis magnetorquer. The control signals are easily manipulated in software. A code snippet of **set_coil(int coil, int dir)** can be found in figure 5.4. SLEEP can be used to put the h-bridges into power-saving mode in order to save energy. BR_FAULT can indicate if something is wrong or not working properly. A fault has occurred if this signal is driven low. More information about faults can be found in the datasheet DRV8834. At this point, the functionality of SLEEP and BR_FAULT have neither been implemented nor tested.

It is obvious that a mechanism for turning the magnetorquers on and off is not enough to achieve precision control. Pulse-Width Modulation is a method that can be applied to solve this problem 2.5. ATxmega256A3U has several built-in PWM interfaces that can run independent from software after they have been initialized. However, this gives less control to the software. It is important that the magnetic field is measured when the magnetorquers are off. This issue can be difficult to manage when the PWM interfaces are used. The solution was to do everything in software and thereby retain more control. The function **coil_output(int16_t output_vector**[**3**]) has been implemented. The input parameter is an array containing the desired coil output values for x, y and z-axis magnetorquer. The inputs can range from -2047 to 2048, which gives a 12-bit accuracy. These limits can be set to any value, but it seems reasonable to give the magnetorquers the same bit resolution that the sensors have. Output values outside the given range will be cropped. The function does not work exactly like normal PWM because it only runs for one cycle. Figure 5.5 together with the code in figure 5.6 gives a good illustration of how it works. First the coils are turned on with a direction according to the sign of the given output value. Then a timer/counter is initiated and three Compare and Capture Channels are enabled, one for each coil (code line 20-22). Each CC Channel will trigger an interrupt when the timer/counter reaches a certain value. When this happens, the corresponding magnetorquer will be turned off. The interrupt signal from CC Channel A will for example turn off the x-coil (code line 28-30). The magnetorquers will stay idle until next time the function is called. The maximum time a coil can be enabled (-2047 or 2048) is defined by COIL_PER (code line 17). For the detumbling test in chapter 6, COIL_PER was set so that the coils could be enabled for maximum 88.0 ms each cycle. More on how to set COIL_PER can be found in the microcontroller's datasheet. Next is a description of the most useful coil driver functions.

Table 5.1: H-Bridge Truth Table

CX_PWM	CX_PHASE	CX_OUT1	CX_OUT2	Direction
0	X	0	0	Off
1	0	L	H	Positive
1	1	H	L	Negative

```
int set_coil(int coil, int dir){ // e.g. set_coil(COIL_X, COIL_POS)
        if (coil == COIL_X){
                if (dir == COIL_OFF){
                        PORTD_OUT &= ~CX_PWM;
                }
                else if (dir == COIL_POS){
                        PORTD_OUT &= ~CX_PHASE;
                        PORTD_OUTSET |= CX_PWM;
                }
                else if (dir == COIL_NEG){
                        PORTD_OUTSET |= CX_PHASE;
                        PORTD_OUTSET |= CX_PWM;
                }
        }
        ...
```

Figure 5.4: Code snippet for controlling the x-axis magnetorquer

void set_coil(int coil, int dir)

Set the direction of a coil. Use the definitions COIL_X, COIL_Y, COIL_Z, COIL_OFF, COIL_POS and COIL_NEG as arguments.

void coil_output(int16_t output_vector[3])

Takes an array as argument containing the output values for the magnetorquers. Will initiate a single PWM cycle and utilizes timer/counter and Compare and Capture Channel interrupts to turn the magnetorquers off. This allows for other code to be executed during the process. The inputs can range from -2047 to 2048 which gives a 12-bit accuracy. Output values outside the given range will be cropped.

int coils_are_off() This function will return 1 if the coil are off, and 0 if they are on. It can be used as a safety measure before magnetic data are acquired.

5.2 Serial Communication via Bluetooth

Since the prototype must be able to move freely, it has to have wireless communication. The Bluetooth module is very easy to set up. All that is required is a computer or

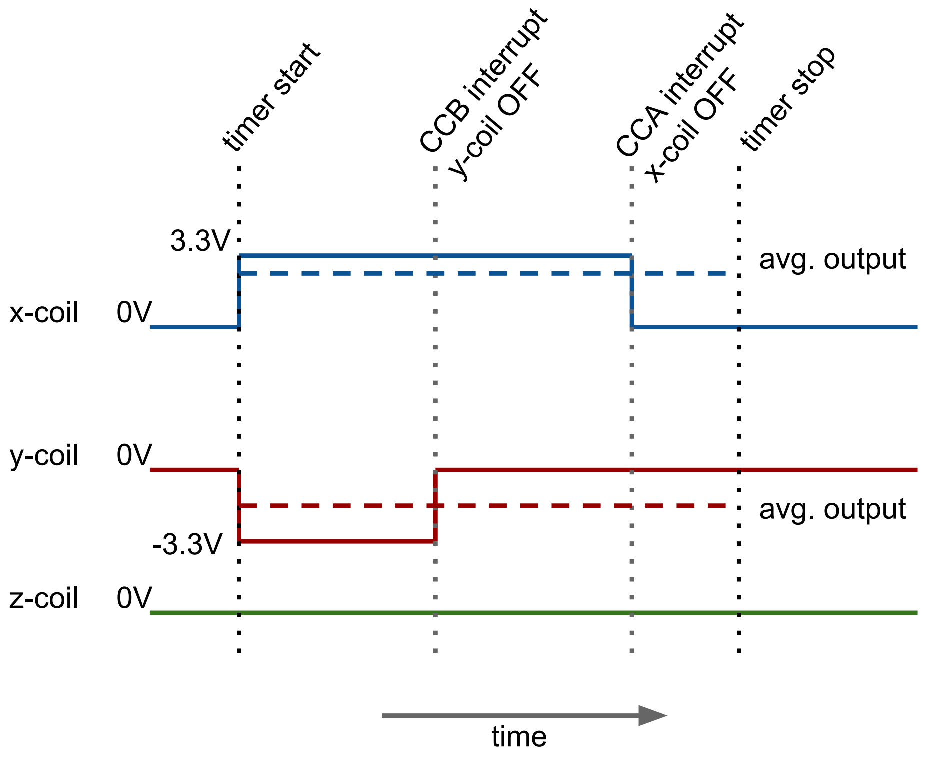


Figure 5.5: Example showing **coil_output**([**1600, -800, 0**])

```
int coil_output(int16_t output_vector[3]){
        coils_off = 0;
        int16_t coil_x_output = output_vector[0];
        if (coil_x_output < 0){
                coil_x_output = -1*coil_x_output;
                set_coil(COIL_X, COIL_NEG);
        }else{
                set_coil(COIL_X, COIL_POS);
        }
        if (coil_x_output > COIL_MAX) {
                coil_x_output = COIL_MAX;
        }
        int16_t coil_y_output = output_vector[1];
        ...

        TCC0.CTRLA = TC_CLKSEL_DIV256_gc;
        TCC0.PER = COIL_PER;
        PMIC.CTRL = PMIC_LOLVLEN_bm;
        TCC0.INTFLAGS = 0b01110001;
        TCC0.CCA = (COIL_PER*(float)coil_x_output)/COIL_MAX;
        TCC0.CCB = (COIL_PER*(float)coil_y_output)/COIL_MAX;
        TCC0.CCC = (COIL_PER*(float)coil_z_output)/COIL_MAX;
        TCC0.CTRLFSET = TC_CMD_RESTART_gc;
        TCC0.INTCTRLB = 0b00010101;
        TCC0.INTCTRLA = TC_OVFINTLVL_LO_gc;
}

ISR(TCC0_CCA_vect){
        set_coil(COIL_X, COIL_OFF);
}
```

Figure 5.6: PWM control code

```
int read_message(){
        if (!message_received){
                return 1;
        }
        if (compareStrings(rx_buffer, "i")){
                mode = MODE_IDLE;
        }
        else if (compareStrings(rx_buffer, "d")){
                mode = MODE_DETUMBLING;
        }
        else if (compareStrings(rx_buffer, "get_log")){
                get_log();
        }
...
```

Figure 5.7: ASCII Message System

a phone with Bluetooth. It works by emulation an UART connection configured at a baud rate of 9600 bps. The module's broadcasted ID is "OmniTek_Bluetooth", and it uses the paring code "1234". In windows, a virtual COM port will automatically appear that can be opened with Putty or any other application that uses serial communication. The module itself has four pins, GND, VCC, RX and TX. It is printed on the PCB that it need 5 volts, but fortunately it works for 3.3 volts as well. These pins are connected to the Bluetooth connector on the ADCS module. Note that RX on the Bluetooth module must be connected to TX on the ADCS module, and vice versa. The microcontroller is configured the same way it would for a normal UART interface. An ASCII messaging system was implemented because it is easy to understand and to use. A message can be of any length and is terminated by either carriage return or newline ('\r' or '\n'). The code snippet in figure 5.7 show how a simple message is read and executed using **read_message()**. The message "get_log" will for example run the function **get_log()**, while the message "d" turns on detumbling mode. An interrupt is triggered every time a character is received in the UART data register. The character is then stored to the message buffer. This is carried out by the interrupt function **ISR(USARTF0_RXC_vect)**. If the character reads '\r' or '\n', the global variable **message_received** is set to 1, indicating that a complete message has been received. The following functions covers the basic functionalities.

read_message()

If a complete message has been received the corresponding action will be executed. If the message is non-valid it will be discarded. If a message has been received, this function must be executed before a new message can be accepted.

int printf (const char * format, ...)

Printf is used to send messages. The function is well documented on the Internet.

bluetooth_init()

Calling this function will enable UART on PORTF with a baudrate of 9600 bps. This is also where **stdout** is defined in order of **printf** to work.

This messaging system works great for the ADCS prototype, but must be redesigned in the future. First of all the ADCS Module will communicate through the backplane using I^{2}C instead of UART. Second, it will have to use the well-defined Cubesat Space Protocol (CSP) [8].

5.3 Implementation of Detumbling algorithm

In this section, the implementation of the B-dot detumbling algorithm (section 2.2) and the control loop will be explained. The B-dot implementation can be divided into two parts. First part is to find the estimate of $\hat{\dot{\boldsymbol{B}}}$ of $\dot{\boldsymbol{B}}$. It can be found with the discrete-time filter in equation 5.2.

$$\hat{\dot{\boldsymbol{B}}}_k = a_2\hat{\dot{\boldsymbol{B}}}_{k-1} + b_2(\boldsymbol{B}_k - \boldsymbol{B}_{k-1}) \tag{5.2}$$

$$\boldsymbol{m}^b = -\boldsymbol{k}\hat{\dot{\boldsymbol{B}}}^b, \qquad \boldsymbol{k} \subset \Re^3, \quad \boldsymbol{k} > 0 \tag{5.3}$$

The c-code implementation of the filter can be seen in figure 5.8. More information on the filter, and how to tune the filter parameters, can be found in [5, p. 16]. The second part of the implementation is the B-dot control equation. By replacing $\dot{\boldsymbol{B}}$ with $\hat{\dot{\boldsymbol{B}}}$ in equation 2.1, we get equation 5.3. The implementation can be seen in figure 5.9. B_DOT_X_GAIN, B_DOT_Y_GAIN and B_DOT_Z_GAIN (line 1-3) are the components of $\boldsymbol{k}$ and decide how much gain the magnetorquers have. Normally these values should be equal. With the required b-dot equations implemented, the next step was to set up the control loop (figure 5.11). It was decided to use a state machine -like approach, with the two states; IDLE and DETUMBLE. The different *modes* (states) can be set through serial communication. The code is written so that it is possible to add new modes later, for example the suggested modes in section 1.2. The loop is driven by a timer that is initiated in **init()** (line 3). The timer defines the frequency of the control loop. Every time the timer runs out, the global variable **new_cycle** is set to 1 (true).

```c
#define A_2 0.5
#define B_2 0.5

volatile int16_t b_dot[3];
volatile int16_t b_dot_prev[3];
volatile int16_t b_prev[3];

void b_dot_estimate(int16_t mag_data[3]){
        for(int i = 0; i < 3; i++){
                b_dot_prev[i] = b_dot[i];
                b_dot[i] = ((A_2 * b_dot_prev[i]) + (B_2 * (mag_data[i] - b_prev[i])));
                b_prev[i] = mag_data[i];
        }
        return;
}
```

Figure 5.8: Implementation of discrete-time filter to find $\hat{\dot{\boldsymbol{B}}}$

```c
#define B_DOT_X_GAIN 20.0
#define B_DOT_Y_GAIN 20.0
#define B_DOT_Z_GAIN 20.0

void b_dot_control(int16_t b_dot[3], int16_t coil_output[3]){
        coil_output[0] = (-1) * B_DOT_X_GAIN * b_dot[0];
        coil_output[1] = (-1) * B_DOT_Y_GAIN * b_dot[1];
        coil_output[2] = (-1) * B_DOT_Z_GAIN * b_dot[2];
        return;
}
```

Figure 5.9: Implementation of the B-dot control equation

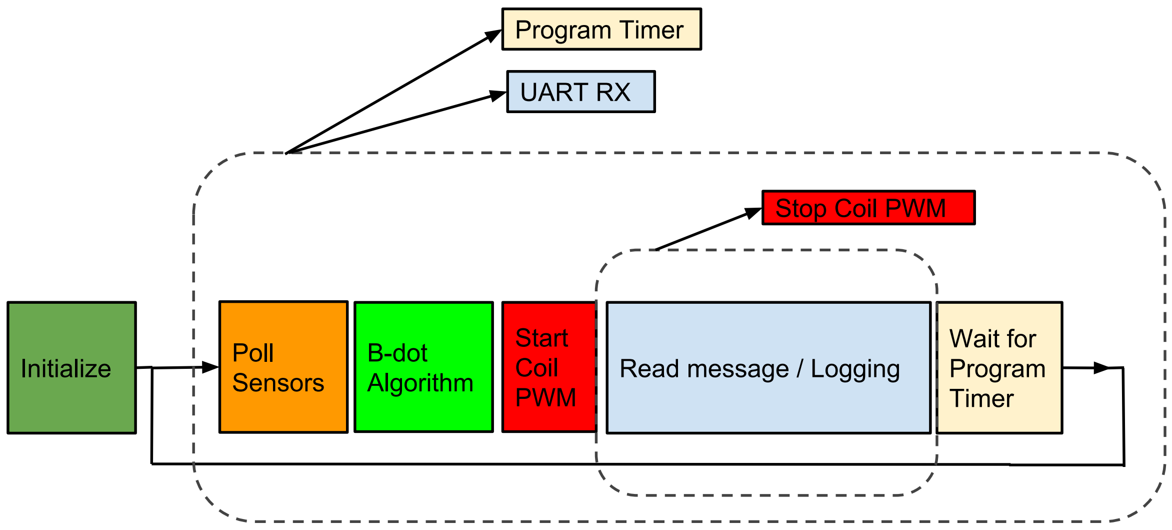


Figure 5.10: This figure show the program flow when detumbling is turned on. The dashed squares illustrate where the different interrupts are enabled

The program is waiting for this signal in code line 5, and clears the variable right afterwards. **new_cycle** checked again at the end of the loop. This is to make sure that the code in between has finished before the next cycle. It is considered as a hard deadline. For the testing in chapter 6, the loop is set to run at 10 Hz. The illustration in figure 5.10 show the program flow when detumbling is enabled.

5.4 Data Logging

If NUTS is launched successfully, the attitude data will be of high interest. This will essentially be the proof that the estimation and control algorithms work. Moreover, if it does not work, a log can be used for troubleshooting. It will also give the opportunity to experiment with different constants and gains, or even algorithms. The satellite's On-Board Computer will have its own system for logging, and it was up for discussion whether the attitude data should be integrated in that log. It was decided to keep them separate for the time being. The reason it that the OBC is most likely to update its log continuously at a given rate, while the attitude data is more interesting for given periods. For instance, to log the transients of the initial detumbling phase. A drawback is the limited memory available in the ATxmega256A3U microcontroller the ADCS uses. This might be an argument for adding external memory to the module. Logging can also be useful while testing the prototype, which is why this functionality has been implemented at this point. It can be done with the functions described below. 16-bit integers are used because the raw sensor data vary from 12 to 16-bit accuracy.

```c
int main(void)
{
        init();
        while(1){
                while(!new_cycle);
                new_cycle = 0;
                poll(sensor_data);
                add_to_log(sensor_data);
                switch(mode){
                        case MODE_IDLE:
                                output_vector[0] = 0;
                                output_vector[1] = 0;
                                output_vector[2] = 0;
                                break;
                        case MODE_DETUMBLING:
                                b_dot_update(&sensor_data[0],
                                   output_vector);
                                coil_output(output_vector);
                                break;
                        default:
                                break;
                }
                read_message();
                while(!coils_are_off());
                _delay_ms(1);
                if (new_cycle == 1){
                        printf("Timer is too fast!\n");
                }
        }
}
```

Figure 5.11: Main loop and state machine

For further development, it might be a good idea to add timestamps to the log.

clear_log()

Set the log pointer to zero. The samples are not actually overwritten.

int add_to_log(int16_t data[DATA_SIZE])

This function will add the data vector of length DATA_SIZE to the log and increase the log pointer. Returns 1 on success and 0 if the log is full.

get_log()

Prints the entire log to the serial port in a CSV format.

5.5 LabView

LabView is a very powerful tool to make control applications. It can display graphs, which is a very good way to represent raw sensor data. Switches and buttons can be used to send commands through the serial port. Therefore, it seemed reasonable to use LabView for this project. It is better to spend a little extra time developing a graphical application than to spend hours interpreting serial data in a command line. LabView uses a drag-and-drop programming language, which has its ups and downs. For the most part, it is very intuitive, but it was sort of a hassle to figure out how to use serial communication and how to interpret strings correctly. In this LabView project, there are two windows; the Front Panel (figure 5.12), and the Block Diagram (figure 5.13). The Front Panel is where the visuals are and is what the user will see. In this case, it contains COM port settings, four charts, and three control buttons. After a connection is established, the program will start polling the ADCS Prototype for sensor measurements and coil output data, and display values in the four charts in real-time. Pushing the buttons Idle, Detumbling or Tumbling will set the prototype to the corresponding mode.

The programming is done in the Block Diagram window. The VISA function blocks are used for sending and receiving serial messages. There is no point in explaining the whole diagram, but a few parts of it will do. Figure 5.14 shows what happens when the Detumble button (on the front panel) is pushed. The box that surrounds the blocks is an if-statement. If the button is pushed, the blocks inside will be executed. The character 'd' and carriage return ('\r') are put together to a string, and then used as an argument to the block that says VISA. This block will then write the string

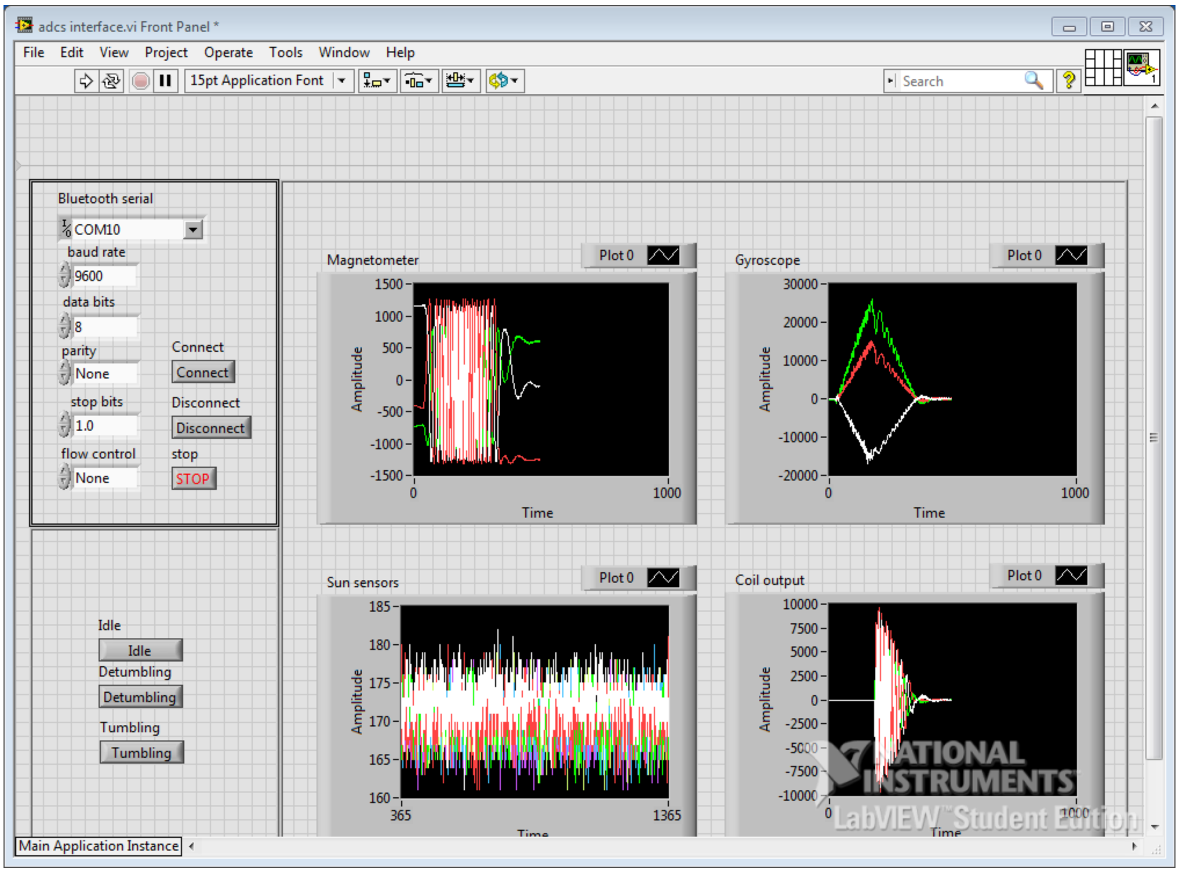


Figure 5.12: LabView. Front Panel of **adcs_interface.vi**

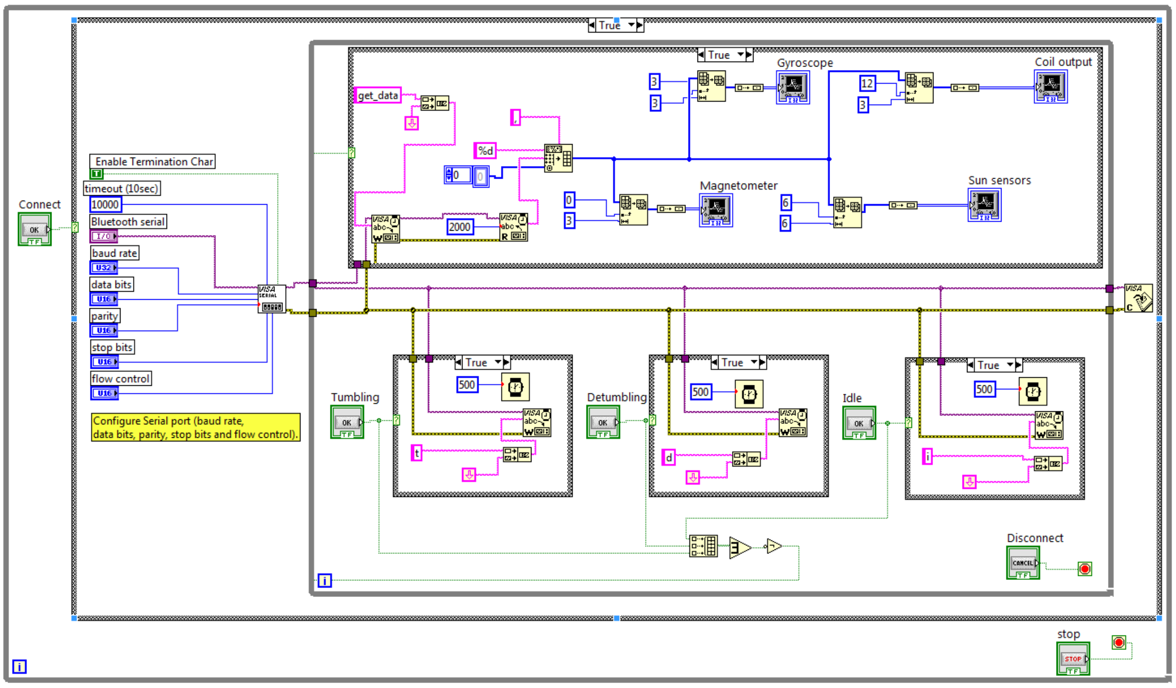


Figure 5.13: LabView. Block Diagram of **adcs_interface.vi**

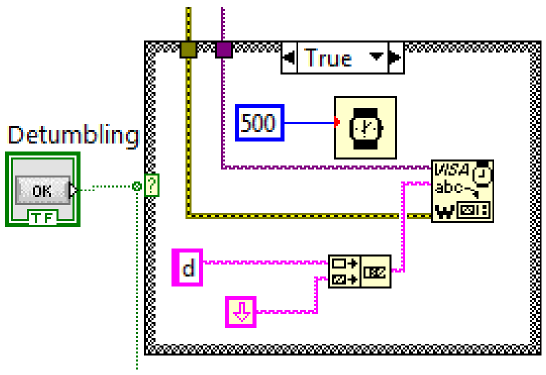


Figure 5.14: If the detumbling button is pushed, the program will send 'd\r' to the serial port.

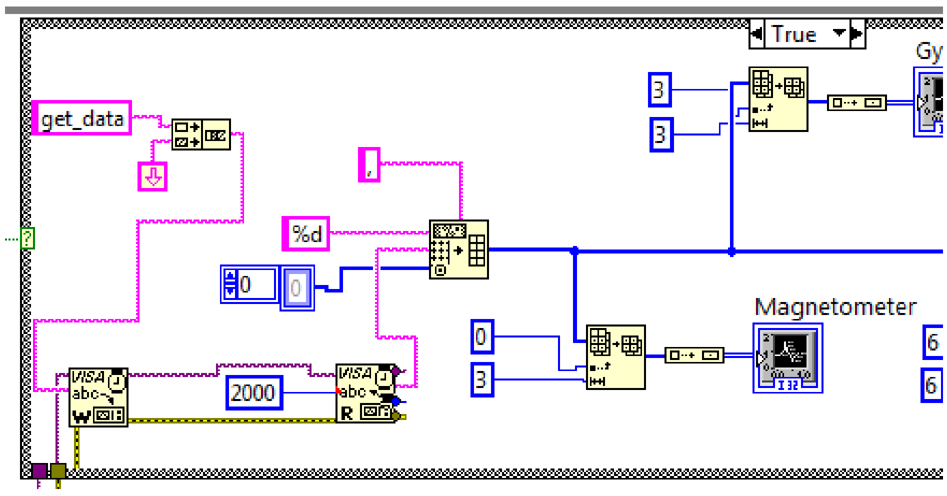


Figure 5.15: Polling the ADCS Prototype for data

to the serial port. When the Prototype receives the string it will change its mode to detumbling (section 5.2). The block that has a watch inside is a timer. The program will not exit the if-statement until the timer has run out. This was implemented so that the Prototype has time to execute the command before it is polled again.

Figure 5.15 show how the polling is executed. The program first sends the command 'get_data', and then it starts to listen for a response using the VISA Read block. The ADCS Prototype will return all the data as a string in CSV format. The data is then formatted to an array using a Spreadsheet String To Array block, which is further sent to the chart blocks.

LabView works great for testing the prototype, but it does not have to stop there. NUTS will also need some kind of software running on the ground station that controls the ADCS, and there is not good reason for why it should not be LabView. But of course, a lot more functionality must be integrated if so. Data logging, parameter tuning, and resetting, to name a few. LabView also has a 3D viewer that would be perfect for orbit and attitude representation. There is, in other words, no doubt that the required resources are available.

Chapter 6

Detumbling test

In this chapter, the performance of the B-dot detumbling algorithm is demonstrated using the test platform. The theory can be found in section 2.2, while details about the implementation is located in section 5.3.

6.1 Test Setup

An illustration of the setup can be seen in figure 6.1, while a few images can be found in figure 6.2. The prototype was tied to the metal rod extended from the magnetic bearing. The length of the thread was so that it hung approximately in the middle of the Helmholtz coil. A 10 A DC power source was connected to the Helmholtz coil. It was desirable to have as strong magnetic field as possible because it would give the magnetorquers a higher torque. The maximum magnetic field is limited by the range of the magnetometer. The current was set to 6 ampere, just under the limit of where the magnetic measurements were saturated. The following scenarios were tested:

1. Detumbling spin around the z-axis, clockwise
2. Detumbling spin around the z-axis, counterclockwise
3. Detumbling spin around the x-axis
4. Detumbling spin around all axes

For all the scenarios, the prototype was manually spun up to an angular velocity of about 4 rad/sec. Then, as the rotation decayed, all the sensor data and the magnetorquer output data were sent to LabView through the Bluetooth serial interface. Every

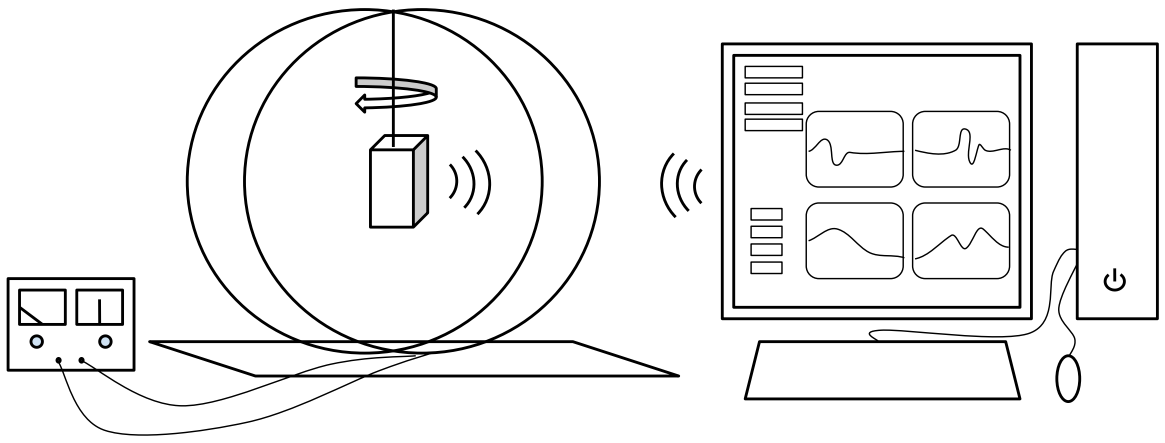
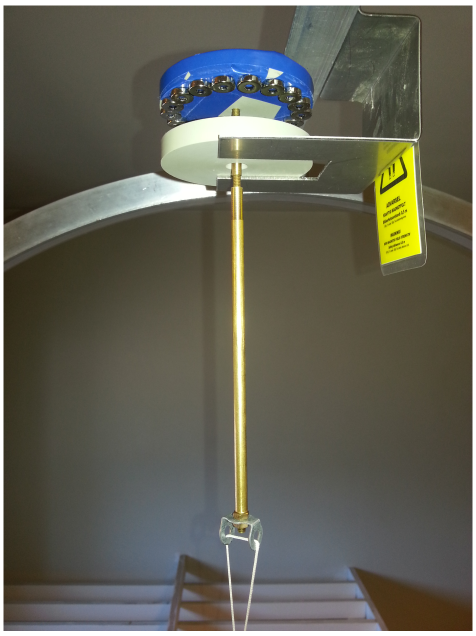


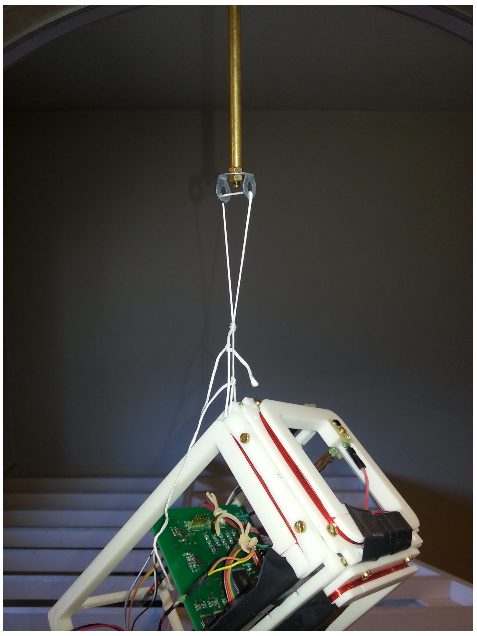
Figure 6.1: An illustration of the test setup. From left to right: Power source, Helmholtz coil with ADCS prototype inside, Computer running LabView. Wireless communication through bluetooth

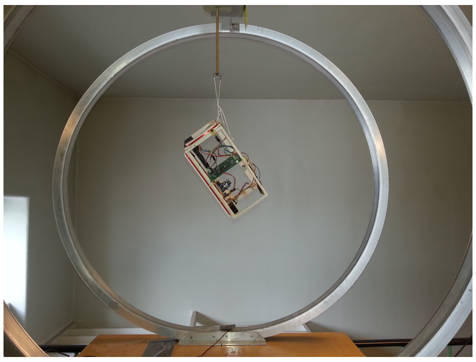
scenario was tested once with detumbling enabled and once without. The angular velocity was expected to decay faster when detumbling was enabled.

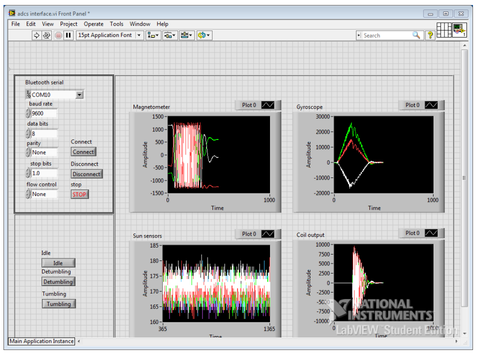
6.2 Results

The results are presented in four similar figures (figure 6.3, 6.4, 6.5 and 6.6), one for each scenario. Each figure contains three plots. The first plot shows angular speed, with and without detumbling enabled. The two time series are derived from x-, y- and z-axis gyroscope data. The second plot displays the magnetic field during detumbling. Measurements are converted from raw sensor output to Gauss. The final plot shows the magnetorquer output that is produced by the B-dot algorithm. Note that the values are 12 bit signed integers, which is related to the PWM duty-cycle. More information about the integers can be found in section 5.1.4.


(a) Magnet bearing and metal rod


(b) The prototype is tied to the metal rod


(c) Helmholtz coil with ADCS prototype inside



(d) LabView ADCS interface

Figure 6.2: Images of the test setup

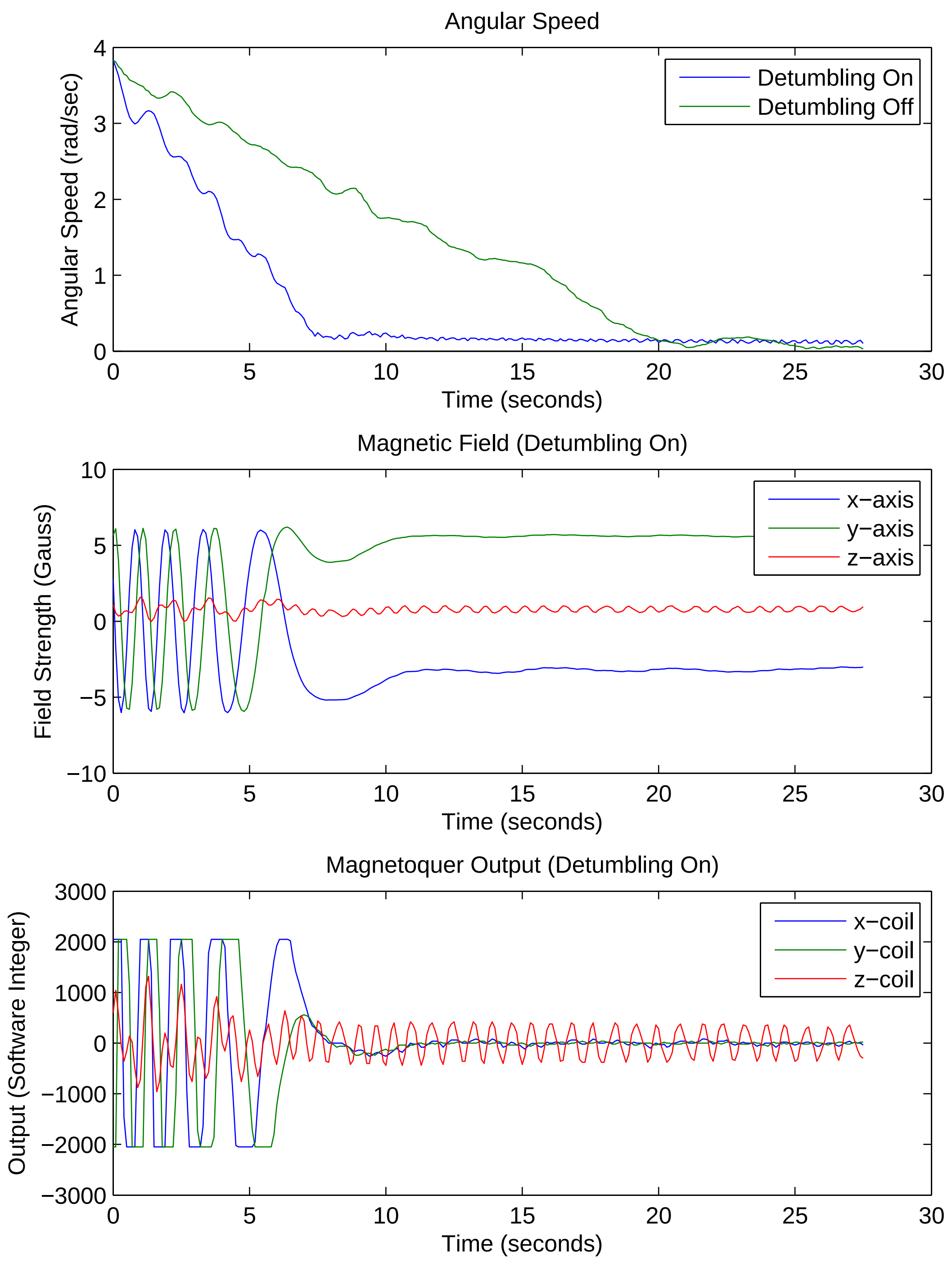


Figure 6.3: Rotation about the Z-axis, Counterclockwise

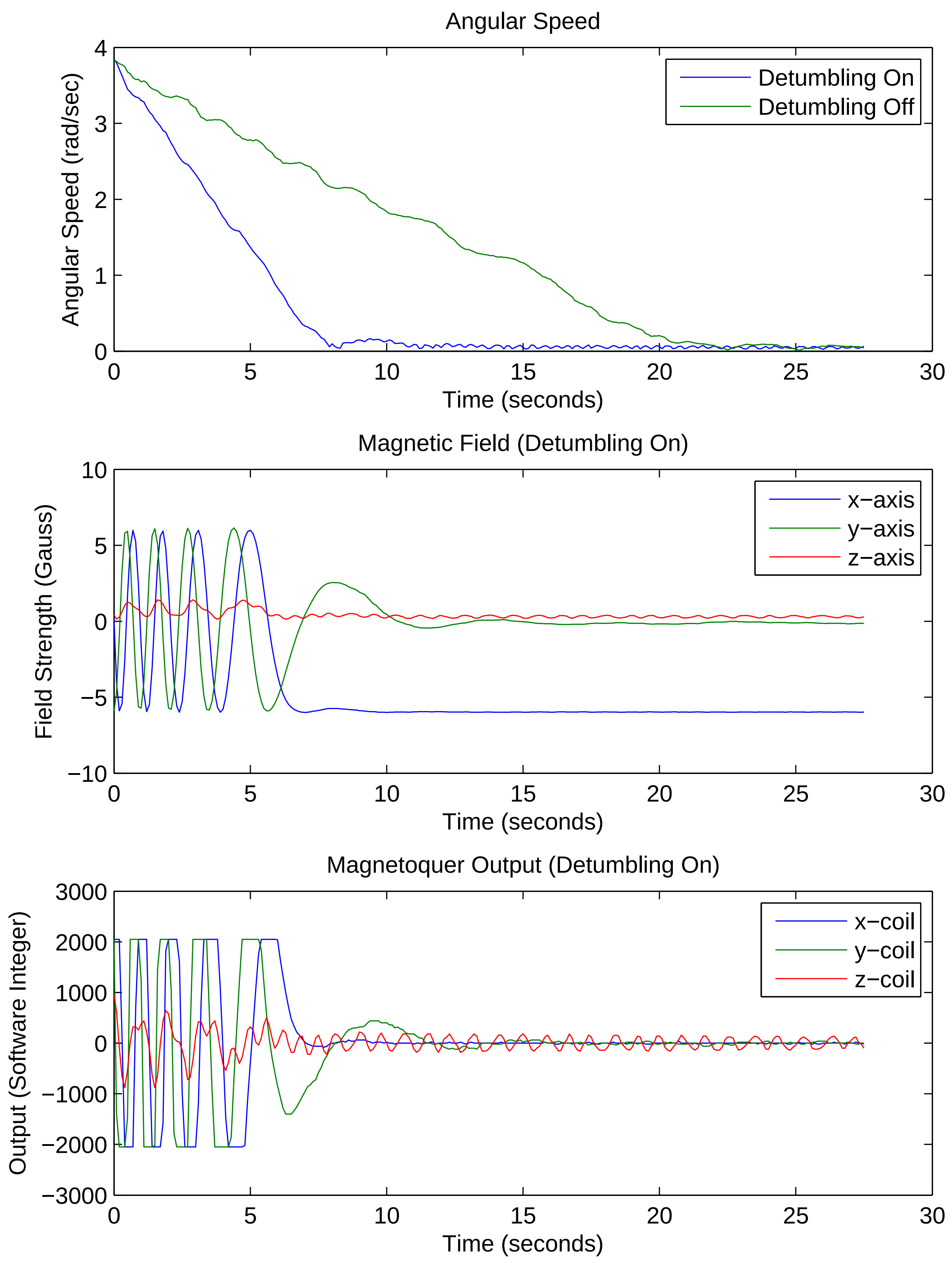


Figure 6.4: Rotation about the Z-axis, Clockwise

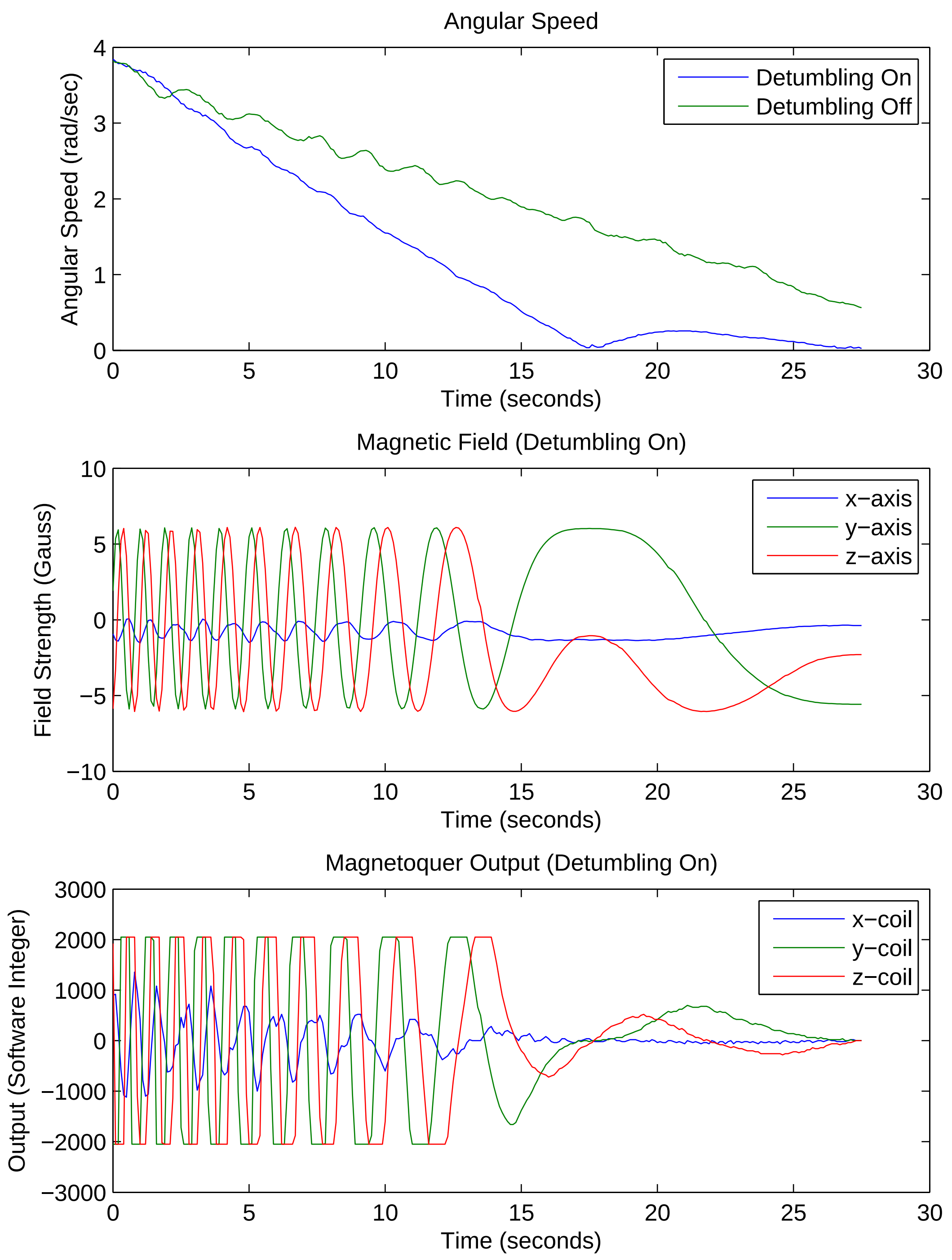


Figure 6.5: Rotation about the X-axis

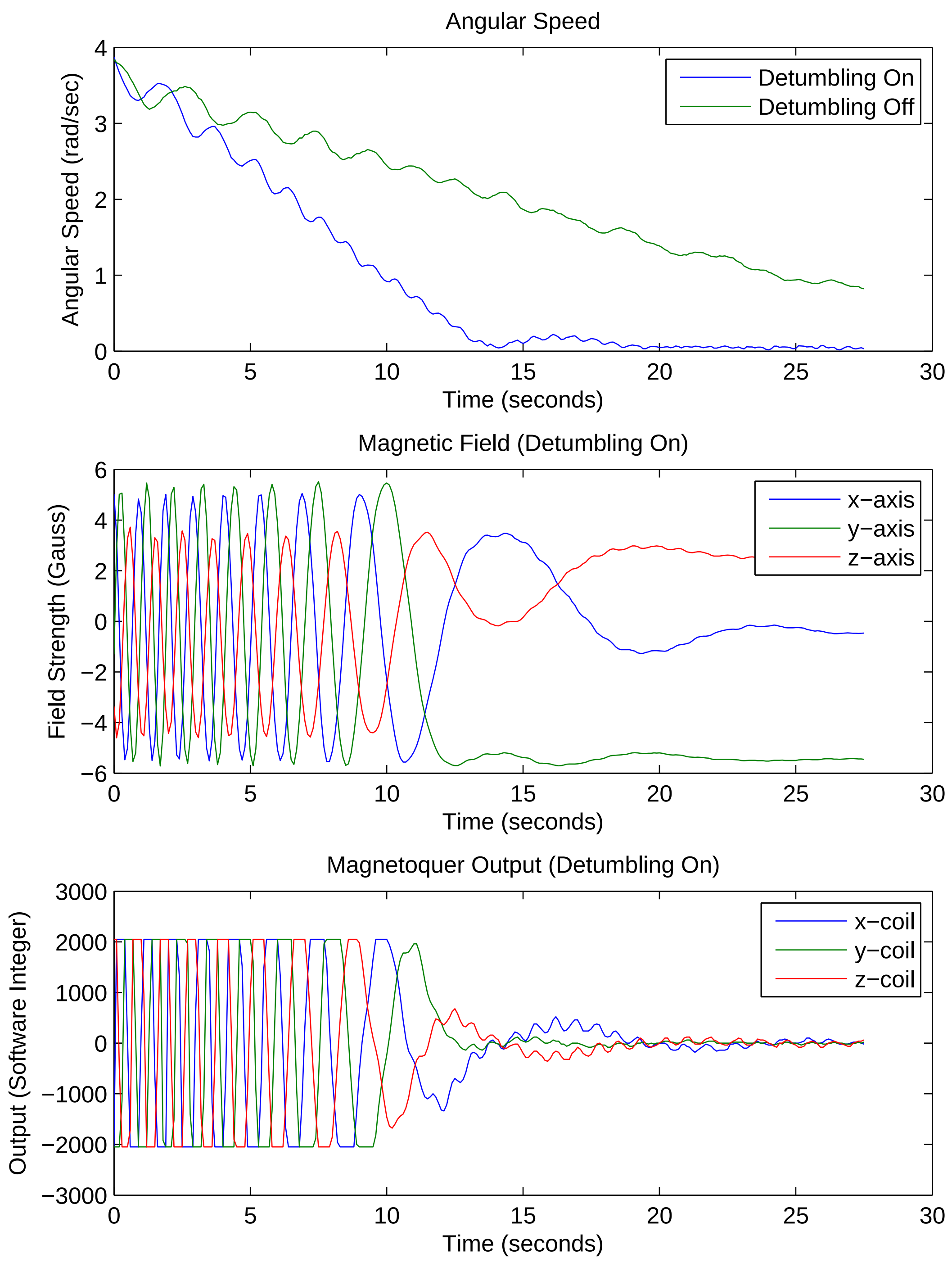


Figure 6.6: Rotation about all axes

Chapter 7

Discussion

7.1 ADCS Module

The ADCS module was designed in the semester project, but some changes were made during this thesis. The H-bridges was replaced by a large type that was easier to solder, and the size of the module itself was changed a bit. The PCB board was manufactured in China while the soldering took place in laboratory of the Department of Electronics and Telecommunications. Once it was finished, the different hardware components had to be tested. At first, all the main components worked except the magnetometer. A lot of time was spent troubleshooting this component, both in software and with an oscilloscope. After several days of work, it seemed that the only possible explanation was bad soldering, even though it had been re-soldered twice already. Luckily, it worked after the third time, and the module was working properly. There are two reasons for why this was problematic. First of all, the magnetometer is very tiny and difficult to work with. Second, the solder mask that is supposed to prevent solder from floating between the pads was not present due to a design fault. If this were to be fixed for the next revision it might be less problematic to do the soldering properly, but it is difficult to know for sure. An alternative is to buy an evaluation board that can be soldered directly to the ADCS module (see figure 7.1).

The sun sensor works, and the output from each photodiode can be measured by the microcontroller's Analog-to-Digital converter. However, the signal gain is much lower than expected. It is obvious that the gain resistors selected in section 3.2 had too low resistance, but it is not so clear why. Either the calculations are wrong, or the test circuit was somehow unlike the final design. Although another op-amp was

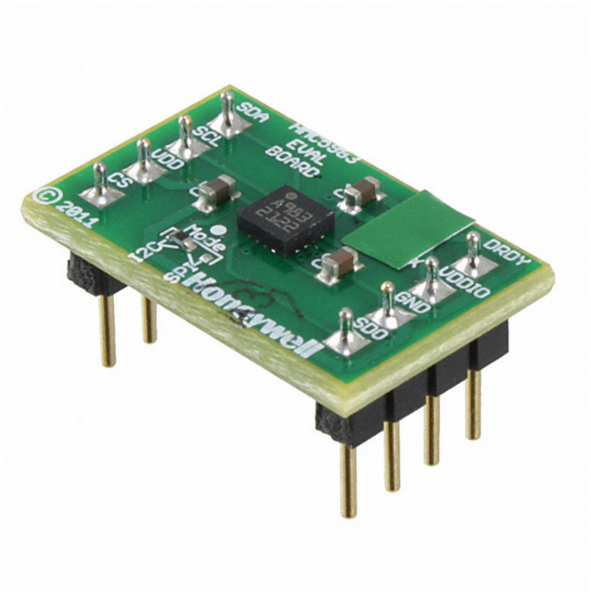


Figure 7.1: HMC5983 Evaluation Kit

used for testing, they should work by the same principles. The issue can be resolved by changing the six gain resistors.

The H-bridges that control the magnetorquers works as expected. They can support a current over 8 times higher than the coils draw. They are therefore expected to stay quite cool during normal operation in space. A fault status pin will go high in case an error should occur. It is connected to the microcontroller but has not been tested because it is difficult to provoke an error without the risk of damaging hardware. There is also an input pin that control sleep mode. It is hard to say how useful this function is. If there is a significant difference in power consumption between sleep mode and idle it might come in handy. However, the system becomes more complex and less error prone with this option compared to heaving the input pin connected to VCC, thus forcing the H-bridge from going to sleep mode. Some data on the current consumption should be acquired before a decision is made. In the current design, the sleep mode pin is connected to a pin on the microcontroller.

The ADCS module works well enough as a prototype because all the crucial components related to control and estimation functions. It does however not quality as a candidate for flight hardware. The I^{2}C interface that is supposed to be present in the backplane connector has been left out by a design fault. It can be quick-fixed by soldering a few wires directly to the module, but this it is generally not considered a practice. Another issue is the connector type used for both the magnetorquer and the photodiodes. Header connectors can easily disconnect during launch because of mechanical stress. Therefor a more secure connector type must be selected for the flight hardware.

The following list sum up issues related to the current ADCS module design.

- Use evaluation board for the magnetometer instead of soldering it directly.
- Change resistors in the sun sensor circuits for higher sensitivity.
- Make a decision whether or not it should be an option to set the H-bridges in sleep mode.
- The I^{2}C interface on the backplane must be connected to the microcontroller.
- The header connectors for the sun sensor and the magnetorquers must be changed to space rated connectors.

7.2 Prototype and Flight Hardware Comparison

The construction of this ADCS prototype was done for several reasons. The most obvious one was that it would make it possible to test the performance of estimation and control algorithms. Now that it is finished, it is clear that it can serve this purpose. The next reason was to get some hands-on experience before building the final flight hardware because practical problems and issues can be difficult to foresee.

The flight hardware will share a number of similarities with the prototype. The ADCS module will not be altered much, except for the changes listed in the previous section. The specifications of the magnetorquers will stay the same, but the thickness of the surrounding frames will be reduced from 3 mm to 0.5 mm. They will be attached outside the carbon frame and not directly to the inner structure. Both the inner structure and the magnetorquer frames will be CNC milled instead of being 3D-printed in order to avoid outgassing from the material.

Small PCBs with a photodiode is attached to every side of the prototype. NUTS will not have an identical setup. It is known that solar panels, and the circuits boards they are attached to, will cover five of the six sides. The photodiodes does not take much space and can be soldered onto the same circuit boards. It does not matter where they are physically placed on the board. The only thing that matters is that the x, y and z pairs are perpendicular to each other, just as they are on the prototype. The sixth photodiode must be placed on the camera side. It is not yet decided how, but the same approach used for the prototype might be feasible. An issue regarding this side of the satellite is the four ”legs” that extend from each corner that can cast shadows over the photodiode. The ”legs” can be seen in the bottom of figure 1.1 in chapter 1. It remains to evaluate the disadvantages of these shadows and if they can be reduced by placing the photodiode intelligently. NUTS will carry a lot more electronics aside from the ADCS. It is uncertain how it will influence the system, especially the analog sensors. This is something that is difficult to test with the prototype. The magnetometer is very sensitive to electromagnetic noise and can be affected by the two radios or by high currents running through wires. However, interferences can go both ways. The magnetorquers can interfere with the radios through electromagnetic noise or through the electrical power system. These issues should be considered and tested in the final NUTS assembly.

7.3 Test Platform

The prototype, the Helmholtz coil and a computer running LabView makes up the Test Platform. It proved to work quite well when the B-dot control algorithm was tested. Still, there are a few issues that must be resolved before it can do more advanced tests, like EQUEST [23] and Non-linear Control [5]. The main problem is how the prototype is mounted in the Helmholtz Coil. The magnetic bearing does not run freely under the heavy weight and ”wants” to rotate to certain angles. Previously when the EiT students created the Helmholtz Coil [3] they used a radial ball bearing. It worked okay for them, but it does not make much sense to use radial when an axial bearing would be much better suited. It seems natural to try and an axial ball bearing for this setup. Another problem with the mounting is the thread. No matter how good the bearing is, there will always be a tiny bit of friction that causes the thread to wind up. The effect of this can be seen in figure 6.5 where the prototype start spinning opposite direction after it has stopped. The thread also causes a lot of wobbling that can be seen throughout the entire detumbling test. This is why it should be removed by extending the metal rod further.

7.4 B-dot Detumbling Test

The tests clearly shows that the detumbling works in all scenarios. In figure 6.3 and 6.4 the rotation stops over twice as fast, 8 seconds with detumbling enabled and 20 seconds with friction alone. In the two other scenarios, it took longer time to stop, but this is most likely because the moment of inertia is higher when the prototype is tilted. The fact that it took longer time for it to stop without detumbling strengthens this claim.

It is important to note that the times-scale of these experiments are shorter than the time-scale will be of the actual detumbling. There are three reasons for this. The first two are related to the angular acceleration that the magnetorquers can create. The prototype has a lower moment of inertia, thus becoming easier to rotate. In addition, the magnetic field induced by the Helmholtz Coil is much larger than the Earth's magnetic field, resulting in a higher torque created by the magnetorquers. From the experiments, the average strength was found to be 6.3 Tesla. In comparison the field strength in a 650 km polar orbit varies from 0.2 to 0.5 Tesla [6]. Let's assume the prototype is 5 times lighter than NUTS. Then the magnetorquers would be 60 to 160 times less efficient in a polar orbit. The final reason is that the Test Platform can only simulate what happens when the rotating axis is perpendicular to the magnetic field. This scenario is very unlikely to happen when NUTS is initially ejected from launch vehicle. Because the torque is given by the cross product between the magnetic moment and the magnetic field vector, it is impossible to stop spin about the magnetic field vector with a set of magnetorquers. As the satellite move along its trajectory the magnetic field will change, and eventually the B-dot controller will manage to stop the all rotation. This is why it can take several orbital periods to complete the detumbling.

Despite the good performance, there is one issue regarding the results. A wobbling with a certain frequency is present in several experiments, and is most prominent in figure 6.3. At first glance, it seems to have a constant amplitude. After a closer look, it seems to decay at a fairly slow rate. The disturbance can also be seen in the magnetic measurements from the experiments where the detumbling was turned off. For that reason, it is fair to assume that the wobbling is not related to the controller. It is never visible in the axis of rotation, which indicates that it propagates through the mounting. The whole setup is very shaky and can easily start to resonate when forces are applied. It is worth mentioning that the magnitude was so small that is was not noticed during the experiments.

Chapter 8

Conclusion

The goal of this thesis was to make a working Attitude Determination and Control System prototype, and to test some of its capabilities. As a continuation of the semester project [22], the thesis stared with the manufacturing of the ADCS Module. Some problems with the magnetometer was encountered along the way, but in the end, the module was working. In section 7.1 a few modifications were suggested for a future design in order to a flight ready module. As for the Sun Sensors, something must have gone wrong when selecting the gain resistors because the output was not nearly as sensitive as expected 3.2. It is recommended to find correct values for these resistors and modify the current module on order to get more accurate sensor readings.

The magnetorquers design was altered a tiny bit before manufacturing in order to make room for more windings. They work as expected, and there is nothing to suggest that they need to be revised before launch. Although, a new set of magnetorquers must be made using a space rated material for the frames.

Then the ADCS Prototype was put together, consisting of the ADCS Module, the three magnetorquers, six photodiodes, a 3D-printed structure and a power supply. In this thesis, it was used to test the implementation of the B-dot control algorithm, and the results show that it works. The Prototype's long-term purpose is so give students a platform for testing software, and more advanced estimation- and control- algorithms.

Chapter 9

Future Work

A lot of work remains before the ADCS is ready for launch. Most of the hardware design is completed, which is why the future work should focus on software and testing. This is now possible with the ADCS Prototype developed in this thesis. NUTS will hopefully be launched in late 2015, which means that there is little time left. The following list contains a set of task that must be completed by then.

- Make a flight-ready iteration of the ADCS Module. See suggested modifications in section 7.1.
- Find a way to estimate the sun vector based on photodiode measurements [19] [20]. Find correct gain resistor (section 3.2) and do testing.
- Implement and test EQUEST using magnetometer, gyroscope and sun sensor data [24] [16] .
- Implement and test Non-Linear control [5]. Improve the Test Platform with a frictionless bearing.
- Define messages and implement CSP (Cubesat Space Protocol) that work over the I^{2}C bus [8].
- Further develop the ADCS interface in LabView. Investigate the possibilities using the software to control the actual satellite.

Bibliography

[1] Alvenes, F. (2012). Satellite Attitude Control System. *Project thesis at the Norwegian University of Science and Technology, Department of Engineering Cybernetics.*

[2] Atmel (2013). 8/16-bit Atmel XMEGA A3U Microcontroller. *Product specifications, Atmel Corporation.*

[3] Bergslid, T. S., Gulbrandsen, F., Hetland, A., and Josephsen, S. (2012). Modell for demonstrasjon av ADCS. *Experts in Team project at the Norwegian University of Science and Technology, Department of Electronics and Telecommunications.*

[4] Birkeland, R. (2011). NUTS-1 Mission Statement. *Technical report, Norwegian University of Science and Technology.*

[5] Bråthen, G. (2013). Design of Attitude Control System of a Double CubeSat. *Master thesis at the Norwegian University of Science and Technology, Department of Engineering Cybernetics*, page 98.

[6] Capo-Lugo, P. A., Rakoczy, J., and Sanders, D. (2014). The b-dot Earth average magnetic field. *Acta Astronautica*, 95:92–100.

[7] Dekker, R. (2000). A Simple Method to Measure Unknown Inductors. *Thecnical report, Ronald's electronic Project Site.*

[8] Gomspace (2011). Network-Layer delivery protocol for CubeSats and embedded systems. *GomSpace ApS.*

[9] Gravdahl, J. T. (2004). Magnetic attitude control for satellites. In *Proceedings of the 43rd IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas*, pages 261–266. Citeseer.

[10] Gravdahl, J. T., Eide, E., Skavhaug, A., Svartveit, K., Fauske, K., and Indergaard, F. M. (2003). Three axis Attitude Determination and Control System for a

picosatellite: Design and implementation. In *Proceedings of the 54th International Astronautical Congress*, volume 7.

[11] Gravdahl, J. T. and Sunde, B. O. (2005). Attitude Determination for the Student Satellite nCubeII: Kalman Filter. *proceedings of the 56th International Astronautical Congress, Fukoka, Japan.*

[12] Hellwig, A. (2000). Helmholtz coil illustration. *Illustration from Wikipedia.*

[13] Holberg, F., Birkeland, R., and Gravdahl, J. (2011). Implementation of three axis attitude determination and control system for a double cubesat. In *Fourth European CubeSat Symposium, Ecole Royale Militaire, Brussels, Belgium*, volume 30.

[14] Holberg, F. S. (2011). Optimal attitude control of a double cubesat using magnetorquers. *Project thesis at the Norwegian University of Science and Technology, Department of Engineering Cybernetics.*

[15] Honsberg, C. and Bowden, S. (2006). Calculation of Solar Insolation. *Technical report, Photovoltaic Education Network.*

[16] Jenssen, K. L. and Yabar, K. H. (2011). Development, Implementation and Testing of Two Attitude Estimation Methods for Cube Satellites. *Master thesis at the Norwegian University of Science and Technology, Department of Engineering Cybernetics.*

[17] Jenssen, K. L., Yabar, K. H., and Gravdahl, J. T. (2011). A comparison of attitude determination methods: theory and experiments. In *proceedings of the 62nd International Astronautical Congress, Cape Town, South Africa*, pages 3–7.

[18] Krogstad, T. R., Gravdahl, J., and Tondel, P. (2005). Explicit model predictive control of a satellite with magnetic torquers. In *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, pages 491–496. IEEE.

[19] Nygren, M. (2012). Using Solar Panels as Sunsensor on NTNU Test Satellite. *Project thesis at the Norwegian University of Science and Technology, Department of Engineering Cybernetics.*

[20] Nygren, M., Birkeland, R., and Gravdahl, J. T. (2013). Using independent combinations of CubeSat solar panels as sun sensors for separating inbound sunlight

information. *2nd IAA Conference on University Satellite Missions and CubeSat Workshop, Rome.*

[21] Overby, E. J. (2004). Attitude control for the Norwegian student satellite nCube. *Norwegian University of Science and Technology, Department of Engineering Cybernetics.*

[22] Rein, O. (2013). Designing a Prototype of an Attitude Determination and Control Subsystem. *Project thesis at the Norwegian University of Science and Technology, Department of Engineering Cybernetics.*

[23] Rinnan, T. B. (2012). Development and Comparison of Estimation Methods for Attitude Determination. *Master thesis at the Norwegian University of Science and Technology, Department of Engineering Cybernetics.*

[24] Rinnan, T. B., Birkeland, R., and Gravdahl, J. (2012). Development on the EQUEST method for attitude determination. In *Fourth European CubeSat Symposium, Ecole Royale Militaire, Brussels, Belgium*, volume 30.

[25] Rohde, J. (2007). Kalman filter for attitude determination of student satellite. *Master thesis at the Norwegian University of Science and Technology, Department of Engineering Cybernetics.*

[26] Tudor, Z. (2011). Design and implementation of attitude control for 3-axes magnetic coil stabilization of a spacecraft. *Master thesis at the Norwegian University of Science and Technology, Department of Engineering Cybernetics.*

Appendix A

Digital Attachments

With this thesis follows a digital attachment, **master_thesis_attachment-oeyvind_rein.zip**. The zipped folder contains the following:

- A video showing the B-dot detumbling algorithm in action
- Source code for the ADCS Prototype. The project file is called **adcs.atsln** and is opened with Atmel Studio 6
- LabView project for interfacing with the ADCS Prototype
- ADCS Module schematics and layout, designed in Altium Designer 13.1, **adcs_module.PrjP** (Available on NUTS's SVN repository)
- Photodiode Mount schematics and Layout, designed in Altium Designer 13.1, **diode_mount.PrjPcb** (Available on NUTS's SVN repository)
- Data from B-dot detumbling experiments
- MATLAB script for calculating magnetorquer properties
- Latex files and figures for this thesis