TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Peeter Org 178231

SOFTWARE FOR TTÜ100 SATELLITE'S ADCS SYSTEM

Master's thesis

Supervisor: Eiko Priidel
MSc

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Peeter Org 178231

TARKVARA TTÜ100 SATELLIIDI ASENDIKONTROLLSÜSTEEMILE

magistritöö

Juhendaja: Eiko Priidel
MSc

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Peeter Org

18/05/2020

Abstract

TTÜ100 is a satellite project created with goal to perform Earth observation using visible light and infrared cameras, test high data rate communications and serve as a technology demonstrator of attitude determination and control, on-board computer and smart power supply [1].

ADCS (Attitude Determination and Control System) is a subsystem of a spacecraft tasked with determining and controlling the attitude (orientation) of the spacecraft. Knowing and being able to control the attitude is useful and necessary for various purposes, covered later in this paper.

The goal of this work is to design and develop software for the ADCS system of TTÜ100 satellite and later update, configure and maintain it.

The goal of this document is to give an overview of the software developed, without disclosing too much about the inner workings to create security concerns.

This thesis is written in English and consists of 47 pages including 7 chapters and 8 figures.

Annotatsioon

Tarkvara TTÜ100 tudengisatelliidi asendikontrollsüsteemile

TTÜ100 on satellidiprojekt, mille eesmärk on vaadelda Maad nähtava valguse ja infrapuna kaameratega, katsetada suure andmeedastuskiirusega sidet ja demonstreerida asendi kontrollimise, pardaarvuti ja intelligentse akusüsteemi tehnoloogiaid [1].

Asendikontrollsüsteem on kosmoseaparaadi alamsüsteem, mille eesmärgiks on määrata ja muuta aparaadi asendit kosmoses. Asendi teadmine ja selle muutmise võimekus on oluline erinavate missiooni eesmärkide täitmiseks, millest räägitakse lähemalt hiljem dokumendis.

Selle töö eesmärk on disainida ja luua tarkvara TTÜ100 satelliidi asendikontrollsüsteemile ning seda hiljem seadistada, uuendada ja hooldada.

Selle dokumendi eesmärk on anda ülevaade arendatud tarkvarast ilma liigselt detailidesse laskumata, et mitte põhjustada turvariske süsteemile.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 47 leheküljel, nende hulgas 7 peatükki, 8 joonist.

List of abbreviations and terms

ACDS	Attitude Determination and Control System
BLDC	Brushless DC motor
CPU	Central Processing Unit
CPU	Central Processing Unit
ECEF	Earth-Centred, Earth-Fixed inertial reference frame
ECI	Earth-cantered Inertial
EEPROM	Electrically Erasable Programmable Read-Only Memory
EKF	Extended Kalman Filter
EPS	Electrical Power System
GPIO	General-Purpose Input-Output
GRV	Gaussian Random Value
I2C	Inter-Integrated Circuit interface
IGRF	International Geomagnetic Reference field
IR	Infrared
kB	Kilobyte
LQR	Linear Quadratic Regulator
MCU	Microcontroller
OBC	On-Board Computer
PC	Personal Computer
PID	Proportional-Integral-Derivative controller
PWM	Pulse Width Modulation
RAM	Random Access memory
RTC	Real-Time Clock
RTOS	Real-Time Operating System
SPI	Serial Peripheral Interface
TalTech	Tallinn University of Technology
UKF	Unscented Kalman Filter

Table of contents

List of figures

1 Introduction

TTU100 is a satellite project funded by Tallinn University of Technology with aims to give students practical engineering experience. Primary mission of the satellite is Earth observation using visible light and near-infrared cameras. For communication with the ground station two radios are used, one in ultra-high frequency range for basic two-way communication and other in X-band range for downloading larger data, like images captured by cameras.

The satellite follows the 1U CubeSat standard, of which most notable is being 10 cm cube in shape and weighing of up to 1.33 kg [1]. Another important property of satellite is that it is uses deployable “wings”, which are essentially two extra solar panels (in addition to those mounted on the sides) that would increase battery charging speed, if properly pointed towards Sun.

1.1 Attitude determination and control system

Attitude determination and control system (ADCS) is a spacecraft’s subsystem tasked with determining and controlling the attitude (orientation) of the craft in space. TTU100 satellite requires an ADCS for various mission requirements, mostly for pointing the cameras and X-band radio towards target positions.

ADCS systems can use various methods for determining attitude, being most commonly vector observations of certain objects, typically Earth’s magnetic field and Sun’s direction.

For attitude control, satellites can use either passive methods, such as gravity booms or permanent magnets, or active methods such as magnetic coils, reaction wheels, thrusters or solar sails.

1.2 Requirements

Requirements of TTÜ100 satellite's ADCS are driven by the X-Band radio and on-board camera.

Primary requirements are [2]:

1. The system must be able to detumble the satellite from spin rates of ±50°/s down to ±0.15° degrees/s (two spins per orbit) within 7 days.
2. The system must have pointing accuracy of 3 degrees
3. The system must implement remotely configurable parameters for attitude control loop a well as readable status for all sensors and control loop states
4. The system must support remote programming
    4.1. Incomplete programming sequence must not cause the system to fail receiving new programming attempt
    4.2. Software part responsible for checking health of the main software image must not be overwritten.

Primary operating modes of the system [2]:

- Detumbling
- Y-Thomson spin
- Tracking geographical point on Earth (for image capture)
- Sun pointing (for maximal power harvesting)

Detumbling is a process of slowing down high rotation speeds after launch, so that normal determination and control algorithms can be switched over to.

Y-Thomson spin is a state where spin axis of the satellite is aligned with axis of the orbit and spinning frequency is aligned with frequency of the orbit, keeping one point of the satellite conveniently always pointed towards Earth [3].

1.3 Goals of the work

As cited from table 19-1 of this preprint of a book section published by NASA [4], design process of an ADCS system should be following:

Step	Inputs	Outputs
1a) Define control modes 1b) Define and derive system-level requirements by control mode	Mission requirements, mission profile, type of insertion for launch vehicle	List of different control modes for during mission. Requirements and constraints
2) Quantify disturbance environment	Spacecraft geometry, orbit, solar/magnetic models, mission profile	Values for torques from external and internal sources
3) Select type of spacecraft control by attitude control mode	Payload, thermal & power needs Orbit, pointing direction Disturbance environment Accuracy requirements	Method for stabilizing & control: three-axis, spinning, gravity gradient, etc.
4) Select and size ADCS hardware	Spacecraft geometry and mass properties, required accuracy, orbit geometry, mission lifetime, space environment, pointing direction, slew rates.	Sensor suite: Earth, Sun Inertial, or other sensing devices. Control actuators: reaction wheels, thrusters, magnetic torquers, etc. Data processing avionics, if any, or processing requirements for others.
5) Define determination and control algorithms	Performance considerations, (stabilization method(s), attitude knowledge & control accuracy, slew rates) balanced against system-level limitations (power and thermal needs, lifetime, jitter, sensitivity, spacecraft processor capability)	Algorithms and parameters for each determination and control mode, logic for changing from one mode to another.
6) Iterate and Document	All of the above	Refined mission and subsystem requirements. More detailed ADCS design. Subsystem and component specifications.

Steps 1-4 of the system's design are largely covered, this work focuses on step 5, which is defining the determination and control algorithms and developing the software. The outputs of this work should therefore be algorithms and parameters for each determination and control mode and logic for changing from one mode to another. The development process is iterative, software is developed and evaluated as a design decision is made and more detailed decisions are made based on existing software.

2 Hardware

For computation and control tasks the system uses STM32F303VE, which is a 32-bit Arm Cortex-M4 based MCU clocked at 72 MHz, having 512 kB of flash storage and 80 kB of RAM. The MCU has a float-point unit, direct memory access capability and a wide variety of peripherals to work with [5] [6].

2.1 Basic control model

The system has following means available for determining attitude [2]:

- Sun sensors to measure direction of the Sun
- Magnetometers to measure direction and strength of Earth's magnetic field
- Gyroscope to measure rotation speed of satellite's body
- IR sensors to measure direction of Earth

and following means to control attitude [2]:

- Magnetic coils (magnetorquers) to create a magnetic field, which will interact with Earth's magnetic field and create a small external force on satellite's body.
- Reaction wheels to that can be spun to create internal force on satellite's body. Those are meant make quick, sharp changes to satellite's attitude.

The system has three of both magnetic coils and wheels, each for one axis.

For communication with external world, the system is connected other subsystems via bus (commonly called "satbus") interface following RS-485 standard. This enables the system to communicate with either the ground station or other subsystems, which are: on-board computer (OBC), communications subsystem (COM) and electrical power system (EPS).

Basic control model is shown in Figure 1.

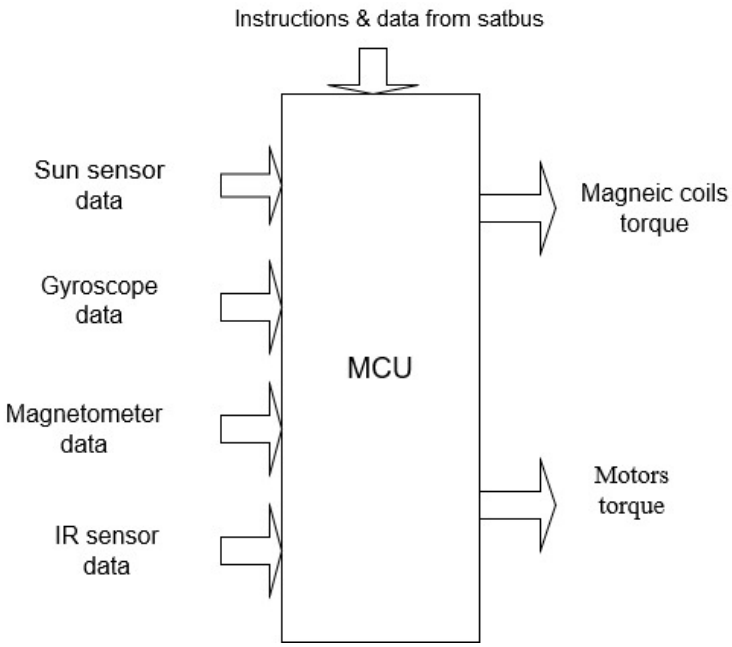

Figure 1. Basic control model

ADCS board is shown in Figure 2 with reaction wheels and connectors for coils and external sensors visible. The MCU and most on-board sensors are located on the other side of the board.

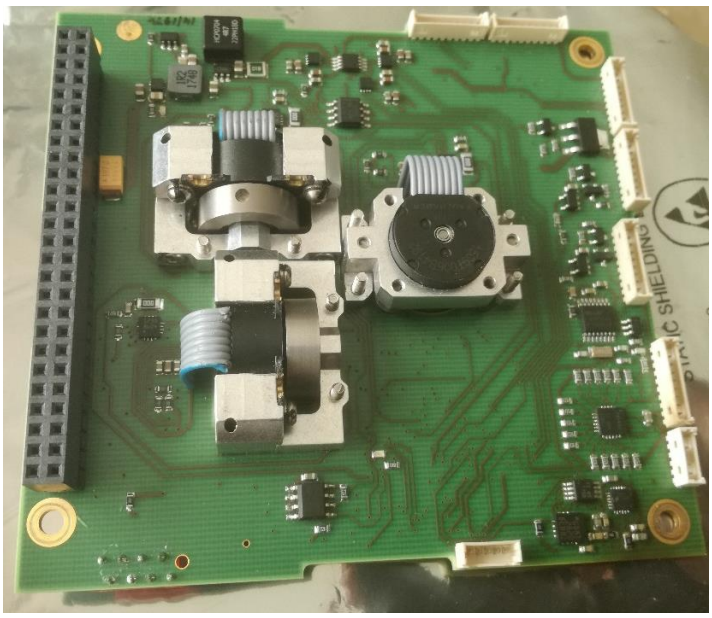

Figure 2. ADCS board

3 Determination software

Determination software is the most complicated part of the system and uses up the largest portion of available computation and storage resources. This section covers selecting software methods for determining the attitude and evaluating performance of the developed software.

3.1 Theory

Several theoretical aspects of attitude determination are explained here.

3.1.1 Representing satellite's attitude

Many means exist for representing the orientation of a rigid body in an external reference frame. One of the simplest and most humanly intuitive is the Euler angles system. Euler angles are represented as three elemental rotations (usually written as α, β and γ) along three axes, which is sufficient for reaching any target reference frame. To perform rotations in Euler angles, rotation matrices are needed. These are:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiplying a 3D vector by $R_x(\alpha)$ would rotate the vector by α degrees in x-axis and so forth. To derive a matrix for general rotations, matrices for all the three axes need to be multiplied together. A rotation matrix representing attitude is commonly called **attitude matrix**.

Using Euler angles has drawbacks: rotation matrices are somewhat memory inefficient, involve expensive trigonometric functions and the angle system has an inherent flaw widely known as “gimbal lock”, where one degree of freedom gets lost when two rotation axes happen to be aligned.

Both the inefficiency and “gimbal lock” problems are successfully addressed by **quaternions**, which represent rotation as follows:

$$q = \left[a^T sin\frac{\varphi}{2}, cos\frac{\varphi}{2}\right]^T$$

Where φ is angle of rotation and {a} is the axis of rotation (3d vector representing Euler axis).

Quaternions can be multiplied using following rule:

$$a \cdot b = \begin{bmatrix} a_4 & a_3 & -a_2 & a_1 \\ -a_3 & a_4 & a_1 & a_2 \\ a_2 & -a_1 & a_4 & a_3 \\ -a_1 & -a_2 & -a_3 & a_4 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

These multiplications represent sequential rotations.

An equally sized rotation in opposite direction is simply:

$$q^{-1} = [q_1\ q_2\ q_3\ -q_4]$$

To rotate a unit vector using quaternion, following rule is applied:

$$p = q^{-1} \cdot v \cdot q$$

Where v is a quaternion with rotation angle 180°, meaning only the Euler axis is represented.

Another important property is that the sign of a quaternion can be changed without changing the meaning:

$$q\ = [q_1\ q_2\ q_3\ q_4]\ =\ [-q_1\ -q_2\ -q_3\ -q_4]\ =\ -q$$

There is a general consensus among the ADCS team members that using Euler angles should be avoided, and quaternions used instead for attitude representation. Author agrees with this.

3.1.2 Wahba's problem

Grace Wahba posed a problem which seeks to find attitude matrix A which minimizes following equation:

$$L(A) = \frac{1}{2}\sum_i a_i |b_i - Ar_i|^2$$

where {b} is a set of vectors in satellite's body reference frame (SBRF) and {r} is a set of vectors in external reference frames, usually in Earth-Centered, Earth-Fixed (ECEF), {a} is a set of non-negative weights and {i} is the number of vectors in both sets. A is the attitude matrix to be found. If input vectors are in ECEF frame, result would be attitude matrix of craft in ECEF frame. The smaller value L(A), the more accurate the determined attitude is [7] [8].

Most of deterministic attitude determination algorithms are based on finding optimal solution to Wahba's problem.

3.2 Determination methods

Attitude determination methods fall into two larger categories: spin-axis methods and three-axis methods. The spin-axis methods determine attitude in form of axis of spin and angle of rotation, while three-axis methods determine attitude in the three degrees of freedom spacecraft has in space.

The list of methods described below is incomplete – many more methods exist and some of the methods described below have their own variations. This list is composed of methods either known to be popular in real use or referenced commonly in studies.

3.2.1 Spin-axis attitude determination methods

These methods are intended for use in spacecraft which are spin-stabilized [9] and have very few and rudimentary sensors available to work with.

Only one sensor is needed for these methods to work, which could be for example: Z-axis magnetometer (Z being spin axis of the craft), Single-Axis sun sensor, Earth Horizon scanner [10]

These methods used to be common in 1970s but receive much less attention nowadays [10]. Being completely outdated, they only deserve a mention in this work and will not be considered for further study or use in this satellite. All of the other methods covered here will fall into category of three-axis attitude determination methods.

Advantages spin-axis methods:

- Requires minimal amount of sensors
- Minimal computing power needed

Disadvantages:

- Requires spacecraft to be spin-stabilized

3.2.2 TRIAD method

TRIAD (Triaxial Attitude Determination) method is the simplest of the three-axis methods, it takes two 3D unit vector measurements produced by on-board sensors and two unit-vector measurements from known associated reference frames. These can, for example, be the Sun and Earth's magnetic field. The reference vectors can be transformed to the corresponding observed vectors using the (unknown) attitude matrix. [9]

The method is based on principle that the desired attitude matrix A can be found from an orthogonal right-handed triad of vectors $\{b_1\ b_2\ b_3\}$ in body frame and $\{r_1\ r_2\ r_3\}$ in reference frame like so [11]:

$$A = [b_1 \vdots b_2 \vdots b_3][r_1 \vdots r_2 \vdots r_3] = r_1 b_1^T + r_2 b_2^T + r_3 b_3^T$$

Two vectors are provided from measurements and models, the third vector can easily be calculated from cross product of the two (normalized) vectors:

$$v_3 = norm(v_1 \times v_2)$$

All three vectors must be orthogonal, which in most real situations is not the case. Therefore, one of the two initial vector pairs needs to be "reconstructed" from v_3 and the other vector pair. This results in two equations for finding the attitude matrix, depending which of the two vector pairs is to be used as the "primary":

$$A_1 = b_1 r_1^T + b_3 r_3^T + (b_1 \times b_3)(r_1 \times r_3)^T$$

$$A_2 = b_2 r_2^T + b_3 r_3^T + (b_1 \times b_3)(r_1 \times r_3)^T$$

Advantages of TRIAD:

- Simple, fast

Disadvantages:

- Only two measurement vectors can be used
- Part of the second measurement is simply ignored [9]
- Measurement (and model) errors need to be ignored

Using only TRIAD would result in poor accuracy and lack of flexibility (only two vectors), in addition to result format not being quaternions.

3.2.3 Singular value decomposition (SVD)

This method involves solving the Wahba's problem using singular value decomposition on matrix B [12]:

$$B = \sum_{i=1}^{n} a_i b_i r_i \tag{1}$$

which is given by:

$$B = USV^T$$

where b an r are body and reference frame matrices, a is the set of non-negative weights.

U and V are orthogonal matrices and

$S = diag(s_1, s_2, s_3)$ with $s_1 \geq s_2 \geq s_3 \geq 0$

$s_{1..3}$ are the singular values of B.

This method is very robust but requires a lot of computations compared to other methods. [12]. Also, output is in undesirable form of attitude matrix.

3.2.4 Davenport's q-method and derivatives

Paul Davenport successfully devised a matrix K, which conveniently converts Wahba's problem to quaternions: [13]:

$$K\overline{q}^{*} = \lambda_{max}\overline{q}^{*}$$

where λ_{max} is the maximal possible characteristic (eigen) value for matrix K and $\bar{q}$ is the unknown attitude quaternion.

$$K = \begin{bmatrix} B + B^T - I * tr[B] & z \\ z^T & tr[B] \end{bmatrix}$$

where 3x3 matrix B is described equation 1.

$$z = \{b_{23} - b_{32}, b_{31} - b_{13}, b_{12} - b_{21}\}^T$$

b_{xy} being members of matrix B.

This equation reduces the problem to finding the largest characteristic value of K, after which deriving the optimal quaternion is a simple process. This method remains the best methods for solving Wahba's problem and very robust algorithms exist for eigen decomposition of matrix K [14]. The original q-method has been superseded by these much faster algorithms listed below.

QUEST (Quaternion Estimator) finds λ_{max} by applying Newton-Raphson method to the characteristic polynomial of matrix K, taking λ_0 as starting value. λ_0 equals the sum of all weights used to calculate matrix B [15] Newton-Raphson method would usually be very inefficient way to calculate λ_{max} , but because is usually λ_0 very close to λ_{max}, the method is able to perform reasonably fast. This is by far the most popular way for satellite attitude determination in spacecraft. [9]

ESOQ (Estimator of the Optimal Quaternion) and **ESOQ2** methods are both devised by Daniele Mortari [16] [17]. These methods calculate λ_{max} identically using the characteristic polynomial of matrix K:

$$\lambda^4 + b\lambda^2 + c\lambda + d = 0$$

$$b = -2(tr[B])^2 + tr[adj(B + B^T)] - z^t z$$

$$c = -tr(adj(K))$$

$$d = det(K)$$

with auxiliary equation and solution:

$$u^3 - bu^2 + 4du - c^2 = 0$$

$$u = 2\sqrt{p}\, cos\left[\frac{1}{3} cos^{-1}\left(\frac{p}{p^{3/2}}\right)\right] + \frac{b}{3}$$

where $p = (b/3)^2 + 4d/3$ and $q = (b/3)^3 - 4db/3 + c^2/2$

And the solution, λ_{max}:

$$\lambda_{max} = \left(\sqrt{u - b} + \sqrt{-u - b - 2\sqrt{u^2 - 4d}}\right)/2$$

Both ESOQ and ESOQ2 evaluate the associated optimal quaternion by computing the maximum modulus vector cross product among four cross product vectors defined in four-dimensional space. To do this, ESOQ implies the computation of seven determinants of 3x3 matrices, while ESOQ2 reduces the quaternion to principal axis and angle, reducing the computation to five determinants of 2x2 matrices [17].

Advantages q-methods:

- Any number of measurement vectors
- Measurement noise is considered (using weights)
- Results in quaternions, which is desirable

Disadvantages:

- More complex than TRIAD

3.3 Filtering methods

The methods listed above are "single frame" methods, which calculate one, deterministic estimation based on one set or frame of measurements, and do not use information about the spacecraft dynamics [11]. To achieve a more reliable reading which also contains angular velocities and maybe angular accelerations, a filtering algorithm needs to be added.

Kalman filter has been the "workhorse" of aerospace for decades and the obvious choice for this task. The filtering method consists of two stages: prediction and update; summarized as follows: [18]

Prediction

$$\hat{x}_k^- = F\hat{x}_{k-i}^+ + Bu_{k-1}$$

$$P_k^- = FP_{k-1}^+F^T + Q$$

Update

$$\tilde{y}_k = z_k - H\hat{x}_k^-$$

$$K_k = P_k^-H^T(R + HP_k^-H^T)^{-1}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k\tilde{y}$$

$$P_k^+ = (I - K_kH)P_k^-$$

where **x** is n-length state vector, P is error covariance matrix, **z** is measurement, u is control input, F is state transition matrix, B is control input matrix, H is measurement input matrix, K is Kalman gain, P is covariance, Q is measurement error, y is measurement residual. Anything marked with a "hat sign" '^' is an estimate of the real system. Superscripts '-' and '+' denote predicted and updated estimates.

Here are the variations of Kalman filters, used commonly in spacecraft [9]:

Linear Kalman filters (LKF) (what is described above) were used for attitude determination in 80s due to limitations on on-board processors. Extended Kalman and Unscented Kalman filters have become more common nowadays, and hardly a satellite exists without one or more on board [9]. Due to non-linear nature of satellite's attitude, using LKF should be avoided to achieve best results.

Extended Kalman filters (EKF) use non-linear transformation f() and h() for state transition instead of state transition matrices F and H.

State prediction and measurement error calculation in EKF goes as follows:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k)$$

$$\tilde{y}_k = z_k - h(\hat{x}_{k-1})$$

Other states are identical to the linear Kalman filter, with state transition matrices F and H are Jacobian matrices defined as follows:

$$F_k = \nabla f_k|_{\hat{x}_{k-1}}$$

$$H_k = \nabla h|_{x_k}$$

Advantages on EKF

- Works very well in case of well-defined state transition models

Disadvantages:

- Computationally heavy
- Difficult to implement due to need to derive the Jacobian matrices
- Limited use cases: transitions need to be differentiable functions
- Known to not perform very well on a wide variety of situations
- Known to be outperformed by UKF in almost every aspect
- Able to obtain only first-order polynomial accuracy at best [19]

Unscented Kalman filter (UKF)was proposed by Jeffrey Uhlmann in 1997 [20]**.** Most inaccuracy problems of EKF are caused by the fact that the gaussian random value (GRV) is propagated through "first-order" linearization of the nonlinear system. The UKF addresses this issue by specifying the GRV through a set of carefully selected points rather than mean and covariance matrix. These points are selected as follows:

$$x_0 = \bar{x}$$

$$x_i = \bar{x} + (\sqrt{(L + \lambda)P_x})_i; \;\; i = 1, \dots, L$$

$$x_i = \bar{x} - \left(\sqrt{(L + \lambda)P_x}\right)_i; \;\; i = L + 1, \dots, 2L$$

$$W_0^{(m)} = \lambda/(L + \lambda)$$

$$W_0^{(c)} = \frac{\lambda}{L + \lambda} + (1 - \alpha^2 + \beta)$$

$$W_i^{(m)} = W_i^{(c)} = \frac{1}{\{2(L + \lambda)\}} \;\; i = 1, \dots, 2L$$

where $\bar{x}$ is mean and P_x covariance of L-dimensional random variable x.

$\lambda = \alpha^2(L + k) - L$ is a scaling parameter.

α determines the spread of sigma points, β is used to incorporate prior knowledge of the distribution of x. $\left(\sqrt{(L + \lambda)P_x}\right)_i$ is the *i*th row of matrix square root.

These sigma points are propagated through the non-linear transform and covariance of the result is approximated from the resulting sigma points [21]. This is called the "unscented transform", and it performs both more accurately and efficiently than the method EKF uses for predicting covariance of the result.

Advantages:

- Able to obtain second-order accuracy
- Rapid implementation, no need to derive Jacobian matrices

- The non-linear function can be approached as ”black box”, meaning any kind of non-linear transition is applicable to UKF
- Both more accurate and efficient than EKF

3.4 Determination algorithm

Since attitude should be represented in quaternion, the q-methods are the most desirable. Since these methods are mathematically equivalent, the decision is reduced to which one is most efficient, which is ESOQ2. [17]

For filtering, LKF should not be used because attitude has non-linear properties. The decision remains between EKF and UKF, out of which UKF is known to perform better in every aspect. The final design choice is to use ESOQ2 in combination with Unscented Kalman filter for attitude determination. Determination flow shown in Figure 3Figure 3.

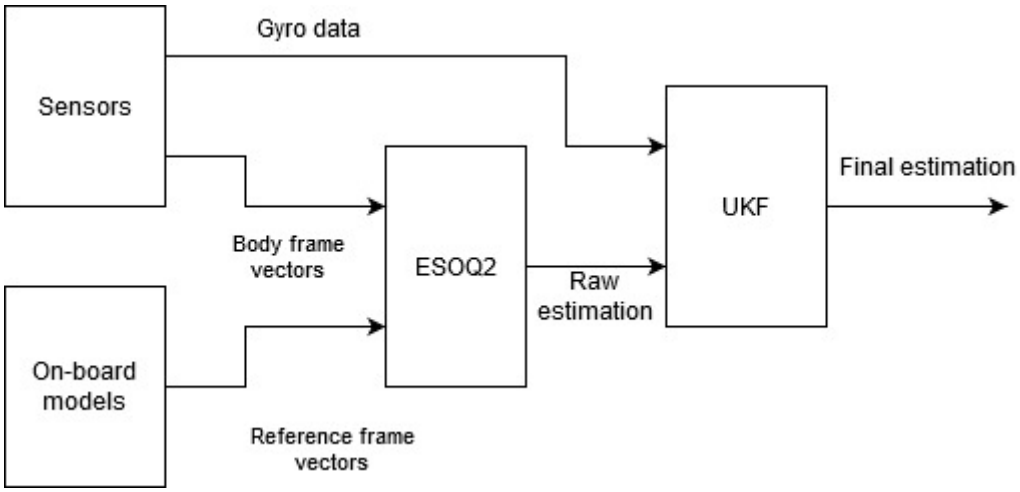

Figure 3. Attitude determination flow

Development of the algorithms was supervised by prof. Alar Leibak of the university.

In most cases where EKF or UKF are used for attitude determination, the measurement and reference vectors are passed directly into the filter and the filter itself performs attitude estimation. Most of them fall into categories of multiplicative or additive Extended Kalman Filters. [14]. In configuration used here, the filter is only tasked with removing noise from estimation of ESOQ2 and estimating the satellite’s other attitude-related states such as angular velocities and accelerations. Author sees several advantages in using this configuration instead. Those include:

- ESOQ2 and other optimal quaternion methods provide an additional output value (loss function) which can indicate the reference vectors estimation is erroneous or there are biases in measurement. (that is, if λ_{max} is significantly different from λ_0).

- Each measurement vector can be assigned their own weights according to their assumed reliabilities and the weights can be changed "on the fly", or even estimated dynamically.

- The measurement vectors and the number of vectors used can be changed without the filter even "knowing" about it (for example Sun vector measurement can be swapped with Earth's direction from IR sensors as the craft enters eclipse).

- This configuration has no "first estimation" problem with the filter – the filter can be initialized with the first result from ESOQ2 which should be accurate enough.

- Modular structure means easier parametrization and separate components can be reused in other missions. For example, the filter could be used for a mission that utilizes a star tracker that already outputs the measurement in quaternion form.

As mentioned, the UKF will estimate only attitude-related states. Estimation of non-attitude states will be done elsewhere, if needed. For example, sensor biases will be estimated within separate Kalman filters tasked with filtering measurements. Satellite body's moments of inertia will not be estimated at all, as the control algorithm does not use this information as input.

Numerical simulations of the ESOQ2 method show that correlation between measurement noise and estimation error is linear. In test result shown in Figure 4, two vector pairs were used and the angle between the reference vectors was 60 degrees. The figure shows only unfiltered, raw data.

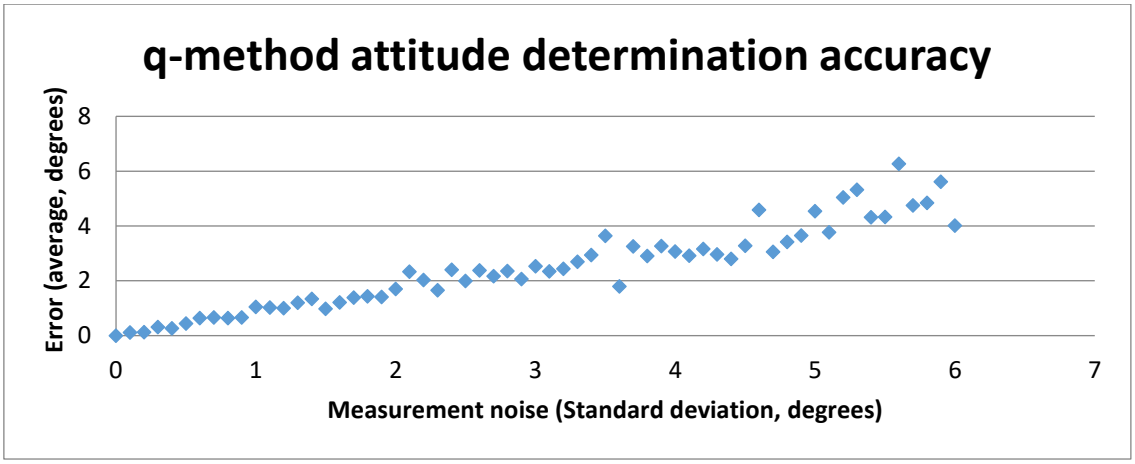

Figure 4. Attitude error measurement noise dependency

When two of the vector pairs used for attitude calculation become parallel or antiparallel, calculating attitude in the three degrees of freedom would become mathematically impossible. Therefore, it is expected that there will be increase in estimation error as measurement vectors come close to this state. Result of numerical simulation with different angles between measurement vectors is shown in Figure 5. In this simulation, noise standard deviation of 3 degrees was used.

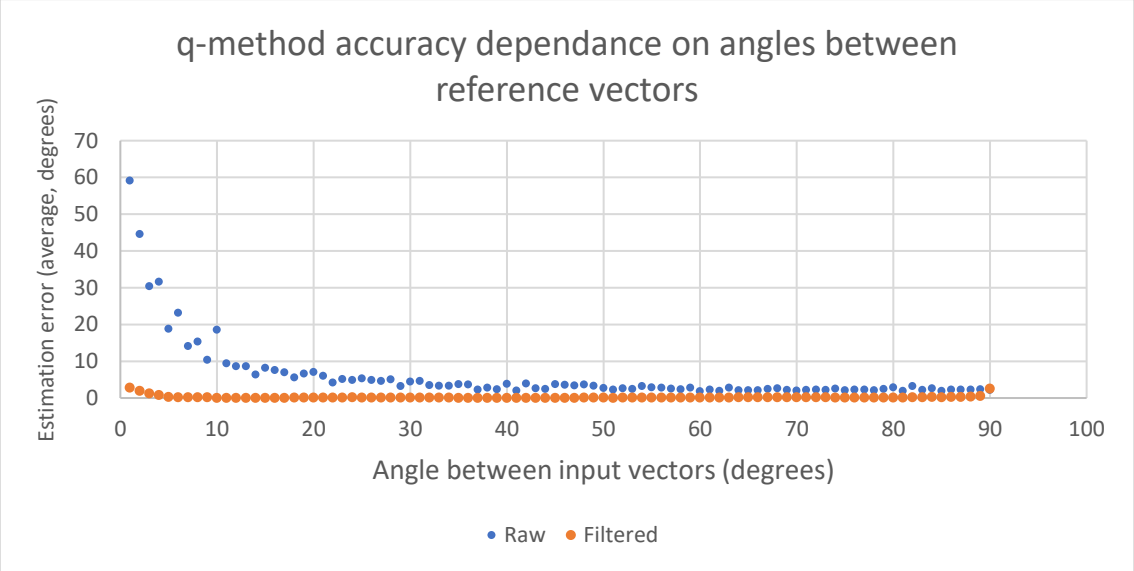

Figure 5. Attitude error measurement vector angle dependency

This simulation would represent final accuracy of the ADCS estimation (with static attitude) at 3-degree standard deviation measurement noise if the reference models were 100% accurate (which is a very unlikely ideal case scenario). The 3-degree standard deviation of input error was chosen intuitively as the “worst case” in a real situation. As

seen from the graph, even with average error of 60 degrees, a Kalman filter is able to reduce the error down to around 3 degrees. The sensors will have their own Kalman filters to reduce noise before passing data to determination algorithm, meaning with similar filtering performance assumed, at 3 degrees of standard deviation measurement error, the raw measurement error would average around 60 degrees. Noise statistics of the real sensors have not been gathered, but graphical visualization of sensor data has shown the error to be clearly smaller than that.

Those simulations were run with static attitude. Additional work is required to incorporate state estimation into the filtering in order to run simulations with a spinning satellite. Those added complexities will inevitably decrease the final estimation accuracy, but even if final estimation error would increase by 3 times, the performance of the algorithm would still be adequate, provided the angle between the measurement vectors remained larger than 4 degrees (and smaller than 176).

Based on results from numerical simulations, it is quite safe to assume that required accuracy can be reached with the current configuration, given the accuracy of the models and sensors meets expectations. Finding this out requires actual telemetry returned from the satellite in space.

3.5 Reference models

To provide the reference vectors for the ESOQ2 algorithm, several on-board models are needed. Most of the models require satellite's position relative to Earth as input. To propagate satellite's position, several algorithms are available, most common of them is SGP4 [22]. This model (like most others available) requires knowing the time (provided by on-board RTC) and orbit model, which is provided by NORAD in the form of two-line element set and uploaded to the satellite, to be stored the system's EEPROM. This will need to be updated once every couple of days. SGP4 model was evaluated as efficient and accurate enough by bachelor student Mohammad Tavassolian. Author has run his own tests after integrating the algorithm in into the ADCS software and can verify that the algorithm is accurate enough and efficient enough to fulfil mission requirements.

To provide reference vector of magnetic field, IGRF (International Geomagnetic Reference Field) model is needed. This was developed by bachelor student Shu Taya and

prof. Alar Leibak. The model uses Maxwell's equation and Legendre functions as mathematical basis. Coefficients provided by International Association of Geomagnetism and Aeronomy are needed by this model and updated as new ones are published, which happens roughly every five years [23].

Reference vector for Sun can easily be calculated from time provided by real-time clock (RTC). This model was developed by Hendrig Sellik as part of his bachelor's work [24].

Earth's direction vector can be derived from satellite's position in orbit.

4 Control software

Control software is tasked with implementing the operating modes defined in requirements. At least three different control modes are needed.

4.1 Control loop options

Since attitude of the craft is known, it is obvious that attitude control should be done in closed loop. A few options to achieve closed loop control listed below.

4.1.1 State feedback loops

State feedback loops use the following system model

$$\dot{x}(t) = Ax(t) + Bu(t)$$

Where x(t) is a vector of system states and u(t) is controller input vector. Idea is to select matrix K:

$$u = -Kx$$

which fulfils the control requirements in a certain way. Many different control loops can be created using this model, example system state being fed back, system output integral being fed back, or system state or system error being fed back. There are several ways to choose the control matrix k.

In **pole placement feedback**, matrix K is calculated so that closed-loop poles of the system (eigenvalues of state transition matrix A) are placed in selected locations. [25]. Selecting the poles to achieve best results, however, can be a counter intuitive process.

Linear Quadratic Regulator (LQR) calculates matrix K using specially selected weights Q and R, so that matrix K would minimize the equation [26]:

$$J = \frac{1}{2}\int_0^\infty (x^t Q x + u^t R u)dt$$

Q is a set of weights that penalises state performance of the system over time (this could be attitude error for example) and R is a set of weights which penalises control input (this could be motor velocities or coil torque). Higher weights for Q would mean the system is more eager to use control inputs to minimize the time to reach target (in our case, attitude). Higher weights for R mean the system would try to optimize usage of control inputs (in our case, coil and motor currents) to conserve power.

4.1.2 PID and variations

PID which stands for proportional-integral-derivative controller, is the single most used dynamic control technique, being used in 85% of all dynamic controllers. [27]

As input, PID takes target and actual system state and calculates the error (which would be zero if target is reached), or simply takes the error as input.

The output of PID controller is sum of three terms: proportional, integral and derivative, hence the name PID. Proportional term is weighted size of the error, derivative term is weighted change in error and integral term is weighted sum of the error over time. Some of the terms can be left out if not needed, which would make it a variation of PID, for example PI if derivative term is left out [27].

The controller can be combined with open-loop components called “feed-forward” terms, which are functions of the target state instead of state error. These are useful when the target state needs output value to be held.

As seen, PID is an extremely simple control technique requiring little knowledge of systems or control theory to implement. Tuning the controller (choosing the constants for the terms) is also simple and intuitive, and the controller is known to reach the target even when imperfectly tuned (unless it is tuned to be completely unstable).

Despite knowing that LQR can give the best possible result (if state models are perfectly accurate), the simplicity of both implementing and configuring PID is impossible overlook. PID parameters would also be easier to “tweak” for more optimized control while the craft is in space. Also, at this rather late stage of software development, when delivery of working software is required soon, no usable models of spacecraft dynamics have been created. Creating those models would steal away time from developing other

important features that need to be added to the system. For those reasons, PID will be the choice for all control-loops used in the system.

4.2 3-axis pointing

For pointing tasks, only the motors will be used, because magnetorquers would perform hopelessly slow compared to motors to make an impact and interfere with magnetic field, making attitude determination difficult. Two separate PID controllers will be needed, one for calculating target velocity of the satellite, taking target and current attitude as input, and other for calculating target velocities for the motors, based on target angular velocity of the spacecraft. In most cases the target attitude also changes in time (because the satellite is moving in orbit relative to point of interest), meaning speed of the target attitude change also needs to be calculated and accounted for. This will simply be added to the final target velocity. The pointing control loop shown in Figure 6.

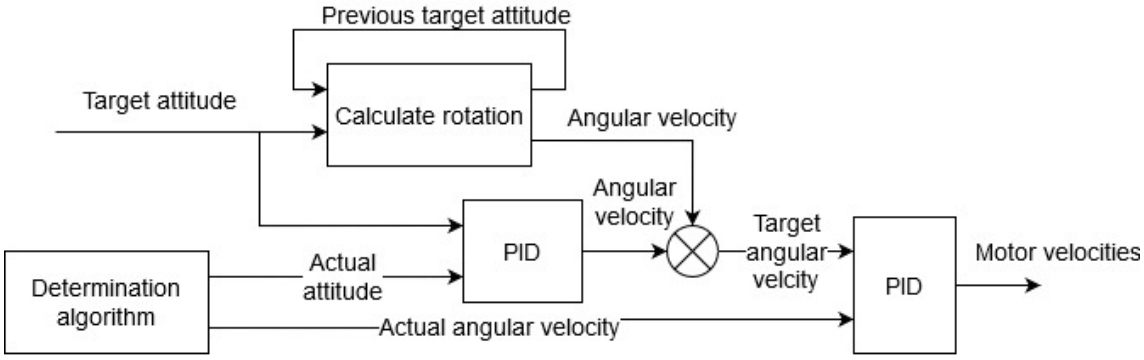

Figure 6. 3-axis pointing flow

For tasks where target attitude will not be changing significantly (like Sun pointing), separate control algorithm will not be implemented. Author does not see the benefit in that since the one shown in Figure 6 will perform in those cases as well, and the overhead of calculating angular velocity based on attitude change is minimal. All pointing tasks will use the same control loop.

4.3 Y-Thomson spin

Y-Thomson spin is a convenient state when images need to be captured of Earth, but rather inconvenient when attempting to point solar panels towards Sun for charging batteries. For the latter, it would be much better to have the satellite attitude in a complete

standstill relative to ECEF. This means that depending on situation, these modes should be switchable. Motors cannot exert external force on satellite's body, meaning constant spin rates would need to be maintained if using motors for switching between those states. This is unwanted because motors should be used sparingly, to reduce wear and minimize power consumption. Therefore, only magnetorquers must be used for switching between Y-Thomson spin and standstill.

A spin rate controller will be implemented for this purpose, which will take magnetic field, target and actual spin rates as inputs generate for magnetorquers accordingly. Only one PID will be needed for this, which is likely of P variation (without derivative and integral terms), because with only magnetorquers used, the change created in attitude states would take significant amount of time (measured in hours rather than minutes or seconds), meaning it is unlikely that derivative and integral terms would make anything useful out of it.

This loop cannot be run continuously, because magnetic fields created by coils interfere with magnetometer reading, meaning torque needs to be paused for taking new measurements of magnetic field and estimating new attitude.

External disturbances on satellite's body will also be countered by occasionally running this control loop.

4.4 Detumbling

After launch, the satellite may be in a fast spin, which makes communication with ground station and attitude determination-control either difficult or impossible. To establish proper communication and start using conventional determination and control algorithms, spin rates need to be significantly reduced. Since motors cannot change the total velocity moments of the satellite's body, detumbling needs to be done using only magnetorquers. To achieve this, the B-Dot control law [28] is used, which tries to minimise change in magnetic field. A modified/simplified B-Dot controller is implemented as follows:

$$B = m_k \times m_{k-1}$$

$$v = norm(B \times m_k)$$

$$g = 1 - e^{-N*|B|}$$

$$T = \upsilon * T_{max} * g$$

T is the output of coils (amount of current passing through) which is a 3D vector as there are three coils. B is change in magnetic field, m_k and m_{k-1} are current and previous normalized magnetic field measurements, υ is the unit vector magnetic field axis to be created by coils, T_{max} is the scalar value representing maximal output of coils and g is the B-Dot gain. $N \geq 1$ is gain constant. The larger is N, the more torque is applied relative to rotation speed.

This loop is run at around 4 Hz, with most of the time spent applying torque. Should the frequency be adjusted, gain constant N should be adjusted accordingly. A tiny delay is needed between releasing the torque and taking a new measurement, to let current in coils and magnetic field around spacecraft to cool off. For power safety, this loop will always be run for a specified amount of iterations and never in an endless loop.

5 Software design

All software for the MCU is written in C99, compiled with GNU Arm Embedded Toolchain using custom makefiles and linker scripts. Service tool for updating software, data exchange, diagnostics and other utilities is written in Python.

To help in handling concurrency, timings and provide some other utilities, a real-time operating system (RTOS) called FreeRTOS is used in the main binary. It is included in the project as source code and together with rest of the code. Firmware update portion runs independently without an operating system.

5.1 Abstract view

The code is strictly divided into three major components, with files in separate directories: HW (hardware), SYS (system) and LOGIC.

LOGIC portion contains the navigation software and other utilities which have only numerical inputs and outputs. This part of the code has no external dependencies , meaning this can be compiled with only the files within the directory and can therefore be included into any other project with minimal effort, and is written along with unit tests and simulations for running on a PC.

HW contains hardware drivers for the system. Dependency on RTOS unwanted, but in many cases unavoidable because some processes have delays and CPU time should be yielded to let other processes use it. Without dependency on RTOS, any smaller firmware binary could be compiled without any operating system included, which could be useful for several utilities.

SYS is the intermediate layer between hardware and logic. Assignments of this part include moving around data within the system, defining and handling communication with other systems, defining priorities of processes, managing system states and configurations, monitoring system health and everything else not directly related

hardware control or navigation. This component has dependencies on every other component of the system, as this is what that binds the system together.

The components and their dependencies are shown in Figure 7.

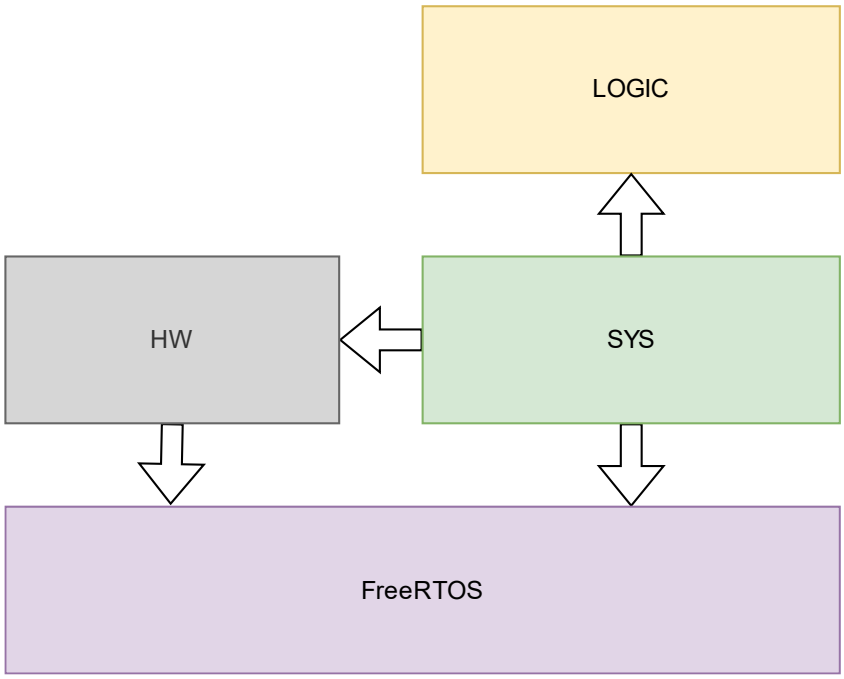

Figure 7. Software abstract view

5.2 Running tasks

A FreeRTOS task is a concept similar to threads in most other operating systems. A task runs in its own stack of specified size, keeps its own program counter and other CPU-related context has an assigned priority.

The ADCS system is divided into six tasks: Environmental Modelling, Data Acquisition, Control, Attitude Determination, Main Loop and Bus Task.

5.2.1 Environmental Modelling task

This task contains the models to provide reference vectors for ESOQ2. These models are SGP4 model, Sun vector model and IGRF model. The task is woken up by an RTC alarm that indicates that model data has been outdated.

Models would normally be updated every second (because RTC precision is one second), but to reduce context switches within the system, this task is woken up once every 5

seconds, then it calculates the new model data five seconds ahead and stores the results in an array. Other tasks can then access the data in the array instantly, reading from array index based on the time since last update.

5.2.2 Data Acquisition task

This task will read each sensor specified as ‘active’, filter the result and copy the data to specified memory location for other tasks to access instantly when needed. Measurement data logging capability is also contained within this task.

This task can be instructed to either go inactive, update measurement data in normal frequency or update measurement data in rapid frequency. This task would measure rapidly during pauses in coil torque when switching between standstill and Y-Thomson spin. During detumbling, this task would be inactive.

5.2.3 Control task

This task contains the control software and executes other commands which take too long for the bus task to carry out or commands which need to be executed at a specific time. The commands are kept in a queue and executed in order. Some of those include:

- Detumbling for a given amount of iterations
- Pointing towards desired coordinates for a specified time
- Carrying out software updates and resets
- Waiting for a specified RTC (Real-Time Clock) alarm

This task’s ability to wait for a specified time in RTC and hold a sequence of commands in a queue gives the system some useful mission-planning capabilities. Example use cases include:

- The system will wait for a time specified the satellite reaches poles, where magnetic field is strongest, then start the detumbling algorithm for best results relative to power usage.
- The system will wait for a specified time until a location of interest is reached, then turn the cameras towards that location for image to be captured.

- The system will wait until eclipse (being in Earth's shadow) passes, then turn solar panels towards Sun.

- The system waits for a specified time of interest and then tells data acquisition task to start logging sensor data, for various scientific or diagnostic purposes.

- Perform several mission objectives (for example the ones listed above) and then perform software update, loading new firmware from external EEPROM into flash.

there are many more possibilities so this capability means that ADCS can act as a secondary OBC if needed.

The RTC allows alarms to be set up to 1 month ahead (though this long will probably never be necessary), with a one second accuracy [5].

To clear any unwanted instructions that are being executed or waiting in the queue, the task needs to be deleted along with the queue and restarted.

5.2.4 Determination task

This task will contain the determination, and will run in fast loop, continuously calculating and updating the attitude estimation for other tasks and subsystems to use. During detumbling process, this task would be inactive. When currents are passing through coils, the attitude would be updated with only gyro data.

5.2.5 Bus task

This task is responsible for monitoring the bus and transmitting/receiving messages. Parsing of each command sent to the system also happens in this task, making this the biggest task in the system in terms of code size. Most of the commands are also executed within this task, but only utility commands which can be carried out in an instant. Commands that take time to be executed, such as attitude control commands, cannot be executed in this task because rest of the communication would be halted. These commands are passed over to the control task.

The bus protocol itself was developed by Madis Kaal and the exact same code runs on every other subsystem, except for EPS, which contains only code written by Veljo Sinivee.

5.2.6 Main loop task

This task will initialize resources, create other tasks in a specified sequence, and after that it will simply keep looping, pinging each task after a specified delay and update the external watchdog. If any of the tasks stops responding to pings, the main loop would halt and the external watchdog will starve, generating a power reset to the system. The only task not monitored is the control task, doing which would be unreasonably difficult with the wide array of different actions with different durations and timings it is performing.

5.3 Bootloader

Software update is the single most mission critical capability of the system. Provided radio communication can be established with the satellite, everything other than update capability in the ADCS software may fail and the mission could be continued. If software update fails and the system gets "bricked", the primary mission objectives cannot be pursued.

Update is performed using a bootloader, which stored in beginning of the internal flash memory. Bootloader will start on power-up, read the reset code from external EEPROM, and based on the reset code will either wait for instructions from the bus or attempt to start the main firmware after checking health of the image. XTEA hash algorithm is used for firmware verification purpose. The hash table along with other information about firmware is located at the end of flash. Flash memory usage is shown in Figure 8.

Figure 8. Flash usage

Bootloader "mode" is also the "safe mode" of the system because nothing is being done except monitoring the bus while the system is in bootloader. This means that any preventable harm will be prevented as long system stays in the bootloader.

The bootloader also has mechanism to prevent a situation where firmware hash is valid, but for some reason does not work properly (either falls into one of fault interrupt handlers or does not update the external watchdog). Normally in this situation the system would keep resetting and trying to start endlessly. This could be caused by hardware degradation caused by radiation. The bootloader keeps a counter on external EEPROM, which is read and incremented every time bootloader is started. If the counter reaches 10, bootloader will refuse to call firmware again and wait for instructions from bus. Once the software has started, the main loop task will reset the counter after three iterations. Because the main loop task pings each other task within an iteration each task will have responded to pings at least three times and the software can be considered stable. This mechanism also helps recover from a faulty firmware which cannot communicate on the bus for some reason, but continues to update the external watchdog so that automatic power reset would not happen and there would be no way to instruct the system to fall back into bootloader. To recover from this, the EPS can be instructed to perform power rest to ADCS system 10 times in a row with a small delay. In this case, the reset counter would reach 10 and system would stay in bootloader and communication (if hardware enables it) could be established.

Throughout most of software development, new versions have been uploaded to the system using the bootloader rather than hardware debugging tools. At the time of writing this thesis, the software development of this system has been in "active phase" for roughly 1,5 years, during which firmware update has been performed for several thousand times, without a single "soft brick" happening. Up in space, firmware update will be performed for maybe 10-15 times (if the system satellite survives long enough to see this happen), meaning the likelihood of an undiscovered software bug surfacing during firmware update in space is extremely low. Based on this, the software update capability is considered 100% stable.

6 Hardware drivers

Hardware drivers are developed using firmware library provided by ST, with some exceptions where MCU registers being accessed directly, when performance is critical or use case is not covered firmware library.

6.1 Motors drivers

The three reactions wheels are driven by BLDC motors each having their own internal Hall effect sensor. The drivers are developed “by the book” as instructed in section 20.3.24 in ST’s reference manual [15]. To capture inputs from the Hall sensors, timers in the STM32 have special mode where Hall sensor inputs are XOR-ed together to detect change in any of the three inputs. This XOR-ed signal is internally connected to another timer responsible for creating the PWM for motor drivers. This timer configuration referred to as master-slave mode in ST manual. Change in signal generates commutation interrupt for the timer PWM, which triggers loading of PWM output values from a preloaded register to output register. For the next commutation event, new values are written in the preloaded registers within the timer’s commutation interrupt handler. Using preloaded values and hardware switching means no delay between Hall sensor update loading new values, meaning maximal efficiency.

The drivers will take target velocity as input and a PID controller will calculate PWM output value based on target and actual motor velocity. The velocity is measured using the capturing timer and PWN values are updated within the same timer’s interrupt handler.

6.2 Magnetorquer drivers

Coil drivers are driven by hardware timers that generate PWM. Direction selected using an H-bridge driven by a GPIO. Pins are connected to MCU so that each coil has their own timer associated with them. This design decision was short-sighted, because it would have

been possible to control all of the coils with just one timer. This would have left the two other timers free to do other useful things.

6.3 Simple sensors

Most of the sensors for measurement are communicated with using simple MCU peripherals such as I2C and SPI.

The system has 6 magnetometers available for use, two outside the board on deployable wings and four on the system's board. The large number of sensors is due need to test performance of several different magnetometers in space. Need to add additional magnetometers became evident when data visualization showed that magnetometers from different manufacturers can perform very differently. Possible effects on magnetic field measurements has also been an important factor to consider in design of satellite's mechanical parts and other subsystems. Any effect motors (mounted directly on the system's board) can have on magnetic field measurement is also unknown. Therefore, telemetry returned from the magnetometers will be an important scientific outcome of the mission and can form a basis for design of a possible next mission. The magnetometers mounted on deployable wings of the craft will provide a useful comparison to those mounted on the system's board, because distortions to the magnetic field will be significantly smaller outside on the wings.

There are 8 ambient solar sensors and 8 IR sensors, one set for each side and two more for the deployable wings. These sensors are communicated with using I2C peripheral and a I2C multiplexer to handle problems with address conflicts.

6.4 Solar cameras

Ambient solar sensors have an inherent problem – accuracy becomes low when falling light is close to 90°, because small changes around this angle would not result in different amounts of light captured, causing almost no change in reading. To make up for this, six 18x18 pixel black-white cameras are installed to determine Sun's direction more accurately in those areas. These cameras were originally intended for use as mouse sensors, but also have a mode to capture raw pixel data which is being used. To read the data, a serial interface is implemented by bit banging GPIOs. Clock signal is shared

between all the cameras and for data each has separate GPIOs assigned to them. This means that read operations will not be performed separately for individual cameras, all of the images from six cameras will be read at once. The read time unfortunately is slow, taking around 200-300ms which may slow down the determination loop, but in turn this would give the most reliable measurement vector available hardware can provide.

6.5 Additional hardware

There are several other hardware features included in the system.

HW watchdog is included for additional reliability. This is a device which has a single GPIO as input and the value needs to be toggled updated every 1.6 seconds. If this does not happen, the watchdog will switch off the system's power supply for a brief moment. This is being updated in the main loop task if all other tasks respond to pings.

LF receiver – a low frequency wake-up receiver is connected to the MCU. This has two inputs, one connected to an unused magnetic coil and the other connected to one of the solar panels. The unused magnetic coil can be used for a low frequency radio receiver and the solar panel can be used for testing visual communication with the satellite. If enough light shines on the solar panel, a voltage change will happen. This way a high-power laser can be used to send messages to the satellite in space. The unused magnetic coil is also connected directly to an analog pin of the MCU, to listen to the input directly.

A **high-power LED** is also connected to the system, this can be used to test visual communication with ground station.

The last two hardware features are not related to main goal of the system and serve a scientific purpose. These were added because there were a few more free pins on the MCU.

7 Summary

At the time of writing this thesis, ACDS software is in following state: firmware update is working, tested and stable, hardware driver development is complete, system architecture is largely in place, reference models for the attitude determination are integrated and most of the determination algorithm is complete and soon will be ready to be integrated.

What still needs to be done: the UKF of determination algorithm needs more work (adding proper state prediction), one-dimensional Kalman filters currently used for sensor filtering should be replaced with proper filters, preferably UKF with dual estimation and bias estimation capability. Control software design is in place but the software itself is largely missing. Accuracy of the reference models needs more verification.

The satellite itself has been handed over to launch provider and will launch in a couple of months probably. Software currently on board the satellite to be launched is rudimentary – it has the bootloader for updating the software, the B-Dot algorithm to detumble the satellite, and ability to return telemetry of all on-board sensors. Calibrating and tuning of the sensors will likely take a couple of more months of gathering telemetry after launch and hopefully by then a feature complete binary will be ready for uploading to begin fulfilling mission objectives.

References

[1] The CubeSat Program, Cal Poly, *CubeSat Design Specification Rev.13,* 2014.

[2] TUT MEKTORY SPACE CENTRE, “System Requirements Specification TMSS-SYS-RS-01,” Tallinn University of Technology, Tallinn, December 2016.

[3] B. W. Young, “Design and Specification of an Attitude Control System for the DANDE Mission,” Massachusetts Institute of Technology, 2006.

[4] S. R. Starin and J. Eterno, “19.1 Attitude Determination and Control Systems (Preprint),” Microcosm Astronautics Books , 2011.

[5] ST Microelectronics, “RM0316 Reference Manual,” [Online]. Available: https://www.st.com/resource/en/reference_manual/DM00043574.pdf.

[6] STMicroelectronics, *STM32F303xD STM32F303xE,* 2016.

[7] G. Wahba, “A Least Squares Estimate of Satellite Attitude,” *SIAM Review,* vol. 8, no. 3, 1966.

[8] D. Motrari and M. F. Landis, “How to Estimate Attitude from Vector Observations,” Astrodynamics Specialist Conference, Girdwood,Alaska, 1999.

[9] J. C. V. d. Ha, “Progress in Satellite Attitude Determination and Control,” *Transactions of the Japan Society for Aeronautical and Space Sciences, Space Technology Japan,* 2009.

[10] S. Tanygin and M. D. Shuster, “Spin-Axis Attitude Estimation,” *The Journal of the Astronautical Sciences,* vol. 55, 2007.

[11] F. L. Markley and D. Mortari, “QUATERNION ATTITUDE ESTIMATION USING VECTOR OBSERVATIONS,” *Journal of the Astronautical Sciences,* 2000.

[12] L. Markley, “Attitude Determination Using Vector Observations and the Singular Value Decomposition,” *Journal of the Astronautical Sciences,* vol. 38, no. 3, 1987.

[13] Computer sciences corporation, “ANALYSIS OF LEAST-SQUARES ATTITUDE DETERMINATION ROUTINE DOAOP,” 1977.

[14] L. F. Markley and J. L. Crassidis, Fundamentals of Spacecraft Attitude Determination and Control, New York: Springer Science+Business Media, 2014.

[15] M. D. Shuster, “The Quest for Better Attitudes,” *The Journal of the Astronautical Sciences,* vol. 54, no. 3&4, 2006.

[16] D. Mortari, “ESOQ: A closed-form solution to the Wahba problem,” Mortari, Daniele, 1997.

[17] D. Mortari, “ESOQ-2 Single-Point Algorithm for Fast Optimal Spacecraft Attitude Determination,” 1997.

[18] Y. Kim and H. Bang, “Introduction to Kalman Filter and Its Applications,” in *Kalman FIlter*, InTechOpen, 2018.

[19] M. C. VanDyke, J. Schwartz and C. Hall, “UNSCENTED KALMAN FILTERING FOR SPACECRAFTATTITUDE STATE AND PARAMETER ESTIMATION,” *Advances in the Astronautical Sciences,* 2004.

[20] J. K. Uhlmann and S. J. Julier, “A New Extension of the Kalman Filter to NonlinearSystems,” in *Proceedings of the SPIE AeroSense International Symposium on Aerospace/Defense Sensing,Simulation and Controls*, Orlando, Florida, 1997.

[21] E. A. Wan and R. van der Merwe, “The Unscented Kalman Filter for Nonlinear Estimation,” Oregon Graduate Institute of Science & Technology, Oregon, 2000.

[22] F. R. Hoots, R. L. Roehrich and T. S. Kelso, “Models for Propagation of NORAD Element Sets,” spacetrack, 1988.

[23] GEOPHYSICS AND GEOCHEMISTRY – Volume II p. 188, Oxford, UK: EOLSS Publications, 2009.

[24] H. Sellik, *KUUPSATELLIIDI ASENDIMÄÄRAMINE PÄIKESE-JA MAGNETVEKTORIGA KASUTADES SVD MEETODIT,* Tallinn: TalTech, 2018.

[25] E. D. Sontag, Mathematical Control Theory: Deterministic Finite-Dimensional Systems, 2nd edition, New York: Springer-Verlag New York, 1998.

[26] P. Tisa and P. Vergez, “Performance Analysis of Control Algorithms for FalconSat-3,” in *16th AAS/AIAA Space Flight Mechanics Conference,* Tampa, Florida, 2006.

[27] R. A. Paz, “The Design of the PID Controller,” New Mexico State University, 2001.

[28] C. A. Stickler and K. T. Alfriend, “Elementary Magnetic Attitude Control System,” *Journal of Spacecraft and Rockets,* vol. 13, no. 5, pp. 282-287, 1976.

[29] R. Zanetti, T. Ainscough, J. Christian and P. D. Spanos, “Q-Method Extended Kalman Filter,” *Journal in Guidance Control and Dynamics,* vol. 38, no. 4, 2015.

[30] C. Bridges, *Detailed Notes for TUT Mektory Nano-Satellite mission,* Tallinn, 2015.