NTNU

Norwegian University of
Science and Technology

Development, Implementation and Testing of Two Attitude Estimation Methods for Cube Satellites

Kristian Lindgård Jenssen
Kaan Huseby Yabar

Master of Science in Engineering Cybernetics
Submission date: May 2011
Supervisor: Jan Tommy Gravdahl, ITK
Co-supervisor: Tor Arne Johansen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Development, Implementation and Testing of Two Attitude Estimation Methods for Cube Satellites

Kristian Lindgård Jenssen & Kaan Huseby Yabar

Department of Control Engineering

Submitted in Partial Fulfillment of the Requirements For the Degree of

MASTER OF SCIENCE

Norwegian University of Science and Technology

Spring 2011

Master's Committee:
Advisor: Jan Tommy Gravdahl

Abstract

As a part of the CUBESAT project at the Norwegian University of Science and Technology (NTNU), this paper studies development and comparison of attitude estimation methods for small cube satellites using low-cost sensors. The algorithms are based on data from two vectorized measurements as well as a gyroscope.

In this paper a new method for attitude estimation has been developed based on QUaternion ESTimation (QUEST). A major concern with QUEST is that it cannot handle non-vectorized measurements such as gyroscope data. Substantial improvements have been made to fuse vectorized and non-vectorized measurements, making the new Extended QUaternion ESTimation (EQUEST) more suitable for attitude estimation. The well known Extended Kalman Filter (EKF) is derived and implemented for comparison. Both methods have been developed and simulated in MATLAB. The code have been rewritten using C language. The methods are compared both theoretically and experimentally with implementation and testing on an AVR microcontroller. Minimum power usage and number of arithmetic operations were considered during the software development.

Testing indicates that the EKF provides a smoother estimation than the newly developed EQUEST. In contrast to EQUEST, the EKF is able to estimate the magnetometer and accelerometer bias. However, the EQUEST has a significantly faster settling time and is less computational costly. Compared to the EKF, EQUEST runs more than 5 times faster. It also requires only 8% of the arithmetic operations of the EKF. Another disadvantage with the EKF is tracking problems that occur when the two vectorized measurements are close to parallel. With vectors close to parallel, the mathematical formulation of the EKF makes tracking of a rotation around the parallel axis extremely difficult. These difficulties are hardly observed in the EQUEST algorithm, which makes it very attractive for attitude estimation.

The attitude control of CUBESATs is often done by magnetorquers, which will affect the local magnetic field. Hence, control and estimation should not be done simultaneously, resulting in the estimation and control switching on and off. For this reason, the long settling time of the EKF makes the EQUEST even more attractive.

The results in this paper indicate that the newly developed EQUEST is highly suitable for projects with either limited budget, space, weight or computational power.

Preface

This project is the concluding thesis of our Master of Science degree provided by the Norwegian University of Science and Technology. We would like to thank our advisor, Professor Jan Tommy Gravdahl, for his motivating involvement and valuable support. We would also emphasize or gratitude to the rest of the student satellite project group, whom has been very helpful.

Trondheim May 26, 2011

Kristian Lindgård Jenssen & Kaan Huseby Yabar

"Take charge of your attitude. Don't let someone else choose it for you."

Contents

Nomenclature

ϵ	Euler axis
$\hat{q}_{pre}$	Predicted quaternions
λ	Eigenvalue
ν	Euler angle
ω	Angular velocity
ϕ	Roll angle
ψ	Yaw angle
θ	Pitch angle
a^b	Accelerometer measurement
B	Magnetic field vector
b	Measured directional unit vector
$bias_a$	Accelerometer bias
$bias_m$	Magnetometer bias
$bias_s$	Sun sensor bias
D	Weighting matrix for gyroscope
g	Gravitational vector
h	Measurement vector
K	Kalman gain
M	Orthogonal eigenvector matrix
m^b	Magnetometer measurement
q	Quaternion
Q_{kal}	Process covariance matrix
r	Known directional unit vector in inertial frame
R_b^a	Rotational matrix representing a rotation from b to a
R_{kal}	Measurement covariance matrix
S	Weighting matrix for prediction term
s	Sun vector

s^b	Measured Sun intensity
t	Time
T_s	Sampling time
u	Input vector
V	Weighting matrix for accelerometer and magnetometer
v	Measurement noise
w	Process noise
x	State vector
z	Measurement vector

List of Figures

1 Introduction

1.1 Motivation

Attitude estimation is a crucial part of the Attitude Control and Determination System (ADCS) for a satellite. In order to control the attitude to a known reference, knowledge about the current attitude is required. Missions such as taking photos of a specific location or to align the solar panels with the sun to optimize battery recharging requires accurate information about the satellites orientation. Many attitude estimation techniques exists, but their range of application varies with size and price. The main challenge for this project lies in the size and weight of the satellite. This induct limitations on the choice of sensors in the attitude estimation. For instance, one of the most accurate sensors, the star tracker, must be omitted from the satellite due to its size. Instead, the attitude must be determined using sensors of small size and light weight.

1.2 Project Background

This work is a subject to the CUBESAT [1] project at the Norwegian University of Science and Technology (NTNU). The NTNU CUBESAT project is a part of the Norwegian Student Satellite Program, ANSAT. The program wishes to urge a cooperation between educational institutions and the industry as well as increasing the interest for technological studies among pupils. Two other Norwegian universities are in the same program, University of Oslo and Narvik University College. Financial funding has been given NTNU by both NAROM (Norwegian Centre for Space- related Educations) and Norwegian Space Centre.

A CUBESAT is a picosatellite where the dimensions of the cube has been standarized to 10×10×10 cm, with a weight limited to 1.3 kg. Due to the low budget of CUBESAT projects, commercial components are commonly used in the design of a satellite. Previously there have been two CUBESAT projects at NTNU. The first satellite, NCube1, was destroyed due to launch failure. Even though the second satellite was launched successfully, communication was never established. The main goal for this third satellite from NTNU, is to establish communication with the satellite from the ground station in Trondheim. The satellite that is under construction has been given the name NTNU Test Satellite (NUTS) [2] . The project is now in the startup-phase, with the first students being involved since January 2011. The total timespan of the project is 4 years, with a planned launch in 2014. NUTS will be a double CUBESAT making the dimensions 10×10×20 cm with the weight limited to 2.6 kg. Many challenges are expected to occur due to the limited space, weight and budget.

The satellite is suppose to have a polar orbit, meaning that it will orbit with an inclination of close to 90 degrees - passing near the magnetic poles. The suggested payloads include a small IR camera and S-band radio. The camera will be used for research purposes, and the S-band radio will be used for high speed data transfer such as images from the camera.

1.3 Previous Work

Attitude estimation is an important part of several fields, among them military systems, navigation systems, rocket science and medical science. As mentioned, two satellites have been developed at NTNU earlier, each containing attitude estimation systems [3]. Svartveit [4] estimated the attitude by using a discrete Kalman filter based on measurements from magnetometer and sun sensor . The solar panels were used as a crude sun sensor. The experiment Svartveit did, indicated that the sun sensor seems to be inaccurate mainly due to the Earths albedo effect. Ose [5] made further work in order to implement the extended Kalman filter (EKF) in MATLAB. Due to the complexity of the EKF, only a linearized version was implemented on a microcontroller. Rohde [6] describes implementation of the extended Kalman filter on a microcontroller, but lack of time resulted in incomplete implementation and testing. With respect to all the challenges the previous NTNU students unveiled, this thesis base the attitude determination system on a different approach. The estimation is based on two vectorized measurements as well as data from a gyroscope. Sabatini [7] developed an extended Kalman filter based on accelerometer, gyroscope and magnetometer, for use in biomedical engineering. The extended Kalman filter developed in this project is adapted from the work done by Sabatini.

Attitude can also be estimated using quaternion estimation (QUEST). Markley [8] describes how two vectorized measurements can be used to estimate orientation. However, the method can only be used for vectorized measurements, which makes the gyroscope unsuited. Psiaki [9] has extended the QUEST in order to handle an arbitrary dynamic model and to estimate errors such as rate-gyro biases. The extended QUEST (EQUEST) developed in this paper, is based on the work done by Psiaki and Markley, with focus on integrating the nonvectorized gyroscope measurements.

1.4 Project Outline

Section 2 contains background information to remind the reader of the most essential theory. It is necessary to be comfortable with the background information in order to understand the main concept of the attitude estimation.

Section 3 gives information about the different sensors that can be used for a satellite. A discussion about their weakness and strengthness can also be found here. An approperiate sensor for the prototype is chosen.

Section 4 describes the attitude estimation in detail. The extended Kalman filter for the NUTS CubeSat is designed here. In addition the extended quaternion estimator is derived, including the mathematical derviation of the solution.

Section 5 explains the connection between the microcontroller and the sensor, as well as interrupt handeling and additional features for the microcontroller. The software development for both attitude algorithms are also described in detail.

Section 6 demonstrates an evaluation and comparison of the two algorithms. Here, both the run time and number of arithmetical operations are discussed.

Section 7 gives a detailed description of the prototype design. Every part of the design process is described here, ranging from the selction of components to the final prototype software implementation.

Section 8 describes all the testing. Testing includes simulations of both algorithms in MATLAB, real-time attitude estimation in MATLAB, real-time attitude estimation running on a microcontroller, comparison of the two algorithms running on a microcontroller, prototype testing and power consumption testing.

Section 9 covers the main challenges for future integration with the rest of the satellite systems. The main focus is the integration with the attitude control system.

Section 10 gives a brief discussion of the results and makes recommandations for future work.

2 Definitions and Notations

2.1 Coordinate Frame

The following frames have been used to describe the attitude of the satellite:

BODY: This frame is attached to the satellite. In the BODY frame the axes coincide with the principle axes of inertia, and the positive z-axis is defined as the vector pointing outwards from the quadratical side where the camera is attached at the satellite. The x- and y-axis are ortogonal to the rectangular sides of the cube. The body frame is illustrated in Figure 1.

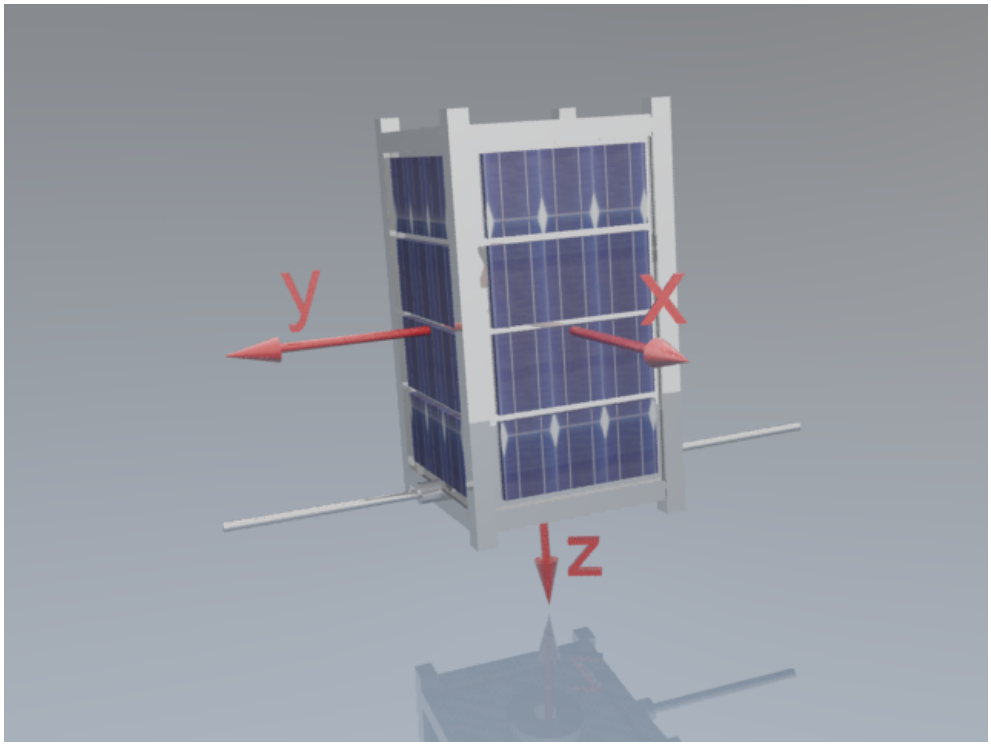


Figure 1: Illustration of the BODY coordinate system.

NED: North-East-Down (NED) is an intertial frame defined relative to Earth's surface with the x-axis pointing towards north, z-axis downwards perpendicular to Earth's reference ellipsoid. The y-axis completes a right handed orthogonal coordinate system, with the positive y-axis pointing towards East (See Figure 2). Earth's reference ellipsoid is a mathematically defined surface fitted to approximate the shape of the Earth.

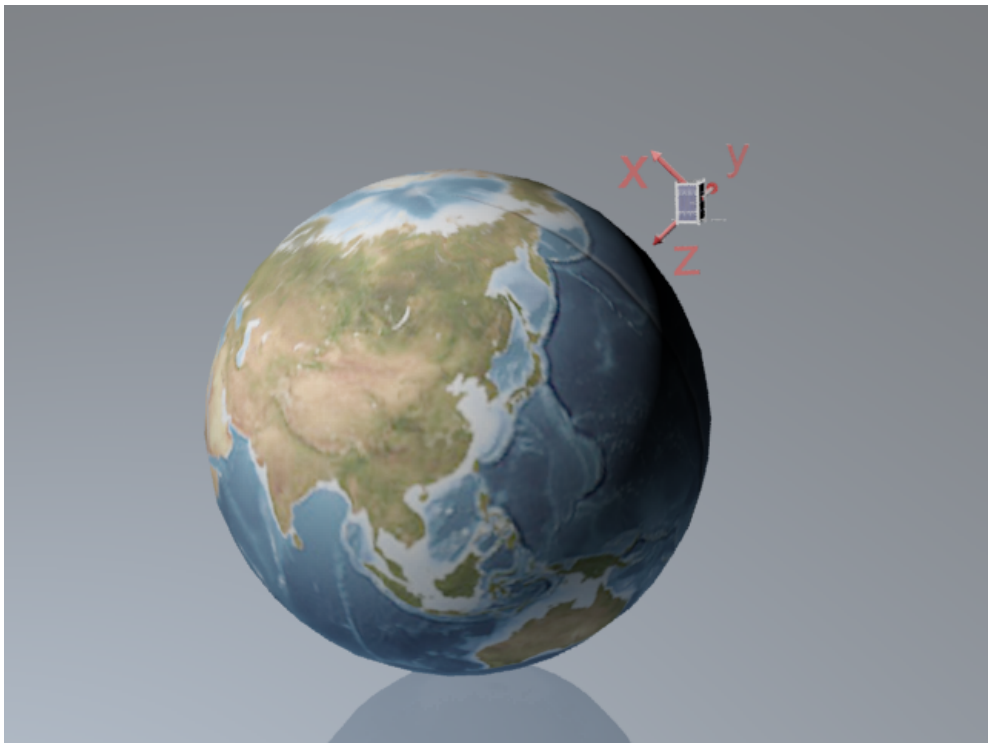


Figure 2: Illustration of the NED coordinate system.

2.2 Rotation Matrix

A rotation matrix R is used to describe a rotation between two coordinate frames[10]. It is common to describe the rotation from one frame to another as three different rotations. First a rotation ψ around the z-axis, then a rotation θ around the rotated y-axis and finally a new rotation ϕ around the current x-axis. The entire rotation from frame "b" to frame "a", is described as

$$R_b^a = R_z(\psi)R_y(\theta)R_x(\phi) \tag{1}$$

A rotation matrix has the following properties:

$$det(R) = 1 \tag{2}$$

$$R^\top = R^{-1} \tag{3}$$

A vector in NED frame k^n can be written in BODY frame k^b using rotation matrix as given below:

$$k^b = R_n^b k^n \tag{4}$$

2.3 Axis-angle Representation

Euler's Theorem:

> The most general motion of a rigid body with a fixed point is a rotation about a fixed axis.

This theorem indicates that it is possible to get from one frame to another using only one rotation about a fixed axis. This axis is called the Euler axis, and the angle rotated is called the Euler angle. Instead of describing the rotation matrix with three rotations, it can be described with one rotation around one fixed axis as

$$R_b^a = \cos\nu I + \sin\nu(\epsilon^a)^\times + (1-\cos\nu)\epsilon^a(\epsilon^a)^\top \tag{5}$$

where ϵ is the Euler axis and ν is the Euler angle. The differential equations relating the Euler axis/angle to the angular velocity, ω, are stated as [11]

$$\dot{\nu} = \epsilon^\top \omega \tag{6}$$

$$\dot{\epsilon} = \frac{1}{2}\left[\epsilon^\times - \cot\frac{\nu}{2}\epsilon^\times\epsilon^\times\right]\omega \tag{7}$$

As the equations above show, there are singularities at $\nu = 0$ and $\nu = \frac{2}{\pi}$. To avoid singularities the rotation can be represented using Quaternions.

2.4 Quaternions

An alternative to the rotations described by Euler angles is the Quaternions[12]. Quaternions are a mathematical way of representing rotations, with the convenient property that it will always be defined. The quaternions are given by equations (8) - (10)

$$q = \begin{bmatrix} q_{13} \\ q_4 \end{bmatrix} \tag{8}$$

where

$$q_{13} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = [\epsilon\sin(\nu/2)] \tag{9}$$

$$q_4 = \cos(\nu/2) \tag{10}$$

Here ν is the rotation angle for axis-angle parametrization and ϵ is a unit vector. Due to the possible spin in a satellite combined with the singularities in Euler angles kinematics, an attitude estimator based on the unit quaternions instead of the Euler angles is prefered. Another property with the unit quaternion is

that it will always satisfy the constraint $q^T q = 1$. The rotation matrix can be represented using quaternions [13]:

$$R(q) = (q_4^2 - \|q_{13}\|^2)I_{3\times 3} + 2q_{13}q_{13}^\top - 2q_4[q_{13}\times] = \Xi^\top(q)\Psi(q) \tag{11}$$

where

$$\Xi(q) = \begin{bmatrix} q_4 I_{3\times 3} + [q_{13}\times] \\ -q_{13}^\top \end{bmatrix} \tag{12}$$

$$\Psi(q) = \begin{bmatrix} q_4 I_{3\times 3} - [q_{13}\times] \\ -q_{13}^\top \end{bmatrix} \tag{13}$$

$[q_{13}\times]$ is the cross-product matrix:

$$[q_{13}\times] = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{14}$$

By inserting equation (12), (13) and (14) into (11), the rotational matrix can be written as:

$$R(q) = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 - q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_2 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_4q_1) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_4q_1) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{15}$$

The kinematic differential equation for unit quaternions is[14]:

$$\dot{q} = \begin{bmatrix} \dot{q}_{13} \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2}\Xi(q)\omega \tag{16}$$

As Equation (16) shows, there are no singularities when representing the kinematics with quaternions. This makes the design of the estimation methods more robust.

2.5 Quaternions to Euler Angles

Because of the problem with singularity all attitude calculations are done using quaternions. However euler angles are easier to understand. Therefore, to illustrate the estimated attitude, the graphs are given with euler angles (roll, pitch and yaw). To convert quaternions to euler angles the following equation can be used [15]:

$$\begin{bmatrix} \phi \\ \gamma \\ \psi \end{bmatrix} = \begin{bmatrix} atan2(R_{32}, R_{33}) \\ -sin^{-1}(R_{31}) \\ atan2(R_{21}, R_{11}) \end{bmatrix} \qquad \gamma \neq \pm 90^\circ \tag{17}$$

where

$$R(q) = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$$

This gives

$$\begin{bmatrix} \phi \\ \gamma \\ \psi \end{bmatrix} = \begin{bmatrix} atan2(2(q_4q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)) \\ arcsin(2(q_4q_2 - q_3q_1)) \\ atan2(2(q_4q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)) \end{bmatrix} \tag{18}$$

2.6 Extended Kalman Filter (EKF)

The Kalman filter is only appliable for linear problems. However, it is possible to extend the filter to also deal with nonlinear problems, but the filter will then lose some of its properties. The extended Kalman filter is adapted for nonlinear problems through a linearization of the nonlinear equations for each iteration. Due to the linearization, the extended Kalman filter is not necessarily optimal and can diverge if initial errors are too large or if the system model is inaccurate. The EKF is an iterative method where the estimates are based on several measurements, as well as a process model.

Defining the nonlinear system as

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \tag{19}$$

$$z_k = h(x_k) + v_k \tag{20}$$

where x is state vector, $f(.)$ describes the system dynamics, u is control input, w is process noise h is measurment model and v is measurement noise. Both noises are assumed to be zero mean Gaussian. The subscript k denotes the discrete time. Using the system descibed above, the equations for the extended Kalman filter can be written as[16]:

Predict:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}) \tag{21}$$

$$P_k^- = F_k P_k F_k^T + Q_{kal} \tag{22}$$

Update:

$$K_k = P_k^- H_k^\top (H_k P_k^- H_k^\top + R_{kal})^{-1} \tag{23}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H_k \hat{x}_k^-) \tag{24}$$

$$P_k = (I - K_k H_k) P_k^- \tag{25}$$

where $\hat{x}$ denotes estimated state vector, P is error covariance matrix, K is calculated Kalman gain and

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1},u} \tag{26}$$

is the derivative of the nonlinear system with respect to the states, x.

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{k-1}} \tag{27}$$

is the derivative of the measurement equations with respect to the states.

As these equations show, the nonlinear model is still used in predicting the new states, whereas a linearized model is used for comparing the measurements with the current states in the update phase and for computing the covariance matrix and the Kalman gain. The R_{kal} matrix represents the measurement covariance matrix, and the Q_{kal} is a matrix representing the process covariance.

2.7 Quaternion Estimation

Another method for estimating the rotation matrix based on the sensor measurements is the QUaternion ESTimator (QUEST)[9]. QUEST will minimize the cost function defined as:

$$\begin{aligned} J_1(q) &= \frac{1}{2}\sum_{j=1}^{n}\frac{1}{\sigma_j^2}(b_j - R_b^i(q)r_j)^\top(b_j - R_b^i(q)r_j) \\ &= \frac{1}{2}\sum_{j=1}^{n}\frac{1}{\sigma_j^2}(b_j^T b_j - 2b_j^\top R_b^i(q)r_j + r_j^\top r_j) \end{aligned} \tag{28}$$

where r_j are known unit vectors in the NED frame, and b_j are unit vectors of the measured observations in the body-fixed frame. σ_j are the standard deviation of the measurement error. However, both r_j and b_j are unit vectors, and this reduces the equation to:

$$J_1(q) = \sum_{j=1}^{n}\frac{1}{\sigma_j^2}(1 - b_j^\top R_b^i(q)r_j) \tag{29}$$

Minimizing J_1 is equal to minimizing

$$J_2(q) = -\sum_{j=1}^{n}\frac{1}{\sigma_j^2}b_j^\top R_b^i(q)r_j \tag{30}$$

or maxmimizing

$$J_3(q) = \sum_{j=1}^{n}\frac{1}{\sigma_j^2}b_j^\top R_b^i(q)r_j \tag{31}$$

The QUEST does not depend on inital conditions, which is a great advantage. Another advantage is that the algorithm can be solved exactly by solving an eigenvalue problem. However, the QUEST algorithm can only estimate the attitude quaternions. The sensor biases remain unknown.

The mathematical formulation of QUEST requires vectorized direction input vectors. This indicates that the QUEST algorithm will not be able to utilize measurements such as the non-vectorized gyroscope data. The lack of possibilities to integrate vectorized and non-vectorized measurements is a disadvantage with QUEST and limits its range of application.

3 Sensors

3.1 Magnetometers

In attitude estimation, magnetometers are used to measure the local magnetic field. By comparing the measured magnetic field with the Earth's magnetic field at the current location, the attitude vector can be estimated. This means that the satellite must know the real magnetic field for every position on its orbit. A look-up table can be implemented on the satellite to keep track of the field for a given position, but this might require an infinite amount of memory. The magnitude and the direction of the Earth's magnetic field varies with location and altitude. In addition, the field slowly changes with time. This leads to almost infinite amount of combination, making the look-up table very large. The look-up table can be made smaller if the table can be updated through ground communication. Since the orbit of the satellite is known, the look-up table only have to contain information about the magnetic field for the upcoming satellite positions. Once the satellite is within range of communication, the table can be replaced with updated field information. This form of look-up table requires data from ground and is not very reliable. If communication fails, the satellite will not be able to estimate correct attitude. Another disadvantage is that updating the table will take some of the communication bandwidth which could have been used to transfer other data. Instead of using look-up tables, a model of the Earth's magnetic field can be implemented. Several models exists. A balance between simple and accurate model is required for the estimation of attitude.

3.2 Gyroscope

Gyroscopes measure angular velocity. Theoretically a gyroscope can track the orientation of the satellite by integrating the change in velocity. In order to provide good estimation the initial orientation must be correct and in addition, the measurement errors should be small. All gyroscopes have some measurement error which is called bias. Biases are usually constant or slowly varying. Because of the biases, the orientation can not be tracked by only using gyros. Gyroscopes can track orientation for a period of time depending on the size of the biases. However, with time the attitude estimation will drift. Today, even cheap gyros are small and can provide quite good measurement data.

3.3 Accelerometer

Accelerometers measure acceleration. On Earth the accelerometer can be used to get a directional vector. If the object is still, or moving with a constant speed, the only measured acceleration will be the gravity. Even with small accelerations the gravity vector will be dominating for the measurements. Therefore, the measured acceleration will indicate the direction of the gravity vector and can be used to find the orientation of a slowly moving object. Note that if the

object do experience large accelerations, the estimation will fail to give correct estimates for the attitude.

For the attitude estimation, the direction of the acceleration is more important than the magnitude. Hence, the measurements can be normalized and compared with the normalized gravity vector in NED coordinates ($\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$). However, the satellite in orbit will experience almost no acceleration and regular accelerometers can obviously not be used. The gravity force in space can be found using Newton's law of universal gravitation and is given below:

$$F = G_c \frac{mass_1 mass_2}{dist^2} \tag{32}$$

where G_c is the gravitational constant, $mass_1$ is the mass of Earth, $mass_2$ is the mass of the satellite and $dist$ is the distance between the satellite and the Earth. For a low Earth orbit, the gravitational force is almost $0.9g$. Due to the ciruclair orbit of the satellite, the gravitational force and the centripetal force balance each other, making the total acceleration $0g$.

The accelerometer is still used in this project due to the simplicity of testing on Earth.

3.4 Sun Sensor

Sun sensors vary with complexity and mission requirements. It could be as simple as a light sensitive diode, or as complex as a optical telescope. The main idea is to measure the direction to the Sun. Small and cheap sun sensors can be bought and placed on each side of the satellite in order to detect the Sun angle. Solar cells are not really sensors, but can be used to detect the direction to the Sun by monitoring the output current. The output from a solar cell depends on the angle between the solar panel and the sun rays. The amplitude of the current varies by the cosine law [17].

The sun sensors mounted on the satellite will be sensitive to every light source in space. In addition to the Sun, light reflected by the Earth (Earth albedo) will have a great influence on the sun sensors. In order to get an accurate estimate of the Sun vecor, Earth albedo compansation in necessary. Calculating the Sun vector without Earth albedo compansation can cause an angular deviation of more than 20°[18].

3.5 Earth- and Horizon sensor

An Earth sensor is a sensor that detects the direction to the Earth. This can for instance be an infrared camera. Another sensor used to sense the direction to the Earth is the horizon sensor. This is an optical instrument that detects the boarder between the warm atmosphere and the cold cosmic background radiation.

3.6 Star Tracker

One of the most reliable sensors are the Star trackers. This is an optical device that measures the positions of stars relative to the sensor. The results are compared with a star catalog that contains the position of many well known stars stored in memory. By using simple geometry it is possible to determine the position of the star sensor based on which stars the sensor sees.

3.7 Sensors Used for Testing

As mentioned above there exists many sensors that can be used for attitude estimation. Which sensors to choose is a question of space, weight and financial budget. The main limitations for the CUBESAT is the small amount of space and weight. Limited power capacity must also be taken into consideration. Because of those limitations sun sensors, gyros and magnetometers were considered a better option than optical devices such as star trackers and Earth horizon sensor.

For the development of attitude estimation methods, and for the prototype design, the sun sensor was replaced by an accelerometer. This was done for testing purposes. Estimating the attitude using sun sensors on Earth would demand a proper test bench. The bench must at least contain a mounted light source moving around the test bench to simulate the Sun, as well as a satellite module with solar cells. Since the project is early in the start up phase, there is no access to the required solar cells. Due to better testing options and limited access to components, the prototype has been designed with an accelerometer instead of a sun sensor. Section 4 will give a mathematical explanation why it is possible to replace the sun sensor with an accelerometer for testing on Earth.

When selecting a sensor for the prototype, only magnetometers, gyros and accelerometers were considered. They can either be bought seperatly or as a complete module. Buying seperatly gives the opportunity to create an own Inertial Measurement Unit (IMU) with desired specifications. Each sensor can be choosen individually to best match the requirements of the prototype. Buying a complete IMU gives limited sensor choice, but requires less work. A complete IMU is more reliable as they already have been tested by companies specialized in IMUs. For this project, buying a complete IMU was prefered. 3 different IMUs were used for testing and implementation. The sensors are decribed below. All IMUs have 3-axis accelerometer, 3-axis gyro and 3-axis magnetometer.

3.7.1 Xsens MTi

Xsens MTi is a miniature, gyro-enhanced attitude and heading reference system. It provides drift free 3D orientation. The sensor gives output in both quaternions as well as raw sensor data. This sensor was mainly used to test the Kalman filter and the EQUEST in MATLAB (See Section 8.1 and 8.2). The Kalman filter uses raw sensor data as input. The estimated attitude is given as quaternions making it possible to compare with the attitude calculations done by Xsens MTi.

The sensor cost about 2500 USD. The dimension of the sensor is $58 \times 58mm$, which makes it unsuitable for the prototype. Xsens MTi is still a good sensor for software test purposes.

3.7.2 9-DOF Razor IMU

The dimensions ($49.53 \times 27.94mm$) of the 9-DOF Razor IMU also makes it unsuitable for the prototype, but because of its simplicity, the sensor was used to test both extended Kalman filter and the extended quaternion estimation (See Section 8.3, 8.4 and 8.5). It can easily be connected to computer using serial RX and TX pins. The sensor operates with a baudrate of $38400bps$ and can be configured to only send raw data separated by commas. The price of the sensor is about 125 USD, making it a fairly cheap investment.

3.7.3 CHIMU IMU

The CHIMU is a miniature, low cost (350 USD) attitude heading reference system. It can provide both raw data and orientation. The dimension of the sensor is $25.4 \times 21.0mm$ making it highly suitable for the prototype. The output data is sent out as a message with defined structure. This sensor was used for the prototype (See Section 7), but because of the fast improvement in small electronics, and the fact that the accelerometer shall be replaced by sun sensors, the final sensor selection for NUTS in 2014 must be chosen carefully.

4 Attitude Estimation

Attitude was estimated using extended Kalman filter and extended quaternion estimation. Both estimation methods use two independent directional vectors and a gyroscope for attitude estimation. Hence, it is indifferent wether the methods are based on sun sensors measuring the direction to the sun, or accelerometers measuring the gravitational force on Earth. The only difference will be the input data and the reference vector for the algorithms. The methods below are developed for an accelerometer, gyroscope and a magnetometer, but the accelerometer can easily be replaced by sun sensors.

4.1 Attitude Estimation Using Extended Kalman Filter

The theory of the extended Kalman filter was explained in section 2.6. There are several ways to configure the extended Kalman filter for attitude estimation based on measurements from an IMU. Here, the state vector was chosen to consist of the quaternions q, accelerometer bias $bias_a$ and the magnetometer bias $bias_m$. The state vector x is given below:

$$x = \begin{bmatrix} q \\ bias_a \\ bias_m \end{bmatrix}$$

where $q = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^\top$, $bias_a = \begin{bmatrix} bias_{a,x} & bias_{a,y} & bias_{a,z} \end{bmatrix}^\top$ and $bias_m = \begin{bmatrix} bias_{m,x} & bias_{m,y} & bias_{m,z} \end{bmatrix}^\top$. The great advantage of estimating the biases of the accelerometer and magnetometer is that the EKF will subtract them from the measurements during the next iteration. This will increase the precision of the measurements and give a more accurate attitude estimate.

In order to include the angular velocity as a state, a dynamical model of the satellite can be used. An alternative is to use the measurements from the gyroscope to estimate the dynamical model for the quaternions.

The differential equation for quaternions can be written in terms of angular velocity $\omega = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^\top$ [7]

$$\dot{q} = \frac{1}{2} \begin{bmatrix} \omega\times & \omega \\ -\omega^\top & 0 \end{bmatrix} q \tag{33}$$

where

$$\omega\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{34}$$

Further, the accelerometer and magnetometer biases can be assumed to be constant. Following differential equations can be written

$$\dot{bias}_a = 0 \tag{35}$$

$$\dot{bias}_m = 0 \tag{36}$$

The differential equation for the entire system is therefore

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \dot{bias}_a \\ \dot{bias}_m \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\begin{bmatrix} \omega\times & \omega \\ -\omega^\top & 0 \end{bmatrix} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} q \\ bias_a \\ bias_m \end{bmatrix} \tag{37}$$

Note that the system equation is linear, making the computation of the EKF less complicated. However, an EKF is still necessary due to the nonlinearity in the measurement model.

The system given in Equation (37), is given in continious time. To implement this on a computer, a discretization is needed. The system is discretized using zero-order hold with sampling time T_s. The discrete system will be given as

$$x_{k+1} = \begin{bmatrix} expm\left(\frac{1}{2}\begin{bmatrix} \omega\times & \omega \\ -\omega^\top & 0 \end{bmatrix} T_s\right) & 0 & 0 \\ 0 & I_{3x3} & 0 \\ 0 & 0 & I_{3x3} \end{bmatrix} x_k \tag{38}$$

with $expm$ being the matrix exponential.

Since all the measurements are done in the body frame and the reference model is given in the NED frame, the sensor model includes a rotational matrix dependent on the unit quaternions. The rotational matrix introduce a non-linearity in the measurement matrix. The measurement Equation (20) is given by the sensor model below.

$$a^b = R_n^b(q)(g^n + a_{real}^n) + bias_a^b \tag{39}$$

$$m^b = R_n^b(q)m_{real}^n + bias_m^b \tag{40}$$

where g^n is the gravity vector and R_n^b is the rotation matrix rotating a vector from NED to BODY frame. In matrix form the measurement model is given by

$$\begin{bmatrix} a^b \\ m^b \end{bmatrix} = \begin{bmatrix} a_x^b \\ a_y^b \\ a_z^b \\ m_x^b \\ m_y^b \\ m_z^b \end{bmatrix} = \begin{bmatrix} R_n^b(q) & 0 \\ 0 & R_n^b(q) \end{bmatrix} \begin{bmatrix} g_x^n + a_{real,x}^n \\ g_y^n + a_{real,y}^n \\ g_z^n + a_{real,z}^n \\ m_{real,x}^n \\ m_{real,y}^n \\ m_{real,z}^n \end{bmatrix} + \begin{bmatrix} bias_a^b \\ bias_m^b \end{bmatrix} \tag{41}$$

Because of the nonlinearity in the measurement model, linearization is required and is derived in Appendix D.

The measurement vector z consist of the measurements from accelerometer a^b and the magnetometer m^b

$$z = \begin{bmatrix} a^b \\ m^b \end{bmatrix}$$

The angular velocity is not either a part of the measurement vector. As previous mentioned, the measurements from the gyroscope will be indirectly integrated in the EKF, through the dynamical model of the quaternions.

The vectors explained above are used in the equations in Section 2.6 to iterativly compute the attitude of the satellite.

4.2 Extended Quaternion Estimator

As mentioned in Section 2.7, the QUEST algorithm is not able to utilize the measurements from the gyroscope. However, it is possible to extend the QUEST and implement an Extended QUaternion ESTimator (EQUEST) in order to include the gyroscope measurements. The main idea behind the EQUEST is to modify the cost function. This is done by adding another term, containing the gyroscope measurements.

By tracking the rotation based on gyroscope measurements it is possible to add a term to Equation (28) that will penalize aberrations from the rotation matrix estimated by the gyroscope measurements only.

$$J_4(q) = \frac{1}{2}\sum_{j=1}^{n}\left\{\frac{1}{\sigma_j^2}(b_j - R_b^i(q)r_j)^\top(b_j - R_b^i(q)r_j)\right\} + \frac{1}{2}(q - \hat{q}_{gyro})D(q - \hat{q}_{gyro})) \tag{42}$$

here $\hat{q}_{gyro}$ is the estimated rotation matrix based on gyroscope tracking, and D is a diagonal weighting matrix. The main idea by using the term $q - \hat{q}_{gyro}$ is to minimize the cost function. Note that the subtraction of two quaternions will not result in an attitude quaternion.

The EQUEST can be expanded further by adding a prediction term. The prediction is most suitable for applications where it is possible to forecast upcoming orientation based on previous behavior. It can also be used to filter out noise. The slow and predictable change of attitude for the satellite makes it possible to use previous attitude calculations to estimate the next orientation. For a short period of time, the attitude change will be minimal. However, as several attitude calculations are done in this period it is possible to establish a linear relation between time and change of attitude. This is illustrated in Figure 3. A deviation from the predicted term can be penalized in the cost function by adding the following term to Equation (42)

$$\frac{1}{2}(q - \hat{q}_{pre})^\top S(q - \hat{q}_{pre}) \tag{43}$$

where q vector contains attitude quaternions, $\hat{q}_{pre}$ is the predicted attitude based on previous observations, S is state weight matrix.

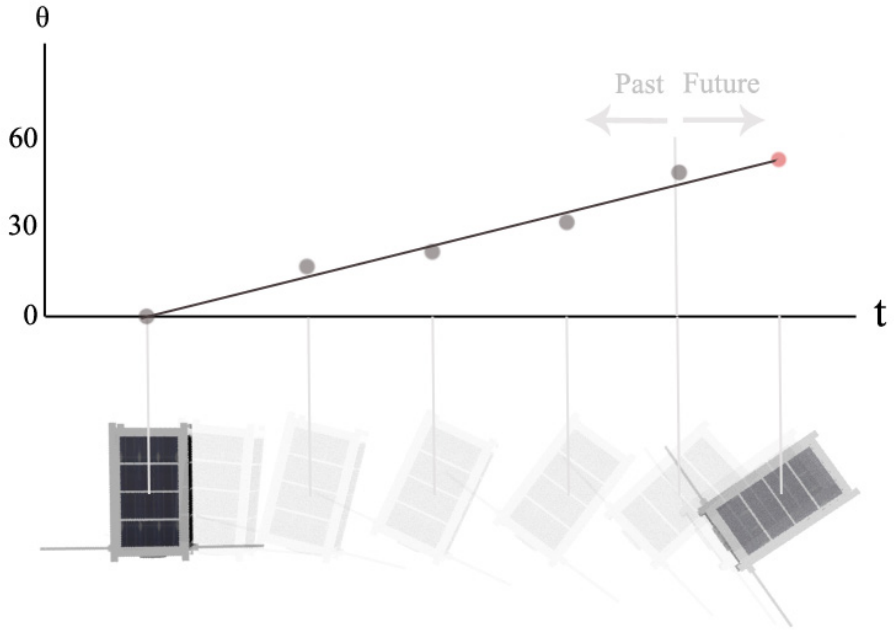


Figure 3: Linear prediction of attitude.

Adding Equation (43) with (42) gives

$$\begin{aligned} J_5(q) &= \frac{1}{2}\sum_{j=1}^{n}\left\{\frac{1}{\sigma_j^2}(b_j - R_b^i(q)r_j)^\top(b_j - R_b^i(q)r_j)\right\} \\ &+ \frac{1}{2}(q-\hat{q}_{gyro})^\top D(q-\hat{q}_{gyro}) + \frac{1}{2}(q-\hat{q}_{pre})^\top S(q-\hat{q}_{pre}) \end{aligned} \quad (44)$$

subject to

$$q^\top q = 1 \quad (45)$$

It is possible to rewrite both extensions to quadratic terms. By writing the entire equation in quadratic form, the cost function can be minimized using well-known optimization techniques[19].

Note that the EQUEST is still not able to estimate the biases. It is possible to estimate the biases using other methods, and then subtract them from the measurements used in EQUEST. Due to computational costs, this is not considered here.

4.3 Solving the Extended Quaternion Estimator

One option for solving the minimization problem given in (44) and (45) is to use the Lagrangian multiplier method. However, the equation has to be rewritten on the special form

$$J(x) = \frac{1}{2}x^\top Gx + x^\top c \tag{46}$$

where G is a positive definite matrix and c is a constant with respect to x. The original QUEST criterion in (28) can be posed in the quadratic form [8]

$$g(q) = -q^\top Vq \tag{47}$$

where V is a symmetric matrix given by

$$V = \begin{bmatrix} U - \varphi I_{3\times 3} & Z \\ Z^\top & \varphi \end{bmatrix} \tag{48}$$

with

$$U = L + L^\top \tag{49}$$

$$L = \sum_{j=1}^{n} \frac{1}{\sigma_j^2}(b_j r_j^\top) \tag{50}$$

$$Z = \begin{bmatrix} L_{23} - L_{32} \\ L_{31} - L_{13} \\ L_{12} - L_{21} \end{bmatrix} \tag{51}$$

$$\varphi = trace(L) \tag{52}$$

The gyroscope tracking needs to be written in the same quadratic form in order to solve EQUEST with Lagrangian multipliers.

$$\frac{1}{2}(q - \hat{q}_{gyro})^\top D(q - \hat{q}_{gyro}) = \frac{1}{2}(q^\top D - \hat{q}_{gyro}^\top D)(q - \hat{q}_{gyro}) \tag{53}$$

$$= \frac{1}{2}(q^\top Dq - q^\top D\hat{q}_{gyro} - \hat{q}_{gyro}^\top Dq + \hat{q}_{gyro}^\top D\hat{q}_{gyro}) \tag{54}$$

Since

$$(\hat{q}_{gyro}^\top Dq)^\top = \hat{q}_{gyro}^\top Dq = q^\top D\hat{q}_{gyro} \tag{55}$$

we get

$$\frac{1}{2}(q - \hat{q}_{gyro})^\top D(q - \hat{q}_{gyro}) = \frac{1}{2}(q^\top Dq - 2q^\top D\hat{q}_{gyro} + \hat{q}_{gyro}^\top D\hat{q}_{gyro}) \tag{56}$$

further, the term $\hat{q}^T D\hat{q}$ will be constant with respect to q. This term will not affect the minimization problem, hence we can remove it from the equation.

Now the gyroscope part of the cost function can be written in quadratic form as:

$$\frac{1}{2}q^\top Dq - q^\top D\hat{q}_{gyro} \tag{57}$$

Exactly the same as above can be done to write the prediction term in quadratic form:

$$\frac{1}{2}q^\top Sq - q^\top S\hat{q}_{pre} \tag{58}$$

By adding Equation (57) and Equation (58) into Equation (47), the entire EQUEST can be written in quadratic form as:

$$J_6 = \frac{1}{2}q^\top (D + S - V)q + q^\top(-D\hat{q}_{gyro} - S\hat{q}_{pre}) \tag{59}$$

Introducing new variables

$$\kappa = D + S - V \tag{60}$$

$$\xi = -D\hat{q}_{gyro} - S\hat{q}_{pre} \tag{61}$$

the problem will be to minimize

$$J(q) = \frac{1}{2}q^\top \kappa q + q^\top \xi \tag{62}$$

subject to

$$q^\top q = 1 \tag{63}$$

The lagrangian equation is now given as

$$\mathcal{L} = \frac{1}{2}q^\top \kappa q + q^\top \xi + \frac{\lambda}{2}(q^\top q - 1) \tag{64}$$

finding the extremum of (64) can be done by finding the derivative with respect to q, and setting the equation equal to zero:

$$\frac{d\mathcal{L}}{dq} = \kappa q + \xi + \lambda I q = 0 \tag{65}$$

$$q = -(\kappa + \lambda I)^{-1}\xi \tag{66}$$

by combining (63) and (66), the constraint now sounds

$$\xi^\top (\kappa + \lambda I)^{-2}\xi = 1 \tag{67}$$

The most positive real λ will give the global minimum for Equation (62) [9]. Further κ is a symmetric matrix, hence it can be decomposed as:

$$\kappa = M \begin{bmatrix} -\lambda_1 & 0 & 0 & 0 \\ 0 & -\lambda_2 & 0 & 0 \\ 0 & 0 & -\lambda_3 & 0 \\ 0 & 0 & 0 & -\lambda_4 \end{bmatrix} M^\top \tag{68}$$

where λ_i are the eigenvalues of κ, and M is an orthogonal eigenvector matrix. During the calculations, it is important that the eigenvalues corresponds to the correct eigenvector. By introducing

$$c = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = M^\top \xi \Leftrightarrow \xi^\top = c^\top M^\top \tag{69}$$

Equation (67) can be written as

$$c^\top M^\top \left[M \begin{bmatrix} -\lambda_1 & 0 & 0 & 0 \\ 0 & -\lambda_2 & 0 & 0 \\ 0 & 0 & -\lambda_3 & 0 \\ 0 & 0 & 0 & -\lambda_4 \end{bmatrix} M^\top + \begin{bmatrix} \lambda & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & \lambda & 0 \\ 0 & 0 & 0 & \lambda \end{bmatrix} \right]^{-2} Mc - 1 = 0 \tag{70}$$

Since the diagonal matrix will affect the eigenvalues, and the inverse of an orthogonal matrix is the transposed of the orthogonal matrix this can be simplified to

$$c^\top M^\top \left[M \begin{bmatrix} \lambda - \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda - \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda - \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda - \lambda_4 \end{bmatrix} M^\top \right]^{-2} Mc - 1 = 0 \tag{71}$$

Now since M is orthogonal, $M^\top M$=I. Further, the inverse of a diagonal matrix can be computed element by element. Equation (71) can now be written as

$$c^\top \begin{bmatrix} \frac{1}{(\lambda-\lambda_1)^2} & 0 & 0 & 0 \\ 0 & \frac{1}{(\lambda-\lambda_2)^2} & 0 & 0 \\ 0 & 0 & \frac{1}{(\lambda-\lambda_3)^2} & 0 \\ 0 & 0 & 0 & \frac{1}{(\lambda-\lambda_4)^2} \end{bmatrix} c - 1 = 0 \tag{72}$$

By writing this out we get

$$\frac{c_1^2}{(\lambda-\lambda_1)^2} + \frac{c_2^2}{(\lambda-\lambda_2)^2} + \frac{c_3^2}{(\lambda-\lambda_3)^2} + \frac{c_4^2}{(\lambda-\lambda_4)^2} - 1 = 0 \tag{73}$$

The optimal λ (λ_{opt}) will be larger than the smallest eigenvalue. After λ_{opt} is identified, it can be substituted back into Equation (66) to find the q that minimizes the cost function.

In contrast to EKF, EQUEST is not able to estimate the bias in accelerometer and magnetometer measurements. The state vector is therefore chosen to be the quaternions

$$x = q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{74}$$

Equation (16) is used to estimate the $\hat{q}_{gyro}$ term in EQUEST.

We propose to calculate the $\hat{q}_{pre}$ term by using simple linear regression with a window size of 10 samples. The next quaternion vector is predicted by using the 10 latest samples, and fitting them to an equation for a line

$$y(t) = b_0 t + b_1 \tag{75}$$

b_0 will represent the slope of the line, while b_1 is the measured value at $t = 0$. With n observations, b_0 and b_1 can be found by solving the following formulas[20]:

$$b_0 = \frac{n\sum_{i=1}^{n} y(t_i)t_i - \sum_{i=1}^{n} t_i \sum_{i=1}^{n} y(t_i)}{n\sum_{i=1}^{n}(t_i^2) - \left(\sum_{i=1}^{n} t_i\right)^2} \tag{76}$$

$$b_1 = \frac{\sum_{i=1}^{n} y(t_i)\sum_{i=1}^{n} t_i^2 - \sum_{i=1}^{n} t_i \sum_{i=1}^{n} t_i y(t_i)}{n\sum_{i=1}^{n} t_i^2 - \left(\sum_{i=1}^{n} t_i\right)^2} \tag{77}$$

The next $\hat{q}_{pre}$ is estimated using the linear relation found with the last 10 samples using the values b_0 and b_1 as line parameters. Note that the prediction term does not have to be linear. It should be adapted for the rest of the system. In case of less strict restrictions on calculational power or time, the predicition term might be an EKF. Obviously this would be to demanding for the attitude estimation on board the satellite, but if computational power allows it this might be a good solution.

4.4 Replacing the Accelerometer with Sun Sensors

It is easy to replace the accelerometer with any other directional measurement. Below the accelereometer is replaced by a sun sensor in order to achieve attitude estimation while orbiting. In the Kalman filter, the states describing the accelerometer must be changed to describe the sun sensors

$$x = \begin{bmatrix} q \\ bias_s \\ bias_m \end{bmatrix}$$

The new system matrix would then be

$$x_{k+1} = \begin{bmatrix} expm\left(\frac{1}{2}\begin{bmatrix} \omega\times & \omega \\ -\omega^\top & 0 \end{bmatrix} Ts\right) & 0 & 0 \\ 0 & I_{3x3} & 0 \\ 0 & 0 & I_{3x3} \end{bmatrix} \begin{bmatrix} q_k \\ bias_{s,k} \\ bias_{m,k} \end{bmatrix} \tag{78}$$

with the following measurement model for the sun sensors

$$s^b = R_n^b(q)s^n + bias_s \tag{79}$$

The measurement vector would also be changed to contain the sun measurements

$$z = \begin{bmatrix} s^b \\ m^b \end{bmatrix}$$

where s^b is the normalized Sun intensity measured in BODY frame calculated from the measured Sun intensity from the solar sensors. This gives one measurement in each direction x, y, z in BODY coordinates.

Since the sun vector is a directional vector, it could be implemented directly in the QUEST or EQUEST algorithm. The replacement would only change the known b and r vectors in Equation (28), where the sun vector replaces the accelerometer/gravity vector.

To use the solar cells as sun sensors, let Int_{x+}, Int_{x-}, Int_{y+}, Int_{y-}, Int_{z+} and Int_{z-} be measured intensity at solarcells on each side of the satellite. Subscript x means side perpendicular to the x-axis , subscript " $+$ " indicates the side on positive axis while " $-$ " indicates the opposite. The measured intensity s^b can be desribed using the intensity from each side as following:

$$s_{meas} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} Int_{x+} \\ Int_{x-} \\ Int_{y+} \\ Int_{y-} \\ Int_{z+} \\ Int_{z-} \end{bmatrix} \tag{80}$$

When replacing the accelerometer with sun sensors, a good albedo model should be developed. This is because the sun sensors will be affected by reflections from the Earth, and could therefore experience great inaccuracy in the estimations. Implementing an albedo model can compensate for these deviations.

5 Software Implementation

The software was written using C language an implemented on an ATMEGA2560, ATMEGA2561 and an ATMEGA128 microcontroller. AVR studio was used as development environment. Note that the ATMEGA 128 was not able to run both methods due to memory issues. The code and description in this section is written for the ATMEGA2560 and requires minor register changes to be adapted for the ATMEGA2561.

5.1 Establishing Communication Between the Atmega Microcontroller and the Sensor

Communication between the microcontroller and the sensor is done using Universal Asynchronous Receiver/Transmitter (UART). To be able to use the UART some predefined registers on the microcontroller must be set. Transmission and reception can be activated by setting the RXEN and TXEN bits in the control register UCSRB. The baudrate register must also be set before UART can be used to send information. The baudrate register is 16-bit, split into two 8-bit registers UBRR0H and UBRR0L. Note that the baudrate register value is not the same as the baudrate for the communication. A formula is needed to convert the desirable baudrate, $Baud$, to a value applied to the baudrate register, Reg. The formula is given below:

$$Reg = \frac{CPU}{16Baud} - 1 \tag{81}$$

where CPU is the clock rate. The code necessary for initilizing UART is summerized below (A baudrate of 38400 was chosen here):

```
#include <avr/io.h>
#define BAUDRATE 38400
#define MYUBRR (F_CPU/16/BAUDRATE-1)

UCSR0B = (1<<RXEN1)|(1<<TXEN1);
UBRR0H = (unsigned char)(MYUBRR>>8);
UBRR0L = (unsigned char)MYUBRR;
```

A method was created to send data from the microcontroller to the sensor and is given below:

```
void USART_vSendByte(uint8_t u8Data){
   while((UCSR0A &(1<<UDRE0)) == 0);
   UDR0 = u8Data;
}
```

The while loop runs until UDR register is ready for new data. This is done by checking the UART data register empty flag (UDRE), also found in UCSRA register in microcontroller. An additional method was written to recieve data from the sensor to the microcontroller. The code is given below:

```
char USART_vReceiveByte(void){
   while((UCSR0A&(1<<RXC0)) == 0);
   return UDR0;
}
```

Here, the while loop runs until data (one byte) have been recieved. When a byte is recieved the UDR register will contain the data. The code given in this section is the minimum code which is necessary to recieve and transfer data using UART. The code can easily be adapted to almost any AVR microcontroller with no or small changes in the code.

5.2 Interrupt Handling

Interrupts allows external events to pause the main program and execute an interrupt service routine (ISR) before resuming the main program where it left off. Here, interrupts will be used with UART. Instead of continously checking for incoming data from UART, interrupts can be used to notify when new data arrives. In order to use interrupt two library headers must be added:

```
#include <avr/io.h>
#include <avr/interrupt.h>
```

To hadle interrupts from UART, global interupt must be allowed. In addition, the "recieve complete interrupt" must be enabled. This is done by setting the bit RXCIE in UCSR0B register:

```
UCSR0B = (1<<RXCIE0);
sei();
```

The function "sei();" activates the global interrupt. Once the main program is interrupted, the interrupt service routine is excecuted. The code which is to be excecuted must be written inside the ISR function with correct vector name:

```
ISR(USART0_RX_vect) {

}
```

USART0_RX_vect is the name of the vector that handles the reception of a byte via UART. The name may be different for other AVR microcontrollers, but can be found in the datasheet[21].

5.3 Implementing the Kalman Filter in C

In order to implement the extended Kalman filter on the microcontroller, the Kalman filter was written using C language. Several diffuculties arrises when the C language is used as several mathematical operations are not supported. First of all, C do not support matrix multiplication which is an essential part of

the Kalman filtering. Secondly the matrix inverse operation does not exist. The third challenge is to find the transpose of a matrix. C supports two-dimensional arrays which looks very similar to a matrix. Using double arrays makes the code more complex. Therefore instead of using two-dimensional arrays, matrices were transfered into a long one-dimensional arrays. An example is given below:

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

Using 3 for-loops, matrix multiplication can be implemented as following:

```
void matrisemul(double *CC, double *AA,  double *BB,
int sizeAm, int sizeAn, int sizeBm,int sizeBn){
   double sum;
   for (int i = 0; i<sizeAn;i++){
      for(int j = 0; j<sizeBn; j++){
         sum = 0;
         for(int k = 0; k<sizeBm;k++){
            sum = sum + AA[i*sizeAn+k]*BB[k*sizeBn+j];
         }
      CC[i*sizeBn+j] = sum;
      }
   }
}
```

The method takes two matrices, written as one-dimensional arrays, as input. The size of the matrices must also be sent to the method. The output is the product of the matrices given as a one dimensional array.

For each iteration in the Kalman filter, only two matrices must be transposed. The code for transposing the matrices is therefore not written as a method. The matrices which is to be transposed have dimensions 10×10 and 6×10 but again given in one dimensional arrays. The C code to find their transposed is as following:

```
for(int i = 0; i < 10; i++){
   for(int j = 0; j < 10; j++){
      A_t[10*i+j]=A[j*10+i];
      if(j<6){
         H_t[6*i+j]=H[j*10+i];
      }
   }
}
```

Computing the inverse of a matrix is only possible if the matrix is positive-definite. The matrix which is to be inverted in the Kalman filter has properties making it always invertable. It is a 6×6 symmetric matrix, meaning it is

Hermetian, and since every eigenvalue in Hermetian matrices is positive, the matrix is positive definite. An efficient way to compute the inverse of a matrix is by performing an LU decomposition (See Appendix E). The algorithm for inverse operation using LU decomposition can be found in "Numerical Recipes in C"[23].

5.4 Implementing the EQUEST in C

The EQUEST was also written using C language. One of the great advantages with the EQUEST, is the few aritmethic operations. There were only two difficulties when the code was rewritten from MATLAB to C: one matrix inversion and one eigenvalue problem. However, it is only a matrix inverse of size 4-by-4. Since the matrix is so small, it is more efficient to invert it using the adjoint method than the LU decomposition. For larger matrices, the adjoint method tends to be very difficult to follow as the operations are increasing with O(n!). To solve the EQUEST, it is necessary to identify the smallest eigenvalue and all eigenvectors of a symmetric 4-by-4 matrix. This is done by implementing the cyclic Jacobi method which returns all eigenvalues and the corresponding eigenvectors of the input matrix. The method was first formulated in 1846 by the German mathematician Carl Gustav Jakob Jacobi[24], and is a common way of identifying eigenvalues.

5.5 Sleep Mode

The attitude is not necessarily estimated continuously. For some period of time the estimation is less important and energy can be saved by shutting down part of the estimation system. On AVR microcontrollers, it is possible to shut down unused modules, so called sleep mode. To enter the sleep mode, the SE bit in SMCR (Sleep Mode Control Register) must be set to logic one and in addition a SLEEP instruction must be executed. To wake up, an "enable" interrupt signal must occur. To be able to use the sleepmode following library must be added:

```
#include <avr/sleep.h>
```

Instead of changing the SE bit to enter sleep mode, some simple commands can be used:

```
set_sleep_mode(SLEEP_MODE_PWR_DOWN);
sleep_enable();
sleep_mode();
```

The first line sets the sleep mode. The microcontroller has six sleep modes, each with different active parts. SLEEP_MODE_PWR_DOWN command activates Power-down mode, which turn off every active clock and both oscillators. The "sleep_enable()" command enables the sleepmode, while "sleep_mode()" is the command that actually puts the micro-controller to sleep. In order to wake up the controller pin change interrupt was chosen among the different options.

Change in PORTB, PIN0 will deactivate the sleep mode. To activate pin change interrupt on PORTB, the following code was required

```
PCMSK0 = (1<<PCINT0);
PCICR = (1<<PCIE0);
```

Once the value on PIN0 goes from high to low or from low to high, the interrupt service routine runs. The interrupt service routine which deactivates the sleepmode is given below

```
ISR(PCINT0_vect){
    sleep_disable();
}
```

5.6 Watchdog Timer

In order to prevent endless loops in the code, it is wise to implement a Watchdog Timer (WDT). This timer will increment itself until a selectable number is reached, then the entire program will restart. In each code loop the watchdog timer must be set to zero. If the code enters an endless loop, the WDT will continiue to increment and reach the selected number. This results in the entire micro controller to restart. It is important to select the number high enough, to let the code finish a code loop under normal circumstances.

To use the watchdog timer, the following library must be added:

```
#include <avr/wdt.h>
```

The following code enables the watchdog timer with a limit of 500ms before a system restart:

```
wdt_enable(WDT0_500MS);
```

Before a new estimation starts, the watchdog timer must be reset.

```
wdt_reset();
```

During sleep mode the watchdog timer has to be turned off. The method below shows how to turn of the watchdog timer

```
void WDT_off(void){
    wdt_reset();
    MCUSR &= ~(1<<WDRF);
    WDTCSR |= (1<<WDCE) | (1<<WDE);
    WDTCSR = 0x00;
}
```

By not clearing the watchdog system reset flag (WDRF) in the MCUSR register the code will run an eternal loop of time-out resets. To prevent unintentional time-outs the WDCE and WDE must take the value one before turning off the watchdog timer. The watchdog timer is turned off by setting WDTCSR to zero.

5.7 Run-time Counter

A timer was implemented to be able to measure the run-time of the two algorithms. The microcontroller has an integrated timer which can be used for this purpose. To start the timer following code should be implemented

```
TCNT1 = 0;
TCCR1B = (3 << CS10);
```

By zeroing TCNT1 register, the timer is reset. The TCCRB register is set in order to prescale the frequency to $\frac{1}{64}$ of the clock frequency. To stop the timer and get the result, the lines below were written:

```
sreg = SREG;
cli();
i = TCNT1;
SREG = sreg;
TCCR1B = 0;
sei();
```

In first line the global interupt flag SREG is stored. Calling "cli()" disables interrupts. The counter value is read from the TCNT register and the timer is stopped by resetting the TCCRB register. "sei()" enables the interrupts.

6 Evaluating the Algorithms

Both the EKF and the EQUEST algorithms are suitable for estimating the attitude of a satellite. As of today, the extended Kalman filter is the most common estimator, and it has been tested in similar applications for decades. This makes the EKF ideal for performance- and run time comparison for the newly developed EQUEST algorithm. Both algorithms have been implemented and executed on an AVR ATMEGA 2560 microcontroller. During the evaluation the microcontroller was running at 8MHz. The ATMEGA 2560 can go up to 16MHz but running close to max speed is not recommended due to instability. The prototype is designed with an ATMEGA 2561, running at 16MHz. The microcontroller was able to run both methods satisfactorily. Most of the variables were implemented as floating point numbers. Computational time can be saved by changing from floating point variables to fixed point, but this is left out for further work.

6.1 Extended Kalman Filter

The EKF requires some matrix operations, among them matrix multiplication and computing the inverse of a 6-by-6 matrix. Since the module should be used on-board a satellite, the memory usage should be as optimal as possible. Hence, it is not desirable to implement an entire math library in order to perform the matrix operations. Therefore only the necessary methods have been implemented at the microcontroller. The linearization contains quite a large amount of numerical operations. The performance of the algorithm has been measured both in expended run time and number of arithmetic operations. The expended run time is found by setting a flag at the start of each cycle, and then resetting the flag after the execution. The run time of the EKF is about 200 milliseconds. By introducing a global counter in the algorithm, it is possible to detect how many arithmetic operations each cycle executes. In average EKF required about 40 000 operations. During the design of the prototype, the implementation of the EKF resulted in a stack overflow at an ATMEGA 128 microcontroller. The ATMEGA 128 has the exact same kernel as the ATMEGA 2561, but half the memory.

6.2 Extended Quaternion Estimation

Compared to the EKF, the EQUEST requires less matrix multiplications, and only a 4-by-4 matrix inversion. However, the eigenvalues and eigenvectors of a 4-by-4 matrix must be found. The EQUEST algorithm, does not require any linearization. Number of arithmetic operations for the EQUEST was found to be about 3200, which is only 8% of the EKF's operations. The run time for EQUEST is about 40ms. This means that EQUEST is about 5 times faster than the EKF. The ATMEGA 128 had no problems running the EQUEST algorithm. However, larger memory might be needed to implement attitude control along with the attitude determination. After implementation of both methods on the

same ATMEGA 2561, the memory was 31.0% full. Further details about the design of the prototype follows in the next section.

7 Designing Prototype

In order to make a functional prototype a systematic design process is followed. The procedure consists of the following steps, and is described in more detail below:

1. Identify electrical components
2. Schematic design
3. Board design
4. Milling circuit board
5. Soldering components
6. Software implementation

There are several electrical components needed in order to make a functional prototype. First all the necessary components were identified. The sensor was selected with the criteria discussed in Section 3, where the CIMU IMU was found most suitable. The microcontroller was selected based on the earlier software development being done on an ATMEGA 2560. To get a stable frequency, an external crystal was used. In the microcontrollers datasheet[22], the crystal setup is described. Two capacitors are used for the crystal connection, with the values given in the datasheet. Also, two capacitors are placed close to the microcontroller in order to ensure stable voltage.

After the components were selected, it was followed up by a logical connection of all the electronic parts. A schematic drawing was created using the freeware edition of Eagle by CADSOFT. In the schematic drawing all the components are added to a blank drawing and then connected logically. Common components exist in a library, whereas the sensor had to be drawn manually with the correct size and input ports. The logical connections are determined by looking through datasheets and deciding which pins to connect to each other. The attitude estimation requires few components, and is fairly easy to connect. The methods are implemented on a microcontroller. To make the prototype as portable as possible it is designed with an USB connection for the computer. The attitude data is transferred from the prototype to the computer so the user can view the attitude live by opening a simple MATLAB file. Since the prototype is made for experimental use only, it will always be connected to a computer during use. The prototype can operate with voltages between 3.3V and 5.5V. The USB connection provides 5V. Hence, no power sources are implemented on the device as it will be powered up through the USB connection. When designing the attitude estimator for use in the CUBESAT, the power will be provided from a battery within the satellite.

The acceleration, change in velocity and the magnetic field are measured by an IMU connected to the microcontroller using RX/TX pins. A 10-pin header (JTAG) is connected to the prototype making it possible to program it. A JTAG

connection is a standarized connection for programming and debugging an embedded system. Figure 4 shows the prototype connected to the AVR Dragon kit using a JTAG connection, while powered up through the USB connection.

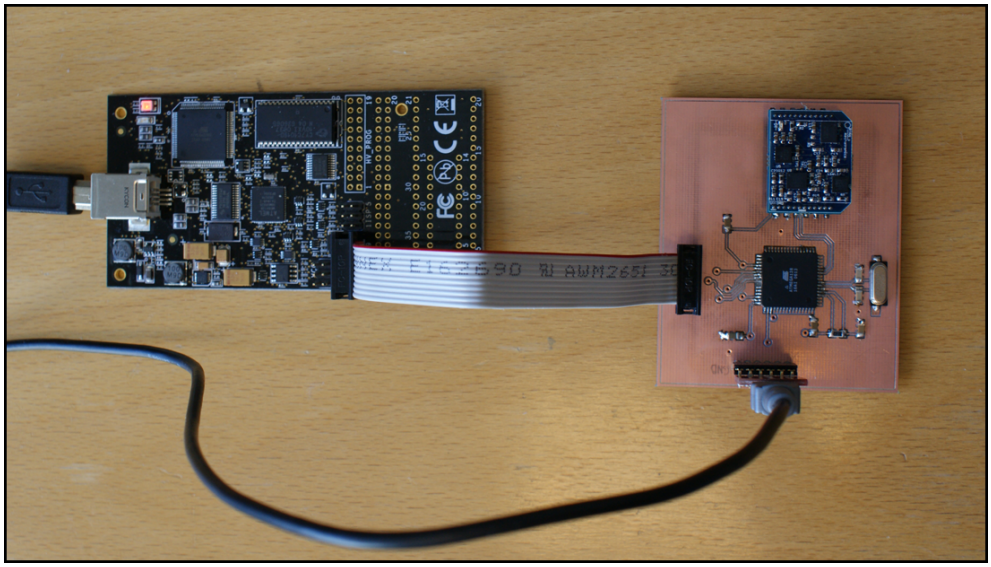


Figure 4: Picture of the prototype connected to an AVR Dragon.

Further, a stable 16 MHz oscillator is provided by an external crystal. Using a crystal is a very common way to provide a more accurate frequency than what the microcontroller can provide by itself. The logical connection with all the components used is showed in Figure 5. The capacitors powering the crystal was chosen to be $12pF$, selected from the recommended values in the microcontrollers datasheet (Recommended between 12-22pF). To avoid unintentional resets, it is recommended that the reset pin, on the microcontroller, is connected. According to the datasheet, the resistor should be at least $10k\Omega$. Hence, the resistor connected to the reset, was selected to be $10k\Omega$. By connecting a capacitor between the RESET pin and ground, the RESET is protected from noise. This capacitor was chosen to be $100nF$. Further, three capacitors of $100nF$ were placed close to the microcontroller in order to provide stable voltage.

The software was first designed and implemented on an ATMEGA 2560 microcontroller connected to an AVR STK600. This controller has 100 pins, making it very hard to solder on a circuit board. Hence, another microcontroller was used for the prototype. Two different prototypes were designed, with the only difference being the choice of microcontroller. During the first prototype design, an ATMEGA 128 microcontroller was used. The ATMEGA 128 has the same kernel as the ATMEGA 2560 and only 64 pins, making it easier to solder on the circuit board. The change of microcontroller required minor changes

in the software - only register names needed to be modified. Even though the ATMEGA 128 has the same kernel as the ATMEGA 2560, it has only 128 kbytes flash memory while ATMEGA 2560 has 256 kbytes. This turned out to be a major problem, as the microcontroller kept resetting due to stack overflow. The EQUEST algorithm was able to run without any problems, but the significantly more computational demanding EKF resulted in memory errors. Therefore, the second prototype was designed with an ATMEGA 2561 microcontroller. The ATMEGA 2561 has the same kernel, and the same memory as ATMEGA 2560, only with 64 pins. In the second prototype, both methods managed to run simultaneously without any complications. Hence this prototype was used for further testing.

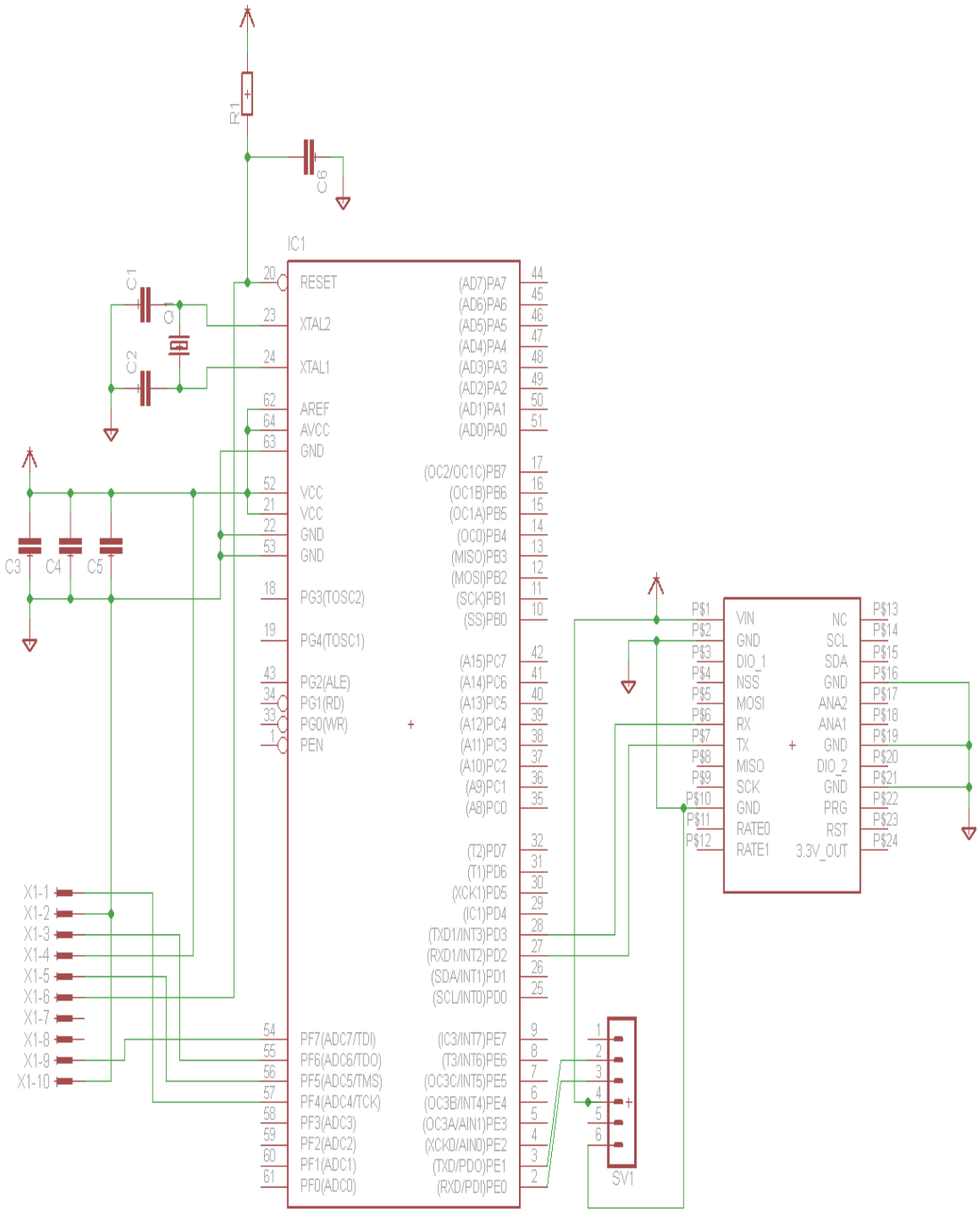


Figure 5: Schematic drawing of the prototype.

Based on the schematic drawing, the board was designed using same software from CADSOFT. All the required components were automatically exported from the schematic drawing, but the physical connections had to be drawn manually. However, the program provides helping lines to determine which pins to connect to each other. The board drawing gives a physical connection between all the components. In most cases some of the paths will overlap. Hence, at least two layers are required. In Figure 6 the blue lines represent a physical connection on the bottom side, whereas the red lines represent a path on the top side of the board. The green circles are vias. A via is a hole in the board, connecting paths at the top with those at the bottom.

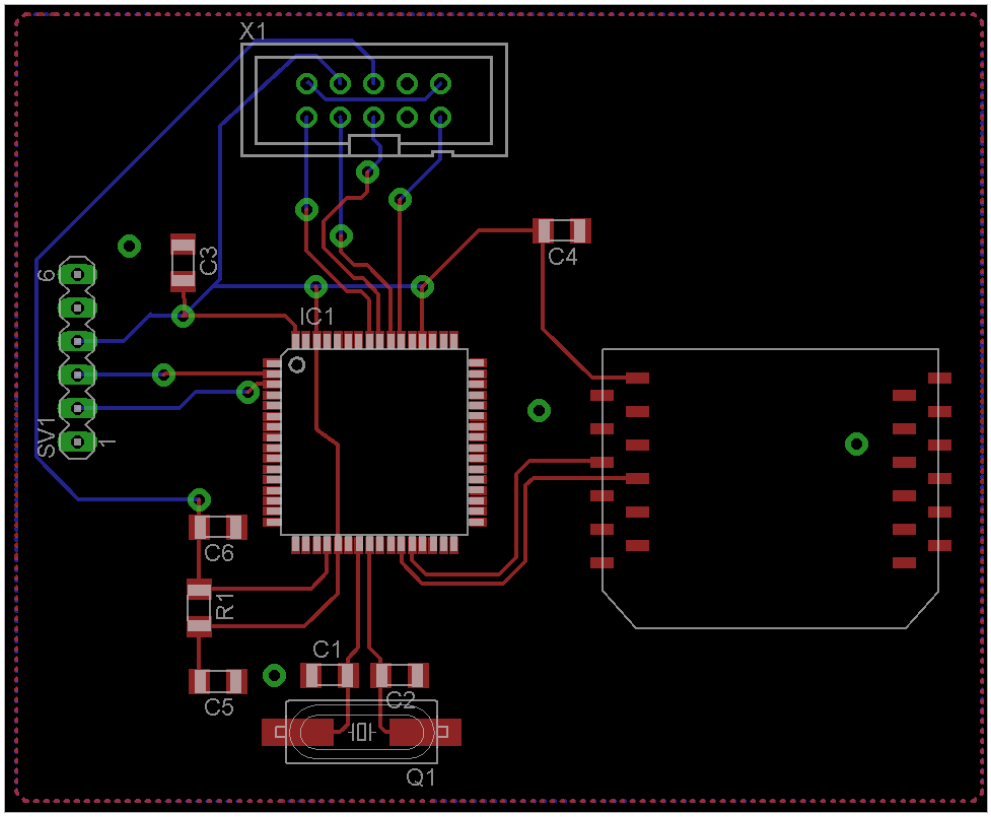


Figure 6: Board drawing of the prototype.

The milling is done using a circuit board plotter from LPKF. The board drawing was exported to the belonging software of the LPKF, and the machine creates a circuit board. When the milling is done, all the vias must be stamped to provide an electrical connection between the two sides.

With all the paths milled into the circuit board, the components can be soldered to the board. This is a process that requires very high precision and

a steady hand. The size of the prototype is showed in Figure 7, here a ruler is placed next to the prototype for comparison.

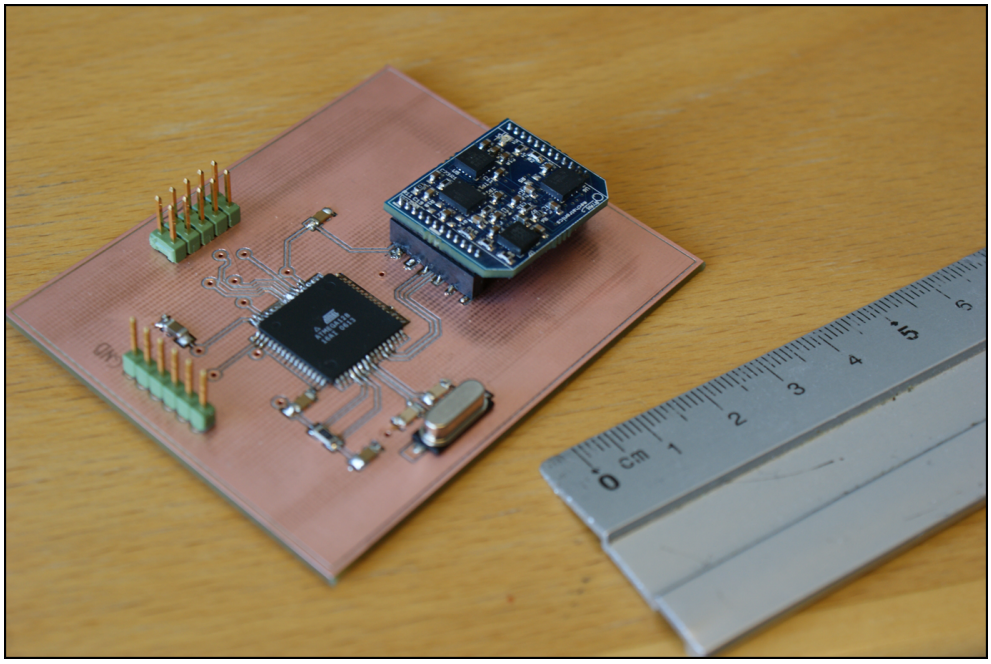


Figure 7: Picture of the attitude estimation prototype.

As mentioned, the prototype is programmed using the JTAG connection. Since an AVR microcontroller is used, the programming is done using an AVR Dragon development board. This board will not power up the unit through the JTAG connection, hence the USB must be connected while programming. Note that some development boards, such as AVR STK600, have power connected to the JTAG. If such boards are used, the USB header should not be connected! This may damage the development board. The complete connection for programming is showed in Figure 4. Under testing the AVR Dragon board will be disconnected, but the USB connection will remain connected. A detailed description of how to use and program the prototype is given in Appendix F.

8 Simulations and Tests

8.1 Testing the Kalman Filter in MATLAB

In the first simulation, the Kalman filter was implemented in MATLAB and compared with estimations done by a sensor (Xsens MTi). Xsens MTi is an attitude and heading reference system and can provide raw data output from gyroscope accelerometer and magnetometer. It can also provide attitude quaternions using a built-in extended Kalman filter. The Xsens sensor was connected to the computer using an USB connection. By changing the orientation of the sensor, data were created and sent to MATLAB. Both the raw data from the IMU and the attitude estimation done by Xsens were recorded. The recorded sensor data were used in the Kalman filter in MATLAB and compared with the recorded estimated attitude. As observed from Figure 8 the Kalman filter estimations in MATLAB are very similar to those done by Xsens. The estimated biases for magnetometer and accelerometer are given in Figure 9.

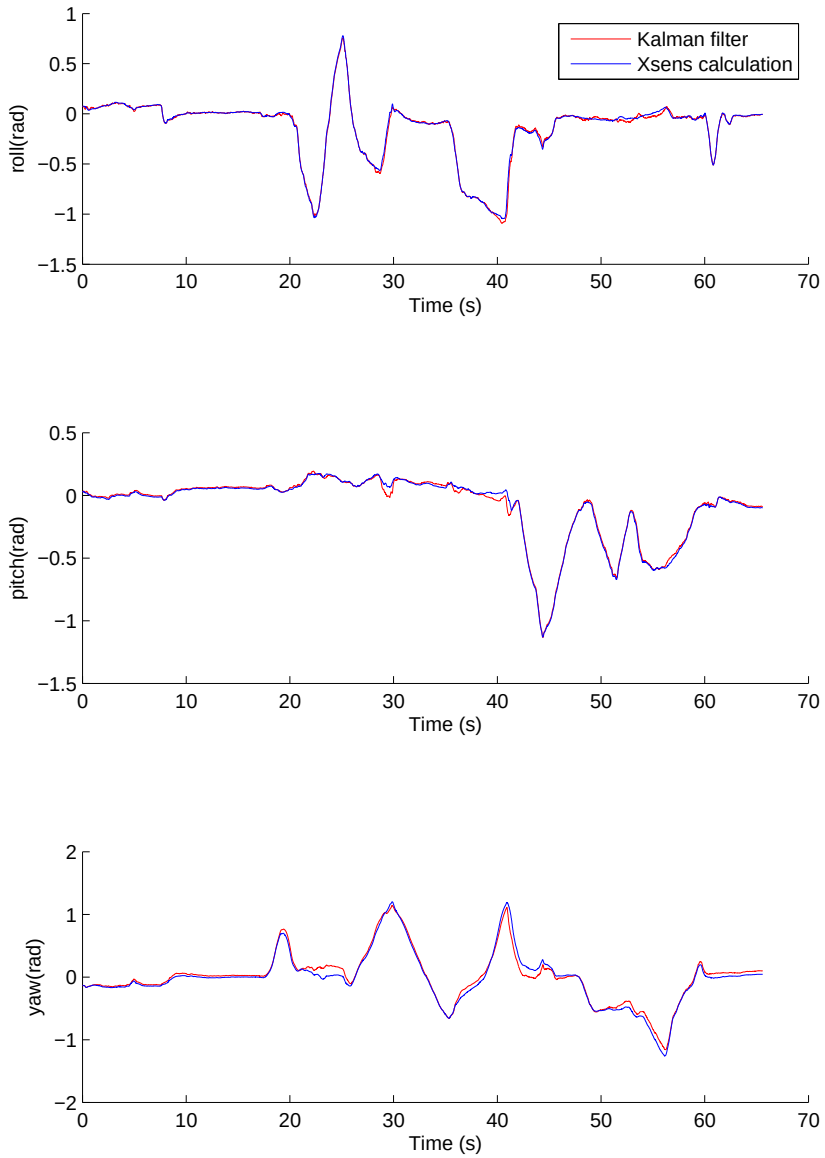


Figure 8: Attitude estimation using Kalman filter in MATLAB.

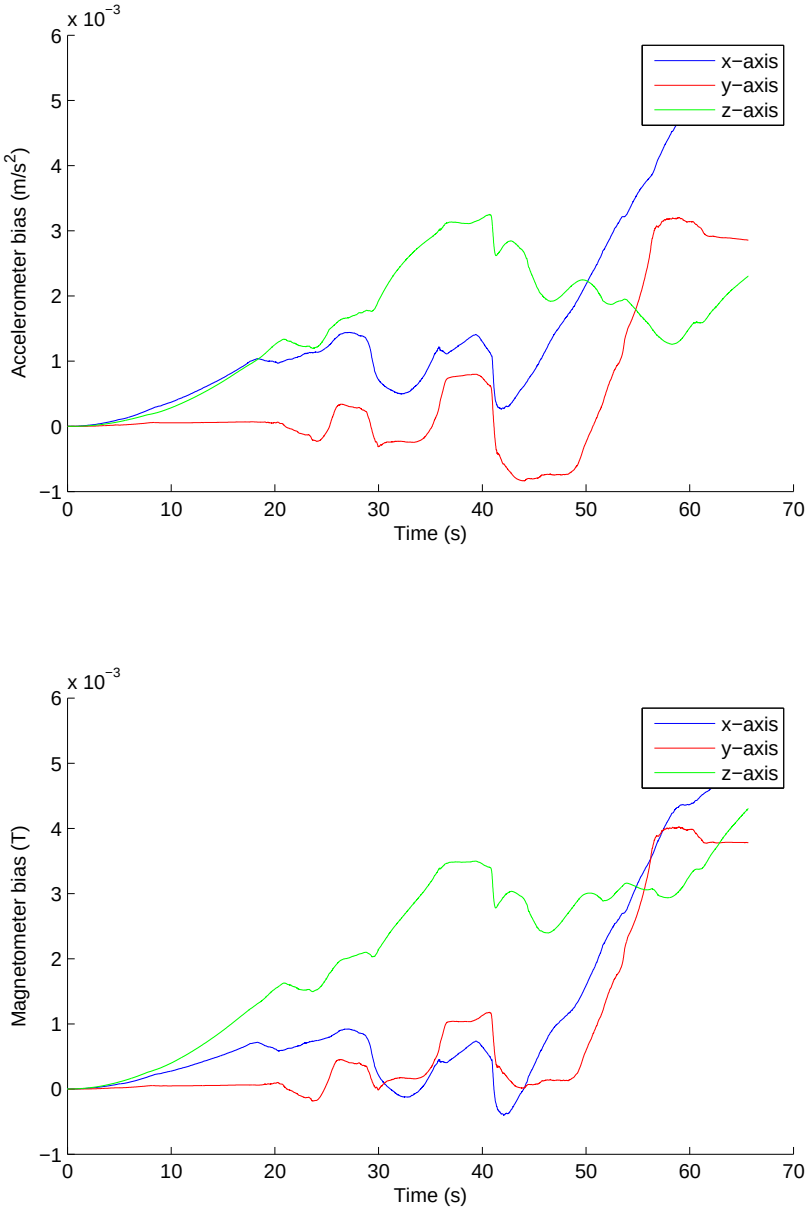


Figure 9: Accelerometer and magnetometer bias.

8.2 Testing the EQUEST in MATLAB

This test is is similar to the one done with the Kalman filter. Raw data from Xsens were sent to EQUEST, and attitude for each time step was calculated. The estimated attitude was compared with the orientation calculated by Xsens (See Figure 10). As observed from the figure, attitude estimated by EQUEST is close to the attitude determined by Xsens.

Note that the deviation in the methods, displayed in Figures 8 and 10, could either be results of estimation errors in the XSENS or in the developed methods. However, the observed erorrs are insignificant, indicating that the methods developed in this paper are able to estimate the orientation.

The linear prediction term will have a smoothering effect on the EQUEST estimations. Figure 11 shows the difference between EQUEST with and without a linear prediction term. The figure clearly illustrates the filtering effect of the linear prediction term. This is especially observed in the yaw-angle. The noise in the yaw-angle, mainly caused by magnetic field fluctuations, is reduced by the prediction term.

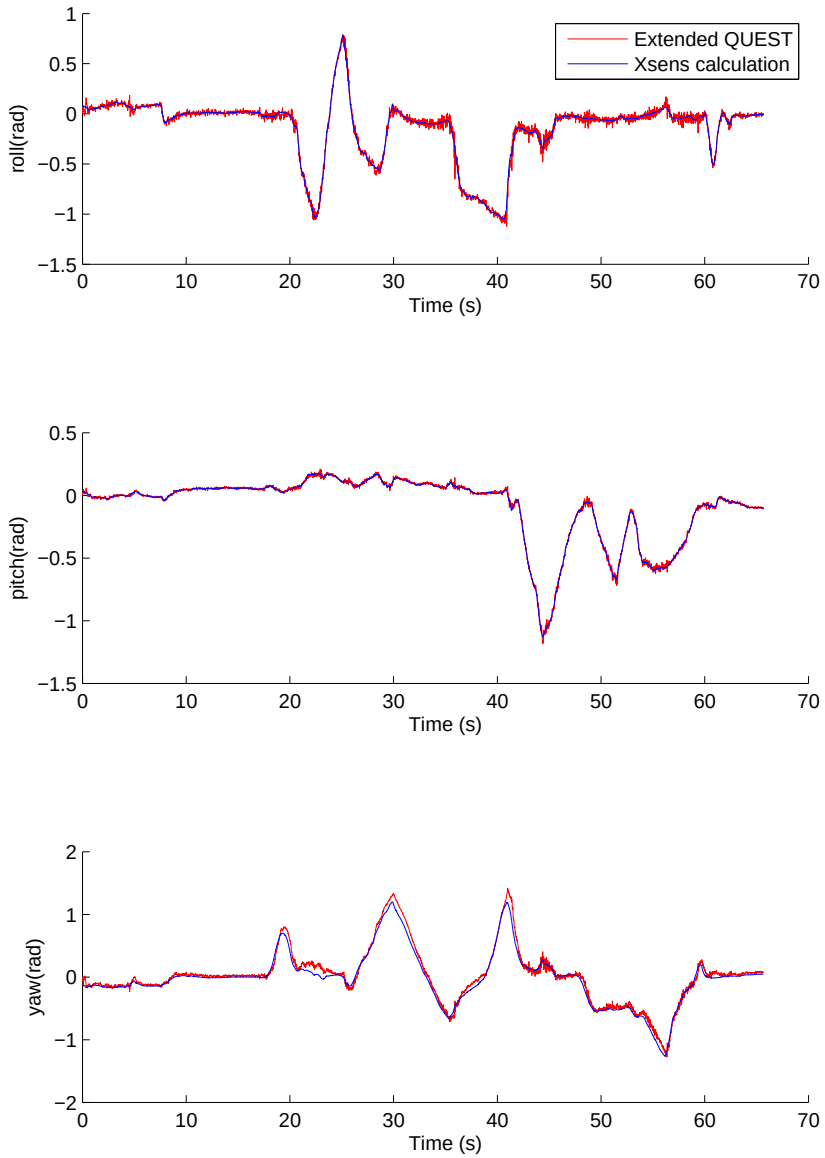


Figure 10: Attitude estimation using EQUEST in MATLAB.

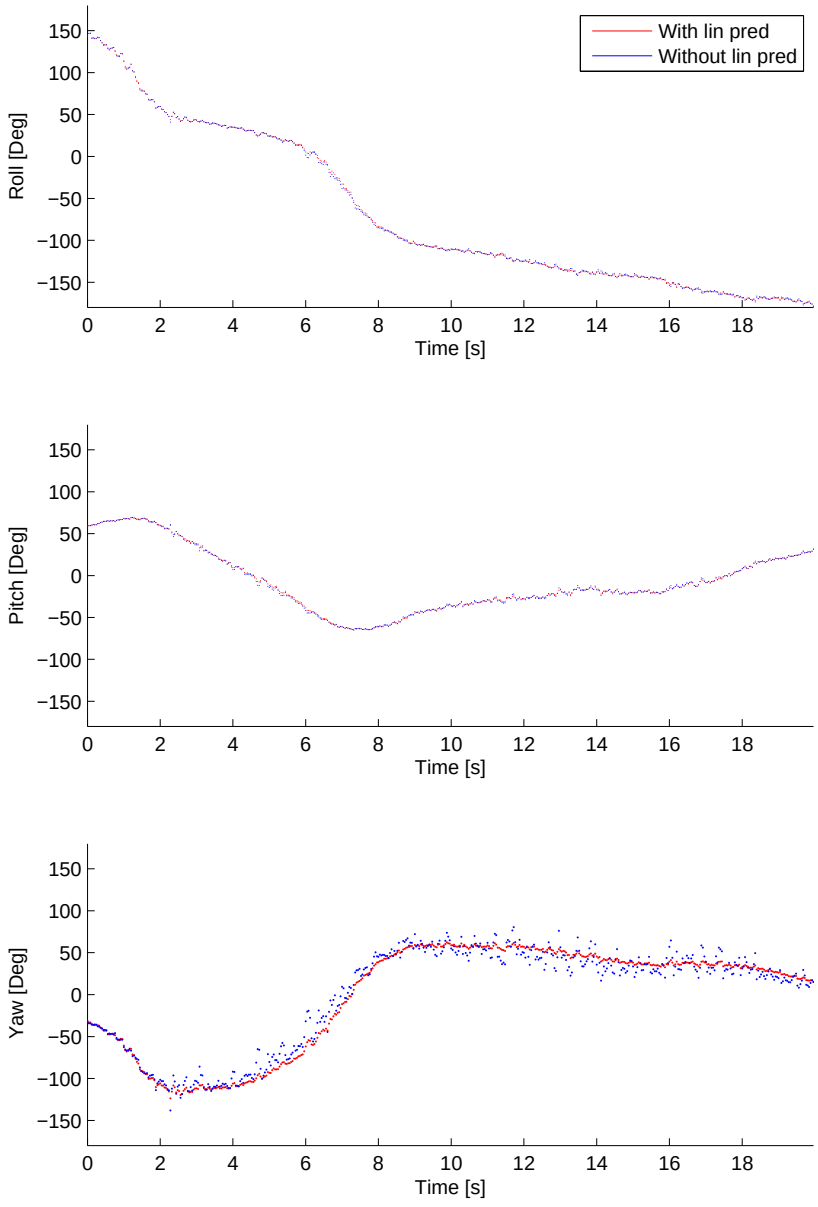


Figure 11: EQUEST with and without prediction.

8.3 Real-time Attitude Estimation Using EKF and EQUEST Implemented in MATLAB

In order to test the real-time performance of the estimation methods, an inertial measurement unit, 9-DOF Razor IMU, was connected to the computer using an USB connection. The IMU contains 3-axis accelerometer, 3-axis magnetometer, 3-axis gyro and is illustrated in Figure 12. Through the USB connection, raw data are sent continiously from the IMU to MATLAB. The raw data contains the measured acceleration, magnetic field and the rotational velocity. By using this data as input for the algorithms designed in MATLAB, the attitude can be estimated in real time. This means that the methods will estimate the orientation of the sensor continiously while it is rotated. A MATLAB program was written to convert the attitude quaternions to a 3D plot of a cube with the estimated orientation. This is a user-friendly way to observe the estimation methods' ability to track rotations.

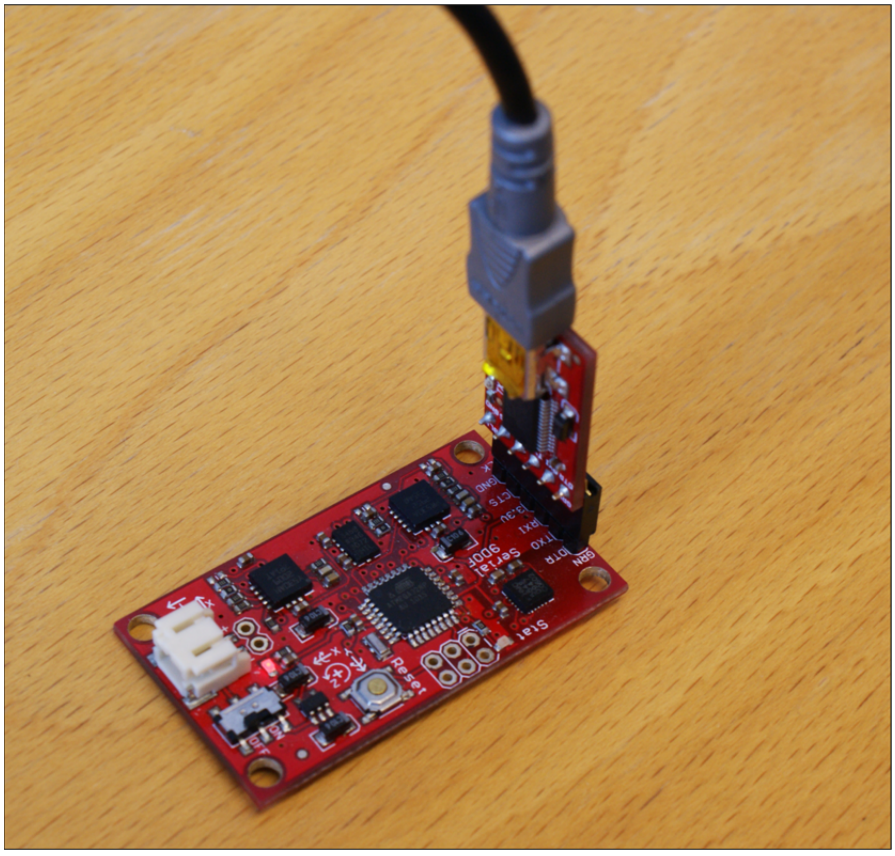


Figure 12: 9-DOF Razor IMU.

To evaluate the performance of the two methods with each other, data were collected from the IMU and stored in a .txt file. Both methods were implemented in the same code in MATLAB, and data were read from the text file. The estimation of both algorithms are showed in Figure 13. The figure shows that both algorithms estimate the approximate same values for pitch, roll and yaw angles. The largest difference can be spotted during the startup phase, where it can be observed that the EKF is lagging behind the EQUEST. If the startup is disregarded, the largest deviation between the two methods was measured to be less than 10°. In this test, the EQUEST was implemented without a prediction term, implicating a higher vulnerability to disturbances.

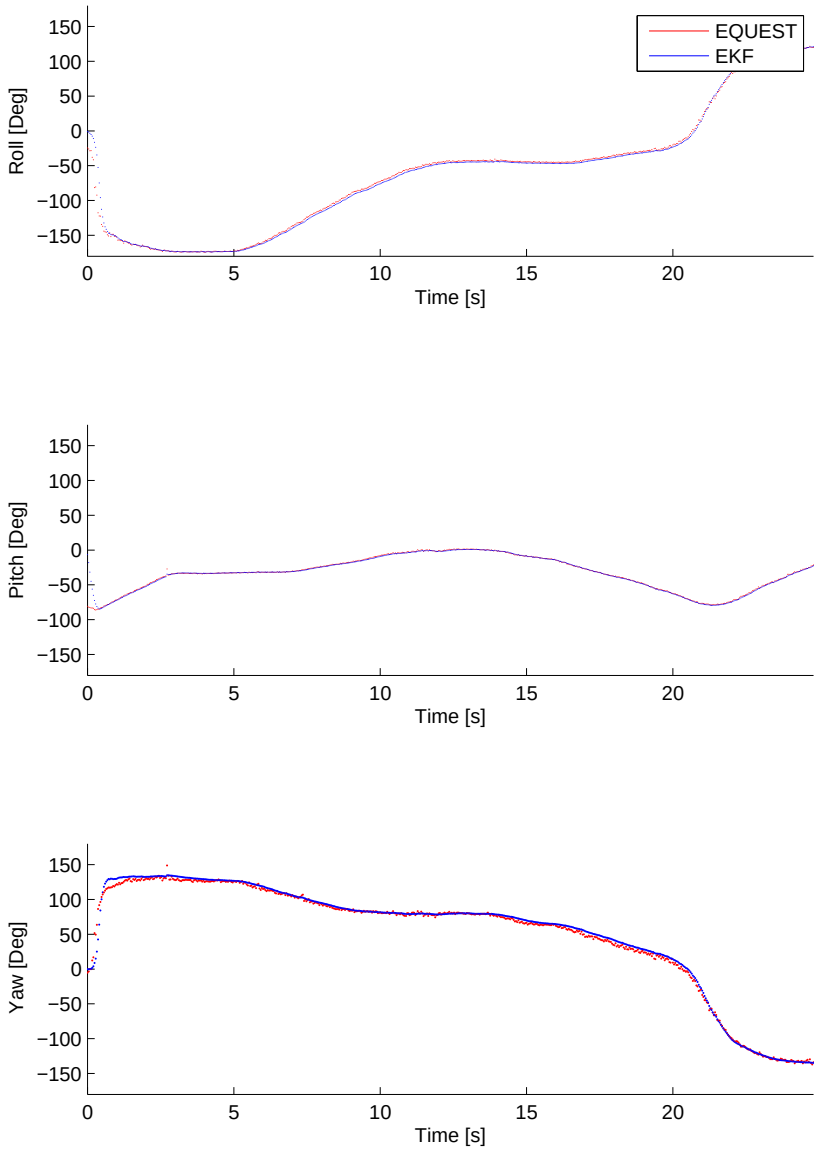


Figure 13: Attitude estimation using EQUEST and EKF in MATLAB.

8.4 Real-time Attitude Estimation EKF and EQUEST Implemented on AVR Microcontroller

To be able to estimate the attitude in space, the EKF and EQUEST must be implemented on a microcontroller. A microcontroller with at least two communication ports is necessary. One port for communication between the IMU and the microcontroller, and the other for communication between the computer and the microcontroller. For the final product, there is no need for communication with a computer, but the second communication port is still needed for communicating with the rest of the satellite. Communication was done using universal asynchronous receiver/transmitter (UART), both with the sensor and the computer. An ATMEGA2560 microcontroller from Atmel was found to be appropriate for testing. This microcontroller was connected to a STK600 development board. The connections between the microcontroller and all other external units goes through the development board. The board provides easy connection to external units using headers, with no need for soldering. Both the sensor and the computer were connected to the microcontroller the connections provided by the development board.

A schematic setup of all the connections are given in Figure 14. Note that, in this test, both methods were implemented on the same microcontroller, but one at a time. Both methods were able to track the rotation of the sensor displayed with the 3D cube drawn in MATLAB. However, this gives no foundation to compare the methods with each other. In order to compare the algorithms, both methods must run simultaneously or with the exact same data input. For instance, a sample data must be found by the sensor. Then the sample data can be used as input for both algorithms.

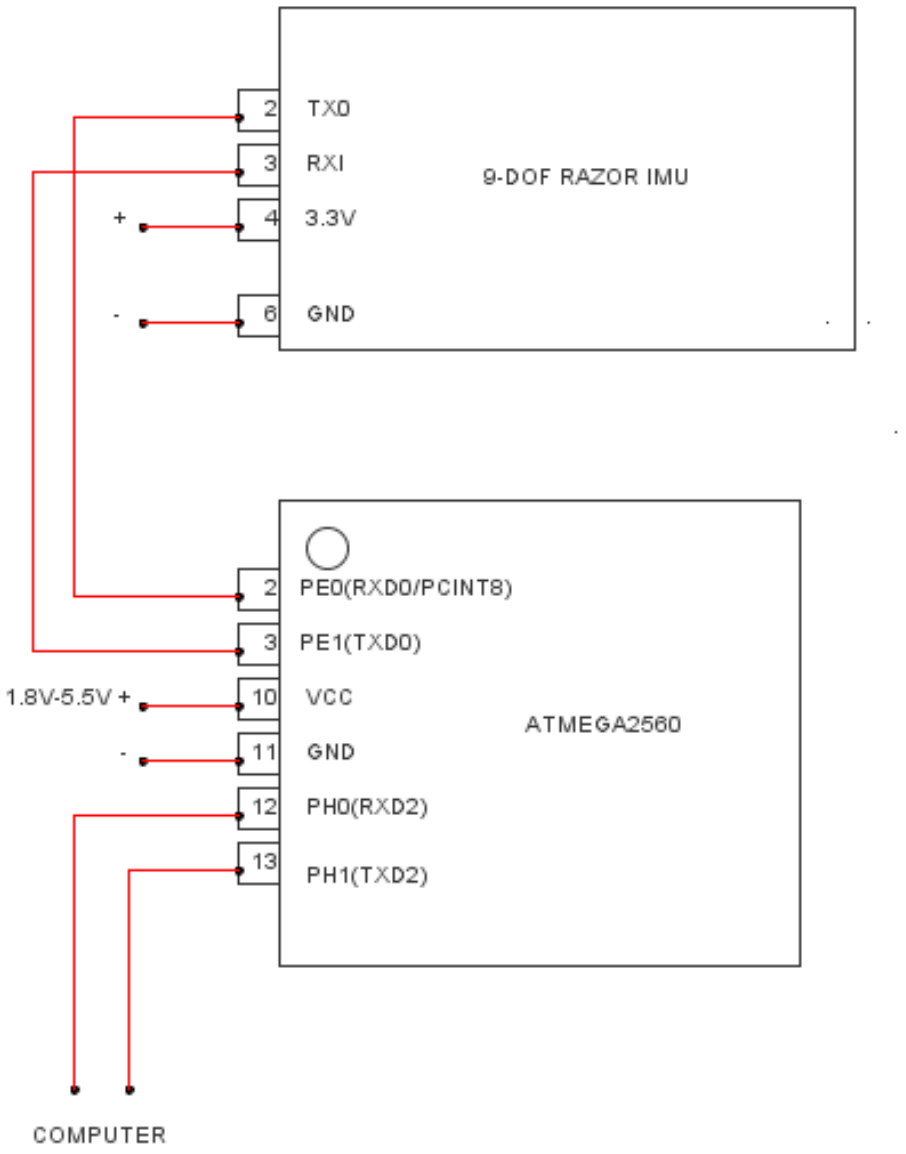


Figure 14: Schematic: Connecting the microcontroller to IMU.

8.5 Comparing EKF With EQUEST Implemented on AVR Microcontroller

In order to compare the two algorithms, both were implemented on the same microcontroller. They both ran simultaneously and by extracting data to a computer, it was possible to compare the performance. The first difference in the two algorithms can be observed in the start-up phase. The start-up is showed in Figure 15. As the figure indicates, the EQUEST has a fast settling time while the EKF uses some seconds to identify the correct orientation. However, both algorithms settles on the same position in the end. EQUEST calculates the final attitude in one iteration. Hence a big jump can be observed between the inital state and the first calculated attitude. Be aware that the EKF and EQUEST had different initial values, explaining the deviation in roll value at time 0.

It is noteworthy that the Kalman filter takes so long to find the correct yaw value. This is partial due to the testing being done in Trondheim, Norway. At this location the normalized magnetic field vector has the value

$$B = \begin{bmatrix} 0.2631 & 0.0086 & -0.9647 \end{bmatrix} \tag{82}$$

while the normalized gravitational vector in NED is

$$g = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{83}$$

It is easy to see that these two vectors are close to parallel. A rotation around the down-axis in the NED frame will be hard to identify as neither the magnetic field vector, nor the gravitational vector change significantly in body coordinates.

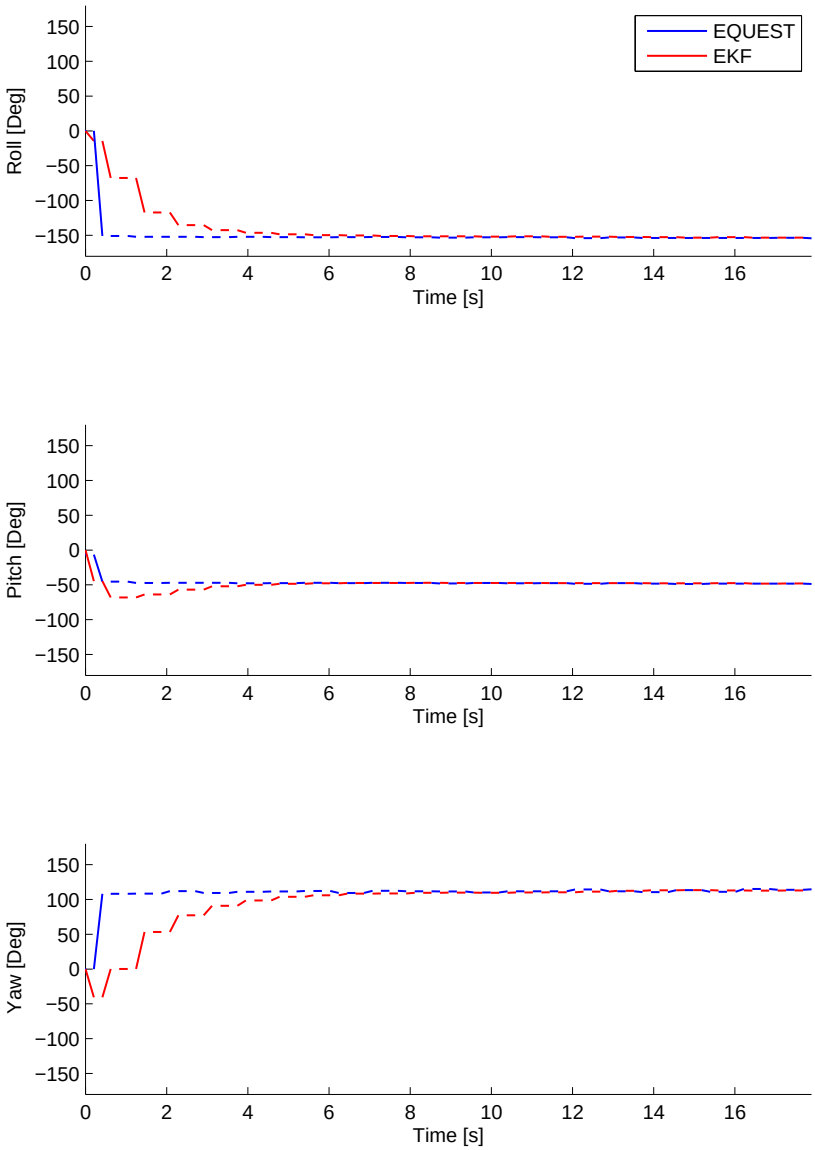


Figure 15: Start-up phase of the two algorithms.

The large time difference in identification of the yaw angle indicates that the two methods should perform quite dissimilar when tracking a rotation around the down-axis in the NED frame. Testing illustrated in Figure 16 strengthen this statement. This figure was taken after both methods had settled, and then a rotation around the down-axis in the NED frame was performed. As the figure clearly shows, EQUEST is able to track the motion far better than the EKF. The reason for the Kalman filters fairly slow estimation of the yaw rotation can be explained mathematically through the rotational matrices. The rotational matrix for rotations strictly around the down axis can be written as

$$R_{down} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{84}$$

Remember that the state update in EKF is given as:

$$x_{k+1} = x_k + K(z_k - \begin{bmatrix} Rg^\top \\ RB^\top \end{bmatrix} - \begin{bmatrix} Bias_a \\ Bias_m \end{bmatrix}) \tag{85}$$

Here, the state update is a product of the Kalman gain, K, and the deviation from the measurements tracked by the rotational term. A rotation around the down axis is described above. Since the gravitational vector is close to parallel with the magnetic field, the rotational term in Equation (85) will now be described as

$$Rg^\top = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{86}$$

$$RB^\top \approx \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \tag{87}$$

Clearly, the rotational term fails to identify the rotational angle around the down-axis due to the magnetic field being parallel with the gravitational vector. As the measurements are B and g, there will be very small deviations from the measurements to the rotated position.

As Figure 17 indicates, the EKF will be slower than the EQUEST when tracking a rotation. However, both algorithms will be able to estimate the attitude. Again, the greatest deviation is observed in the yaw angle, with a maximum of almost 50°.

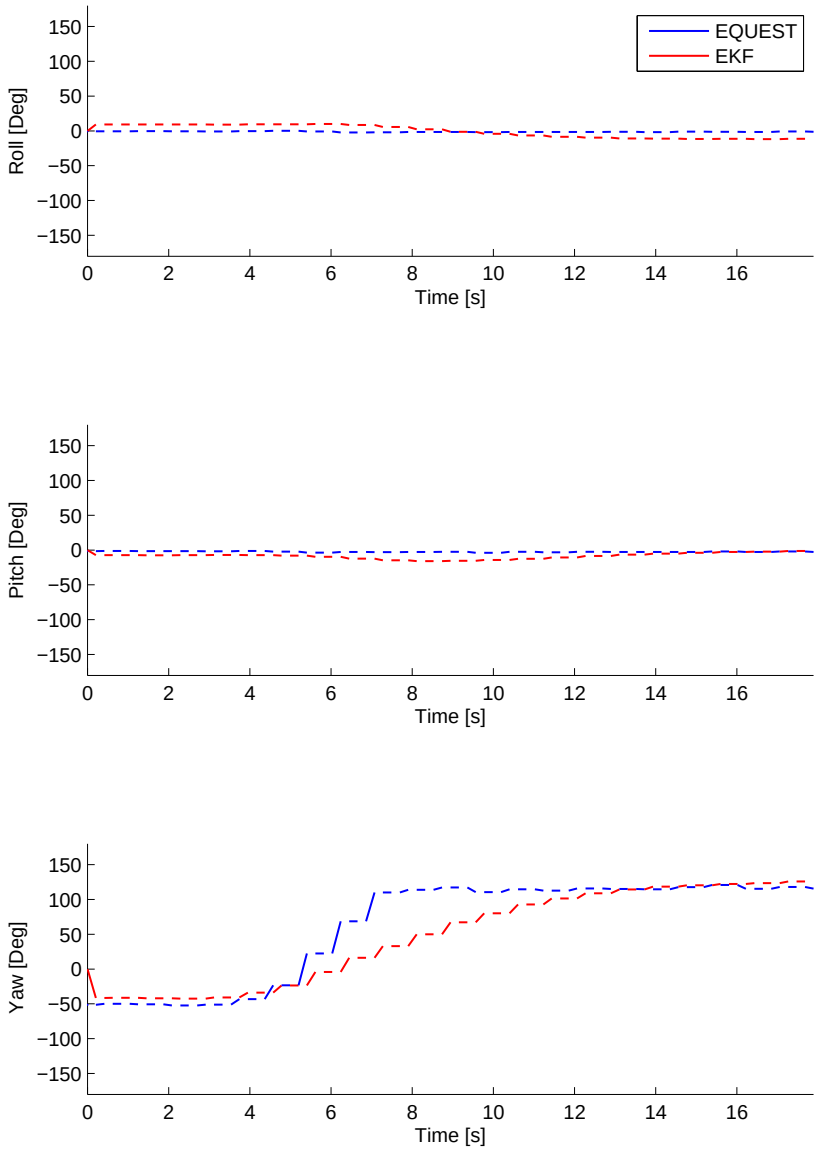


Figure 16: The two methods response to a rotation around the down-axis in the NED frame.

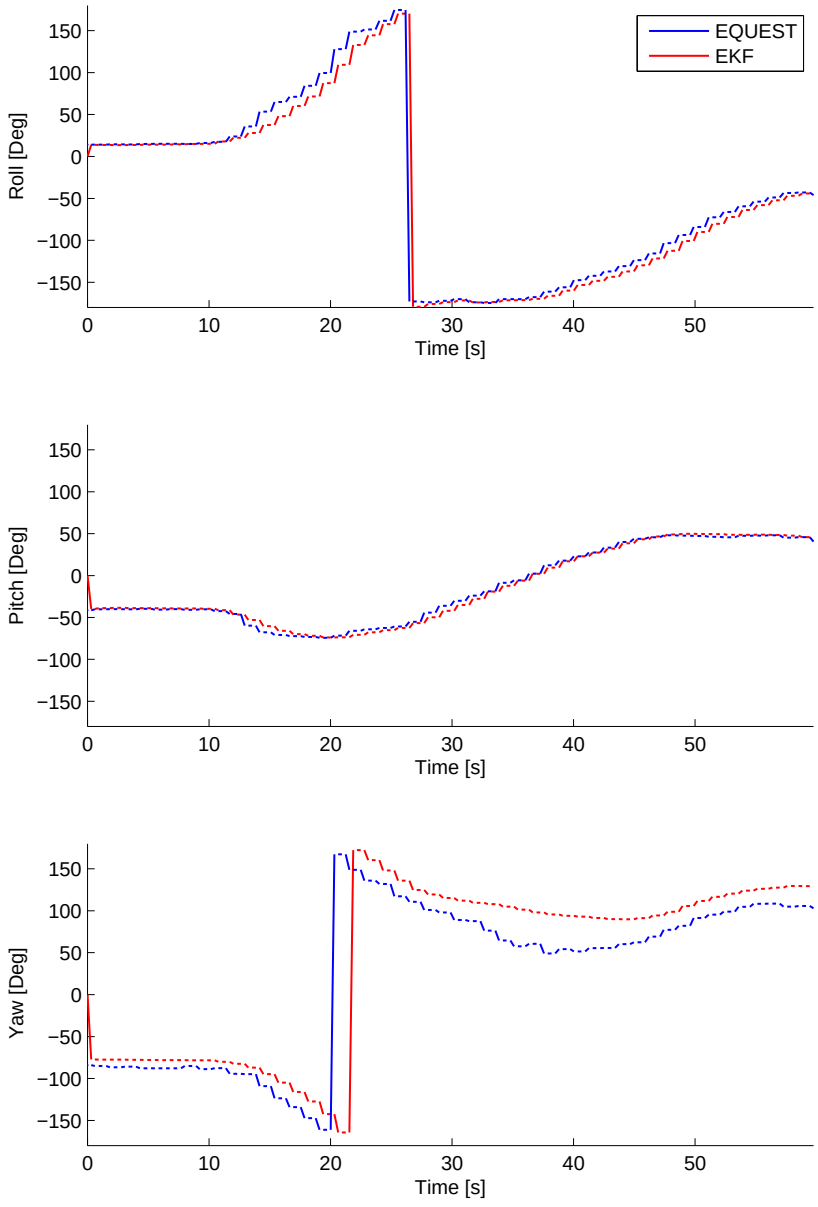


Figure 17: The two methods response to rotations.

8.6 Testing the Prototype

The first test of the prototype was a visual test performed by making a drawing at the computer. The prototype estimates the attitude, and sends the estimated quaternions to the computer through the USB connection. The quaternions received by the computer is handled by a small MATLAB program, which plots a 3D drawing of a cube on the screen in real time. The Figure 18 illustrates the first testing. Note that this test does not assure that the attitude is estimated correctly, but it gives a fine indication that the algorithm is able to track the rotations.

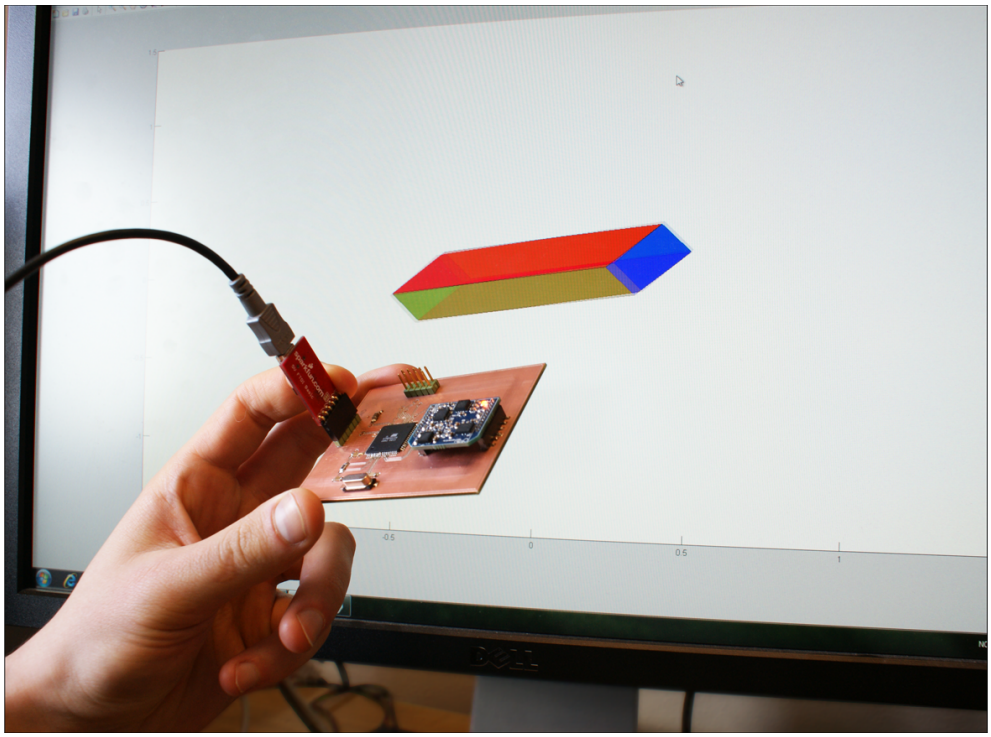


Figure 18: Testing the prototype while plotting the attitude live on a computer.

The second test is done to assure that the estimation methods are able to calculate the correct attitude. Since both algorithms were implemented on the same microcontroller, they were also able to run simultaneously. This makes the prototype testing easier, as one test gives results for both methods. The first test indicated that the prototype was able to track rotations. However, it does not say much about the accuracy of the estimates. In order to test the accuracy of the algorithms, the prototype was strapped to an ABB IRB 140 industrial robot. This is a programmable robot arm with 6 degrees of freedom. The IRB 140 is shown in Figure 19, were all the six joints are numbered.

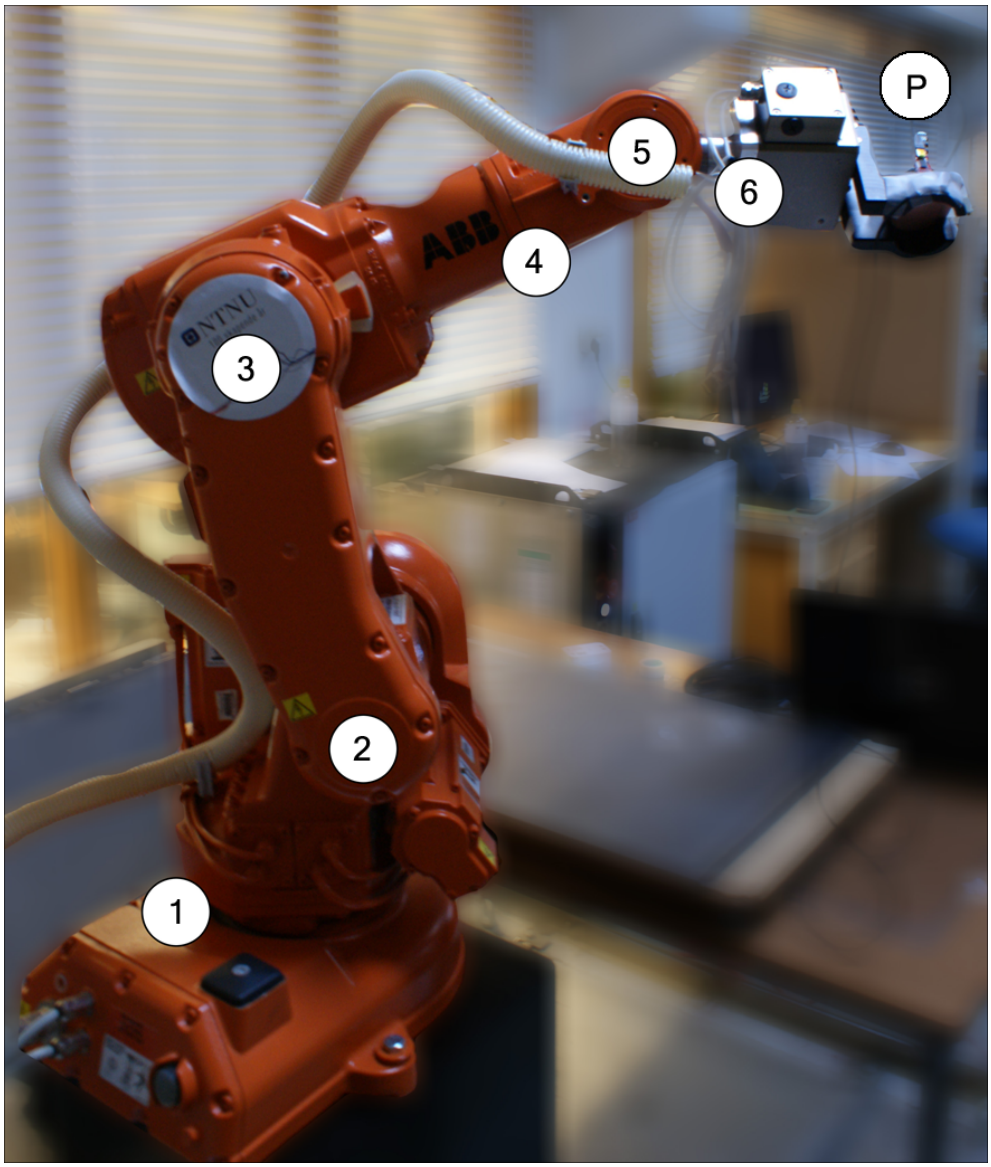


Figure 19: The ABB IRB140 with all the joints numbered. The prototype is strapped to the robot, marked with the letter P.

Here the prototype is strapped to the robot and marked with the letter P on the figure. The robot does not provide an output for the true path followed. In order to test the accuracy of the algorithms, the robot was programmed to make one rotation at a time with a given angle. First the entire robot arm was aligned such that a rotation in joint 5 would perform a pitch rotation for the

prototype, and a rotation in joint 6 would perform a roll rotation.

Figure 20 shows the output from the prototype when the robot makes a 90° rotation in joint 6, waits for 8 seconds and then rotates back to the initial position. As the figure indicates, there is no movement in neither pitch nor yaw when the robot performs the rotation in joint 6. Also, the figure suggests that the EQUEST is slightly faster than the EKF. This has been showed in all the previous tests, and due to the mathematical properties of the two algorithms it is reasonable to draw this conclusion. Another advantage of this test, is the robots ability of providing a constant rotational velocity and smooth movements. By selecting a fairly low speed, realistic movements for an object floating in space can be simulated.

It is very hard to observe the accuracy of the two algorithms from the figure below. Figure 21 shows the robot performing a 90° rotation in joint 6, before making a 60° rotation in joint 5. The results from the EKF is marked. The figure clearly indicates that the EKF is a bit inaccurate in both the roll and pitch measurements. When the robot performed a 90° roll rotation, the EKF estimated a rotation of 98.5°. This gives an error of almost 10%. For a 60° pitch rotation, the EKF estimated a total rotation of 71.5°. This is an overshoot of 11.5°, with a total error of more than 19%.

Output for the exact same test, only EQUEST values marked, is illustrated in Figure 22. As can be read out of the figure, the estimated roll with the EQUEST method is about 89°. This gives an error slightly above 1%. For the pitch maneuver the EQUEST estimated a total deflection close to 57°. This gives an offset of 3°, or 5%. These results indicates that the EQUEST algorithm provides quite accurate estimates, while the EKF struggles with a small overshoot.

The big deviation for the EKF can be caused by several factors. The industrial robot creates a large local magnetic field, disturbing the magnetometer measurements. This might affect the EKF through the tuning parameters. Ideally the EKF is tuned by indentifying the measurement uncertainty and the model uncertainty. Obviously, the uncertainty of the magnetometer measurement is highly affected by the fluctuations in the local magnetic field. Hence, the magnetic disturbance from the robot will directly affect the tuning of the EKF.

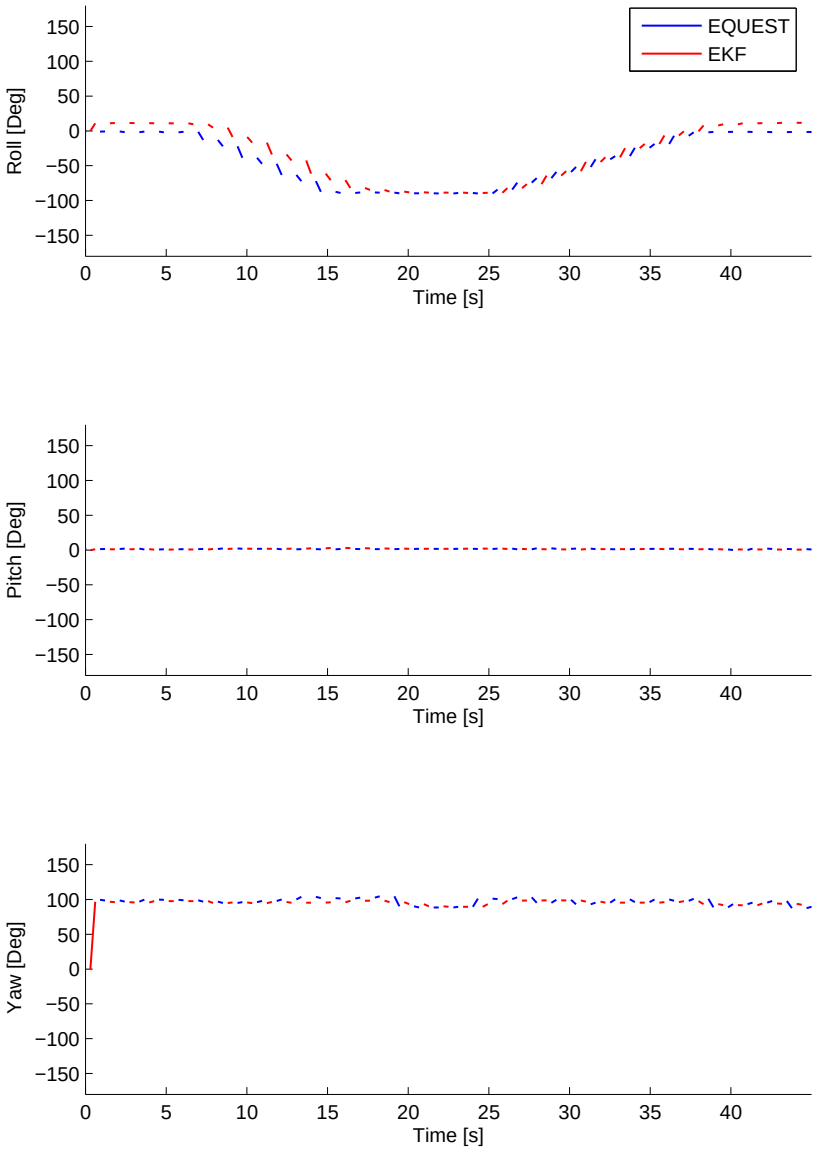


Figure 20: The robot performs a maneuver making the prototype roll.

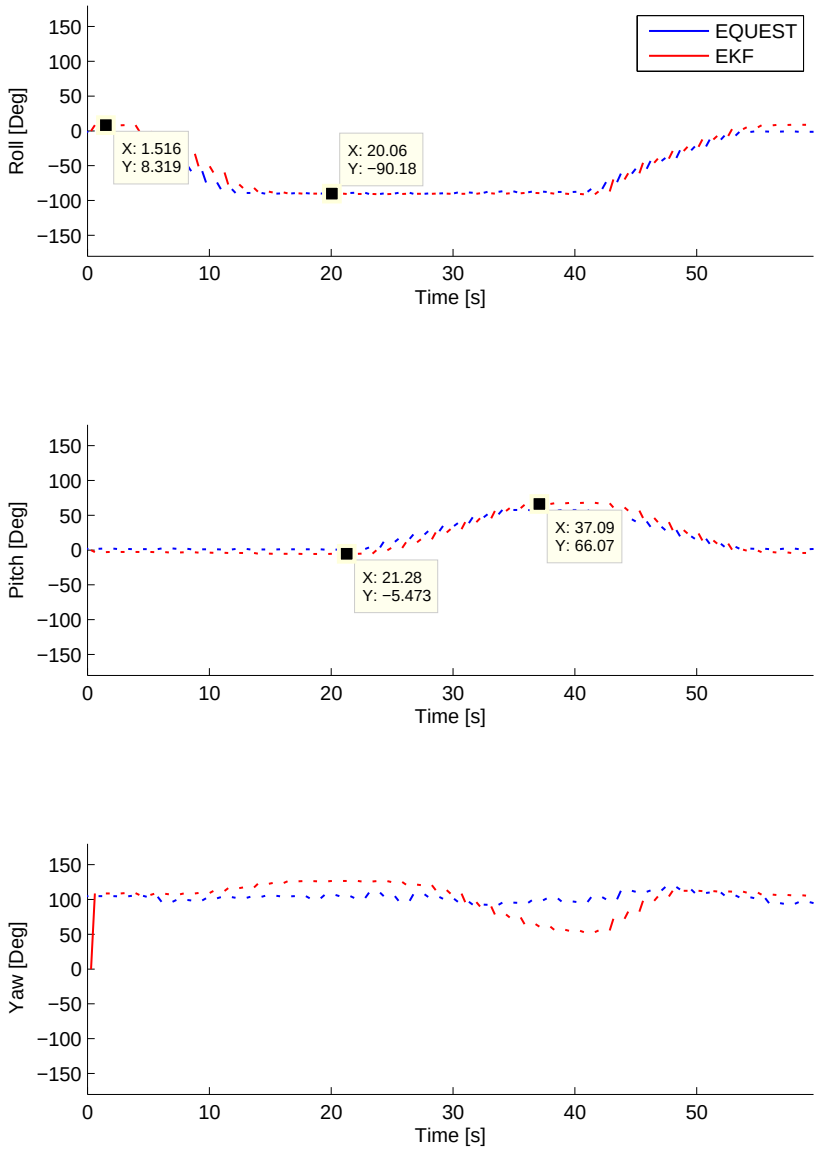


Figure 21: The prototype output with a rotation in both roll and pitch with the EKF marked.

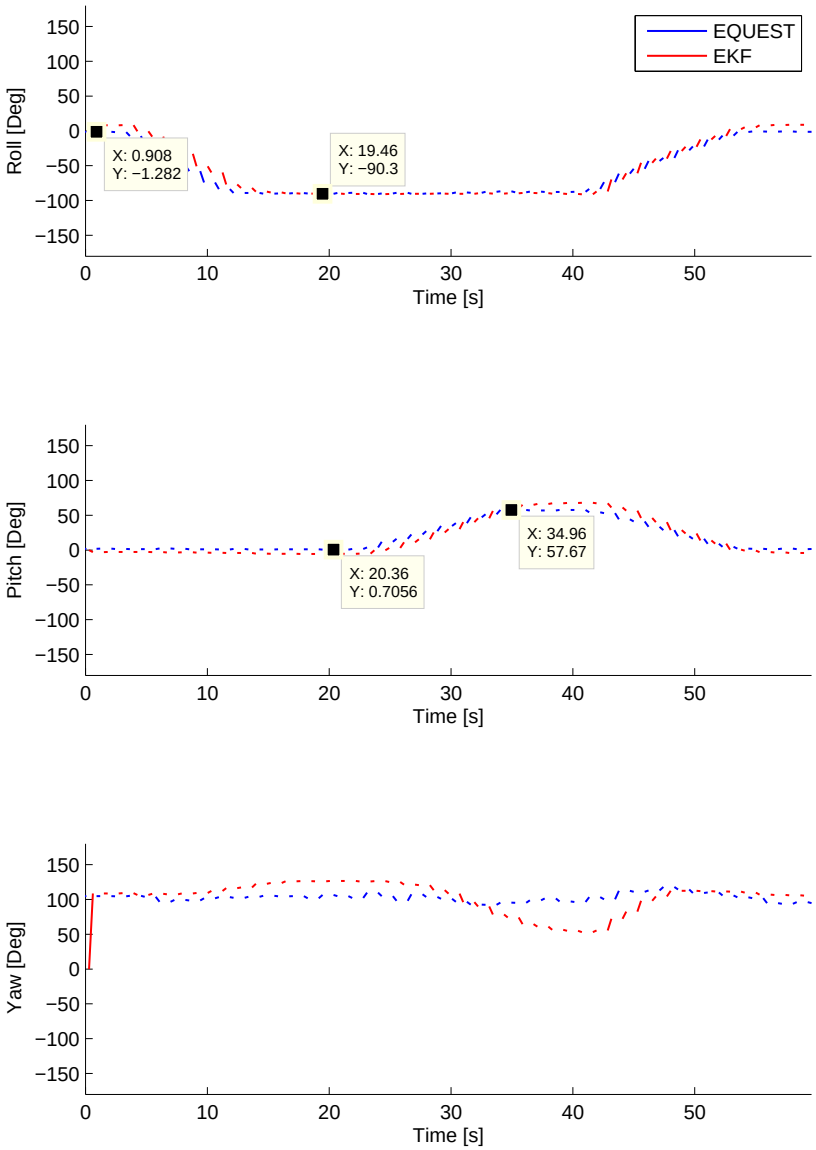


Figure 22: The prototype output with a rotation in both roll and pitch with the EQUEST marked.

Several rotations among the yaw angle were also performed, with various results. A yaw rotation is mostly measured by a change in the magnetic field. There is a high concentration of electronic parts in each joint of the test robot, as well as a high voltage power supply cabin close to the robot. This gives the local magnetic field frequent- and variable fluctuations while the robot is moving. Hence, it was impossible to conclude from the tests among the yaw angle, and the results are omitted. The figures from the tests with the robot indicate fluctuations as the EKF and the EQUEST estimates different yaw rotations. As indicated in Figure 23, this problem was not observed during tests with low magnetic field disturbance. These tests are done by rotating the prototype far away from electronic components. Instead of using the robot to get accurate rotations, the prototype was rotated by hand. When the prototype is still the estimated yaw angles are close to equal, as indicated around time t = 60. However, the EKF has problems tracking fast changes in angles.

One of the properties with the EKF, is the ability to filter the output. The linear term in EQUEST provides similar properties, but heavy filtering will reduce the accuracy of the estimation. To test the filtering, the prototype is exposed to both moderate and heavy shaking. This is also done by hand and is given in Figures 24 and 25. During shaking, the filtering properties of the EKF clearly reveals itself. This is specially observed during heavy shake testing. The EQUEST algorithm provides acceptable filtering during moderate shaking. Unfortunately, the EQUEST is not capable of filtering too heavy disturbances.

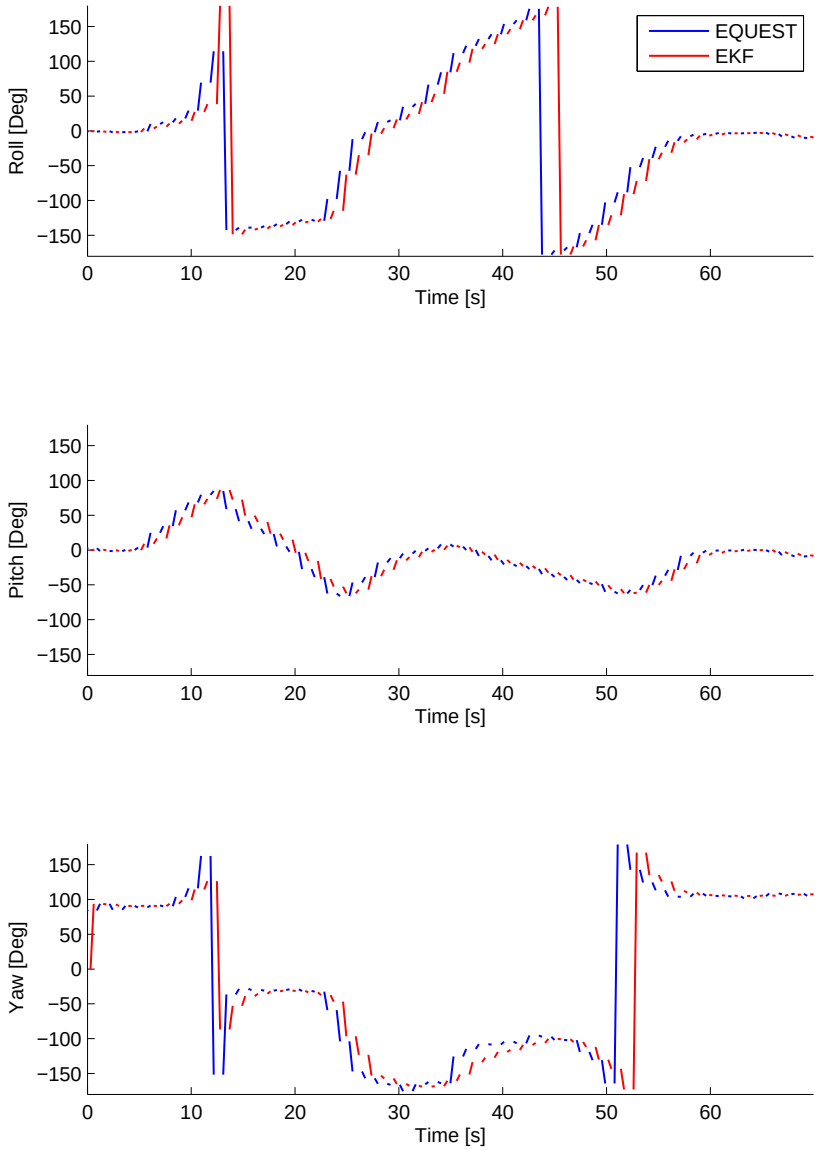


Figure 23: The prototype output with a low magnetic field disturbance.

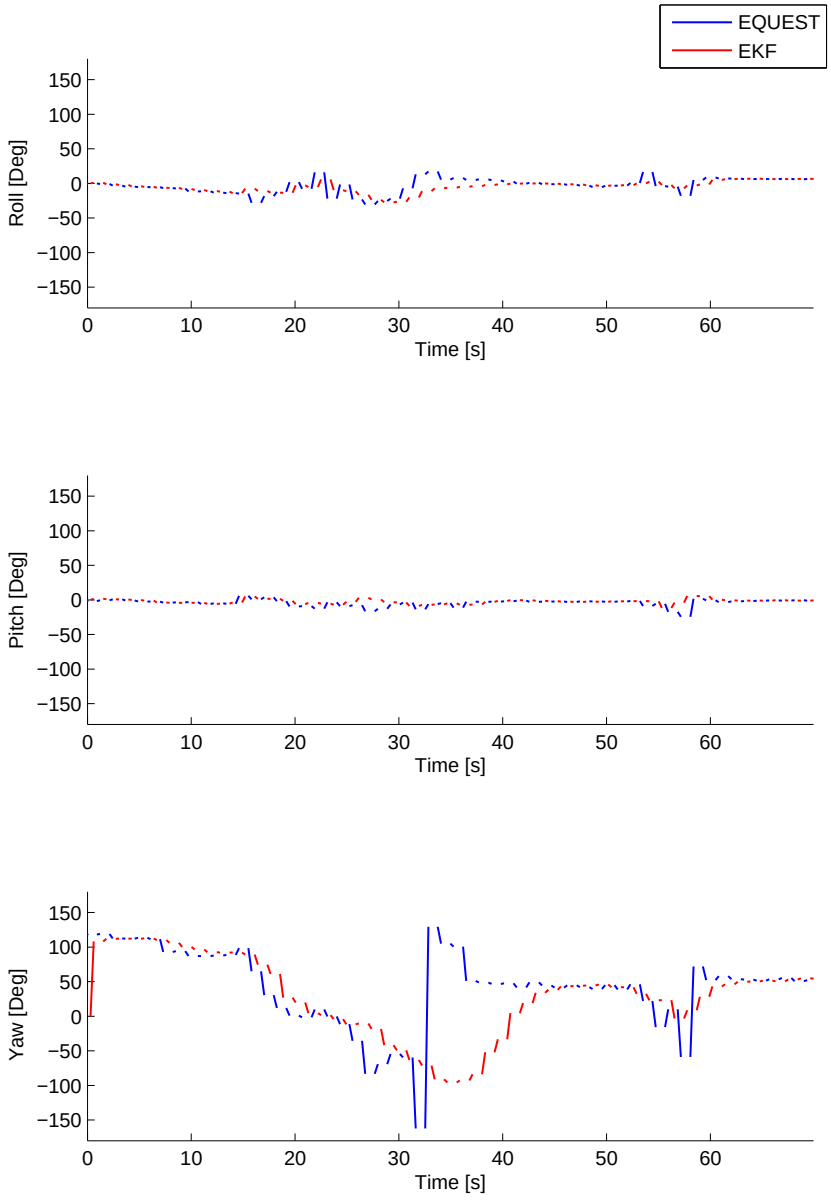


Figure 24: Moderate shaking of the prototype.

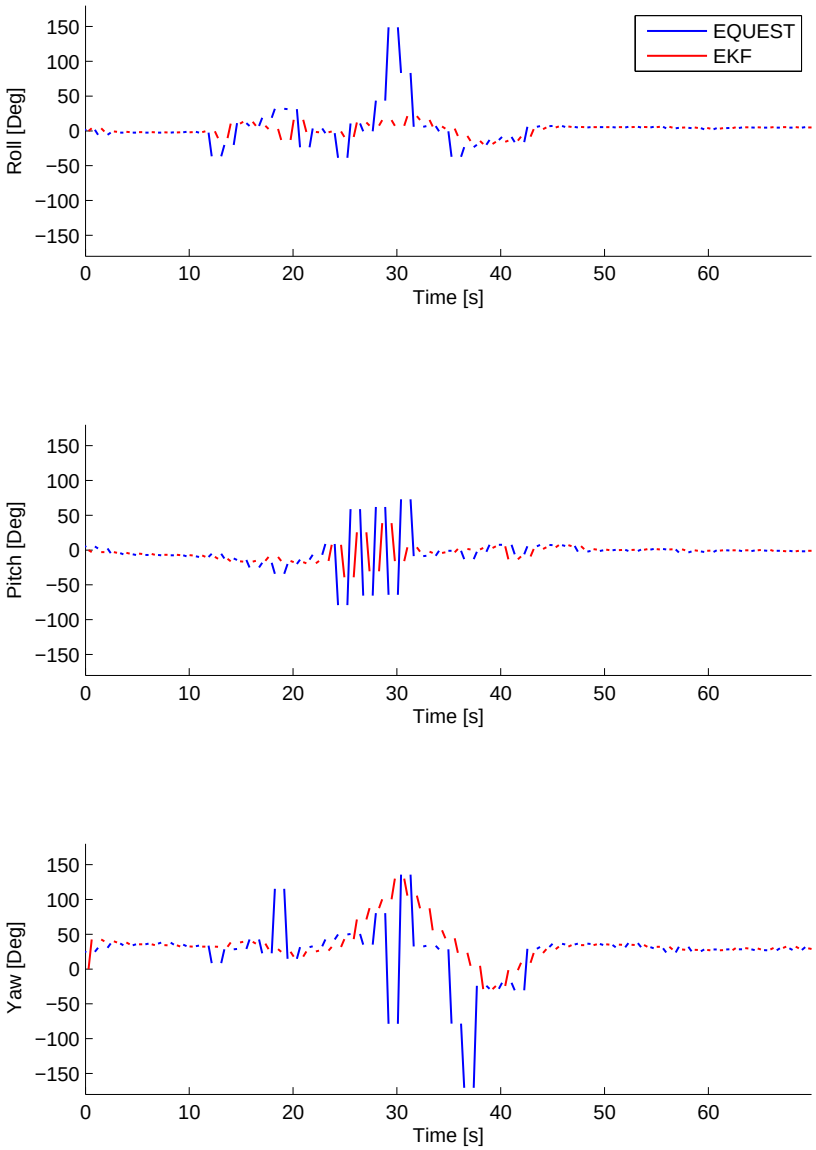


Figure 25: Heavy shaking of the prototype.

8.7 Power Consumption

By using a multimeter, the power consumption of the prototype is easily monitored. The voltage was supplied through the USB connection, providing 5V for the entire test. During regular use, with the microcontroller running both algorithms, the current was found to be 73mA. This gives a total power consumption of 363 mW. By letting the controller go into sleep mode and then doing the measurement again, the current went down to 61mA. This means that during eclipse the power consumption can be reduced from 363mW to 305mW, only by activating sleep mode on the microcontroller. It should be kept in mind that the power consumption of the microcontroller increases quadratically with the clock speed. The sensor will still be running though. According to the datasheet, it will use an average of 40 mA with the supplied voltage of 3.3V.

Measurements have also been done with the prototype running at 3.3V. Powering down to 3.3V reduces the current usage to 45 mA during regular use, and 34 mA while sleeping. However, the ATMEGA 2561 microcontroller, attached to the prototype, demands voltage of at least 4.5V to operate at 16Mhz. Hence, there are no guaranty that these results are accurate. By changing the crystal the speed can be reduced to 8Mhz, resulting in the microcontroller being able to operate at 3.3V. The CHIMU sensor will draw approximately the same amount of current at 3.3V as it does on 5V. This means that the total power consumption will be less at 3.3V for the sensor. The voltage for the sensor should therefore be 3.3V, and the microcontroller should operate at the lowest voltage possible while still being able to perform the attitude calculations.

It is impossible to power down the CHIMU sensor used for the prototype. However, an alternative is to create an electronic switch by changing the prototype design. The switch should power down the sensor activated by a control signal from the microcontroller. It is recommended to consider this option before the final satellite construction.

8.8 Start-up Time

As described in Section 8.7, putting both the microcontroller and the IMU to sleep may save essential amounts of power. The microcontroller should also be set to sleep mode during attitude control. When the magnetic actuators are on, the local magnetic field will be disturbed. If attitude is estimated during this period, the estimations will be inaccurate. Since the microcontroller and the IMU have to be on for some period and off for some period of time, it is important to know their start-up time. The microcontroller wakes up much faster than the IMU, specially because the microcontroller can be set to sleep mode instead of turning it off. The IMU must be shut down since no sleep mode exists for the IMU. The start-up time for the the IMU is given in datasheet and is 0.1 sec. Therefore at least 0.1 sec is required before attitude can be estimated.

9 System Integration

9.1 Integrating with Attitude Control

The attitude control is mainly divided into two parts:

1. *Control during detumbling:* The CUBESAT will be sent to space using a rocket containing several other satellites. The satellites are released into space almost simultaneously using a mechanical device. In order to avoid collision, the satellites are stacked on top of each other in an unsymmetrical way. A spin is induced to the satellites movement because of the mechanical device releasing it. This spin must be damped before the satellite can be controlled to the desired orientation. The control process of the damping, does not require any knowledge about the absolute orientation, but only the change in angular velocity and info about the local magnetic field. It means that during detumbling, gyro and magnetometer data can be directly sent to attitude control instead of estimating the orientation. Some logic must be implemented in order to send the data from the IMU directly to the attitude control system. If both attitude estimation and control are implemented on the same microcontroller, it is easy to send the data. If the microcontroller is not capable of handling both control and estimation, they might have to be implemented on two separate controllers introducing some additional logic for the communicate between them.

2. *Control in orbit:* To point the satellite toward the Earth, full orientation must be known. The prototype estimates attitude in NED frame. For control of the satellite in orbit, it is common to use orbit frame instead of NED frame. To be able control the satellite correctly the estimated attitude in NED coordinates must be transferred to orbit frame using rotation matrices. As it is for the detumbling, it is also important for control in orbit to be able to switch between estimation and control.

For both estimation and control, the model of magnetic field is required. The NUTS will use an IGRF model in order to get the best possible data about the magnetic field while orbiting [25]. If control and estimation are implemented on same controller, they can share the same field model. If they are implemented on separate controller, either two magnetic field models must be implemented or some additional communication between the microcontrollers must be introduced.

9.2 Integrating with the Rest of the System

The satellite contains several subsystems, not only the attitude determination and control. Other required systems are power management, camera, ground communication and much more. These parts does not require any attitude sensor data, but may have interest of knowing the estimated orientation. The

estimated orientation is important for the camera to be able to take images of locations of interest or it can be important for analytical reasons. Therefore communication with the rest of the system must be established. The developed prototype got two communication lines, one between the microcontroller and the IMU, and the second between the microcontroller and the computer. The communication between the microcontroller and the computer is not necessary for the final satellite and can instead be used for communication with the rest of the system. It is important to communicate both attitute data and subsystem information through this communication line. If the estimation of the satellite crashes, it should be possible to detect the error and restart it from another system.

10 Conclusion

10.1 Discussion

Two methods were derived, tested and compared in this project. The QUEST algorithm has been developed further to EQUEST, to include non-vectorized measurements in the estimations. An Extended Kalman Filter was fitted for the CUBESAT project at NTNU, and implemented with a sensor fusion model instead of a dynamical model of the satellite.

The results given in this paper indicates that both the EKF and the EQUEST are able to estimate the attitude for a satellite. Through testing, the EKF proved its abilities of stable tracking of orientation. One of the greatest advantages with the EKF was observed during shake testing. The filtering effect was able to reduce the noise during shaking, and the EKF still provided a stable estimation.

In order to compare the two methods, they ran simultaniously with the same input data. In all tests, the EQUEST tracked the attitude faster than the EKF. The test done with the ABB robot indicated that EQUEST had better accuracy than EKF. The EKF had some overshoot, but adjusted to the correct value with time. The overshoot and inaccuracy in the EKF might have been caused by poor tuning parameters. The tuning parameters of EQUEST is more intuitivly chosen compared to those on EKF. Finding good tuning parameters for the process covariance matrix for EKF is not easy and often many attempts are required.

The power consumption was weighted during the development of the attitude determination system. As mentioned earlier, the EKF was 5 times slower than the EQUEST algorithm. This indicates that the speed of the microcontroller running EQUEST, could be reduced to $\frac{1}{5}$ and still estimate the attitude at the same time as an EKF running at full speed. Remembering that the power consumption of the microcontroller has a quadratic relationship with the clock speed, EQUEST can save significant amounts of power.

Another ability that makes the EQUEST useful with respect to the CUBESAT project is the startup time. As the satellite will be controlled by magnetourqers, the local magnetic field will be greatly affected by the attitude control. Hence, it is important that the attitude estimation and the attitude control is strictly seperated. Since the estimated attitude will be inaccurate during control, the results would be useless. By switching the attitude determination off, power can be saved. Therefore a short startup time is preferred.

10.2 Future Work

It is crucial for the attitude estimation of the final satellite that the accelerometer is replaced by a vectorized measurement. This could for instance be a sun sensor, as described earlier, an Earth sensor or another vectorized measurement. Replacing the accelerometer will not change the estimation methods, except for the input reference and the measurement vector. If the sun sensor is chosen,

an albedo model should be implemented as well as a Sun reference model. By following the recommended substitution in Section 4.4, it should be possible to adapt the estimations.

Some software improvements can be done to the attitude determination. Rewriting the code from floating point variables to fixed point variables will save both memory and computational time and should be done before the satellite is launched.

Due to ionospheric radiation, the electrical components are vulnerable to errors. It is very hard to estimate the lifespan of the electronics. Hence, two attitude determination modules should be considered for implementation in the final flight module. The modules are small and light weighted and if the space and weight budget allows it, multiple attitude estimation modules is recommended.

Since attitude estimation is a subsystem of the entire satellite, some of the improvements requires that other modules are completed. The attitude determination is partly based on the Earth's magnetic field. It is possible to improve the accuracy of the estimations if one could measure and estimate the local magnetic field created by all the electronic components within the satellite. This noise can be measured and feed forwarded to the attitude estimation in order to subtract it from sensor output.

The sleep mode implemented on the microcontroller, discussed in Section 5, should be handled from an external module in the satellite. Therefore communication must be established with other modules in the system.

The developed prototype is powered from the USB connection. Obviously, this is not possible in the on-board flight module. Hence a connection providing power is needed for the final module. The final circuit board should be designed and fitted for system integration. Instead of using the IMU selected on the prototype, the final module should be designed with individual sensors fitted for the application.

A IAC 2011 Accepted Abstract

This is the abstract submitted to the International Astronautical Congress 2011. The abstract was accepted for an 15 minutes oral presentation on the main event of the congress.

Abstract

As a part of the cubesat project at the Norwegian University of Science and Technology, this paper study development and comparison of attitude estimation methods for cube satellites using low-cost IMUs. The estimation is done using data from 3-axis gyroscope, 3-axis accelerometer and 3-axis magnetometer.

In this paper a new method for attitude estimation has been developed based on QUaternion ESTimation (QUEST). A major concern with QUEST is that it cannot handle non-vectorized measurements such as gyroscope data. Substantial improvements have been made to fuse vectorized and non-vectorized measurements, making the new Extended QUaternion ESTimation (EQUEST) more suitable for attitude estimation. The well known Extended Kalman Filter (EKF) is derived and implemented for comparison. Both methods have been developed and simulated in MATLAB. Then the codes have been rewritten using C language. The methods are compared both theoretically and experimentally with implementation and testing on an AVR microcontroller. Minimum power usage and number of arithmetic operations were considered during the software development.

Testing indicates that the EKF provides a smoother estimation than the newly developed EQUEST. In contrast to EQUEST, the EKF is able to estimate the magnetometer and accelerometer bias. However, the EQUEST is having a significantly faster settling time and is less computational costly. Compared to the EKF, EQUEST runs more than 5 times faster. It also requires only 8% of the arithmetic operations of the EKF. Another disadvantage with the EKF is tracking problems that occur close to Earth's magnetic poles. At these locations, the gravitational vector and the magnetic field vector are almost parallel. With vectors close to parallel, the mathematical formulation of the EKF makes tracking of a rotation around NED down-axis extremely difficult. These difficulties are hardly observed in the EQUEST algorithm, which makes it very attractive for attitude estimation in the Polar Regions.

The attitude control of cubesats is often done by magnetorquers, which will affect the local magnetic field. Hence, control and estimation should not be done simultaneously, resulting in the estimation and control switching on and off. For this reason, the long settling time of the EKF makes the EQUEST even more attractive.

The results in this paper indicate that the newly developed EQUEST is highly suitable for projects with either limited budget, space, weight or computational power.

B IAC 2011 Paper

The final paper submitted to the IAC2011 congress. This paper is written in cooperation with Professor Jan Tommy Gravdahl using the style guidelines provided by IAC.

A COMPARISON OF ATTITUDE DETERMINATION METHODS: THEORY AND EXPERIMENTS

Kristian Lindgård Jenssen

Department of Engineering Cybernetics

Norwegian University of Science and Technology(NTNU)

N-7491 TRONDHEIM, NORWAY

kristian.jenssen@gmail.com

Kaan Huseby Yabar*, Jan Tommy Gravdahl†

Abstract

In this paper two attitude determination methods are derived, implemented and compared. The QUaternion ESTimator (QUEST) algorithm is extended to include non-vectorized measurements, whereas the well known Extended Kalman Filter has been implemented for performance comparison. The methods have been developed for the Norwegian University Test Satellite (NUTS), as a part of the CubeSat project at the Norwegian University of Science and Technology (NTNU). Due to the specifications of CubeSats, both methods are customized for satellites with limited weight-, size- or financial budgets. The attitude estimation is based on two vectorized measurements and data from a gyroscope. Both methods have been developed and simulated in MATLAB. The code have been rewritten using C language. The methods are compared both theoretically and experimentally with implementation and testing on an AVR microcontroller.

Testing indicates that the EKF provides a smoother estimation than the newly developed EQUEST. In contrast to EQUEST, the EKF is able to estimate sensor biases. However, the EQUEST has significantly faster settling time and is less computational costly. Compared to the EKF, EQUEST runs more than 5 times faster. It also requires only 8% of the arithmetic operations of the EKF. Another disadvantage with the EKF is tracking problems that occur when the two vectorized measurements are close to parallel. With vectors close to parallel, the mathematical formulation of the EKF makes tracking of a rotation around the parallel axis extremely difficult. These difficulties are hardly observed in the EQUEST algorithm, which makes it very attractive for attitude estimation.

For small satellites the magnetic field of the Earth is often used for attitude determination. A substantial number of these satellites use magnetorquers for attitude control, affecting the local magnetic field. Hence, control and estimation should not be done simultaneously, resulting in the estimation and control switching on and off. For this reason, the long settling time of the EKF makes the EQUEST even more attractive.

1. BACKGROUND

Attitude determination is an important subsystem in satellites of all sizes. Knowledge of the satellites orientation is crucial to perform space missions such as nadir pointing control. Two common methods to estimate the orientation are the Kalman Filter (KF) and the Quaternion Estimator (QUEST) [1][2][3]. The KF tries to fuse system dynamics, input data and sensor measurements to estimate the best possible attitude, whereas the QUEST calculates the attitude by minimizing a cost function relating sensor measurements with known references.

The equipment used for estimating the attitude vary with price, power consumption and physical size. The attitude estimation systems in this paper are based on both the QUEST and the KF, designed especially for CubeSats [4]. A CubeSat is a picosatellite where the dimensions of the cube has been standardized to 10×10×10 cm, with a weight limited to 1.3

*Department of Engineering Cybernetics - NTNU, Norway, kayabar@gmail.com

†Department of Engineering Cybernetics - NTNU, Norway, jan.tommy.gravdahl@itk.ntnu.no

kg. NTNU takes part in a student CubeSat project, designing a double cube - NUTS [5]. As the name indicates, a double cube is 10×10×20 cm, with a weight limited to 2.6 kg. The NUTS (NTNU Test Satellite) project was started in September 2010. The project is part of the Norwegian student satellite program run by NAROM (Norwegian Centre for Space-related Education). The projects goal is to design, manufacture and launch a double CubeSat by 2014. As main payload an IR-camera is planned, as well as a short-range RF experiment. The satellite will fly two transceivers in the amateur radio bands. Ten students were involved in the project first half of 2011.

Two satellites have been developed at NTNU earlier, each containing attitude estimation [6] . Svartveit [7] estimated the attitude by using a discrete Kalman filter based on measurements from magnetometer and sun sensor. The solar panels were used as a crude sun sensor. The experiment Svartveit did, indicated that the sun sensor seems to be inaccurate mainly due to the Earths albedo effect. Ose [8] made further work in order to implement the extended Kalman filter (EKF) in MATLAB. Due to the complexity of the EKF, only a linearized version was implemented on a microcontroller. With respect to all the challenges the previous NTNU students unveiled, this paper base the attitude determination system on a different approach. The estimation is based on two vectorized measurements as well as data from a gyroscope. Sabatini [9] developed an extended Kalman filter based on accelerometer, gyroscope and magnetometer, for use in biomedical engineering. The extended Kalman filter developed in this project is adapted from the work done by Sabatini to fit the satellite.

Attitude can also be estimated using quaternion estimation (QUEST). Markley [3] describes how two vectorized measurements can be used to estimate orientation. However, the method can only be used for vectorized measurements, which makes the gyroscope unsuited. Psiaki [1] has extended the QUEST in order to handle an arbitrary dynamic model and to estimate errors such as rate-gyro biases. The extended QUEST (EQUEST) developed in this paper, is based on the work done by Psiaki and Markley, with focus on integrating the nonvectorized gyroscope measurements.

2. THEORY

2.1. Coordinate Frames

The following frames have been used to describe the attitude of the satellite

BODY: This frame is attached to the satellite. In the BODY frame the axes coincide with the principle axes of inertia, and the positive z-axis is defined as the vector pointing outwards from the quadratic side of the satellite. The x- and y-axis are orthogonal to the rectangular sides of the cube.

NED: North-East-Down (NED) is an intertial frame defined relative to Earth's surface with the x-axis pointing towards north, z-axis downwards perpendicular to Earth's reference ellipsoid. The y-axis completes a right handed orthogonal coordinate system, with the positive y-axis pointing towards East. Earth's reference ellipsoid is a mathematically defined surface fitted to approximate the shape of the Earth.

2.2. Rotation Matrix

A rotation matrix R is used to describe a rotation between two coordinate frames. It is common to describe the rotation from one frame to another as three different rotations. First a rotation ψ around the z-axis, then a rotation θ around the rotated y-axis and finally a new rotation ϕ around the current x-axis. The entire rotation from frame "a" to frame "b", is described as

$$R_b^a = R_z(\psi)R_y(\theta)R_x(\phi) \quad [2.1]$$

The rotation matrix can be represented using quaternions [10]

$$R(q) = (q_4^2 - \|q_{13}\|^2)I_{3\times 3} + 2q_{13}q_{13}^\top - 2q_4[q_{13}\times] \quad [2.2]$$

A rotation matrix has the following properties

$$det(R) = 1 \quad [2.3]$$

$$R^\top = R^{-1} \quad [2.4]$$

A vector in NED frame k^n can be written in BODY frame k^b using rotation matrix as given below

$$k^b = R_n^b k^n \quad [2.5]$$

2.3. Extended Kalman Filter (EKF)

The Kalman filter is only applicable for linear systems. However, it is possible to extend the filter to also deal with nonlinear problems, but the filter will then lose some of its properties. The EKF is adapted for nonlinear problems through a linearization of the nonlinear equations for each iteration. Due to the linearization, the EKF is not necessarily optimal and

can diverge if initial errors are too large or if the system model is inaccurate. The EKF is an iterative method where the estimates are based on several measurements, as well as a process model.

Defining the discrete nonlinear system as

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \quad [2.6]$$

$$z_k = h(x_k) + v_k \quad [2.7]$$

where x is state vector, $f(.)$ describes the system dynamics, u is control input, w is process noise, h is the measurment model and v is measurement noise. Both measurement and process noise are assumed to be zero mean Gaussian. The subscript k denotes discrete time. Using the system described above, the equations for the extended Kalman filter can be written as [11]

Predict:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}) \quad [2.8]$$

$$P_k^- = F_k P_k F_k^T + Q_{kal} \quad [2.9]$$

Update:

$$K_k = P_k^- H_k^\top (H_k P_k^- H_k^\top + R_{kal})^{-1} \quad [2.10]$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H_k \hat{x}_k^-) \quad [2.11]$$

$$P_k = (I - K_k H_k) P_k^- \quad [2.12]$$

where $\hat{x}$ denotes estimated state vector, P is error covariance matrix, K is calculated Kalman gain and

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1}, u} \quad [2.13]$$

is the derivative of the nonlinear system with respect to the states, and

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{k-1}} \quad [2.14]$$

is the derivative of the measurement equations with respect to the states.

As these equations show, the nonlinear model is still used in predicting the new states, whereas a linearized model is used for comparing the measurements with the current states in the update phase and for computing the covariance matrix and the Kalman gain. R_{kal} is the measurement covariance matrix, and Q_{kal} is the process covariance.

2.4. Quaternion Estimation

Another method for estimation of the rotation matrix based on the sensor measurements is the QUaternion ESTimator (QUEST) [1]. QUEST will minimize the cost function defined as

$$J(q) = \frac{1}{2}\sum_{j=1}^{n} \frac{1}{\sigma_j^2}(b_j - R_b^i(q) r_j)^\top (b_j - R_b^i(q) r_j)$$

$$= \frac{1}{2}\sum_{j=1}^{n} \frac{1}{\sigma_j^2}(b_j^T b_j - 2 b_j^\top R_b^i(q) r_j + r_j^\top r_j) \quad [2.15]$$

where r_j are known unit vectors in the NED frame, and b_j are unit vectors of the measured observations in the body-fixed frame. σ_j are the standard deviation of the measurement error. However, both r_j and b_j are unit vectors, and this reduces the equation to

$$J(q) = \sum_{j=1}^{n} \frac{1}{\sigma_j^2}(1 - b_j^\top R_b^i(q) r_j) \quad [2.16]$$

Minimizing J is equivalent to maximizing

$$J_{QUEST}(q) = \sum_{j=1}^{n} \frac{1}{\sigma_j^2} b_j^\top R_b^i(q) r_j \quad [2.17]$$

The QUEST does not depend on initial conditions, which is a great advantage. Another advantage is that the algorithm can be solved exactly by solving an eigenvalue problem. However, the QUEST algorithm can only estimate the attitude quaternions.

2.5. Attitude Estimation Using Extended Kalman Filter

The theory of the EKF was explained in Section 2.3. There are several ways to configure the EKF for attitude estimation based on vectorized measurements and a gyroscope. Here, the state vector was chosen to consist of the quaternions q and the biases from both vectorized measurements b_{v1} and b_{v2}. The state vector x is given below

$$x = \begin{bmatrix} q \\ b_{v1} \\ b_{v2} \end{bmatrix}$$

where

$$q = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^\top$$

$$b_{v1} = \begin{bmatrix} b_{v1,x} & b_{v1,y} & b_{v1,z} \end{bmatrix}^\top$$

$$b_{v2} = \begin{bmatrix} b_{v2,x} & b_{v2,y} & b_{v2,z} \end{bmatrix}^\top$$

The great advantage of estimating the biases of the two vectorized measurements is that the EKF will subtract them from the measurements during the next iteration. This will increase the precision of the measurements and give a more accurate attitude estimate. Since all the measurements are done in the BODY frame and the reference model is given in the NED frame, the sensor model includes a rotational matrix dependent on the unit quaternions. The rotational matrix introduce a non-linearity in the measurement matrix. The measurement model z is given by the two vectorized measurements as

$$z = \begin{bmatrix} v_1^b \\ v_2^b \end{bmatrix} \tag{2.18}$$

where

$$v_1^b = R_n^b(q) v_{1,real}^n + b_{v1}^b \tag{2.19}$$

$$v_2^b = R_n^b(q) v_{2,real}^n + b_{v2}^b \tag{2.20}$$

In order to include angular velocity as a state, a dynamical model of the satellite can be used. An alternative is to use the measurements from the gyroscope to estimate the dynamical model for the quaternions. The kinematic differential equation for unit quaternions can be written as [9]

$$\dot{q} = \frac{1}{2} \begin{bmatrix} \omega\times & \omega \\ -\omega^\top & 0 \end{bmatrix} q \tag{2.21}$$

where

$$\omega\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{2.22}$$

with

$$\omega = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^\top$$

being the angular velocity measured in the body frame. Further, constant bias can be assumed from the two vectorized measurements:

$$\dot{b}_{v1} = 0 \tag{2.23}$$

$$\dot{b}_{v2} = 0 \tag{2.24}$$

The differential equation for the entire system is therefore [9]

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \dot{b}_{v1} \\ \dot{b}_{v2} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2} \begin{bmatrix} \omega\times & \omega \\ -\omega^\top & 0 \end{bmatrix} & 0\ 0 \\ & 0\ 0 \\ 0 & 0\ 0 \end{bmatrix} \begin{bmatrix} q \\ b_{v1} \\ b_{v2} \end{bmatrix} \tag{2.25}$$

Note that the system equation is linear, making the computation of the EKF less complicated. However, an EKF is still necessary due to the nonlinearity in the measurement model.

The system in Equation (2.25), is given in continuous time. For computer implementation, discretization is required. The system is discretized using zero-order hold with sampling time T_s. The discrete system will be given as

$$x_{k+1} = \begin{bmatrix} F & 0 & 0 \\ 0 & I_{3x3} & 0 \\ 0 & 0 & I_{3x3} \end{bmatrix} x_k \tag{2.26}$$

where

$$F = expm\left(\frac{1}{2} \begin{bmatrix} \omega\times & \omega \\ -\omega^\top & 0 \end{bmatrix} T_s \right) \tag{2.27}$$

with $expm$ being the matrix exponential.

Note that the angular velocity is not a part of the measurement vector, nor the state vector. As mentioned earlier, the measurements from the gyroscope will be indirectly integrated in the EKF through the dynamical model of the quaternions.

The vectors explained above are used in the equations in Section 2.3 to iteratively compute the attitude of the satellite.

2.6. Extended Quaternion Estimator

As mentioned in Section 2.4, the QUEST algorithm is not able to utilize the measurements from the gyroscope. However, it is possible to extend the QUEST and implement an Extended QUaternion ESTimator (EQUEST) in order to include the gyroscope measurements. The main idea behind the EQUEST is to modify the cost function. This is done by adding another term, containing the gyroscope measurements.

By tracking the rotation based on gyroscope measurements, it is possible to penalize aberrations from the rotation matrix estimated by the gyroscope alone.

$$J_{gyro}(q) = \frac{1}{2}(q - \hat{q}_{gyro})^\top D(q - \hat{q}_{gyro}) \qquad [2.28]$$

here $\hat{q}_{gyro}$ is the estimated attitude quaternion based on gyroscope tracking, and D is a diagonal weighting matrix. The main idea by using the term $q - \hat{q}_{gyro}$ is to minimize the cost function. Note that the subtraction will not result in an attitude quaternion.

The EQUEST can be expanded further by adding a prediction term. The prediction is most suitable for applications where it is possible to forecast upcoming orientation based on previous behavior. It can also be used to filter out noise. The slow and predictable change of attitude for the satellite makes it possible to use previous attitude calculations to estimate future orientation. For a short period of time, the attitude change will be minimal. However, as several attitude calculations are done in this period it is possible to establish a linear relation between time and change of attitude. This is illustrated in Figure 2.1. A deviation from the predicted term can be penalized in the cost function by adding the following term

$$J_{pre}(q) = \frac{1}{2}(q - \hat{q}_{pre})^\top S(q - \hat{q}_{pre}) \qquad [2.29]$$

where q vector contains attitude quaternions, $\hat{q}_{pre}$ is the predicted attitude based on previous observations, S is state weight matrix.

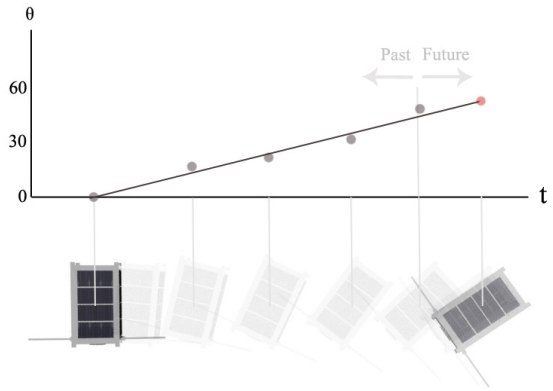


Figure 2.1: Linear prediction based on past orientations. Note that the motion is exaggerated for illustrational purposes.

Extending Equation (2.17) with the two terms described in (2.28) and (2.29) gives

$$\begin{aligned} J_{EQUEST} &= J_{QUEST} + J_{gyro} + J_{pre} \\ &= \frac{1}{2}\sum_{j=1}^{n}\left\{\frac{1}{\sigma_j^2}(b_j - R_b^i(q)r_j)^\top(b_j - R_b^i(q)r_j)\right\} \\ &+ \frac{1}{2}(q - \hat{q}_{gyro})^\top D(q - \hat{q}_{gyro}) \\ &+ \frac{1}{2}(q - \hat{q}_{pre})^\top S(q - \hat{q}_{pre}) \end{aligned} \qquad [2.30]$$

subject to

$$q^\top q = 1 \qquad [2.31]$$

It is possible to rewrite both extensions to quadratic terms. By writing the entire equation in quadratic form, the cost function can be minimized using well-known optimization techniques[12].

Note that the EQUEST is still not able to estimate the biases. It is possible to estimate the biases using other methods, and then subtract them from the measurements used in EQUEST. Due to computational costs, this is not considered here.

2.7. Solving the Extended Quaternion Estimator

One option for solving the minimization problem given in (2.30) and (2.31) is to use the Lagrangian multiplier method. However, the cost function has to be written on the special form

$$J(x) = \frac{1}{2}x^\top Gx + x^\top c \qquad [2.32]$$

where G is a positive definite matrix and c is constant with respect to x. The original QUEST criterion in (2.15) can be posed in the quadratic form [3]

$$g(q) = -q^\top Vq \qquad [2.33]$$

where V is a symmetric matrix given by

$$V = \begin{bmatrix} U - \varphi I_{3\times 3} & Z \\ Z^\top & \varphi \end{bmatrix} \quad [2.34]$$

with

$$U = L + L^\top \quad [2.35]$$

$$L = \sum_{j=1}^{n} \frac{1}{\sigma_j^2}(b_j r_j^\top) \quad [2.36]$$

$$Z = \begin{bmatrix} L_{23} - L_{32} \\ L_{31} - L_{13} \\ L_{12} - L_{21} \end{bmatrix} \quad [2.37]$$

$$\varphi = trace(L) \quad [2.38]$$

The gyroscope tracking needs to be written in the same quadratic form in order to solve EQUEST with Lagrangian multipliers:

$$J_{gyro}(q) = \frac{1}{2}(q - \hat{q}_{gyro})^\top D(q - \hat{q}_{gyro}) \quad [2.39]$$

$$= \frac{1}{2}(q^\top Dq - q^\top D\hat{q}_{gyro} - \hat{q}_{gyro}^\top Dq + \hat{q}_{gyro}^\top D\hat{q}_{gyro}). \quad [2.40]$$

Since

$$(\hat{q}_{gyro}^\top Dq)^\top = \hat{q}_{gyro}^\top Dq = q^\top D\hat{q}_{gyro} \quad [2.41]$$

we can rewrite (2.39) as

$$J_{gyro}(q) = \frac{1}{2}(q^\top Dq - 2q^\top D\hat{q}_{gyro} + \hat{q}_{gyro}^\top D\hat{q}_{gyro}) \quad [2.42]$$

Further, the term $\hat{q}_{gyro}^T D\hat{q}_{gyro}$ will be constant with respect to q. This term will not affect the minimization problem, hence it can be removed. Now the gyroscope part of the cost function can be written in quadratic form as

$$J_{gyro}(q) = \frac{1}{2}q^\top Dq - q^\top D\hat{q}_{gyro} \quad [2.43]$$

Exactly the same as above can be done to write the prediction term in quadratic form

$$J_{pre}(q) = \frac{1}{2}q^\top Sq - q^\top S\hat{q}_{pre} \quad [2.44]$$

By adding (2.43) and (2.44) with (2.33), the entire EQUEST can be written in quadratic form as:

$$J_{EQUEST}(q) = \frac{1}{2}q^\top (D + S - V)q + q^\top(-D\hat{q}_{gyro} - S\hat{q}_{pre}) \quad [2.45]$$

Introducing new variables

$$\kappa = D + S - V \quad [2.46]$$

$$\xi = -D\hat{q}_{gyro} - S\hat{q}_{pre} \quad [2.47]$$

the problem will be to minimize

$$J_{EQUEST}(q) = \frac{1}{2}q^\top \kappa q + q^\top \xi \quad [2.48]$$

subject to

$$q^\top q = 1 \quad [2.49]$$

The Lagrangian equation is now given as

$$\mathcal{L} = \frac{1}{2}q^\top \kappa q + q^\top \xi + \frac{\lambda}{2}(q^\top q - 1) \quad [2.50]$$

The q that minimizes (2.48) is found as

$$\frac{d\mathcal{L}}{dq} = \kappa q + \xi + \lambda I q = 0 \quad [2.51]$$

$$q = -(\kappa + \lambda I)^{-1}\xi. \quad [2.52]$$

By combining (2.49) and (2.52), the constraint can be written as

$$\xi^\top (\kappa + \lambda I)^{-2}\xi = 1 \quad [2.53]$$

The largest positive real λ will give the global minimum for Equation (2.48)[1]. Further κ is a symmetric matrix, hence it can be decomposed as

$$\kappa = M \begin{bmatrix} -\lambda_1 & 0 & 0 & 0 \\ 0 & -\lambda_2 & 0 & 0 \\ 0 & 0 & -\lambda_3 & 0 \\ 0 & 0 & 0 & -\lambda_4 \end{bmatrix} M^\top \quad [2.54]$$

where λ_i are the eigenvalues of κ, and M is an orthogonal eigenvector matrix. By introducing

$$c = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = M^\top \xi \Leftrightarrow \xi^\top = c^\top M^\top, \quad [2.55]$$

(2.53) can be written as

$$c^\top M^\top \left\{ M \begin{bmatrix} -\lambda_1 & 0 & 0 & 0 \\ 0 & -\lambda_2 & 0 & 0 \\ 0 & 0 & -\lambda_3 & 0 \\ 0 & 0 & 0 & -\lambda_4 \end{bmatrix} M^\top + \begin{bmatrix} \lambda & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & \lambda & 0 \\ 0 & 0 & 0 & \lambda \end{bmatrix} \right\}^{-2} Mc - 1 = 0 \quad [2.56]$$

Since the diagonal matrix will affect the eigenvalues, and the inverse of an orthogonal matrix is the transposed of the orthogonal matrix, this can be simplified to

$$c^\top M^\top \Upsilon^{-2} M c - 1 = 0 \quad [2.57]$$

where

$$\Upsilon = M \begin{bmatrix} \lambda - \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda - \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda - \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda - \lambda_4 \end{bmatrix} M^\top \quad [2.58]$$

Now since M is orthogonal, $M^\top M = I$. Further, the inverse of a diagonal matrix can be computed element by element. Equation (2.57) can now be written as

$$c^\top \begin{bmatrix} \frac{1}{(\lambda-\lambda_1)^2} & 0 & 0 & 0 \\ 0 & \frac{1}{(\lambda-\lambda_2)^2} & 0 & 0 \\ 0 & 0 & \frac{1}{(\lambda-\lambda_3)^2} & 0 \\ 0 & 0 & 0 & \frac{1}{(\lambda-\lambda_4)^2} \end{bmatrix} c - 1 = 0 \quad [2.59]$$

or

$$\frac{c_1^2}{(\lambda-\lambda_1)^2} + \frac{c_2^2}{(\lambda-\lambda_2)^2} + \frac{c_3^2}{(\lambda-\lambda_3)^2} + \frac{c_4^2}{(\lambda-\lambda_4)^2} - 1 = 0 \quad [2.60]$$

The optimal λ (λ_{opt}) will be larger than the smallest eigenvalue[1]. After λ_{opt} is identified, it can be substituted back into Equation (2.52) to find the q that minimizes the cost function.

In contrast to EKF, EQUEST is not able to estimate the sensor biases. The state vector is therefore chosen to be the quaternion

$$x = q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \quad [2.61]$$

Equation (2.21) is used to estimate the $\hat{q}_{gyro}$ term in EQUEST. We propose to calculate the $\hat{q}_{pre}$ term by using simple linear regression with a window size of 10 samples. The next quaternion vector is predicted by using the 10 latest samples, and fitting them to an equation for a line

$$y(t) = b_0 t + b_1 \quad [2.62]$$

b_0 will represent the slope of the line, while b_1 is the measured value at $t = 0$. With n observations, b_0 and b_1 can be found by solving the following formulas [14]

$$b_0 = \frac{n\sum_{i=1}^{n} y(t_i)t_i - \sum_{i=1}^{n} t_i \sum_{i=1}^{n} y(t_i)}{n\sum_{i=1}^{n} (t_i^2) - \left(\sum_{i=1}^{n} t_i\right)^2} \quad [2.63]$$

$$b_1 = \frac{\sum_{i=1}^{n} y(t_i) \sum_{i=1}^{n} t_i^2 - \sum_{i=1}^{n} t_i \sum_{i=1}^{n} t_i y(t_i)}{n\sum_{i=1}^{n} t_i^2 - \left(\sum_{i=1}^{n} t_i\right)^2} \quad [2.64]$$

The next $\hat{q}_{pre}$ is estimated using the linear relation found with the last 10 samples using the values b_0 and b_1 as slope parameters. Note that the prediction term does not have to be linear. It should be adapted for the rest of the system. In case of less strict restrictions on calculations power or time, the prediction term might even be an EKF.

3. IMPLEMENTATION

3.1. Microcontroller

In order to test the estimation methods, both EKF and EQUEST were implemented in MATLAB. Both methods were later implemented on microcontrollers. During testing, an 8-bit microcontroller from Atmel was used. In addition to the estimation methods, several important functions were configured. A watchdog timer was implemented to reset the microcontroller in case the software run into an endless loop. Sleep mode for the microcontroller were also configured. The satellite does not have to estimate the attitude all the time. When no attitude data is required, activating the sleep mode can save considerable amount of power.

3.2. Implementing the Kalman Filter in C

In order to implement the extended Kalman filter on the microcontroller, it was written using C language. This introduces some difficulties, as several mathematical operations are not supported in C. First of all, C does not support matrix multiplication which is an essential part of the Kalman filtering. Secondly the matrix inverse operation does not exist. The third challenge is to find the transpose of a matrix. Since the module should be used on-board a satellite, the memory usage should be as optimal as possible. Hence, it is not desirable to implement an entire math library in order to perform the matrix operations. Therefore only the necessary methods have been implemented on the microcontroller.

C supports two-dimensional arrays which looks very similar to a matrix. Using double arrays makes the code more complex. Therefore, matrices were transformed into one dimensional arrays before implementation. An example is given below.

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

Using 3 for-loops, matrix multiplication can be implemented. For each iteration in the Kalman filter, two matrices must be transposed. These matrices have dimensions 10×10 and 6×10, and can easily be transposed using two for-loops.

Computing the inverse of a matrix is only possible if the matrix is positive-definite. The matrix which is to be inverted in the Kalman filter has properties making it always invertible. It is a 6×6 symmetric matrix, meaning it is Hermetian, and since every eigenvalue in Hermetian matrices is positive, the matrix is positive definite. An efficient way to compute the inverse of a matrix is by performing an LU decomposition. The algorithm for inverse operation using LU decomposition can be found in [15]. The code in this book is optimized for personal computers implying minor changes to adapt it for a microcontroller.

3.3. Implementing the EQUEST in C

The EQUEST was also written using C language. Two challenges arise when the code is rewritten from MATLAB to C: one matrix inversion and one eigenvalue problem. However, it is only a matrix inverse of size 4-by-4. Since the matrix is so small, it is more efficient to invert it using the adjoint method than the LU decomposition. For larger matrices, the adjoint method tends to be computational costly as the number of operations increase as O(n!). To solve the EQUEST, it is necessary to identify the smallest eigenvalue and all eigenvectors of a symmetric 4-by-4 matrix. This is done by implementing the cyclic Jacobi method which returns all eigenvalues and the corresponding eigenvectors of the input matrix [16].

4. RESULTS AND SIMULATIONS

4.1. Extended Kalman Filter

The performance of the algorithms have been evaluated both in expended run time and number of arithmetic operations. The expended run time is found by setting a flag at the start of each cycle, and then resetting the flag after the execution. The run time of the EKF is about 200 milliseconds. By introducing a global counter in the algorithm, it is possible to detect how many arithmetic operations each cycle executes. The linearization contains quite a large amount of numerical operations. On average EKF required about 40 000 operations.

4.2. Extended Quaternion Estimation

Compared to the EKF, the EQUEST requires less matrix multiplications, and only a 4-by-4 matrix inversion. However, the eigenvalues and eigenvectors of a 4-by-4 matrix must be found. The EQUEST algorithm, does not require any linearization. The number of arithmetic operations for the EQUEST was found to be about 3200, which is only 8% of the EKF's operations. The run time for EQUEST is about 40ms. This means that EQUEST is approximately 5 times faster than the EKF. The linear prediction term will have a low-pass filtering effect, as high frequent changes in position will be suppressed. This is illustrated in Figure 4.1. The figure clearly indicates that the EQUEST with linear prediction is much smoother than without prediction.

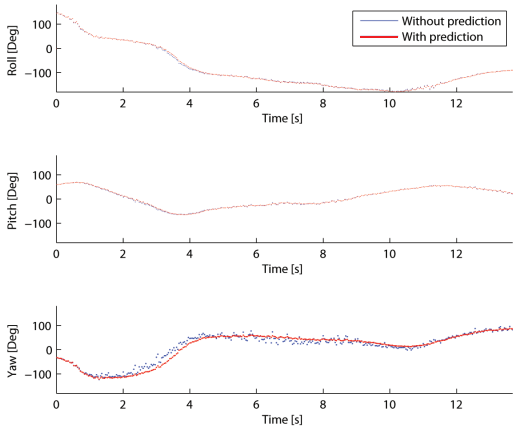


Figure 4.1: EQUEST with and without linear prediction

4.3. Experimental Comparison of the Two Methods

Hardware configuration

For testing purposes, a prototype was designed. The prototype was based on an CHIMU Micro AHRS IMU consisting of a three-axes accelerometer, magnetometer and gyroscope. The accelerometer can not be used onboard the satellite. However, due to test simplicity it was chosen as one of the vectorized measurements for the prototype. Data from the IMU is sent to an AVR ATMEGA2561 microcontroller, where both attitude estimation methods are implemented. A picture of the prototype is given in Figure 4.2.

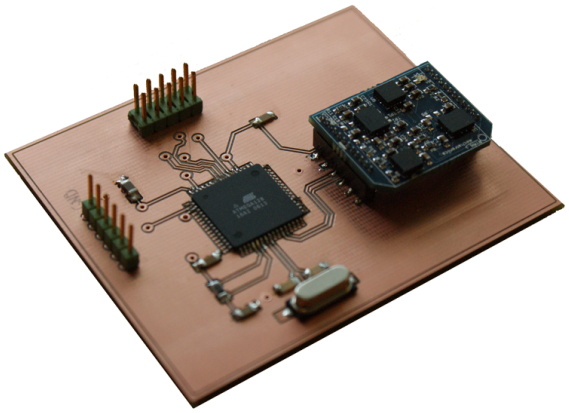


Figure 4.2: Designed prototype

The following experiments are done using the prototype running both methods simultaniously. The estimated attitude from the prototype were continiously sent to the computer for plotting through serial communication.

Tracking

To compare the performance of the newly developed EQUEST with the well-known EKF, both methods were implemented on the same microcontroller. They had the same input data and ran simultaneously sending the estimated attitude to a computer. Figure 4.3 indicates that both methods are able to track arbitrary rotations. The figure shows that the estimated orientation is almost identical for both EKF and EQUEST. However, it can be observed that the EQUEST has faster tracking than the EKF. EQUEST and EKF solve the attitude problem in quite different ways. Whereas EQUEST solves the problem in one iteration, the EKF iteratively calculates the solution. Hence, the EQUEST provides a faster estimation for quick orientational changes.

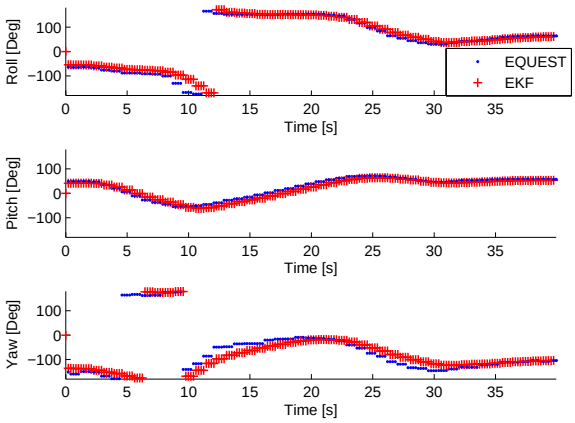


Figure 4.3: EKF vs EQUEST.

Start-up

One of the greatest differences between EKF and EQUEST is observed during the start-up phase. The EKF uses many iterations to converge towards the correct attitude. For each iteration, the EKF will improve the estimated orientation. The number of iterations used by the EKF will be dependent on its tuning parameters. In addition the start-up phase of the EKF will be greatly influenced by the dissipation of the vectorized measurements. Perpendicular measurement vectors will give the fastest start-up phase. However, the EQUEST will solve an eigenvalue problem to achieve the correct attitude in one iteration. An example of the initial start-up is showed in Figure 4.4. The figure clearly indicates the start-up differences of the methods.

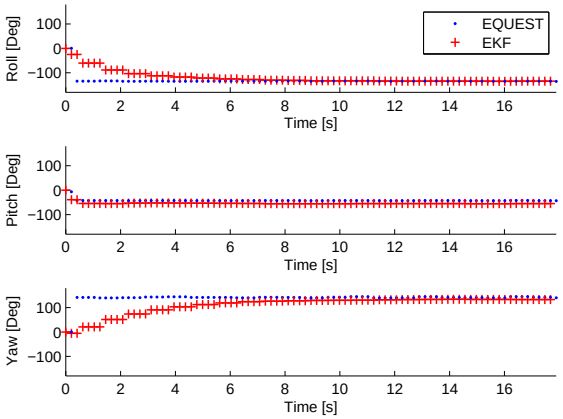


Figure 4.4: Start-up phase of the two algorithms.

Parallel input vectors

The performance of both methods are strictly dependent on the two directional vectors. For optimal results, the vectors should be close to perpendicular. However, in some cases the two vectorized measurements may be close to parallel. This is observed in Figure 4.5, where an accelerometer and a magnetometer were used as vectorized measurements for testing. The normalized acceleration measured in NED coordinates is

$$g = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad [4.1]$$

and the normalized magnetic field vector in Trondheim, Norway is

$$B = \begin{bmatrix} 0.2631 \\ 0.0086 \\ 0.9647 \end{bmatrix} \quad [4.2]$$

with the Down-axis being the dominating value. Both vectors are now close to parallel with the down axis. Performing a rotation around this axis, will cause problems for the EKF due to the mathematical formulation. The rotational matrix for rotations around the down axis can be written as

$$R_{down} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad [4.3]$$

Remember that the state update in EKF is given as

$$x_{k+1} = x_k + K(z_k - \begin{bmatrix} Rg^\top \\ RB^\top \end{bmatrix} - \begin{bmatrix} b_{v1} \\ b_{v2} \end{bmatrix}) \quad [4.4]$$

Here, the state update is a product of the Kalman gain, K, and the deviation from the measurements tracked by the rotational term. A rotation around the down axis is described above. Since the gravitational vector is close to parallel with the magnetic field, the rotational term in Equation (4.4) will now be described as

$$Rg^\top = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad [4.5]$$

$$RB^\top \approx \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \quad [4.6]$$

clearly, the rotational term fails to identify the rotational angle around the down-axis due to the magnetic field being almost parallel with the gravitational vector. As the measurements are B and g, there will be very small deviations from the measurements to the rotated position.

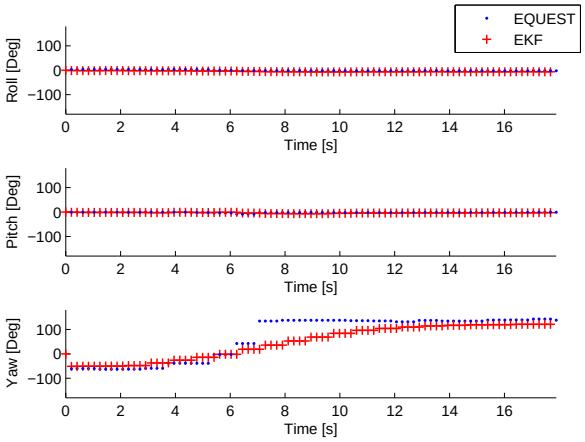


Figure 4.5: The two methods response to a rotation around the down-axis in the NED frame.

5. CONCLUSION

In this paper, two methods were derived, tested and compared. The QUEST algorithm has been developed further to EQUEST, to include non-vectorized measurements in the estimations. An Extended Kalman Filter was fitted for the CubeSat project at NTNU, and implemented with a sensor fusion model instead of a dynamical model of the satellite.

Testing indicates that both methods are able to estimate the orientation using a single microcontroller. The methods were compared theoretically and experimently. In contrast to EKF, the EQUEST is unable to estimate the sensor biases. However, the EQUEST uses only 8% of the arithmetical operations required by the EKF. The runtime of the EQUEST was measured to be more than 5 times faster than the EKF.

NUTS will most likely be designed with a magnetometer as one of the vectorized measurements. As the satellite will be controlled by magnetourqers, the local magnetic field will be greatly affected by the attitude control. Hence, it is important to seperate the attitude estimation and the attitude control. This is done through a turn-based switching between estimation and control. This implies that a short start-up phase for the estimation is preferred. As the EQUEST solves the attitude problem in one iteration, it is more attractive for satellites using this technique than the iterative EKF.

[1] Psiaki,M.L., "Attitude-Determination Filtering via Extended Quaternion Estimation", Journal of Guidance, Control and Dynamics, Vol.23, No.2, March-April 2000.

[2] Cheng,Y. and Shuster,M.D, "Robustness and Accuracy of the Quest Algorithm", presented as paper AAS 07-102, 17th AAS/AIAA Space Flight Mechanics Meeting, Sedona, Arizona, January 28 - February 2, 2007.

[3] Markley,L.F. and Mortari,D., "Quaternion Attitude Estimation Using Vector Observations", The Journal of the Astronautical Sciences, Vol.48, April-September 2000.

[4] CubeSat Design Specification Rev. 12, The CubeSat Program, Cal Poly SLO, Retrieved May 2011.

[5] NTNU Test Satellite, Retrieved May 2011, http://nuts.iet.ntnu.no

[6] Gravdahl,J.T., Skavhaug,A., Svartveit,K., Fauske.K.M. and Indergaard,F.M. "Three axis Attitude Determination and Control System for a picosatellite: Design and implementation", presented as paper IAC-03-A.5.07, IAC 2003.

[7] Svartveit,K., "Attitude determination of the NCUBE satellite", Master Thesis, Department of Engineering Cybernetics, June 2003.

[8] Ose, S.S., " Attitude determination for the Norwegian student satellite nCube", Master Thesis, Department of Engineering Cybernetics, June 2004.

[9] Sabatini,A.M., "Quaternion-Based Extended Kalman Filter for Determining Orientation by Inertial and Magnetic Sensing", IEEE Transaction on Biomedical Engineering, Vol.53, No. 7, July 2006.

[10] Crassidis,J.L., Markley,F.L., and Cheng,Y., "Survey of Nonlinear Attitude Estimation Methods", Journal of Guidance, Control, and Dynamics, Vol.30, No.1, January-February 2007.

[11] Ribeiro,M.I., "Kalman and Extended Kalman Filters: Concept, Derivation and Properties", Institute for Systems and Robotics, Lisboa, Portugal, February 2004.

[12] Nocedal,J. and Wright,S.J. "Numerical Optimization", Springer Series in Operations Research, 1999.

[13] Crassidis, Crassidis,J.L. and Markley,F.L., "Attitude Estimation Using Modified Rodrigues Parameters", Proceedings of the Flight Mechanics/Esitmation Theory Symposium, Cp-1996-3333, NASA Goddard Space Flight Center, Greenbelt, MD, 1996.

[14] Weisstein,E.W., "Least Squares Fitting", From MathWorld-A Wolfram Web Resource. mathworld.wolfram.com/LeastSquaresFitting.html

[15] Press,W.H., Teukolsky,S.A., Vetterling,W.T. and Flannery,B.P., "Numerical Recipes in C - The Art of Scientific Computation", Second Edition, Cambridge University Press, pp. 43-49, 1992.

[16] Jacobi,C.G.J., "Über ein leichtes Verfahren, die in der Theorie der Säkularstörungen vorkommenden Gleichungen numerisch aufzulösen", Crelle's Journal 30, pp. 51-94, 1846.

C ANSAT Workshop 2011

The NUTS project were presented on the ANSAT Workshop at Andøya Rocketrange in May 2011. Here follows the presentation, held by us, regarding the attitude determination system of the student satellite.

Attitude Determination
ANSAT Workshop 2011

Kristian L. Jenssen & Kaan H. Yabar
Andøya Rocketrange
19.05.2011

NTNU
Innovation and Creativity

Importance of Attitude Estimation

NTNU
Innovation and Creativity

Principle Attitude Determination

— Attitude can be determined using two vectorized measurements
— Sensors:
  - Sun sensor
  - Magnetometer
  - Gyro
  - Earth horizon sensor
  - Star sensor
— Estimation methods:
  - Kalman Filter (KF)
  - Quaternion Estimator (QUEST)

NTNU
Innovation and Creativity

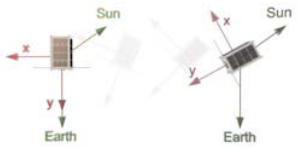
Extended Kalman Filter (EKF)

— Nonlinear model
— Sensor fusioning
— Model prediction using gyroscope

NTNU
Innovation and Creativity

QUEST

$$J(q) = \frac{1}{2}\sum_{j=1}^{n}\left\{\frac{1}{\sigma_j^2}(b_j - R_b^i(q)r_j)^\top(b_j - R_b^i(q)r_j)\right\}$$

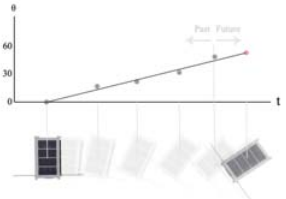


Extending with Gyroscope

$$J(q) = \frac{1}{2}\sum_{j=1}^{n}\left\{\frac{1}{\sigma_j^2}(b_j - R_b^i(q)r_j)^\top(b_j - R_b^i(q)r_j)\right\} + \frac{1}{2}(q - q_{gyro})^\top D(q - q_{gyro})$$

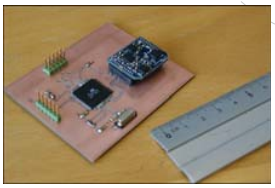
Extending with Linear Prediction

$$J(q) = \frac{1}{2}\sum_{j=1}^{n}\left\{\frac{1}{\sigma_j^2}(b_j - R_b^i(q)r_j)^\top(b_j - R_b^i(q)r_j)\right\} + \frac{1}{2}(q - q_{gyro})^\top D(q - q_{gyro}) + \frac{1}{2}(q - \hat{q}_{pre})^\top S(q - \hat{q}_{pre})$$

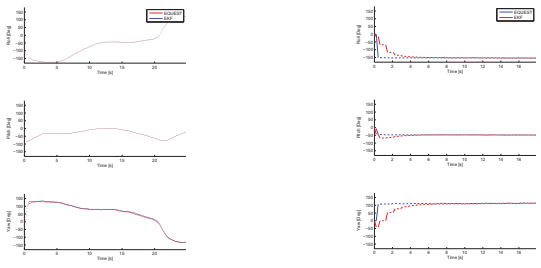


Implementation

- Implementation of ADCS
  - Matrix operations
  - Runtime
  - Power consumption
- Prototype



EKF vs EQUEST



EKF vs EQUEST

EKF

- Well known
- Estimates sensor bias
- Good filtering effect
- Requires about 40000 arithmetic operations

EQUEST

- Finds solution in one iteration
- Requires about 3200 arithmetic operations
- About 5 times faster than implemented EKF
- More intuitive tuning parameters than EKF

D Linearization

$$\begin{bmatrix} a_x^b \\ a_y^b \\ a_z^b \\ m_x^b \\ m_y^b \\ m_z^b \end{bmatrix} = \begin{bmatrix} R_n^b(q) \\ R_n^b(q) \end{bmatrix} \begin{bmatrix} g_x^n + a_{real,x}^b \\ g_y^n + a_{real,y}^b \\ g_z^n + a_{real,z}^b \\ m_{real,x}^b \\ m_{real,x}^b \\ m_{real,x}^b \end{bmatrix} + \begin{bmatrix} bias_{a,x} \\ bias_{a,y} \\ bias_{a,z} \\ bias_{m,x} \\ bias_{m,y} \\ bias_{m,z} \end{bmatrix} \tag{88}$$

Let:

$$g_x^n + a_{real,x}^b = a_1$$

$$g_y^n + a_{real,y}^b = a_2$$

$$g_z^n + a_{real,z}^b = a_3$$

$$m_x^b = m_1$$

$$m_y^b = m_2$$

$$m_z^b = m_3$$

$$num = (q_1^2 + q_2^2 + q_3^2 + q_4^2)^{\frac{3}{2}}$$

Then, the linearization is given as following:

$$\begin{aligned} \frac{\partial a_x^b}{\partial q_1} &= \frac{1}{num}(a_1 q_1^3 + 3a_1 q_1 q_2^2 + 3a_1 q_1 q_3^2 + a_1 q_1 q_4^2 - 2a_2 q_1 q_3 q_4 + 2a_2 q_2^3 \\ &+ 2a_2 q_2 q_3^2 + 2a_2 q_2 q_4^2 + 2a_3 q_1 q_2 q_4 + 2a_3 q_2^2 q_3 + 2a_3 q_3^3 + 2a_3 q_3 q_4^2) \end{aligned} \tag{89}$$

$$\begin{aligned} \frac{\partial a_x^b}{\partial q_2} &= \frac{1}{num}(-3a_1 q_1^2 q_3 - a_1 q_2^3 - a1q2q_3^2 - 3a_1 q_2 q_4^2 + 2a_2 q_1^3 \\ &+ 2a_2 q_1 q_3^2 + 2a_2 q_1 q_4^2 - 2a_2 q_2 q_3 q_4 - 2a_3 q_1^2 q_4 \\ &- 2a_3 q_1 q_2 q_3 - 2a_3 q_3^2 q_4 - 2a_3 q_4^3) \end{aligned} \tag{90}$$

$$\begin{aligned} \frac{\partial a_x^b}{\partial q_3} &= \frac{1}{num}(-3a_1 {q_1}^2 q_3 - a_1 q_2^2 q_3 - a_1 q_3^3 - 3a_1 q_3 q_4^2 + 2a_2 q_1^2 q_4 - 2a_2 q_1 q_2 q_3 \\ &+ 2a_2 q_2^2 q_4 + 2a_2 q_4^3 + 2a_3 q_1^3 + 2a_3 q_1 q_2^2 + 2a_3 q_1 q_4^2 + 2a_3 q_2 q_3 q_4) \end{aligned} \tag{91}$$

$$\frac{\partial a_x^b}{\partial q_4} = -\frac{1}{num}(-a_1q_1^2q_4 - 3a_1q_2^2q_4 - 3a_1q_3^2q_4 - a_1q_4^3 - 2a_2q_1^2q_3 + 2a_2q_1q_2q_4 - 2a_2q_2^2q_3 - 2a_2q_3^3 + 2a_3q_1^2q_2 + 2a_3q_1q_3q_4 + 2a_3q_2^3 + 2a_3q_2q_3^2) \quad (92)$$

$$\frac{\partial a_x^b}{\partial bias_{a,x}} = 1 \quad (93)$$

$$\frac{\partial a_x^b}{\partial bias_{a,y}} = \frac{\partial a_x^b}{\partial bias_{a,z}} = \frac{\partial a_x^b}{\partial bias_{m,x}} = \frac{\partial a_x^b}{\partial bias_{m,y}} = \frac{\partial a_x^b}{\partial bias_{m,z}} = 0 \quad (94)$$

$$\frac{\partial a_y^b}{\partial q_1} = \frac{1}{num}(2(a_1(q_1q_3q_4 + q_2^3 + q_2(q_3^2 + q_4^2)) + a_3(q_4 * (q_2^2 + q_3^2 + q_4^2) - q_1q_2q_3)) - a_2q_1(q_1^2 + 3q_2^2 + q_3^2 + 3q_4^2)) \quad (95)$$

$$\frac{\partial a_y^b}{\partial q_2} = \frac{1}{num}(2a_1q_1^3 + 2a_1q_1q_3^2 + 2a_1q_1q_4^2 + 2a_1q_2q_3q_4 + 3a_2q_1^2q_2 + a_2q_2^3 + 3a_2q_2q_3^2 + a_2q_2q_4^2 + 2a_3q_1^2q_3 - 2a_3q_1q_2q_4 + 2a_3q_3^3 + 2a_3q_3q_4^2) \quad (96)$$

$$\frac{\partial a_y^b}{\partial q_3} = \frac{1}{num}(-2a_1q_1^2q_4 - 2a_1q_1q_2q_3 - 2a_1q_2^2q_4 - 2a_1q_4^3 - a_2q_1^2q_3 - 3a_2q_2^2q_3 - a_2q_3^3 - 3a_2q_3q_4^2 + 2a_3q_1^2q_2 - 2a_3q_1q_3q_4 + 2a_3q_2^3 + 2a_3q_2q_4^2) \quad (97)$$

$$\frac{\partial a_y^b}{\partial q_4} = \frac{1}{num}(-2a_1q_1^2q_3 - 2a_1q_1q_2q_4 - 2a_1q_2^2q_3 - 2a_1q_3^3 + 3a_2q_1^2q_4 + 2a_2q_2^2q_4 + 3a_2q_3^2q_4 + a_2q_4^3 + 2a_3q_1^3 + 2a_3q_1q_2^2 + 2a3_3q_1q_3^2 - 2a_3q_2q_3q_4) \quad (98)$$

$$\frac{\partial a_y^b}{\partial bias_{a,y}} = 1 \quad (99)$$

$$\frac{\partial a_y^b}{\partial bias_{a,x}} = \frac{\partial a_y^b}{\partial bias_{a,z}} = \frac{\partial a_y^b}{\partial bias_{m,x}} = \frac{\partial a_y^b}{\partial bias_{m,y}} = \frac{\partial a_y^b}{\partial bias_{m,z}} = 0 \quad (100)$$

$$\frac{\partial a_z^b}{\partial q_1} = -\frac{1}{num}(2a_1q_1q_2q_4 - 2a_1q_2^2q_3 - 2a_1q_3^3 - 2a_1q_3q_4^2 + 2a_2q_1q_2q_3 + 2a_2q_2^2q_4 + 2a_2q_3^2q_4 + 2a_2q_4^3 + a_3q_1^3 + a_3q_1q_2^2 + 3a_3q_1q_3^2 + 3a_3q_1q_4^2) \quad (101)$$

$$\begin{aligned}\frac{\partial a_z^b}{\partial q_2} &= \frac{1}{num}(2(a_1(q_1^2q_4 - q_1q_2q_3 + q_3^2q_4 + q_4^3) + a_2(q_1^2q_3 \\ &+ q_1q_2q_4 + q_3^3 + q_3q_4^2)) - a_3q_2(q_1^2 + q_2^2 + 3(q_3^2 + q_4^2))) \end{aligned} \tag{102}$$

$$\begin{aligned}\frac{\partial a_z^b}{\partial q_3} &= \frac{1}{num}(2a_1q_1^3 + 2a_1q_1q_2^2 + 2a_1q_1q_4^2 - 2a_1q_2q_3q_4 + 2a_2q_1^2q_2 + 2a_2q_1q_3q_4 \\ &+ 2a_2q_2^3 + 2a_2q_2q_4^2 + 3a_3q_1^2q_3 + 3a_3q_2^2q_3 + a_3q_3^3 + a_3q_3q_4^2) \end{aligned} \tag{103}$$

$$\begin{aligned}\frac{\partial a_z^b}{\partial q_4} &= -\frac{1}{num}(-2a_1q_1^2q_2 + 2a_1q_1q_3q_4 - 2a_1q_2^3 - 2a_1q_2q_3^2 + 2a_2q_1^3 + 2a_2q_1q_2^2 \\ &+ 2a_2q_1q_3^2 + 2a_2q_2q_3q_4 - 3a_3q_1^2q_4 - 3a_3q_2^2q_4 - a_3q_3^2q_4 - a_3q_4^3) \end{aligned} \tag{104}$$

$$\frac{\partial a_z^b}{\partial bias_{a,z}} = 1 \tag{105}$$

$$\frac{\partial a_z^b}{\partial bias_{a,x}} = \frac{\partial a_z^b}{\partial bias_{a,y}} = \frac{\partial a_z^b}{\partial bias_{m,x}} = \frac{\partial a_z^b}{\partial bias_{m,y}} = \frac{\partial a_z^b}{\partial bias_{m,z}} = 0 \tag{106}$$

$$\begin{aligned}\frac{\partial m_x^b}{\partial q_1} &= \frac{1}{num}(m_1q_1^3 + 3m_1q_1q_2^2 + 3m_1q_1q_3^2 + m_1q_1q_4^2 - 2m_2q_1q_3q_4 \\ &+ 2m_2q_2^3 + 2m_2q_2q_3^2 + 2m_2q_2q_4^2 + 2m_3q_1q_2q_4 \\ &+ 2m_3q_2^2q_3 + 2m_3q_3^3 + 2m_3q_3q_4^2) \end{aligned} \tag{107}$$

$$\begin{aligned}\frac{\partial m_x^b}{\partial q_2} &= \frac{1}{num}(-3m_1q_1^2q_3 - m_1q_2^3 - m1q2q_3^2 - 3m_1q_2q_4^2 + 2m_2q_1^3 \\ &+ 2m_2q_1q_3^2 + 2m_2q_1q_4^2 - 2m_2q_2q_3q_4 - 2m_3q_1^2q_4 \\ &- 2m_3q_1q_2q_3 - 2m_3q_3^2q_4 - 2m_3q_4^3) \end{aligned} \tag{108}$$

$$\begin{aligned}\frac{\partial m_x^b}{\partial q_3} &= \frac{1}{num}(-3m_1{q_1}^2q_3 - m_1q_2^2q_3 - m_1q_3^3 - 3m_1q_3q_4^2 + 2m_2q_1^2q_4 - 2m_2q_1q_2q_3 \\ &+ 2m_2q_2^2q_4 + 2m_2q_4^3 + 2m_3q_1^3 + 2m_3q_1q_2^2 + 2m_3q_1q_4^2 + 2m_3q_2q_3q_4) \end{aligned} \tag{109}$$

$$\begin{aligned}\frac{\partial m_x^b}{\partial q_4} &= -\frac{1}{num}(-m_1q_1^2q_4 - 3m_1q_2^2q_4 - 3m_1q_3^2q_4 - m_1q_4^3 - 2m_2q_1^2q_3 + 2m_2q_1q_2q_4 \\ &- 2m_2q_2^2q_3 - 2m_2q_3^3 + 2m_3q_1^2q_2 + 2m_3q_1q_3q_4 + 2m_3q_2^3 + 2m_3q_2q_3^2) \end{aligned} \tag{110}$$

$$\frac{\partial m_x^b}{\partial bias_{m,x}} = 1 \tag{111}$$

$$\frac{\partial m_x^b}{\partial bias_{m,y}} = \frac{\partial m_x^b}{\partial bias_{m,z}} = \frac{\partial m_x^b}{\partial bias_{a,x}} = \frac{\partial m_x^b}{\partial bias_{a,y}} = \frac{\partial m_x^b}{\partial bias_{a,z}} = 0 \tag{112}$$

$$\begin{aligned} \frac{\partial m_y^b}{\partial q_1} &= \tfrac{1}{num} \quad (2(m_1(q_1q_3q_4 + q_2^3 + q_2(q_3^2 + q_4^2)) \\ &+ \quad m_3(q_4 * (q_2^2 + q_3^2 + q_4^2) - q_1q_2q_3)) \\ &- \quad m_2q_1(q_1^2 + 3q_2^2 + q_3^2 + 3q_4^2)) \end{aligned} \tag{113}$$

$$\begin{aligned} \frac{\partial m_y^b}{\partial q_2} &= \frac{1}{num}(2m_1q_1^3 + 2m_1q_1q_3^2 + 2m_1q_1q_4^2 + 2m_1q_2q_3q_4 \\ &+ 3m_2q_1^2q_2 + m_2q_2^3 + 3m_2q_2q_3^2 + m_2q_2q_4^2 \\ &+ 2m_3q_1^2q_3 - 2m_3q_1q_2q_4 + 2m_3q_3^3 + 2m_3q_3q_4^2) \end{aligned} \tag{114}$$

$$\begin{aligned} \frac{\partial m_y^b}{\partial q_3} &= \frac{1}{num}(-2m_1q_1^2q_4 - 2m_1q_1q_2q_3 - 2m_1q_2^2q_4 - 2m_1q_4^3 - m_2q_1^2q_3 - 3m_2q_2^2q_3 \\ &- m_2q_3^3 - 3m_2q_3q_4^2 + 2m_3q_1^2q_2 - 2m_3q_1q_3q_4 + 2m_3q_2^3 + 2m_3q_2q_4^2) \end{aligned} \tag{115}$$

$$\begin{aligned} \frac{\partial m_y^b}{\partial q_4} &= \frac{1}{num}(-2m_1q_1^2q_3 - 2m_1q_1q_2q_4 - 2m_1q_2^2q_3 - 2m_1q_3^3 + 3m_2q_1^2q_4 + 2m_2q_2^2q_4 \\ &+ 3m_2q_3^2q_4 + m_2q_4^3 + 2m_3q_1^3 + 2m_3q_1q_2^2 + 2m_3q_1q_3^2 - 2m_3q_2q_3q_4) \end{aligned} \tag{116}$$

$$\frac{\partial m_y^b}{\partial bias_{m,y}} = 1 \tag{117}$$

$$\frac{\partial m_y^b}{\partial bias_{m,x}} = \frac{\partial m_y^b}{\partial bias_{m,z}} = \frac{\partial m_y^b}{\partial bias_{a,x}} = \frac{\partial m_y^b}{\partial bias_{a,y}} = \frac{\partial m_y^b}{\partial bias_{a,z}} = 0 \tag{118}$$

$$\begin{aligned} \frac{\partial m_z^b}{\partial q_1} &= -\frac{1}{num}(2m_1q_1q_2q_4 - 2m_1q_2^2q_3 - 2m_1q_3^3 - 2m_1q_3q_4^2 + 2m_2q_1q_2q_3 + 2m_2q_2^2q_4 \\ &+ 2m_2q_3^2q_4 + 2m_2q_4^3 + m_3q_1^3 + m_3q_1q_2^2 + 3m_3q_1q_3^2 + 3m_3q_1q_4^2) \end{aligned} \tag{119}$$

$$\frac{\partial m_z^b}{\partial q_2} = \frac{1}{num}(2(m_1(q_1^2 q_4 - q_1 q_2 q_3 + q_3^2 q_4 + q_4^3) + m_2(q_1^2 q_3 + q_1 q_2 q_4 + q_3^3 + q_3 q_4^2)) - m_3 q_2(q_1^2 + q_2^2 + 3(q_3^2 + q_4^2))) \quad (120)$$

$$\frac{\partial m_z^b}{\partial q_3} = \frac{1}{num}(2m_1 q_1^3 + 2m_1 q_1 q_2^2 + 2m_1 q_1 q_4^2 - 2m_1 q_2 q_3 q_4 + 2m_2 q_1^2 q_2 + 2m_2 q_1 q_3 q_4 + 2m_2 q_2^3 + 2m_2 q_2 q_4^2 + 3m_3 q_1^2 q_3 + 3m_3 q_2^2 q_3 + m_3 q_3^3 + m_3 q_3 q_4^2) \quad (121)$$

$$\frac{\partial m_z^b}{\partial q_4} = -\frac{1}{num}(-2m_1 q_1^2 q_2 + 2m_1 q_1 q_3 q_4 - 2m_1 q_2^3 - 2m_1 q_2 q_3^2 + 2m_2 q_1^3 + 2m_2 q_1 q_2^2 + 2m_2 q_1 q_3^2 + 2m_2 q_2 q_3 q_4 - 3m_3 q_1^2 q_4 - 3m_3 q_2^2 q_4 - m_3 q_3^2 q_4 - m_3 q_4^3) \quad (122)$$

$$\frac{\partial m_z^b}{\partial bias_{m,z}} = 1 \quad (123)$$

$$\frac{\partial m_z^b}{\partial bias_{m,x}} = \frac{\partial m_z^b}{\partial bias_{m,y}} = \frac{\partial m_z^b}{\partial bias_{a,x}} = \frac{\partial m_z^b}{\partial bias_{a,y}} = \frac{\partial m_z^b}{\partial bias_{a,z}} = 0 \quad (124)$$

E Matrix Inverse Using LU Decomposition

LU decomposition is a procedure for composing a square matrix A with dimension N into a product of a lower triangular matrix L and an upper triangular matrix U :

$$A = LU \tag{125}$$

Writing explicitly:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots \\ a_{21} & a_{22} & a_{23} & \dots \\ a_{31} & a_{32} & a_{33} & \\ \vdots & \vdots & & \ddots \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & \dots \\ l_{21} & l_{22} & 0 & \dots \\ l_{31} & l_{32} & l_{33} & \\ \vdots & \vdots & & \ddots \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots \\ 0 & u_{22} & u_{23} & \dots \\ 0 & 0 & u_{33} & \\ \vdots & \vdots & & \ddots \end{bmatrix} \tag{126}$$

$$A = \begin{bmatrix} l_{11}u_{11} & l_{11}u_{12} & l_{11}u_{13} & \dots \\ l_{21}u_{11} & l_{21}u_{12} + l_{22}u_{22} & l_{21}u_{13} + l_{22}u_{23} & \dots \\ l_{31}u_{11} & l_{31}u_{12} + l_{32}u_{22} & l_{31}u_{13} + l_{32}u_{23} + l_{33}u_{33} & \\ \vdots & \vdots & & \ddots \end{bmatrix} \tag{127}$$

Three types of equations occurs:

$$i < j \; l_{i1}u_{1j} + l_{i2}u_{2j} + \cdots + l_{ii}u_{ij} = a_{ij} \tag{128}$$

$$i = j \; l_{i1}u_{1j} + l_{i2}u_{2j} + \cdots + l_{ii}u_{jj} = a_{ij} \tag{129}$$

$$i > j \; l_{i1}u_{1j} + l_{i2}u_{2j} + \cdots + l_{ij}u_{jj} = a_{ij} \tag{130}$$

This gives N^2 equations and $N^2 + N$ unknowns and can be solved using Crout's method by just arranging the equations in a certain order [23]. LU decomposition can be used to find the inverse of a matrix by solving linear set. Let say that it is desirable to solve for X in following linear set:

$$AX = B \tag{131}$$

If B is set to be the identity matrix, then the solution X is the inverse of A:

$$AA^{-1} = I = B \tag{132}$$

This means that with B being identity matrix, the inverse of A can be found by solving a linear set. Following procedure can be used to solve the linear set using LU decomposition:

Writing the decomposition:

$$AX = (LU)X = L(UX) = B \tag{133}$$

Solve for Y:

$$LY = B \tag{134}$$

Solve for X:

$$UX = Y \tag{135}$$

The great advantage here is that U and L is triangular, which makes the equations trivial to solve.

F Prototype Userguide

F.1 Programming the Microcontroller on the Prototype Using AVR Studio and AVR Dragon

To program the microcontroller, it is important to have knowledge about four of the menus in the AVR Studio. These menus can be selected from the toolbar, showed in Figure 26.

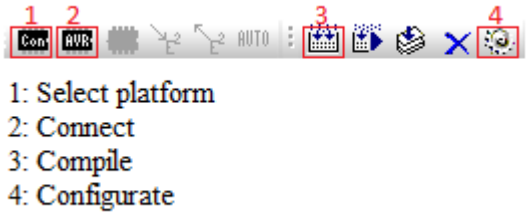


Figure 26: The toolbar to compile, configure and connect the AVR microcontroller.

In order to compile the code correctly, it is important to use the configurations menu. In the configurations menu, the correct device and the operating frequency should be selected. This is displayed in Figure 27. When the configurations menu is selected properly, the code can be compiled by pushing the Compile button in the toolbar showed in the figure above. After the compilation, a .hex file is created and placed in the Output File Directory, this file should be flashed to the microcontrollers memory using the connect button.

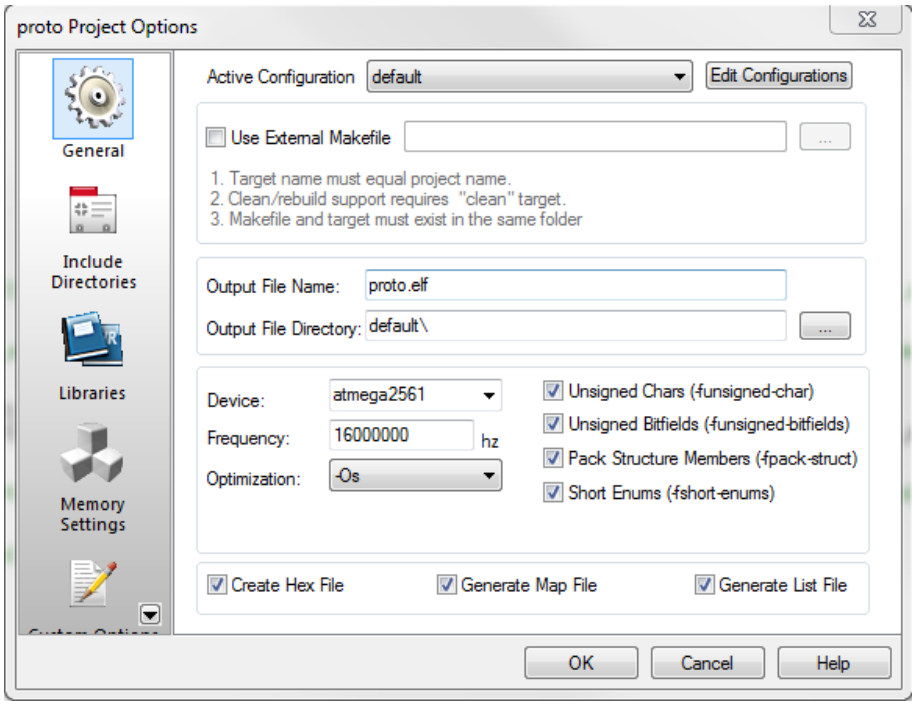


Figure 27: Configure project.

Before using the connect button, the correct platform must be selected using the select platform button. Figure 28 illustrates the connection using an AVR dragon on USB port. After selecting platform, the microcontroller can be flashed.

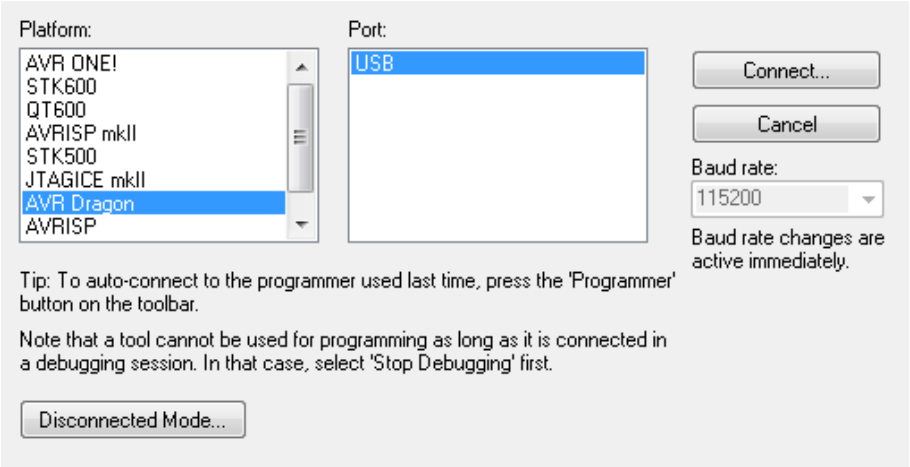


Figure 28: Platform selection.

Clicking the connect button opens the menu displayed in Figure 29. As displayed in the figure, there are eight tabs. In the "Main" tab, the correct device should be chosen and confirmed by clicking the "Read Signature" button. The "Programming Mode and Target Settings" should be selected to "JTAG mode".

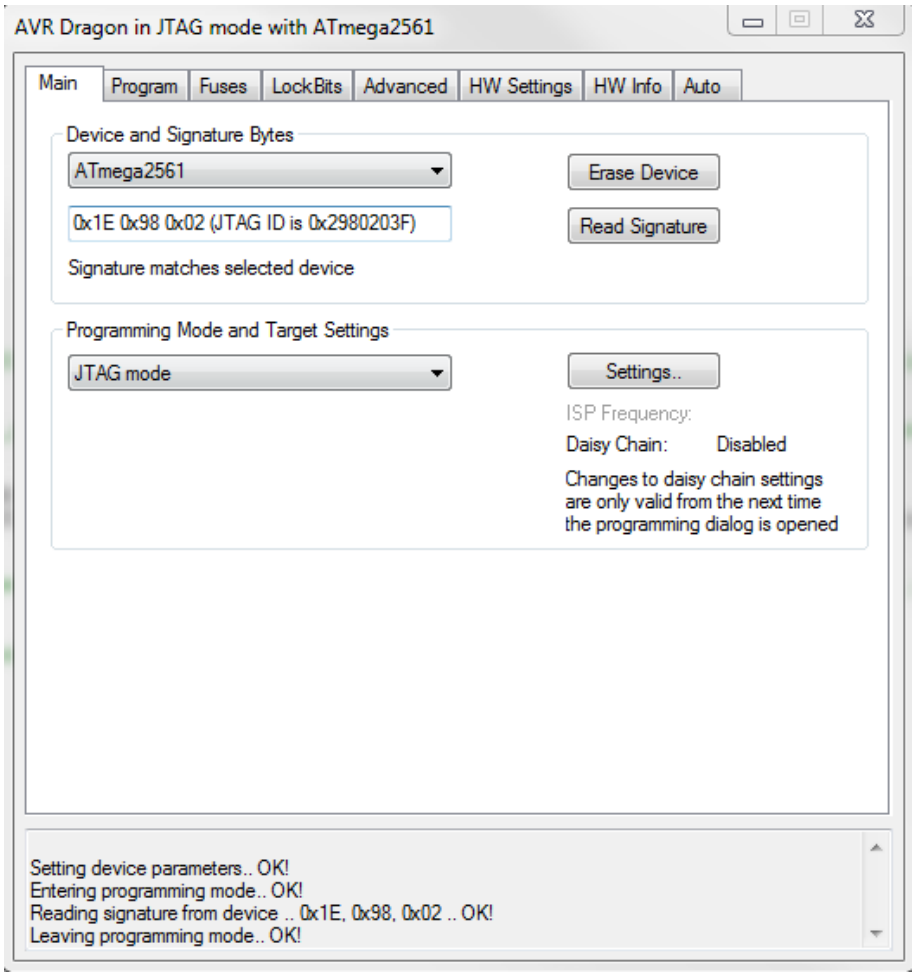


Figure 29: JTAG connection.

Under the "Fuses" tab, "OCDEN" and "JTAGEN" checkboxes should be selected. Since the prototype uses an external crystal running at 16 MHz, "Ext. Crystal Osc. 8.0- MHz" must be chosen in the "SUT_CKSEL" as given in Figure 30.

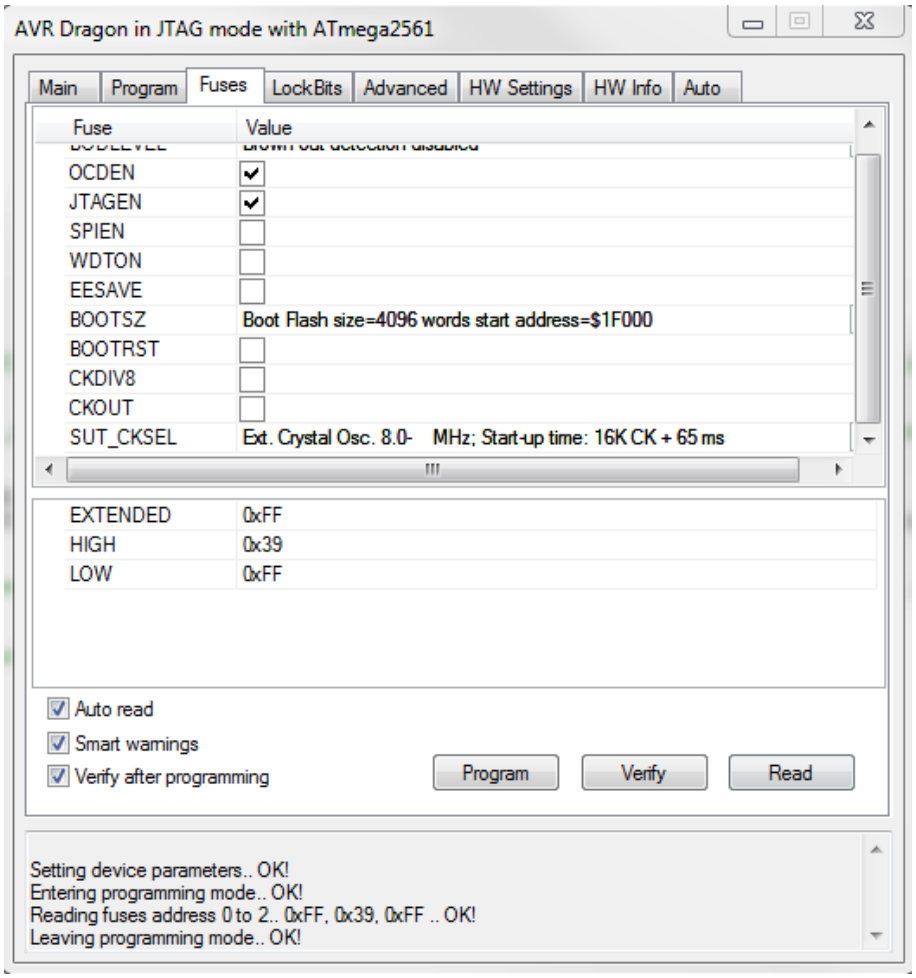


Figure 30: Fuses.

After completing the settings explained above, the microcontroller should be ready for flashing. In the "Program" tab, the .hex file created earlier should be selected as "Input HEX File", in the "Flash" section. The microcontroller is flashed by pressing the "Program" button.

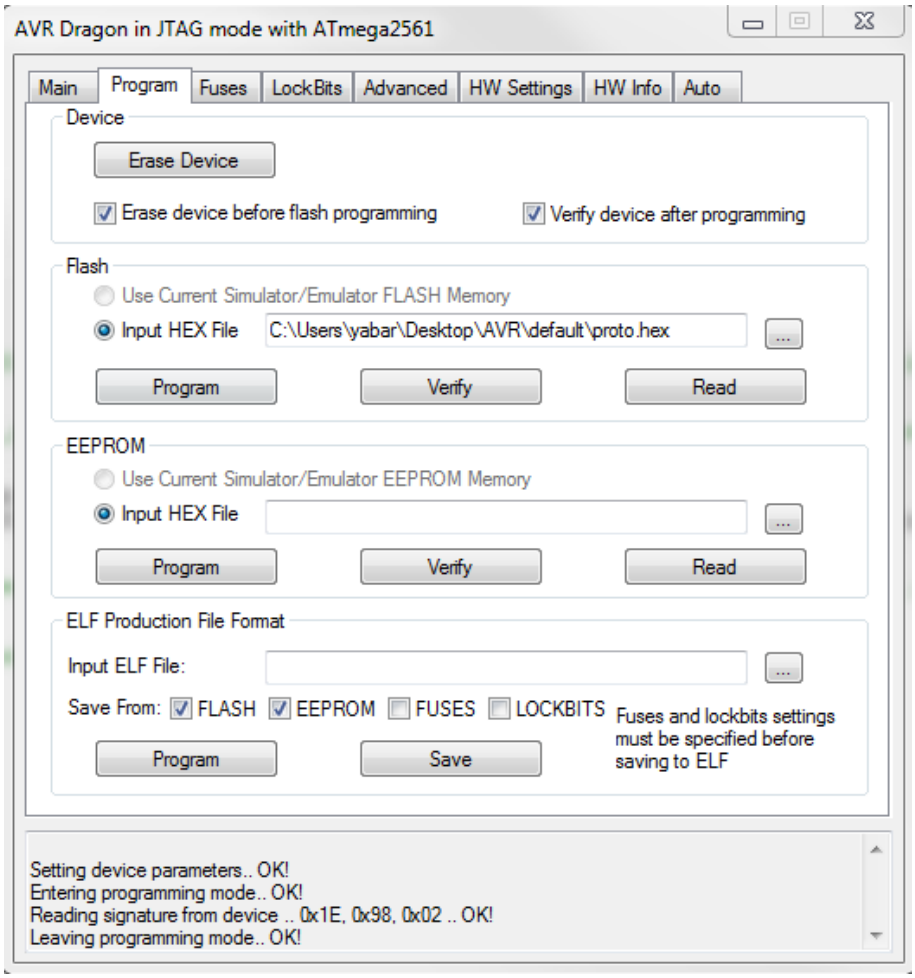


Figure 31: Programming the microcontroller.

F.2 Output Attitude Data from Prototype to MATLAB

The data from prototype is sent to computer through the usb to serial connection. It is possible to get two different illustrations of the output:

1. Display attitude as a 3D cube.
2. Display attitude with graph (roll, pitch and yaw).

To display the attitude as a 3D cube, attitude estimated either by EKF or EQUEST must be sent to computer. It is important to note that the MATLAB code is only designed for extracting data from one of the methods. For EQUEST the microcontroller must be programmed such that the only remaining data sent to computer is:

```
printf ("%d,%d,%d,%d,\n", (int) (equest_x[0]*10000),
 (int) (equest_x[1]*10000), (int) (equest_x[2]*10000), (int) (equest_x[3]*10000));
```

For EKF, the line above can be replaced with:

```
printf ("%d,%d,%d,%d,\n", (int) (x[0]*10000),
 (int) (x[1]*10000), (int) (x[2]*10000), (int) (x[3]*10000));
```

All other prints must be commented out. Since data from only one of the methods can be displayed, running both methods will be superfluous. If you want to run only the EKF replace all lines containing

```
method=1;
```

with

```
method=0;
```

For EQUEST, do the opposite. The MATLAB file that should be executed is named "microplot.m".

If it is desirable to display the attitude as roll, pitch and yaw, attitude data from both EKF and EQUEST should be extracted simultaneously. Again, remove all prints and add following two lines:

```
printf ("%d,%d,%d,%d,%d\n", (int) (x[0]*10000),
 (int) (x[1]*10000), (int) (x[2]*10000), (int) (x[3]*10000),1);
printf ("%d,%d,%d,%d,%d\n", (int) (equest_x[0]*10000),
 (int) (equest_x[1]*10000), (int) (equest_x[2]*10000), (int) (equest_x[3]*10000),2);
```

For graph, the MATLAB file "liveplotKALQUEST.m" should be executed. Note that the serial port number in the MATLAB file must be consistent with the port connected to the prototype.

References

[1] CubeSat Design Specification Rev. 12, The CubeSat Program, Cal Poly SLO, Retrieved May 2011.

[2] "NTNU Test Satellite", Retrieved May 2011, http://nuts.iet.ntnu.no

[3] Gravdahl,J.T., Skavhaug,A., Svartveit,K., Fauske.K.M. and Indergaard,F.M. "Three axis Attitude Determination and Control System for a picosatellite: Design and implementation", presented as paper IAC-03-A.5.07, IAC 2003.

[4] Svartveit, K., "Attitude determination of the NCUBE satellite", Master Thesis, Department of Engineering Cybernetics, June 2003.

[5] Ose, S.S., " Attitude determination for the Norwegian student satellite nCube", Master Thesis, Department of Engineering Cybernetics, NTNU, June 2004.

[6] Rohde, J., " Kalman filter for attitude determination of student satellite", Master Thesis, Department of Engineering Cybernetics, NTNU, July 2007.

[7] Sabatini,A.M., "Quaternion-Based Extended Kalman Filter for Determining Orientation by Inertial and Magnetic Sensing", IEEE Transaction on Biomedical Engineering, Vol.53, No. 7, July 2006.

[8] Markley, L.F. and Mortari, D., "Quaternion Attitude Estimation Using Vector Observations", The Journal of the Astronautical Sciences, Vol.48, April-September 2000.

[9] Psiaki, M.L., "Attitude-Determination Filtering via Extended Quaternion Estimation", Journal of Guidance, Control and Dynamics, Vol.23, No.2, March-April 2000.

[10] Egeland, O. and Gravdahl, J.T, "Modeling and Simulation for Automatic Control", Marine Cybernetics, pp. 218-223, 2002.

[11] Hall, C.D., "Attitude Measurements", Spacecraft Attitude Dynamics and Control, March 18, 2003.

[12] Hamilton, W.R., "On Quaternions, or On a New System of Imaginaries in Algebra", Philosophical Magazine, Vol.25, pp. 489-495, 1844.

[13] Crassidis, J.L., Markley, F.L., and Cheng,Y., "Survey of Nonlinear Attitude Estimation Methods", Journal of Guidance, Control, and Dynamics, Vol.30, No.1, January-February 2007.

[14] Crassidis, Crassidis, J. L. and Markley F. L., "Attitude Estimation Using Modified Rodrigues Parameters", Proceedings of the Flight Mechanics/Esitmation Theory Symposium, Cp-1996-3333, NASA Goddard Space Flight Center, Greenbelt, MD, 1996.

[15] Vik,B., "Euler Angles from Quaternions", Integrated Satellite and Intertial Navigation Systems, Department of Engineering Cybernetics, NTNU, Trondheim, Norway.

[16] Ribeiro, M.I., "Kalman and Extended Kalman Filters: Concept, Derivation and Properties", Institute for Systems and Robotics, Lisboa, Portugal, February 2004.

[17] Bhanderi,D.D.V., "Attitude Estimation From Magnetometer and Earth-Albedo-Corrected Coarse Sun Sensor Measurements", Ph.D. Thesis, Department of Control Engineering Aalborg University, August, 2005.

[18] Appel, P., "Spacecraft Attitude Determination with Earth Albedo Corrected Sun Sensor Measurement", Acta Astronautica, IAA, 2004.

[19] Nocedal, J. and Wright, S.J. "Numerical Optimization", Springer Series in Operations Research, 1999.

[20] Weisstein,E.W., "Least Squares Fitting", From MathWorld-A Wolfram Web Resource. mathworld.wolfram.com/LeastSquaresFitting.html

[21] "Atmega128 Datasheet", Retrieved February 10, 2011, http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

[22] "Atmega2560/2561 Datasheet", Retrieved February 12, 2011, http://www.atmel.com/dyn/resources/prod_documents/doc2549.pdf

[23] Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P., "Numerical Recipes in C - The Art of Scientific Computation", Second Edition, Cambridge University Press, pp. 43-49, 1992.

[24] Jacobi, C.G.J., "Über ein leichtes Verfahren, die in der Theorie der Säkularstörungen vorkommenden Gleichungen numerisch aufzulösen", Crelle's Journal 30, pp. 51-94, 1846.

[25] Tudor, Z., "Design and Implementation of Attitude Control for 3-Axis Magnetic Coil Stabilization of a Spacecraft", Master Thesis, Department of Engineering Cybernetics, NTNU, June 2011.