



UNIVERSITY OF AGDER

MODELING, DESIGN AND EXPERIMENTAL STUDY FOR A QUADCOPTER SYSTEM CONSTRUCTION

by

Øyvind Magnussen
Kjell Eivind Skjønhaug

SUPERVISORS:
MORTEN OTTESTAD
HAMID REZA KARIMI
ANNE MÜLLER

"This Master's Thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education.

UNIVERSITY OF AGDER, 2011
DEPARTMENT OF ENGINEERING
FACULTY OF TECHNOLOGY AND SCIENCE

Abstract

The popularity of quadcopters are increasing as the sensors and control systems are getting more advanced. The quadcopter is naturally unstable, has a complex dynamic model and six degrees of freedom. Even with four motors it is underactuated, and cannot move translatable without rotating about one of its axes. There are many commercially quadcopters for sale, but was not considered as it would result in a backwards design process where the design is decided before the needs. To meet the university's lab facilities a quadcopter was designed to fit a single board RIO from National Instruments. This is the preferred platform used by the university and it has high performance characteristics. The quadcopter requires an extensive control system in order to fly. With many parameters it is difficult to implement and tune a regulator for such a system. In addition to an even more complex regulator many sensors are needed in order to make the quadcopter autonomous. For the attitude estimation sensor fusion is required to get a robust and reliable measurement. Based on the dynamic model the sensors needed was chosen. They were tested one by one before implementation in a control system. A tilt regulator was able to stabilize the quadcopter around one axis in a test rig. Building a quadcopter for an educational purpose, and to discover and resolve its complexity is an experimental project, as this has never been done at this university. The final quadcopter concept is well suited for further experimental work.

Acknowledgment

We would like to use this opportunity to thank the University for allowing us to write this thesis, which we have really enjoyed working on. Would also like to thank our appointed supervisors from the university, which have been very helpful when there were need for assistance. The faulty deserves credit for their willingness to fund our project and allowing us to use parts available.

Would like to thank the laboratory personnel for their great assistance during the build, and to allow us to set up an permanent office at their location. Their fast response and manufacturing time have allowed the project to progress without unnecessary delay.

Contents

Abstract	3
Preface	1
List of Figures	5
List of Tables	7
List of Abbreviations	8
List of Symbols	9
1 Introduction	10
1.1 About Quadcopter	11
2 Reference Geometry	12
2.1 Coordinate System	12
2.2 Rotation Matrix	13
2.3 Direction Cosine Matrix	14
2.4 Motor Reference	16
2.5 Coordinate Summary	16
3 Dynamic Model	17
3.1 Forces	17
3.1.1 Mass and Acceleration	17
3.1.2 Gravity	17
3.1.3 Propeller Thrust	17
3.1.4 Force Drag	18
3.1.5 Disturbance	18
3.1.6 Force Equations	18
3.2 Moments	19
3.2.1 Thrust Moment	19
3.2.2 Motor Inertia Moment	19
3.2.3 Quadcopter Inertia	20
3.2.4 Moment Equations	20
3.2.5 Euler rates	20
3.3 Dynamics Summary	21
4 Sensors and Components	22
4.1 Inertial Measurement Unit	22
4.1.1 IMU Test	23
4.2 RPM	26
4.3 Voltage and Current Measurement	28
4.4 Rangefinder	29
4.5 ESC	29
4.6 XBee Transceiver	30

4.7	Control board	31
4.8	Batteries	31
4.9	Propeller	32
4.10	Motor	33
4.11	Motor and Propeller testing	33
4.11.1	Theoretical Performance	33
4.11.2	Test bench	34
4.11.3	Force vs Torque	37
4.11.4	ESC	39
5	Attitude and Position Estimation	40
5.1	Attitude Estimation	40
5.1.1	DCM from gyro signals	42
5.1.2	Re-normalize the DCM	42
5.1.3	Drift Correction	43
5.1.4	Euler angles	44
5.1.5	IMU testing	45
5.2	Position Estimation	46
5.2.1	Global Positioning System	46
5.2.2	Internal Navigation System	46
5.2.3	Visual Recognition	46
5.3	Summary	47
6	Construction	48
6.1	Mechanical	48
6.1.1	Prototype design	48
6.1.2	Revised Design	49
6.2	Test Rig	52
6.3	Electrical	53
6.3.1	Brakeout Board	53
7	Control System	54
7.1	Yaw-controller	54
7.2	Phi-Theta controller	55
7.3	Current controller	57
8	Programming	58
8.1	Data flow	58
8.2	XBee	61
8.2.1	CRC	61
8.2.2	Data flow	61
8.2.3	XBee data conversion	62
8.3	RC Transmitter	64
8.4	IMU	66
8.5	RPM	69
8.6	Volt and Current	71
8.7	Control System	72
8.7.1	Yaw controller	72
8.7.2	Phi and Theta controller	73
8.8	ESC	74
8.9	Initialization of the Control System	76
8.9.1	ESC Calibration	76
8.9.2	Motor Test	77
8.9.3	Gyro Calibration	77
8.10	Remote observer	77
8.11	Programming Summary	78

9 Full Test	79
9.1 Test Bench	79
9.2 Free Flight	81
10 Conclusion	82
Bibliography	84
Appendices	86
A Parts	87
B Mechanical Drawings; Quadcopter	88
C Programming	95
C.1 UART Tx	95
C.2 UART Rx	95
C.3 SPI VI	96
C.4 ADIS16400	97
C.5 IMU Scaling	99
C.6 RPM	100
C.7 Yaw controller disabled	100
C.8 Theta controller mapping	101
C.9 ESC RPM to PWM	101
C.10 ESC Saturation	102
C.11 ESC Calibration	102
C.12 Motor Test	105

List of Figures

2.1	<i>The blue coordinates are the navigation frame, while the black is the body coordinates</i>	12
2.2	<i>Euler angles</i>	13
2.3	<i>Initial position</i>	14
2.4	<i>+90 degrees around X_n</i>	15
2.5	<i>Additional -30 degrees around Z_n</i>	15
2.6	<i>Sketch of where the motors are mounted and direction of propeller movement</i>	16
4.1	<i>ADIS16400 IMU from Analog Device</i>	22
4.2	<i>The test bench used as angle reference for the IMU</i>	23
4.3	<i>Theta of the static response for the non calibrated IMU</i>	24
4.4	<i>Theta of the static response for the calibrated IMU</i>	24
4.5	<i>Theta of the dynamic response for the IMU</i>	25
4.6	<i>Hall effect sensor</i>	26
4.7	<i>The principle of hall effect [21]</i>	27
4.8	<i>Hall effect sensor circuit diagram</i>	27
4.9	<i>Hall effect sensor mounted on the quadcopter</i>	28
4.10	<i>Current and voltage sensor for the battery</i>	28
4.11	<i>Sharp rangefinder</i>	29
4.12	<i>XBee transceiver</i>	30
4.13	<i>NI sbRIO-9631 controller card</i>	31
4.14	<i>Batteries used, to the right 5000 mAh and to the left 450 mAh</i>	32
4.15	<i>APC 12x6 propeller</i>	32
4.16	<i>Hyperion ZS3020-10</i>	33
4.17	<i>Torque transducer 0-10Nm from AEP</i>	34
4.18	<i>Test setup</i>	35
4.19	<i>Test results of current vs RPM</i>	36
4.20	<i>Test results of Force and Torque</i>	37
4.21	<i>Linear relationship of the Force and Torque</i>	38
4.22	<i>RPM at a given PWM</i>	39
5.1	<i>Rotating locally with the sum of 0 degrees, model in the pictures is downloaded from "http://www.topfreemodel.com/Models/Transport/Boeing-757-aircraft-SolidWorks-Models.html"</i>	40
5.2	<i>Block diagram of the attitude estimation</i>	41
5.3	<i>Drift correction test</i>	45
5.4	<i>Dynamic response</i>	45
5.5	<i>How to measure distance with the camera</i>	46
6.1	<i>To the left, the first motor bracket while to the right the revised one</i>	49
6.2	<i>Design progress</i>	50
6.3	<i>Arm with out cuts</i>	50
6.4	<i>Fem Analysis</i>	51
6.5	<i>Final design of quadcopter</i>	52
6.6	<i>The test rig</i>	52
6.7	<i>The brakeout boards</i>	53

7.1	<i>Main view of the control system, grayed out blocs are not in use</i>	54
7.2	<i>Block diagram of the phi and theta controller</i>	55
7.3	<i>Principle drawing of the ΔF with angular velocity set to -1 rad/sec</i>	56
8.1	<i>Block diagram of the dataflow.</i>	58
8.2	<i>Flow chart of the control system</i>	59
8.3	<i>Overview of the FPGA program</i>	60
8.4	<i>Overview of the RT-target program</i>	60
8.5	<i>XBee implementation on the FPGA</i>	61
8.6	<i>Scaling of the 8-bit numbers</i>	63
8.7	<i>XBee conversion in the control loop</i>	63
8.8	<i>RC transmitter readings on the FPGA</i>	64
8.9	<i>RC transmitter pwm measurement</i>	64
8.10	<i>RC conversion in the control loop</i>	65
8.11	<i>Illustration figure of a SPI data exchange referred from the quadcopter</i>	66
8.12	<i>IMU reading on the FPGA</i>	67
8.13	<i>IMU and its conversion in the control loop</i>	68
8.14	<i>Attitude estimation subVI</i>	68
8.15	<i>Illustration figure of the time period P for the rpm measurement.</i>	69
8.16	<i>LabVIEW code for the rpm measurement.</i>	69
8.17	<i>RPM in the control loop</i>	70
8.18	<i>Volt and current readings on the FPGA</i>	71
8.19	<i>Volt and current in the control loop</i>	71
8.20	<i>Control system in the control loop</i>	72
8.21	<i>Yaw controller realized in LabVIEW</i>	72
8.22	<i>Mapping of values to the phi regulator</i>	73
8.23	<i>Regulator implemented in LabVIEW</i>	73
8.24	<i>ESC in the control loop</i>	74
8.25	<i>ESC control on the FPGA</i>	75
8.26	<i>ESC control on the FPGA</i>	75
8.27	<i>ESC calibration procedure</i>	76
8.28	<i>ESC control on the FPGA</i>	76
8.29	<i>ESC control on the FPGA</i>	77
9.1	<i>Quadcopter placed in the test bench</i>	79
9.2	<i>Tilt test from 0.2 rad to 0 rad</i>	80
9.3	<i>Four impacts where the controller tries to hold zero radians</i>	80

List of Tables

4.1	Phase voltage sequence for a brushless DC motor [28]	26
4.2	Motor torque results, test 2	35
4.3	Motor torque results, test 3	35
8.1	Data sent from the base station to the quadcopter, with its real values extrema	62
8.2	Data sent from the quadcopter to the base station, with its real values extrema	62
8.3	Data sent from the RC transmitter to the quadcopter, with its real values extrema	64
8.4	Data read from the IMU and its address	66

List of Abbreviations

CRC	-	Cyclic Redundancy Check
DCM	-	Direction Cosine Matrix
ESC	-	Electronic Speed Controller
FEM	-	Finite Element Method
FPGA	-	Field Programmable Gate Array
IMU	-	Inertial Measurement Unit
MV	-	Measured values
PWM	-	Pulse Width Modulation
RC	-	Remote Control
rpm	-	Revolutions per minute
Rx	-	Receive
sbRIO	-	Single Board Reconfigurable Input/Output
SP	-	Setpoints for the controller
SPI	-	Serial Peripheral Interface
Tx	-	Transmitt
UART	-	Universal Asynchronous Receiver/Transmitter
VI	-	Virtual Instrument, a LabVIEW program
subVI	-	Subroutine in a LabVIEW VI
lipo	-	lithium polymer

List of Symbols

b	-	Referred to body frame, onboard.
n	-	Referred to navigation frame, fixed
σ	-	General angle
$\dot{\sigma}$	-	General angular velocity
ϕ	-	Euler angle for the roll
θ	-	Euler angle for the pitch
ψ	-	Euler angle for the yaw
\underline{C}_u^v	-	Transformation matrix from u-frame to v-frame

1. Introduction

Autonomous quadcopters are a part of the future. They have been researched in many applications and some are capable of doing aggressive maneuvers [18]. The main drawback with those projects is however the price of the lab facilities. The quadcopter is a perfect mechatronic product and the University of Agder decided to dive into its complex knowledge. In this thesis a quadcopter is built from scratch, where the sensors and equipment are carefully selected.

The objective is to build a fully functioning quadcopter sensor platform. The quadcopter has to be easy to program, have autonomous capabilities and be able to lift a payload of a couple of kilograms. In addition to the physical elements the main aspects of the quadcopter has to be discovered. With four motors the quadcopter has many degrees of freedom, but yet it is underactuated so it cannot move translatable without first rotate around one of its axes. The quadcopter is also naturally unstable. This in addition to the challenging dynamics of the quadcopter, a stable flight requires many sensors with high accuracy combined with a fast and robust control system. The sensor data, especially from the inertial measurement unit which estimates the attitude, has to be merged in order to get the most out of the system. As described in this report it is important to understand the main challenges of the quadcopter in order to control it.

Prior to the master thesis there was a preliminary project, the quadcopter investigation [14]. In this project The main design were founded and the major components were selected. The report did also contain a literature chapter were it was focusing on the main difficulties with the quadcopter. The main problems while constructing a quadcopter were often described as a lack of good sensors and their reliability. Based on this research, a lot of time during this thesis were spent on the sensors and how to get accurate and reliable signals.

The report have been divided in different chapters and ordered in the sequence that makes it easy to follow the work progres. It starts with the coordinate systems and dynamics of the quadcopter. This enhances the knowledge of dynamics in 3D space. Then the sensors are selected and tested. Based on the selected sensors a method of estimating the attitude of the quadcopter is described. With the sensors tested and selected the construction can incorporate the sensors in the process. Once the entire platform is complete a regulator for tilt control is designed. The regulator and sensors are all programmed on the quadcopter which then is ready for testing. The testing starts in a test bench that tests the stability of the regulator for tilt around one of the axis. When the quadcopter is stable in the rig a free flight test is performed. Every discovery is summarized in the conclusion chapter.

1.1 About Quadcopter

A quadcopter is an aircraft with four main propellers that provides lift. The four motors which drives the propeller are fixed onto an frame that often is made like an cross, made from two beams.

The quadcopter is almost exclusively used for small drones or radio controlled units today, but there has been built full scale aircrafts. In 1922 an French engineer, Etienne Edmond Oemichen, built and flew his second helicopter design. This helicopter, "Oehmichen No.2", had an X-shaped frame with an rotor at each arm. With this design, several records were set. In 1924 The model were the first rotaty wing to complete a flight to fly 1 kilometer. [17]

The quadcopter is an popular concept for an drone, because of its properties. The major advantages for the quadcopter is its ability to hover, and takeoff vertically. This makes the quadcopter useful for many tasks and allows it to be operated in nearly any environments.

While an conventional helicopter design, which one main rotor and one tail rotor, have the same properties, the quadcopter is advantageous in several fields. The quadcopter have no moving parts except for the rotating motors and propeller that rotate about fixed axes, while the conventional helicopter have a complex hub to make it possible to start an translative movement. The quadcopter is also less prone to vibration and it is more robust when it comes to the placement of placement of center of gravity. It is also easier, due to the small size, to cover the rotors and thus making it more safe to fly indoors

The typical design for an quadcopter has as stated earlier, no moving parts. The motors and theirs propellers are mounted onto the airframe, this configuration gives the quadcopter an interesting dynamic. Since the propellers are fixed to the frame, the only way to induce a lateral motion is to tilt the entire airframe. To tilt the quadcopter, the moment about one axis have to alter. To change the moment, one or both motor have to either increase or lower its amount of thrust. If just one of the motors adjust the thrust, there will be an unbalance in the rotational force about the yaw axis, therefore the motors would have to increase and lower the thrust with the same amount to keep the quadcopter still about the yaw axis. This is also the reason for why the motors turning the same direction is mounted opposite to each other, to be able to facility this operation. If the motors had been spinning opposite direction to each other, one could not regulate tilt, without inducing another tilt or yaw moment.

Unlike a conventional helicopter, the quadcopter does not have an tail rotor to help controlling the yaw motion. Instead the quadcopter rely on controlling the torque forces from the motors to control the yaw forces. Since the quadcopter is built with four motors where two are spinning clockwise (cw) and the other two spinning counterclockwise (ccw), the torque from the motors will cancel each other. This is only valid as long as each pair of motors have the same torque, if one of the motors gets a reduced torque the aircraft will start to spin in the air.

2. Reference Geometry

In this chapter coordinate systems and reference geometry is defined. The chapter also defines motor position and its direction of rotation. The last part of this chapter is a method to transform from one coordinate system to another.

2.1 Coordinate System

When the quadcopter is navigating in three dimensional space there are two different coordinate systems present. One is the body coordinate system, indexed 'b', which is affected by the motors. The other is the navigation frame, indexed 'n', were forces like gravitation has influence. The body coordinate system will move along with the quadcopter, while the navigation coordinate system is the reference point for the quadcopter. The reference coordinate system can be placed anywhere, but has to be fixed once the quadcopter starts moving. One assumption for the coordinate systems is the neglected curvature of the earth. A quadcopter has a limited area to navigate in and this assumption will not affect any result. In avionics the Z-axis is normally pointing towards the earth. To ease the comparison with other projects the Z-axis is also pointing downwards in this project. The quadcopter is able to rotate around its own axes with an angular velocity. This angular velocity is denoted as $\dot{\sigma}$ with an index for the corresponding axis.

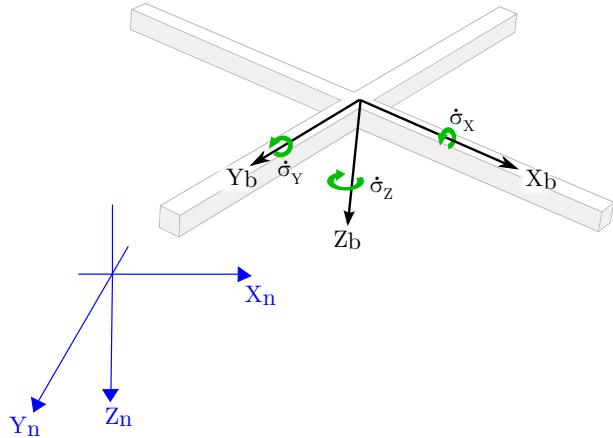


Figure 2.1: The blue coordinates are the navigation frame, while the black is the body coordinates

In order to get the orientation of the quadcopter in terms of angles, the quadcopters coordinate system has to be linked to the navigation coordinates. The way this is done is by the implementation of Euler angles. There are three Euler angles, ϕ , θ and ψ , known as roll, pitch and yaw. This notation is often used in avionics.

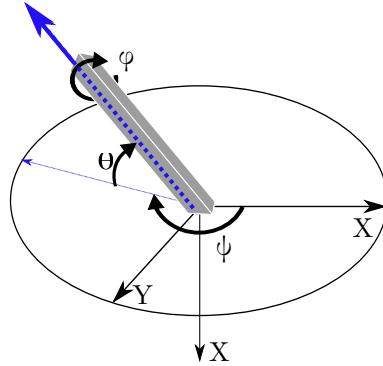


Figure 2.2: Euler angles

2.2 Rotation Matrix

To make a dynamic model, the forces and moments acting on the quadcopter must be found. The forces must also be oriented so they influence the quadcopter in the correct way, no matter how quadcopter is oriented. Gravity will always point the Z-direction of the navigation frame, but will have vector components referred to the quadcopter dependent of its orientation. A vector can be oriented to any position with three successive rotations. In order to overcome the gravity vector the quadcopters orientation must be rotated, but not in the physical way, to determine how much force the motors has to put out to keep it hovering. The rotations can be done one at a time. Equations 2.2 show the matrices that can be used to rotate about a single axis.

$$\underline{\underline{R}}_{\sigma_x}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \underline{\underline{R}}_{\sigma_y}(\theta) = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \underline{\underline{R}}_{\sigma_z}(\psi) = \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

To be able to calculate the total conversion from the body frame to the navigation frame all the three matrixes can be multiplied together. The result is a complete rotation matrix (Equation 2.2) [11]. The motor force vector can be expressed in the navigation frame if multiplied with this rotation matrix. Once the force is represented in the navigation frame the total thrust needed to hover can be found.

$$\underline{\underline{C}}_b^n = \underline{\underline{R}}_{\sigma_x}(\phi) \underline{\underline{R}}_{\sigma_y}(\theta) \underline{\underline{R}}_{\sigma_z}(\psi)$$

$$\underline{\underline{C}}_b^n = \begin{bmatrix} \cos \theta \cos \psi & \cos \theta \sin \psi & -\sin \theta \\ \sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & \sin \phi \cos \theta \\ \sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi & \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi & \cos \phi \cos \theta \end{bmatrix} \quad (2.2)$$

The calculated matrix gives the conversion from body to navigation frame. It is also needed in some cases to calculate from navigation to body frame. This can be done by transposing the result $\underline{\underline{C}}_b^n$ matrix, $\underline{\underline{C}}_b^b = \underline{\underline{C}}_b^n T$

[15].

$$\underline{\underline{C}}_n^b = \begin{bmatrix} \cos \theta \cos \psi & \sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi & \sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi \\ \cos \theta \sin \psi & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi \\ -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta \end{bmatrix} \quad (2.3)$$

2.3 Direction Cosine Matrix

The rotation matrix is also known as the direction cosine matrix (DCM). The direction cosine matrix is a 3x3 transformation matrix. It describes the relation between the attitude of the quadcopter and the earth reference frame [3]. Its name comes from the fact that the element in the i'th row and j'th column, r_{ij} , represents the cosine of the angle between the i-axis of the reference frame and the j-axis of the body frame. Another property of the DCM is that the columns represents the unit vector in the body axis projected along the reference axis, and the rows represents the unit vectors for the reference frame. For instance the first row describes the X_n -axis with respect to the quadcopters X-, Y- and Z-axis.

$$C_b^n = \begin{array}{c|ccc} & \text{Quadcopter Z} & \text{Quadcopter Y} & \text{Quadcopter X} \\ \hline & r_{xz} & r_{xy} & r_{xx} & \text{Earth X} \\ & r_{yz} & r_{yy} & r_{yx} & \text{Earth Y} \\ & r_{zx} & r_{zy} & r_{zz} & \text{Earth Z} \end{array} \quad (2.4)$$

An example of two successive rotations is described in the figures below. The DCM matrix to the right of the figure is the corresponding DCM.

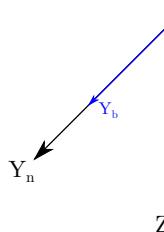


Figure 2.3:
Initial position

$$\underline{\underline{C}}_b^n = \begin{bmatrix} X_b & Y_b & Z_b \\ \downarrow & \downarrow & \downarrow \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \leftarrow \begin{array}{l} X_n \\ Y_n \\ Z_n \end{array} \quad (2.5)$$

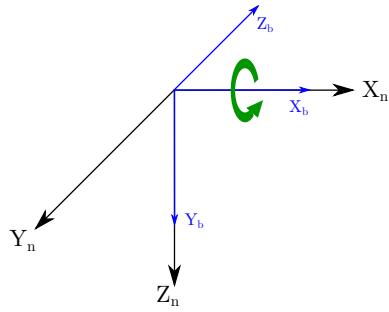


Figure 2.4:
+90 degrees around X_n

$$\underline{\underline{C}}_b^n = \begin{bmatrix} X_b & Y_b & Z_b \\ \downarrow & \downarrow & \downarrow \\ 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{array}{l} \leftarrow X_n \\ \leftarrow Y_n \\ \leftarrow Z_n \end{array} \quad (2.6)$$

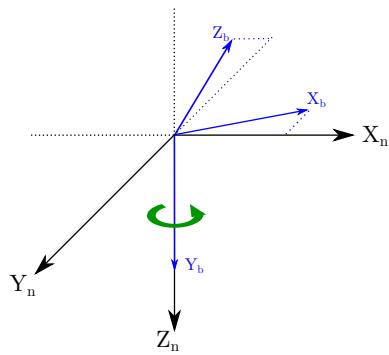


Figure 2.5:
Additional -30 degrees around Z_n

$$\underline{\underline{C}}_b^n = \begin{bmatrix} X_b & Y_b & Z_b \\ \downarrow & \downarrow & \downarrow \\ 0.866 & 0 & -0.5 \\ -0.5 & 0 & -0.866 \\ 0 & 1 & 0 \end{bmatrix} \begin{array}{l} \leftarrow X_n \\ \leftarrow Y_n \\ \leftarrow Z_n \end{array} \quad (2.7)$$

With the theory that states that the rows and columns are unit vectors and that the rows are describing the earth reference frame with respect to the quadcopters coordinates and vice versa, the three states are described:

Figure 2.3:

In the initial position, each Euler angle is zero. Comparing C_b^n from equation 2.5 with the text from equation 2.4 the quadcopters X-axis is only to have an X-component from the earth reference frame, and no component from the Y- and Z-axis. The Y_b has only the Y_n and the Z_b has only Z_n . Comparing from the navigation frames point of view gives the same result.

Figure 2.4:

The quadcopter is then rotated with a +90 degrees roll. According to equation 2.6 and equation 2.4 the X-axis is still along the X_n axis. This is also as figure 2.4 shows. The equation says that the quadcopters Y-axis is parallel to the Z_b axis. The quadcopters Z-axis is parallel to the Y_n but in negative direction. Figure 2.4 confirms equation 2.6.

Figure 2.5:

In the last figure the quadcopter is in addition rotated -30 degrees around the Z_n axis. The new transformation matrix is found by multiplication of 2.6 and the DCM for the last angle. Equation 2.7 shows that the quadcopters Y-axis is parallel to the Z-axis of the earth coordinate system.

The X_b and Z_b has no component of the Z_n hence they are located in the $X_n Y_n$ -plane. X_b has a component of 0.866 from X_n . The Y_n component of X_b is in the negative direction of Y_n and with a magnitude

of 0.5. Equation 2.7 also states that the X_n component consists of a component of both X_b and Z_b .

By inspection of each term of the matrices, it can be verified that the figure is a drawing of the corresponding matrix.

Each vectors R in the body frame can be transformed into the earth frame by multiplying the vector with the DCM, referred as C_b^n .

$$\underline{R}_{navigation} = \underline{\underline{C}}_b^n \cdot \underline{R}_{body}$$

2.4 Motor Reference

The motors rotation and position is also of significant. Motor one and two are placed along the X-axis and rotate in the clockwise direction. Motor three and four are placed on the Y-axis and rotate in the counterclockwise direction (Figure 2.6).

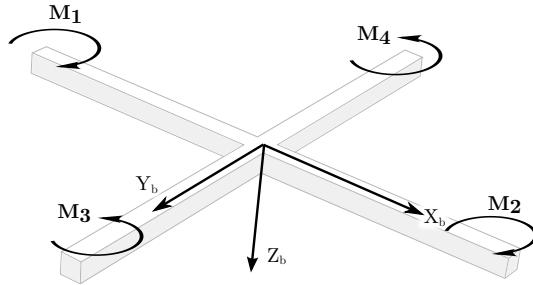


Figure 2.6: Sketch of where the motors are mounted and direction of propeller movement

2.5 Coordinate Summary

The coordinate systems has been fully defined (chapter 2.1). The difference between the body and navigation coordinate system is also defined and how to convert from one system to the other. The motors have been set at their representative position on the axis in chapter 2.4, also the direction of rotation for the motors are described. To be able to derive the dynamic equations of the quadcopter this is all needed.

3. Dynamic Model

3.1 Forces

To start setting up an dynamic model, the most basic thing is to map which forces is acting on the module. In this section the forces like gravity and thrust from propeller will be explained how it affect an quadcopter. The position for the motors can be seen in figure 2.6 in chapter 2.4. Each motor have its own force acting in negative direction and also an moment that is acting in the opposite direction of the rotation of the motors. Quaternion is not taken into consideration.

3.1.1 Mass and Acceleration

Newton first law gives us.

$$\underline{F}_{mass} = m_{tot} \cdot \begin{bmatrix} \ddot{X}_b \\ \ddot{Y}_b \\ \ddot{Z}_b \end{bmatrix}$$

3.1.2 Gravity

The gravity vector, F_g , is pulling the quadcopter in positive z-direction in the navigation frame. It can be represented in the body frame by multiplying the vector by the transformation matrix:

$$\begin{aligned} \underline{F}_{gb} &= m_{tot} \cdot \underline{C}_n^b \cdot \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \\ \underline{F}_{gb} &= m_{tot} \cdot g \cdot \begin{bmatrix} \sin(\theta) \\ -\sin(\phi) \cdot \cos(\theta) \\ \cos(\phi) \cdot \cos(\theta) \end{bmatrix} \end{aligned}$$

3.1.3 Propeller Thrust

The force from the propellers are assumed that always will be parallel to the Z-axis in the body frame, hence there will be no force in the X- and Y-direction. This assumption yields only if the propeller is a rigid body with no flapping. In real life this is not the case, but the flapping is neglected from the equation. The generated thrust is a function of the propeller dimensions, and the rpm.

$$\underline{F}_{thrust} = \begin{bmatrix} 0 \\ 0 \\ F_{M1} + F_{M2} + F_{M3} + F_{M4} \end{bmatrix}$$

3.1.4 Force Drag

As an object moves through the air it will experience drag forces that will work in the opposite direction the object is traveling. This drag force is generated by the friction in the air, this effect has the same properties of an dampening device in a system.

The Drag force is determined by the formula [2]:

$$F_{Drag} = \frac{1}{2} \cdot \rho \cdot v^2 \cdot Sw \cdot C_D \quad (3.1)$$

where:

ρ = Air density

v = Velocity

Sw = Reference area

C_D = Drag coefficient

The problem with equation 3.1.4 is the parameters Sw and C_D . Due to the advanced aerodynamic from the thrust, there was no good way to determine the drag. There was planned was to make some physical tests to gather some more data on the subject.

Unfortunately the constrained amount of time, did not allow this subject to be further explored and the drag were unfortunately not correctly implemented to the dynamic model.

3.1.5 Disturbance

Other forces like the Coriolis force from the earth, wind and Euler forces are considered as a disturbance, summarized as $F_{disturbance}$.

3.1.6 Force Equations

The forces explained earlier is summed together to find the total forces acting on the quadcopter:

$$\begin{aligned} m_{tot} \cdot \ddot{r} &= F_{gravity} - F_{thrust} + F_{disturbance} + F_{drag} \\ \ddot{r} &= \frac{1}{m_{tot}} (F_{gravity} - F_{thrust} + F_{disturbance} + F_{drag}) \\ \begin{bmatrix} \ddot{X}_b \\ \ddot{Y}_b \\ \ddot{Z}_b \end{bmatrix} &= \frac{1}{m_{tot}} \left(g \cdot \begin{bmatrix} \sin(\theta) \\ -\sin(\phi) \cdot \cos(\theta) \\ \cos(\phi) \cdot \cos(\theta) \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ F_{M1} + F_{M2} + F_{M3} + F_{M4} \end{bmatrix} + F_{disturbance} + F_{drag} \right) \end{aligned}$$

The acceleration of the body frame is converted into navigation frame by using the transformation matrix:

$$\begin{bmatrix} \ddot{X}_n \\ \ddot{Y}_n \\ \ddot{Z}_n \end{bmatrix} = \underline{\underline{C}}_b^n \cdot \begin{bmatrix} \ddot{X}_b \\ \ddot{Y}_b \\ \ddot{Z}_b \end{bmatrix}$$

3.2 Moments

The forces that have been explained in the last section decide the movement in translative position. The forces described also act on the quadcopter and creates a moment about the different axes. Generally the following equation describe how the quadcopter changes its attitude:

$$\underline{M} = \underline{\underline{I}} \dot{\underline{\sigma}} + \dot{\underline{\sigma}} \times (\underline{\underline{I}} \cdot \dot{\underline{\sigma}})$$

\underline{M} = moment from the thrust + moment from the motor-propeller inertia

$$\underline{M} = \underline{M}_{thrust} + \underline{M}_{motorinertia}$$

$\underline{\underline{I}}$ = Inertia Matrix

3.2.1 Thrust Moment

The \underline{M}_{thrust} is a part of the external moments acting on the system, described by the propeller thrust F_{thrust} and the distance ' l' from CG to the center of the propeller:

$$\begin{aligned}\underline{M}_{thrust} &= \begin{bmatrix} M_X \\ M_Y \\ M_Z \end{bmatrix} \\ M_X &= (F_4 - F_3) \cdot l \\ M_Y &= (F_2 - F_1) \cdot l \\ M_Z &= (-F_1 - F_2 + F_3 + F_4) \cdot Tq\end{aligned}$$

Tq is a constant which converts the thrust into moment.

3.2.2 Motor Inertia Moment

The motor and propeller angular velocity is causing a moment when the quadcopter is being tilted. Since the motor is only spinning parallel to the Z-axis the force can be described as:

$$\begin{aligned}\underline{M} &= \underline{\omega}_{motor} \times \dot{\underline{\sigma}} \cdot \underline{\underline{I}}_{motor} \\ \underline{M}_{motorinertia} &= \begin{bmatrix} 0 \\ 0 \\ \omega_{Pi} \end{bmatrix} \times \begin{bmatrix} \dot{\sigma}_x \\ \dot{\sigma}_y \\ \dot{\sigma}_z \end{bmatrix} \cdot \underline{\underline{I}}_{motor} \\ \underline{M}_{motorinertia} &= \begin{bmatrix} -\dot{\sigma}_y \cdot \omega_{Pi} \\ \dot{\sigma}_x \cdot \omega_{Pi} \\ 0 \end{bmatrix} \cdot \underline{\underline{I}}_{motor}\end{aligned}$$

Where the 'i' denotes the motor number. There should be noted that motors three and four rotates counterclockwise and thereby a negative angular velocity. Summing for all four motor gives:

$$\underline{M}_{Pi} = \begin{bmatrix} -\dot{\sigma}_y \cdot (\omega_{P1} + \omega_{P2} + \omega_{P3} + \omega_{P4}) \\ \dot{\sigma}_x \cdot (\omega_{P1} + \omega_{P2} + \omega_{P3} + \omega_{P4}) \\ 0 \end{bmatrix} \cdot \underline{\underline{I}}_{motor}$$

3.2.3 Quadcopter Inertia

The inertia matrix is given by:

$$\underline{\underline{I}} = \begin{bmatrix} I_{11} & -I_{12} & -I_{13} \\ -I_{21} & I_{22} & -I_{23} \\ -I_{31} & -I_{32} & I_{33} \end{bmatrix}$$

3.2.4 Moment Equations

$$\begin{aligned} \underline{M}_{thrust} + \underline{M}_{motorinertia} &= \underline{\underline{I}} \cdot \ddot{\underline{\sigma}} + \dot{\underline{\sigma}} \times \underline{\underline{I}} \cdot \dot{\underline{\sigma}} \\ \underline{\underline{I}} \cdot \ddot{\underline{\sigma}} &= \underline{M}_{thrust} + \underline{M}_{motorinertia} - \dot{\underline{\sigma}} \times \underline{\underline{I}} \cdot \dot{\underline{\sigma}} \\ \ddot{\underline{\sigma}} &= \underline{\underline{I}}^{-1} \cdot (\underline{M}_{thrust} + \underline{M}_{motorinertia} - \dot{\underline{\sigma}} \times \underline{\underline{I}} \cdot \dot{\underline{\sigma}}) \end{aligned}$$

Rearranging the matrices gives:

$$\begin{bmatrix} \ddot{\sigma}_X \\ \ddot{\sigma}_Y \\ \ddot{\sigma}_Z \end{bmatrix} = \begin{bmatrix} \frac{(F_4 - F_3) \cdot l}{I_{11}} \\ \frac{(F_2 - F_1) \cdot l}{I_{22}} \\ \frac{(-F_1 - F_2 + F_3 + F_4) \cdot Tq}{I_{33}} \end{bmatrix} + \begin{bmatrix} -\dot{\sigma}_Y \cdot \frac{1}{I_{11}} (\omega_{P1} + \omega_{P2} + \omega_{P3} + \omega_{P4}) \\ \dot{\sigma}_X \cdot \frac{1}{I_{22}} (\omega_{P1} + \omega_{P2} + \omega_{P3} + \omega_{P4}) \\ 0 \end{bmatrix} \cdot \underline{\underline{I}}_{motor} - \begin{bmatrix} \dot{\sigma}_Y \cdot \dot{\sigma}_Z \cdot \frac{(I_{33} - I_{22})}{I_{11}} \\ \dot{\sigma}_X \cdot \dot{\sigma}_Z \cdot \frac{(I_{11} - I_{33})}{I_{22}} \\ \dot{\sigma}_X \cdot \dot{\sigma}_Y \cdot \frac{(I_{22} - I_{11})}{I_{33}} \end{bmatrix}$$

3.2.5 Euler rates

The local angular velocities are related to the Euler angles by this relation

$$\begin{aligned} \begin{bmatrix} \dot{\sigma}_X \\ \dot{\sigma}_Y \\ \dot{\sigma}_Z \end{bmatrix} &= \underline{R}_X \cdot \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \underline{R}_X \underline{R}_Y \cdot \begin{bmatrix} 0 \\ \dot{\sigma} \\ 0 \end{bmatrix} + \underline{R}_X \underline{R}_Y \underline{R}_Z \cdot \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \\ \begin{bmatrix} \dot{\sigma}_X \\ \dot{\sigma}_Y \\ \dot{\sigma}_Z \end{bmatrix} &= \begin{bmatrix} \cos(\theta) \cos(\psi) & \sin(\psi) & 0 \\ -\cos(\theta) \sin(\psi) & \cos(\psi) & 0 \\ \sin(\theta) & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \end{aligned}$$

From this the derivative of the Euler angles can be found:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \frac{1}{\cos(\theta)} \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \cos(\theta) \sin(\psi) & \cos(\theta) \cos(\psi) & 0 \\ -\sin(\theta) \cos(\psi) & \sin(\theta) \sin(\psi) & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} \dot{\sigma}_X \\ \dot{\sigma}_Y \\ \dot{\sigma}_Z \end{bmatrix}$$

From this equation the angular velocities used in chapter 2.2 is found. Hence the transformation matrix will be updated for every loop iteration.

3.3 Dynamics Summary

By summing the forces the acceleration of the quadcopter in its body frame is found. By use of the transformation matrix those accelerations are converted into the body frame. And by integrating its acceleration in the navigation frame the position can be calculated.

The quadcopter attitude is determined by summing the moments. With different thrust on the motors the quadcopter will start to tilt or yaw.

4. Sensors and Components

In this chapter, the sensors and main components are described. The sensors are explained how they work and there is also told why different sensors and sensor types were chosen. The chosen sensors are also tested to be sure the measurements given are accurate. How the testing have been performed and its results will be shown in a separate subsection to the particular sensor.

4.1 Inertial Measurement Unit

The inertial measurement unit used onboard is the ADIS16400 from Analog Devices. The IMU is a complete inertial system that includes a triaxial gyroscope, a triaxial accelerometer, and a triaxial magnetometer. The IMU communicates with the SPI protocol.



Figure 4.1: *ADIS16400 IMU from Analog Device*

The gyroscope measures the angular velocity of the IMU. Gyros are significantly affected by noise but they drift [10]. The drift can, to a certain point, be corrected for. By integrating the angular velocity around one axis the angle can be found.

$$\sigma_X = \int \dot{\sigma}_X dt \quad (4.1)$$

The accelerometer can measure both static acceleration, like gravity, and dynamic accelerations. The drawback of the accelerometer is that it is affected by white Gaussian noise [24]. It is only capable of measuring a vector that has a component perpendicular to the earth, and cannot measure absolute rotation parallel to the earth. For rotation around one axis the angle can be found by:

$$\sigma_X = \arctan \frac{Acc_X}{\sqrt{Acc_Y^2 + Acc_Z^2}} \quad (4.2)$$

The magnetometer measures the earth's magnetic field. This field is perpendicular to the gravitational field of the earth. This makes it possible to measure the lacking orientation from the accelerometer. It is however not possible to measure angles parallel to the magnetic field. The magnetometer angles are calculated as:

$$\sigma_X = \arctan \frac{Magn_X}{\sqrt{Magn_Y^2 + Magn_Z^2}} \quad (4.3)$$

The sensors in the IMU are complementary since they all can measure the same angle, but with different methods.

4.1.1 IMU Test

In order to verify the dynamics of and the response of the IMU, it was mounted on a test bench in order to test it. The test bench is a part of an inverted pendulum, and consists of a shaft with an encoder. The encoder has a total of 3600 pulses per revolution. Used as a quadrature encoder the total pulses increases to 14400. Hence the resolution of the sensor is 0.436e-3 rad/pulse (0.025deg/pulse). The IMU was mounted on the test bench according to Figure 4.2.

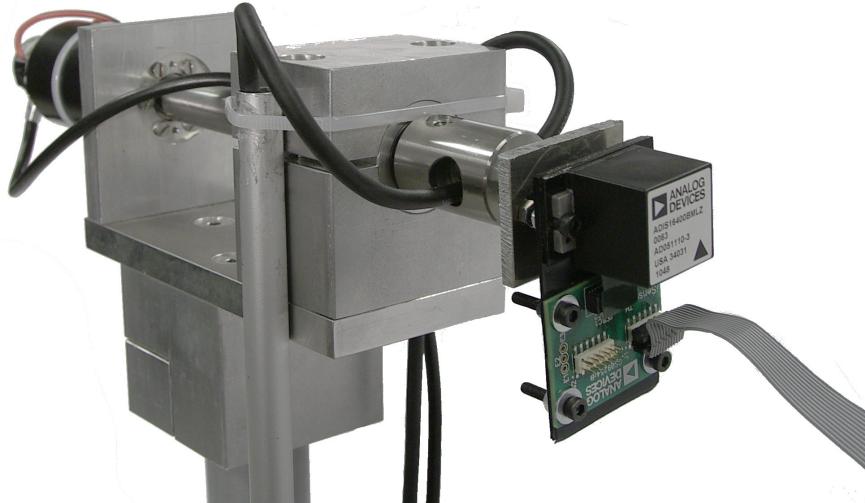


Figure 4.2: The test bench used as angle reference for the IMU

With this setup the real angle from the encoder and the angle from the inertial sensor could be measured and compared. The unit is factory calibrated but the gyroscope needed some fine adjustment in order to be good enough. Figure 4.3 shows the unfiltered static response for the three sensors.

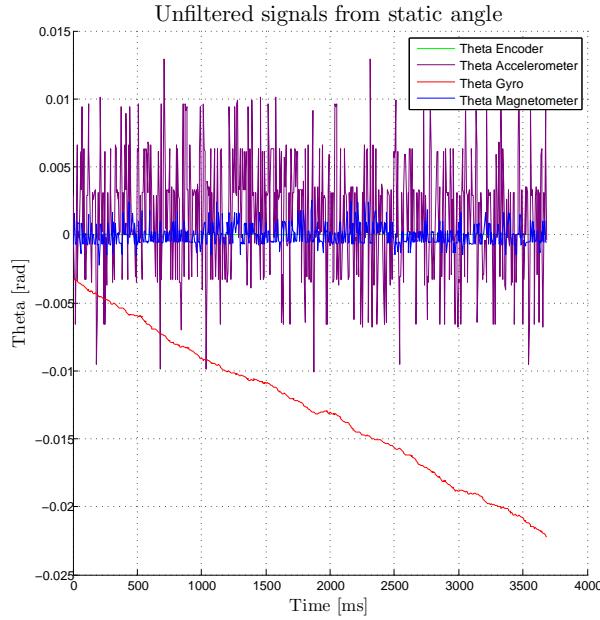


Figure 4.3: *Theta of the static response for the non calibrated IMU*

Figure 4.3 shows that the gyroscope is drifting when being integrated. The accelerometer has a lot more noise than the magnetometer, but both of them has white noise. The encoder is showing zero with no noise or drift. The gyroscope was calibrated and the result was a lot better (Figure 4.4). Over a longer time the gyroscope was drifting.

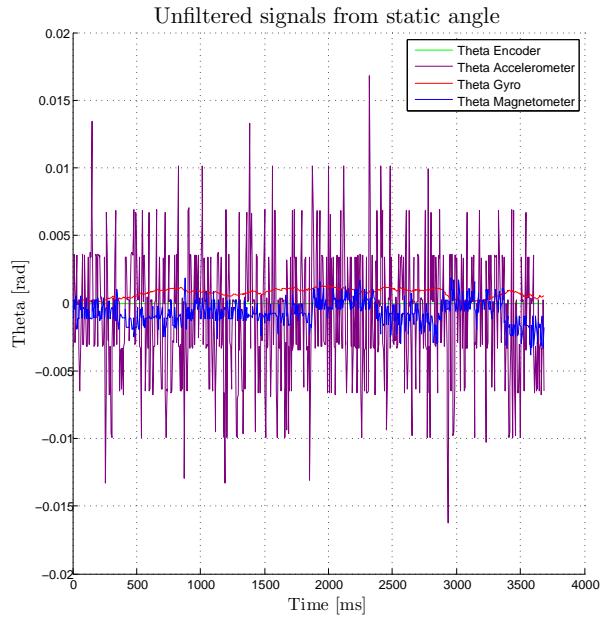


Figure 4.4: *Theta of the static response for the calibrated IMU*

The dynamic performance of the sensor was also tested.

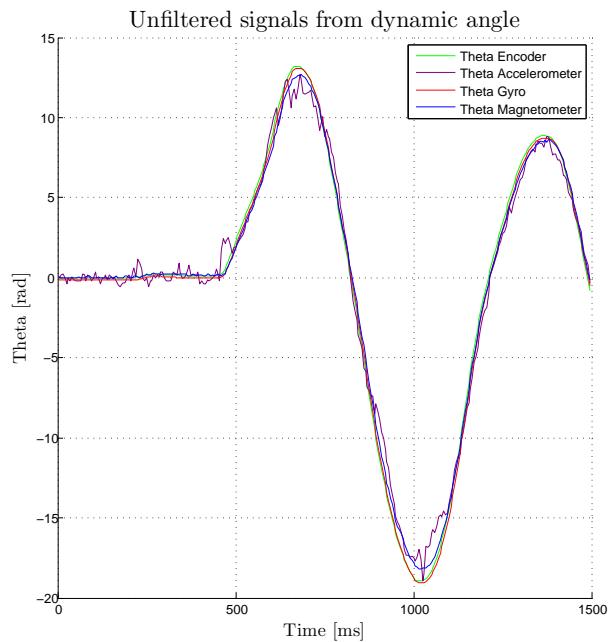


Figure 4.5: *Theta* of the dynamic response for the IMU

The gyroscope has fast dynamics and very little noise. The drawback is that it tends to drift off since it is being integrated. The accelerometer is however slower, it is not affected by drift but it has noise.

4.2 RPM

There are many methods to measure the rpm of a brushless DC motor. The first method tested was to measure the frequency of two of the motor phases. One advantage using this type of sensor is that there is few components required, and none of them are mechanical. And compared to an optical measurement it is unaffected by room lighting and no light source is required.

The phase sequence of the brushless DC motor is described below:

A	B	C
DC+	Off	DC-
DC+	DC-	Off
Off	DC-	DC+
DC-	Off	DC+
DC-	DC+	Off
Off	DC+	DC-

Table 4.1: Phase voltage sequence for a brushless DC motor [28]

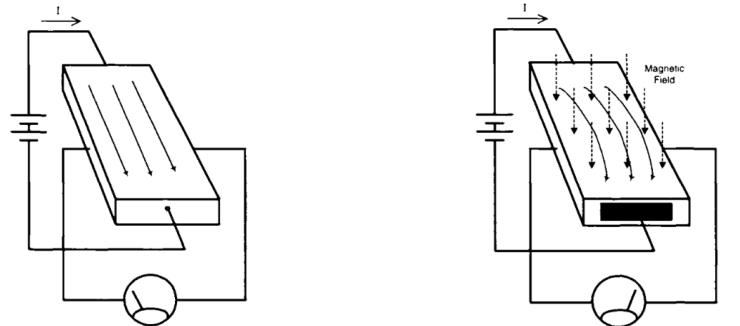
During the three first steps phase A will be higher than phase B and vice versa for the next three steps. If phase A and B are connected to a comparator the frequency of the phases can be measured and hence the rpm. A disadvantage is that some controller and motor combinations may produce a lot of commutation noise which can cause false rpm readings. And because of this this method were rejected.

The method chosen for measuring the rpm was by using hall effect sensors.



Figure 4.6: *Hall effect sensor*

The hall effect sensors work on the principle that when a current moves trough a conductive material the voltage across the material will be zero volt 4.7(a). If a magnetic field were to be applied to the material, a small voltage would appear 4.7(b) [21].



(a) Current passing trough zero flux field

(b) Current passing trough a flux field

Figure 4.7: *The principle of hall effect [21]*

The hall effect sensor has an internal trigger circuit and amplifier. It needs 5V in order to operate, and an external pull-up resistor.

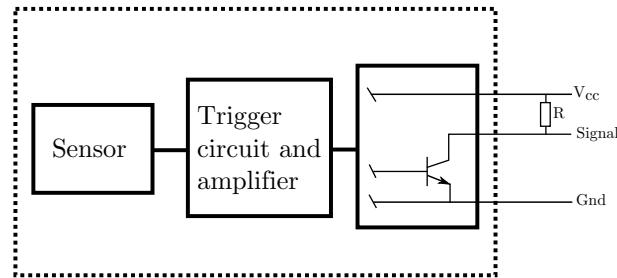


Figure 4.8: *Hall effect sensor circuit diagram*

The sensor used is a bipolar hall effect sensor. This sensor sets the signal high with one magnetic field direction, and resets it by a reverse direction. Since the motors have parma magnets fitted inside the outer rotating shell configured with north and south pole for every second magnet, the sensor can pick up the magnetic fields as they pass. With seven north and south pole magnets the bipolar hall effect sensor will give seven pulses pr rotation. With seven pulses pr rotation the rpm can easily be calculated.

The hall effect sensor is mounted very close to the motor in order to pick up the signal. The sensor also has to be mounted perpendicular to the motor to prevent it from sensing the magnets wrong.

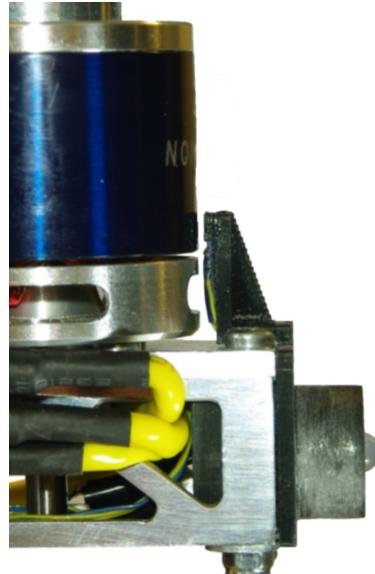


Figure 4.9: *Hall effect sensor mounted on the quadcopter*

4.3 Voltage and Current Measurement

The motors and the controller card is run by batteries and it is a must to know how much power that is remaining in the batteries at any given time. Therefore voltage and current sensors are used to monitor the usage and status of the batteries.



Figure 4.10: *Current and voltage sensor for the battery*

"AttoPilot Voltage and Current Sense Breakout" were chosen because of its small size and relative low cost. The sensor is soldered to the wire between the battery and the power consumer (motors or sbRIO board). The current is measured by measuring the voltage over a shunt resistor. The board has two analog output signals ranging from 0 - 3.3 V, one for the current and one for the voltage. The resolution is $63.69 \frac{mV}{Volt}$ and $36.60 \frac{mV}{Amp}$, and the maximum ratings are 51.8V and 89.4A.

4.4 Rangefinder

To verify the distance and make sure the quadcopter doesn't crash into objects during flight and landing the distance to the ground has to be measured. This distance could also be used do update and regulate the estimated position of the quadcopter [5.2](#).

The most commonly ways to measure distance is by either optical or super sonic sensors. The super sonic sensors was considered but because of the rugged surroundings the quadcopter could be flying in this was not considered to be a good solution.

A proximity sensor from Sharp was chosen, this sensor is sending out infrared light and detects it when reflected from a surface. From the reflected beam the distance to the object can be calculated. This sensor is very accurate and suits its purpose well. The model chosen is "GP2Y0A02YK0F" which have an measurement range from 20 to 150 cm. The sensor returns an analog signal to indicate the range measured.



Figure 4.11: *Sharp rangefinder*

4.5 ESC

The choice of the ESC was done in the preliminary report [\[14\]](#). The ESC chosen is a YGE 60. The ESC suits the motors well with its maximum current of 60 amperes. The ESC has, in contrast to most others commercial products, an option to receive signals up to 100 Hz.

High update frequency of the motors are desirable when it comes to the control system. The YGE ESC also offers lower response time due to a lower internal resistance than the normal ESC on the market [\[7\]](#). The YGE controllers are a good choice for this project.

4.6 XBee Transceiver

The XBee transceiver was chosen to serve as the wireless link between the basestation and the quadcopter. The XBee was chosen because of its simplicity and long range, but also because of knowledge of it.

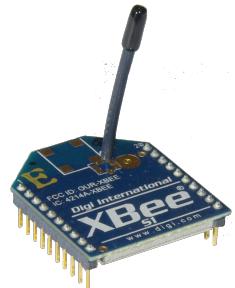


Figure 4.12: *XBee transceiver*

The XBee is an transceiver which can be configured to transmit and receive up to 250 kbps. During testing it was found that the base station could not handle high baudrates and therefore a baudrate of 19200 bps were chosen.

The XBee is sold in several different variations where strength and antenna configurations can be chosen. For this project an XBee set with 1 mW signal strength and a wire antenna were chosen. Its range is up to 100 meters outdoor and has an indicator for its received signal strength. This indicator is not yet implemented. The XBee can be replaced by another unit if the range is to be increased. The longest range available is 64 kilometers.

4.7 Control board

The board to run the data processing and control system had been chosen by the University to be an Single-Board RIO 9631 (sbRIO) from National Instrument. The card is programmed using LabVIEW with an ethernet cable. It consists of two main modules, one reconfigurable field-programmable gate array (FPGA) module which handles the I/O data at an integer level, and one real time processor operating at 266 MHz, both are programmable with LabVIEW

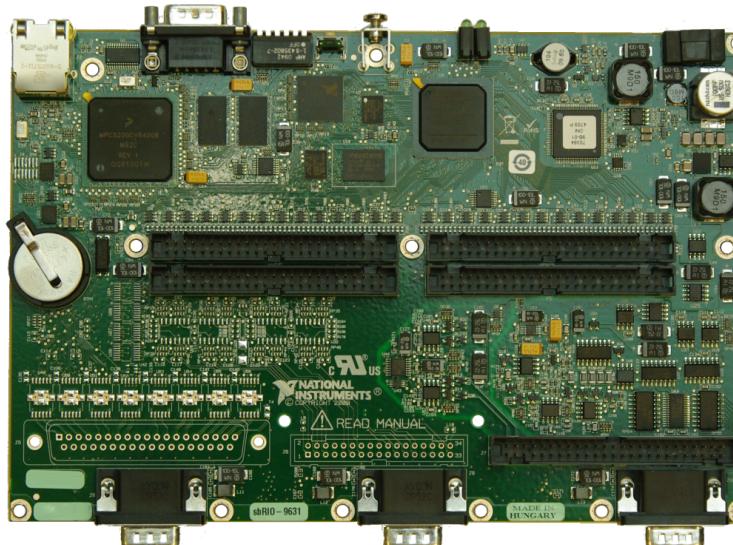


Figure 4.13: *NI sbRIO-9631 controller card*

The FPGA module is a stand alone module that operates without the need of a command from the RT-processor. An FPGA module is a microchip where the hardware is programmed to line up according to the program with the help of transistors. This makes programs run extremely fast and can run tasks parallel.[22]

4.8 Batteries

The batteries is a major component in this project. The reason for its importance is because it is the component that will decide how long the quadcopter can be airborne and it is also the most heavy components mounted. The choice was lithium polymer (lipo) batteries due to their good power to weight ratio [5] and their ability to deliver enough output power.



Figure 4.14: *Batteries used, to the right 5000 mAh and to the left 450 mAh*

To reduce the total weight two motors use the same battery. The motor battery is a three cell lipo from Flighthpower. These batteries yields 11.1 volts and 5000 mAh to drive the motors. It can continuously feed 150 amperes and peak up to 300 amperes.

The sbRIO also need power and so do the sensors. The controller board requires 19-30 volts, which is above what a regular three cell lipo battery is capable of. To achieve the high volt two batteries were coupled in series. The batteries are made by Hyperion and yields 11.1 volts and 450 mAh (figure 4.14).

4.9 Propeller

The propeller was chosen in the pre-study [14] for this thesis. It was concluded that the choice should be APC 2x6 propellers. This is a nylon propeller, 12 inches long and with a pitch of 6 inches. The pitch tells that in one rotation the propeller would move 6 inches in translative direction [19].



Figure 4.15: *APC 12x6 propeller*

The propeller was chosen by the fact that they could easily be bought as both regular and pusher version. Pusher propellers are mirrored versions that is optimized to rotate in clockwise rotation instead of counter clockwise. They also put out enough power to implement other sensors or equipment on a later stage.

4.10 Motor

The motors selected for the quadcopter were also decided in the pre-study [14]. The motors chosen are Hyperion ZS 3020-10 (Figure 4.16). This motor is a BLDC motor made for model aircrafts. The reason for the choice of this motor is its easy to get an hold of and its fits well with the propellers chosen.

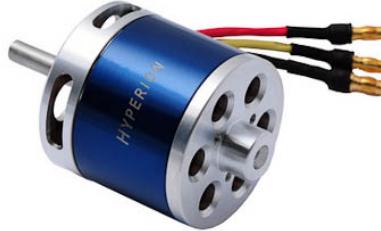


Figure 4.16: *Hyperion ZS3020-10*

The motor is very powerful with 456 watts and a low kv rate of $922 \frac{rpm}{volt}$. The low kv is desirable as this means it has less rotational speed and higher torque.

4.11 Motor and Propeller testing

A motor test was performed in order to complete the dynamic model. The main part of the test was to determine the moment each motor generates with the APC 12x6 propeller at a certain rpm as this data was not provided by the manufacturer.

4.11.1 Theoretical Performance

A brushless DC-motor (BLDC) works on the same principals as a brushed DC-motor. The difference is that the commutator for the BLDC is integrated in the speed controller, while for a DC-motor it is a part of the motor. Hence the performance equations are the same for both of them.

K_v is the relationship between the rpm and the voltage and is given for most of the BLDC's. For the motor chosen in this project:

$$K_v = 922 \left[\frac{rpm}{volt} \right]$$

By using the electrical-mechanical relationship, the torque constant K_t can be found.

$$\text{Electrical power} \cdot \text{efficiency} = \text{Mechanical power}$$

$$E \cdot I \cdot \sqrt{3} = N \cdot \frac{2\pi}{60} \cdot T$$

$$\frac{E}{N} = 0.0604599788 \cdot \frac{T}{I}$$

where

E = motor voltage rms [V]

I = motor current [A]

N = angular velocity of the motor [rpm]

T = motor torque [Nm]

$$\frac{E}{N} = \frac{1}{K_v}$$
$$\frac{T}{I} = K_t$$

thus a constant relationship between K_v and K_t is obtained

$$K_t = \frac{1}{0.06046} \cdot \frac{1}{K_v}$$
$$K_t = 0.01794$$

And by this the torque can be calculated

$$T = I \cdot K_t \quad (4.4)$$

4.11.2 Test bench

The Hyperion ZS3020-10 motor with backmount was fixed to a bracket with machine screws. The motor torque was transferred from the bracket to the torque transducer by bolts and a socket for a socket wrench. A pusher propeller was used in order to push the whole assembly together. The torque transducer was fixed by pressing it onto the board by clamps.



Figure 4.17: Torque transducer 0-10Nm from AEP



Figure 4.18: *Test setup*

The torque transducer generates an analog signal, where a change of 1mV/V corresponds to 10Nm. With an excitation voltage of 5 volt the signal will change by 5mv in order to go from 0 to 10Nm. This was read by an NI cDAQ-9172 and the NI 9237 analog input card from National Instruments.

A mean value of 50.000 samples were calculated for the current and the torque measurement before running it through a smoothing filter in LabVIEW. The rpm was only filtered by the smoothing filter.

The test was run three times, where the first test is neglected because of lack of data. The second and third run gave equal results.

Test2:

RPM	Torque [Nm]	Current [A]	K_t
8437	0.304	28.329	0.01073
6385	0.166	12.300	0.01350
4568	0.082	5.125	0.01592
3632	0.052	3.109	0.01686
2078	0.018	1.201	0.01458

Table 4.2: Motor torque results, test 2

Test3:

RPM	Torque [Nm]	Current [A]	K_t
8651	0.326	30.159	0.0108
7332	0.226	17.848	0.0126
6295	0.164	11.592	0.0141
4621	0.087	5.277	0.0165
1774	0.015	1.157	0.0131

Table 4.3: Motor torque results, test 3

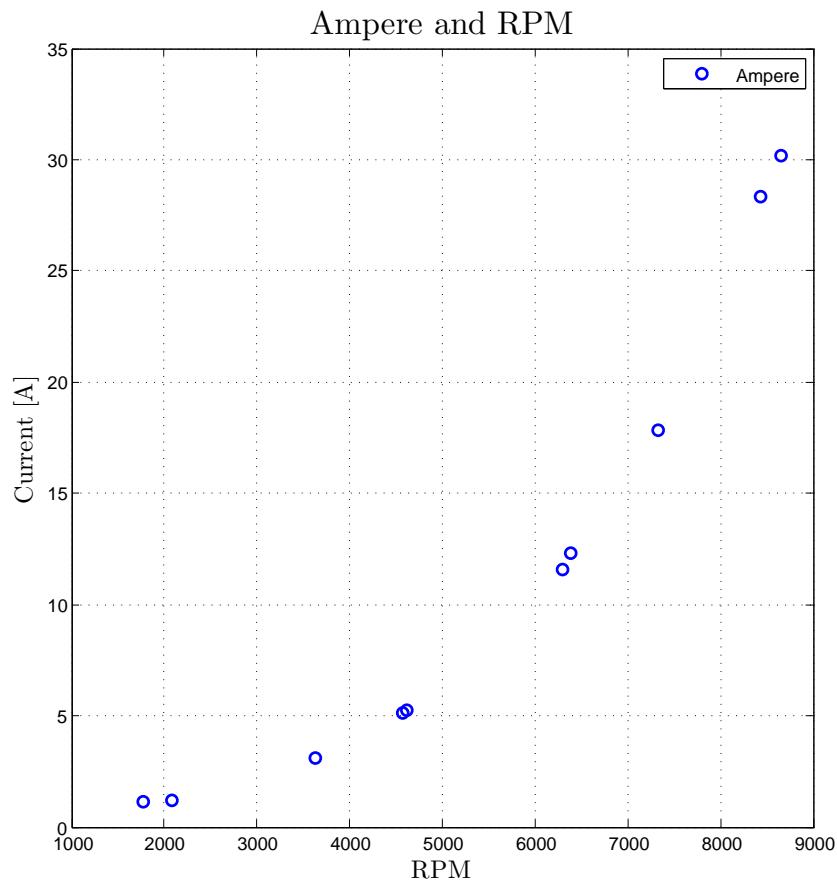


Figure 4.19: *Test results of current vs RPM*

K_t is according to the tests not a constant since it varies with the rpm. But what's more important, is to get an equation for the torque with the rpm as input.

4.11.3 Force vs Torque

The force and torque relationship is important for the yaw controller. With a linear relationship the torque can be adjusted independent of the force. The measured force and torque data were plotted using Matlab.

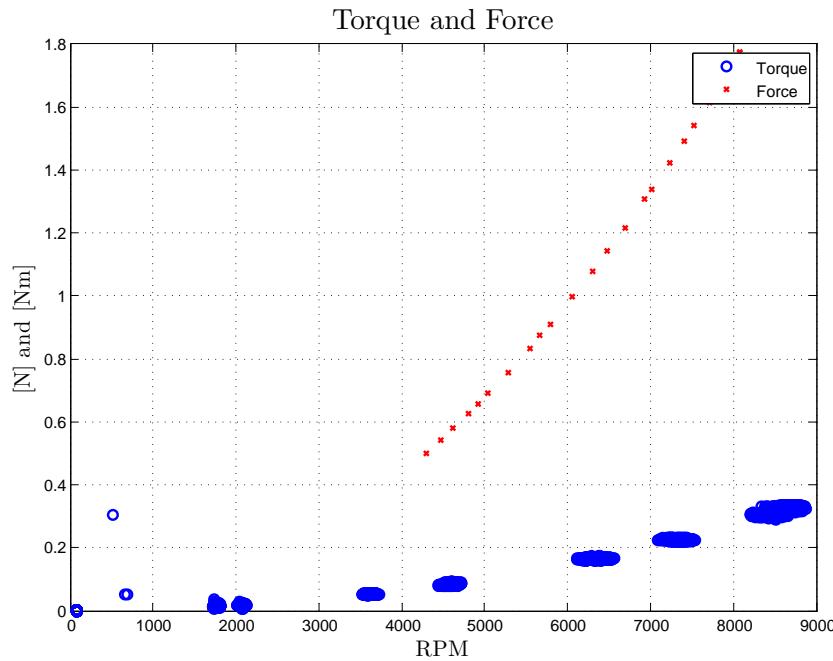


Figure 4.20: *Test results of Force and Torque*

By the Matlab curve fitting tool equations for the two functions in figure 4.20 were found. The propeller thrust is calculated using the same equation as for the torque, and the thrust is calculated as a quadratic function of the velocity [2].

$$Thrust = \frac{1}{2} \cdot \rho \cdot v^2 \cdot Sw \cdot C_L \quad (4.5)$$

Because of this a quadratic function was selected for the thrust and torque.

$$Torque = 4.6055 \cdot 10^{-9} \cdot x^2 - 3.2534 \cdot 10^{-6} \cdot x + 0.0032171 \quad (4.6)$$

$$Force = 2.7132 \cdot 10^{-8} \cdot x^2 + 7.7106 \cdot 10^{-7} \cdot x - 0.0026378 \quad (4.7)$$

By multiplying the force by a trial and error factor of $\frac{1}{6.3}$ a linear relationship was found.

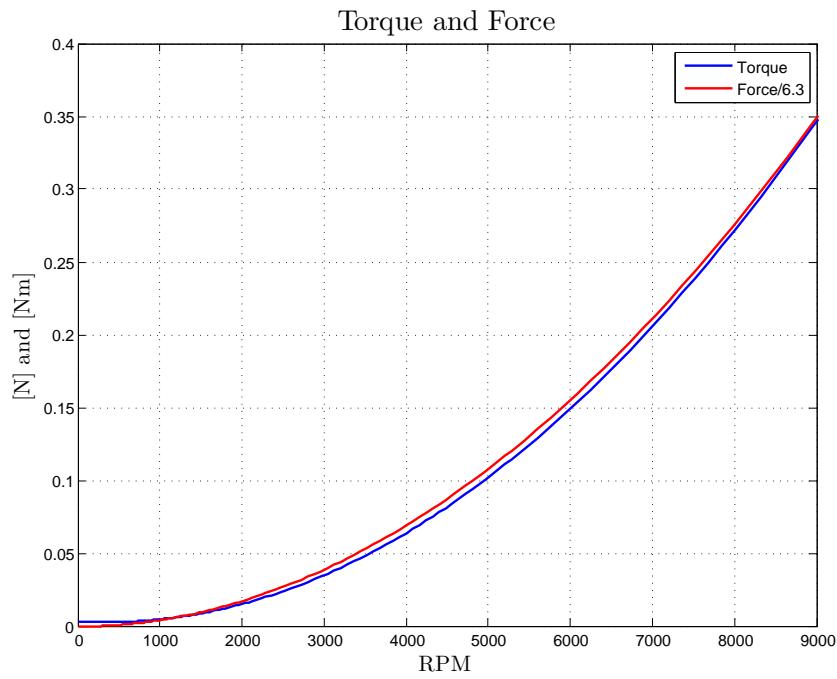


Figure 4.21: *Linear relationship of the Force and Torque*

Hence the force and the torque has a linear relationship where

$$Torque = \frac{1}{6.3} \cdot Force \quad (4.8)$$

4.11.4 ESC

The ESC is operating at 100Hz with a standard RC-signal which is a high pulse for 1-2 milliseconds. The ESC has an internal regulator that controls the speed of the motor. If there is a linear relationship between the RC-signal sent to the ESC and the real rpm of the motor the onboard control system for the angles can convert its rpm setpoint directly into an RC-signal. This relationship was tested by applying a constant RC-signal for the ESC and measure the rpm of the motor.

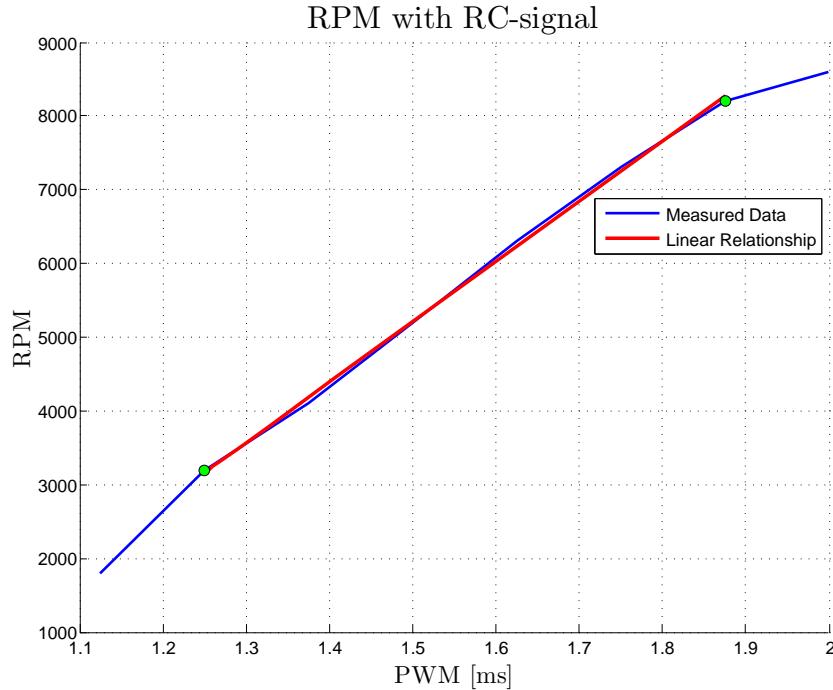


Figure 4.22: *RPM at a given PWM*

The data is pretty linear between the two green data points. Out side of them the curve is not that linear, but this region is far from the rpm that requires hoovering. Hence a linear relationship was found based on the data between the two green data points, equation 4.9.

$$RPM = 8160 \cdot PWM - 7033.3 \quad (4.9)$$

5. Attitude and Position Estimation

5.1 Attitude Estimation

The IMU is strapped down to the quadcopter and measuring acceleration and angular velocity for the body coordinate system on the quadcopter. For a single axis system the attitude can easily be determined by for instance a first order complimentary filter where the gyro is being integrated as described in chapter 4.1. But for a multi axis system the attitude cannot be determined that easily because the attitude is dependant of the sequence of the rotation.

Imagine the following scenario. An aeroplane is flying a straight line (fig. 5.1(a)) and only using a three axis gyroscope to measure its attitude. Then the aeroplane pitches up +90 degrees (fig. 5.1(b)). At this point the integration of the gyros gives +90 degrees pitch. Then the aeroplane rolls +90 degrees (fig. 5.1(c)). The two gyros will now show +90 degrees pitch, and roll. To decrees the +90 degrees pitch the plane has to do a -90 degrees pitch (fig. 5.1(d)). The measured angles will now be 0 degrees pitch and +90 degrees roll. Then the plane rolls -90 degrees (fig. 5.1(e)). At this point both the gyros shows an angle of 0 degrees pitch and roll, but the attitude of the aeroplane is not the same as the initial position.

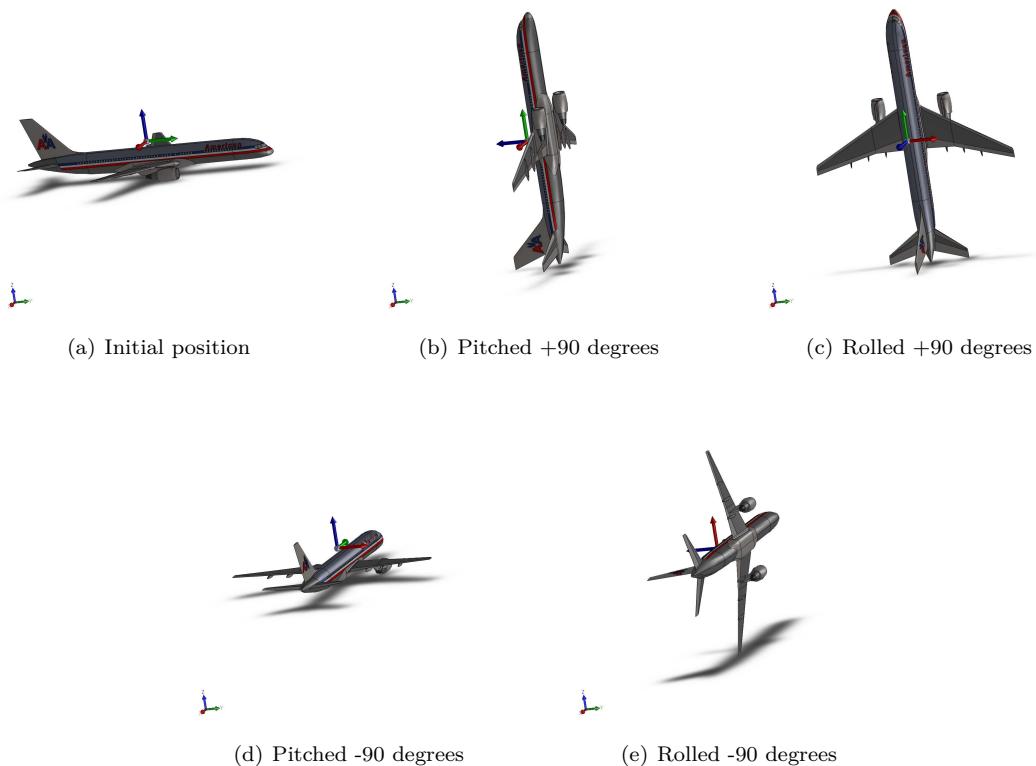


Figure 5.1: Rotating locally with the sum of 0 degrees, model in the pictures is downloaded from "<http://www.topfreemodel.com/SolidWorks-Models/Transport/Boeing-757-aircraft-SolidWorks-Models.html>"

A gyroless attitude estimation has previous been tested, but with too high error margin for a quadcopter [1]. The Kalman approach is also tried but is more difficult to implement and does not result in better estimation [13].

The attitude estimator chosen is a complementary filter [16]. This is the most common filter used on UAV's. It provide a robust tradeoff between a good short term precision from the gyroscopes and a long term accuracy provided by accelerometers [4]. This complementary filter is also proved asymptotically stable [26].

The estimation of the DCM and the Euler angles from is divided into four sections [20]. An overview of the process is shown in figure 5.2.

1. Direction cosine from gyro signals
2. Renormalization of the DCM
3. Drift correction
4. Calculating the Euler angles

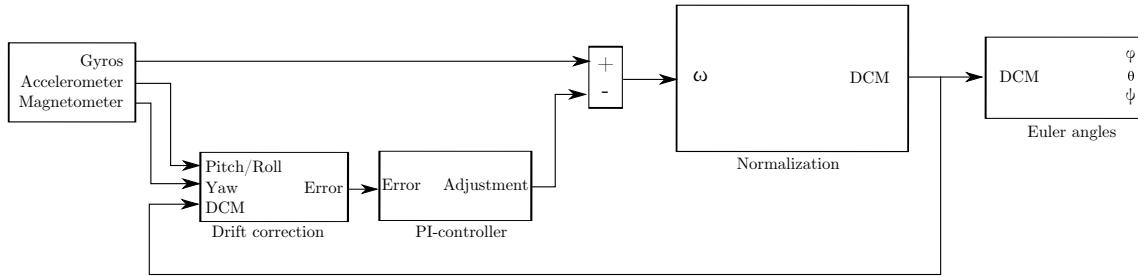


Figure 5.2: Block diagram of the attitude estimation

5.1.1 DCM from gyro signals

The time derivative of the DCM can be expressed by [3]:

$$\dot{\underline{C}}_b^n = \underline{C}_b^n \cdot \underline{\Omega} \quad (5.1)$$

where Ω is a skew symmetric matrix of the angular rates

$$\underline{\Omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \quad (5.2)$$

The new updated DCM is found by euler integration of equation 5.1

$$\underline{C}_b^n = \underline{C}_b^n (\underline{I} + \underline{\Omega} \cdot \Delta t) \quad (5.3)$$

5.1.2 Re-normalize the DCM

The DCM is, or should be, an orthogonal matrix, that is all three vectors are perpendicular. The error can be calculated as the dot-product of the X-column and Y-column of the DCM. Ideally this dot-product will be equal zero, that is if the vectors are orthogonal [25].

$$E_{norm} = \begin{bmatrix} r_{xx} \\ r_{yx} \\ r_{zx} \end{bmatrix} \cdot \begin{bmatrix} r_{xy} \\ r_{yy} \\ r_{zy} \end{bmatrix}$$

$$E_{norm} = \begin{bmatrix} r_{xx} & r_{yx} & r_{zx} \end{bmatrix} \begin{bmatrix} r_{xy} \\ r_{yy} \\ r_{zy} \end{bmatrix}$$

The error is the total error for the two vectors. The contribution for each vector is therefor closely to half of it. It is therefor divided by two before added to the X-column and Y-column. *Apportioning the error equally to each vector yields a lower residual error after the correction than if the error were assigned entirely to one of the vectors.*

$$\underline{X}_{orthogonal} = \underline{X}_b - \frac{E_{norm}}{2} \cdot \underline{Y}_b$$

$$\underline{Y}_{orthogonal} = \underline{Y}_b - \frac{E_{norm}}{2} \cdot \underline{X}_b$$

The Z-column is also to be an orthogonal vector to the X- and Y-vector. This is calculated as the cross product of the normalized X-vector and Y-vector.

$$\underline{Z}_{orthogonal} = \underline{X}_{orthogonal} \times \underline{Y}_{orthogonal}$$

The rows and columns are unit vectors, hence the magnitude should be equal to one. This is corrected by dividing each element in each row of the body vectors by the magnitude of the vector. Since the magnitude never will be significant different than one, the Taylor series can be used as simplification.

$$\underline{X}_{normalized} = \frac{1}{2}(3 - \underline{X}_{orthogonal} \cdot \underline{X}_{orthogonal}) \cdot \underline{X}_{orthogonal}$$

$$\underline{Y}_{normalized} = \frac{1}{2}(3 - \underline{Y}_{orthogonal} \cdot \underline{Y}_{orthogonal}) \cdot \underline{Y}_{orthogonal}$$

$$\underline{Z}_{normalized} = \frac{1}{2}(3 - \underline{Z}_{orthogonal} \cdot \underline{Z}_{orthogonal}) \cdot \underline{Z}_{orthogonal}$$

The DCM can now be updated with the new column vectors

$$\underline{\underline{DCM}}_{normalized} = [\underline{X}_{normalized} \mid \underline{Y}_{normalized} \mid \underline{Z}_{normalized}]$$

5.1.3 Drift Correction

The update of the direction cosine matrix is done by integrating the gyros. In chapter 4.1 it was found that the gyros are not significant affected by noise, but they have an offset. Due to integration of the gyro signals the offset will result in drifting errors. The accelerometers are not affected by drift and are used to correct the integration of the gyros.

For two vectors that are parallel the cross product is equal zero [6]. This is used to correct the drift of the gyros and hence the DCM. The accelerometer vector is a measurement of the attitude of the quadcopter. Since it is only affected by the gravitational force which is pointing in the Z-direction of reference frame the cross product of those two vectors is to be zero when they are parallel. The length of the accelerometer vector is not taken into consideration yet. Equation 2.4 shows that the Z vector of the reference frame is equal to $\underline{C}_b^n(3,:)$

Centrifugal acceleration is not taken into consideration since the acceleration will be close to zero for a quadcopter. For a aeroplane it will be significant, and maybe for a quadcopter doing aerobatics. But for slow motion and hovering it is considered as a small disturbance.

The drift error is calculated as a vector of the cross product of the estimated Z-direction in the earth reference frame, and the true Z-direction measured by the accelerometers.

$$\begin{aligned} \underline{E}_{drift} &= \underline{C}_b^n(:,3) \times \underline{Acc} \\ \underline{E}_{drift} &= \begin{bmatrix} r_{xz} \\ r_{yz} \\ r_{zz} \end{bmatrix} \times \begin{bmatrix} Acc_x \\ Acc_y \\ Acc_z \end{bmatrix} \\ \underline{E}_{drift} &= \begin{bmatrix} Acc_z \cdot R_y - Acc_y \cdot R_z \\ Acc_x \cdot R_z - Acc_z \cdot R_x \\ Acc_y \cdot R_x - Acc_x \cdot R_y \end{bmatrix} \end{aligned} \quad (5.4)$$

Since the two vector lengths used to calculate the drift are constant the error is a vector whose length describes the drift. The drift is corrected by a PI-controller and subtracted from the rotation rate vector $\underline{\omega}$.

$$\begin{aligned} \underline{\omega}_P &= \underline{E}_{drift} \cdot K_p \\ \underline{\omega}_I &= \underline{\omega}_I + \underline{E}_{drift} \cdot K_i \end{aligned}$$

The accelerometers are not scaled so the proportional gain K_p has to be adjusted to suit the raw value of the accelerometer. The same is for the integral gain K_i .

Even though the drift will be corrected, the offset should be reduced as much as possible by a calibration. Because with a high gyro offset the PI-controller has to have a higher gain to correct for the drift. With a higher PI-gain the system is affected more by accelerometer noise.

$$\underline{\omega} = \underline{\omega}_{gyros} - \underline{\omega}_P - \underline{\omega}_I$$

The new $\underline{\omega}$ used to update the DCM (equ. 5.2) is now corrected for pitch and roll drift.

5.1.4 Euler angles

The Euler angles can be calculated from the terms of the DCM. By investigating the terms of the rotation matrix (equ. 2.2) it can be shown that ϕ can be extracted by the term $DCM(3, 2)$ and $DCM(3, 3)$.

$$\begin{aligned}\frac{DCM(3, 2)}{DCM(3, 3)} &= \frac{\sin(\phi) \cdot \cos(\theta)}{\cos(\phi) \cdot \cos(\theta)} \\ &= \frac{\sin(\phi)}{\cos(\phi)} \\ &= \tan(\phi)\end{aligned}$$

Using arcus tangent resolves the angle ϕ . The same method is used to calculate θ and ψ .

$$\begin{aligned}\phi &= \arctan 2 \left(\frac{DCM(3, 2)}{DCM(3, 3)} \right) \\ \theta &= -\sin(DCM(3, 1)) \\ \psi &= \arctan 2 \left(\frac{DCM(2, 1)}{DCM(1, 1)} \right)\end{aligned}$$

θ is the pitch of the quadcopter. Since it is extracted from a sine term, it will only be ± 90 degrees. The roll and yaw is extracted from an arctan 2 function which gives an angle of ± 180 degrees. This is just as expected. An aeroplane, and a quadcopter is never flying $+120$ degrees. It can pitch 120 degrees, but that is the same as pitching 60 degrees with a 180 degrees roll and yaw. So every position of the quadcopter is determined by the three Euler angles.

5.1.5 IMU testing

The attitude estimator was tested and compared to an encoder in the same test bench as described in chapter 4.2. The first test was a static drift correction test where the sensor starts at 0 radians, but the real angle is just above -0.12 radians. The low gain from the accelerometer makes the drift correction slow, but with low noise.

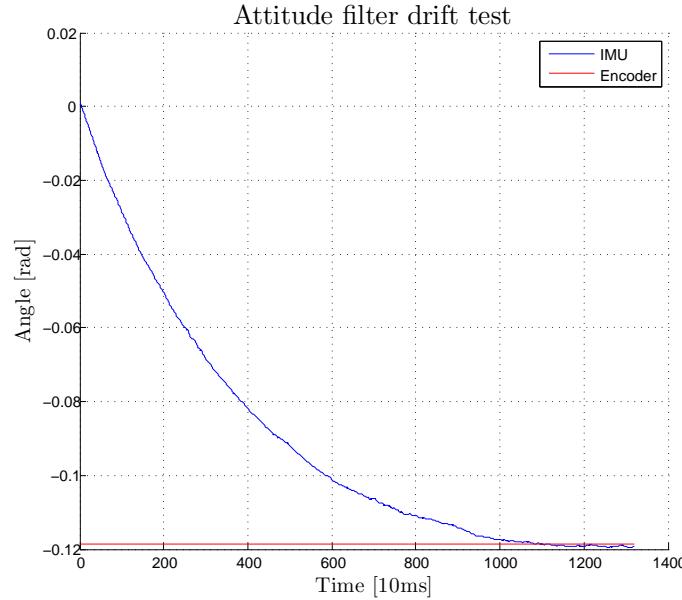


Figure 5.3: *Drift correction test*

The other test was a dynamic test. In this test the IMU was rotated with the encoder. The small angle error between the encoder and the IMU is most likely due to misalignment of the test bench.

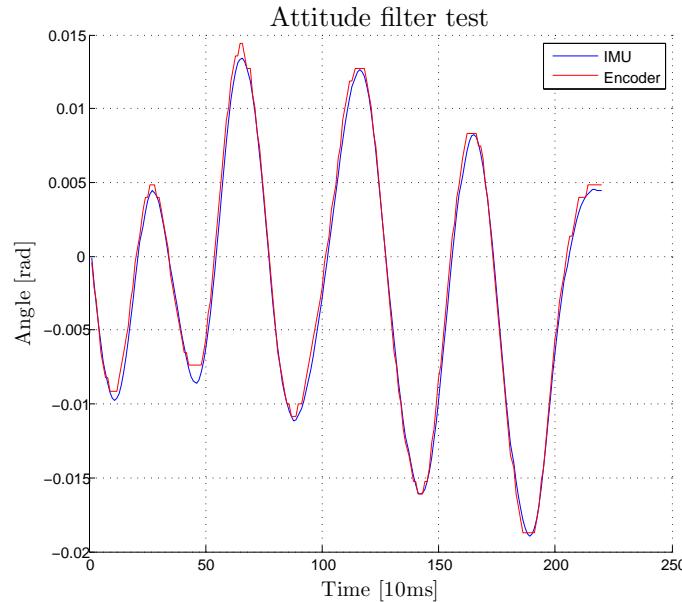


Figure 5.4: *Dynamic response*

5.2 Position Estimation

An important feature for an autonomous robot is its ability to know where it is at any time. There are several methods to solve position calculation, ranging from triangulation from known positions in the area to internal navigation system that calculate an approximated position with the help from onboard sensors.

5.2.1 Global Positioning System

GPS is a good method to use while moving outdoors. The information is gathered from satellites and calculated to give the current position of the receiver. Information of position, heading and velocity can easily be calculated and used to verify the robots current position.

The quadcopter has been fitted with a GPS module. As the plan for this project at this stage is to fly indoor the GPS was merely mounted to the quadcopter for later use. Even though the GPS is not used in the control system, a program was made to read the information from the GPS to make sure the module was working properly.

5.2.2 Internal Navigation System

INS is used by many autonomous vehicles. This system bases on the information gathered from sensors without external references. A method to calculate the position is with the help of accelerometers and gyros. The accelerometer used in this project has a limit of ± 18 g. This makes the INS inaccurate due to its low resolution.

A problem with an INS is the fact that it will become inaccurate over time and it would need some sort of sensors to update the position. However the quadcopter is to fly indoor at this stage hence disturbances like wind will not induce an angle on the quadcopter translative movement.

The thought for this project was to use the dynamic model from chapter 3 together with the attitude of the quadcopter to estimate the position. Unfortunately the constrained amount of time, did not allow this subject to be further explored and the INS were unfortunately not implemented to the quadcopter.

5.2.3 Visual Recognition

A valid method to find an objects position is by using a camera and reference points or sensors to triangulate the position. Depending on what references used and the area operating in, this method can be a robust method to verify the position. In addition to verifying the position, a camera could be used to detect objects to avoid.

A camera with a laser mounted, as in figure 5.5, could also be used as an range sensor.

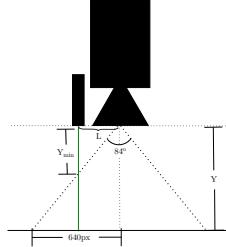


Figure 5.5: How to measure distance with the camera

The distance to the point where the laser reflects will have a constant relation to the number of pixels the point is from the center of the picture. Unfortunately the constrained amount of time, did not allow this subject to be further explored and the camera was unfortunately not implemented to the quadcopter.

5.3 Summary

The attitude estimation is working as expected. The roll is ranging between ± 180 degrees, pitch is ± 90 degrees while the yaw is 0-360 degrees. A pitch of more than 90 degrees does not make sense for a flying aircraft. If the quadcopter for instance pitches 120 degrees this is the same as if the quadcopter rolls 180 degrees, yaws 180 degrees with a pitch of 60 degrees. This is automatically taken care of by the estimator. The estimator is in other words capable of orienting the quadcopter in any angle.

In order to get the best out of the filter a gyro calibration is needed. With little drift on the gyros accelerometer gains can be reduced. With a small gain on the accelerometer less noise is added. But if the gain is too low, the estimator will not be able to correct for drift.

Testing shows that the estimator is fast and can track the angle very accurately. The drifting rate gyros takes care of the short term accuracy with almost no noise, while the noisy accelerometers takes care of the long term accuracy and corrects the drift.

6. Construction

The construction is divided in two main sections, which are the mechanical and electrical construction. It should be noted in the mechanical section, there is produced a design of the quadcopter (chapter 6.1.1), this design were on a later stage redesigned due to practical reasons which is explained in the same chapter (6.1.2)

The final design of the quadcopter can be found in chapter 6.1.2.

An general mechanical drawings for the final design for the quadcopter, can be found in appendix B.

6.1 Mechanical

One advantage for a quadcopter is the possibility to make the cross-section close to identically for the x-z plane and y-z plane. By making the cross-sections as identical as possible the dynamic modeling and programming of the control system will be easier.

In the preliminary report [14] a prototype was designed. This designed was manufactured and evaluated. There were made 3D CAD models of all the parts in SolidWorks. This allowed changes to be examined before the quadcopter was manufactured. Another feature of having a 3D model is that inertia matrix (chapter 3) can be calculated automatically.

6.1.1 Prototype design

The first design of the quadcopter was designed to be as simple as possible, and was created in the preliminary project [14]. The main frame was made from two aluminium bars with cross section dimensions of 20x20 millimeters. Aluminium was chosen because of its high strength and low weight, it is also relatively cheap and easy to get access to.

In the center of the frame it was decided to make a plate where the control board and sensors could be fitted. This plate would be made from ABS plastic with the help of the university's 3D printer. Using the printer allowed for an accurate placement of fixture points and slots for the beams, thus making the plate to sit stable upon the main frame.

The motors had to be secured with screws from underneath. The motors came with original brackets intended for fastening the motors but the design of these brackets, would not suit the design for of the quadcopter. Therefore new brackets were designed to fit the slimmer fastening area the aluminium beams would provide.

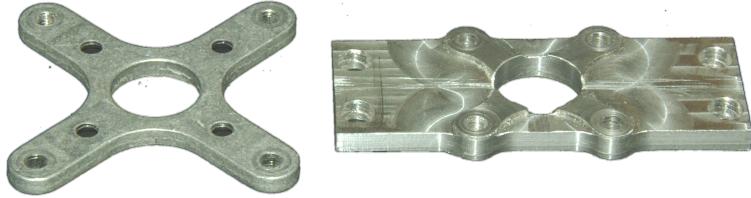


Figure 6.1: To the left, the first motor bracket while to the right the revised one

The brackets were milled out from aluminium by the university's milling machine, and the holes for mounting the motors onto the brackets were drilled afterwards.

6.1.2 Revised Design

The quadcopter had a working frame and design but had several disadvantages. The two largest drawbacks were that the IMU sensor was not placed in the center of gravity. It was several centimeters from the center of gravity and would give dynamical acceleration because of its translative motions due to the placement, and not just sense the gravity [8]. This effect could be taken care of in the programming, if the quadcopter was not to be redesigned.

Another big drawback was the distance between the control board and the propeller. To get the IMU away from the magnetic fields of the battery it was placed between the control board and the aluminium beams. The clearance between the control board and the propeller was too low and the propellers could damage the control board during a crash or a hard landing.

When designing the second version of the quadcopter, the first step was to place the IMU sensor. The sensor was to be placed in the center of gravity of the quadcopter, that is close to the center of the whole assembly. Then the four motors, propellers and motor brackets were placed around the IMU along the X- and Y-axis. Temporary arms to hold the motors were also placed.

Step two was to place the controller board. There was no requirements on where this had to be placed, other than its physical size. It was chosen to place the controller board above the IMU to give easy access to the connections. The batteries were also placed, these were placed below the IMU in an attempt to cancel out the weight above the IMU.

The rest of components were added to the assembly. The placement of the components gave the new layout for the top plate design. The top plate was designed and added to the assembly where the center of mass was checked and confirmed to be at the center of the IMU sensor.

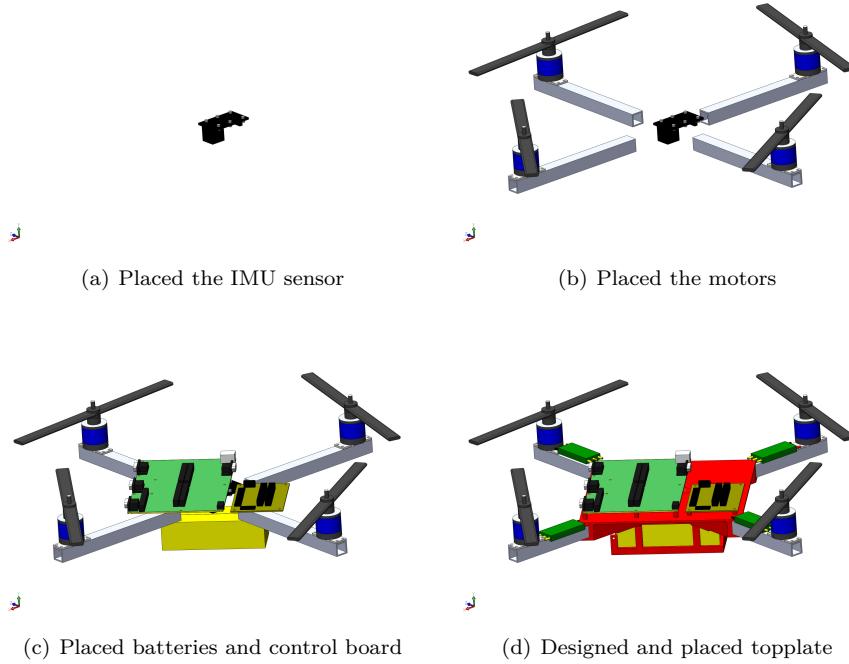


Figure 6.2: *Design progress*

The build of the second version took far less time than the first. The reason for this is because the amount of thought put into the design. The top plate was printed with the 3D plastic printer as the first version, and it was designed with fastening holes and knobs to secure components. The motor brackets that were made for the first version were reused for the second version. The aluminium arms were cut into correct length and there were drilled holes for the motor brackets. The aluminium arms were further improved by designing cuts to halve the weight.



Figure 6.3: *Arm with out cuts*

To fasten the control board onto the top plate screws made of nylon was used. This could save the control board from breaking incase of a crash, as the screws would snap before the board.

As the beams would no longer make up the cross as the earlier version did, all of the forces will be transferred from the beams into the top plate. Therefore a new strength analysis was needed to be sure that the plastic plate would handle the stress. It should be noted that the finite element method analysis was done to give an idea about the level of forces that would affect the module.

The analysis was done with a force of 30 N from each motor and the weight from the batteries were also implemented. The model was constrained at an area at the middle of the top of the top plate, this to be sure that the top plate would get the most stress.

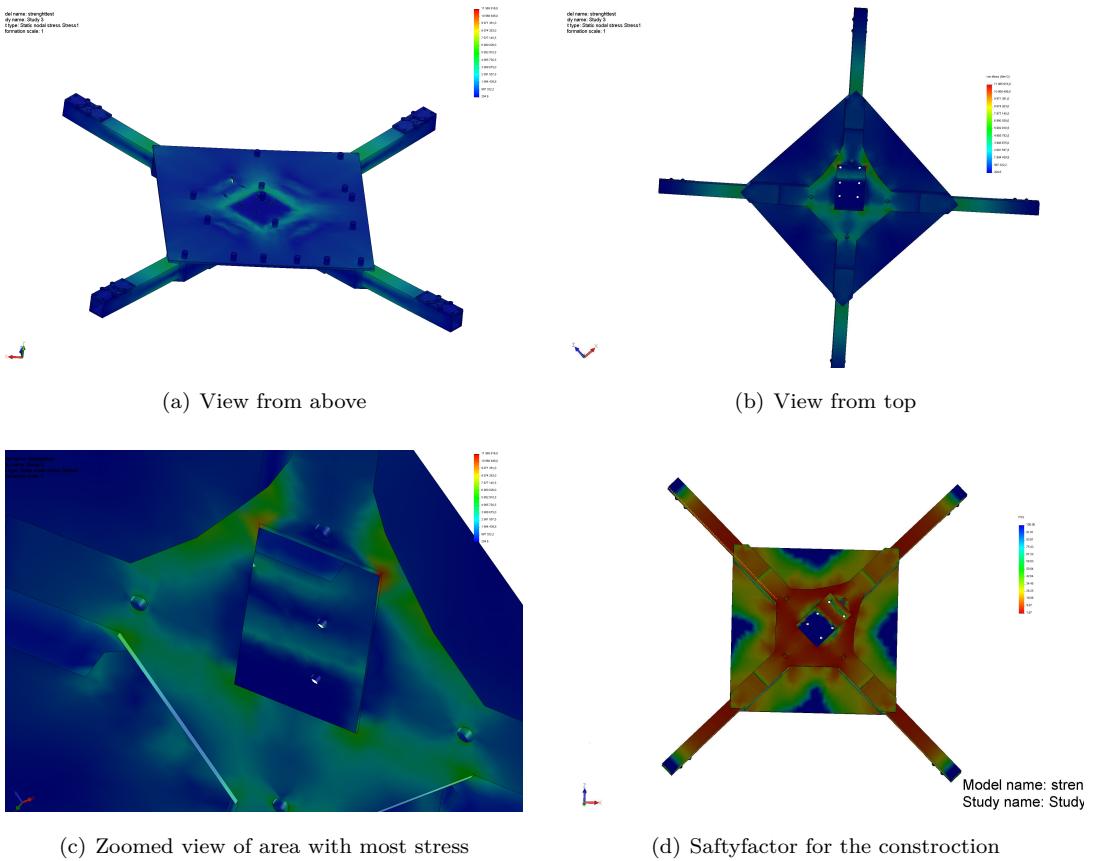


Figure 6.4: *Fem Analysis*

The plastic have been tested in an earlier project and proved to have an yield strength of $19,2 \text{ N/mm}^2$ with an elasticity modulus of $1,3 \text{ kN/mm}^2$. The test showed also that the elasticity for the plastic were close to linear up to the yield point [27]. The maximum force registered from the analysis was $11,97 \text{ N/mm}^2$ which is well below the limit. In Figure 6.4(d) the structure's level of safety factor is shown. The lowest safety factor is 1,67.

There were designed end knobs to be fitted in the end of the aluminium arms (Figure 4.8). The knob were designed with an cylinder in the end, this cylinder were designed to fit perfectly in the test rig (chapter 6.2). There were also made a holder for the RPM sensor and a slot to cover the cables from the sensor. In the middle of the cylinder at the end there is mounted an RGB led which were planned to use as navigation lights. These parts were made from plastic in the 3d printer at the University.



Figure 6.5: *Final design of quadcopter*

Figure 6.5 shows the quadcopter complete with all sensors fitted.

6.2 Test Rig

With the design of the quadcopter as it was build, it could do some serious damage both to humans and to the surroundings if control is lost. Therefore a safe environment to test the quadcopter was needed. A test rig was built which allowed the quadcopter to rotate around one axis while the other two axes were confined. Also the three translative motions were fixed to prevent any serious accidents to occur.



Figure 6.6: *The test rig*

The rig also have an option to allow one translative motion as well as one rotation. This is done by dismount and rotate the vertical arms. The arms have on this side a slot 20 cm long and thereby allowing the quadcopter to move up and down.

The rig is made up of 20x20mm aluminium square beams. The holes and slots made to make the assembly fit was done with the university's milling machine.

6.3 Electrical

6.3.1 Brakeout Board

In order to ease the connections from the controller card a brakeout board was made. The board was designed in Eagle and the PCB was milled by a CNC machine. With the brakeout board, the connections are made by standard plugs. The brakeout board is powered by the sbRIO and provides +5 volt, ground and signal for the rpm sensor, signal and ground for the ESC's, IMU connector, analog input for the volt and current measurement. In addition the brakeout board has also a connector for the XBee wireless transceiver with a 3.3 volt regulator for it.

Since the lipo batteries can be ruined if they reach under a certain level it is necessary supervise the voltage of the sbRIO at all time. Therefore an "AttoPilot" (chapter 4.3) was mounted between the sbRIO and the batteries. To prevent the controller board to draw too much current a 1 ampere fuse was also implemented to protect the sbRIO.

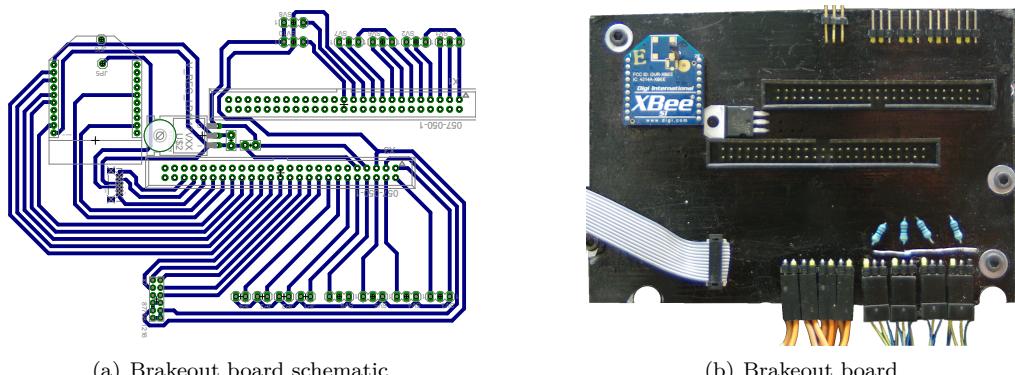


Figure 6.7: *The brakeout boards*

7. Control System

It was decided to implement a simple controller to test the performance of both the test rig and the quadcopter [29] [9]. Testing gave impressive results, and in addition to the size of the thesis it was decided to not implement a more advanced control system [23]. The control system is based on figure 7.1.

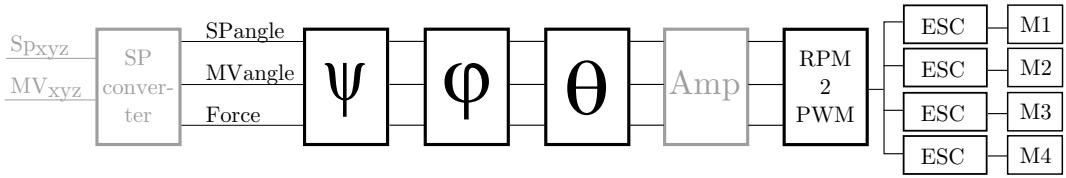


Figure 7.1: Main view of the control system, grayed out blocs are not in use

The main purpose of the control system is to control the quadcopter at given euler angles, phi, theta and psi. A position control will be an improvement but is not implemented. The position controller can calculate different position setpoints for the angle controller described in this chapter and the total motor thrust, and hence work independent of the angle controller. Since the position controller is not yet implemented the control system in this chapter is described as it was tested.

7.1 Yaw-controller

The total motor thrust is set as a variable. This thrust is the thrust that is required to hoover or to accelerate the quadcopter in the local Z-axis. The more thrust, the faster the quadcopter will accelerate. But as described in chapter 3.2.1 the yaw is controlled by altering the motor thrust between the four motors. The yaw controller divides the total force to the two motor pairs, 1-2 and 3-4. This way the total force is kept constant and a moment is applied to yaw the quadcopter around the body Z-axis.

Since the relationship of the motor thrust and moment is constant the quadcopter will not yaw as long as the total thrust is equal for the two motor pairs.

7.2 Phi-Theta controller

The phi and theta controller is the controller that controls the phi and theta angle of the quadcopter. This is done by dividing the motor pair force from the yaw controller to the two motors. This is done in two steps, first the phi angle is corrected, then theta. Both of them use the same controller, shown in figure 7.2.

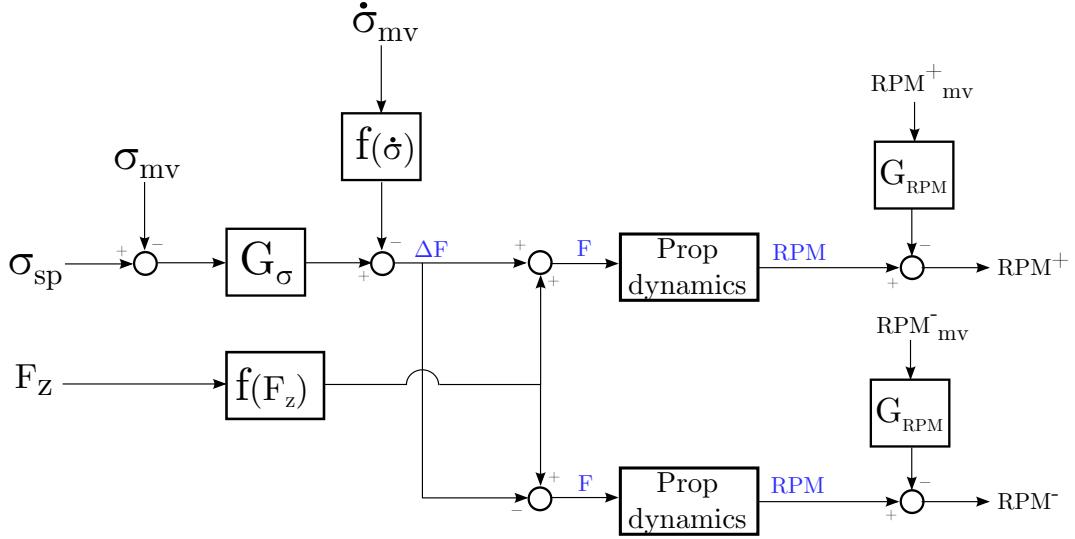


Figure 7.2: Block diagram of the phi and theta controller

The total force from the yaw controller which the motor pair is to put out, F_Z , is the force that is divided by the two motors. For one of the motors the force factor is increased, and for the other motor the force is decreased by the same value, namely ΔF . The difference of the two motor forces creates a moment which will correct the angular error. At first ΔF is calculated by the angle error of the angle setpoint and the measured angle in addition to a gain. To put a damper to the regulator ΔF is reduced with the angular velocity around the given axis. This way the moment will be reduced with an increased angular velocity and will slow down the system.

The way the angular velocity reduces the force difference is a bit special, it is not a pure gain. Because the higher the angular error is the higher angular velocity should be allowed to faster eliminate the error. The angular velocity should also be restricted more if the quadcopter is rotating away from the setpoint than it should be if it is rotation towards it. Thus a function was implemented.

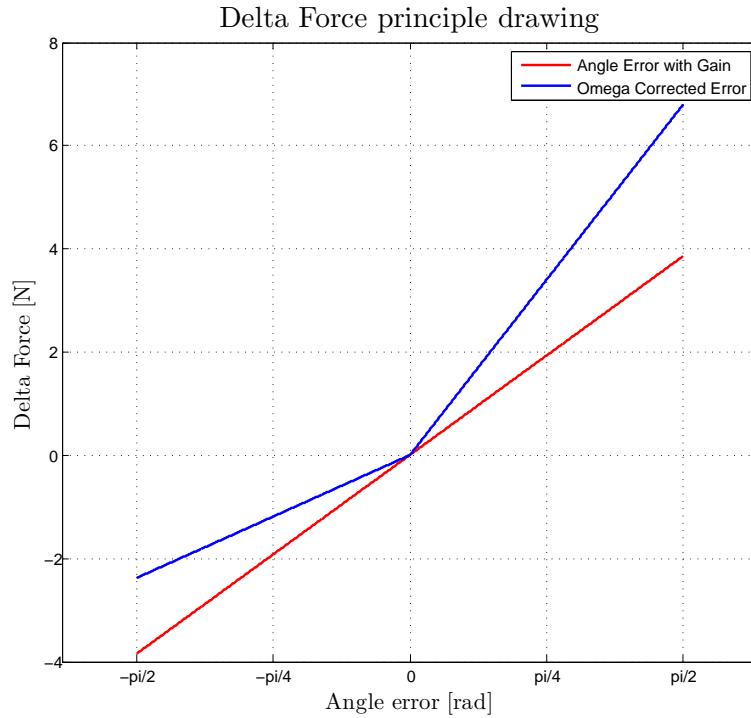


Figure 7.3: Principle drawing of the ΔF with angular velocity set to -1 rad/sec

The function of the angular velocity is calculated as

$$f(\dot{\sigma}) = \text{Omega}_{XY} \cdot \frac{1}{2} \cdot \left[\left(\text{abs}(1 + \text{sign}(\text{AngleError} \cdot \text{Omega}_{XY})) \cdot \frac{\text{Omega}_{GradientDec}}{\frac{\pi}{2}} \cdot \text{abs}(\text{AngleError}) \right) + \left(\text{abs}(1 - \text{sign}(\text{AngleError} \cdot \text{Omega}_{XY})) \cdot \frac{\text{Omega}_{GradientInc}}{\frac{\pi}{2}} \cdot \text{abs}(\text{AngleError}) \right) \right] \quad (7.1)$$

From the propeller testing (Chapter 4.11.1) a relationship between the propellers rpm and the propeller thrust was found. This is used in the control system to convert the desired force into the corresponding rpm. A P-controller is used to correct the desired rpm with the measured rpm. RPM-sp is the rpm needed for the desired task, while the difference is the controller gain.

7.3 Current controller

A current controller is considered as an improvement to the system, and not yet implemented. The current has a way faster response than the rpm, hence a faster controller can be implemented. The sensors needed for this task is tested and ready for use.

8. Programming

8.1 Data flow

The quadcopter and the control system is designed to be autonomous, i.e. it should be able to do a mission without the need of a radio link to the base station. The computer is used as a surveillance unit where all the needed data from the quadcopter is monitored. Setpoints are sent to the quadcopter from the base station computer to the quadcopter, and the quadcopter is then acting.

Locally on the quadcopter the sbRIO card is handling all the data. Sensor data are read by the FPGA module and sent to the real time processor. The real time processor is computing the sensor data in addition to the communication data from the XBee. Based on this the control system is controlling the quadcopter. The dataflow is shown schematically in Figure 8.1.

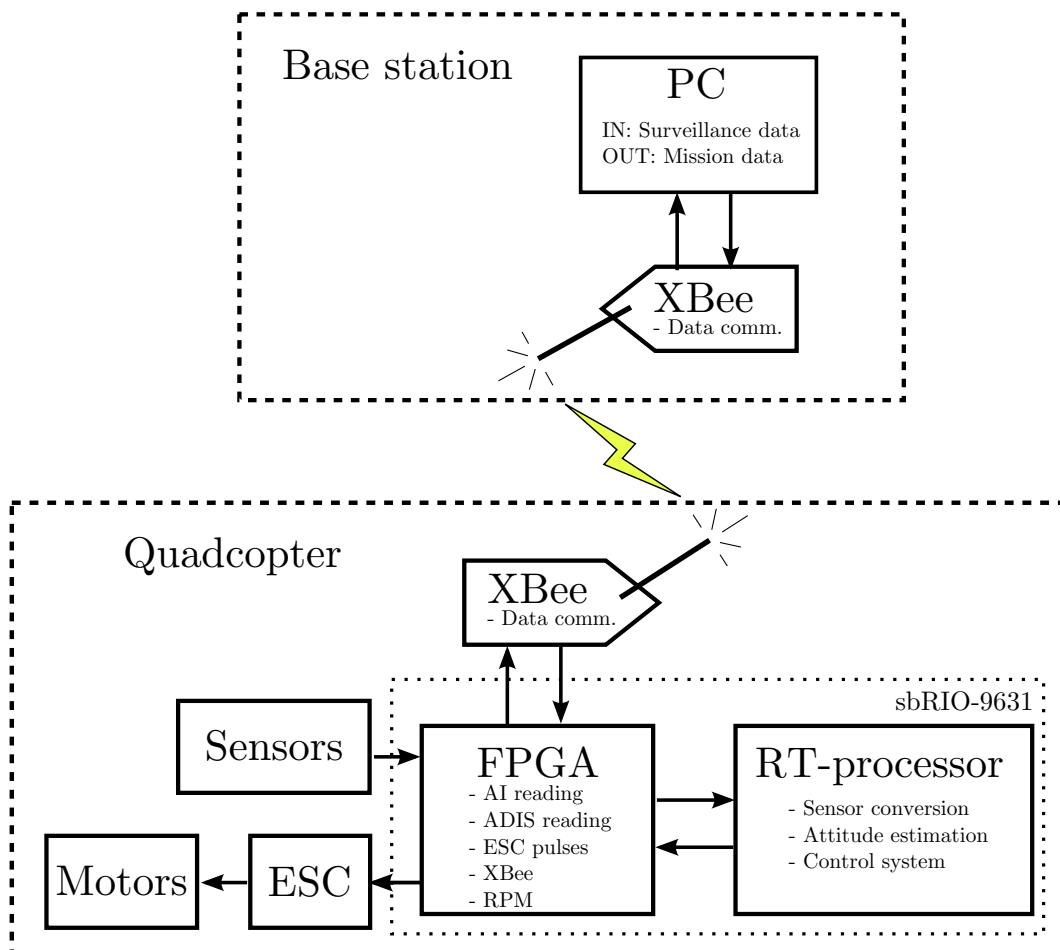


Figure 8.1: *Block diagram of the dataflow.*

Before the quadcopter is to take off it needs to run through a calibration and test sequence. When this is completed the main control loop will run continuously. This is the loop that will control the euler angles of the quadcopter.

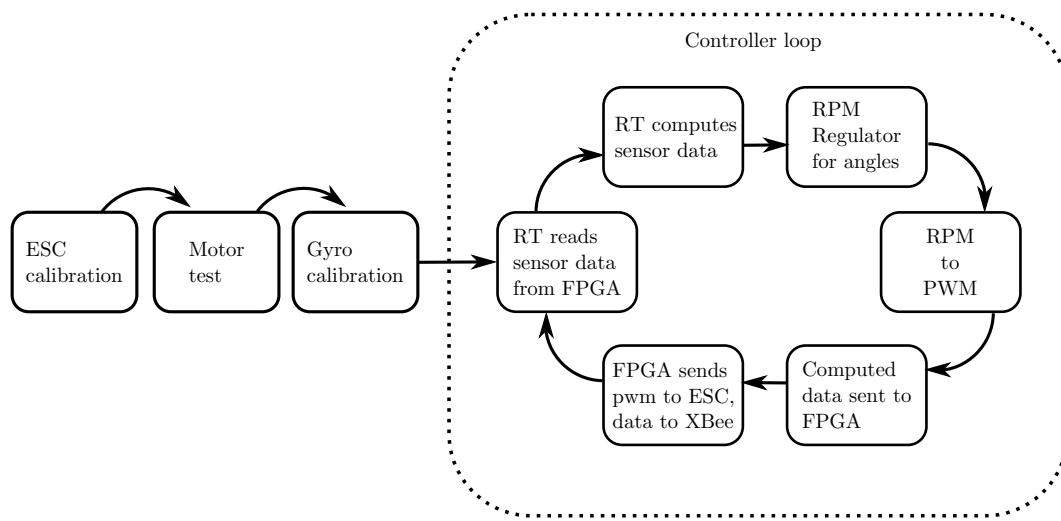


Figure 8.2: Flow chart of the control system

Since the controller loop is using every component on the quadcopter it is described first. The controller loop is not affected by the prior loops hence the programming structure will not be affected. The sensors will be explained on an FPGA level first, since this is the origin of the I/O. Then the regulator will be explained. The output of the regulator is rpm for the motor. The pwm generation for the ESC is the last part of the controller loop that is explained. Each section will show where the program code is implemented in the controller loop with a figure. In this figure everything except of the explained code will be grayed out. By staring to explain the controller loop it will be easier to get an idea of how the different sensors are implemented. After the controller loop is explained the ESC calibration, motor test and gyro calibration will be explained.

The LabVIEW program consists of many virtual instruments (VI's) and subVI's which is placed inside a VI. On the FPGA it is possible to run loops in parallel. With parallel loops it is possible to run many VI's at the same time where they all can read and write to the I/O's. A total overview of the entire FPGA program is shown in Figure 8.3.

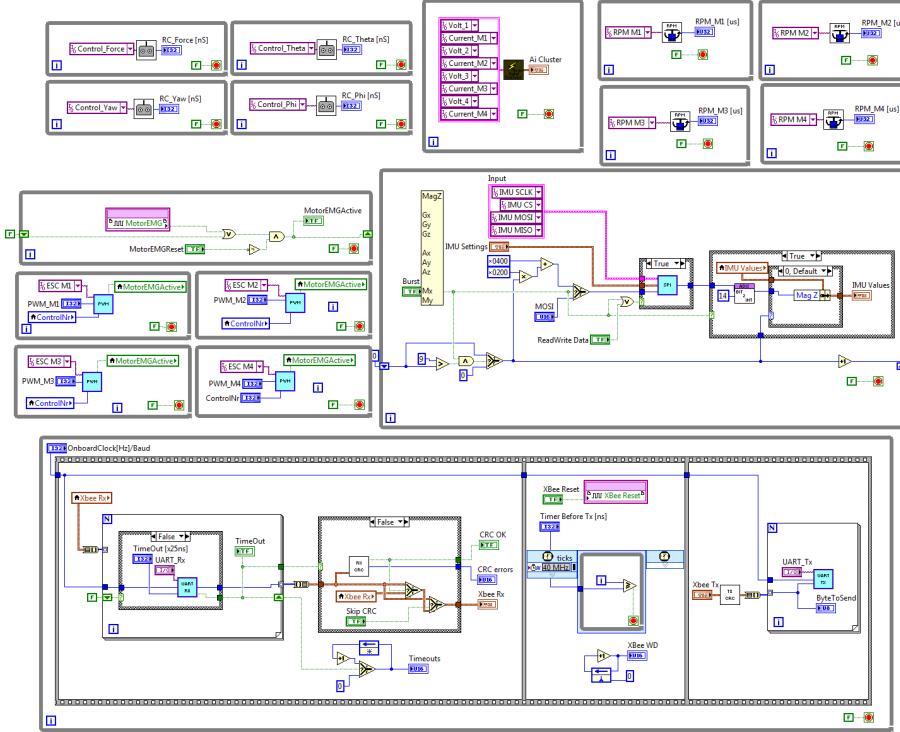


Figure 8.3: Overview of the FPGA program

It is not possible to run loops in parallel on the RT-Target. Therefore the program must be built with a proper dataflow. An overview of the RT-target program is shown in Figure 8.4.

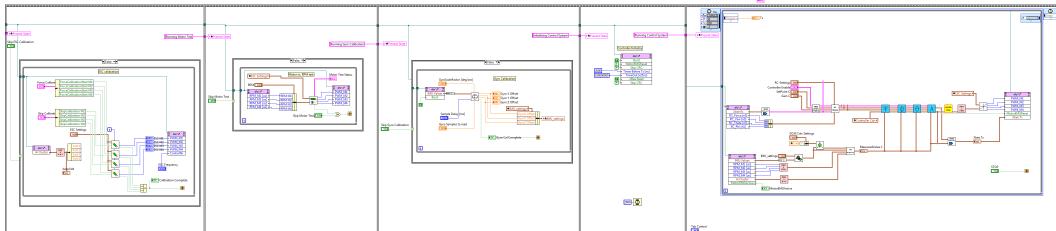


Figure 8.4: Overview of the RT-target program

Since the FPGA continuously reads the sensor data the control system does not have to ask the FPGA to read the different sensors. Once the control system needs to get a sensor value it is already read by the FPGA and can be read just like a local variable. This enhances the cycle time significantly, and also simplifies the code.

To fully understand the aspects of this chapter some basic LabVIEW programming is needed.

8.2 XBee

The XBee communicates with a standard serial communication link, UART or RS232. To reduce the load of the real time processor and keeping the cycle time to a minimum UART communication from the FPGA was chosen for the XBee. There are different UART subVI's on the internet, but they are quite difficult to understand, and is often not written for a specific task. Hence UART receive and transmit subVI's was created (appendix C.1 and C.2).

8.2.1 CRC

In order to verify the sent and received data a cyclic redundancy check, CRC, is implemented. This is a very simple check which prior to a data transfer adds every data value, but without carry. The calculated value is also a 8-bit integer. This value is also sent to the receiver. The receiver then do the same math by adding the numbers together, but without the CRC number. The calculated CRC and the received one are then compared. If they match the received data is considered valid, if not the data are rejected. There is still a chance of corrupted data even if the CRC is correct, but the probability is very small.

8.2.2 Data flow

The XBee transmission is as everything on the FPGA independent of the control system. It will send and receive data as fast as it can, and will send the same data over and over if its not changed by the control system. The quadcopter first receives its data, wait for a short amount of time and then transmits its data. The quadcopter is the master of the transmissions. So in case the quadcopter and the base station gets out of synchronization, i.e. both of them are waiting to receive data, an interrupt is triggered on the FPGA. The quadcopter then stops waiting for data, and instead starts its data transmission. This way they get synchronized.

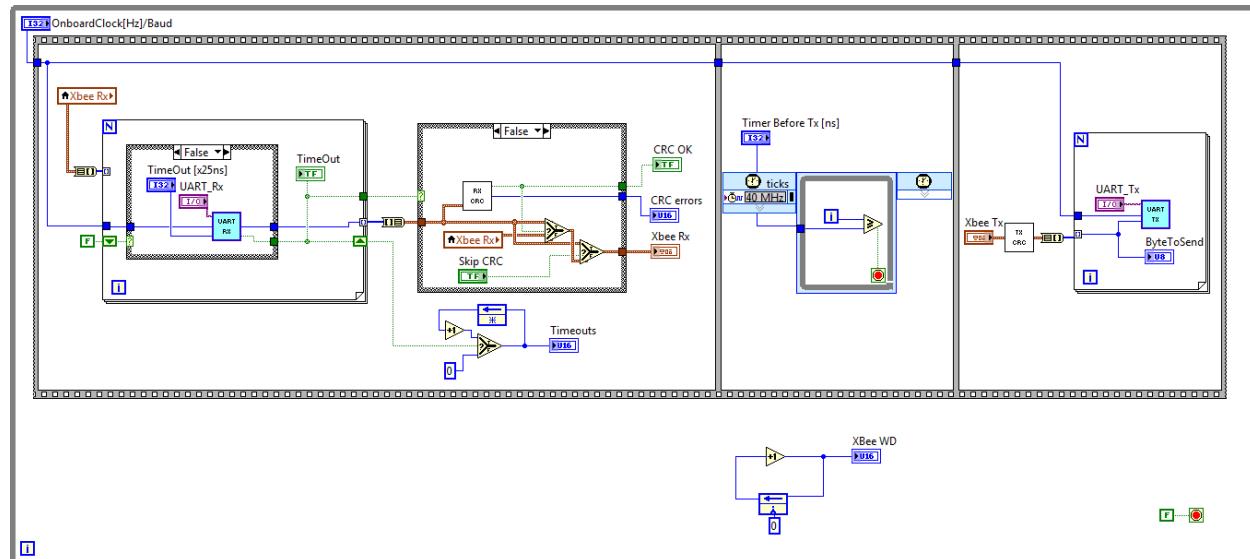


Figure 8.5: XBee implementation on the FPGA

8.2.3 XBee data conversion

The XBee is operating with 8 bits, so the data going back and forth must be converted into 8 bits integers. It would be possible to combine many bytes in order to send a floating point number, but a resolution of 8 bits per data was considered good enough, it will also reduce the amount of data being sent over the bus. Every data in the tables, 8.1 and 8.2, are sent at each cycle, and in the given order. The gains and force are described in the control system chapter 7.

An 8 bit alarm array is also implemented in order to send different alarms. And a enable byte is also sent from the base station which can enable different controllers on the quadcopter.

Base station to quadcopter:

What	Min Value	Max Value
Setpoint X	-2.5 [m]	+2.5 [m]
Setpoint Y	-2.5 [m]	+2.5 [m]
Setpoint Z	-2.5 [m]	+2.5 [m]
Gain Angle Error	1	0
Gain AngleDotInc	1	0
Gain AngleDotDec	1	0
Force	4 [kg]	0
Enable byte	N/A	N/A
Alarm byte	N/A	N/A
CRC	0	255

Table 8.1: Data sent from the base station to the quadcopter, with its real values extrema

Quadcopter to base station:

What	Min Value	Max Value
Alarm	N/A	N/A
Angle Phi	-90 [deg]	+90 [deg]
Angle Theta	-90 [deg]	+90 [deg]
Angle Psi	-90 [deg]	+90 [deg]
Position X	-2.5 [m]	+2.5 [m]
Position Y	-2.5 [m]	+2.5 [m]
Position Z	-2.5 [m]	+2.5 [m]
Setpoint Phi	-90 [deg]	+90 [deg]
Setpoint Theta	-90 [deg]	+90 [deg]
Setpoint Psi	-90 [deg]	+90 [deg]
RPM motor 1	0 [rpm]	10000 [rpm]
RPM motor 2	0 [rpm]	10000 [rpm]
RPM motor 3	0 [rpm]	10000 [rpm]
RPM motor 4	0 [rpm]	10000 [rpm]
Volt motor bat 1	9 [V]	14 [V]
Volt motor bat 2	9 [V]	14 [V]
Volt sbRIO bat	20 [V]	25 [V]
Status	N/A	N/A
CRC	0	255

Table 8.2: Data sent from the quadcopter to the base station, with its real values extrema

The conversion is done using the standard formula for a straight line:

$$Y = ax + b \quad (8.1)$$

Based on 8.1 the two equations for generating the 8-bit integer and converting it back to a real value are:

$$Value = 8_{bit} \cdot \frac{Max - Min}{255} + Min \quad (8.2)$$

$$8_{bit} = 255 \cdot \frac{Value - Min}{Max - Min} \quad (8.3)$$

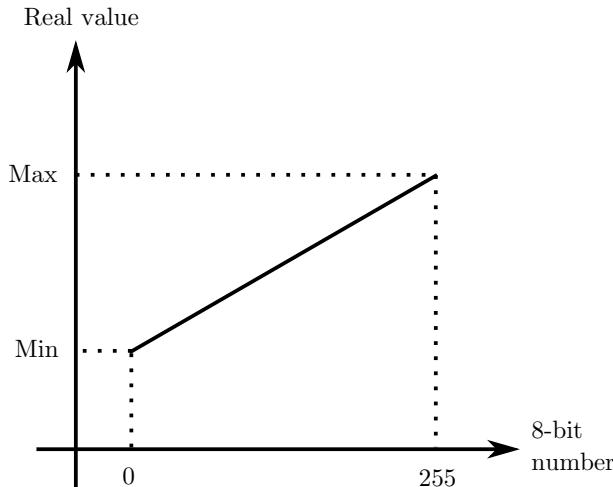


Figure 8.6: Scaling of the 8-bit numbers

Both of the conversions are done by the RT-processor in the main control loop.

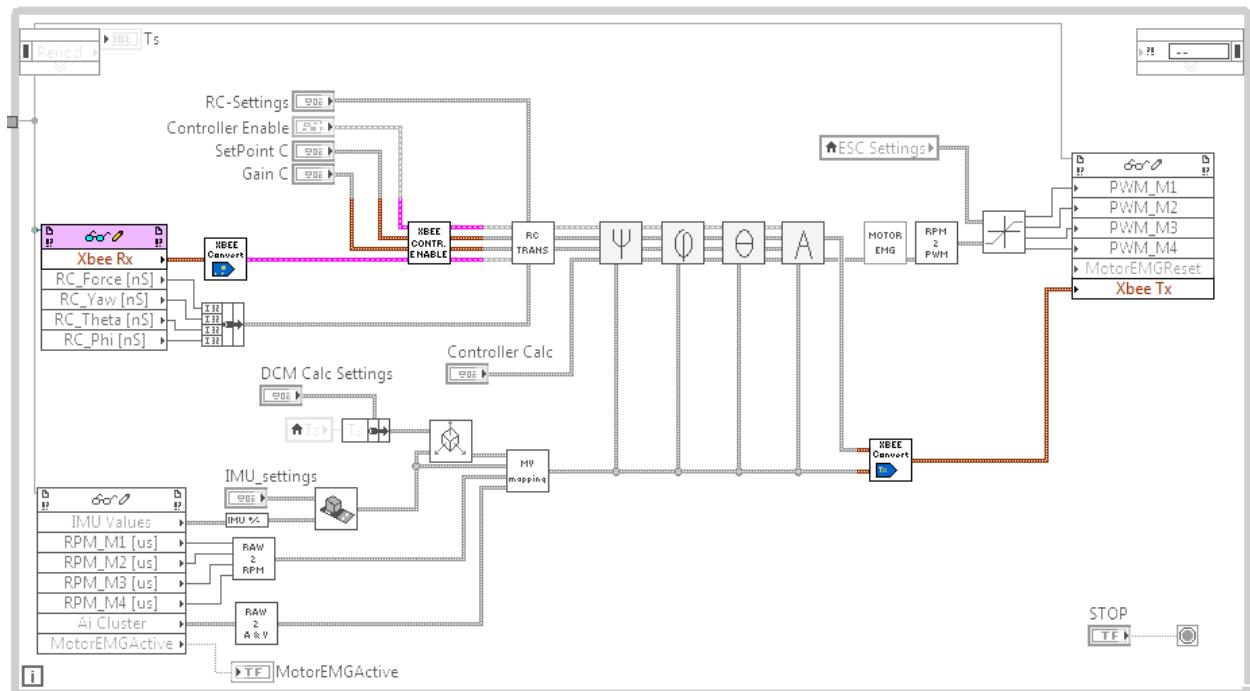


Figure 8.7: XBee conversion in the control loop

8.3 RC Transmitter

In order to ease the testing of the control system an RC transmitter was implemented. The transmitter controls the angle setpoints of the quadcopter, it does not control the different ESC directly. This is a much easier way of controlling the quadcopter in case of an unstable happening, than typing different commands over the XBee radio link [12]. The transmitter is only implemented for testing purpose. By using the XBee the transmitter control can be turned on or off.

The receiver connected to the quadcopter is using RC signals of 1-2 ms at 50 Hz. Four inputs were connected to the sbRIO in order to control the total force and yaw, and setpoints for the phi and theta angle. Note that the transmitter is not controlling the motors them selves, just setpoints for the regulator.

The signals are read on the FPGA:

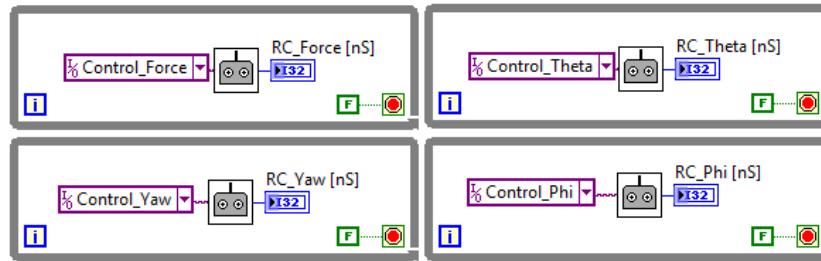


Figure 8.8: *RC transmitter readings on the FPGA*

To measure the RC-signal a timed loop of 25 nanoseconds cycle time counts the number of loop iteration elapsed for a high signal. This results in a very accurate resolution.

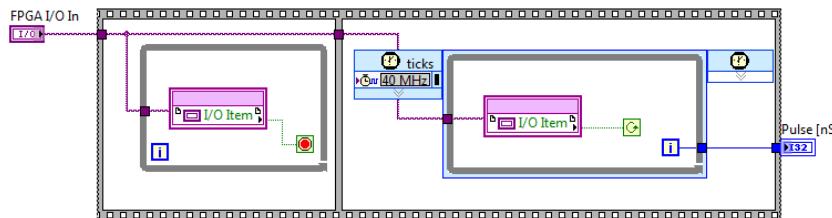


Figure 8.9: *RC transmitter pwm measurement*

In the control loop the 8 bit integer is converted to a real value just like the XBee conversion. The minimum and maximum values are presented in table 8.3.

Base station to quadcopter:

What	Min Value	Max Value
Setpoint Force	0 [kg]	4 [kg]
Setpoint Psi	-0.5	0.5
Setpoint Phi	-0.2 [rad]	0.2 [rad]
Setpoint Theta	-0.2 [rad]	0.2 [rad]

Table 8.3: Data sent from the RC transmitter to the quadcopter, with its real values extrema

A yaw controller is not implemented. The transmitter is anyway controlling the yaw to some point. A value of -1 equals thrust only from the counter clockwise rotating motors, and +1 for the clockwise ones. With a value of 0 the quadcopter will output equal force to the two motor pairs.

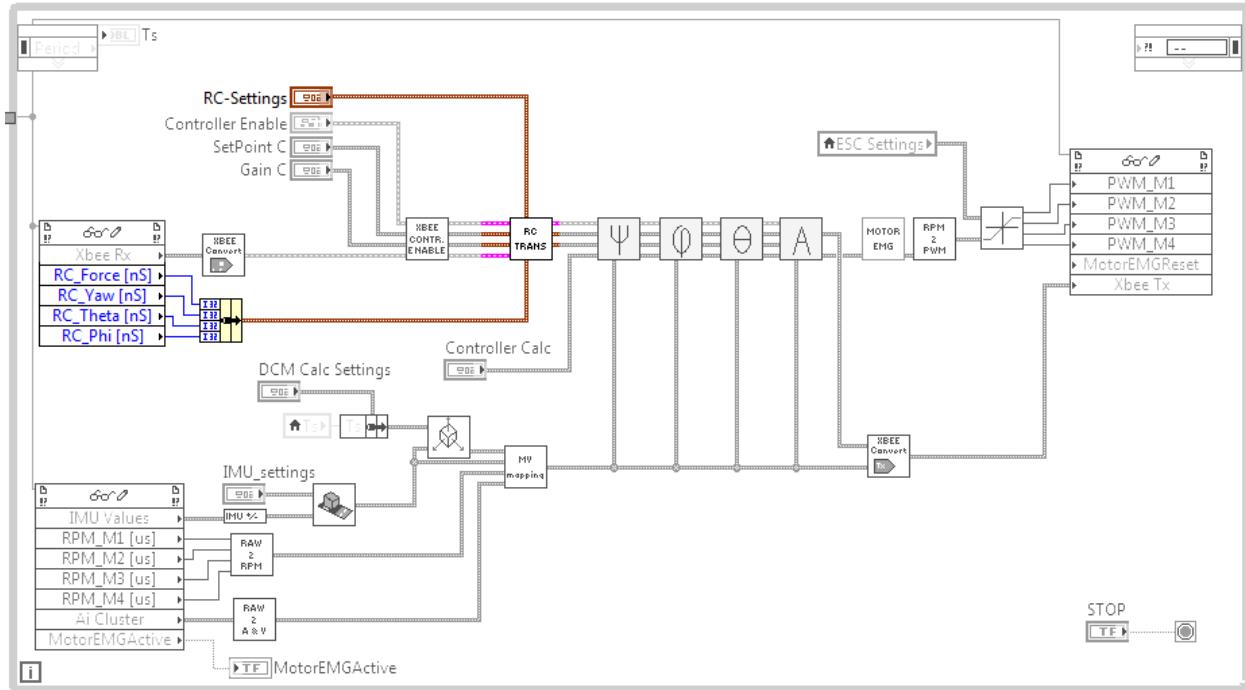


Figure 8.10: *RC conversion in the control loop*

8.4 IMU

The IMU is using the SPI protocol, this makes it a bit more advanced. The SPI needs a master, which is the sbRIO, and a slave, the IMU. In general the SPI protocol works by the master requests a specific data from the slave by sending an address to read from. The slave then returns the data from the specified address back to the master. There is one wire for data out and one for data in. This makes it possible to send and receive data simultaneously between the master and the slave. When the master requests a value the clock pulse is started. During this cycle the master sends a request for a value, and since the slave cannot reply the requested data before the transmission is completed, it transmit the value of the previous request, see Figure 8.11.

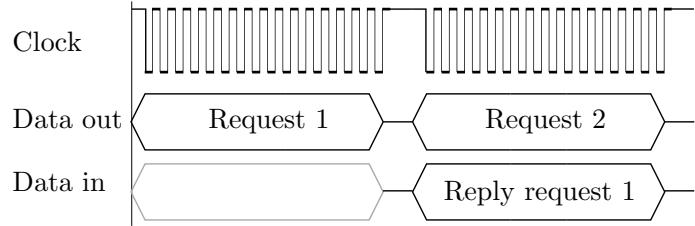


Figure 8.11: Illustration figure of a SPI data exchange referred from the quadcopter

The control system should also just read the values of the IMU straight from the FPGA. This means that the SPI protocol needs to be located on the FPGA. A SPI subVI like this was not found and had to be made from scratch. The SPI VI (appendix C.3) is implemented on the FPGA and is designed to read and write commands from the IMU.

The following is read from the IMU:

Data to read	Address [hex]
Gyroscope X	0x 0400
Gyroscope Y	0x 0600
Gyroscope Z	0x 0800
Accelerometer X	0x 0A00
Accelerometer Y	0x 0C00
Accelerometer Z	0x 0E00
Magnetometer X	0x 1000
Magnetometer Y	0x 01200
Magnetometer Z	0x 01400

Table 8.4: Data read from the IMU and its address

The IMU VI reads the IMU continuously, and is nine cycles long, one for each of the nine elements. Since the reply of the new request is the previous request, the first data to be read is the Magnetometer Z, then the Gyroscope X, Y, Z and so on. The addresses to read are not stored as arrays, since arrays occupies a lot of space on the FPGA and is hard to handle. But according table 8.4 the addresses starts with 0x0400 and is increasing with 0x0200 for each cycle. After nine cycles it starts all over again. This logic is implemented in order to read from the correct addresses.

Even though the shift registers are 16 bits long, the data transmitted is only 14 bits. The conversion from the binary 14 bits number to a signed integer is also done on the FPGA. Finally the value is stored in the cluster containing every data from the IMU.

The IMU VI is also capable of sending one address request in addition to the "burst mode" described above. This lets the operator access every register in the IMU (appendix C.4). The IMU VI is illustrated in Figure 8.12.

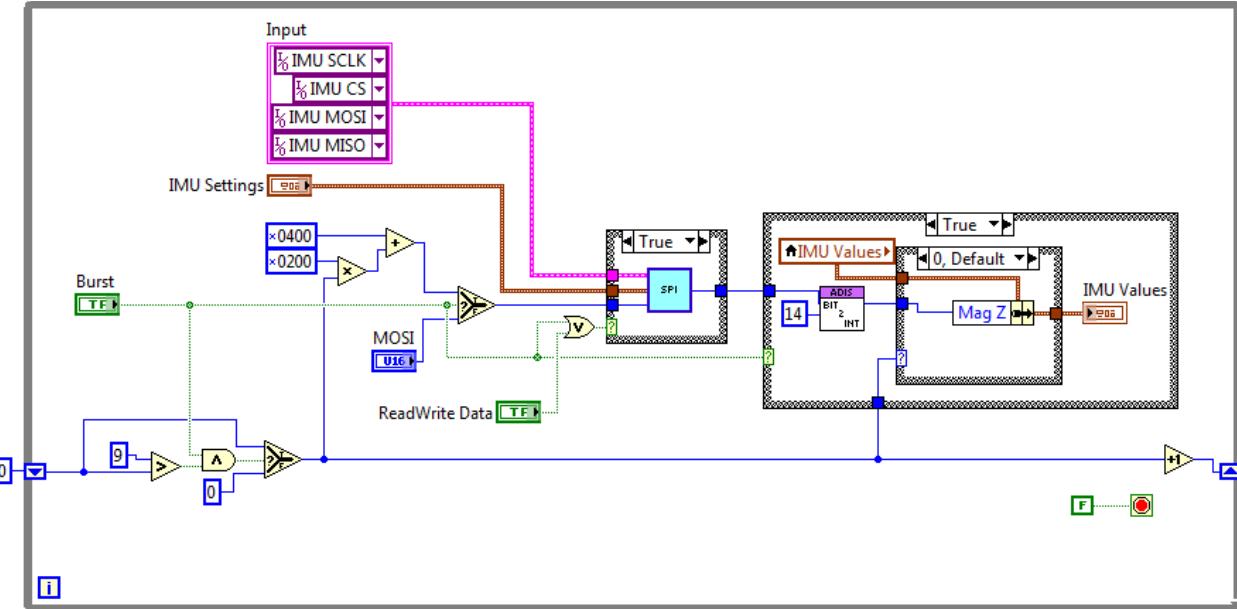


Figure 8.12: IMU reading on the FPGA

With the burst mode and the way the SPI is implemented on the FPGA it takes $9\mu s$ to sample one sample. With a total of 9 samples during normal operation it takes less than 0.1 ms to sample every needed data. This makes it possible to average many data without affecting the cycle time of the RT-target.

In the controller loop the values from the FPGA come as a cluster. At first the sign of the values are corrected to get the same positive direction as Figure 2.1. Then the IMU characteristics like scaling factors and offsets are taken into account (appendix C.5). The final VI is then updating the attitude estimation.

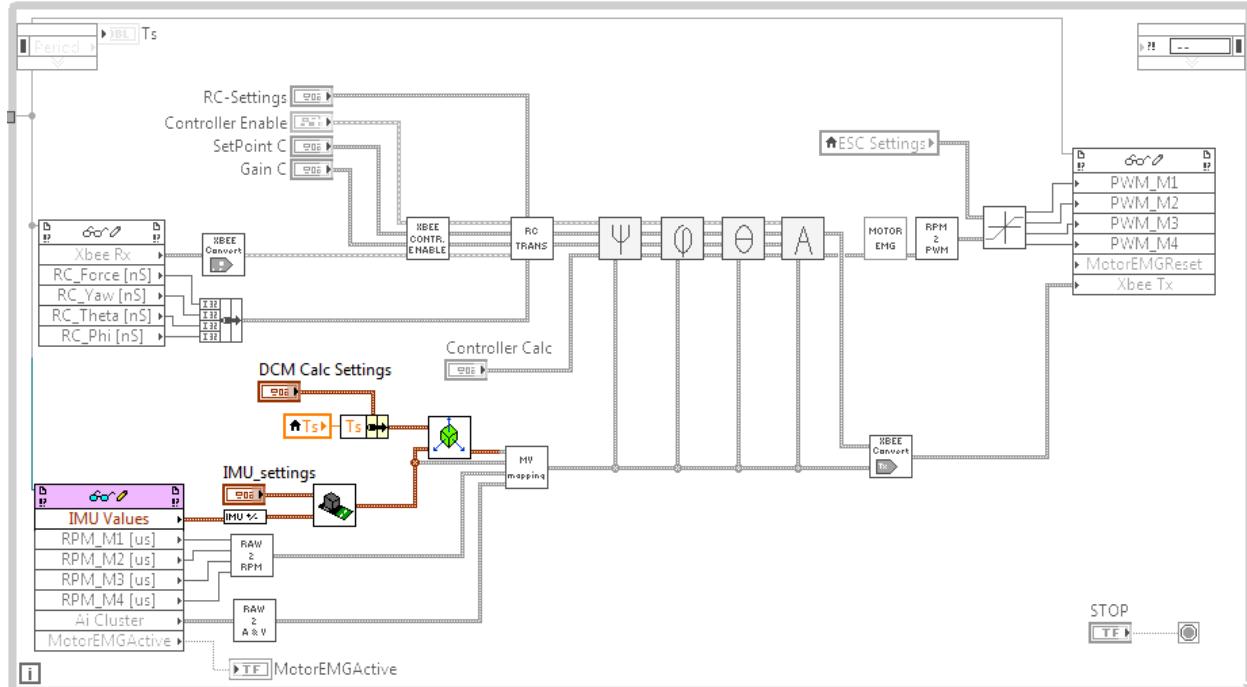


Figure 8.13: IMU and its conversion in the control loop

The attitude estimation is programmed as explained in Figure 5.2, and the following sections. It is capable of determine any rotation of the quadcopter.

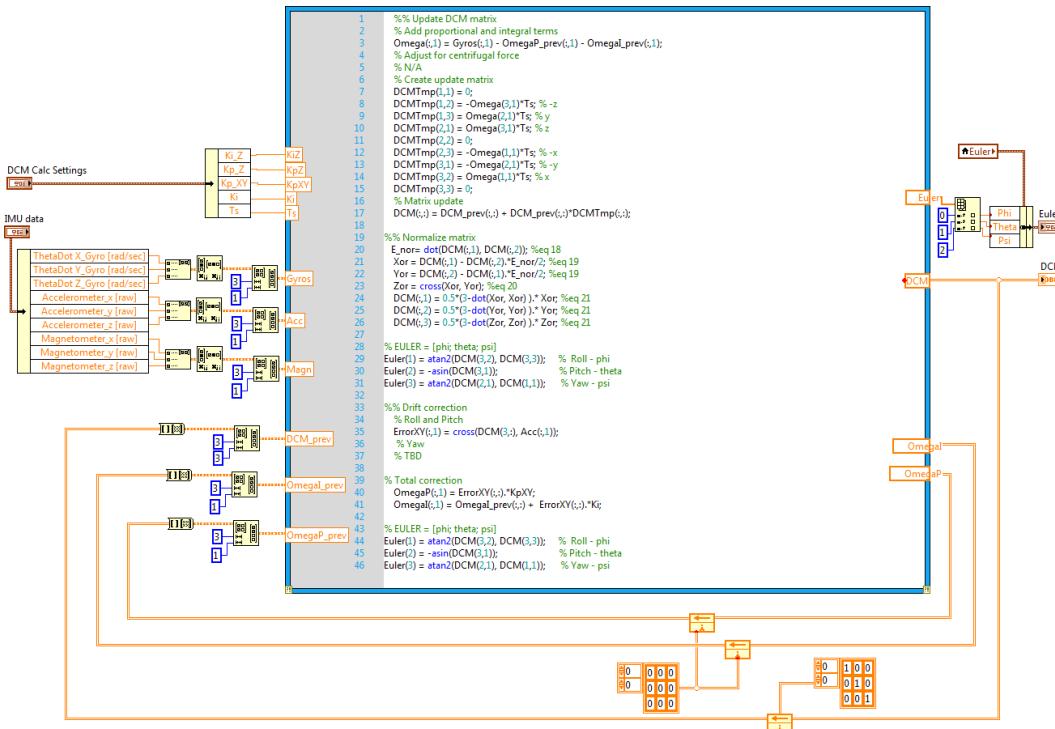


Figure 8.14: Attitude estimation subVI

8.5 RPM

The rpm is measured using hall effect sensors. The sensor puts out zero or five volts, depending of the motors rotation. The width of the magnets are not known nor the air gap between them. To measure the rpm the period from zero to zero volt is therefore measured. It is assumed an equal distance between the magnets.

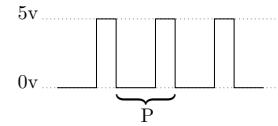


Figure 8.15: Illustration figure of the time period P for the rpm measurement.

The rpm measurement is implemented on the FPGA module and consists of three frames. The first two frames makes sure the pulse has just become low, the last frame measures the periode P in μs .

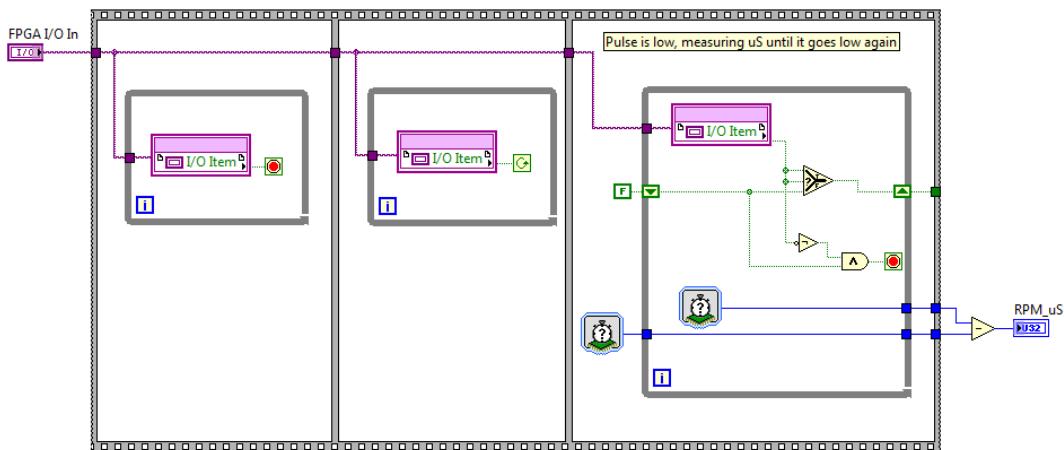


Figure 8.16: LabVIEW code for the rpm measurement.

The real rpm is calculated in the control system. With 7 periods pr rotation (chapter 4.2) the rpm is calculated as:

$$RPM = \frac{1}{P \cdot 10^{-6}[s/period] \cdot 7[periods/rotation]} \cdot \frac{60[s]}{1[min]} \quad (8.4)$$

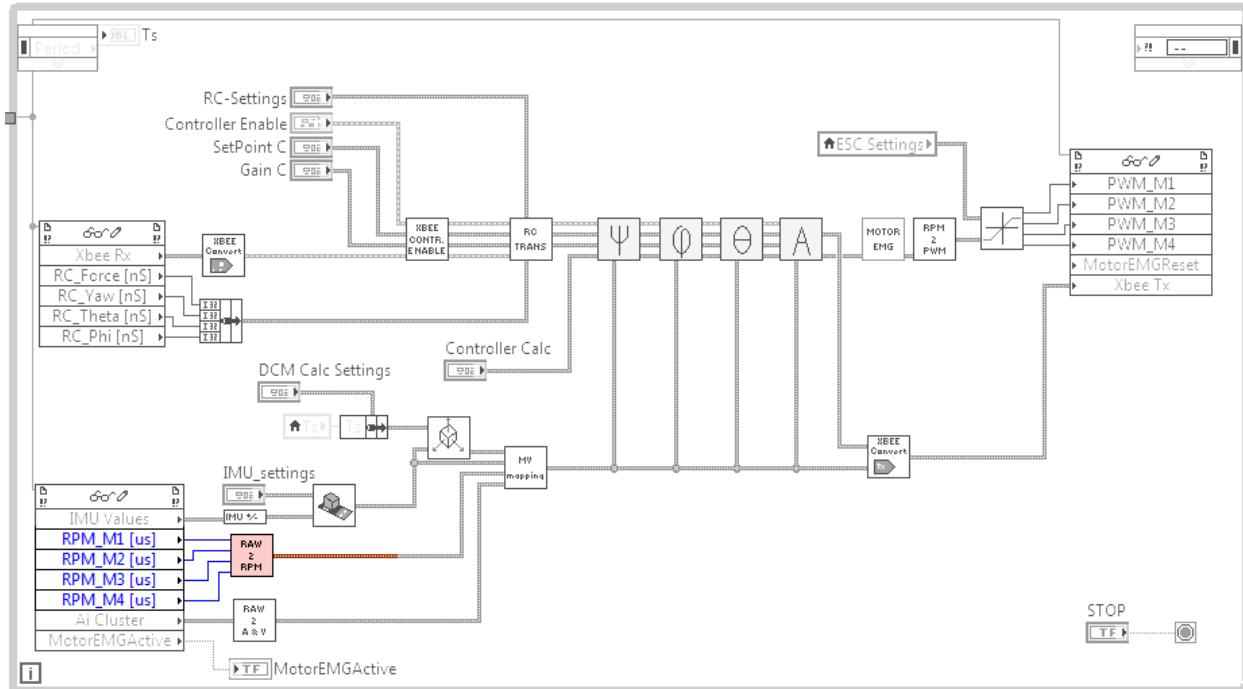


Figure 8.17: *RPM in the control loop*

8.6 Volt and Current

The volt and current value is an average of 1000 samples.

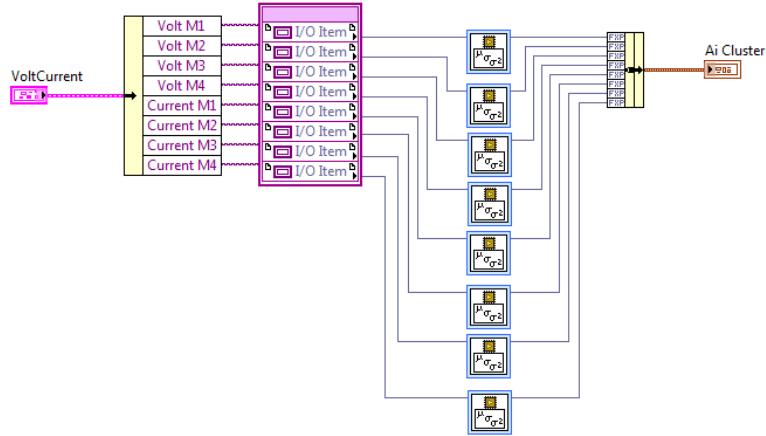


Figure 8.18: Volt and current readings on the FPGA

On the FPGA the readings come as a fixed point, corresponding to the analog value. To convert the fixed point value to the real voltage and current it needs to be multiplied by the scaling factors, (chapter 4.3), the conversion is shown in appendix C.6.

The volt and current are the last inputs prior to the control system.

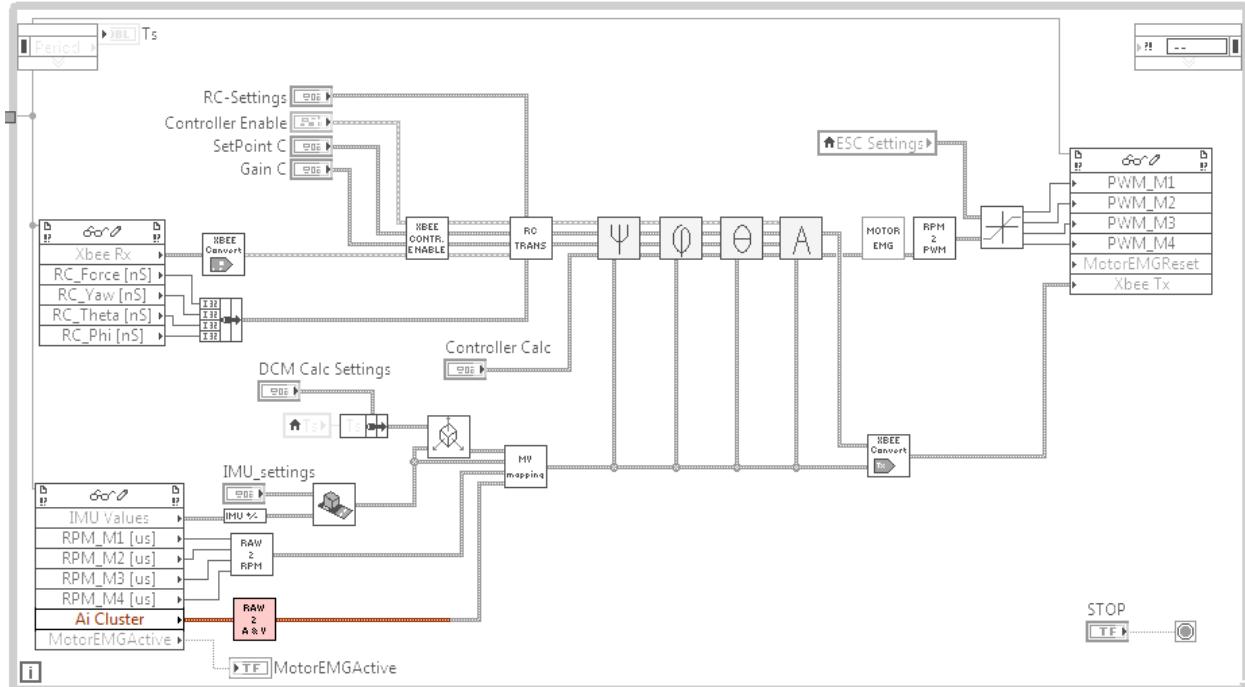


Figure 8.19: Volt and current in the control loop

8.7 Control System

When the sensors are read the control system starts to adjust the rpm for each motor with respect to the setpoints. The first subVI for the control system is the yaw controller. This subVI will distribute the total force from the setpoint to the two motor pairs. This way the yaw controller does not adjust the rpm, but the force each motor pair is to put out. The phi and theta controller will then adjust the given force from the yaw controller to its two motors in order to reach the setpoint.

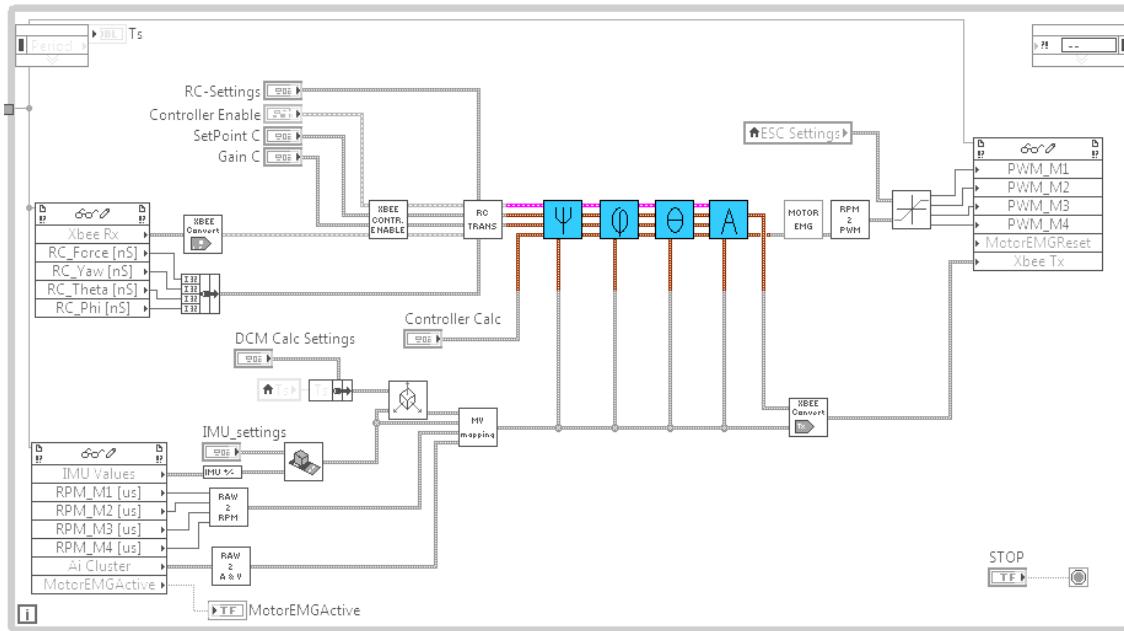


Figure 8.20: *Control system in the control loop*

8.7.1 Yaw controller

Due to the fact that the attitude estimation does not correct for drift around the Z-axis, the control system does not have a yaw controller. But it is possible to control the yaw with the transmitter. If the yaw controller is disabled the total thrust is divided equally by the two motor pairs C.7. But when enabled the difference is controlled by the transmitter.

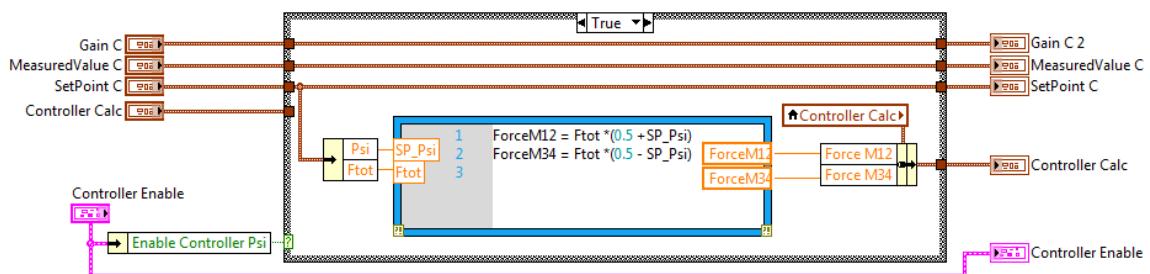


Figure 8.21: *Yaw controller realized in LabVIEW*

8.7.2 Phi and Theta controller

In order to have the same regulator for both phi and theta one subroutine is used for both of them. But they have different setpoints and gains, so the subVI's in Figure 8.20 are just mapping the data into the regulator. And the regulator subVI is located inside those (Figure 8.22). Since both motor 1 and 3 can give negative angular velocity they have the same input to the regulator. This is the same for motor 2 and 4.

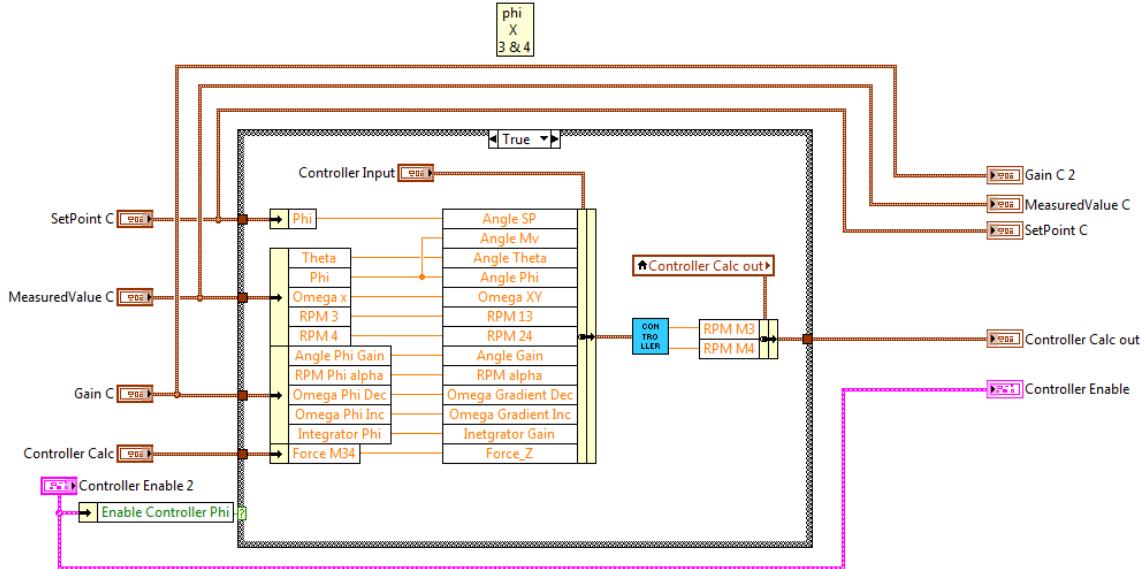


Figure 8.22: Mapping of values to the phi regulator

The mapping of the theta controller can be found in appendix C.8. The regulator is written inside a math script and is the same as the regulator described in chapter 7.2.

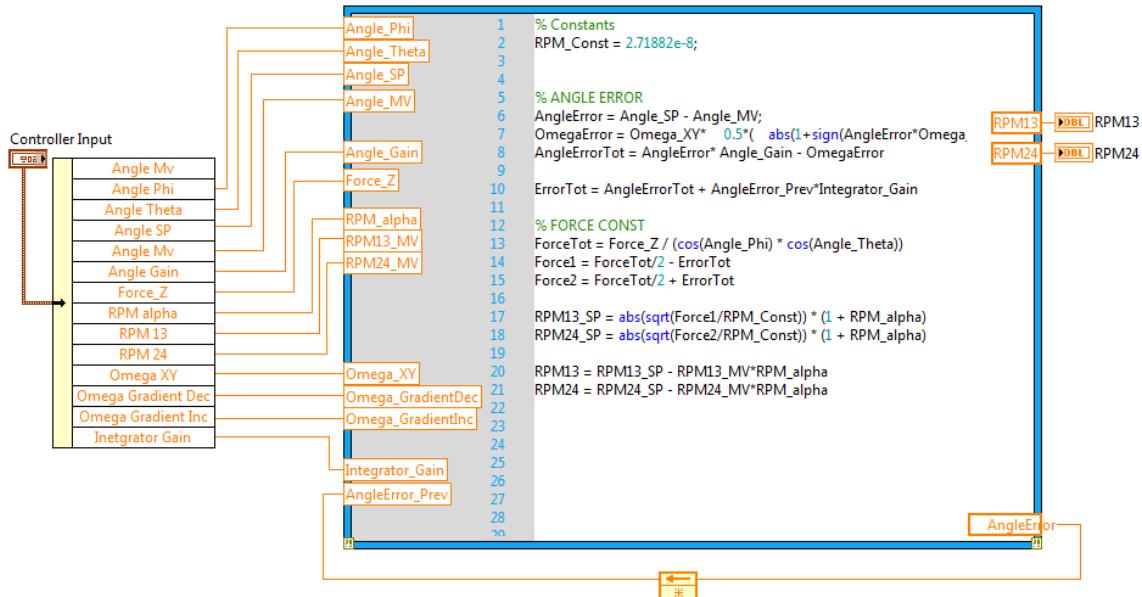


Figure 8.23: Regulator implemented in LabVIEW

8.8 ESC

In the main control loop the rpm is generated from the angle controller. In chapter 4.11.4 a linear relationship between the rpm and the pwm was found. In this relationship the timing is in milliseconds, but the FPGA operates with loop cycles of 25 nanoseconds. Equation 4.9 can be expressed as (LabVIEW code in appendix C.9):

$$\text{Loop cycles} = \text{RPM} \cdot 4.9 + 35000 \quad (8.5)$$

Before the number of loop cycles are sent to the FPGA a saturation block is implemented to prevent the FPGA to generate a false signal. This block limits the loop cycles to be between 40000 and 80000 cycles (appendix C.10).

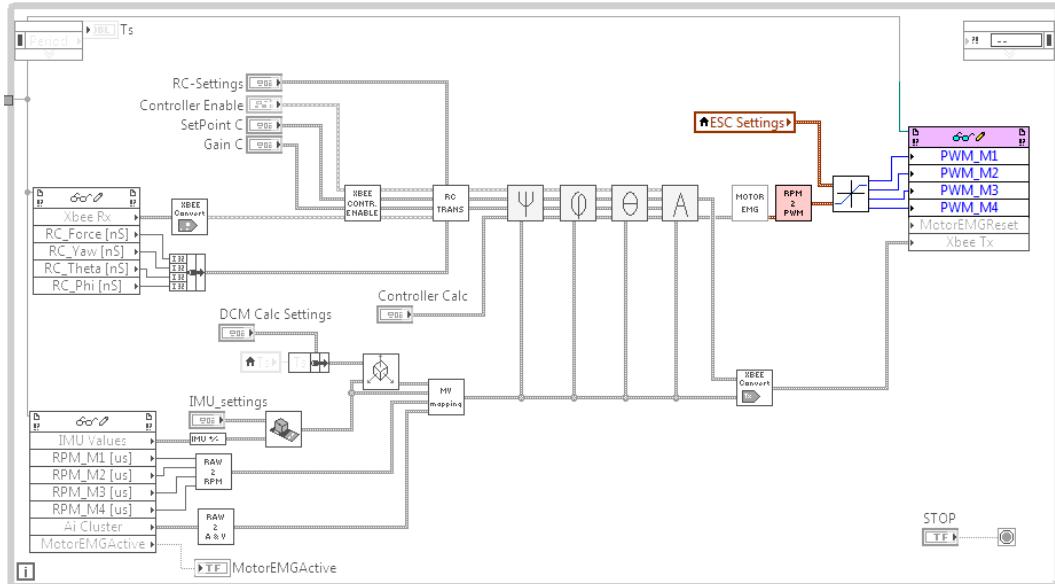


Figure 8.24: *ESC in the control loop*

On the FPGA the four motors are controlled separately. An external emergency stop is also implemented to preserve the safety of the quadcopter and the nearby personnel. The E-stop is activated with a high signal at one of the digital inputs. As the inputs are pulled high with internal pull-up resistors the E-stop is triggered once the wire between the input and ground is pulled apart. The E-stop forces a pulse length of 1ms to the ESC, which will stop the motor rotation.

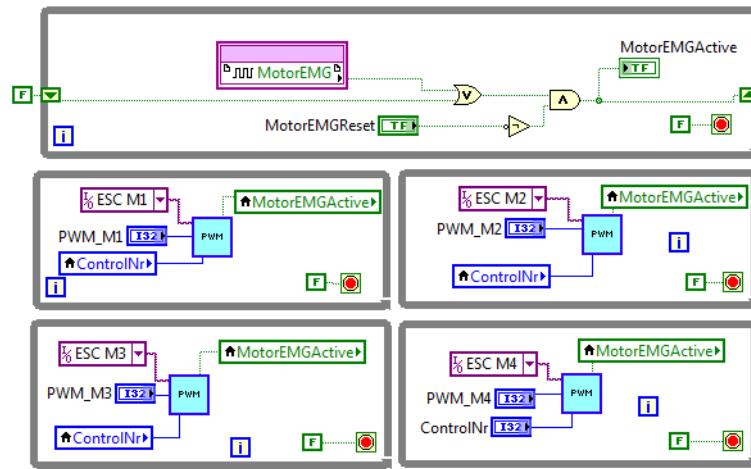


Figure 8.25: *ESC control on the FPGA*

The subVI controlling the pulse length is a timed while loop. The cycle time of the timed while loop is 25ns. Hence 40000 loop cycles equals 1ms and 80000 equals 2ms.

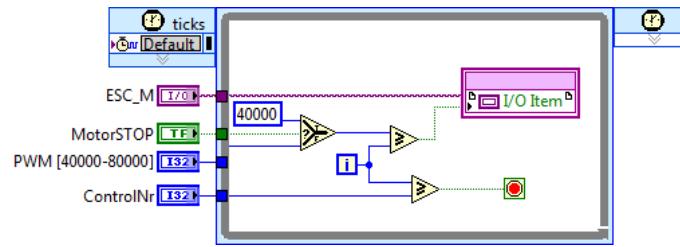


Figure 8.26: *ESC control on the FPGA*

8.9 Initialization of the Control System

The quadcopter is being initialized prior to the control loop. The initialization consists of ESC calibration, motor test and gyro calibration. The initialization is placed in a flat sequence structure and starts with the ESC calibration.

8.9.1 ESC Calibration

The ESC is calibrated to make sure the maximum and minimum speed command are set correctly. The calibration procedure starts off by feeding maximum speed command to the ESC before it is powered up. The max pulse will tell the ESC that a calibration will be performed. Once the ESC is powered up and beeping with pulses the signal is dropped down to minimum. Then a ramp from minimum to maximum is sent to the ESC. The calibration ends with a minimum speed command.

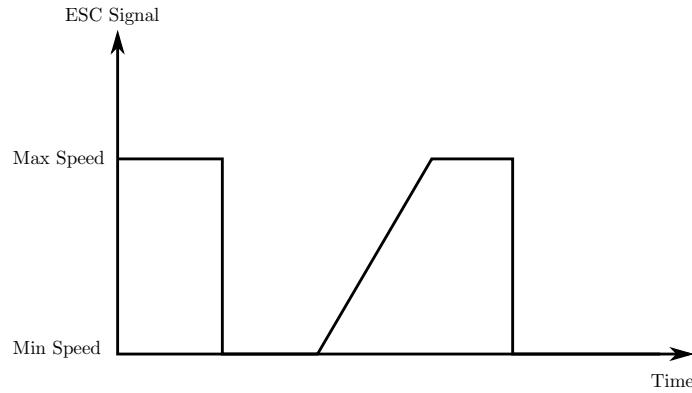


Figure 8.27: *ESC calibration procedure*

During the ESC calibration the voltage on each ESC is measured. Once the ESC is powered up the ESC calibration will start automatically. It is also possible to force a start of the motor calibration, and to skip the calibration. Those two options are mainly used during testing in the test rig. The whole procedure can be found in the appendix C.11.

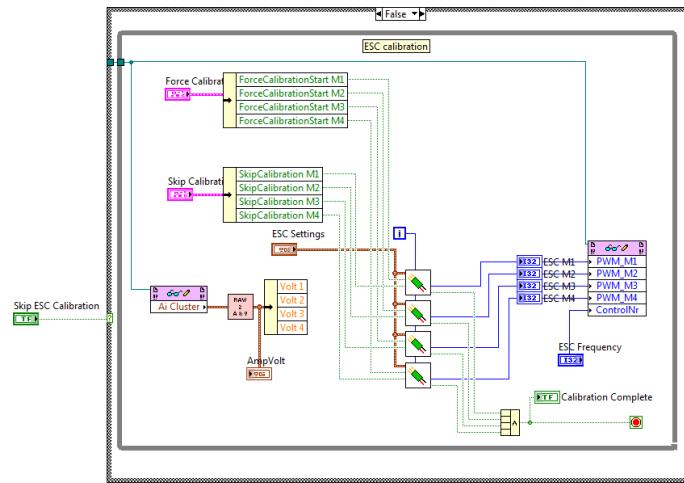


Figure 8.28: *ESC control on the FPGA*

8.9.2 Motor Test

The motor test is a test that verifies that each motor works as expected. The test runs one motor at a time at a low rpm. It then verifies that the motor that is being tested really is the motor that should rotate, and that the rpm is between a limit set in the program. If the motor is rotating too slow or too fast the motor test fails.

8.9.3 Gyro Calibration

The ADIS16400 has an internal gyro calibration. But a calibration procedure was implemented to always have control of the processes going on. The gyro calibration is calculating the mean value of 10000 samples. The offset is subtracted in the IMU calculation (chapter 8.4).

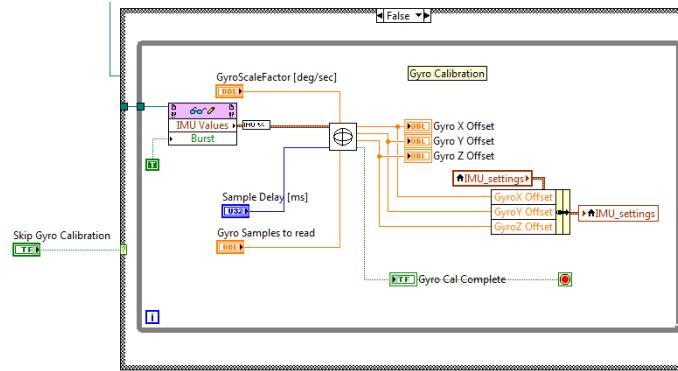


Figure 8.29: *ESC control on the FPGA*

The IMU calibration procedure for motor one is fully shown in appendix C.12.

8.10 Remote observer

To be able to monitor the quadcopter while in operation, there is needed for an Wirelink from the quadcopter to an basestation. There were programmed an LabVIEW program where data was received and interpreted. The information gathered from the quadcopter was Alarm status, angles, estimated position, angles setpoint, rpm for each motors, voltage for each battery and different kind of status messages.

The Program is built of 3 stages, the first stage is the receiving of data bytes from the quadcopter. The data received is put through an CRC routine as described in chapter 8.2.1, to make sure the data received is valid, if the CRC check is valid, the data is sent for further processing, if not, the last valid data will be the data that's processed.

The middle sequence structure, the data which are to be shown are mounted to indicators and blocks to handle the data at the right method. In the last sequence the input the user can send to the quadcopter from the front screen, is converted and packed ready to be sent. before it is sent to the XBee, an CRC number is added to the end of the string in the same way as described in chapter 8.2.1.

The front panel have three different tabs that can be chosen. It is important to notice that the alarm block which is placed to the left in the tabs is the same in every block, this is to make an uniform view so the user always know where to look, whenever the tab is operated. The first tab is the graphical view. Here the data is processed so the user can watch real time angular rotation of the quadcopter in a 3d space. There is also

made a map that shows the quadcopter position and the route it have flown. Other data as battery level, RPM etc is also shown in logical way, this view is high demanding of the computer where its being run. the second tab is a an operating view with the information only as numbers. Last tab is the cominication setup, where the settings for the wirelink can be altered.

8.11 Programming Summary

The programming on the sbRIO located on the quadcopter is split in two parts. One part is located on the FPGA level, while the other part is processed by the real time processor. The FPGA handles the I/O's very fast, and does parallel loops. With parallel loops the FPGA can read and write to every I/O at the same time, and one loop is not dependent on the others. The FPGA can only handle integer numbers and fixed points. Many functions are not supported on the FPGA so the complete control system is implemented on the RT-target.

The RT-target first initializes the control system with motor calibration, motor test and a gyro calibration. Once the controller loop is initialized it will run continuously and stabilize the quadcopter. For every cycle the controller loop is reading sensor data from the FPGA, updating the motor rpm with the regulators and also update the FPGA with new values for the ESC and the XBee wireless communication.

The total cycle time of the control system is 20ms which is considered high for controlling a quadcopter.

The remote observer is capable of observing what is happening on the quacopter in operation. For testing purpose it was also implemented a gain control of the regulator. This made it possible to adjust the regulator with no wires attached. The values are sent as 8 bit integers which was considered to have good enough resolution for the values sent. A CRC is also implemented in order to verify the received data. As constructed the wireless link works flawlessly.

9. Full Test

9.1 Test Bench

To start the test of the control system the quadcopter were placed in the test bench. the quadcopter were in the start hooked up to power supply, to allow for more concentrated testing. When hooked up to the power supply the quadcopter would have wires hanging down on one side and thus making the center of gravity some offset. This effect was not in concern as the testing would mostly focus on response than error angle.

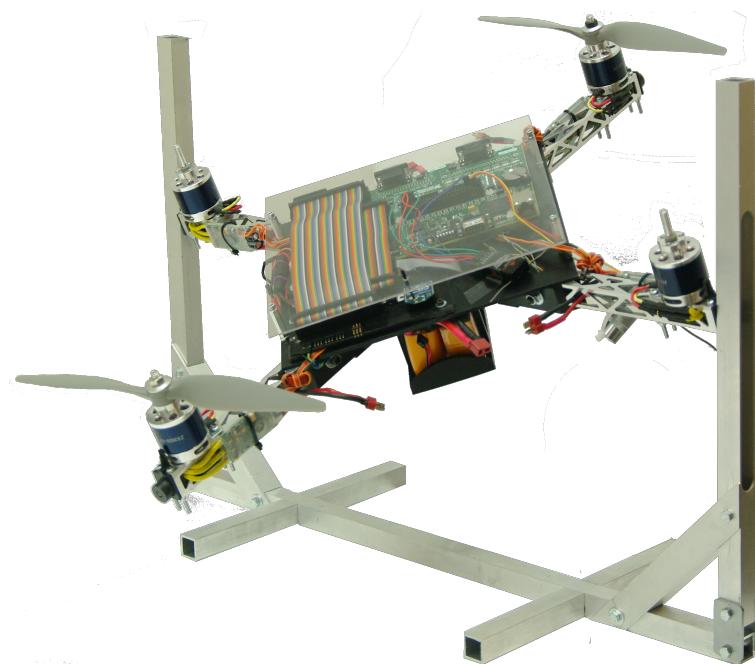


Figure 9.1: *Quadcopter placed in the test bench*

The testing showed the quadcopter could be quick to regulate if knocked out of position, and the controller parameter were tweaked to give an optimal performance with the current controller. Two tests was logged, one with a step from 0.2 radians to 0 radians, and one where an external impact affects the system:

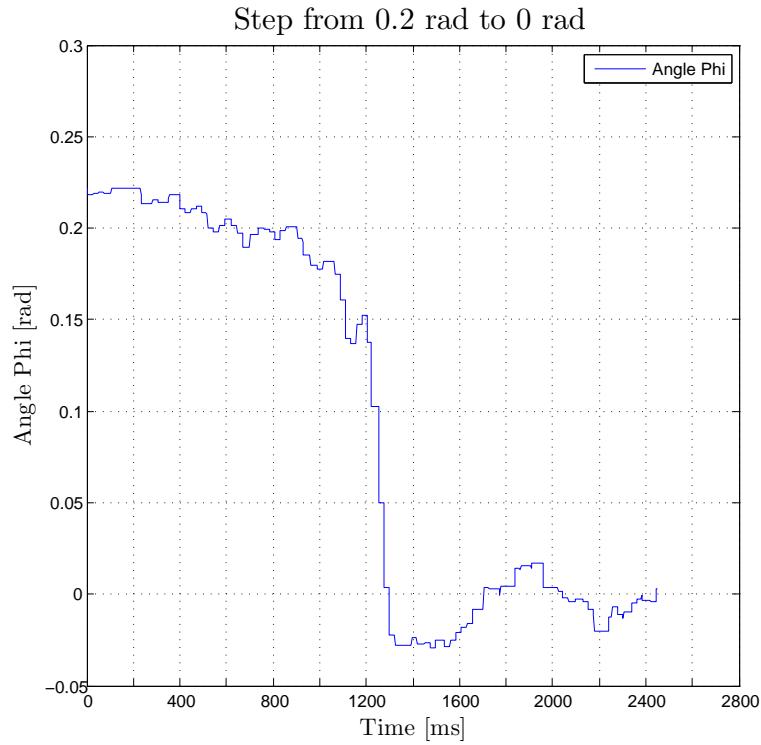


Figure 9.2: *Tilt test from 0.2 rad to 0 rad*

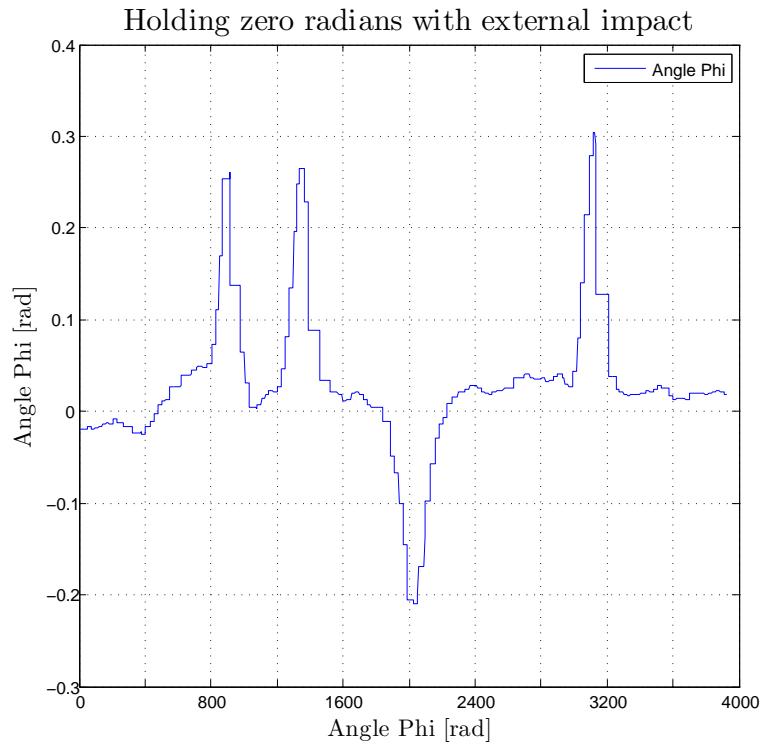


Figure 9.3: *Four impacts where the controller tries to hold zero radians*

A problem that were countered while working in the test bench, was an effect which made the controller to behave different at slower speed than intended for hoovering. It was theorized that this effect is a result

of the moment of inertia of the propeller and motor. The inertia would have an greater impact at lower speeds since the thrust from the propellers is depended of speed squared. To achieve a percentage increase in thrust at low speed requires a higher percentage speed increase than needed at higher speed. This could be overcome with a gain that will change with the square of the force. This way the gains could be higher for lower forces, and be automatically reduced with higher force. This would be correct since the change of force is a square function.

9.2 Free Flight

Free flight was tested but the quadcopter was not able to stabilize. Free flight is quite different than testing in the test rig. In the rig there is friction present, which adds damping to the system. This and the fact that the stabilization in the rig was tuned at wrong thrust are some of the reasons that the quadcopter did not fly stable.

10. Conclusion

A dynamic model of the quadcopter is derived. Deriving this model requires knowledge of rotation matrices which converts forces between different reference systems. This conversion is also described in the report. The dynamic model is generated by summing forces and moments for the quadcopter. The dynamic model is derived to get an understanding of how the quadcopter behaves. 3D CAD models of each component was created in order to get the inertia matrix of the quadcopter. The dynamic model was not simulated using Matlab or any other simulation software. But it was used to derive the attitude estimator of the quadcopter.

The attitude estimator is working very good in terms of measuring the attitude of the quadcopter referred to the fixed navigation frame. It is robust and it measures any rotation possible for the quadcopter. A drift correction is also implemented for the tilt angles phi and theta. Drift correction is not yet implemented for the yaw, but this was not considered essential in order to make the quadcopter to fly. In addition to the inertial measurement unit the other sensors are also working very good. The rpm of the motors are continuously monitored and have almost no noise. Voltage and current sensors are also doing good job and is a vital part of the controller board which needs a certain volt in order to operate.

The motors and propellers are generating enough force to lift a payload of approximately 6 kg. The high motor force can also be used for fast stabilizing and heavy control of the quadcopter. The motors in a combination with the electronic speed controller has a very low time response compared to other systems, the ESC operate at 100Hz in contrast to the regular 50Hz controllers.

From the 3D model of the quadcopter parts were made. Plastic parts were printed on a 3D printer, and the aluminum parts were milled in a CNC. From the 3D model the center of gravity was adjusted to be localized in the center of the inertial measurement unit. This is not a must but it makes the attitude estimation less complex.

The quadcopter can communicate with a base station over a wireless radio link using XBee transceivers. Through the wireless link controller gains were set, and so the setpoints to the quadcopter. The radio link made it possible to tune the regulator without the need of any wires attached to the quadcopter.

Prior to free flight stabilization around one axis was tested. This was achieved in a test rig built for the quadcopter. The quadcopter got stable in the test rig with the regulator designed. It was able to hold a setpoint of 0 degrees, and to go from 0 to 11.5 degrees quite fast. The regulator also corrected for different impacts. But even though it was stable in the test rig it was not able to stabilize in free flight. More tuning is needed on the regulator, and maybe it also needs to be revised.

The built quadcopter is a good educational platform where many sensors are implemented. There is also room for even more sensors and maybe a IP-camera for visual recognition. It is relatively easy to program and to expand to more advanced tasks. And with some adjustments on the tilt regulator the quadcopter is ready to fly.

In order to make the quadcopter autonomous the following research topics should be investigated further:

- Tune and revise the controller to stabilize free flight.
- Add yaw drift correction for the IMU.
- Add yaw regulator to the controller.
- Implement a position estimator, or an absolute method.
- Add a conversion from translative position error to tilt of the quadcopter

Bibliography

- [1] Ian Petersen Abhijit Kallapur and Sreenatha Anavatti. A robust gyroless attitude estimation scheme for a small fixed-wing unmanned aerial vehicle. *Proceedings of the 7th Asian Control Conference, Hong Kong, China, August 27-29, 2009.*
- [2] David Allerton. *Principles of Flight Simulation*. Wiley, 2009.
- [3] Paritosh Banerjee Amitava Bose, Somnath Puri. *Modern Inertial Sensors And Systems*. Asoke K. Gosh, PHI Learning Private Limited, M-97, Connaught Circus, New Delhi-110001, 2008.
- [4] A.J. Baerveldt and R. Klang. A low-cost and low-weight attitude estimation system for an autonomous helicopter. *IEEE International Conference on Intelligent Engineering Systems*, 1997.
- [5] Cuong C. Quach Edward F. Hogge Thomas H. Strom Boyd L. Hill Sixto L. Vazquez Kai Goebel Bhaskar Saha, Edwin Koshimoto. Battery health management system for electric uavs. *Aerospace Conference, 2011 IEEE*, 2011.
- [6] Steven G. Krantz Brian E. Blank. *Calculus multivariable*. Key College Publishing, 2005.
- [7] Vardan Semerjyan Corentin Cheron, Aaron Dennis and YangQuan Chen. A multifunctional hil testbed for multirotor vtol uav actuator. *Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference*, 2010.
- [8] P.C. Rakshit D. Chattopadhyay. *Elements Of Physics, Volum 1*. New Age International Publisher, 2004.
- [9] Jorge Miguel Brito Domingues. Quadrotor prototype. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, 2009.
- [10] S. Theil F. Holzapfel. *Advances in Aerospace Guidance, Navigation and Control*. Springer-Verlag, 2011.
- [11] Christopher Jekeli. *Inertial navigation systems with geodetic applications*. Walter de Gruyter GmbH & co, 2000.
- [12] Peter I. Corke Jonathan M. Roberts and Gregg Buskey. Low-cost flight control system for small autonomous helicopter. *IEEE International Conference on Robotics and Automstion*, 2003.
- [13] Roumeliotis S.I. Jun M. and Sukhatme G.S. State estimation via sensor modeling for helicopter control using an indirect kalman flter. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1999.
- [14] Øyvind Magnussen Kjell E. Skjønhaug. The quadcopter investigation. Technical report, University of Agder, 2010.
- [15] Jack B. Kuipers. *Quaternions and rotation sequences*. Princeton University Press, 2002.
- [16] R. Mahony M. Euston, J. Kim and T. Hamel. A complementary filter for attitude estimation of a fixed-wing uav. *6th International Conference on Field and Service Robotics, FSR07*, 2007.
- [17] Stanley S. McGowen. *Helicopters: an illustrated history of their impact*. ABC-CLIO, Inc, 2005.

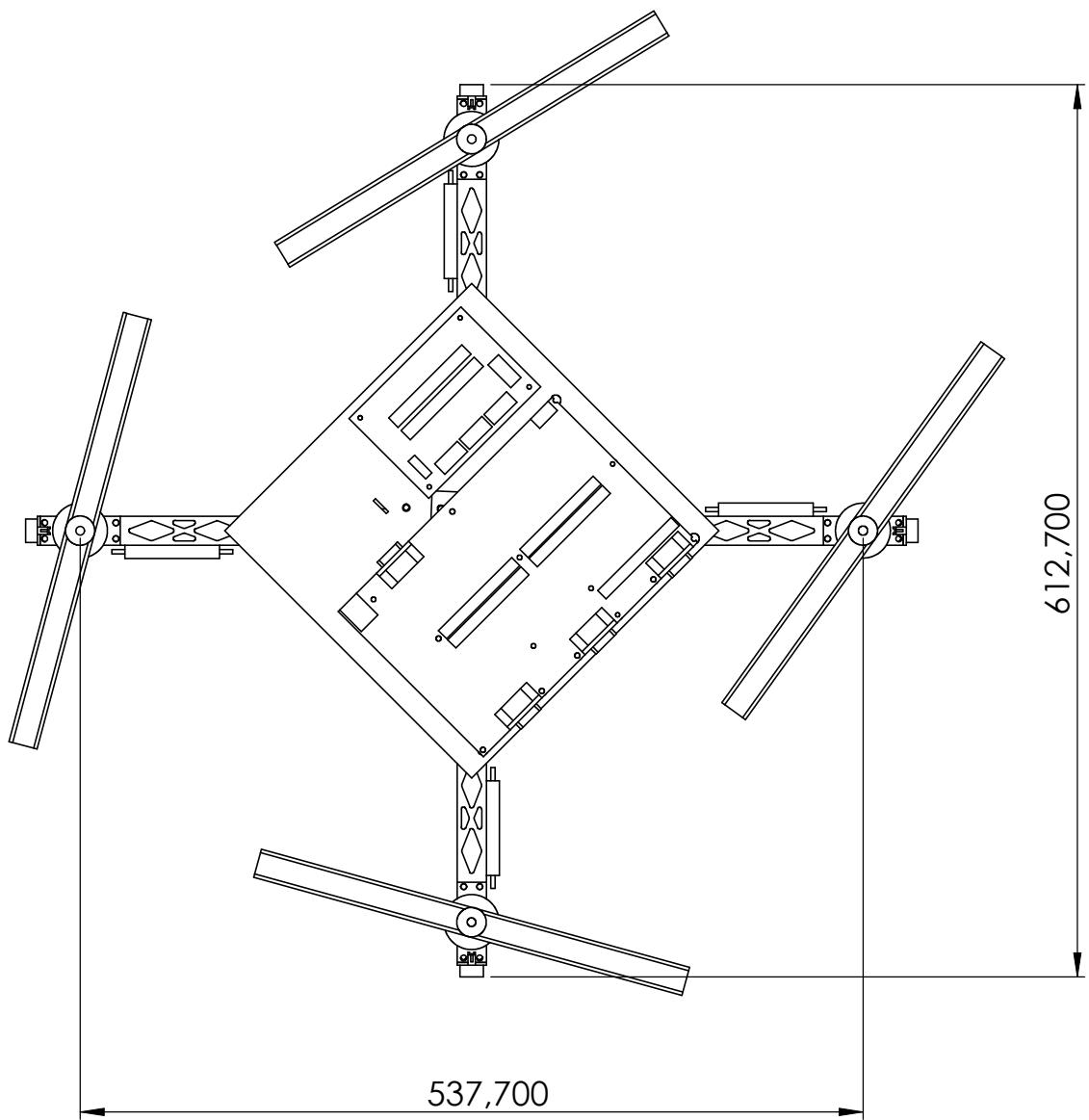
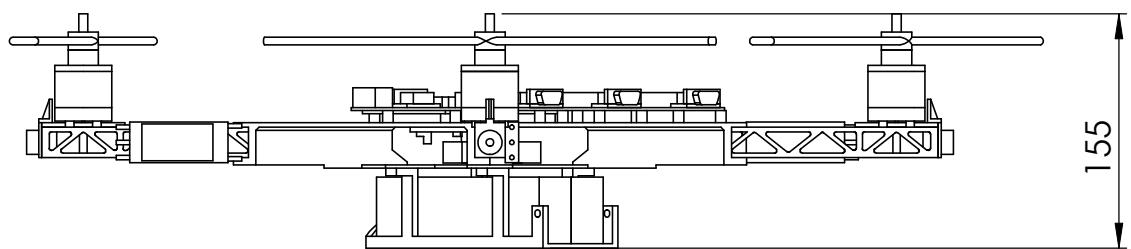
-
- [18] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. In *Proceedings of the International Symposium on Experimental Robotics*, Dec 2010.
 - [19] Warren F. Phillips. *Mechanics of flight*. John Wileys & Sons, inc., 2004.
 - [20] William Premerlani and Paul Bizard. Direction cosine matrix imu: Theory. Technical report, 2009.
 - [21] Edward Ramsden. *Hall-effect sensors: theory and applications*. Elsevier Inc., 2nd. edition, 2006.
 - [22] John McAllister Roger Woods. *FPGA-based implementation of signal processing systems*. John Wiley & Sons, 2008.
 - [23] Roland Siegwart Samir Bouabdallah, André Noth. Pid vs lq control techniques applied to an indoor micro quadrotor. *IEEE*, 2004.
 - [24] A. Smyth and R. Betti. *The 4th International Workshop on Structural Control*. DEStech Public, Inc, 2004.
 - [25] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2003.
 - [26] R. Mahony T. Hamel. Attitude estimation on $\text{so}(3)$ based on direct inertial measurements. *IEEE International Conference on Robotics and Automation Orlando, Florida - May 2006*, 2006.
 - [27] Kjell Eivind Skjønhaug Tor Martin Heggland, Øyvind Magnussen. 6 dof sensor module. Technical report, University of Agder, 2010.
 - [28] Padmaraja Yedamale. Brushless dc (bldc) motor fundamentals. Technical report, Microchip Technology Inc., 2003.
 - [29] Arda Özgür Kivrak. Design of control systems for a quadrotor flight vehicle equipped with inertial sensors. Master's thesis, The Department of Mechatronics Engineering, Attilim University, 2006.

Appendices

A. Parts

Sensors	Where	Part Number
IMU	www.digikey.com	ADIS16400/PCBZ-ND
Current/Voltage	www.sparkfun.com	SEN-09028
Rangefinder - IR	www.sparkfun.com	Sharp GP2Y0A02YK0F
Camera	www.multicom.no	Axis M10xx
GPS	www.sparkfun.com	50 Channel D2523T
Control system		
RIO	www.ni.com	NI sbRIO-9631
Battery	www.elefun.no	Hyperion G3 CX, 450 mAh
Xbee - Tranciever	www.sparkfun.com	WRL-08665
Xbee - socket to USB	www.sparkfun.com	WRL-08687
Xbee - socket to RS232	www.sparkfun.com	WRL-09111
Drive system		
Motor - Hyperion ZS30	www.elefun.no	Hyperion ZS 3020 10-Turn 922Kv
Motor - Backmount	www.elefun.no	Hyperion Z30 Series Backmount - LONG
ESC - YGE60	www.shop.rc-now.com	YGE 60
Battery	www.elefun.no	Flightpower EON-X 5000mAh
Charger	www.elefun.no	not yet known
Cable drive system	www.elefun.no	AM-1303
Propeller - APC 12x6	www.elefun.no	APC 12x6E
Propeller - APC 12x6 Push	www.elefun.no	APC 12x6EP
Miscellaneous		
IMU-connector	www.farnell.no	clm-112-02-L-D
RIO connector female	www.elfa.se	43-150-81
RIO connector male	www.elfa.se	43-150-57
Flatkabel 50pin	www.elfa.se	55-663-51
3.3v regulator	www.sparkfun.com	LD1117V33
Frame system		
Aluminum bar	www.thomic.no	
Aluminum plate - Motor	www.thomic.no	
Paint - Black	www.fargerike.no	

B. Mechanical Drawings; Quadcopter



UNIVERSITY OF AGDER
FACULTY OF ENGINEERING AND SCIENCE

Date: 07.04.2011

Weight: 2892.81 g

Material: NA

Project: Quadcopter

Part: Assembly

Designed by:

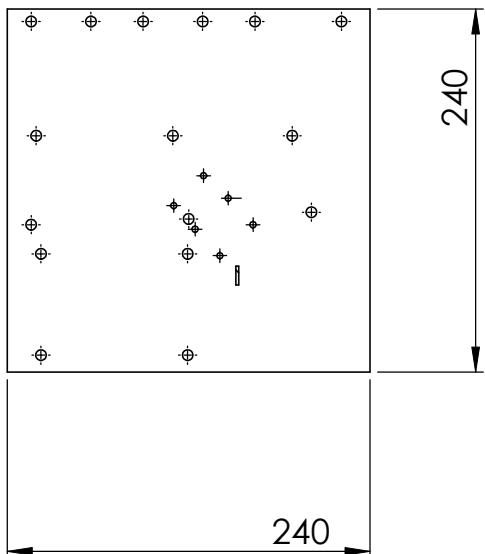
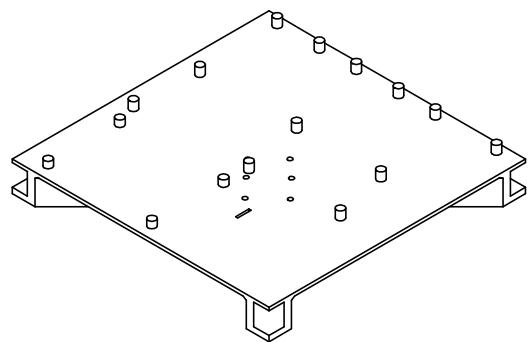
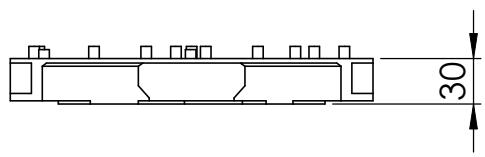
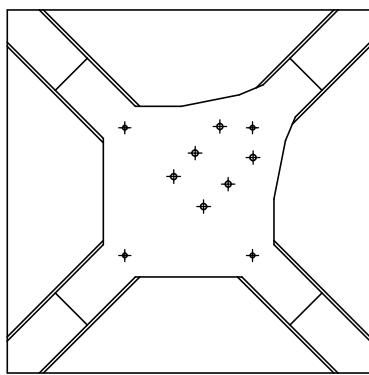
Øyvind Magnussen, Kjell E. Skjønhaug

Class/assigment:

MAS 500; Master Thesis

Scale: 1/5

Sheeet nr: 1/6



Part is to printed by 3d printer, from CAD model



UNIVERSITY OF AGDER
FACULTY OF ENGINEERING AND SCIENCE

Date: 07.04.2011

Weight: 311.00 g

Material: Plastic

Project: Quadcopter

Part: **Revisioned topplate**

Designed by:

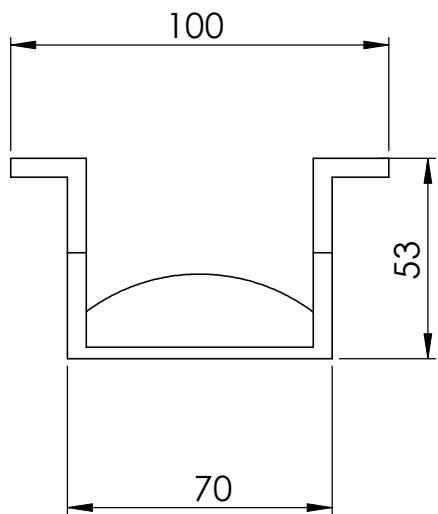
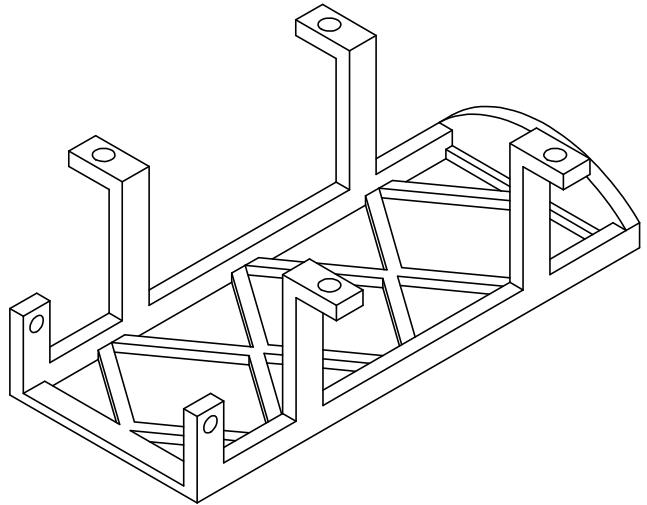
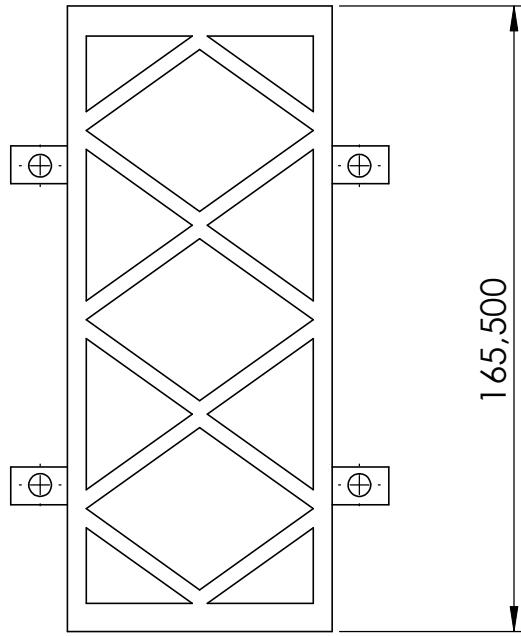
Øyvind Magnussen, Kjell E. Skjønhaug

Class/assigment:

MAS 500; Master Thesis

Scale: 1:5

Sheeet nr: 2/6



Part is to printed by 3d printer, from CAD model



UNIVERSITY OF AGDER
FACULTY OF ENGINEERING AND SCIENCE

Date: 07.04.2011

Weight: 34.93 g

Material: Plastic

Project: Quadcopter

Part: Battery_bracket

Designed by:

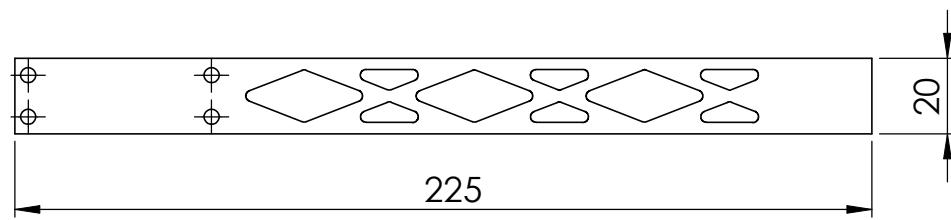
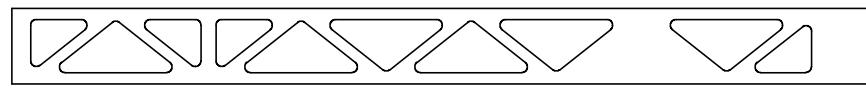
Øyvind Magnussen, Kjell E. Skjønhaug

Class/assigment:

MAS 500; Master Thesis

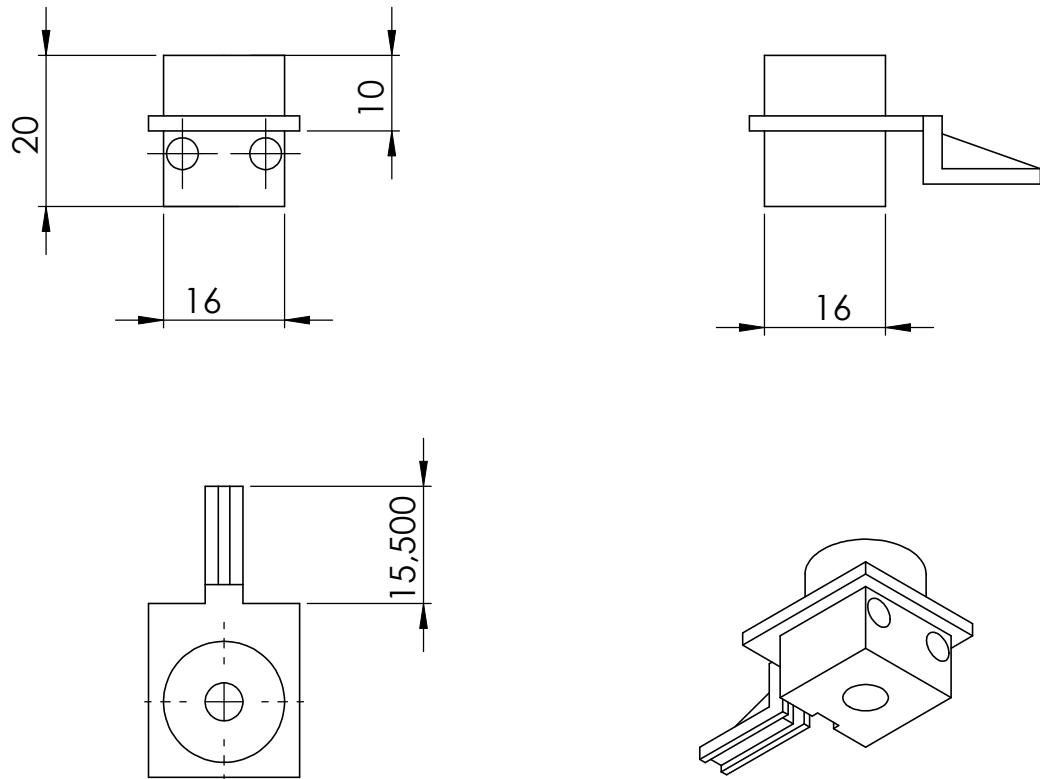
Scale: 1:2

Sheeet nr: 3/6



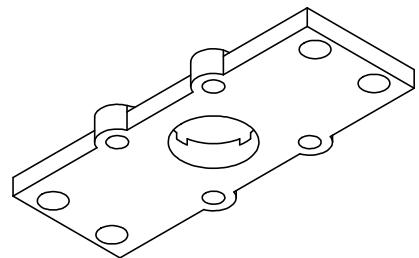
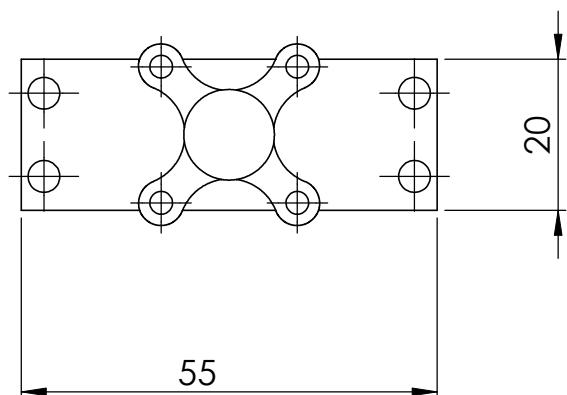
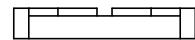
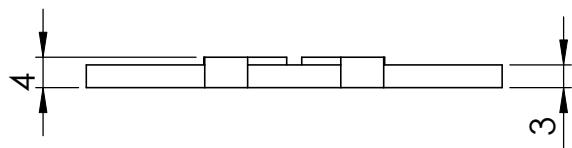
Part is to be milled from CAD model

 UNIVERSITY OF AGDER FACULTY OF ENGINEERING AND SCIENCE		Date: 07.04.2011
		Weight: 20.39 g
		Material: Aluminium
Project: Quadcopter	Part: Arm	
Designed by: Øyvind Magnussen, Kjell E. Skjønhaug	Class/assignment: MAS 500; Master Thesis	Scale: 1:2 Sheet nr: 4/6



Part is to printed by 3d printer, from CAD model

 UNIVERSITY OF AGDER FACULTY OF ENGINEERING AND SCIENCE		Date: 07.04.2011 Weight: 3.75 g Material: Plastic
Project: Quadcopter		Part: End knob
Designed by: Øyvind Magnussen, Kjell E. Skjønhaug	Class/assigment: MAS 500; Master Thesis	Scale: 1:1 Sheeet nr: 5/6



Part is to be milled from CAD model



UNIVERSITY OF AGDER
FACULTY OF ENGINEERING AND SCIENCE

Date: 07.04.2011

Weight: 9.00 g

Material: Aluminium

Project: Quadcopter

Part: Motorbracket

Designed by:

Øyvind Magnussen, Kjell E. Skjønhaug

Class/assigment:

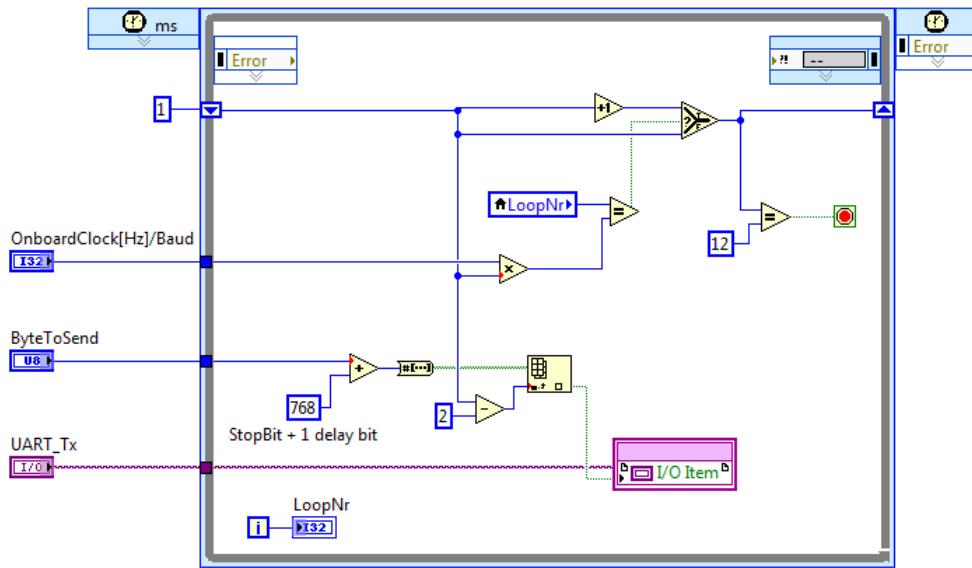
MAS 500; Master Thesis

Scale: 1:1

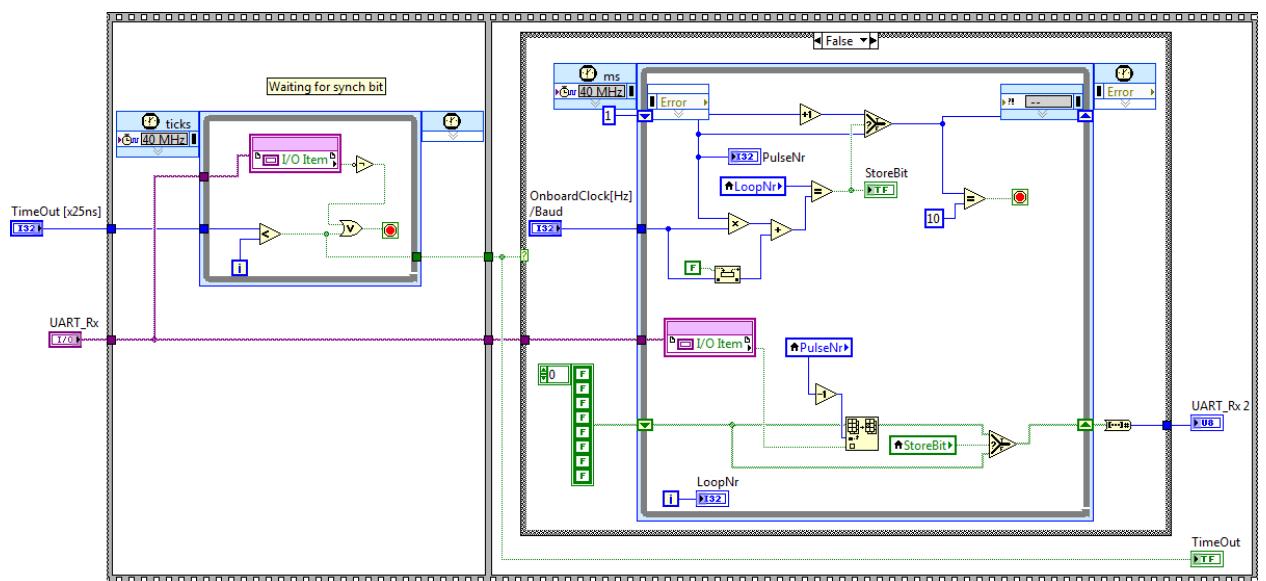
Sheet nr: 6/6

C. Programming

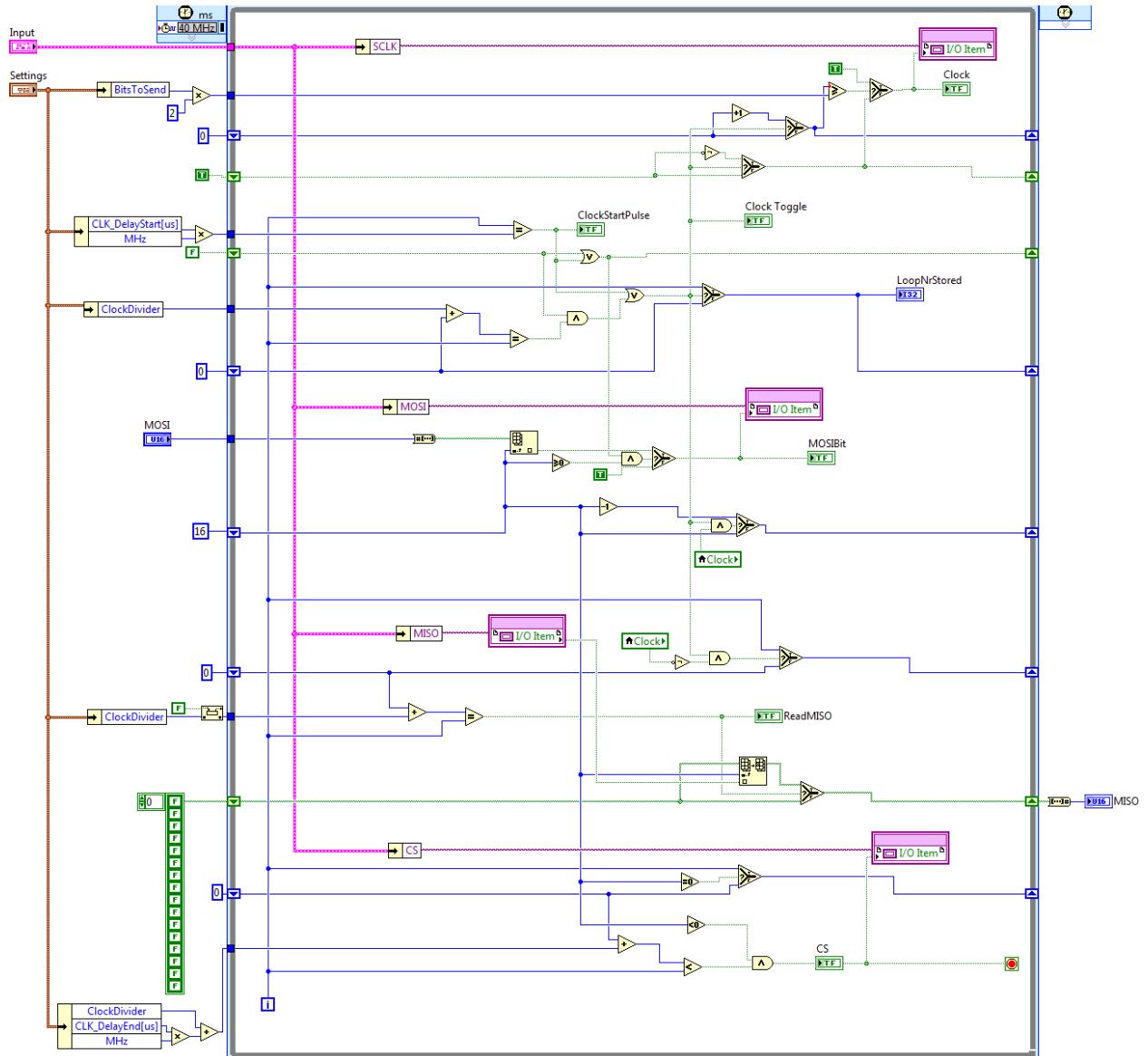
C.1 UART Tx



C.2 UART Rx



C.3 SPI VI

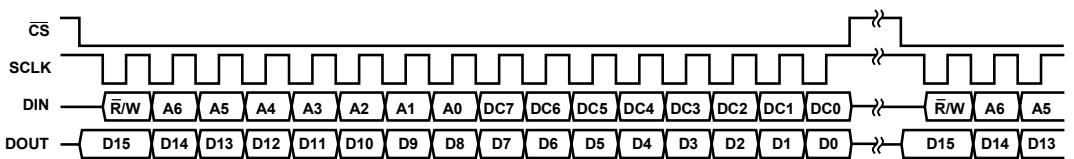


C.4 ADIS16400

MEMORY MAP**Table 8. User Register Memory Map**

Name	R/W	Flash Backup	Address ¹	Default	Function	Bit Assignments
FLASH_CNT	R	Yes	0x00	N/A	Flash memory write count	N/A
SUPPLY_OUT	R	No	0x02	N/A	Power supply measurement	See Table 9
XGYRO_OUT	R	No	0x04	N/A	X-axis gyroscope output	See Table 9
YGYRO_OUT	R	No	0x06	N/A	Y-axis gyroscope output	See Table 9
ZGYRO_OUT	R	No	0x08	N/A	Z-axis gyroscope output	See Table 9
XACCL_OUT	R	No	0x0A	N/A	X-axis accelerometer output	See Table 9
YACCL_OUT	R	No	0x0C	N/A	Y-axis accelerometer output	See Table 9
ZACCL_OUT	R	No	0x0E	N/A	Z-axis accelerometer output	See Table 9
XMAGN_OUT	R	No	0x10	N/A	X-axis magnetometer measurement	See Table 9
YMAGN_OUT	R	No	0x12	N/A	Y-axis magnetometer measurement	See Table 9
ZMAGN_OUT	R	No	0x14	N/A	Z-axis magnetometer measurement	See Table 9
TEMP_OUT	R	No	0x16	N/A	Temperature output	See Table 9
AUX_ADC	R	No	0x18	N/A	Auxiliary ADC measurement	See Table 9
XGYRO_OFF	R/W	Yes	0x1A	0x0000	X-axis gyroscope bias offset factor	See Table 10
YGYRO_OFF	R/W	Yes	0x1C	0x0000	Y-axis gyroscope bias offset factor	See Table 10
ZGYRO_OFF	R/W	Yes	0x1E	0x0000	Z-axis gyroscope bias offset factor	See Table 10
XACCL_OFF	R/W	Yes	0x20	0x0000	X-axis acceleration bias offset factor	See Table 11
YACCL_OFF	R/W	Yes	0x22	0x0000	Y-axis acceleration bias offset factor	See Table 11
ZACCL_OFF	R/W	Yes	0x24	0x0000	Z-axis acceleration bias offset factor	See Table 11
XMAGN_HIF	R/W	Yes	0x26	0x0000	X-axis magnetometer, hard-iron factor	See Table 12
YMAGN_HIF	R/W	Yes	0x28	0x0000	Y-axis magnetometer, hard-iron factor	See Table 12
ZMAGN_HIF	R/W	Yes	0x2A	0x0000	Z-axis magnetometer, hard-iron factor	See Table 12
XMAGN_SIF	R/W	Yes	0x2C	0x0800	X-axis magnetometer, soft-iron factor	See Table 13
YMAGN_SIF	R/W	Yes	0x2E	0x0800	Y-axis magnetometer, soft-iron factor	See Table 13
ZMAGN_SIF	R/W	Yes	0x30	0x0800	Z-axis magnetometer, soft-iron factor	See Table 13
GPIO_CTRL	R/W	No	0x32	0x0000	Auxiliary digital input/output control	See Table 18
MSC_CTRL	R/W	Yes	0x34	0x0006	Miscellaneous control	See Table 19
SMPL_PRD	R/W	Yes	0x36	0x0001	Internal sample period (rate) control	See Table 15
SENS_AVG	R/W	Yes	0x38	0x0402	Dynamic range and digital filter control	See Table 17
SLP_CNT	W	No	0x3A	0x0000	Sleep mode control	See Table 16
DIAG_STAT	R	No	0x3C	0x0000	System status	See Table 23
GLOB_CMD	W	N/A	0x3E	0x0000	System command	See Table 14
ALM_MAG1	R/W	Yes	0x40	0x0000	Alarm 1 amplitude threshold	See Table 25
ALM_MAG2	R/W	Yes	0x42	0x0000	Alarm 2 amplitude threshold	See Table 25
ALM_SMPL1	R/W	Yes	0x44	0x0000	Alarm 1 sample size	See Table 26
ALM_SMPL2	R/W	Yes	0x46	0x0000	Alarm 2 sample size	See Table 26
ALM_CTRL	R/W	Yes	0x48	0x0000	Alarm control	See Table 24
AUX_DAC	R/W	No	0x4A	0x0000	Auxiliary DAC data	See Table 20
PRODUCT_ID			0x4C to 0x55		Reserved	
			0x56	0x4105	Product identifier	

¹ Each register contains two bytes. The address of the lower byte is displayed. The address of the upper byte is equal to the address of the lower byte plus 1.

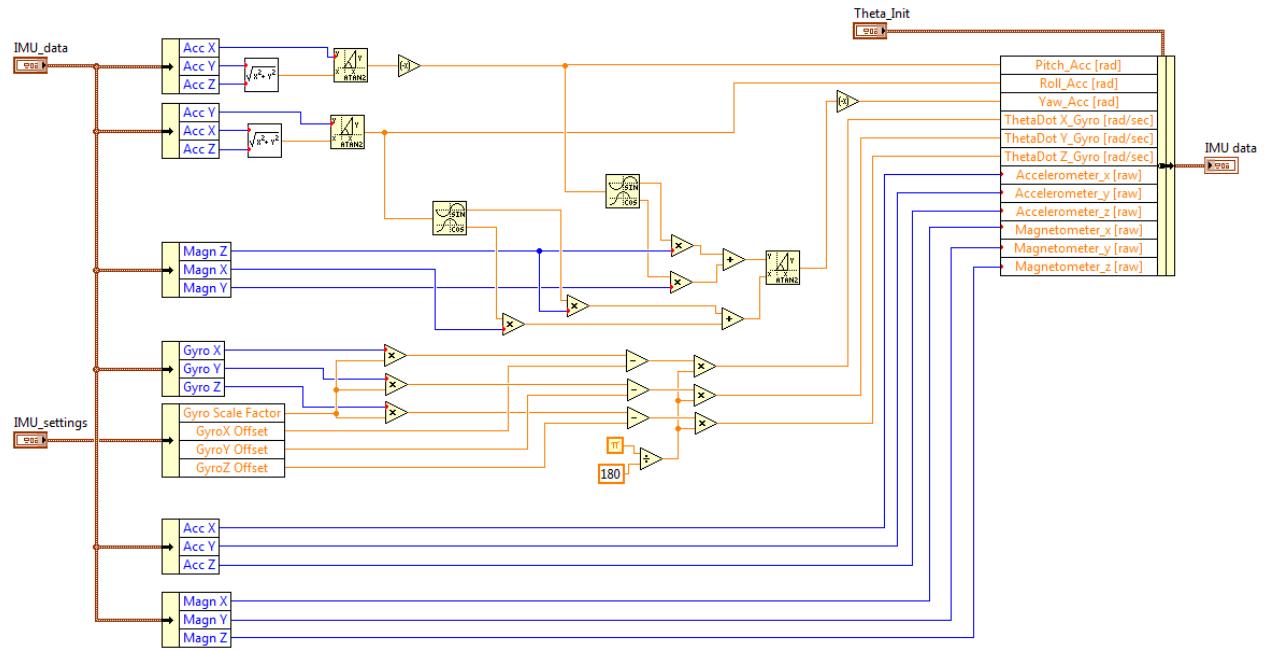
**NOTES**

1. DOUT BITS ARE BASED ON THE PREVIOUS 16-BIT SEQUENCE ($\bar{R} = 0$).

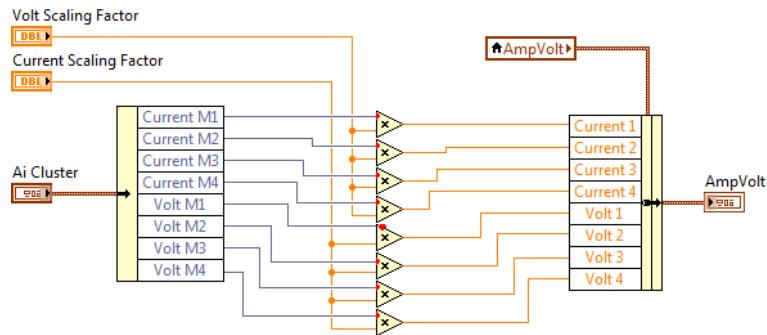
07907-011

Figure 11. Output Register Bit Assignments

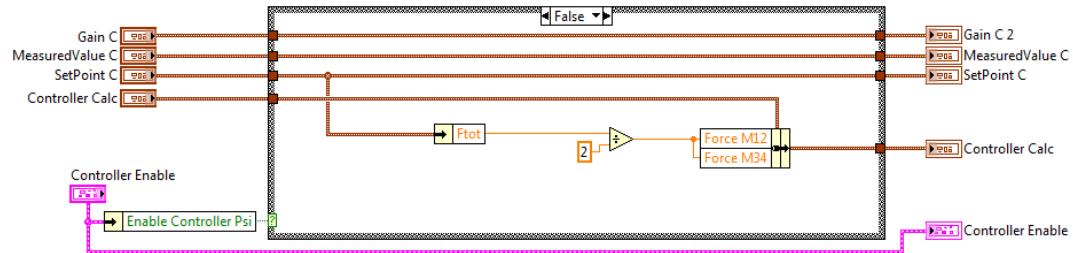
C.5 IMU Scaling



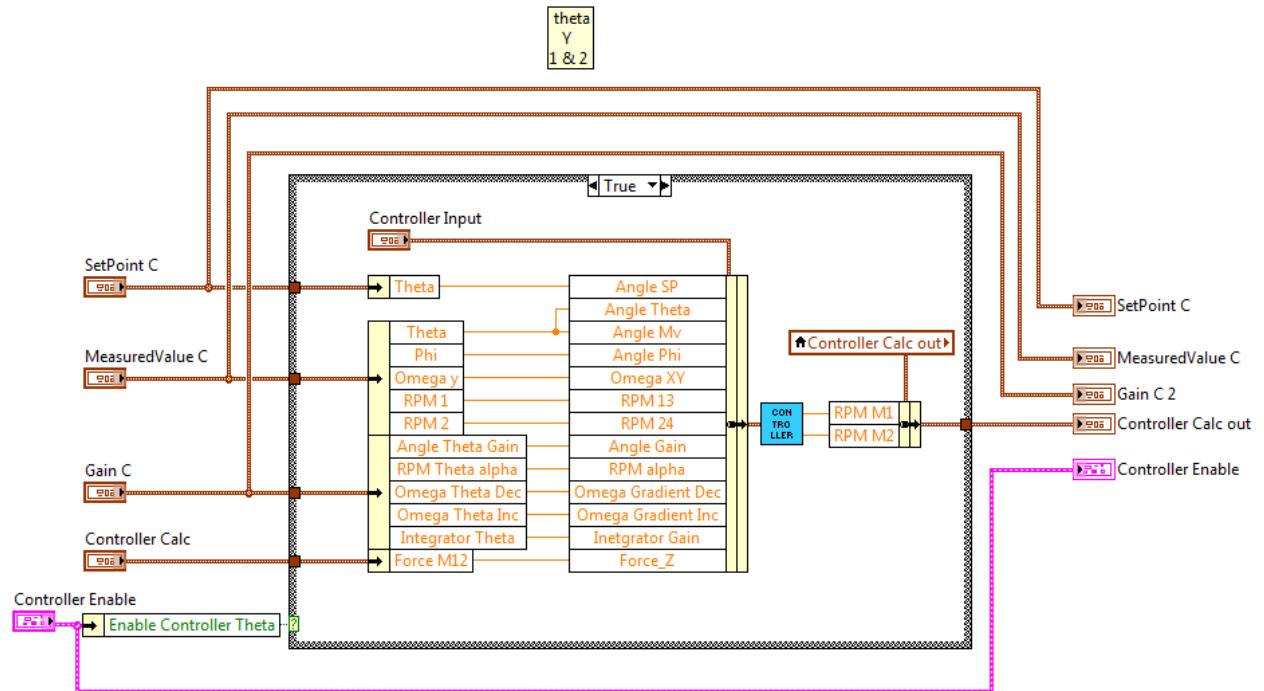
C.6 RPM



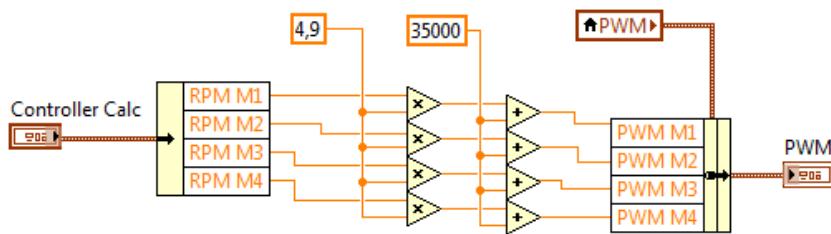
C.7 Yaw controller disabled



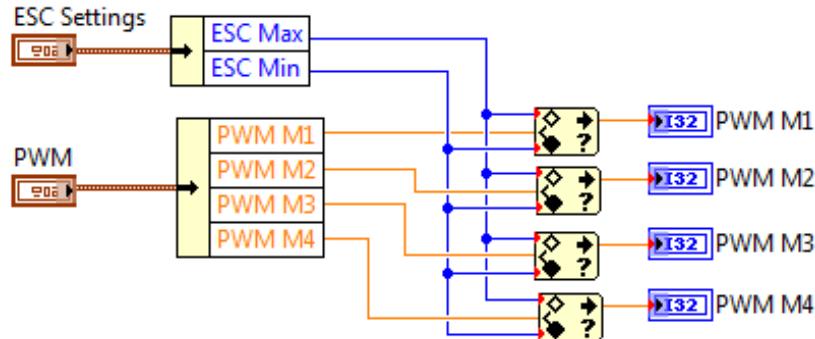
C.8 Theta controller mapping



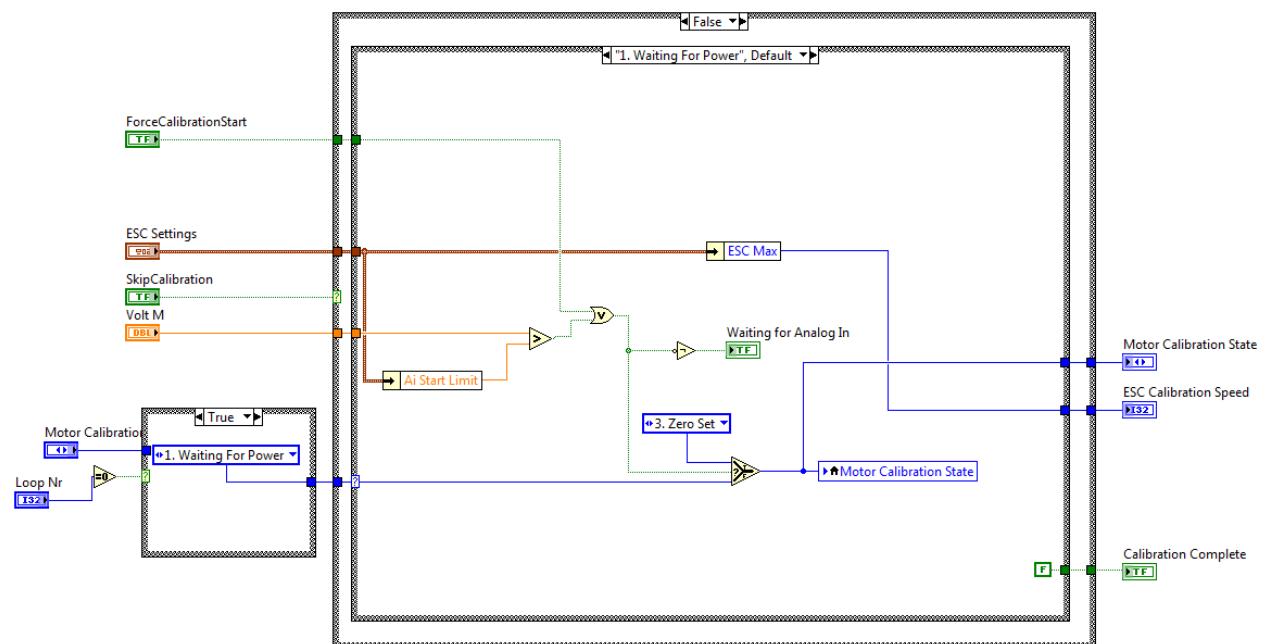
C.9 ESC RPM to PWM

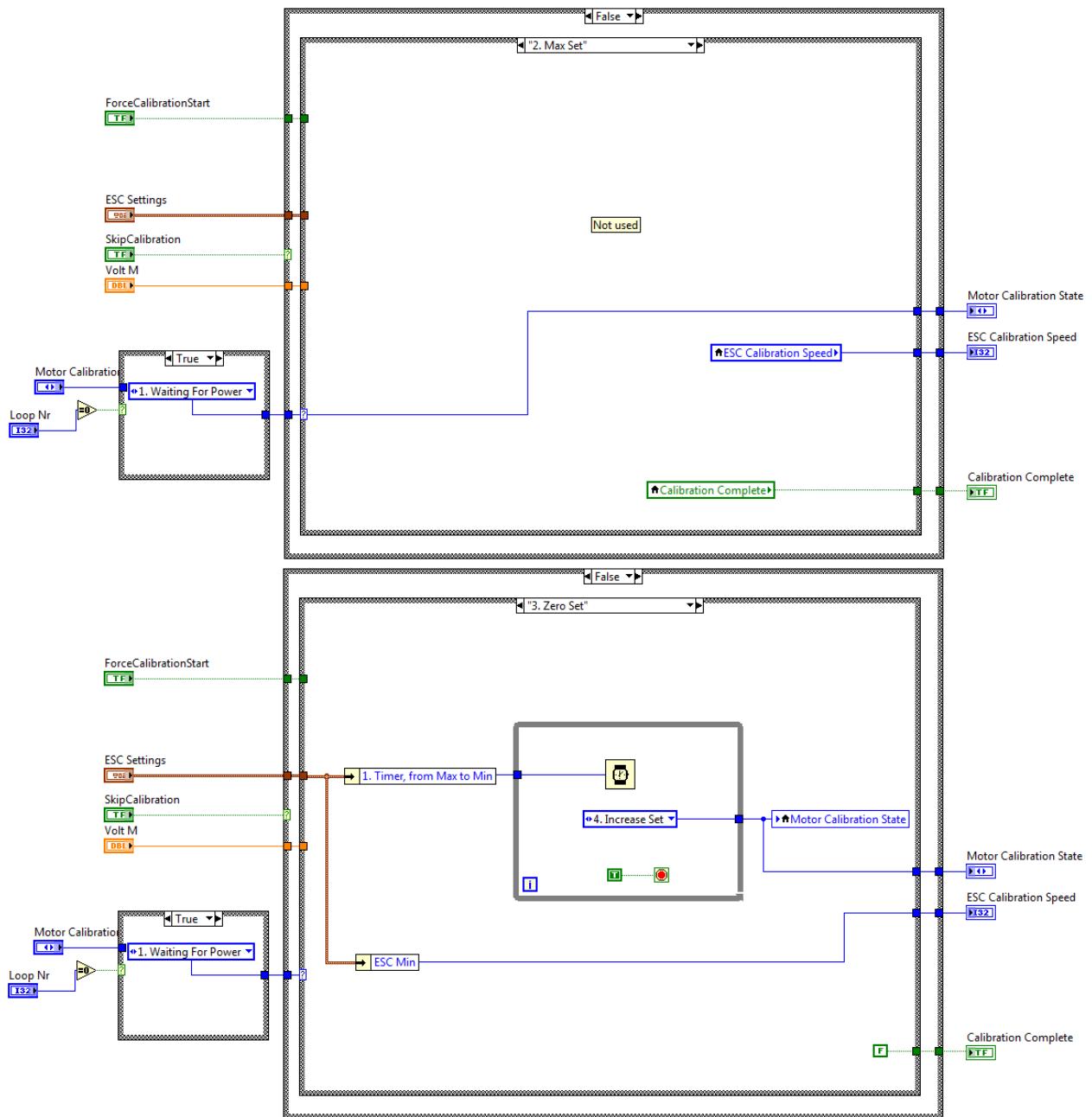


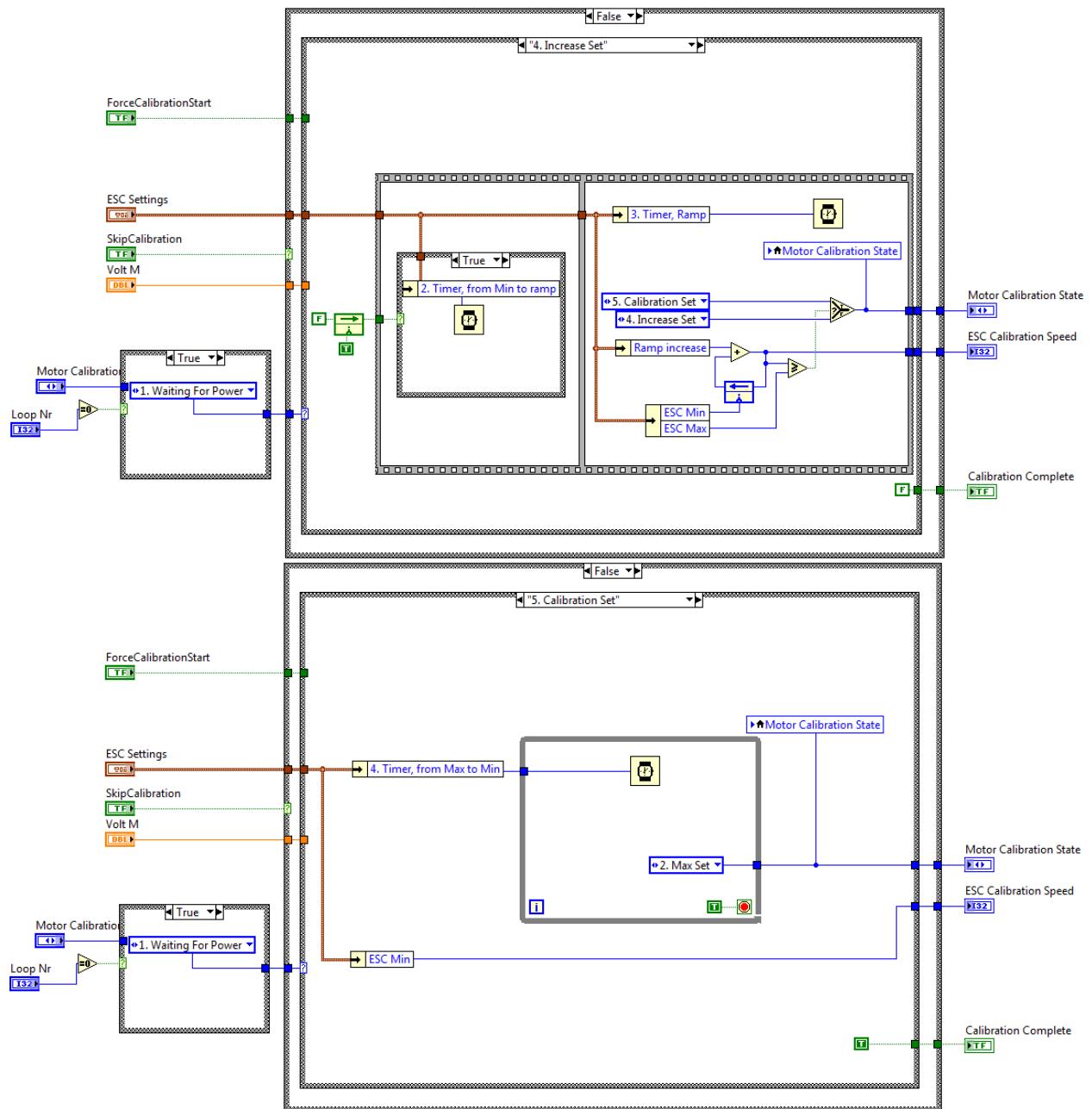
C.10 ESC Saturation



C.11 ESC Calibration







C.12 Motor Test

