NTNU – Trondheim
Norwegian University of
Science and Technology

Drop and Recovery of Sensor Nodes Using UAVs

Vegard Voldsund

June 2014

MASTER THESIS
Department of Engineering Cybernetics
Centre for Autonomous Marine Operations and Systems
Norwegian University of Science and Technology

Supervisor 1: Professor Tor Arne Johansen
Supervisor 2: PhD Candidate Kristian Klausen

Problem Description

Mechanisms for drop and recovery of sensor nodes using UAVs (unmanned aerial vehicles)

Vegard Voldsund
Department of Engineering Cybernetics
NTNU

Project description:
Design and investigate methods for sensor pickup and deployment by multicopters (quad/hex). The following items should be considered:

1. Overall system description with detailed module interaction schemes and protocols.
2. Study and discuss different solution schemes for this kind of mission. Include both simple and complex mechanisms. Discuss necessary assumptions for the design of the sensor node for each proposed solution.
3. Design and create one of the proposed mechanical mechanisms for sensor deployment and pickup to be mounted on a hexacopter.
4. Study how the multicopter should approach the target for pickup/deployment.
5. Study and implement camera-based methods for node tracking needed for pickup.
6. The results should be verified by simulations. Major system components should be tested in experiments.
7. Conclude findings in a report. Include Matlab/C-code as digital appendices.

Supervisor: Tor Arne Johansen
Co-Supervisor: Kristian Klausen

Abstract

Sammendrag

(Norwegian translation of the abstract)

Preface

This project is written as a part of my MSc degree at the Department of Engineering Cybernetics at NTNU and is part of research conducted by the Center of Autonomous Marine Operations and Systems (AMOS). The work in this project will be continued into my master thesis.

The execution of this project would not have been possible without the financial backing from AMOS and the usage of equipment at the electronics and mechanical workshops at the Department of Engineering Cybernetics, NTNU. I will like to thank the guys at these workshops for valuable help and guidance.

I would also like to thank my supervisors Professor Tor Arne Johansen and PhD Candidate Kristian Klausen for guidance and encouragement along the way.

Vegard Voldsund

Acronyms

AMOS Centre for Autonomous Marine Operations and Systems

Contents

Chapter 1

Introduction

1.1 Background and motivation

Unmanned aerial vehicles (UAVs) are aircraft without a human operator on board. They can fly autonomously or be remotely operated by a pilot on the ground. The use of UAVs have exploded in recent years. This is due to a lot of ongoing research and development conducted by companies, scientists and enthusiasts. Some of this research and development have resulted in open source projects that have made advanced UAV technology availably at very low cost.

The UAV used in this project will be based on one of these open source UAV solutions. This solution is chosen because it is cost efficient, results of the project will be easy to use by others, and it is a way to get going fast and still be able to do some customization.

This project is a part of research conducted by the Center of Autonomous Marine Operations and Systems (AMOS) at NTNU. AMOS is now establishing a new laboratory with field experimental capabilities (UAV-Lab). This project is related to this lab. To be a part of the research at AMOS opens up a lot of opportunities, but it also means that some things have to be standardized. For instance the choice of UAV and the use of PandaBoard as onboard computer follows standards decided by AMOS. Use of DUNE as an environment to structure and develop the software running on the onboard computer and the use of NEPTUS as a ground control station is also according to AMOS standards.

The goal of this project is to create a mechanism that can be used for sensor node pickup and deployment by the use of multicopters. The sensor node will be a lightweight packet that can contain different sensors depending on the mission. These sensor nodes will be dropped into the sea where they will float on the surface. Examples of use for the sensor nodes can be to log temperature, currents, salinity or water quality. Hence they can be very useful in for instance climate research or for detecting oil spills.

Both fixed-wing and multicopter UAVs will be part of the sensor node pickup and deploy-

ment system and supplement each other in the UAV-lab. Multicopters will be used in coastal areas and at relatively good weather conditions, while fixed-wing UAVs have a much greater range and will be used for longer missions and in rougher weather conditions.

The project will be continued into a master thesis where the created mechanism will be used for sensor node pickup and deployment. This means that the goal of this project is not to create a fully operational system with seamless integration between the modules, but to make different modules that are ready to be used in the master thesis. This report should be read in this context.

1.2 Previous work

As already mentioned there has been a lot of research on UAVs in recent years, but most of this research have focused on fixed-wing aerial vehicles. And the applications have usually been limited to monitoring and search [**?**].

The latest couple of years have shown some research on rotary-wing aircraft interacting with the environment in different ways. There have been conducted research where the UAV is used to manipulate its environment. Two examples of this using two different strategies are [**?**] and [**?**]. Lippiello and Ruggiero are inspired by the use of impedance control in robot manipulation tasks, while Jiang and Voyles alters a hexacopter by tilting the motors, making it possible to create a horizontal force without tilting the hexacopter.

The most relevant applications to highlight in this report are those that use different strategies for drop off and pickup of objects. Possibilities of using an avian catching fish as an inspiration for a quadcopter based gripper system to be able to execute high speed pickup have been explored by [**?**]. A gripper on the end of a link with two joints is used to replicate an avians leg and claw. Good results were achieved as the quadcopter was able to grasp targets at speeds up to 3 m/s. The trajectory of the pickup was decided pre run time to match the trajectory of the avian. Feedback to the controller was given using VICON[1].

The use of a helicopter to grasp objects on the ground are explored in [**?**]. The ability to grasp objects of different shapes and without a perfect lined up position is in focus for the gripper design. Tests demonstrated an ability to grasp objects of different shapes and sizes under human control of the helicopter. The most difficult objects were grasped 67 % of the time, while the easiest object was grasped 100 % of the time.

Estimation of payload parameters are explored in [**?**] as well as mechanical design and controller for aerial grasping. Relevant parameters to estimate are mass and inertia. These estimates can be used to adapt the controller and to check if the object was successfully picked up or not.

[1] A very accurate external motion capture system

1.3 Contribution and scope of this report

Some of the research presented in the previous section picks up objects lying on the ground (in some of them at exactly known positions). The fact that the aim in the master project is to pick up sensor nodes that are floating affected by waves, wind and currents add another dimension to the challenge. Navigation at sea means that one can not rely on external sensor systems like for instance VICO. This means that accurate positioning above the sensor node will be a challenge. The UAV will have to rely on poor position measurements and some added sensor system, for instance camera, to close the loop between the UAV and the sensor node.

This project aims to connect known techniques and theory from different fields in new ways to make pickup and deployment of sensor nodes by the use of UAVs at sea possible.

1.4 Organization of this report

The UAV used in this project is described in Chapter 2. This description includes the most relevant features to give the reader insight into which possibilities that lie within the UAV. The PandaBoard that is used as onboard computer is also described for the same reasons.

Different possible design solutions are described in Chapter 3. The chapter starts with some design choices that are common for all the solutions and continues with specifics about the different proposals. Both advantages and limitations of the different solutions are considered before the chapter ends in a conclusion of which design solution to implement.

An introduction to the relevant reference frames used for navigation, control and object tracking, as well as some features of the open source computer vision library OpenCV are given in Chapter 4.

Chapter 5 presents the implementation of the chosen design solution. It covers mechanical design, node tracking using camera, communication between the different modules, some circuit design and power supply.

The tests of the design solution with associated results are described in Chapter 6. Some of the key features that are tested are weight limitations on the sensor node, accuracy of the node tracking and robustness of the mechanical design.

The implemented design solution is discussed in Chapter 8, before conclusions are drawn in Chapter 9.

The digital appendix included with this report contains source code used for the experiments and the node tracking algorithm. Solidworks models of the 3D-printed parts are included as a resource if the reader wants to replicate the design solution.

Chapter 2

Background Theory

2.1 Reference Frames

The main reference frames that are relevant for navigation and control are ECEF, NED and BODY. A brief introduction to these reference frames based on [Fossen, 2011] and [Vik, 2012] follows. Denavit-Hartenberg convention (DH convention) is a useful tool for defining reference frames. A brief introduction to DH convention based on [Spong et al., 2005] follows.

ECEF

The Earth-centered Earth-fixed (ECEF) reference frame has its origin fixed to the center of the earth while rotating with the earth. The x-axis is defined to point at the intersection between the 0° longitude and the 0° latitude. The z-axis points along the earths rotation axis and the y-axis complete the right handed orthogonal coordinate system.

The position in the ECEF frame can be expressed both with Cartesian coordinates (x_e, y_e, z_e) and with ellipsoidal coordinates (longitude (l), latitude (μ), height (h)). The transformation from Cartesian ECEF-coordinates to ellipsoidal ECEF-coordinates is given by

$$\begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} = \begin{bmatrix} (N+h)\cos\mu\cos l \\ (N+h)\cos\mu\sin l \\ (\frac{r_p^2}{r_e^2}N+h)\sin\mu \end{bmatrix} \tag{2.1}$$

where $r_e = 6378137$ m is the equatorial radius of ellipsoid and $r_p = 6356752$ m is the polar axis radius of the ellipsoid as defined in WGS-84. The parameter N is the radius of curvature in prime vectorial obtained from [Vik, 2012].

$$N = \frac{r_e^2}{\sqrt{r_e^2\cos^2\mu + r_p^2\sin^2\mu}} \tag{2.2}$$

The transformation from Cartesian coordinates to ellipsoid coordinates is a bit more complicated. Longitude is calculated straight forward as

$$l = \tan^{-1}(\frac{y_e}{x_e}) \tag{2.3}$$

but the calculations of latitude and height are implicit equations

$$\tan(\mu) = \frac{z}{p}(1 - e^2\frac{N}{N+h})^{-1} \tag{2.4}$$

$$h = \frac{p}{\cos(\mu)} - N \tag{2.5}$$

where e is the eccentricity of the Earth given by

$$e = \sqrt{1 - (\frac{r_p}{r_e})^2} \tag{2.6}$$

There are several algorithms that can be used to solve these implicit equations, see for instance Algorithm 2.4 in [Fossen, 2011].

NED

The north-east-down (NED) reference frame is moving with the body. The x-axis is always pointing north, the y-axis is pointing east and the z-axis i pointing down normal to the Earth surface.

BODY

The BODY reference frame is body fixed and rotates and moves with the body. It is usually defined with the x-axis pointing along the longitudinal axis, the y-axis pointing along the transversal axis and the z-axis pointing along the normal axis of the body.

Transformation Between ECEF and NED

A vector defined in NED can be transformed to the ECEF frame by the use of a rotation matrix that defines the relationship between NED and ECEF reference frames.

$$\boldsymbol{p}^e = \boldsymbol{R}_n^e(\boldsymbol{\Theta}_{en})\boldsymbol{p}^n \tag{2.7}$$

$$\boldsymbol{\Theta}_{en} = \begin{bmatrix} l & \mu & h \end{bmatrix}^T \tag{2.8}$$

Where $\boldsymbol{R}_n^e(\boldsymbol{\Theta}_{en})$ is found by first performing a rotation l about the z-axis and then a rotation $(-\mu - \frac{\pi}{2})$ about the y-axis. This gives

$$\boldsymbol{R}_n^e(\boldsymbol{\Theta}_{en}) = \begin{bmatrix} -\cos(l)\sin(\mu) & -\sin(-l) & -\cos(l)\cos(\mu) \\ -\sin(l)\sin(\mu) & \cos(l) & -\sin(l)\cos(\mu) \\ \cos(\mu) & 0 & -\sin(\mu) \end{bmatrix} \tag{2.9}$$

Transformation Between NED and BODY

A vector defined in BODY can be transformed to the NED frame by the use of the Euler angle rotation matrix.

$$\boldsymbol{p}^n = \boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb})\boldsymbol{p}^b \tag{2.10}$$

Where $\boldsymbol{\Theta}_{nb} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^T$ are the Euler angles roll, pitch and yaw.

$$\boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi c_\phi s_\theta \\ s_\psi c_\theta & c_\psi c_\phi + s_\phi s_\theta s_\psi & -c_\psi s_\phi + s_\theta s_\psi c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{2.11}$$

The notation s and c with an angle as subscript is used to represent $\sin(angle)$ and $cos(angle)$ respectively. This notation is used throughout this report.

A BODY fixed angular velocity vector $\boldsymbol{\omega}_{b/n}^b = [p, q, r]^T$ and the Euler rate vector are related through a transformation matrix according to equation (2.13).

$$\dot{\boldsymbol{\Theta}}_{nb} = \boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb})\boldsymbol{\omega}_{b/n}^b \tag{2.12}$$

$$\boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi/c_\theta & c_\phi/c_\theta \end{bmatrix} \tag{2.13}$$

Denavit-Hartenberg Convention

The DH convention is a systematic procedure for relating orientation and position of different reference frames, and a tool to select frames. The homogeneous transformation A_i (the transformation matrix from reference system $i-1$ to reference system i) is represented as a product of four basic transformations

$$\begin{aligned} \boldsymbol{A}_i &= \boldsymbol{Rot}_{z,\theta_i}\boldsymbol{Trans}_{z,d_i}\boldsymbol{Trans}_{x,a_i}\boldsymbol{Rot}_{x,\alpha-i} \\ &= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.14}$$

where θ_i is rotation around the z-axis, d_i is transversal movement along the z-axis, a_i is the transversal movement along the new x-axis direction and α_i is the rotation around the new x-axis direction. There exists unique values of θ_i, d_i, a_i and α_i to make equation (2.14) valid if the frames have the features:

- The axis x_i is perpendicular to the axes z_{i-1}
- The axis x_i intersects the axis z_{i-1}

The transformation matrix T_j^i expresses the position and orientation of frame $o_j x_j y_j z_j$ with respect to frame $o_i x_i y_i z_i$ and is calculated as

$$\boldsymbol{T}_j^i = \begin{cases} \boldsymbol{A}_{i+1}\boldsymbol{A}_{i+2}...\boldsymbol{A}_{j-1}\boldsymbol{A}_j & \text{if } i < j \\ I & \text{if } i = j \\ (\boldsymbol{T}_i^j)^{-1} & \text{if } j > i \end{cases} \tag{2.15}$$

$\boldsymbol{T}_j^i$ for $i < j$ contain a rotation matrix $\boldsymbol{R}_i^j$ from frame i to j and the position $\boldsymbol{o}_j^i$ of the origin of frame j with respect to frame i.

$$\boldsymbol{T}_j^i = \begin{bmatrix} \boldsymbol{R}_j^i & \boldsymbol{o}_j^i \\ 0 & 1 \end{bmatrix} \tag{2.16}$$

2.2 Rigid-Body Kinetics

The motion of rigid bodies can be expressed according to [Fossen, 2011] as shown in equation 2.17.

$$\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}_{RB} \tag{2.17}$$

Where $\boldsymbol{M}_{RB}$ is the rigid-body mass matrix, $\boldsymbol{C}_{RB}$ is the rigid-body Coriolis and centripetal matrix due to rotation about the inertial frame. $\boldsymbol{\nu} = [u, v, w, p, q, r]^T$ is the velocity vector expressed in the BODY-frame, and $\boldsymbol{\tau}_{RB} = [X, Y, Z, K, M, N]^T$ is the forces and moments acting upon the body, also expressed in the BODY-frame.

Using Newton-Euler equations and assuming center of origin and center of gravity in the same place, [Fossen, 2011] derives the rigid-body mass matrix and the rigid-body Coriolis and centripetal matrix as shown in equation 2.18.

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_g \end{bmatrix} \quad \boldsymbol{C}_{RB} = \begin{bmatrix} m\boldsymbol{S}(\boldsymbol{\omega}) & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & -\boldsymbol{S}(\boldsymbol{I}_g\boldsymbol{\omega}) \end{bmatrix} \tag{2.18}$$

Here $\boldsymbol{I}_g$ is the inertia matrix, m is the mass, $\boldsymbol{\omega} = [p, q, r]^T$ is the rotational velocity and $\boldsymbol{S}()$ is the skew symmetric matrix operator.

2.3 Computer Vision

2.3.1 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library that has more than 2500 optimized algorithms [**?**]. A few of these algorithms will be briefly explained here. The ones explained are the most relevant for object recognition.

Color Recognition

A digital picture is essentially a matrix of values describing the picture. This matrix is dependent of the color space the picture is defined in. A picture captured from for instance a web-camera is defined in the BGR (Blue-Green-Red) color space, which means that the colors in the picture are defined by combinations of these colors. The color is defined by the relationship between these values, while the brightness is defined by how high these values are. Hence it could be difficult to select threshold values if one is looking for an object of a specific color, because different light conditions would give very different values for the color. To make the thresholding more intuitive one can transform the picture into the HSV (Hue-Saturation-Value) color space. The hue value is unique for a specific color and describes the base color. The saturation describes the strength of the color and the value is a measure of brightness. This makes it simpler to find intuitive thresholds for specific object colors that can handle different lightning conditions. An visualization of the HSV-color space is found in Figure 2.1.

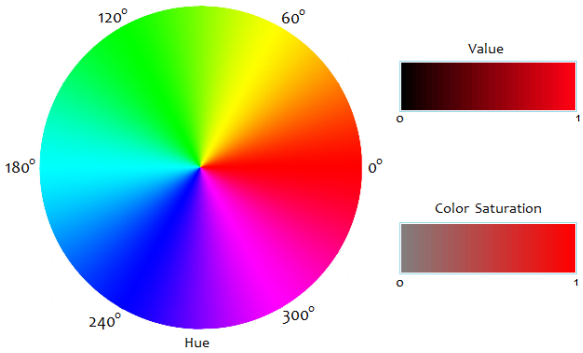


Figure 2.1: HSV-color wheel *Courtesy of had2know.com*

The HSV-picture is thresholded with the desired intervals of the HSV-values. This results in a binary image where the pixel values are one if the color is found and zero if the color is different than the desired color. With good choices of colors for the object and the thresholding values, one would get a good understanding of where the object is.

Canny Edge Detector and Moments

The Canny Edge Detector uses an algorithm presented in [Canny, 1986] that detects edges. It can be used to detect contours. If the Canny Edge Detector is used on a binary image like the one described in the previous section it would find the contour surrounding the area of the detected object.

Then this contour could be fed to the moment function in OpenCV which calculates the center of moment for the contour, which is the center of the outline of the object.

Cascade Classifier

The use of cascade classifiers includes two major stages, training and detection [OpenCV.org, 2013]. Training is executed once only, while detection is executed run time. Training takes two different sets of samples, positive and negative samples. The positive samples are samples containing the object, while the negative samples are samples without the object. These samples are run through a cascade classifier to create a "rule" of what to look for in the detection phase.

There exists several different cascade classifiers, two of these are implemented in OpenCV. These are Haar- and LBP-classifiers. They use different features and have different runtime. Details on these algorithm are outside the scope of this text.

Accuracy of the use of cascade classifiers is dependent on the object to be detected, and the number of samples used for training. For instance one positive sample can be sufficient for detection of a rigid object, while detection of for instance faces will need hundreds or even thousands of positive samples.

There are tools in OpenCV that take the positive samples and creates many new positive samples of it. This is done by randomly rotating the object around all three axes, changing the objects intensity and placing it on random backgrounds [OpenCV.org, 2013].

SURF

Speeded Up Robust Features (SURF) is a further development and speeded up algorithm on the basis of SIFT (Scale-Invariant Feature Transform) [Mordvintsev and Abid, 2013]. These algorithms use a single picture of the object for the recognition. Based on different techniques beyond the scope of this text, key points are found describing the object. The key points is assigned an orientation to achieve invariance to image rotation, and the way the key point is selected make the algorithm scale invariant [OpenCV, 2013].

For commercial use one should note that both SURF and SIFT is patented and part of a non-free module in OpenCV.

2.4 Haversine Formula

Chapter 3

Description of the UAV

The UAV used in this project as a base for the pickup and deployment mechanism is an ArduCopter Hexacopter (see Figure 3.1). The ArduCopter uses the ArduPilot Mega 2.6 flight control unit (hereinafter referred to as APM). The UAV will also be equipped with a PandaBoard as an onboard computer.

Relevant features of the APM and the PandaBoard are described below.

Figure 3.1: ArduCopter Hexacopter *Courtesy arducopter.co.uk*

3.1 APM

The APM is an open source flight control unit supporting multicopters, traditional helicopters, fixed wing aircraft and rovers [ArduPilot, 2013]. Software is developed and supported

by the DIYDrones community. At the moment the community has more than 45 000 members (November 2013) and an active forum where one could get help and useful tips and tricks.

3.1.1 Modes of Operation

The APM can operate in many different modes of operation [Ardupilot, 2013]. The most relevant for this project are:

- Stabilize - Manual flight mode that automaticly levels the UAV and maintains the current heading
- Auto - The UAV tracks predefined waypoints
- Guided - The next waypoint is defined in flight
- RTL (Return to Launch) - The UAV returns to the position where it was armed and hovers
- LAND - The UAV lands, shut-down the motors and disarms

Control signals in the different modes are given either with PWM-signals[1] or through serial communication using the MAVLink protocol (the MAVLink protocol will be briefly explained below). The PWM-signals are usually sent from a 2.4 GHz radio via a receiver on the UAV, while the serial communication is usually sent from a ground station via a telemetry link to the APM. These signals could easily be replicated by the PandaBoard. A picture of the APM with a voltage regulator ann an external magnetometer and GPS module is found in Figure 3.2.

Figure 3.2: APM 2.6 with a voltage regulator and an external magnetometer and GPS module *Courtesy diydrones.com*

[1]Pulse Width Modulated signals

3.1.2 Sensors

The APM is equipped with several sensors that are utilized for navigation and control. These will be briefly explained below.

Barometer

A barometer is an instrument that is used to measure air pressure [Merriam Webster, 2013]. The barometric formula

$$p(h) = p(0)e^{-\frac{mgh}{kT}} \tag{3.1}$$

relates the pressure $p(h)$ of an isothermal, ideal gas of molecular mass m at some height h to its pressure $p(0)$ at height $h = 0$, where g is the acceleration of gravity, k the Boltzmann constant, and T the temperature. This formula applies reasonably well to the lower troposphere. For altitudes up to 6 km the error is less than 5 % [Berberan-Santos et al., 1997].

The barometer in the APM is based on piezoresistive technology. Piezoresisivity is a common sensing principle for micro machined sensor [Liu, 2011] that uses the fact that resistivity of some materials changes with applied stress [Mason and Thurston, 1957]. This feature is used in the barometer, when the air pressure varies, the pressure on the material in the barometer varies which means that resistivity varies. A mapping from resistivity to pressure is used to calculate altitude referenced to start altitude. Altitude calculations by the use of barometers can be sensitive to changing weather conditions.

The barometer in the APM is the MS5611-01BA03 by Measurement Specialties, which according to the producer has a resolution of 10 cm.

Magnetometer

A magnetometer is an instrument for measurement of magnetic fields. Depending on the setup they can measure strength of a magnetic field or both strength and direction of the field [Store Norske Leksikon, 2013]. The magnetometer in the APM is a three-axes magnetometer. This means that both the strength and direction of the magnetic field can be measured. The magnetometer measures the force created by the magnetic field on an energized conductor. This force is called the Lorentz Force and follows the formula

$$\boldsymbol{F} = q\boldsymbol{v} \times \boldsymbol{B} \tag{3.2}$$

where q is charge, $\boldsymbol{F}$ is the Lorentz Force and $\boldsymbol{B}$ is the magnetic field. The charge q is assumed to be known and $\boldsymbol{F}$ can be measured using piezoresistive principles. Then the magnetic field is easily found using the formula in equation (3.2). This field is pointing towards north (excluding disturbances from for instance the motors on the UAV), which will be utilized in the APMs IMU (described in the next section) to get more accurate attitude measurements.

The magnetometer in the APM is a HMC5883L from Honeywell.

Inertial Measurement Unit

The APM contains an Inertial Measurement Unit (IMU). An IMU consists of an ISA (Inertial Sensor Assembly), hardware and low level software. The ISA is a cluster of three gyroscopes and three accelerometers that measure angular velocity and acceleration respectively [Vik, 2012]. The IMU can also use magnetometer measurements. In the APM the magnetometer is not a part of the IMU, but it has an interface where it communicates with the magnetometer to make it possible to utilize the magnetometer measurements in the calculations of the attitude of the UAV.

Conceptually the accelerometer measures the movement of a damped mass hanging in a spring. To transform this movement into an electric signal, piezoresistive principles are utilized. In this case is it the acceleration that creates deformation in the piezoresistive material.

The gyroscopes are also based on MEMS technology. MEMS-gyroscopes are usually implemented with a tuning fork configuration. Two masses oscillate in opposite directions of each other. When these masses experiences angular velocity the Coriolis force act in opposite directions on the masses. This results in a measurable capacitance change which is proportional to the angular velocity of the UAV [Jay Esfandyari, 2010].

The IMU in the APM is a MPU-6000 from Inven Sense.

GPS

The GPS module that is connected to the APM contains an ublox LEA-6H module [3DRobotics, 2013]. It uses Navstar GPS but can also support GLONASS and Galileo. Communication to the APM is done via UART[2] with an update frequency of 5 Hz. Position accuracy is given by the datasheet to be 2.5 m CEP[3][u blox, 2012]. GPS can also be used for altitude measurements, but the nature of the GPS is that altitude measurements will have even less accuracy than position measurements.

3.1.3 Interfaces

The APM has several interfaces that make it flexible and suitable for research and development. It has dedicated connection points for GPS and telemetry. These are interfaced using UART. It also has an unused UART-port available for other units and applications. Input from the radio and output for the motor controllers have dedicated ports that operates with PWM-signals. A connection point to access the APMs I^2C bus is also available. Several units can communicate using this bus. It has also a lot of unused I/O-pins available for further development. These can for instance be used to connect an Ultrasonic Range Finder (there are several of them that are supported by the APM).

[2]Universal Asynchronous Receiver/Transmitter

[3]CEP (Circular Error Probability) defines the radius of a circle centered in the true position containing 50 % of the GPS measurements [iGage, 2013]

MAVLink

The APM communicates with its surroundings using UART with a subset of the communication protocol MAVLink[4]. MAVLink is a lightweight, header-only marshalling library for micro air vehicles [QGroundControl, 2013]. MAVLink has a lot of predefined messages in addition to the possibility of creating custom messages. An XML-file contains the definition of the different message types. An example of one of the message definitions is shown in Figure 3.3.

```
<message id="24" name="GPS_RAW_INT">
        <description>The global position, as returned by the Global Positioning System (GPS). This is NOT the global
        <field type="uint64_t" name="time_usec">Timestamp (microseconds since UNIX epoch or microseconds since system
        <field type="uint8_t" name="fix_type">0-1: no fix, 2: 2D fix, 3: 3D fix. Some applications will not use the v
        <field type="int32_t" name="lat">Latitude (WGS84), in degrees * 1E7</field>
        <field type="int32_t" name="lon">Longitude (WGS84), in degrees * 1E7</field>
        <field type="int32_t" name="alt">Altitude (WGS84), in meters * 1000 (positive for up)</field>
        <field type="uint16_t" name="eph">GPS HDOP horizontal dilution of position in cm (m*100). If unknown, set to:
        <field type="uint16_t" name="epv">GPS VDOP horizontal dilution of position in cm (m*100). If unknown, set to:
        <field type="uint16_t" name="vel">GPS ground speed (m/s * 100). If unknown, set to: UINT16_MAX</field>
        <field type="uint16_t" name="cog">Course over ground (NOT heading, but direction of movement) in degrees * 10
        <field type="uint8_t" name="satellites_visible">Number of satellites visible. If unknown, set to 255</field>
</message>
```

Figure 3.3: Message definition of message with ID 24 *Courtesy of wikipedia.org*

The MAVLink protocol can be used both to get status information from the APM and to give commands to the APM.

3.2 PandaBoard ES

The version of the PandaBoard used in this project is the PandaBoard ES Revision B2 (hereinafter referred to as PandaBoard). The PandaBoard is a small but powerful computer based on an OMAP™ 4 Processor. The OMAP™ 4 Processor is designed for high performance applications within a low power envelope [Texas Instruments, 2013] and contains a Dual-core ARM®1.2 GHz CPU. The PandaBoard do also have 1 GB RAM and a port to insert a SD-card for additional memory [pandaboard.org, 2013].

3.2.1 Interfaces

The PandaBoard has several interfaces that make it a good platform for development. It has two expansion connectors with 28 pins each. The functions of these pins includes general purpose I/O, SPI, I^2C, USB, UART, audio, power and support for additional memory [pandaboard.org, 2011]. This means that the the PandaBoard is able of communicating with its surroundings using most of the most popular bus standards.

The PandaBoard does also include a camera header available for development of camera solutions. e-con Systems has developed a camera (e-CAM51_44x) especially for the PandaBoard and this header. The camera is a 5 MP auto focus camera able to provide 720p HD

[4]Micro Air Vehicle Communication Protocol

video streaming at 60 fps [e-con Systems, 2013b]. Communication with the camera uses the MIPI[5] CSI-2 standard. CSI-2 is a standard that provides a robust, scalable, low power and high speed interface for imaging solutions [MIPI, 2013].

3.3 Software

The software developed in this thesis is build upon and integrated into DUNE, IMC and NEPTUS. These software solutions are developed by the Underwater Systems and Technology Laboratory (LSTS) research team at the University of Porto. The LSTS research team is specialized in development and operation of unmanned vehicles and tools for deployment of vehicles in networked systems [LSTS, 2014a].

3.3.1 IMC

The Inter-Module Communication (IMC) protocol is both used for communication between the different nodes in the networked systems and for inter-process communication in DUNE [LSTS, 2014c].

The IMC message definitions are included in a XML file. There exists a wide variety of predefined messages, and it is also really easy to add custom made messages. An example of a message definition is shown in Figure 3.4 to show the structure and simplicity of the IMC message protocol.

```
<message id="403" name="Desired Roll" abbrev="DesiredRoll" source="vehicle">
  <description>
    Desired Roll angle reference value for the control layer.
  </description>
  <field name="Value" abbrev="value" type="fp64_t" unit="rad">
    <description>
      The value of the desired roll angle.
    </description>
  </field>
</message>
```

Figure 3.4: Example of IMC message definition

3.3.2 Dune

DUNE: Unified Navigational Environment is software meant to be running on the onboard computer in the unmanned vehicle. DUNE is responsible for every interaction with sensors, payloads and actuators. In addition is it used for communications, navigation, control, maneuvering, plan execution and vehicle supervision [LSTS, 2014b]. A lot of these features are already implemented, but one of the strong suits of DUNE is how simple it is to create

[5]Mobile Industry Processor Interface

new Tasks that share information using IMC-messages shared over the Message Bus. This makes a flexible system where code can easily be structured in a modular fashion. This Task interaction is displayed in Figure 3.5.

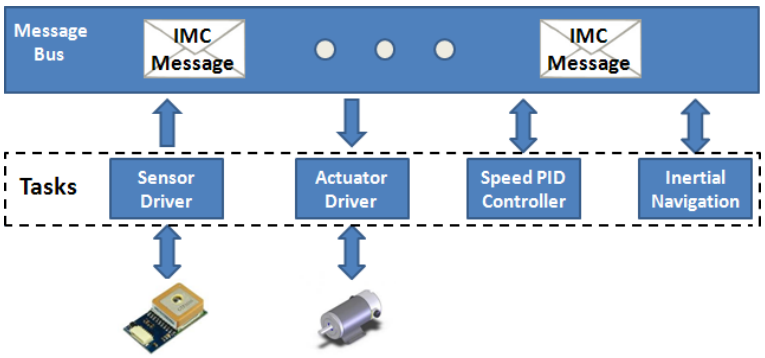


Figure 3.5: Task interaction using IMC *Courtesy lsts.fe.up.pt*

The DUNE project is quite huge containing a lot of pre made and specialized tasks. Only a few of these will be used in each of the applications that use DUNE. To be able to define which tasks that are run an .ini file is used. The .ini files activates the relevant tasks, makes it possible to set task specific parameters and define vehicle types etc. This is a neat way to use the best features of DUNE and still have good control of the software structure.

3.3.3 Neptus

Neptus is a command and control software used to command and monitor unmanned systems. Neptus can be used to observe real time data from the vehicles, it can also be used to log data and revise data from earlier missions. It does also have some useful features that is used for planning, execution and review and analysis of missions [LSTS, 2014d]. Neptus is designed in a way that facilitates development and adoption of new features.

Chapter 4

Pickup and Deployment Mechanism

The aim of the project leading up to this master thesis was to explore different design solutions that could be used to facilitate drop and recovery operations of sensor nodes using AUVs. The project report [Voldsund, 2013] contains a discussion of different solutions, before concluding upon one solution that was implemented. The implemented solution has been further developed in this master thesis, but most of the concepts remain. Readers are advised to consult the project report for discussions concerning the drop and recovery mechanism, while the resulting, though improved mechanism, is presented in this chapter.

4.1 Overall Description

To get sufficient accuracy in the pickup phase it was decided that a solution where a gripper can be lowered down towards the sensor node was the beast approach. The different parts in this setup was 3D-modeled and 3D-printed. A box was designed to contain the PandaBoard and to mount the gripper mechanism to. A system with gears was designed to be able to lower the gripper in an accurate and reliable fashion. The operation of the mechanism is displayed in Figure 4.1. A camera was mounted to the gripper-platform to be able to track the sensor node and confirm whether the node is picked up or not.

(a) Gripper raised

(b) Gripper lowered

Figure 4.1: Drop and recovery mechanism mounted to the hexacopter

4.2 Hardware

4.2.1 IR-LED Position Sensor

The essence of the IR-LED position sensor is the IR-LED and the photodiode. These need some support circuitry to function. The circuit with the LED is fairly simple. It contains only of the IR-led and a resistor. The resistor is there to limit the current.

The circuit for the photodiode is a bit more complicated. The photodiode chosen is a PIN type diode from Everlight. It is connected reversed biased, and does not conduct ordinarily, but starts to conduct when exposed to IR. The more IR the more it conducts. This last fact means that the signal will wary between 0 and 5 V depending on how much IR it "sees". In this application the LED and the photodiode are mounted in a way that the photodiode only "sees" the IR when the gripper platform is in position. Hence it is no need for an analog reading, the signal could be connected directly to a digital I/O port to give ether a in position or an out of position value. An opamp used as an voltage follower is included in the design, this isolates the output from the signal source. The operational amplifier used is an MC3405P from Motorola. A resistor is used to limit current.

A decoupling capacitor was added. Decoupling capacitors are used to give noise a path to ground and keep the power in the circuit smooth [Catsoulis, 2005]. The schematics of the circuit design can be seen in Figure 4.2.

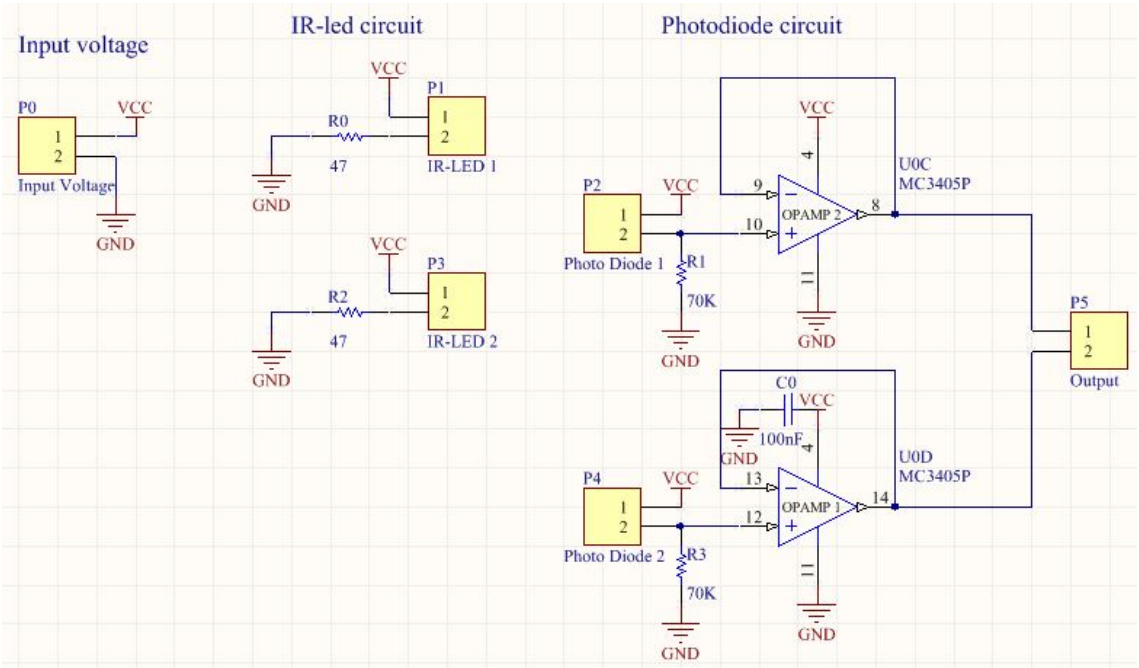


Figure 4.2: Schematics for the IR-LED Position Sensor

4.2.2 H-Bridge

To be able to control direction and speed of the DC-motor lowering and raising the gripper, a H-bridge is needed. The H-bridge chosen for this is the DRV8801 h-bridge motor driver from Texas Instruments mounted to a break out board. The simplicity of this design makes it small in size and very low weight. Nominal output current is rated to 1 A, while it can handle peaks up to 2.8 A for a few seconds. It operates with motor supply voltages between 8 and 36 V and logic supply voltages between 3.3 and 6.5 volts [pololu.com, 2014]. This makes it very versatile. The main drawback with the simplicity of this chip and break out board is that it could quite easily get over heated if driven with high current over time. This should not be a problem in this application because the nominal current of the chosen motor is given to be 50 mA and the time used to lower or raise the gripper is limited.

4.2.3 Level Translator

Communication will go both to and from the PandaBoard. The level translator will need at least eight channels, three for controlling the H-bridge, one for servo control, two for UART and two for the IR-LED position sensor signals. For this reason the ADG3300BRUZ eight channel bidirectional level translator from Analog Devices was chosen. It is a convenient and easy to use level translator, just connect the desired low voltage and low voltage signals at one side and the desired hight voltage and high voltage signals at the other side. Two decoupling capacitors are also connected. The schematics of the circuit design can be seen in Figure 4.3.

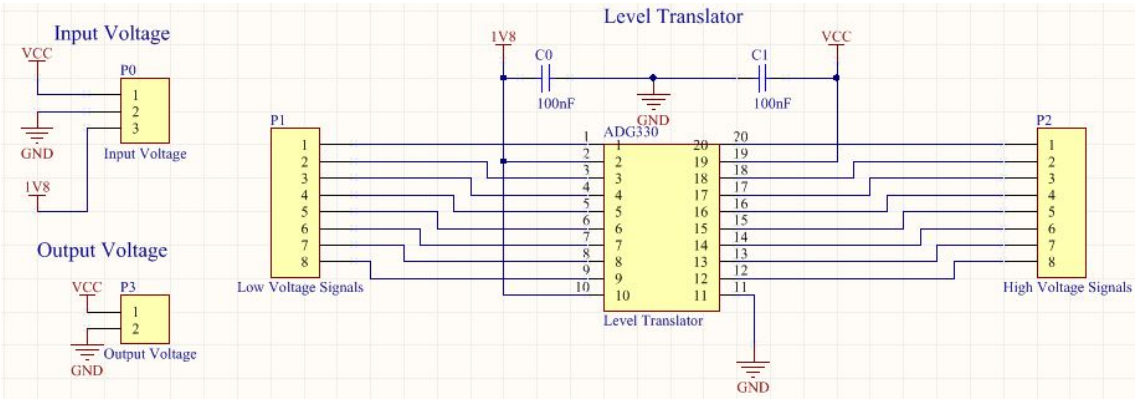


Figure 4.3: Schematics for the Level Translator

4.2.4 PWM-Controller

The servo mounted to the gripper is controlled by using a PWM[1]-signal. Where the time where the signal is high is transferable to the servos position reference. Typically is the time period for PWM signals controlling RC equipment 20 ms. With a high pulse of 1 ms meaning 0 degrees and 2 ms meaning 180 degrees. This means that to be able to control the servo in a stable and accurate way, the PWM-signal needs to be accurate. To get such an accurate signal using the PandaBoard can be challenging. It is possible to generate PWM with the PandaBoard using GPIO-pins and a real time operating system. But the operating system running on the PandaBoard in this project is a stripped down version of a Linux kernel without real time possibilities. In addition will the gripper be controlled through which further increases difficulties with timing.

To cope with this problems a PWM-controller was designed. The basis for the PWM-controller is an ATtiny85 8-bit microcontroller. This microcontroller is chosen for its simplicity. The microcontroller is programmed using ISP. To be able to program the microcontroller in circuit pins are added to all the pins, even though not all will be used in the application. This will also increase the possibilities for further developing if needed. A decoupling capacitor was also added to ensure noise free input voltage for the microcontroller.

Two bits will be used to control the PWM-controller with the PandaBoard. One bit turns on and off the signal, while the other bit gives the desired position of the gripper (either open or closed). The schematics of this circuit is shown in Figure 4.4, while the code running on the ATtiny85 is included in the digital appendix.

[1]Pulse Width Modulated

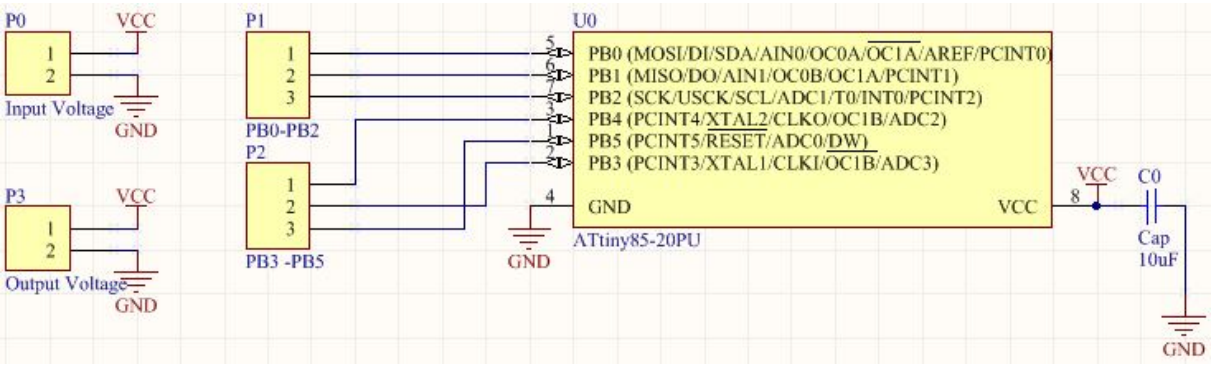


Figure 4.4: Schematics for the PWM-Controller

4.3 Communication Between the Modules

An overview of the different modules and the communication between them is shown in Figure 4.3.

The PandaBoard communicates with the camera through a dedicated port that uses the MIPI CSI-2 standard. The rest of the communication to and from the PandaBoard goes via the level translator described in the previous section. The servo for the gripper is controlled with a PWM-signal while the H-bridge is controlled using three digital outputs. The communication between the PandaBoard and the APM uses UART and the MAVLink protocol. Relevant information here will be attitude and position data, and set points for the controller in the APM.

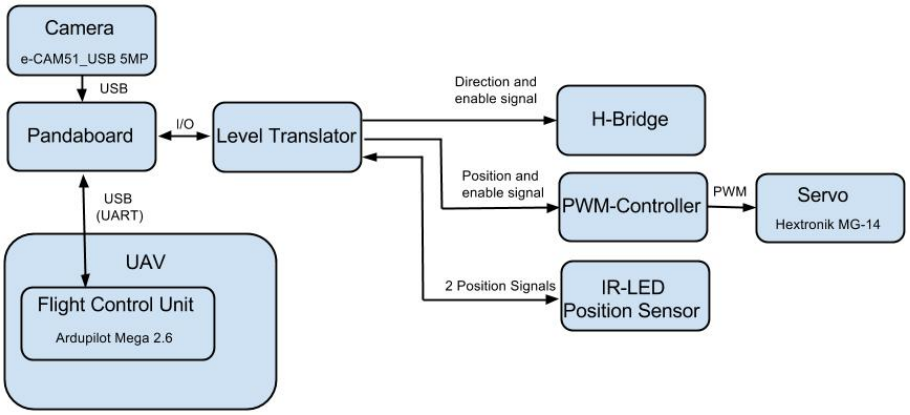


Figure 4.5: Communication between the modules

4.4 Power Supply

The UAV contains a battery and a voltage regulator that regulates the voltage down to 5 V. It is possible to use this voltage source for the parts developed in this project. But using the regulated voltage source could easily result in to much current drawn, which will result in restarts of the APM and the Pandaboard. This could be disastrous for the UAV. To avoid this problem one could use a dedicated voltage regulator for the parts developed in this project, but that will decrease the range of the UAV and increase the chance of losing the UAV in case of problems with the parts developed in this project. Hence the safest and best solution is to add a separate battery and voltage regulator, this will increase weight which will reduce range of the UAV, but this is considered to be a reasonable trade-off. To be able to decide which battery and voltage regulator to use, power demands are investigated and summarized in Table 4.1.

Table 4.1: Power demands

Unit	**Voltage [V]**	**Current Peak [mA]**	**Current Normal [mA]**	**Source of info**
Pandaboard	5	1200	700[2]	Pandaboard FAQ [omappedia.org, 2012]
Servo	5	1500	800	Measured
DC motor	12	50	50	Data sheet [Mclennan, 2013]
Level Translator	5 and 1.8	0.17E-3	5E-3	Data sheet [Analog Devices, 2005]
IR-LED Position Sensor	5	150	450	Measured
Camera	5	150	150	Manufacturer web page [e-con Systems, 2013a]
SUM	-	**3050**	**2150**	

Selecting battery is a trade-off between capacity and weight. The battery should be able to power both the motor and the ICs[3]. Hence a three cell Lithium-Polymer battery is chosen. These kind of batteries deliver 11.1 V, are able to deliver a lot of power fast and able to hold a lot of power in relation to its weight. The specific battery chosen is a 1000 mAh 20-30 C (means that it is able of delivering an instantaneous current of 20-30 times the capacity) battery from Haiyin. The battery is able to support the peak currents, but it might not be able to operate for very long if the servo and motor is used a lot in the mission. The battery will be replaced by a heavier one with greater capacity if necessary.

To reduce the chance for unstable voltages and current delivered to the PandaBoard, two voltage regulators are used, one for the PandaBoard and one for the rest of the equipment.

[3]Integrated Circuits

Especially the servo can be a cause for ripple in the delivered voltage and current. This choices result in the power circuit displayed in Figure 4.6.

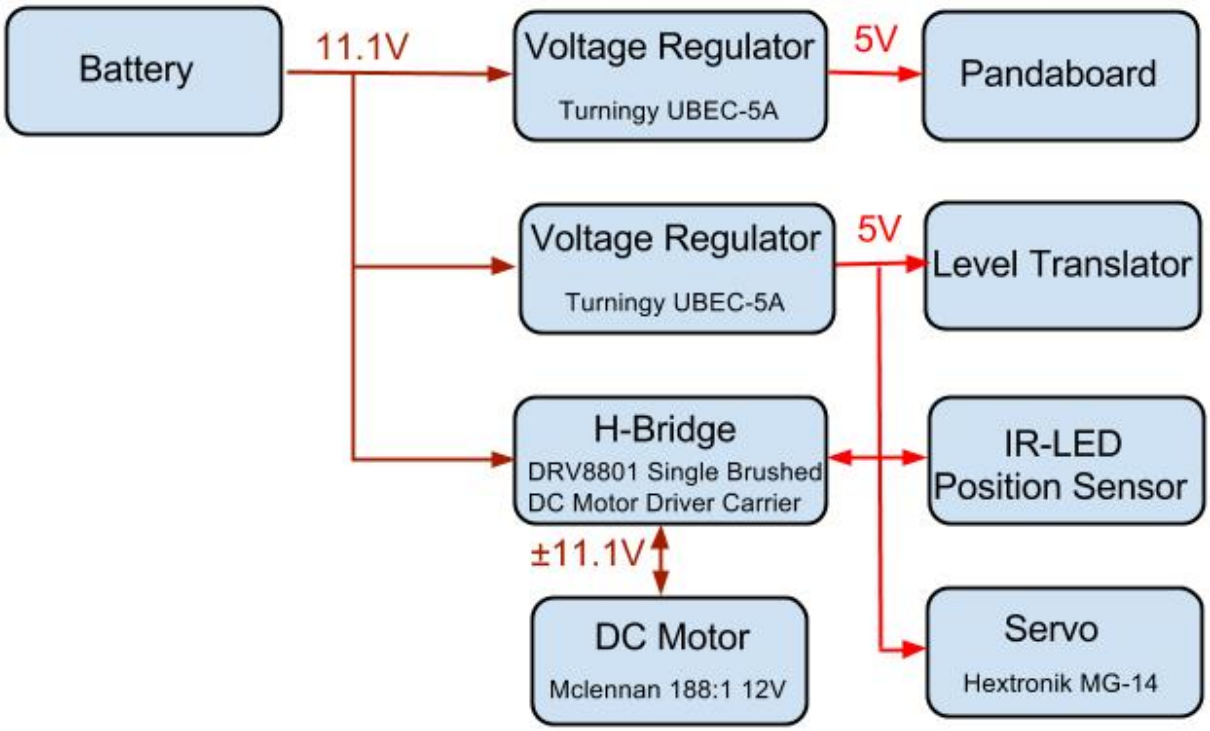


Figure 4.6: Power circuit

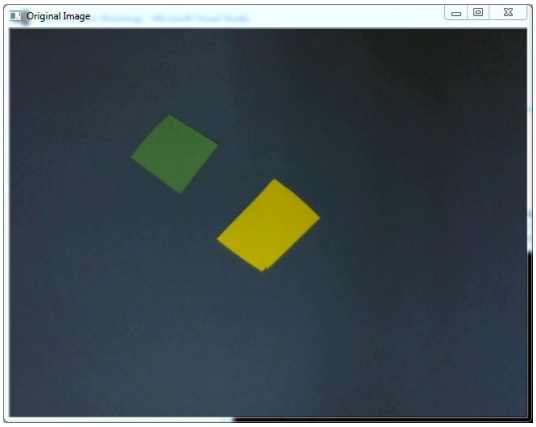
4.5 Camera Application

The main application of the camera is to track the sensor node. The camera will also be used in the pickup to verify whether the gripper got hold of the sensor node or not (one could have used estimation of inertia for this task like [**?**], but the position of the camera makes it very convenient to use for this task). To be able to track the sensor node the camera will need to recognize the sensor node. This can be done in multiple ways as described briefly in Section 2.3.1. The SURF method can be efficient under the right circumstances, but some simple tests revealed some weaknesses. It turned out that only a few points on a picture of the pickup mount was marked as corners that it could use to search for. Of course the pickup mount could be made in a way that it would have more detail, but because it is quite small and it should be possible to detect at a distance this method was discarded. A classifier could be developed, but it will have weaknesses when it comes to rotation, and it would need a lot of example pictures to make it robust. Of this reasons the classifier solution was discarded as well. The much simpler solution of color detection was implemented.

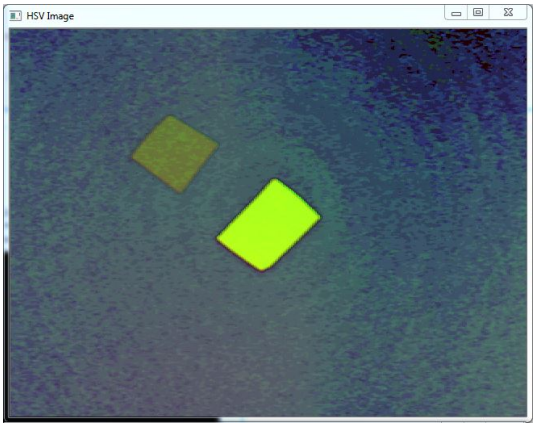
4.5.1 Tracking the Sensor Node

To be able to use color detection the sensor node need to have some colors that sticks out. The mount for the sensor node (Figure **??**) should have two different colors to make it possible to use color detection to get the orientation of the mount. The ocean is blue, which means that the sensor node should have colors that are as far away from blue on the HSV-wheel (Figure 2.1). Yellow is an obvious choice because it is directly opposite of blue and it has a narrow band, which is good for noise cancellation. The colors next to yellow is red and green. Red contains both the lowest an highest hue values, which will make calculations more complex, hence green is chosen for the other color.

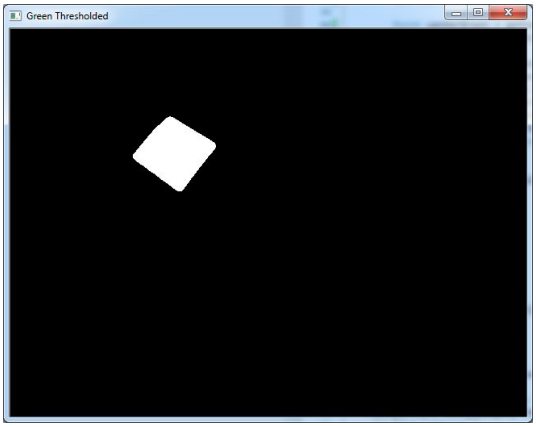
The picture from the camera is converted to the HSV-color space and copied to create two pictures. One of the pictures is thresholded with the HSV-values for yellow, while the other is thresholded with the values for green. This creates two binary pictures where the yellow and green fields are marked respectively. Then the "center of mass" of the two pictures are calculated. This centres of masses are used as centres of the two parts of the pickup mount and used to calculate the orientation of the mount and the center of the mount. This procedure is demonstrated in Figure 4.7. These points are in the picture frame. For the measurements to make sense one need a mapping from this pixel position to the position of the mount in NED.



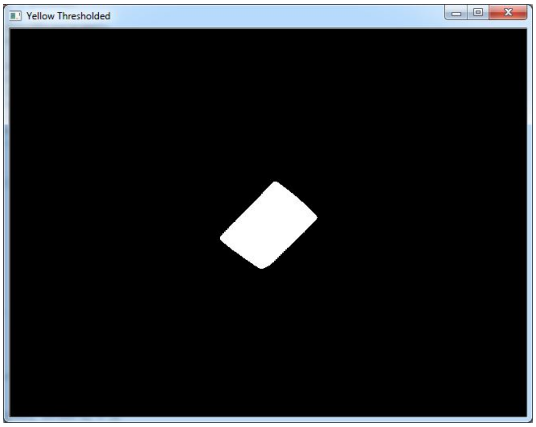
(a) Original image



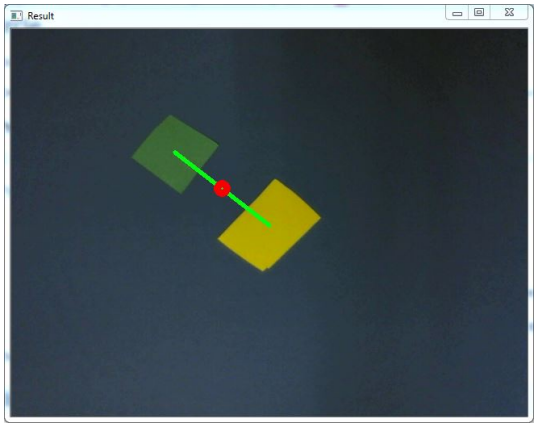
(b) HSV image



(c) Green thresholded



(d) Yellow thresholded



(e) Result

Figure 4.7: Use of color recognition to track the mount of the sensor node

For control purposes a position of the sensor node in NED is good because this will be the same value as the error in position if the controller is trying to get to the exact position of

the sensor node. For estimator and robustness the sensor nodes position should be referenced in ECEF. If the position of the sensor node is referenced in ECEF, it will be simpler to filter out weird measurements, estimation of the movements of the sensor node with current and waves becomes possible and the controller will have something to navigate towards even if the camera looses sight of the sensor node for a moment.

To get the position of the sensor node in NED, first a transformation from BODY to NED is conducted by the use of the rotation matrix from BODY to NED in equation (2.11). DH convention is used to create consequential reference frames from BODY leading up to a reference frame in the sensor node. This is done to exploit the fact that the transformation matrix from BODY to the sensor node reference frame will contain the position of the sensor node reference frame expressed in BODY as seen in equation (2.16).

The first new frame is defined in the center of the camera lens. The homogeneous transformation A_1 from BODY to the camera frame is carried out as a movement d_1 along the BODY z-axis which gives the relation below. This coordinate frame and subsequent frames are visualized in Figure 4.8. The parameters related to the transformations are also marked in the same figure.

$$\boldsymbol{A}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

The next coordinate system is defined with the z-axis pointing towards the sensor node and the pixel position of the sensor node in the picture frame. This is done by rotating around the z-axis of α degrees and then rotating around the x-axis of β degrees.

To find α and β some calculations needs to be done. The origin of the picture plane is in the topmost left corner. To get the angles to rotate, the origin is moved to the center of the picture frame by defining $\delta x = x_o - x_c$ and $\delta y = y_o - y_c$. The center of the picture frame is the point (x_c, y_c) while the pixel position of the sensor node is the point (x_o, y_o). The picture frame is defined to be at a distance of one meter away from the camera lens and the length between each pixel at this distance (L) is measured using an object of known length. The angles α and β are then calculated.

$$\alpha = atan2(\delta y, \delta x) \tag{4.2}$$
$$d = L\sqrt{\delta x^2 + \delta y^2} \tag{4.3}$$
$$\beta = \tan^{-1}(d) \tag{4.4}$$
$$\tag{4.5}$$

The resulting homogeneous transformation matrix is

$$\boldsymbol{A}_2 = \begin{bmatrix} c_\alpha & -s_\alpha & 0 & 0 \\ s_\alpha & c_\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\beta & -s_\beta & 0 \\ 0 & s_\beta & c_\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_\alpha & -s_\alpha c_\beta & s_\alpha s_\beta & 0 \\ s_\alpha & c_\alpha c_\beta & -c_\alpha s_\beta & 0 \\ 0 & s_\beta & c_\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

The last reference frame is defined in the sensor node. To get there a movement of d_2 along the z-axis is necessary. To calculate this distance, the angle (γ) between the z-axis in NED and the z-axis in the picture frame reference system needs to be calculated. The transformation matrix from NED to picture frame is calculated to be able to read out the direction of the z-axis in the picture frame reference system.

$$\boldsymbol{T}_{pf}^{n} = \boldsymbol{R}_{b}^{n}(\boldsymbol{\Theta}_{nb})\boldsymbol{A}_1\boldsymbol{A}_2 \tag{4.7}$$

Out of this transformation matrix one can read out the direction of the z-axis of the picture frame reference system $\boldsymbol{z}_{pf}^{n}$ while the direction of the z-axis of NED is trivial.

$$\boldsymbol{z}_{pf}^{n} = \begin{bmatrix} s_\alpha s_\beta c_\psi c_\theta - c_\alpha s_\beta(-s_\psi c_\phi + c_\psi s_\theta s_\phi) + c_\beta(c_\psi c_\phi + s_\psi s_\theta s_\phi) \\ s_\alpha s_\beta s_\psi c_\theta - c_\alpha s_\beta(c_\psi c_\phi + s_\psi s_\theta s_\phi) + c_\beta(-c_\psi s_\phi + s_\psi s_\theta s_\phi) \\ -s_\alpha s_\beta s_\theta - c_\alpha s_\beta c_\theta s_\phi + c_\beta c_\theta c_\phi \end{bmatrix} \boldsymbol{z}_{n}^{n} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.8}$$

The angle γ between these two vectors can be calculated using dot product, and the fact that the vectors in the rotation matrices are normalized. This combined with basic trigonometry gives these two relations.

$$\cos\gamma = \frac{\boldsymbol{z}_{pf}^{n} \cdot \boldsymbol{z}_{n}^{n}}{|\boldsymbol{z}_{pf}^{n}||\boldsymbol{z}_{n}^{n}|} = z_{pf_3}^{n} \tag{4.9}$$

$$\cos\gamma = \frac{h - d_1 c_\theta c_\phi}{d_2} \tag{4.10}$$

These relations combined gives the following expression for d_2 and the homogeneous transform.

$$d_2 = \frac{h - d_1 c_\theta c_\phi}{z_{pf_3}^{n}} \tag{4.11}$$

$$\boldsymbol{A}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.12}$$

$$\boldsymbol{T}_{obj}^{n} = \boldsymbol{T}_{pf}^{n}\boldsymbol{A}_3 \tag{4.13}$$

Calculations of equation (4.13) will read out the position of the origin of the reference frame in the sensor node (according to equation (2.16)). This gives

$$\boldsymbol{p}_{obj}^{n} = \begin{bmatrix} d_2 z_{pf_1}^{n} + d_1(s_\psi s_\phi + c_\psi s_\theta c_\phi) \\ d_2 z_{pf_2}^{n} + d_1(-c_\psi s_\phi + s_\psi s_\theta c_\phi) \\ h \end{bmatrix} \tag{4.14}$$

For the purpose of tracking this vector is then transformed into the ECEF coordinate frame by the transform in equation (2.8).

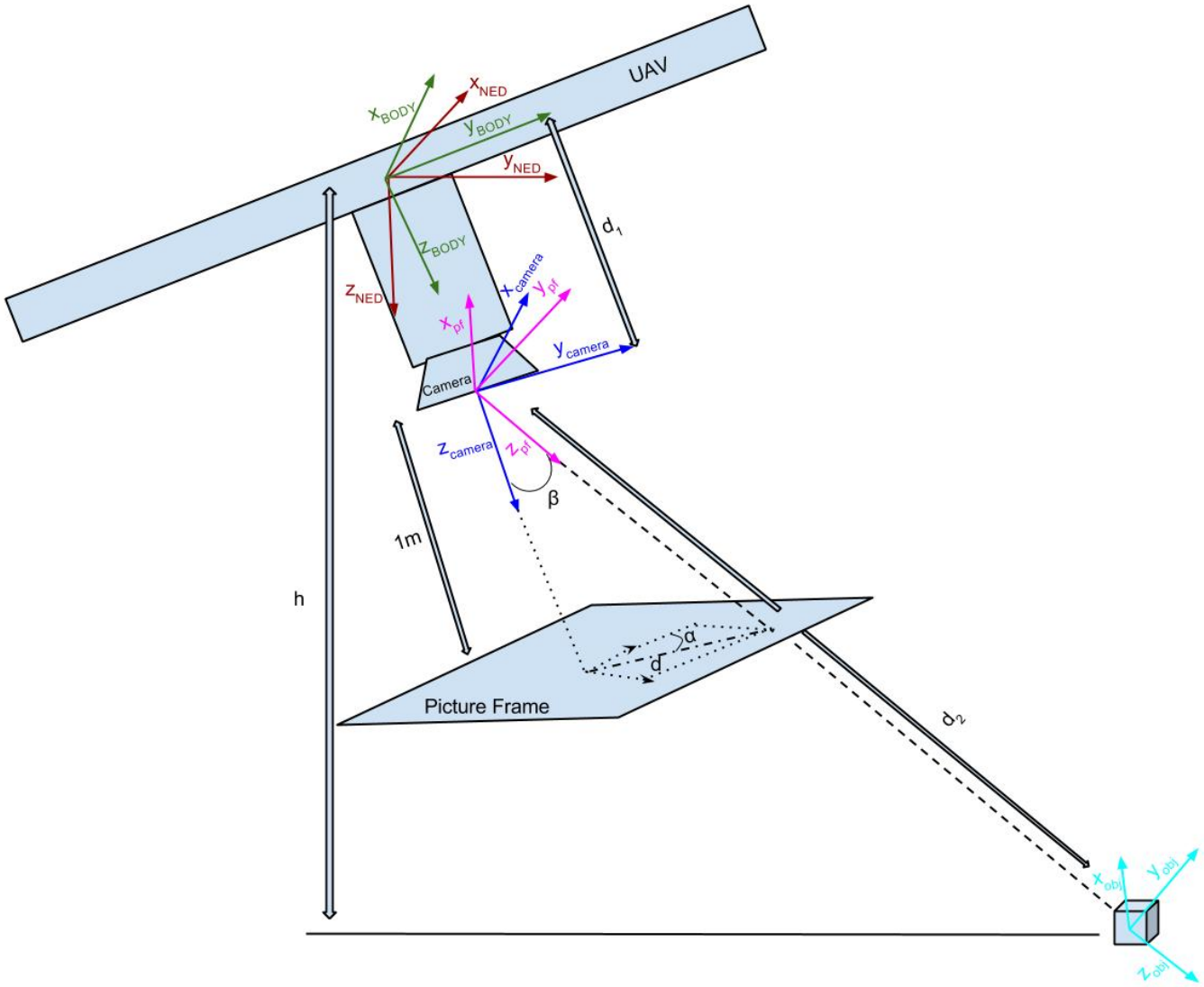


Figure 4.8: Visualization of the different coordinate systems

Chapter 5

Control Structure

5.1 Position Controller

A position controller (controlling North and East positions) for the AUV was derived using Newtons second law of motion. The controller is meant to be used for short distance movements, like in the node pickup phase. The controller calculates reference values for roll and pitch for the UAV. These reference values are then fed to the low level roll and pitch controllers in the APM. A mathematical derivation and a description of the implementation of the controller follows.

5.1.1 Mathematical Description

The forces acting upon the UAV is if one excludes disturbances, gravity and the thrust caused by the propellers. Gravity acts in Down direction, which means that it does not influence the North or East position of the UAV. Thrust on the other hand acts in negative z direction referenced in the BODY - frame. Hence this value should be transformed to the NED - frame to be able to control the horizontal NED - position. This is done using the rotational matrix given in equation (2.11), and the result is shown in equation (5.1).

$$\boldsymbol{R}_b^n(\boldsymbol{\theta}) \begin{bmatrix} 0 \\ 0 \\ -T \end{bmatrix} = T \begin{bmatrix} s_\psi s_\phi + c_\psi c_\phi s_\theta \\ -c_\psi s_\phi + s_\theta s_\psi c_\phi \\ c_\theta c_\phi \end{bmatrix} \tag{5.1}$$

Applying small angle approximation for roll and pitch angles simplifies equation (5.1) to equation(5.2).

$$T \begin{bmatrix} -s_\psi & -c_\psi \\ c_\psi & -s_\psi \end{bmatrix} \begin{bmatrix} \phi \\ \theta \end{bmatrix} \tag{5.2}$$

Using Newtons second law of motion and defining roll and pitch angles as control variables used as input to a low level controller results in equation (5.3).

$$\begin{bmatrix} \ddot{N} \\ \ddot{E} \end{bmatrix} = \frac{T}{m} \begin{bmatrix} -s_\psi & -c_\psi \\ c_\psi & -s_\psi \end{bmatrix} \begin{bmatrix} \phi_r \\ \theta_r \end{bmatrix} = \boldsymbol{B}\boldsymbol{u} \tag{5.3}$$

Equation (5.3) can be transformed into a forced mass-spring-damper system by setting $\boldsymbol{u}$ as shown in equation (5.4) and inverting $\boldsymbol{B}$ as shown in equation (5.5). Note that $\boldsymbol{\eta}$ here is used as a subset of the NED position vector and is $\boldsymbol{\eta} = [NE]^T$. The resulting force mass-spring-damper system is expressed in equation (5.6).

$$\boldsymbol{u} = \boldsymbol{B}^{-1}(-\boldsymbol{K}_d\dot{\boldsymbol{\eta}} - \boldsymbol{K}_p\boldsymbol{\eta} + \boldsymbol{K}_p\boldsymbol{\eta_r}) \tag{5.4}$$

$$\boldsymbol{B}^{-1} = \frac{m}{T}\begin{bmatrix} -s_\psi & c_\psi \\ -c_\psi & -s_\psi \end{bmatrix} \tag{5.5}$$

$$\ddot{\boldsymbol{\eta}} + \boldsymbol{K}_d\dot{\boldsymbol{\eta}} + \boldsymbol{K}_p\boldsymbol{\eta} \;\; = \;\; \boldsymbol{K}_p\boldsymbol{\eta_r} \tag{5.6}$$

Proportional and derivational controller gains can be chosen $\boldsymbol{K}_p = \omega_0^2\boldsymbol{I}_{2x2}$ and $\boldsymbol{K}_d = 2\xi\omega_0\boldsymbol{I}_{2x2}$ according to the demands of the system.

Some early stage testing revealed the need for integral action in the controller. The testing was conducted inside so wind was not the issue, but a constant deviation was present due to inaccuracies in the sensors and low level controllers of the APM. The integral term is according to equation (5.7).

$$\boldsymbol{u}_i = \boldsymbol{B}^{-1}\boldsymbol{K}_i\int_0^t(\boldsymbol{\eta}_r - \boldsymbol{\eta}) \tag{5.7}$$

The augmented controller then becomes

$$\boldsymbol{u} = \boldsymbol{B}^{-1}(-\boldsymbol{K}_d\dot{\boldsymbol{\eta}} - \boldsymbol{K}_p\boldsymbol{\eta} + \boldsymbol{K}_p\boldsymbol{\eta}_r + \boldsymbol{K}_i\int_0^t(\boldsymbol{\eta}_r - \boldsymbol{\eta})) \tag{5.8}$$

5.1.2 Implementation

Some practical issues arise in the implementation of the controller in equation (5.4). These issues are related to the size of the thrust (T), the roll and pitch angles and choosing controller gains.

The size of the thrust is varying and will influence the controller quite a lot. To reduce this influence is the controller used when altitude is constant or not varying to much. Hence it is used in situations where the thrust equals the weight of the UAV and the approximation $T = mg$ is used.

Roll and pitch angles needs to be kept small for the mathematics behind the controller to be valid. To obtain this both roll and pitch angle references sent to the APM are limited to be within $\pm$ 0.2 radians. These limitations are also important for safety reasons to avoid that huge roll and pitch angles are demanded by the controller if the set point is far away from the UAV. The way that the controller is implemented will for position references far

away give constant roll and pitch reference values limited by the $\pm$ 0.2 radians rule, this will lead to increasing speeds with following overshoot. Hence as already mentioned this controller is meant for short distance movements only. For longer distances either a speed controller should be used or a reference model should be used combined with this controller.

The controller gains are chosen to give a critically damped, fast but not to aggressive system. This is done by first of all selecting $\xi = 1$ to make the system critically damped. the natural frequency is chosen through simulations and verified by software in the loop testing, before verifying by real world test. Simulations are conducted by the use of MATLAB and Simulink, where the UAV including low level controllers and thrust allocation was modelled and simulated.

5.2 Software Implementation

This section describes the structure and functionality of the software controlling the UAV and drop and recovery mechanism. An overview of the different tasks and how they communicate is shown in Figure (5.1). An explanation of each task follows.

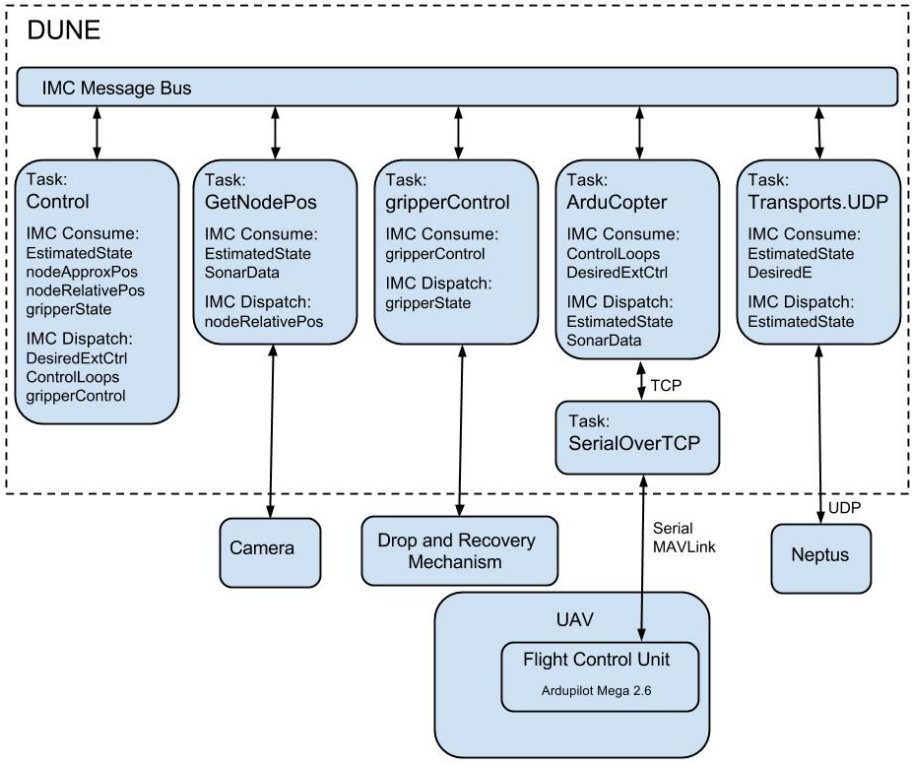


Figure 5.1: Software structure

getNodePos

This task uses color recognition to track the sensor node according to the procedure described in section 4.5.1. It uses the camera in combination with attitude data from the APM and sonar measurements to calculate the position of the sensor node relative to the UAV. Also the heading of the mount of the sensor node is calculated. This information and a parameter telling whether the sensor node is spotted by the camera or not is put on the IMC-bus in the nodeRelativePos message.

gripperControl

The gripper control task consumes gripperControl messages from the IMC-bus and controls the gripper and gripperplatform according to the received commands. The gripper is controlled by setting a gripper enable and a gripper position bit on the PandaBards GPIO port. The position bit is high for open position and low or closed position. the gripperplatform is controlled by a motor which lowers and raises the platform. The motor is controlled by setting an enable bit and a direction bit on the PandaBoards GPIO port. Signals from the LED-sensor is read from the GPIO port to be able to see when the gripperplatform is in the desired position (either lowered or raised). After executing the commands it dispatches a gripperState message to the IMC-bus containing the positions of the gripper and gripperplatform.

ArduCopter

The ArduCopter task is a result of work done by the Hexacopter group at AMOS. This task plays the role of a bridge between the APM and DUNE. ArduCopter sets up which messages that should be received from the APM and sends control messages to the APM.

Transports.UDP

The Transports.UDP task consumes messages from the IMC bus and sends them by the use of UDP. This is used as an interface between DUNE and Neptus.

Control

The control task is the main task in this control structure. It operates with a state variable to be able to take actions based on the phase of the mission. The different states are UNDEF, Initial searc, approching node, secondary search and node picket up. The system starts up in the UNDEF state and leaves it when it receives attitude data from the APM (via the IMC bus). Then the state switches to initial search and the UAV is guided to the approximate position of the sensor node which is given in the nodeApproxPos IMC message (this is for the time being defined by the ini file, further development will be to integrate this into Neptus). The UAV is guided towards the sensor node using the position controller described in Section 5.1, the error in position is calculated by the use of the haversine formula*. The camera is constantly searching for the sensor node while flying towards the approximate node

position. When the camera finds the node the state will change to Approching node, the gripperplatform will be lowered and the gripper will be opened. In the Approaching node state the nodeRelativePos measurements will be used as feedback to the position controller. This means that the UAV will be positioned straight above the sensor node. When sufficiently close to the position straight above the sensor node will the UAV start to descend upon the sensor node. When the node is reachable to the gripper will the gripper close. If the gripper catches the node will the state change to Node picked up and the AUV will return and the gripperplatform will be raised. If while descending down upon the sensor node the camera looses sight of the sensor node, the state will change to Secondary search and the UAV will try to get back to the position of the sensor node to spot it again.

Chapter 6

Simulation of the System

To be able to verify the controller and the general control structure, the system was simulated using MATLAB and Simulink. This meant that a mathematical model of the hexacopter needed to be derived and that the low level controllers and thrust allocation of the APM needed to be simulated using some approximations and assumtions on its behaviour. The different models are derived below, then they are put together to a full system including the controller to be able to put the controller to the test.

6.1 Mathematical Model of the Hexacopter

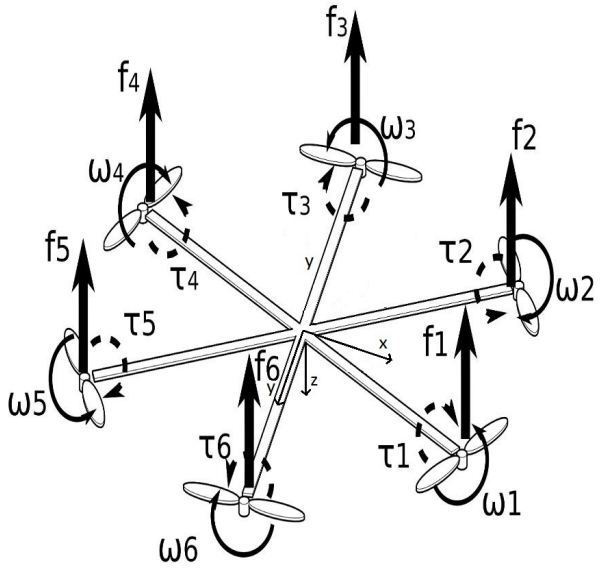


Figure 6.1: Model of hexacopter *Courtesy of [Alaimo et al., 2013]*

To model the dynamics of the hexacopter equations (2.17) and (2.18) are used. The key component that needs to be derived is the torque vector $\boldsymbol{\tau}_{RB}$, which is expressed in the

BODY-frame. The thrust from each propeller is according to [Alaimo et al., 2013] expressed as a lift constant times the squared angular speed of the propeller. In addition they approximate the moment caused around the propeller axis as a drag constant times the squared angular speed of the propeller plus the inertia moment of the propeller times the angular acceleration of the propeller. This is shown in equations (6.1) and (6.2).

$$\boldsymbol{f}_i = \begin{bmatrix} 0 & 0 & k\omega_i^2 \end{bmatrix}^T \tag{6.1}$$

$$\tau_{M_i} = b\omega_i^2 + I_{M_i}\dot{\omega}_i \tag{6.2}$$

Where k is the lift constant, b is the drag constant, I_m is the inertia moment of a propeller and ω_i is the angular speed of propeller i. These equations and some simple geometry gives the following force and moment balances, where ϕ is the roll angle, θ is the pitch angle and l is the length of the arm from the center of gravity to the center of the propeller.

$$\boldsymbol{\tau}_{RB} = \begin{bmatrix} -mg\sin(\theta) \\ mg\cos(\theta)\sin(\phi) \\ mg\cos(\theta)\cos(\phi) - k\sum_{i=1}^{6}\omega_i^2 \\ kl(-\frac{1}{2}\omega_1^2 + \frac{1}{2}\omega_2^2 + \omega_3^2 + \frac{1}{2}\omega_4^2 - \frac{1}{2}\omega_5^2 - \omega_6^2 \\ \frac{\sqrt{3}}{2}kl(\omega_1^2 + \omega_2^2 - \omega_4^2 - \omega_5^2) \\ b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2 + \omega_5^2 - \omega_6^2) + I_m(\dot{\omega}_1^2 + \dot{\omega}_2^2 + \dot{\omega}_3^2 + \dot{\omega}_4^2 + \dot{\omega}_5^2 + \dot{\omega}_6^2) \end{bmatrix} \tag{6.3}$$

Not considering the different parameters, all the information needed to use equations (2.17) and (2.18) are present, which gives.

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}_{RB}^{-1}(\boldsymbol{\tau}_{RB} - \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}) \tag{6.4}$$

This equation is then transformed to the NED frame using the rotation matrix in equation (2.11) and the transformation matrix in equation (2.13) as shown in equation (6.5).

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb}) & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb}) \end{bmatrix} \boldsymbol{\nu} \tag{6.5}$$

Where $\boldsymbol{\eta} = [\boldsymbol{p}^n, \boldsymbol{\Theta}_{nb}]^T$ is the position in NED and the Euler angles.

6.2 Simulation of Low Level Controllers and Thrust Allocation in the APM

The attitude controllers in the APM are approximated as PD-controllers. One must assume that the controllers of the APM follows the references given, PD-controllers will do this, hence this is a fair approximation. The height controller of the APM is approximated as a PID-controller.

The purpose of the thrust allocation algorithm is to convert desired force or moments in the different degrees of freedom into desired thrust from the different motors. The control vector given by the height and attitude controllers will contain desired thrust in negative z-direction and desired moments around the different axes. This vector is given by $\boldsymbol{\tau}_c = [T\ \tau_{\phi_c}\ \tau_{\theta_c}\ \tau_{\psi_c}]^T$. Using equation (6.3) combined with the control vector gives the following relationship.

$$\boldsymbol{\tau}_c = \begin{bmatrix} k & k & k & k & k & k \\ -\frac{1}{2}kl & \frac{1}{2}kl & kl & \frac{1}{2}kl & -\frac{1}{2}kl & -kl \\ \frac{\sqrt{3}}{2}kl & \frac{\sqrt{3}}{2}kl & 0 & -\frac{\sqrt{3}}{2}kl & -\frac{\sqrt{3}}{2}kl & 0 \\ b & -b & b & -b & b & -b \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \\ \omega_5^2 \\ \omega_6^2 \end{bmatrix} = \boldsymbol{A}\boldsymbol{u} \tag{6.6}$$

$$\boldsymbol{u} = \boldsymbol{A}^+\boldsymbol{\tau}_c \tag{6.7}$$

$$\boldsymbol{A}^+ = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1} \tag{6.8}$$

The desired input to each motor is calculated using equation (6.7), where the pseudo inverse of $\boldsymbol{A}$ is calculated as shown in equation (6.8).

6.3 Simulation of the Camera Algorithm

To simulate the measurements from the camera, the algorithm given in Section 4.5.1 is used to find the position of each of the corners of the camera frame. Then a ray casting algorithm is used to determine if the position of the sensor node is within the camera frame. The ray casting algorithm is outside the scope of this report.

If the sensor node is within the camera frame, perfect measurements of both node position in relation to the hexacopter and the heading of the node are sent back to the control algorithm.

6.4 Simulink Model

All the components for the simulator explained abov,e in addition to a PandaBoard component containing the controller described in the previous chapter is put together to create the Simulink model shown in Figure 6.2.

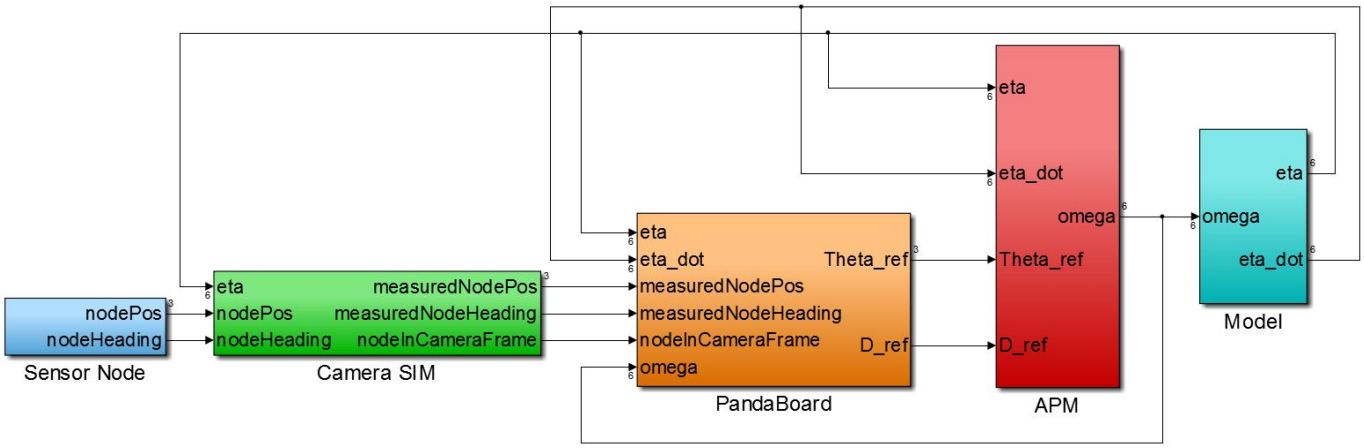


Figure 6.2: Simulink

6.5 Test of Controller

The main scenario simulated is as close to the lab tests that will be conducted as possible. As the main challenge of the hexacopter controller is the pickup phase, the simulation will focus on this phase. In each of the simulations the hexacopter ramps up from a ground start to a height of 4 m. The sensor node is assumed to be lying 0.5 m North and 0.5 m East of the starting point. As the hexacopter is flying the simulated camera will give measurements of the position of the sensor node, but only if the sensor node would have been in the camera frame. If the camera is unable to spot the sensor node, the position reference used for the controller will be the current position. The hexacopter will also try to maintain the same heading as the sensor node. If the sensor node is in the camera frame, the reference heading for the hexacopter will be updated, if the view of the sensor node is lost, the reference will be the last measured node heading

Three scenarios will be simulated, first a simulation with no disturbances, then a simulation with constant disturbance and lastly a simulation with varying disturbances.

6.5.1 Simulation Without Disturbances

The scenario was first simulated using the proposed controller from equation (5.4) without any disturbances present.

Results

The North-East-Down position of the sensor node is plotted in Figure 6.3. It can be seen from the figure how the reference position is updated when the camera spots the sensor node.

And one should also note that the hexacopter ramps down all the way to the sensor node without loosing sight of the sensor node.

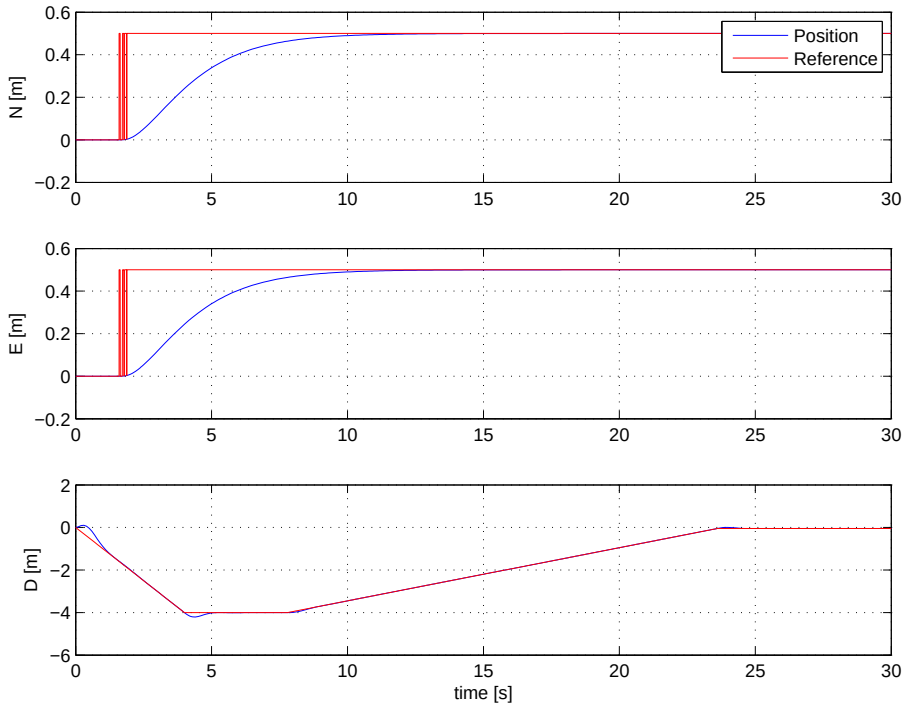


Figure 6.3: North-East-Down position of the hexacopter

Figure 6.4 shows the the North-East position of the hexacopter and the sensor node. It also shows the camera frame and whether the node is spotted by the camera or not. It can also be seen from this plot how the hexacopter yaws to keep the same heading as the sensor node (this can be seen even better from attitude plots shown in Appendix A).

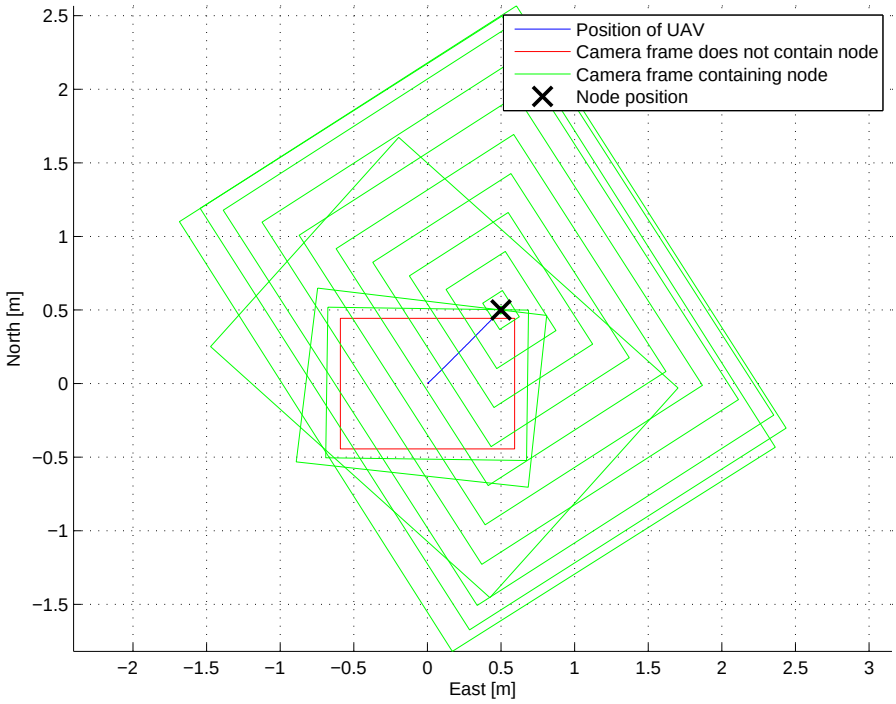


Figure 6.4: North-East-Down position of the hexacopter, sensor node and camera frame

Discussion

The hexacopter is able to lower itself down on the exact spot of the sensor node without loosing sight of the sensor node. This is very promising results, although some early tests in the lab showed that the controller used in this simulation would give some stationary deviations. The lab tests were conducted inside so wind played no part in this error. The error is probably due to unbalances of the hexacopter and inaccuracies of the low level controllers of the APM. Hence integral action was added to the controller. A simulation with constant disturbance and integral action follows.

6.5.2 Simulation With Constant Disturbances

The disturbance added could for instance represent a wind affecting the hexacopter with a force of 0.5 N in North-direction.

Results

As can be seen from Figure 6.6 and Figure 6.6 the hexacopter overshoots the reference position. The controller is able to cancel out the disturbance and make the hexacopter descend upon the sensor node.

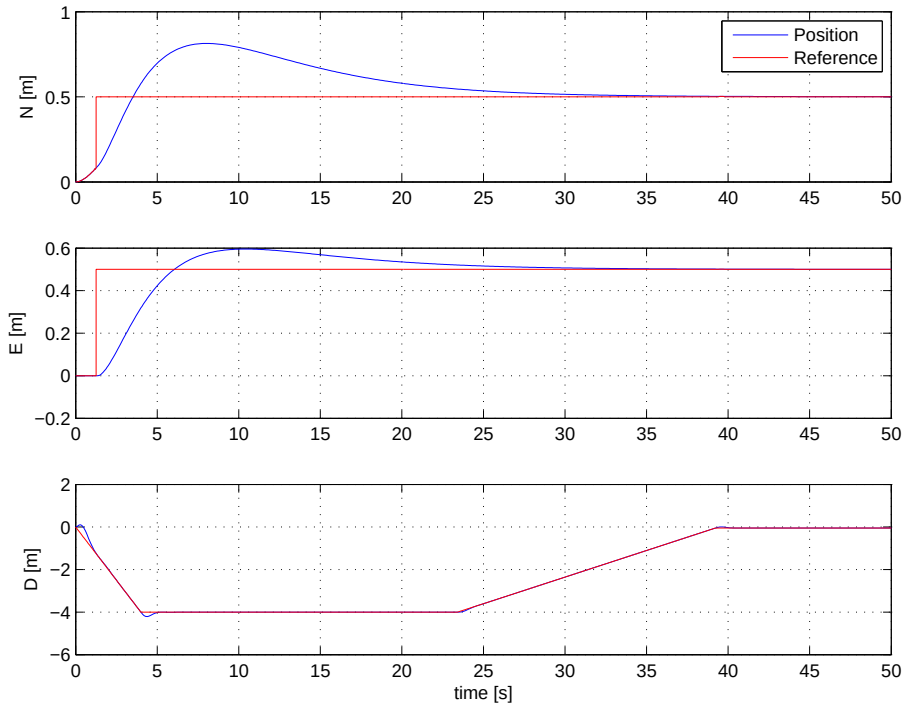


Figure 6.5: North-East-Down position of the hexacopter

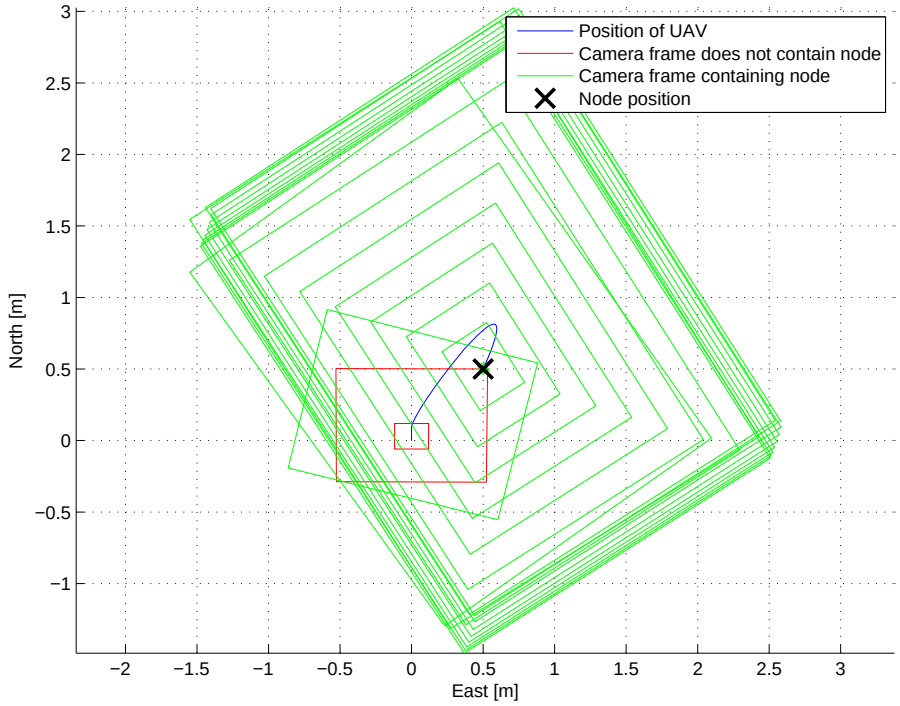


Figure 6.6: North-East-Down position of the hexacopter, sensor node and camera frame

Discussion

The overshoot of the position reference is due to the I-term of the controller and is quite natural due to the step in reference position. The fact that the disturbances is working in North-direction will also lead to an overshoot in North-direction, due to the added forces in that direction. The controller is able to correct for the disturbance and descend upon the sensor node, which is very good.

6.5.3 Simulation With Varying Disturbances

The controller is tested in simulations where the disturbances are varying in both North and East-direction using integrated band limited white noise where the integral is limited by $\pm$ 1 N in both directions. The magnitude and direction of the disturbance is plotted in the appendix.

Results

The hexacopter stays approximately straight above the sensor node, and is able to lower itself a bit down towards the sensor node before it looses the sight of the node and drifts away. Both these observations are easily seen in Figures 6.7 and 6.8.

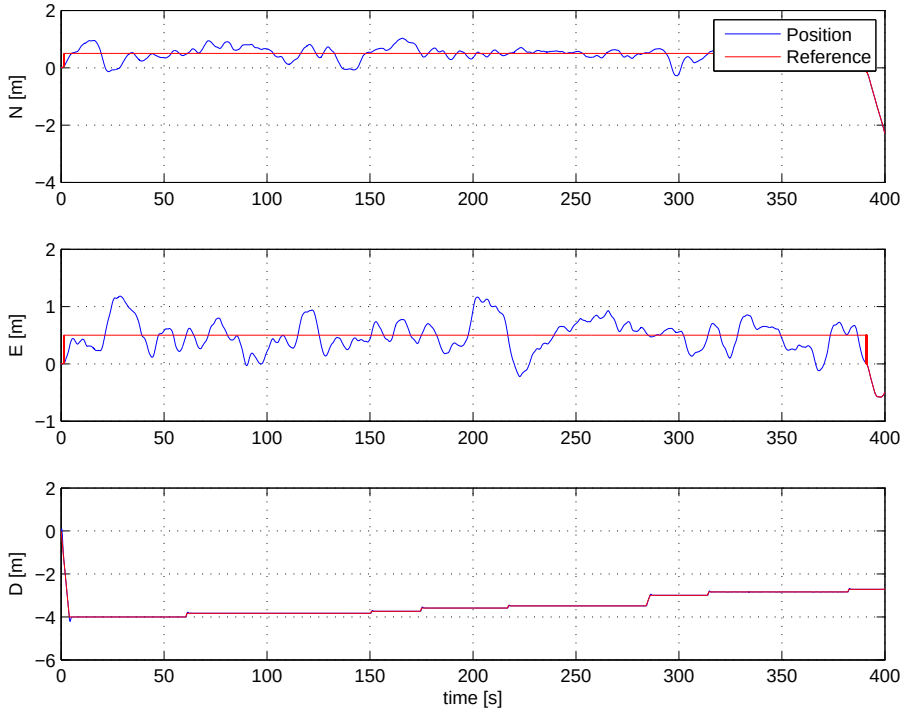


Figure 6.7: North-East-Down position of the hexacopter

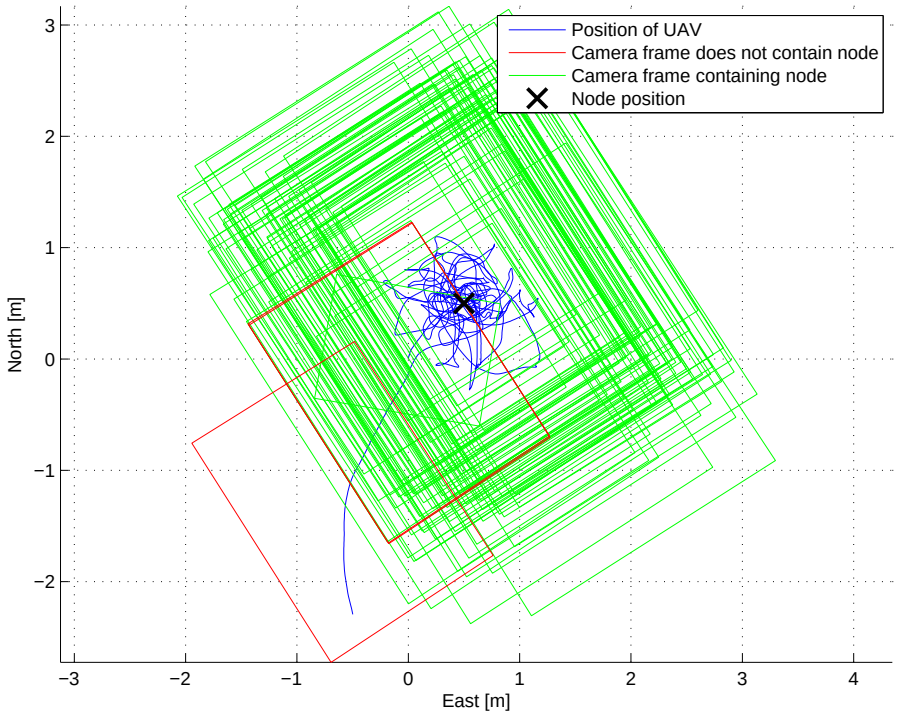


Figure 6.8: North-East-Down position of the hexacopter, sensor node and camera frame

Discussion

The results show that the controller is able to handle some random disturbances but sooner or later these disturbances will lead to problems where the camera will loose sight of the sensor node and then drift off. This will make pick up in such conditions impossible.

6.5.4 Simulation With Constant Disturbances on the Hexacopter and Disturbances Affecting the Sensor Node

A more realistic simulation is to simulate the system with constant disturbance, e.g. due to wind or inaccuracies in the APM. And let the sensor node be affected by some varying noise, which makes it change heading and position. The noise is modelled as random walk.

Results

As seen from Figures 6.9 and 6.10 the hexacopter tracks both position and heading of the sensor node, and is actually able to get down to 5 cm above the sensor node before loosing sight of it and drifting off.

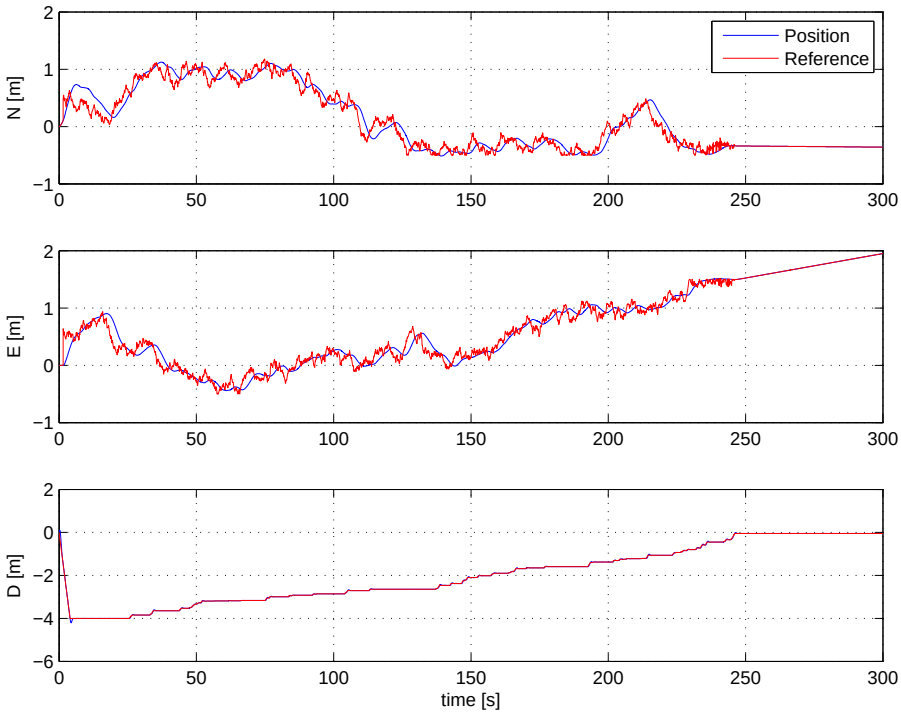


Figure 6.9: North-East-Down position of the hexacopter

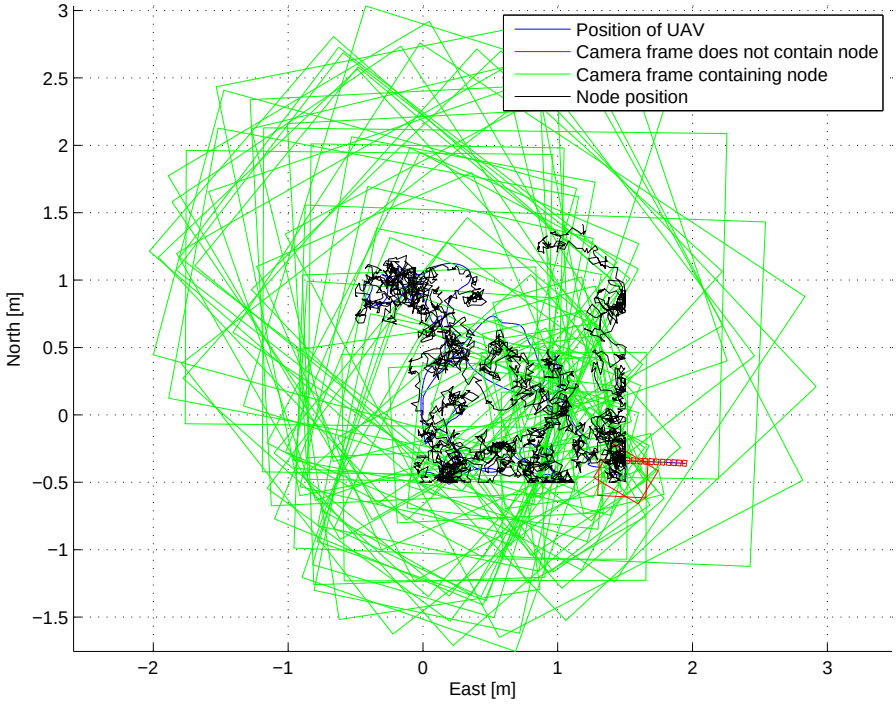


Figure 6.10: North-East-Down position of the hexacopter, sensor node and camera frame

Discussion

The hexacopters abilities to track the sensor nodes position and heading is quite promising. The main issue that is revealed is the need for some way to get back to the previous position where the sensor node was spotted last. The simulations have also shown that this setup is not able to handle rough weather conditions, meaning that operation should be limited to days with only small waves and little wind.

Chapter 7

Software In The Loop Testing

When working with UAVs safety is a critical issue. To be able to verify software before test flights is of great importance and will make a much simpler workflow. This chapter will go through the setup of the software in the loop (SITL) test. And explain how the different systems are interfaced to each other. The different SITL-tests conducted will also be presented.

7.1 SITL-Setup

The main purpose of the SITL-test is to verify the control software presented in Chapter 5.2.

The SITL-setup contains of six main components. These components are DUNE, Neptus, ArduCopter SITL, ArduPilot, MAVProxy and APM Planner. These have different functions, and not all of them are necessary. The interface between the different components is displayed in Figure 7.1, and their roles are briefly explained below.

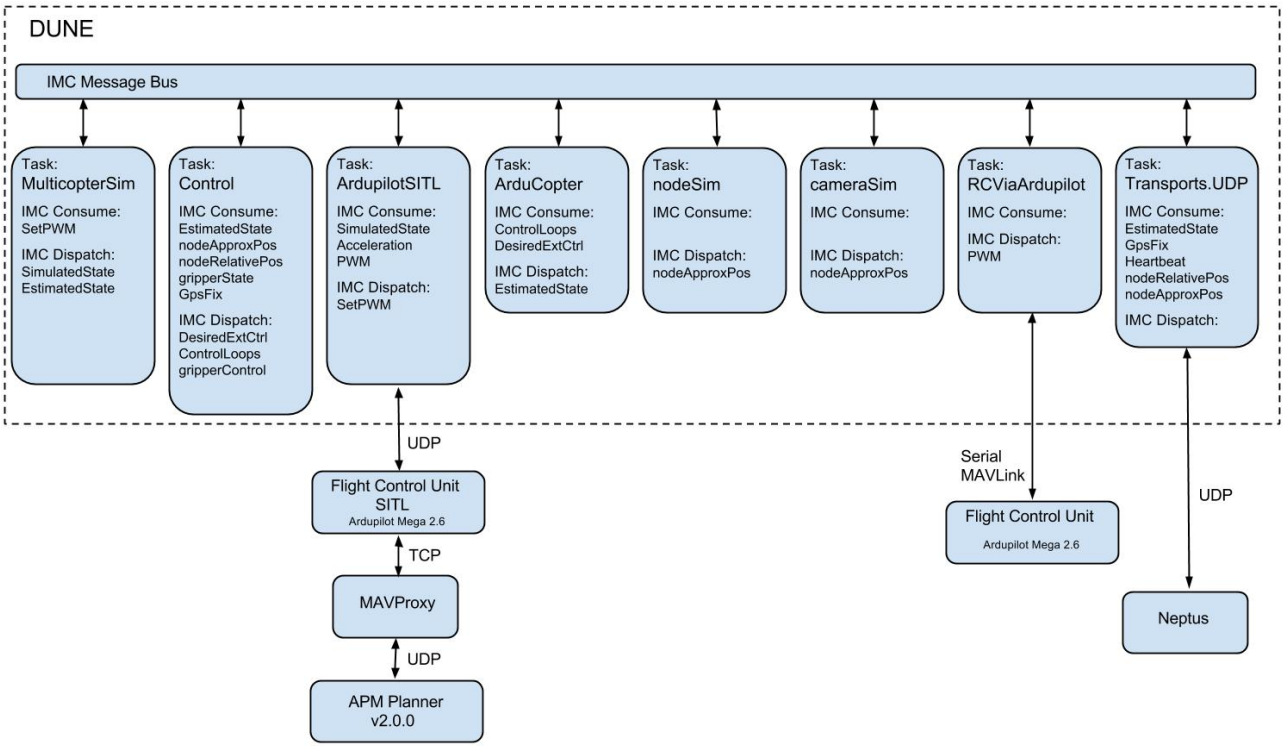


Figure 7.1: Setup used for SITL-testing

DUNE

DUNE is the part of the SITL that is put to the test. To be able to see if the control algorithms work. Some of the features needs to be simulated. This is done in Task that only are run in SITL mode. A model of a multicopter is used to calculate realistic attitude and position states based of the actuator inputs given to the Task. There is another Task that simulates the camera measurements. This is done in the same way as explained in the Simulation Chapter.

Neptus

Neptus is used to monitor values of the relevant IMC-messages. It is used in exactly the same way that it is used in real missions.

ArduCopter SITL

The ArduCopter project contains a SITL-feature. This means that one can choose to compile the project for SITL instead of for use with hardware. This feature is very practical both for testing changes done to the ArduCopter project and to test the way DUNE and ArduCopter interface each other. The ArduCopter code has been altered by the hexacopter team at AMOS to include a DUNE mode. This mode is verified using the SITL feature.

ArduPilot

To be able to test manual control in the SITL setup, ArduPilot hardware (running ArduCopter code) is included to send PWM values from a radio into the SITL.

MAVProxy

MAVProxy is a simple ground control station, but it is not used for that in this setup. The task for MAVProxy here is simply to be a bridge between the ArduCopter SITL and AMP Planner, which is another ground control station. This bridge is needed because the ArduCopter SITL use TCP to emulate a serial port used for communication while APM Planner connects using UDP.

APM Planner

APM Planner is a very versatile ground control station developed as an open source project. The version used in this setup is an altered version developed by the hexacopter group at AMOS to include DUNE mode. It is convenient to use this ground control station because it gives easy access to state values at the AarduCopter SITL and an simple interface to modify parameters used by the ArduCopter SITL. It can also be used to arm the ArduCopter SITL and to change mode.

7.2 Pickup Test Without Disturbances

The SITL-simulator has support for different types of disturbances, these disturbances could for instance be environmental disturbances like wind or sensor inaccuracies like drift or noise. The first SITL-test is conducted without any disturbances. The system has already been simulated without disturbances in the previous chapter, but another test will provide additional information on how the ArduCopter software and the control structure of DUNE functions together. Especially interesting moments to consider is how delays due to the interface and limited controller input rate between the modules will affect the accuracy of the controller.

Again it is the pickup phase that is tested. The hexacopter is flown manually in above the location of the simulated sensor node. Then the hexacopter will be switched into DUNE mode and the pickup will begin.

7.2.1 Results

Figure 7.2 shows the hexacopters positioning itself over the sensor node before beginning to descend upon it. Position errors are approximately ± two centimetres in the NE-plane, and the hexacopter is able to descend all the way down to the sensor node.

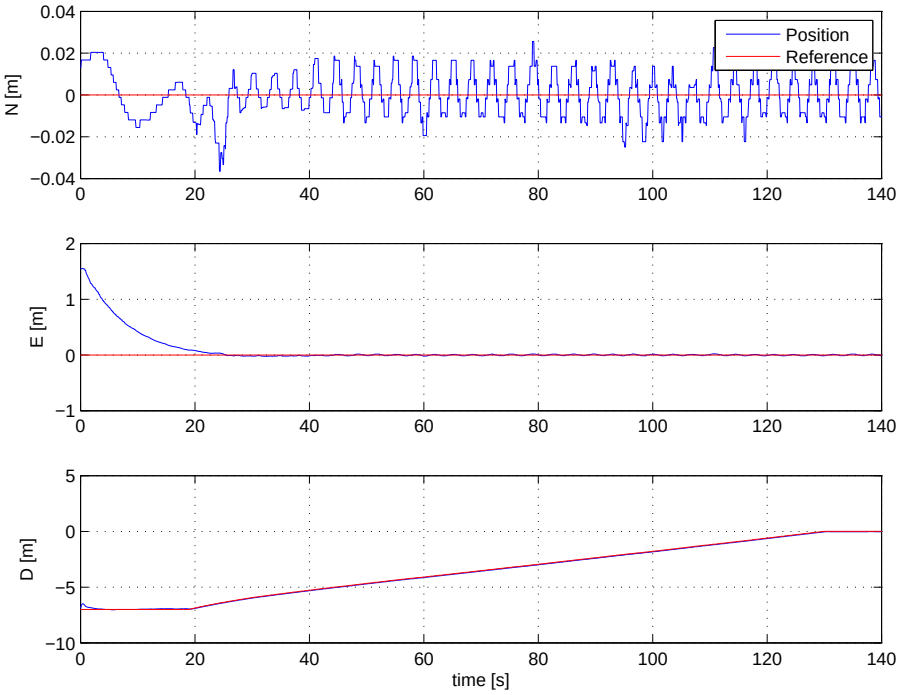


Figure 7.2: North-East-Down position of the hexacopter

Desired attitude versus measured attitude is plotted in Figure 7.3. One can see from the plots that the the magnitude of the measured attitude is a bit greater than the magnitude of the desired attitude and that the measured attitude is a bit delayed versus the desired.

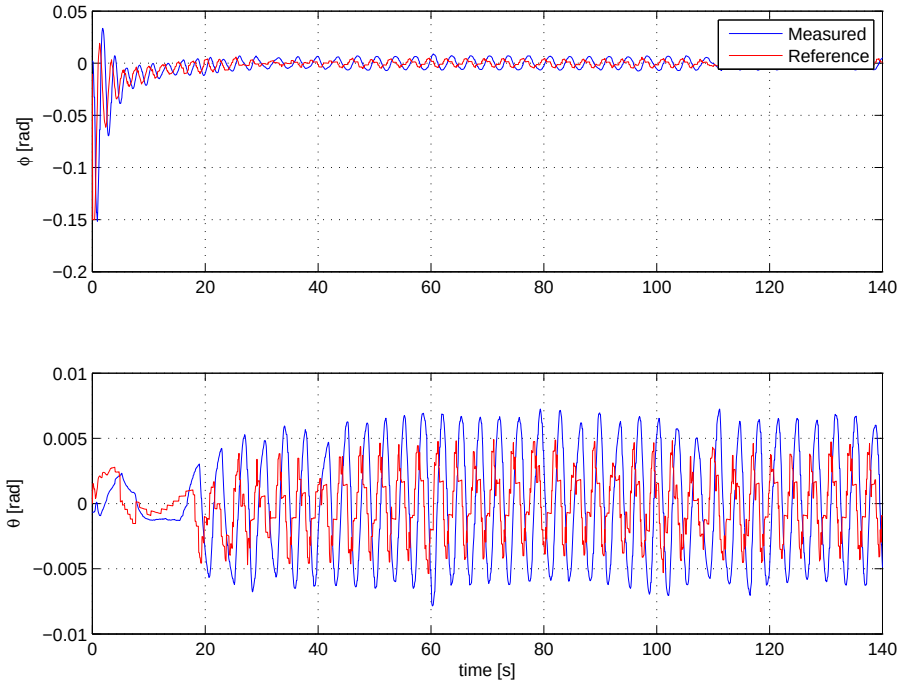


Figure 7.3: Attitude of the hexacopter

7.2.2 Discussion

The results show that the control structure is able to conduct a pickup of a sensor node, even though delays due to communications and inaccurate low level controllers of the APM makes small oscillations in the position of the hexacopter.

7.3 Pickup Test With Disturbances

The tests conducted in the previous section was able to verify the control structure, and showed promising results in relation to the communication between the APM and DUNE. But the real system will have many causes for inaccuracies that the test did not take into account. Hence another test was conducted to test the robustness of the system in a more realistic setting. The test is conducted by using simulation parameters that are part of the ArduCopterSITL to add drift and sensor noise to the system. Flight tests conducted have shown that drift is a major problem for the system, hence this is a realistic and important test.

7.3.1 Results

the position of the hexacopter above the sensor node can be seen in Figure 7.4. One can see from this figure that the position in the NE-plane oscillates a bit and that there is a small deviation in the position. The hexacopter is able to lower itself all the way down to the

sensor node even though it looses sight of the sensor node at some points when it is close to he ground. It is able to regain sight of the node because of the D-term in the controller that makes the hexacopter stay in the same area. The effect is shown Figure 7.5.

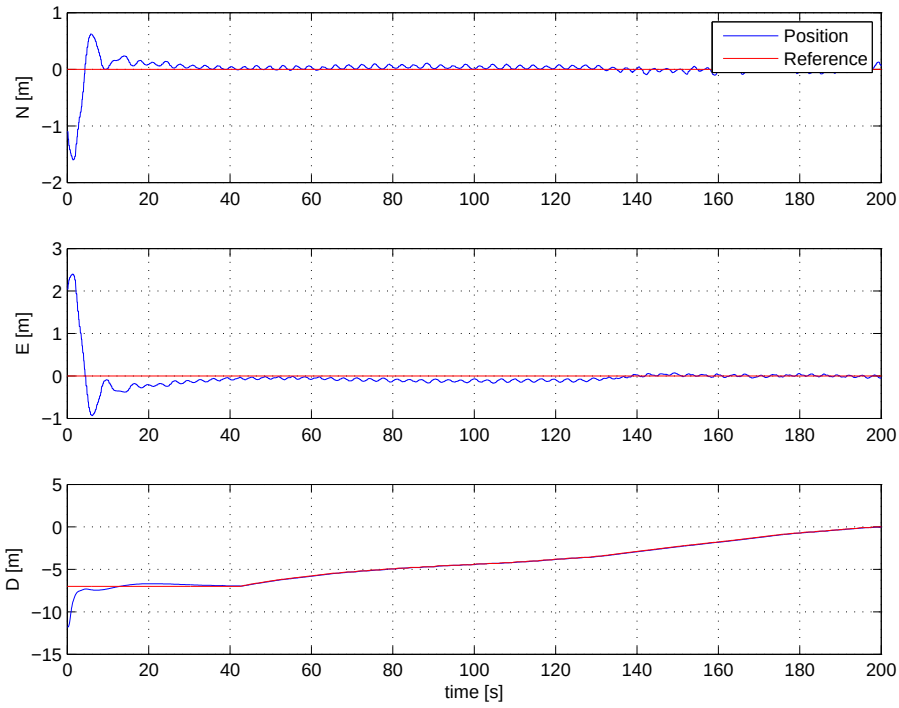


Figure 7.4: North-East-Down position of the hexacopter

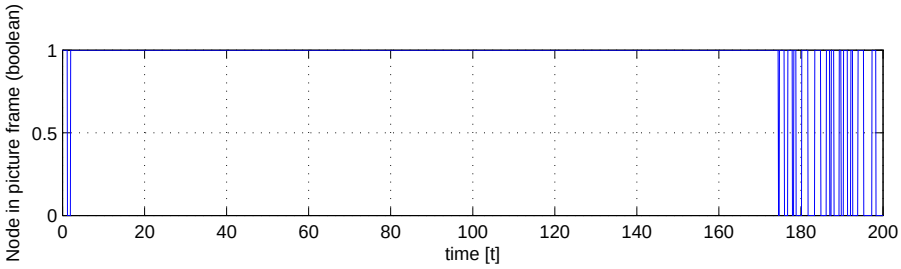


Figure 7.5: Showing whether the node is in the picture frame or not

The low level attitude controller follows the reference in approximately the same way as in the previous test.

Several more tests were conducted with the same settings. Some of them did not show as good performance as the displayed results. Some were not able to get closer to the node than 0.5 meters, before drifting off.

7.3.2 Discussion

The hexacopter is able to lower itself down to the sensor node even though the hexacopter is exposed to drift and sensor noise. There is a small deviation in position even though there is integral action in the controller. This is due to the fact that the integral term is limited to avoid integral wind up. And because the drift can change direction the limit on the integral term is quite low. It is considered to be better with a small deviation than facing the risk of huge overshoots due to big integral terms.

It is really promising that the hexacopter is able to stay still and regain sight of the hexacopter even though drift effects the system. But the simulated GPS speeds are probably more accurate than the GPS-measurements one can hope to get in the real world, so to be able to keep the node in sight at all times when descending is crucial for success.

As mentioned the results varied a bit, showing the unpredictability one faces when working with noisy sensors and drift. The simulated camera has the same field of view as the camera used for real life testing. There exists web-cameras with greater field of view than the one used. The tests was rerun with a camera with greater field of view to see if the success rate would be increased

Chapter 8

Testing and Results

8.1 Node Tracking

Points for node tracking test: - Equipment - Panda or Computer; Panda, computer for Neptus - Implementere sonar igjen - Sende bilder til Neptus / Dune - laser - kompass

8.1.1 Result of Test

Chapter 9

Discussion

Chapter 10

Conclusion

Bibliography

3DRobotics (2013). store.3drobotics.com/products/3dr-gps-ublox-with-compass.

Alaimo, A., Artale, V., Milazzo, C., Ricciardello, A., and Trefiletti, L. (2013). Mathematical modeling and control of a hexacopter. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 1043–1050.

Analog Devices (2005). Adg3300 datasheet. www.analog.com/static/imported-files/data_sheets/ADG3300.pdf.

ArduPilot (2013). Ardupilot development site. dev.ardupilot.com/.

Ardupilot (2013). Setting up flight modes. copter.ardupilot.com/wiki/flight-modes/.

Berberan-Santos, M. N., Bodunov, E. N., and Pogliani, L. (1997). On the barometric formula. *American Journal of Physics*, 65(5):404–412.

Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698.

Catsoulis, J. (2005). *Designing Embedded Hardware*. O'Reilly Media, second edition edition.

e-con Systems (2013a). Camera module features. www.e-consystems.com/OMAP4-MIPI-Camera-Board.asp.

e-con Systems (2013b). e-cam51_44x hardware user manual. www.e-consystems.com/doc_OMAP4_MIPI_Camera.asp.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 1 edition.

iGage (2013). Gps accuracy. www.igage.com/mp/GPSAccuracy.htm.

Jay Esfandyari, Roberto De Nuccio, G. X. (2010). Introduction to mems gyroscopes. electroiq.com/blog/2010/11/introduction-to-mems-gyroscopes/.

Liu, C. (2011). *Foundations of MEMS (2nd Edition)*. Prentice Hall, 2 edition.

LSTS (2014a). About lsts. http://lsts.fe.up.pt/about.

LSTS (2014b). Dune: Unified navigation environment. http://lsts.fe.up.pt/software/dune.

LSTS (2014c). Imc, inter-module communication protocol. http://lsts.fe.up.pt/software/imc.

LSTS (2014d). Neptus command and control software. http://lsts.fe.up.pt/software/neptus.

Mason, W. P. and Thurston, R. N. (1957). Use of piezoresistive materials in the measurement of displacement, force, and torque. *The Journal of the Acoustical Society of America*, 29(10):1096–1101.

Mclennan (2013). Geared dc instrument motor 1271 series. www.farnell.com/datasheets/526521.pdf.

Merriam Webster (2013). Definition barometer. www.merriam-webster.com/dictionary/barometer.

MIPI (2013). Camera interface specifications. www.mipi.org/specifications/camera-interface#CSI2.

Mordvintsev, A. and Abid, K. (2013). Introduction to surf (speeded-up robust features). opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html.

omappedia.org (2011). Pandaboard test data. omappedia.org/wiki/Panda_Test_Data.

omappedia.org (2012). Pandaboard faq. omappedia.org/wiki/PandaBoard_FAQ.

OpenCV (2013). Introduction to sift (scale-invariant feature transform). docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html.

OpenCV.org (2013). Cascade classifier training. docs.opencv.org/doc/user_guide/ug_traincascade.html.

pandaboard.org (2011). Pandaboard es system reference manual. pandaboard.org/sites/default/files/board_reference/ES/Panda_Board_Spec_DOC-21054_REV0_1.pdf.

pandaboard.org (2013). Pandaboard es technical specs. pandaboard.org/content/platform.

pololu.com (2014). www.pololu.com/product/2136.

QGroundControl (2013). Mavlink micro air vehicle communication protocol. qgroundcontrol.org/mavlink/start.

Sørensen, A. (2013). *Marine Control Systems - Propulsion and Motion Control of Ships and Ocean Structures*.

Spong, M. W., Hutchinson, S., and Vidyasagar, M. (2005). *Robot Modeling and Control*. Wiley, 1 edition.

Store Norske Leksikon (2013). Definition magnetometer. `snl.no/magnetometer`.

Texas Instruments (2013). Omap^{tm} 4 processors. `www.ti.com/lsds/ti/omap-applications-processors/omap-4-processors-products.page`.

u blox (2012). Datasheet lea-6 series. `www.u-blox.com/images/downloads/Product_Docs/LEA-6_ProductSummary_%28GPS.G6-HW-09002%29.pdf`.

Vik, B. (2012). *Integrated Satellite and Inertial Navigation Systems*. Deperatment of Engineering Cybernetics, NTNU.

Voldsund, V. (2013). *Mechanisms for Drop and Recovery of Sensor Nodes Using UAVs*. Deperatment of Engineering Cybernetics, NTNU.