Drop and Recovery of Sensor Nodes Using UAVs

Vegard Voldsund

Master of Science in Cybernetics and Robotics
Submission date: June 2014
Supervisor: Tor Arne Johansen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

NTNU **Norwegian University of** **Science and Technology**	**Faculty of Information Technology,** **Mathematics and Electrical Engineering** **Department of Engineering Cybernetics**

MSC THESIS DESCRIPTION SHEET

Name:	Vegard Voldsund
Department:	Engineering Cybernetics
Thesis title (Norwegian):	Slipp og gjennvinning av sensornoder ved bruk av UAVer (unmanned aerial vehicles)
Thesis title (English):	Drop and recovery of sensor nodes using UAVs (unmanned aerial vehicles)

Thesis 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. Design and create one mechanical mechanisms for sensor deployment and pickup to be mounted on a hexacopter.
3. Study how the multicopter should approach the target for pickup/deployment using the created mechanism, and create control-laws capable for this maneuver.
4. Study and implement camera-based methods for node tracking needed for pickup.
5. The control-implementation should be conducted in a suitable framework. (E.g. Dune)
6. The results should be verified by experiments.
7. Conclude findings in a report. Include Matlab/C-code as digital appendices.

Start date:	2014-01-20
Due date:	2014-06-16

Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Professor Tor Arne Johansen, Dept. of Eng. Cybernetics, NTNU
Co-supervisor:	MSc Kristian Klausen, Dept. of Eng. Cybernetics, NTNU

Abstract

The goal of this project is to make a system able of executing drop and recovery of sensor nodes at sea by the use of UAVs.

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.

This project is a continuation of a project executed by the author [Voldsund, 2013] where a drop and recovery mechanism to be mounted on a multicopter UAV was developed. The mechanism has been further developed in this project and integrated to an UAV.

A control structure for the UAV has been developed and tested using simulator, software-in-the-loop tests and tests in the lab. Simulations and software-in-the-loop tests showed that the control structure was able to execute drop and recovery. But the needed accuracy for recovery of sensor nodes using the mentioned mechanism was not achieved in the lab tests. Measures that could be taken to enhance accuracy to make the system able to conduct drop and recovery of sensor nodes are discussed in this thesis.

Sammendrag

(Norwegian translation of the abstract)

Målet med dette prosjektet er å lage et system som kan utføre slipp og gjenvinning av sensornoder til sjøs ved hjelp av UAVer.

Sensornoden vil være en lett enhet som kan inneholde forskjellig sensorer. Eksempler på bruk kan være logging av temperatur, havstrømmer, saltinnhold eller vannkvalitet. De kan med andre ord være veldig nyttige til for eksempel klimaforskning eller deteksjon av oljeutslipp.

Dette prosjektet er en videreføring av et prosjekt utført av forfatteren [Voldsund, 2013] hvor en mekanisme for slipp og gjenvinning for bruk montert på en UAV ble utviklet. Denne mekanismen har blitt videre utviklet og integrert med en UAV i dette prosjektet.

En kontrollstruktur for en UAV har blitt utviklet og testet ved hjelp av simuleringer, software-in-the-loop tester og tester på laben. Simuleringene og software-in-the-loop testene viste at kontrollstrukturen er kapabel til å utføre slipp og gjenvinning av sensornoder. Men den nødvendige presisjonen for slipp og gjenvinning av sensor noder ved bruk av den nevnte mekanismen ble ikke oppnådd i lab-testene. Tiltak man kan utføre for å øke presisjonen og gjøre systemet kapabelt til å utføre slipp og gjenvinning av sensornoder er diskutert i denne rapporten.

Preface

This project is executed as a final part of my MSc degree at the Department of Engineering Cybernetics at NTNU. The project is part of research conducted by the Center of Autonomous Marine Operations and Systems (AMOS). To be a part of the research team at AMOS and especially the Hexacopter team has been rewarding in terms of good cooperation and knowledge exchange.

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 MSc Kristian Klausen for guidance and encouragement along the way.

Vegard Voldsund

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 available 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 are 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 are also according to AMOS standards.

This master project is a continuation of a project carried out in the fall of 2013 by the author. The goal of that project [Voldsund, 2013] was to create a mechanism that can be used for sensor node drop and recovery by the use of multicopters. In the mentioned project, different design solutions were discussed and evaluated before one solution was implemented. This implemented design solution is the basis of this project. In this project the mechanism will be further improved and put into a fully operating system with the UAV. Control software for the UAV will be developed and different steps towards the goal of sensor node drop and recovery by the use of a UAV and the mentioned mechanism will be taken.

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 deployment 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.

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 [Mellinger et al., 2011].

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 [Jiang and Voyles, 2013] and [Lippiello and Ruggiero, 2012]. 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 [Thomas et al., 2013]. 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 [Pounds and Dollar, 2014]. 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 [Mellinger et al., 2011] 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.

1.3 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 this 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.

[1]A very accurate external motion capture system

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

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 and some basic rigid-body kinetics are given in Chapter 2.

The UAV used in this project is described in Chapter 3. 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 and some relevant software are also described for the same reasons.

The pickup and deployment mechanism is presented in Chapter 4. The chapter starts with an overall description, before going more detailed into both mechanical and electrical design.

Chapter 5 presents the control structure designed. It covers controller design, node tracking using camera, communication between the different modules and a brief description of the developed software.

To verify the controller design a simulator running in Simulink was developed. Chapter 6 presents the mathematical model the simulator is built upon, and a description of the different elements that have been simulated. Results from different simulation scenarios are presented and discussed.

To be able to verify the control structure using DUNE and ArduPilot Mega 2.6 together, software-in-the-loop tests were conducted. The setup of these test are given in Chapter 7. The results from the different tests are presented and discussed.

Final tests are conducted in the lab, the different scenarios tested accompanied with the results of the tests and some discussion of the results are given in Chapter 8.

A general discussion of the design that take into account the results from all the different tests and test setups are given in Chapter 9, before conclusions are drawn in Chapter 10.

The digital appendix included with this report contains all the source code used in the control structure. The simulator developed in Simulink and Solidworks models of the 3D-printed parts are included as a resource. Videos of the tests conducted in the lab are also included to give the reader a better overview of the results.

Chapter 2

Background Theory

This chapter gives a short introduction to different reference frames used for navigation and control. Some basic rigid-body kinematics and computer vision theory are also presented.

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.

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

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

Where $\boldsymbol{R}^e_n(\boldsymbol{\Theta}_{en})$ 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

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

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.

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

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

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

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).

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

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

Calculate Distance and Bearing Between Two ECEF-Locations

The position of a vehicle is in navigation often given as a longitude and a latitude value. To be able to use this values in a feedback for a position controller, it could be convenient to be able to calculate the distance and bearing between the current point and a reference point. This could be done in multiple ways, depending on the required efficiency and accuracy of the solution. Different algorithms do also have different performances based on how long the distance between the points is, i.e. some take the curvature of the earth into account, some do not. Those which does not consider curvature of the earth are useless over long distances but can be very effective and accurate over short distances.

In this project only short distances are considered, hence an effective approach using Pythagoras theorem on an equirectangular projection given by [Veness, 2014] is used and rendered here. An equirectangular projection is a map projection where the earth is given as a flat surface with equally spaced parallel longitude lines. These calculations are inaccurate, but they are sufficiently good for the applications they are used for in this project.

The distance between the two GPS-positions is calculated by equations (2.14) to (2.16).

$$x = \Delta l \cos(\mu_m) \tag{2.14}$$

$$y = \Delta\mu \tag{2.15}$$

$$d = R\sqrt{x^2 + y^2} \tag{2.16}$$

Here x and y is the position in the equirectangular projection, which is calculated using the differences in longitude Δl and latitude $\Delta\mu$ and the mean value of the latitude ϕ_m. R is the radii of the earth given in kilometres (6371 km). The bearing β from point one to point two is given by equation (2.17).

$$\beta = atan2(\sin(\Delta l \cos(\mu_2), \cos(\mu_1)\sin(\mu_2) - \sin(\mu_1)\cos(\mu_2)\cos(\Delta l)) \tag{2.17}$$

Combining the distance and the bearing between the two points gives all the needed information to use in for instance a feedback loop of a position controller.

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 $\boldsymbol{A}_i$ (the transformation matrix from reference system $i-1$ to reference system i) is represented as a product of four basic transformations

$$\begin{aligned}
\boldsymbol{A}_i &= \boldsymbol{Rot}_{z,\theta_i}\boldsymbol{Trans}_{z,d_i}\boldsymbol{Trans}_{x,a_i}\boldsymbol{Rot}_{x,\alpha-i} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2.18}$$

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.

The transformation matrix $\boldsymbol{T}_j^i$ 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

$$\boldsymbol{T}_j^i = \begin{cases} \boldsymbol{A}_{i+1}\boldsymbol{A}_{i+2}...\boldsymbol{A}_{j-1}\boldsymbol{A}_j & \text{if } i < j \\ I & \text{if } i = j \\ (\boldsymbol{T}_i^j)^{-1} & \text{if } j > i \end{cases} \tag{2.19}$$

$\boldsymbol{T}_j^i$ 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.

$$\boldsymbol{T}_j^i = \begin{bmatrix} \boldsymbol{R}_j^i & \boldsymbol{o}_j^i \\ 0 & 1 \end{bmatrix} \tag{2.20}$$

2.2 Rigid-Body Kinetics

The motion of rigid bodies can be expressed according to [Fossen, 2011] as shown in equation 2.21.

$$\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}_{RB} \tag{2.21}$$

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 2.22.

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_g \end{bmatrix} \quad \boldsymbol{C}_{RB} = \begin{bmatrix} m\boldsymbol{S}(\boldsymbol{\omega}) & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & -\boldsymbol{S}(\boldsymbol{I}_g\boldsymbol{\omega}) \end{bmatrix} \tag{2.22}$$

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 - 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 [OpenCV.org, 2013a]. 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 webcamera 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.

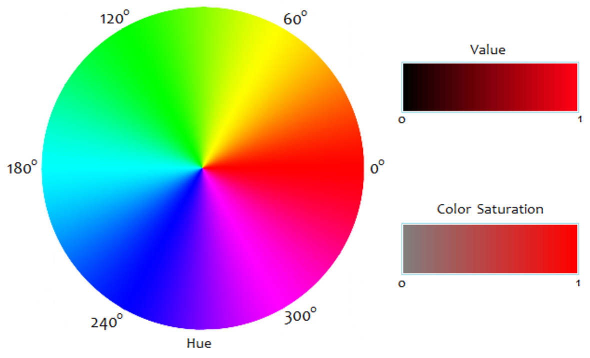


Figure 2.1: HSV-color wheel *Courtesy of had2know.com*

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, 2013b]. 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, 2013b].

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.

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. Some highly relevant software for this project are also briefly described.

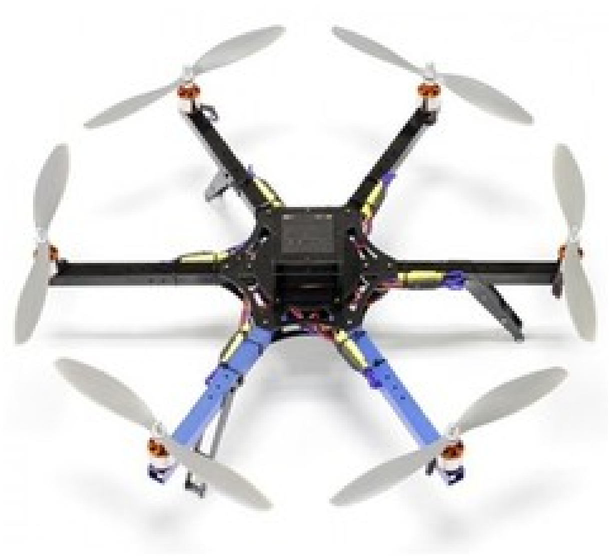


Figure 3.1: ArduCopter Hexacopter *Courtesy arducopter.co.uk*

3.1 APM

The APM is an open source flight control unit supporting multicopters, traditional helicopters, fixed wing aircraft and rovers [ArduPilot, 2013]. The software running on the APM in this project is the ArduCopter project, which is the software solution used for multicopters. Software is developed and supported by the DIYDrones community. At the moment the community has more than 50 000 members (June 2014) 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

In addition to these modes, a DUNE mode has been developed by the Hexacopter group at AMOS. This is a mode where setpoints for the controllers on the APM can be set from an external device, i.e. PandaBoard. 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 and an external magnetometer and GPS module is found in Figure 3.2.

[1]Pulse Width Modulated signals

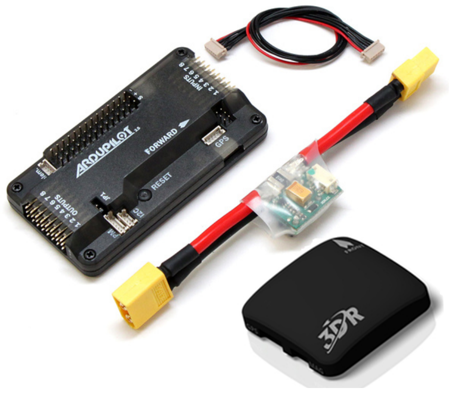


Figure 3.2: APM 2.6 with a voltage regulator and an external magnetometer and GPS module *Courtesy diydrones.com*

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 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

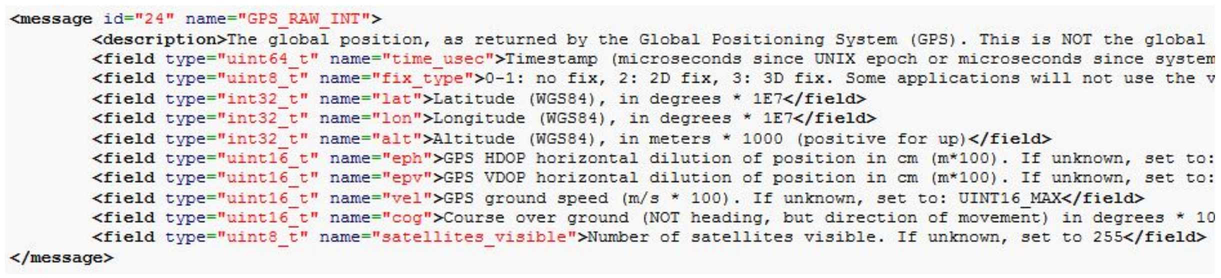
The APM has several interfaces that make it flexible and suitable for research and development. It has dedicated connection points for GPS and telemetry. These are interfaced using UART. It also has an unused UART-port available for other units and applications. Input from the radio and output for the 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 has 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

[4]Micro Air Vehicle Communication Protocol

[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.

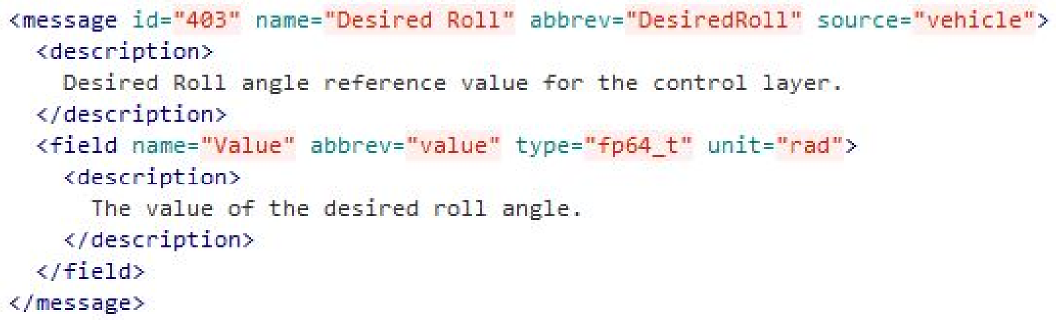
3.3 Software

The software developed in this project is build upon and integrated into IMC, DUNE 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 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.

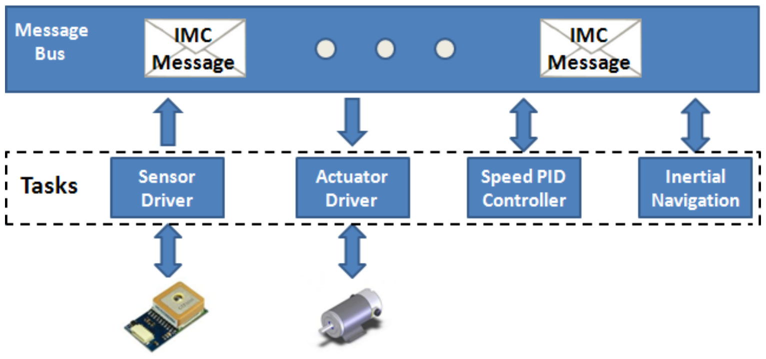


Figure 3.5: Task interaction using IMC *Courtesy lsts.fe.up.pt*

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 a configuration file is used. Relevant task parameters, vehicle types etc. are defined in the configuration file. 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

Drop and Recovery Mechanism

The aim of the project leading up to this master project was to explore different mechanisms that could be used to facilitate drop and recovery operations of sensor nodes using UAVs. 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 project. Readers are advised to consult the project report for discussions concerning the design solution of the drop and recovery mechanism, while the resulting, though improved mechanism, is presented in this chapter.

4.1 Overall Description

To get sufficient accuracy during recovery it was decided that a solution where a gripper can be lowered down towards the sensor node was the best 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. It was designed in such a way that different equipment can be mounted to it, and it is used in several of the projects conducted by the AMOS hexacopter-group. The box is displayed in Figure 4.1.

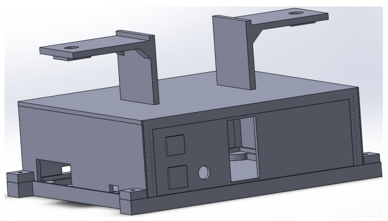


Figure 4.1: Box to contain the PandaBoard

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 4.2. 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 description of the mechanical design of the gripper and lowering mechanism is presented in Section 4.2.

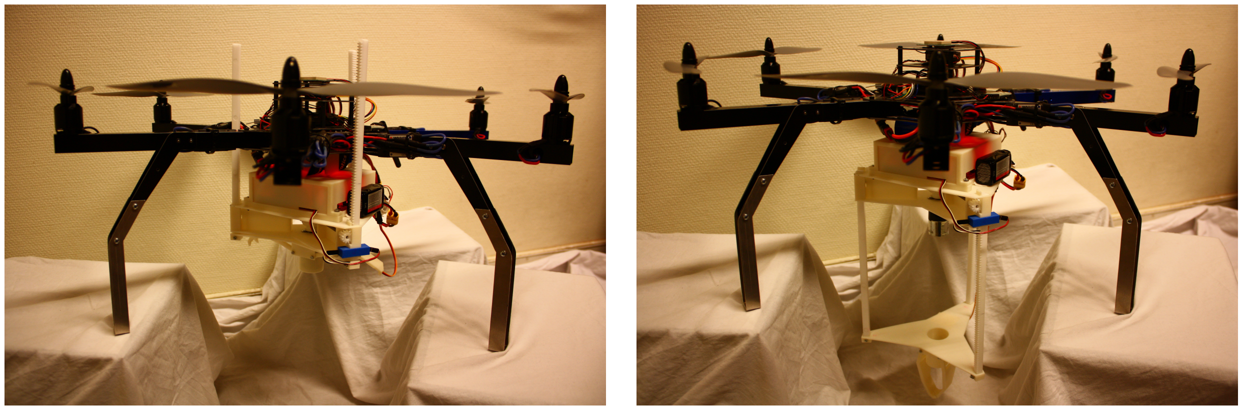

(a) Gripper raised (b) Gripper lowered

Figure 4.2: Drop and recovery mechanism mounted to the hexacopter

4.2 Mechanical Design

To be able to use gears to push the platform up and down, rack gears were used. The rack gears were fastened to the gripper platform at one end, while the other end went up through the gear platform. To be sure that all the rack gears move at the same time and with the same speed, the same motor was used to drive them all. This meant that a transmission needed to be designed. The platform for the gears and transmission was 3D-modelled and 3D-printed, and the gears was mounted. This can be seen in Figure 4.3.

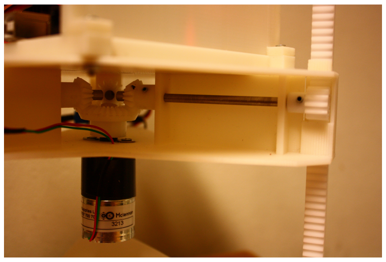


Figure 4.3: Gear

A DC motor was chosen to drive the gears. Alternatively a servo could have been used, but that would have been more spacious in a position where space really matters. It would require a lot of torque to drive the transmission and lift the gripper platform. Hence torque was the most important parameter in the process of choosing a motor. The motor chosen was a 1271 series 188:1 geared DC motor from Mclennan able of providing a torque of 14 Ncm at the same time as its small in size and relatively light weighted. The fact that it is geared increases the torque at the same time as it decreases the speed of the motor. The rated speed is 9 rpm, which means that it will use approximately 20 seconds to raise or lower the gripper platform. This is reckoned to be within acceptable time consumption. To be able to control the DC motor a H-bridge was needed. The chosen H-bridge is presented in Section 4.3.2.

Tests of pickup weight limitations was conducted in [Voldsund, 2013]. This tests showed that the mechanism was able to lift weights of approximately 1 kg. Time consumption increased with the weight, but speed was quite hight when lifting the targeted sensor node weight of 200 g.

To know when the gripper platform is in the extreme positions a measurement of the platforms position is needed. How this was solved is explained in Section 4.3.1.

Gripper

When designing the gripper mechanism, several points needed to be considered. It should be as simple, robust and power efficient as possible. At the same time it should not build too much below the UAV and it should be made sure that the camera gets free sight.

The gripper was designed in a way that it could be controlled using only one servo. This was possible by using gears. The gripper was 3D-modelled and 3D-printed. See Figure 4.4 to understand how the gripper parts interact. The servo used was a MG-14 from Hextronik. This servo was chosen due to the fact that it is relatively light and sufficiently strong, while being metal geared, which is good for robustness.

The gripper needed to be mounted on a platform that is connected to the rack gears to be able to raise and lower the gripper. The platform is supposed to be raised up as close to the UAV as possible. The motor connected to the gear platform sticks down below the gear platform and is in the way for the gripper platform. This was solved by designing a hollow cylinder on the gripper platform for the motor to fit into when the gripper platform is raised. The cylinder is there to have a position to mount the camera that is centred and have free sight down while searching for the node. This placement of the camera also makes it possible to use the camera to verify whether the gripper has caught the node or not. The gripper platform also needed a mount for the rack gear, the gripper and the servo for the gripper. Also the gripper platform was 3D-modelled and 3D-printed.

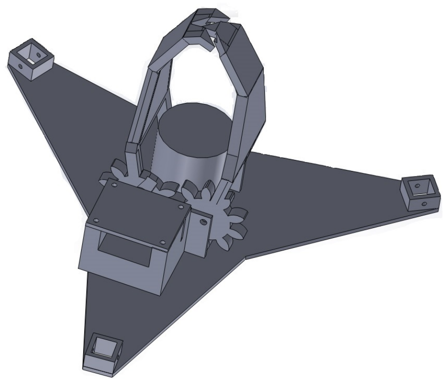


Figure 4.4: Gripper mounted on the gripper platform

4.3 Hardware

The pickup and deployment mechanism consists among other things of some hardware to be able to control the motor and the servo and to be able to be interfaced to the PandaBoard. Some of this hardware was store bought and some was designed and produced as a part of this project.

To be able to know when the gripper platform is in its extreme positions (fully raised or fully lowered), a position sensor based on IR-LEDs and photodiodes was developed. The servo of the gripper is controlled by a PWM signal. To make sure that the PWM signal is accurate enough some circuitry containing a micro controller was developed. The PandaBoards I/O-pins operate at a logical voltage level of 1.8 V while other parts of the hardware used in this project operates at 5 V. This makes a translation necessary for the PandaBoard to be able to communicate with the rest of the hardware. This was also designed. The reasoning and design of these hardware components in addition to descriptions of a store bought H-bridge and sonar are included in the following sections.

4.3.1 IR-LED Position Sensor

Holes where drilled in the rack gears and the IR-LED position sensors were placed in such a way that when the platform is in position the IR light shines through the hole and a photodiode can sense the light on the other side. Two position sensors were mounted, one for raised position and one for lowered position. The position sensor mounted using a 3D modelled and 3D printed mount can be seen in Figure 4.5.

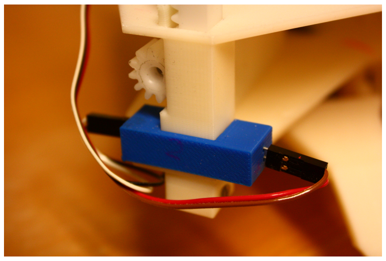


Figure 4.5: Mounting of IR-LED position sensor

The key components of the IR-LED position sensor is the IR-LED and the photodiode. These need some support circuitry in order to function. The circuit with the LED is fairly simple. It consists 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 normally not conduct, 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 senses. In this application the LED and the photodiode are mounted in a way that the photodiode only senses the IR when the gripper platform is in position. Hence the signal could be connected directly to a digital I/O port to give either 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 in order 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 4.6.

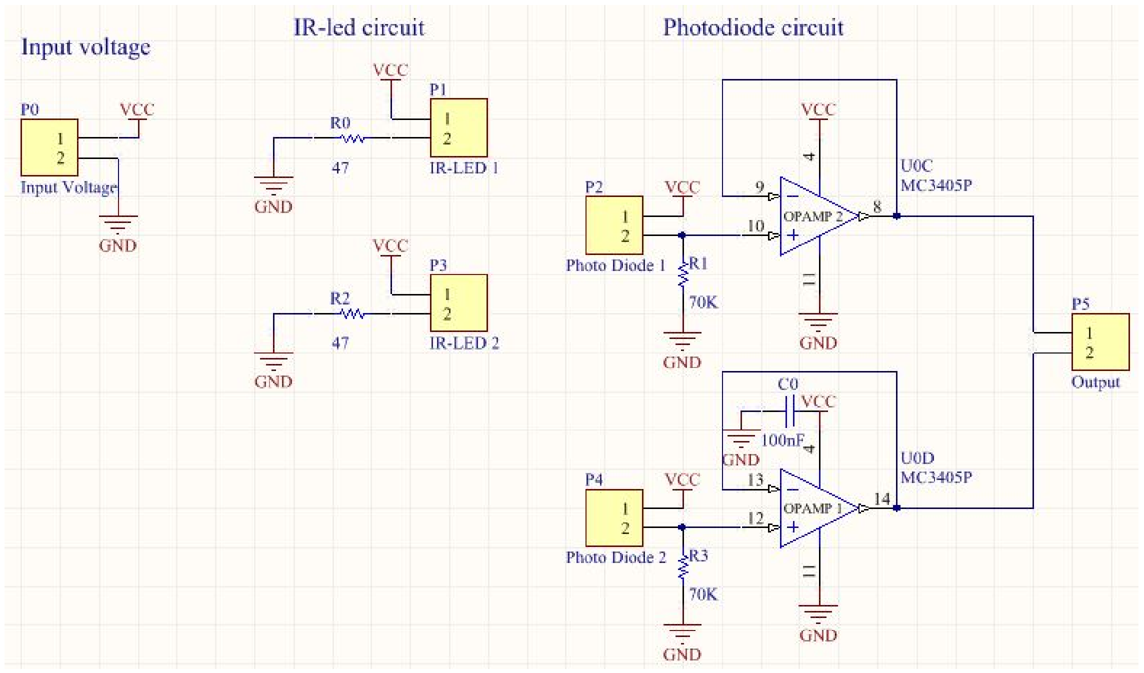


Figure 4.6: Schematics for the IR-LED Position Sensor

4.3.2 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 can 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.

4.3.3 PWM-Controller

The servo mounted to the gripper is controlled by using a PWM signal. The pulse width of the signal is translatable to the servos position reference. In the servo used the time period for the PWM signal is 20 ms. A pulse width of 1 ms mean 0 degrees reference, and 2 ms mean 180 degrees reference. 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 signals 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 the gripper will be controlled through DUNE which further increases difficulties with timing as DUNE has its own scheduling as well.

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 (In System Programming). To be able to program the microcontroller in circuit, headers are connected to all the pins of the microcontroller, 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 is 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 4.7, while the code running on the ATtiny85 is included in the digital appendix.

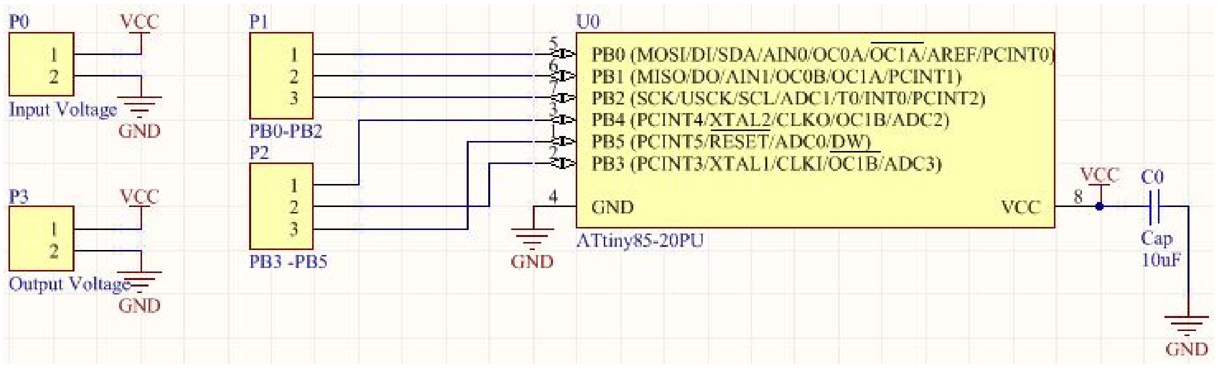

P0
VCC
Input Voltage
GND
P3
Output Voltage
P1
PB0-PB2
P2
PB3 -PB5
U0
PB0 (MOSI/DI/SDA/AIN0/OC0A/OC1A/AREF/PCINT0)
PB1 (MISO/DO/AIN1/OC0B/OC1A/PCINT1)
PB2 (SCK/USCK/SCL/ADC1/T0/INT0/PCINT2)
PB4 (PCINT4/XTAL2/CLKO/OC1B/ADC2)
PB5 (PCINT5/RESET/ADC0/DW)
PB3 (PCINT3/XTAL1/CLKI/OC1B/ADC3)
ATtiny85-20PU
C0
Cap
10uF

Figure 4.7: Schematics for the PWM-Controller

4.3.4 Level Translator

Communication will go both to and from the PandaBoard. The level translator will need at least six channels: two for controlling the H-bridge, two for servo control 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. The desired low voltage level and low voltage signals are connected at one side and the desired hight voltage level and high voltage signals are connected at the other side. Two decoupling capacitors are also connected. The schematics of the circuit design can be seen in Figure 4.8.

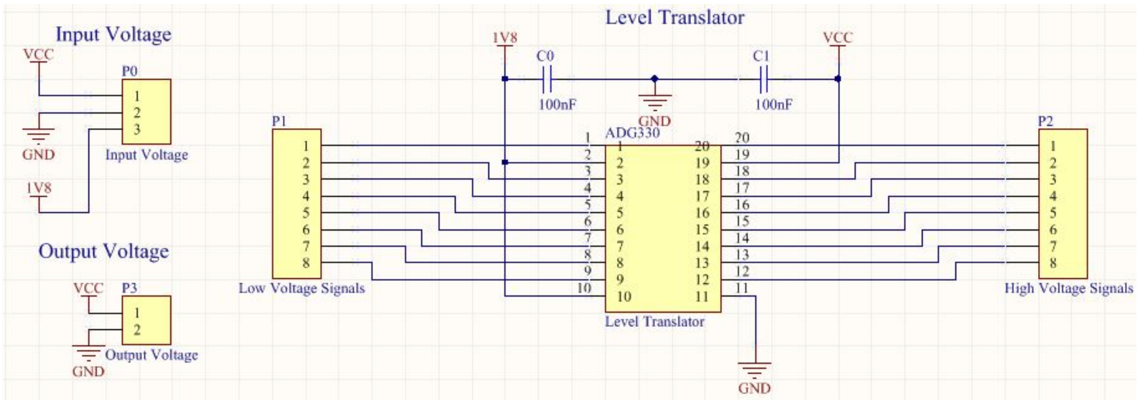


Figure 4.8: Schematics for the Level Translator

4.3.5 Sonar

Accurate height measurements are critical in the pickup of the sensor node. They are both important to avoid crashing into the water and to get reliable measurements of the position of the sensor node by the use of camera. The APM has barometric sensors to measure height, in addition to height measurements from GPS. Neither of these measurements are reliable enough to by used for this application.

There exists several altitude measurement principles that could be used for this application. The most relevant are ultrasound and laser. They both have different pros and cons. A more in depth analysis of the pros and cons are given in [Voldsund, 2013], but the key points are rendered here.

Laser range finders can be used for much longer ranges than ultra sound range finders (sonars), but the fact that this application is meant to be used at sea posts a problem with lasers. The laser beam can travel through the water and calculate the distance to the sea floor instead of to the surface. This problem is strongly reduced with sonars. The UAV will be flying low while searching for the sensor node, so the range of a sonar will be sufficient.

The chosen sonar for this application is the XL-MaxSonar-EZ4 from MaxBotix. It has several interfaces that can be used to read out the altitude measurements. The interface used in this application is an analogue voltage between 0 and 5 V. This interface is used because the APM has built in functionality to read sonar values using its internal ADC. To avoid noise in the signal from the sonar, the signal is low pass filtered using a simple RC-filter with a resistor value of 10 Ω and a capacitor of 100 μF and a shielded cable is used for the signal. The range of the sonar spans from 20 cm to 765 cm and altitude measurements are read at a 10 Hz rate with a resoltion of 1 cm [MaxBotix, 2014]. This is a perfect range for this application. MaxBotix has several similar sonars available. The main difference between them is the beam angle. The EZ4 has the narrowest beam, and is chosen because of that. The narrow beam makes it possible to mount the sonar on the UAV and still be able to avoid measuring the distance to the lowered gripper platform instead of the distance to the sea surface or the ground.

4.4 Communication Between the Modules

An overview of the different modules and the communication between them is shown in Figure 4.9.

The camera is connected to the PandaBoard using USB. The APM is also connected by the use of an USB port, but 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.

The rest of the communication to and from the PandaBoard goes via the level translator described in the previous section.

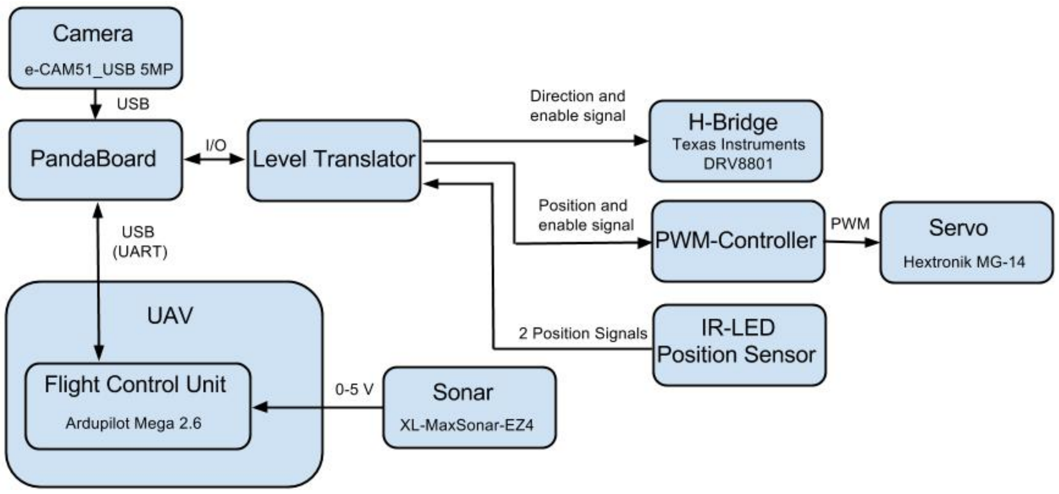


Figure 4.9: Communication between the modules

4.5 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. Hence the safest 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.

Selecting battery is a trade-off between capacity and weight. The battery should be able to power both the motor and the ICs[1]. 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 are able to hold a lot of power in relation to its weight. The specific battery chosen is a 1000 mAh 20-30 C (this 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 can be replaced by a heavier one with greater capacity if necessary. But the specifications of the chosen battery seems reasonable according to power demand calculations conducted in [Voldsund, 2013].

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. Especially the servo can be a cause for ripple in the delivered voltage and current. These choices result in the power circuit for the drop and recovery mechanism as displayed in Figure 4.10.

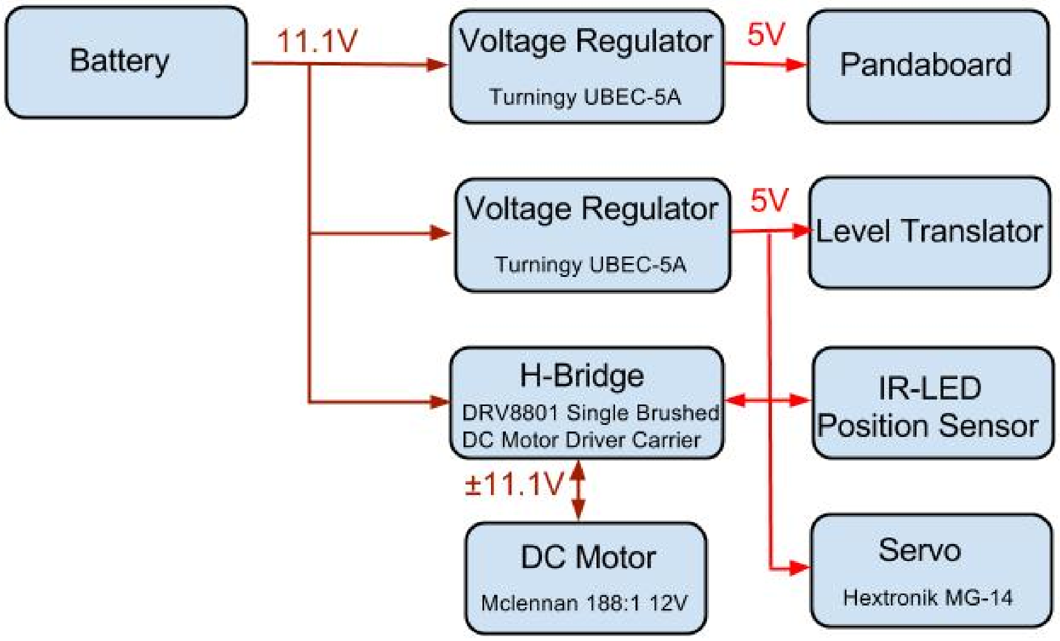


Figure 4.10: Power circuit

[1]Integrated Circuits

Chapter 5

Control Structure

5.1 Controllers

A position controller (controlling North and East positions) for the UAV 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. A waypoint tracking algorithm using a velocity controller was also developed to be used for long distance movements. In addition to the position and speed controller, both a altitude controller and heading controller were used.

5.1.1 Position Controller

To be able to use Newtons second law of motion the forces $\boldsymbol{F}^n$ acting upon the UAV in the NED frame are derived. 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).

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

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

$$\boldsymbol{F}^n(\boldsymbol{\Theta}, T) = T \begin{bmatrix} -s_\psi & -c_\psi \\ c_\psi & -s_\psi \end{bmatrix} \begin{bmatrix} \phi \\ \theta \end{bmatrix} \tag{5.2}$$

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

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

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 $\boldsymbol{\eta} = [N \;\; E]^T$. The resulting force mass-spring-damper system is expressed in equation (5.6). The fact that the controller turns the system into a forced mass-spring-damper system implies stability of the controller.

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

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

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

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. Here ω_0 is the natural frequency and ξ is the damping ratio. The damping ratio can be set to 1 to create a critically damped system, and the natural frequency can be tuned to give the desired response.

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

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

The augmented controller then becomes

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

Setpoints for roll and pitch are limited to ensure safe operation. This also makes sure that the small angle approximation is valid.

5.1.2 Altitude and Heading Controllers

The altitude and heading controllers are both fully dependent on low level controllers in the APM. The reference altitude and heading are sent to the APM. The desired values are typically found by ramping towards a desired value.

5.1.3 Waypoint Tracking

The position controller should for several reasons only be used to move short distances. For large distances the controller will reach angle saturation immediately and the UAV will move faster and faster. The high speeds will lead to massive overshoots when reaching the desired position. The angle saturation will also mean that the UAV not necessarily will fly the shortest distance to the desired location, but move in a path with a kink in it. To avoid this and have full control of the speed even for long distances a velocity controller was developed. The derivation of the velocity controller follows the same logic as the position controller. Equation (5.3) describes the system and the control input to create a PI velocity controller is given in equation (5.9).

$$\boldsymbol{u} = \boldsymbol{B}^{-1}(\boldsymbol{K}_p(\dot{\boldsymbol{\eta}}_r - \dot{\boldsymbol{\eta}}) + \boldsymbol{K}_i \int_0^t (\dot{\boldsymbol{\eta}}_r - \dot{\boldsymbol{\eta}})) \tag{5.9}$$

The desired velocity $\dot{\boldsymbol{\eta}}_r$ is calculated by using the algorithm in Section 2.1 to find the bearing β to the next waypoint. And by the use of the predefined desired speed reference $|\dot{\boldsymbol{\eta}}|_r$ as shown in equations (5.10) and (5.11).

$$\dot{N}_r = |\dot{\boldsymbol{\eta}}|_r \sin(\beta) \tag{5.10}$$

$$\dot{E}_r = |\dot{\boldsymbol{\eta}}|_r \cos(\beta) \tag{5.11}$$

When the UAV reaches a waypoint, operation depending on the current mission is activated. This means that the UAV will execute either pickup, drop or landing. Then the UAV flies to the next waypoint, this goes on until all the waypoints is visited and their associated mission is executed. A more sophisticated operation for the phase where the UAV reaches a waypoint should be developed as future work. For instance should a search pattern be executed to find the sensor node when the UAV approaches a pickup waypoint. The pickup waypoint will only gives an approximate position of the sensor node.

5.2 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 gets hold of the sensor node or not (one could have used estimation of inertia for this task like [Mellinger et al., 2011], 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. 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 with more distinct features, but because it is quite small and it should be possible to detect at a distance this method was discarded. A classifier could have been developed, but it would have weaknesses when it comes to rotation, and it would need a lot of example pictures to make it robust. For this reasons the classifier solution was discarded as well. The much simpler solution of color detection was implemented.

The camera used in this application is the e-CAM51_USB 5MP camera from e-con Systems. According to the producer it is able to provide HD video at 30 fps, and has a 60° field of view.

5.2.1 Tracking the Sensor Node

To be able to use color detection the sensor node needs to have some colors that sticks out. The mount for the sensor node (Figure 5.1) 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.

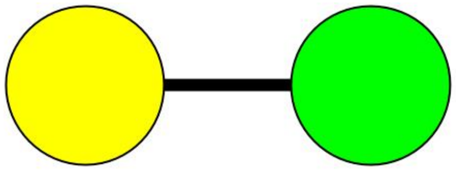
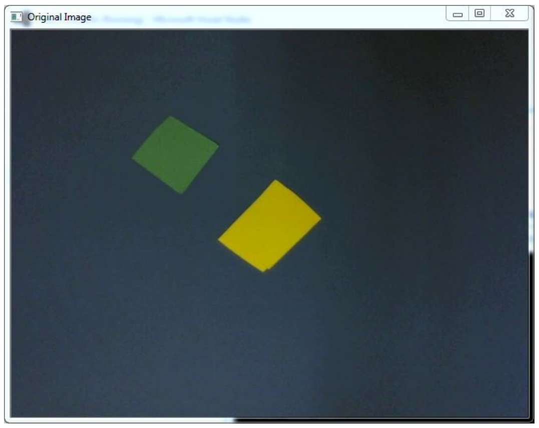
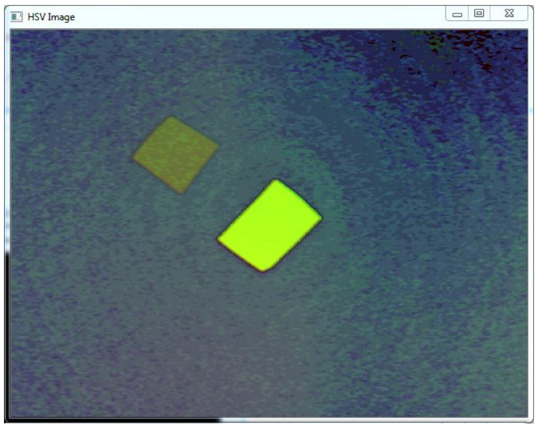


Figure 5.1: Mount for sensor node

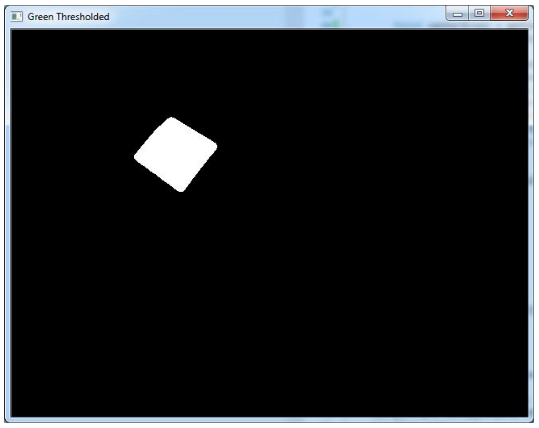
The picture from the camera is converted to the HSV-color space and copied to create two pictures. One of the pictures is thresholded with the HSV-values for yellow, while the other is thresholded with the values for green. This creates two binary pictures where the yellow and green fields are marked 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 5.2. For the measurements to make sense one need a mapping from this pixel position to the position of the mount in NED.



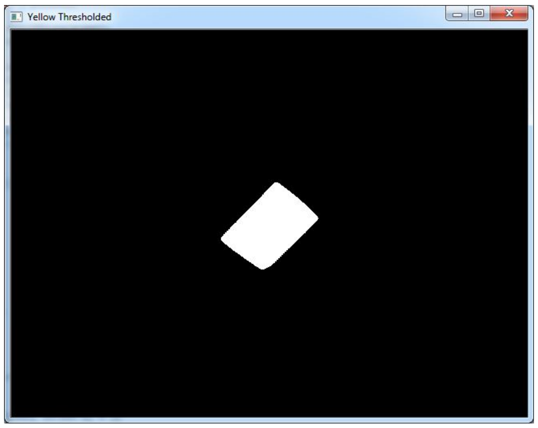
(a) Original image



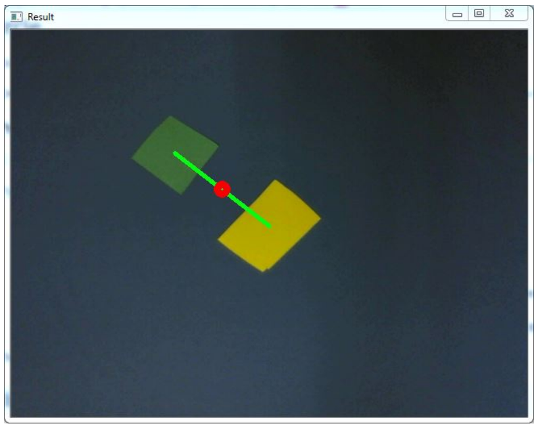
(b) HSV image



(c) Green thresholded



(d) Yellow thresholded



(e) Result

Figure 5.2: 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 (2.20).

The first new frame is defined in the center of the camera lens. The homogeneous transformation $\boldsymbol{A}_1$ 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 5.3. The parameters related to the transformations are also marked in the same figure.

$$\boldsymbol{A}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.12}$$

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.

$$\alpha = -atan2(-\delta y, \delta x) \tag{5.13}$$
$$d = L\sqrt{\delta x^2 + \delta y^2} \tag{5.14}$$
$$\beta = -\tan^{-1}(d) \tag{5.15}$$
$$\tag{5.16}$$

The resulting homogeneous transformation matrix is

$$\boldsymbol{A}_2 = \begin{bmatrix} c_\alpha & -s_\alpha & 0 & 0 \\ s_\alpha & c_\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\beta & -s_\beta & 0 \\ 0 & s_\beta & c_\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_\alpha & -s_\alpha c_\beta & s_\alpha s_\beta & 0 \\ s_\alpha & c_\alpha c_\beta & -c_\alpha s_\beta & 0 \\ 0 & s_\beta & c_\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.17}$$

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.

$$\boldsymbol{T}^n_{pf} = \boldsymbol{R}^n_b(\boldsymbol{\Theta}_{nb})\boldsymbol{A}_1\boldsymbol{A}_2 \tag{5.18}$$

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

$$\boldsymbol{z}^n_{pf} = \begin{bmatrix} s_\alpha s_\beta c_\psi c_\theta - c_\alpha s_\beta(-s_\psi c_\phi + c_\psi s_\theta s_\phi) + c_\beta(s_\psi s_\phi + c_\psi c_\phi s_\theta) \\ s_\alpha s_\beta s_\psi c_\theta - c_\alpha s_\beta(c_\psi c_\phi + s_\psi s_\theta s_\phi) + c_\beta(-c_\psi s_\phi + s_\theta s_\psi s_\phi) \\ -s_\alpha s_\beta s_\theta - c_\alpha s_\beta c_\theta s_\phi + c_\beta c_\theta c_\phi \end{bmatrix} \boldsymbol{z}^n_n = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5.19}$$

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.

$$\cos\gamma = \frac{\boldsymbol{z}^n_{pf} \cdot \boldsymbol{z}^n_n}{|\boldsymbol{z}^n_{pf}||\boldsymbol{z}^n_n|} = z^n_{pf_3} \tag{5.20}$$

$$\cos\gamma = \frac{h - d_1 c_\theta c_\phi}{d_2} \tag{5.21}$$

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

$$d_2 = \frac{h - d_1 c_\theta c_\phi}{z^n_{pf_3}} \tag{5.22}$$

$$\boldsymbol{A}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.23}$$

$$\boldsymbol{T}^n_{obj} = \boldsymbol{T}^n_{pf}\boldsymbol{A}_3 \tag{5.24}$$

Calculations of equation (5.24) will read out the position of the origin of the reference frame in the sensor node (according to equation (2.20)). This gives

$$\boldsymbol{p}^n_{obj} = \begin{bmatrix} d_2 z^n_{pf_1} + d_1(s_\psi s_\phi + c_\psi s_\theta c_\phi) \\ d_2 z^n_{pf_2} + d_1(-c_\psi s_\phi + s_\psi s_\theta c_\phi) \\ h \end{bmatrix} \tag{5.25}$$

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

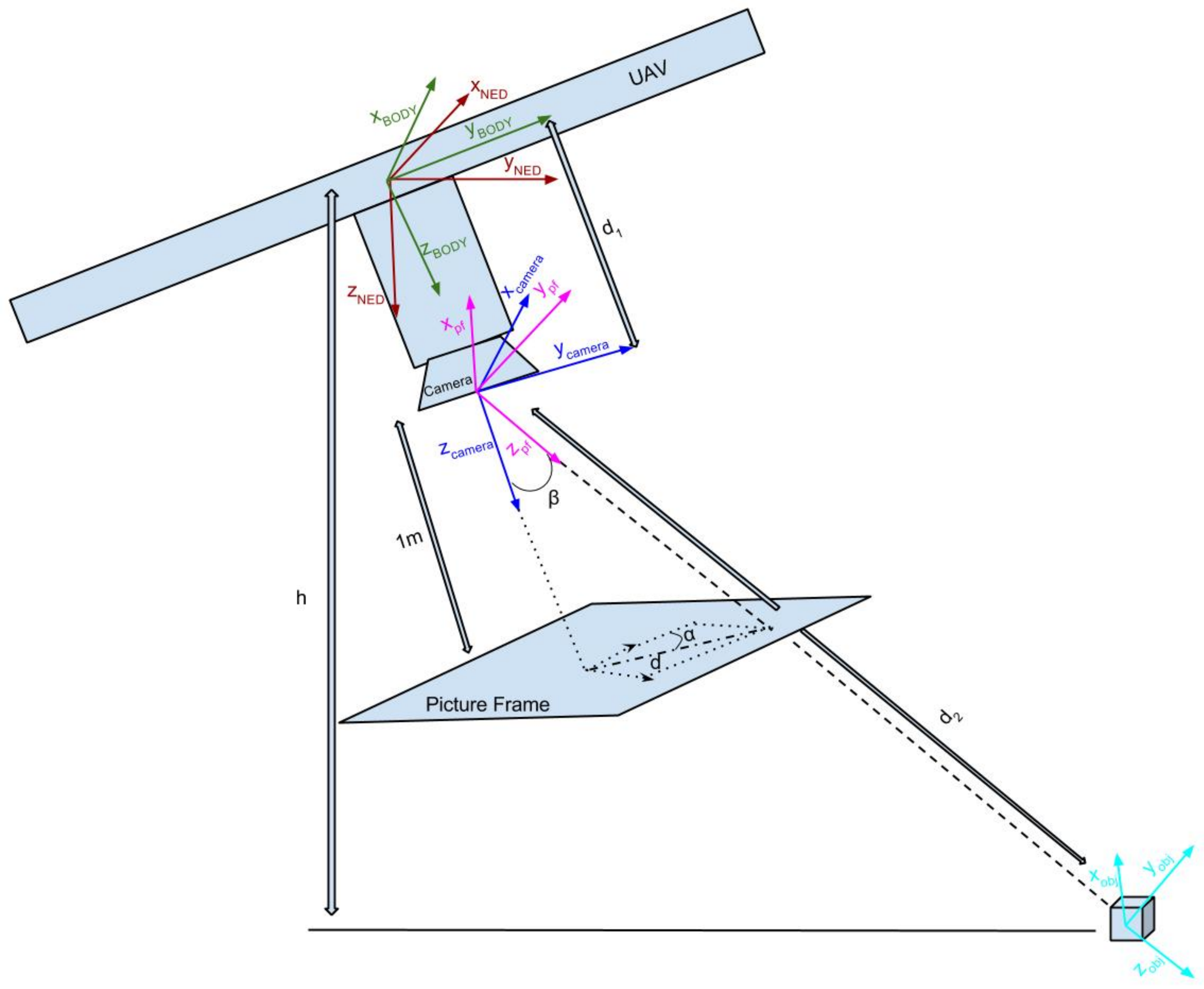


Figure 5.3: Visualization of the different coordinate systems

5.2.2 Velocity Calculations

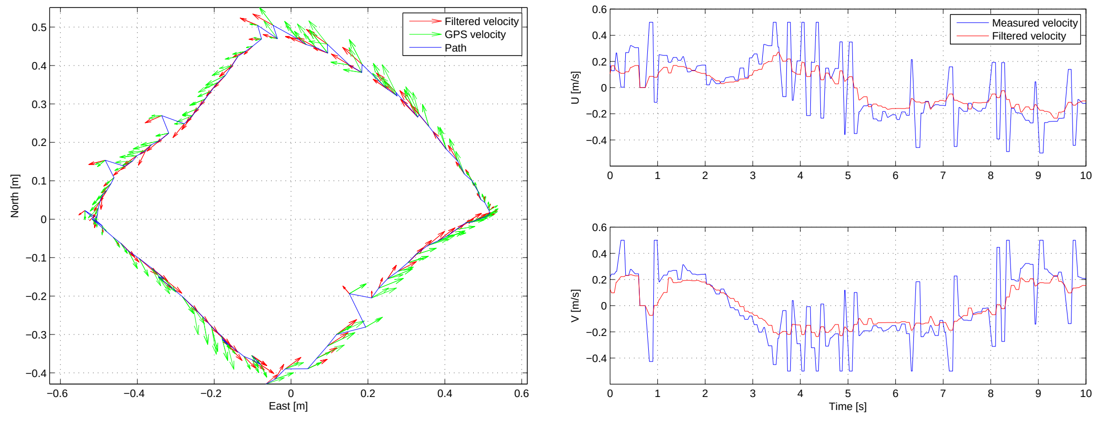
Testing revealed some major concerns in relation to the velocity measurements from the GPS of the APM. Good velocity measurements are crucial for the position controller in an application that needs to be as accurate as this one. It turned out that the accuracy of the GPS velocity measurements are very varying. Sometimes the GPS velocity was quite good and some times the GPS velocity vector was normal to the movement of the UAV. When the

GPS measurements was at its worst, the damping term of the position controller made the UAV circle the sensor node until sight of the node was lost.

In order to be as independent of the unreliable GPS velocity measurements as possible, velocity was calculated using the measurements from the camera frame. These calculations resulted in very noisy velocity measurements. To handle this a FIR-filter was used to get smoother measurements.

The velocity measurements calculated using the camera are used when the sensor node is in the picture frame. When the sensor node is outside the picture frame, GPS velocity measurements are used.

The performance of both the GPS measurements and the filtered calculated velocity measurements are shown in Figure 5.4(a) where the UAV is moved in a square pattern above the sensor node. The calculated velocity measurements and the filtered calculated velocity measurements are displayed in Figure 5.4(b).



(a) NE plot of the path of the UAV (b) Measured and filtered velocity

Figure 5.4: Velocity measurements

The GPS velocity measurements are actually at its best in this figure, but one can still observe that the filtered measured velocity follows the direction the UAV much better. There could be several reasons for this. First of all is the GPS inaccurate with a CEP of 2.5 m, secondly the position measurements of the UAV is calculated using the heading of the UAV. The heading is measured by the use of a magnetometer which can be inaccurate, hence the

GPS could have a different perception of where north is than the camera tracking. This could lead to an offset in the direction of the GPS velocity.

When the camera tracking is used as a basis for the velocity measurements, all the measurements and the controller uses the same heading in its calculations, hence poor heading measurements are cancelled out. As one can see from Figure 5.4(b), the calculated velocity are quite noisy, but the filtered velocities are quite smooth and there is little delay. One can see that the filtered velocity makes sense from Figure 5.4(a).

5.3 Software Implementation

The system consists of three main software components: the ArduCopter software running on the APM, OpenCV which is used by the camera application and DUNE running on the PandaBoard. An overview of the modifications and additions to these components is given in the following sections.

5.3.1 ArduCopter

A new mode of operation (DUNE mode) was added to the ArduCopter project by the Hexacopter group at AMOS. The result of this modification was that when in DUNE mode, desired roll, desired pitch, desired yaw and desired altitude could be set using the serial interface and the MAVLink protocol.

In this application accurate altitude measurements are necessary for both the altitude controller and the node tracking algorithm. As mentioned the APM has already an interface for connection of the selected sonar. But sonar measurements were not sent over the serial bus, hence the code was altered so that it send the sonar measurements every time a position measurement is sent. In that way DUNE is able to receive the sonar measurements to use them in the node tracking algorithm. Another alteration made in relation to the sonar was to use sonar measurements instead of barometric measurements in the altitude controllers of the UAV. This gave much more accurate altitude control in the relevant altitude span of this application.

5.3.2 OpenCV

The OpenCV library is mostly used in its original form, but some issues caused by the PandaBord made it necessary to make a few alterations to the library.

Default camera resolution was defined to be 640 $\times$ 480 pixels. The application has no need for high resolution as the sensor node is just as easy to spot with lower resolution. Hence a resolution of 320 $\times$ 240 pixels was desired to reduce the processing time of the color recognition by a factor of four for each frame processed. The resolution is normally possible to alter using OpenCV commands, but these commands had no effect using the cross compiled OpenCV library on the PandaBoard. Changing the settings of the camera using the command line on the PandaBoard had no effect either as OpenCV changed the settings back. The fix for this problem was to change definitions of default resolution in the cap_v4l.cpp and cap_libv4l.cpp files in the OpenCV library. The definitions DEFAULT_V4L_WIDTH and DEFAULT_V4L_HEIGHT was changed to respectively 320 and 240.

Another challenge with the camera stream was that the camera buffered frames when the PandaBoard was unable to receive the frames at the same rate as the camera captured them. This lead to a situation where the PandaBoard received frames that were two-three samples old. The solution to solve this problem became to force OpenCV to drop old frames instead of buffering them. This was done by changing the parameters MAX_V4L_BUFFERS and DEFAULT_V4L_BUFFERS to 1 in the files mentioned above.

After these changes were done, the library was recompiled and the desired results were achieved.

5.3.3 DUNE

This section describes the structure and functionality of the DUNE software controlling the UAV and the drop and recovery mechanism. An overview of the different tasks and how they communicate is shown in Figure (5.5). An explanation of each task follows. There are several other tasks that run in the background with some support functionality, as for instance monitoring of CPU usage and logging of different parameters. These tasks are omitted in the following explanation to get a better overview of the most relevant tasks.

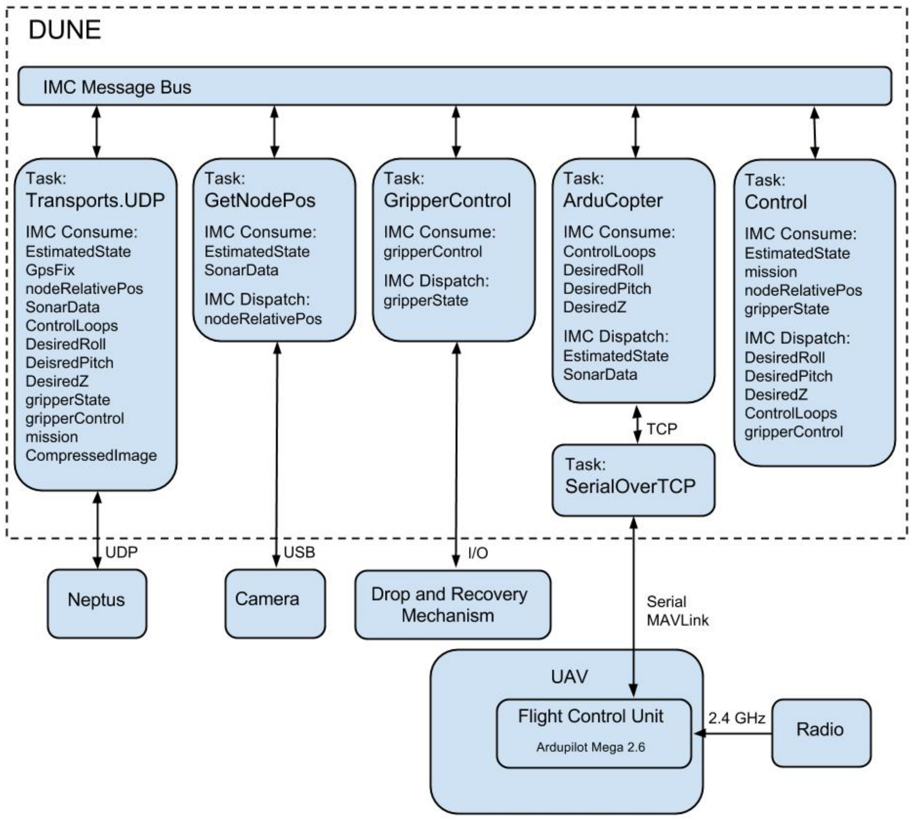

DUNE
IMC Message Bus
Task: Transports.UDP
IMC Consume: EstimatedState GpsFix nodeRelativePos SonarData ControlLoops DesiredRoll DeisredPitch DesiredZ gripperState gripperControl mission CompressedImage
Task: GetNodePos
IMC Consume: EstimatedState SonarData
IMC Dispatch: nodeRelativePos
Task: GripperControl
IMC Consume: gripperControl
IMC Dispatch: gripperState
Task: ArduCopter
IMC Consume: ControlLoops DesiredRoll DesiredPitch DesiredZ
IMC Dispatch: EstimatedState SonarData
Task: Control
IMC Consume: EstimatedState mission nodeRelativePos gripperState
IMC Dispatch: DesiredRoll DesiredPitch DesiredZ ControlLoops gripperControl
TCP
Task: SerialOverTCP
UDP
USB
I/O
Neptus
Camera
Drop and Recovery Mechanism
Serial MAVLink
UAV
Flight Control Unit
Ardupilot Mega 2.6
2.4 GHz
Radio

Figure 5.5: Software structure

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. This task is developed by LSTS.

GetNodePos

This task uses color recognition to track the sensor node according to the procedure described in Section 5.2.1. 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.

The performance of the node tracking algorithm is crucial for the accuracy of the system as the measurements are used in the feedback for the position controller. The node tracking is dependent on attitude and altitude measurements that are updated as recently as possible. Hence extra care needed to be made in the structure of the code to make sure that all IMC messages available are consumed before running the calculations to get the node position.

Another major concern is the frame rate one can achieve from the camera and the processing time of each frame. Unfortunately the frame rate varied quite a lot. To get a stable and reliable frame rate this task was defined as a periodic task that was run at 10 Hz.

GripperControl

The gripper control task consumes gripperControl messages from the IMC-bus and controls the gripper and gripper platform according to the received commands. The gripper is controlled by setting a gripper enable bit and a gripper position bit on the PandaBards GPIO port. The position bit is high for open position and low for closed position. The gripper platform 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 gripper platform 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 gripper platform.

To be able to control the GPIO ports of the PandaBoard mux-settings needed to be set on the PandaBoard, and the relevant pins needed to be activated and direction of the pins needed to be defined. This was done by sending terminal commands using the system function in C++.

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. The task has been further modified to reduce the amount of messages transferred between DUNE and the APM and to include an interface to receive sonar messages from the APM and put the values on the IMC-bus. It is good to reduce the unnecessary communication with the APM because it is already working close to its limits. An update frequency of approximately 25 Hz was achieved to and from the APM.

Control

The control task is the main task in the control structure. The task is inactive when DUNE starts up, and is activated when the radio used to control the UAV switches mode into DUNE-mode. It will also be deactivated if the pilot switches the UAV out of DUNE-mode. This is a feature that simplifies operation and enhances safety greatly. In this task is it defined which states that will be controlled by the task and which states the pilot can control using the radio. During testing in this project the control task had some different goals and controlled different states. This is better explained where the different test are presented.

This task receives missions from the IMC bus. These missions can be either drop or pickup at a specified waypoint. The missions are currently sent to the IMC bus by defining the missions in the configuration file for DUNE. As future work could this missions for instance be sent from Neptus.

The missions are executed in the order as they are received and when the queue of missions to execute is empty the UAV will return to the place where DUNE mode was activated and land.

The UAV will fly at an altitude of 7 meters between the waypoints using the waypoint tracking algorithm. An altitude of 7 meters is chosen because it is in the upper limit of what the sonar can handle. When reaching a pickup waypoint the UAV will use the camera tracking algorithm in combination with the position controller to get placed straight above the sensor node, the gripper will open and the gripper platform will be lowered. While being sufficiently close to the position straight above the sensor node, the altitude controller will ramp down until the sensor node can be gripped by the gripper. After the gripper has gripped the sensor node, the gripper platform will be raised and the UAV will continue to the next waypoint in the queue (or return and land if the queue is empty).

Velocity measurements are during pickup calculated using the camera as long as the sensor node is in the picture frame. When the sensor node is outside the picture frame, GPS velocity mea- surements are used.

When reaching a drop waypoint the UAV will stop and then drop the sensor node by opening the gripper, before continuing on the next waypoint in the queue. When the UAV reaches the waypoint where it is supposed to land, the altitude is ramped down until the UAV touches the ground.

Support Software

In addition to the already mentioned tasks were two tasks developed to support operation. These tasks are not shown as a part of the software structure as they only support operation in special phases.

One task that displays a stream of the processed camera stream was developed to be used to easily verify the camera tracking. This task is run as part of DUNE on a ground control computer if desired. It receives the stream using a Wi-Fi link to the UAV and displays it. This stream could for future work be received by Neptus instead.

The other support task that was created was a task that could be used to calibrate the thresholds for the camera tracking. Using a graphical interface the threshold values for the HSV values of the mount of the sensor node can easily be found. The code used for this is strongly based on [Fernando, 2104]. This task is run as part of DUNE on a computer using the same camera as the camera application. This task is very useful as thresholds for the camera application needs to be calibrated to fit the environment of operation.

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 assumptions 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

To model the dynamics of the hexacopter equations (2.21) and (2.22) 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 calculations are shown in equations (6.1) and (6.2). A model of a hexacopter is shown in Figure 6.1 where forces, torques and angular speed of the propellers are marked.

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

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

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.

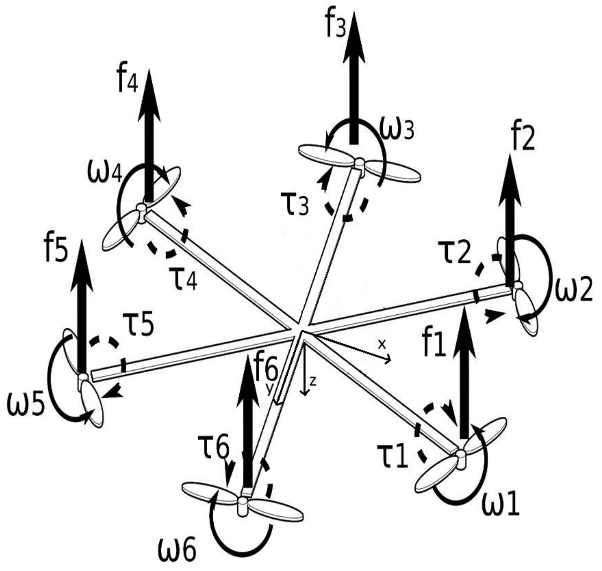


Figure 6.1: Model of hexacopter *Courtesy of [Alaimo et al., 2013]*

Equations (6.1) and (6.2) 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 (2.21) and (2.22) 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 and the altitude controller is approximated as a PID-controller. One must assume that the controllers of the APM follows the references given, the chosen controllers will do this, hence this is a fair approximation.

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 $\boldsymbol{\tau}_c = [T_c \ \tau_{\phi_c} \ \tau_{\theta_c} \ \tau_{\psi_c}]^T$. Using equation (6.3) combined with the control vector gives the following relationship.

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

The desired input to each motor is calculated using

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

where the pseudo inverse of $\boldsymbol{A}$ is calculated

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

6.3 Simulation of the Camera Algorithm

To simulate the measurements from the camera, the algorithm given in Section 5.2.1 was used to find the position of each of the corners of the camera frame. Then a ray casting algorithm [Franklin, 2014] was 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 should be sent back to the control algorithm.

6.4 Simulink Model

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

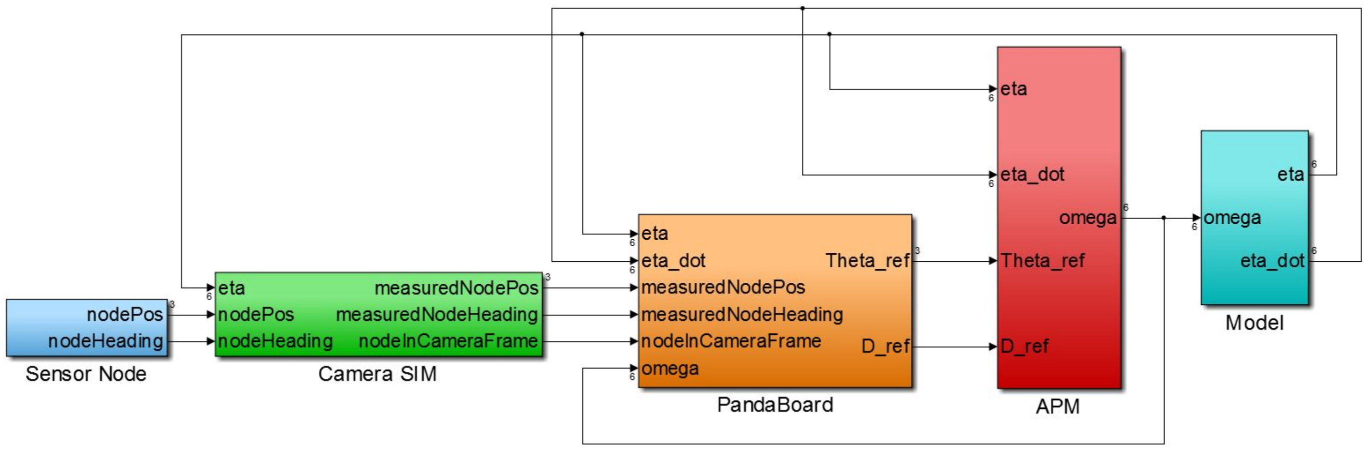


Figure 6.2: Simulink model

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 is 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.

Four scenarios was simulated: first a simulation with no disturbances, then a simulation with constant disturbance, then a simulation with varying disturbances and finally a simulation with disturbances affecting both the sensor node and the hexacopter.

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

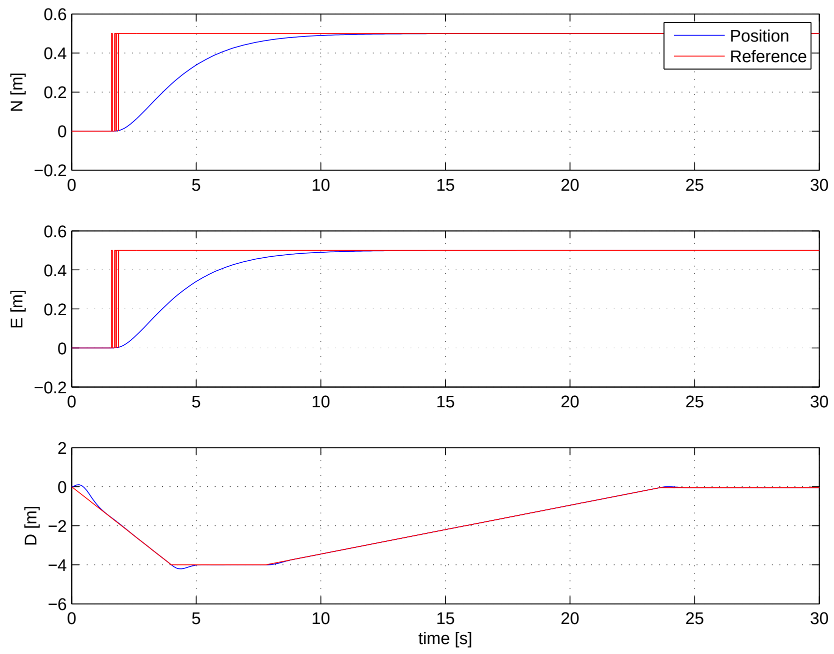


Figure 6.3: North-East-Down position of the hexacopter

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.

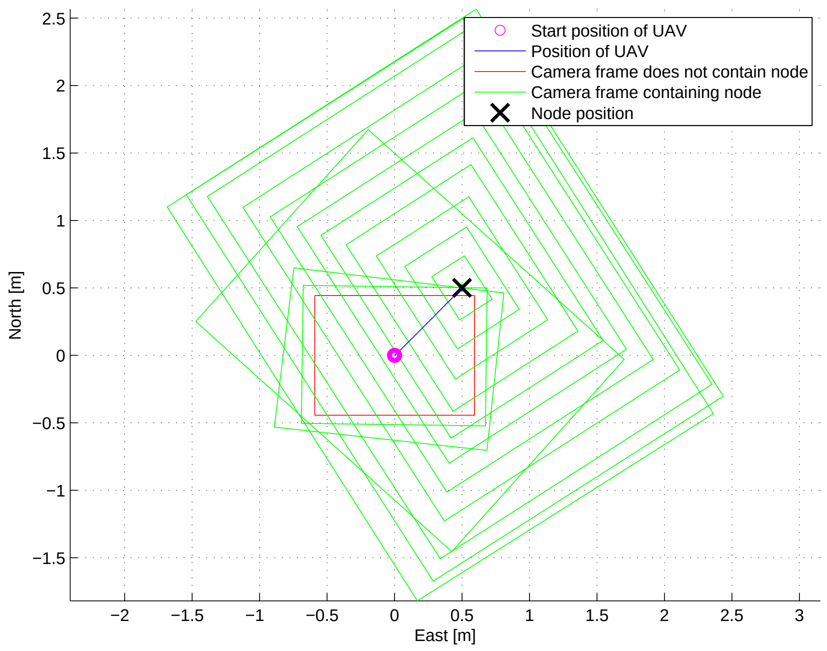


Figure 6.4: North-East-Down position of the hexacopter, sensor node and camera frame

Discussion

The hexacopter was able to lower itself down on the exact spot of the sensor node without loosing sight of the sensor node. This is a very promising result, although an environment without any disturbances is probably not a very realistic one.

6.5.2 Simulation With Constant Disturbances

Early tests in the lab showed that the controller used in the previous simulations 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 was performed. It was chosen to add a constant force of 0.5 N in North direction, this could for instance represent wind.

Results

As can be seen from Figure 6.5 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.

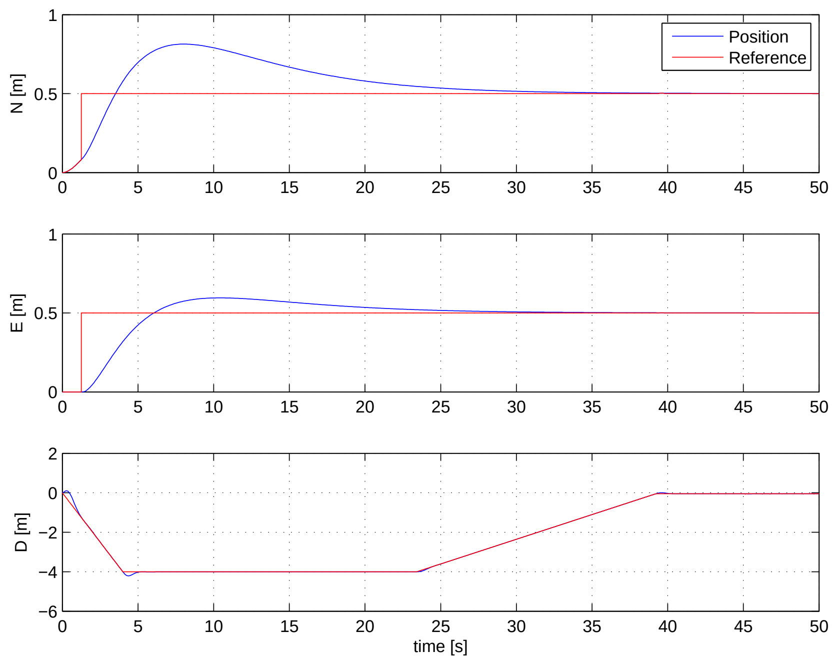


Figure 6.5: North-East-Down position of the hexacopter

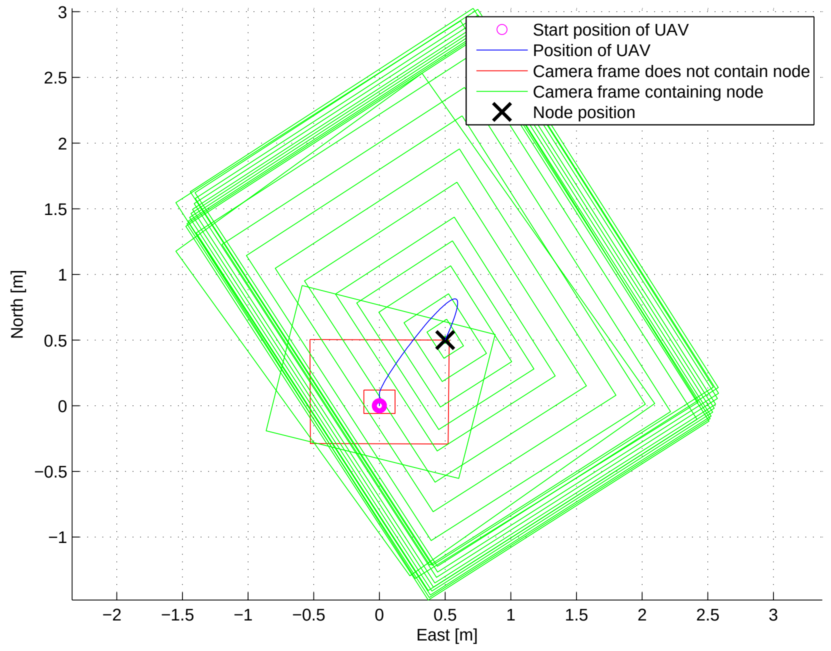


Figure 6.6: North-East-Down position of the hexacopter, sensor node and camera frame

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 disturbance 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.

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.

Results

The hexacopter stays approximately straight above the sensor node, and is able to lower itself a bit 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. When the sensor node is outside the camera frame, the reference for the controller is the same as the current position. That is why it looks like the reference leads the UAV to drift of in the end of the plots in Figure 6.7.

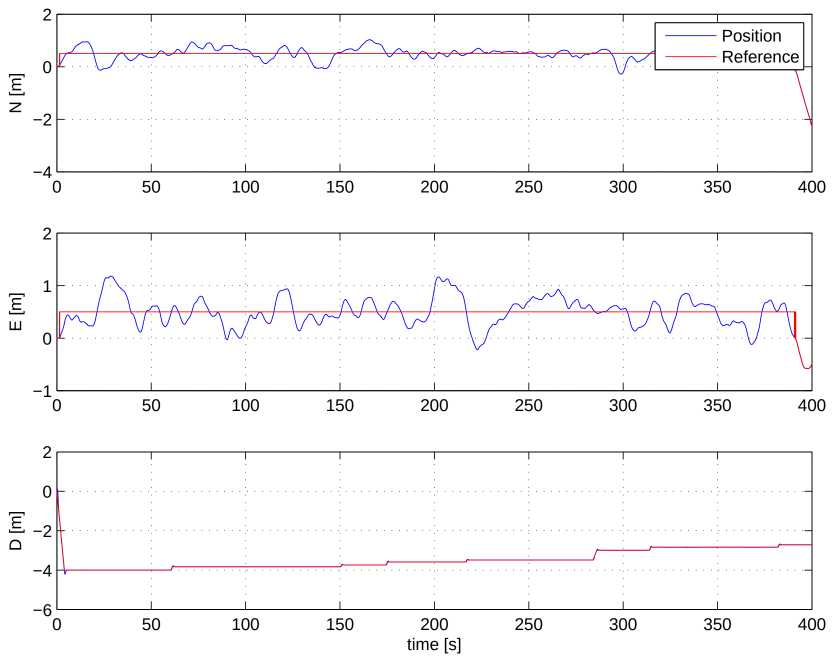


Figure 6.7: North-East-Down position of the hexacopter

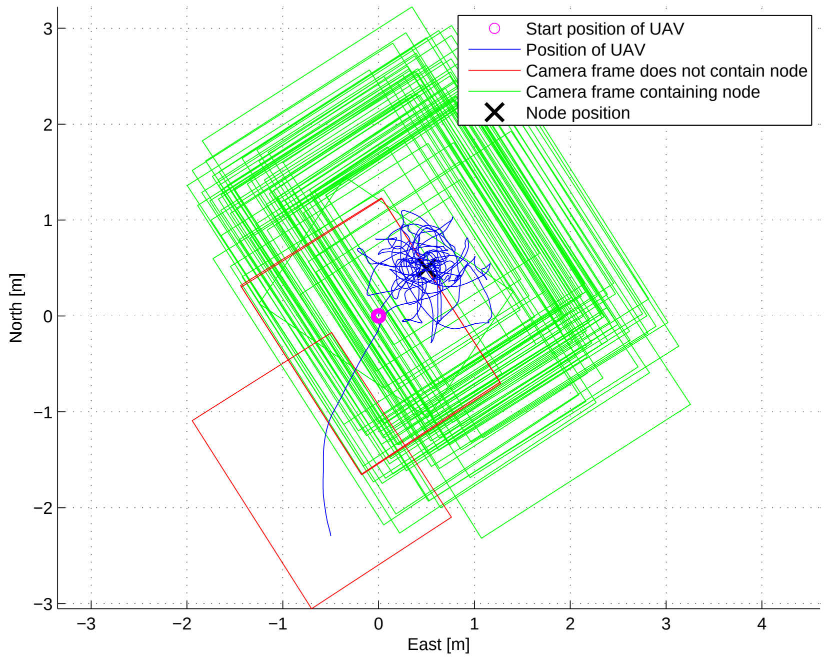


Figure 6.8: North-East-Down position of the hexacopter, sensor node and camera frame

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. When flying close to the sensor node, the area covered by the camera is small, hence a random force acting upon the hexacopter in the last stage of pickup will easily lead to a 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 with disturbances on the sensor node in addition to a constant disturbance on the hexacopter, e.g. due to wind or inaccuracies in the APM. The constant disturbance on the hexacopter is a force of 0.5 N acting in North direction. The sensor node is affected by varying noise, which makes it change heading and position. The position of the node is modelled as random walk where the position is limited to a square of 2 times 2 meters, while the heading of the node is modelled as unconstrained 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 drift off.

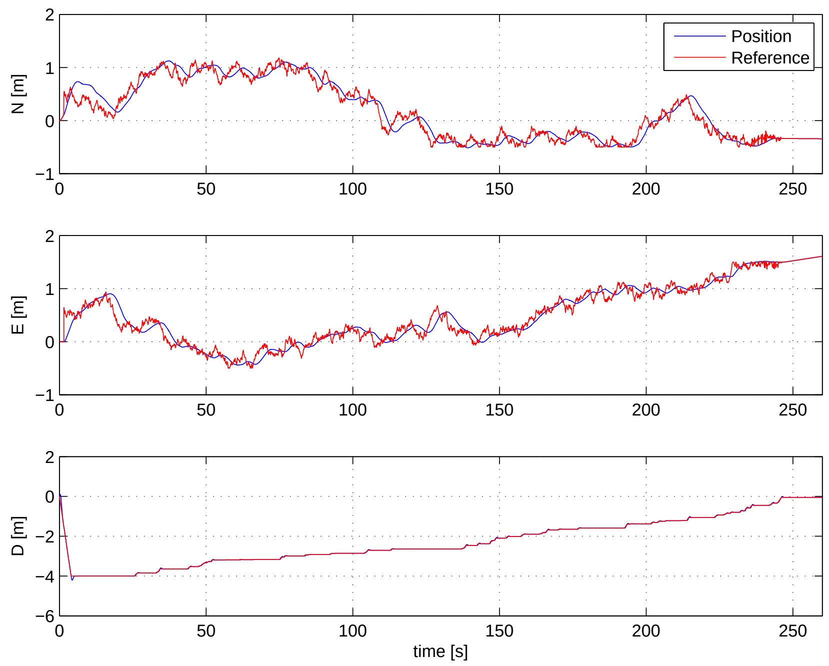


Figure 6.9: North-East-Down position of the hexacopter

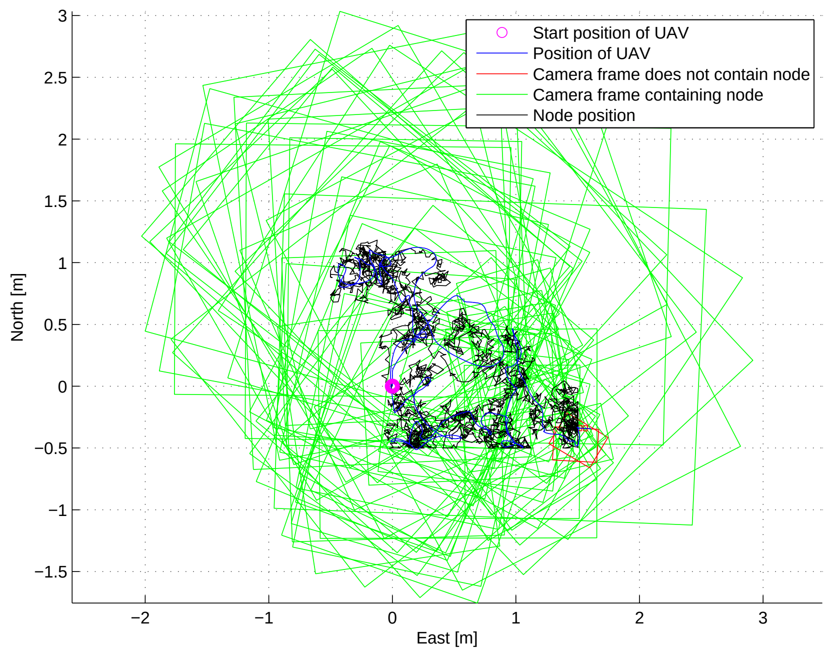


Figure 6.10: North-East-Down position of the hexacopter, sensor node and camera frame

Discussion

The hexacopters abilities to track the sensor nodes position and heading is quite promising. The main issue that is revealed is that it could loose sight of the sensor node and drift off when affected by disturbances. The hexacopter is not able to get back on track after a drift off, this reveals a 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 disturbances as rough weather conditions, meaning that success operation is 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 a software-in-the-loop (SIL) test, and explain how the different software are interfaced to each other. The different SIL tests conducted will also be presented.

7.1 SIL-Setup

The main purpose of the SIL test is to verify the control software presented in Chapter 5.3.3.

The SIL-setup contains six main components. These components are: DUNE, Neptus, ArduCopter SIL, ArduPilot, MAVProxy and APM Planner. The interface between the different components is displayed in Figure 7.1, and their roles are briefly explained below.

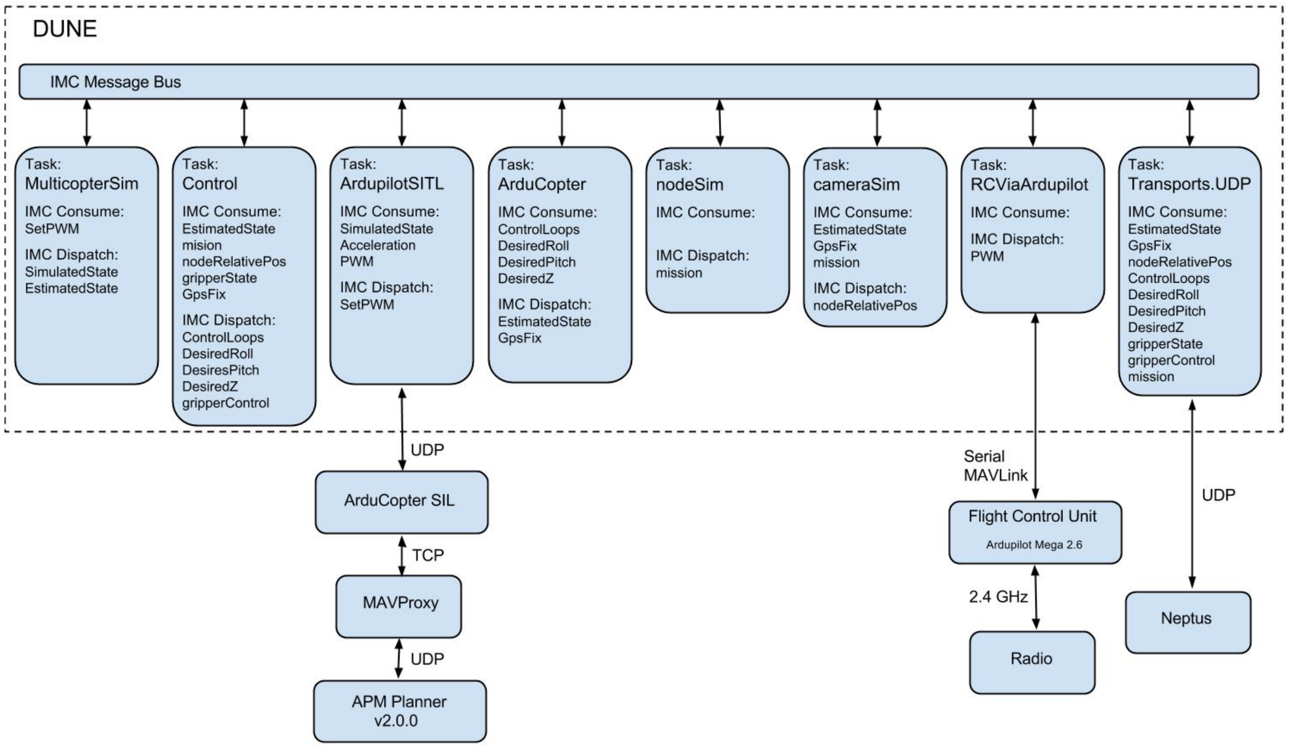

DUNE
IMC Message Bus
Task: MulticopterSim
IMC Consume: SetPWM
IMC Dispatch: SimulatedState EstimatedState
Task: Control
IMC Consume: EstimatedState mision nodeRelativePos gripperState GpsFix
IMC Dispatch: ControlLoops DesiredRoll DesiresPitch DesiredZ gripperControl
Task: ArdupilotSITL
IMC Consume: SimulatedState Acceleration PWM
IMC Dispatch: SetPWM
Task: ArduCopter
IMC Consume: ControlLoops DesiredRoll DesiredPitch DesiredZ
IMC Dispatch: EstimatedState GpsFix
Task: nodeSim
IMC Consume:
IMC Dispatch: mission
Task: cameraSim
IMC Consume: EstimatedState GpsFix mission
IMC Dispatch: nodeRelativePos
Task: RCViaArdupilot
IMC Consume:
IMC Dispatch: PWM
Task: Transports.UDP
IMC Consume: EstimatedState GpsFix nodeRelativePos ControlLoops DesiredRoll DesiredPitch DesiredZ gripperState gripperControl mission
UDP
ArduCopter SIL
TCP
MAVProxy
UDP
APM Planner v2.0.0
Serial MAVLink
Flight Control Unit
Ardupilot Mega 2.6
2.4 GHz
Radio
UDP
Neptus

Figure 7.1: Setup used for SIL-testing

DUNE

DUNE is the part of the SIL that is put to the test. To be able to see if the control algorithms work, some of the features need to be simulated. This is done in tasks that only are run in SIL mode. A model of a multicopter is used to calculate realistic attitude and position states based on 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 Section 6.3.

ArduCopter SIL

The ArduCopter project contains a SIL-feature. This means that one can choose to compile the project for SIL instead of for use with hardware. This feature is very practical both for testing modifications 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 SIL feature.

ArduPilot

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

MAVProxy

MAVProxy is a simple ground control station, but it is not used for that purpose in this setup. The task for MAVProxy here is simply to be a bridge between the ArduCopter SIL and AMP Planner, which is another ground control station. This bridge is needed because the ArduCopter SIL 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 SIL and an simple interface to modify parameters used by the ArduCopter SIL. It can also be used to arm the ArduCopter SIL and to change mode.

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. It is convenient to use to get a quick overview of the different states, but it is not a necessary part of the SIL setup.

7.2 Pickup Test Without Disturbances

The SIL-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 SIL-test is conducted without any disturbances. The system has already been simulated without disturbances in the previous chapter, but SIL 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 ± 2 cm in the NE-plane, and the hexacopter is able to descend all the way down to the sensor node.

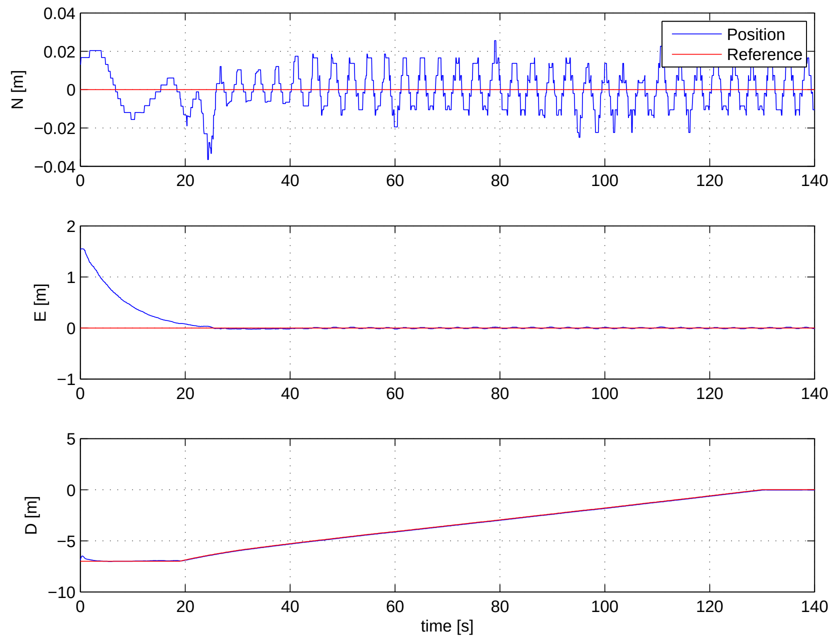


Figure 7.2: North-East-Down position of the hexacopter

Reference 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 reference attitude and that the measured attitude is a bit delayed versus the reference.

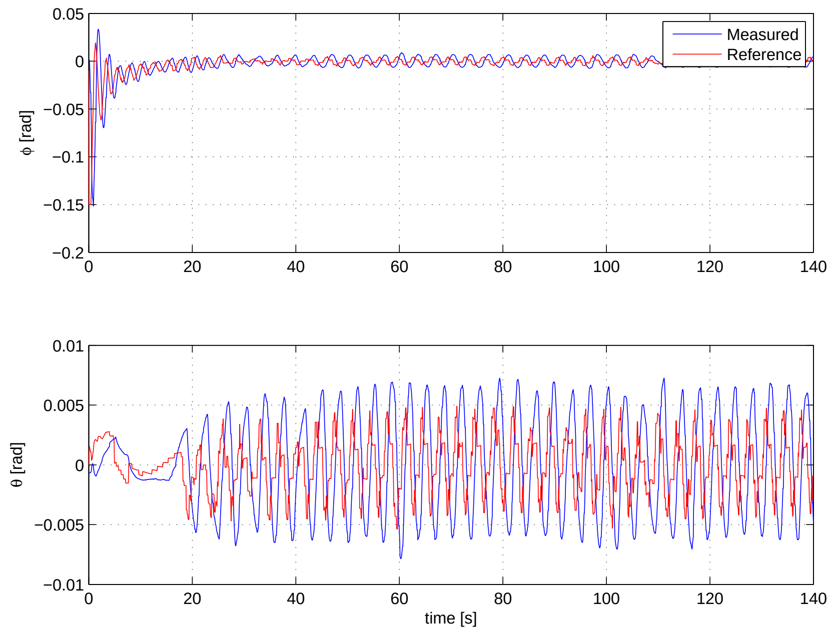


Figure 7.3: Attitude of the hexacopter

7.2.2 Discussion

The control structure is considered to be able to conduct a pickup of a sensor node as it manages to lower the hexacopter down to the sensor node. It is considered to be accurate enough 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 were able to verify the control structure, and showed promising results for 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 ArduCopter SIL to add drift and sensor noise to the system. Some early stage flight tests in the lab 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 and whether the sensor node is in the picture frame or not is shown in Figure 7.5. One can see from this figure that the position in the NE-plane oscillates slightly 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 the ground. It is able to regain sight of the node because of the damping term in the controller that makes the hexacopter stay in the same area.

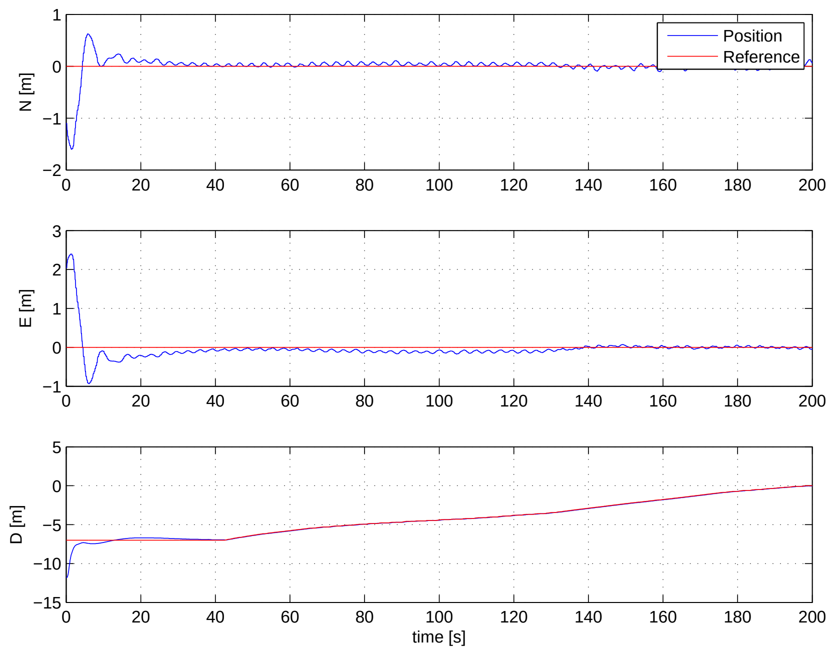


Figure 7.4: North-East-Down position of the hexacopter

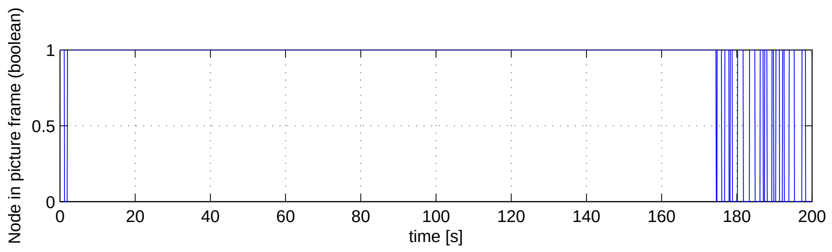


Figure 7.5: Showing whether the node is in the picture frame or not

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

Several additional 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

It was shown that the hexacopter is able to lower itself down to the sensor node even though it 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 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 velocities are more accurate than the velocity 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.

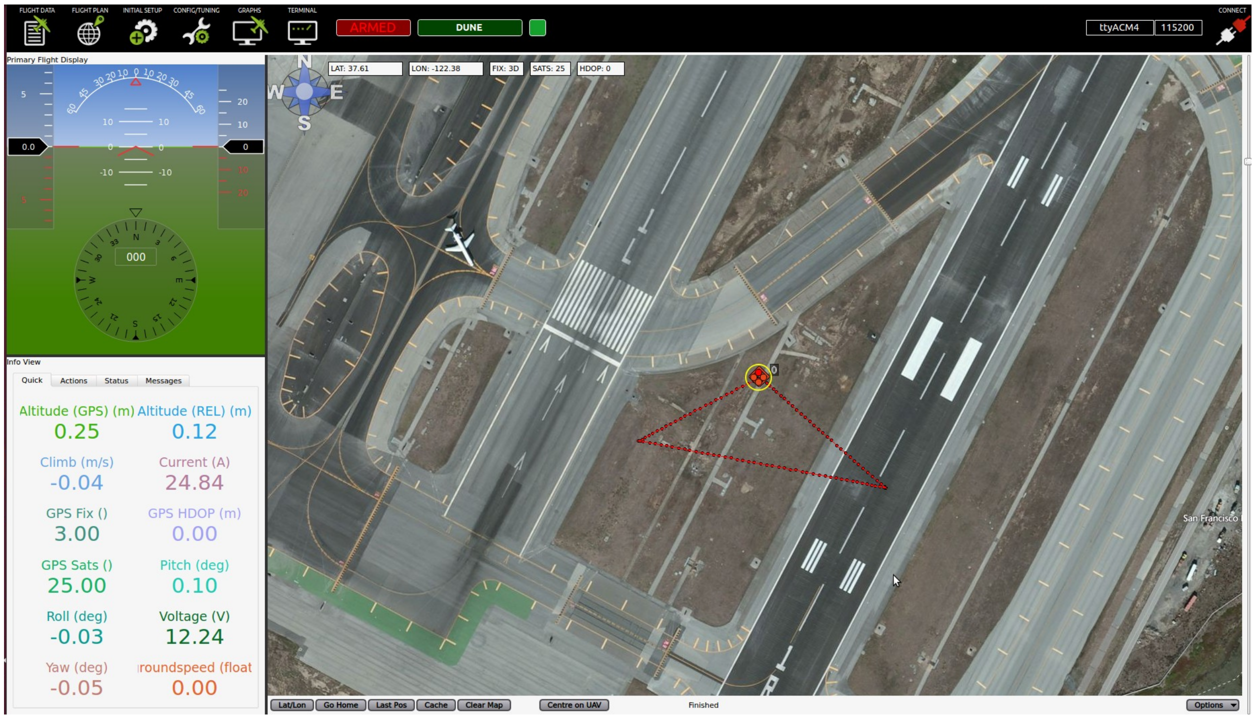
The simulated camera has the same field of view as the camera used for real life testing. There exists web-cameras with greater field of view than the one used to decrease the problem with the camera loosing sight of the sensor node.

7.4 Drop and Recovery Mission

In this test an entire drop and recovery mission is executed. Two wayponts are sent on the IMC bus to be executed by the control task. Signals to the task that controls the drop and recovery mechanism are monitored to verify the control logic. Two waypoints where sent on the IMC bus, first a pickup waypoint a bit South East of the start point, then a drop waypoint South West of the start point.

7.4.1 Results

A screenshot of APM Planner at the end of the mission is shown in Figure 7.6. The path of the UAV to the waypoints and then back to the start position to land can be seen in the figure.



ttyACM4 115200

LAT: 37.61 LON: -122.38 FIX: 3D SATS: 25 HDOP: 0

Finished

Altitude (GPS) (m)	Altitude (REL) (m)
0.25	0.12
Climb (m/s)	Current (A)
-0.04	24.84
GPS Fix ()	GPS HDOP (m)
3.00	0.00
GPS Sats ()	Pitch (deg)
25.00	0.10
Roll (deg)	Voltage (V)
-0.03	12.24
Yaw (deg)	roundspeed (float
-0.05	0.00

Figure 7.6: Screenshot of APM Planner at the end of the mission

Plots of the speed and altitude of the UAV are included in Figure 7.7 to get a better idea of how the mission is carried out. One can easily see the different parts of the mission in this figure.

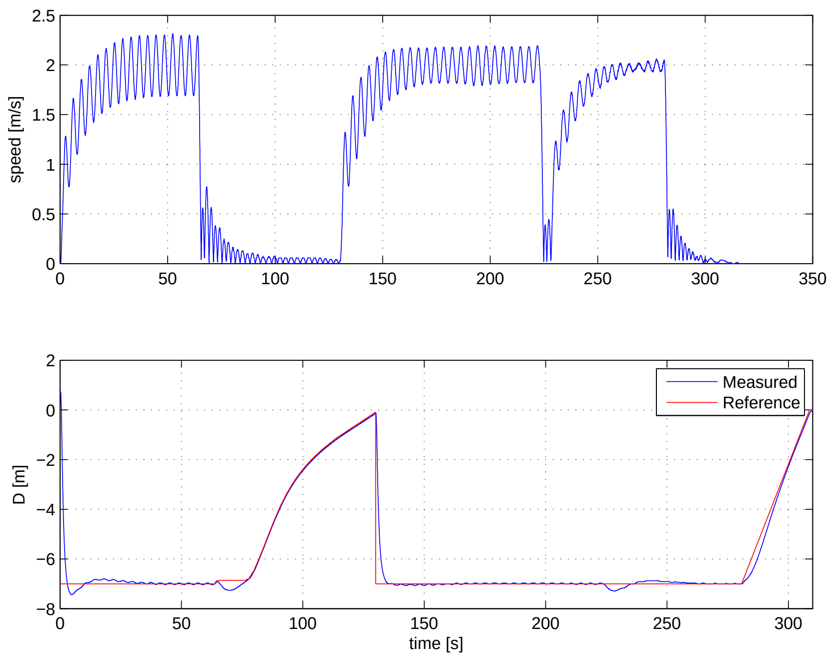


Figure 7.7: Speed and height of the UAV during the mission

The control signals for the drop and recovery mechanism were observed and they were according to the specification given in the description of the control task.

7.4.2 Discussion

The mission is carried out according to the design. All the different controllers work together and the different waypoints are tracked and pickup and drop are executed according to plan.

The speed oscillates a bit when travelling between the waypoints, but this is only natural as there is little damping in the system, and there is no damping in the controller. The performance of the velocity controller are considered to be good.

Chapter 8

Testing in the Lab

Simulations have proven the abilities of the controller and SIL tests have verified that the control structure presented in Chapter 5.5 can be able to execute a pickup of a sensor node. In addition both the simulations and the SIL tests have revealed some weaknesses of the system when facing disturbances or inaccuracies in the APM.

DUNE will during these tests run on a PandaBoard communicating with the APM using a serial interface. It is interesting to perform a real life test using this setup to see how the reduced performance of the PandaBoard compared to a regular computer and how the communication link between the PandaBoard and the APM, will affect the performance. Another point that needs to be considered is the accuracy of the sensors used, and how much the hexacopter drifts on its own accord.

The lab used for testing is indoors. This is both positive and negative. The roof is only approximately 3 m high. This makes it difficult to test other parts of operation than the absolutely last bit of the pickup where the area showed in the picture frame is small, making it more difficult to get a stable position above the node from the start. The GPS-signal can be weaker due to the roof, but performance of the GPS would not have been good anyway. Of this reason both position and speed measurements are based on the camera application, except when the sensor node is outside of the picture frame, then speed measurements are based on GPS and position errors are set to 0. On the plus side there is no wind, but there can be a bit of added turbulence created by the propellers and the hard surface of the floor.

Some early stage testing revealed some weaknesses in the heading controller of the APM. The heading overshooted the reference quite a bit and while changing heading, the controller introduced oscillations in roll and pitch. This heading controller of the APM should be looked into as future work, but it was not a priority in this project. Hence all this tests are conducted without automatic heading control.

The tests are performed stepwise, meaning that one feature is tested, then another one is added until the full system is tested. Videos of all the tests presented in this chapter are included in the digital appendix.

8.1 Hover Above the Sensor Node

The goal of this test is to hover straight above the sensor node. The hexacopter is flown manually up above the sensor node. Then the system is switched into DUNE-mode, meaning that the controller kicks in and roll and pitch reference values are sent from DUNE to the APM. Throttle and yaw are still controlled by the pilot using the radio.

8.1.1 Result

The North and East error measurements are displayed in Figure 8.1. From the figure one can see that the UAV is able to hover above the sensor node with a position error within $\pm$ 20 cm from the desired position.

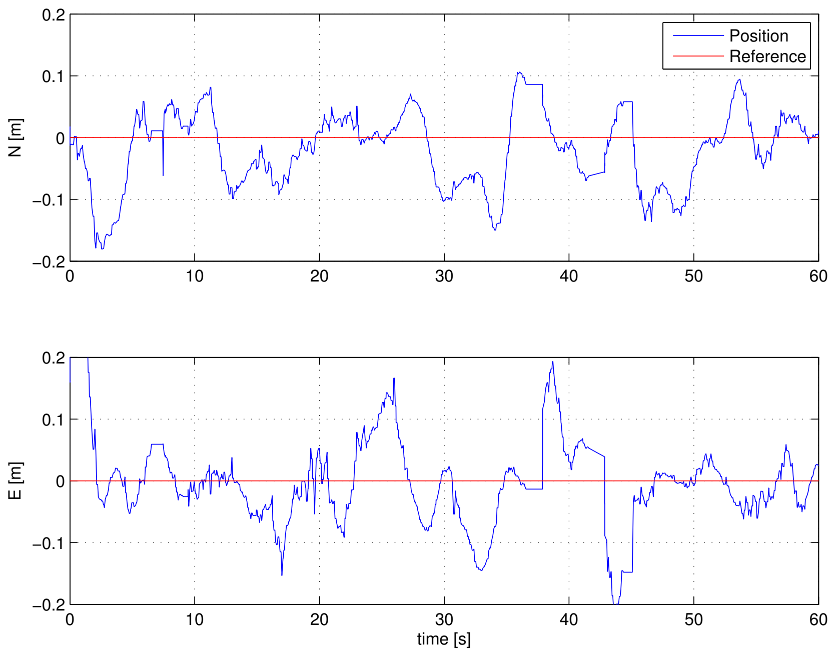


Figure 8.1: North-East position of the hexacopter

The measured and reference roll and pitch angles are shown in Figure 8.2. It can be seen from the plot that the measured roll and pitch is a bit delayed from the reference and that the measured overshoots the reference a bit.

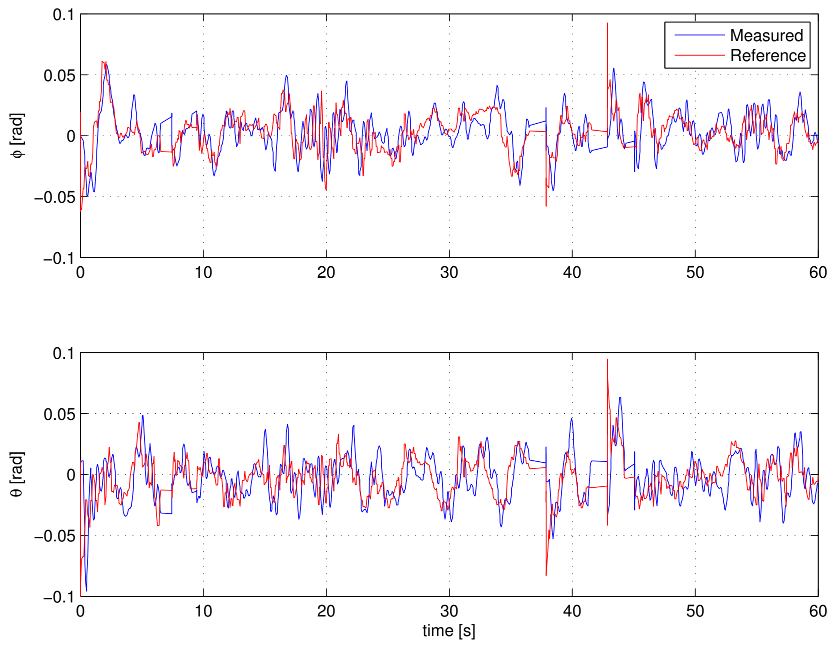


Figure 8.2: Attitude of the hexacopter

8.1.2 Discussion

The hexacopter is able to hover above the sensor node and maintain approximately the desired position above the sensor node. The position is accurate enough for the hexacopter to keep the node in the camera frame but the errors can be too big to be able to descend down upon the node. The pilot was in charge of the throttle in this test, and as the test was indoors it was difficult to maintain a stable height. This might have affected the results in a negative way.

The reference roll and pitch values seems quite noisy. This is mostly due to the derivative term of the controller and the low frame rate on the position measurements. The low level controllers in the APM struggles to follow the reference roll and pitch, but they are able to follow the main tendencies of the reference values. The measured values show that there is some overshoot in the low level controllers.

8.2 Descend Down to the Sensor Node

In this test the altitude controller is added to the system. The hexacopter starts on the ground next to the node position, then the altitude reference ramps up until the hexacopter is 1.5 meters above the ground. The position controller tries to maintain a position straight above the sensor node. When the hexacopter is sufficiently close to the desired altitude and position, the altitude reference is decreased until the hexacopter has descended all the way down to the sensor node.

8.2.1 Result

The position error of the hexacopter during the test is shown in Figure 8.3. As one can see from the figure the hexacopter is able to hover above the sensor node and keep it in frame while descending all the way down to the node and landing. On the altitude measurement one can see a spike in height a little after 5 seconds. This spike is an erroneous measurement.

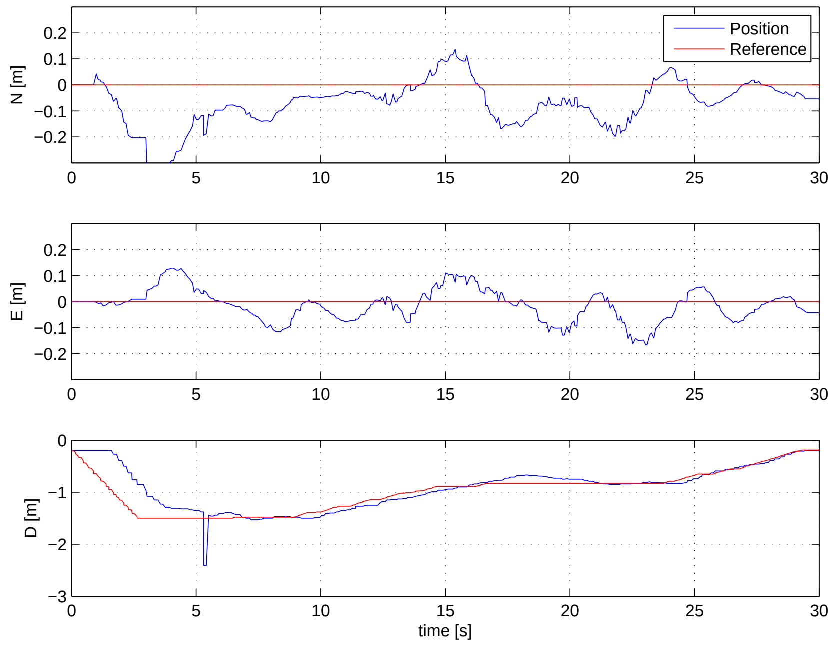


Figure 8.3: North-East-Down position of the hexacopter

The measured and reference roll and pitch angles are plotted in Figure 8.4. These results look quite similar to the ones from the hover test, with the difference of the reference roll and pith angles being less noisy.

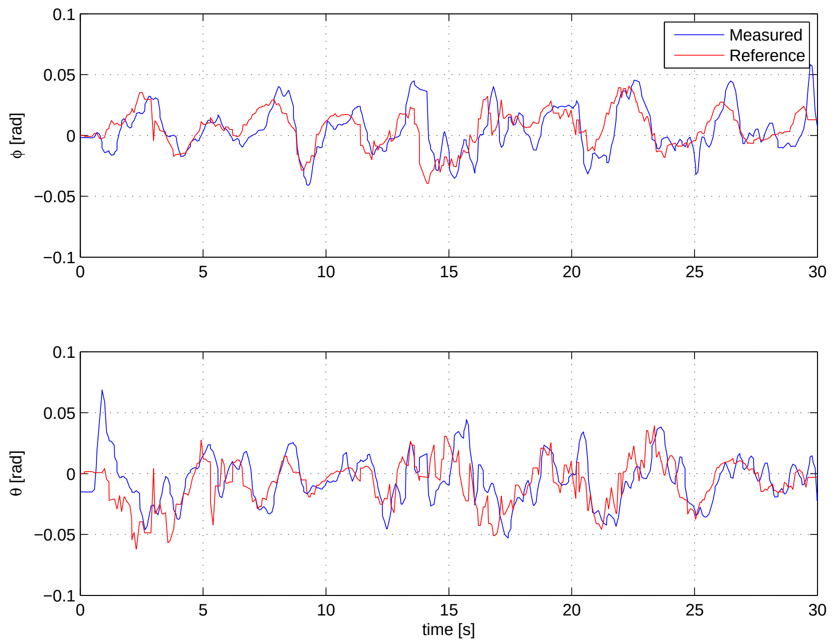


Figure 8.4: Attitude of the hexacopter

8.2.2 Discussion

The hexacopter is actually able to land on the sensor node, which is promising. But one can also see from the plots that there is an error in NE-position while descending. This error is not big, but there does not need to be a big error for the camera to loose sight of the sensor node in the last stages of the pickup phase.

The erroneous altitude measurement can be troublesome, it will lead to wrong position and speed calculations. The effect this error has on the position calculation can be easily spotted on the north position measurement in Figure 8.3. There can be several reasons for the sonar to measure wrong altitude. The sonar can be affected by electromagnetic noise from the ESCs, batteries or motors, it can be affected by the downwash from the motors or acoustic noise from the motors. Operating with high roll or pitch angles could lead to a much greater altitude measured due to the fact that instead of being reflected back to the sonar, the noise is sent off in another direction. Looking at roll and pitch angles at the specific moment of the bad measurement, one can see that the roll and pitch angles are kept small, eliminating the last explanation. The sonar was in this test mounted down below the hexacopter, which should minimize the risk of it being affected by noise. Other tests showed that such bad measurements occurred sporadically, but seldom for more than one sample. Hence one should consider implementing some logic in the APM to take care of and reduce the effects of this bad measurements.

One can see from the altitude plot that the desired altitude are unevenly ramped down towards the sensor node. The reason for this uneven ramp is that the ramping function uses acceptance parameters to see if the position of the hexacopter is sufficiently close to the desired position while ramping down. This is done to reduce the risk of loosing sight of the sensor node while descending.

8.3 Imitation of Pickup Phase

This test is executed in the same way as the previous test, except that instead of landing on the sensor node, the hexacopter is supposed to hover close to it (0.5 m above) for a sufficient amount of time for a pickup, and then ascend again. In that way a pickup operation is imitated. The ramp to reduce the altitude reference is implemented a bit differently in this test. In the previous test there were continuous checks to see if the hexacopter was in position straight above the sensor node while ramping down. In this test the ramp starts when the hexacopter is located sufficiently close to the position straight above the sensor node and the speed is sufficiently low, and then the ramp continues unaffected of the position of the hexacopter in accordance to the sensor node.

8.3.1 Result

The position of the hexacopter during this test is shown in Figure 8.5. The reference and measured roll and pitch angles plot looks quite similar to the previous ones, and is therefore not displayed.

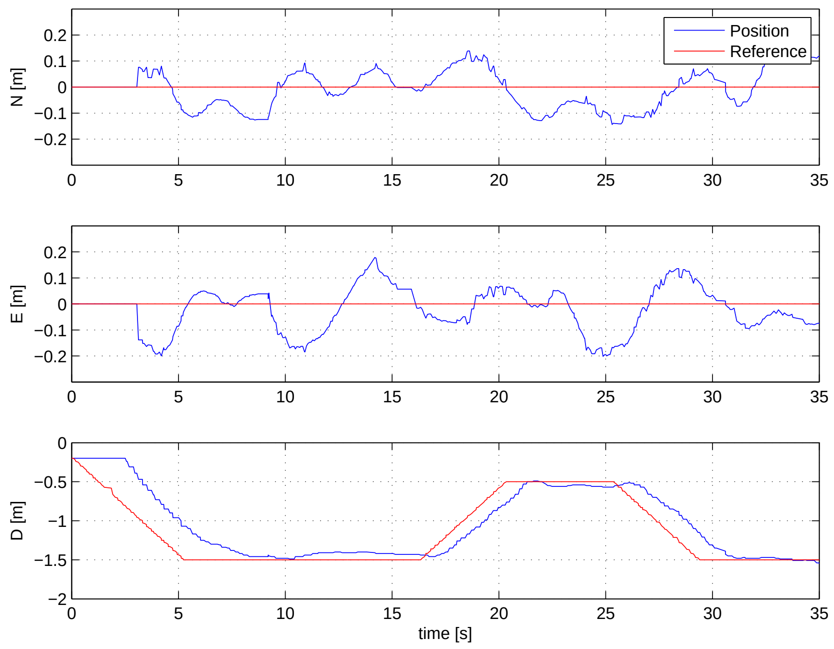


Figure 8.5: North-East-Down position of the hexacopter

8.3.2 Discussion

As one can see from the figure the hexacopter is more or less able to maintain a position above the sensor node. The use of a continuous descend ramp resulted in a smoother and faster response. But this approach also increases the risk of loosing sight of the sensor node while descending. One should consider to implement a check that takes the height above the sensor node into account when checking if the hexacopter is sufficiently close to the position straight above the sensor node. The hexacopter is able to hover 0.5 m over the sensor node for the desired amount of time, unfortunately the chosen pickup mechanism needs to be able to hover a lot closer.

8.4 Test of Drop and Recovery Mechanism

In this test the drop and recovery mechanism is mounted to the hexacopter. The operation tested is an pickup operation. Unfortunately due to to lack of accuracy in the last parts of the pickup phase of the hexacopter, this test is conducted while manually lifting the hexacopter and observing the operation of the drop and recovery mechanism.

8.4.1 Result

The hexacopter makes sure that the gripper is open and that the gripper platform is raised when it is switched into dune mode. The gripper platform is lowered when the hexacopter is starting to "hover" above the sensor node. When the hexacopter is lowered down to the position where the gripper can grip the mount of the sensor node, the gripper closes and the gripper platform are raised.

8.4.2 Discussion

The drop and recovery mechanism functioned flawless in combination with the PandaBoard and DUNE. It did exactly what it was supposed to do, when it was supposed to do it.

8.5 Hover Above the Sensor Node with Gripper Mounted

In this test the drop and recovery mechanism is mounted and the hexacopter is set to hover above the sensor node while the pilot has control of the throttle. The gripper platform is lowered while hovering. It is especially interesting to monitor the effect of the added mass and the effect of the gripper platform being lowered in this test.

8.5.1 Result

The position of the hexacopter while hovering above the sensor node is shown in Figure 8.6.

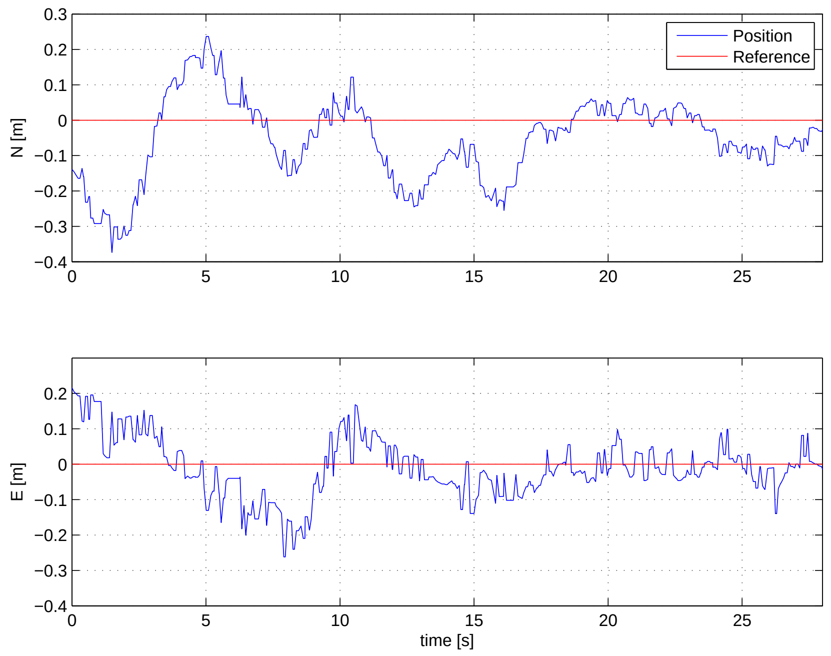


Figure 8.6: North-East-Down position of the hexacopter

The reference and measured roll and pitch angles plots are displayed in Figure 8.7. One can see from these plots that the reference roll and pitch angle values looks quite noisy. The reference roll and pitch values are strongly dependent on the measured velocity of the hexacopter. Hence plots of the measured velocity are included in Figure 8.8 for reference.

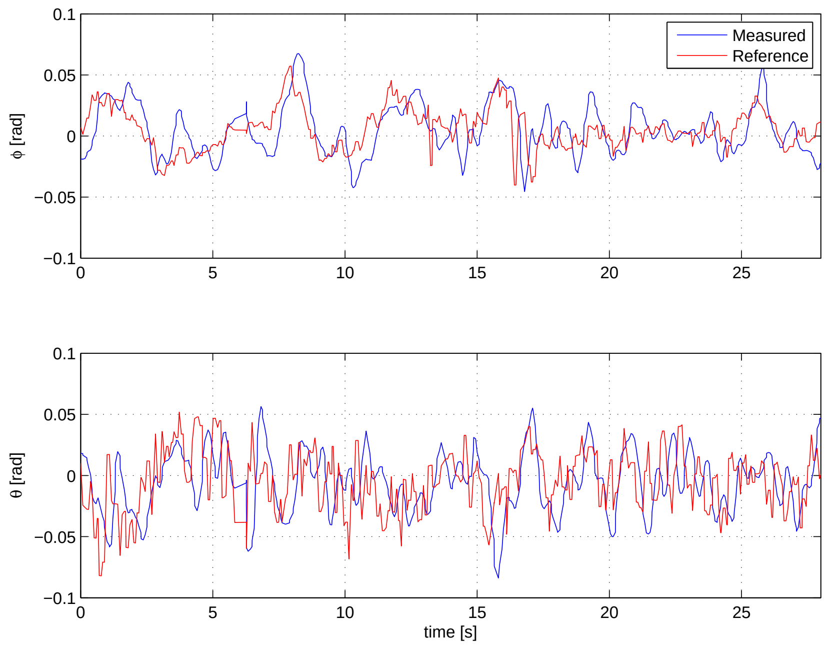


Figure 8.7: Attitude of the hexacopter

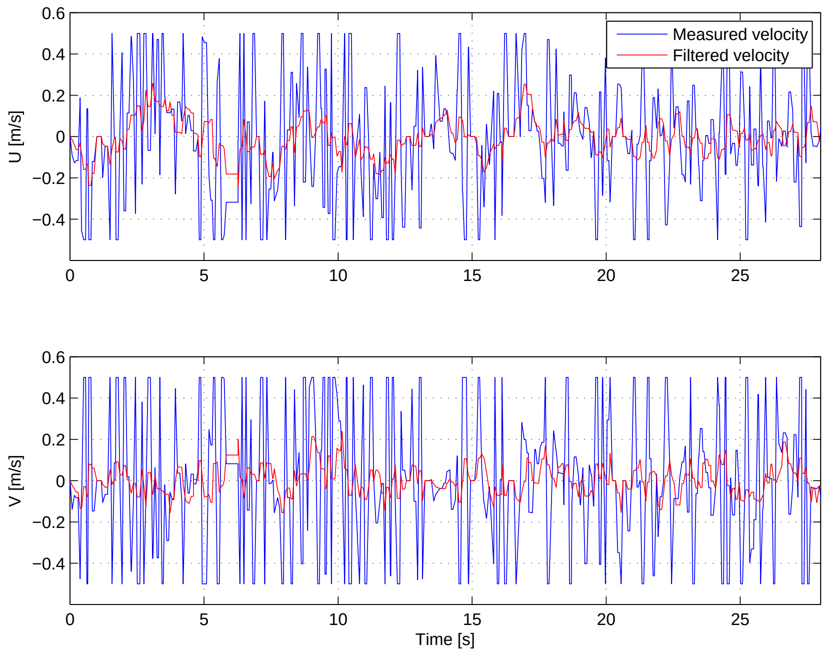


Figure 8.8: Velocity of the hexacopter

8.5.2 Discussion

The system is able to handle the added weight without problems. The low level controllers could be retuned to compensate for the fact that more force is needed to obtain desired roll and pitch angles, but it seems like the controllers are able to handle this in an OK fashion already.

The measurements of positional error looks more noisy in this test than in the previous tests. This test is executed with the gripper platform lowered and with the camera mounted on the bottom of this gripper platform. The compensation for this arm in the measurements are calculated using attitude measurements received from the APM. Hence bad or delayed attitude measurements will have a great impact of the position measurements. The noise on the position measurements is probably due to this effect, and because of this noise the velocity calculations becomes even more noisy. However it still looks like the filter is able to reduce the effect on the velocity measurements.

Chapter 9

Discussion

The different test scenarios have to some extent already been discussed. But the basis of the tests have been quite different, because of the variety of features to test and the different platforms the tests have been conducted on, i.e: simulation in Simulink, SIL-test and testing in the lab. This chapter aims to discuss the different results and take into account all the information gathered in the tests on the different platforms. Finally suggestions for future work are given.

9.1 Drop and Recovery Mechanism

The drop and recovery mechanism has mainly been tested in the lab test, and some additional tests were conducted in [Voldsund, 2013]. Both the camera application and the electrical and mechanical design of the drop and recovery mechanism will be evaluated.

9.1.1 Camera Application

The accuracy of the camera algorithm has been tested in [Voldsund, 2013] with good results. It has also been thoroughly tested in the lab as the measurements from the camera has been essential for all the tests. The camera algorithm has shown some weaknesses when combined with the PandaBoard and DUNE and some weaknesses related to the placement of the camera.

Using the PandaBoard and DUNE, a stable frame rate of only 10 Hz could be achieved. A higher frame rate would be desirable. The PandaBoard has a camera header using the

MIPI[1] CSI-2 standard. CSI-2 is a standard that provides a robust, scalable, low power and high speed interface for imaging solutions [MIPI, 2013]. 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 video streaming at 60 fps [e-con Systems, 2013]. Tests should be conducted to check whether this camera can deliver the promised frame rate in communication with a PandaBoard. A higher frame rate would reduce the delay of the position measurements and reduce the noise in the velocity measurements.

Another challenge with the camera application in combination with the mechanical design is the fact that the view of the sensor node is easily lost when the UAV is close to pickup. One measure that could be taken to help this situation is to use a camera with a greater field of view. Figure 9.1 shows how large area the camera frame will cover from a height of one meter and the diagonal field of view (DFOV) of the different cameras. The figure show that using another webcam can help with the node tracking in the pickup phase. The e-cam51_44x has a greater field of view than the used camera, which makes it even more interesting to check out. Some webcams with more extreme field of views does also exist. They are designed for the business marked and can have greater demands to the specification of the computer it is used with.

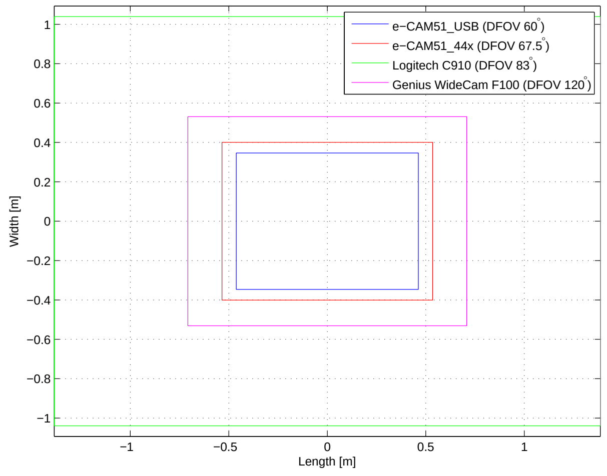


Figure 9.1: Camera frames of different webcams

There also exists some wide angle lenses intended for mobile phone cameras that can be attached outside the original lens. This can help the situation, and is a cheap and simple

[1]Mobile Industry Processor Interface

solution to the problem. One of these was ordered to be tested for the application, but unfortunately it got lost in the mail.

9.1.2 Mechanical and Electrical Design

Both the mechanical and electrical design is proven to be robust and reliable. But there are a few drawbacks with the design of the drop and recovery mechanism that makes pickup operations troublesome.

The gripper need very accurate control of the UAV to be able to pick up a sensor node. This kind of precision was unfortunately not achieved in this project. The placement of the camera is an issue that already has been discussed, and some possible solutions have been mentioned.

Another point that can be troublesome, is the fact that in the pickup phase the UAV is hovering approximately 40 cm above the sensor node. Testing in the lab revealed that the downwash from the propellers had a tendency to move the dummy sensor node used in the experiment. In real operations the sensor node will be floating partly submerged in water. Hopefully the water resistance will limit the effect from the propeller downwash on the sensor node. One could also consider to use longer rack gears to increase the hovering height for pickup.

9.2 Control of the UAV

The simulations showed that the position controller was accurate enough to control the UAV during the pickup phase depending of the amount of disturbances acting upon the system. The controller was able to handle constant disturbances in a good way, but got problems with strong and varying disturbances. The most realistic operation environment is relatively nice weather conditions with little wind where the wind comes from a nearly constant direction. The sensor node might be affected by some currents and/or propeller downwash. The controller seems to be able to handle this operation environment.

The SIL test showed that the control structure was able to execute pickup operations both with and without drift added to the SIL test. The SIL test showed that the control of the

APM from DUNE should function well. It showed that the low level controllers of the APM had an overshoot, but was in general able to follow the desired roll and pitch angles fed to the controller from DUNE.

When testing the control structure in the lab, it became clear that the control structure had some challenges. An update rate of 25 Hz between the APM an the PandaBoard was achieved. The fact that the updates had to be sent over the serial link meant that a small delay was introduced to the system. This delay was probably most troublesome for the camera algorithm when the camera was mounted to the bottom of the gripper platform as the position calculations was very dependent upon roll, pitch and yaw angles due to the long arm of the gripper.

The 25 Hz update frequency should be high enough for desired control inputs sent to the APM, but due to the low frame rate (10 fps) of the camera, the update frequency of the controller is in reality only 10 Hz. One solution to this problem is to control the UAV with only one device instead of two devices communicating with each other. The APM was used in this project because it is a convenient way to get started fast, as all the sensors and actuators already are interfaced and supported. There exists a more powerful flight control solution running the same software as the APM. This solution is the Pixhawk from 3DR which can handle more functionality (the APM is already running at its limits). Another solution that could be very interesting to explore further is the BeaglePilot project[2]. The aim of the BeaglePilot project is to make an autopilot for Linux-based computers (more specifically the BeagleBone Black (BBB)). They are trying to port the ArduPilot project to be run on the BBB, handling sensor and actuator communication with real time demands. They have also designed a sensor cape for the BBB, the Pixhawk Fire Cape. The fact that this is supposed to run on a Linux-based computer means that it would be possible to run DUNE alongside the ported APM software. Hence it might be possible to use the developed control structure with the BeglePilot project without to much hassle. This project is work in progress, but would be very interesting to follow.

[2]ardupilotbeaglebone.wordpress.com

9.3 Ability to Execute Sensor Node Drop and Recovery Operations

The drop and recovery mechanism functions flawlessly. The main concern is the accuracy in the UAV it demands to function efficiently. The accuracy is more than good enough for drop operations. The main challenge is the pickup operation. In the previous sections the main problems related to pickup using the developed system were discussed. A few changes that it would be interesting to explore further to make the system able of both drop and recovery of sensor nodes were also proposed.

The SIL test of the drop and recovery operation showed that the control structure is able to execute such missions.

9.4 Future Work

Several measures need to be taken to make the system able to execute drop and recovery operations using the proposed drop and recovery mechanism. These measures and a few suggestions to further development and improvements of the system is presented in the following list.

- A camera with a greater field of view should be tested.
- Measures to increase the frame rate from the camera should be further explored. A good place to begin these explorations is to check out the e-CAM51_44x camera, which is especially designed for use with PandaBoard and according to the producer is able of 60 fps.
- To decrease delay and increase the rate of the controllers and measurements, solutions where only one device is use to control the UAV should be explored. The BeaglePilot project could be a great starting point.
- A console for Neptus where missions could be planned, monitored, executed and reviewed should be developed. This could for instance include a map where one could point and click to set drop, pickup and land locations. A camera stream from the camera searching for the sensor node could also be included.

- The heading controller of the ArduCopter code is not stable enough for tracking heading references without creating undesired effects. This should be looked into to make the UAV able to align itself with the sensor node for pickup.
- A search pattern should be developed to make sure that the sensor node is found during pickup as the waypoint for pickup will be an approximate location.
- An algorithm to regain pickup operations when sight of the sensor node is lost should be developed.
- Object tracking less dependent of calibration can be developed, either by using some other tracking principle or by including an adaption algorithm to the tracking.

Chapter 10

Conclusion

The goal of this project was to develop a system able to conduct drop and recovery of sensor nodes by the use of a multicopter UAV. A mechanism to facilitate drop and recovery operations had been developed in an earlier project by the author [Voldsund, 2013]. The mechanism has been further developed and tested in this project.

The mechanism developed has turned out to be a robust and reliable system able of conducting both drop and recovery operations. However it also puts quite high demands to the required accuracy of the UAV in the pickup phase. This accuracy demand is not entirely met in this project, but the designed control structure has showed in both simulations and in SIL tests that it can be able to execute drop and recover under the right circumstances.

The desired accuracy in the pickup phase was not achieved during testing in the lab. There are several reasons for this, and different measures that can be taken to help on the problem, the most important measures are given below.

One of the main problems was due the placement of the camera. The camera is placed on the bottom of the gripper platform, which means that in the end of the pickup phase, the camera might loose sight of the sensor node. To reduce the risk of this, a camera with a greater field of view can be used. In addition measures that increase the precision of the UAV will reduce these problems as the ability to hover straight above the sensor node will be increased.

To be able to increase the precision of the UAV, one can use a solution where only one device is used to control the UAV, instead of the two devices that is used in this project (APM and PandaBoard). This will reduce the delay in the system, and better control over also the low level controllers is gained.

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.

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

Fernando, S. (2104). Opencv tutorial c++ - color detection & object tracking. `opencv-srf.blogspot.no/2010/09/object-detection-using-color-seperation.html`.

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

Franklin, W. R. (2014). Point inclusion in polygon test. `www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html`.

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/`.

Jiang, G. and Voyles, R. (2013). Hexrotor uav platform enabling dextrous aerial mobile manipulation.

Lippiello, V. and Ruggiero, F. (2012). Cartesian impedance control of a uav with a robotic arm. In *10th International IFAC Symposium on Robot Control*.

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.

MaxBotix (2014). Mb1240 xl-maxsonar-ez4 high performance ultrasonic rangefinder. `www.maxbotix.com/Ultrasonic_Sensors/MB1240.htm`.

Mellinger, D., Lindsey, Q., Shomin, M., and Kumar, V. (2011). Design, modeling, estimation and control for aerial grasping and manipulation. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2668–2673.

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`.

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 (2013a). About opencv. `opencv.org/about.html`.

OpenCV.org (2013b). 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`.

Pounds, P. and Dollar, A. (2014). Aerial grasping from a helicopter uav platform. In Khatib, O., Kumar, V., and Sukhatme, G., editors, *Experimental Robotics*, volume 79 of *Springer Tracts in Advanced Robotics*, pages 269–283. Springer Berlin Heidelberg.

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

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`.

Thomas, J., Polin, J., Sreenath, K., and Kumar, V. (2013). Avian-Inspired Grasping for Quadrotor Micro UAVs. In *IDETC/CIE*. ASME.

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

Veness, C. (2014). Calculate distance, bearing and more between latitude/longitude points. `www.movable-type.co.uk/scripts/latlong.html`.

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.

Acronyms

ADC	Analog to Digital Converter
AMOS	Centre for Autonomous Marine Operations and Systems
APM 2.6	Ardupilot Mega 2.6
BBB	BeagleBone Black
BGR	Blue-Green-Red
CEP	Circular Error Probability
CPU	Central Processing Unit
DFOV	Diagonal Field of View
DH Convention	Denavit-Hartenberg Convention
ECEF	Earth-centered Earth-fixed
GPS	Global Positioning System
HSV	Hue-Saturation-Value
IC	Integrated Circuit
IMC	Inter-Module Communication protocol
IMU	Inertial Measurement Unit
I/O	Input/Output
IR	Infrared Radiation
ISA	Inertial Sensor Assembly
ISP	In System Programming
LBP	Local Binary Patterns
LED	Light Emitting Diode
LSTS	Laboratório de Sistemas e Tecnologias Subaquáticas
MAVLink	Micr Air Vehicle Protocol
MEMS	Microelectromechanical Systems
MIPI	Mobile Industry Processor Interface

NED	North-East-Down
OpenCV	Open Source Computer Vision Library
PWM	Pulse-Width Modulation
RAM	Random Access Memory
SIFT	Scale-Invariant Feature Transform
SIL	Software-in-the-Loop
SURF	Speeded-Up Robust Features
UART	Universal Asynchronous Receiver/Transmitter
UAV	Unmanned Aerial Vehicle
WGS-84	World Geodetic System 84