NTNU – Trondheim
Norwegian University of
Science and Technology

POSITION CONTROL FOR AUTOMATIC LANDING OF UAV IN A NET ON SHIP

Kjetil Hope Sørbø

Submission date: December 2015
Responsible professor: Thor Inge Fossen, Tor Arne Johansen

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

Automatic landing of a fixed wing Unmanned Aerial Vehicle (UAV) in a net on a ship require an accurate positioning system. There exist today high-end systems with such capability for special applications, e.g military systems and costly commercial systems. To increase general availability the system must consist of low-cost components. An alternative is the use of low-cost Global Navigation Satellite System (GNSS) receiver and apply Real Time Kinematic GPS (RTK-GPS), which can provide centimeter level position accuracy. However the processing time for the Real Time Kinematic GPS (RTK-GPS) system results in degraded accuracy when exposed to highly dynamical behaviour.

This work present two alternative software and hardware position systems suitable for use in navigation system which apply RTK-GPS, namely Real-Time Kinematic Library (RTKLIB) with a Ublox Lea M8T receiver and Piksi. Both the Piksi and the Ublox receivers are single-frequency GNSS receivers. The Piksi supports RTK-GPS, while the Ublox can send raw GNSS data to RTKLIB. The RTKLIB version used in this work is an altered version from the latest release. These systems will in this work be compared and their individual capability to provide accurate position estimate will be evaluated.

The RTK-GPS system is implemented in DUNE(DUNE:Unified Navigation Environment) framework running on an embedded payload computer on-board the Unmanned Aerial Vehicle (UAV).

The performance of these position systems are in this work investigated by experimental testing. These tests of the navigation system was successfully performed, and have proven that the navigation system is sufficient from integration into a control and guidance system.

Of the two positioning system the RTKLIB combined with the Ublox LEA M8T receiver proved superior to the Piksi system.

Contents

Acronyms

DGPS Differential GPS.

DOP Dilution Of Precision.

ECEF Earth-Centered-Earth Fixed.

ECI Earth-Centered-Inertial.

ENU East North Up.

EPO Expanded PolyOlefin.

GLONASS Global Navigation Satellite System.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

GPST GPS Time.

IMC Inter-Module Communication.

INS Inertial Navigation System.

L1 Link 1.

L2 Link 2.

L5 Link 5.

LAMBDA Least-squares AMBiguity Decorrelation Adjustment.

LSTS Underwater Systems and Technology Laboratory.

NED North East Down.

RTK-GPS Real Time Kinematic GPS.

RTKLIB Real-Time Kinematic Library.

SHF Super-High Frequency.

SIL Software In the Loop.

TOW Time Of Week.

UAV Unmanned Aerial Vehicle.

UHF Ultra-High Frequency.

UTC Coordinated Universal Time.

WGS-84 World Geodetic System, 1984.

Chapter 1

Introduction

1.1 Background and motivation

Recent development of flying UAVs have been recognized to provide an attractive alternative to work previously performed by manned operations. Typical work which has attracted attention includes inspection, aerial photography, environmental surveillance and search and rescue. Today UAVs are mostly operated over land, however in the future this will include over sea as well. This will give some challenges which must be overcome. One of these challenges is that the UAV need to be able to perform a autonomous landing.

An UAV can provide an attractive alternative for many maritime operation where today manned aircraft or satellites is the only solution. In the maritime sector UAV can be used in iceberg management, monitoring of oil spills, search and rescue and maritime traffic monitoring.

An important premise for successful and safe UAV operation, in particular at sea, is the provision of a robust system for safe landing of the UAV on a vessel following completed operations. In order to perform an automatic landing a path planner, a guidance system, and an accurate position estimation system is required, this in addition to the low level control system in the UAV.

Existing landing system can guide the UAV towards a net, but they are expensive and restricted to a few UAVs. A pilot can land the UAV, however a better alternative would be if the UAV could land it self by the use of a net. In order to make the UAV able to perform a automatic landing a minimum requirement is that it knows its position at any time. This will require an accurate position estimation in real time. A highly accurate position sensor is expensive, however it's possible to achieve accurate solution with low cost sensors. This can be done by combining two GNSS receivers to estimate the relative position of one of the receivers in respect of the other receiver highly accurately.

The path planer and the guidance system used as basis for this work were developed as previous master thesis work [Marcus, 2015]. A key component in the guidance system is the position estimation system, which for the guidance referenced was developed by [Amstrup, 2015]. However, this system has been concluded to be in-sufficient with respect to provide means required for automatic landing.

Automatic landing in a net has privously been successful performed in the NTNU MSc thesis [Robert and Lie, 2014]. They managed to land a UAV in a stationary net using RTK-GPS. However only 50% of the landing attempt was successful.

An other successful automatic landing was done in the Stellenbosch University MSc thesis [Adriaan, 2013] using Differential GPS (DGPS). This MSc thesis gives a description on the control system required to perform an automatic landing, however the system in the thesis require a runway in order to land.

This project work will continue the research done by Spockeli [Amstrup, 2015], and use a new GNSS receiver that will be used together with the open source program, RTKLIB. The goal is to establish a system with accurate local position estimate, such that in the future a UAV net landing can be performed automatic. The navigation system must be accurate enough to correctly estimate if the UAV is following the generated landing path or if the deviation from the path is large enough such that an evasion manoeuvre is required. The position evasion criteria used in [Marcus, 2015] is $\pm 1m$ cross-track error and $\pm 1m$ altitude error. The automatic net landing system will use RTK-GPS for relative position estimation. The main goal for this work is to describe the gaps of the available position system sufficient to scope further work required for closure of such gaps ultimately providing means for position estimating sufficient for completion of automatic landing.

This work will be done at the UAVLAB, which is a research lab at NTNU. The UAVLAB is a test facility for software and hardware, which include inertial navigations system,global satellite navigation systems and unmanned aerial systems.

1.2 Literature Review

An alternative low-cost system that can be used for automatic landing is vision based landing system. In the paper [Jin et al., 2013] a vision based landing system, which would detect the recovery net, and plan a landing path is proposed. The system was successfully tested and is a valid alternative for a low-cost autonomous landing system. An other vision based landing system was proposed in [Paul and Michael, 2012]. This paper describes an intelligent vision aided landing system that can detect and generate landing waypoints for a unsurveyed airfield. The drawback with a vision based landing system is that it require much computational power. In addition the visual line of sight can quickly decrease during a operation.

Real Time Kinematic GPS (RTK-GPS) apply phase measurement of the GNSS signal for position estimation. In order to get a accurate position estimate the integer ambiguity must be resolved. An integer ambiguity resolution strategy was proposed in [Geoffrey, 1989], and demonstrated centimeter level accuracy for a baseline up to $2000km$. Further studies on integer ambiguity resolution strategies resolved in the Least-squares AMBiguity Decorrelation Adjustment (LAMBDA) strategy, which was proposed in [P.J.G., 1994], and further disused in [P.J.G., 1995, Teunissen P.J.G and C.C.J.M., 1995]. The Least-squares AMBiguity Decorrelation Adjustment (LAMBDA) has been wildly used, and has proven a quick strategy to resolve the integer ambiguity, which makes it ideal for RTK-GPS systems.

The use of RTK-GPS for accurate position estimation has been studied in [W., 2013]. The paper proposed how to create a low-cost RTK-GPS system that can accurate measure the position in 3D. They used raw GNSS data from the GNSS receiver and the program library Real-Time Kinematic Library (RTKLIB) to estimate the position of a trolley in real time.

In the paper [W. and M., 2011] it was studied high precision positioning of micro-sized UAV using RTK-GPS. The system used a GNSS receiver that used a ground based augmentation system as a base station. The use of a ground based augmentation system us advantageous if the UAV can communicate with the ground station. For UAV operations where information from a ground based augmentation system is not available a local reference station must be considered.

1.3 Scope of work

The scope of work is to validate the performance of suitable positioning systems by study and testing, and to identify gaps required to be closed for successful implementation for a integrated autonomous Guidance, Navigation and Control system which will allow for automatic landing of a UAV at sea. This will include: The scope of this work is to test the performance of ublox-LEA M8T GNSS receiver

Testing of the performance of Ublox LEA M8T
Compare the performance of RTKLIB and Piksi
Compare the real time estimate with the post processed estimate

in a real time differential Differential GPS (DGPS) configuration. The RTK-GPS solution will be calculated with the open source program RTKLIB, which will communicate with a task in DUNE. The solution from RTKLIB will be compared against the solution from Piksi, and a post processed solution from RTKLIB. The result from the experiment will be used in the discussion on how to perform an automatic net landing.

1.4 Layout

This section is currently not up to date

Chapter 1 Intro

Chapter 2 Theory about coordinate system

Chapter 3 Theory about GNSS system with focus on the GPS

Chapter 4 Hardware and software

Chapter 5 Test and result

Chapter 6 Conclusion, discussion and further work

Chapter 2

UAV Navigation System

This chapter will explain the UAV navigation Real Time Kinematic GPS (RTK-GPS) system, as well as how it may be used in an automatic landing system. The first section include an overview of the system, including a description of the current system. The next part will contain different reference frame used in the RTK-GPS module. Then the different Global Navigation Satellite System (GNSS) system will be presented as well as atributes in the Global Positioning System (GPS). Then a quick overview over the different error sources that can affect a GNSS system, before the concept Differential GPS (DGPS) is presented. Lastly the accuracy of the term RTK-GPS will be explained.

2.1 System Layout

An integrated system required for automatic landing will typically consist of four sub-systems as shown in figure 2.1. Today these systems are individually available or in development; however not integrated into a proven working system allowing for automatic landing of UAVs. The four main systems comprises of the navigation part, the guidance part, the control part and the user interface.

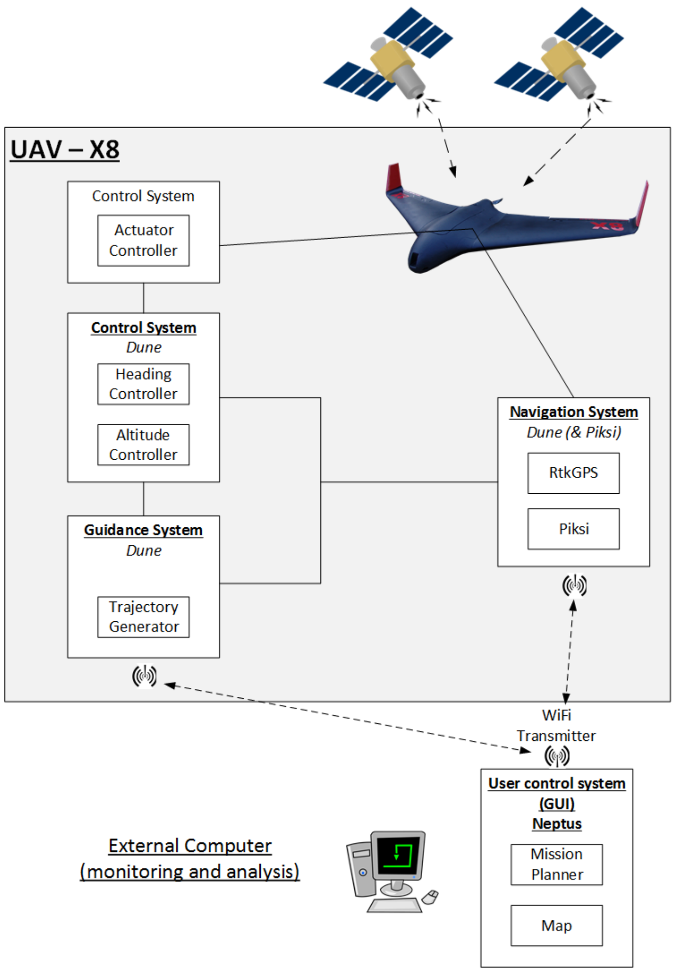


Figure 2.1: Overview of the automatic net landing system

The navigation system apply RTK-GPS to estimate the relative position of the UAV. More details of the RTK-GPS is given in section 2.6.

The control and guidance system has currently only been tested in Software In the Loop (SIL) simulations with Ardupilot, which shows promising result likely to be sufficient for automatic landing applications. However this module has not yet been implemented to support the use of RTK-GPS as required for performing automatic landing.

The RTK-GPS solution is compute by two different system ,which will be compared against each other, namely Piksi by Swift Navigation and RTKLIB by T.

Takasu. The latter is independent on the type of receiver, and thus will be the main focus of this work. More detail about Piksi and Rtklib will be given in 3.1.3 and 3.2.2.

The current state of the system is that it consist of two parts that has not yet been integrated with each other. The main focus of this project work is the positioning part. The plan is to use RTK-GPS for positioning estimation. The second part is the guidance and control. Currently there has only been done simulation of the guidance system. It shows promising results, but is yet to be integrated with RTK-GPS. Figure 2.1 gives a overview of the different modules in the system.

2.2 Reference frames

A GNSS system calculate an estimated position estimated on the Earth where the receiver is. For the position to have any meaning a reference system has to be defined where a frame can be consider as an inertial frame. Based on this a reference frame can be fixed to the Earth rotation, but for local navigation it must be referred to surface of the Earth. This section will present the different reference frame used in global navigation systems.

2.2.1 ECI

Earth-Centered-Inertial (ECI) frame is considered a inertial frame for terrestrial navigation. The origin is fixed in the center of the Earth, and the axis is fixed in space. This frame can be considered as an non-accelerating where Newton's laws of motion applies. This is suitable for control system applications as Newtons laws on equation of motion can be used to model the system for control applications.

2.2.2 ECEF

The Earth-Centered-Earth Fixed (ECEF) coordinate system is defined with the origin in the center of the Earth with it's x-axis point toward the intersection between the Greenwich meridian and Equator (0 deg longitude, 0 deg latitude). The z-axis points along the Earth's rotational axis, and the y-axis complete the right handed orthogonal coordinate system. The Earth-Centered-Earth Fixed (ECEF) system can be represented in either Cartesian coordinates (xyz) or ellipsoidal coordinates (longitude, latitude and height). The ECEF frame rotate relative to the ECI frame at an angular velocity of $\omega_e = 7.2921 \times 10^{-5} rad/s$, where ω_e is the Earth rotation. Due to the relatively low rotation speed some system can consider the ECEF frame as inertial, which can simplify the equation of motion for a control system.

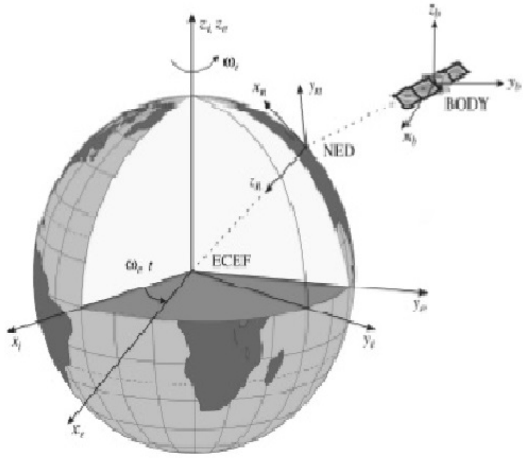


Figure 2.2: The ECEF frame. Picture from [I, 2011]

2.2.3 Local reference frame

The North East Down (NED) and East North Up (ENU) frame is defined as relative to the Earth reference ellipsoid (World Geodetic System, 1984 (WGS-84)). For the NED frame the x axis points in the direction to the true North, y axis towards East while the z axis points downward to completed the right handed orthogonal coordinate system. The ENU has the x and y axis exchange place with respect to the NED frame, and the z axis point upwards instead of downwards.

2.3 Global Navigation Satellite Systems

There are currently two operational GNSS constellations with global coverage, American Global Positioning System (GPS) and Russian Global Navigation Satellite System (GLONASS). Other GNSS constellations that will be operational in the near future is the Chinese BeiDou and European Galileo.

The GPS satellites transmits continuously using two radio frequencies in the L-band referred to as Link 1 (L1) and Link 2 (L2). The L-band covers frequencies between 1 GHz and 2GHz, and is a subset of the Ultra-High Frequency (UHF) band.

The receiver needs at least four satellite to be able to estimate the receiver position. Three of the satellite is used for the position, and the fourth is used to calculate the receiver clock bias. The position is calculated in the ECEF reference frame 2.2.2. The geometry of the satellite constellation affect the accuracy of the position estimation. Poor geometry increase the effect of error in the GPS signal.

There are two basic ways to measure the GPS signal to estimate a position, which is code and phase measurement. Of the two phase measurment is the most accurate, however also the least reliable due to the integer ambiguities. In code measurement the information in the GPS signal is used to calculate the psudorange between the receiver and the satellite. The psudorange is the geometric range from the receiver to the satellite in addition to the delay introduce from various error sources. In phase measurement the receiver measures the phase of the GNSS satellite and compares it against a receiver generated signal. Then by knowing the phase and the frequency function, as well at the start and end epoch time the geometric range between the receiver and satellite can be estimated. It's phase measurement that is mostly used in RTK-GPS. In phase measurement it's advantageous to know the unknown number of whole cycles between the satellite and the receiver, referred to as integer ambiguity. Estimation of integer ambiguity is called integer ambiguity resolution.

Coordinated Universal Time (UTC) is standard time on the Earth of which all clock are referred to. UTC time is defined with the use of atomic clocks, which is also used in GPS. However the GPS apply an other time standard namely the GPS Time (GPST). The GPST is also based on atomic clock in the satellites. Due to the distance from the Earth the GPST deviates from the UTC time. To correct this the GPS keeps track of the offset between GPST and UTC. The offset is included in the GPS message as leap seconds to be added to the GPST. GPST can be given as Time Of Week (TOW), which include the number of weeks since 1980-01-06T00:00Z. TOW is given in seconds, and is reset each week when the week number is incremented. More information about the GPS can be found in [Bjørnar, 2014, Pratap and Per, 2011]

2.4 Error sources

In order to get high accuracy in the position estimation the different error sources must be identified and removed if possible. This section will identify some of the most significant error sources that can affect the GPS signal, and how to remove or mitigate them in the estimation.

2.4.1 Clock error

There is drift in both the satellite clock and the receiver clock. The atomic clock in the satellites makes the clock drift negligible from the user perspective. The receiver clock tend to drift, and if not taken into account will cause large deviations in the position estimate from the true position. This error is remove by including a fourth satellite in the position computation. The satellite clock error given in the satellite message.

2.4.2 Ionospheric and Trophospheric Delays

When the GPS signals travel though the atmosphere there will be a delay caused by the different atmospheric layers.

Ionospheric delay

Gas molecules in the ionosphere becomes ionized by the ultraviolet rays that is emitted by the sun, which release free electrons. These electron can influence electromagnetic wave propagation, such as GNSS signals. The delay that the single get from the ionosphere may cause a error the the order of $1 - 10 meters$. The error can be mitigated by using a double frequency receiver, or by applying a mathematical model to estimate the delay. Both those cases is with a single receiver, however by including a second receiver the GNSS solution system can assume that both receiver receive signal in the same epoch, which means that the signals have experienced the same delay. This will be further explain in 2.5.2.

Tropospheric delay

The tropospheric delay is a function of the local temperature, pressure and relative humidity. The delay can vary from 2.4meters to 25 meters depending on the elevation angle of the satellites. The error can be mitigated by applying a mathematical model to estimate the tropospheric delay, and by using a elevation mask can remove all satellites with a elevation angle bellow a certain threshold. Error caused by tropospheric delay can be removed in the same manners as ionospheric delay when using two or more receivers. This will be further explain in 2.5.2.

2.4.3 Ephemeris Errors

A satellite is not able to perfectly follow a given orbit, therefore there will be a deviation between satellite position given to the receiver and the true position of the satellite. This is called the ephemeris error. The true position of a satellite is monitored and corrected by the owner of the GNSS constellation, but error between each correction can be expected.

2.4.4 Multipath

One of the primary source of error in in a GNSS receiver is multipath. Multipath happens when the satellite signal is reflected by a nearby surface before if reach the GNSS antenna. The delay introduced in the signal can make the receiver believe that its position is several meters away form its true position. The easiest way to mitigated multipath is to place the antenna at a location with open skies, and no tall structures nearby.

2.5 Differential GPS

Differential GPS (DGPS) consist of at least two receivers, where one is called a base station and the rest rovers. The two receivers are within range of a communication channel over which they are communicating, as shown in figure 2.3. The base station has a known position, and the rover estimate the baseline from itself to the base station.

In this project work only phase measurements will be considered for position estimation method. The problem with precise relative positioning with phase measurement is problem of estimation of integer ambiguities.

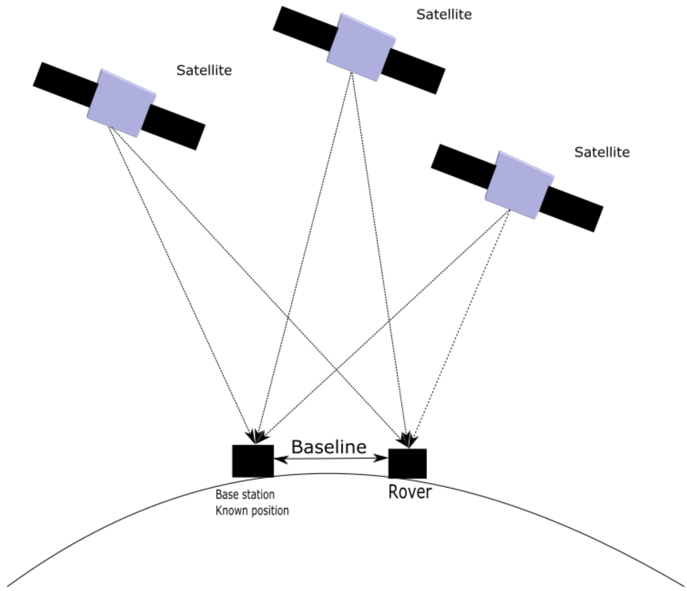


Figure 2.3: Concept figure of Differential GPS (DGPS)

2.5.1 Cycle slip

When the integer number of cycles experience a sudden jump in value due to loss of lock of a receiver phase lock it's called cycle slip. The effect of cycle slip is a bias in the measurement large enough to make navigation difficult. The effect of cycle slip will appear as seen in figure 2.4.

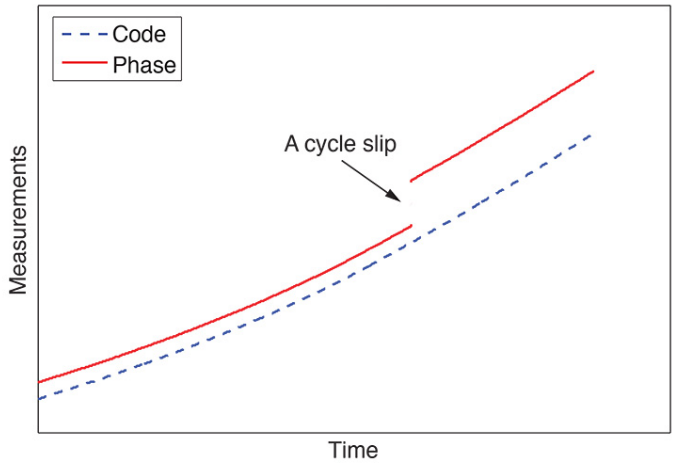


Figure 2.4: Cycle slip. Picture from http://gpsworld.com/wp-content/uploads/2014/01/I-Fig1.jpg

2.5.2 Error mitigation in DGPS

In DGPS the rover considers that both the rover and base station receive GPS signal that has experienced the same delay. Thus the rover can remove all error sources that is correlated with the base station. This assumption holds given that the baseline is not to long. For longer baseline other methods must be applied to correct for atmospheric delay. DGPS error mitigation do not include multipath. Multipath is an uncorrelated error, and thus must be corrected locally by both the rover and base station.

2.6 RTK GPS

Real Time Kinematic GPS (RTK-GPS) is a variant of DGPS system where both the base station and rover receives GPS signals and the distance between is calculated accurately. The distance between the rover and base station is referred to as a baseline.

RTK-GPS can either provide a kinematic setting or a moving baseline setting. The difference between the two is that in kinematic the base station has a known stationary position, while in moving baseline the base station position is unknown and allowed to move. The unknown base station is calculated with a single receiver, with the accuracy that entails. Since the base station position is calculated with a single receiver the RTK-GPS system with moving baseline can never have better global accuracy then what it will get with a single receiver. The advantage with the moving baseline configuration is that RTK-GPS can be used to find the relative position between two dynamical system using GNSS in real time. This will be the case in automatic ship landing system, where the base station is on a ship thus must be allowed to move. The advantage with kinematic mode is that it can give a more accurate position estimate. The relative position of the rover can be given in either the NED or enu frame.

RTK-GPS systems needs to calculate the position estimate quicker then a standard system. This is done by sacrificing the correctness of the solution, however in the resent time efficient software and hardware has reduced the this factor.

Chapter 3

System Components and implementation

This chapter will present the different software and hardware components used in this project work, in addition to how they are implemented and connected to each other. The two first sections in this chapter will present the hardware and software components respectfully. This is followed by a overview over the components in the UAV and base station that is used to create the RTK-GPS system. The two last chapters will explain how the software and hardware implementation respectfully.

3.1 Hardware

The different hardware components used in this work is a UAV, payload computer, GNSS receiver and antenna.

3.1.1 UAV

The UAV used is a Skywalker X8, see figure 3.1, which is a fixed wing Unmanned Aerial Vehicle (UAV) that is moulded out of Expanded PolyOlefin (EPO), which makes it a cheap and robust platform for prototype testing and modifications. The large space within the fuselage makes it ideal for experimental payload and projects with modest requirements.

Figure 3.1: X8 Skywalker from Skywalker Technologies, Picture from www.campilot.tv/blog/win-x8

The X8 has a wingspan of $2120mm$ which allow for a Maximum Take-Off Wight of $4.2kg$, including $1kg$ for the payload. The X8 used in this work is outfitted in according to the specification given in [Zolich A. P. and K., 2015]. The components that is used in the X8 at the UAVLAB is given in table 3.1.

Sensors	3DR ublox GPS with Compass Kit
	Pixhawk Airspeed Sensor Kit
Servo	Hitec HS-5125MG
Motor	Hacker A40
ECU	Jeti Master Spin 66 Pro
Main Battery	1 Zippy Compact 4S 5000 mAh
Autopilot	3DR Pixhawk
Primary digital data link	Ubiquiti Rocket M5

Table 3.1: Components in the X8 at the UAVLAB

3.1.2 Embedded Computer

The embedded computer chosen as payload in the X8 is a Beaglebone Black, shown in figure 3.2. The Beaglebone was chosen because of its sufficient computation power, low energy consumption and variety of communication interfaces. For ease interface access the the Uavlab at NTNU has developed an extension board called "CAPE".

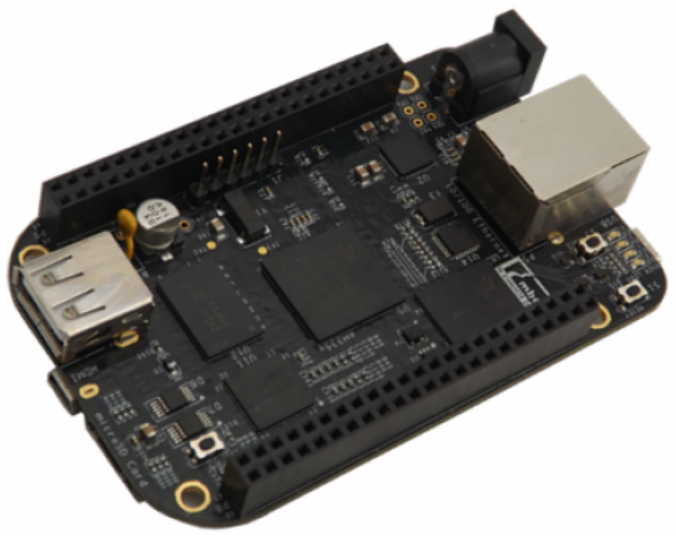


Figure 3.2: BeagleBone Black element 14, Picture from http://www.element14.com

The Beaglebone black supports linux operation systems, and it is compatible with the Underwater Systems and Technology Laboratory (LSTS) toolchain, which will be return to in 3.2.

3.1.3 GNSS receiver

The system will use two types of GNSS receivers, namely the Ublox LEA M8T and the Piksi system.

The Ublox LEA M8T is a new generation of low-cost GNSS receiver from ublox. The receiver support sending out raw GNSS data from both GPS and GLONASS in the same configuration. The receiver have great performance in acquisition and tracking sensitivity of GNSS satellites. More technical details can be found in [Ubl, a,b].

Figure 3.3: Ublox LEA M8T, Picture from http://www.csgshop.com

Piksi

The Piksi system is a low cost, high performance GPS receiver with RTK-GPS functionality with capability for centimeter level relative positioning accuracy developed by Swift Navigation, shown in figure 3.4. Piksi is ideal for autonomous vehicles because of its small form factor, fast position solution update rate and low power consumption. More detailed information about Piksi can be found in [Pik]

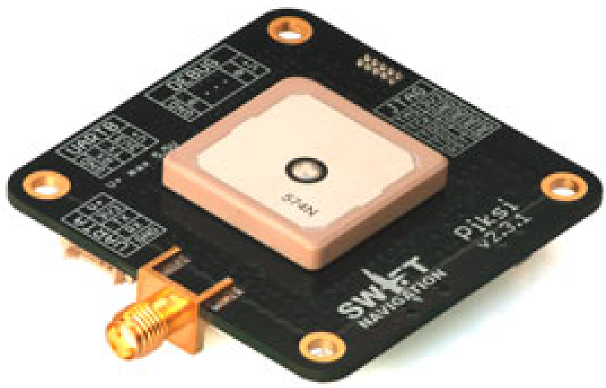


Figure 3.4: Piksi receiver, Picture from www.swiftnav.com

The communication structure for Piksi used in this project work is seen in figure 3.5.

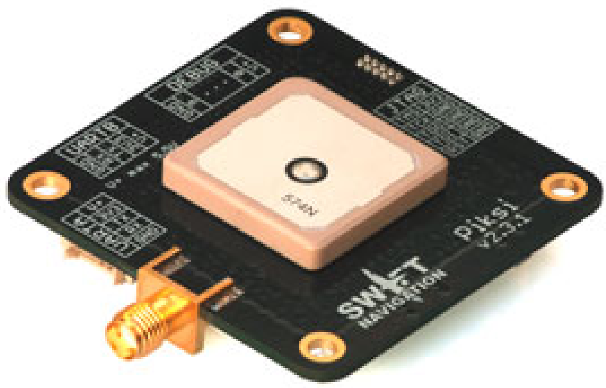


Figure 3.5: Communication structure for Piksi

3.1.4 GNSS Antenna

The navigation system will use two GNSS antenna, one for the UAV and the other for the base station. The main criteria for the UAV antenna is that it's small,compact and have a light weight.

The M1227HTC-A-SMA, seen in figure 3.6 has been used in other UAV setups at the UAVLAB with good results. The antenna is small, compact and with an light weight of $17g$. It's design for L1/L2 gps/glonass bands. Further information or specification can be found in the datasheet [max]

The base station do not have any restriction on size, weight or aerodynamic. The important factor for a base station antenna is that it has good multipath rejection. Also the base station position should be calculated as accurate as possible which impose further restriction on interference handling, phase center stability and noise rejection.

The Novatel GPS-701-GG, seen i figure 3.7, was chosen as the base station antenna. The antenna has excellent multipath rejection whit a highly stable phase center. It has reception for both GPS and GLONASS L1 signals.Further information or specification can be found in the datasheet [nov].

Figure 3.6: The rover GNSS antenna, Picture from http://sigma.octopart.com/21411362/image/Maxtena-M1227HCT-A-SMA.jpg

Figure 3.7: The base station GNSS antenna, Picture from http://www.novatel.com

3.2 Software

This section contain the different software that is used in the X8 system. The following sections contain the operating system that runs on the base station and rover, the middleware used to perform the different tasks, the messages protocol, the missionplaner program and rtklib which will have the main focus.

3.2.1 LSTS toolchain

The Laboratório de Sistemas e Tecnologia Subaquática (LSTS) toolchain is used a platform for software implementation. This is a flexible, scalable, open-source software that supports integration and control of various types of unmanned vehicles [Pinto et al., 2013]. The toolchain includes a operating system, runtime environment, message protocol and a command and control for operations. The following program is included in the toolchain

GLUED

Glued is a minimal Linux operating system distribution, and design with embedded system in mind. It is platform independent, easy to configure and contain only the necessary packages to run on a embedded system. This makes GLUED a light and fast distribution. GLUED is configured through a single configuration file that which can be created for a specific system.

Dune

Dune (DUNE Uniform Navigation Environment) is a runtime environment for unmanned systems. DUNE is operating system independent and can run on-board the embedded computer. It can interact with the sensors, actuators and payload. It can also be used for tasks like communication, navigation, control, manoeuvring, plan execution and supervision.

Dune works by setting up individual task that can dispatch and subscribe to different Inter-Module Communication (IMC) messages.

IMC

The Inter-Module Communication (IMC) protocol is build to interconnect systems of vehicles,sensors and human operators. The protocol is a messages-oriented protocol that enable exchange of real-time information about the environment and updated objectives, such that the participant in the communication can pursue a common goal cooperatively. IMC has a standard way of dispatching and consuming messages, which abstracts hardware configuration from the software. All IMC messages are logged by DUNE.

Neptus

Neptus is an open source command and control software that can be used for a single or fleet of unmanned vehicles with different types of sensors. The operator can observe real time data from a vehicle, review previous missions and plan future mission. In this work Neptus has been used to extract data logged by Dune during a defined mission, and to monitor the integer ambiguity solution from RTKLIB and Piksi.

3.2.2 RTKLIB

Real-Time Kinematic Library (RTKLIB) is a open source program package for standard and precise positioning with GNSS developed by T. Takasu. RTKLIB can use raw GNSS data to estimate the position of the rover. RTKLIB can be configured to give a position solution in real time in differential mode. Figure 3.8 shows how RTKLIB can be used in a RTKGNSS mode. The two main moduels here is str2str and rtkrcv. Both will be explained more closely in the following sections. More information about RTKLIB can be found in [Rtk].

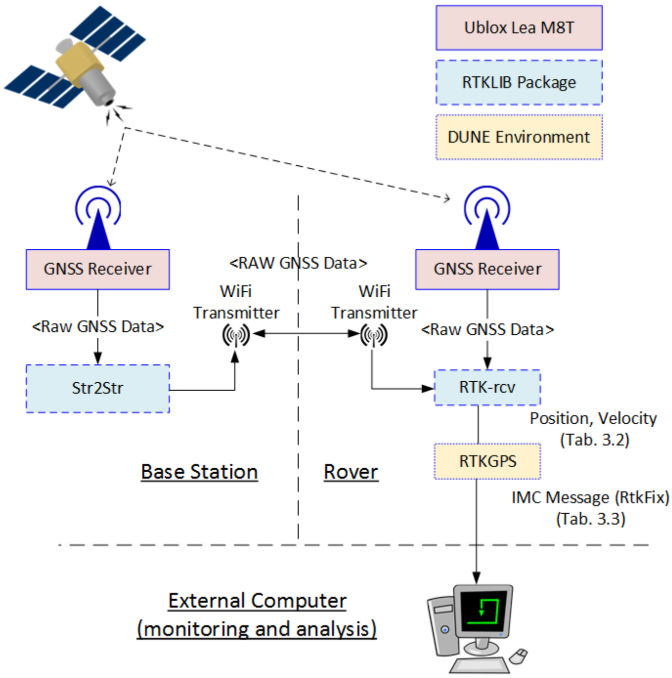


Figure 3.8: The communication structure of RTKLIB

Rtkrcv

As part of the RTKLIB Rtkrcv is used to calculate the position of the rover in real time. Rtkrcv can be configured to have two output streams. The structure of the output stream is given the RTKLIB manual[Rtk]. It's desired in a automatic landing system that have that velocity solution. This is was not provided in the newest version of RTKLIB, and therefore the source code was altered to send out the velocity data. The position output is in ENU format and the full output structure is shown in table 3.2

Header	**Content**
1 Time	The epoch time of the solution indicate the true receiver signal reception time. Can have the following format: yyyy/mm/dd HH:MM:SS.SSS: Calender time in GPST, UTC or JST. WWWW SSSSSSS.SSS: GPS week and TOW in seconds
2 Receiver Position	The rover receive antenna position
3 Quality flag (Q)	The flag which indicates the solution quality. 1:Fixed 2:Float 5:Single
4 Number of valid satellites (ns)	The number of valid satellites for solution estimation.
5 Standard deviation	The estimated standard deviation of the solution assuming a priori error model and error parameters by the positioning options
6 Age of differential	The time difference between the observation data epochs of the rover receiver and base station in second.
7 Ratio factor	The ratio factor of "ratio-test" for standard integer ambiguity validation strategy
8 Receiver velocity	The velocity of the rover. Given only when output is in enu format

Table 3.2: Rtklib output solution format

When set configured as a differential GPS rtkrcv uses the LAMBDA method to resolve the integer ambiguity., which is further explained in [Teunissen P.J.G and C.C.J.M., 1995]. The solution is considered fixed if the ration between the best estimate and the second best estimate is above a certain threshold.

The position solution is calculated with a Extended Kalman Filter, that can have different structure depending on how rtkrcv is configured

Str2str

Str2str is used as a base station program that can retrieve raw GNSS data and create a tcp sever, which the rtkrcv program can connect itself as a client. Str2str sends out RMTC3 formatted messages, however it can be configured to send whatever comes in as input.

3.3 Implementation

This section explain how the RTK-GPS navigation system has been implemented in the UAV system used in the testing. The implementation includes both a software and a hardware part, as shown i figure 3.9.

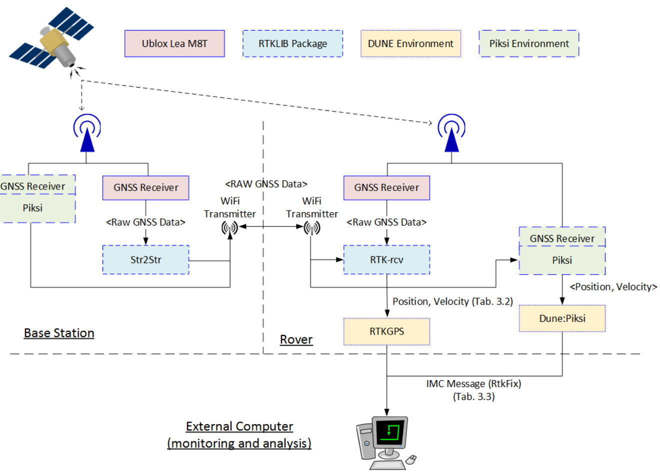


Figure 3.9: Hardware Software interaction

3.3.1 Software implementation

The software implemented is shown in figure 3.9, where the different modules are available from the UAVLAB. The detailed implementation has been trough configuration of the config files, including alteration of the existing implementation.

RTKGPS

The RTK-GPS module in the navigation system includes RTKLIB, and the two Dune tasks RTKGPS and Piksi. The RTKGPS task is connected to RTKLIB though a virtual connection, and the Piksi task has a physical connection to the Piksi receiver, as shown i figure 3.9.

RTKLIB is implemented into the base station and the rover. The base station implementation use the str2str program to communicate with the Ublox over a uart cable, outputs raw GNSS data over tcp to the rover. The rover uses the rtkrcv program from RTKLIB to estimate the position of the rover. Rtkrcv receive raw GNSS data from both the str2str program and the Ublox installed in the UAV, as shown in figure 3.8. Rtkrcv is configured in a moving baseline configuration to simulate the behaviour that is expected during a landing on a ship. The configuration file is included in A.

The output from the RTKGPS task is a IMC message called RtkFix, see table 3.3, which include the relative position of the UAV as well as the velocity, type of integer solution and the GPS Time Of Week (TOW). The IMC message is further sent to an external computer for monitoring over TCP/IP (Wifi).

Header	**Content**
tow	Gps time of Week
n	Baseline North coordinate
e	Baseline East coordinate
d	Baseline Down coordinate
v_n	Velocity North coordinate
v_e	Velocity East coordinate
v_d	Velocity Down coordinate
iar_hyp	Number of hypotheses in the Integer Ambiguity Resolution
iar_ratio	Quality ratio of Integer Ambiguity Resolution
type	Type of fix: None = No solution, but RTK task is running Obs = No solution, but receiving observations Float = Floating point solution of Integer Ambiguity Resolution Fix = Fixed(single) solution of Integer Ambiguity Resolution

Table 3.3: The IMC message RtkFix

3.3.2 Hardware implementation

Both the base station and UAV has been fitted with a GPS antenna splitter, seen i figure 3.10, such that both receivers receive the same signals from the antennas. GLUED is used as the operating system in the Beaglebone on both the base station and the UAV. The Piksi and Ublox is connected with the Beaglebone over uart cables. The primary data-link between the UAV and the base station with Ubiquiti AirMax radios. The embedded computer uses GLUED as its operating system, and on it runs both Dune and RTKLIB. The Piksi and Ublox is connected to the BeagleBone over uart cables. More information on the hardware setup used in the X8 and the Base station is given in [Zolich A. P. and K., 2015]

Figure 3.10: GPS antenna splitter

Chapter 4

Experimental testing

This chapter contain the result from the test that were performed. The goal with these tests was to test the performance of the Ublox LEA M8T, compare the performance of the Real-Time Kinematic Library (RTKLIB) with the Piksi alternative, and to compare the real time estimate from both system with the post-processed solution. The comparison test was performed with the pixi and ublox connected to the same antenna at both the rover and the base station. Hence the deviation in the position estimate can only come from the receivers. All position and velocity data is given in the NED frame, however altitude is only used for the flight test.

4.1 Performance testing of UAV Position System

The experimental test was split in two independent test, one where the X8 UAV was carried around on a open field to test the performance of the RTK-GPS system in a more controlled environment, and a second test where the UAV was flying by means of pilot control. The goal of the first was to log data from both RTK-GPS system in optimal condition for the navigation system. The objective of the second test was to test the systems in a more realistic environment for application in an automatic net landing system.

4.1.1 Test 1: Test of the RTK-GPS navigation system

The first test was repeated twice where results from both runs are reported. Both tests were performed on the same day, which was cloudless and at a time with good satellite constellation geometry. The raw data from the Ublox receiver was post processed with RTKLIB, which is assumed more accurate than real time processed data. Therefore a estimate of error is defined as:

$$e(t) = p_r(t) - p_p(t) \tag{4.1}$$

where $p_r(t)$ and $p_p(t)$ is defined as the position solution from the real time system and the position solution from the post processed solution respectfully. $e(t)$ is used as a measure on the performance of the position estimate relative to the assumed more accurate post-processed estimate. It should be noted that in order to compare the different time-series the position data was synchronized with each other. From the error the cumulative standard deviation was calculated using the matlab function "std".

First test In the first session of the test the UAV was carried around on a open field, and later placed exactly on the same place where it started. As expected both systems provided a position estimate with fixed integer solution that followed the true path, and further confirms that both system performed in a similar manner. Therefore both systems are suitable for further comparison of position estimation in a flying test. Figure 4.1 shows a North East plot of how the walk was. The plot contain only the fixed solution from both the piksi and rtklib.

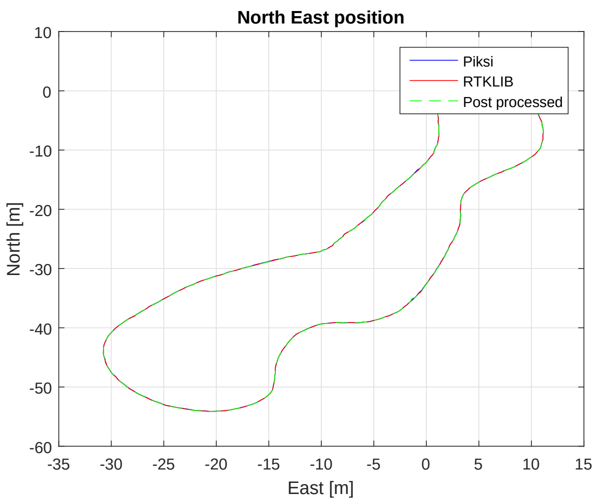


Figure 4.1: The North/East position during the first test

Figure 4.2 shows the down position, as well as how the integer ambiguity solution was during the experiment. As seen in the figure both the Piksi and RTKLIB manage to keep there fixed solution. The position solution from both the Piksi and RTKLIB agrees with the post processed solution.

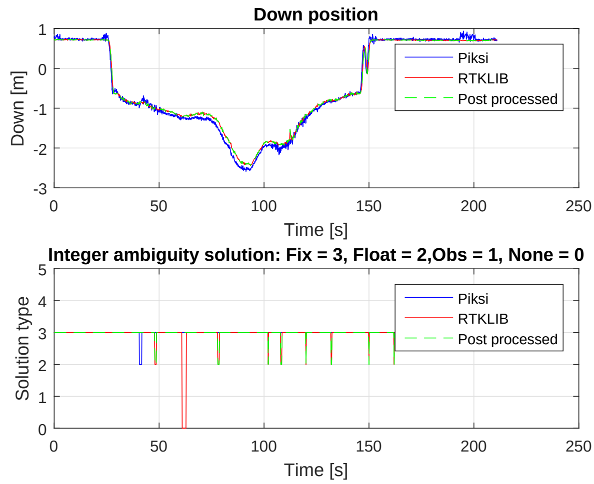


Figure 4.2: The Down position and integer ambiguity solution during the first test

As seen in figure 4.1 the difference between the different solution is quite small, which is confirmed in the error plot shown in figure 4.3 and 4.4

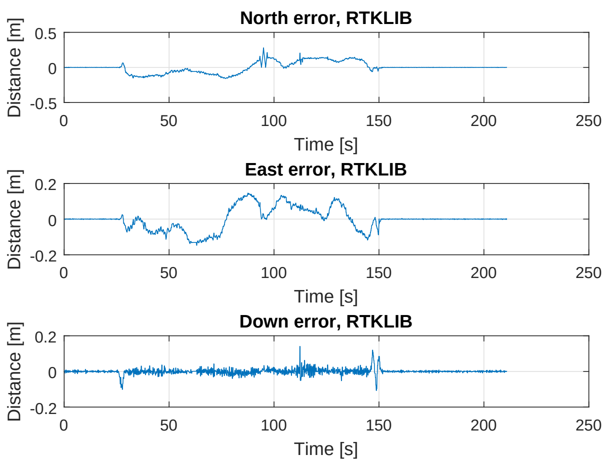


Figure 4.3: The error from RTKLIB

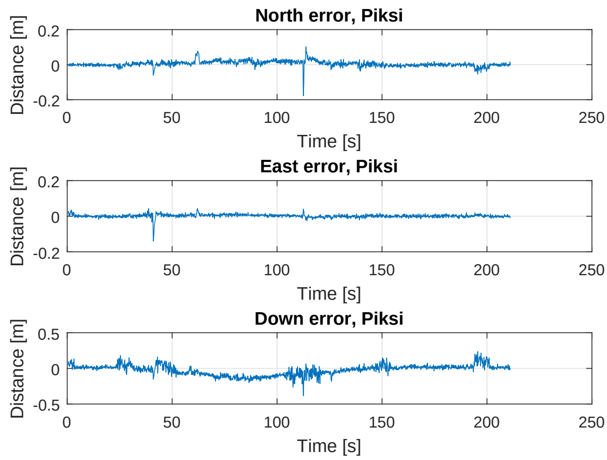


Figure 4.4: The error from Piksi

The cumulative standard deviation for the error from both the Piksi and RTKLIB indicate that the estimate has a high precision, as seen in figure 4.6 and 4.5. The high precision in the navigation system is critical in a automatic net landing system for the system to be able to follow a landing path.

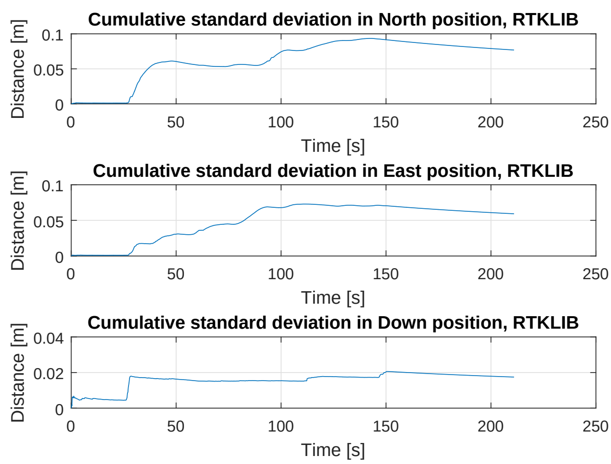


Figure 4.5: The cumulative standard deviation from RTKLIB

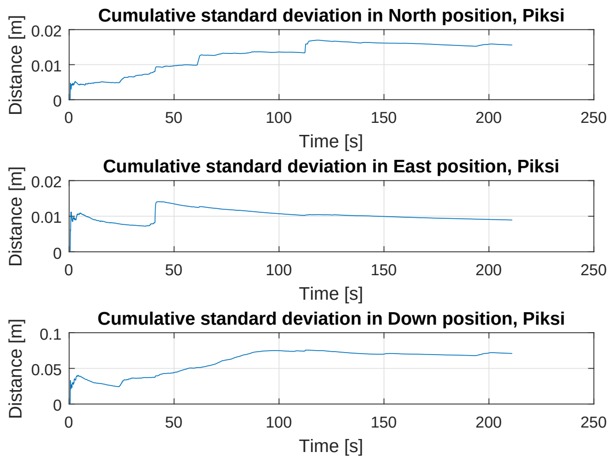


Figure 4.6: The cumulative standard deviation from Piksi

Both the Piksi and RTKLIB agrees to the same estimate the North and East velocity, as seen in figure 4.7. However the Down velocity from RTKLIB is more noisy then the Piksi. This behaviour could be introduced because the UAV was carried around.

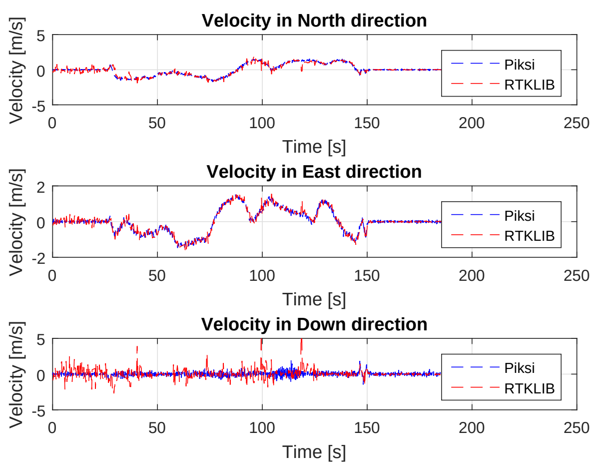


Figure 4.7: The velocity from the first test

The different receiver used in RTKLIB and Piksi was not able to track the same satellites at all time. Figure 4.8 shows that the Ublox LEA M8T receiver connected to RTKLIB managed to track more satellite then the receiver used in Piksi.

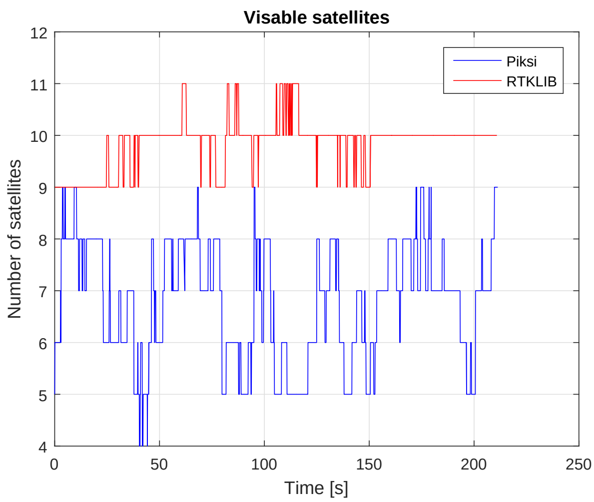


Figure 4.8: Number of visible satellites in the first test

The position estimate of the rover is a relative position in reference to the base station. Due to the fact that the position of the base station is calculated with a single receiver with one frequency, there will be introduced a bias in the position estimate. Figure 4.9 shows the North, East position of the the first walk. The true position is exactly the same, but in the figure the distance is approximately $5-10cm$. The distance from the base station to the estimated start and stop position was calculated to be $3.3553m$ and $3.2914m$ respectfully. The measured distance was approximately $3.3m$. That gives an initial error off $0.02m$. This gives an accuracy level at centimeter level at stationary condition.

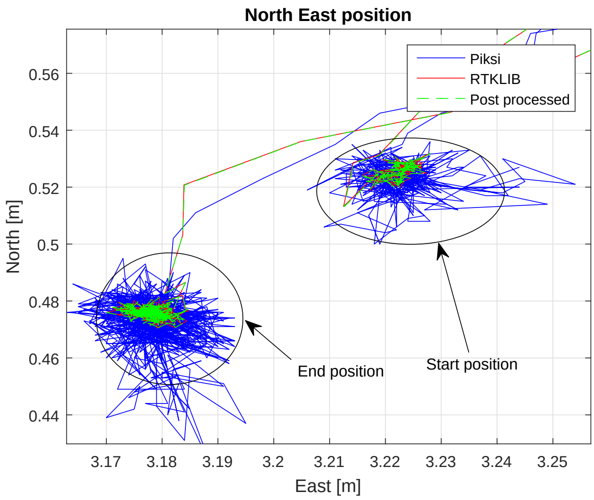


Figure 4.9: Enhanced North/East plot from the first test

The position estimate from RTKLIB is delayed in comparison to the solution from the Piksi. Figure 4.10 shows that both the post processed solution and the real time solution is delay by 0.5 secounds compared to the Piksi. This could be how RTKLIB resolve the millisecond in Time Of Week (TOW), and may not be seen as an extra delay seen from the control systems perspective.

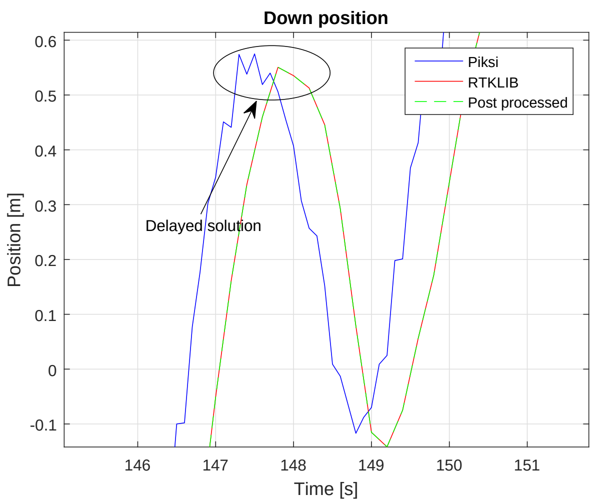
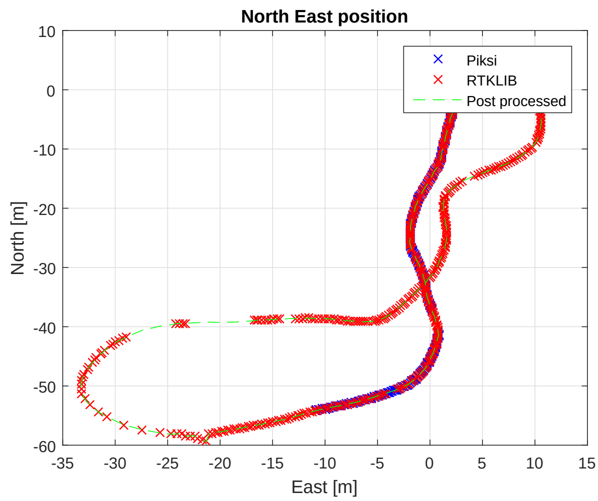


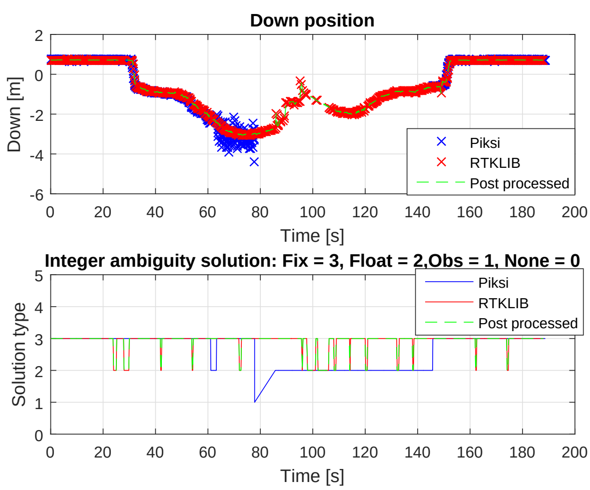
Figure 4.10: Enhanced Down plot from the first test

Given that the error given in figure 4.4 and 4.3 is never has a greater absolute value then $0.2m$ it's possible to assume that the true error will be bellow $1m$ which was given as an evasion criterion in the MSc thesis by [Marcus, 2015], if the RTK-GPS system has a fixed integer solution.

Second test The second session was performed few minutes after the first, with the same weather condition. During the second session the Piksi lost its fixed integer solution, while RTKLIB managed to keep its fixed integer solution as seen in figure 4.11a and 4.11b.



(a) The North/East position during the second test



(b) The Down position and integer ambiguity solution during the second test

The reason for why the Piksi lost its fixed solution might be because it lost track of several satellite, as seen in figure 4.12. Since both receive share the same antenna it can be concluded that the satellite tracing performance in the Ublox is superior to the Piksi. Even when the Piksi managed to regain track of the satellite it lost, it took 60 seconds before it regain a fixed integer solution.

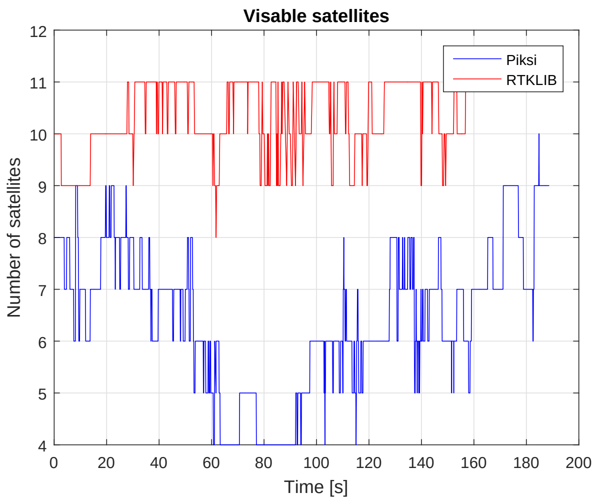


Figure 4.12: Number of visible satellites in the first test

4.1.2 In-flight test

A flight test with the UAV was performed at Udduvoll. Because of bad weather only one flight test was performed. Prior to take-of it was only the RTKLIB which was able to provide a fixed solution. Hence only performance from the RTKLIB is considered in this test, as a navigation system must have a fixed solution before the automatic net landing system can start.

During the flight test the integer ambiguity solution were more float then fixed as seen in figure 4.14 and 4.13, which affected the measurement. The same behaviour will be seen during the landing, however if the float solution is accurate enough the system should be able to perform an automatic landing.

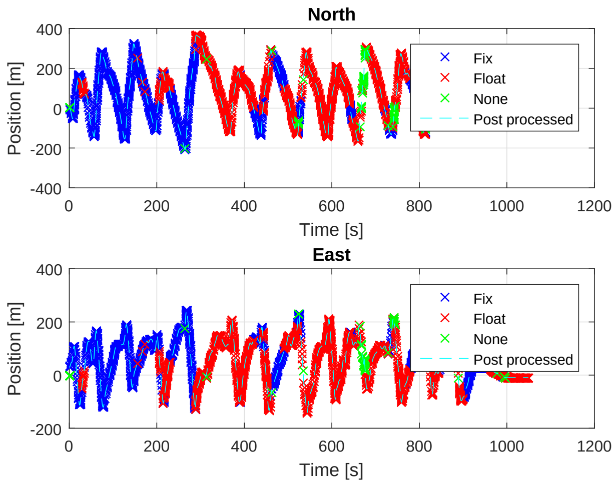


Figure 4.13: The North and East position during the flight

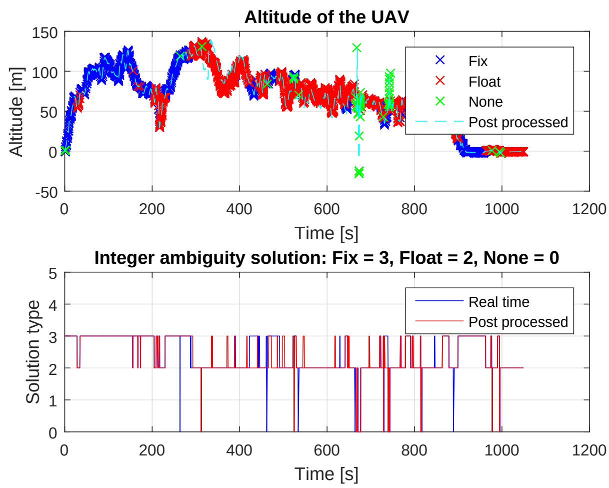


Figure 4.14: The altitude during the flight

The main reason for the lost fixed solution is because of the number of valid satellite the receiver can track experience large variation, as seen i figure 4.15. A problem that was experienced during the flight is that large roll and pitch angel of the UAV puts the antennas in shadow zones, i.e. lose communication with different satellites. That is a problem that can be solved in a control system by setting restriction on the dynamical behaviour of the UAV especially before and during landing.

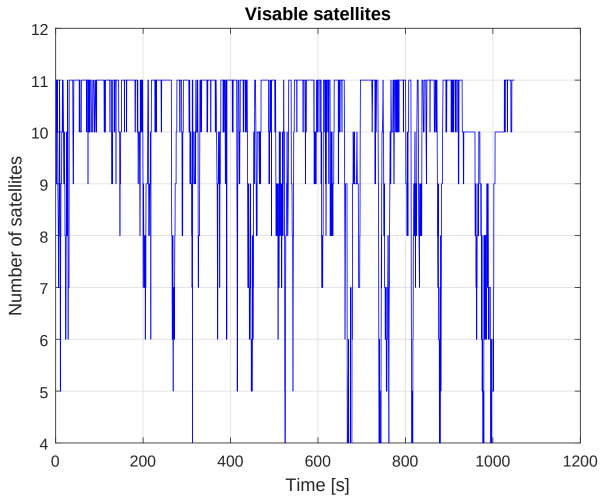


Figure 4.15: Number of visible satellites during the flight

Landing

As expected the RTK-GPS system had problem with keeping it's fixed integer solution. The RTK-GPS position estimate of the landing path is shown in figure 4.16. The system kept changing between float and fixed solution during the landing phase. The Down position as well as the integer ambiguity solution for the landing phase is seen in 4.17. The RTK-GPS system was unable to maintain a fixed solution, however it did maintain a float solution. Further test with a control system is required in order to test if also the float solution is accurate enough to perform a automatic net landing.

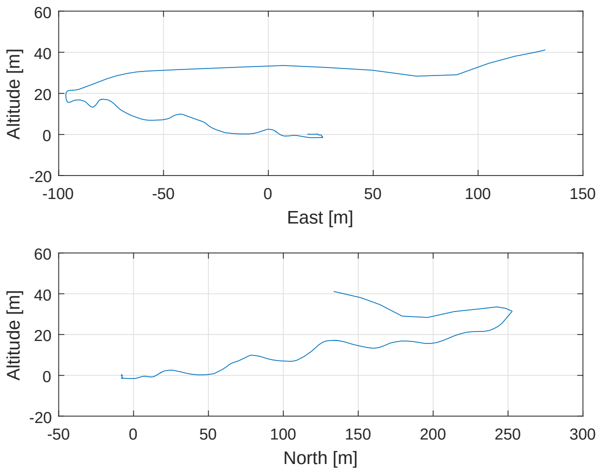


Figure 4.16: The landing path

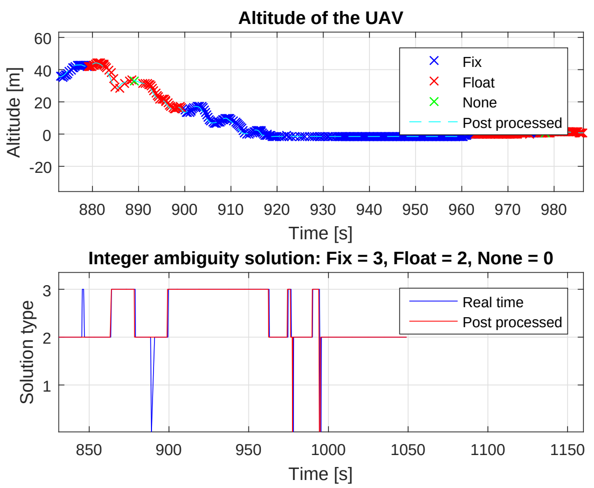


Figure 4.17: The altitude during the landing

From the error plot seen in figure 4.18 it is able to estimate it's own position with an error bellow 1 meter most of the time. However this is compared to the post processed estimate, which also will diverge from the true value since it is based on the same raw data as the real time solution.

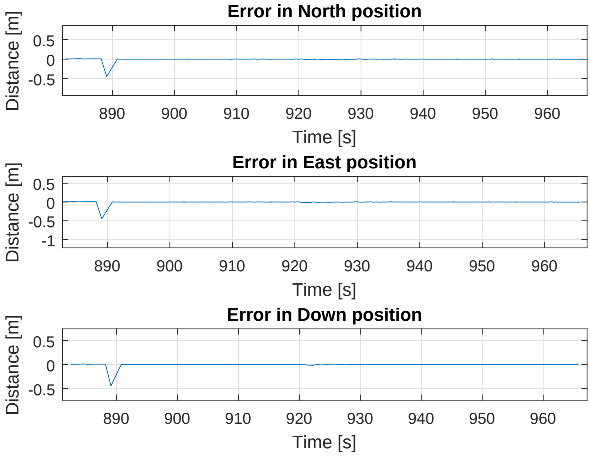


Figure 4.18: The error during the landing

The landing velocity seen in figure 4.19 confirms the precision in the estimate that was seen in the first test. However the altitude velocity is less noisy, and can be used in a control system.

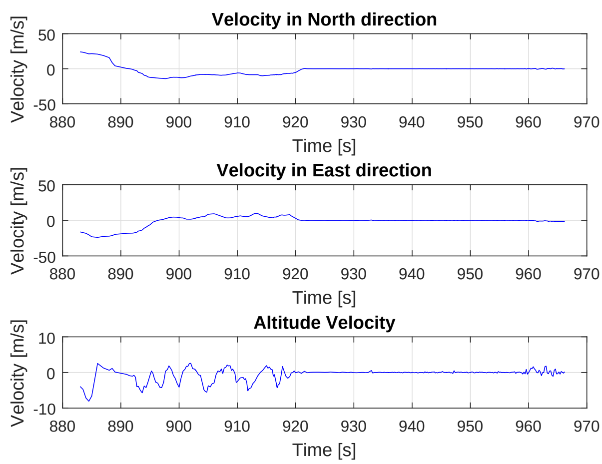


Figure 4.19: The velocity during the landing

Chapter 5

Conclusion and recommendation for further work

This chapter will discus the result from the field tests, which will be used to conclude the performance of the individual RTK-GPS system. The last part of this chapter will include suggestion for further work regarding the design of a automatic net landing system.

5.1 RTK-GPS lab testing

It was seen in the lab testing that both the Piksi and RTKLIB was able to get a fixed solution, although the RTKLIB was faster. In the first test were the UAV was carried both system gave similar result during the first session UAV, however during the second the Piksi lost its fixed solution. RTKLIB had also problem during this session, but unlike the Piksi managed to find a fixed solution again. The ability for the navigation system to quickly recover its integer solution is critical for a automatic net landing system.

The RTK-GPS system achieves centimeter level accuracy when stationary. When the UAV is in motion this will decrease down to decimeter accuracy. This can be assumed is the behaviour for the navigation system based on the stationary accuracy test, and the high precision in the position estimate. That should be good enough in a navigation system for a automatic landing system.

The output solution from RTKLIB has a Time Of Week (TOW) value that is constant half a second delayed compared to the Piksi. It could be that RTKLIB interpolate the microseconds of the TOW value from the different satellites, such that a control system do not get a delayed position estimate. However for a integrated navigation system this pose a problem. The integration becomes more difficult when it's unclear how old the position estimate is.

The tracking performance from the two receiver type indicate that the Ublox is superior to the Piksi. It always manage to track more satellites, and managed to track them longer.

5.2 RTK-GPS in-flight test

The flight test showed that keeping the fixed integer solution during a flight is difficult, most likely because of the dynamic behaviour of the UAV. The landing phase of the UAV operation should be kept independent from the rest of the operation. Therefore landing specific constraint cannot be imposed on the UAV behaviour. In order to then get a best relative position estimate the RTK-GPS must be able to recover it's fixed integer solution as quickly as possible while the UAV is in flight.

Before the take-off the Piksi had yet solved its integer ambiguity, while RTKLIB had. That might be because of it kept track of fever satellites. However the navigation system must be able to resolve its integer ambiguity quickly to increase the probability of performing a safe and successfully automatic landing.

During the flight the Ublox lost track of more satellite then during the GPS lab tests. The antenna follows the orientation of the UAV and therefore some satellite will be blocked by the fuselage. This must be consider when designing a guidance system for automatic landing. To ensure that the antenna has a clear view of the sky, the UAV should try to have a maximum roll when the automatic landing system is engaged.

The velocity estimate from Piksi and RTKLIB in the first test have similar estimate for the North and East component, but differ in the Down component. In the flight test the RTKLIB altitude velocity was better, such that it can be used in a control system.

5.3 Further work

The Ublox receiver and the GNSS antennas are able to receive both GPS and GLONASS L1 signals, which should be exploited to increase the number of valid satellite available for the RTK-GPS system. This could also give better constellation geometry, and help the system resolve its integer ambiguity faster.

The RTK-GPS system must be integrated with the existing control and guidance system, and a test landing in a stationary net must be performed.

A RTK-GPS system can use both the Piksi and RTKLIB such that the automatic landing system has redundancy in position, and velocity estimation. A validation task must then be design to choose which of the system should send the rtkfix IMC message to the rest of the system.

A RTK-GPS/glsins integration navigation system can be implemented in the automatic landing system. A RTK-GPS/glsins integration was proposed in [Amstrup, 2015], which can be used.

A position estimation system that can accurately predict were the UAV will be a few seconds ahead of time, such that the UAV can better know were it is instead of were if was.

Setting constraints on the path such that the antenna has a clear view of the satellites at all time during the landing phase.

References

Piksi Datasheet ver. 2.3.1.

RTKLIB ver. 2.4.2 Manual.

NEO/LEA-M8T u-blox M8 concurrent GNSS timing modules, Data Sheet, a.

U-blox M8 Receiver Description Including Protocol Specification, b.

M1227HCT-A-SMA L1/L2 GPS-GLONASS Active Antenna, Data Sheet.

Novatel, GPS-701-GG and GPS-702-GG, Data Sheet.

Smit Samuel Jacobus Adriaan. Autonomous landing of a fixed-wing unmanned aerial vehicle using differential gps. Master's thesis, Stellenbosch: Stellenbosch University, 2013.

Spockeli Bjørn Amstrup. Integration of rtk gps and imu for accurate uav positioning. Master's thesis, Norwegian University of Science and Technology, NTNU, 2015.

Vik Bjørnar. *Integrated Satellite and Inertial Navigation Systems.* Department of Engineering Cybernetics, NTNU, 2014.

Blewitt Geoffrey. Carrier phase ambiguity resolution for the global positioning system applied to geodetic baselines uo to 2000 km. *Journal of Geophysical Research*, 94(B8):10,187–10,203, 1989.

Fossen Thor I. *Handbook of marine craft hydrodynamics and motion control.* John Wiley & Sons, 2011.

Kim H Jin, Kim Mingu, Lim Hyon, Park Chulwoo, Yoon Seungho, Lee Daewon, Choi Hyunjin, Oh Gyeongtaek, Park Jongho, and Kim Youdan. Fully autonomous vision-based net-recovery landing system for a fixed-wing uav. *Mechatronics, IEEE/ASME Transactions on*, 18(4):1320–1333, 2013.

Frølich Marcus. Automatic ship landing system for fixed-wing uav. Master's thesis, Norwegian University of Science and Technology, NTNU, 2015.

Williams Paul and Crump Michael. Intelligent landing system for landing uavs at unsurveyed airfields. In *28th International Congress of the Aeronautical Sciences*, 2012.

Joel Pinto, Paulo S Dias, Rui P Martins, Joao Fortuna, Eduardo Marques, and Joao Sousa. The lsts toolchain for networked vehicle systems. In *OCEANS-Bergen, 2013 MTS/IEEE*, pages 1–9. IEEE, 2013.

Teunissen P.J.G. A new method for fast carrier phase ambiguity estimation. *IEEE*, pages 562–573, 1994.

Teunissen P.J.G. The least-squares ambiguity decorrelation adjustment: a method for fast gps integer ambiguity estimation. *Journal of Geodesy*, 70:65–82, 1995.

Misra Pratap and Enge Per. *Global Positioning System Signals, Measurements and Performance Revised Second Edition.* Ganga-Jamuna Press, 2011.

Skulstad Robert and Syversen Christoffer Lie. Low-cost instrumentation system for recovery of fixed-wing uav in a net. Master's thesis, Norwegian University of Science and Technology, NTNU, 2014.

de Jonge P.J. Teunissen P.J.G and Tiberius C.C.J.M. The lambda-method for fast gps surveying. Presented at the International Symposium "GPS Technology Applications" Bucharest,Romania, 1995.

Stempfhuber W. 3d-rtk capability of single gnss receivers. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2013.

Stempfhuber W. and Buchholz M. A precise, low-cost rtk gnss system for uav applications. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2011.

Cisek K. P. Zolich A. P., Johansen T. A. and Klausen K. Unmanned aerial system architecture for maritime missions. design and hardware description. In *Education and Development of Unmanned Aerial Systems (RED-UAS)*, 2015.

Appendix A

Rtklib Configuration

A.1 str2str

Need start commando and rate configuration of the ublox

A.2 rtkrcv

Configuration given bellow. Summarize the important parts of the configuration.

```
# RTKNAVI options (2015/10/30 13:05:07, v.2.4.2)

pos1-posmode       =movingbase # (0:single,1:dgps,2:kinematic,3:static,
4:movingbase,5:fixed,6:ppp-kine,7:ppp-static)
pos1-frequency     =l1         # (1:l1,2:l1+l2,3:l1+l2+l5,
4:l1+l2+l5+l6,5:l1+l2+l5+l6+l7)
pos1-soltype       =forward    # (0:forward,1:backward,2:combined)
pos1-elmask        =10         # (deg)
pos1-snrmask_r     =off        # (0:off,1:on)
pos1-snrmask_b     =off        # (0:off,1:on)
pos1-snrmask_L1    =0,0,0,0,0,0,0,0,0
pos1-snrmask_L2    =0,0,0,0,0,0,0,0,0
pos1-snrmask_L5    =0,0,0,0,0,0,0,0,0
pos1-dynamics      =on        # (0:off,1:on)
pos1-tidecorr      =off        # (0:off,1:on,2:otl)
pos1-ionoopt       =brdc       # (0:off,1:brdc,2:sbas,3:dual-freq,
4:est-stec,5:ionex-tec,6:qzs-brdc,7:qzs-lex,8:vtec_sf,9:vtec_ef,10:gtec)
pos1-tropopt       =saas       # (0:off,1:saas,2:sbas,3:est-ztd,4:est-ztdgrad)
pos1-sateph        =brdc       # (0:brdc,1:precise,2:brdc+sbas,
```

```
3:brdc+ssrapc,4:brdc+ssrcom)
pos1-posopt1       =off         # (0:off,1:on)
pos1-posopt2       =off         # (0:off,1:on)
pos1-posopt3       =off         # (0:off,1:on)
pos1-posopt4       =off         # (0:off,1:on)
pos1-posopt5       =off         # (0:off,1:on)
pos1-exclsats      =            # (prn ...)
pos1-navsys        =1           # (1:gps+2:sbas+4:glo+8:gal+16:qzs+32:comp)
pos2-armode        =fix-and-hold # (0:off,1:continuous,
2:instantaneous,3:fix-and-hold)
pos2-gloarmode     =off         # (0:off,1:on,2:autocal)
pos2-bdsarmode     =off         # (0:off,1:on)
pos2-arthres       =3
pos2-arlockcnt     =0
pos2-arelmask      =0           # (deg)
pos2-arminfix      =10
pos2-elmaskhold    =0           # (deg)
pos2-aroutcnt      =5
pos2-maxage        =30          # (s)
pos2-syncsol       =on         # (0:off,1:on)
pos2-slipthres     =0.05        # (m)
pos2-rejionno      =30          # (m)
pos2-rejgdop       =30
pos2-niter         =1
pos2-baselen       =0           # (m)
pos2-basesig       =0           # (m)
out-solformat      =llh         # (0:llh,1:xyz,2:enu,3:nmea)
out-outhead        =off         # (0:off,1:on)
out-outopt         =off         # (0:off,1:on)
out-timesys        =gpst        # (0:gpst,1:utc,2:jst)
out-timeform       =tow         # (0:tow,1:hms)
out-timendec       =3
out-degform        =deg         # (0:deg,1:dms)
out-fieldsep       =
out-height         =ellipsoidal # (0:ellipsoidal,1:geodetic)
out-geoid          =internal    # (0:internal,1:egm96,
2:egm08_2.5,3:egm08_1,4:gsi2000)
out-solstatic      =all         # (0:all,1:single)
out-nmeaintv1      =0           # (s)
out-nmeaintv2      =0           # (s)
out-outstat        =state         # (0:off,1:state,2:residual)
```

```
stats-eratio1      =100
stats-eratio2      =100
stats-errphase     =0.003       # (m)
stats-errphaseel   =0.003       # (m)
stats-errphasebl   =0           # (m/10km)
stats-errdoppler   =1           # (Hz)
stats-stdbias      =30          # (m)
stats-stdiono      =0.03        # (m)
stats-stdtrop      =0.3         # (m)
stats-prnaccelh    =10          # (m/s^2)
stats-prnaccelv    =10          # (m/s^2)
stats-prnbias      =0.0001      # (m)
stats-prniono      =0.001       # (m)
stats-prntrop      =0.0001      # (m)
stats-clkstab      =5e-12       # (s/s)
ant1-postype       =llh         # (0:llh,1:xyz,2:single,
3:posfile,4:rinexhead,5:rtcm)
ant1-pos1          =90          # (deg|m)
ant1-pos2          =0           # (deg|m)
ant1-pos3          =-6335367.6285 # (m|m)
ant1-anttype       =
ant1-antdele       =0           # (m)
ant1-antdeln       =0           # (m)
ant1-antdelu       =0           # (m)
ant2-postype       =llh         # (0:llh,1:xyz,2:single,
3:posfile,4:rinexhead,5:rtcm)
ant2-pos1          =90          # (deg|m)
ant2-pos2          =0           # (deg|m)
ant2-pos3          =-6335367.6285 # (m|m)
ant2-anttype       =
ant2-antdele       =0           # (m)
ant2-antdeln       =0           # (m)
ant2-antdelu       =0           # (m)
misc-timeinterp    =off         # (0:off,1:on)
misc-sbasatsel     =0           # (0:all)
misc-rnxopt1       =
misc-rnxopt2       =
file-satantfile    =
file-rcvantfile    =
file-staposfile    =
file-geoidfile     =
```

```
file-ionofile      =
file-dcbfile       =
file-eopfile       =
file-blqfile       =
file-tempdir       =/tmp/
file-geexefile     =
file-solstatfile   =
file-tracefile     =
#

inpstr1-type       =serial     # (0:off,1:serial,2:file,
3:tcpsvr,4:tcpcli,7:ntripcli,8:ftp,9:http)
inpstr2-type       =tcpcli     # (0:off,1:serial,2:file,
3:tcpsvr,4:tcpcli,7:ntripcli,8:ftp,9:http)
inpstr3-type       =off        # (0:off,1:serial,2:file,
3:tcpsvr,4:tcpcli,7:ntripcli,8:ftp,9:http)
inpstr1-path       =uart/2:115200:8:n:1:off
inpstr2-path       =:@10.0.60.51:50022/:
inpstr3-path       =
inpstr1-format     =ubx        # (0:rtcm2,1:rtcm3,2:oem4,
3:oem3,4:ubx,5:ss2,6:hemis,7:skytraq,8:gw10,9:javad,10:nvs,11:binex,12:rt17,15:sp3)
inpstr2-format     =ubx        # (0:rtcm2,1:rtcm3,2:oem4,
3:oem3,4:ubx,5:ss2,6:hemis,7:skytraq,8:gw10,9:javad,10:nvs,11:binex,12:rt17,15:sp3)
inpstr3-format     =rtcm2      # (0:rtcm2,1:rtcm3,2:oem4,
3:oem3,4:ubx,5:ss2,6:hemis,7:skytraq,8:gw10,9:javad,10:nvs,11:binex,12:rt17,15:sp3)
inpstr2-nmeareq    =off        # (0:off,1:latlon,2:single)
inpstr2-nmealat    =0          # (deg)
inpstr2-nmealon    =0          # (deg)
outstr1-type       =serial        # (0:off,1:serial,2:file,
3:tcpsvr,4:tcpcli,6:ntripsvr)
outstr2-type       =file        # (0:off,1:serial,2:file,
3:tcpsvr,4:tcpcli,6:ntripsvr)
outstr1-path       =../tmp/ttyV0:115200:8:n:1:off
outstr2-path       =/opt/lsts/rtklib/log/rtklib_output%Y%m%d%h%M.pos
outstr1-format     =enu        # (0:llh,1:xyz,2:enu,3:nmea)
outstr2-format     =enu        # (0:llh,1:xyz,2:enu,3:nmea)
logstr1-type       =file        # (0:off,1:serial,2:file,
3:tcpsvr,4:tcpcli,6:ntripsvr)
logstr2-type       =file        # (0:off,1:serial,2:file,
3:tcpsvr,4:tcpcli,6:ntripsvr)
logstr3-type       =off        # (0:off,1:serial,2:file,
```

```
3:tcpsvr,4:tcpcli,6:ntripsvr)
logstr1-path       =/opt/lsts/rtklib/log/rtklib_ubxstream_x8_log%Y%m%d%h%M.ubx
logstr2-path       =/opt/lsts/rtklib/log/rtklib_ubxstream_base_log%Y%m%d%h%M.ubx
logstr3-path       =
misc-svrcycle      =50          # (ms)
misc-timeout       =30000       # (ms)
misc-reconnect     =30000       # (ms)
misc-nmeacycle     =5000        # (ms)
misc-buffsize      =32768       # (bytes)
misc-navmsgsel     =all         # (0:all,1:rover,2:base,3:corr)
misc-proxyaddr     =
misc-fswapmargin   =30          # (s)
#misc-startcmd =./rtkstart.sh
#misc.startcmd =./rtkshut.sh
file-cmdfile1 =/etc/rtklib/data/ubx_raw_10hz.cmd
file-cmdfile2 =/etc/rtklib/data/ubx_raw_10hz.cmd
```