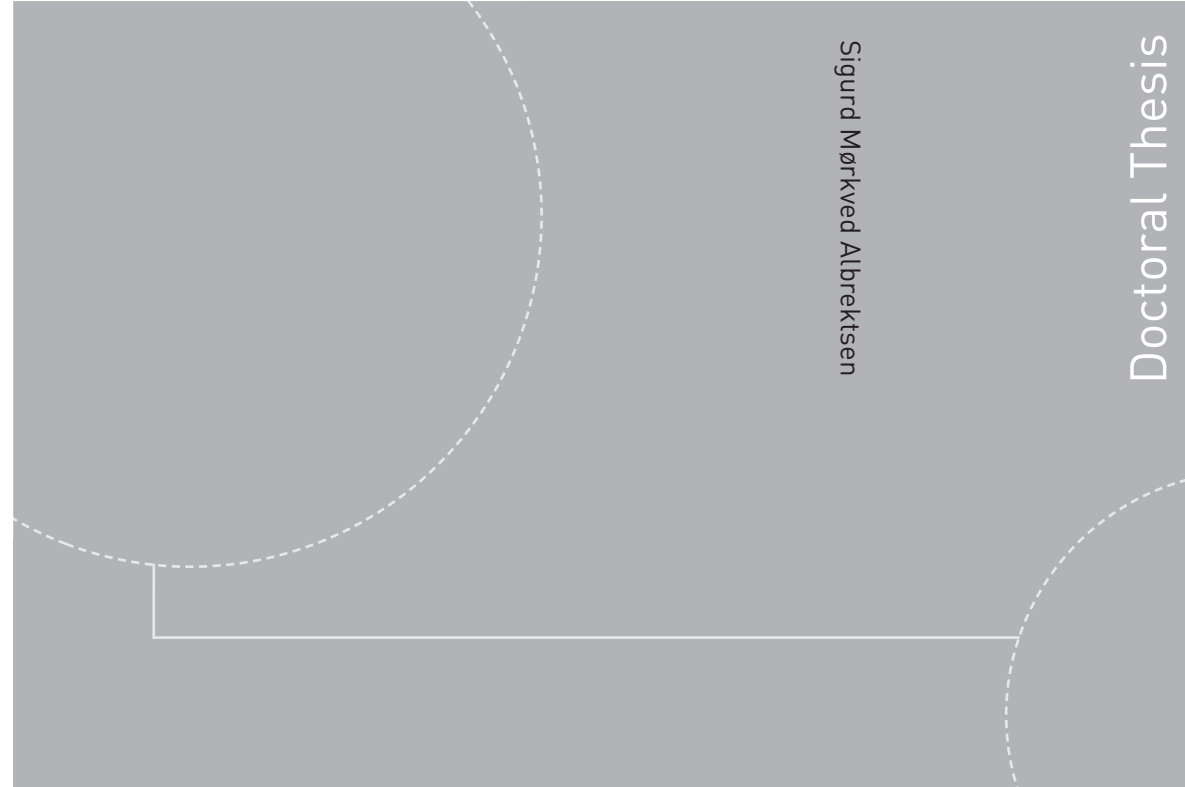


ISBN 978-82-326-3340-1 (printed version)
ISBN 978-82-326-3341-8 (electronic version)
ISSN 1503-8181



Sigurd Mørkved Albrechtsen
**Sensor Synchronization and
Navigation in GNSS-Denied
Environments for
Unmanned Aerial Vehicles**

Sigurd Mørkved Albrektsen

Sensor Synchronization and Navigation in GNSS-Denied Environments for Unmanned Aerial Vehicles

Thesis for the degree of Philosophiae Doctor

Trondheim, September 2018

Norwegian University of Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics



Norwegian University of
Science and Technology

NTNU

Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

© Sigurd Mørkved Albrektsen

ISBN 978-82-326-3340-1 (printed version)

ISBN 978-82-326-3341-8 (electronic version)

ISSN 1503-8181

ITK-report: 2018-12-W

Doctoral theses at NTNU, 2018:275



Printed by Skipnes Kommunikasjon as

For my loving family

Summary

In the last few years, the emergence of civilian unmanned aerial vehicles (UAVs) has sparked the interest of both the industry and the academic community. However, as systems become more popular and more widespread, the risk they represent increases as well. Today's solutions rely heavily on global navigation satellite systems (GNSS), which have been shown to have several weaknesses against malicious attacks – the signal can either be blocked through jamming, or a fake position signal can be sent, tricking the UAV into thinking it is at another location through spoofing.

When developing a navigation and sensor solution for UAVs, there are two main considerations that must be made. The solution must provide accurate sensor measurements, and the UAV must be able to operate safely and securely in the mission's environment. This thesis first presents a re-configurable sensor timing and synchronization solution, intended for use in UAV payloads, with two implementations. This solution reads the time-of-validity (TOV) from different sensors and attaches these as timestamps to the sensor's data messages. The solution can also be integrated with sensors such as cameras by triggering when they should capture images and storing the TOV through the camera's flash output. The solution's operation is verified through tens of experiments with beyond visual line of sight (BVLOS) fixed-wing UAVs, and an implementation has been shown to give $1.90\text{ }\mu\text{s}$ drift per second, with a resolution of 10 ns.

To make a UAV navigation solution that is robust to the single point of failure (SPOF) that GNSS represents, and to allow operation in GNSS denied environments, solutions for GNSS-free navigation is discussed in the second part of this thesis. First a camera based optical flow solution is presented, that enables throw-and-go functionality, which is verified through an implementation on a small multi-rotor UAV.

To enable navigation without GNSS for long-range, long-duration flights, a phased array radio system (PARS) is used as an absolute positioning sensor. First a solution with only the PARS and a barometer is presented, then these sensors are used to aid a nonlinear observer with an IMU and magnetometer. A root mean squared accuracy of 27 m was achieved with the first solution, and a root mean square accuracy of 13 m was achieved with the second solution.

Finally, a GNSS-spoofing detector was implemented that automatically detects spoofing and selects the best available solution of a highly accurate real-time kinematic (RTK) GNSS solution, which is sensitive to jamming, and a PARS NLO solution, which is highly robust, but less accurate.

Preface

This thesis presents the research results of my doctoral studies, performed at the Norwegian University of Science and Technology, at the Department of Engineering Cybernetics. It was financed by the Norwegian Research Council (NRF), through the FRIPRO project Low-Cost Integrated Navigation Systems using Nonlinear Observer Theory (LowCostNav), grant no. 221666, and the NRF project Maritime activities and offshore operations (MAROFF), grant no. 269480. The project was associated with the Center for Autonomous Marine Operations and Systems (NTNU AMOS), grant no. 223254. The work was completed under supervision of Professor Tor Arne Johansen (ITK, NTNU), Dr. Torleiv H. Bryne (ITK, NTNU) and Professor Thor I. Fossen (ITK, NTNU).

Acknowledgements

First and foremost, I would like to thank my supervisor Tor Arne Johansen for his guidance and support through my PhD. His insight and the blazingly fast response to emails, even when sent late at night on holidays, and his trust in me and my work have been of tremendous value.

Furthermore, I would like to thank my co-supervisor Torleiv H. Bryne for the great cooperation with the phased array radio navigation papers. It has been a pleasure working with him for the last few years. I would also like to thank my second co-supervisor Thor I. Fossen for admitting me as a part of the LowCostNav project.

I am also grateful to our pilots Pål Kvaløy and Lars Semb for their assistance with all UAV-related operations, from construction of the UAVs at the lab, to mission planning and maneuvering the UAVs in the field. I also thank Torbjørn Houge and Carl Erik Stephansen at Maritime Robotics,

for their help with constructing and flying the Penguin B, and Atle Sægrov and Tor Berg at Radionor Communications for their help with the Phased Array Radio, and quick support when it was needed the most.

Furthermore, I'm thankful for the support from Stephan Weiss and Roland Brockers who were my supervisors for my stay at NASA's Jet Propulsion Laboratory in Pasadena, CA. Their assistance with the throw-and-go UAVs led to a very exciting and educational experience.

Also, thanks to the members of the UAV lab and especially the navigation team for their work on UAV development and for their helpful suggestions throughout these years; especially Robert H. Rogne, Kristian Klausen, Kasper T. Borup, Jakob M. Hansen, and Lorenzo Fusini. Furthermore, I thank the administrative staff at the Department of Engineering Cybernetics for their help.

To my family and friends, thank you for your encouragement, advice and positive distractions through these years. The late-night gaming sessions, the skiing trips, the hacking competitions, the dinners, the concerts, and the barn festivals have all really helped me unwind after stressful days at work.

Finally, I want to thank my wife Tonje and my mother Torill for the immense support and care they show me. Thank you!

Thank you all for your help and support!

- Sigurd

Contents

Summary **v**

Preface **vii**

Contents **xii**

Glossary **xiii**

1 Introduction **1**

 1.1 Background 1

 1.1.1 Fixed-wing vs. multi-rotors 3

 1.1.2 Beyond visual line-of-sight operations (BVLOS) 4

 1.1.3 Position sensors 5

 1.1.4 Camera-based navigation 6

 1.1.5 Sensor fusion 7

 1.2 Motivation 8

 1.2.1 Phased array radio system (PARS) 9

 1.2.2 Reconfigurable sensor payloads 9

 1.3 Contributions 10

 1.4 Publications 13

 1.4.1 Additional publications 14

I	Hardware Sensor Synchronization	17
2	SyncBoard - A High-Accuracy Sensor Timing Board for UAV Payloads	19
2.1	Introduction	19
2.2	System overview	22
2.2.1	System clocks	23
2.2.2	Connectors	26
2.2.3	SyncBoard sensor output	28
2.2.4	Sensor message queueing	30
2.2.5	SyncBoard configuration and status	31
2.2.6	Performance	33
2.3	Experimental results	35
2.4	Conclusion	41
3	User-Configurable Timing and Navigation for UAVs	43
3.1	Introduction	43
3.1.1	Related Work	45
3.1.2	Contributions	47
3.2	Sensor System Timing Integration	47
3.3	Hardware Sensor Timing	49
3.3.1	Communication Overview	50
3.3.2	Data Envelope	51
3.3.3	Sensor Input Handler	52
3.3.4	Sensor Timing Hardware Output	56
3.3.5	Sensor Configuration	60
3.4	SenTiBoard	62
3.4.1	SenTiUtils	64
3.4.2	SenTiStack	66
3.4.3	Verification	68
3.5	Conclusions	72
3.A	Ring buffer memcpy	74
II	Navigation in GNSS-Denied Environments	77
4	Inertial Optical Flow for Throw-and-Go Micro Air Vehicles	79
4.1	Introduction and related work	79
4.2	Computation of optical measurements	81
4.2.1	Camera velocity and terrain-plane parameter computation	82

4.3	Temporal misalignment and non-flat terrain	84
4.4	Extended Kalman filter	88
4.5	Performance evaluation	91
4.6	Conclusion	94
5	Navigation of UAV Using Phased Array Radio	97
5.1	Introduction	97
5.2	Phased array radio navigation system	99
5.3	Experimental system overview	99
5.3.1	Real-time kinematic GNSS	101
5.4	Ground radio pose estimation	104
5.5	Experimental results	105
5.5.1	Raw radio measurements	106
5.5.2	Barometric vertical correction	107
5.6	Conclusion	110
6	Phased Array Radio System Aided Inertial Navigation for Unmanned Aerial Vehicles	113
6.1	Introduction	113
6.1.1	Chapter overview	114
6.2	Preliminaries	114
6.2.1	Notation	114
6.2.2	Coordinate frames	115
6.2.3	Inertial measurement units	115
6.2.4	Strapdown equations	116
6.3	Phased array radio system positioning	116
6.3.1	PARS base station pose	116
6.4	Nonlinear observer for aided INS	119
6.4.1	Aiding sensors	119
6.4.2	Attitude observer	120
6.4.3	Translational motion observer	121
6.5	Full-scale test setup	122
6.6	Results	123
6.6.1	Reference measurements	123
6.6.2	Performance metrics	124
6.6.3	Raw PARS measurements	124
6.6.4	Results: aided INS	124
6.7	Updated results	132
6.8	Summary	134

7	Robust and Secure UAV Navigation Using GNSS, Phased-array Radio System and Inertial Sensor Fusion	135
7.1	Introduction	135
7.1.1	Main contribution	136
7.1.2	Outline	136
7.2	Positioning	137
7.2.1	PARS and barometer positioning	137
7.3	System implementation	138
7.3.1	NLO and TMO observers	138
7.3.2	Spoofing detector	139
7.3.3	Kalman filter detection	140
7.3.4	Covariance based detection	141
7.3.5	Output smoothing	141
7.4	Experimental results	142
7.4.1	Ground station	142
7.4.2	RTK Spoofing	143
7.5	Results	143
7.5.1	Spoofing setup	143
7.5.2	Tuning	143
7.5.3	Performance metrics	144
7.5.4	Discussion	145
7.6	Concluding remarks	146
8	Conclusions	151
8.1	Summary and conclusions	151
8.2	Recommendations for further work	153
A	Appendix	155
A.1	Payload power control board	155
	Bibliography	159

Glossary

AES advanced encryption standard
AGL above ground level
AHRS attitude and heading reference system
AME absolute mean error
AOA angle of attack
ARS angular rate sensor

BLOS beyond line-of-sight
BVLOS beyond visual line-of-sight

CTS clear to send

DC direct current
DMA direct memory access
DOA direction of arrival
DoF degrees of freedom
DUNE DUNE: unified navigation environment

ECEF Earth centered Earth fixed
ECI Earth centered inertial
EKF extended Kalman filter
eMMC embedded multi-media controller
ENU East-North-up

FIFO first in, first out

GigE gigabit ethernet

GNSS global navigation satellite system

GPIO general purpose input/output

GPS global positioning system

I²C inter-integrated circuit

IC input capture

ICSP in-circuit serial programming

IMU inertial measurement unit

INS inertial navigation system

IR infrared radiation, used to specify cameras recording in the infrared spectrum

IR-LSQ iterative re-weighted least squares

ISR interrupt subroutine

LDO low-dropout regulator

LF low frequency

LIDAR light detection and ranging

LTV linear time-varying

MBps megabyte per second

Mbps megabit per second

MDS microcontroller development system

MEKF multiplicative extended Kalman filter

MHT multiple hypothesis tracking

MiB mebibyte = 2^{20} bytes = 1024 kibibytes = 1048576 bytes

NLO non-linear observer

NTNU Norwegian university of science and technology

NTP network time protocol

OF optical flow

OS operating system

PARS phased array radio system

PCB printed circuit board

PVA position velocity attitude

PVT position velocity time

RANSAC random sample consensus

RGB red green blue, used to specify cameras recording images in colors visible for humans

RMS root mean square

ROS robot operating system

RS-232 recommended standard 232

RS-422 recommended standard 422

RTK real-time kinematic

RTS ready to send

RTT round-trip time

SD secure digital

SenTiBoard sensor timing board

SLAM simultaneous localization and mapping

SNR signal-to-noise ratio

SPI serial peripheral interface bus

SPOF single point of failure

SSD solid-state drive

STD standard deviation

SUAV small unmanned aerial vehicle

SVD singular value decomposition

TMO translational motion observer

TOA is the time-of-arrival – the time when a full sensor message has arrived at the receiver

TOT is the time-of-transport – the time when the first byte of a data package is received

TOV is the time-of-validity – the time when a sensor reading was validated; often referred to as PPS (pulse-per-second) in timing applications

TTL transistor-transistor logic

TWI two-wire interface

U(S)ART universal (synchronous and) asynchronous receiver-transmitter

UAS unmanned aircraft system

UAV unmanned aerial vehicle

UHF ultra high frequency

USB universal serial bus

USGES uniform semiglobal exponential stability

USV unmanned surface vehicle

VHF very high frequency

VO visual odometry

WVGA wide video graphics array

YAML YAML ain't markup language

1.1 Background

Up until the early 2000s, unmanned aerial vehicles (UAVs) with beyond visual line-of-sight (BVLOS) capabilities were almost exclusively available for military use. In recent years, advances in UAV-dependent technologies, such as lithium-polymer batteries with reduced weight and increased capacity, efficient brushless DC-motors, and availability of cheap and lightweight sensor systems for navigation, have made UAVs increasingly more popular. Today, you can go to an electronics store and buy a UAV with a camera, a navigation system and a stabilizing, waypoint following controller for a few hundred US dollars. With these advances in UAV technology we are already seeing many interesting cases where use of UAV technology can save both time and labor.

For example, in search-and-rescue [1, 29], where fast response time can be life-critical, UAVs can be a very useful tool to quickly get an overview of a large area, and to use tools such as infrared cameras to look for people in dangerous areas. Environmental hazard monitoring, traffic management and pollution monitoring in cities and water are other areas where UAVs have been used [98, 149].

There is also much development in the research sector in several fields, such as biology, where UAVs are used for example for observing seasonal changes in freshwater marshes [85] or collection of hyperspectral images which can be used to detect environmental changes [37], and even astronomy with the Mars helicopter which will be used to give an overview of the area around a Mars rover [48].

We also see development for industrial applications, such as power line

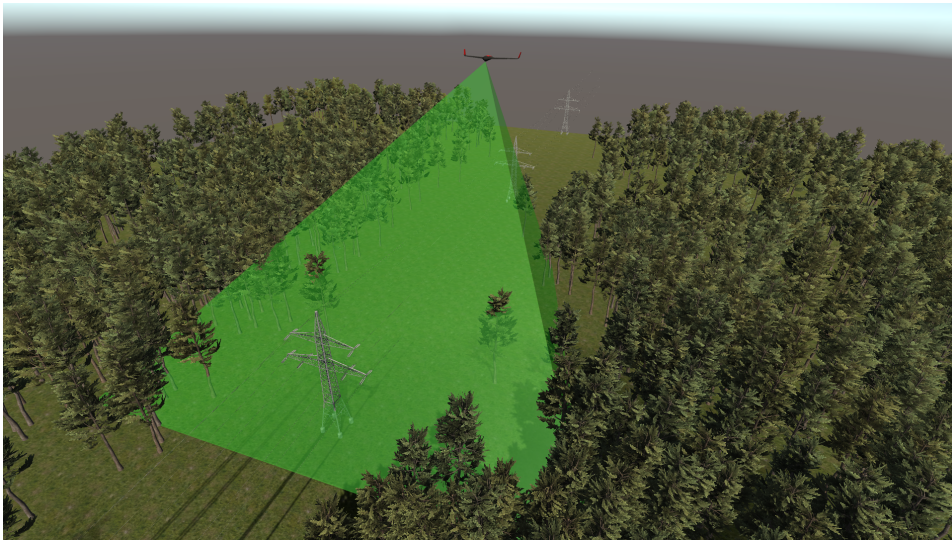


Figure 1.1: Illustration of a power-line inspection task, where a UAV is used for monitoring the vegetation around a power line. The green frustum represents the area visible by a camera.

inspection [25] and bridge inspection [30], which today are very tedious tasks, with high associated costs for regularly needed maintenance. Another industrial application which is expected to rely on UAVs in the future is precision agriculture. We already see development in this field, as can be seen in [59, 132]. UAV applications in precision agriculture typically relies on georeferencing, that is relating camera images to places in the real world. To accurately do direct georeferencing, a high-quality position and attitude solution is needed, along with proper time-synchronization of camera images. We also see UAV usage in civil engineering, where UAVs are used to collect 3D point clouds from images which are used to make models for construction sites [120]. To estimate the reservoirs levels in the summer, power companies based on hydroelectric power in regions where much water are bound up in snow are interested in the snow depth monitoring [24]. This is also true for ski resorts and emergency personnel monitoring avalanche risks. An illustration of a power-line inspection application is given in Figure 1.1.

These are just a few of the many applications of UAVs we see in the field today, but a common challenge for all these applications is that to perform the tasks, robust and accurate navigation is required. Today's systems rely heavily on GNSS to provide absolute position references to function properly. Without an aiding GNSS reference, position estimates will drift

and deteriorate over time. This is further discussed in Section 1.1.3.

1.1.1 Fixed-wing vs. multi-rotors

In this thesis we are using the two most common types of UAVs: fixed-wings and multi-rotors. Other UAV types, such as single-rotors (helicopters), flapping wing and fixed-wing hybrids are not in the scope of this thesis. Some tasks are better performed by either fixed-wings or multi-rotors, and which is the best choice for a specific application depends on that application's particular requirements. An overview of pros and cons of fixed-wings versus multi-rotors are given in Table 1.1.

A challenge with both multi-rotors and fixed-wing UAVs is legislation and regulations. Due to the recent developments in the field, legislation lags behind, and in some countries, UAVs are governed under the same regulations as manned aircrafts, some of which are impossible to fulfill on smaller UAVs. One of these rules are the need for robustness, enforced by for example triple-redundant systems, which often is not implementable on a small UAV due to the payload size and weight constraints.

Multi-rotors and fixed-wing UAVs have different positive and negative sides. Multi-rotors are typically easier to fly than fixed-wings, they need virtually no space for take-off and landing, they can hover in place to inspect nearby objects. They have, however, a low endurance, low payload capacity, and can be very sensitive weather conditions, such as wind. Fixed-wings on the other hand have a long range, high speed, and a higher payload capacity than multi-rotors, but typically need a runway or catapult for take-off, a landing strip or net for landing and are typically heavier than multi-rotors, which results in a large momentum when combined with the high speed. As the momentum increases, so does the potential damage the UAV represents if control of the UAV is lost. Multi-rotors also tend to have a more fragile structure than fixed-wing UAVs, and they have multiple motors with multiple rotors, which tends to be fragile as well. Multi-rotors, especially hexa-copters or octo-copters, can sometimes function even if a motor has failed[111].

With both multi-rotors and fixed-wings accurate navigation is required. Multi-rotors can have very rapid dynamics that, if handled correctly, can be used to perform complex maneuvers, such as throw-and-go take-offs. Without proper attitude-control, however, the UAV can behave unstable in the air, as the pitch and roll directly influences the UAV's acceleration in the horizontal plane. When performing measurements with fixed-wing UAVs, such as photogrammetry, improper navigation and timing solutions

Table 1.1: Fixed-wing and multi-rotor pros and cons

	Pros	Cons	Typical application
Fixed-wing	Range	Take-off	Power-line inspection
	High speed	Landing	Maritime surveillance
	Flight time	Weight	Military operations
	Max payload		
Multi-rotor	Hovering	Endurance	Building inspection
	Easy to fly	Payload	Security
	Redundant motors	Sturdiness	Journalism
	Precision control	Weather	
Both	Aerial view	Regulations	Search and rescue
		Robustness	Agriculture
			Communication relay

will result in inaccurate results when referencing the images to the real-world positions.

1.1.2 Beyond visual line-of-sight operations (BVLOS)

When moving to BVLOS operations, two major challenges for UAV operations arises: navigation and communication. To be able to safely operate a UAV when flying BVLOS, the operator needs to know UAV's position, its velocity and its attitude (PVA). To be able to calculate the navigation solution, the UAV is dependent on a set of sensors, typically consisting of: an absolute position reference, typically provided by a GNSS receiver; acceleration and angular rate measurements, provided by an IMU; and a heading reference, provided by either a magnetometer or a multiple GNSS receiver setup. In addition, other sensors such as barometers, airspeed, angle-of-attack and optical flow sensors can be used to complement the sensors to improve the navigation solution or replace sensors if they are not available.

When flying BVLOS, by definition, the UAV is no longer visible by the pilot or operator. This increases the risks associated with performing missions, as safety is strongly dependent on both navigation and communication. With communication links there exist multiple off-the-shelf solutions with long range. Examples are tracking antennas, Iridium satellite communication (and the upcoming Iridium NEXT), phased array radio systems and VHF and UHF solutions. For UAV safety, typically only navigation information is needed to be transmitted, and to ensure safe operations in a control-tower managed airspace, a telemetry update every few seconds

is sufficient, and this does not require a high transmission capacity. By adding multiple redundant links, reducing the risk associated with loss of communication can be reduced. Note that the mission may require a higher transmission capacity to for example transmit camera images.

Due to the extended range of BVLOS, the time spent returning from the flight is increased, which might be critical in case of an emergency. If the UAV consumes more power than expected, or some sort of error occurs, a risk-reducing implementation would be to be able to switch off the power of the non-critical components.

To summarize, special risk analysis containing navigation, communication and power consumption is needed when performing BVLOS operations.

1.1.3 Position sensors

Sensors used for estimation of position can primarily be divided into two categories: absolute and inertial. Absolute position sensors measure position directly, which results in bounded position errors. Inertial sensors, on the other hand, measure derivatives of the position, such as velocity and acceleration, and have unbounded position errors. The reason for this is that the position must be estimate based on the previously estimated position, and noise and other uncertainties accumulate over time. Typical inertial sensors are accelerometers and gyroscopes, which measure acceleration and angular velocity respectively. Typical absolute sensors are GNSS receivers and laser altimeters.

The main advantage of inertial sensors is that they typically have a high bandwidth, which make them able to encompass rapid-changing system dynamics. They are also tending to be lightweight and compact, and they do not need external components, such as antennas. In addition, they are virtually impossible to influence from external sources, making them very robust to malicious intents.

For absolute positional measurements in the horizontal plane, few alternatives to GNSS exist when considering ranges of several kilometers. Since the 1970s the Loran-C low frequency (LF) radio-based system was the primary position solution, but as GNSS solutions have emerged, the Loran-C network has been largely discontinued[72, 138]. Advantages with the Loran-C system are closeness to the transmitters, making physical maintenance and alterations easier, and that the entire system is inside the atmosphere, making it more robust to atmospheric disruptions, such as space weather. Incentives to re-implement Loran-C, or variants such as eLoran, have been

proposed by the US Coast Guard[139], but these systems are not available at the time of writing.

Another alternative for absolute positional measurements is by distributing several radios at different locations. By measuring the round-trip time (RTT) to each of the radios, the distance to each of the radio can be estimated, and a position can be calculated if the angle between the radios is sufficiently large. Although this method can provide accurate results, it has high infrastructural requirements as several radios need to be operative at multiple locations.

1.1.4 Camera-based navigation

In addition to GNSS-based positioning systems and inertial navigation system (INS)-sensors, computer vision can be used to aid position velocity attitude (PVA)-observers. There are two main methods of camera vision: georeferencing and optical flow. Georeferencing is the process of detecting real-world objects in an image. If the position of the object is known in the real-world, this would give heading and elevation angles toward the observer. In addition, if the size of the real-world object is known, and can be detected by the algorithm, the range to the object is known, and the position of the observer can be calculated. The last step is, however, often very difficult to properly estimate due to the quality of camera images from high altitudes. For robustness and to suppress false positives, several objects should be detected simultaneously. As georeferencing relies on detecting known objects, it is mostly performed in controlled environments, such as UAV laboratories, with marker-based solutions, and when performing missions in unknown environments, optical flow algorithms are preferred.

Optical flow relies on detecting the same features in consecutive camera images, and thus deduce the movement of the camera between the images. Opposed to the georeferencing approach, the observed objects do not need to be known in advanced, and automatic detection of features that are likely to appear in consecutive images can be performed. A commonly used method for automatically detecting such features is the Shi-Tomasi detector, described in the aptly named article *Good Features to Track*[119]. When several such features are found in the first image, the optical flow algorithm attempts to detect corresponding features in the next image. Detection of features is typically done using a template matching algorithm, that is, selecting a small patch around a feature in the first image and finding a similar patch in the second image. A vector is then created from the feature in the first image to the corresponding feature in the second image. This

vector now represents the optical flow from the first to the second image, which again represents the motion of the camera (in the opposite direction of the features).

As finding new features to track typically is computationally expensive, the features detected in the second image can be used to search for features in the third image, and so on. When doing this optimization, however, the number of tracked features will decrease over time. This can be either due to the tracked feature is no longer visible by the camera, either that the source of the feature is outside of the image frame or due to obstruction by another object, or it can be that the matching algorithm is unable to find the template in the new image. When an insufficient number of features are tracked, the detection algorithm can be re-executed and the matching process continues.

1.1.5 Sensor fusion

To combine all the sensor readings into one PVA solution a sensor fusion algorithm is used. In the current state-of-the-art there are several alternatives, such as the extended Kalman filter (EKF), the multiplicative extended Kalman filter (MEKF), and non-linear observers (NLOs). The extended Kalman filter is a nonlinear version of the Kalman filter which linearizes the system around an operational point and solves the linearized version optimally. A limitation with the EKF is that it typically stores the attitude estimates as the Euler angles; roll, pitch and yaw. This representation is problematic in two situations; when the UAV is pointing directly upwards and when it is pointing directly downwards, as it then creates singularities with an infinite number of solutions for the yaw/roll axes. To solve this singularity, the multiplicative Kalman filter is a modified version of the EKF that instead of representing the attitude as Euler angles, uses a quaternion - a rotation vector with four elements. This avoids singularities in the solution, and uses quaternion multiplication to alter the attitude estimates, ensuring a consistently valid attitude representation [86].

Another set of sensor fusion algorithms is non-linear observers (NLOs). NLOs guarantees robustness by explicitly calculating stability properties of the observer. They often have global stability properties, making them robust highly to process noise and other disturbances. Opposed to (M)EKF observers, they do not rely on linearizing around a working point but have a global attraction area to a valid solution. They are, however, more difficult to design for optimal results, and a stability analysis must be done for each system - a process which is not necessary when using linearized Kalman filter variants.

1.2 Motivation

In advanced UAV uses such as hyperspectral imaging, photogrammetry, and gravimetry, accurate estimation of the UAV's position, velocity, and attitude (PVA), in addition to the sensor measurements' time of is essential for high-quality results. Hyperspectral cameras typically only record a single scan line per frame, and when flying drones at high velocity, the position changes significantly per frame. For all camera systems, the attitude significantly influences the observed position. Even a small change in the rotation of the UAV results in a relatively large change in the ground position captured by the camera, especially when flying at high altitudes.

To achieve high-quality PVA estimates and time synchronization of camera images, hardware synchronization of sensor messages is needed. As most high-bandwidth sensors, such as IMUs, measure derivatives of the PVA measurements needed for navigation, timing between the measurements are essential to achieve as high accuracy as possible.

When using GNSS-receivers for positioning, the receiver typically has a significant internal processing time. To compensate for this, the receivers typically output a time-of-validity (TOV) pulse at every whole second in UTC-time¹. By registering the TOV and assigning this to the following sensor message, an accurate measure of when the sensor was valid is recorded. Although the sensor message may not be valid at the exact second, a field in the sensor message typically specifies this offset.

Furthermore, when a camera is used, this needs to be synchronized with the other sensors. This can be done either by triggering the camera, i.e. sending a pulse when the camera should capture an image, or by capturing an external trigger from the camera, typically the camera's flash trigger output. The trigger is typically synchronized with the TOV message from a connected GNSS-receiver to simplify the synchronization with the position estimates.

As usage of UAVs transitions into industrial use in large domains such as precision agriculture, with more UAV usage, safe operation becomes increasingly important. In the future it is likely that UAVs become more powerful and carry more equipment than today, which makes them represent a significant danger if controlled with malicious intent. As systems today heavily rely on GNSS for navigation, attacks on the GNSS receivers therefore represents a high risk.

As seen in both national [61] and international media [23, 116] spoofing

¹UTC time is used by GPS, other GNSS variants use different absolute time measurements.

and jamming of GNSS is not only a theoretical implementation seen in research laboratories, but a real-life risk of public concern.

In addition, loss of GNSS due to non-malicious sources, such as hardware or software errors, operation in areas with poor coverage, or operation in areas with high noise or strong reflections, the UAV must still be able to navigate. The same applies for unexpected space weather, where "Solar bursts are a potential threat to safety-critical systems based on GNSS" according to [124]. Although not strictly GNSS *denial*, the selective availability feature implemented in some GNSS systems also represents a threat to accurate navigation. Selective availability significantly reduces the accuracy of GNSS systems by disrupting the available timing measurements [151].

1.2.1 Phased array radio system (PARS)

To be able to handle loss of GNSS, be it from malicious sources, operation in GNSS-denied environments, or from hardware or software failure, a redundant system is needed. Although dead-reckoning using INS-sensors with a magnetometer and airspeed velocity measurements are accurate for short periods of time, absolute position measurements are required for extended duration operations. One method for obtaining such measurements is by using a phased array radio system (PARS), which uses an array of n by m antenna elements in one radio to detect the direction of arrival (DOA) of incoming radio signals. If in addition to the DOA, the RTT is measured, a simultaneous 3-degrees of freedom (DoF) position measurement can be estimated from a single radio at a single location. Multiple methods for estimating DOA exist in current literature[77], perhaps most notably Schmidt's MUSIC [114] and Roy and Kaliath's ESPRIT [110].

Another advantage of the PARS is that it in addition to being an absolute position sensor, communication is provided through the same hardware. By using beamforming, phased array radio systems can deliver transfer capacities of several megabytes over a range of tens of kilometers. Phased array radio systems are thus able to simultaneously solve the two major challenges with BVLOS UAV operations.

1.2.2 Reconfigurable sensor payloads

Due to the limited space and weight available onboard a UAVs, there is often a need to specialize the payload for each specific mission. However, if space, weight and mission restrictions permit doing so, dividing the

payload into flight critical equipment (avionics) and mission sensor equipment (mission-payload), might reduce the risks associated with changing the installed equipment in a UAV for research purposes. An example of a mission-payload for bridge inspection might consist of a regular RGB-camera, a GNSS-receiver, an IMU, and an SSD-disk for storage, while a mission-payload for measuring changes in gravity using MEMS accelerometers could consist of multiple GNSS-receivers (for position and attitude calculation) and multiple IMUs (for gravity measurements). In some cases, payloads must be miniaturized and optimized to minimize the weight to allow maximal flight-time, while in other cases flight-time is less important, but only the highest quality measurements using multiple sensors will give sufficiently accurate results.

In addition to mission-specific payloads, rapid improvements in UAV sensors, such as MEMS IMUs and GNSS receivers, causes static payloads to be quickly outdated. To be able to keep up-to-date with the advances in technology, sensors must be easily integratable with sensor solutions. By having a reconfigurable solution, the efforts required for system integration is greatly reduced, as redesigning and production of a new system is not needed.

When testing an experimental payload, an important feature is being able to power the payload on and off. This is because the mission-payload may have to be turned off to conserve power, it may create electromagnetic disturbances or behave unexpectedly in some other way. In addition, the mission-payload may contain expensive equipment that, in an emergency, must be unpowered. For example, when performing an emergency landing in water, some electronic components may survive if depowered. A solution for a system that does this is described in Section A.1.

1.3 Contributions

This thesis is split into two parts. The first part describes requirements, design and implementations of hardware sensor timing solutions, while navigation in GNSS denied environments is discussed in the second part.

Part 1 - Hardware Sensor Synchronization

Chapter 2. SyncBoard - A High-Accuracy Sensor Timing Board for UAV Payloads

This chapter presents the SyncBoard, which is a reconfigurable sensor timing board that accurately records the Time of Validity (TOV) from sensors,

while simultaneously acting as a protocol conversion tool. It is intended to be usable by people without expertise in hardware development. To allow accurate timing, the SyncBoard has a high-performance microcontroller with a 32-bit timer clock running at a maximum of 100 MHz. By capturing the TOV signal using the Input/Interrupt Capture function of the microcontroller, sensor triggers can be captured with an accuracy of 10 ns, which is sufficient for most UAV applications.

The SyncBoard supports sensor input from several commonly used protocols, in addition to simple digital and analog input. Sensors of different rates and data sizes are supported, and TOV detection can be configured to comply with the specifications of the sensor. For usage with very high-bandwidth sensors, such as cameras, the SyncBoard can output an external signal to be used as a trigger for the sensor. To be compatible with most on-board computers on UAVs, the SyncBoard primarily uses a USB interface for configuration and sensor output, but output can also be transmitted by any of the available protocols if needed. The SyncBoard also has functionality to switch on and off power to individual sensors as needed.

Chapter 3. User-Configurable Timing and Navigation for UAVs

Sensor timing using dedicated hardware is the de-facto method to achieve optimal sensor performance, but the solutions available today have limited flexibility and require significant effort when changing sensors. This chapter presents requirements and suggestions for a highly accurate, reconfigurable sensor timing system that simplifies integration of sensor systems and navigation systems for UAVs. Both typical avionic sensors, like GNSS receivers and IMUs, and more complex sensors, such as cameras, are supported. To verify the design, an implementation named the SenTiBoard was created, along with a software support package and a baseline sensor-suite.

With the solution presented in this paper we get a measurement resolution of 10 nanoseconds and we can transfer up to 7.6 megabytes per second. If the sensor suite includes a GNSS receiver with a pulse-per-second (PPS) reference, the sensor measurements can be related to an absolute time reference (UTC) with a clock drift of 1.9 microseconds per second RMS. An experiment was carried out, using a Mini Cruiser fixed-wing UAV, where errors in georeferencing infrared images were reduced with a factor of 4 when compared to a software synchronization method.

Part 2 - Navigation in GNSS Denied Environments

Chapter 4. Inertial Optical Flow for Throw-and-Go Micro Air Vehicles

In this chapter, we describe a novel method using only optical flow from a single camera and inertial information to quickly initialize, deploy, and autonomously stabilize an inherently unstable aerial vehicle. Our approach requires a minimal number of tracked features in only two consecutive frames and inertial readings, eliminating the need for long feature tracks or local maps and rendering it inherently fail-safe.

We show theoretically, in simulation, and in real experiments that we can reliably estimate and control the vehicle velocity, full attitude, and metric distance to the scene while self-calibrating inertial intrinsics and sensor extrinsics. In fact, the fast initialization, self-calibration, and inherent fail-safe property leads to the first visual-inertial throw-and-go capable system.

Chapter 5. Navigation of UAV Using Phased Array Radio

Navigation and communication are two of the major challenges when flying unmanned aerial vehicles (UAVs) beyond visual line of sight (BVLOS), in particular when Global Navigation Satellite Systems (GNSS) are unavailable. The phased array radio system (PARS) used in this chapter aims to solve both communication and navigation with one system. To enable high efficiency data transfer, the radio system uses electronic beamforming to direct the energy from the ground radio towards the UAV, but to be able to do this, the ground radio needs to know the bearing and elevation angles towards the UAV. By measuring the round-trip time to compute the range and observing the direction of the incoming signal to compute the bearing and elevation angles, the phased array radio system is able to find the position of the UAV, relative to the ground radio, in all three dimensions.

The phased array system is shown to provide absolute measurements for UAV navigation in radio line of sight, which can be used as a redundant system to GNSS measurements. By merging the radio measurements with barometer altitude data, a mean accuracy of approximately 24 m with a standard deviation of approximately 17 m compared to the real time kinematic (RTK) satellite navigation solution is achieved on a UAV flight at a distance of approximately 5 km.

Chapter 6. Phased Array Radio System Aided Inertial Navigation for Unmanned Aerial Vehicles

This chapter uses the phased array radio system (PARS) described in Chapter 5 to aid a micro-electro-mechanical inertial navigation system (INS), estimating position, velocity and attitude. The solution is independent of global navigation satellite system (GNSS) for positioning and highly resistant to malicious sources, such as spoofing and jamming.

The state estimator presented in this chapter fuses range and bearing measurements from the PARS with the measurements from an on-board inertial measurement unit, a magnetometer and a barometer. By aiding the INS with PARS position measurements, magnetometer readings and barometric measurements, drift-free PVA estimates are obtained. The PARS measurements can be used for navigation alongside today's GNSS solutions, or as a redundant backup system running in parallel.

To validate the observer, an experiment was carried out with a fixed wing UAV on an approximately 35-minute flight with a maximal distance of 5.35 km from the base station. During this flight a root-mean-square (RMS) accuracy of 26.3 m compared to a real-time kinematic GNSS solution was achieved. Note that the updated results chapter is not in the original publication, which show an RMS positional accuracy of 17.6 m on a 37-minute flight with a maximal distance of 4.75 km from the base station.

Chapter 7. Robust and Secure UAV Navigation Using GNSS, Phased-Array Radio System and Inertial Sensor Fusion

As shown in Chapter 5 and Chapter 6, PARS equipment has the potential to provide position measurements that are accurate within tens of meters. As PARS solutions typically have significantly higher signal-to-noise ratio (SNR) and strongly encrypted messages, they are robust towards typical malicious attacks against GNSS solutions.

This chapter presents a method for an inertial navigation system which is aided using redundant position sensors. The high-accuracy RTK GNSS solution is the primary position reference, when it is available. The PARS is used to detect if GNSS solution is being spoofed or jammed and is used as the fall-back positioning solution. The proposed system is verified through an experiment where GNSS-spoofing is added during post-processing.

1.4 Publications

This thesis is based on the following publications:

1. S. M. Albrektsen and T. A. Johansen. SyncBoard - a high accuracy sensor timing board for UAV payloads. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1706–1715, Miami, Florida, USA, June 13–16 2017. doi: 10.1109/ICUAS.2017.7991410
2. Sigurd M. Albrektsen and Tor Arne Johansen. User-configurable timing and navigation for UAVs. *Sensors*, 18(8):2468, 2018. ISSN 1424-8220. doi: 10.3390/s18082468. URL <http://www.mdpi.com/1424-8220/18/8/2468>
3. S. Weiss, R. Brockers, S. M. Albrektsen, and L. Matthies. Inertial optical flow for throw-and-go micro air vehicles. In *2015 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 262–269, Waikoloa, Hawaii, USA, January 5–9 2015. doi: 10.1109/WACV.2015.42
4. S. M. Albrektsen, A Sægrov, and T. A Johansen. Navigation of UAV using phased array radio. In *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 138–143, Linköping, Sweden, October 3–5 2017. doi: 10.1109/RED-UAS.2017.8101657
5. S. M. Albrektsen, T. H. Bryne, and T. A Johansen. Phased array radio system aided inertial navigation for unmanned aerial vehicles. In *2018 IEEE Aerospace Conference, Big Sky, Montana, USA, March 3–10 2018*. doi: 10.1109/AERO.2018.8396433
6. S. M. Albrektsen, T. H. Bryne, and T. A Johansen. Robust and secure UAV navigation using GNSS, phased-array radio system and inertial sensor fusion. In *IEEE Conference on Control Technology and Applications (CCTA)*, Copenhagen, Denmark, August 21–24 2018

1.4.1 Additional publications

In addition to the publications this thesis is based on, the following publications were made during the PhD:

1. Håkon H. Helgesen, Frederik S. Leira, Torleiv H. Bryne, Sigurd M. Albrektsen, Thor I. Fossen, and Tor Arne Johansen. Real-time georeferencing of thermal images using small fixed-wing UAVs in maritime environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2018. (submitted)

2. M. Ludvigsen, S. M. Albrektsen, K. Cisek, T. A. Johansen, P. Norgren, R. Skjetne, A. Zolich, P. Sousa Dias, S. Ferreira, J. B. de Sousa, T. O. Fossum, Ø. Sture, T. Røbekk Krogstad, Ø. Midtgaard, V. Hovstein, and E. Vågsholm. Network of heterogeneous autonomous vehicles for marine research and management. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–7, Sept 2016. doi: 10.1109/OCEANS.2016.7761494

Part I

Hardware Sensor Synchronization

SyncBoard - A High-Accuracy Sensor Timing Board for UAV Payloads

This chapter is based on S. M. Albrektsen and T. A. Johansen. SyncBoard - a high accuracy sensor timing board for UAV payloads. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1706–1715, Miami, Florida, USA, June 13–16 2017. doi: 10.1109/ICUAS.2017.7991410

2.1 Introduction

The SyncBoard is a configurable, small form factor, microprocessor-based synchronization board that aims to provide a user-friendly way to read from a variety of sensors, while assigning highly accurate timestamps to each sensor message. To achieve accurate sensor message timestamping, the SyncBoard uses a microprocessor with Interrupt/Input Capture (IC) capabilities to detect and capture Time Of Validity (TOV) triggers from sensors, and assigns the captured 32-bit timer data to each sensor data package. To be compatible with a large variety of unmanned aerial system (UAS) applications with different sensor packages, the SyncBoard supports several different protocols, namely UART, RS-232, RS-422, SPI, and CAN protocols, and can easily be reconfigured by the end user to efficiently read from the sensors. To make the SyncBoard accessible for users, both sensor data transfer and configuration is primarily done through a Universal Serial Bus (USB) 2.0 interface.

High accuracy timing of data is needed in many navigation applications, as an unknown timing delay can greatly decrease the accuracy of estima-

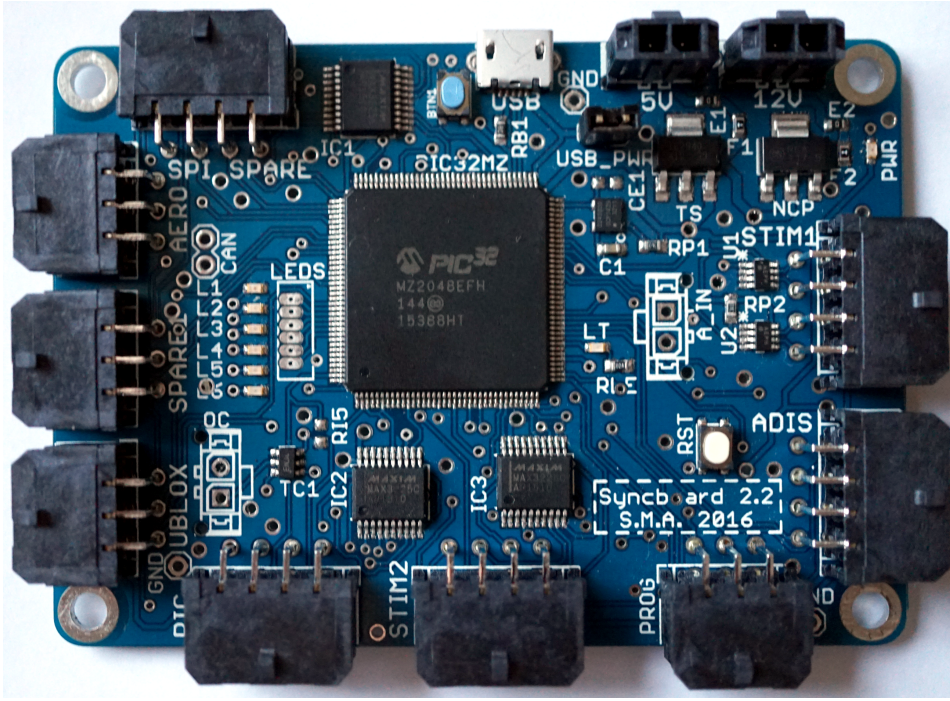


Figure 2.1: The top side of the Syncboard v2.2

tion filters, as described in [49, p.114]: "When Kalman filter measurements compare the outputs of two different navigation systems, it is important to ensure that those outputs correspond to the same time of validity. Otherwise, differences in the navigation system outputs due to the time lag between them will be falsely attributed by the Kalman filter to the states, corrupting the estimates of those states. The greater the level of dynamics encountered, the larger the impact of a given time-synchronization error will be."

Similarly, [122] concludes: "A large time synchronization error between the sampling instants of the GPS receiver and the IMU sensor in a GPS-aided INS has been observed to seriously degrade the performance of the system."

To address the synchronization problem, multiple hardware solutions for sensor timing have been developed. In [78] a time synchronization system that synchronizes analog input with GPS data using a laptop and an analog to digital converter was developed. Per the experiments done by [113] the velocity errors rise linearly with time delays, while the root mean square (RMS) error of the position errors did not change greatly due to the direct

access to a positional measurement. Several other publications are using specialized hardware solutions for synchronizing Inertial Measurement Unit (IMU) and Global Navigation Satellite System (GNSS) data [31, 27, 28, 121, 79, 108].

Several systems for triggering imaging systems synchronized with GNSS clock signals have also been developed. Due to its usage in stereo vision, Light Detection And Ranging (LIDAR) 3D mapping and for photogrammetry, an unmanned system that can capture images or point clouds with a highly accurate attitude and camera position is very valuable. In [112], the dedicated hardware synchronizes two cameras and an IMU to obtain depth images from the stereo camera and ego-motion using a Gumstix Overo. [64] uses a "parallel hardware device called SyncBox" to synchronize camera and GPS data with an accuracy of 5 us, however, the SyncBox is not further described in the article. In [45] a GlobalSat DG 100 GPS logger is used to trigger a camera and synchronize the images with GPS position for photogrammetry.

This chapter presents a hardware synchronization board that in addition to providing highly accurate timestamps of sensor messages time of validity, focuses on re-usability and user friendliness. The current state of the art in hardware synchronization is to have a specialized board for synchronizing a specific, limited set of sensors, typically three or less, that needs major rework if a sensor is changed. Experiences from our unmanned vehicles laboratory show that sensors needed for each application vary, along with the available space and weight of the payload. In addition, sensors are replaced as new models with better specifications become available. Because of the frequent change of sensors, the SyncBoard is designed to be easily configurable to the different protocols, data formats and data rates that the sensors use. With high reconfigurability, a small and light form factor, external sensor triggering, and support for simultaneous recording of up to 8 accurately timed high-rate sensors, the SyncBoard surpasses both commercially available alternatives and the state of the art in hardware sensor timing boards for UAVs.

In this chapter we will first present an overview of the SyncBoard system, with timing specifications, hardware capabilities, output specifications and performance evaluation from a lab environment. Section 2.3 then discusses experiments and publications where the SyncBoard was used on different platforms and with different sensor packages. Finally, the chapter is summarized and concluded in Section 2.4.

2.2 System overview

When using a USB to RS-232 or RS-422 device to read from a sensor, on a standard computer with a non-real-time operating system, there are several steps that can introduce a delay from the time of validity to the message is available to the user.

1. There might be a delay from when the sensor reading is valid until the first bit of the data is transferred.
2. A delay is introduced due to the finite transfer speed of the data from the sensor to the USB device.
3. Instead of sending one byte at the time, the USB device might read several bytes at once, then send them all in one USB package.
4. Typically, USB communication is initialized by the host which polls the connected devices to check if the device wants to send data. This might introduce a delay when waiting for the host to poll the USB device for new data.
5. The data needs to be transferred from the USB device to the host. Data transferred over USB is sent in a package structure, which adds slightly to the overall transfer time.
6. The driver on the user computer might buffer transferred data instead of transmitting one byte as soon as it has been transferred.
7. The software that reads the data from the sensor might need time to become ready to process the received sensor message.
8. The current timestamp is read from the system when the software has received the data.

When using a microcontroller instead of a standard computer, and a sensor with a Time of Validity (TOV) trigger, most of the steps above are eliminated, and the data has a timestamp after the following steps:

1. The sensor outputs a TOV signal at the instant the measurement is registered by the sensor.
2. On the next clock cycle, the microcontroller registers the transition on one of the Interrupt Capture (IC) pins which registers the current value of an internal timer.

This second method of timing sensor messages clearly gives a much more precise and accurate timestamp; however, it requires additional hardware in the form of a microcontroller, and designing functional, efficient, and small systems with microcontrollers both requires expertise and is time consuming.

Although companies such as Arduino[12] aim to make microcontrollers more accessible through their inexpensive, easy-to-use, open source platform, knowledge about programming both microcontrollers and real-time systems is needed to implement a platform that reads data and provides accurate timestamps to sensor messages. In addition, when combining data from several sensors it is very convenient to collect as much of the timing data as possible on one board. During our flights, we often wanted to collect data from a large variety of sensors: a GNSS receiver, IMUs from different manufacturers, a laser altimeter, a magnetometer, a camera, and the autopilot. If all these sensors were to be attached to an Arduino, there would not be enough Interrupt/Input Capture (IC) ports for each sensor to have its own dedicated IC.

It is often important that the end-users can easily switch between different sensors. Different applications require different sets of equipment, and a timing board needs to be easily configurable to adapt to the projects' needs. This emphasizes the need for easy reconfigurability, as we want users without experience in hardware development to be able to use the SyncBoard efficiently and to interface sensors as needed.

To fit in a wide range of UAV payloads, the SyncBoard has been designed with a small form factor of 87 mm by 63 mm, with a height of 13 mm. It has four mounting holes, one in each corner, and a weight of 31 g. The SyncBoard's power consumption is measured to be 0.22 A at 5.0 V, but the power consumption may increase as external sensors are powered through the SyncBoard.

2.2.1 System clocks

On an unmanned vehicle system, there are typically several clocks that needs to be synchronized. If we have a UAV with the following system: an autopilot with a GPS receiver and a magnetometer, an on-board computer with a connected camera, and a SyncBoard with a GPS receiver and tactical grade IMUs, there are four different clocks that need to be synchronized. An overview of this system can be seen in Figure 2.2. The first clock is the internal clock from the autopilot, which is typically given in milliseconds from the autopilot was powered on. The second clock is the "real-time" clock (RTC) on the on-board computer, which typically uses a 32.768 kHz

crystal to keep the time in seconds since midnight January 1st, 1970. The third clock signal is the timer on the SyncBoard which is an unsigned 32-bit timer that counts from 0 to 2^{32} , then overflows and starts counting from 0 again. The last clock is the calculated GPS time from the GPS receivers, which is regarded as the most accurate clock, because it based on the atomic clocks on the GPS satellites.

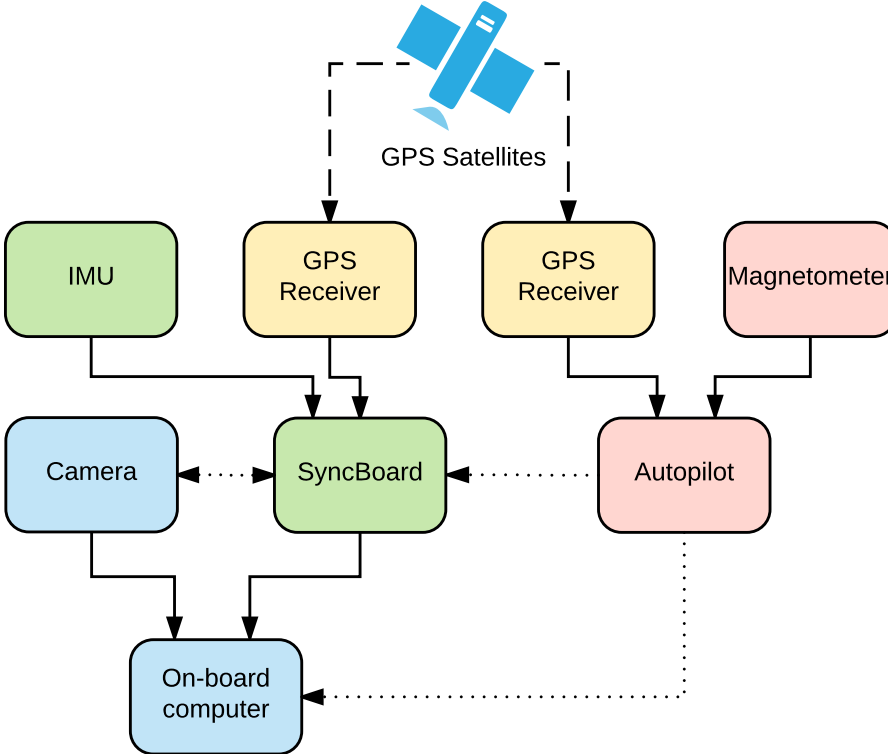


Figure 2.2: System clock overview. The solid lines mark wired connections, the dashed lines mark wireless signals and the dotted lines mark optional connections.

To be able to relate the messages in one timeframe to messages in another timeframe, single messages with timestamps two or more different sources are used. The GPS clock and the SyncBoard are accurately synchronized through the timestamped TOV pulse from the GPS receiver (PPS - pulse per second) and the incoming sensor message containing the GPS timing solution. These messages contain both the SyncBoard clock and the GPS time at the time of recording, and consequently give a simultaneous sample of both clocks, which is used for synchronizing timeframes. The autopilot

also receives, timestamps and logs messages containing the GPS time, but with a timestamp from the autopilot's clock. Because of this the messages logged by the autopilot, in the autopilot's timeframe can be related in time to the messages logged by the SyncBoard in the SyncBoard's timeframe through the highly accurate GPS clock. An alternative solution for this synchronization would be to have the autopilot output data directly to the SyncBoard.

If the on-board computer has a GPIO (General Purpose Input/Output) pin which can be accurately triggered, this can be used as a synchronization trigger that is sent to the SyncBoard as a TOV signal at one of the input ports. If no such pin is available, accurately relating the measurements in the on-board computer's timeframe with measurements in the SyncBoard's timeframe is more challenging, due to the varying delay of the USB protocol. This may not be needed in many applications, but a specialized routine can nevertheless be executed on the SyncBoard to measure the round-trip time USB messages, which timestamps the message as it is sent from the on-board computer ($t_{oc,1}$), then when it is received by the SyncBoard ($t_{sb,1}$), before it is sent by the SyncBoard ($t_{sb,2}$) and then when it is received by the on-board computer ($t_{oc,2}$). By assuming that the transfer and receive time of the USB is the same, and defining p_{timer} to be the period of each clock tick of the timer ($1 / 100 \text{ MHz} = 10 \text{ ns}$), the time delay in one direction can be estimated to be:

$$\Delta t = \frac{(t_{oc,2} - t_{oc,1}) - (t_{sb,2} - t_{sb,1})p_{\text{timer}}}{2}$$

As the clocks of the SyncBoard and the on-board computer will drift independently, this synchronization routine should be performed at regular intervals if data is logged directly on the SyncBoard. However, as this time-delay is only an estimation, either the SyncBoard's TOV or external trigger features should be used in applications that require highly accurate time measurements. To ensure correct timing in the example above, the camera connected to the on-board computer can either send TOV pulses to one of the input ports of the SyncBoard, or the SyncBoard can be configured to trigger the camera with an output-pulse at specified intervals.

Both internally on the microcontroller and in the output sensor package format presented in section 2.2.3, the TOV value is stored as a 32-bit unsigned integer value. This data type is chosen because it is the highest timer value that can be configured by the microcontroller. With a 32-bit unsigned timer running at 100 MHz the system can run for approximately 42.95 s before overflowing.

To handle timer overflow, the on-board computer should timestamp the sensor-messages as they are received. As sensor data is continuously transferred from the SyncBoard to the on-board computer, the number of overflows of the TOV timer can easily be calculated if the messages are timestamped by the by the on-board computer in its time-frame. Thereby accurate timing of the package can be easily calculated by multiplying the number of overflows with $2^{32}p_{timer}$.

2.2.2 Connectors

Table 2.1: Overview of the main SyncBoard connectors

Sensor ID	Protocol	Power	Connector	IC	OC
Sensor 1	RS-232	5 V	8-pin	✓	
Sensor 2	RS-232	5 V	6-pin	✓	
Sensor 3	RS-232	5 V	6-pin	✓	
Sensor 4	RS-422	5 V	8-pin	✓	✓
Sensor 5	TTL UART	5 V	6-pin	✓	
Sensor 6	RS-232 / RS-422	5 V	8-pin	✓	
Sensor 7	SPI	3.3 V	8-pin	✓	✓
Sensor 8	SPI	3.3 V	8-pin	✓	
Sensor 9	CAN		2-pin		
Sensor 10	Analog input		2-pin		

To support a variety of current and future devices, the SyncBoard provides several input ports with different protocols. The microcontroller used on the SyncBoard, the Microchip PIC32MZ2048EFH, has 6 UART (Universal asynchronous receiver/transmitter) connections and all of these are connected to output ports on the SyncBoard. One UART port is available at transistor-transistor logic (TTL) voltage level only, three are available with a 1 Mbps RS-232 protocol through MAX3225EAP microchips, one is available with a 20 Mbps RS-422 protocol through a pair of MAX3362EKA microchips, and the last port can switch between a hardware flow controlled RS-232 port and a receive-only RS-422 port. Two of the RS-232 ports have support for hardware flow control using two additional wires; namely Request To Send (RTS) and Clear To Send (CTS). There are also two SPI (Serial Peripheral Interface) ports that can be used for sensors, with a maximum speed of 50 MHz, one Controller Area Network (CAN bus) port and

one analog input. In addition, most pins can be used as simple digital inputs or outputs. Table 2.1 lists a quick overview of the communication ports on the SyncBoard.

To provide power and timing capabilities, the TTL UART ports and non-hardware flow controlled RS-232 ports have 6-pin connectors on the SyncBoard, while the hardware flow controlled RS-232 ports, the RS-422 ports and the SPI ports have 8-pin connectors. The extra pins on each port consist of ground, power and dedicated Interrupt/Input Capture (IC) pins. The power pins are controlled by the microcontroller through 2.4 A Vishay SIP32402ADNP load switches, with 3.3 V for the SPI devices and 5 V for the others. By allowing the microcontroller to individually control the power for each of the connected sensors, a sensor can be automatically be restarted if a persisting fault is detected, without restarting the entire system. The interrupt capture pins are specialized interrupt pins that, in addition to triggering an interrupt in the microcontroller, also capture the value of a 32-bit timer at the same time as the edge of the interrupt is detected. This provides the highly accurate timing information which is the most important feature of the SyncBoard.

The board also supports accurate timing of signals by sending a trigger signal to a port with an empty package. This is useful for example when recording camera images which are too large to be read directly through the SyncBoard. When using this type of sensor package timing, correlating timestamps on the SyncBoard with the correct sensor packages is challenging, especially if the frame rate of the camera is high. One might solve this problem by making sure that the camera starts capturing images after the SyncBoard is recording, and then counting the images, assigning the first image to the first trigger of the SyncBoard. Synchronization issues can occur if an image is captured by the camera, and thus triggering the SyncBoard, but then the image is not transferred correctly. As images typically are not retransferred, as there usually is not enough throughput to both allow the retransfer of the dropped images and still send the incoming images, the frames are dropped. When a frame is dropped, the n -th image might therefore be assigned the $(n-1)$ -th timestamp. This issue can be solved by choosing a camera that has an internal counter of the captured camera images and that sends this information along with every camera frame, or make sure that the frame rate is low enough that the timestamps uniquely match to a SyncBoard trigger.

In addition to the Interrupt Capture, one of the RS-422 ports and one of the SPI ports have an Output Compare port, which can be used to trigger sensors externally. This can for example be used to synchronize two different

IMUs with an interpolated signal from the GNSS receiver's GPS time-pulse (PPS), to compare their performance. There is one more output compare connector which is intended to be used for triggering a camera to synchronize images with GPS locations. This signal goes through a Microchip TC1240A charge-pump voltage doubler to increase the signal voltage as some cameras, such as the IDS GigE uEye UI-5250CP, require a minimum of 5 V to register an input trigger signal.

The analog input port and the CAN port do not have interrupt capture pins by default, but one of the other ports' IC pin can be used if needed. The input of the analog pin on the microcontroller has a range from 0 V to 3.3 V. To scale the input signal down to a maximum of 3.3 V a voltage divider is provided on the board. By default, the resistors in the voltage divider are 1.0 k Ω and 4.7 k Ω , giving a scaling factor of $\frac{1000}{1000+4700} \approx 17.5\%$. The CAN bus runs at up to 1 Mbps with full CAN 2.0B compliance [94, p. 485].

When connecting to the SyncBoard using a USB port, the SyncBoard can draw power from it as well, but due to the 0.5 A current restriction of the USB 2.0 standard, the port might not provide enough power for all connected sensors. If a higher current is required, additional power can be provided through a connector on the board. The 5 V input, regardless if the source is from the USB port or the external connector, has a 5 A fuse to protect both the components on the SyncBoard itself and the connected sensors that are powered from the SyncBoard.

2.2.3 SyncBoard sensor output

To make the SyncBoard available for most on-board computers on UAVs, sensor data is transferred through a USB 2.0 port by default, but one of the sensor ports can be used as a data-output port if needed. To guarantee the integrity of the sensor packages and that the proper timing data is sent, the SyncBoard wraps each sensor message in its own package format.

In Figure 2.3 the package format that the SyncBoard uses when sending sensor data to the on-board computer can be read. *Sync* is a set of synchronization bytes and should always be 0x425E. *Length* is the number of bytes in DATA + 14 bytes. The 14 extra bytes are the package header bytes (bytes number 4 through 15), plus an additional 2 for the package checksum, which are excluded from the sensor message length as they are always included in the message. The sync bytes and the length itself are excluded from the package length to make it easier to read the remainder of the package on the on-board computer. The *ID* field is the sensor ID that sent the data, which is required to differentiate between sensors when

having several of the same sensor connected, or sensors with a similar package structure. *Res.* is a reserved byte which is currently set to 0. *CHK_H* is a two-byte header checksum, the Fletcher-8 checksum [35], calculated as described in Listing 2.1. The Time of Validity (*TOV*) is the time when the last IC was triggered. If TOV-triggering is not supported by the sensor, the TOV field can be set to the Time of Transport (TOT), which is the time when the first byte of the message was transferred. The Time of Arrival (*TOA*) is the time when the sensor message has finished transferring to the SyncBoard and marks the earliest time the package can be parsed and used in an online application, such as a state estimator or filter. The *CHK_PKG* field is a checksum calculated for the data after the *CHK_H* field in the package, using the same method as for *CHK_H*, to the integrity of the remaining data. The 0-bytes at the end of the package is a padding to ensure that the package aligns with a full 4-bytes to minimize the work for the microcontroller when sending data using USB. When using Direct Memory Access (DMA) the PIC32MZ requires the source data buffer to be aligned to a 4-byte offset, if not it will round the start byte down to the closest 4-byte offset. During some experiments, an additional TOT field was added after the TOV field, to measure the difference of these timings. Three different methods for buffering sensor messages are described below.

Listing 2.1: Checksum calculation

```
uint16_t get_checksum(char *data, size_t datalen) {
    // A and B are 8-bit numbers that will overflow
    // such that 255 + 1 = 0
    uint8_t A = 0;
    uint8_t B = 0;

    for (size_t ix=0; ix < datalen; ix++) {
        A += data[ix];
        B += A;
    }

    // Combine A and B to one 16-bit number
    return (A << 8) & B;
}
```

¹The zero bytes are the zero to three extra padding zeros required by the queuing method in section 2.2.4

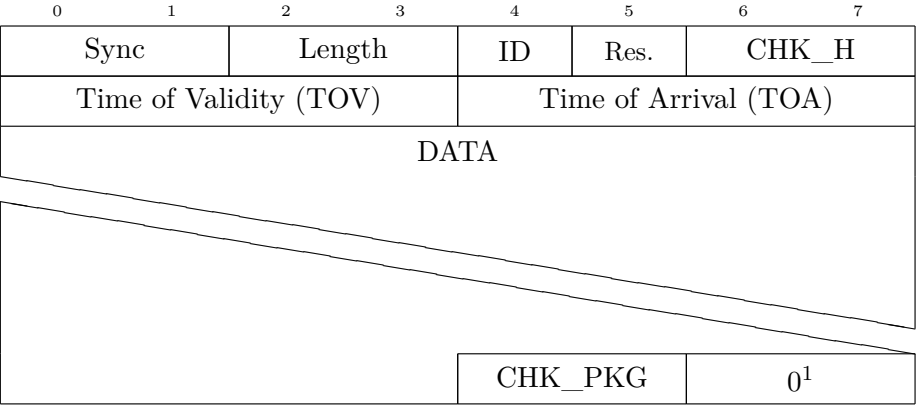


Figure 2.3: Output sensor package format as sent from the SyncBoard.

2.2.4 Sensor message queueing

When transferring data packages of different sizes and rates, such as when using a GNSS receiver with raw satellite readings at 10 Hz and a high frame rate IMU at 2000 Hz, an efficient queueing method is needed to correctly transfer all the messages. To ensure that the transferred data is properly aligned after each sensor package, each package is padded with 0-bytes so that each package is divisible by 4. By doing this, the SyncBoard can send each queued package through the DMA-enabled USB protocol of the microcontroller, and thereby the time spent when scheduling USB-writes is low. Low USB-write schedule delays are critical as this code might be executed while in an interrupt routine, and the time spent while inside of an interrupt routine should be minimized. Pseudo code for this queueing method is given in Listing 2.2. Note that code for handling queue over-/underflow and interrupts are removed from this listing to better show the queueing strategy. By containing the scheduled data in a single array, the differently sized packages are packed on top of each other, minimizing the amount of allocated space needed for package queueing.

Listing 2.2: Sensor message queueing

```
char[QUEUE_SIZE] queue;
uint16_t back_ix;
uint16_t front_ix;
uint16_t queue_size;

usb_queue_package(char *data, uint16_t datalen) {
    uint8_t extra = 4 - (datalen % 4);
```

```

uint16_t fixed_len = datalen + extra;
memcpy(data, &queue[front_ix], fixed_len);
front_ix = (front_ix + fixed_len) % QUEUE_SIZE;
queued_size += fixed_len;

// Schedule write if one is not already scheduled
if(!write_scheduled)
    usb_schedule_write();
}

usb_schedule_write() {
    // Calculate how much data we can write
    // without exceeding the buffer sizes
    uint16_t to_write;
    to_write = min(QUEUE_SIZE - back_ix, queue_size);
    to_write = max(to_write, USB_SEND_MAX);

    // Schedule the write
    pic32_write_data(&queue[back_ix], to_write);
}

usb_write_finished(uint16_t bytes_written) {
    queue_size -= bytes_written;
    back_ix = (back_ix + bytes_written) % QUEUE_SIZE;

    // Stop if there is no more data to write
    if(queue_size == 0) {
        write_scheduled = false;
        return;
    }

    usb_schedule_write();
}

```

2.2.5 SyncBoard configuration and status

Although the main usage of the SyncBoard's USB port is to transfer sensor data from the SyncBoard to an on-board computer, configuration of the SyncBoard and status messages from the SyncBoard can also be read through the USB connection. Configuration of the SyncBoard can either be done by connecting to the board using a serial port communications program, such as minicom[95] or GNU Screen[42], or by using a Graphical User Interface (GUI) program written in the PythonTM programming language specifically created for this purpose on an external desktop computer.

When connecting through the command line interface status information and debug messages can be shown. Commands that have been implemented

at this time are: showing the current rate at which messages are received; showing the last received message from a sensor; showing the value of the main timer; and turning the power to a sensor on and off. Periodic reporting of the rate of sensor messages can also be turned on, so that the rate of each sensor is printed every second. These debugging tools have shown to be very useful, especially when setting up the configuration to a new sensor for the first time.

As the SyncBoard endpoints are configured as USB communications device class (CDC) type endpoints, many Linux distributions ("distros") such as Debian and Ubuntu have drivers automatically installed by the kernel. Using these drivers, the SyncBoard typically shows up as either a `/dev/ttyAMC` or a `/dev/ttyUSB` device, with a number appended. The number that is appended depends on how many other tty-devices are already in the system which makes it difficult to differentiate which device is the SyncBoard. To easily find the SyncBoard, an udev-rule[137] is written to match the USB specified `idVendor` and `idProduct` of the SyncBoard, with the extra parameter `SYMLINK+="ttySyncboard%E{ID_USB_INTERFACE_NUM}"`. This tells udev to create two symbolic links to the SyncBoard, `/dev/ttySyncboard00` and `/dev/ttySyncboard02`, one for each of the USB endpoints. The configuration endpoint is called `Syncoard02`, opposed to `Syncoard01`, due to a bulk transfer endpoint defined by the CDC class configuration on the endpoint with ID 1.

To quickly show the current status of the board without the use of an external computer, a set of Light Emitting Diodes (LEDs) are placed on the board. These are very convenient as a last-minute check that the system is running as expected, for example before launching a UAV. There are two green LEDs, four yellow LEDs and two red LEDs. The first green LED is a simple *power present* LED, which lights up if 3.3 V is present on the board. This is useful to show if a fuse has blown, or if there is something else wrong with the power supply. The second green light is linked to the main timing loop of the SyncBoard and blinks at a rate of 2 Hz. The yellow LEDs are tied to the sensors and they light up if there are errors in the sensor data, or if the rates of the sensors are not within the expected limits. One of the red LEDs light up if there are repeated errors in the reading of sensor messages or the USB transfer buffer is full, and the last red LED is lit if the USB connection is missing or faulty.

To enable configuration of the device without rewriting the microcontroller's internal flash, by a non-volatile ferroelectric RAM (F-RAM) circuit on the PCB. The device can now be reconfigured without the use of an external programmer to easily switch between different sensors and data

rates.

To reconfigure the SyncBoard, a user connects using the same physical connection that the sensor data is read from, but through a different USB endpoint. Here the user can change baud rates, protocol line control, synchronization bytes, interrupt capture set-ups, SPI polling data, and external triggering signals for cameras or IMUs. A transition diagram showing the possible states of the SyncBoard, and how the configuration file is loaded, can be seen in Figure 2.4.

2.2.6 Performance

To test that the SyncBoard is able to handle the large data rates provided by the sensors, it was set to simultaneously record data from two high-rate IMUs and one GNSS receiver. The IMUs used were the Sensoror STIM300 and the Analog Devices ADIS16488A, with relatively short messages, and the GNSS receiver was the u-blox LEA M8T, with a lower frame rate, but longer messages. Almost 11 million sensor messages were recorded during this test, and the difference in TOV between each message from each sensor was calculated. Table 2.2 lists the expected and recorded number of clock ticks between each sensor message and Table 2.3 lists the requested and the calculated recorded sensor rates. According to the ADIS16488A datasheet [9, p.31], the timing jitter on the TOV signal for that sensor is $1.4\mu\text{s}$, and the measurements of the ADIS done by the SyncBoard are well within the specified values, with an average difference of $0.157\mu\text{s}$ from the expected value.

Table 2.2: Mean number of clock ticks from a 1-hour recording. All units are in internal clock ticks at 100 MHz, that is 10 ns

Sensor	Expected	Recorded	Std.dev.
ADIS16488A	40 650	40 634.276	79.778
STIM300	200 000	200 002.148	148.766
LEA-M8T	10 000 000	10 000 009.737	13.983

Table 2.3: Calculated rates from 1-hour recording

Sensor	Requested rate	Recorded rate	Difference [%]
ADIS16488A	2460 Hz	2460.974 Hz	-0.039 59 %
STIM300	500 Hz	499.995 Hz	0.001 05 %
LEA-M8T	10 Hz	10.000 Hz	-0.000 02 %

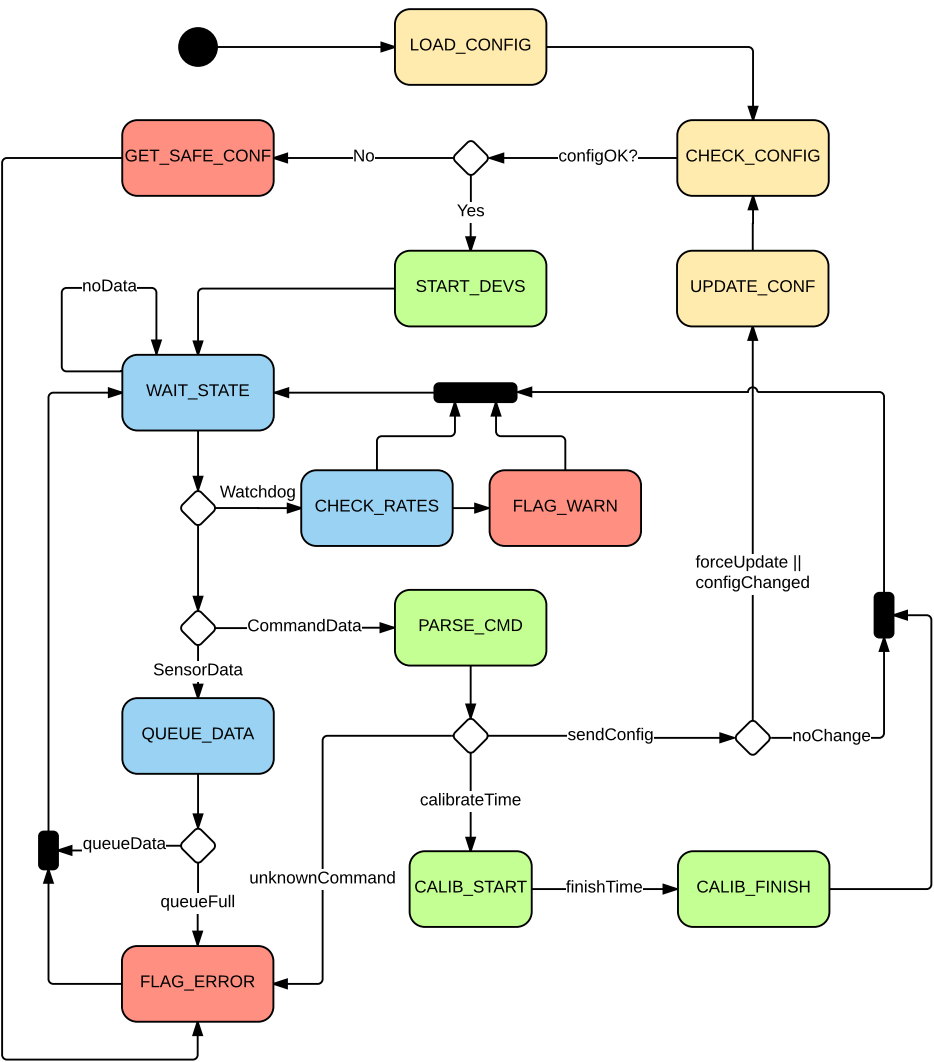


Figure 2.4: SyncBoard transition diagram

2.3 Experimental results

The SyncBoard has already been used in several successful experiments, with different sensor configurations and on different vehicles, since the first version of the SyncBoard was ready in late 2014. This section presents an overview of these results and the SyncBoard's role in the publications. Several yet unpublished experiments as of February 2017 are described at the end of the section.

Several of the experiments were done using a sensor structure as shown in Figure 2.5. A TOV-recorded GNSS receiver (u-blox LEA 6T or LEA M8T), and one or more TOV-recorded IMUs (Sensoror STIM300 and Analog Devices ADIS16488A) are present in most configurations. Additional sensors such as the Honeywell HMR2300 magnetometer, Latitude Engineering AGL laser altimeter, the Aeroprobe air-data probe, a connection to the vehicles autopilot (such as Could Cap's Piccolo II or the PIXHAWK), in addition to a Gigabit Ethernet camera which is triggered by the SyncBoard are added as needed. These are logged to an on-board computer, such as a Intel NUC or a Hardkernel ODROID XU4, and the data is stored either on an embedded MultiMediaCard (eMMC) or, if more storage space is needed, on a solid state drive (SSD). With the connection to the autopilot, extra sensor data and information about operational parameters such as current way-points, mode of operation and engine control signals can be logged.

In [39] the author uses a Sensoror STIM300 with gyroscope, accelerometer, and inclinometer output recorded at 300 Hz, a u-blox LEA-6T GNSS receiver recorded at 10 Hz with a pulse-per-second signal recorded at 1 Hz, along with an IDS GigE uEye UI-5250CP color camera with an 8 mm lens. The IMU and the GNSS receiver are recorded through the SyncBoard, and the camera is triggered by the SyncBoard, to guarantee that the images are synchronized with the GNSS receiver's measurements. The measurements were carried out using a Penguin B fixed wing UAV platform at Eggemoen Aviation and Technology Park. These measurements allowed Fusini et al. to confirm the validity of the analysis and the design of a uniformly semi-globally exponentially stable (USGES) non-linear observer for estimation of attitude, gyro bias, position, velocity and acceleration for a fixed-wing UAV. Experimental verification of a continuous epipolar optical flow (CEOF) non-linear observer was performed in [62] using the same sensor kit and the same UAV.

Furthermore, in [40] the same IMU, GNSS receiver and camera were used, but in this experiment a Honeywell HMR2300 magnetometer, running at 1 Hz and a Latitude Engineering's Above Ground Level Sensor running

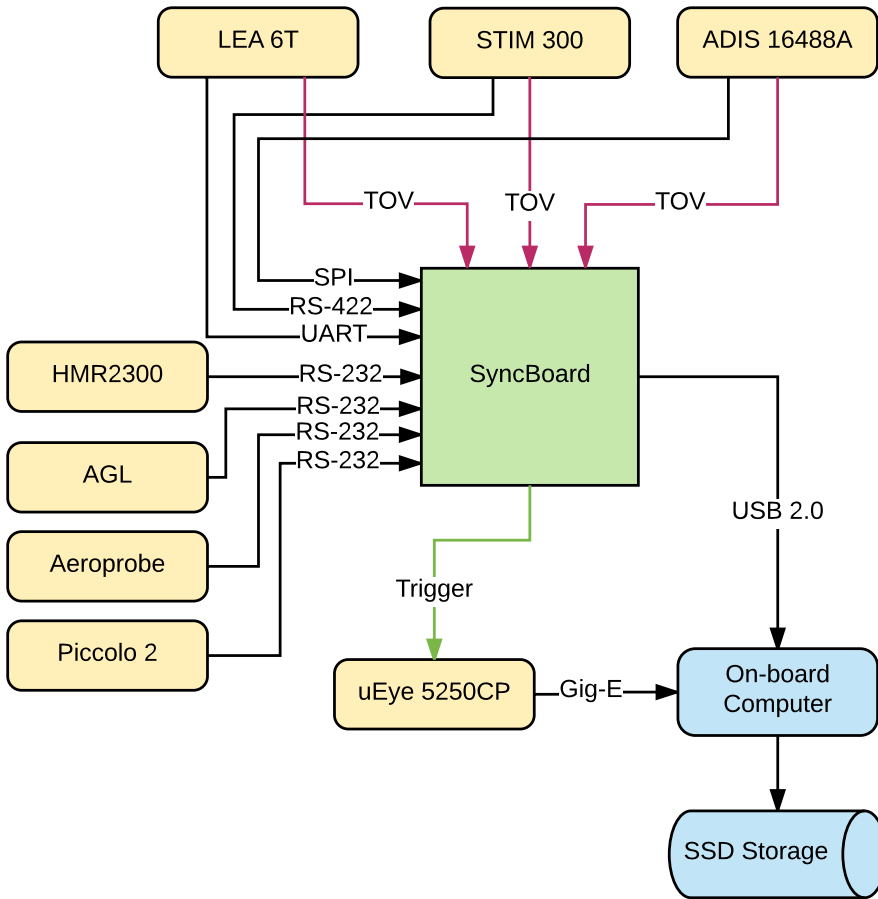


Figure 2.5: SyncBoard payload configuration with simultaneous recording from a GNSS receiver, two IMUs, a magnetometer, a laser altimeter, an air sensor probe, and the autopilot. The Camera is triggered to be synchronized with GNSS measurements, but the images are read through the on-board computer. SSD storage is needed due to the size of the camera images. This configuration was primarily flown in a UAV Factory Penguin-B fixed wing UAV platform.

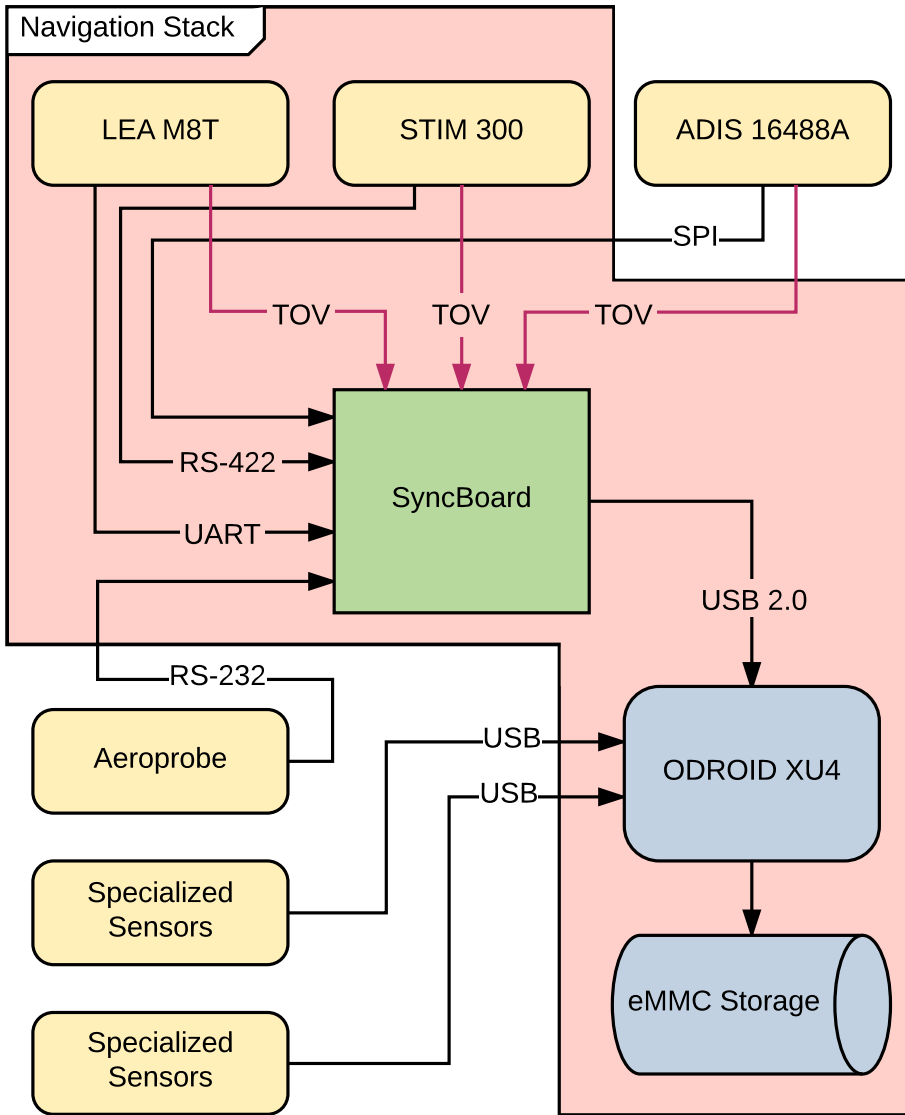


Figure 2.6: SyncBoard payload configuration with an Aeroprobe, a set of pressure sensors, a pair of accelerometers connected to the wing tips of the plane, an IMU in addition to the Navigation Stack (Figure 2.7). This sensor setup was flown in a Skywalker X8.

at 1 Hz were added. In this paper the authors proved global exponential stability (GES) of their proposed observer and verified it experimentally using data captured with the SyncBoard.

In [68] the authors have developed an observer that estimates wind velocity, Angle-Of-Attack (AOA) and Sideslip Angle (SSA) of a UAV, only using the kinematic relationships, thus bypassing the need for an aerodynamic model of the aircraft. A u-blox LEA-6T at 5 Hz with a base station for RTK post processing, an ADIS16488 IMU at 410 Hz and a Piccolo autopilot with 1 Hz measurements of air speed data were logged through the SyncBoard. Using the experimental data collected through the SyncBoard, the authors conclude that "the estimates appears to have slightly stronger correlations for AOA and significantly better correlations for SSA than the ones achieved by the Piccolo sensor suite."

In [51] the author uses a similar setup, with a u-blox LEA-6T GNSS-receiver at 5 Hz, an ADIS16488 IMU at 410 Hz in a Penguin B. In this paper the effect of time delay of signals in observers and methods for handling such delays are discussed and experimentally verified. This sensor setup has also been used in [69] and [19].

The SyncBoard is not only usable in UAVs, in [52] it is used in a manned aircraft to verify "a method for handling time-delayed GNSS measurement in a loosely-coupled strapdown GNSS/INS system." In an unpublished experiment the SyncBoard was also used to collect data on board an unmanned surface vehicle (USV), namely the Maritime Robotics' (MR) Telemetron.

A collaborative experiment between the Norwegian University of Science and Technology (NTNU), Kongsberg Seatex, Underwater Systems and Technology Laboratory (LSTS), and Maritime Robotics (MR) was done in April 2016 where the SyncBoard was used on board the Skywalker X8 fixed wing UAV to log STIM300 IMU and u-blox LEA-M8T data[82]. A similar experiment was carried out with NTNU, LSTS, KTH Royal Institute of Technology and MR in November 2016, where in addition to logging data, live camera images were streamed from the UAV to both the UAV command center and to the MR Telemetron [100].

There have also been several experiments, unpublished as of February 2017, where a phased array radio antenna system has been added, in addition to a GNSS receiver and a STIM300, on a Skywalker X8 fixed wing UAV. From these experiments we are expected to extract valuable insight in position estimation using phased array antenna systems, and data for observer verification.

From the autumn of 2016 through the winter of 2017 experiments using the high accuracy air-data probe from Aeroprobe have been conducted. The

results from these experiments are yet to be published, but the aim is to verify several novel methods for estimation of wind velocity, angle-of-attack and lift coefficients. For these data sets additional special sensors are added and logged directly to the on-board computer, as shown in Figure 2.6.

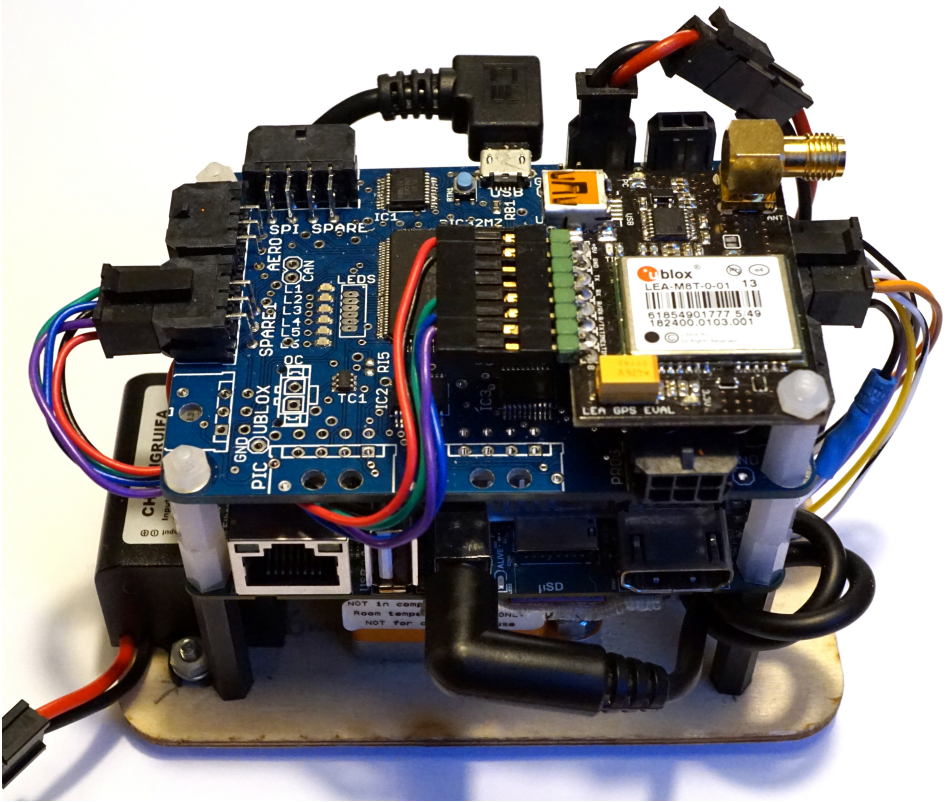


Figure 2.7: The Navigation Stack consists of a u-blox LEA M8T GNSS receiver, a STIM300 IMU, the SyncBoard, and an ODROID XU4 on-board computer with eMMC storage. This stack provides a complete plug-and-play navigation logging solution with a tactical grade IMU, the orange block at the bottom, and a GNSS receiver with RTK capabilities on the top.

The above experiments and publications show that the SyncBoard is an important tool to help when reading time-critical data from a variety of sensors. The adaptability of the board has been shown through its usage with various sensors that use different protocols, sampling rates and sensor data sizes. When handling very large data sizes, such as camera images, the SyncBoard has successfully been used as a synchronization trigger to allow georeferencing of the images. The applicability of the SyncBoard has been

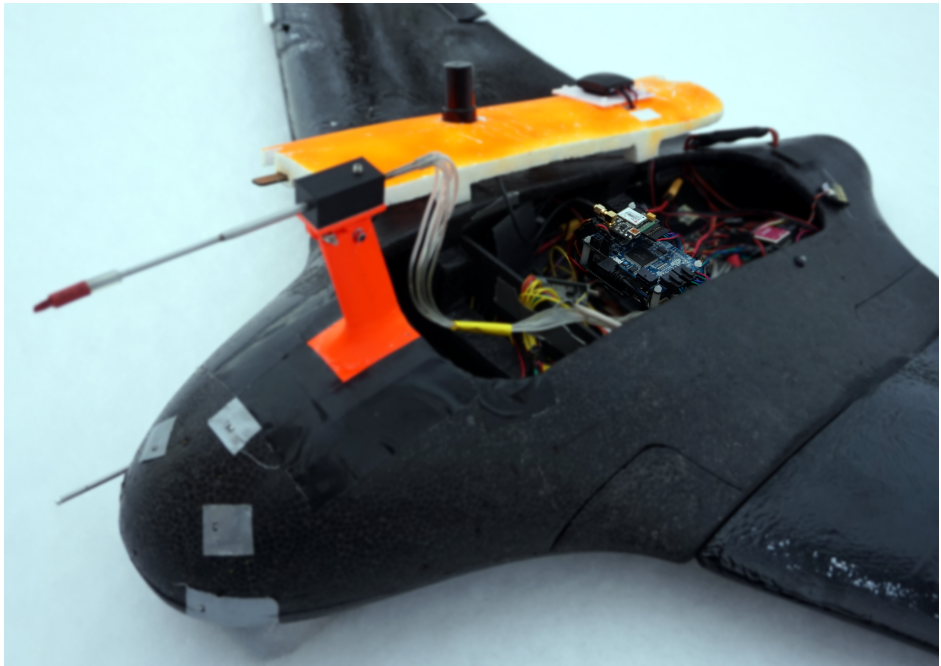


Figure 2.8: The SyncBoard on the Navigation Stack mounted in a Skywalker X8 fixed wing UAV

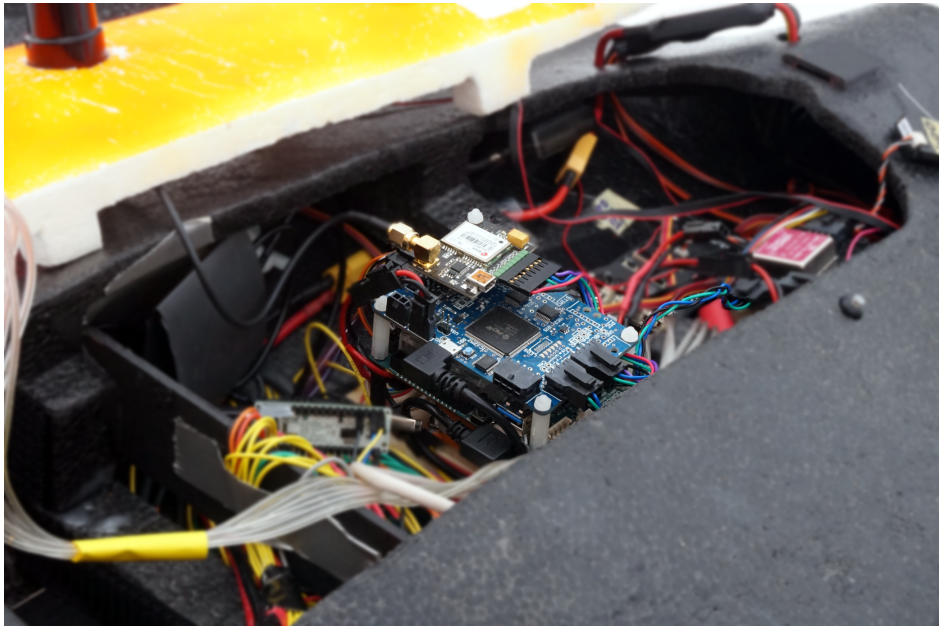


Figure 2.9: A closer view of the mounted Navigation Stack

shown through its usage in collecting data on different vehicles: on UAVs such as the Penguin B and the Skywalker X8, and USVs such as the MR Telemetron, in addition to a GA Slingsby T67C manned aircraft.

2.4 Conclusion

In this chapter the highly accurate, reconfigurable timing and synchronization board, the SyncBoard, has been described, and its role in several research projects have been introduced. A variety of sensors using different protocols have been successfully interfaced and used in multiple experiments. As the SyncBoard can also be used to accurately trigger a camera to synchronize it with GNSS time and position in order to georeference the images which is an essential part of enabling UAVs to perform accurate photogrammetry and 3D imaging.

To test that the SyncBoard can precisely log from high performance sensors, controlled experiments were performed in the laboratory where both high frame rate sensors with smaller data packages, and lower frame rate sensors with longer data packages were recorded simultaneously. The SyncBoard was able to precisely log the data well within the accuracy needed for most high-speed UAV applications.

The SyncBoard presented in this chapter has proven to be a useful in a variety of UAV sensor systems, and has already shown its use multiple experiments, which is verified by several publications until now and more planned publications in the near future. As accurate timing of sensor messages is an important feature needed for precise position and attitude estimation systems, the SyncBoard is a tool that highly simplifies the process of acquiring high-quality sensor data for navigation solutions.

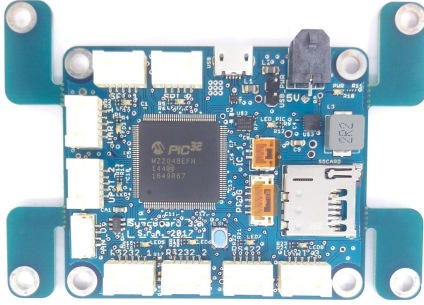
User-Configurable Timing and Navigation for UAVs

The work in this chapter is based on Sigurd M. Albrektsen and Tor Arne Johansen. User-configurable timing and navigation for UAVs. *Sensors*, 18(8):2468, 2018. ISSN 1424-8220. doi: 10.3390/s18082468. URL <http://www.mdpi.com/1424-8220/18/8/2468>

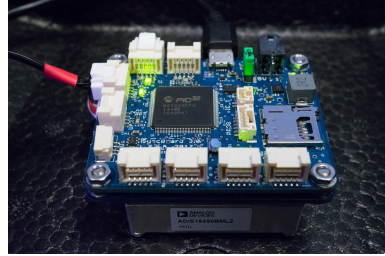
3.1 Introduction

Reconfigurable sensor timing hardware, that accurately records the TOV of the data, is an integral part of a highly accurate and maintainable multi-sensor navigation and image acquisition solution. TOV is often named pulse per second (PPS) in the setting of timing systems and might be different from when it is transmitted. This especially is the case when there are large delays from the TOV to the package is transmitted from the sensor, which typically is the case for global navigation satellite system (GNSS) receivers, which need to analyze the incoming data and calculate a solution before transmitting it[52]. As many sensors, such as inertial measurement units (IMUs), measure derivatives of the desired values (acceleration, angular rate) and not the desired values (position, velocity, attitude) directly, an accurate measure of the measurement's TOV is essential when integrating with positioning reference and image capture systems operating at high frame rates. Moreover, the high speed and fast attitude dynamics of UAVs means that time synchronization errors may have a significant impact on the measurement quality, as the measurement quality does not only depend on the value, but also the temporal accuracy of the measurement's timestamp. Thus, hardware synchronization is required to provide the highest quality measurements possible. An implementation of such a system is depicted

in Figure 3.1. A visualization of errors due to a time delay can be seen in Figure 3.2, where a simulated system with perfect INS and GNSS sensors are integrated with and without a time delay. A very simple sensor fusion algorithm is implemented, where the position estimate is set to the GNSS value when it is received.



(a) The SenTiBoard including removable adapter legs.



(b) The SenTiStack in the smaller configuration.



(c) Example of mounting in a Skywalker X8 UAV

Figure 3.1: The implemented sensor timing board (SenTiBoard) in various configurations. In 3.1(a) the full-sized SenTiBoard with legs for attaching to an ODroid XU4 is shown; in 3.1(b) the SenTiStack is shown in a small configuration, without an onboard computer; and in 3.1(c) the configuration in 3.1(b) is shown mounted in a fixed wing UAV.

Developing embedded systems, especially systems without an OS, is different from writing software for modern desktop computers. Limitations on processing power, threading/interrupt behavior, real-time considerations, memory capacity, memory handling, and high requirements for stability differentiates an embedded system's firmware from conventional software running on a PC. This makes writing correct firmware difficult. Even when considering life-critical embedded systems such as pacemakers and implantable cardioverter defibrillators, recalls or defect warnings over a one-year period

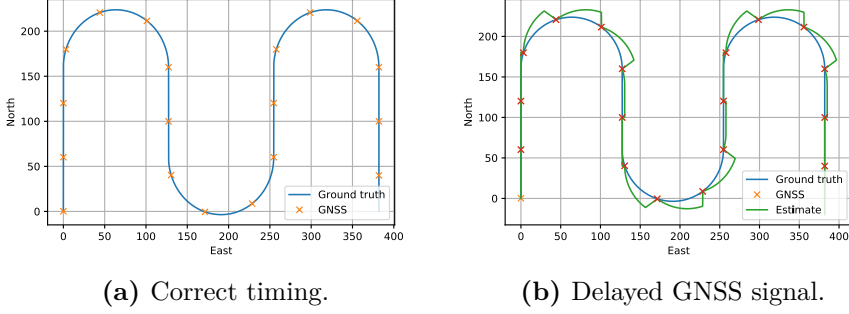


Figure 3.2: Comparison of a system with perfect timing 3.2(a) and a system with imperfect timing 3.2(b). The UAV has a velocity of 20 m s^{-1} an uncompensated GNSS time delay of 1 s. Note that this time delay is exaggerated to show the effect clearly and is larger than an expected delay in an implemented navigation system.

is about one to 15 and nearly one to six, respectively[33], and such systems are expected to be designed to be safe and extensively tested.

3.1.1 Related Work

In the current academic state of the art, several hardware timing solutions have been used, but few have been properly described. Several publications are using some named product with little description or references to details about the hardware and methods used. Examples of such devices are the SyncBox[64], the ADAS[71, 16] with the RefBox[129], the integration platform[27] and the VIASAT[115]. Solutions where the authors measure timestamps in a single integrated system, with specific sensors, also exist. An example is [112], where a Gumstix Overo processor running Linux with a real-time extension is used to read data from an IMU and register hardware synchronization signals from two cameras. Although all these systems seem to provide sufficiently accurate results for their use, they seem highly specialized for the task and there is little focus on maintainability and integration of different sensors.

Autopilots often aim to solve similar problems with hardware synchronization of GNSS solutions with IMU data. Examples of autopilots are the closed source CloudCap Piccolo, and the open hardware systems PIX-HAWK[92] and Paparazzi[17]. The main challenge with the closed source systems is that a very limited number of sensors can be used, and integration of unsupported sensors is virtually impossible as the source code is not available. As an example, the Piccolo only supports two specified

IMUs—the Crista Inertial Sensor and the Navigator GPS/INS Navigation System[22]. Integration of sensors unsupported by the open hardware systems are possible, as the firmware is available, but this can be challenging and requires expertise in embedded software development and alteration of the autopilot’s firmware, which is flight safety critical and should therefore only be altered with care.

To simplify embedded hardware development, several solutions to lower the expertise needed for creating such systems have been created. Examples of such microcontroller development systems (MDSs) are the open-source Arduino electronics platform[12], the Teensy[104], and the discontinued Intel Edison project[65]. In addition, single-board computers such as the Raspberry Pi (R-Pi 3 B+)[38] the ODroid XU4[53] have become popular due to their small form factor and easy operation. Finally, there are traditional microcontrollers such as the Microchip PIC32 MZ[94]. An overview of the hardware capabilities of a selection of these systems is presented in Table 3.1. These systems have different strengths and weaknesses, with the single-board computers typically having more powerful processors, but less support for protocols channels. As described later, in Section 3.3, the number of input captures (ICs) is critical for accurate sensor timing, and thus the two most promising approaches are the Arduino Due and developing a system using a microcontroller. The Arduino Due is, however, lacking when it comes to the number of U(S)ART and SPI ports, but smaller systems could be created with this system.

Table 3.1: Comparison of hardware capabilities of embedded systems.

Name	Type	Clock Speed	IC	U(S)ART /SPI	USB Ver.
Arduino MEGA	MDS	16 MHz	1	4/1	-
Arduino Due	MDS	84 MHz	6	3/1	2.0
Teensy	MDS	180 MHz	2	6/3	2.0
Intel Edison	MDS	500 MHz	0	2/1	2.0
R-Pi 3 B+	Single-board	4×1.4 GHz	0	2/1	2.0
ODroid XU4	Single-board	4×2 GHz	0	1/1	3.0
PIC32 MZ	Micro-controller	252 MHz	9	6/6	2.0

3.1.2 Contributions

This chapter presents a set of required and favorable features of a reconfigurable hardware sensor timing system, that simplifies system integration when time synchronizing sensor data, without compromising sensor accuracy. Suggestions for system modularization; a data frame format; and synchronization, queueing, timing and configuration methods are proposed. By using dedicated hardware features such as IC to record the TOV from sensors, the temporal error is minimized. To verify the validity of the system design, a hardware implementation, visualized in Figure 3.1, was created, where acquired measurements are referenced to a 100 MHz clock, which results in a temporal resolution of 10 ns. To further simplify system integration a software library with utilities and a high-accuracy navigation payload are presented. Results from lab tests and an experiment are presented and evaluated.

This chapter starts by describing key terms related to sensor timing, in addition to proposing required and desired features of a sensor system integration tool in Section 3.2. Then, in Section 3.3, solutions for the challenges with sensor timing are proposed, before an implementation, named the SenTiBoard, is presented in Section 3.4, along with supporting software and hardware. The implemented solution is verified in two different scenarios, before the chapter is concluded in Section 3.5.

3.2 Sensor System Timing Integration

When associating timestamps to sensor data, there are several points in time that can be of interest. First, is the time-of-validity (TOV), which is the time at which the measurement is considered to be valid. Second, is the time-of-transport (TOT), which is when the first detectable part of the sensor message is received by the receiving platform. Finally, the time-of-arrival (TOA), which is when the full sensor message has been received, which denotes the earliest time when another algorithm can verify the sensor message's integrity and parse and use the data. A visualization of these timestamps are given in Figure 3.3, where there are two transmission lines, one signaling when the data is valid, and one transferring the values measured by the sensor. In GNSS systems, the TOV is often called PPS (pulse per second) or 1PPS. We prefer the term TOV over PPS as we find it more descriptive, especially when framerates are different than 1 Hz. If we imagine an object position detection sensor, that analyses where an object is in an image, and then transfers this detection to a computer, these times-

tamps could be as follows. The TOV would be the time when the image was captured by the camera. Then the system would analyze the image and calculate the object's position in the image. Then this result would be transmitted from the sensor and the TOT would be when the start of the sensor message is received at the computer. Note that this time could be significantly delayed, and that this delay could vary with many factors, such as the number of visible features in the image, the available resources on the sensor system, if other data is transferred on the same transmission line, and when the computer is ready to receive the message. The TOA would be when the whole data package has been transferred from the sensor and received at the computer. Although the TOA is the least relevant timestamp of the three, it is the one that is the easiest to implement, as one could write an application that first reads a full message from a sensor, and then assigns a timestamp to the message afterwards.

When designing a sensor timing system there are several features which are required for correct behavior: the resolution of the timestamps needs to be sufficiently high, the system must have enough interrupt capture pins to record TOVs, and the system must support the protocols used by the sensors. In addition, some sensors, such as stereo camera setups, need to be hardware-triggered to achieve high-quality temporal accuracy and synchronization. When using sensors that need to stitch together several sensor readings, high-quality attitude and position estimates are essential for a satisfactory result. An example is hyperspectral cameras, which typically only reads a single line of data per frame. As even small errors in the attitude, for example in a UAV's roll angle, will result in a large error in the measured position, especially at high altitudes, these errors must be minimized. By using sensor timing hardware, the temporal accuracy of each of the data lines can be significantly improved. By having navigation sensors connected to the same system as the hyperspectral camera, the TOV of each line from the camera can be accurately associated to the position and attitude of the UAV at the time of capture. This allows the hyperspectral lines to be transformed from the camera's local coordinate frame, to a global coordinate frame, and simplifying the process of accurately stitching the lines together to a complete image[36].

Furthermore, the system must be able to transmit the incoming messages to a receiving platform and considerations must also be made about the format in which the sensor data is transmitted. In addition to these requirements, there are several preferred features that are not strictly required. Such features are user configurability, a minimum of 32-bit clock timer, integrated storage support, individual sensor power control, inte-

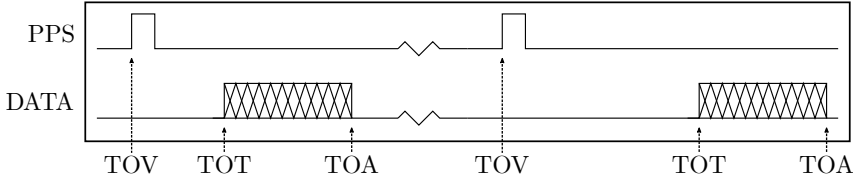


Figure 3.3: Visualization of the time-of-validity (TOV), time-of-transport (TOT) and time-of-arrival (TOA). The PPS signal line produces a rising edge when the measurement is valid, and the data is transferred after a delay. Note this delay between TOV and TOT may vary, for example due to internal calculation delays.

grated sensor navigation fusion algorithms, and system status reporting.

By having a user configurable system, a system integrator can for example upgrade a sensor to a newer model without requiring alterations of the hardware timing system, as long as the system supports the protocol used by the new sensor. With the rapid development in MEMS technology, new and improved sensors are released to the consumer market continuously. The effect of this can for example be seen in the Analog Device’s ADIS product line listed in Table 3.2, where the sensors have improved significantly over the last few years. Note that this table only considers one type of sensor typically found in a navigation payload, from a single manufacturer in a specific price range, and as sensor systems consist of several types of sensors, alterations to the system must be made almost yearly to stay up-to-date with the newest technology.

Table 3.2: ADIS IMU product line gyro stability comparison [11].

Part ID	Release Date	Gyro Stability	Max Sample Rate
16485	Dec. 2012	$6.25^{\circ}/\text{h}$	2460 Hz
16488A	May 2014	$5.1^{\circ}/\text{h}$	2460 Hz
16490	Apr. 2017	$1.8^{\circ}/\text{h}$	4250 Hz
16495	Nov 2017	$0.8^{\circ}/\text{h}$	4250 Hz

3.3 Hardware Sensor Timing

To achieve as accurate timing measurements as possible, a hardware sensor timing system is typically built around a microcontroller with specialized features for parallel registration of hardware interrupts, called input capture (IC), also known as interrupt capture. The IC-pins act as regular interrupt pins, except that in addition to triggering an interrupt subroutine (ISR) when an edge is detected, which is the normal behavior of interrupt pins, the

value of the clock timer when the edge was detected is recorded. This enables a microcontroller to record the time when the value of an IC pin changes, typically on the first clock cycle after the event happened, minimizing the delay of the recorded time.

Several sensors need to be connected to the sensor timing system for it to be able to relate the different measurement timestamps to each other. By using the system to read the sensors' data in addition to the TOVs, synchronization is simplified as the system can associate the timestamps to the sensor messages directly. This is, however, not always possible. Some sensors, for example cameras, send too much data for a typical microcontroller to process. Such sensors can still be synchronized with the other sensors, either by having the timing system send a trigger pulse to the sensor, and/or by registering a TOV signal from the sensor and then associating this trigger with the sensor message at a later stage. As cameras typically have a trigger to an external flash, which is synchronized with the image capture process, this can typically be used as a TOV measurement.

The system proposed in this chapter is divided into several parts. First, an overview of how the different parts of the system communicates is given, followed by a description of the data envelope, which is used to both store timing information and other necessary data about the sensor message. Then, a sensor input handler is presented, which is divided into package separation and synchronization; and timestamp association. An example of how these methods work is also given. Then suggestions for transmission of the received data to an external are made, along with some considerations for implementations on a microprocessor. Finally, two methods for configuring the hardware sensor timing board are presented.

3.3.1 Communication Overview

To keep a hardware sensor timing system's code maintainable, the authors suggest modularizing the design by creating subsystems that interact with each other only through communication channels. This design is inspired by *occam's*[109] or *Go's*[43] concept of channels. By modularizing the system based on how it communicates, we can modify a sub-module with minimal disruption to other sub-modules (p. 203[57]). As the microcontroller typically runs without an operative system or a scheduler/hypervisor, it is not possible to wait for transfer channels to become ready. As the system cannot wait for communication to become ready by both the main loop and an ISR, an alternative scheme for communication must be implemented and suggestions for such systems are given later in this section. During normal operations there are two main communication channel types in the system:

the transfer from a sensor to the microcontrollers internal buffer, and the transfer from the internal buffer to an external computer and/or logging to the SD-card. This system is visualized in Figure 3.4, with the Sensor Handler and the Output Queue modules running onboard the microcontroller.

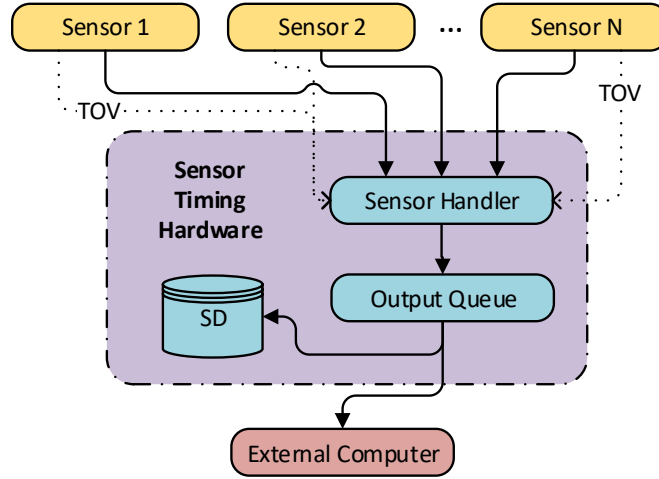


Figure 3.4: Proposed modularization of a hardware sensor timing system, which is modularized by communication.

3.3.2 Data Envelope

To create a unified parsing system that can relate all sensor messages to a common clock, the sensor data must be associated with one or more timestamps. As not all sensors provide a timestamp as a part of the sensor message, and if a timestamp is included it is typically in the sensor's local clock frame, a timestamp from a clock common for all the sensors must be associated with the data. To solve this, each sensor message can be wrapped in an envelope format, such as the one depicted in Figure 3.5. This format is similar to the one presented in Figure 2.3, but it includes an envelope format revision, an optional onboard timestamp (only available when logging to an onboard computer), and the TOT-timestamp. By having a revision ID changes such as which timestamps the messages provide can be distinguished. In the presented version of the envelope there are three clock recordings: the TOV is the time-of-validity – the time when a sensor reading was validated; often referred to as PPS (pulse-per-second) in timing applications, the TOA is the time-of-arrival – the time when a full sensor message has arrived at the receiver, and finally the TOT is the time-of-

transport – the time when the first byte of a data package is received.

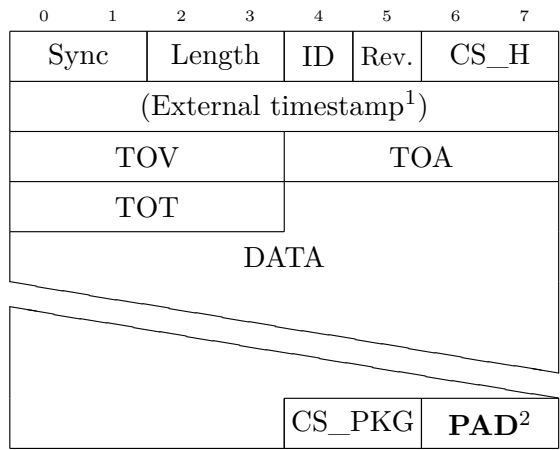


Figure 3.5: A sensor data envelope. To assign timestamps to sensor message packages, sensor data can be wrapped in this format. ¹An external timestamp can be added when the sensor data is transferred to an external computer. ²The **PAD** bytes are zero to three extra padding bytes required by the queueing method described in Section 3.3.4

3.3.3 Sensor Input Handler

There are two main methods of sensor data transfer: asynchronous transfer initiated by the sensor, which is typical for UART and UART-like communication; and polled transfer initiated by the microcontroller, which is typical for SPI communication. To create a layer of abstraction from the communication protocol, each byte received from a sensor can be passed through a sensor handler function. By designing a multi-step module that separates single sensor readings from the datastream, and registers appropriate timestamp information and associates that with each message, only a single module needs to be maintained and tested regardless of where the data was received from. Furthermore, this simplifies the configuration process, as the same configuration interface can be used for all sensor message types. A proposed sensor handler module is described below.

Package Separation and Synchronization

To provide support for a variety of sensors simultaneously, information about how to synchronization with each sensor’s data stream needs to be configured individually for each sensor. The most common methods for synchronization of sensor data are as follows:

1. Start synchronization bytes with fixed length;
2. Start synchronization bytes with dynamic length;
3. End synchronization bytes;
4. Zero-length packages.

The first method is typically used if a sensor sends a binary stream of messages in a single format, for example from an IMU sending accelerations and angular rates. In this mode, the sensor handler module does not need to parse the length of the package from the data-stream itself as it is already specified by the format.

The second method is typically used if a sensor outputs messages of different lengths, for example a GNSS receiver that outputs satellite information. These messages typically have varying length based on how many satellites are visible at the time. To be able to use this in a binary stream, the sensor needs to provide how many bytes a package consists of – the message’s length (L). In this mode the sensor handler module needs information about where in a message L is stored, how many bytes L consists of and the endianness and signedness of L . Although a sensor message cannot have a negative length, one can imagine a format where negative values are used to indicate errors, and therefore the signedness must be specified. In addition, this configuration must be able to be configured to read a specified number of additional bytes, such as header files not included in L .

The third method is most commonly used if the sensor is configured to send data in a human readable (ASCII) format. To avoid erroneous division of packages, the end synchronization bytes must be a series of bytes that is guaranteed to not be a part of a sensor message, except, of course, at the end. When used with a human readable format, where each line of text is a sensor reading, the newline character (`\n`), optionally in combination with the carriage return (`\r`), can be used. An example of such formats is the NMEA protocols for GNSS data [99].

The final method is intended for use with systems that do not send any data to the sensor timing hardware itself, except for a single pulse. This is the case for camera frames that are read by an external computer, but sends a TOV message, for example from the camera’s flash output, to the sensor timing hardware. With such sensors, synchronization is trivial: as soon as a trigger is received a package with the recorded timestamp is transmitted to the sensor handler.

In addition to these methods one might imagine a fifth method using a temporal delimiter where the sensor sends a sensor measurement, followed

by a pause, followed by a new distinct sensor measurement. The sensor timing system could then detect the pause, either by a timeout or by comparing a timestamp since the previous byte was received, and separate the measurement as needed. Note that the latter method is simpler to implement but introduces a larger delay from the message has been received until it is transferred to an onboard computer.

When using the two first synchronization methods the system may become desynchronized if an error occurs in the communication link with a sensor. A desynchronized situation is detected if the bytes after reading all the package data are something else than the synchronization bytes. If this is the case, the system should raise a warning flag and skip bytes until the synchronization bytes are received. In the third and fifth modes, the system cannot be desynchronized, but the sensor can send an arbitrary long package without sending the end synchronization byte sequence. If this happens a buffer-overflow warning should be flagged, and the sensor timing hardware can clear the already buffered package and start reading the next sensor message. If an operator is supervising the process, the error message can be acknowledged and cleared by him or her by sending a command to the timing system.

Timestamp Association

Timestamp association is done in two separate places, first after the synchronization bytes of the package has been read and then when the whole package has been transferred. The timestamp from when the first byte was received is stored as the message's TOT, and if the sensor supports a dedicated TOV-output, the timestamp of the last received TOV for that sensor is associated with the sensor message. Note that the TOV might not update for every message. For example, if a GNSS-receiver outputs satellite observations for use with real-time kinematic (RTK)-GNSS at a rate of 10 Hz, but the navigation (and thereby time) solution is only sent at a rate of 1 Hz, the receiver might only produce a 1 Hz TOV signal (PPS). In these scenarios, the delay from TOV to TOT can be estimated using the messages with a time solution and compensated for by assuming that this delay is known and constant[52].

When the last byte of the sensor message is handled, the TOA is assigned. This value represents the earliest possible time when an application could start to process the sensor data. The TOA message is mostly used for post-process analysis of the system and is not a vital part of a real-time application.

When all the bytes of a sensor package are read, and the package has

been assigned the timestamps, the information should be stored in a structured format. In this step, the total length of the data and the timestamps are copied to the correct places in the sensor data envelope (Figure 3.5), and checksums are calculated and inserted. By reserving space for these fields in the buffer before the sensor data is collected, additional copying due to moving the data is avoided and only the 22 bytes ($8 + (3 \times 4) + 2$ for header, three timestamps and checksum) are written to the process. It is, nonetheless, vital that this operation is not performed in an ISR, as the checksum calculation must iterate the whole sensor package before finishing, which might take more time than is acceptable to spend in an ISR.

Example Message

Imagine that we have a uBlox GNSS receiver connected to the sensor hardware timing board, and that it sends a short message with two bytes (in addition to its own headers) to the board. The binary sensor message could look like this: [B5 62 05 01 02 00 FF FF 1E 61], in addition let's say that we have are not currently synchronized with the stream, and therefore we receive some other data before the message. The full message we receive is therefore: [aa aa B5 62 05 01 02 00 FF FF 1E 61 B5 62]. According to the uBlox datasheet we have configured the system to look for a synchronization id: [B5 62] and a message length in a little-endian format of two bytes, located 4 bytes into the package, with 8 bytes in addition to the length we read due to the uBlox header and the checksum. This would correspond to the second synchronization method – start synchronization with dynamic length.

Before the first part of the message is received, we register an edge on the IC-pin and the TOV is stored in a variable associated with the sensor port. Then, after some time, we receive the first byte (aa). We register the timestamp tot_1 , but as this is not the expected synchronization bytes of the uBlox protocol (B5), we do not increase the current package length, but instead flag a warning to the user. The same happens for the next byte we receive (also aa). When we receive the third byte (B5), it matches what we have configured the system for. We store the timestamp in tot_1 , and increase the current package length to 1. When we receive the next byte (62), the timestamp is stored in another value (tot_2) and as the synchronization bytes matches both in value and in length, we register that we (most likely) have achieved synchronization, increase the package length to 2 and store the value tot_1 as TOT. We also copy the latest TOV to the sensor message. To find the length, we have to have a sensor message size at least 4 (length offset) $+ 2$ (size of length) $= 6$ bytes. As we have found the TOT, we no

longer need to record timestamps for the bytes.

The next few bytes – 05, 01 and 02 – we store in the current buffer, but when the byte after that arrives (00) we have a buffer size of 6, and we can find the expected length of the sensor message. We read the last two bytes in the buffer as a little endian integer, and we find out that we have a total expected package size of $2 + 8 = 10$ bytes. We continue to read until the current package size is 10, and when we receive the last (61), we mark the sensor message as completed, we attach the TOA, and switch to another write buffer to be ready for the next sensor message.

3.3.4 Sensor Timing Hardware Output

When a full package has been read by the sensor handler and packaged in the envelope format, the data is transferred to the data output module. This module needs to handle both small packages at a very high rate, typically IMUs with up to thousands of measurements per second, and large packages with a lower rate, for example raw data GNSS messages at ten measurements per second. Furthermore, we want the timing system to utilize direct memory access (DMA) features of the processor.

One of the most used and available protocols on the market today is the USB-protocol, which in addition to providing high transfer speeds and having the option to deliver power to attached devices, is available on almost all modern computers – both desktop computers and single-board computers such as the ODroid XU4. Through a single USB connection, it is possible to have several endpoints, which may have different functions which makes it flexible. Some microcontrollers, such as the PIC32 MZ, also allow USB-transfers using DMA, which makes them a CPU-efficient protocol. Some systems, however, have limitations on which memory addresses can be accessed when using USB through DMA.

To enable efficient data transfer, processors can use DMA for data transfers. The DMA module is responsible for copying a set of data from one part of the processors memory to either another part of the memory, or to an external interface such as the USB interface. Without DMA the CPU needs to explicitly copy one word at a time from one memory address to another, which is inefficient both because the CPU could do something more useful, and because it might not be fast enough to keep up with higher transfer rates. A downside with some DMA implementations is that the starting address must be aligned in memory. Figure 3.6a shows a visualization of 16 bytes of memory with four queued messages, shown in four different colors. Now we imagine that these messages are transferred one by one using DMA with a 4-byte alignment requirement. First the green package is transmit-

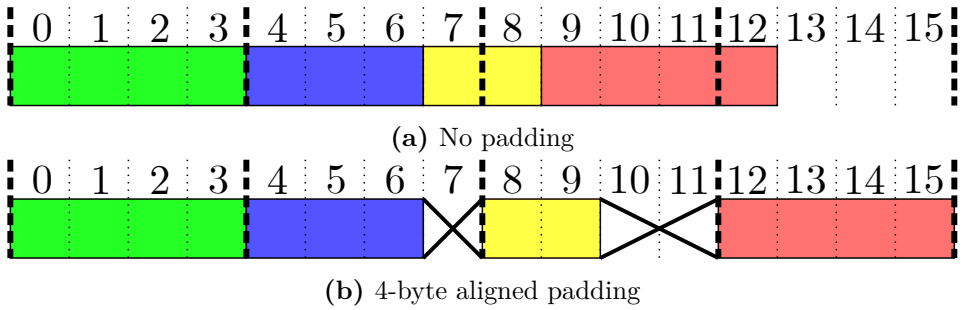


Figure 3.6: Four queued packages with and without padding. The crosses represent don't-care data, which may be zeroed out.

ted: the index is 0, the four bytes are written as expected, and the index is increased to 4. Then the blue package is transmitted: index is 4, three bytes are transmitted, and the index is increased to 7. Now, the yellow package is transmitted, but since the index is 7, it is no longer on the boundary and the DMA rounds the index down to the closest alignment and two bytes from there, resulting in that bytes 4 and 5, instead of 7 and 8 are transferred. Then the index is increased with 2 from 7 to 9. The next package (red) is 4 bytes long, but the index is still not aligned. Hence, the DMA copies 4 bytes from index 8 instead of from index 9, resulting in another erroneous transfer.

A proposed implementation that efficiently handles both messages of varying size and rate, and limitations in accessible memory addresses is given in Listing 3.1. The queueing implementation assumes the following interface with the processor's USB-interface: a USB-transfer is initiated with a function named `usb_write_dat` that takes a pointer to the buffer to be written and the number of bytes to write from that buffer. Only one USB-write can be active at a time. When the transfer is finished, the USB-subsystem calls a function named `usb_write_finished` with a parameter containing the number of bytes that were written by the USB-interface. The number of bytes are less-than or equal to the requested number of bytes, but it is assumed that this number is always aligned to the specific interval specified by the DMA module of the processor. Furthermore, two functions for disabling and enabling interrupts on the processor, called `usb_interrupt_disable` and `usb_interrupt_enable` are assumed available. The `memcpy_mod` function copies data from a regular array to a ring buffer and is defined in Listing 3.2 in Appendix 3.A.

To address the problem due to DMA-alignment, code in Listing 3.1 automatically aligns each queued package to a 4-byte offset by padding the

end of each package with up to 3 dummy-bytes, referred to as the **extra** variable in the code and the **PAD**-bytes in Figure 3.5. This allows any number of packages to be transmitted using DMA, allowing the system to fill the DMA-buffer if enough data is queued, without introducing issues with alignment, as every package is ensured to end at an aligned index. This approach is visualized in Figure 3.6b.

The `write_scheduled` variable does not need to be protected by the critical section, as it does not alter the behavior of the system, even though it is shared between the functions. If the variable is false when the function is called, no write events are in progress and thus the system cannot be interrupted by a finished write. The only exception to this is after a write has been scheduled at the very end of the function, after which the variable is no longer used. If the variable is true when entering the critical section, a write event should not be scheduled by the `usb_queue_package` function, and the `write_scheduled` remains unchanged.

The code provided in the listing is thread-safe under the following two assumptions: the `usb_queue_package` function is only called from the main loop, and the `usb_write_finished` is only called by the USB module when it has finished writing.

Listing 3.1: Sensor message queueing.

```
#define BUFFER_SIZE (1024 * 4)
uint8_t usb_write_buffer[BUFFER_SIZE] __ALIGNED__;
uint16_t queue_size;
uint16_t back_ix;
uint16_t front_ix;
bool usb_write_is_scheduled;

void usb_schedule_write() {
    // Make sure that we don't use variables we haven't checked
    .
    size_t _queue_size = queue_size;
    size_t _back_ix = back_ix;

    uint16_t send_size;
    // Verify that we do not write outside the queue
    size_t last_byte_pos = _back_ix + _queue_size;
    if (last_byte_pos > BUFFER_SIZE) {
        send_size = BUFFER_SIZE - _back_ix;
    } else {
        send_size = _queue_size;
    }

    usb_write_data(&usb_write_buffer[_back_ix], send_size);
```

```

}

void usb_write_finished(size_t datalen) {
    // Remove the written data from the queue
    queue_size -= datalen;
    back_ix = (back_ix + datalen) % BUFFER_SIZE;

    // Check if there is more data to transmit
    if (queue_size > 0) {
        usb_schedule_write();
    } else {
        usb_write_is_scheduled = false;
    }
}

void usb_queue_package(uint8_t *data, uint16_t datalen) {
    // Calculate padding bytes and the total size
    uint8_t extra = (4 - (datalen % 4)) % 4;
    uint16_t len_extra = datalen + extra;

    if (queue_size + len_extra >= BUFFER_SIZE) {
        return; // Prevent queue overflow - skip messages if
                // queue is full
    }

    // Copy the data to the queue
    front_ix = memcpy_mod(usb_write_buffer, data, len_extra,
                          front_ix, BUFFER_SIZE);

    // Critical section
    usb_interrupt_disable();
    queue_size += len_extra;
    bool schedule_write = !usb_write_is_scheduled;
    usb_interrupt_enable();

    // Only schedule write if needed
    if (schedule_write) {
        usb_write_is_scheduled = true;
        usb_schedule_write();
    }
}

```

As the `queue_size` variable is changed both in the `usb_queue_package` and `usb_write_finished` functions, it needs to be protected in the critical section of `usb_queue_package`. A temporary variable, `write_should_be_scheduled`, is also created in this section. This variable indicates if the `usb_queue_package` should schedule the next write event, or if a write event

is already in progress and the next write event should be scheduled in USB callback routine.

Microprocessor Considerations

When implementing this method on a microprocessor, special considerations to the time spent in ISRs must be taken. Although the processor primarily runs in one main loop, this loop can be temporarily paused when handling ISRs. When an interrupt event occurs, the microprocessor sets a flag that the interrupt event has occurred and then launches an ISR and clears the flag. If multiple interrupt events happen simultaneously the associated flags are set and ISRs are executed in order of priority. If two interrupt events of the same type happen while the microprocessor is handling a higher-priority interrupt event, the flag for the lower-priority interrupt event can only be set once, and one of the events are dropped. To prevent loss of data or other events, the time spent in ISRs should be minimized.

This method minimizes the time spent in the `usb_write_finished`-ISR, and the only time the data transfer could be further increased is when the end of the `queue` is reached, as only the data from the `back_ix` to the end of the queue is transferred. By choosing a sufficiently large `QUEUE_SIZE`, and thus reducing the frequency of the queue overflows, the loss of transfer capacity is negligible.

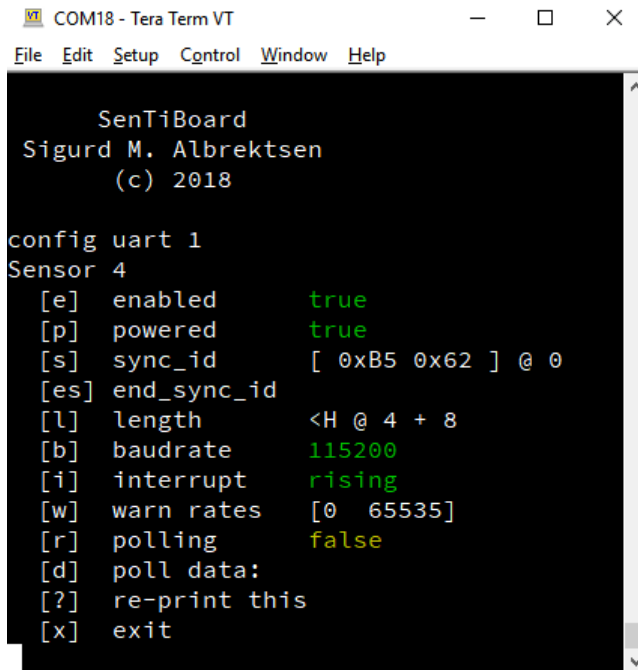
3.3.5 Sensor Configuration

As a developer of a sensor system is not necessarily an expert in hardware design, real-time systems or embedded system development, a hardware sensor timing system must be reconfigurable without impacting the overall system operation. To ensure that the system operation is intact, the developer needs to configure the hardware timing system without rewriting the firmware itself.

To configure a hardware sensor timing system, we suggest providing two distinct methods: using a USB command-line interface (CLI), and using configuration files on an SD-card or other removable storage. The main benefit of a USB command-line interface method is that it does not require any additional software except a standard VT102 terminal emulator such as Minicom[95] or PuTTY[106] – everything else is contained on the timing system itself. This method is ideal if changes must be made in the field or when configuring a new sensor for the first time, as the changes happen immediately. The configuration can then be written to the microcontroller's non-volatile memory, so the configuration is kept even if the

system is restarted. A screenshot of a command-line version of a sensor configuration module is shown in Figure 3.7.

The second method of configuring a hardware sensor timing system is to use configuration files placed on an SD-card or similar storage. This is an efficient method to use when identical sensors, or at least sensors with identical interfaces, are used in multiple sensor suites. To allow reuse of a configuration, the configuration for a specific sensor can be stored in a distinct file. This configuration can then be referenced to in the main configuration file, even several times if multiple identical sensors are connected simultaneously. An example of textual configuration files are shown in Figure 3.8. Reuse of sensor configuration files decreases system integration time, reduces the chance of errors as fewer files need to be changed if a sensor's configuration has changed, and follows the "Don't Repeat Yourself" philosophy common in computer science [147].



```

COM18 - Tera Term VT
File Edit Setup Control Window Help

SenTiBoard
Sigurd M. Albrektsen
(c) 2018

config uart 1
Sensor 4
[e] enabled      true
[p] powered      true
[s] sync_id      [ 0xB5 0x62 ] @ 0
[es] end_sync_id
[l] length       <H @ 4 + 8
[b] baudrate     115200
[i] interrupt     rising
[w] warn rates    [0 65535]
[r] polling       false
[d] poll data:
[?] re-print this
[x] exit

```

Figure 3.7: An example of a CLI for sensor port configuration.

```
sensors:
  ublox:
    powered: true
    sync_id: [0xB5, 0x62]
    length_size: 2
    length_placement: 4
    length_extra_bytes: 8
    baudrate: 460800
    interrupt: rising

  stim300:
    powered: true
    sync_id: [0x93]
    length_fixed: 38
    baudrate: 921600
    interrupt: falling
```

Sensor specification

```
configuration:
  uart1:
    sensor: ublox
    # Override default
    # baudrate value:
    baudrate: 115200

  uart2:
    sensor: ublox

  rs422:
    sensor: stim300
```

Port specification

Figure 3.8: An example of text files for sensor port configuration, in YAML.

3.4 SenTiBoard

To verify the proposed design, we have implemented the SenTiBoard, which is a reconfigurable sensor timing system that uses a PIC32MZ microprocessor running at 200 MHz, with a 32-bit timing counter running at 100 MHz to which the sensor readings are referenced. Communication with the SenTiBoard from an embedded or desktop computer is done through a double-endpointed High-Speed USB 2.0 interface; the first endpoint is used for configuration and debugging, and the second endpoint is used for sensor data transfer. If needed, for example if using an onboard computer without USB, sensor output can be redirected through one of the sensor ports. To record accurate timestamps from sensors, the SenTiBoard has a total of 9 IC-pins—allowing the system to simultaneously capture the TOV triggers from 9 sensors simultaneously. In addition, each sensor port can be triggered using the output compare (OC) functionality of the microcontroller. The physical dimensions of the board are 60 mm × 50 mm, and to simplify integration with the ODroid XU4, the removable adaptor legs shown in Figure 3.1a are attached to the board, with mounting holes that align with those of the ODroid. An overview of the SenTiBoard’s hardware is given in Figure 3.9.

By making the system reconfigurable, the firmware itself can be kept

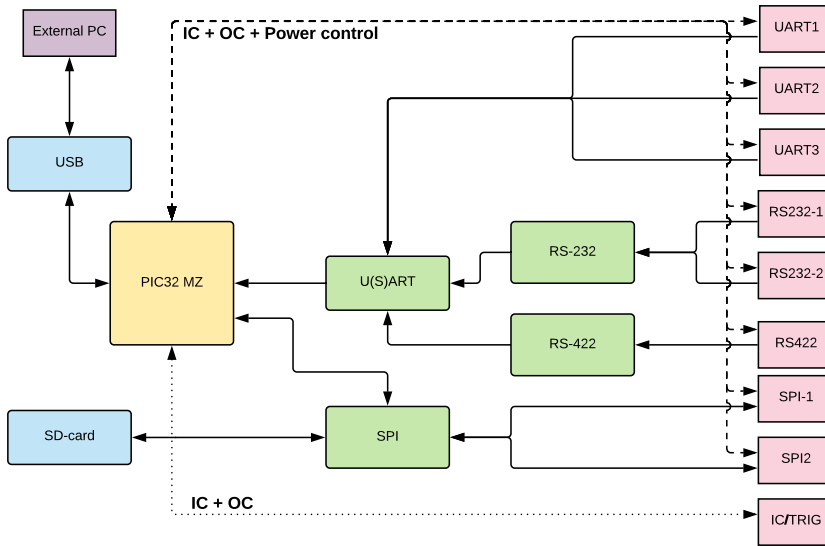


Figure 3.9: Overview of the hardware of the SenTiBoard. Each of the sensor ports (pink) have a IC-input pin, a OC-output pin and a power control pin in addition to pins for the associated protocol. The pink IC/Trig port only contains an IC-input pin and an OC-output pin.

more stable, and creating specialized versions for every sensor combination can be avoided. This allows the firmware to be reused and maintained by experts that know the system well, while still being adaptable to the sensor-suites by non-experts, which again leads to fewer errors as shown by Mohagheghi et al.[97]: “Our results showed that reused components have lower defect-density than nonreused ones (almost 50% less)”.

For sensor communication, the SenTiBoard supports several protocols. By default, three ports with U(S)ART support, two ports with RS-232 support, one port with RS-422 support, and two SPI ports are implemented. Additionally, the three UART ports can be configured to use the I²C, also known as TWI, protocol, and other protocols can also be interfaced with the system. The SenTiBoard can also generate pulses that act as triggers for sensors. This is useful for example for triggering when a camera should record a photo, or to sample multiple IMUs simultaneously.

Power control is another feature of the SenTiBoard. Each sensor port has a controllable power output pin on either 3.3 V (for the SPI-ports), or 5 V (for the other ports), which are independently controlled by the microcontroller. This allows the SenTiBoard to turn on and off sensors as needed; for example, if a sensor is misbehaving and needs to be restarted, or

in an emergency where the sensor needs to be powered off. There are also a set of error-LEDs, one per sensor port in addition to one for the whole system, that are used to indicate errors on the corresponding sensor and general errors.

The SenTiBoard has been previously used with a variety of sensor configurations in several experiments on UAVs, unmanned surface vehicles (USVs), and manned aircraft as discussed in Chapter 2.

3.4.1 SenTiUtils

To support system integration with the SenTiBoard and the SenTiStack (see Section 3.4.2), a utilities software package named SenTiUtils has been created. This package consists of documentation and software that interacts with the SenTiBoard, and software that converts the binary sensor data into higher level formats preferred by end-users, such as MATLAB's .mat-format and NumPy for Python's .npy files. The SenTiUtils package is split into five modules: Documentation, Testing, Logging, Parsing, Supervisor

Documentation

The documentation module contains details about the SenTiBoard hardware, the connectors used, how to configure the system, and the general structure of the intended usage of the SenTiBoard and the SenTiUtils package itself. This module also contains 3D-models of the board and connectors, sensor port specifications and mounting information. In addition, recommended configurations and tips for integrating commonly used sensors are provided.

Testing

The testing module contains software for testing basic interaction between the SenTiBoard and the host computer. There are two scripts in this module; `sentiboard_rate.py` and `senti_package_print.py`. The `sentiboard_rate.py` script reads all sensor packages from the SenTiBoard and prints a list of the rate of each of the incoming packages every 0.5 s. This gives the user a quick overview to see if the packages are received as expected. The `senti_package_print.py` script can be used when integrating a new sensor with the SenTiBoard. This prints the bytes contained in each sensor message in a human readable text format which again can be compared and verified with the expected output from the sensor described in the manual or datasheet provided by the sensor's manufacturer. Note that this module

has nothing to do with testing the firmware of the SenTiBoard, but rather testing of the sensor configuration and communication.

Logging

The logging module consists of a logging system which is recommended to use with the SenTiBoard. The logging system serves five main functions. Firstly, it creates a folder structure that ensures that logs are not overwritten when the system is restarted, by separating data in to different flights when it starts. Secondly, it separates messages from each sensor into distributed streams and distinct files. The number of sensor messages received by each sensor is printed by the logging system. This allows a user to supervise that the SenTiBoard and the logging system operates as intended by checking the log messages and the file system of each sensor and allows the user to only parse data from the sensors needed for their application. Thirdly, the logging system creates a new logging-folder at specified intervals, set to 15 minutes by default. The intention behind this time-splitting is to limit the impact of a file becoming corrupted which might happen if the onboard computer is powered off abruptly. Fourthly, the logging system provides an interface that automatically starts both the logging system itself and other necessary software for the sensor system when the onboard computer boots. Examples of such software are imaging logging software and logging of sensors not connected to SenTiBoards. Finally, the logging system injects a timestamp from the onboard computer into the stream of sensor data received by the onboard computer from the system. This allows the sensor packages on the SenTiBoard to be synchronized with messages not connected to either the SenTiBoard or a GNSS-receiver, for example when using cameras without external triggering pins.

Parsing

The parsing module contains code responsible for interpreting the binary stream or stored files from the sensors, wrapped by the SenTiBoard, to data processable by sensor fusion algorithms and other applications. There are two implementations of this system, one written in Python and one written in C++. The Python implementation is primarily used for converting to MATLAB and Python NumPy formats, while the C++ implementation is intended for use for relaying messages to other systems such as ROS[107], DUNE[32] and the SenTiBoard supervisor system. To be extendible to support a variety of sensors, the parsing module consists of two parts: a general package reader that extracts the timing and metadata from the SenTiBoard

envelope, and an interface for the specialized parses which is specific for each sensor type. The sensor data is then transmitted in a *Message* structure, which is specific to each sensor. To streamline integration of sensors of the same type, generic message types such as *ImuMessage* or *GnssMessage* can be used to handle IMUs and GNSS receivers from different manufacturers without having to change the external code. These message types can be extended as needed.

Supervisor

The supervisor module consists of a graphical visualization tool for showing real-time data as it is received from the SenTiBoard. This tool does not have any large framework dependencies, such as ROS, so it is valuable when using the SenTiBoard without such frameworks. The tool receives parsed sensor messages and visualizes the data in one of three main views. The first view shows a map with both the current and historical position data from the GNSS-receiver, the second view either shows a plot of the acceleration, gyroscope and magnetometer readings from the IMU(s) and magnetometer(s). The third view shows current camera images and is only active if a camera is connected. If a sensor fusion algorithm, such as an extended Kalman filter (EKF) is used, the first and second view can show the estimated position, the attitude and the heading instead of the measurement data.

3.4.2 SenTiStack

When developing new sensor fusion algorithms, having access to a high-quality reference is of great value. By comparing the results to a reference, an indicator on the performance of the newly developed algorithm can be obtained. The SenTiStack is a high-accuracy navigation payload that is designed to work with the SenTiBoard, and it serves two functions: primarily it provides high quality sensors and a high-accuracy navigation solution, but it is also a fully working and maintained example on how to configure and use the SenTiBoard. In addition to providing a solution, additional sensors can be added at will, except for the ports the sensors for the SenTiStack occupy. Sensors that have integrated with the SenTiStack so far are autopilots, RGB, IR and hyperspectral cameras, air data/pressure sensors, phased array radio navigation equipment and specialized sensors created at our university, but other system can be integrated as well. A benefit with using the SenTiStack as a reference system when developing algorithms that in addition to provide a navigation solution, also provides the sensor data the

estimates are based on, so that a new algorithm can be tested with identical sensor input.

Although the intended use is for UAVs, the SenTiStack has been used on other platforms such as ground rovers, unmanned surface vehicles (USVs) and manned aircraft. The SenTiStack consists of a SenTiBoard, an ODroid XU4 single-board computer, one or more uBlox M8T GNSS receivers and an IMU. Several different IMUs have been used; the Sensoror STIM 300[117], the Analog Devices ADIS16490[10], the Analog Devices ADIS16488A[9], and a combination of multiple IMUs simultaneously. In addition, a magnetometer connected to the UAV's autopilot has been used. An overview of the hardware of the SenTiStack is given in Figure 3.10.

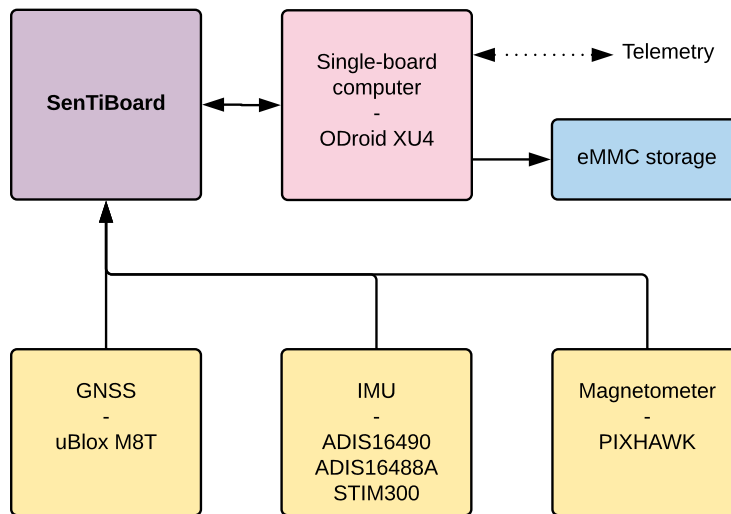


Figure 3.10: Overview of the components of the SenTiStack. In each of the sensor boxes (yellow), one or more of the options can be chosen, depending on the specific mission requirements. The telemetry link is optional, but can provide useful data about the system status, current sensor measurements, and provide data for RTK-GNSS calculations.

As the SenTiStack is used with a specific set of sensors, additional support for this system can be included. This includes pre-made parsers and integration with middleware systems such as ROS and DUNE. Furthermore, we are currently implementing a state estimator, onboard the SenTiBoard, and this will primarily support the sensors on the SenTiStack. By recording multi-GNSS pseudorange, Doppler, carrier phase, phase lock and signal

quality information from the onboard GNSS receiver and recording corresponding data at a local base station, a highly accurate RTK-solution can be calculated using the open source solution RTKLIB[130]. By using the RTK-GNSS solution to aid IMU and magnetometer measurements in a non-linear observer, or an EKF, PVA estimates can be obtained [52]. Thus, the SenTiStack is developed to be a complete high-accuracy navigation and timing system with tactical grade sensors, a hardware sensor timing board, and sensor fusion algorithms.

3.4.3 Verification

The operation of the implemented hardware sensor timing system was verified in two ways. First, the queueing method's maximal transfer rate was tested by transmitting dummy sensor messages with a data size and rate distribution typically seen in a sensor configuration. Then the clock accuracy was tested with a UAV experiment, where a GNSS receiver was used to provide a TOV and timing solution.

Transfer Rate Verification

To verify that the SenTiBoard's throughput using the proposed queueing strategy is sufficient, a performance benchmark was performed. In this benchmark a dummy sensor suite is implemented in the firmware. This dummy-suite only sends mocked data packages, but the size and rate of the sensor messages are based on real sensors. The following imagined sensors with the specified messages are connected: a uBlox M8T GNSS receiver transmitting a navigation solution at 1 Hz and raw measurements at 10 Hz with sizes 100 bytes and 648 bytes respectively (based on UBX-NAV-PVT and UBX-RXM-RAWX with 20 satellite readings (pp. 307, 337 [136])); a Sensoror STIM300[117] with rate, acceleration, inclination and temperature, transmitting 58 bytes at 2500 Hz; and two ADIS16490 transmitting 32 bytes (sync-id, temperature, delta-angles, delta-velocities and status) at 5000 Hz. The values 2500 Hz and 5000 Hz were chosen for convenience because they are the closest to a 5000 Hz counter already implemented on the SenTiBoard, and these values are sufficiently close to the real sensor values of 2000 Hz and 4250 Hz for testing purposes. In addition to the bytes from each sensor message, the 22–25 bytes from the SenTiBoard envelope are added to each package, depending on the number of *extra* bytes.

With the test-configuration given above, the SenTiBoard could transfer all the packages as required. To test the maximal transfer speed of the SenTiBoard, we gradually increased the rate of the timer with until it

reached a factor of 10, and the system still worked as expected, with very minor variances due to startup and shutdown timings. At this rate a total of 100,000 ADIS messages, 25,000 STIM messages and 110 uBlox messages are handled every second, and the total transfer speed is 7.66 MiB/s. When we further increased the speedup factor beyond 10.5, the system started to drop a significant number of packages, and we can see the data transfer does not increase further. See Table 3.3 and Figure 3.11 for details. The received messages are likely to be slightly different from the expected messages as some time is used to start and stop the process.

Speed factor	Received Messages	Expected Messages	Transfer Rate	Expected Rate
1	125,178	125,110	0.77 MiB/s	0.77 MiB/s
2	250,278	250,220	1.53 MiB/s	1.53 MiB/s
5	625,430	625,550	3.83 MiB/s	3.83 MiB/s
8	1,000,597	1,000,880	6.13 MiB/s	6.13 MiB/s
10	1,249,119	1,251,100	7.66 MiB/s	7.67 MiB/s
10.5	1,309,422	1,313,655	8.03 MiB/s	8.05 MiB/s
11	1,344,635	1,376,210	8.19 MiB/s	8.44 MiB/s
11.5	1,340,067	1,438,765	8.19 MiB/s	8.82 MiB/s

Table 3.3: Average transfer speed and number of messages received during a 10 second period. Each value is the average of 5 consecutive recordings.

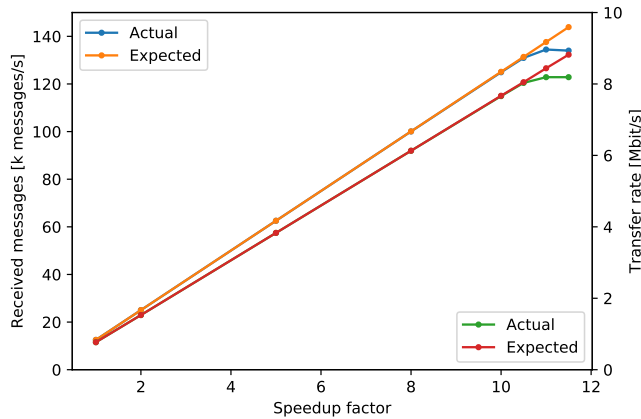


Figure 3.11: Visualization of Table 3.3.

Accuracy Analysis

To analyze the accuracy of the SenTiBoard's timestamping an experiment was carried out, where a flight with the SenTiStack onboard a Skywalker X8 fixed-wing UAV, was performed. The uBlox M8T GNSS receiver was set to output a position velocity time (PVT) solution and corresponding TOV signal with a rate of 1 Hz, and reported an accuracy of 3.35 ns RMS during the flight. To calculate the relative accuracy, the difference between each TOV message was calculated. An RMS difference of 1.90 μ s to the expected 1 s output from the GNSS receiver was found, and the result from the flight is shown in Figure 3.12. This inaccuracy is likely due to a drift in the oscillator and can be reduced, either by choosing a more stable primary oscillator, by attaching a stable secondary oscillator, or by increasing the output rate of the GNSS receivers' TOV signal, assuming that the drift of the GNSS receiver is negligible. In addition, one may compensate for clock drift using a mathematical model for the drift due to variations of the temperature[26].

Without using hardware synchronization to record a sensor's TOV, the most common way of timestamping messages is to record the time when the first byte has been transferred (TOT). This method is inaccurate as it depends on several factors, such as internal processing time, the number of messages transferred before the desired message is transmitted, the size of these messages, and the transfer rate from the sensor. The RMS difference of the first received byte of the GNSS PVT message to the expected 1 s rate was 53.56 ms and the TOV and TOT values are visualized in Figure 3.13.

To compare the accumulative effect of these inaccuracies over time, the following calculation was performed:

$$e(n) = t(n) - (\text{TOV}(0) + n) \quad (3.1)$$

where $t(n)$ is the TOV or TOT value n seconds into the dataset and $\text{TOV}(0)$ is the first TOV recording. The $e(n)$ of the TOV timestamp is shown in Figure 3.14, and the $e(n)$ of the TOT timestamp is shown in Figure 3.15, with the $e(n)$ of TOV for comparison. This error starts at the first measurement and then measure how much the proceeding measurements varies from the expected 1 s per measurement. As the measurement rate from the GNSS receiver is synchronized with a GNSS clock, it should be both precise and drift free. Hence, $e(n)$ provides an estimate of the drift of the SenTiBoard's clock over time.

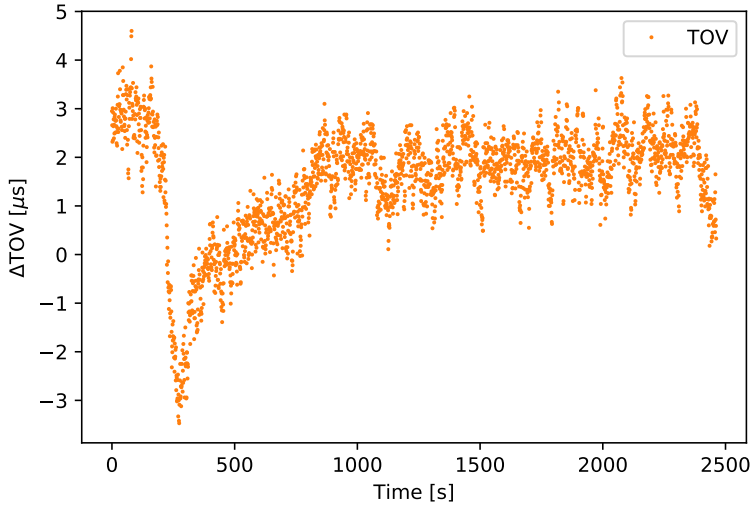


Figure 3.12: TOV accuracy, in microseconds.

Camera Payload

To illustrate the use of the SenTiBoard in a complete remote sensing system, another experiment was carried out using the SenTiStack. This time with a SenTiBoard, an ODroid XU4 onboard computer, an ADIS 16490 IMU, a STIM 300 IMU, two uBlox M8T GNSS receivers and a FLIR Tau2 longwave infrared thermal camera. The main goal of this experiment was to track two surface vehicles, a manned boat and the Maritime Robotics Telemetron USV, using an infrared camera onboard a Cruiser-Mini fixed-wing UAV. The reason for having two GNSS antennas in this payload is to be able to estimate the heading of the UAV. To achieve a high quality RTK GNSS solution, an additional GNSS receiver, in addition to the two GNSS receivers onboard the UAV, was placed at the base station. This base station is located at a fixed point and used as a reference to compensate for atmospheric disturbances.

To accurately synchronize the camera images with the UAV's position, the SenTiBoard was used to trigger an image capture circuit connected to the camera, using the OC functionality of the SenTiBoard. As the OC is connected to the same hardware timer running the microcontroller as the IC, the accuracy of the trigger signal is comparable to the results found in Section 3.4.3. The capture circuit attaches a timestamp, in milliseconds, to each camera image, and the timestamp can be set to zero using an external

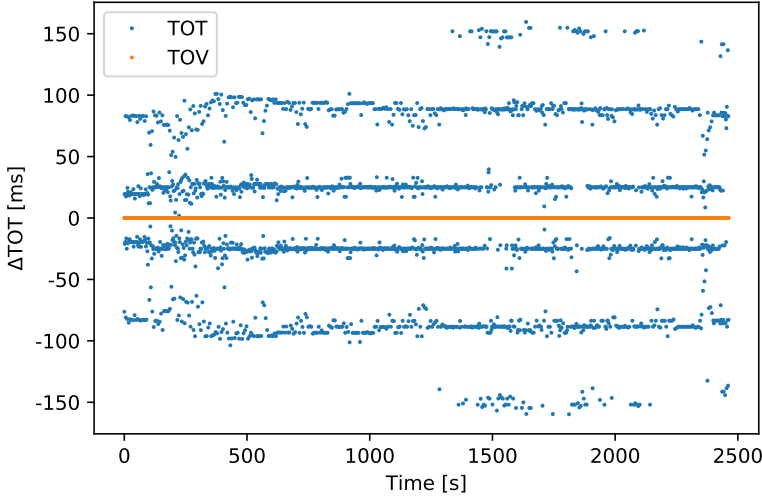


Figure 3.13: TOT accuracy vs. TOV accuracy, in milliseconds.

input pulse. To synchronize this input pulse with a GNSS receiver, the SentTiBoard is configured to emit an output pulse which is synchronized with a TOV pulse from one of the attached GNSS receivers, although at a higher framerate. This pulse is sent to the input trigger of the timing circuit board, which then makes the associated timestamp from the capture board synchronized with GNSS-time. A partial track of the position of the USVs and the UAV are visualized in Figure 3.16 and examples of infrared images captured during the mission are shown in Figure 3.17.

An analysis of the results from this experiment is given in [55] where the authors estimate an improvement in georeferencing errors from 10 m–15 m at an altitude of 100 m, to a mean of 9 m at an altitude of 350 m–400 m, when comparing to a solution with the same camera without GNSS synchronization. When accounting for the altitude difference, this corresponds to an accuracy improvement of a factor of 4, as improved accuracy is achieved at 4 times the altitude. This accuracy improvement is due to several factors, such as improved navigation and camera synchronization.

3.5 Conclusions

The high speed and fast dynamics of unmanned aerial vehicles make them valuable assets in many applications today, but these features also pose a

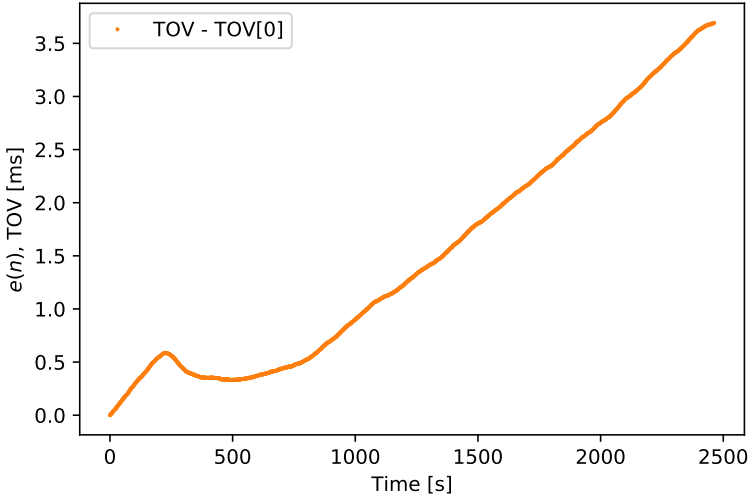


Figure 3.14: $e(n)$ for TOV recordings, measured in milliseconds.

challenge when it comes to sensor accuracy. As we want UAVs to perform more complex tasks, such as performing hyperspectral imaging, high accuracy is needed—not only for the sensor values, but also in the time domain. The rapid movements a UAV is capable of make an attached sensor to also move quickly, and thus cause large inaccuracies if the associated timestamp is imprecise.

This chapter describes required and desirable features of a reconfigurable hardware sensor timing system. An overall system design, a method for assigning timestamps to sensor messages using a data envelope, an approach for receiving sensor data and efficiently transmitting timestamped messages has been described. By making the solution reconfigurable, both the software and the hardware of the sensor timing system can be kept stable, which increases the reliability of the system, without compromising on sensor accuracy. Two methods are suggested for reconfiguration: one based on a command-line interface, and one based on files stored on a removable media such as an SD-card.

To verify the functionality of the suggested system a hardware implementation, the SenTiBoard, has been created. The system has a 32-bit, 100 MHz timer and uses interrupt capture to accurately record timestamps. It supports several commonly used protocols, power control of individual sensor ports, and triggering of external sensors such as cameras. In addition to

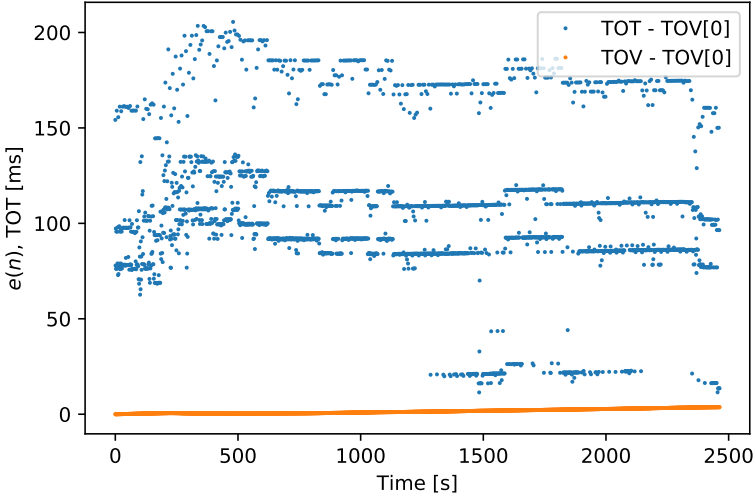


Figure 3.15: $e(n)$ for TOT vs. TOV, measured in milliseconds.

the system itself, a support system that consist of both software for testing and verification, and hardware solutions for providing a rapid navigation solution for UAVs, consisting of a RTK-GNSS capable GNSS receiver, one or two IMUs, and a magnetometer has been created.

The implementation has been tested and a maximal transfer speed of more than 8 MiB/s was achieved, allowing the board to transfer more than 130,000 packages per second. A relative accuracy of $1.90\mu\text{s}$ per second was measured, and an absolute accuracy of 2.75 ms after 2000 s when not compensating for the drift. An experiment with a hardware synchronized thermal camera was also performed and georeferencing errors were reduced with a factor of 4, compared to a software synchronization solution.

3.A Ring buffer memcopy

Support methods for copying to and from ring buffers are described in Listing 3.2. Note that the implementation on the SenTiBoard are optimized to avoid using the modulo operator which has a high computational cost.

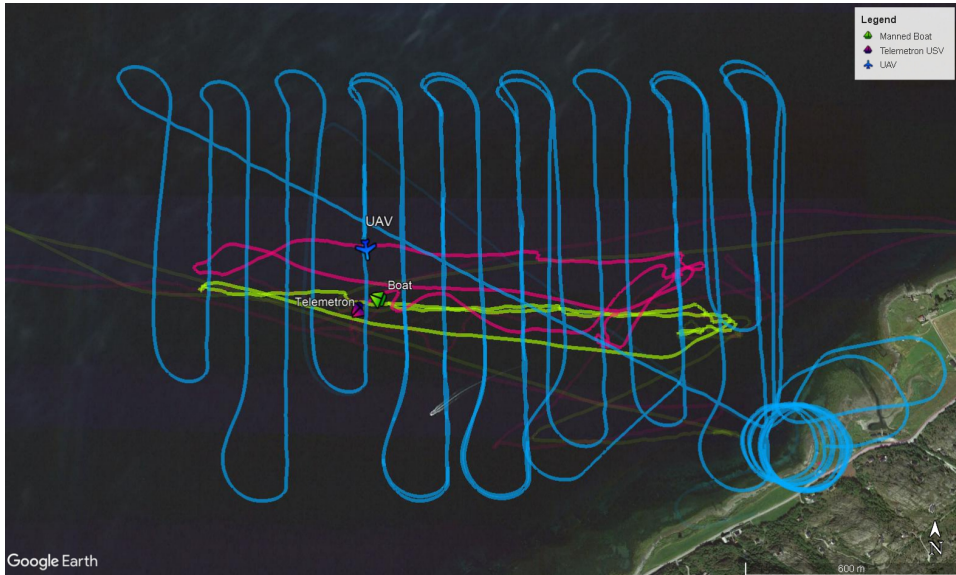


Figure 3.16: A partial GNSS track of the UAV's and the two surface vehicles' position.

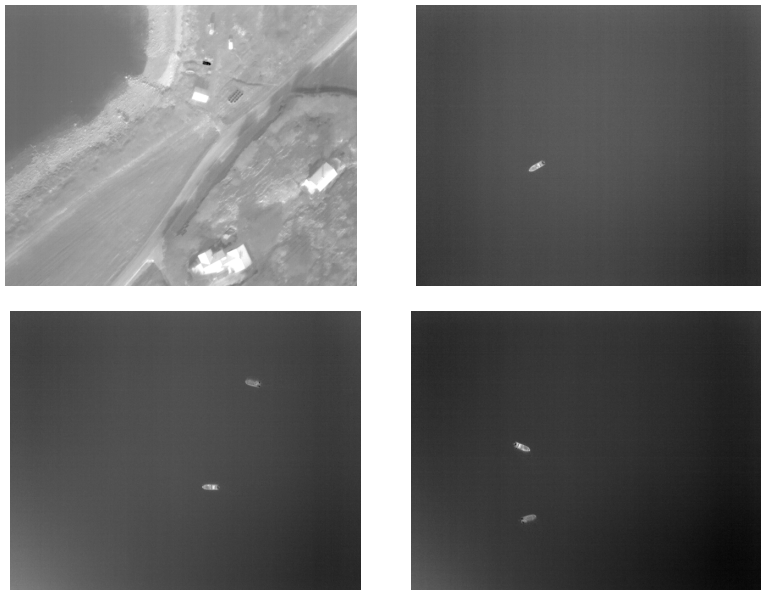


Figure 3.17: Infrared images captured during the mission.

Listing 3.2: Modulo memcpy

```
size_t memcpy_mod(char *dst, char *src, size_t datalen,
                  size_t front_ix, size_t queuelen) {
    size_t i;
    for (i = 0; i < datalen; i++) {
        dst[(front_ix + i) % queuelen] = src[i];
    }
    return (front_ix + datalen) % queuelen;
}

size_t memcpy_demod(char *dst, char *src, size_t datalen,
                   size_t front_ix, size_t queuelen) {
    size_t i;
    for (i = 0; i < datalen; i++) {
        dst[i] = src[(front_ix + i) % queuelen];
    }
    return (front_ix + datalen) % queuelen;
}
```

Part II

Navigation in GNSS-Denied
Environments

Inertial Optical Flow for Throw-and-Go Micro Air Vehicles

This chapter is based on S. Weiss, R. Brockers, S. M. Albrektsen, and L. Matthies. Inertial optical flow for throw-and-go micro air vehicles. In *2015 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 262–269, Waikoloa, Hawaii, USA, January 5–9 2015. doi: 10.1109/WACV.2015.42.

4.1 Introduction and related work

Small Unmanned Aerial Vehicles (SUAVs) have significantly gained importance in the past years. Industry is already using these vehicles on a regular basis for aerial mapping and inspection in the mining, and oil and gas sector, as well as for aerial photography, and search and rescue operations in fire fighting scenarios. For large area mapping and surveillance, mostly fixed wing platforms are used in open space where GPS availability is granted. However, for inspection tasks, underground mining operations, or search and rescue tasks in confined spaces an agile system with GPS independent autonomy is key.

Whereas the demand for high agility asks for naturally unstable platforms – and thus good state estimation and control algorithms – GPS independence in areas which are difficult to access require innovative solutions for state estimation with on-board sensing and processing. Multicopter systems (e.g. quadrotors) have a distinct advantage in their hovering capabilities and agility to counteract strong winds and other external disturbances. Navigating such a system is particularly challenging since there is no “hold”

function as compared to ground vehicles that can be entered as a safety regime (e.g. holding all actuators will only result in zero velocity after a crash). Furthermore, a misalignment to gravity results in an acceleration in one direction causing the position to drift quadratically in time. Similarly, simply integrating IMU accelerations for position hold will soon result in a crash due to noise and bias terms on the accelerometers. A minimal requirement for keeping a quadrotor airborne is a continuous estimate of the metric velocity¹ and the gravity aligned attitude. The SUAV may still drift in position and yaw (linear in time), however, given an obstacle free area or a protective case, the vehicle remains airborne.

There is a large body of work for indoor SUAV control using motion tracking systems [93, 83] and for outdoor operations using GPS signals [2, 148]. In contrast to these approaches that depend on external positioning information, this paper focuses on SUAV control in environments where an external tracking setup is infeasible (e.g. large outdoor areas) and where GPS signals may be corrupted or unavailable (e.g. in cities, caves etc.).

A popular approach is to control and navigate SUAVs based on (local) maps without the need of a motion capturing system or GPS. This is often done using known markers, pre-built maps or maps built on the fly using SLAM or (keyframe based) visual odometry (VO) techniques. Such approaches usually control the vehicle in its 6DoF pose (position and attitude). The drift in position and yaw is either very low or even eliminated by using known, fixed structures as reference. Sensors that are commonly used for map generation are laser scanners [13, 118] or cameras incorporating known markers [21, 146] or SLAM/VO techniques [18, 3, 144]. Since laser scanners are still too heavy and power hungry for small quadrotor systems, our work focuses on vision based approaches. Common to all approaches using a map for motion estimation is that the map can get corrupted or lost. In such a case recovery is difficult if not impossible and the vehicle is prone to crash.

In order to avoid the issue of a map loss, we follow the approach of not having any type of feature history except the feature matches between two consecutive images: i.e. optical flow (OF). A history free approach augments the algorithm robustness due to the independence on past readings. OF approaches that do not include 6DoF inertial measurements are presented in [56, 150]. While already showing the capabilities of OF for SUAV navigation, these approaches act on a reactive manner to keep the vehicle away from ground or from obstacles. Reactive control for position-keeping or trajectory-navigation of an unstable micro air vehicle is not sufficient, since the unavailability of metric information can result in instability.

More recent work includes 6DoF inertial reading and successfully estimates not only the metric velocity [80, 44] but also inertial biases, inter-sensor transformations, and a gravity aligned attitude of the SUAV [141] - the minimal requirements in order to keep a SUAV airborne.

Our work is a continuation of the work in [142] with the novelty that we 1) include a method for fast initialization, 2) develop a robust outlier rejection both for temporal miscalibration and for errors due to noise in the optical flow, and 3) proof robust state estimation and control in extreme situations such as throwing the SUAV in the air to deploy it.

Compared to commercial products, our approach does not require an active sensor (e.g. ultra sonic altimeter) nor GPS. This reduces the weight and power consumption of the overall sensor suite. More important, our approach does not require a gravity aligned nor completely flat ground plane for correct functioning. In fact, our approach gains on performance in inclined terrain as explained later. Furthermore, our approach is self-calibrating and is algorithm focused, rather than platform focused. That is, any platform that can mount a camera and an inertial sensor can achieve the performance we show in this paper. There is no need for sophisticated hardware nor (time) synchronization mechanisms.

The remainder of this chapter is organized as follows. In Section 4.2 we briefly describe the computation of the optical measurements that are obtained from optical flow – the arbitrarily scaled 3D camera velocity vector and the terrainplane parameters in the camera frame. These measurements are later fused with the inertial cues in a filter approach. In Section 4.3 we describe the effects of temporal miscalibration between the inertial sensor and camera on these measurements and discuss the robustness against non-flat terrain. Section 4.4 briefly describes our EKF framework showing the capability of estimating metric velocity, full attitude, inertial biases, sensor extrinsics, as well as the terrain inclination and metric terrain distance to the SUAV. We also discuss our fast initialization approach and the ability to render the system truly *throw-and-go*. In Section 4.5, we verify our approach in real world experiments where the SUAV is deployed by literally just throwing it in the air. Section 4.6 concludes the chapter.

4.2 Computation of optical measurements

In this section, we will briefly review how optical flow can be used to compute the camera body velocity and the current terrain plane parameters in the camera frame up to a unifying, but arbitrary scale Λ . Note that we only require two consecutive feature matches (i.e. optical flow) to compute the

velocity, and three for the plane parameters. We do not store any feature history nor local map. We use the continuous epipolar constraint and the additional information of the angular velocities measured by the Inertial Measurement Unit (IMU).

4.2.1 Camera velocity and terrain-plane parameter computation

The motion (i.e. flow) of a feature in the camera frame is given by

$$\dot{X}(t) = [\vec{\omega}(t)]X(t) + \vec{V}(t) \quad (4.1)$$

Where X is the 3D feature position, \vec{V} the camera velocity, and $[\vec{\omega}(t)]$ the skew symmetric matrix of the camera angular velocities. We can define a scale factor L_f which scales a unit feature direction vector \vec{x} to a metric 3D point: $X = L_f \vec{x}$. Similarly, L_v scales the unit velocity direction vector \vec{v} to the metric 3D velocity vector $\vec{V} = L_v \vec{v}$. Inserting this in (4.1) and eliminating the scale factors by multiplying with $[\vec{\omega}(t)]\vec{x}$ yields the continuous epipolar constraint [84]:

$$\dot{\vec{x}}^T [\vec{v}(t)]\vec{x} + \vec{x}^T [\vec{\omega}(t)] [\vec{v}(t)]\vec{x} = 0 \quad (4.2)$$

Un-rotating the feature direction vectors with the angular velocities from the IMU eliminates the second term in (4.2) and reduces the problem to

$$([\dot{\vec{x}}(t)]\vec{x})^T \vec{v} = 0 \quad (4.3)$$

This equation can be solved for \vec{v} using N features and SVD. Note, that the complexity of the SVD is only of dimension three (for the 3D velocity vector \vec{v}).

As suggested in [84] and from (4.1) with $\vec{\omega} = 0$ using the IMU, any (un-rotated) feature i used in (4.3) needs to fulfill its motion equation

$$\dot{\lambda}_i(t)\vec{x}_i(t) + \lambda(t)\dot{\vec{x}}_i(t) = \eta\vec{v}(t) \quad (4.4)$$

The scale factor λ_i is different for every unit length feature direction vector \vec{x}_i to reflect the 3D structure of the terrain. When stacking all λ_i , $\dot{\lambda}_i$, and η into the vector

$$\vec{\lambda} = \begin{bmatrix} \lambda_1 & \dots & \lambda_n & \dot{\lambda}_1 & \dots & \dot{\lambda}_n & \eta \end{bmatrix} \quad (4.5)$$

(4.4) can be rephrased as the SVD problem

$$M(\vec{x}, \dot{\vec{x}}, \vec{v})\vec{\lambda} = 0 \quad (4.6)$$

The solution $\vec{\lambda}$ unifies all scale factors in a consistent way up to one common scale factor Λ . This is essentially a continuous triangulation of feature points since we can reconstruct the 3D scene and camera velocity up to common scale Λ with:

$$X_i = L_f \vec{x}_i = \Lambda \lambda_i \vec{x}_i \quad (4.7)$$

$$\vec{V} = L_v \vec{v} = \Lambda \eta \vec{v} \quad (4.8)$$

We note at this point, that the scene can be of any 3D structure and that the algorithm is not bound to planar terrain.

(4.7) represents a 3D point cloud in Euclidean space. We can compute this point cloud measuring OF and using (4.3) and (4.6) up to the common scale factor Λ . We define the term terrain-plane as the plane fitted to this point cloud by the regression

$$[\vec{n}_{tp}^T, d_{tp}][\lambda_i \vec{x}_i^T, 1]^T = 0 \quad (4.9)$$

with the plane normal vector \vec{n}_{tp} and the distance to the origin (i.e. camera center) d_{tp} .

To compute the visual measurements - i.e. the unscaled 3D velocity vector v , the terrain plane normal n_{tp} , and the unscaled distance d_{tp} - feature matches in two images are required and (4.3), (4.6), and (4.9) have to be solved. Using the simplification in [142], solving (4.3), (4.6), and (4.9) have low computational complexity, leaving the burden on the feature matching part. To speed up this process we use a vision based prediction step. We explicitly do not use the prediction from the state estimator described in Section 4.4 to keep the vision module independent and decoupled from the state estimator (and its maybe yet unconverged estimates). Instead, we use (4.4) and approximate the time derivative to predict the feature position \vec{x}_{i2} in the second image. We assume that the feature distance is much larger than its temporal change $\lambda \gg \dot{\lambda} \approx 0$, and thus $\lambda_{i1} \approx \lambda_{i2}$. This is particularly true for feature direction vectors \vec{x}_i perpendicular to the velocity vector \vec{v} (e.g. UAV with a down looking camera and moving in the xy-plane). There $\dot{\lambda} \equiv 0$. From (4.4) we can then infer \vec{x}_{i2} as follows

$$\frac{\lambda_{i2} \vec{x}_{i2} - \lambda_{i1} \vec{x}_{i1}}{dt} = \eta \vec{v} \quad (4.10)$$

$$\vec{x}_{i2} = dt \frac{\eta \vec{v}}{\lambda_i} + \vec{x}_{i1} \quad \text{with} \quad \lambda_{i1} = \lambda_{i2} \quad (4.11)$$

We can compute an initial value for $\eta \vec{v}$ using only a few high-quality features evenly spread in the image. With this initial value, we can predict

additional feature positions accurately within a very small search window. The decreased search window size for the majority of the features to be matched leads to a noticeable algorithmic speed-up of 24 %.

(4.9) requires the observed features to be roughly planar. In [142] the authors assume that the terrain is flat and it does not change quickly in the world frame such that they can use the plane parameters together with IMU readings to estimate relative yaw and distance between the SUAV and the plane. However, in practice bushes and trees in the field of view may present challenging situations to maintain this assumption. Furthermore, inaccurate time calibration between the IMU and camera can lead to wrong feature unrotation and thus failures in the algorithm. We extend the approach in [142] by extracting planar surfaces more reliably and by mitigating the influence of wrong time synchronization. The increased algorithm robustness enables the SUAV to be deployed and used quickly (e.g. by simply tossing it into the air) in real-world scenarios.

4.3 Temporal misalignment and non-flat terrain

Incorrect time synchronization between IMU and camera lead to a wrong un-rotation of the optical flow and feature direction vectors violating (4.3) since $\omega \neq 0$. This is different than noise and introduces a bias to the estimate. To solve (4.3) we compute the cross product between the feature direction vector and the optical flow vector for each feature. This yields a bundle of normal vectors which are all required to be perpendicular to the unknown velocity direction vector. Figure 4.1 illustrates this relation for a camera motion parallel and perpendicular to the scene surface. The normal vectors are expected to end on a circle on the unit sphere defining a plane which includes the camera center (yellow circle in Figure 4.1).

The difficulty is to find a robust approach for the plane regression on the normal vector bundle. A RANSAC approach is impractical for this problem because the abort criteria and other parameters are difficult to define. Figure 4.2 shows the same setup as in Figure 4.1 but representing real data. The yellow dots mark the normal vectors to find \vec{v} . Depending on the parameters, RANSAC is very likely to find a plane in a local minimum (marked as a red line in the figure). Hence, it is desirable to have a parameter-free approach which is adaptive to the vector distribution. We have chosen an iterative re-weighted least squares (IR-LSQ) approach which is parameter free, robust against noise, and against some amount of temporal misalignment.

Temporal misalignment introduces a fix pattern in the distribution of the

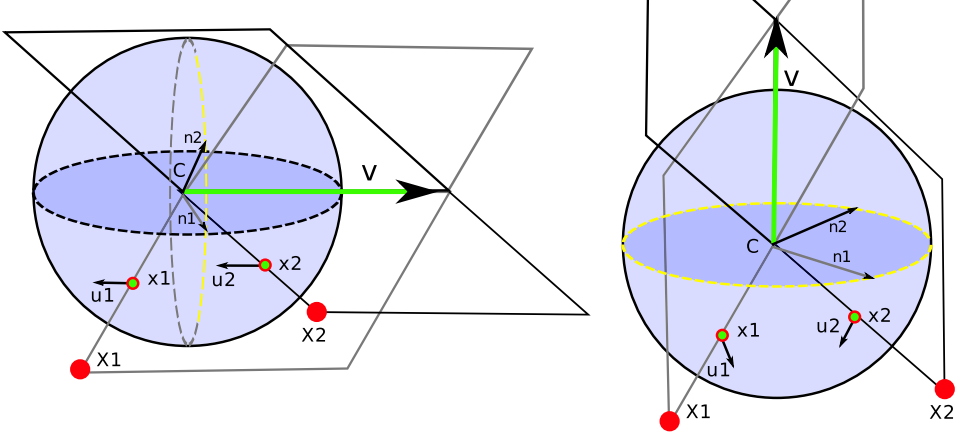


Figure 4.1: Schematic drawing of the camera C moving parallel (left) and perpendicular (right) to a scene. X_i denotes a 3D feature in the world with its direction vector x_i on the unit sphere and its optical flow vector $u_i = \dot{x}_i$. Each pair of x_i and u_i span a plane with normal vector $n_i = [u_i]x$. All n_i end in a circle on the unit sphere (yellow dashed line) and span a plane with the camera center C . The normal of this plane is the camera velocity direction vector v .

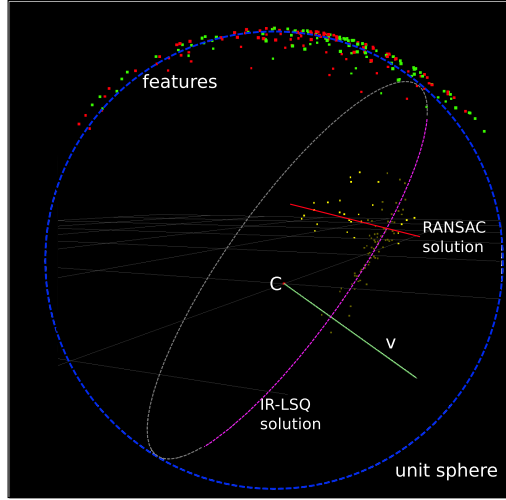


Figure 4.2: Real data showing the tracked feature direction vectors in frame 1 (red dots, \vec{x}) and frame 2 (green dots) to compute the optical flow $\dot{\vec{x}}$ and the normal vectors $[\dot{\vec{x}}]\vec{x}$ (yellow dots). A RANSAC approach is dependent on parameter setting and likely to result in a local minima (red line) whereas our iterative reweighted least squares (IR-LSQ) approach selects the correct inliers (dark yellow dots) and finds the right plane (magenta circle) and its normal vector (green line, \vec{v}).

normal vectors $[\dot{\vec{x}}]\vec{x}$. We measure the effects of temporal misalignment by an angular offset in the un-rotation step. This gives a performance overview independent of the current angular velocity of the vehicle. As an example, an angular offset of one degree translates in a temporal IMU-camera offset of 10ms if the vehicle currently turns at 100deg/s. Based on this angular offset, we measure the angular error of the computed velocity direction vector \vec{v} and terrain-plane normal \vec{n}_{tp} with respect to ground truth. Figure 4.3 depicts the performance upon different angular offsets in roll and yaw if the camera is moving parallel to the scene (a pitch offset is comparable to a roll offset). Whereas our algorithm can mitigate temporal misalignments resulting in an angular offset of up to 12deg in roll for both \vec{v} and \vec{n}_{tp} it can only mitigate about 6deg offset in yaw for \vec{v} and quickly deteriorates for \vec{n}_{tp} . Note that these effects switch for roll, pitch, and yaw according to the feature location and the relative camera motion.

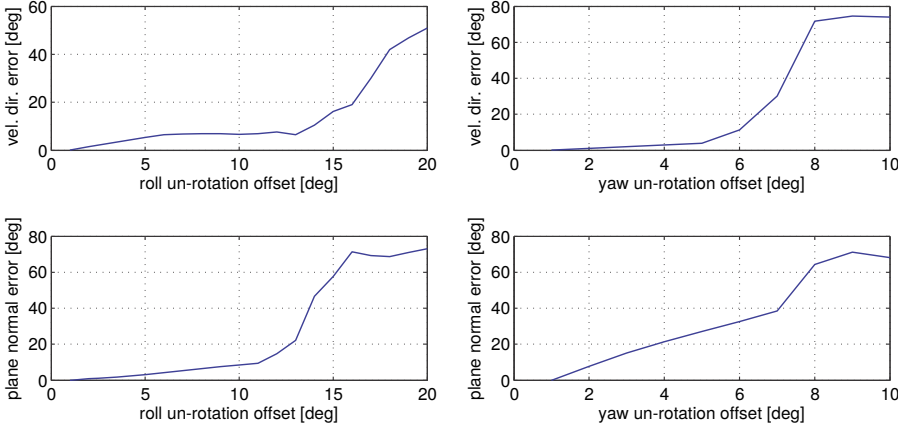


Figure 4.3: Computation error for \vec{v} and \vec{n}_{tp} with different angular offsets for un-rotation. These offsets reflect a time misalignment between IMU and camera. An offset of one degree is a time offset of 10ms if the vehicle executes motion of 100deg/s. The maximum angular velocity measured in controlled flight does not exceed 60deg/s in roll and pitch – an offset of one degree would translate into a time misalignment of 17ms. In yaw, the angular velocities in controlled flight are much smaller (cf. Fig. 8).

The difference in performance comes from the different effect an offset has in each axis relative to the current camera motion and feature location. Figure 4.4 depicts the different distortions of the normal vectors $[\dot{\vec{x}}]\vec{x}$ and the terrainplane. The wide spread of $[\dot{\vec{x}}]\vec{x}$ and the large distortion of the plane caused by a yaw offset makes it difficult for any regression algorithm.

In our experiments, the SUAV did not exceed 60deg/s in roll and pitch,

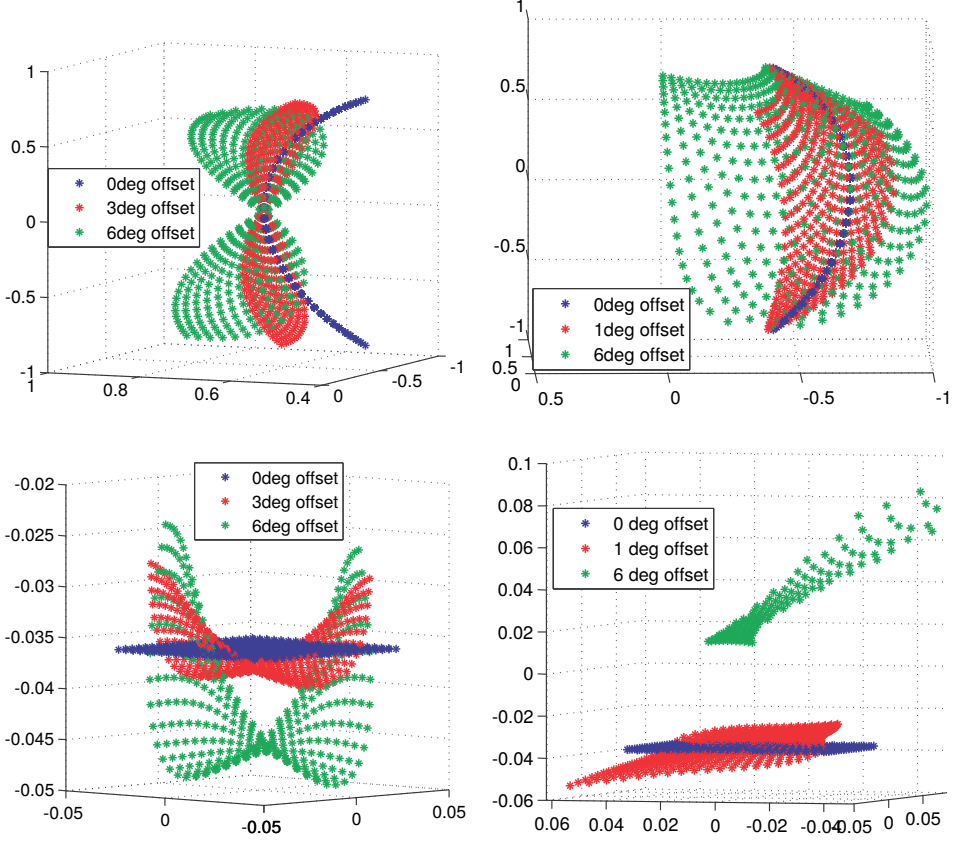


Figure 4.4: Distributions of \vec{v} and \vec{n}_{tp} for different angular offsets: top: $[\dot{\vec{x}}] \vec{x}$ for different roll (left) and yaw (right) offsets; bottom: plane triangulation for different roll (left) and yaw (right) offsets. Whereas an offset in roll and pitch is simpler to mitigate for our IR-LSQ algorithm, the wide spread caused by a yaw offset makes a correct regression difficult. Note the different scales in the axes of the plots.

and 11.5deg/s in yaw during controlled flight. From Figure 4.3 we see that our approach mitigates 12deg offset in roll and pitch well for both \vec{v} and \vec{n}_{tp} . This translates in a tolerance of 200ms for a time misalignment between IMU and camera. We note that with NTP-like software time synchronization methods, the misalignment can be kept below 2ms. 200ms time misalignment with 11.5deg/s in yaw would translate into an offset of 2.5deg which can be well mitigated (Figure 4.3). We will discuss in Section 4.5 how we can reliably handle abrupt angular motion over a short period.

While outliers due to noise are rejected robustly both in the calculation of \vec{v} and \vec{n}_{tp} due to the IR-LSQ approach, non-flat terrain for the terrain-plane regression to compute \vec{n}_{tp} needs a separate analysis. A terrain often has a dominant plane with several positive or negative objects on it. As the SUAV flies over these objects the parameters of the dominant plane should roughly be kept constant since this is the model we will be applying later when adding the IMU cues in a filter framework. Our parameter-free IR-LSQ approach adapts well to such situations converging quickly to the dominant plane of the terrain ignoring disturbances from positive or negative objects. In Figure 4.5 we show real data captured above non-flat ground. The live camera image shows the calculated optical flow on the ground and on the objects. Our approach computed correctly the ground plane normal vector \vec{n}_{tp} . Outdoors, bushes and trees are simpler to eliminate because of their lack of planar characteristics biasing the result.

4.4 Extended Kalman filter

So far, we extracted the unscaled 3D velocity vector v , the terrain plane normal n_{tp} , and the unscaled distance d_{tp} from the camera to the terrain plane from optical flow using (4.3), (4.6), and (4.9). (4.6) combines all scale factors in $\vec{\lambda}$ up to one unifying but arbitrary scale factor Λ . In order to estimate this factor reliably using IMU inputs, it should not arbitrarily change from one optical flow measurement to the other. We use the approach from [142] and normalize $\vec{\lambda}$ by d_{tp} such that ${}_n\vec{\lambda} = \frac{\vec{\lambda}}{d_{tp}}$. This way, Λ will only change if the distance to the scene changes. In fact this normalization links the value of Λ directly to the metric distance between the camera and the scene.

Sudden and large changes in distance to the scene as they occur when throwing the SUAV for fast deployment result in equivalent changes in Λ . In fact, since ${}_{\text{metric}}d_{tp} = \Lambda$ short-term integration of the accelerometer to determine the change in ${}_{\text{metric}}d_{tp}$ can directly be used to propagate Λ in an EKF prediction step [142]. This allows accurate state tracking even in

contains the IMU-centered SUAV position p_w^i , velocity v_w^i and attitude q_w^i with respect to the world frame. It also contains the IMU biases on gyroscopes b_ω and accelerometers b_a , the common visual scale factor Λ and the 6DoF transformation between the IMU and the camera in translation p_i^c and rotation q_i^c . Thus, we provide a self-calibrating and so-called power-on-and-go system.

The terrain plane is represented using three parameters: two for the unit normal vector (elevation α and azimuth β) and one for the distance of the plane to the origin. Without loss of generality, we can anchor the world frame in the terrain plane such that its distance to the origin and azimuth vanishes. A non-linear observability analysis reveals that α is observable and – if $\alpha \neq 0$ (i.e. if the plane is inclined with respect to gravity) – the full attitude including yaw with respect to the plane inclination of the SUAV becomes observable. In fact, the only unobservable states in χ are the two dimensions of the position vector parallel to the terrain plane.

While the uncertainty for the process noise is given by the IMU manufacturer and from signal analysis of the accelerometer and gyroscope, we need similar information for the visual updates. Since all visual measurements are computed from least squares methods, this is straight forward by using the residuals. This uncertainty information not only allows to apply correct measurement noise for an EKF update step but also to gate the measurements if they seem to be invalid according to a χ^2 test. If the test fails, the visual update is discarded and the EKF continues to integrate the IMU readings until a valid visual update is available. Since all visual updates are independent and only require two consecutive frames, this gating can directly be applied and used to bridge short term failures of the vision module without further specific considerations.

We use the approach in a similar way as described in [142] for the EKF system definition. For fast initialization we use a manually measured IMU-camera 6DoF transformation p_i^c , q_i^c for initialization, set the bias value $b_\omega = 0$, and $b_a = 0$, and assume a 1 m distance to the scene (typical setup when holding the SUAV at waist height). The initial attitude q_w^i is computed by using two constraints: First, the gravity measured by the accelerometer (we assume little to no motion for initialization) must align with the gravity in the world frame $g = R(q_i^w)a$. Second, the azimuth for the terrain plane normal vector in the world frame $n_{tp}^w = R(q_i^w)R(q_i^c)n_{tp}$ must vanish. This will additionally yield

$$v_w^i = R(q_i^w)(R(q_i^w)\Lambda_n \eta v - \lfloor \omega \rfloor p_i^c) \quad (4.15)$$

and the initialization of the metric scene distance is straight forward.

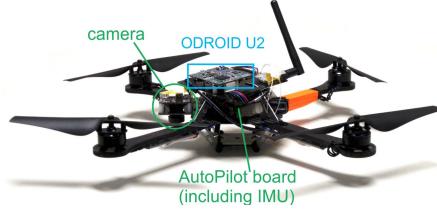


Figure 4.6: 500 g AscTec Hummingbird equipped with a 12 g Exynos 4412 (quad Cortex ARM A9, 1.7 GHz) processing board and a global shutter WVGA camera.

This initialization only needs one visual and one inertial measurement (i.e. two consecutive camera frames) and is sufficiently close to the real values to allow a very quick convergence.

4.5 Performance evaluation

We implemented our approach on-board a quadrotor, the 500 g AscTec Hummingbird, equipped with a 12 g Exynos 4412 processing board and a global shutter WVGA camera (Figure 4.6) using ROS for inter process communication. Even though our approach would run at more than 50 Hz on this platform, for the following experiments, we ran the camera at 30 Hz and the IMU at 1 kHz to provide a safety margin. A low level attitude control auto pilot board (60 MHz ARM7) hosts the IMU and performs the EKF prediction step for short term control. The camera is plugged to the Exynos where we perform the more complex image processing and EKF update step. We synchronize the auto pilot board with the Exynos by an NTP like protocol. Thus, camera and IMU are not hardware synchronized. The time jitter on the USB bus plus the inaccuracy of the NTP time synchronization is still well below our margins for IMU-camera time misalignment discussed in Section 4.3.

For rapid deployment, we literally throw the SUAV in the air just after switching it on. Stabilization after such a drastic maneuver with essentially no initialization, hardware time synchronization, or previous calibration is only possible with our improved and robust visual measurements, our normalization method for accurate scale tracking, and robust initialization approach.

For performance testing, we hold the SUAV in the hand, switch on the motors and then start the state estimation. No particular motion is required for initialization (as for example translation is needed to initialize map-based SLAM approaches). Since we use any two consecutive camera frames



Figure 4.7: Throwing a small rotorcraft. With the fail-safe state estimation using only two consecutive camera frames without any feature history or (local) map, we are able to literally toss the rotorcraft in the air for deployment.

to initialize the system states, we can immediately throw the vehicle in the air for deployment. Figure 4.7 depicts such a throw.

Fig. 8 to Fig. 12 depict the different states and IMU readings during the throw maneuver. From $t=32s$ to $t=33s$ we hold the vehicle as depicted in Fig. 7 in the lowest position. We throw the SUAV just after $t=33s$ where it enters a parabola flight due to gravity only controlling its attitude (we deliberately delay the position control to have some free flight). Just after $t=34s$ we start the position control and the vehicle stabilizes within the next 1.5s until about $t=35.5s$. During the next 5-7s the visual scale factor fully converges.

Fig. 8 depicts the gyroscope readings during the maneuver. Immediately after releasing the SUAV, the low-level attitude controller stabilizes the helicopter. The throw and this stabilization maneuver cause angular velocities up to 6rad/sec. This causes motion blur and errors in the vision module computing velocity and scene plane parameters. However, the IMU can bridge these short moments by integrating the inertial readings. Any time two consecutive images yield reliable optical flow readings, they can be used as an EKF update, if not, the visual input is discarded and the IMU simply keeps integrating. Errors in some frames do not cause issues since we do not have a feature history or a (local) map which could get corrupted. Erroneous frames are simply skipped without consequences in our approach.

Fig. 9 depicts the accelerometer readings. At the point of maximum

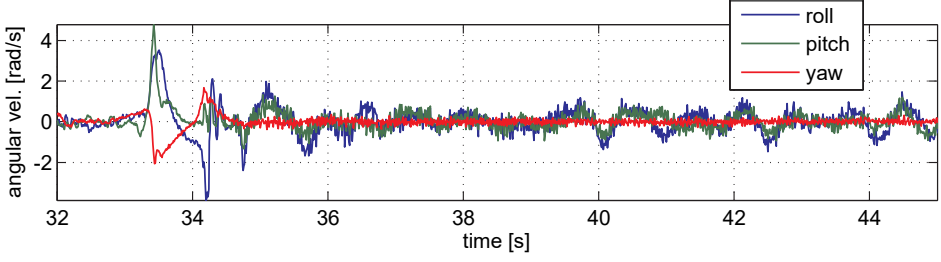


Figure 4.8: Angular velocities while throwing the SUAV in the air. At the throw the angular velocity reaches a maximum of 6rad/s.

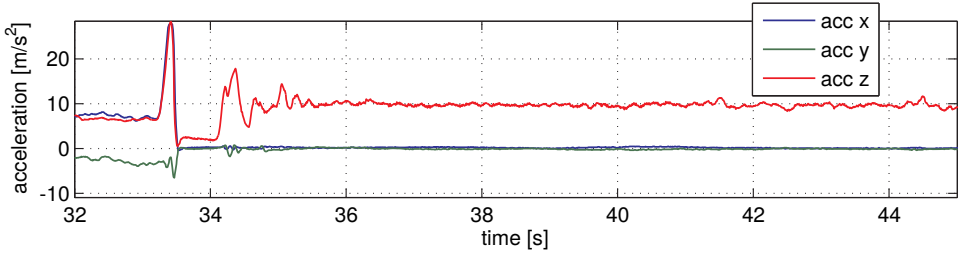


Figure 4.9: The acceleration graph shows the different stages of the throw. The SUAV experiences large acceleration at the beginning, enters a free fall flight as soon as we release it and experiences gravity once the position controller stabilizes it in hover mode. We use the low gravity threshold during free fall for a timer to actively delay the start of the position controller.

excitation, the acceleration reaches up to 40m/s^2 . Once the SUAV is released, it enters a free fall flight barely measuring any acceleration. We use this low acceleration point for a timer to start the position control. Just after $t=34\text{s}$ this control starts and stabilizes the SUAV within 1.5s to hover mode, hence the 9.81m/s^2 in the z-accelerometer after stabilization.

Fig. 10 depicts the velocity the SUAV experiences. It reaches a maximum of 3.7m/s . The free fall period shows as a clean linear velocity decrease until the position controller starts just after $t=34\text{s}$.

Since the proposed algorithm can estimate the metric scale Λ using inertial cues, the metric scene distance is observable. Thus, the controller can keep the SUAV at a fix distance with respect to the scene. Fig. 11 shows this estimated distance during the throw. The parabola flown during free fall after releasing the vehicle is well visible. After $t=34\text{s}$ the controller starts and tries to keep the SUAV leveled. A clear overshoot is visible and at about $t=40\text{s}$ (i.e. 7s after tossing the vehicle) the SUAV is stabilized at a distance of roughly 1.4m to the scene. The overshoot is mainly caused

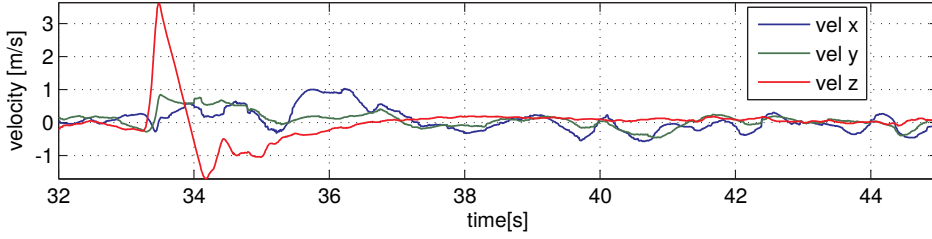


Figure 4.10: Velocity plot of the thrown SUAV. The maximal velocity is as high as 3.7m/s. The free fall period is well visible in the linear velocity decrease until just after $t=34$ s when the position controller starts.

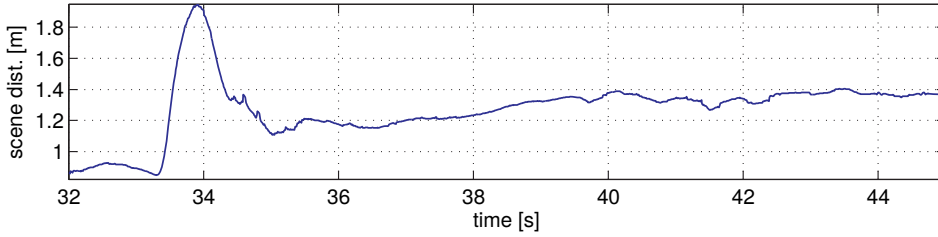


Figure 4.11: Estimation of the metric scene distance. This state is observable since the metric scale factor Λ is observable. Thus, the SUAV can be controlled to keep constant scene distance. The state is accurately tracked during the throw and the free fall phase until the controller starts and keeps the vehicle at a height of about 1.4m above the scene.

by the non-converged metric scale Λ . As the vehicle moves and excites the accelerometers, more information is acquired to converge this state and the position control becomes more precise. In fact, this is a positive feedback since an unconverged scale factor leads to imprecise position control, which leads to excitation, which leads to information for better scale factor estimation.

Figure 4.12 shows the attitude estimation. Note that the above described state estimation includes the drift free estimation of yaw. The attitude is independent of the scale factor convergence and can thus be stabilized much more quickly. Note in the graph that, after stabilization, the yaw is roughly constant since it is observable.

4.6 Conclusion

In this chapter, we presented a GPS independent inertialoptical flow (IOF) approach which significantly advances the state-of-the-art with respect to

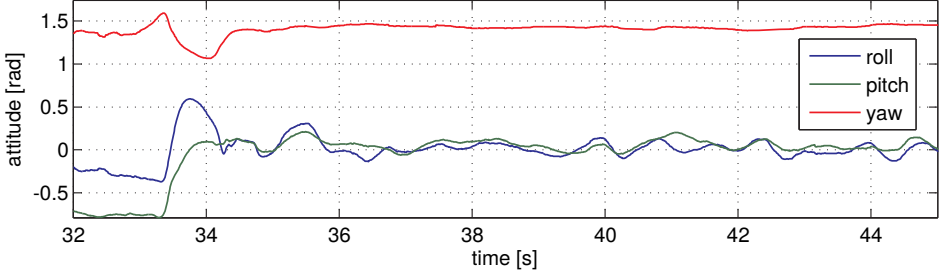


Figure 4.12: Attitude estimation while throwing the SUAV. Note, that our framework provides a drift free estimation of the full attitude including yaw. Apart from some corrective actions in roll and pitch the graph shows this non-drifting behavior.

fast and robust deployment of small rotorcraft SUAVs without the need for specific initialization procedures or previous calibration.

We only assume pre-calibrated intrinsic camera parameters. The framework fully self-calibrates the drifting IMU biases, the IMU-camera 6DoF extrinsic transformation, and the metric scale to estimate metric velocity and scene distance for accurate SUAV control. Our approach does not require hardware time-synchronization. Instead, we use a soft time-synchronization via an NTP like protocol. We analyzed in detail the effects of time misalignment and showed that our approach is robust against misalignment, typically occurring on real systems.

Our optical flow based vision module computes the 3D velocity vector in cluttered, 3D environment. Since the scene distance computation requires the extraction of the dominant terrain plane, we developed an approach which is robust against clutter on this plane allowing correct vision-based measurements in most scenarios.

The robustness against time misalignment and scene clutter together with our proposed robust initialization method which uses only two consecutive camera frames culminated in a framework which practically gives instantaneous state estimates after powering up the vehicle. Erroneous measurements can simply be skipped since we do not use any feature history or (local) map. This allows drastic short-term excitations and originated in the first *throw-and-go* SUAV purely navigating on a single camera and IMU.

Navigation of UAV Using Phased Array Radio

This chapter is based on S. M. Albrektsen, A Sægrov, and T. A Johansen. Navigation of UAV using phased array radio. In *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 138–143, Linköping, Sweden, October 3–5 2017. doi: 10.1109/RED-UAS.2017.8101657

5.1 Introduction

Today, two major challenges with unmanned aerial vehicle (UAV) flights beyond visual line of sight (BVLOS) are navigation and communication. UAVs' abilities to cover large distances in a short amount of time, and their maneuverability, make them valuable tools for many civilian tasks such as surveillance of power lines, search and rescue and scientific research. To be able to safely perform these tasks, however, an operator needs to know where the UAVs are, what they are sensing, and he or she needs to be able to send updated commands to the UAV.

When it comes to positioning sensors there are two main categories; absolute positioning systems, which measure a position in relation to a fixed point, and relative positioning systems, which measure a position in relation to the previous position. When using relative positioning sensors, the errors accumulate, as old estimates are added to the current estimate, and thus the accuracy of the estimated position will deteriorate over time. Conversely, an absolute positioning system will typically have bounded errors, and the errors will not vary over time as they are measured directly. The most commonly used absolute positioning systems today are global navigation

satellite systems (GNSS) such as Global Positioning System (GPS), due to the low integration cost and global coverage.

There are, unfortunately, challenges when relying solely on GNSS positioning. Hardware failures, operation in areas with weak signal reception due to multipath or atmospheric effects, and malicious disruption of the signal through jamming [103], spoofing [74], selective availability (SA) [76], or unintended electronic interference from other systems, makes it vital to have alternative sources of navigation. The state-of-the-art solution to GPS-less navigation is using computer vision [14, 41, 54, 73, 75, 81, 96, 145]. The main weakness of vision-based solutions is the need for visual features and such systems are therefore susceptible to both atmospheric and light conditions. When flying over a calm ocean there are also few visible features, even with high visibility. Another challenge with computer vision systems is that image analysis is both complex and computationally expensive. As the downlink data-rate from a UAV to the ground usually is too limited to transfer a high-quality video stream, especially on long-range flights, data analysis needs to be performed on-board. Due to the limited onboard computational power it is advantageous to reduce the computational power needed by the navigation system, to be able to focus on the mission specific tasks instead.

By using a phased array radio system (PARS), both the challenges of communication and navigation are addressed. By using electrical beam-forming, the PARS is able to direct the energy from the transmitting antenna elements towards the receiving radio [88, 133, 134, 140]. In addition to allowing efficient high-rate data transfer over long distances, the system is also able to provide positioning data. The position estimates are obtained by accurately timing the round-trip time of the signal and analyzing the direction of the incoming radio waves [87, 128]. This positioning system provides a local absolute position measurement, it is GNSS-independent, and no complex signal analysis needs to be performed on board the UAV.

This chapter studies a PARS as an alternative to GNSS-based positioning. As the PARS system provides absolute position estimates, it is drift free and thus a valuable sensor for long duration flights. Due to a degradation in vertical accuracy, a solution of using a barometer as an altimeter to compensate for this is effect is proposed. The system is tested and verified through a 35 min flight over the ocean and compared to a RTK (Real Time Kinematic) GPS solution.

This chapter first gives an overview of the PARS used in Section 5.2. Then an overview of the proposed UAV system, in which to use the PARS, is presented in Section 5.3. A method of estimating the pose of the ground

radio is given in Section 5.4. Experimental results from a flight with a UAV are then presented in Section 5.5 with results from radio measurements in Section 5.5.1 and barometer aided results, for handling vertical inaccuracies, in Section 5.5.2. Finally, a conclusion is given in Section 5.6.

5.2 Phased array radio navigation system

By accurately recording the time at which radio messages are sent from the ground radio and the time at which the corresponding wireless ACK-responses are received from the UAV radio, the round-trip time (RTT) of the signals are calculated by subtracting the internal computation time. This RTT is then used to calculate the distance between the ground radio and the UAV, knowing the speed of radio waves in air and the internal delays of the system. These direction and distance measurements then provide local absolute position measurements for the UAV, by knowing the pose of the ground radio.

Although the radio system does not have a global coverage, as opposed to GNSS, a sector of tens of kilometers can be covered from a single ground radio. The frustum covered by the ground radio spans 90° in both the horizontal direction and the vertical direction, with a maximal range of 60 km. Multiple ground radios can be used to ensure coverage of a larger operational area if needed. Note that the accuracy of the positional measurements in Cartesian coordinates will deteriorate proportionally with the measured distance, as the accuracy of the radio is specified as bearing and elevation angles from the ground radio.

As the PARS has a much higher signal-to-noise ratio (SNR) than GNSS signals transmitted from satellite orbit, jamming or spoofing the PARS position measurements are much more difficult. The PARS is also directional, as opposed to GNSS, and thus a malicious source needs to be in the visible sector of the ground radio to disrupt the position estimates. In addition, position estimates sent from the ground radio can be strongly encrypted to verify the sender's origin.

5.3 Experimental system overview

The phased array radio system used in this experiment is the Radionor Communications CRE2 189 PARS ground radio with a 8 by 8 grid of antenna elements, and an CRE2-144-LW with four antenna elements placed on the nose of the UAV. The onboard CRE2-144-LW has a weight of 85 g, dimensions of 120 mm x 65 mm x 13.3 mm, and uses AES-256 encryption.

The CRE-144-LW is depicted in Fig. 5.1. To provide high efficiency data transfer from a ground station to a UAV, the PARS uses electrical beam-forming to focus the transmitted energy in one direction. To properly be able to use beamforming, the ground radio needs to know the direction and preferably the distance to the UAV. To find this direction, the ground radio first sends an omnidirectional "ping"-signal, then observes the incoming response from the UAV radio, to find the direction towards UAV's antennas. The directional vector from the ground radio's antennas towards the UAV's radio antennas is calculated by observing how the signals are received by the different antenna elements on the ground radio. By using electrical beam-forming to increase the performance of the radio system, a maximum user data throughput of 15 Mbps at 20 km, 7 Mbps at 30 km, and 2.3 Mbps at 60 km is achieved. An overview of the experimental setup is given in Figure 5.2.

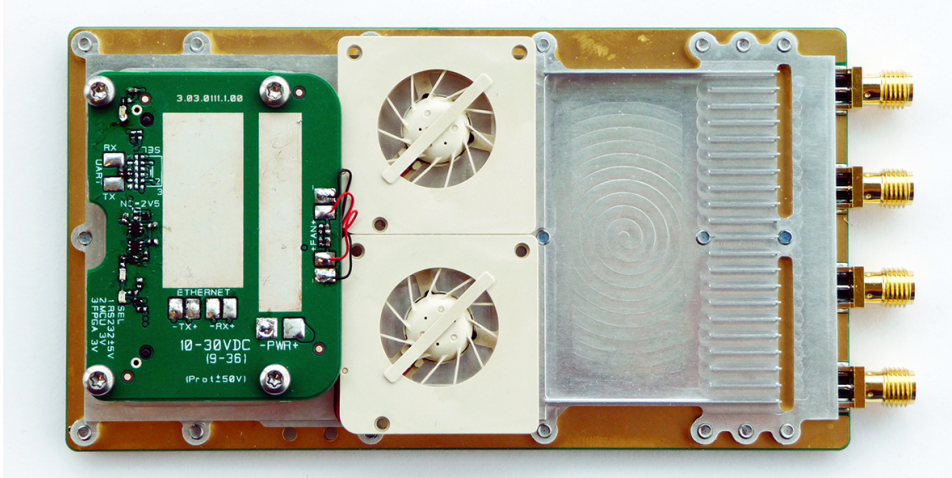


Figure 5.1: The CRE2-144-LW phased array radio

As can be seen in Figure 5.3, the system on board the UAV is split in to three main parts; the avionics, the SenTiStack, and the PARS. The avionics is responsible for the flight critical components of the UAV and can operate without any other parts of the system during normal conditions. This system is based around the PIXHAWK[92] autopilot with a LEA-7N GPS receiver [135], the PX4 Airspeed Sensor based on the MS4525DO sensor [90], and the standard integrated PIXHAWK sensor suite with IMUs[66, 126, 127] and a MEAS MS5611 barometer [89].

The SenTiStack, described in Section 3.4.2, is responsible for providing a more accurate and robust navigation solution for the UAV using RTK

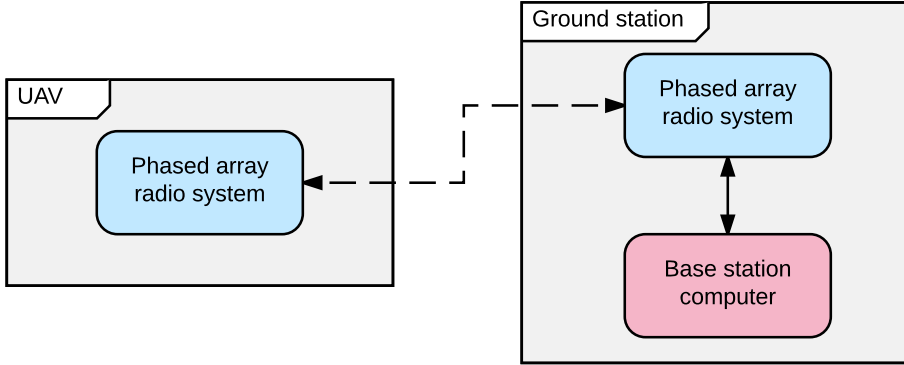


Figure 5.2: A minimal overview of a UAV and ground station system. For clarity, the avionics are not shown.

GPS, a tactical grade IMU and a hardware synchronization board. To be able to calculate a high precision position solution based on GPS, raw satellite data with carrier-phase measurements are recorded on the base station. These measurements can either be transmitted from the ground station to the UAV, or be stored on the ground station and only be used for post-processing. A separate GPS receiver, the ublox LEA M8T [136], is used due to the difficulties of relaying raw GPS measurements from the GPS receiver attached to the PIXHAWK.

To accurately synchronize the data a hardware synchronization board, the SyncBoard, described in Chapter 2, is used to capture the PPS (pulse per second) signal from the GPS receiver and the time of validity (TOV) from the IMU, and attach accurate timestamps to the corresponding sensor messages. The data recorded by the SyncBoard is then transferred through USB to the ODroid-XU4 on-board computer and stored on the attached embedded multi-media controller (EMMC) storage.

5.3.1 Real-time kinematic GNSS

To improve the accuracy of GNSS positioning systems, the real-time kinematic (RTK) solution is commonly used. By having a stationary ground station with a known location and comparing the phase of the signal's carrier wave received at the ground station to the measurements received on-board the UAV, the RTK position can be calculated. By multiplying the carrier wavelength with the number of whole cycles between the satellite and the UAV, the phase difference can be compensated for and centimeter-level pre-

cision can be achieved. This makes it a good option for a ground-truth when evaluating the performance of the PARS.

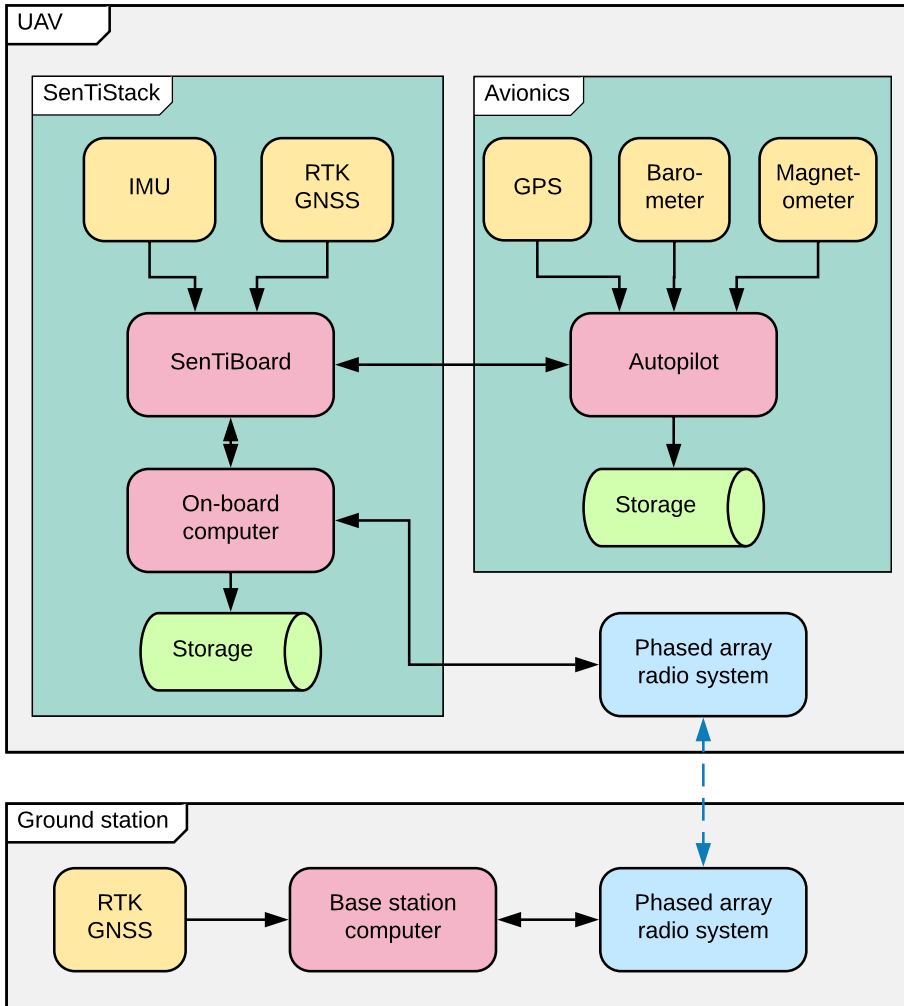


Figure 5.3: System overview of the UAV payload and ground station. The dashed line between the radios represents wireless communication. For simplicity, some components of the avionics are not shown in this overview.

5.4 Ground radio pose estimation

Each time the ground radio is moved, its pose needs to be estimated. This can be done manually, by measuring the position using a GNSS receiver and the attitude using an inclinometer and a compass, but a more accurate, time efficient and elegant method is to estimate the pose automatically.

By accurately calculating the position of the UAV using RTK satellite navigation, and accurately synchronizing the measurements from the ground radio measurements with the GPS clock, the radio-pose which would produce the observed measurements can be calculated. To compensate for noise and other inaccuracies of the measurements, several such measurement pairs are used. In [60], summarized by [123], this is done by taking the singular value decomposition (SVD) of the mean-subtracted measurements and then using the result to calculate the rotation matrix, \mathbf{R} , for the ground radio in the local East-North-Up (ENU) coordinate system.

First we convert the radio measurements, given in bearing, elevation and range, to Cartesian coordinates. Then we organize the measurements from the RTK solution and the radio measurements in two separate vectors.

$$\mathbf{p}_{\text{rtk}} = \begin{bmatrix} (x, y, z)_{1,\text{rtk}} \\ (x, y, z)_{2,\text{rtk}} \\ \vdots \\ (x, y, z)_{n,\text{rtk}} \end{bmatrix} \quad \mathbf{p}_{\text{radio}} = \begin{bmatrix} (x, y, z)_{1,\text{radio}} \\ (x, y, z)_{2,\text{radio}} \\ \vdots \\ (x, y, z)_{n,\text{radio}} \end{bmatrix} \quad (5.1)$$

Then we subtract the mean value of the measurements:

$$\mathbf{p}_{\text{rtk}}^* = \mathbf{p}_{\text{rtk}} - \bar{\mathbf{p}}_{\text{rtk}} \quad (5.2)$$

$$\mathbf{p}_{\text{radio}}^* = \mathbf{p}_{\text{radio}} - \bar{\mathbf{p}}_{\text{radio}} \quad (5.3)$$

and use the SVD as follows:

$$(\mathbf{p}_{\text{radio}}^*)^\top \cdot \mathbf{p}_{\text{rtk}}^* = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\top \quad (5.4)$$

Then we calculate the rotation matrix as in Equation 25 in [123]. The $\det(\mathbf{V}\mathbf{U}^\top)$ term comes from the special orientation reflection case described in [123].

$$\mathbf{R} = \mathbf{V} \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & & \det(\mathbf{V}\mathbf{U}^\top) \end{pmatrix} \mathbf{U}^\top \quad (5.5)$$

The position of the ground radio, t , is then calculated by:

$$t = \bar{p}_{\text{rtk}} - R\bar{p}_{\text{radio}} \quad (5.6)$$

To achieve more accurate results, the measurements can be weighted based on the estimated accuracy from the sensors.

5.5 Experimental results

To verify the positional measurements from the PARS, an experiment was carried out using a Skywalker X8 UAV at Agdenes outside Trondheim, Norway on June 23rd 2016 in good weather conditions and a forecasted wind of 15 km/h.

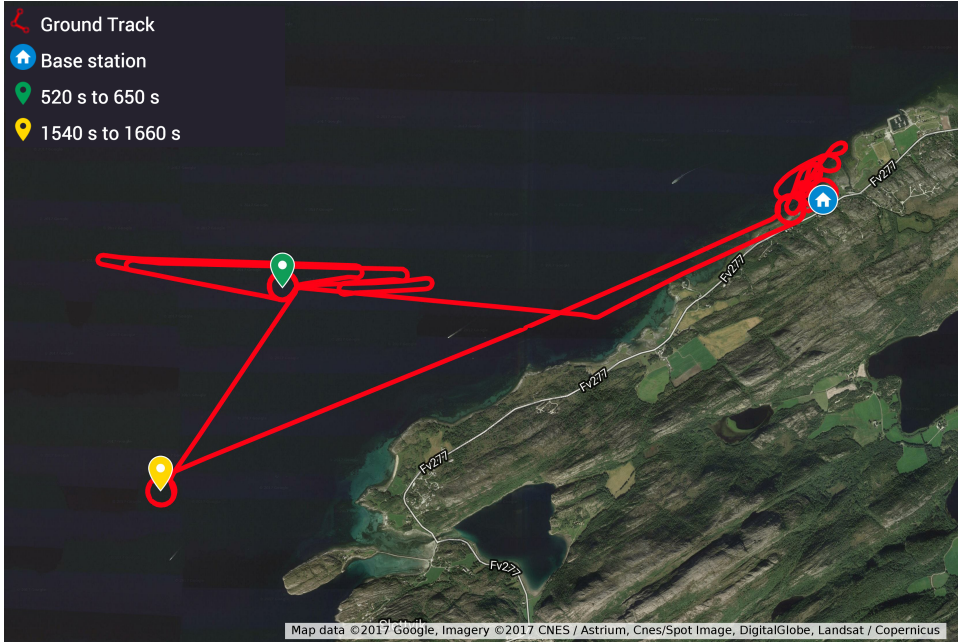


Figure 5.4: A map of the flight track, with indicators showing the loitering circles from 520 s to 650 s and 1540 s to 1660 s after takeoff. The icon with the building indicates the base station.

By recording GPS measurements at a stationary location close to the mission area, the accuracy of the GPS measurements can be improved, by compensating for local atmospheric disturbances. An open source RTK solver, RTKLIB, was used to find the real-time kinematic GPS solution. This solution is labeled RTK in the plots in this section. Note that the

RTK solution is only used to calibrate the position of the PARS and for verification but is not used to improve the radio measurements.

To ensure that the data for the ground-pose calibration is reliable, UAV missions can be preceded by a calibration step. To calibrate the ground radio pose, the UAV should be piloted in an area with good GPS coverage, while recording PARS measurements. This step can either be a separate flight, or the initial part of a longer mission. The recorded dataset can then be used in the calibration step described in Section 5.4.

5.5.1 Raw radio measurements

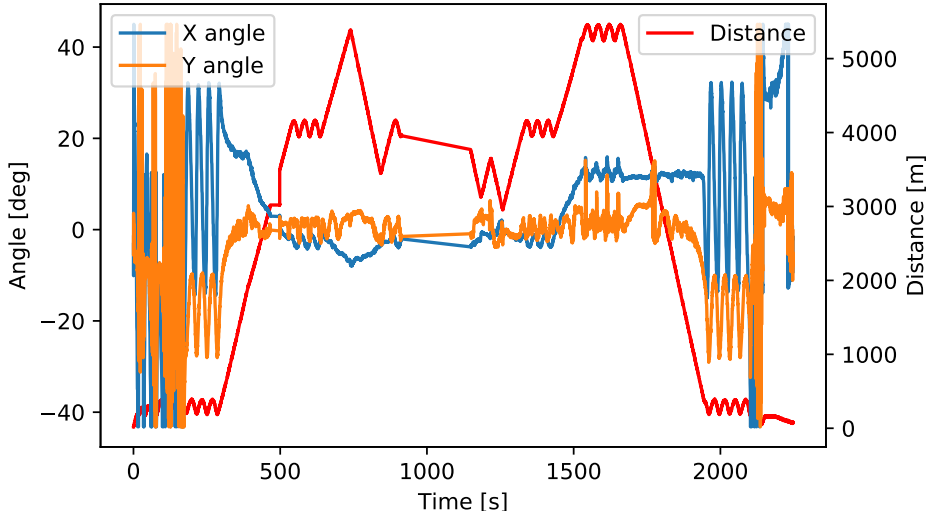


Figure 5.5: Measurements from the phased array radio system

Figure 5.5 shows the measurements from a complete UAV flight with a maximum measured distance of approximately 5 km from the ground station. When the UAV is close to the base station, from 0 s to 175 s after takeoff, the measured angle fluctuates as the UAV is outside the visible sector of the radio. Note that although the UAV maintains a fixed altitude, the angle measurement in the y-direction varies with approximately 4° at about 4125 m. This is an inaccuracy likely due to reflections of the radio beams in the surface of the water, and results in an error of the position measurement from the radio of $4125 \text{ m} \sin(4^\circ) \approx 288 \text{ m}$. Figure 5.6 shows a ground track of the flight from both the RTK GPS and radio measurements, with the radio position and opening frustum in the horizontal plane shown.

Figure 5.7 shows ENU plots from both the RTK GPS measurements, and Figure 5.8 shows zoomed-in views at times the UAV was loitering. In the *Horizontal* column of Table 5.2 the number of measurements in the horizontal plane within a specified accuracy are summarized. The error is calculated as the root-mean-square (RMS) of difference from the RTK position measurements.

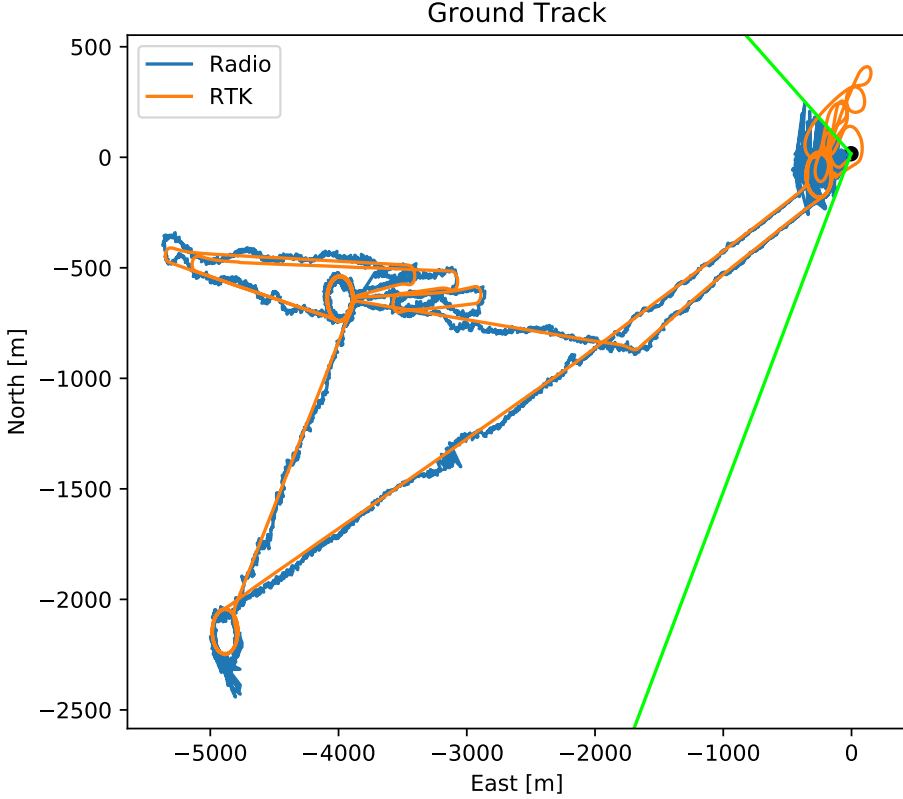


Figure 5.6: Ground track of the PARS measurements and the RTK solution. The black circle marks the radio position, and the green lines indicate the visible frustum from the radio in the horizontal plane.

5.5.2 Barometric vertical correction

To compensate for the inaccuracies of the radio measurements in the vertical plane, measurements from the standard barometer (altimeter) connected to the PIXHAWK autopilot is used. As can be seen from Figure 5.9, these

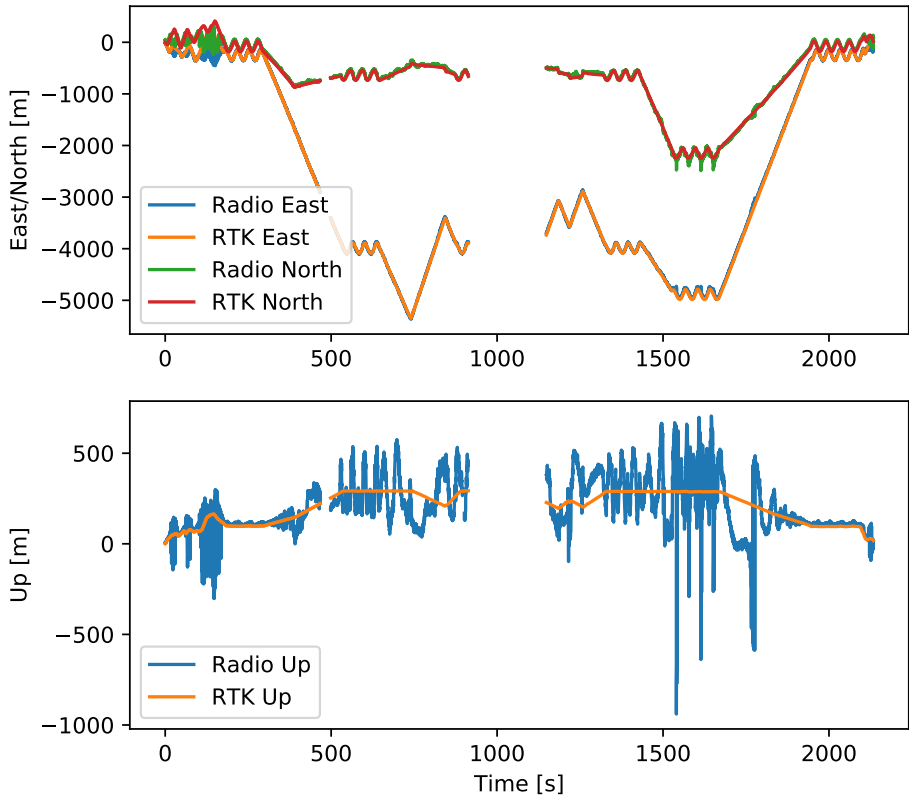


Figure 5.7: Position measurements from the PARS, and RTK-GPS measurements for reference. The data from 897-901s and 913-1148s are missing due to a file transfer that disrupted the measurements.

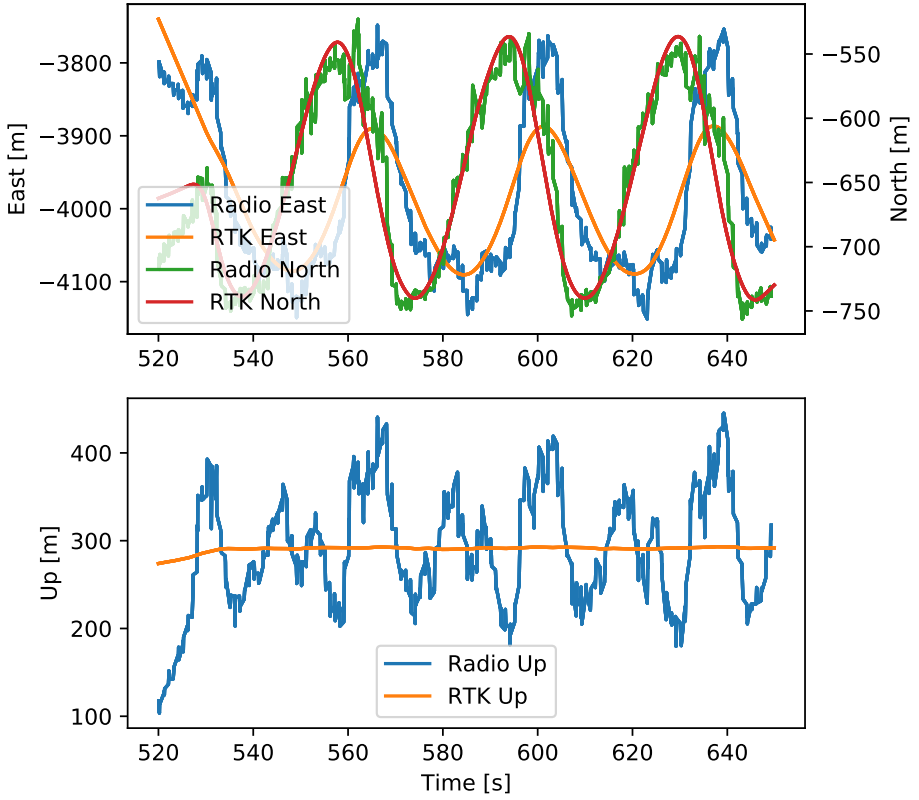


Figure 5.8: PARS measurements from 520s to 650s of loitering UAV at a fixed altitude, and RTK-GPS measurements for reference

measurements are significantly closer to the RTK measurements than the radio measurements. Although the errors of a barometer are unbounded and vary with weather conditions, they tend to be considerably more stable than for example position estimates based on IMU, and should provide sufficiently accurate readings for the duration of a typical flight in most scenarios. A full plot of the combined data can be seen in Figure 5.9.

Table 5.1 lists these numbers in addition to the mean error and standard deviation of the error of both the radio-only and barometer supported measurements. Figure 5.10 contains a cumulative histogram of the RMS error and Table 5.2 lists example points from the histogram.

Table 5.1: Comparison of radio-only and radio measurements compensated with barometer measurements

	Radio only	Radio + barometer
Mean error	96.22 m	24.23 m
Standard deviation	95.26 m	16.66 m

Table 5.2: Percentage of measurements that are within a certain accuracy. The Horizontal measurements are radio-only measurements without the vertical components.

Measurements	Horizontal	Radio only	Radio + barometer
20.00 m	52.71 %	19.40 %	50.46 %
30.00 m	76.52 %	29.72 %	74.90 %
40.00 m	88.23 %	36.44 %	87.89 %
100.00 m	99.69 %	63.65 %	99.69 %
200.00 m	99.94 %	87.83 %	99.94 %

5.6 Conclusion

In this chapter we have studied a novel method of using a PARS and a barometer for absolute positioning of a UAV. As the phased array radio system is primarily used for communication, the system as a whole solves two major challenges with UAV operations. As radio beam reflections, caused by the ocean surface, disrupt the vertical accuracy of the position measurements of the UAV, a barometer is used as a vertical reference.

The PARS has been tested and verified with experimental results, and when combined with barometer readings, the mean error compared to RTK

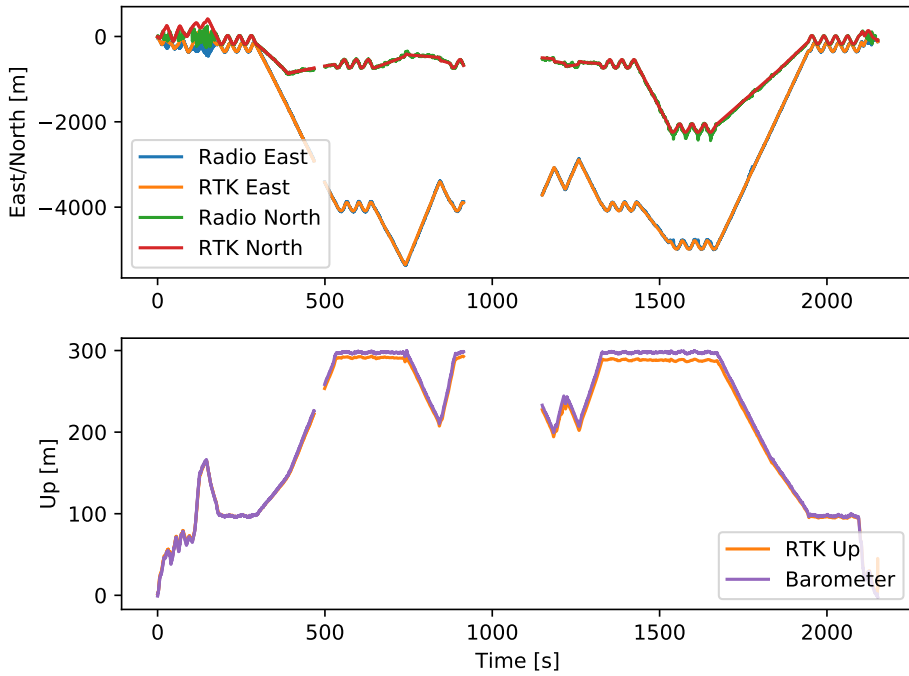


Figure 5.9: RTK-GPS plotted against radio measurements in the horizontal plane and barometer readings in the vertical plane.

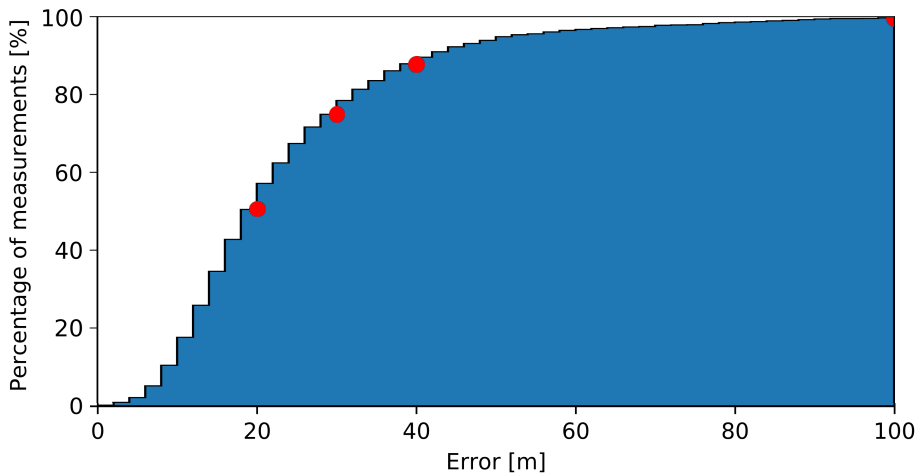


Figure 5.10: Cumulative histogram of the root mean square of radio measurement errors in all three directions.

GPS is 24.23 m and 87.89 % of the measurements are within 40 m of the RTK GPS measurements. As the system provides absolute position measurements, independent from GNSS measurements, it is a valuable navigation source for BVLOS UAV flights.

Phased Array Radio System Aided Inertial Navigation for Unmanned Aerial Vehicles

This chapter is based on S. M. Albrektsen, T. H. Bryne, and T. A. Johansen. Phased array radio system aided inertial navigation for unmanned aerial vehicles. In *2018 IEEE Aerospace Conference*, Big Sky, Montana, USA, March 3–10 2018. doi: 10.1109/AERO.2018.8396433. Note that Section 6.7 is not in the original publication.

6.1 Introduction

In this chapter the accuracy of the navigation solution of the absolute PARS measurements, previously presented in Chapter 5, are improved by using an inertial navigation system (INS). The INS is used to improve the position estimates in-between the radio measurements, and to improve the bandwidth of the system. It can furthermore act as a smoothing filter on the position estimates with a large variance and it makes the attitude of the UAV observable when combined with a magnetometer. The INS is, however, only accurate in short time intervals as the measurements it provides are relative to the previously estimated state, and these errors accumulate with time.

This chapter presents a navigation solution based on measurements from a PARS along with an INS, a barometer and a magnetometer. By aiding a high-bandwidth IMU with the absolute position measurements from a PARS, along with altitude measurements from a barometer and heading from a magnetometer, we achieve a high-bandwidth, drift-free navigation solution that is independent of GNSS. We test our filter in an experiment

and compare the results to a real-time kinematics (RTK) GNSS solution. Compared to this solution we achieve a combined root-mean-square error of approximately 26.3 m.

6.1.1 Chapter overview

We start by defining the preliminaries in Section 6.2 before we introduce the necessary steps needed to use the PARS as a positioning system in Section 6.3. We continue with presenting our nonlinear observer for aided INS in Section 6.4. An experiment was carried out, and a description of the system and hardware used is described in Section 6.5. The results from this experiment are presented in Section 6.6. An updated experiment was also performed, using identical hardware, but with updated software, and results from this experiment are presented in Section 6.7. Finally, a conclusion is given in Section 6.8.

6.2 Preliminaries

Before presenting the PARS-based positioning, and the PARS-aided INS, we state some preliminaries.

6.2.1 Notation

The Euclidean vector norm is denoted $\|\cdot\|_2$. The $n \times n$ identity matrix is denoted \mathbf{I}_n . Moreover, the transpose of a vector or a matrix is denoted $(\cdot)^\top$. Coordinate frames are denoted with $\{\cdot\}$. $\mathbf{S}(\cdot) \in SS(3)$ represents the skew symmetric matrix such that $\mathbf{S}(\mathbf{z}_1)\mathbf{z}_2 = \mathbf{z}_1 \times \mathbf{z}_2$ for two vectors $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^3$. In addition, $\mathbf{z}_{bc}^a \in \mathbb{R}^3$ denotes a vector \mathbf{z} , to frame $\{c\}$, relative $\{b\}$, decomposed in $\{a\}$. Moreover, \otimes denotes the Hamiltonian quaternion product. Saturation is represented by sat_\star , where the subscript indicates the saturation limit.

The rotation matrix, $\mathbf{R}_a^b \in SO(3)$, describes the rotation between two given frames $\{a\}$ and $\{b\}$. Equivalently, the rotation between $\{a\}$ and $\{b\}$ may be represented using the unit quaternion $\mathbf{q}_a^b = (s, \mathbf{r}^\top)^\top$ where $s \in \mathbb{R}^1$ is the real part of the quaternion and $\mathbf{r} \in \mathbb{R}^3$ is the vector part. In addition, the Euler angles (roll, pitch and yaw) are given as

$$\boldsymbol{\Theta} = (\phi, \theta, \psi)^\top. \quad (6.1)$$

Latitude and longitude on Earth is represented by $\mu \in [-\pi/2, \pi/2]$ and $\lambda \in (-\pi, \pi]$, respectively.

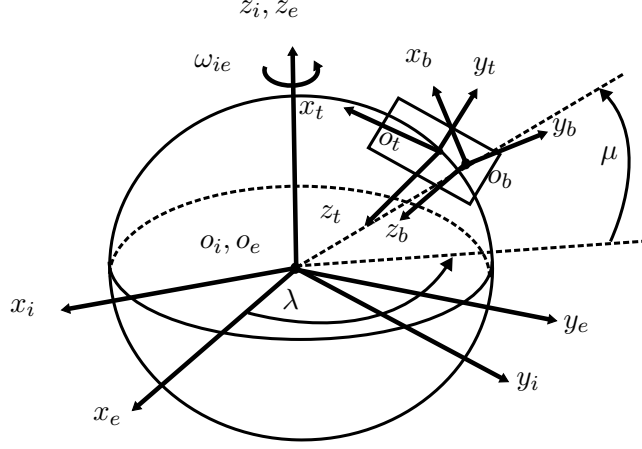


Figure 6.1: Definitions of the BODY, Tangent, ECEF and ECI reference frames.

6.2.2 Coordinate frames

This paper considers four coordinate frames; The Earth Centered Inertial (ECI) frame, the Earth Centered Earth Fixed (ECEF) frame, a tangent frame equivalent of an Earth-fixed North East Down (NED) frame, and the BODY reference frame, denoted $\{i\}$, $\{e\}$, $\{t\}$, and $\{b\}$, respectively (see Figure 6.1). The NED directions are respectively denoted N, E, D.

6.2.3 Inertial measurement units

A simplified measurement model of an IMU, providing specific force and angular rate sensor (ARS) measurements, is given as

$$\mathbf{f}_{\text{IMU}}^b = \mathbf{f}_{ib}^b + \mathbf{b}_{\text{acc}}^b + \mathbf{w}_{\text{acc}}^b \quad (6.2)$$

$$\boldsymbol{\omega}_{\text{IMU}}^b = \boldsymbol{\omega}_{ib}^b + \mathbf{b}_{\text{ars}}^b + \mathbf{w}_{\text{ars}}^b \quad (6.3)$$

where \mathbf{f}_{ib}^b is the specific force, relating to the acceleration and gravity vector, $\mathbf{g}_b^t = (0, 0, g)^\top$ through

$$\begin{aligned} \mathbf{f}_{ib}^b &= \mathbf{R}_n^b \dot{\mathbf{v}}_{ib}^t - \mathbf{R}_t^b \mathbf{g}_b^t \\ &= \mathbf{a}_{ib}^b + \mathbf{S}(\boldsymbol{\omega}_{ib}^b) \mathbf{v}_{ib}^b - \mathbf{R}_t^b \mathbf{g}_b^t. \end{aligned} \quad (6.4)$$

$\boldsymbol{\omega}_{ib}^b$ represents angular velocity, while \mathbf{v}_{ib}^b , represents the BODY-fixed linear velocity. The BODY-fixed acceleration is represented by \mathbf{a}_{ib}^b , while $\mathbf{S}(\boldsymbol{\omega}_{ib}^b) \mathbf{v}_{ib}^b$ constitutes the centripetal accelerations. \mathbf{b}_\star^b represent the accelerometer

(acc) biases, and the angular rate sensor (ars) biases, respectively. \mathbf{w}_\star^b represent noise.

6.2.4 Strapdown equations

The NLO-based INS is derived using

$$\dot{\mathbf{p}}_{tb}^t = \mathbf{v}_{tb}^t \quad (6.5)$$

$$\dot{\mathbf{v}}_{tb}^t = -2\mathbf{S}(\boldsymbol{\omega}_{it}^t)\mathbf{v}_{tb}^t + \mathbf{R}_b^t \mathbf{f}_{ib}^b + \mathbf{g}_b^t \quad (6.6)$$

$$\dot{\mathbf{q}}_b^t = \frac{1}{2}\mathbf{q}_b^t \otimes \begin{pmatrix} 0 \\ \boldsymbol{\omega}_{ib}^b \end{pmatrix} - \frac{1}{2}\mathbf{q}_b^t \otimes \begin{pmatrix} 0 \\ \boldsymbol{\omega}_{it}^t \end{pmatrix} \quad (6.7)$$

as strapdown equations. Moreover,

$$\boldsymbol{\omega}_{it}^t = \boldsymbol{\omega}_{ie}^t = \begin{pmatrix} \cos(\mu) \\ 0 \\ -\sin(\lambda) \end{pmatrix} \boldsymbol{\omega}_{ie}, \quad (6.8)$$

[34], due to $\{t\}$ being Earth fixed and thus $\boldsymbol{\omega}_{et}^t = \mathbf{0}_{3 \times 1}$. \mathbf{p}_{tb}^t and \mathbf{v}_{tb}^t represents the position and velocity vectors, respectively.

6.3 Phased array radio system positioning

As described in Section 6.1 the range, elevation and bearing from the ground antenna towards the UAV can be calculated by observing incoming signals from the UAV. To be able to use these measurements for navigation, they need to be rotated into the UAV's positional reference frame. To do this, the pose of the base station needs to be known.

6.3.1 PARS base station pose

To be able to use the PARS system for different experiments, a mobile PARS base station is used. A downside of this approach is that the pose of the base station needs to be calibrated on a per-mission basis. Although a rough estimate can be done manually, an automatic calibration routine is advantageous, not only to save time, but also to increase the accuracy of the pose estimate.

To ensure high-quality data for the base station pose estimation, missions can be preceded by a calibration phase, where the UAV is maneuvered in an area with good GNSS coverage and optimal visibility from the base station. From this data, an RTK solution can be calculated, and by using

the position measurements from the PARS, the pose of the base station PARS can be estimated as described in Chapter 5.

Positioning: range/bearing/elevation measurements

Although the primary functionality of the PARS is data transfer, the system can also be used as an absolute positioning measurement system. By observing the phase difference of the incoming signal between the different antenna elements in the radio array, the bearing and elevation of the UAV can be observed, in the ground radio's frame of reference, $\{r\}$. There exists a variety of methods that aim to solve this problem of direction-of-arrival (DOA) [77], perhaps most notably Schmidt's MUSIC [114] and Roy and Kaliath's ESPRIT [110]. Furthermore, by accurately timing the transmission time of the signal and subtracting internal processing time, the range measurement is found. To be able to use these measurements for navigation, they need to be rotated and translated into the UAV's positional reference frame. Calculations of the pose from the mobile PARS ground station to the tangent frame is described in Chapter 5. As the calculation of the PARS measurements are done on the ground, a minimal amount of processing power is needed by the computer on-board the UAV.

The PARS range, elevation and bearing measurements can be used to calculate the relative position of the UAV in a local Earth-fixed frame. When using the tangent frame, as done in this paper, the range/bearing measurements related to the UAV position, through the radio coordinate system $\{r\}$ using,

$$\rho_m = \rho_u + \varepsilon_\rho, \quad (6.9)$$

$$\psi_m = \psi_u + \varepsilon_\psi, \quad (6.10)$$

$$\theta_m = \theta_u + \varepsilon_\theta, \quad (6.11)$$

where

$$\rho_u = \|\mathbf{p}_{\text{PARS}}^r\|_2, \quad (6.12)$$

$$\tan(\psi_u) = p_{rb,y}^r / p_{rb,x}^r, \quad (6.13)$$

$$\tan(\theta_u) = -p_{rb,z}^r / \rho_{\text{hor}} \quad (6.14)$$

with

$$\|\mathbf{p}_{\text{PARS}}^r\|_2 = \sqrt{(p_{rb,x}^r)^2 + (p_{rb,y}^r)^2 + (p_{rb,z}^r)^2} \quad (6.15)$$

and

$$\bar{\rho}_u = \sqrt{(p_{rb,x}^r)^2 + (p_{rb,y}^r)^2} \quad (6.16)$$

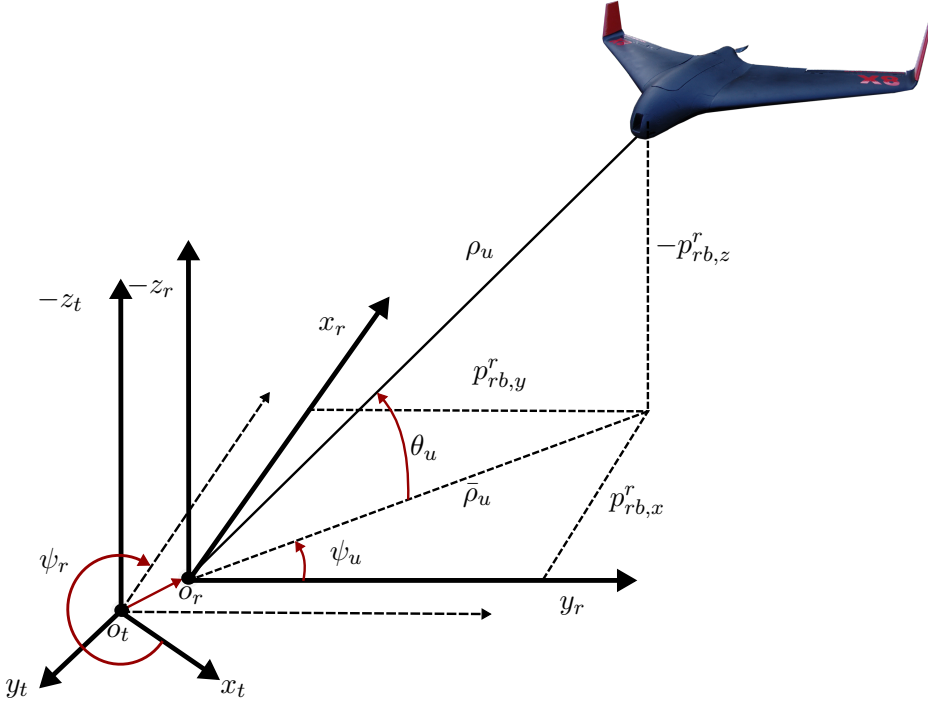


Figure 6.2: Range/bearing measurements. The red vector denotes the vector from the $\{t\}$ frame to the $\{r\}$, given in the $\{t\}$ frame, denoted \mathbf{l}_{tr}^r . The angle ψ_r represents the azimuth angle between the $\{t\}$ frame and the $\{t\}$ frame.

while ε_\star represents noise. Moreover, the relationships of (6.12)–(6.14) are like those in [131, Ch. 13.6.2.2], used for radar tracking of aircraft, and can derived from

$$\mathbf{p}_{\text{PARS}}^r = \begin{pmatrix} p_{rb,x}^r \\ p_{rb,y}^r \\ p_{rb,z}^r \end{pmatrix} = \begin{pmatrix} \rho_u \cos(\psi_u) \cos(\theta_u) \\ \rho_u \sin(\psi_u) \cos(\theta_u) \\ -\rho_u \sin(\theta_u) \end{pmatrix}. \quad (6.17)$$

according to Figure 6.2. Based on (7.1), the PARS position is given in the tangent frame as $\mathbf{p}_{\text{PARS}}^t = \mathbf{R}_r^t(\psi_r) \mathbf{p}_{\text{PARS}}^r + \mathbf{l}_{tr}^t$, where \mathbf{l}_{tr}^t is the vector from the $\{t\}$ frame to the $\{r\}$ frame, decomposed in the $\{t\}$. ψ_r represents the azimuth angle from $\{t\}$ to $\{r\}$. ψ_r is obtained in during calibration of the PARS ground antenna.

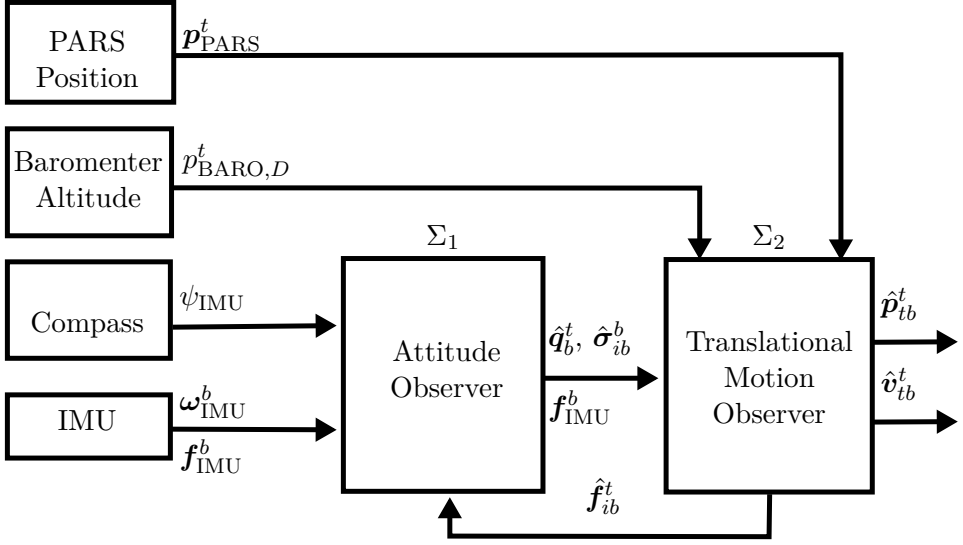


Figure 6.3: NLO structure overview

6.4 Nonlinear observer for aided INS

The position, velocity and attitude (PVA) of the UAS is estimated using a feedback-interconnected nonlinear observer integration strategy as depicted in Figure 6.3, based on the work of [47] and references therein. The PVA estimation is carried out in two steps. First the attitude is estimated by using rate gyro, specific force and heading reference measurements. The attitude observer is further aided by the second step, consisting of a Translational Motion Observer (TMO) providing specific force estimates in the navigation frame, together with 3-DOF position and velocity estimates based on the estimated attitude, in addition to, specific force, and aiding sensors.

6.4.1 Aiding sensors

The aiding measurements in the attitude observer is the accelerometer used for leveling, and the UAVs autopilot compass, ψ_{auto} . Due to signal reflections in the ocean surface, the vertical accuracy of the PARS is reduced significantly. To compensate for this inaccuracy, the TMO is aided by a barometer in addition to using the horizontal PARS position obtained in Section 6.3.1, similarly as in Chapter 5.

6.4.2 Attitude observer

The NLO for estimating the attitude between the $\{b\}$ and the $\{t\}$ frame is given similar to [47],

$$\Sigma_1 : \begin{cases} \dot{\hat{\mathbf{q}}}_b^t = \frac{1}{2} \hat{\mathbf{q}}_b^t \otimes \begin{pmatrix} 0 \\ \hat{\boldsymbol{\omega}}_{ib}^b \end{pmatrix} - \frac{1}{2} \begin{pmatrix} 0 \\ \boldsymbol{\omega}_{it}^t \end{pmatrix} \otimes \hat{\mathbf{q}}_b^t, & (6.18a) \\ \hat{\boldsymbol{\omega}}_{ib}^b = \boldsymbol{\omega}_{IMU}^b - \hat{\mathbf{b}}_{ars}^b + \hat{\boldsymbol{\sigma}}_{ib}^b, & (6.18b) \\ \hat{\mathbf{b}}_{ars}^b = \text{Proj}(\hat{\mathbf{b}}_{ars}^b, -k_I \hat{\boldsymbol{\sigma}}_{ib}^b), & (6.18c) \end{cases}$$

where $\text{Proj}(\star, \star)$ denotes the angular rate bias projection algorithm ensuring that $\|\hat{\mathbf{b}}_{ars}^b\|_2 \leq M_{\hat{\mathbf{b}}_{ars}}$ for $M_{\hat{\mathbf{b}}_{ars}} > M_{b_{ars}}$ [46], and k_I is the gain associated with the rate gyro bias estimation. The NLO is structurally the same as in [47], where the attitude between the $\{b\}$ and the $\{e\}$ frame was estimated. Moreover, the observer's nonlinear injection term, $\hat{\boldsymbol{\sigma}}_{ib}^b$, is given as

$$\hat{\boldsymbol{\sigma}}_{ib}^b = k_1 \underline{\mathbf{v}}_1^b \times \mathbf{R}^\top(\hat{\mathbf{q}}_b^t) \underline{\mathbf{v}}_1^t + k_2 \underline{\mathbf{v}}_2^b \times \mathbf{R}^\top(\hat{\mathbf{q}}_b^t) \underline{\mathbf{v}}_2^t, \quad (6.19)$$

where the measurement vectors $\underline{\mathbf{v}}_{1,2}^b$ and reference vectors $\underline{\mathbf{v}}_{1,2}^t$ are calculated using

$$\underline{\mathbf{v}}_1^b = \underline{\mathbf{f}}^b, \quad \underline{\mathbf{v}}_1^t = \underline{\mathbf{f}}^t, \quad (6.20)$$

$$\underline{\mathbf{v}}_2^b = \underline{\mathbf{f}}^b \times \underline{\mathbf{c}}^b, \quad \underline{\mathbf{v}}_2^t = \underline{\mathbf{f}}^t \times \underline{\mathbf{c}}^t. \quad (6.21)$$

Furthermore, the measurement and corresponding reference vector pairs in (6.20)–(6.21) are constructed as

$$\underline{\mathbf{f}}^b = \frac{\mathbf{f}_{IMU}^b}{\|\mathbf{f}_{IMU}^b\|_2}, \quad \underline{\mathbf{f}}^t = \frac{\text{sat}_{M_f}(\hat{\mathbf{f}}_{ib}^t)}{\|\text{sat}_{M_f}(\hat{\mathbf{f}}_{ib}^t)\|_2}, \quad (6.22)$$

$$\underline{\mathbf{c}}^b = \begin{pmatrix} \cos(\psi_{\text{auto}}) \\ -\sin(\psi_{\text{auto}}) \\ 0 \end{pmatrix}, \quad \underline{\mathbf{c}}^t = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad (6.23)$$

where ψ_{auto} is a heading measurement provided from a given heading reference such as a compass or a attitude and heading reference system (AHRS). $\hat{\mathbf{f}}_{ib}^t$ is the estimated specific force, provided by the TMO, presented next in Section 6.4.3, as depicted in Figure 6.3. The benefit of using normalized vectors is that the vector pairs only provide direction, hence these are dimensionless, such that the gains $k_{1,2}$ can be considered as cut-off frequencies of the complementary filter Σ_1 [63]. Since the gains have unit rad/s, $\hat{\boldsymbol{\sigma}}_{ib}^b$ obtains the same unit as $\boldsymbol{\omega}_{IMU}^b$.

6.4.3 Translational motion observer

The TMO is similar to that of [47], except that here the tangent frame is used as navigation frame, and given as follows,

$$\Sigma_2 : \begin{cases} \dot{\hat{\mathbf{p}}}_{tb}^t = \hat{\mathbf{v}}_{tb}^t + \vartheta \mathbf{K}_{pp}^0 \tilde{\mathbf{y}}_{tb}^t & (6.24a) \end{cases}$$

$$\begin{cases} \dot{\hat{\mathbf{v}}}_{tb}^t = -2\mathbf{S}(\boldsymbol{\omega}_{ie}^t) \mathbf{v}_{tb}^t + \hat{\mathbf{f}}_{ib}^t + \mathbf{g}_b^t + \vartheta^2 \mathbf{K}_{vp}^0 \tilde{\mathbf{y}}_{tb}^t & (6.24b) \end{cases}$$

$$\begin{cases} \dot{\boldsymbol{\xi}}_{ib}^t = -\mathbf{R}(\hat{\mathbf{q}}_b^t) \mathbf{S}(\hat{\boldsymbol{\sigma}}_{ib}^b) \mathbf{f}_{IMU}^b + \vartheta^3 \mathbf{K}_{\xi p}^0 \tilde{\mathbf{y}}_{tb}^t & (6.24c) \end{cases}$$

$$\begin{cases} \hat{\mathbf{f}}_{ib}^t = \mathbf{R}(\hat{\mathbf{q}}_b^t) \mathbf{f}_{IMU}^b + \boldsymbol{\xi}_{ib}^t, & (6.24d) \end{cases}$$

where

$$\tilde{\mathbf{y}}_{tb}^t = \begin{pmatrix} p_{\text{PARS},N}^t \\ p_{\text{PARS},E}^t \\ p_{\text{BARO},D}^t \end{pmatrix} - \hat{\mathbf{p}}^t, \quad (6.25)$$

while \mathbf{K}_\star are gains associated with the PARS and the barometer measurements. $\boldsymbol{\xi}_{ib}^t$ is an auxiliary state used to estimate \mathbf{f}_{ib}^t . ϑ is a high-gain like parameter used to guarantee stability. Furthermore, by noting the linear time-varying (LTV) structure of (6.24) and defining

$$\mathbf{x} := \begin{pmatrix} \mathbf{p}_{tb}^t; & \mathbf{v}_{tb}^t; & \boldsymbol{\xi}_{ib}^t \end{pmatrix}, \quad (6.26)$$

the TMO can be written on LTV form as

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}(t)\mathbf{u} + \mathbf{D}(t, \hat{\mathbf{x}}) + \mathbf{K}(t)(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}), \quad (6.27)$$

with the system matrices,

$$\mathbf{A} = \begin{pmatrix} \mathbf{0}_{3 \times 3} & \mathbf{I}_3 & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{I}_3 \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} \end{pmatrix}, \mathbf{B}(t) = \begin{pmatrix} \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{R}(\hat{\mathbf{q}}_b^t) & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{R}(\hat{\mathbf{q}}_b^t) \end{pmatrix}, \quad (6.28)$$

the measurement matrix,

$$\mathbf{C} = \begin{pmatrix} \mathbf{I}_3 & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} \end{pmatrix}, \quad (6.29)$$

the vector,

$$\mathbf{D}(t, \hat{\mathbf{x}}) = \begin{pmatrix} \mathbf{0}_{3 \times 1}; & -2\mathbf{S}(\boldsymbol{\omega}_{ie}^t) \hat{\mathbf{v}}_{tb}^t + \mathbf{g}_b^t; & \mathbf{0}_{3 \times 1} \end{pmatrix}, \quad (6.30)$$

and the gain matrix,

$$\mathbf{K}(t) = \begin{pmatrix} \mathbf{K}_{pp} \\ \mathbf{K}_{vp} \\ \mathbf{K}_{\xi p} \end{pmatrix} = \begin{pmatrix} \vartheta \mathbf{K}_{pp}^0 \\ \vartheta^2 \mathbf{K}_{vp}^0 \\ \vartheta^3 \mathbf{K}_{\xi p}^0 \end{pmatrix}, \quad (6.31)$$

where

$$\mathbf{K}^0(t) = \left((\mathbf{K}_{pp}^0)^\top \quad (\mathbf{K}_{vp}^0)^\top \quad (\mathbf{K}_{\xi p}^0)^\top \right)^\top \quad (6.32)$$

is given obtain with $\mathbf{K}^0(t) = \mathbf{P}(t)\mathbf{C}^\top \mathbf{R}^{-1}(t)$, with $\mathbf{P}(t) = \mathbf{P}^\top(t) > 0$ being the solution of the time-scaled Riccati equation

$$\begin{aligned} \frac{1}{\vartheta} \dot{\mathbf{P}}(t) &= \mathbf{A}\mathbf{P}(t) + \mathbf{P}(t)\mathbf{A}^\top - \mathbf{P}\mathbf{C}^\top \mathbf{R}^{-1}(t)\mathbf{C}^\top \mathbf{P}(t) \\ &\quad + \mathbf{B}(\hat{\mathbf{q}}_b^t)\mathbf{Q}(t)\mathbf{B}^\top(\hat{\mathbf{q}}_b^t). \end{aligned} \quad (6.33)$$

Finally, the input is given as

$$\mathbf{u} = \left(\mathbf{f}_{\text{IMU}}^b; -\mathbf{S}(\hat{\boldsymbol{\sigma}}_{ib}^b)\mathbf{f}_{\text{IMU}}^b \right). \quad (6.34)$$

Moreover, the error states of the TMO can be defined as $\tilde{\mathbf{p}}_{tb}^t := \mathbf{p}_{tb}^t - \hat{\mathbf{p}}_{tb}^t$, $\tilde{\mathbf{v}}_{tb}^t := \mathbf{v}_{tb}^t - \hat{\mathbf{v}}_{tb}^t$, and $\tilde{\mathbf{f}}_{tb}^t := \mathbf{f}_{ib}^t - \hat{\mathbf{f}}_{ib}^t$, where the latter is obtained through a combination of (6.24c)–(6.24d), the resulting the error state is obtained,

$$\tilde{\mathbf{x}} := \left(\tilde{\mathbf{p}}_{tb}^t; \quad \tilde{\mathbf{v}}_{tb}^t; \quad \tilde{\mathbf{f}}_{ib}^t \right). \quad (6.35)$$

The corresponding error dynamics of the origin of Σ_2 is then obtained as

$$\dot{\tilde{\mathbf{x}}} = (\mathbf{A} - \mathbf{K}(t)\mathbf{C})\tilde{\mathbf{x}} + \boldsymbol{\rho}_1(t, \tilde{\mathbf{x}}) + \boldsymbol{\rho}_2(t, \boldsymbol{\chi}), \quad (6.36)$$

with

$$\boldsymbol{\rho}_1(t, \tilde{\mathbf{x}}) = \left(\mathbf{0}_{3 \times 1}; \quad -2\mathbf{S}(\boldsymbol{\omega}_{ie}^t)\tilde{\mathbf{v}}_{tb}^t; \quad \mathbf{0}_{3 \times 1} \right), \quad (6.37)$$

$$\boldsymbol{\rho}_2(t, \boldsymbol{\chi}) = \left(\mathbf{0}_{3 \times 1}; \quad \mathbf{0}_{3 \times 1}; \quad \tilde{\mathbf{d}}(t, \boldsymbol{\chi}) \right), \quad (6.38)$$

and where,

$$\begin{aligned} \tilde{\mathbf{d}}(t, \boldsymbol{\chi}) &= (\mathbf{I}_3 - \mathbf{R}(\tilde{\mathbf{q}})^\top) \mathbf{R}_b^t \left(\mathbf{S}(\boldsymbol{\omega}_{ib}^b)\mathbf{f}_{ib}^b + \dot{\mathbf{f}}_{ib}^b \right) \\ &\quad - \mathbf{S}(\boldsymbol{\omega}_{it}^t) (\mathbf{I}_3 - \mathbf{R}^\top(\tilde{\mathbf{q}})) \mathbf{R}_b^t \mathbf{f}_{ib}^b - \mathbf{R}^\top(\tilde{\mathbf{q}}) \mathbf{R}_b^t \mathbf{S}(\tilde{\mathbf{b}}_{\text{ars}}^b) \mathbf{f}_{ib}^b, \end{aligned} \quad (6.39)$$

similar to [47] and [70]. Hence, semiglobal exponential stability properties can be achieved as in the cited works.

6.5 Full-scale test setup

To verify the nonlinear observer with positional measurements from the PARS, we use the data from the experiment described in Section 5.5. In

addition to the PARS positioning information, data from several other sensors were recorded during this flight.

On-board the UAV, the payload is split into three parts; the on-board PARS, the experimental navigation stack, and the flight-critical avionics. This division is done to be able to easily move subsystems between different platforms, and the division is based on functionality: the PARS' primary function is to provide a communication link between the ground and the UAV payload, the navigation stack is responsible for providing a high-quality navigation solution for the UAV, and the avionics is responsible for all flight-critical functionality.

The SenTiStack consists of a Hardkernel Odroid XU4 [53] on-board computer, with a SenTiBoard, described in Chapter 3. The SenTiBoard reads and accurately records the timestamps of the incoming messages from a STIM 300 IMU [117] and a u-blox LEA-M8T GNSS receiver [136]. We use a PIXHAWK autopilot [91] with a 3DR GPS module containing a u-blox NEO-7N GPS receiver [135] and a Honeywell HMC5883L digital compass [58]. The barometer used in this paper is the PIXHAWK's integrated MEAS MS5611 [89].

For convenience, the data from the autopilot and navigation stack are synchronized using the GPS-time timestamps. To fully be able to operate without GPS coverage, this synchronization is not feasible, but receiving barometer and magnetometer data from the either from the autopilot's communication interface or through additional external sensors is a trivial alteration.

6.6 Results

6.6.1 Reference measurements

To evaluate the performance of the position estimates from the PARS aided NLO, an RTK GNSS solution was calculated. This solution has centimeter-level accuracy, which is sufficient to be considered a ground-truth when compared to the PARS NLO. The RTK GNSS solution is denoted as *RTK GNSS* in the figures and is shown with a green line.

The performance of the attitude observer is compared to the on-board autopilot's (the Pixhawk's) AHRS. Although the Pixhawk uses relatively low-cost sensors, it is well-tested, and provides an attitude solution which is independent from the PARS NLO. Note that this solution is not sufficiently accurate to be considered a ground truth, and we cannot say if the AHRS or the NLO performs better, but it should at least show the trends of the

system. The Pixhawk's AHRS solution is denoted as *Pixhawk AHRS* in the figures and is shown with a green line.

6.6.2 Performance metrics

The results statistics presented in this paper is based on three performance metrics:

- Absolute Mean Error (AME),
- Standard Deviation (STD) and,
- Root mean square (RMS) error

6.6.3 Raw PARS measurements

The raw range, bearing and elevation measurements are shown in Figure 6.4. Two file transfers disrupted the position measurements from 466 s - 498 s and 913 s - 1145 s, annotated in Figure 6.4 as A - B and C - D respectively. During the period from 0 s - 172 s, annotated as E - F, the UAV is circling near the ground antenna to ascend to cruising altitude. This causes the UAV to enter and exit the visible sector of the ground antenna, and when the UAV is outside this sector, the positioning does not work correctly. During the period from 2100 s - 2147 s, annotated as G - H, the UAV is landing and is also outside the visible sector of the UAV.

6.6.4 Results: aided INS

To compare the effect of the feedback-interconnection described in Section 6.4, two versions of the NLO were realized: one with feedback-interconnection turned on, and one with feedback-interconnection turned off. Both the observers were tuned equally. The gains for the NLO were chosen as $k_1 = 0.095$, $k_2 = 0.6$, and $k_I = 0.0007$, and the TMO was tuned as follows:

$$\begin{aligned} \mathbf{R}_{\text{PARS}} &= \text{diag}\left(7^2, 7^2, 0.0632^2\right), \\ \mathbf{Q}_{\text{PARS}} &= \text{diag}\left(0.0308^2 \cdot \mathbf{I}_3, 50 \cdot \mathbf{I}_3\right). \end{aligned}$$

As position measurements are missing in the intervals from 466 s to 498 s and 913 s to 1145 s, the filters rapidly drift off during these intervals, as they rely only on dead reckoning, which the observer is not tuned to handle. These intervals heavily skew the metrics for accuracy, and to the better represent the performance of the PARS, the results presented are

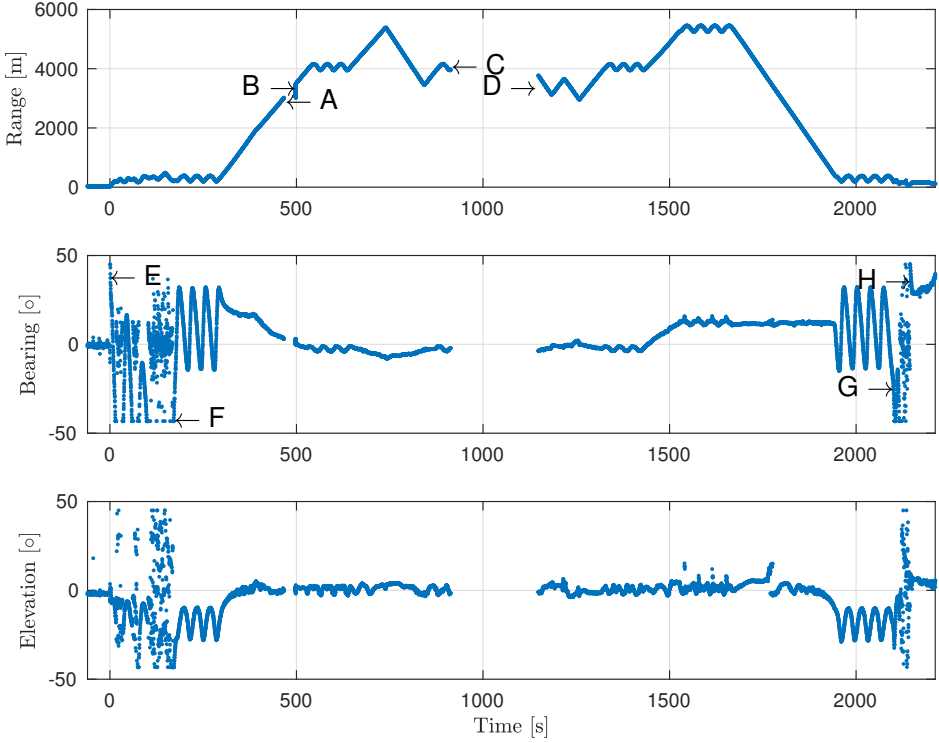


Figure 6.4: Raw PARS measurements.

filtered to exclude these intervals. The estimates are still used internally in the filter, but the intervals are removed from the statistics in Tables 6.1 to 6.2 and the plots in Figure 6.9.

As can be seen when comparing Figure 6.6 to Figure 6.7, the NLO solution without feedback interconnection give better results when there are no position measurements from the radio. This is expected as the feedback interconnection is more sensitive to loss of the aiding sensor. To compensate for this, an improvement was made to the filter, where it turns off the feedback interconnection if there has not been any position measurements in a certain amount of time (set to 1 s in these results).

For this experiment we only had access to a single ground antenna, and limited options for antenna positioning. Due to these limitations we could not cover the whole area of the mission, particularly around the take-off and landing areas. In the following statistics the period when the UAV was outside the visible frustum of the ground antenna are omitted to represent a more realistic performance of the system.

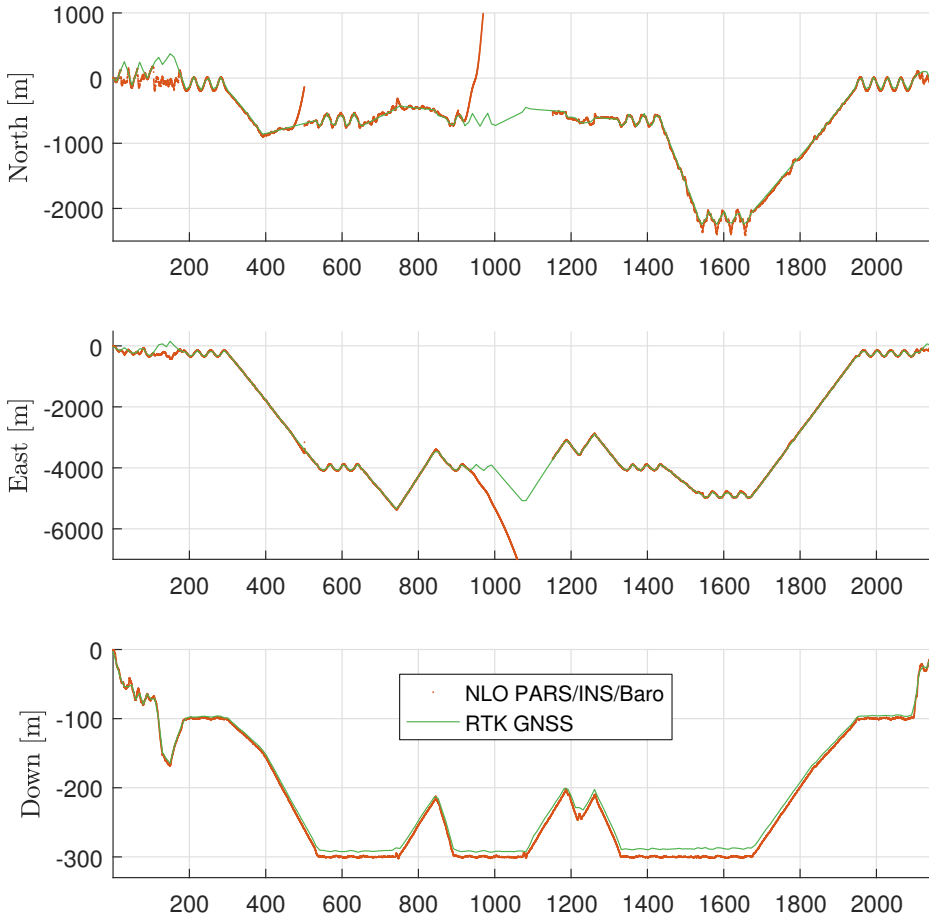


Figure 6.5: Position solution from NLO (with feedback interconnection).

From the comparisons given in Figure 6.9, and by comparing tables 6.1 and 6.2, we can see that the feedback interconnection does not significantly alter the performance of the position estimates, even when omitting the dead-reckoning parts of the flights. We see, however, that the estimates of both the pitch and roll are significantly improved by the feedback interconnection.

The velocity solution from the nonlinear observer and the Pixhawk's internal extended Kalman filter is shown in Figure 6.10.

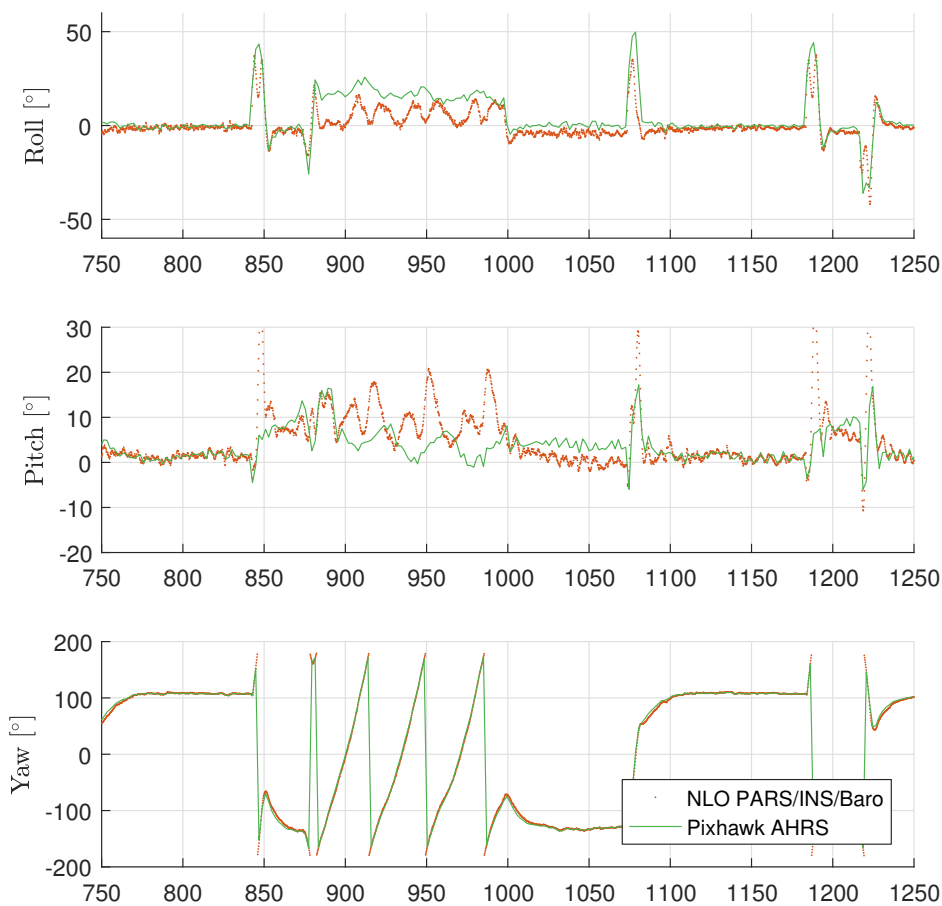


Figure 6.6: Attitude solution from NLO without feedback interconnection (position measurements missing from 913 s to 1145 s)

Table 6.1: PARS/BARO/INS (with feedback interconnection): Error statistics, compared to RTK and Pixhawk respectively

	North [m]	East [m]	Down [m]
AME:	16.78	6.86	7.17
STD:	23.34	9.00	3.56
RMS:	23.35	9.22	7.97

	Roll [deg]	Pitch [deg]	Yaw [deg]
AME:	2.67	2.70	1.98
STD:	4.02	4.04	3.41
RMS:	4.03	4.09	3.41

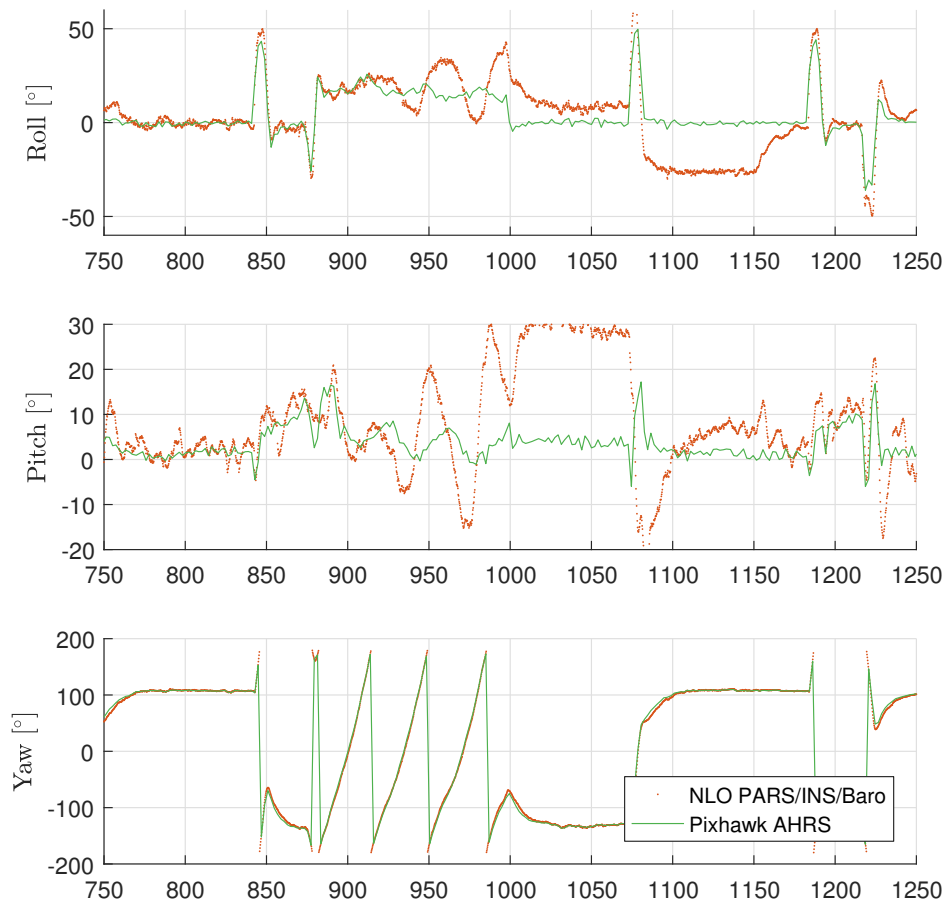


Figure 6.7: Attitude solution from NLO with feedback interconnection (position measurements missing from 913s to 1145s)

Table 6.2: PARS/BARO/INS (no feedback interconnection): Error statistics relative RTK and Pixhawk

	North [m]	East [m]	Down [m]
AME:	16.76	7.59	7.18
STD:	23.28	9.53	3.56
RMS:	23.29	9.81	7.97

	Roll [deg]	Pitch [deg]	Yaw [deg]
AME:	6.33	4.33	1.99
STD:	6.88	6.00	3.33
RMS:	8.69	6.25	3.33

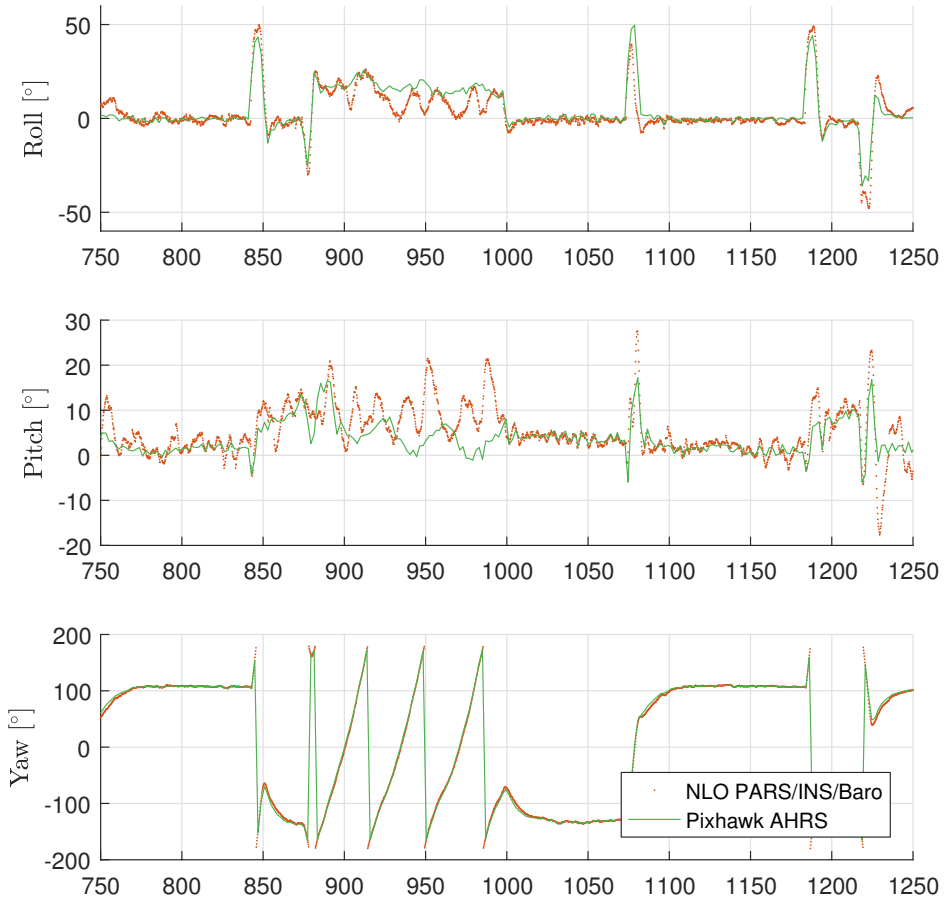
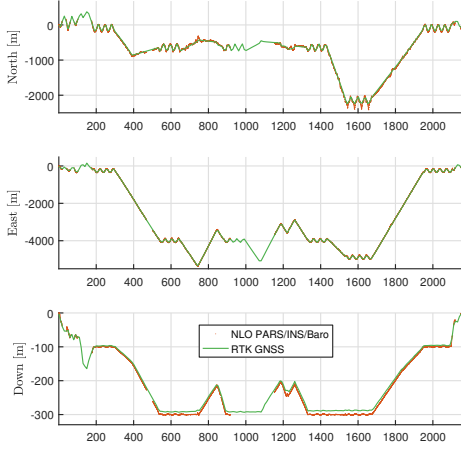
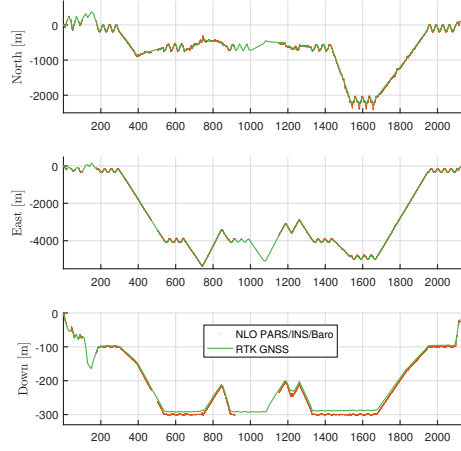


Figure 6.8: Attitude solution from NLO with thresholded feedback interconnection (position measurements missing from 913 s to 1145 s)

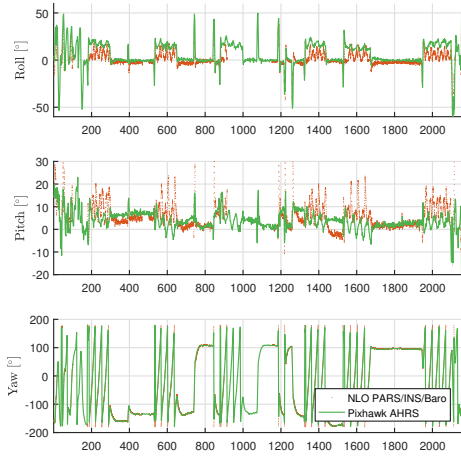
(a) Position estimate with no feedback-interconnection



(b) Position estimates with feedback-interconnection



(c) Attitude with no feedback-interconnection



(d) Attitude with feedback interconnection

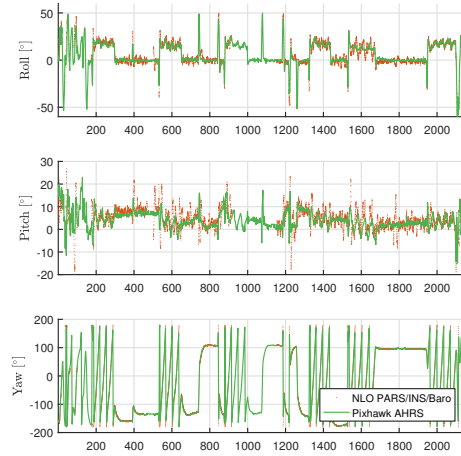


Figure 6.9: Comparison of RTK GPS with position estimates from the NLO and attitude estimates, with and without feedback interconnection. The results are filtered to exclude the periods without radio position estimates and the periods when the UAV is outside the visible region of the radio antenna.

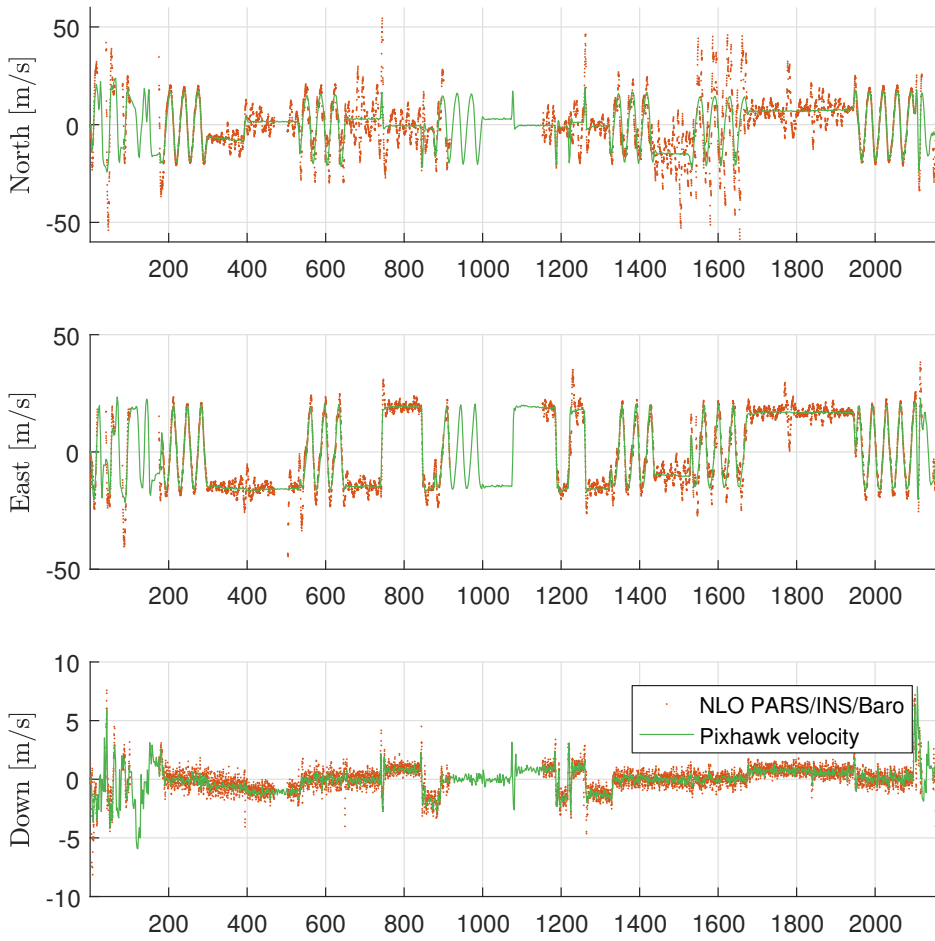


Figure 6.10: Velocity solution from NLO with feedback interconnection

6.7 Updated results

To further evaluate the PARS aided NLO, a secondary experiment was carried out. The experiment used identical sensor suite, except for updated software running PARS, and the DOA detection algorithm for position calculation of the PARS. For this dataset compensation for the skew mounting alignment error of the IMU was estimated which also increases the performance of the NLO.

Table 6.3: PARS/BARO/INS (with feedback interconnection) relative RTK GNSS/Pixhawk solution

	North	East	Down	Norm
AME:	8.34	4.53	3.19	10.01
STD:	15.77	4.09	0.78	16.31
RMS:	16.41	5.62	3.28	17.65

	Roll	Pitch	Yaw	Norm
AME:	2.18	3.63	0.90	4.33
STD:	3.60	3.05	2.87	5.52
RMS:	3.60	4.42	2.88	6.39

Table 6.4: PARS/BARO position solution

	North	East	Down	Norm
AME:	17.29	7.40	3.19	19.07
STD:	21.20	9.05	0.71	23.06
RMS:	24.72	10.60	3.27	27.09

When comparing the direct PARS/BARO solution presented in Table 6.4 to the PARS/BARO/INS solution presented in Table 6.3, we see a significant improvement in the position estimates, from 27.09m RMS with the PARS/BARO solution to 17.65m RMS with the PARS/BARO aided NLO solution. We also see a significant improvement in the attitude estimates with this dataset.

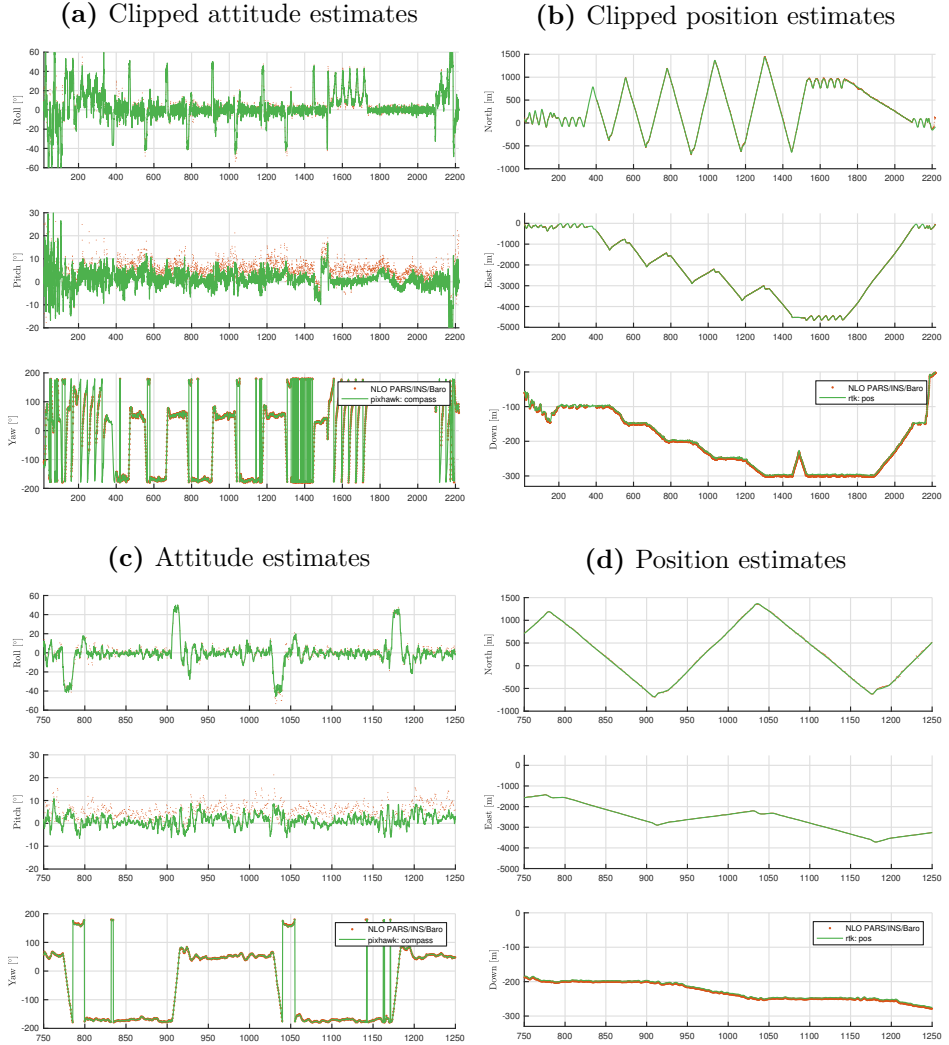


Figure 6.11: Results from the secondary experiment. The top figures show the full observer output and the bottom figures show a zoomed-in view of part of the data.

6.8 Summary

The nonlinear observer presented in this chapter fuses range and bearing measurements from the PARS with the measurements from the on-board inertial measurement unit, a magnetometer and a barometer. By aiding the INS with PARS measurements, compass readings, and altitude readings from a barometer, drift-free position, velocity and attitude estimates are obtained. In addition, the PARS measurements can be used for positioning alongside today's GNSS solutions, or as a redundant backup system running in parallel.

We have verified the performance of the PARS aided navigation filter through a BVLOS test experiment, by comparing the estimated position from the PARS aided navigation filter with real-time kinematic GNSS solution. The estimated attitude was compared with the autopilot's attitude estimates.

Compared to the RTK GNSS solution, we achieved an accuracy of approximately 26.3 m RMS on a flight with a maximal distance of 5.35 km from the ground station. Compared to the Pixhawk autopilot's AHRS solution, RMS attitude accuracy of approximately 4.0° , 4.1° and 3.4° were achieved in roll, pitch and yaw respectively.

In the second experiment, we improved the accuracy to RMS position error to 17.6 m on a 37 minute flight with a maximal distance of 4.75 km from the base station, with RMS attitude accuracy of approximately 3.6° , 4.4° and 2.9° for roll, pitch and yaw respectively.

Robust and Secure UAV Navigation Using GNSS, Phased-array Radio System and Inertial Sensor Fusion

This chapter is based on S. M. Albrektsen, T. H. Bryne, and T. A. Johansen. Robust and secure UAV navigation using GNSS, phased-array radio system and inertial sensor fusion. In *IEEE Conference on Control Technology and Applications (CCTA)*, Copenhagen, Denmark, August 21–24 2018.

7.1 Introduction

As previously mentioned, the most common solution for drift-free positioning has for several years been using global navigation satellite systems (GNSS). This solution has several attractive features: global coverage, lightweight receivers, and high accuracy - especially when using a real-time kinematic (RTK) solution. However, due to the low signal-to-noise ratio (SNR) of GNSS systems, these navigation solutions are prone to jamming [103] and spoofing [74], where the latter got much public attention in January 2016, when a US Navy patrol boat ended up in Iranian territorial waters [125]. In addition, a single error in either hardware or software can inhibit a GNSS positioning solution to work as intended. With more frequent UAV-usage in both the civil and military sector, being able to handle loss of GNSS positioning due to either errors or malicious sources becomes increasingly important.

A study of countermeasures to GNSS spoofing attacks is given in [67], where alterations to existing methods reduce the vulnerability to spoofing interference by monitoring the GNSS signal over time. These methods do,

however, not protect against jamming attacks and hardware failure. As a result, they are not sufficient as a redundant navigation system.

The proposed strategy for spoofing detection in this chapter is inherently dependent of the accuracy of the PARS. Since the positioning is based on both range and direction of arrival of the signal, signal reflections will be an issue. Alternative strategies than the one proposed in this paper can be based on GNSS equipment and advanced integrity monitoring techniques. In [105], the usage of two GNSS receivers for GNSS direction-of-arrival sensing combined with signal distortion detection is advised. The benefit of PARS in this respect, especially compared to civilian GNSS, is that in addition to having encrypted communication, PARS is based on signal direction-of-arrival detection, such than any spoofing attempts has to be carried out in the same sector that is covered by the PARS base antenna.

7.1.1 Main contribution

This chapter presents an inertial navigation system (INS) aided by a highly accurate RTK GNSS positioning solution when it is available, but also detects GNSS spoofing, based on the more secure, but less accurate, PARS positioning system. If GNSS spoofing is detected, PARS will be used as the aiding position sensor.

The solution consists of two observers running in parallel: the first observer using the RTK GNSS solution as the position reference, and the second observer using PARS and barometer measurements as the position reference. The estimates from these observers are inputs to a supervisor module that detects spoofing and provides the best available solution.

To verify the solution experimental data from a fixed wing, beyond visual line of sight experiment was used, with a simulated GNSS-spoofing attack added during post-processing.

7.1.2 Outline

In this chapter we start by introducing the GNSS RTK positioning system and necessary steps needed to improve the performance of the PARS as a positioning sensor in Section 7.2. We continue with presenting our non-linear observer for aided INS and spoofing supervisor in Section 7.3. An experiment was carried out, and a description of the system and hardware used is given in Section 7.4. The results from the experiment are presented in Section 7.5 and conclusive remarks are given in Section 7.6.

7.2 Positioning

7.2.1 PARS and barometer positioning

Similarly to Chapter 2 and Chapter 3 the vertical PARS position measurement based on the range and elevation angle is some times very noisy, likely due to reflections in the ocean surface. From Section 6.3.1 we have:

$$\mathbf{p}_{\text{PARS}}^r = \begin{pmatrix} p_{rb,x}^r \\ p_{rb,y}^r \\ p_{rb,z}^r \end{pmatrix} = \begin{pmatrix} \rho_u \cos(\psi_u) \cos(\theta_u) \\ \rho_u \sin(\psi_u) \cos(\theta_u) \\ -\rho_u \sin(\theta_u) \end{pmatrix}. \quad (7.1)$$

To improve on this, the vertical measurement in (7.1) is replaced with an altitude measurement based on barometric pressure $\gamma_{\text{BARO},m} = p_{\text{BARO},z}^r + \varepsilon_{\text{BARO}}$. Moreover, to prevent the elevation angle measurement to affect the horizontal positioning, the PARS range was altitude compensated using the barometer measurement

$$\bar{\rho}_m = \sqrt{\rho_m^2 - \gamma_{\text{BARO},m}^2}, \quad (7.2)$$

where $\bar{\rho}_m$ then becomes a measurement of horizontal range

$$\bar{\rho}_u = \sqrt{(p_{rb,x}^r)^2 + (p_{rb,y}^r)^2} \quad (7.3)$$

, cf. Figure 6.2, such that Cartesian position measurement becomes,

$$\mathbf{p}_{\text{PARS,BARO}}^r = \begin{pmatrix} \bar{\rho}_m \cos(\psi_m) \\ \bar{\rho}_m \sin(\psi_m) \\ \gamma_{\text{BARO},m} \end{pmatrix}, \quad (7.4)$$

decoupling the horizontal components of the measurement $\mathbf{p}_{\text{PARS,BARO}}^r$ from the elevation angle measurement. The covariance of $\mathbf{p}_{\text{PARS,BARO}}^r$, denoted $\bar{\mathbf{R}}(t)$, is based on the original measurements, ρ_m , ψ_m and $\gamma_{\text{BARO},m}$ and their covariance $\mathbf{R}_{\text{PARS}}(t) = \text{diag}(E[\varepsilon_\rho^2], E[\varepsilon_\psi^2], E[\varepsilon_{\text{BARO}}^2])$.

$\bar{\mathbf{R}}(t) = \mathbf{M}(t) \mathbf{R}_{\text{PARS}}(t) \mathbf{M}^\top(t)$ is derived through linearization [15, Ch. 1.6] by assuming that the noise is Gaussian, where $\mathbf{M}(t)$ is the Jacobian of $\mathbf{p}_{\text{PARS,BARO}}^r$ w.r.t. to the noise $\boldsymbol{\varepsilon} = (\varepsilon_\rho \quad \varepsilon_\psi \quad \varepsilon_{\text{BARO}})^\top$. This results in:

$$\mathbf{M}(t) = \frac{\partial \mathbf{p}_{\text{PARS,BARO}}^r}{\partial \boldsymbol{\varepsilon}} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{pmatrix} \quad (7.5)$$

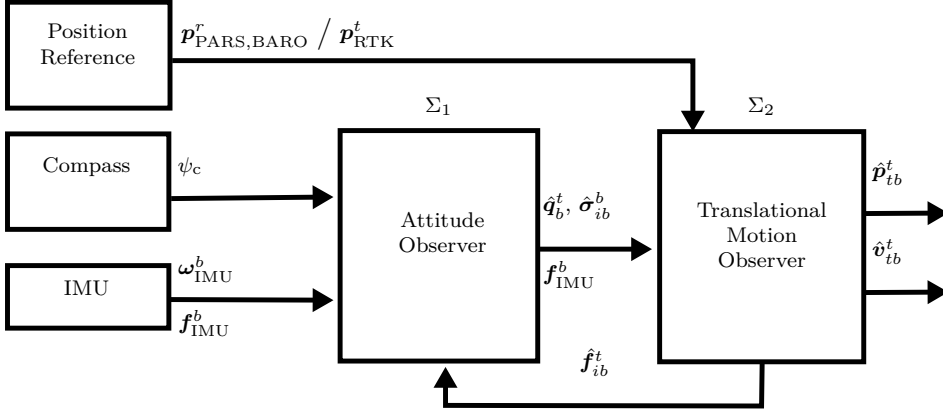


Figure 7.1: Feedback-interconnected observer framework. The observer design is based upon integration of ARS and ACC measurements. The compass measurement ψ_c is used as a aiding heading measurements in the nonlinear observer. In addition, the two position references $\mathbf{p}_{\text{PARS,BARO}}^r$ or $\mathbf{p}_{\text{RTK}}^t$, given in the $\{r\}$ and $\{t\}$ frames, are used to aid the translational motion observer.

with

$$\begin{aligned} m_{11} &= \frac{\cos(\psi_m)\rho_m}{\bar{\rho}_m} & m_{21} &= \frac{\sin(\psi_m)\rho_m}{\bar{\rho}_m} \\ m_{12} &= -\sin(\psi_m)\bar{\rho}_m & m_{22} &= \cos(\psi_m)\bar{\rho}_m \\ m_{13} &= -\frac{\cos(\psi_m)\gamma_{\text{BARO},m}}{\bar{\rho}_m} & m_{23} &= -\frac{\sin(\psi_m)\gamma_{\text{BARO},m}}{\bar{\rho}_m}. \end{aligned}$$

In addition, the measurement (7.4) can be given directly in the $\{t\}$ frame by taking $\mathbf{p}_{\text{PARS,BARO}}^t = \mathbf{R}_r^t \mathbf{p}_{\text{PARS,BARO}}^r$ and transforming the corresponding covariance to the $\{t\}$ frame using

$$\bar{\mathbf{R}}(t) = \mathbf{R}_r^t \mathbf{M}(t) \mathbf{R}_{\text{PARS}}(t) \mathbf{M}^\top(t) \mathbf{R}_t^r. \quad (7.6)$$

7.3 System implementation

7.3.1 NLO and TMO observers

The observer design consist of two feedback-interconnected observers, similar as in Chapter 6, as depicted in Figure 7.1. The first observer is a nonlinear attitude observer, estimating attitude and the ARS bias. The second observer is a translational motion observer estimating position, velocity and specific force given in the navigation frame. These observers are

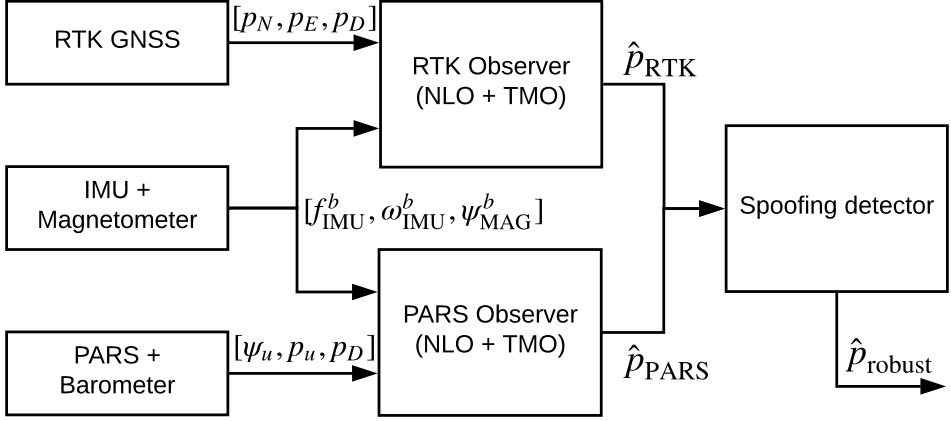


Figure 7.2: Spoofing prevention system overview.

referred to as the NLO and TMO respectively in this paper. The observer framework is structurally the same as in [47], where the attitude between the $\{b\}$ and the $\{e\}$ frame was estimated.

The error dynamics of the feedback interconnected observer is uniformly semiglobal exponentially stable, under certain conditions, with positive definite matrices \mathbf{Q} and $\bar{\mathbf{R}}$, see [47] for details.

To account for any colored noise in the PARS measurements, additional states can be introduced such as in [102]. By choosing appropriated error states and corresponding models, the gain of the TMO can use a time-scaled Riccati equation that can account for the additional uncertainty when propagating the covariance[20].

7.3.2 Spoofing detector

An overview of the spoofing prevention system is given in Figure 7.2. The concept of the spoofing detector is the base assumption that the PARS as a positioning system cannot easily be spoofed due to:

- The high signal-to-noise ratio
- Only signals originated inside of the visible sector of the radio are considered in the DOA calculation
- The communication on the PARS-network is encrypted, ensuring the sender's origin

There are, however, a different set of challenges with the PARS, such as reflections and inaccuracies in the algorithm used for detecting the incoming

angles.

To determine if $\mathbf{p}_{\text{RTK}}^t$ is subjected to a malicious attack, we suggest using one of two detection methods. The first suggestion is based on a Kalman filter and the second is based on position compared to estimated covariance.

7.3.3 Kalman filter detection

The first suggested method is a supervisor module which is based on the following state-space model and measurement vector:

$$\dot{\mathbf{x}}_1 = \mathbf{x}_2 \quad (7.7)$$

$$\dot{\mathbf{x}}_2 = \mathbf{w} \quad (7.8)$$

$$\mathbf{z} = \mathbf{x}_1 = \hat{\mathbf{p}}_{tb,\text{PARS}}^t - \hat{\mathbf{p}}_{\text{RTK}}^t \quad (7.9)$$

used to estimate the difference between position estimates of the INSs with RTK and PARS aiding, respectively. \mathbf{w} is some Gaussian process noise. This results in the Kalman filter:

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{K}_s(\mathbf{z} - \mathbf{H}\mathbf{x}), \quad (7.10)$$

$$\mathbf{K}_s = \mathbf{P}_s \mathbf{H}^\top \mathbf{R}_s^{-1} \quad (7.11)$$

$$\dot{\mathbf{P}}_s = \mathbf{F}\mathbf{P}_s + \mathbf{P}_s \mathbf{F}^\top - \mathbf{K}_s \mathbf{R}_s \mathbf{K}_s^\top + \mathbf{E} \mathbf{Q}_s \mathbf{E}^\top \quad (7.12)$$

where

$$\mathbf{F} = \begin{pmatrix} \mathbf{0}_{3 \times 3} & \mathbf{I}_3 \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} \end{pmatrix}, \mathbf{H} = \begin{pmatrix} \mathbf{I}_3 & \mathbf{0}_{3 \times 3} \end{pmatrix}, \mathbf{E} = \begin{pmatrix} \mathbf{0}_{3 \times 3} \\ \mathbf{I}_3 \end{pmatrix}. \quad (7.13)$$

Concerning the tuning of the KF, we define measurement error covariance:

$$\mathbf{R}_s := \mathbf{P}_{\text{PARS,pos}} + \mathbf{P}_{\text{RTK,pos}}, \quad (7.14)$$

where $\mathbf{P}_{\star,\text{pos}}$ indicates the upper left three by three matrix of the full covariance matrix \mathbf{P}_\star , obtained from the two TMOs in Chapter 6. \mathbf{Q}_s is consider a tuning matrix chosen dependent on how responsive one would like the filter to be. The RTK measurement is considered to be spoofed if the output of the RTK-aided INS differ from that of the INS aided by PARS. The INS position to the user is then given by the module:

$$\hat{\mathbf{p}}_{\text{robust}}^t = \begin{cases} \hat{\mathbf{p}}_{tb,\text{RTK}}^t & \text{if } \|\mathbf{x}_1\| < \theta_{\text{threshold}}(t), \\ \hat{\mathbf{p}}_{tb,\text{PARS}}^t & \text{otherwise,} \end{cases} \quad (7.15)$$

where $\theta_{\text{threshold}}(t)$ is a user-defined threshold. This threshold should typically vary with the distance from the PARS to the UAV as the uncertainty of the PARS position measurements in Cartesian coordinates increases proportionally with the range.

7.3.4 Covariance based detection

The second method is based on a twofold strategy:

- Monitoring the discrepancy between the two observer outputs
- Monitoring the residuals of the two observers

The discrepancy between the observer is monitored using

$$T_{\text{RTK,PARS}} = \tilde{\mathbf{p}}^\top (\mathbf{P}_{\text{PARS}} + \mathbf{P}_{\text{RTK}})^{-1} \tilde{\mathbf{p}} \sim \chi_3^2. \quad (7.16)$$

The observer position difference is given as $\tilde{\mathbf{p}} = \hat{\mathbf{p}}_{tb,\text{PARS}}^t - \hat{\mathbf{p}}_{tb,\text{RTK}}^t$ resulting in $T_{\text{RTK,PARS}}$ being chi-squared distributed, [50]. Combining this with using e.g. a CUSUM [101] algorithm to monitor the mean of the residual in the given observers one can conclude that GNSS is spoofed if

1. The mean of $\tilde{\mathbf{y}}^t$ in the RTK aided observer changes
2. The mean of $\tilde{\mathbf{y}}^t$ in the PARS aided observer does not change
3. $T_{\text{RTK,PARS}}$ surpasses the critical value of $\chi_{\alpha,3}^2$

where α is the significance value of choice. This results in the default (\mathcal{H}_0) and alternative hypothesis (\mathcal{H}_1)

$$\begin{aligned} \mathcal{H}_0 : \mathbf{p}_{\text{RTK}}^t & \text{ is a valid position measurement.} \\ \mathcal{H}_1 : \mathbf{p}_{\text{RTK}}^t & \text{ is subjected to a spoofing attempt.} \end{aligned}$$

7.3.5 Output smoothing

To prevent switching back and forth between the two INS's when the difference between the sensors are around the threshold value, hysteresis functionality can be implemented. This can both be done on the detection level, to avoid that spikes in the observer's state estimates too strongly influences the detector, and on the output level to guarantee that the spoofing attack has terminated before switching back. Moreover, since the position, velocity and attitude estimates will be considered by the autopilot, the output of the supervisor can be weighted when switching between the two hypotheses to prevent steps in the INS data. This can be done using exponential function such that

$$\hat{\mathbf{p}}_{\text{robust}}^t = \left(1 - e^{-\alpha(t-t_0)}\right) \hat{\mathbf{p}}_{tb,\text{PARS}}^t + e^{-\alpha(t-t_0)} \hat{\mathbf{p}}_{tb,\text{RTK}}^t, \quad (7.17)$$

when switching between the RTK-based INS and the PARS-based. α is a tuning variable and t_0 is the time of the switch between $\hat{\mathbf{p}}_{tb,\text{RTK}}^t$ and $\hat{\mathbf{p}}_{tb,\text{PARS}}^t$ as the module output.

7.4 Experimental results

To verify the validity of the observer and supervisor, an experiment was carried out on December 14th, 2017 in good, although cold, weather conditions at Agdenes outside of Trondheim, Norway. A flight of 37 min with a Skywalker X8 UAV was performed. The payload was identical to the one described in Section 6.5, but some software updates were made to improve the system performance.

7.4.1 Ground station

To pilot the UAV, register base GNSS data for the RTK solution, and calculate the PARS positioning data, a ground station was set up. This consisted of a laptop computer, a uBlox M8T GNSS receiver, and a Radionor Communications CRE2 189 PARS. The CRE2 189 radio consists of a 60 antenna elements, and the PARS system was set to operate in a 15 Mbit/s mode, providing a maximal distance of up to 15 km. The PARS modules communicate in the 5 GHz band.



Figure 7.3: Track of the UAV flight, based on RTK GNSS position.

7.4.2 RTK Spoofing

To simulate a spoofing process of the RTK positioning, we imagine that a malicious agent wants to create an offset to the on-board-calculated position of the UAV. If we assume that the intended landing area is close to the take-off area, and the malicious agent observes the take-off of the UAV, the agent can calculate a position offset that would deceive the positioning system into flying the UAV to a different area when it returns for landing. In this scenario the malicious agent can slowly adjust the offset to make it difficult to detect the spoofing process by using inertial sensors.

7.5 Results

7.5.1 Spoofing setup

In this experiment we set up the RTK-spoofing procedure to initiate 907.6 s after takeoff and create an offset to the RTK-signal, increasing with 0.1 m per sample at a sample rate of 10 Hz, resulting in a velocity change of 1 m s^{-1} until an offset of -1000 m in both East and North direction is reached. Furthermore, we imagine that after the malicious agent discovered that the GNSS-spoofing attempt was ineffective, the spoofing attempt is terminated after being active for 1000 s, at 1907.5 s after take-off. This allows us to show the automatic recovery-feature of the supervisor filter.

Figure 7.4 illustrates the observer outputs from the observer using real RTK measurements and the observer using spoofed RTK measurements. It also indicates when the GNSS-spoofing is active. The spike that occurs on the spoofed input at the end of the spoofing period is due to inaccurate velocity estimates which are induced due to the large correction in position when the position measurement snaps back.

7.5.2 Tuning

Both the RTK and PARS aided NLOs and TMOs were tuned equivalent with the exception of the \mathbf{R} matrices in the TMOs. The \mathbf{Q} and \mathbf{R}_\star matrices were set as follows:

$$\begin{aligned}\mathbf{Q} &= \text{diag} \left(0.0308^2 \cdot \mathbf{I}_3, 50 \cdot \mathbf{I}_3 \right) \\ \mathbf{R}_{\text{PARS}} &= \text{diag} \left(5^2, (2^\circ)^2, 0.5^2 \right) \\ \mathbf{R}_{\text{RTK}} &= \text{diag} \left(0.05^2, 0.05^2, 0.1^2 \right)\end{aligned}$$

Table 7.1: Detector key-figures. Time measurements are given as seconds after take-off and distance measurements are given as the 2-norm of the difference from the output from the observer with the non-spoofed RTK position reference.

	Start [s]	End [s]	Offset [m]	End offset [m]
Actual:	907.60	1907.50	0.00	1414.35
Detector:	932.89	1910.89	35.97	22.98

The \mathbf{Q} matrix of the first supervisor implementation was chosen as

$$\mathbf{Q}_s = \text{diag}(1, 1, 0.5), \quad (7.18)$$

and the spoofing-threshold, $\theta_{\text{threshold}}(t)$, was set to the largest value of 20 m and $0.015 \cdot \rho_m$ at the given timestep. Furthermore we added a hysteresis function so that three consecutive position measurements must be classified as inaccurate before the supervisor evaluates that the system is being spoofed and outputs the more robust measurement. For the second method, based on covariance a threshold value of 18^2 m^2 was chosen.

7.5.3 Performance metrics

To compare the performance of the different filter setups the following metrics are used: Absolute mean error (AME), standard deviation (STD), and root mean squared (RMS).

The absolute mean error is the sum of the absolute difference between the estimate and the reference value, divided by the number of samples. The standard deviation is the square root of the variance of the set, where the variance is the expected value of the squared deviation from the mean. The root mean squared value is the root of the mean of the squared of each of the points in the data set.

Figure 7.5 plots the output of the detector with both RTK measurements and spoofed RTK measurements as references. A line indicating when spoofing is detected is also given. Table 7.1 lists the actual starting and ending times of the RTK spoofing, and the offset at the given times. The period when spoofing is detected, and the distance between the RTK and spoofed RTK observers at these times are also given in this table.

In Tables 7.2 and 7.3 the performance metrics for the RTK observer and the PARS observer when they are compared to the RTK measurements are listed. An error-plot of the detector output compared to the RTK measurements is given in Figure 7.6.

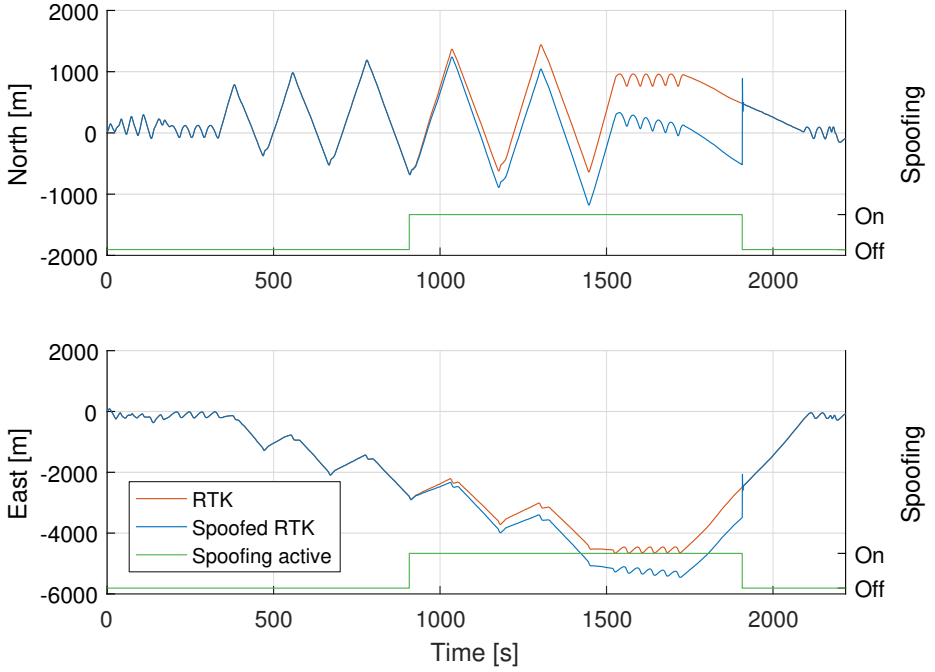


Figure 7.4: RTK filter output with and without spoofing. The green line indicates when the spoofing is active.

7.5.4 Discussion

From the results provided in this section we see that when the spoofing attack is recognized by the detector as intended as the magnitude of the spoofed offset becomes significant. Furthermore we see that the system output of the detector changes according to the estimated spoofed state, and that it recovers as intended. The computational overhead of using these detection methods is primarily due to running the two observers in parallel, as the supervisor module is lightweight.

As illustrated in Figure 7.7, the spoofing observer also handles a spoofing attack with a slower change in position. We can see that the time before the attack can be detected by the filter increases, which is expected as the impact of the attack is less severe. However, as the offset increases, this spoofing is also detected. The time before asserting a detection can be reduced by lowering the threshold value, but this will increase the likelihood of false positives.

Although the results in this paper looks promising, more testing should

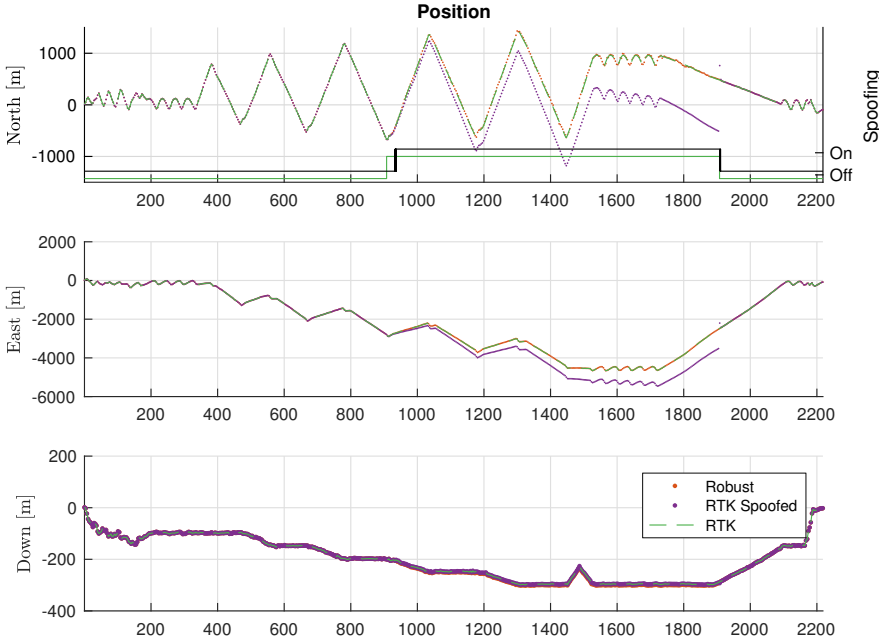


Figure 7.5: Output of the detector plotted against spoofed filter output. The dashed green line is the un-spoofed RTK solution provided as a reference. The black 'spoofing' line shows when the filter detects a spoofing attempt, while the solid green line shows when the spoofing is active.

be done before concluding on the reliability of the PARS as a position sensor. The occurrence of an errors, possibly due to reflections of the PARS signal, may be more likely than a spoofing attack, especially in short intervals where the PARS measurements are inaccurate.

7.6 Concluding remarks

In this chapter we have implemented a drift-free positioning system that automatically detects and handles spoofing of GNSS positioning solutions. By using a phased array radio system (PARS), which is robust against malicious attacks such as spoofing and jamming, as a secondary position reference, security of the UAV can be maintained, even during spoofing attacks on the UAV.

An experiment was carried out and a simulated spoofing attempt was

Table 7.2: PARS observer performance compared to RTK measurements

	North [m]	East [m]	Down [m]	Norm [m]
AME:	7.76	4.48	3.18	9.51
STD:	10.35	3.55	0.78	10.97
RMS:	11.00	5.31	3.28	12.65

Table 7.3: RTK observer performance compared to RTK measurements

	North [m]	East [m]	Down [m]	Norm [m]
AME:	0.02	0.02	0.03	0.04
STD:	0.03	0.02	0.04	0.06
RMS:	0.03	0.02	0.05	0.06

added in software. With our defined threshold the spoofing attempt was first detected when the RTK estimate was approximately 36 m away from the expected value. Furthermore, we showed that the supervisor automatically recovered to the high-quality position estimate when the spoofing attempt was terminated.

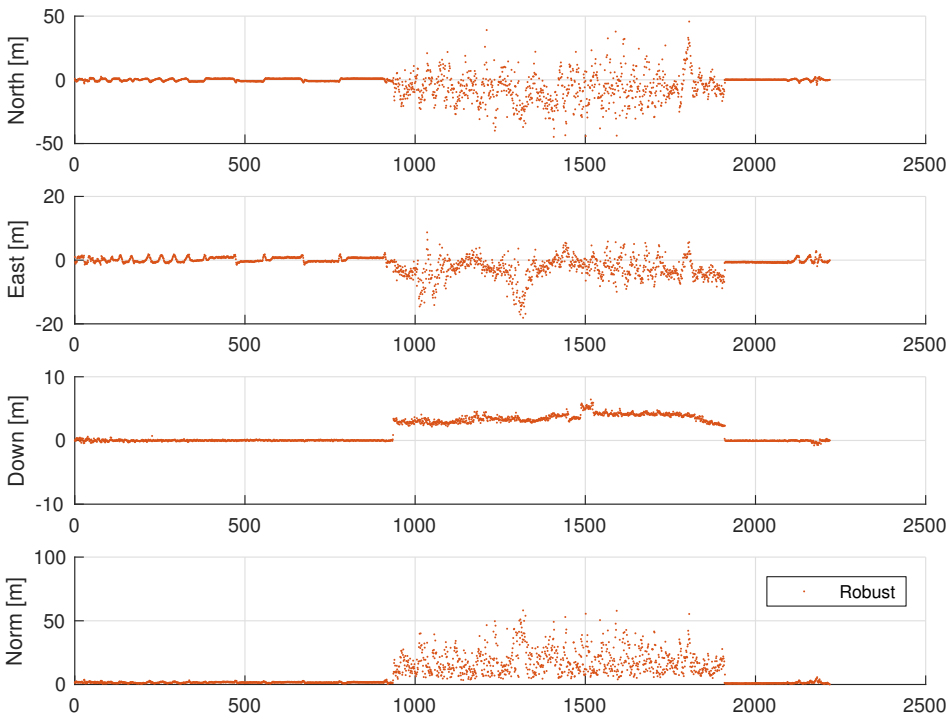


Figure 7.6: Error of the detector output compared to RTK measurements.

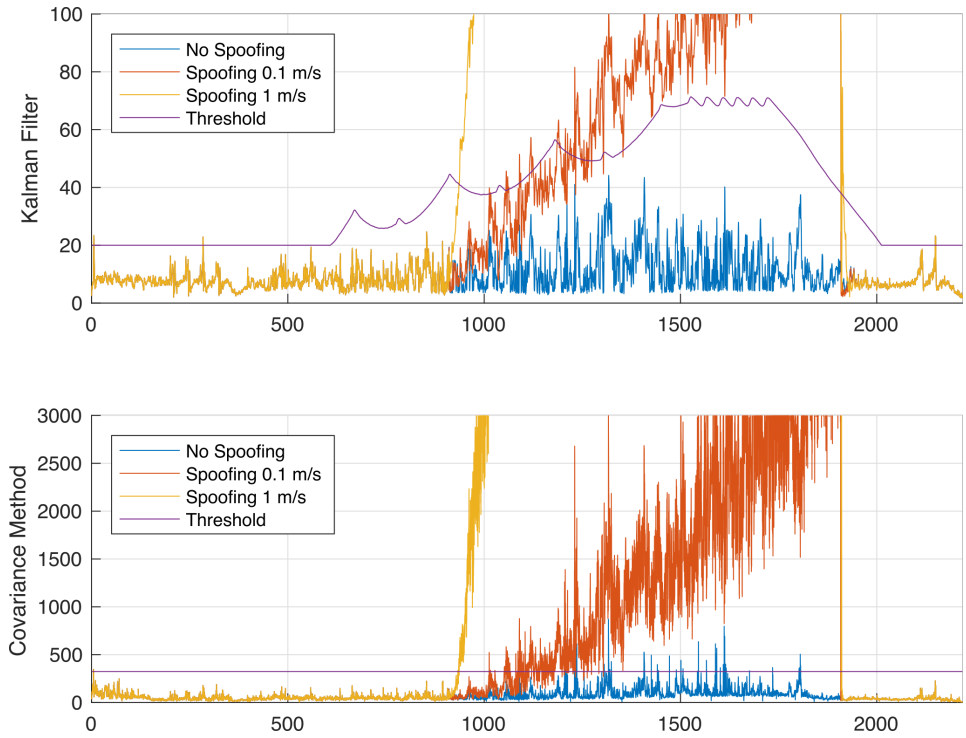


Figure 7.7: Detector calculations and threshold. A spoofing attack with an offset-velocity of 0.1 m s^{-1} is added to show the difference in behavior.

8.1 Summary and conclusions

The motivation behind this thesis was twofold. First, to create a well-tested and field-proven system for high-accuracy sensor timing and synchronization; second, to develop navigation solutions that can operate in GNSS-denied environments.

In the first part of this thesis, requirements for a high-accuracy hardware sensor timing system were presented. To utilize unmanned aerial vehicles' desirable features for collecting high-quality sensor data in complex tasks, hardware synchronization of sensor data is needed. In most remote sensing tasks, the sensor measurements must be associated with the position and attitude of the UAV. As UAVs typically have high velocities and fast system dynamics, errors in sensor messages' associated timestamps will result in inaccuracies in the associated position and attitude estimates, which again lead to inaccurate sensor measurements. In this thesis, a reconfigurable system that minimizes these unknown temporal errors from sensors has been introduced. By allowing system integrators to reconfigure the sensor timing system, the payload can be altered without redesigning and testing the hardware and firmware of the timing system.

Two implementations, the SyncBoard and its successor, the SenTiBoard, were created, tested with mock-implementations in the laboratory and tens of experiments in the field. The SenTiBoard has been shown to have an RMS accuracy of $1.90\text{ }\mu\text{s}$ drift per second, and by using the solution, a geo-referencing error was improved by a factor of four. Both solutions simplify sensor integration and maintenance of sensor payloads, which has been verified by implementation in several different sensor platforms, with a variety

of sensors.

In addition to the hardware sensor timing systems, a UAV navigation system has been presented, and together with the timing systems these have been implemented with GNSS-receivers, IMUs, magnetometers, an air-pressure/data probe, phased array radio systems, RGB cameras, IR cameras, a hyperspectral camera, laser altimeters, UAV autopilots and other specialized sensor systems, and these setups have been used on UAVs, USVs and manned aircraft.

The second part of this thesis discusses navigation in GNSS-denied environments and handling loss of GNSS due to failures or malicious intentions. As UAVs become increasingly available for civilian use, and adopted by industry, the risk associated with UAV operations increases. Most systems today are critically dependent on GNSS to operate safely, and GNSS is highly susceptible to malicious attacks, due to the weak signal strength. To reduce the requirement of GNSS availability, this thesis suggests two different approaches for operating in GNSS-denied environments; one based on camera measurements, and one based on phased array radio system navigation. The latter method is explored in three stages: with only PARS measurements and a barometer; with PARS measurements, a barometer and an IMU; and a hybrid approach where the solution fusing PARS, IMU and a barometer is used as a redundant alternative to RTK GNSS. When using these methods, GNSS is no longer a single point of failure.

The first chapter in this part discusses a throw-and-go UAV which uses inertial optical flow and an IMU to stabilize after being thrown in the air. To be able to analyze the data sufficiently quickly in such situations, optimizations were made to increase the performance of the system. An experiment was carried out where the UAV was thrown in the air and successfully stabilized at the desired height.

To provide GNSS-less navigation at long-distance, long-endurance missions, a PARS was used to provide absolute position measurements. Experiments were performed, and first only the PARS and a barometer was used, then this solution was used to aid an NLO to provide PVA estimates. With the PARS and barometer only, an RMS accuracy of 27 m was achieved, and using the NLO this was improved to approximately 18 m RMS and further reduced to 13 m by implementing more advanced covariance handling.

Finally, the NLO solution was used as a countermeasure to GNSS-spoofing. A system that automatically detects and handles spoofing of GNSS was created and the best solution available was chosen by the system. Using this strategy, the high-accuracy GNSS solution can be used if no spoofing is detected, and the PARS solution which is more robust to

malicious intents can be used otherwise. The system is created so that it automatically recovers when a spoofing attack is discontinued.

With the solutions presented in this thesis, high-accuracy GNSS sensors solutions can easily be integrated, and using the robust and secure navigation sensor, a combined approach delivering the best from the two systems is provided.

8.2 Recommendations for further work

To minimize the hardware required for navigation estimation when using the hardware sensor timing system, one could omit the onboard computer and store the data on an SD-card or other onboard storage. To further increase the usability of this sensor logging system, a sensor fusion implementation, such as a multiplicative extended Kalman filter (MEKF), that provides position, velocity and attitude estimates of the UAV could be implemented and synchronized with the sensor data. Similarly, the calculations for the RTK GNSS solution could be performed onboard the SenTiBoard. This is useful in online applications with minimal sensor configurations, as the position estimates are highly improved. To be able to perform this, the sensor data must be parsed by the timing board, and the measurements must be integrated. If these tasks are too computationally demanding for a microcontroller to perform, solutions for FPGAs could be implemented and integrated on the hardware sensor timing system.

It would also be interesting to produce a hardware sensor timing system with some of the core navigation sensors integrated on the board directly. This would simplify integration of sensor fusion algorithms, as the sensors and the data formats are known by the firmware, and thus the sensors can use the sensor measurements.

To support a wider range of sensor combinations, another improvement to a hardware sensor timing system would be to implement a modular design for the communication ports. In such a design, instead of having protocol transceivers such as the MAX3225 and MAX3362 on the sensor timing hardware, the UART ports can be exposed directly through a specified interface. Through this interface such transceivers could be added on an as-need basis with a modular connector. This would allow the user to freely choose how many RS232, RS422 or other protocols that can be interfaced through UART as needed, and which connectors to use.

To verify that the accuracy of the navigation system based on phased array radio is sufficient to perform a navigation task in a real-life scenario, this should be tested. Although the navigation system shows promising

results, it has not yet been integrated with an autopilot and flown using this as the navigation source. To be able to do so, safety precautions, such as being able to switch to GNSS in case of an emergency should be implemented as well.

To be able to rely on the vertical component of the radio measurements, the accuracy needs to be improved significantly. This inaccuracy should be further investigated to verify if it is due to reflections by the ocean surface. If both the actual direction and the reflected signals are detectable, the algorithm which calculates bearing and elevation angles from the incoming signals, can be altered to provide one or more alternative measurements, instead of only the best-guess of the signal. With such an alteration, techniques such as Multiple Hypothesis Tracking (MHT) can be used to make sure that the real signal is correctly tracked by the navigation system.

It would also be interesting to make a flight with multiple ground radios that observe the UAV from different positions on the ground. This would in theory not only improve the position estimates by having more independent readings, but also allow a larger area to be visible by the radios. This configuration will also allow us to verify the error sources of the radio system, such as reflections from the ocean surface. One might also imagine a hand-off scenario where when the UAV leaves a sector covered by one radio, the second radio provides the position estimates.

To improve the position estimates from the radio observer, an implementation that compensates for the curvature of the Earth should be implemented. The implementations presented in this thesis assume that the Earth is flat, which is an assumption that is accurate when covering small areas. When covering larger distances, however, this inaccuracy increases. We want to adapt the filter to handle a non-flat Earth model, to compare the effect of using different models when flying missions of tens of kilometers.

Finally, to be able to use the PARS as a spoofing detector in a real-life scenario, further testing and robustification of the direction-of-arrival detection should be performed. PARS-navigation is a new technology, especially when compared to GNSS, and it is more prone to outliers from for example reflections.



Appendix

A.1 Payload power control board

An important feature of the non-avionics part of the payload is to be able to power off the system in case of an emergency. In one of our payloads, using the Piccolo SL Autopilot, we wanted to integrate this feature with Piccolo's command center software, so we could control the payload from the command center. Unfortunately, due to the limited number of general purpose input/output (GPIO) pins controllable from the command center, no pins were available to us for this purpose. There are, however, 1000 waypoints configurable directly from the interface, and each of these waypoints have a User Byte field which can be set to a user configurable value.

Thus, to be able to control the power output of the non-avionics part of the payload one can create a power control board with four connections: power in, power out, communication in, and communication out. Both the power in and the power out connectors need to have at least three pins: positive voltage, ground and an enable-pin which can be used to switch a relay, or an external DC/DC converter, on or off. The power-in connector has the same pins, but the enable-pin is connected to ground. By providing identical connectors one can easily add or remove the power control board as needed.

To be able to control the power from the ground station, the board needs to monitor the communication between the ground station and the autopilot. This can be done by "wiretapping" the transmission line from the telemetry link onboard the UAV to the autopilot, by connecting the communication channel from the telemetry link to the power control board and from the power control board to the autopilot. As the power control board

does not transmit any data on the communication channel, the communication link can go directly through the board with only passive connections to the communication lines. Like the power connectors, the communication connectors should be identical, to allow an operator to easily add or remove the board as needed.

When the communication can be monitored, the board can read the waypoint messages sent from the ground station to the autopilot, including the configurable User Byte. By assigning special values to different values of the User Byte different commands can be sent to the board. Such commands can be *switch power on*, *switch power off*, or *restart*.

A board with the aforementioned functionality was implemented using an Atmel ATtiny84 microprocessor with dimensions 42 mm x 42 mm. A fuse was also added in-between the live wires of the power lines and a DC/DC low-dropout regulator (LDO) voltage regulator to provide the appropriate voltage for the microprocessor. The board supports either an RS-232 connection or a TTL-level connection. Schematics and PCB layout of the payload power control board are given in Figure A.1 and Figure A.2.

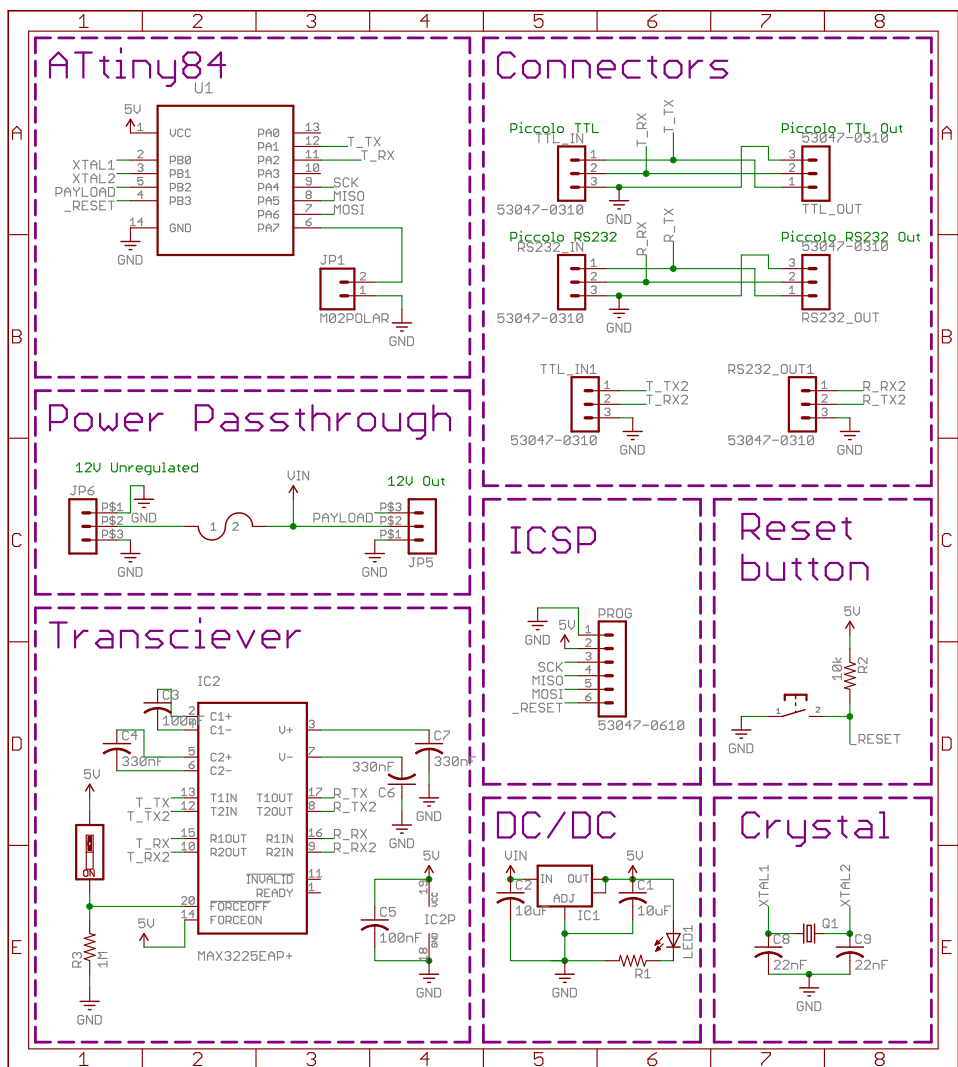


Figure A.1: Schematics of the payload power control board

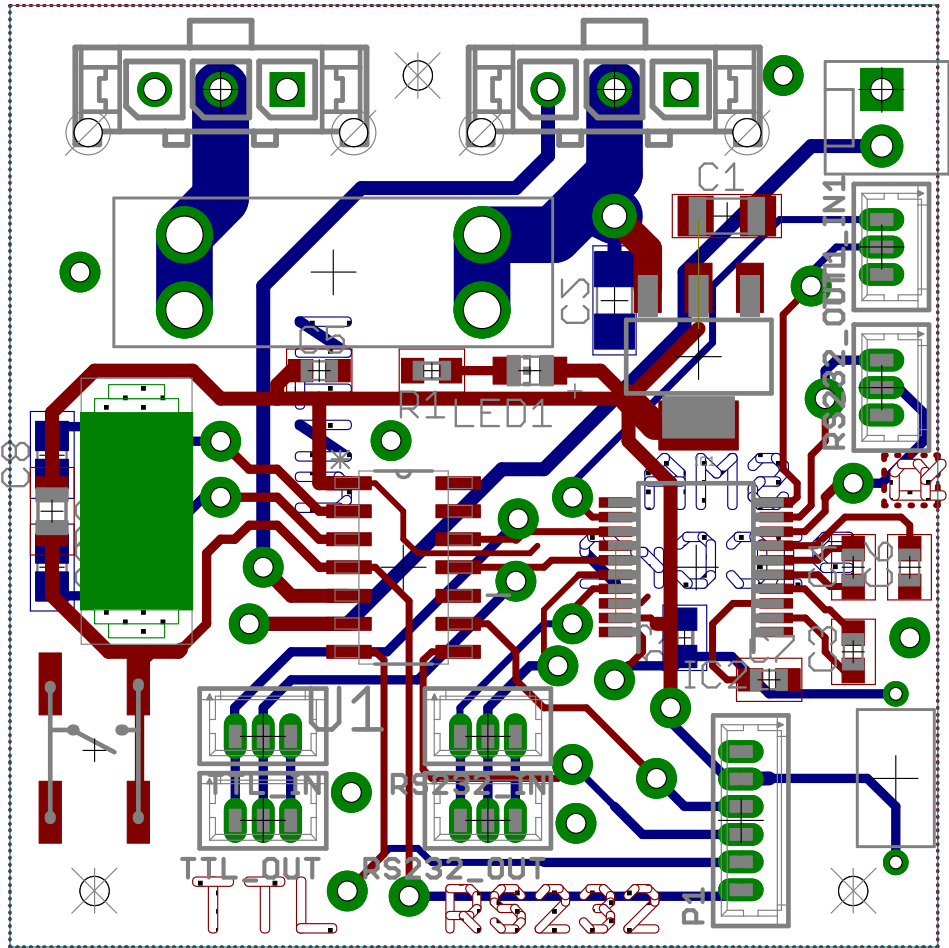


Figure A.2: PCB layout of the payload power control board

Bibliography

- [1] Goodrich Michael A., Morse Bryan S., Gerhardt Damon, Cooper Joseph L., Quigley Morgan, Adams Julie A., and Humphrey Curtis. Supporting wilderness search and rescue using a camera-equipped mini UAV. *Journal of Field Robotics*, 25(1-2):89–110, 2008. doi: 10.1002/rob.20226. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20226>.
- [2] N. Abdelkrim, N. Aouf, A. Tsourdos, and B. White. Robust nonlinear filtering for INS/GPS UAV localization. In *2008 16th Mediterranean Conference on Control and Automation*, pages 695–702, June 2008. doi: 10.1109/MED.2008.4602149.
- [3] S. Ahrens, D. Levine, G. Andrews, and J. P. How. Vision-based guidance and control of a hovering vehicle in unknown, GPS-denied environments. In *2009 IEEE International Conference on Robotics and Automation*, pages 2643–2648, May 2009.
- [4] S. M. Albrektsen and T. A. Johansen. SyncBoard - a high accuracy sensor timing board for UAV payloads. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1706–1715, Miami, Florida, USA, June 13–16 2017. doi: 10.1109/ICUAS.2017.7991410.
- [5] S. M. Albrektsen, A Sægrov, and T. A Johansen. Navigation of UAV using phased array radio. In *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-*

- UAS), pages 138–143, Linköping, Sweden, October 3–5 2017. doi: 10.1109/RED-UAS.2017.8101657.
- [6] S. M. Albrektsen, T. H. Bryne, and T. A. Johansen. Phased array radio system aided inertial navigation for unmanned aerial vehicles. In *2018 IEEE Aerospace Conference*, Big Sky, Montana, USA, March 3–10 2018. doi: 10.1109/AERO.2018.8396433.
- [7] S. M. Albrektsen, T. H. Bryne, and T. A. Johansen. Robust and secure UAV navigation using GNSS, phased-array radio system and inertial sensor fusion. In *IEEE Conference on Control Technology and Applications (CCTA)*, Copenhagen, Denmark, August 21–24 2018.
- [8] Sigurd M. Albrektsen and Tor Arne Johansen. User-configurable timing and navigation for UAVs. *Sensors*, 18(8):2468, 2018. ISSN 1424-8220. doi: 10.3390/s18082468. URL <http://www.mdpi.com/1424-8220/18/8/2468>.
- [9] *ADIS16488A*. Analog Devices, November 2016. Rev. D, <http://www.analog.com/media/en/technical-documentation/data-sheets/ADIS16488A.pdf>.
- [10] *ADIS164890*. Analog Devices, March 2018. Rev. B, <http://www.analog.com/media/en/technical-documentation/data-sheets/adis16490.pdf>.
- [11] Analog Devices Inertial Measurement Units. Analog devices inertial measurement units. <http://www.analog.com/en/parametricsearch/11172#/sel=row28|row24|row25|row27>, 2018. [Accessed 2018-04-18].
- [12] Arduino. <https://www.arduino.cc/>, 2017.
- [13] Abraham Bachrach, Ruijie He, and Nicholas Roy. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1:217–228, December 2009.
- [14] Abraham Bachrach, Samuel Prentice, Ruijie He, and Nicholas Roy. RANGE—robust autonomous navigation in GPS-denied environments. *Journal of Field Robotics*, 28(5):644–666, 2011. ISSN 1556-4967. doi: 10.1002/rob.20400.
- [15] Yaakov Bar-Shalom and Xiao-Rong Li. *Multitarget-Multisensor Tracking: Principles and Techniques*. YBS Publishing, 1995.

-
- [16] Mohamed Brahmi. *Reference Systems for Environmental Perception*, pages 205–221. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-36455-6. doi: 10.1007/978-3-642-36455-6_9. URL https://doi.org/10.1007/978-3-642-36455-6_9.
 - [17] Pascal Brisset, Antoine Drouin, Michel Gorraz, Pierre-Selim Huard, and Jeremy Tyler. The Paparazzi solution. In *MAV 2006, 2nd US-European Competition and Workshop on Micro Air Vehicles*, Sandestin, United States, October 2006. URL <https://hal-enac.archives-ouvertes.fr/hal-01004157>.
 - [18] Roland Brockers, Sara Susca, David Zhu, and Larry Matthies. Fully self-contained vision-aided navigation and landing of a micro air vehicle independent from external sensor inputs. In *Proc.SPI, Unmanned Systems Technology XIV*, volume 8387, pages 8387 – 8387 – 10, 2012. doi: 10.1117/12.919278. URL <https://doi.org/10.1117/12.919278>.
 - [19] T. H. Bryne, J. M. Hansen, R. H. Rogne, N. Sokolova, T. I. Fossen, and T. A. Johansen. Nonlinear observers for integrated INS/GNSS navigation - implementation aspects. In *IEEE Control Systems Magazine*, 2017. (accepted).
 - [20] T. H. Bryne, J. M. Hansen, R. H. Rogne, N. Sokolova, T. I. Fossen, and T. A. Johansen. Nonlinear observers for integrated INS/GNSS navigation – Implementation aspects. *IEEE Control Systems Magazine*, 37(3):59–86, 2017.
 - [21] T. Cheviron, T. Hamel, R. Mahony, and G. Baldwin. Robust nonlinear fusion of inertial and visual data for position, velocity and attitude estimation of UAV. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2010–2016, April 2007. doi: 10.1109/ROBOT.2007.363617.
 - [22] *CCT Product List*. Cloud Cap Technology, February 2014. http://www.cloudcaptech.com/images/uploads/documents/CCT_Product_List.pdf.
 - [23] Muhammad Darwish. Did russia make this ship disappear? <http://money.cnn.com/2017/11/03/technology/gps-spoofing-russia/index.html>, November 2017. [Accessed 2018-05-16].

- [24] C. De Michele, F. Avanzi, D. Passoni, R. Barzaghi, L. Pinto, P. Dosso, A. Ghezzi, R. Gianatti, and G. Della Vedova. Using a fixed-wing UAS to map snow depth distribution: an evaluation at peak accumulation. *The Cryosphere*, 10(2):511–522, 2016. doi: 10.5194/tc-10-511-2016. URL <https://www.the-cryosphere.net/10/511/2016/>.
- [25] Chung Deng, Shengwei Wang, Zhi Huang, Zhongfu Tan, and Junyong Liu. Unmanned aerial vehicles for power line inspection: A cooperative way in platforms and communications. *J. Commun*, 9(9):687–692, 2014.
- [26] U. Denier. Analysis and design of an ultralow-power CMOS relaxation oscillator. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(8):1973–1982, Aug 2010. ISSN 1549-8328. doi: 10.1109/TCSI.2010.2041504.
- [27] W Ding, J Wang, P Mumford, Y Li, and C Rizos. Time synchronization design for integrated positioning and georeferencing systems. In *Proceedings of SSC*, pages 1265–1274, 2005.
- [28] Weidong Ding, Jinling Wang, Yong Li, Peter Mumford, and Chris Rizos. Time synchronization error and calibration in integrated GP-S/INS systems. *ETRI journal*, 30(1):59–67, 2008.
- [29] Patrick Doherty and Piotr Rudol. A UAV search and rescue scenario with human body detection and geolocalization. In Mehmet A. Orgun and John Thornton, editors, *AI 2007: Advances in Artificial Intelligence*, pages 1–13, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-76928-6.
- [30] S. Dorafshan, M. Maguire, N. V. Hoffer, and C. Coopmans. Challenges in bridge inspection using small unmanned aerial systems: Results and lessons learned. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1722–1730, June 2017. doi: 10.1109/ICUAS.2017.7991459.
- [31] Raul Dorobantu and Benedikt Zebhauser. Field evaluation of a low-cost strapdown IMU by means GPS. *Ortung und Navigation*, 1(1999): 51–65, 1999.
- [32] DUNE. Dune: Unified navigation environment. <https://www.lsts.pt/toolchain/dune>, 2017. [Accessed 2018-04-09].

-
- [33] C. Ebert and C. Jones. Embedded software: Facts, figures, and future. *Computer*, 42(4):42–52, April 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.118.
- [34] J. A. Farrell. *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill, 2008.
- [35] Fletcher 8. TCP alternate checksum options. <http://www.ietf.org/rfc/rfc1145.txt>, 1990.
- [36] J Fortuna and T A Johansen. A lightweight payload for hyperspectral remote sensing using small UAVs. In *9th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, 2018. (Accepted).
- [37] João Fortuna and Harald Martens. Multivariate data modelling for de-shadowing of airborne hyperspectral imaging. *Journal of Spectral Imaging*, 6(1):a2, 2017. ISSN 2040-4565. doi: 10.1255/jsi.2017.a2. URL <https://doi.org/10.1255/jsi.2017.a2>.
- [38] Raspberry Pi Foundation. Raspberry pi 3 model b+. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>, 2018. [Accessed 2018-07-15].
- [39] L. Fusini, J. Hosen, H. H. Helgesen, T. A. Johansen, and T. I. Fossen. Experimental validation of a uniformly semi-globally exponentially stable non-linear observer for GNSS- and camera-aided inertial navigation for fixed-wing UAVs. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 851–860, June 2015. doi: 10.1109/ICUAS.2015.7152371.
- [40] L. Fusini, T. A. Johansen, and T. I. Fossen. A globally exponentially stable non-linear velocity observer for vision-aided UAV dead reckoning. In *2016 IEEE Aerospace Conference*, pages 1–9, March 2016. doi: 10.1109/AERO.2016.7500606.
- [41] Lorenzo Fusini, T. A. Johansen, and Thor Inge Fossen. Dead reckoning of a fixed-wing UAV with inertial navigation aided by optical flow. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1520–1259, June 2017.
- [42] GNU Screen. <https://www.gnu.org/software/screen/>, 2017.

-
- [43] GoLang. Channels. <https://tour.golang.org/concurrency/2>, 2018. [Accessed 2018-03-27].
 - [44] V. Grabe, H. H. Bühlhoff, and P. Robuffo Giordano. Robust optical-flow based self-motion estimation for a quadrotor UAV. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2153–2159, Oct 2012. doi: 10.1109/IROS.2012.6386234.
 - [45] GJ Grenzdörffer, A Engel, and B Teichert. The photogrammetric potential of low-cost UAVs in forestry and agriculture. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 31(B3):1207–1214, 2008.
 - [46] H. F. Grip, T. I. Fossen, T. A. Johansen, and A. Saberi. Attitude estimation using biased gyro and vector measurements with time-varying reference vectors. *IEEE Transactions on Automatic Control*, 57(5): 1332—1338, 2012. doi: 10.1109/TAC.2011.2173415.
 - [47] H. F. Grip, T. I. Fossen, T. A. Johansen, and A. Saberi. Nonlinear observer for GNSS-aided inertial navigation with quaternion-based attitude estimation. In *Proc. of the American Contr. Conf.*, pages 272–279, Washington, DC, June 2013. doi: 10.1109/ACC.2013.6579849.
 - [48] Håvard Fjær Grip, Wayne Johnson, Carlos Malpica, Daniel P Scharf, Milan Mandić, Larry Young, Brian Allan, Bérénice Mettler, and Miguel San Martin. Flight dynamics of a Mars helicopter. In *Proc. European Rotorcraft Forum*, 2017.
 - [49] Paul D. Groves. *Principles of GNSS, Inertial, and Multisensor integrated Navigation Systems*. GNSS Technology and Applications series. Artech House, 2 edition, 2013.
 - [50] Fredrik Gustafsson. *Statistical Sensor Fusion*. Studentlitteratur, 2nd edition, 2012.
 - [51] J. M. Hansen, T. I. Fossen, and T. A. Johansen. Nonlinear observer for INS aided by time-delayed GNSS measurements: Implementation and UAV experiments. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 157–166, June 2015. doi: 10.1109/ICUAS.2015.7152287.
 - [52] Jakob M. Hansen, Thor I. Fossen, and Tor Arne Johansen. Nonlinear observer design for GNSS-aided inertial navigation systems with time-delayed GNSS measurements. *Control Engineering Practice*, 60:

- 39 – 50, 2017. ISSN 0967-0661. doi: 10.1016/j.conengprac.2016.11.016. URL <http://www.sciencedirect.com/science/article/pii/S0967066116302751>. (accepted).
- [53] *User Manual ODROID-XU4*. Hard Kernel, 2015. URL <https://magazine.odroid.com/odroid-xu4>.
- [54] J. Hardy, J. Strader, J. N. Gross, Y. Gu, M. Keck, J. Douglas, and C. N. Taylor. Unmanned aerial vehicle relative navigation in GPS denied environments. In *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 344–352, April 2016. doi: 10.1109/PLANS.2016.7479719.
- [55] Håkon H. Helgesen, Frederik S. Leira, Torleiv H. Bryne, Sigurd M. Albrektsen, Thor I. Fossen, and Tor Arne Johansen. Real-time georeferencing of thermal images using small fixed-wing UAVs in maritime environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2018. (submitted).
- [56] Bruno Hérisse, Tarek Hamel, Robert Mahony, and François-Xavier Russotto. A terrain-following control approach for a VTOL unmanned aerial vehicle using average optical flow. *Autonomous Robots*, 29(3): 381–399, Nov 2010. ISSN 1573-7527. doi: 10.1007/s10514-010-9208-x. URL <https://doi.org/10.1007/s10514-010-9208-x>.
- [57] Charles Antony Richard Hoare. *Communicating Sequential Processes*. Prentice Hall International, May 2015.
- [58] *3-Axis Digital Compass IC HMC5883L*. Honeywell, 2 2013. Rev. E.
- [59] Eija Honkavaara, Heikki Saari, Jere Kaivosoja, Ilkka Pölönen, Teemu Hakala, Paula Litkey, Jussi Mäkynen, and Liisa Pesonen. Processing and assessment of spectrometric, stereoscopic imagery collected using a lightweight UAV spectral camera for precision agriculture. *Remote Sensing*, 5(10):5006–5039, 2013. ISSN 2072-4292. doi: 10.3390/rs5105006. URL <http://www.mdpi.com/2072-4292/5/10/5006>.
- [60] Alfred Horn. Doubly stochastic matrices and the diagonal of a rotation matrix. *American Journal of Mathematics*, 76(3):620–630, 1954. ISSN 00029327, 10806377. URL <http://www.jstor.org/stable/2372705>.
- [61] Knut-Sverre Horn. E-tjenesten bekrefter: Russerne jammet GPS-signaler bevisst. https://www.nrk.no/finnmark/e-tjenesten-bekrefter_

- [-russerne-jammet-gps-signaler-bevisst-1.13721504](#), October 2017. [Accessed 2018-05-16].
- [62] Jesper Hosen, Håkon H. Helgesen, Lorenzo Fusini, Thor I. Fossen, and Tor A. Johansen. Vision-aided nonlinear observer for fixed-wing unmanned aerial vehicle navigation. *Journal of Guidance, Control, and Dynamics*, 39(8):1777–1789, 2016.
 - [63] Minh-Duc Hua, Guillaume Ducard, T. Hamel, R. Mahony, and K. Rudin. Implementation of a nonlinear attitude estimator for aerial robotic vehicles. *IEEE Transactions On Control System Technology*, 22(1):201–212, 2014. doi: 10.1109/TCST.2013.2251635.
 - [64] T. Huck, A. Westenberger, M. Fritzsche, T. Schwarz, and K. Dietmayer. Precise timestamping and temporal synchronization in multi-sensor fusion. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 242–247, June 2011. doi: 10.1109/IVS.2011.5940472.
 - [65] Intel. Intel® edison development platform. <https://software.intel.com/en-us/iot/hardware/discontinued>, 2018. [Accessed 2018-07-15].
 - [66] *MPU-6000 and MPU-6050*. InvenSense, 8 2013. Rev. 3.4.
 - [67] Ali Jafarnia-Jahromi, Saeed Daneshmand, and Gérard Lachapelle. Spoofing countermeasure for gnss receivers—a review of current and future research trends. *Proc. on the 4th Intern Colloquium on Scientific and Fundamental Aspects of the Galileo Programme*, pages 1–8, 2013.
 - [68] T. A. Johansen, A. Cristofaro, K. Sørensen, J. M. Hansen, and T. I. Fossen. On estimation of wind velocity, angle-of-attack and sideslip angle of small UAVs using standard sensors. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 510–519, June 2015. doi: 10.1109/ICUAS.2015.7152330.
 - [69] T. A. Johansen, J.M. Hansen, and T. I. Fossen. Nonlinear observer for tightly integrated inertial navigation aided by pseudo-range measurements. *Journal of Dynamic Systems, Measurement, and Control*, 139(1)(011007):1–10, 2016. doi: 10.1115/1.4034496.
 - [70] T. A. Johansen, J. M. Hansen, and T. I. Fossen. Nonlinear observer for tightly integrated inertial navigation aided by pseudo-range mea-

- surements. *ASME Journal of Dynamic Systems, Measurement and Control*, 139(1):011007–011007–10, 2017. doi: 10.1115/1.4034496.
- [71] M. Kais, D. Millescamp, D. Betaille, B. Lusetti, and A. Chapelon. A multi-sensor acquisition architecture and real-time reference for sensor and fusion methods benchmarking. In *2006 IEEE Intelligent Vehicles Symposium*, pages 418–423, 2006. doi: 10.1109/IVS.2006.1689664.
- [72] Kartverket. Closure of Loran C stations. *Notices to Mariners (Efs)*, 146(1):26, January 2015. <https://kartverket.no/efs-documents/editions/2015/efs01-2015.pdf>.
- [73] F. Kendoul and K. Nonami. A visual navigation system for autonomous flight of micro air vehicles. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3888–3893, Oct 2009. doi: 10.1109/IROS.2009.5354730.
- [74] Andrew J. Kerns, Daniel P. Shepard, Jahshan A. Bhatti, and Todd E. Humphreys. Unmanned aircraft capture and control via GPS spoofing. *Journal of Field Robotics*, 31(4):617–636, 2014. ISSN 1556-4967. doi: 10.1002/rob.21513.
- [75] T. Krajník, M. Nitsche, S. Pedre, L. Přeučil, and M. E. Mejail. A simple visual navigation system for an UAV. In *International Multi-Conference on Systems, Signals Devices*, pages 1–6, March 2012. doi: 10.1109/SSD.2012.6198031.
- [76] Gregory T. Kremer, Rudolph M. Kalafus, Peter V. W. Loomis, and James C. Reynolds. The effect of selective availability on differential GPS corrections. *Navigation*, 37(1):39–52, 1990. ISSN 2161-4296. doi: 10.1002/j.2161-4296.1990.tb01533.x. URL <http://dx.doi.org/10.1002/j.2161-4296.1990.tb01533.x>.
- [77] H. Krim and M. Viberg. Two decades of array signal processing research: the parametric approach. *IEEE Signal Processing Magazine*, 13(4):67–94, Jul 1996. ISSN 1053-5888. doi: 10.1109/79.526899.
- [78] Binghao Li, Chris Rizos, Hyung Keun Lee, and Hung Kyu Lee. A GPS-slaved time synchronization system for hybrid navigation. *GPS Solutions*, 10(3):207–217, 2006. ISSN 1521-1886. doi: 10.1007/s10291-006-0022-z. URL <http://dx.doi.org/10.1007/s10291-006-0022-z>.

- [79] Yong Li, Peter Mumford, and Chris Rizos. Seamless navigation through GPS outages – a low-cost GPS/INS solution. *Inside GNSS, July/August*, 3(5):39–45, 2008.
- [80] V. Lippiello, G. Loianno, and B. Siciliano. MAV indoor navigation based on a closed-form solution for absolute scale velocity estimation using optical flow and inertial data. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 3566–3571, Dec 2011. doi: 10.1109/CDC.2011.6160577.
- [81] Chang Liu, John Nash, and Stephen Prior. A low-cost vision-based unmanned aerial system for extremely low-light GPS-denied navigation and thermal imaging. *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, 9(10):1750–1757, October 2015. URL <https://eprints.soton.ac.uk/399629/>.
- [82] M. Ludvigsen, S. M. Albrektsen, K. Cisek, T. A. Johansen, P. Norgren, R. Skjetne, A. Zolich, P. Sousa Dias, S. Ferreira, J. B. de Sousa, T. O. Fossum, Ø. Sture, T. Røbekk Krogstad, Ø. Midtgaard, V. Hovstein, and E. Vågsholm. Network of heterogeneous autonomous vehicles for marine research and management. In *OCEANS 2016 MT-S/IEEE Monterey*, pages 1–7, Sept 2016. doi: 10.1109/OCEANS.2016.7761494.
- [83] S. Lupashin, A. Schöllig, M. Sherback, and R. D’Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *2010 IEEE International Conference on Robotics and Automation*, pages 1642–1648, May 2010. doi: 10.1109/ROBOT.2010.5509452.
- [84] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An invitation to 3D vision : from images to geometric models*. Springer, 2000.
- [85] James V Marcaccio, Chantel E Markle, and Patricia Chow-Fraser. Use of fixed-wing and multi-rotor unmanned aerial vehicles to map dynamic changes in a freshwater marsh. *Journal of Unmanned Vehicle Systems*, 4(3):193–202, 2016.
- [86] F Landis Markley. Attitude error representations for kalman filtering. *Journal of guidance, control, and dynamics*, 26(2):311–317, 2003.
- [87] P. P. Markopoulos, N. Tsagkarakis, D. A. Pados, and G. N. Karystinos. Direction-of-arrival estimation by l1-norm principal components. In *2016 IEEE International Symposium on Phased Array Systems and*

- Technology (PAST)*, pages 1–6, Oct 2016. doi: 10.1109/ARRAY.2016.7832585.
- [88] D. T. McGrath. Wideband arrays and polarization synthesis. In *2016 IEEE International Symposium on Phased Array Systems and Technology (PAST)*, pages 1–5, Oct 2016. doi: 10.1109/ARRAY.2016.7832639.
 - [89] *MS5611-01BA03 Barometric Pressure Sensor*. Measurement Specialties, 9 2015.
 - [90] *MS4525DO*. Measurement Specialties, 7 2016.
 - [91] L. Meier, D. Honegger, and M. Pollefeys. PX4: A node-based multi-threaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, may 2015.
 - [92] Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, 33(1):21–39, Aug 2012. ISSN 1573-7527. doi: 10.1007/s10514-012-9281-4. URL <https://doi.org/10.1007/s10514-012-9281-4>.
 - [93] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The GRASP multiple micro-UAV testbed. *IEEE Robotics Automation Magazine*, 17(3):56–65, Sept 2010. ISSN 1070-9932.
 - [94] *PIC32MZ Embedded Connectivity with Floating Point Unit (EF) Family*. Microchip, <http://ww1.microchip.com/downloads/en/DeviceDoc/60001320D.pdf>, November 2016. Rev. D.
 - [95] Minicom. Alioth minicom project home. <https://alioth.debian.org/projects/minicom>, 2017. [Accessed 2018-03-27].
 - [96] Shatadal Mishra and Srikanth Saripalli. *GPS-free navigation for micro aerial vehicles*, pages 189–193. American Helicopter Society International, 2015. ISBN 9781510810129.
 - [97] Parastoo Mohagheghi, Reidar Conradi, Ole M. Killi, and Henrik Schwarz. An empirical study of software reuse vs. defect-density and stability. In *Proceedings of the 26th International Conference on Software Engineering, ICSE '04*, pages 282–292, Washington, DC,

-
- USA, 2004. IEEE Computer Society. ISBN 0-7695-2163-0. URL <http://dl.acm.org/citation.cfm?id=998675.999433>.
- [98] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar. Uavs for smart cities: Opportunities and challenges. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 267–273, May 2014. doi: 10.1109/ICUAS.2014.6842265.
- [99] *NMEA 0183 Standard For Interfacing Marine Electronic Devices*. National Marine Electronics Association, 3.01 edition, January 2002.
- [100] NetOcean. FEUP reforça cooperação com a Noruega [FEUP strengthens cooperation with Norway], 2016. URL <http://www.jornaldaeconomiadomar.com/feup-reforca-cooperacao-noruega/>. News article, Portuguese only.
- [101] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1-2):100–115, 1954. doi: 10.1093/biomet/41.1-2.100. URL <http://dx.doi.org/10.1093/biomet/41.1-2.100>.
- [102] M. G. Petovello, K. O’Keefe, G. Lachapelle, and M. E. Cannon. Consideration of time-correlated errors in a Kalman filter applicable to gnss. *Journal of Geodesy*, 83(1):51—56, 2009. doi: 10.1007/s00190-008-0231-z.
- [103] Aron Pinker and Charles Smith. Vulnerability of the GPS signal to jamming. *GPS Solutions*, 3(2):19–27, 1999. ISSN 1080-5370. doi: 10.1007/PL00012788.
- [104] PJRC. Teensy usb development board. <https://www.pjrc.com/teensy/>, 2018. [Accessed 2018-07-15].
- [105] Mark L. Psiaki, Todd E. Humphreys, and Brian Stauffer. Attackers can spoof navigation signals without our knowledge. Here’s how to fight back GPS lies. *IEEE Spectrum*, 53(8):26–53, 2016.
- [106] PuTTY. Putty: a free ssh and telnet client. <https://www.chiark.greenend.org.uk/~sgtatham/putty/>, 2017. [Accessed 2018-05-03].
- [107] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

-
- [108] Martin Rehak, Romain Mabillard, and Jan Skaloud. A micro-UAV with the capability of direct georeferencing. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, pages 317–323, 2013.
- [109] A.W. Roscoe and C.A.R. Hoare. The laws of occam programming. *Theoretical Computer Science*, 60(2):177 – 229, 1988. ISSN 0304-3975. doi: 10.1016/0304-3975(88)90049-7. URL <http://www.sciencedirect.com/science/article/pii/0304397588900497>.
- [110] R. Roy and T. Kailath. ESPRIT-estimation of signal parameters via rotational invariance techniques. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(7):984–995, Jul 1989. ISSN 0096-3518. doi: 10.1109/29.32276.
- [111] M. Saied, B. Lussier, I. Fantoni, C. Francis, H. Shraim, and G. Sanahuja. Fault diagnosis and fault-tolerant control strategy for rotor failure in an octorotor. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5266–5271, May 2015. doi: 10.1109/ICRA.2015.7139933.
- [112] K. Schmid and H. Hirschmüller. Stereo vision and IMU based real-time ego-motion and depth image computation on a handheld device. In *2013 IEEE International Conference on Robotics and Automation*, pages 4671–4678, May 2013. doi: 10.1109/ICRA.2013.6631242.
- [113] K. Schmid, F. Ruess, M. Suppa, and D. Burschka. State estimation for highly dynamic flying systems using key frame odometry with varying time delays. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004, Oct 2012. doi: 10.1109/IROS.2012.6385969.
- [114] R. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, 34(3):276–280, Mar 1986. ISSN 0018-926X. doi: 10.1109/TAP.1986.1143830.
- [115] K. P. Schwarz, H. E. Martell, N. El-Sheimy, R. Li, M. A. Chapman, and D. Cosandier. Viasat - a mobile highway survey system of high accuracy. In *Vehicle Navigation and Information Systems Conference, 1993., Proceedings of the IEEE-IEE*, pages 476–481, Oct 1993. doi: 10.1109/VNIS.1993.585676.
- [116] Clare Sebastian. Getting lost near the kremlin? russia could be 'GPS spoofing'. <http://money.cnn.com/2016/12/02/technology/>

- kremlin-gps-signals/index.html, December 2016. [Accessed 2018-05-16].
- [117] *STIM300 Inertia Measurement Unit*. sensoror, February 2018. TS1524, rev. 24, <https://www.sensoror.com/media/1305/ts1524r24-datasheet-stim300.pdf>.
 - [118] S. Shen, N. Michael, and V. Kumar. Autonomous multi-floor indoor navigation with a computationally constrained MAV. In *2011 IEEE International Conference on Robotics and Automation*, pages 20–25, May 2011. doi: 10.1109/ICRA.2011.5980357.
 - [119] Jianbo Shi and C. Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, June 1994. doi: 10.1109/CVPR.1994.323794.
 - [120] Sebastian Siebert and Jochen Teizer. Mobile 3d mapping for surveying earthwork projects using an unmanned aerial vehicle (UAV) system. *Automation in Construction*, 41:1 – 14, 2014. ISSN 0926-5805. doi: 10.1016/j.autcon.2014.01.004. URL <http://www.sciencedirect.com/science/article/pii/S0926580514000193>.
 - [121] Jan Skaloud and Patrick Viret. GPS/INS integration. *European Journal of Navigation*, 2(4):40–44, 2004.
 - [122] I. Skog and P. Handel. Time synchronization errors in loosely coupled GPS-aided inertial navigation systems. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1014–1023, Dec 2011. ISSN 1524-9050. doi: 10.1109/TITS.2011.2126569.
 - [123] Olga Sorkine-Hornung and Michael Rabinovich. Least-squares rigid motion using SVD, January 2017. URL https://igl.ethz.ch/projects/ARAP/svd_rot.pdf.
 - [124] V. Sreeja. Impact and mitigation of space weather effects on gnss receiver performance. *Geoscience Letters*, 3(1):24, Aug 2016. ISSN 2196-4092. doi: 10.1186/s40562-016-0057-0. URL <https://doi.org/10.1186/s40562-016-0057-0>.
 - [125] Barbara Starr and Ryan Browne. Ash Carter: ‘Navigational error’ behind U.S. sailors ending up in iran. CNN: <http://edition.cnn.com/2016/01/14/politics/navy-boats-iran-waters/index.html>, January 15 2016.

-
- [126] *L3GD20H MEMS motion sensor*. STMicroelectronics, 3 2013. Rev. 2.
 - [127] *LSM303D 3D accelerometer and 3D magnetometer*. STMicroelectronics, 11 2013. Rev. 2.
 - [128] P. Stoica and K. C. Sharman. Maximum likelihood methods for direction-of-arrival estimation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(7):1132–1143, Jul 1990. ISSN 0096-3518. doi: 10.1109/29.57542.
 - [129] B. Strasser, A. Siegel, K.-H. Siedersberger, M. Maurer, and H. Bubb. Vernetzung von Test- und Simulationsmethoden für die Entwicklung von Fahrerassistenzsystemen (FAS). In *Tagung Aktive Sicherheit durch Fahrerassistenz*, 2010. URL http://www.ftm.mw.tum.de/uploads/media/18_strasser.pdf.
 - [130] Tomoji Takasu and Akio Yasuda. Development of the low-cost rtk-gps receiver with an open source program package rtklib. In *International symposium on GPS/GNSS*, pages 4–6. International Convention Center Jeju Korea, 2009.
 - [131] David H. Titterton and John L. Weston. *Strapdown inertial navigation technology*. Institution of Electrical Engineers and American Institute of Aeronautics and Astronautics, 2nd edition, 2004.
 - [132] Pratap Tokekar, Joshua Vander Hook, David Mulla, and Volkan Isler. Sensor planning for a symbiotic UAV and UGV system for precision agriculture. *IEEE Transactions on Robotics*, 32(6):1498–1511, 2016.
 - [133] N. Tsagkarakis, P. P. Markopoulos, and D. A. Pados. Direction finding by complex L1-principal-component analysis. In *2015 IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 475–479, June 2015. doi: 10.1109/SPAWC.2015.7227083.
 - [134] G. M. Turner and C. Christodoulou. FDTD analysis of phased array antennas. *IEEE Transactions on Antennas and Propagation*, 47(4): 661–667, Apr 1999. ISSN 0018-926X. doi: 10.1109/8.768805.
 - [135] *NEO-7 u-blox 7 GNSS modules*. u-blox, November 2014. URL https://www.u-blox.com/sites/default/files/products/documents/NEO-7_DataSheet_%28UBX-13003830%29.pdf. R07.

- [136] *u-blox 8 / u-blox M8 Receiver Description*. u-blox, March 2018. R15, [https://www.u-blox.com/sites/default/files/products/documents/u-blox8-M8_ReceiverDescrProtSpec_\(UBX-13003221\)_Public.pdf](https://www.u-blox.com/sites/default/files/products/documents/u-blox8-M8_ReceiverDescrProtSpec_(UBX-13003221)_Public.pdf).
- [137] udev - Linux dynamic device management. <https://wiki.debian.org/udev>, 2017.
- [138] DHS U.S. Coast Guard. Terminate long range aids to navigation (Loran-C) signal. *Federal Register*, 75(4):998, January 2010. https://www.navcen.uscg.gov/pdf/loran/geninfo/USCG_FRDOC_0001-1185.pdf.
- [139] DHS U.S. Coast Guard. Backup global positioning system (H.R. 2825), June 2017. <https://www.gps.gov/policy/legislation/loran-c/#cgauth2017>.
- [140] H. H. Vo, C. C. Chen, P. Hagan, and Y. Bayram. A very low-profile UWB phased array antenna design for supporting wide angle beam steering. In *2016 IEEE International Symposium on Phased Array Systems and Technology (PAST)*, pages 1–8, Oct 2016. doi: 10.1109/ARRAY.2016.7832578.
- [141] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In *2012 IEEE International Conference on Robotics and Automation*, pages 957–964, May 2012. doi: 10.1109/ICRA.2012.6225147.
- [142] S. Weiss, R. Brockers, and L. Matthies. 4DoF drift free navigation using inertial cues and optical flow. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4180–4186, Nov 2013. doi: 10.1109/IROS.2013.6696955.
- [143] S. Weiss, R. Brockers, S. M. Albrektsen, and L. Matthies. Inertial optical flow for throw-and-go micro air vehicles. In *2015 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 262–269, Waikoloa, Hawaii, USA, January 5–9 2015. doi: 10.1109/WACV.2015.42.
- [144] Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Monocular-SLAM based navigation for autonomous micro helicopters in GPS-denied environments. *J. Field Robot.*, 28(6):854–874, November 2011. ISSN 1556-4959. doi: 10.1002/rob.20412.

-
- [145] Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011. ISSN 1556-4967. doi: 10.1002/rob.20412.
- [146] Karl Engelbert Wenzel, Andreas Masselli, and Andreas Zell. Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle. *Journal of Intelligent & Robotic Systems*, 61(1):221–238, Jan 2011. ISSN 1573-0409. doi: 10.1007/s10846-010-9473-0. URL <https://doi.org/10.1007/s10846-010-9473-0>.
- [147] Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, and Paul Wilson. Best practices for scientific computing. *PLOS Biology*, 12(1):1–7, 01 2014. doi: 10.1371/journal.pbio.1001745. URL <https://doi.org/10.1371/journal.pbio.1001745>.
- [148] B. Yun, K. Peng, and B. M. Chen. Enhancement of GPS signals for automatic control of a UAV helicopter system. In *2007 IEEE International Conference on Control and Automation*, pages 1185–1189, May 2007. doi: 10.1109/ICCA.2007.4376547.
- [149] W. Zang, J. Lin, Y. Wang, and H. Tao. Investigating small-scale water pollution with uav remote sensing technology. In *World Automation Congress 2012*, pages 1–4, June 2012.
- [150] J. C. Zufferey and D. Floreano. Toward 30-gram autonomous indoor aircraft: Vision-based obstacle avoidance and altitude control. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2594–2599, April 2005. doi: 10.1109/ROBOT.2005.1570504.
- [151] J.F. Zumberge and G. Gendt. The demise of selective availability and implications for the international gps service. *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy*, 26(6): 637 – 644, 2001. ISSN 1464-1895. doi: 10.1016/S1464-1895(01)00113-2. URL <http://www.sciencedirect.com/science/article/pii/S1464189501001132>. Proceedings of the First COST Action 716 Workshop Towards Operational GPS Meteorology and the Second Network Workshop of the International GPS Service (IGS).