Improved DGNSS-based Positioning of Micro UAV Platforms for Sensor Web Services

Master's Thesis

Submitted to the

University of Münster

Institute for Geoinformatics

Submitted by:	Jakob Geipel
Registration number:	367650
Supervisor:	Prof. Dr. Edzer Pebesma
Co-supervisor:	Dr. Torsten Prinz

Münster, February 2012

Acknowledgments

I profoundly thank my supervisor, Prof. Dr. Edzer Pebesma, for sharing his suggestions and for his generous support of the ifgicopter project.

I also thank my co-supervisor, Dr. Torsten Prinz, for invoking the ifgicopter project and his guidance and insightful ideas, which helped me to accomplish this work.

Moreover, I want to express my gratitude to Sven Jürß for offering a MD4-1000 Sensor Platform for deployment and especially Tobias Matschke for his time and patience in helping me with all platform related questions.

Furthermore, I want to thank Peter Sulmann, Patrick Wisotzki and Matthes Rieke for their support and advice in implementing hard- and software components.

Finally, I wish to acknowledge my sister, Christine, for proofreading this work and giving me helpful feedback.

Annotations

As this thesis contains unreferenced figures and tables, I want to clarify that they were designed by myself. Pictograms that occur in the figures have been published for free or non-commercial use.

Abstract

The usage of Micro Unmanned Aerial Vehicles (MUAV) as mobile sensor platforms is constantly increasing in the scientific, as well as in the civilian sector. A variety of requirements evolve from upcoming mission tasks like documentation, surveying and inspection in agriculture and geography, as well as in the industry. Many applications, such as the creation of orthoimages or the inspection of industrial plants need accurate position information in real-time, both for safety-in-flight reasons and for enriching sensor data by the provision of location.

As current MUAVs make use of common Global Positioning System receivers and, therefore, do not guarantee reliable high-precision positioning, this work examines the demands on an improved Differential Global Navigation Satellite System (DGNSS) positioning system for its integration into an existing MUAV platform. It proposes a flexible system architecture and presents a modular prototype that offers the possibility to exchange discrete components for making use of more sophisticated technologies like Precise DGNSS. The described prototype already guarantees horizontal positioning accuracy of 35 cm in real-time, which can be considered as sufficient for the majority of applications.

Consequently, this work focuses on the integration of position and additional navigation data into an existing Sensor Platform Framework software, which is able to synchronize sensor and navigation information on-the-fly. It introduces a MUAV platform-specific Input-Plugin for decoding the telemetry data stream and for the communication with the framework. As the framework is able to forward the processed geodata in a standardized way according to the guidelines of the Open Geospatial Consortium Inc., the data can be exploited by any kind of Sensor Web Service in near real-time.

Table of Contents

Nomenclature

2dRMS	Two Distance Root Mean Square (93-98%)
C/A	Coarse/Acquisition
CEP50	Circular Error Probability (50%)
CEP95	Circular Error Probability (95%)
DGNSS	Differential Global Navigation Satellite System
DGPS	Differential Global Positioning System
DOP	Dilution of Precision
dRMS	Distance Root Mean Square (63-68%)
EGNOS	European Geostationary Navigation Overlay Service
GBAS	Ground Bases Augmentation System
GLONASS	Globalnaya Navigatsionnaya Sputnikovaya Sistema
GNSS	Global Navigation Satellite System
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
MEMS	Micro Electro Mechanical System
MSAS	Multi-functional Satellite Augmentation System
MUAV	Micro Unmanned Aerial Vehicle
NC	Navigation Controller
NMEA 0183	National Marine Electronics Association 0183 Interface Standard
Ntrip	Networked Transport of RTCM via Internet Protocol
O&M	Observations & Measurements
OGC	Open Geospatial Consortium Inc.
PDGNSS	Precise DGNSS

RC	Remote Control
RMS	Root Mean Square (68.3%)
RTCM	Radio Technical Commission for Maritime Services
RTK-GNSS	Real-Time Kinematic GNSS
SAPOS	Satellitenpositionierungsdienst der deutschen Landesvermessung (German Correction Signal Service)
SAPOS HEPS	SAPOS Hochpräziser Echtzeit-Positionierungs-Service (High precision positioning service)
SBAS	Satellite Based Augmentation System
SensorML	Sensor Model Language
SIM	Subscriber Identification Module
SOS	Sensor Observation Service
SPS	Satellite Positioning Service
SWE	Sensor Web Enablement
URL	Uniform Resource Locator
URN	Uniform Resource Name
USB	Universal Serial Bus
UTM	Universal Transversal Mercator System
VDC	Voltage Direct Current
VRS	Virtuelle Referenz Station (non-physical reference station)
WAAS	Wide Area Augmentation System
WGS84	World Geodetic System 84
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

List of Figures

List of Tables

List of Listings

1. Introduction

Micro Unmanned Aerial Vehicles (MUAV) are a recent development as powerful means in fields, i.e. surveying, monitoring and disaster management. As they are equipped with additional sensors, MUAVs evolve to mobile sensor platforms, which can be used to gather sensor observations in various mission scenarios. MUAVs strongly rely on their internal navigation sensors, which guarantee stable maneuver in flight and offer the possibilty to produce geodata by enriching sensor observations with the sensor platform's location. Besides, these navigation sensors enable MUAVs to automatically execute predefined waypoint routes and waypoint-specific mission tasks. Supporting these abilities, it is of interest to improve the MUAVs navigation ability and to facilitate access to their navigation data.

1.1. Motivation

MUAVs are commonly sold as preconfigured systems with definite hardware and software components. Navigation performance and data access heavily depend on manufacturer-specific navigation sensors and software interfaces.

The absolute positioning efficiency of MUAVs currently builds on simple code-based Global Navigation Satellite System (GNSS) receivers and their resulting accuracy perfomance. Flight missions in industry, i.e. the inspection of industrial plants, pipelines or solar parks demand high precision absolute positioning. MUAVs often need to approach objects of interest and at the same time avoid damages to the sensed object or themselves in case of a crash. Besides, scientific research, especially in the field of close-range photogrammetry, requires accurate direct sensor orientation [Remondino et al., 2011]. In the cited work, it is mentioned that the use of more sophisticated GNSS positioning techniques would improve the quality of positioning, but would result in too complex, expensive and heavy systems. This statement leads to the question of developing a MUAV-specifc positioning system, which is able to improve absolute positioning by using inexpensive and low weight components. However, as costs are an important factor in sales, this work concentrates on the technical possibilities that are given to develop a MUAV-specific improved Differential GNSS (DGNSS) positioning system.

Besides improving absolute positioning, the acquisition and distribution of MUAV navigation sensor data is a crucial element for the use of gathered sensor observations. According to [Jirka et al., 2010], the integration of sensor data into various application systems, is a challenging task due to the variety of different data formats. As every MUAV producer builds on generic data formats and software applications, there is a high demand in exchanging sensor observations in an interoperable and standardized way. The Open Geospatial Consortium Inc. (OGC), which is an international organisation for encouraging the development of open geostandards, educed the Sensor Web Enablement (SWE) initiative. The SWE initiative resulted in a framework, which offers two sets of standards for sensor data exchange. Using these two sets as a basis, this thesis concentrates on the implementation of a software, which enables a MUAV to forward its navigation sensor information to the Sensor Web and to distribute it via Sensor Web Services (SWS) on-the-fly.

1.2. Structure

The thesis is structured in eight chapters, which can be roughly divided into three blocks. The first block introduces the thesis' objectives and the motivation, which led to writing this work (see Chapter 1). It contains a description of the fundamentals of the MD4-1000 MUAV, DGNSS and OGC's SWE and establishes the basis for the understanding of the concepts and strategies, which are used in the second part (see Chapter 2). Moreover, Chapter 3 analyzes the necessary requirements.

The second block describes the architecture of the improved DGNSS-based positioning solution and the design of the software, which has been developed for providing the platform's navigation data to SWSs. Chapter 4 shows the concepts and strategies, whereas Chapter 5 considers the implementation of all hardware and software components.

The last block comprises an analysis of the implemented positioning system's accuracy and addresses the possibility of using the platform's navigation sensor data in SWS (see Chapter 6). Chapter 7 discusses the outcomes and outlines for possible future work on the MD4-1000's positioning system and the usage of its sensor data. Finally, Chapter 8 summarizes this Master's Thesis.

2. Fundamentals

The following sections will provide a short introduction to the basics of common rotary MUAVs, using the example of the MD4-1000 Sensor Platform and present an overview of the technology, which is used for positioning and navigation. Furthermore, the concept of Differential GNSS will be illustrated to give fundamental understanding and its use in the course of this thesis. Ultimately, the Sensor Web and a solution to forward sensor data to Sensor Web Services will be elucidated.

This chapter will establish the basis for the understanding of the concepts and strategies of an improved DGNSS-based positioning architecture and the provision of MUAV sensor data to Sensor Web Services.

2.1. MD4-1000 as MUAV Sensor Platform

The MD4-1000 is a MUAV Sensor Platform, developed for unmanned aerial missions in the fields of documentation, coordination, exploration, surveying, communication, inspection and observation [MD, 2011c].

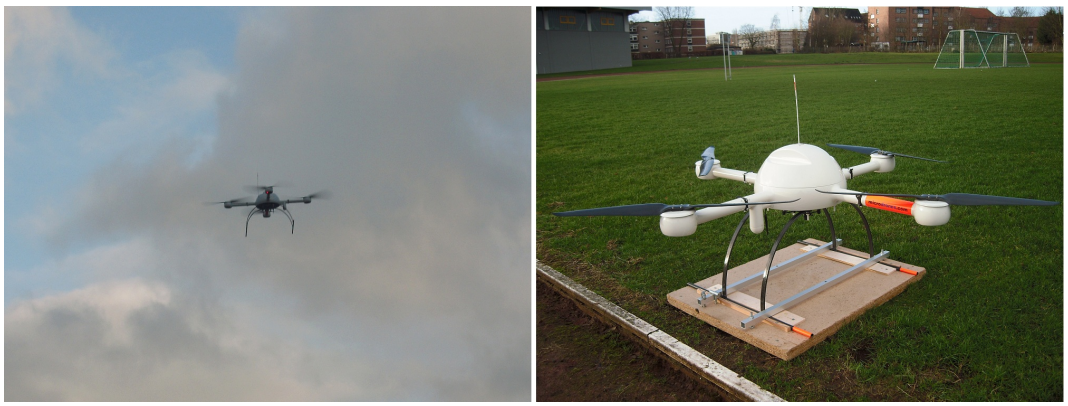

Figure 2.1.: MD4-1000 Sensor Platform in the air and on the ground.

The MD4-1000 is a remote controlled (RC) MUAV with four electrically operated rotors (see Figure 2.1). Due to its dimensions of 1030 mm from rotor shaft to rotor shaft and its weight of appr. 2650 g, this MUAV is capable of carrying a maximum

sensor payload mass of 1200 g at a flight time of up to 70 minutes. Detailed technical information about the MD4-1000 is given in Table 2.1.

Technical Specifications	
Climb rate	7.5 $\frac{m}{s}$
Cruising speed	15 $\frac{m}{s}$
Peak thrust	118 N
Vehicle mass	approx. 2650 g (dep. on configuration)
Recommended payload mass	800 g
Maximum payload mass	1200 g
Maximum take-off weight	5550 g
Dimensions	1030 mm from rotor shaft to rotor shaft
Flight time	up to 70 minutes (dep. on load/wind/battery)
Battery	22.2 V, 6S2P 12.2 Ah or 6S3P 18.3 Ah LiPo
Operational Conditions	
Temperature	−10 °C to +50 °C
Humidity	max. 90 %
Wind tolerance	steady pictures up to 6 $\frac{m}{s}$
Flight radius	min. 500 m on RC, with WP up to 40 km
Ceiling altitude	up to 1000 m
Take-off altitude	up to 4000 m ASL

Table 2.1.: Technical specifications and operational conditions of the MD4-1000 (following [MD, 2011b]).

The flight characteristics of the MD4-1000 resemble those of a helicopter with its possibility to roll, pitch and yaw, as well as to take off and land vertically. Figure 2.2 shows that in contrast to a helicopter's single rotor system, rolling, pitching and yawing of a MD4-1000 is managed by using its four inter-coordinated rotors at different rotary speeds [Büchi, 2010; p. 9].

Since the stabilization of the MUAV is achieved through adjusting each rotor's rotary speed individually, this would become far too complex for manual handling by a human pilot. As a consequence, a MUAV is equipped with several navigation sensors and a microcontroller unit to exploit the sensor information for a constant determination of the MUAV's position and attitude [Büchi, 2010; p. 14]. This information is interpreted by the microcontroller and processed to rotor control commands in order to guarantee stable maneuver in flight, position hold and, moreover, to execute predefined flight plan programs autonomously. Due to these facts, the sensors are essential components for the navigation of a MUAV and, therefore, briefly intro-

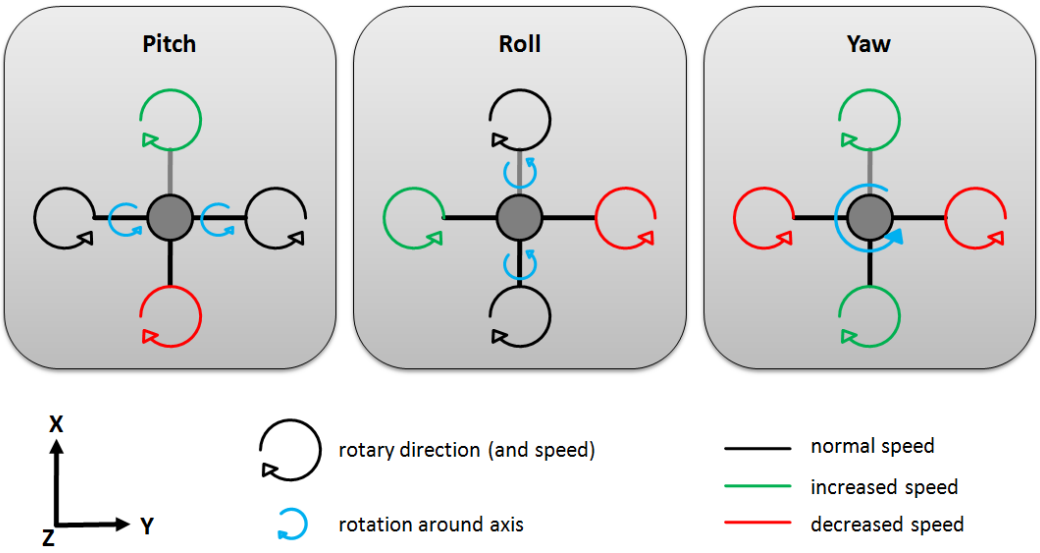

Figure 2.2.: Attitude of a MUAV at different rotary speeds.

duced in the following three sections. Subsequent to these sections, the integration of the individual sensors to a combined navigation system is described followed by auxiliary information about the MD4-1000 Sensor Platform System.

2.1.1. Inertial Navigation System

The Inertial Navigation System (INS) constitutes one of the key components of every MUAV. Inertial navigation is a self-contained navigation technique, in which measurements provided by accelerometers and gyroscopes are used to track the position and orientation of an object relative to a known starting point, orientation and velocity [Woodman, 2007]. It is in principle a dead reckoning navigation technique, based on a combination of a microcontroller for processing and an Inertial Measurement Unit (IMU) for measuring sensor values.

As modern Micro Electro Mechanical System (MEMS) sensors are small, lightweight and cheap in production, these MEMS are the preferred inertial sensors of Micro UAVs in todays use [Hoffmann, 2010]. Accordingly, these sensors are used in the MD4-1000 as well, which is equipped with three orthogonally aligned rate gyroscopes to measure angular velocities and three also orthogonally aligned accelerometers to detect linear accelerations. In combination, these sensors form the MD4-1000's IMU (see Figure 2.3).

In order to determine position and orientation, the microdrones Navigation Controller (NC) makes use of the sensor information provided by the IMU and processes this information via a strapdown inertial navigation algorithm. This algorithm

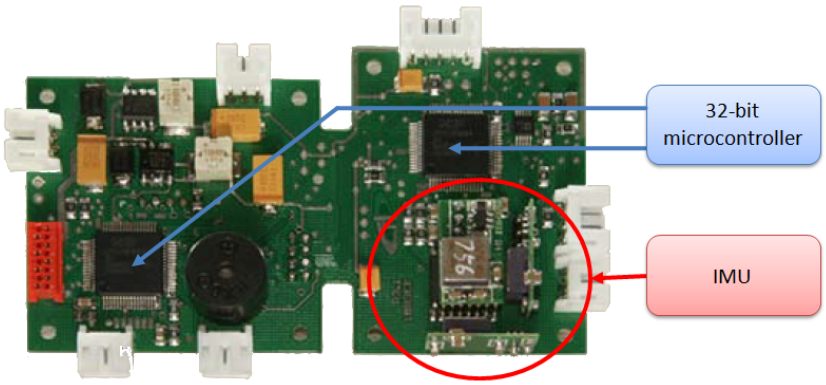

Figure 2.3.: Avionics of a MD4-1000 with IMU and two 32-bit microcontrollers [MD, 2011b].

is designed to calculate the current navigation solution by using detected angular velocities and accelerations of a preceding point in time [Wendel, 2007; p. 45].

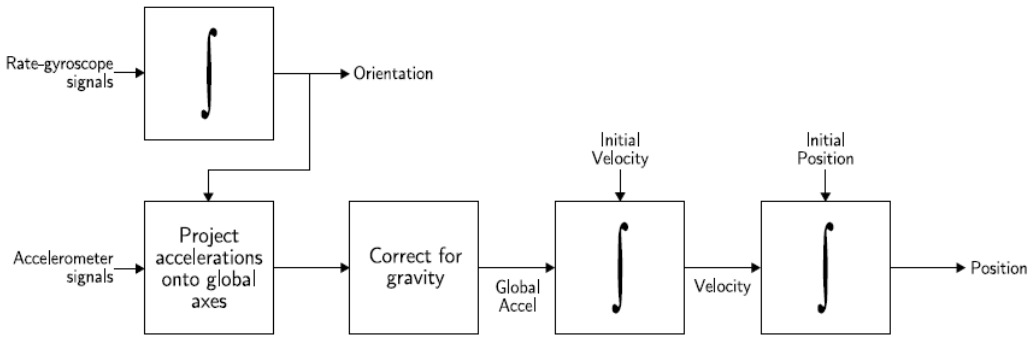

Figure 2.4.: Strapdown inertial navigation algorithm [Woodman, 2007].

Figure 2.4 depicts a basic overview of a strapdown algorithm. The MUAV's orientation change, relative to the preceding measurement, is tracked by integrating the angular velocity signals, obtained from the rate gyroscopes. The change in position is exploited in several steps. First of all, the acceleration signals gathered from the accelerometers are projected into a global frame of reference. Second, acceleration due to gravity is subtracted. Third, the remaining accelerations are integrated twice; to obtain velocity by a first integration and to gain displacement by a second one [Woodman, 2007]. More detailed information about the IMU, the strapdown algorithm and the different frames of reference, which are involved, can be found for instance in [Hoffmann, 2010], [Wendel, 2007] and [Woodman, 2007].

2.1.2. Global Navigation Satellite System

Since the INS provides relative position and attitude information related to an inertial starting point, additional navigation information is needed to determine absolute spatial orientation. Hence, the MD4-1000 is equipped with a GNSS receiver to obtain satellite based position information to determine the starting point in an earth centered, earth fixed reference frame.

The built-in receiver of the MD4-1000 at hand is a standard low-cost ublox LEA-4H module, connected to a passive antenna. The system is capable of tracking satellites from the US-American Global Positioning System (GPS) and to calculate its position by exploiting the coarse/acquisition (C/A) code, which is modulated to the L1 carrier signal distributed by the GPS satellites. Besides, this configuration is able to gather signal corrections broadcasted by Satellite Based Augmentation Systems (SBAS), such as the European Geostationary Navigation Overlay Service (EGNOS) and to process them in the positioning algorithm. This setting is already a satellite based augmented Differential GPS (DGPS) solution that ensures 2.0 m CEP50[1] accuracy [Ublox, 2007]. A detailed description of its evaluated accuracy can be found in Section 6.1.3. For further reading about the basic GNSS positioning principle please refer to [Bauer, 2003], [Wendel, 2007] and [El-Rabbany, 2002].

2.1.3. Magnetometer and Barometer

In addition to the above mentioned navigation sensors, the MD4-1000 is equipped with two further sensors, i.e. a magnetometer and a barometer, backing the INS and GNSS solution. The following paragraphs will briefly explain the functionality of these two sensors.

A standard MUAV magnetometer is a MEMS 3-axis magnetic field sensor. By using a coarse position information for the determination of the Earth's magnetic field, in combination with updated attitude data of the MUAV, the magnetometer forms a tilt compensated compass. It therefore provides absolute orientation of the yaw-vector to the MUAV navigation system [Wendel, 2007; p. 285].

As the accuracy of the altitude information of the MUAV navigation solution can be increased by a barometric sensor, such a sensor is used in the MD4-1000 [MD, 2011b] to gather information by detecting changes in the atmospheric air pressure. These changes in air pressure are converted to relative height measurements related to

[1] CEP50 = Circular Error Probability: The radius of a horizontal circle, centered at the antenna's true position, containing 50 % of the fixes.

the starting height to support the navigation system's calculation of altitude [Buss, 2011].

2.1.4. Navigation Sensor Integration

Integrated navigation is a combination of different navigation sensors and navigation principles. By combining different sensors and principles, disadvantages of using a single sensor or principle can be compensated. As a result, the MUAV navigation system is characterized by a higher redundancy and performance [Wendel, 2007; p. 191].

All aforementioned navigation sensors and navigation principles are integrated in the navigation system of the MD4-1000. Figure 2.5 shows a work flow of how these different components act together, to create a reliable navigation solution.

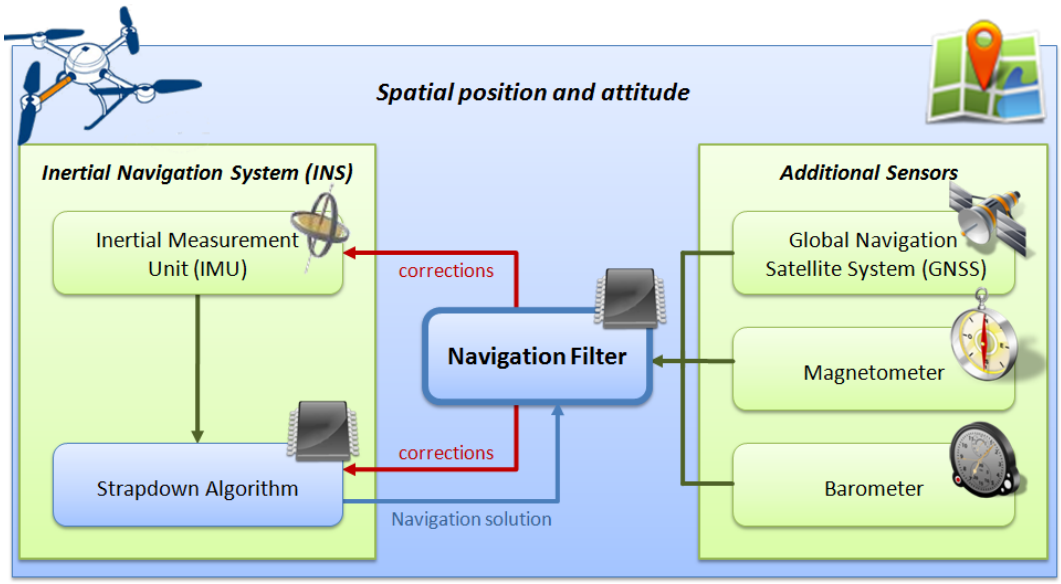

Figure 2.5.: Determination of spatial position and attitude by integrating different navigation sensors and principles.

The central element of this integrated navigation system is a collection of navigation filters, which are run on the microdrones NC. These filters, which are algorithms, combine and process sensor data provided by the different navigation sensors. The INS, which is placed on the left side of this overview, is part of the main calculation loop. The relative position and attitude values, calculated by the strapdown algorithm, form a first navigation solution for the MUAV's navigation system. As MEMS gyroscopes and accelerometers possess errors, which increase over time [Woodman, 2007], position and velocity information provided by the GNSS, is integrated into the navigation filters to support the INS. In addition, absolute yaw-angle values from the

magnetometer, as well as relative height changes from the barometer are supplied to the different filters. Combining all this information, the navigation filter algorithms calculate real-time corrections to re-adjust the INS on-the-fly. By using this technique, it is guaranteed that the best possible navigation solution is calculated. Moreover, if one sensor or navigation principle is omitted because of technical problems or reasons of inaccuracy, the redundancy of navigation filters will compensate this loss of information [Wendel, 2007; p.283].

2.1.5. Sensors as Payload

To complete the MD4-1000 Sensor Platform System, this MUAV has increased payload capability (see 2.1), in order to be able to mount additional sensors for detecting phenomena of many kinds. Thus, the MUAV is equipped with a payload take-up system on its bottom side, to attach carbon fiber mounting frames customized for the individual sensor types (see Figure 2.6). In the majority of cases, camera systems are mounted to gather visual data on-the-fly [MD, 2011b]. Additionally, this mounting system offers functionality to install any kind of low-weight sensor, e.g. environmentally sensing devices for the detection of humidity, temperature, gas concentration and many more.

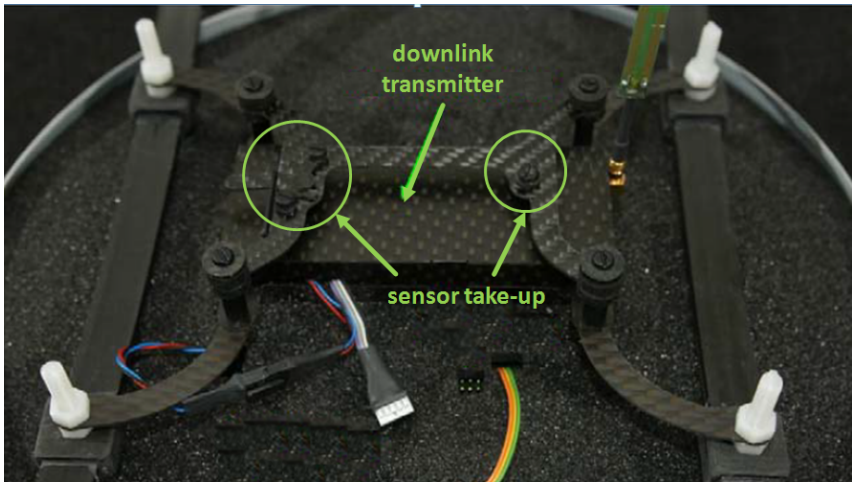

Figure 2.6.: Payload take-up system and downlink transmitter of a MD4-1000 (following [MD, 2011b]).

2.1.6. Telemetry

The MD4-1000 is able to broadcast a system downlink message, which provides sensor information about its navigation mode, machine status, RC values, motors, timers, position, speed, attitude, altitude, temperature, magnetometer, distance to

its first position fix, waypoints and payload, as well as a video signal from a potentially mounted optical camera system [MD, 2010b]. Therefore, a downlink transmitter is used (see Figure 2.6), which operates in a frequency range of 2.2 GHz to 2.6 GHz with a low signal runtime of approximately 40 ms [MD, 2011b].

The transmitted signals are received by a base receiver station, which automatically preprocesses the telemetry signal of the MUAV, for the use in a USB-connected notebook. A special mdCockpit software, which has been installed on the notebook, enables a real-time surveillance of all flight parameters and provides the downlink message in two ways for subsequent processing [MD, 2010a]. The downlink message is stored in a log file on the notebook's hard disk following predefined formatting rules, which can be seen in [MD, 2010b]. In addition, the mdCockpit application program is implementing a named pipe server for querying data in real time [MD, 2011a].

So far, the most important technical capabilities of the MD4-1000 Sensor Platform have been identified in the preceding sections. Moreover, it was shown how the interaction of all navigation components leads to a reliable navigation solution. As this thesis does not cover the complete range of possibilities to improve the navigation of a MUAV, it concentrates on the improvement of one single component of the MD4-1000's navigation system; the determination of absolute position information via DGNSS. The principle of DGNSS will be introduced in the following section.

2.2. Differential Global Navigation Satellite Systems

With the upcoming of several emancipated Global Navigation Satellite Systems, such as the US-American GPS, the Russian Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS) and the developing systems GALILEO (European Union), as well as the BeiDou/Compass (China), it is important to clarify that GNSS specifies not only the commonly used GPS. GNSS is used as a rather general term for using any of these satellite systems or a combination of several, when determining position.

Since the prevailing absolute GNSS position solutions are defective due to satellite clock errors, atmospheric refraction of the signals and errors in the satellites' orbits propagation, it is crucial to compensate or at least to decrease these errors, for calculating a more accurate position. Hence, relative positioning methods can be used to reduce the effect of these errors by subtracting code or carrier phase measurements of corresponding satellite signals, which have been detected simultaneously on different receivers [Bauer, 2003; p. 221]. The following section will concentrate on

the usage of code measurements by a technique widely known as Differential GNSS or DGNSS.

2.2.1. Working Principle

The idea of DGNSS is the calculation and transmission of C/A code corrections by a reference station to improve a user's DGNSS receiver's L1 pseudorange determination. By correcting pseudorange measurements in the user station's positioning algorithm, satellite clock errors are completely eliminated, while ionospheric runtime errors, as well as negative impacts from defective satellite orbits are reduced. Additionally, multipath and noise effects are minimized through L1 carrier phase smoothing algorithms, both on the reference station's side and the user station's side [Bauer, 2003; p. 223]. As a consequence, horizontal position accuracy can be improved to approximately 0.4 m [Kettemann, 2003].

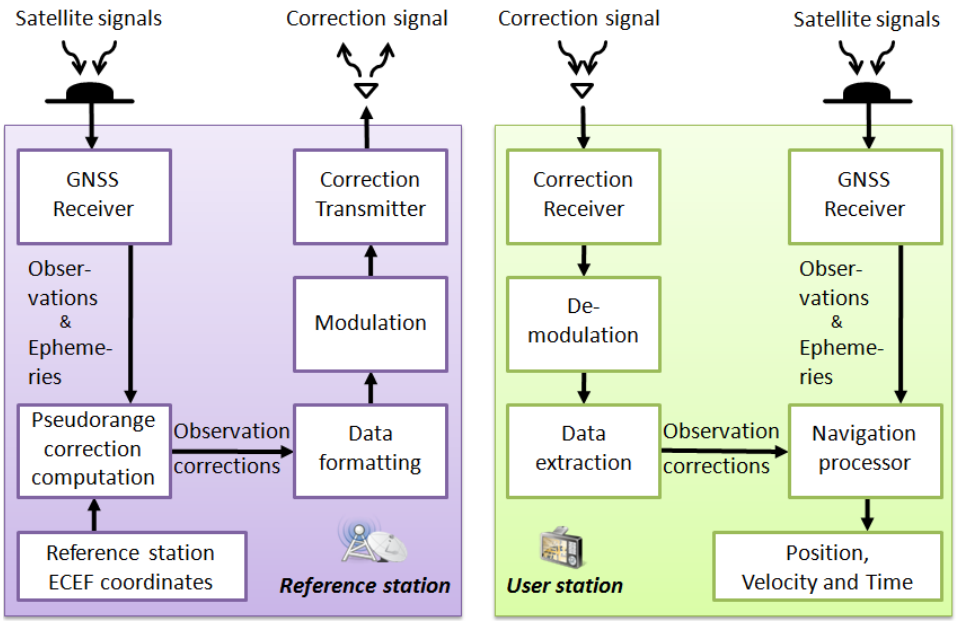

Figure 2.7.: DGNSS data flow between reference station and user station (following [Bauer, 2003; p. 223]).

Figure 2.7 shows the data flow of the DGNSS working principle. In the following, the requirements and functions of a reference station, as depicted in the purple box, will be described. A reference station uses a GNSS receiver to track satellite signals broadcasted by the GNSS satellites. The retrieved observations and ephemeries are used to calculate a pseudorange based position of the reference station. This calculated position is then compared to the true position of the reference station. The bias between these two positions is used to calculate corrections for the tracked pseudorange solutions. These observation corrections then help to obtain more accurate

coordinates when applied on the user station. Therefore, the observation corrections are transformed to a standardized correction data exchange format and modulated to a correction transmitter's signal which is broadcasted to the user.

The user station, depicted in the green box, is acquiring the correction signal by a correction receiver and extracts the correction data after its demodulation from the signal. The observation corrections are subsequently fed to a navigation processor and thereby combined with observations and ephemeries that have been tracked by the user station's GNSS receiver. The navigation processor then computes improved position, velocity and time of the user station, by using corresponding pseudoranges and corrections.

An important underlying assumption for this technique is that errors, which have been determined on the reference station's side, are equal to those of the user station's side, as long as the user's position is situated no farther than a few hundred kilometers from the reference station [Bauer, 2003; p. 224]. Otherwise, the calculated corrections are not modeled in a sufficient accuracy and cannot be used to compensate the true errors, which appear at the user's location. The next subsection will demonstrate, how this crucial assumption is often achieved by using correction signal services that operate their own reference station networks.

2.2.2. Correction Signal Services

Correction Signal Services can be divided into two categories, depending on their reference station network's scale and their correction signal broadcasting technique. Services, which use geostationary satellites to broadcast their correction signals over wide areas are known as Satellite Based Augmentation Systems (SBAS). However, services using terrestrial communication on local or regional level are called Ground Based Augmentation Systems (GBAS) [Dodel and Häupler, 2009; p. 219].

Up until now, SBASs, such as the Wide Area Augmentation System (WAAS), the European Geostationary Overlay Service (EGNOS) or the Multi-functional Satellite Augmentation System (MSAS) function as augmentation for the US-American GPS. As they are designed to provide code corrections for a wide service area, the expected position accuracy is around 1.5 m [EGNOS, 2012]. The above mentioned and in Section 2.1.2 quoted MD4-1000's standard GNSS module achieves for instance its best position accuracy of 2.0 m CEP by using EGNOS.

Figure 2.8 illustrates the concept of a SBAS like EGNOS. Such a system can be roughly divided into two segments, the space and the ground segment. The first part of the space segment consists of the GNSS satellites, which broadcast their

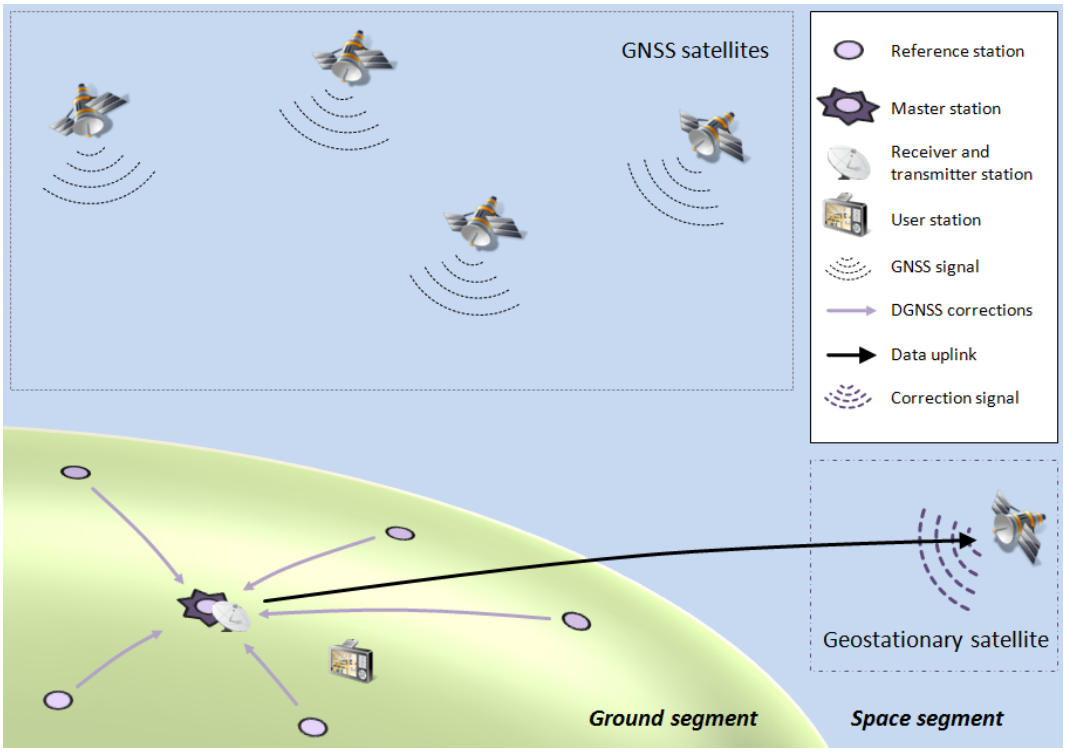

Figure 2.8.: SBAS working principle and satellite based correction signal reception (following [El-Rabbany, 2002; p. 97]).

signals to a network of spatial distributed reference stations on Earth. As every reference station knows its own exact position, it is able to preprocess the observation measurements, to compute signal corrections for the GNSS satellites, which can be tracked in its region. The preprocessing follows the working principle mentioned in Section 2.2.1. The computed information is forwarded to a master station via a terrestrial link and subsequently analyzed and combined to determine a number of correction parameters for each GNSS satellite, which are effective within the system coverage area. In a next step, these parameters are uploaded to the geostationary satellites, which form the second part of the space segment. The parameters are then rebroadcasted to Earth, to provide user stations with DGNSS correction sets, which are valid for their specific locations [El-Rabbany, 2002; p. 96].

In contrast to satellite based broadcasting, GBASs distribute their correction signal via terrestrial techniques like e.g. radio (100 kHz - 300 MHz) or mobile communication networks (450, 900, 1800 MHz) [Zogg, 2011; p. 109]. Furthermore, the services' coverage areas are smaller and their reference stations' density is higher, compared to SBAS services. As the distance between reference station and user station plays a decisive role in the determination of correction parameters, the accuracy of GBASs based on code corrections is increased by up to 300 % compared to SBASs, depending on the GNSS receiver's hardware (see i.e. [ASCOS, 2009]).

GBASs are commonly operated at an area of country level or federal state level, like for instance the German SAPOS (*Satellitenpositionierungsdienst der deutschen Landesvermessung*) , ascos Satellite Positionig Services (SPS) or Radiobeacon DGPS, which is operated by the German Maritime Authority (*Wasser- und Schifffahrtsverwaltung*).

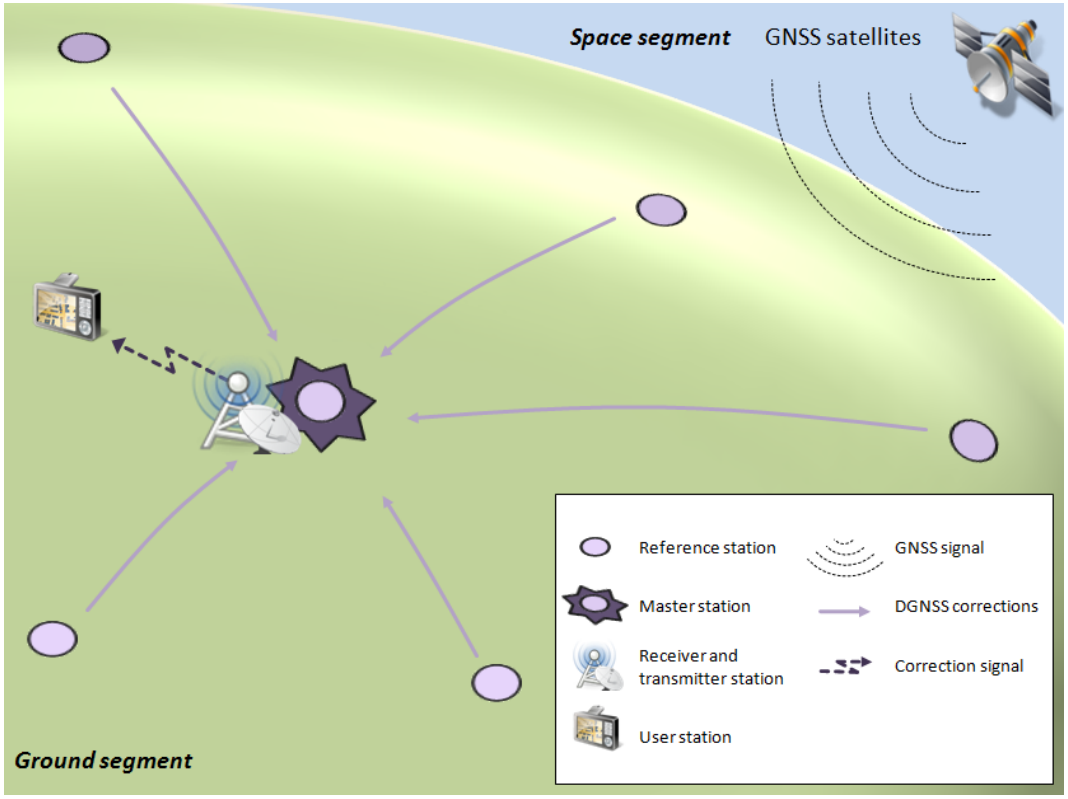

Figure 2.9.: GBAS working principle and ground based correction signal reception (following [El-Rabbany, 2002; p. 97]).

As Figure 2.9 shows, the main difference in the working principle compared to SBASs lies in the omission of geostationary satellites for broadcasting correction signals. Here, terrestrial transmitter stations are used to provide user stations with the needed corrections. All other parts of the data flow and data processing are treated in analogy to the methodology described in Section 2.2.1 and the preceding paragraphs.

After evaluating the basic concept of DGNSS, as well as the major differences in broadcast and accuracy of correction signal services, the next section of this work will focus on the theoretical background for the predominantly software related part of this thesis and give an overview of the Sensor Web Enablement Initiative of the Open Geospatial Consortium Inc..

2.3. Sensor Web Enablement

Access to sensor data of the MD4-1000 Sensor Platform currently depends on specific interfaces and software provided by Microdrones GmbH. These do not follow any conventional standards. Hence, it is of interest to increase the MD4-1000's potential by the standardization of its sensor data to facilitate its integration into various application sytems.

A multitude of such systems can be found in domains, such as environmental monitioring, public security, risk monitoring, disaster management and traffic management. As these systems strongly rely on sensors, access to and communication with the sensors in a standardized way is required [Jirka et al., 2010]. The problem of standardization is addressed by the OGC's SWE initiative, which refers to web accessible sensor networks and archived sensor data. The SWE builds the basis for the Sensor Web as a coordinated observation infrastructure and facilitates discovery, access and, where applicable, control of sensor networks and sensor data by using open standard protocols and interfaces [Botts et al., 2008]. Therefore, the following section will give a basic understanding of the SWE framework. Subsequently, Sensor Bus and Sensor Platform Framework, as suitable means for the standardization and integration of the MD4-1000's Sensor Platform, are introduced.

2.3.1. SWE Architecture

The development of the SWE architecture has been driven by an OGC working group and resulted in a SWE framework of open standars, for exploiting web-connected sensor and sensor systems of all types. These standards enable the web-based discovery of sensor systems, observations and observation processes. Furthermore, the standards support exchange, and processing of sensor observations, and in addition, the tasking of sensor systems [Botts et al., 2008].

The functionality of the framework includes access to sensor parameters and retrieval of observations and coverages [Botts et al., 2008]. It provides a common data format for sensor data and a metadata model for the description of sensors and the related measurement processes. In addition, the framework enables subscription to and publishing of alerts, which are defined by certain criteria and, furthermore, offers mechanisms for tasking and controlling sensors [Jirka et al., 2010]. These requirements are realized by offering two sets of standards, the information model and the service model (see Figure 2.10). The information model contains standardized data models and formats for encoding sensor data, as well as metadata. According to the information model, the service model provides several web service interfaces,

which come along with the needed funtictionality of the Sensor Web [Jirka et al., 2010]. The most important standards for understanding the approach of integrating the MD4-1000 Sensor Platform into the Sensor Web are briefly summarized in the following paragraphs. Information about the unmentioned standards is given in [Botts et al., 2008] and [OGC, 2012].

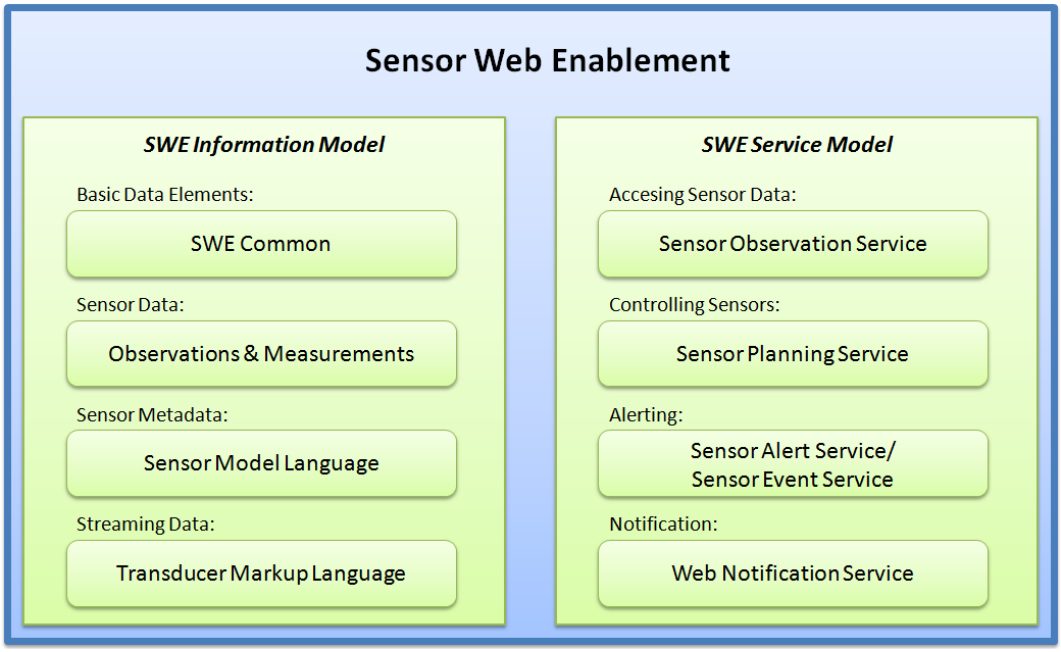

Figure 2.10.: Overview of the SWE framework with its information and service model (following [Jirka et al., 2010]).

SWE Common

The underlying idea of SWE Common is to provide a general basis of data elements constisting of elementary data building blocks, which are re-used by all other standards of the SWE framework [Jirka et al., 2010].

Observations & Measurements

The Observations & Measurements (O&M) standard is designed for the exchange of sensor data. Therefore, it is a crucial element of the SWE framework, as it defines a conceptual model and XML encoding for real-time, as well as archived observations and measurements. An O&M document consists of one or severeal observations, which are determined as action and result in values, describing phenomena. An observation is modelled as a feature and binds a result to a feature of interest, upon which the observation was made [OGC, 2007a].

Sensor Model Language

The Sensor Model Language (SensorML) is the sensor metadata standard. Its main function is to provide descriptions of sensors and sensor systems for inventory management and to provide sensor and process information in support of resource and observation discovery. Moreover, it supports the processing and analysis of sensor observations and the geolocation of observed values via the provision of performance characteristics and gives explicit description of the observation process [OGC, 2007b].

Sensor Observation Service

The aim of the Sensor Observation Service (SOS) is to provide access to observations from sensors and sensor systems in a standardized way, which is consistent for all sensor systems, including remote, in-situ, fixed and mobile sensors [OGC, 2007c]. SOS is a standard web service interface for requesting, filtering, and retrieving observations and sensor system information and is therefore regarded as intermediate piece between a client and the sensor data. There are two ways of retrieving the sensor data; from an observation repository or a near real-time sensor channel, such as the MD4-1000 [Botts et al., 2008]. Thus, the SOS is based on the aforementioned O&M specification for modelling sensor observations and the SensorML specification for modelling sensor and sensor systems [OGC, 2007c].

Sensor Alert Service and Sensor Event Service

The Sensor Alert Service (SAS) and Sensor Event Service (SES) are two different approaches for defining alert criteria and subscription to alerts. As many sensors are used for building monitoring systems, which dispatch alert notifications in the case of predefined conditions, the SAS and SES offer the potential to filter sensor data by these conditions [Jirka et al., 2010]. The difference in SAS and SES can be seen in the SES's ability to define alert criterias, for instance at the occurence of a critical observation value after the occurence of a preceding non-critical value. This ability is called complex event processing [Rieke, 2010].

2.3.2. Sensor Bus

In general, the integration of SWE services and sensor devices is still executed manually, due to a conceptual gap between these two layers. As this leads to extensive efforts in integrating new services and sensors, this approach is contrary to the aim

of reaching interoperability. A solution for this problem is the Sensor Bus, which is an intermediary layer for a seamless integration of sensor networks and Sensor Web [Broering et al., 2010].

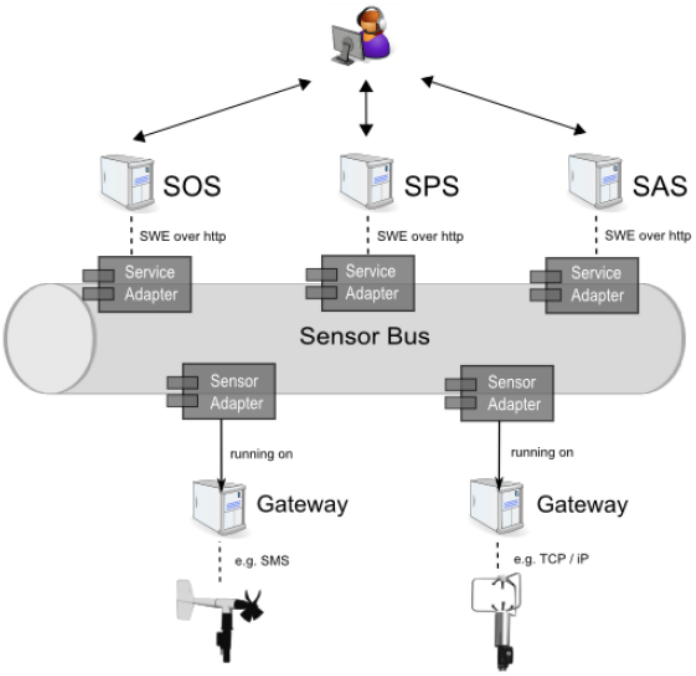

Figure 2.11.: Sensor Bus architecture with Service Adapter and Sensor Adapter interfaces [Broering et al., 2010].

The Sensor Bus as intermediary layer is externally designed as a logical bus [Broering et al., 2010] and shown in Figure 2.11. This logical bus is a topology that follows the Message Bus pattern [Hohpe and Woolf, 2003; p. 139] and defines the communication between network components without regard to their phyisical interconnection. It serves with a common communication infrastructure for network components to publish messages to the bus or to subscribe to the bus for receiving messages, in a push-based communication style. These components, SWE services and sensors, can subscribe and publish through interfaces, for which pluggable adapters can be developed. These adapters guarantee the conversion from the service or sensor specific communication protocol to the internal bus protocol [Broering et al., 2010].

Interactions between the sensor network layer, the Sensor Web and the intermediary layer are realized through particular bus messages; the format of which is compact to preserve bandwidth and system resources [Broering et al., 2010]. Table 2.2 lists the Sensor Bus messages for the subscription of components.

To register a sensor at the Sensor Bus and publish its existence, the `RegSen` message is sent. Additional parameters specify a unique sensor's ID and a Uniform Resource

Interaction	Bus Message Protocol
Sensor Registration	`RegSen*<sensor id>*<sensor description URL>`
Data Publication	`PubData*<sensor id>*<time tag>*<data>`
Service Registration	1. `RegServ*<service URL>` 2. `SubServ*<service URL>*<sensor A id>` 3. `SubServ*<service URL>*<sensor B id>` ...

Table 2.2.: Selected Sensor Bus Message Protocols (following [Broering et al., 2010] and [Rieke, 2010]).

Locator (URL), pointing to a SensorML document, which contains the sensor's description.
After having subscribed to the Sensor Bus, the sensor adapter transmits the `PubData` message via the Sensor Bus with information about the sensors' observed data.
The subscription of a Sensor Web Service follows this principle in a sequential approach. In a first step, `RegServ` is called to publish the service's URL. In a next step, one or several `SubServ` messages are sent over the bus to subscribe the service for one specific or several sensors [Broering et al., 2010].

Besides these messages for data acquisition, the Sensor Bus offers the possibility to control sensors by Sensor Planning Services. As this function is not applied in the integration of the MD4-1000 Sensor Platform, the corresponding messages are not explained. Further information about these messages is given in [Broering et al., 2010].

2.3.3. Sensor Platform Framework

After having established the Sensor Bus technology, the Sensor Platform Framework[2] (SPF) will now be introduced. The SPF offers two basic functions. It is able to synchronize registered sensor data streams, as for instance the MD4-1000's navigation data stream with additional data streams of mounted sensor devices. Moreover, the resulting data is published on the Web and made accessible in an interoperable way, using the Sensor Bus technology [Rieke et al., 2011].

[2]Sensor Platform Framework, published at 52 °North Initiative for Geospatial Open Source Software GmbH (http://52north.org/communities/sensorweb/incubation/ifgicopter/spf/index.html)

The SPF features both, reusable design along with a reusable implementation and a high coherence between the framework classes [Rieke et al., 2011]. Its key characteristics are extension points, which serve as interfaces for data source Input-Plugins and output component Output-Plugins, which can be written in a flexible manner. An overview of the framework's architecture is given in Figure 2.12.

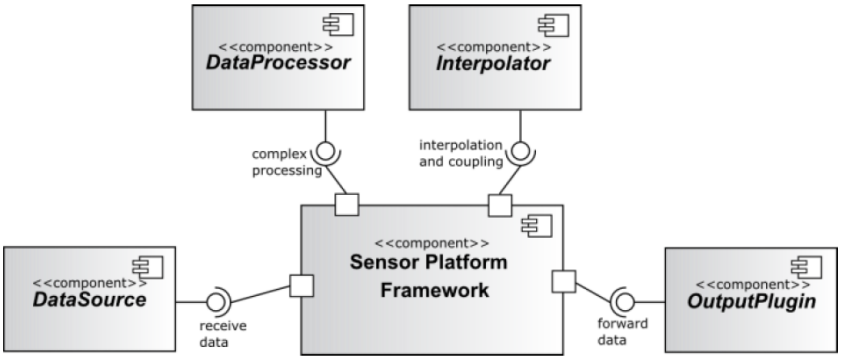

Figure 2.12.: Sensor Platform Framework model with extension points [Rieke et al., 2011].

In a first step, the sensors need to be connected to the framework. This task is carried out by a data source Input-Plugin. This Input-Plugin establishes connection to the framework and processes the incoming data to give the framework access to the observations and their metadata encoded in SensorML. Moreover, the Input-Plugin contains a XML document, which specifies the plugin-specific behaviour of the framework; especially the generation of output data based on a periodic time interval or on the availability of a specific phenomenon [Rieke, 2010].

The framework's core is designed to manage incoming data of registered data sources, by collecting data as soon as it is available. This data is added to the frameworks InputPluginCollector class, which gatheres all incoming data. The InputPluginCollector class interpolates the data according to the aforementioned plugin-specific behaviour. By calling the doOutput method, the processed data is forwarded to the FrameworkEngine class for consequent data output by a suitable Output-Plugin [Rieke, 2010].

The output plugin interface is also designed as an extension point to the framework, to support multiple output formats [Rieke, 2010]. An implemented Output-Plugin retrieves the processed data from the FrameworkEngine class and converts it to a specific format. As this work concentrates on the integration of the MD4-1000's navigation data into the Sensor Web, the output format needs to set focus on the SWE standards, which have been introduced in Section 2.3.1.

3. Requirement Analysis

Regarding the aim of improving the current standard absolute positioning solution, this chapter defines the essential demands, which are made on an improved DGNSS positioning system to reach a valuable accuracy and a best possible integration into the MD4-1000 Sensor Platform's navigation system. Moreover, the requirements of a suitable integration of the sensor platform's navigation data into the Sensor Web in light of a subsequent use by Sensor Web Services are exposed.

3.1. DGNSS Positioning System Requirements

The following sections will demonstrate the predominantly hardware related requirements of an improved DGNSS positioning system, as well as taking into consideration the demands on the interface between DGNSS receiver and the MD4-1000's NC. This information forms the basis for the selection and combination of the positioning system's components, which will be shown in detail in Section 4.1.

3.1.1. Improvement of Absolute Positioning

The main objective of the advanced positioning system is to improve its absolute position solution accuracy. A best possible accuracy of approximately 0.4 m, as mentioned in Section 2.2.1 and based on the introduced GBAS techniques in Section 2.2.2, is aspired. Ideally, the system should make use of all available fully, operational GNSS (GPS and GLONASS) as of January 2012 to improve the measurement's Dilution of Precision (DOP) values by increasing the amount of visible GNSS satellites [Tamazin, 2011]. This way, negative impacts on positioning are reduced in locations where several GNSS satellites cannot be taken into account, due to shadowing effects that are caused by obstacles.

Furthermore, the positioning system should be capable of automatically switching to less accurate DGNSS support by using adequate SBASs in the case that a suitable GBAS is not available in the area of interest.

3.1.2. Extensibility Through More Sophisticated Technologies

Besides the demanded implementation of DGNSS techniques, the positioning system should be conceptualized in a modular way. As the development of improved hardware and software is rapidly progressing, it is important to avoid out of the box systems to offer the possibility of exchanging singular components by others, which are using more sophisticated technologies.

This requirement also aims at the extensibility of the positioning system, by using more accurate relative positioning support known as Real-Time Kinematic GNSS (RTK-GNSS) or Precise DGNSS (PDGNSS). The underlying architecture of the mentioned support solution resemble those of a ground based DGNSS. The difference lies in the more superior and more expensive receiver's hardware and software in combination with more efficient antennas. These receivers are capable of processing C/A code and carrier phase observations on both L1 and L2 signals and adapted correction signal sets. As a consequence, atmospheric corrections are eliminated more precisely and signal runtimes are determined in a higher degree of accuracy, which leads to more improved positioning at cm-level [Bauer, 2003; p. 224].

3.1.3. Real-Time Accuracy Estimation for Safety in Flight Reasons

Another important aspect of the positioning system is the ability of estimating a reliable position accuracy in real-time. Since the the MD4-1000 is an aerial vehicle used for monitoring and sensing in close range of objects of interest, it is essential to avoid crashes and consequential damages to the MUAV and the sensed objects. Hence, the MD4-1000's navigation system is able to alter its interior positioning routines. In cases that the GNSS position exceeds a predefined accuracy threshold or completely looses position, the navigation system switches to navigation filters, which are based on the remaining navigation sensors' information [Wendel, 2007; p. 287].

3.1.4. Low Weight and Low Power Consumption Components

Taking into account, that the flight time and the payload capability of a MUAV heavily depend on the vehicle's mass and its battery power (see 2.1), it is evident that the new electronic components should follow the principle of minimization. Therefore, the power consumption and moreover, the dimension and weight of the components should be kept low to have as less impact as possible on the payload capability and flight time.

3.1.5. Integration into MD4-1000 Navigation System

The integration into the MUAV demands some MD4-1000 specific requirements the positioning system has to function with. Among these requirements, is on the one hand need for an appropriate interface between the positioning system and the MD4-1000 NC. As the NC offers a serial interface for data connectivity, the positioning system has to serve with a suitable communication port to forward its data to the NC. On the other hand, the system's data output rate has to work at an adequate frequency of 4 Hz, in order to be in coordination with the MD4-1000's internal navigation filters. The NC's firmware, moreover, necessitates a certain positioning information, the positioning system has to provide. This information and the predefined internal data structure this information has to be applied to, is shown in Listing 3.1.

Listing 3.1: Internal navigation data structure of a MD4-1000, containing GNSS information values for time, position, velocity, accuracy and status.

```
struct navsol_msg
{
  U32 itow;    // [ms]  time of week
  S32 frac;    // [ns]  fractional remainder of rounded itow
  S16 week;    //       GPS week
  U08 gpsfix;  //       GPS fix type (0=none, 2=2D, 3=3D)
  U08 flags;   //       Bit3: itow valid, Bit2: week valid
               //       Bit1: differential used, Bit0: fix valid
  S32 ecef_x;  // [cm]  ECEF x coordinate
  S32 ecef_y;  // [cm]  ECEF y coordinate
  S32 ecef_z;  // [cm]  ECEF z coordinate
  U32 pacc;    // [cm]  3D position accuracy estimation
  S32 ecef_vx; // [cm/s] ECEF x velocity
  S32 ecef_vy; // [cm/s] ECEF y velocity
  S32 ecef_vz; // [cm/s] ECEF z velocity
  U32 sacc;    // [cm/s] 3D speed accuracy estimation
  U16 pdop;    //       current PDOP
  U08 res1;    //       reserved
  U08 num_sv;  //       number of visible sv
  U32 res2;    //       reserved
};
```

3.2. MD4-1000 Sensor Data Integration Requirements

Besides from focusing on the requirements of an improved positioning system, this thesis also addresses the integration of the MD4-1000's sensor data in the Sensor Web. The Sensor Web as a coordinated observation infrastructure builds on service-oriented interfaces, to provide observation data and metadata of sensor systems. Section 2.3.1 showed that these interfaces require sensor information and observation

data in a standardized way. Consequently, these standards must be considered for the conception of a suitable MD4-1000 sensor data architecture. This consideration resulted in requirements, which have been taken into account in the design of the architecture and will be described in the following sections.

3.2.1. Near Real-Time Data Downlink

The O&M standard (see Section 2.3.1) allows the exchange of real-time and archived observations. As the MD4-1000 is a mobile aerial sensor platform, the demand of real-time data access cannot be satisfied by convential wired networks. Therefore, the telemetry data downlink, introduced in Section 2.1.6, has to be taken into consideration. It offers a near real-time access to the sensor platform's data via a named pipe server. As a consequence, the MD4-1000's sensor data architecture has to serve with a software, which is able to retrieve the telemetry data stream via the named pipe server.

3.2.2. Standardized Sensor Data

Section 2.3 already addressed the problem of sensor data reusability, which occurs with manufacturer-specific interfaces and software. As the Sensor Bus architecture is based on the SWE initiative and its involved standard definitions, it is indispensable to establish a mechanism to encode the MD4-1000's sensor data in conformance with these definitions.

3.2.3. Sensor Data Access

After having considered retrieval and standardization of the MD4-1000's sensor data, access to this data needs to be established. Therefore, the sensor data architecture has to provide a possibility to register the MD4-1000 Sensor Platform and to publish its sensor data via the Sensor Web. As a consequence, SWSs of any kind are enabled to implement the provided sensor data in application systems of various domains.

3.3. Objectives

As a result of the aformentioned requirements, the objectives of this thesis are the improvement of the MD4-1000's absolute positioning accuracy and the provision of the enhanced sensor platform's navigation data for Sensor Web Services.

The improvement in positioning accuracy is based on a DGNSS approach. The standard positioning system's hardware will be exchanged by a more efficient one, which will follow MUAV-specific requirements, such as small size, low weight and low power consumption. The hardware will be able to provide real-time navigation accuracy estimations for safety in flight reasons and the integration into the MD4-1000 Sensor Platform will be guaranteed by modifying the NC's firmware.

Although this thesis focuses on improving the positioning accuracy, the MD4-1000's navigation data will be retrieved, standardized and made accessible in near real-time. Therefore, a software architecture will be developed, which decodes the MD4-1000's telemetry data stream, processes the sensor data in accordance to SWE standards and enables access to the sensor platform's information and observations by SWSs.

4. Concepts and Strategies

This chapter describes the approach, which has been made to increase the MD4-1000's absolute positioning accuracy and points out the underlying concept of the developed architecture. Besides, it gives an overview of the conceptual firmware modification to integrate the architecture into the existing system. The second part explains the strategies to relay the MUAV's positioning data to the Sensor Web and the possibility to synchronize this data with additional sensor data sources, by using the Sensor Platform Framework.

4.1. DGNSS Positioning System Architecture

This first section is concerned with the complete conceptual architecture of the improved positioning system. As a modular concept for extensibility is aspired (see Section 3.1.2), individual hardware components for gathering more accurate positioning information and for the acquisition of correction data are listed. The section further outlines the communication interface between the DGNSS positioning system and the MD4-1000's NC and a way to realize the data transfer.

4.1.1. DGNSS Receiver and Antenna

As a first step towards an improved positioning system, the existing GNSS hardware components have to be replaced by more effective ones, guaranteeing a better positioning solution. Therefore, the standard GNSS receiver, which has been introduced with its most significant features in Section 2.1.2 will be exchanged by an advanced DGNSS receiver, using GPS and GLONASS as it is demanded in Section 3.1.1. In addition, the receiver needs to be able to acquire and process correction signals provided by both, SBASs and GBASs to guarantee advanced positioning by switching to SBAS support in areas where GBASs cannot be retrieved. In cases that GBASs are availlable, the receiver also needs to offer a suitable serial interface to connect a radio modem that operates communication to ground based Correction Signal Services and reception of their provided correction signals.

Besides, the passive GPS antenna used at present, has to be exchanged by an active one with a higher gain and the possibility to receive both GPS and GLONASS signals.

4.1.2. Radio Modem and Communication With Correction Signal Service

After having established the needed functionality of a new receiver and antenna, it is obvious, that this new hardware needs to be able to retrieve aforementioned SBAS correction signals, which are broadcasted by geostationary satellites (see Section 2.2.2) to process signal corrections of lower accuracy.

In the case of using signal corrections of higher accuracy, an additional mean of communication between the receiver and an appropriate ground based Correction Signal Service is required. This mean forms a radio modem, capable of sending position information to a Correction Signal Service and also of receiving location-dependent correction data to forward it in real-time to the receiver for subsequent processing.

The underlying message format for sending position information is the National Marine Electronics Association 0183 Interface Standard (NMEA 0183) data transmission protocol [NMEA, 2002]. Listing 4.1 shows the required `$GPGGA` message, which contains the receiver's position information. The standard for receiving DGNSS signal corrections is defined by the Radio Technical Commission for Maritime Services (RTCM). As the radio modem is intended to establish connections via the General Packet Radio Service (GPRS), the standard's version that is going to be used is RTCM 10410.0 for Networked Transport of RTCM via Internet Protocol (Ntrip), which is more complex and can be seen in detail in [RTCM, 2004].

Listing 4.1: Example of a NMEA 0183 `$GPGGA` message.

```
1 $GPGGA,114217.43,5107.8321,N,00935.4613,E,1,09,1.8,63.2,M,24.7,M,,*6E
2       |        |          |            | |   |     |      |
3       UTC time Latitude   Longitude    | HDOP|     |      Checksum
4       HHMMSS.SS DDMM.MMMM DDDMM.MMMM   SV    Height Geoid height
```

4.1.3. MD4-1000 Firmware Integration

Having introduced an improved DGNSS receiver, antenna and a suitable radio modem, which together establish the wanted modular DGNSS hardware setup, the next step is the integration of the positioning information, gathered by this setup, into

the MD4-1000's firmware. Therefore, the receiver has to be connected to the MD4-1000's NC via a serial interface to push the processed data to the NC, where it is received and decoded. A modification of the NC's firmware has to be carried out, by implementing a parsing routine in the C programming language. This routine must be able to handle the incoming data stream and to split it into its individual information sets. Consequently, the information sets have to be parsed into the intended data fields which are listed in the NC's `navsol_msg` (see Listing 3.1).

At this point, the improved DGNSS setup is complete and is summarized in the following section.

4.1.4. DGNSS Positioning System Overview

Figure 4.1 gives an overview of the data flow of the DGNSS positioning system, which follows the explanations in Section 2.2.1. After the DGNSS receiver's first position fix, it forwards the position information as NMEA 0183 `$GPGGA` message via its serial interface to a connected radio modem. Having received this information, the radio modem builds up a GPRS connection to a local service provider, starts to register to the Correction Signal Service's system and hands over the `$GPGGA` string. There, the position information is extracted and signal corrections are determined for that specific location. In a consequent step, the processed signal corrections are formatted, according to RTCM 10410.0 and handed back to the radio modem. Afterwards, this information is pushed to the DGNSS receiver navigation processor and used to compute more accurate DGNSS positioning solutions. As the signal corrections lose their validity by the change of the satellites' constellation and, therefore, with proceeding in time, the described cycle is repeated every 5 seconds to have guaranteed up to date signal corrections for the current location.

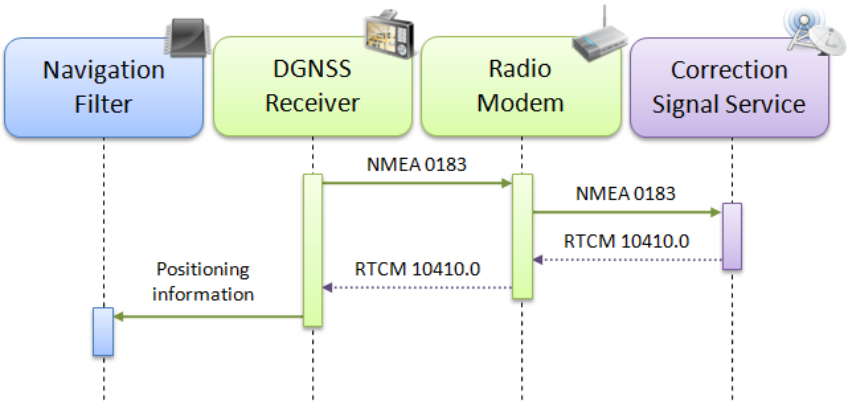

Figure 4.1.: Position and correction data flow of the DGNSS positioning system.

In a last step, the improved positioning solutions, which have been calculated, are forwarded by the receiver to the MD4-1000's NC via a second serial interface with an update rate of 4 Hz, as demanded in Section 3.1.5. There, they are decoded and parsed to the data fields of the `navsol_msg` and subsequently used for the computation of the MD4-1000's navigation solution.

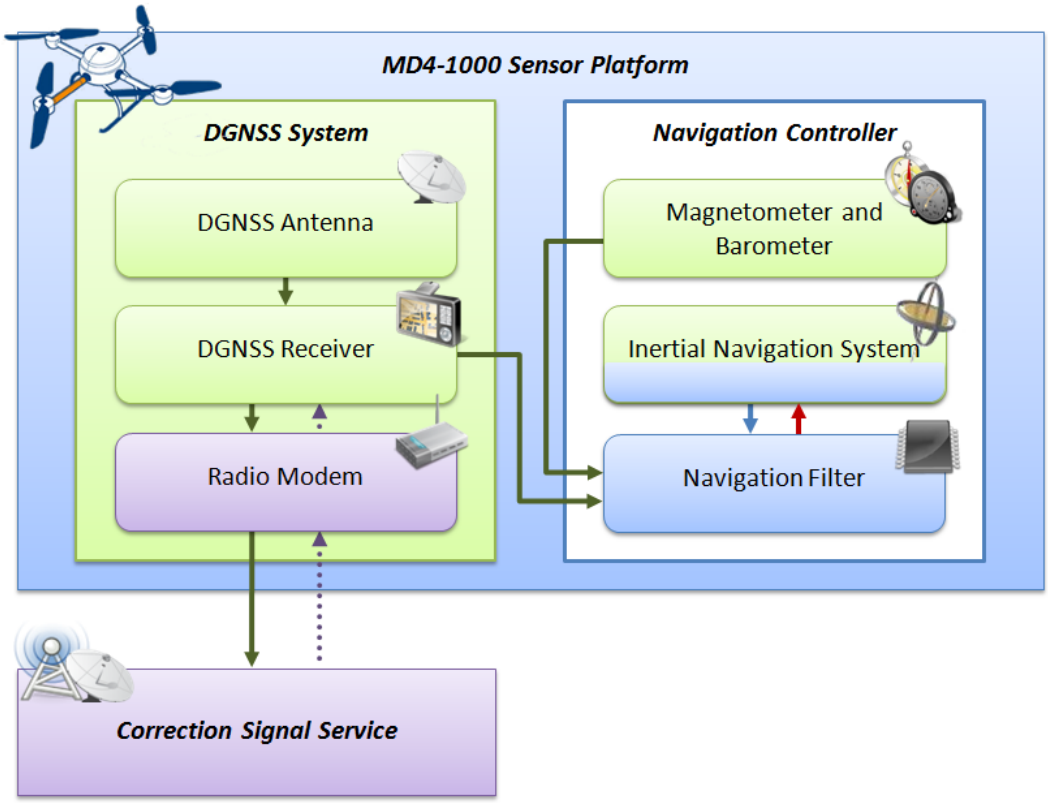

Figure 4.2.: Overview of the improved DGNSS positioning system, embedded in the MD4-1000 Sensor Platform.

After having outlined the data flow, the complete architectural draft is summarized in the following. As Figure 4.2 shows, the improved system is embedded into the MD4-1000 Sensor Platform, which is marked as the all-embracing blue box. The figure follows the explanations of how to acquire spatial position and attitude by computing a navigation solution from integrated navigation sensors, which are given in Section 2.1.4. The green box on the left hand side represents the improved modular DGNSS positioning system consisting of the active DGNSS antenna, the DGNSS receiver and the radio modem for data communication with the Correction Signal Service, which is located on the lower side of the figure and not part of the MUAV. The NC including the magnetometer, barometer, INS and the microcontroller, on which the navigation filters are run is shown in the white box on the right hand side. The data communication is visualized through the linking arrows and follows

both, the just mentioned data flow of the improved DGNSS positioning system (see Figure 4.1) and the navigation sensor's data flow from Figure 2.4.

In summary, this and the preceding sections established the basic components of the improved modular DGNSS positioning system, the connection to a Correction Signal Service and the integration into the existing MD4-1000 Navigation System. The underlying hardware and its requirements, the data flow of the components, as well as the modification of the existing NC firmware have been pointed out and the necessary steps for an implementation, which is described in Section 5.1 have been determined.

4.2. Data Provision for Sensor Web Services

This section elucidates the approach of receiving the MD4-1000's navigation data, encoding the data according to SWE standards, and making the data accessible by SWSs. Thus, it introduces the complete architectural draft of the MD4-1000 navigation sensor data architecture, regarding the aforementioned requirements of Section 3.2. It describes acquisition and decoding of the telemetry data stream and subsequent integration in the SPF. There, the data is preprocessed according to the SWE standards and is forwarded to the Sensor Web via a suitable Sensor Bus Output-Plugin.

4.2.1. Telemetry Data Decoding

Regarding the requirement of Section 3.2.1, a first step towards offering MD4-1000 navigation sensor data for SWSs is real-time data acquisition. Section 2.1.6 already showed that the MD4-1000 is equipped with a telemetry data signal that can be received by a base receiver station. The vendor-specific mdCockpit application program, which is run on a connected notebook, decodes this signal and implements a named pipe server for querying the transferred data by external programs. The pipe server is based on the Windows pipe file system and can be accessed by using normal I/O functions of the Windows operating system [MD, 2011a].

Therefore, a parsing routine will be developed, which is able to detect and access the implemented pipe server file. After having sent a specific `DOWNLINK` command, the pipe server will respond with a string containing the current binary data from the mdCockpit Downlink Decoder of the adressed MUAV. As this data is hex-encoded as ASCII-text[3] with a length of 256 bytes, the data string is consequently decoded

[3] ASCII is the American Standard Code for Information Interchange

and parsed to acquire time, position, velocity, attitude and accuracy information. Detailed information about the acquired navigation sensor values is apparent from Table 4.1 [MD, 2011a].

Category	**Sensor information**
Time	Operating time code
	Flight time code
	GPS time
Position	GNSS ECEF X
	GNSS ECEF Y
	GNSS ECEF Z
	LLA Latitude
	LLA Longitude
	LLA Altitude
	Relative Altitude
Velocity	GNSS Speed X
	GNSS Speed Y
	GNSS Speed Z
Attitude	Roll
	Pitch
	Yaw
Accuracy	Position accuracy
	Velocity accuracy
RC commands	Image trigger

Table 4.1.: MD4-1000 navigation sensor data values, retrieved from the sensor data stream.

4.2.2. Sensor Platform Integration into Sensor Platform Framework

The developed parsing routine will be implemented in a MD4-1000 specific SPF Input-Plugin to guarantee integration of the gathered sensor data. Section 2.3.3 described the possibilities of the SPF, which is capable of acquiring different sensor data streams, interpolating these to combined sensor observations, and forwarding the combined data via suitable Output-Plugins in a SWE conformable way. This thesis only concentrates on the integration of one navigation sensor data stream. However, the developed Input-Plugin is extendable by decoding and parsing additional sensor data streams, i.e. the MD4-1000's navigation sensor information and sensor observations from a mounted camera system.

The Input-Plugin is based on an XML document, which specifies the behavior of data output according to the availabilty of predefined observations. In addition, the document determines the output of phenomena. Listing 4.2 shows such an XML document. It establishes that upon availabilty (<spf:AvailabilityBehaviour>) of position information (<spf:outputProperties>), distinct navigation sensor data (<spf:mandatory Properties>) will be output along with the position. In this case, the navigation sensor data's output (<spf:output>) is limited to Position, GPS Time, GNSS Position Accuracy, Attitude Roll, Attitude Pitch, Attitude Yaw and Relative Altitude.

Listing 4.2: XML document for the definition of the MD4-1000 Input-Plugin's output behaviour.

```
<?xml version="1.0" encoding="UTF-8"?>
<spf:plugin xmlns:spf="http://ifgi.uni-muenster.de/~m_riek02/spf/0.1"
 name="urn:ifgi:id:microdrones">
 <spf:output>
  <spf:AvailabilityBehaviour>
   <spf:outputProperties>
    <spf:property>Position</spf:property>
   </spf:outputProperties>
  </spf:AvailabilityBehaviour>
   <spf:mandatoryProperties>
     <spf:property>GPS Time</spf:property>
     <spf:property>GNSS Position Accuracy</spf:property>
    <spf:property>Attitude Roll</spf:property>
    <spf:property>Attitude Pitch</spf:property>
    <spf:property>Attitude Yaw</spf:property>
    <spf:property>Relative Altitude</spf:property>
   </spf:mandatoryProperties>
 </spf:output>
 <SensorML />
</spf:plugin>
```

Another important section of the XML document is the SensorML description (<SensorML>), which defines the metadata of sensors and phenomena that shall be processed by the SPF. Because of the document's size, Listing 4.3 shows a snippet of the description, which is part of Listing 4.2. It establishes a list of inputs (<InputList>) containing discrete input elements (<input>), which reference the aforementioned properties (<spf:property>) by a name attribute. Moreover, each input element is specified by a data type, a definition identifier Uniform Resource Name (URN) and an according unit of measurement (<swe:uom>). Using this SensorML description, the SPF processes every input element according to the specifications, which have been chosen in Listing 4.2.

Listing 4.3: SensorML definition for specifying the MD4-1000 Input-Plugin's phenomena.

```
    <inputs>
     <InputList>
```

```
    <input name="Position">
     <swe:Position referenceFrame="urn:ogc:def:crs:EPSG::4326">
      <swe:location>
       <swe:Vector>
        <swe:coordinate name="LLA Position Latitude">
         <swe:Quantity>
          <swe:uom code="deg" />
         </swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="LLA Position Longitude">
         <swe:Quantity>
          <swe:uom code="deg" />
         </swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="LLA Position Altitude">
         <swe:Quantity>
          <swe:uom code="m" />
         </swe:Quantity>
        </swe:coordinate>
       </swe:Vector>
      </swe:location>
     </swe:Position>
    </input>
    <input name="GPS Time">
     <swe:Quantity definition="urn:ogc:def:dataType:OGC:1.1:time">
      <swe:uom code="ms" />
     </swe:Quantity>
    </input>
    <input name="GNSS Position Accuracy">
     <swe:Quantity definition="urn:ogc:def:dataType:OGC:1.1:measure">
      <swe:uom code="m" />
     </swe:Quantity>
    </input>
    <input name="Attitude Roll">
     <swe:Quantity definition="urn:ogc:def:dataType:OGC:1.1:measure">
      <swe:uom code="deg" />
     </swe:Quantity>
    </input>
    <input name="Attitude Pitch">
     <swe:Quantity definition="urn:ogc:def:dataType:OGC:1.1:measure">
      <swe:uom code="deg" />
     </swe:Quantity>
    </input>
    <input name="Attitude Yaw">
     <swe:Quantity definition="urn:ogc:def:dataType:OGC:1.1:measure">
      <swe:uom code="deg" />
     </swe:Quantity>
    </input>
    <input name="Relative Altitude">
     <swe:Quantity definition="urn:ogc:def:dataType:OGC:1.1:measure">
      <swe:uom code="m" />
     </swe:Quantity>
    </input>
   </InputList>
  </inputs>
```

The parsing routine in combination with the XML and SensorML definitions form the basis of the implemented MD4-1000 specific Input-Plugin, which will be introduced in Section 5.2. As SensorML ensures an encoding for the output of navigation sensor data following the standards of SWE, as well as the document's reusability by several SWSs, the requirements of Section 3.2.2 have been taken into consideration.

4.2.3. Sensor Platform Framework and Sensor Bus Data Connection

Having implemented an appropriate Input-Plugin, the SPF automatically processes the sensor data in the specified way. Therefore, the last step in designing a MD4-1000 navigation sensor data architecture is to ensure valuable sensor data output for subsequent use in the Sensor Web, as demanded in Section 3.2.3.

Besides the input plugin interface, the SPF provides an output plugin interface (see Section 2.3.3). This output plugin interface receives the processed sensor data and its according SensorML definitions from the SPF core. As a consequence, a MD4-1000 specific Output-Plugin could be implemented, which is able to publish sensor data to a certain SWS. As the objective of this thesis is to offer the sensor platform's navigation data to a variety of SWSs, the sensor data architecture follows the approach of the Sensor Bus as intermediary layer between the MD4-1000 Sensor Platform and the Sensor Web (see Section 2.3.2). Thus, the design of the navigation sensor data architecture considers the use of a Sensor Bus Output-Plugin for encoding all information to Sensor Bus messages. Subsequently, the Output-Plugin registers the MD4-1000 Sensor Platform and publishes its sensor information to the Sensor Bus.

However, the development of this plugin is not part of this thesis. Therefore, the sensor data architecture adapts an already implemented Sensor Bus Output-Plugin[4].

4.2.4. MD4-1000 Navigation Sensor Data Architecture

At this point, the sensor data architecture can be considered as complete. Figure 4.3 establishes an overview of the MD4-1000's navigation sensor data architecture's workflow. This figure is quite revealing in several ways. First of all, the MD4-1000 Sensor Platform, situated on the upper left side, transmits its navigation sensor data to a base receiver station, as depicted in the white box. Second, the mdCockpit

[4]Sensor Bus Output-Plugin, published at 52 °North Initiative for Geospatial Open Source Software GmbH (http://52north.org/communities/sensorweb/incubation/ifgicopter/spf/index.html)

downlink decoder application program receives the data stream and implements a named pipe server. Third, the MD4-1000 Input-Plugin as part of the SPF, indicated by three blue boxes, addresses the named pipe server to retrieve the data stream. Subsequently, the plugin decodes and parses the data stream's content and forwards it with detailed XML and SensorML descriptions to the SPF's core. Fourth, the core processes the sensor data according to these definitions and provides the data for a Sensor Bus Output-Plugin. Fifth, The Sensor Bus Output-Plugin formats the data to Sensor Bus messages and registers the MD4-1000 at the Sensor Bus. Sixth, any SWS, depicted in the green box on the lower left side, can register to the Sensor Bus by a suitable Service Adapter and retrieve the navigation sensor data for subsequent use.

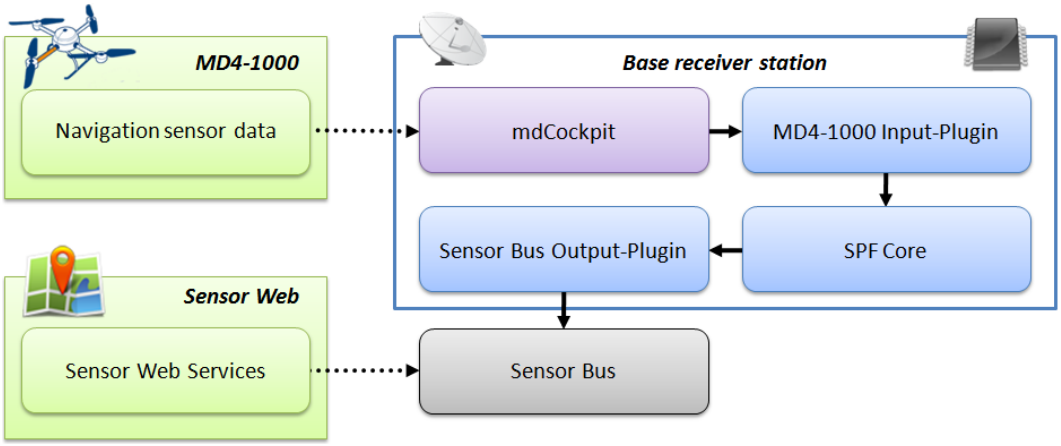

Figure 4.3.: Overview of the MD4-1000 navigation sensor data architecture.

5. Implementation

After having described the concepts and strategies in Chapter 4, this chapter introduces the implementation of the prototypical DGNSS positioning system and its integration into the MD4-1000 Sensor Platform. Moreover, the developed Sensor Platform Framework Input-Plugin for the reception and decryption of the sensor data is explained.

5.1. Prototypical DGNSS Positioning System

The prototypical DGNSS positioning system is implemented according to the concept and architectural draft, which has been outlined previously (see Section 4.1). Thereby, the choice of the hardware components and of the correction signal acquisition method, as well as the firmware related modifications are made with regard to the requirements, which are listed in Section 3.1. These individual parts of the complete prototype and their discrete configurations are described in the following.

5.1.1. Hardware Components

The prototype consists of three major hardware components that form its core and offer the improved DGNSS functionality. The first component, a more sophisticated active DGNSS antenna is used instead of the standard antenna listed in Section 2.1.2. The new antenna of the type Antcom G5Ant-1AT1 (see Figure 5.1) serves the purpose of receiving L1 satellite signals from both GPS and GLONASS, which has been established in the architectural concept of Section 4.1.1.

As Table 5.1 shows, the Antcom G5Ant-1AT1 is conform to the requirements of low weight with a mass of 105 g and low power consumption with a mean power of 0.120 W at an input voltage of 3.3 VDC (see Section 3.1.4). Besides, it is well suited for MUAVs because of its minor dimensions of 53 x 53 x 21 mm, compared to other standard active DGNSS antennas, which are of greater size [Antcom, 2009].

Figure 5.1.: Antcom G5Ant-1AT1 DGNSS antenna and connection cable.

Technical Specifications	
GNSS support	GPS (L1/L2/L5) + GLONASS (L1/L2)
Dimensions	53 x 53 x 21 mm
Weight	105 g
Input voltage	+2.5 to +24.0 VDC
Power consumption	0.120 W @ 3.3 VDC

Table 5.1.: Selected technical specifications of the Antcom G5Ant-1AT1 DGNSS antenna (following [Antcom, 2009]).

Moreover, the Antcom G5Ant-1AT1 is already capable of using L2 satellite signals that are needed for a future RTK-GNSS solution. Thus, the antenna already fulfills the goal of the demanded option for extensibility exacted in Section 3.1.2.

The second component of the prototype is an improved DGNSS receiver, which features combined L1 GPS and GLONASS code and carrier phase tracking for increased positioning accuracy and availability. The used NovAtel OEMStar receiver offers additional support of both, SBASs and GBASs for DGNSS positioning. It is capable of tracking eight GPS, four GLONASS and two SBAS satellites at the same time and works with an automatic routine to switch from SBAS to GBAS support as soon as ground based signal corrections are received [NovAtel, 2010b]. It therefore complies with the demands of improving absolute positioning, mentioned in Section 3.1.1. Figure 5.2 shows the receiver board with its external antenna input, two serial COM ports for the connection of a suitable radio modem and the MD4-1000's NC, which is needed to meet the requirements of Section 3.1.5.

The most important technical specifications of the NovAtel OEMStar receiver are listed in Table 5.2. It is able to process satellite signals with a horizontal position

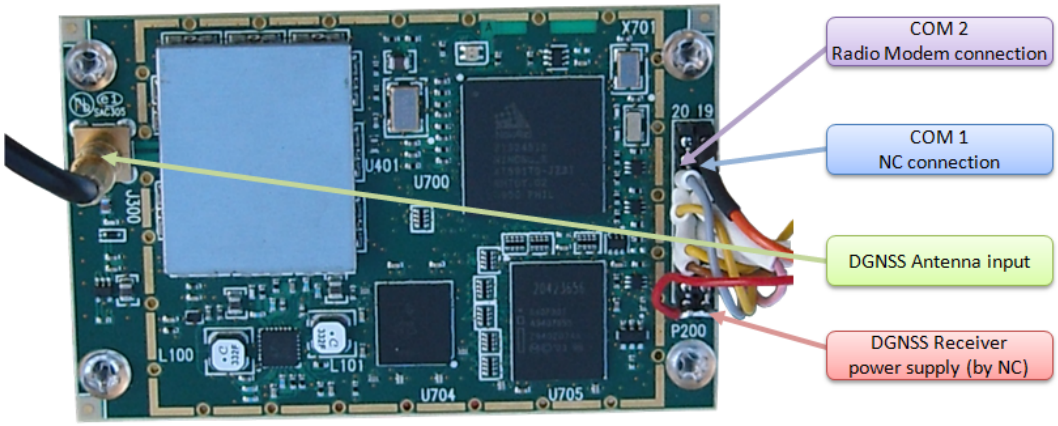

Figure 5.2.: NovAtel OEMStar DGNSS receiver and realization of communication.

Technical Specifications	
Channel configuration	8 GPS L1 + 4 GLO L1 + 2 SBAS
Horizontal position accuracy SBAS	0.7 m (RMS)
Horizontal position accuracy DGPS	0.5 m (RMS)
Dimensions	46 x 71 x 13 mm
Weight	18 g
Input voltage	+3.3 to +5.25 VDC
Power consumption	0.360 W
Communication ports	2 LV-TTL serial ports capable of 300 to 230400 bps

Table 5.2.: Selected technical specifications of the Novatel OEMStar DGNSS receiver (following [NovAtel, 2010b]).

accuracy of 0.7 m RMS[5] in SBAS and 0.5 m RMS in GBAS mode. The achieved position accuracy information in GBAS mode is evaluated later on in Section 6.1.3. Besides, the receiver features real-time position and, furthermore, speed accuracy estimations to meet the safety in flight information demanded in Section 3.1.3. These estimations are acquired through appropriate information logs, which are retrieved by the receiver and introduced in Section 5.1.4. Regarding its low weight of 18 g and its low power consumption of 0.360 W, the requirements of Section 3.1.4 are satisfied. The power supply is established by using the MD4-1000's internal battery and the COM1 NC connection.

[5]RMS = Root Mean Square: The square root of the average of the squared horizontal position errors, containing 65 % of the fixes. [NovAtel, 2003]

The third component forms an appropriate radio modem for ground based correction signal acquisition. As the radio modem needs to register to a Correction Signal Service, to forward a `$GPGGA` message and subsequently to retrieve RTCM corrections as described in Section 4.1.2, the prototypical implementation of the DGNSS system makes use of the Allsat come2ascos radio modem, which has been especially developed for the communication between DGNSS receivers and Correction Signal Services. Figure 5.3 shows that the modem is equipped with a GSM/GPRS antenna input and a SIM card to subscribe to a local telephone service provider in order to be able to use a GPRS connection for the correction signal transport. The marked serial port is used for the data flow between radio modem and DGNSS receiver.

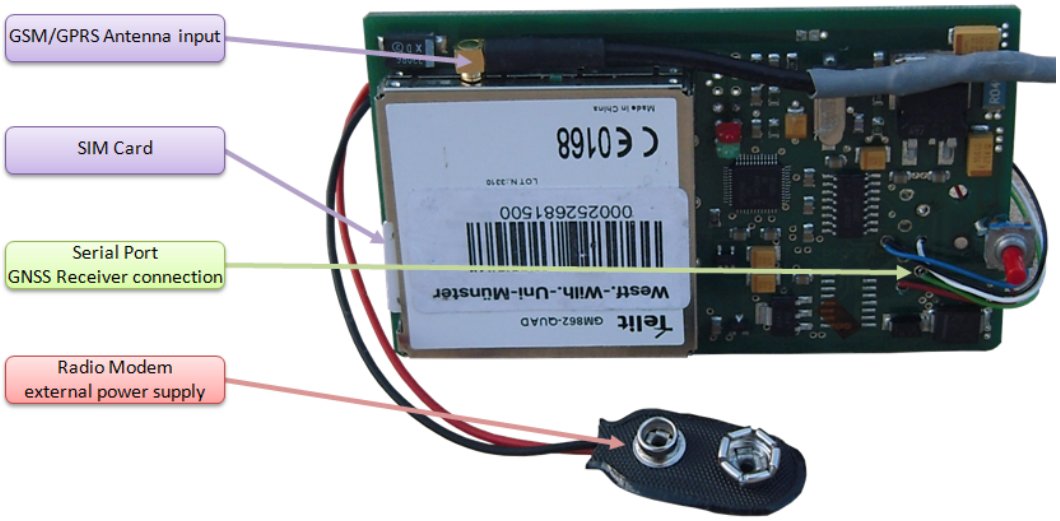

Figure 5.3.: Allsat come2ascos radio modem and realization of communication.

Technical Specifications	
Dimensions	100 x 60 x 30 mm, reduced to 90 x 55 x 10 mm
Weight	220 g, reduced to 56 g
Input voltage	+8.0 to +13.5 VDC
Power consumption	3.6 W @ 12 VDC GPRS/Ntrip
Communication format	RTCM, CMR and other
Communication port	1 RS232 capable of 2400 to 57600 bauds
GSM/GPRS Modem	FINMEK TELIT S.p.A.
GSM/GPRS Band	900/1800/1900 MHz

Table 5.3.: Selected technical specifications of the Allsat come2ascos radio modem (following [Allsat, 2008]).

In contrast to the OEMStar, the come2ascos needs an external power supply because of a higher input voltage of 8.0 to 13.5 VDC (see Table 5.3) and a relatively high power consumption of 3.6 W at an input voltage of 12 VDC. For this reason, it is connected to the MD4-1000's telemetry link's power line, which provides the demanded input voltage. Besides that, a voltage transformer is implemented between the come2ascos and the OEMStar to enable data communication via the serial interface by regulating the different voltage levels. As the come2ascos is delivered in an enclosure, this enclosure is removed from the board to reduce the dimensions to 90 x 55 x 10 mm and its weight from 220 g to 75 g, in order to meet the requirements in Section 3.1.4.

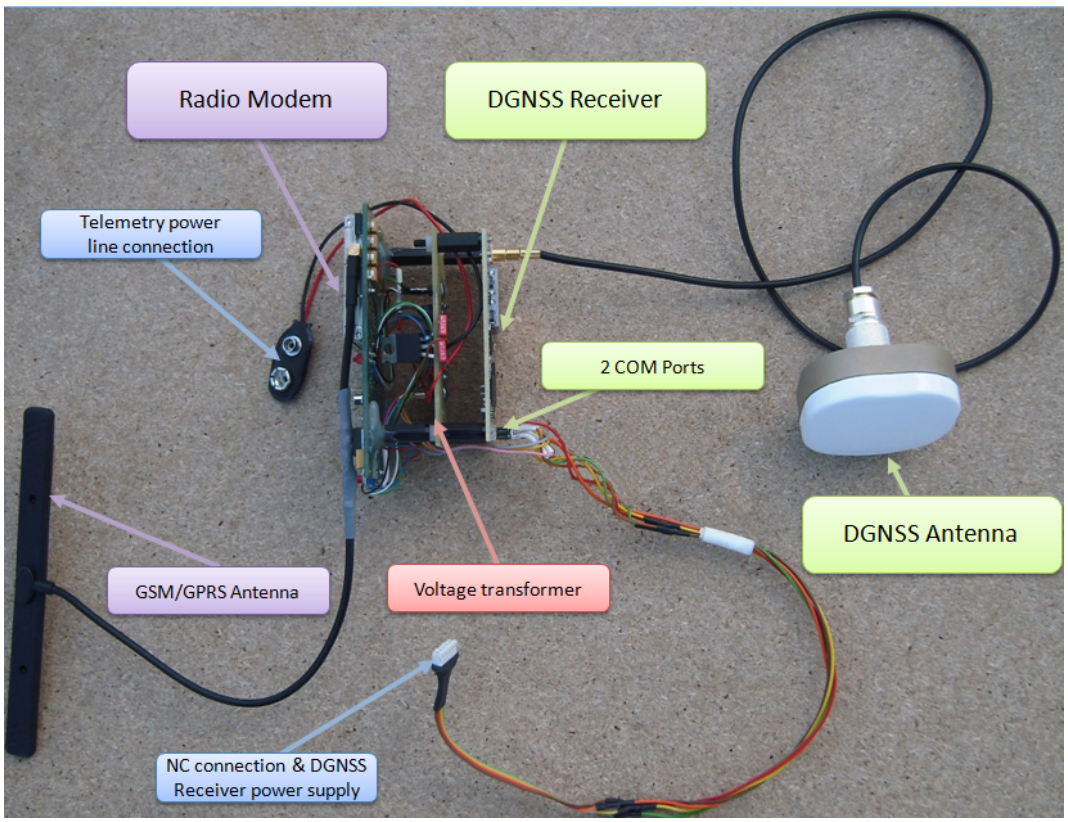

Figure 5.4.: Overview of the prototypical DGNSS positioning system's components and connection.

After having introduced the three core components, Figure 5.4 gives an overview of the complete prototypical DGNSS hardware setup with all its components. It consists of the DGNSS antenna and the DGNSS receiver, which are interconnected by the black cable located in the upper right corner. This cable offers satellite signal transmission and the power supply of the antenna, which is obtained from the receiver. The receiver offers two COM ports. COM1 is connected to a bundle of wires with a special plug to connect to the MD4-1000's NC for data transfer and power supply of the receiver. COM2 is connected to the radio modem for commu-

nication of position and signal correction sets by interposing a voltage transformer. The radio modem needs an appropriate GSM/GPRS antenna for network connection and an extra power supply to the MD4-1000's telemetry link's power line. All these components form the entire setup with an absolute weight of approximately 300 g.

5.1.2. Hardware Configuration

This Section addresses the firmware related configuration of two of the core components, the DGNSS receiver and the radio modem. Both components have been configured by using a serial interface connection and a conventional terminal program.

Listing 5.1: NovAtel OEMStar receiver configuration parameters for positioning and communication.

```
1 SELECTCHANCONFIG 5
2 SBASCONTROL ENABLE AUTO 0 NONE
3 COM COM1 57600
4 COM COM2 19200
5 INTERFACEMODE COM2 RTCMV3 NOVATEL OFF
6 LOG COM1 PSRDOPA ONTIME 0.25
7 LOG COM1 BESTXYZA ONTIME 0.25
8 LOG COM2 GPGGA ONTIME 5
```

Listing 5.1 shows the configuration parameters, which have been chosen for adjusting the OEMStar DGNSS receiver. Besides the `SELECTCHANCONFIG 5` command, which sets the used channels for satellite tracking to eight GPS, four GLONASS and two SBAS satellites and the automatically enabling of SBAS support by `SBASCONTROL ENABLE AUTO 0 NONE`, the most important commands are the control of the interface mode `INTERFACEMODE COM2 RTCMV3 NOVATEL OFF` between receiver and radio modem and the log messages, which are pushed to the serial interfaces and explained in the next two sections. Further information about the configuration commands can be found in [NovAtel, 2010a].

Listing 5.2: Allsat come2ascos radio modem configuration parameters for data handling and correction signal acquisition.

```
1 $PSLL,CONFIG,NTRIP,19K2,internet.t-d1.de,t-d1,gprs,62.40.9.7:2101,user:password,
      MOUNT:VRS_3_NW,SKIP-TABLE,NEED-GGA
```

The configuration of the come2ascos radio modem is mentioned in Listing 5.2. These parameters determine, amongst others, the GPRS gateway (`internet.t-d1.de`) of the telephone service provider, the URL and port of the Correction Signal Service (`62.40.9.7:2101`), as well as the mount point (`MOUNT:VRS_3_NW`) for the retrieving

of signal correction sets. A detailed description of all parameters is given in [Allsat, 2008].

5.1.3. Correction Signal Acquisition

The correction signal acquisition follows the explanations of Section 4.1.2. The receiver is configured by the `LOG COM2 GPGGA ONTIME 5` command to send a `$GPGGA` message (see Listing 4.1) at an output rate of five seconds. This output is done on COM2, the communication port to the radio modem. As soon as the radio modem receives the `$GPGGA` message, it starts to register to the high precision SAPOS HEPS Correction Signal Service to retrieve signal correction sets, encoded in the RTCM standard. Listing 5.2 shows that the radio modem is configured to exact GPS and GLONASS corrections and additional coordinate transformation parameters from a non-physical reference station (VRS), which has been computed by the information of the Correction Signal Service's reference stations network and has been located in the close surrounding of the `$GPGGA` position [SAPOS, 2012].

The SAPOS HEPS guarantees horizontal positioning accuracy of 1 - 2 cm (RMS) through using adequate GNSS hardware equipment and suitable satellites' constellations [SAPOS, 2010]. As the prototypical setup is based on L1 signal DGNSS technique, the assured accuracy is not achieved but the signal corrections are more than sufficient for the setup's positioning and, therefore, already meet the requirements of a future upgrade to RTK-GNSS as demanded in Section 3.1.2.

5.1.4. MD4-1000 Firmware Modification

Completing the integration of the positioning sytem, the information, processed by the receiver, is forwarded to the MD4-1000's Navigation Controller with an output rate of 4 Hz. The NC's firmware is modified by a position information parsing routine, which is written in the C programming language. The software is designed to accept two consecutively output log messages, to fill the data fields of the MD4-1000's internal data structure (see Listing 3.1).

It is obvious from Listing 5.3 that the output logs (`PSRDOPA` and `BESTXYZA`) are subdivided into a `header` and `body` string, which are separated by a semicolon. The `header` contains information about the GNSS time. The `body` contains log specific information about e.g. DOP values, the status of the current positioning solution, the coordinates in X,Y,Z, the estimated position accuracies, the number of tracked satellites and many more, which is a requirement of the data structure.

Listing 5.3: NovAtel PSRDOPA and BESTXYZA message logs for communication with the MD4-1000's NC.

```
GNSS Logs (Novatel)

PSRDOPA: (DOPs of current SVs)
#PSRDOPA, port, ?, ?, gps time status, gps week, gps seconds, ?, ?, ?;gdop, pdop, hdop, htdop
    , tdop, cutoff, #prns, prn, next prn...

BESTXYZA: (Cartesian coord pos)
#BESTXYZA, port, ?, ?, gps time status, gps week, gps seconds, ?, ?, ?;p-solstat, p-type, p-x
    , p-y, p-z, p-xstd, p-ystd, p-zstd, v-solstat, v-type, v-x, v-y, v-z, v-xstd, v-ystd, v-
    zstd, stnid, v-latency, diff_age, sol_age, #SV, #solnSV, #ggl1, rsrvd, rsrvd, ext sol
    stat, rsrvd, sig mask
```

Listing 5.4 shows the most important steps of the parsing routine. In a first step, the incomming log message is handed over to the parsing method `gps_parse_novatel()` as ASCII string (`input_buffer`). The function analyzes whether the `input_buffer` contains characters and if so, tries to split the ASCII string into its `header` and `body` using `gps_split_message()`. In case of success, the two corresponding character arrays are returned. The routine then distinguishes between `PSRDOPA` and `BESTXYZA`. If the `input_buffer` was filled with the `PSRDOPA` log, the `header` array is used to extract GNSS solution status and time information and in a consquent step, the `body` array is used to acquire DOP information. If the `input_buffer` was filled with the `BESTXYZA` log, it is only the information contained in the `body` array that is used for getting position and accuracy values. No matter which character array was detected, the acquired information is subsequently written to the data fields of the `navsol_msg` and internal flags are set.

Listing 5.4: MD4-1000 NC parsing routine for NovAtel OEMStar PSRDOPA and BESTXYZA message logs.

```
void gps_parse_novatel(char* input_buffer)
{
 // Check if input_buffer contains characters
 if(input_buffer != NULL)
 {
  // Split input_buffer into header and body
  char header[1024];
  char body[1024];
  gps_split_message(input_buffer, header, body);

  // Message Type #PSRDOPA ////////////////////////////////////////////////////////
  if(memcmp(header,"#PSRDOPA",8) == 0)
  {
   int i = 0;
   char *token;
   char *t_status;   // GPS Status
   // ...

```

```
19    // Split header into tokens and parse to #PSRDOPA Variables (Acquire GNSS status and time
          information)
20    token = header;
21    while(token != NULL)
22    {
23     switch(i)
24     {
25      case 4:
26      {
27       // GPS Status -> t_status
28       t_status = token;
29       break;
30      }
31      // ...
32     }
33     i++;
34     token = gps_skip_to_next_token(token);
35    }
36
37    // Split body into tokens and parse to #PSRDOPA Variables (Acquire DOP information)
38    // ...
39
40    // Fill navsol_msg
41    msg.itow = (int) ((t_seconds)*1e3);
42    // ...
43
44    // Set flags
45    if(memcmp(t_status,"FINESTEERING",12) == 0)
46    {
47     msg.flags = (msg.flags | (1 << 2));
48     // ...
49    }
50   }
51
52   // Message Type #BESTXYZA ////////////////////////////////////////////////////////
53   else if(memcmp(header,"#BESTXYZA",9) == 0)
54   {
55    // ...
56   }
57  }
58 }
```

5.2. Sensor Platform Framework Input-Plugin

This section describes the developed MD4-1000 specific Input-Plugin, which implements the SPF input plugin interface, using the Java programming language. Its functionallity follows the demands of Section 3.2 and the architectural draft, which has been established in Section 4.2.

Figure 5.5 shows a simple UML[6] class diagram of the `IfgicopterInputPluginMD` class, depicted in the green box, and the implemented SPF interfaces, depicted in red boxes. This diagram lists all attributes and operations of the individual classes. The most important operations of the `IfgicopterInputPluginMD` class, namely `init()`, `parseDownlink()`, `getNewData()` and `getConfigFile()`, will be explained in the following, using Java code snippets.

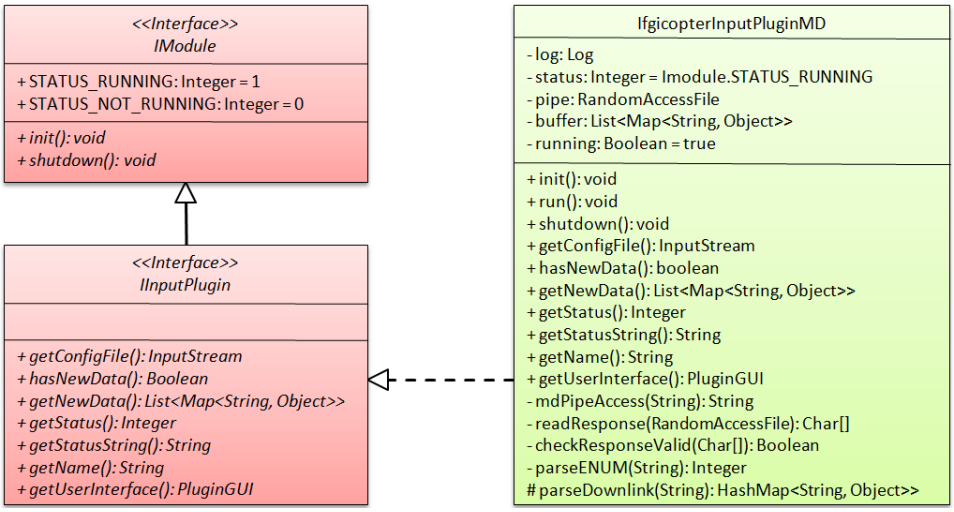

Figure 5.5.: UML class diagramm of the `IfgicopterInputPluginMD` class and its implemented interfaces.

Listing 5.5 shows the `init()` method of the `IfgicopterInputPluginMD` class. This method establishes a connection to the named pipe server, which is implemented by a running mdCockpit Downlink Decoder Dialogue (see 4.2.1). In a first step, an analysis for a running Downlink Decoder Dialogue is carried out. If a valid Downlink Decoder Dialogue is detected, its unique identification number (`dld`) is determined and a thread for receiving the MD4-1000's sensor data is implemented in a subsequent step. In this thread's `run()` method, a while-loop queries the named pipe server for the current sensor data at a rate of 1 Hz, by using the `mdPipeAccess()` method. In a next step, the queried sensor data is parsed by the `parseDownlink()` method, which is the main function for extracting the sensor data information. This method is called with a sensor data string parameter, which has been retrieved as a return statement from the `mdPipeAccess()` method and is encoded according to the explanations in Section 4.2.1. Detailed information about the string's format is given in [MD, 2011a].

[6]The Unified Modeling Language (UML) is a standardized notation in the field of software engineering.

Listing 5.5: `IfgicopterInputPluginMD`'s `init()` method for controlling access and query of a named pipe server, as well as parsing of MD4-1000 navigation sensor data.

```
@Override
public void init() throws Exception {
 // pipe server connect to pipe
 pipe = new RandomAccessFile("\\\\.\\pipe\\mdCockpitPipeService", "rw");
 log.info("Pipe has been opened....");

 // check ENUM for connected Downlink Decoder Number
 final int dld = parseENUM(mdPipeAccess("ENUM\n"));
 if(dld != -1){
  log.info("Downlink Decoder detected.");
  // threading to get DOWNLINK Data (rate: 1Hz)
  new Thread(new Runnable() {
   @Override
   public void run() {
    while (running) {
     // get downlink message from pipe server
     HashMap<String, Object> data;
     try {
      data = parseDownlink(mdPipeAccess("DOWNLINK " + "DLD"+dld+"\n"));
      synchronized (buffer) {
       buffer.add(data);
      }
     } catch (IOException e) {
      log.error(e.getMessage(), e);
     }
     try {
      Thread.sleep(1000);
     } catch (InterruptedException e) {
      log.error(e.getMessage(), e);
     }
    }
   }
  }).start();
 }
 else {
  log.warn("Downlink Decoder not detected.");
 }
}
```

Focusing on `parseDownlink()`, it becomes apparent that the sensor data string is primarily split into its 256 ASCII-represented hex-encoded bytes (see Listing 5.6). In doing so, each byte's superfluous digits, namely `0x`, are cropped for subsequent processing. Following this, the bytes are regrouped from little endian to big endian for a conversion from hex to decimal representation. Then, the conversion to decimal observation values is carried out with respect to the position of the bytes in the inital sensor data string. The information about the relation between observation value and byte position can be deduced from [MD, 2011a]. Finally, the converted values are added to a `HashMap<String, Object>`, where `String` defines the sensor

observation's name, and `Object` defines the data type and value. For the subsequent step of processing and standardization, it is important that the observations' naming is in analogy to the specified names in the SensorML (<input>) elements (see Listing 4.3).

Listing 5.6: `IfgicopterInputPluginMD`'s `parseDownlink()` method as for parsing a received sensor data string.

```
1  /*
2   *  Method to parse DOWNLINK Data content
3   */
4  protected static HashMap<String, Object> parseDownlink(String responseBody) throws
       IOException {
5   // create a HashMap
6   HashMap<String, Object> hashMap = new HashMap<String, Object>();
7
8   if(responseBody != null){
9    // split data string into discrete components
10   String[] hexDataLines;
11   String[] hexData = null;
12   ArrayList<String> lHexData = new ArrayList<String>();
13   hexDataLines = responseBody.split("\n");
14   for (int i = 0; i < hexDataLines.length; i++){
15    hexData = hexDataLines[i].split(",");
16    for (int j = 0; j < hexData.length; j++){
17     // fill lHexData with all 256 values and crop them to the last 2 digits (e.g. 0x9C -> 9C)
18     lHexData.add(hexData[j].substring(2));
19    }
20   }
21
22   /*
23    * regroup all values of lHexData from little Endian to big Endian
24    * convert all values to decimal values
25    * add all values to HashMap
26    */
27   // conversion variable i
28   Long i;
29   // GPS Time Of Week [ms] (DWORD)
30   String gpsTOW = lHexData.get(31) + lHexData.get(30) + lHexData.get(29) + lHexData.get(28);
31   // GPS Week (DWORD)
32   String gpsW = lHexData.get(35) + lHexData.get(34) + lHexData.get(33) + lHexData.get(32);
33        // Set GPS Time (in ms)
34   int weekMS = 604800000;
35   Long gpsMS = new Long(Long.parseLong(gpsTOW, 16)) + (new Long(Long.parseLong(gpsW, 16)) *
         weekMS);
36   hashMap.put("GPS Time", gpsMS);
37   // LLA Position Latitude [Decimal Degrees] (double)
38   String gpsLlaLat = lHexData.get(103) + lHexData.get(102) + lHexData.get(101) + lHexData.get
         (100) + lHexData.get(99) + lHexData.get(98) + lHexData.get(97) + lHexData.get(96);
39        i = Long.parseLong(gpsLlaLat, 16);
40        hashMap.put("LLA Position Latitude", Math.round(new Double(Double.longBitsToDouble(i.
              longValue()))*1E7)/1E7);
41   // LLA Position Longitude [Decimal Degrees] (double)
42   String gpsLlaLon = lHexData.get(111) + lHexData.get(110) + lHexData.get(109) + lHexData.get
         (108) + lHexData.get(107) + lHexData.get(106) + lHexData.get(105) + lHexData.get(104);
43        i = Long.parseLong(gpsLlaLon, 16);
```

```
44          hashMap.put("LLA Position Longitude", Math.round(new Double(Double.longBitsToDouble(i
                .longValue()))*1E7)/1E7);
45    // LLA Position Altitude [m] (double)
46    String gpsLlaAlt = lHexData.get(119) + lHexData.get(118) + lHexData.get(117) + lHexData.get
          (116) + lHexData.get(115) + lHexData.get(114) + lHexData.get(113) + lHexData.get(112);
47          i = Long.parseLong(gpsLlaAlt, 16);
48          hashMap.put("LLA Position Altitude", Math.round(new Double(Double.longBitsToDouble(i.
                longValue()))*1E2)/1E2);
49          // GNSS Position Accuracy [m] (FLOAT)
50    String gpsPA = lHexData.get(95) + lHexData.get(94) + lHexData.get(93) + lHexData.get(92);
51          i = Long.parseLong(gpsPA, 16);
52          hashMap.put("GNSS Position Accuracy", Math.round(new Float(Float.intBitsToFloat(i.
                intValue()))*1E2)/1E2);
53    // Attitude Roll [Degrees] (FLOAT) ...
54    // Attitude Pitch [Degrees] (FLOAT) ...
55    // Attitude Yaw [Degrees] (FLOAT) ...
56    // Relative Altitude [m] (FLOAT) ...
57   }
58   return hashMap;
59  }
```

Taking up on the while loop of Listing 5.5, the returned `HashMap<String, Object>` is finally added to a synchronized `buffer`, which holds a list of `HashMap<String, Object>` containing the acquired sensor data information.

Listing 5.7 illustrates two essential methods for further processing and standardization, carried out by the SPF's core. The SPF uses `getNewData()` to retrieve a copy of the current sensor data list and, in addition, calls `getConfigFile()` for gathering the required XML and SensorML definitions, which were introduced in Section 4.2.2.

Listing 5.7: `IfgicopterInputPluginMD`'s `getNewData()` and `getConfigFile()` methods for handing over acquired sensor data and sensor information to the SPF's core.

```
1   // method to hand over a list of sensor data (List<Map<String, Object>>)
2   @Override
3   public List<Map<String, Object>> getNewData() {
4    synchronized (buffer) {
5     List<Map<String, Object>> copy = new ArrayList<Map<String, Object>>(buffer);
6     buffer.clear();
7     return copy;
8    }
9   }
10
11  // method to return a FileInputStream giving access to definitions of the Input-Plugin's
        behaviour and sensor, as well as sensor data information
12  @Override
13  public InputStream getConfigFile() {
14   // IInputPlugin description spf:plugin
15   File f = new File("config/spf/input-microdrones.xml");
16   if (f.exists()) {
17    try {
```

```
18      return new FileInputStream(f);
19     } catch (FileNotFoundException e) {
20      log.warn(e.getMessage(), e);
21     }
22    }
23    return getClass().getResourceAsStream("/config/spf/input-microdrones.xml");
24   }
```

Summarizing this section, a MD4-1000 specific Input-Plugin was developed, which is able to retrieve the MD4-1000's navigation sensor data at a rate of 1 Hz. Moreover, the plugin decodes the received sensor data strings and converts them according to predefined SensorML specifications for subsequent processing and provision. At this point, the MD4-1000's navigation sensor data is integrated into the SPF, following the SWE definitions, and can be published via a suitable Sensor Bus Output-Plugin to any SWS. Section 6.2 will show the operability of the `IfgicopterInputPluginMD` with an exemplatory Sensor Bus Output-Plugin.

5.3. Interaction and Overview

After having outlined the implemented prototypical DGNSS positioning system and the developed MD4-1000 specific Input-Plugin, this section describes interaction and overview of the established navigation sensor data architecture and its components.

Figure 5.6 illustrates the components and data flow, starting with the acquisition of improved positioning information by the DGNSS positioning system, depicted as a green box on the upper left side. First of all, the position information is acquired, following the principle, which has been introduced in Figure 4.1. Second, the information is forwarded to the NC via receiver specific message logs, and subsequently combined with the data of the additional navigation sensors (see Figure 4.2). Third, the NC calculates a navigation solution and passes this solution and some platform-specific information to a downlink transmitter. Fourth, the transmitter broadcasts a telemetry signal stream, which is received by a base receiver station and a connected notebook, depicted as white box. Fifth, a mdCockpit Downlink Decoder application program implements a named pipe server for the retrieval of the signal stream's information (see Section 4.2.1). Sixth, the MD4-1000 specific Input-Plugin accesses the named pipe server at regular intervals of 1 s, decodes the returned sensor data string and parses the desired navigation information (see Section 5.2). Seventh, the navigation information and the according XML and SensorML descriptions (see Section 4.2.2) are handed over to the SPF's core for processing. Eigth, a suitable Sensor Bus Output-Plugin, which implements a Sensor Adapter, retrieves the processed data

from the core, registers the MD4-1000 at the Sensor Bus a and publishes its data (see Section 4.2.3). Finally, the data can be accessed by any SWS via the Sensor Bus architecture.

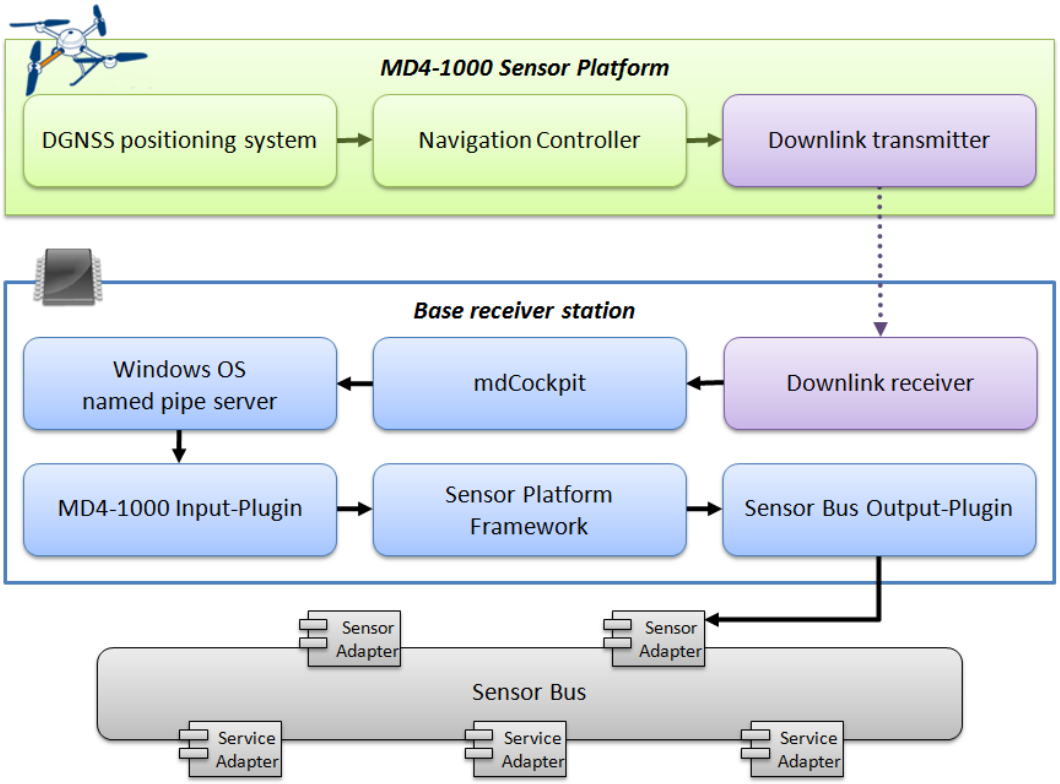

Figure 5.6.: Overview and interaction of the improved DGNSS positioning system and the developed navigation sensor data architecture

6. Proof of Concepts

The preceding chapters established the complete architecture and interaction of the prototypical DGNSS positioning system, its integration into the MD4-1000 Sensor Platform and the sensor platform's data integration into the Sensor Platform Framework for subsequent usage in Sensor Web Services. This chapter evaluates the overall concept and the interaction of all components. Thus, the requirements, which have been demanded in Chapter 3 are compared with the implemented functionality of the concept. Thereby, this evaluation emphasizes in particular the implementation of a valuable MD4-1000 positioning improvement and the near real-time sensor data integration into the Sensor Web by the developed Sensor Platform Framework Input-Plugin.

6.1. Evaluation of the Prototypical DGNSS Positioning System

It is shown in Section 5.1 that the prototypical DGNSS positioning system is designed in a modular way for a potential exchange of discrete components. Actually, it is already well prepared for the use of more sophisticated technologies like RTK-GNSS, which has been demanded in Section 3.1.2. Since the antenna is suited for signal acquisition of several GNSS on a combination of L1 and L2 and the received real-time DGNSS corrections are of high precision (see Sections 5.1.1 and 5.1.3), it is merely the receiver that needs to be replaced by a more efficient one, in order to be able to guarantee positioning with even higher accuracy than it is analyzed in the following in Section 6.1.3.

According to Section 5.1.1, all implemented components are of minor size and, therefore, suitable for the integration into the MD4-1000. Regarding the requirements of low power consumption in Section 3.1.4, the DGNSS receiver and antenna have been proven to be adequate. The radio modem consumes power of approximately 3.6 W in average and, moreover, uses the telemetry power line for its need of an external power supply. However, with regard to the limited capacity of the MD4-1000's battery an optimum cannot be achieved yet.

Two further important aspects are realized by the modification of the MD4-1000's firmware, which is explained in Section 5.1.4. Firstly, the estimation of the positioning system's accuracy, as required in Section 3.1.3, is calculated by exploiting the receiver's `BESTXYZ` log in the implemented parsing routine. Secondly, the `PSRDOP` log in addition to the before mentioned log, guarantee the integration of the receiver's positioning information into the sensor platform's navigation system as claimed in Section 3.1.5.

The following sections will evaluate the essential requirement of an improved absolute positioning solution, as established in Section 3.1.1. These sections describe absolute positioning tests of a MD4-1000, equipped with the prototypical DGNSS positioning system, which have been carried out on a predefined test field in Münster.

6.1.1. Test Field Horstmarer Landweg

As the evaluation of positioning requires a reliable source of reference, a test field was set up on the sports ground at Horstmarer Landweg in Münster. The test field consists of ten Ground Control Points (GCP), which were defined by two measurements with an advanced RTK-GNSS equipment. These GCPs are represented by the cover plates' centers of the sports ground's sprinkler system. Their coordinates were determined in the World Geodetic System 84 (WGS84) Reference System with an achieved position accuracy of approximately 2 cm, using SAPOS-HEPS. Thereby, an adequate and reliable reference for further position measurements, which are intended to be at a best possible accuracy of 0.4 m (see Section 3.1.1) is given.

6.1.2. Positioning System Tests

The accuracy of the improved DGNSS positioning system was evaluated by carrying out measurements under realistic conditions. Therefore, the components of the positioning system were mounted to a MD4-1000 Sensor Platform (see Figure 6.1). Whereas the DGNSS antenna was integrated into the MUAV's cap to guarantee best possible acquisition of GNSS satellite signals, the receiver and radio modem were mounted to the take-up system on the bottom side as a workaround for the test and were not embedded into the body .

The positioning system tests were carried out on the predefined test field Horstmarer Landweg. For this purpose, the MD4-1000 was placed on a GCP and static position measurements were made for both setup types, the standard ublox LEA-4H with SBAS support (DGPS/EGNOS) and the implemented OEMStar with SAPOS support (DGNSS/SAPOS). One test per setup type was made with an update rate of 4

Figure 6.1.: Static position measurement of a MD4-1000 with mounted DGNSS-based positioning system on a predefined GCP.

Hz over a time period of approximately 15 minutes. Thereafter, the measurements' accuracies for the individual test were evaluated and the results are introduced in the following section.

6.1.3. Statistical Analysis of Positioning Accuracy

In a first step of the statistical analysis, the measured position data was prepared for processing. As the acquired GNSS coordinates are represented in WGS84 both as X,Y,Z and as geodetic longitude, latitude and ellipsoidal height, it is cumbersome to do metrical accuracy statements in horizontal and vertical direction. Therefore, the coordinates were transformed in the Universal Transversal Mercartor (UTM) system to facilitate the interpretability of the achieved accuracies by comparing Easting, Northing and Height values.

After the transformation, all heights were corrected by the identified antenna's offset and the coordinates of each position data set were compared to the coordinates of the GCP. By substraction, biases were calculated, representing the position errors in the directions of the according coordinate axis'. Figure 6.2 shows time series of the position errors, with respect to their individual coordinate directions. The proceeding in time is plotted on the X-axis, whereas the position error is plotted on the Y-axis. It is already apparent from this figure that the errors of the DGPS/EGNOS setup,

which are indicated by a blue line, show a higher amplitude as the errors of the DGNSS/SAPOS setup, represented as a red line. Moreover, Height errors in both setups are identified as larger than errors in Northing and Easting. This is traced back to a disadvantageous satellite constellation, due to the unchangable fact that satellites can only be observed from the upper side of the Earth's surface.

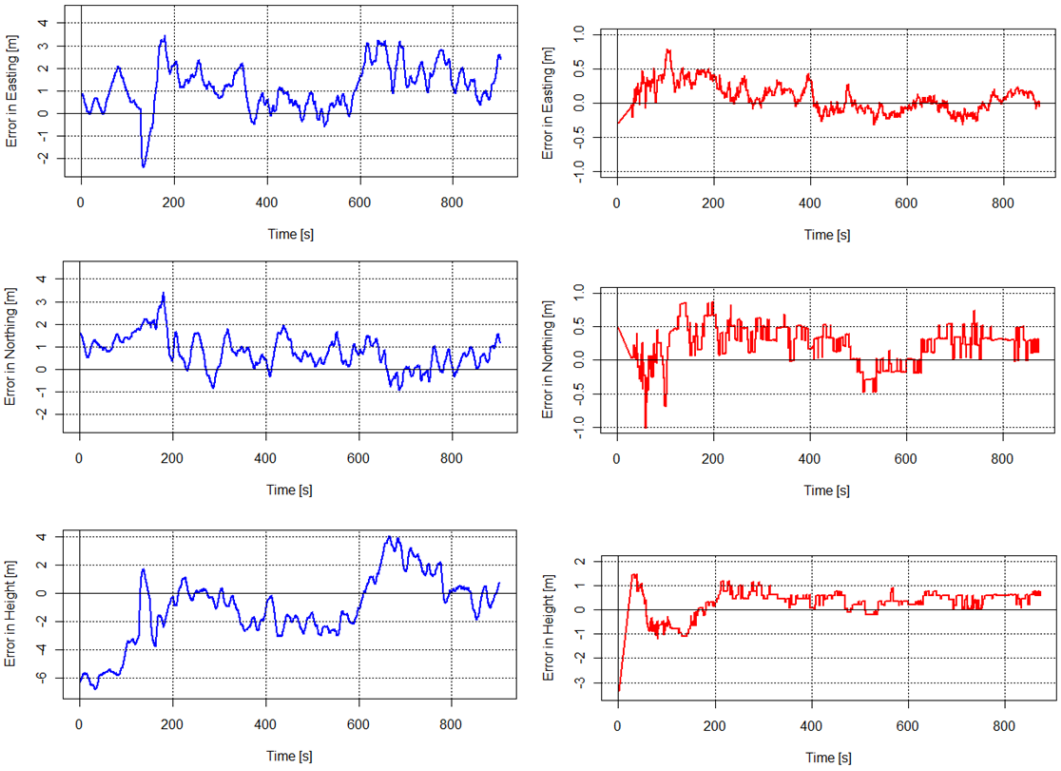

Figure 6.2.: Static position measurement for t = 900 s (DGPS/EGNOS = blue) and for t = 870 s (DGNSS/SAPOS = red) in Easting, Northing and Height.

The process of calculating standard deviations, to indicate the position accuracy, expects that position errors are normally distributed. Moreover, most of the statistical values used for the determination of horizontal position accuracy assume that the position errors are Rayleigh-distributed [Weltzien, 2003]. Even though this assumption can rarely be found to be completely correct, it is still essential to establish, whether the data sets are close to this assumption. Hence, the data sets were evaluated with regard to their normal distribution and the their position errors' degree of correlation.

The calculation of a correlation coefficient of the DGPS/EGNOS setup's position errors in Easting and Northing resulted in a value of -0.05, which shows that the errors of the two coordinate directions are almost completely uncorrelated. In contrast, the correlation coefficient of the DGNSS/SAPOS setup resulted in a higher, yet relatively small, value of 0.27 and indicates a slight positive correlation. With

regard to the result of the aforementioned setup, one would have assumed that the DGNSS/SAPOS setup's position errors in Easting and Northing would be uncorrelated. Whether the slight correlation originates from the surrounding conditions or the DGNSS/SAPOS setup's positioning mechanism, could not have been inferred.

Sequential GNSS position errors tend to move over time with respect to their preceeding error value and do not vary randomly around an expected mean. As a consequence, it is necessary to determine a minimal test duration, to ensure that this behavior does not excessively influence the conclusion of position accuracy. Therefore, each axis' position error sample was evaluated by its autocorrelation function. The autocorrelation function calculates the correlation between a sample's observations as a function of separation in time between these observations. A high correlation coefficient between two lagged time series' indicates that the observations of the first time series strongly influence the observations of the second. In contrast, a low correlation coefficient indicates a small influence. Regarding GNSS observations, autocorrelation at small lags is usually high and tends to decrease with proceeding in time. Therefore, the autocorrelation functions were analyzed by searching for the occurence of significant low autocorrelation. Subsequently, the corresponding lags were multiplied by the receivers' observation rate of 0.25 s to transform the lag values into observation times. Comparing these observation times to the overall test duration, one can infer whether the test duration was appropriate.

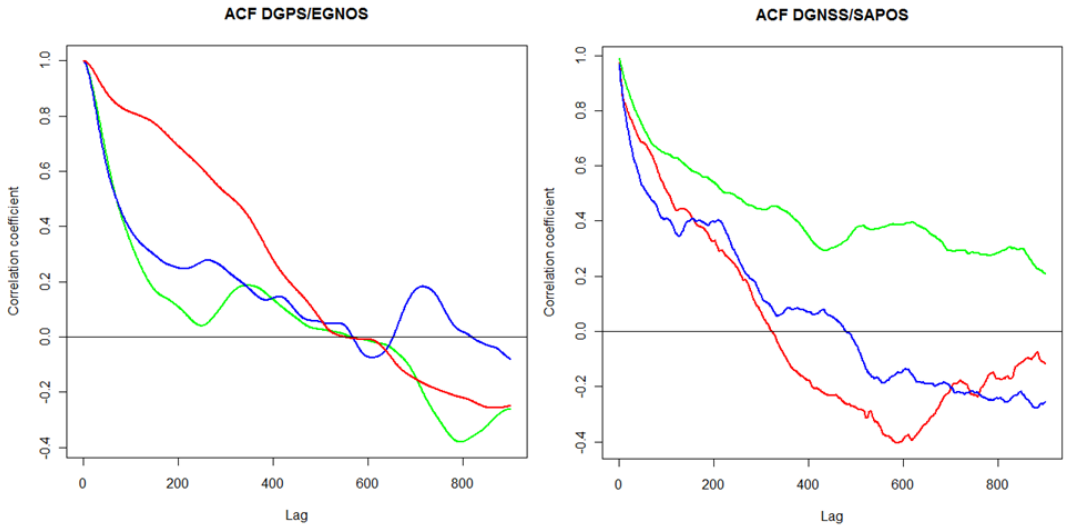

Figure 6.3.: Lag 1 to lag 900 autocorrelation function of postion errors in Easting, Northing and Height (Easting = green, Northing = blue, Height = red).

Figure 6.3 shows autocorrelation plots for position error time series in Easting (green), Northing (blue) and Height (red). These plots were calculated for lags at a range from 1 to 900. The number of lags that were used for calculation is plotted to

the X-axis, whereas the correlation coefficient between the initial position error time series and the lagged position error time series is plotted to the Y-axis. Both setups were considered, the DGPS/EGNOS setup on the left hand side and the DGNSS/-SAPOS setup on the right hand side. For the reason of comparability, the lag-specific correlation values were plotted as continuous lines, instead of representing them in discrete bar charts. This figure is quite revealing in several ways. Regarding both setups, it is obvious that the position errors vary slowly over time, indicated by decreasing functions without significant drops to a correlation coefficient of zero.
The DGPS/EGNOS setup shows that the correlation coefficient decreases quickly in Easting and already approximates zero at a lag of 240. Besides, the correlation coefficient in Northing decreases a bit slower compared to Easting. Together with the correlation coefficients in Easting and Height, it crosses zero at a lag of approx. 560. However the correlation coefficient of Height crosses zero together with Easting and Northing around a lag of 560, its decrease is slower. This indicates, that Height errors were higher correlated until this lag. Choosing a lag of 580, it is inferred that the position errors of the DGPS/EGNOS setup are no longer influenced by their initial errors after a period of 120 s.
Regarding the DGNSS/SAPOS setup, it is obvious that the correlation coefficients in Northing and Height quickly decrease and cross zero at a lag of 320 in Height and 490 in Northing. Therefore, a reach of low correlation close to the DGPS/EGNOS observations' can be assumed. In contrast, the correlation coefficients' curve in Easting shows a very slow decrease and approximates a value of 0.2 at a lag of 900. Therefore, a slight correlation can still be assumed at this point in time.

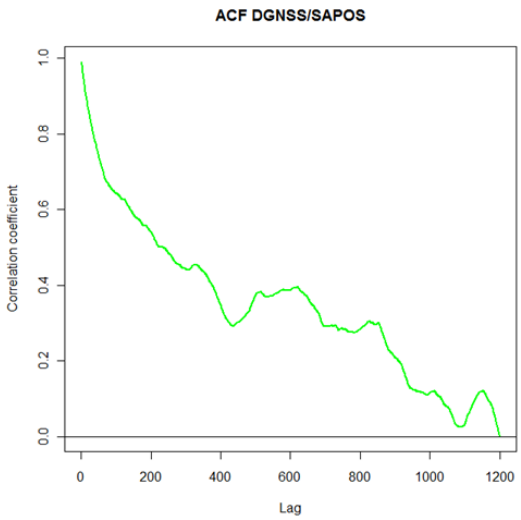

Figure 6.4.: Lag 1 to lag 1200 autocorrelation function of DGNSS/SAPOS postition errors in Easting.

Having calculated an autocorrelation function for higher lag numbers, a first crossing of zero occured at a lag of 1200, which is equivalent to a measurement time of 5 min (see Figure 6.4). As evaluation of autocorrelation loose validity with an increase of lag numbers, valuable conclusion of low correlation cannot be guaranteed. Regarding these findings, the test duration for both setups was set to a period of approx. 15 min to guarantee that initial position errors do not influence the complete measurements.

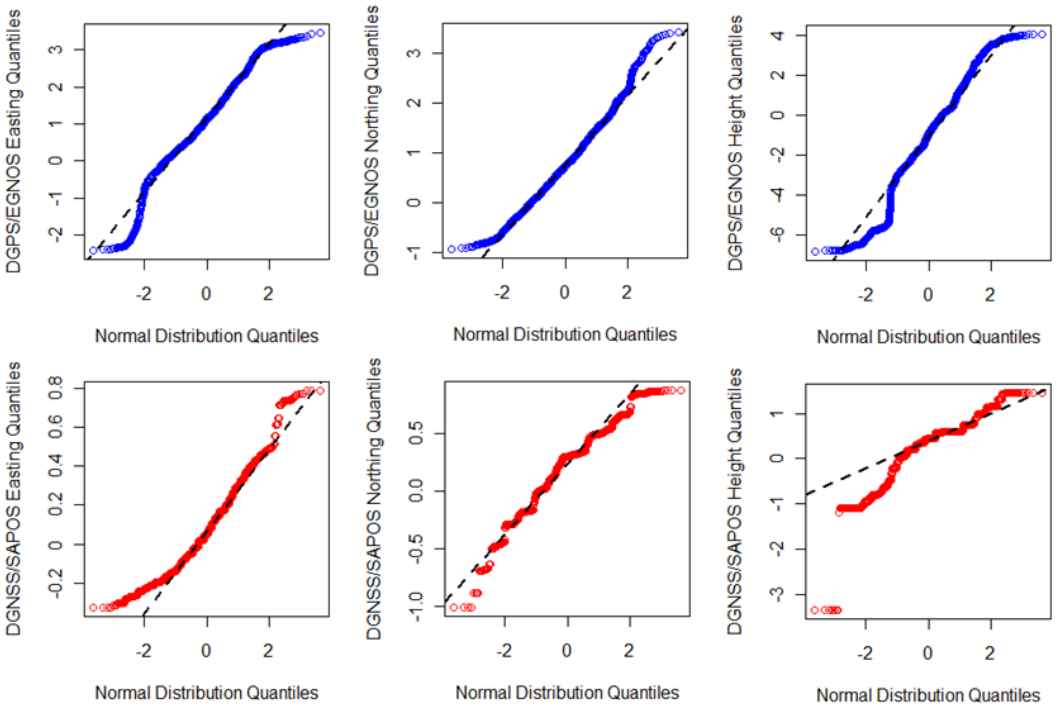

Figure 6.5.: Normal QQ-Plots of the position errors in Easting, Northing and Height (DGPS/EGNOS = blue and DGNSS/SAPOS = red).

Figure 6.5 shows a collection of Normal QQ-Plots of each position error direction. The Normal QQ-Plots are probability plots, which compare the individual position error probability distribution with a standard normal probability distribution, by plotting their quantiles against each other. The colored points correspond to quantiles of the position errors probability distributions (Y-axis) and are plotted against the same quantile of the standard normal probability distribution (X-axis). As these plots show, every position error probability distribution approaches the standard normal probability distribution to a certain degree. This is detected by a similarity of the colored points' trends with the dashed lines in the plots. However, it is obvious that none of the trends fit perfectly. As depicted in the figure, the tails of the position errors probability distributions do not follow the probablities of the standard normal distribution in that quantiles range.
Notably, the values in the lower left corner of the Normal QQ-Plot of the DGNSS/-

SAPOS Height postion error are far from being normally distributed. It is assumed, that this is due to outliers, such as the initial Height measurements and subsequent observation errors, which possess a very high deviation (see Figure 6.2). These outliers could be explained by the DGNSS/SAPOS setup's receiver's use of carrier phase smoothing techniques. These techniques lead to the fact that initial measurements are of minor accuracy compared to following measurements [Kettemann, 2003]. As this evaluation focuses on the determination of positioning accuracies under realistic conditions, the sample has not been modified to exclude these outliers.
With regard to the samples sizes, the amount of outliers was considered as small. As a consequence, normal distribution of the position errors in every direction was assumed for a following determination of position accuracies.

According to the results of correlation and deviation, normal distribution and Rayleigh distribution can be assumed for all position error data sets. Consequently, statistical values, such as one and two dimensional standard deviations were computed according to [Weltzien, 2003]. The results, as shown in Table 6.1 and Table 6.2, indicate that the improved DGNSS positioning system is working at a significant higher absolute positioning accuracy as the standard positioning system.

1D accuracy values (RMS) [m]			
	Easting	Northing	Height
DGPS/EGNOS	1.52	1.05	2.62
DGNSS/SAPOS	0.22	0.37	0.60

Table 6.1.: GNSS positioning tests' resulting 1D accuracy values (RMS).

It is apparent from Table 6.1 that the accuracy in determination of height has been improved to one fourth of the standard positionig sytem's accuracy. As the DGPS/-EGNOS setup resulted in a value of 2.6 m (RMS), the improved DGNSS/SAPOS setup resulted in a value of 0.6 m (RMS).

2D accuracy values [m]				
Accuracies see [Weltzien, 2003]	dRMS (63-68%)	2dRMS (93-98%)	CEP50 (50%)	CEP95 (95%)
DGPS/EGNOS	1.85	3.69	1.52	3.15
DGNSS/SAPOS	0.43	0.85	0.35	0.72

Table 6.2.: GNSS positioning tests' resulting 2D accuracy values.

Combining the standard deviations in Easting and Northing from Table 6.1, statistical accuracy values have been calculated for horizontal accuracy estimation. According to Table 6.2, the accuracy improvement to one fourth of the DGPS/EGNOS setup is confirmed. Regarding the 2dRMS, the DGNSS/SAPOS setup improves the position accuracy to approximately 0.9 m within a probability range of 93-98%. In contrast, the standard DGPS/EGNOS setup guarantees accuracy of a merely 3.7 m within the same probability range.

The aforementioned findings are visualized in a cumulative relative frequency graph (see Figure 6.6), which shows the added probability values of both setups' two dimensional position errors. These values are plotted to the Y-axis, as the resulting error within the corresponding probability range is plotted to the X-axis. The red line clearly shows the fast increase of the DGNSS/SAPOS setup's curve, which indicates that the position error increases slowly and stays at a relatively low level of 1.0 m within almost 100% of the measurements. In contrast, the blue line climbs slower. This means that the DGPS/EGNOS position error grows faster compared to the improved setup. The DGPS/EGNOS position error already exceeds 1 m in approximately 75% of the measurements.

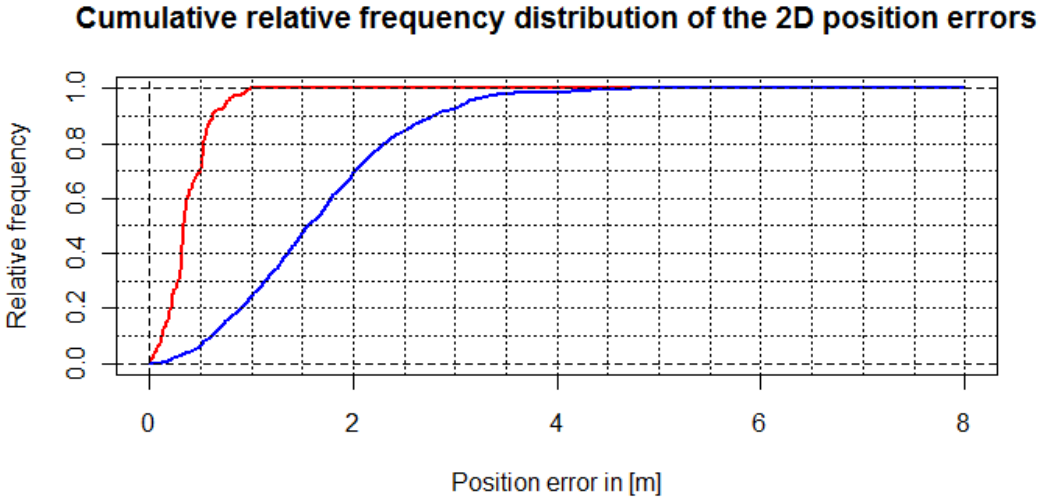

Figure 6.6.: Cumulative relative frequency graph of 2D position errors (DGPS/EGNOS = blue and DGNSS/SAPOS = red).

Finally, Figure 6.7 is quite revealing in several ways. As the position error in Easting is plotted to the X-axis and the position error in Northing is plotted to the Y-axis, the crosses indicate the discrete position errors of all measurements. Red colours indicate the improved DGNSS/SAPOS setup, whereas blue colors are used to indicate the DGPS/EGNOS setup. The dashed circles are drawn with a radius of the corresponding CEP95 value. They therefore contain 95% of the position measurements. It is obvious, that the red crosses of the DGNSS/SAPOS solution are centered around

the coordinates of the Ground Control Point (O,O) and possess a higher accuracy, compared to the blue crosses of the DGPS/EGNOS solution. The distribution of the DGNSS/SAPOS position errors is almost evenly around the GCP with a slight tendency towards northeast. In contrast, the distribution of the DGPS/EGNOS position errors cannot be considered as equally distributed around the origin. They are characterized by a strong tendency towards northeast and to a high degree lie within the corresponding quadrant. This tendency is not considered as a constant offset, which is given at every measurement at every point in time. It can be assumed that this tendency is due to the predominant satellite's constellation at the time the measurement was carried out. Furthermore, this tendency is expected to weaken and even to turn around by measurements at different constellations over longer time periods (e.g. 12 hours).

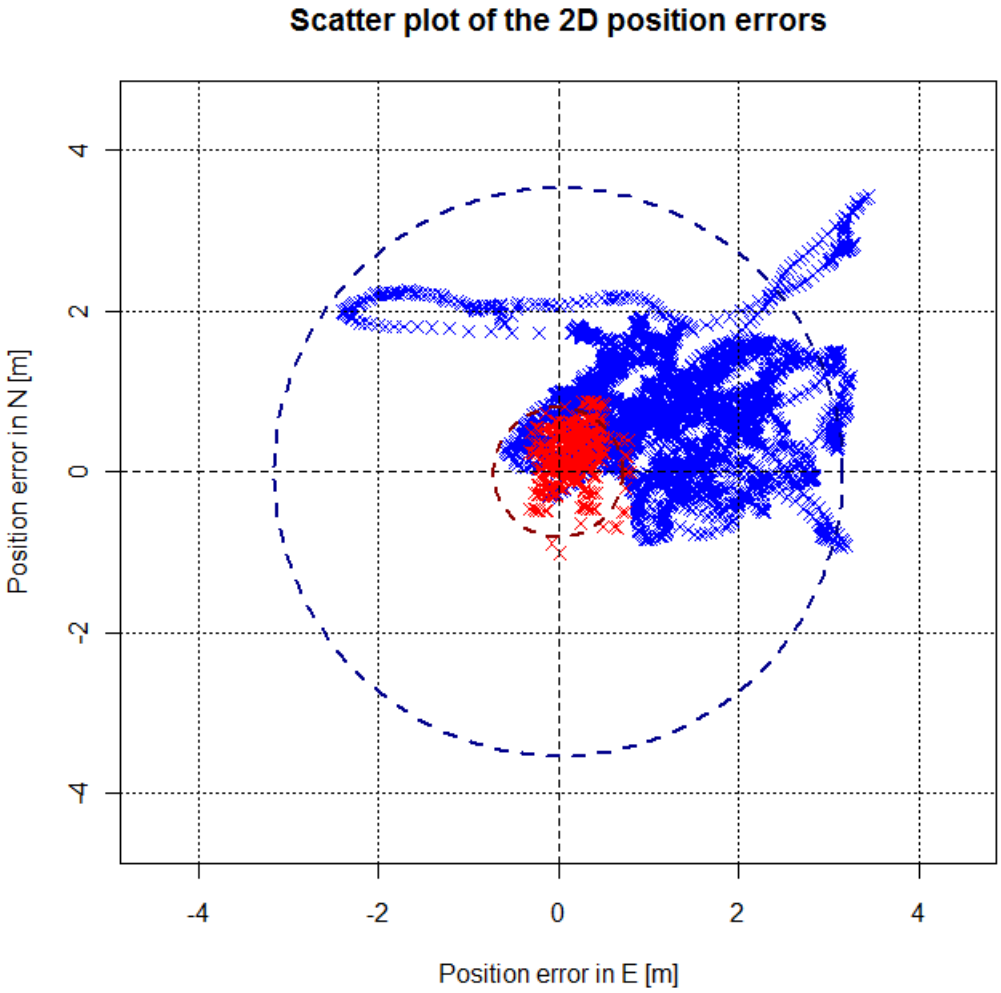

Figure 6.7.: Scatter plot of 2D position errors with (DGPS/EGNOS = blue and DGNSS/SAPOS = red and circles = CEP95).

Summarizing this statistical evaluation with regard to the aformentioned results, the aim of improving the MD4-1000's absolute position accuracy can be considered as achieved. The accuracy's improvement can be approximately determined by a factor of 4.

6.2. Evaluation of Sensor Platform Data Integration for Sensor Web Services

By developing a MD4-1000 Input-Plugin, the demanded functionality of the requirement analysis has been considered in several ways. Sections 4.2.1 and 5.2 showed a mechanism for near real-time navigation sensor data acquisition by querying a named pipe server, implemented by a mdCockpit Downlink Decoder application program. As the Input-Plugin is specified to query at a rate of 1 Hz, near real-time data acquisiton is guaranteed.
Moreover, an adapted MD4-1000 SensorML description was created and made accessible for the SPF (see Sections 4.2.2 and 5.2). As a consequence, the SPF is able to process the MD4-1000 sensor data in accordance with the demanded standards of SWE (see Section 3.2.2).
Section 2.3.2 showed that building on the Sensor Bus architecture, sensor data access for any kind of SWS can be established. Therefore, the sensor data has been integrated into the SPF, which functions as an intermediate between the MD4-1000 Sensor Platform and the Sensor Bus. Offering a suitable Sensor Bus Output-Plugin, which is mentioned in Section 4.2.3, the data is made accessible in the Sensor Web.

The following section will show two exemplary Output-Plugins, wich have been used to proof the established concept of the MD4-1000 navigation sensor data architecture.

MD4-1000 Input-Plugin and SensorVis Output-Plugin

Figure 6.8 illustrates a MD4-1000 mdCockpit Downlink Decoder Dialogue. As the mdCockpit application program already offers graphical user interfaces for monitoring the MD4-1000's sensor data and attitude on-the-fly, the developed Input-Plugin does not consider MD4-1000 specific visualization. However, the functionality of the MD4-1000 Input-Plugin has been tested by making use of the SensorVis Output-Plugin[7], which provides adapted visualization of sensor data, implemented in a 3D virtual globe environment.

[7] SensorVis Output-Plugin, published at 52 °North Initiative for Geospatial Open Source Software GmbH (https://wiki.52north.org/bin/view/Sensornet/SensorVis)

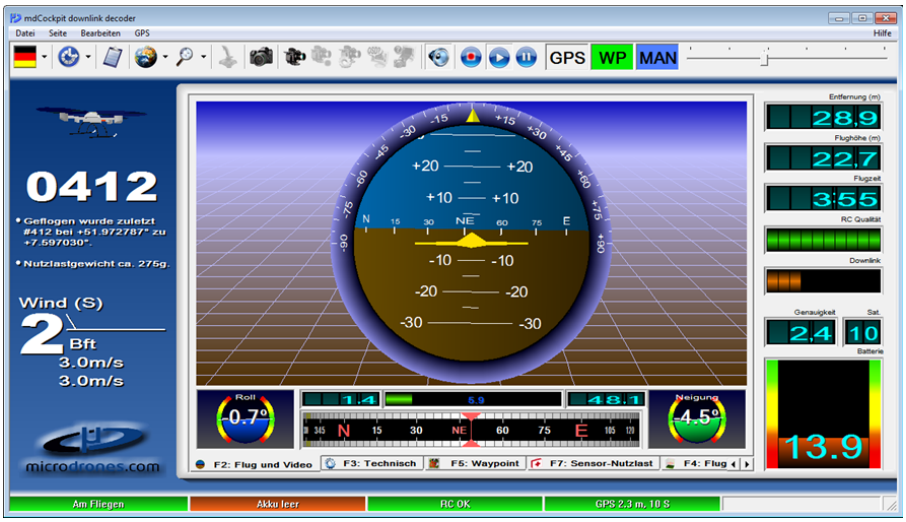

Figure 6.8.: Overview of MD4-1000 navigation sensor data in a mdCockpit Downlink Decoder Dialogue.

Figure 6.9 shows a sample flight track of a MD4-1000. The colored spheres represent the MUAV's positions and relative altitudes, as it is selected in the `Values` box on the left side of the map window. There, all retreived position values can be identified and selected. As it is obvious, the desired navigation sensor data values, which have been determined in the SensorML document in Listing 4.3, can be accessed and illustrated.

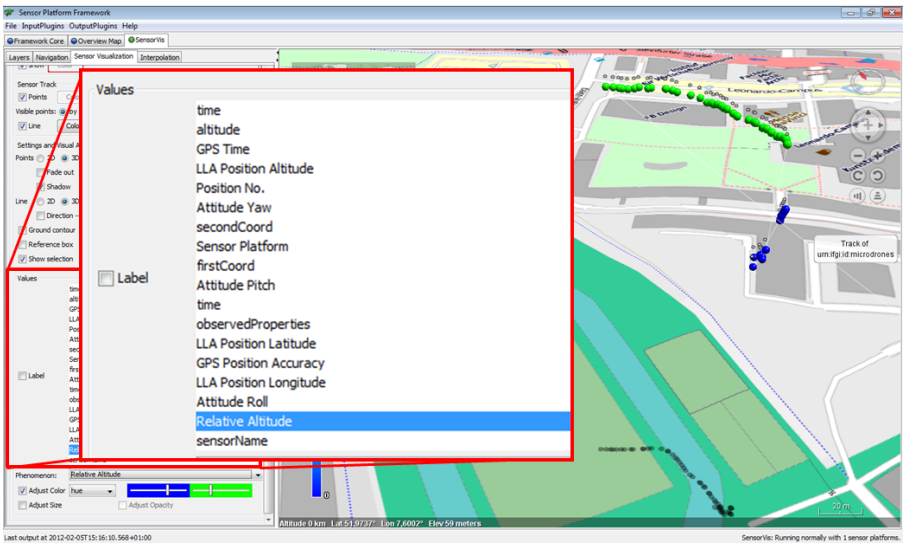

Figure 6.9.: Visualization of a MD4-1000 flight track using the SensorVis Output-Plugin.

MD4-1000 Input-Plugin and Sensor Bus Output-Plugin

The Sensor Bus is a logical concept and, therefore, can be implemented using various infrastructures. The Sensor Bus Output-Plugin builds on the Extensible Messaging and Presence Protocol (XMPP), which is an open-standard communications protocoll for Instant Messaging based on XML [Jabber, 2004].

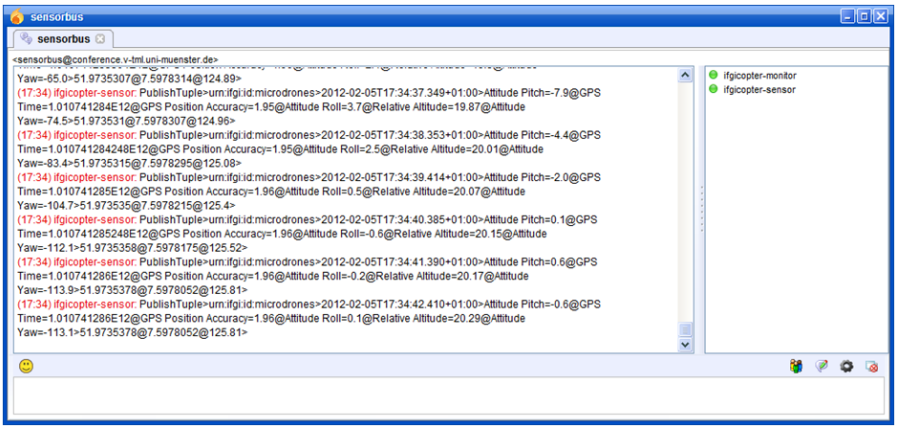

Figure 6.10.: Visualization of the Sensor Bus Output-Plugin's Bus Messages with Spark Instant Messenger.

Figure 6.10 illustrates a conversation dialog of the Spark Instant Messenger. This messenger was used to visualize Bus Messages, which are published to the Sensor Bus via the Sensor Bus Output-Plugin. The received Bus Messages clearly show that the required navigation sensor values are entirely published as tuples, specified by `sensor id` and `system time`, containing `Attitude Pitch`, `GPS Time`, `Position Accuracy`, `Attitude Roll`, `Relative Altitude`, `Attitude Yaw`, `LLA Latitude`, `LLA Longitude` and `LLA Altitude`. Regarding the `GPS Time` property, it is remarkable that the value does not increase constantly in the same intervall as the `system time`. As incoming `GPS Time` values do not show this behaviour, it is assumed that the interpolation mechanism of the SPF's core influences the output in a negative way.

7. Discussion and Outlook

Regarding the prototypical implementation of the concepts and architectures, which have been introduced in Chapter 4, one can infer that the requirements of Chapter 3 have been considered and a complete data flow from generating improved position data to making this data accessible by SWSs has been established. Chapter 5 outlined this prototypical implementation, whereas Chapter 6 proved its functionality. The previous position accuracy has been significantly improved by a factor of 4 and the integration of the navigation sensor data has been established via an adapted MD4-1000 Input-Plugin in combination with the SPF and the Sensor Bus technology. Nevertheless, implementation and testing brought up ideas for improvement, which will be discussed in the following.

The results of the positioning tests showed a significant improvement in position accuracy. As the data basis of the analysis was considered as adequate but not optimal, further testing should be carried out, to confirm the findings of Section 6.1.3. Moreover, it is of great interest whether these findings can be proven in dynamic flight tests. In this scenario, the determination of positon errors cannot be established by the use of GCPs. Therefore, the evaluation should be supported by the use of geodetic total stations, which are able to track the MUAV's position at an accuracy of subcentimeter level [Bäumker and Przybilla, 2011].

As Section 3.1.2 already mentioned, the modular DGNSS positioning system should be further improved by exchanging its DGNSS receiver with a RTK-GNSS enabled receiver. First RTK-GPS tests have been carried out by a working group at the University of Technology in Hamburg. This group presents an implementation of a RTK-GPS positioning system, embedded into a MUAV, which guarantees a horizontal accuracy of 8 cm (2dRMS) [Pilz et al., 2011].

The implemented Allsat come2ascos radio modem originated problems in its power supply, which had to be established via the telemetry downlink's power line. Moreover, the radio modem was not able to connect to SAPOS on every attempt. As this problem was detected by coincidence and could not be reproduced, a solution to fix it has not been found yet. Therefore, it would be utile to exchange the radio modem by a more stable and lower power consumpting one. As the DGNSS positioning system is designed in a modular way, this exchange could be carried out quite easily.

Another possibility is to integrate a radio modem directly into the MD4-1000's NC. By this, a permanent MD4-1000 internet connection could be established, not only for signal correction acquisiton, but also for tasking the MD4-1000, i.e. to execute a changed waypoint flight.

The MD4-1000 sensor data stream contains more information than just the navigation sensor data. In order to be able to use the additional sensor and system information in application systems, the developed Input-Plugin should be augmented by this information. As the Sensor Platform Framework's interpolation mechanism does not support data output without interpolation according to the occurence of specified <spf:outputProperties>, the SPF needs to be adapted to offer data output of untouched values, i.e. RC commands, absolute numbers like the amount of visible space vehicles or numbers that indicate a certain system status.

Finally, the use of the named pipe server, which is implemented by the mdCockpit Downlink Decoder Dialogue, must be questioned. Running the Downlink Decoder Dialogue and the Sensor Platform Framework together, the perfomance of the SPF, especially of the SensorVis Output-Plugin, is significantly reduced. While retreiving real-time data, this behaviour is barely acceptable by using a Intel(R)Core(TM)2 CPU at 1.66 GHz and 3.00 GB RAM. Emulating flight data from flight logs, causes temporarily hold-ups of the complete system. These findings have not been analyzed in detail, yet. As these observations never occured, while using the SPF and its Output-Plugins, it is assumed, that the Downlink Decoder Dialogue causes this intensive wastage of system resources. Therefore, performance tests need to be carried out to clarify this problem. In the case that the assumption is correct, an improvement of the mdCockpit application program is required. As this improvement is manufacturer-dependent, a decoding without using the named pipe server must be considered. Consequently, the data stream needs to be accessed at the base receiver station's downlink interface and decoded by implementing an appropriate software.

8. Summary

This Master's Thesis covers the development of an improved MD4-1000 Sensor Platform specific positioning system and the integration of the enhanced navigation sensor data into the Sensor Web.

In the beginning of this work, the need of improved and standardized MUAV position data for use cases in industry and science is assessed. Subsequently, the MD4-1000 Sensor Platform, the different possibilities of GNSS-based position acquisition, as well as the concepts of the Sensor Web Enablement initiative, the Sensor Bus and the Sensor Platform Framework are explained. After introducing these basic components and concepts, a requirement analysis for designing an improved DGNSS positionig system and for providing the MD4-1000's navigation sensor data to any kind of Sensor Web Service, is carried out.

In a consequent conception phase, a hard- and software architecture is designed, according to the aforementioned findings, and realized by a subsequent prototypical implementation. The improvement in position is acquired by developing a modular DGNSS positioning system, which is able to receive and process a combination of GPS, GLONASS and SAPOS signals. After a necessary adaption of the Navigation Controller's firmware, the enhanced data is embedded into the MD4-1000 navigation system and transmitted to a base receiver station. A microdrones-specific application program implements a named pipe server, which is queried by an established Input-Plugin to receive the data. In the following, the data is processed by the Sensor Platform Framework and standardized in a predefined way. These definitons are made with regard to the standards of the Sensor Web Enablement for an adequate data output via a suitable Sensor Bus Output-Plugin.

In addition, an evaluation of the aforementioned implementation is carried out. The improvement in positioning is analyzed by realistic field tests and deduced statistical estimations. The navigation sensor data's integration into the Sensor Platform Framework as well as into the Sensor Bus is examined by checking the data output, generated by two existing Output-Plugins. Subseqently, the findings of the evaluation are discussed and further possibilities of improvement are presented.

Bibliography

[Allsat GmbH 2008] Allsat GmbH: *Quick Reference come2ascos and come2ascos blue*, March 2008 5.3, 5.1.2

[Antcom Corporation 2009] Antcom Corporation: *Antcom New G5 Antennas*, June 2009 5.1.1, 5.1

[AXIO-NET GmbH 2009] AXIO-NET GmbH: *ascos ED*. http://www.ascos.de/uploads/tx_v3downloads/Produktblatt_ED_2009_01.pdf. Version: 2009 2.2.2

[Bauer 2003] Bauer, Manfred: *Vermessung und Ortung mit Satelliten*. 5. Auflage. Herbert Wichmann Verlag, 2003 2.1.2, 2.2, 2.2.1, 2.7, 2.2.1, 3.1.2

[Bezirksregierung Köln 2010] Bezirksregierung Köln: *Varianten der Echtzeitdienste im SAPOS NRW*. http://www.bezreg-koeln.nrw.de/brk_internet/organisation/abteilung07_produkte/raumbezug/sapos/heps/varianten.pdf. Version: December 2010 5.1.3

[Bezirksregierung Köln 2012] Bezirksregierung Köln: *SAPOS - HEPS: Hochpräziser Echtzeit-Positionierungs-Service*. http://www.bezreg-koeln.nrw.de/brk_internet/organisation/abteilung07_produkte/raumbezug/sapos/heps/index.html. Version: 2012, Call: 22.01.2012 5.1.3

[Botts et al. 2008] Botts, Mike ; Percivall, George ; Reed, Carl ; Davidson, John: OGC Sensor Web Enablement: Overview and High Level Architecture. In: Nittel, S (Ed.) ; Labrinidis, A (Ed.) ; Stefanidis, A (Ed.): *GeoSensor Networks, 2nd International Conference, GSN 2006* Bd. 4540, Springer, 2008, S. 175–190 2.3, 2.3.1, 2.3.1

[Broering et al. 2010] Broering, Arne ; Foerster, Theodor ; Jirka, Simon ; Priess, Carsten: Sensor Bus: An Intermediary Layer for Linking Geosensor Networks and the Sensor Web. In: Liao, Lindi (Ed.): *Proceedings of COM.Geo 2010, 1st International Conference on Computing for Geospatial Research and Applications*, ACM, 2010 2.3.2, 2.11, 2.3.2, 2.2, 2.3.2

[Buss 2011] Buss, Holger: *heightsensor*. http://mikrokopter.de/ucwiki/en/heightsensor. Version: 2011, Call: 27.08.2011 2.1.3

[Bäumker and Przybilla 2011] Bäumker, M ; Przybilla, H.-J: INVESTIGATIONS ON THE ACCURACY OF THE NAVIGATION DATA OF UNMANNED AERIAL VEHICLES USING THE EXAMPLE OF THE SYSTEM MIKROKOPTER. In: Eisenbeiss, Henry (Ed.) ; Kunz, M (Ed.) ; Ingensand, H (Ed.): *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Proceedings of the International Conference on Unmanned Aerial Vehicle in Geomatics (UAV-g). September 14 - 16, 2011. Zürich, Switzerland* Bd. XXXVIII-1/C22, ACM, 2011 7

[Büchi 2010] Büchi, Roland: *Faszination Quadrokopter*. Verlag für Technik und Handwerk, 2010 2.1, 2.1

[Dodel and Häupler 2009] Dodel, Hans ; Häupler, Dieter: *Satellitennavigation*. 2. Auflage. Springer, 2009 2.2.2

[El-Rabbany 2002] El-Rabbany, Ahmed: *Introduction to GPS*. Artech House Inc., 2002 2.1.2, 2.8, 2.2.2, 2.9

[European Space Agency 2012] European Space Agency: *What is EGNOS?* http://www.esa.int/esaNA/egnos.html. Version: 2012, Call: 14.01.2012 2.2.2

[Hoffmann 2010] Hoffmann, Jonas: *Kombination von Inertialsensoren und GPS zur Navigation*, Institut für Informatik der Freien Universität Berlin, Master's Thesis, 2010 2.1.1, 2.1.1

[Hohpe and Woolf 2003] Hohpe, Gregor ; Woolf, Bobby: *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Longman Publishing, 2003 2.3.2

[Jabber Software Foundation 2004] Jabber Software Foundation: *Extensible Messaging and Presence Protocol (XMPP): Core*. http://xmpp.org/rfcs/rfc3920.html. Version: 2004, Call: 05.02.2012 6.2

[Jirka et al. 2010] Jirka, Simon ; Broering, Arne ; Alexander, Walkowski C: Sensor Web in Practice - Building Productive Systems. In: *GEOInformatics - Magazine for Surveying, Mapping & GIS Professionals* (2010), September, S. 42–25 1.1, 2.3, 2.3.1, 2.10, 2.3.1, 2.3.1

[Kettemann 2003] Kettemann, Rainer: GPS-Verfahren - Einsatzgebiete - Rahmenbedingungen - Kombinationslösungen. In: *GPS für GIS im Umweltbereich und Naturschutz* (2003) 2.2.1, 6.1.3

[Microdrones GmbH 2010a] Microdrones GmbH: *mdCockpit Standard Edition - Preliminary Users Manual*, May 2010 2.1.6

[Microdrones GmbH 2010b] Microdrones GmbH: *microdrones downlink*, August 2010 2.1.6

[Microdrones GmbH 2011a] Microdrones GmbH: *Adaptation of languages for the mdCockpit software*, September 2011 2.1.6, 4.2.1, 5.2, 5.2

[Microdrones GmbH 2011b] Microdrones GmbH: *Bedienungsanleitung md4-1000*, November 2011 2.1, 2.3, 2.1.3, 2.1.5, 2.6, 2.1.6

[Microdrones GmbH 2011c] Microdrones GmbH: *Flyer microdrones md4-1000*. http://www.microdrones.com/downloads/md4-1000/md4-1000_Flyer_englisch_web.pdf. Version: 2011 2.1

[National Marine Electronics Association 2002] National Marine Electronics Association: *NMEA 3.01*, January 2002 4.1.2

[NovAtel Inc. 2003] NovAtel Inc.: *GPS Position Accuracy Measures*, December 2003 5

[NovAtel Inc. 2010a] NovAtel Inc.: *OEMStar Firmware Reference Manual*, August 2010 5.1.2

[NovAtel Inc. 2010b] NovAtel Inc.: *OEMStar Installation and Operation User Manual*, October 2010 5.1.1, 5.2

[Open Geospatial Consortium Inc. (OGC) 2007a] Open Geospatial Consortium Inc. (OGC): *Observations and Measurements - Part 1 - Observation schema*. http://www.opengeospatial.org/standards/om. Version: 2007, Call: 30.01.2012 2.3.1

[Open Geospatial Consortium Inc. (OGC) 2007b] Open Geospatial Consortium Inc. (OGC): *OpenGIS® Sensor Model Language (SensorML) Implementation Specification*. http://www.opengeospatial.org/standards/sensorml. Version: 2007, Call: 30.01.2012 2.3.1

[Open Geospatial Consortium Inc. (OGC) 2007c] Open Geospatial Consortium Inc. (OGC): *Sensor Observation Service*. http://www.opengeospatial.org/standards/sos. Version: 2007, Call: 30.01.2012 2.3.1

[Open Geospatial Consortium Inc. (OGC) 2012] Open Geospatial Consortium Inc. (OGC): *OGC® Standards and Specifications*. http://www.opengeospatial.org/standards. Version: 2012, Call: 30.01.2012 2.3.1

[Pilz et al. 2011] Pilz, Ulf ; GropengieSSer, Willem ; Walder, Florian ; Witt, Jonas ; Werner, Herbert: Quadrocopter Localization Using RTK-GPS and Vision-Based Trajectory Tracking. In: Jeschke, S (Ed.) ; Liu, H (Ed.) ;

Schilberg, D (Ed.): *Intelligent Robotics and Applications, 4th International Conference, ICIRA 2011* Bd. 7101, Springer, 2011, S. 12–21 7

[Radio Technical Commission for Maritime Services 2004] Radio Technical Commission for Maritime Services: *RTCM STANDARD 10410.0*, September 2004 4.1.2

[Remondino et al. 2011] Remondino, F ; Barazzetti, L ; Nex, F ; Scaioni, M ; Sarazzi, D: UAV PHOTOGRAMMETRY FOR MAPPING AND 3D MODELING - CURRENT STATUS AND FUTURE PERSPECTIVES -. In: Eisenbeiss, Henry (Ed.) ; Kunz, M (Ed.) ; Ingensand, H (Ed.): *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Proceedings of the International Conference on Unmanned Aerial Vehicle in Geomatics (UAV-g). September 14 - 16, 2011. Zürich, Switzerland* Bd. XXXVIII-1/C22, ACM, 2011 1.1

[Rieke 2010] Rieke, Matthes: *Entwicklung eines Frameworks zur Anbindung von Multi-Sensor-Plattformen an das Sensor Web*, Institute for Geoinformatics at Westfälische Wilhelms-Universität Münster, Master's Thesis, December 2010 2.3.1, 2.2, 2.3.3

[Rieke et al. 2011] Rieke, Matthes ; Foerster, Theodor ; Broering, Arne: *Unmanned Aerial Vehicles as Mobile Multi-sensor Platforms*. 2011. – The 14th AGILE International Conference on Geographic Information Science. 18.-21. April 2011. Utrecht, Netherlands. 2.3.3, 2.12

[Tamazin 2011] Tamazin, Mohamed Essam Hassan R.: *Benefits of Combined GPS/GLONASS Processing for High Sensitivity Receivers*, Department of Geomatics Engineering at the University of Calgary, Master's Thesis, 2011 3.1.1

[u-blox AG 2007] u-blox AG: *Antaris 4 GPS Modules Data Sheet*. http://www.u-blox.com/images/downloads/Product_Docs/LEA-4x_Data_Sheet%28GPS.G4-MS4-06143%29.pdf. Version: 2007 2.1.2

[Weltzien 2003] Weltzien, Cornelia: GPS-Empfänger Vergleich / Deutsche Landwirtschafts-Gesellschaft e.V. 2003 (5148F). – DLG-Prüfbericht 6.1.3, 6.1.3, 6.1.3

[Wendel 2007] Wendel, Jan: *Integrierte Navigationssysteme*. Oldenbourg Wissenschaftsverlag GmbH, 2007 2.1.1, 2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.4, 3.1.3

[Woodman 2007] Woodman, Oliver J.: An introduction to inertial navigation / Computer Laboratory of the University of Cambridge. 2007. – Technical Report 2.1.1, 2.4, 2.1.1, 2.1.4

[Zogg 2011] Zogg, Jean-Marie: *GPS und GNSS: Grundlagen der Ortung und Navigation mit Satelliten.* u-blox AG, 2011 2.2.2

A. Appendix

A.1. Software

This section introduces the software application programs, which have been used for writing this Master's Thesis, configuring the GNSS hardware components and implementing the developed program code.

Eclipse IDE

The Eclipse IDE has been used for the textural design and for the implementation of all program code.
Website: http://www.eclipse.org

GIMP

The GNU Image Manipulation Program has been used for image manipulation.
Website: http://www.gimp.org

LaTeX

This thesis has been written with LaTeX. Thereby, the TeX Live distribution has been chosen. The Eclipse IDE with the Texlipse-Plugin has been used for editing.
Websites: http://www.tug.org/texlive, http://texlipse.sourceforge.net

PowerPoint

Most of the figures have been designed with Microsoft PowerPoint.
Website: http://office.microsoft.com/en-us/powerpoint/

R

Statistical analysis and graphs have been made with the R software environment for statistical computing and graphics.
Website: http://www.r-project.org/

Tera Term

The Tera Term software terminal emulator has been used for configuring the NovAtel OEMStar receiver and the Allsat come2ascos radio modem.
Website: http://hp.vector.co.jp/authors/VA002416/teraterm.html

A.2. Data CD

A compact disc with a digital appendix is attached to this Master's Thesis, containing the following data.

Manual

A manual for installing and running the developed MD4-1000 sensor data architecture.
Location: cd:\

Java Runtime Environment

A JRE installer. As the Sensor Platform Framework builds on the Java programming language, a Java Runtime Environment needs to be installed.
Location: cd:\Java\

Master's Thesis

A PDF-version of this Master's Thesis.
Location: cd:\MastersThesis\

Microdrones: mdCockpit and Log Files

The mdCockpit application program, a firmware installer for the developed MD4-1000 NC firmware, an exemplatory flight log file and the recorded position test files for the DGPS/EGNOS, as well as the DGNSS/SAPOS setup.
Location: cd:\Microdrones\

Source Code: MD4-1000 Input-Plugin and MD4-1000 NC Parsing Routine

The developed source code of the MD4-1000 Input-Plugin and the MD4-1000 NC parsing routine.
Location: cd:\SourceCode\

Spark Instant Messenger

A SparkIM installer for receiving the Sensor Bus Output-Plugin's Bus Messages.
Location: cd:\SparkIM\

Sensor Platform Framework

An executable version of the Sensor Platform Framework, including the MD4-1000 Input-Plugin, as well as the SensorVis and Sensor Bus Output-Plugins.
Location: cd:\SPFramework\

Test Field Horstmarer Landweg

Overview of the test field Horstmarer Landweg and coordinates of the established GCPs.
Location: cd:\TestField\

XML Definition

The developed MD4-1000 Input-Plugin specific XML file, containing XML and SensorML definitions.
Location: cd:\XML\

Affirmation

I herewith declare that I have written the Master's Thesis

> *Improved DGNSS-based Positioning of Micro UAV Platforms for Sensor Web Services*

independently. I have not used any other sources or aids than those I explicitly reffered to in this work. I clearly marked and specified all of the sources that I employed when producing this academic work, either literally or in content.

Mï£ · nster, February 6, 2012

Jakob Geipel