UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4 Informatik

Anschluß eines GPS-Moduls an einen Mikrocontroller und Versuch der Präzitionssteigerung

Diplomarbeit

zur Erlangung des Grades eines
Diplom-Informatikers
im Studiengang Informatik

vorgelegt von:

Mario Schneider
Matrikelnummer: 119820182

Erstgutachter:	Prof. Dr. Christoph Steigner Institut für Informatik
Zweitgutachter:	Dr. Merten Joost Institut für Integrierte Naturwissenschaften

Koblenz, im August 2010

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Inhaltsverzeichnis

Abkürzungsverzeichnis

ALU ..... **A**rithmetic **L**ogical **U**nit
ASCII ..... **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
C/A ..... **C**orase/**A**quisition
CDMA ..... **C**ode **D**ivision **M**ultiple **A**ccess
CMOS ..... **C**omplementary **M**etal **O**xide **S**emiconductor
CRC ..... **C**yclic **R**edundancy **C**heck
CS ..... **C**hip-**S**elect
CS ..... **C**ommercial **S**ervice
CTC ..... **C**lear **T**imer on **C**ompare match
DDR ..... **D**ata **D**irection **R**egister
DGPS ..... **D**ifferential **G**lobal **P**ositioning **S**ystem
DOP ..... **D**ilution **O**f **P**recision
EGNOS ..... **E**uropean **G**eostationary **N**avigation **O**verlay **S**ervice
EICRA ..... **E**xternal **I**nterrupt **C**ontrol **R**egister **A**
EIFR ..... **E**xternal **I**nterrupt **F**lag **R**egister
EIMSK ..... **E**xternal **I**nterrupt **M**ask Register
ESA ..... **E**uropean **S**pace **A**gency
FIFO ..... **F**irst **I**n **F**irst **O**ut
FSK ..... **F**requency **S**hift **K**eying
GDOP ..... **G**eometric **D**ilution **O**f **P**recision
GEO ..... geostationären Orbit
GLONASS .... **GLO**bales **NA**vigations-**S**atelliten-**S**ystem
GNSS ..... **G**lobal **N**avigation **S**atellite **S**ystem
GPS ..... **G**lobal **P**ositioning **S**ystem
HDOP ..... **H**orizontal **D**ilution **O**f **P**recision
HOW ..... **H**and **O**ver **W**ord
IMU ..... **I**nertial **M**easurment **U**nit
INS ..... **I**nertial-**N**avigationssystem
ISP ..... **I**n-**S**ystem **P**rogrammer
ISR ..... **I**nterrupt **S**ense **C**ontrol
ISR ..... **I**nterrupt **S**ervice **R**outine

LEO ..... **L**ow **E**arth **O**rbit
LFSR ..... **L**inear **F**eedback **S**hift **R**egister
LORAN ..... **LO**ng **R**ange **A**rea **N**avigation
LVTTL ..... **L**ow **V**oltage **T**ransistor-**T**ransistor-**L**ogik
MISO ..... **M**aster **I**n **S**lave **O**ut
MOSI ..... **M**aster **O**ut **S**lave **I**n
MSAS ..... **M**ulti-Functional **S**atellite **A**ugmentation **S**ystem
NAVSTAR-GPS **NAV**igational **S**atellite **T**iming **A**nd **R**anging - **G**lobal **P**ositioning **S**ystem
NC ..... **N**ot **C**onnected
NDB ..... **N**on **D**irectional **B**eacons
NGA ..... **N**ational **G**eospatial-Intelligence **A**gency
NMEA ..... **N**ational **M**arine **E**lectronics **A**ssociation
NTRIP ..... **N**etworked **T**ransport of **R**TCM via **I**nternet **P**rotocol
OCR ..... **O**utput **C**ompare **R**egister
OMEGA ..... **OM**nidirectional **E**lectronic **G**round **A**ntennas
P-Code ..... **P**rotected oder **P**recise-Code
PAP ..... **P**rogramm **A**blauf **P**lan
PDOP ..... **P**osition **D**ilution **O**f **P**recision
POR ..... **P**ower **O**n **R**eset
PPS ..... **P**recise **P**ositioning **S**ervice
PRN ..... **P**seudo **R**andom **N**oise
PRS ..... **P**ublic **R**egulated **S**ervice
PWM ..... **P**ulse-**W**idth **M**odulation,Pulsweitenmodulation
RTCM ..... **R**adio **T**echnical **C**ommission for **M**aritime Services
RxD ..... Receive Data
SA ..... **S**elective **A**vailability
SAR ..... **S**earch **A**nd **R**escue
SCK ..... **S**ystem **C**lo**c**k
SoL ..... **S**afety-**o**f-**L**ife
SPCR ..... **SP**I **C**ontrol **R**egister
SPDR ..... **SP**I **D**ata **R**egister
SPI ..... **S**erial **P**eripheral **I**nterface
SPIF ..... **SP**I **I**nterrupt **F**lag
SPS ..... **S**tandard **P**ositioning **S**ervice
SPSR ..... **SP**I **S**tatus **R**egister
SS ..... **S**lave-**S**elect
TCCR ..... **T**imer **C**ounter **C**ontrol **R**egister

TDOP ..... **T**ime **D**ilution **O**f **P**recision
TIMSK ..... **T**imer/Counter **I**nterrupt **M**ask
TLM ..... **T**ele**m**etry-Word
TOIE ..... **T**imer/Counter **O**verflow **I**nterrupt **E**nable
TTFF ..... **T**ime **T**o **F**irst **F**ix
TTL ..... **T**ransistor-**T**ransistor-**L**ogic
TxD ..... Transmit Data
UART ..... **U**niversal **A**synchronous **R**eceiver and **T**ransmitter
UDR ..... **U**ART I/O **D**ata **R**egister
VDOP ..... **V**ertical **D**ilution **O**f **P**recision
WAAS ..... **W**ide **A**rea **A**ugmentation **S**ystem
WGS ..... **W**orld **G**eodetic **S**ystem

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

1 Einleitung und Zielsetzung

In einer Welt, die immer mehr durch den Einsatz von Technik geprägt wird, gewinnen Systeme, welche eigenständig Aufgaben in einem vorgegebenen Kontext übernehmen können, mehr und mehr an Bedeutung. Solche Systeme werden als autonome Systeme bezeichnet, die ein eigenständiges Verhalten und Interaktionen mit ihrer Umwelt und anderen Systemen beinhalten. Zu diesen Systemen zählen auch autonome mobile Roboter. Sie sind sozusagen intelligente Fahrzeuge, die ihnen übertragene Aufgaben ohne menschliche Unterstützung durchführen können. Um ihre Umwelt wahrzunehmen, mit ihr interagieren und in ihr navigieren zu können benötigen diese Systeme verschiedene Arten von Sensoren. Somit ist die Robotikforschung stark interdisziplinär.

Diese Arbeit befasst sich ausschliesslich mit dem Bereich der Navigation, insbesondere mit der Positionsbestimmung, welche für eine gute Navigation die Grundlage bildet. Ein heutzutage häufig benutztes, weltweit verfügbares System zur Positionsbestimmung ist GPS. Ziel dieser Arbeit ist der Anschluss eines GPS-Moduls an einen Mikrocontroller und der Versuch die Positionsgenauigkeit lokal zu präzisieren, um so einem autonomen System die Möglichkeit zu eröffnen, Aufgaben, die eine höhere Genauigkeit erfordern, zu bewerkstelligen.

Das zu entwickelnde System sollte aus möglichst kostengünstigen Komponenten zusammengestellt werden und die gleiche Schnittstelle wie das GPS-Modul realisieren. Die dieser Arbeit zugrundeliegende Vorgehensweise besteht aus der Ermittlung von Korrekturdaten durch eine Referenzstation, Übermittlung dieser Daten per Funk, Verrechnung dieser Daten und Ausgabe NMEA-konformer Datensätze über eine serielle Schnittstelle.

Im zweiten Kapitel dieser Arbeit liegt das Hauptaugenmerk auf der Satellitennavigation. Zunächst wird die Geschichte der Navigation von den Anfängen bis hin zur heutigen Satellitennavigation kurz revue passieren lassen. Anschließend wird der Aufbau und die Funktionsweise eines globalen Satelliten Navigationssystems anhand des NAVSTAR-GPS grob erläutert, sowie dessen Probleme und Fehlerquellen aufgewiesen.

Das nächste Kapitel befasst sich mit einigen Hardware-Komponenten, die zur Realisierung des Projektes verwendet wurden. Neben dem Funkmodul *RFM12* und dem GPS-Modul *HI-204II* ist die Hauptkomponente der Mikrocontroller *ATmega168*. Es wird ein kurzer Überblick über die Eigenschaften des Mikrocontrollers gegeben und die zum Einsatz gekommenen Features werden genauer erläutert.

Das vierte Kapitel soll eine kurze Übersicht über die verwendete Software geben. Hierzu zählen neben der Software, die bei der Programmierung des Mikrocontrollers zum Einsatz kam, auch zwei Programme, die zur Datenakquise genutzt wurden.

Kapitel fünf ist in zwei Abschnitte aufgeteilt. Auf der einen Seite steht die Umsetzung des Projekts im Bereich der Hardware. Hier wird auf den generellen Anschluss der einzelnen Komponenten sowie auf die Anbindung der einzelnen Module an den Mikrocontroller auf technischer Ebene eingegangen. Der zweite Abschnitt des Kapitels befasst sich mit der für dieses Projekt entwickelten Software. Die einzelnen Funktionen und Variablen werden systematisch anhand ihrer Dateizugehörigkeit in ihrer Funktion, Implementation und ihrer Schnittstellen erläutert.

Im letzten Kapitel werden die Ergebnisse der durchgeführten Messreihen zusammengetragen und ausgewertet. Zunächst wird jeder der sieben verschiedenen Lösungsansätze anhand einiger ausgewählter Messreihen bewertet und auf positive Aspekte und Probleme hingewiesen. Abschliessend wird eine Betrachtung über alle Lösungsansätze und Messreihen hinweg vorgenommen und zu einem Gesamtergebnis zusammengefasst.

Abstract

In a world which is stamped more and more by the application of technology, the systems which can take over duties independently in a given context win more and more in meaning. Such systems are called autonomous systems which contain an independent behaviour and interactions with their environment and other systems. Autonomous mobile robots also count to these systems. They are, so to speak, intelligent vehicles which can carry out transferred duties without human support. To perceive their environment, to be able to navigate and interoperate within, these systems need different kinds of sensors. Therefore the robotics research is very interdisciplinary.

This work deals exclusively with the area of the navigation, in particular with the position regulation which forms the basis for a good navigation. A world-wide available system often used nowadays for positioning GPS. The aim of this work is the connection of a GPS module with a microcontroller and the attempt to precise the exactness of the positioning locally, in order to open the possibility for such an autonomous system to manage duties which require a higher exactness.

The system to be developed should be put together from very reasonable components and should have the same interface like the GPS module. The approach forming the basis of this work exists of the inquiry of correction data by a base station, transmission of these data via radio-frequency and issue of NMEA correspondent records with a serial interface.

In the second chapter of this work the main focus lies on the satellite navigation. At first a look at the history of navigation from it's beginnings up until today's satellite navigation. Afterwards the construction and the functionality of a global satellite navigation system is roughly explained with the help of the NAVSTAR GPS, as well it's his problems and sources of error.

The next chapter deals with some hardware components which were used for the realisation of the project. Beside the radio module to *RFM12* and the GPS module *HI-204II* the main component is the microcontroller *ATmega168*. A short overview about the qualities of the microcontroller is given and the features being used are explained more in depth.

The fourth chapter gives a short overview of the used software. Beside the software which was used programming the microcontroller, two other programmes were used for data acquiring.

Chapter five is split in two segments. On one hand you have the realisation of the hardware part. Here the general connection of the single components as well as the binding of the single modules to the microcontroller are explained on a technical level. The second part of the chapter deals with the software developed for this project. The single functions and variables are explained systematically with the help of their file affiliation, function, implementation and interfaces.

In the last chapter the results of the carried out series of measurement are gathered and evaluated. First each of the seven different solution attempts are evaluated with the help of some selected series of measurement and positive aspects and problems are pointed out. Finally a look at all solution attempts and series of measurement is taken and combined to an overall result.

2 Navigation und GPS

2.1 Geschichte der Navigation

Seit die Menschen auf unserem Planeten Reisen unternehmen (was bedeutet, dass sie irgendwann wieder zu ihrem Ausgangspunkt zurück gelangen wollen), ist es für sie bedeutsam den Ausgangspunkt und ihre aktuelle Position zu kennen.

Zur Navigation allgemein gehören drei verschiedene Teilbereiche. Der erste Bereich ist die Ortsbestimmung im Raum. Diese bezieht sich in dieser Arbeit insbesondere auf die Bestimmung der geographischen Position mittels eines Satellitenortungssystems. Der zweite Bereich befasst sich mit der Ermittlung des optimalen Weges, der zum Zielort führt. Dieser Teilbereich der Navigation wird in dieser Arbeit nicht behandelt. Er umfasst Themen wie Routenberechnung und Hinderniserkennung. Der letzte Aspekt ist das Führen bzw. Steuern des Fahrzeuges, welcher ebenfalls nicht unmittelbar mit dieser Arbeit im Zusammenhang steht.

Der Bereich mit der längsten Historie ist der der Ortsbestimmung. Im frühen Ägypten und Indien wurden Landmarken und Küstenlinien zu Orientierung genutzt. Das erste Hilfsmittel, das zur globalen Ortsbestimmung von den Menschen verwendet wurde, ist der Sternenhimmel. Die astronomische Ortsbestimmung wurde durch Höhenwinkel-Messung zur Sonne, den Fixsternen oder den Planeten durchgeführt. Die Genauigkeit der Ortsbestimmung mit dem Jakobsstab liegt bei etwa 100 km, mit modernen Sextanten konnte die Präzision auf ungefähr 2 km gesteigert werden. Die Problematik bei dieser Art der Ortsbestimmung ist die Notwendigkeit der freien Sicht auf die Gestirne, welche bei bedecktem Himmel nicht gegeben ist.

Das nächste Hilfsmittel, das Verwendung fand, war der Magnetkompass. Dieser wird schon seit über 1000 Jahren eingesetzt [1] und ist unabhängig von Tageszeit und Wetter. Sowohl in der griechischen Antike als auch zur gleichen Zeit in

[1] schriftlich erwähnt wurde der sogenannte „nasse Kompass“ , ein in Wasser befindlicher Splitter eines Magnetsteins, erstmals im 11. Jahrhundert

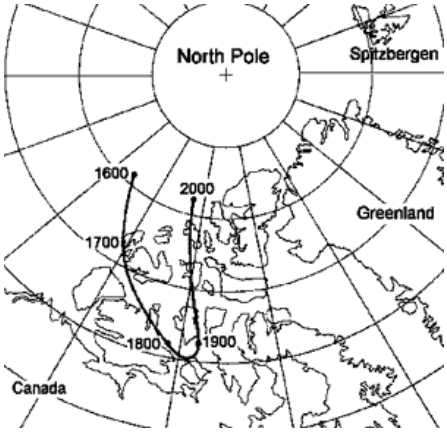


Abbildung 2.1: Langfristige Wanderung des magnetischen Nordpols

Quelle: http://www.kowoma.de/gps/zusatzerklaerungen/nordpolwander.gif

China war bekannt, dass sich Splitter von Magnetsteinen immer in die gleiche Richtung auspendeln. Die Ausrichtung erfolgt nach dem Magnetfeld der Erde. Die magnetischen Pole stimmen allerdings nicht mit den geographischen Polen der Erde überein. So befindet sich zum Beispiel der magnetische Nordpol zur Zeit einige Kilometer nördlich der Nordpolarmeerinseln Kanadas, etwa 2000 km vom geographischen Nordpol entfernt. Das langfristige Wandern des Pols (vgl. hierzu Abb. 2.1) hängt mit den geologischen Aktivitäten im Erdinneren zusammen, während die täglichen Schwankungen, welche ungefähr 80 km betragen, von der variierenden Sonneneinstrahlung abhängen. Die Abweichung zwischen geographischer und magnetischer Nordrichtung wird Deklination genannt. Die ersten Kompanten hatten damals eine Einteilung von 16 bis hin zu 68 Strichen, während man heute im Allgemeinen mit einer Einteilung von 360° arbeitet.

Abbildung 2.2: Hadley-Sextant

Quelle: http://www.planet-schule.de/sf/multimedia/animationen/navigation/sextant.jpg

Um noch besser navigieren zu können, war es von Vorteil seine genaue Position zu kennen. Mit Hilfe des von John Hadley entwickelten Sextanten (Abb. 2.2) ließ sich die Bestimmung des Breitengrades bis auf wenige Kilometer genau

ermitteln. Das Problem bei der Positionsbestimmung bei längeren Reisen lag in der Ermittlung des aktuellen Längengrades. Nach verschiedenen Methoden und Lösungsansätzen setzte sich im Jahr 1735 ein Verfahren durch, welches auf Zeitmessung basiert. Grund hierfür war die Erstellung eines relativ genauen Chronometers durch den Briten John Harrison. Der Chronometer wurde am Anfang einer langen Schiffsreise auf die Sonnenzeit des Greenwich-Meridians eingestellt. Aus dem Zeitunterschied zwischen der angezeigten Zeit und der durch Peilung von Sonne oder Gestirnen errechneten Ortszeit, ließ sich der Längengrad für die damalige Zeit hinreichend genau ermitteln.

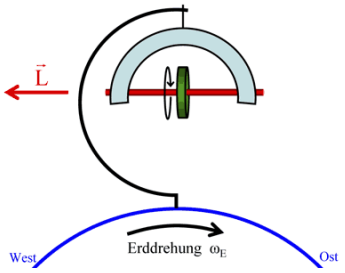


Abbildung 2.3: Funktionsprinzip eines Gyroskops

Quelle: http://pgd5.physik.hu-berlin.de/struma/klame49.gif

Seit dem Jahre 1851 gibt es ein weiteres Instrument zur Navigation: den Kreiselkompass[2]. Er richtet sich, wie der „normale“ Kompass, ebenfalls Richtung in Norden aus, ist aber im Gegensatz zu diesem unabhängig vom Magnetfeld der Erde. Während der „normale“ Kompass Richtung magnetischem Nordpol zeigt, richtet sich der Kreiselkompass zum geographischen Nordpol hin aus. Dies ist der Schnittpunkt der Erdachse mit der Erdoberfläche in der nördlichen Hemisphäre. Zum Ausrichten gen Norden nutzt der Kreiselkompass die Rotation der Erde. Die axialsymmetrische Kreiselscheibe ist so gelagert, dass sich die Drehachse parallel zur Erdoberfläche ausrichten kann, und somit in eine Horizontalebene gezwungen wird. Die Kreiselscheibe dreht sich, angetrieben von einem Elektromotor, mit bis zu 40.000 Umdrehungen pro Minute. Aufgrund der Rotationsgesetze der Physik[3] ist die Drehachse des Kreisels bemüht sich parallel zur Erdrotationsachse auzurichten. Durch die Erddrehung wird die Lage der Horizontalebene im Raum verändert, während der Kreisel versucht die Lage seiner Drehachse unverändert zu lassen. Da aber die Drehachse in der Horizontalebene gehalten wird, zum Beispiel durch Schwimmer in einer Flüssigkeit oder durch Pendelgewichte, kann sich die Drehachse nur Richtung geographischem Nordpol

[2] auch Gyroskop genannt
[3] auch Kreiselgesetze genannt, siehe [BS98]:348

oder Südpol ausrichten[4]. Der Kreiselkompass wird zur Navigation eingesetzt, aber auch als „künstlicher Horizont“ in der Flugzeugtechnik.

Durch die rasante Entwicklung in der Technik ergaben sich auch neue Möglichkeiten um Sensoren, welche für die Navigation genutzt werden können, herzustellen. Durch die neuen Erkenntnisse besonders in der Physik, der Schwingungsmechanik und der Elektrotechnik, war die Herstellung von Inertialsensoren möglich. Inertialsensoren dienen zur Messung linearer Beschleunigungs- und Rotationskräfte. Eine inertiale Messeinheit (Inertial Measurement Unit, kurz IMU) besteht aus mehreren Inertialsensoren und kann die Beschleunigung in sechs Freiheitsgraden ermitteln. Führend bei der Entwicklung der Grundlagen zur Trägheitsnavigation war der deutsche Forscher Siegfried Reisch.

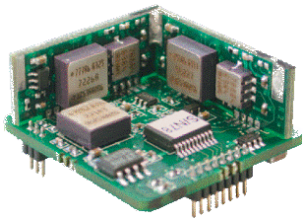


Abbildung 2.4: Inertialsensoren, Inertial Measurement Unit (IMU)

Quelle: http://www.barnardmicrosystems.com/clip_image103.gif

Mit der Entwicklung der Elektronik und des Funks war es möglich eine genauere Positionsbestimmung durchzuführen. Das Prinzip, welches der Ortsbestimmung zugrunde liegt, wird als Hyperbelnavigation, Funknavigation oder auch Radionavigation bezeichnet. Zu dieser Art der Navigation werden sogenannte Funkketten benötigt, die jeweils aus einer Master-Station und zwei bis fünf weiteren Stationen in einigen 100 km Entfernung bestehen. Eine Kette sendet, ausgehend von der Master-Station, synchronisierte Funksignale aus. Im Empfängergerät wird dann unter Nutzung der zeitlichen Differenz mit der die Signale ankommen, die Position berechnet. Zu den bekanntesten Systemen der Radionavigation zählen *OMEGA* (OMnidirectional Electronic Ground Antennas), *NDB* (Non Directional Beacons), *LORAN-C* (Long Range Area Navigation) und die militärische Variante *LORAN-D*. Das *LORAN-C* System wird seit 1957/58 in den USA von der Küstenwache (US Coast Guard) eingesetzt. Auch heute wird *LORAN-C* noch in weiten Teilen der Welt genutzt, da es auf Grund seiner langwelligen Signalcharakteristik auch Orte erreicht, die durch das moderne GPS nicht erreicht werden. *LORAN-C* verwendet eine Trägerfrequenz von 100kHz und hat damit eine Reichweite von etwas über 1000 km. Die

[4] die Ausrichtung hängt von der Rotationsrichtung der Drehachse des Kreisels ab

Trägerfrequenz und die damit verbundene Reichweite sind aber auch genau das Problem dieser Technik. Benutzt man eine langwelligere Frequenz, so ist zwar eine Positionsbestimmung in einer größeren Entfernung möglich; diese ist aber wesentlich ungenauer. Benutzt man hingegen eine höhere Frequenz, erhöht sich zwar die Genauigkeit der Positionsbestimmung, jedoch nimmt die Reichweite des Signals erheblich ab. Damit müsste mit entsprechenden Mehrkosten die Anzahl der Funkketten erhöht werden. Die Lösung dieses Dilemmas führt uns zum Hauptthema dieses Kapitels: das Global Positioning System. Durch Verlagerung der Sender von der Erdoberfläche in Satelliten kann auch mit hochfrequenten Signalen eine recht gute Flächenabdeckung erzielt werden. Das Prinzip beider Verfahren ist sich allerdings recht ähnlich.

Eine kurze Übersicht über die zeitliche Entwicklung der Navigation ist durch die Timeline auf Seite 10 dargestellt. Sie zeigt die wichtigsten Stationen und erstreckt sich von 4000 v.Chr. mit der ersten Navigation bis hin zu den neuesten Navigations- und Informationsgeräten, basierend auf einer Positionsbestimmung mittels GPS.

Koppel- und Astro-Navigation

In Indien und Ägypten wird die Navigation zuerst entwickelt. Landmarken und Küstenlinien werden mit Parametern wie Strömung, Tiefen, Windeschwindigkeit und Sonnenstand kombiniert. **(4000 v.Chr.)**

Navigation nach Gestirnen

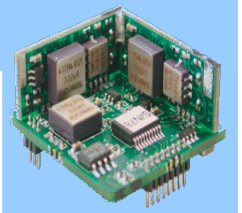
Die Bestimmung des Breitengrades ist mit Astrolab und Jakobsstab recht gut möglich. Die Position in Ost-West-Richtung (Längengrad) kann jedoch nur geschätzt werden

Oktant

John Hadley, englischer Astronom und Mathematiker, erfindet den Oktanten, der als direkter Vorläufer des Sextanten gilt. Die Bestimmung des Breitengrades ist damit auf wenige Kilometer genau möglich. **(1731)**

Trägheitsnavigation

Der deutsche Forscher Siegfried Reisch entwickelt die Grundlagen der Trägheitsnavigation, die aus der Beschleunigung die relative Ortsänderung errechnet. **(1941)**



Mobile Navigationsgeräte

Nach GPS-Mäusen für Notebooks und PDA's kommen endlich Geräte auf den Markt, die ausschließlich Navigationszwecken dienen. **(2005)**

GPS-Navigation im Auto

Mitsubishi und Pioneer stellen unabhängig voneinander das jeweils „weltweit erste“ GPS-Navi für's Auto vor. **(1990)**

Location Base Services

Die reine Positionsbestimmung ist Nebensache. Wichtiger ist die Ergänzung dieser Informationen um aktuelle Verkehrsdaten – z.B. TMC – sowie der Verknüpfung des Standortes und der Fahrtrichtung mit aktuellen Angeboten und Informationen aus Kultur, Kunst und Kommerz. **(2008)**

4000v.Chr. 1000n.Chr. 1100 1200 1300 1400 1420 1500 1600 1700 1731 1735 1800 1851 1900 1941 1978 1990 2001 2005 2006 2007 2008

Erster Europäer in Nordamerika

Der Isländer Leif Erikson segelt als erster Europäer nach Nordamerika – per Sternen-Navigation. **(1000 n.Chr.)**

Kompass

In Europa tauchen die ersten Komponenten auf. Es ist ungeklärt, ob dies eine eigene Erfindung ist oder ob die Technik aus China importiert wurde. **(ca. 1200)**

Navigation wird zur Wissenschaft

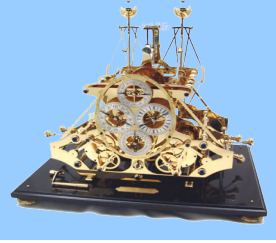
Heinrich dem Seefahrer, Mitglied der portugiesischen Königsfamilie, wird die Gründung der Escola Nautica, der ersten Seefahrer-Schule in Europa, zugeschrieben. **(1420)**

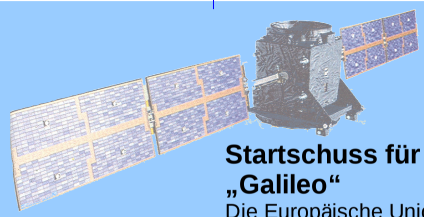
Kreiselkompass

Der Kreiselkompass oder auch Gyroskop genannt, richtet sich unabhängig vom Magnetfeld der Erde in Nord-Südrichtung aus. **(1851)**

Chronometer

Der Brite John Harrison baut die erste von vier hochgenauen Uhren, die die Zeit des Heimathafens anzeigen und über die Zeitverschiebung den Längengrad bestimmen können. **(1735)**





Startschuss für „Galileo“

Die Europäische Union plant ein eigenes Satelliten-Navigationssystem das 2013 fertig sein soll. Das „Galileo“-System wird weitgehend kompatibel zur „NAVSTAR“ sein. **(2001)**

Erster GPS-Satellit gestartet

Als Militärprojekt des US-Verteidigungs-ministeriums startet das „NAVSTAR“-Projekt, welches seit 1983 auch für zivile Zwecke nutzbar ist. **(1978)**

Navigation mit dem Handy

Mit dem N95 von Nokia ist die Navigation auch auf dem Mobiltelefon angekommen. Zwar gab es bereits vorher Handys mit GPS-Funktion – den Durchbruch schaffte allerdings erst Nokia. **(2007)**

2.2 Global Positioning System

Ein Global Positioning System, kurz GPS, bezeichnet allgemein ein System, meist satellitengestützt, das zur weltweiten Positionsbestimmung und darüber hinaus zur Navigation verwendet werden kann. Wie bei vielen technologischen Entwicklungen war auch hier das Militär die treibende Kraft. Bevor allerdings auf die Geschichte, den Aufbau und auf die etwas komplizierteren Berechnungsalgorithmen eingegangen wird, soll in einer kleinen Einleitung erst einmal die Idee, das Grundprinzip, eines heutigen *GNSS*[5] erläutert werden.

Wie allgemein bekannt, basiert GPS auf Satellitentechnologie. Das ist soweit auch richtig, allerdings gibt es hierzu bereits die ersten falschen Vorstellungen. Eine der häufigsten Annahmen, welche ich auch schon in einer „Technik"-Sendung im Fernsehen gehört habe, ist, dass es im Weltraum einen Satelliten gibt, der meine Position ortet und mir dann meine aktuelle Position zuschickt. Mit etwas technischem Verständnis leuchtet jedoch ein, dass unser mobiles GPS-Gerät dann sowohl einen Empfänger, als auch einen Sender, welcher unser Ortungssignal bis in den Orbit sendet, enthalten müsste. Dadurch würden diese Geräte allerdings wesentlich größer werden als es die heutigen Modelle sind. Auf der anderen Seite würde dies bedeuten, dass der Satellit bei der immer weiter steigenden Anzahl an GPS-Geräten mit der Ortung aller GPS-Geräte und dem „Zurücksenden" der Positionsdaten sehr viel zu tun hätte. Hier entstünde dann schon das nächste Problem: wie bekäme unser Empfänger seine Positionsdaten? Über eine eigene Frequenz? Über eine Frequenz mit einer Broadcast-Nachricht, die nur unser Gerät entschlüsseln könnte? Dann müsste der Satellit für jedes GPS-Gerät einen eigenen Code vorliegen haben. Wie wir sehen, kommen wir von einem Problem zum nächsten. So kommen wir also nicht weiter.

Die Grundidee bei der Berechnung der Position mittels GPS ist es, die Entfernung zu mehreren Referenzpositionen zu kennen und durch Bilden von Entfernungskreisen um diese Referenzpunkte den Schnittpunkt, bzw. die aktuelle Position, zu ermitteln. Um dieses Verfahren besser zu verstehen, soll an dieser Stelle ein kleines Beispiel (vergleiche hierzu Abb. 2.6) im Zweidimensionalen dienen.

[5] Global Navigation Satellite System

Bekannt seien die Punkte A, B und C mit ihren jeweiligen Koordinaten in einem festgelegten Referenzsystem, sowie die Entfernungen des gesuchten Punktes zu den einzelen Punkten:

- Punkt A (4|3) mit einer Entfernung von 3 LE[6] zu dem gesuchten Punkt
- Punkt B (11|6) mit einer Entfernung von 5 LE zu dem gesuchten Punkt
- Punkt C (7|7) mit einer Entfernung von 4 LE zu dem gesuchten Punkt

Konstruiert man nun um die drei Punkte einen Kreis mit dem Radius der Entfernung, so schneiden sich die Kreise alle in einem Punkt (Trilateration[7]). Dieser Punkt ist die gesuchte Position S mit den ungefähren Koordinaten (7|3) im Referenzsystem. Die bekannten Punkte stellen die Satelliten dar, die Nachrichten in Form von „Ich bin Satellit X, befinde mich an der Position Y und die aktuelle Zeit ist Z“ versenden.

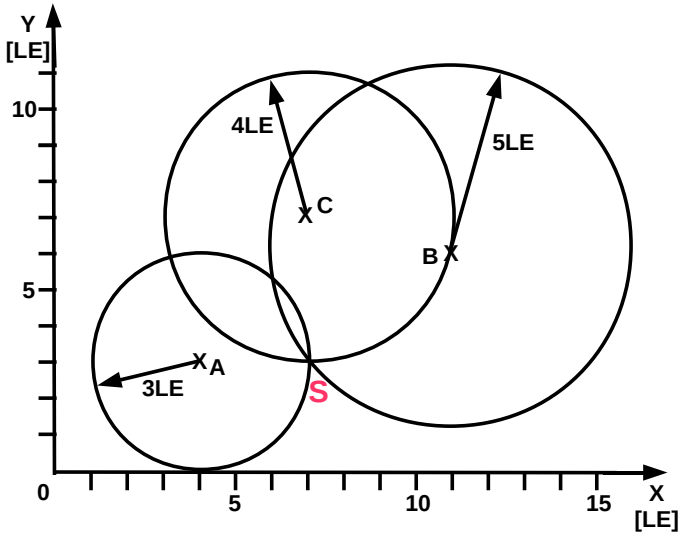


Abbildung 2.6: Beispiel zur 2D-Positionsbestimmung mittels Trilateration

Um die eben erwähnte Problematik der Überlastung auf seiten der Satelliten bei der Berechnung zu vermeiden, wird die Bestimmung der eigenen aktuellen Position in die einzelnen mobilen GPS-Geräte verlagert. Durch die immer weiter steigende Leistung und Integration von Schaltkreisen ist die notwendige Rechenleistung und gleichzeitige Kompaktheit heutzutage kein Problem mehr. Wo aber kommen die zur Berechnung notwendigen Informationen her? Einen Teil der Informationen bekommt unser GPS-Gerät direkt aus dem Weltraum

[6] LE ist eine Längeneinheit in unserem Referenzsystem
[7] Entfernungsmessung von drei Punkten aus

geschickt. Allerdings genügen zur Berechnung der aktuellen eigenen Position, wie in unserem kleinen Beispiel zu sehen war, nicht nur die Informationen von einem Satellit, sondern unser GPS-Gerät muss von mehreren Satelliten gleichzeitig die einzelnen Signale empfangen. Die restlichen Informationen werden aus Messungen im GPS-Gerät selbst gewonnen. Die wichtigste Information, die ermittelt wird, ist die Signallaufzeit vom Satellit zum GPS-Empfänger. Aus dieser kann dann näherungsweise die Entfernung zu diesem Satelliten berechnet werden. Welche Werte gemessen und welche Informationen genau daraus gewonnen werden, wird in den Kapiteln 2.2.3 bis 2.2.5 erläutert.

Wie im vorherigen Absatz angesprochen, muss ein GPS-Empfänger Informationen von mehreren Satelliten empfangen, um die eigene aktuelle Position zu berechnen. Mit Hilfe des Codemultiplexverfahrens (CDMA)[8] wird erreicht, dass alle Satelliten über die gleiche Frequenz ihre Informationen versenden können, und der GPS-Empfänger trotzdem die verschiedenen Informationen der einzelnen Satelliten zeitgleich empfangen, zuordnen und auswerten kann. Für die zivile Nutzung von GPS wird die Frequenz L1=1575,42 MHz genutzt. Der sogenannte Spreizcode, der auf diese Frequenz aufmoduliert wird, ist der C/A-Code[9]. Des Weiteren senden die Satelliten auf der Frequenz L2=1227,60 MHz die gleichen Informationen aus, welche allerdings für militärische Zwecke vorgesehen sind. Worin genau der Unterschied zwischen den beiden Signalen besteht und wie genau die Signale aussehen wird im Kapitel 2.2.3.1 eingehend behandelt.

Die von GPS verwendeten elektromagnetischen Schwingungen mit den Frequenzen L1 und L2 haben ähnliche Ausbreitungseigenschaften wie das sichtbare Licht. Das bedeutet, dass um einen guten GPS-Empfang zu bekommen, man möglichst freie Sicht zum Himmel haben sollte, da die Signale weder Stein noch Wasser noch sonstige Materialien gut durchdringen[10]. Ältere GPS-Empfänger hatten sogar bereits bei sehr dichter Belaubung in Wäldern Probleme bei der Ortung. Heutige Geräte haben eine verbesserte Empfangstechnik, die jedoch in Tunneln, Tiefgaragen und Gebäuden auch keine robuste Ortung ermöglicht. Um eine möglichst genaue Positionsbestimmung durchführen zu können sollten die sichtbaren Satelliten eine möglichst gleichmäßig verteilte Konstellation aufweisen, sowie einen minimalen Erhebungswinkel über dem Horizont besitzen[11].

[8] Code Division Multiple Access, siehe [Leh03]:21
[9] Corase/Aquisition, weiteres in Kapitel 2.2.3.1
[10] auch aufgrund einer relativ kleinen Sendeleistung der Satelliten
[11] siehe Abschnitt 2.2.7

2.2.1 Geschichte

Wie bereits am Anfang von Kapitel 2.2 erwähnt, liegt der Ursprung für GPS im militärischen Bereich. Das erste Satellitennavigationssystem hieß *Transit* und wurde nach sechs Jahren Entwicklung erstmals im Jahre 1958 von der US-Marine in Betrieb genommen. Das *Transit*-System diente ursprünglich zur Zielführung von Raketen und erreichte eine Genauigkeit von bis zu 15 Metern. Zwischen den Jahren 1967 und 1996 konnte das System auch für zivile Zwecke genutzt werden. Da es aber nur vier *Transit*-Satelliten in der Umlaufbahn gab, konnte eine Ortung nur alle 30 bis 110 Minuten durchgeführt werden. Mit diesen Zeitabständen konnte natürlich keine global nutzbare Navigation stattfinden.

Auch in der ehemaligen Sowjetunion wurde in Richtung Funknavigationssysteme geforscht. Seit 1960 wurden von russischer Seite vier verschiedene Satellitentypen in die Umlaufbahn gebracht:

- *Tsiklon*
- *Parus*
- *Tsikada*
- *Uragan*

Die Satelliten des Typs *Tsiklon* und *Parus* standen lediglich für militärische Zwecke zur Verfügung. Nach Inbetriebnahme der *Tsikada*-Satelliten wurde das System auch für die Navigation der russischen Handelsflotte genutzt und unter dem Namen *Tsikada* zusammengefasst.

Nach der Entspannungspolitik in den frühen 70er Jahren begann ein Rüstungswettlauf zwischen den USA und der ehemaligen Sowjetunion. Somit wurde das Verlangen nach neuen Systemen, die mit einer höheren Präzision global nutzbar sind, immer größer.

Der Nachfolger des amerikanischen *Transit*-Systems ist das *NAVSTAR-GPS*[12]. Es ist das zur Zeit weltweit meistgenutzte und leistungsfähigste *GNSS*, welches sowohl militärisch als auch zivil genutzt wird. Dieses System wird, wie auch der Vorgänger, vom US-Militär (genauer: dem *Departement of Defense*) verwaltet und betrieben. Es untersteht dem *NAVSTAR-GPS Joint Program Office* des *Space and Missile Systems Center* der *Los Angeles Air Force Base* in Los Angeles, Kalifornien. Hier werden die Entscheidungen über den aktuellen Betrieb

[12] Navigational Satellite Timing and Ranging - Global Positioning System

und die Zukunft des Systems gefällt. Des Weiteren werden hier die Satelliten sowie spezielle Empfänger für das US-Militär gefertigt.

Neben dem Ortungs- und Navigationssystem der zweiten Generation der USA gab es natürlich auch ein entsprechendes System der damaligen UdSSR[13]. Dieses System ist unter dem Namen *GLONASS*[14] bekannt. Es ähnelt im Aufbau stark dem GPS-System des US-Militärs. Die Anzahl der im Orbit befindlichen Satelliten schwankte in den letzen Jahren erheblich. Nach dem Zerfall der Sowjetunion sank die Anzahl zuerst drastisch, da die *Uragan*-Satelliten nur eine Lebensdauer von rund 3 Jahren besaßen. Seit 2001 jedoch werden wieder weiterentwickelte Satelliten in die Umlaufbahn gebracht. Sie können theoretisch genauso wie das „normale“ GPS genutzt werden, weswegen auch eine Zusammenarbeit mit dem *GALILEO*-Projekt in Planung ist (vgl. Kapitel 2.3).

Im Folgenden werde ich mein Hauptaugenmerk auf das amerikanische *NAVSTAR-GPS* legen, welches heutzutage im allgemeinen Sprachgebrauch als „GPS“ bezeichnet wird. Diese weitverbreitete Kurzform werde ich für den Rest dieser Arbeit übernehmen, was es für Leser, die sich nur mit einzelnen Abschnitten befassen, etwas einfacher machen soll. In den nächsten drei Abschnitten werden die drei grundlegenden Bestandteile der GPS-Topologie,

- das Raumsegment
- das Kontrollsegment
- das Nutzersegment

in die GPS üblicherweise unterteilt wird, genauer betrachtet.

2.2.2 Aufbau

2.2.2.1 Das Raumsegment

Das *NAVSTAR-GPS* ist ein Global Navigation Satellite System (GNSS). Dies bedeutet, dass Positionsbestimmung und Navigation mittels Funksignalen von Satelliten aus realisiert wird. Die Satelliten, die für dieses System verwendet werden, befinden sich im sogenannten *Medium Earth Orbit* (MEO). Der *Medium Earth Orbit* befindet sich zwischen dem *Low Earth Orbit* (LEO) und dem geostationären Orbit (GEO) und umfasst alle Umlaufbahnen zwischen 1.000

[13] Union der Sozialistischen Sowjetrepubliken, Sowjetunion
[14] Глобалнaja Нащигазионнaja Спутникощaja Система für Globalnaja Nawigazionnaja Sputnikowaja Sistema (GLObales NAvigations-Satelliten-System)

und 36.000 km Höhe. Die Flughöhe (in der Fachsprache Bahnhöhe genannt) der GPS-Satelliten beträgt in etwa 20200 km. Es wurde diese "geringe" Bahnhöhe[15] gewählt, um die Empfangsgerätegröße für die Positionsbestimmung möglichst klein zu halten. Aufgrund der geringen Bahnhöhe ist die Bahngeschwindigkeit dementsprechend hoch. Um nun sicher zu stellen, dass zu jeder Zeit das Signal von mindestens drei bis vier Satelliten (warum diese Anzahl nötig ist siehe Abschnitt 2.2.5) zu empfangen ist, sind mindestens 24 Satelliten notwendig.

Die Satelliten bewegen sich auf einer Umlaufbahn mit einer Inklination von 55°, was bedeutet, dass die Umlaufbahn zur Äquatorebene um 55° geneigt ist (siehe Abb. 2.7). Somit bewegen sich die Satelliten zwischen 55° nördlicher und 55° südlicher Breite (vergleiche hierzu Groundtrack[16] in Abb. 2.8). Durch die Inklination wird vermieden, dass sich zu viele Satelliten über den Polregionen befinden, bzw. diese Gebiete mit ihrer Signalabdeckung versorgen. Durch die große Flächenabdeckung der einzelnen Satelliten ist allerdings auch in den Polregionen der GPS-Empfang gesichert.

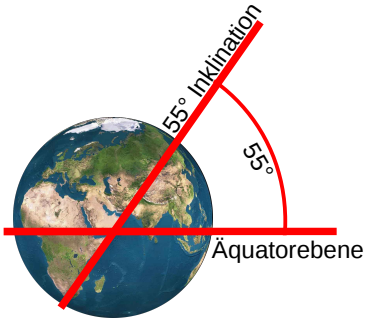


Abbildung 2.7: Inklination der GPS-Satelliten

Zur Zeit befinden sich 32 GPS-Satelliten im Orbit[17], von denen einer außer Betrieb ist. Die GPS-Satelliten umkreisen die Erde auf sechs verschiedenen Bahnebenen (Orbits). Der Überflugpunkt der einzelnen Bahnebenen über den Äquator ist um jeweils 60 Längengrade verschoben. Durch die Anordnung ist eine gleichmäßige Verteilung gewährleistet und sie ermöglicht unter guten Voraussetzungen einen Empfang von acht bis zehn Satellitensignalen. Jeder Satellit umkreist die Erde mit einer Geschwindigkeit von $3,9\frac{km}{s}$, was zu einer Erdumlaufzeit von 11 Stunden und 58 Minuten führt. Somit überfliegt der gleiche Satellit jeden Tag bei seinem zweiten Umlauf die gleiche Position um 4 Minuten früher als am vorhergedenden Tag.

[15] im vergleich dazu beträgt die Bahnhöhe des geostationären TV-Satelitten ASTRA ca. 36.000 km, siehe [BP07]
[16] Als Groundtrack bezeichnet man den Weg eines Satelliten über der Erdoberfläche
[17] aktuelle Informationen zu den GPS-Satelliten unter ftp://tycho.usno.navy.mil/pub/gps/gpstd.txt

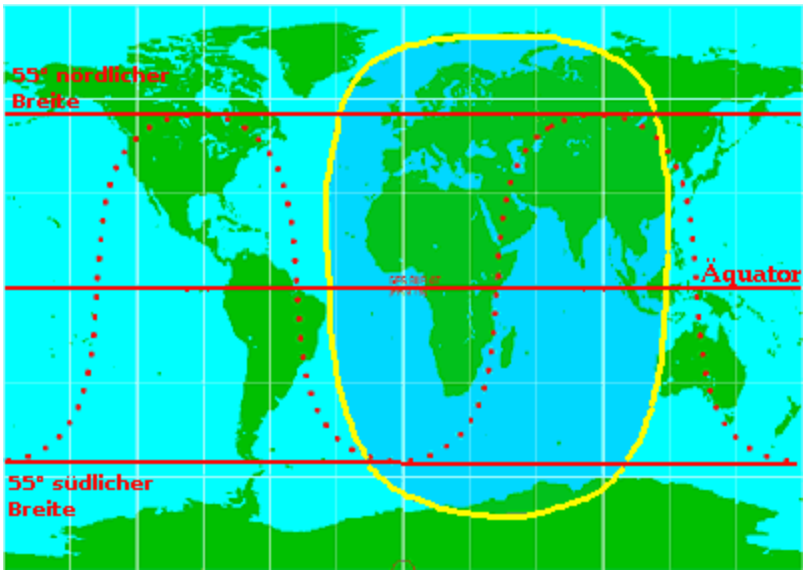


Abbildung 2.8: Groundtrack des GPS-Satelliten PRN18
vom 18.10.2001 0:00 Uhr bis 19.10.2001 0:00 Uhr

Quelle: http://www.kowoma.de/en/gps/groundtrack_prn18.gif

Zur Positionsbestimmung mittels GPS wird eine möglichst genaue Uhrzeit benötigt (warum dies so ist, vergleiche Kapitel 2.2.4.1). Deshalb befinden sich in den einzelnen GPS-Satelliten jeweils mindestens drei Atomuhren. Diese kontrollieren sich gegenseitig und die für das generierte Signal maßgebende Atomuhr wechselt von Zeit zu Zeit. Des Weiteren besitzen alle Satelliten Hilfsraketen, mit denen sie ihre Umlaufbahn gegebenenfalls korrigieren können, Solarpanels, die zur Stromerzeugung dienen, und Nickel-Cadmium-Zellen, welche Energie speichern, um den Satellit auch im Erdschatten mit Strom zu versorgen.

Es gibt mittlerweile eine Reihe verschiedener Typen von GPS-Satelliten, die im Laufe der Jahre immer weiter entwickelt worden sind:

- *Block I*
- *Block II*
- *Block IIA*
- *Block IIR*
- *Block IIR-M*
- *Block IIF* (ab 2010)
- *Block III* (ab 2013)

Block I-Satelliten wurden von 1978 bis 1985 von der *Vandenberg Air Force Base* in Kalifornien in die Umlaufbahn gebracht. Von den ursprünglich elf Satelliten ist heute jedoch kein einziger mehr in Betrieb, obwohl sie ihre geplante Lebensdauer von 4,5 Jahren weit überschritten (bis zu 13 Jahre Betrieb). Sie sollten eigentlich nur zum Test des Systems eingesetzt werden, wurden aber durch ihre unvorhergesehene längere Lebensdauer auch noch nach der kommerziellen Freigabe verwendet. Die Solarpanels der *Block I*-Satelliten hatten eine Leistung von ungefähr 400 Watt. Die Satelliten hatten ein Gewicht von ca. 845 kg und besaßen eine Sendeleistung von 50 Watt[18].

Ab dem Jahr 1989 wurden dann die neuen Satelliten vom Typ *Block II* vom Raketenstartgelände der U.S. Air Force am Cape Canaveral in Florida gestartet. Die *Block II*-Satelliten waren wesentlich größer als die alten *Block I*-Satelliten. Sie hatten eine Spannweite der Solarpanels von mehr als fünf Metern und wogen mit etwa 1,5 Tonnen etwas weniger als das Doppelte der *Block I*-Satelliten. Die Solarpanels der neuen Satelliten leisten allerdings mit rund 750 Watt auch fast das Doppelte der Panels, die bei *Block I*-Satelliten verwendet wurden. Ausgelegt sind die *Block II*-Satelliten für eine Betriebsdauer von ungefähr 7,5 Jahren, wobei viele Satelliten dieses Typs schon länger in Betrieb sind.

Es gibt fünf verschiedene Typen von *Block II*-Satelliten, die vom Grundaufbau jedoch alle gleich sind. In *Block II*- und *Block IIA*[19]-Satelliten sind jeweils zwei Rubidium- und zwei Cäsium-Atomuhren eingebaut, *Block IIR*-Satelliten besitzen drei Rubidium-Uhren. Diese Uhren besitzen eine Uhrenstabilität von mindestens 10^{-13}, dies bedeutet, dass die Uhr in einer Million Jahren maximal eine Abweichung von einer Sekunde besitzt. Ab den Satelliten des Typs *Block IIF* werden sogenannte Wasserstoff-Maser[20] als Frequenznormale für die Atomuhren verwendet, welche die Genauigkeit der Zeitmessung noch einmal steigerten. Die Frequenz der Atomuhren beträgt 10,23 MHz und ist die Grundfrequenz aller GPS-Satelliten.

Alle *Block II*-Satelliten senden ihre Signale auf zwei Frequenzen, der L1=1575,42 MHz und L2=1227,60 MHz. Ab den Satelliten des Typs *Block IIR-M* wird für die zivile Nutzung ein zweites Signal L2C[21], sowie noch ein weiteres Signal, das jedoch nur für militärische Zwecke zur Verfügung steht (L1M)[22], ausgesendet. Das L2C Signal kann auf einer Frequenz von 1227,5

[18] im Vergleich dazu besitzen Satelliten für Fernsehsignale eine Sendeleistung von über 100 Watt, siehe [BP07]:1042
[19] A für „advanced“
[20] Microwave Amplification by Stimulated Emission of Radiation
[21] C für Civilian (zu deutsch: zivil)
[22] M für Military (zu deutsch: militärisch)

MHz angesteuert werden; damit soll eine höhere Genauigkeit und durch eine erhöhte Sendeleistung auch ein besserer Empfang ermöglicht werden. Zur Zeit dient diese Frequenz allerdings nur Testzwecken, da es noch keine Endgeräte dafür gibt und sich bis jetzt auch nur drei Satelliten im Orbit befinden, die dieses Signal aussenden.

Die Satelliten des Typs *Block II* kosteten jeder bis zu 75 Millionen Dollar, zuzüglich der Kosten für das Installieren im Orbit von jeweils 50 Millionen Dollar. Neben den Geräten, die zum Betrieb von GPS benötigt werden, befinden sich noch andere Sensoren im Satelliten, mit denen zum Beispiel Atomexplosionen festgestellt werden können. Die Finanzierung wird von den USA übernommen, die ihrerseits aber von Herstellern entsprechender GPS-Empfänger Lizenzgebühren einfordern.

Gegenwärtig sind die Satelliten des Typs *Block IIF* im Bau, die noch eine weitere Frequenz L5 zur zivilen Nutzung senden sollen. Ab dem Jahr 2013 ist geplant Satelliten vom Typ *Block III* zu fertigen, die eventuell schon wieder auf die Aussendung des L1 und L2 Signals verzichten. Der aktuelle Zustand des Raumsegments von GPS, d.h. Anzahl, Typ und Status der Satelliten, ist unter folgender Internetadresse abrufbar: ftp://tycho.usno.navy.mil/pub/gps/gpstd.txt.

SVN	PRN	Raketenstart	in Betriebnahme	Status	aktuelle Atomuhr	Typ/Name
23	32	26.11.90	26.02.08	Unusable 16 Mar 20:17 UT to 17 Mar 04:00 UT for maintenance	RB	II-A
24	24	04.07.91	30.08.91	OK	CS	II-A
25	25	30.08.93	28.09.93	OK	RB	IIA
26	26	07.07.92	23.07.92	OK	RB	II-A
27	27	09.09.92	30.09.92	OK	RB	IIA
30	30	12.09.96	01.10.96	OK	CS	IIA
33	3	28.03.96	09.04.96	OK	CS	II-A
34	4	26.10.93	22.11.93	OK	RB	II-A
36	6	10.03.94	28.03.94	OK	RB	IIA
38	8	06.11.97	18.12.97	OK	RB	IIA
39	9	26.06.93	21.07.93	unusable 23 Mar 23:30 UT to 24 Mar 11:30 UT for maintenance	CS	IIA
40	10	16.07.96	15.08.96	OK	CS	IIA
41	14	10.11.00	10.12.00	OK	RB	II-R
43	13	23.07.97	31.01.98	OK	RB	II-R
44	28	16.07.00	17.08.00	OK	RB	IIR
45	21	31.03.03	12.04.03	OK	RB	II-R
46	11	07.10.99	03.01.00	OK	RB	II-R
47	22	21.12.03	12.01.04	unusable 25 Mar 13:00 UT to 26 Mar 01:00 UT for maintenance	RB	II-R
48	7	15.03.08	24.03.08	OK	RB	IIR-M
49	1	24.03.09	unusable until further notice	unusable until further notice	RB	IIR-M
50	5	17.08.09	27.08.09	OK	RB	IIR-M
51	20	11.05.00	01.06.00	OK	RB	II-R
52	31	25.09.06	12.10.06	OK	RB	IIR-M
53	17	26.09.05	16.12.05	OK	RB	IIR-M
54	18	30.01.01	15.02.01	OK	RB	II-R
55	15	17.10.07	31.10.07	OK	RB	IIR-M
56	16	29.01.03	18.02.03	OK	RB	IIR
57	29	20.12.07	02.01.08	OK	RB	IIR-M
58	12	17.11.06	13.12.06	unusable 30 Mar 05:00 UT to 17:00 UT for maintenance	RB	IIR-M
59	19	20.03.04	05.04.04	OK	RB	II-R
60	23	23.06.04	09.07.04	OK	RB	II-R
61	2	06.11.04	22.11.04	OK	RB	II-R

Tabelle 2.1: Die aktuell im Orbit befindlichen GPS Satelliten, Quelle: ftp://tycho.usno.navy.mil/pub/gps/gpstd.txt (20.03.2010)
SVN $\hat{=}$ Space Vehicle Number; PRN $\hat{=}$ Pseudo Random Noise

Abbildung 2.9: Atomuhr eines GPS-Statelliten
Quelle: http://www.kowoma.de/gps/Atomuhr.jpg

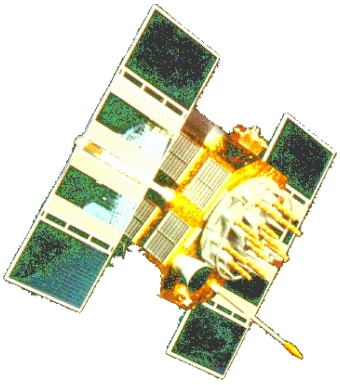


Abbildung 2.10: GPS-Satellit vom Typ Block II
Quelle: http://ivvgeo.uni-muenster.de /Vorlesung/GPS_Script/Bilder /Grundlagen/abb2_1_2.gif

2.2.2.2 Das Kontrollsegment

Das Kontrollsegment, oder auch Bodensegment, besteht aus mehreren Stationen, die über die ganze Erde verteilt sind und das *NAVSTAR-GPS* in seiner Funktionalität überwachen und steuern. Das Kontrollsegment besteht aus sechs Stationen, einer Master Control Station und fünf reinen Monitoring Stationen. Die Hauptaufgaben des Kontrollsegments bestehen in den folgenden vier Punkten:

- Kontrolle des Gesamtsystems
- Bestimmung der GPS-Systemzeit
- Vorausberechnung von Daten zur Steuerung der Satelliten
- Datenübermittlung

Die Master Control Station befindet sich auf der *Schriever Air Force Base* in Colorado Springs, Colorado, USA. Das hier stationierte „*50th Space Wing's 2nd Space Operations Squadron*“ ist für den operationalen Betrieb des *NAVSTAR-GPS* verantwortlich. Hier werden alle Informationen der anderen Stationen gesammelt und ausgewertet. Die Informationen über die Systemzeit der einzelnen Satelliten, sowie deren aktuelle Position und Flugbahn, werden permanent in Echtzeit ausgewertet und überwacht. Die wichtigsten Daten sind die Ephemeridendaten und die Atomuhrzeit der einzelnen Satelliten. Ephemeridendaten

sind im Allgemeinen eine tabellarische Auflistung von Positionsdaten eines sich bewegenden, astronomischen Objektes. Das sind, bei GPS die Aufzeichnungen über die Flugbahn der einzelnen GPS-Satelliten. Diese Daten werden in der Master Control Station verarbeitet um aus ihnen die erforderlichen Parameter zur Bahnkorrektur zu berechnen. Die Korrektur wird dann duch die Antriebsvorrichtungen an den Satelliten durchgeführt. Des Weiteren wird die vorraussichtliche Flugbahn der Satelliten berechnet. Diese Daten werden dann über die sogenannten Sendestationen an die GPS-Satelliten geschickt, welche diese wiederum über das normale GPS-Signal an die GPS-Empfänger weiter geben. Diese Informationen werden auch als „Broadcast Ephemeriden“ bezeichnet. Zusätzlich zu den bisher genannten Aufgaben verfügt die Master Control Station über die gleichen Funktionalitäten wie die Monitorstationen, auf die weiter unten noch genauer eingegangen wird.

Auch die offizielle GPS-Systemzeit wird in der Master Control Station festgelegt und von hier aus mit allen GPS-Satelliten synchronisiert. Außerdem kontrolliert die Master Control Station die Atomuhren der Satelliten und legt fest welche Atomuhr in den einzelnen Satellilten zum Generieren des GPS-Signals verwendet wird.

Neben der Master Control Station gibt es, wie schon am Anfang dieses Kapitels erwähnt, fünf Monitorstationen, die über den ganzen Globus verteilt sind (vergleiche hierzu Abbildung 2.11). Diese passiven Stationen sind eingentlich nur fest installierte GPS-Empfänger. Sie empfangen die Signale von allen „sichtbaren“ Satelliten und leiten diese an die Master Control Station in Colorado Springs weiter, wo sie dann weiterverarbeitet werden. Die fünf Monitorstationen in

- *Hawaii* (USA, Pazifik)
- *Ascension Islands* (Großbritannien, Süd-Atlantik)
- *Diego Garcia* (Großbritannien, Indischer Ozean)
- *Kwajalein* (Marshallinseln, Süd-Pazifik)
- *Cape Canaveral* (USA, Florida)

befinden sich alle auf Stützpunkten des US-Militärs. Durch ihre Verteilung ist gesichert, dass im Sendebereich jedes Satelliten mindestens eine Monitorstation liegt. Die Stützpunkte *Ascension Islands*, *Diego Garcia* und *Kwajalein* sind

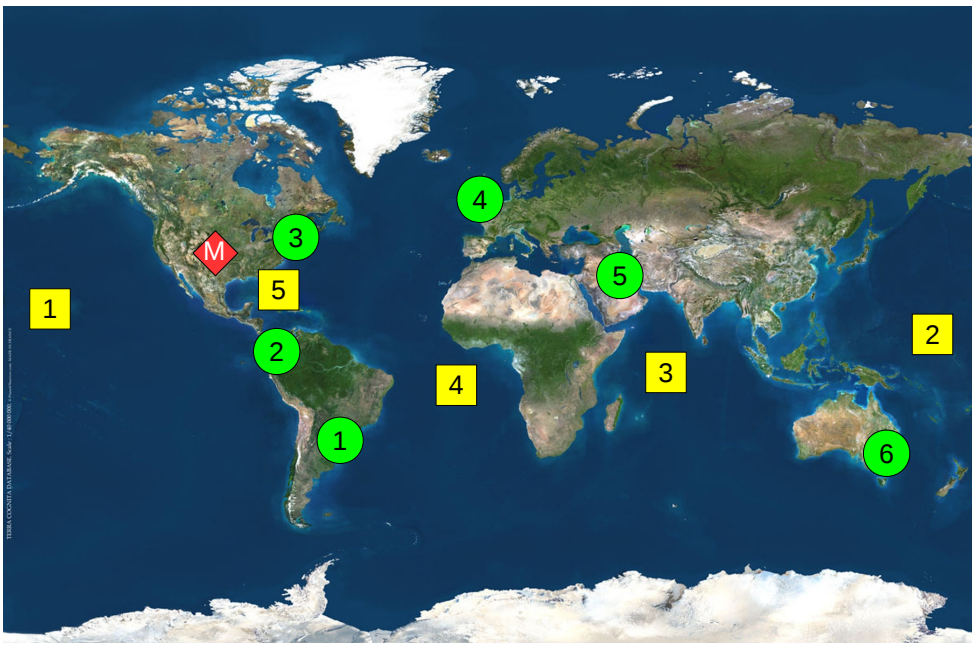


Abbildung 2.11: GPS-Kontrollstationen

Rot: M Master Station Colorado

Gelb: Monitorstation 1 Hawaii, 2 Kwajalein, 3 Diego Garcia, 4 Ascension Islands, 5 Cape Canaveral

Grün: NGA Monitorstation 1 Argentinien, 2 Ecquador, 3 Washington DC, 4 England, 5 Bahrain, 6 Australien

gleichzeitig auch noch Sendestationen. Sie besitzen zusätzlich noch Bodenantennen, mit denen sie Signale zu den Satelliten senden können. Diese Übertragung wird über ein sogennantes S-Band Signal vorgenommen, das eine Bandbreite von ca. 2000 MHz bis 4000 MHz besitzt. Im normalen Betrieb bekommen die Satelliten zweimal am Tag über dieses S-Band Daten zugeschickt. Die Satelliten ab dem Typ *Block IIR* können sogar direkt miteinander kommunizieren und sind dadurch in der Lage sich gegenseitig zu kontrollieren und zu korrigieren. Somit könnten sie im Extremfall bis zu 180 Tage ohne Kontakt zu einer Bodenstation ihre Funktionalität aufrecht erhalten.

Seit Ende des Jahres 2005 gibt es noch sechs weitere reine Monitorstationen. Diese gehören allerdings nicht zum eigentlichen *NAVSTAR-GPS*-Bodensegment, sondern zur *National Geospatial-Intelligence Agency* (kurz: NGA). Sie ist eine zentrale US-Behörde für militärische, geheimdienstliche und kommerzielle kartografische Auswertung und Aufklärung. Die Monitorstationen sind ebenfalls über den Globus verteilt (vergleiche hierzu Abbildung 2.11) und senden, wie auch die eigentlichen Monitorstationen, ihre empfangenen GPS-Daten an die Master Control Station nach Colorado. Durch diese zusätzlichen Standorte wird erreicht, dass im Sendebereich jedes GPS-Satelliten zwei Moni-

torstationen liegen, was zu einer höheren Präzision des Gesamtsystems führt. Für die Zunkunft hat die NGA den Bau von weiteren fünf Monitorstationen geplant, wodurch sich dann sogar mindestens drei Monitorstationen im Sendebereich eines jeden Satelliten befinden würde. Dadurch könnte die Präzison und Zuverlässigkeit des Systems noch weiter gesteigert werden.

Durch die permanente Kontrolle können Fehler frühzeitig erkannt und die entsprechenden Korrekturmaßnahmen durch die Master-Station umgehend eingeleitet werden. Alle essentiellen Aufgaben des GPS werden von der US-Armee durchgeführt, in deren Hand das GPS vollständig liegt. Deshalb befinden sich auch alle Bodenstationen, ausgenommen die Sationen der NGA natürlich, auf US-amerikanischen oder britischem Hoheitsgebiet.

2.2.2.3 Das Nutzersegment

Abbildung 2.12: verschiedene GPS-Geräte des Nutzersegmentes

Das dritte und letzte Segment der GPS-Topologie ist das Nutzersegment. Es umfasst alle Arten von GPS-Empfängern, also sowohl zivil als auch militärisch genutzte Geräte. Alle GPS-Empfänger sind reine Empfangsgeräte, die nur die Signale der GPS-Satelliten aufnehmen und auswerten. Somit ist das Nutzersegment eine passive Komponente des Systems.

Die GPS-Geräte des Nutzersegments bestehen im Allgemeinen aus folgenden sechs Bausteinen, deren Zusammenspiel in Abbildung 2.13 verdeutlicht wird:

- GPS-Antenne mit Vorverstärker
- Oszillator
- Hochfrequenz (HF)-Einheit
- Signalverarbeitungsprozessor
- Datenverarbeitungsprozessor
- Schnittstellen, Bedieneinheit

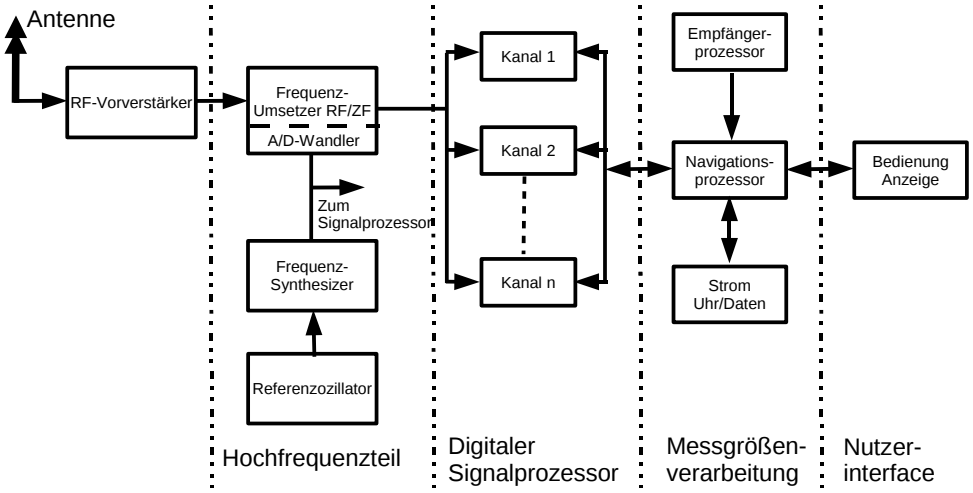


Abbildung 2.13: Schematischer Aufbau eines GPS-Endgerätes

Zum Empfang der von den Satelliten ausgestrahlen Signale benötigt jedes GPS-Empfangsgerät eine ungerichtete Antenne. Sie kann im Gerät selbst verbaut oder auch als externe Antenne (für einen besseren Empfang) mit dem Gerät verbunden sein. Es werden meist Spiralantennen, gekreuzte Dipole oder Patch-Antennen verwendet. Das empfangene Signal wird durch entsprechende Entstör- und Verstärkerschaltungen aufbereitet und an den Signalverarbeitungsprozessor weitergeleitet.

Die HF-Einheit übernimmt die Aufgabe der Frequenzumsetzung und der Verstärkung des Signals, sowie dessen Umwandlung in einen digitalen Datenstrom. Die auf den beiden verschiedenen Frequenzen (Frequenz L1 und L2) empfangenen Signale der Satelliten werden auf eine Zwischenfrequenz aufmoduliert und das sogenannte Nutzsignal verstärkt. Der nachfolgende A/D-Wandler[23] transformiert den analog vorliegenden Datenstrom in einen hochfrequenten digitalen Datenstrom und leitet diesen weiter an den Signalverarbeitungsprozessor. Die HF-Einheit ist die Komponente eines GPS-Empfängers, die sich über die Jahre am meisten verändert hat. Die ersten zivilen GPS-Geräte waren lediglich Einfrequenzempfänger, die ausschließlich mit dem L1-Signal arbeiteten. Militärische, sowie die heutigen zivilen GPS-Geräte, sind sogenannte Doppelfrequenzempfänger, die sowohl das L1-Signal, als auch das L2-Signal auswerten. Auch im empfangstechnischen Bereich wurden die GPS-Geräte weiterentwickelt. So gab es ehemals nur sogenannte sequentielle Empfänger, die nur einen Hardwarekanal besaßen und somit zu einem Zeitpunkt nur das Singnal eines Satelliten auswerten konnten. Dadurch entstanden relativ große Ungenauigkeiten; heute werden

[23] Analog/Digital Wandler

diese Geräte daher kaum noch genutzt. Die zweite Generation hatte auch nur einen Hardwarekanal, konnte aber durch ein sehr schnelles Multiplexen einen Pseudo-Parallelempfang simulieren, der die Genauigkeit der GPS-Geräte erheblich steigerte. Noch genauer ist der heutige Standard: der Mehrkanalempfänger. Er kann mit bis zu zwölf Hardwarekanälen auch bis zu 12 verschiedene Signale von unterschiedlichen Satelliten empfangen. Durch diesen synchronen Empfang erreicht man eine hohe Datenakquise, die fast in Echtzeit abläuft und somit eine hohe Präzision gewährleistet.

Die Signalverarbeitungsprozessoren werten den vom A/D-Wandler übermittelten Datenstrom aus und leiten die daraus berecheneten Rohdaten an den Datenverarbeitungsprozessor weiter. Zur Auswertung des digitalen Datenstroms besitzen die Signalverarbeitungsprozessoren Codegeneratoren, die sowohl den C/A-Code als auch den P-Code als Referenz-PRN[24]-Impulsfolge erzeugen. Wie genau die Signalverarbeitungsprozessoren aus dem empfangenen Datenstrom und der Referenz-PRN-Impulsfolge die Signallaufzeit und weitere Informationen gewinnen, wird genauer in Kapitel 2.2.4.1 besprochen.

Die eigentliche Positionsberechnung findet im Datenverarbeitungsprozessor statt. Hier werden die Rohdaten der Signalverarbeitungsprozessoren wie Trägerphasen, Codephasen und Navigationsdaten verarbeitet, sowie die einzelnen Signalverarbeitungsprozessoren gesteuert. Durch welche Verfahren die Position berechnet und präzisiert wird behandelt Kapitel 2.2.5.

Am Anfang dieses Kapitels über GPS wurde schon kurz auf die Funktionsweise des Systems hingewiesen. Hierbei spielte die Laufzeitmessung der einzelnen Signale von den verschiedenen GPS-Satelliten zum Empfänger eine entscheidende Rolle. Je genauer diese Messung ist, um so genauer kann die tatsächliche Position bestimmt werden. Am präzisesten könnte die Messung der Signallaufzeit mit einer im GPS-Empfänger integrierten Atomuhr durchgeführt werden. Dies würde aber die einzelnen Empfangsgeräte zu kostenintensiv und für einen mobilen Gebrauch zu groß werden lassen. Deswegen werden in den heutigen GPS-Geräten temperaturstabilisierte Quarzoszillatoren verwendet. Ihre Qualität ist maßgeblich für die Qualität der Referenzuhr verantwortlich und damit für die Genauigkeit des Empfängers. Alle Zeittakte und Frequenzen werden von diesem Oszillator abgeleitet. Wie eine hohe Genauigkeit auch ohne eine hochpräzise Atomuhr im Empfänger erreicht werden kann wird im Abschnitt 2.2.4.1 eingehend erläutert.

[24] Pseudo Random Noise

Die letzte Komponente ist die Schnittstelle oder das Bedienteil. Über diese Komponente werden die fertigen Positionsdaten an den Benutzer oder eine noch nachfolgende Komponente übermittelt. Dies kann sowohl ein einfaches Display sein, das nur die aktuelle Position in Längen- und Breitengrad anzeigt, als auch ein komplettes Navigationssystem, welches zurückgelegte Wege speichert und mögliche Routen zu einem vorgegebenen Ziel berechnet. Im Falle einer einfachen Schnittstelle, die nur zur Übermittlung von Positionsdaten dient, werden die Informationen über definierte Protokolle übertragen. Die bekanntesten Protokolle sind NMEA und SiRF, deren Spezifikation im Anhang zu finden ist (siehe Anhang A.4).

2.2.3 Funktionsweise des NAVSTAR-GPS

Dieses Kapitel beschäftigt sich mit der grundlegenden Funktionsweise von heutigen GPS-Geräten. Hierzu zählen das Zusammenspiel zwischen dem Raum- und dem Nutzersegment, als auch die im Nutzersegment ablaufenden mathematischen Berechnungen.

2.2.3.1 Das Signal

2.2.3.2 Die physikalische Struktur

GPS-Satelliten senden zur Zeit auf zwei Frequenzen ihre Informationen aus dem Orbit. Die erste Frequenz namens L1 dient der zivilen Nutzung. Sie leitet sich aus der Grundfrequenz[25] (10,23 MHz) des Satelliten ab. Sie ist um das 154-fache größer als die Grundfrequenz und beträgt somit 1575,42 MHz. Die zweite genutzte Frequenz ist die Frequenz L2=1227,60 MHz. Sie ist das 120-fache der Grundfrequenz des Satelliten und für die militärische Nutzung vorgesehen (vgl. Abbildung 2.14 auf Seite 29).

In der Einleitung dieses Kapitels wurde schon auf die Probleme hinsichtlich der Nutzung einer Frequenz durch mehrere Sender hingewiesen und Lösungsansätze angesprochen. Da alle Satelliten die gleichen Frequenzen benutzen, müssen mittels des CDMA-Verfahrens mehrere logische Kanäle geschaffen werden, damit der GPS-Empfänger die unterschiedlichen Informationen der einzelnen Satelliten empfangen und zuordnen kann. Beim CDMA-Verfahren moduliert jeder Sender der Trägerfrequenz einen unikaten Code auf, welcher zur Unterscheidung von anderen Sendern dient.

[25] vergleiche Kapitel 2.2.2.1

Dieser Code besitzt eine bestimmte Länge von Nullen und Einsen und könnte frei gewählt werden. Der Code, den die GPS-Satelliten auf der L1-Frequenz verwenden, besitzt eine Länge von 1023 Bits. Diese Bits werden allerdings im Zusammenhang mit CDMA nicht Bits sondern Chips genannt, um so den Unterschied zwischen den Bits der Nutzdaten und denen des Spreizcodes, die keine Nutzinformationen beinhalten, deutlich zu machen[26]. Die sogenannte Chips-Rate beträgt 1,023 Mcps[27]. Daraus folgt, dass der 1023 Bit lange Spreizcode 1000 mal in der Sekunde gesendet wird. Der Spreizcode den die GPS-Satelliten auf der Frequenz L1 verwenden, wird als C/A[28]-Code bezeichnet. Jeder einzelne Satellit besitzt einen eigenen Code, der quasi zufällig gewählt wurde. Tatsächlich aber stammen die verschiedenen Codes aus zwei Codegeneratoren für Goldfolgen mit den folgenden Generatorpolynomen[29]:

- $G_1 = 1 + x^3 + x^{10}$
- $G_2 = 1 + x^2 + x^3 + x^6 + x^8 + x^9 + x^{10}$

Die unterschiedlichen Codes entstehen durch Phasenverschiebung, also durch Änderung des Anfangszustands eines Generators. Jeder Satellit moduliert dann seinen Code als Pseudo Random Noise[30]-Impulsfolge auf die Trägerfrequenz auf. Der Vorteil dieser nicht zufällig gewählten Codes liegt darin, dass sie im sogenannten Coderaum fast orthogonal zueinander stehen[31] und sich dadurch kaum gegenseitig beeinflussen. So können mit Hilfe der Kreuzkorrelation(vgl. Kapitel 2.2.4.1), welche auch bei der Bestimmung der Signallaufzeit verwendet wird, sehr gut die verschiedenen Signale wiederhergestellt werden.

Der C/A-Code kann mit einem künstlichen Fehler versehen werden um die Genauigkeit bei der Positionsbestimmung zu senken. Dies wird allerdings nur in Krisenzeiten auf das zivile Signal angewendet, um so feindlichen Einheiten eine Nutzung von GPS zu erschweren. Die Verfälschung wird durch zeitliche Schwankung (Jitter) im Taktsignal erreicht[32]. Diese Funktionalität wird Selective Availability genannt (kurz: SA). Sie ist allerdings seit dem 2. Mai 2000[33] abgeschaltet. Der C/A-Code dient zur Bereitstellung des Standard-Ortungsservices (Standard Positioning Service, Abkürzung SPS).

[26] vergleiche hierzu [Sch03]:82, 112
[27] Megachips per second
[28] Coarse/Acquisition, frei übersetzt: grobe Datenerfassung
[29] mehr zu Goldfolgen und Codegeneratoren [GK09]
[30] kurz: PRN
[31] mehr hierzu in [GK09]
[32] vergleiche hierzu [HWLC01]
[33] in der Nacht vom 1. auf den 2. Mai 5:05Uhr (MEZ), siehe http://army-gps.robins.af.mil/Announcements/sa-off_press_release.htm oder [KH06]:4

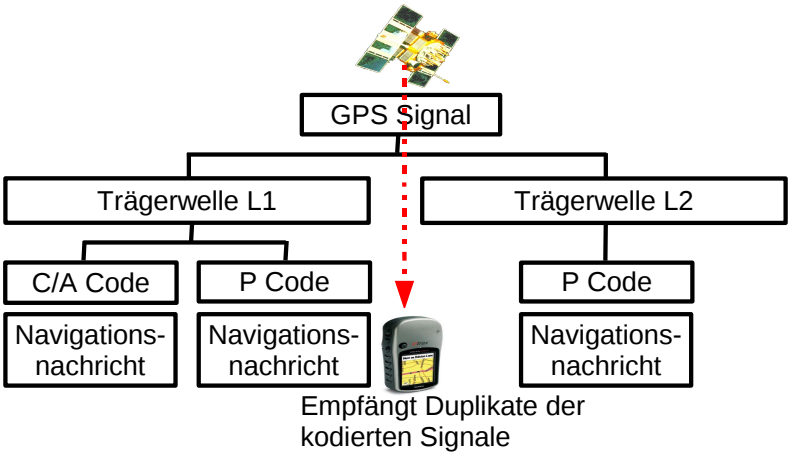


Abbildung 2.14: GPS Signalstruktur

Der zweite bei GPS verwendete Code ist der P-Code (Protected oder Precise-Code = Feinerfassung) bzw. der P/Y-Code, der den Präzisions-Ortungsservice (Precise Positioning Service, Abkürzung PPS) ermöglicht. Er wird sowohl auf die Frequenz L1 als auch auf die Frequenz L2 aufmoduliert (90° phasenverschoben zum C/A-Code). Der P-Code ist eine JPL-Folge, die mit Hilfe eines Codegenarators mit vier linearen Schieberegistern (kurz LFSR) erzeugt wird [34]. Der Code ist, wie der C/A-Code, öffentlich bekannt. Jeweils zwei Schieberegister erzeugen die sogenannten Codes X1 und X2. Der X1-Code besitzt eine Länge von 15 345 000 Bit und der X2-Code eine Länge von 15 345 037 Bit. Aufgrund des Bildungsgesetzes für JPL-Folgen ergibt sich folgende Länge des P-Codes:

$$15345000 * 15345037 = 2,34569592765 * 10^{14}$$

Der P-Code wird mit einer Bit-Rate, genauer gesagt mit einer Chiprate, von 10,23Mcps gesendet. Dadurch ergibt sich eine zeitliche Länge (Ausbreitung) des P-Codes nach folgender Rechnung von ca. 38 Wochen:

$$\frac{2,34569592765 * 10^{14} chips}{10230000 \frac{chips}{Sek}} = 23017555,5 Sek =$$

$$= 383625,925 Min = 6393,76541667 Std \approx 266,4 Tage \approx 38 Wochen$$

Jeder GPS-Satellit bekommt einen Teil des P-Codes zugewiesen, der eine zeitliche Ausdehnung von einer Woche umfasst. Somit setzt jeder GPS Satellit zum

[34] vgl. [GK09]

Wochenanfang (Sonntag 0.00 Uhr GPS-Zeit[35]) seinen P-Codegenerator wieder auf seinen Anfangswert zurück, der dann wieder eine Woche lang den speziellen Codeabschnitt generiert. Die höhere Genauigkeit von GPS-Empfängern, die mit dem P-Code arbeiten, beruhen einerseits auf der höheren Chiprate und andererseits auf dem nicht künstlich verfälschten Taktsignal (SA).

Des Weiteren kann der P-Code noch verschlüsselt werden. Dieses nicht öffentliche Verfahren namens Anti-Spoofing (engl.: to spoof, beschwindeln) hat zwei Effekte. Der erste dient, wie der Name schon sagt, der Vorbeugung der Verfälschung des GPS-Signals. So kann zum Beispiel ein regionaler Störsender in einem Krisengebiet von militärischen GPS-Empfängern erkannt werden. Der zweite Effekt ist, dass GPS-Empfänger, die nicht mit der Verschlüsselung vertraut sind, das GPS-Signal auf der L2 Frequenz nicht entschlüsseln, und somit auch nicht den PPS nutzen können. Den verschlüsselten P-Code nennt man auch P/Y-Code. Seine Generatorpolynome sind nicht öffentlich bekannt.

Von dem ungefähr 38 Wochen langen P-Code sind 32 einzelne Wochensegmente für die verschiedenen GPS-Satelliten reserviert. Die restlichen Codesegmente stehen der Überprüfung und Wartung der Satelliten zur Verfügung.

Die Informationen, die mit Hilfe des P-Code bzw. P/Y-Code versendet werden, sind exakt die gleichen, auch im verwendeten Format, wie die mit dem C/A-Code versendeten. Tabelle 2.2 vedeutlicht noch einmal die Unterschiede bzw. die Gemeinsamkeiten zwischen dem C/A-Signal und dem P/Y-Signal. Im folgenden Abschnitt wird die Information beschrieben, die dem physikalischen Signal aufmoduliert wird.

Code	**C/A-Code**	**P/Y-Code**
Generation	zwei 10-Bit Schieberegister	vier 10-Bit Schieberegister
Geschwindigkeit	1,023 Mcps	10,23 Mcps
Länge	1023 chips	$2,34569592765 * 10^{14}$ chips
Periode	0,001 s	23 017 555,5 s
Service	SPS	PPS
Information	GPS-Navigationsmitteilung	GPS-Navigationsmitteilung

Tabelle 2.2: Vergleich zwischen C/A- und P/Y-Code

[35] Die GPS-Zeit ist kontinuierlich und unterscheidet sich von der UTC-Zeit (oder Weltzeit), da die UTC-Zeit von Zeit zu Zeit angepasst wird (z.B. Einfügen einer Schaltsekunde)

2.2.3.3 Die Navigationsmitteilung

In der Einleitung dieses Kapitels wurde schon auf die Grundidee von GPS eingegangen: man bestimmt die eigene Position mit Hilfe der Entfernung zu bekannten Punkten, welche die Satelliten darstellen. Die Informationen über die Position der Satelliten beziehen die GPS-Empfänger über GPS-Navigationsmitteilung. Eine GPS-Nachricht umfasst insgesammt 1500 Bits und wird als ein Frame (Rahmen) bezeichnet. Ein Frame ist in fünf Subframes (Unterrahmen) mit je 300 Bits aufgeteilt. Die grobe Aufteilung der Informationen, die auf die einzelnen Subframes verteilt werden, sieht folgendermaßen aus:

- **1. Subframe**:
  Mit diesen Parametern wird das Uhrenverhalten bzw. der Uhrenzustand der jeweiligen aktuell verwendeten Atomuhr eines Satelliten beschrieben. Aus diesen Daten und den mit jedem Subframe versendeten HOW[36]-Daten kann jedes GPS-Gerät die aktuelle GPS-Zeit berechenen, die zur Bestimmung der Position benötigt wird.

- **2. und 3. Subframe**:
  Da die einzelnen Satelliten nicht geostationär sind, senden sie permanent aktuelle Daten zu ihrer aktuellen Umlaufbahn (Ephemeriden-Daten). So kann jeder GPS-Empfänger die aktuelle Position jedes Satelliten, von dem Daten empfangen werden, berechnen und so die zur Positionsbestimmung notwendigen bekannten Punkte ermitteln.

- **4. Subframe**:
  Der vierte Subframe enthält sogenannte Almanachdaten für die Satelliten 25 - 32, Ionosphärenkorrekturdaten, spezielle Nachrichten sowie UTC-Zeitinformationen, die im ASCII[37]-Format kodiert sind. Die Almanachdaten umfassen in vereinfachter Form Informationen über die Identifikationsnummer und Bahnparameter aller Satelliten, deren technischen Zustand und ihre momentane Konfiguration. Durch diese groben Informationen kann ein GPS-Empfänger schneller eine Ortung durchführen, da er so ungefähr weiß nach welchen GPS-Satelliten er suchen muss. Da hier eine sehr große Menge an zu versendenden Daten vorliegt, wird diese aufgeteilt. Die Verteilung erfolgt über 25 Frames und so dauert es 12,5 Minuten um die kompletten Daten zu empfangen.

[36] Hand Over Word, vgl. hierzu [KH06]:143
[37] American Standard Code for Information Interchange, vgl. Anhang A.3

- **5. Subframe**:
  Der letzte Unterrahmen enthält die Almanachdaten für die Satelliten 1 - 24 sowie Zeit und GPS-Wochennummer. Auch hier sind die Daten über insgesamt 25 Frames verteilt.

In Abbildung 2.15 ist der generelle Aufbau einer GPS-Nachricht abgebildet und im Anhang A.2 befindet sich eine detaillierte Übersicht. Jeder Subframe wird weiter in zehn Worte mit jeweils 30 Bit unterteilt. Das erste Wort jedes Subframes ist das Telemetry-Word (TLM), das neben Informationen zur Aktualität der Ephemeridendaten eine Preamble enthält. Die Preamble, bestehend aus der Bitfolge '10001011', steht am Anfang des Wortes und dient der Synchronisation bzw. zum Auffinden des Anfangs eines Subframes.

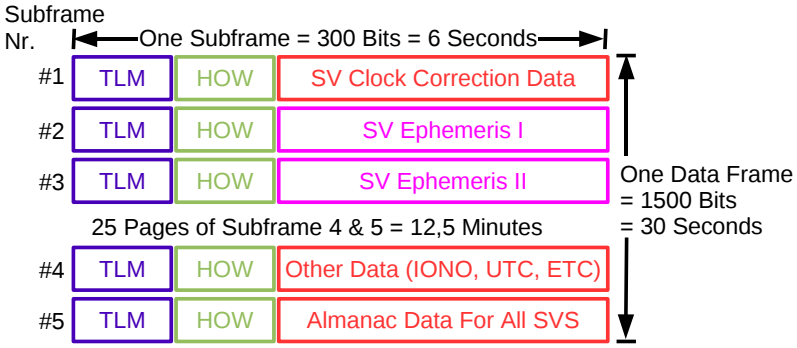


Abbildung 2.15: Genereller Aufbau einer GPS-Nachricht

Das zweite Wort jedes Subframes ist das sogenannte HOW[38], welches angibt wie oft sechs Sekunden seit Sonntag 0.00 Uhr vergangen sind. Diese Zählvariable wird auch als Z-Count bezeichnet und hat eine Länge von 17 Bit. Diese Information verwenden militärische GPS-Geräte für ein schnelleres Umschalten auf den P- bzw P/Y-Code, da sie so ungefähr den aktuellen Codeabschnitt des sehr langen Codes abschätzen können. Abbildung 2.16 zeigt den genaueren Aufbau der beiden ersten Wörter 'TML und 'HOW'.

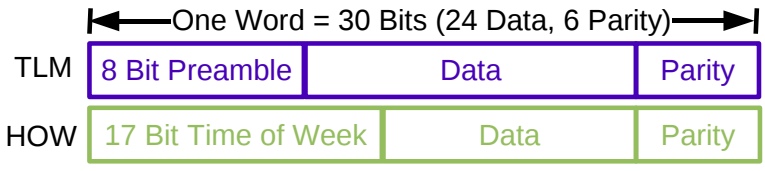


Abbildung 2.16: Detailierter Aufbau der Wörter TML und HOW

[38] Hand Over Word

2.2.4 Beobachtungsgrößen

Seit Mitte des 20. Jahrhunders sind Verfahren bekannt mit deren Hilfe man die Möglichkeit hat Entfernungen mittels Radiowellen zu ermitteln. Sie basieren auf der gemessenen Signallaufzeit, die die Radiowelle von der Sendestation bis zum Empfänger benötigt. Die Signalausbreitungsgeschwindigkeit von Radiowellen beträgt, genau wie die Lichtgeschwindigkeit, 299792458 Meter pro Sekunde. Multipliziert man nun die gemessene Signallaufzeit mit der Signalausbreitungsgeschwindigkeit, so erhält man die Entfernung zwischen Sender und Empfänger. Da die Signalausbreitungsgeschwindigkeit extrem hoch ist, ist wie schon erwähnt eine sehr genaue Messung der Signallaufzeit erforderlich. Durch einen Fehler von 1ms bei der Signallaufzeitmessung entsteht so ein Entfernungsfehler von ungefähr 300 km[39].

Zur Ortung mittels GPS gibt es verschiedene Beobachtungsgrößen (Observables), die gemessen und ausgewertet werden. Die Beobachtungsgrößen sind die Radiowelle als Trägersignal selbst, sowie der auf dieser Trägerwelle aufmodulierte digitale Code[40] (C/A-Code bzw. P- oder P/Y-Code). Während sich die Auswertung der Codephase auf den aufmodulierten, digitalen Spreizcode bezieht, befassen sich der Doppler-Count und die Trägerphase mit der Trägerwelle selbst. In den folgenden drei Abschnitten wird nur kurz auf die einzelnen Beobachtungsgrößen und ihre Auswertung eingegangen, da eine genauere Betrachtung der teilweise sehr komplexen mathematischen Verfahren den Rahmen dieser Arbeit sprengen würde.

2.2.4.1 Uhrensynchronisation und Codephase

Zur groben, dafür aber schnellen Positionsbestimmung verwenden die meisten GPS-Geräte eine Methode, die durch Betrachtung des PRN-Codes der einzelnen Satelliten ermöglicht wird. Es wird versucht mittels Messung der Laufzeiten von Codes die genaue Signallaufzeit zu berechenen. Zunächst muss allerdings die Uhr im GPS-Empfänger mit den Atomuhren in den Satelliten synchronisiert werden. So kann dann durch einfache Differenzbildung des Sendezeitpunkts T_s und des Empfangszeitpunkts T_e die Signallaufzeit errechnet werden.

Da sich die Satelliten in einer Umlaufbahn von 20200 km Höhe befinden und zumeist nicht senkrecht über dem GPS-Empfänger stehen, wird von einer Grundentfernung von 21000 km ausgegangen (vgl. Abb 2.17). Dies führt bei einer

[39] $299792458\frac{m}{s} = 299792,458\frac{m}{ms} = 299,792458\frac{km}{ms}$
[40] genauer: der Spreizcode, vgl Kapitel 2.2.3.1

Signalausbreitung mit Lichtgeschwindigkeit ($c = 299792458\frac{m}{s}$) zu einer Signallaufzeit von ungefähr 0,07 Sekunden. Empfängt also ein GPS-Gerät seine erste GPS-Nachricht stellt es seine interne Quarzuhr auf den enthaltenen Zeitstempel, abzüglich der Signallaufzeit von 0,07 Sekunden ein. Dies ist natürlich noch nicht die tatsächliche GPS-Zeit, da ja nur von einem angenommenen Grundwert ausgegangen wurde, stellt aber zumindest eine Annäherung dar.

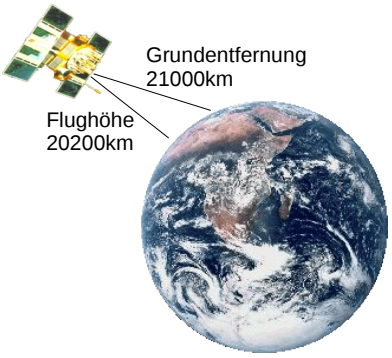


Abbildung 2.17: Unterschied zwischen Flughöhe und Grundentfernung eines GPS-Satelliten

Der Zeitversatz zwischen den Atomuhren der Satelliten (die alle absolut synchron laufen) und der Quarzuhr des GPS-Empfängers wird auch als Uhrenfehler oder Bias bezeichnet. Die Ermittlung der Signallaufzeiten der nachfolgenden GPS-Nachrichten unterliegen also alle diesem Uhrenfehler und führen so auch zur fehlerhaften Berechnung der Entfernung zwischen Satellit und GPS-Empfängergerät. Die so ermittelten Entfernungen werden deshalb auch als Pseudoentfernungen oder Pseudoranges bezeichnet. Betrachten wir zur Verdeutlichung dieses Problems als Beispiel einmal die Situation mit zwei Satelliten und einer zweidimensionalen Welt. Nehmen wir an, dass der GPS-Empfänger seine erste GPS-Nachricht schon empfangen hat und somit seine Uhr schon einmal ungefähr eingestellt hat. In Abbildung 2.18 ist der Zeitpunkt des ersten Empfangs zweier GPS-Nachrichten von zwei verschiedenen GPS-Satelliten nach der ersten vorläufigen Uhreinstellung dargestellt.

Der tatsächliche Standort unseres GPS-Empfängers ist in Punkt S. Die Uhrzeit, die er sich aus der ersten GPS-Nachricht und der Grundentfernung errechnet hat, weicht um 2 ms von der eigentlichen GPS-Zeit ab. Der GPS-Empfänger erhält kurz nacheinander erst eine GPS-Nachricht von Satellit S1 und dann eine Nachricht von Satellit S2. Nun bildet das GPS-Gerät die Differenz der beiden verschiedenen Empfangszeiten (16:00:076 und 16:00:078) mit dem in der Nachricht enthaltenen Zeitstempel (16:00:000 bei beiden Nachrichten) und bekommt

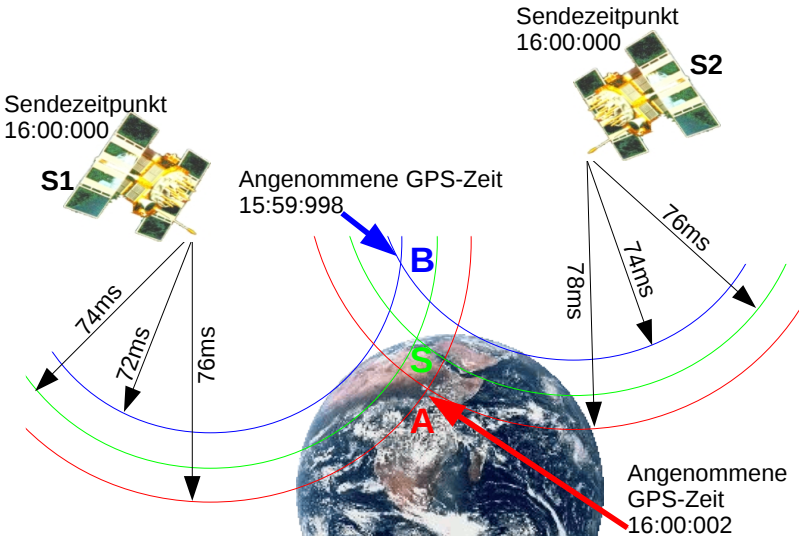


Abbildung 2.18: Verschiedene Pseudoentfernungen durch Uhrenfehler (Bias) von 2ms und -2ms, bei einer tatsächlichen Signallaufzeit von 74ms bei Satellit S1 und 76ms bei Satellit S2

so zwei Signallaufzeiten, die um genau 2 ms zu lang sind; 76 ms bei der Nachricht von Satellit S1 und 78 ms bei der Nachricht von Satellit S2. Die daraufhin berechneten Pseudoranges schneiden sich in Punkt A. Dies wäre also der angenommen Standort des Empfängers. Vorausgesetzt, dass aufgrund der Annahme der Grundentfernung ein Uhrenfehler von -2ms entstanden wäre (wir wissen ja noch nicht wie groß der Uhrenfehler wirklich ist), würden wir zu kurze Signallaufzeiten erhalten und somit von kürzeren Pseudoentfernungen ausgehen. Der Schnittpunkt der beiden Entfernungskreise läge dann im Punkt B.

Wie wir leicht sehen können gibt es also mehrere potentielle Standorte. Welcher ist aber der tatsächliche Standort? In Annahme, dass wir uns auf der Erdoberfläche befinden und nicht im Erdinnern oder im Weltraum, könnten wir den Uhrenfehler berechnen. Da die Erde allerdings keine exakte geometrische Kugel ist führt dies zu Problemen (vgl. Kapitel 2.2.8). Auch mit dem angenähertem Geoid des WGS-84[41] wäre so zum Beispiel die Positionsbestimmung eines Flugzeuges, das sich mehrere Kilometer über der Erdoberfläche bewegt, nicht möglich. Wenn wir nun in unserem zweidimensionalen Beispiel noch einen dritten Satelliten hinzufügen, können wir den Uhrenfehler genau berechnen. Es gibt nämlich nur einen einzigen Punkt aller annehmbaren Pseudoranges, in

[41] durch verschiedene Parameter der Erdoberfläche angepasster Rotationsellipsoid, näheres hierzu in Kapitel 2.2.8

dem alle Signallaufzeiten den gleichen Fehler aufweisen. Dies ist unser gesuchter Standort. Abbildung 2.19 verdeutlicht diese Überlegung nocheinmal.

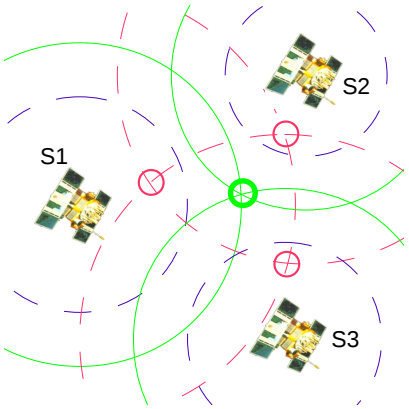


Abbildung 2.19: Berechnung des Uhrenfehlers im Zweidimensionalen mit Hilfe von 3 GPS-Satelliten

In unserem zweidimensionalen Beispiel benötigen wir also drei Satelliten um den Uhrenfehler zu ermitteln und eine Positionsbestimmung durchzuführen. In der Realität befinden wir uns jedoch in einer dreidimensionalen Welt. Somit müssen wir neben dem Uhrenfehler und den x- und y-Koordinaten noch die dritte Koordinate (z-Koordinate) berechnen. Hierzu wird das Signal von einem vierten Satelliten nötig um mit dem GPS-Empfänger theoretisch die genaue GPS-Zeit der Atomuhren in den Satelliten zu berechen. Durch physikalische Gegebenheiten, wie unterschiedliche Refraktion[42] (atmosphärische Störungen) der einzelnen Signale und nur annähernder Lichtgeschwindigkeit, entstehen trotzdem Fehler, sodass die ermittelten Entfernungen immer noch sogenannte Pseudoentfernungen sind.

Für eine möglichst exakte Entfernungsbestimmung müsste ein GPS-Gerät in der Lage sein, Zeitunterschiede im Nanosekundenbereich zu messen. Diese Messgenauigkeit kann aber nicht von den verwendeten Quarzuhren geleistet werden. Um eine noch genauere Synchronisation und somit eine noch genauere Signallaufzeitmessung durchzuführen wird die PRN-Impulsfolge jedes empfangenen Satellitensignals verwendet. Zunächst beschränken wir uns auf den C/A-Code, der auf ein Trägersignal mit der Sendefrequenz (f_s) 1575,42 Mhz aufmoduliert wird. Aufgrund der Bewegung der Satelliten bzw. auch der Bewegung des GPS-Gerätes wird das Satellitensignal dopplerverschoben mit der Frequenz f_d empfangen[43].

[42] siehe 2.2.6
[43] Genaueres zum Doppler-Effekt in Kapitel 2.2.5.1

Der GPS-Empfänger erzeugt intern eine Referenzfrequenz f_r, vergleicht diese mit der Frequenz f_d des empfangenen Signals und versucht permanent, die Referenzfrequenz an die Frequenz f_d des Signals anzupassen, sodass $f_r = f_d$ ist. Auf dieses intern erzeugte Trägersignal wird, wie auch im Satelliten, der entsprechende PRN-Code aufmoduliert. So entsteht ein Duplikat des Satellitensignals, allerdings mit einer zeitlichen Verschiebung (siehe Abb.2.20).

Mit Hilfe des Kreuzkorrelationsprozesses ist es möglich die Zeit zu ermitteln, die die beiden Signale beim Empfang im GPS-Gerät zueinander verschoben sind. Zur Veranschaulichung des Korrelationsprozesses betrachten wir wieder ein kleines Beispiel. In Abbildung 2.21 ist die Ausgangssituation zu sehen, die sowohl das Referenzsignal als auch das empfangene Satellitensignal darstellt. Hierbei ist zu beachten, dass eine '1' im Code auch als '1' bei der Kreuzkorrelation dargestellt wird, eine '0' im Code jedoch mit '-1'.

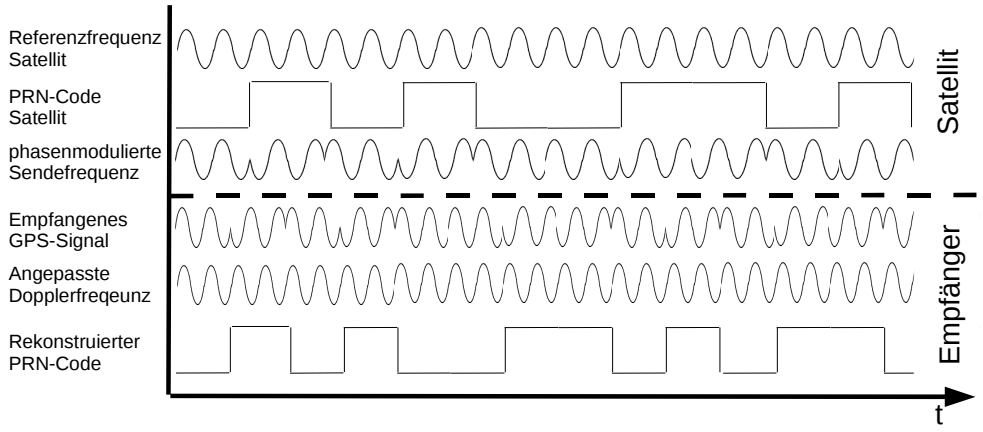


Abbildung 2.20: Erzeugung und Rekonstruktion eines PRN-Signals

Bei der Kreuzkorrelation werden nun die Werte der beiden Signale einer Zeiteinheit (hier ca. $0{,}9775\mu s$[44]) miteinander multipliziert. Bei einer Übereinstimmung der beiden Werte erhalten wir somit als Ergebnis '1', und bei unterschiedlichen Signalwerten eine '-1'. Diese Produktbildung wird nun für jede Zeiteinheit über die gesamte Länge des Codes durchgeführt. In Abbildung 2.21 ist nur ein Teilstück des PRN-Codes dargestellt. Danach schlussendlich wird die Summe über die einzelnen Produkte gebildet. Als Ergebnis erhalten wir für unser Teilstück einen Wert von '2'. Die Summe des gesamten Codes wird noch durch Division durch 1023 auf 1 normiert.

[44] entspricht genau einer Bitlänge des PRN-Codes: $\frac{1s}{1000*1023} \approx 0,9775\mu s$

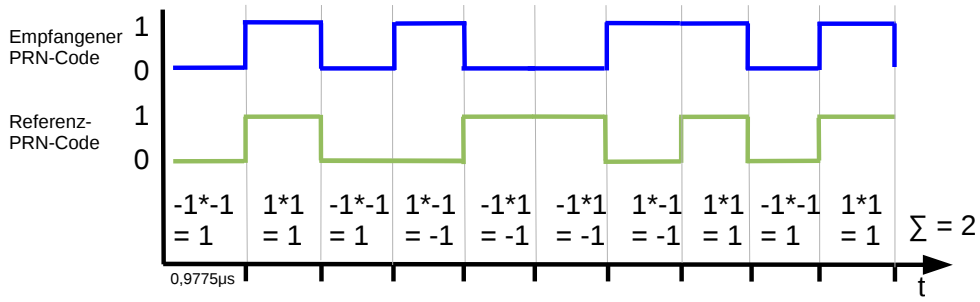


Abbildung 2.21: Ausgangssituation bei dem Empfang eines PRN-Codes

Im Folgenden wird der Referenzcode jeweils um eine Zeiteinheit verschoben und die oben beschriebenen Schritte werden wiederholt. In Abbildung 2.22 ist exemplarisch einmal die Kreuzkorrelation bei einer Verschiebung des Referenzsignals um -4 bis 6 Zeiteinheiten dargestellt. Bei der Länge einer Zeiteinheit von ca. 0,9775μs kann man so 1023 Verschiebungen durchführen. Die Ergebnisse der einzelnen Kreuzkorrelationen werden dann in einer Korrelationsfunktion zusammengefasst. Das Maximum der Korrelationsfunktion ist leicht zu erkennen und wird in unserem Beispiel bei einer Verschiebung von zwei Zeiteinheiten erreicht. Somit sind die Signale um $2 * 0,9775\mu s$ zueinander verschoben.

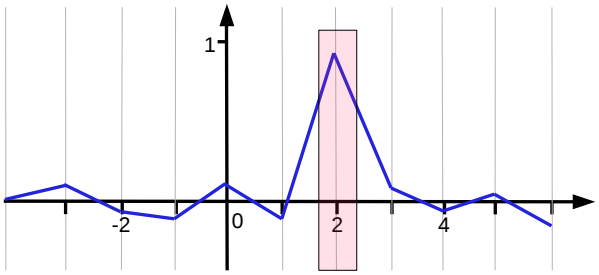


Abbildung 2.22: Beispiel einer Korrelationsfunktion
(durch Divison durch 1023 auf 1 normiert)

Die zeitliche Verschiebung des Referenzsignals kann mit dem heutigen technischen Standard bis auf ca. ein Prozent eines Bits genau an das empfangene Signal angenähert werden. Dies entspricht, bei einer ungefähren Signallänge von 300 km des gesamten C/A-Codes, einer Bitlänge von 300 m, also einer Genauigkeit von etwa 3 Metern (1 Prozent).

Die oben beschriebene Zeitdifferenz ist allerdings noch nicht die gesamte Signallaufzeit vom Satelliten zum Empfänger. Da der C/A-Code sich jede Millisekunde wiederholt, kann die so bestimmte Zeitdifferenz auch nur innerhalb einer Millisekunde liegen. Die gesamte Signallaufzeit liegt jedoch im Bereich von ungefähr 70 ms (vergleiche hierzu Kapitel 2.2.4.1). Wie lange ist nun die

tatsächliche Signallaufzeit? Um diese zu ermitteln gibt es mehrere, mehr oder weniger genaue, Möglichkeiten. In der Praxis hängt die verwendete Methode vom GPS-Gerät ab; meist wird jedoch eine Kombination aus mehreren Methoden angewendet. Die einfachste hiervon ist die Nutzung der internen Quarzuhr, die eine grobe Laufzeitmessung ermöglicht.

Eine weitere Überlegung basiert auf der Annahme, dass sich der C/A-Code pro Millisekunde bzw. alle 300 km wiederholt:

$$1ms * c = 0,001s * 299792458\frac{m}{s} \approx 300km \tag{2.1}$$

Somit erhalten wir mehrere mögliche Signallaufzeiten, die sich jeweils genau um 1ms voneinander unterscheiden, was einer räumlichen Entfernungen von ca. 300 km entspricht. Nun können wir durch verschieden Annahmen, ähnlich wie bei der Bestimmung des Uhrenfehlers, eine höchst wahrscheinliche Position vermuten.

Wir gehen davon aus:

- dass wir uns in der Nähe der Erdoberfläche, bzw. genauer in der Nähe der Geoidoberfläche, befinden und nicht im Erdinnern oder im Weltraum.
- dass das GPS-Gerät aufgrund der Konstellation der „sichtbaren“ GPS-Satelliten und den empfangenen Almanachdaten eine angenäherte Position bestimmen kann. (Dies kann jedoch bei Inbetriebnahme des Gerätes bis zu 12,5 Minuten dauern[45]).
- dass wir uns seit der letzen Positionsbestimmung (die meist nur eine Sekunde zurückliegt) nicht weiter als 300 km bewegt haben. Diese Annahme führt jedoch zu Problemen bei Geräten, die noch keine Positionsbestimmung durchgeführt haben oder abgeschaltet und an einer weit entferneten Position wieder in Betrieb genommen wurden. Aus diesem Grund bieten manche GPS-Geräte die Möglichkeit beim Start eine ungefähre Positionsangabe einzugeben.

Die bisher beschriebene Vorgehensweise zur Bestimmung der Entfernung vom Empfänger zu den Satelliten wurden anhand des C/A-Codes erläutert. Die gleichen Methoden werden aber auch bei der Laufzeitmessung mit Hilfe des P-Codes, bzw. des militärisch genutzten P/Y-Codes, verwendet. Da der P-Code allerdings wesentlich länger ist (vgl. hierzu Kapitel 2.2.3.1), würde eine

[45] vgl. Anfang Kapitel 2.2.3.3

komplette Erstellung der Kreuzkorrelationsfunktion viel zu lange dauern. Man führt deshalb zuerst die Synchronisation des C/A-Codes mit der Referenzfrequenz durch und bezeichnet dies auch als „Einrasten des C/A-Codes“. Mit Hilfe des Hand-Over Word (HOW, siehe Kapitel 2.2.3.3), wird der ungefähre aktuelle Codeabschnitt des P- bzw. P/Y-Codes geschätzt und dann mit Hilfe der Kreuzkorrelation in der direkten Umgebung um diese Codestelle die tatsächliche Codeverschiebung ermittelt. Die so bestimmte Verschiebung ist aufgrund der 10-fach höheren Bitrate bei der Aussendung auch dementsprechend genauer, nämlich bis zu 30 cm Annäherung.

Im folgenden Kapitel werden diese Überlegungen in ein mathematisches Modell überführt, welches zur Positionsberechnung im Standard-Ortungsservice (SPS) verwendet wird.

2.2.5 Positionsbestimmung

Modellierung der Pseudoentfernung Die im vorherigen Kapitel ermittelten Signallaufzeiten sind nicht die genauen, tatsächlichen Laufzeiten. Durch Uhrenfehler, Refraktion und weitere Fehlerquellen, auf die in Abschnitt 2.2.6 eingegangen wird, wird die Zeitmessung verfälscht. Somit stimmen die daraus berechenbaren Entfernungen nicht mit der tatsächlichen Entfernung von Satellit und Empfänger überein. Die sogenannte Pseudoentfernung p_i zwischen dem Satelliten i und der aus der Codephase gemessenen Signallaufzeit τ_i berechnet sich nach folgender Formel:

$$p_i = c_0 * \tau_i \tag{2.2}$$

Da elektromagnetische Schwingungen eine ähnliche Ausbreitungsgeschwindigkeit wie Licht besitzen, wird auch hier mit der Konstante c_0 der Lichtgeschwindigkeit[46] gearbeitet. Versuchen wir uns jetzt bewußt zu machen, was eine Pseudoentfernung überhaupt ist. Sie besteht aus der tatsächlichen geometrischen Entfernung r_i, plus oder minus einer Strecke, die sich aus der fehlerhaften Laufzeitmessung aufgrund des Uhrenfehlers berechnet:

$$p_i = r_i + c_0 * \Delta t_u \tag{2.3}$$

Betrachten wir uns nun einmal die tatsächliche Entfernung r_i des Satelliten i zu unserem aktuellen Standpunkt. Die Koordinaten eines jeden Punktes werden

[46] $c_0 = 299\,792\,458\ \frac{\mathrm{m}}{\mathrm{s}} = 2{,}997\,924\,58 \cdot 10^8\ \frac{\mathrm{m}}{\mathrm{s}}$

im Bezugsystem WSG-84[47], einem dreidimensionalen Koordinatensystem, mit einem Positionsvektor angegeben. Der Punkt, an dem sich der Satellit aktuell befindet wird mit dem Positionsvektor $\vec{X_{S_i}}$, mit den Koordinaten x_{S_i}, y_{S_i} und z_{S_i} ausgedrückt. Analog dazu ist der Positionsvektor des Standortes $\vec{X_p}$ mit den Koordinaten x_p, y_p und z_p bezeichnet. In den folgenden Schritten greifen wir auf einige einfache Formeln aus der analytischen Geometrie zurück. Der Betrag der Differenz der beiden Positionsvektoren entspricht genau der Entfernung zwischen den beiden Punkten, welche zu ermitteln ist.

$$r_i = |\vec{X_{S_i}} - \vec{X_p}| \tag{2.4}$$

$$r_i = \sqrt{[(x_{s_i} - x_p)^2 + (y_{s_i} - y_p)^2 + (z_{s_i} - z_p)^2]} \tag{2.5}$$

Setzen wir nun unsere tatsächliche geometrische Entfernung in die Gleichung 2.3 ein, so erhalten wir folgende Formel für die Pseudoentfernung zwischen einem Satelliten i und dem Standort des GPS-Empfängers:

$$p_i^{'} = \sqrt{[(x_{s_i} - x_p)^2 + (y_{s_i} - y_p)^2 + (z_{s_i} - z_p)^2]} + c_0 * \Delta t_u \tag{2.6}$$

Es gibt drei Verfahren, die ein GPS-Gerät nutzen kann, um aus den empfangenen Daten die aktuelle Position zu ermitteln:

- das geschlossene Verfahren
- das iterative Verfahren
- ein Verfahren mit Kalman-Filter

Geschlossenes Verfahren Das geschlossene Verfahren ist schon seit einigen Jahren nicht mehr den Anforderungen gewachsen und wird somit auch nicht mehr verwendet. Aufgrund der heutzutage geringen Bedeutung verzichte ich an dieser Stelle auf eine genauere Betrachtung.

Iteratives Verfahren Das iterative Verfahren basiert auf der Linearisierung nach Taylor und fand vor allem früher, aber auch heute noch, Verwendung, da die Realisierung des Algorithmus sehr einfach und auf Seiten der Hardware kostengünstig ist. Wie im Kapitel 2.2.2.1 schon erwähnt wurde, werden zur genauen Positionsbestimmung im dreidimensionalen Raum mittels GPS mindestens vier Satelliten benötigt. So wird aus oben genannter Formel 2.6 ein nichtlineares

[47] vgl. Kapitel 2.2.8

Gleichungssystem mit vier Formeln, indiziert von eins bis vier. Die durch den Uhrenfehler $c_0 * \Delta t_u$ entstandene Differenz wird in diesen Gleichungen durch k ersetzt und umfasst zusätzlich zu diesem auch noch andere Fehlerquellen. Der Algorithmus beginnt mit den Koordinaten eines Näherungspunktes. Dieser liegt innerhalb des Dreiecks, das durch die Schnittpunkte jeweils zweier gemessener Pseudoentfernungen entstandenen ist (vgl. hierzu 2.19 auf Seite 36). Die näherungsweise angenommenen Koordinaten sind $x_p^{'}$, $y_p^{'}$ und $z_p^{'}$. Somit sind $x_p^{'}$, $y_p^{'}$, $z_p^{'}$ und k' die angenommenen Näherungswerte in unserem Gleichungssystem; $x_{s_i}, y_{s_i}, z_{s_i}$ sind bekannte Größen, die aus den empfangenen GPS-Nachrichten gewonnen werden und $r_i^{'}$ sind die daraus berechneten Näherungsentfernungen.

$$r_1^{'} = \sqrt{\left[(x_{s_1} - x_p^{'})^2 + (y_{s_1} - y_p^{'})^2 + (z_{s_1} - z_p^{'})^2\right]} + k^{'} \tag{2.7}$$

$$r_2^{'} = \sqrt{\left[(x_{s_2} - x_p^{'})^2 + (y_{s_2} - y_p^{'})^2 + (z_{s_2} - z_p^{'})^2\right]} + k^{'} \tag{2.8}$$

$$r_3^{'} = \sqrt{\left[(x_{s_3} - x_p^{'})^2 + (y_{s_3} - y_p^{'})^2 + (z_{s_3} - z_p^{'})^2\right]} + k^{'} \tag{2.9}$$

$$r_4^{'} = \sqrt{\left[(x_{s_4} - x_p^{'})^2 + (y_{s_4} - y_p^{'})^2 + (z_{s_4} - z_p^{'})^2\right]} + k^{'} \tag{2.10}$$

Diese vier Näherungsentfernungen entsprechen nicht den tatsächlich gemessenen Pseudoentfernungen. Die Abweichungen der jeweils gemessenen Pseudoentfernungen zu den einzelnen Satelliten wird in den folgenden Formeln ausgedrückt:

$$\Delta\rho_1 = \rho_1 - r_1^{'} \tag{2.11}$$

$$\Delta\rho_2 = \rho_2 - r_2^{'} \tag{2.12}$$

$$\Delta\rho_3 = \rho_3 - r_3^{'} \tag{2.13}$$

$$\Delta\rho_4 = \rho_4 - r_4^{'} \tag{2.14}$$

$$zusammengefasst : \Delta\rho_i = \rho_i - r_i^{'} \tag{2.15}$$

Diese Abweichungen lassen auch die Koordinaten unseres Standortes von denen des Näherungspunktes divergieren. In den nachstehenden Formeln werden die Differenzen zwischen den angenommenen Koordinaten und denen unseres realen Standortes beschrieben:

$$\Delta x_p = x_p - x_p^{'} \tag{2.16}$$

$$\Delta y_p = y_p - y_p^{'} \tag{2.17}$$

$$\Delta z_p = z_p - z_p^{'} \tag{2.18}$$

Auch bei unserer angenommenen Fehlergröße k' ist eine Diskrepanz zur eigentlichen Größe k zu beachten:

$$\Delta k = k - k^{'} \tag{2.19}$$

Die in den Gleichungen 2.16 bis 2.18 beschriebenen Differenzen zwischen unseren Näherungskoordinaten und unserem Standort haben alle unterschiedlichen Einfluss auf die Abweichungen von den einzelnen Pseudoentfernungen. Die Diskrepanz von unserer Fehlergröße k (Gleichung 2.19) hat ebenfalls Einfluss auf die Abweichung der einzelnen Pseudoentfernungen, ist aber bei allen Pseudoentfernungen gleich groß.

$$\Delta\rho_1 = a_1\Delta x_p + b_1\Delta y_p + c_1\Delta z_p + \Delta k \tag{2.20}$$

$$\Delta\rho_2 = a_2\Delta x_p + b_2\Delta y_p + c_2\Delta z_p + \Delta k \tag{2.21}$$

$$\Delta\rho_3 = a_3\Delta x_p + b_3\Delta y_p + c_3\Delta z_p + \Delta k \tag{2.22}$$

$$\Delta\rho_4 = a_4\Delta x_p + b_4\Delta y_p + c_4\Delta z_p + \Delta k \tag{2.23}$$

Die Faktoren a_i, b_i und c_i beschreiben den Einfluss auf die jeweiligen Abweichungen von den Pseudoentfernungen („gemessene" Strecke zwischen Satellit und Empfänger), und werden deshalb auch als Streckenkoeffizienten bezeichnet. Sie lassen sich aus den aktuellen Positionskoordinaten des jeweiligen Satelliten, den Koordinaten unseres angenommenen Näherungsstandortes, sowie der Näherungsentfernung, berechnen. Es ergeben sich also bei vier Satelliten zwölf Formeln (i=1,2,3,4):

$$a_i = -\frac{x_i - x_p^{'}}{r_i^{'}} \tag{2.24}$$

$$b_i = -\frac{y_i - y_p^{'}}{r_i^{'}} \tag{2.25}$$

$$c_i = -\frac{z_i - z_p^{'}}{r_i^{'}} \tag{2.26}$$

Um später die einzelnen Koordinaten bestimmen zu können, müssen wir erst Gleichung 2.20 - 2.23 nach den Abweichungen der einzelnen Koordinaten von unseren Näherungskoordinaten auflösen. Zur besseren Übersicht stellen wir das

Gleichungssystem zunächst einmal in Matrixform dar:

$$\begin{pmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \Delta\rho_3 \\ \Delta\rho_4 \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & c_1 & 1 \\ a_2 & b_2 & c_2 & 1 \\ a_3 & b_3 & c_3 & 1 \\ a_4 & b_4 & c_4 & 1 \end{pmatrix} \begin{pmatrix} \Delta x_p \\ \Delta y_p \\ \Delta z_p \\ \Delta k \end{pmatrix} \tag{2.27}$$

Mit der Koeffizientenmatrix A und dem Variablenvektor Δx ergibt sich in Kurzform folgende Gleichung:

$$\Delta\rho = A\Delta x \tag{2.28}$$

Durch Multiplikation der Gleichung mit der Inversen der Matrix A entsteht die Gleichung:

$$A^{-1}\Delta\rho = \Delta x \tag{2.29}$$

$$\text{bzw. } \Delta x = A^{-1}\Delta\rho \tag{2.30}$$

Schreibt man diese Gleichung wieder ausführlich auf, erhalten wir folgende Matrix-Form:

$$\begin{pmatrix} \Delta x_p \\ \Delta y_p \\ \Delta z_p \\ \Delta k \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & c_1 & 1 \\ a_2 & b_2 & c_2 & 1 \\ a_3 & b_3 & c_3 & 1 \\ a_4 & b_4 & c_4 & 1 \end{pmatrix}^{-1} \begin{pmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \Delta\rho_3 \\ \Delta\rho_4 \end{pmatrix} \tag{2.31}$$

Die Ergebnisse die man so für Δx_p, Δy_p, Δz_p und Δk bekommt setzt man nun in die Gleichungen 2.16 bis 2.18 und 2.19 ein, die man zuvor nach x_p, y_p, z_p und k auflöst:

$$x_p = x_p^{'} + \Delta x_p \tag{2.32}$$

$$y_p = y_p^{'} + \Delta y_p \tag{2.33}$$

$$z_p = z_p^{'} + \Delta z_p \tag{2.34}$$

$$k = k^{'} + \Delta k \tag{2.35}$$

Falls die Koordinaten der gewählten Näherungsposition zu weit entfernt von den tatsächlichen Koordinaten gewählt wurden, wird der Algorithmus wiederholt. Als Eingabeparameter werden nun die Koordinaten und die Fehlergröße k des vorherigen Durchlaufs verwendet. Das ganze Verfahren wird so oft wiederholt, bis sich die Positionskoordinaten und der Fehlerparameter nicht mehr ändern.

Es ist gut zu erkennen, dass mindestens vier Pseudoentfernungen benötigt werden um die drei Koordinaten und den Fehlerparameter k zu bestimmen. Sind jedoch mehr Satelliten verfügbar, bzw. kann man mehrere Pseudoentfernungen messen, erhalten wir zwar zusätzliche Gleichungen, aber die Anzahl der Unbekannten bleibt gleich. Die Lösung des Gleichungssystems erfolgt in ähnlicher Weise, führt aber zu genaueren Ergebnissen und verringert die Anzahl der Durchläufe.

Kalman-Filter Ein weiteres Verfahren zur Positionsbestimmung verwendet einen sogenannten Kalman-Filter. Der hier angewandte Algorithmus ist ein digitaler Algorithmus, der auf dem Prinzip der Rekursion beruht. Mit Hilfe des Kalman-Filters wird auf stochastischer Basis der Zustand eines Systems (Status) geschätzt. Der Filter hat eine sog. Prädiktor-Korrektor-Struktur, d.h. zunächst wird auf Grund der Systemeingangsdaten der wahrscheinlichste Ausgangswert (in unserem Fall die aktuelle Position) vorhergesagt (prädiziert) und dieser dann mit dem tatsächlich gemessenen Ausgangswert (in unserem Fall die gemessenen Pseudoentfernungen) verglichen. Die Differenz der beiden Werte wird gewichtet und dann zur Verbesserung (Korrektur) des aktuellen Zustandes verwendet[48]. Der Status unseres Systems bezieht sich auf die Position, die Entfernung zu den einzelnen Satelliten, die GPS-Zeit und die Geschwindigkeit des GPS-Gerätes.

In Abbildung 2.23 ist der grundsätzliche Verlauf von Informationen eines Kalman-Filters in einem GPS-Gerät dargestellt. Diesen Ablauf, der sich ab der 2. Phase immer wiederholt, kann man grob in fünf Phasen unterteilen:

1. Phase Die erste Phase ist die Initialisierung. Hier werden die Anfangswerte der zentralen Daten des Kalman-Filters geschätzt. Um eine grobe Statusschätzung abzugeben werden die Statuswerte der letzten Aktivität des Gerätes, die sich noch im Speicher befinden, verwendet. Dieser absolute Zustandsvektor $\vec{x}(0)$ sollte realitätsnahe Werte besitzen, um so ein Divergieren des Filters zu vermeiden. Zudem wird das Zustandsinkrement mit Nullen initialisiert:

$$\delta\vec{x}(0) = 0 \tag{2.36}$$

[48] vgl. [Man10]:161

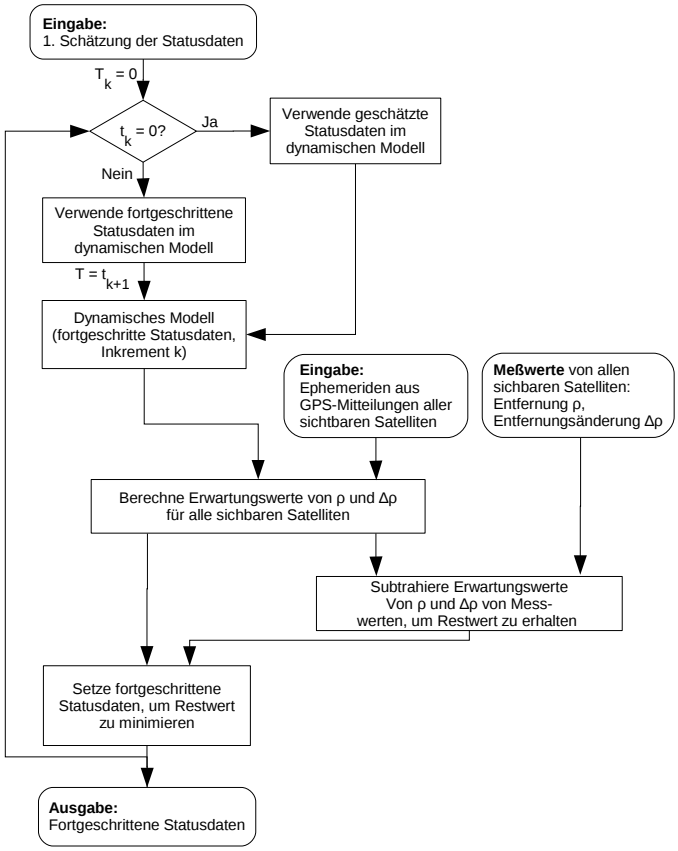


Abbildung 2.23: Datenfluss in einem GPS-Gerät unter Verwendung des Kalman-Filters ([Man10]:162 verändert)

Des Weiteren werden zeitvariable Zustandsgrößen $\vec{x}(t)$ geschätzt, sowie deren Kovarianzmatrix $\Sigma(t)$ initialisiert indem man ihre Hauptdiagonale mit großen Werten G_i belegt.

$$\Sigma(0) = \begin{pmatrix} G_1 & 0 & \cdots & 0 \\ 0 & G_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & G_n \end{pmatrix} \tag{2.37}$$

Die Kovarianzmatrix $\Sigma(t)$ beinhaltet die Abhängigkeit der einzelnen Zufallsgrößen (wie z.B Rauschen, Refraktion, usw.) voneinander. Da auf der Hauptdiagonalen die Abhängigkeit einer Zufallsgröße von sich selbst darstellt wird, wird diese mit einem entsprechend hohem Wert belegt. Mit Hilfe des Kalman-Filters werden aus „alten“ Statusdaten neue ermittelt. Bei der Initialisierung werden die „alten“ Statusdaten (hier mit einem Minus gekennzeichnet) mit den

Statusdaten zum Zeitpunkt 0 gleichgesetzt:

$$\vec{x}(-) = \vec{x}(0) \qquad \delta\vec{x}(-) = \delta\vec{x}(0) \qquad \Sigma(-) = \Sigma(0) \tag{2.38}$$

2. Phase In der 2. Phase überprüft das GPS-Gerät, ob es schon im Betrieb ist und der Kalman-Filter bereits durchlaufen wurden. Ist das Gerät gerade erst eingeschaltet worden, werden als Ausgangspunkt für das dynamische Modell die bei der Initialisierung geschätzten Daten verwendet. Hat das Gerät schon eine Rekursion durchlaufen, werden die fortgeschrittenen Daten aus dem dynamischen Modell verwendet.

3. Phase Innerhalb der dritten Phase werden mit Hilfe der Stochastik verschiedene Erwartungswerte berechnet. Als Datengrundlage dienen die Statusdaten des letzten Berechnungszeitpunktes (z.B. Position, Richtung, Geschwindigkeit), aktuelle Bahndaten (Ephemeriden) der einzelnen Satelliten, die aus den empfangenen Navigationsnachrichten gewonnen werden, sowie die zuletzt verwendeten Zufallsgrößen. Aus diesen Größen werden, unter Verwendung von Schätzwerten für die mit der Zeit fortgeschrittene Positionsänderung, Erwartungswerte sowohl für die Entfernung ρ, als auch für die Entfernungsänderung $\Delta\rho$ zu den einzelnen Satelliten für den kommenden Auswertungszeitpunkt, berechnet.

4. Phase In der vierten Phase werden die Entfernungen, bzw die Entfernungsänderungen, zu den einzelnen Satelliten von denen ein Signal empfangen wird, gemessen. Dabei werden zur Berechnung die Verfahren der Codemessung[49], des Doppler-Counts[50] und der Träger-/Phasenmessung[51] eingesetzt. Die so ermittelten Messwerte werden mit den in der dritten Phase berechneten Erwartungswerten verglichen. Davon werden die Erwartungswerte der Messungen dieser Phase abgezogen. Die durch die Differenz gewonnen Werte werden allgemein auch als Restwerte bezeichnet. Sind Messwerte und Erwartungswerte genau gleich, erhalten wir einen Restwert von Null. Dies tritt in der Regel aber nicht auf.

5. Phase In der letzten Phase des rekursiven Algorithmus' werden die alten Ausgangsdaten um den Restwert korrigiert. Die in der vierten Phase ermittelten Restwerte bringen den Fehler, der in der dritten Phase berechneten Erwartungs-

[49] siehe Kapitel 2.2.4.1
[50] siehe 2.2.5.1
[51] siehe 2.2.5.2

Eigenschaften	GPS	Inertial Systeme
Unabhängigkeit	–	✓
Verlässlichkeit	gering	hoch
Datenrate	gering (um 5Hz)	hoch (um 60 Hz)
Initialisierung	eigenständig	notwendig
Langzeitstabilität	sehr gut	schlecht
Kurzzeitstabilität	akzeptabel	sehr gut

Tabelle 2.3: Gegenüberstellung der Fehlercharakteristik von GPS und INS

werte, zum Ausdruck. Der Filter ändert daraufhin seine Schätzwerte um den Restwert

$$\Delta\vec{x}(+) \leftarrow \Delta\vec{x}(-) \qquad \Sigma(+) \leftarrow \Sigma(-) \tag{2.39}$$

Zum Schluss werden die geänderten, geschätzten Statusdaten auf das dynamische Modell zurückgekoppelt[52]. Damit wird der Prozess des Schätzens mit neuen, aktualisierten Statuswerten durchlaufen und führt zur Rekursion.

Ein Vorteil des Kalman-Filters ist das Arbeiten mit einzelnen Messergebnissen. So kann zum Beispiel für kurze Zeit die Messung der Entfernung zu einem Satelliten ausfallen, durch den berechneten Erwartungswert diese Lücke aber kurzfristig überbrückt werden. Ein weiterer Vorteil des Kalman-Filters ist die Möglichkeit der Integration eines Inertial-Navigationssystems (INS).

Das Funktionsprinzip inertialer Navigationssysteme beruht auf der Koppelnavigation (Dead-Reckoning). Die Positionsänderung wird hierbei aus der zurückgelegten Strecke mit der entsprechenden Orientierung berechnet. Zum Messen von Strecke und Winkel werden verschiedensten Sensoren verwendet.

GPS ist von der Satellitensichtbarkeit abhängig und kann durch verschiedene physikalische Gegebenheiten nicht ständig Daten liefern (Datenraten eines GPS-Gerätes liegen zwischen 0,1 und 10 Hz). Weitere Nachteile sind der systembedingt geringe Dynamikbereich und die Störanfälligkeit. Inertiale Systeme hingegen liefern hohe Datenraten und eine große Dynamik. Bei Messungen über kurze Zeiträume erreichen sie eine relativ hohe Genauigkeit. Durch das sogenannte Driften von Sensoren vergrößern sich die Fehler bei INS zunehmend. Zu Beginn einer Messung muß das System stets mit dem aktuellen Startort initialisiert werden. In Tabelle 2.3 sind die Eigenschaften der beiden Systeme noch einmal gegenübergestellt.

[52] Rückkopplung in Phase 2, vgl. Abbildung 2.23

Wenn man sich die Tabelle 2.3 einmal genau anschaut, kann man feststellen, dass die Stärken des einen Systems genau da liegen, wo das andere System Schwächen aufweist. Zum Beispiel kann GPS die Initialisierung von INS mit dem aktuellen Standort unterstüzen. Des Weiteren kann GPS die langfristig entstehenden Fehler (durch Drift der Sensoren) bei INS immer wieder zurücksetzen. Inertial Systeme können hingegen den „langen“ Zeitraum zwischen zwei GPS-Ortungen mit ihren kurzzeitig recht genauen Daten überbrücken. Auch die Daten von INS sind gegenüber GPS rauschfrei und nicht von außen störbar. Durch diese unterschiedlichen Fehlercharakteristiken ist eine Integration von GPS und INS mittels Kalman-Filter eine nahezu ideale Lösung. In Abbildung 2.24 ist das Prinzip dieser Kombination dargestellt.

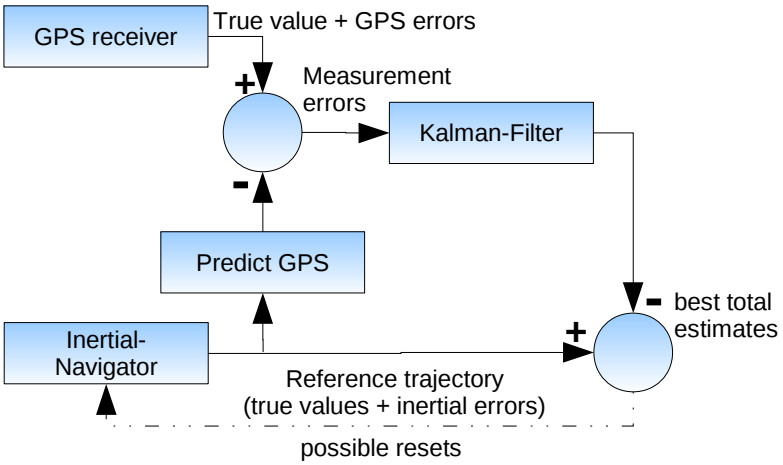


Abbildung 2.24: Prinzip der Integration von GPS und INS unter Verwendung des Kalman-Filters

2.2.5.1 Doppler-Count

Wie schon in Kapitel 2.2.4.1 angedeutet, ist die empfangene Frequenz nicht gleich der Frequenz, mit der die Satelliten die Signale aussenden. Wenn sowohl der Sender als auch der Empfänger stationär wären, wäre die Sendefrequenz gleich der Empfangsfrequenz. Im Falle von GPS sind zumindest die Satelliten, welche die Signale aussenden, immer in Bewegung. Die GPS-Empfänger sind zum Beispiel bei Vermessungen zwar stationär, aber im Navigationseinsatz zumeist auch selbst in Bewegung.

Wenn sowohl Sender als auch Empfänger stationär sind, ist der Abstand konstant und sie besitzen keine Relativgeschwindigkeit zueinander. Sobald sich allerdings eines der beiden Geräte relativ gesehen zum anderen in Bewegung be-

findet ändert sich ihr Abstand zueinander permanent und es entsteht zwischen den beiden eine Relativgeschwindigkeit. Somit besitzt das empfangene Signal eine von der Sendefrequenz abweichende Frequenz. Dieses Phänomen wird in der Physik auch als Doppler-Effekt bezeichnet.

Betrachten wir diese Erscheinung einmal genauer. Ein sich bewegender Sender S sendet zur Zeit $t = t_0$ ein Signal mit der Frequenz f aus. Die Schwingungsdauer beträgt T, also ist die Frequenz $f = \frac{1}{T}$. Innerhalb einer Schwingungsdauer T bewegt sich der Sender S dem Singnal „hinterher“ , was bedeutet, dass sich der Sender auf den Empfänger zubewegt. Die Strecke, die er während dieser Zeit zurücklegt, wird als Δs bezeichnet (zur Verdeutlichung siehe Abbildung 2.25).

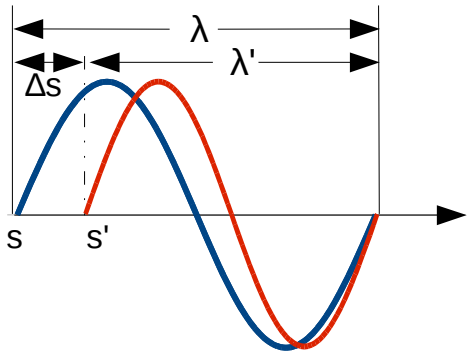


Abbildung 2.25: Illustrierung des Doppler-Effekts

Die Geschwindigkeit, mit der sich der Sender S dem Empfänger nähert, beträgt demnach:

$$v = \frac{\Delta s}{T} \tag{2.40}$$

Erreichen das ursprüngliche Signal und das dopplerverschobene Signal wieder den Anfangszustand, besitzen sie die gleiche Phase. Zu diesem Zeitpunkt hat sich die ursprüngliche Gesamtlänge, die der Wellenlänge λ entspricht, um den Betrag Δs verringert. Die sich daraus ergebende Wellenlänge des empfangenen Signals berechnet sich folgendermaßen:

$$\lambda^{'} = \lambda - \Delta s \tag{2.41}$$

Stellt man nun die Gleichung 2.40 nach Δs um, setzt dies dann in Gleichung 2.41 ein und ersetzt T durch $\frac{1}{f}$, erhält man folgende Gleichung für die Wellenlänge des empfangenen Signals:

$$\lambda^{'} = \lambda - \frac{v}{f} \tag{2.42}$$

Die allgemeine Formel für die Wellenlänge setzt sich aus der Ausbreitungsgeschwindigkeit der Welle und ihrer Frequenz zusammen. In unserem Fall breitet sich die Welle mit annährend Lichtgeschwindigkeit aus, sodass mit der Konstanten der Lichtgeschwindigkeit c folgende Gleichung entsteht:

$$\lambda = \frac{c}{f} \tag{2.43}$$

Setzt man dies in die Gleichung 2.42 ein, und klammert den Faktor $\frac{c}{f}$ aus, erhalten wir:

$$\lambda' = \frac{c}{f} - \frac{v}{f} = \frac{c}{f} * (1 - \frac{v}{c}) \tag{2.44}$$

Für λ' setzen wir nun nach Gleichung 2.43 analog $\frac{c}{f'}$ ein. Um jetzt die Verschiebung der Frequenz f auf f' zu bekommen, lösen wir nach der Frequenz f' auf und erhalten eine Formel, die sowohl von der Sendefrequenz f als auch von der Geschwindigkeit v abhängt:

$$\frac{c}{f'} = \frac{c}{f} * (1 - \frac{v}{c}) \tag{2.45}$$

$$f' = f * \frac{1}{(1 - \frac{v}{c})} \tag{2.46}$$

Da die Geschwindigkeit des Senders v gegenüber der Ausbreitungsgeschwindigkeit der elektromagnetischen Welle c sehr klein ist, kann man näherungsweise für die Gleichung 2.46 auch folgende Gleichung formulieren[53]:

$$f' = f(1 + \frac{v}{c}) \tag{2.47}$$

Die Differenz der Frequenz des empfangenen Signals f' und der Frequenz des ausgesendeten Signals f wird auch als Doppler-Frequenzverschiebung bezeichnet, oder kurz Doppler-Frequenz f_d:

$$f_d = f' - f \tag{2.48}$$

$$f_d = f(1 + \frac{v}{c}) - f \tag{2.49}$$

$$f_d = f + (f\frac{v}{c}) - f \tag{2.50}$$

$$f_d = f\frac{v}{c} \tag{2.51}$$

Eine exakte Betrachtung des Doppler-Effekts ist nur dann möglich, wenn relativistische Einflüsse Berücksichtigung finden. Diese Betrachtung würde allerdings

[53] näheres dazu im Anhang A.5.1

den Umfang dieser Ausarbeitung übersteigen, weshalb ich an dieser Stelle auf einschlägige Literatur der theoretischen Physik verweisen möchte (z.B. [Fay02]). In der Praxis wird näherungsweise mit der folgender Formel gearbeitet, die sich nur um eine Berücksichtigung des Winkels γ von unserer Gleichung 2.51 unterscheidet[54]:

$$f_d = f\frac{v\cos\gamma}{c} \tag{2.52}$$

Setzt man in diese Formel den aktuellen Winkel γ (welchen man aus den GPS-Mitteilungen der einzelnen GPS-Satelliten ermitteln kann), die bekannte Sendefrequenz f und die Doppler-Frequenz f_d (welche mit Gleichung 2.48 und Messung der empfangenen Frequenz berechnet werden kann) ein, kann man die Relativgeschwindigkeit v zwischen Satellit und Empfänger bestimmen:

$$v = \frac{c}{f\cos\gamma}f_d \tag{2.53}$$

Da man aus den empfangenen GPS-Mitteilungen auch die Geschwindigkeit der einzelnen Satelliten ermitteln kann, ist es möglich daraus auch auf die Geschwindigkeit des GPS-Empfängers zu schließen. Dies ist mit einer Genauigkeit von rund 0,18km/h möglich (vgl. [SW07]).

Auch die Änderung der Entfernung zwischen Satellit und GPS-Empfänger kann mit Hilfe des Doppler-Effekts bestimmt werden. Hierzu werden die Wellenzyklen der Dopplerfrequenz zwischen zwei Zeitpunkten ermittelt. Bei einer konstanten Frequenz berechnet man die Anzahl der Wellenzyklen (Schwingungen) aus $f = \frac{n}{t}$ mit:

$$n = f * t \tag{2.54}$$

Da sich bei GPS die Satelliten und Empfänger meistens nicht gleichmäßig bewegen, wird das Signal nicht mit einer konstanten Frequenz empfangen, sondern mit einer sich im Normalfall ständig ändernden Doppler-Frequenz. In Abbildung 2.26 ist der Verlauf einer Dopplerfrequenz-Funktion (f_{ds}) eines stationären Empfängers zu sehen. Ein Ansteigen der Funktion bedeutet die Erhöhung der Dopplerfrequenz, d.h. der Sender bewegt sich immer schneller auf den Empfänger zu (Abschnitt T_1 bis T_2). Ist die Geschwindigkeit des Senders konstant, so ist auch die Funktion der Dopplerfrequenz konstant (Abschnitt T_2 bis T_3). In Abschnitt T_3 bis T_4 verringert sich die Geschwindigkeit immer weiter bis

[54] Der Winkel γ wird zwischen der Geraden von Satellit und Empfänger, und dem Geschwindigkeitsvektor des Satelliten aufgespannt

zum Zeitpunkt T_4, ab dem die Geschwindigkeit Null und damit die empfangene Frequenz gleich der gesendeten Frequenz ist.

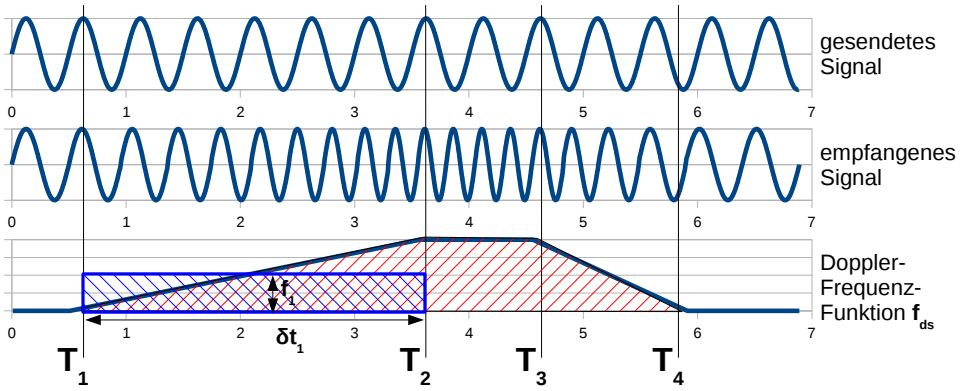


Abbildung 2.26: Doppler-Funktion

Um nun die Anzahl der Wellenzyklen innerhalb eines Zeitraums näherungsweise bestimmen zu können, wird dieser Zeitraum in mehrere Zeitintervalle unterteilt. Zur Berechnung der Anzahl der Wellenzyklen wird die mittlere Frequenz des Zeitintervalls verwendet und nach Gleichung 2.54 mit der Länge des Intervalls multipiziert. Diese Vorgehensweise verdeutlicht das blau schraffierte Rechteck in Abbildung 2.26. f_1 ist die mittlere Frequenz des Zeitintervalls δt_1. Somit ist die Anzahl der Schwingungen im Intervall δt_1:

$$n_1 = \delta t_1 * f_1 \tag{2.55}$$

Addiert man die Anzahl der Wellenzyklen aller Abschnitte zusammen, erhält man näherungsweise die Anzahl der Schwingungen im gesamten Zeitraum von T_1 bis T_m:

$$n = n_1 + n_2 + ... + n_m \tag{2.56}$$

$$n = \delta t_1 * f_1 + \delta t_2 * f_2 + ... + \delta t_m * f_m \tag{2.57}$$

mit f_i: mittlere Frequenz im Zeitintervall δt_i

Lässt man nun die Intervalle immer kleiner werden und deren Länge gegen Null laufen, so erhalten wir genau das aus der Analysis bekannte Integral der Funktion (rot schraffiertes Rechteck). Somit ist das Integral der Dopplerfrequenz-Funktion f_{ds} näherungsweise gleich mit der Anzahl der Wellenzyklen und wird

auch als integrierter Doppler-Count C_d zwischen T_1 und T_m bezeichnet:

$$\text{für } \delta t_i \to 0 \qquad C_d = \int_{T_1}^{T_m} f_{ds} dt \tag{2.58}$$

C_d ist ein Maß für die Entfernungsdifferenz zwischen dem Empfänger und zwei Bahnpositionen eines Satelliten zu zwei verschiedenen Zeitpunkten T_1 und T_m. Mit $\rho = \lambda * n$ kommt man mit dem integrierten Doppler-Count und der Dopplerfrequenz-Funktion f_{ds} auf folgende Gleichung zur Berechnung der Entfernungsdifferenz:

$$\Delta\rho = \frac{c}{f} * \left[C_d - \underbrace{f_{ds} * (T_m - T_1)}_{\text{genäherte Anzahl von Wellenzyklen}} \right] \tag{2.59}$$

Mit Hilfe der ermittelten Differenzen der vier Satelliten kann dann die aktuelle Empfängerposition berechnet werden.

2.2.5.2 Trägerphasenmessung

Bei dem Verfahren der Trägerphasenmessung wird zur Ermittlung der Entfernung zwischen einem Satellit und dem GPS-Empfänger nicht die Laufzeit des Signals verwendet, sondern es werden die Eigenschaften der Welle genutzt, die das GPS-Signal transportiert. Wie in Abschnitt 2.2.5.1 schon erwähnt, kann man durch Kenntnis der Frequenzen der Tägerwellen L1 und L2, die Wellenlänge eines Wellenzyklus von L1 und L2 mit $\lambda = \frac{c}{f}$ berechnen. Nun versucht man zu ermitteln, wie viele Wellenzyklen der Trägerwelle zwischen einem Satelliten und dem Empfänger liegen. Mit Hilfe der Trägerphasenmessung kann die Entfernung im Millimeterbereich genau bestimmt werden. Da bei diesem Verfahren komplexe mathematische Algorithmen zur Anwendung kommen, die aber im weiteren Verlauf dieser Diplomarbeit keine Rolle spielen sollen, werde ich an dieser Stelle nur kurz auf die Idee des Prinzips eingehen[55].

Auch bei diesem Messverfahren wird im Empfänger ein Referenzsignal erzeugt. Die eigentliche Beobachtungsgröße ist ein niederfrequentes Mischsignal, welches durch Überlagerung des intern generierten Referenzsignals mit dem empfangenen Signal gebildet wird. Durch diese Phasendifferenzbildung ist die Auswertung wesentlich einfacher. Die Phasendifferenz Φsetzt sich folgendermaßen

[55] für eine genauere Beschreibung des Verfahrens vergleiche [KH06]

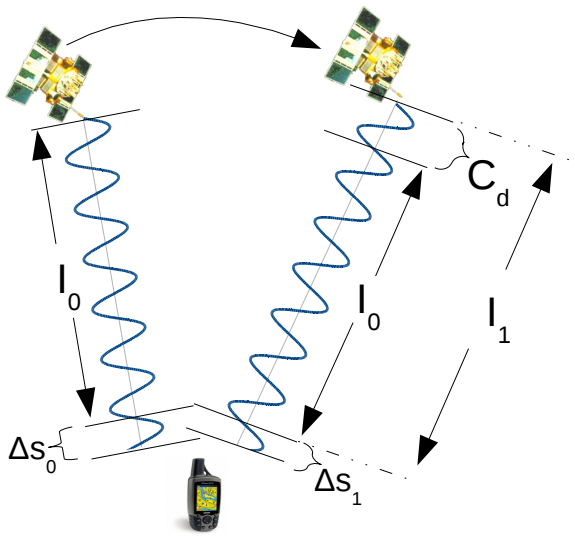


Abbildung 2.27: Trägerphasenmessung

zusammen:

$$\Phi(t) = \Phi_r(t) - \Phi_e(t - \delta t) \tag{2.60}$$

Hierbei ist δt der Uhrenfehler, der zwischen Satellit und Empfänger besteht und durch eine Nullphasenverschiebung korrigiert wird und Φ_r die Phase des Referenzsignals bzw. Φ_e die des empfangenen Signals . Der Phasenunterschied ist $\Delta\varphi = \Phi(T_1) - \Phi(T_2)$ und steht mit der in dieser Zeit zurückgelegten Strecke Δs und der Wellenlänge der Trägerfrequenz in folgendem Verhältnis:

$$\frac{\Delta\varphi}{2\pi} = \frac{\Delta s}{\lambda} \tag{2.61}$$

Da die Phase des Signals sich im Abstand einer Wellenlänge λ (bzw. von 2π) immer wiederholt, kann man nur die Strecke innerhalb einer Wellenlänge λ bestimmen. Wie oft sich die Phase bis zum Eintreffen beim Empfänger wiederholt, ist unbekannt. Man bezeichnet dies auch als Phasenmehrdeutigkeit oder Ambiguity (vgl. hierzu Abbildung 2.27). Um dieses Faktum in die Gleichung 2.61 einzubringen, wird der Gleichung der Mehrdeutigkeitsfaktor n hinzugefügt.

$$\frac{\Delta\varphi + n2\pi}{2\pi} = \frac{\Delta s + n\lambda}{\lambda} \qquad , mit \;\; n \in \mathbb{N}_0 \tag{2.62}$$

Die Entfernung s zwischen Satellit und Empfänger setzt sich aus der Länge des gemessenenen Reststücks der Welle Δs und der Länge der vollen Wellenzüge $n\lambda$ zusammen ($s := \Delta s + n\lambda$), sodass man durch Umformen von Gleichung 2.62

zu folgender Formel zur Entfernungsberechnung kommt:

$$s = \Delta s + n\lambda = \frac{\lambda}{2\pi}(\Delta\varphi + n2\pi) \tag{2.63}$$

Das Problem an dieser Formel ist die Bestimmung der Phasenmehrdeutigkeit (Ambiguity), bzw. des Phasenmehrdeutigkeitsfaktors n. Die Berechnung der Anfangsmehrdeutigkeit I_0, auch Initialisierung genannt, ist eine sehr komplexe mathematische Aufgabe, die mit Hilfe von stochastischen Verfahren bestimmt wird[56]. Die darauf folgenden Phasenmehrdeutigkeitsfaktoren I_i der kommenden Messzeitpunkte setzen sich aus der vorherigen Mehrdeutigkeit I_{i-1} und dem integrierten Doppler-Count zwischen den beiden Messzeitpunkten aus Kapitel 2.2.5.1 zusammen und betragen $I_i = I_{i-1} + C_d$ (vergleiche hierzu Abbildung 2.27).

2.2.6 Genauigkeit und mögliche Fehlerquellen

Eine entscheidende Frage ist, welche Genauigkeit mit GPS erreicht werden kann bzw. wie groß auftretende Positionsfehler sind. Wenn man über einen längeren Zeitraum einen GPS-Empfänger an einem festen Ort installiert, stellt man fest, dass die ausgegebenen Positionskoordinaten leichten Abweichungen unterliegen (Anhang A.1, Abbildung A.2). Ursache dafür können mehrere Fehlerquellen sein. Allgemein kann man zwischen den folgenden drei Fehlerarten unterscheiden:

- **zufällige Fehler**
- **systematische Fehler**
- **grobe Fehler**

Zufällige Fehler treten meist als Folge aktueller Messbedingungen auf. Dazu zählen unter anderem die Wetterlage, der Standort (z.B. in einem dichten Wald, zwischen Hochhäusern) und andere, vom System nicht vorhersehbare, Umstände. Diese Fehler sind nur zum Teil korrigierbar. Hingegen sind **systematische Fehler** sehr gut zu korrigieren, da diese nachvollziehbar und reproduzierbar sind. Sie sind grundsätzlich physikalischer und/oder gerätetechnischer Natur. Die Korrektur wird während der Datenverarbeitung durchgeführt. **Grobe Fehler** sind relativ selten und können zum Beispiel durch unsachgemäße Handha-

[56] dies wären z.B die Lambda-Methode, Doppeldifferenzen oder die Ambiguity Function, welche in der gängigen Fachliteratur zu finden und nachzulesen sind (z.B. [CCDF00])

bung eines GPS-Empfängers verursacht werden. Sie sind also nutzerbedingt bzw. durch Fehlkonstruktionen bedingt.

Speziell auf GPS bezogen unterteilt Mansfeld ([Man10]:171) die Fehler weiter:

- Fehler bei der Messung der Pseudoentfernung im Signalweg mit den Anteilen:
  - Satelliten
  - Ausbreitungsweg und Mehrwegeausbreitung
  - Empfänger
- Fehler durch äußere Einflüsse
  - Interferenz
- Fehler bei der Positionsbestimmung infolge geometrischer Verhältnisse
- Fehler bei der Geschwindigkeitsbestimmung
- Fehler bei der Ermittlung der Zeitinformationen

Die auf Grund der Satellitendaten fehlerhaft ermittelten Pseudoentfernungen basieren im wesentlichen auf zwei Aspekten. Zum Einen sind dies die von den Satelliten ausgesendeten Ephemeriden (Bahndaten). Die Ephemeriden für jeden Satellit werden jeweils für eine gesamte Woche vorausberechnet. Diese Berechnungen sind allerdings nur bis zu einem bestimmten Grad zutreffend. Durch permanente Kontrollmessungen der GPS-Monitorstationen werden Abweichungen festgestellt und die daraus berechneten Korrekturdaten stündlich an die Satelliten gesendet. Die Satelliten wiederum geben diese Informationen mittels der Navigationsmitteilungen an die GPS-Empfänger weiter. Die Abweichungen von den berechneten Ephimeriden (z.B. durch Schwankungen von Gravitationskräften) während einer Stunde bleiben so unberücksichtigt, liegen aber quantitativ unter einem Meter ([Man10]:171). Die zweite Fehlerquelle ist die Abweichung der einzelnen Satelliten von der GPS-Systemzeit; aber auch hieraus resultieren nur kleinere Differenzen.

Ebenfalls relevant ist der Weg, den die elektromagnetische Welle zurücklegen muss. Beim Durchqueren der einzelnen Schichten der Atmosphäre kommt es zu variierenden Laufzeitverzögerungen und unterschiedlichen Brechungswinkeln (Abbildung 2.28). Durch die verschiedenen Brechungswinkel an den einzelnen

Schichten kommt es zu Richtungsänderungen und somit zu fehlerhaften Laufzeitberechnungen. Auch die Laufzeit an sich kann in den unterschiedlichen atmosphärischen Schichten variieren. Den größten Einfluss auf das Signal hat die Ionosphäre. Die durch die Atmosphäre verursachten Fehler (Atmosphärische Effekte oder auch Refraktion) sind, da sie zu den systematischen Fehlern gehören, zu einem großen Teil durch geeignete Korrekturmodelle zu beheben (zum Beispiel durch Vergleich des L1- und L2-Signals).

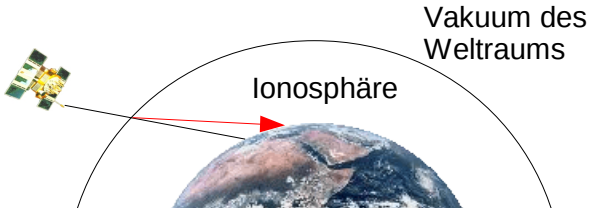
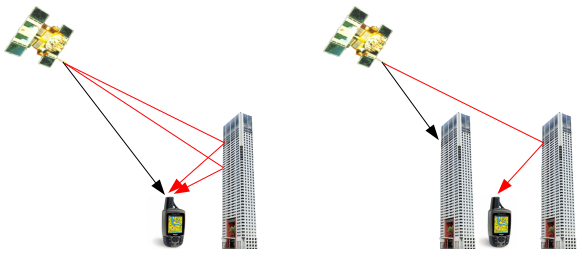


Abbildung 2.28: astronomische Refraktion

Weitere Probleme bei der Positionsbestimmung sind Mehrwegeausbreitung und Abschattung. In Abbildung 2.29 sind diese beiden Effekte illustriert. Die Mehrwegeausbreitung (Multipath) tritt dann auf, wenn die elektromagnetische Welle auf Objekte trifft, die das Signal reflektieren. Dadurch kommt das gleiche Signal zum Beispiel zwei Mal beim Empfänger an. Durch Messung der Signalstärke kann diese Fehlerquelle jedoch zu großen Teilen kompensiert werden. Bei der Abschattung befindet sich ein Objekt zwischen Satellit und GPS-Emfpänger, der Satellit ist also nicht sichtbar[57]. Trifft ein Signal jedoch durch Reflektion doch noch beim Empfänger ein, ist es schwierig dies zu erkennen (aber zum Teil durch die Polarisation des Signals doch möglich).



(a) Mehrwegeausbreitung (b) Abschattung (und Mehrwegeausbreitung)

Abbildung 2.29: Fehlerquellen beim Signalempfang

Auch beim Empfänger selbst sind Fehlerquellen vorhanden. Da zur Übertragung der Informationen die Phasenmodulation genutzt wird, hat die Phasen-

[57] vergleiche zum Begriff „sichtbar“ Kapitel 2.2.7

komponente des Rauschens großen Einfluss auf das zu messende Signal. Die Signallaufzeit im Empfänger selbst und die Ozillatorinstabilitäten müssen bei einem so zeitempfindlichen System wie GPS Berücksichtigung finden.

Fehler durch äußere Einflüsse zählen zu den zufälligen Fehlern. Die Beeinflußung des GPS-Signals kann durch andere hochfrequente Ausstrahlungen in der Nähe geschehen (zum Beispiel durch TV-Sender). Auch sonstige elektrische Anlagen, wie Hochspannungsleitungen, Oberleitungen, Transformatoren in unmittelbarer Umgebung der Antenne des GPS-Empfängers können die Satellitensignale stören. Bis zu einer gewissen Stärke können diese Interferenzen allerdings herausgefiltert werden (über den PRN-Code, siehe Kapitel 2.2.3.1).

Auf die Fehler, die infolge geometrischer Verhältnisse auftreten, wird im folgenden Kapitel 2.2.7 genauer eingegangen. Es handelt sich um die sogenannten DOP[58]-Faktoren. Mit ihrer Hilfe werden Aussagen über die Fehlerwahrscheinlichkeit getroffen, die von der momentanen geometrischen Konstellation der Satelliten abhängt.

Trotz der Synchronisierung (siehe Kapitel 2.2.4.1) der Empfängeruhr mit der GPS-Uhrzeit ist die Zeitkomponente immer noch eine Fehlerquelle. Durch Rundungsfehler bei der Berechnung und durch relativistische Einflüsse kommt es zu geringen Abweichungen, die aber bei einem so zeitempfindlichen System wie GPS dennoch eine Rolle spielen. Die relativistischen Einflüsse treten aufgrund der hohen Geschwindigkeit (3874 $\frac{m}{s}$) und der Flüghöhe (geringeres Gravitationsfeld) der Satelliten auf. Diese beiden Effekte sind zwar gegenläufig, jedoch überwiegt der Einfuss des Gravitationsfeldes, sodass die Uhren im Satelliten für den Betrachter auf der Erdoberfläche schneller zu gehen scheinen. Da dies ein systematischer Fehler ist, kann dem entgegengewirkt werden. Hierzu wird anstatt der theoretischen Frequenz von 10,23 MHz, eine Frequenz von 10,229999995453 Mhz[59] verwendet.

Zuletzt sei noch eine künstliche Fehlerquelle genannt, die sogenannte Selective Availability (SA). Da GPS ursprünglich für militärische Zwecke genutzt wurde, verfälschte man das Signal für die zivile Nutzung. Hierzu wurden der C/A-Code mit einem künstlichen Uhrenfehler versehen und etwas abweichende Ephemeridendaten ausgesendet. Das Kontrollsegment des GPS konnte so die Genauigkeit maßgeblich beeinflussen. Die militärischen Nutzer konnten trotz des fehlerhaften Signals über Informationen, die per P/Y-Code versendet wurden, eine korrekte Ortung durchführen. Mit aktivierter SA verschlechtert sich

[58] Dilution of Precision
[59] [Emb06]

die Ortung um bis zu 100m. Außerdem ist eine Höhenbestimmung mittels GPS nur mit ausgeschalteter SA möglich. Die Selective Availability war bis ins Jahr 2000 die größte Fehlerquelle für den zivilen Nutzer des SPS (Standard Positioning Service). Seit dem 2. Mai 2000, 5:05 Uhr (MEZ) ist SA bis auf weiteres deaktiviert[60].

Abschließend wird in Tabelle 2.4 eine Übersicht über den Einfluss der verschiedenen Fehlerquellen gegeben. Im Bereich des Raum- und Kontrollsegments sind die entstehenden Abweichungen bei C/A-Code und P/Y-Code gleich. Erst im Bereich des Nutzersegments sind Unterschiede zwischen den beiden Codes zu erkennen. Die größte Diskrepanz entsteht durch die Laufzeitverzögerung der Signale in den verschiedenen Schichten der Atmosphäre.

Segment	**Fehlerquelle**	**C/A-Code** in Metern	**P/Y-Code** in Metern
Raumsegment	Satelliten-Instabilität	3,0	3,0
	Satellitenbahn-Störungen	1,0	1,0
	andere Ursachen (z.B. Sonnenwind)	0,5	0,5
Kontrollsegment	Fehler in vorausgesagten Ephemeriden	4,2	4,2
	Messfehler der Monitorstationen	0,9	0,9
Nutzersegment	Laufzeitverzögerung		
	-Ionosphäre	8,0	1,2
	-Troposphäre	2,5	1,5
	Empfängerrauschen, Messauflösung	1,5	1,5
	Mehrwegeausbreitung	2,5	1,2
	Interferenzen	0,5	0,5
gesamtes System	quadratischer Mittelwert	10,3	6,0

Tabelle 2.4: Fehlerbilanz für die Messung der Pseudoentfernung mit Hilfe des C/A-Codes (SPS) und des P/Y-Codes (PPS), Quelle: [Man10]:193

2.2.7 Sichtbarkeit und Verfügbarkeit

Im Zusammenhang mit GPS fallen immer wieder die beiden Begriffe Sichtbarkeit und Verfügbarkeit. Der Begriff der Sichtbarkeit (englisch: visibility) bezieht sich auf einen einzelnen GPS-Satelliten und den Ort, an dem sich der GPS-Empfänger befindet. Ein Satellit gilt für den GPS-Empfänger als sichtbar, wenn die direkte Verbindungslinie zwischen dem Satelliten und dem Empfänger frei von elektromagnetischen Hindernissen ist. Wäre die Erde ein perfekte geometrische Kugel könnten wir einen Satelliten oberhalb der Horizontalebene als sichtbar bezeichnen, hätten also einen Sichtbarkeitswinkel von $\alpha = 180°$ (Abbildung 2.30).

[60] http://army-gps.robins.af.mil/Announcements/sa-off_press_release.htm

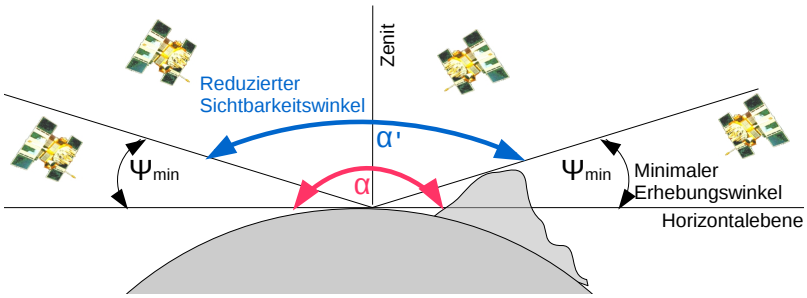


Abbildung 2.30: Sichtbarkeit und Sichtbarkeitswinkel von GPS-Satelliten

Da aber die Erde durch ihre Rotation an sich schon keine perfekte Kugel ist, und weiterhin die Sichtbarkeit durch verschiedene Objekte auf der Erdoberfläche wie Berge, Wälder und Gebäude gestört ist, beträgt der Sichtbarkeitswinkel zumeist weniger als 180°. Auch Satelliten, die sich dicht oberhalb der Horizontalebene befinden, haben sich als problematisch herausgestellt. Durch die größere Entfernung[61] tritt eine stärkere, durch die Atmosphäre verursachte, Laufzeitverzögerung auf. Um diese nachteiligen Einflussfaktoren zu vermeiden, werden nur Satelliten oberhalb eines gewissen Erhebungswinkels zur Ortung verwendet. Die Satelliten müssen sich also über dem minimalen Erhebungswinkel ψ_{min} (Mask Angle) oberhalb der Horizontalebene befinden, was zu einem reduzierten Sichtbarkeitswinkel $\alpha^{'}$ führt (siehe Abbildung 2.30). In der Praxis sind minimale Erhebungswinkel zwischen 5° und 10° üblich.

Die Verfügbarkeit (englisch: availability) bezieht sich nicht nur auf einen Satelliten, sondern auf die Durchführbarkeit einer Ortung, also eher auf das gesamte System. Wie wir bereits wissen, benötigen wir vier Satelliten um eine Positionsbestimmung durchzuführen, also müssen auch mindestens vier Satelliten gleichzeitig für den GPS-Empfänger sichtbar sein. Der Begriff Verfügbarkeit umfasst allerdings noch eine andere Komponente, nämlich den DOP-Faktor.

Der Dilution of Precision[62]-Faktor (kurz: DOP-Faktor) ist ein Maß für den Ortungsfehler und eine stochastische Größe. Er hängt von der aktuellen Konstellation der Satelliten ab. Eine entscheidende Rolle spielt dabei der Satellitenabstandswinkel γ, der im direkten Zusammenhang mit dem Schnittwinkel der Kreisbögen um die Satelliten steht. Abbildung 2.31 verdeutlicht diesen Zusammenhang, wobei der Schnittwinkel β den Winkel der an die Kreisbögen angenäherten Asymptoten umfasst.

[61] verlgleiche hierzu auch Abbildung 2.17 in Kapitel 2.2.4.1
[62] wörtliche Übersetzung: Verschlechterung der Präzision

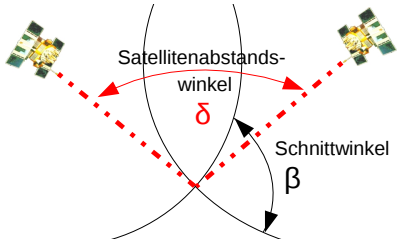


Abbildung 2.31: Satellitenabstandswinkel und Schnittwinkel der Signalkreisbögen

Kennzahl	Formel	Fehlerangabe
Position DOP	$PDOP = \frac{\sigma_p}{\sigma_r}$	für 3D-Positionsbestimmung
Horizontal DOP	$HDOP = \frac{\sigma_h}{\sigma_r}$	für 2D-Positionsbestimmung in der Horizontalebene
Vertical DOP	$VDOP = \frac{\sigma_v}{\sigma_r}$	für 2D-Positionsbestimmung in der Vertikalebene
Time DOP	$TDOP = \frac{\sigma_T}{\sigma_r}$	für Uhrenabweichung
Geometric DOP	$GDOP = \sqrt{(PDOP)^2 + (TDOP)^2}$	für das gesamte System

σ_h : Standardabweichung in der Horizontalebene
σ_p : Standardabweichung des Positionsfehlers
σ_r : Standardabweichung des Entfernungsfehlers
σ_v : Standardabweichung in der Vertikalebene
σ_T : Standardabweichung des Uhrenfehlers

Tabelle 2.5: Übersicht über die verschiedenen Fehlerkennzahlen

Stellt man sich nun mit Hilfe der Abbildung 2.18 aus Kapitel 2.2.4.1 einmal verschiedene größere und kleinere Schnittwinkel β vor, so bemerkt man, dass sich die Fläche, in der sich unsere potentiellen Standorte befinden, in ihrer Größe unterscheiden. Somit ist bei einer kleineren Fläche der Positionsfehler, der entstehen kann, auch kleiner als bei einer größeren Fläche. Die kleinste Fehlerfläche entsteht, wenn der Abstandswinkel zwischen zwei Satelliten 90° beträgt.

Des Weiteren treten neben den Abweichungen bei der Positionsbestimmung auch Differenzen bei der Entfernungsbestimmung und beim Uhrenfehler auf. Diese Abweichungen werden mit Hilfe von Standardabweichungen aus der Stochastik beschrieben. Bei ihrer Betrachtung und der dafür verantwortlichen Fehlerursachen hat es sich als sinnvoll herausgestellt, nicht nur ein Maß für alle auftretenden Fehler einzuführen, sondern mehrere spezielle Kennzahlen zu definieren. In Tabelle 2.5 sind die verschiedenen Kennzahlen, die für verschiedene Aufgaben geeignet sind, zusammengefasst.

Kennzahl	Formel	Fehlerangabe
1	Ideal	This is the highest possible confidence level to be used for applications demanding the highest possible precision at all times.
2-3	Excellent	At this confidence level, positional measurements are considered accurate enough to meet all but the most sensitive applications.
4-6	Good	Represents a level that marks the minimum appropriate for making business decisions. Positional measurements could be used to make reliable in-route navigation suggestions to the user.
7-8	Moderate	Positional measurements could be used for calculations, but the fix quality could still be improved. A more open view of the sky is recommended.
9-20	Fair	Represents a low confidence level. Positional measurements should be discarded or used only to indicate a very rough estimate of the current position.
21-50	Poor	At this level, measurements are inaccurate by as much as 300 meters with a 6-meter accurate device and should be discarded.

Tabelle 2.6: Einstufung der DOP-Faktoren in verschiedene Bereiche
Quelle: [Yue09]

Diese Kennzahlen werden permanent im GPS-Empfänger neu berechnet und eine Ortung nur dann ausgegeben, wenn die verschiedenen Kennzahlen in gewissen Toleranzbereichen liegen. Diese Bereiche können von Empfänger zu Empfänger variieren. Eine Einstufung der Werte von DOP-Faktoren ist in Tabelle 2.6 dargestellt. Sind also mindestens vier Satelliten sichtbar und liegen die einzelnen Kennzahlen in den vorgegebenen Toleranzbereichen, gilt GPS als verfügbar.

2.2.8 Bezugssystem WGS-84

Sämtliche Verfahren der Ortung und Navigation basieren auf der Kenntnis der Position von ein oder mehreren Bezugspunkten. Der oder die Bezugspunkte besitzen feste Koordinaten, die insgesamt einen Referenzrahmen oder auch ein Bezugssystem zur Ortung bilden. Ortungssysteme die ohne Satelliten arbeiten[63], besitzen im Allgemeinen nur eine beschränkte Reichweite. Sie haben somit eher einen lokalen Charakter und führen meist zu lokal angepassten Bezugssystemen.

GPS hingegen ist ein von Satelliten gestütztes Ortungssystem, welches global verfügbar ist. Als Bezugspunkte kommen nur Orte in Frage, die für alle beteiligten Komponenten stationär sind. Mit einem raumfesten Bezugssystem könnte man alle Bewegungsabläufe von natürlichen und künstlichen Himmelskörpern

[63] diese Systeme werden auch als terrestrische Systeme bezeichnet

(z.B. Satelliten, Mond, andere Planeten oder Sterne) beschreiben. Zur Ermittlung von Standpunkten auf der Erdoberfläche würde sich ein solches System allerdings nicht eignen, da sich die Koordinaten durch die Rotation der Erde und die Bewegung um die Sonne ständig ändern würden. Als Bezugssystem wird deshalb „nur“ ein sogenanntes „erdfestes Bezugssystem“ verwendet.

Ein erdfestes Bezugssystem ist international wie folgt festgelegt worden:

- Ursprung des Systems: Massenmittelpunkt der Erde
- Z-Achse: Erdrotationsachse
- Y-Achse: Senkrecht auf der Z-Achse stehend

Der Bezugspunkt, der Ursprung des Systems, ist der Massenmittelpunkt der Erde, da sich sowohl die Erde um diesen Punkt dreht, als auch das Zentrum der elliptischen Umlaufbahnen der Satelliten sich in diesem Punkt befindet. Damit ist dieser Punkt für alle Komponenten des System inertial[64]. Da sich der Massenmittelpunkt der Erde (Geozentrum) allerdings um die Sonne bewegt, was für unsere Anforderungen aber unerheblich ist, wird das erdfeste Bezugssystem als „Quasi-Inertialsystem“ bezeichnet. Das Koordinatensystem ist daher mit dem Erdkörper verbunden.

Um nun Koordinaten zu einem Punkt auf der Erdoberfläche in unserem Bezugssystem angeben zu können, benötigen wir noch Kenntnis über die Frage, wo sich die Erdoberfläche überhaupt befindet. Da die Erde keine vollkommene Kugel darstellt, sondern durch ihre Rotation an den Polen etwas abgeflacht ist, wird als Berechnungsgrundlage ein Rotationsellipsoid verwendet. Die grundlegenden Parameter für einen Rotationsellipsoiden sind die beiden Halbachsen a und b, sowie die Abplattung $f = \frac{(a-b)}{a}$ (vergleiche hierzu Abbildung 2.32).

Die Erde ist allerdings auch kein geometrisch perfekter Rotationsellipsoid. Durch Berge und Täler, sowie durch die unterschiedliche Verteilung von Landmasse und Wasser, besitzt die Erde eine abweichende Form und ist daher auch kein einfach zu berechnender mathematischer Körper. Um sich der tatsächlichen Form der Erde anzunähern werden weitere Parameter zur Veränderung des Rotationsellipsoiden einbezogen. Das bei GPS verwendete Modell ist das *World Geodetic System* aus dem Jahr 1984 (kurz: WGS-84). Es ist ein sogenanntes geodätisches Datum das aus insgesamt neun Parametern besteht, wovon die wichtigsten in der Tabelle 2.7 zu sehen sind.

[64] von lateinisch iners „untätig, träge“

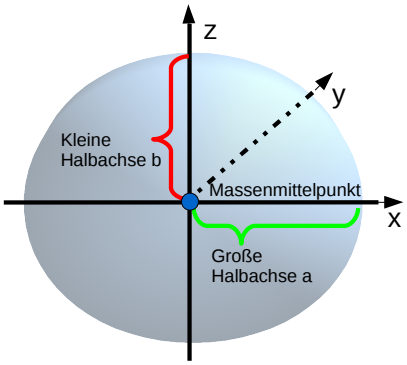


Abbildung 2.32: Rotationsellipsoid

Parameter	**Wert im WGS-84**
große Halbachse a	6 378 135 m
Abplattung f	$\frac{1}{298,26}$
Winkelgeschwindigkeit der Erdrotation ω	$7,292115147 * 10^{-5} \frac{rad}{s}$
geozentrische Gravitationskonstante GM_E	$398600,8 \frac{km^3}{s^2}$

Tabelle 2.7: Die wichtigsten Parameter des WGS-84

Zu den Parametern zählen auch physikalische Größen, die den Rotationsellipsoiden weiter in Richtung realer Form der Erde formieren. Somit ist unser Bezugskörper kein rein mathematisches Gebilde mehr und wird deshalb auch nicht mehr als Rotationsellipsoid bezeichnet, sondern als Geoid ([HZ09]). Ein dreidimensionales Modell dieses WGS-84 Geoids ist in Abbildung 2.33 zu sehen. Um die Deformation des Rotationsellipsoiden besser zu erkennen sind die Höhenunterschiede vergrößert dargestellt. Das WGS-84 ist ein Modell, das sich im Mittel der Realität der Erde schon sehr gut nähert. Basierend auf WGS-84 gibt es noch weitere Modelle, die den Geoiden lokal noch besser annähern. Diese spielen besonders bei der Bestimmung von Höhenangaben eine größere Rolle. Mittlerweile gibt es für über 200 Regionen der Erde eigene geodätische Daten[65].

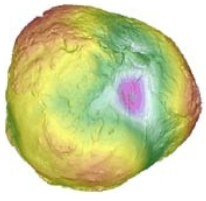


Abbildung 2.33: Ca. 15000-fach überhöhtes 3D-Modell des WGS-84 Geoids

Quelle: http://www.kowoma.de/gps/geo/geoid_kugel.jpg

[65] nachzuschlagen unter [BLR83]
oder im Internet unter http://www.colorado.edu/geography/gcraft/notes/datum/edlist.html

2.3 Alternativen und Zukunft von Satelliten-Navigationssystemen

Zum Abschluss dieses zweiten Kapitels 1 möchte ich noch einen kurzen Überblick über aktuelle und zukünftige Alternativen zu GPS geben. GPS hat in den letzten Jahren einen imensen Bedeutungszuwachs erfahren, sowohl im privaten Bereich (z.B. Autonavigation), als auch in der Wirtschaft. In den Jahren 2004 bis 2006 hat sich der Absatz von ca. 60 000 Geräten auf ca. 1,8 Millionen auf das 30-fache gesteigert. Mit ursächlich hierfür mag der heutzutage für viele Privatnutzer erschwingliche Kaufpreis von durchschnittlich 150-200€ sein. Laut Prognosen (Quelle: [CP06]) soll die Zahl der Nutzer von Satellitenortungssystemen bis zum Jahr 2020 weltweit auf ca. 3,6 Milliarden steigen. Deshalb wird an präzisionssteigernden Alternativen gearbeitet, um diesen Markt auszuweiten.

Differential GPS (kurz: DGPS) ist keine Alternative im eigentlichen Sinne, denn DGPS ist lediglich eine Erweiterung des bestehenden GPS. Die Funktionsweise zur Steigerung der Präzision von GPS ist im Prinzip sehr einfach. Um durch verschiedene Fehlerquellen (siehe Kapitel 2.2.6) entstandenen Abweichungen zu korrigieren nutzt DGPS Referenzstationen. Eine solche Referenzstation ist stationär und exakt vermessen. Sie empfängt, wie auch der mobile DGPS-Empfänger, die normalen GPS-Signale der Satelliten. Nun berechnet die Referenzstation mit Hilfe der üblichen Verfahren die jeweilige Pseudoentfernung zu den unterschiedlichen Satelliten, und vergleicht diese mit den mathematisch berechneten Entfernungen, die auf ihren fixen Koordinaten basieren. Daraus ergeben sich Korrekturwerte, die die aktuelle Beeinflussung der Fehlerquellen, wie zum Beispiel Wetter und Ephemeriedenfehler, beinhalten.

Jeder Korrekturwert wird dann von der Referenzstation über separate Übertragungssysteme als Korrektursignal ausgesendet. Das mobile DGPS-Gerät empfängt diese Informationen und verrechnet sie mit den eigenen lokalen Meßwerten (siehe Abbildung 2.34). So kann man Ungenauigkeiten bei der Positionsberechnung auf einen Meter und weniger reduzieren, wenn sich das DGPS-Gerät in einem Umkreis von 200 km um die Referenzstation befindet. DGPS verwendet somit eine relative Positionsbestimmung (durch Hilfe einer Referenzstation), im Gegensatz zum herkömmlichen GPS, welches eine absolute Positionsbestimmung verwendet. Das Prinzip des DGPS liegt auch dem in dieser Arbeit entwickelten System zugrunde (vgl. Kapitel 5).

Wie schon erwähnt, kann das Korrektursignal über verschiedene Systeme versendet werden. So können etwa auch weitere, vom eigentlichen GPS-System

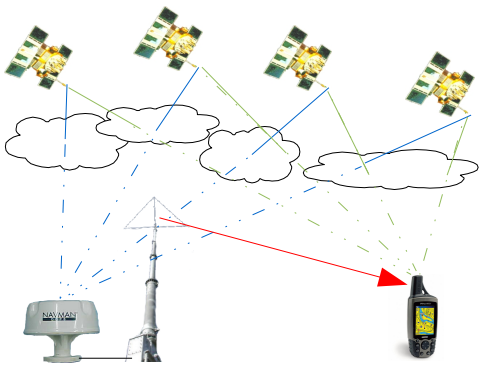


Abbildung 2.34: Prinzip und Aufbau des DGPS

unabhängige Satelliten, die ermittelten Korrekturwerte versenden. Das in den USA hauptsächlich verwendete System heißt *WAAS* (Wide Area Augmentation System, seit 2003), das europäische System EGNOS (European Geostationary Navigation Overlay Service, seit 2006) und das japanisch-asiatische System MSAS (Multi-Functional Satellite Augmentation System, seit 2005). Die hierzu verwendeten Satelliten sind geostationär. Des Weitern wird das Korrektursignal mancherorts auch mittels Langwellensendern ausgestrahlt[66] oder man kann es per NTRIP (Networked Transport of RTCM via Internet Protocol) über das Internet beziehen.

Ebenfalls ist der Ausbau des bestehenden GPS schon geplant und befindet sich teilweise in der Umsetzung. Es wird eine Verbesserung in allen Segmenten angestrebt.

Auch im Bereich der Signale sind Modifikationen geplant. Neben der Änderung der Informationsstruktur des auf der L1-Frequenz ausgesendeten zivilen Signals, sollen auf der bis jetzt nur militärisch genutzten Frequenz L2 zivil nutzbare Informationen aufmoduliert werden (L2C). Zusätzlich soll eine weitere Frequenz L5 in Betrieb genommen werden, die ebenfalls für zivile Zwecke genutzt werden soll. Mit Hilfe dieser Informationen ist dann eine schnellere und robustere Auflösung der Mehrdeutigkeit möglich(Ambiguity Resolution).

Zukünftig in Umlauf gebrachte Satelliten werden sich von den aktuellen, in Kapitel 2.2.2.1 beschrieben, unterscheiden. Da eine neue Sendefrequenz genutzt werden soll, müssen die Satelliten in der Lage sein diese auch auszusenden. Zudem werden innerhalb der Satelliten technische Neuerungen vorgenommen, die jedoch nach außen hin nicht direkt bemerkbar sind, wie z.B. der Einbau ver-

[66] wie zum Beispiel in Koblenz
mehr Informationen unter www.wsv.de/fvt/dgps/koblenz/index.html, Quelle: [Ver09]

besserter Atomuhren. Zwei Veränderungen im Raumsegment werden für den Benutzer allerdings deutlich bemerkbar sein: zum Einen soll die Anzahl der sich im Orbit befindlichen Satelliten erhöht werden, um neben einer besseren Verfügbarkeit auch eine höhere Präzision zu erhalten (siehe DOP-Faktor in Kapitel 2.2.7). Außerdem soll die Sendeleistung um einige dB[67] angehoben werden, wodurch die Störungen durch verschiedene Objekte verringert werden (z.B. im Laubwald).

Das Kontrollsegment soll ausgebaut und aufgrund der in den letzten Jahren sehr weit fortgeschrittenen Informationstechnik reorganisiert werden. Man plant zusätzlich sechs weitere Monitorstationen einzurichten sowie die alten Stationen zu modernisieren, um so für neue Kontroll- und Überwachungsaufgaben ausgerüstet zu sein. Auch das Nutzersegment wird sich technisch ändern um so die angebotenen Dienste in Anspruch nehmen zu können. Der Wandel bezieht sich ebenfalls sowohl auf die Art der Nutzung, als auch auf die Einsatzgebiete.

Ein alternatives System zu GPS stellt *GLONASS* dar (schon in Kapitel 2.2.1 erwähnt). Dieses russische[68] System ist dem amerikanischen GPS sehr ähnlich und wurde Mitte der siebziger Jahre geplant. Es wurde zwar erst mit vier Jahren Verzögerung zu GPS gestartet, hatte aber gleichfalls militärischen Nutzen als Hintergrund der Entwicklung. Der Vollausbau wurde im Jahre 1997 abgeschlossen. Seitdem konnte aus finanziellen Gründen der Zustand des Systems nicht gehalten werden. Um die Zukunft dieses Systems zu sichern wird versucht Kooperationen zu schließen, was durch die Entspannung der politischen Lage möglich wird. Zum Einen mit dem systematisch ähnlichen GPS. So könnte mit über 40 Satelliten die Präzision erheblich gesteigert werden. Zum Anderen steht das europäische System *GALILEO* zur Diskussion, welches im nächsten Abschnitt kurz umrissen wird.

Wie wir erfahren haben, besitzen sowohl GPS als auch *GLONASS* militärischen Charakter. Sie unterstehen beide der militärischen Kontrolle eines Landes und sind somit für zivile Nutzer in den Punkten Kontinuität, Integrität und Messgenauigkeit mit Risiko behaftet. In den Bereichen Verkehrswesen, Wirtschaft und Wissenschaft (z.B. Geodäsie), als auch an politischem Interesse, hat GPS allerdings in den letzten Jahren sehr an Bedeutung gewonnen. Aus diesem Grund ist in Europa der Wunsch nach einem eigenen Ortungssystem, das permanet zur zivilen Nutzung zur Verfügung steht, aufgekommen. Im Jahre 2002 wurde ein derartiges Projekt vom Europäischen Rat beschlossen und erhielt den Namen *GALILEO*.

[67] Dezibel, Hilfsmaßeinheit zur Kennzeichnung von Pegeln und Maßen
[68] ehemals sowjetische

Das von der *European Commission* (EC) und der *European Space Agency* (*ESA*) geleitete Projekt soll ein *GNSS* 2 sein, also ein Global Navigation Satellite System der zweiten Generation. Das Grundprinzip von *GALILEO* entspricht dem von GPS und *GLONASS*, soll aber durch die Verwendung neuerer Technologien aufgewertet werden und dem zivilen Nutzen uneingeschränkt zur Verfügung stehen. Ein Vergleich einiger technischer Details ist in Tabelle 2.8 gegeben.

Merkmal	**Galileo**	**GPS**
Anzahl Satelliten	mind. 27	mind. 24
Flughöhe	23 200 km	20 200 km
Satellitengewicht	ca. 700 kg	ca. 2000 kg
Erdumrundung	alle 14 Std.	alle 12 Std.
Inklination	56°	55°
Frequenzen	L1=1575,42 MHz L5=1176 MHz E6=1278,75 MHz	L1=1575,42 MHz L5=1176 MHz L2=1227,6 MHz
Betreiber	zivil	US-Militär

Tabelle 2.8: Vergleich einiger Merkmale von *GALILEO* und GPS

GALILEO wird vorraussichtlich fünf verschiedene Dienste anbieten, die für Privatanwender, Firmen und Staatsorgane gedacht sind. Der Offene Dienst (Open Service, OS) steht in direkter Konkurrenz zum amerikanischen GPS. Er soll, ebenfalls wie GPS, frei und kostenlos empfangbar sein, allerdings müssen auch hier die Hersteller entsprechender Empfänger-Lizenzgebühren entrichten. Der Offene Dienst soll im Vergleich zu GPS aber wesentlich genauer sein. Der zweite zivil nutzbare Dienst ist der Kommerzielle Dienst (Commercial Service, CS). Er ist kostenpflichtig und kann gegen eine Gebühr verschlüsselte Zusatzsignale verarbeiten, um eine noch genauere Ortung durchzuführen (teilweise bis auf weniger als 10 cm). Der Sicherheitkritische Dienst (Safety-of-Life, SoL) arbeitet mit einer Zuverlässigkeitsgarantie. Er ist z.B. für den Luft- und Schienenverkehr gedacht und wird bei drohenden Ortungsfehlern Sekunden vorher mit einer Warnung den Ausfall signalisieren. Der Regulierte Dienst oder Staatliche Dienst (Public Regulated Service, PRS) ist weitgehend gegen Störungen und Verfälschungen gesichert und bietet eine hohe Genauigkeit und Zuverlässigkeit. Das Signal ist ebenfalls verschlüsselt und steht ausschließlich hoheitlichen Diensten, wie zum Beispiel Polizei, Zoll, Militär und Geheimdiensten, zur Verfügung. Der Such- und Rettungsdienst (Search And Rescue, SAR) kann weltweite Notrufe z.B. von Flugzeugen und Schiffen orten. Erstmalig soll auch eine Rückantwort der Rettungsstelle an den Notsender möglich sein.

Das Problem der Finanzierung der Investitionskosten von rund 3,9 Milliarden Euro ließ es immer wieder zu Verschiebungen des Zeitplans kommen. War die Inbetriebnahme ursprünglich für Ende 2008 geplant, befindet sich *GALILEO* zur Zeit (Anfang 2010) mit drei Satelliten im Testbetrieb[69]. Seit Ende 2007 scheint auch die Finzierung geregelt, sodass geplant ist *GALILEO* 2013 endgültig in Betrieb zu nehmen. Neben der EU, die durch Umschichten von nicht abgerufenen Agrarfördermitteln mit 2,4 Milliarden Euro den Anteil eines abgesprungenen Industriekonsortiums übernahm, sind auch weitere nicht-EU Staaten finanziell an dem Projekt beteiligt. Deutschland ist mit 500 Millionen Euro einer der größten Geldgeber und erhielt dafür im Gegenzug den Standort einer Kontrollstation.

[69] mehr aktuelle Informationen zu GALILEO unter www.esa.int/esaNA/galileo.html

3 Verwendete Hardware

Auf Seiten der Hardware wurden drei Hauptkomponenten zur Realisierung dieses Projektes verwendet:

- der Mikrocontroller
- das Funkmodul
- das GPS-Modul

Die Mikrocontroller der AVR-Serie von *Atmel*[1] gehören zu den meist eingesetzten Mikrocontrollern überhaupt. Damit sind sie auch mit am besten dokumentiert, was mir, bei der für einen Informatiker eher ungewöhnlichen Arbeit, sehr geholfen hat. Zunächst verwendete ich einen *ATmega8*, stellte aber während der Programmentwicklung bald fest, dass der In-System Self-Programmable Flash Speicher von 8 kByte zu klein war. Daraufhin entschied ich mich für einen *ATmega168*, der "pin-kompatibel" zum *ATmega8* ist, jedoch mit 16 kByte mit doppelt soviel Flashspeicher ausgestattet ist.

Dieser Mikrocontroller benötigt eine 5V-Betriebsspannung und stellt die für das Projekt nötigen Hardware-Features zur Verfügung. Im folgenden Abschnitt gehe ich zunächst kurz auf einige relevanten Features des *ATmega168*, sowie die damit zusammenhängenden Register und Konfigurationsbits, ein.

3.1 Der Mikrocontroller *ATmega168*

Der *Atmel ATmega168* ist ein CMOS 8-Bit Mikrocontroller der AVR-Serie mit einer RISC-Architektur. Er kann mit einer Taktrate von bis zu 20 MHz bei einer Betriebsspannung von 2,7 bis 5,5 Volt und einer Stromaufnahme von bis zu 5,5 mA betrieben werden. Als Speicher stehen insgesamt 16 kByte Flash Speicher für Programme, 512 Byte EEPROM Speicher für Daten sowie 1 kByte RAM zur Verfügung. Des Weiteren verfügt der *ATmega168* über 23 I/O-Lines.

[1] Atmel Corporation, www.atmel.com

Diese können generell zur Ausgabe/Eingabe von Daten verwendet werden, aber auch zur Steuerung anderer Geräte. Einige dieser I/O-Lines besitzen noch eine weitere individuelle Funktionalität, die über verschiedene Register aktiviert, deaktiviert oder konfiguriert werden können. In Abb. 3.1 ist die Pinbelegung des *ATmega168* zu sehen:

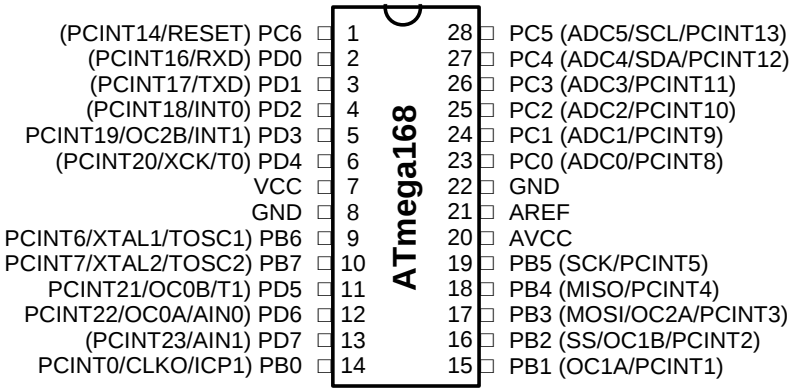


Abbildung 3.1: Pinbelegung des *ATmega168*

Der *ATmega168* verfügt über einen internen Oszillator, der mit einer Taktfrequenz von 8 MHz schwingt. Mit Hilfe des System Clock Prescaler kann man den Mikrocontroller mit unterschiedlichen Frequenzen arbeiten lassen. Der System Clock Prescaler teilt die 8 MHz durch den im CLKPR-Register eingestellten Wert (1,2,4,8,16,32,64,128,256). Da die Frequenzgenauigkeit des internen Ozillators jedoch maßgeblich von der Temperatur und der Betriebsspannung abhängig ist, wird ein externer Quarz als Taktquelle eingesetzt. Dieser befindet sich auf dem verwendeten Funkmodul *RFM12*, auf das im nächsten Abschnitt etwas genauer eingegangen wird. Um einen externen Taktgeber zu verwenden, müssen einzelne Fuses des *ATmega168*, die für die Systemuhr verantwortlich sind, konfiguriert werden. Dies sind die Fuses CKSEL0,CKSEL1,CKSEL2 und CKSEL3, die alle auf „programmed“ gesetzt werden müssen. Abbildung 3.2 zeigt die Programmierung der Fuses mit dem Programm KontrollerLab, das im Bereich Software (Kapitel 4.3) noch kurz vorgestellt wird.

Die Programmierung der Fuses kann über die ISP[2]-Schnittstelle des *ATmega168* vorgenommen werden. Über diese Schnittstelle können ebenfalls Programme in den Flash Speicher geschrieben und auch Daten in das interne EEPROM abgelegt werden.

[2]ISP:In-System Programmer, d.h. der Mikrocontroller muss zur Programmierung nicht aus seinem System/Schaltung herausgenommen werden

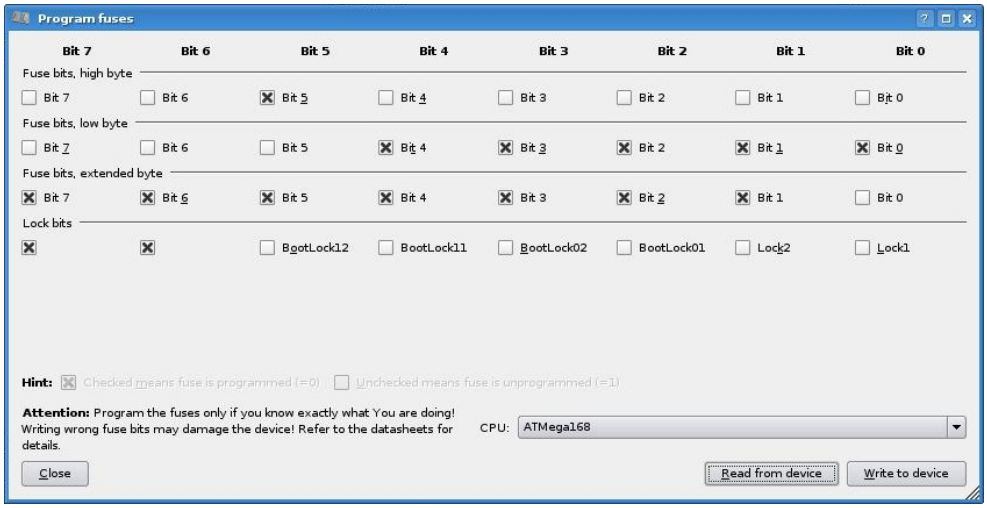


Abbildung 3.2: Screenshot: Programmierung der Fuses mittels KontrollerLab

3.1.1 Die Register

Der *ATmega168* besitzt 88 verschiedene Register mit jeweils acht Bit. Man kann in Schreib/Lese- und in so genannte Statusregister unterteilen. Die Schreib/Lese-Register können je nach Bedarf des Programms geschrieben und gelesen werden. Im Gegensatz hierzu sind die Statusregister nur zum Auslesen des Status vorgesehen, in dem sich der Prozessor momentan befindet. Die ALU[3] steht in direkter Verbindung zu den Registern und kann auf Grund dessen ohne ein Laden in den ACCU[4] auf diese zugreifen.

Im Datenblatt des *ATmega168* (auf der beiliegenden CD zu finden) findet man ab Seite 325 eine Tabelle mit allen vorhandenen Registern. In den folgenden fünf Abschnitten werde ich kurz einige Features und Register des *ATmega168* vorstellen, die bei der Realisierung dieses Projektes von Bedeutung waren.

3.1.2 Die Ports

Die Ports des Mikrocontrollers dienen als Schnittstellen zwischen dem Mikroprozessor und seiner „Umwelt“. Der Zugriff auf die drei Ports des *ATmega168* findet über zwei 8-Bit bzw. ein 7-Bit I/O-Register statt. Die Ports bestehen aus acht bzw. sieben I/O-Lines, die unabhängig voneinander als Ein- oder Ausgänge konfiguriert und genutzt werden können. Für die Konfiguration und das Schreiben/Lesen von Daten besitzt jeder Port jeweils drei zuständige Register.

[3] **A**rithmetic **L**ogical **U**nit

[4] accumulator; Akkumulator-Register, in das Daten geschrieben werden und auf das die ALU direkten Zugriff hat

Für die Konfiguration eines Ports mit seinen I/O-Lines ist das Register DDRx[5] zuständig (Hierbei steht 'x' für B,C oder D). Durch Löschen eines entsprechenden Bits wird die dazugehörige I/O-Line als Eingang konfiguriert, durch Setzen als Ausgang. Ist eine I/O-Line, oder auch Pin, als Ausgang konfiguriert worden, kann durch das Setzen bzw. Löschen des zuständigen Bits im PORTx Register, ein High- oder Low-Pegel an diesem Pin erzeugt werden. Ist ein Pin als Eingang konfiguriert worden, kann der aktuelle Pegel, welcher an diesem Pin anliegt, über das entsprechende PINx-Register ausgelesen werden. Zusätzlich kann bei der Konfiguration als Eingang über das PORTx-Register der Pull-up Widerstand des Pins aktiviert (Setzen) oder deaktiviert (Löschen) werden.

Ein Teil des Port C steuert die auf der Platine befindlichen LEDs an. Ein anderer Teil des Ports wird zur Realisierung einer Software UART[6] verwendet, da der *ATmega168* nur einen Hardware UART besitzt aber zur Kommunikation mit dem GPS-Modul sowie zur Ausgabe des GPS-NMEA Datensatzes zwei UARTs nötig sind (mehr dazu in Kapitel 5.2).

3.1.3 Die Interrupts

Die Bandbreite von Anwendungen für Mikrocontroller ist sehr groß. Geschwindigkeit, und die damit verbundene Reaktionsgeschwindigkeit auf eintretende Ereignisse, ist eine häufig gestellte Anforderung an Mikrocontroller. Wenn der Mikrocontroller durch ständiges Abfragen (pollen) von Eingängen auf potentielle Ereignisse warten würde, würde das viel unnötige Zeit in Anspruch nehmen. Dazu müsste das Programm eine Schleife durchlaufen, die permanent den Pegel bzw. den Status der einzelnen Eingänge überprüft, um dann bei Eintreten eines Ereignisses innerhalb der Schleife auf dieses reagieren zu können. Eine sehr effektive aber auch elegante Methode diese Problem zu lösen und den Vorgang zu beschleunigen, ist das Prinzip von Interrupts.

Der Mikrokontroller arbeitet unabhängig von den eventuell eintretenden Ereignissen ein Programm seriell ab. Tritt nun ein entsprechendes Ereignis (Interrupt) auf, auf das der Mikrokontroller reagieren soll, springt dieser in eine sogenannte Interruptroutine, arbeitet diese ab und kehrt danach wieder ins das eigentliche Programm an die Stelle zurück, an dem er es zuvor verlassen hat. In Abbildung 3.3 ist die schematische Abarbeitung eines Interrupts dargestellt.

[5] **D**ata **D**irection **R**egister
[6] mehr zu UART in Abschnitt 3.1.4

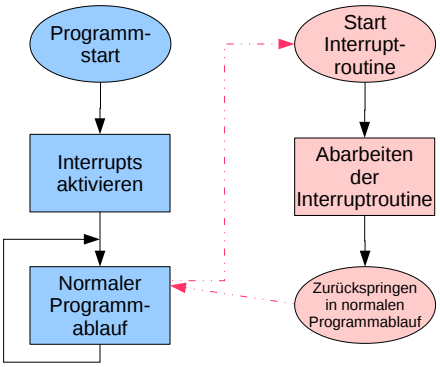


Abbildung 3.3: Programmverlauf mit Interruptroutine (ISR)

Der *ATmega168* verfügt über die folgenden fünf verschiedenen Interrupts:

1. Reset
2. Externe Interrupts
3. Timer/Counter Interrupts
4. UART Interrupts
5. Analoger Komperator

Die Reihenfolge dieser Aufzählung gibt die Priorität der verschiedenen Interrupts wieder. Dies bedeutet, dass zum Beispiel während der Abarbeitung der Interruptroutine eines externen Interrupts jederzeit das Auslösen des Reset Interrupts erfolgen kann, der den Mikrocontroller unverzüglich zurücksetzt und neu startet. Die Interrupts des analogen Komperators sind bei der Realisierung des Projektes nicht von Bedeutung. Um den Gebrauch von Interrupts überhaupt zu ermöglichen muss im Statusregister SREG des Mikrocontrollers das Global Interrupt Enable (Bit 7) gesetzt werden. Das Deaktivieren von Interrupts des Mikrocontrollers wird durch das Löschen der entsprechenden Bits bewirkt.

3.1.3.1 Externe Interrupts

Für Ereignisse, die ausserhalb des Mikrocontrollers auftreten können, sind die sogennanten externen Interrupts zuständig. Der *ATmega168* verfügt über zwei externe Interrupts, welche an den Pins 4 und 5 des Mikrocontrollers ausgelöst werden können. Die beiden externen Interrupts können über das External Interrupt Mask Register (kurz: EIMSK) aktiviert oder deaktiviert werden. Durch

ISC11	ISC10	Bedeutung
0	0	Low-Level an INT1 löst einen Interrupt aus
0	1	Pegelwechsel an INT1 löst einen Interrupt aus
1	0	eine fallende Flanke an INT1 löst einen Interrupt aus
1	1	eine steigende Flanke an INT1 löst einen Interrupt aus

Tabelle 3.1: Konfigurationsmöglichkeiten der Interruptauslösung von Interrupt INT1 mittels den Bits ISC11 und ISC10, analog dazu für Interrupt INT0 mit ISC01 und ISC00

das Setzen des jeweiligen Bits (Bit 0 für Interrupt 0 und Bit 1 für Interrupt 1) im EIMSK wird der Interrupt aktiviert. Zum Deaktivieren muss dieses Bit entsprechend gelöscht werden.

Tritt ein Ereignis an einem Interrupt auf und ist dieser aktiviert, so wird ein Flag-Bit im External Interrupt Flag Register (kurz: EIFR) gesetzt. Springt nun der Mikrocontroller in die entsprechende Interruptroutine, wird dieses Bit wieder gelöscht, um so den Interrupt für ein erneutes Auftreten eines Ereignisses zu reaktivieren.

Über das External Interrupt Control Register A (kurz: EICRA) kann konfiguriert werden, auf welches Ereignis die beiden Interrupts INT0 und INT1 reagieren sollen. In Tabelle 3.1 sind die verschiedenen Einstellungsmöglichkeiten der Interrupts aufgelistet, die durch die entsprechenden Interrupt Sense Control-Bits (kurz: ISC) eingestellt werden können.

3.1.3.2 Die Timer/Counter

Im letzten Abschnitt wurden externe Interrupts behandelt, die durch ein Ereignis ausserhalb des Mikrocontrollers ausgelöst werden können. Im Gegensatz hierzu gibt es interne Interrupts, welche innerhalb des Mikrocontrollers auftreten können. Dies sind zum Beispiel die Timer/Counter-Funktionen des *ATmega168*, die das Springen in eine entsprechende Interruptroutine auslösen können.

Der *ATmega168* verfügt über zwei 8-Bit und einen 16-Bit Timer/Counter die grundsätzlich die gleiche Funktion haben. Der Wertebereich der 8-Bit Timer/Counter erstreckt sich von 0 bis 255 und der des 16-Bit Timer/Counter 0 bis 65535. Zusätzlich sind die Timer/Counter in der Lage eine Pulsweitenmodulation (PWM) durchzuführen. Diese sind jedoch für mein Projekt nicht relevant, weshalb auf diesen Aspekt nicht weiter eingegangen wird.

Da die drei Timer/Counter recht ähnlich aufgebaut sind, werde ich im Folgenden nur auf den Timer/Counter1 eingehen. Denn nur dieser fand in meinem Programm Verwendung, und das auch lediglich als Zähler. Der Mikrocontroller inkrementiert mit einer bestimmten Geschwindigkeit den Wert des Registers TCNT[7]1 jeweils um Eins bis zum oben erwähntem Höchstwert und fängt danach wieder bei Null an. Beim 16-Bit Timer/Counter besteht dieses TCNT1-Register aus den beiden 8-Bit Registern TCNT1L (Lower Bytes) und TCNT1H (Higher Bytes). Die Geschwindigkeit, mit der der Mikrocontroller die Inkrementierung durchführt, wird über das Register TCCR1B eingestellt.

Über die Bits CS12, CS11, CS10 (Bits 0-2) wird ein sogenannter Prescaler konfiguriert. Dieser verringert die Zählgeschwindigkeit auf einen bestimmten Bruchteil des Systemtakts. In der Tabelle 3.2 sind die unterschiedlichen Konfigurationsmöglichkeiten des Prescalers aufgelistet.

CS12	CS11	CS10	Einstellung
0	0	0	der Timer/Counter wird gestoppt
0	0	1	der CPU-Takt
0	1	0	CPU-Takt / 8
0	1	1	CPU-Takt / 64
1	0	0	CPU-Takt / 256
1	0	1	CPU-Takt / 1024
1	1	0	Takt durch fallende Flanke an T1
1	1	1	Takt durch steigende Flanke an T1

Tabelle 3.2: Konfigurationsmöglichkeiten des Timer/Counter TCNT1

Weitere wichtige Register sind die TIMSKx[8]-Register. Durch Setzen des TOIEx[9]1 Bits wird die Interruptfunktion des jeweiligen Counters aktiviert. Dies bedeutet, dass beim Übergang von 65535 (bzw. 255 bei einem 8-Bit Timer/Counter) nach 0 ein interner Interrupt ausgelöst wird. Mit Hilfe dieser Methode können zeitabhängige Aufgaben einfach realisiert werden.

Mit Hilfe des OCRx[10] Registers, bestehend aus OCRxA und OCRxB-Register, kann man einen Wert angeben, der permanent mit dem aktuellen Stand des TCNTx-Registers verglichen wird. Stimmen der Wert im OCRx-Register mit dem im TCNTx-Register überein, wird beim gesetzten OCIExA-Bit im entsprechenden TIMSKx-Register ein Interrupt ausgelöst.

[7] Timer/Counter Data Register Timer
[8] Timer/Counter Interrupt Mask
[9] Timer/Counter Overflow Interrupt Enable
[10] Output Compare Register

Der Vollständigkeit halber soll noch das Register TCCR[11]xA erwähnt werden, in dem Einstellungen zur Pulsweitenmodulierung möglich sind.

3.1.4 Die UART

Die UART[12] ist eine serielle Schnittstelle, über die der Mikrocontroller mit anderen Geräten kommunizieren kann. Die Datenübertragung läuft über zwei Pins. Über einen werden Daten gesendet (TxD[13]) und über den anderen werden Daten empfangen (RxD[14]). Übertragen werden die Nutzdaten in einem sogenannten Frame, dessen Format teilweise konfiguriert werden kann. Das am häufigsten verwendete Format wird als '8N1'[15] bezeichnet und findet auch in dieser Arbeit Anwendung. Neben der Konfiguration des zu übertragenen Frames muss noch eine Übertragungsgeschwindigkeit, die Baud-Rate, festgelegt werden. Der *ATmega168* besitzt so eine fertige UART, welche über die Register UCSR0A, UCSR0B, UCSR0C, UBRR0H und UBRR0L konfiguriert werden kann.

Die UART kann beim Empfang von Daten in zwei verschiedenen Modi betrieben werden. Im Polling-Modus wird in einer Schleife im Programm permanent der Status des RXC-Bit im Register UCSR0A abgefragt, welches den Empfang eines kompletten Frames anzeigt. Ist das Bit gesetzt, können die Nutzdaten aus dem Register UDR0[16] gelesen werden. Das RXC-Bit wird dann wieder automatisch gelöscht und der Mikrocontroller ist wieder bereit für den Empfang eines neuen Frames.

Im Interrup-Modus wird nicht in einer Schleife auf das Eintreffen eines Frames gewartet. Hat der Mikrocontroller einen Frame empfangen, setzt er das RXC-Bit und löst, wenn das RXCIE0-Bit im Register RXCIE0 gesetzt ist, einen Interrupt aus. Auf diesen kann dann die entsprechende Interrupt-Routine reagieren und einen eigenen Programmteil abarbeiten.

3.1.5 SPI

Die SPI[17]-Schnittstelle ist ein synchroner, serieller Bus mit einer Master-Slave-Struktur. Neben den beiden Datenleitungen gibt es noch eine Slave-Select oder

[11] Timer Counter Control Register
[12] Universal Asynchronous Receiver and Transmitter
[13] Transmit Data
[14] Receive Data
[15] 1 Startbit, 8 Bit Nutzdaten, kein Parity-Bit, 1 Stop-Bit
[16] UART I/O Data Register
[17] Serial Peripheral Interface

auch Chip-Select (kurz: SS oder auch CS) Leitung, wovon eine bei einem Aufbau mit mehreren Slaves für die Kommunikation ausgewählt werden kann. Die Übertragungsrate ist abhängig vom Taktsignal, das auf die SCK[18]-Leitung gelegt wird und bis in den MHz-Bereich gehen kann. So können durch die Möglichkeit des gleichzeitigen Empfangens und Sendens von Daten und der schnellen Übertragungsgeschwindigkeit hohe Datenübertragungraten erzielt werden.

Der Mikrocontroller kann sich bei der Nutzung von SPI in zwei Modi befinden. Entweder gibt er als Master im System den Takt auf die SCK-Leitung und wählt den Slave aus mit dem er kommunizieren möchte, oder er ist einer von vielleicht mehreren Slaves, der erst vom Master des Systems zur Datenübertragung selektiert wird, und über die SCK-Leitung den Übertragungstakt des Masters empfängt. Diese grundlegende Konfiguration der SPI, und einige weitere Optionen, werden über das Register SPCR[19] des *ATmega168* vorgenommen.

Unabhängig vom Master- oder Slave-Modus wird die Kommunikation des *ATmega168* mit anderen Peripherie-Geräten über die beiden Register SPSR[20] und SPDR[21] vorgenommen. Ist die Übertragung eines Bytes komplett, wird im SPSR das Bit SPIF[22]-Bit gesetzt und ein Interrupt ausgelöst, der in der entsprechenden Interrupt-Routine behandelt werden kann.

3.2 Das Funkmodul *RFM12*

Um die Korrekturdaten kabellos zur mobilen Einheit übertragen zu können ist eine Funkverbindung nötig. Die Funkmodule der RFM-Serie der Firma *HopeRF*[23] sind kostengünstige ISM[24]-Band Funkmodule. Aufgrund der kompakten Bauform und der mit ca. 0,3μA geringen Stromaufnahme im Standby-Mode, eignen sich die Module sehr gut für den mobilen Einsatz und somit auch für dieses Projekt. Die Reichweite der Funkmodule beträgt mit einer einfachen kleinen Drahtantenne etwa 100 Meter im Freien.

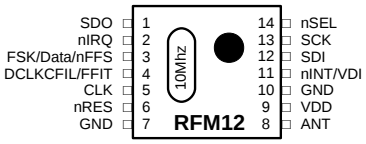
Die Funkmodule übertragen ihre Daten im 433MHz-Frequenzbereich und arbeiten somit in einem der ISM-Bänder, welche kosten- und anmeldefrei nutzbar sind. Der 433MHz-Frequenzbereich liegt zwischen 433,05 MHz und 434,79 MHz.

[18] System Clock
[19] SPI Control Register
[20] SPI Status Register
[21] SPI Data Register
[22] SPI Interrupt Flag
[23] www.hoperf.com
[24] Industrial, Scientific and Medical

Die Daten werden mit Hilfe der FSK[25]-Modulierung auf die Trägerfrequenz mit einer Datenrate von bis zu 115,2 kbps aufmoduliert. Angesteuert und programmiert werden können die Module über die SPI Schnittstelle, welche ich schon kurz im vorangegangenen Abschnitt 3.1.5 vorgestellt habe.

Bei der Realisierung der Funkverbindung werden in diesem Projekt zwei *RFM12*-Funkmodule (Abb. 3.4(a)) verwendet. Diese zeichnen sich im Gegensatz zu anderen Modulen der RFM-Serie dadurch aus, dass sie sowohl Daten empfangen als auch Daten senden können. In Abbildung 3.4(b) ist die Anschlussbelegung der *RFM12*-Module und in Tabelle 3.3 die entsprechende Legende zu sehen.

(a) Das Modul



(b) Pinbelegung

Abbildung 3.4: Das Funkmodul *RFM12*

SDO	Ausgang	Serial data output with bus hold
nIRQ	Ausgang	Interrupts request output (active low)
FSK/DATA/ nFFS	Eingang/Ausgang/ Eingang	Transmit FSK data input/Received data output(FIFO not used)/ FIFO select
DCLK/CFIL/ FFIT	Ausgang/analoger Ausgang und Eingang/Ausgang	Clock output (no FIFIO)/external filter capacitor (analog mode)/FIFO interupts (active high) when FIFO level set to 1, FIFO empty interruption can be achieved
CLK	Ausgang	Clock output for external microcontroller
nRES	Ausgang	Reset output (active low)
GND	Spannungsversorgung	Power ground
nSEL	Eingang	Chip select (active low)
SCK	Eingang	SPI clock input
SDI	Eingang	SPI data input
nINT/VDI	Eingang/Ausgang	Interrupt input (active low)/Valid data indicator
VDD	Spannungsversorgung	Positive power supply
ANT	Antenne	antenna

Tabelle 3.3: Legende der Pinbelegung aus Abb. 3.4(b)

[25]Frequency Shift Keying

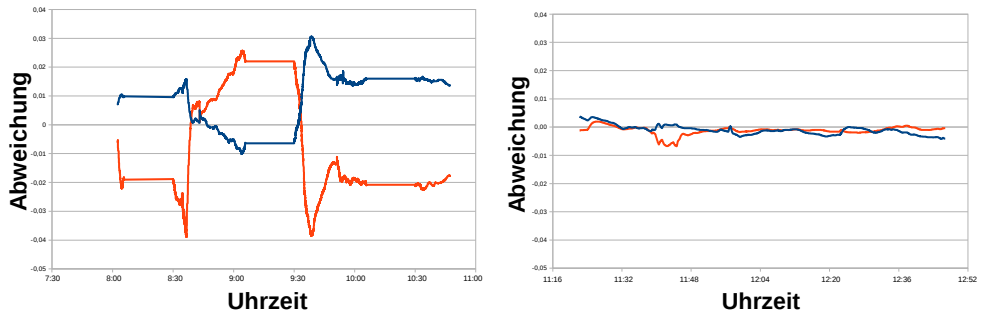
3.3 Das GPS-Modul HI-204III

Die Wahl der GPS-Module, die bei der Umsetzung des Projektes verwendet werden sollten, war wesentlich schwieriger als erwartet. Da die Hardware für dieses Projekt möglichst preiswert sein sollte, erwarb ich zunächst ein älteres GPS-Modul für ca. 10 Euro. Dieses Modul war noch mit der Technologie der ersten kommerziellen GPS-Geräte ausgestattet. Aus den ersten Messreihen war aber schon recht deutlich zu erkennen, dass dieses Modul für die geforderte Genauigkeit nicht annähernd geeignet war. Die ausgegebene Position wich teilweise bis zu 100 Meter von der eigentlichen Position ab. Auch die Konstanz der Messwerte war nicht zufriedenstellend, da die Positionswerte innerhalb einer Minute mehrere Sprünge um bis zu 10 Metern aufwiesen.

Um nun ein geeignetes GPS-Modul zu finden, bestellte ich nach und nach noch drei weitere Module. Unter anderem war das Modul *GPS-MS1E* der Firma *µ-blox*[26], schon für 20 Euro erhältlich. Exemplarisch soll hier eine Messreihe mit diesem Modul vorgestellt werden. In Abbildung 3.5(a) sind die Abweichungen der Längen- und Breitenkoordinate einer Messung dargestellt. Auffällig hierbei ist der gegenläufige Verlauf der Graphen. Dies deutet auf einen systematischen Fehler hin, welcher eventuell durch Berechnungen eleminiert werden kann. Allerdings trat dieses Phänomen nur bei wenigen Messreihen auf, was wiederum gegen eine Systematik spricht. Ebenfalls bemerkenswert sind die extremen Schwankungen der Graphen. Diese betragen z.B. im Fall der Längenkoordinate bis zu 0,04' (Minuten). Dies entspricht einer Abweichung von ungefähr 74 m, und ist somit indiskutabel.

Das GPS-Modul *HI-204III* von *Haicom*[27] lieferte bereits in den ersten Messreihen vielversprechende Ergebnisse. Es ist mit 50 Euro schon eines der teureren Geräte in diesem Segment, ermittelte allerdings die Position mit einer Genauigkeit von unter zehn Metern. Auch ein sprunghafter Verlauf der Positionsangaben über einen mittelfristigen Zeitraum blieb aus. Abbildung 3.5(b) zeigt die Längen- und Breitenkoordinaten der ersten Messung. Die beiden Graphen haben einen glatteren und teilweise sogar parallelen Verlauf. Außerdem weisen sie nicht so extremen Schwankungen auf. Dass der größte Ausschlag bei beiden Graphen zur gleichen Zeit auftritt, lässt auf eine lokal gegebene Störung der Messung schließen. Das Ziel dieser Arbeit ist, genau solche Schwankungen durch Referenzmessung und Korrekturdaten zu eleminieren.

[26] www.mblox.com
[27] www.haicom.com.tw oder www.haicomholland.nl



(a) *µ-blox* `GPS-MS1E` (b) *Haicom HI-204III*

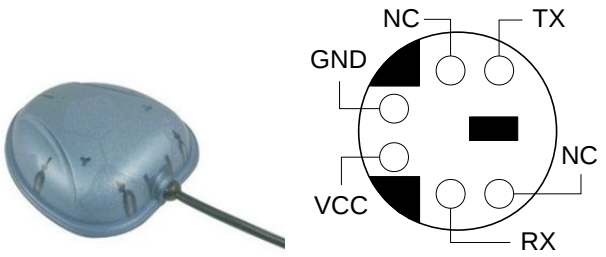
Abbildung 3.5: Vergleich der beiden Messreihenergebnisse der GPS-Module *µ-blox GPS-MS1E* und *Haicom HI-204III*

Das *HI-204III* basiert auf dem *SiRF Star III* Chip der Firma *SiRF*[28]. Dieser Chip zeichnet sich durch eine sehr schnelle 'Time to first fix' (TTFF) aus. Er besitzt 20 Kanäle und kann somit alle in Sichtweite befindlichen Satelliten gleichzeitig zur Auswertung benutzen. Des Weiteren ist der Chip in der Lage die Signale von *WAAS* und *EGNOS* (vergleiche hierzu Kapitel 2.3) auszuwerten und somit die Positionsgenauigkeit weiter zu verbessern. Es hat eine hohe Eingangsempfindlichkeit von -159dBm. Zur Kommunikation mit anderen Geräten dient eine UART-Schnittstelle.

Das Hardware-Interface ist ein 6-poliger PS/II mini-DIN Stecker (Abb. 3.6(b)), von dem allerdings nur vier Pole genutzt werden. Zwei Pins dienen der Spannungsversorgung, die 3,3V oder von 3.8V bis zu 12V betragen kann (Pin 1 $\hat{=}$ Masse, Pin 2 $\hat{=}$ Vcc). Die beiden anderen Leitungen dienen bei der seriellen Kommunikation als Dateneingang (Pin 4, Rx) bzw. -ausgang (Pin 5, Tx). Ihr Logikpegel hängt von der an das Modul angelegten Spannung ab. Bei 3.3V findet der Datenaustausch mit LVTTL[29] statt und bei einer Spanung von 3.8V bis 12V mit dem Logikpegel der RS232.

Über dieses Hardware-Interface können einzelne Parameter des GPS-Moduls gesetzt, zwischen verschiedenen Ausgabeformaten gewählt und Statusinformationen überwacht werden. Als Standart wird zur Kommunikation das 8N1-Protokoll(vgl. 3.1.4) und eine Baud-Rate von 4800bps verwendet. Als Ausgabeformat für die Positionsinformationen unterstützt das Modul die sogenannte NMEA-0183-Spezifikation der *National Marine Electronics Association*[30]. Wei-

[28] www.sirf.com
[29] Low Voltage Transistor-Transistor-Logik
[30] www.nmea.org



(a) Das Modul (b) Pinbelegung

Abbildung 3.6: Das GPS-Modul *Haicom HI-204III*

tere Informationen zum Format, sowie die verschiedenen Typen der Mitteilungen, sind im Anhang zu finden (siehe Anhang A.4.1).

4 Verwendete Software

4.1 AVRTerm

AvrTerm ist ein Terminalprogramm von Roland Walter[1]. Dieses Programm dient zur Kommunikation eines PCs mit einem an der seriellen Schnittstelle angeschlossenen Gerät. Das Programm verfügt über verschiedene Darstellungsformen von Daten, die von der seriellen Schnittstelle eintreffen, und kann auch Daten in unterschiedlicher Kodierung über die serielle Schnittstelle versenden. Ich habe *AVRTerm* zum Testen und Debuggen von Programmen auf dem Mikrocontroller, sowie zur Überwachung der NMEA-Mitteilungen der GPS-Module genutzt.

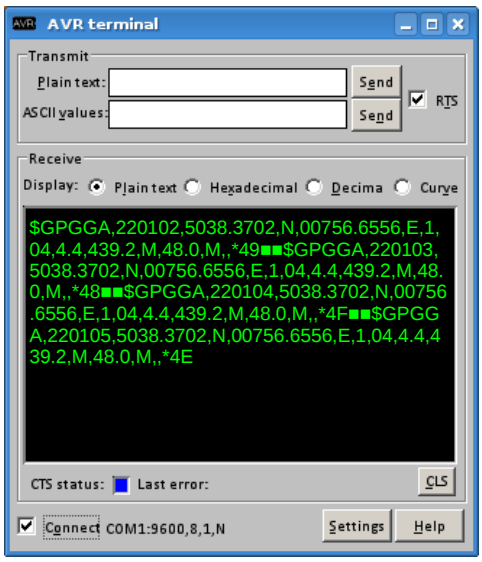


Abbildung 4.1: Das Programm *AVRTerm*

[1] www.rowalt.de/mc/avr/toolsd.htm

4.2 WinPlotGPS

WinPlotGPS ist ein Programm, mit dem Daten von bis zu zwei GPS-Receivern dargestellt werden können. Die folgenden Typen von NMEA-Mitteilungen können auf verschiedene Arten visualisiert und analysiert werden: GPRMC, GPGGA, GPGSA and GPGSV (mehr zu diesen NMEA-Daten im Anhang A.4.1). Des Weiteren können alle empfangenen NMEA-Mitteilungen in einer Textdatei gespeichert und somit später mit anderen Programmen weiter analysiert und bearbeitet werden. Alle für diese Arbeit durchgeführten Messreihen sind mit Hilfe dieses Programmes aufgezeichnet worden.

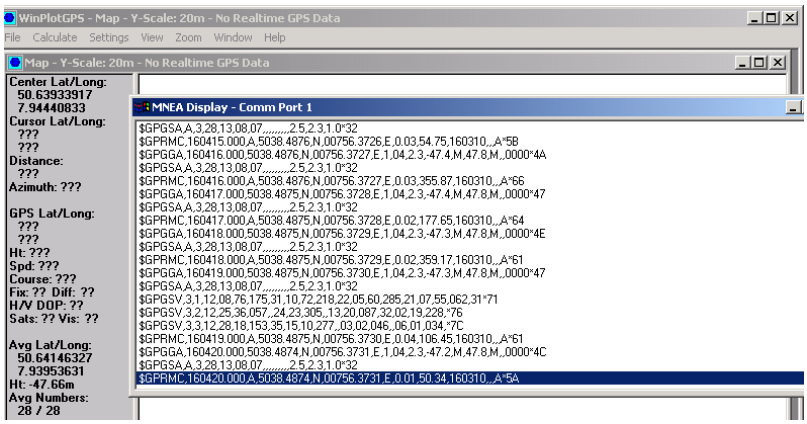


Abbildung 4.2: Screenshot des Programms *WinPlotGPS*

4.3 KontrollerLab

KontrollerLab ist eine Mikrocontroller-Entwicklungsumgebung für KDE unter Linux. Das Programm ist Freeware und kann von der Homepage der Entwickler *cadmaniac* heruntergeladen werden [2]. Mit Hilfe der Software kann man bequem Projekte zum Programmieren von Mikrocontrollern anlegen, konfigurieren und bearbeiten. *KontrollerLab* unterstützt eine Vielzahl von AVR-Mikrocontrollern, die auch über verschiedene Programmer direkt programmiert, gelöscht oder zurückgesetzt werden können. Den in C geschriebenen Quellcode kann man aus dem Programm heraus compilieren und auch direkt auf den angeschlossenen Mikrocontroller schreiben. Auch das Auslesen und Schreiben der sogenannten Fuses eines Mikrocontrollers kann mit *KontrollerLab* vorgenommen werden. *KontrollerLab* verfügt auch über einen Debugging-Modus, den ich aber nicht getestet oder verwendet habe.

[2] www.cadmaniac.org/projectMain.php?projectName=kontrollerlab

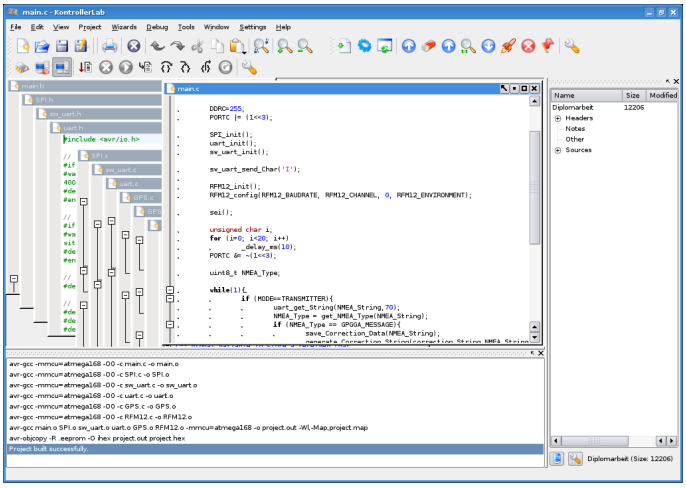


Abbildung 4.3: Screenshot *KontrollerLab*

4.4 Eagle

Die Schaltpläne (Anhang A.7) und die daraus resultierenden Platinen (Anhang A.8) sind mit dem Platinen-Layout-Editor *Eagle* erstellt worden. Dieses Programm ist zwar keine Freeware, jedoch können kleinere Platinen in der Testversion kostenlos erstellt werden. Das Programm verfügt über eine umfassende Bibliothek mit den verschiedensten Bauteilen. Nach dem Entwurf des Schaltplans kann der integrierte Autorouter ein automatisches Platinenlayout erzeugen, welches dann noch manuell nachbearbeitet werden kann.

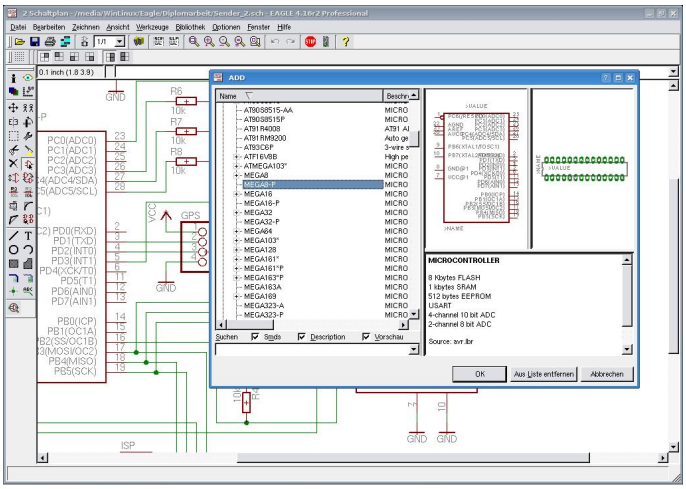


Abbildung 4.4: Screenshot *Eagle*

5 Die Projektkomponenten Hardware und Software

Ziel dieser Arbeit ist es ein System auf Basis von GPS zu entwickeln, welches eine höhere Genauigkeit bietet. Das Prinzip, das diesem Projekt zu Grunde liegt, ist das des Differential GPS[1]. Eine eigene Referenzstation besitzt ein GPS-Modul, ermittelt lokale Korrekturwerte und versendet diese über ein Funksignal, das eine Reichweite von ca. 100m besitzt. Ein autonomes System, wie zum Beispiel ein Roboter, empfängt diese Korrekturwerte und verrechnet sie mit den Daten des eigenen GPS-Moduls. Die beiden verwendeten GPS-Module nutzen ihrerseits schon DGPS um eine bessere Positionsbestimmung zu ermöglichen; durch die in dieser Arbeit erstellten Komponenten sollen jedoch noch lokalere Korrekturwerte ermittelt werden um die Positionsbestimmung noch weiter zu verbessern.

Das Projekt kann man in zwei Hauptbereiche aufteilen. Auf der einen Seite steht die Hardware. Sie umfasst den Entwurf eines Schaltplans und dessen Realisierung in Form der Erstellung einer Platine und deren Bestückung. Die andere Seite ist die Software mit der der verwendete Mikrocontroller programmiert werden soll. Hier muss die Steuerung und Nutzung des Funkmoduls *RFM12*, der Empfang von NMEA-Datensätzen über die UART-Schnittstelle, die Verarbeitung der Daten und die Ausgabe eines NMEA-Datensatzes über eine Software-UART umgesetzt werden.

[1] vgl. Kapitel 2.3

5.1 Die Hardware

Der zweiteilige Versuchsaufbau besteht aus dem Sender und dem Empfänger. Auf Seiten der Hardware, also der Platine und deren Bestückung, sind Sender und Empfänger baugleich. Sie bestehen aus folgenden vier Hauptbereichen:

- Stromversorgung
- Anschluss des GPS-Moduls
- Anschluss des Funk-Moduls
- Schnittstelle zur RS-232

In den folgenden fünf kurzen Abschnitten wird der Aufbau der einzelnen Bereiche beschrieben.

5.1.1 Die Stromversorgung

Zur Stromversorgung wird bei diesem Projekt ein externens Netzteil mit 12V Gleichspannung eingesetzt. Mit diesen 12V wird das GPS-Modul direkt versorgt. Im Datenblatt des GPS-Moduls *HI-204III* ist ein Versorgungsspannungsbereich zwischen 3.8V und 12V als Spannung angegeben. In der Praxis stellte sich jedoch heraus, dass das Modul mit einer Spannung weniger als 10V kein auswertbares RS-232 Signal mehr ausgibt. Auch der alternative Betrieb mit einer Spannung von 3.3V führte zu Problemen bei der Auswertung des Signals.

Der Rest der Schaltung benötigt eine Betriebspannung von 5V. Um diese bereit zu stellen, wird ein Festspannungsregler vom Typ 7805 eingesetzt. Um sowohl vor, als auch hinter dem Spannungsregler die Spannung zu glätten bzw. zu stabilisieren, werden jeweils ein größerer Elektrolytkondensator mit 4,7μF (C1 und C2) und ein kleinerer Keramik-Kondensator mit 100nF (C3 und C4) zur Filterung von Spannungspeaks verwendet (vergleiche Abb. 5.1).

5.1.2 Anschluss des GPS-Moduls

Das GPS-Modul wird an die UART-Schnittstelle des Mikrocontrollers *ATmega168* über Pin 2 (RxD) und Pin 3 (TxD) angeschlossen. Diese beiden Pins, sowie Masse und 5V, sind auf der Platine über eine Stiftleiste nach „aussen“ geführt, um ein GPS-Modul einfach und schnell anschließen zu können.

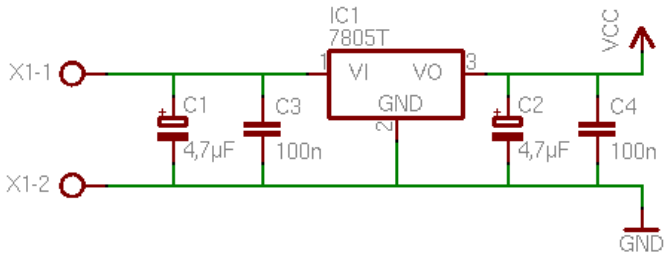


Abbildung 5.1: Schaltplan der Stromversorgung

Dies war auf Grund der verschiedenen GPS-Module mit diversen zum Einsatz gekommenen Schnittstellen nötig. Somit musste bei Verwendung eines neuen GPS-Modul nur ein neuer Adapter angefertigt und über eine Buchsenleiste an die Hauptplatine angeschlossen werden.

Das letztendlich eingesetzte GPS-Modul wird aus oben genannten Gründen (vgl. vorherigen Abschnitt 5.1.1) direkt an die 12V des externen Netzteils angeschlossen. Da der Mikrocontroller mit einem TTL[2]-Signal arbeitet und das GPS-Modul somit mit einem RS-232 Signal, ist eine direkte Verbindung der beiden Komponenten nicht möglich. Es muss ein sogenannter Pegelwandler (oder auch Pegelumsetzer) dazwischen geschaltet werden. Solch ein Pegelwandler ist der MAX232[3], der mit Hilfe einer Ladungspumpe, bestehend aus den Kondensatoren C1 bis C4 (Abb. 5.2), die Pegelumsetzung leistet. So wird der Dateneingang (Pin 2, RxD) des Mikrocontrollers über den Pegelwandler mit dem Datenausgang des GPS-Moduls (Pin 5, Tx) verbunden und umgekehrt (Pin 3 des Mikrocontrollers (TxD) mit Pin 4 des GPS-Moduls (Rx)).

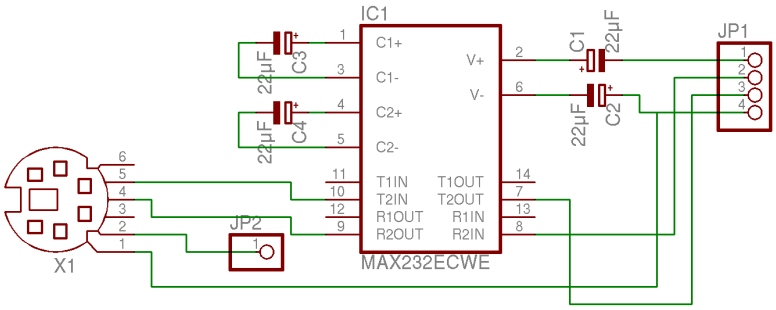


Abbildung 5.2: Verwendeter Adapter zum Anschluss des GPS-Moduls

[2]Transistor-Transistor-Logic
[3]Datenblatt befindet sich auf der beiliegenden CD

5.1.3 Anschluss des Funk-Moduls

Um das Funkmodul möglichst einfach ins System zu integrieren, habe ich zwei Stiftleisten mit einem Rastermaß von 1,27 mm (RM 1,27) verwendet, an denen das Modul einfach festgelötet werden konnte. Die Pinbelegung des Funkmoduls *RFM12* ist in Abbildung 3.4(b) aus Kapitel 3.2 zu sehen. Das Modul benötigt eine Betriebsspannung von 5V, welche aus der stabilisierten Spannung des Festspannungsreglers 7805 (vgl. Abschnitt Stromversorgung) bezogen werden kann. Hierzu werden die Pins 7 und 10 mit Masse verbunden, und Pin 9 mit Vcc 5V.

Da der größte Teil der Kommunikation zwischen Mikrocontroller und Funkmodul über die SPI-Schnittstelle der beiden Komponenten läuft, müssen die entsprechenden Pins der Schnittstelle miteinander verbunden werden. Hierzu wird der Datenausgang des Mikrocontrollers MOSI (Pin 17) mit dem Datenausgang SDI des Funkmoduls (Pin 12), und der Dateneingang des Mikrocontrollers MISO (Pin 18) mit dem Datenausgang SDO des Funkmoduls (Pin 1) verbunden. Des Weiteren wird der Takt der Übertragung, den der Mikrocontroller über SCK (Pin 19) ausgibt, an Pin 13 (SCK) des Funkmoduls angeschlossen. Um die SPI-Schnittstelle zu komplettieren wird noch die CS[4]-Leitung benötigt. Da das Funkmodul *RFM12* einen negierten CS-Eingang besitzt und als einziger Slave des Mikrocontroller dauerhaft selektiert werden kann, wird der nSEL des Funkmoduls (Pin 14) über einen Pullup-Widerstand (R4) mit CS des Mikrocontrollers (Pin 16) verbunden.

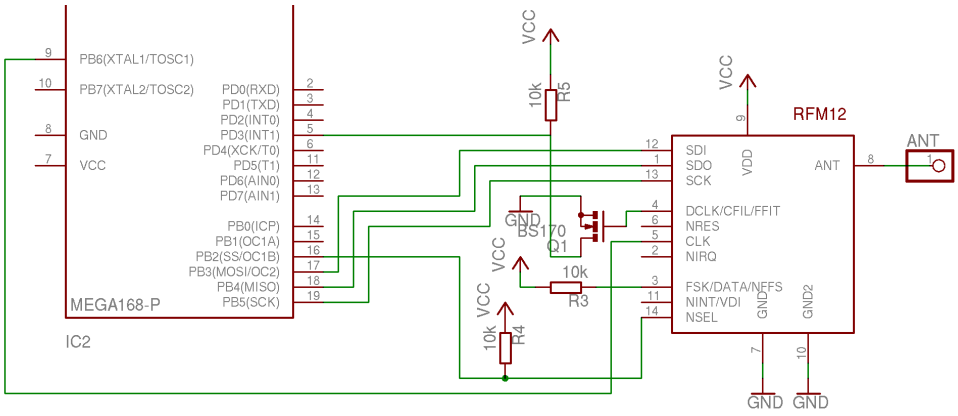


Abbildung 5.3: Schaltplan Anschluss Funkmodul *RFM12* an *ATmega168*

Weil das Funkmodul im FIFO-Mode betrieben werden soll (vgl. hierzu Kapitel 5.2.8.7), müssen noch Pin 3 (FSK/DATA/nFFS) und Pin 4 (DCLK/CFIL/FFIT) angeschlossen werden. In diesem Mode ist Pin 3 (nFFS)

[4]Chip Select, oder auch Slave Select (SS)

ein Eingang und muss auf high-Pegel gezogen werden, was über den Pullup-Widerstand R3 geschieht. Pin 4 (FFIT) ist ein Interrupt-Ausgang und wird mit dem Interrupt-Eingang (INT1, Pin 5) des Mikrocontrollers über einen CMOS-Transistor invertiert angeschlossen. Da der *ATmega168* einen flexibel konfigurierbaren Interrupt besitzt, wäre der Inverter zwar nicht notwendig, macht aber die Schaltung für andere Controller möglichst einfach nutzbar.

Auf Grund der Tatsache, dass das Funkmodul einen Quarz-Oszillator besitzt, welcher genauer schwingt als der interne Oszillator des Mikrocontrollers, kann dieser auch als Taktquelle für den Mikrocontroller verwendet werden. Hierzu wird Pin 5 (CLK) des *RFM12* auf Pin 9 (XTAL1/TOSC1) des Controllers gelegt. Des Weitern verfügt das Funkmodul noch über die Möglichkeit eine Antenne über Pin 8 anzuschließen. Zur besseren Befestigung wurde hierzu ein enstsprechender Anschluss auf die Hauptplatine gelegt. Die restlichen Pins des *RFM12* bleiben ungenutzt und können offen bleiben (NC[5]).

5.1.4 Schnittstelle zur RS-232

Zur Nutzung der durch den Mikrocontroller bearbeiteten Daten muss in der Schaltung noch ein passender Ausgang zur Verfügung stehen. Da sehr viele Geräte eine RS-232 Schnittstelle besitzen, so auch ein PC mit dessen Hilfe GPS-Daten sehr komfortabel protokolliert und analysiert werden können, habe ich mich für diesen Schnittstellentyp entschieden. Realisiert wurde das Ganze auf Seiten der Hardware, ähnlich wie in Abschnitt 5.1.3, mit Hilfe des MAX232. Als Datenausgang wird jedoch Pin 23 (PC0) über eine Software-UART (vgl. Kapitel 5.2.10) benutzt. Da die Schaltung lediglich Daten senden und keine Daten wie z.B. Steuerkommandos empfangen soll, wird auf eine Dateneingangsleitung verzichtet.

5.1.5 Sonstiges

Die Schaltung besitzt eine Poweranzeige-LED mit einem Vorwiderstand von 1kΩ, die eine korrekte Spannungsversorgung anzeigt. Informationen über die Funkaktivität der Schaltung werden mit drei Status-LEDs signalisiert. Sie werden über die Pins 26, 27, 28 (PC3, PC4, PC5) des Mikrocontrollers angesteuert und sind ebenfalls über einen Vorwiderstand von 1kΩ mit Masse verbunden.

[5] not connected

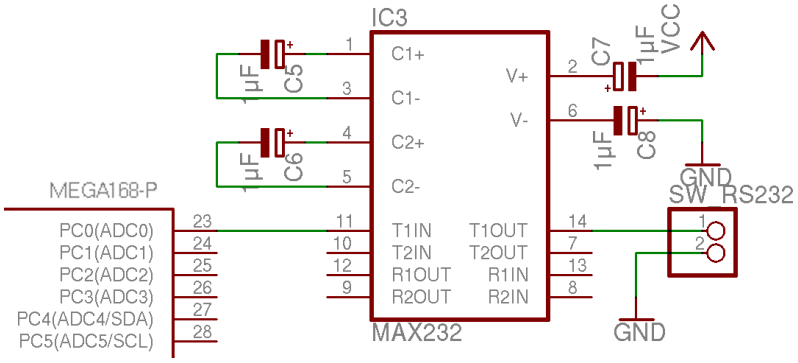


Abbildung 5.4: Schaltplan Schnittstelle RS232

Zur einfachen Programmierung des Mikrocontrollers ist noch eine ISP-Schnittstelle[6] in die Schaltung integriert worden. Sie führt die nötigen Pins des Controllers (MISO, MOSI, SCK und $\overline{RESET}$ mit Pullup Widerstand) über einen 10-poligen Wannenstecker nach „außen“.

5.2 Die Software

Zur Programmierung der Software für den Mikrocontroller wurde die Programmiersprache C verwendet. Zur Übersetzung des C-Quellcodes in Maschinensprache wurde der frei erhältliche Compiler 'gcc'[7] mit der Bibliothek 'avr-libc'[8] genutzt. Weiterhin gibt es vorgefertigte Bibliotheken zum Beispiel zur Nutzung der UART-Schnittstelle eines AVR-Mikrocontrollers, auf die ich allerdings nicht zurückgegriffen habe. Auf Grund des begrenzten Programmspeichers habe ich mich dazu entschieden die notwendigen Funktionen selbst zu schreiben, um diese möglichst klein zu halten.

Die zu entwickelnde Software muss zwei verschiedene Aufgaben erfüllen. Zum Einen kann der Mikrocontroller als lokaler Server bzw. Sender fungieren, zum Anderen kann er als mobiler Empfänger zum Einsatz kommen. Als lokaler Sender von Korrekturdaten muss die Software über die Hardware-UART, an der ein GPS-Modul angeschlossen ist, zusätzlich das Empfangen von Zeichenketten, den NMEA-Datensätzen, ermöglichen. Aus den so erhaltenen GPS-Informationen muss die Software auf Grund der vorgegebenen stationären GPS-Positionsdaten Korrekturwerte berechnen. Die so berechneten Korrekturwerte sollen die aktuellen, lokalen, spezifischen Fehler, die z.B. durch das Wetter beeinflusst wer-

[6] vgl Kapitel 3.1
[7] herunter zu laden unter: ftp.gnu.org/gnu/gcc/gcc-4.1.2/
[8] herunter zu laden unter: www.nongnu.org/avr-libc/

den, widerspiegeln. Über das an der SPI-Schnittstelle angeschlossene Funkmodul *RFM12* soll der Server dann diese Korrekturdaten versenden.

Andererseits muss die Software die Aufgaben des mobilen Empfängers implementieren. Im Gegensatz zum lokalen Sender muss hier die Software den Empfang von Korrekturdaten über das Funkmodul und deren Abspeicherung leisten. Analog zum Sender ist jedoch das Entgegennehmen von GPS-Mitteilungen über das an der Hardware-UART angeschlossene GPS-Modul. Aus den Positionsdaten des GPS-Moduls und den aktuell abgespeicherten Korrekturdaten muss die Software nun einen neuen berichtigten NMEA-Datensatz erstellen. Da der Mikrocontroller *ATmega168* über lediglich eine Hardware-UART verfügt, muss das Programm eine Software-UART-Schnittstelle implementieren, über die dann die Ausgabe dieses Datensatzes erfolgen kann. In Abbildung 5.5 ist ein schematischer Programm Ablauf Plan (PAP) des Hauptprogramms (*main.c*) abgebildet.

In der endgültigen Version der Software werden zur Korrektur der Positionsdaten im Empfänger nur die Werte für die GPS-Positionsangaben der Länge und Breite verwendet. Die zeitweise mit einbezogenen Werte der Anzahl der Satelliten und der GPS-Wert 'HDOP' wurden auf Grund nicht signifikanten Einflusses in der finalen Version nicht mehr verwendet. Mehr zu dieser Problematik ist bei der Auswertung der Testdaten in Kapitel 6.2.3 nachzulesen.

In Anhang A.9 befindet sich ein vollständiger Ausdruck des C-Quellcodes der Software. Dies umfasst sowohl die C-Dateien als auch die dazugehörigen Header-Dateien. Der komplette Quellcode ist auch auf der beiliegenden CD zu finden. In den folgenden Abschnitten werden die Header-Dateien sowie die einzelnen Funktionen der C-Dateien erläutert. Zu den im Programm verwendeten Registern und Bits sind weitere Informationen im Anhang A.6 und im Abkürzungsverzeichnis vorhanden.

5.2.1 main.h

Die Haupt-Header-Datei *main.h* bindet zunächst einmal alle Header-Dateien einiger im kompletten Programm verwendete GCC-Standardbibliotheken ein. Die Konstanten *TRANSMITTER*, *RECEIVER* und *GPGGA_MESSAGE* werden in mehreren Teilen des Programms benötigt und wurden deswegen in der Haupt-Header-Datei definiert. Die vier anschliessend definierten Konstanten sind grundlegende Einstellungen für das gesamte Programm. *F_CPU* bekommt als Wert die Frequenz des Mikrocontrollers in Hertz zugewiesen. Mit

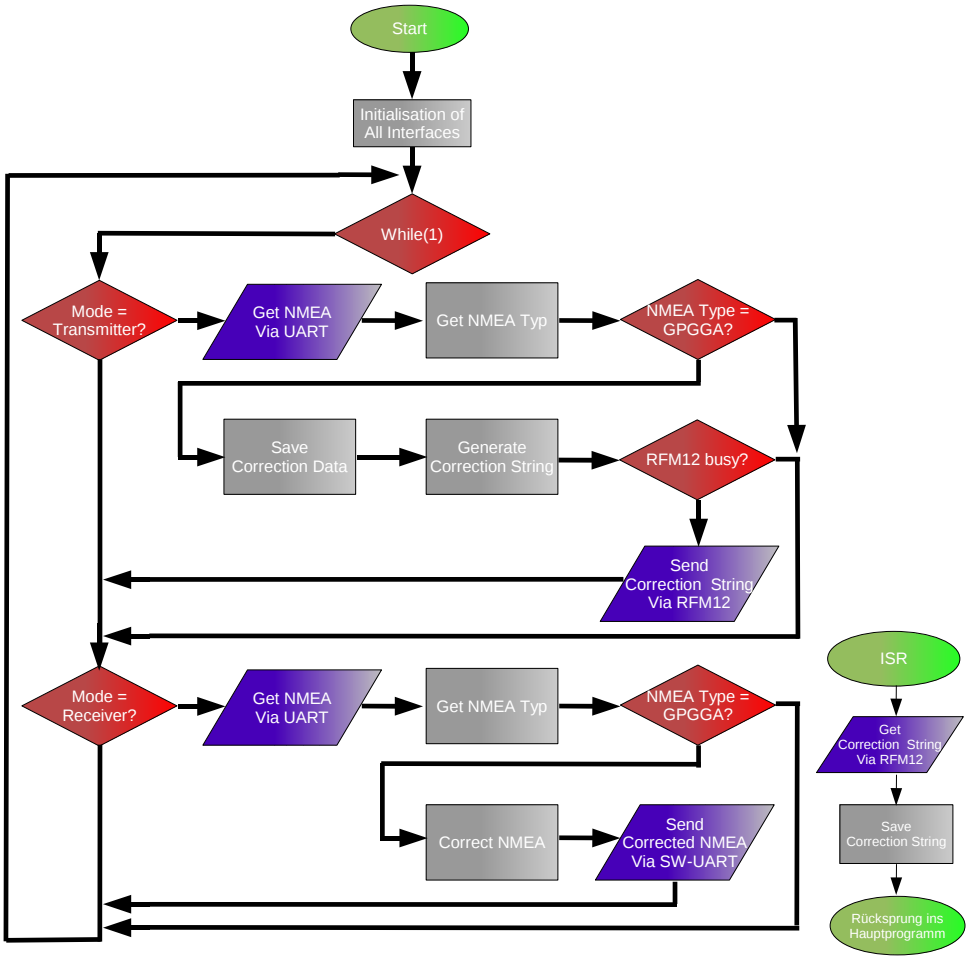


Abbildung 5.5: Schematischer Programm Ablauf Plan (PAP) des Hauptprogramms

UART_BAUD_RATE und *SW_UART_BAUD_RATE* werden die Baudraten für die Kommunikation über die Hardware-UART und Software-UART festgelegt. Die Konstante *MODE*, welche die Funktionsweise des Mikrocontrollers festlegt, kann entweder den Wert *TRANSMITTER* (1) oder *RECEIVER* (0) annehmen.

In der Header-Datei *main.h* wird ebenfalls eine Datenstruktur namens *convert* erstellt und umgehend eine Variable *CONVERT_TWO_BYTES* dieses Typs erzeugt. Diese Datenstruktur ist mit dem Schlüsselwort **union** versehen, was bedeutet, dass die einzelnen Variablen der Datenstruktur den gleichen Speicherplatz belegen. Die Variablen können auch unterschiedlichen Datentyps sein; der reservierte Speicherplatz richtet sich nach dem Platzbedarf des größten Elementes. In unserem Fall belegen die Variablen *two_bytes*, vom Typ integer, und das

zwei-elementige char-Array *byte* den gleichen Speicherplatz. Somit kann durch die Zuweisung eines 16-bit integer-Wertes zu dem Element *two_bytes*, ein recht einfacher Zugriff auf die beiden Felder des char-Arrays das „lower"- und das „higher"-Byte des integer-Wertes erfolgen.

Zum Schluss werden noch die einzelnen eigenen Header-Dateien des Projektes sowie die Header-Datei der Standardbibliothek 'delay' eingebunden, welche die Werte aus den vorherigen Definitionen der *main.h* benötigen.

5.2.2 uart.h

In der Header-Datei *uart.h* werden grundlegende Einstellungen für die Benutzung der Hardware-UART vorgenommen. Zuerst eingebunden wurde die Header-Datei *io.h*, eine Standardbibliothek des *AVR-GCC*, die für die Verwendung der Ports eines Mikrocontrollers zur Ein- und Ausgabe benötigt wird. Die folgenden beiden Definitionen der Konstanten *F_CPU* (Frequenz des Mikrocontrollers in Hz) und *UART_BAUD_RATE*(Baudrate der Hardware-UART) werden nur dann vorgenommen und es erfolgt eine Warnung, wenn diese noch nicht vorher in der Haupt-Header-Datei *main.h* definiert worden sind. Aus den Werten der beiden Konstanten wird dann in Zeile 16 die Konstante *UART_UBRR_VALUE* berechnet, welche zum Timing der UART verwendet wird. Die hierzu verwendete Formel habe ich aus dem Datenblatt[9] des *ATmega168* von Seite 164 übernommen.

Mir war wichtig den Quellcode möglichst einfach auf andere Mikrocontroller als den *ATmega168* portieren zu können. Zu diesem Zweck dienen die drei Konstanten *UART_DDR*, *UART_TX* und *UART_RX*, welche das mikrocontrollertypische DDR[10] und Pins für die Hardware-UART angeben. Im Fall des *ATmega168* sind Pin 0 (Eingangspin der UART) und Pin 1 (Ausgangspin der UART) des Port D für die Kommunikation über die Hardware-UART zuständig.

5.2.3 GPS.h

Zunächst wird in der Header-Datei *GPS.h* die *AVR-GCC*-Standardbibliothek *crc16* mittels '#include <util/crc16.h>' geladen. Die in dieser Bibliothek vorhandenen Funktionen werden in der Datei *GPS.c* zur Berechnung der CRC-Prüfsumme einer Korrekturdaten-Mitteilung benötigt.

[9] das Datenblatt des *ATmega168* befindet sich auf der beiliegenden CD
[10] Data Direction Register

Die GPS-Positionsdaten des stationären Servers werden in der Header-Datei in vier Konstanten gespeichert. Sowohl die Längen- als auch die Breitenkoordinate werden jeweils in zwei Integer-Variablen abgespeichert. Auf Grund von Rechenungenauigkeiten, nicht nur mit Variablen vom Typ float, sondern auch bei Variablen vom Typ double, habe ich die Koordinatenwerte in Vor- und Nachkommastellen aufgeteilt und in eigenen Integer-Variablen abgespeichert. Die Vorkommastellen der Längenkoordinate werden in der Konstanten *SERVER_LOCATION_LONGITUDE_PP*[11], die Nachkommastellen in der Konstanten *SERVER_LOCATION_LONGITUDE_DP*[12] abgelegt. Analog hierzu sind die Konstanten *SERVER_LOCATION_LATITUDE_PP* und *SERVER_LOCATION_LATITUDE_DP* für die Breitenkoordinate vorgesehen.

5.2.4 RFM12.h

In der Header-Datei *RFM12.h* werden zusätzlich zur Festlegung einiger Konstanten auch die grundlegenden Einstellungen des verwendeten Funkmoduls *RFM12* vorgenommen. Das definierte Makro *RFM12_FREQUENCY(frequency)* berechnet aus einer gegebenen Frequenz einen entsprechenden Wert[13], der zur Konfiguration der Sende- und Empfangsfrequenz in das verantwortliche Register 'Frequency Setting Command' des *RFM12* geschrieben werden kann. Der eigentliche Wert, der in das Register geschrieben wird, wird allerdings erst in Zeile 178 der Datei *RFM12.c* berechnet (vgl. hierzu Anhang A.9.8).

Die drei Konstanten *QUIET*, *NORMAL* und *NOISY* sind zur einfachen Konfiguration der Signalverstärkung in Abhängigkeit der Umgebung, in der gesendet werden soll, vorgesehen. Die folgenden vier Konstanten dienen zur Festlegung des Ports (*LED_PORT*) und der Pins (*LED_TX*, *LED_RX*, *LED_ERR*), an denen die drei Status-LED's für die Funkübertragung angeschlossen werden. Bei meiner Realisierung sind dies die Pins 3, 4 und 5 an Port C. In den Zeilen 17 bis 23 werden der Port und die Pins festgelegt, an denen das Funkmodul *RFM12* angeschlossen wird. Abschließend werden die grundlegenden Einstellungen des Funkmoduls mit Hilfe der Konstanten *RFM12_CHANNEL*, *RFM12_BAUDRATE*, *RFM12_ENVIRONMENT*, *RFM12_BASEBAND_FREQUENCY* vorgenommen.

[11] PP für Pre-Decimal Place
[12] DP für Decimal Place
[13] die Berechnung des Wertes ist im Datenblatt des *RFM12* (RFM12.pdf), welches sich auf der beiliegenden CD befindet, auf Seite 8 beschrieben

5.2.5 sw_uart.h

Der Aufbau der Datei *sw_uart.h* ähnelt sehr dem Aufbau der Datei *uart.h*. Die dort deklarierten Konstanten der Software-UART sind analog zu der Definition bezüglich der Hardware-UART. Zusätich wird jedoch in der *sw_uart.h* noch die Standardbibliothek 'interrupt' eingebunden, da das Timing der Software-UART über einen Interrupt-Timer des Mikrocontrollers realisiert wird. Zum Schluss wird noch der Port definiert über den die Software-UART laufen soll (hier Port C).

5.2.6 SPI.h

Die Header-Datei *SPI.h* dient der Definition einiger Konstanten zur Verwendung der SPI-Schnittstelle eines Mikrocontrollers. Wie auch in der *uart.h* (Kapitel 5.2.2) wurde Wert auf die Portabilität auf andere Mikrocontroller gelegt. Daher werden die Konstanten *SPI_PORT*, *SPI_DDR*, *SPI_SS*, *SPI_MOSI*, *SPI_MISO* und *SPI_SCK* mit den für den *ATmega168* spezifischen SPI-Schnittstellenwerten belegt. Da über die SPI-Schnittstelle Daten ausgetauscht werden, werden die beiden Standardbibliotheken 'io' und 'interrupt' mit eingebunden.

5.2.7 GPS.c

Die Datei *GPS.c* umfasst mehrere Funktionen, die zur Verarbeitung von NMEA-Datensätzen und Korrektur-Strings benötigt werden. Hierzu zählen die zwei Funktionen *String_to_integer* und *String_to_float*, welche in ähnlicher Form zwar in Standardbibliotheken vorhanden sind, aber auf Grund von Problemen bei der Verwendung nicht eingesetzt wurden. Mehr dazu in den entsprechenden Funktionsbeschreibungen.

Der Kern der Funktionen besteht jedoch in der Verarbeitung von empfangenen NMEA-Datensätzen, dem Erstellen und Verarbeiten von Korrekturstrings, sowie der Anpassung von NMEA-Datensätzen mittels dieser Korrekturdaten. Zur Datenspeicherung werden die globalen Variablen *Correction_Data_NS*, *Correction_Data_WE*, *Correction_Data_Satellites* und *Correction_Data_hdop* deklariert. Das zweidimensionale char-Array *NMEA_data* dient zur Zwischenspeicherung der relevanten Daten eines empfangenen NMEA-Datensatzes in String-Form.

5.2.7.1 String_to_integer(char* String)

Diese Funktion wandelt eine Zeichenkette, die durch einen Pointer auf ein Char-Array übergeben wird, in eine Integerzahl um und gibt diese zurück. Die in der Bibliothek 'stdlib' vorhandene Funktion wurde nicht verwendet, da durch eine negative Zahl und/oder Leerstellen bei der Konvertierung Probleme entstanden. Mit Hilfe der while-Schleife durchläuft der Zeiger auf das char-Array (Zeile 50 bis 60) dieses von vorn nach hinten. Durch die Abfrage von *help!=32* wird auf ein eventuell vorhandenes Abschluss-Zeichen, das nicht zur Zahl gehört, geprüft und, falls vorhanden, nicht zur Wertberechnung verwendet. Die innere if-Abfrage kontrolliert, ob das aktuelle Zeichen, auf das der Pointer zeigt und das in *help* zwischengespeichert ist, ein Minuszeichen ist (ASCII-Code 45[14]). Ist dies der Fall, wird die am Anfang mit Null initialisierte Variable *minus* auf 1 gesetzt und somit die Zahl als negative Zahl identifiziert. Andernfalls wird davon ausgegangen, dass das aktuelle Zeichen eine Ziffer darstellt. Zur Berechnung des Zahlenwertes des Zeichens wird von dem in *help* abgespeicherten ASCII-Wert 48 abgezogen und zum aktuellen Berechnungswert des bisherigen Strings hinzuaddiert, welcher zuvor durch Multiplikation mit 10 auf die nächste Ziffernstelle vorbereitet wurde.

Nachdem das char-Array komplett durchlaufen wurde, wird in der if-Abfrage geprüft ob ein Minuszeichen vor der Zahl stand. Ist dies der Fall gewesen, so wird der bisher berechnete Wert der Zeichenkette mit (-1) multipliziert und somit negiert.

5.2.7.2 String_to_float(char* String)

Aus ähnlichen Gründen wie bei der Funktion *String_to_integer* (vgl. Kapitel 5.2.7.1), wurde auch eine Funktion zur Umwandlung von einer Zeichenkette in eine float-Zahl benötigt. Hierzu wurde mit Hilfe der Standardfunktion *strsep* vom übergebenen String (Pointer auf ein char-Array) die Vorkommastellen extrahiert. Die eben erwähnte Funktion *String_to_integer* wandelt diesen Teilstring in eine Zahl um. Danach wird überprüft ob die Zahl negativ ist. Für diesen Fall wird die Zahl durch die Multiplikation mit (-1) wieder positiv und die Variable *minus* wird auf 1 gesetzt, um zu vermerken, dass eine negative Zahl vorliegt. *minus* wird dann zum Schluss der Funktion überprüft und eventuell durch eine weitere Multiplikation mit (-1) negiert.

[14] siehe Anhang A.3

Nach dem erneuten Aufruf von *strsep* erhalten wir den Reststring mit den Nachkommastellen. In der while-Schleife wird jedes ASCII-Zeichen in den entsprechenden Integer-Wert umgewandelt (vgl. Funktion *String_ to_ integer*) und durch die Multiplikation mit *factor*, welcher durch Potenzieren von 0,1 entsteht und den Wert an die entsprechende Dezimalstelle verschiebt, berechnet und zum bisherigen Wert von *value* hinzuaddiert.

5.2.7.3 get_NMEA_Type(char *NMEA_String)

Diese Funktion dient zur Ermittlung des NMEA-Message-Typs der übergebenen NMEA-Zeichenkette. Die Funktion *strstr* aus der Standardbibliothek *string* sucht einen String (erstes Argument, Pointer auf char-Array) nach dem Vorkommen eines bestimmten Teilstrings (zweites Argument, Pointer auf char-Array) ab. Falls der Teilstring in der zu durchsuchenden Zeichenkette vorhanden ist, liefert die Funktion den Pointer (gespeichert in *help*) auf das erste Zeichen des Teilstrings in der zu durchsuchenden Zeichenkette. Mit Hilfe der darauf folgenden if-Anweisung gibt die Funktion *get_ NMEA_ Type* einen in der Datei *main.h* definierten Wert zurück, sofern der Pointer in *help* ungleich dem NULL-Pointer ist.

5.2.7.4 tokenize_NMEA_data(char *NMEA_String)

tokenize_ NMEA_ data filtert aus einer übergebenen kompletten NMEA-Nachricht (Pointer auf char-Array) die für die Korrektur relevanten Informationen heraus und speichert diese im zweidimensionalen char-Array *NMEA_ Data* ab. Dazu wird an die entsprechenden Punkte des char-Array gesprungen, die notwendige Anzahl an Zeichen mit Hilfe der Funktion *memcpy* in das Array *NMEA_ Data*-Feld kopiert und an der betreffenden Stelle durch '\0' abgeschlossen. Dieser Vorgang wird in der folgenden Reihenfolge der einzelnen Informationen wiederholt:

1. Vorkommastelle der Längenangabe
2. Nachkommastelle der Längenangabe
3. Vorkommastelle der Breitenangabe
4. Nachkommastelle der Breitenangabe
5. Anzahl der Satelliten
6. horizontale Positionsgenauigkeit HDOP

5.2.7.5 save_Correction_Data(char* NMEA_String)

Das durch die Funktion *tokenize_NMEA_Data* erzeugte zweidimensionale char-Array wird in dieser Funktion als Ausgangspunkt für die Berechnung der Korrekturwerte verwendet. Zunächst werden zur Berechnung der beiden Längenkoordinaten-Korrekturwerte, die der Differenz der aktuellen Koordinaten zu den festen Server-Koordinaten entsprechen, die Nachkommastellen des aktuellen NMEA-Datensatzes (mit Hilfe der Funktion *String_to_integer* in eine Zahl umgewandelt) von den festen Nachkommastellen der Längenkoordinaten des Servers subtrahiert. Weichen die Vorkommastellen des aktuellen NMEA-Datensatzes von denen der festen Server-Koordinantenvorkommastellen ab, muss dieser Sprung durch die Subtraktion des berechnenten Korrekturwertes von 10000 zur Berechnung des Unterschiedes der Koordinaten berücksichtigt werden.

Die Berechnung der beiden Breitenkoordinaten-Korrekturwerte verläuft analog zur Ermittlung der Längenkoordinaten-Korrekturwerte. Die „Korrekturwerte“ für die Anzahl der Satelliten und von HDOP, werden einfach durch Umwandlung der Zeichenkette in eine Zahl „berechnet“.

5.2.7.6 generate_Correction_String(char* Correction_String,char* NMEA_String)

Der mit Hilfe der Funkmodule zu übertragende String beinhaltet die Korrekturdaten des stationären Servers. Dieser Korrekturstring wird durch die Funktion *generate_Correction_String* erzeugt. Zunächst wird das Startsymbol '$' an den Anfang des mit Hilfe eines Pointers übergebenen char-Array gesetzt. Danach wird der Korrekturwert für den Längengrad (*Correction_Data_NS*) mit der Funktion *dtostrf* umgewandelt und in dem Hilfsstring *hstring* gespeichert. Dieser Hilfsstring wird mit der Funktion *strcat* an den bestehenden Korrekturstring *Correction_String* angehängt, gefolgt von einem Komma. Analog hierzu wird der Korrekturwert für den Breitengrad dem Korrekturstring hinzugefügt.

Die Anzahl der Satelliten, von denen ein Signal empfangen wird, ist ebenfalls Teil des Korrekturstrings. Hier wird mit der if-Anweisung eine Unterscheidung zwischen ein- und zweistelliger Anzahl durchgeführt, um eine Leerstelle zu vermeiden. Die Funktion *dtostrf* wandelt die Anzahl dementsprechend in einen Ein- oder Zweizeichen String um (2. Argument der Funktion). Auch nach dem Hinzufügen dieser Zeichenkette wird dem Korrekturstring *Correction_String* ein Komma angehängt. Als letzer Korrekturwert wird der HDOP-Wert eingear-

beitet. Bei der Umwandlung von *Correction_Data_hdop* ist jedoch zu beachten, dass *dtostrf* eine Nachkommastelle berücksichtigen muss (3. Argument der Funktion).

Um die korrekte Übertragung des Korrekturstrings zu überprüfen wird eine CRC-Prüfsumme verwendet. Zur Abgrenzung von den eigentlichen Nutzdaten (Korrekturdaten) wird an das Ende des bisherigen *Correction_String* ein Stern ('*') gehängt. In den Zeilen 227 und 228 wird eine Kopie des Zeigers auf den *Correction_String* erzeugt und um ein Zeichen nach rechts gesetzt. Mit der for-Schleife wird der String weiter bis zum vorletzten Zeichen durchlaufen und jeweils das aktuelle Zeichen mit der Funktion *_crc_ibutton_update* aus der Bibliothek *crc16* in die Berechnung der CRC-Prüfsumme einbezogen. Auf diese Weise wird die Prüfsumme des Korrekturstrings zwischen den Zeichen '$' und '*' berechnet. Diese Prüfsumme wird als ASCII-Zeichen in das erste Feld des Hilfsarrays *tail* geschrieben. Als weitere Werte werden die ASCII-Werte 13 (Carriage return) und 10 (Line feed) als Abschlusszeichen in Feld zwei und drei des zu übertragenden Korrekturstrings gesetzt. In das letzte Feld wird das Abschlusssymbol für einen String ('\0') in C geschrieben. Zum Schluss wird das Hilfsarray an das Ende des bisherigen Korrekturstring gehängt.

```
236   char tail[4];
237   tail[0]=crc;
238   tail[1]=13;
239   tail[2]=10;
240   tail[3]='\0';
241   strcat(Correction_String,tail);
```

Quellcode 5.1: Abschließen des NMEA-Strings

5.2.7.7 decode_Correction_String(char* Correction_String)

Die Funktion *decode_Correction_String* ist das Gegenstück zur Funktion *generate_Correction_String*. Während auf der Senderseite der zu übertragende String erzeugt wird, werden auf der Empfängerseite der empfangene String mittels CRC-Prüfsumme überprüft und die Nutzdaten extrahiert. Nach der Deklaration einiger Hilfsvariablen wird durch die Funktion *strsep* vom empfangenen String die CRC-Prüfsumme, welche durch das Symbol '*' von den Nutzdaten getrennt ist, abgespalten und in *received_crc* gespeichert. Wie in der Funktion *generate_Correction_String* wird der Zeiger auf den Korrekturstring

um ein Zeichen nach rechts gesetzt und mit der for-Schleife und der Funktion *_crc_ibutton_update* die CRC-Prüfsumme des Korrekturstrings berechnet. Ist die errechnete Prüfsumme *crc* gleich der abgespaltenen CRC-Prüfsumme *received_crc*, war die Übertragung erfolgreich und die Korrekturdaten können aus den empfangenen Nutzdaten *received_Data* ermittelt werden. Andernfalls ist die Übertragung fehlerhalft gewesen, die Korrekturdaten des Empfängers werden nicht aktualisiert und die Error-LED wird aktiviert.

Um aus den Nutzdaten *received_Data* die aktuellen Korrekturwerte zu ermitteln, werden diese in der while-Schleife mit der Funktion *strsep* und dem Trennzeichen ',' Wert für Wert als String herausgelöst und in der Variablen *Data* gespeichert. Mit Hilfe der beiden Funktionen *String_to_integer* und *String_to_float* sowie der Variablen *counter*, welche die Anzahl der Argumente zählt, werden die entsprechenden Korrekturwerte berechnet und in den globalen Variablen *Correction_Data_NS*, *Correction_Data_WE*, *Correction_Data_Satellites* und *Correction_Data_hdop* gespeichert.

5.2.7.8 hex(uint8_t i)

Diese Funktion wandelt eine eingehende integer-Zahl (von 0 bis 15) in das entsprechende ASCII-Zeichen um. Die Werte 0 bis 9 werden durch die Addition von 48 auf das jeweilige ASCII-Zahlen-Symbol gesetzt. Die Werte 10 bis 15 bekommen durch die Addition von 55 die betreffenden ASCII-Buchstaben-Symbole 'A' bis 'F' zugewiesen.

5.2.7.9 correct_NMEA(char* NMEA_String)

Wie die Funktion *decode_Correction_String*, ist auch *correct_NMEA* eine Funktion des Empfängers. Sie korrigiert den über die Hardware-UART-Schnittstelle empfangenen NMEA-String mit den über das Funkmodul empfangenen Korrekturdaten. Der über UART empfangene NMEA-String wird der Funktion mittels eines Pointers auf das entsprechende char-Array übergeben. Die aktuellen Korrekturdaten wurden durch die Funktion *decode_Correction_String* in den globalen Variablen *Correction_Data_NS*, *Correction_Data_WE*, *Correction_Data_Satellites* und *Correction_Data_hdop* gespeichert.

Im Anschluss an die Definition einiger Variablen wird der aktuell erhaltene NMEA-String des GPS-Moduls mit Hilfe der Funktion *tokenize_NMEA_data* (vgl. Abschnitt 5.2.7.4) zerlegt und die verschiedenen Positionsdaten in dem

zweidimensionalen char-Array *NMEA_data* abgelegt. Auch hier wird die Längen- und Breitenangabe jeweils in Vor- und Nachkommastelle zerlegt. Die Gründe hierfür sind in Kapitel 5.2.3 zu finden.

Die in diesem Absatz folgenden Erläuterungen beziehen sich auf den Quellcode 5.2. Zunächst wird die Breitenkoordinate des aktuellen NMEA-Strings mit den aktuellen Korrekturdaten berichtigt. Hierzu wird als erstes zu der, durch *String_to_integer* in eine integer-Zahl umgewandelten Nachkommastellen, der derzeitige Breitenkorrekturwert hinzuaddiert und in der Variablen *corrected_DP* gespeichert. In der Variablen *corrected_PP* wird die umgewandelte aktuelle Vorkommastelle des Breitengrades abgelegt. Durch die beiden folgenden if-Anweisungen wird einem eventuellen, durch die Korrekturberechnung der Nachkommastelle entstandenen Übertrag auf die Vorkommastelle, Rechnung getragen. Ist die korrigierte Nachkommastelle negativ wird die Vorkommastelle um Eins dekrementiert und die Nachkommastelle dementsprechend um 10000 erhöht. Ist die korrigierte Nachkommastelle größer als 10000 muss die Vorkommastelle um Eins inkrementiert und von der Nachkommastelle 10000 abgezogen werden.

```
328   corrected_PP = String_to_integer(NMEA_data[1]) +
          Correction_Data_NS;
329   corrected_DP = String_to_integer(NMEA_data[0]);
330   if (corrected_PP<0){
331     corrected_DP = corrected_DP - 1;
332     corrected_PP = corrected_PP + 10000;
333   }
334   if (corrected_PP>=10000){
335     corrected_DP = corrected_DP + 1;
336     corrected_PP = corrected_PP - 10000;
337   }
338
339   dtostrf(corrected_DP,4,0,corrected_String);
340   strcat(corrected_String,".");
341   dtostrf(corrected_PP,4,0,hstring);
342   strcat(corrected_String,hstring);
343   strcat(corrected_String,",N,");
```

Quellcode 5.2: Erstellen eines korrigierten Strings

Mit *dtostrf* wird dann die korrigierte Vorkommastelle in einen String zurückgewandelt und in *corrected_String* abgelegt. In den folgenden Zeilen werden mit

strcat an diesen *corrected_String* ein Dezimalpunkt, die umgewandelten Nachkommastellen (welche in der Hilfsvariablen *hstring* zwischengespeichert wurden) und die Zeichen ',N,' angefügt. Analog zu dieser Vorgehensweise wird in den Zeilen 345 bis 360 der *corrected_String* um die Informationen des Längengrades ergänzt.

Nachdem der korrigierte *corrected_String* erzeugt wurde, muss dieser noch in den originalen NMEA-String, der vom GPS-Modul empfangen wurde, eingearbeitet werden. Hierzu wird zunächst der Pointer auf den NMEA-String in die Variable *help_Pointer* kopiert. Dann wird der Pointer um 18 Stellen nach rechts verschoben und steht somit an der ersten Zeichenposition der Breitengrad-Vorkommastelle. Nun wird in der while-Schleife und mit Hilfe der Zählvariablen *counter* sowohl das char-Array von *corrected_String*, als auch die folgenden Stellen ab der Position, auf die der Pointer *help_Pointer* zeigt, bis zum Ende von *corrected_String* durchlaufen. Ist das aktuelle Zeichen des char-Array *corrected_String* gleich dem Leerzeichen (ASCII-Code 32) wird eine Null ('0') als führende Null der Koordinate an die entsprechende Stelle des NMEA-Strings geschrieben. Diese ist nötig, da im NMEA-Standard die Anzahl der Stellen der Koordinaten festgelegt ist und es keine Leerstellen im NMEA-String geben darf. Ist das aktuelle Zeichen des char-Array *corrected_String* kein Leerzeichen, so wird dieses Zeichen direkt an die betreffende Stelle des NMEA-Strings geschrieben.

Durch das stellenweise Überschreiben des originalen NMEA-Strings mit den korrigierten Positionskoordinaten ändert sich natürlich auch die NMEA-Prüfsumme (vgl. Anhang A.4.1) des Strings. Da über die Software-UART eine NMEA-konforme Schnittstelle implementiert werden soll, muss auch die NMEA-Prüfsumme auf den neu erzeugten String angepasst werden. Der Startwert der Prüfsumme ist Null. Mit der for-Schleife wird der neue NMEA-String zwischen '$' und '*' durchlaufen und der ASCII-Wert des aktuellen Zeichens mit dem vorherigen Prüfsummenwert mittels XOR-Operation verrechnet. Die am Ende berechnete Prüfsumme ist ein 8-bit Wert, der durch die Funktion *hex* in zwei Hexadezimalzeichen umgewandelt wird (Zeilen 386 und 387). Zum Schluss wird die alte Prüfsumme des NMEA-Strings durch die beiden Hexadezimalwerte *crc_String_1* und *crc_String_2* überschrieben. Der nun vollständig aktualisierte NMEA-String kann jetzt über die Software-UART (siehe main.c in 5.2.12.1) ausgegeben werden.

5.2.8 RFM12.c

RFM12.c ist eine C-Datei, die mehrere Funktionen zur Initialisierung, Konfiguration sowie Funktionen zum Senden und Empfangen von Daten über das Funkmodul *RFM12* bereitstellt. Die Kommunikation mit dem Funkmodul wird über die SPI-Schnittstelle und die dazugehörige Datei *SPI.c* bereitgestellt. Die Befehle, die zur Konfiguration und dem Senden und Empfangen von Daten notwendig sind, können aus den auf der CD befindlichen Datenblättern des *RFM12* entnommen werden. Auf der CD sind mehrere Versionen von Datenblättern zum Funkmodul enthalten, da diese in sich nicht stimmig sind und auch an einigen Stellen Fehler enthalten, was die Inbetriebnahme des Funkmoduls teilweise sehr erschwerte.

Die Struktur der 16-Bit Befehle mit denen das Funkmodul angesteuert werden kann, ist größtenteils gleich. Die acht höherwertigen Bits geben den Befehl an, während die acht niederwertigen Bits die Parameter umfassen. Als Beispiel soll hier einmal der Befehl 'Low Battery Detector and Microcontroller Clock Divider Command' dienen. Die acht höherwertigen Bits beinhalten den eigentlichen Befehl und besitzen den festen Wert '0xC0'[15]. Die restlichen acht Bits dienen der Einstellung einer Grenzspannung (fünf Bits) und der Festlegung einer Frequenz (drei Bits) (vergleiche hierzu *RFM12* Datenblatt 'RFM12.pdf' Seite 13). Der Wert dieser acht Bits wird ebenfalls in hexadezimaler Form notiert. Im Folgenden wird ein Befehl in der Form 'C0xx' dargestellt. Im Datenblatt des *RFM12* ('RF12B.pdf') befinden sich die verwendeten Befehle sowie ihre Standarteinstellung bei Power On Reset (POR).

Viele Funktionen in dieser Datei dienen der Berechnung eines Befehls und dessen Übermittlung an das Funkmodul. Diese sind zumeist mit *RFM12_set_xxxx* bezeichnet worden. Zur Berechnung des gewünschten Befehls wird als Ausgangsbasis der feste Befehlswert (z.B. '0xC0') und als Wert aller Parameter der Wert '0x00' (also acht Nullen) verwendet. Mit der oder-Funktion '|' und der Operation '<<' zum Bitverschieben der Programmiersprache C werden die einzelnen Bits der Parameter gesetzt und der gesamte Befehl direkt mit Hilfe der Funktion *RFM12_transmit* an das Funkmodul *RFM12* übertragen.

Die Variablen, die am Anfang der Datei deklariert werden, sind funktionsübergreifend und müssen deshalb global zugänglich sein. Die Variablen *Received_Data*, *new_Char*, *received_Char*, *Correction_String* und *Correction_String_Counter* werden in einer Interrupt Service Routine zum Abspei-

[15] 0x als Präfix für die hexadezimale Darstellung

chern von Daten oder Zuständen verwendet, die asynchron abläuft. Die Variable *busy* ist ein globales Flag, welches den Beschäftigungszustand des Funkmoduls festhält.

5.2.8.1 RFM12_transmit(unsigned short value)

Diese Funktion übernimmt die Kommunikation des Mikrocontrollers mit dem über die SPI-Schnittstelle angeschlossenen Funkmodul *RFM12*. Der Datenaustausch ist bidirektional und findet byte-weise statt. Da die meisten *RFM12*-Befehle aus einem Kommando-Byte und einem Daten-Byte bestehen, ist der Eingabeparameter der Funktion *value* zwei Byte groß. Dieser Wert wird in die union-Variable *val* geschrieben und steht somit byte-weise als *val.byte*[0] und *val.byte*[1] bereit[16]. Mit den Fuktionen *SPI_ on* und *SPI_ off* wird das Funkmodul *RFM12* zunächst über die SPI-Schnittstelle als Slave selektiert und später wieder deaktiviert. In der Zwischenzeit wird zuerst das high-Byte *val.byte*[1] und dann das low-Byte *val.byte*[0] über die SPI gesendet und die jeweiligen Antwortdaten in den entsprechenden Variablen gespeichert. Die so empfangenen Daten gibt die Funktion dann wieder als zusammengefassten zwei Byte großen Wert zurück.

5.2.8.2 RFM12_set_Signal(unsigned char bandwidth, unsigned char gain, unsigned char drssi)

RFM12_set_Signal konfiguriert die grundlegende Signalcharakteristik des *RFM12*-Funksignals. Der Befehlscode für das 'Receiver Control Command' ist '0x9xxx'. Als feste Konfiguration wird Pin 20 des *RFM12* als VDI-Output und die VDI-Response-Time auf 'medium' gesetzt[17]. Somit ergibt sich zunächst der Wert 'Ox95xx'[18]. Die restlichen Einstellungen werden der Funktion als Parameter *gain*, *bandwidth* und *drssi* übergeben.

5.2.8.3 RFM12_set_Frequency(unsigned short frequency)

Die Frequenz, mit der das Funkmodul *RFM12* senden bzw. empfangen soll, wird mit Hilfe dieser Funktion eingestellt. Das Basis-Frequenzband der verwendeten *RFM12*-Funkmodule ist das 433Mhz-Band. Innerhalb dieses Bereiches kann das

[16] vgl. hierzu Deklaration der union-Struktur in Kapitel 5.2.1
[17] vgl. hierzu Datenblatt 'RFM12_code.pdf' Seite 3 (auf beiliegender CD vorhanden)
[18] vergleiche Datenblatt des RMF12

Modul 3867 verschiedene Frequenzen zur Funkübertragung nutzen. Die zu nutzende Frequenz kann mittels des Frequency Setting Command '0xAxxx' ausgewählt werden. Der entsprechende Wert der Frequenz, der mit dem Kommando übertragen wird, muss laut Datenblatt zwischen 36 und 3903 liegen, was durch die if- bzw. else if-Abfrage sichergestellt wird.

5.2.8.4 RFM12_set_Baud_Rate(unsigned short Baud_Rate)

Die Funktion *RFM12_set_Baud_Rate* dient dem Festlegen der Baudrate die zur Funkübertragung genutzt werden soll. Die *RFM12* Funkmodule können mit einer Baudrate von 600 bps bis hin zu 115200 bps (115,2 kbps) betrieben werden. Hierzu stehen zwei verschiedene Modi zur Verfügung. Einmal die 'Clock Recovery' im 'slow mode' und einmal im 'fast mode'. Welcher Mode verwendet werden soll wird über das 'cs'-Bit im 'Data Rate Command' (0xC6xx) eingestellt (vgl. Datenblatt 'RF12B.pdf'). Die Baudraten vom 'slow mode' und 'fast mode' überschneiden sich zwischen ca. 2 kbps und ca. 42 kbps. Auf Grund der Fehlerraten bei der Übertragung wird mittels der if-Anweisung in Zeile 102 bei einer Baudrate von unter 5400 bps der 'slow mode' verwendet und ab einer Baudrate von 5400 bps der 'fast mode'.

Die restlichen sieben Bits des 'Data Rate Command' sind für das Justieren der Baudrate zuständig. Der Wert dieser 7-bit-Variablen (im Datenblatt mit R bezeichnet, vgl. Datenblatt 'RFM12.pdf') wird über folgende Formel aus dem Datenblatt berechnet:

$$R = (10000/29/(1 + cs * 7)/Baudrate\ in\ kbps) - 1$$

Daraus folgt gerundet:

- $R = (\frac{43104}{Baudrate\ in\ kbps}) - 1$, für den 'slow mode'
- $R = (\frac{344828}{Baudrate\ in\ kbps}) - 1$, für den 'fast mode'

5.2.8.5 RFM12_set_Power(unsigned char power, unsigned char deviation)

Neben den grundlegenden Signalcharakteristika, die über die Funktion *RFM12_set_Signal* (siehe Abschnitt 5.2.8.2) eingestellt werden könnnen, kann man auf weitere Eigenschaften des Signals Einfluss nehmen. Mit den drei niederwertigsten Bits des Befehls '0x98xx' kann die Ausgangsleistung des zu sendenden Signals reguliert werden (p0 bis p2). Die nächsten fünf höherwertigen

Bits dienen der Einstellung der FSK[19]-Modulation des Funkmoduls, welches mit der in *RFM12_set_Signal* konfigurierten Trägerfrequenz arbeitet.

5.2.8.6 RFM12_ready(void)

Mit dem Schlüsselwort **inline** wird erreicht, dass der in der Funktion verwendete Assembler-Code nicht durch den Compiler „wegoptimiert“ wird. Um die Kommunikation mit dem *RFM12* zu starten, muss dieses zunächst über die SPI-Schnittstelle aktiviert werden. Da die Aktivierung einen kurzen Moment dauert, wird der Mikrocontroller über die Assembler-Anweisung *asm("nop")*[20] zwei Arbeitstakte ruhen. Danach verweilt er in einer Warteschleife bis das Funkmodul *RFM12* den SDO-Pin auf 'high' legt, und somit seine Betriebsbereitschaft signalisiert.

5.2.8.7 RFM12_RX_Mode(void)

Standardmäßig befinden sich die *RFM12* Funkmodule in einem empfangsbereiten Zustand, der nur zum Senden von Daten verlassen wird. Dieser wird mit dem Aufruf der Funktion *RFM12_RX_Mode* erreicht. Zunächst wird die globale Variable *busy* auf Null gesetzt. Sie signalisiert innerhalb der Mikrocontroller-Software, dass zur Zeit nichts gesendet werden soll und sich das Funkmodul im Empfangsmodus befindet. Danach wird das Modul durch *RFM12_transmit(0x82C8)* mit den entsprechenden Parameter ('er' und 'ebb')[21] in einen empfangsbereiten Zustand versetzt.

Da zum Empfangen von Daten der FIFO-Puffer des *RFM12* benutzt werden soll, muss dieser noch konfiguriert und aktiviert werden. Im FIFO-Mode kann man die sogenannte 'SYNC-Word-Detection' verwenden. Ist diese aktiviert, wartet das Modul bis die Daten '0x2D' und '0xD4' empfangen wurden. Sobald dieses Bitmuster erkannt wurde, schreibt das Funkmodul alle weiteren Daten in den FIFO-Puffer bis der FIFO-Mode deaktiviert und die Mustererkennung neu startet wird. Zum Konfigurieren des FIFO-Puffers (8-Bit FIFO Interrupt[22]) und Aktivieren der 'SYNC-Word-Detection' wird der Befehl '0xCA81' an das Modul geschickt. Da das Modul etwas Zeit benötigt um die 'SYNC-Word-Detection' zu starten, wird erst 0,8 ms gewartet bis mit '0xCA83' und *RFM12_transmit(0)*

[19] Frequency Shift Keying
[20] nop ≙ no operation
[21] Datenblatt 'RF12B.pdf' Seite 14
[22] Wenn das Funkmodul 8 Bit empfangen hat, löst es über den FFIT-Anschluss einen Interrupt aus

der FIFO-Mode aktiviert wird. Die beiden letzten Zeilen der Funktion aktivieren den Interrupt 'INT1' des Mikrocontrollers, über den das *RFM12* den vollständigen Empfang eines Bytes signalisieren kann (FFIT).

5.2.8.8 RFM12_Stop_RX(void)

RFM12_Stop_RX beendet den standardmäßigen Empfangsmodus. Zum Datenversand wird die globale Variable *busy* auf Eins gesetzt (vgl. 5.2.8.7). Durch Deaktivierung des Interrupts 'INT1' am Mikrocontroller kann das Funkmodul den Mikrocontroller nicht mehr bei seiner Arbeit unterbrechen. Mit dem Befehl *RFM12_transmit(0x8208)* wird die Empfangsfunktion des *RFM12* gestoppt. Zum Schluss wird noch eine Millisekunde auf das Ende der Abarbeitung des Kommandos gewartet.

5.2.8.9 RFM12_config(unsigned short baudrate, unsigned char channel, unsigned char power, unsigned char environment)

Mit der Funktion *RFM12_config* werden die grundlegenden Einstellungen des *RFM12* Funkmoduls vorgenommen. Die Eingabeparameter *baudrate*, *channel*, *power* und *environment* werden neben anderen festen Konfigurationsparametern an die entsprechenden Funktionen weitergegeben. Die in der *RFM12.h* festgelegte Konstante *RFM12_BASEBAND_FREQUENCY* und der Parameter *channel* legen mittels der Funktion *RFM12_set_Frequency* und dem Makro *RFM12_FREQUENCY* (siehe 5.2.4) die Empfangs- bzw. Sendefrequenz des Moduls fest [23]. *RFM12_set_Power(0, 5)* legt eine Ausgangsleistung von 6mW und einen Frequenzshift von 90kHz bei der Modulation fest. Durch *RFM12_set_Signal(4, environment, 0)* wird eine Bandbreite von 200kHz und ein DRSSI-Schwellenwert von -103dBm eingestellt. Der Grad der Verstärkung ist abhängig vom Eingabeparameter *environment*. Zur Festlegung der Baudrate wird einfach der entsprechende Wert *baudrate* an die Funktion *RFM12_set_Baud_Rate* weitergegeben.

5.2.8.10 RFM12_init(void)

Mittels dieser Funktion werden neben der Initialisierung des Funkmoduls auch einige weitere Variablen vorbelegt. Die ersten drei Zeilen dienen der Konfiguration des Ports, über den das Modul an den Mikrocontroller angeschlossen

[23] *RFM12_BASEBAND_FREQUENCY* + (*channel* * 325kHz)

ist, also der SPI-Schnittstelle. In den beiden folgenden Zeilen wird der Pin des Interrupts 'INT1' als Eingang konfiguriert und der interne Pull-up-Widerstand aktiviert. Durch Setzen der entsprechenden Bits im SPI Control Register (SPCR) wird nun der Mikrocontroller zum SPI-Master deklariert.

Da das Funkmodul bei einem Kaltstart einige Zeit zum Abarbeiten seiner Startroutinen benötigt, wird durch die while-Schleife 200 ms auf das Modul gewartet. Tabelle 5.1 zeigt einige Grundeinstellungen, die dann zur Initialisierung des Moduls vorgenommen werden.

Kommando	**Bedeutung**
0xC0E0	Clock Output Frequency 10 Mhz Low Battery Detector Grenzspannung 2,2V (standardmäßig deaktiviert)
0x80D7	Aktivieren des Internal Data Register aktiviert FIFO-Mode Crystal Load Capacitance 16 pF (Standard)
0xC2AB	Aktivieren von Clock Recovery Auto Lock Control internal Digital Filter DQD (data quality detection) Level 3
0xCA81	FFIT Bitanzahl: 8 Synchron Pattern FIFO deaktiviert Deaktivieren von highly sensitive RESET Mode
0xE000	Deaktivieren des Wake-Up-Timers
0xC800	Deaktivieren des Low Duty Cycles
0xC4F7	AFC (Automatic Frequency Control) unabhängig von VDI Autotuning von -2,5kHz bis 7,5kHz um die Frequenz genauer Berechnungsmodus AFC-Offset freischalten AFC-Berechnung aktivieren

Tabelle 5.1: Grundeinstellungen des Funkmoduls *RFM12*

Am Ende der Funktion werden noch die drei globalen Variablen *received_Char*, *busy* und *Correction_String_Counter* mit Null initialisiert, bevor durch Aufruf von *RFM12_RX_Mode* das Funkmodul in den wartenden Empfangsmodus gesetzt wird.

5.2.8.11 RFM12_TX_Byte(unsigned char value)

RFM12_TX_Byte dient der Übertragung eines Datenbytes und wird in der Funktion *RFM12_send_Byte* aufgerufen. Hiermit werden die zur Übertragung

einleitenden bzw. abschließenden Schritte vorgenommen (z.B. das Senden der SYNC-Daten).

Nachdem mit *RFM12_ready()* darauf gewartet wurde, dass das *RFM12*-Modul sendebereit ist, wird das eigentliche Datenbyte übertragen. Der Parameter, der *RFM12_transmit* übergeben wird, berechnet sich aus der ODER-Verknüpfung des Transmitter 'Register Write Command' '0xB800' und des Datenbytes. Falls das Datenbyte den Wert '0x00' oder '0xFF' besitzt, also nur aus Nullen bzw. Einsen besteht, wird im Anschluss noch ein sogenanntes Stuffbyte eingefügt um genügend Pegelwechsel bei der Übertragung zu haben.

5.2.8.12 RFM12_RX_Byte(void)

Mit dem Befehl '0xB000'[24] wird aus dem FIFO-Puffer des Funkmoduls ein Byte ausgelesen. Da die SPI-Schnittstelle, über die das Funkmodul an den Mikrocontroller angeschlossen ist, eine bidirektionale, vollduplexfähige Verbindung ist, wird durch das Senden des Befehls gleichzeitig ein Byte vom Modul übertragen. Dieses Byte wird in der Variablen *value* gespeichert. Die folgende if-Anweisung überprüft *value*, ob es den Wert '0x00' oder '0xFF' besitzt. Ist dies der Fall, wird das bei der Übertragung angehängte Stuffbyte[25] noch aus dem FIFO-Puffer ausgelesen, aber nicht gespeichert. Die Funktion bekommt als Rückgabewert das eigentliche Datenbyte zugewiesen.

5.2.8.13 RFM12_send_Byte(unsigned char byte)

Im Unterschied zur Funktion *RFM12_TX_Byte* (siehe Abschnitt 5.2.8.11) ist *RFM12_send_Byte* für die komplette Übertragung eines Bytes verantwortlich. In der ersten und in der letzten Zeile der Funktion wird die Status-LED, welche das Senden von Daten signalisiert, ein- bzw. ausgeschaltet. Um die Übertragung zu starten, muss das Funkmodul zunächst in den 'Transmitter-Mode' gesetzt werden. Dies geschieht über das 'Power Management Command' '0x82xx' in dem die Bits 'et' und 'es' gesetzt werden (siehe Datenblatt des *RFM12*).

Nach Sendung des Nutzdatenbytes werden durch den Aufruf von *RFM12_TX_Byte* mit dem Parameter Null einige Fülldaten gesendet, um die Übertragung robuster zu machen. Mit dem Befehl '0x8208' wird zum Ende der Übertragung der 'Transmitter-Modus' des Funkmoduls wieder verlassen.

[24] Receiver FIFO Read Command, siehe Datenblatt des *RFM12*
[25] vgl. Funktion *RFM12_TX_Byte* in Kapitel 5.2.8.11

5.2.8.14 ISR(SIG_INTERRUPT1)

Die Interrupt-Service-Routine *INTERRUPT1* (siehe Quellcode 5.3) wird durch einen Interrupt an Pin 5 des Mikrocontrollers ausgelöst, über den der FFIT des Funkmoduls *RFM12* angeschlossen ist. Das Funkmodul signalisiert so dem Mikrocontroller, dass im FIFO-Puffer Daten vorliegen. Da nach jedem Nutzbyte noch ein Füllbyte übertragen wird[26], ist also nur jedes zweite Byte ein Nutzbyte. Zur Protokollierung dieses Sachverhaltes dient die Variable *Received_Data*, welche alternierend den Wert Null oder Eins zugewiesen bekommt (Zeile 292 und 311). Ist *Received_Data* gleich Eins, so wurde gerade ein Nutzdatenbyte empfangen und als nächstes sollte ein Füllbyte ankommen. Ist *Received_Data* hingegen Null, zeugt dies vom Empfang eines Füllbytes und als nächstes sollte ein Nutzdatenbyte folgen. Falls durch Übertragungsfehler dieser Rhytmus einmal gestört werden sollte, sorgt die Syncron-Pattern-Erkennung wieder für eine Synchronisation.

Ist *Received_Data* gleich Null, leitet die if-Anweisung die Bearbeitung des Empfangs eines Nutzdatenbytes ein. Zunächst wird durch die RX-LED der Empfang von Daten signalisiert und dementsprechend die Variable *Received_Data* auf Eins gesetzt. Das vom Funkmodul empfangene Byte wird durch die Funktion *RFM12_RX_Byte* aus dem FIFO-Puffer ausgelesen und in der Variablen *new_Char* zwischengespeichert. Danach wird das neue Zeichen an das Ende des globalen char-Array *Correction_String* angehängt und die dafür zuständige Zählvariable *Correction_String_Counter* um Eins inkrementiert (Zeile 295 und 296). Ist das empfangene Zeichen gleich dem ASCII-Code 10 (line feed), so ist das Ende des zu übertragenden Strings erreicht. In diesem Fall wird das Stringabschlusszeichen '\0' an den *Correction_String* gehängt. Dieser String, bzw. das char-Array, wird an die Funktion *decode_Correction_String* übergeben und die Zählvariable *Correction_String_Counter* wird wieder auf Null zurückgesetzt, um einen neuen String zu empfangen.

Ist so ein Nutzdatenbyte empfangen worden, sollte das nächste empfangene Byte ein Füllbyte sein, und die if-Anweisung leitet dessen Verarbeitung ein. Mit dem Befehl '0xCA81' wird zunächst das Füllen des FIFO-Puffers gestoppt. Unmittelbar danach wird mit dem Befehl '0xCA83' der FIFO-Puffer wieder aktiviert und somit auch die Synchron-Pattern-Detection neu gestartet (Zeile 306 und 307). Zum Abschluss des Empfangs wird die RX-LED wieder ausgeschaltet, der globalen Variablen *received_Char*, welche den Empfang eines Byte bzw.

[26] vergleiche hierzu Funktion *RFM12_send_Byte* in Kapitel 5.2.8.13

ASCII-Zeichens signalisiert, der Wert Eins und der alternierenden Variablen *Received_Data* der Wert Null zugewiesen.

```
287
288 ISR(SIG_INTERRUPT1){
289 if (!Received_Data)
290   {
291     LED_PORT |= (1<<LED_RX);
292     Received_Data=1;
293     new_Char = RFM12_RX_Byte();
294
295     Correction_String[Correction_String_Counter]=
          new_Char;
296     Correction_String_Counter++;
297
298     if (new_Char == 10){
299       Correction_String[Correction_String_Counter]='\0';
300       decode_Correction_String(Correction_String);
301       Correction_String_Counter=0;
302     }
303   }
304   else
305   {
306     RFM12_transmit(0xCA81);
307     RFM12_transmit(0xCA83);
308     LED_PORT &= ~(1<<LED_RX);
309     received_Char=1;
310
311     Received_Data=0;
312   }
```

Quellcode 5.3: Interrupt Routine ISR(SIG_INTERRUPT1)

5.2.8.15 RFM12_send_Char(unsigned char c)

Um ein char-Zeichen über das Funkmodul *RFM12* zu versenden, muss zunächst der standardmäßig wartende Empfangsmodus ausgeschaltet werden. Danach werden mit der Funktion *RFM12_send_Byte* alle weiteren Schritte zur Übertragung des char-Zeichens *c* eingeleitet. Nach der Übertragung wird das Funkmodul wieder in den Empfangsmodus zurückversetzt.

5.2.8.16 RFM12_received_Char(void)

Mit dieser Funktion wird lediglich die globale Variable *received_Char* auf ihren Wert überprüft, und dieser dann an den Funktionsaufruf zurückgegeben. Nach der vollständigen Übertragung eines char-Zeichens, inklusive Empfang des Füllbytes, wird in der Interrupt-Service-Routine *ISR(SIG_INTERRUPT1)* der Wert von *received_Char* auf Eins gesetzt. In der im folgenden Abschnitt erläuterten Funktion *RFM12_get_Char* wird *received_Char* wieder auf Null, zurückgesetzt.

5.2.8.17 RFM12_get_Char(void)

Zunächst verweilt *RFM12_get_Char* in einer while-Schleife bis die Funktion *RFM12_received_Char* signalisiert, dass die Übertragung eines char-Zeichens abgeschlossen ist. Um den Empfang eines weiteren Zeichens, welches dann wieder in *new_Char* abgelegt werden kann, zu ermöglichen wird die globale Variable *received_Char* wieder auf Null zurückgesetzt. Direkt im Anschluss wird das gerade empfangene Zeichen *new_Char* an den Funktionsaufruf zurückgegeben.

5.2.8.18 RFM12_is_busy(void)

Die Funktion gibt lediglich den Wert der globalen Variablen *busy* zurück. Ist das Funkmodul gerade mit Senden beschäftigt, wird der Wert Eins zurückgegeben. Befindet sich das Modul im wartenden empfangsbereiten Zustand, wird der Wert Null zurückgegeben.

5.2.8.19 RFM12_send_String(char *String)

Neben dem Senden einzelner char-Zeichen können durch die Funktion *RFM12_send_String* auch ganze Zeichenketten versendet werden. In einer while-Schleife wird der String vom ersten bis zum Stringabschlusszeichen durchlaufen. Hierzu wird der Zeiger auf das char-Array nach jedem Durchlauf mit '*String++' um ein Zeichen weitergesetzt. Gesendet wird das aktuelle char-Zeichen mit der Funktion *RFM12_send_Char* (siehe 5.2.8.15).

5.2.8.20 RFM12_get_String(char String_Buffer[40])

Analog zum Senden einer ganzen Zeichenkette (*RFM12_send_String*) dient die Funktion *RFM12_get_String* dem Empfangen einer solchen. Der Funkti-

on wird ein char-Array mit 40 Feldern übergeben. Mit der lokalen Variablen *counter* wird dieses char-Array *String_Buffer* von vorne beginnend durchlaufen. Zunächst wird nur das erste Zeichen des Strings mittels *RFM12_get_Char* (vgl. 5.2.8.17) empfangen und in *Next_char* zwischengespeichert bevor es an die erste Stelle im char-Array geschrieben wird.

In der while-Schleife wird zuerst die Zählvariable *counter* um Eins inkrementiert, dann das nächste Zeichen vom Funkmodul *RFM12* geholt und zum Schluss dieses Zeichen an die entsprechende Stelle im char-Array abgelegt. Nachdem die while-Schleife durch den Empfang des ASCII-Zeichens für 'Line Feed' (10) beendet wurde, wird an die nächste Stelle im *String_Buffer* das Stringabschlusszeichen '\0' eingetragen, und so der empfangene String abgeschlossen.

5.2.9 uart.c

Der Mikrocontroller *ATmega168* besitzt eine Hardware-UART. Um diese Schnittstelle nutzen zu können befinden sich in dieser Datei eine Funktion zur Initialisierung und Konfiguration der Hardware-UART, sowie einige Funktionen zum Empfang und Versand von Zeichen und Zeichenketten.

5.2.9.1 uart_init(void)

Die Aktivierung und Konfiguration der Hardware-UART wird durch die Funktion *uart_init* vorgenommen. Durch das Setzen der Bits *RXEN0* und *TXEN0* im Register *UCSR0B* werden der Transmitter und der Receiver der Hardware-UART aktiviert. Da das an die UART angeschlossene GPS-Modul das 8N1-Protokoll[27] verwendet, muss dies dementsprechend konfiguriert werden. Das Bit *UCSZ02* im Register *UCSR0B* und die Bits *UCSZ00* und *UCSZ01* im Register *UCSR0C* hängen zusammen und bestimmen die Datenbits eines Frames. Um eine Größe von acht Bit einzustellen, werden die Bits *UCSZ00* und *UCSZ01* gesetzt und das Bit *UCSZ02* gelöscht. Durch das Löschen der Bits *USBS0*, *UPM01* und *UPM00* im Register *UCSR0C* wird die Anzahl der Stopbits auf Eins gesetzt und der Parity-Mode deaktiviert. Zur Verwendung der asynchronen Kommunikation werden die beiden Bits *UMSEL00* und *UMSEL01* gelöscht. Vergleiche hierzu das Datenblatt auf der CD Seite 183.

Des Weiteren muss noch die Geschwindigkeit der Kommunikation eingestellt werden. Hierzu wird in der Header-Datei *uart.h* die Baudrate

[27] 8 Databits, No Parity, 1 Stop-Bit

UART_BAUD_RATE festgelegt. Ebenfalls in dieser Datei wird aus der gerade erwähnten Baudrate der entsprechende Wert des Registers *UBRR0* berechnet und in *UART_UBRR_VALUE* gespeichert. Dieser 16-Bit-Wert wird in die beiden Teilregister *UBRR0H* (für die acht höherwertigen Bits) und *UBRR0L* (für die acht niederwertigen Bits) eingetragen. Zum Abschluss der Initialisierung werden im zuständigen Data Direction Register (DDR) des Ports der Hardware-UART noch der RX-Pin als Eingang (Löschen des entsprechenden Bits) und der TX-Pin als Ausgang (Setzen des entsprechenden Bits) eingerichtet.

5.2.9.2 uart_send_Char(unsigned char Char)

Mit dieser Funktion kann ein char-Zeichen über die Hardware-UART des Mikrocontrollers versendet werden. Die Funktion wartet in der while-Schleife so lange, bis das Bit *UDRE0* im UART Controll and Status Register *UCSR0A* gesetzt ist und somit signalisiert wird, dass der Sendepuffer leer ist. Danach wird das zu sendende Zeichen in das Datenaustauschregister *UDR0* geschrieben.

5.2.9.3 uart_send_String(char *String)

Die Funktion *uart_send_String* dient dem Versenden einer ganzen Zeichenkette. Der durch den Zeiger auf ein char-Array übergebene String wird durchlaufen und das jeweils aktuelle Zeichen mit Hilfe der Funktion *uart_send_Char* versendet.

5.2.9.4 uart_get_Char(void)

Zunächst verweilt die Funktion *uart_get_Char* in einer while-Schleife bis durch das Setzen des Bits *RXC0* im Register *UCSR0A* signalisiert wird, dass Daten im Datenpuffer vorliegen. Das empfangene Zeichen wird ausgelesen und an den Funktionsaufruf zurückgegeben. Automatisch löscht der Mikrocontroller das Bit und ist sofort wieder empfangsbereit.

5.2.9.5 uart_get_String(char* String_Buffer)

Zum Empfang einer ganzen Zeichenkette dient die Funktion *uart_get_String*. Als erstes wird mit der im vorherigen Abschnitt erläuterten Funktion *uart_get_Char* ein Zeichen aus dem Datenpuffer geholt. Die while-Schleife wird nur dann betreten bzw. so lange wiederholt, bis das empfangene Zeichen ein 'Li-

ne Feed' (ASCII 10) ist. Innerhalb der Schleife wird das erhaltene char-Zeichen an die Stelle gespeichert, auf die der beim Funktionsaufruf übergebene Pointer zeigt; danach wird der Pointer um Eins inkrementiert. Anschließend wird das nächste Zeichen aus dem Datenpuffer geholt.

Nach Verlassen der while-Schleife wird das letzte empfangene Zeichen abgespeichert, der Zeiger nochmals um Eins erhöht, und so an die nächste und somit letzte Stelle des Strings gesetzt, um dann das Stringabschlusszeichen zu schreiben.

5.2.10 sw_uart.c

Da für dieses Projekt zwei UART-Schnittstellen benötigt werden, und der verwendete Mikrocontroller *ATmega168* nur eine Hardware-UART besitzt, muss eine zweite Schnittstelle in Softwareform implementiert werden. Neben zwei freien Pins zur Kommunikation (RX und TX) wird noch ein interner Timer benötigt, der zur zeitlichen Abstimmung der Signalerstellung erforderlich ist. Der hier verwendete Timer ist der 16-Bit 'Timer/Counter1' des *ATmega168*.

Bei diesem Projekt wird die Software-UART lediglich zur Ausgabe der korrigierten NMEA-Datensätze verwendet, die Kommunikation ist also unidirektional. Auf Grund dessen werden im Programm nur die zum Senden von Daten relevanten Teile berücksichtigt. Die Variable *sw_uart_Outframe* wird als Rahmen (Frame) zum Versenden von Nutzdaten in mehreren Funktionen verwendet und muss deswegen als globale Variable deklariert werden.

5.2.10.1 sw_uart_init(void)

Wie im vorherigen Abschnitt begründet, wird lediglich ein zum Senden verwendeter Pin benötigt. Welcher Pin für die Kommunikation vorgesehen ist, wird in der Datei *sw_uart.h* mit Hilfe der Konstanten *SW_UART_DDR*, *SW_UART_PORT* und *SW_UART_TX* festgelegt. Dementsprechend muss bei der Initialisierung der Schnittstelle dieser Pin durch das Setzten des betreffenden Bits im DDR als Ausgang eingerichtet werden.

Die Konfiguration des benötigten 'Timer/Counter1' erfolgt über die drei Register *OCR1A*, *TCCR1A* und *TCCR1B*. Die Register *TCCR1A* und *TCCR1B* dienen zur Einstellung des Verhaltens bzw. der Funktion des 'Timer/Counter1'. Mit den Bits WGM10 bis WGM13 wird die Funktion als Timer CTC[28] akti-

[28] Clear Timer on Compare match

viert. Bei dieser Funktion wird das Zählregister wieder auf Null gesetzt und ein Interrupt ausgelöst, wenn das Zählregister den gleichen Wert erreicht, der in OCR1A angegeben ist. Um vorerst das Auslösen eines Interrupts zu deaktivieren, wird mit der Standard-Funktion *cli()* das Global Interrupt Enable Bit im Status-Register gelöscht. Als Takt-Quelle für das Inkrementieren des Zählregisters wird durch das Setzten des Bits *CS10* (die Bits CS11 und CS12 sind standardmäßig gelöscht) der Systemtakt des Mikrocontrollers verwendet.

Der Wert, der in das Register *OCR1A* eingetragen, wird berechnet sich aus den in der Datei *sw_uart.h* deklarierten Konstanten *F_CPU* und *SW_UART_BAUD_RATE*. Um einen ganz zahligen 16-Bit Wert zu erhalten, der in das Register geschrieben werden soll, wird das Ergebnis der Division „gecastet“. Das *sw_uart_Outframe*, mit dessen Hilfe Daten gesendet werden sollen, wird auf Null initialisiert, was bedeutet, dass keine Daten zum Versenden vorliegen.

5.2.10.2 sw_uart_send_Char(char Char)

Mit dem einleitenden Schlüsselwort **inline** wird das „Wegoptimieren“ des verwendeten Assembler-Codes verhindert. In der while-Schleife (Zeile 35 bis 39, vgl. Quellcode 5.4), in der der Assembler-Code vorkommt, werden zunächst die Interrupts aktiviert, dann wartet der Mikrocontroller indem er mit 'nop' einen Arbeitstakt lang untätig bleibt. Nach der Wartezeit werden alle Interrupts deaktiviert um zu kontrollieren, ob dass *sw_uart_Outframe* gleich Null ist. Ist dies nicht der Fall wird weiter gewartet. Wenn *sw_uart_Outframe* gleich Null ist, ist das letzte char-Zeichen komplett gesendet worden und es kann ein neues char-Zeichen *Char* in den zu versendenden Rahmen, auch hier wird das Übertragungsprotokoll 8N1 verwendet (Zeile 41), geschoben werden.

Dieser Rahmen (Frame) besteht aus elf Bits. Das niederwertigste Bit ist das Startbit mit dem Wert Null. Danach kommen aufsteigend die acht Datenbits des zu versendenden Zeichens. Den Abschluss des Frames bilden ein Stopbit und eine Endmarke, welche beide den Wert Eins besitzen. Um die Datenübertragung zu starten wird zum Abschluss der Funktion der Output Compare A Match Interrupt (OCI1) gesetzt und aktiviert, sowie das Global Interrupt Enable Bit im Status-Register gesetzt.

```
34 void inline sw_uart_send_Char(char Char){
35   do{
36     sei();
37     __asm volatile ("nop");
38     cli();
39   }while (sw_uart_Outframe);
40
41   sw_uart_Outframe = (3 << 9) | (((uint8_t) Char) << 1);
         // frame = *.P.7.6.5.4.3.2.1.0.S   S=Start(0), P
       =Stop(1), *=End-Marker(1)
42
43   TIMSK1 |= (1 << OCIE1A);
44   TIFR1  |= (1 << OCF1A);
45
46   sei();
47 }
```

Quellcode 5.4: Erstellen *sw_uart_Outframe* und Einleitung der Übertragung

5.2.10.3 SIGNAL(SIG_OUTPUT_COMPARE1A)

Diese Interrupt Service Routine (ISR) wird dann aufgerufen, wenn der entsprechende Interrupt aktiviert ist und das Zählregister *TCNT1* (bestehend aus *TCNT1H* und *TCNT1L*) den gleichen Wert erreicht hat wie im Vergleichsregister OCR1A vorgegeben. Um eine Unterbrechung der ISR zu vermeiden, werden zu Beginn alle Interrupts deaktiviert.

Der zu versendende Frame *sw_uart_Outframe* wird in der lokalen Variablen *data* zwischengespeichert. Durch die if-Abfrage wird festgestellt, ob das niederwertigste Bit eine Null oder eine Eins ist, und dementsprechend wird der Ausgabe-Pin der Software-UART-Schnittstelle auf 'low' oder 'high' gesetzt. In der folgenden if-Anweisung wird der Wert von *data* überprüft. Ist dieser gleich Eins, ist nur noch die Endmarke vorhanden und somit der Frame komplett übertragen worden. Der OCI1 kann deaktiviert werden.

Nachdem auf diese Weise ein Bit des *sw_uart_Outframe* übertragen wurde, werden mit der Anweisung '*sw_uart_Outframe = data >> 1*' alle Bits um eine Stelle nach rechts verschoben. Das niederwertigste Bit fällt weg und auf der linken Seite wird eine Null aufgefüllt. Dadurch wird der *sw_uart_Outframe*

systematisch von rechts nach links abgearbeitet. Zum Schluss der ISR werden die Interrupts wieder aktiviert.

5.2.10.4 sw_uart_send_String(char *String)

Analog zu den Funktionen *uart_send_String* und *RFM12_send_String* ist die Funktion *sw_uart_send_String* für das Senden einer ganzen Zeichenkette über die Software-UART-Schnittstelle zuständig. In dieser Funktion wird *sw_uart_send_Char* für das Versenden der einzelnen char-Zeichen verwendet.

5.2.11 SPI.c

Die Datei *SPI.c* umfasst nur vier Funktionen, welche die grundlegenden Vorraussetzungen bereitstellt, die für eine Kommunikation über die SPI-Schnittstelle notwendig sind.

5.2.11.1 SPI_on(void)

Diese Funktion dient der Selektion bzw. der Aktivierung des SPI-Slaves. Der entsprechende Pin $\overline{SS}$ der SPI-Schnittstelle (hier Pin 2 an Port B) wird auf 'low'-Pegel gesetzt.

5.2.11.2 SPI_off(void)

Analog zur Funktion *SPI_on* wird durch Setzen des $\overline{SS}$-Pins auf 'high' der SPI-Slave deaktiviert.

5.2.11.3 SPI_init(void)

Zur Initialisierung der SPI-Schnittstelle werden im zuständigen Data Direction Register (DDR) der Pin MISO als Eingang die Pins MOSI, SCK und SS als Ausgang konfiguriert. Durch Setzen der Bits *SPR0* und *SPR1* im SPI Control Register (SPCR) wird die Taktung der SPI-Schnittstelle auf $\frac{\text{Taktfrequenz des Mikrocontrollers}}{128}$ gesetzt. Ebenfalls werden im SPCR die Bits *SPIE*, *SPE* und *MSTR* gesetzt, um den SPI-Interrupt und die SPI-Schnittstelle allgemein zu aktivieren sowie den Mikrocontroller als Master fest zu legen[29]. Zum

[29] vergleiche hierzu das auf der CD beiliegende Datenblatt des *ATmega168*

Abschluss der Initialisierung wird mit der Funktion *SPI_on()* der am Pin $\overline{SS}$ angeschlossene Slave, in diesem Fall das Funkmodul *RFM12*, selektiert.

5.2.11.4 SPI_send_receive_Byte(unsigned char send_Byte)

Da die SPI-Schnittstelle eine vollduplexfähige Kommunikation bereitstellt, wird beim Senden eines Bytes an den Slave zeitgleich ein Byte vom Slave an den Master übertragen. Das vom Master an den Slave zu sendende Byte wird als Eingabeparameter *send_Byte* an die Funktion übergeben. Durch das Schreiben des Bytes in das SPI Data Register (SPDR) wird die Übertragung gestartet. Die while-Schleife wird so lange durchlaufen, bis das SPI Interrupt Flag (SPIF) im SPI Status Register (SPSR) gesetzt wird, was signalisiert, dass der serielle Datenaustausch abgeschlossen ist. Anschließend kann das vom Slave gesendete Byte aus dem SPDR geholt und in der lokalen Variablen *received_Byte* abgespeichert werden, bevor es an den Funktionsaufruf zurückgegeben wird.

5.2.12 main.c

In der Datei *main.c* (Quellcode 5.5) befindet sich lediglich die Funktion *main(void)*, die das Hauptprogramm der Software beinhaltet. Hier wird die Initialisierung der Schnittstellen (Zeile 8 bis 13) und die Konfiguration des Funkmoduls (Zeile 17 und 18) durchgeführt. Die in der Datei *main.h* festgelegte Funktion des Mikrocontrollers als Sender (*TRANSMITTER*) oder Empfänger (*RECEIVER*) steuert die if-Abfragen in Zeile 30 und 42, und somit den relevanten abzuarbeitenden Quellcode.

5.2.12.1 main(void)

Um eine ungestörte Initialisierung aller Schnittstellen vornehmen zu können, werden zunächst mit *cli()* alle Interrupts deaktiviert. Als erstes wird der Port, an dem die Status-LED's angeschlossen sind (Port C des Mikrocontrollers), als Ausgabeport konfiguriert und als Anzeige, dass der Mikrocontroller seinen Betrieb aufgenommen hat, werden alle drei LED's eingeschaltet (Zeile 8 und 9). Danach werden die Schnittstellen SPI, UART und die Software-UART durch Aufruf der zugehörigen Funktionen initialisiert. Um zu überprüfen ob die Kommunikation über die Software-UART mit einem angeschlossenen Empfänger korrekt funktioniert (z.B. richtig eingestellte Baudrate), wird ein 'I' übertragen.

Die Inbetriebnahme und Konfiguration des Funkmoduls *RFM12* wird über die Funktionen *RFM12_init* und *RFM12_config* sowie der in der Datei *RFM12.h* festgelegten Parameter vorgenommen. Zum Abschluss der Initialisierung wird mit Hilfe der while-Schleife 200ms gewartet bis die Status-LEDs wieder ausgeschaltet werden (Zeile 25).

Der Hauptteil der Funktion *main* liegt in der while-Schleife, welche unaufhörlich durchlaufen wird. Durch die if-Abfragen innerhalb der Schleife wird die Funktion des Mikrocontrollers grundlegend bestimmt. Ist die Konstante *MODE* in der Datei *main.h* als *TRANSMITTER* festgelegt worden, soll der Mikrocontroller als stationärer Server (Referenzstation) fungieren. Somit ist über die UART-Schnittstelle der GPS-Empfänger angeschlossen, welcher dem Server die aktuellen GPS-Positionsdaten per NMEA-Datensätze übermittelt. Diese werden durch die Funktion *uart_get_String* eingelesen. Direkt im Anschluss wird durch *get_NMEA_Type* der Typ des NMEA-Datensatzes ermittelt. Ist dieser eine GPGGA-Mitteilung, die für die Berechnung der Korrekturdaten Verwendung findet, wird der Datensatz mit der Funktion *save_Correction_Data* zerlegt und die notwendigen GPS-Daten werden gespeichert. Mit Hilfe der Funktion *generate_Correction_String* wird aus diesen Daten der Korrekturstring erstellt, welcher über das Funkmodul *RFM12* ausgesendet werden soll. Bevor dies allerdings mit *RFM12_send_String* geschieht, wird überprüft, ob das Funkmodul sendebereit ist (Zeile 36).

Als mobiler Empfänger konfiguriert (*MODE = RECEIVER*), muss der Mikrocontroller, analog zur Server-Funktion, zunächst über die UART-Schnittstelle die aktuellen GPS-Positionsdaten empfangen und den Typ des NMEA-Datensatzes bestimmen. Auch hier sind nur die NMEA-Datensätze vom Typ GPGGA relevant. Der empfangene Datensatz wird zuerst durch die Funktion *correct_NMEA* korrigiert und danach über die Software-UART als String ausgegeben. Die Korrektur wird mit den aktuell gespeicherten Korrekturdaten durchgeführt, die asynchron und interruptgesteuert über das Funkmodul empfangen werden. (Vergleiche hierzu die Funktionen *correct_NMEA* in Kapitel 5.2.7.9, *ISR(SIG_INTERRUPT1)* in Kapitel 5.2.8.14 und *decode_Correction_String* in Kaptil 5.2.7.7.)

```
#include "main.h"

int main(void){
  cli();
  char* NMEA_String[100];
  char* correction_String[28];

  DDRC=255;
  PORTC |= (1<<3);

  SPI_init();
  uart_init();
  sw_uart_init();

  sw_uart_send_Char('I');

  RFM12_init();
  RFM12_config(RFM12_BAUDRATE, RFM12_CHANNEL, 0,
      RFM12_ENVIRONMENT);

  sei();

  unsigned char i;
  for (i=0; i<20; i++)
    _delay_ms(10);
  PORTC &= ~(1<<3);

  uint8_t NMEA_Type;

  while(1){
    if (MODE == TRANSMITTER){
      uart_get_String(NMEA_String,70);
      NMEA_Type = get_NMEA_Type(NMEA_String);
      if (NMEA_Type == GPGGA_MESSAGE){
        save_Correction_Data(NMEA_String);
        generate_Correction_String(correction_String);
        if (!RFM12_is_busy()){
          RFM12_send_String(correction_String);
        }
      }
    }

    if (MODE == RECEIVER){
```

```
43          uart_get_String(NMEA_String,70);
44          NMEA_Type = get_NMEA_Type(NMEA_String);
45          if (NMEA_Type == GPGGA_MESSAGE){
46            correct_NMEA(NMEA_String);
47            sw_uart_send_String(NMEA_String);
48          }
49        }
50    }
51 }
```

Quellcode 5.5: Das Hauptprogramm

6 Versuchsverlauf und Auswertung

Nachdem ich mich aufgrund erster Testreihen auf das GPS-Modul *HI-204III* von *Haicom* (siehe Kapitel 3.3) für das Projekt festgelegt hatte, wurden zwei dieser Module bestellt. Die in diesem Kapitel folgende Auswertung bezieht sich ausschließlich auf Testreihen und Ergebnisse, die mit Hilfe dieser beiden Module erstellt wurden. Auf der beiliegenden CD findet man neben den original NMEA-Datensätzen der GPS-Module auch Tabellen, in denen die relevanten Daten der Messreihen zusammengefasst sind, sowie Auswertungstabellen[1] und einige Diagramme zur Veranschaulichung der Messergebnisse.

In den Tabellen-Dateien befinden sich in den Datenblättern 'Stat' und 'Mobil' die relevanten Daten, welche aus den NMEA-Datensätze herausgefiltert wurden, sowie die Differenzwerte der gemessenen Koordinaten zu den theoretischen Sollkoordinaten. Im Tabellenblatt 'Daten' sind die Werte der Referenzstation und der mobilen Einheit zusammengefasst. In den weiteren Spalten sind die Differenzen der vom Mikrocontroller korrigierten Koordinaten (die mit Hilfe von sieben verschiedenen Formeln berechnet wurden[2]) und die Sollkoordinaten der mobilen Einheit aufgelistet.

Der folgende Abschnitt beschreibt den Versuchsaufbau, welcher für die Messreihen verwendet wurde. Danach werden einzelne Messreihen untersucht und bewertet, bevor im letzten Abschnitt des Kapitels eine Gesamtbewertung vorgenommen wird.

6.1 Der Versuchsaufbau

Das zentrale Element des Versuchsaufbaus ist die stationäre Referenzeinheit. Um möglichst genaue Koordinaten dieses wichtigen Bezugspunktes zu gewährleisten, habe ich einen genau vermessenen Punkt in der Gemeinde Hahn (bei Bad Marienberg) zu Grunde gelegt. Dieser befindet sich bei 50° 37" 53,98" nördlicher Breite und 7° 56' 25,50" östlicher Länge. Da die GPS-Module das NMEA-

[1] Erläuterungen hierzu im Dokument 'Legende.pdf' auf der beiliegenden CD
[2] die einzelnen Formeln werden in den folgenden Abschnitten vorgestellt

	Referenzstation	Position 1	Position 2	Position 3
Entfernung zur Referenzstaion [m]	0m	10,34m	21,92m	16,18m
Lage von Referenzstation [Grad]	0°	≈ 154°	≈ 88°	≈ 54°
Breitenkoordinate in Grad	50°37′53.98”N	50°37′53.68”N	50°37′53.98”N	50°37′54.28”N
Längenkoordinate in Grad	7°56′25.50” E	7°56′25.74” E	7°56′26.62” E	7°56′26.18” E
Breitenkoordinate NMEA-konform	5037.8996 N	5037.8946 N	5037.8996 N	5037.9046 N
Längenkoordinate NMEA-konform	00756.4250 E	00756.4290 E	00756.4436 E	00756.4363 E

Tabelle 6.1: Übersicht Messstationen

Datenformat zur Übertragung verwenden und somit alle gespeicherten Datensätze dieses Format besitzen, wird im Folgenden nur noch die NMEA-konforme Schreibweise der Form 'GGMM.DDDD' für die Breite und 'GGGMM.DDDD' für die Länge verwendet[3]. Zur Umrechnug werden die Sekunden (mit Nachkommastelle) durch 60 geteilt um auf eine Dezimal-Minutenangabe zu kommen:

$$50°37'[\frac{53,98''}{60}] \;\widehat{=}\; 50°37,8996' \;\widehat{=}\; 5037.8996 \quad \textit{(NMEA konform)}$$
$$7°56'[\frac{25,50''}{60}] \;\widehat{=}\; 7°56,4250' \;\widehat{=}\; 00756.4250 \quad \textit{(NMEA konform)}$$

Neben dem festen Standpunkt des einen GPS-Moduls der Referenzstation, gibt es drei weitere Punkte, an denen das mobile GPS-Modul platziert wurde. Diese befinden sich in einigen Metern Entfernung vom stationären Referenzpunkt. In Tabelle 6.1 sind die drei Punkte mit ihren Koordinaten (in beiden Formaten), der Abstand zum Referenzpunkt und die Grad-Abweichung von Norden (ausgehend vom Referenzpunkt) aufgelistet. Bild 6.1 verdeutlicht noch einmal die Lage der vier Punkte durch ein Satellitenbild. Im Anhang A.1 ist Abbildung A.1 mit etwas detaillierteren geometrischen Informationen zu sehen. Die Abbildung A.2 soll die im Verlauf einer Messreihe auftretenden Abweichungen und die Lage der verschiedenen Messpunkte im Größenvergleich deutlich machen.

Neben dem GPS-Modul gehören noch zwei weitere Komponenten zu einer Messeinheit. Um die Messdaten der einzelnen Versuchsreihen aufzuzeichen wird jeweils ein Laptop verwendet. Zwischen diesen beiden Komponenten befindet sich die im vorherigen Kapitel beschriebene, selbst entwickelte Schaltung, welche Daten vom GPS-Modul empfängt, neue Daten berechnet und beides über eine RS232-Schnittstelle an den Laptop weitergibt.

[3] G $\widehat{=}$ eine Stelle der Grad-Angabe, M $\widehat{=}$ eine Stelle der Minuten-Angabe, D $\widehat{=}$ eine Stelle der Dezimal-Minuten-Angabe

Abbildung 6.1: Lageplan der vier Messpunkte des Versuchsaufbaus

6.2 Auswertung einiger Versuchsreihen

Um die Genauigkeit der mittels GPS bestimmten Position zu verbessern, sollen aktuelle Messwerte der Referenzposition und deren Abweichungen von der eigentlichen Position verwendet werden. Die durch lokale Wetterverhältnisse und Konstellation der GPS-Satelliten auftretenden Ungenauigkeiten sollen somit eleminiert werden, womit das Prinzip der Korrektur dem des *DGPS* (siehe Kapitel 2.3) ähnlich ist. Allerdings könnten durch die eigenen vor Ort bestimmten GPS-Werte die lokalen Einflussfaktoren besser berücksichtigt werden.

In diesem Kapitel werden einige statistische Maßzahlen angewandt, um verschiedene Aspekte der Mess- und Datenreihen zu beschreiben und zu bewerten. Die Maßzahl des arithmetischen Mittelwertes ist allerdings zur Bewertung einer kontinuierlichen Verbesserung der Positionsbestimmung weniger geeignet. Liegen viele Werte extrem über oder unter den Werten der eigentlichen Position, so kann dennoch ein sehr guter Mittelwert erzielt werden. Dieses Manko des arithmetischen Mittelwertes wird bei der Berechnung der mittleren absoluten Abweichungen berücksichtigt, wo nur der Abstand und nicht die Richtung der Abweichung einfließt. Die allgemein übliche und aussagekräftigste Maßzahl zur Bewertung der hier vorhandenen Messreihen ist jedoch die Standardabweichung. Die Formeln zur Berechnung dieser Maßzahlen ist im Anhang A.5 zu finden. In Abschnitt 6.3.1 dieses Kapitels ist in den Tabellen 6.2, 6.3 und 6.4 eine Übersicht über den arithmetischen Mittelwert, die absolute mittlere Abweichung und die Standardabweichung aller Messreihen vorhanden.

6.2.1 Versuchsreihe 1

Zunächst verwendete ich die einfachste Form der Korrektur. Hierzu berechnet die Referenzstation die Abweichung der Längen- und Breitenkoordinaten aus den bekannten Koordinaten ihres Standortes und den aktuell durch das GPS-Modul ermittelten Koordinaten. Diese Korrekturwerte werden über das Funkmodul *RFM12* an die mobile Einheit übermittelt, welche dann die ihrerseits gemessenen GPS-Koordinaten direkt um die entsprechenden Werte korrigiert. In den folgenden Ausführungen und Formeln wird die Breitenkoordinate als x-Koordinate bezeichnet, und die Längenkoordinate als y-Koordinate. Mit den Indizes 's' und 'm' wird ein Wert der stationären Referenzstation bzw. der mobilen Einheit zugeordnet. Die Abkürzung 'gem' steht für den aktuell gemessenen Wert und 'fest' für die festgelegten Koordinaten der Referenzstation. Die korrigierten Koordinaten der mobilen Einheit x_m und y_m berechnen sich über folgende Formeln:

FORMEL 1

$$x_m = x_m^{gem} - (x_s^{gem} - x_s^{fest}) \tag{6.1}$$

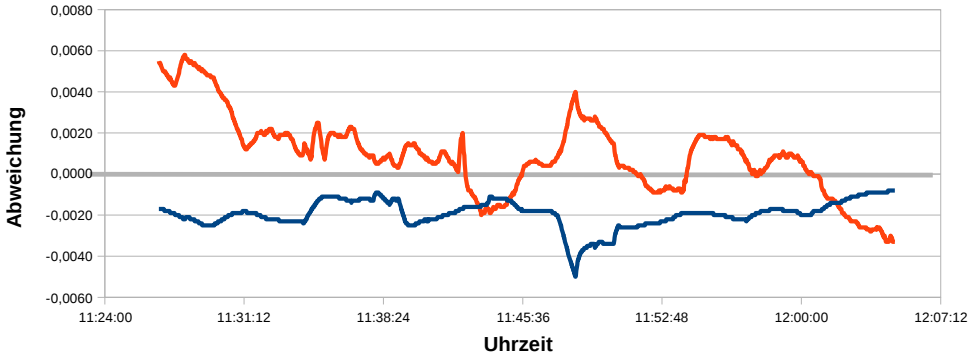
$$y_m = y_m^{gem} - (y_s^{gem} - y_s^{fest}) \tag{6.2}$$

Die erste Messreihe umfasste einen Zeitraum von 11:26:48 Uhr bis 12:04:50, also 38 Minuten und 2 Sekunden. Die in diesem Zeitraum erfassten NMEA-Datensätze der Referenzstation befinden sich in der Datei '01_Position_Ref.gps'. Die NMEA-Datensätze der mobilen Einheit, die sich bei dieser Messreihe am Standort 'Position 1' befand, sind in der Datei '01_Position_1.gps' gespeichert.

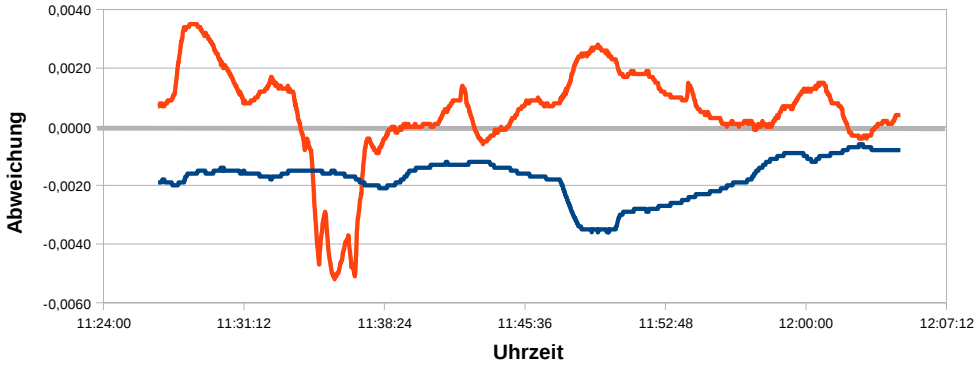
In der Datei '01_Position_1.xls' sind im Tabellenblatt 'Daten' in den Spalten 'F1 X' und 'F1 Y' die Differenz zwischen dem Sollwert und den mit Hilfe der beiden oben erwähnten Formeln berechnenten Koordinaten aufgelistet. In Abbildung 6.2(a) ist der Verlauf der Abweichung der Breitenkoordinate und der Verlauf der Abweichung der nicht korrigierten Breitenkoordinate dargestellt. Analog dazu die Verläufe der Längenkoordinaten in Abbildung 6.2(b). Man kann in beiden Abbildungen erkennen, dass der Graph der korrigierten Daten größtenteils näher an der x-Achse liegt. Dies bedeutet, dass die korrigierten Koordinaten näher an den Sollkoordinaten des Standortes liegen. Somit ist eine Verbesserung der Positionsbestimmung erreicht worden, erkennbar auch durch die mittlere absolute Abweichung[4] (vgl. Tabelle 6.3, Seite 149). Lag diese zu-

[4] Formel im Anhang A.5.3

nächst bei 0,0019' (Minuten) für die Breitenkoordinate und bei 0,0017' für die Längenkoordinate, liegt sie nach der Korrektur bei der Breitenkoordinate bei 0,0017' und bei der Längenkoordinate bei 0,0012' Bei Betrachtung der Standardabweichungen ist das Ergebnis allerdings different. Hier ist nur bei der y-Koordinate eine Verbesserung festzustellen.



(a) Abweichungen der x-Koordinate unbearbeitet und korrigiert

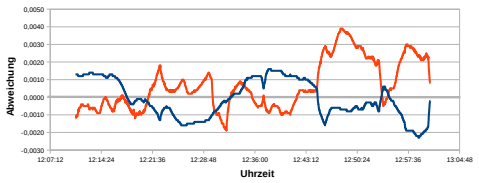


(b) Abweichungen der y-Koordinate unbearbeitet und korrigiert

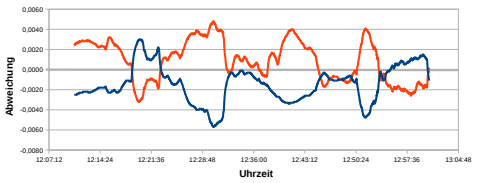
Abbildung 6.2: Graphische Darstellung von Messreihe 1 mit Korrektur durch Formel 1

Leider ist keine Glättung der Graphen der korrigierten Werte festzustellen. Teilweise sind die Abweichungsspitzen größer als die der unbearbeiteten Daten. Auch sind die korrigierten Koordinaten in einigen Abschnitten konstant schlechter. Somit ist keine stetige Verbesserung der Positionsbestimmung zu erkennen. Nach dem ersten unbefriedigenden Ergebnis wurden weitere Messreihen zur Untersuchung der Optimierung durchführt. Diese sind in den Abbildungen 6.3(a) bis 6.3(d) zu sehen.

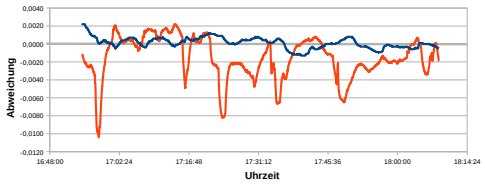
Auch bei diesen beiden Messreihen ist keine Glättung des Graphen zu erkennen. Die Graphen der korrigierten Daten weisen sogar wesentlich größere Schwankungen auf. Im Gegensatz zur ersten Messreihe ist die mittlere absolute Abwei-



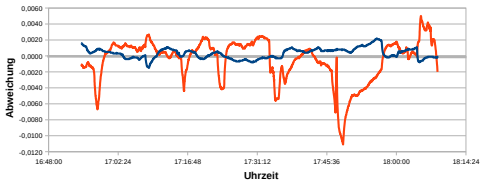
(a) Abweichungen der x-Koordinate bei Messreihe 2 unbearbeitet und korrigiert



(b) Abweichungen der y-Koordinate bei Messreihe 2 unbearbeitet und korrigiert



(c) Abweichungen der x-Koordinate bei Messreihe 4 unbearbeitet und korrigiert



(d) Abweichungen der y-Koordinate bei Messreihe 4 unbearbeitet und korrigiert

Abbildung 6.3: Graphische Darstellung von Messreihe 2 und 4 mit Korrektur durch Formel 1

chung der Messreihe 2 bei der x-Koordinate sogar schlechter geworden. Bei der Messreihe 4 ist eine weitere Verschlechterung festzustellen. Hier ist die mittlere absolute Abweichung beider Koordinaten erheblich schlechter geworden als die der ursprünglich gemessenen Koordinaten. Auch bei der Standardabweichung für die x-Koordinate ist eine Verschlechterung von 0,0022' zu verzeichnen. Dies lässt darauf schließen, dass eine Korrektur der Positionsdaten mit den Formeln 6.1 und 6.2 nicht zu einer zufriedenstellenden Steigerung der Genauigkeit führt.

6.2.2 Versuchsreihe 2

Der zweite Ansatz zur Verbesserung basiert auf den HDOP-Werten der NMEA-Datensätze. Mit diesen Werten geben die GPS-Module eine Kennzahl für die Genauigkeit der 2D-Positionsbestimmung in der Horizontalebene[5]. Hierbei werden die aktuellen HDOP-Werte der beiden GPS-Module miteinander verglichen. Ist der HDOP-Wert des stationären Referenzmoduls kleiner als der des mobilen Moduls, werden die Längen- und Breitenkoordinaten um die aktuellen Korrekturwerte, wie in der ersten Versuchsreihe beschrieben, berichtigt. Ist jedoch der HDOP-Wert des GPS-Moduls, welches an der mobilen Einheit angeschlossen ist, kleiner oder gleich dem des Referenzmoduls, werden die gerade gemessenen Koordinaten unverändert übernommen. Die Zuordnungsforschrift der bearbeiteten Koordinaten x_m und y_m wird mit folgenden Formeln beschrieben:

[5] vergleiche hierzu Kapitel 2.2.7

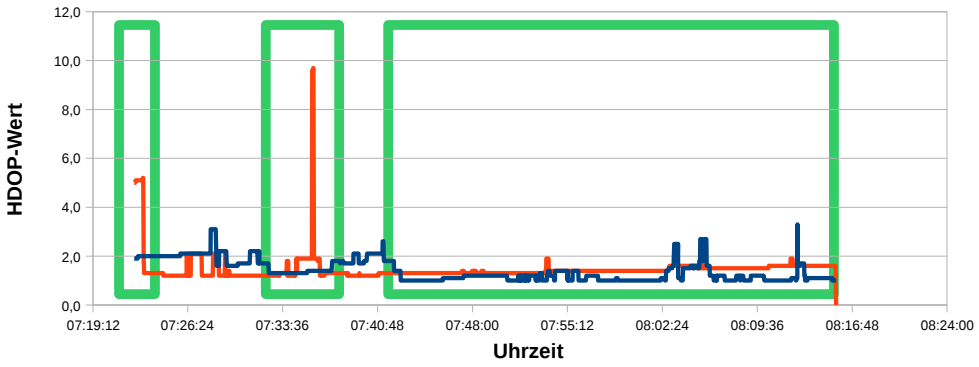
FORMEL 2

$$x_m = \begin{cases} x_m^{gem} - (x_s^{gem} - x_s^{fest}) & \text{, wenn } HDOP_s < HDOP_m \\ x_m^{gem} & \text{, sonst} \end{cases} \tag{6.3}$$

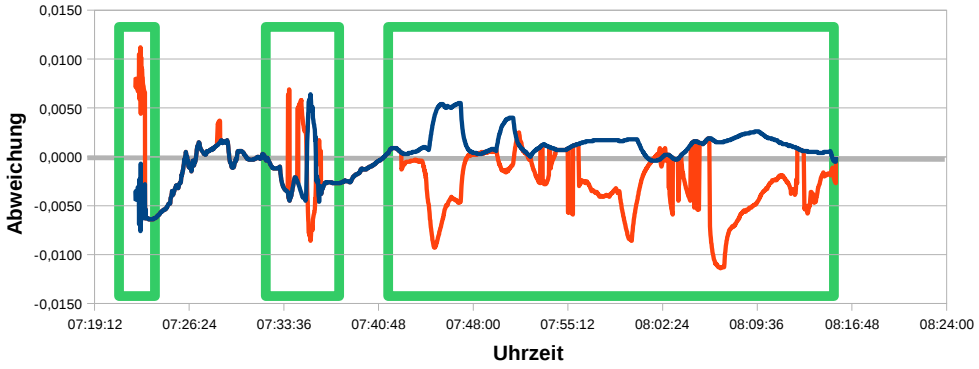
$$y_m = \begin{cases} y_m^{gem} - (y_s^{gem} - y_s^{fest}) & \text{, wenn } HDOP_s < HDOP_m \\ y_m^{gem} & \text{, sonst} \end{cases} \tag{6.4}$$

Abbildung 6.4 zeigt den Verlauf der HDOP-Werte beider Stationen (a), der unveränderten Koordinaten und der korrigierten Koordinaten für die Breiten- (b) und Längenangabe (c) der Messreihe 5. In den Abbildungen sind drei größere Bereiche zu erkennen (grün), in denen der HDOP-Wert der mobilen Einheit über der der Referenzeinheit liegt. In diesen Bereichen werden die gemessenen Koordinaten mit den entsprechenden Korrekturwerten der Referenzstation verrechnet. Bei beiden Graphen der Koordinatenwerte ist zu erkennen, dass, wie bei den ersten beiden Messreihen, der Verlauf eher unregelmäßiger wird.

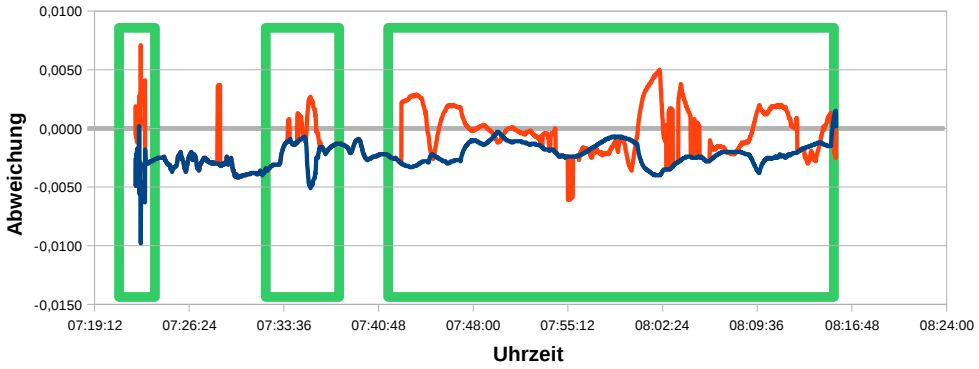
Teilweise kann man bei der Längenkoordinate innerhalb der drei markierten Bereiche eine leichte Verschiebung des Graphen in Richtung der x-Achse erkennen. Außerhalb dieser Bereiche sind die beiden Graphen für die ursprünglichen Werte und die bearbeiteten Werte mit Ausnahme eines einzigen Ausreißers gleich. Bei den Breitenkoordinaten in Abbildung 6.4(b) ist jedoch eine leichte Verschiebung des Graphen der korrigierten Werte weg von der x-Achse zu erkennen. Dies drückt sich auch in den mittleren absoluten Abweichungen der Messdaten aus. Bei der Längenkoordinate ist noch eine Verbesserung der Abweichung von 0,0023' auf 0,0020' festzustellen, bei der Breitenkoordinate hingegen ist eine Verschlechterung von nicht aktzeptablen 0,0011' zu verzeichnen. Dies unterstreichen auch die Standardabweichungen, welche bei der x-Koordinate sogar um 0,0014' schlechter ist.



(a) HDOP-Werte der Referenzstation und der mobilen Einheit



(b) Abweichungen der x-Koordinate unbearbeitet und korrigiert



(c) Abweichungen der y-Koordinate unbearbeitet und korrigiert

Abbildung 6.4: Graphische Darstellung von Messreihe 5 mit Korrektur durch Formel 2

6.2.3 Versuchsreihe 3

Die großen Abweichungen der Korrekturwerte der Referenzstation verursachen starke Unregelmäßigkeiten der bisherigen Graphen. Um diese zu vermeiden wird versucht einen Grenzwert für den HDOP-Wert zu setzen, ab dem eine Korrektur vorgenommen wird. Nach einigen Messreihen hat sich eine Grenze von 1,2

als geeignet herausgestellt. Ist der HDOP des NMEA-Datensatzes der mobilen Testeinheit größer oder gleich 1,2, so wird der Messwert der Koordinaten der mobilen Einheit um den Korrekturwert der Referenzeinheit berichtigt. Somit wird ein auf Grund des HDOP-Wertes für gut befundener Messwert der mobilen Einheit nicht durch extreme Korrekturwerte der Referenzstation negativ beeinflusst. Zur Berechnung der Breiten- und Längenkoordinate wird bei einem guten HDOP-Wert der mobilen Einheit, also ein HDOP-Wert unter 1,2, keine Korrektur vorgenommen. Ist der HDOP-Wert schlechter als 1,2, so wird die Formel aus der ersten Versuchsreihe zur Korrektur verwendet. Die dazugehörige Zuordnungsvorschrift für die beiden Koordinatenwerte sieht folgendermaßen aus:

FORMEL 3

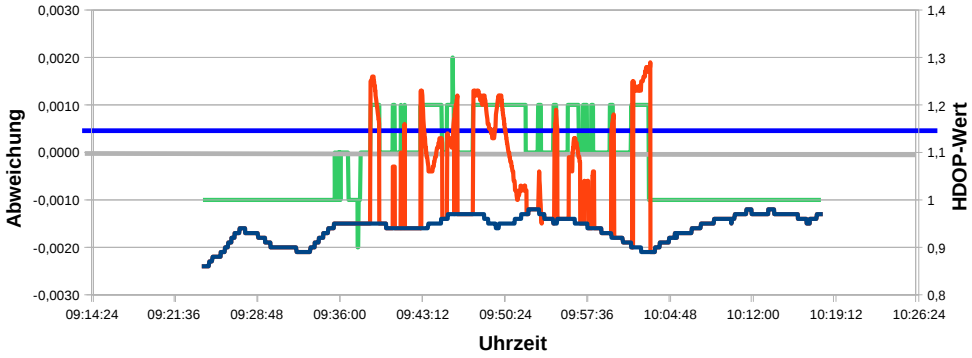
$$x_m = \begin{cases} x_m^{gem} & \text{, wenn } HDOP_m < 1,2 \\ x_m^{gem} - (x_s^{gem} - x_s^{fest}) & \text{, sonst} \end{cases} \tag{6.5}$$

$$y_m = \begin{cases} y_m^{gem} & \text{, wenn } HDOP_m < 1,2 \\ y_m^{gem} - (y_s^{gem} - y_s^{fest}) & \text{, sonst} \end{cases} \tag{6.6}$$

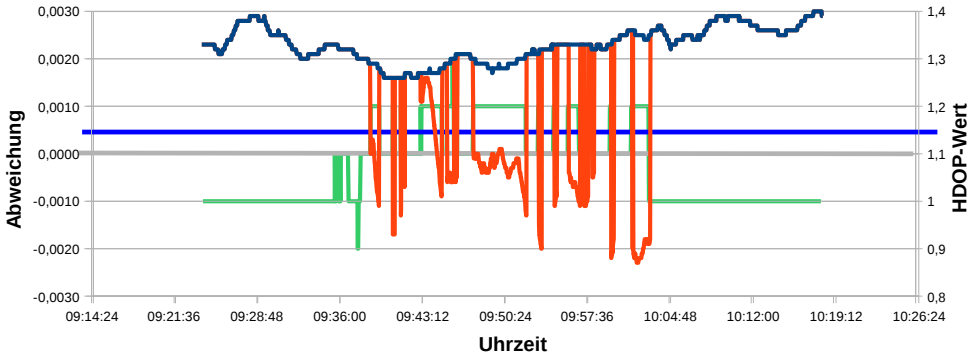
Wie in Abbildung 6.5(a) und 6.5(b) zu erkennen ist, weisen die beiden Graphen der korrigierten Koordinaten punktuell extreme Abweichungen auf. Bei dieser Messreihe ist der HDOP-Wert der mobilen Einheit meistens unter der Grenze von 1,2. Beim Vergleich der Koordinaten-Graphen und der des HDOP-Wertes kann man feststellen, dass sich die extremen Abweichungen genau an den Stellen befinden, an denen der HDOP-Wert 1,2 übersteigt. Diese Abweichungen machen den Graphen zwar unruhiger, verbessern aber die Koordinatengenauigkeit in diesen Bereichen. Das Problem ist, dass die Verbesserung so nur punktuell ist, und nicht zu einer konstanten Genauigkeitssteigerung der Messergebnisse führt. In der nächsten Versuchsreihe wird ein weiterer Ansatz auf Basis von HDOP-Werten vorgestellt, der allerdings sowohl die HDOP-Werte der mobilen Einheit, als auch die der Referenzstation mit einfließen lässt.

6.2.4 Versuchsreihe 4

Um dem Nachteil der nur punktuellen Verbesserung der vorherigen Versuchsreihe entgegen zu wirken, wird versucht den Korrekturwert der stationären Referenzstation permanent in die Berechnung mit einfließen zu lassen. Wie in Versuchsreihe 3 soll jedoch auch der HDOP-Wert der mobilen Einheit mit ein-



(a) Abweichungen der x-Koordinate unbearbeitet und korrigiert und HDOP-Wert



(b) Abweichungen der y-Koordinate unbearbeitet und korrigiert und HDOP-Wert

Abbildung 6.5: Graphische Darstellung von Messreihe 6 mit Korrektur durch Formel 3, sowie HDOP-Wert der Referenzstation

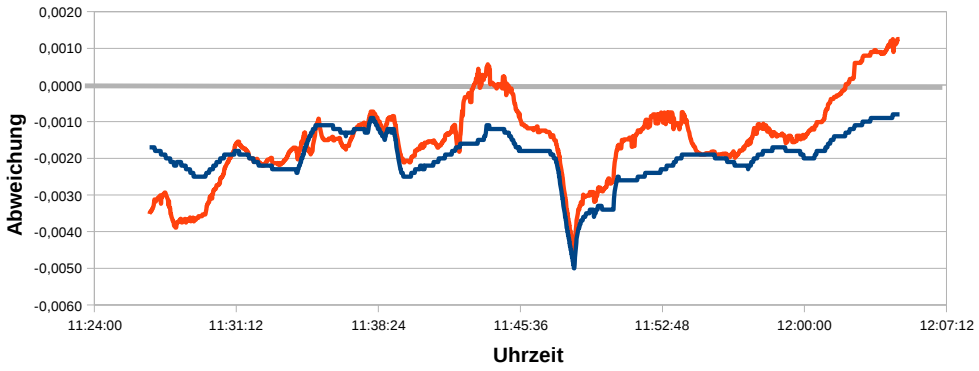
bezogen werden. Der Ansatz bei der folgenden Berechnung ist, dass der Einfluss des Korrekturwertes der Referenzstation, nämlich die Differenz der Messwerte und der Sollwerte, abhängig ist von dem Verhältnis der HDOP-Werte der Referenzstation zu denen der mobilen Einheit. Je kleiner der HDOP-Wert der mobilen Einheit ist, desto weniger soll der Korrekturwert der Referenzstation die Berechnung beeinflussen. Um diesen Zusammenhang auszudrücken, wird der Korrekturwert mit einem Koeffizienten, der das Verhältnis der beiden HDOP-Werte wiedergibt, multipliziert. Dieser Koeffizient besteht aus dem HDOP-Wert der mobilen Einheit geteilt durch die Summe der HDOP-Werte der beiden Einheiten. Somit wird bei einem besseren HDOP-Wert der mobilen Einheit der Koeffizient kleiner und in Folge dessen wird auch der Einfluss des Korrekturwertes kleiner. Mit der folgenden Zuordnungsvorschrift werden die korrigierten Längen- und Breitenkoordinaten berechnet:

FORMEL 4

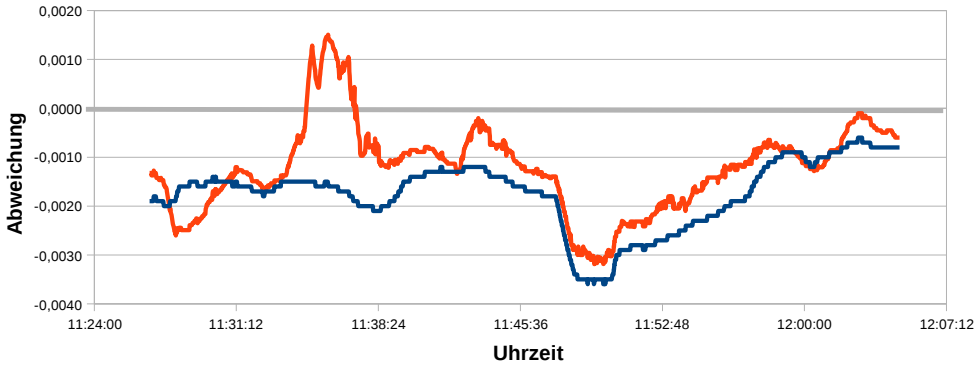
$$x_m = x_m^{gem} - \left[\left(\frac{HDOP_m}{HDOP_m + HDOP_s}\right) * (x_s^{gem} - x_s^{fest})\right] \tag{6.7}$$

$$y_m = y_m^{gem} - \left[\left(\frac{HDOP_m}{HDOP_m + HDOP_s}\right) * (y_s^{gem} - y_s^{fest})\right] \tag{6.8}$$

Vergleicht man die Graphen der ursprünglichen Messwerte der mobilen Einheit mit den neu berechneten Werten von Messreihe 8, so kann man zwar nicht unbedingt eine Glättung des Graphen erkennen, aber die korrigierten Werte liegen in großen Bereichen des Graphen näher an der x-Achse (Abbildung 6.6). Auch die unregelmäßigen extremen Abweichungen sind nicht mehr in dem Maße der vorherigen Versuchsreihen vorhanden. Punktuell sind jedoch sowohl bei der Breiten- als auch bei der Längenkoordinate die neu berechneten Koordinatenwerte schlechter als die gemessenen Werte.



(a) Abweichungen der x-Koordinate unbearbeitet und korrigiert

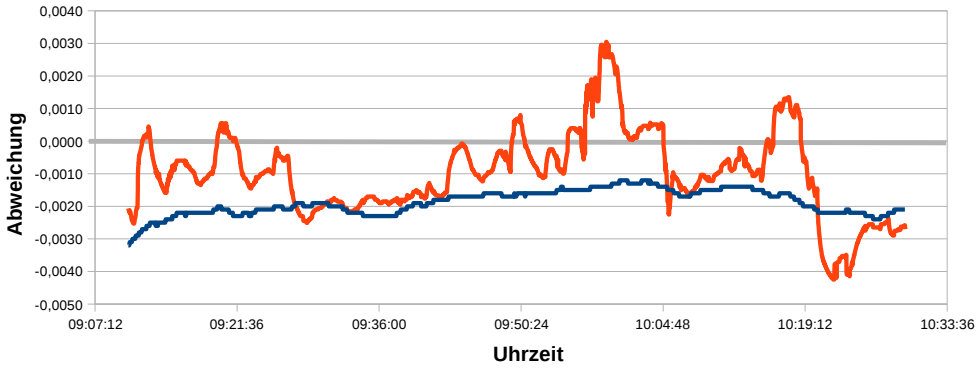


(b) Abweichungen der y-Koordinate unbearbeitet und korrigiert

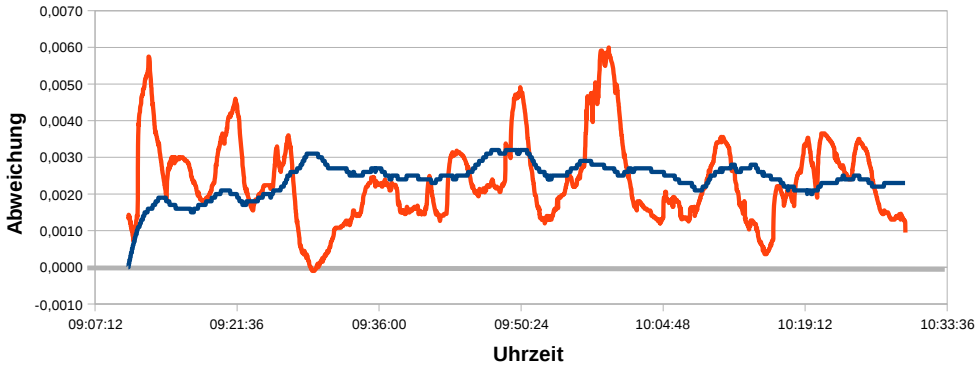
Abbildung 6.6: Graphische Darstellung von Messreihe 8 mit Korrektur durch Formel 4

Diese leichte, aber relativ konstante, Verbesserung der Koordinaten kann man auch an verschiedenen Maßzahlen der Datenreihen erkennen. Diese sind in der

Datei '08_Position_1.xls', 'Übersicht_Maßzahlen.xls' und in den Tabellen auf Seite 148 bis 150 zu finden. Die Werte für die absolute mittlere Abweichung sind sowohl für die Breiten- als auch die Längenkoordinate besser. Die Standardabweichung ist ebenfalls bei beiden Koordinaten besser geworden (um 0,0002' bei den x-Werten und 0,0004' bei den y-Werten). Insgesamt scheint der in dieser Versuchsreihe gewählte Ansatz vielversprechender zu sein als die bisherigen. Um das Verbesserungspotenzial des Ansatzes genauer zu untersuchen sind in Abbildung 6.7 die Graphen einer weiteren Messreihe zu sehen.



(a) Abweichungen der x-Koordinate unbearbeitet und korrigiert



(b) Abweichungen der y-Koordinate unbearbeitet und korrigiert

Abbildung 6.7: Graphische Darstellung von Messreihe 11 mit Korrektur durch Formel 4

Sowohl bei der Breiten- als auch bei der Längenkoordinate ist der Graph der neu berechnenten Werte in dieser Messreihe 11 offensichtlich wieder wesentlich unruhiger geworden. Bei den x-Werten liegt der Graph der korrigierten Werte zumeist oberhalb und näher an der x-Achse als der Graph der ürsprünglichen Messwerte. Dies spiegelt sich auch in den Maßzahlen der absoluten mittleren Abweichung und der Standardabweichung wider[6], die um 0,0006' bzw.

[6] siehe Datei '11_Position_2.xls', 'Übersicht_Maßzahlen.xls' und in der Tabelle 6.4

um 0,0003' besser werden. Bei den y-Werten sind die Ergebnisse weniger zufriedenstellend. Der Graph der korrigierten Messwerte pendelt quasi um den der originalen Messwerte. Der größere Teil des Graphen liegt zwar näher an der x-Achse, aber eine Reihe extremer Abweichungen liegt jedoch weiter von der x-Achse entfernt. Dies ist auch bei den beiden Maßzahlen der absoluten mittleren Abweichung und der Standardabweichung festzustellen. Ist die absolute mittlere Abweichung der korrigierten Daten noch ein wenig besser (0,0001'), so haben die extremen Abweichungen auf die Maßzahl Standardabweichung größere Auswirkungen[7] und beeinflussen diese negativ. Hier verschlechtert sich der Wert von 0,0024' auf 0,0026'.

Nach den anfänglich guten Ergebnissen führte jedoch die Analyse einiger weiterer Messreihen zu dem Schluss, dass der von den Modulen berechnete HDOP-Wert nicht aussagekräftig genug ist. Ein extremes Beispiel hierfür ist die Messreihe 12. Hier liegt zu Beginn der Messreihe, bei einem sehr guten HDOP-Wert von 1,1, die Abweichung der y-Koordinate der mobilen Einheit bei über 0,0200'. Im Durchschnitt beträgt der HDOP-Wert 1,06 mit einer Standardabweichung von 0,0296'. Weitere Ansätze auf Basis der HDOP-Werte scheinen also ebenfalls nicht sehr erfolgsversprechend.

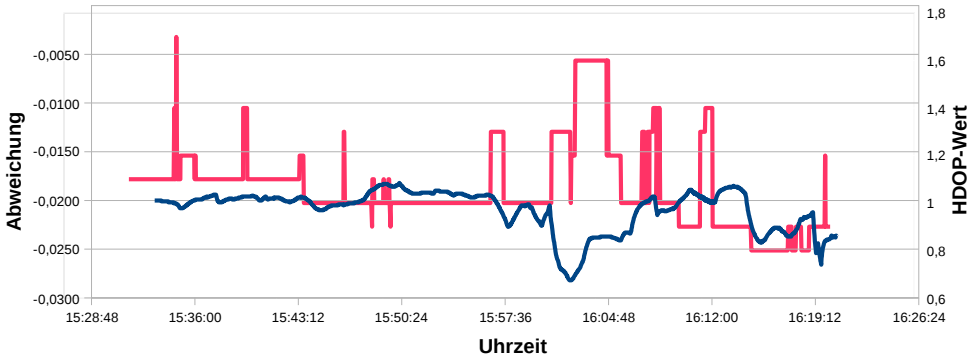


Abbildung 6.8: Graphische Darstellung von Messreihe 12 mit HDOP

6.2.5 Versuchsreihe 5

Um den teilweise starken Schwankungen entgegen zu wirken, die sich bei manchen Korrekturrechnungen sogar verstärkt haben, basieren die folgenden drei Ansätze auf der statistischen Maßzahl des Mittelwertes. Es wird versucht mit Hilfe des Mittelwertes der Abweichung der bisherigen Messdaten einen durchschnittlichen Korrekturwert zu errechnen. In dieser Versuchsreihe wird der Mit-

[7] vergleiche Formel A.5.5 im Anhang

telwert der Abweichung der Breiten- und Längenkoordinate der stationären Referenzeinheit bestimmt und mit diesem die entsprechenden aktuellen Messdaten der mobilen Station korrigiert. So sollen die temporären Schwankungen bei der Positionsmessung der Referenzstation geglättet werden, und ein an aktuelle Gegebenheiten (Wetter, Satellitenkonstellation, usw.) angepasster durchschnittlicher Korrekturwert bestimmt werden.

Zur Berechnung der Mittelwerte der Abweichungen für die Breiten- und Längenkoordinate der Referenzstation werden die Messdaten der ersten Messung, mit dem Index 1, bis zur letzten aktuellen Messung, mit dem Index t, verwendet. Die folgenden beiden Formeln dienen zur Berechnung der aktuellen korrigierten Breiten- und Längenkoordinaten der mobilen Einheit:

FORMEL 5

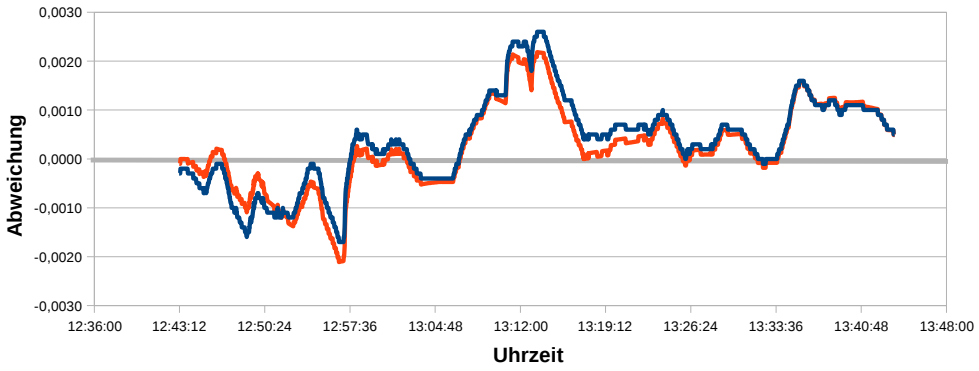
$$x_m = x_m^{gem} - \left[\sum_{i=1}^{t}(x_s^{gem_i} - x_s^{fest}) * \frac{1}{t}\right] \tag{6.9}$$

$$y_m = y_m^{gem} - \left[\sum_{i=1}^{t}(y_s^{gem_i} - y_s^{fest}) * \frac{1}{t}\right] \tag{6.10}$$

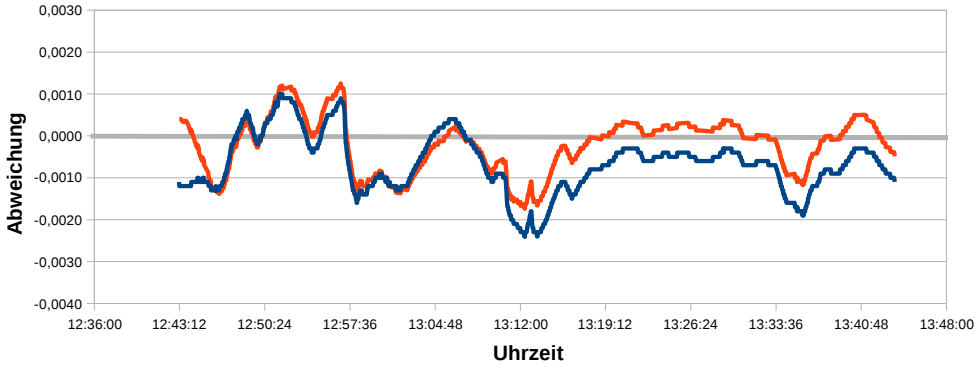
Die Graphen zur Messreihe 14 sind in Abbildung 6.9(a) und 6.9(b) dargestellt. Anfangs der Messreihe sind die berechneten Mittelwerte noch nicht sehr aussagekräftig. Sie können sich noch relativ schnell ändern, und somit springen die Graphen der neu berechneten Werte zeitweise über oder mal unter den Graphen der unveränderten Koordinaten. Mit der Zeit wird der Mittelwert stabiler und spontane Änderungen der Korrekturdaten der Referenzstation haben weniger Einfluss. Dies ist besonders gut bei den Graphen der y-Werte in Abbildung 6.9(b) zur erkennen. Hier verlaufen die beiden Graphen am Ende der Messreihe mit relativ konstantem Abstand nebeneinander. Bei dieser Messreihe ist so zum Ende hin eine stetige Verbesserung in der Genauigkeit der Koordinaten zu erkennen. Dies drückt sich auch in den Werten des statistischen Mittels, der absoluten mittleren Abweichung und der Standardabweichung aus[8]. Der Mittelwert nähert sich von 0,0004' auf 0,0003' bzw. von -0,0007' auf -0,0002'. Der absolute Mittelwert wird um 0,0001' bzw. um 0,0003' besser und die Standardabweichung verbessert sich ebenfalls um 0,0001' bzw. 0,0003'.

Durch die im vorherigen Abschnitt beschriebene Parallelität der Graphen entstehen aber auch Probleme. Aus den bisherigen Messreihen ist zu erkennen, dass zum Beispiel die Korrekturwerte der Referenzstation insgesamt wesentlich

[8] vergleiche hierzu die Tabellen 6.2 bis 6.4, Datei '14_Position_1.xls' und Datei 'Übersicht_Maßzahlen'



(a) Abweichungen der x-Koordinate unbearbeitet und korrigiert

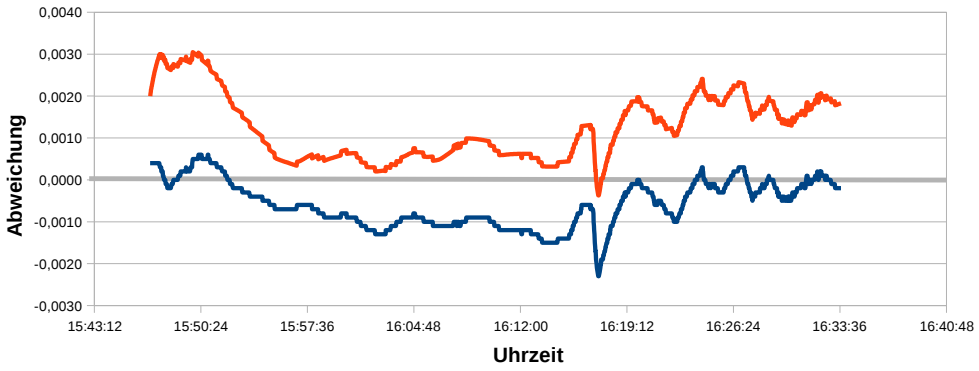


(b) Abweichungen der y-Koordinate unbearbeitet und korrigiert

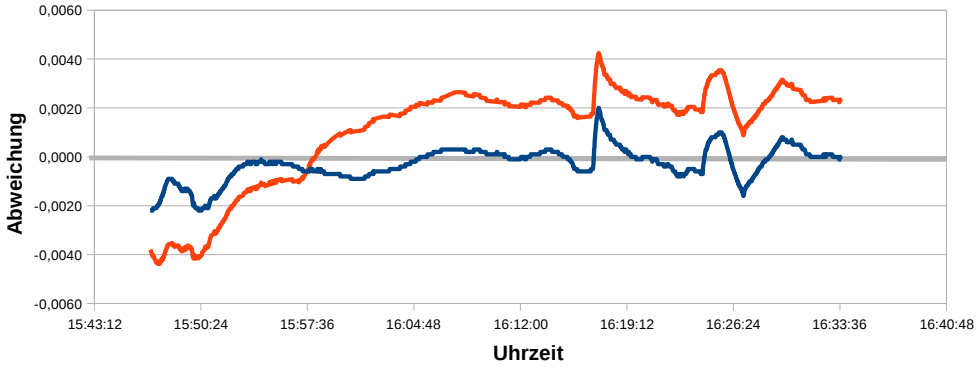
Abbildung 6.9: Graphische Darstellung von Messreihe 14 mit Korrektur durch Formel 5

schlechter als die der mobilen Einheit sein können. Dadurch kann es vorkommen, dass die vielleicht guten Werte der mobilen Messeinheit konstant durch die schlechten Korrekturwerte der Referenzstation negativ beeinflusst werden (Abbildung 6.10). Die entsprechenden Werte für die Standardabweichung der Datenreihe 16 (vgl. Tabelle auf Seite 150) untersteichen die Problematik. Bei den x-Werten tritt eine Verschlechterung von 0,0008' auf 0,0015' auf; bei den y-Werten sogar von 0,0007' auf 0,0024'.

Auch wenn die Korrekturwerte der Referenzstation vergleichsweise gut sind, kommt es bei Messdaten der mobilen Einheit, die über einen längeren Zeitraum um die x-Achse schwingen, zu Problemen. Durch den relativ konstanten Korrekturwert befindet sich der Graph der korrigierten Daten entweder permanent über oder permanet unter dem der originalen Messdaten. Somit kann über diesen, um die x-Achse schwingenden Bereich, keine stetige Verbesserung erzielt werden. Aufgrund dieser nicht lösbaren Schwierigkeiten ist auch dieser Ansatz gescheitert.



(a) Abweichungen der x-Koordinate unbearbeitet und korrigiert



(b) Abweichungen der y-Koordinate unbearbeitet und korrigiert

Abbildung 6.10: Graphische Darstellung von Messreihe 16 mit Korrektur durch Formel 5

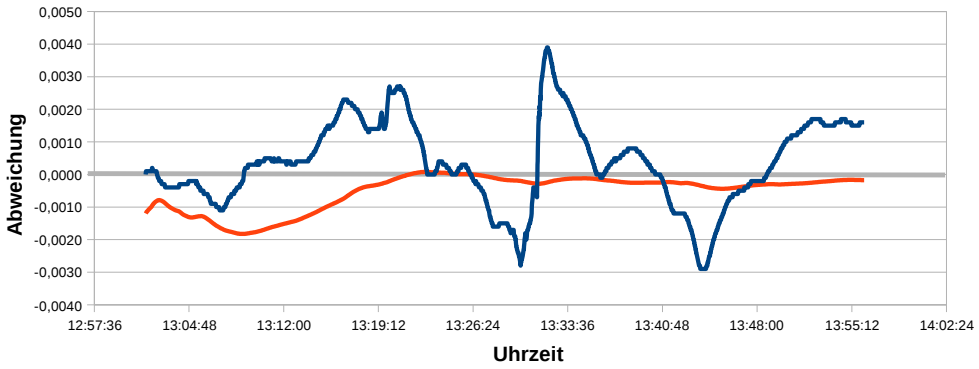
6.2.6 Versuchsreihe 6

Der nächste Ansatz basiert, wie schon anfangs des vorherigen Abschnittes erwähnt, ebenfalls auf dem statistischen Mittelwert. Hier wird, wie beim letzten Ansatz, der Mittelwert der Abweichung der bisherigen Messdaten der Referenzstation verwendet. Dieser Korrekturwert wird jedoch dieses Mal nicht direkt mit den aktuellen Messdaten verrechnet, sondern es wird zusätzlich noch der Mittelwert der Messdaten der mobilen Einheit gebildet. Man subtrahiert also vom Mittelwert der Breiten- und Längenkoordinaten den jeweiligen Mittelwert der Abweichungen der Referenzstation. Die entsprechende Zuordnungsvorschrift für die Breitenkoordinate ist in Formel 6.11 bzw. für die Längenkoordinate in Formel 6.12 angegeben.

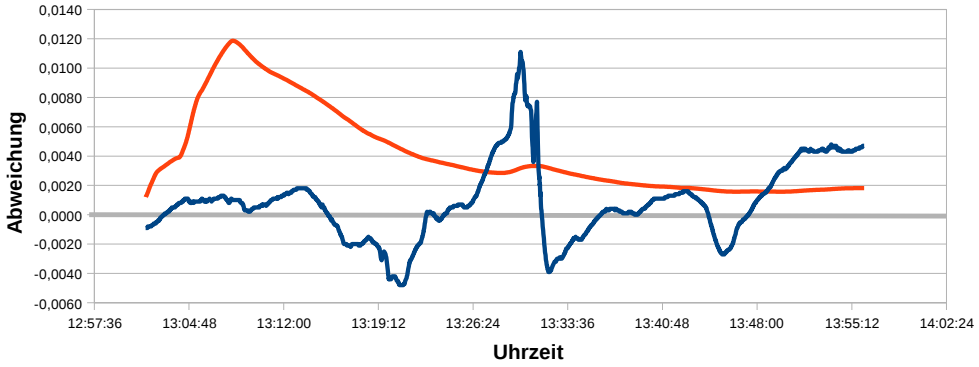
FORMEL 6

$$x_m = \left[\sum_{i=1}^{t} x_m^{gem_i} * \frac{1}{t}\right] - \left[\sum_{i=1}^{t} (x_s^{gem_i} - x_s^{fest}) * \frac{1}{t}\right] \tag{6.11}$$

$$y_m = \left[\sum_{i=1}^{t} y_m^{gem_i} * \frac{1}{t}\right] - \left[\sum_{i=1}^{t} (y_s^{gem_i} - y_s^{fest}) * \frac{1}{t}\right] \tag{6.12}$$



(a) Abweichungen der x-Koordinate unbearbeitet und korrigiert



(b) Abweichungen der y-Koordinate unbearbeitet und korrigiert

Abbildung 6.11: Graphische Darstellung von Messreihe 17 mit Korrektur durch Formel 6

Die Graphen der Messreihe 17 (Abb. 6.11) weisen, abgesehen von den bereits erwähnten Problemen zum Beginn der Korrektur (siehe hier ca. 13.04 - 13.10 Uhr bei y-Koordinate), einen schon sehr guten Verlauf auf. Mit zunehmender Dauer nähern sich die Graphen der x-Achse an, auch wenn der Graph der Längenkoordinate zum Ende hin eine leichte Tendenz nach oben besitzt. Insgesamt ist der Verlauf beider Graphen aber recht ruhig. Die statistischen Maßzahlen der Messreihe untermauern allerdings diese Verbesserung nicht uneingeschränkt (Tabellen 6.2 bis 6.4, Datei 'Übersicht_Maßzahlen.xls'). Zum Beispiel ist der

Mittelwert bei der Breitenkoordinate etwas und bei der Längenkoordinate sogar extrem (um 0,0034') schlechter. Bei der absoluten mittleren Abweichung und der Standardabweichung sind die Werte für die x-Koordinate besser, bei der y-Koordinate hingegen schlechter geworden. Die Verschlechterung ist jedoch leicht zu begründen, da der Graph der Datenreihe sich noch relativ weit von der x-Achse entfernt befindet und der Graph der Originaldaten zeitweise nahe der x-Achse verläuft.

Nach den vielversprechenden Ergebnissen der Messreihe 17, wiesen die nächsten Messungen allerdings wieder Probleme auf. In Abbildung 6.12 ist der Verlauf des Graphen der x-Koordinate der Messreihe '18_Position_2' zu sehen. Anders als bei den letzten Messungen konvergiert hier der Graph nicht gegen die x-Achse, sondern divergiert von ihr. Allerdings ist die Standardabweichung der korrigierten Breitenkoordinate leicht verbessert (von 0,0009' auf 0,0008'). Dies liegt an dem relativ gleichmäßigen Verlauf des Graphen, im Gegensatz zum Graphen der Originaldaten, der viele extreme Werte aufweist. Die weiteren Messreihen brachten durchweg uneinheitliche Ergebnisse, die auf Schwierigkeiten der Basis dieses Ansatzes deuten. Auf ein weiteres Problem dieser Herangehensweise wird am Ende des nächsten Abschnittes eingegangen.

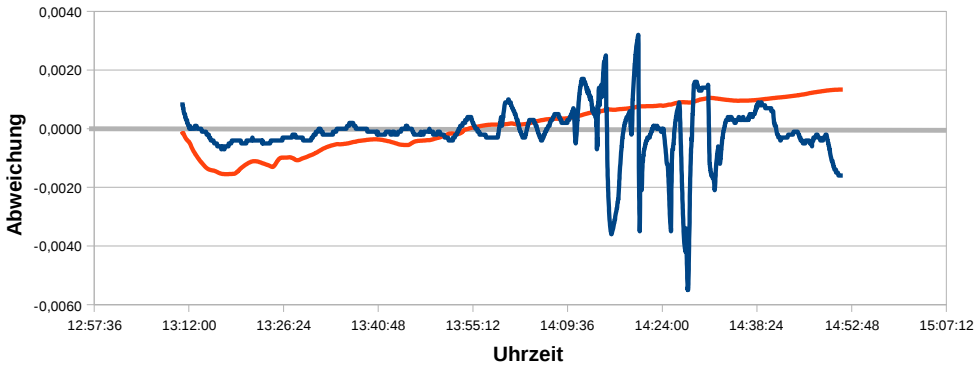


Abbildung 6.12: Graphische Darstellung der Abweichung der x-Koordinate von Messreihe 18 mit Korrektur durch Formel 6 unbearbeitet und korrigiert

6.2.7 Versuchsreihe 7

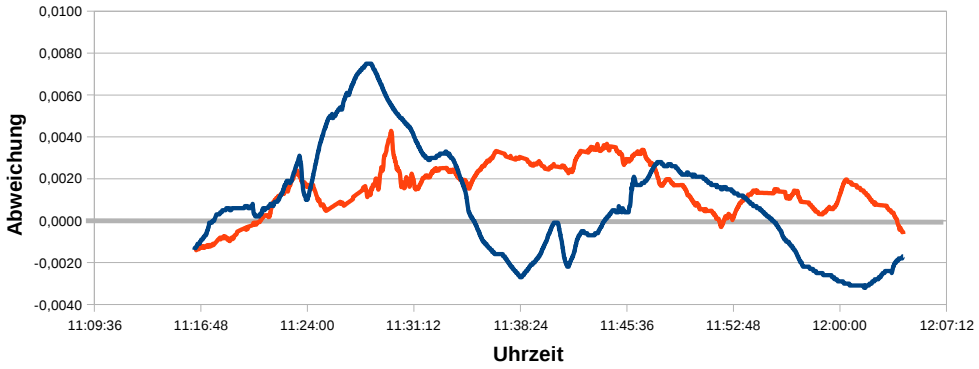
Da der Ansatz in Versuchsreihe 6 als Grundlage nur die beiden statistischen Mittelwerte verwendet, wird in dieser Messreihe versucht, noch eine aktuelle Komponente hinzu zu nehmen, nämlich die aktuelle Abweichung der Messung der Referenzstation. Dieser Korrekturwert wird vom Mittelwert der Abweichung

nochmals abgezogen. Somit besteht der Ansatz zum Einen aus statistischen Werten, welche die vergangenen Messungen mit berücksichtigen, zum Anderen aus aktuellen Aspekten, welche momentane Gegebenheiten (wie z.B. Satellitenkonstellation) mit einfließen lassen. Zur Neuberechnung der Breiten- und Längenkoordinaten werden folgende Zuordnungvorschriften verwendet:

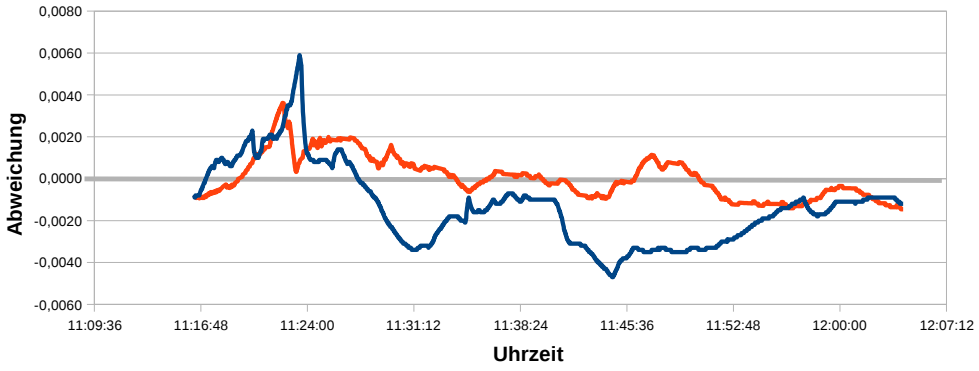
FORMEL 7

$$x_m = \left[\sum_{i=1}^{t} x_m^{gem_i} * \frac{1}{t}\right] - \left[\left(\sum_{i=1}^{t}(x_s^{gem_i} - x_s^{fest}) * \frac{1}{t}\right) - x_s^{gem_t}\right] \tag{6.13}$$

$$y_m = \left[\sum_{i=1}^{t} y_m^{gem_i} * \frac{1}{t}\right] - \left[\left(\sum_{i=1}^{t}(y_s^{gem_i} - y_s^{fest}) * \frac{1}{t}\right) - y_s^{gem_t}\right] \tag{6.14}$$



(a) Abweichungen der x-Koordinate unbearbeitet und korrigiert



(b) Abweichungen der y-Koordinate unbearbeitet und korrigiert

Abbildung 6.13: Graphische Darstellung von Messreihe 21 mit Korrektur durch Formel 7

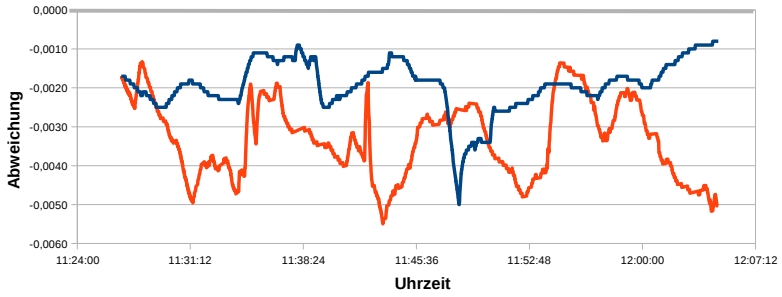
An den Graphen der Messreihe '21_Position_3' (Abbildung 6.13) ist gut zu erkennen, dass der Graph nicht mehr so glatt wie in Versuchsreihe 6 verläuft.

Dies ist auf die Verwendung der aktuellen Komponente bei der Berechnung zurückzuführen. Insgesamt ist der Verlauf der Graphen der neu berechneten Werte, sowohl der Breiten- als auch der Längenkoordinate, zufriedenstellend. Sie weisen beide keine so extremen Abweichungen auf wie die der Originaldaten und sie befinden sich, über die gesamte Zeitspanne gesehen, auch näher an der x-Achse. Dies bestätigen auch die absolute mittlere Abweichung und die Standardabweichung. Diese Maßzahlen verbessern sich in beiden Fällen sowohl für die x-Koordinate als auch für die y-Koordinate erheblich (bis zu 0,0013').

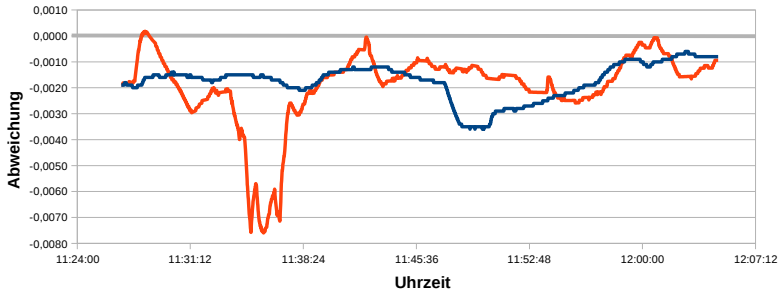
Wie auch bei den anderen Versuchen ergaben weitere Messreihen keine dauerhafte Verbesserung der Positionsgenauigkeit. Ersichtlich ist die extreme Verschlechterung beider Koordinaten in der Messreihe '01_Position_1'. In Abbildung 6.14 ist der Verlauf der Graphen für die beiden Koordinaten abgebildet. Es ist gut zu erkennen, dass der Verlauf durch die Korrektur wesentlich unruhiger wird, und dass auch die extremen Abweichungen größer geworden sind. Somit ist auch dieser Ansatz gescheitert.

Die beiden Abbildungen 6.15(a) und 6.15(b) verdeutlichen anschaulich die Problematik, die zum Scheitern aller hier erwähnten Ansätze geführt hat. Die Abbildungen zeigen die Abweichung der Breitenkoordinate (x-Werte) von den Soll-Koordinaten der stationären Referenzstation und der mobilen Einheit für die Messreihen '01_Position_1' und '18_Position_2'. In beiden Fällen sind keine Gemeinsamkeiten im Verlauf der Graphen zu erkennen. Für die schlechten Ergebnisse ist auch nicht ein einzelnes Modul verantwortlich, sondern wie beispielsweise Messreihe 1 gezeigt hat, weisen beide Module einen sehr unruhigen und schlechten Verlauf auf.

Wie bereits am Ende des letzten Abschnittes erwähnt, beinhalten die beiden letzten Ansätze noch ein weiteres Problem, nämlich die Verwendung des Koordinatenmittelwertes der mobilen Einheit, was allerdings für die durchgeführten Messreihen keine Rolle spielt. Dabei wurde die mobile Einheit an drei verschiedenen Standorten während der Messungen ausschließlich stationär eingesetzt, sodass der dann verwendete Mittelwert zur Verbesserung der Messergebnisse sinnvoll gewesen ist. Die Aufgabenstellung dieser Arbeit umfasst allerdings die Verbesserung der Positionsbestimmung einer sich bewegenden Einheit, wo ein Mittelwert der vergangenen Positionsdaten keinen Sinn macht.

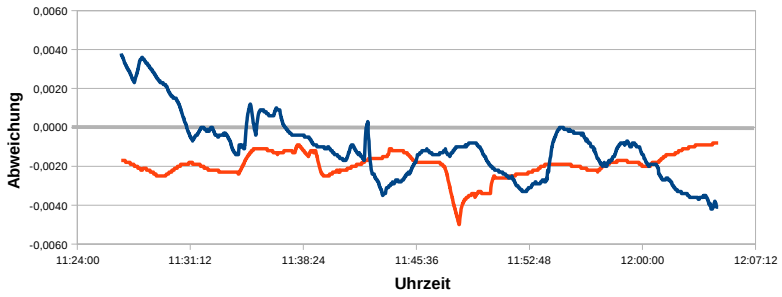


(a) Abweichungen der x-Koordinate unbearbeitet und korrigiert

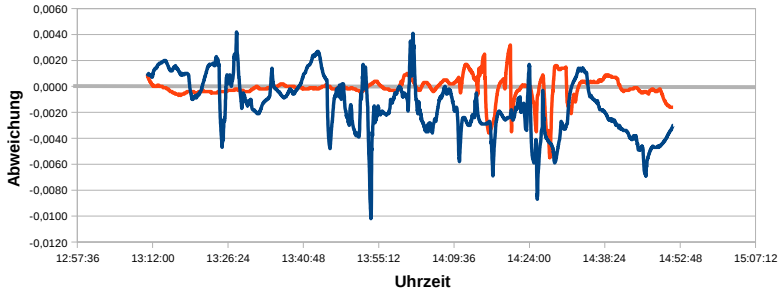


(b) Abweichungen der y-Koordinate unbearbeitet und korrigiert

Abbildung 6.14: Graphische Darstellung von Messreihe 1 mit Korrektur durch Formel 7



(a) Messreihe 1, Wert der stationären und mobilen Einheit



(b) Messreihe 18, Wert der stationären und mobilen Einheit

Abbildung 6.15: Graphische Darstellung der Abweichung der x-Koordinate der Referenzstation und der mobilen Einheit von ihren Sollwerten

6.3 Übergreifende Auswertung aller Messreihen

6.3.1 Betrachtung einiger statistischer Maßzahlen

Die bisher betrachteten Messreihen sind exemplarisch zur Analyse ausgewählt worden. Dazwischen wurden teilweise noch weitere Messungen durchgeführt, die sich aber nicht für eine intensivere Betrachtung eigneten. Um eine noch fundiertere Auswertung zu erhalten, wurden insgesamt 44 Messreihen durchgeführt und konnten zur Analyse verwendet werden. Ihre statistischen Maßzahlen sind im Einzelnen in der Datei 'Übersicht_Maßzahlen.xls' auf der beiliegenden Daten-CD zusammengefasst.

Interessant ist der Vergleich zwischen den ursprünglichen, nicht korrigierten Werten der mobilen Station (Spalten 'M X' und 'M Y') und den durch die einzelnen Formeln korrigierten Werten (Spalte 'F1 X', 'F1 Y, 'F2 X', 'F2 Y' usw.). In den Tabellen 6.2 bis 6.4 sind die Werte, welche durch die Korrekturrechnung besser geworden sind, grün hinterlegt. Werte, die sich durch die Korrektur verschlechtert haben, sind rot hinterlegt. Am Ende jeder Tabelle ist neben dem Mittelwert der verschiedenen Maßzahlen aller Messreihen noch die Anzahl der besser, schlechter und gleich gebliebenen Werte aufgelistet.

Betrachtet man die Maßzahl des arithmetischen Mittelwertes (Tabelle 6.2), so sind bei den sieben verschiedenen Korrekturformeln im Durchschnitt 16,71 der 44 Messreihen durch die Korrektur besser geworden. Formel 2 weist die größte Anzahl an Verbesserungen auf. 24 bei der x-Koordinate und 30 Verbesserungen bei der y-Koordinate. Allerdings ist einzig beim Einsatz der Formel 4 die durchschnittliche Verbesserung des Mittelwertes sowohl bei der x-Koordinate (0,0001') als auch bei der y-Koordinate (-0,0004')[9] zu verzeichnen. Insgesamt werden die meisten Mittelwerte der 44 Messreihen bei allen Formeln schlechter. Wie jedoch schon am Anfang des Kapitels 6.2 erwähnt, ist der arithmetische Mittelwert auch nicht die geeignete Maßzahl zur Bewertung dieser Messreihen.

Schon beim ersten Blick auf die Tabelle der mittleren absoluten Abweichungen (Tabelle 6.3) ist zu erkennen, dass die Verbesserungen durch die verschiedenen Formeln stark zurückgegangen sind. Waren es bei der Betrachtung des arithmetischen Mittelwertes noch 16,71 Verbesserungen im Durchschnitt, so sind es bei der Betrachtung der mittleren absoluten Abweichungen nur noch 9,93. Bei intensiverem Studium der einzelnen Messreihen stellt man fest, dass viele Werte durch die Korrektur mit den einzelnen Formeln beträchtlich schlechter gewor-

[9] der Übersicht wegen sind die Werte in den Zellen auf vier Nachkommastellen gerundet

den sind. Dies spiegelt sich auch in den Mittelwerten der durch die Formeln korrigierten Werte wider. War der Durchschnitt der korrigierten Werte einiger Ansätze beim arithmetischen Mittel noch besser geworden, so ist bei der mittleren absoluten Abweichung keine über alle Messreihen durchgängig Verbesserung erzielt worden. Auffällig bei der Betrachtung der Tabelle ist, dass bei manchen Messreihen fast alle Formeln zur Verbesserung der mittleren absoluten Abweichungen führen. Dies lässt darauf schließen, dass in dieser Versuchsreihe der Verlauf der von der Referenzstation und der mobilen Einheit aufgezeichneten Daten, sich mehr ähneln, als in den anderen Messreihen.

Weniger aussagekräftig sind die Maßzahlen der minimalen und maximalen Werte (auf der beiliegenden CD zu finden) da es bei dieser Arbeit um eine kontinuierliche Verbesserung der Positionsdaten geht. Wie auch bei den mittleren absoluten Abweichungen, ist hier zu erkennen, dass durch die Korrektur ein Großteil der Werte eher schlechter als besser wurde. Durchschnittlich ist über alle Formeln hinweg 29,50 Mal der minimale Wert schlechter geworden (bei 44 Messreihen) und bei den maximalen Werten sogar 34,14 Mal. Aber auch diese Zahlen bekräftigen, dass die Korrektur durch die verschiedenen Formeln fehlgeschlagen ist.

Die Ergebnisse der mittleren absoluten Abweichungen ähneln denen der Standardabweichungen. Diese leitet sich von der Varianz (vgl. Anhang A.5.4) ab, welche an sich nur eine untergeordnete Rolle spielt. In Tabelle 6.4 sind die Standardabweichungen aller Messreihen aufgelistet[10].

[10] Diese Tabelle ist auch als Tabellenblatt in der Datei 'Übersicht_Maßzahlen' zu finden

Messreihe	Ort	S X	S Y	S Sat	S Hdop	M X	M Y	M Sat	M Hdop	F1 X	F1 Y	F2 X	F2 Y	F3 X	F3 Y	F4 X	F4 Y	F5 X	F5 Y	F6 X	F6 Y	F7 X	F7 Y
1	Position 1	-0,0010	-0,0012	7,2904	2,1791	-0,0019	-0,0017	8,1187	1,6845	0,0009	0,0005	-0,0017	-0,0015	0,0009	0,0005	-0,0015	-0,0012	-0,0023	-0,0007	-0,0023	-0,0007	-0,0033	-0,0019
2	Position 3	0,0006	-0,0005	9,1668	1,0210	-0,0001	-0,0015	9,2482	0,9604	0,0007	0,0009	0,0001	-0,0003	-0,0002	-0,0004	-0,0004	-0,0012	-0,0004	-0,0014	0,0001	-0,0016	0,0006	-0,0021
3	Position 2	-0,0027	-0,0015	8,0535	1,1022	0,0007	-0,0003	7,5214	1,2118	-0,0034	-0,0012	-0,0021	-0,0005	-0,0012	0,0002	0,0021	0,0005	0,0030	0,0015	0,0030	0,0010	0,0003	-0,0005
4	Position 1	-0,0014	-0,0002	7,2366	1,3360	0,0001	0,0003	8,1802	1,1276	-0,0016	-0,0005	0,0001	0,0004	-0,0005	-0,0005	0,0008	0,0004	0,0011	0,0003	0,0014	0,0003	-0,0001	0,0002
5	Position 3	-0,0006	-0,0017	7,0526	1,4059	0,0003	-0,0023	6,5666	1,4746	-0,0009	0,0006	-0,0020	-0,0010	-0,0009	0,0006	0,0008	-0,0014	-0,0011	-0,0012	-0,0025	-0,0015	-0,0031	-0,0032
6	Position 2	-0,0006	0,0015	8,0764	1,2560	-0,0016	0,0023	8,2199	1,0706	0,0010	-0,0008	-0,0014	0,0020	-0,0012	0,0017	-0,0013	0,0016	-0,0014	-0,0005	-0,0016	-0,0005	-0,0022	0,0009
7	Position 2	-0,0009	-0,0003	7,2963	1,6678	-0,0009	0,0012	7,9153	1,0626	0,0000	-0,0015	-0,0007	0,0010	-0,0013	0,0006	-0,0006	0,0013	0,0009	0,0026	0,0012	0,0023	0,0003	0,0019
8	Position 1	-0,0010	-0,0012	7,2904	2,1791	-0,0019	-0,0017	8,1187	1,6845	0,0009	0,0005	-0,0017	-0,0015	0,0009	0,0005	-0,0015	-0,0012	-0,0023	-0,0007	-0,0023	-0,0007	-0,0033	-0,0019
9	Position 1	-0,0005	0,0011	7,7219	1,6602	0,0000	0,0008	8,2525	1,5374	-0,0004	0,0003	0,0001	0,0009	0,0002	0,0008	0,0002	0,0003	0,0012	0,0003	0,0013	0,0003	0,0008	0,0014
10	Position 1	-0,0020	-0,0008	7,1967	1,4330	-0,0002	0,0001	7,9272	1,1721	-0,0018	-0,0008	-0,0004	-0,0003	-0,0011	-0,0010	0,0007	0,0004	0,0022	0,0006	0,0025	0,0003	0,0005	-0,0004
11	Position 2	-0,0019	0,0001	8,0089	1,1862	-0,0019	0,0024	8,3242	1,0455	0,0000	-0,0023	-0,0018	0,0020	-0,0018	0,0017	-0,0010	0,0023	0,0006	0,0032	0,0003	0,0029	-0,0016	0,0030
12	Position 2	-0,0003	0,0193	6,9779	1,3502	-0,0022	-0,0210	7,7443	1,0639	0,0019	0,0402	-0,0019	-0,0171	-0,0013	-0,0087	-0,0021	-0,0295	-0,0015	-0,0401	-0,0016	-0,0394	-0,0020	-0,0201
13	Position 3	-0,0010	-0,0013	6,3235	1,6664	-0,0007	-0,0013	7,7981	1,2588	-0,0003	0,0000	-0,0005	-0,0012	-0,0007	-0,0012	-0,0001	-0,0008	0,0017	0,0004	0,0023	0,0018	0,0013	0,0004
14	Position 1	0,0000	-0,0006	8,7145	0,9849	0,0004	-0,0007	8,1759	1,0566	-0,0004	0,0001	0,0000	-0,0001	0,0004	-0,0005	0,0004	-0,0004	0,0003	-0,0002	-0,0003	-0,0001	-0,0002	-0,0007
15	Position 3	0,0007	0,0015	8,2003	1,4277	-0,0009	0,0003	8,3290	1,2648	0,0017	0,0012	-0,0003	0,0001	0,0005	0,0015	-0,0012	-0,0004	-0,0013	-0,0019	-0,0017	-0,0008	-0,0010	0,0006
16	Position 1	-0,0020	-0,0023	7,9658	1,0661	-0,0006	-0,0003	6,6978	1,3498	-0,0014	-0,0021	-0,0014	-0,0014	-0,0010	-0,0023	0,0006	0,0010	0,0012	0,0011	0,0014	0,0006	-0,0006	-0,0017
17	Position 3	0,0005	-0,0010	7,6532	1,1391	0,0003	0,0008	7,6734	1,1557	0,0002	-0,0018	0,0000	0,0009	0,0004	-0,0002	0,0001	0,0013	-0,0004	0,0047	-0,0005	0,0042	0,0000	0,0032
18	Position 2	-0,0015	-0,0016	6,9519	1,3231	-0,0001	-0,0004	7,4120	1,2521	-0,0013	-0,0012	-0,0002	-0,0006	-0,0010	-0,0009	0,0005	0,0004	0,0001	0,0008	0,0001	0,0011	-0,0014	-0,0005
19	Position 3	0,0000	-0,0010	7,7603	1,1483	-0,0003	0,0006	7,3656	1,2587	0,0003	-0,0017	-0,0006	0,0001	-0,0007	-0,0005	-0,0003	0,0011	-0,0007	0,0019	-0,0010	0,0013	-0,0010	0,0003
20	Position 1	0,0005	-0,0012	8,6447	1,1082	0,0002	-0,0001	8,7921	1,0361	0,0003	-0,0010	0,0005	-0,0012	-0,0001	0,0006	-0,0001	0,0005	-0,0005	0,0002	0,0000	-0,0003	0,0005	-0,0014
21	Position 3	-0,0006	-0,0007	7,7046	1,8490	0,0008	-0,0013	7,7327	1,5681	-0,0015	0,0007	0,0006	0,0002	-0,0015	0,0007	0,0011	-0,0010	0,0015	-0,0003	0,0022	0,0007	0,0015	0,0001
22	Position 1	-0,0011	0,0010	8,2310	1,3258	-0,0005	-0,0005	8,3192	1,1934	-0,0005	0,0016	-0,0006	-0,0003	-0,0005	0,0000	0,0000	-0,0010	0,0004	-0,0027	0,0005	-0,0022	-0,0006	-0,0012
23	Position 2	-0,0009	0,0008	7,4877	1,2867	-0,0011	0,0015	8,2453	1,0597	0,0002	-0,0007	-0,0010	0,0011	-0,0013	0,0007	-0,0007	0,0011	0,0002	0,0013	0,0007	0,0009	-0,0002	0,0018
24	Position 1	-0,0004	0,0005	6,2033	1,5018	-0,0004	-0,0011	7,2394	1,1720	0,0000	0,0016	-0,0004	-0,0008	0,0000	-0,0001	-0,0001	-0,0013	-0,0011	-0,0015	-0,0010	-0,0013	-0,0014	-0,0008
25	Position 3	-0,0013	-0,0009	6,7976	1,2855	-0,0001	-0,0010	7,6252	1,1636	-0,0012	0,0001	-0,0004	-0,0022	-0,0003	-0,0018	0,0006	-0,0006	0,0008	-0,0005	0,0009	-0,0014	-0,0004	-0,0023
26	Position 3	-0,0014	-0,0032	6,8866	1,4633	0,0001	-0,0006	6,8309	1,5107	-0,0015	-0,0026	-0,0002	-0,0011	-0,0015	-0,0026	0,0008	0,0010	0,0009	0,0018	0,0007	0,0027	-0,0007	-0,0006
27	Position 3	-0,0024	-0,0010	6,9075	1,4837	-0,0004	0,0012	6,4702	1,6360	-0,0020	-0,0022	-0,0010	-0,0013	-0,0019	-0,0018	0,0009	0,0017	0,0008	0,0012	0,0007	0,0016	-0,0017	0,0006
28	Position 3	-0,0010	0,0001	6,5967	1,3392	-0,0009	-0,0010	7,8025	1,1832	-0,0001	0,0010	-0,0006	-0,0006	-0,0011	-0,0005	-0,0005	-0,0010	0,0002	-0,0013	0,0008	-0,0004	-0,0002	-0,0003
29	Position 1	-0,0011	0,0006	8,2575	1,6099	0,0001	-0,0005	8,6793	1,2952	-0,0012	0,0011	0,0003	-0,0003	-0,0012	0,0011	0,0006	-0,0008	0,0025	-0,0011	0,0023	-0,0008	0,0011	-0,0002
30	Position 3	-0,0012	0,0001	6,5528	1,5704	-0,0005	0,0027	6,9354	1,5269	-0,0006	-0,0026	-0,0004	0,0000	-0,0007	-0,0017	0,0000	0,0027	-0,0003	0,0019	0,0005	0,0028	-0,0006	0,0029
31	Position 3	-0,0032	-0,0021	7,6341	1,1737	0,0006	0,0013	7,6724	1,1626	-0,0037	-0,0035	-0,0013	-0,0010	-0,0004	-0,0006	0,0022	0,0024	0,0027	0,0028	0,0020	0,0024	-0,0012	0,0002
32	Position 3	-0,0015	-0,0025	6,7891	1,3819	-0,0004	-0,0003	6,6351	1,4388	-0,0011	-0,0023	-0,0012	-0,0007	-0,0018	-0,0016	0,0004	0,0010	0,0016	0,0020	0,0006	0,0032	-0,0008	0,0007
33	Position 1	-0,0033	-0,0016	7,0779	1,3849	-0,0003	0,0002	7,6005	1,2236	-0,0030	-0,0017	-0,0004	-0,0001	-0,0012	-0,0007	0,0012	0,0009	0,0019	0,0014	0,0020	0,0015	-0,0013	-0,0001
34	Position 3	-0,0017	-0,0009	7,0559	1,3924	-0,0005	0,0010	7,6540	1,3676	-0,0011	-0,0019	-0,0008	-0,0008	-0,0014	-0,0021	0,0003	0,0015	0,0012	0,0027	0,0022	0,0043	0,0006	0,0035
35	Position 1	-0,0025	-0,0010	7,7761	1,1601	0,0007	0,0003	7,7833	1,1685	-0,0031	-0,0013	-0,0004	-0,0003	-0,0008	-0,0006	0,0019	0,0008	0,0044	0,0021	0,0043	0,0024	0,0019	0,0014
36	Position 2	0,0001	-0,0014	6,8311	1,3408	0,0016	0,0010	7,0315	1,2385	-0,0015	-0,0024	0,0011	0,0002	-0,0012	-0,0015	0,0015	0,0017	0,0009	0,0022	0,0004	0,0022	0,0005	0,0008
37	Position 1	-0,0007	0,0003	8,7559	1,3709	-0,0003	-0,0014	8,6931	1,2186	-0,0004	0,0017	-0,0004	-0,0002	-0,0004	0,0015	0,0000	-0,0015	0,0018	-0,0015	0,0011	-0,0013	0,0004	-0,0011
38	Position 1	-0,0009	-0,0004	7,6715	1,2397	0,0004	0,0006	7,8660	1,1373	-0,0013	-0,0010	-0,0002	-0,0001	-0,0002	-0,0001	0,0009	0,0008	0,0032	0,0022	0,0027	0,0020	0,0018	0,0016
39	Position 2	0,0022	0,0028	8,1194	1,4301	-0,0001	0,0016	7,9454	1,4809	0,0023	0,0012	0,0000	0,0014	0,0008	0,0016	-0,0012	0,0002	-0,0018	-0,0005	-0,0023	-0,0002	-0,0001	0,0026
40	Position 1	-0,0001	0,0012	8,1845	1,5956	-0,0010	-0,0001	8,6066	1,3475	0,0008	0,0014	-0,0002	0,0001	0,0008	0,0014	-0,0009	-0,0007	-0,0001	-0,0011	-0,0010	-0,0009	-0,0011	0,0003
41	Position 1	-0,0021	-0,0018	7,2753	1,3739	-0,0002	-0,0004	7,4217	1,2625	-0,0020	-0,0014	-0,0005	-0,0006	-0,0018	-0,0014	0,0009	0,0005	0,0014	0,0010	0,0014	0,0011	-0,0008	-0,0007
42	Position 2	0,0010	-0,0005	8,4923	0,9736	-0,0008	0,0007	9,2511	0,9303	0,0018	-0,0012	-0,0002	0,0000	-0,0008	0,0007	-0,0013	0,0009	-0,0020	0,0012	-0,0024	0,0013	-0,0014	0,0008
43	Position 1	-0,0003	0,0027	8,6367	1,1033	-0,0002	0,0006	8,2345	1,0503	-0,0002	0,0022	0,0000	0,0009	0,0000	0,0009	0,0000	-0,0008	0,0001	-0,0027	-0,0001	-0,0023	-0,0004	0,0005
44	Position 1	-0,0021	-0,0007	6,6525	1,5066	0,0000	0,0003	8,0929	1,1414	-0,0020	-0,0010	0,0000	0,0003	-0,0008	-0,0010	0,0009	0,0006	0,0021	0,0010	0,0024	0,0008	0,0003	0,0001
	Mittelwert	-0,0009	0,0000	7,5491	1,3814	-0,0003	-0,0004	7,8352	1,2547	-0,0006	0,0004	-0,0006	-0,0006	-0,0007	-0,0004	0,0001	-0,0004	0,0005	-0,0004	0,0005	-0,0002	-0,0005	-0,0003

																							Mittelwert
besser										16	14	24	30	15	22	21	14	13	13	13	11	11	17	16,71
schlechter										28	30	20	14	28	21	23	30	31	31	31	33	33	27	27,14
gleich										0	0	0	0	1	1	0	0	0	0	0	0	0	0	0,14

Tabelle 6.2: Übersicht arithmetisches Mittel aus 'Übersicht_Maßzahlen.xls'

Messreihe	Ort	S X	S Y	M X	M Y	F1 X	F1 Y	F2 X	F2 Y	F3 X	F3 Y	F4 X	F4 Y	F5 X	F5 Y	F6 X	F6 Y	F7 X	F7 Y
1	Position 1	0,0016	0,0014	0,0019	0,0017	0,0017	0,0012	0,0019	0,0016	0,0017	0,0012	0,0016	0,0013	0,0023	0,0009	0,0023	0,0007	0,0033	0,0019
2	Position 3	0,0010	0,0008	0,0009	0,0019	0,0011	0,0019	0,0009	0,0019	0,0008	0,0020	0,0010	0,0018	0,0010	0,0020	0,0003	0,0016	0,0010	0,0021
3	Position 2	0,0037	0,0026	0,0007	0,0010	0,0042	0,0029	0,0033	0,0024	0,0023	0,0017	0,0025	0,0018	0,0030	0,0017	0,0030	0,0011	0,0027	0,0022
4	Position 1	0,0021	0,0017	0,0005	0,0006	0,0021	0,0019	0,0005	0,0006	0,0011	0,0014	0,0011	0,0011	0,0011	0,0007	0,0014	0,0005	0,0018	0,0018
5	Position 3	0,0026	0,0018	0,0017	0,0023	0,0037	0,0018	0,0028	0,0020	0,0037	0,0018	0,0026	0,0016	0,0026	0,0013	0,0028	0,0015	0,0032	0,0032
6	Position 2	0,0011	0,0022	0,0016	0,0023	0,0014	0,0017	0,0015	0,0021	0,0014	0,0020	0,0014	0,0017	0,0014	0,0011	0,0016	0,0008	0,0024	0,0016
7	Position 2	0,0017	0,0030	0,0009	0,0012	0,0018	0,0031	0,0010	0,0013	0,0014	0,0016	0,0010	0,0017	0,0009	0,0026	0,0012	0,0023	0,0014	0,0030
8	Position 1	0,0016	0,0014	0,0019	0,0017	0,0017	0,0012	0,0019	0,0016	0,0017	0,0012	0,0016	0,0013	0,0023	0,0009	0,0023	0,0007	0,0033	0,0019
9	Position 1	0,0017	0,0014	0,0009	0,0009	0,0023	0,0012	0,0011	0,0011	0,0019	0,0012	0,0015	0,0007	0,0016	0,0006	0,0014	0,0004	0,0019	0,0015
10	Position 1	0,0020	0,0019	0,0004	0,0007	0,0019	0,0019	0,0007	0,0009	0,0014	0,0016	0,0009	0,0010	0,0022	0,0007	0,0025	0,0006	0,0015	0,0016
11	Position 2	0,0024	0,0021	0,0019	0,0024	0,0019	0,0026	0,0019	0,0024	0,0019	0,0024	0,0013	0,0023	0,0006	0,0032	0,0004	0,0029	0,0022	0,0034
12	Position 2	0,0006	0,0193	0,0023	0,0210	0,0020	0,0402	0,0023	0,0222	0,0023	0,0246	0,0022	0,0295	0,0017	0,0401	0,0017	0,0393	0,0020	0,0201
13	Position 3	0,0018	0,0014	0,0012	0,0018	0,0026	0,0015	0,0014	0,0018	0,0022	0,0021	0,0017	0,0016	0,0026	0,0011	0,0023	0,0018	0,0017	0,0011
14	Position 1	0,0009	0,0017	0,0008	0,0008	0,0012	0,0017	0,0010	0,0013	0,0008	0,0009	0,0009	0,0010	0,0007	0,0005	0,0004	0,0003	0,0009	0,0016
15	Position 3	0,0016	0,0017	0,0017	0,0018	0,0021	0,0023	0,0017	0,0018	0,0018	0,0022	0,0017	0,0018	0,0019	0,0024	0,0017	0,0008	0,0017	0,0013
16	Position 1	0,0023	0,0029	0,0007	0,0005	0,0019	0,0030	0,0018	0,0023	0,0014	0,0026	0,0010	0,0017	0,0012	0,0022	0,0014	0,0019	0,0017	0,0025
17	Position 3	0,0010	0,0029	0,0011	0,0020	0,0014	0,0040	0,0012	0,0021	0,0012	0,0021	0,0011	0,0028	0,0012	0,0048	0,0005	0,0042	0,0009	0,0045
18	Position 2	0,0021	0,0021	0,0005	0,0005	0,0021	0,0019	0,0007	0,0008	0,0014	0,0013	0,0011	0,0009	0,0010	0,0011	0,0007	0,0013	0,0018	0,0017
19	Position 3	0,0010	0,0016	0,0010	0,0017	0,0016	0,0027	0,0013	0,0023	0,0014	0,0023	0,0013	0,0021	0,0012	0,0020	0,0010	0,0013	0,0012	0,0014
20	Position 1	0,0012	0,0015	0,0007	0,0007	0,0011	0,0020	0,0009	0,0016	0,0007	0,0008	0,0007	0,0013	0,0008	0,0011	0,0002	0,0008	0,0012	0,0017
21	Position 3	0,0009	0,0009	0,0022	0,0020	0,0026	0,0019	0,0023	0,0018	0,0026	0,0019	0,0023	0,0019	0,0024	0,0016	0,0022	0,0011	0,0017	0,0008
22	Position 1	0,0012	0,0016	0,0008	0,0008	0,0010	0,0018	0,0008	0,0008	0,0007	0,0007	0,0008	0,0011	0,0009	0,0027	0,0006	0,0022	0,0011	0,0014
23	Position 2	0,0015	0,0026	0,0011	0,0015	0,0017	0,0024	0,0012	0,0016	0,0016	0,0019	0,0011	0,0014	0,0007	0,0014	0,0007	0,0010	0,0013	0,0029
24	Position 1	0,0014	0,0006	0,0004	0,0011	0,0013	0,0016	0,0005	0,0012	0,0007	0,0013	0,0006	0,0014	0,0012	0,0015	0,0010	0,0013	0,0016	0,0010
25	Position 3	0,0016	0,0014	0,0009	0,0022	0,0018	0,0028	0,0012	0,0027	0,0012	0,0025	0,0012	0,0025	0,0011	0,0019	0,0009	0,0014	0,0013	0,0024
26	Position 3	0,0026	0,0035	0,0005	0,0021	0,0027	0,0031	0,0011	0,0024	0,0027	0,0031	0,0015	0,0019	0,0009	0,0020	0,0007	0,0027	0,0023	0,0022
27	Position 3	0,0029	0,0023	0,0010	0,0016	0,0030	0,0029	0,0018	0,0025	0,0029	0,0028	0,0018	0,0021	0,0014	0,0017	0,0012	0,0016	0,0023	0,0024
28	Position 3	0,0018	0,0006	0,0015	0,0013	0,0021	0,0013	0,0016	0,0013	0,0019	0,0013	0,0016	0,0013	0,0010	0,0014	0,0013	0,0007	0,0014	0,0010
29	Position 1	0,0024	0,0007	0,0004	0,0007	0,0024	0,0012	0,0006	0,0007	0,0024	0,0012	0,0011	0,0009	0,0026	0,0011	0,0024	0,0008	0,0025	0,0007
30	Position 3	0,0027	0,0014	0,0015	0,0031	0,0026	0,0032	0,0015	0,0031	0,0023	0,0032	0,0015	0,0030	0,0013	0,0026	0,0007	0,0028	0,0025	0,0030
31	Position 3	0,0039	0,0028	0,0010	0,0021	0,0045	0,0046	0,0025	0,0034	0,0019	0,0030	0,0026	0,0033	0,0029	0,0035	0,0021	0,0027	0,0027	0,0025
32	Position 3	0,0025	0,0027	0,0025	0,0023	0,0034	0,0036	0,0030	0,0028	0,0034	0,0033	0,0026	0,0026	0,0028	0,0026	0,0011	0,0032	0,0028	0,0021
33	Position 1	0,0033	0,0021	0,0004	0,0007	0,0031	0,0022	0,0006	0,0008	0,0014	0,0014	0,0013	0,0011	0,0019	0,0014	0,0020	0,0015	0,0018	0,0017
34	Position 3	0,0023	0,0014	0,0014	0,0020	0,0027	0,0031	0,0017	0,0026	0,0024	0,0030	0,0018	0,0026	0,0015	0,0030	0,0022	0,0043	0,0020	0,0035
35	Position 1	0,0034	0,0018	0,0007	0,0008	0,0036	0,0018	0,0015	0,0011	0,0018	0,0013	0,0020	0,0011	0,0044	0,0021	0,0043	0,0024	0,0032	0,0019
36	Position 2	0,0018	0,0021	0,0019	0,0012	0,0026	0,0026	0,0020	0,0017	0,0024	0,0023	0,0020	0,0017	0,0013	0,0022	0,0007	0,0022	0,0019	0,0018
37	Position 1	0,0016	0,0004	0,0007	0,0014	0,0014	0,0017	0,0008	0,0015	0,0014	0,0017	0,0007	0,0015	0,0018	0,0015	0,0011	0,0013	0,0014	0,0011
38	Position 1	0,0029	0,0019	0,0009	0,0008	0,0029	0,0021	0,0015	0,0012	0,0015	0,0012	0,0016	0,0013	0,0032	0,0022	0,0027	0,0020	0,0030	0,0022
39	Position 2	0,0022	0,0029	0,0008	0,0016	0,0025	0,0017	0,0010	0,0015	0,0018	0,0017	0,0015	0,0007	0,0019	0,0012	0,0023	0,0010	0,0012	0,0026
40	Position 1	0,0020	0,0013	0,0011	0,0004	0,0023	0,0014	0,0014	0,0005	0,0023	0,0014	0,0014	0,0008	0,0014	0,0011	0,0010	0,0009	0,0025	0,0006
41	Position 1	0,0022	0,0023	0,0002	0,0005	0,0021	0,0020	0,0006	0,0007	0,0019	0,0019	0,0010	0,0009	0,0014	0,0011	0,0014	0,0012	0,0013	0,0017
42	Position 2	0,0013	0,0019	0,0009	0,0007	0,0018	0,0020	0,0011	0,0013	0,0009	0,0007	0,0013	0,0012	0,0020	0,0012	0,0024	0,0013	0,0015	0,0020
43	Position 1	0,0008	0,0028	0,0003	0,0006	0,0008	0,0024	0,0004	0,0010	0,0004	0,0010	0,0005	0,0010	0,0004	0,0029	0,0003	0,0025	0,0008	0,0019
44	Position 1	0,0022	0,0017	0,0005	0,0006	0,0022	0,0021	0,0005	0,0006	0,0012	0,0015	0,0010	0,0012	0,0021	0,0012	0,0024	0,0010	0,0013	0,0018
Mittelwert		0,0019	0,0023	0,0011	0,0018	0,0022	0,0031	0,0014	0,0021	0,0017	0,0023	0,0014	0,0022	0,0017	0,0026	0,0015	0,0024	0,0019	0,0024

																				Mittelwert
besser						4	8	8	10	8	8	7	14	9	13	15	20	4	11	9,93
schlechter						40	36	36	34	35	35	37	30	35	31	29	24	40	33	33,93
gleich						0	0	0	0	1	1	0	0	0	0	0	0	0	0	0,14

Tabelle 6.3: Übersicht absolute mittlere Abweichung aus 'Übersicht_Maßzahlen.xls'

Messreihe	Ort	S X	S Y	M X	M Y	F1 X	F1 Y	F2 X	F2 Y	F3 X	F3 Y	F4 X	F4 Y	F5 X	F5 Y	F6 X	F6 Y	F7 X	F7 Y
1	Position 1	0,0020	0,0019	0,0021	0,0019	0,0021	0,0017	0,0020	0,0018	0,0021	0,0017	0,0019	0,0015	0,0027	0,0011	0,0026	0,0010	0,0035	0,0024
2	Position 3	0,0012	0,0010	0,0011	0,0023	0,0015	0,0022	0,0011	0,0023	0,0010	0,0024	0,0012	0,0022	0,0012	0,0024	0,0004	0,0017	0,0012	0,0023
3	Position 2	0,0044	0,0033	0,0009	0,0012	0,0048	0,0035	0,0043	0,0031	0,0033	0,0024	0,0028	0,0021	0,0032	0,0023	0,0031	0,0019	0,0035	0,0029
4	Position 1	0,0027	0,0024	0,0006	0,0007	0,0028	0,0027	0,0007	0,0008	0,0017	0,0023	0,0014	0,0015	0,0013	0,0009	0,0016	0,0007	0,0022	0,0024
5	Position 3	0,0032	0,0024	0,0023	0,0025	0,0045	0,0023	0,0037	0,0023	0,0045	0,0023	0,0033	0,0020	0,0035	0,0016	0,0038	0,0017	0,0039	0,0036
6	Position 2	0,0016	0,0027	0,0016	0,0023	0,0018	0,0025	0,0016	0,0022	0,0015	0,0022	0,0015	0,0020	0,0016	0,0014	0,0017	0,0013	0,0026	0,0021
7	Position 2	0,0022	0,0036	0,0010	0,0013	0,0023	0,0037	0,0011	0,0016	0,0019	0,0020	0,0011	0,0020	0,0014	0,0027	0,0016	0,0023	0,0018	0,0040
8	Position 1	0,0020	0,0019	0,0021	0,0019	0,0021	0,0017	0,0020	0,0018	0,0021	0,0017	0,0019	0,0015	0,0027	0,0011	0,0026	0,0010	0,0035	0,0024
9	Position 1	0,0020	0,0017	0,0012	0,0011	0,0028	0,0014	0,0015	0,0012	0,0024	0,0014	0,0019	0,0009	0,0020	0,0008	0,0016	0,0005	0,0022	0,0018
10	Position 1	0,0026	0,0024	0,0005	0,0008	0,0026	0,0024	0,0014	0,0016	0,0020	0,0023	0,0014	0,0013	0,0026	0,0010	0,0028	0,0008	0,0020	0,0021
11	Position 2	0,0031	0,0027	0,0019	0,0024	0,0025	0,0034	0,0019	0,0025	0,0020	0,0025	0,0016	0,0026	0,0008	0,0034	0,0005	0,0030	0,0028	0,0039
12	Position 2	0,0007	0,0193	0,0027	0,0211	0,0024	0,0403	0,0026	0,0228	0,0026	0,0260	0,0025	0,0296	0,0021	0,0401	0,0017	0,0394	0,0022	0,0201
13	Position 3	0,0024	0,0016	0,0016	0,0021	0,0030	0,0019	0,0019	0,0021	0,0027	0,0024	0,0020	0,0019	0,0028	0,0017	0,0025	0,0024	0,0023	0,0015
14	Position 1	0,0012	0,0022	0,0010	0,0010	0,0014	0,0020	0,0013	0,0017	0,0010	0,0011	0,0011	0,0012	0,0009	0,0007	0,0005	0,0004	0,0012	0,0022
15	Position 3	0,0019	0,0022	0,0022	0,0023	0,0025	0,0027	0,0022	0,0022	0,0023	0,0026	0,0021	0,0022	0,0024	0,0027	0,0017	0,0010	0,0021	0,0017
16	Position 1	0,0029	0,0038	0,0008	0,0007	0,0026	0,0038	0,0025	0,0030	0,0021	0,0035	0,0014	0,0021	0,0015	0,0024	0,0015	0,0021	0,0023	0,0035
17	Position 3	0,0012	0,0049	0,0013	0,0027	0,0017	0,0058	0,0015	0,0028	0,0015	0,0028	0,0014	0,0037	0,0014	0,0059	0,0008	0,0051	0,0011	0,0051
18	Position 2	0,0026	0,0027	0,0009	0,0007	0,0026	0,0025	0,0012	0,0014	0,0021	0,0021	0,0014	0,0012	0,0012	0,0012	0,0008	0,0014	0,0023	0,0021
19	Position 3	0,0012	0,0021	0,0013	0,0020	0,0020	0,0034	0,0017	0,0028	0,0017	0,0029	0,0016	0,0026	0,0015	0,0026	0,0012	0,0014	0,0015	0,0017
20	Position 1	0,0014	0,0024	0,0008	0,0009	0,0014	0,0028	0,0011	0,0025	0,0009	0,0011	0,0009	0,0017	0,0009	0,0013	0,0003	0,0008	0,0014	0,0023
21	Position 3	0,0011	0,0010	0,0028	0,0023	0,0033	0,0023	0,0029	0,0021	0,0033	0,0023	0,0030	0,0022	0,0030	0,0020	0,0024	0,0014	0,0020	0,0010
22	Position 1	0,0015	0,0019	0,0011	0,0011	0,0013	0,0021	0,0011	0,0011	0,0009	0,0010	0,0010	0,0013	0,0011	0,0028	0,0007	0,0023	0,0012	0,0018
23	Position 2	0,0021	0,0032	0,0013	0,0016	0,0020	0,0030	0,0014	0,0020	0,0019	0,0024	0,0013	0,0018	0,0009	0,0016	0,0010	0,0011	0,0018	0,0036
24	Position 1	0,0020	0,0009	0,0006	0,0014	0,0018	0,0019	0,0008	0,0015	0,0011	0,0015	0,0009	0,0016	0,0014	0,0018	0,0013	0,0013	0,0021	0,0011
25	Position 3	0,0022	0,0025	0,0011	0,0027	0,0024	0,0040	0,0018	0,0037	0,0017	0,0035	0,0016	0,0032	0,0015	0,0025	0,0012	0,0015	0,0018	0,0031
26	Position 3	0,0034	0,0040	0,0007	0,0024	0,0036	0,0039	0,0016	0,0030	0,0036	0,0039	0,0020	0,0026	0,0012	0,0030	0,0009	0,0028	0,0030	0,0028
27	Position 3	0,0039	0,0029	0,0014	0,0021	0,0041	0,0036	0,0026	0,0031	0,0040	0,0034	0,0024	0,0026	0,0019	0,0022	0,0014	0,0017	0,0034	0,0028
28	Position 3	0,0024	0,0008	0,0019	0,0016	0,0028	0,0017	0,0019	0,0016	0,0024	0,0016	0,0020	0,0016	0,0013	0,0017	0,0016	0,0008	0,0019	0,0012
29	Position 1	0,0031	0,0010	0,0006	0,0008	0,0030	0,0015	0,0008	0,0008	0,0030	0,0015	0,0013	0,0011	0,0028	0,0012	0,0026	0,0008	0,0029	0,0008
30	Position 3	0,0036	0,0018	0,0019	0,0040	0,0037	0,0043	0,0020	0,0041	0,0034	0,0043	0,0022	0,0041	0,0017	0,0035	0,0008	0,0030	0,0034	0,0034
31	Position 3	0,0047	0,0038	0,0013	0,0029	0,0056	0,0057	0,0038	0,0047	0,0028	0,0044	0,0033	0,0042	0,0034	0,0041	0,0026	0,0031	0,0036	0,0033
32	Position 3	0,0032	0,0036	0,0033	0,0037	0,0044	0,0054	0,0040	0,0045	0,0043	0,0052	0,0036	0,0042	0,0039	0,0041	0,0012	0,0033	0,0034	0,0026
33	Position 1	0,0041	0,0028	0,0005	0,0008	0,0039	0,0028	0,0008	0,0010	0,0020	0,0017	0,0017	0,0014	0,0021	0,0017	0,0021	0,0015	0,0025	0,0023
34	Position 3	0,0030	0,0021	0,0016	0,0026	0,0035	0,0039	0,0022	0,0036	0,0032	0,0039	0,0022	0,0032	0,0018	0,0039	0,0023	0,0047	0,0024	0,0040
35	Position 1	0,0045	0,0024	0,0010	0,0010	0,0046	0,0024	0,0026	0,0017	0,0031	0,0019	0,0025	0,0015	0,0050	0,0028	0,0047	0,0026	0,0040	0,0026
36	Position 2	0,0023	0,0025	0,0024	0,0014	0,0033	0,0032	0,0026	0,0022	0,0031	0,0029	0,0026	0,0021	0,0020	0,0024	0,0009	0,0023	0,0022	0,0022
37	Position 1	0,0021	0,0005	0,0009	0,0015	0,0018	0,0018	0,0011	0,0016	0,0018	0,0018	0,0009	0,0016	0,0019	0,0016	0,0013	0,0014	0,0017	0,0012
38	Position 1	0,0040	0,0025	0,0013	0,0011	0,0039	0,0028	0,0024	0,0019	0,0024	0,0019	0,0022	0,0017	0,0036	0,0027	0,0029	0,0023	0,0039	0,0027
39	Position 2	0,0026	0,0031	0,0011	0,0018	0,0028	0,0019	0,0014	0,0017	0,0023	0,0019	0,0016	0,0010	0,0021	0,0016	0,0023	0,0014	0,0014	0,0030
40	Position 1	0,0026	0,0015	0,0017	0,0005	0,0028	0,0016	0,0020	0,0007	0,0028	0,0016	0,0019	0,0009	0,0019	0,0012	0,0015	0,0010	0,0032	0,0008
41	Position 1	0,0027	0,0028	0,0003	0,0007	0,0026	0,0025	0,0011	0,0011	0,0025	0,0024	0,0013	0,0012	0,0014	0,0013	0,0014	0,0013	0,0017	0,0022
42	Position 2	0,0016	0,0025	0,0010	0,0008	0,0022	0,0026	0,0013	0,0020	0,0010	0,0008	0,0015	0,0015	0,0022	0,0013	0,0025	0,0014	0,0019	0,0026
43	Position 1	0,0010	0,0034	0,0004	0,0008	0,0010	0,0029	0,0006	0,0016	0,0006	0,0017	0,0006	0,0013	0,0004	0,0031	0,0003	0,0026	0,0010	0,0023
44	Position 1	0,0027	0,0023	0,0005	0,0007	0,0028	0,0028	0,0005	0,0007	0,0018	0,0024	0,0013	0,0015	0,0022	0,0014	0,0025	0,0012	0,0016	0,0023
	Mittelwert	0,0025	0,0028	0,0013	0,0021	0,0028	0,0037	0,0018	0,0026	0,0023	0,0029	0,0018	0,0026	0,0020	0,0030	0,0017	0,0026	0,0024	0,0028

																				Mittelwert
besser						1	5	7	10	4	6	7	11	8	12	19	22	6	12	9,29
schlechter						43	39	37	34	39	37	37	33	36	32	25	22	38	32	34,57
gleich						0	0	0	0	1	1	0	0	0	0	0	0	0	0	0,14

Tabelle 6.4: Übersicht Standardabweichungen aus 'Übersicht_Maßzahlen.xls'

Durchschnittlich sind von 44 lediglich 9,29 Messreihen durch die Korrektur mittels einer der verschiedenen Formeln besser geworden. Wie auch bei der mittleren absoluten Abweichung sind einige Messreihen dabei, welche eine Verbesserung beider Koordinatenwerte durch fast alle Formeln aufweisen (Messreihe 1, 6, 8). Es sind ebenso Messreihen dabei, die nur eine Verbesserung der x- oder y-Koordinate durch die verschiedenen Formeln zeigen (Messreihe 5 und 13). Eine konstante Steigerung der Genauigkeit der Positionsbestimmung, gemessen an der Standardabweichung, ist aber nicht zu erkennen.

Der erste Ansatz zur Verbesserung (vgl. Abschnitt 6.2.1, Formel 1) weist die schlechtesten Ergebnisse auf. Mit 43 Verschlechterungen bei der x-Koordinate und 39 Verschlechterungen bei der y-Koordinate bei 44 Messreihen, ist dieser Ansatz auf Grund der statistischen Maßzahl der Standardabweichung völlig unbrauchbar. Dies drückt sich auch im Mittelwert der Standardabweichungen aller 44 Messreihen aus. Dieser fällt von 0,0013' bei der x-Koordinate der originalen Daten auf 0,0028' ab und bei der y-Koordinate von 0,0021' auf 0,0037'. Dies sind auch die schlechtesten Mittelwerte aller sieben Formeln.

Mit 19 Verbesserungen bei der x-Koordinate und 22 bei der y-Koordinate weist Ansatz 6 die besten Ergebnisse auf. Aber auch hier kann man nicht von einer konstanten Steigerung der Genauigkeit sprechen. Die Hälfte aller Messreihen wurden bei der Neuberechnung der Koordinaten durch die Formeln 6.11 und 6.12 schlechter. Der Mittelwert der Standardabweichungen aller Messreihen verschlechterte sich um 0,0013' auf 0,0017' bei der x-Koordinate, was aber immer noch den besten Wert aller sieben Formeln darstellt. Bei der y-Koordinate ist der Mittelwert, trotz der maximalen Anzahl an Verbesserungen, nicht der Beste. Hier liefert Formel 2 mit 0,002604' gegenüber 0,002644' von Formel 6 das bessere Resultat[11]. Trotz des besten Ergebnisses aller Ansätze, genügt auch Ansatz 6 nicht den Ansprüchen einer kontinuierlichen Verbesserung. Hinzu kommen bei diesem Ansatz die Probleme, die schon in Kapitel 6.2.7 erwähnt wurden.

6.3.2 Korrelation der Messreihen

Die im letzten Abschnitt betrachteten Maßzahlen lassen darauf schließen, dass die angestrebte Verbesserung der Positionsgenauigkeit durch die beiden GPS-Module nicht erreicht werden kann. Um eine solche Verbesserung zu erzielen, müssen die gemessenen Werte der Referenzstation bzw. deren Abweichungen vom tatsächlichen Standort in Beziehung zu den Messfehlern der mobilen Ein-

[11] nicht in Tabelle 6.4 zu erkennen, da hier auf vier Nachkommastellen gerundet wurde, vgl. hierzu Datei 'Übersicht_Maßzahlen'

heit stehen. Der Zusammenhang der Abweichungen der Referenzstation und der Abweichungen der mobilen Einheit von den eigentlichen Koordinaten sollte annährend linear sein. In der Statistik wird der Korrelationskoeffizient, oder auch Maßkorrelationskoeffizient nach *Bravais* und *Pearson*, als normiertes und symmetrisches Zusammenhangsmaß für die lineare Abhängigkeit von bivariaten Datenreihen verwendet[12]:

$$r_{xy} = r_{yx} = \frac{\frac{1}{n} * \sum_{i=1}^{n} (x_i - \bar{x}) * (y_i - \bar{y})}{\sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n} * \frac{\sum_{i=1}^{n} (y_i - \bar{y})^2}{n}}} \tag{6.15}$$

Hierbei stehen $\bar{x}$ und $\bar{y}$ für die Mittelwerte der jeweiligen Datenreihen. Der Zähler des Bruches entspricht somit der Kovarianz der beiden Datenreihen. Die Wurzel im Nenner besteht aus dem Produkt der Varianz beider Reihen. Spaltet man die Wurzel auf, so ergibt sich ein Produkt aus den Wurzeln der Varianzen, was dem Produkt der Standardabweichungen $\sigma(x)$ bzw. $\sigma(y)$ entspricht[13]. Daraus resultierend kann man die Formel aus 6.15 folgendermaßen verkürzt darstellen:

$$r_{xy} = r_{yx} = \frac{Cov(x, y)}{\sigma(x) * \sigma(y)} \tag{6.16}$$

Der Korrelationskoeffizient liegt im Bereich von $-1 \leq r_{xy} = r_{yx} \leq 1$. Bei Werten für r nahe $+1$ spricht man von einem gleichläufigen oder auch positiven linearen Zusammenhang. Bei Werten um -1 wird von einem gegenläufigen oder auch negativen linearen Zusammenhang gesprochen. Je größer der Betrag von r ist, um so wahrscheinlicher lässt sich eine stärkere lineare Beziehung der beiden Datenreihen vermuten. Hingegen ist bei einem Wert um Null von keiner solchen Abhängigkeit auszugehen. Um eine kontinuierliche Verbesserung der Positionsgenauigkeit zu erreichen, sollte der Zusammenhang zwischen der Abweichung der Referenzstation und der Abweichungen der mobilen Einheit positiv linear sein. Dies würde bedeuten, dass der Korrelationskoeffizient aus Formel 6.16 der Datenreihen sich in der Nähe von $+1$ befinden sollte.

In Tabelle 6.5[14] sind die einzelnen Korrelationskoeffizienten aller Messreihen für jeweils die x- und die y-Koordinate zusammengefasst. Am Ende der Tabelle befinden sich die beiden dazugehörigen Mittelwerte. Diese liegen mit 0,006 bei der x-Koordinate und mit 0,013 bei der y-Koordinate sehr nahe bei Null. Somit

[12] vgl. [Eck99]:74
[13] vgl. hierzu Anhang A.5
[14] auch in Datei 'Übersicht_Maßzahlen' Tabellenblatt 'Korrelationskoeffizient' zu finden

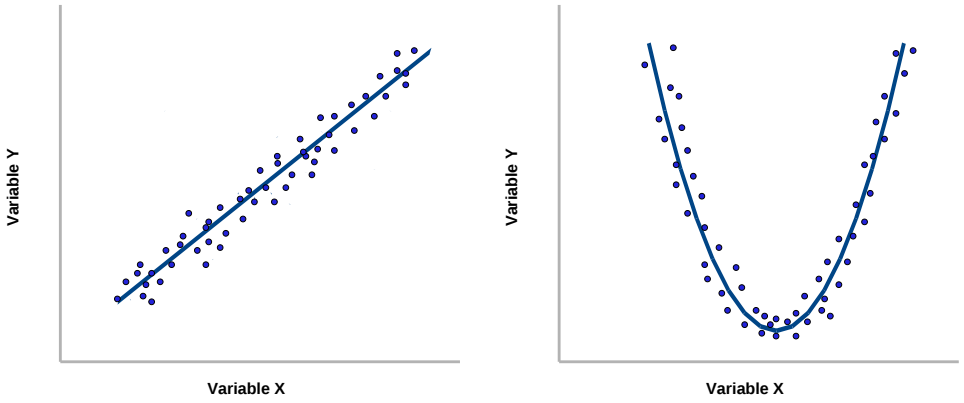
Messreihe	Ort	x	y
1	Position 1	-0,041043	0,036692
2	Position 3	0,169227	-0,014445
3	Position 2	0,158203	-0,168533
4	Position 1	0,046311	-0,298687
5	Position 3	-0,311541	-0,062251
6	Position 2	-0,038443	-0,006719
7	Position 2	-0,191779	0,170772
8	Position 1	-0,041043	0,036692
9	Position 1	-0,530117	0,084266
10	Position 1	-0,213841	0,165522
11	Position 2	-0,044717	0,079479
12	Position 2	0,051920	0,000834
13	Position 3	-0,219545	-0,065146
14	Position 1	0,147752	0,145869
15	Position 3	0,414005	0,155910
16	Position 1	0,034659	-0,121656
17	Position 3	-0,020311	-0,023698
18	Position 2	0,081249	0,126919
19	Position 3	-0,242571	-0,165856
20	Position 1	0,290712	-0,347806
21	Position 3	-0,149377	-0,187496
22	Position 1	0,128818	0,375063
23	Position 2	-0,063822	0,106150
24	Position 1	0,294446	0,038169
25	Position 3	-0,019917	-0,345614
26	Position 3	-0,135798	0,119961
27	Position 3	-0,078070	0,185052
28	Position 3	-0,057315	0,167080
29	Position 1	0,300087	-0,048662
30	Position 3	0,107696	0,021129
31	Position 3	-0,324798	-0,216963
32	Position 3	0,021550	-0,140521
33	Position 1	-0,113115	0,258232
34	Position 3	-0,232370	-0,209967
35	Position 1	0,375886	0,303553
36	Position 2	-0,001874	0,123351
37	Position 1	0,428263	0,023815
38	Position 1	0,259583	0,023866
39	Position 2	0,122725	0,002978
40	Position 1	0,147614	0,150858
41	Position 1	-0,118993	0,192847
42	Position 2	0,090064	0,159472
43	Position 1	-0,024995	0,155125
44	Position 1	-0,167842	-0,413249
	Mittelwert	**0,006535**	**0,013009**

Tabelle 6.5: Übersicht Korrelationskoeffizienten aus 'Übersicht_Maßzahlen.xls'

ist kein linearer Zusammenhang zu vermuten. Auch die einzelnen Korrelationskoeffizienten bestätigen dies. Manchmal befindet sich ein betragsmäßig größerer Wert über der Null, in anderen Fällen unter der Null. Dies würde einmal auf eine gleichläufige, und das andere Mal auf eine gegenläufige Abhängigkeit deuten. Insgesamt treten wenige Werte auf, die betragsmäßig größer 0,3 sind und erst ab diesem Wert wird im allgemeinen Beurteilungen von einem schwachen Zusammenhang gesprochen.

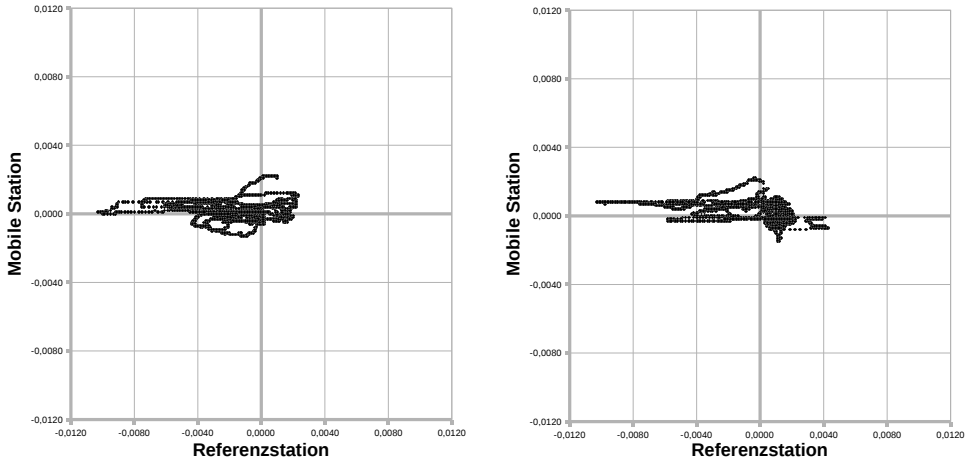
Aus der Betrachtung der Korrelationskoeffizienten kann man somit schließen, dass es mit einer sehr hohen Wahrscheinlichkeit keinen linearen Zusammenhang zwischen den Abweichungen der Referenzstation und denen der mobilen Station

gibt. Neben linearen gibt es jedoch noch andere Abhängigkeiten, wie zum Beispiel über eine logarithmische, Exponential- oder Potenzfunktion. Streuungsdiagramme sind ein guter Indikator solcher Abhängigkeiten. Diese nicht-linearen Zusammenhänge der Messreihen sind zwar nicht zu vermuten, um dies aber auszuschließen, sind in den Abbildungen 6.17 bis 6.20 einige Streudiagramme abgebildet. Keine der Punktewolken weist eine regelmäßige Form auf und somit ist auch eine nicht-lineare Abhängigkeit nicht naheliegend.



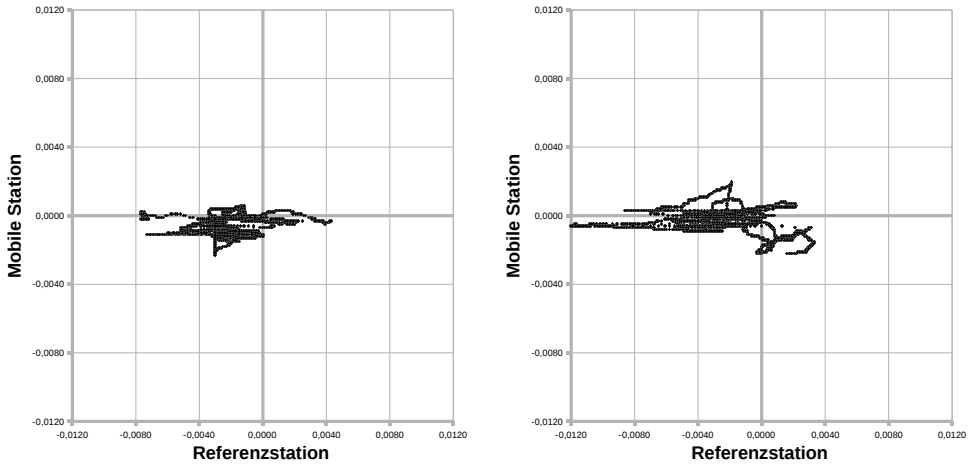
(a) linearer Zusammenhang (b) quadratischer Zusammenhang

Abbildung 6.16: exemplarische Streuungsdiagramme



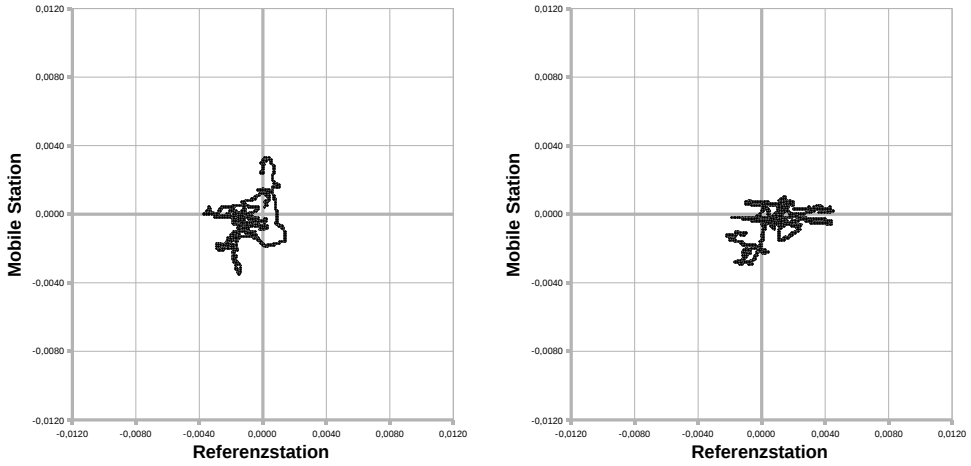
(a) x-Koordinate (b) y-Koordinate

Abbildung 6.17: Streuungsdiagramme der Messreihe 4



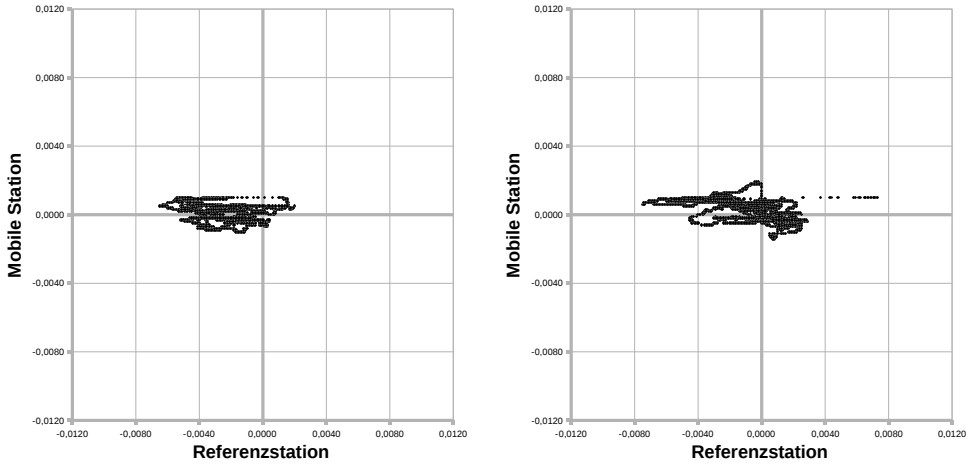
(a) x-Koordinate (b) y-Koordinate

Abbildung 6.18: Streuungsdiagramme der Messreihe 16



(a) x-Koordinate (b) y-Koordinate

Abbildung 6.19: Streuungsdiagramme der Messreihe 22



(a) x-Koordinate (b) y-Koordinate

Abbildung 6.20: Streuungsdiagramme der Messreihe 44

6.4 Bewertung der Ergebnisse

Nach Auswertung und Betrachtung der statistischen Maßzahlen aller Messreihen und den Ergebnissen der Korrelationskoeffizienten kann man festhalten, dass auf Grund des fehlenden Zusammenhangs zwischen den Messergebnissen der Referenzstation und der mobilen Einheit eine Verbesserung der Genauigkeit bei der Positionsbestimmung in dieser Projektkonfiguration grundsätzlich nicht möglich war. Dies wird durch eine abschließende Messreihe noch einmal deutlich belegt. Hierbei befinden sich die beiden GPS-Module am gleichen Ort (Standpunkt der Referenzstation der vorherigen Messungen), der Zeitraum umfasst ca. 76 Stunden und lieferte somit über 274000 Wertepaare. In Tabelle 6.6 sind einige statistische Maßzahlen zu dieser Datenreihe aufgelistet.

Die Abbildungen 6.21 und 6.22 zeigen die Streuungsdiagramme der Breiten- (x-Werte) und der Längenkoordinaten (y-Werte) der Messung. Wie auch in den vorherigen Diagrammen ist keine systematische Verteilung zu erkennen. Die Messreihen der beiden GPS-Module korrelieren nicht und somit ist auch keine systematische Fehlerkorrektur möglich. Die Problematik bei der Realisierung liegt in der Ungenauigkeit bzw. in dem nicht korrelierenden Verlauf der Messungen beider GPS-Empfänger. Trotz ihrer Baugleichheit scheinen auf Grund von Toleranzen der Komponenten die Messergebnisse zu unterschiedlich zu sein. Die Probleme entstehen durch die hohe Empfindlichkeit des Systems. Eine Abweichung von 1ns zum Beispiel bedeutet eine Änderung der Position um 0,00016'. Trotz der beiden relativ teuren und guten GPS-Module *HI-204III* von *Haicom* ist somit keine permanente Verbesserung bei der Positionsbestimmung möglich.

Maßzahl	Modul A	Modul B
Arithmetischer Mittelwert		
Breitenkoordinate	-0,000631	0,000380
Längenkoordinate	0,003778	0,000998
Standardabweichung		
Breitenkoordinate	0,002728	0,002864
Längenkoordinate	0,003778	0,003702
Korrelationskoeffizient		
Breitenkoordinate	0,000850	
Längenkoordinate	0,001274	

Tabelle 6.6: statistische Maßzahlen der abschließenden Messreihe

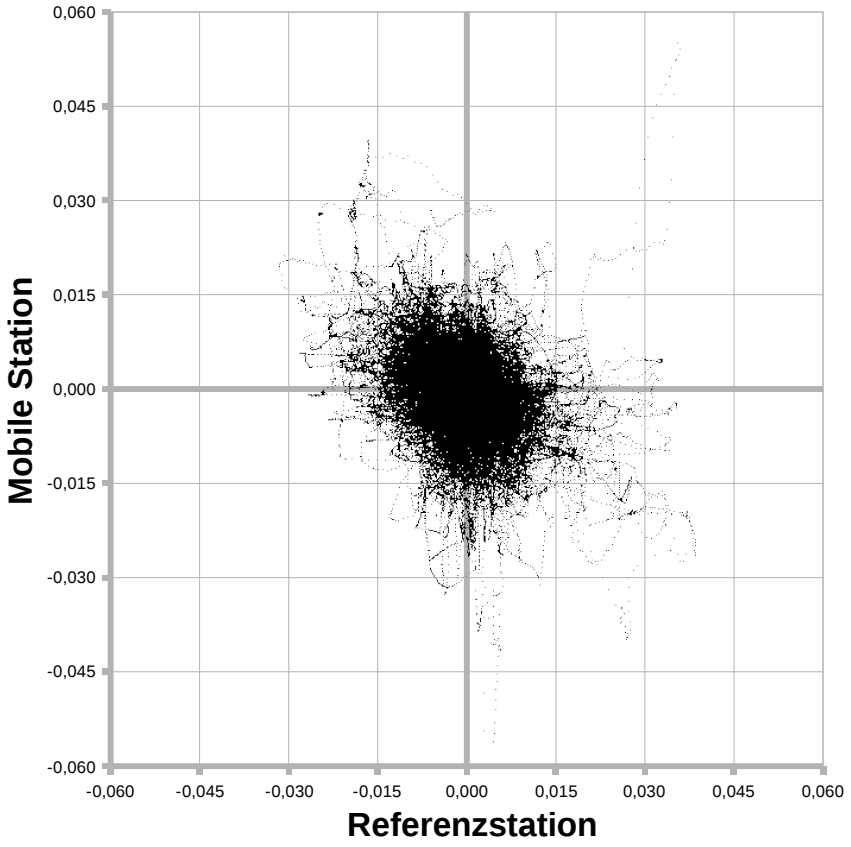


Abbildung 6.21: Streuungsdiagramm der x-Werte der abschließenden Messreihe

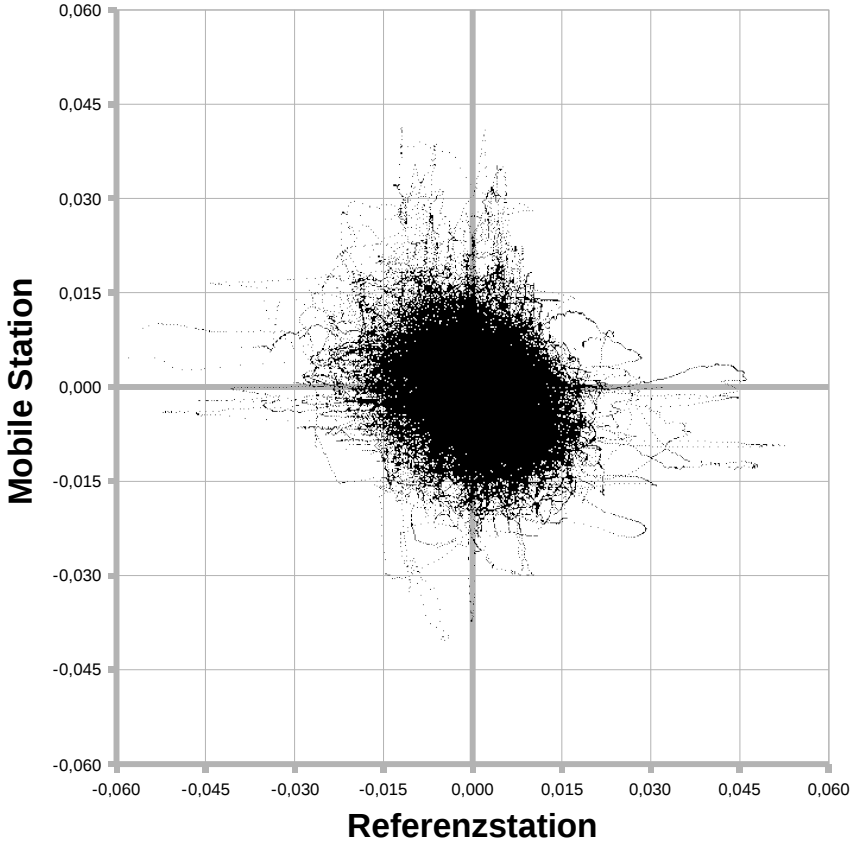


Abbildung 6.22: Streuungsdiagramm der y-Werte der abschließenden Messreihe

7 Fazit

Die ersten Messreihen, welche noch mit alten GPS-Modulen durchgeführt wurden, waren im Bezug auf eine Steigerung der Genauigkeit bei der Positionsbestimmung sehr vielversprechend. Eine Verbesserung wäre zwar möglich gewesen, aber die Module an sich waren mit bis zu 100 Metern Abweichung zu unpräzise für den Einsatz in der lokalen Navigation eines autonomen Systems. Die im endgültigen Versuchsaufbau verwendeten GPS-Module wiesen auch ohne den Versuch der Korrektur eine wesentlich höhere Genauigkeit auf.

Leider führte eine direkte Verrechnung der Korrekturdaten in den ersten Messreihen mit den letztendlich eingesetzten GPS-Modulen nicht zu einer Verbesserung bei der Positionsbestimmung. In drei weiteren Ansätzen wurde der von den Modulen gelieferte HDOP-Wert mit einbezogen und floss mit Hilfe unterschiedlicher Formeln in die Korrektur mit ein. Da auch die hierzu durchgeführten Messreihen nicht die gewünschten Ergebnisse brachten, wurde versucht mit Hilfe der Software statistische Mittelwerte zu berechnen und auf deren Basis weitere drei Ansätze zu entwickeln.

Die Weiterentwicklung der Mikrocontroller-Software mit verschieden Algorithmen zur Berechnung der Korrekturwerte brachten keine konstante Verbesserung der Ergebnisse. Die Positionsgenauigkeit war zwar immer noch höher als die mit den alten Modulen, eine stetige Steigerung der Exaktheit mit Hilfe des Mikrocontrollers war aber nicht zu erzielen. Der Grund hierfür liegt bei einem zu unterschiedlichen Verlauf der Messwerte beider Module, die nicht annährend korrelieren und somit eine stetige Verbesserung nicht möglich war. Die Ursache ist bei einem so hochempfindlichen System wie GPS, trotz der Baugleichheit der Module, der zu große Toleranzbereich der verbauten Komponenten.

Das in dieser Arbeit entwickelte System kann aber dennoch zu einer Steigerung der Positionsgenauigkeit mittels GPS eingesetzt werden. Da die Schnittstellen zum Anbinden von verschiedenen Komponenten allgemein gehalten wurden, ist der Anschluss anderer Module leicht möglich. So könnten höherwertige GPS-Module, die untereinander „stimmiger“ sind, zum Einsatz kommen.

Insgesamt ist die entwickelte Software derart programmiert, dass sie auch leicht in anderen Anwendungen genutzt werden kann. Die Implementation der Steuerung der Funkmodule *RFM12* ist komplett in den beiden Dateien *RFM12.h* und *RFM12.c* enthalten. Die RFM12-Software kann sehr einfach zum Aufbau einer bidirektionalen Funkverbindung verwendet werden. Diese könnte zum Beispiel zum Senden von Steuerungskommandos an einen Roboter sowie zum Empfang von Status und Messdaten eines solchen eingesetzt werden.

Trotz einiger Probleme bei der Umsetzung des Projektes und den leider nicht zufriedenstellenden Ergebnissen war die Erstellung dieser Arbeit interessant und hat das Interesse für weitere Entwicklungen im Bereich der Microkontroller geweckt. Abschließend möchte ich mich bei Herrn Prof. Dr. Christoph Steigner und Dr. Merten Joost für die Betreuung dieser Arbeit bedanken.

A Anhang

A.1 Karten

Abbildung A.1: Lageplan der vier Messpunkte im Versuchsaufbau

Position	Abweichung von Norden	Entfernung zur Referenzstation
Position 1	$\approx 54°$	≈16,18m
Position 2	$\approx 88°$	≈21,92m
Position 3	$\approx 154°$	≈10,34m

Tabelle A.1: Position der drei Messpunkte in Bezug auf die Referenzstation

Abbildung A.2: Verlauf der Messreihe 1 der Referenzstation

A.2 Aufbau einer GPS-Nachricht

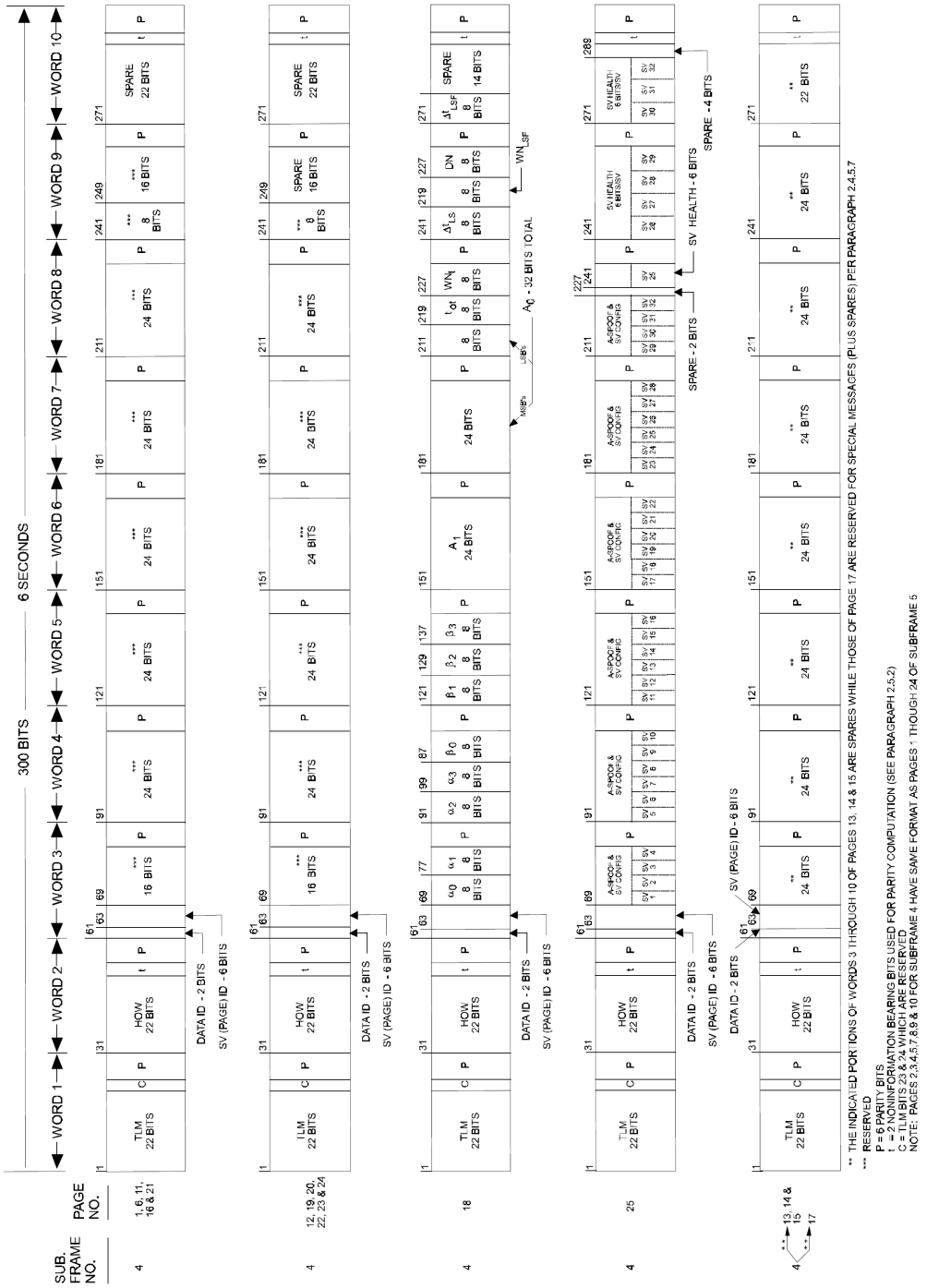


Abbildung A.3: GPS-Nachricht Bild 1 von 2

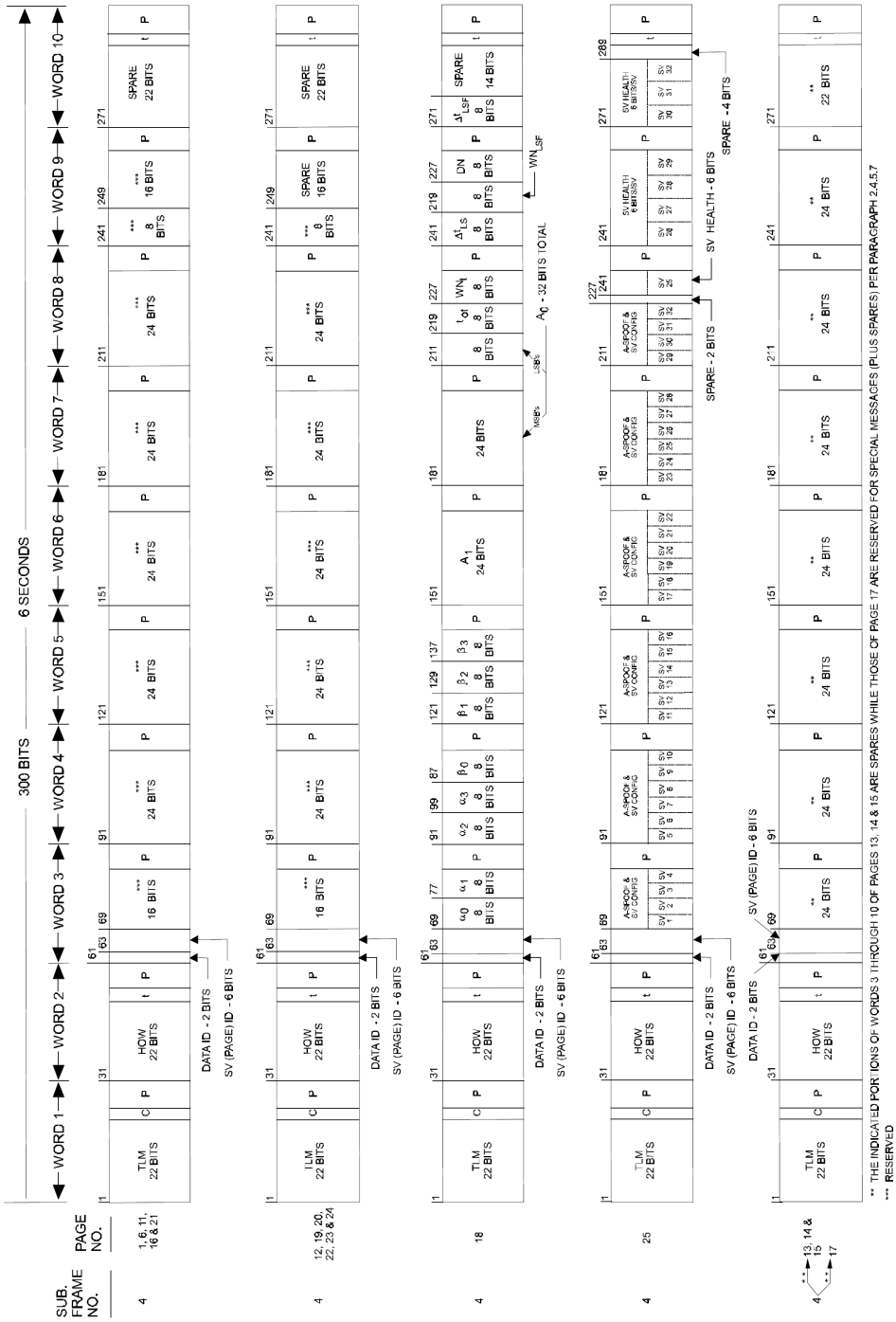


Abbildung A.4: GPS-Nachricht Bild 2 von 2

A.3 ASCII-Tabelle

Dez	Hex	Okt	Zeichen	Dez	Hex	Okt	Zeichen
0	0x00	000	NUL	32	0x20	040	SP
1	0x01	001	SOH	33	0x21	041	!
2	0x02	002	STX	34	0x22	042	“
3	0x03	003	ETX	35	0x23	043	#
4	0x04	004	EOT	36	0x24	044	$
5	0x05	005	ENQ	37	0x25	045	%
6	0x06	006	ACK	38	0x26	046	&
7	0x07	007	BEL	39	0x27	047	’
8	0x08	010	BS	40	0x28	050	(
9	0x09	011	TAB	41	0x29	051	)
10	0x0A	012	LF	42	0x2A	052	*
11	0x0B	013	VT	43	0x2B	053	+
12	0x0C	014	FF	44	0x2C	054	,
13	0x0D	015	CR	45	0x2D	055	-
14	0x0E	016	SO	46	0x2E	056	.
15	0x0F	017	SI	47	0x2F	057	/
16	0x10	020	DLE	48	0x30	060	0
17	0x11	021	DC1	49	0x31	061	1
18	0x12	022	DC2	50	0x32	062	2
19	0x13	023	DC3	51	0x33	063	3
20	0x14	024	DC4	52	0x34	064	4
21	0x15	025	NAK	53	0x35	065	5
22	0x16	026	SYN	54	0x36	066	6
23	0x17	027	ETB	55	0x37	067	7
24	0x18	030	CAN	56	0x38	070	8
25	0x19	031	EM	57	0x39	071	9
26	0x1A	032	SUB	58	0x3A	072	:
27	0x1B	033	ESC	59	0x3B	073	;
28	0x1C	034	FS	60	0x3C	074	«
29	0x1D	035	GS	61	0x3D	075	=
30	0x1E	036	RS	62	0x3E	076	»
31	0x1F	037	US	63	0x3F	077	?

Dez	Hex	Okt	Zeichen	Dez	Hex	Okt	Zeichen
64	0x40	100	@	96	0x60	140	‘
65	0x41	101	A	97	0x61	141	a
66	0x42	102	B	98	0x62	142	b
67	0x43	103	C	99	0x63	143	c
68	0x44	104	D	100	0x64	144	d
69	0x45	105	E	101	0x65	145	e
70	0x46	106	F	102	0x66	146	f
71	0x47	107	G	103	0x67	147	g
72	0x48	110	H	104	0x68	150	h
73	0x49	111	I	105	0x69	151	i
74	0x4A	112	J	106	0x6A	152	j
75	0x4B	113	K	107	0x6B	153	k
76	0x4C	114	L	108	0x6C	154	l
77	0x4D	115	M	109	0x6D	155	m
78	0x4E	116	N	110	0x6E	156	n
79	0x4F	117	O	111	0x6F	157	o
80	0x50	120	P	112	0x70	160	p
81	0x51	121	Q	113	0x71	161	q
82	0x52	122	R	114	0x72	162	r
83	0x53	123	S	115	0x73	163	s
84	0x54	124	T	116	0x74	164	t
85	0x55	125	U	117	0x75	165	u
86	0x56	126	V	118	0x76	166	v
87	0x57	127	W	119	0x77	167	w
88	0x58	130	X	120	0x78	170	x
89	0x59	131	Y	121	0x79	171	y
90	0x5A	132	Z	122	0x7A	172	z
91	0x5B	133	[	123	0x7B	173	{
92	0x5C	134	\	124	0x7C	174	|
93	0x5D	135	]	125	0x7D	175	}
94	0x5E	136	^	126	0x7E	176	-
95	0x5F	137	_	127	0x7F	177	DEL

Tabelle A.3: ASCII Zeichentabelle

A.4 Protokolle

A.4.1 NMEA

Die Prüfsumme 'CS' in allen NMEA-Nachrichten-Typen besteht aus einer zweistelligen Hexadezimalzahl, die sich durch ein (bitweise) exklusiv-oder (1+1=0, 1+0=1, 0+0=0) aller Zeichen zwischen dem '$' und dem '*' berechnen.

$GPRMC,hhmmss,a,bbbb.bbbb,c,ddddd.dddd,e,fff.f,ggg.g,ddmmyy,hhh.h,k*CS	
RMC	Satellites in view
hhmmss	UTC time in hhmmss.ss format, 000000.00 235959.99
a	Status, 'V' = navigation receiver warning, 'A' = valid position
bbbb.bbbb	Latitude in ddmm.mmmm format, Leading zeros transmitted
c	Latitude hemisphere indicator, N' = North, 'S' = South
ddddd.dddd	Longitude in dddmm.mmmm format, Leading zeros transmitted
e	Longitude hemisphere indicator, E' = East, 'W' = West
fff.f	Speed over ground, 000.0 999.9 knots
ggg.g	Course over ground, 000.0 359.9 degrees
ddmmyy	UTC date of position fix, ddmmyy format
hhh.h	Magnetic variation, 000.0 180.0 degrees
k	Magnetic variation direction, 'E' = East, 'W' = West
CS	Checksum, the checksum field starts with *

Tabelle A.4: Aufbau NMEA RMC-Nachricht

$GPGSV,a,b,cc,dd,ee,fff,gg,....*CS	
GSV	Satellites in view
a	Total number of GSV messages to be transmitted
b	Number of current GSV message
cc	Total number of satellites in view, 00 12
dd	Satellite PRN number, GPS: 01 32, SBAS: 33 64 (33 = PRN120)
ee	Satellite elevation number, 00 90 degrees
fff	Satellite azimuth angle, 000 359 degrees
gg	C/No, 00 99 dBNull when not tracking
....	for up to 4 satellites per sentence
checksum	Checksum, the checksum field starts with *

Tabelle A.5: Aufbau NMEA GSV-Nachricht

$GPGGA,hhmmss,aaaa.aaaa,b,ccccc.cccc,d,e,ff,g.g,hhhhh.h,k,llll.l,m,nnn,oooo*CS	
GGA	Global Positioning System Fix Data
hhmmss	UTC time in hhmmss.ss format, 000000.00 235959.99
aaaa.aaaa	Latitude in ddmm.mmmm format, Leading zeros transmitted
b	Latitude hemisphere indicator, N' = North, 'S' = South
ccccc.cccc	Longitude in dddmm.mmmm format, Leading zeros transmitted
d	Longitude hemisphere indicator, E' = East, 'W' = West
e	Position fix quality indicator
ff	Number of satellites in use, 00 12
g.g	Horizontal dilution of precision, 00.0 99.9
hhhhh.h	Antenna height above/below mean sea level, -9999.9 17999.9
k	M (Meter) or f (feet)
llll.l	Geoidal height, -999.9 9999.9
m	M (Meter) or f (feet)
nnn	Age of DGPS data since last valid RTCM in seconds
oooo	Differential reference station ID, 0000 1023, NULL when DGPS not used
CS	Checksum, the checksum field starts with *

Tabelle A.6: Aufbau NMEA GGA-Nachricht

$GPGSA,a,b,cc,....,dd.d,ee.e,ff.f*CS	
GSA	GPS DOP and active satellites
a	Mode, 'M' = Manual, 'A' = Automatic
b	Fix type, 1 = not available, 2 = 2D fix, 3 = 3D fix
cc	PRN number, 01 to 32, of satellite
....	used in solution, up to 12 transmitted
dd.d	Position dilution of precision, 00.0 to 99.9
ee.e	Horizontal dilution of precision, 00.0 to 99.9
ff.f	Vertical dilution of precision, 00.0 to 99.9
CS	Checksum, the checksum field starts with *

Tabelle A.7: Aufbau NMEA GSA-Nachricht

A.4.2 SiRF

Transport Message

Start Sequence	Payload Length	Payload	Message Checksum	End Sequence
0xA01, 0xA2	Two-bytes (15-bits)	Up to 2^{10} -1(<1023)	Two-bytes (15-bits)	0xB0, 0xB3

Payload Length

High Byte	Low Byte
$\leq$ 0x7F	Any value

Payload Data

D	Discrete –The field consists of a bit mapped value, or subfields of groups of bits that are described in the Description field. Values should be considered unsigned
S	Signed – The field contains a signed integer value in two's complement format
U	Unsigned – The field contains an unsigned integer value
Dbl	Double precision floating point
Sgl	Single precision floating point

Checksum

High Byte	Low Byte
$\leq$ 0x7F	Any value

A.5 Mathematisches

A.5.1 Herleitung Formel 2.47

$f' = f * \frac{1}{(1-\frac{v}{c})}$ | erweitern mit $1 + \frac{v}{c}$

$f' = f * \frac{1}{(1-\frac{v}{c})} * \frac{1+\frac{v}{c}}{1+\frac{v}{c}}$

$$f' = f * (1 + \tfrac{v}{c}) * \underbrace{\frac{1}{1 - \frac{v^2}{c^2}}}_{\approx 1, \text{ da } \frac{v^2}{c^2} \to 0, \text{ für } v << c}$$

$$\overset{\text{näherungsweise}}{\Rightarrow} f' = f * (1 + \tfrac{v}{c})$$

A.5.2 Arithmetischer Mittelwert

$$MW_{arithm} = \frac{1}{n} * \sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + ... + x_{n-1} + x_n}{n} \tag{A.1}$$

A.5.3 Mittlere absolute Abweichung

$$MW_{absolut} = \frac{1}{n} * \sum_{i=1}^{n} |x_i| = \frac{|x_1| + |x_2| + ... + |x_{n-1}| + |x_n|}{n} \tag{A.2}$$

A.5.4 Varianz

$$Var(X) = E((X - \mu)^2) = \frac{1}{n} * \sum_{i=1}^{n} (x_i - \mu)^2 \tag{A.3}$$

mit μ als Erwartungswert E(X)

A.5.5 Standardabweichung

$$\sigma_X = \sqrt{Var(X)} = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (x_i - \mu)^2} \tag{A.4}$$

A.5.6 Kovarianz

$$\varrho(X,Y) = E\left((X - \mu_x) * (Y - \mu_y)\right) = \frac{1}{n} * \sum_{i=1}^{n} (x_i - \mu_x) * (y_i - \mu_y) \tag{A.5}$$

A.5.7 Korrelationskoeffizient

$$\varrho(X,Y) = \frac{Cov(X,Y)}{\sqrt{\sigma_X} * \sqrt{\sigma_Y}} \tag{A.6}$$

A.6 ATmega168 Register

A Port Data Register - PORTx

Bit	7	6	5	4	3	2	1	0	
	PORTx7	PORTx6	PORTx5	PORTx4	PORTx3	PORTx2	PORTx1	PORTx0	PORTx
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

A Port Data Dirction Register – DDRx

Bit	7	6	5	4	3	2	1	0	
	DDx7	DDx6	DDx5	DDx4	DDx3	DDx2	DDx1	DDx0	DDRx
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

A Port Input Pins Address – PINx

Bit	7	6	5	4	3	2	1	0	
	PINx7	PINx6	PINx5	PINx4	PINx3	PINx2	PINx1	PINx0	PINx
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

USART I/O Data Register – UDR0

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR0 (Read)
	TXB[7:0]								UDR0 (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

USART Control and Status Register - USCR0A

Bit	7	6	5	4	3	2	1	0	
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2XA	MPCM0	USCR0A
Read/Write	R	R/W	R	R	R	R	R/W	R/W	

USART Control and Status Register – USCR0B

Bit	7	6	5	4	3	2	1	0	
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	USCR0B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	

USART Control and Status Register – USCR0C

Bit	7	6	5	4	3	2	1	0	
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	USCR0C
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

USART Baud Rate Register – UBRR0

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	UBRR0[11:8]				UBRR0H
	UBRR0[7:0]								UBRR0L
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

SPI Control Register SPCR

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	-	-	-	-	-	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	

SPI Data Register – SPDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Output Compare Register 1 A – OCR1A

Bit	15	14	13	12	11	10	9	8	
	OCR1a[15:8]								OCR1AH
	OCR1a[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	-	-	-	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	

Timer/Counter1 Interrupt Mask Register – TIMSK1

Bit	7	6	5	4	3	2	1	0	
	-	-	ICIE1	-	-	OCIE1B	OCIE2B	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	

Timer/Counter1 Interrupt Flag Register – TIFR1

Bit	7	6	5	4	3	2	1	0	
	-	-	ICR1	-	-	OCF1B	OCF1A	TOV1	TIFR
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	

A.7 Schaltpläne

A.7.1 GPS-Modul-Adapter

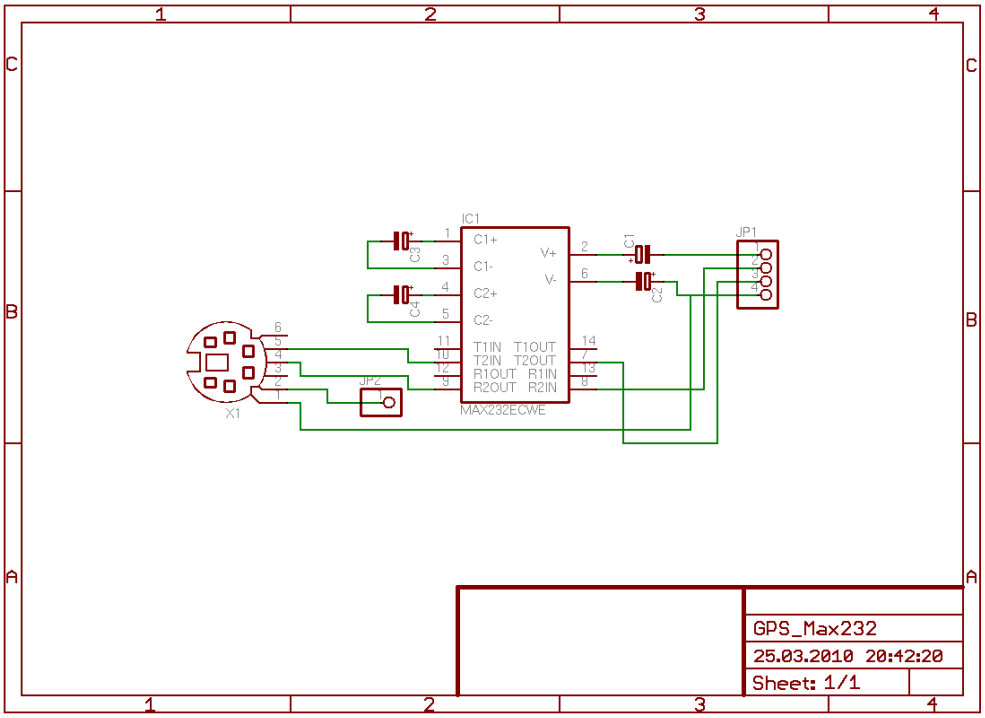


Abbildung A.5: Schaltplan des GPS-Modul-Adapters zum Anschluß an die Hauptplatine

A.7.2 Mainboard

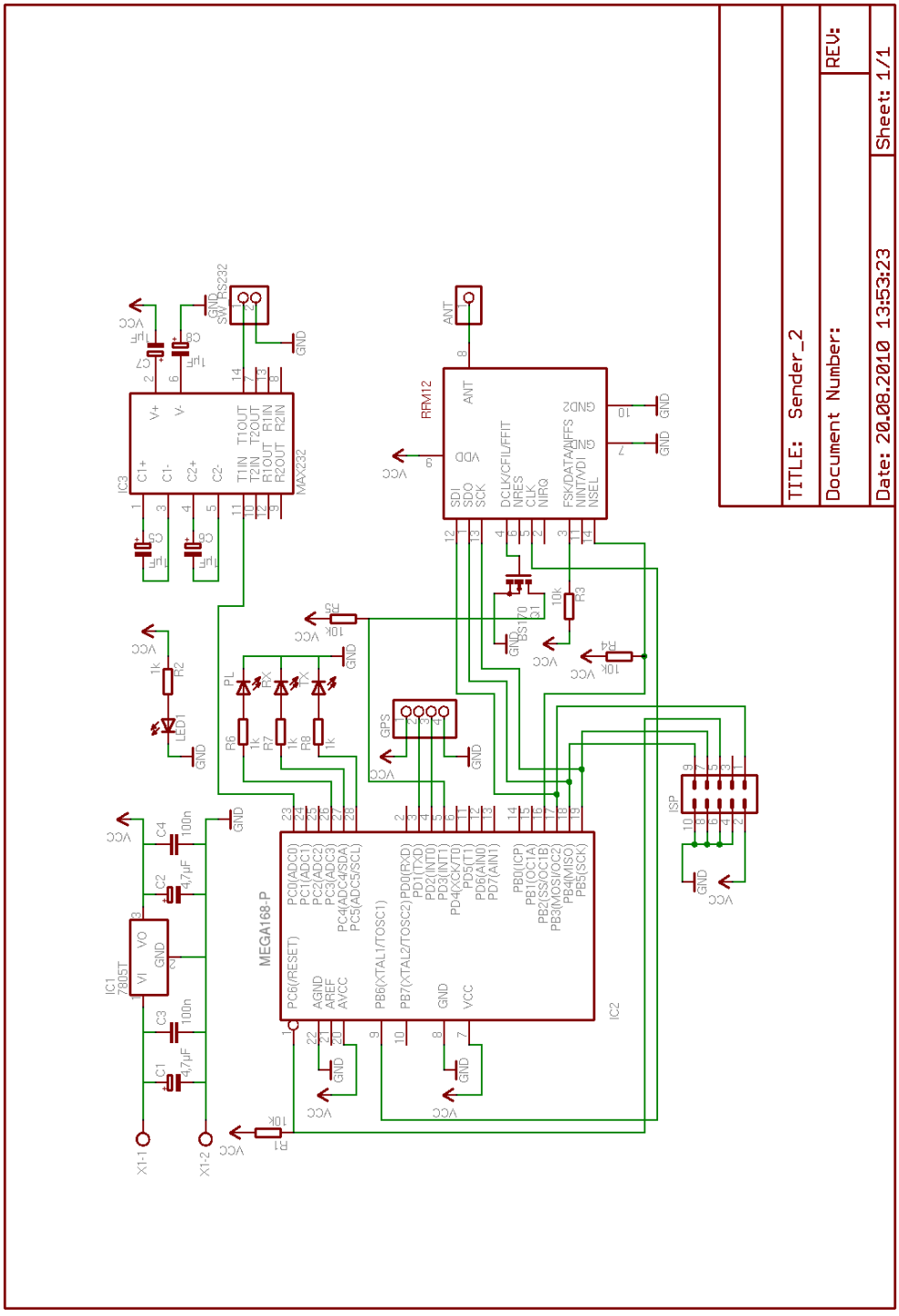


Abbildung A.6: Schaltplan des Mainboards

A.8 Platinen-Layout

A.8.1 GPS-Modul-Adapter

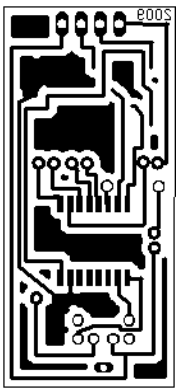


Abbildung A.7: Ätzvorlage für GPS-Modul-Adapter

A.8.2 Mainboard

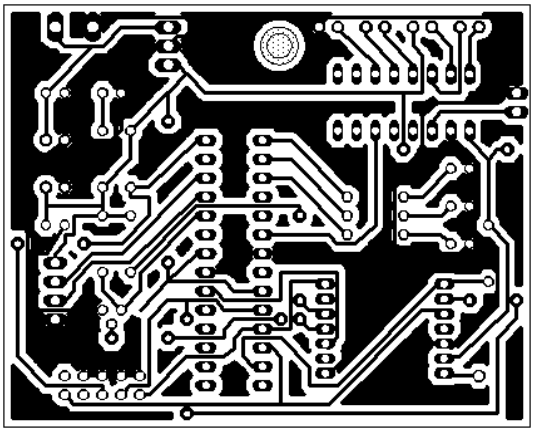


Abbildung A.8: Ätzvorlage für Mainboard

A.9 Programmausdruck

A.9.1 main.h

Quellcode A.1: main.h

```
1  // include avr standard routines
2  #include <avr/io.h>
3  #include <avr/interrupt.h>
4  #include <avr/sleep.h>
5  // stdlib includes functions involving memory allocation, process control,
       conversions and others
6  #include <stdlib.h>
7  #include <string.h>
8
9  #define TRANSMITTER 1
10 #define RECEIVER  0
11
12 #define GPGGA_MESSAGE 1
13
14 #define F_CPU 10000000UL
15 #define SW_UART_BAUD_RATE 9600UL
16
17 #define UART_BAUD_RATE 19200UL
18 #define MODE  TRANSMITTER
19
20 // data-structure to convert a value of 2 bytes easily into two several
       bytes
21 typedef union convert {
22   unsigned int two_bytes;
23   unsigned char byte[2];
24 } CONVERT_TWO_BYTES;
25
26 #include "uart.h"
27 #include "SPI.h"
28 #include "RFM12.h"
29 #include "sw_uart.h"
30 #include "GPS.h"
31 #include <util/delay.h>
```

A.9.2 main.c

Quellcode A.2: main.c

```
1  #include "main.h"
2
3  int main(void){
4    cli();
5    char* NMEA_String[100];
6    char* correction_String[28];
7
8    DDRC=255;
9    PORTC |= (1<<3);
10
11   SPI_init();
12   uart_init();
13   sw_uart_init();
14
15   sw_uart_send_Char('I');
16
17   RFM12_init();
18   RFM12_config(RFM12_BAUDRATE, RFM12_CHANNEL, 0, RFM12_ENVIRONMENT);
19
20   sei();
21
22   unsigned char i;
23   for (i=0; i<20; i++)
24     _delay_ms(10);
25   PORTC &= ~(1<<3);
26
27   uint8_t NMEA_Type;
28
29   while(1){
30     if (MODE == TRANSMITTER){
31       uart_get_String(NMEA_String,70);
32       NMEA_Type = get_NMEA_Type(NMEA_String);
33       if (NMEA_Type == GPGGA_MESSAGE){
34         save_Correction_Data(NMEA_String);
35         generate_Correction_String(correction_String);
36         if (!RFM12_is_busy()){
37           RFM12_send_String(correction_String);
38         }
39       }
40     }
```

```
41
42      if (MODE == RECEIVER){
43        uart_get_String(NMEA_String,70);
44        NMEA_Type = get_NMEA_Type(NMEA_String);
45        if (NMEA_Type == GPGGA_MESSAGE){
46          correct_NMEA(NMEA_String);
47          sw_uart_send_String(NMEA_String);
48        }
49      }
50    }
51  }
```

A.9.3 GPS.h

Quellcode A.3: GPS.h

```
1  #include <util/crc16.h>
2
3  /** @def SERVER_LOCATION_LONGITUDE_PP
4   *  static server location longitude pre-decimal places
5   */
6  #define SERVER_LOCATION_LONGITUDE_PP 5037
7
8  /** @def SERVER_LOCATION_LONGITUDE_DP
9   *  static server location longitude decimal places
10  */
11 #define SERVER_LOCATION_LONGITUDE_DP 8996
12
13 /** @def SERVER_LOCATION_LATITUDE_PP
14  *  static server location latitude pre-decimal places
15  */
16 #define SERVER_LOCATION_LATITUDE_PP 756
17
18 /** @def SERVER_LOCATION_LATITUDE_DP
19  *  static server location latitude decimal places
20  */
21 #define SERVER_LOCATION_LATITUDE_DP 4250
```

A.9.4 GPS.c

Quellcode A.4: GPS.c

```
1  #include "main.h"
2  #include "GPS.h"
3
4  /** Global variable to store NMEA_Data
5   *
6   *  Global char-array with two dimensions to store actual NMEA-data in data-
       strings.
7   */
8  char NMEA_data[6][6];
9
10 /** Global variable for the actual GPS North-South correction data
11  *
12  *  Global variable to store the actual GPS North-South correction data
13  */
14 int Correction_Data_NS;
15
16 /** Global variable for the actual GPS West-East correction data
17  *
18  *  Global variable to store the actual GPS West-East correction data
19  */
20 int Correction_Data_WE;
21
22 /** Global variable for the actual GPS satellites for correction data
23  *
24  *  Global variable to store the actual count of GPS satellites for
       correction data
25  */
26 int Correction_Data_Satellites;
27
28 /** Global variable for the actual GPS hdop for correction data
29  *
30  *  Global variable to store the actual count of GPS hdop for correction
       data
31  */
32 float Correction_Data_hdop;
33
34 /**
35  * \brief Function to convert a char-array into an integer value.
36  *
```

```
37  *  This function converts a char-array into an integer value and returns it
        .
38  *
39  * @param[in] String  pointer to char-array which is to convert
40  * @return  Integer value of the char-array
41  *
42  */
43 int String_to_integer(char* String){
44
45   int value;
46   uint8_t minus;
47
48   value = 0;
49   minus = 0;
50   while( *String ){
51     char help = *String++;
52     if (help != 32){
53       if (help==45){
54         minus = 1;
55       }
56       else{
57         value = (value*10)+(help-48);
58       }
59     }
60   }
61   if (minus==1){
62     value = value * (-1);
63   }
64   return value;
65 }
66
67 /**
68  * \brief Function to convert a char-array into a float value.
69  *
70  *  This function converts a char-array into a float value and returns it.
71  *
72  * @param[in] String  pointer to char-array which is to convert
73  * @return  Float value of the char-array
74  *
75  */
76 float String_to_float(char* String){
77   char delimiters = '.';
78   char* Stringpart;
```

```
79    float value;
80    uint8_t minus;
81    float factor;
82
83    minus = 0;
84    value = 0;
85
86    Stringpart = strsep (&String,&delimiters); //strip of places before the
          decimal point because of accuracy
87    value = String_to_integer(Stringpart);
88    if (value<0){
89      value = value * (-1);
90      minus = 1;
91    }
92
93    Stringpart = strsep (&String,&delimiters);
94    factor = 0.1;
95    while( *Stringpart ){
96      char help = *Stringpart++;
97      value = value + (factor*(help-48));
98      factor = factor * 0.1;
99    }
100
101   if (minus == 1){
102     value = value * (-1);
103   }
104   return value;
105 }
106
107 /**
108  * \brief Function to check the NMEA type of a message.
109  *
110  *  This function tests the incoming char-array for the NMEA-type "GPGGA".
          If it matches, it returns 1, else it returns 0.
111  *
112  * @param[in] NMEA_String Pointer to char-array (NMEA-Message) to test
113  * @return  1 if the incoming char-array is a GPGGA-Message, else 0
114  *
115  */
116 uint8_t get_NMEA_Type(char *NMEA_String){
117   char* test;
118   test = strstr(NMEA_String,"GPGGA");
119   if (test!=NULL){
```

```
120     return GPGGA_MESSAGE;
121   }
122   else{
123     return 0;
124   }
125 }
126
127 /**
128  * \brief Function to tokenize a NMEA-Message.
129  *
130  *  This function tokenzizes an incoming NMEA-Message and saves the relevant
        parts into the global array NMEA_Data.
131  *
132  * @param[in] NMEA_String Pointer to a char-array (NMEA-Message) which is to
        tokenize
133  *
134  */
135 void tokenize_NMEA_data(char *NMEA_String){
136
137   NMEA_String = NMEA_String + 18;
138   memcpy(NMEA_data[0], NMEA_String,4 );
139   NMEA_data[0][4] = '\0';
140
141   NMEA_String = NMEA_String + 5;
142   memcpy(NMEA_data[1], NMEA_String,4 );
143   NMEA_data[1][4] = '\0';
144
145   NMEA_String = NMEA_String + 7;
146   memcpy(NMEA_data[2], NMEA_String,5 );
147   NMEA_data[2][5] = '\0';
148
149   NMEA_String = NMEA_String + 6;
150   memcpy(NMEA_data[3], NMEA_String,4 );
151   NMEA_data[3][4] = '\0';
152
153   NMEA_String = NMEA_String + 9;
154   memcpy(NMEA_data[4], NMEA_String,2 );
155   NMEA_data[4][2] = '\0';
156
157   NMEA_String = NMEA_String + 3;
158   memcpy(NMEA_data[5], NMEA_String,3 );
159   NMEA_data[5][3] = '\0';
160
```

```
161 }
162
163
164 /**
165  * \brief Function to save the actual correction data.
166  *
167  *  This function tokenizes the incoming char-array (NMEA-Message),
        calculates the actual correction data and stores it in
168  *  global variables.
169  *
170  * @param[in] NMEA_String Pointer to a char-array (NMEA-Message) from which
        is to calculate the correction data
171  *
172  */
173 void save_Correction_Data(char* NMEA_String){
174   tokenize_NMEA_data(NMEA_String);
175
176   Correction_Data_NS = SERVER_LOCATION_LONGITUDE_DP - String_to_integer(
        NMEA_data[1]);
177   if (String_to_integer(NMEA_data[0])!=SERVER_LOCATION_LONGITUDE_PP){
178     Correction_Data_NS = 10000 - Correction_Data_NS;
179   }
180   Correction_Data_WE = SERVER_LOCATION_LATITUDE_DP - String_to_integer(
        NMEA_data[3]);
181   if (String_to_integer(NMEA_data[2])!=SERVER_LOCATION_LATITUDE_PP){
182     Correction_Data_WE = 10000 - Correction_Data_WE;
183   }
184
185   Correction_Data_Satellites = String_to_integer(NMEA_data[4]);
186   Correction_Data_hdop = String_to_float(NMEA_data[5]);
187 }
188
189 /**
190  * \brief Function to generate correction string.
191  *
192  *  This function generates the whole correction string from the actual
        correction data and adds a crc-checksum.
193  *
194  * @param[out]  Correction_String Pointer to the char-array the correction
        string is stored
195  * @param[in] NMEA_String Pointer to a char-array from the actual correction
        data is calculated
196  *
```

```
197  */
198 void generate_Correction_String (char* Correction_String){
199   char hstring[12];
200 
201   strcpy(Correction_String,"$");
202 
203   dtostrf(Correction_Data_NS,5,0,hstring);
204   strcat(Correction_String,hstring);
205   strcat(Correction_String,",");
206 
207   dtostrf(Correction_Data_WE,5,0,hstring);
208   strcat(Correction_String,hstring);
209   strcat(Correction_String,",");
210 
211   if (Correction_Data_Satellites<10){
212     dtostrf(Correction_Data_Satellites,1,0,hstring);
213   }
214   else{
215     dtostrf(Correction_Data_Satellites,2,0,hstring);
216   }
217 
218   strcat(Correction_String,hstring);
219   strcat(Correction_String,",");
220 
221   dtostrf(Correction_Data_hdop,3,1,hstring);
222   strcat(Correction_String,hstring);
223 
224   strcat(Correction_String,"*");
225 
226   char* Copy_of_Correction_String;
227   Copy_of_Correction_String=Correction_String;
228   *Copy_of_Correction_String++;
229   char crc;
230   crc = 0;
231   uint8_t i;
232   for (i=0; i<(strlen(Copy_of_Correction_String)-1); i++){
233     crc = _crc_ibutton_update(crc, Copy_of_Correction_String[i]);
234   }
235 
236   char tail[4];
237   tail[0]=crc;
238   tail[1]=13;
239   tail[2]=10;
```

```
240   tail[3]='\0';
241   strcat(Correction_String,tail);
242 }
243
244 /**
245  * \brief Function to decode a received correction string.
246  *
247  *  This function tests the crc-checksum of a received correction string. If
       the crc-checksum is OK, the function
248  *  decodes the correction data and stores it in global variables.
249  *
250  * @param[in] Correction_String Pointer to the char-array where the received
       correction string is stored
251  *
252  */
253 void decode_Correction_String(char* Correction_String){
254   char* received_Data;
255   char* crc_String;
256   char received_crc;
257
258   received_Data = strsep (&Correction_String,"*");
259   crc_String =strsep (&Correction_String,"*");
260   received_crc = crc_String[0];
261
262   *received_Data++;
263
264   char crc;
265   crc = 0;
266   uint8_t i;
267   for (i=0; i<strlen(received_Data); i++){
268     crc = _crc_ibutton_update(crc, received_Data[i]);
269   }
270
271   if (crc==received_crc){
272     uint8_t counter;
273     char* Data;
274     counter = 0;
275     while (received_Data!=NULL){
276       Data  = strsep (&received_Data,",");
277       switch (counter){
278         case 0:Correction_Data_NS = String_to_integer(Data);break;
279         case 1:Correction_Data_WE = String_to_integer(Data);break;
280         case 2:Correction_Data_Satellites = String_to_integer(Data);break;
```

```
281           case 3:Correction_Data_hdop = String_to_float(Data);break;
282         }
283         counter++;
284       }
285     }
286     else{
287       LED_PORT |= (1<<LED_ERR);
288     }
289 }
290
291 /**
292  * \brief Function to convert integer value (0-15) into hexadecimal.
293  *
294  *  This function converts an integer value between 0 and 15 into
        hexadecimal char-letter.
295  *
296  * @param[in] i Integer value (0-15) which is to convert into a char-letter
297  * @return  the hexadecimal letter of the incoming integer value
298  *
299  */
300 char hex(uint8_t i){
301   char c;
302   if (i<10){
303     c =(char)(i + 48);
304   }
305   else{
306     c =(char)(i + 55);
307   }
308   return c;
309 }
310
311 /**
312  * \brief Function which corrects the incoming NMEA-string
313  *
314  *  This function corrects the incoming NMEA-string. It tokenizes the NMEA-
        string, gets the important data, corrects the
315  *  data with the actual correction data and overrides the corresponding
        chars in the NMEA-string
316  *
317  * @param[in,out] NMEA_String Integer value (0-15) which is to convert into
        a char-letter
318  *
319  */
```

```
320 void correct_NMEA(char* NMEA_String){
321   char hstring[8];
322   char corrected_String[23];
323   int corrected_DP;
324   int corrected_PP;
325
326   tokenize_NMEA_data(NMEA_String);
327
328   corrected_PP = String_to_integer(NMEA_data[1]) + Correction_Data_NS;
329   corrected_DP = String_to_integer(NMEA_data[0]);
330   if (corrected_PP<0){
331     corrected_DP = corrected_DP - 1;
332     corrected_PP = corrected_PP + 10000;
333   }
334   if (corrected_PP>=10000){
335     corrected_DP = corrected_DP + 1;
336     corrected_PP = corrected_PP - 10000;
337   }
338
339   dtostrf(corrected_DP,4,0,corrected_String);
340   strcat(corrected_String,".");
341   dtostrf(corrected_PP,4,0,hstring);
342   strcat(corrected_String,hstring);
343   strcat(corrected_String,",N,");
344
345   corrected_PP = String_to_integer(NMEA_data[3]) + Correction_Data_WE;
346   corrected_DP = String_to_integer(NMEA_data[2]);
347   if (corrected_PP<0){
348     corrected_DP = corrected_DP - 1;
349     corrected_PP = corrected_PP + 10000;
350   }
351   if (corrected_PP>=10000){
352     corrected_DP = corrected_DP + 1;
353     corrected_PP = corrected_PP - 10000;
354   }
355
356   dtostrf(corrected_DP,5,0,hstring);
357   strcat(corrected_String,hstring);
358   strcat(corrected_String,".");
359   dtostrf(corrected_PP,4,0,hstring);
360   strcat(corrected_String,hstring);
361
362   char* help_Pointer;
```

```
363   help_Pointer = NMEA_String;
364   uint8_t counter;
365   help_Pointer = help_Pointer + 18;
366   counter = 0;
367   while(counter<22){
368     if (corrected_String[counter]==32){
369       help_Pointer[counter] = '0';
370     }
371     else{
372       help_Pointer[counter] = corrected_String[counter];
373     }
374     counter = counter + 1;
375   }
376
377   char crc;
378   crc = 0;
379   uint8_t i;
380   for (i=1; i<strlen(NMEA_String)-5; i++){
381     crc ^= NMEA_String[i];
382   }
383
384   char crc_String_1;
385   char crc_String_2;
386   crc_String_2 = hex((uint8_t)crc % 16);
387   crc_String_1 = hex(((uint8_t)((uint8_t)crc / 16)));
388   NMEA_String[strlen(NMEA_String)-4] =crc_String_1;
389   NMEA_String[strlen(NMEA_String)-3] =crc_String_2;
390 }
```

A.9.5 SPI.h

Quellcode A.5: SPI.h

```
1 #include <avr/io.h>
2
3 // definition of the SPI port
4 #define SPI_PORT PORTB
5 #define SPI_DDR DDRB
6 #define SPI_SS 2
7 #define SPI_MOSI 3
8 #define SPI_MISO 4
9 #define SPI_SCK 5
```

A.9.6 SPI.c

Quellcode A.6: SPI.c

```
1
2  #include "main.h"
3  #include "SPI.h"
4
5  /**
6   * \brief Function to activate SPI-slave for communication.
7   *
8   *  This function activates SPI-slave for communication by setting
        responsible SS (Slave Select) pin low (0).
9   *
10  */
11 void SPI_on(void){
12   SPI_PORT &= ~(1<<SPI_SS);
13 }
14
15 /**
16  * \brief Function to disable SPI-slave for communication.
17  *
18  *  This function disables SPI-slave for communication by setting
        responsible SS (Slave Select) pin high (1).
19  *
20  */
21 void SPI_off(void){
22   SPI_PORT |= (1<<SPI_SS);
23 }
24
25 /**
26  * \brief Function to initialize SPI interface.
27  *
28  *  This function initializes SPI interface and activates SPI-slave.
29  *
30  */
31 void SPI_init(void){
32   SPI_DDR &= ~(1<<SPI_MISO);
33   SPI_DDR |= (1<<SPI_MOSI)|(1<<SPI_SCK)|(1<<SPI_SS);
34   SPI_PORT |= (1<<SPI_SS);
35   SPCR |= (1<<SPR1)|(1<<SPR0);
36   SPCR |= (1<<SPIE)|(1<<SPE)|(1<<MSTR);
37   SPI_on();
38 }
```

```
39
40 /**
41  * \brief Function to send/receive data to/from SPI-slave.
42  *
43  *  This function sends data to SPI-slave and simultaneously receives data
        from it.
44  *
45  * @param[in] send_Byte The byte to send to SPI-slave
46  * @return  The byte received from SPI-slave
47  *
48  */
49 unsigned char SPI_send_receive_Byte(unsigned char send_Byte){
50
51   unsigned char received_Byte;
52
53   SPDR = send_Byte;
54   while(!(SPSR & (1<<SPIF))){
55   }
56   received_Byte = SPDR;
57   return received_Byte;
58 }
```

A.9.7 RFM12.h

Quellcode A.7: RFM12.h

```
1  #include <avr/io.h>
2  #include <avr/interrupt.h>
3
4  // macro to calculate a frequency value for the RFM12 out of frequency in
       MHz
5  #define RFM12_FREQUENCY(frequency)  ((unsigned short)((frequency-430.0)
       /0.0025))
6
7  #define QUIET    1
8  #define NORMAL     2
9  #define NOISY    3
10
11 #define LED_PORT PORTC
12 #define LED_DDR DDRC
13 #define LED_TX     5
14 #define LED_RX     4
15 #define LED_ERR    3
16
17 #define RFM12_PORT  PORTB
18 #define RFM12_DDR DDRB
19 #define RFM12_PIN  PINB
20 #define RFM12_CS     2
21 #define RFM12_SDI    3
22 #define RFM12_SDO    4
23 #define RFM12_SCK    5
24
25 #define RFM12_INT_DDR DDRD
26 #define RFM12_INT_PORT  PORTD
27 #define RFM12_INT_PIN 4
28
29 // definition of some RFM12 configuration values
30 #define RFM12_CHANNEL      1
31 #define RFM12_BAUDRATE       20000
32 #define RFM12_ENVIRONMENT    NORMAL
33 #define RFM12_BASEBAND_FREQUENCY  433.4
```

A.9.8 RFM12.c

Quellcode A.8: RFM12.c

```
1  #include "main.h"
2  #include "RFM12.h"
3  #include <util/delay.h>
4
5
6  /** Global variable to store received data
7   *
8   *  Global char to store actual received data.
9   */
10 unsigned char Received_Data;
11
12 /** Global variable to store a received char
13  *
14  *  Global char to store the actual received char.
15  */
16 unsigned char new_Char;
17
18 /** Global variable to store received_Char state
19  *
20  *  Global char to store the state if a new char is received.
21  */
22 volatile unsigned char received_Char;
23
24 /** Global variable to store the state of the RFM12
25  *
26  *  Global variable to store the actual state of the RFM12.
27  */
28 uint8_t busy;
29
30 /** Global variable to store received chars
31  *
32  *  Global char-array to store the received chars.
33  */
34 char Correction_String[40];
35
36 /** Global variable to store the count of received chars
37  *
38  *  Global integer variable to store the actual count of received chars.
39  */
40 int Correction_String_Counter = 0;
```

```
41
42 /**
43  * \brief Function to transmit data to RFM12.
44  *
45  *  This function transmits two bytes (1 byte command, 1 byte data) to RFM12
        via SPI interface.
46  *
47  * @param[in] value Two byte value to transmit (1 byte command, 1 byte data)
48  * @return  two byte data received  from RFM12
49  *
50  */
51 unsigned short RFM12_transmit(unsigned short value){
52   CONVERT_TWO_BYTES val;
53    val.two_bytes=value;
54   SPI_on();
55    val.byte[1]=SPI_send_receive_Byte(val.byte[1]);
56    val.byte[0]=SPI_send_receive_Byte(val.byte[0]);
57   SPI_off();
58    return val.two_bytes;
59 }
60
61 /**
62  * \brief Function to set the signal characteristic of RFM12.
63  *
64  *  This function sets the bandwidth
65  *
66  * @param[in] bandwidth Baseband bandwidth of RFM12 in kHz
67  * @param[in] gain    Incoming signal gain (dependend on environment)
68  * @param[in] drssi   DRSSI (Digital Received Signal Strength Indication)
        threshold
69  *
70  */
71 void RFM12_set_Signal(unsigned char bandwidth, unsigned char gain, unsigned
      char drssi){
72   RFM12_transmit(0x9500|((gain&3)<<3)|((bandwidth&7)<<5)|(drssi&7));
73 }
74
75 /**
76  * \brief Function to set frequency of the RFM12.
77  *
78  *  This function sets the baseband frequency of the RFM12.
79  *
80  * @param[in] frequency Baseband frequency of RFM12 in kHz
```

```
81  *
82  */
83 void RFM12_set_Frequency(unsigned short frequency){
84   if (frequency<96)
85     frequency=96;
86   else if (frequency>3903)
87     frequency=3903;
88   RFM12_transmit(0xA000|frequency);
89 }
90
91 /**
92  * \brief Function to set baudrate of the RFM12.
93  *
94  *  This function sets the baudrate of the RFM12.
95  *
96  * @param[in] Baud_Rate Baudrate of RFM12 in kHz
97  *
98  */
99 void RFM12_set_Baud_Rate(unsigned short Baud_Rate){
100   if (Baud_Rate<664)
101     Baud_Rate=664;
102   if (Baud_Rate<5400)
103     RFM12_transmit(0xC680|((43104/Baud_Rate)-1));
104   else
105     RFM12_transmit(0xC600|((344828UL/Baud_Rate)-1));
106 }
107
108 /**
109  * \brief Function to set transmission power of the RFM12.
110  *
111  *  This function sets the transmission power of the RFM12 and the deviation
      of high- and low- level.
112  *
113  * @param[in] power   Transmission power
114  * @param[in] deviation Frequency deviation
115  *
116  */
117 void RFM12_set_Power(unsigned char power, unsigned char deviation){
118   RFM12_transmit(0x9800|(power&7)|((deviation&15)<<4));
119 }
120
121 /**
122  * \brief Function to wait till RFM12 is ready.
```

```
123  *
124  *  This function selects RFM12 via SPI and waits until RFM12 is ready.
125  *
126  */
127 static inline void RFM12_ready(void){
128   SPI_on();
129   asm("nop");
130   asm("nop");
131    while (!(RFM12_PIN&(1<<RFM12_SDO)));
132 }
133
134 /**
135  * \brief Function to set RFM12 into RX mode.
136  *
137  *  This function sets the RFM12 into receiving mode and activates the
         microcontroller interrupt.
138  *
139  */
140 void RFM12_RX_Mode(void){
141    busy = 0;
142    RFM12_transmit(0x82C8);
143    RFM12_transmit(0xCA81);
144    _delay_ms(.8);
145    RFM12_transmit(0xCA83);
146    RFM12_transmit(0);
147    EIFR=(1<<INTF1);
148   EIMSK|=(1<<INT1);
149 }
150
151 /**
152  * \brief Function to stop RFM12 RX mode.
153  *
154  *  This function stops the receiving mode of the RFM12 and deactivates
         microcontroller interrupt.
155  *
156  */
157 void RFM12_Stop_RX(void){
158    busy = 1;
159   EIMSK&=~(1<<INT1);
160    RFM12_transmit(0x8208);
161    _delay_ms(1);
162 }
163
```

```
164
165
166 /**
167  * \brief Function to configure RFM12.
168  *
169  *  This function configures some operating values of the RFM12.
170  *
171  * @param[in] baudrate  Baudrate of the RFM12
172  * @param[in] channel   Channel of the RFM12
173  * @param[in] power   Transmission power of the RFM12
174  * @param[in] environment Environment of the RFM12 (QUIET,NORMAL,NOISY)
175  *
176  */
177 void RFM12_config(unsigned short baudrate, unsigned char channel, unsigned
        char power, unsigned char environment){
178   RFM12_set_Frequency(RFM12_FREQUENCY(RFM12_BASEBAND_FREQUENCY)+13*channel);
179     RFM12_set_Power(0, 5);
180     RFM12_set_Signal(4, environment, 0);
181   RFM12_set_Baud_Rate(baudrate);
182 }
183
184 /**
185  * \brief Function to initialize RFM12.
186  *
187  *  This function configures microcontroller for communication with the
        RFM12, initializes RFM12 with some basic values,
188  *  and set the module into RX mode.
189  *
190  */
191 void RFM12_init(void)
192 {
193   RFM12_DDR&=~(1<<RFM12_SDO);
194   RFM12_DDR|=(1<<RFM12_SDI)|(1<<RFM12_SCK)|(1<<RFM12_CS);
195   RFM12_PORT=(1<<RFM12_CS);
196   DDRD&=~8;
197   PORTD|=8;
198   SPCR=(1<<SPE)|(1<<MSTR);
199
200   unsigned char i;
201   for (i=0; i<20; i++){
202     _delay_ms(10);
203   }
204
```

```
205
206   RFM12_transmit(0xC0E0);
207   RFM12_transmit(0x80D7);
208   RFM12_transmit(0xC2AB);
209   RFM12_transmit(0xCA81);
210   RFM12_transmit(0xE000);
211   RFM12_transmit(0xC800);
212   RFM12_transmit(0xC4F7);
213
214   received_Char=0;
215   busy = 0;
216   Correction_String_Counter = 0;
217
218  RFM12_RX_Mode();
219 }
220
221
222 /**
223  * \brief Function to send byte via RFM12.
224  *
225  *  This function transmits a byte via RFM12.
226  *
227  * @param[in] value Byte to transmit via RFM12
228  *
229  */
230 void RFM12_TX_Byte(unsigned char value){
231   RFM12_ready();
232   RFM12_transmit(0xB800|value);
233   if ((value==0x00)||(value==0xFF)){
234     RFM12_ready();
235     RFM12_transmit(0xB8AA);
236   }
237 }
238
239 /**
240  * \brief Function to receive byte via RFM12.
241  *
242  *  This function receives a byte via RFM12 and returns it.
243  *
244  * @return byte received via RFM12
245  *
246  */
247 unsigned char RFM12_RX_Byte(void){
```

```
248   unsigned char value;
249   value =RFM12_transmit(0xB000);
250   if ((value==0x00)||(value==0xFF)){
251     RFM12_ready();
252     RFM12_transmit(0xB000);
253   }
254   return value;
255 }
256
257 /**
258  * \brief Function to send data byte via RFM12.
259  *
260  *  This function sets RFM12 into transmission mode, transmits some sync
        bytes, transmits the data byte and stops TX mode.
261  *
262  * @param[in] byte  Data byte to transmit via RFM12
263  *
264  */
265 void RFM12_send_Byte(unsigned char byte){
266   LED_PORT |= (1<<LED_TX);
267   RFM12_transmit(0x8238);
268   RFM12_ready();
269   RFM12_transmit(0xB8AA);
270   RFM12_ready();
271   RFM12_transmit(0xB8AA);
272   RFM12_ready();
273   RFM12_transmit(0xB8AA);
274   RFM12_ready();
275   RFM12_transmit(0xB82D);
276   RFM12_ready();
277   RFM12_transmit(0xB8D4);
278
279   RFM12_TX_Byte(byte);
280
281   RFM12_TX_Byte(0);
282   RFM12_transmit(0x8208);
283   LED_PORT &= ~(1<<LED_TX);
284 }
285
286
287
288 ISR(SIG_INTERRUPT1){
289 if (!Received_Data)
```

```
290   {
291     LED_PORT |= (1<<LED_RX);
292     Received_Data=1;
293     new_Char = RFM12_RX_Byte();
294 
295     Correction_String[Correction_String_Counter]=new_Char;
296     Correction_String_Counter++;
297 
298     if (new_Char == 10){
299       Correction_String[Correction_String_Counter]='\0';
300       decode_Correction_String(Correction_String);
301       Correction_String_Counter=0;
302     }
303   }
304   else
305   {
306     RFM12_transmit(0xCA81);
307     RFM12_transmit(0xCA83);
308     LED_PORT &= ~(1<<LED_RX);
309     received_Char=1;
310 
311     Received_Data=0;
312   }
313 }
314 
315 /**
316  * \brief Function to send a char via RFM12.
317  *
318  *  This function stops RFM12 RX mode, sends a char and activates RX mode
        again.
319  *
320  * @param[in] c Char to transmit via RFM12
321  *
322  */
323 void RFM12_send_Char(unsigned char c){
324   RFM12_Stop_RX();
325   RFM12_send_Byte(c);
326   RFM12_RX_Mode();
327 }
328 
329 /**
330  * \brief Function to check if a char is received via RFM12.
331  *
```

```
332  *  This function checks if a char is received via RFM12 and returns 1, else
         it returns 0.
333  *
334  * @return  1 if a char is received, 0 if not.
335  *
336  */
337 unsigned char RFM12_received_Char(void){
338   if(received_Char == 0)
339     return 0;
340   else
341     return 1;
342 }
343
344 /**
345  * \brief Function to get a received char from RFM12.
346  *
347  *  This function waits until RFM12 received a char and returns it.
348  *
349  * @return  char received by RFM12
350  *
351  */
352 unsigned char RFM12_get_Char(void){
353     while (!RFM12_received_Char());
354   received_Char = 0;
355     return new_Char;
356 }
357
358 /**
359  * \brief Function to check if RFM12 is  busy.
360  *
361  *  This function checks if RFM12 is in TX mode and if it is busy the
       function returns 0.
362  *
363  * @return  1 if a char is received, 0 if not.
364  *
365  */
366 unsigned char RFM12_is_busy(void)
367 {
368   if(busy == 0)
369     return 0;
370   else
371     return 1;
372 }
```

```
373
374 /**
375  * \brief Function to send a char-array via RFM12.
376  *
377  *  This function sends a char-array via RFM12.
378  *
379  * @param[in] String  Pointer to char-array, which should be send
380  *
381  */
382 void RFM12_send_String( char *String ){
383   while( *String ){
384     RFM12_send_Char( *String++ );
385   }
386 }
387
388 /**
389  * \brief Function to get a char-array received by RFM12.
390  *
391  *  This function sends a char-array via RFM12.
392  *
393  * @param[in,out] String_Buffer Char-array, where the received chars should
     be stored
394  *
395  */
396 void RFM12_get_String(char String_Buffer[40]){
397   uint8_t Next_char;
398
399   int counter;
400   counter = 0;
401
402   Next_char = RFM12_get_Char();
403   String_Buffer[counter]=Next_char;
404
405
406   while (Next_char!=10){
407     counter++;
408     Next_char = RFM12_get_Char();
409     String_Buffer[counter]=Next_char;
410   }
411
412   counter++;
413   String_Buffer[counter]='\0';
414 }
```

A.9.9 uart.h

Quellcode A.9: uart.h

```
1  #include <avr/io.h>
2
3  // definition of default CPU frequency
4  #ifndef F_CPU
5  #warning "F_CPU was not defined, is now defined with 4000000"
6  #define F_CPU 4000000UL
7  #endif
8
9  // definition of default hardware uart baud rate
10 #ifndef UART_BAUD_RATE
11 #warning "UART_BAUD_RATE was not defined, is now defined with 4800"
12 #define UART_BAUD_RATE 4800UL
13 #endif
14
15 // calculation and definition of UBBR register value
16 #define UART_UBRR_VALUE ((F_CPU/UART_BAUD_RATE/16)-1)
17
18 // definition of hardware uart port
19 #define uart_DDR DDRD
20 #define uart_TX 1
21 #define uart_RX 0
```

A.9.10 uart.c

Quellcode A.10: uart.c

```
1  #include "main.h"
2  #include "uart.h"
3
4  /**
5   * \brief Function to initialize uart.
6   *
7   *  This function initializes the uart interface.
8   *
9   */
10 void uart_init(void){
11   UCSR0B = (1<<RXEN0)|(1<<TXEN0);
12
13   UCSR0B &= ~(1<<UCSZ02); // Asynchron 8N1
14   UCSR0C |= (1<<UCSZ00)|(1<<UCSZ01);     // Asynchron 8N1
15   UCSR0C &= ~(1<<UPM01) & ~(1<<UPM00) & ~(1<<USBS0);     // Asynchron 8N1
16   UCSR0C &= ~(1<<UMSEL01)& ~(1<<UMSEL00); // Asynchron 8N1
17
18   UBRR0H = (unsigned char)(UART_UBRR_VALUE>>8);
19   UBRR0L = (unsigned char)UART_UBRR_VALUE;
20   uart_DDR |= (1<<uart_TX);
21   uart_DDR &= ~(1<<uart_RX);
22 }
23
24 /**
25  * \brief Function to send a char via uart.
26  *
27  *  This function sends a char via uart.
28  *
29  * @param[in] Char  Char to send
30  *
31  */
32 void uart_send_Char(unsigned char Char){
33   while (!(UCSR0A & (1<<UDRE0))){
34   }
35   UDR0 = Char;
36 }
37
38 /**
39  * \brief Function to send a char-array via uart.
40  *
```

```
41  *  This function sends a char-array via uart.
42  *
43  * @param[in] String  Pointer to char-array to send
44  *
45  */
46 void uart_send_String(char *String){
47   while (*String){
48     uart_send_Char(*String);
49     String++;
50   }
51 }
52
53 /**
54  * \brief Function to get a char received via uart.
55  *
56  *  This function waits until a char is received via uart and returns it.
57  *
58  * @return  char received via uart
59  *
60  */
61 unsigned char uart_get_Char(void){
62   while (!(UCSR0A & (1<<RXC0))){
63   }
64   return UDR0;
65 }
66
67 /**
68  * \brief Function to get char-array received via uart.
69  *
70  *  This function receives chars via uart and stores it in a char-array
        until the received char is an ascii 10.
71  *
72  * @param[in,out] String_Buffer Char-array, where the received chars should
        be stored
73  *
74  */
75 void uart_get_String(char* String_Buffer){
76   uint8_t Next_char;
77
78   Next_char = uart_get_Char();
79
80   while(Next_char != 10){
81     *String_Buffer++ = Next_char;
```

```
82      Next_char = uart_get_Char();
83    }
84
85    *String_Buffer++ = Next_char;
86    *String_Buffer++ = '\0';
87  }
```

A.9.11 sw_uart.h

Quellcode A.11: sw_uart.h

```
1  #include <avr/io.h>
2  #include <avr/interrupt.h>
3
4  // definition of default CPU frequency
5  #ifndef F_CPU
6  #warning "F_CPU was not defined, is now defined with 4000000"
7  #define F_CPU 4000000UL
8  #endif
9
10 // definition of default software uart baud rate
11 #ifndef SW_UART_BAUD_RATE
12 #warning "SW_UART_BAUD_RATE was not defined, is now defined with 4800"
13 #define SW_UART_BAUD_RATE 4800UL
14 #endif
15
16 // definition of software uart port
17 #define SW_UART_DDR DDRC
18 #define SW_UART_PORT PORTC
19 #define SW_UART_TX 0
20 #define SW_UART_RX 0
```

A.9.12 sw_uart.c

Quellcode A.12: sw_uart.c

```
1  #include "main.h"
2  #include "sw_uart.h"
3
4  static volatile uint16_t sw_uart_Outframe;
5
6  /**
7   * \brief Function to initialize software uart.
8   *
9   *  This function initializes the software uart interface.
10  *
11  */
12 void sw_uart_init(void){
13    cli();
14
15    TCCR1A = 0;
16    TCCR1B = (1 << WGM12) | (1 << CS10) | (0 << ICES1) | (1 << ICNC1);
17
18    OCR1A = (uint16_t) ((uint32_t) F_CPU/SW_UART_BAUD_RATE);
19
20    SW_UART_PORT |= (1 << SW_UART_TX);
21    SW_UART_DDR  |= (1 << SW_UART_TX);
22    sw_uart_Outframe = 0;
23
24 }
25
26 /**
27  * \brief Function to send a char via software uart.
28  *
29  *  This function sends a char via software uart.
30  *
31  * @param[in] Char  Char to send
32  *
33  */
34 void inline sw_uart_send_Char(char Char){
35    do{
36      sei();
37      __asm volatile ("nop");
38      cli();
39    }while (sw_uart_Outframe);
40
```

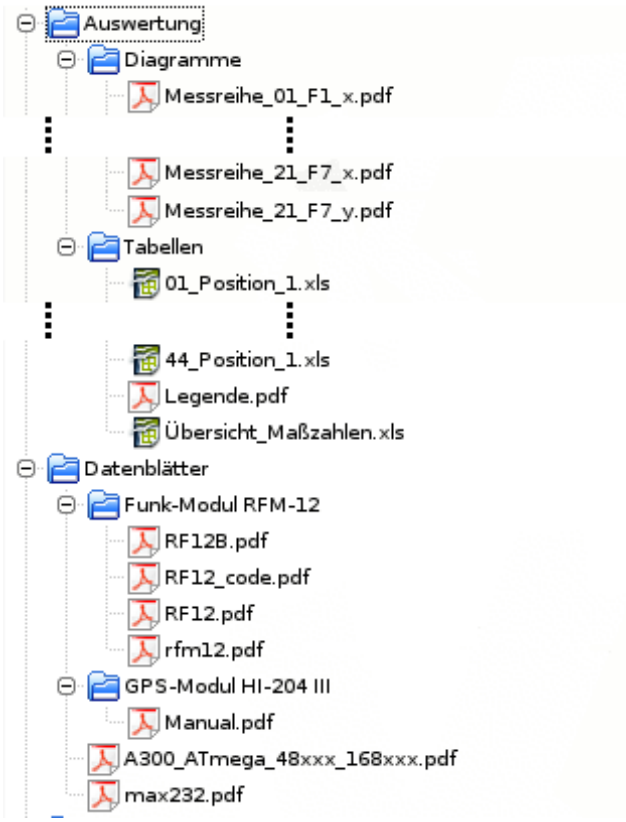
```
41    sw_uart_Outframe = (3 << 9) | (((uint8_t) Char) << 1);   // frame = *.P
          .7.6.5.4.3.2.1.0.S    S=Start(0), P=Stop(1), *=End-Marker(1)
42
43    TIMSK1 |= (1 << OCIE1A);
44    TIFR1    |= (1 << OCF1A);
45
46    sei();
47  }
48
49  /**
50   * \brief Interrupt routine of Timer 1.
51   *
52   *  This interrupt routine handles the interrupt triggered by comparison of
          OCR1A and TCNT1. Each call, the next bit of
53   *  sw_uart_Outframe is generated.
54   *
55   *
56   */
57  SIGNAL (SIG_OUTPUT_COMPARE1A){
58    cli();
59    uint16_t data = sw_uart_Outframe;
60    if (data & 1){
61      SW_UART_PORT |=  (1 << SW_UART_TX);
62    }
63    else{
64      SW_UART_PORT &= ~(1 << SW_UART_TX);
65    }
66
67    if (1 == data){
68      TIMSK1 &= ~(1 << OCIE1A);
69    }
70
71    sw_uart_Outframe = data >> 1;
72    sei();
73  }
74
75  /**
76   * \brief Function to send a char-array via software uart.
77   *
78   *  This function sends a char-array via software uart.
79   *
80   * @param[in] String  Pointer to char-array to send
81   *
```

```
82  */
83 void sw_uart_send_String(char *String){
84   while( *String ){
85     sw_uart_send_Char ( *String++ );
86   }
87 }
```

A.10 Inhalt der beiliegenden Daten-CD

Die beiliegende Daten-CD enthält im Hauptverzeichnis eine digitale Version dieser Arbeit. In dem Ordner 'Digitale Literatur' befinden sich alle von mir verwendeten digitalen Dokumente. Die originalen Daten (NMEA-Datensätze) der 44 Messreihen, sowohl der Referenzstation als auch der mobilen Einheit, sind im Ordner 'GPS Log Dateien' in dem jeweiligen Unterordner abgespeichert. Im Verzeichnis 'Auswertung' sind im Unterordner 'Tabellen' alle zur Auswertung verwendeten Daten in Excel-Dateien zu finden. Hierzu zählen die 44 Messreihen sowie die Übersicht aller Maßzahlen der Datenreihen in der Datei 'Übersicht_Maßzahlen.xls'. Der in dieser Arbeit entwickelte Quellcode zur Programmierung der Mikrocontroller mit der entsprechenden Dokumentation ('refman.pdf') ist ebenfalls auf der CD zu finden. Auch die mit dem Programm Eagle erstellten Schaltpläne und Platinenlayouts sind auf der CD vorhanden. Im Verzeichnis 'Datenblätter' sind alle Datenblätter der verwendeten Hardware in den entsprechenden Unterordnern bereit gestellt.

Der folgende Dateibaum soll eine Übersicht über Inhalt und Aufbau der Daten-CD geben:



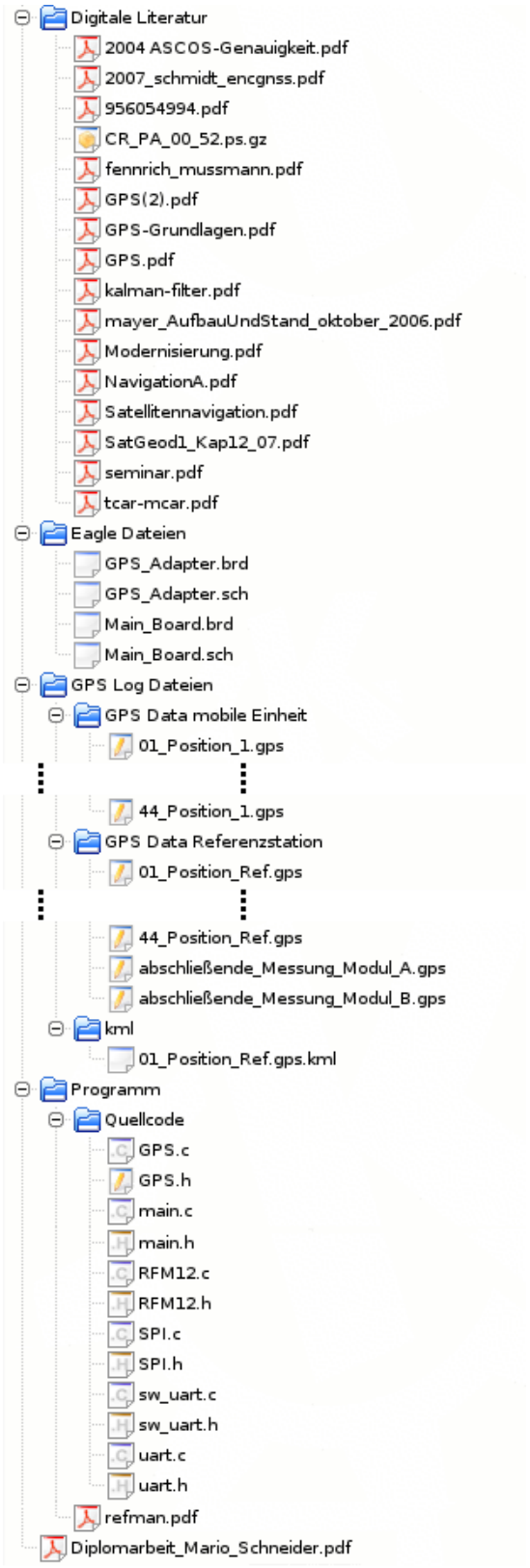


Abbildung A.9: Dateibaum der beiliegenden CD

Stichwortverzeichnis

Literaturverzeichnis

[Bau97] Bauer, M.: *Vermessung und Ortung mit Satelliten.* 4. Auflage. Heidelberg : Herbert Wichmann Verlag, 1997

[BG75] Bahrenberg, G. ; Giese, E.: *Statistische Methoden und ihre Anwendung in der Geographie.* Stuttgart : Teubner, 1975

[BH97] Brown, R.G. ; Hwang, P.Y.C.: *Introduction to Random Signals and Applied Kalman Filtering.* Third edition. New York : John Wiley & Sons, Inc., 1997

[BLR83] Brennecke, J. ; Lelgemann, D. ; Reinhart, E.: A European Astro-Gravimetric Geoid. (1983), Nr. 269

[BP07] Böge, Wolfgang ; Plassmann, Wilfried: *Vieweg Handbuch Elektrotechnik.* 4. Auflage. Wiesbaden : Vieweg Teubner, 2007

[BS98] Bergmann ; Schäfer: *Lehrbuch der Experimentalphysik, Band 1 Mechanik - Relativität - Wärme.* 11. Auflage. New York, Berlin : de Gruyter, 1998

[CCDF00] Chaitin-Chaitin, F. (Hrsg.) ; Dallakyan, S. (Hrsg.) ; Fraysse, V. (Hrsg.) ; CERFACES (Veranst.): *GPS-Carrier Phase Ambiguity Resolution.* 2000

[CP06] Christian Pantle, Christian: Galileo. In: *Focus* (2006), Nr. 50, S. 176–177

[Dan96] Dana, Peter: *Reference Ellipsoids and Geodetic Datum Transformation Parameters.* Version: 1996. `http://www.colorado.edu/geography/gcraft/notes/datum/edlist.html`, Abruf: 24.03.2010

[Eck99] Eckstein, Peter P.: *Repetitorium Statistik.* 3. Auflage. Wiesbaden : Gabler, 1999

[EKD92] Eckey, Hans-Friedrich ; Kosfeld, Reinhold ; Dreger, Christian: *Statistik.* Wiesbaden : Gabler, 1992

[Emb06] Embacher, Franz: *Relativistische Korrekturen für GPS.* Version: 2006. `http://homepage.univie.ac.at/franz.embacher/rel.html`, Abruf: 12.03.2010

[Fay02] Fayngold, Moses: *Special Relativity.* 1. Edition. Weinheim : Wiley-WCH Verlag GmbH, 2002

[Fri90] Friess, Peter: *Kinematische Positionsbestimmung für die Aerotriangulation mit dem NAVSTAR Global Positioning System.* München : Bayerische Akademie der Wissenschaften, 1990

[GK09] Gessler, Ralf ; Krause, Thomas: *Wireless Netzwerke für den Nahbereich.* 1. Auflage. Wiesbaden : Vieweg + Teubner, 2009

[HWLC01] Hofmann-Wellenhof, B. ; Lichtenegger, H. ; Collins, J.: *GPS - Theory and Practice.* 5. Edition. Wien, New York : Springer, 2001

[HZ09] Helmholtz-Zentrum: *Helmholtz-Zentrum.* Version: 2009. `http://www.gfz-potsdam.de`, Abruf: 18.07.2009

[KH06] Kaplan, Elliott D. ; Hegarty, Christopher J.: *GPS-Principles and Applications.* Boston, London : Artech House, 2006

[Kre79] Kreyszig, Erwin: *Statistische Methoden und ihre Anwendungen.* 7. Auflage. Göttingen : Vandenhoeck & Rupprecht, 1979

[Kum00] Kumm, Werner: *GPS Global Positioning System.* Bielefeld : Delius Klasing Verlag, 2000

[KW09] Köhne, Anja ; Wössner, Michael: *GPS.* Version: 2009. `http://www.kowoma.de/gps/`, Abruf: 18.07.2009

[Leh03] Lehner, Franz: *Mobile und drahtlose Informationssysteme.* Heidelberg : Springer, 2003

[Lei97] Leinen, Stefan: *Hochpräzise Positionierung über grosse Entfernungen und in Echtzeit mit dem Global Positioning System.* 1. Auflage. München : C.H. Beck Verlag, 1997

[Man10] Mansfeld, Werner: *Satellitenortung und Navigation.* Wiesbaden : Vieweg + Teubner, 2010

[Puh01] Puhani, Josef: *Statistik, Einführung mit praktischen Beispielen.* 9. Auflage. Würzburg : Lexika Verlag, 2001

[Sch95] Schwarze, Volker S.: *Satellitengeodatische Positionierung in der Relativistischen Raum-Zeit*. 1. Auflage. München : C.H. Beck Verlag, 1995

[Sch03] Schiller, Jochen: *Mobilkommunikation*. 2. Auflage. München : Pearson Studium, 2003

[Sch06] Schönwiese, Christian-Dietrich: *Praktische Statistik*. 4. Auflage. Mörlenbach : Gebrüder Borntraeger, 2006

[SW07] Schmidt, Doris ; Winner, Hermann: Evaluation of the Potential of Global Navigation Satellite Systems for Advanced Driver Assistance Systems. (2007), Nr. 1084

[Tro08] Troitsch, Gunnar: Navigation. In: *Chip* (2008), Nr. 12, S. 64–65

[Ver09] Verkehrstechniken, Fachstelle der WSV f.: *DGPS Station Koblenz*. Version: 2009. `http://fvt.wsv.de/dgps/koblenz/index.html`, Abruf: 24.03.2010

[Yue09] Yuen, Ming F.: Dilution of Precision (DOP) Calculation for Mission Planning Purposes. (2009). `http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA497140&Location=U2&doc=GetTRDoc.pdf`