

Studiengang: Informatik
Betreuer: Daniel Herrscher
Prüfer: Prof. Rothermel

Beginn am: 15.05.2003
Beendet am: 14.11.2003

CR-Klassifikation: B.4.1, C.2.4, I.6.7

Studienarbeit-Nr. 1904

Emulation eines Positionierungsgeräts

Benedict Weisshaar

Fakultät Elektrotechnik, Informatik,
Informationstechnik
Universität Stuttgart
**Institut für Parallele und
Verteilte Systeme (IPVS)**
Universitätsstr. 38
70569 Stuttgart

Inhaltsverzeichnis

I. Einführung

1. Übersicht	4
2. Aufgabenstellung	4
3. Motivation	5
3.1 Positionsbestimmung	5
3.2 Netzwerkemulation	6
3.3 Emulation eines Positionierungsgerätes	6
4. Ziele	6

II. Verwandte Arbeit

1. Simulation eines Positionierungsgerätes	7
1.1 Rapidly Deployable Radio Network	7
2. Netzwerkemulation	8
2.1 Network Emulation Testbed	8
2.2 Utah Emulation Network	9
2.3 NRL Mobile Network Emulator	9
2.4 WINE GLASS	10
3. Positionierung in realen (physischen) Testbeds	10
3.1 APE	10
3.2 RUSH	11
4. NMEA-Generatoren	11
4.1 NMEAWiz	12
4.2 RECSIM III	12
4.3 NemaTalker	12
4.4 GPS Simulator	12
4.5 NMEA Server	12

III. Grundlagen

1. Positionsbestimmung	14
1.1 Ausprägung von Positionsdaten	14
1.2 Awareness	15
1.3 Verfahren	15
Tracking	
Positioning	
2. Positionierungstechnologien	16
2.1 Übersicht	16
2.2 Innerhalb von Gebäuden	17
2.2.1 Visuell	17
Visual Tags	
2.2.2 Infrarotbaken	17
Active Badge	
WIPS	
2.2.3 Ultraschall	18
Active Bat	
Cricket	
2.2.4 Funkbaken	19
SpotON	
RFid	
2.3 Netzwerkbasierend	19
2.3.1 Funknetze	20
WLAN	
2.3.2 Mobilfunk	20
GSM	
2.4 Satellitenbasierend	21
2.4.1 GPS	22
2.4.2 DGPS	25
2.4.3 IDGPS	25
2.4.4 WAAS	25
2.4.5 EGNOS	26
2.4.6 GALILEO	26
2.4.7 GLONASS	26
3. Das NMEA-Protokoll	26
3.1 Eigenschaften, Aufbau, Datenformat	26
3.2 SenderIDs	27
3.3 NachrichtIDs	28
4. Geographische Koordinaten	29
4.1 Die Erde als Kugel	29
4.2 Die Erde als Ellipsoid	29
4.3 Die Erde als Geoid	30
4.4 Das internationale geographische Koordinatensystem	30

IV. Virtuelles Positionierungsgerät

1. Architektur	31
1.1 Infrastruktur: NET	31
1.2 VgpsServer	32
1.3 VgpsClient	32
1.4 Gesamtbild	33
2. Funktionsweise und Bedienung	34
2.1 VgpsServer	34
2.2 VgpsClient	38
3. Implementierte NMEA-Nachrichten	41

V. Ausblick

1. Ausblick	50
--------------------	-----------

VI. Referenzen

1. Referenzen	51
----------------------	-----------

I. Einführung

1. Übersicht

Die vorliegende schriftliche Ausarbeitung der Studienarbeit mit dem Thema „Emulation eines Positionierungsgerätes“ gliedert sich in sechs Kapitel. Ein kurzer Überblick sowie ein Einstieg in die behandelte Thematik werden zunächst in Kapitel I gegeben – wesentliche Punkte sind die ursprüngliche Aufgabenstellung der Studienarbeit, sowie die Motivation zur Emulation eines Positionierungsgerätes. Kapitel II bietet einen Einblick in (Forschungs-) Projekte, tatsächlich genutzte Emulationsumgebungen und in bereits existierende Software, welche sich relevant und/oder verwandt zu dieser Studienarbeit zeigten. Interessant sind hier insbesondere das *Network Emulation Testbed* (NET), in welches das implementierte virtuelle Gerät integriert werden soll, sowie das *Rapidly Deployable Radio Network* (RDRN), das sich in einem anderen Rahmen ebenso vor die Aufgabe gestellt sieht, ein Positionierungsgerät emulieren zu müssen. Kapitel III veranschaulicht wesentliche Grundlagen, welche der Studienarbeit zugrunde gelegt werden: die Ausprägung von Positionsdaten werden hier behandelt, Begriffe wie „awareness“, „tracking“ und „positioning“ werden erläutert, verschiedene Positionierungstechnologien werden vorgestellt. So zum Beispiel die Positionierung in- und außerhalb von Gebäuden, oder Verfahren zur netzwerk- oder satellitengestützter Positionsbestimmung. Ebenso angesprochen werden in diesem Kapitel das NMEA-Protokoll, welches von allen GPS-Empfängern zur Informationsübertragung genutzt wird, und einige Grundlagen zu geographischen Koordinaten und dem Subjekt dieser, der Erdkörper. Kapitel IV spezifiziert die Funktionsweise sowie die Bedienung des implementierten virtuellen Gerätes, und beschreibt, welche NMEA-Nachrichten implementiert wurden. Kapitel V liefert einen kurzen Ausblick über die gewonnenen Erkenntnisse, und schließlich widmet sich Kapitel VI den herangezogenen Quellen und Autoren, welche maßgeblich für diese schriftliche Ausarbeitung herangezogen wurden.

2. Aufgabenstellung

"Das *Network Emulation Testbed* (NET) der Abteilung Verteilte Systeme ist ein flexibel vernetzter PC-Cluster mit 64 Knoten, in dem beliebige Netztopologien und deren Eigenschaften nachgebildet werden können. Damit bietet das Testbed eine ideale Umgebung zur vergleichenden Leistungsanalyse von verteilten Anwendungen und Netzprotokollen. Neben

statischen Netztopologien gewinnt die Emulation von Netzen mit mobilen Teilnehmern zunehmend an Bedeutung.

Moderne Anwendungen und Protokolle im Mobile-Computing-Bereich beziehen auch die Umgebungssituation in ihre Entscheidungen ein ('context-awareness'). Die wichtigste Kontextinformation ist hierbei die Position des Nutzers bzw. des Geräts. Um diese Information auch auf den emulierten mobilen Endgeräten im NET zur Verfügung stellen zu können, ist es nötig, das Verhalten eines Positionierungsgeräts realistisch nachzubilden.

Im Rahmen dieser Studienarbeit sind zunächst existierende Positionierungsgeräte hinsichtlich ihrer Anwendungsschnittstelle zu untersuchen. Daraufhin sind verschiedene Methoden zu erarbeiten, wie ein virtuelles Gerät eine typische Anwendungsschnittstelle realistisch nachbilden kann.

Die Zielsetzung hierbei ist, dass es für den Benutzer eines solchen Gerätes anhand des Verhaltens nicht unterscheidbar ist, ob es sich um ein echtes oder ein emuliertes Gerät handelt. Eine vielversprechende Methode soll ausgewählt und danach als Prototyp eines emulierten Positionierungsgeräts unter Linux realisiert werden. Das virtuelle Gerät ist in das NET-System zu integrieren und in einem Emulationsszenario mit mobilen Knoten zu testen. Dabei soll eine existierende Anwendung benutzt werden, die auf Positionsinformationen angewiesen ist.

Die Ergebnisse der Arbeit sind in einer schriftlichen Ausarbeitung zu dokumentieren und in einem Abschlussvortrag im Kolloquium der Abteilung Verteilte Systeme zu präsentieren."

3. Motivation

3.1 Positionsbestimmung

Eine effektive Positionsbestimmung ist in sehr vielen und unterschiedlichen Bereichen von großer Bedeutung. Besonders relevant ist sie auch für Anwendungen, die in einem mobilen Umfeld ihren Einsatz finden, wie zum Beispiel Navigations- und Steuerungssoftware in Flugzeugen, bei ortsabhängiger Werbung, oder in der Seefahrt.

3.2 Netzwerkemulation

Durch die zunehmende Verbreitung verteilter und vernetzter Systeme und moderner Kommunikationsprotokolle wächst der Bedarf nach Testmöglichkeiten und -Umgebungen in gleichem Maße. Meist wurden Simulationssysteme herangezogen, um diesen Anforderungen zu genügen; gelegentlich wurde auch unter realen Umständen getestet. Beide Ansätze haben Vorteile, aber auch Nachteile. Simulationen werden oft durch komplexe Szenarien eingeschränkt: die meisten Simulatoren funktionieren einwandfrei mit kleinen Modellen, darüber hinaus kommt es dann jedoch oft zu Problemen. Reale Testbeds wiederum haben den Nachteil, dass oft ein erheblicher Aufwand getrieben werden muss, um einen Test zu realisieren. Die Emulation versucht sich an einem Mittelweg. Zu den intelligenten Verfahren der Simulation mischen sich reale Komponenten, wodurch der Test realistischer wird und Parameter mit einbezogen werden, die in einer reinen Simulation möglicherweise keine Beachtung gefunden hätten.

3.3 Emulation eines Positionierungsgerätes

Mit der Verbreitung verteilter Systeme und Protokolle wächst auch zunehmend die Anzahl moderner Anwendungen im Mobile-Computing-Bereich. Daher besteht auch in diesem Bereich ein großer Bedarf an Testmöglichkeiten und Testumgebungen, insbesondere wird es in diesem Umfeld immer wichtiger, die Positionsdaten mobiler Anwender zu kennen. Allerdings gibt es gegenwärtig relativ wenig Systeme, die eine gute Testgrundlage bilden könnten. Häufig finden die Tests noch unter realen Bedingungen statt, was nicht nur einen erheblichen Aufwand bedeutet, sondern darüber hinaus noch ein beträchtlicher Kostenfaktor darstellen kann. Diesem Umstand soll mit der Suche nach sinnvollen Alternativen begegnet werden.

4. Ziele

Diese Arbeit zielt auf die Entwicklung einer sinnvollen Alternative zu bestehenden Ansätzen des Tests im Mobile-Computing-Bereich; dazu soll ein reales Positionierungsgerät emuliert und in eine komplexe Testumgebung integriert werden.

Gegenwärtig existiert eine große Menge verschiedener Positionierungsgeräte, welche ihren Einsatz in den unterschiedlichsten Bereichen finden (siehe III); die vorliegende Arbeit jedoch zielt auf die Emulation *eines* konkreten Positionierungsgerätes – aus der Vielfalt unterschiedlicher Geräteklassen wurde der GPS-Empfänger gewählt, aus dem wesentlichen Grund, dass GPS

die Positionsbestimmung eines Subjekts weltweit ermöglichen, und sich daher besonders für die Anforderungen des NETs an ein emuliertes Positionierungsgerät eignet.

In diesem Zuge ist ein virtuelles Gerät¹ gesucht:

- das das Verhalten eines realen GPS-Empfängers möglichst realistisch nachbildet (*Transparenz*), und
- das sinnvoll in das NET integriert werden kann.

II. Verwandte Arbeit

1. Simulation eines Positionierungsgerätes

1.1 Rapidly Deployable Radio Network

Das *Rapidly Deployable Radio Network* (RDRN) ist ein Projekt am Informations- und Telekommunikationstechnologiezentrum der Universität Kansas und hat das Ziel, Strukturen, Protokolle, Hard- und Software-Prototypen zu entwerfen, die den schnellen Einsatz eines Hochgeschwindigkeitsnetzes in Militär- und Katastrophengebieten ermöglichen soll, in welchen die Kommunikationsinfrastruktur nicht vorhanden ist, zerstört wurde, oder zusammengebrochen ist. RDRN ist ein schnurloses ATM-Netzwerk und besteht aus mobilen Kommunikationsknoten, angebracht entweder an fixen Positionen, oder an mobilen Plattformen wie z.B. Fahrzeuge und Hubschrauber. Während des Einsatzes greifen die Knoten auf GPS-Informationen zurück, um automatisch eine hoch skalierbare, fehlertolerante Netzwerkinfrastruktur herzustellen und zu konfigurieren (siehe [1]). Beim Entwurf von Anwendungen für das RDRN besteht die Anforderung, diese möglichst effektiv zu testen; da es sich aber als denkbar ungeeignet herausstellt, bei jedem Test ein komplexes RDRN einzusetzen, wird auf die Emulation zurückgegriffen. In der Realität bewegen und positionieren sich die Kommunikationsknoten mit Hilfe von Daten, die über eine serielle Schnittstelle von GPS-Empfängern zur Verfügung gestellt werden – während einer Emulation werden diese Daten von einem sogenannten „Emulationsmanager“ anhand einer Szenariodatei berechnet, und über ein UDP-Socket verschickt. Der hier verfolgte Ansatz hat eine hohe Ähnlichkeit zum Ansatz, dem diese Studienarbeit zugrunde liegt (siehe auch: II-2.1). Der wesentliche Unterschied liegt darin, dass in einer RDRN-Emulation der Emulationsmanager die

¹ Anmerkung: ist im Verlauf dieser schriftlichen Ausarbeitung von einem „virtuellen Gerät“ die Rede, so ist dies auf das im Rahmen dieser Studienarbeit implementierte virtuelle GPS-Gerät bezogen.

Berechnung und das Senden der GPS-Daten selbst übernimmt, wogegen diese Aufgabe von dem im Rahmen dieser Studienarbeit implementierten virtuellen Gerät übernommen werden kann. Demzufolge ist ein Einsatz des virtuellen Gerätes in der RDRN-Emulation denkbar. Allerdings implementiert das virtuelle Gerät eine Vielzahl verschiedener NMEA-Nachrichten (siehe III-3), währenddessen für RDRN nur wenige Daten benötigt werden, die gegebenenfalls bereits aus einer einzigen NMEA-Message gegeben sind – also besteht eventuell ein unnutzer Überfluss an Daten, die für RDRN nicht notwendig sind. Andererseits lässt sich das virtuelle Gerät so konfigurieren, dass nur die benötigten Nachrichten erzeugt werden (siehe IV-2).

Anmerkung: hier von einer Simulation die Rede, da bei RDRN die GPS-Daten den zu testenden Anwendungen über UDP-Pakete übergeben werden; beim virtuellen Gerät passiert dies über eine serielle Schnittstelle, wie bei einem tatsächlichen GPS-Empfänger – insbesondere also die Verwendung einer realen Komponente, im Gegensatz zu RDRN, wo dies nicht so ist – womit in diesem Fall von einer Emulation gesprochen wird. Zugegebenermaßen fällt hier die Unterscheidung zwischen Simulation und Emulation verschwindend gering aus.

2. Netzwerkemulation

2.1 Network Emulation Testbed

Das *Network Emulation Testbed* (NET), als Zielumgebung des implementierten virtuellen Geräts, verdient deshalb einen besonderen Augenmerk.

Aus der Erkenntnis, dass viele der existierenden Emulationsumgebungen entweder auf zentrale Echtzeitsimulationskomponenten basieren, oder sich auf die Emulation bestimmter Netzwerkeigenschaften beschränken, zielt das NET darauf, eine möglichst genaue Übereinstimmung zwischen dem Verhalten des emulierten Netzwerkes und dem Verhalten des realen Netzwerkes zu erreichen (welche Ansätze hierbei verwendet werden, siehe [2]). Mit NET ist ein Emulationswerkzeug gegeben, das vergleichende Leistungsanalysen von verteilten Anwendungen und Netzprotokollen ermöglicht. Da mobile und ortsbezogene Anwendungen immer mehr an Bedeutung gewinnen, ist es naheliegend, das NET so zu erweitern, dass die Emulation auch mit mobilen Teilnehmern ermöglicht wird. Als wesentliche Komponente kommt nun das in dieser Studienarbeit implementierte virtuelle Gerät hinzu. Angesteuert über sogenannte „Trace Files“ (siehe IV-2), stellt das virtuelle Gerät GPS-Daten – insbesondere Positions-Informationen – auf den emulierten mobilen

Endgeräten zur Verfügung. Damit können komplexe (mobile) Szenarien inszeniert, getestet, und ausgewertet werden.

2.2 Utah Emulation Network

Als verwandtes Projekt zum NET verfolgt das *Utah Emulation Network* (Netbed) den Ansatz, reale Netzwerke, Netzwerksimulatoren, und Netzwerkemulatoren in einem gemeinsamen, transparenten Framework zu integrieren, und entsprechende Dienste zum Test und Analyse verteilter Systeme und Netzprotokolle bereitzustellen. Selbstgesteckte Ziele sind Bedienungsfreundlichkeit, Bedienungsvielfältigkeit, und möglichst realitätsnahes Verhalten bei Emulationen (siehe [3]). Bei der Vielfalt an Konfigurationsmöglichkeiten und Optionen, die das Framework bietet, überrascht es, dass keine zusätzliche, auf ortsbezogene und mobile Anwendungen zugeschnittene Dienste angeboten werden. Eine Integration des virtuellen Gerätes in das Framework des Netbeds erscheint möglich, somit könnten bei Emulationsläufen dann GPS-Daten zur Verfügung gestellt werden.

2.3 NRL Mobile Network Emulator

Aus der Erkenntnis, dass verteilte Anwendungen, verteilte Systeme, und Rechnernetze ein immer größeren Bestandteil der (künftigen) Informationstechnologie bilden, und sich der Bereich mobiler wireless Netzwerke fortwährend entfaltet, bemüht sich das *Naval Research Laboratory* (NRL) der United States Navy um eine preiswerte und flexible mobile wireless Test-Umgebung, der sogenannte *Mobile Network Emulator* (MNE). Ziel ist es, Forschern und Entwicklern die Möglichkeit zu bieten, mobile Netzwerk-Technologie und komplexe Szenarien möglichst flexibel, bequem, und preiswert zu testen (siehe [3]).

MNE verfolgt einen anderen Ansatz als das im Rahmen der Studienarbeit implementierte Gerät: Bewegungsabläufe werden durch einen Software-Prozess simuliert, der während einer Emulation Positionsinformationen (diese beschränken sich auf GPS-Längen- und Breitengradangaben) in den gemeinsamen Hauptspeicher schreibt. Emulationsknoten greifen dann auf diese dynamische Information zurück, um ihre eigene Position zu bestimmen, und um zu errechnen ob sie – ausgehend von ihrer eigenen Position und der Position der anderen Knoten, die sie ebenso aus dem gemeinsamen Speicher lesen können – in der Lage sind, mit anderen Knoten effektiv zu kommunizieren, oder ob dies Aufgrund störender Faktoren (z.B. zu große Entfernungen) in der Realität nicht den Tatsachen entsprechen würde. Anders als bei diesem Ansatz, in dem die Positionsinformationen einfach in den gemeinsamen Speicher geschrieben werden, stellt das virtuelle Gerät diese Informationen realistischer zur Verfügung – entweder über eine serielle

Schnittstelle, wie es bei einem tatsächlichen GPS-Empfänger der Fall ist, oder über eine Datei.

2.4 WINE GLASS

Im Rahmen des fünften FTE-Programms (Forschung und technologische Entwicklung) der Europäischen Union findet sich das Projekt *Wireless IP Network as a Generic platform for Location Aware Service Support* wieder, kurz WINE GLASS. WINE GLASS bemüht sich um ein wireless Internet Testbed mit dem Ziel, eine Umgebung für Forschung, Entwicklung, Test, Integration, Validierung und Evaluation von orts- und QoS-bezogenen Anwendungen (unter Anderem) zu erschaffen.

Auch hier spielen Umgebungssituationen und -Informationen eine wichtige Rolle: ein sogenannter "Location Server" hält eine Positions-Datenbank, welche die Abbildung aller IP-Subnetze auf eine bestimmte geographische Position ermöglicht. Knoten können nun „network positioning queries“ senden – der Location Server ermittelt über die IP-Adresse des Knotens das entsprechende IP-Subnetz, mappt dieses anhand seiner Datenbank auf die entsprechende geographische Position, und antwortet dem anfragenden Knoten mit GPS-Daten (siehe [4]).

Insbesondere erscheint dieser Ansatz, anders als bei dem virtuellen Gerät, in erster Sicht etwas ungenau: zwei Knoten in demselben IP-Subnetz befinden sich demzufolge an derselben geographischen Position.

3. Positionierung in realen (physischen) Testbeds

3.1 APE

APE ist die Abkürzung für *Ad-hoc Protocol Evaluation*, dem entsprechend stellt das APE Testbed eine Testumgebung für wireless multi-hop ad hoc Netzprotokolle dar. Das APE erscheint an dieser Stelle erwähnenswert, weil es selbst ein Mobilitätsbegriff – und damit eine rudimentäre relative Positionserkennung – definiert, und sich explizit gegen GPS als Positionsbestimmung erklärt. Die „virtual Mobility“ basiert auf SNR (signal-to-noise ratio): physische Distanz (und damit die etwaige Position) zwischen Knoten wird anhand der Fluktuationen der Signalstärken ermittelt. Diese berechnete Distanz entspricht dann nicht unbedingt der realen Distanz zwischen den Knoten, da Reflexionen auftreten oder Objekte sich in die Sichtlinie zweier Knoten schieben können. Trotz allem wird kein GPS verwendet, da GPS-Positionierung innerhalb von Gebäuden praktisch nicht möglich ist. Weitere Argumente, die bei APE gegen GPS sprechen, sind die höheren Kosten und die erhöhte Komplexität beim Einsatz.

Weiteres über APE siehe [5].

3.2 RUSH

RUSH (*Rice University SHuttlebus project*) motiviert sich aus der Erkenntnis, dass auf Simulation basierende Testumgebungen Vor-, aber auch Nachteile haben. So wird bei einer Simulation beispielsweise häufig nur eine sehr kleine Fläche betrachtet, in der sich meist auch nur eine sehr kleine Anzahl von Knoten bewegen. Auch werden bei Simulationen selten reale wireless-Verbindungen behandelt, was zu Leistungseinbußen führen kann, sobald eine mobile Umgebung in Frage kommt. Insbesondere diesen Nachteilen möchte RUSH Alternativen gegenüber stellen: das RUSH-Testbed ist eine Testumgebung für GPS-bezogene und für nicht-GPS-bezogene ad-hoc Netzwerkanwendungen (siehe [6]).

Das RUSH-Testbed besteht aus sogenannten „WINDs“ (*Wireless Integrated Network Device*), welche entweder als eine GPS-bezogene fixe Basisstation, oder als ein mobiler Knoten fungieren können. Insbesondere verfügt jedes WIND über ein GPS-Empfängermodul – über einen eigenen Gerätetreiber werden die NMEA-Daten dann formatiert und die Positionsdaten dem WIND zur Verfügung gestellt.

4. NMEA-Generatoren

Im Wesentlichen für Testzwecke und für Simulationen motiviert, gibt es eine Reihe kommerzieller Anwendungen, welche NMEA-Nachrichten generieren – einige davon bieten eine ähnliche Funktionalität wie das virtuelle Gerät. Der wesentliche Unterschied liegt darin, dass diese Anwendungen ihre GPS-Daten entweder in einem Simulationsmodus erzeugen, oder ausgehend von früher aufgezeichneten GPS-Informationen, während das virtuelle Gerät durch relative Koordinaten ansteuerbar ist, und aus diesen die entsprechende NMEA-Nachrichten generiert – somit entsteht eine kontrollierte, (durch ein Szenario) gesteuerte Erzeugung dieser Nachrichten. Darüber hinaus unterscheidet sich die Architektur des virtuellen Gerätes – das aus einem Server und (potentiell unendlich vielen) Clients besteht – von der meist Standalone-Architektur der erwähnten Anwendungen.

In diesem Abschnitt wird Software angesprochen, die wegen bestimmter Eigenschaften besonders interessant erscheint. Eine vollständige Aufzählung ist hier jedoch nicht gegeben.

Anmerkung: die Unterscheidung zwischen GPS und NMEA mag an dieser Stelle schwer fallen. GPS – Global Positioning System – wird von GPS-

Empfängern zur Positionsbestimmung benutzt. Das NMEA-Protokoll wird von den Empfängern herangezogen, um die berechneten Daten zur Verfügung zu stellen. Kapitel III behandelt das NMEA-Protokoll im Detail.

4.1 NMEAWiz

NMEAWiz ist ein vielseitiger NMEA-Generator für die Windows-Plattformen. Über eine graphische Benutzeroberfläche werden Parameter wie Richtung, Geschwindigkeit, Position, oder Höhe angegeben, und NMEAWiz stellt – ausgehend von diesen Daten – entsprechende NMEA-Nachrichten über eine vorher definierte serielle Schnittstelle zur Verfügung. NMEAWiz kann mehrfach instanziiert werden, womit es möglich wird, parallel über mehrere serielle Schnittstellen NMEA-Daten zur Verfügung zu stellen. Es ist möglich, die Ausgaberate des NMEAWiz zu regulieren, damit die Reaktion von NMEA-abhängigen Anwendungen auf Datenüber- oder Datenunterversorgung sichtbar wird. Außerdem bietet NMEAWiz noch die Möglichkeit, Ein- oder Ausgaben zur späteren Weiterverwendung aufzuzeichnen. Siehe auch [7].

4.2 RECSIM III

RECSIM bietet eine weitgehend ähnliche Funktionalität wie NMEAWiz. Das Besondere jedoch an RECSIM ist ein NMEA-Parser, den man so konfigurieren kann, dass er alle NMEA-Nachrichten senden kann – selbst sogenannte „proprietäre“ und selbstdefinierte Nachrichten. Siehe auch [8].

4.3 NemaTalker

Wie die oben besprochenen NMEA-Generatoren sendet NemaTalker NMEA-Nachrichten über eine serielle Schnittstelle. Darüber hinaus ist NemaTalker ab Version 1.3.0 sogar in der Lage, diese Daten über das UDP-Protokoll an einen Client zu senden. Siehe auch [9].

4.4 GPS Simulator

GPS Simulator kommt aus dem selben Hause wie NemaTalker, und kann als eine eingeschränkte Version von diesem gesehen werden. Die Anzahl der unterstützten NMEA-Nachrichten sind wesentlich geringer, und eine Versendung dieser über UDP ist nicht möglich. Siehe auch [10].

4.5 NMEA Server

NMEA Server ist freie Software, und wurde für die Linux Plattform entwickelt. Motiviert aus der Tatsache, dass PCs über eine beschränkte Anzahl serieller Schnittstellen verfügen, zielt NMEA Server im Wesentlichen darauf, von einem GPS-Empfänger empfangene Nachrichten mehreren Clients über IP zur Verfügung zu stellen. NMEA Server speichert empfangene Nachrichten gegebenenfalls zur späteren Wiedergabe ab, und ermöglicht es seinen Clients

gezielt jene Nachrichten zu abonnieren, an welchen Interesse besteht, und andere zu verwerfen. Siehe auch [11].
Genau wie NMEA Server bietet auch das virtuelle Gerät die Möglichkeit, bestimmte Nachrichten zu abonnieren.

III. Grundlagen

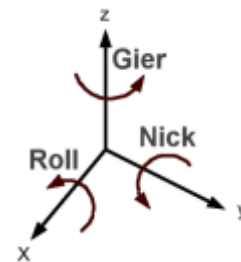
1. Positionsbestimmung

Die Notwendigkeit einer effektiven Positionsbestimmung liegt auf der Hand. Schon die alten Seefahrer sahen sich auf offenem Ozean mit dem Problem konfrontiert, ihre Position zu bestimmen, und behelfen sich mit der Astronomie und dem Kompass. Gegenwärtig ist die Navigation noch immer ein Thema; doch darüber hinaus existiert zwischenzeitlich eine Vielzahl an Bereichen, wo die Positionsbestimmung ebenfalls nicht mehr wegzudenken ist.

1.1 Ausprägung von Positionsdaten

Je nach Kontext können Positionsdaten sehr unterschiedliche Ausprägungen vorweisen; beispielsweise können weltweit eindeutige geographische Koordinaten gefordert sein, ausgeprägt durch Längen- und Breitengrad, und eventuell eine Höhenangabe (siehe III-4).

Möglicherweise ist die Anforderung gegeben, die Geschwindigkeit eines Objektes an einer bestimmten Position ermitteln zu müssen, häufig ist auch die Orientierung im Raum (Roll-Nick-Gier-Winkel) gesucht. Andererseits können oft relative Positionen schon den Anforderungen genügen, zum Beispiel X,Y-Koordinaten



relativ zu einer bestimmten geographischen Position, wie es bei dem virtuellen Gerät der Fall ist – die relativen X,Y-Koordinaten aus einem Emulationsszenario werden durch das virtuelle Gerät auf konkrete geographische Koordinaten abgebildet (siehe IV-1). In manchen Fällen kommt es gar vor, dass kein Interesse an der geographischen Position besteht, sondern nur an der Semantik der Position – ein dramatisches Beispiel: jeder Mensch wird sich dafür interessieren, ob er sich gerade in einem Minenfeld bewegt, oder nicht – die genaue Position in dem Minenfeld ist unwesentlich. Ein harmloseres Beispiel: die meisten für den Menschen lesbaren Verzeichnisse benutzen semantische Positionen wie Straßennamen, Hausnummern, Postleitzahlen, usw..

Aus erster Sicht lassen sich Positionsdaten also grob durch

- weltweit eindeutige geographische Koordinaten,
- relative Koordinaten, oder durch
- semantische Positionen

ausprägen. Je nach Umfeld, Anwendungsbereich und Anforderungen fallen die Positionsausprägungen sehr unterschiedlich aus.

1.2 Awareness

Ehe an dieser Stelle die Semantik des Begriffes „Awareness“ – zu Deutsch „Bewusstsein“ – beschrieben wird, muss beachtet werden, dass die Semantik eines Begriffes je nach Kontext variiert, in welchem der Begriff verwendet wird.

Ist im Mobile-Computing-Bereich und in dieser Studienarbeit von „Awareness“ die Rede, so geht es um den Einbezug ihrer Umgebungssituation durch Anwendungen und Protokolle. Es lassen sich drei Arten von Awareness unterscheiden:

- *Location Awareness*: die Anwendung oder das Protokoll ist sich ihrer aktuellen Position bewusst.
- *Situation Awareness*: die Anwendung oder das Protokoll ist sich ihrer aktuellen Situation bewusst: es besteht Bewusstsein nicht nur über die aktuelle Position, sondern auch über die Umgebungsbedingungen.
- *Context Awareness*: ist der Oberbegriff für jede Art von Kontext.

Im Rahmen dieser Studienarbeit ist die Position die wichtigste Kontext-Information.

1.3 Verfahren

Trotz aller Vielfalt gegenwärtiger Positionierungstechnologien (siehe III-2), liegen diesen einige wenige Basistechniken zugrunde. Diese lauten:

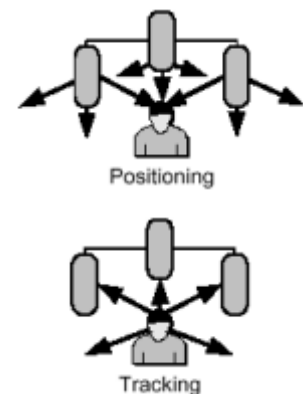
- *Angle of Arrival (AOA)*: bei diesem Ansatz wird die Richtung ermittelt, aus welcher ein Signal kommt. Dazu werden Antennen mit Richtungs-Charakteristik benötigt: entweder sind mehrere Antennen oder mechanische Drehung notwendig. Dies wird beispielsweise in Radaranlagen verwendet.
- *Cell of Origin (COO)*: zur Positionsbestimmung werden sogenannte „Zellen“ benutzt. Diese Zellen werden von Basisstationen und ihren Signalen aufgespannt, meist ist die Reichweite der Signale beschränkt – ist dies nicht der Fall, können die Signalstärken gemessen werden, siehe unten. Beim Empfang eines Signals befindet sich der Empfänger demnach innerhalb der Zelle einer Basisstation, deren Position bekannt ist.
- *Messung der Signalstärke*: Signalstärken werden mit der Entfernung zum Sender abgeschwächt, somit lässt sich die Stärke als ein Maß für den Abstand ansehen. Ein wesentliches Problem ist die Tatsache, dass

ein Signal nicht nur von dem Abstand zur Basisstation abhängig ist, sondern auch von einer Vielfalt von Umgebungsbedingungen. Vergleiche mit II-3.1 und siehe III-2.2.4.

- *Time of Arrival (TOA)*: dieser Ansatz basiert auf der Tatsache, dass Signale eine endliche Ausbreitungsgeschwindigkeit vorweisen. Mit Hilfe sehr präziser Uhren kann dann die Laufzeit von Signalen verschiedener Signalquellen gemessen, und daraus die Position ermittelt werden. GPS verwendet diesen Ansatz. Siehe III-2.4.1.

Ein System zur Positionsbestimmung lässt sich weiter zu einer der beiden folgenden Kategorien zählen:

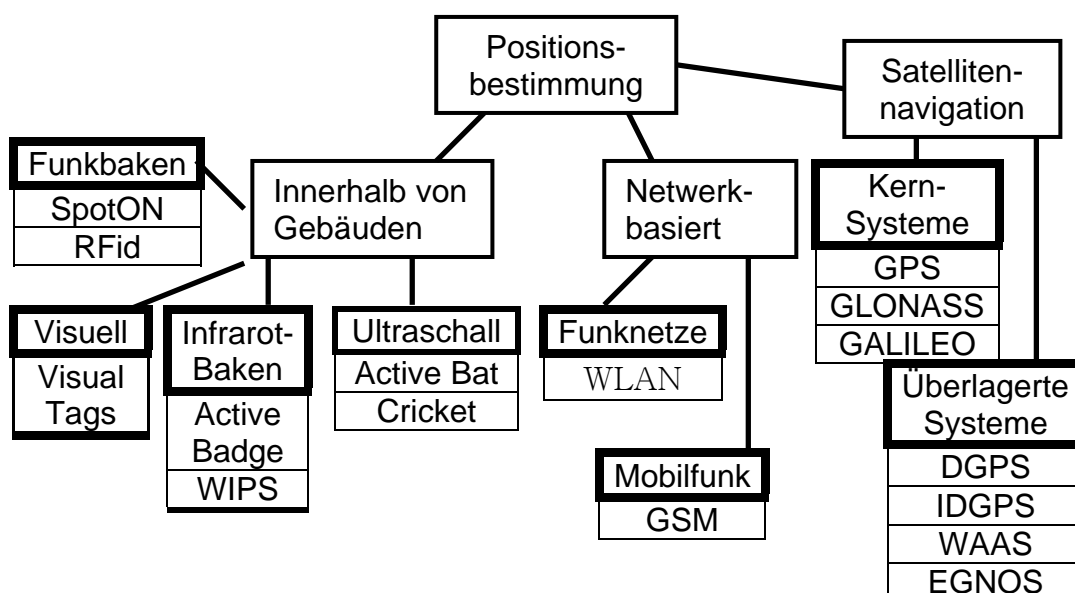
- *Positioning*: ausgehend von empfangenen Signalen wird die Position von dem (sich bewegenden) Objekt selbst ermittelt. Die Positionsinformation fällt an dem Objekt selbst an.
- *Tracking*: das (sich bewegende) Objekt trägt eine Marke, über die es von dem positionsbestimmenden Netzwerk erkannt wird. Die Positionsinformation fällt an Netzwerk an.



2. Positionierungstechnologien

2.1 Übersicht

Die große Vielfalt unterschiedlicher Positionierungstechnologien findet ihre Ursache in einer noch größeren Vielfalt unterschiedlicher Anforderungen und Anwendungsgebieten. Viele haben noch immer den Status unausgereifter Forschungsprojekte, während andere eine große Verbreitung gefunden haben.



2.2 Innerhalb von Gebäuden

Der Grund für die vielen Ansätze zur Positionsbestimmung innerhalb von Gebäuden, auch als Indoor-Positioning bezeichnet, liegt im Wesentlichen darin, dass Satelliten innerhalb von Gebäuden praktisch nicht empfangen werden können. Zu den Anforderungen, die an ein Indoorpositionierungssystem gestellt werden, zählen:

- Installations- und Verwaltungskosten
- Installationsaufwand
- Genauigkeit der Positionsbestimmung
- Verfügbarkeit

Die nachfolgend vorgestellte Positionierungstechnologien weisen eine unterschiedliche Eignung zu diesen Anforderungen auf.

2.2.1 Visuell

Es existieren verschiedene Verfahren, die Position eines mobilen Objekts anhand der Farbe und der Form von Räumen zu bestimmen – Visuelle Positionsbestimmung erfolgt mit Bilderkennungssystemen. Es können auch spezielle Etiketten, sogenannte „Visual Tags“ zu Einsatz kommen. Visual Tags – zum Beispiel Muster aus roten und grünen Quadraten, oder Kreise mit farbigen Segmenten – lassen sich besonders gut aus Kamerabildern herausrechnen und können verschiedene IDs kodieren. Objekte, die positioniert werden sollen, werden mit einem dieser Tags gekennzeichnet; dann können, unter anderem, Entfernungen zur Kamera und die Lage im Raum berechnet werden. Allerdings benötigt die Bildverarbeitung eine hohe Rechenleistung, so dass eine praktische Anwendung selten in Frage kommt.

2.2.2 Infrarot-Baken

Die am weit verbreitetsten Positionierungssysteme im Indoor-Bereich arbeiten mit Infrarot. Die Signale werden teilweise ähnlich wie bei Fernbedienungen, teilweise über die IrDA-Technik übertragen – charakteristisch ist hier deshalb eine Reichweite von wenigen Metern. Im Allgemeinen werden die Signale von Wänden reflektiert, so dass eine direkte Ausrichtung zwischen Sender und Empfänger oft nicht notwendig ist.

Active Badges

Zur Positionsbestimmung mit Active Badges trägt jedes zu positionierende Objekt einen kleinen Infrarotsender, genannt „Active Badge“. Diese sendet regelmäßig ein Infrarotsignal mit einer Objektkennung (ID) aus, welches von

stationären Infrarotsensoren empfangen wird. Die Sensoren sind über ein Netzwerk mit einem sogenannten „Location Server“ verbunden, an dem sämtliche Informationen anfallen und die Position berechnet wird. Im Gegensatz zu den ersten Varianten können die Badges bei den meisten neueren Systemen auch Daten empfangen; dadurch kann die Position eines Objektes auch dem Objekt selbst bewusst gemacht werden.

WIPS

Im Gegensatz zu den Active Badges, ein *Tracking*-System, ist WIPS ein *Positioning*-System. Die Infrarotsignale werden von der stationären Infrastruktur gesendet, und von dem zu positionierendem Objekt empfangen. Daraus berechnet das Objekt dann selbst seine Position. Falls die Positionsdaten zentral abrufbar sein sollen, muss jede Badge ihre Position an den Location Server übertragen, der diese Daten wiederum aufbereiten und zurück an die Badge senden kann. Da die Information am Objekt anfällt, müssen die sendende Baken nicht untereinander vernetzt werden – ein bedeutender Vorteil bei der Installation und Verwaltung des Systems.

2.2.3 Ultraschall

Active Bat

Bei der Positionsbestimmung durch ein Active Bat System wird die Position durch einen Location Server ermittelt, der mit einem Sensornetzwerk vernetzt ist. Auf Anforderung sendet das Bat per Funk ein Ultraschallsignal aus, das von den Sensoren empfangen wird. Der Server ermittelt dann anhand der unterschiedlichen Signallaufzeiten, wie weit das Bat von jedem Sensor entfernt ist, und errechnet daraus die Position. Charakteristisch für dieses System ist eine sehr exakte Positionsbestimmung innerhalb eines Raumes mit einer Genauigkeit von bis zu wenigen Zentimetern; allerdings erfordert es ein gitterförmiges Netz von Ultraschallsensoren mit einem Abstand von circa 1,2 Meter, was nicht nur die Anschaffungskosten in die Höhe treibt, sondern auch ein bedeutender Installations- und Wartungsaufwand darstellt – vermutlich der Grund, weshalb das System bisher keine allzu große Verbreitung gefunden hat.

Cricket

Cricket basiert auch auf Ultraschall, allerdings werden die Ultraschallsignale von der stationären Ultraschallinfrastruktur gesendet, und von dem Objekt, dessen Position bestimmt werden soll, empfangen. Zur Zeitmessung wird parallel ein Funkimpuls gesendet. Es handelt sich hier um ein *Positioning*-Ansatz. Die Entwicklung von Cricket zielt primär jedoch nicht auf eine genaue

Positionsbestimmung, vielmehr sollen dem Objekt Dienstinformationen über den Funkimpuls zur Verfügung gestellt werden.

2.2.4 Funkbaken

Funkbasierte Systeme sind, nach den Infrarotsystemen, die zweithäufigst verbreitete Variante von Indoor-Positionierungssystemen. Weil Funkwellen durch (dünne) Wände dringen können, wird eine höhere Reichweite erreicht. Damit ist eine weitmaschigere Sensorendichte gegeben, als beispielsweise bei Infrarotsystemen – allerdings sind Funk-Komponenten teurer als Infrarot-Komponenten. Darüber hinaus können Störungen durch andere Funksysteme auftreten.

SpotON

SpotON folgt dem *Tracking*-Prinzip. Das Objekt, dessen Position bestimmt werden soll, wird mit einer Funk-Badge bestückt. Das Funksignal, das von der Badge gesendet wird, wird von stationären Funksensoren empfangen; die Position des Objekts wird dann wiederum von einem Location Server bestimmt, der mit dem Funksensor-Netzwerk vernetzt ist. SpotON benutzt die jeweilige Signalstärke des empfangenen Signals als Maß für den Abstand. Die Signalstärke wird allerdings nicht nur durch den Abstand, sondern auch durch Hindernisse, insbesondere Wände, beeinflusst.

Vergleiche mit II-3.1 und III-1.3.

RFid

RFid steht für „Radiofrequenz-Indikatoren“. Dabei handelt es sich um kleine Systeme mit Prozessor, Speicher und Antenne, aber ohne Stromversorgung – die Versorgungsenergie wird aus dem Funksignal selbst gewonnen. Über die Funksignale können Daten in den Speicher eingelesen, oder simple Befehle zum Senden von Dateninhalten ausgeführt werden. Dabei können Entfernungen von etwa 1,5 Meter überbrückt werden. RFids mit integrierter Stromversorgung werden „active Tags“ genannt: die Reichweite, der Speicher, die Größe und das Gewicht betragen dann mehr als bei den funkversorgenen passive Tags.

RFids werden gegenwärtig hauptsächlich zur Überwachung des Transports von Objekten während der Produktion eingesetzt.

2.3 Netzwerkbasiert

Der Aufbau eines neuen Positionierungssystems impliziert meist hohe Kosten und einen großen Aufwand. Demzufolge ist es naheliegend, anstatt dessen bereits bestehende drahtlose Infrastrukturen auch für die Positionsbestimmung zu benutzen. Und tatsächlich eignen sich viele dieser Infrastrukturen sehr gut für dies Vorhaben. In dem man Basistechniken

überlagert (siehe III-1.3), kann die Positionsbestimmung in manchen Fällen verfeinert werden.

2.3.1 Funknetze

WLAN

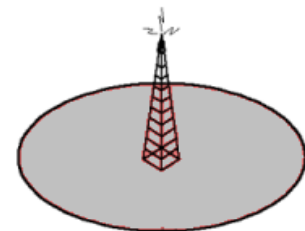
Durch die zunehmende Verbreitung von WLAN-Geräten tritt die auf den Funksignalen dieser Geräte basierende Positionsbestimmung immer mehr in der Vordergrund. Zunächst wird eine Liste möglicher Orte bestimmt; während einer sogenannten „Trainingsphase“ wird an jedem dieser Orte die Signalstärke der Basisstationen gemessen, und zusammen mit dem entsprechenden Ort in einer Tabelle eingetragen. Die während der Betriebsphase gemessenen Signalstärken werden dann mit dieser Tabelle verglichen; der Ort, auf den die gemessenen Signalstärken am ehesten zutreffen, ergibt die aktuelle Position. Der wesentliche Nachteil dieses Systems ist die aufwendige Trainingsphase: je genauer und detaillierter sie abläuft, desto genauer ist später die Positionsbestimmung – es ergibt sich ein beträchtlicher Aufwand. Kommen neue Basisstationen dazu, werden Basisstationen bewegt, bauliche Veränderungen vorgenommen, oder treten sonstige erhebliche Änderungen der Umgebungsbedingungen ein, so muss die Trainingsphase wiederholt werden.

2.3.2 Mobilfunk

GSM

Die GSM-Kommunikationsinfrastruktur kann direkt über die *cell of origin* Technik (siehe III-1.3) zur Positionsbestimmung benutzt werden. Bucht sich ein Teilnehmer in das Netz ein, so registriert ein „Zellen-Server“, in welcher Zelle sich dieser befindet; diese Zelleninformation kann später – üblicherweise dann, wenn eine Verbindung zu dem Teilnehmer aufgebaut werden soll – abgerufen werden.

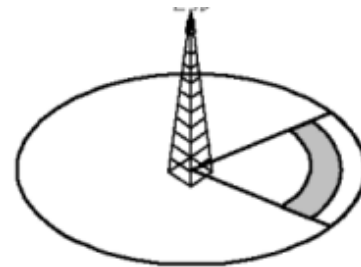
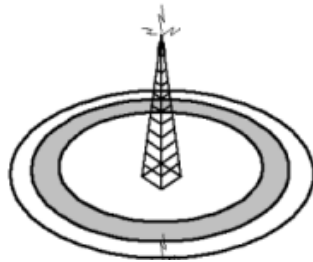
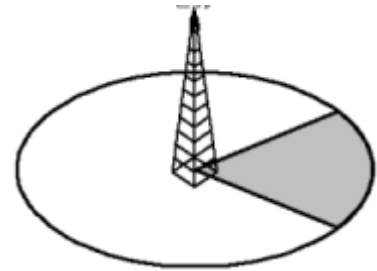
Positionsbestimmung: über den Zellen-Server wird die *Cell Global Identity* (CGI) der Zelle ermittelt, in der sich der Teilnehmer befindet, und damit grob – der Zellendurchmesser liegt üblicherweise zwischen 2 und 35 Km – der Standort bestimmt.



Falls Sektor-Antennen verwendet werden, wird der Bereich auf ein Winkelsegment (180°, 120° oder 90°) eingeschränkt:

Die Positionsbestimmung lässt sich durch die *Timing Advance* Technik noch weiter verfeinern: über den Zeitversatz zwischen der Basisstation und

dem Teilnehmer kann die Entfernung zur Basisstation in Stufen von circa 550 Meter genau berechnet werden:



Cell Global Identity + Timing Advance Sektorantenne + *Timing Advance*

Wireless Assisted GPS: gelegentlich sind in neueren Mobiltelefonen bereits GPS-Empfänger integriert. Durch die Kombination von Zellen- und GPS-Informationen wird eine wesentlich genauere Positionsbestimmung erreicht. Dazu muss die bestehende Infrastruktur erweitert werden: alle 200 Kilometer wird ein Location Server hinzugefügt. Zur Positionsbestimmung werden von dem Mobiltelefon dann GPS-Daten empfangen, aufbereitet und an den nächsten Location Server gesendet; die Position wird dann von dem Location Server anhand der GPS-Daten des Mobiltelefons, der Zellinformationen der Zelle, in der sich das Telefon befindet, und aus GPS-Daten eines Referenzpunktes (vergleiche differentielles GPS, III-2.4.2) ermittelt, und zurück an das Mobiltelefon gesandt.

2.4 Satellitenbasiert

Der Wunsch nach einem weltweit einsetzbarem und hoch genauem Positionierungssystem ließ die Idee der Satellitennavigation schon in den 60er Jahren aufkommen, und motivierte wesentlich die Entwicklung gegenwärtiger Satellitennavigationssysteme. Solche Systeme werden von Umweltbedingungen zwar kaum beeinflusst, der ausschlaggebende Nachteil jedoch ist die Tatsache, dass der Satellitenempfang innerhalb von Gebäuden üblicherweise nicht möglich ist, wodurch hier andere Technologien notwendig werden.

2.4.1 GPS

Das *Navigation System with Timing and Ranging* – *Global Positioning System*, oder einfach nur „GPS“, wurde 1970 von dem US-Verteidigungsministerium konzipiert und 25 Jahre später schließlich voll einsatzfähig. Es besteht aus 24 Satelliten (und bis zu 4 Reservesatelliten) auf 6 Bahnen, die auf einer Höhe von circa 20200 Kilometer die Erde umkreisen, und circa 12 Stunden pro Umlauf benötigen; jeder Satellit verfügt über eine Atomuhr – die Uhren der verschiedenen Satelliten werden ständig synchronisiert und überwacht. Mindestens 4, höchstens 8 dieser Satelliten sind von jedem Punkt der Erde gleichzeitig sichtbar.



Die Satelliten bilden das sogenannte „Raumsegment“, eines der insgesamt 3 Segmente, die das GPS bilden. Die weiteren Segmente sind das Benutzersegment, das aus allen Nutzern des Systems besteht, und das Kontrollsegment, das zuständig ist für die Synchronisation der Atomuhren, für die Überwachung der Satellitenbahnen, und für das Einrichten neuer und Ausmustern alter Satelliten.

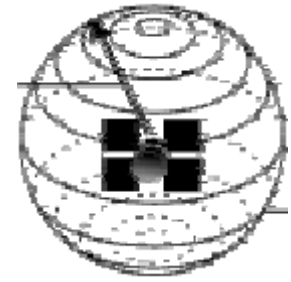
Positionsbestimmung mit GPS:

Jeder Satellit sendet kontinuierlich ein eigenes, spezifisches Signal aus. Dieses Signal wird „Pseudo Random Code“ oder auch „Pseudo Random Noise“ genannt, überträgt mit 50 Bit/s Uhrzeit und Bahndaten, und wird zur Ermittlung der Signallaufzeit zwischen dem Satelliten und dem Empfänger benutzt:

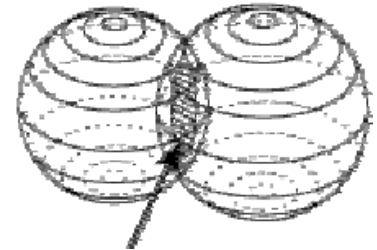


Satellit und Empfänger generieren denselben PRC zur gleichen Zeit. Durch den Vergleich des lokal generierten mit dem empfangenen Signal wird die Phasenverschiebung zwischen diesen errechnet, und damit die Signallaufzeit. Anhand der Signallaufzeiten und der Kenntnis der genauen Position der Satelliten kann der Abstand zwischen diesen und dem Empfänger bestimmt werden.

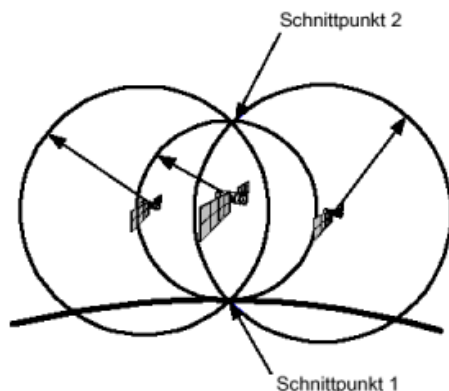
Die Positionsbestimmung erfolgt nun mit Hilfe der ermessenen Abständen. Ist nur ein einziger Abstand bekannt, so befindet sich der Empfänger auf der Oberfläche einer Kugel, dessen Mittelpunkt ein Satellit, und dessen Radius der gemessene Abstand von diesem zum Empfänger ist:



Ist ein weiterer Abstand zu einem zweiten Satelliten gegeben, befindet sich der Empfänger irgendwo auf der Schnittfläche der Kugeloberflächen der beiden aufgespannten Kugeln:



Mit dem Abstand zu einem dritten Satellit befindet sich der Empfänger schließlich genau auf einem Schnittpunkt der drei Kugeloberflächen:



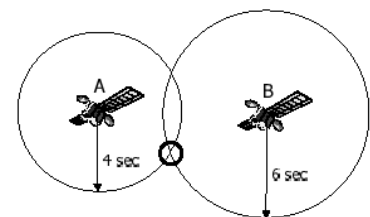
Einer der Schnittpunkte liegt in dem Weltall und kommt damit für die Position nicht in Frage; der zweite Schnittpunkt ergibt die Position.

Das schwierigste Problem bei GPS ist die exakte Zeitsynchronisation: eine Abweichung von 1 μs entspricht circa 300 Meter! Das Problem ist auf Satellitenseite unwesentlich, da dort Atomuhren zur Verfügung stehen; auf der Empfängerseite jedoch steht überwiegend

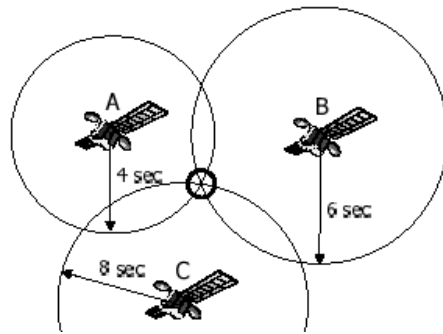
keine solche Zeitmessung zur Verfügung. Unter Hinzunahme des Abstandes zu einem vierten Satelliten lässt sich dieses Problem lösen.

Der Einfachheit halber soll das Verfahren an dieser Stelle anhand eines Beispiels im Zweidimensionalen erläutert werden – im Dreidimensionalen wird analog vorgegangen.

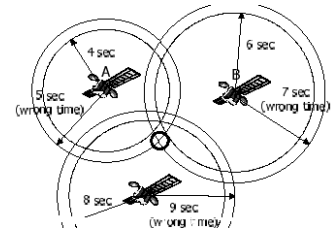
Eine zweidimensionale Positionsbestimmung ergäbe zunächst etwa folgendes Bild:



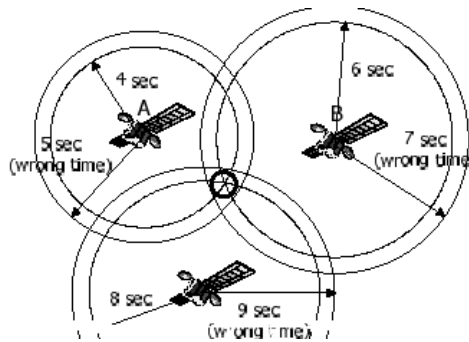
Falls eine korrekte Uhrensynchronisation vorliegt, würde durch die Messung des Abstandes zu einem dritten (im Dreidimensionalen: zu einem vierten) Satelliten die Position P bestätigt werden, also:



Sind die Uhren jedoch falsch synchronisiert, wird mit dem dritten Abstand jedoch kein Schnittpunkt erzielt. Die fehlerhafte Zeitmessung betrifft alle drei Messungen in gleichem Maße (da alle Satelliten durch ihre Atomuhren immer synchronisiert sind!). Der



Empfänger kann nun alle Messungen mit demselben Faktor korrigieren, bis ein Schnittpunkt gefunden wird – und damit die korrekte Position sowie die exakte Zeit.



Trotz aller Vorteile und der Genauigkeit, die GPS liefert, gibt es dennoch einige Fehlerquellen, welche die Positionsbestimmung verfälschen können:

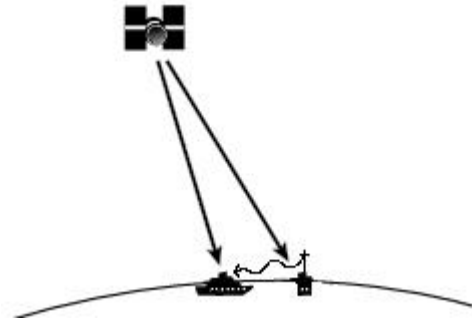
- Schwankungen in der Umlaufbahn der Satelliten
- Störungen durch Druckunterschiede, Wetterverhältnisse und Ähnliches, in der Atmosphäre
- Uhrenfehler (Atomuhren in den Satelliten)
- Signalreflektionen in der Umgebung des Empfängers
- Störungen durch geladene Teilchen in der Ionosphäre, und/oder durch Wasserdampfmoleküle in der Troposphäre
- *Geometric Dilution of Precision* (GDOP): je näher zwei Satelliten, die zur Positionsbestimmung herangezogen werden, zueinander stehen, desto schlechter ist die erzielte Genauigkeit
- Bis zum 01.05.2000 wurde die Genauigkeit von GPS künstlich durch die sogenannte „*Selective Availability*“ herabgesetzt; gegenwärtig steht die

ehemals nur dem Militär zur Verfügung stehende hohe Genauigkeit jedoch jedem Benutzer zur Verfügung

Diese Fehlerquellen sind die maßgebliche Motivation für *Differential GPS* (DGPS).

2.4.2 DGPS

Die Idee bei differentielltem GPS ist die Hinzunahme eines stationären Referenzpunktes, dessen Position exakt bekannt ist. Dieser feste Empfänger bestimmt fortlaufend seine eigene Position mit GPS, vergleicht diese mit seiner exakten bekannten Position, ermittelt daraus den momentanen Fehler, und sendet diesen schließlich an alle sich in seiner Umgebung befindenden GPS-Empfänger. Die GPS-Empfänger können diesen Fehler dann in ihre Positionsberechnung mit einbeziehen.



2.4.3 IDGPS

IDGPS steht für „inverted DGPS“, und wurde ursprünglich für solche Situationen konzipiert, bei denen die Position eines Teilnehmers für den Teilnehmer selbst zwar keine hohe Bedeutung darstellt, für eine Koordinierungszentrale oder Ähnliches jedoch umso mehr. Bei diesem Verfahren übermitteln alle mobilen GPS-Empfänger ihre Position an einen zentralen Server, der an diesen dann DGPS-Korrekturen vornimmt. Vorteilhaft ist die Tatsache, dass die mobilen Teilnehmer keine teureren DGPS-Geräte einsetzen müssen, allerdings müssen die verwendeten Geräte unbedingt in der Lage sein dem Server mitzuteilen, welche Satelliten sie zu ihrer Positionsbestimmung herangezogen haben, da dieser andernfalls keine sinnvolle DGPS-Korrektur vornehmen kann.

2.4.4 WAAS

Das *Wide Area Augmentation System*, kurz WAAS, basiert auf einem DGPS-ähnlichem Ansatz, nur dass auch die Korrekturdaten über Satelliten verbreitet werden. Dazu senden zunächst sämtliche feste (DGPS-) Referenzpunkte ihre Korrekturdaten an eine sogenannte „Master Control Station“, diese leitet die Daten dann an einen geostationären Satelliten weiter, von dem sie dann schließlich zum Benutzer gelangen.

WAAS ist derzeit nur in den USA verfügbar.

2.4.5 EGNOS

Das *European Geostationary Navigation Overlay System* ist das europäische Pendant zu WAAS, verwendet GPS, und wird voraussichtlich 2004 fertiggestellt.

2.4.6 GALILEO

GALILEO ist ein Projekt der Europäischen Union, und soll nach seiner Fertigstellung im Jahr 2008 eine noch genauere Positionsbestimmung ermöglichen als GPS. Wegen der Unvollständigkeit des Projektes soll hier jedoch nicht näher darauf eingegangen werden.

2.4.7 GLONASS

Das *Globalnaya Navigacionnaya Sputnikovaya Sistema* ist das russische Pendant zu GPS. Da seit 10.01.2000 nur noch 10 Satelliten in Betrieb sind, ist das System praktisch jedoch nicht mehr nutzbar.

3. Das NMEA-Protokoll

NMEA steht für *National Marine Electronics Association*. Der von diesem Verein entwickelte Protokoll-Standard, der als Kommunikationsschnittstelle einer großen Gerätevielfalt fungiert, trägt selbst die Bezeichnung „NMEA“, und wird insbesondere von GPS-Empfängern zur Datenausgabe verwendet. An dieser Stelle muss darauf hingewiesen werden, dass der NMEA-Standard *nicht* frei erhältlich ist. Der volle Standard ist an der Adresse [12] erhältlich. Einzelheiten über viele der gängigen Datensätze sind zwar bekannt; das soll jedoch nicht darüber hinwegtäuschen, dass die im Rahmen dieser Studienarbeit benutzte Information gegebenenfalls unvollständig ist oder eventuell falsch ausgelegt wurde!

3.1 Eigenschaften, Aufbau, Datenformat

NMEA-0183 verwendet, mit Ausnahme der zwei Sonderzeichen CR & LF, nur druckbare ASCII-Zeichen. Daten werden bei 4800 Baud übertragen, mit 8 Datenbits, keiner Parität und einem Stopbit.

Der Aufbau von NMEA-Nachrichten ist Folgender: jede Nachricht beginnt mit einem Dollarzeichen “\$“, danach folgen in dieser Reihenfolge:

- eine SenderID: besteht aus genau zwei Zeichen und gibt an, welche Art von Gerät sendet

- eine NachrichtID: besteht aus genau drei Zeichen, beschreibt implizit die zu erwartende Anzahl der folgenden Datenfelder und was diese beinhalten (Kontext)
- eine variable Anzahl an Datenfeldern (abhängig von der jeweiligen NachrichtID), getrennt durch Kommata
- eine (in den meisten Fällen optionale) Prüfsumme, eingeleitet durch einen “*” und bestehend aus zwei hexadezimalen Ziffern; diese zwei Ziffern stellen das exklusive ODER aller Zeichen zwischen, aber nicht einschließlich, dem “\$“ und dem “*” dar
- ein abschließendes CR/LF (Carriage Return/Line Feed, z.D. Wagenrücklauf und Zeilenvorschub).

Darüber hinaus erlaubt der NMEA-Standard die Definition hersteller-spezifischen Nachrichten: diese beginnen mit “\$P“, gefolgt von einer drei Zeichen langen HerstellerID und von jenen Daten, die der jeweilige Hersteller für sinnvoll betrachtet, jedoch dem Stil der herkömmlichen Nachrichten folgend.

Die maximale Länge einer NMEA-Nachricht beträgt 82 Zeichen, einschließlich “\$“ und CR/LF.

Für den Fall, dass keine Daten für ein bestimmtes Feld zur Verfügung stehen sollten, wird dieses einfach weggelassen; die Kommata jedoch, die dieses Feld delimitieren würden, werden trotzdem gesandt (ohne Leerzeichen zwischen diesen, es wird also ein “leeres Feld“ gesandt).

3.2 SenderIDs

Die in einer NMEA-Nachricht immer enthaltene SenderID identifiziert das Gerät, das die Nachricht verschickt hat. SenderIDs spielen für diese Studienarbeit eine unwesentliche Rolle und sind hier nur der Vollständigkeit halber aufgelistet.

Sendendes Gerät	Genauer Gerätetyp	SenderID
Autopilot	General	AG
	Magnetic	AP
Communications	Digital Selective Calling (DSC)	CD
	Satellite	CS
	Radio-Telephone (MF/HF)	CT
	Radio-Telephone (VHF)	CV
	Scanning Receiver	CX
DECCA Navigation		DE

Direction Finder		DF
Electronic Chart Display and Information System (ECDIS)		EC
Emergency Position Indication Beacon (EPIRB)		EP
Engine room Monitoring Systems		ER
<i>Global Positioning System (GPS)</i>		<i>GP</i>
Heading Sensors	Compass, Magnetic	HC
	Gyro, North Seeking	HE
	Gyro, Non-North Seeking	HN
Integrated Instrumentation		II
Integrated Navigation		IN
LORAN	Loran-A	LA
	Loran-C	LC
OMEGA Navigation System		OM
Proprietary code		P
Radar and/or ARPA		RA
Sounder, depth		SD
Electronic Positioning System		TR
Sounder, scanning		SS
Turn Rate Indicator		TI
TRANSIT Navigation System		TR
Geschwindigkeitsmesser	Doppler	VD
	Speed Log, Water, Magnetic	VM
	Speed Log, Water, Mechanical	VW
Transducer		YX
Timekeepers, Time/Date	Atomic Clock	ZA
	Chronometer	ZC
	Quartz	ZQ
	Radio Update, WWV or WWVH	ZV
Weather Instruments		WI

3.3 NachrichtIDs

Der wesentliche Unterschied zwischen zwei NMEA-Nachrichten ist die NachrichtID. Über die NachrichtID werden die Anzahl und das Format der darauffolgenden Datenfelder implizit beschrieben. Eine detaillierte Übersicht über alle implementierte NachrichtIDs ist in IV-3 zu finden.

4. Geographische Koordinaten

Um in einem Koordinatensystem die Entfernung und die Richtung zwischen zwei Punkten zu berechnen, ist es notwendig, ein möglichst exaktes Modell der Erde zu besitzen.

4.1 Die Erde als Kugel

Für einfache navigatorische Berechnungen kann es durchaus ausreichen, die Erde als eine vollkommene Kugel zu betrachten – der Erdradius beträgt 6370 Kilometer, der Erdumfang liegt bei 40003,2 Kilometer.

4.2 Die Erde als Ellipsoid

Wird der Erdkörper genauer betrachtet, stellt sich heraus, dass die Erde rotations- und gravitationsbedingt an den Polen leicht abgeplattet und am Äquator dafür etwas aufgewölbt ist – die Erde ist ein sogenanntes Rotationsellipsoid. Darauf muss bei großmaßstäbigeren Navigationsberechnungen geachtet werden. Um diesem Umstand einzubeziehen wurden im Laufe der Jahre eine Vielfalt unterschiedlicher Modelle der Erde entwickelt. Die gegenwärtig wichtigsten und gängigsten Ellipsoidmodelle zeigt folgende Tabelle:

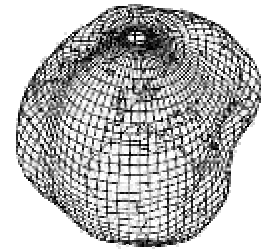
Ellipsoidmodell	Geodätisches Datum (map datum)	X	Y
Besselscher Ellipsoid	Potsdam Datum	6377,397 Km	6356,079 Km
European Terrestrial Reference System	ETRS89	6378,137 Km	6356,752 Km
Internationaler Ellipsoid	ED50	6378,388 Km	6356,912 Km
World Geodetic System 1984	WGS84	6378,137 Km	6356,752 Km

Koordinaten in verschiedenen Bezugssystemen lassen sich zwar prinzipiell ineinander umrechnen, allerdings ist dies oft nicht ganz einfach.

WGS84 ist das bei GPS-Empfängern am häufigst benutztes map datum.

4.3 Die Erde als Geoid

Tatsächlich ist die Erde aber weder eine Kugel, noch ein Rotationsellipsoid. Wegen seiner unterschiedlichen Massedichte besitzt der Erdkörper keine regelmäßige Figur – die Erde stellt vielmehr eine unregelmäßige Fläche dar, welche als „Geoid“ bezeichnet wird. Mathematisch ist ein Geoid wegen seiner Unregelmäßigkeiten schwer



erfassbar – weshalb er sich für Navigationsberechnungen eher schlecht eignet. Jedoch kommt er insbesondere für Höhenmessungen in Betracht.

4.4 Das internationale geographische Koordinatensystem

Beim internationalen geographischen Koordinatensystem handelt es sich um ein dreidimensionales Koordinatensystem mit senkrecht aufeinander stehenden Kreislinien, den Breitenkreisen (horizontal) und den Längenkreisen (vertikal). Ein halber Längengrad wird Meridian genannt. Das System wird oft auch als geographisches Koordinatensystem von Greenwich bezeichnet. Vom Äquator aus werden 90° nach Norden bzw. 90° nach Süden gezählt, was den Breitengrad (Englisch: Latitude) ergibt, und von dem Nullmeridian von Greenwich werden 180° nach Westen bzw. 180° nach Osten gezählt, was dem Längengrad entspricht (Englisch: Longitude).

Ein Grad lässt sich weiter unterteilen in $60'$ Bogenminuten, und diese unterteilen sich wiederum in $60''$ Bogensekunden. Die Angabe in Bogensekunden ist allerdings aus Rechenrunden nicht sonderlich geeignet; daher ist es praktischer und auch üblich, Dezimalminuten anstatt Bogensekunden zu verwenden.

Beispiel:

Gradangabe für $1,9578^\circ$ in		Rechenweg
Bogensekunden	Dezimalminuten	
$1^\circ 57' 28''$	$1^\circ 57,47'$	$1,9578^\circ = 1^\circ + 0,9578^\circ$ $0,9578^\circ = 57,468'$ $57,468' = 57' + 0,468'$ $0,468' = 28,08''$

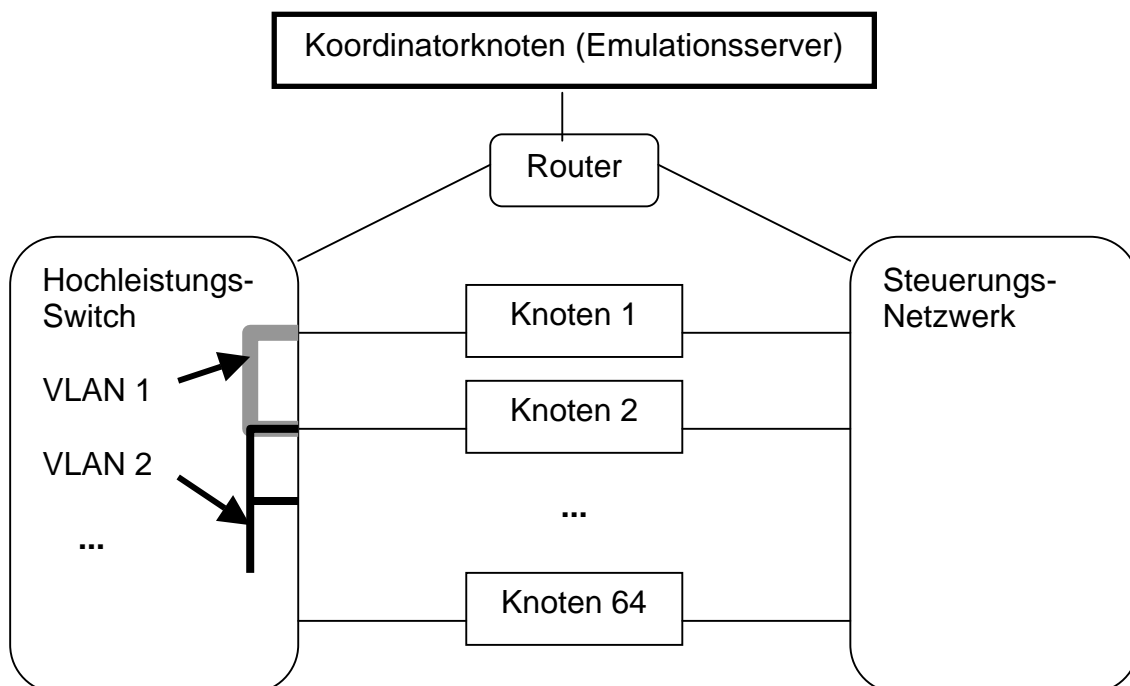
Jede Position auf der Erdoberfläche lässt sich mit der Angabe des Breitengrads und des Längengrads eindeutig beschreiben. Ein Beispiel für eine (eindeutige) Position: $45^\circ 17,37' S$ und $030^\circ 25,83' W$.

IV. Virtuelles Positionierungsgerät

1. Architektur

1.1 Infrastruktur: NET

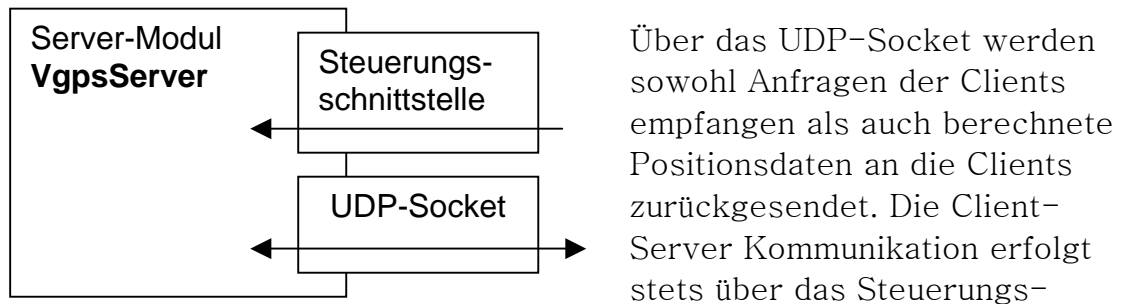
Das NET besteht im Wesentlichen aus einem flexibel vernetztem PC-Cluster mit 64 Knoten, verbunden mit einem Hochleistungsswitch und einem separaten Kontrollnetzwerk. Verbindungen zwischen den (Emulations-) Knoten werden durch sogenannte „virtual LANs“ (VLANs) aufgebaut, wobei jedes dieser VLANs über ein eigenes, virtuelles network device ansprechbar ist. So wird es möglich, gegebenenfalls eine hoch komplexe Topologie zwischen den einzelnen Knoten aufzubauen; außerdem ist an jedes dieser virtuellen network devices eine Instanz von NETShaper gekoppelt, so dass der Datenfluss spezifisch gesteuert und beeinflusst werden kann (siehe [2]). Während einer Emulation ein ausgezeichnete Koordinator-Knoten (Server) die wesentliche Rolle: der Server steuert aufgrund der ihm vorliegenden Modelle den Ablauf der Emulation; Parameter werden beschrieben und für jeden Client-Knoten bestimmt, wie dieser sich verhalten soll. Das zugrunde liegende Betriebssystem ist Linux.



1.2 VgpsServer

Die Serverseite des virtuellen Gerätes, fortan „VgpsServer“ genannt, wurde konzipiert, um auf dem Koordinatorknoten des NETs zu laufen. Das ist sinnvoll, denn während einer Emulation wird das Emulationsszenario von diesem Knoten gesteuert. Somit muss der emulationssteuernde Prozess lediglich die von VgpsServer zur Verfügung gestellte Funktionalität nutzen, um (virtuelle) Bewegungen der Emulationsknoten zu erzeugen.

VgpsServer stellt folgende Schnittstellen zur Verfügung:

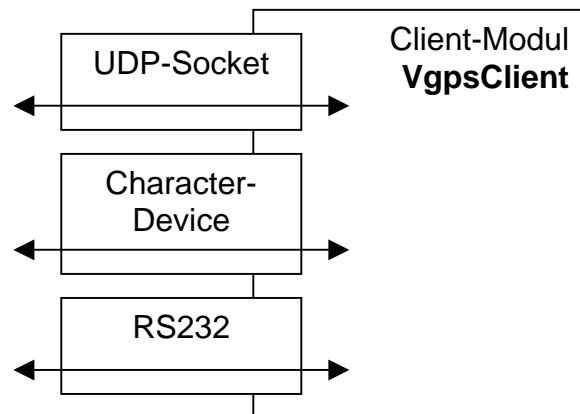


Über das UDP-Socket werden sowohl Anfragen der Clients empfangen als auch berechnete Positionsdaten an die Clients zurückgesendet. Die Client-Server Kommunikation erfolgt stets über das Steuerungsnetzwerk, somit wird das Emulationsnetzwerk nicht unnötig belastet. Über die Steuerungsschnittstelle wird die Emulation sämtlicher Clients auf den Emulationsknoten gesteuert.

1.3 VgpsClient

Die Clientseite des virtuellen Gerätes wird als „VgpsClient“ bezeichnet. VgpsClient kann auf einem und demselben Emulationsknoten mehrfach instanziiert werden, und stellt dort GPS-Daten zur Verfügung;

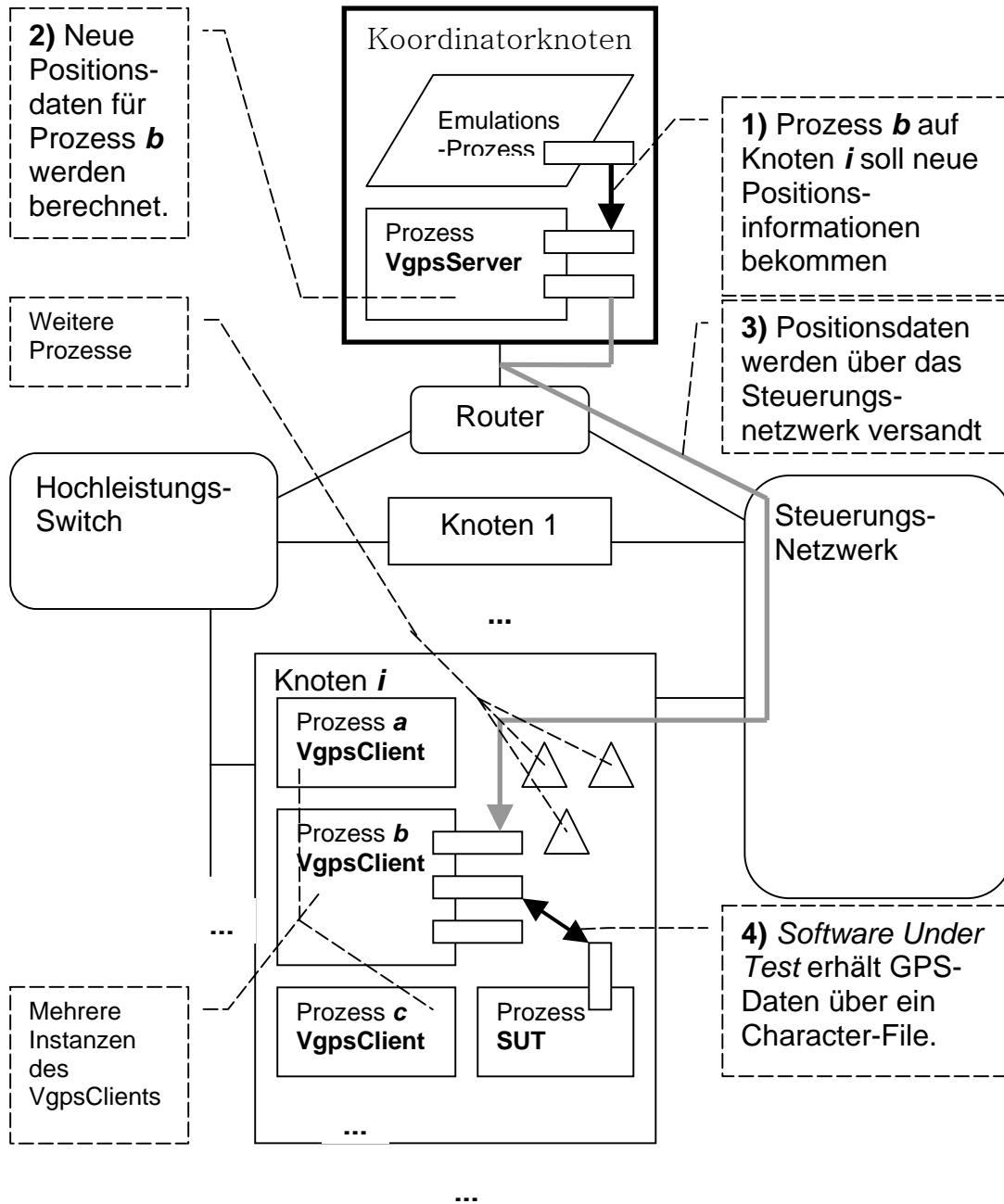
Zu diesem Zweck stehen zwei Optionen zur Wahl: die GPS-Daten können wie bei einem realen GPS-Empfänger über eine serielle Schnittstelle zur Verfügung gestellt werden, oder sie werden über ein sogenanntes „Character Device“ gesendet. Bei beiden Optionen ist es möglich,



Server-Anfragen zu stellen. Über das UDP-Socket erfolgt die Client-Server Kommunikation (verläuft über das Steuerungsnetzwerk).

1.4 Gesamtbild

Anhand eines fiktiven Emulationsablaufes auf dem NET soll ein Gesamteindruck über das virtuelle Gerät vermittelt werden.



2. Funktionsweise und Bedienung

2.1 VgpsServer

VgpsServer wurde als optionales Modul konzipiert, das bei Bedarf von einem Prozess instanziiert werden sollte und das diesem dann seine Funktionalität zur Verfügung stellt. Es ist zwar auch möglich, VgpsServer als selbstständige Java-Applikation zu betreiben, doch zunächst soll hier der vorgesehene Betrieb als optional instanziiertes Modul detaillierter betrachtet werden.

Zur Instanziierung stehen fünf Konstruktoren zur Verfügung. Diese unterscheiden sich untereinander nur in der Anzahl der Defaultwerte, die bei späteren Berechnungen eingesetzt werden. Im Folgenden soll näher darauf eingegangen werden.

```
public VgpsServer(InetAddress addr, int svrPort, int lag, int labm,  
                 int labd, char las, int log, int lobm, int lobd,  
                 char los)
```

Dieser Konstruktor ist zugleich der Vollständigste: es werden keine Defaultwerte eingesetzt.

Beim Instanzieren benötigt VgpsServer zwei wesentliche Informationen:

- eine lokale IP-Adresse und einen gültigen Port, an welche das Kommunikationssocket gebunden werden soll, und
- Daten zur geographischen Verankerung des abstrakten Koordinatensystems auf eine konkrete geographische Position.

Mit Hilfe der Parameter **addr** und **svrPort** wird das Kommunikationssocket der VgpsServer-Instanz initialisiert – das Socket wird an die entsprechende IP (oder an den entsprechenden Hostname) und an den entsprechenden Port gebunden. Zur geographischen Verankerung wird der abstrakte Punkt (X,Y) = (0,0) auf die Parameter des Konstruktors verankert.

addr	lokale IP-Adresse oder Hostname, an den das Kommunikationssocket gebunden werden soll
svrPort	gültige Portnummer, an die das Kommunikationssocket gebunden werden soll (zwischen 0 und 65535)
lag	Gradangabe für den Breitengrad (zwischen 0 und 90, falls 90, müssen lab und lad gleich 0 sein)
lab	Anzahl der Bogenminuten für den Breitengrad (zwischen 0 und 59)
lad	Anzahl der Dezimalminuten für den Breitengrad (zwischen 0 und 99)
las	Nördliche (N) oder Südliche (S) Hemisphäre (Breitengrad)

log	Gradangabe für den Längengrad (zwischen 0 und 180, falls 180, müssen lob und lod gleich 0 sein)
lob	Anzahl der Bogenminuten für den Längengrad (zwischen 0 und 59)
lod	Anzahl der Dezimalminuten für den Längengrad (zwischen 0 und 99)
los	Östliche (E) oder Westliche (W) Hemisphäre (Längengrad)

Werden Parameter weggelassen bzw. unvollständige Konstruktoren benutzt, so müssen Defaultwerte eingesetzt werden. Folgende Tabelle veranschaulicht, welche Werte als Default in Frage kommen:

public VgpsServer()	addr	Socket wird an eine freie IP-Adresse gebunden, falls vorhanden.
public VgpsServer(int svrPort)	svrPort	Socket wird an den Port 9753 gebunden, falls erlaubt
public VgpsServer(InetAddress addr, int svrPort) public VgpsServer(int svrPort, int lag, int labm, int labd, char las, int log, int lobm, int lobd, char los)	lag	0°
	lab	0'
	lad	0.00'
	las	N
	log	0°
	lob	0'
	lod	0.00'
	los	E
		00°00,00' N 000°00,00' E Schnittpunkt zwischen Äquator und Nullmeridian von Greenwich

Nach der Instanziierung stehen dem aufrufendem Prozess die Methoden der Instanz zur Verfügung; zunächst ist es aber noch wichtig, den Prozess „VgpsServer“ zu starten. Dafür steht folgende Methode bereit:

public void startServer()

Nach diesem Aufruf ist VgpsServer vollständig gestartet und bereit, Anfragen von Clients entgegen zu nehmen.

Zur Steuerung der (virtuellen) Positionen der Clients sind zwei Methoden gegeben; wie bei den Konstruktoren gilt auch hier: falls die Methode mit weniger Parametern bevorzugt wird, werden Defaultwerte eingesetzt.

```
public void newPosition(InetAddress cltAddr, int cltPort,
    float originX, float originY, float currentX,
    float currentY, float destinationX,
    float destinationY, float speed)
```

cltAddr	IP-Adresse (bzw. Hostname) des Clients, den die Bewegung betrifft
cltPort	Port des Clients, den die Bewegung betrifft
originX	Relative Startkoordinaten des Clients (in dem abstrakten, d.h. relativen Koordinatensystem)
originY	
currentX	Momentane (relative) Position des Clients
currentY	
destinationX	(Relativer) Zielpunkt des Clients, d.h. Client bewegt sich auf diesen Punkt zu
destinationY	
speed	Momentane Geschwindigkeit

Ein vollständiger Aufruf kann mit der folgenden Methode erzielt werden:

```
public void newPosition(InetAddress cltAddr, int cltPort,
    float originX, float originY, float currentX,
    float currentY, float destinationX,
    float destinationY, float speed, int jamCode,
    int satsInSight, int satsTracked,
    GregorianCalendar utc,
    int curMagnNorthLatGrad, int curMagnNorthLatBows,
    int curMagnNorthLatDegs, char curMagnNorthLatSide,
    int curMagnNorthLonGrad, int curMagnNorthLonBows,
    int curMagnNorthLonDegs, char curMagnNorthLonSide,
    String hdop, String heightAboveMeanSeaLevel,
    char heightAMSLUnit, String heightGeoidAboveWgs84,
    char heightGAWUnit, String timeSinceLastDgpsUpdate,
    String dgpsStationId, int[] prn, String pdop,
    String vdop, String[] satsAltitude,
    String[] satsAzimuth, String[] satsSnr,
    String curMagnDeclination,
    char curMagnDeclinUnit)
```

In folgender Tabelle sind nur die Parameter berücksichtigt, welche die beiden Methoden von einander unterscheiden:

jamCode	ein realer GPS-Empfänger wird gestört, falls er sich innerhalb eines Gebäudes oder in der Nähe von Hochspannungsleitungen befindet. Andere Störungsquellen sind unter Umständen auch gegeben. Durch diesen Parameter wird dem virtuellen Gerät
---------	--

	signalisiert, welche äußere Umstände evtl. Einfluss haben. Der Wert 0 signalisiert OK, der Wert 1 signalisiert eine schwache Störung, der Wert 2 eine stärkere Störung usw..
satsInSight	Anzahl der Satelliten, die von dem virtuellen Gerät aus sichtbar sind.
satsTracked	Anzahl Satelliten, die von dem virtuellen Gerät tatsächlich für die Berechnungen herangezogen werden.
utc	UTC, Universal Coordinated Time.
curMagnNorthLatGrad, curMagnNorthLatBows, curMagnNorthLatDegs, curMagnNorthLatSide, curMagnNorthLonGrad, curMagnNorthLonBows, curMagnNorthLonDegs, curMagnNorthLonSide	Der magnetische und der geographische Nordpol der Erde fallen NICHT zusammen. Der magnetische Nordpol ist nicht fix, seine Position ändert sich im Laufe der Jahre wesentlich. Daraus hat jede Position auf der Erde ihren eigenen magn. Nordpol: ein Kompass peilt nicht etwa den magnetischen Pol an, sondern richtet sich nach den Feldlinien der Erde aus. Diese bestehen aus der Summe der Einwirkungen des magn. Pols, aber auch anderer nicht zu vernachlässigender Faktoren. Alleine anhand der gelieferten abstrakten Koordinaten kann das virtuelle GPS-Gerät unmöglich zur Kenntnis des magnetischen Pols einer bestimmten Position kommen. Daher ist hier die Möglichkeit gegeben, dem virtuellen Gerät dies mitzuteilen.
hdop	Horizontal Dilution of Precision.
heightAboveMeanSeaLevel	Höhenmeter über dem mittleren Wasserstand.
heightAMSLUnit	Einheit für die Höhe, entweder 'K' (Kilometer) oder Meilen 'N'.
heightGeoidAboveWgs84	Höhe des Geoids über dem WGS84-Ellipsoid.
heightGAWUnit	Einheit für <i>heightGeoidAboveWgs84</i>
timeSinceLastDgpsUpdate	Zeit, seitdem von der DGPS-Basisstation letztlich Ausgleichdaten empfangen wurden.
dgpsStationId	Kennung der DGPS-Basisstation (0000 bis 1023).
prn	"Pseudo-Random Noise", spezifischer Code eines bestimmten Satellits.
pdop	Overall Dilution of Precision.
vdop	Vertical Dilution of Precision.
satsAltitude	Höhe der Satelliten (in Grad).
satsAzimuth	Azimut (in Grad) zu geographisch Nord.
satsSnr	"Signal-to-Noise-Ratio", Signalstärke der einzelnen

	Satelliten.
<code>curMagnDeclination</code>	Declination an diesem bestimmten Ort.
<code>curMagnDeclinUnit</code>	Declination östlich (E) oder westlich (W).

Bei der Verwendung der vollständigen Methode ist besonders darauf zu achten, dass die Array-Parameter stets dieselbe Anzahl Elemente beinhalten. Für einen Index i stellen `prn[i]`, `satsAltitude[i]`, `satsAzimuth[i]` und `satsSnr[i]` den Datensatz für den Satelliten i dar.

Ausser den beiden zu Bewegungserzeugung angesprochenen Methoden, wurden zu besseren Steuerungszwecken noch folgende implementiert:

`public void stopServer()`

Der Aufruf beendet den Empfangsprozess, als Konsequenz werden keine Client-Anfragen mehr beachtet. Die Instanz bleibt jedoch erhalten, und weitere Bewegungsmethoden-Aufrufe dürfen erfolgen.

`public void stopServerAndShutdownClients()`

Bietet dieselbe Funktionalität wie obige Methode, beendet zuzüglich jedoch alle Client-Prozesse.

`public void shutdownClient(InetAddress addr, int port)`

Bietet die Möglichkeit, gezielt einen bestimmten Client zu beenden.

`public void shutdownAllClients()`

Beendet alle Client-Prozesse, bleibt selbst jedoch aktiv.

Wird `VgpsServer` als selbstständige Java-Applikation betrieben, so werden zwar Antworten auf eventuelle Client-Anfragen gesendet; da jedoch die Bewegungsmethoden nicht aufgerufen werden, wird jeder Client stets regungslos an dem selben Ort verweilen.

2.2 VgpsClient

Die Clientseite des virtuellen Gerätes, der `VgpsClient`, wurde dazu konzipiert, als selbstständige Java-Anwendung auf einem Rechner GPS-Daten zur Verfügung zu stellen.

`VgpsClient` wird über die Linux-Kommandozeile (oder aus einem Skript heraus) gestartet, und akzeptiert folgende Argumente:

-E	Aktiviert die <i>CompactGPS</i> -Emulation. In diesem Fall wird die Abonnieerung bestimmter Nachrichten deaktiviert.
-sip <server IP>	Spezifiziert die IP-Adresse (oder den Hostname) des Servers. Default ist eine lokal verfügbare IP.
-sport <server Port>	Spezifiziert die Portnummer des Servers. Default ist 9753.
-cip <client IP>	Bindet das Clientsocket an eine bestimmte IP-Adresse, falls vorhanden. Default ist eine lokal verfügbare IP.
-cport <client Port>	Bindet das ClientSocket an einen bestimmten Port, falls erlaubt. Default ist 9753.
-h	Zeigt eine Kurzhilfe auf dem Bildschirm. Diese Nachricht wird auch angezeigt, falls falsche Argumente geliefert werden.
-stdout	Die GPS-Daten werden auf Standard-Out ausgegeben. (Default)
-rs232 [1 2]	Ausgabe der GPS-Daten auf einer seriellen Schnittstelle; optional kann zwischen COM1 und COM2 gewählt werden, default ist COM1.
-pty <symbolic link>	Die Ausgabe der GPS-Daten erfolgt über ein sogenanntes "pseudo-terminal". Es wird ein symbolischer Link erzeugt, der auf das entsprechende pty zeigt. Optional kann der Name für den symbolischen Link angegeben werden, default ist „vgpsinoutf“.

Wenn der Start des VgpsClients erfolgreich verläuft, d.h., keine Fehler beim Parsen der Kommandozeilenparameter aufgetreten sind, wird kurz danach eine „\$PVGP,AWAKE“-Nachricht zum Server gesandt.



Damit erfolgt eine Registrierung bei dem Server, und dieser antwortet mit einem Satz momentan abonniertes Nachrichten.



Diese Nachrichten werden von VgpsClient nun lokal gespeichert, und jede Sekunde ausgegeben. Ob die Ausgabe über Standard-out, einer seriellen Schnittstelle oder über ein Character Device erfolgt, ist Kommandozeilen-abhängig.

Sobald von einem Emulationsszenario bestimmt wird, dass eine Bewegung des Knotens stattfinden soll, berechnet VgpsServer anhand relativer Koordinaten die momentane geographische Position des Clients, und sendet diesem die entsprechenden Änderungen zu. Der Client speichert diese wieder lokal und wiederholt die Ausgabe alle Sekunde.

Im Regelfall wird die Kommunikation meist einseitig vom Server zum Client verlaufen. Der Client hat jedoch nicht desto trotz die Möglichkeit, selbst dynamisch zu agieren – etwa, um eine spezifische Nachricht (-engruppe) zu abonnieren, oder, um dem Server neue Wegpunkte mitzuteilen, die er gerne in die Berechnung mit einbezogen hätte. Folgende Tabelle zeigt NMEA-Sätze, die über die Schnittstelle zur Anwendung entgegen genommen werden:

\$PVGP,SHUTDOWN*	Beendet den Client.
\$PVGP,SUBSCRIBE,###,###,###,###,###*	Abboniert eine bestimmte Nachricht (-engruppe). Funktioniert nur, falls VgpsClient ohne den „-E“ Parameter gestartet wurde. Für „###“ können alle NachrichtIDs eingesetzt werden, die implementiert wurden (siehe IV-4).
\$PVGP,RESETWPS*	Löscht alle Wegpunkte, die mit „\$GPWPL...“ eingegeben wurden.
\$GPWPL,llll.ll,N,yyyy.yy,W,WPID*hx<CR><LF>	Fügt einen Wegpunkt zur Liste hinzu; dieser Wegpunkt wird in künftigen Berechnungen mit einbezogen.

4. Implementierte NMEA-Nachrichten

Im Folgenden werden die NMEA-Nachrichten vorgestellt, die implementiert wurden. Eine detaillierte Beschreibung der Datenfelder gibt Aufschluss über die Semantik der einzelnen Nachrichten.

AAM – “Waypoint Arrival Alarm“

Format: $\$##AAM,A,V,x.x,K,WPID*hx<CR><LF>$

Kurzbeschreibung: AAM signalisiert die Ankunft am Radius bzw. die Überschreitung eines Wegpunktes.

Bedeutung der Datenfelder:

- $\$$ – Anfang einer neuen NMEA-Nachricht
- $##$ – SenderID (siehe III-3.2)
- AAM – NachrichtID: Datenfelder sind als “Waypoint Arrival Alarm“-Daten zu interpretieren
- A – Boolean: Wegpunktradius wurde betreten. Werte:
 - A = Active (true)
 - V = Void (false)
- V – Boolean: Wegpunkt wurde überschritten. Werte: A oder V
- $x.x$ – Float: Radius des Wegpunktes
- K – Einheit für den Radius. Werte:
 - K = Kilometer
 - N = Nautische Meilen
- $WPID$ – Wegpunkt ID
- $*hx$ – Prüfsumme (hex)
- $<CR>$ – Carriage Return (Wagenrücklauf)
- $<LF>$ – Line Feed (Zeilenvorschub)

APA – “Autopilot Sentence A“

Format: $\$##APA,A,V,x.x,L,N,V,A,xxx,M,WPID*hx<CR><LF>$

Kurzbeschreibung: APA wird von einigen GPS-Empfängern zum Zwecke der Steuerung eines Autopiloten zur Verfügung gestellt.

Bedeutung der Datenfelder:

- A – Boolean: gibt Aufschluss über die Zuverlässigkeit der Messung. Werte:
 - A = Messung OK
 - V = Warnung, Loran-C Blinkung oder SNR Warnung, Messung nicht zuverlässig oder ungültig.
- V – Boolean von Loran-C: Zyklus-Lock Warnung. Werte:
 - A = OK
 - V = Loran-C Warnung

- x.x* - Ausmaß der Kursabweichung
- L* - Angabe zur Kurskorrektur. Werte:
 - R* = zur Kurskorrektur nach RECHTS steuern
 - L* = zur Kurskorrektur nach LINKS steuern
- N* - Einheit für die Kursabweichung. Werte: K oder N
- V* - Boolean: Wegpunktradius überschritten. Werte: A oder V
- A* - Boolean: Wegpunkt überschritten. Werte: A oder V
- xxx* - Peilung (Grad) Start zu Ziel
- M* - Art der Peilung. Werte:
 - T* = True: rechtweisende Peilung, geographisch Nord
 - M* = Magnetic: missweisende Peilung, magnetisch Nord
- WPID* - Wegpunkt ID

APB – "Autopilot Sentence B"

Format: *###APB,A,V,x.x,R,K,V,V,x.x,T,WPID,x.x,M,x.x,T*hx<CR><LF>*

Kurzbeschreibung wie APA, Bedeutung der Datenfelder:

- A* - Boolean: gibt Aufschluss über die Zuverlässigkeit der Messung.
Werte:
 - A* = Messung OK
 - V* = Warnung, Loran-C Blinkung oder SNR Warnung, Messung nicht zuverlässig oder ungültig.
- V* - Boolean von Loran-C: Zyklus-Lock Warnung. Werte:
 - A* = OK
 - V* = Loran-C Warnung
- x.x* - Ausmaß der Kursabweichung
- R* - Angabe zur Kurskorrektur. Werte:
 - R* = zur Kurskorrektur nach RECHTS steuern
 - L* = zur Kurskorrektur nach LINKS steuern
- K* - Einheit für die Kursabweichung. Werte: K oder N
- V* - Boolean: Wegpunktradius überschritten. Werte: A oder V
- V* - Boolean: Wegpunkt überschritten. Werte: A oder V
- x.x* - Peilung (Grad) Start zu Ziel
- T* - Art der Peilung. Werte:
 - T* = True: rechtweisende Peilung, geographisch Nord
 - M* = Magnetic: missweisende Peilung, magnetisch Nord
- WPID* - Wegpunkt ID
- x.x* - Peilung, aktuelle Position zu Ziel
- M* - Art der Peilung. Werte: T oder M
- x.x* - Anzulegender Kurs zum Ziel-Wegpunkt
- T* - Art der Peilung. Werte: T oder M

BEC – "Bearing and Distance to Waypoint, Dead Reckoning"

Format:

*###BEC,hhmmss.ss,lll.ll,N,yyyyy.yy,W,x.x,T,x.x,M,x.x,N,WPID*hx<CR><LF>*

Kurzbeschreibung: BEC beschreibt die Peilung und Entfernung zu einem Wegpunkt.

Bedeutung der Datenfelder:

hhmmss.ss - Universal Coordinated Time (UTC)

lll.ll - Breitengrad des Wegpunktes

N - Breitengrad-Zusatz. Werte:

N = Nord

S = Süd

yyyyy.yy - Längengrad des Wegpunktes

W - Längengrad-Zusatz. Werte:

E = Ost

W = West

x.x - Rechtweisende Peilung

T - True (rechtweisende Peilung)

x.x - Missweisende Peilung

M - Magnetic (missweisende Peilung)

x.x - Nautische Meilen

N - Nautical miles

WPID - Wegpunkt ID

BOD - "Bearing, Waypoint to Waypoint"

Format: *###BOD,x.x,T,x.x,M,NEWP,OLWP*hx<CR><LF>*

Kurzbeschreibung: Peilung bezüglich der Geraden durch NEWP und OLWP.

Bedeutung der Datenfelder:

x.x - Rechtweisende Peilung

T - True

x.x - Missweisende Peilung

M - Magnetic

NEWP - ID des nächsten Wegpunktes

OLWP - ID des alten Wegpunktes

GGA - "Global Positioning System Fix Data"

Format:

*###GGA,hhmmss.ss,lll.ll,N,yyyyy.yy,W,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hx<CR><LF>*

Kurzbeschreibung: GGA liefert die aktuelle Position, sowie Daten über die Genauigkeit der Messung.

Bedeutung der Datenfelder:

hhmmss.ss - Universal Coordinated Time (UTC)

lll.ll - Breitengrad des Wegpunktes

- N* - Breitengrad-Zusatz. Werte: N oder S
- yyyyy.yy* - Längengrad des Wegpunktes
- W* - Längengrad-Zusatz. Werte: E oder W
- x* - Zuverlässigkeit der Messung. Werte:
 - 0 = Messung ungültig
 - 1 = Messung hat GPS-Qualität
 - 2 = Messung hat DGPS-Qualität
- xx* - Anzahl erfasster Satelliten
- x.x* - Genauigkeit, HDOP (Horizontal Dilution of Precision)
- x.x* - Höhe unter/über dem mittleren Wasserstand (Geoid)
- M* - Einheit für die Höhe: Meter
- x.x* - Höhendifferenz zwischen dem mittleren Wasserstand (Geoid) und dem WGS84-Ellipsoid; liegt der mittlere Wasserstand unter dem Ellipsoid, wird dies durch “-“ signalisiert
- M* - Einheit für die Höhendifferenz: Meter
- x.x* - Datum der DGPS-Daten, Zeit in Sekunden ab letztem Update, Feld wird weggelassen, falls kein DGPS benutzt wird/zur Verfügung steht
- xxxx* - ID der DGPS-Referenzstation, Werte von 0000 bis 1023

GLL – “Geographic Position, Latitude and Longitude“

Format: *###GLL,III.II,N,yyyyy.yy,W,hhmmss.ss,A*hx<CR><LF>*

Kurzbeschreibung: GLL liefert die aktuelle Position.

Bedeutung der Datenfelder:

- III.II* - Breitengrad des Wegpunktes
- N* - Breitengrad-Zusatz. Werte: N oder S
- yyyyy.yy* - Längengrad des Wegpunktes
- W* - Längengrad-Zusatz. Werte: E oder W
- hhmmss.ss* - Universal Coordinated Time (UTC)
- A* - Boolean: gibt Aufschluss über die Gültigkeit der Daten. Werte:
 - A = Daten sind gültig
 - V = Daten sind ungültig bzw. unzuverlässig

GSA – “GPS DOP and active Satellites“

Format: *###GSA,A,y,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x*x*hx<CR><LF>*

Kurzbeschreibung : GSA informiert über die Anzahl der aktiven Satelliten, sowie über die Genauigkeit der Messung.

Bedeutung der Datenfelder:

- A* - Positionsbestimmung, 2D oder 3D. Werte:
 - A = Automatisch
 - M = Manuell
- y* - Momentane Art der Positionsbestimmung. Werte:

1 = Keine Positionsbestimmung
 2 = 2D-Positionsbestimmung
 3 = 3D-Positionsbestimmung
x, ..., x - PRN-Nummern von bis zu 12 Satelliten
x.x - Genauigkeit, PDOP (Dilution of Precision) in Meter
x.x - Horizontale Genauigkeit, HDOP (Horizontal Dilution of Precision) in Meter
x.x - Vertikale Genauigkeit, VDOP (Vertical Dilution of Precision) in Meter

GSV – “Satellites in View“

Format: $###GSV,x,x,x,x,x,x,x,...*hx\langle CR\rangle\langle LF\rangle$

Kurzbeschreibung: GSV beschreibt Daten über die Satelliten, welche von dem GPS-Gerät momentan gefunden werden können.

Bedeutung der Datenfelder:

- x* - Gesamtanzahl der zu übertragenden GSV-Nachrichten
- x* - Nummer der aktuellen Nachricht
- x* - Anzahl erfasster Satelliten
- x* - Nummer des aktuellen Satellits
- x* - Höhe in Grad
- x* - Azimut in Grad zu geographisch Nord
- x* - SNR (Signal to Noise Ratio) in dB
- ... - Weitere Satelliteninfos zu anderen Satelliten

HDM – “Heading, Magnetic“

Format: $###HDM,x.x,M*hx\langle CR\rangle\langle LF\rangle$

Kurzbeschreibung: HDM beschreibt die missweisende Peilung.

Bedeutung der Datenfelder:

- x.x* - anliegender Kurs, missweisend
- M* - missweisend (magnetisch) Nord

HDT – “Heading, True“

Format: $###HDT,x.x,T*hx\langle CR\rangle\langle LF\rangle$

Kurzbeschreibung: HDT beschreibt die rechtweisende Peilung.

Bedeutung der Datenfelder:

- x.x* - anliegender Kurs, rechtweisend
- T* - rechtweisend (geographisch) Nord

HSC – “Heading Steering Command“

Format: $###HSC,x.x,T,x.x,M*hx\langle CR\rangle\langle LF\rangle$

Kurzbeschreibung : HSC beschreibt sowohl missweisende als auch rechtweisende Peilung.

Bedeutung :

- x.x* - anliegender Kurs, rechtweisend
- T* - True
- x.x* - anliegender Kurs, missweisend
- M* - Magnetic

RMB – "Recommended Minimum Navigation Information"

Format:

*###RMB,A,x.x,R,NEWP,OLWP,III.II,N,yyyyy.yy,W,x.x,x.x,x.x,A*hX<CR><LF>*

Kurzbeschreibung: RMB wird gesendet, sobald eine Route (Weg) aktiviert wird. Liefert weitgehend ähnliche Daten wie APA/APB.

Bedeutung der Datenfelder:

- A* - Boolean: gibt Aufschluss über die Zuverlässigkeit der Daten.

Werte:

A = Daten OK

V = Warnung, Daten nicht zuverlässig oder ungültig

- x.x* - Ausmaß der Kursabweichung

- R* - Angabe zur Kurskorrektur. Werte:

R = zur Kurskorrektur nach RECHTS steuern

L = zur Kurskorrektur nach LINKS steuern

NEWP - ID des nächsten Wegpunktes

OLWP - ID des alten Wegpunktes

III.II - Breitengrad des Zielwegpunktes

N - Breitengrad-Zusatz. Werte: N oder S

yyyyy.yy - Längengrad des Zielwegpunktes

W - Längengrad-Zusatz. Werte: E oder W

x.x - Reichweite zum Ziel, ausgedrückt in nautischen Meilen

x.x - rechtweisende Peilung zum Ziel

x.x - Geschwindigkeit richtung Ziel, ausgedrückt in Knoten

A - Boolean: Wegpunktradius überschritten. Werte: A oder V

RMC – "Recommended minimum specific GPS/Transit data"

Format:

*###RMC,hhmmss.ss,A,III,II,N,yyyyy.yy,W,x.x,x.x,xxxxxx,x.x,W*hX<CR><LF>*

Kurzbeschreibung: RMC drückt aktuelle Position, Geschwindigkeit und Zeit aus.

Bedeutung der Datenfelder:

hhmmss.ss - Universal Coordinated Time (UTC)

- A* - Boolean: gibt Aufschluss über die Zuverlässigkeit der Daten.

Werte:

A = Daten OK

V = Warnung, Daten nicht zuverlässig oder ungültig

- lll.ll* - Breitengrad
- N* - Breitengrad-Zusatz. Werte: N oder S
- yyyyy.yy* - Längengrad
- W* - Längengrad-Zusatz. Werte: E oder W
- x.x* - Geschwindigkeit über Grund, ausgedrückt in Knoten
- x.x* - anliegender Kurs, rechtweisend
- xxxxxx* - Datum, im Format ttmjj
- x.x* - Magnetische Deklination in Grad
- W* - Einheit für die Deklination. Werte: E oder W

RTE – “Waypoints in active route“

Format: *###RTE,x.x,x.x,C,WP01,WP02,...*hx<CR><LF>*

Kurzbeschreibung: RTE beschreibt die Namen aller in der aktiven Route genutzten Wegpunkte.

Bedeutung der Datenfelder:

- x.x* - Gesamtanzahl der zu übertragenden RTE-Nachrichten
- x.x* - Nummer der aktuellen Nachricht
- C* - Nachricht-Modus. Werte:
 - C* = Komplette Route, alle Wegpunkte
 - W* = aktuelle Route, letzter verlassener Wegpunkt, nächster Wegpunkt, dann die Übrigen
- WP01* - Wegpunkt ID
- WP02* - Wegpunkt ID
- ... - Wegpunkt ID

VTG – “Track made good and Ground Speed“

Format: *###VTG,x.x,T,x.x,M,x.x,N,x.x,K*hx<CR><LF>*

Kurzbeschreibung: VTG übermittelt sowohl rechtweisende als auch missweisende Peilung, sowie die aktuelle Geschwindigkeit, ausgedrückt sowohl in Knoten als auch in Kilometer.

Bedeutung der Datenfelder:

- x.x* - anliegender Kurs, rechtweisend
- T* - True
- x.x* - anliegender Kurs, missweisend
- M* - Magnetic
- x.x* - Geschwindigkeit, ausgedrückt in Knoten
- N* - Knoten
- x.x* - Geschwindigkeit, ausgedrückt in Kilometer pro Stunde
- K* - Kilometer pro Stunde

WCV – “Waypoint Closure Velocity“

Format: *###WCV,x.x,N,WPID*hx<CR><LF>*

Kurzbeschreibung: Geschwindigkeit, auf den Wegpunkt zu.

Bedeutung der Datenfelder:

- x.x* - Geschwindigkeit, ausgedrückt in Knoten
- N* - Knoten
- WPID* - Wegpunkt ID

WNC – “Distance, Waypoint to Waypoint“

Format: *###WNC,x.x,N,x.x,K,NEWP,OLWP*hx<CR><LF>*

Kurzbeschreibung: Entfernung zwischen zwei Wegpunkten.

Bedeutung der Datenfelder:

- x.x* - Entfernung, ausgedrückt in nautischen Meilen
- N* - Nautische Meilen
- x.x* - Entfernung, ausgedrückt in Kilometern
- K* - Kilometer
- NEWP* - ID des nächsten Wegpunktes
- OLWP* - ID des alten Wegpunktes

WPL – “Waypoint Location“

Format: *###WPL,III.II,N,yyyy.yy,W,WPID*hx<CR><LF>*

Kurzbeschreibung: Position eines Wegpunktes.

Bedeutung der Datenfelder:

- III.II* - Breitengrad des Wegpunktes
- N* - Breitengrad-Zusatz. Werte: N oder S
- yyyy.yy* - Längengrad des Wegpunktes
- W* - Längengrad-Zusatz. Werte: E oder W
- WPID* - Wegpunkt ID

XTE – “Cross-Track Error, Measured“

Format: *###XTE,A,V,x.x,R,K*hx<CR><LF>*

Kurzbeschreibung: beschreibt die Abweichung von der Route.

Bedeutung der Datenfelder:

- A* - Boolean: gibt Aufschluss über die Zuverlässigkeit der Messung.
Werte:
 - A* = Messung OK
 - V* = Warnung, Loran-C Blinkung oder SNR Warnung, Messung nicht zuverlässig oder ungültig.
- V* - Boolean von Loran-C: Zyklus-Lock Warnung. Werte:
 - A* = OK
 - V* = Loran-C Warnung
- x.x* - Ausmaß der Kursabweichung
- R* - Angabe zur Kurskorrektur. Werte:
 - R* = zur Kurskorrektur nach RECHTS steuern

L = zur Kurskorrektur nach LINKS steuern
K - Einheit für die Kursabweichung. Werte: K oder N

XTR – “Cross-Track Error, Dead Reckoning”

Format: $###XTR,x.x,R,N*hX<CR><LF>$

Kurzbeschreibung: siehe XTE.

Bedeutung der Datenfelder:

x.x - Ausmaß der Kursabweichung
R - Angabe zur Kurskorrektur. Werte:
R = zur Kurskorrektur nach RECHTS steuern
L = zur Kurskorrektur nach LINKS steuern
N - Einheit für die Kursabweichung. Werte: K oder N

ZDA – “Time and Date“

Format: $###ZDA,hhmmss.ss,xx,xx,xxxx,xx,xx*hX<CR><LF>$

Kurzbeschreibung: liefert Zeit und Datum.

Bedeutung der Datenfelder:

hhmmss.ss - Universal Coordinated Time (UTC)
xx - Tag, von 01 bis 31
xx - Monat, von 01 bis 12
xxxx - Jahr
xx - Lokale Zeitzone, Stunden
xx - Lokale Zeitzone, Minuten

ZFO – “UTC and Time from Origin Waypoint“

Format: $###ZFO,hhmmss.ss,hhmmss.ss,WPID*hX<CR><LF>$

Kurzbeschreibung: liefert die abgelaufene Zeit seit dem Verlassen des letzten Wegpunktes.

Bedeutung der Datenfelder:

hhmmss.ss - Universal Coordinated Time (UTC)
hhmmss.ss - Verstrichene Zeit
WPID - ID des letzten Wegpunktes

V. Ausblick

Herkömmliche Positionierungsgeräte dienen im Wesentlichen der Positionsbestimmung und der Navigation; die Emulation eines Positionierungsgerätes kann indes in vielerlei Hinsicht deutliche Vorteile bringen – insbesondere beim Test ortsbezogener Anwendungen. Zunächst lassen sich die Anschaffungskosten für die entsprechende Hardware (Positionierungsgerät) reduzieren bzw. komplett streichen; anhand tatsächlicher oder abstrakter Daten (zum Beispiel relative Koordinaten in einem anderen Koordinatensystem), kann ein emuliertes Positionierungsgerät unabhängig von seiner momentanen „Position“ willkürliche bzw. wohldefinierte Positionsinformationen erzeugen, welche dann den zu testenden Anwendungen zur Verfügung gestellt werden können – mit anderen Worten, es ist im Gegensatz zu seinem realen Pendant keine Mobilität des virtuellen Gerätes nötig, um ein Szenario zu simulieren, in welchem ein Objekt sich von einem wohldefinierten Startpunkt zu einem ebenso vorgegebenen Endpunkt bewegt. Ein emuliertes Positionierungsgerät ist in der Lage, ein breiteres Spektrum an Informationen bzw. Daten bereit zu stellen, in dem es Funktionen und Funktionsweisen verschiedener realer Geräte implementiert und vereinigt – somit ergibt sich die Möglichkeit, ausgiebigere Tests durchzuführen. Es lassen sich komplexe Testszenarien aufbauen, in dem mehrere Instanzen eines emulierten Gerätes erzeugt werden; mehrere, von einander unabhängige Anwendungen, gekoppelt an eine eigene Instanz, können zu einem großen, komplexen Modell zusammenfließen, welches dann selbst als Subjekt eines Tests auftreten kann. Letztendlich, wenn die Transparenz des emulierten Positionierungsgeräts gegeben ist, ergibt sich für die zu testende Anwendung eine möglichst realitätsnahe Testumgebung, so dass nach bestandem Test der Schritt hin zum realen Einsatz sehr klein aus- oder gar ganz entfällt.

V. Referenzen

- [1] Joseph Evans, Gary Minden, K. Sam Shanmugan, Victor S. Frost, Ben Ewy, Ricardo Sanchez, Craig Sparks, Mohan Kambhammetty, Jim Roberts, Richard Plumb, and Dave Petr: “The Rapidly Deployable Radio Network”
<http://www.ittc.ukans.edu/RDRN/>
- [2] Daniel Herrscher, Kurt Roethermel, „A Dynamic Network Scenario Emulation Tool”
<http://www.informatik.uni-stuttgart.de/ipvr/vs/de/people/herrscdl/Herrscher2002-2.pdf>
- [3] William Chao, Joseph P. Macker, und Jeffrey W. Weston: “Naval Research Laboratory: Mobile Network Emulator”
<http://downloads.pf.itd.navy.mil/proteantools/MNE-1.5.pdf>
- [4] WINE GLASS: “**Wireless IP NEtwork as a Generic platform for Location Aware Service Support**”
<http://wineglass.tilab.com/>
- [5] Erik Nordström, Christian Tschudin: “APE – a Large Scale Ad Hoc Network Testbed for Reproducible Performance Tests”
<http://www.csd.uu.se/courses/course-material/xjobb/docs-reports/Nordstrom-2002.pdf>
- [6] Erik Welsh, Patrick Murphy, J.Patrick Frantz: “RUSH – A Mobile Testbed for GPS-Based ITS/IVC and Ad Hoc Routing Experimentation”
- [7] “NMEAWiz, a full-featured NMEA message generator”
<http://www.chesapeakeotech.com/nmeawiz.html>
- [8] “RECSIM III – *the* Simulator for NMEA testing”
<http://www.effective-solutions.co.uk/recsim.html>
- [9] „NemaTalker – a versatile test tool and education aid for NMEA developers”
<http://www.sailsoft.nl/NemaTalker.htm>
- [10] “GPS Simulator: a GPS Simulator for PC”
<http://www.sailsoft.nl/gpssimul.htm>
- [11] Chuck Taylor: “NMEA Server”
<http://home.hiwaay.net/~taylorc/gps/nmea-server/>
- [12] NMEA
7 Riggs Avenue
Severna Park, MD 21146
<mailto:director@nmea.org>
<http://www.nmea.org>

- [13] Peter Bennett's GPS and NMEA site
<http://vancouver-webpages.com/peter/>
- [14] Karsten Angstmann: "Automatische Positionsbestimmung – Seminar zum Studienprojekt ViLiS"
- [15] Dr. Jörg Roth: „Mobile Computing – Positionsbestimmung“
- [16] <http://www.nmea.de>

Ich versichere, dass ich diese Arbeit selbstständig verfasst und nur die angegebenen Hilfsmittel verwendet habe.

Benedict Weisshaar