Fachbereich: Landschaftsarchitektur, Geoinformatik, Geodäsie, Bauingenieurwesen
Fachgebiet: Geoinformatik

Bachelorarbeit

„Entwicklung eines virtuellen Sensors für Vermessungsaufgaben“

Zum Erlangen des akademischen Grades
„Bachelor of Engineering“ (B.Eng.)

Vorgelegt von:
Matthias Hamann

Betreuer:
Prof. Dr. -Ing. Karl Foppe
Prof. Dr. -Ing. Andreas Wehrenpfennig

Neubrandenburg, 08. September 2010

urn:nbn:de:gbv:519-thesis2010-0485-9

Eidesstattliche Erklärung

Hiermit versichere ich, Matthias Hamann, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Neubrandenburg im September 2010

Zusammenfassung

Die vorliegende Bachelorarbeit erörtert die Entwicklung eines virtuellen Sensors für Vermessungsaufgaben. Ausgangspunkt ist die Erläuterung zur geodätischen Überwachungsmessung und zur computergestützten Überwachungsmessung. Hier wird speziell das Forschungsprojekt der Hochschule Neubrandenburg dargestellt. Es handelt sich dabei um ein datenbankorientiertes Monitoringsystem, kurz DABAMOS, welches sich zur Zeit in der Entwicklungsphase befindet. Zur Validierung von DABAMOS wird der virtuelle Sensor benötigt, da er verschiedenste Messwerte simulieren kann. Zur Umsetzung des virtuellen Sensors werden anschließend verschiedene Messgeräte und Sensoren sowie Software zur Erstellung von virtuellen Sensoren vorgestellt. Des Weiteren findet eine kurze Darstellung von Messeinflüssen statt. Hierbei werden Fehler, die bei der Messung entstehen, und Messunsicherheiten, die bei der Ermittlung von Simulationsparametern eine Rolle spielen, betrachtet. Aus den behandelten Anforderungen wurde der virtuelle Sensor entworfen und implementiert. Abschließend wurden verschiedene Testläufe durchgeführt.

Abstract

This bachelor thesis discusses the development of a virtual sensor for surveying. The starting point is the explanation of geodetic measurements and computer based monitoring measurements. This is specifically displayed with the research project of the University of Neubrandenburg. This project is a database oriented monitoring system, short DABAMOS, which is currently under development. The virtual sensor is needed to validate DABAMOS because it can simulate a variety of measured values. A presentment of several measurement instruments, sensors and software to create virtual sensors has been done to show the implementation of the virtual sensor. In addition, there is a short presentation of measuring effects. This presentation shows errors that have been made during the measurements, and it also displays measurement uncertainties, which play an important role in determination of simulation parameters. The virtual sensor has been created and implemented considering all shown requirements. Last but not least, several test runs have been done.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

In der vergangenen Zeit erschütterten Unfälle und Naturkatastrophen, wie der Einsturz der Eissporthalle in Bad Reichenhall oder die Abbrüche der Kreideküste auf Rügen, der Einsturz des Kölner Stadtarchivs oder Hangrutschungen im ehemaligen Tagebaugebiet bei Nachterstedt in Sachsen-Anhalt, die zum Teil auch Opfer forderten, ständig die Welt. Diese Katastrophen machen den Bedarf an Monitoringsystemen, die eine automatisierte permanente Überwachung durchführen, deutlich. Hierbei spielt die Entwicklung von leistungsstarken Messgeräten und Sensoren sowie die Computertechnik und Kommunikationstechnologie eine tragende Rolle.

1.2 Problemstellung und Zielsetzung

Aus einem internen Forschungsprojekt der Hochschule Neubrandenburg unter der Leitung von Prof. Dr. -Ing. Karl Foppe wurde das automatische datenbankorientierte Monitoringsystem, kurz DABAMOS, entwickelt. Im Rahmen dieser Bachelorarbeit entsteht ein generischer, virtueller Sensor, der eine unabhängige Testumgebung für Monitoringsysteme, insbesondere DABAMOS, bieten soll.

1.3 Aufbau der Arbeit

Das zweite Kapitel beschäftigt sich mit den Erläuterungen zur geodätischen und computergestützten Überwachungsmessung sowie dem Monitoringsystem DABAMOS. Im dritten Kapitel findet eine Analyse der Messgeräte statt. Verschiedene Faktoren zur Messung werden im Kapitel vier beleuchtet. Hier wird besonders auf Fehler, Messunsicherheiten und Zeitabhängigkeiten eingegangen. Das fünfte Kapitel analysiert vorhandene Software-Produkte mit deren Hilfe virtuelle Sensoren erstellt werden können. Im sechsten Kapitel findet der Entwurf des virtuellen Sensors statt. Hierfür werden die speziellen Anforderungen aus den vorherigen Kapiteln aufgenommen. Das siebente Kapitel beschreibt das Vorgehen zur Implementierung des virtuellen Sensors. Hierbei werden Inhalte und Übersichten zur Konzeptionierung dargestellt. Im achten Kapitel folgt eine Validierung. Zusammenfassend wird im neunten Kapitel ein Ausblick und das Fazit gegeben.

2 Grundlagen

Dieses Kapitel erläutert die Begriffe geodätische Überwachungsmessung, computergestützte Bauwerksüberwachung und das Monitoringsystem DABAMOS.

2.1 Geodätische Überwachungsmessung

Geodätische Überwachungsmessungen, als ein Aufgabengebiet der Ingenieurgeodäsie, stellen geometrische Veränderungen eines Messobjektes fest. Hierbei werden Deformationen, das heißt Bewegungen und Verformungen erfasst, diese tragen dann zum Verständnis und Nachweis bei. Es werden sowohl künstliche, von Menschen geschaffene Bauten, als auch natürliche Objekte überwacht. Zu den typischen Bauwerken in der Ingenieurvermessung gehören Brückenbauwerke, Stauanlagen, Türme, Verkehrsanlagen, Maschinen- und Industrieanlagen. Natürliche Objekte werden oft im Zusammenhang mit anderen Wissenschaften (z.B.: Klimaforschung, Geo- und Hydrographie) betrachtet. Beispiele wären die Überwachung von geodynamischen Prozessen, wie Krustenbewegungen, Hangrutschungen und Gletscherbewegungen. Durch die Aktivität der Sonne wird die Geodynamik beeinflusst. Dies betrifft alle Objekte, die mit der Strahlung der Sonne wechselwirken. Messobjekte, speziell Gebäude, erhalten so eine spezifische Eigenbewegung, die sich mindestens in einem Tagesgang beziehungsweise Jahresgang widerspiegeln. Ausgelöst durch die Erwärmung der Baumaterialien erfolgt eine Ausdehnung und damit eine Verformung des Bauobjektes.

„Die hauptsächliche Aufgabe geodätischer Überwachungsmessungen besteht allgemein in dem Nachweis eines gegenüber den erwarteten Veränderungen abweichenden Verhaltens ausgewählter Messpunkte eines Messobjektes beziehungsweise in der Bestätigung per prognostizierten Veränderungen von Messpunkten: Mit Hilfe geodätischer Überwachungsmessungen lassen sich geometrische Beziehungen zwischen materiellen Punkten in ihrem zeitlichen Verlauf quantifizieren.“[1]
Hier wird deutlich, dass nur durch regelmäßige Überwachungsmessungen Annahmen zur geometrischen Veränderung von Messobjekten und deren Veränderungen gegenüber der Umgebung getroffen werden können. Diese sollten immer dann durchgeführt werden, wenn der Verdacht auf Deformationen am Messobjekt besteht. Die Frage nach

den Ursachen dieser zu erwarteten oder ermittelten Veränderung sollte immer in Verbindung mit der Überwachungsmessung stehen.[1]
Zusammenfassend beschreiben Welch, Heunecke und Kuhlmann im Handbuch Ingenieurgeodäsie: Auswertung geodätischer Überwachungsmessungen[1] die Zielsetzungen von geodätischen Überwachungsmessungen in folgenden Stichpunkten:

- Beitrag zum Nachweis der Funktions- und Standsicherheit eines Messobjektes, Beitrag zur Gewährleistung eines störungsfreien Betriebes
- rechtzeitiges Erfassen von Veränderungen, um Gefährdungen für das Messobjekt sowie die Umgebung zu vermeiden oder wenigstens zu mindern
- Beweissicherung zur Klärung der Ursachen von Schäden
- Möglichkeiten zur Prognose des mutmaßlichen Verhaltens in der näheren Zukunft und des Verhaltens unter bestimmten Belastungsfällen
- Überprüfung von Konstruktions- und Materialeigenschaften mit dem Ziel der Verbesserung der mechanischen Modellvorstellung über das Objekt und sein Verfahren; hieraus folgt die Möglichkeit einer verbesserten Prognose
- Erkenntnisgewinn für vergleichbare Messobjekte, insbesondere im Hinblick auf zukünftige Planungen ähnlicher Bauwerke und für die Sanierung

Geodätische Überwachungsmessungen können epochal oder permanent durchgeführt werden. Epochale Messungen werden bei näherungsweise bekanntem Bauwerksverhalten oder gleichmäßigen und langsamen Bewegungen durchgeführt. Im Gegensatz dazu werden permanente Messungen bei unregelmäßigem oder schlecht vorhersehbarem Deformationsverhalten des Messobjektes angewandt.[1]

Bei der epochalen Messung findet eine Messung in einem bestimmten zeitlichen Intervall, wie zum Beispiel alle 30 Tage statt, bei der die Messpunkte am Objekt nacheinander aufgemessen werden. Das hat zur Folge, dass die einzelnen Messwerte zu verschiedenen Zeitpunkten aufgenommen werden. Eine Auswertung der Deformation findet im Nachhinein statt und kann dann nur durch einen Vergleich der Einzelmessung

aus den vorhergehenden Epochen bewertet werden. Um diese Messung permanent durchführen zu können, bedarf es der Hilfe von computergestützten automatischen Überwachungsmessungen.[1]

Zukünftig wird die Bedeutung von computergestützten geodätischen Überwachungsmessungen steigen. Gründe sind der zunehmende Platzmangel, die Errichtung von gewagten Ingenieurbauwerken, wie Großbauwerken, oder aufwendige Sanierungsarbeiten durch zunehmendes Alter der Gebäude.[2]

2.2 Computergestützte Überwachungsmessung

Speziell die computergestützte Überwachung kombiniert die geodätische Überwachungsmessung mit Informatik, Sensorik und anderen Teilbereichen. Dabei besteht die rechentechnische Unterstützung darin, dass keine manuelle Messung im eigentlichen Sinne erfolgt, sondern diese mit Hilfe von geodätischen Messgeräten und Sensoren automatisiert durchgeführt wird. Diese Automatisierung ermöglicht die permanente Messung. Im Gegensatz zur epochalen Messung sind die Messinstrumente fest installiert und liefern regelmäßig Messergebnisse.
Damit ein zeitlicher Bezug hergestellt werden kann, werden alle Messungen zu Punkten am Messobjekt mit einem Zeitstempel versehen. Parallel dazu werden alle nötigen äußeren Einflüsse, wie z.B. Luftdruck, Luftfeuchtigkeit, Lufttemperatur und Verkehrsaufkommen mitbestimmt. Diese Messwerte können anschließend miteinander korreliert werden. Ziel ist es, ein Modell für das Bewegungsverhalten im Zusammenhang mit dem Messobjekt und seiner Umwelt herzustellen.
Im Idealfall (siehe Abbildung 1) besteht ein automatisches Überwachungsmesssystem aus Messstationen, die mit einer Reihe von Sensoren verbunden sind. Diese erzeugen kontinuierlich Messwerte und kommunizieren mit den Messstationen, die bei Stromausfall weiter arbeiten, um einen Datenverlust zu vermeiden. Die dabei anfallenden Daten werden zwischengespeichert und über eine beliebige Verbindung an einen Server geschickt. Die Datenspeicherung sollte unabhängig vom Messort geschehen. Die Datenübertragung muss ebenfalls fehlerfrei und stabil ablaufen.[3]

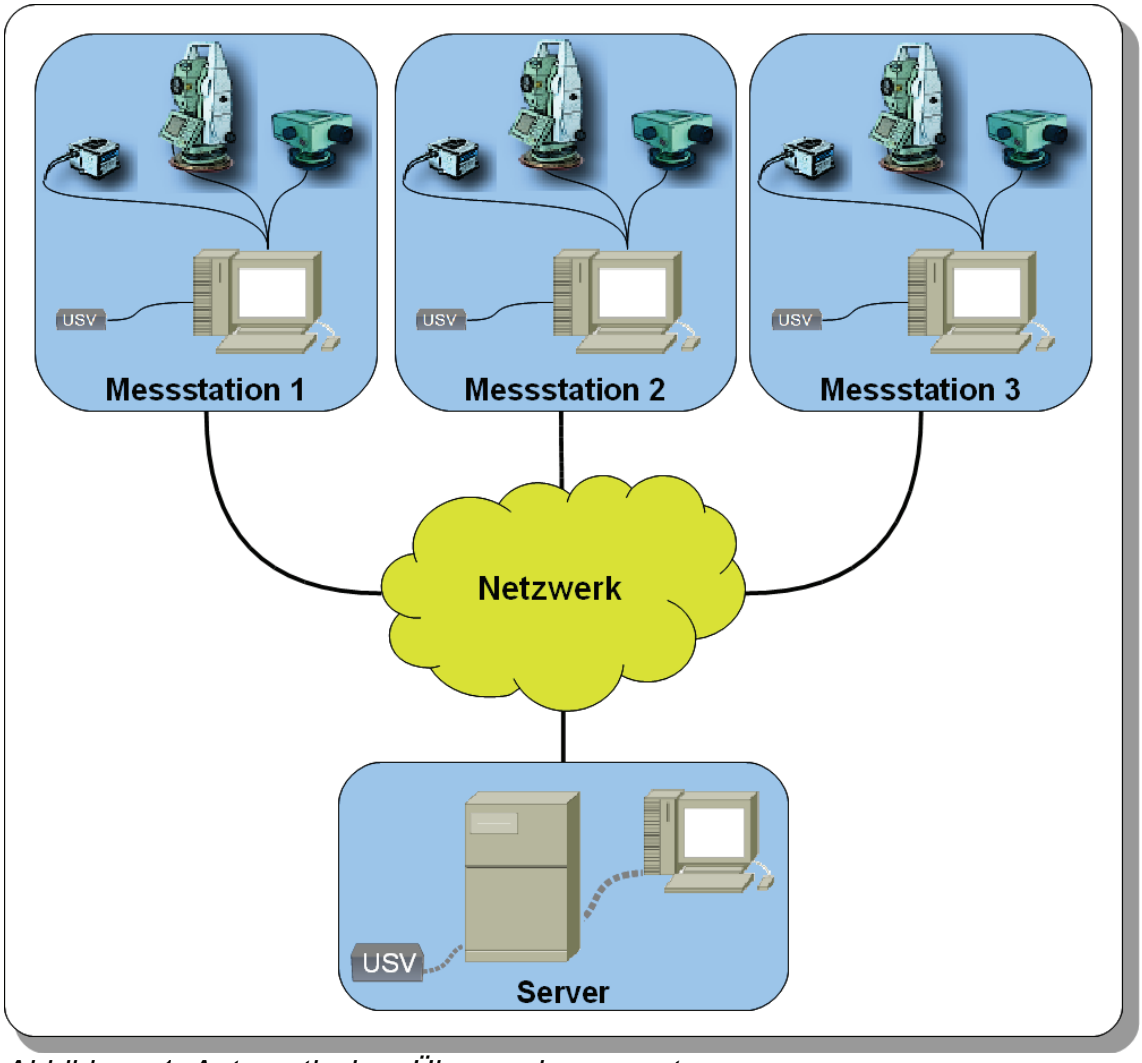

Abbildung 1: Automatisches Überwachungssystem

Vorteil eines solchen Systems ist, dass die Daten sofort digital vorliegen und in Echtzeit ausgewertet werden können. Dies ist eine Grundvoraussetzung für ein in Echtzeit funktionierendes Überwachungssystem mit Alarmierungsfunktion, welche z.B. per SMS realisiert werden kann. Die eingesetzten geodätischen Messgeräte und Sensoren variieren je nach Überwachungsaufgabe. Gleichzeitig müssen die Kosten und damit die Art und Anzahl der Sensoren auf ein vertretbares Maß begrenzt werden.[3]

Um computergestützte Überwachungssysteme entwickeln zu können, bedarf es einer Testumgebung, die die Validierung der Software ermöglicht. Relevante Punkte sind Performance- und Geschwindigkeitstests sowie Tests zur Stabilität und Datenverarbeitung. Ein Teil dieser Testumgebung kann mit einem virtuellen Sensor realisiert werden. Virtuelle Sensoren, oder auch Softsensoren, sind Softwaresysteme, die sich wie

Sensoren verhalten. Sie detektieren keine physikalischen Gegebenheiten von Objekten, sondern generieren anhand eines mathematischen Modells Messwerte.

2.3 DABAMOS

DABAMOS als ein Projekt der Hochschule Neubrandenburg unter Leitung von Prof. Dr. -Ing. Karl Foppe ist ein generisches computergestütztes Monitoringsystem, welches in erster Linie zur Überwachung von Messobjekten gedacht ist. Da es über eine XML-Datei konfigurierbar ist, lassen sich verschiedene Typen von Sensoren einstellen und konfigurieren. Dadurch kann DABAMOS nicht nur geodätische Sensoren ansteuern, sondern alle Sensoren bei denen die Kommunikationsstrings bekannt sind. Die Datenübertragung findet über serielle Schnittstellen wie RS232, RS485, USB oder Netzwerkschnittstellen statt.

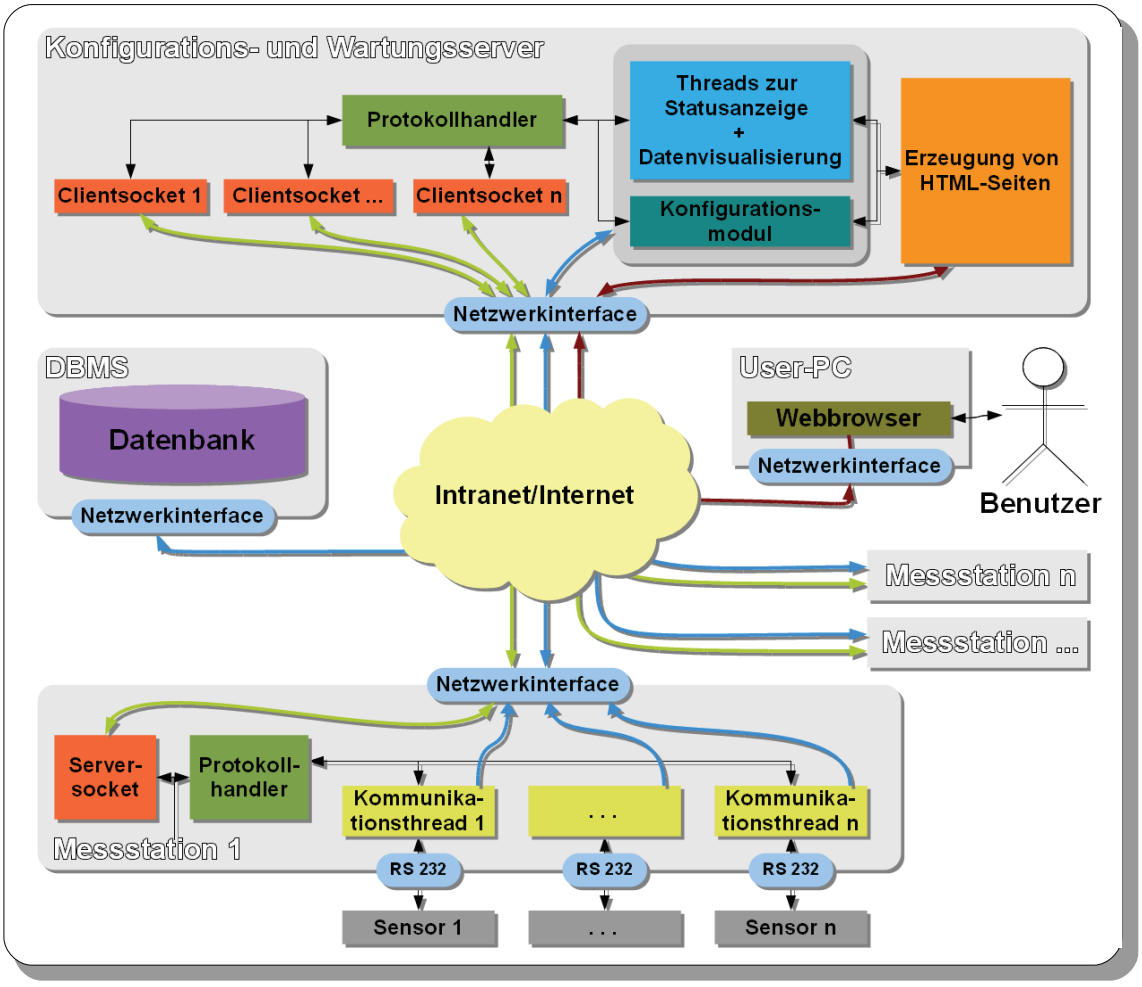

Abbildung 2: Aufbau DABAMOS (Wolff 2010)[4]

DABAMOS besteht (siehe Abbildung 2) aus dem Konfigurations- und Wartungsserver, welcher der Erzeugung der grafischen Nutzerschnittstelle dient. Diese Schnittstelle wird durch HTML-Seiten realisiert und auf dem User-PC über einen beliebigen Webbrowser aufgerufen. Von dort aus finden sämtliche Konfigurationen und Interaktionen mit den Messstationen statt. Die einzelnen Messstationen können mehrere Sensoren ansteuern. Die erzeugten Daten werden sowohl in einer Datenbank, die sich auf der Messstation befindet, gespeichert als auch in einer Datenbank, die sich an einem beliebigen Ort befindet. Grund dafür ist das Vermeiden von Datenverlusten, die bei einem Verbindungsabbruch entstehen. Die Kommunikation findet über das Netzwerk oder das Internet statt. Dadurch findet eine räumliche Trennung der Einzelkomponenten statt. Mehr Details dazu in Christian Wolff, Entwicklung eines Webinterfaces zur Fernwartung von Überwachungsmesssystemen, 2010.[4]

3 Darstellung physikalischer Messgeräte und Sensoren

In diesem Kapitel werden Messgeräte und ihre Funktion sowie Kommunikation beschrieben.

3.1 Geodätische Messgeräte

Geodätische Messgeräte und Sensoren quantisieren physikalische Vorgänge. Es gibt Messgeräte die Strecken, Winkel und Richtungen, Höhen, Koordinaten und andere meteorologische Größen wie Lufttemperatur, Luftdruck und Luftfeuchtigkeit messen. Diesbezüglich stehen Instrumente wie Tachymeter, Nivel, Nivelliere, Klimamessgeräte, meteorologische Instrumente oder GPS Messgeräte zur Verfügung, die aus einer Kombination von verschiedenen aufeinander abgestimmten Sensoren bestehen. Dabei ist jeder Sensor von unterschiedlichen physikalischen Einflüssen abhängig, die wiederum das gesamte System beeinflussen.[5]

Die bekanntesten Hersteller geodätischer Messgeräte sind Leica Geosystems AG, Trimbel Navigation Ltd. und Topcon AG. Meteorologische Instrumente werden unter anderem von REVUE THOMMEN AG, E+E Elektronik GmbH, Ammonit Gesellschaft für Messtechnik mbH hergestellt.

Abhängig vom Budget, den Entfernungen zum Messcomputer und der Übertragungsrate gibt es verschiedene Übertragungsmedien und -techniken. Kabelgebundene Sensoren werden in der Regel über RS232, RS485, USB oder über ein LAN betrieben. Funkbasierte Übertragungen können über ein lokales WLAN oder WiMax realisiert werden. Dank des immer besseren Mobilfunknetzes ist es ebenso möglich die anfallenden Daten über GSM, GRPS, UMTS oder HSDPA zu verschicken. Die Kommunikation findet in der Regel über eine Befehlsstruktur, bestehend aus Präfix, Befehl und Suffix, statt.[5]

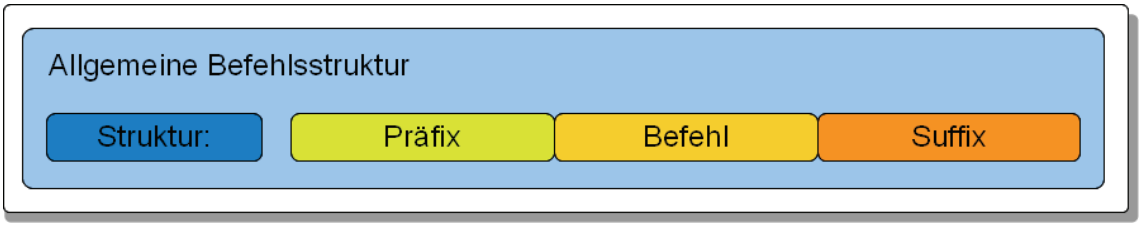

Abbildung 3: Allgemeine Befehlsstruktur

3.1.1 Elektronische Tachymeter: Leica TPS1200

Das elektronische Tachymeter ist ein komplexes Messgerät, welches aus einer Reihe von Sensoren besteht. Es kombiniert die Winkel- und Streckenmessung und ermittelt dadurch Punkte durch polares Anhängen.

Das elektronische Tachymeter findet in der automatischen Überwachungsmessung Anwendung. Die Voraussetzungen dafür, ergeben aus der computergestützten Fernsteuerung der Messgeräte. Ausgangspunkt für eine automatische Zielpunkterfassung ist ein motorisiertes Tachymeter. Es muss sich über einen Computer bedienen lassen, dies wird bei Leica durch eine serielle Verbindung wie RS232/USB realisiert. Zur Durchführung von Präzisionsmessungen sollte es sich um ein Präzisionstachymeter handeln. Die Genauigkeitsanforderungen der Präzisionsvermessung sind Streckenmessgenauigkeiten von 2mm + 2ppm und Winkelmessgenauigkeiten von 0,15mgon (0,5“). Weiterhin sollte das Tachymeter über eine „Automatic Target Recognition“ – kurz ATR – verfügen, um eine Feinzielerfassung gewährleisten zu können.

Aus diesen Voraussetzungen eignen sich folgende Totalstationen der Serie TPS1200 der Firma Leica: TCA, TCP, TCRA, TCRP.[5]

Die Kommunikation findet über RS232 mittels Geocom-Schnittstelle statt. Es wird eine vorgegebene Aneinanderreihung von ASCII-Zeichen an das Messgerät übertragen, wodurch eine Aktion des Messgerätes folgt. Das Messinstrument antwortet ebenfalls in einem vorgegebenen Syntax.[6]

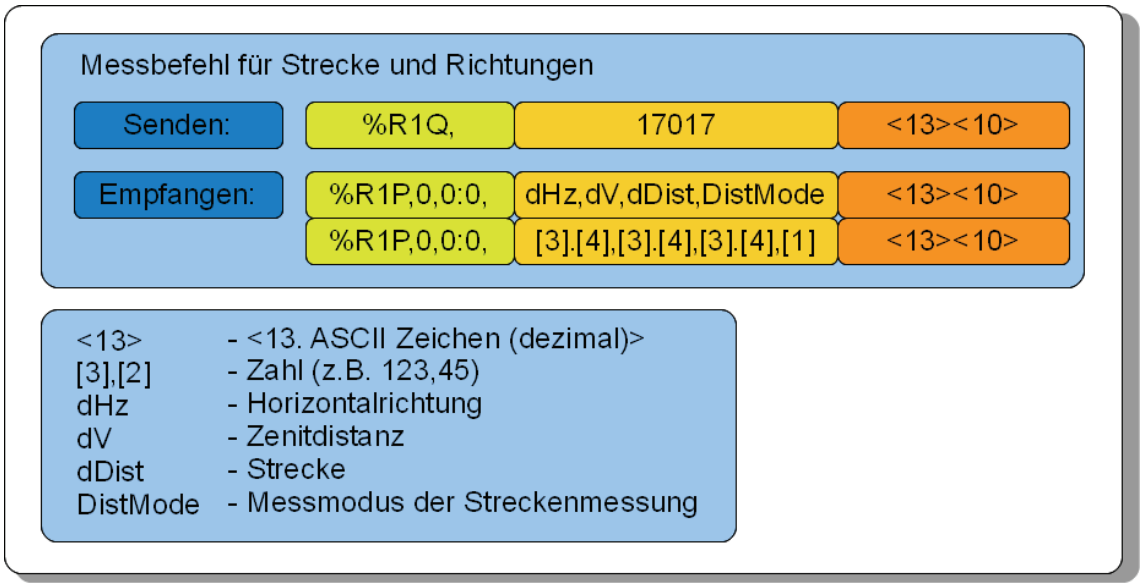

Abbildung 4: Messbefehl für Strecke und Richtungen

3.1.2 Neigungsmessgerät: Leica NIVEL210

Neigungsmessgeräte messen Winkel zwischen einer Bezugsrichtung und der wahren Lotrichtung. Dabei gibt es verschiedene Systeme: Pendel- und Flüssigkeitssysteme. Bei Flüssigkeitssystemen wird das Verhalten eines Flüssigkeitshorizontes mithilfe von optischen, induktiven, kapazitiven oder resistiven Bauelementen ermittelt. Pendelsysteme hingegen detektieren eine Differenz zwischen einer Bezugsrichtung im Gerät und der wahren Lotrichtung.[7]

Das NIVEL210 von Leica ist ein hochpräzises Neigungsmessinstrument, dass Neigungen mittels elektrooptischem Prinzip in zwei zueinander orthogonalen Richtungen mit Hilfe eines Flüssigkeitshorizontes misst. Dabei beträgt die Auflösung 0,001mrad. Die Kommunikation wird per RS232 oder im Verbund von mehreren NIVEL220 mittels RS485 realisiert. Die Standardabweichung des Messgerätes im Bereich von -1,51mrad bis +1,51mrad beträgt ±0,0047mrad. Die maximal zu erreichende Messfrequenz liegt bei 50 Hertz.[8]

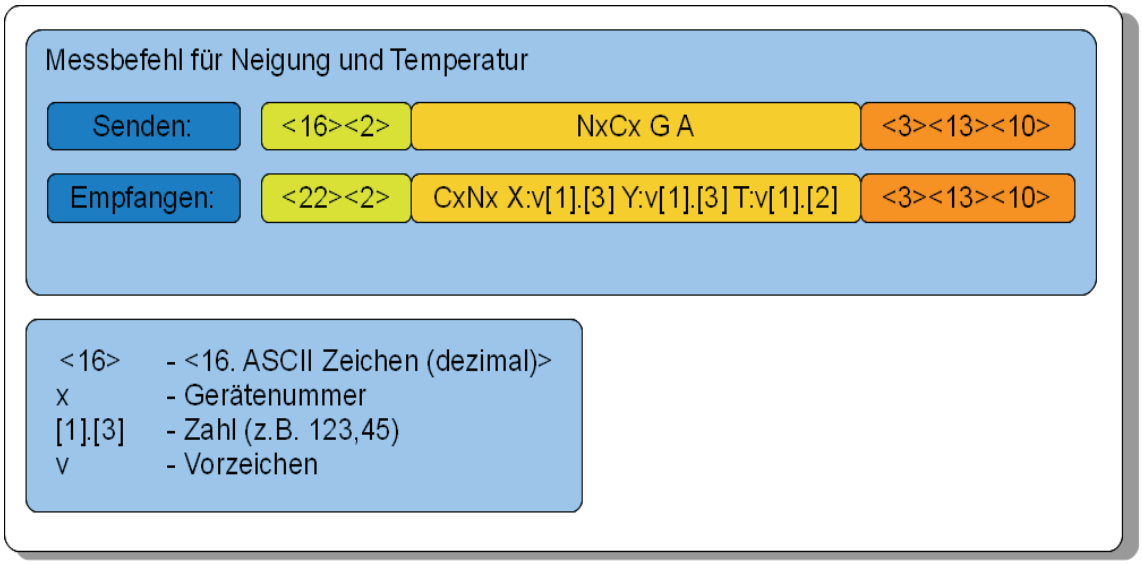

Abbildung 5: Messbefehl für Neigung und Temperatur

3.1.3 Nivelliere: Leica DNA03 und DNA10

Bei der Überwachungsmessung kommen Nivelliere für das Detektieren von Höhenunterschieden zum Einsatz. Es werden Höhenpunkte am Messobjekt vermarkt, die dann permanent mit einem Nivellier überprüft werden. Aus den entstehenden Differenzen kann das Bewegungsverhalten des Messpunktes am Messobjekt quantifiziert werden.
Bei automatischen Überwachungsmessungen wird eine Kommunikation des Nivelliergerätes mit dem Computer vorausgesetzt. Die Messung wird durch einen Messbefehl ausgelöst, der eine Antwort des Messgerätes zur Folge hat.
Die Digitalnivelliere, DNA03 und DNA10, von der Firma Leica eignen sich bei einem Messbereich von 1,8m -110m Reichweite. Der Hersteller gibt eine Standardabweichung von 0,3mm (DNA03) beziehungsweise 0,9mm (DNA10) bei einer elektronischer Messung mit Invarlatte pro 1km Doppelnivellement (ISO17123 2) an. Die Kommunikation erfolgt per GSI Format via RS232.[9]
Als Nachteil stellt sich eine fehlende Motorisierung heraus, dadurch lässt sich pro Messgerät nur ein Messpunkt überwachen.

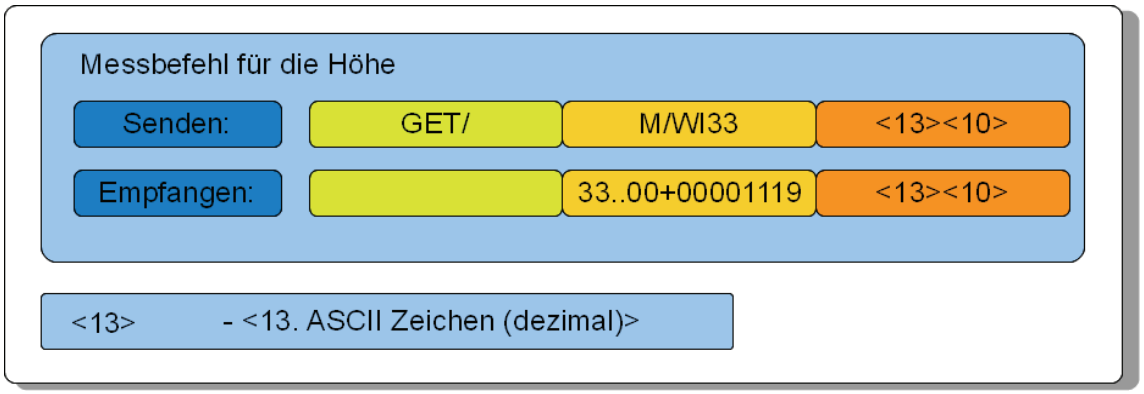

Abbildung 6: Messbefehl für die Höhe

3.2 Meteorologische Sensoren

Die typischen meteorologischen Sensoren in der Vermessungstechnik sind Kombinationen aus mehreren Sensoren. Relevante klimatische Daten für Überwachungsmessungen sind Luftdruck, Lufttemperatur und Luftfeuchtigkeit, Windgeschwindigkeit und -richtung sowie Niederschlagsmenge.
Mit der THOMMEN Meteo-Station HM30 lassen sich klimatische Daten, wie Luftdruck, Luftfeuchte und Lufttemperatur erfassen.[10]

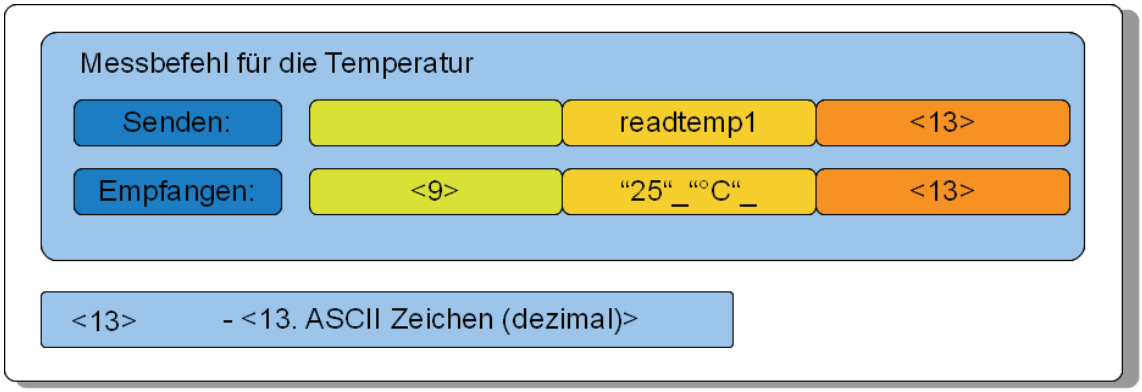

Abbildung 7: Messbefehl für die Temperatur

3.3 Fazit

Alle Messgeräte von Leica werden über RS232 angesteuert und benutzen zur Kommunikation ASCII-Zeichenketten. Die Struktur dieser Zeichenketten ist hersteller- und geräteabhängig. Alle Messgeräte kommunizieren nach dem gleichen Prinzip. Es wird ein Befehl an das Messgerät geschickt und erhält, sofern es sich um ein Messbefehl handelt, eine Antwort. Das Nivel beherrscht zu dem einen Modus, in dem kontinuierlich Messwerte geliefert werden. Die Messgeräte haben unterschiedliche Messgeschwindigkeiten. Der Nivel zum Beispiel misst mit 50Hz (Herstellerangabe) im kontinuierlichen Modus und hat damit eine höhere Messgeschwindigkeit als das Tachymeter und das Nivelliergerät. Die Messgeräte unterliegen unterschiedlichen Fehlereinflüssen. Messunsicherheiten setzen sich aus den einzelnen Messunsicherheiten der Komponenten zusammen. Die Messgrößen unterliegen unterschiedlichen zeitlichen Abhängigkeiten, die aus Umwelteinflüssen hervorgehen.

4 Betrachtung von Messeinflüssen

Dieses Kapitel beschäftigt sich mit Messeinflüssen, dabei werden Fehler und Fehlerarten sowie die Messunsicherheit betrachtet. Die Ermittlung wird anhand eines Beispiels zur Kovarianzfortpflanzung erläutert. Am Ende des Kapitels werden zeitabhängige Messgrößen, speziell der Tages-, der Jahresgang sowie der Trend beschrieben.

4.1 Messabweichungen

Wegen der bei der Messungen wirkenden Einflüsse treten unvermeidlich Messabweichungen auf.[11]

Damit die Abweichungen minimiert werden können, muss eine regelmäßige Messmittelüberwachung stattfinden. Dazu werden die Messgeräte kalibriert, justiert oder geeicht. Kalibrieren ist das Ermitteln des Zusammenhangs zwischen dem gemessenen Wert und dem tatsächlichen Wert der Messgröße. Kalibrieren nach dem Eichgesetz ist Eichen. Justieren ist die Maßnahme zur Minimierung der Abweichung zwischen Messwert und dem tatsächlichen Wert. Es kann mechanisch eingegriffen werden. Bei Geräten mit elektronischer Datenverarbeitung kann eine „softwaremäßige Justierung“ erfolgen. Hierbei werden Korrekturwerte „offsets“ in die Gerätesoftware eingegeben. Das Messgerät berücksichtigt diese Korrekturwerte während der Messung.[5]

Die Auflösung eines Messgerätes beschreibt, in wie weit ein Messinstrument in der Lage ist, zwei eng benachbarte Messwerte getrennt zu erfassen und darzustellen. Bei einem digital anzeigenden Instrument entspricht die Auflösung einer Einzelmessung der kleinsten Stelleneinheit.[11]

4.2 Fehlerarten und Fehler

Nach DIN Norm 1319 wird unter Fehler die Differenz zwischen dem gemessenen Wert und dem tatsächlichen Wert einer Messgröße verstanden. Fehler werden in verschiedene Fehlerarten unterschieden:

- grobe Fehler
- systematische Fehler
- zufällige Fehler

Grobe Fehler entstehen in der Regel durch Fehlverhalten des Benutzers während der Messung. Sie übersteigen die Standardabweichung mindestens um das zwei- bis dreifache und sind daher leicht zu detektieren. Systematische Fehler sind Fehler, die sich bei jedem Messvorgang erneut und gleich auswirken und daher schwer feststellbar sind. Sie können zum Beispiel bei Nichtberücksichtigung der Eigenerwärmung des Messgerätes, oder durch Anbringen falscher Kalibrierparameter entstehen. Zufällige Fehler sind Streuungen, um den wahren Wert. Sie können, sofern die Streuung nicht einseitig ist, durch mehrfaches Messen und anschließendes Mitteln der Messwerte eliminiert werden.[5]
Die Zusammensetzung eines Messwertes wird anhand der Normalverteilung erläutert:

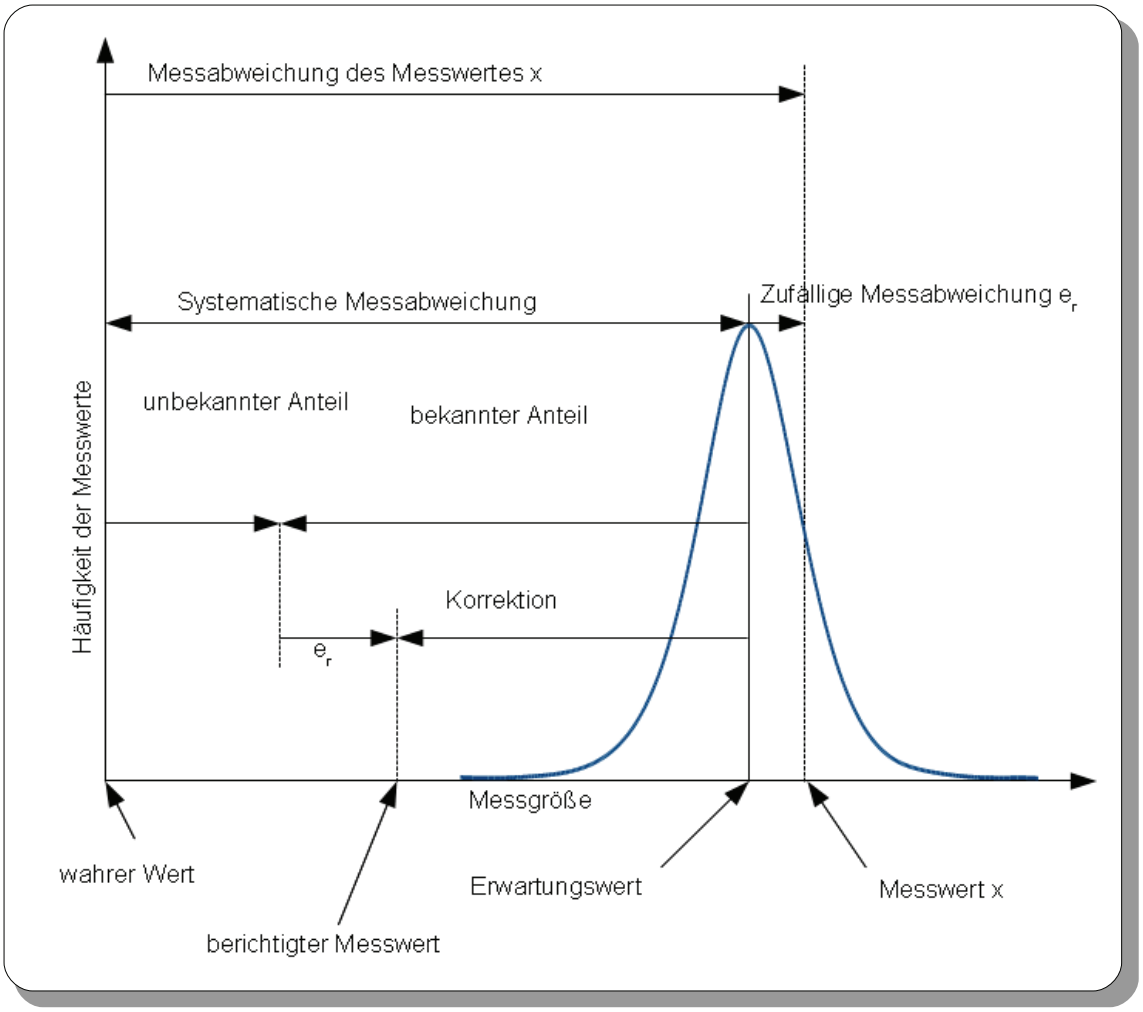

Abbildung 8: Messabweichung[11]

Der wahre Wert einer Messgröße ist unbekannt. Der wahre Wert ist der reale Wert, der bei der Ermittlung unter vorherrschenden Bedingungen entsteht. Der richtige Wert nähert den wahren Wert an, dabei ist die Abweichung vernachlässigbar klein. Die Genauigkeit ist als Messabweichung des Messwertes vom richtigen Wert definiert. Systematische und zufällige Abweichung bilden zusammen die Messabweichung. Durch mehrfach Messen streuen sich die Messwerte, um den Erwartungswert. Die arithmetische Mittlung der Beobachtungen nähert den Erwartungswert an. Das hat zur Folge, dass zufällig wirkende Abweichungen minimiert werden. Die Standardabweichung wird als Kenngröße für die Streuung angegeben. Die systematische Abweichung begrenzt die erreichbaren Genauigkeiten von den meisten geodätischen Messungen, da sie nur unvollständig bekannt ist. Sie führen nach Korrektion zum berichtigten Messwert.[11]

4.3 Messunsicherheit

Die Messunsicherheit ist ein Kennwert für die Streuung der Messwerte, die einer Messgröße zugeordnet werden können. Diese wird als Intervall angegeben, welches sich aus der Gesamtheit der Einzelunsicherheiten, der bei der Messung auftretenden Faktoren, ergibt. Im Allgemeinen wird von zufällig streuenden Messwerten ausgegangen, die durch die Varianz gekennzeichnet sind. Beobachtungen können auch weitere, oft systematisch wirkende Anteile haben.
Dadurch, dass die Messgrößen nicht direkt ermittelt werden, sondern aus Beobachtungen abgeleitet werden, stellen sich die Fragen[12]:

- Wie stark wirken sich die Abweichungen von den Beobachtungen auf die abgeleiteten Messgrößen aus?
- Mit welchem Fehler sind die abgeleiteten Größen behaftet?

Daher gilt es ein repräsentatives Genauigkeitsmaß abzuschätzen. Durch die Weiterentwicklung der Sensoren werden interne Messprozesse immer unüberschaubarer und es lässt sich kein verlässliches Maß für die tatsächliche Genauigkeit der Messgröße erhalten. Selbst durch Wiederholungsmessungen ist die Präzision kaum zu erfassen. Wesentlich ist, dass systematisch wirkende Einflüsse die heutigen Messergebnisqualitäten wesentlich stärker beeinflussen als die zufälligen Anteile. Methodische Ansätze des GUM - Guide to the Expression of Uncertainty in Measurement - setzen sich aus

mehreren Komponenten zusammen , die in 2 Kategorien eingeteilt sind:
Die Komponente A, die mit statistischen Komponenten ermittelt werden. Diese werden durch die Varianzschätzung und deren Freiheitsgraden angegeben. Die Bestimmung erfolgt durch Mittelbildung, Ausgleichung und Varianzfortpflanzung. Die Standardunsicherheiten ist als $s_i = u_{Ai}$ bestimmt.
Die Komponente B wird auf eine andere Weise ermittelt. Hierbei findet eine Abschätzung der Einflussfaktoren wie systematische Effekte und weitere Messergebnis beeinflussende externe Faktoren (z.B. Zentrierunsicherheit, Refraktion) statt. Diese werden als Näherungen entsprechender Standardabweichungen betrachtet und durch Größen u_{Bi} charakterisiert. Beispiele für eine elektrooptische Distanzmessung zur Ableitung einer Horizontalstrecke sind die externen Effekte:

- instrumentelle Effekte, wie Alterung, Temperaturabhängigkeit, Phaseninhomogenität
- Kalibrierung, wie Maßstabsfaktor, Additionskorrektur, zyklische Effekte, zeitliche Gültigkeit
- Punktdefinition und Punktaufbau, wie Qualität der Vermarkung, Zentriergenauigkeiten, Horizontierung
- meteorologische Parameter, wie Erfassung der Meteo-Parameter, Modell der Berechnung eines repräsentativen Berechnungsindizes[13][14]

Die Unsicherheit u_C setzt sich aus den Unsicherheiten vom Typ A und Typ B unter Anwendung des Varianzfortpflanzungsgesetzes für unkorrelierte Größen zusammen, dabei ist u_C stets positiv:

$$u_C = \sqrt{u_{Ai}^2 + u_{Bi}^2}$$

In der weiterführenden Berechnung werden ausschließlich die zufälligen Anteile betrachtet. Die Ermittlung der Varianz kann über eine Ausgleichungsrechnung geschehen, die sich aus dem stochastischen und dem funktionalen Modell zusammensetzt. Das stochastische Modell besteht aus einer Kovarianzmatrix der Beobachtungen.

Kovarianzmatrix der Beobachtungen *Beobachtungsvektor*

$$\Sigma_{ll}=\begin{bmatrix}\sigma_1^2 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \sigma_n^2\end{bmatrix} \qquad \underset{n,1}{L}=\begin{bmatrix}D\\ t\\ V\end{bmatrix}$$

Die Kovarianzmatrix ist eine quadratische Matrix. Die Hauptdiagonale enthält die Varianzen der Beobachtungen. Anstelle der „0“ auf der Nebendiagonalen können die Korrelationen zwischen den Beobachtungen eingesetzt werden. Oftmals ist dies aber aus Gründen von mangelnden Informationen nicht gegeben.

Das funktionale Modell beschreibt Abhängigkeiten der Beobachtungsunbekannten. Im Allgemeinen ist dieser Zusammenhang nicht linear. Grund dafür ist, dass in der Geodäsie häufig trigonometrische Funktionen und andere nichtlineare Funktionen genutzt werden. Die Ausgleichungsrechnung funktioniert ausschließlich in einem linearen Modell, sodass eine partielle Differenzierung und Linearisierung der Funktionen statt finden muss.[12]

Funktionaler Zusammenhang $\underset{u,1}{X}=\underset{u,1}{\phi}\,(\underset{n,1}{L})$ *mit* $\underset{n,1}{L}\begin{bmatrix}D\\ t\\ V\end{bmatrix}$

$$\begin{bmatrix}y_p\\ x_p\\ z_p\end{bmatrix}=\begin{bmatrix}\phi_1(L)\\ \phi_2(L)\\ \phi_3(L)\end{bmatrix}=\begin{bmatrix}f_1(D,t,V)\\ f_2(D,t,V)\\ f_3(D,t,V)\end{bmatrix}$$

Allgemeines Kovarianzfortpflanzungsgesetz

$$\Sigma_{\hat{x}\hat{x}}=\underline{F}\cdot\underline{\Sigma}_{ll}\cdot\underline{F}^T$$

Nach Ausgleichung liefert sie eine Genauigkeitsangabe, die sich aus den Unsicherheiten der einzelnen Komponenten ergibt.

Kovarianzmatrix der Beobachtungen

$$\underset{u,u}{\Sigma}_{XX}=\begin{bmatrix}\underline{\sigma^2_{y_p}} & \sigma_{y_p x_p} & \sigma_{y_p z_p}\\ \sigma_{x_p y_p} & \underline{\sigma^2_{x_p}} & \sigma_{x_p z_p}\\ \sigma_{z_p y_p} & \sigma_{z_p x_p} & \underline{\sigma^2_{z_p}}\end{bmatrix}$$

Zur Ermittlung der Varianz der einzelnen Koordinatenkomponenten werden die Wurzeln der Werte, die sich auf der Kovarianzmatrix-Hauptdiagonalen der Beobachtungen befindet, gezogen.

Varianz der Koordinate

$$\sigma_{yp} = +\sqrt{\sigma_{yp}^2}$$
$$\sigma_{xp} = +\sqrt{\sigma_{xp}^2}$$
$$\sigma_{zp} = +\sqrt{\sigma_{zp}^2}$$

Im folgenden wird ein Beispiel zur Kovarianzfortpflanzung gegeben.
Die bei der Messung mit dem Tachymeter ermittelten Messwerte sind nur endlich genau bestimmbar. Deshalb werden den einzelnen Messwerten Genauigkeitsmaße zugeordnet.

gegeben:

Messwerte vom Tachymeter

$t = 54{,}5678\ gon \quad \sigma_t = 2{,}0\ mgon$
$D = 758{,}345\ m \quad \sigma_D = 15\text{mm}$
$V = 98.3456\text{gon} \quad \sigma_V = 3{,}0\ mgon$

Standpunktkoordinaten

$y_0 = 0$
$x_0 = 0$
$z_0 = 0$

Berechnung der Koordinaten

$$y_p = y_0 + D \cdot \sin(V) \cdot \sin(t)$$
$$x_p = x_0 + D \cdot \sin(V) \cdot \cos(t)$$
$$z_p = z_0 + D \cdot \cos(V)$$

$n = 3$

Beobachtungsvektor

$$\underset{n,1}{L} = \begin{bmatrix} D \\ t \\ V \end{bmatrix} = \begin{bmatrix} 758{,}345\ m \\ 54{,}5678\ gon \\ 98{,}3456\ gon \end{bmatrix}$$

Kovarianzmatrix der Beobachtungen

$$\underset{n}{\Sigma_{LL}}, n = \begin{bmatrix} \underline{\sigma_D^2} & \sigma_{Dt} & \sigma_{DV} \\ \sigma_{tD} & \underline{\sigma_t^2} & \sigma_{tV} \\ \sigma_{VD} & \sigma_{Vt} & \underline{\sigma_V^2} \end{bmatrix} = \begin{bmatrix} 225\text{mm}^2 & 0 & 0 \\ 0 & 4\text{mgon}^2 & 0 \\ 0 & 0 & 9\text{mgon}^2 \end{bmatrix}$$

gesucht:

$u = 3$

Unbekannten Vektor *Kovarianzmatrix der Beobachtungen*

$$\underset{u,1}{X} = \begin{bmatrix} y_p \\ x_p \\ z_p \end{bmatrix} \qquad \underset{u,u}{\Sigma_{XX}} = \begin{bmatrix} \underline{\sigma^2_{y_p}} & \sigma_{y_p x_p} & \sigma_{y_p z_p} \\ \sigma_{x_p y_p} & \underline{\sigma^2_{x_p}} & \sigma_{x_p z_p} \\ \sigma_{z_p y_p} & \sigma_{z_p x_p} & \underline{\sigma^2_{z_p}} \end{bmatrix}$$

Funktionaler Zusammenhang $\underset{u,1}{X} = \underset{u,1}{\phi}\,\underset{n,1}{(L)}$ *mit* $\underset{n,1}{L} \begin{bmatrix} D \\ t \\ V \end{bmatrix}$

$$\begin{bmatrix} y_p \\ x_p \\ z_p \end{bmatrix} = \begin{bmatrix} \phi_1(L) \\ \phi_2(L) \\ \phi_3(L) \end{bmatrix} = \begin{bmatrix} f_1(D,t,V) \\ f_2(D,t,V) \\ f_3(D,t,V) \end{bmatrix} \Leftrightarrow \begin{bmatrix} y_0 + D \cdot \sin(V) \cdot \cos(t) \\ x_0 + D \cdot \sin(V) \cdot \sin(t) \\ z_0 + D \cdot \cos(V) \end{bmatrix} = \begin{bmatrix} 159{,}3687 \\ -360{,}4139 \\ 647{,}9125 \end{bmatrix} [m]$$

Die 3D-Punktkoordinaten wurden durch polares Anhängen, aus einer Strecke und zwei Richtungen berechnet.

Differentialquotienten (partielle Ableitungen)

$$\frac{\partial \phi_1(L)}{\partial l_1} = \frac{\partial y_p}{\partial D} = \sin(V) \cdot \cos(t)$$
$$\frac{\partial \phi_1(L)}{\partial l_2} = \frac{\partial y_p}{\partial t} = -D \cdot \sin(V) \cdot \sin(t)$$
$$\frac{\partial \phi_1(L)}{\partial l_3} = \frac{\partial y_p}{\partial V} = D \cdot \cos(V) \cdot \cos(t)$$

$$\frac{\partial \phi_2(L)}{\partial l_1} = \frac{\partial x_p}{\partial D} = \sin(V) \cdot \sin(t)$$
$$\frac{\partial \phi_2(L)}{\partial l_2} = \frac{\partial x_p}{\partial t} = D \cdot \sin(V) \cdot \cos(t)$$
$$\frac{\partial \phi_2(L)}{\partial l_3} = \frac{\partial x_p}{\partial V} = D \cdot \cos(V) \cdot \sin(t)$$

$$\frac{\partial \phi_3(L)}{\partial l_1} = \frac{\partial z_p}{\partial D} = \cos(V)$$
$$\frac{\partial \phi_3(L)}{\partial l_2} = \frac{\partial z_p}{\partial t} = 0$$
$$\frac{\partial \phi_3(L)}{\partial l_3} = \frac{\partial z_p}{\partial V} = -D \cdot \sin(V)$$

Funktionalmatrix

$$\underset{u,n}{F}=\begin{bmatrix}\frac{\partial\phi_1(L)}{\partial l_1} & \frac{\partial\phi_1(L)}{\partial l_2} & \frac{\partial\phi_1(L)}{\partial l_3}\\ \frac{\partial\phi_2(L)}{\partial l_1} & \frac{\partial\phi_2(L)}{\partial l_2} & \frac{\partial\phi_2(L)}{\partial l_3}\\ \frac{\partial\phi_3(L)}{\partial l_1} & \frac{\partial\phi_3(L)}{\partial l_2} & \frac{\partial\phi_3(L)}{\partial l_3}\end{bmatrix}=\begin{bmatrix}\sin(V)\cdot\cos(t) & -D\cdot\sin(V)\cdot\sin(t)\cdot\frac{1}{\rho} & D\cdot\cos(V)\cdot\cos(t)\cdot\frac{1}{\rho}\\ \sin(V)\cdot\sin(t) & D\cdot\sin(V)\cdot\cos(t)\cdot\frac{1}{\rho} & D\cdot\cos(V)\cdot\sin(t)\cdot\frac{1}{\rho}\\ \cos(V) & 0 & -D\cdot\sin(V)\cdot\frac{1}{\rho}\end{bmatrix}$$

$$mit\ \rho=\frac{200\text{gon}}{\pi}$$

Allgemeines Kovarianzfortpflanzungsgesetz

$$\Sigma_{XX}=F\cdot\Sigma_{LL}\cdot F^T$$

$$\begin{bmatrix}\underline{\sigma^2_{y_p}} & \sigma_{y_px_p} & \sigma_{y_pz_p}\\ \sigma_{x_py_p} & \underline{\sigma^2_{x_p}} & \sigma_{x_pz_p}\\ \sigma_{z_py_p} & \sigma_{z_px_p} & \underline{\sigma^2_{z_p}}\end{bmatrix}=\begin{bmatrix}0{,}6544 & -0{,}9121 & 0{,}0205\\ 0{,}7557 & 0{,}7898 & 0{,}0237\\ 0{,}0260 & 0 & -1{,}2065\end{bmatrix}\cdot\begin{bmatrix}2{,}25\cdot10^{-4}m^2 & 0 & 0\\ 0 & 4\cdot10^{-6}gon^2 & 0\\ 0 & 0 & 9\cdot10^{-6}gon^2\end{bmatrix}\cdot\begin{bmatrix}0{,}6544 & 0{,}7557 & 0{,}0260\\ -0{,}9121 & 0{,}7898 & 0\\ 0{,}0205 & 0{,}0237 & -1{,}2065\end{bmatrix}$$

Ergebnis der Kovarianzfortpflanzung

$$\Sigma_{XX}=\begin{bmatrix}\underline{\sigma^2_{y_p}} & \sigma_{y_px_p} & \sigma_{y_pz_p}\\ \sigma_{x_py_p} & \underline{\sigma^2_{x_p}} & \sigma_{x_pz_p}\\ \sigma_{z_py_p} & \sigma_{z_px_p} & \underline{\sigma^2_{z_p}}\end{bmatrix}=\begin{bmatrix}9{,}9685\cdot10^{-5} & 1{,}1416\cdot10^{-4} & 3{,}6056\cdot10^{-6}\\ 1{,}1416\cdot10^{-4} & 1{,}3099\cdot10^{-4} & 4{,}1635\cdot10^{-6}\\ 3{,}6056\cdot10^{-6} & 4{,}1635\cdot10^{-6} & 1{,}3253\cdot10^{-5}\end{bmatrix}$$

Varianz der Koordinate

$$\sigma_{yp}=\sqrt{\sigma^2_{yp}}=9{,}9842\,mm$$
$$\sigma_{xp}=\sqrt{\sigma^2_{xp}}=11{,}4451\,mm$$
$$\sigma_{zp}=\sqrt{\sigma^2_{zp}}=3{,}6405\,mm$$

Zusammenfassend wird festgestellt, dass die Koordinatenkomponenten x mit 10mm, y mit 11mm, und z mit 4mm durch Kovarianzfortpflanzung aus der Messung resultieren. Dabei wurden eine Streckenmessgenauigkeit von 15mm und eine Horizontalrichtungsgenauigkeit von 2mgon und eine Vertikalrichtungsgenauigkeit von 3mgon angenommen. Die jeweiligen Varianzen werden zur Erzeugung des simulierten Messwertes benötigt.

4.4 Zeitabhängige Messungen

Alle Objekte, die mit der Strahlung der Sonne wechselwirken, weisen spezifische Eigenbewegungen auf. Bedingt durch die jahreszeitliche und tageszeitliche Erwärmung der Messobjekte entstehen oszillierende Effekte. Durch die über den Tag entstehende Temperaturveränderung dehnt sich das Messobjekt aus beziehungsweise zieht sich zusammen. Dieses physikalische Verhalten entsteht zeitversetzt, da es eine gewisse Zeitdauer benötigt, bis die Erwärmung/Abkühlung eintritt. Durch die jahreszeitliche Temperaturschwankung wird der Temperaturbereich des Tages je nach Jahreszeit verschoben. Andere Bewegungsverhalten resultieren aus weiteren Umweltfaktoren, wie Wind, Regen oder Schnee oder aus Gesellschaftsfaktoren, Straßenverkehr, Rush-Hour oder Sonntagsfahrverbot für Lastkraftwagen. Dadurch muss zur Messung die richtige Temperatur und Uhrzeit mit aufgenommen werden, damit diese bei der Auswertung mit der Messgröße in Korrelation gebracht werden kann. Bei der Auswertung müssen diese Phänomene und ein möglicher Trend berücksichtigt werden.[15]

5 Software zur Erstellung virtueller Sensoren

Der Inhalt dieses Kapitels beschäftigt sich mit der Analyse von vorhandenen Softwareprodukten zur Erstellung virtueller Sensoren/ Messinstrumenten.

Es gibt verschiedene Hersteller beziehungsweise Open-Source-Produkte, die Softwarelösungen zum Erstellen von virtuellen Sensoren anbieten. Diese unterschieden sich sowohl in der Schwierigkeit der Handhabung, der Transparenz der dabei durchgeführten Methoden als auch in deren Funktionalitäten.

Betrachtet werden folgende Ansätze:

- Die Entwicklung mittels LabVIEW von National Instruments oder MyOpenLab, ein Open-Source-Produkt von Carmelo Salafia
- Die Programmierung in Java

5.1 LabVIEW

LabVIEW ist ein grafisches Programmiersystem von National Instruments. Das Akronym steht für „Laboratory Virtual Instrumentation Engineering Workbench“. Es handelt sich um ein leistungsstarkes System, das die Möglichkeit bietet virtuelle Instrumente (VI) zu erstellen und zu konfigurieren. Diese VIs enthalten ein Frontpanel, welches die Benutzerschnittstelle darstellt und das Blockdiagramm, in dem sich der Programmcode befindet. Für nicht umfangreiche Problemstellungen ist die grafische Programmierung recht gut lesbar, aber für umfangreichere Anwendungen ist die Übersicht nicht mehr gegeben. LabVIEW Programme können nur in der Entwicklungsumgebung bearbeitet und erstellt werden. Diese programmierten VIs können in ausführbare LabVIEW Programme compiliert werden und in einer speziellen Laufzeitumgebung ausgeführt werden. Die Lizenzgebühren liegen zwischen 1299€ und 4899€ (Hersteller 22.08.2010).[16]

5.2 MyOpenLab

MyOpenLab ist eine grafische Programmierumgebung auf Basis von Java. Es werden einige Standardoperationen zur Verfügung gestellt, mit denen Programme realisiert

werden können. Die Philosophie dabei ist: „Von Nutzer für Nutzer“. Die Entwicklung von eigenen grafischen Programmierelementen ist unter einem vorgegebenen Rahmen möglich, daher ist es grundsätzlich durchführbar. Bei MyOpenLab handelt es sich um ein Open-Source-Produkt und es fallen daher keine Lizenzgebühren an.

5.3 Java

Die objektorientierte Programmiersprache Java gehört seit Januar 2010 zu Oracle und ist ein Bestandteil der Java Technologie. Die in Bytecode übersetzten Programme werden in einer JAVA-Laufzeitumgebung ausgeführt, der JVM (Java Virtual Machine). Durch diese sind Java-Applikationen plattformunabhängig und funktionieren in der Regel ohne Anpassung auf verschiedenen Computern und Betriebssystemen, für die eine JVM existiert. Vorteil dieses Ansatzes ist die einfachere Entwicklung des virtuellen Sensors, der durch eine Konfigurationsdatei von außen einstellbar ist. [17]

5.4 Fazit

Zusammenfassend wird deutlich, dass sich alle Produkte für die Lösung der Problemstellung eignen.

Bei LabVIEW erfolgt die Programmierung grafisch mit Hilfe der Programmiersprache „G“. Der Programmieraufwand ist geringer als bei textbasierten Programmiersprachen, da vorgegebenen Elemente verbunden werden. LabVIEW ist sehr kostenintensiv, für die Erstellung eines virtuellen Sensors wird LabView Base benötigt und ein Application Builder. Mit diesem können ausführbare Dateien erstellt werden, die in einer kostenfreien Laufzeitumgebung ausführbar sind. Die Kosten dafür belaufen sich auf 1.299€ für die Entwicklungsumgebung und 1.049€ für den Application Builder.[16]

Ein kostenfreies Produkt ist MyOpenLab. Die Entwicklung eines virtuellen Sensors in MyOpenLab ist grundsätzlich möglich. Dazu muss das Programm, um fehlende Elemente erweitert werden. Der vorgegebene Standard zur Entwicklung der grafischen Elemente erfordert einen erhöhten Zeitaufwand bei der Programmierung.

Im Gegensatz dazu ist die Entwicklung eines virtuellen Sensors ebenso in Java möglich. Vorteile dieser Programmiersprache sind unter anderem die einfache Programmierung, die Plattformunabhängigkeit, eine hohe Arbeitsgeschwindigkeit und die Vertei-

lung der Arbeitsabläufe durch Multithreading. Java bietet die Möglichkeit flexibler und präziser zu entwickeln. Aus dem DABAMOS Projekt stehen verschiedene Klassen und Entwicklungen zur Verfügung, die hierbei genutzt werden können.

Software	Programmierung	Kosten	Geeignet
LabView	graphisch „G“	2.348,00€	ja
MyOpenLab	Graphisch, Java Code	keine	ja
Java	Java Code	keine	ja

Abbildung 9: Übersicht der Software zur Erstellung von virtuellen Sensoren

Obwohl der virtuelle Sensor mit allen Ansätzen realisierbar ist, eignet sich Java durch die Flexibilität und Erfahrung am besten zur Lösung.

6 Entwurf des virtuellen Sensors

In diesem Kapitel wird der virtuelle Sensor entworfen. Es wird darauf eingegangen, wie die entstandenen Anforderungen umgesetzt werden können.

6.1 Anforderungen

Aus der Fülle der Anforderungen geht hervor, dass zum Testen von Software und zur Fehlersimulation ein generisches Modell entworfen werden muss. Zusammenfassend müssen für den virtuellen Sensor folgende Anforderungen berücksichtigt werden:

- Kommunikation:
  - RS232
  - voll duplex
  - Einstellungen zum COMPort
- Konfiguration:
  - Sende- und Empfangsbefehle aus ASCII-Zeichenketten mit unterschiedlichen Trennzeichen
  - Abtastrate oder Intervall
- Mathematik:
  - Einstellung der Varianz oder Standardabweichung, ermittelt durch Kovarianz-fortpflanzung im Vornherein
  - Tagesgang, Jahresgang, Trend
- Programmierung:
  - Java
- Geschwindigkeit:
  - Messintervall von 50 Hz

6.2 Ableitung des virtuellen Sensors

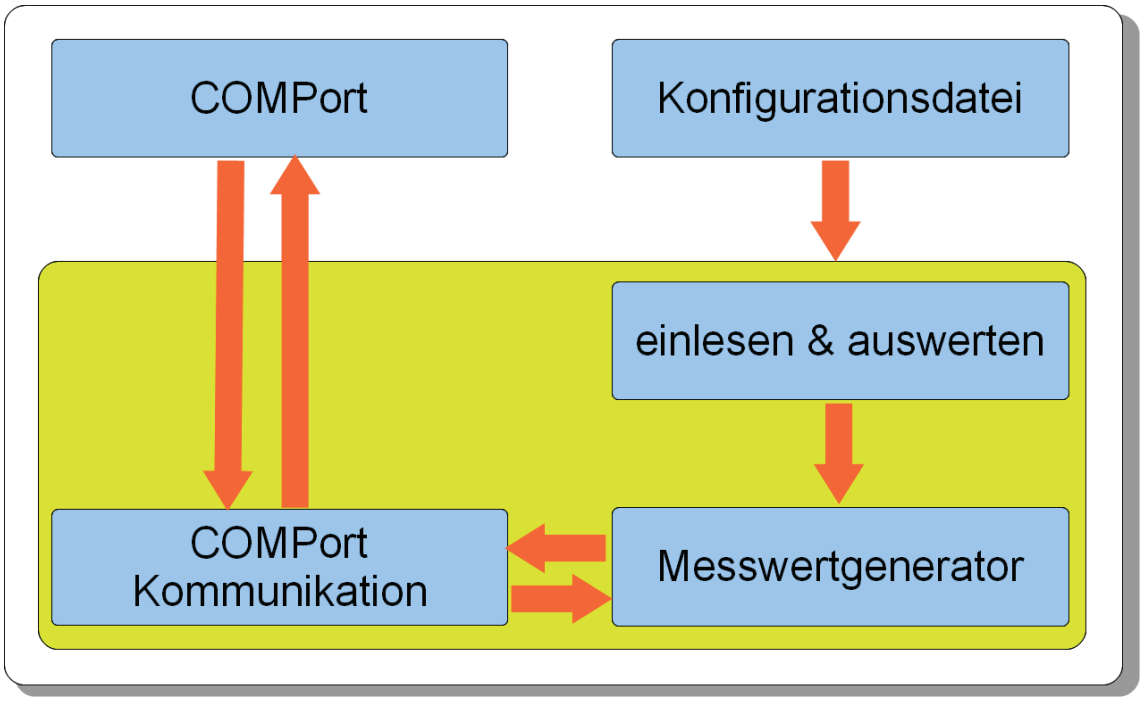

Abbildung 10: Entwurfsschema des virtuellen Sensors

Die Kommunikation zu dem Sensor muss über einen seriellen Port z.B. RS232 realisiert sein. Dieser ermöglicht anschließend die Einstellungen zu der COMPort-ID, der Baudrate, der Parität, den Stopbits und den Datenbits. Die Kommunikation muss voll duplex ablaufen. Der Sensor muss eine Konfiguration im Vorfeld zu lassen.
Es müssen Einstellungen zum Intervall vorgenommen werden können, da das Messsystem mit verschiedenen Abtastraten arbeiten kann. Somit ist eine Synchronisation mit den unterschiedlichen Zeitabläufen gewährleistet, die für die Simulation des Tages-, des Jahresgangs und des Trends notwendig sind.

Die Sende- und Antwortbefehle müssen einstellbar sein. Sie bestehen aus ASCII-Zeichenketten inklusive Steuerzeichen. Die Trennzeichen müssen einstellbar sein, damit der Start beziehungsweise das Ende des Messstrings erkannt werden können. Die simulierten Messwerte müssen zur Laufzeit mit verschiedenen Parametern berechnet werden können. Die Varianz oder die Standardabweichung müssen ebenfalls einstellbar sein wie die Parameter für den Tages-, den Jahresgang und den Trend. Für Debug-Aufgaben von zu testenden Messsystemen wie DABAMOS sollten Einstellungen zum

Verhältnis zwischen reeller und simulierter Zeit vornehmbar sein. Vorteilhaft ist dies, um Jahresgänge mit wenigen Stützpunkten in einem kurzen Zeitraum zu simulieren. Die Geschwindigkeit der Messwerterzeugung sollte 50 Hertz erreichen.

6.3 Arbeitsweise des Messwertgenerators

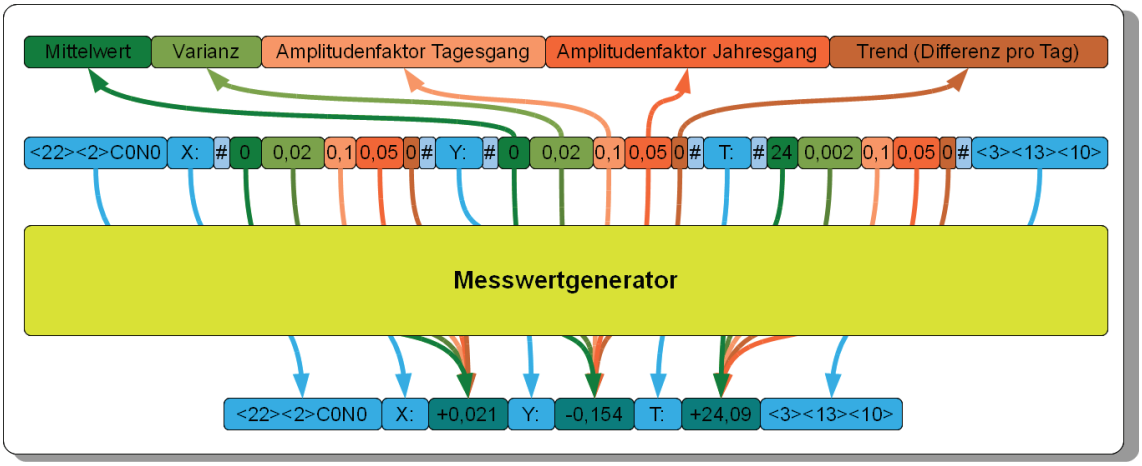

Abbildung 11: Funktionsweise des Generators

Die eingehenden und verglichenen Daten werden an den Generator geschickt. Dieser durchsucht den String nach #-Zeichen, welche einen Block einschließen. In den Blöcken stehen Angaben zur Erstellung des jeweilig simulierten Messwertes. Dieser wird anhand des Mittelwertes und der Varianz zum einzelnen Wert generiert. Der Amplitudenfaktor für den Tagesgang regelt die Größe der Amplitude des Tagesganges, analog dazu der Amplitudenfaktor des Jahresganges. Die Angabe zum Trend verweist auf die jeweilige Differenz pro Jahr.
In Abbildung 11 wird ein Beispielstring eines Leica Nivel 210 erklärt. Nach Auswertung der drei Blöcke werden Messwerte in den angegebenen Genauigkeitsbereichen erzeugt.

6.4 Berechnung des Jahres-, Tagesganges und des Trends

Die Notwendigkeit von Simulationen des Tages- und Jahresganges sowie des Trends wurde in Punkt 4.4 aufgezeigt. Damit die Realisierung statt finden kann, wurden folgende Parameter festgelegt:

$I - Intervall\ ,\ Synchronisationsparameter$
$R - Ratio$
$n - Anzahl\ der\ Messwerte\ pro\ Tag$
$i - Laufvariable$
$x_{pos} - Abstand\ vom\ Koordinatenursprung$
$s - Amplitudenfaktor\ des\ Jahresganges$
$t - Amplitudenfaktor\ des\ Tagesganges$
$\Delta apn - Differenz\ des\ Jahrestrends\ pro\ n$
$MW - erzeugter\ Messwert$
$y - Messwert\ mit\ Tages-,\ Jahresgang\ und\ Trend$

Die Parameter I , R , s , t , Δapn und die Varianz bilden die Schnittstellen zum Anwender. Das Intervall I gibt an, wie hoch die Abtastgeschwindigkeit ausgehend vom Messsystem in Sekunden ist. Es fungiert als Synchronisationsparameter, da von 86.400 Sekunden (Sekunden pro Tag) ausgegangen wird. Wenn im Intervall von 10 Sekunden ein Befehl an den Sensor geschickt wird, braucht dieser nur 86.400 Messwerte pro Tag erzeugen. Soll die Simulation schneller, das heißt ein Tag innerhalb einer Minute bei einer Abtastrate von einer Sekunde ablaufen, wird ein Ration von 1.440 $(24 \cdot 60)$ benötigt. Der Ration R regelt das Verhältnis zwischen Simulationszeit und der Sensorzeit. Aus dem Verhältnis von den Sekunden pro Tag zu dem Intervall und Ratio ergibt sich die Anzahl n der Messwerte pro Tag. Die Laufvariable i erhöht sich pro Abfrage um eins. Mit ihr wird der Abstand zum Koordinatenursprung x_{pos} berechnet.

$$x_{pos} = \frac{2\pi}{i \cdot n}$$

Die Faktoren s und t beschreiben jeweils die Amplitudenhöhen des Jahres- beziehungsweise des Tagesganges. Δapn ist die Differenz des Jahrestrends pro n .

$$n = \frac{24 \cdot 60 \cdot 60}{I \cdot R}$$

MW ist der mit Hilfe des Zufallszahlengenerators simulierte Messwert, welcher sich aus dem vorgegebenen Mittelwert und der vorgegebenen Varianz zusammen setzt.

$$y_{0<i\leq n} = s \cdot \sin(x_{pos}) + t \cdot \sin(365 \cdot x_{pos}) + \Delta apn \cdot i + MW$$

6.5 Benutzerschnittstelle

Die Handhabung soll möglichst einfach gehalten werden, daher wird auf eine grafische Oberfläche verzichtet. Alle Einstellungen zur Berechnung der zu erzeugenden Messwerte werden vorab konfiguriert.

7 Implementierung

In diesem Kapitel werden die Implementierung des virtuellen Sensors und die zugrunde liegenden Algorithmen beschrieben. Hierzu wird auf die Klassenstruktur eingegangen und einzelne Komponenten werden schematisch erläutert.

7.1 Systemvoraussetzung und Entwicklungsumgebung

Die Implementierung des virtuellen Sensors wurde unter Einsatz folgender Werkzeuge realisiert:

- Betriebssystem: Windows XP Professional SP3
- Programmiersprache: Java 1.6.0_21
- Entwicklungsumgebung: Netbeans 6.9 mit Java Development Kit
- Java-Pakete:
  - Mallet 2.0.5: Mathematikbibliothek[18]
  - XStream 1.3.1: Bibliothek zur Serialisierung von Java-Objekten
  - RxTx 1.3-7: Paket zur Kommunikation mit verschiedenen Schnittstellen
- Testumgebung: selbst geschriebenes Tool zur Kommunikation über den RS232 und ein Null-modem emulator (Quelle: http://com0com.sourceforge.net Stand: 15.08.2010)

Die Softwareentwicklung fand auf einem Notebook mit Intel® Core™ Duo Prozessor mit 2GHZ und 2GB Arbeitsspeicher statt. Die Mindestvoraussetzungen für den Einsatz der entwickelten Software sind:

- Prozessor-Geschwindigkeit: 1000MHz
- Arbeitsspeicher: 512MB
- Betriebssystem: Unix, Mac-OS , ab Windows 2000
- Laufzeitumgebung: Java Runtime Environment 1.6

7.2 Schematischer Aufbau des virtuellen Sensors

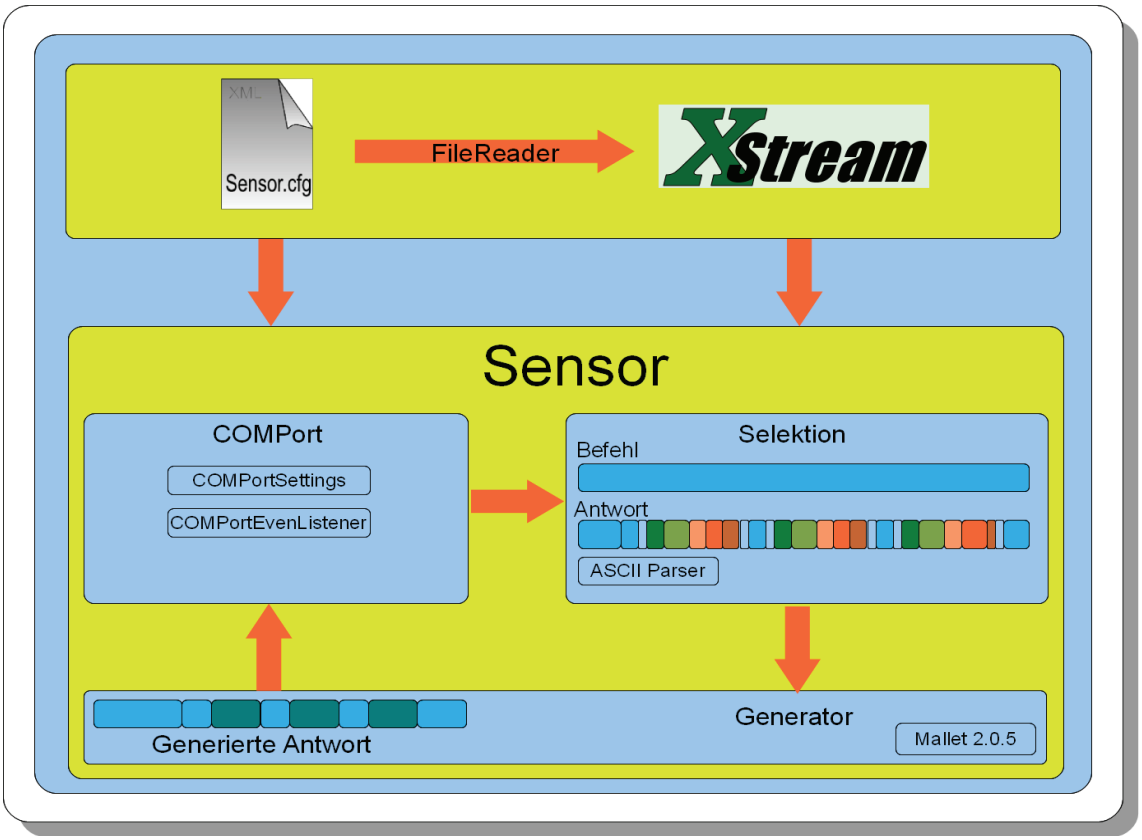

Abbildung 12: Schematischer Aufbau des virtuellen Sensors

Die Abbildung 12 beschreibt den Aufbau des virtuellen Sensors. Die im Vorfeld erstellte Konfigdatei (Sensor.cfg) wird eingelesen und mittels XStream deserialisiert. Das dabei entstehende Sensor-Objekt beinhaltet alle Parameter zum COMPort. Das COMPort-Objekt hat einen EventListener, der auf ankommende Strings (Befehle) reagiert und in der Lage ist, Antworten zu senden. Die ankommenden Befehle werden mit denen aus einer Liste des Sensor-Objektes verglichen und der passende Antwort-String wird mit Angaben zur Berechnung der Zufallswerte selektiert. Der ausgewählte Antwort-String beinhaltet Blöcke, die mit # getrennt sind und zur Berechnung dienen. Dieser String wird an den Generator weiter geleitet. Anschließend erzeugt dieser anhand der Parameter die Zufallszahlen mit Hilfe der Mallet 2.0.5 Bibliothek und fügt diese an den passenden Stellen des Antwort-Strings ein. Der erzeugte Antwort-String wird dann an das COMPortObjekt übermittelt und heraus gesendet.

7.3 Komponentenstruktur

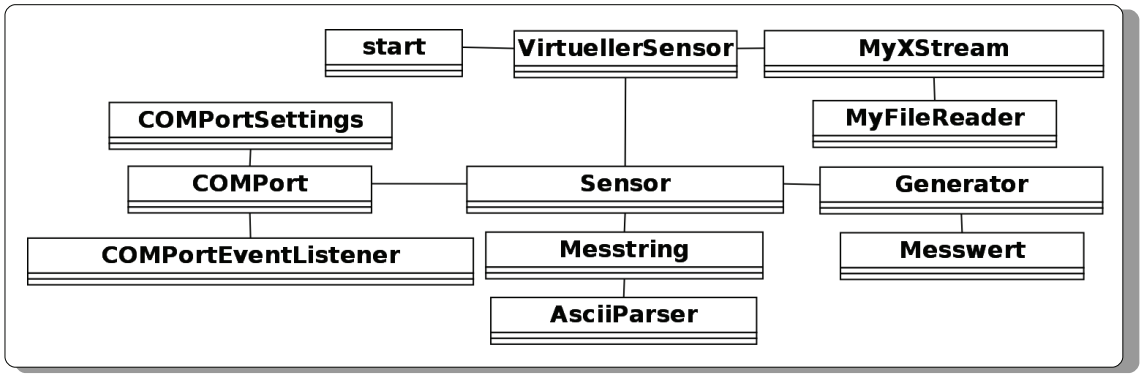

Abbildung 13: Klassendiagramm der Applikation
(Attribute/Operationen ausgeblendet)

Der „virtuelleSensor“ wird in der Klasse „start“ instantiiert und läuft in einem Thread. Der „virtuelleSensor“ instantiiert aus der eingelesenen Konfigurationsdatei ein Objekt der „Sensor“ Klasse. Die Konfigurationsdatei wird mit dem „MyFileReader“ eingelesen und dann mittels MyXStream deserialisiert. Das Sensor-Objekt instantiiert ein „COMPort“ Objekt, welches „COMPortSettings“ nutzt, um die Einstellungen der Schnittstelle zu erhalten. Der „COMPortEventListener“ reagiert auf über die Schnittstelle ankommende Zeichen und wird vom COMPort-Objekt instantiiert. Das deserialisierte Sensor-Objekt enthält instantiierte „Messtring“´s. Die „Messtring“ Klasse nutzt ein instantiiertes „Ascii-Parser“ Objekt. Das Sensor-Objekt instantiiert nach dem Erhalt eines gültigen Befehlsstrings ein Objekt der Klasse „Generator“ und lässt dieses in einem Thread laufen. Parallel dazu wird der dazugehörige Antwortstring im Generator zerlegt und eine „Messwert“ Objekt Liste angelegt.

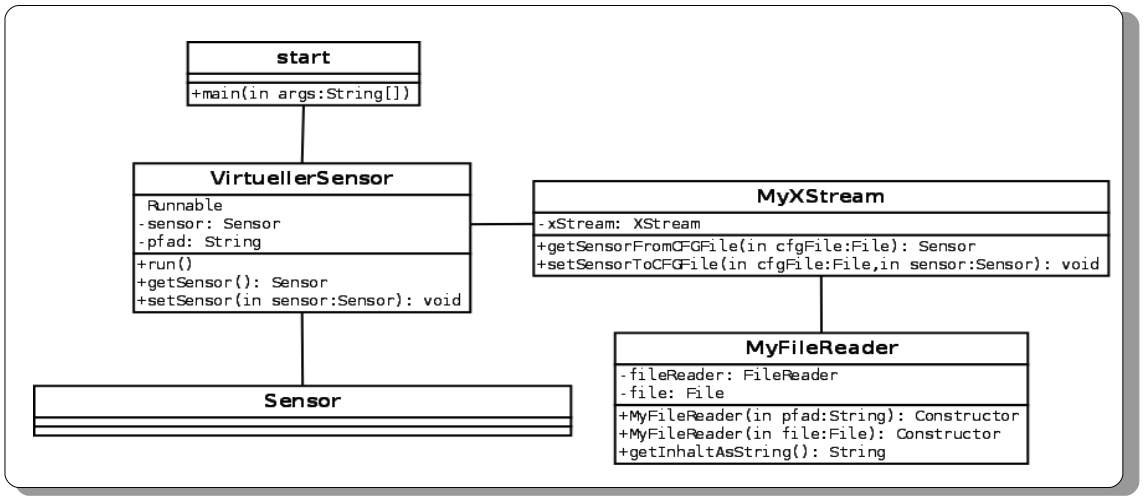

Abbildung 14: Klassendiagramm "Start und Deserialisierung"

Die „start“ Klasse enthält die main Methode, welcher beim Start des Programmes der Konfigurationsdateipfad übergeben wird. Sie instantiiert ein Objekt der „Virtueller-Sensor“ Klasse, welche Runnable implementiert. Dieses Objekt läuft dann in einem Thread, der von der main Methode gestartet wird. Das „Sensor“-Objekt wird über das „MyXStream“-Objekt deserialisiert. Dazu wird das „File“-Objekt der Konfigurationsdatei an „MyXStream“ und dann an „MyFileReader“ übergeben. Das „MyFileReader“-Objekt, welches in „MyXStream“ instantiiert wurde, bietet Funktionen zum kompletten Einlesen der Konfigurationsdatei an.

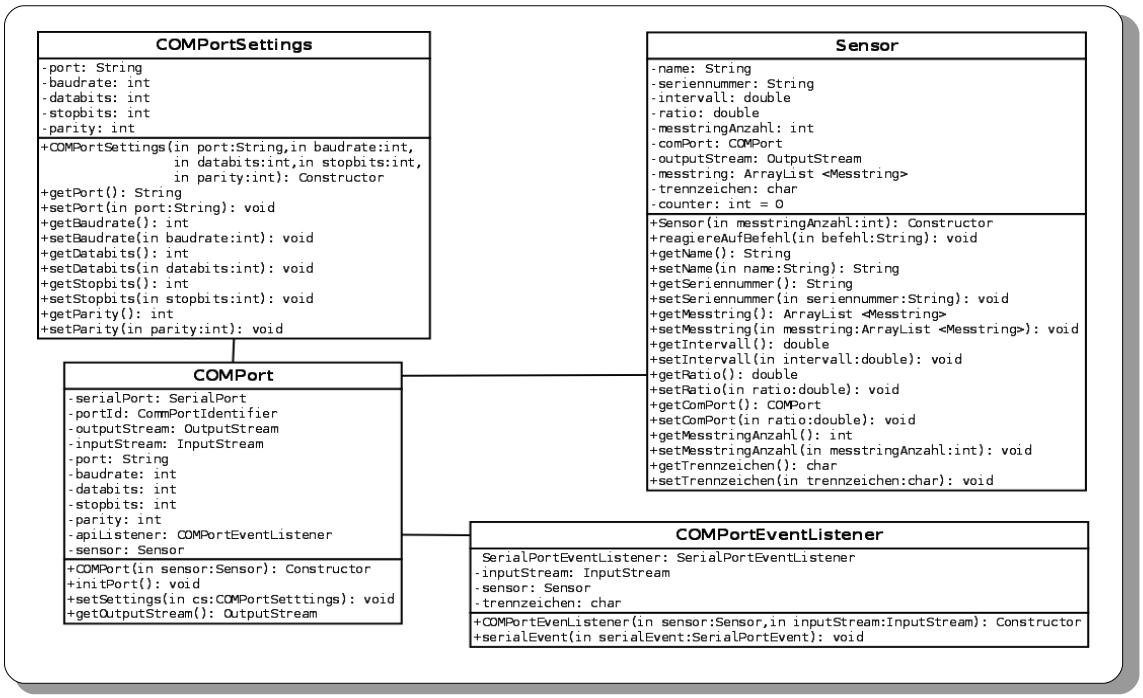

Abbildung 15: Klassendiagramm Sensor und COMPort

Das „Sensor“-Objekt instantiiert ein Objekt der Klasse „COMPort“, welches die Verbindung vom Messsystem zum virtuellen Sensor realisiert. Das Datenobjekt „COMPortSettings“ enthält sämtliche Einstellungen zum „COMPort“. Nach der Instantiierung wird das „COMPort“-Objekt mit Hilfe der „COMPortSettings“ initialisiert. Das „COMPort“-Objekt instantiiert einen „COMPortEventListener“, welches auf ankommende Zeichen des COMPorts reagiert. Es hält eine Instanz des „Sensor“-Objektes und dient als Observer. Hierbei werden die ankommenden Zeichen auf das vereinbarte Trennzeichen verglichen und eine Meldung an das „Sensor“-Objekt geschickt. Diese besteht aus den einzelnen zusammengesetzten ankommenden Zeichen und bildet den Befehlsstring.

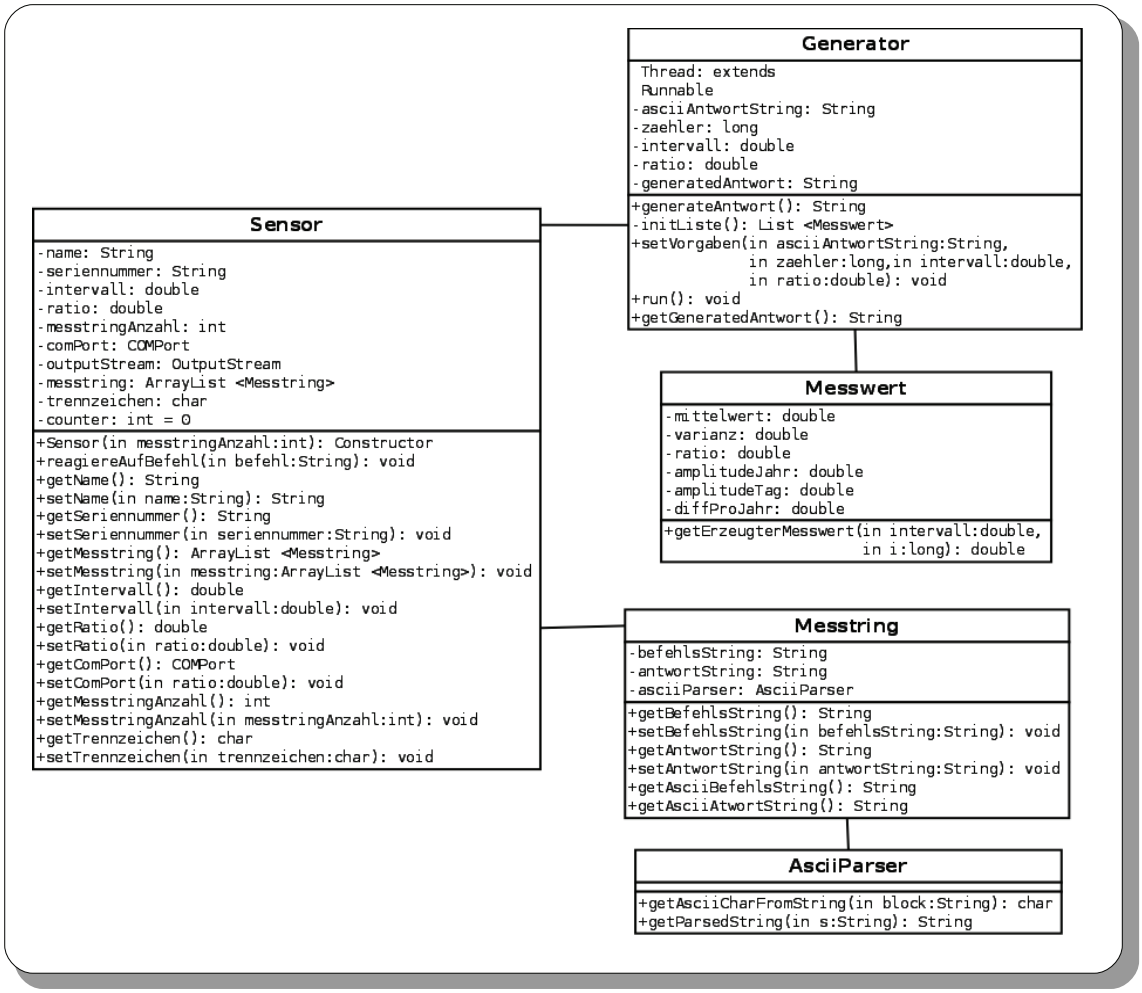

Abbildung 16: Klassendiagramm Generator

Das „Sensor“-Objekt enthält eine Liste von „Messtring“-Objekten, die dem Vergleich der ankommenden Befehlsstrings und zur Auswahl des korrekten Antwortstrings dienen.

Jedes „Messtring“-Objekt enthält ein „AsciiParser“-Objekt, welches für die Umwandlung der Strings aus der Konfigurationsdatei benötigt wird. Die Klasse „AsciiParser“ bietet Methoden an, mit denen Strings, die folgende Strukturen enthalten: „<Dezimal Zahl>“ oder <Hexadezimal Zahl> (Beispiel: <13> oder <0xD> für carriage return) in Strings mit in Java verwendbaren ASCII-Zeichen umgewandelt werden (Beispiel: carriage return in Java: \u0013). Des Weiteren instantiiert das „Sensor“-Objekt ein „Generator“-Objekt, welches als Thread gestartet wird. Das „Generator“-Objekt erbt von Thread und implementiert Runnable. Das „Generator“-Objekt legt nach der Auswahl des Antwortstrings eine „Messwert“-Objekt Liste an. Diese besteht aus Messwerten, die aus den Vorgaben des Antwortstrings der Konfigurationsdatei generiert wurden und wird dann zur Erstellung des Antwortstring benötigt. Dieser wird auf den Outputstream des „COMPort“ geschrieben.

7.4 Aufbau der Konfigurationsdatei

Der virtuelle Sensor wird mit Hilfe einer Konfigurationsdatei eingestellt. Diese ist mit Hilfe von XStream erstellt worden und bildet ein serialisiertes Sensor-Objekt ab, welches in Java geschrieben ist. Die Datei ist in verschiedene Abschnitte gegliedert. Es werden Daten zum Sensor, zum COMPort und zu den Messstrings gespeichert.

Beispiel für die Konfiguration eines virtuellen Nivel 210:

```
<virtuellersensor.Sensor>
  <name>Sensorname</name>
  <intervall>1.0</intervall>
  <ratio>1.0</ratio>
  <messtringAnzahl>1</messtringAnzahl>
  <comPort>
    <port>COM1</port>
    <baudrate>9600</baudrate>
    <databits>8</databits>
    <stopbits>1</stopbits>
    <parity>0</parity>
    <sensor reference="../.."/>
  </comPort>
  <messtring>
    <virtuellersensor.Messtring>
      <befehlsString>&lt;16&gt;&lt;2&gt;N0C0 G
A&lt;3&gt;&lt;13&gt;&lt;10&gt;</befehlsString>
      <antwortsString>&lt;0x22&gt;&lt;0x2&gt;C0N0
X:#0.000;0.002;0.1;0.05;0.001# Y:#0.000;0.002;0.1;0.05;0.001#
T:#24.00;0.002;0.1;0.05;0#&lt;0x3&gt;&lt;0x6&gt;</antwortsString>
      <asciiParser/>
    </virtuellersensor.Messtring>
  </messtring>
  <trennzeichen>&lt;10&gt;</trennzeichen>
  <counter>0</counter>
</virtuellersensor.Sensor>
```

XStream ersetzt „<“ in „<“ und „>“ in „>“, da XStream die xml-Tags mit den gleichen Zeichen benutzt.

7.5 Bedienung

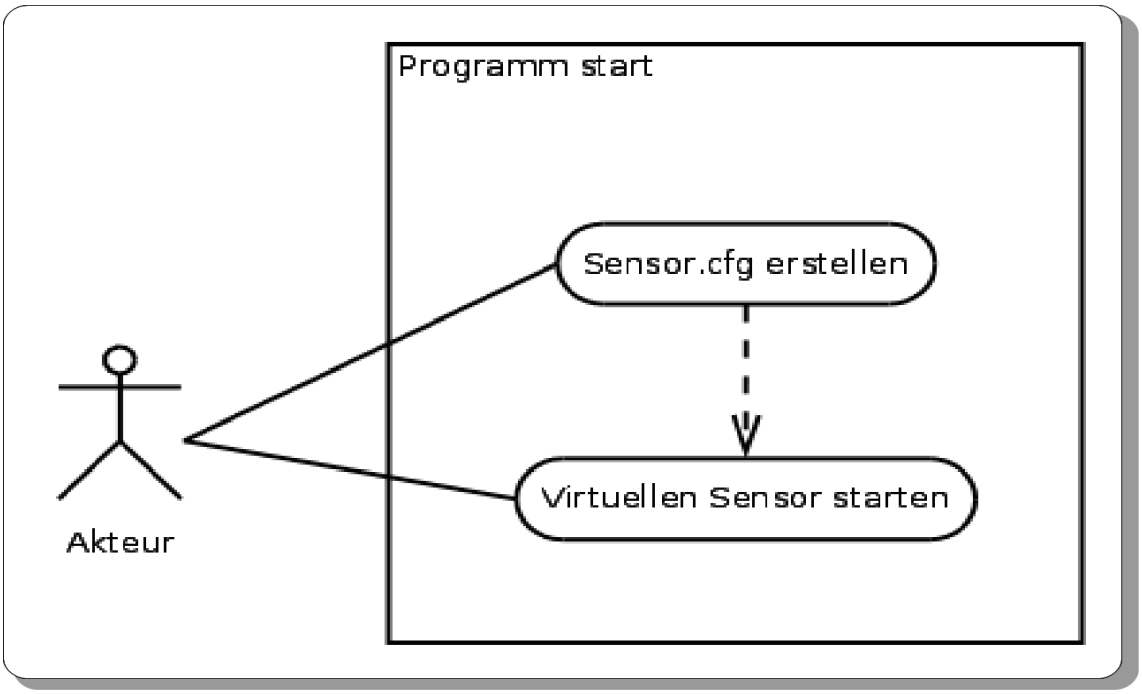

Abbildung 17: Bedienung des virtuellen Sensors

Der Benutzer erstellt eine Konfigurationsdatei, welche beim Programmstart über die Konsole mit dem Befehl: *java -jar virtuellersensor.jar „Pfad der Konfigdatei“* übergeben wird. Es erfolgen Standardausgaben zum Sensor und zur Verbindung.

Nach dem Start der Applikation wartet diese auf Befehle, die über den, in der Konfigurationsdatei angegebenen COMPort, eingehen. Wenn ein eingehender Befehl mit einem Befehl aus den Konfigurationsvorgaben übereinstimmt, sendet das Programm die generierte Antwort zurück auf den COMPort. Der Ablauf ist in Abbildung 18 dargestellt.

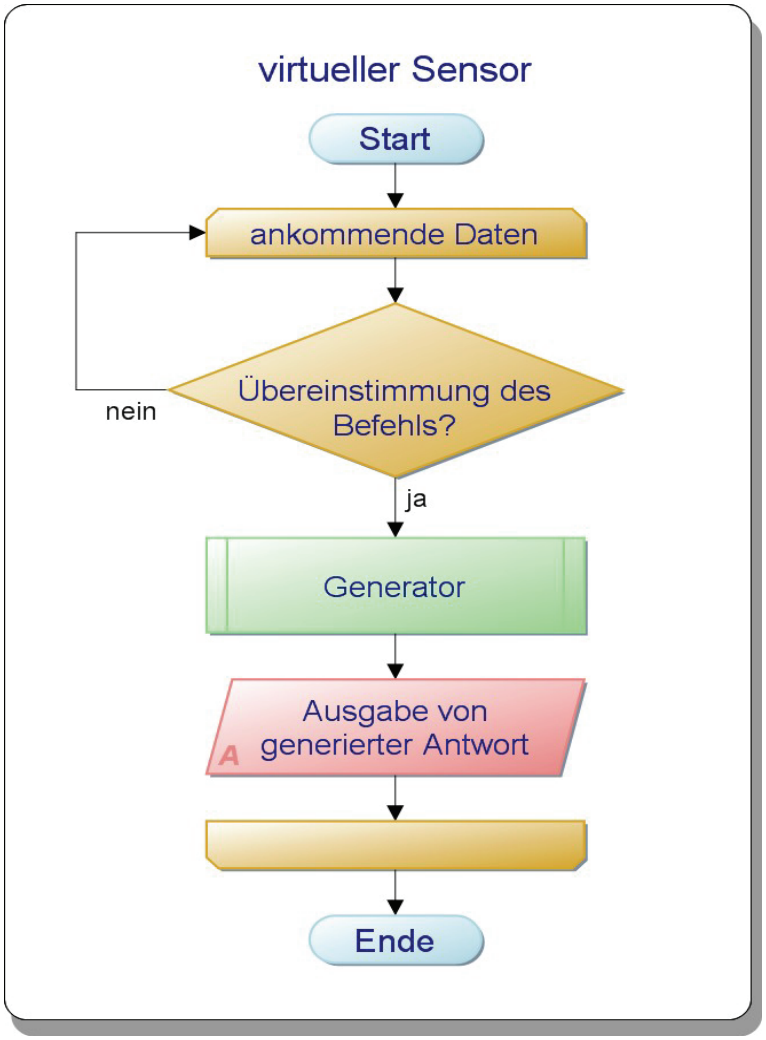

Abbildung 18: Programmablaufplan

8 Testlauf

8.1 Testlauf 1 – Zusammensetzung der Messwerte

Dieser Testlauf soll das Prinzip der Erzeugung von Messwerten mit Tages-, Jahresgang und Trend beschreiben.

Die Abbildung 19 zeigt die Simulation von 1.400 Messwerten mit Tagesgang. Es wurden folgende Parametern zur Erzeugung eingesetzt:

- Mittelwert: 10
- Varianz = 0,005
- Intervall = 1
- Ratio = 3600 (24 Werte pro Tag)
- Tagesgang t = 0,25
- $y_{0<i\leq n} = t \cdot \sin(365 \cdot x_{pos}) + MW$

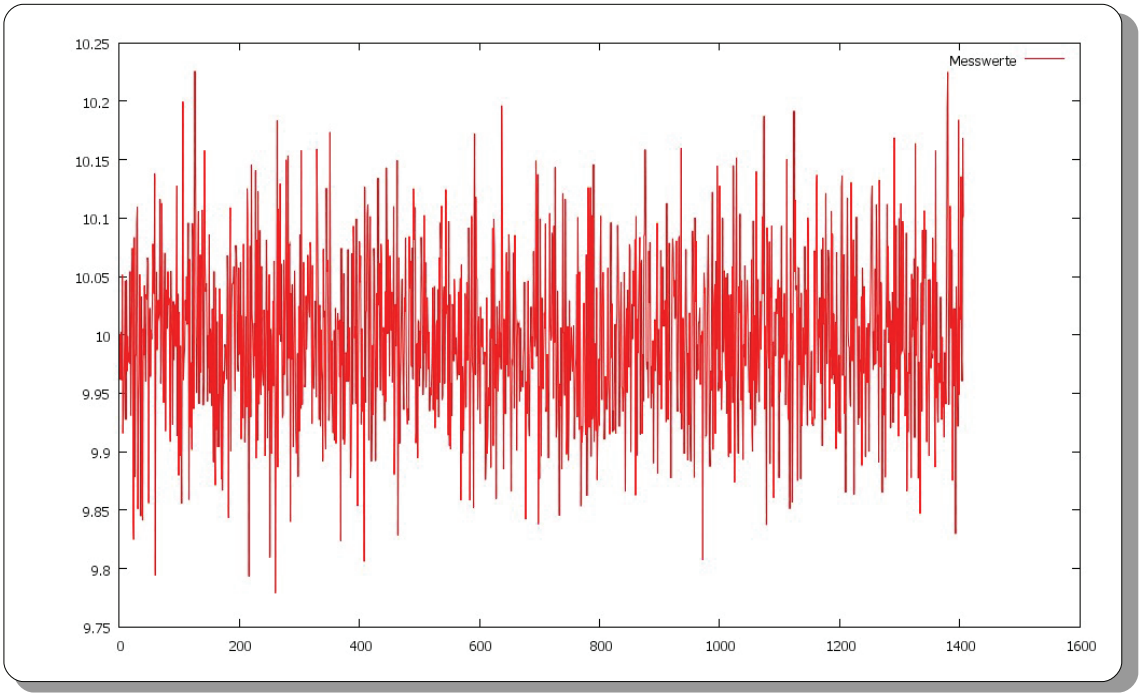

Abbildung 19: Messwerte mit Tagesgang

Die Abbildung 20 zeigt 1400 Messwerte, die zusätzlich mit einem Jahresgang versehen sind. Sie wurden nach folgenden Parametern erzeugt:

- Varianz = 0,005
- Intervall = 1
- Ratio = 3.600
- Tagesgang t = 0,25
- Jahresgang s = 1
- $y_{0<i\leq n} = s \cdot \sin(x_{pos}) + t \cdot \sin(365 \cdot x_{pos}) + MW$

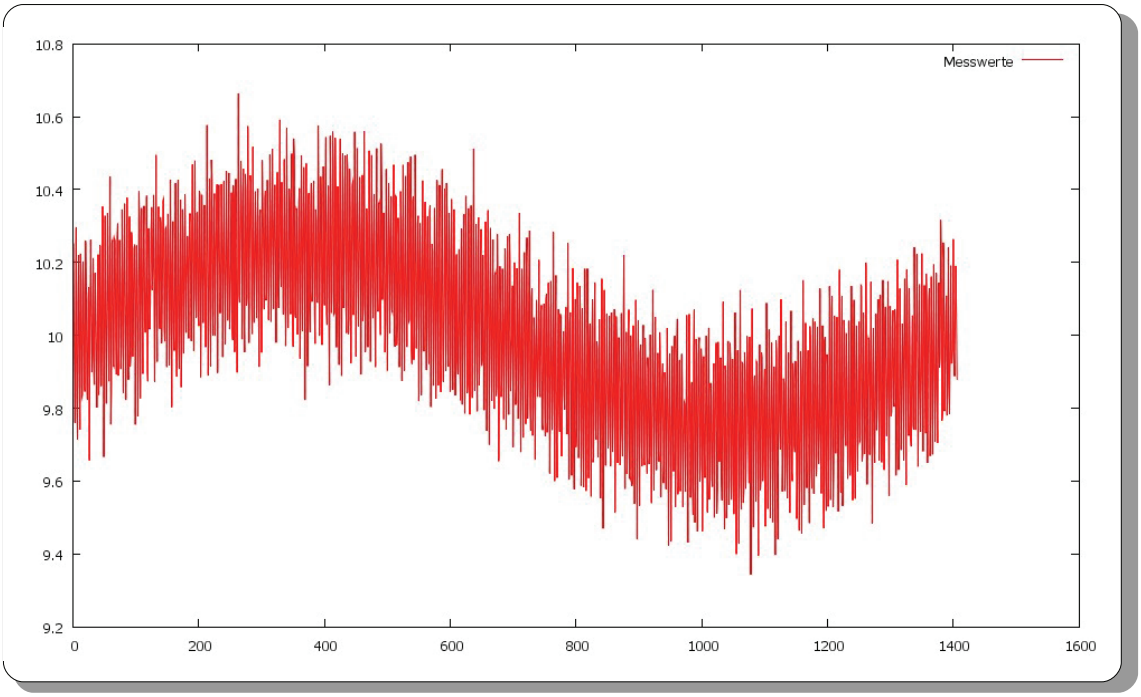

Abbildung 20: Messwert mit Tages- und Jahresgang

Die Abbildung 21 zeigt 1400 Messwerte, die zusätzlich zum Tages- und Jahresgang einen Trend beinhalten. Diese wurden nach folgenden Parametern erzeugt:

- Varianz = 0,005
- Intervall = 1
- Ratio = 3.600
- Tagesgang t = 0,25
- Jahresgang s = 1
- Trend = 1,825
- $y_{0<i\leq n} = s \cdot \sin(x_{pos}) + t \cdot \sin(365 \cdot x_{pos}) + \Delta apn \cdot i + MW$

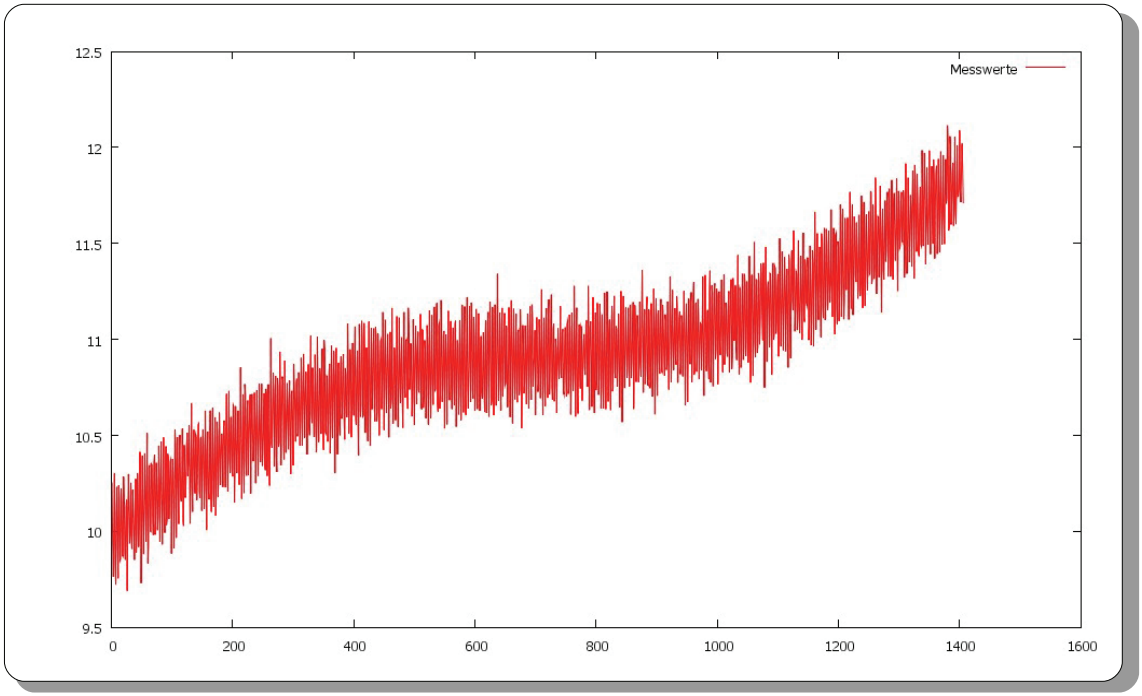

Abbildung 21: Messwert mit Tages-, Jahresgang und Trend

8.2 Testlauf 2 – Simulation von 2 Jahren

Die Abbildung 22 auf Seite 47 zeigt einen Testlauf mit dem DABAMOS Prototypen. Es wurden 17520 Messwerte in 292 Minuten erzeugt (Intervall/Abtastrate von einer Sekunde) und anschließend mit gnuplot 4.4 geplottet. Folgende Parameter wurden für die Simulation an dem virtuellen Sensor eingestellt:

- Mittelwert = 20
- Varianz = 0.005
- Intervall = 1
- Ratio = 3600
- Jahrestrend = 0,3

Bei einem Ratio von 3600 wurde pro Simulationsstunde ein Messwert erzeugt. Daraus ergeben sich 8760 Messwerte pro Jahr.

Inhalt der Konfigdatei zur Simulation:

```
<virtuellersensor.Sensor>
  <name>Sensorname</name>
  <intervall>1.0</intervall>
  <ratio>3600.0</ratio>
  <messtringAnzahl>1</messtringAnzahl>
  <comPort>
    <port>COM1</port>
    <baudrate>9600</baudrate>
    <databits>8</databits>
    <stopbits>1</stopbits>
    <parity>0</parity>
    <sensor reference="../.."/>
  </comPort>
  <messtring>
    <virtuellersensor.Messtring>
      <befehlsString>&lt;16&gt;Test;&lt;13&gt;&lt;10&gt;</befehlsString>
      <antwortsString>&lt;0x22&gt;&lt;0x2&gt;#20.0;0.005;1;2;0.3#&lt;0x6&gt;<
/antwortsString>
      <asciiParser/>
    </virtuellersensor.Messtring>
  </messtring>
  <trennzeichen>&lt;x6&gt;</trennzeichen>
  <counter>0</counter>
</virtuellersensor.Sensor>
```

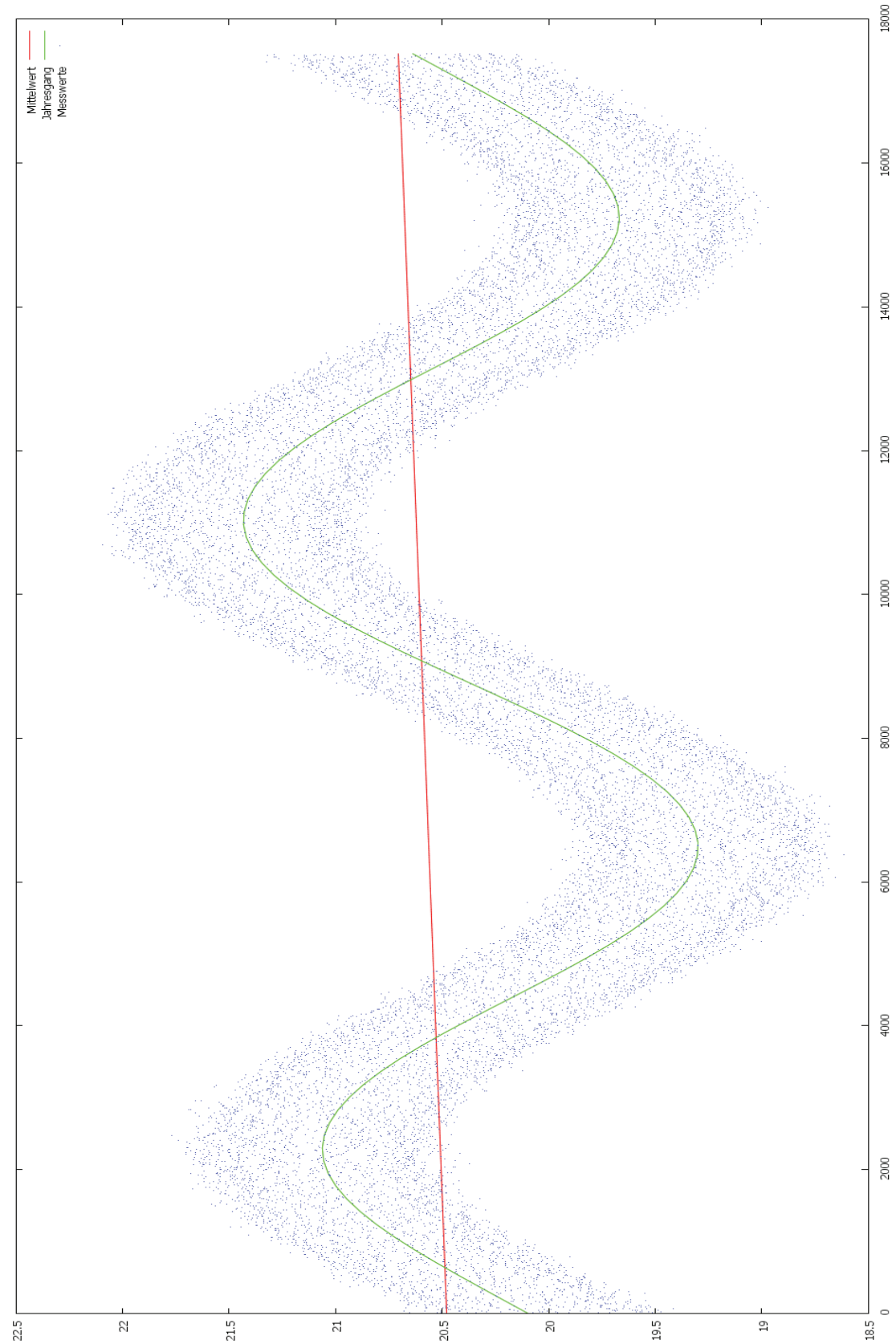

Abbildung 22: Testlauf für 2 Jahre, 17520 Messwerte

9 Schlussbetrachtung

Abschließend wird ein Ausblick auf die Erweiterbarkeit und ein kurzes Fazit gegeben.

9.1 Ausblick

Selbst wenn der Sensor in der derzeitigen Form gut funktioniert, sind weitere Schritte denkbar, die zur Verbesserung des Programmes führen:

- Entwicklung einer grafischen Oberfläche zur Konfiguration und Fehlersimulation während der Laufzeit
- Implementierung einer Kovarianzfortpflanzung zur Berechnung der Varianz aus Messungen mit abgeleiteten Größen
- Verbesserung des mathematischen Modells, sodass die simulierten Messwerte von einander abhängig sind (z.B.: durch die Erwärmung der Sonne dehnen sich Objekte unterschiedlich aus)
- die Simulation anhand vorhandener Messwerte
- den virtueller Sensor um weitere Schnittstellen erweitern (z.B.: Direktverbindung über TCP/IP)

9.2 Fazit

Die Überwachungsmessung, als ein Aufgabengebiet der Ingenieursvermessung hat einen großen Stellenwert. Daher gewinnen automatisierte Überwachungsmesssysteme, wie DABAMOS, immer mehr an Bedeutung.

Das Ziel dieser Arbeit war es, einen generischen, virtuellen Sensor für Vermessungsaufgaben zu entwickeln. Dieser wurde in Java programmiert und lässt eine Generierung von Messwerten aus vorab einstellbaren Parametern zu. Der virtuelle Sensor, der Messgeräte simuliert, dient unter anderem DABAMOS als Testumgebung. Bei der Generierung von Messwerten müssen verschiedene Messeinflüsse beachtet werden. Dafür ist eine Kovarianzfortpflanzungsrechnung zur Bestimmung von Messunsicherheiten von abgeleiteten Messgrößen anzuwenden. Die Kommunikation mit dem virtuellen Sensor erfolgt über RS232. Dieser ist damit in der Lage mit DABAMOS wie ein beliebiges Messgerät, welches mittels ASCII-Zeichen kommuniziert,

zu agieren (siehe Abbildung 23).

Der Sensor wurde in einem simulierten Tages- und Jahreslauf erfolgreich getestet und liefert für DABAMOS verwertbare Messwerte. In Zukunft wird DABAMOS um ein Auswerteprogramm erweitert. Der virtuelle Sensor trägt dabei zur erleichterten Entwicklung bei.

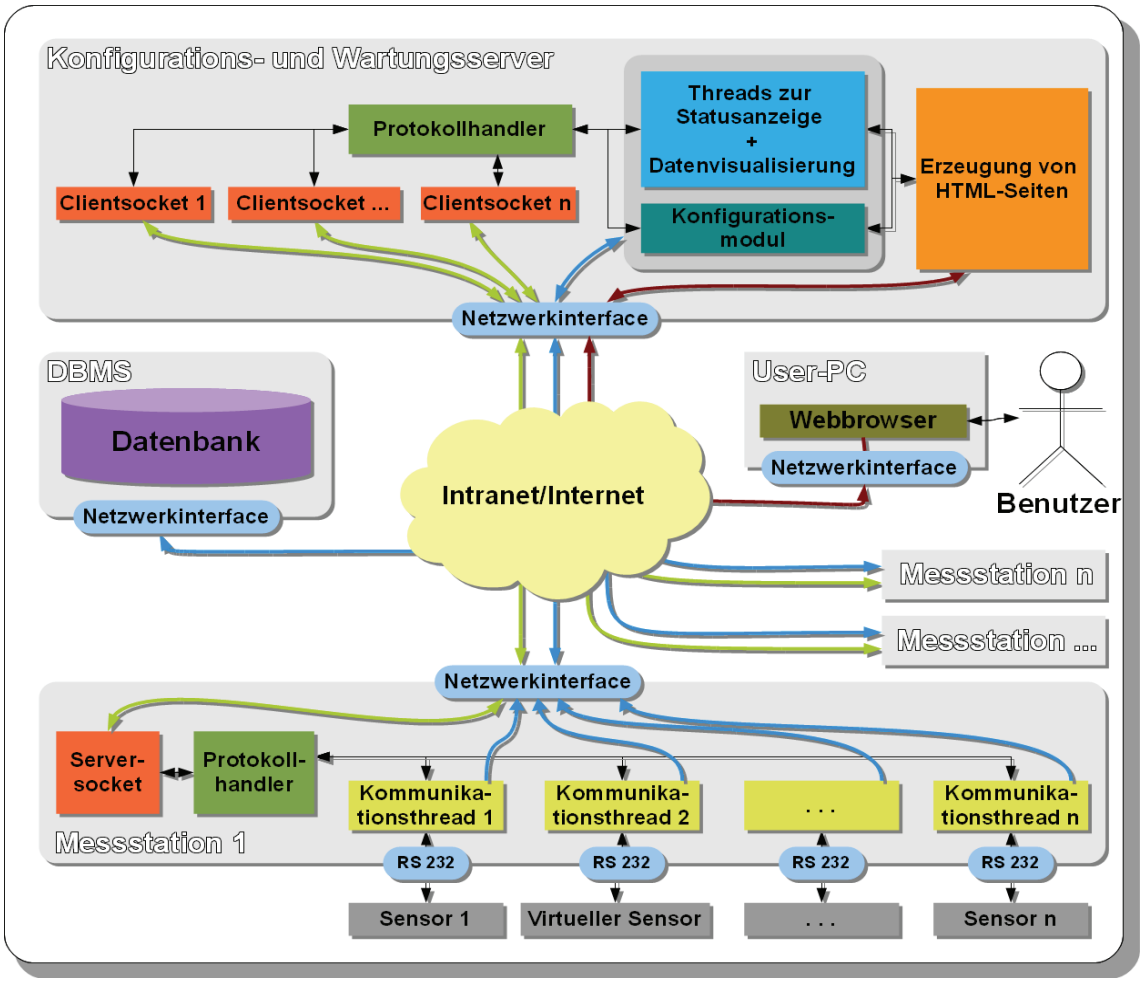

Abbildung 23: Aufbau DABAMOS (Originalabbildung aus Wolff 2010)[4]

Abbildungsverzeichnis

Literaturverzeichnis

1: Welsch/Heunecke/Kuhlmann, Handbuch Ingenieurgeodäsie: Auswertung geodätischer Überwachungsmessungen, 2000

2: Welsch/Heunecke/Kuhlmann, Handbuch Ingenieurgeodäsie: Auswertung geodätischer Überwachungsmessungen, 2000

3: LPI - Ingenieurgesellschaft mbH, Bauwerksmonitoring, 2010, http://www.lpi-ing.de/arbeitsgebiete/bauwerksmonitoring.html

4: Christian Wolff, Entwicklung eines Webinterfaces zur Fernwartung von Überwachungsmesssystemen, 2010

5: Deumlich/Steiger, Instrumentenkunde der Vermessungstechnik, 2002

6: Leica Geosystems, Leica TPS1200 GeoCOM Reference Manual, 2006

7: Schlemmer, Grundlagen der Sensorik - Eine Instrumentenkunde für Vermessungsingenieure, 1996

8: Leica Geosystems: NIVEL200 - Gebrauchsanweisung,

9: Leica Geosystems, GSI ONLINE for Leica TPS and DNA, 2002

10: Thommen, Bedienungsanleitung Meteo Station HM30, 2008

11: Staiger, Rudolf, Motivation und Strategie zur Prüfung geodätischer Instrumente, 2001

12: Foppe, Karl, Statistik und Fehlerlehre I - Kovarianzfortpflanzung, 2010

13: Niemeier, Wolfgang, Ausgleichsrechnung - Statistische Auswertemethoden, 2008

14: Heister, Hansbert, Genauigkeitsmaß in der geodätischen Messtechnik, 2010

15: Wolfgang Niemeier,Rudolf Staiger,Karl Foppe,Hans Neuner, Zeitabhängige Messgrößen - Verborgene Schätze in unseren Daten 85.DVW-Seminar, 2009

16: National Instruments, LabVIEW, 2010, http://www.ni.com/labview/buy/d/

17: Dr. Thomas Kröckertskothen, RRZN, Java 2 - Grundlagen und Einführung, 2006

18: Mallet, About Mallet, 2009, http://mallet.cs.umass.edu/index.php