



Fachbereich Mathematik / Informatik

# **Entwicklung einer Client-Server-basierten Robotersteuerung**

**Diplomarbeit  
im Studiengang Informatik**

von

**Sebastian Hühn**

**Matrikel-Nr: 1653667**

**shuehn@informatik.uni-bremen.de**

Gutachter

**Prof. Dr. Karl-Heinz Rödiger**

**Dr. Dieter Müller**

Bremen, 10. November 2008

## **Danksagung**

Ich möchte meinem Professor Karl-Heinz Rödiger für die stets hilfsbereite Betreuung meiner Diplomarbeit danken. Meinen Eltern danke ich für das mir entgegengebrachte Vertrauen und die jahrelange Unterstützung, durch die diese Diplomarbeit erst möglich geworden ist. Meiner Freundin Julia danke ich für die seelische Unterstützung während meiner Diplomarbeit.

# Inhaltsverzeichnis

<b>1 Einleitung .....</b>	<b>5</b>
1.1 Motivation .....	5
1.2 Problemstellung.....	5
1.3 Übersicht über das Dokument .....	6
<b>2 Analyse bestehender Systeme.....</b>	<b>7</b>
2.1 Kommunikation über Gerätetreiber .....	7
2.2 Microsoft Robotics Studio / Modellsprache.....	8
2.3 Hardwarenahe Plattformarchitekturen .....	10
2.4 Vor- und Nachteile der verschiedenen Systeme im Vergleich .....	11
2.4.1 <i>Funktionalität</i> .....	12
2.4.2 <i>Benutzbarkeit</i> .....	12
2.4.3 <i>Effizienz</i> .....	13
2.4.4 <i>Änderbarkeit</i> .....	13
2.4.5 <i>Übertragbarkeit</i> .....	14
2.5 Fazit.....	14
<b>3 Anforderungsdefinition .....</b>	<b>15</b>
3.1 Der Client .....	18
3.1.1 <i>Client Systemanforderungen</i> .....	21
3.1.2 <i>Client Sicherheitsmanagement</i> .....	22
3.1.3 <i>Client Benutzbarkeitsanforderungen</i> .....	23
3.2 Der Server .....	24
3.2.1 <i>Verwendung neuer Übertragungstechniken</i> .....	27
3.2.2 <i>Server Systemanforderungen</i> .....	30
3.2.3 <i>Server Sicherheitsmanagement</i> .....	31
3.2.4 <i>Server Benutzbarkeitsanforderungen</i> .....	31
3.3 Ausführungsbeispiel.....	32
3.4 Mögliche Anwendungsszenarien .....	33
<b>4 Definition des Testsystems.....</b>	<b>34</b>
4.1 Der Testroboter .....	34
4.1.1 <i>Mikrocontroller</i> .....	34
4.1.2 <i>Mechanik</i> .....	35
4.1.3 <i>Sensorik</i> .....	36
4.2 Der Server Prototyp.....	39
4.3 Der Client Prototyp .....	40
<b>5 Testszenarien und Bewertung der Ergebnisse.....</b>	<b>41</b>
5.1 Der Minensuchroboter.....	41
5.2 Die seismologische Mess-Station.....	47

<b>6 Fazit und Ausblick .....</b>	<b>50</b>
6.1 Vor- und Nachteile des entwickelten Systems .....	50
6.2 Mögliche Verbesserungen.....	52
<b>7 Eidesstattliche Erklärung.....</b>	<b>55</b>
<b>8 Literaturverzeichnis.....</b>	<b>56</b>
8.1 Literatur.....	56
8.2 Online-Quellen .....	57
<b>A Anhang .....</b>	<b>58</b>
A.1 Software Anleitungen.....	58
A.1.1 <i>Client Prototyp</i> .....	58
A.1.2 <i>Server Prototyp</i> .....	60
A.2 Testroboter Konstruktionsbeschreibung .....	61
A.3 Beispiel-Bytecode-Listen .....	63
<b>B CD-ROM.....</b>	<b>65</b>
B.1 Inhalt.....	65

# **1 Einleitung**

Ziel der Diplomarbeit ist, die Portier- und Adaptierbarkeit von Steuerungs-Software für Robotersysteme zu erhöhen und die Schnittstellen-Kompatibilität mit den verschiedenen in einem Robotersystem verwendeten Mikrocontrollern zu verbessern. Hierzu soll ein neues Konzept basierend auf Steuercodes erarbeitet und ein Software-Prototyp entwickelt werden.

## **1.1 Motivation**

Roboter sind in der heutigen Zeit nicht mehr wegzudenken. Sie kommen u.a. überall dort zum Einsatz, wo menschliches Leben gefährdet wäre (z.B. Minen- und Bombenentschärfung, Erkundung unwegsamer oder geologisch instabiler Gebiete, Umweltmessungen) oder aus Kosten- und Effizienzgründen (z.B. planetare Erforschung, Gelände- und Gebäudeüberwachung). Eine fehlerfreie und effiziente Kommunikationsschnittstelle ist dabei unerlässlich. Auf Grund der sich ständig verändernden und neu entstehenden Einsatzgebiete in der Robotik sowie in der Anlagentechnik im Allgemeinen werden entsprechend hohe Anforderungen an die zur Systemsteuerung benötigte Software gestellt.

Robotik ist ein langjähriges Hobby von mir, welches ich in meiner Freizeit gerne verfolge. So habe ich auch mit Begeisterung an verschiedenen Wettbewerben teilgenommen, wie z.B. der *Spacecompetition* (Finalist), welche von verschiedenen Unternehmen (LEGO, ESA, Siemens, Hitachi, Intospace) unter der Schirmherrschaft von Ulf Merbold, Prof. Dr. Thomas Christaller und Prof. Dr. Alois Knoll ins Leben gerufen wurde. Zudem ist auch die Entwicklung, Analyse und Vermarktung verschiedener Programme für embedded Windows CE Geräte eine seit langen Jahren von mir ausgeübte Tätigkeit. So bin ich mit der Materie vertraut und verfüge über geeignete Plattformen, die für die Durchführung der Diplomarbeit benötigt werden.

## **1.2 Problemstellung**

In der Robotertechnik und in der Systemsteuerung im Allgemeinen kommen bei einem typischen Anwendungsfall mehrere technische Komponenten unterschiedlicher

Hersteller (wie Laserscanner, Roboterarm, Servos für Antrieb) in einem einzigen System zum Einsatz; sie müssen von einer zentralen ebenso wie dezentralen Steuerungs-Software angesteuert werden. Die Hardware- und Software-Schnittstellen der einzelnen Komponenten sind je nach Hersteller verschieden, d.h. jede dieser Komponenten verfügt über unterschiedliche Mikrocontroller, Schnittstellen und Treiber für verschiedene Plattformen, mit der die Steuerungs-Software kompatibel sein muss. Daraus resultieren eine Vielzahl verschiedener Soft- und Hardware-technischer Anforderungen an das Gesamtsystem, um die Kommunikation mit den einzelnen Komponenten gewährleisten zu können.

Bei der Entwicklung neuer Robotersysteme bzw. Systemsteuerungen muss daher ein hoher Aufwand bei der Implementierung geeigneter Steuerungs-Software betrieben werden. Ggf. müssen Treiber für die Mikrocontroller angepasst oder neu geschrieben werden. Zudem gestaltet sich eine Portierung der Steuerungs-Software auf andere Systeme schwierig, wenn dort zur Kommunikation mit den technischen Komponenten andere Mikrocontroller und Treiber zum Einsatz kommen, die mit der zu portierenden Steuerungs-Software nicht kompatibel sind.

### **1.3 Übersicht über das Dokument**

Die Diplomarbeit ist in fünf Abschnitte unterteilt. In dem folgenden Kapitel 2 werden zunächst drei existierende Systeme vorgestellt, die zur Steuerung eines Robotersystems Verwendung finden. Diese Systeme werden anhand von Qualitätsmerkmalen bewertet und gegenübergestellt. In Kapitel 3 wird dann ein neues Konzept zur Steuerung eines Robotersystems vorgestellt und erklärt. Zur Umsetzung des Konzepts werden die Anforderungen an die Hard- und Software erarbeitet und definiert. Um diese Umsetzung testen zu können, wird im Anschluss in Kapitel 4 ein Testsystem vorgestellt, welches aus einem entwickelten Software-Prototypen und einem Testroboter besteht. Kapitel 5 behandelt die Durchführungen und Auswertung von zwei Testszenarien zur Verifikation des Software-Prototypen unter Zuhilfenahme des Testroboters. Zuletzt wird in Kapitel 6 das entwickelte System anhand von Qualitätsmerkmalen bewertet und den in Kapitel 2 vorgestellten Systemen gegenübergestellt. Zudem wird auf Vor- und Nachteile hingewiesen. Mögliche Verbesserungen für das entwickelte System werden als Ausblick aufgezeigt und erläutert.

## 2 Analyse bestehender Systeme

Im Folgenden werden verschiedene Techniken zur Software-basierten Ansteuerung von Robotersystemen und zu deren Kommunikation vorgestellt und im Hinblick auf ihre Portier- und Adaptierbarkeit bewertet. Autonome bzw. automatisierte Robotersysteme und -anlagen sind nicht Gegenstand dieser Diplomarbeit und der damit verbundenen Problemstellung. D.h., es werden nicht Systeme und deren Portierbarkeit betrachtet, die ausschließlich anhand von vorgegebenen Programmabläufen gesteuert werden, wie z.B. bei der Computerized Numerical Control (CNC), sondern die, die durch menschliche Direkteingabe kontrolliert und überwacht werden. Dabei wird allerdings nicht ausgeschlossen, dass parallel zu der Direkteingabe auch ein Programm zur automatischen Steuerung auf einem Robotersystem existiert, jedoch soll hier nur auf die Softwarelösungen zur Direktsteuerung und zur Kommunikation eingegangen werden.

### 2.1 Kommunikation über Gerätetreiber

In der Robotik ist die Anzahl an Steuerungs-Software so zahlreich wie die möglichen Einsatzgebiete der Robotersysteme selbst. An dieser Stelle wird daher nicht konkret auf eines der vielen Produkte näher eingegangen. Es werden hingegen die verschiedenen Kommunikationstechniken im Allgemeinen beschrieben, welche die Mehrheit der Softwareprodukte als technische Umsetzung zur Kommunikation mit dem jeweiligen Robotersystem verwenden. Diese Steuerungs-Software verwendet zur Ansteuerung der in einem Robotersystem verwendeten Mikrocontroller Gerätetreiber. Diese Gerätetreiber sind abhängig von Betriebssystem-Plattform und Architektur, d.h. sie können nicht auf einer beliebigen Plattform eingesetzt werden und nur mit einem bestimmten Mikrocontroller bzw. einer Mikrocontrollerfamilie kommunizieren.

Daher ist durch die Verwendung von Gerätetreibern zur Kommunikation mit einem Robotersystem die Möglichkeit einer Portierung der Steuerungs-Software auf andere Robotersysteme nur gering, sollten diese über eine andere Betriebssystem-Plattform verfügen und/oder andere Mikrocontroller bzw. eine andere Architektur verwenden. Hier ist es von Nöten, die Gerätetreiber anzupassen bzw. auszutauschen. Insofern

keine Gerätetreiber für das Zielsystem vorhanden sind, müssen diese implementiert werden, wodurch ein zusätzlicher Aufwand bei einer Portierung entstehen würde.

Zudem muss bei einer Portierung der Steuerungs-Software auf ein anderes Robotersystem darauf geachtet werden, dass sich der Aufgabenbereich ändern kann. Dazu zählen z.B. zusätzliche oder veränderte Aktoren und Sensoren und damit verbundene Datenwerte, die von der Steuerungs-Software ausgelesen und repräsentiert werden müssen. Dies bedeutet für die Steuerungs-Software, dass sie bei einer Portierung über eine hohe Adaptierbarkeit verfügen muss, um an den veränderten Aufgabenbereich angepasst werden zu können. Sollte dies nicht der Fall sein bzw. die Steuerungs-Software vom Aufgabenbereich des neuen Zielsystems zu sehr abweichen, muss ggf. der Programmcode der Steuerungs-Software an das Zielsystem durch Änderungen der Implementierung angeglichen werden, um eine Portierung zu ermöglichen. Daraus erfolgt ein zusätzlicher Aufwand bei einem Portierungsversuch.

## **2.2 Microsoft Robotics Studio / Modellsprache**

Einen anderen Ansatz bietet Microsoft mit dem sogenannten Microsoft Robotics Studio [Microsoft 2008a]. Hierbei handelt es sich um eine nicht-kommerzielle Entwicklungs- und Simulationsumgebung speziell für Robotersysteme. Zwar finden weiterhin Gerätetreiber zur Kommunikation mit den jeweiligen Mikrocontrollern eines Robotersystems Verwendung, jedoch bietet dieses Entwicklungsstudio als Besonderheit, neben einer 3D-Simulationsumgebung und den gängigen Programmiersprachen wie C#, Visual Basic .NET und Jscript, zusätzlich eine von Microsoft entwickelte Modellsprache, die sog. VPL (Visual Programming Language [Microsoft 2008b]; siehe Abb. 1). Die mit dieser grafischen Programmiersprache erstellte Steuerungs-Software für ein Robotersystem lässt sich bei einem Portierungsversuch auf ein anderes Robotersystem nun dahingehend an das neue System adaptieren, dass das entwickelte Modell in der VPL Programmumgebung angepasst wird. Aufgrund der Einfachheit einer solchen Modellsprache im direkten Vergleich zu den codebasierten Programmiersprachen ist ein geringerer Mehraufwand bei einer Portierung auf ein neues Robotersystem gegeben.

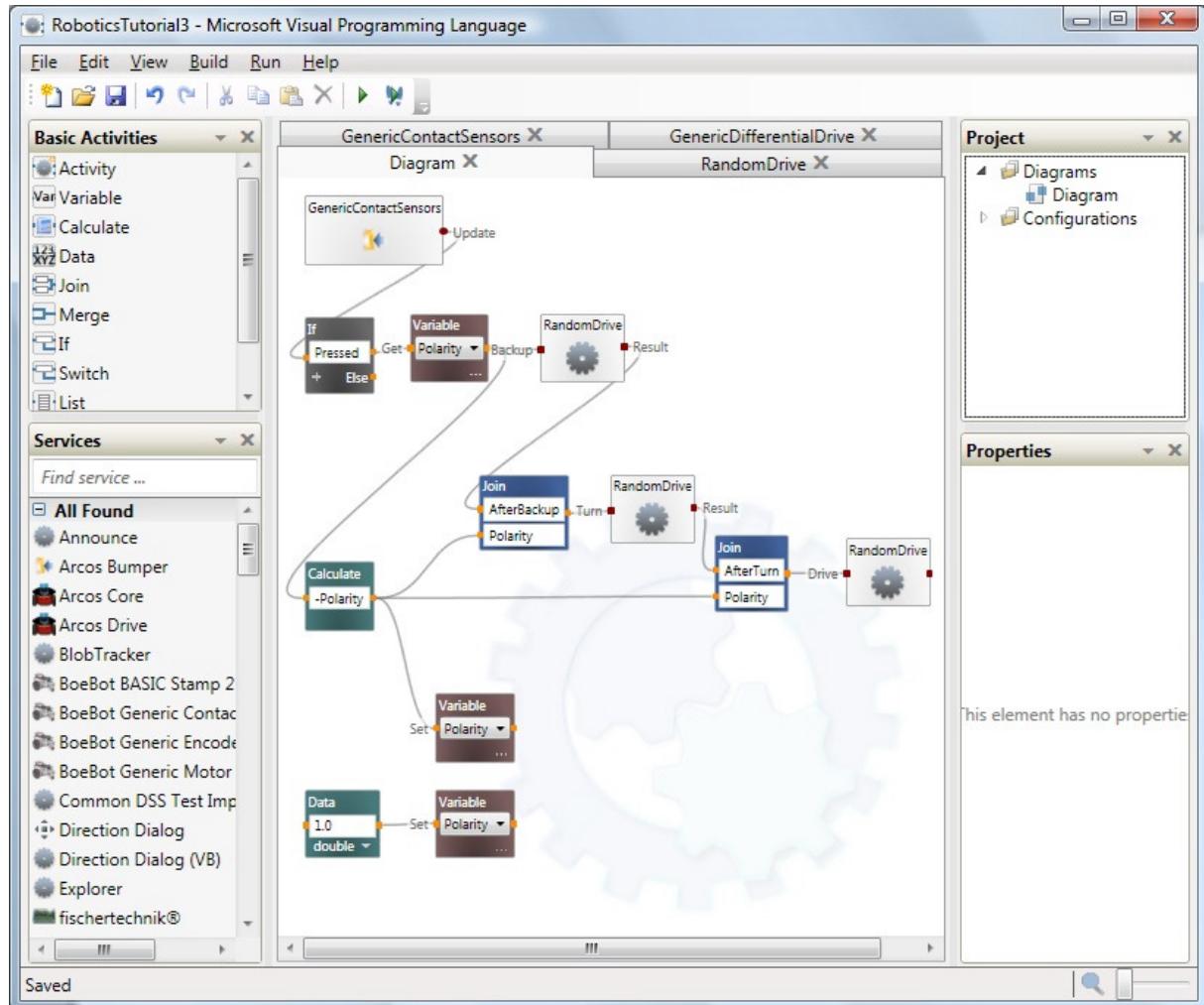


Abbildung 1: Die VPL-Entwicklungsumgebung von Microsoft.

Da diese Umgebung aber ebenfalls die Einbindung von Gerätetreibern zur Kommunikation mit dem jeweiligen Robotersystem erfordert, bleiben die Probleme der bereits unter 3.1 beschriebenen Treiberabhängigkeit weiterhin bestehen. Hinzu kommt, dass das Microsoft Robotics Studio ausschließlich für Betriebssystem-Plattformen der Windows-Familie ausgelegt ist, wodurch eine Portierung der aus dem Entwicklungsstudio resultierenden Steuerungs-Software auf andere Plattformen bereits begrenzt wird.

Eine weitere modellsprachenbasierte Entwicklungsumgebung, welche in der Robotik ebenfalls Einsatz findet, ist LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench [LabVIEW 2008]) der Firma National Instruments, welche jedoch nicht wie das Microsoft Robotic Studio ausschließlich auf diesen Einsatzzweck spezialisiert wurde.

## **2.3 Hardwarenahe Plattformarchitekturen**

Eine weitere verbreitete Vorgehensweise in der Robotersteuerung ist die Verwendung von hardwarenahen, d.h. im Robotersystem integrierten Plattformen, wie z.B. eine Firmware oder rudimentäre Betriebssysteme, die direkt auf die Mikrocontroller aufgesetzt werden und dadurch eine weitere Kommunikationsschicht zwischen der eigentlichen (hardwarefernen) Steuerungs-Software und den Mikrocontrollern selbst bieten.

Diese Firmware bzw. die hardwarenahe Plattform stellt in der Regel einen Dienst zur Kommunikation mit der hardwarefernen Steuerungs-Software zur Verfügung. Bei einer Änderung der Mikrocontrollerarchitektur des Robotersystems muss daher ggf. nur die hardwarenahe Plattform an die neuen Bedingungen angepasst werden. Die hardwareferne Steuerungs-Software kann hingegen von diesen Änderungen unberührt bleiben, da diese Steuerungs-Software von der im Robotersystem verwendeten Mikrocontrollerarchitektur keine Kenntnis haben muss auf Grund der Zwischenschicht durch die hardwarenahe Plattform, welche die eigentliche Kommunikationschnittstelle zu den Mikrocontrollern darstellt. Dies gilt allerdings nur, insofern sich die hardwarenahe Plattform bei einer Portierung auf ein anderes Robotersystem nicht ändert, z.B. bei Verwendung eines anderen Betriebssystems. Zudem erfordern Veränderungen des Einsatzgebietes vom Robotersystem wie z.B. der sensortechnischen Datenerfassung auch bei dieser Methode eine Adaptierung der hardwarefernen Steuerungs-Software. Außerdem kann auch hier wieder die in 3.1 beschriebene Treiberabhängigkeit auf der Ebene der hardwarenahen Plattform zum Tragen kommen; damit verlagert sich ggf. nur der Aufwand bei einer Portierung.

Ein Beispiel für ein solches System sind die LANpointCE Terminals der Firma Intelligent Instrumentation<sup>1</sup>, welche in der Anlagensteuerung wie z.B. bei Großdruckanlagen häufig Verwendung finden. Diese Terminals stellen ein abgeschlossenes System bestehend aus Mikrocontroller zur I/O Steuerung und integriertem Windows CE Betriebssystem dar.

---

<sup>1</sup> LANpointCE Terminals: <http://www.lanpoint.com/>

## **2.4 Vor- und Nachteile der verschiedenen Systeme im Vergleich**

Im Folgenden sollen die drei Systeme unter Verwendung des ISO/IEC 9126 Standards [Rupp 2007] aus Sicht eines Benutzers gegenübergestellt und bewertet werden. Hierzu werden die Qualitätsmerkmale Funktionalität, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit für die einzelnen Systeme betrachtet.

Bei der Funktionalität wird hierbei bewertet, inwieweit geforderte Anforderungen zur Kommunikation mit einem Robotersystem von den Systemen erfüllt werden. Benutzbarkeit bewertet den Aufwand, der betrieben werden muss, um das jeweilige System zur Kommunikation mit einem Robotersystem bzw. zu dessen Steuerung verwenden zu können. Effizienz misst das Verhältnis zwischen dem Leistungsumfang und dem benötigten Aufwand, diese Systeme einzusetzen. Änderbarkeit misst in diesem Fall den Aufwand, um die Software an Änderungen des Robotersystems anzupassen. Übertragbarkeit bewertet die drei Systeme dahingehend, in wie weit diese sich auf andere Plattformen, d.h. in diesem Fall auf andere Robotersysteme übertragen lassen.

Die Tabelle 1 zeigt die Bewertungsergebnisse der drei Systeme in einer Gegenüberstellung. Die Bewertungen der Qualitätsmerkmale werden im Folgenden näher beschrieben.

	Kommunikation über Gerätetreiber	Microsoft Robotics Studio / Modellsprache	Hardwarenahe Plattformen
Funktionalität	++	+	++
Benutzbarkeit	n/a	++	n/a
Effizienz	++	0	+
Änderbarkeit	-	+	-
Übertragbarkeit	-	0	-

**Tabelle 1: Bewertung und Gegenüberstellung der vorgestellten Systeme**

- ++ sehr gut**
- + gut**
- 0 mittelmäßig**
- schlecht**
- sehr schlecht**

### **2.4.1 Funktionalität**

Gerätetreiber bieten in der Regel ein hohes Maß an Funktionalität. Wenn ein Gerätetreiber korrekt implementiert wurde, kann dieser den kompletten Funktionsumfang eines Mikrocontrollers unterstützen, für den der Gerätetreiber entwickelt wurde und diesen Funktionsumfang einer Steuerungs-Software angemessen bereitstellen. So mit können theoretisch alle Anforderungen erfüllt werden, vorausgesetzt die verwendete Steuerungs-Software nutzt den gebotenen Funktionsumfang.

Ähnlich verhält es sich beim Microsoft Robotics Studio, da dieses Treiber-basiert ist, d.h. auf Gerätetreiber zur Kommunikation mit einem Mikrocontroller zurückgreifen muss. Jedoch kann unter Umständen nicht der gesamte Funktionsumfang eines Gerätetreibers in der vom Microsoft Robotics Studio gebotenen Modellsprache unterstützt werden, da nicht für jede theoretisch mögliche Funktion eine Repräsentationsmöglichkeit in der Modellsprache besteht und erst integriert werden müsste. Daher können ggf. nicht alle Anforderungen erfüllt werden, obwohl der Gerätetreiber selbst diese unterstützen würde.

Hardwarenahe Plattformen haben die gleiche Funktionalität wie Gerätetreiber, da sie speziell für die eingesetzten Robotersysteme bzw. die verwendeten Mikrocontroller entworfen und implementiert werden, so dass der komplette Funktionsumfang eines Mikrocontrollers unterstützt wird und somit alle Anforderungen erfüllt werden können.

### **2.4.2 Benutzbarkeit**

Über Benutzbarkeit kann bei treiberbasierten Systemen und bei hardwarenahen Plattformen keine allgemein gültige Aussage getroffen werden, da das Qualitätsmerkmal hierbei je nach eingesetzter Steuerungs-Software, die eines dieser Systeme zur Kommunikation mit dem Robotersystem bzw. Mikrocontroller verwendet, stark variieren kann.

Die Benutzbarkeit des Microsoft Robotics Studio ist dank der gut strukturierten Oberfläche und der Einfachheit der Modellsprache sehr gut. Die Anwendung ist einfach zu bedienen. Die Modellsprache ist für den Benutzer verständlich aufgebaut und mit geringem Aufwand zu erlernen, wobei der Grad des Aufwandes mit der Komplexität der Anforderungen steigen kann.

### **2.4.3 Effizienz**

Gerätetreiber bieten eine sehr gute Effizienz. In der Regel sind sie genau auf den Funktionsumfang eines spezifischen Mikrocontrollers zugeschnitten und dafür entwickelt worden, sodass Antwort- und Verarbeitungszeiten sowie die CPU-Belastung sehr gering sind.

Beim Microsoft Robotics Studio können allerdings die Anforderungen der reinen grafischen Repräsentation und die Echtzeitbearbeitung der Modellsprache während der Laufzeit die Effizienz mindern, da sich die Antwort- und Verarbeitungszeiten sowie die CPU-Belastung sehr erhöhen können.

Hardwarenahe Plattformen sind ähnlich den Gerätetreibern in der Regel speziell für ein System konzipiert und entwickelt worden, wodurch eine gute Effizienz erzielt wird. Jedoch besitzen solche hardwarenahen Systeme häufig nur geringe Speicher- und CPU-Kapazitäten, die die Effizienz mindern können.

### **2.4.4 Änderbarkeit**

Der Benutzer kann Änderungen an Gerätetreibern nur vornehmen, wenn ihm der Quellcode des Gerätetreibers und eine entsprechende Entwicklungsumgebung zur Verfügung stehen. Zudem ist großes Fachwissen über die verwendete Programmiersprache erforderlich, um die gewünschten Änderungen am Gerätetreiber innerhalb des Quellcodes vornehmen zu können.

Das Microsoft Robotics Studio bietet dank der Modellsprache eine sehr gute Änderbarkeit. Der Benutzer kann Änderungen dank der grafischen Repräsentation mit geringem Aufwand innerhalb der Modellsprache vornehmen, vorausgesetzt es sind die erforderlichen Elemente vorhanden. So muss z.B. beim Einsatz eines neuen Sensors ggf. zunächst ein neues grafisches Element entwickelt und der Modellsprache hinzugefügt werden, bevor es genutzt werden kann.

Bei hardwarenahen Plattformen verhält es sich ähnlich den treiberbasierten Systemen, da in der Regel hierfür ebenfalls die Änderungen vom Benutzer direkt im Quellcode vorgenommen werden müssen und der Aufwand sich ähnlich gestaltet.

## **2.4.5 Übertragbarkeit**

Gerätetreiber sind in der Regel betriebssystemabhängig und nur mit einem bestimmten Mikrocontroller bzw. einer Mikrocontrollerfamilie kompatibel. Ändern sich bei einer Übertragung der Steuerungs-Software auf ein anderes System die Betriebssystem-Plattform und/oder die anzusteuernden Mikrocontroller, ist der Gerätetreiber ggf. nicht mehr einsatzfähig und muss ersetzt oder angepasst werden.

Die Übertragbarkeit der Modellsprache des Microsoft Robotics Studio ist dadurch beschränkt, dass die Programme nur auf Systemen mit einer Windows-Betriebssystem-Plattform angewandt werden können. Allerdings ist innerhalb dieser Plattformen eine Übertragung unproblematisch. Ggf. muss der Quellcode für die neue Plattform innerhalb der Entwicklungsumgebung neu erstellt werden.

Da hardwarenahe Plattformen ebenso wie die Gerätetreiber in der Regel betriebssystemabhängig sind und für ein bestimmtes System konzipiert und entwickelt wurden, ist eine Übertragung auf ein anderes, sich durch Betriebssystem-Plattform und/oder Mikrocontroller-Architektur unterscheidendes System nur mit erheblichem Mehraufwand möglich, da Änderungen im Quellcode nötig sind.

## **2.5 Fazit**

Treiberbasierte Systeme und hardwarenahe Plattformen sind sehr effizient und genau auf die jeweiligen Systeme und Anforderungen zugeschnitten, allerdings auf Grund ihrer schlechten Änder- und Übertragbarkeit für Portierungen auf andere Systeme bzw. andere Aufgabenbereiche nicht besonders gut geeignet.

Das Microsoft Robotics Studio bietet durch die Modellsprache mit seinen guten Änder- und Übertragbarkeitseigenschaften hingegen gute Portierungsmöglichkeiten und Anpassungen des Aufgabenbereichs, jedoch zu Lasten der Effizienz.

### **3 Anforderungsdefinition**

Ein anderer Ansatz, der im Zuge dieser Diplomarbeit entwickelt werden soll, ist die Verwendung von Steuercodes, sog. Opcodes, zur direkten Kommunikation mit den Mikrocontrollern ohne Verwendung spezifischer Treiber zur Ansteuerung. So unterstützen eine Vielzahl handelsüblicher Mikrocontroller ein solches Verfahren, d.h. sie verfügen über Sub-Protokolle zur direkten Verarbeitung von Bytecodes, die an ihre Schnittstellen gesendet werden. Um dabei eine größtmögliche Unabhängigkeit von der Architektur des jeweiligen Robotersystems zu erreichen, sollen beliebige von einem Benutzer zuvor in einer Steuerungs-Software hinterlegte Steuerbefehlssätze, bestehend aus Bytecode-Folgen, von einer Arbeitsstation koordiniert direkt an die Mikrocontroller des Robotersystems gesendet und auch wieder ausgelesen werden. Die ausgelesenen Bytecode-Folgen sollen auf der anderen Seite wieder entschlüsselt und für den Benutzer in einer lesbaren Form auf der Arbeitsstation von der Steuerungs-Software repräsentiert werden.

Um eine Standortunabhängigkeit zwischen Benutzer bzw. Arbeitsstation und dem Robotersystem zu ermöglichen, soll das geplante Softwaresystem als Client-Server-basierte Architektur mit TCP/IP Protokoll konzipiert werden. D.h. die Software soll in eine in das Robotersystem integrierte Serverkomponente (hardwarenah und zentral) und eine auf der Arbeitsstation installierte Clientkomponente (hardwarefern und dezentral) unterteilt werden. Die Client-Software dient der Verwaltung der Steuerbefehlssätze und der Steuerung des Robotersystems durch den Benutzer. Die Server-Software dient als Schnittstelle zwischen Arbeitsstation und den im Robotersystem verwendeten Mikrocontrollern und koordiniert die Kommunikation. Durch diese Client-Server-basierte Architektur ist es auch möglich, dass mehrere Arbeitsstationen ortsunabhängig an der Steuerung und Überwachung eines einzigen Robotersystems teilhaben können. Eine Beispieldarstellung dieses neuen Konzepts ist in Abbildung 3 zu sehen. Abbildung 2 zeigt hingegen eine Beispieldarstellung für ein klassisches, treiberbasiertes System.

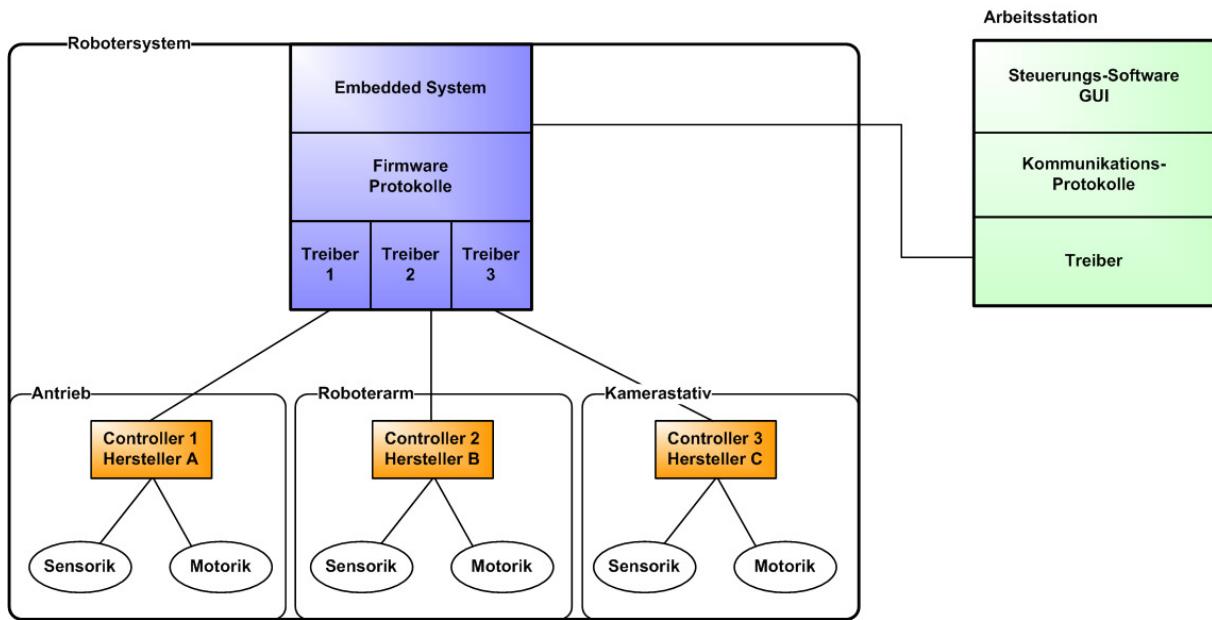


Abbildung 2: Vereinfachte Darstellung einer Beispiel-Infrastruktur eines herkömmlichen Robotersystems. Für jeden Mikrocontroller wird ein eigener Treiber benötigt.

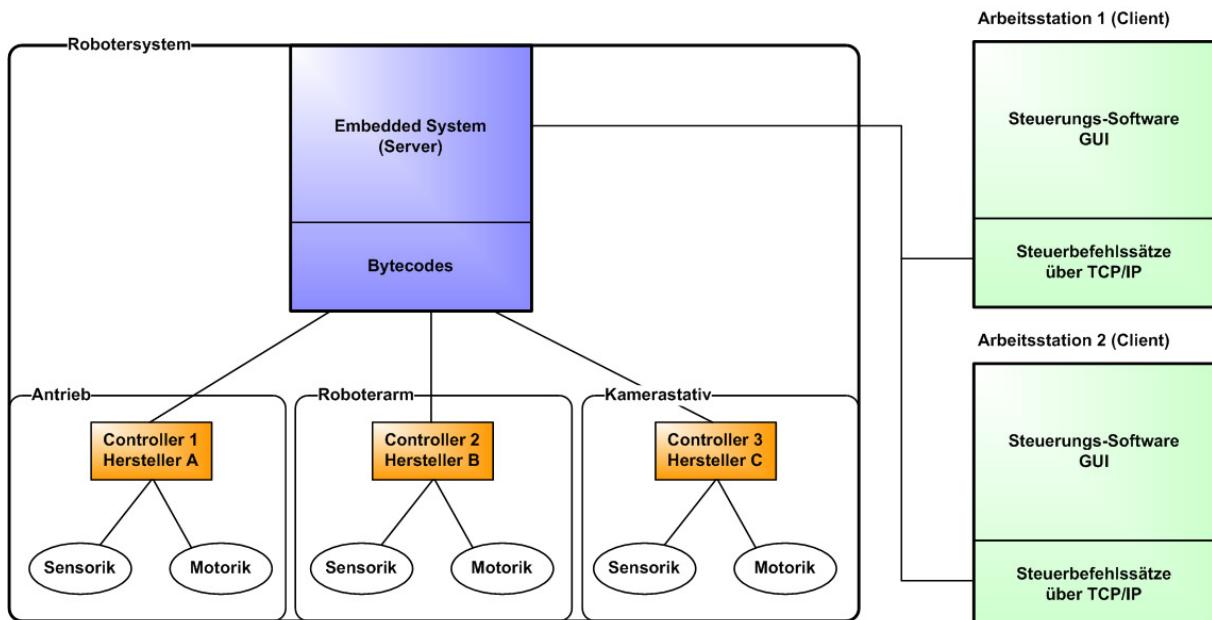
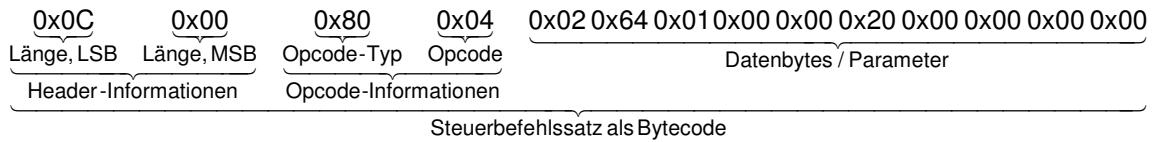


Abbildung 3: Neues Konzept im Vergleich – Die Steuerbefehlssätze werden von den Clients an den Server gesendet und von dort direkt als Bytecodes an die Mikrocontroller. Es werden keine Mikrocontroller-spezifischen Treiber benötigt.

Im Folgenden soll ein Beispiel für den Aufbau eines solchen Steuerbefehlssatzes zum Ansteuern eines Servomotors, der an einem Mikrocontroller angeschlossen ist, näher beschrieben werden. Der Steuerbefehlssatz ist wie folgt aufgebaut:



Das oben beschriebene Beispiel setzt sich neben den Längeninformationen im Header, in diesem Fall eine Länge von 12 Bytes, die nach dem Header folgen, aus folgenden Informationen für den Mikrocontroller zusammen, die sukzessiv eingelesen werden:

0x80: Der folgende Opcode ist ein direkter Befehl; es wird keine Antwort erwartet.

0x04: Ein Ausgang am Mikrocontroller soll angesteuert werden.

0x02: Es handelt sich um Ausgang 3.

0x64: Der Ausgang soll mit 100% Spannungsleistung angesteuert werden.

0x01: Ausgang direkt mit der gewünschten Spannung versehen; kein Hochfahren.

0x00: Keine Synchronisation mit anderen Ausgängen;

0x00: Keine Intervallschaltung;

0x20: Ausgang öffnen.

→ Der Motor an Ausgang 3 läuft mit 100% Leistung an.

Die letzten vier Bytes enthalten in diesem Fall nur den Wert Null, werden allerdings bei dem Beispiel-Opcode *0x04* dennoch benötigt, da der Mikrocontroller hierbei nach dem Opcode 10 Bytes als Parameter erwartet. Wird dies nicht berücksichtigt, werden die darauf folgenden Steuerbefehlssätze vom Mikrocontroller nicht erkannt, da die ersten vier Bytes des Folgesteuerbefehlssatzes noch als Parameter des vorangehenden Steuerbefehlssatzes verarbeitet werden. Somit wird der Folgebefehl nicht mehr korrekt erkannt, da Header- und Opcode-Informationen nicht mehr vorhanden sind und nicht mehr vom Mikrocontroller gelesen werden können. Folglich ist es zwingend nötig, dass die verwendeten Steuerbefehlssätze vom Benutzer korrekt zusammengestellt und in der Steuerungs-Software hinterlegt wurden.

Die Server-Software sendet die Steuerbefehlssätze bestehend aus Header, Opcode und Parametern als Bytecode direkt über die Schnittstelle an die jeweiligen Mikrocontroller. Sie liest die Antworten vom Mikrocontroller auch wieder als Bytecode aus. Ein Treiber oder Dienst zur Ansteuerung wird hierbei nicht benötigt. Benutzer hinterlegen die Steuerbefehle bzw. den gesamten Befehlssatz für den jeweiligen Mikrocontroller in der Steuerungs-Software. Deshalb müssen bei einer Portierung oder Veränderung des Systems nur die entsprechenden Steuerbefehle in der Software angepasst werden. Dies kann z.B. über ein dialogbasiertes Interface realisiert werden, so dass keine Implementierungsarbeit bei einer Portierung nötig ist.

Durch die Separierung der Steuerungs-Software in eine dezentrale, hardwareferne (Client-) und in eine zentrale, hardwarenahe (Server-) Komponente ist für den Benutzer eine positionsunabhängige Kommunikation mit dem System möglich. Des Weiteren kann so die Systemsteuerung von mehreren Arbeitsstationen gleichzeitig realisiert werden. Diese Client-Server-Architektur wird im Folgenden näher erläutert.

### **3.1 Der Client**

Der Client soll als dezentrale, hardwareferne, d.h. vom Standort des Robotersystems unabhängige Anwendung konzipiert werden. Als Betriebssystem-Plattform soll Windows zum Einsatz kommen. Der benötigte Funktionsumfang der Client-Software besteht aus fünf wichtigen Komponenten:

1. Verwaltung von Steuerbefehlen

Der Benutzer muss in der Lage sein, die für den Betrieb des Robotersystems benötigten Steuerbefehle in der Client-Software abspeichern, nachträglich bearbeiten und verwalten zu können. Die Steuerbefehle sollten dabei in einem für den Benutzer lesbaren Format gespeichert werden. D.h. die Steuerbefehle sollten vom Benutzer nicht als Byte- bzw. Binärkode eingetragen werden müssen. Vielmehr soll hexadezimaler Code als Repräsentation der Steuerbefehle von der Client-Software verarbeitet und angezeigt werden können.

## 2. Senden von Steuerbefehlen zum Server

Die Client-Software muss in der Lage sein, die hinterlegten Steuerbefehle an den Server verschicken zu können. Hierzu ist die Einbindung eines TCP/IP Protokolls in die Client-Software nötig. Dem Benutzer muss die Möglichkeit gegeben sein, eine beliebige, gültige IP-Adresse und eine Port Nummer einzustellen zu können, an die die Steuerbefehle gesendet werden sollen. Ferner müssen die einzelnen, hinterlegten Steuerbefehle vom Benutzer mit gewünschten Aktionen verknüpft werden können, um das Senden eines Steuerbefehls auslösen zu können. Solche Aktionen können Tastenbefehle oder Schaltflächen in einer dialogbasierten Benutzeroberfläche sein.

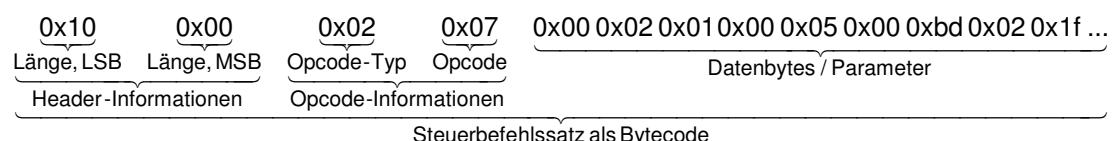
### 3. Empfangen von Steuerbefehlen vom Server

Die Client-Software muss in der Lage sein, die vom Server verschickten Steuerbefehle empfangen zu können. Hierfür soll ebenfalls das für das Senden verwendete TCP/IP Protokoll eingesetzt werden. Die Client-Software muss auf die vom Benutzer eingetragenen IP Adresse und Port Nummer lauschen und alle eingehenden Steuerbefehle erkennen und annehmen. Die empfangenen Steuerbefehle müssen dabei zur weiteren Verarbeitung zwischengespeichert werden.

## 4. Repräsentieren der Sensordaten

Um die empfangenen Steuerbefehlssätze auszuwerten und zu repräsentieren, muss der Benutzer definieren können, welche Steuerbefehlssätze wie ausgelesen werden sollen. Die Steuerbefehlssätze sind anhand des jeweiligen Opcodes eindeutig identifizierbar. D.h. der Benutzer muss die Opcodes in der Client-Software definieren können, die beim Empfangen beachtet werden sollen, und angeben, welche Bytestellen von dem jeweiligen durch den Opcode identifizierten Steuerbefehlssatz ausgewertet werden sollen.

Beispiel für einen empfangenen Code mit Daten als Antwort auf eine Sensor-abfrage:



Wichtig für die Zerlegung der Antwort sind zunächst die Opcode Informationen. In diesem Beispiel gibt der Opcode-Typ *0x02* der Client-Software Auskunft darüber, dass es sich um eine Antwort des Mikrocontrollers handelt; der Opcode *0x07* besagt, dass es sich bei den Folgedaten um einen Inputwert eines spezifischen Eingangs am Mikrocontroller handelt, an dem z.B. ein Sensor angeschlossen wurde. Über die Längeninformationen im Header kann die Client-Software nun die Länge der zu verarbeitenden Datenbytes ermitteln. Im oben beschriebenen Beispiel hat der Steuerbefehlssatz laut Header eine Länge von 16 Bytes. Davon müssen die zwei Bytes Opcode-Informationen abgezogen werden, so dass die folgenden 14 Bytes die eigentlichen Informationen des Sensorwertes enthalten, die von der Client-Software repräsentiert werden sollen.

Nun ist zu beachten, dass sich das Format der Steuerbefehle verschiedener Mikrocontroller unterscheiden kann. So können die Länge und der Aufbau von Header und Opcode je nach Mikrocontroller differieren. Für die Bearbeitung von Steuerbefehlen muss die Client-Software Informationen darüber haben, an welcher Bytestelle des Steuerbefehlssatzes sich die Längeninformation für die Datenbytes befindet (beginnend mit Byte 0) und an welcher Stelle der Opcode-Typ bzw. der eigentliche Opcode zu finden ist. Diese Informationen muss der Benutzer bei der Konfiguration der Client-Software einmalig für die verwendeten Mikrocontroller hinterlegen, d.h. die Formatierung der zu erwartenden Steuerbefehlssätze definieren.

## 5. Login für Benutzerzugangskontrolle

Da die Server-Software über eine Zugangskontrolle für die Benutzer verfügen soll, ist es notwendig, dass die Client-Software über eine Login-Funktion verfügt, über die sich der Benutzer beim Server authentifizieren kann.

In der folgenden Abbildung 4 wird ein Anwendungsfall-Diagramm gezeigt, welches aus dem Funktionsumfang resultiert.

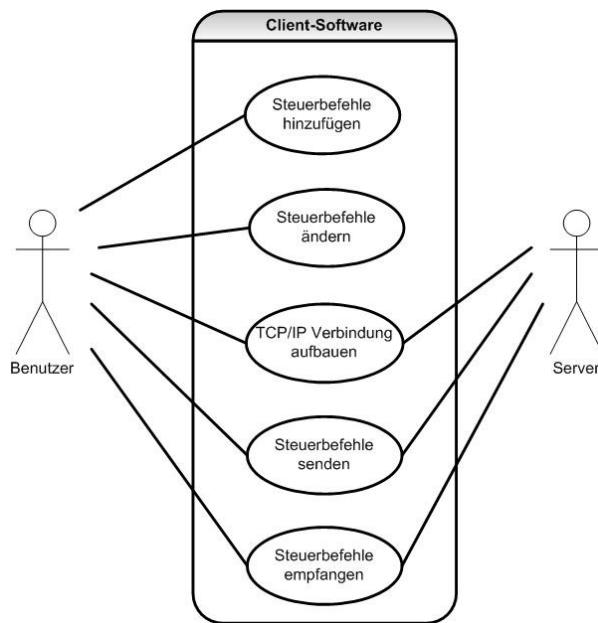


Abbildung 4: Anwendungsfall-Diagramm für die Client-Software.

### 3.1.1 Client Systemanforderungen

Um die reibungslose Funktion der Client-Software gewährleisten zu können, sind folgende Anforderungen an die Hard- und Software zwingend erforderlich:

- Betriebssystem: Windows 95 oder höher
- ein Single-Core-Prozessor mit mindestens 400 MHz
- 64 MB Arbeitsspeicher oder mehr
- mindestens 100 MB freier Speicherplatz
- installiertes TCP/IP Protokoll
- LAN, Wireless LAN oder Internet-Anschluss mit mindestens 14 Kbits/s symmetrische Übertragungsrate

Dies sind die Mindestanforderungen an das System, auf dem die Client-Software zum Einsatz kommen soll, um eine grundlegende Funktion sicherstellen zu können. Die Anforderungen können aber je nach Einsatzgebiet differieren bzw. ansteigen. So kann sich die Anforderung an die Speichergröße je nach Art der Client-Konfiguration,

d.h. der Anzahl verwendeter Steuerbefehle, verändern. Zudem erhöht sich der Prozessor- und Arbeitsspeicherbedarf mit der Anzahl zu verarbeitender und zu repräsentierender Sensordaten und der Art der grafischen Darstellung dieser Daten. Die Anforderung an die Übertragungsrate kann mit der Anzahl zeitgleich zu versendender Steuerbefehle ebenfalls differieren. Bei einer Übertragungsrate von 14 Kbits/s und einer durchschnittlichen Größe eines Steuerbefehlssatzes von 16 Bytes können so theoretisch etwa 1750 Steuerbefehlssätze gleichzeitig pro Sekunde verschickt oder empfangen werden, wenn keine anderen Prozesse die Verbindung belasten.

### **3.1.2 Client Sicherheitsmanagement**

Da die Client-Software zum Datenaustausch mit dem Server Verbindungen über Wireless LAN oder das Internet nutzen und der Server theoretisch öffentlich zugänglich sein kann, müssen präventive Maßnahmen ergriffen werden, um diese Verbindungen vor einem Eingreifen unbefugter Dritter zu schützen. Hierbei können standardisierte Verfahren und Techniken aus der Netzwerksicherung verwendet werden. Wireless LAN Verbindungen sollten durch eine Netzwerkauthentifizierung wie z.B. Wi-Fi Protected Access (WPA) oder WPA2 und unter Verwendung einer Temporal Key Integrity Protocol (TKIP) oder einer Advanced Encryption Standard (AES) Datenverschlüsselung vor dem Zugriff Dritter geschützt werden. Zudem würde sich eine MAC-Adressenfilterung anbieten.

Bei einer Verbindung über das Internet sollten für den Datenaustausch Verschlüsselungsprotokolle wie SSL oder SSH zum Einsatz kommen. Diese Netzwerkprotokolle werden von Windows bereitgestellt. Zudem sollte auf dem Client-System eine Firewall als zusätzlicher Schutz vor unbefugtem Zugang eingerichtet sein. Zusätzlich zu den genannten Sicherungsmaßnahmen für die Client-Server-Verbindung muss eine Passwort geschützte Benutzerverwaltung den unbefugten Zugang zum Server unterbinden, um ein Fremdeinwirken in die Robotersteuerung zu verhindern. Dies kann über die in Windows integrierten Zugangskontrollmechanismen wie die Systemanmeldung erfolgen. Eine Zutrittskontrolle für die Client-Arbeitsstation durch einen extra gesicherten Raum kann ebenfalls als zusätzliche Sicherungsmaßnahme angewendet werden.

### 3.1.3 Client Benutzbarkeitsanforderungen

Für die Konfiguration der Client-Software ist Fachwissen über die Funktionsweise von Steuerbefehlen erforderlich. Der Benutzer muss beim Hinterlegen der gewünschten Steuerbefehle über entsprechendes Informationsmaterial verfügen, das Auskunft über die vom Mikrocontroller unterstützten Steuerbefehlssätze gibt. Für die Bedienung der Client-Software im Betriebszustand wird kein besonderes Fachwissen vorausgesetzt. Benutzer müssen lediglich Grundkenntnisse in der Bedienung von Windows-basierten Oberflächen besitzen. Die GUI der Client-Software sollte dementsprechend übersichtlich gestaltet sein und über eine gut strukturierte Menüführung verfügen. Abbildung 5 zeigt ein Aktivitätsdiagramm für die Menüführung.

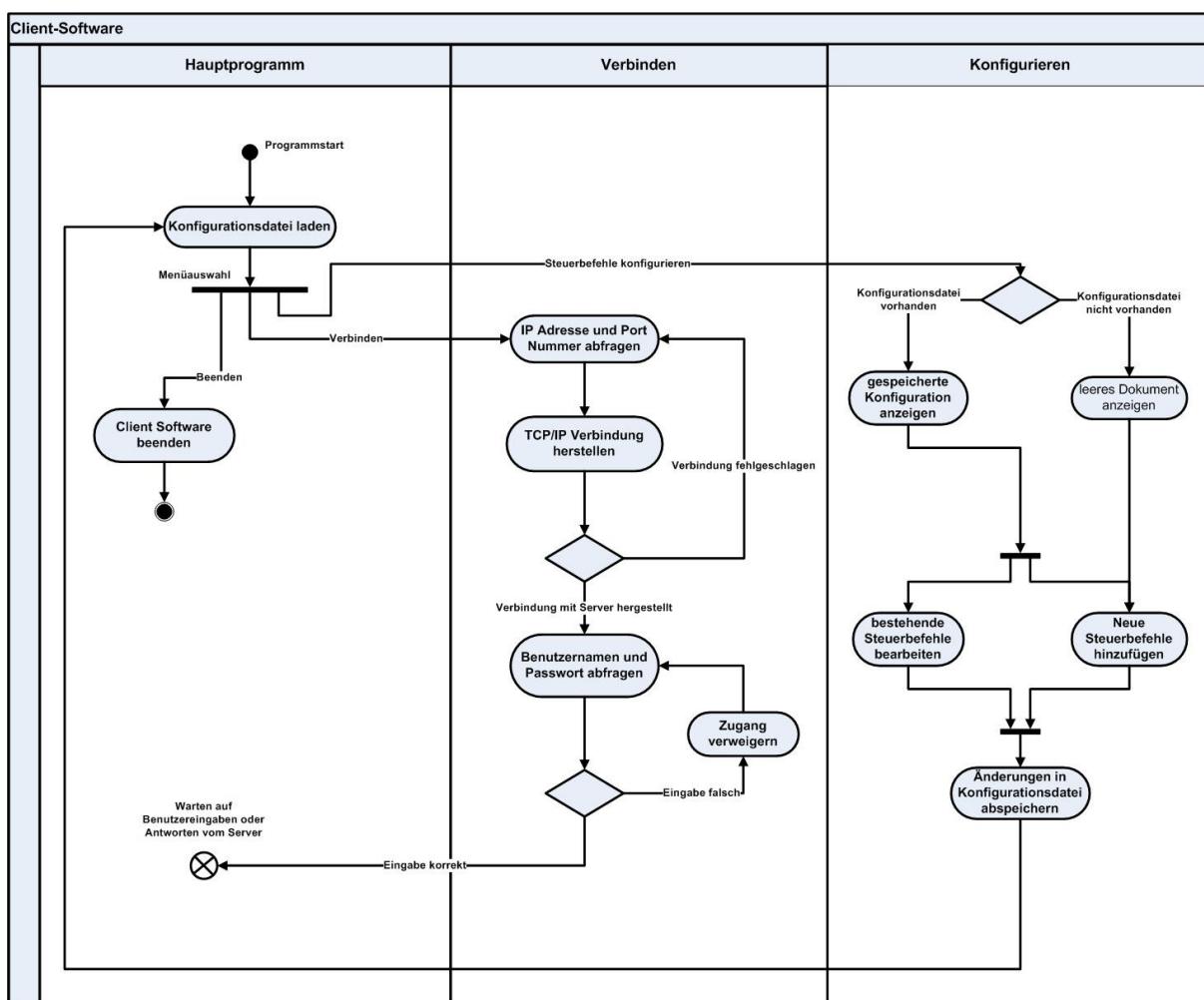


Abbildung 5: Aktivitätsdiagramm für die Menüführung der Client-Software.

### **3.2 Der Server**

Der Server soll als zentrale, hardwarenahe, d.h. in dem Robotersystem integrierte Anwendung konzipiert werden. Die Hauptaufgabe des Servers besteht darin, die vom Client empfangenen Steuerbefehlssätze zu verarbeiten, in Bytecodes umzuwandeln und an die angeschlossenen Mikrocontroller zu senden. Antworten der Mikrocontroller sollen ausgelesen, vom Bytecode in ein hexadezimales Format umgewandelt und an die Clients gesendet werden.

Als Plattform soll Windows CE zum Einsatz kommen. Windows CE bietet einige entscheidende Vorteile für die Verwendung in einem Robotersystem. Es unterstützt verschiedene Prozessorfamilien und bietet dadurch eine gute Systemkompatibilität. Multitasking und Multithreading sowie 256 Prioritätsstufen sind möglich, wodurch auch zeitkritische und komplexere Anwendungen auf einem Windows CE Betriebssystem möglich sind. Ferner wird eine umfangreiche Protokoll- und Treiberunterstützung geboten. Programmierer können auf eine Untermenge der Win32 API zurückgreifen, wodurch der Implementierungsaufwand verringert werden kann. Ebenfalls wichtig in der Robotertechnik ist die Eigenschaft von Windows CE, dass der Speicherbedarf des kleinsten möglichen Image bei nur etwa 200 kB liegt und die Größe von Windows CE im Allgemeinen skalierbar ist. Dadurch kann je nach Einsatzgebiet der Funktionsumfang und somit die Größe dem Bedarf angepasst werden. Windows CE bietet ebenfalls spezielle Unterstützung für batteriebetriebene Geräte, d.h. es verfügt auch über Strom-sparende Funktionen, welche in der Robotertechnik von Interesse sind. Ein weiterer nicht unerheblicher Vorteil von Windows CE ist der Service seitens Microsoft. So stellt Microsoft die Entwicklungsumgebung *embedded Visual C++* und SDKs für Visual Studio.NET kostenlos zur Verfügung. Zudem ist das .NET Compact Framework für Windows CE frei erhältlich.

Hinzu kommt, dass Windows CE eine weit verbreitete Plattform ist, die in vielen Geräten und Systemen eingesetzt wird. Beispiele hierfür sind die bereits erwähnten LANpointCE Terminals der Firma Intelligent Instrumentation<sup>2</sup> oder aber auch handelsübliche Geräte wie Pocket PCs, Handheld PCs oder die als Mobiltelefon bekannten Smartphones. Auf Grund der großen Verfügbarkeit solcher Geräte sind die Anschaffungskosten gering. Diese handelsüblichen Geräte sind zudem kompakt, haben einen geringen Energiebedarf und besitzen eine Vielzahl an integrierten Schnittstellen wie USB, Infrarot, Wireless LAN und Bluetooth. Daher sind solche Geräte für den

---

<sup>2</sup> LANpointCE Terminals: <http://www.lanpoint.com/>

Einsatz in einem Robotersystem eine praktikable und kostengünstige Wahl. Ein Beispiel für einen solchen Einsatz sind die zweibeinigen Roboter Mr. DD Junior und Mr. DD Junior 2 (siehe Abb. 6) der Robocup Gruppe *Darmstadt Dribblers* der Technischen Universität Darmstadt<sup>3</sup>.



**Abbildung 6: Der Roboter Mr. DD Junior 2 der Darmstadt Dribblers.**

Der benötigte Funktionsumfang der Server-Software besteht aus sieben wichtigen Komponenten:

1. Umwandeln von Steuerbefehlen

Da die Clients die Steuerbefehle in einem hexadezimalen Format versenden und verarbeiten, der Server allerdings mit den Mikrocontrollern im Bytecode-Format kommuniziert, muss die Server-Software in der Lage sein, die hexadezimalen Steuerbefehle in Bytecodes bzw. Bytecodes wieder in ein hexadezimales Format umrechnen zu können.

2. Konfiguration der Schnittstelle zum Mikrokontroller

Die Server-Software muss über eine entsprechende Verbindung zum Mikrocontroller verfügen, die zur Kommunikation verwendet werden kann. Für eine solche Verbindung soll die Server-Software die gängigsten Schnittstellen un-

---

<sup>3</sup> Darmstadt Dribblers: <http://www.dribblers.de/>

terstützen, zu denen parallele, serielle, USB, Bluetooth und Infrarot Verbindungen zählen. Der Benutzer muss verbindungsspezifische Parameter wie Parität, Start-/Stoppbits, Bytegröße und Baudrate in der Server-Software frei konfigurieren können.

### 3. Empfangen von Steuerbefehlen vom Client

Die Server-Software muss in der Lage sein, die von den Clients versendeten Steuerbefehle zu empfangen. Hierzu ist eine Unterstützung des TCP/IP Protokolls nötig. Bestehende TCP/IP Verbindungen mit den Clients müssen getrennt abgehört und entsprechende Empfangsereignisse abgefangen und verarbeitet werden können.

### 4. Senden von Steuerbefehlen an den Mikrokontroller

Nachdem die von den Clients gesendeten Steuerbefehle von der Server-Software in ein Bytecode-Format umgewandelt wurden, müssen sie von dieser an den Mikrokontroller gesendet werden. Für die Übertragung muss die Server-Software die vom Benutzer spezifizierte Schnittstellen-Konfiguration verwenden.

### 5. Auslesen der Steuerbefehle vom Mikrokontroller

Die Server-Software muss in der Lage sein, die vom Mikrokontroller gesendeten Steuerbefehle (Antworten bzw. Sensordaten) über die bestehende Verbindung empfangen und für eine weitere Verarbeitung speichern zu können.

### 6. Senden von Steuerbefehlen an die Clients

Die von dem Mikrokontroller an den Server übertragenen Steuerbefehle (Antworten bzw. Sensordaten) sollen an die Clients gesendet werden, nachdem sie in ein hexadezimales Format umgewandelt wurden. Das Versenden der Steuerbefehle soll im Broadcasting-Verfahren geschehen, so dass alle verbundenen Clients alle Daten erhalten. Die Weiterverarbeitung der Daten erfolgt im jeweiligen Client je nach Konfiguration, d.h. der Benutzer des Clients entscheidet, welche Daten repräsentiert werden sollen (siehe auch *3.1 Repräsentieren der Sensordaten*).

## 7. Benutzer- und Rechteverwaltung

Im Zuge des Sicherheitsmanagements soll die Server-Software über eine Benutzerverwaltung verfügen, die den Zugang der Clients bzw. deren Benutzer reglementiert (siehe auch *3.1 Login für Benutzerzugangskontrolle*). Hierzu müssen die einzelnen Benutzerkonten auf dem Server konfigurierbar sein. Zudem soll die Server-Software über eine einfache Rechteverwaltung verfügen, welche es ermöglicht, dem einzelnen Benutzer die Rechte einzuräumen, Steuerbefehle an den Mikrokontroller senden und von diesem empfangen zu können oder die Steuerbefehle nur senden oder nur empfangen zu können. So können auch Kollisionen von Befehlen verschiedener Benutzer serverseitig durch eine entsprechende Benutzerverwaltung abgefangen und vermieden werden.

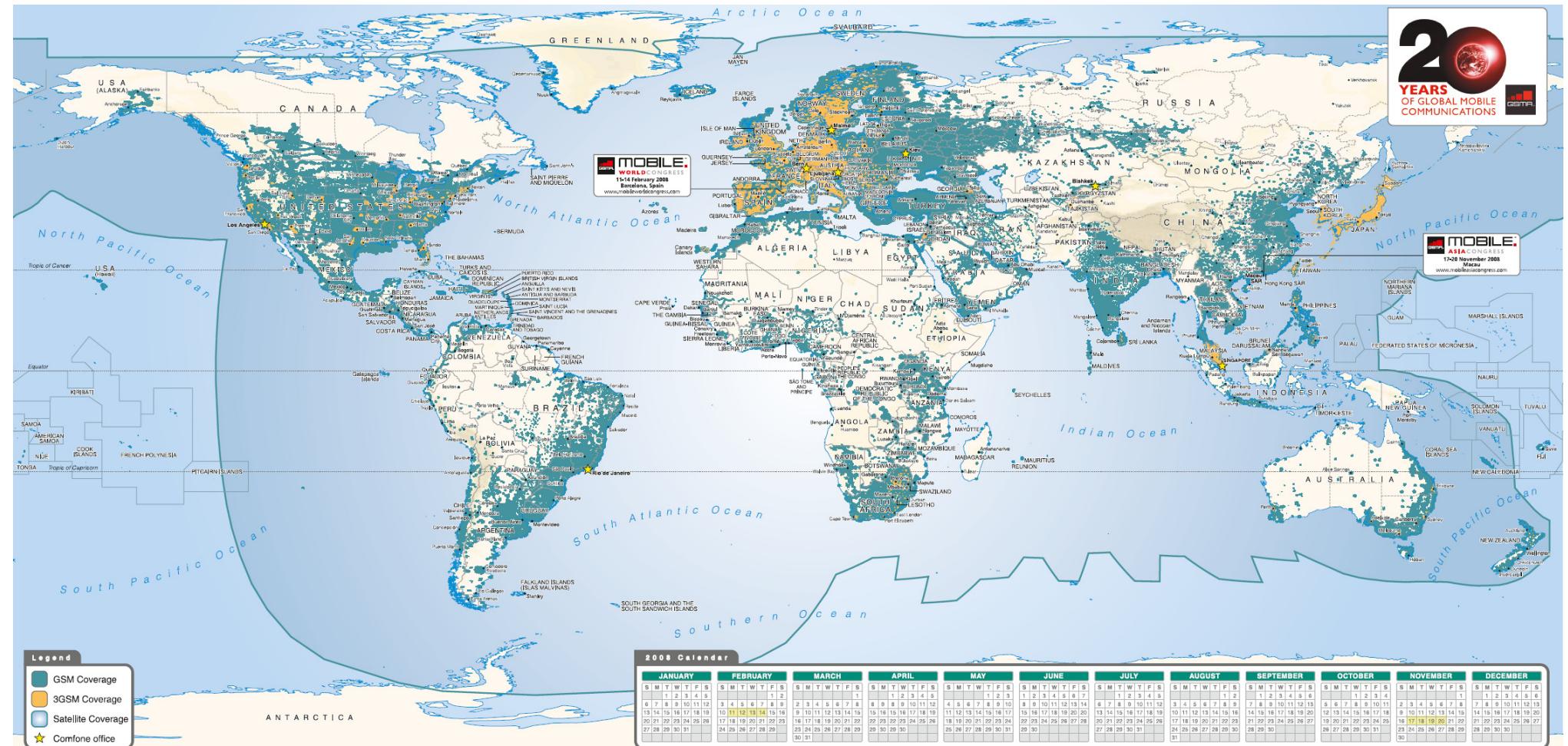
### 3.2.1 Verwendung neuer Übertragungstechniken

Ein weiterer Vorteil von Windows CE ist die Unterstützung von verschiedenen Kommunikationsstandards aus dem Mobiltelefonbereich, wie es heutzutage z.B. vermehrt in sogenannten Smartphones mit Windows CE Betriebssystem Verwendung findet. So unterstützt Windows CE Netzstandards wie Global System for Mobile Communications (GSM) und dem neuen Universal Mobile Telecommunications System (UMTS) bzw. 3GSM. Für den Datenaustausch über die genannten Netze werden Dienste wie General Packet Radio Service (GPRS) für GSM und High Speed Downlink Packet Access (HSDPA) für UMTS unterstützt und entsprechende Protokolle von Windows CE bereitgestellt. Diese Netzstandards zeichnen sich dank ihrer digitalen Technik durch eine besonders geringe Fehlerrate und eine geringe Störanfälligkeit bei Datenübertragungen aus. Zudem sind die GSM und UMTS Module heutzutage besonders platz- und energiesparend, da sie für den Mobiltelefonmarkt konzipiert wurden.

Daraus entsteht die Idee, diese Netze zur besseren Erreichbarkeit des Servers als zusätzliche Schnittstelle neben LAN und Wireless LAN in Betracht zu ziehen, um eine größtmögliche Positionsunabhängigkeit zwischen Clients und Server zu erreichen. So ist eine LAN Verbindung kabelgebunden und für eine Wireless LAN Verbindung ist eine Basisstation in der Nähe des Robotersystems erforderlich. Das GSM Netz hingegen ist auch über Satelliten nahezu weltweit verfügbar; das 3GSM / UMTS

Netz ist in Europa beinahe flächendeckend verfügbar und befindet sich im stetigen Ausbau (siehe Abb. 7 auf der nächsten Seite). Verfügt der Server über ein GSM- oder UMTS-Modul, kann er sich von einem beliebigen Standort mittels dieser Netze mit dem Internet über ein TCP/IP Protokoll verbinden und ist somit für die Clients über eine IP-Adresse erreichbar.

Allerdings sollte diese Schnittstelle neben LAN und Wireless LAN nur als dritte Option zum Datenaustausch in Betracht gezogen werden, da die UMTS- und vor allem die GSM-Netze im Vergleich nur eine geringe Bandbreite bieten. So kann über eine GPRS Datenverbindung nur eine Übertragungsrate von 9 bis 14 Kbits/s im Idealfall erreicht werden. Da diese Werte allerdings örtlich bedingten starken Schwankungen unterliegen können, stellen sie bei höherem Datenvolumen einen Flaschenhals dar und sind wegen entstehender Verzögerungen bei zeitkritischen Steuerungssystemen nicht zu empfehlen. Zudem verursacht die Nutzung der GSM- bzw. UMTS-Netze weitere Kosten, da eine Gebührenabhängigkeit besteht.



**Abbildung 7: Die weltweite Netzabdeckung 2008 [GSM 2008].**

### **3.2.2 Server Systemanforderungen**

Um die reibungslose Funktion der Server-Software gewährleisten zu können, sind folgende Anforderungen an die Hard- und Software zwingend erforderlich:

- Betriebssystem: Windows CE 3.0 oder höher
- Prozessor mit mindestens 133 MHz
- 32 MB Arbeitsspeicher oder mehr
- mindestens 2 MB freier Speicherplatz
- installiertes TCP/IP Protokoll
- LAN-, Wireless LAN- oder Internet-Anschluss mit mindestens 14 Kbits/s symmetrische Übertragungsrate
- parallele, serielle, USB, Bluetooth und/oder Infrarot Schnittstellen
- optional ein GSM-Modul mit GPRS Unterstützung oder ein UMTS-Modul mit HSDPA Unterstützung

Dies sind die Mindestanforderungen an das System, auf dem die Server-Software zum Einsatz kommen soll, um eine grundlegende Funktionalität sicherstellen zu können. Die Anforderungen können aber je nach Einsatzgebiet differieren bzw. ansteigen. Dies ist abhängig von der Anzahl von Clients und Mikrokontrollern, die zeitgleich mit der Server-Software verbunden sind. Zudem spielt für die Höhe der Anforderungen auch die Größe der Datenströme eine Rolle, also die Anzahl der Steuerbefehle, die von der Serversoftware zeitgleich ausgelesen, verarbeitet, versendet und empfangen werden müssen. Dies kann den Arbeitsspeicherbedarf und die Prozessorauslastung, aber auch die Anforderungen an die Übertragungsrate zwischen Server und Clients erhöhen. Wenn man von einer durchschnittlichen Größe eines Steuerbefehlssatzes von 16 Bytes ausgeht, können bei einer Übertragungsrate von 14 Kbit/s theoretisch etwa 1750 Steuerbefehlssätze gleichzeitig pro Sekunde verschickt oder empfangen werden, sofern keine anderen Prozesse die Verbindung belasten.

In der Theorie können alle gängigen Windows CE Versionen ab 3.0 unterstützt werden. Dazu zählen Handheld PC, Pocket PC und Smartphone Versionen. Die Server-Software muss ggf. auf die jeweils unterstützte Auflösung angepasst und für die verwendete Windows CE Version und Prozessorfamilie (z.B. MIPS, ARM und PXA) neu kompiliert werden.

### **3.2.3 Server Sicherheitsmanagement**

Auf dem Server sollten bei der Nutzung von Wireless LAN oder dem Internet als Schnittstelle zwischen Client und Server die gleichen Authentifizierungs- und Verschlüsselungsverfahren zum Einsatz kommen wie bereits in 4.1.2 für den Client beschrieben, um den Datenaustausch vor unbefugtem Zugriff Dritter zu schützen. Da sich der Server hardwarenah auf dem Robotersystem befinden soll und so bei mobilen Systemen bzw. Systemen für den Außeneinsatz eine Zutrittskontrolle nicht gewährleistet werden kann, sollten zusätzliche Sicherungsmaßnahmen gegen unbefugten Zugang zur Benutzungsoberfläche des Servers eingerichtet werden.

Da als Plattform Windows CE zum Einsatz kommen soll, können die vom Betriebssystem bereitgestellten Verfahren hierfür genutzt werden. Dazu zählen Passwort geschützte Benutzeranmeldung oder auch ein biometrischer Fingerabdruckscanner. Zusätzlich wäre es möglich im Betriebszustand, nachdem die Server-Anwendung gestartet wurde, die Benutzungsoberfläche in einen Kiosk-Modus zu versetzen. D.h. auf das Betriebssystem oder auf andere Anwendungen kann kein Zugang mehr erlangt werden. Als Zugriffskontrolle zur Server-Software kann deren Benutzerverwaltung dienen.

### **3.2.4 Server Benutzbarkeitsanforderungen**

Für die Konfiguration, d.h. die Einstellungen der Schnittstellen zu den mit dem Server-System verbundenen Mikrocontrollern, ist Fachwissen erforderlich. Zudem müssen hierfür Schnittstellen-spezifische Informationen zu den einzelnen Mikrocontrollern vorliegen, um die korrekten Parameter einzustellen zu können. Für die Bedienung der Server-Software an sich wird kein besonderes Fachwissen vorausgesetzt. Der Benutzer muss lediglich Grundkenntnisse in der Bedienung von Windows-basierten Oberflächen besitzen. Die GUI der Server-Software sollte dementsprechend übersichtlich gestaltet sein und über eine gut strukturierte Menüführung verfügen. Befindet sich die Server-Software im Betriebszustand, d.h. wurde sie konfiguriert, ist keine weitere Bedienung durch einen Benutzer mehr erforderlich.

### 3.3 Ausführungsbeispiel

Auf der Arbeitsstation befindet sich die dezentrale, hardwareferne Steuerungs-Software (Client) als Benutzungsschnittstelle für den Benutzer. Das Robotersystem verfügt über die zentrale, hardwarenahe Steuerungs-Software (Server) für die Kommunikation zwischen Arbeitsstation und den angeschlossenen Mikrocontrollern (siehe Abb. 8). Benutzer hinterlegen die zur Systemsteuerung benötigten Steuercodes bzw. ganze Steuerbefehlssätze zur Direktsteuerung des Robotersystems sowie zum Auslesen der sensorischen Daten über eine grafische Oberfläche auf der Arbeitsstation und dem eingebetteten System; sie verknüpfen sie mit den gewünschten Funktionen (z.B. Tasturbefehlen) und weisen sie den im System verwendeten Mikrocontrollern zu. Nach dieser Konfiguration ist die Steuerungs-Software in der Lage, die Systemsteuerung zu gewährleisten.

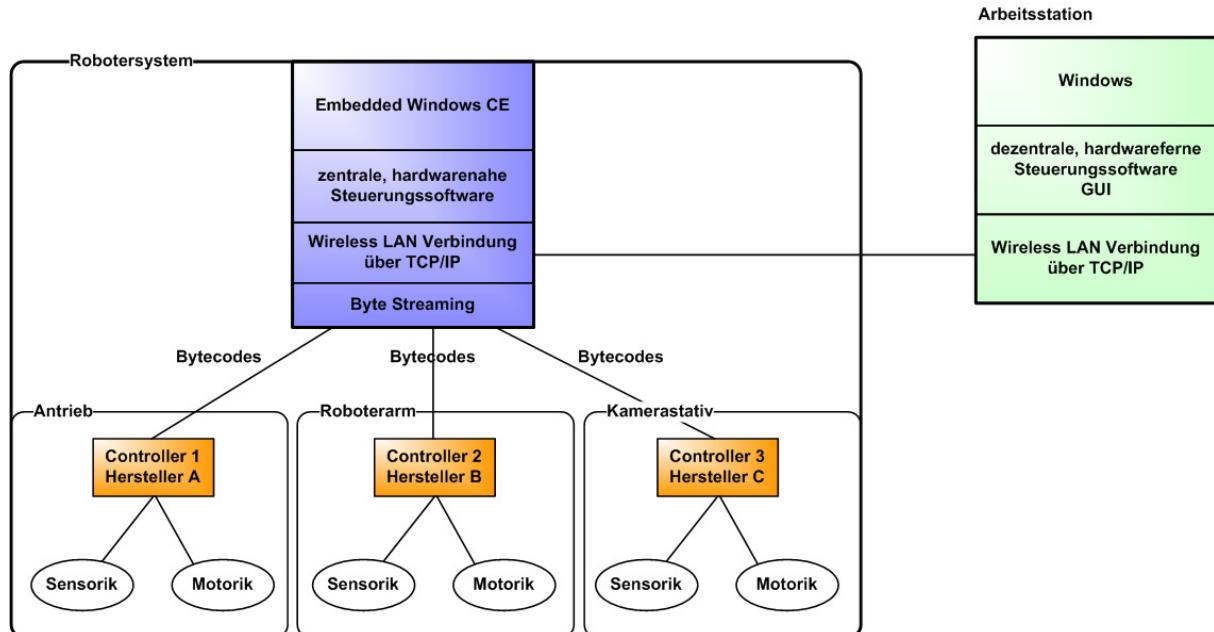
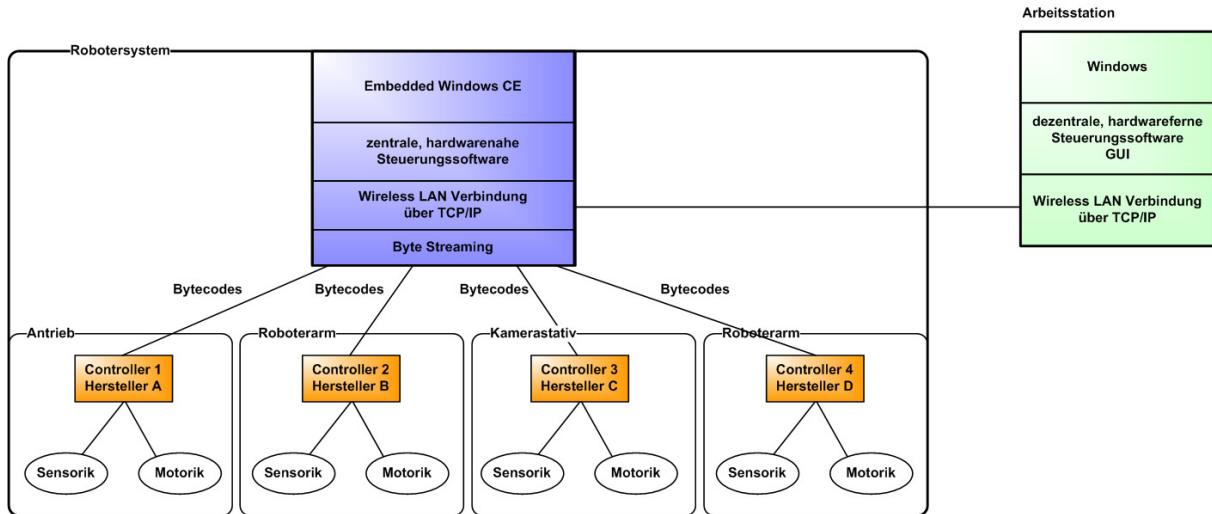


Abbildung 8: Beispiel-Infrastruktur eines Robotersystems.

Wird diesem Robotersystem nun ein weiterer Roboterarm von einem anderen Hersteller und mit einem anderen Mikrocontroller hinzugefügt (siehe Abb. 9), müssen Benutzer lediglich die benötigten Steuerbefehle bzw. den Befehlssatz für den hinzugefügten Mikrocontroller in der Steuerungs-Software mittels der grafischen Oberfläche ergänzen. Eine größere Anpassung der Steuerungs-Software ist nicht nötig.



**Abbildung 9: Beispiel-Infrastruktur eines Robotersystems mit zusätzlichem Controller.**

### 3.4 Mögliche Anwendungsszenarien

Das in dieser Diplomarbeit vorgestellte Client-Server-basierte System umfasst eine sehr große Bandbreite an möglichen Anwendungsszenarien und Einsatzgebieten. So ist es in der Theorie möglich, jeden Mikrocontroller anzusteuern oder Daten auszulesen, vorausgesetzt der Mikrocontroller unterstützt eine direkte Ansteuerung über Bytecodes und verfügt zudem über eine vom System unterstützte Schnittstelle wie z.B. ein serieller Anschluss oder USB. Außerdem sollten die benötigten Bytecodes dem Benutzer bekannt sein.

Sind diese Voraussetzungen erfüllt, können mit diesem Steuerungssystem nicht nur Robotersysteme gesteuert werden, sondern es können vielmehr alle erdenklichen Mess- und Regelsystem theoretisch bedient und überwacht werden. Die Client-Server-basierte Architektur sowie die unterstützten Übertragungstechniken und die damit erzielte Standortunabhängigkeit erweitern die möglichen Anwendungsgebiete des Steuerungssystems zusätzlich. Das Spektrum reicht dabei von Wetter- und seismologischen Stationen bis hin zur Druck- und Gasmesstechnik.

In Kapitel 5 werden zwei konkrete Anwendungsszenarien als Testumgebung vorgestellt und die Durchführung erläutert.

## **4 Definition des Testsystems**

Folgend wird das Testsystem definiert, welches zur Entwicklung und zum Testen der Client-Server-basierten Robotersteuerung verwendet werden soll.

### **4.1 Der Testroboter**

Um die Funktionalität der Client- und Server-Software testen und demonstrieren zu können, ist es von Nöten, einen geeigneten Testroboter zu konstruieren. Hierfür soll das Mindstorms NXT System verwendet werden. Neben einem handelsüblichen 32-bit ARM7 Mikrocontroller mit 256 Kbytes Speicher und einer Bluetooth-Schnittstelle werden von dem Mindstorms NXT System geeignete sensorische Komponenten wie z.B. ein Ultraschall-, Kompass- und Lagesensor bereitgestellt. Dieses Robotersystem ist zwar nicht robust genug für ein längeres Einsatzszenario im Außenbereich, allerdings ist es auf Grund der Flexibilität für Testzwecke gut geeignet. Folgend werden die einzelnen Komponenten des Testroboters aufgeführt, der für diese Diplomarbeit konstruiert wurde.

#### **4.1.1 Mikrocontroller**

Der Mindstorms NXT Mikrocontroller verwendet einen Atmel 32-bit ARM Hauptprozessor (Produktnummer AT91SAM7S256), welcher auf einem 32-bit ARM7TDMI RISC Prozessor basiert und mit einer Taktrate von 48 MHz arbeitet. Dieser Prozessortyp ist in der Multimedia- und Kommunikationstechnik weit verbreitet und findet heutzutage häufig in portablen Geräten Gebrauch. Der Hauptprozessor hat im Mindstorms NXT Zugriff auf einen 256 KByte großen Flash-Hauptspeicher und einen 64 KByte großen Arbeitsspeicher. Dadurch ist es möglich, auf dem NXT eine Firmware zu installieren, welche im Hauptspeicher hinterlegte Programme zur reaktiven Steuerung des Mindstorms NXT speichern und verarbeiten kann. So können während der Testdurchführungen parallel zur geplanten Direktsteuerung durch die Client-Software rudimentäre, autonome Steueralgorithmen auf dem Mikrocontroller laufen,

damit der Testroboter im Falle von Verbindungsstörungen zwischen Server- und Client-Software sich nicht durch unkontrollierte Fahrmanöver selbst gefährdet.

Als Co-Prozessor verwendet der Mindstorms NXT einen Atmel 8-bit AVR Prozessor (Produktnummer ATmega48), der mit einer Taktrate von 8 MHz arbeitet. Dieser Co-Prozessor wird für mathematische Berechnungen verwendet. Er hat hierfür im NXT Zugriff auf einen 4 KByte großen Flash-Hauptspeicher und einen 512 Byte großen Arbeitsspeicher.

Der Mindstorms NXT Mikrocontroller bietet zur Kommunikation mit seiner Umwelt eine Bluetooth 2.0 + EDR Schnittstelle, welche ebenfalls ein Standard Parallel Port (SPP) Profil unterstützt [LEGO 2006c]. Daneben wird zusätzlich auch eine USB 2.0 Schnittstelle mit einer Übertragungsrate von 12 Mbit/s bereitgestellt. Der Mikrocontroller wird von einem Lithium-Polymer Akkupack gespeist.

Eine der wichtigsten Anforderungen an das Testsystem ist die Verfügbarkeit der vom Mikrocontroller unterstützen Opcodes, die Definition des Steuerbefehlsformats sowie die Schnittstellenspezifikationen. Die LEGO Group hat hierfür wissenschaftliche Dokumente veröffentlicht, die die entsprechenden Details bereitstellen [LEGO 2006a-c]. Somit erfüllt der Mikrocontroller des Mindstorms NXT alle Kriterien zur Nutzung der geplanten Client-Server-Software und ist als Testsystem geeignet.

#### 4.1.2 Mechanik

Diese Diplomarbeit hat nicht zum Ziel, neue Maßstäbe auf dem Gebiet des Maschinenbaus zu setzen. Dennoch soll der Testroboter über eine grundlegende Funktionalität wie eine adäquate mechanische Plattform mit verschiedenen Aktoren verfügen, um die Fähigkeiten des Systems hinreichend testen zu können. So soll sich der Testroboter auch fortbewegen können und einen entsprechenden mechanischen Antrieb besitzen. Realisiert wird dies durch ein federgelagertes Kettenlaufwerk, welches durch zwei separat anzusteuernde Servomotoren an der Front des Testroboters angetrieben wird. Durch diese Antriebslösung erlangt der Testroboter ein hohes Maß an Mobilität. Er kann auf einer Stelle manövriren, ist geländefähig und besitzt dennoch eine adäquate Höchstgeschwindigkeit.

Als zusätzlichen Aktor besitzt der Testroboter eine schwenkbare Plattform, die von einem Servomotor angetrieben wird. Diese Plattform dient der Befestigung des Win-

dows CE Testgeräts Samsung SGH-i600 (siehe auch *4.2 Der Server Prototyp*) auf dem Testroboter. Die Schwenkfunktion soll dazu dienen, die Ausrichtung einer Videokamera unabhängig von der Ausrichtung des Testroboters zu simulieren. Insgesamt verfügt der Testroboter also über drei Aktoren in Form von drei Servomotoren, die vom Mikrocontroller angesteuert werden können.

#### 4.1.3 Sensorik

Um entsprechende sensorische Daten vom Testroboter durch die Client-Software repräsentieren zu können, sollen im Testroboter verschiedene sensorische Komponenten verbaut sein, die unterschiedliche Datenwerte liefern und vom Mikrocontroller ausgelesen werden können. So verfügt der Testroboter insgesamt über sieben verbaute Sensoren. Dazu zählen drei Rotationssensoren, ein Lage- und Beschleunigungssensor, ein Kompasssensor, ein Ultraschallsensor und ein aktiver Lichtsensor. Die Rotationssensoren sind jeweils in den drei Servomotoren integriert und geben Auskunft über deren Achsstellungen bis zu einem Grad genau. Durch diese Sensoren können verschiedene Messungen durchgeführt werden. So können die Geschwindigkeit des Testroboters, der zurückgelegte Weg und ggf. Störungen wie z.B. eine Laufunruhe oder ein Blockieren der Achse ermittelt werden.

Der Lage- und Beschleunigungssensor misst die Lage des Testroboters im dreidimensionalen Raum. D.h. der Sensor liefert Werte über die Neigung in X-, Y- und Z-Richtung zurück. Zudem gibt er einen Wert über die Beschleunigung des Testroboters jeweils in X-, Y- und Z-Richtung bis zwei g zurück.

Der Kompasssensor gibt Auskunft über die Himmelsrichtung, in die sich der Testroboter bewegt. Das Erdmagnetfeld kann vom Sensor bis zu einem Grad genau gemessen werden. Elektromagnetische Störungen werden automatisch kompensiert. Da der Kompasssensor zudem digital ist und keine mechanischen Elemente besitzt, kann die Messung der Himmelsrichtung auch nicht durch physische Einwirkungen wie z.B. durch Stöße oder Lageänderungen beeinflusst werden.

Der Ultraschallsensor sendet einen hochfrequenten Ton. Wird dieser Ton von einem Objekt vor dem Sensor zurückgeworfen, misst der Sensor die Zeit, die vom Aussenrand des Tones bis zum Empfangen benötigt wurde. Es wird der vom Ton zurückgelegte Weg zwischen Sensor und Objekt gemessen, wodurch es also möglich ist, Ob-

jekte und die Entfernung zu diesen zu ermitteln. Der Ultraschallsensor ist in der Front des Testroboters eingebaut und in Fahrtrichtung ausgerichtet.

Der aktive Lichtsensor emittiert ein rotes Licht und misst Veränderungen der Helligkeit und Intensität des Spektrums innerhalb des emittierten Lichts. Dadurch ist es möglich, sowohl Auskunft über die Helligkeit als auch über die Farbe einer Oberfläche zu geben. Der Lichtsensor wird an der Front des Testroboters eingebaut und auf den Boden ausgerichtet.

Die folgende Abbildung 10 auf der nächsten Seite zeigt den konstruierten Testroboter, Arbeitstitel *MOUSE* (**M**echanical **O**perational **U**nit **S**killed in **E**xploration).

# MOUSE

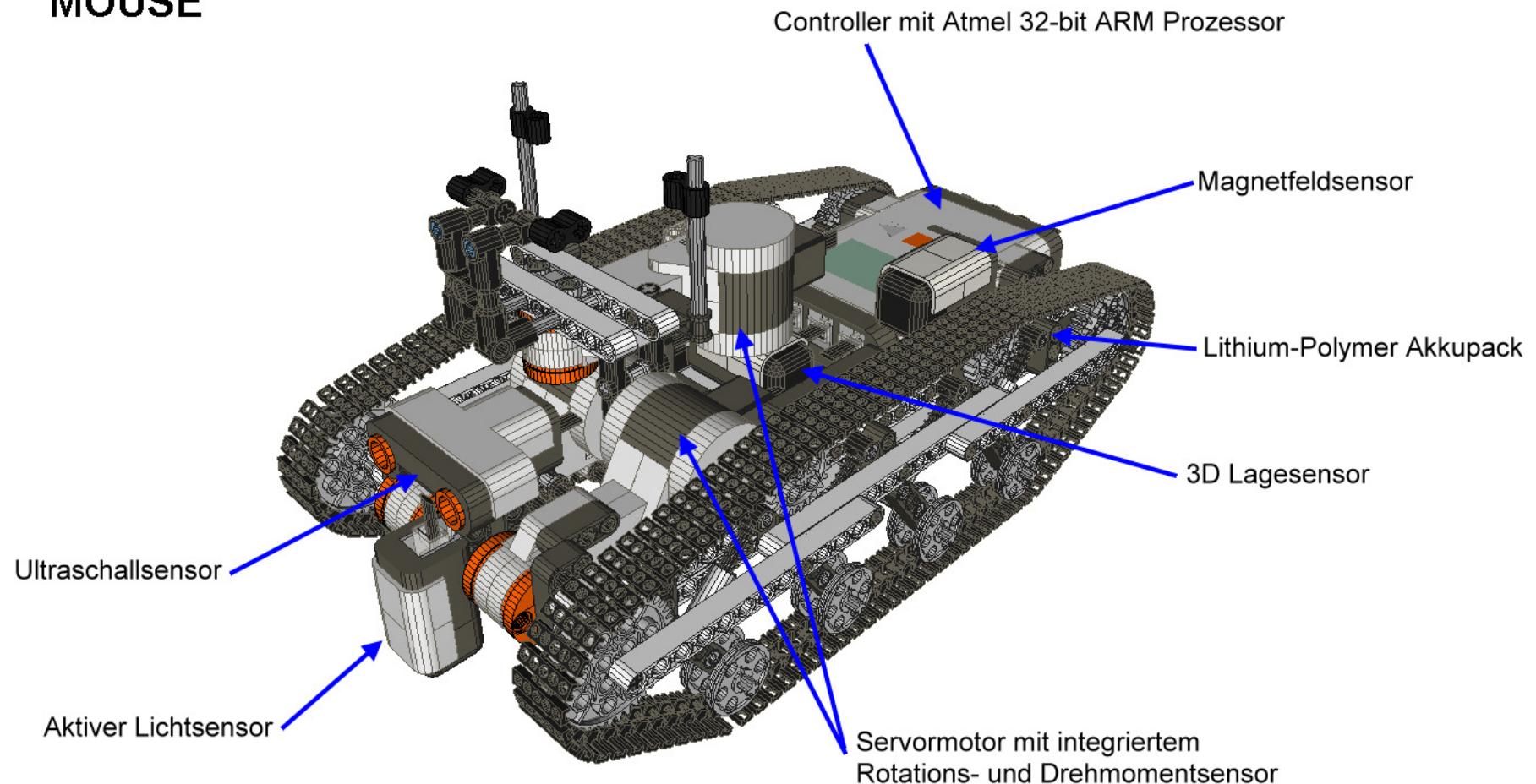


Abbildung 10: Der NXT Testroboter; Arbeitstitel **MOUSE** (Mechanical Operational Unit Skilled in Exploration).

## **4.2 Der Server Prototyp**

Als Server Prototyp soll das Smartphone SGH-i600 der Firma Samsung zum Einsatz kommen. Dieses Smartphone verwendet als Betriebssystem Windows Mobile 6.0 (Windows CE 5.2.1437). Als Prozessor ist ein Texas Instruments OMAP 1710 (Produktnummer ARM926TEJ) mit einer Taktrate von 220 MHz verbaut. Das Smartphone verfügt über einen 128 MB RAM Speicherbaustein. Dieser wird vom Betriebssystem sowohl als Hauptspeicher als auch als Arbeitsspeicher verwendet. Die Aufteilung des Speichers erfolgt je nach Bedarf bzw. Systemauslastung automatisch. In der Regel kann von 64 MB Hauptspeicher und 64 MB Arbeitsspeicher ausgegangen werden. Zudem ist in dem Samsung SGH-i600 ein 64 MB ROM Speicherbaustein verbaut, welcher das Betriebssystem beinhaltet. Dieser Speicher kann jedoch auch zum Abspeichern eigener Daten verwendet werden. Als Schnittstellen bietet das Smartphone Bluetooth, Wireless LAN sowie eine kabelgebundene Lösung. Allerdings können die Wireless LAN- und Bluetooth-Schnittstelle nicht parallel betrieben werden. Ferner sind ein GSM- und ein UMTS-Modul integriert, welche auf den Frequenzbändern 900, 1800 und 19000 MHz senden und empfangen können. Das Samsung SGH-i600 erfüllt somit die in 4.2.2 beschriebenen Systemanforderungen für die Server-Software und eignet sich daher zum Testen des Software-Prototypen.

Der Prototyp der Server-Software (Arbeitstitel *Robocontrol-Server*) soll Steuerbefehle der Client-Software entgegennehmen und an den Mindstorms NXT Mikrocontroller mittels der im Samsung SGH-i600 integrierten Bluetooth-Schnittstelle gesendet. Sensorinformationen des Mindstorms NXT Mikrocontrollers sollen wiederum vom Samsung SGH-i600 ausgelesen und an die Client-Software übertragen werden. Da auf dem Smartphone die Wireless LAN- und Bluetooth-Schnittstelle nicht parallel betrieben werden können, die Bluetooth-Schnittstelle jedoch für die Verbindung mit dem Mindstorms NXT zwingend erforderlich ist, muss bei der Verbindung zwischen Samsung SGH-i600 und Client auf eine kabelgebundene LAN-Verbindung zum Durchführen der Tests zurückgegriffen werden.

### 4.3 Der Client Prototyp

Zum Testen des Prototypen der Client-Software (Arbeitstitel *RoboControl-Client*) soll ein Samsung X20 XVM 1730 Notebook zum Einsatz kommen. Es verfügt über einen Intel Pentium M Prozessor mit einer Taktrate von 1,73 GHz. Das Notebook besitzt eine 100 GB große Festplatte und 1 GB Arbeitsspeicher. Zudem verfügt es über geeignete Schnittstellen wie USB, LAN und Wireless LAN. Als Betriebssystem wird Windows XP SP3 verwendet. Das Samsung X20 XVM 1730 Notebook erfüllt somit die in 4.1.1 beschriebenen Systemanforderungen und ist daher zum Testen des Software-Prototypen geeignet.

Zum Durchführen der Tests wird das Notebook, wie in 5.2 bereits beschrieben, mit dem Samsung SGH-i600 über eine kabelgebundene LAN-Schnittstelle verbunden. Der Prototyp der Client-Software soll eingegebene Steuerbefehle über diese Schnittstelle an die Server-Software senden, aber auch zur Visualisierung der vom Testroboter versendeten Informationen dienen. Hierzu zählen die Daten, die von den im Testroboter verbauten Sensoren geliefert werden. Abbildung 11 zeigt das Hauptprogramm der Client-Software.

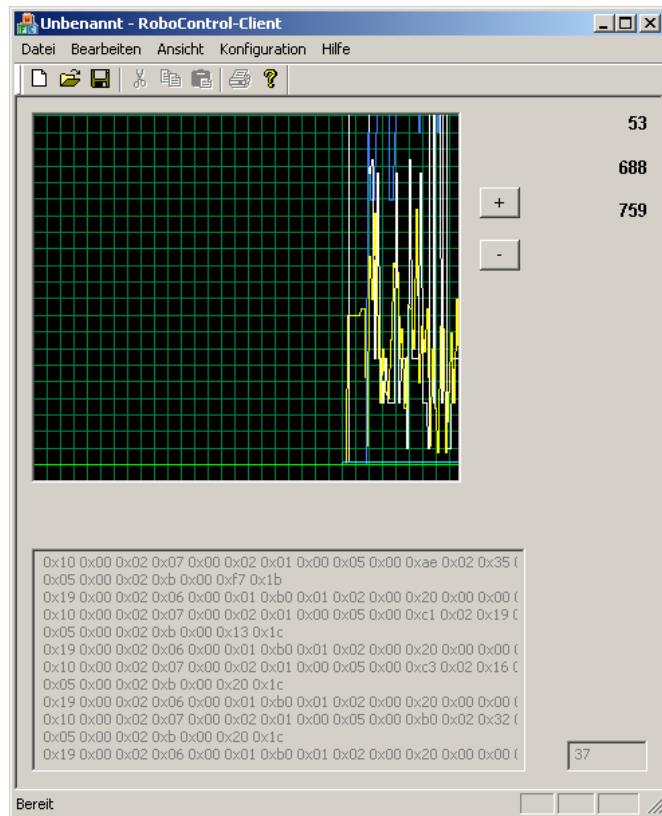


Abbildung 11: Client-Prototyp *RoboControl*.

## 5 Testszenarien und Bewertung der Ergebnisse

Um die Funktionalität des Software-Prototypen zu testen, müssen geeignete Testbedingungen erstellt werden und eine Umgebung für den Testroboter zur Durchführung der Tests aufgebaut werden. Folgend werden zwei Szenarien vorgestellt, welche zum Testen des Software-Prototypen verwendet wurden.

### 5.1 Der Minensuchroboter

Bei dem Minensuchroboter-Szenario soll der Testroboter über die Client-Software durch ein simuliertes Minenfeld gesteuert werden. Stößt der Testroboter dabei auf eine Mine, so soll dies durch die Sensorik des Testroboters in der Client-Software angezeigt werden. Dieses Testszenario soll dazu dienen, sowohl das Versenden von Steuerbefehlen als auch das Empfangen und Auswerten von Sensordaten zu testen. Für die Durchführung des Tests wurde ein Parcours aufgebaut. Dieser besteht aus einer weißen Grundfläche, auf der in unregelmäßig Abständen zehn cm große, rote Scheiben verteilt wurden. Diese roten Scheiben sollen die Minen simulieren. Das Erfassen der Minen soll durch den Lichtsensor des Testroboters geschehen, in dem die Veränderungen des Farbspektrums von weiß nach rot gemessen und angezeigt werden. Unter realen Bedingungen könnte dies z.B. die Veränderung des magnetischen Feldes eines Metalldetektionssensors beim Aufspüren einer echten Mine oder eines Sprengsatzes sein.

Neben dem Aufbau des Parcours muss zur Testvorbereitung die Client-Software mit den benötigten Steuerbefehlen einmalig konfiguriert werden. Damit der Benutzer den Testroboter durch das Minenfeld steuern kann, sind folgende Steuerbefehle nötig:

Richtungsänderung vorwärts:

0x0C 0x00 0x80 0x04 0x01 0x64 0x01 0x02 0x00 0x20 0x00 0x00 0x00 0x00  
Ausgang Spannung \_\_\_\_\_  
Steuerbefehl für Servomotor 1

0x0C 0x00 0x80 0x04 0x02 0x64 0x01 0x02 0x00 0x20 0x00 0x00 0x00 0x00  
Ausgang Spannung \_\_\_\_\_  
Steuerbefehl für Servomotor 2

Dieser Steuerbefehl setzt sich aus zwei Befehlen zusammen. Er schaltet die beiden Servomotoren des Kettenlaufwerks in eine synchrone Drehrichtung und setzt sie unter volle Spannung.

Die Drehrichtung und Geschwindigkeit der Achsen wird durch die anzulegende Spannungsleistung gesteuert. Der Wertebereich kann dabei von –100 (hexadezimal 0x9C) bis 100 (hexadezimal 0x64) reichen. Der Wert –100 bedeutet in der Praxis für den Servomotor eine Drehrichtung mit Höchstgeschwindigkeit entgegengesetzt des Uhrzeigersinns, der Wert 100 hingegen eine Drehrichtung mit Höchstgeschwindigkeit im Uhrzeigersinn.

Dadurch bewegt sich bei dem oben beschriebenen Steuerbefehl mit einem Wert von 100 der Testroboter vorwärts.

Richtungsänderung rückwärts:

0x0C 0x00 0x80 0x04 0x01 0x9C 0x01 0x02 0x00 0x20 0x00 0x00 0x00 0x00  
Ausgang Spannung

---

Steuerbefehl für Servomotor 1

0x0C 0x00 0x80 0x04 0x02 0x9C 0x01 0x02 0x00 0x20 0x00 0x00 0x00 0x00  
Ausgang Spannung

---

Steuerbefehl für Servomotor 2

Dieser Steuerbefehl ist ähnlich aufgebaut wie der Befehl für die Vorwärtsbewegung. Jedoch wird die Spannungsleistung mit –100 (hexadezimal 0x9C) umgekehrt und dadurch die Drehrichtung der Servomotoren des Kettenlaufwerks geändert. Der Testroboter bewegt sich rückwärts.

Richtungsänderung links:

0x0C 0x00 0x80 0x04 0x01 0xB0 0x01 0x02 0x00 0x20 0x00 0x00 0x00 0x00  
Ausgang Spannung

---

Steuerbefehl für Servomotor 1

0x0C 0x00 0x80 0x04 0x02 0x50 0x01 0x02 0x00 0x20 0x00 0x00 0x00 0x00  
Ausgang Spannung

---

Steuerbefehl für Servomotor 2

Richtungsänderung rechts :

0x0C 0x00 0x80 0x04 0x01 0x50 0x01 0x02 0x00 0x20 0x00 0x00 0x00 0x00  
Ausgang Spannung  
Steuerbefehl für Servomotor 1

0x0C 0x00 0x80 0x04 0x02 0xB0 0x01 0x02 0x00 0x20 0x00 0x00 0x00 0x00  
Ausgang Spannung  
Steuerbefehl für Servomotor 2

Um die Richtung des Testroboters ändern zu können, ist es nötig, dass sich die Servomotoren des Kettenlaufwerks asynchron bewegen. D.h. die Antriebsachsen müssen sich gegenläufig bewegen. So wird an dem einen Servomotor eine Spannungsleistung von 80 (hexadezimal 0x50) und an dem anderen eine Spannungsleistung von -80 (hexadezimal 0xB0) angelegt. Dadurch dreht sich der Testroboter links bzw. rechts herum. Es wird hierbei keine volle Spannungsleistung angelegt, da sich bei den Tests gezeigt hat, dass der Testroboter bei einer höheren Spannungsleistung eine unkontrollierbare Drehgeschwindigkeit erreichen würde und damit eine exakte Richtungsänderung erschwert wird.

Bewegung anhalten:

0x0C 0x00 0x80 0x04 0x01 0x00 0x01 0x02 0x00 0x20 0x00 0x00 0x00 0x00  
Ausgang Spannung  
Steuerbefehl für Servomotor 1

0x0C 0x00 0x80 0x04 0x02 0x00 0x01 0x02 0x00 0x20 0x00 0x00 0x00 0x00  
Ausgang Spannung  
Steuerbefehl für Servomotor 2

Um den Testroboter anzuhalten, wird durch die Steuerbefehle die an die Servomotoren angelegte Spannungsleistung einfach auf Null gesetzt, wodurch die Servomotoren stoppen. Den obigen Steuerbefehlen müssen in der Client-Software entsprechende Aktionen zugewiesen werden, bei denen die Steuerbefehle an den Server gesendet werden sollen. Für die Durchführung des Testszenarios werden den Steuerbefehlen Tasten auf der Tastatur als auszulösende Aktion zugewiesen.

Die Minen auf dem Parcours sollen durch den Lichtsensor des Testroboters aufgespürt werden, in dem die Änderungen des Farbspektrums zwischen der weißen Unterlage des Parcours und den roten Minen in der Client-Software angezeigt werden. D.h. die Client-Software muss so konfiguriert werden, dass die vom Server gesendeten Antworten des Mikrocontrollers auf die Sensordaten des Lichtsensors hin gefiltert werden.

Beispielantwort des Servers:

0x10	0x00	0x02	0x07	0x00	0x02	0x01	0x00	0x05	0x00	0xbd	0x02	0x1f	0x01	0x1f	0x01
				Opcode für gelesene Datenwerte von Eingang Nr. 3								Intensität gemessen vom Lichtsensor			

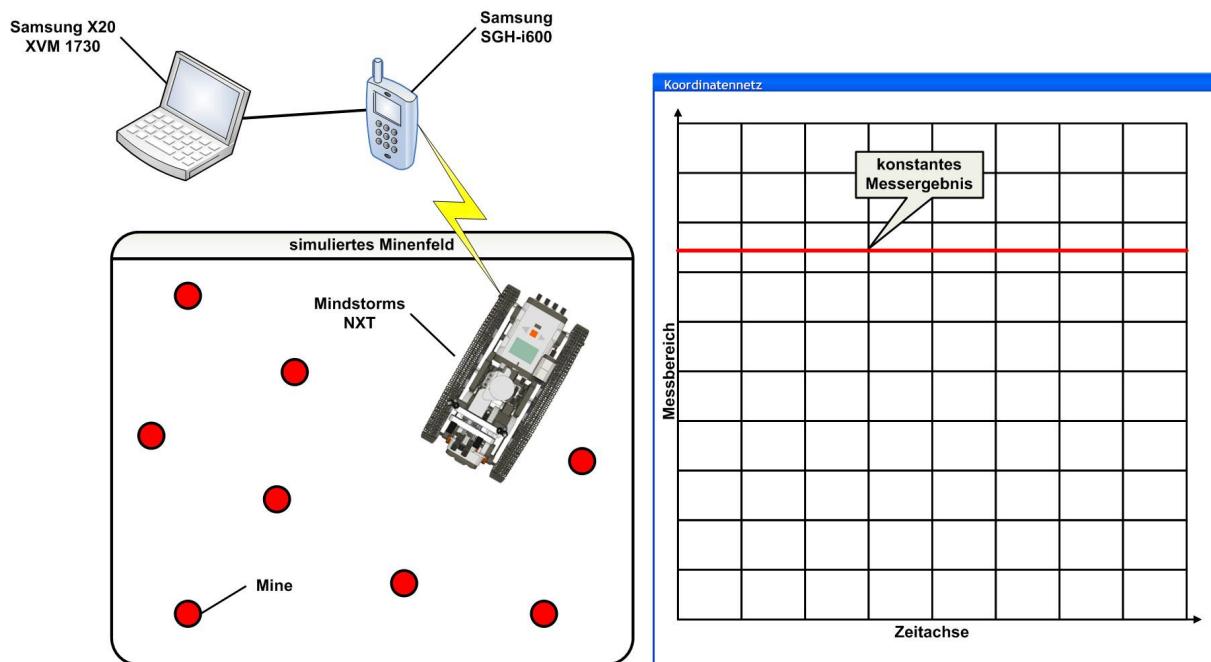
Der Lichtsensor ist am Eingang Nr. 3 des Mindstorms NXT angeschlossen. Der Opcode für eine Antwort, die die eingelesenen Datenwerte des Eingangs Nr. 3 zurückliefert, lautet beim Mindstorms NXT Mikrocontroller 0x07. Also muss die Client-Software die Antworten vom Server nach dem Opcode 0x07 filtern. Der für das Testszenario benötigte Wert befindet sich bei dem Antwort-Bytecode an fünfzehnter Stelle. Dies muss in der Client-Software ebenfalls konfiguriert werden. Es muss also Opcode 0x07 als zu filternde Antwort und die fünfzehnte Stelle des Bytecodes als zugehöriger zu repräsentierender Wert eingestellt werden. Zudem muss ausgewählt werden, wie dieser Wert in der Client-Software repräsentiert werden soll. Für die Durchführung des Testszenarios soll der aktuelle Wert als Liniendiagramm angezeigt werden.

Neben der Konfiguration der Client-Software muss ebenfalls die Server-Software konfiguriert werden. Es muss eine Bluetooth-Verbindung zwischen dem Mindstorms NXT und dem Server, also dem Samsung SGH-i600, hergestellt werden. Ferner müssen in der Server-Software die Verbindungsparameter für die Bluetooth-Schnittstelle einmalig konfiguriert werden. Die Baudrate wird auf 460800 und die Bytegröße auf acht gesetzt. Die Anzahl der Stoppbits beträgt eins und eine Parität wird nicht benötigt.

Das Testszenario wird wie folgt durchgeführt:

Der Testroboter wird nach der Konfiguration der Client- und Server-Software in den Parcours gesetzt. Eine Verbindung wird zwischen dem Client und dem Server über

eine IP Adresse hergestellt. Nach einem erfolgreichen Verbindungsaufbau wird in der Client-Software ein zweidimensionales Koordinatennetz angezeigt. In diesem Koordinatennetz ist während des Ruhezustands des Testroboters zu beobachten, wie sich eine horizontale Linie im höheren Messbereich des Koordinatennetzes aufbaut (siehe Abb. 12). Dies ist dadurch zu erklären, dass der Lichtsensor des Testroboters sich über der weißen Unterlage des Parcours befindet und sich der Testroboter nicht bewegt. Die weiße Oberfläche liefert einen hohen Helligkeits-Intensitätswert.



**Abbildung 12: Versuchsaufbau und Messergebnis; der Testroboter wird nicht bewegt.**

Der Testroboter wird nun über den Parcours gesteuert. Es ist zu beobachten, dass die Linie im Koordinatensystem nun leichten Schwankungen im Messbereich unterliegt (siehe Abb. 13). Dies sind geringfügige Fluktuationen der Helligkeits-Intensität, die durch die Bewegung des Testroboters entstehen. So kann durch die Bewegungen wie z.B. Beschleunigung und Richtungsänderungen der Abstand zwischen Lichtsensor und Parcours-Unterlage etwas schwanken und somit zu leicht veränderten Messergebnissen führen. Zudem wirft der Testroboter je nach Ausrichtung einen Schatten auf die zu untersuchende Oberfläche. Da dies allerdings nur geringe Fluktuationen im Wertebereich sind, ist eine Kompensation nicht nötig.

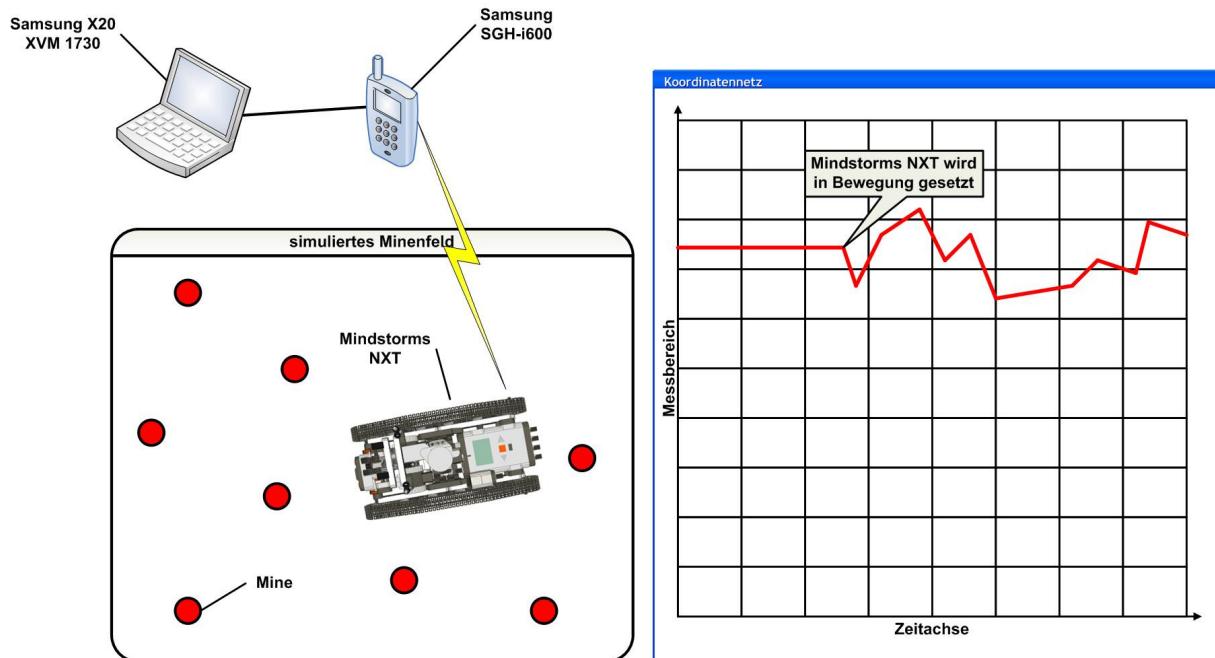


Abbildung 13: Der Testroboter wird in Bewegung gesetzt.

Fährt der Testroboter nun über eine Mine, ist ein signifikanter, negativer Ausschlag der Linie im Messbereich zu beobachten (siehe Abb. 14). Dieser Ausschlag ist durch eine starke Veränderung in der vom Lichtsensor gemessenen Helligkeits-Intensität zu erklären. Wird eine der roten Testminen überfahren, ändert sich die Oberflächenbeschaffenheit durch den Farbwechsel von weiß auf rot. Dies wird vom Lichtsensor erfasst und von der Client-Software korrekt repräsentiert.

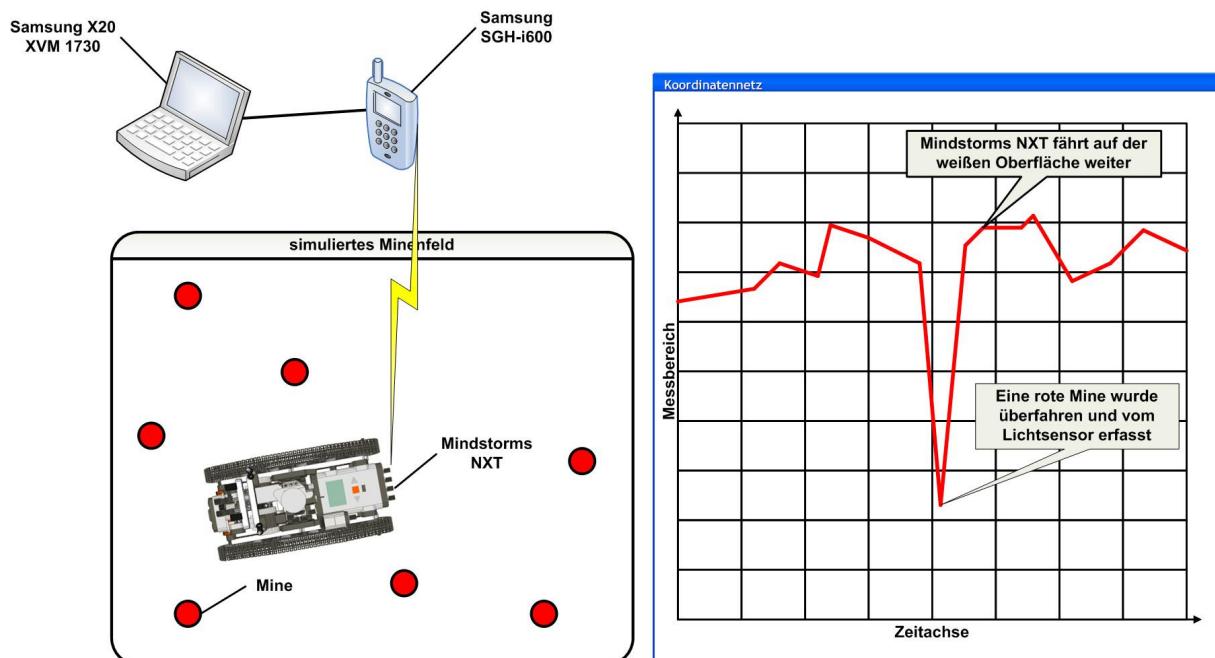


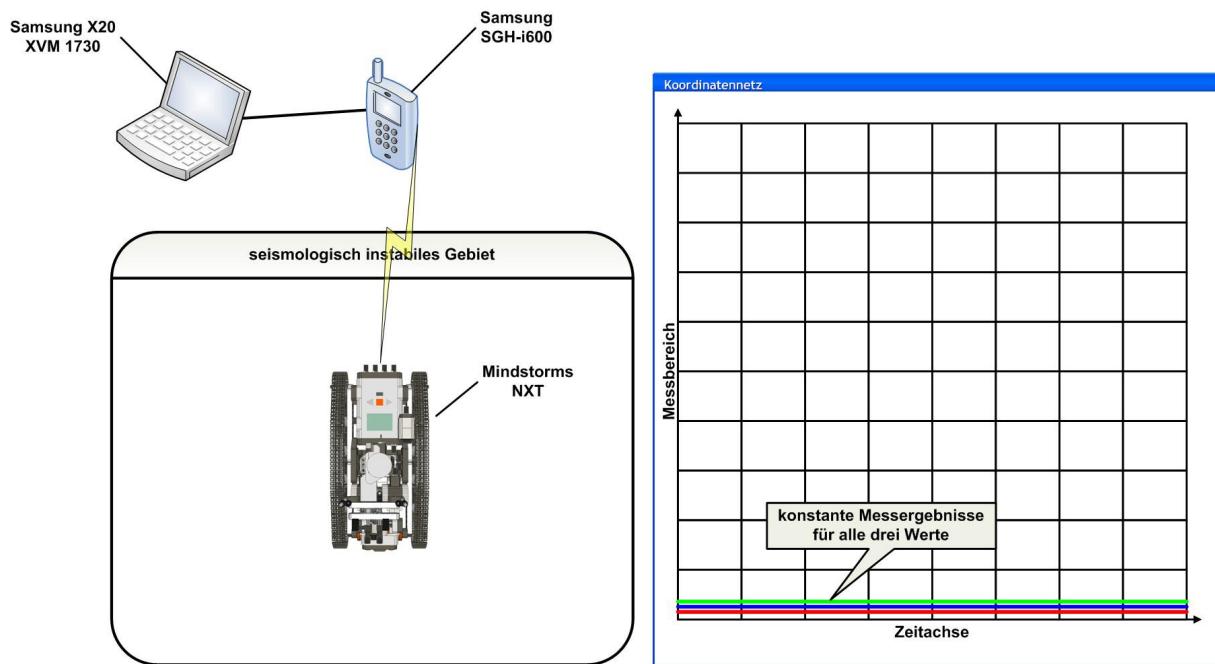
Abbildung 14: Der Testroboter überfährt eine Mine.

Da der Testroboter sich weiterbewegt und somit der Kontakt mit der roten Testmine nur kurz ist, wird von der Linie im Koordinatensystem auch nur ein kurzer Ausschlag beschrieben. Jedes mal, wenn die Linie einen signifikanten negativen Ausschlag im Messbereich aufzeichnet, wurde also eine Mine gefunden.

## **5.2 Die seismologische Mess-Station**

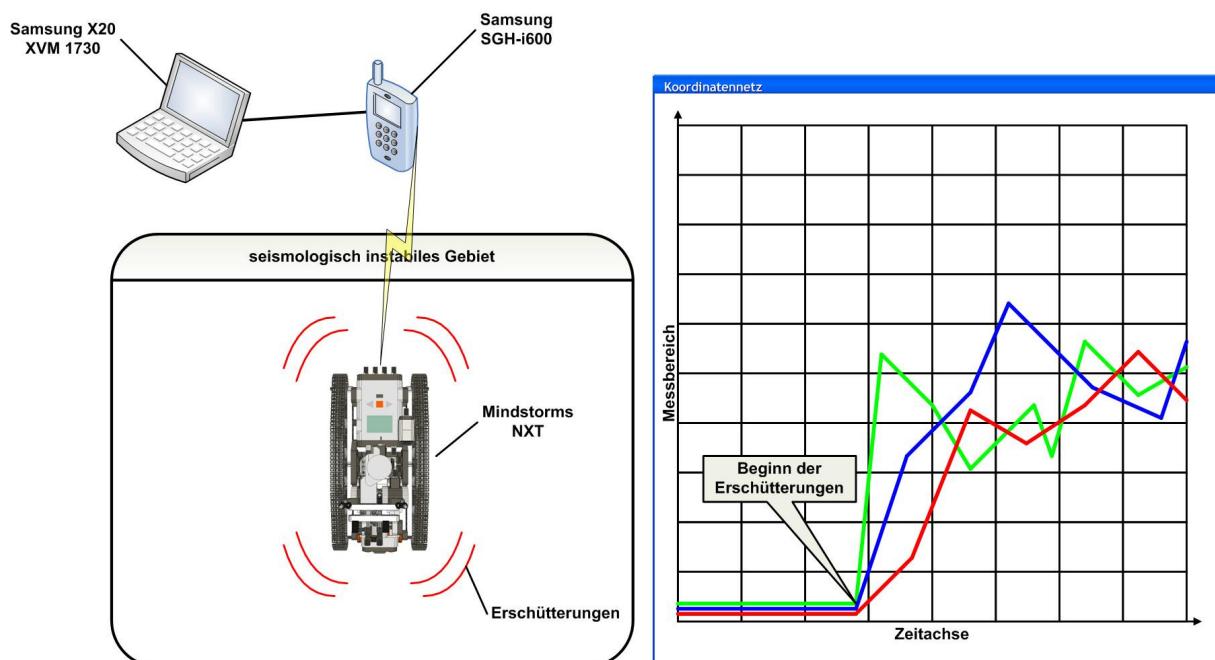
Dieses Testszenario soll die Repräsentation mehrerer Sensordaten in der Client-Software prüfen. Hierzu wird mit dem Testroboter eine stationäre, seismologische Mess-Station simuliert, mit der Erschütterungen erfasst und in der Client-Software repräsentiert werden sollen. Zur Durchführung dieses Tests ist es nicht nötig, Steuerbefehle in der Client-Software zur Steuerung des Testroboters zu hinterlegen, da bei dem Test die mechanischen Komponenten des Testroboters nicht benötigt werden. Es muss hingegen in der Client-Software konfiguriert werden, welche Server-Antworten gefiltert und wie diese repräsentiert werden sollen. Zur Simulation einer seismologischen Messung soll der Lagesensor des Testroboters verwendet werden. Es müssen die Werte der Beschleunigung jeweils in X-, Y- und Z-Richtung ausgelesen und repräsentiert werden. Da jeder Wert separat repräsentiert werden soll, muss für jeden Wert auch ein separater Eintrag in der Konfiguration der Client-Software vorgenommen werden. Die zu filternden Opcodes sind zwar bei allen drei Werten identisch, jedoch ist die Position des jeweiligen Wertes innerhalb des Antwort-Bytecodes unterschiedlich. Das Verfahren zur Konfiguration gleicht dem aus 6.1, nur dass nun drei Sensordaten aus den Server-Antworten separat gefiltert werden. Als Repräsentationsform wird wieder ein Liniendiagramm gewählt.

Der Testroboter wird nun auf eine Tischplatte gestellt und eine Verbindung zwischen Client und Server hergestellt. Die Tischplatte soll ein seismologisch instabiles Gebiet simulieren. In der Client-Software sind nun im Koordinatensystem drei horizontal verlaufende Linien zu beobachten, die einen absolut geradlinigen Verlauf im unteren Wertebereich aufzeichnen (siehe Abb. 15). Da sich der Testroboter nicht bewegt und der Lagesensor somit auch keiner Beschleunigung unterliegt, verändern sich auch nicht die gemessenen Werte.



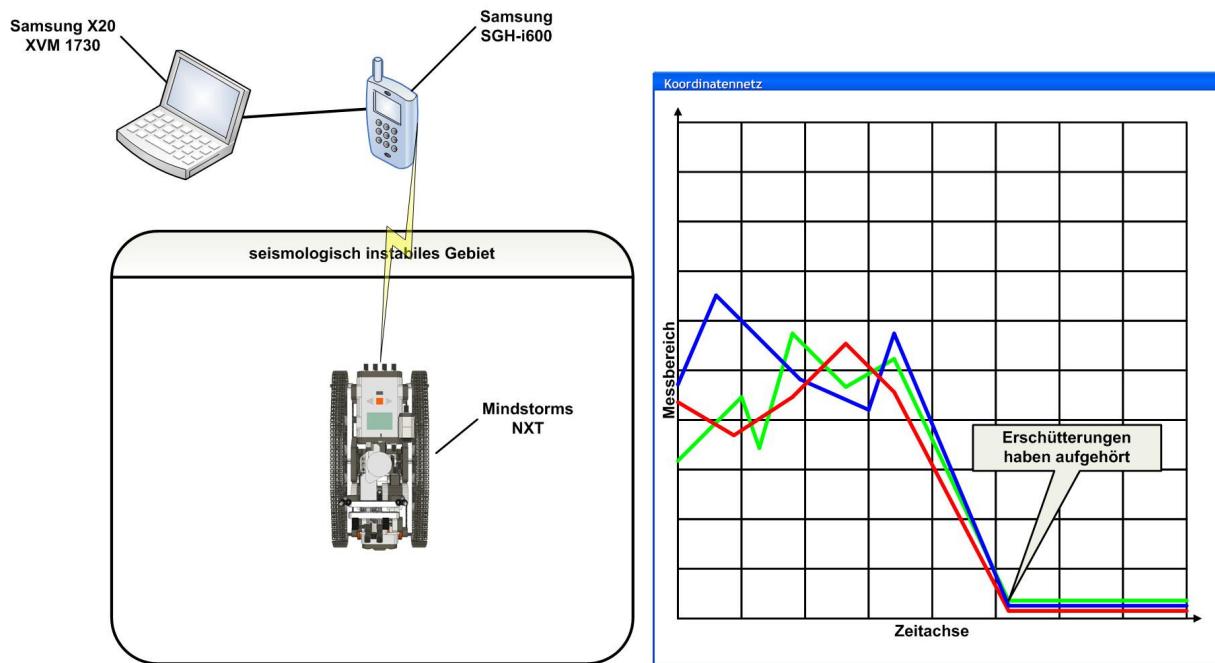
**Abbildung 15: Versuchsaufbau und Messergebnisse; der Testroboter wird nicht bewegt.**

Die Tischplatte wird nun durch kräftige Stöße über einen Zeitraum von 15 Sekunden in Erschütterung versetzt. Hierdurch soll eine seismologische Instabilität simuliert werden. Es ist zu beobachten, wie sich der Linienvorlauf der drei Linien in einen höheren Wertebereich mit unterschiedlichen Maxima verschiebt, dann wieder abfällt und erneut ansteigt (siehe Abb. 16).



**Abbildung 16: Der Testroboter wird Erschütterungen ausgesetzt.**

Hören die Tisch-Stöße nach den 15 Sekunden auf, fallen die Linien wieder in den unteren Wertebereich ab und der Linienverlauf bleibt horizontal (siehe Abb. 17). Dieser Linienverlauf ist dadurch zu erklären, dass der Lagesensor während der Erschütterungen ständigen Beschleunigungen in X-, Y- und Z-Richtungen unterliegt. Diese Änderungen werden korrekt im Koordinatensystem durch den Linienverlauf repräsentiert.



**Abbildung 17:** Die Erschütterungen werden eingestellt.

Im Umkehrschluss kann also gesagt werden, wenn der Linienverlauf in den höheren Wertebereich ausschlägt, wird der Untergrund des Testroboters erschüttet. Die Höhe der Ausschläge gibt die Intensität eines solch simulierten Erdbebens wieder.

## 6 Fazit und Ausblick

Die Prototypen der Client- und Server-Software funktionieren gemäß den in Kapitel 3 definierten Anforderungen. Mit Hilfe des Testsystems konnte gezeigt werden, dass es möglich ist, beliebige Steuerbefehle je nach Anwendungsszenario zu verwalten und an frei konfigurierbare Mikrocontroller zu senden. Das Auslesen und Auswerten von beliebigen Datenwerten ist ebenfalls möglich. Auf den Einsatz von spezifischen Treibern zur Ansteuerung des Test-Mikrocontrollers konnte bei dem entwickelten Konzept verzichtet werden.

### 6.1 Vor- und Nachteile des entwickelten Systems

Im Folgenden soll das Opcode-basierte System unter Verwendung des ISO/IEC 9126 Standards [Rupp 2007] aus Sicht eines Benutzers bewertet und mit den drei Systemen aus Kapitel 3 verglichen werden. Hierzu werden wie in Kapitel 3.4 die Qualitätsmerkmale Funktionalität, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit für das System betrachtet.

	Kommunikation über Gerätetreiber	Microsoft Robotics Studio / Modellsprache	hardwarenahe Plattformen	Opcode-basiertes System
Funktionalität	++	+	++	+
Benutzbarkeit	n/a	++	n/a	0
Effizienz	++	0	+	++
Änderbarkeit	-	+	-	++
Übertragbarkeit	-	0	-	+

Tabelle 2: Bewertung und Gegenüberstellung der vorgestellten Systeme

- ++ sehr gut
- + gut
- 0 mittelmäßig
- schlecht
- sehr schlecht

Das Opcode-basierte System bietet ein hohes Maß an Funktionalität. Mit diesem System können in der Theorie alle Funktionen eines Mikrocontrollers unterstützt werden, vorausgesetzt, es existiert ein Steuerbefehl für die Funktionen. Allerdings besteht bei dem Opcode-basierten System das Problem, dass eingegebene Steuerbefehle nicht auf Richtigkeit geprüft werden können. So kann bei der Ausführung die Richtigkeit der Funktionen nicht garantiert werden.

Benutzbarkeit ist bei dem Opcode-basierten System nur mittelmäßig. Der Benutzer muss Fachwissen über die Funktionsweise von Steuerbefehlen besitzen. Bei den entwickelten Software-Prototypen müssen die Steuerbefehle einzeln zusammenge stellt und in das System eingespeist werden. Allerdings ist das Hinterlegen von Steuerbefehlen nur einmalig nötig, so lange das zu steuernde System nicht verändert wird. Zudem wird vorausgesetzt, dass der Benutzer selbstständig die eingegebenen Steuerbefehle auf Korrektheit überprüft. Theoretisch wird jeder Steuerbefehl vom Mikrocontroller verarbeitet, auch wenn der Steuerbefehl fehlerhaft ist. Es gibt keine Fehlertoleranz.

Da Steuerbefehle im Bytecode-Format direkt versendet werden, die Größe der Steuerbefehle somit sehr gering ist und auf zusätzliche Dienste wie Treiber verzichtet wird, ist die Effizienz sehr hoch. Die Anforderungen an Speicher und CPU sind bei diesem Verfahren im Vergleich zu den anderen Systemen gering.

Änderbarkeit ist im Vergleich bei dem Opcode-basierten System sehr gut. Der Benutzer benötigt keine Programmierkenntnisse. Änderungen beschränken sich auf das Anpassen von gespeicherten Steuerbefehlen oder das Hinzufügen von neuen Steuerbefehlen. So ist auch das Testen dieser Änderungen nur mit einem geringen Aufwand verbunden. Änderungen können sofort angewandt werden. Es ist nicht wie bei den anderen Systemen nötig, den Programmcode neu zu kompilieren. Dadurch können auch die Kosten bei System-Integration und -Tests reduziert werden.

Das Opcode-basierte System bietet eine gute Übertragbarkeit. Durch das direkte Versenden von Maschinenbefehlen und die daraus resultierende Treiberunabhängigkeit wird eine hohe Portierbarkeit auf unterschiedliche Robotersysteme erreicht. Voraussetzung ist jedoch, dass die anzusteuernden Mikrocontroller eine direkte Verarbeitung von Steuerbefehlen unterstützen und der Benutzer die Steuerbefehle von dem Zielsystem kennt. Der Einsatz von Windows für den Software-Prototypen beschränkt die Möglichkeit einer Portierung auf andere Betriebssystem-Plattformen.

Jedoch kann das vorgestellte Verfahren auch für andere Betriebssystem-Plattformen entwickelt werden.

Vor und Nachteile auf einen Blick:

- + Hohe Portierbarkeit und Modularität.
- + Kostenreduktion bei der Entwicklung und Integration neuer Systeme.
- + Einfache Adaptierbarkeit.
- + Geringer Ressourcenbedarf.
- Keine Fehlertoleranz bei der Eingabe von Steuerbefehlen.
- Hoher Konfigurationsaufwand für den Benutzer.

## **6.2 Mögliche Verbesserungen**

Um bei dem Opcode-basierten System den hohen Konfigurationsaufwand verringern zu können, müssen das Erstellen und Verwalten der Steuerbefehle für den Benutzer vereinfacht werden. So wäre eine Datenbank denkbar, in der ein Datenpool an Steuerbefehlen gängiger Mikrocontroller angelegt und dem Benutzer zur Verfügung gestellt wird. Allerdings wird es nicht möglich sein, alle theoretisch gültigen Steuerbefehle als Ganzes in einer solchen Datenbank zu hinterlegen. Dies würde eine zu hohe Anzahl an Datensätzen und somit einen zu hohen Verwaltungsaufwand bedeuten. So kann sich ein Steuerbefehlssatz aus verschiedenen Komponenten wie Header- und Opcode-Informationen sowie Daten- und Befehlsparametern zusammensetzen. Die Daten- und Befehlsparameter wiederum können sich je nach benötigter Funktion aus einer Vielzahl an Werten unterschiedlich zulässiger Wertebereiche zusammensetzen. Wenn nun ein Opcode 16 Datenbytes als Parameter hat, von denen jeder einen möglichen Wertebereich von 0 bis 15 besitzt, so wären theoretisch 18446744073709551616 Zustände für die Parameter dieses einen Opcodes möglich. Das bedeutet, dass für diesen einzigen Opcode in der Theorie 18446744073709551616 Steuerbefehle denkbar wären.

Daher sollte die Datenbank so konzipiert werden, dass ein Datensatz für jeden Opcode angelegt wird, der von einem spezifischen Mikrocontroller unterstützt wird. Allerdings wird kein ganzer Steuerbefehl gespeichert. Stattdessen wäre es denkbar, zusätzlich für jeden Opcode eines Mikrocontrollers ein Regelkatalog für die Zusam-

mensetzung der Steuerbefehle in der Datenbank zu hinterlegen. Dazu zählen das Format des Headers sowie das Format der Parameter wie Byteanzahl und gültiger Wertebereich. Für jeden Mikrocontroller sollte also eine Opcode-Liste und ein Regelkatalog in der Datenbank hinterlegt werden. Um die Benutzbarkeit weiter zu verbessern, sollte zudem eine Beschreibung für den jeweiligen Opcode in der Datenbank gespeichert werden. Durch eine solche Datenbank und einer entsprechenden grafischen Oberfläche könnte der Benutzer sich die für seine Zwecke benötigten Opcodes heraussuchen. Mittels des Regelkatalogs für den Opcode könnten nun automatisch die Header-Informationen ergänzt werden. Zudem könnte der Benutzer automatisch nach den benötigten Parametern abgefragt werden. Der zulässige Wertebereich für die einzelnen Parameter kann während der Eingabe durch den Regelkatalog auf Gültigkeit überprüft werden. Dadurch ließe sich auch die Fehlertoleranz wesentlich verbessern, weil ein falsch zusammengestellter Steuerbefehl auf diese Weise nicht mehr möglich ist. Eine Drag&Drop Funktion für das Zusammenstellen der Steuerbefehle wäre als zusätzliche Verbesserung der Bedienbarkeit denkbar.

Eine weitere denkbare Verbesserung wäre das Hinzufügen einer Makrofunktion. Dadurch wäre es möglich, je nach Mächtigkeit der Makros, bestimmte Ereignisgesteuerte Prozesse zu erstellen und damit Abläufe zu automatisieren. So könnte der Benutzer z.B. ein Makro erstellen, das beim Erreichen eines definierten Sensorwertes einen festgelegten Steuerbefehl sendet. Dadurch wäre eine rudimentäre, automatische Steuerung möglich.

Weitere Verbesserungen sind bei der grafischen Repräsentation von Sensor- und Datenwerten möglich. So gibt es eine Vielzahl an Repräsentationsmöglichkeiten, die sich je nach Einsatzgebiet bzw. Sensorik ändern können. Dies steht auch in Abhängigkeit des zu repräsentierenden Wertes. So kann dies z.B. nur eine dezimale Zahl, aber auch eine Vector-Matrix sein, welche der Benutzer als dreidimensionales Netzdiagramm dargestellt haben möchte. Eine Vector-Matrix kann aber auch Bildinformationen enthalten, die von der Client-Software repräsentiert werden müssen. Als Verbesserung kann die Client-Software um Repräsentationsformen, die unterstützt werden sollen, stetig ergänzt werden.

Eine zusätzliche Verbesserung wäre in diesem Zusammenhang eine Speicher- und Ladefunktion für die empfangenen Sensor- bzw. Datenwerte. Dadurch wäre der Benutzer in der Lage, die aufgezeichneten Werte zu einem späteren Zeitpunkt erneut in

der Client-Software zu betrachten. Eine Exportfunktion, z.B. nach Excel, wäre hierzu eine Ergänzung.

Abbildung 18 zeigt eine mögliche Weiterentwicklung der grafischen Oberfläche der Client-Software. Diese kann z.B. dreidimensionale Gitterdiagramme und Bildinformationen darstellen. Es handelt sich hierbei um ein Multiple Document Interface (MDI). Jeder Repräsentationsart, wie z.B. dem dreidimensionalem Gitterdiagramm, kann der Benutzer ein eigenständiges Fenster innerhalb der Programmoberfläche zuweisen. Dadurch kann die Adaptierbarkeit und die Übersichtlichkeit der Client-Software verbessert werden.

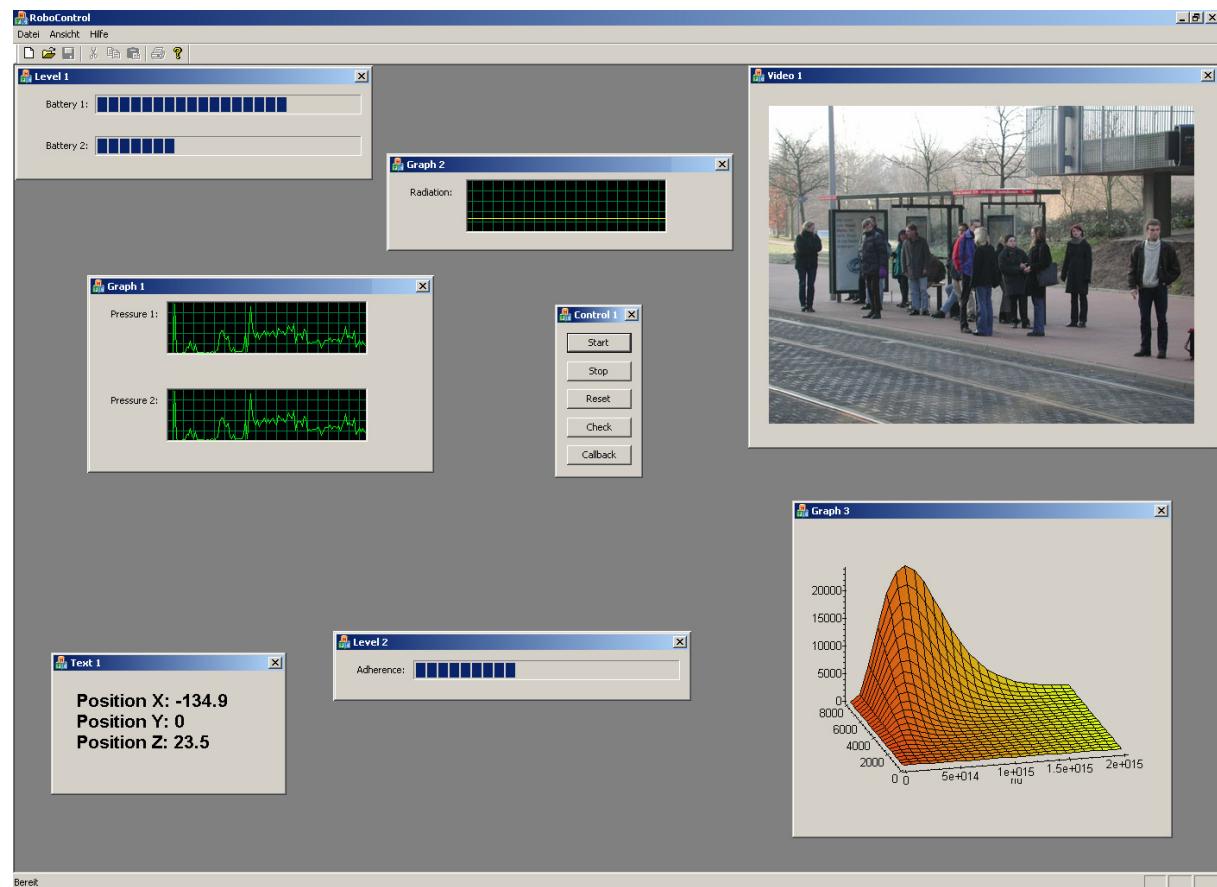


Abbildung 18: Verbesserter GUI Prototyp der Client-Software.

## **7 Eidesstattliche Erklärung**

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Bremen, 10. November 2008

---

Unterschrift

## 8 Literaturverzeichnis

### 8.1 Literatur

- [Bormann 2002] Bormann, C. und J. Ott: Konzepte der Internet Technik.  
SPC TEIA Lehrbuch Vlg., Berlin 2002
- [Kaiser 1996] Kaiser, J.: Technische Informatik: Prozessorarchitektur.  
Universität Ulm, Fakultät für Informatik, Ulm 1996
- [Knudsen 2000] Knudsen, J. B. und M. L. Noga: Das inoffizielle Handbuch für  
LEGO MINDSTORMS Roboter. O'Reilly, Köln 2000
- [LEGO 2006a] The LEGO Group (Hrsg.): LEGO MINDSTORMS NXT:  
Communication Protocol. The LEGO Group, Billund 2006
- [LEGO 2006b] The LEGO Group (Hrsg.): LEGO MINDSTORMS NXT:  
Direct Commands. The LEGO Group, Billund 2006
- [LEGO 2006c] The LEGO Group (Hrsg.): LEGO MINDSTORMS NXT:  
ARM7 Bluetooth Interface Specification. The LEGO Group,  
Billund 2006
- [LEGO 2006d] The LEGO Group (Hrsg.): LEGO MINDSTORMS NXT:  
Bluetooth Developer Kit. The LEGO Group, Billund 2006
- [Rupp 2007] Rupp, C.: Requirements-Engineering und -Management: Profes-  
sionelle, iterative Anforderungsanalyse für die Praxis, 4. Aufl.  
Hanser Fachbuchverlag, München 2007
- [Staab 2004] Staab, H.: Roboter-Echtzeitsteuerungen mit Windows CE.  
Fraunhofer Institut für Produktionstechnik und Automatisierung,  
Stuttgart 2004
- [Tanenbaum 2000] Tanenbaum, A. S.: Computernetzwerke, 3. Aufl.  
Pearson Studium, München 2000
- [Zeller 2005] Zeller, F.: Mensch-Roboter Interaktion: Eine sprachwissen-  
schaftliche Perspektive. Universität Kassel, Fachbereich Anglis-  
tik/Romanistik, Kassel 2005

## **8.2 *Online-Quellen***

- [GSM 2008] Global mobile Suppliers Association (Hrsg.): GSM/3G Stats.  
URL: <http://www.gsacom.com/news/statistics.php4>  
am 24.07.2008
- [LabVIEW 2008] Laboratory Virtual Instrumentation Engineering Workbench.  
URL: <http://www.ni.com/labview/d/>  
am 07.10.2008
- [Microsoft 2008a] Microsoft Corporation (Hrsg.): Microsoft Robotics Studio.  
URL: <http://msdn.microsoft.com/en-us/robotics/default.aspx>  
am 02.08.2008
- [Microsoft 2008b] Microsoft Corporation (Hrsg.): VPL Introduction.  
URL: <http://msdn.microsoft.com/library/bb483088> am 02.08.2008

# A Anhang

## A.1 Software Anleitungen

### A.1.1 Client Prototyp

Nachdem der Benutzer die Client-Software gestartet hat, stehen ihm folgende Menüpunkte über die Menüleiste zur Verfügung:

#### Datei -> Verbindung herstellen

Wird dieser Menüpunkt ausgewählt, erscheint ein neues Dialogfenster. Hier kann eine gültige IP-Adresse und eine Port-Nummer eingegeben werden, unter denen der Server zu erreichen ist. Hat der Benutzer in einer früheren Sitzung bereits eine IP-Adresse und eine Port-Nummer eingegeben, werden diese in den Eingabefeldern angezeigt. Zudem muss in dem Dialogfenster ein Benutzername und ein Passwort eingegeben werden. Wurden alle Einträge vorgenommen, kann der Benutzer über die Schaltfläche *Verbinden* eine Verbindung zum Server herstellen. Konnte eine Verbindung zum Server erfolgreich hergestellt werden, schließt sich das Dialogfenster und eine Statusmeldung zeigt im Hauptprogramm den erfolgreichen Verbindungsauftbau an. Ist der Server hingegen unter der angegebenen IP-Adresse nicht erreichbar oder hat der Benutzer einen falschen Benutzernamen und/oder ein falsches Passwort eingegeben, erscheint ein Nachrichtenfenster mit einer entsprechenden Hinweismeldung.

Nachdem eine Verbindung hergestellt wurde, kann der Benutzer im Hauptprogramm die von ihm definierten Aktionen (siehe auch *Steuerbefehle bearbeiten*) ausführen. Antworten vom Server werden hinsichtlich der vom Benutzer definierten Filter (siehe auch *Filter definieren*) gefiltert und repräsentiert.

#### Datei -> Verbindung trennen

Die Verbindung zum Server wird getrennt und die Datenanzeige gestoppt.

#### Konfiguration -> Steuerbefehle bearbeiten

Dieser Menüpunkt ruft ein neues Dialogfenster auf. Hier kann der Benutzer gespeicherte Steuerbefehle bearbeiten oder neue Steuerbefehle anlegen. Im Listenfeld in-

nerhalb des Dialogfensters werden die Bezeichnungen, Codes und Zuweisungen der gespeicherten Steuerbefehle angezeigt. Wird einer der Einträge in der Liste ausgewählt, kann über die Schaltfläche *Bearbeiten* der Eintrag verändert werden. Es erscheint ein weiteres Dialogfenster. Hier werden die Bezeichnung und der hinterlegte Steuerbefehl in einem Textfeld angezeigt. Diese Eingaben kann der Benutzer beliebig verändern. Zudem kann der Benutzer dem Steuerbefehl über ein Dropdown-Menü eine Aktion wie z.B. einen Tastenbefehl zuweisen. Über die Schaltfläche *Speichern* bzw. *Abbrechen* können die Änderungen gespeichert bzw. verworfen werden. Der Benutzer gelangt dann wieder zum Listenfeld.

Um einen neuen Steuerbefehl hinzuzufügen, muss der Benutzer die Schaltfläche *Hinzufügen* wählen. Es wird ein Dialogfenster mit den leeren Eingabefeldern für Bezeichnung, Steuerbefehl und Aktion aufgerufen. Hat der Benutzer die geforderten Eingaben getätigt, kann über die Schaltfläche *Speichern* bzw. *Abbrechen* die Eingabe gespeichert bzw. verworfen werden. Der Benutzer gelangt dann wieder zum Listenfeld. Die Bezeichnung des neu erstellten Steuerbefehls wird im Listenfeld aufgeführt.

### **Konfiguration -> Filter definieren**

Über diesen Menüpunkt kann der Benutzer festlegen, welche Antworten vom Server gefiltert und im Hauptprogramm repräsentiert werden sollen. Wurde der Menüpunkt ausgewählt, erscheint ein neues Dialogfenster. Die Bedienung ähnelt der der Steuerbefehl-Verwaltung. Anstelle von Steuerbefehlen muss der Benutzer hier einen Opcode definieren, nach dem gefiltert werden soll. Zudem muss eine Bytestelle angegeben werden, dessen Wert repräsentiert werden soll. Wurde ein Filter definiert, werden die Antworten vom Server hinsichtlich des definierten Opcodes gefiltert und der gewünschte Bytewert über die gewählte Darstellungsform im Hauptprogramm repräsentiert.

### **Datei -> Programm beenden**

Über diesen Menüpunkt kann der Benutzer die Client-Software beenden.

## A.1.2 Server Prototyp

Nachdem der Benutzer die Server-Software gestartet hat, stehen ihm folgende Menüpunkte zur Verfügung:

### Konfigurieren

Wählt der Benutzer diese Schaltfläche, wird ein neues Dialogfenster geöffnet. Hier kann die Schnittstelle zum Mikrocontroller konfiguriert werden. Es stehen Eingabefelder für die Angaben Portnummer, Baudrate, Bytegröße, Parität und Stoppbits zur Verfügung, die der Benutzer auswählen kann. Sollten noch keine Eingaben zuvor getätigt worden sein, werden vordefinierte Eingaben angezeigt. Wurde eine Eingabe getätigt, kann diese über die Schaltfläche *Speichern* bzw. *Abbrechen* gespeichert bzw. verworfen werden. Der Benutzer gelangt zum Hauptprogramm zurück. Zugriffsrechte können nur in der Datei *user.ini* hinzugefügt bzw. geändert werden, die sich im Programmordner der Server-Software befindet.

### Controller verbinden

Durch Auswahl dieser Schaltfläche wird die Schnittstelle zum Mikrocontroller unter Verwendung der gespeicherten Konfigurationsparameter (siehe auch *Konfigurieren*) geöffnet. Wurde die Schnittstelle erfolgreich geöffnet, wird dies über eine Statusanzeige im Hauptprogramm angezeigt. Ist die Verbindung hingegen gescheitert, wird dies über ein Nachrichtenfenster mitgeteilt.

Nachdem eine Verbindung mit dem Mikrocontroller hergestellt wurde, öffnet der Server automatisch die TCP/IP Schnittstelle. Eingehende Verbindungsversuche über diese Schnittstelle werden nun entgegengenommen und auf Authentizität geprüft.

### Verbindung trennen

Die Verbindung zu einem verbundenen Mikrocontroller wird getrennt.

### Programm beenden

Über diesen Menüpunkt kann der Benutzer die Server-Software beenden.

## A.2 Testroboter Konstruktionsbeschreibung

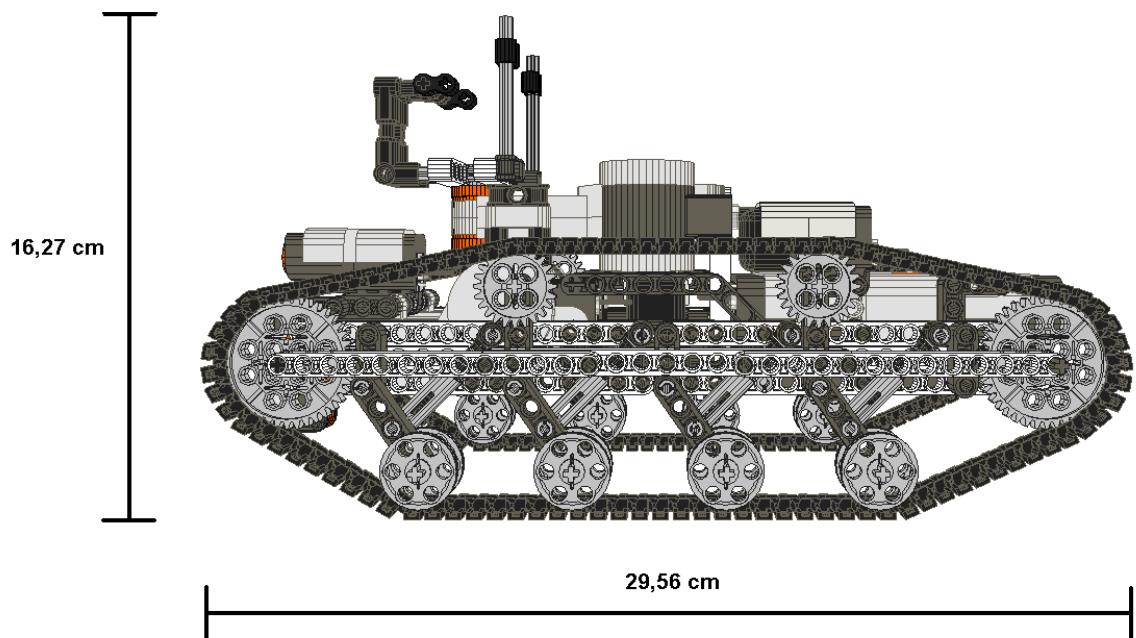


Abbildung 19: Seitenansicht.

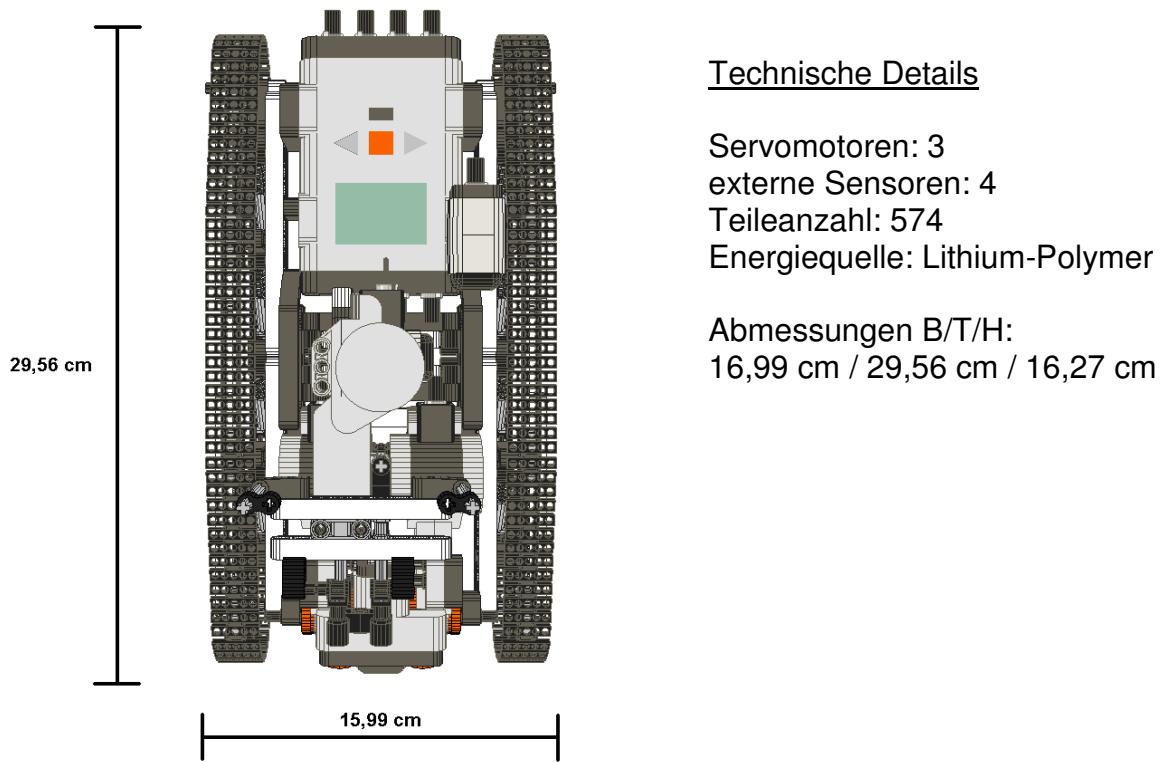


Abbildung 20: Sicht von oben.

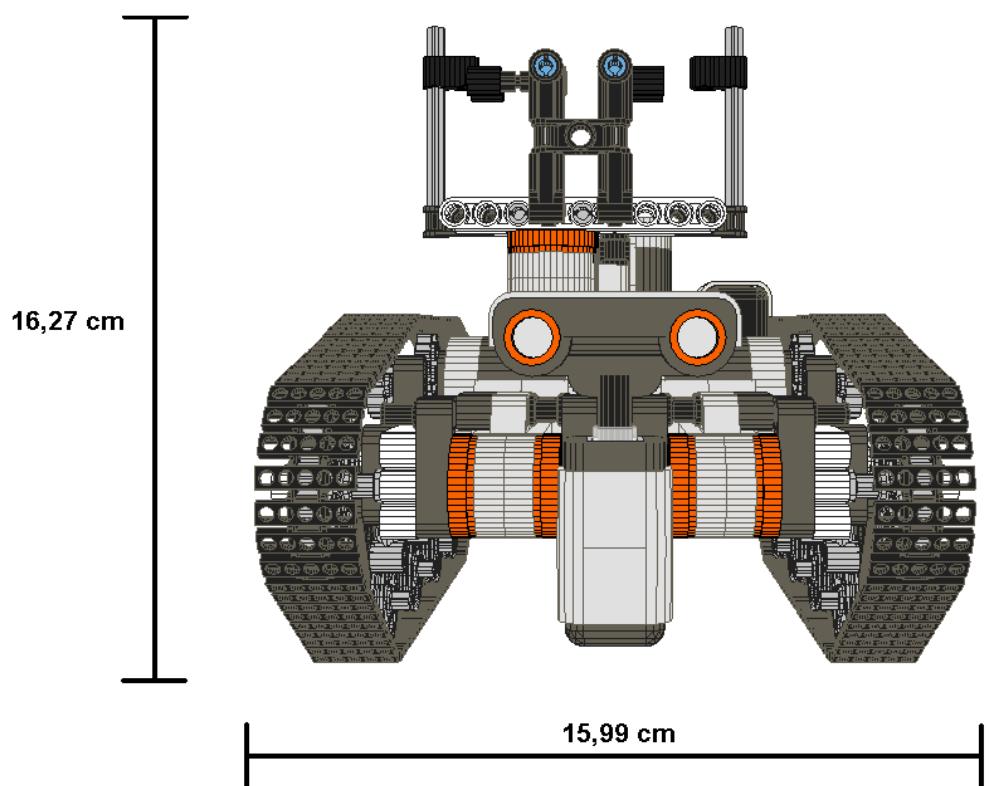


Abbildung 21: Frontansicht.

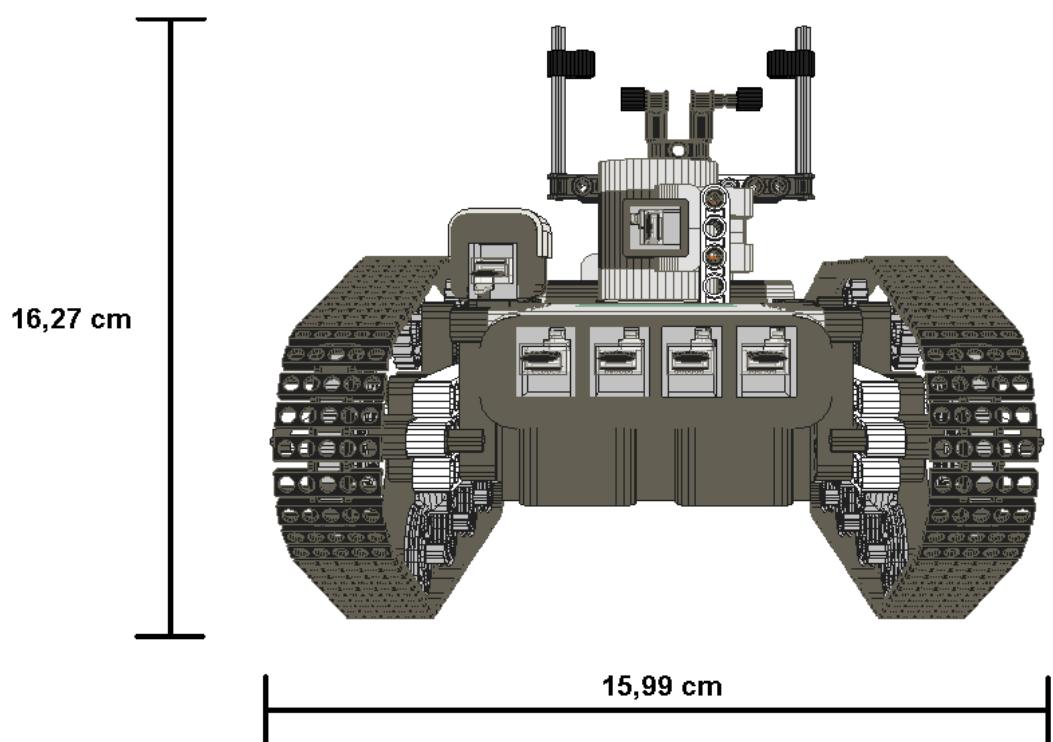


Abbildung 22: Heckansicht.

## A.3 Beispiel-Bytecode-Listen



Version: 1.00

### GETOUTPUTSTATE

Byte 0: 0x00 or 0x80  
Byte 1: 0x06  
Byte 2: Output port (Range: 0 – 2)

Return package:

Byte 0: 0x02  
Byte 1: 0x06  
Byte 2: Status Byte  
Byte 3: Output port (Range: 0 – 2)  
Byte 4: Power set point (-100 - 100)  
Byte 5: Mode (bit-field)  
Byte 6: Regulation mode (UBYTE; enumerated)  
Byte 7: Turn Ratio (SBYTE; -100 – 100)  
Byte 8: RunState (UBYTE; enumerated)  
Byte 9 – 12: TachoLimit (ULONG; Current limit on a movement in progress, if any)  
Byte 13 – 16: TachoCount (SLONG; Internal count. Number of counts since last reset of the motor counter)  
Byte 17 – 20: BlockTachoCount (SLONG; Current position relative to last programmed movement)  
Byte 21 – 24: RotationCount (SLONG; Current position relative to last reset of the rotation sensor for this motor)

Look at the entry for SetOutputState for details about the enumerations.

### GETINPUTVALUES

Byte 0: 0x00 or 0x80  
Byte 1: 0x07  
Byte 2: Input port (Range: 0 – 3)

Return package:

Byte 0: 0x02  
Byte 1: 0x07  
Byte 2: Status Byte  
Byte 3: Input port (Range: 0 – 3)  
Byte 4: Valid? (Boolean; TRUE if new data value should be seen as valid data)  
Byte 5: Calibrated? (Boolean; TRUE if calibration file found and used for “Calibrated Value” field below)  
Byte 6: Sensor type (enumerated)  
Byte 7: Sensor mode (enumerated)  
Byte 8 – 9: Raw A/D value (UWORD; device dependent)  
Byte 10 – 11: Normalized A/D value (UWORD; type dependent; Range: 0 - 1023)  
Byte 12 – 13: Scaled value (SWORD; mode dependent)  
Byte 14 – 15: Calibrated value (SWORD; Value scaled according to calibration. CURRENTLY UNUSED.)

Look at the entry SetInputModule for details about the enumerations.

### RESETINPUTSCALEDVALUE

Byte 0: 0x00 or 0x80  
Byte 1: 0x08  
Byte 2: Input port (Range: 0 – 3)

Return package:

Byte 0: 0x02  
Byte 1: 0x08  
Byte 2: Status Byte

## KEEPALIVE

Byte 0: 0x00 or 0x80  
Byte 1: 0x0D

Return package:

Byte 0: 0x02  
Byte 1: 0x0D  
Byte 2: Status Byte  
Byte 3 – 6: Current sleep time limit, milliseconds (ULONG)

## LSGETSTATUS

Byte 0: 0x00 or 0x80  
Byte 1: 0x0E  
Byte 2: Port (0 – 3)

Return package:

Byte 0: 0x02  
Byte 1: 0x0E  
Byte 2: Status Byte  
Byte 3: Bytes Ready (count of available bytes to read)

## LSWRITE

Byte 0: 0x00 or 0x80  
Byte 1: 0x0F  
Byte 2: Port (0 – 3)  
Byte 3: Tx Data Length (bytes)  
Byte 4: Rx Data Length (bytes)  
Byte 5 – N: Tx Data, where N = Tx Data Length + 4

For LS communication on the NXT, data lengths are limited to 16 bytes per command. Rx Data Length MUST be specified in the write command since reading from the device is done on a master-slave basis.

Return package:

Byte 0: 0x02  
Byte 1: 0x0F  
Byte 2: Status Byte

## LSREAD

Byte 0: 0x00 or 0x80  
Byte 1: 0x10  
Byte 2: Port (0 – 3)

Return package:

Byte 0: 0x02  
Byte 1: 0x10  
Byte 2: Status Byte  
Byte 3: Bytes Read  
Byte 3 - 19: Rx Data (padded)

For LS communication on the NXT, data lengths are limited to 16 bytes per command. Furthermore, this protocol does not support variable-length return packages, so the response will always contain 16 data bytes, with invalid data bytes padded with zeroes.

## **B CD-ROM**

### ***B.1 Inhalt***

- RoboControl-Client Programm kompiliert für Windows
- RoboControl-Client Quelltext
- RoboControl-Server Programm kompiliert für Windows Mobile für Smartphone 6.0
- RoboControl-Server Quelltext
- MOUSE MLCAD Zeichnungen
- PDF Dokument von dieser Diplomarbeit