# Konzeption und Entwicklung einer Smartphone-gestützten Schiffskontrolle zur Kollisionsvermeidung bei Schiffsmodellen

**Bachelorarbeit**

zur Erlangung des akademischen Grades Bachelor of Science
an der Hochschule für Technik und Wirtschaft Berlin,
Fachbereich Wirtschaftswissenschaften II,
Studiengang Angewandte Informatik

vorgelegt von Andreas Günther

1. Betreuer: Prof. Dr. Jürgen Sieck
2. Betreuer: Anton Mezhiborskiy

Berlin, den 26. Januar 2010

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

Mobiltelefone haben heute den Charakter von kleinen Computern mit denen der Nutzer unterwegs E-Mails abrufen, Bankgeschäfte online tätigen und komplexe Anwendungen ausführen kann. Sie sind mit Drahtlostechnologien für den Datenaustausch ausgestattet, haben berührungssensitive Bildschirme und besitzen verschiedenste Sensoren. Durch den Erfolg des iPhones und der Technologien anderer Anbieter, wie Google oder Microsoft, und deren Bereitstellung der jeweiligen Software Development Kits (SDK) bzw. Plattformen, die potenziell jeden befähigen Programme für solche Geräte zu entwickeln, konnten neue Anwendungen entstehen die mit der Umgebung interagieren und diese sogar in diese Applikationen integrieren. Warum nicht den Kühlschrank über das Smartphone befragen, was an Lebensmitteln benötigt wird oder den Fernseher und die Hifi-Anlage darüber bedienen? Bei der Gestaltung solcher Software sind der Kreativität der Entwickler keine Grenzen gesetzt. So wird in Zukunft der Alltag immer mehr mit der digitalen Welt verschmelzen und sich das Anwendungsgebiet dieser Geräte weiter ausdehnen. So ist es denkbar mit Hilfe von Smartphones Drohnen zu steuern um in gefährliches oder unzugängliches Gelände vorzudringen oder einen Beitrag zur Erforschung von künstlicher Intelligenz beitragen.

## 1.1 Zielsetzung

Ziel dieser Arbeit ist es prototypisch ein System zu entwickeln, das mit einer autonomen Steuerlogik (KI) ausgestattet werden kann, durch diese einem Modellschiff ermöglicht wird Hindernisse zu erkennen und diesen autonom

auszuweichen, wenn die Gefahr einer Kollision besteht. Dazu soll das Schiff mit einem Sensor ausgestattet werden, welcher Auskunft über die Entfernung zu einem Objekt gibt. Eine Smartphoneanwendung wertet anschließend diese Entfernungsdaten aus und überträgt entsprechende Befehle, zur Steuerung des Schiffs, drahtlos an dieses. Hierzu muss ein geeignetes Smartphone ausgewählt werden sowie eine entsprechende Drahtlostechnologie zur Datenübertragung. Das mobile Endgerät wird mit einer entsprechenden Anwendung ausgestattet, die es erlaubt verschiedene Steuerlogiken zur Kollisionsvermeidung auszuführen sowie das Schiff manuell zu Steuern. Weiterhin ist es nötig das Modellschiff mit geeigneter Elektronik, entsprechenden Sensoren und Aktoren auszustatten und aufzubauen. So sollen Befehle vom Smartphone Sensordaten empfangen, verarbeitet und diese Befehle vom Schiffsmodell umgesetzt werden können.

## 1.2 Aufbau der Arbeit

Zu Beginn werden grundlegende Aspekte potenziell einsetzbarer Technologien sowie mobile Geräte vorgestellt und verglichen, um zu ermitteln welche im Rahmen dieser Arbeit zum Einsatz kommen werden. Auch werden hier bereits existierende Systeme zur autonomen Steuerung exemplarisch vorgestellt. Dies soll als Grundlage für die darauf folgende Analyse der Anforderungen, die an das System gestellt werden, dienen.

Während der Anforderungsanalyse wird die Einsatzumgebung analysiert: Welche besonderen Gegebenheiten bestehen hier und welche Anforderungen ergeben sich dadurch für das System? Auch werden hier Anwendungsfälle, die innerhalb des Systems auftreten identifiziert und dargestellt.

Im Anschluss wird das System entworfen. Dazu gehören: der konzeptionelle Aufbau des Systems und dessen Komponenten sowie die Spezifizierung der Form, der zur Steuerung eingesetzten Daten. Weiterer Aspekt dieses Kapitels ist der Entwurf einer Benutzeroberfläche für die Smartphoneanwendung.

Darauf folgend wird dargestellt wie der Systementwurf technisch umgesetzt wurde. Dabei wird zum einen auf die Realisierung der Steuerung auf Seiten des Schiffsmodells und zum anderen auf die softwaretechnische Implementierung der mobilen Anwendung eingegangen.

Abschließend wird das entstandene System anhand der identifizierten Anforderungen bewertet und seine Funktionalität demonstriert.

# Kapitel 2

# Grundlagen

Dieses Kapitel geht grundlegend auf Teilbereiche dieser Arbeit ein, die Voraussetzungen für eine erfolgreiche Realisierung des Systems sind.

Zunächst werden Smartphones vorgestellt, um letztendlich das zum Einsatz kommende mobile Endgerät zu identifizieren. Dabei wird darauf geachtet, welche Drahtlostechnologien diese bieten und wie hoch der erforderliche Aufwand ist, um Anwendungen für diese Geräte zu entwickeln.

Anschließend werden Drahtlostechnologien vorgestellt, die zur Datenkommunikation zwischen Smartphone und Schiffsmodell bereitstehen.

Abschließend wird auf Aspekte der technischen Realisierung des Schiffsmodells eingegangen: Mit welchen Sensoren es ihm ermöglicht werden kann, Hindernisse zu erkennen und welche Aktoren für die Steuerung einsetzbar sind.

## 2.1 Smartphones

Smartphones sind mobile Endgeräte, die die Funktionalitäten von Mobiltelefonen und PDAs vereinen [Wik09b]. Sie erweitern sozusagen die klassischen Mobilfunkanwendungen zur Kommunikation mit anderen Teilnehmern um weitere Anwendungen, wie beispielsweise die Terminplanung, das Abrufen von Onlinediensten, Multimediaanwendungen oder direkter Datenaustausch zwischen Endgeräten von Nutzern. Besonders die Möglichkeit der mobilen Verfügbarkeit von Online-Diensten, wie das Surfen im Internet oder das Ab-

rufen von Emails, macht diese Geräte attraktiv für den Endverbraucher. Dazu sind Smartphones mit den entsprechenden Drahtlostechnologien, wie WLAN oder UMTS ausgestattet. Bluetooth und Infrarot sind neben den eben genannten Technologien zur Verbindung mit Internetdiensten ebenso zum drahtlosen Datenaustausch vorhaben. Wobei Infrarot mehr und mehr durch Bluetooth ersetzt wird.

Abbildung 2.1: Beispiele für Smartphones: Apple iPhone und HTC G1 [Appa], [Gooa]

### 2.1.1 Apple iPhone OS

Zur Anwendungsentwicklung bieten einige Smartphones zu ihrem Betriebssystem passende Entwicklungsumgebungen und SDK[1]s. Beispielsweise bietet Apple für das von ihnen entwickelte iPhone OS mit dem iPhone SDK eine entsprechende Umgebung zur Anwendungsentwicklung. Das iPhone OS ist in vier Schichten (siehe Abbildung 2.2 ) unterteilt, die dem Entwickler verschiedene Schnittstellen für das Erstellen von Anwendungen bereitstellt. Die unteren Schichten Core Service und Core OS stellen grundlegende Dienste für die oberen Schichten Media und Cocoa Touch bereit. Diese beiden unteren Schichten bilden somit die Basis der Anwendungsentwicklung für das iPhone. Entwickler arbeiten in der Regel auf den beiden oberen Schichten, da diese einen erweiterten Funktionsumfang und höhere Dienste beherbergen, die aber auf die Dienste der unteren Basis-Schichten zurückgreifen [App09b].

Voraussetzung für die Entwicklung von iPhone-Applikationen ist neben einer kostenpflichtigen Registrierung beim Apple iPhone Developer Program auch eine entsprechende Kenntnis der Programmiersprache Ojective C, die auch für die Mac OS Anwendungsentwicklung genutzt wird. Objective C ist

[1]Software Development Kit

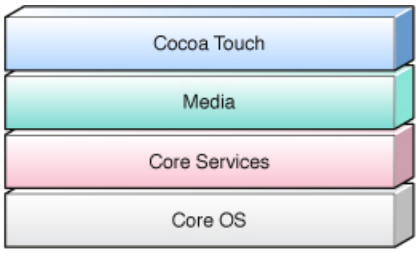


Abbildung 2.2: iPhone OS Architektur [Appb]

eine Weiterentwicklung des ANSI[2] C Standards und basiert zusätzlich auf Smaltalk, um die Objektorientierung zu gewährleisten. Eine entsprechende Entwicklungsumgebung für das iPhone ist mit XCode zur Zeit nur für das Mac OS verfügbar. Somit ist die Entwicklung nur für einen eingeschränkten Entwicklerkreis möglich.

### 2.1.2 Android OS

Im Gegensatz zum iPhone OS bietet das quelloffene Android Betriebssystem, das von Google und der Open Hand Set Alliance entwickelt wurde, mit der Entwicklungsumgebung Eclipse eine für jeden zugängliche Entwicklungsumgebung. Sowohl Mac als auch PC Benutzer, ob unter Linux oder Windows sind somit in der Lage Anwendungen für Smartphones mit Android Betriebssystem zu entwickeln. Google stellt hierzu ein entsprechendes Plugin für Eclipse sowie das auf Java SE[3] basierende Android SDK bereit. Das Android OS besteht aus fünf Schichten, deren Kern ein Linux Kernel ist. Dieser bildet die Abstraktionschicht zur Hardware des Geräts. Neben kompletten Basisanwendungen, wie E-Mail Client und Internet Browser, ist im System eine Davik Virtual Machine integriert, die für die Verwaltung und Ausführung der Anwendungen verantwortlich ist. Für den Entwickler werden entsprechende Java Bibliotheken für die Programmierung bereitgestellt [Gooc].

[2] American National Standards Institute
[3] Java Standard Edition

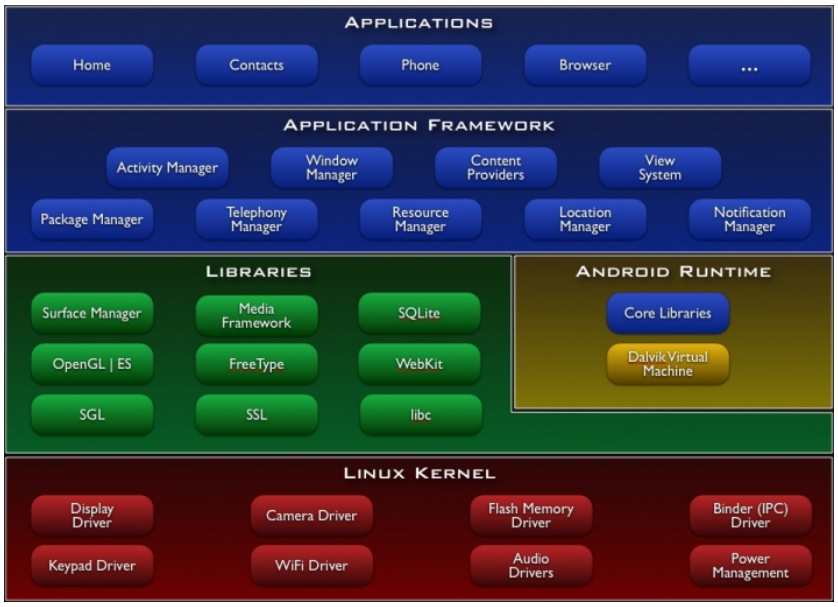


Abbildung 2.3: Android System Stack [Goob]

### Applikationen

Jede Applikation innerhalb des Android-Systems läuft als eigenständiger Prozess, der von der Laufzeitumgebung verwaltet wird. Diese Prozesse haben somit nicht selbst die Kontrolle über ihren Zustand im eigenen Lebenszyklus. Das bedeutet, dass, um jederzeit zu gewährleisten auf Benutzereingaben oder eingehende Anrufe reagieren zu können, das System in der Lage sein muss nicht mehr benötigte Prozesse zu beenden wenn nicht genügend Ressourcen zur Verfügung stehen. Dies wird durch eine Priorisierung der Prozesszustände *nicht aktiv*, *Hintergrundprozess*, *aktiver Dienst*, *sichtbarer Prozess* und *aktiver Prozess* realisiert (siehe Abbildung 2.4 )[HM09, Seite 77].

### Benutzerschnittstellen

Um auf Eingaben von Benutzern zu reagieren oder Informationen auf dem Bildschirm eines Android-Smartphones anzeigen zu können arbeitet das System mit sogenannten Aktivitäten (eng. Activity). Diese Aktivitäten bilden zusammen mit Steuerelementen, die zur Eingabe von Benutzerinteaktionen oder zur Darstellung von Inhalten auf dem Bildschirm genutzt werden, die Benutzerschnittstelle.

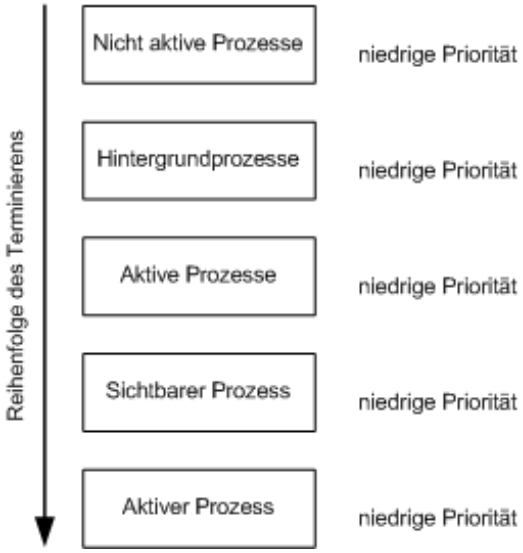


Abbildung 2.4: Reihenfolge der Prozessterminierung

Hinter jeder Bildschirmanzeige auf dem Gerät verbirgt sich eine Aktivität die eine Reihe von Steuerelementen verwaltet und weitere Aktivitäten aufrufen kann, wenn beispielsweise eine Anwenderinteraktion dies verlangt. Aktivitäten können, je nach Position auf dem von der Laufzeitumgebung verwalteten Stack die Zustände *aktiv*, *gestoppt*, *pausiert* oder *zerstört* annehmen. Im Zustand *aktiv* ist diese Aktivität das oberste Element auf den Stack und ist für den Benutzer zu sehen. Eine Aktivität verliert ihren Focus und wechselt in den Zustand gestoppt, wenn eine andere den Fokus erhält oder die Aktivität geschlossen wird. Der Zustand *zerstört* tritt nur dann auf, wenn die von ihr belegten Ressourcen benötigt werden. Dann wird sie zu diesem Zweck aus dem Speicher entfernt [HM09, Seite 85]

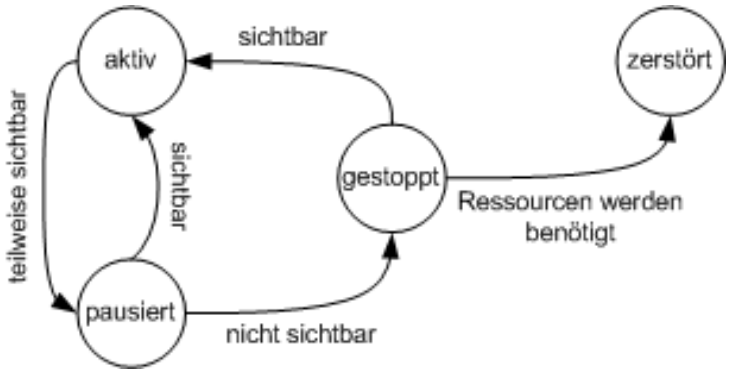


Abbildung 2.5: Zustandsübergänge im Lebenszyklus einer Aktivität

Um Funktionalität und Layout der Steuerelemente zur Eingabe und Ausgabe von Informationen zu gewährleisten werden die Layoutinformationen der Views in Form von XML-Dateien gespeichert. Für dynamische Ansichen können die View-Eigenschaften aber auch zur Laufzeit des Programms angepasst werden. Neben einfachen Views, die genau eine Bildschirmkomponente darstellen, gibt es auch Gruppen-Views (eng. *GroupView*), die eine Reihe von Views beinhalten können. Views sind innerhalb einer Aktivität für das Zeichnen und die Ereignisverwaltung der Benutzerschnittstelle verantwortlich [HM09, Seite 97]. In Listing 2.1 ist beispielhaft ein XML-Layout angegeben, das aus einem *LinearLayout*, einem *GroupView*, besteht und ein einfaches *TextView* und einen *Button* enthält.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                android:layout_width="fill_parent"
4                android:layout_height="fill_parent"
5                android:orientation="vertical" >
6      <TextView android:id="@+id/text"
7                android:layout_width="wrap_content"
8                android:layout_height="wrap_content"
9                android:text="Hello, I am a TextView" />
10     <Button android:id="@+id/button"
11             android:layout_width="wrap_content"
12             android:layout_height="wrap_content"
13             android:text="Hello, I am a Button" />
14 </LinearLayout>
15 <!-- Quelle: http://developer.android.com/guide/topics/ui/index.html -->
```

Listing 2.1: Beispiel eines XML-Layouts

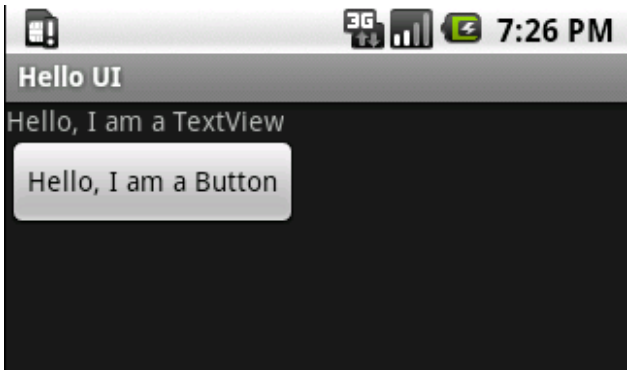


Abbildung 2.6: Benutzeroberfläche entsprechend dem Layout aus Listing 2.1

#### Geräte

Zur Zeit sind Endgeräte von HTC, Samsung, Motorola und T-Mobile verfügbar, die Android als Betriebssystem einsetzten. HTC, die zusammen mit T-Mobile das G1 als erstes Android-Smartphone im Otober 2008 auf den Markt gebracht haben, bietet die Meisten Modelle an. Weitere Modelle sind unter anderen von Motorola und Sony Ericsson auf dem Markt vorhanden [Wik10].

### 2.1.3 Weitere Betriebssysteme

Die Betriebsysteme für Smartphones beschränken sich nicht nur auf die beiden bereits genannten. Das webOS von Palm, das auf dem Palm Pre eingesetzt wird, das von Microsoft entwickelte Windows Mobile, welches mit dem .Net Compact Framework arbeitet und das Symbian OS sind ebenso auf solchen Geräten anzutreffen. Für diese Systeme werden ebenfalls entsprechende SDKs und Entwicklungsumgebungen zur mobilen Anwendugsentwicklung angeboten. Da diese Systeme, bedingt durch die Ausstattung der damit verbundenen Geräte, ebenso geeignet wären die geforderten Anforderungen an das, im Rahmen dieser Arbeit zu realisierende System zu erfüllen, wird sich im Folgenden, aufgrund dieser Vielfalt, exemplarisch auf eine Auswahl zwischen den im Vorfeld genauer betrachteten Betriebssytemen Android und iPhone OS beschränkt. Eine zusätzliche Betrachtung aller mobilen Betriebssysteme würde den Rahmen dieser Arbeit überschreiten.

### 2.1.4 Zusammenfassung

Prinzipiell eignen sich sowohl iPhone OS als auch ein mit dem Anroid Betriebssystem ausgestattetem Smartphone, wie das HTC Hero, für die Prototypentwicklung. Beide bieten potentiell einsetzbare Drahtlostechnologien sowie Entwicklungsumgebungen für die Anwendungserstellung. Doch durch die plattformunabhänige Anwendungsentwicklung für das Android Betriebssystem, wird dieses für den Prototypen des Systems zum Einsatz kommen. Besonders in Bezug auf das spätere Anwendungsgebiet (siehe 3.1.1) des Systems ist dies von Vorteil. So kann eine breite Entwicklergemeinde entstehen, die nicht auf ein bestimmtes System zur Anwendungsentwicklung beschränkt ist. Wie bereits erwähnt ist die iPhone-Anwendungsentwicklung ausschließlich unter dem Mac OS möglich, wohingegen die Entwicklung für das Android

Betriebsystem plattformunabhängig ist.

## 2.2 Drahtlose Kommunikation

Im Allgemeinen beschreibt drahtlose Kommunikation in der IT die kabelungebundene Datenübertragung. Smartphones bieten hier eine Reihe von Technologien, die zur drahtlosen Datenübertragung eingesetzt werden. Besonders die lizenzfreien und somit kostenlosen Funktechniken werden für die Umsetzung des Prototypen von Interesse sein. Diese sind WLAN, insbesondere der 802.11b/g Standard, und Bluetooth, da beide im lizenzfreien 2,4GHZ ISM[4]-Frequenzband arbeiten. Für Smartphone-gestützte Systeme die Distanzen von mehr als 100m überwinden müssen, werden die kostenpflichtigen Mobilfunktechnologien GSM, EDGE oder UMTS zum Einsatz kommen müssen. Im Rahmen dieser Arbeit wird allerdings davon ausgegangen, dass sich der Prototyp innerhalb eines kleineren Radius bewegt und somit WLAN oder Bluetooth für die Datenkommunikation ausreichend ist.

### 2.2.1 Bluetooth

Bei Bluetooth handelt es sich um einen offenen Industriestandard, der durch die Bluetooth SIG[5] spezifiziert wurde und Standards für diese Technologie veröffentlicht. Sie spezifiziert insgesamt drei Geräteklassen, die sich durch Stromverbrauch und Sendeleistung unterscheiden. So können je nach Geräteklasse 10m bis 100m zur Datenübertragung überbrückt werden (siehe Tabelle 2.1 ).

| | |
|---|---|
| Klasse3 | Niedrigste Leistungsklasse, max. Entfernung 10m |
| Klasse2 | Mittlere Leistungsklasse, max. Entfernung 20m |
| Klasse1 | Höchste Leistungsklasse, max. Entfernung 100m |

Tabelle 2.1: Bluetooth Geräteklassen [Abd06, Seite 21]

Bluetooth erlaubt den Zusammenschluss von zwei bis acht Geräten in einem Piconet. Dabei fungiert ein Gerät als Master, mit dem sich alle anderen Geräte des Netztes syschroniseren und der den Datenverkehr zwischen den

[4]Idustrial-Scientific-Medical-Frequenzband; lizenzfreier Frequenzbereiche für allgemeine Aufgaben (Babyphone, kabellose Kopfhörer, Autozentralverriegelung)

[5]Special Inerest Group

Stationen verwaltet. Zur Realisierung größerer Netzte können mehrere Piconets zu einem Scatternet zusammengeschlossen werden. Hier dient ein Gerät als Vermittler zwischen den Piconets.

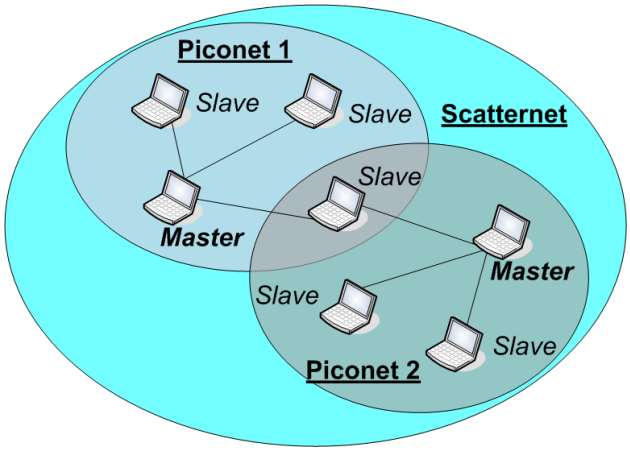


Abbildung 2.7: Scatternet

Da die derzeitigen Android Geräte am Markt noch mit der Version 1.5 ausgeliefert werden, kann die Bluetoothschnittstelle zur Datenübertragung nicht genutzt werden. Sie ist erst wieder in der kürzlich veröffentlichten Version 2.0[6] für Entwickler zugreifbar. Somit ist Bluetooth prinzipiell nutzbar, allerdings müsste dann diskutiert werden inwiefern die Reichweite der in den Geräten verbauten Bluetoothmodule für den Einsatz in einem weiteren Prototypen ausreichend ist. Aus diesem Grund wird für die drahtlose Datenübertragung auf WLAN zurückgegriffen.

### 2.2.2 Wireless LAN

Grundlegend wurde WLAN 1997 durch die IEEE im Standard 802.11 spezifiziert [Abd06, Seite 60]. Heute gibt es eine Reihe von Erweiterungen, mit denen man im Vergleich zum ursprünglichen Standard höhere Datenraten erreichen kann, die aber auch Sicherheitsaspekte oder die Kommunikation zwischen den Stationen betreffen. Die aktuellste Erweiterung ist die 802.11n Spezifikation, die im September 2009 offiziell verabschiedet wurde [Ins09]. Sie zeichnet sich besonders durch eine Bruttodatenrate von 600Mbps aus. Die gebräuchlichsten Erweiterungen des 802.11 Standards sind die 802.11b/g

[6]Release: Dezember 2009

Spezifikationen, die Datenraten von 11Mbps bzw. 54Mbps erreichen. Die Datenübertragung findet bei diesen Standards im freien 2,4GHz-ISM Frequenzband statt, wobei es auch Abkömmlinge des ursprünglichen Standards gibt, die Daten im 5GHz Bereich übertragen (z.B. 802.11a).

| Norm | Max. Bruttodatenraten | Frequenzband |
|---|---|---|
| 802.11 | 2Mbps | 2,4GHz |
| 802.11a | 54Mbps | 5GHz |
| 802.11b/g | 11Mbp/54Mbps | 2,4GHz |
| 802.11n | 600Mbps | 2,4GHz und 5GHz |

Tabelle 2.2: Einige Varianten von IEEE 802.11

Es stehen zwei Topologien zur Verfügung um den Datenaustausch im WLAN zu realisieren. In der Ad-hoc-Topologie kommunizieren alle Stationen, die sich in Reichweite einer WLAN-Zelle befinden direkt miteinander (Peer-to-Peer). In der Infrastruktur-Topologie hingegen müssen alle Stationen erst eine Verbindung zu einem Vermittler herstellen, dem Access Point (AP). Der AP verwaltet alle angeschlossenen Stationen und regelt die Datenübertragung von einer Station zur anderen. Diese Topologie ist am häufigsten anzutreffen, da sie über den AP auch eine Anbindung an kabelgebundene Netzwerke ermöglicht, was bei Ad-hoc Netzwerken nicht möglich ist. Fällt in einem Infrastrukturnetz der zentrale Knoten aus, ist die gesamte Kommunikation in ihm lahmgelgegt, wo hingegen in einem Ad-hoc Netz lediglich die Kommunikation zu der ausgefallenen Station unterbrochen ist.

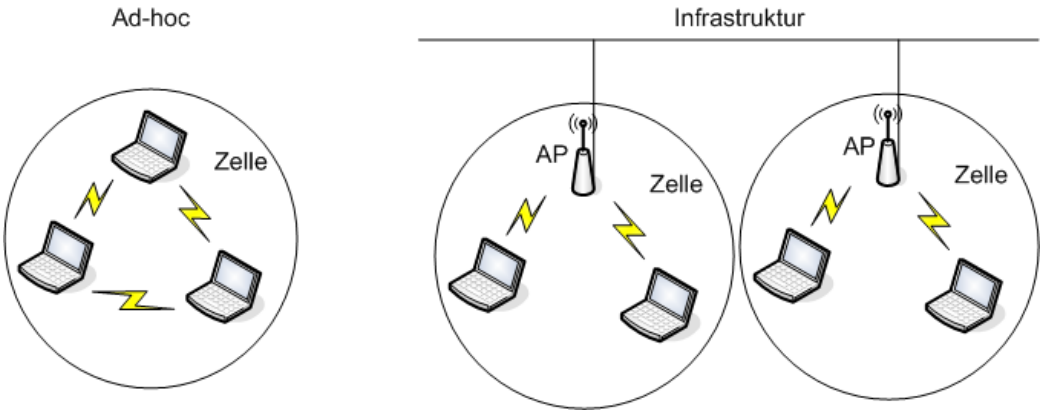


Abbildung 2.8: WLAN Topologien

## 2.3 Physical Computing

Unser Leben wird mehr und mehr durch Computer bestimmt. Sie begleiten uns tagtäglich bei allen Aktivitäten, egal ob wir eine E-Mail mit dem heimischen PC versenden, uns per Mobiltelefon zum Kaffee verabreden, einen Snack aus einem Automaten ziehen oder uns abends mit der elektrischen Zahnbürste die Zähne putzen. In all diesen Geräten stecken kleine Computer. Diese werden hier nicht mehr ausschließlich als Geräte begriffen, die Eingaben über Tastatur und Maus annehmen, Programme ausführen und deren Ergebnise auf einem Bildschirm darstellen. Vielmehr sind diese Geräte vielfältiger in ihren Formen und Bedienmöglichkeiten und können sogar selbstständig Dinge erfassen. Allgemein kann man sie als eine "[...] Verbindung von physikalischen Systemen mit Software und Hardware zu einem interaktiven Ganzen" [MJA09, Seite 31] betrachten. Physikal Computing schafft hier eine Verbindung zwischen realer Welt und der rechnergestützten Verarbeitung von Daten mit Hilfe von Computern oder Mikrocontrollern. Dem Benutzer kann so "[...] durch gegenständliche Bedienschnittstellen und Modelle realer Gegenstände das rechnergestützte Arbeiten in der gewohnten haptisch erfahrbaren Welt ermöglicht." [Hel08, Seite 65] werden. Damit lässt sich der Umgang mit diesen intelligenten Geräten deutlich flexibler gestalten und auf den Benutzer direkt anpassen. [MJA09, Seite 32]. Ein Multi-Touch-Tisch ist dafür ein gutes Anwendungsbeispiel. Für die Verbindung der Maschine mit der realen Umgebung spielen hierbei Sensoren und Aktoren eine wichtige Rolle.

### 2.3.1 Sensoren und Aktoren

Sensoren und Aktoren erlauben es einem technischen System Veränderungen in der Umgebung zu erfassen oder Einfluss auf diese zu nehmen. Sensoren sind in der Lage physikalische Eigenschaften der realen Welt, wie Temperatur, Druck, Helligkeit, Gewicht oder Feuchtigkeit wahrzunehmen und zu messen. Die Messungen können dabei ungenau, widersprüchlich oder mehrdeutig sein, daher ist es meist noch erforderlich, die erhaltenen Rohdaten anschließend weiter zu verarbeiten, sie beispielsweise durch Ausschluss bestimmter Werte zu verfeinern [Neh02, Seite 25]. Aktoren hingegen wandeln eine physikalische Eingangsgröße in eine andere Ausgangsgröße um. Innerhalb des Anwendungsgebiets Physikal Computing wird als Eingangsgröße Elektrizität verwendet, wobei die Ausgangsgrößen vielseitig sind. Beispiele hierfür sind Ton, Licht, Wärme oder Bewegung [MJA09, Seite 171, 189].

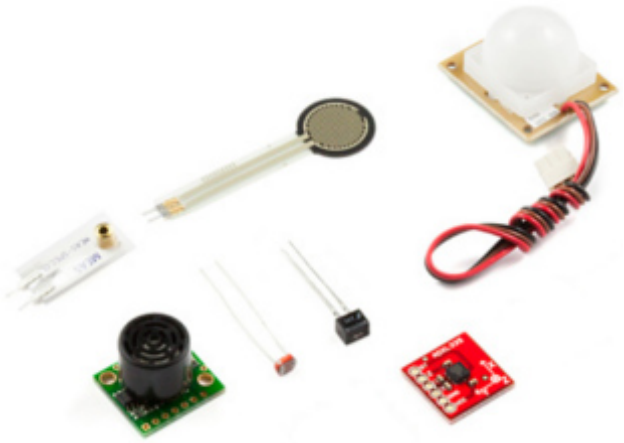

Abbildung 2.9: Beispiele für Sensoren (von oben links nach unten rechts: Vibrationssensor, Drucksensor, Bewegungssensor, Ultraschallsensor, Photozelle, Phototransistor, Beschleunigungssensor)

Die Automatisierung des Schiffsprototypen bedingt natürlich den Einsatz von Sensoren und Aktoren. Aktoren werden für die Steuerung klassisch in Form von Motoren für das Ruder und den Antrieb eingesetzt. Für die Erfassung von Hindernissen können optische Sensoren oder Ultraschallsensoren eingesetzt werden. Sie basieren auf demselben Prinzip. Sie senden Schall oder Licht aus und empfangen die durch Hindernisse verursachte Reflexion des ausgesendeten Mediums. Anhand der Zeit zwischen Senden und Empfangen kann die Entfernung zu einem Objekt berechnet werden. Hier wird ein Ultraschallsensor Anwendung finden, da ein passender optischer Sensor nicht zur Verfügung steht. Auf die Möglichkeit die Objekterkennung mit Hilfe einer Kamera vorzunehmen wird aufgrund der aufwendigen Bildverarbeitung bewusst verzichtet.

### 2.3.2 Arduino

Zur rechnergestützten Verarbeitung der ermittelten Sensordaten sowie zur Ansteuerung der Aktoren wird eine Schnittstelle benötigt, die eine Verbinung zwischen der Welt der Dinge und der digitalen Welt der Bits und Bytes schafft. Dazu steht die Arduino Plattform zur Verfügung, mit dem diese Verknüpfung realisiert werden kann. Sie besteht zum einem aus der Arduino-Entwicklungsumgebung und zum anderen aus dem Arduino-Board. Mit der Entwicklungsumgebung werden Programme für das Arduino-Board geschrie-

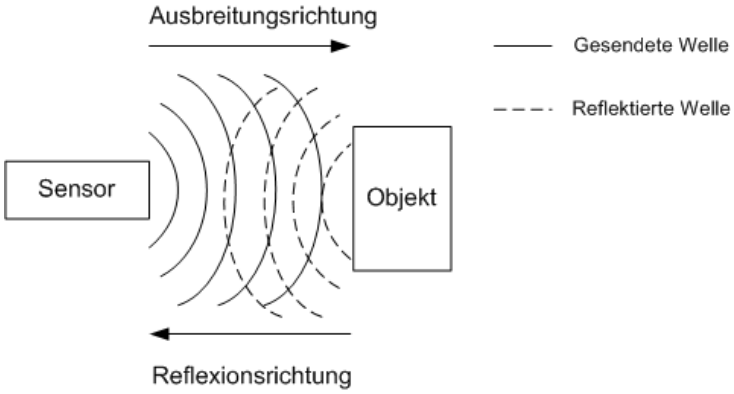


Abbildung 2.10: Prinzip der Abstandsmessung durch Ultraschall

ben, die anschließend auf diesem ausgeführt werden. So können die Daten von den angeschlossenen Sensoren ausgelesen und Aktoren angesteuert werden. Für das Anschließen von Sensoren/Aktoren bietet das Arduino-Board eine Reihe von analogen Eingängen sowie digitale Ein- und Ausgänge und Anschlüsse für die Stromversorgung der Bauteile. Sowohl Software als auch die Schaltpläne des Boards stehen als Open Source zur Verfügung [MJA09, Seite 5]. Daher sind auf dem Markt auch verschiedene Ausführungen des Boards erhältlich. Der Selbstbau ist, das nötige Know-How vorausgesetzt, natürlich auch nicht ausgeschlossen. Auch werden Erweiterungen, die sogenannten Shields, die die Entwicklung von komplexeren Systemen erleichtern, z.B. Enternet-Shield oder Motor-Shield, angeboten.

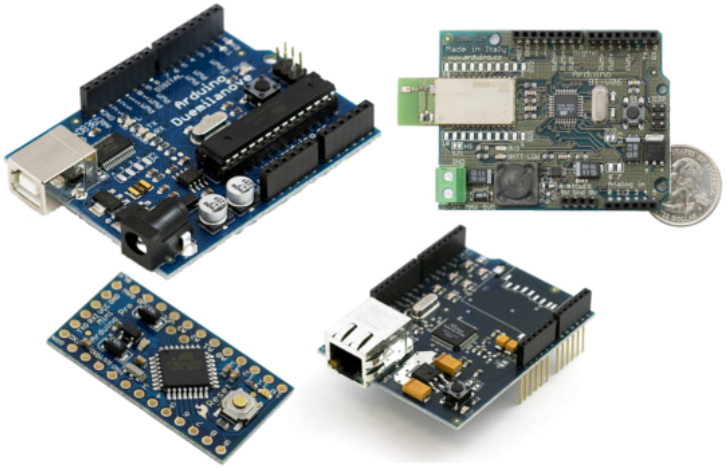

Abbildung 2.11: Arduino USB, Arduino Bluetooth, Arduino Mini und Arduino Ehternet Shield (von oben links nach unten rechts) [Spa]

## 2.4 Beispiele autonomer mobiler Systeme

Berachtet man den Bereich der Robotik innerhalb des Physical Computings sind dort häufig autonome mobile Systeme zur Erkundung von Umgebungen, die für Menschen unzugänglich oder gefährlich sind anzutreffen. Aber auch auf dem Gebiet der Erforschung von künstlicher Intelligenz eignen sie sich "hervorragend dafür, Hypothesen über intelligentes Verhalten, Wahrnehmung und Kognition zu überprüfen und zu verbessern."[Neh02, Seite 11] Im Folgenden werden exemplarisch drei solcher autonomen mobilen Systeme vorgestellt.

### 2.4.1 Schiffsteuerung des Miniatur Wunderland in Hamburg

Das Miniatur Wunderland ist nach seiner Eröffnung 2001 zu einer der beliebtesten Touristenattraktionen im Großraum Hamburg aufgestiegen [Ham]. Die einstige Idee die Größte Modelleisenbahn der Welt zu bauen wurde bis heute ehrgeizig verfolgt und umgesetzt. Doch es finden sich auf den 6.400m$^2$ Ausstellungsfläche nicht ausschließlich Züge. Im laufe der Jahre wurde die Anlange durch Straßennetz, Beleuchtungssystem sowie einem Teil der Nordsee, auf dem Modellschiffe verkehren ergänzt. Zur Zeit werden die Schiffe noch manuell per Funkfernsteuerung gelenkt. Doch es wird bereits seit mehreren Jahren an einer automatischen Steuerung der Schiffe gearbeitet.

"Voraussetzung für einen automatisierten, computerkontrollierten Betrieb ist [...] eine permanente und äußerst präzise Ortung aller auf der Nordostsee befindlichen Schiffe" [Min]. Zu Beginn dieses ehrgeizigen Projekts, im Januar 2004, wurde zunächst versucht die Ortung mit Hilfe von Infrarotkameras vorzunehmen. Dieser Ansatz wurde allerdings verworfen, nachdem man feststellte, dass zur Behandlung verschiedener Eiflussfaktoren, wie Schiffhöhe und Wasserstand ein zu komplexes System benötigt wird. Daher kam die Idee auf das System mit Hilfe von Ultraschallsensoren zu realisieren. Dabei wurde die Position der Schiffe mit Hilfe von Kreuzpeilung[7] bestimmt. Tests mit dieser Methode waren sehr vielversprechend. Aufgrund von Störungen im System, die durch einen Unfall mit einem Funkgerät auftraten, wurde dazu übergegangen mit Tonfolgen zu arbeiten, um unabhängig von äußeren Einflüssen zu sein. Parallel zu diesem System wurde eine Infrarotsystem

[7]Standortermittlung mit Hilfe von zwei Standpunkten (Peilungen)

entwickelt. Über eine entsprechende Mechanik wird das infrarote Licht als Fächer über die überwachte Fläche ausgestrahlt. Sensoren in den Schiffen nehmen dieses Licht wahr, wodurch erkannt wird wo sich das Schiff gerade auf der Anlage befindet. Dieses hybride System befindet sich zur Zeit in der abschließenden Testphase und wird wohl bald zum Einsatz kommen [Min].

### 2.4.2 DeepC

Ziel dieses 2001 initierten, aber bereits eingestellten Entwicklungsprojekts war es ein unbemanntes, autonomes Unterwasserfahrzeug (AUV[8]) zu entwickeln, das flexibel für verschiedene maritime Anwendugsgebiete, wie die Vermessung des Meeresbodens oder die Inspektion von Unterwasserkabeln oder Pipelines, einsetzbar ist. Bis dahin wurden solche Aufgaben ausschließlich von kabelgebundenen Unterwasserfahrzeugen, sogenannten ROVs[9], durchgeführt [Bun01].

Abbildung 2.12: Das AUV DeepC [SCH05]

Der mit Brennstoffzellen betriebene *DeepC* besteht im wesentlichen aus zwei Druckbehältern und einem Nutzlastmodul, das mit verschiedenen technischen Geräten für die jeweilige Aufgabe ausgestattet werden kann. Es ist zusätzlich mit entsprechenden Sensoren für die Navigation unter und über Wasser sowie mit einem Kurzstreckensonar zur Objekterkennung ausgestattet. Die autonome Steuerung wurde bei diesem AUV um ein intelligentes Missionsmanagement erweitert, womit es ihm ermöglicht wurde, den für eine bestimmte Aufgabe erstellten Missionsplan auszuführen. Dabei war es dem *DeepC* auch möglich in bestimmten Situationen in den Missionsplan einzugreifen, etwa wenn die Erfüllung dieser durch ein Hindernis gefährdet sein

[8] Autonomous Underwater Vehicle
[9] Remotely Operated Vehicles

sollte. In diesem Fall wurde durch eine Hindernisvermeidung "[...] die geplante Mission solange unterbrochen, bis ein gefahrloses Fortsetzen der ursprünglich vorgegebenen Fahrstrecke möglich ist. "[Pfü04] Im wesentlichen basierte das Softwaremodell der autonomen Steuerung auf dem *menschlichen Handlungsmodell nach Rassmussen*. Rassmussen untersuchte die kognitiven Kontrollmechanissmen des Menschen und entdeckte, das diese in drei Ebenen eingeteilt sind: *fähigkeitsbasierte Ebene*, *regelbasierte Ebene* und *wissensbasierte Ebene*. Durch die Anwendung diese Modells auf das zu erstellende System wurde die in Abbildung 2.13 abgebildete Softwarearchitektur entwickelt.

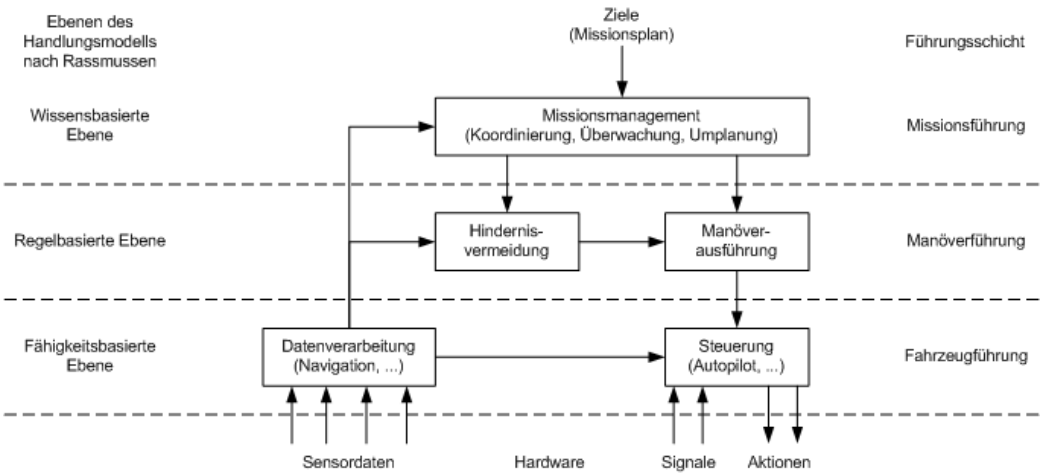


Abbildung 2.13: Verwendete Softwarearchitektur des AUV DeepC [Pfü04, Seite 20]

*DeepC* wurde zum ersten mal im Februar 2005 eingesetzt und im Herbst 2005 wurde das Projekt mit einer Demonstation im Tiefwasser der Biskaya abgeschlossen. [SCH05, Seite 37]

### 2.4.3 MARVIN MarkII

Der Marvin Mark II ist ein von der Technischen Universität Berlin im Rahmen des COMETS Projekts[10] entwickelter Modellhelikopter, der autonom eine vorgegebene Route von dreidimensionalen Wegpunkten abfliegen kann. Das COMETS Projekt beschäftigte sich mit der Entwicklung verteilter Kontrollsysteme in Verbindung mit UAVs[11], was auch Helikopter und Luftschiffe

[10] http://www.comets-uavs.org/
[11] Unmanned Aerial Vehicles

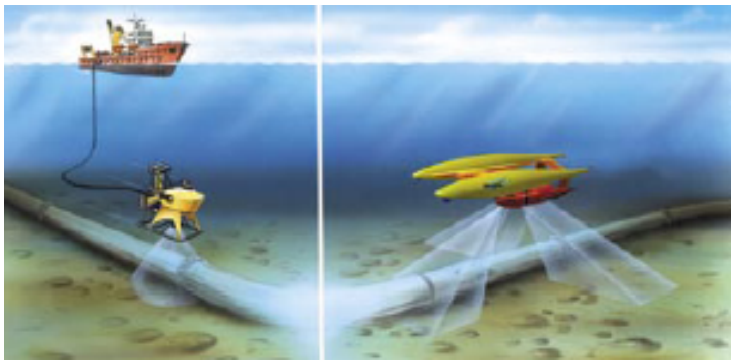

Abbildung 2.14: Unterwasser-Pipeline-Inspektion - links: mit ROV, rechts: mit DeepC [SCH05]

einschloss. [com06] Von Seiten der TU Berlin wurde dafür der Marvin Helikopter entwickelt, welcher auf einem kommerziellen Modellhelikopter von der Firma Aero-Tec basiert und mit einer Reihe von Sensoren und anderem technischen Equipmet ausgestattet wurde um das autonome fliegen zu ermöglichen. Es ist auch möglich Nutzlast zu befestigt um beispielsweise Luftaufnahmen der Umgebung zu machen. [Tec07]

Abbildung 2.15: Marvin Mark II in autonomen Flug [Teca]

Durch den Systemaufbau ist es zudem möglich zwischen manuellem Betrieb, über eine Fernsteuerung, und autonomen Betrieb während des Fluges hin- und herzuschalten. Dabei ist die manuelle Steuerung lediglich dazu da in unvorhergesehenen Situationen eingreifen zu können. Beispielsweise wird au-

tomatisch in den manuellen Betrieb gewechselt, wenn die Leistung der Batterien der Steuerelektronik für den autonomen Betrieb nicht mehr ausreicht. [Tec06b] Zur autonomen Steuerung des Helikopters generiert ein integrierter Mikrocontroller entsprechende Signale für die Servomotoren, die die Rotorblätter in entsprechende Stellungen zum Lenken des Helikopters bringen. Das Kontrollprogramm des Helikopters, das sich auf dem Mikrocontroler befindet erhält über die angeschlossenen Sensoren Informationen über Position und Lage des Helikopters. Zur genauen Positionserkennung kommt DGPS[12] und zur Bewegungserkennung kommt eine Kombination aus einer Reihe von Geschwindigkeits-, Beschleunigungs- und Magnetfeldsensoren zum Einsatz [Tec06b]. Über ein Mission Control Programm am Boden können dreidimensionale Wegpunkte für den autonomen Flug festgelegt werden, die entsprechend dem aktuellen Flugmodus, *straight line* und *curved*, angeflogen werden. Beim *straight line*-Modus fliegt der Helikopter bis zum nächsten Wegpunkt und stoppt dort. Anschließend dreht er sich in Richtung des nächsten Wegpunts und fliegt anschließend zu diesem. Prinzipiell funktioniert dies beim *curved* Modus ebenso. Dabei hält das Fluggerät allerdings nicht an jedem Wegpunkt an um sich in Richtung des nächsten zu drehen. Dies geschieht bereits während des Anflugs auf den aktuellen Wegpunkt. [Tec06a]

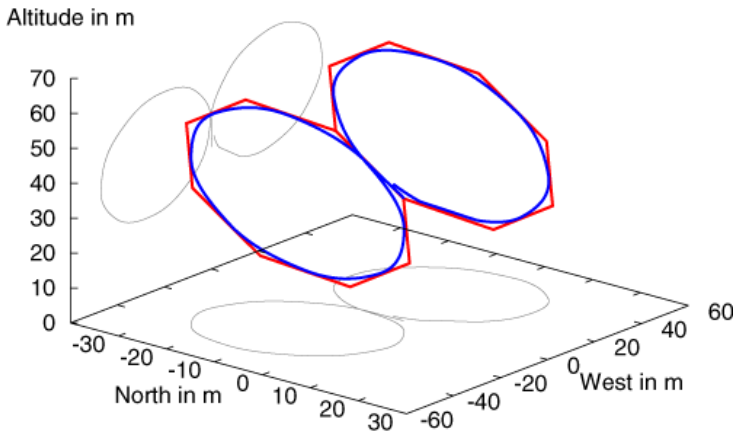


Abbildung 2.16: Simulierter Flug. Die roten Linien beschreiben einen Pfad in *straight line* Flugmodus, die blauen Linien einen Pfad in *curved* Flugmodus. [Tecb]

[12]Differential Global Positioning System

# Kapitel 3

# Anforderungsanalyse

In diesem Kapitel sollen die Anforderungen an das zu entwickelnde System identifiziert werden. Dazu werden zunächst das spätere Awendungsgebiet und die zur Verfügungstehenden Mittel zur Realisierung angeführt. Anhand dessen werden die Anforderungen, die an das System gestellt werden abgeleitet und Anwendungsfälle identifiziert.

## 3.1 Anwendungsumgebung

### 3.1.1 Einsatzgebiet

Da sich die automatische Steuerung eines Schiffsmodells in die mobile Robitik einordnen lässt, sind potentiell auch alle Anwendungsbreiche der mobilen Robotik für dieses System denkbar. Zum einen kommen solche Systeme in für den Menschen unzugänglichen oder gefährlichen Umgebungen zum Einsatz. Zum anderen liegt ein weiterer Anwendungsbereich im Gebiet der künstlichen Intelligenz (KI)[Neh02].

Das System, das hier prototypisch entwickelt werden soll findet hier im Bereich der Forschung und Entwicklung von künstlicher Intelligenz seinen Anwendungsbereich. Um den Ehrgeiz und die Kreativität in diesem Gebiet zu fördern werden Wettbewerbe veranstaltet, bei denen die Entwickler ihre implementierten KIs gegeneinander antreten lassen können. Die Software, mit der das System zu diesem Zweck ausgestattet wird hat in diesem Umfeld die Aufgabe das Modellboot autonom durch einen Parcours zu steuern. Dieser

ist mit Hindernissen ausgestattet, die das System mit Hilfe des am Schiffsmodell befestigten Sensors erkennen kann und anhand dessen das Schiff um die Hindernisse herummanövriert. Anhand der Schnelligkeit und der Anzahl der fehlerhaften Manöver des Bootes kann somit die Güte der eingesetzten KI bestimmt werden. Das Smartphone steckt hier den Rahmen des Möglichen bei der KI-Implementierung ab. Da auf ihm nur begrenzte Resourcen vorhanden sind, in Bezug auf Speicher, Leistung und die durch das SDK limitierten Möglichkeiten, kann auch nur innerhalb dieses Rahmens entwickelt werden. Auch die Ausstattung des Schiffmodells, das mit Hilfe der KI gesteuert wird grenzt den Umfang der Softwaresteuerung ein. Dadurch hat jeder Teilnehmer die selben technischen Voraussetzungen um sich dem Wettstreit zu stellen.

Die innerhalb des Wettbewerbs gemachten Erfahrungen können auch zur Entwicklung von künstlicher Intelligenz in anderen Anwendungsgebieten ebenso beitragen, da man innerhalb der KI-Entwicklung immer wieder auf ähnliche Probleme stößt.

### 3.1.2 Rahmenbedingungen für die Entwicklung

Die bereitstehende Hardware für die Umsetzung des Systems bildet die Grundlage für den späteren Systementwurf. Sie bestimmt weiche softwaretechnischen Mittel eingesetzt werden können und welche Infrastruktur vorherrschen muss. Im besonderen Bestimmt das Smartphone diese Rahmenbedingungen, da es die zentrale Steuereinheit des Systems darstellt. Das Schiffsmodell hat im Wesentlichen nur die Aufgabe die grundlegenden Parameter für die Steuerlogik zu liefern und die daraus resultierenden Steuerbefehle umzusetzen.

Als Steuereinheit kommt das HTC Hero zum Einsatz. Es besitzt das erforderliche, in 2.1 beschriebene Android Betriebssystem, und bietet darüber hinaus auch die erfoderliche WLAN-Funktechnologie für die Datenübertragung. Als Programmiersprache wird Java Anwendung finden. Eine Andere Sprache zur softwaretechnischen Umsetzung auf dem Smartphone kann nicht eingesetzt werden, da die Wahl des Beriebssystems ausschließlich Java als Sprache zur Anwendungsentwicklung zulässt.

Für den Bau des Schiffsmodells stehen folgende Komponenten zur Verfügung:

- Pollux Schlepper Schnellbausatz
- Arduino Duemilanove

- WLAN Modul WiFly GSX 802.11b/g
- Ultraschallsensor Maxbotix LV-EZ1
- Ardumoto - Motor Driver Shield
- Servomotor für die Ruderanlage
- Komplettantrieb Multispeed 140

Zentrales Element wird hier das Arduino Duemilanove sein, das wie schon in 2.3.2 erwähnt die Schnittstelle zwischen realer Umgebung und der rechentechnischen Systemkomponenten ist.

## 3.2 Systemanforderungen

Ähnlich wie bei einer klassischen Fernsteuerung für Modellboote soll eine Schiffssteuerung auf einem Smartphone entwickelt werden. Aus dieser Aufgabenstellung lassen sich eine Reihe von Anforderungen an das Gesamtsystem ableiten.

Die Steuerung soll hier ausschließlich über das Smartphone geschehen. Dabei ist es nebensächlich, ob die Befehle für Geschwindigkeit und Richtung des Schiffs manuell durch den Smartphonenutzer oder autonom mit Hilfe der auf dem mobilen Endgerät befindlichen KI erzeugt werden. Für die autonome Steuerung soll ein entsprechendes Softareniterface, innerhalb des Steuerprogramms, geschaffen werden, das es erlaubt diese leicht auszutauschen. Für die manuelle Bedienung der Fernsteuerung soll dem Nutzer eine einfache und intuitive Benutzeroberfläche geboten werden, die es ihm ermöglicht zum einen Steuerbefehle zu generieren und zum anderen Informationen über den aktuellen Zustand des Schiffs, z.B. Geschwindigkeit und Richtung, zu erhalten. Zwar liegt das Anwendungsgebiet primär bei der autonomen Steuerung des Modells, dennoch ist es nötig durch die manuelle Bedienung grundlegende Funktionstests am System vornehmen zu können, um etwa festzustellen ob der Antrieb ordnungsgemäß funktioniert. Damit das Modell auch noch bei einer fehlerhaften automatischen Steuerlogik manövrierfähig bleibt soll die automatische Steuerung in solchen Fällen abgeschaltet werden können.

Da das Smartphone eine Fernsteuerung des Schiffs darstellt sollte auch die Datenübertragung ebenso wie bei einer klassischen Funkfernsteuerung direkt

zwischen beiden Teilsystemen erfolgen. Bricht die Datenverbindung Smartphoneseitig ab, sollte das Schiff, um etwaige Schäden oder sogar Verlust zu vermeiden, entsprechend reagieren. Die zu verarbeitenden Daten für das System sollten unnötigen Overhead vermeiden, da unter Umständen sonst die Interpretation und Ermittlung der daraus resultierenden Reaktion zu viel Zeit in Anspruch nehmen könnten. Es soll sich auf die wesentlichen Daten beschränkt werden.

Durch modularen Aufbau der Komponenten, Hardware- wie auch Softwareseitig, soll es ermöglicht werden den Prototypen leicht zu warten und für andere Anwendungsgebiet zu erweitern. Das muss im Besonderen bei der Konzeption die KI-Komponente des Systems beachtet werden, da sie, gerade bei Wettkämpfen sehr oft ausgetauscht, angepasst bzw. erweitert wird.

## 3.3 Use-Case-Analyse

Bei der Betrachtung der gestellten Aufgabe lassen sich zwei Teilsysteme identifizieren, für die entsprechende Anwendungsfälle beschrieben werden müssen, die sich aus den Systemanforderungen ableiten lassen.

### 3.3.1 Smartphone

Die Steuerung auf dem Smartphone hat wie bereits erwähnt die Aufgabe die vom Schiff gesendeten Entfernungsdaten zu einem Objekt zu empfangen und anschließend zu verarbeiten. Hier wird die aktuelle Situation in der sich das Schiff befindet ermittelt und anhand dessen entsprechende Steuerbefehle für Geschwindigkeit und Fahrtrichtung daraus generiert und anschließend an das Modell gesendet. Das Generieren der Steuerbefehle kann auch durch den Smartphonenutzer beeinflusst werden indem er über das Userinterface des Geräts entsprechende Steuerbefehle eingibt, die anschließend an das Modellschiff gesendet werden. Dies soll Abbildung 3.1 verdeutlichen.

### 3.3.2 Schiffsmodell

In Abbildung 3.2 sind die Anwendungsfälle, die auf der Seites des Modells auftreten aufgeführt. Es sendet die empfangenen Daten des Ultraschallsen-

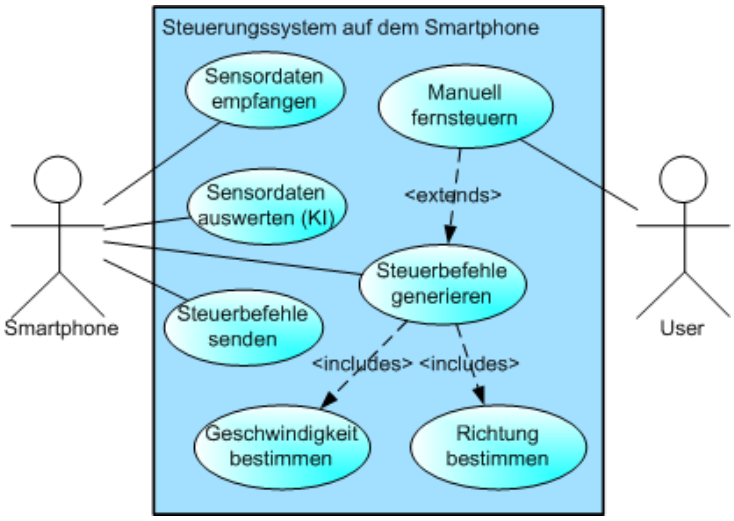


Abbildung 3.1: Anwendungsfälle des Steuerungssystems

sors, der die Entfernung des Schiffs zu einem Hindernis misst, an das Smartphone, die dort weiterverarbeitet werden. Anschließend erhält das Schiff entsprechende Steuerbefehle, die von ihm anschließend in der realen Umgebung umgesetzt werden, indem Ruder bzw. Schiffsschraube bewegt werden.

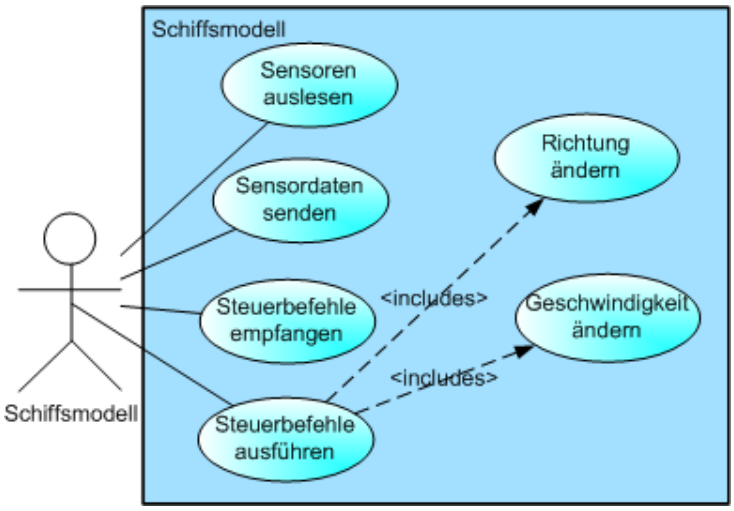


Abbildung 3.2: Anwendungsfälle des Schiffsmodells

# Kapitel 4

# Systementwurf

Das System gliedert sich in zwei Teilsysteme. Zum einen in das Schiffsmodell und zum anderen in eine Smartphone-Anwendung, mit der es möglich sein soll das Modell manuell über eine Benutzerschittstelle bzw. über eine integrierte Logik autonom fernzusteuern. Dazu ist es notwendig eine geeignete Architektur für beide Teilsysteme zu finden und eine Schnittstelle festzulegen, über die beide Teilsysteme miteinander Daten und Befehle austauschen können.

## 4.1 Architektur

Das System soll eine verteilte Architektur besitzen, wobei beide Teile, Schiff und Smartphone, über eine drahtlose Schnittstelle Daten austauschen können, wie in Abbildung 4.1 dargestellt. Das Smartphone ist dabei für die Steuerung des Schiffs und für die Auswertung der Sensordaten verantwortlich. Das Modell hat die Aufgabe Sensordaten dem Smartphone zur Verfügung zu stellen und die empfangenen Steuerbefehle umzusetzen.

Das System wird in 3 Schichten, im Sinnes des *Model-View-Controller Architekturmusters* in *Präsentationsschicht*, *Anwendungsschicht* und *Datenschicht* unterteilt. Jede Schicht ist in sich geschlossenen und kommuniziert ausschließlich über festgelegte Schnittstellen mit benachbarten Schichten. Durch diese Aufteilung der einzelnen Systemkomponenten in verschiedene Schichten ist das System flexibel gegenüber Änderungen oder Erweiterungen und erleichtert zudem die Wiederverwendbarket der Komponenten.

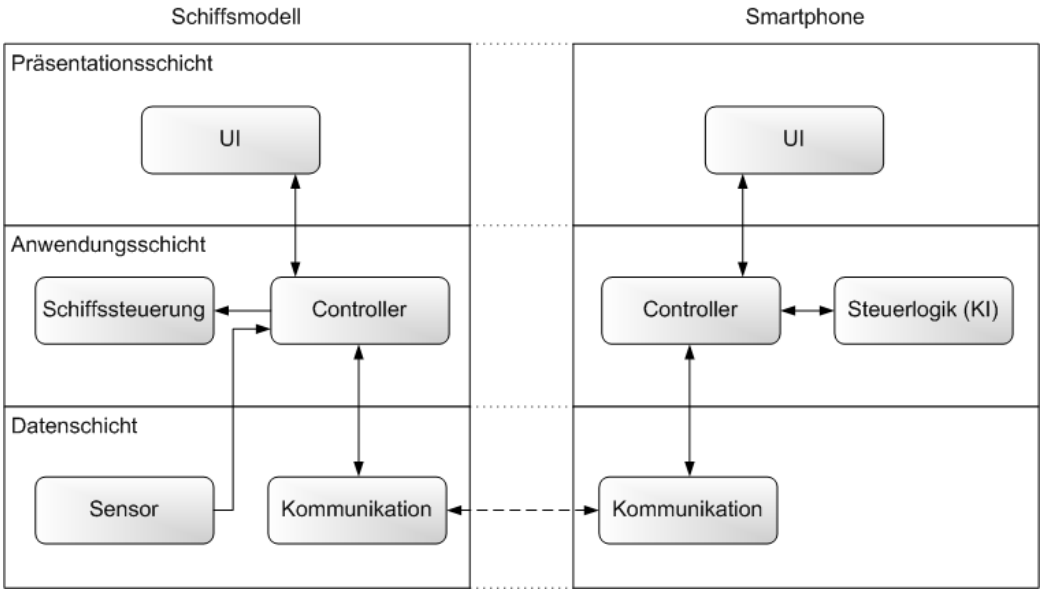


Abbildung 4.1: Architektur des Systems

Im Gegensatz zum klassischen *MVC-Modell*, in dem die *View-Komponente* direkten Zugriff auf die *Model-Komponente* hat, findet hier die Kommunikation ausschließlich über den *Controller* statt. Durch diese Trennung wird die Wiederverwendbarkeit der Komponenten für Anwendungen mit ähnlicher Problemstellung erhöht. In Abbildung 4.2 sind beide Varianten des *MVC Architekturmusters* gegenübergestellt.

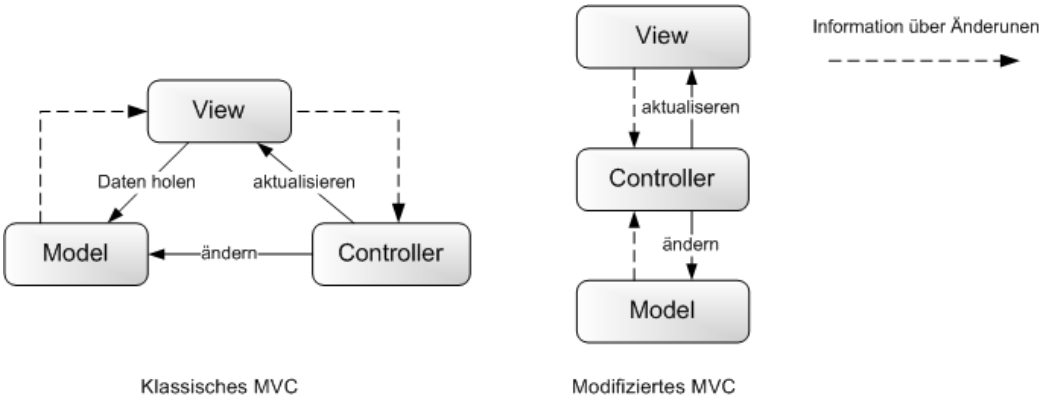


Abbildung 4.2: Versionen des MCV-Pattern

## 4.2 Datenschicht

Die *Datenschicht* hat im wesentlichen die Aufgabe die *Anwendungsschicht* zu informieren, wenn Daten zur Verarbeitung empfangen wurden, bzw. Daten, die von der *Anwendungsschicht* bereitgestellt werden zu versenden. Entfernungen zwischen dem Schiff und einem Objekt in Reichweite werden vom *Sensor* erfasst und an die *Controller-Komponente* der *Anwendungsschicht* weitergegeben. Für den Versand und den Empfang von Daten sind auf beiden Seiten, Schiff und Smartphone, die *Kommunikations-Komponenten* verandwortlich. Eine Datenhaltung findet nicht statt, da übertragene Daten direkt verarbeitet werden.

### 4.2.1 Datenmodell

Zur Datenübertragung zwischen Schiff und Smartphone wird ein einfacher Datenrahmen (eng. Dataframe) sowie verschiedene Typen für diesen spezifiziert. Ein Datenrahmen stellt einen bestimmten Befehl, der durch das System auszuführen ist dar. Die verschiedenen Rahmentypen und deren Funktion werden in 4.2.2 genauer beschrieben.

Der Datenrahmen besitzt eine einfache Struktur aus vier Byte, wobei das erste Byte zur Identifizierung des Datenrahmens dient, indem für ihn ein konstanter Wert festgelegt wird. Diesem *Start-Byte* folgt ein *Type-Byte*, das den Typ des Rahmens angibt. Der Rahmentyp spezifiziert so die Bedeutung des Werts, der im *Value-Byte* übertragen wird. Das *Stop-Byte* markiert abschließend das Ende des Rahmens. In Abbildung 4.3 ist diese Struktur bildlich dargestellt.

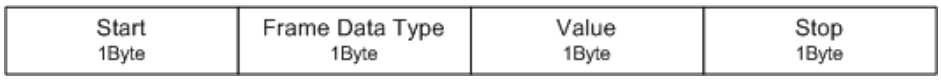


Abbildung 4.3: Datenrahmen zur Datenübertragung

Durch das festgelegte *Start-Byte* sowie das abschließende *Stop-Byte* kann innerhalb der *Datenschicht* der Teilsysteme ermittelt werden, ob es sich um einen solchen, für die Kommunikation zwischen Schiff und Smartphone erforderlichen Datenrahmen handelt. Wird einer dieser Rahmen empfangen, wird die *Anwendungsschicht* über das eintreffen der Daten informiert. In allen anderen Fällen werden die Daten verworfen.

### 4.2.2 Datentypen

Es werden sieben verschiedene Datentypen für den Rahmen festgelegt. Damit wird die softwaretechnische Auswertung und Verarbeitung der Daten vereinfacht, da Fehlinterpretationen durch diese feste Zuordnung eines Typs ausgeschlossen werden können.

#### Datentyp *Alive*

Dieser Datentyp signalisiert dem System, das eine Verbindung zwischen Schiffsmodell und dem Smartphone besteht aber keine Daten zur Verarbeitung bereitstehen. Ein Datenpaket mit diesem Typ wird ausschließlich unidirektional vom Smartphone zum Schiff übertragen. Stehen Daten zur Verarbeitung bereit werden diese in einem Datenrahmen des entsprechenden Typs gesandt und erfüllen somit auch die Funktion des *Alive*-Typs. Wird die Verbindung unterbrochen und das Schiff erhält innerhalb einer bestimmten Zeitspanne keine Daten mehr hält es selbstständig an. Somit wird ein unkontrolliertes Weiterfahren des Schiffes verhindert.

#### Datentyp *Init Ranging* und *Stop Ranging*

Mit einem Datenrahmen des Typs *Init Ranging* wird auf der Seite des Modellschiffs die Entfernungsmessung, zu in Reichweite befindlichen Objekten aktiviert. Mit dem Befehl *Stop Ranging* wird diese wieder deaktiviert.

#### Datentyp *Sonar*

Ein Datenrahmen von diesem Typ enthält einen Wert, der die Entfernung in Zentimetern zwischen den Schiff und einem in Reichweite befindlichen Objekts angibt. Dieser Rahmentyp wird ausschließlich unidirektional vom Schiff zum Smartphone übertragen, wo die Entfernungsinformationen weiterverarbeitet werden.

#### Datentyp *Speed Forward* und *Speed Backward*

Mit diesen Datentypen wird die Geschwindigkeit des Schiffs gesteuert. Dabei gibt ein Rahmen des Typs *Speed Forward* an, dass das Schiff sich vorwärts bewegen soll. Der Rahmentyp *Speed Backward* gibt die Geschwindigkeit in die entsprechende Gegenrichtung an.

#### Datentyp *Direction*

Mit Hilfe dieses Rahmentyps wird die Ruderstellung des Schiffs gesteuert. Der Wert des Rahmens gibt dabei eine entsprechende Gradzahl für die Stellung des Ruders im Bereich von 0° bis 180° an.

## 4.3 Anwendungsschicht

Die Aufgabe der *Anwendungsschicht* ist es bereitgestellte Daten der *Datenschicht* zu verarbeiten und gegebenenfalls der *Präsentationsschicht* zur Verfügung zu stellen bzw. auf Ereignisse, die von ihr ausgelöst werden zu reagieren. Sie beinhaltet somit die gesamte Logik des jeweiligen Teilsystems. Die *Anwendungsschicht* des Schiffsmodell verarbeitet die von der *Anwendungsschicht* der Smartphone-Applikation generierten Steuerbefehle und setzt diese um.

In Abbildung 4.4 ist der wesentliche Workflow zur Fernsteuerung des Schiffsmodells für beide Teilsysteme dargestellt. Die beiden Arten zur Steuerung, manuell oder automatisch, sind hier bereits integriert.

### 4.3.1 Manuelle Steuerung

Zur manuellen Steuerung des Schiffes werden die Benutzeraktionen von der *Präsentationsschicht*, die zu einer Erhöhung oder Reduzierung der Fahrtgeschwindigkeit oder einer Änderung der Richtung führen, vom *Controller* in Steuerbefehle umgewandelt und anschließend über die *Datenschicht* an das Schiffsmodell gesandt. Das Schiff setzt dann über seine Aktoren die Steuerbefehle um. Das Auswerten von Schiffsdaten durch die *Steuerlogik* wird vernachlässigt, sobald eine Benutzeraktion zur Steuerung des Schiffs durch

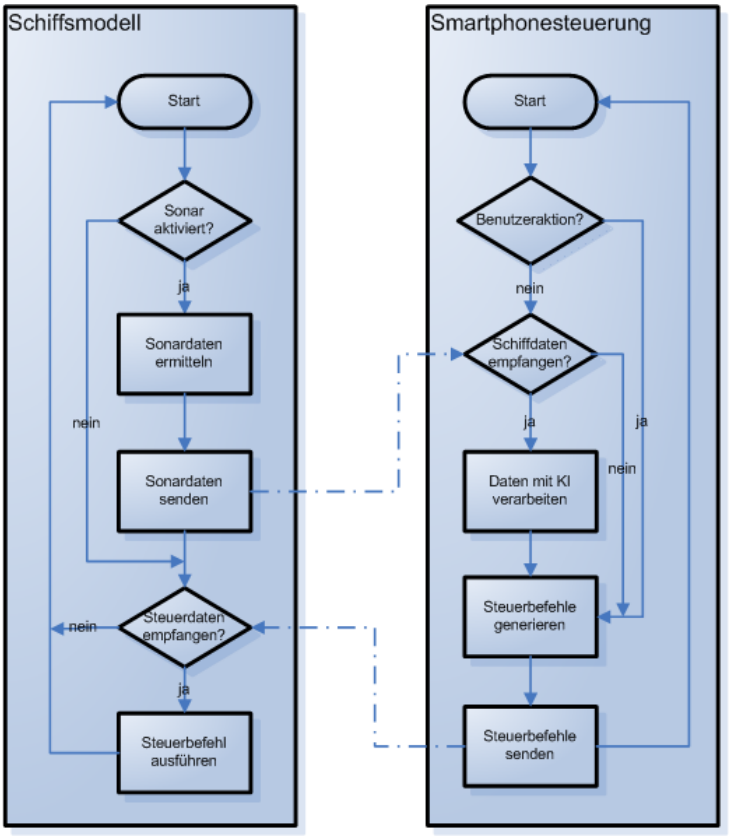


Abbildung 4.4: Diagramm zum Programmablauf

die Smartphone-Applikation entdeckt wird.

### 4.3.2 Automatische Steuerung

Die automatische Steuerung setzt voraus, dass das Ermitteln von Sensordaten aktiviert ist, ansonsten können von der Smartphone-Applikation keine Steuerbefehle generiert werden, da ihr so die Grundlage dafür fehlen würde. Ist die Entfernungsmessung eingeschaltet werden zunächst die entsprechenden Daten vom Schiffssensor ermittelt und an das Smartphone gesendet. Dort werden diese vom *Controller* entgegengenommen und an die *Steuerlogik* zur Verarbeitung weitergeleitet. Sie entscheidet anhand dieser Daten welches Manöver das Schiff durchführen soll um im Falle einer Kollisionsgefahr, diese zu vermeiden. Wie im einzelnen die Manöverentscheidung getroffen wird, hängt von der jeweiligen Implementierung der *Steuerlogik* ab. Ein konkreter Entwurf einer Logik kann aufgrund der in 3.1.1 beschriebenen Anwendungsumgebung nicht vorgenommen werden, da diese innerhalb des Einsatzgebietes ständig ausgetauscht wird. In 4.3.3 wird allerdings ein möglicher Ansatz für die Konzeption einer solchen Logik vorgestellt. Der *Controller* generiert aus der getroffenen Entscheidung entsprechende Steuerbefehle für das Modellschiff und leitet sie zur Übertragung an die *Datenschicht* weiter, welche vom Schiff empfangen und umgesetzt werden.

### 4.3.3 Konzept einer Steuerlogik

Anhand einer Steuerlogik soll entschieden werden wie das Schiff in bestimmten Situatuionen reagieren soll. Als Grundlage der Entscheidungsfindung dienen Sensordaten, die vom Schiff ermittelt und an das Smartphone gesandt werden. Bei den Sensordaten handelt es sich um Entfernungen des Schiffs zu einem Objekt, das im überwachten Bereich des am Schiff angebrachten Ultraschallsensors liegt. Es wird immer die kürzeste Distanz zu einem Objekt ermittelt, d.h. befinden sich mehrere Objekte innerhalb der Reichweite des Sensors, wird ausschließlich die Entfernung zum nächstgelegenen Objekt ermittelt.

Um nun zu entscheiden welches Manöver das Schiff durchführt um einem Hindernis auszuweichen, werden innerhalb des überwachten Bereichs *Gefahrenstufen* (*Level*) definiert. Jede Stufe beschreibt dazu wie hoch die Gefahr einer Kollision ist und wie sich das Schiff verhält, wenn ein Objekt inner-

halb dieses Teilbereichs entdeckt wurde. In Abbildung 4.5 ist exemplarisch eine Einteilung in vier *Gefahrenstufen* vorgenommen worden. Je höher das *Level* in dem ein Hindernis ermittelt wurde desto höher ist die Gefahr einer Kollision.

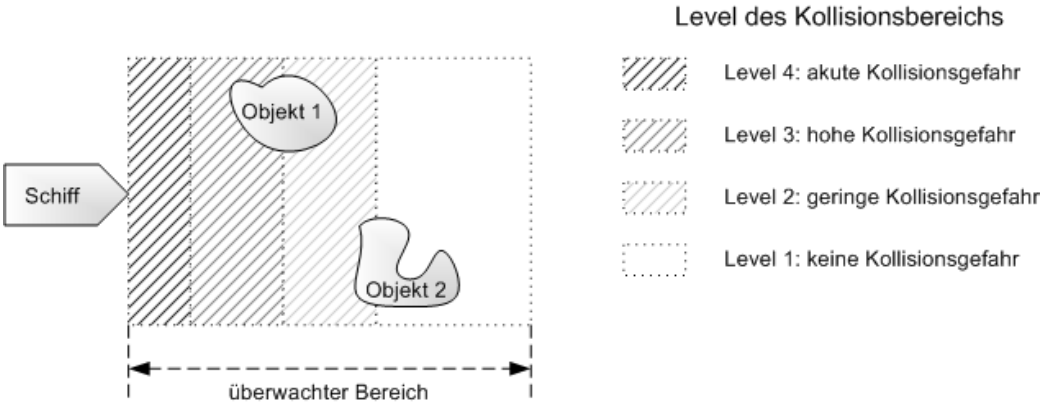


Abbildung 4.5: Einteilung des überwachten Bereiches zur Kollisionsvermeidung

Nachdem die Gefahrenstufe für das Modellschiff ermittelt wurde müssen entsprechende Gegenmaßnahmen ergriffen werden, um eine Kollision zu vermeiden. Dazu wird für jede Stufe festgelegt welches Manöver das Schiff ausführen soll. In Tabelle 4.1 sind Manöver für die vier *Gefahrenstufen* aus Abbildung 4.5 aufgeführt.

| Gefahrenstufe | Manöver |
|---|---|
| Level 1 | Geschwindigkeit und Richtung beibehalten |
| Level 2 | Geschwindigkeit beibehalten und Richtung ändern |
| Level 3 | Geschwindigkeit verringern und Richtung ändern |
| Level 4 | volle Kraft zurück |

Tabelle 4.1: Schiffmanöver in Abhängigkeit der Gefahrenstufe

Da vom Ultraschallsensor bis auf die Entfernung des Objekts zum Schiff keine weiteren Positionsdaten (Objekt liegt mehr rechts oder mehr links vom Schiff) ermittelt werden können, muss auch eine entsprechende Logik für die Richtungsänderung konzipiert werden. Vorstellbar wäre dazu, dass das Schiff für eine gewisse Zeit in eine bestimmte Richtung gelenkt wird. Verschwindet das Objekt innerhalb dieser Zeit nicht aus dem Gefahrenbereich, lenkt man das Schiff in die entsprechend andere Richtung. Führen beide Änderungen

der Fahrtrichtung nicht zur freien Fahrt, kann davon ausgegangen werden, dass das Hindernis zu groß ist, um es automatisch zu umfahren und verlässt sich lieber auf die manuelle Steuerung.

## 4.4 Präsentationsschicht

Die *Präsentationsschicht* dient als Schnittstelle zwischen System und Benutzer. Über diese Schnittstelle, dem *User Interface (UI)*, können Informationen über den Status des Systems angezeigt und dem Nutzer Möglichkeiten zur Steuerung des Systems gegeben werden.

Das Schiffsmodell bietet als *UI* lediglich eine Statusanzeige für die Drahtlosverbindung. Dazu sollen verschiedenfarbige LEDs zum Einsatz kommen, die Auskunft darüber geben ob das Modellschiff in einem WLAN-Netz angemeldet ist, ob eine Verbindung zum Smartphone hergestellt wurde und ob ein Datenaustausch statt findet. In Tabelle 4.2 ist angegeben welche LED-Zustände welchem Verbindungsstatus entsprechen. Die möglichen Status sind dem Handbuch des eingesetzten WLAN-Moduls entnommen.

| Condition | Red LED | Yellow LED | Green LED |
|---|---|---|---|
| ON solid | Not Associated | | Connected over TCP |
| Fast blink | | Rx/Tx data transfer | No IP address |
| Slow blink | Associated, No Internet | | IP address OK |
| OFF | Associated, Internet OK | | |

Tabelle 4.2: Bedeutung der LED-Zustände [RN09, Seite 4]

Das *User Interface* der Smartphone-Applikation soll dem Nutzer neben Angaben zur Geschwindigkeit und Richtung des Schiffs und dessen Entfernung zu einem Objekt auch die Möglichkeit der manuellen Steuerung des Schiffs bereitstellen. Die Statusinformationen des Schiffes werden für diesen ersten Prototypen durch eine einfache Textausgabe realisiert. Zur manuellen Steuerung des Schiffs werden einfache *Gesten* über den Touchscreen des Smartphones vom System angenommen. Durch Berühren des Bildschirms mit einem Finger und dem anschließende Ziehen dieses (*Slide*) in eine der in Tabelle 4.3 aufgeführten Richtungen kann der Nutzer, entsprechend der ausgeführten Geste das Schiff steuern. Durch zweimaliges, kurzes Antippen (*DoubleTap*) des Bildschirms kann der Nutzer das Schiff sofort zum Stillstand bringen.

Durch längeres berühren des Bildschirms kann die automatische Steuerung des Schiffs zu- bzw. abgeschaltet werden.

| Geste | Befehl |
|---|---|
| Slide nach oben | Geschwindigkeit erhöhen |
| Slide nach unten | Geschwindigkeit verringern |
| Slide nach rechts | Schiff nach rechts lenken |
| Slide nach links | Schiff nach links lenken |
| Doppelt tippen | Schiff stoppen |
| Lang gedrückt halten | autom. Schiffssteuerlogik ein-/ausschalten |

Tabelle 4.3: Gesten zur Schiffsteuerung

Alle Gesten können auf der gesamten Fläche des Touchscreens ausgeführt werden. Durch diese Art der Steuerung wird eine intuitive, leicht zu erlernende Bedienung des Systems geschaffen. Zusätzlich sollen auf dieser Fläche Informationen über den aktuellen Status des Schiffsmodells angezeigt werden.

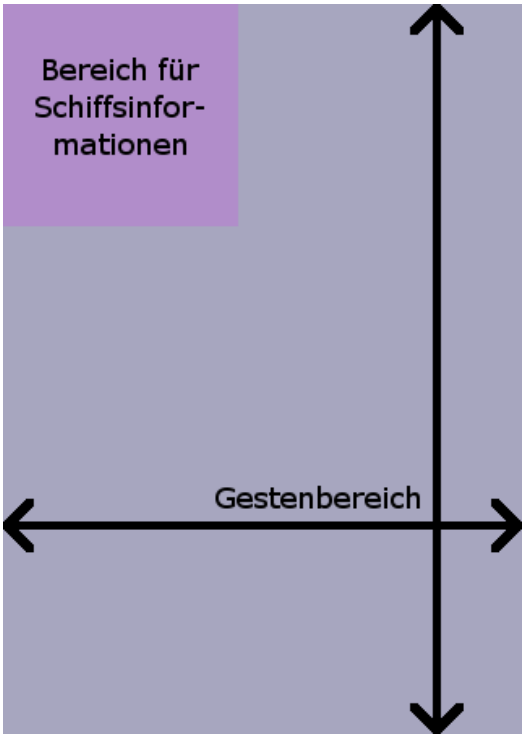


Abbildung 4.6: Entwurf der Benutzeroberfläche des Prototypen

# Kapitel 5

# Implementierung

Im Anschluss an den in Kapitel 4 vorgenommenen Entwurf des Systems wurde dieses prototypisch Umgesetzt. Dieses Kapitel beschreibt dessen Realisierung und geht dazu auf ausgewählte Aspekte der Implementierung ein.

## 5.1 Steuerung des Schiffsmodells

Die Schiffsteuerung besteht zum einen aus den in 3.1.2 aufgeführten Hardwarekomponenten sowie aus einem Steuerungsprogramm, das die Steuerlogik für die Schiffselektronik darstellt.

### 5.1.1 Hardwaretechnische Umsetzung

Zentrales Element innerhalb der Schiffselektronik ist das *Arduino-Board*. An ihm sind alle weiteren Komponenten angeschlossen, die mit Hilfe der eingebetteten Steuerlogik angesteuert werden. Bis auf den Gleichstrommotor, zum Vorantreiben des Schiffs, sind alle Komponenten direkt mit dem *Arduino*-Mikrocontroller verbunden. Der Antriebsmotor wird vom Controller über das *Ardumotor-Shield* angesprochen, da dieser Motoren nicht direkt ansteuern kann. Über die an das WLAN-Modul angeschlossenen LEDs wird der Status der Drahtlosverbindung angezeigt. Wie die einzelnen Komponenten im Detail an das Arduino angeschlossen sind kann Abbildung 5.1 entnom-

men werden[1].

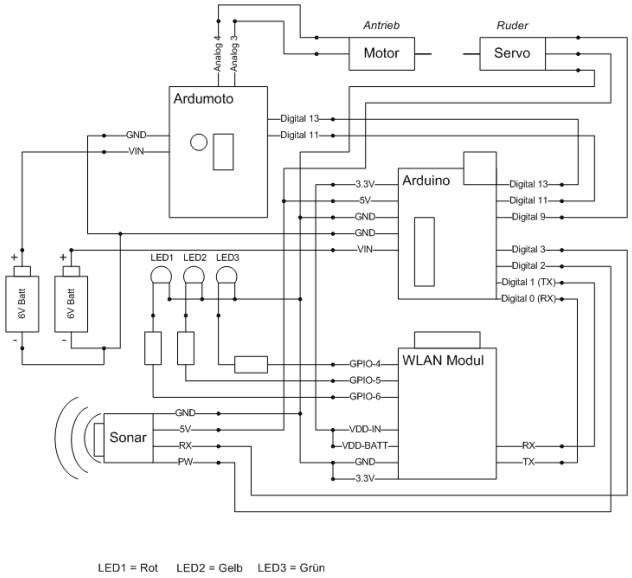


Abbildung 5.1: Schaltskizze der Schiffselektronik

### 5.1.2 Steuerlogik

Das Steuerungsprogramm wurde mit der Arduino IDE[2] in der Version 0017, die frei von der Webseite[3] des Arduino Projekts bezogen werden kann entwickelt. Die Steuerlogik besteht im Wesentlichen aus den zwei Hauptfunktionen *setup* und *loop* sowie aus einer Reihe von Hilfsfunktionen. Die *setup*-Funktion wird vom Arduino nur einmal zur Initialisierung der Hardware aufgerufen. Die *loop*-Funktion hingegen wird zyklisch vom Arduino ausgeführt. Alle darin befindlichen Befehle werden sequenziell ausgeführt und definieren so den Programmablauf, der in Abbildung 4.4 für das Schiffsmodell spezifiziert wurde.

#### Daten empfangen und Senden

Die Datenübertragung findet über die serielle Schnittstelle des Arduino statt, die in jedem Durchlauf der *loop*-Funktion auf eingegangene Daten mit der

[1]Eine größere Version der Skizze ist im Anhang A.2 zu finden.
[2]Integrated Development Environment
[3]http://www.Arduino.cc

Funktion *getRemoteData* überprüft wird. Sind Daten verfügbar, wird zunächst geprüft ob das erste empfangene Byte dem *Start-Byte*, des in 4.2.1 spezifizierten Datenrahmens entspricht. Ist dies nicht der Fall, wird in gleicher Weise mit dem folgendem Byte im nächsten Durchlauf der *loop*-Funktion verfahren. Erkennt das System den Beginn eines solchen Rahmens, ließt es die restlichen Bytes des Rahmens und speichert diese zur späteren Weiterverarbeitung.

```
1  boolean getRemoteData(){
2    int availableBytes = Serial.available();
3    if(availableBytes >= FRAME_LENGTH){
4      byte start = (byte)Serial.read();
5      if(start == DATA_START){
6        receivedData[POS_START_BYTE] = start;
7        receivedData[POS_TYPE_BYTE] = (byte)Serial.read();
8        receivedData[POS_DATA_BYTE] = (byte)Serial.read();
9        receivedData[POS_STOP_BYTE] = (byte)Serial.read();
10       return true;
11     }
12   }
13   return false;
14 }
```

Listing 5.1: Empfang von Daten beim Schiffsmodell

Um welchen Datentyp es sich beim empfangenen Rahmen handelt wird durch Vergleich des *Type-Bytes* mit Konstanten für die einzelnen Typen ermittelt. Die in Tabelle 5.1 aufgeführten Konstantenwerte sind dazu Systemweit eindeutig.

Zum Versand der Sensordaten an das Shmartphone, werden die Nutzdaten in einem Datenrahmen verpackt und per *Serial.write()* versendet. Listing 5.2 zeigt wie dies im Programmcode umgesetzt wurde.

```
1  void sendSensorData(int data){
2    // define the frame
3    byte frame[FRAME_LENGTH];
4    frame[POS_START_BYTE] = (byte)DATA_START;
5    frame[POS_TYPE_BYTE] = (byte)DATA_TYPE_SENSOR;
6    frame[POS_DATA_BYTE] = (byte)data;
7    frame[POS_STOP_BYTE] = (byte)DATA_END;
8    Serial.write(frame, FRAME_LENGTH);
9  }
```

Listing 5.2: Senden von Sensordaten

| Konstante | Wert |
|---|---|
| DATA_START | 255 |
| DATA_EMPTY | 0 |
| DATA_END | 0 |
| DATA_TYPE_ALIVE | 1 |
| DATA_TYPE_SENSOR | 2 |
| DATA_TYPE_CONTROL_SPEED_FORWARD | 3 |
| DATA_TYPE_CONTROL_SPEED_BACKWARD | 4 |
| DATA_TYPE_CONTROL_DIRECTION | 5 |
| DATA_TYPE_INIT_RANGING | 6 |
| DATA_TYPE_STOP_RANGING | 7 |

Tabelle 5.1: Systemweit eindeutige Konstanten

### Ermitteln der Entfernung zu einem Objekt

Sensordaten werden direkt vom entsprechenden Pin des Arduino mit der Funktion *pulseIn* gelesen. Die erhaltenen Daten entsprechen der Entfernung des Schiffs zu einem Objekt in Inch und werden direkt in Zentimeter umgerechnet. Da der Datenrahmen für die Übermittlung der Distanz nur ein Byte vorsieht, ist der durch das System maximal erkennbare Abstand auf 255cm begrenzt. Da dieser Wert aber bereits für das *Start-Byte* des Datenrahmens reserviert ist, wird die maximal erkennbare Entfernung auf 250cm festgelegt. Diese Funktionalität wird durch die in Listing 5.3 aufgeführte Funktion *getSensorData* umgesetzt.

```
1  const int MAX_RANGE = 250; //max range in cm
2  [...]
3  int getSensorData(int pin){
4    int pulse = pulseIn(pin, HIGH);
5    //147uS per inch
6    int inches = pulse/147;
7    //change inches to centimetres
8    int cm = inches * 2.54;
9    //limit the distance to 250 cm
10   if(cm > MAX_RANGE){
11     cm = MAX_RANGE;
12   }
13   return cm;
14 }
```

Listing 5.3: Ermitteln und Umrechnen der Sensordaten in Zentimeter

### Ansteuerung des Antriebs

Antriebsmotor und Schiffsruder werden durch die Funktionen *setShipSpeed* und *setSteeringAngle* realisiert. Das Ruder wird mit Hilfe der Servo bibliothek angesteuert, indem ein Winkelmaß für die Ruderstellung an die *write* Methode des *Servo-Objekts* übergeben wird. Die Richtung der Schiffsbewegung wir über das Setzen des digitalen *motorDirectionPin* auf HIGH oder LOW angegeben, bevor die eigentliche Geschwindigkeit per *analogWrite* gesetzt wird.

```
1  void setSteeringAngle(){
2    // just set the angle of the oar
3    steeringServo.write(steeringAngle);
4  }
5  [...]
6  void setShipSpeed(int value, boolean isBackward){
7
8    // choose direction of propeller
9    int d = HIGH;
10   if(isBackward){
11     d = LOW;
12   }
13   // set direction
14   digitalWrite(motorDirectionPin, d);
15   // set speed value
16   analogWrite(motorSpeedPin, value);
17 }
```

Listing 5.4: Ansteuerung der Aktoren für Geschwindigkeit und Richtung

## 5.2 Smartphone-Applikation zur Fernsteuerung

An dieser Stelle soll nun auf die softwaretechnische Umsetzung der Smartphone-Applikation eingegangen werden. Dazu wird im Folgenden die Entwicklungsumgebung, mit der sie Entwickelt wurde vorgestellt. Darauf folgt die Beschreibung der Projektstruktur sowie die Beschreibung ausgewählter Aspekte der Implementierung der Daten-, Anwendungs- und Präsentationsschicht.

### 5.2.1 Entwicklungsumgebung

Zur Entwicklung der Sorftware wurde die Eclipse IDE[4] in der Version 3.5 mit dem Android Development Toolkit (ADT) in der Version 0.9 verwendet. Als mobiles Endgerät kam das HTC Hero zum Einsatz, das vom Hersteller mit der Android OS Version 1.5 geliefert wurde.

Da das HTC Hero werkseitg nicht mit WLAN Netzten im Ad-hoc-Modus arbeiten kann musste für den drahtlosen Datenaustausch zwischen Schiff und Telefon die Infrastrukturarchitektur genutzt werden.

### 5.2.2 Projektstruktur

Das Projekt wurde zur Abgrenzung der einzelnen Komponenten in entsprechende *Packages* unterteilt.

- shipremote
- shipremote.ai
- shipremote.ai.base
- shipremote.communication
- shipremote.control
- shipremote.ui

Diese *Packages* enthalten jeweils die Klassen, die die Funktionalitäten der jeweiligen Komponente realisieren. Eine Übersicht der Klassen und der Zugehörigen *Packages* können dem Anhang A.1 entnommen werden.

### 5.2.3 Datenschicht

Basis dieser Schicht ist die *TCPClient* Klasse, die mit Hilfe eines *Sockets* Daten an das Schiffsmodell senden und Daten von ihm empfangen kann.

[4]Integrated Development Environment

Um die Applikation beim warten auf Daten nicht zu blockieren wurde das "Horchen" innerhalb eines gesonderten Threads vorgenommen. Dieser wird mit der Methode *listen()* gestartet und prüft zyklisch ob Daten auf dem Übertragungskanal vorhanden sind.

```
1  public void run() {
2    while (this.isListening) {
3      byte[] receivedData;
4      try {
5        receivedData = this.readData();
6        if (receivedData != null) {
7          // get the frame bytes
8          final Byte[] shipDataBytes = this.getFrameBytes(receivedData);
9          // convert to ShipData
10         final ShipData shipData = this.bytesToShipData(shipDataBytes);
11         if (shipData != null) {
12           // inform the listener
13           this.informListenerAboutDataReceived(shipData);
14         }
15       }
16     } catch (final IOException e) {
17       this.handleException(e);
18       // stops the thread
19       this.isListening = false;
20     }
21   }
22 }
```

Listing 5.5: Prüfung des Übertragungskanals auf eingehende Daten

Es können sich Objekte, die das *IDataReceiverListener*-Inerface implementieren am *TCPClient* registrieren, welche dann von ihm informiert werden, sobald Daten eingetroffen sind. Diesen Objekten werden die empfangenen Daten in Form eines *ShipData*-Objekts zur Verarbeitung bereit gestellt.

Zum Senden von Daten an das Schiff, die als *ShipData* vorliegen, erfolgt eine Umwandlung mit Hilfe der Methode *shipDataToBytes()* in ein Byte-Array, welches anschließend über den Datenkanal übertragen wird.

```
1  public boolean sendData(ShipData shipCommand) throws IOException {
2    // get byte frame for ShipData
3    byte[] data = this.shipDataToBytes(shipCommand);
4    if (this.outStream != null && data != null) {
5      // send data
6      this.outStream.write(data);
7      return true;
```

```
8     }
9     return false;
10  }
```

Listing 5.6: Senden von Daten an das Modellschiff

### 5.2.4 Anwendungsschicht

Die Klasse *Main* ist die Hauptaktivität der Androidanwendung, die den Lebenszyklus der Anwendung durch die überschriebenen Methoden *onStart()*, *onStop()* und *onRestart()*, der abstrakten *Activity*-Klasse, abbildet.

Hier wird eine Instanz der *ShipController*-Klasse erstelt, welche die Steuerzentrale der Anwendung darstellt. Sie wird sowohl von der UI-Schicht über Änderungen, als auch von der Datenschicht über das Eintreffen von Daten informiert, da sie an beiden als Observer registriert ist (siehe Listing 5.7 ). Sie verwaltet den gesamten Datenaustausch zwischen diesen Schichten.

```
1  public ShipController(final GesturesView gestureView,
2      final INetworkClient tcpClient, final IShipAI shipAI) {
3    this.gestureView = gestureView;
4    this.gestureView.addSlideListener(this);
5    this.tcpClient = tcpClient;
6    this.tcpClient.addDataReceiverListener(this);
7    this.shipAI = shipAI;
8    this.speedControl = new SpeedControl();
9    this.steeringControl = new SteeringControl();
10   this.aliveTimer = new Timer();
11 }
```

Listing 5.7: Constructor der *ShipController*-Klasse

#### Benachrichtigung bei eingegangenen Schiffsdaten

Wie bereits in 5.2.3 beschrieben erfolgt die Prüfung des Datenkanals auf Daten innerhalb eines separaten Threads, der über die *dataReceived()* Methode den *ShipController* informiert, dass Daten bereitstehen. Das wird mit Hilfe einer Handler-Message threadübergreifend und threadsicher vorgenommen, die vom Conroller ausgewertet wird um so im Anschluss die entsprechenden Vorgänge zur Verarbeitung der Schiffsdaten einzuleiten.

```
1  public synchronized void dataReceived(ShipData shipData) {
2    // called from listener thread of INetworkClient class
3    if (!this.isUserInputAvailable) {
4      // set the receivedShipData property for later processing
5      this.receivedShipData = shipData;
6      // send message to Main-Thread(UI-Thread) to process data
7      this.sendEmptyMessage(ShipController.SHIPDATARECEIVED);
8    }
9  }
10 [...]
11 public void handleMessage(final Message msg) {
12   switch (msg.what) {
13     case SHIPDATARECEIVED:
14       // process received sensor data
15       this.computeReceivedData();
16       this.updateUI();
17       break;
18     case IAMALIVE:
19       // tell the ship im still there
20       this.sendCommand(new ShipData(ShipDataType.ALIVE, 0));
21       break;
22     case ShipAIProcessor.DATA_PROCESSED_BY_AI:
23       // the ship ai finished the processing
24       if (this.updateSpeedAndSteeringAngleWithAIDecisionData()) {
25         this.sendCommandsFromAIToShip();
26       }
27       break;
28   }
29   super.handleMessage(msg);
30 }
```

Listing 5.8: Benachrichtigung des *ShipControllers* über eingegangene Daten mit Hilfe vom Messages

### Generieren von Steuerbefehlen

Das Generieren von Steuerbefehlen geschieht an zwei Stellen des Systems: Einmal nachdem eine *Geste* durch den Nutzer beendet wurde und nach der Verarbeitung der Sensordaten durch die *Schiffslogik (KI)*.

**Nach Gesten** Während der Nutzer eine Geste auf dem Display vollführt, erhält der Controller entsprechende Daten, die ihm zur Änderung der Geschwindigkeits- bzw. Richtungswerte veranlasst. Ist die Geste beendet sendet er, je nach dem

welche Geste zuletzt ausgeführt wurde, die aktuellen Daten an das Modellschiff.

```
1  public void onSlide(final SlideDirection direction, final int value) {
2    if (this.verticalSlide(direction)) {
3      // set the speed for the ship
4      this.setSpeedByDirection(direction);
5      this.lastSlideCommandType = this.getCurrentSpeedDirectionType();
6    } else if (this.horizontalSlide(direction)) {
7      // set the steering angle for the ship
8      this.setSteeringAngleByDirection(direction);
9      this.lastSlideCommandType = ShipDataType.DIRECTION;
10   } else {
11     // not a valid gesture
12     this.onSlideGestureAbord();
13     return;
14   }
15   // update the UI
16   this.updateUI();
17 }
18 [...]
19 public void onSlideEnd() {
20   // get the command value
21   final int value = this.getShipDataValueByLastComandType();
22   final ShipData shipCommand = new ShipData(this.lastSlideCommandType, value);
23   // send the data
24   this.sendCommand(shipCommand);
25   this.isUserInputAvailable = false;
26 }
```

Listing 5.9: Generieren von Steuerbefehlen - Gesten

**Nach Verarbeitung von Schiffsdaten** Nach der Verarbeitung der Schiffsdaten durch die KI, werden *SpeedControl* und *SteeringControl* mit den Daten aus der von der KI gelieferten Entscheidung zur Steuerung des Schiffes aktualisiert und anschließend an das Schiff gesendet. Wie des im Detail Implementiert ist, wird im Folgenden dargestellt.

### Einbindung der Steuerlogik (KI)

Mit Hilfe der Interfaces *IShipAI* und *IShipAIDecision* können verschiedene Implementierungen von Steuerlogiken in das System eingebunden werden. Die eigentliche Logik implementiert dazu das *IShipAI*-Interface, das nach der

Datenverarbeitung eine Instanz einer *IShipAIDescision*-Implementierung liefert. Über einen *ShipAIProcessor* wir die eingebundene KI-Logik ausgeführt, indem die Interface-Methode *getAIDecision()* des IShipAI-Interfaces aufgerufen wird. Die Logik wird in einem gesonderten Thread ausgeführt, um den restlichen Anwendungsablauf nicht zu blockieren.

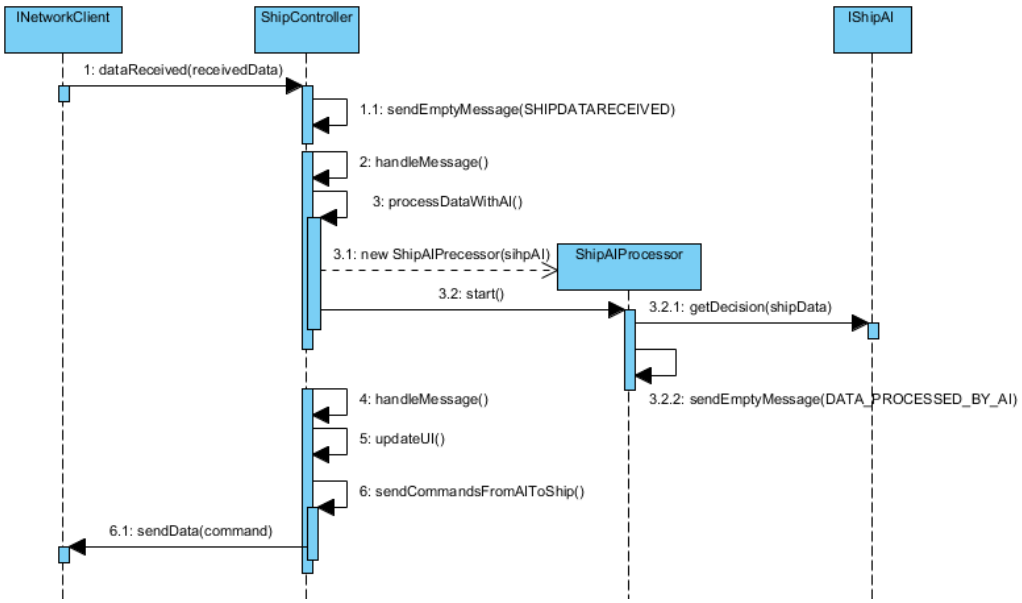


Abbildung 5.2: Ablauf der Verarbeitung der Schiffsdaten durch die KI

Ist die Verarbeitung der empfangenen Daten abgeschlossen, wird der *ShipController* mit Hilfe der *Message DATA_PROCESSED_BY_AI* darüber informiert und innerhalb der *handleMessage()* Methode verarbeitet (siehe Listing 5.8 ). Der *ShipController* aktualisiert nun die Werte für Geschwindigkeit und Richtung mit denen, die durch die Schiffslogik aufgrund der Sensordaten festgelegt wurden und versendet im Anschluss entsprechende Steuerbefehle an das Schiff.

### Exemplarische Implementierung der Schiffslogik

In Anlehnung des Konzepts aus 4.3.3 wurde exemplarisch eine automatische Steuerlogik implementiert. Sie Arbeitet mit Gefahrenstufen, denen entsprechend festgelegte Befehle zur Steuerung des Shiffs hinterlegt sind. Konkret wurde diese Logik in den Klasse *DangerLevelAI* umgesetzt, welche das *IShipAI*-Interface implementiert. Innerhalb der *DangerLevelAI* werden über eine

*DangerLevelFactory* Klasse vier verschiedene Gefahrenstufen (*DangerLevel*) definiert. Für jedes *DangerLevel* ist ein Bereich sowie Steuerbefehle in Form eines *DangerLevelAIDecision*-Onjekts für das Schiff hinterlegt, die vom ihm ausgeführt werden sollen, sobald ein Objekt im definierten Bereich der Gefahrenstufe entdeckt wird. Wie die Gefahrenstufen im einzelnen Definiert sind kann Tabelle 5.2 entnommen werden. Negative Geschwindigkeit bedeutet hier, dass das Schiff angewiesen wird rückwärts zu fahren. Die Klasse *DangerLevelAIDecision* stellt hier die Entscheidung der Steuerlogik dar und somit eine Implementierung des *IShipAIDecision*-Interfaces.

```
1  public class DangerLevelAI implements IShipAI {
2  [...]
3    private void createDangerLevels() {
4      this.dangerLevels = new ArrayList<DangerLevel>();
5      // create all available danger levels from the factory
6      for (int i = 1; i <= DangerLevelFactory.MAX_DANGERLEVEL; i++) {
7        this.dangerLevels.add(DangerLevelFactory.getDangerLevel(i));
8      }
9    }
10 [...]
11 }
```

Listing 5.10: Definition der Gefahrenstufen (*DangerLevel*)

| Gefahrenstufe | Breich | Manöver |
|---|---|---|
| Level 1 | 2,0m bis 2,5m | Geschwindigkeit: 100; Ruderstellung: 90° |
| Level 2 | 1,5m bis 2,0m | Geschwindigkeit: 100; Ruderstellung 60° |
| Level 3 | 1,0m bis 1,5m | Geschwindigkeit: 60; Ruderstellung: 45° |
| Level 4 | 0,0m bis 1,0m | Geschwindigkeit: -100; Ruderstellung: 90° |

Tabelle 5.2: Definierte Gefahrenstufen (vgl. Tabelle 4.1 )

Wird an der Klasse *DangerLevelAI*, die durch das Interface *IShipAI* definierte Methode *getDesision()* aufgerufen, wird anhand der Objektentfernung ermittelt welche Gefahrenstufe vorliegt. Die Objektentfernung kann aus dem übergebenen *ShipState*-Objekt ermittelt werden. Anschließend werden die im *DangerLevel* definierte *DangerLevelAIDecision* zurückgegeben. In Listing 5.11 ist zu sehen wie dies Programmiertechnisch umgesetzt wurde.

```
1  public class DangerLevelAI implements IShipAI {
2  [...]
3    public IShipAIDecision getDecision(ShipState data) {
```

```
4       for (DangerLevel dangerLevel : this.dangerLevels) {
5         if (dangerLevel.getLevelRange().isInRange(data.getObjectDistance())) {
6           return dangerLevel.getDecision(data);
7         }
8       }
9       return DangerLevel.getDefaultDecision(data);
10    }
11  [...]
12  }
```

Listing 5.11: Ermitteln der aktuellen Gefahrenstufe und Rückgabe der entsprechenden Befehlsdaten.

Diese Schiffssteuerlogik ist standardmäßig deaktiviert, kann aber im Betrieb durch langes Drücken (*LongPress*) auf den Bildschirm des Smartphones zugeschaltet bzw. abgeschaltet werden.

### 5.2.5 Präsentationsschicht

Die Schnitstelle zum Benutzer des Smartphones wird durch das *GesturesView* gebildet, das speziell zum Darstellen von Text und Erkennen von Gesten erstellt wurde. Dazu wurde diese Klasse von der *TextView*-Klasse des Android Systems abgeleitet und zusätzlich eine Gestenerkennung implementiert. Es können so die primitiven Gesten *SlideUp* (Slide nach oben), *SlideDown* (Slide nach unten), *SlideRight*(Slide nach rechts) und *SlideLeft* (Slide nach links) erkannt werden.

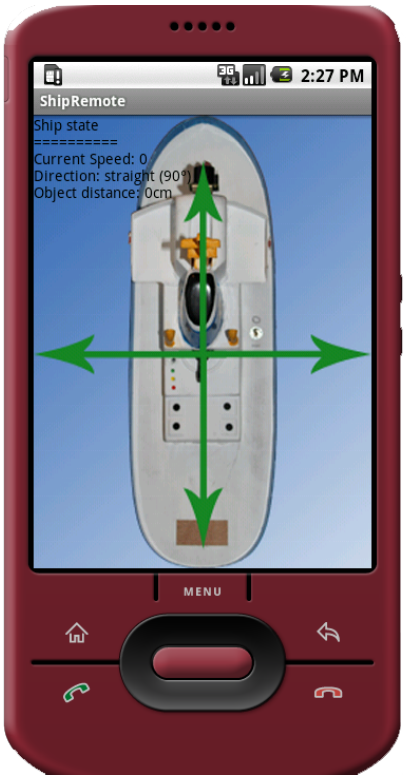


Abbildung 5.3: Benutzeroberfläche des Prototypen

Berührt der Nutzer den Bildschirm mit dem Finger, wird dieser Punkt als Startpunkt der Geste festgelegt. Bewegt er nun den Finger weiter auf dem Bildschirm, wird die Differenz zwischen Startpunkt und der aktuellen Fingerposition (*Slide-Wert*) Berechnet, wobei ausschließlich zwischen horizontaler und vertikaler Bewegungsrichtung unterschieden wird. Das geschieht durch die Ermittlung ob der aktuelle Punkt sich oberhalb oder unterhalb des Startpunktes befindet. Mit einem Schwellwert (Threshold) werden horizontaler und vertikaler Bereich definiert, innerhalb dessen eine Fingerbewegung als Geste erkannt werden kann. Der *Slide-Wert* wird anschließend an die registrierten *Listener* weitergegeben.

```
1 private void onActionDown(final MotionEvent event) {
2   // set start point of gesture
3   this.gestureStart = this.getScreenPoint(event);
4   this.isGestureAbord = false;
5   // inform observer
6   this.informObserverAboutActionDown();
7 }
8 [...]
```

```
 9  private void onActionMove( MotionEvent event) {
10    if (this.gestureStart == null) {
11      return;
12    }
13    // get current point
14     Point currentPoint = this.getScreenPoint(event);
15    // calculate difference between start point and current point
16     Point diffPoint = this.getDifferencePoint(this.gestureStart,
17        currentPoint);
18    // define moving areas
19     Bounds bounds = this.getSlideBounds();
20    if (this.pointMatchesThresold(diffPoint)) {
21      this.currentSlideValue = 0;
22      // detect the slide direction and value
23      this.slideDirection = this.getSlideDirectionWithinBounds(currentPoint,
24          diffPoint, bounds);
25      if (this.slideDirection == SlideDirection.NONE) {
26        this.onGestureAbord();
27        return;
28      }
29      this.currentSlideValue = this.getValueForSlideDirection(diffPoint);
30      // inform the observer if value changed
31      this.informSlideListener();
32      this.lastValue = this.currentSlideValue;
33    }
34  }
```

Listing 5.12: Ermittlung der Gesten

Zusätzlich zu der Erkennung der Slide-Gesten werden zwei weitere Gesten erkannt, die vom *ShipController* verarbeitet werden. Zum einen das doppelte Tippen (*DoubleTap*) auf den Bildschirm und zum anderen langes Drücken (*LongPress*) auf das Display. *DoubleTap* ermöglicht es das Schiff anzuhalten und mit *LongPress* kann die autonome Shiffssteuerung zugeschaltet bzw. abgeschaltet werden.

# Kapitel 6

# Evaluierung und Demonstration

In diesem Kapitel sollen nun die Ergebnisse des im Rahmen der Arbeit entstandenen Prototypen zur Shmartphone-gestützten Kollisionsvermeidung anhand der in Kapitel 3 beschriebenen Anforderungen bewertet und vorgestellt werden.

## 6.1 Evaluierung

Die Bewertung des Prototypen gliedert sich in vier Teile, die sich auf die Anforderungen an das System beziehen sowie die Nutzbarkeit für das Einsatzgebiet des Systems betrachten.

### 6.1.1 Benutzeroberfläche

Zur manuellen Steuerung sollte eine intuitiv zu bedienende Benutzeroberfläche geschaffen werden, mit dessen Hilfe der Nutzer das Schiff zum einen Steuern kann und zum anderen Informationen über das Schiff erhält.

Durch den Einsatz des berührungssensitiven Displays des HTC Hero Mobiltelefons konnte eine Benutzeroberfläche geschaffen werden, die sowohl Gesten zur Steuerung des Schiffs annimmt, als auch zur Anzeige der Schiffsdaten für Geschwindigkeit, Fahrtrichtung und Entfernung zu Objekten, die potenziell

eine Gefahr für dieses darstellen können, dient. Die manuelle Steuerung ist auf die sechs einfach zu erlernende Gesten *SlideUp*, *SldeDown*, *SlideRight*, *SlideLeft*, *DoubleTap* und *LongPress* beschränkt, die entsprechende Befehle implizieren. So impliziert ein Slide nach rechts, das sich das Schiff nach dieser Geste nach rechts bewegt. Dies wird zusätzlich durch die Darstellung des Schiffsrumpfs auf dem Display unterstützt. Lediglich die Steuergesten *DoubleTap*, zum Anhalten des Schiffs und *LongPress*, zur Zu- und Abschaltung der autonomen Steuerung, sind nicht sofort ersichtlich und müssen dem Nutzer mitgeteilt werden.

### 6.1.2 Modularität

Aus der Anforderungsanalyse geht hervor, dass das System durch modularen Aufbau Seitens der Hart- und Software eine leichtere Wartung sowie die Erweiterung für andere Anwendungsgebiete ermöglicht werden sollte. Besonders sollte auf der Seite des Smartphones die Steuerlogik (KI) möglichst leicht austauschbar sein.

Modularität auf Seiten der Hardware wurde erreicht, indem entsprechende Hardwaremodule für die einzelnen Aufgaben, Motorsteuerung, Rudersteuerung und Datenerfassung sowie Datenübertragung und -empfang eingesetzt wurden. Jede Komponente kann leicht ausgetauscht werden, da entsprechende Schnittstellen, die es erlauben einfach per Steckverbindung neue Komponenten anzuschließen, vorhanden sind. Unter Umständen könnte für den Einsatz einer neue Komponente eine Anpassung der Systemsoftware des Schiffsmicrokontrollers nötig sein.

Durch die Umsetzung der in Kapitel 4 beschriebenen Architektur ist auch Softwareseitig der Austausch der dort Dargestellten Komponenten mit Hilfe entsprechend vorhandener Schnittstellen leicht umsetzbar. Eine autonome Steuerlogik kann einfach durch die Nutzung der entsprechenden Interfaces in das System eingebunden werden, was exemplarisch durch die Implementierung einer an die in Kapitel 4 entworfenen Steuerlogik gezeigt wurde.

### 6.1.3 Datenübertragung

Die Datenübertragung sollte direkt zwischen Schiff und Smartphone erfolgen und sich dabei nur auf die zur Steuerung des Schiffs notwendigen Daten beschränken um die Zeit zur Interpretation der Daten auf beiden Seiten mög-

lichst gering zu halten. Außerdem sollte das Schiff auf einen Verbindungsabbruch entsprechend Reagieren.

Dadurch, das am HTC Hero die Erkennung von Ad-hoc WLAN-Netzen deaktiviert ist und eine Aktivierung zu aufwendig ist, konnte ein direkter Datenaustausch zwischen Smartphone und Schiffsmodell zu diesem Zeitpunkt nicht realisiert werden. Stattdessen wurde mit Hilfe eines Access Points (AP) ein Infrastrukturnetzt aufgebaut. Dies hat zur Folge, das Smartphone und Modellschiff innerhalb des selben WLAN-Netzes Verfügbar sein müssen, das vom AP gebildet wird.

Durch die Beschränkung auf vier Byte des Datenrahmens zur Datenübertragung mit festgelegten Rahmentypen (siehe 4.2.2) wurde eine kompakte Struktur geschaffen, die Fehlinterpretationen durch das System ausschließt und somit maßgeblich zur Stabilität des Systems beiträgt. Durch die Rahmenstruktur können Befehle eindeutig aus dem Datenstrom identifiziert werden und die Typisierung des Rahmens trägt dazu bei, dass die Nutzdaten eindeutig einer Funktion zugeordnet werden können.

Auf einen Verbindungsabbruch reagiert das Schiff indem es selbstständig anhält. Treffen nach einer bestimmten Zeit keine Befehle mehr am Schiff ein, wird davon ausgegangen, das die Verbindung getrennt wurde und das Schiff stoppt. Dadurch wird verhindert, dass das Schiff unkontrolliert weiter fährt und eventuell Schaden nimmt oder abhanden kommt, da es keine Steuerbefehle entgegen nehmen kann.

### 6.1.4 Funktionalität in Hinblick auf das Einsatzgebiet

Im Rahmen des Einsatzgebietes (siehe 3.1.1) soll das Schiff mit Hilfe einer im Shmartphone integrierten autonomen Steuerung (KI) autonom durch einen Hindernisparcours manövrieren. Dazu muss es in der Lage sein Hindernisse zu erkennen und entsprechend der Situation selbstständig entscheiden, wie eine Kollision vermieden werden kann.

Zur Hinderniserkennung wurde das Schiffsmodell mit einem Ultraschallsensor ausgestattet, dessen Daten sekündlich an das Shmartphone übertragen werden. Hierzu wird ein Bereich von bis zu 2,5 Metern vom Bug aus in Fahrtrichtung überwacht. Die erhaltenen Daten werden durch die Schiffssteuerlogik auf dem Smartphone verarbeitet und entsprechend der Situation Steuerbefehle generiert und an das Schiff zur Umsetzung gesandt.

Der Ultraschallsensor liefert zuverlässig Daten zur Entfernung des Schiffs zu einem Objekt. Ist kein Objekt im überwachten Bereich, liefert er die maximale Reichweite des Überwachten Bereichs (2,5m) zurück. Der gesamte Bereich ist durch die automatische Steuerlogik logisch in vier Bereiche, sogenannten Gefahrenstufen unterteilt. Jeder Gefahrenstufe sind entsprechend der potentiellen Kollisionsgefahr Befehle hinterlegt um einen Zusammenstoß mit einem Hindernis zu vermeiden. Wird beispielsweise ein Objekt im Bereich zwischen 1,5m und 1,0m entdeckt weißt die Steuerlogik das Schiff an die Geschwindigkeit auf 60 zu drosseln und das Ruder in einem Winkel von 45° auszurichten. Analog geschieht dies für alle in Tabelle 5.2 aufgeführten Gefahrenstufen. Die autonome Steuerung des Schiffs kann durch längeres berühren des Bildschirms sowohl zugeschaltet, als auch abgeschaltet werden.

Ist die autonome Steuerung abgeschaltet, dient zur Steuerung des Schiffs ausschließlich die manuelle Steuerung um Kollisionen mit Hindernissen zu vermeiden. Bei ausgiebigen Tests wurde festgestellt, dass der Antriebsmotor erst ab einer bestimmten Geschwindigkeit genügend Kraft aufbringt um selbständig die Welle, mit der die Schiffsschraube verbunden ist, anzutreiben. Ist die Welle bereits in Bewegung kann die Geschwindigkeit anschließend unter diesen Schwellwert korrigiert werden. Somit gibt es einen Geschwindigkeitsbereich, in dem das Schiff keine Fahrt aufnimmt.

Weiterhin wurde festgestellt, dass auf Grund der Art, wie die Rudersteuerung technisch zu realisieren war, die möglichen Ruderstellungen nicht den gesamten Bereich von 0° bis 180° abbilden können. Das liegt im wesentlichen an der Übersetzung von Ruderservo zum Ruder, die nur einen Bereich von ca. 90° abdeckt. Dabei handelt es sich um eine starre Verbindung über eine Achse, die das Ruder in die entsprechende Stellung des Servomotors bringt. In Abbildung 6.1 ist schematisch dargestellt wie die Rudersteuerung verbaut wurde und wie groß der tatsächliche Bereich für die Ruderstellung ist.

Auch wenn die manuelle Steuerung den genannten Einschränkungen, die ebenso auch für die autonome Steuerung gelten, unterliegt, ist es dennoch möglich das Schiff im Rahmen dieser präzise zu steuern. Somit erfüllt das System die Anforderungen an die manuellen Steuerung des Schiffsmodells, da sie lediglich dazu ausgelegt war grundlegende Funktionstests durchzuführen und im Fall eines KI-Ausfalls weiter Kontrolle über das Schiff zu behalten.

Innerhalb der Testumgebung kam es zeitweise zu Verzögerungen bei der Datenübertragung, was zur Folge hatte, dass die übertragenen Befehle an das Schiff nicht unmittelbar umgesetzt werden konnten. Unter Umständen kann es hier zu solch langen Latenzen kommen, dass im Gefahrenfall das Schiff

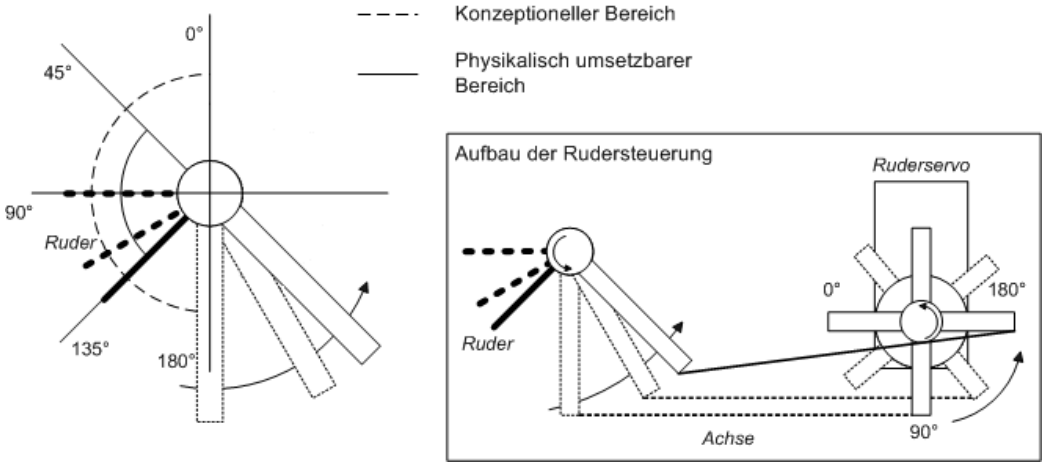


Abbildung 6.1: Konzeptioneller und tatsächlicher Bereich der möglichen Ruderstellungen

nicht rechtzeitig reagieren kann und eine Kollision zu befürchten ist. Auch kam es teilweise zu Verbindungsabbrüchen, deren Ursache nicht klar ermittelt werden konnte. Mögliche Ursache, besonders für die temporär hohen Latenzzeiten, könnten Interferenzen mit anderen WLAN-Netzen im Umfeld sein, die den selben Übertragungskanal nutzen. Nach einem Wechsel des Funkkanals für die Datenübertragung im Netz des Systems, verringerte sich die Häufigkeit der verzögerten Übertragungen. Trotz dieser Anpassung können hohe Latenzzeiten und unvermittelte Verbindungsabbrüche jedoch nie ganz ausgeschlossen werden und müssen beim Betrieb des Systems mit einkalkuliert werden.

## 6.2 Demonstration

Innerhalb dieses Abschnitts soll das entstandene System vorgestellt werden. Zunächst wird der entstandene Schiffsprototyp vorgestellt und anschließend wird mit Hilfe von Fototserien auf die Funktionsweise der manuellen und automatischen Steuerung eingegangen[1].

[1]Auf der beigelegten CD befinden sich entsprechende Videos, in denen die Funktionsweisen der manuellen und der autonomen Steuerung gezeigt werden.

### 6.2.1 Der Prototyp

Im Rahmen dieser Arbeit ist der in Abbildung 6.2 dargestellte Prototyp entstanden. Er besteht aus dem Schiffsmodell eines Schleppers und der Anwendung *ShipRemote* für das HTC Hero Mobiltelefon. Auf Deck des Modells ist der Ultraschallsensor zur Erkennung von Hindernissen angebracht. Daneben sind zur Prüfung der WLAN-Verbindung LEDs im hinteren Bereich des Schornsteins angebracht sowie ein Schalter zum Ein- und Ausschalten des Modells.


Abbildung 6.2: Systemprototyp

Die mobile Anwendung *ShipRemote* ist auf dem Mobiltelefon installiert und kann aus dem Programmmenü gestartet werden. Im Folgenden wird die Steuerung des Schiffs mit Hilfe dieser Anwendung beschrieben.

### 6.2.2 Steuerung durch Gesten

Nachdem die Anwendung *ShipRemote* aus dem Programmenü gestartet wurde wird versucht eine Verbindung zum Schiff aufzubauen. Schlägt dies fehl, erhält der Anwender eine entsprechende Fehlermeldung und das Programm wird automatisch beendet. Wurde erfolgreich eine Verbindung hergestellt kann der Benutzer nun auf dem Bildschirm das Modellschiff aus der Vogelperspektive sowie Informationen über den aktuellen Schiffsstatus in der oberen linken ecke des Bildschirms sehen.


Abbildung 6.3: Start der Applikation *ShipRemote*. Links: Eintrag im Programmmenü, Mitte: Verbindung zum Schiff fehlgeschlagen; Rechts: Hauptbildschirm der Anwendung

Nun kann der Nutzer mit Hilfe von Gesten, die er auf dem Display ausführt das Schiff steuern. Zum Beschleunigen des Schiffes zieht man einen Finger nach oben über das Display (*SlideUp*). Dabei ändert sich der Wert für die Geschwindigkeit auf dem Bildschirm. Je weiter der Nutzer den Finger von der Startposition seiner Geste nach oben zieht umso höher wird die Geschwindigkeit. Wird nun der Finger wieder vom Display genommen sendet die Anwendung den eingestellten Wert an das Schiff, welches nun die Geschwindigkeit erhöht. Die entsprechend entgegengesetzte Geste (*SlideDown*) verringert die Geschwindigkeit. Das Schiff fährt rückwärts, sobald durch verringern der Geschwindigkeit ein Wert kleiner als 0 erreicht wird.

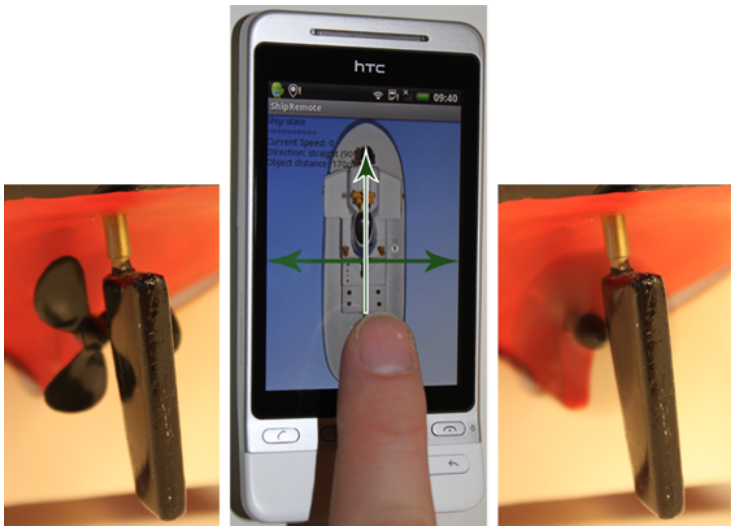


Abbildung 6.4: Beschleunigen des Schiffes mit der Geste *SlideUp*

Um das Schiff zu lenken Zieht man den Finger in die Entsprechende Richtung nach links (*SlideLeft*) oder rechts (*SlideRight*). Der eingestellte Winkel für die Ruderstellung wird analog zur Geschwindigkeitsänderung nach Beendigung der Geste an das Schiff übermittelt.

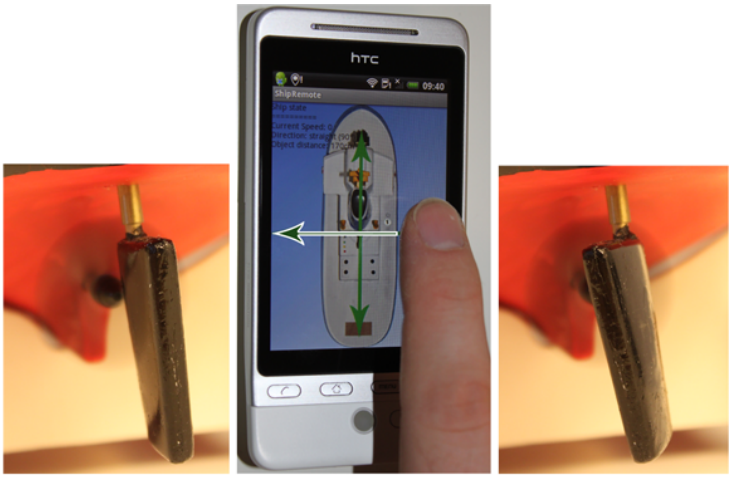


Abbildung 6.5: Das Schiff nach links lenken mit der Geste *SlideLeft*

Das Schiff kann durch doppeltes, kurz aufeinander folgendes Tippen (*Double-Tap*) auf das Display gestoppt werden. Dabei wird die Geschwindigkeit durch die Anwendung auf 0 und das Ruder auf 90° eingestellt.

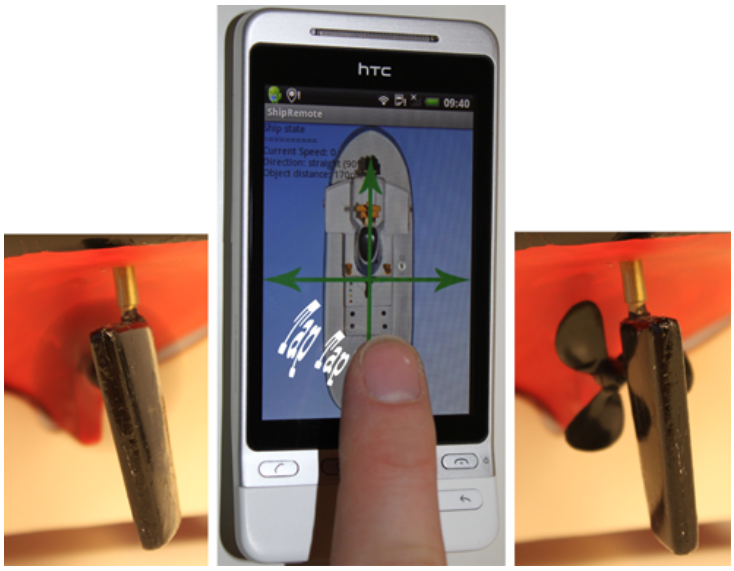

Abbildung 6.6: Das Schiff stoppen mit der Geste (*DoubleTap*)

### 6.2.3 Automatische Steuerung

Die automatische Steuerung ist standardmäßig deaktiviert. Um sie zu aktivieren hält der Anwender einen Finger ca. 2 Sekunden auf dem Display gedrückt. Er erhält eine Mitteilung auf dem Bildschirm das die KI nun aktiviert ist. Mit der selben Geste (*LongPress*) kann der Nutzer die KI auch wieder abschalten.


Abbildung 6.7: Ein- und Ausschalten der automatischen Steuerung durch langes Berühren des Bilschirms (*LongPress*)

Ist die autonome Steuerung aktiviert wird das Schiff die von der Steuerlogik generierte Befehle ausführen. Je nach Implementierung dieser kann das Verhalten des Schiffsmodells variieren. Die hier exemplarisch integrierte Steuerung entscheidet in Abhängigkeit, des durch den Ultraschallsensor gemessenen Abstands zu einem Hindernis welche Steuerbefehle an das Schiff gesendet werden. Je näher ein Hindernis dem Modellschiff kommt, desto größer sind die Bemühungen eine Kollision zu vermeiden. Befindet sich ein Hindernis im Bereich von 2,0m bis 2,5m oder darüber hinaus, fährt das Schiff mit voller Fahrt vorwärts gerade aus. Erst ab einer Entfernung von weniger als 2,0m schlägt das Schiff bei voller Fahrt einen leichten Ausweichkurs nach links ein. Verringert sich der Abstand des Objekts weiter unter 1,5m wird die Geschwindigkeit gedrosselt und stärker nach links gelenkt. Bei einem Abstand von unter 1,0m wird es Kritisch für das Schiff und es gibt volle Kraft zurück bis der Abstand wieder über dieser kritischen Distanz liegt. Ist das Ausweichmanöver erfolgreich, d.h. das Hindernis ist außerhalb aus dem Überwachungsberch, nimmt das Schiff wieder volle Fahrt gerade aus auf. In Abbildung 6.8 ist dieses Verhalten schematisch dargestellt.

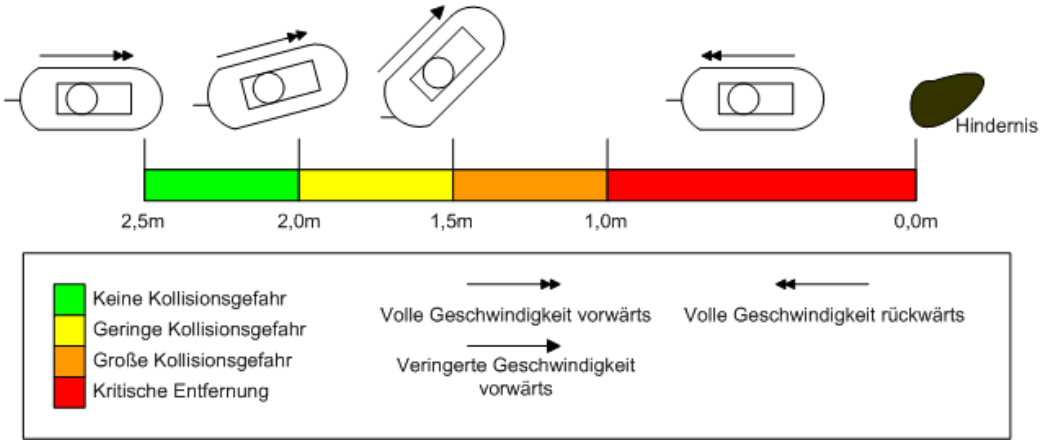


Abbildung 6.8: Schematische Darstellung des Schiffverhaltens bei aktiver autonomer Steuerung

# Kapitel 7

# Zusammenfassung und Ausblick

## 7.1 Zusammenfassung

Ziel dieser Arbeit war es ein prototypisches System zu entwickeln, das es einem Modellschiff ermöglicht autonom Hindernissen auszuweichen. Ein Smartphone sollte als zentrale Steuereinhet dienen, welches eine zu entwickelnde Software zur Auswertung von Sensordaten und zur Steuerung des Schiffes beherbergt. Auch war Aufgabe, die Kommunikation zwischen Schiffsmodell und Mobiltelefon mit Hilfe einer Drahtlostechnologie zu Realisieren.

Zur Lösung dieser Aufgabe wurden im theoretischen Teil dieser Arbeit auf die thematisch zugehörigen Technologien und Geräte eingegangen. Es wurden Smartphones betrachtet, die potentiell geeignete waren und eines für die tatsächliche Umsetzung des Systems gewählt. Auch wurden Drahtlostechnologien vorgestellt und entschieden das WLAN für den Datenaustausch zwischen Schiff und Mobiltelefon dienen sollte. Zur technischen Umsetzung des Prototypen war es weiterhin nötig thematisch auf das Physical Computing einzugehen und dessen Umfeld zu betrachten. Auch wurden bereits existierende Systeme vorgestellt, die autonom ihrem Einsatzgebiet entsprechende Aufgaben erfüllen können.

Im weiteren Verlauf der Arbeit wurde aus den identifizierten Anforderungen das Gesamtsystem entworfen. Hier wurde die Architektur der beiden Teilsysteme Schiffsmodell und Smartphone festgelegt und spezifiziert wie der Datenaustausch zwischen diesen zu realisieren ist. Auch ein Konzept für eine automatische Schiffskontrolle wurde hier erarbeitet.

Darauf folgend wurde dargestellt wie die entworfenen Teile Implementiert wurden, wobei die Umsetzung der Smartphoneanwendung und dem Schiffsmodell getrennt betrachtet wurde. Auf Seiten der mobilen Applikation wurde zudem auch eine, an den Entwurf angelehnte autonome Steuerung implementiert.

Die Funktionstüchtigkeit des Systems wurde während der Evaluierung sowohl für die manuelle, als auch für die automatische Steuerung getestet und bewertet. Dabei kam zum Vorschein, dass es, was die technische Umsetzung der Befehle durch das Schiff zu Ungenauigkeiten kommt. Dennoch kann das System innerhalb dieser Einschränkungen betrieben werden. Die manuelle sowie auch die automatische Steuerung wurden abschließend demonstrativ dargestellt und erläutert wie das System zu bedienen ist.

Ergebnis dieser Arbeit ist ein System, welches mit Hilfe einer Smartphoneapplikation zur manuellen sowie auch zur automatischen Kollisionsvermeidung durch eine entsprechende integrierbare Steuerlogik eingesetzt werden kann. Somit wurde die Aufgabe, eine Smartphone-gesteuerte Schiffskontrolle zur Kollisionsvermeidung zu entwerfen und zu implementieren, erfüllt.

## 7.2 Ausblick

Weiterentwicklungen des Systems sollten in erster Linie versuchen die derzeitige Infrastruktur-Topologie des WLAN-Netzes gegen die Ad-hoc-Topologie zu ersetzen. Somit würde das System von der Notwendigkeit eines Access Points entkoppelt werden, womit sich eine weitaus höhere Flexibilitat des Systems ergäbe, da so zum Betrieb nur noch Schiffsmodell und das Smartphone als Steuereinheit benötigt würden.

Weiterer Ansatzpunkt für eine Erweiterung des Systems stellt die Schnittstelle der Schiffslogik dar. Durch die Einbindung der Steuerlogik auf Ebene des Dateisystems könnte die Modularität weiter erhöht werden. Läge beispielsweise die KI in Form einer jar-Datei vor, könnte diese von der Basisanwendung während der Laufzeit in das System eingebunden oder ausgetauscht werden. So bestünde keine Notwendigkeit mehr KI-Entwicklern den Quellcode der Basisanwendung offenzulegen, da sie lediglich nur noch gegen das Interface der KI-Schnittstelle entwickeln müssten.

Die Steuerlogik selber liegt hier nur in einer sehr simplen Form vor, aber potenziell sind auch komplexere Logiken denkbar. Sie könnten auf vorherige

Werte des Sensors zurückgreifen und anhand eines Werteverlaufs eine Entscheidung treffen oder dynamisch Geschwindigkeit und Richtung des Schiffs je nach Situation anpassen. Der Prototyp nutzt derzeit ausschließlich festgelegte Werte zur automatischen Steuerung.

Wie die Betrachtung verschiedener Smartphones im theoretischen Teil dieser Arbeit gezeigt hat sind andere Smartphone-Platformen ebenso geeignet für den Einsatz als Steuerzentrale für das Schiff. Durch die festgelegten Befehle und Form der Datenübertragung sollte dies auch nicht mit allzuviel Aufwand möglich sein.

# Literaturverzeichnis


[Abd06] Abdalla, Samer: *Standards und Risiken drahtloser Kommunikation Risikoanalyse des IEEE 802.11 Standards (WLAN).* VDM Verlag Dr. Müller e.K., 2006

[Alb08] Alby, Tom: *Das Mobile Web.* Carl Hanser Verlag München, 2008

[Hel08] Hellige, Hans Dieter: *Mensch-Computer-Interface.* transcript Verlag, 2008

[HHC07] Houda Labiod ; Hossam Afifi ; Constantino De Santis: *Wi-Fi Bluetooth ZigBee and WiMax.* Springer, 2007

[HM09] Heiko Mosemann ; Matthias Klose: *Android Anwendungen für das Handy-Betriebssystem erfolgreich programmieren.* Carl Hanser Verlag München Wien, 2009

[Jör02] Jörg F. Wollert: *Das Bluetooth Handbuch -Technologie -Software -Einsatzfelder -Systementwicklung -Wettbewerb.* Franzis Verlag GmbH, 2002

[Köh03] Köhre, Thomas: *Wireless LAN.* Markt+Technik Verlag, 2003

[MD03] Mathias Hein ; Dr. Bernd Maciejewski: *Wireless LAN Funknetzte in der Praxis.* Franzis' Verlag GmbH, 2003

[MJA09] Manuel Odendahl ; Julian Finn ; Alex Wenger: *Arduino - Physical Computing für Bastler, Designer und Geeks.* O'Reilly Verlag GmbH & Co. KG, 2009

[Neh02] Nehmzow, Ulrich: *Mobile Robotik.* Springer Verlag Berlin Heidelberg, 2002

[Pfü04] Pfützenreuter, Torsten: *Intelligentes Missionsanagement für autonome mobile Systeme*, Technische Universität Ilmenau, Diss., 2004. – Erschienen im Cuvillier Verlag Göttingen 2005

[Rec06] Rech, Jörg: *Wireless LANs.* Heise Zeitschriften Verlag GmbH & Co.KG, Hannover, 2006

[Web08] Weber, Wibke: *Kompendium Informationsdesign.* Springer-Verlag Berlin Heidelberg, 2008

# Internetquellen


[App09a] Apple Inc.: *iPhone Dev Center - Apple Developer Connection*. http://developer.apple.com/iphone/index.action, 2009. – [Online; Stand 23. Oktober 2009]

[App09b] Apple Inc.: *iPhone Dev Center: iPhone OS Technology Overview: iPhone OS Technologies*. http://developer.apple.com/iphone/library/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html, 10 2009. – [Online; Stand 23. Oktober 2009]

[App09c] Apple Inc.: *iPhone Developer Program - Apple Developer Connection*. http://developer.apple.com/iPhone/program, 2009. – [Online; Stand 1. Oktober 2009]

[App09d] Apple Inc.: *Mac Dev Center: The Objective-C 2.0 Programming Language: Introduction to The Objective-C 2.0 Programming Language*. http://developer.apple.com/mac/library/DOCUMENTATION/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html, 2009. – [Online; Stand 1. Oktober 2009]

[ard] arduino.cc: *Arduino - HomePage*. http://www.arduino.cc/, . – [Online; Stand 15. Oktober 2009]

[Bun01] Bundesministerium für Bildung und Forschung: *BMBF: DeepC: Entwicklung eines autonomen Unterwasserfahrzeuges wird mit 8 Millionen Mark gefördert*. http://www.bmbf.de/press/342.php, 03 2001. – [Online; Stand 20. November 2009]

[com06] comets-uavs.org: *COMETS Project web site*. http://www.comets-uavs.org/, 09 2006. – [Online; Stand 29. Dezember 2009]

[Ele] Elektronik-Kompendium.de: *IEEE 802.11n*. http://www.elektronik-kompendium.de/sites/net/1102071.htm, . – [Online; Stand 14. Oktober 2009]

[Gooa] Google Inc.: *Open Handset Alliance*. http://www.openhandsetalliance.com/oha_members.html, . – [Online; Stand 3. Oktober 2009]

[Goob] Google Inc.: *User Interface |Android Developers*. http://developer.android.com/guide/topics/ui/index.html, . – [Online; Stand 16. Dezember 2009]

[Gooc] Google Inc.: *What is Android? |Android Developers*. http://developer.android.com/guide/basics/what-is-android.html, . – [Online; Stand 16. Dezember 2009]

[Gre09] Greif, Björn: *Bluetooth-3.0-Spezifikation verabschiedet*. http://www.zdnet.de/news/wirtschaft_investition_hardware_bluetooth_3_0_spezifikation_verabschiedet_story-39001021-41003184-1.htm, April 2009. – [Online; Stand 12. September 2009]

[Ham] Hamburg.de: *Miniatur Wunderland Hamburg Geschichte*. http://www.hamburg.de/miniatur-wunderland/244846/miniatur-wunderland-hinter-den-kulissen.html, . – [Online; Stand 13. November 2009]

[hei07a] heise online: *Apples iPhone ab 29. Juni in den USA erhältlich*. http://www.heise.de/newsticker/Apples-iPhone-ab-29-Juni-in-den-USA-erhaeltlich--/meldung/90566, 06 2007. – [Online; Stand 1. Oktober 2009]

[hei07b] heise online: *Macworld: Das iPhone von Apple gibt es wirklich*. http://www.heise.de/newsticker/Macworld-Das-iPhone-von-Apple-gibt-es-wirklich--/meldung/83454, 01 2007. – [Online; Stand 1. Oktober 2009]

[Ins09] Institute of Electrical and Electronics Engineers, Inc.: *IEEE Ratifies 802.11n, Wireless LAN Specification to Provide Significantly Improved Data Throughput and Range*. http://www.elektronik-kompendium.de/sites/net/1102071.htm, 09 2009. – [Online; Stand 14. Oktober 2009]

[Min] Miniatur Wunderland Hamburg GmbH: *Miniatur Wunderland Hamburg - Modellbau Modelleisenbahn Hamburg*. http://www.miniatur-wunderland.de/, . – [Online; Stand 13. November 2009]

[RN09] Roving Networks, Inc.: *WIFLY GSX RN-131G & RN-134 802.11 b/g wireless LAN Module*. http://www.sparkfun.com/datasheets/Wireless/WiFi/WiFlyGSX-um.pdf, 2009. – [Online; Stand 26. Januar 2010]

[SCH05] SCHIFFBAU INDUSTRIE: *Autonomes Unterwasserfahrzeug "DeepC"*. http://www.schiffundhafen.de/fileadmin/user_upload/Schiffbau_Industrie/si051_3240.pdf, 01 2005. – [Online; Stand 26. Januar 2010]

[Sun08] Sun Microsystems, Inc.: *Sun Announces JVM for iPhone*. http://channelsun.sun.com/video/open-source/java/1631259654/sun+announces+jvm+for+iphone/1685952691, 03 2008. – [Online; Stand 3. Oktober 2009]

[Tec06a] Technische Universität Berlin: *Marvin Mark II Flight*. http://pdv.cs.tu-berlin.de/MARVIN/mark_ii_frameset_flight.html, 04 2006. – [Online; Stand 29. Dezember 2009]

[Tec06b] Technische Universität Berlin: *Marvin Mark II System Description*. http://pdv.cs.tu-berlin.de/MARVIN/mark_ii_frameset_system.html, 04 2006. – [Online; Stand 29. Dezember 2009]

[Tec07] Technische Universität Berlin: *Marvin Mark II*. http://pdv.cs.tu-berlin.de/MARVIN/mark_ii_frameset_introduction.html, 12 2007. – [Online; Stand 29. Dezember 2009]

[TNS08] TNS Infratest Holding GmbH & Co. KG: *Fehlende Begeisterung für die Nutzung von Smartphones*. http://www.tns-infratest.com/presse/pdf/Presse/20080304_TNS_Infratest_GTI_200708.pdf, 04 2008. – [Online; Stand 9. Juli 2009]

[Wik09a] Wikipedia: *Physical computing — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Physical_computing&oldid=314762246, 2009. – [Online; Stand 16. Oktober 2009]

[Wik09b] Wikipedia: *Smartphone — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Smartphone&oldid=65043546, 2009. – [Online; Stand 1. Oktober 2009]

[Wik10] Wikipedia: *Android (Betriebssystem) — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Android_(Betriebssystem)&oldid=69469508, 2010. – [Online; Stand 19. Januar 2010]

# Bildquellen


[Appa] Apple Inc.: *photos-hardware-05-20090608*. http://images.apple.com/iphone/gallery/images/photos-hardware-05-20090608.jpg, . – [Online; Stand 4. Oktober 2009]

[Appb] Apple Inc.: *SystemLayers*. http://developer.apple.com/iphone/library/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Art/SystemLayers.jpg, . – [Online; Stand 23. Oktober 2009]

[Gooa] Google Inc.: *g1-hpp-de*. http://www.google.de/mobile/images/g1-hpp-de.jpg, . – [Online; Stand 4. Oktober 2009]

[Goob] Google Inc.: *system-architecture*. http://developer.android.com/images/system-architecture.jpg, . – [Online; Stand 4. Oktober 2009]

[Spa] SparkFun Electronics: *SparkFun Electronics*. http://www.sparkfun.com, . – [Online; Stand 10. Oktober 2009]

[Teca] Technische Universität Berlin: *Heli1_flying*. http://pdv.cs.tu-berlin.de/MARVIN/MarkII-Pics/thumbs/Heli1_flying_thumb.jpg, . – [Online; Stand 29. Dezember 2009]

[Tecb] Technische Universität Berlin: *Helipoy_eight*. http://pdv.cs.tu-berlin.de/MARVIN/MarkII-Pics/Helipoy_eight.gif, . – [Online; Stand 29. Dezember 2009]

# Abbildungsverzeichnis

# Listings

# Tabellenverzeichnis

# Anhang A

# Diagramme

# A.1 UML-Diagramme

## A.1.1 Package shipremote - Komponentenübersicht

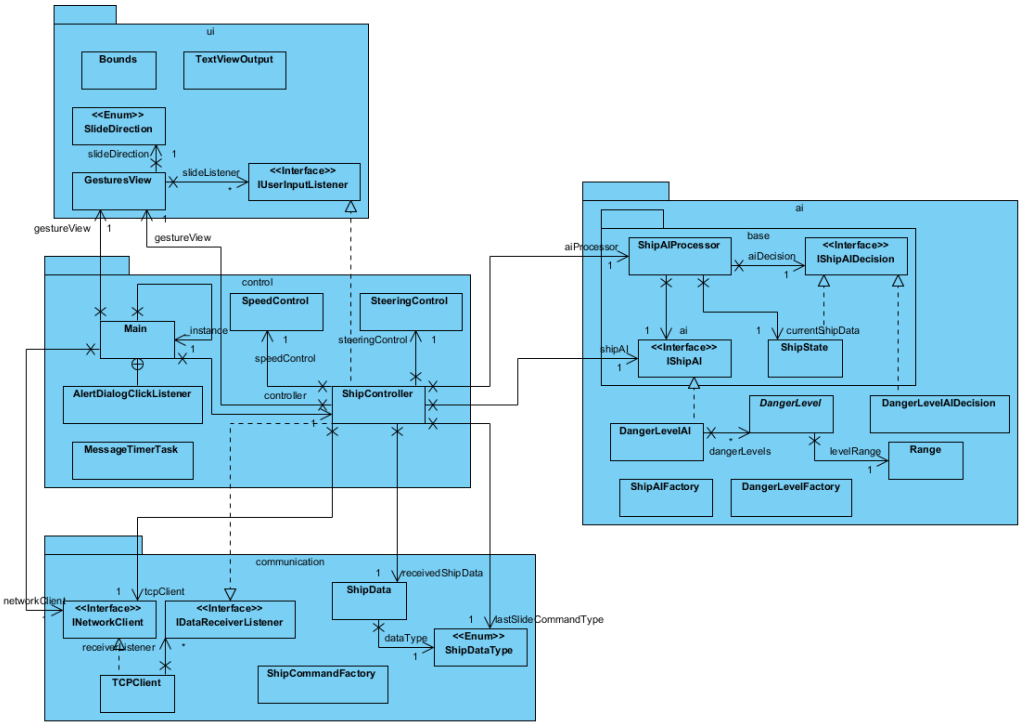


Abbildung A.1: Kalssendiagramm des Packages shipremote

### A.1.2 Package shipremote.ui

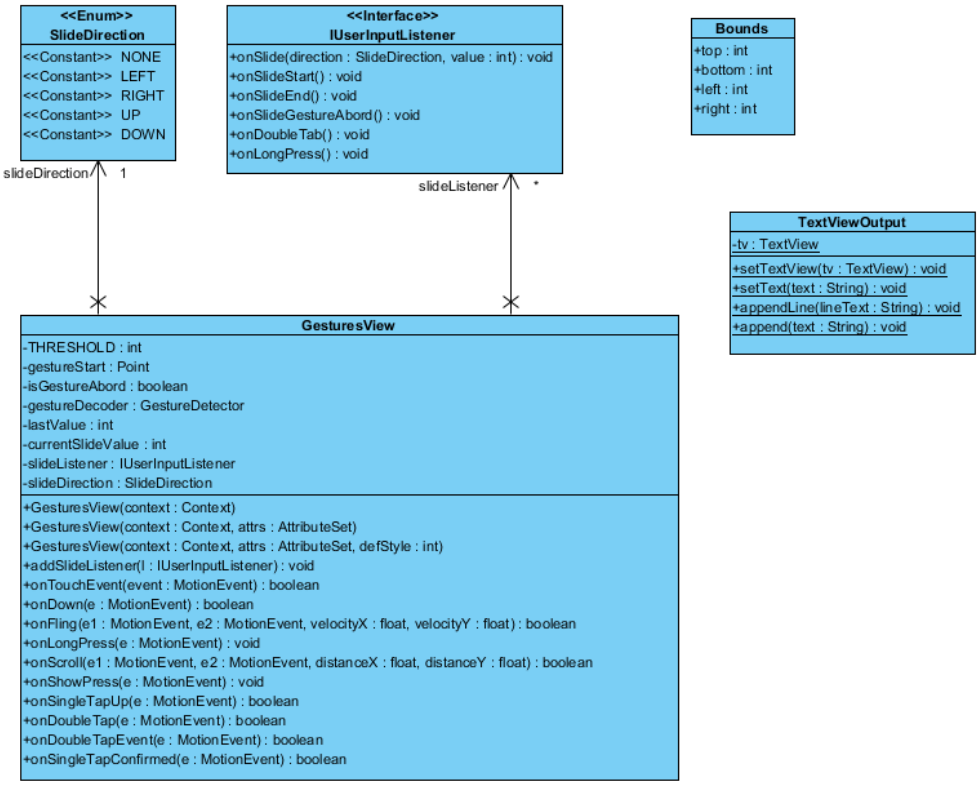


Abbildung A.2: Kalssendiagramm des Packages shipremote.ui

### A.1.3 Package shipremote.control

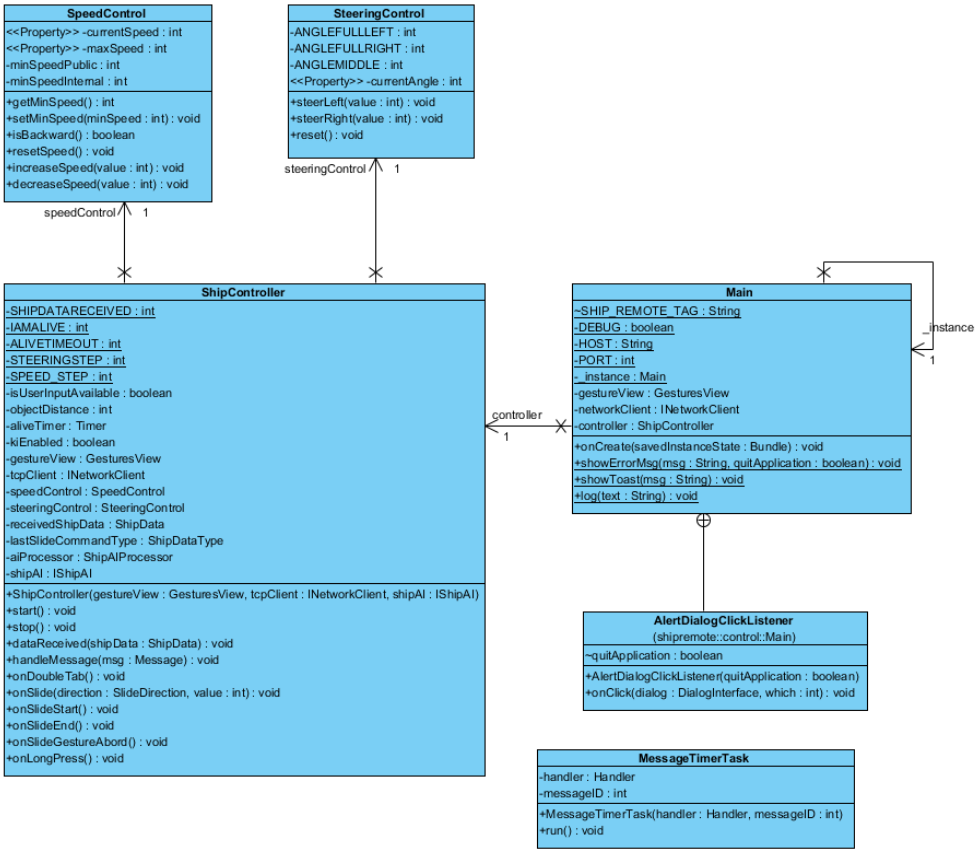


Abbildung A.3: Kalssendiagramm des Packages shipremote.control

### A.1.4 Package shipremote.ai.base

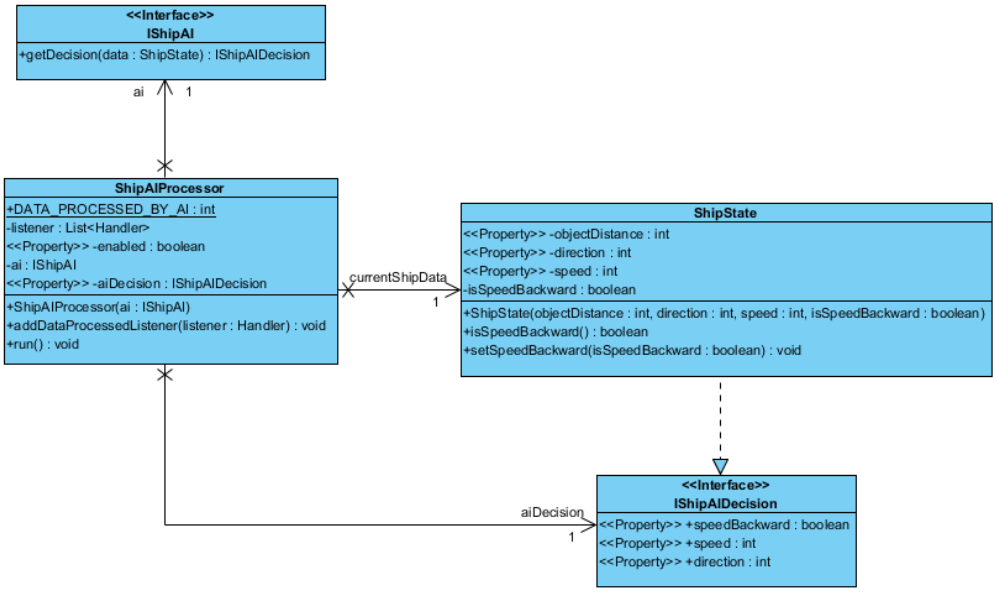


Abbildung A.4: Kalssendiagramm des Packages shipremote.ai.base

### A.1.5 Package shipremote.ai

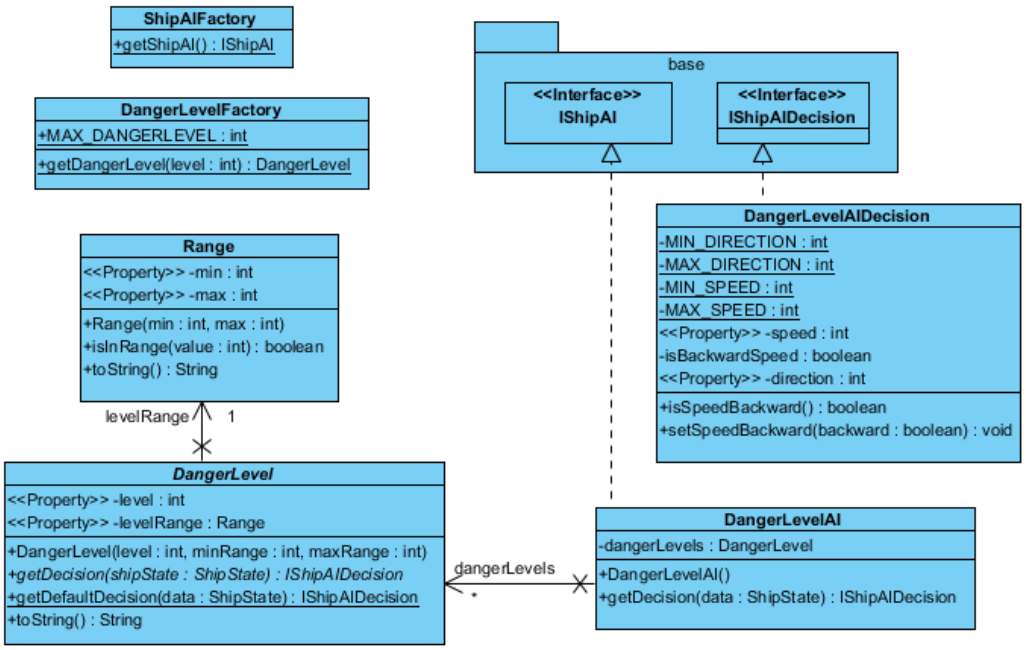


Abbildung A.5: Kalssendiagramm des Packages shipremote.ai

### A.1.6 Package shipremote.communication

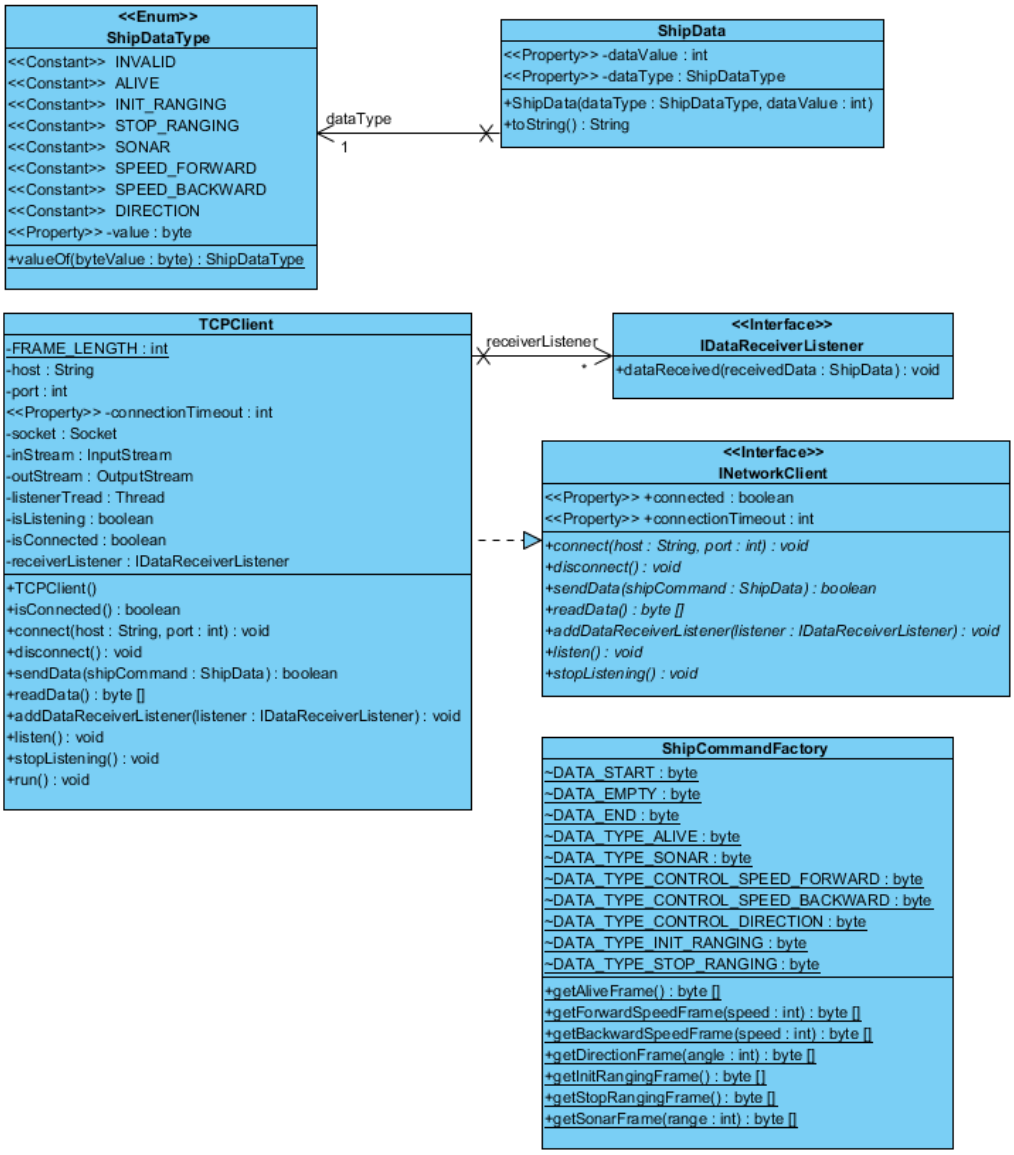


Abbildung A.6: Kalssendiagramm des Packages shipremote.communication

## A.2 Schaltskizze der Schiffselektronik

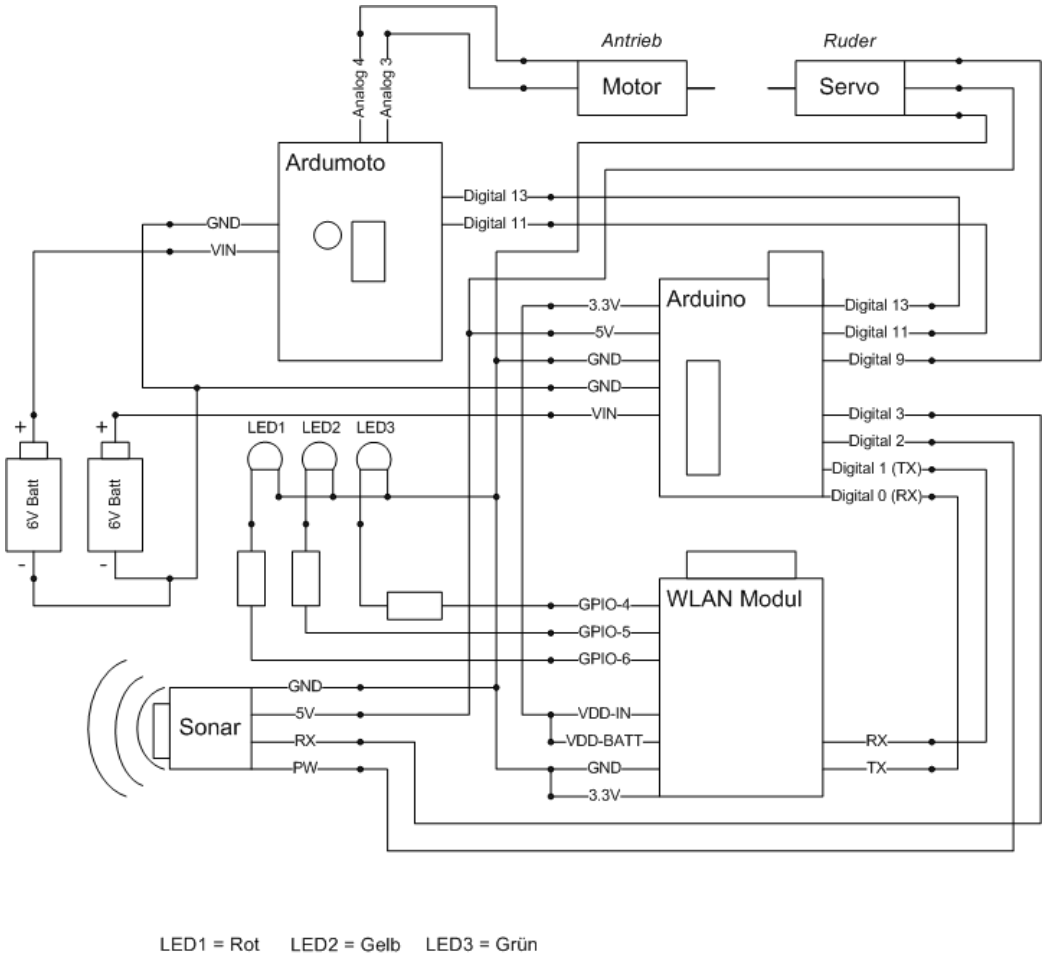


Abbildung A.7: Schaltskizze der Schiffselektronik

# Anhang B

# Fotos

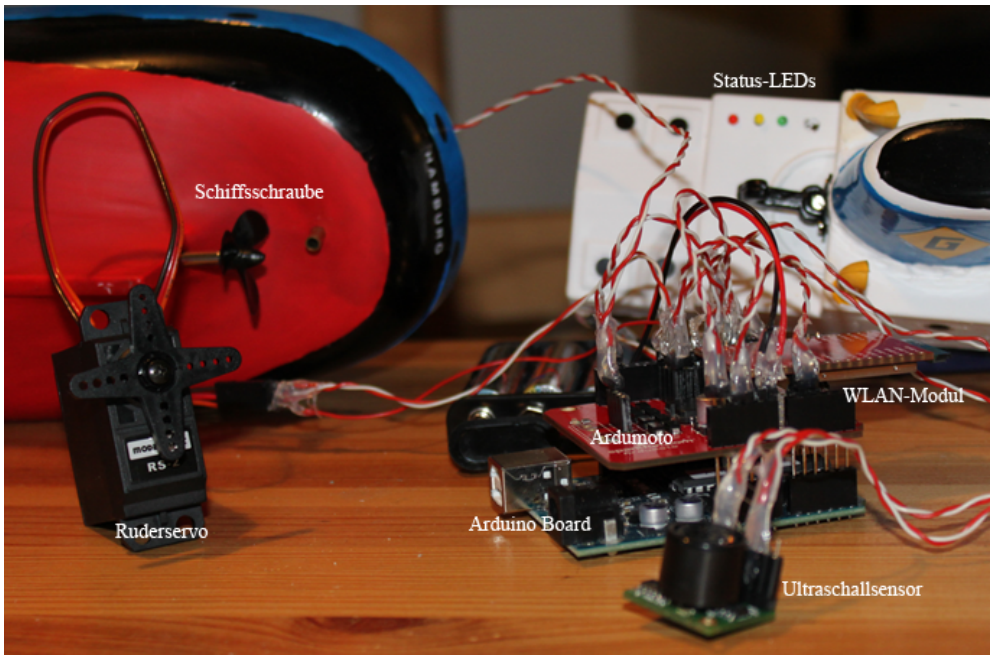


Abbildung B.1: Komponenten der Schiffselektronik

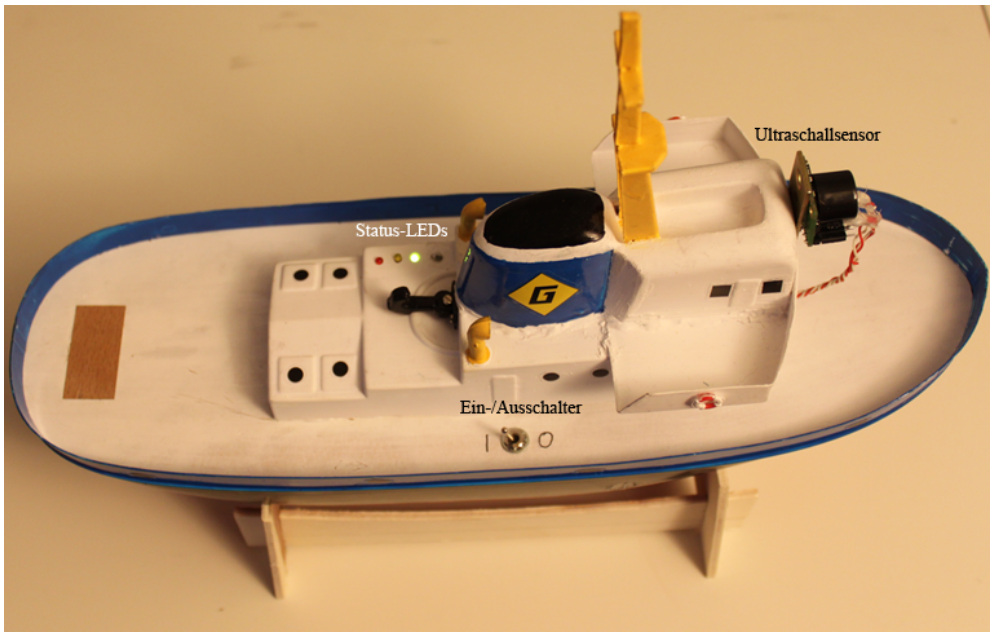


Abbildung B.2: Schiffsdeck und daran angebrachte technische Elemente

# Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

*Ort, Datum* *Andreas Günther*