Diplomarbeit

Ein Zigbee-Funkmodul zur Kommunikation unter autonomen Robotern.

vorgelegt von

Helge Rickens

am 26.08.2005

Studiengang Technische Informatik

Betreuender Prüfer: Prof. Dr. Gunter Klemke

Zweitprüfer: Prof. Dr. Reinhard Baran

Fachbereich Elektrotechnik und Informatik
Department of Electrical Engineering and Computer Science

Helge Rickens

Ein Zigbee-Funkmodul zur Kommunikation unter autonomen Robotern

Diplomarbeit
im Studiengang Technische Informatik

am Fachbereich Elektrotechnik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Gunter Klemke
Zweitgutachter: Prof. Dr. Reinhard Baran

Abgegeben am 26.08.2005

Helge Rickens

Thema der Diplomarbeit
Ein ZigBee-Funkmodul zur Kommunikation
unter autonomen Robotern

Stichworte

Zigbee, IEEE 802.15.4, CAN, Roboter, Roboterfußball, Kommunikation

Kurzzusammenfassung

Diese Arbeit beschäftigt sich damit, den an der HAW-Hamburg verwendeten AKSEN-Board-Robotern die Kommunikation per Funk zu ermöglichen. Die Roboter werden unter anderem für Robotfußballspiele eingesetzt, wo eine Kommunikationsmöglichkeit sehr hilfreich ist. Dafür wurde das Hardware-Modul CAtZi entwickelt und eine theoretische Lösung für ein notwendiges Protokoll erstellt. Unter Verwendung dieses Protokolls und Einsatz der ZigBee-Technologie, ermöglicht CAtZi die bidirektionale Funkübertragung von Daten des kabelgebundenen CAN-Busses, der Teil der AKSEN-Boards ist.

Helge Rickens

Title of the paper
A ZigBee-Module for Radio Communication between autonomous Robots

Keywords

Zigbee, IEEE 802.15.4, CAN, robots, robot soccer, communication

Abstract

This work is about the design and realization of a hard- and software solution, which allows the AKSEN-board robots, that are used at the HAW-Hamburg, to communicate via radio frequencies. Among other applications, these robots are used in robot soccer matches, where this type of communication is extremely helpful. For this purpose the hardware module CAtZi has been developed and a theoretical solution for the necessary protocol has been given. Together with this protocol and employing the ZigBee technology, CatZi allows for the bidirectional RF-transmission of data from the wired CAN bus, that is part of the AKSEN boards.

Danksagung

Hiermit möchte ich mich bei meinen Professoren, Kommilitonen und Freunden bedanken, die es mir ermöglicht haben, mein Studium zu absolvieren, das den Abschluss in dieser Arbeit fand. Mein besonderer Dank gilt Sven Kieckbusch, der es mit unermüdlicher Korrekturarbeit geschafft hat, aus meinen Wortwürmern verständliche Sätze zustande zu bringen. Weiter gilt er meiner Frau, die es zuwege gebracht hat, mir trotz der Geburt unseren zweiten Tochter Enya, die Zeit zu geben meine Diplomarbeit zu beenden. Dabei soll nicht unerwähnt bleiben, dass meine zwei Jahre alte Tochter Kira sehr lieb war, und mich die meiste Zeit in Ruhe hat arbeiten lassen.

Inhaltsverzeichnis

1. **Einleitung** ... 7
   1.1. Szenario Roboterfußball ... 7
   1.2. Motivation ... 8
   1.3. Die Aufgabe ... 10
      1.3.1. Die Aufgabe im Szenario ... 10

2. **Analyse der Anforderungen** ... 12
   2.1. Analyse der Hardwareanforderung ... 12
   2.2. Auswahl der Funktechniken ... 14
      2.2.1. Wireless Local Area Network (W-LAN) ... 14
      2.2.2. Bluetooth ... 14
      2.2.3. ZigBee ... 15
      2.2.4. nanoNET ... 15
      2.2.5. Vergleich der Funktechniken ... 16
      2.2.6. Anbindung des Funksystem an das AKSEN-Board ... 16
   2.3. Das Modul ... 17
   2.4. Das Protokoll ... 17

3. **Grundlagen über die Übertragungsverfahren** ... 18
   3.1. CAN-Bus ... 18
   3.2. ZigBee ... 22
      3.2.1. ZigBee-Frame ... 25

4. **Design des Hardwaremodules CAtZi** ... 29
   4.1. Die Funktionsbereiche ... 30
      4.1.1. CAN-Bus ... 30
      4.1.2. ZigBee ... 31
      4.1.3. Microcontroller ... 32
   4.2. Zusammenführung der Sektionen von CAtZ ... 33
      4.2.1. Spannungsversorgung ... 34
      4.2.2. Bedienelemente ... 35
      4.2.3. Steckverbindungen ... 35
      4.2.4. Der Schaltplan ... 36

5. **Software Design** ... 38
   5.1. Protokoll MAU ... 39
      5.1.1. Datenframe ... 40
      5.1.2. Richtlinien ... 44
      5.1.3. Konfigurationsnachrichten im MAU ... 47
   5.2. Sequenzdiagramm ... 48
   5.3. Harel-Automat ... 49

6. **Test** ... 50
   6.1. Test des Layouts von CAtZi auf Funktionsfähigkeit ... 50
   6.2. Test der Übertragungsrate von CAtZi ... 51

7. **Resumé** 54
7.1. Zusammenfassung 54
7.2. Zukunftsaussichten 54
7.3. Fazit 55

A) Abbildungsverzeichnis 56
B) Tabellenverzeichnis 58
C) Literaturverzeichnis 59
D) Weiterführende Dokumente 61

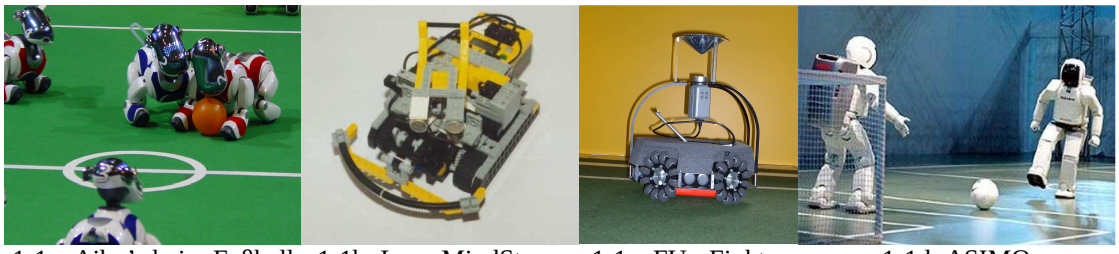
1 Einleitung

In dieser Arbeit geht es darum, den an der HAW-Hamburg verwendeten Robotern eine Möglichkeit zu geben, per Funk miteinander zu kommunizieren. Dadurch können Aufgaben im Team gelöst werden. Um Anforderungen an die Kommunikation und deren Zweck zu verdeutlichen und allgemein verständlich zu erklären wird ein Szenario zur Hilfe genommen. Weiter vermittelt die Arbeit Grundlagen zu den hierzu eingesetzten Technologien, um deren Funktionsweisen aufzuzeigen. Nachdem die Aufgabenstellung konkretisiert wurde, wird auf die in dieser Arbeit entworfene Hardware eingegangen, wobei Entwicklung, Aufbau und Funktion betrachtet werden. Für die verwendeten Roboter werden verschiedene Übertragungsverfahren benötigt, um eine Kommunikation zu ermöglichen. Um übergreifend arbeiten zu können, wird hierzu eine theoretische Softwarelösung entwickelt. Diese Lösung bezieht sich zum einen speziell auf die oben erwähnte Hardware, zum anderen wird sie von anderen Teilnehmern benötigt, um ebenfalls mit den Robotern kommunizieren zu können.

1.1 Szenario Roboterfußball

Roboterfußball gibt es in sehr unterschiedlichen Variationen, angefangen bei reinen Softwarelösungen zur Simulation in 2D- und 3D-Umgebungen, was der Erforschung der Künstlichen Intelligenz dient, über realen Fußball von sehr unterschiedlich komplex aufgebauten Robotern. Dabei können die Roboter sowohl reine Akteure darstellen, die per Funk von außen gesteuert werden, als auch völlig autonom. Letztere führen alle Sensoren mit sich, werten sie selber aus und stimmen ihre Taktik mit ihren Teamkollegen ab. Dabei werden zum Teil Spielzeugroboter wie z.B. der Hund Aibo [SONY-1] oder die Lego MindStorms [LEGO-1] verwendet. Darüber hinaus gibt es eine breite Palette an Eigenkonstruktionen bis hin zu komplexen Konstruktionen, die von der Industrie zu Forschungszwecken entwickelt wurden. Dazu gehört z.B. der humanoide Roboter ASIMO [HOND-1]. Für viele dieser Roboter gibt es schon Fußballligen, in denen sogar Weltmeisterschaften ausgetragen

werden. Das Reglement wird zumeist von Jahr zu Jahr angepasst, um der steigenden Leistungsfähigkeit der Roboter gerecht zu werden und diese zu fördern. Die beiden wichtigsten Meisterschaften sind wohl der RoboCup [ROCA-1] und die FIRA-Weltmeisterschaft [FIRA-1].



1-1a: Aibo's beim Fußball spielen | 1-1b: Lego MindStorm Konstruktion | 1-1c: FU - Fighter | 1-1d: ASIMO

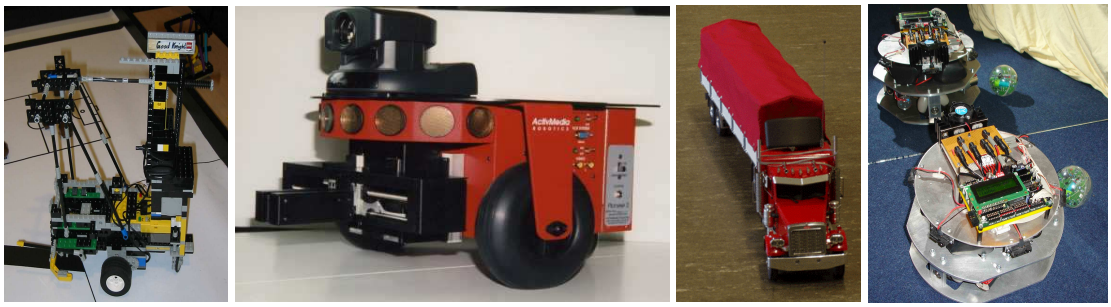
Abbildung 1-1: Auswahl verschiedener Roboter

1.2 Motivation

An der HAW-Hamburg wird mit verschiedenen autonomen Robotern gearbeitet, z.B. einem LKW, den MindStorm-Systemen von Lego, dem Pioneer und einem auf dem AKSEN-Board basierenden Roboter. Eine ausführliche Behandlung des Themas Roboterfußball an der HAW-Hamburg findet sich in der Studienarbeit von Oliver Köckritz [KÖCK-1]. Nicht alle diese Roboter sind zum Fußballspielen gleichermaßen geeignet. Der LKW wird hauptsächlich für Arbeiten und Untersuchungen an unterstützenden Systemen für den Fahrer bis hin zum autonomen Fahren im Straßenverkehr verwendet. Die MindStorms sind dank des Bauklotzprinzips von Lego sehr vielseitig einsetzbar, allerdings sind ihren sensorischen und aktorischen Möglichkeiten sowie die Rechenleistung sehr eingeschränkt. Es gibt hierzu Arbeiten, wie die Diplomarbeit von Dietmar Cordes und Gunther Lemm [COLE03], die mit ihrer Entwicklung des LePoMUX versuchen, die genannten Einschränkungen zu beseitigen. Der LePoMUX erweitert die normalerweise zur Verfügung stehenden 3 Eingänge und 3 Ausgänge auf insgesamt 16 Ports. Damit stehen dem MindStorm sechs Motorausgänge, ein Beacon-Detektor[1], vier Distanzsensor-Eingänge und fünf LEGO-kompatible Sensor-Anschlüsse zur Verfügung, was eine deutliche Verbesserung darstellt. Der Pioneer ist einer der ersten Roboter an der HAW-Hamburg und hat eine fest vorgegebene Hardware-Plattform. Das AKSEN-Board der FH-Brandenburg [FHBR-1] ist ein Controller für reaktive Roboter, der eine gute Grundlage für eigenentwickelte Roboter bildet. So entwickelt an der HAW-Hamburg Michael Ziener im Rahmen seiner Diplomarbeit

[1] Sensor zur Erkennung von IR-Leuchtfeuern.

[ZIEN05] einen, Roboter, der auf die Konstruktion von Michael Manger aufbaut, die er im Rahmen seiner Diplomarbeit [MANG04] erstellte.



1-2a: MindStorm | 1-2b: Pioneer | 1-2c: LKW mit Aufleger | 1-2d: AKSEN-Roboter

Abbildung 1-2: Roboter an der HAW-Hamburg

Warum bringt man Robotern das Fußballspielen bei? Es geht dabei nicht um das Fußballspielen an sich, sondern darum, unter den vorgegebenen Raumbedingungen (Regelwerke der entsprechen Fußballligen) möglichst leistungsfähige Roboter zu entwickeln, die in der Lage sind, ein Höchstmaß an selbstständiger Navigation und Handlungsfähigkeit zu erlangen. Dazu müssen die Roboter ein genaues Bild ihrer Umgebung bekommen und daraus verschiedene Positionen (z.B. die eigene, die des Balls, die der Mitspieler) erkennen, um Situationen bewerten und danach entsprechend handeln zu können. Speziell Fußball bietet nun neben den oben angeführten Anforderungen noch ein paar weitere Vorteile. Die Leistungsfähigkeit der verschiedenen Roboter lässt sich leicht anhand der Siege vergleichen. Da Fußball eine Mannschaftssportart ist, benötigt es auch eine Abstimmung der verschiedenen Mitspieler untereinander. Dabei kann es sein, dass die Spieler eines Teams aus verschiedenen Roboterkonstruktionen bestehen, die an ihre jeweilige Aufgabe angepasst sind. Weiterhin sind an fußballspielende Roboter nicht so hohe mechanische Anforderungen gestellt wie sie bei manchen anderen Sportarten nötig wären, z.B. einen Ball zu dribbeln wie beim Basketball. Als letztes Argument bleibt, dass Wettkämpfe Ehrgeiz steigern und das sich Fußball weltweit großer Beliebtheit erfreut, sowohl bei Leuten, die es spielen, als auch bei den Zuschauern.
Es wurden mit den Mindstorms und dem AKSEN-Board-Roboter schon Fußballspiele veranstaltet, doch waren diese zumeist eins gegen eins. Mit den MindStorm kann im begrenzten Umfang auch über die Infarotschnittstelle kommuniziert werden, allerdings ist diese Kommunikation unbefriedigend aufgrund der großen Störanfälligkeit, z.B. gegenüber dem Umgebungslicht, und der daraus resultierenden geringen Reichweite und Datentransferleistung. Um also einen besseren Einsatz von mehreren Robotern zu ermöglichen, benötigen die entsprechenden Roboter untereinander eine Möglichkeit, gut miteinander zu

kommunizieren. Daher ist das Ziel dieser Arbeit, dem AKSEN-Roboter über Funk eine Möglichkeit der Kommunikation zu geben.

1.3 Die Aufgabe

Um die AKSEN-Board-Roboter noch vielseitiger einsetzen zu können, sollen diese in die Lage versetzt werden, Daten sowohl mit anderen AKSEN-Board-Robotern als auch mit anderen Geräten wie Sensoren oder Servern auszutauschen. Server könnten als Schiedsrichter oder Trainer fungieren, die auf einem Host wie z.B. PC oder Laptop außerhalb des Spielfeldes laufen. Da das Spiel in einem überschaubaren Feld stattfindet, sind die Anforderungen an die Reichweite auf einige Meter beschränkt. Da die AKSEN-Boards nur begrenzte Resourcen zur Verfügung haben, um die Daten zu verarbeiten, ist auch die zu übertragende Datenmenge überschaubar. Die offensichtlich in Frage kommenden Systeme sind sicherlich W-Lan und Bluetooth. Eher unbekanntere Vertreter sind ZigBee und nanoNET. Die verschiedenen Systeme werden in Rahmen dieser Arbeit in 2.2 vorgestellt und verglichen.

1.3.1 Die Aufgabe erläutert im Szenario

„In einem Fußballspiel von autonomen Robotern wird drei gegen drei gespielt, wobei sich die Spieler untereinander über Funk koordinieren dürfen. Des Weiteren darf man ein externes System nach der aktuellen Position des Balles und seiner eigenen Position fragen.“ So könnte ein Ausschnitt aus einem Reglement aussehen. Um nun möglichst erfolgreich das Spiel zu bestreiten, könnten die Roboter untereinander, abhängig von ihrer Position, ausmachen, welcher Roboter welche Aufgabe übernimmt. So könnte sich der erste auf den Ball zubewegen, während der zweite sich in eine günstige Position begibt, um einen Pass anzunehmen und damit anschließend auf das Tor zu schießen. Der dritte Roboter übernimmt derweil die Aufgabe des Torwartes. Weiter könnte beim externen System nach der aktuellen Position des Balles gefragt werden, wenn der Ball gerade vom Roboter nicht wahrgenommen wird, um eine lange Suche zu vermeiden. Dies ist besonders dann der Fall, wenn der Ball gerade von einem anderen Spieler verdeckt wird.
Der Ablauf für das oben geschilderte Vorgehen könnte wie folgt aufgebaut sein. Dabei wird die in Abbildung 1-3 gezeigte Situation, aus Sicht von Team Gelb, angenommen.

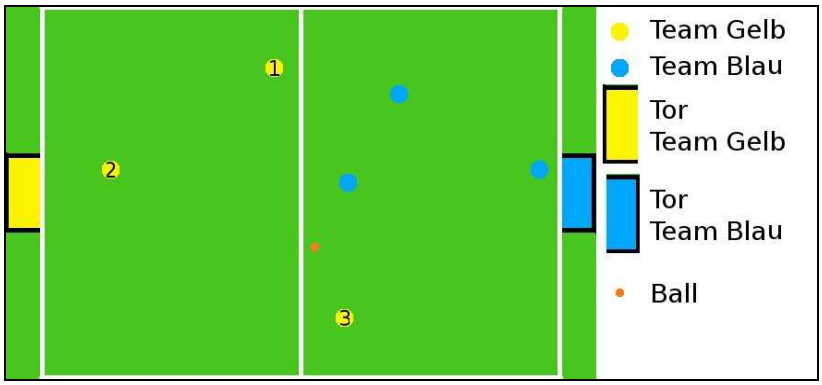


Abbildung 1-3: Schematische Darstellung einer Fußballszene

Roboter 1 ist als Mannschaftskapitän festgelegt und entscheidet somit über die eingesetzte Taktik. Nach einer Abfrage nach der Position des Balles wird die Entscheidung getroffen, dass ein Angriff gestartet werden soll. Dafür sendet er an alle Teammitglieder das Signal zum Angriff. Daraufhin senden alle Roboter eine Anfrage nach der eigenen Position und der des Balles. Roboter 2 und 3 senden ihre Koordinaten an Roboter 1, der diese auswertet und daraus die Rollenverteilung ermittelt und diese den anderen Robotern mitteilt. Roboter 1 entscheidet, dass Roboter 2 die Aufgabe des Torwartes übernimmt, während Roboter 3 die Aufgabe bekommt, den Ball in Besitz zu bringen. Roboter 1 gibt sich selbst die Aufgabe, sich in eine gute Position für eine Passannahme zu stellen.
Ob das geschilderte Vorgehen ein sinnvoller Ansatz ist, spielt hier keine Rolle und soll hier auch nicht weiter behandelt werden. Es geht ausschließlich darum, anhand dieses Beispieles den notwendigen Einsatz der Kommunikation zu verdeutlichen. Um diese Aufgabe mit den AKSEN-Board-Robotern lösen zu können, brauchen sie eine Kommunikationsmöglichkeit, die bisher jedoch nur in einer kabelgebundenen Form vorhanden ist (CAN-Bus).

2 Analyse der Anforderungen

Die an die Hard- und Software gestellten Anforderungen werden in diesem Abschnitt im Detail betrachtet.

2.1 Analyse der Hardwareanforderungen

Das Modul soll in erster Linie als Erweiterung für das AKSEN-Board [AKSE-1], im Besonderen im Einsatz von autonomen Robotern entwickelt werden. Das AKSEN-Board stellt eine Reihe von Anschlüssen zur Verfügung, unterschiedlich gut teilweise auch gar nicht eignen, um daran eine Funkverbindung anzuschließen. Zur Verfügung stehen 16 Digital-Ein- und Ausgänge, zwei analoge Eingänge, vier LED-Ausgänge, ein IR-Ausgang, vier Motor-Ports, ein Encoder-Port, ein Servo-Port und ein CAN-Bus. Die meisten der Anschlüsse kommen aufgrund ihres Einsatzzweckes nicht als zusätzliche Kommunikationsports in Frage. Dazu gehören die analogen Eingänge, die eine Spannung zwischen null und fünf Volt auf einen Wert zwischen null und 255 digitalisieren; die Motor-Ports, die zum Treiben von Motoren vorgesehen sind; die LED- und IR- Ausgänge zum Anschließen von LEDs und IR-LEDs; der Servo-Port, der Steuern eines Servomotors über PWM[1] gedacht ist und der Encoder-Port, der Übergänge von High nach Low zählt. Es bleiben also die digitalen Ein- und Ausgänge, die serielle Schnittstelle und der CAN-Bus als mögliche Kommunikationsschnittstellen. Dabei muss bedacht werden, dass der Zweck der digitalen Ein- und Ausgänge der Anschluss von zusätzlichen Sensoren ist und eine Belegung der Ein- und Ausgänge den Funktionsrahmen des AKSEN-Boards einschränkt.
Um eine Entscheidung über die Schnittstelle zu treffen werden Daten über das System gebraucht, welches angeschlossen werden soll. Voraussetzungen für das gewünschte System sind zum einen ein möglichst geringer Stromverbrauch, um den batteriebetriebenen Robotern eine möglichst lange Laufzeit zu ermöglichen, zum

[1] Pulsweitenmodulation

anderen eine robuste Übertragung der Daten. Außerdem ist es außerordentlich wichtig, dass eine ausreichend große Datentransferrate zur Verfügung gestellt wird. Um darüber eine Aussage treffen zu können, soll hier eine Abschätzung durchgeführt werden.
Mehr als 30 Kommunikationsteilnehmer sollten für das Szenario nicht benötigt werden (2 x 11 Spieler, 2 Trainer, 1 Schiedsrichter und einige externe Sensoren). Die Größe der Nutzdaten dürfte recht gering sein, da im Wesentlichen Koordinaten oder Befehle übermittelt werden. Im Allgemeinen müssten vier Byte ausreichen, um z.B. 16 Bit X,Y-Koordinaten zu übertragen. Um ein Maß zu bekommen, wurden folgende Werte zugrunde gelegt: Jeder Spieler einer Mannschaft sendet alle 100 ms eine Nachricht mit einer Länge von acht Byte. Weiter senden die Trainer alle 500 ms eine Nachricht (vier Byte) an ihre Mannschaft. Der Schiedsrichter sendet nur nach Bedarf und wird mit durchschnittlich einer Nachricht (8 Byte) pro Sekunde angenommen. Dann gibt es einen Sensor, der alle 10 ms die Koordinaten des Balles sendet (4 Byte). Weiter hat jede Mannschaft zwei weitere externe Sensoren im Einsatz, die alle 25 ms und 50 ms 4 Byte senden (Tabelle 2-1).

Sender	Empfänger	Anz. Sender	Größe in Byte	1 Nachricht x msec	kbit/s
Spieler	Team	11	8	100	7,04
Spieler	Team	11	8	100	7,04
Trainer	Team	2	4	500	0,128
Schiedsrichter	Alle	1	8	1000	0,064
Sensor 1	Alle	1	4	10	3,2
Sensor 2	Alle	1	8	100	0,64
Sensor 3	Team	1	4	50	0,64
Sensor 4	Team	1	4	50	0,64
Sensor 5	Team	1	4	25	1,28
Sensor 6	Team	1	4	25	1,28
		31	56		**21,952**

Tabelle 2-1: Abschätzung der Datentransferrate

Daraus ergibt sich eine Nettodatenrate von knapp 22 kbit/s. Durch nötigen Overhead für die Übertragung wird mehr Transferleistung benötigt, was hier großzügig mit einer Verdopplung berücksichtig werden soll. Weiter muss damit gerechnet werden, dass durch Störungen mehrmals gesendet werden muss, was ebenfalls großzügig mit einer Verdopplung der Datentransferrate berücksichtig wird, so dass 88 kbit/s errechnet werden. Eine Datentransferrate von ca. 100 kbit/s kann daher als ausreichend angesehen werden. Weiter sollte eine sichere Reichweite von 10 Metern erreicht werden, um ausreichend große Spielfelder abzudecken wie z.B. das aus der RoboCup Middle Size Liga.

2.2 Auswahl der Funktechniken

Es sollen hier die in Frage kommenden Funktechniken vorgestellt werden und auf ihre Eignung hin untersucht werden. Bei den Funktechniken handelt es sich um W-Lan, Bluetooth, ZigBee und nanoNET.

2.2.1 Wireless Local Area Network (W-LAN)

Das Einsatzgebiet von W-LAN ist vor allem die Vernetzung von mobilen Geräten, wie Laptops, Notebooks und PDAs untereinander, aber auch die Verbindung mit klassischen LANs und ins Internet, wobei auch immer mehr stationäre Rechner über W-Lan verbunden werden, da das umständliche und teure Verlegen von Kabeln entfällt. W-Lan wird zumeist mit dem Standard IEEE 802.11 betrieben, wobei es noch weitere Unterscheidungen im IEEE 802.11 gibt, die sich auf die genutzte Frequenz und die zu erreichende Datenrate beziehen. Die verbreitesten Vertreter sind IEEE 802.11a, 802.11b und 802.11g mit maximalen Datenraten von 11 Mbit/s und 54 Mbit/s. Auf dem Markt verbreiten sich zunehmend auch herstellerspezifische Verfahren, die IEEE 802.11a und IEEE 802.11g verbinden, um so Datenraten bis zu 108 Mbit/s zu ermöglichen. Weiterhin bietet W-Lan große Sende-Reichweiten (z.B. werden für IEEE 802.11a typischerweise Reichweiten für den Außenbereich von 400 m angeben). Der Stromverbrauch liegt hier typischerweise beim Senden bei ca. 350 bis 500 mA, beim Empfangen bei ca. 250 bis 400 mA und im Standby bei 15 bis 100 mA. Die Anzahl der Teilnehmer im Netzwerk hängt von der verwendeten Infrastruktur und dem Protokoll ab (z.B. bei IPv4 ca. vier Milliarden). Jedoch kann ein Access Point meist nur 64 bis 256 Clients verwalten. Der Access Point kann aber wiederum mit weiteren Access Points verbunden werden. Es ist auch zu bedenken, dass alle sich in Funkreichweite befindlichen Geräte sich die Übertragungsbandbreite teilen müssen, was im Übrigen für alle hier vorgestellten Techniken gilt.

2.2.2 Bluetooth

Das Einsatzgebiet für Bluetooth ist die Funkvernetzung von Geräten mit geringer Reichweite, wie die Verbindung von Handys und PDAs untereinander oder auch mit Computern, sowie die Anbindung von Peripheriegeräten wie Headsets oder Tastaturen und Mäusen. Die maximale Anzahl von Geräten in einem Bluetooth-Netzwerk beträgt 255, wovon jedoch nur acht gleichzeitig in Betrieb sein können. Bluetooth gruppiert die Geräte in drei Reichweitenklassen, wobei Klasse I eine Reichweite von 100 m, Klasse II eine von 20 m und Klasse III eine von 10 m

erreicht. Die Datenrate hängt vom verwendeten Modus ab und kann bei asymmetrischem Betrieb maximal 732,2 kbit/s in eine Richtung und 57,6 kbit/s in die Gegenrichtung erreichen oder im symmetrischen Betrieb mit 433,9 kbit/s in beide Richtungen arbeiten. Die Stromaufnahme eines Klasse-III-Gerätes liegt beim Senden bei ca. 10 mA und beim Empfangen bei ca 60 mA. Während des Sleep-Modus beträgt die Stromaufnahme 3 mA.

2.2.3 ZigBee

Der offenen Funknetz-Standard ZigBee ist eine junge Technologie, deren erste Spezifikation in der Version 1.0 Ende 2004 herausgegeben wurde. Dabei baut ZigBee auf den ebenfalls offenen Standard IEEE 802.15.4 auf und ist entwickelt worden, um zuverlässige, kostengünstige und verlustleistungsarme Sensor- und Aktor-Netzwerke zu ermöglichen. Die bei ZigBee erreichbare Datentransferrate hängt stark vom verwendeten Frequenzband ab und beträgt maximal 250 kbit/s. Dabei sind Reichweiten von 10 bis 70 Meter zu erreichen. Beim Senden und Empfangen wird ein Strom von ca. 20 mA aufgenommen. Im sparsamsten Modus (Power Down) kann eine Stromaufnahme von weniger als 10 µA erreicht werden. Die Anzahl der Geräte, die in einem Funknetz adressiert werden können, beträgt $2^{64} \approx 18$ Trillionen Stück.

2.2.4 nanoNET

Bei nanoNET kommt das völlig neue Funkübertragungsverfahren MDMA[2] zum Einsatz, welches in 802.15.4a standardisiert werden soll und so später auch in anderen Bereichen wie W-Lan oder ZigBee zum Einsatz kommen kann. Mit Fertigstellung der Standardisierung wird Ende 2006 gerechnet. nanoNET wie auch MDMA sind Entwicklungen der Firma Nanotron Technologies GmbH, die auch die Patente an den Entwicklungen besitzt. MDMA soll eine verbesserte Ausnutzung der Funkkanalkapazitäten erreichen. Das wiederum soll nanoNET in die Lage versetzen, bei einem geringen Stromverbrauch und hohen Datentransferraten störunempfindliche Sensor- und Aktor-Netzwerke mit großen Reichweiten zu ermöglichen. Die Stromaufnahme liegt beim Senden bei ca. 80 mA und während des Empfangs bei 35 mA. Im Sleep-Modus soll eine Reduktion der Stromaufnahme auf unter 2 µA möglich sein. Die dabei zu erzielende Datentransferrate liegt bei maximal 2 Mbit/s. Dabei können im Freifeld bis zu 900 Meter überbrückt werden, in geschlossenen Räumen ca. 60 Meter. Der Adressraum von nanoNET bietet Platz für $2^{48} \approx 300$ Billionen Geräte.

[2] Multi Dimensional Multiple Access

2.2.5 Vergleich der Funktechniken

W-Lan hat bei weitestem die höchste Datenrate, jedoch eine 10 bis 20-fach größere Stromaufnahme im Vergleich zu den anderen Funktechniken. Bluetooth hat das Problem, dass nur acht Teilnehmer gleichzeitig aktiv sein können, was unter Umständen zu wenig sein kann. ZigBee und nanoNET eignen sich für die Erfüllung der Aufgabe, da bei ZigBee eine Datenrate von 250 kbit/s und bei nanoNET bis zu 2 MBit/s zur Verfügung stehen. Weiter ist bei beiden Systemen bei geringem Stromverbrauch eine Reichweite von bis zu 60 Metern in Innenräumen zu erreichen. Die Wahl fiel auf ZigBee aufgrund der Tatsache, dass es sich um einen offenen Standard eines großen Konsortiums handelt, dessen Spezifikation veröffentlicht ist. Weiterhin erfolgen an der HAW-Hamburg momentan mehrere Arbeiten mit Bezug zum Thema ZigBee. So können zum einen Ressourcen gemeinsam genutzt werden und zum anderen können spätere Arbeiten auf die verschieden vorausgegangenen Arbeiten aufbauen.

2.2.6 Anbindung des Funksystems an das AKSEN-Board

Eine Anbindung der Funkverbindung kann auf zwei Arten geschehen. Zum einen wird das Funkmodul direkt an den Mikrocontroller des AKSEN-Boards gekoppelt und übernimmt somit die direkte Steuerung und Verwaltung der Daten, was den Vorteil hat, dass wenig Hardware benötigt wird. Der Nachteil ist, dass der Mikrocontroller Rechen- und Speicherkapazitäten für die Kommunikation verwenden muss, die somit nicht mehr für andere Aufgaben zur Verfügung stehen. Bei der zweiten Variante übernimmt ein separates Modul die eigentliche Kommunikation mit den anderen Teilnehmern und der Austausch findet nur zwischen dem AKSEN-Board und dem Modul statt. Die Vorteile sind, dass vorhandene Schnittstellen des AKSEN-Boards genutzt werden können und dieses nur minimale Ressourcen für die Kommunikation aufbringen muss. Der Nachteil ist, dass ein aufwendiges Modul benötigt wird und eventuell eine Anpassung der Protokolle vorgenommen werden muss. Dies jedoch hat den Vorteil, dass die Kommunikation unabhängig von der verwendeten Funktechnik ist und bei Bedarf so leichter geändert werden kann. Die Wahl fiel auf den CAN-Bus, da er die benötigte Bandbreite zur Verfügung stellt, im Gegensatz zur seriellen Schnittstelle, die nur mit 9600 Baud arbeitet. Es kommt dadurch zu keinen Einschränkungen im Einsatz des AKSEN-Boards durch Belegung der digitalen Ein- und Ausgänge. Der CAN-Bus bleibt durch die Gegebenheit, dass es sich um einen Bus handelt, auch weiterhin zur Verfügung. Daraus folgt, dass ein Modul entwickelt werden muss, das eine Verbindung über CAN-Bus und ZigBee ermöglicht.

2.3 Das Modul

Um dem AKSEN-Board die Möglichkeit zu geben über Funk zu kommunizieren wird ein Hardwaremodul zur Erweiterung des AKSEN-Boards benötigt. Dieses Modul soll das AKSEN-Board in die Lage versetzen, die per CAN-Bus an das Modul gesendeten Daten per Funk, genauer per ZigBee, an das gewünschte Ziel weiterzuleiten. Auch der umgekehrte Weg über das Modul soll möglich sein, also die vom Modul per Funk empfangenen Daten per CAN-Bus an das AKSEN-Board weiterzuleiten. Das Modul wird im Folgenden mit CAtZi bezeichnet (CAN to ZigBee). CAtZi sollte Punkt-zu-Punkt-Verbindungen, Punkt-zu–Gruppe- und Broadcast-Nachrichten ermöglichen, um sowohl Nachrichten unter den Spielern zur Absprache zu ermöglichen, als auch Anweisungen des Trainers an die gesamte Mannschaft möglich zu machen und schließlich Nachrichten des Schiedsrichters an alle übermitteln zu können. Um volle Funktionalität auch in einer Umgebung bieten zu können, in der die Roboter nur untereinander kommunizieren dürfen, sollte die Möglichkeit bestehen, dass eines der Module die Aufgabe des Koordinators übernimmt. Dementsprechend muss das CAtZi als FFD entworfen werden, was den Vorteil hat, dass es dann auch als Router zur Erhöhung der Netzreichweite dienen kann (vergleiche Kapitel 3). Um so eine gezielte Weiterleitung über die beiden unterschiedlichen Verfahren zu ermöglichen und diese so miteinander zu vereinen, wird ein übergeordnetes Protokoll benötigt.

2.4 Das Protokoll

Die Hauptaufgabe des Protokolles ist es, eine Zuordnung der Nachrichten zu deren Empfängern zu ermöglichen und das über beide Übertragungsverfahren, CAN und ZigBee, hinweg. Hauptaugenmerk ist dabei die Kommunikation zwischen verschiedenen AKSEN-Boards. Wenn möglich, sollte auch die Kommunikation mit einem Hostrechner wie auch mit einfachen ZigBee-Geräten ermöglicht werden. Im Falle der einfachen ZigBee Geräte ist die volle Funktionalität jedoch nicht erforderlich. Weiterhin ist es vorteilhaft, wenn das Modul von außen über das Protokoll konfiguriert werden kann. So wird eine Anpassung nur an dieser einen externen Stelle nötig und kann z.B. in der Halbzeit verändert werden. Weiterhin wird die Kommunikation vereinfacht, wenn häufig genutzte Nachrichten im Protokoll festgelegt werden, wie z.B. das Versenden von Ortsdaten durch Koordinaten.

3 Grundlagen der Übertragungsverfahren

In diesem Abschnitt soll ein grundlegendes Verständnis über die zum Einsatz kommenden Übertragungsverfahren ZigBee und CAN[1] gegeben werden. Zur tiefergehenden Behandlung der Themen wird auf die folgenden Quellen verwiesen: Zum Thema CAN die Referenzen [BOSC-1], [ETSC02], [LOWR00] und zum Thema ZigBee die Referenzen [IEEE03], [ZIGB04].
Gerade über das Thema ZigBee kann nur ein kleiner Überblick geben werden, da die ZigBee Spezifikation knapp 400 Seiten umfasst, zu denen noch knapp 700 Seiten des Standards IEEE 802.15.4 kommen, auf den ZigBee aufbaut.

3.1 CAN-Bus

Der CAN-Bus gehört zu den Feldbussen[2] und wurde 1983 von der Firma Bosch entwickelt. Ziel war es, die Anzahl der Kabel in den Kabelbäumen, die in Automobilen zum Einsatz kommen, zu reduzieren, um so Gewicht zu sparen. Der CAN-Bus bildet eine linienförmige Topologie, an die Geräte per Stichleitung angeschlossen werden. Die maximal zulässige Länge des Busses hängt von der Signallaufzeit ab und somit von der Bitrate. Der Zusammenhang kommt durch das Arbitrationsverfahren[3] [ETSC02]. Dabei muss ein Puls (Bit) mit seiner

---

[1] CAN - **C**ontroller **A**rea **N**etwork
[2] „Ein Feldbus ist ein industrielles Kommunikationssystem, das eine Vielzahl von Feldgeräten wie Messfühler (Sensoren), Stellglieder und Antriebe (Aktoren) mit einem Steuerungsgerät verbindet. Die Feldbustechnik wurde in der 1980er Jahren entwickelt, um die bis dahin übliche Parallelverdrahtung binärer Signale sowie die analoge Signalübertragung durch digitale Übertragungstechnik zu ersetzen. Heute sind unterschiedliche Feldbussysteme mit unterschiedlichen Eigenschaften am Markt etabliert, wie z.B. PROFIBUS, INTERBUS, ControlNet oder CAN. Seit 1999 werden Feldbusse in der Norm IEC 61158 ("Digital data communication for measurement and control - Fieldbus for use in industrial control systems") weltweit standardisiert.“[WIKI-1]
[3] Arbitrierung: Aushandeln des Medienzugriffs

Ausbreitungsgeschwindigkeit innerhalb einer bestimmten Zeit, abhängig von der Bitrate, die doppelte Buslänge zurücklegen können. Die Tabelle 3-1 zeigt verschiedene Kabellängen mit den dazu maximal möglichen Bitraten. Die maximale Datenrate beträgt 1 MBit/s und die minimale Rate liegt bei 5kBit/s. Es müssen an einem Bus alle Geräte mit der gleichen Geschwindigkeit arbeiten. Es können theoretisch beliebig viele Geräte an den Bus angeschlossen werden, da keine Adressierung der Geräte vorgenommen wird, sondern die Nachrichtenobjekte gekennzeichnet werden. Ein solches Nachrichtenobjekt könnte z.B. eine Motordrehzahl sein. Die tatsächlich mögliche Anzahl an Geräten hängt vom genauen Netzaufbau ab und liegt bei Standardlösungen bei 32 Geräten und z.B. bei SDS-Treibern bei bis zu 128 Geräten. Jedes Gerät wertet dann selber aus, ob die gesendete Nachricht von Interesse ist und verarbeitet werden muss. Jeder einzelne Objectidentifier kennzeichnet ein Nachrichtenobjekt und ist einem Sender zugeordnet, d.h. nur dieser Sender darf Nachrichten mit dem ihm zugeordneten Objectidentifer versenden, da ansonsten die Arbitrierung nicht funktioniert. Ausnahme bildet die Remotenachricht, deren Inhalt festgelegt ist und somit auch von mehreren Geräten gleichzeitig gesendet werden kann. Das heisst, eine Nachricht mit einem bestimmten Objectidentifier darf immer nur von einem Gerät kommen und kann von einem, mehreren oder allen andern Geräten empfangen werden. Die Arbitrierung arbeitet nach dem CSMA/CD+CR-Verfahren[4]. Der Objectidentifier ist auch gleichzeitig, durch das Arbitrierungsverfahren bedingt, die Priorität der Nachricht, wobei die höchste Priorität mit dem Wert null gekennzeichnet ist. Die maximal mögliche Anzahl der verschieden Nachrichtenobjekte, und somit auch der verschiedenen Prioritäten, hängt vom eingesetzten Protokoll ab und ist bei CAN 2.0A, auch Basic- oder Standard-CAN genannt, 11 Bit lang, (2^{11} = 2048 verschiedenen Nachrichten) und bei CAN 2.0B, auch Extended- oder Full-CAN genannt, 29 Bit lang (2^{29} = 536870912 verschiedenen Nachrichten). Abbildung 3-1 zeigt den Aufbau eines Datenframes bei CAN 2.0B. Neben dem Datenframe gibt es noch einen Remote-Frame zum Anfordern von Daten, sowie einen Error-Frame, der gesendet wird sobald ein Teilnehmer einen Fehler entdeckt, und einen Overload-Frame, der gesendet wird, um die nächste Nachricht zu verzögern. Beim CAN-Protokoll dürfen, zum Zwecke der Nachsynchronisation, immer nur fünf rezessive oder fünf dominate Bits aufeinander

Bitrate	Kabellänge
10 kbit/s	6,7 km
20 kbit/s	3,3 km
50 kbit/s	1,3 km
125 kbit/s	530 m
250 kbit/s	270 m
500 kbit/s	130 m
1 Mbit/s	40 m

Tabelle 3-1: CAN Übertragungsraten zu Kabellängen

[4] CSMA/CD+CR - **C**arrier **S**ense **M**ultiple **A**ccess / **C**ollision **D**etection + **C**ollision **R**esolution
Dabei werden während der Identifier-Übertragung Kollisionen erkannt und die Nachricht mit der höchsten Priorität wird weiter übertragen.

folgen. Nach fünf gleichen Bits folgt dann immer ein Stuff-Bit (komplementäres Bit). Die Stuff-Bits werden auf der Empfangsseite wieder ausgefiltert.

Dataframe CAN 2.0 B (29 Bit Identifier)															
1	11	1	1	18	1	1	1	4	0-64	15	1	1	1	7	3+
SOF	Identifier 1	SRR	IDE	Identifier 2	RTR	r1	r0	DL Länge der Daten	Daten 0 bis 8 Byte	CRC	DEL	ACK	DEL	EOF	IFS

Abbildung 3-1: CAN 2.0B Data Frame

- SOF: Start of Frame / Startbit
- Identifier 1+2 ergeben zusammen die ID der Nachricht und auch die Priorität.
- SRR: Substitute Remote Request
- IDE: Identifier Extension, gibt an, ob es sich um CAN2.0A oder B handelt.
- RTR: Remote Transmission Request, entscheidet ob es sich um ein Data- oder Remote-Frame handelt.
- r0 und r1 sind reserviert.
- DL enthält die Länge der zu senden Daten in Byte und darf laut Spezifikation nur Werte von 0 bis 8 enthalten.
- Daten sind die Nutzdaten und enthalten 0 bis 8 Byte (es werden immer ganze Byte gesendet).
- CRC[5]: Fehlererkennung über alle vorangegangenen Bits.
- DEL: Delimiter
- ACK: Acknowledge / Bestätigungsfeld
- EOF: End of Frame
- IFS: Inter Frame Space

Der Remote-Frame ist genauso aufgebaut wie der Data-Frame und wird durch das RTR Bit unterschieden, wobei das Datenfeld keine Daten enthält. Sowohl Error- als auch Overload-Frame dürfen gegen die oben erwähnte Bit-Stuffing-Regel verstoßen. Der Unterschied ist, dass ein ERROR-Frame gesendet wird, sobald ein Fehler erkannt wird und der Overload-Frame nur nach Abschluss einer Nachricht gesendet werden darf.

[5] CRC - **C**yclic **r**edundancy **c**heck

3.2 ZigBee-Netzwerk

Die ZigBee-Alliance, der mehr als 100 Unternehmen angehören, fördert den offenen ZigBee-Standard, der den Standard IEEE 802.15.4 als Plattform nutzt. Das Ziel ist ein Funksystem mit niedrigem Energieverbrauch, hoher Sicherheit und äußerster Flexibilität zu schaffen. Die ursprünglichen vorgesehenen Einsatzgebiete sind Aktor-Sensor-Netzwerke für die Industrie zur Steuerung, Überwachung und Automatisierung von Fertigungsprozessen. Ein weiteres Einsatzgebiet ist die Gebäudeautomatisierung.

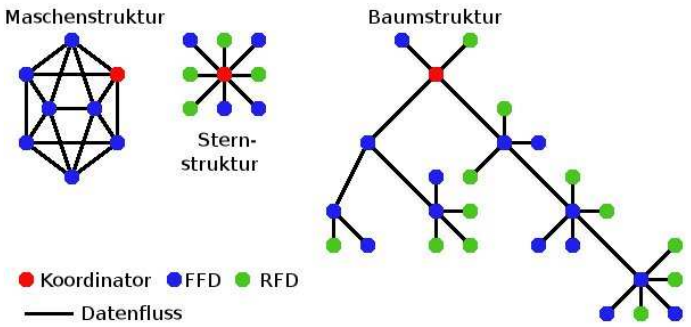


Abbildung 3-2: Mögliche ZigBee-Topologien

Die Topologie des ZigBee-Netzes kann als Stern-, Baum- oder Maschennetz aufgebaut sein. (Siehe Abbildung 3-2). Die einzelnen Knoten bestehen dabei aus Full-Function-Devices[6] (FFD) und Reduced-Function-Devices[7] (RFD). Dabei nehmen die Geräte eine der folgenden Rollen ein: Koordinator, Alternativ-Koordinator, Router oder Endgerät. Ein RFD kann dabei immer nur die Rolle eines Endgerätes übernehmen. Jede Rolle kann auch die Aufgaben der untergeordneten Rolle enthalten.
In jedem Netzwerk gibt es genau einen Koordinator und keinen, einen oder mehrere sogenannter Alternativ-Koordinatoren, die im Falle eines Ausfalles des Koordinators dessen Aufgabe übernehmen. Ein solcher Ausfall kann z.B. durch Verlassen der Reichweite des Netzes oder durch einen technischen Defekt begründet sein.
Die FFD und RFD unterscheiden sich auch darin, wie sie in den verschiedenen Topologien eingesetzt werden können. In einem Maschennetz können nur Full-Function-Devices eingesetzt werden. RFD nehmen immer die Position von Blättern in ihrer Struktur ein, da sie immer ein FFD als Kommunikationspartner benötigen. Reduced-Function-Devices sind in der Regel für eine spezielle Aufgabe konzipiert

[6] Gerät mit voller Funktionalität
[7] Gerät mit reduzierter Funktionalität

und können somit kostengünstiger und energiesparender als Full-Function-Devices sein.
Zur Adressierung der Geräte können 16-Bit oder 64-Bit Adressen verwendet werden. Bei der Anmeldung des Gerätes im Netz wird ihm vom Koordinator ein PAN[8]-Identifier zugewiesen, der eine Länge von 16-Bit hat, und über den die Geräte dann angesprochen werden. Die zu erzielenden Datenraten hängen vom verwendeten Frequenzbereich ab. Das 2,4 GHz-Band stellt 12 Kanäle mit jeweils 250 kbit/s zur Verfügung, das 915 MHz-Band 4 Kanäle mit jeweils 40 kbit/s, das 868 MHz-Band nur einen Kanal mit 20 kbit/s.

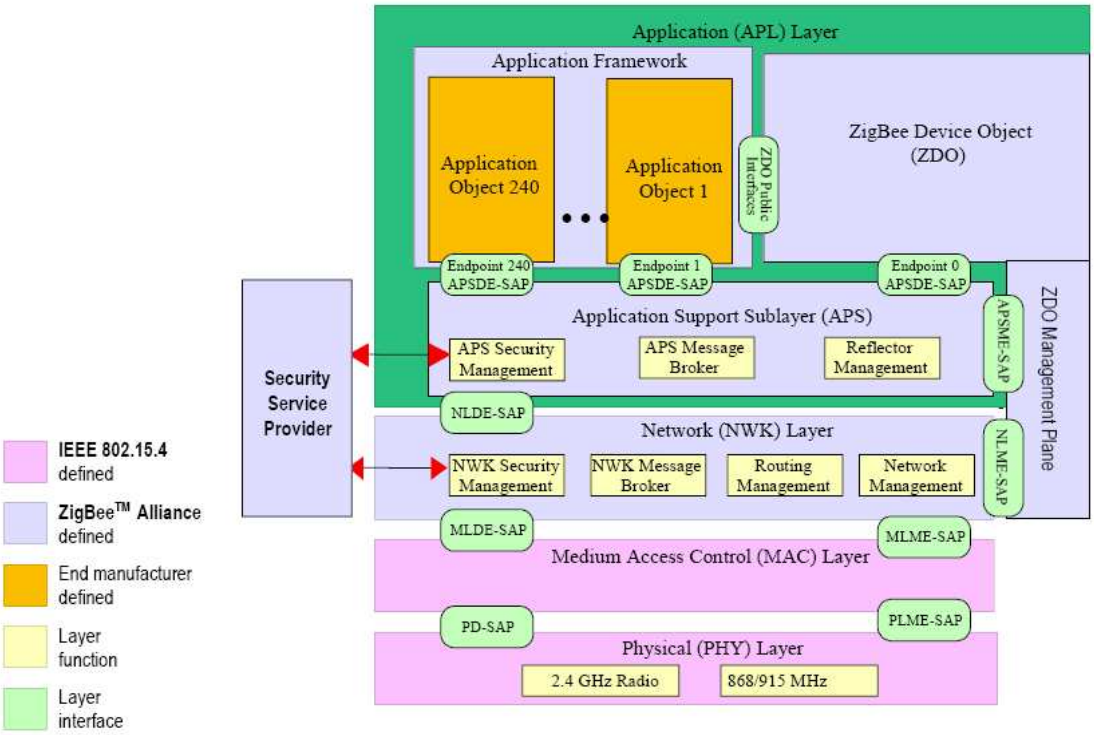


Abbildung 3-3: Überblick über die ZigBee Stack Architektur

„Der Protokollstapel für ZigBee und IEEE802.15.4 ist in Abbildung 3-3 gezeigt. Er besteht aus folgenden Elementen:

- Die physikalische Übertragungsschicht (PHY-Layer) beschreibt die Übertragung der Bits mit Hilfe von Frequenzspreizverfahren im 868-/915- bzw. im 2,4-GHz-Band mit einer Bruttodatenrate von bis zu 250 kbit/s.
- Die Kanalzugriffsschicht regelt die Adressierung der Stationen, die grundlegende Verwaltung von kleinen „Peer-to-Peer“- oder Stern-Netzen

[8] PAN = personal area network

sowie den Kanalzugriff. Als Kanalzugriffsalgorithmus dient ein „Listen-Before-Talk“-Verfahren (LBT), das zusätzliche Funktionen für die Kollisionsvermeidung sowie die Übertragung periodischer Datenströme besitzt (Carrier Sense Multiple Access, Collision Avoidance – CSMA/CA). Die Schichten 1 und 2 wurden vom IEEE im Standard 802.15.4 im Oktober 2003 verabschiedet. Hervorzuheben ist hierbei, dass Teile der PHY- und die komplette MAC-Funktion in den gegenwärtig verfügbaren Produkten in Software auf dem Host-Controller ausgeführt wird.

- In der Netzwerkschicht besteht die Möglichkeit, Stern-Topologien miteinander zu verbinden, sodass ein „Clustered-Tree“-Netzwerk entsteht. Die Netzwerkschicht kümmert sich hierbei um die logische Adressierung der Stationen, das statische oder dynamische Routing sowie um die Verwaltung der Security-Toolbox. Die Netzwerkschicht erlaubt dabei den Aufbau von Multi-Hop-Netzwerken, die ein recht hohes Maß von Adaptivität erreichen und insbesondere die Reichweitenproblematik durch die Möglichkeit der Paketweiterleitung entschärfen.
- Schließlich wird noch die Anwendungsschicht beschrieben. Hierbei geht es, neben der Initialisierung der Verbindungsfunktion mit Hilfe des so genannten „Generic Device Object“ (GDO), um die sehr detaillierte Beschreibung des Gerätefunktionsumfangs. Als erste Beispiele finden sich im ZigBee-Standard Anwendungsprofile für Lichtschalter, Dimmer und Lampen. Die Protokolle der Netzwerk- und Anwendungsschichten wurden von der ZigBee-Alliance im Dezember 2004 in der Version 1.0 verabschiedet.“ [SIKO-1]

ZigBee beschreibt alle sieben Schichten des OSI-Models, wobei festgelegt ist, dass die unteren drei Schichten durch den Standard IEEE 802.1.4 abgedeckt werden. Die Schichten vier bis sechs werden dann durch den ZigBee-Protokoll-Stack abgedeckt, während die siebte Schicht die eigentliche Anwendung darstellt. Abbildung 3-4 zeigt eine grafische Darstellung der Zuordnung der ZigBee Protokolle zum OSI-Model.

<table>
<tr><th colspan="2">OSI-Model-Schicht</th><th>Englisch</th><th>ZigBee Protokolle</th><th></th></tr>
<tr><td>7</td><td>Anwendung</td><td>Application</td><td>Anwendungs-Software</td><td rowspan="4">ZigBee Protokoll Stack</td></tr>
<tr><td>6</td><td>Darstellung</td><td>Presentation</td><td>Application Objects</td></tr>
<tr><td>5</td><td>Sitzung</td><td>Session</td><td>ZB APS SW</td></tr>
<tr><td>4</td><td>Transport</td><td>Transport</td><td>ZB NKW SW</td></tr>
<tr><td>3</td><td>Vermittlung</td><td>Network</td><td>MAC SW</td><td rowspan="3">IEEE 802.15.4</td></tr>
<tr><td>2</td><td>Sicherung</td><td>Data Link</td><td>PHY SW</td></tr>
<tr><td>1</td><td>Bitübertragung</td><td>Physical</td><td>HF-Chip</td></tr>
</table>

Abbildung 3-4: ZigBee im OSI-Model

Der ZigBee Protokollstapel ist streng modular aufgebaut, so dass der Anwender-Software möglich ist, auf jeder beliebigen Schicht aufzusetzen. Daraus ergeben sich eine Vielzahl von Implementierungsmöglichkeiten, die in der Abbildung 3-5 dargestellt sind. [SIKO-01]

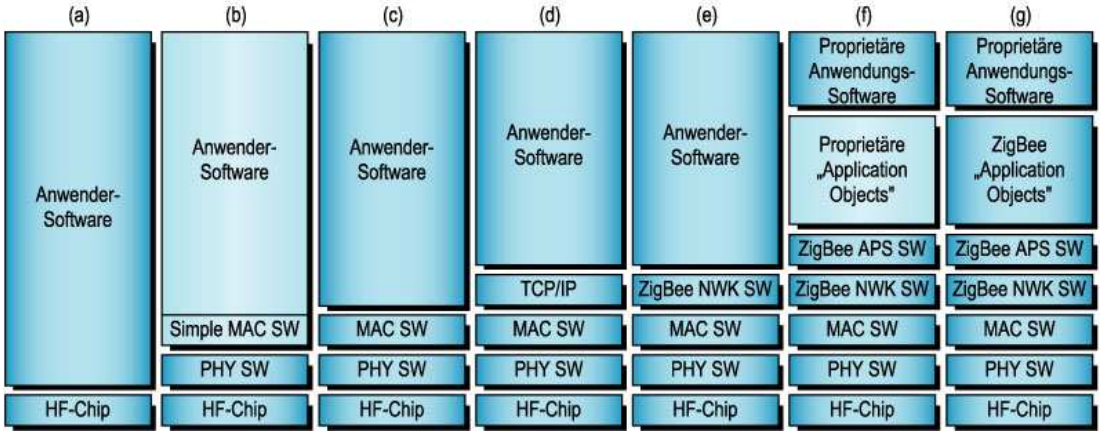


Abbildung 3-5: Implementierungsmöglichkeiten auf dem ZigBee Protokollstapel

3.2.1 ZigBee-Frame

Hier soll, wie auch schon im Abschnitt 3.1 CAN-Bus, der Frameaufbau des ZigBee-Standards kurz betrachtet werden. Bei ZigBee gibt es das APS-Frame[9] und das NPDU[10]-Frame. Für die direkte Anbindung an die Anwendungsschicht ist das APS-Frame zuständig, das unterteilt ist in APS-Header und APS-Payload. Das NPDU-Frame gehört zum NKW[11]-Layer, ist zwischen der Anwendungsschicht und MAC-Schicht angeordnet (Abbildung 3-3) und ist für die Netzwerkkommunikation zuständig. Das NPDU-Frame ist unterteilt in NWK-Header und NWK-Payload.

Der genaue Aufbau des APS-Frame ist der Abbildung 3-6 zu entnehmen.

<table>
<tr><td>Octets:1</td><td>0/1</td><td>0/1</td><td>0/2</td><td>0/1</td><td>Variabel</td></tr>
<tr><td rowspan="2">Frame control</td><td>Destination endpoint</td><td>Cluster Identifier</td><td>Profile Identifier</td><td>Source endpoint</td><td rowspan="2">Frame payload</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="5">APS header</td><td>APS payload</td></tr>
</table>
Abbildung 3-6: APS-Frame

[9] APS = Application support sub-layer
[10] NPDU = Network layer protocol data unit
[11] NWK = Network

Das Frame-Control ist acht Bit lang, hat den in Abbildung 3-7 zu entnehmenden Aufbau und enthält die Eigenschaften des APS-Frames.

Bits: 0-1	2-3	4	5	6	7
Frame Type	Delivery Mode	Indirect address mode	Security	Ack. request	Reserved

Abbildung 3-7: APS-Frame-Control

Der Frame-Type kann die aus Tabelle 3-2 zu entnehmenden Werte annehmen. Die Werte vom Delivery-Mode finden sich in der gleichen Tabelle.

Value Frame Type Delivery Mode	Frame type name	Delivery mode name
00	Data	Normal unicast delivery
01	Command	Indirect addressing
10	Acknowledgement	Broadcast
11	Reserved	Reserved

Tabelle 3-2: APS-Frame-Type und APS-Delivery-Mode

Die Sektion „Adressing fields“ besteht aus den folgenden vier Bereichen:

- Der Destination-Endpoint enthält das Endziel des Frames.
- Der Cluster-Identifier enthält die Cluster-ID des Endziels oder des Koordinators.
- Der Profile Identifier enthält die Art der Nachricht, um ein Ausfiltern von ungewünschten Nachrichten zu ermöglichen. Einige Werte sind bestimmten Nachrichten fest zugeordnet.
- Der Source-Endpoint gibt den Endpunkt des Initiators des Frames an.

Der Frame-Aufbau für die verschiedenen Frame-Types ist im Folgenden dargestellt. Der Aufbau eines APS-Daten-Frames ist Abbildung 3-8 gezeigt.

Octets:1	0/1	1	2	0/1	Variabel
Frame control	Destination Endpoint	Cluster Identifier	Profile Identifier	Source endpoint	Data payload
APS header					APS payload

Abbildung 3-8: APS-Daten-Frame

Der Aufbau des APS-Command-Frame ist in Abbildung 3-9 dargestellt.

Octets:1	1	Variabel
Frame Control	APS command identifier	APS command payload
APS header	APS payload	

Abbildung 3-9: APS-Command-Frame

Der Aufbau des APS-Acknowledgement-Frame ist schließlich in Abbildung 3-10 gezeigt.

Octets:1	0/1	1	2	0/1
Frame control	Destination Endpoint	Cluster Identifier	Profile Identifier	Source endpoint
APS header				

Abbildung 3-10: APS-Acknowledgement-Frame

Als nächstes folgt der Aufbau des NPDU-Frame dessen weitere Aufteilung in Abbildung 3-11 zu sehen ist.

Octets:2	2	2	1	1	Variabel
Frame control	Destination Address	Source Address	Radius	Sequence Number	Frame payload
	Routing fields				
NWK Header					NWK Payload

Abbildung 3-11: NKW-Frame

Das Frame-Control ist 16 Bit lang und hat den aus Abbildung 3-12 zu entnehmenden Aufbau. Er enthält wie auch schon beim APS die Eigenschaften des Frames.

Bit: 0-1	2-5	6-7	8	9	10-15
Frame Type	Protocol version	Discover Route	Reserved	Security	Reserved

Abbildung 3-12: NKW-Frame-Control

Der Frame-Type kann die aus Tabelle 3-3 zu entnehmenden Werte annehmen.

Frame-Type Wert	Frame-Type Name
00	Data
01	NWK command
10-11	Reserved

Tabelle 3-3: NKW-Frame-Type Werte

Das Protocol-Version-Feld enthält die Protokollversion, die vier Bit lang ist und hat in der aktuellen Version von ZigBee den Wert 0x01, der auch den momentan einzigen gültigen Wert darstellt.

Die Werte für das Discover-Route können der Tabelle 3-4 entnommen werden.

Discover-Route Werte	Discover-Route Name
00	Suppress route discovery
01	Enable route discovery
10	Force route discovery
11	Reserved

Tabelle 3-4: Discover-Route

Das Security-Bit ist eins, wenn der Frame NWK-Security nutz und null wenn er sie nicht nutzt.

Die Sektion „Routing fields“ besteht aus den folgenden vier Bereichen:

- Die Destination-Adress enthält die Ziel-Netzwerkadresse, die identisch sein sollte mit der kurzen MAC-Adresse aus dem Standard IEEE 802.15.4.
- Die Source-Adress enthält die Quell-Netzwerkadresse, die identisch sein sollte mit der kurzen MAC-Adresse aus dem Standard IEEE 802.15.4.
- Der Radius enthält die Anzahl der maximal erlaubten Sprünge bis zum Ziel.
- Die Sequence-Number wird verwendet, um erkennen zu können, ob alle Nachrichten empfangen wurden. Das geschieht über eine fortlaufende Zahl.

Für das Senden eines NKW-Daten-Frame ist der Aufbau dann wie in Abbildung 3-13.

<table>
<tr><td>Occtets:2</td><td>2</td><td>2</td><td>1</td><td>1</td><td>Variabel</td></tr>
<tr><td rowspan="2">Frame control</td><td>Destination Address</td><td>Source Address</td><td>Radius</td><td>Sequence Number</td><td rowspan="2">Data payload</td></tr>
<tr><td colspan="4">Routing fields</td></tr>
<tr><td colspan="5">NWK Header</td><td>NWK Payload</td></tr>
</table>

Abbildung 3-13: NKW-Data-Frame

Und der für das NWK-Command-Frame wie in Abbildung 3-14.

<table>
<tr><td>Occtets:2</td><td>2</td><td>2</td><td>1</td><td>1</td><td>1</td><td>Variabel</td></tr>
<tr><td rowspan="2">Frame control</td><td>Destination Address</td><td>Source Address</td><td>Radius</td><td>Sequence Number</td><td rowspan="2">NKW Command Identifier</td><td rowspan="2">NWK Command Payload</td></tr>
<tr><td colspan="4">Routing fields</td></tr>
<tr><td colspan="5">NWK Header</td><td colspan="2">NWK Payload</td></tr>
</table>

Abbildung 3-14: NKW-Command-Frame

Die beiden darunter liegen Schichten MAC und PHY haben auch wieder einen eigenen Frame-Aufbau, der hier aber nicht weiter betrachtet werden soll, da dies den Umfang der Arbeit übersteigen würde. Es soll aber nicht unerwähnt bleiben, dass der Standard IEEE 802.15.4 sogenannte Beacons (Leuchtfeuer) zur Verfügung stellt, mit deren Hilfe Zeitabschnitte für einzelne Geräte oder Dienste reserviert werden können.

4 Design des Hardwaremodules CAtZi

Das CAtZi-Modul wird speziell für die Zusammenarbeit mit dem AKSEN-Board entwickelt, woraus resultiert, dass eine gute Anpassung an die Gegebenheiten des AKSEN-Boards vorgenommen wird. Die Anpassungen beziehen sich auf die Bereiche der mechanischen Gegebenheiten, wie auch der elektrischen. Grob kann CAtZi in fünf funktionelle Bereiche aufgeteilt werden (Abbildung 4-1). Die drei wichtigen Bereiche sind der CAN-Bus, ZigBee und der Mikrocontroller. Die beiden Bereiche Bedienelemente und Ein- und Ausgänge sind für die eigentliche Funktion nicht unbedingt notwendig.

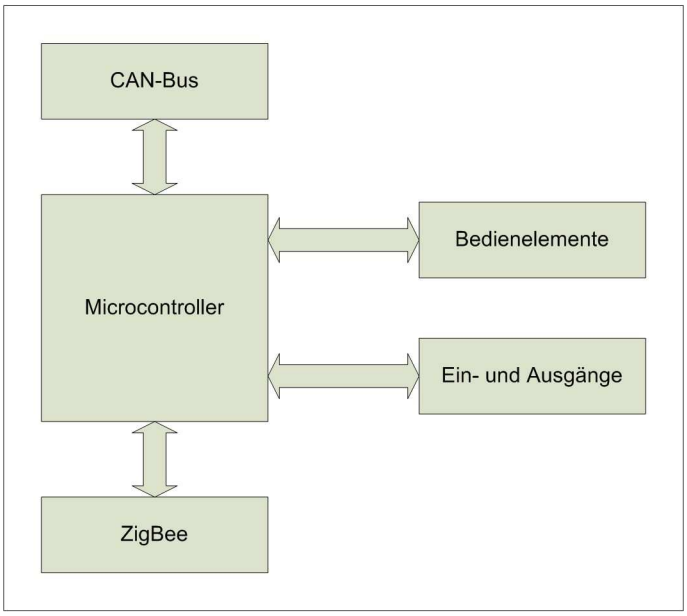


Abbildung 4-1: Blockschaltbild CAtZi

Bei den mechanischen Anforderungen geht es darum eine möglichst einfache Integration in die vorhandene Umgebung zu ermöglichen. Dazu gehören die Maße des Moduls, um so einen gestapelten Aufbau der Platinen zu ermöglichen, die

Anordnung der Steckkontakte um eine optimierte Leiterführung zu erlangen, sowie Taster und Schalter um eine übersichtliche Umgebung zu schaffen.

Auch das elektrische Design soll zudem das Arbeiten vereinfachen und Fehler vermeiden helfen. Bei einer zu der des AKSEN-Boards identischen Belegung der Kontakte von Stecker und Buchsen werden Verwechslungen durch den Einsatz einheitlicher Kabel vermieden und daraus resultierende Beschädigungen ausgeschlossen. Hier sind besonders die Spannungsversorgung und der CAN-Bus-Anschluss zu berücksichtigen, da hier das größte Gefahrenpotenzial vorliegt.

4.1 Die Funktionsbereiche

In den folgenden Abschnitten wird das Vorgehen für die einzelnen Sektionen erläutert.

4.1.1 CAN-Bus

Wie aus den Anforderungen hervorgeht, wird eine CAN-Schnittstelle zum Anschluss an das AKSEN-Board benötigt. Sie sollte eine einstellbare Datentransferrate haben, mit bis zu einem MBit arbeiten können und in der Lage sein nach dem CAN 2.0B Protokoll zu arbeiten, um so die volle Funktionsfähigkeit des AKSEN-Boards ausnutzen zu können. Eine gute Vorausfilterung der CAN-ID durch den CAN-Controller entlastet den Mikrocontroller, so dass dieser Nachrichten die ihn nicht betreffen gar nicht erst erhält und seine Resource für andere Aufgabe nutzen kann. Es gibt eine große Auswahl von CAN-Controllern, wobei ein häufig eingesetztes Modell mit großem Funktionsumfang, der SJA1000 von der Firma Philips, auf dem AKSEN-Board Verwendung findet. Mit ihm ist sowohl das Arbeiten nach dem Protokoll CAN 2.0A wie auch nach CAN 2.0B möglich. Darüber hinaus bietet er noch die folgenden Eigenschaften:

- 64-Byte Ring-Empfangs-Buffer.
- 13 Byte Sendepuffer zur Aufnahme einer kompletten CAN2.0B-Nachricht
- Bitraten bis zu 1-Mbit/s
- PeliCAN Mode. In diesem stehen noch folgende Funktionen zur Verfügung:
    - Lese- / Schreibzugriff auf den Errorzähler
    - Einstellbares Fehlerwarnschwelle

- Auslesen des letzten Fehlers
- Error Interrupt für jeden CAN-Bus-Fehler
- Interrupt bei verlorenen Arbitration mit Angabe der Bitposition
- Erweiterter Akzeptanzfilter (4-byte code, 4-byte mask)

Es wird zudem ein CAN-Treiber gebraucht, wobei üblicherweise zum SJA1000 der PCA82C250 oder PCA82C251 verwendet wird, die sich nur darin unterscheiden, bis zu welchen Spannungswerten gearbeitet werden kann. Zum Anschließen des CAN-Busses werden derselbe Steckertyp und dieselbe Belegung verwendet wie beim AKSEN-Board. Dabei handelt es sich um einen zehnpoligen Wannenstecker. Es würde auch ein sechspoliger Wannenstecker ausreichen, doch ist dieser teuerer. Weiter gibt es einen Abschlusswiderstand für den CAN-Bus, von 120 Ohm, der bei Bedarf zugeschaltet werden kann. Außerdem ist die Spannungsversorgung so zu entwerfen, dass folgende Betriebsmöglichkeiten bestehen:

- Die CAN Baugruppe wird vom Board versorgt
- Die CAN Baugruppe wird vom CAN-Bus versorgt
- Die CAN-Baugruppe wird vom Board versorgt und der CAN-Bus wird zusätzlich mit 5V gespeist
- CAtZi wird durch den CAN-Bus mit 5V versorgt

4.1.2 ZigBee

Für die Funkverbindung wird ZigBee eingesetzt. Da es eine sehr junge Technik ist, gibt es auf dem Markt zurzeit noch recht wenig Produkte. Im Rahmen dieser Arbeit kommt das CC2420EM Modul (Abbildung 4-2) der Firma ChipCon zum Einsatz, Das im Wesentlichen aus dem CC2420 desselben Herstellers besteht. Der CC2420 ist eine Single-Chip-ZigBee-Lösung für das 2,4 GHz Band, der dem IEEE 802.15.4 Standard konform ist. Er bietet:

- Übertragungsraten bis 250 kbit/s.
- 128 Byte Sendepuffer.
- 128 Byte Empfangspuffer.
- Batterie-Monitor.
- Geringer Stromverbrauch. (Beim Senden 17,4mA und beim Empfangen 19,7mA)

Abbildung 4-2: CC2420EM

Die Wahl dieses Bausteins wurde getroffen, da sowohl das CC2420DBK-Demonstration-Board-Kit wie auch das CC2420DK-Development-Kit aus demselben Hause an der HAW-Hamburg zur Verfügung stehen und die weiteren Leistungsmerkmale der verschieden Herstelle sich nicht wesentlich unterscheiden. Die Entscheidung für das CC2420EM Model im Gegensatz zu einer Implementierung des CC2420 auf der Platine hat den einfachen praktischen Grund, dass es den Baustein CC2420 nur in einem QLP-48 Gehäuse gibt, das eine Kantenlänge von 7x7 mm und einem Pinabstand von 0,50 mm aufweist. Die manuelle Bestückung einer Platine mit Komponenten dieser Größe ist unnötig aufwendig. Da es im Rahmen der Anforderungen nicht erforderlich ist dieses besonders kleine Bauteil zu verwenden, wurde zum Aufbau dieses Prototyps das CE2420EM-Modul gewählt, das auf das CAtZi-Modul aufgesteckt wird.

4.1.3 Mikrocontroller

Bei dem Mikrocontroller handelt es sich um einen ATMega128. Die Auswahl fiel dabei auf die ATMega-Reihe von Atmel, da es von der Firma ChipCon ein Referenz-Design mit dem ATMega128L gibt. Der deutlich günstigere ATMega32 bietet für einen Prototypen mit zusätzlichen Tastern und LEDs zu wenig digitale I/Os und ist auch vom verfügbaren Speicher eventuell nicht ausreichend. Die ATMega64 und ATMega128 sind nahezu identisch, mit dem Hauptunterschied, dass der zur Verfügung stehende Speicher beim ATMega64 kleiner ausfällt. Um für den Prototypen sicher zu gehen, wurde der größere Controller (ATMega128) gewählt, der sich im Preis auch nicht stark vom ATMega64 unterscheidet. Eine spätere Umstellung auf den ATMega64 sollte, so lange der Speicher ausreicht, problemlos möglich sein, da er pinkompatibel ist und alle wesentlichen Eigenschaften des ATMega128 ebenfalls aufweist. Hier eine kurze, unvollständige Auflistung der Eigenschaften des ATMega128:

- 8-Bit Mikrocontroller mit RISC-Architektur
- 128kByte In-System Reprogrammable Flash
- 4kByte EEPROM
- 4kByte SRAM
- Master / Slave SPI Interface für In-System Programming
- JTAG Interface
- Vier Timer/Counter
- 8 PWM Kanäle
- 8 Kanal 10-Bit ADC
- UART Interface
- Extern und Intern auslösbare Interrupts
- Sechs verschiedene Stomsparmodi

Ausführliche Informationen über den Microkontroller ATMega128 sind dem Datenblatt [D-AT-1] zu entnehmen.

4.2 Zusammenführung der Sektionen von CAtZi

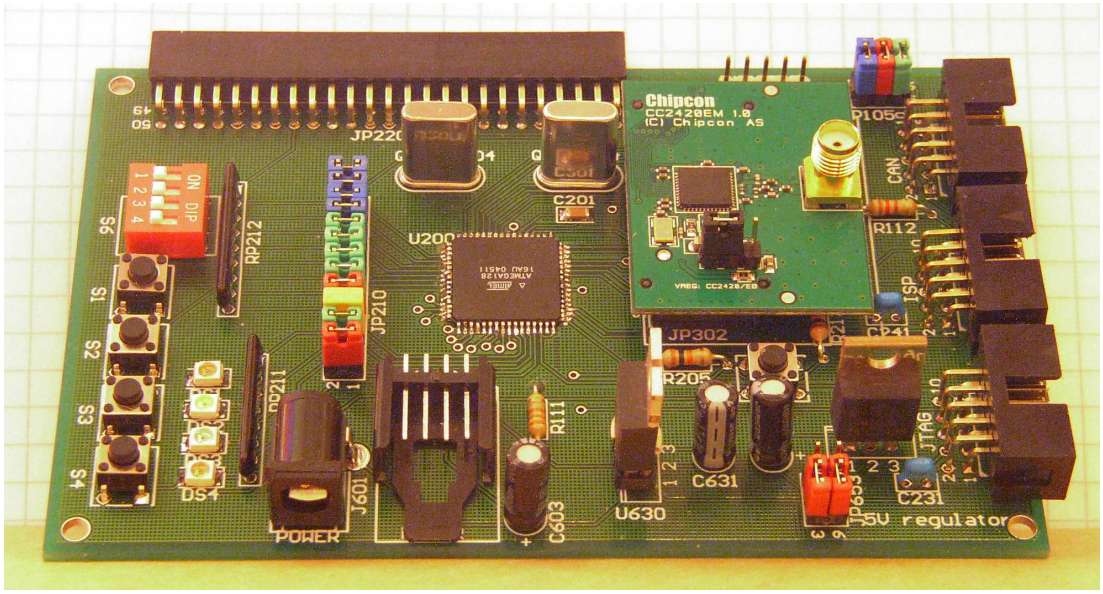


Abbildung 4-3: CAtZi Platine Oberseite

Die Anbindung des Zigbeemoduls an den Mikrocontroller wurde aus dem Referenzdesign vom Hersteller ChipCon [D-CC-1] übernommen und den hier erforderlichen Bedingungen angepasst. Für die CAN-Sektion wurde auf ein früheres Projekt der HAW-Hamburg, das Teile der Entwicklung des AKSEN-Boards beinhaltet [AKSE04], zurückgegriffen und die CAN-Anbindung weitestgehend übernommen. Somit wird auf verschiedene funktionierende Module zurückgegriffen, so dass die Funktionstüchtigkeit der Systeme bereits umfangreich getestet wurde.

Die erste nötige Anpassung war, nicht benötigte Module des ZigBee-Boards wie den zusätzlichen Speicher zu entfernen um die Datenleitungen des Mikrocontroller für den CAN-Controller nutzbar zu machen. Weiterhin musste eine Anpassung der Spannungspegel vorgenommen werden, da das ausgewählte ZigBee-Modul mit 3,3V arbeitet, während die ausgewählten Teile für die CAN-Schaltung mit 5V arbeiten. Es wurde entschieden, die Anpassung zwischen dem Mikrocontroller und dem ZigBee-Modul vorzunehmen statt zwischen dem Mikrocontroller und dem CAN-Controller. Dafür war folgender Grund ausschlaggebend: Den Mikrocontroller gibt es in zwei Varianten, als ATMega128 und ATMega128L, wobei die L Ausführung mit 3,3V

und die andere mit 5V arbeitet. Der Vorteil der 5V Variante ist, dass Pegel mit einer Spannung ab 3,0V als Highsignal an den Ports akzeptiert werden und damit eine Anpassung nur dann nötig ist, wenn es sich um einen Signallauf vom Microprozessor (Digital Output) zum ZigBee-Modul (Digital Input) handelt, der auch bidirektional sein kann. Daraus ergibt es sich, dass bei sechs der insgesamt zehn benötigten Verbindungen eine Spannungsanpassung nötig ist. Auch der CAN-Controller akzeptiert Highsignale ab ca. 3V, jedoch bleiben hier von den insgesamt 13 nötigen Verbindungen 10, die einer Anpassung bedürfen, was den Einsatz von zwei Konverterbausteinen oder eines einzelnen größeren bedarf, da diese üblicherweise acht oder 16 Bit breit sind. So ist bei der ersten Variante nur ein zusätzlicher Baustein nötig. Zum Einsatz kommt hier der Baustein SN74CB3T3245DW[1], ein 8-Bit-Bus-Switch, der eine Anpassung der Spannungspegel von 5V auf V_{CC} (hier 3,3V) übernimmt und bidirektional mit automatischer Richtungserkennung arbeitet, so dass auch die 3,3V Signale durchgereicht werden.
Die Kombination des SJA 1000 mit einem ATMega-Mikrocontroller kam schon in anderen Projekten für das AKSEN-Board zum Einsatz. Es wurde die Schaltung von [AKSN04] weitgehend übernommen und die nötigen Anpassungen wurden realisiert. Im genannten Projekt wurde ein ATMega32 verwendet, so dass das Design entsprechend angepasst werden musste, ebenso wie zur Verwendung des ChipCon-Referenzdesigns für das ZigBee-Modul CC2420EM.
Zum Betrieb wird noch die Spannungsversorgung benötig. Weiter sind einige Bedienelemente und Anschlüsse Sinnvoll die über die eigentliche Kommunikations-aufgabe hinausgehen, aber zum Test des Prototyps hilfreich sind.

4.2.1 Spannungsversorgung

Im Bereich „Power“ (Abbildung 4-5 Seite 37 oben links) werden die benötigten Spannungen bereitgestellt, die das Board zum Betrieb benötigt, also 3,3V und 5V. Die Versorgungsspannung wird über einen der beiden Powerconnectoren (J601 als 2,1mm Hohlstecker-Einbaubuchse oder J602 als vierpolige gewinkelte Stiftleiste) (Abbildung 4-4 links, bzw. rechts) eingespeist. Es werden hier zwei verschieden Stecker angeboten, da eventuell geplant ist, die auf dem Roboter momentan verwendeten Hohlstecker durch vierpolige Stiftleisten auszutauschen. Als Versorgungsspannung sind entweder stabilisierte 5V oder unstabilisierte 6,5V bis 15V anzulegen. Bei Betrieb mit einer Spannung zwischen 6,5V und 15V werden die benötigten 5V durch einen Linearregler bereitgestellt. Die Einstellung, wie das Board

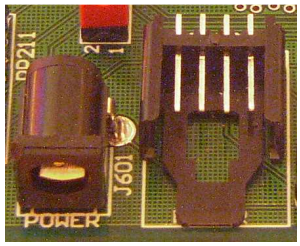


Abbildung 4-4: Powerconnectoren

[1] Datenblatt ist im Anhang zu finden [D-TI-1].

mit Spannung versorgt wird, erfolgt über Jumper. Die 3,3V werden immer durch einen Linearregler erzeugt, der entweder mit den 5V oder den 6,5V bis 15V gespeist wird.

4.2.2 Bedienelemente

Die Taster, Schalter und LEDs können für verschiedene Zwecke genutzt werden, z.B. als Hilfsmittel beim Debuggen, als Statusanzeigen im Betrieb, zur Auswahl von Betriebsmodi und zum Einstellen von Adressen. Zur Verfügung stehen fünf Mikroschalter, wobei einer fest mit der RESET-Leitung verbunden ist, um einen Hardreset durchführen zu können. Weiterhin gibt es vier DIP-Schalter, sowie vier LEDs (2 rote, 1 gelbe, 1 grüne)(Tabelle 4-1). Diese können alle, bis auf den Reset-Taster, durch Jumper vom Mikrocontroller getrennt werden, um so die digitalen I/O-Ports des Mikrocontrollers extern nutzen zu können. Diese I/O-Ports werden, in 4.2.3 genauer erläutert, über die Buchse JP220 zur Verfügung gestellt.

Belegung	ID	Gerät
	DS1	LED rot
	DS2	LED gelb
	DS3	LED grün
	DS4	LED rot
	S1	Taster
	S2	Taster
	S3	Taster
	S4	Taster
	S5	Resettaster
	S6	4 DIP Schalter

Tabelle 4-1: Taster, Schalter und LEDs

4.2.3 Steckverbindungen

Es existieren mehreren Steckverbindungen außer den bisher erläuterten Steckverbindungen für Strom und den CAN-Bus. Die anderen werden für den eigentlichen Betrieb nicht benötigt, stellen jedoch gute Möglichkeiten dar, das CAtZi über die eigentliche Aufgabe hinaus zu nutzen. Tabelle 4-2 gibt den Bezeichner der jeweiligen Verbindung mit Steckertyp und Funktion an.

Steckverbindungen		
JP 220	2x25-polige Buchsenleiste	12 x Digitaler I/O mit 5V und GND
JC 230	Wannenstecker, 10-polig	JTAG
JC 240	Wannenstecker, 10-polig	ISP
JC 250	1x5-polige Stiftleiste	RX/TX (Seriell)

Tabelle 4-2: Steckverbindungen

Über eine 50-polige Buchsenleiste können an die zwölf digitalen I/O Ports, die auch für die Taster, Schalter und LEDs verwendet werden und die per Jumper abgeschaltet werden können, externe Aktoren und Sensoren angeschlossen werden. Für jede

Datenleitung ist eine separate Spannungsversorgung vorgesehen, so dass die Sensoren und Aktoren hierüber mit 5V versorgt werden können und keine eigene Spannungsversorgung benötigen. Der Aufbau ist so gestaltet, dass die Stecker, die für das AKSEN–Board zum Einsatz kommen, auch hier verwendet werden können. Das Schema dabei sieht vor, dass folgende Reihenfolge von links nach rechts, sowohl für die obere wie auch für die untere Reihe verwendet wird: GND, Vcc=5V, nicht belegt, Datenleitung.

Zum Programmieren, Emulieren und Debuggen des Mikrocontrollers ist die JTAG[2] Schnittstelle des Mikrocontrollers nach außen geführt. Die Schnittstelle ist über einen 10-poligen Wannenstecker bereitgestellt. Zum Betrieb wird ein sogenanntes JTAG-Interface benötigt, wie z.b. das AVR JTAGICE mkII, welches ebenfalls von der Firma Atmel angeboten wird.

Eine günstige Möglichkeit den Mikrocontroller zu programmieren stellt die ISP Schnittstelle dar. Diese ist ebenfalls über einen 10-poligen Wannenstecker zu erreichen. Auch hier wäre eine 6-polige Ausführung ausreichend, es wurde aber aus den oben bereits erwähnten Gründen ein 10-poliger Stecker verwendet. Die Belegung entspricht der von Atmel verwendeten Belegung.

Da ein direkter Austausch zwischen CAN und Zigbee nicht möglich ist, muss eine Anpassung stattfinden. Das betrifft vor allem die Adressierung, denn CAN arbeitet ganz ohne Adressen. Die Nachrichten werden stattdessen mit einer ID versehen, anhand der jedes Gerät entscheidet, ob diese Nachricht von Interesse ist und verarbeitet werden soll oder die Nachricht verworfen werden kann. Zigbee hingegen arbeitet mit 16 oder 64 Bit großen Adressen. Weiterhin sind die Größen der Nutzdaten pro Frame unterschiedlich: Während CAN nur maximal 8 Byte pro Frame übertragen kann, sind es bei ZigBee, genauer im PHY-Layer, bis zu 127 Byte, wovon einige Byte für den Overhead der darüber liegenden Schichten abgezogen werden müssen.

[2] JTAG steht für Joint Test Action Group und wird auch Boundary Scan Test genannt.

4.2.4 Der Schaltplan

In Abbildung 4-5 wird der aus dem Design entstandene Schaltplan gezeigt. Hier finden sich die sechs verschiedene Bereiche wieder, die anfangs im Blockschaltbild (Abbildung 4-1) eingeführt und anschließend erklärt wurden.

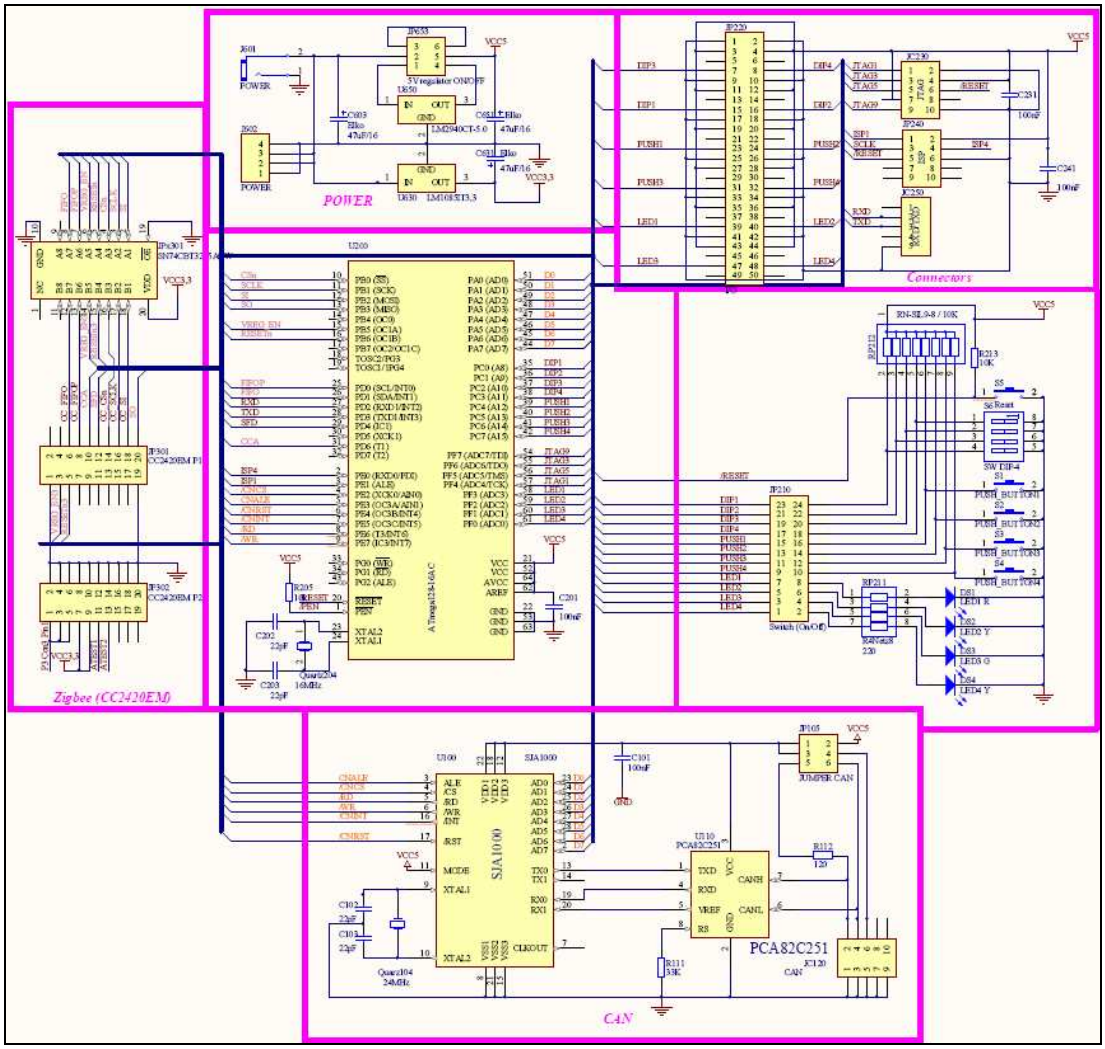


Abbildung 4-5: Schaltplan von CAtZi Rev. 1.0

5 Software Design

Wie in 4.2.3 erwähnt, ist eine Eins-zu-Eins-Umsetzung zwischen Zigbee und CAN nicht möglich, was ein übergeordnetes Protokoll erforderlich macht. Dafür werden verschiedene Arten von Nachrichten benötigt, die im Folgenden kurz beschrieben werden.

- **Die Datennachricht** dient zum Versenden oder Anfordern von Nutzdaten. Dabei sollte es die Möglichkeit geben, Standardnachrichten vereinfacht zu versenden.
- **Die Kontrollnachricht** dient zum einen dazu, das Netzwerk zu organisieren, (z.B. Anfragen nach den verfügbaren Teilnehmern) und zum anderen dazu, das CAtZi zu konfigurieren.
- **Die Errornachricht** ist für die Fehlerbehandlung zuständig.

Wenn man die Aufgaben von CAtZi ins OSI-Schichten Model übertragen möchte, deckt CAtZi die transportorientierten Schichten eins bis vier ab. Die Kommunikation zwischen den einzelnen Endgeräten verläuft dann über das zu entwickelnde Protokoll ab, das den Namen MAU[1] enthält. Der Verlauf der Komunikation wird in
Abbildung 5-1 gezeigt.

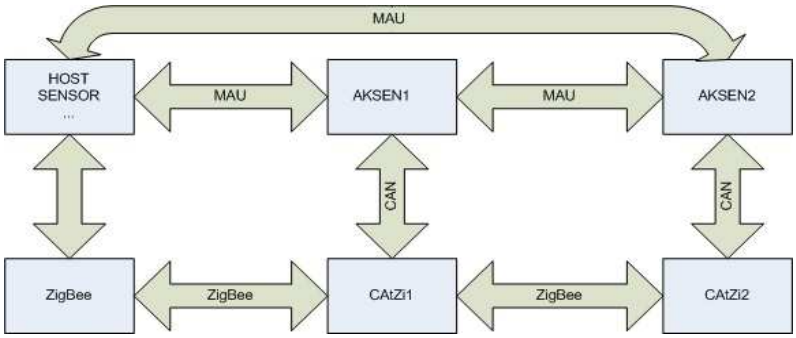


Abbildung 5-1: Kommunikationswege der Protokolle

[1] In Anlehnung an die Kommunikation zwischen Katzen (CAtZi)

Für simple Geräte, die ihre Daten nicht per MAU übermittel können, soll CAtZi eine minimale Host-Funktion bieten, um die Daten zur Verfügung zu stellen. Die zu erfüllenden Dienste des Servers sollen sein:

- Es sollte möglich sein, die vom Gerät gesendeten Daten an eine zuvor festgelegte Empfängeradresse zu senden. Dabei sollte die Adresse sowohl die eines einzelnen Empfängers, als auch die von einer Gruppe oder einer Fraktion sein können.
- Das AKSEN-Board soll die Möglichkeit haben, das Gerät nach Daten zu fragen. Dabei ist folgende Einschränkung zu berücksichtigen: Gehört das Gerät nicht der Empfangsadresse an, bekommt es keine Antwortnachricht.

Eine Darstellung der Kommunikationswege für einfache Geräte findet sich in Abbildung 5-2.

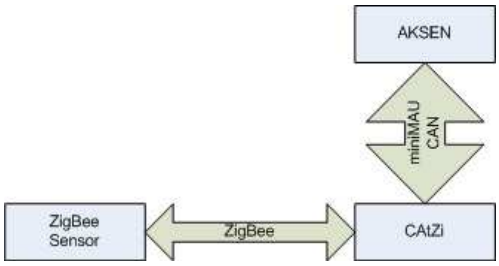


Abbildung 5-2: Kommunikationswege bei Verwendung einfacher Geräte

Der im Folgenden vorgestellte Entwurf definiert zum einen das Protokoll MAU, dessen Implementation und die Konfigurationsmöglichkeiten, die vom Protokoll zur Verfügung gestellt werden. Des Weiteren werden einige Richtlinien zur Verwendung von MAU gegeben. Diese betreffen sowohl die Vergabe der Adressen, wie auch die Verwendung der Messagetypen. Im Entwurf findet sich auch die Realisierung der Konfiguration von CAtZi über CAN und ZigBee. Abschließend wird gezeigt, wie man einfache ZigBee-Geräte in das System integrieren kann, auch wenn diese nicht in der Lage sind über MAU zu kommunizieren.

5.1 Protokoll MAU

Das Protokoll MAU hat zwei verschiedene Arten von Frames: Datenframes und Konfigurationsframes. Beide Typen werden im Folgenden erläutert.

5.1.1 Datenframe

Für den Datentransfer ist es zunächst nötig, dass die Parameter von MAU festgelegt werden. Diese Parameter gelten für alle Teilnehmer, die miteinander kommunizieren sollen. Wie die Konfiguration vorgenommen werden kann wird in 5.1.3 erläutert. Hier werden zunächst die Parameter und der Datenframeaufbau erklärt.

5.1.1.1 Message-Identifier

Der erste Parameter ist die Größe des verwendeten CAN-ID-Bereiches, im Folgenden mit Message Identifier (MsgID) bezeichnet, und dessen Aufteilung, wobei verschiedene Regeln für die MsgID eingehalten werden müssen. Die MsgID setzt sich aus drei Teilen zusammen: Dem Empfänger-Identifier (EmpfID), dem Sender-Identifier (SendID) und dem Nachrichtentyp-Identifier (TypeID). Der Aufbau der EmpfID und der SendID muss identisch sein. Die EmpfID und SendID, die als Adresse dienen, sind wiederum aus drei Teilen zusammengesetzt: der Fraktion, der Gruppe und dem Spieler. Dabei wird der Fraktion und der Gruppe eine Bitanzahl zwischen null und vier zugewiesen, für die Spieler liegt die Bitanzahl zwischen eins und sechs. Abhängig vom Spieltyp und der Strategie kann unter Umständen auf die Fraktion und/oder Gruppe verzichtet werden. Die Aufgaben der drei Teile sind:

Fraktionen sollten genutzt werden, wenn es verschiedene Gruppen mit Untergruppen gibt, und die Mitglieder der gleichen Untergruppen ähnliche Aufgaben übernehmen. Um beim Beispiel Roboterfußball zu bleiben, könnten die Fraktionen die verschieden Mannschaften sein.

Gruppen bilden funktionale Einheiten. Beim Fußball zu nutzen wäre das z.B. der Sturm, das Mittelfeld, die Abwehr und der Torwart.

Spieler sind die konkreten Akteure der einzelnen Gruppen und Fraktionen. Das könnten dann z.B. Mittelstürmer, linkes Mittelfeld, Torwart usw. sein.

Der TypeID wird ebenfalls eine Bitgröße zwischen eins und acht zugeordnet, wobei diese die maximal mögliche Anzahl der verschieden Nachrichten bestimmt. Die Reihenfolge, in der die Nachricht zusammengesetzt ist, ist TypeID, EmpfId und SendID. Eine schematische Darstellung des Aufbaus findet sich in Abbildung 5-3.

MsgID:						
TypeID	EmpfID			SendID		
	1-14 Bit			1-14 Bit		
	Fraktion	Gruppe	Spieler	Fraktion	Gruppe	Spieler
1-8 Bit	0-4 Bit	0-4 Bit	1-6 Bit	0-4 Bit	0-4 Bit	1-6 Bit
	EmpfID und SendID müssen Identisch aufgebaut sein.					

Abbildung 5-3: Aufbau des Message-Identifier

Es gibt noch zwei weitere Werte, die nur insofern mit dem Protokoll zu tun haben, als dass sie die genaue Position der MsgID in der CAN-ID angeben und nur für den Betrieb des CAN-Buses nötig sind, um eine individuelle Vergabe von CAN-IDs für Nachrichten zu erlauben. Auch müssen diese Werte nicht bei allen Teilnehmern gleich sein. Der erste dieser Werte ist die Bitstellenanzahl vor der MsgID und die sich daraus rechnerisch ergebende Bitstellenanzahl nach der MsgID, die zusammen die CAN-ID bilden. Werden nicht alle Stellen von der MsgID belegt, kann somit eine individuelle Adressierung auf jedem Roboter festgelegt werden. Dabei ist zu beachten, dass die Gesamtanzahl an Bitstellen bei CAN2.0B auf 29 festgelegt ist. Daraus ergibt sich auch, dass eine Konfiguration mit Maximalwerten nicht möglich ist da 14Bit+14Bit+8Bit = 36Bit ergeben. Das ist beabsichtigt, um flexibel den Anforderungen von verschiedenen Anwendungen gerecht zu werden, wobei natürlich nicht jede Eventualität berücksichtigen werden kann.

5.1.1.2 Adressen

Die SendID und EmpfID stellen die Adressen vom Sender und Empfänger dar. Bei der Vergabe der Adressen ist nun zu berücksichtigen, dass der höchste Wert in jeder Kategorie, der abhängig von der zur Verfügung gestellten Bitanzahl ist, eine Mitteilung an alle Angehörigen der entsprechenden Kategorie darstellt. Eine Ausnahme bildet hier die Bitanzahl 1. Da es hier nur 2 Teilnehmer gibt, ist eine Funktion „senden-an-alle“ durch den höchsten Wert nicht sinnvoll. Aus diesen Bedingungen ergeben sich für die einzelnen Kategorien die in Tabelle 5-1 angegebenen, zur Verfügung stehenden Kapazitäten.

Kategorie	Min	Max
Fraktionen:	0	15
Gruppen:	0	15
Spieler:	2	63
Gesamt:	2	14175

Tabelle 5-1: Größen der Adresskategorien

Daraus ergibt sich, dass bei 14175 Teilnehmern nur zwei verschiedene Nachrichtentypen versendet werden können, wobei später noch Hinweise gegeben werden, wie die Nachrichtentypen eingesetzt und die tatsächlich unterschiedlichen Nachrichtentypen erhöht werden können. Bei der maximalen Ausnutzung der Nachrichtentypen von 256 bleiben, je nach eingestellten Parametern, maximal 945 adressierbare Teilnehmer, wobei ein Bit ungenutzt bleibt, so dass das bessere Verhältnis wohl bei 128 Nachrichtentypen und damit einer maximalen Teilnehmerzahl von 1890 liegt. In diesem Fall werden alle 29 Bit ausgenutzt und so eine große Anzahl an Nachrichten und Teilnehmern zur Verfügung gestellt, wobei die Entscheidung letztendlich immer der Situation entsprechend gefällt werden muss.

Um das zu verdeutlichen folgen einige Beispiele, wobei für A gilt, dass die Fraktion (F) eine Länge von einem, die Gruppe (G) eine Länge von zwei und der Spieler (S) eine Länge von drei Bit hat. Für B gilt, dass die Fraktion eine Länge von zwei, die Gruppe eine Länge von drei und der Spieler eine Länge von zwei Bit hat. Es wird hier, wie in der Informatik üblich, von null an gezählt. Gibt es z.B. drei Spieler, sind diese mit 0,1 und 2 durchnummeriert.

Beispiel 1: Eine Nachricht an alle Spieler aller Gruppen und Fraktionen schicken. (Bei A sind zwei Nachrichten nötig):

A					
F	G	G	S	S	S
0	1	1	1	1	1
1	1	1	1	1	1

B						
F	F	G	G	G	S	S
1	1	1	1	1	1	1

Abbildung 5-4: Nachricht an alle Spieler

Da es bei A nur zwei Fraktionen gibt, gilt hier die Sonderregel, dass nicht mit dem höchsten Wert an alle gesendet wird.

Beispiel 2: Eine Nachricht an alle Spieler der Gruppe 2 von Fraktion 1:

A					
F	G	G	S	S	S
1	1	0	1	1	1

B						
F	F	G	G	G	S	S
0	1	0	1	0	1	1

Abbildung 5-5: Nachricht an eine Gruppe

Wie man sieht, werden alle Bits der Spieler auf 1 gesetzt, was den höchsten Wert ergibt und somit eine Nachricht an alle Spieler in dieser Kategorie ergibt.

Beispiel 3: Eine Nachricht an Spieler 3 der Gruppe 2 der Fraktion 0 senden:

A					
F	G	G	S	S	S
0	1	0	0	1	1

B						
F	F	G	G	G	S	S
-	-	-	-	-	-	-

Abbildung 5-6: Nachricht an einen Spieler

Bei B ist dieser Befehl nicht möglich, da es nur die Spieler 0 bis 2 gibt. Eine Drei im Spieler-Adress-Feld wäre wieder eine Nachricht an alle Spieler der Gruppe 2 von Fraktion 0.

Beispiel 4: Eine Nachricht an alle Spieler aller Fraktionen in der Gruppe 1 senden:

A					
F	G	G	S	S	S
0	0	1	1	1	1
1	0	1	1	1	1

B						
F	F	G	G	G	S	S
1	1	0	0	1	1	1

Abbildung 5-7: Nachricht an alle Gruppen 1 aller Fraktionen

Bei A sind wieder 2 Nachrichten nötig, da es auch hier nicht möglich ist, an beide Fraktionen auf einmal zu senden.

5.1.1.3 Type-Identifier

Der Type-Identifier gibt an, welche Bedeutung die Nachricht hat. Dabei sind die Bedeutungen frei definierbar. Mit den Bedeutungen sind z.B. die aktuellen Koordinaten des Balles gemeint, aber auch, ob es sich um eine Fehlermeldung handelt. Dennoch sollen hier einige Regeln und Richtlinien gegeben werden, um einen eigenen Aufbau zu erleichtern.

Typen für Anfragen von Daten sollten in der Regel nicht definiert werden, sondern, wie bei CAN üblich, über die Länge des Datenpaketes geregelt werden, sodass ein Datenpaket mit der Länge Null immer eine Anfrage bedeutet. Die Länge kann den Typ dabei noch detaillierter kodieren. Sollen z.B. die Ball-Koordinaten übertragen werden und jede Achse hat einen zwei Byte großen Wert, würde bei einer Länge von zwei nur der Wert der X-Achse, bei eine Länge von vier X- und Y-Achse und bei einer Länge von sechs X-, Y- und Z-Achse übertragen werden. Es könnte auch der verwendete Datentyp damit kodiert werden. Länge zwei für unsigned short, Länge drei für signed short, Länge vier für unsigned long und Länge fünf für signed long. Eine weitere Möglichkeit wäre, die Anzahl der Wegpunkte zu übermitteln. Die anzusteuernden Punkte werden mit XY-Koordinaten angegeben und für jede Achse

ist ein Byte vorgesehen. So wäre bei einer Länge von zwei ein Wegpunkt anzufahren und bei einer Länge von acht vier Wegpunkte. Es gibt hier eine Vielzahl von Möglichkeiten, die Länge als Zusatzinformationen auszuwerten. Man sollte der Übersichtlichkeit wegen aber nur logisch verknüpfte Informationen zusammenfassen.

Der Aufbau der Typen sollte nach Möglichkeit in logische Kategorien zusammengefasst werden. Dabei sollte die wichtigste Gruppe den niedrigsten Wert zugewiesen bekommen, um die höchste Priorität zu erhalten. Das ist auch der Grund, warum die TypeID als erstes übermittelt wird. Denn so gibt die Art der Nachricht die Dringlichkeit an und nicht etwa der Empfänger oder Sender. Man sollte dabei jedoch immer bedenken, dass Prioritäten nur im Bereich der CAN-Komunikation zum Tragen kommen.

5.1.2 Richtlinien

Die Richtlinien werden für die folgenden drei Teilbereiche SendID & EmpfID, TypeID und Schiedsrichternachrichten vergeben. Diese Richtlinien müssen entsprechend der Gegebenheiten interpretiert werden.

5.1.2.1 SendID & EmpfID

Werden mehr als zwei Fraktionen (F) genutzt und gibt es dabei Rollen, die nicht den einzelnen Mannschaften zugeordnet werden können, wie z.B. Kampfrichter oder Sensoren, die ihre Daten für alle zur Verfügung stellen, sollten diese in der Fraktion null zusammen gefasst werden. Dabei sollte der Schiedsrichter, soweit vorhanden, in Gruppe (G) null sein und die Spielernummer (S) null erhalten. Die weitere Aufteilung ist Tabelle 5-2 (eins bis sechs) zu entnehmen. Auf die restlichen Fraktionen werden dann die Mannschaften verteilt. Ist es erlaubt, dass Mannschaften vom Spielfeldrand her unterstützt werden, z.B. durch einen Trainer oder von Sensoren, sollten diese nach Möglichkeit eine eigene Gruppe erhalten. Die Gruppe der Externen sollte die höchstmögliche Zahl erhalten. Die Einteilung der Mannschaften findet sich ebenfalls in der Tabelle 5-2 (sieben bis zehn). Dabei ist nur der Aufbau einer einzelnen Mannschaft aufgezeigt, bei weiteren Mannschaften wird genauso verfahren.

SendID & EmpfID										
Nr.	F_X	F_1	F_0	G_Z	G_1	G_0	S_Z	S_1	S_0	*Beschreibung*
1	0	0	0	0	0	0	0	0	0	Schiedsrichter
2	0	0	0	0	0	0	0	0	1	Linienrichter
3	0	0	0	0	0	0	x	x	x	weitere Kampfrichter o. ähnliches
4	0	0	0	0	0	1	0	0	1	Ortsdaten 1
5	0	0	0	0	0	1	0	1	0	Ortsdaten 2
6	0	0	0	0	1	x	x	x	x	Sonstige Sensoren
7	x	x	1	0	0	0	0	0	0	Torwart oder Verteidigung inkl. Torw.
8	x	x	1	0	0	1	x	x	x	Feldspieler
9	x	x	1	1	1	0	0	0	0	Trainer
10	x	x	1	1	1	0	x	x	x	Ext. Sensoren der Mannschaft

Tabelle 5-2: Richtlinien SendID- & EmpfID-Vergabe

5.1.2.2 TypeID

Die Vergabe der TypeID sollte nach Wichtigkeit der Nachrichten erfolgen, wobei die Nachrichten mit zunehmender Zahl unwichtiger werden Null hat die oberste Priorität. Dabei sollten die folgenden Gruppen eine Einteilung erleichtern. Die Aufzählung erfolgt in der Reihenfolge der Wichtigkeit:
Reglementierendes sind vor allem Nachrichten des Schiedsrichters. Wird ein Schiedsrichter verwendet, sollte dieser seine Nachricht mit der TypeID null versenden.
Steuererrelevantes sind Datennachrichten, die von den Robotern benötigt werden, um agieren zu können. Dazu gehören z.B. die Koordinaten des Balles und/oder der Spieler.
Spieler-Spieler-Interaktion betrifft die Abstimmung der Spieler untereinander, um z.B. taktische Spielzüge wie Passspiele zu ermöglichen.
Interaktion mit der Restwelt sollte alle Nachrichten enthalten, die nicht dringend für das Spielgeschehen benötigt werden, wie z.B. Anweisungen des Trainers, die Taktik zu ändern oder die Synchronisation von (Orts-) Daten.
Die Anordnung ist noch einmal in Tabelle 5-3 aufgezeigt.

TypeID				
0	0	0	Schiedsrichter-Mitteilungen	Wichtig
			...	^
			Steuerrelevante Daten	
			wie z.B. Ballkoordinaten	
			...	
			Spieler-Spieler-Interaktion	
			...	
			Interaktion mit der Restwelt	
			z.B Trainer	v
1	1	1	...	Unwichtig

Tabelle 5-3: Richtlinien zur TypeID

5.1.2.3 Schiedsrichter-Nachrichten

Ein Identifier der TypeID sollte normalerweise für Schiedsrichter-Nachrichten ausreichen. Das erste Byte der Nachricht (DB0) enthält dann die Anweisung entsprechend Tabelle 5-4. Die weiteren Daten-Bytes werden für zusätzliche Informationen genutzt. Enthält die Beschreibung ein „für (DBx-y)“ so sind damit Daten-Bytes für die Adresse gemeint. Hier steht entsprechend die Adresse eines konkreten Spielers oder die einer ganzen Mannschaft. Die Nachrichten mit einem DB0-Wert größer 0x0F sind frei definierbar. Hier könnten z.B. auch kombinierte Anweisung definiert werden, wie „gelbe Karte für (DB1-2) → Freistoß für (DB2-4)“.

Schiedsrichter Nachrichten									
länge	DB 0								Beschreibung
0	---								Nicht belegt
1	0	0	0	0	0	0	0	0	Start/ Anpfiff / Anstoß
2-3	0	0	0	0	0	0	0	1	gelbe Karte für Spieler (DB1-2)
2-3	0	0	0	0	0	0	1	0	rote Karte für Spieler (DB1-2)
2-3	0	0	0	0	0	0	1	1	gelb/rote Karte für Spieler (DB1-2)
2-3	0	0	0	0	0	1	0	0	Verwarnung für Spieler (DB1-2)
2-3	0	0	0	0	0	1	0	1	Direkter Freistoß Mannschaft (DB1-2)
2-3	0	0	0	0	0	1	1	0	Indirekter Freistoß für Mannschaft (DB1-2)
2-3	0	0	0	0	0	1	1	1	Abseits Spieler (DB1-2)
2-3	0	0	0	0	1	0	0	0	Abstoss für Mannschaft (DB1-2)
2-3	0	0	0	0	1	0	0	1	Ecke rechts für Mannschaft (DB1-2)
2-3	0	0	0	0	1	0	1	0	Ecke links für Mannschaft (DB1-2)
2-3	0	0	0	0	1	0	1	1	Einwurf für Mannschaft (DB1-2)
1	0	0	0	0	1	1	0	0	Spielunterbrechung
>1	0	0	0	0	1	1	0	0	Spielunterbrechung für x Sec
2-3	0	0	0	0	1	1	0	1	Tor für Mannschaft (DB1-2)
4-5	0	0	0	0	1	1	0	1	Tor für Mannschaft (DB1-2) / Ergbniss (DB2-4)
2	0	0	0	0	1	1	1	0	Restspielzeit in Sekunden (normal)
3	0	0	0	0	1	1	1	0	Restspielzeit in Millisekunden (normal)
4	0	0	0	0	1	1	1	0	Restspielzeit in Sekunden (Gesamt / Verlängerung)
6	0	0	0	0	1	1	1	0	Restspielzeit in Millisekunden (Gesamt / Verlängerung)
1	0	0	0	0	1	1	1	1	Spielende
x	>0x0F								Nicht belegt

Tabelle 5-4: Richtlinien für Schiedsrichter Nachrichten

5.1.3 Konfigurationsnachrichten im MAU

Konfigurationsnachrichten bekommen eine eigene Adresse, die unabhängig ist von der MsgID. Werden für die MsgID alle 29 Bit verwendet, so ist eine Konfiguration über den CAN-Bus nicht möglich. Das ZigBee-Modul kann in einen Zustand versetzt werden, in dem es keine Konfigurationsnachrichten mehr annimmt, um Manipulationsversuche zu erschweren. Die ersten acht Bit einer Konfigurationsnachricht geben die Konfiguration (KonfID) an. Ist die Länge der Nachricht gleich eins[2], wird der entsprechende Wert der KonfID abgefragt. Tabelle 5-5 listet die verschiedenen Nachrichtentypen auf.

Msg	**Beschreibung**	**Wert**
	CAN-Bus-Bereich	
0x01	CAN-Akzeptanz-Bereich setzen / ändern	DB1-DB3&D4(0-3) enthalten die Maske
0x02	CAN-Akzeptanz-Code setzen / ändern	DB1-DB3&D4(0-3) enthalten den Code
0x03	CAN-Akzeptaz-Code und –Maske setzen / ändern.	DB1-DB3&D4(0-3) enthalten die Maske. DB4(4-7)&DB5-DB7 enthalten den Code
0x04	CAN-Übertragungsrate ändern	DB1 = CAN_BUS_TIMING0 DB2 = CAN_BUS_TIMING1
0x05	Error	Fehlercode
0x06 -0x3F	Reserviert für zukünftige Erweiterungen	
	ZigBee Bereich	
0x41	ZigBee Type einstellen	DB0(4-7) = PAN-Type
0x42	ZigBee-PAN-Identifier und/oder 16-Bit Adresse vorgeben	DB1-DB2 = PAN-ID(!=0x00) DB3-DB4 = 16Bit-Adr (!=0x00) Bei 0x00 keine Änderung.
0x43	ZigBee-64-Bit Adresse vorgeben Teil1	DB1-DB5 = Bit 00-31
0x44	ZigBee-64-Bit Adresse vorgeben Teil2	DB1-DB5 = Bit 32-63
0x45	Error	Fehlercode
0x46 -0x7F	Reserviert für zukünftige Erweiterungen	
	MAU-Bereich	
0x81	MsgID-Längen	DB1: Länge der oberen CAN-ID(0-26) DB2: Länge TypeID (1-8) DB3: Länge Fraktion (0-4) DB4: Länge Gruppe (0-4) DB5: Länge Spieler (1-6)
0x82	CAN-ID	DB1-m[3]: Wert der oberen CAN-ID DBx-y: Wert der unteren CAN-ID
0x83 -0xFF	Reserviert für zukünftige Erweiterungen	

Tabelle 5-5: Erläuterung der KonfID

[2] Entspricht einer Länge von einem Byte
[3] m = {2,3}; x = m+1; y = x+m-1 => Länge der CAN-Nachricht = {5,7}

5.2 Sequenzdiagramm

Die Abbildung 5-8 zeigt den Ablauf einer Anfrage nach Koordinaten mit entsprechender Antwort zwischen zwei AKSEN-Boards über das CAtZi-Modul.

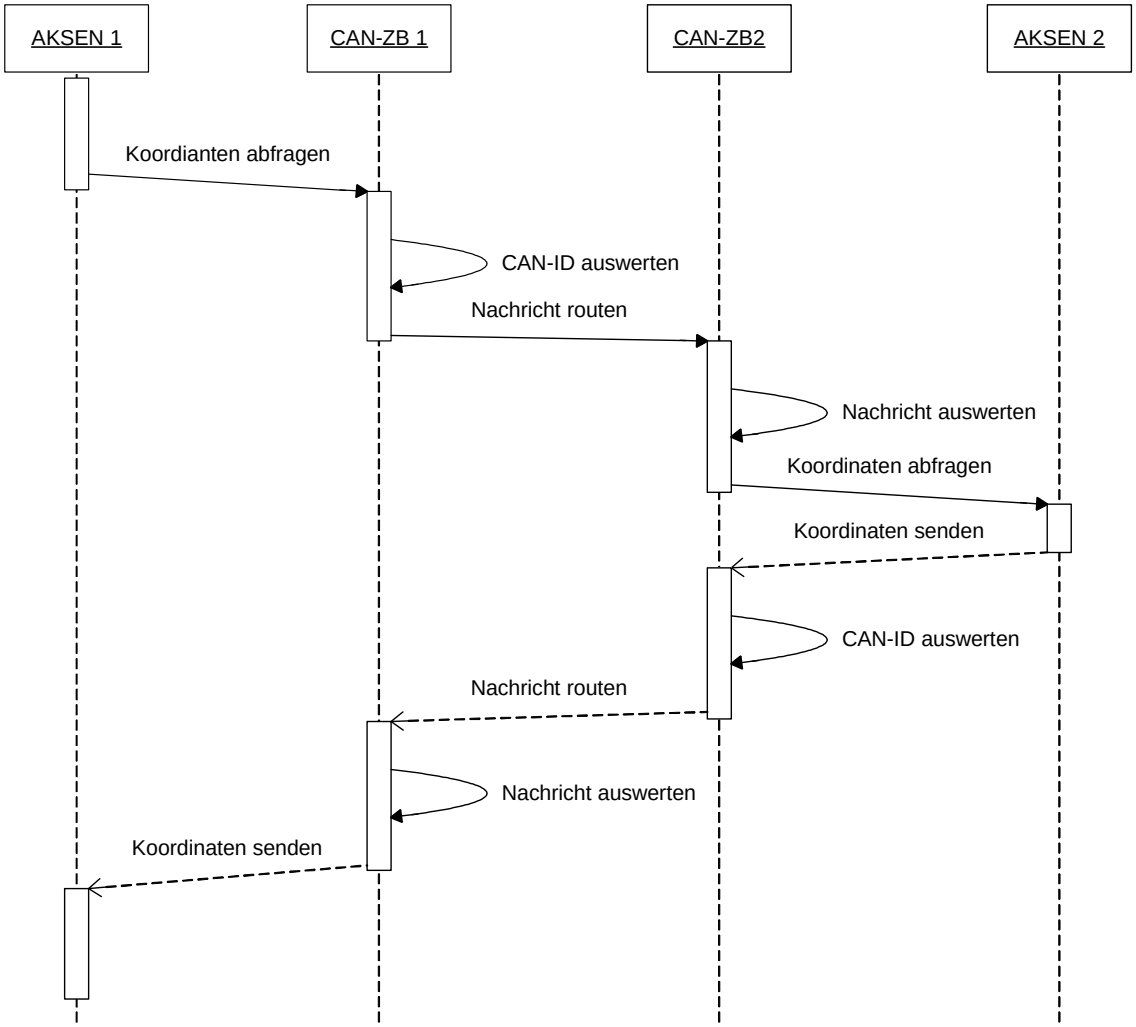


Abbildung 5-8: Sequenzdiagramm

5.3 Harel-Automat

Der in Abbildung 5-9 gezeigte Automat zeigt, wie eine eingetroffene Nachricht vom CAtZi-Modul verarbeitet wird.

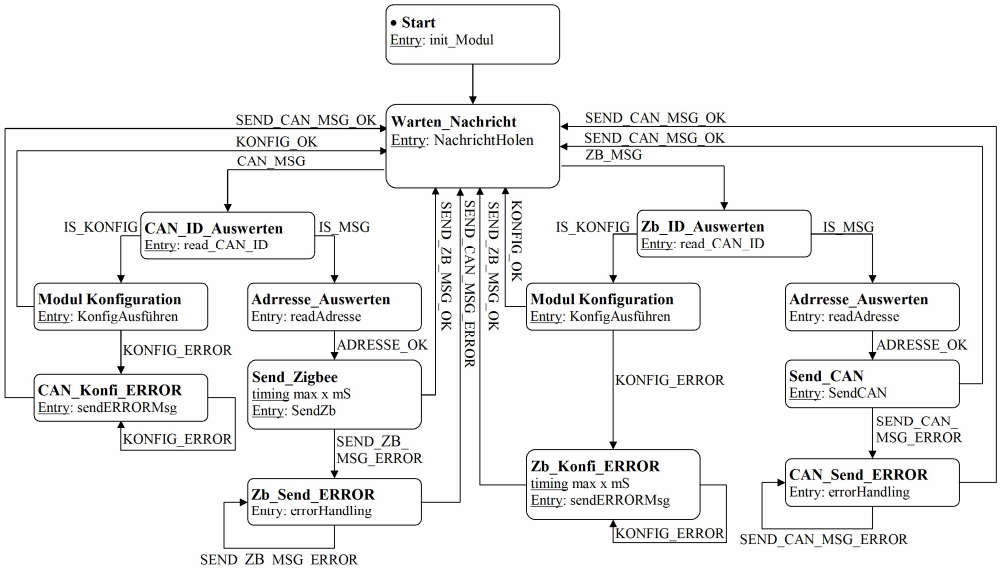


Abbildung 5-9: Harel Automat

6 Tests

6.1 Test des Layouts von CAtZi auf Funktionsfähigkeit

Um mögliche Fehler im Layout oder der Schaltung zu finden, wird bei der ersten Bestückung der Platine Schritt für Schritt vorgegangen. Dabei werden zuerst die Komponenten für die 5V-Spannungsversorgung bestückt und dann auf Funktion überprüft. Dazu werden ein Amperemeter und ein Voltmeter angeschlossen, so dass die Spannungs- und Stromwerte beobachtet werden können. Dabei weisen eine hohe Stromaufnahme oder eine abfallende Spannung auf Fehler hin. Bei der 3,3V-Spannungsversorgung wird genauso vorgegangen wie bei der 5V-Spannungsversorgung.
Nachdem sichergestellt ist, dass die richtigen Versorgungsspannungen zur Verfügung stehen, wird als nächstes die Platine mit dem Mikrocontroller ATMega128 und den dazugehörigen Komponenten bestückt. Um Beschädigungen durch Fehler zu minimieren, wird die Spannung langsam erhöht. So ist ein eventueller Fehler durch einen unerwartet hohen Strom zu erkennen, der dann sofort wieder abgeschaltet werden kann. Eine zusätzliche Möglichkeit bietet ein Labornetzgerät mit Strombegrenzung, um große Ströme gar nicht erst aufkommen zu lassen. Mit demselben Verfahren wird die Platine mit den Komponenten CAN-Controller, CAN-Treiber, BUS-Switch, CC2420EM-Modul und schließlich Taster, LEDs und Steckverbindungen bestückt.
Ist bis hier alles in Ordnung, wird mit einem einfachen Programm die Funktion des Microcontrollers getestet. Dazu wird das CC2420EM-Modul wieder entfernt und die Spannungsversorgung von der CAN-Sektion getrennt (Jumper JP105 1-2), da beide momentan noch nicht benötigt werden.
Als erster Softwaretest dient ein Programm, welches zuerst die Taster S1 bis S4 und anschließend die Dipschalter von S6 abfragt und entsprechend auf den LEDs DS1 bis DS4 ausgibt. Nachdem dieser Test erfolgreich abgeschlossen ist, kann von der allgemeinen Funktionsfähigkeit des Mikrocontrollers ausgegangen werden. Als nächstes wird die CAN-Sektion wieder mit Spannung versorgt und ein Programm gibt eine sich ständige wiederholende Nachricht auf den CAN-Bus. Um den Erfolg zu kontrollieren gibt es drei Möglichkeiten. Die erste und sicherste ist, den CAN-Bus mit einem Oszilloskop zu beobachten. Hier kann genau überwacht werden, was auf den Bus geschickt wird. Es werden keine Informationen über Busfrequenz und Abtastzeitpunkte benötigt. Die zweite Möglichkeit ist ein CAN-Analyser, der aufgrund seines Funktionsumfanges eine gewisse Zeit an Einarbeitung bedarf und die Kenntnis der Busfrequenz und des Abtastzeitpunktes voraussetzt, dafür jedoch

eine komfortable Oberfläche bietet. Die dritte und letzte Möglichkeit ist ein beliebiges Full-CAN-Gerät, das signalisieren kann, ob eine gültige Nachricht eingetroffen ist. Bei dieser Methode sind Fehlerursachen allerdings nur schwer zu identifizieren.
Nachdem nun sichergestellt ist, dass das Versenden von CAN-Nachrichten funktioniert, muss der Empfang getestet werden. Dazu wird ein Sender benötigt der Full-CAN beherrscht. Das Empfangsprogramm sollte zunächst zeigen, dass der Empfang von Nachrichten funktioniert. Ist das geschehen, kann angefangen werden verschiedene Parameter auszutesten, wie z.B. Busgeschwindigkeit, Akzeptanz-Maske und Akzeptanz-Code.
Nachdem sichergestellt ist, dass der CAN-Betrieb funktioniert, kann die CAN-Sektion wieder abgeschaltet werden. Als nächstes wird die Funkübertragung getestet, wofür das CC2420EM-Modul auf das CAtZi-Board gesteckt wird. Das Vorgehen ist hier ähnlich wie beim CAN-Bus. Es wird erst versucht eine Nachricht zu senden. Der Erfolg kann mit dem Packet-Sniffer für das CC2420DK-Board von der Firma ChipCon kontrolliert werden. Funktioniert das Senden, wird der Empfang von Funknachrichten getestet.
Funktioniert beides, kann zum Schluss ein Komplettest durchgeführt werden, in dem die volle Funktionsstrecke getestet wird. Hierzu wird einem CAtZi Board eine CAN-Nachricht übermittelt, nach deren Erhalt dieses eine Funk-Nachricht an ein zweites CAtZi-Board schickt, welches die Nachricht per CAN-Bus weiter ans Endgerät schickt. Damit ist die Funktion von CAtZi erfolgreich getestet.

6.2 Test der Übertragungsrate von CAtZi

Dem oben schon erwähnten Packet-Sniffer kann man die Übertragungszeit gesendeter Nachrichten entnehmen (Tabelle 6-1 zeigt einige exemplarische Werte). Daraus lässt sich errechen, dass bei der maximalen Transferrate von 250kbit/s Latenzzeiten von 1346 µs zwischen den Nachrichten auftreten. Die Netto-Transferrate ergibt sich aus der Brutto-Transferrate über das Verhältnis von Nutzdaten zu Overhead, also NT = BT x Nd/Oh. Das ergibt für das Senden mit IEEE 802.15.4 einen Netto-Transferraten-Bereich von 13,88 kbit/s bis 218,05 kbit/s. Dazu kommt noch der vom MAU-Protokoll verursachte Overhead. Hier ergibt die Kombination von 29 Adressbits (4 Byte) für das MAU-Protokoll mit dem Versenden eines einzelnen Daten-Bytes die ungünstigste Konstellation einer Übertragung. Aus den 5 Byte folgt dann eine Netto-Transferrate von ca. 11,4 kbit/s. Die günstigste Konstellation ergibt sich aus einer ein Byte großen Adresse mit acht Daten-Bytes, was einer maximalen Netto-Transferleistung von 76,9

Brutto Länge	Netto Länge	Sendezeit
in Byte		in µs
18	1	1932
27	10	2220
67	50	3500
117	100	5100
133	116	5612

Tabelle 6-1: Sendezeit für Nachrichten

kbit/s entspricht. Weiter kann davon ausgegangen werden, dass in den meisten Fällen zwei Byte für die Adresse ausreichend sind und die Länge der meisten Nachrichten ein, vier oder acht Byte lang sind. Damit kann von einer durchschnittlichen Übertragung von 4 Byte Daten ausgegangen werden, was einer Netto-Transferrate von 43,5 kbit/s entspricht. Die Netto-Transferrate vom CAN-Bus (CAN2.0B mit 1Mbit/s) für vier Bytes beträgt hingegen 244,3 kbit/s. Tabelle 6-2 zeigt eine Gegenüberstellung der Netto-Transferraten von CAN2.0B und Funknachrichten nach IEEE 802.15.4. Die letzte Spalte gibt das Verhältnis zwischen beiden Standards an.

Datenlänge	Netto-Transferr. in kbit/s		Verhältnis
in Byte	CAN2.0B	IEEE 802.15.4	CAN / IEEE
1	61,1	12,5	4,89
2	122,1	23,8	5,13
3	183,2	34,1	5,37
4	244,3	43,5	5,62
5	305,3	52,1	5,86
6	366,4	60,0	6,11
7	427,5	67,3	6,35
8	488,5	74,1	6,59
115		214,0	---
Für IEEE 802.15.4 sind zwei Byte für die Adresse eingerechnet			

Tabelle 6-2: Netto-Transferraten CAN und IEEE 802.15.4

Um bei starken Nachrichten-Aufkommen einen möglichst großen Datendurchsatz erzielen zu können, sollten mehre CAN-Nachrichten in einer Funk-Nachricht übertragen werden. Damit lässt sich der Overhead verringern und die Netto-Transferrate steigern. Wie stark die Steigerung ausfällt, hängt maßgeblich davon ab, wie viele Nachrichten auf einmal gesendet werden und wie diese aufgebaut sind (Adressgröße und Datenmenge). Durch die bis über sechsfache Netto-Transferleistung von CAN können während der Übertragung bis zu sechs Nachrichten eintreffen. Weiter kann es sein, dass zwischendurch andere Sender den Ether belegen und somit ein Vielfaches an Zeit zur Verfügung steht. Nun ist es unrealistisch, dass ständig Daten per CAN gesendet werden. Es ist jedoch nicht auszuschließen, dass einige Nachrichten zusammen übertragen werden können. Es ergeben sich für das Versenden von mehreren Nachrichten auf einmal Netto-Transferraten von 18,5 kbit/s bis 169 kbit/s.

Mit den hier erworbenen Kenntnissen soll noch einmal auf die in Kapitel 2.1 gemachte Abschätzung eingegangen werden. Dort wurde ein Netto-Transferrate von knapp 22 kbit/s ermittelt, Da die kleinste Netto-Transferrate unter dem geforderten Wert liegt, muss eine genauere Betrachtung vorgenommen werden. Die Anforderungen der Abschätzung machen den Einsatz einer drei Byte großen

Adressierung erforderlich. Weiter sind die Nachrichtengrößen mit vier oder acht Byte angeben. Addiert man nun alle abgeschätzten Transferraten für 4 Daten-Byte und für 8 Daten-Byte zusammen erhält man für 14,78 kbit/s und 7,17 kbit/s. Dem stehen 41,67 kibt/s und 71,43 kbit/s an Netto-Transferleistung gegenüber. Damit sind beide Werte für sich genommen kleiner der benötigten Transferleistung. Ins Verhältnis gesetzten folgt daraus dann, dass 36% der Transferrate von den 17,78 kbit/s beleget werden und weitere 10% von den 14,78 kbit/s, das ergibt eine gesamte Auslastung von 46% bei idealen Bedingungen und sollte somit ausreichen.

7 Resumé

In diesem Kapitel wird die Arbeit kurz zusammengefasst und es werden einige Möglichkeiten für zukünftige Aufgaben betrachtet. Abschließend wird ein Fazit über die Arbeit gezogen.

7.1 Zusammenfassung

Ziel der Arbeit war es, Robotern, die auf dem AKSEN-Board aufgebaut sind, eine Kommunikationsmöglichkeit über Funk zu geben. Dazu wurde untersucht, welche Möglichkeiten das AKSEN-Board bietet, um eine Funkverbindung zu ermöglichen. Es wurde entschieden, dass ein externes Funk-Interface besonders günstig über die im Aksen-Board bereits integrierte CAN-Schnittstelle angesprochen werden kann. Daraufhin wurden verschiedene in Frage kommende Funk-Technologien wie W-Lan, Bluetooth, ZigBee und NanoNet verglichen und bewertet, um daraus eine geeignete auszuwählen. Die Technologien CAN und ZigBee wurden als besonders geeignet identifiziert. Um den Robotern eine Kommunikation per Funkverbindung zu ermöglichen, wurde das CAtZi-Modul entwickelt. Dieses ermöglicht es, durch einen Mikrocontroller Nachrichten über CAN und ZigBee gemeinsam zu versenden. Dazu wurden zuerst eine Schaltung und ein geeignetes Layout entworfen, die dann in einer Double-Layer-Struktur realisiert wurden. Weiter wurde ein theoretisches Protokoll entwickelt, das externen Geräten die Konfiguration von CAtZi ermöglicht und das die Übertragung der CAN-Datenpakete auf der ZigBee-Ebene verwaltet. Zusätzlich wurden Testprogramme erstellt, um die grundlegenden Funktionen von CAtZi zu testen.

7.2 Zukunftsaussichten

Ein möglicher Entwicklungsweg für CAtZi ist der Einsatz an anderen CAN-Geräten. Hierfür ist eine Erarbeitung eines minimalistischen Layouts möglich, bei dem eine Platzersparnis von über 50% gut erreichbar sein sollte. Interessanter könnte es sein, zusätzliche Funktionalität auf dem Board zu integrieren. So könnten die Daten von direkt angeschlossenen Sensoren vorverarbeitet und dem AKSEN-Board zur Verfügung gestellt werden. Ein weiter interessanter Aspekt ist, das Board mit weiteren Kommunikationsschnittstellen, wie z.B. Bluetooth, auszustatten, um ein breites Spektrum an Kommunikationsmöglichkeiten zu bieten. Dabei sollte jedoch berücksichtigt werden, dass einzelne Sektionen abschaltbar sind, um weiterhin einen sparsamen Gebrauch zu ermöglichen. Ebenfalls interessant wäre die Untersuchung, inwieweit man Sensordaten, die per ZigBee empfangen werden, mit dem CAtZi-Modul in eine echtzeitfähige CAN-Bus Umgebung einbinden kann.

7.3 Fazit

Mit der Entwicklung des CAtZi-Modules ist eine solide Grundlage geschaffen worden, um AKSEN-Boards die Möglichkeit zu geben, über ZigBee zu funken. Zudem bietet es genug Spielraum zur Realisierung weiterer Einsatzmöglichkeiten. Der hauptsächlich vorgesehene Einsatzrahmen für das CAtZi-Modul sind Roboter, die auf dem Aksen-Board basieren. Diese stellen eine 5V-Versorgung zur Verfügung, die auf dem CatZi-Modul durch Linearregler auf ein 3.3V-Niveau zur Versorgung des ZigBee-Moduls umgesetzt werden. Durch Verwendung einer zusätzlichen externen 3.3V-Versorgung oder durch den Einsatz von verhältnismäßig teuren DC/DC-Konvertern ließe sich die Energieaufnahme des CatZi-Moduls weiter reduzieren. Weiter ist es durch eine Optimierung der Belegung des Mikrocontrollers möglich, mehr Funktionalitäten des Mikrocontrollers zur Verfügung zu stellen. Wie sich anhand der Tests gezeigt hat, ist CAtZi in der Lage, die an das Board gestellten Anforderungen gut zu erfüllen. Da im gegebenen Zeitrahmen die Software nicht im vollen Umfang realisiert werden konnte, kann keine Aussage darüber getroffen werden, ob der verwendete Mikrocontroller richtig dimensioniert ist. Das stellt jedoch nur ein kleines Problem dar, da es vom Hersteller Atmel sowohl kleinere als auch größere Ausführungen gibt, die leicht durch kleinere Anpassungen im vorhandenen Design eingesetzt werden können. Der Nachweis, ob das hier entwickelte Protokoll MAU den gestellten Anforderungen entspricht, kann leider nicht erbracht werden, da es nur zu einer theoretischen Erarbeitung gekommen ist. Jedoch ermöglicht es einen flexiblen Einsatz, der eine Anwendung auch jenseits von Roboterfußball sinnvoll erscheinen lässt.

A) Abbildungsverzeichnis

1-1	Auswahl verschiedener Roboter a: Aibo's beim Fußball spielen Quelle: http://www-2.cs.cmu.edu/~robosoccer/image-gallery/legged/2003/aibo-with-ball1.jpg Zugriffdatum: 17.07.2005 b: Lego MindStrom Konstruktion Quelle: http://www.robots.newcastle.edu.au/Images2004/DSC02002.JPG Zugriffdatum: 17.07.2005 c: FU – Fighter Quelle: http://robocup.mi.fu-berlin.de/bilderGross/start_4.jpg Zugriffdatum: 14.08.2005 d: ASIMO Quelle: http://www.motherdigital.info/archives/2005/01/vision_robot_so.html Zugriffdatum: 17.07.2005	8
1-2	Roboter an der HAW-Hamburg a: MindStorm Quelle: http://users.informatik.haw-hamburg.de/~kvl/ws02-03/IMG_0018.jpg Zugriffdatum: 17.07.2005 b: Pioneer Quelle: http://users.informatik.haw-hamburg.de/~pioneer/FH-Labor/pio2-01.jpg Zugriffdatum: 17.07.2005 c: LKW mit Aufleger Quelle: http://www.informatik.haw-hamburg.de/cms/uploads/pics/lkwGanz.GIF Zugriffdatum: 20.08.2005 d: AKSEN-Roboter Quelle: Michael Ziener	9
1-3	Schematische Darstellung einer Fußballszene Quelle: Helge Rickens	11
3-1	CAN 2.0B Data Frame Quelle: Helge Rickens	20
3-2	Mögliche ZigBee-Topologien Quelle: Helge Rickens	21
3-3	Überblick über die ZigBee Stack Architektur Quelle: http://www.elektroniknet.de/topics/kommunikation/fachthemen/2005/0012/index.htm Zugriffdatum: 06.08.2005	22
3-4	ZigBee im OSI-Model Quelle: Helge Rickens	23
3-5	Implementierungsmöglichkeiten auf dem ZigBee Protokollstapel Quelle: http://www.elektroniknet.de/topics/kommunikation/fachthemen/2005/0012/index.htm Zugriffdatum: 06.08.2005	24
3-6	APS-Frame Quelle: ZigBee Specification ,Version 1.0, 2004	24
3-7	APS-Frame-Control Quelle: ZigBee Specification ,Version 1.0, 2004	25
3-8	APS-Daten-Frame	25

	Quelle: ZigBee Specification ,Version 1.0, 2004	
3-9	APS-Comand-Frame Quelle: ZigBee Specification ,Version 1.0, 2004	26
3-10	APS-Acknowledgement-Frame Quelle: ZigBee Specification ,Version 1.0, 2004	26
3-11	NKW-Frame Quelle: ZigBee Specification ,Version 1.0, 2004	26
3-12	NKW-Frame-Control Quelle: ZigBee Specification ,Version 1.0, 2004	26
3-13	NKW-Data-Frame Quelle: ZigBee Specification ,Version 1.0, 2004	27
3-14	NWK-Command-Frame Quelle: ZigBee Specification ,Version 1.0, 2004	28
4-1	Blockschaltbild CAtZi Quelle: Helge Rickens	29
4-2	CC2420ME Quelle: Helge Rickens	31
4-3	CAtZi Platine Oberseite Quelle: Helge Rickens	33
4-4	Powerconnectoren Quelle: Helge Rickens	34
4-5	Schaltplan von CAtZi Rev. 1.0 Quelle: Helge Rickens	37
5-1	Kommunikationswege der Protokolle Quelle: Helge Rickens	38
5-2	Kommunikationswege bei Verwendung einfacher Geräte Quelle: Helge Rickens	39
5-3	Aufbau des Message-Identifier Quelle: Helge Rickens	41
5-4	Nachricht an alle Spieler Quelle: Helge Rickens	42
5-5	Nachricht an eine Gruppe Quelle: Helge Rickens	42
5-6	Nachricht an einen Spieler Quelle: Helge Rickens	43
5-7	Nachricht an alle Gruppen 1 aller Fraktionen Quelle: Helge Rickens	43
5-8	Sequenzdiagramm Quelle: Helge Rickens	48
5-9	Harel-Automat Quelle: Helge Rickens	49

B) Tabellenverzeichnis

2-1	Abschätzung der Datentransferrate Quelle: Helge Rickens	13
3-1	CAN Übertragungsraten zu Kabellänge Quelle: Helge Rickens	19
3-2	APS-Frame-Type und APS-Delivery-Mod Quelle: ZigBee Specification ,Version 1.0, 2004	25
3-3	NKW-Frame-Type Werte Quelle: ZigBee Specification ,Version 1.0, 2004	26
3-4	Discover-Route Quelle: ZigBee Specification ,Version 1.0, 2004	27
4-1	Taster, Schalter und LEDs Quelle: Helge Rickens	35
4-2	Steckverbindungen Quelle: Helge Rickens	35
5-1	Größen der Adresskategorien Quelle: Helge Rickens	41
5-2	Richtlinien SendID- & EmpfID-Vergabe Quelle: Helge Rickens	45
5-3	Richtlinien zur TypeID Quelle: Helge Rickens	45
5-4	Richtlinien für Schiedsrichter Nachrichten Quelle: Helge Rickens	46
5-5	Erläuterung der KonfID Quelle: Helge Rickens	47
6-1	Sendezeit für Nachrichten Quelle: Helge Rickens	51
6-2	Netto-Transferraten CAN und IEEE 802.15.4 Quelle: Helge Rickens	52

C) Literaturverzeichnis

AKSE04	Gunter Klemke; Compact-Flash-Modul Version: 1.0 - 10.6.2004
BOSC-1	Bosch; Controller Area Network (CAN) http://www.can.bosch.com/ (11.06.2005)
COLE03	Dietmar Cordes, Gunther Lemm; Entwicklung einer mikroprozessorbasierten Sensor- / Aktorerweiterung für das LEGO-Mindstorm System http://users.informatik.haw-hamburg.de/~kvl/cordes/diplom.pdf Hochschule für Angewandte Wissenschaften Hamburg, Studienarbeit, Oktober 2003. (17.07.2005)
D-AT-1	Datenblatt; ATMEL ATMega128(L); 8-bit AVR Microcontroller with 128K Bytes In-System Programmable Flash; Rev. 2467M–AVR–11/04;
D-CC-1	User Manual; SmartRF ® CC2420DBK Demonstration Board Kit Version: Rev. 1.3
D-CC-2	Datenblatt; ChipCon CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver; Version: Rev. 1.2, 09.06.2004
D-PH-1	Datenblatt; Philips SJA1000 Stand-alone CAN controller; Version vom 04.01.2000
D-PH-2	Datenblatt; Philips PCA82C251 CAN transceiver for 24 V systems; Version vom 13.01.2000
D-TI-1	Datenblatt; Texas Instruments SN74CB3T3245 8-BIT FET BUS SWITCH 2.5-V/3.3-V LOW-VOLTAGE WITH 5-V-TOLERANT LEVEL SHIFTER; Version: SCDS136–OCTOBER 2003–REVISED MARCH 2005
ETSC02	Konrad Etschberger; Controller-Area-Network Grundlagen Protokolle Bausteine Anwendungen; 3. Aufl.; München, Wien; Hanser, 2002
FHBR-1	Fachhochschule Brandenburg; Labor für künstliche Intelligenz; AKSEN - Controller für reaktive Roboter http://ots.fh-brandenburg.de/aksen (11.06.2005)
FIRA-1	Federation of International Robosoccer Association http://www.fira.net (17.07.2005)
HOND-1	The Honda Humanoid Robots http://www.honda-robots.com/ (17.07.2005)
IEEE03	IEEE 802.15.4:2003; Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)
KÖCK-1	Oliver Köckritz; Roboter Fussball im Robot-Lab der HAW-Hamburg http://users.informatik.haw-hamburg.de/~kvl/koeckritz/studien.pdf Hochschule für Angewandte Wissenschaften Hamburg, Studienarbeit, Mai 2005. (17.07.2005)
LEGO-1	LEGO http://www.lego.de/ (17.07.2005)

LOWR00	Wolfhard Lowrenz (Hrsg.); CAN Controller Area Network Grundlagen und Praxis; 4. Aufl.; Heidelberg; Hüthig, 2000
MANG04	Michael Manger; Design und Realisierung einer experimentellen Plattform für Roboterfußball; Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, Mai 2004. http://users.informatik.haw-hamburg.de/~kvl/manger/diplom.pdf (29.07.2005)
ROCU-1	RoboCup World Championship http://www.robocup.org (17.07.2005)
SIKO-1	Axel Sikora; Kommunikation fürs Bienenhaus http://www.elektroniknet.de/topics/kommunikation/fachthemen/2005/0012/index.htm (06.08.2005)
SONY-1	Sony Aibo http://www.eu.aibo.com/ (17.07.2005)
WIKI-1	Wikipedia - Feldbus http://de.wikipedia.org/wiki/Feldbus (18.07.2005)
ZIEN05	Michael Ziener; Konstruktion eines programmierbaren, omnidirektionalen Roboters; Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, August 2005.
ZIGB04	ZigBee Alliance ; ZigBee Specification; Version 1.0 Dezember 2004

D) Weiterführende Dokumente

Die Dokumente finden sich auf der bei liegender CD.

- CAtZi
  - Layout und Schaltplan
  - Testprogramme
- Datenblätter und Handbücher
  - ATMega128(L)
  - CC2420
  - CC2420DBK User Manual
  - CC2420DB User Manual
  - LM1085
  - PCA82C251
  - SJA100
  - SN74CB3T3245
- Internetquellen
  - Wikipedia – Feldbus
  - Elektroniknet - Kommunikation fürs Bienenhaus
- Sontiges
  - AKSEN-CompaktFlash Projekt
- Standards
  - ZigBee-Specification
  - 802.15.4-2003
  - Bosch can 2 spec
- Studien- und Diplomarbeiten
  - Die im Literaturverzeichnis aufgeführten Arbeiten (Ausnahme: Die Arbeit von Micheal Ziener, da sie noch nicht Veröffentlich ist)

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach§24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Ort, Datum	Unterschrift des Studenten