

Bachelorarbeit

Sven Maroske

Ein kognitives Verfahren zur Unterstützung von
Funkkommunikation zwischen autonomen Systemen

Sven Maroske

Ein kognitives Verfahren zur Unterstützung von
Funkkommunikation zwischen autonomen Systemen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Stephan Pareigis
Zweitgutachter : Prof. Dr. rer. nat Reinhard Baran

Abgegeben am 8. Januar 2007

Sven Maroske

Thema der Bachelorarbeit

Ein kognitives Verfahren zur Unterstützung von Funkkommunikation zwischen autonomen Systemen

Stichworte

autonome Systeme, Sensorik, kognitive Verfahren, AT90can128, Atmega32, CMUcam2, GP2D12

Kurzzusammenfassung

Es soll eine auf Funktechnik basierte Verständigung zwischen zwei Autonomen Systemen durch eine zusätzliche Bilderkennung unterstützt werden. Die Bilderkennung wird mit Hilfe von Kameras, welche sich auf den jeweiligen Systemen befinden, betrieben. Dadurch ist es beispielsweise möglich, dass Roboter miteinander ein Ausweichmanöver direkt abstimmen können.

Sven Maroske

Title of the paper

A cognitive procedure for the support of radio communication between autonomous systems

Keywords

Autonomous systems, Sensor technology, cognitive procedures, AT90can128, Atmega32, CMUcam2, GP2D12

Abstract

A communication between two autonomous systems, based on radio, is to be supported by a cognitive procedure. The cognitive procedure is operated by cameras, which are on the respective systems. So it is possible that these robots can co-ordinate an evasive maneuver together.

Inhaltsverzeichnis

Inhaltsverzeichnis	IV
Tabellenverzeichnis	VI
Bilderverzeichnis	VII
1 Einleitung	8
2 Grundlagen	10
2.1 Autonome mobile Systeme	10
2.2 Sensoren autonomer mobiler Systeme	10
3 Hardware	13
3.1 Atmega32 Controller (RN- Control).....	13
3.2 At90can128 Controller.....	16
3.3 Infrarot Sensoren (Sharp GP2D12).....	18
3.4 Funkmodul (RN- Funk).....	20
3.5 Kameramodul (CMUcam2).....	22
4 Implementierung	25
4.1 Systemarchitektur.....	25
4.2 I ² c(TWI)- Schnittstelle	27
4.3 RS232- Schnittstelle.....	30
4.4 Funkübertragung	31
4.5 Auswerten der Kamera.....	33
4.6 Motorsteuerung	34
4.7 Abstandserkennung	34
4.8 Vom Suchen und Finden	35
5 Konfiguration	37
5.1 I ² c(TWI)- Schnittstelle	37
5.2 RS232- Schnittstelle.....	39
5.3 Kameramodul	41

6 Experimentelle Ergebnisse	43
6.1 Sensorik.....	43
6.2 Programmablauf	44
6.3 Datenübertragung	48
7 Resümee.....	52
7.1 Erweiterungsmöglichkeiten.....	52
7.2 Zusammenfassung dieser Arbeit	Fehler! Textmarke nicht definiert.
Literaturverzeichnis.....	56
Anhang	57

Tabellenverzeichnis

Tabelle 3.1: Charakteristik des Infrarotsensors [10]	19
Tabelle 3.2: Auszug aus der "Command List" der Kamera [9].....	24
Tabelle 6.1: Sensordaten mit den dazugehörigen Reaktionen.....	44

Bilderverzeichnis

Bild 3.1: Atmega32 Controller Board	14
Bild 3.2: AT90can128 Controller Board	16
Bild 3.3: Arbeitsweise der Infrarotsensoren	18
Bild 3.4: Sharp Infrarotsensor GP2D12	19
Bild 3.5: Kennlinie des Infrarotsensors [11]	20
Bild 3.6: Funkmodul RN- Funk.....	21
Bild 3.7: Kameramodul CMUcam2.....	23
Bild 4.1: Die beiden autonomen mobilen Systeme	26
Bild 4.2: Zusammenhänge und Schnittstellen in einer Systemübersicht	27
Bild 4.3: Allgemeiner Aufbau einer I ² c- Schnittstelle [5]	27
Bild 4.4: Normaler Datentransfer einer I ² c(TWI)- Schnittstelle [6].....	28
Bild 4.5: TWCR Register der TWI- Schnittstelle des Atmega32 [7].....	29
Bild 4.6: TWDR Register der TWI- Schnittstelle des Atmega32 [8].....	29
Bild 4.7: “i2c_send_start“ Funktion der I ² c- Schnittstelle.....	29
Bild 4.8: “i2c_send_byte“ Funktion der I ² c- Schnittstelle.....	30
Bild 4.9: Funktion zum Senden eines Zeichens über die RS232- Schnittstelle	31
Bild 4.10: Check Funktion für das Funkmodul	33
Bild 5.1: TWBR Register der TWI- Schnittstelle des Atmega32 [7].....	37
Bild 5.2: TWAR Register der TWI- Schnittstelle des Atmega32 [8].....	38
Bild 5.3: TWCR Register der TWI- Schnittstelle des Atmega32 [7].....	38
Bild 5.4: Initialisierungsfunktion der I ² c- Schnittstelle	38
Bild 5.5: UCSR0A Register des UARTS0 vom AT90can128 [1]	39
Bild 5.6: UCSR0B Register des UARTS0 vom AT90can128 [2].....	40
Bild 5.7: UCSR0C Register des UARTS0 vom AT90can128 [3].....	40
Bild 5.8: UBRR0H/L Register des UARTS0 vom AT90can128 [4]	41
Bild 5.9: Initialisierungsfunktion der RS232- Schnittstelle.....	41
Bild 5.10: Jumperfeld der CMUcam2 zur Konfiguration der Boudrate.....	42
Bild 5.11: Initialisierungsfunktion der Kamera.....	42
Bild 6.1: Sequenzdiagramm zur normalen Abwicklung des Atmega32	45
Bild 6.2: Sequenzdiagramm zur normalen Abwicklung des AT90can128	47
Bild 6.3: Resetbefehl und Antwort des Funkmoduls.....	48
Bild 6.4: “Fundnachricht“, welche über das Funkmodul geschickt wird.....	48
Bild 6.5: “Bestätigungsnachricht“, welche über das Funkmodul empfangen wird.....	49
Bild 6.6: Diagramm einer Abfrage der Kamera mit 0 Nachricht	50
Bild 6.7: “Fundnachricht“ des Kameramoduls.....	50
Bild 6.8: Sensordaten beim Übertragen über die I ² c- Schnittstelle	51

1 Einleitung

Autonome mobile Systeme sind darauf ausgelegt ohne Hilfe von außen, Aufgaben in einem bestimmten vorgegebenen Rahmen zu erledigen.

Zum jetzigen Entwicklungsstand sind autonome Systeme noch sehr auf die ihnen gestellte Aufgabe beschränkt, wie zum Beispiel den Transport von Gegenständen in einem Lager. Solche Systeme können nicht nur in ihrer Funktion, sondern auch hinsichtlich ihrer Kapazität nur eingeschränkt eingesetzt werden. So ist es solch einem Transportsystem zum Beispiel nur möglich bis zu einem vorgegebenen Gütergewicht zu heben. Sollte es nun nötig sein größere Gewichte zu heben, gibt es natürlich die Möglichkeit größere autonome Systeme zu bauen und einzusetzen. Dies erfordert aber zum einen eine weite Voraussicht, wie sich die Einsatzgebiete solcher autonomer Systeme ändern werden, zum anderen kann so eine Weiterentwicklung sehr teuer und aufwendig sein.

Eine andere Möglichkeit ergibt sich durch die Kooperation von mehreren Transportsystemen, welche dann das für eines der Systeme zu schwere Gut gemeinsam abtransportieren. Wenn so eine Kooperation eingeleitet wird, ist es zur näheren Abstimmung erst einmal notwendig, dass die kooperierenden Systeme zueinander finden. Diese Arbeit beschäftigt sich mit diesem Problem des "Findens". Das Ziel dabei ist es, dass sich zwei in einem Raum befindliche autonome mobile Systeme finden und austauschen können.

Dabei ist es zum einen notwendig, dass die autonomen mobilen Systeme sich völlig frei im Raum bewegen können, ohne dabei überall gegen zu fahren. Zum anderen sollen sie mittels einer Kamera das gegenüberliegende autonome System finden. Unterstützt wird dieser Suchvorgang mit einem Funkmodul, welches eine Kommunikation zwischen den beiden Systemen erlaubt. Am Ende soll sich dann folgende Situation ergeben: Beide Systeme sollen den jeweiligen "Gegenüber" mittels der Kamera gefunden haben und sich jeweils vor einander befinden. Dies führt dazu, dass sich beide Systeme am Ende der Aktion gegenüber stehen.

Der Vorgang des "Findens" der beiden Systeme ist vergleichbar mit dem Kennlernen zweier Personen. Jemand betritt einen Raum und wird auf eine andere Person in diesem Raum

aufmerksam. Er geht zu dieser unbekanntem Person hin und spricht diese an. Die Person dreht sich dann voller Interesse um, um zu erkennen wer Gefallen an ihrer Person gefunden hat.

2 Grundlagen

In diesem Kapitel werde ich auf ein paar Grundlagen eingehen, welche zum Verständnis der Problemlösung beitragen sollen.

2.1 Autonome mobile Systeme

Autonome mobile Roboter sind Systeme, welche vollkommen unabhängig vom Menschen, Aufgaben erledigen und sich dabei selbständig und autark in der ihnen vorgesetzten Umgebung zurecht finden müssen. Dafür sind sie mit mehr oder weniger intelligenten Sensoren ausgestattet, welche sie bei der Umsetzung der gestellten Aufgabe unterstützen. Eine Definition könnte zum Beispiel so aussehen.

Autonome Systeme sind technologische Systeme, die selbstständig ihre Umwelt analysieren, sich anpassen, lernen, sich selbst zu organisieren, fortbewegen und auf Basis eigenständiger Schlussfolgerungen agieren und reagieren, ohne hierzu Eingriffe des Menschen zu benötigen [13]

Autonome mobile Systeme werden in den meisten Fällen eingesetzt, wenn wir Menschen an unsere natürlichen Grenzen stoßen. Sei es zum Beispiel bei der Erforschung von unbekanntem Planeten, bei der es im Augenblick nicht möglich ist mal eben hinzufiegen, oder beim Räumen von Mienenfeldern. Eingesetzt werden sie aber auch zur Erleichterung von uns lästigen oder zu aufwendigen Aufgaben, wie zum Beispiel beim Transport von Gütern oder beim Wischen des Flurs.

2.2 Sensoren autonomer mobiler Systeme

Zur Wahrnehmung ihrer Umwelt sind autonome mobile Systeme auf Sensoren angewiesen, welche ihnen die benötigten Daten liefern können. Dabei definiert sich ein Sensor wie folgt:

*Ein **Sensor** (v. lat. sensus, das Gefühl, die Empfindung) ist ein technisches Bauteil, das bestimmte physikalische oder chemische Eigenschaften (z. B.: Wärmestrahlung, Temperatur, Feuchtigkeit, Druck, Schall, Helligkeit oder Beschleunigung) und/oder die stoffliche*

Beschaffenheit seiner Umgebung qualitativ oder als Messgröße quantitativ erfassen kann. Diese Größen werden mittels physikalischer oder chemischer Effekte erfasst und in weiterverarbeitbare Größen (meist elektrische Signale) umgewandelt. [12]

So ein autonomes mobiles System ist typischerweise mit Abstandssensoren ausgerüstet. Diese Sensoren haben die Aufgabe den Abstand zu einem Hindernis festzustellen und dem System mitzuteilen, welches dann gegebenenfalls mit einem Ausweichmanöver darauf reagiert.

Zu diesen Abstandssensoren kommen dann noch für die Erfüllung seiner Aufgabe weitere benötigten Sensoren, wie zum Beispiel Kameras, Druck- oder Beschleunigungssensoren hinzu. So ist es notwendig einem Greifarm Drucksensoren einzubauen, wenn dieser empfindliche Lasten heben soll. Wenn eine Aufgabe mit optischen Mitteln erfüllt werden soll, wie zum Beispiel dem Folgen einer Bewegung, so benötigt das System Kameras, die ihre Umgebung aufnehmen.

Im Großen und Ganzen lassen sich Sensoren in zwei Hauptgruppen einteilen. Da wäre zum einen die Gruppe der aktiven Sensoren und zum anderen die Gruppe der passiven Sensoren. Dabei ist eine mögliche Definition für passive Sensoren:

Passive Sensoren bestehen nur aus passiven Elementen (Spule, Widerstand, Kondensator). Hier wird eine Änderung der elektrischen Eigenschaften, wie z. B. Widerstand, Kapazität usw. bewirkt. Passive Sensoren benötigen zur Signalerzeugung eine Hilfsstromquelle. [12]

Und eine Definition für aktive Sensoren könnte so aussehen:

Aktive Sensoren wandeln mechanische Energie, thermische Energie, Lichtenergie oder chemische Energie in elektrische Energie um. Man kann sie daher als Spannungserzeuger sehen. Der Vorgang beruht auf einem Umwandlungseffekt, wie z. B. dem Thermoeffekt, dem Fotoeffekt, dem Piezoeffekt usw. [12]

Aktive Sensoren melden in regelmäßigen Abständen ihren derzeitigen Messwert, während passive Sensoren hingegen auf eine zur Arbeit stimulierende Nachricht warten.

Bei Verwendung von Sensoren muss im Allgemeinen darauf geachtet werden, welche Störungen im Umfeld auftreten können. So ist es zum Beispiel möglich, dass eine Kamera bei veränderten Lichtverhältnissen ganz anders reagiert als bei vorhergehenden Tests. So verändert sich beispielsweise eine Farbe bei unterschiedlichem Lichteinfall. Wenn nun eine

bestimmte Farbe gesucht wird kann es passieren, dass durch veränderte Lichtverhältnisse die gesuchte Farbe aus dem angegebenen Toleranzbereich fällt. Dies führt dazu, dass sie nicht als die gesuchte Farbe erkannt wird.

Gleichermaßen muss darauf geachtet werden, dass das System blind wird falls sein Sensorsystem ausfallen sollte. Das kann dazu führen, dass es nicht mehr in der Lage ist, seine Aufgabe korrekt und zuverlässig zu erfüllen. Durch so einen Ausfall eines Sensorsystems ist es denkbar, dass erhebliche Schäden entstehen könnten. Um dieses zu verhindern, müssen Sensoren je nach Wichtigkeit überwacht und kontrolliert werden. Dies ist zum Beispiel durch zusätzliche Sensoren oder durch Annahme von möglichen Sensorwerten möglich. Die Wahl des verwendeten Überwachungssystems unterliegt dabei zumeist technischen und ökonomischen Gesichtspunkten.

3 Hardware

In dem folgenden Kapitel wird auf die in dieser Bachelorarbeit verwendete Hardware ausführlicher eingegangen. Einzelne Komponenten werden im folgenden detaillierter beschrieben.

3.1 Atmega32 Controller (RN- Control)

Bei dem für diese Bachelorarbeit verwendeten Controller Board (siehe Bild 3.1) handelt es sich um ein Board, welches speziell für Experimente und Entwicklungen im Microcontrollerbereich entwickelt wurde. Das Board ist mit einem Atmega 32 Prozessor ausgerüstet. Entworfen wurde es für die Firma “Roboternetz“, welche auch den Vertrieb über ihre Internetseite übernimmt. Die Portleitungen des Prozessors sind zur besseren Benutzung mittels Stecker nach außen geführt worden.

Der Atmega32 von “Atmel“ ist ein Prozessor welcher eine 8- Bit Risc- Architektur besitzt. Der Chip enthält außerdem einen 8- Kanal und 10- Bit Analog Digital Converter, zum Programmieren ist eine JTAG- Schnittstelle bereitgestellt. Betrieben wird der Prozessor mit einem bis zu 16 MHz Clock und 2,7 – 5,5 Volt.

Durch die mitgelieferte Programmierstelle ist es möglich ein ON- Board Debugging vorzunehmen, welches den ganzen Programmierungsprozess doch sehr vereinfacht. Im Manual des Prozessors stehen zusätzlich viele Beispiele und Anregungen wie einzelnen Schnittstellen und Funktionen eingestellt und bedient werden können.

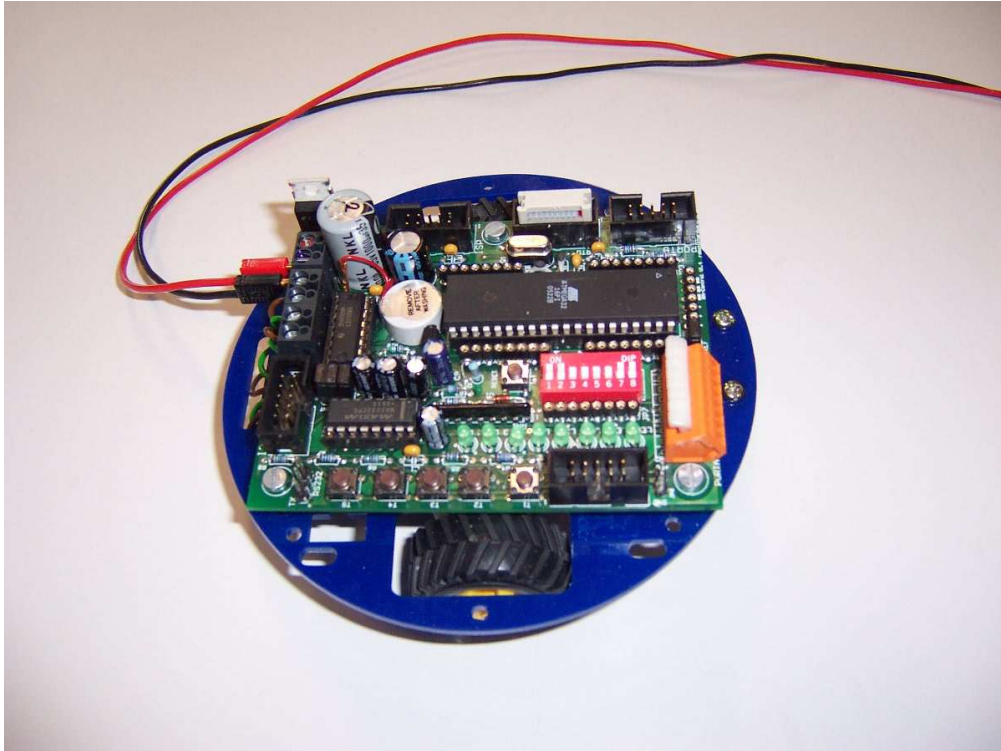


Bild 3.1: Atmega32 Controller Board

In dieser Arbeit hat das Board die Aufgabe die komplette Steuerung zu übernehmen. Dabei wertet es die Sensordaten die vom AT90can128 Controller geliefert werden aus und setzt diese auf Steuerdaten für die beiden Motoren um. Zusätzlich zu dieser Steuerung übernimmt der Controller auch die Überwachung und Durchführung des Suchvorgangs.

Merkmale und Eigenschaften des Atmega32 Controllers:

Dieses Board stellt folgende Funktionen und Module zur Verfügung:

- Eine serielle RS232- Schnittstelle
- Eine I²c Schnittstelle
- Einen Motortreiber für bis zu zwei Getriebemotoren
- Eine ISP- Programmierschnittstelle
- Einen Mini- Lautsprecher
- Einen Reset- Taster
- Fünf Taster zur beliebigen Benutzung
- Acht Leuchtdioden, welche jede für sich einzeln an- und abschaltbar sind
- Vier Ports, welche frei zur Verfügung stehen aber Überschneidungen mit oben genannten Funktionen haben können
- Einen Analog Digital Converter mit acht Eingängen
- 16- MHz Quarz

3.2 At90can128 Controller

Das At90can128 Controller Board wurde in dem Labor der HAW entwickelt. Bei diesem Board wurden alle Ports des At90can128 Prozessors, ebenfalls von "Atmel", nach außen geführt. Betrieben wird dieser Prozessor in der neuesten Ausgabe des Boards mit einem 16-MHz Clock.

Dieser Prozessor besitzt, genauso wie der Atmega32, eine 8- Bit Risc- Architektur. Eine Besonderheit gegenüber dem Atmega32 ist der integrierte CAN Bus- Controller, welcher mit 2.0A- und 2.0B-Spezifikation kompatibel ist. Ansonsten verfügt er über die gleichen Funktionen wie der Atmega32 Prozessor. Die Programmierung erfolgt hier ebenfalls über eine JTAG- Programmierschnittstelle. Genauso wie beim Atmega32 finden sich im Manual viele Beispiele und Informationen zur Handhabung.

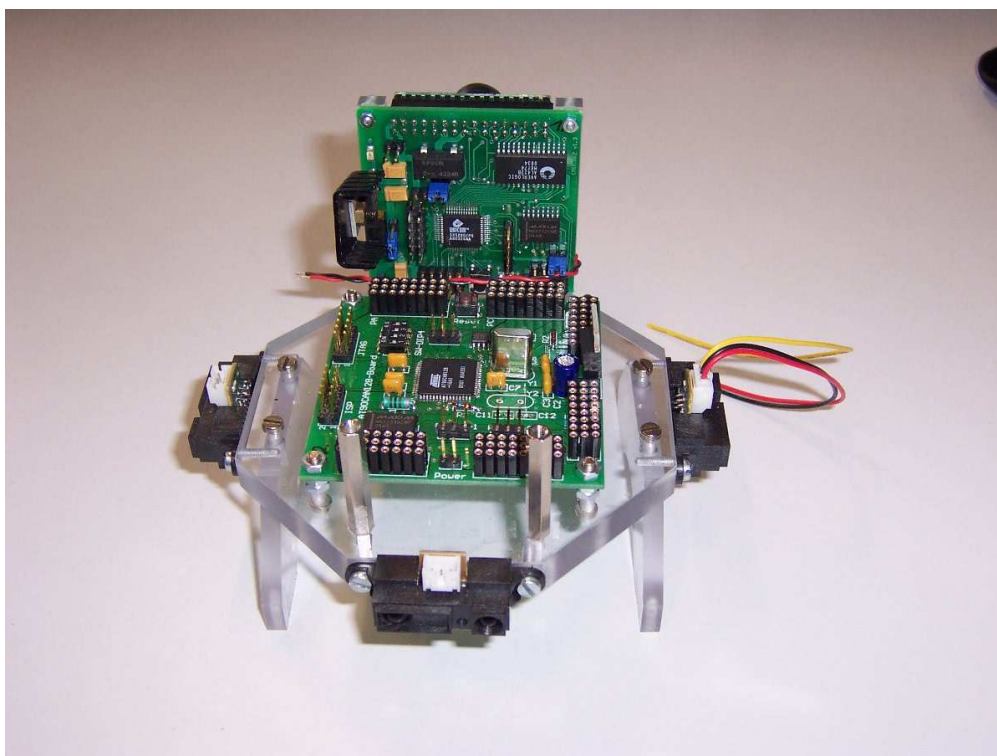


Bild 3.2: AT90can128 Controller Board

Verwendung fand das Board in dieser Arbeit als "Informationssammler". In seinen Aufgabenbereich gehört das Abfragen der Infrarotabstandssensorenmesswerte und der Kamerainformationen. Zusätzlich übernimmt es die Auswertung der über Funk eingegangenen Nachrichten.

Merkmale und Eigenschaften des AT90can128 Controllers:

Folgende Funktionen stellt dieses Board zur Verfügung:

- Zwei serielle RS232- Schnittstellen
- Eine I²c Schnittstelle
- Einen CAN - Bus
- Eine ISP Programmierschnittstelle
- Einen Reset- Taster
- Einen Analog Digital Converter mit acht Eingängen
- Sechs frei belegbare Ausgänge, welche sich allerdings mit vorhergenannten Funktionen überschneiden können
- 8- bzw. 16- MHz Quarz

3.3 Infrarot Sensoren (Sharp GP2D12)

Die Infrarotsensoren liefern ein analoges Signal, welches den Abstand zu einem Hindernis widerspiegelt. Dabei wird allerdings nur der Abstand zu einem Punkt gemessen, welcher sich ungefähr auf einer Linie mit dem Mittelpunkt zwischen Infrarotsender und Empfänger befindet. Durch diese Einschränkung ist es nicht möglich, den gesamten Bereich um den Roboter herum zu überwachen, ohne Unmengen von Sensoren einzusetzen. Um dies zu verdeutlichen habe ich hier einmal ein Bild eingefügt, welches die Funktionsweise des Sensors darstellt.

Trotz dieser Einschränkung wird der Sensor häufig verwendet. Dies ist darauf zurückzuführen, dass er eine kostengünstige Möglichkeit darstellt, Daten aus dem Umfeld aufzunehmen. Weitere Vorteile dieses Sensors sind seine einfache Handhabung, ein geringer Stromverbrauch und ein geringer Rechenaufwand. Diese Eigenschaften machen ihn gerade im Microcontrollerbereich sehr interessant.

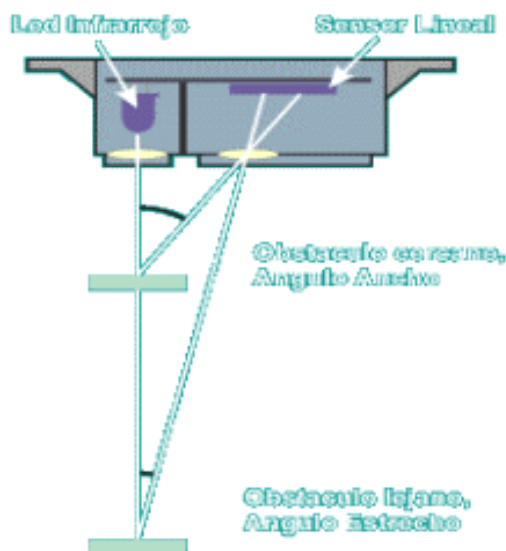


Bild 3.3: Arbeitsweise der Infrarotsensoren

Wie aus dem Bild 3.3 zu sehen ist, wird von einem Infrarotsender ein Infrarotstrahl in gerader Richtung ausgestrahlt. Trifft dieser Strahl nun auf ein Hindernis wird er in einem bestimmten Winkel zurückgeschickt und trifft auf die Empfangslinse. Hinter dieser Linse befindet sich eine Art Lineal mit Hilfe dessen sich der Winkel und damit die Entfernung bestimmen lassen.



Bild 3.4: Sharp Infrarotsensor GP2D12

In dieser Arbeit dienen diese Sensoren der Erkennung von Abständen zu eventuellen Hindernissen. Die dabei gelieferten Werte werden analysiert und entsprechend darauf reagiert.

Merkmale und Eigenschaften der Infrarotsensoren:

■ Electro-optical Characteristics

(Ta=25°C, Vcc=5V)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Distance measuring range	ΔL	^{*1} ^{*3}	10	–	80	cm
Output terminal voltage	GP2D12	V _o L=80cm ^{*1}	0.25	0.4	0.55	V
	GP2D15	V _{oH} Output voltage at High ^{*1}	V _{cc} – 0.3	–	–	–
V _{oL} Output voltage at Low ^{*1}		–	–	0.6	–	V
Difference of output voltage	GP2D12	ΔV_o Output change at L=80cm to 10cm ^{*1}	1.75	2.0	2.25	V
Distance characteristics of output	GP2D15	V _o ^{*1} ^{*2} ^{*4}	21	24	27	cm
Average Dissipation current	I _{cc}	L=80cm ^{*1}	–	33	50	mA

Note) L : Distance to reflective object.

^{*1} Using reflective object : White paper (Made by Kodak Co. Ltd. gray cards R-27 · white face, reflective ratio ; 90%).

^{*2} We ship the device after the following adjustment : Output switching distance L=24cm±3cm must be measured by the sensor.

^{*3} Distance measuring range of the optical sensor system.

^{*4} Output switching has a hysteresis width. The distance specified by V_o should be the one with which the output L switches to the output H.

Tabelle 3.1: Charakteristik des Infrarotsensors [10]

Aus der oben gezeigten Tabelle (Bild 3.1) ist ersichtlich, dass der Messbereich dieses Sensors zwischen 10cm und 80cm liegt. Die analoge Spannung liegt dabei in einem Bereich von ca. 0,4V bei einer Entfernung von 80cm und ca. 2,6V bei einer 10cm betragenden Entfernung.

In den Grenzbereichen lässt die Genauigkeit des Sensors allerdings stark nach, wodurch diese Werte doch sehr mit Vorsicht zu betrachten sind. Um solche Werte trotzdem verwenden zu können, ist es notwendig sie aufzubereiten. Dabei ist die Bildung eines arithmetischen Mittelwertes über eine bestimmte Anzahl von Messungen eine übliche Methode.

Die analoge Spannung des Sensors ändert sich nicht linear mit dem Abstand. Um dies einmal zu verdeutlichen, wird im folgenden Bild (Bild 3.5) die Kennlinie dargestellt, wie sie der Hersteller im Manual mitliefert. In diesem Bild ist auch schön zusehen, wie undefiniert die gelieferten Werte des Sensor in dem Bereich unter 10cm ist.

Fig.6 Analog Output Voltage vs.Distance to Reflective Object

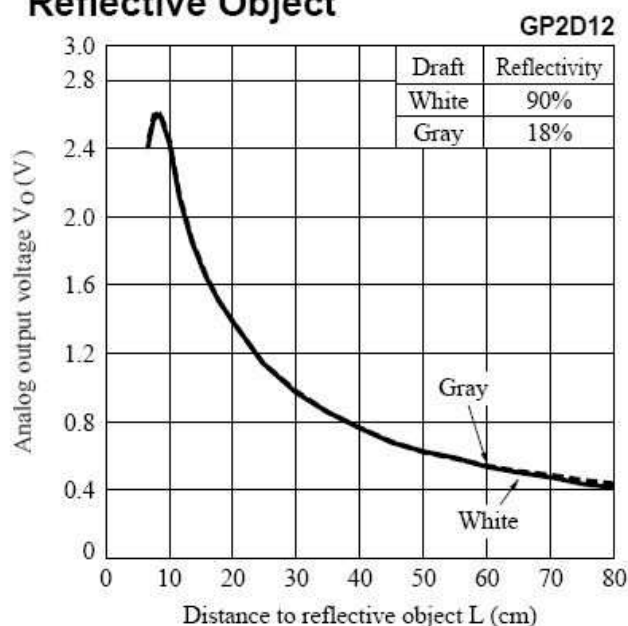


Bild 3.5: Kennlinie des Infrarotsensors [11]

3.4 Funkmodul (RN- Funk)

Dieses Funkmodul wurde von der Firma "Roboternetz" entwickelt, um schnell und komfortabel eine Funkverbindung zwischen zwei Systemen aufbauen zu können. Die kompakte Form und die Möglichkeit es mit unterschiedlichen Frequenzbändern auszurüsten ermöglichen einen vielseitigen Einsatz. Konstruiert ist das Modul wie eine Art "Blackbox". Dabei wurde darauf geachtet, dass die Verwendung sich nicht von zwei mit Kabel verbundenen seriellen Schnittstellen unterscheidet. Aufbereiten und das Verschicken der Nachricht wird von einer Firmware gehandhabt, auf die der Nutzer keinen Einfluss hat. Die Möglichkeit unterschiedliche Frequenzbänder wählen zu können ohne dabei alles ändern zu müssen hat den Vorteil, dass in unterschiedlichen Bereichen auf Störquellen flexibel reagiert werden kann.

Kleines Board um schnell und einfach ein Funkmodul an PC- und/oder Controller anzuschließen

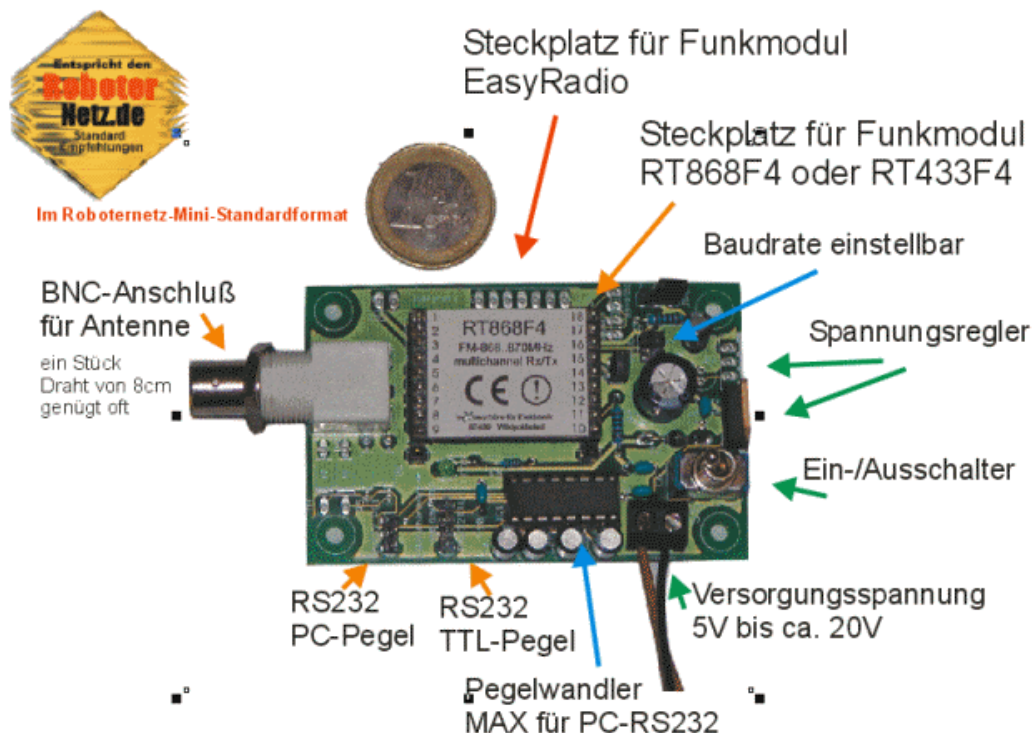


Bild 3.6: Funkmodul RN- Funk

In dieser Arbeit dient das Modul dazu Nachrichten zwischen den beiden autonomen mobilen Systemen auszutauschen. Wie zum Beispiel die Nachricht, dass der eine Roboter den anderen Roboter gesehen hat.

Merkmale und Eigenschaften des Funkmoduls:

Die Firmware des Boards übernimmt die Aufgabe, eine Nachricht die über die serielle Schnittstelle eingeht, so aufzubereiten, dass sie als Funknachricht übertragen werden kann. Der Prozess des Umwandeln und Sendens der Nachricht wurde dabei bewusst vom Nutzer ferngehalten.

- Eine RS232- Schnittstelle
- Eine RS232- Schnittstelle mit TTL
- Stromversorgung mit 5- 20V möglich

- Konfiguration der Schnittstelle durch Jumper
- Drei unterschiedliche Frequenzbänder möglich

3.5 Kameramodul (CMUcam2)

Das Kameramodul ist mit einer intelligenten Firmware ausgerüstet, welche je nach gewünschter Funktion, ihre Daten so aufbereitet, dass sie direkt verwendet werden können. Durch diese Vorstufe der Auswertung ist es relativ einfach die Kamera in bestehende Systeme zu integrieren. Die Steuerung und Konfiguration erfolgt über eine serielle Schnittstelle. Die dazu benötigten Befehle sind im Manual gut beschrieben. So bedeutet die Nachricht "TC 15 25 18 35 40 50", dass die Kamera ihr Bild nach der durch die Werte definierten Farbe absucht. Dabei werden die Grenzen von Rot, Blau und Grün in der Reihenfolge Rmin Rmax Gmin Gmax Bmin Bmax definiert.

Eine mögliche Antwort: "T 75 100 100 85 50 125" stellt ein sogenanntes "T- Paket" dar, welches auf alle "Trackbefehle" als Antwort geschickt wird. Dieses enthält drei Koordinaten, welche den Mittelpunkt der größten Farbkonzentration, die obere linke und die untere rechte Ecke angeben.

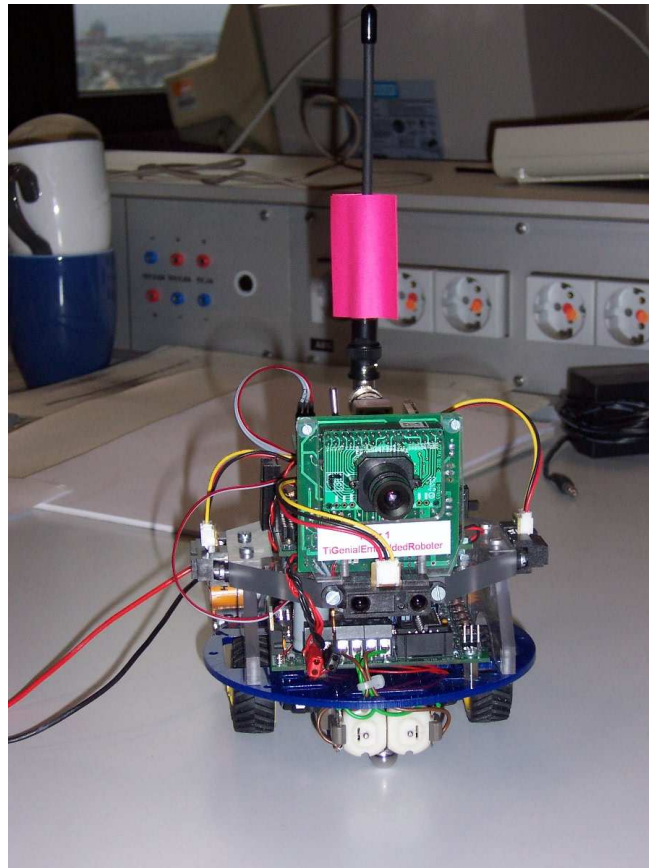


Bild 3.7: Kameramodul CMUcam2

Verwendung findet die Kamera in dieser Arbeit als optisches Mittel zum Aufspüren des gegenüberliegenden autonomen Systems. Zu diesem Zweck wurde dieses mit einer farblichen Markierung versehen, welche gut mit dieser Kamera zu erkennen ist.

Merkmale und Eigenschaften des Kameramoduls:

- Leichte Bedienung durch integrierte Firmware
- Leichte Konfiguration durch Firmware
- Fünf Servo- Ports
- Konfiguration der Schnittstelle durch Jumper
- Antwort des Moduls besteht aus vordefinierten Paketen

Den Umfang der vielen Möglichkeiten dieser Kamera zeigt die Tabelle 3.2. Sie stellt einen Auszug der “Command- List“ aus dem Manual dar.

Buffer Commands

BM	Buffer Mode	30
RF	Read Frame	47

Camera Module Commands

CR	Camera Register	31
CP	Camera Power	32
CT	Camera Type	32

Data Rate Commands

DM	Delay Mode	33
PM	Poll Mode	46
PS	Packet Skip	47
RM	Raw Mode	48
PF	Packet Filter	46
OM	Output Packet Mask	45

Servo Commands

SV	Servo Position	53
SP	Servo Parameters	52
GS	Get Servo Position	36
SM	Servo Mask	51
SO	Servo Output	51

Image Windowing Commands

SF	Send Frame	50
----	------------	----

Color Tracking Commands

TC	Track Color	53
TI	Track Inverted	53
TW	Track Window	54
NF	Noise Filter	44
LM	Line Mode	40
GT	Get Tracking Parameters	37
ST	Set Tracking Parameters	52

Histogram Commands

GH	Get Histogram	35
HC	Histogram Config	38
HT	Histogram Track	39

Frame Differencing Commands

FD	Frame Difference	34
DC	Difference Channel	32
LF	Load Frame	39
MD	Mask Difference	44
UD	Upload Difference	55
HD	HiRes Difference	38
LM	Line Mode	40

Color Statistics Commands

GM	Get Mean	36
LM	Line Mode	40

Tabelle 3.2: Auszug aus der "Command List" der Kamera [9]

4 Implementierung

4.1 Systemarchitektur

In diesem Abschnitt möchte ich einen Überblick über die einzelnen Komponenten des Systems geben. Auf die genaue Arbeitsweise einzelner Komponenten gehe ich dann noch im weiteren Verlauf dieses Kapitels ein. Die Konfiguration der einzelnen Komponenten wird in einem gesonderten Kapitel (5. Konfiguration) erläutert.

Das Gesamtsystem besteht, wie im Kapitel 3 bereits erwähnt, aus einem Atmega32 Controller Board (RN- Control) mit zwei Motoren, einem At90can128 Board, einer Kamera (CMUCam2), einem Funkmodul (RN-Funk) sowie einigen Infrarotsensoren von Sharp(GP2D12).

Die einzelnen Module werden im Gesamtsystem für unterschiedliche Aufgaben und Tätigkeiten eingesetzt.

Der Atmega32 Controller übernimmt die Aufgabe der Steuerung des Roboters. In seinem Bereich fällt die Steuerung der beiden Motoren, die Überwachung des Suchablaufs und die Auswertung der Sensordaten, sowie die Ausführung der daraus folgenden Reaktion

Die Hauptaufgabe des AT90can128 Board ist die Informationssammlung. Dazu gehören die Auswertung der über Funk empfangenen Nachrichten und die Kommunikation mit der Kamera, genauso wie das Überwachen der Abstände zu den Seiten, um Kollisionen zu vermeiden.

Um seine Umgebung beobachten und Hindernisse erkennen zu können, verfügt das Board über vier Infrarotsensoren, durch welche die Abdeckung der vier Seiten des Roboters realisiert werden kann. Mit Hilfe dieser Sensoren hat der Roboter dann die Möglichkeit, sich nach hinten und vorne sowie nach links und rechts zu orientieren. Die Kommunikation zwischen diesen beiden Boards erfolgt über eine I²c- Schnittstelle.

Die Kamera wird in dieser Arbeit eingesetzt, ihre Umgebung nach einem bestimmten Farbmuster zu untersuchen. Sie gibt jeweils eine Meldung darüber ab, ob das Muster gefunden wurde oder nicht.

Das Funkmodul dient zur Kommunikation mit dem gesuchten System. Das Modul bildet dabei eine Grundlage zur gegenseitigen Abstimmung der beiden Roboter.

Wie oben erwähnt, unterstützen die Infrarotsensoren aus dem Hause Sharp den AT90can128 Controller dabei mögliche Kollisionen mit Hindernissen zu vermeiden. Diese Unterstützung erfolgt über eine an den Controller gesendete analoge Spannung, welche den Abstand zu einem Hindernis widerspiegelt.

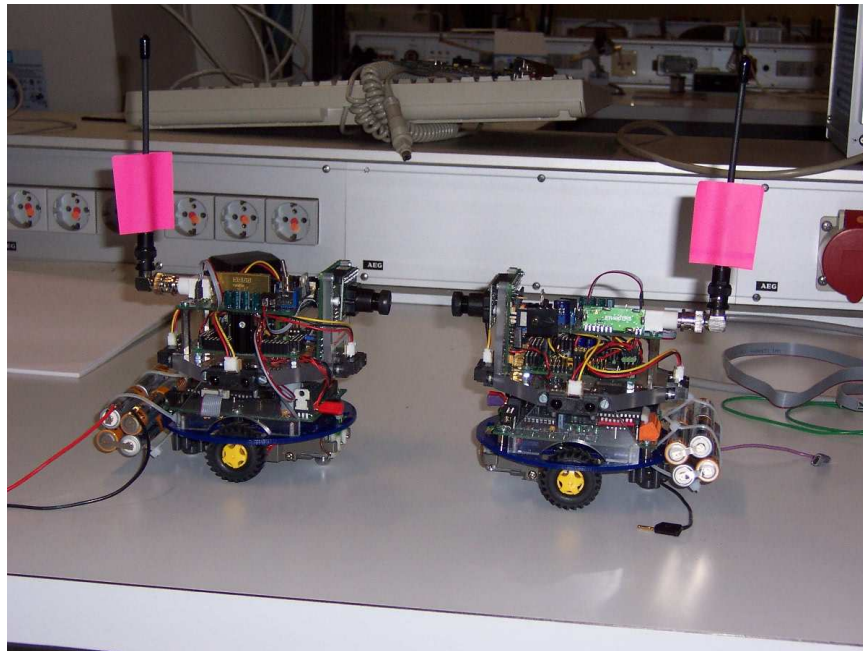


Bild 4.1: Die beiden autonomen mobilen Systeme

Das folgende Bild 4.2 verdeutlicht den Systemaufbau. Es soll zeigen wie im einzelnen die Module und Controller zusammenhängen und über welche Schnittstellen sie miteinander kommunizieren.

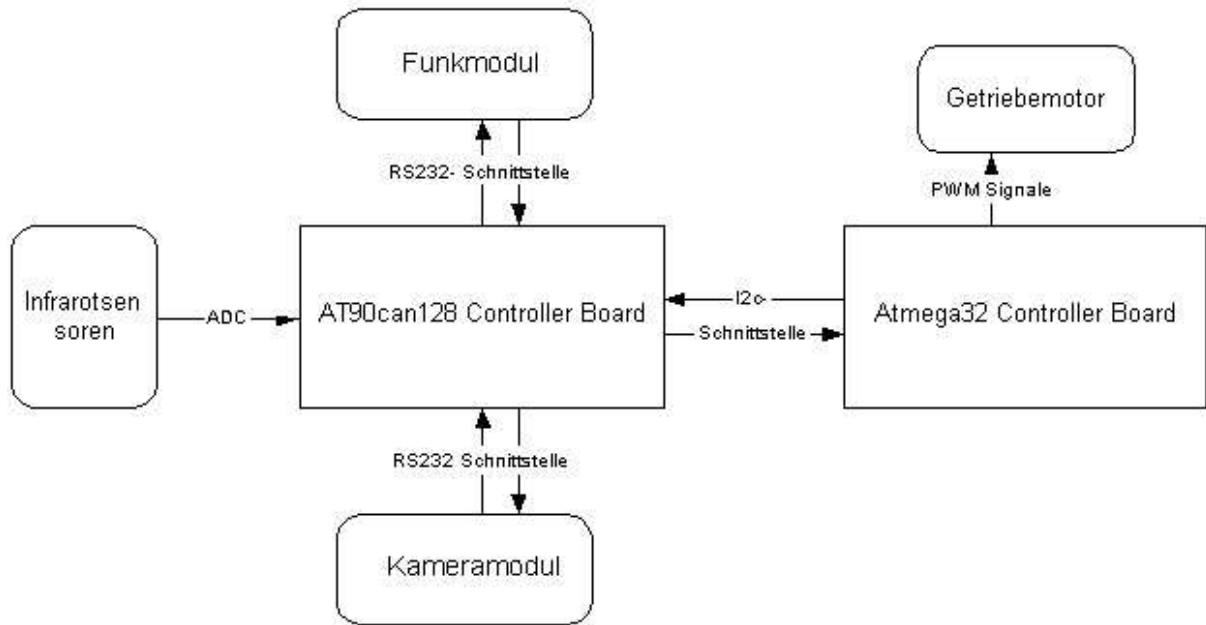
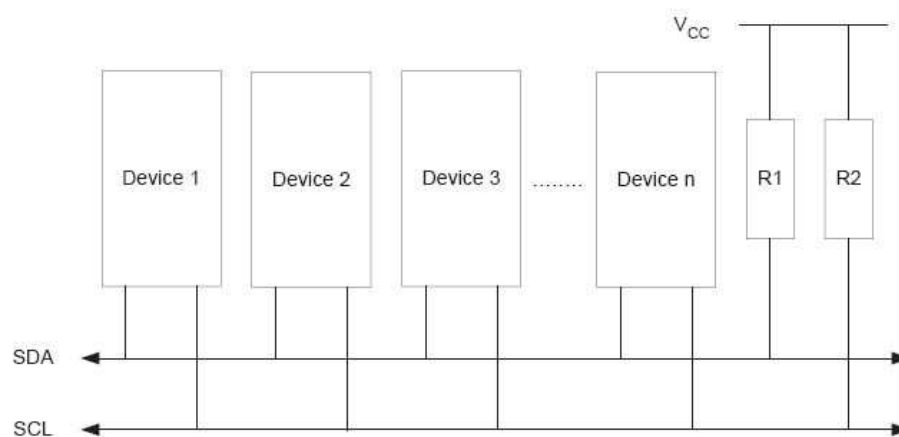


Bild 4.2: Zusammenhänge und Schnittstellen in einer Systemübersicht

4.2 I²c(TWI)- Schnittstelle

Die I²c- Schnittstelle dient als Kommunikationsmedium zwischen den beiden Boards. Das Bild 4.3 zeigt den allgemeinen Aufbau einer I²c(TWI)- Schnittstelle.

Bild 4.3: Allgemeiner Aufbau einer I²c- Schnittstelle [5]

Betrieben wird die Schnittstelle im Multimasterbetrieb. Dies bedeutet das beide Boards unabhängig voneinander den Sendevorgang starten können. Notwendig ist dies, damit jedes Board dem anderen Board Nachrichten übermitteln kann, wenn sich bestimmte Ereignisse

einstellen. Zum Beispiel fordert der Atmega32 die Sensordaten an, während der AT90can128 dem Atmega32 mitteilt, dass er eine Funknachricht empfangen hat. Durch den Multimasterbetrieb ist es notwendig den Takt bei beiden Boards einzustellen. Anderenfalls könnte es passieren das kein gemeinsamer Takt gefunden werden kann. Sollte diese Einstellung nicht erfolgen ist eine Kommunikation über diese Schnittstelle nicht möglich, da Nachrichten vom Empfänger falsch interpretiert werden.

Ein Sendevorgang bei der I²c(Twi)- Schnittstelle besteht aus einer Startsequenz gefolgt von einer sieben Bit langen Adresse des Empfängers und dem Read/ Write Bit. Danach folgen die Nutzdaten. Sind die Nutzdaten alle übertragen, wird vom Sender eine Stopsequenz gesendet, welche den Vorgang beendet. Eine normale Übertragung eines Datenstroms sieht dann wie im Bild 4.4 gezeigt aus.

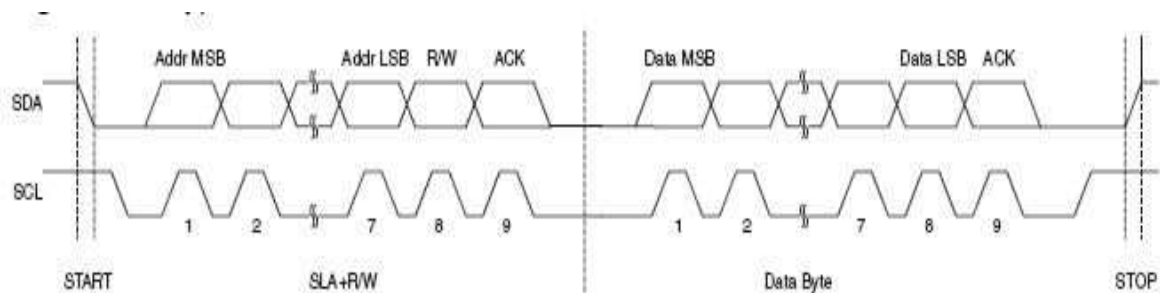


Bild 4.4: Normaler Datentransfer einer I²c(TWI)- Schnittstelle [6]

Vor dem Senden der Daten über die Schnittstelle sollte überprüft werden, ob diese frei ist. Falls die Schnittstelle belegt sein sollte und trotzdem gesendet wird könnten Probleme durch Verfälschung der Sendedaten auftreten, da in einen bereits laufenden Sendeprozess geschrieben wird.

Die einzelnen für diese Bachelorarbeit benötigten Funktionen, wie zum Beispiel “i2c_send_Start()“, “i2c_send_Stop()“ oder “i2c_send_byte()“, benutzen vom Atmel Prozessor bereitgestellte Funktionen der internen TWI- Schnittstelle. So bietet die Schnittstelle zum Beispiel die Funktion “TWSTA“ im “Control Register“ “TWCR“, welche bei Aktivierung eine Startsequenz schickt oder eine Funktion “TWSTO“, welche das Senden einer Stopsequenz ausführt.

Diese Übertragungsfunktionen sind bereits in mehrfacher Ausführung verbreitet und können aus vielen Quellen, wie zum Beispiel das Internet oder dem Manual des Prozessors bezogen werden. Dabei muss allerdings beachtet werden, dass es Unterschiede zwischen

verschiedenen Atmels geben kann und somit eventuell einige Register angepasst werden müssen. Die dazu benötigten Daten können im zum Atmel gehörigen Manual nachgelesen werden. Weiterhin stehen im Manual einige Beispiele zur Nutzung der Schnittstellen.

So dient das “TWCR“ (TWI Control Register) zur Steuerung der Schnittstelle. Dazu verfügt es über Funktionen, welche die Schnittstelle aktiviert und deaktiviert oder vordefinierte Übertragungsvorgänge, wie zum Beispiel die Start- oder Stoppssequenz, einleiten. Zusätzlich enthält dieses Register allerdings auch benötigte Funktionen zum Interrupthandling.

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bild 4.5: TWCR Register der TWI- Schnittstelle des Atmega32 [7]

Daten die über diese Schnittstelle verschickt werden sollen oder als letztes empfangen wurden, werden in das “TWDR“ (TWI Data Register) geschrieben. Dies geschieht entweder vom Nutzer beim Senden oder von der Schnittstelle wenn etwas empfangen wurde.

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

Bild 4.6: TWDR Register der TWI- Schnittstelle des Atmega32 [8]

Wie eine Funktion der I²c- Schnittstelle aussieht möchte ich an zwei Beispielen verdeutlichen.

```

unsigned char i2c_send_start (void)
{
    /*writing a one to TWINT clears it, TWSTA=start, TWEN=TWI-enable*/
    TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
    /*wait, until start condition has been sent --> ACK*/
    while (!(TWCR & (1<<TWINT)));
    return TWSR;
}

```

Bild 4.7:“i2c_send _start“ Funktion der I²c- Schnittstelle

Die Funktion im Bild 4.7 stellt die Funktion in der die Startsequenz erzeugt wird dar. Zum Erzeugen der Sequenz wird nur das “Control Register“ benötigt. In diesem wird das diesen Vorgang einleitende Bit (TWSTA) auf 1 gesetzt. Dazu muss die Schnittstelle aktiviert

(TWEN) werden. “TWINT“ muss mit einer 1 gelöscht werden. Dann wird gewartet bis die Schnittstelle das “TWINT“ Bit wieder auf 1 zurück gesetzt hat. Dieses signalisiert, dass der Vorgang abgeschlossen wurde. Zur Kontrolle wird am Ende noch das Statusregister zurückgegeben.

```

unsigned char i2c_send_byte (unsigned char byte)
{
    /*TWDR contains byte to send*/
    TWDR = byte;
    /*send content of TWDR*/
    TWCR = (1<<TWINT) | (1<<TWEN);
    /*wait, until byte has been sent --> ACK*/
    while (!(TWCR & (1<<TWINT)));
    return TWSR;
}

```

Bild 4.8: “i2c_send_byte“ Funktion der I²C- Schnittstelle

Die im Bild 4.8 gezeigte Funktion dient dazu ein Byte Nutzdaten zu verschicken. Das zu sendende Byte wird ins Datenregister (TWDR) der Schnittstelle geschrieben. Nach der Aktivierung der Schnittstelle wird das Byte dann übertragen und auf das “TWINT“ Bit gewartet.

4.3 RS232- Schnittstelle

Bei dieser Arbeit werden zwei serielle RS232- Schnittstellen zur Kommunikation des AT90can128 Board mit der Kamera und dem Funkmodul eingesetzt. Diese beiden Schnittstellen sind auf dem Board bereits vorhanden. Sie werden über sogenannte UARTs¹ konfiguriert und gesteuert.

Bei einem Sendevorgang wird erst einmal gewartet und überprüft, ob die Schnittstelle frei ist. Sobald dies der Fall ist wird das zu sendende Byte ins Datenregister des zu benutzenden UART geschrieben. Der UART übernimmt dann die Umsetzung der Daten auf den Leitungscodex.

Die Vorgehensweise in den Übertragungsfunktionen der RS232- Schnittstelle sind ebenfalls, wie die der I²C- Schnittstelle, sehr verbreitet und können aus mehreren Quellen bezogen

¹ (Universal (Synchronous) Asynchronous Receiver Transmitter)

werden. Einige Beispiele können auch dem Manual des “Atmel“ entnommen werden. Bild 4.9 zeigt ein Beispiel wie so eine Funktion aussehen kann.

```

/* Zeichen senden 0 */
uint8_t Usart_Tx0(unsigned char c)
{
    while ( !(UCSROA & _BV(UDRE0)) ); /* warten bis senden moeglich */
    UDR0 = c; /* Schreibt das Zeichen x auf die Schnittstelle */

    return 0;
}

```

Bild 4.9: Funktion zum Senden eines Zeichens über die RS232- Schnittstelle

Die Funktion “usart_tx0(unsigned char c)“ sendet ein Zeichen über die RS232- Schnittstelle. Dazu überprüft die Funktion erst einmal ob die Schnittstelle frei ist. Ist diese frei wird das zusendende Zeichen ins Datenregister (UDR0) der Schnittstelle geschrieben und damit verschickt.

4.4 Funkübertragung

Sobald das gegenüberliegende autonome System gefunden wurde wird an dieses mitgeteilt. Diese Funkübertragung erfolgt indem ein Byte über die eine RS232- Schnittstelle an das Funkmodul geschickt wird. Wenn der Sendebuffer des Moduls voll ist oder eine bestimmte Zeit, die etwa einer doppelten Sendezeit einer Nachricht entspricht, kein neues Byte dazu gekommen ist, werden die bisher zu sendenden Daten verschickt.

Falls ein Byte empfangen wurde, wird dieses über die RS232- Schnittstelle an das Board weitergeleitet. Das Board ist dann dafür verantwortlich einen Interrupt auszulösen. Die Interrupt Service Routine muss das Byte auslesen und es für die spätere Verarbeitung in einen Ringbuffer schreiben. Dieser wird regelmäßig vom At90can128 Board überprüft und ausgewertet. Dabei durchläuft der Controller den Empfangsbuffer und sucht nach dem Anfang einer Nachricht. Wenn dieser Anfang gefunden wurde, werden die folgenden Zeichen als Nachricht interpretiert und ausgelesen. Dies läuft solange bis ein Zeichen kommt, welches das Ende der Nachricht bedeutet. Ist nun die empfangene Nachricht verwertbar, wird entsprechend auf diese reagiert.

Das Versenden kann nicht beeinflusst werden. Das Funkmodul kann dabei als eine Art Blackbox mit einem Ein- und Ausgang betrachtet werden. Diese Eigenart soll dafür sorgen,

dass sich die beiden RS232- Schnittstellen so verhalten als seien sie mit einem einfachen seriellen Kabel verbunden.

Zur Überwachung und Überprüfung der empfangenen Funkdaten beginnen diese immer mit einem "a", welches den Anfang der Datenübertragung signalisiert. Die Daten enden immer auf einem "e", wenn alle Daten empfangen wurden.

Dazwischen befindliche Zeichen signalisieren dann die eigentlichen Nachrichten. Für diese Arbeit sind nur Zwei Nachrichten notwendig. Zum einen wird ein "f" verschickt, wenn das andere autonome System gesehen wurde. Als Antwort auf diese Nachricht wird dann ein "c" geschickt. Diese Nachricht sagt aus, dass der "Fund" bestätigt wird und das autonome System entsprechend handelt. Das System handelt indem es seinerseits versucht den Sender der "Fundnachricht" im näheren Umkreis zu finden. Sobald das autonome System gefunden wurde wird die Nachricht solange wiederholt gesendet bis sie bestätigt wird.

Die folgende Funktion (Bild 5.10) durchsucht den Empfangsbuffer für das Funkmodul auf Nachrichten. Wenn Nachrichten eingetroffen sind, dann durchsucht sie den Empfangsbuffer nach dem Anfang der Nachricht ("a"). Danach wird nach der eigentlichen Nachricht ("f","c") gesucht. Die Suche nach dem Ende ("e") schließt das ganze ab. Wird beim Durchsuchen eine "Fundnachricht" gefunden, wird dieses dem Atmega32 Controller mitgeteilt und eine "Bestätigungsnachricht" zurückgeschickt. Wird eine "Bestätigungsnachricht" gefunden wird dieses für später gespeichert. Am Ende wird noch der Buffer geleert wenn er einen bestimmte Füllstatus erreicht hat.

```

void check_Radio(void)
{
    unsigned char radio_state=0x00;
    if(radio_buffercount>radio_readcount) //falls daten empfangen
    {
        while(radio_readcount<radio_buffercount && radio_state!=0x03) //solange noch daten vorhanden und kein endsignal
        {
            if(radio_buffer[radio_readcount]=='a' && radio_state==0x00) //wenn startpunkt gefunden
            {
                radio_state=0x01;
            }
            else if(radio_buffer[radio_readcount]=='f' && radio_state==0x01) //wenn gegenüber mich gefunden hat
            {
                i2c_set_outputs(mega32_empf, 0x04);
                radio_state=0x02;
            }
            else if(radio_buffer[radio_readcount]=='e' && radio_state==0x02) //wenn endpoint gefunden
            {
                radio_state=0x03;
            }
            else if(radio_buffer[radio_readcount]=='k' && radio_state==0x01) //wenn eine bestätigung gefunden
            {
                radio_state=0x04;
            }
            else if(radio_buffer[radio_readcount]=='e' && radio_state==0x04) //wenn endpoint nach einer bestätigung gefunden
            {
                radio_state=0x05;
            }
            radio_readcount++;
        }
    }
    if(radio_state==0x03)
    {
        send_Radio('k');
    }
    else if(radio_state==0x05)
    {
        findack=0x01;
    }
    //wenn alle daten ausgelesen, und eine bestimmte anzahl an daten empfangen wurden, counts resetten
    if(radio_readcount==radio_buffercount && radio_buffercount>15)
    {
        radio_buffercount=0x00;
        radio_readcount=0x00;
    }
}

```

Bild 4.10: Check Funktion für das Funkmodul

4.5 Auswerten der Kamera

Die Bilderkennung bzw. Farberkennung wird gestartet indem der Kamera ein “Track Color“ Befehl geschickt wird. Dieser Befehl enthält die Toleranzbereiche der gesuchten Farbe in ihren Rot-, Grün- und Blautönen. Die Kamera antwortet auf diese Nachricht, indem sie ein so genanntes “T- Paket“ zurückschickt. Dieses “T- Paket“ startet immer mit einem großen “T“ gefolgt von einer X- und Y- Koordinate, welche dem Mittelwert des Bereichs der Farbe entspricht. Darauf folgen die Werte der linken oberen und der rechten unteren Ecke, die jeweils wieder aus einer X- und Y- Koordinate bestehen.

Beispiel: T 125 80 150 70 100 90

Die Antwort der Kamera, wird genauso wie bei den zuvor beschriebenen Funknachrichten, erst einmal mittels eines Interrupts in einen Ringbuffer geschrieben. An dieser Stelle wartet sie auf ihre Auswertung, welche in regelmäßigen Abständen erfolgt.

Für die Implementierung dieser Bachelorarbeit reicht es nur die X- und Y-Koordinate des Mittelwerts zu betrachten. Da es nur wichtig ist zu wissen ob sich das gesuchte autonome System in einem bestimmten Winkel vor einem befindet. Die genaue Position in diesem Bereich muss nicht bekannt sein.

Sollten die ermittelten Werte nun in einem Bereich vor dem Roboter liegen, gilt es als Fund eines anderen autonomen mobilen Systems und die Fahrt wird unterbrochen.

Die Funktion `“wait_for_Cam_ACK_TPak()“` entspricht dem Aufbau der im Bild 4.10 gezeigte Funktion. Sie prüft dabei auf die Nachricht `“ACK“` oder das Antwortpaket der Kamera. Dies erfolgt immer nachdem ein Befehl zur Kamera gesendet wurden.

4.6 Motorsteuerung

Die Motorsteuerung erfolgt über zwei PWM, welche vom Atmega32 Board bereitgestellt werden. Diesen PWM kann der Takt vorgegeben werden. Durch diesen vorgegebenen Takt kann eine Fahrt in unterschiedlichen Geschwindigkeiten realisiert werden. Durch die getrennte Steuerung der beiden Motoren, ist es möglich Drehungen auf der Stelle durchzuführen, wodurch der Roboter sehr beweglich wird.

Die Steuerung des Roboters ist auf ein Minimum der eigentlich möglichen Funktionen beschränkt. Funktionen wie zum Beispiel langgezogene Kurven wurden wegen der Komplexität und Unbedeutsamkeit für diese Arbeit weggelassen. So ist es nur möglich rückwärts und vorwärts zu fahren, sowie rechts und links zu drehen. Die dabei gefahrene Geschwindigkeit wurde voreingestellt, damit sich einheitliche Ergebnisse erzielen lassen und diese nicht durch unterschiedliche Geschwindigkeiten verfälscht werden.

4.7 Abstandserkennung

Die Abstandserkennung erfolgt mit Hilfe von vier Infrarotsensoren vom Typ GPD12 der Firma Sharp. Diese Sensoren liefern eine analoge Spannung von 0,5V – 2,55V. Aus diesem Grund sind die Sensoren an den ADC des At90can128 angeschlossen. Dieser wandelt die gelieferten Spannungen in einen digitalen 10- Bit Wert um. Durch die Einstellung `“left-aligned“` werden die oberen acht Bit des Ergebnisses ins `“Highbyte“` des Datenregisters geschrieben. Dadurch reicht es aus, nur diese acht Bit zu betrachten, da die unteren beiden

Bits das Ergebnis nur im Millimeterbereich verändern würden. Sollte nun aber eine Genauigkeit in diesem Messbereich gefordert sein, kann es wiederum notwendig werden diese beiden Bits mit in die Berechnungen einzubeziehen.

Um nun grobe Messfehler auszuschließen wird ein arithmetischer Mittelwert über drei Wandlungen des ADC gebildet.

Die Auswertung der Sensoren erfolgt direkt nach der Umwandlung. Je nach Ergebnis wird eine 1 oder 0 zurückgegeben. Wobei eine 1 bedeutet, dass sich ein Hindernis im kritischen Bereich befindet. Aus den vier zurückgegeben 1en oder 0en wird dann ein Byte zusammengesetzt, wobei die unteren vier Bits die Werte der Sensoren darstellen. Aufgeführt werden die Sensoren in der Reihenfolge Links, Vorne, Rechts und Hinten. Dieses Byte wird dann zur Abrufung bereitgestellt.

Wenn der Roboter sich nun im Fahrbetrieb befindet, werden die Sensordaten vom Atmega32 Controller angefordert. Dieser liest das Byte aus, und entsprechend mit einem Ausweichmanöver. So bedeuten die Sensordaten "0 0 0 0 1 1 0 0", dass vor und links von dem Roboter ein Hindernis liegt. In der Folge würde der Roboter solange nach rechts drehen bis sich vor ihm kein Hindernis mehr befindet. Daraus ergeben sich die Sensordaten "0 0 0 0 1 0 0 1", denn jetzt befindet sich die Hindernisse links und hinter ihm und der Weg nach vorne ist frei.

4.8 Vom Suchen und Finden

Sobald das At90can128 Board aktiviert wird fängt es an seine Umgebung zu beobachten. Dies geschieht zum einem mit den Infrarotabstandssensoren und zum anderem mittels der Kamera. Zusätzlich "lauscht" das Board auf Funknachrichten des anderen autonomen Systems.

Der Atmega32 Controller wartet nach seiner Aktivierung auf Anweisungen zu warten. In dem Fall, dass der Befehl zum Losfahren über einen Taster gegeben wurde, fängt der Atmega32 an vom At90can128 die Sensordaten anzufordern und auf diese mit entsprechenden Ausweichmanövern zu reagieren.

Sollte der Fall eintreten, dass das gesuchte Objekt von der Kamera entdeckt wurde, sendet der At90can128 eine "Fundnachricht" über die I²c- Schnittstelle an den Atmega32 und über Funk an das andere gefundene autonome System. Diese Nachricht wird dann solange wiederholt

gesendet bis eine Bestätigung des anderen Systems eingeht oder dieses wieder aus dem Sichtbereich verschwunden ist.

Auf die "Fundnachricht" des At90can128 reagiert der Atmega32 indem er erst einmal stehen bleibt und die Abfrage der Sensordaten einstellt. Diese müssen nicht mehr abgefragt werden, weil sonst ein Ausweichmanöver eingeleitet werden könnte, wenn sich das andere autonome System zu dicht vor dem Roboter befindet.

Die Funknachricht, dass das gesuchte autonome System gefunden wurde, bewirkt zunächst einen Stopp. Danach wird sich so lange nach rechts gedreht bis das andere System mit der Kamera entdeckt wird. Wenn dies der Fall ist wird eine Nachricht an das eigene Atmega32 Board geschickt und über Funk dem anderen autonomen System Bescheid gegeben. Wenn sich nun beide Systeme entdeckt haben und auch eine Bestätigung darüber erfolgt ist, schalten sich diese ab.

5 Konfiguration

In diesem Kapitel möchte ich einen kurzen Überblick über die Konfigurationen einzelner Systeme geben, welche in dieser Bachelorarbeit eingesetzt wurden.

5.1 I²c(TWI)- Schnittstelle

Die I²c (TWI)- Schnittstelle wird bei der Initialisierung auf einen Übertragungstakt von 100KHz eingestellt. Der dafür benötigte Teiler wird dabei mit Hilfe der folgenden Formel berechnet.

$$\text{Teiler} = ((\text{cpu-freq}/\text{scl-freq})-16)/2$$

cpu-freq = Geschwindigkeit des Prozessors, scl-freq = gewünschter Übertragungstakt

Weiterhin wird die Schnittstelle, wie oben beschrieben, im Multimasterbetrieb betrieben. Dies führt dazu, dass beide Boards auf den gleichen Takt eingestellt werden müssen. Zusätzlich ist es notwendig beiden eine "Slave- Adresse" zu geben, weil sonst ein Board für das andere nicht erreichbar ist. Ansonsten werden beide Schnittstellencontroller wie im "Master/Slave-Betrieb" konfiguriert.

Die Konfiguration der Schnittstelle erfolgt über Register. Die aufgeführten Register sind dem Manual des Atmega32 entnommen und können sich bei anderen Prozessoren unterscheiden.

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bild 5.1: TWBR Register der TWI- Schnittstelle des Atmega32 [7]

Im "TWBR" (TWI Bit Rate Register) wird der Teiler zur Erreichung des gewünschten Taktes eingetragen, welcher aus der oberen Formel berechnet wurde.

Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

Bild 5.2: TWAR Register der TWI- Schnittstelle des Atmega32 [8]

Das "TWAR" (TWI Address Register) dient als Speicherort für die "Slave- Adresse" über welche diese Schnittstelle im "I²c- Ring" zu finden ist. Zusätzlich kann in diesem Register am Bit "TWGCE" eingestellt werden, ob auf einen allgemeinen Ruf reagiert werden soll oder nicht.

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bild 5.3: TWCR Register der TWI- Schnittstelle des Atmega32 [7]

Dieses "TWCR" (TWI Control Register) beinhaltet Funktionen welche eine Start- (TWSTA) oder eine Stopsequenz (TWSTO) einleiten. In diesem Register kann die Schnittstelle (TWEN) und die Interrupts (TWIE) aktiviert werden. Zusätzlich zeigt das Register an, wenn ein Vorgang abgeschlossen wurde (TWINT) oder die Schnittstelle gerade belegt ist (TWWC). Des weiteren wird hier festgelegt, ob eine Acknowledge Meldung (TWEA) gesendet wird, wenn bestimmte Ereignisse eintreten. Solche Ereignisse sind das Empfangen der eigenen Adresse, das Empfangen eines allgemeinen Rufes oder das Empfangen eines Datenpakets.

In dieser Arbeit wurde diese Schnittstelle auf beiden Seiten wie folgt konfiguriert.

```
void i2c_init (void)
{
    TWBR = 72; //100kHz bei 16Mhz Prozessor ==((cpu-freq/sc1-freq)-16)/2
    TWAR = 0xC0; //
    TWCR = (1<<TWEN)|(1<<TWEA)|(1<<TWIE);
}
```

Bild 5.4: Initialisierungsfunktion der I²c- Schnittstelle

Wie aus dem Bild erkenntlich, wurde die Adresse (TWAR), bei dem hier gezeigten AT90can128 auf 0xC0 gesetzt, die Adresse des Atmega32 wurde auf 0xD0 gesetzt. Das "TWBR" Register wurde mit einer 72 als Teiler beschrieben, der nach der obigen Formel berechnet wurde. Im "TWCR" "Control- Register" wurde die Schnittstelle aktiviert (TWEN)

und ebenfalls das “Auto ACK“ (TWEA). Des weiteren wurden die Interrupts freigegeben (TWIE).

5.2 RS232- Schnittstelle

In dieser Arbeit werden, wie schon mal erwähnt, zwei serielle RS232- Schnittstellen verwendet. Diese beiden Schnittstellen werden über UARTS definiert, die vom Prozessor bereitgestellt werden. Die Konfiguration dieser UARTS erfolgt, genauso wie bei I²c- Schnittstelle, über mehrere Register. Und genauso können sich diese Register von Prozessor zu Prozessor unterscheiden.

In dieser Arbeit wurden die beiden RS232- Schnittstellen identisch eingestellt. Betrieben werden sie asynchron mit einer Geschwindigkeit von 19200 Bauds, einer Nutzdatenübertragung von acht Bit Länge, einem Stoppbit und keinem Handshake. Um diese Einstellungen vorzunehmen müssen folgende Register beschrieben werden. Die nun folgenden Register stammen aus dem Manual des AT90can128 Prozessors und sind jeweils in doppelter Ausführung vorhanden, das kleine “n“ steht dabei für Register 0 oder 1, je nachdem welches Register verwendet wird.

Bit	7	6	5	4	3	2	1	0	
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	UCSR0A
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Bild 5.5: UCSR0A Register des UARTS0 vom AT90can128 [1]

Das “UCSRnA“ (USARTn Control and Status Register A) überwacht die Schnittstelle. Dabei zeigt es an wenn Daten empfangen (RXCn) aber noch nicht abgeholt wurden, eine Übertragung fehlerfrei stattgefunden hat (TXCn) oder das Datenregister leer ist (UDREn). Des weiteren meldet dieses Register, wenn verschiedene Fehler bei der Übertragung aufgetreten sind. Dies sind “Frame Error“ (Fen), “Parity Error“ (UPEn) und “Data Overrun“ (DORn). Als Einstellungsmöglichkeiten verfügt dieses Register zum einen über die Möglichkeit, bei einer asynchronen Übertragung, die Übertragungsgeschwindigkeit zu verdoppeln (U2Xn) zum anderen über die Möglichkeit einen “Multi- processor Communication Mode“ einzuschalten. Dieser ignoriert alle Übertragungen, die keine Adresse mitführen.

Bit	7	6	5	4	3	2	1	0	
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	UCSR0B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bild 5.6: UCSR0B Register des UARTS0 vom AT90can128 [2]

Dieses “UCSRnB“ (USARTn Control and Status Register B) Register übernimmt das Handling der Interrupts. In diesem Register ist es möglich die Interrupts für den Empfang (RXCIEn), dem erfolgreichen Senden (TXCIEn) und für ein leeres Datenregister (UDRIEn) zu aktivieren. Des weiteren wird in diesem Register entschieden welche Funktion die Schnittstelle erfüllt. Es kann dabei entschieden werden, ob sie nur empfangen, senden oder ob sie beides tun soll. Zusammen mit den Bits “UCSnC1“ und “UCSnC0“ des “UCSRnC“ Registers wird im “UCSZn2“ die Anzahl der Datenbits eingestellt. Bit “RXB8n“ und “TXB8n“ enthalten das neunte Bit bei einer neun Bit langen Datenübertragung.

Bit	7	6	5	4	3	2	1	0	
	-	UMSEL0	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	UCSR0C
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

Bild 5.7: UCSR0C Register des UARTS0 vom AT90can128 [3]

In dem “UCSRnC“ (USARTn Control and Status Register C) Register wird entschieden ob eine asynchrone oder synchrone Übertragung erfolgen soll (UMSELn). Darauf folgen dann zwei Bit, welche zum Einstellen des Parity Modes dienen (UPMn1/UPMn0). Dabei kann zwischen keiner Parity und Odd- oder Even- Parity entschieden werden. Nun folgt die Einstellung für die Anzahl der Stoppbits (USBSn), gefolgt von den zwei Bits, welche die Anzahl der Datenbits beeinflussen (UCSnC1/UCSnC0). Als letztes wird noch die Polarität der Clock eingestellt (UCPOLn). Dabei handelt es sich um die Einstellung auf welcher Flanke, bei einer synchronen Übertragung, Daten übertragen werden.

Als letztes ist es noch möglich die Baudrate einzustellen. Dies geschieht mit Hilfe von zwei 8-Bit Registern (UBRRnL/UBRRnH).

Bit	15	14	13	12	11	10	9	8	
	UBRR0[11:8]								UBRR0H
	UBRR0[7:0]								UBRR0L
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bild 5.8: UBRR0H/L Register des UARTS0 vom AT90can128 [4]

Diese beiden Register sind unter dem Befehl “UBRRn“ zusammengefasst. Bei einer Zuweisung wird der Wert automatisch auf beide Register aufgeteilt.

In der im folgenden Bild 5.9 gezeigten Funktion handelt es sich um die zur Initialisierung verwendete Funktion für die RS232-Schnittstelle. Diese konfiguriert die Schnittstelle auf die oben erwähnten Einstellungen.

```
void init_USART0(void)
{
    /* USART0 */
    /* setting up the USART */
    /* 16Mhz 19200 Baud No Flow Control 8 data 1 stopbit No Parity */
    UCSRC = (0<<UMSEL0)|(0<<UPM01)|(0<<UPM00)|(0<<USBS0)|(1<<UCSZ01)|(1<<UCSZ00)|(0<<UCPOL0);
    UBRR0L = 51; //19200 Baud bei 16Mhz
    UCSRB = (1<<TXEN0)|(1<<RXEN0)|(1<<RXCIE0); //Transmitter Enable, Receiver Enable, Receive Interrupt Enable
}
```

Bild 5.9: Initialisierungsfunktion der RS232- Schnittstelle

5.3 Kameramodul

Die Konfiguration der seriellen RS232- Schnittstelle der Kamera erfolgt über Jumper, welche sich auf der Platine befinden (Bild 5.10). Allerdings kann hier nur die Baudrate eingestellt werden. Die Anzahl der Datenbits und Stoppbits sind von der Firmware vorgegeben und können dem Manual entnommen werden. Die eigentliche Konfiguration der Kamera erfolgt daraufhin über diese Schnittstelle, indem Befehle an die mitgelieferte Firmware gesendet werden.

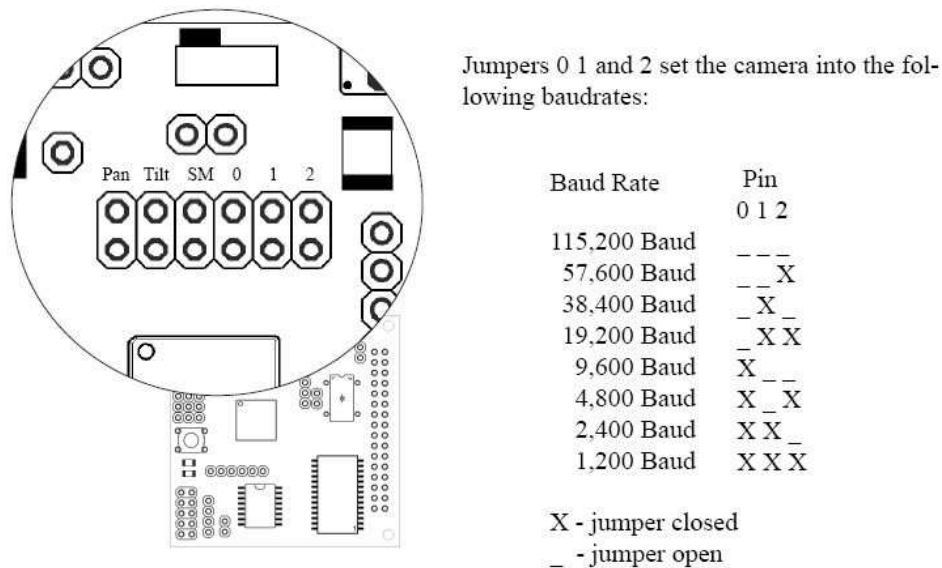


Bild 5.10: Jumperfeld der CMUcam2 zur Konfiguration der Boudrate

In dieser Arbeit wird die Kamera im Pollmode betrieben. Dies bewirkt das sie nicht ununterbrochen das Ergebnis des letzten Befehls an die Schnittstelle schickt, sondern jeden Befehl der eingeht nur einmal ausführt. Alle anderen Einstellungen wurden auf den vordefinierten Werten belassen.

```
unsigned char cam_init()
{
  //Camera einstellen
  while(send_to_Cam("RS",0xFF));          //Reset
  //while(send_to_Cam("DM 255",0x01)); //Delay Mode between transmitted char
  while(send_to_Cam("PM 1",0x02));      //Poll Mode
  return 0;
}
```

Bild 5.11: Initialisierungsfunktion der Kamera

Die in Bild 5.10 gezeigte Funktion dient zur Initialisierung der Kamera. Zuerst wird ein allgemeiner Reset ("RS") ausgeführt, daraufhin wird die Kamera noch in den Pollmode ("PM 1") versetzt.

6 Experimentelle Ergebnisse

In diesem Kapitel möchte ich auf Ergebnisse eingehen welche während der Bearbeitungszeit auftraten.

6.1 Sensorik

Durch die vier Sensoren ergeben sich 16 Möglichkeiten, wie sich Hindernisse um den Roboter verteilen können. Die Implementierung sieht ein Rückwärts fahren nicht vor, wodurch Hindernisse hinter dem Roboter nicht von belang sind. Die Anzahl der Möglichkeiten reduziert sich deshalb auf neun verschiedene mögliche Sensorwerte. Diese Anzahl ergibt sich aus drei Sensoren (links, vorne, rechts) und der Unterscheidung der beiden Fälle, bei denen alle drei ein Hindernis erkennen und der hintere Sensor den Ausschlag gibt ob weiter gefahren wird oder nicht. Folgend stelle ich diese verschiedenen Möglichkeiten und die darauf erfolgenden Reaktion in einer Tabelle dar. Ausgangslage ist dabei, dass sich der Roboter in einer Vorwärtsfahrt befindet. In der linken Spalte steht dabei das bereits oben erwähnte, vom AT90can128 Board gelieferte "Sensorbyte", welches die Vorauswertung der vier Sensoren beinhaltet. Die rechte Spalte enthält die auf die Sensorwerte erfolgende Reaktion, wie sie vom Atmega32 Controller ausgeführt wird.

Das kleine x bedeutet "keine Definition" während das große X für den hinteren Sensor steht, wobei es in diesem Fall egal ist welchen Wert dieser Sensor liefert.

<u>Sensorwert</u> <u>(x-x-x-x-Links-Vorne-Rechts-Hinten)</u>	<u>Reaktion</u>
0-0-0-0-0-0-0-X Kein Sensor	Keine
0-0-0-0-0-0-1-X Rechts	Keine
0-0-0-0-0-1-0-X Vorne	Drehung nach Rechts
0-0-0-0-0-1-1-X Vorne Rechts	Drehung nach Links
0-0-0-0-1-0-0-X Links	Keine
0-0-0-0-1-0-1-X Links Rechts	Keine
0-0-0-0-1-1-0-X Links Vorne	Drehung nach Rechts
0-0-0-0-1-1-1-0 Links Vorne Rechts	Drehung nach Rechts
0-0-0-0-1-1-1-1 Alle	Stopp

Tabelle 6.1: Sensordaten mit den dazugehörigen Reaktionen

6.2 Programmablauf

Im folgenden Diagramm (Bild 6.1) ist der Ablauf dargestellt, wie er bei einer normalen Abwicklung auf dem Atmega32 Controller auftritt. Der Atmega32 Controller beginnt am Anfang mit der Konfiguration der Schnittstellen danach überprüft er, ob der Fahrbetrieb aktiviert wurde. Wenn dies nicht der Fall ist geht das Programm weiter zur Überprüfung, ob er selbst gefunden wurde. Ist er allerdings im Fahrbetrieb, beschafft er sich die Sensordaten vom AT90can128 Controller und reagiert entsprechend (Tabelle 6.1). Wenn die Abfrage ob er selbst gefunden wurde negativ ausfällt, geht das Programm zurück zur Überprüfung auf den Fahrbetrieb. Bei positiver Auswertung wird überprüft, ob er selber den anderen gefunden hat und wenn ja wird dem AT90can128 Controller das Ende des Vorgangs mitgeteilt. Daraufhin beendet der Controller das Programm. Bei einem negativen Ergebnis der Überprüfung auf eigenen Fund, gibt der Controller den Motoren den Befehl zum Rechtsdrehen und geht wieder zur Überprüfung des Fahrbetriebs.

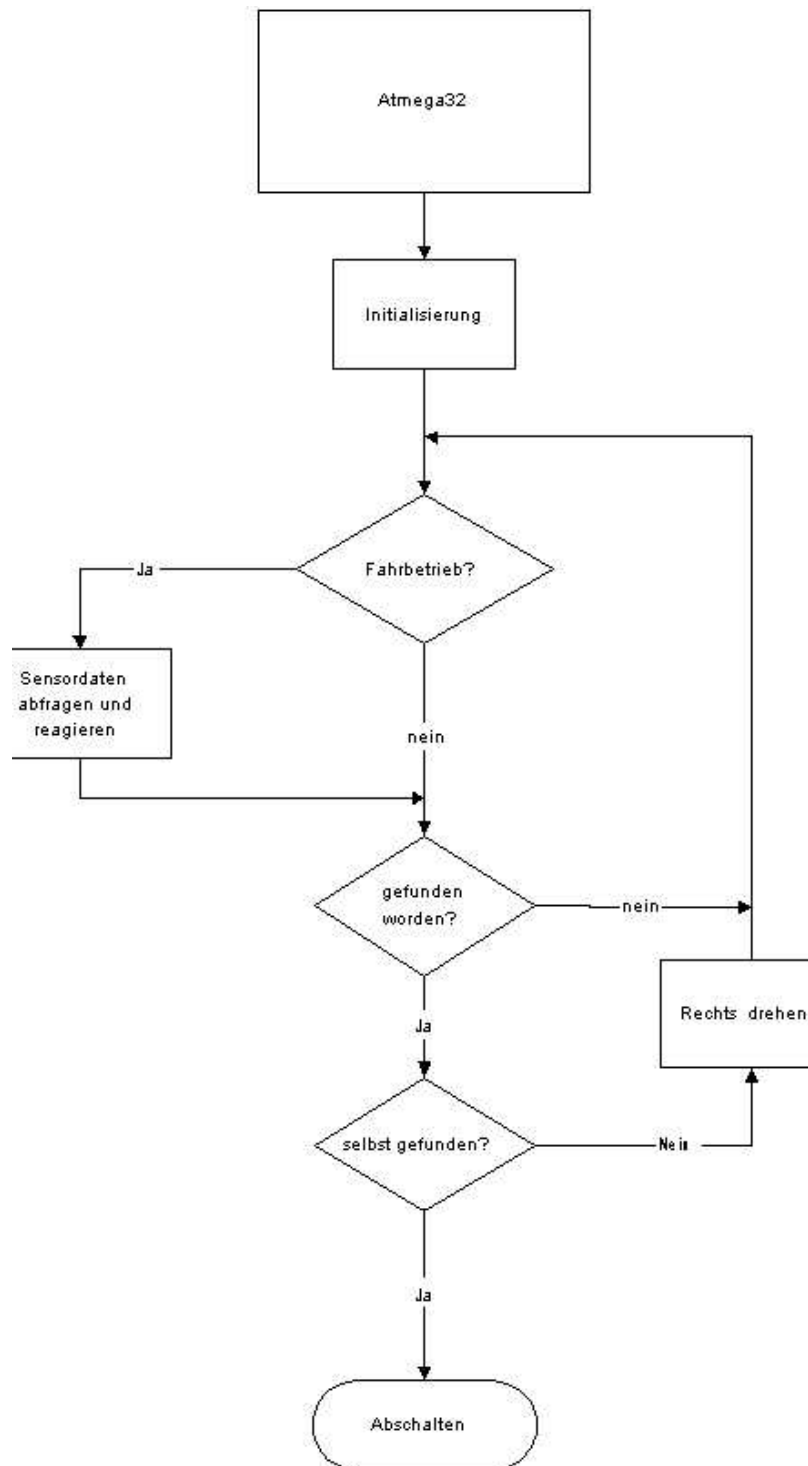


Bild 6.1: Sequenzdiagramm zur normalen Abwicklung des Atmega32

Der AT90can128 Controller beginnt nach der Initialisierung seiner Schnittstellen und Module damit seine Umgebung abzutasten. Dabei fängt er mit den Sensoren an und speichert das Ergebnis, damit es vom Atmega32 Controller abgeholt werden kann. Darauf folgt die Überprüfung der Kamera. Bei einem Fund des anderen Systems wird überprüft, ob eine Bestätigung auf eine früher gesendete Fundnachricht eingegangen ist. Wenn dies nicht der Fall ist wird eine "Fundmeldung" an den Atmega32 Controller geschickt, gefolgt von einer "Fundmeldung", welche über Funk zum anderen System geht. Sollte eine Bestätigung eingegangen sein, werden diese beiden Nachrichten unterdrückt und es geht mit der Überprüfung des Funks weiter. Bei keinem Fund geht es ebenfalls mit dem Funk weiter. Bei der Überprüfung der Funknachrichten weiter. Bei einer eingegangenen "Fundnachricht" wird dies dem Atmega32 Controller mitgeteilt. Sollte eine Bestätigung eingegangen sein, wird sich das gemerkt. Bei keiner Nachricht und nach der Verarbeitung der beiden anderen Nachrichten überprüft der Controller, ob der Vorgang durch den Atmega32 Controller beendet wurde. Wenn dies der Fall ist beendet er sein Programm ebenfalls. Anderenfalls fängt er wieder bei der Überprüfung der Sensoren an. Dieser Vorgang ist noch einmal im folgenden Sequenzdiagramm (Bild 6.2) dargestellt.

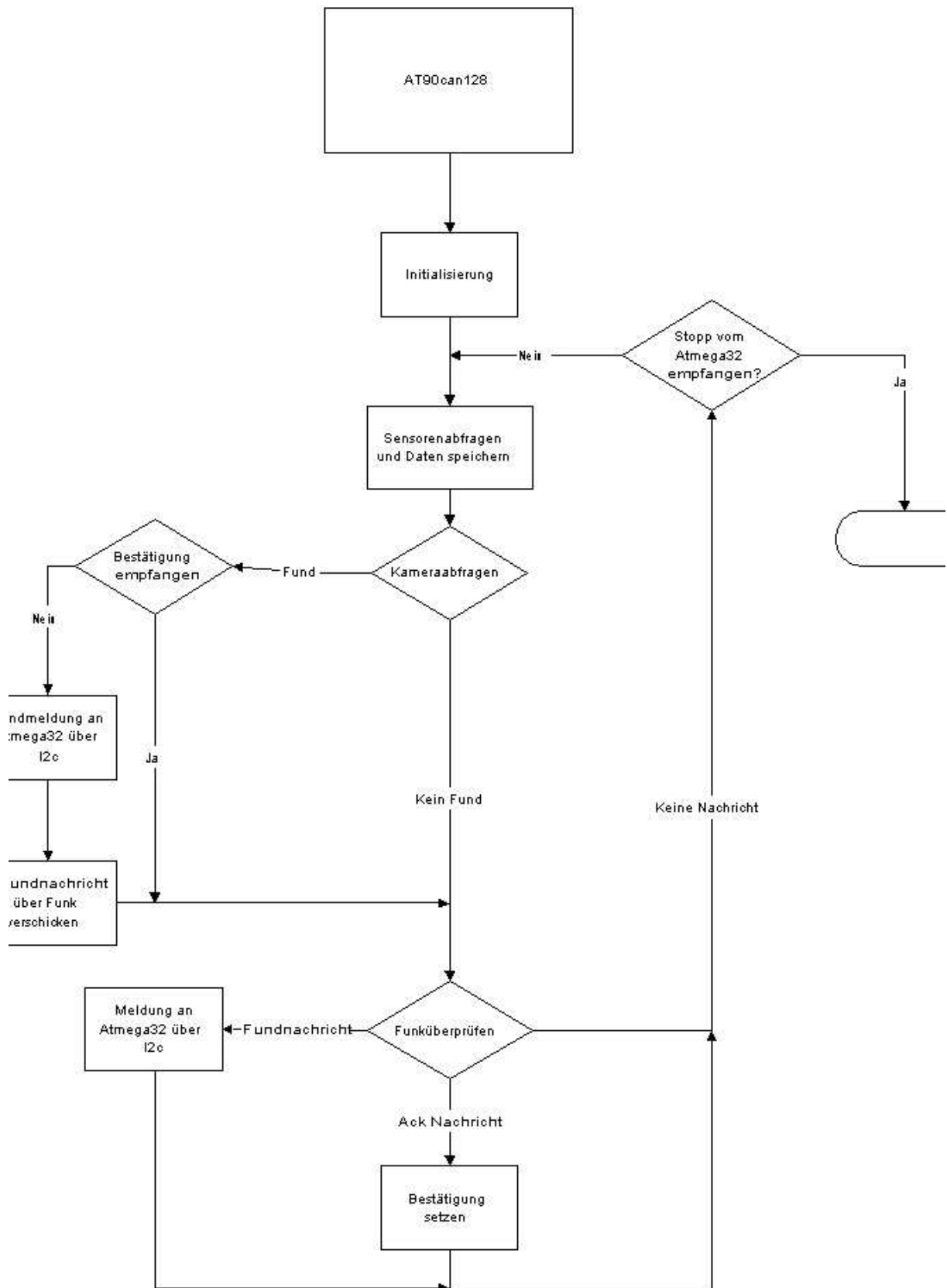


Bild 6.2: Sequenzdiagramm zur normalen Abwicklung des AT90can128

6.3 Datenübertragung

In den folgenden Diagrammen werden einzelne Datenübertragungen über die Schnittstellen dargestellt. So werden beim Reset des Funkmoduls nicht nur der Befehl für den Reset zum Modul geschickt, sondern das Funkmodul schickt auch noch seinen kompletten Datensatz zurück. Der Befehl zum Reset ist in rot und die Antwort in blau dargestellt. Schön zu sehen ist in diesem Diagramm auch der Ruhepegel der Schnittstelle.

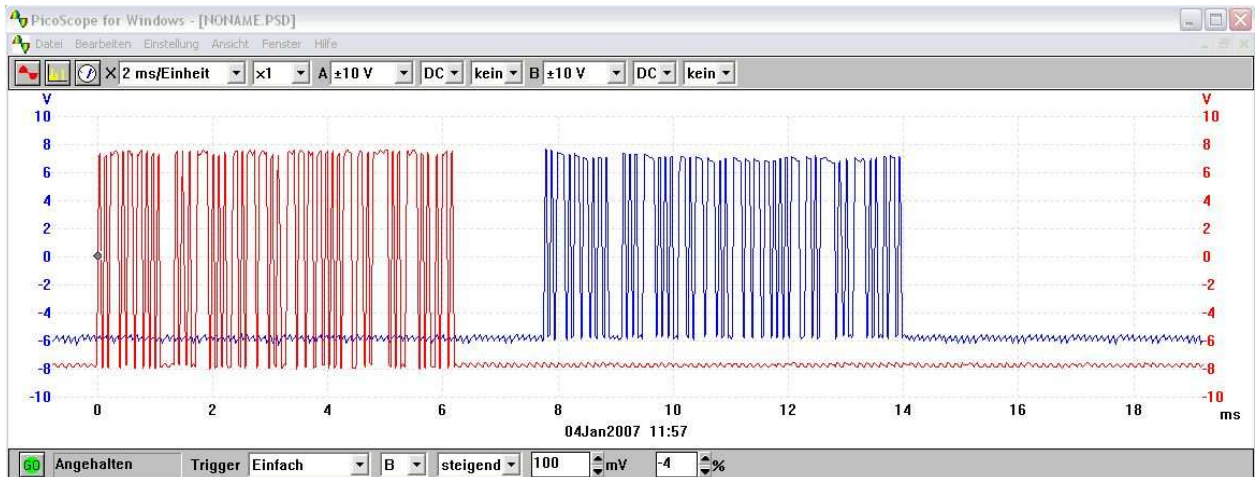


Bild 6.3: Resetbefehl und Antwort des Funkmoduls

Die für den Betrieb benötigten Nachrichten sind die “Fundnachricht“ und die “Bestätigungsnachricht“. Da sie beide einen ähnlichen Aufbau aufweisen, sind sich die Diagramme sehr ähnlich. Beide fangen mit einem “a“ an und hören auf einem “e“ auf.

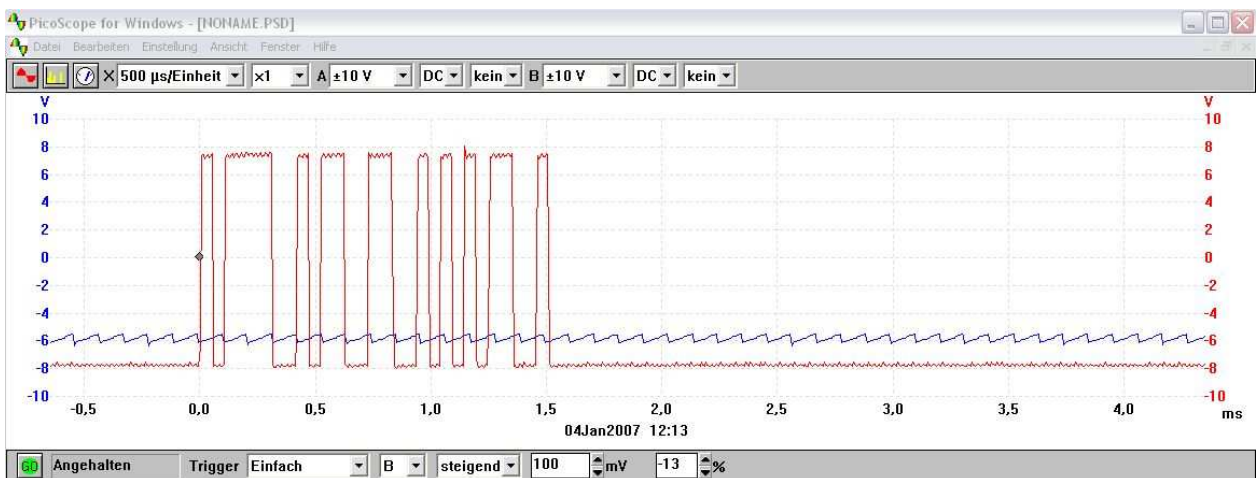


Bild 6.4: “Fundnachricht“, welche über das Funkmodul geschickt wird

In diesem Diagramm (Bild 6.3) sind Anfang und Ende der Zeichen gut zu sehen. So liegt das “a“ zwischen 0,0 und 0,5ms, darauf folgt das “f“, welches sich zwischen 0,5 und 1,0ms befindet. Zum Abschluss folgt dann das “e“, welches zwischen 1,0 und 1,5ms zu finden ist. Daraus lässt sich folgern, dass eine Übertragung eines Zeichens über eine RS232 Schnittstelle eine Dauer von ungefähr 0,5ms hat.

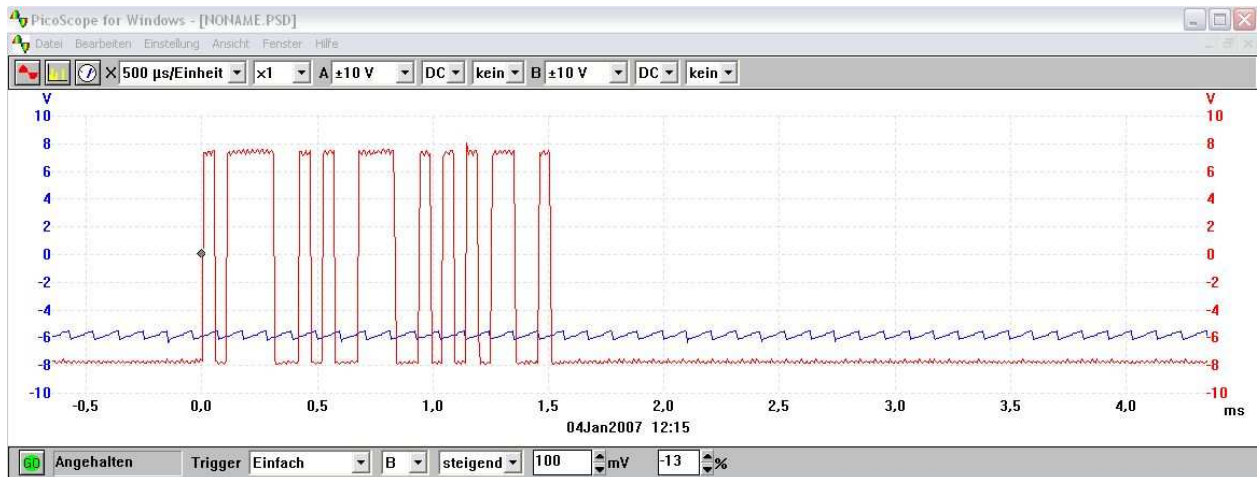


Bild 6.5: “Bestätigungsnachricht“, welche über das Funkmodul empfangen wird

Auch bei diesem Diagramm (Bild 6.4) liegt die eigentlich Nachricht zwischen 0,5 und 1,0ms. Aus diesem Diagramm lässt sich die Art der Übertragungsweise dieser Schnittstelle gut ablesen. Wird davon ausgegangen, dass ein “Highpegel“ eine 0 und ein “Lowpegel“ eine 1 bedeutet, ergibt sich folgende Übertragung aus diesem Diagramm.

0 10000110 10 11000110 10 10100110 1 = 0 a 10 c 10 e 1

Es ist noch zu beachten, dass das niederwertigste Bit immer zuerst geschrieben wird und jedes Datenbyte mit einer 0 als Startbit anfängt und mit einer 1 als Stoppbit aufhört. Bei der konfigurierten Baudrate von 19200Baud, 8- Bit Nutzdaten und einem Stoppbit, ergeben sich aus den folgenden Formeln ein Datendurchsatz von 15360Bit/s und eine Übertragungsdauer von 52µs/Bit.

$$\text{Datendurchsatz} = (\text{Baudrate} * \text{Nutzdaten} / \text{Gesamtlänge Bit/s}) = (19200 * 8 / 10 \text{ Bit/s}) = 15360 \text{ Bit/s}$$

$$\text{Übertragungsdauer} = (1 \text{ sek} / \text{Baudrate} * \text{Gesamtlänge}) = (1 \text{ sek} / 19200 * 10) = 52 \mu\text{s}$$

Aus diesen beiden Werten lässt sich dann eine Übertragungsdauer für diese beiden Nachrichten errechnen. So dauert das Senden so einer Nachricht $52 \mu\text{s} * 30 = 1560 \mu\text{s}$.

Bei der Kamera hingegen werden viel mehr Daten bei einem Vorgang übertragen. So sieht eine Abfrage der Kamera wie im folgenden Bild 6.6 gezeigt aus.

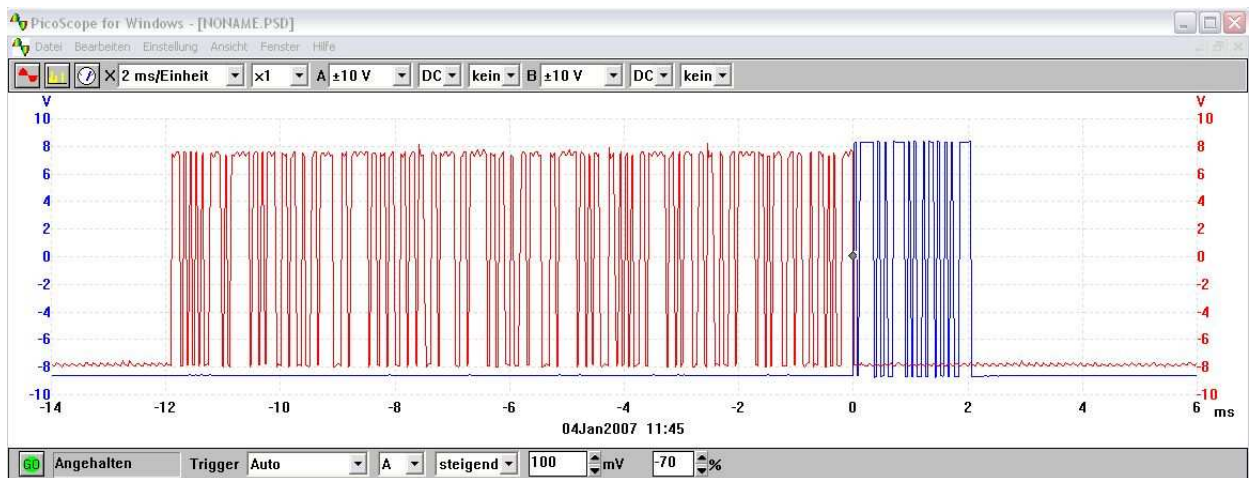


Bild 6.6: Diagramm einer Abfrage der Kamera mit 0 Nachricht

Zu beachten ist, dass in diesem Diagramm (Bild 6.6) die Skala auf 2 ms/Einheit erhöht wurde. Gezeigt ist hier die Nachricht "TC 168 220 68 95 16 16", wobei jedes Zeichen einem Byte entspricht. Daraus ergibt sich dann eine Datenlänge von 22Byte, die Leerzeichen mit eingerechnet, oder 176Bit. Die Übertragungslänge ist dann $(22 * 10\text{Bit}) = 220\text{Bit}$ und es braucht eine Übertragungsdauer von $220 * 52\mu\text{s} = 11440\mu\text{s} = 11,44\text{ms}$. Dieses Diagramm zeigt eine Antwort, in der die gewünschte Farbe nicht gefunden wurde. Deshalb kommen nach dem T nur noch 0en, wodurch sie sehr kurz wird. Eine eigentliche "Fundnachricht" schwankt sehr in seiner Länge je nachdem wie viele Stellen die gesendeten Koordinaten haben. Ein Beispiel für eine "Fundnachricht" zeigt das folgende Bild 6.7.

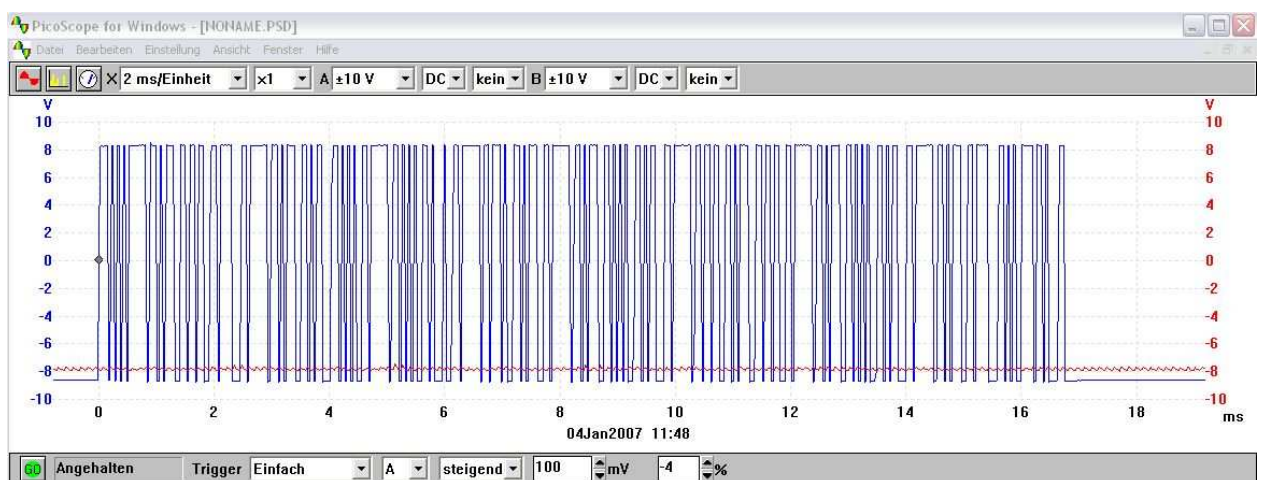


Bild 6.7: "Fundnachricht" des Kameramoduls

Die oben gezeigte “Fundnachricht“ hat eine Länge von fast 17ms woraus sich eine ungefähre Länge von 320Bit ergibt, was wiederum einer Länge von 32 Byte Nutzdaten entspricht.

Die Übertragung über die I²c- Schnittstelle hingegen sieht ganz anders aus. Im folgenden Bild 6.8 werden die Sensordaten beim AT90can128 Controller vom Atmega32 Controller abgeholt.

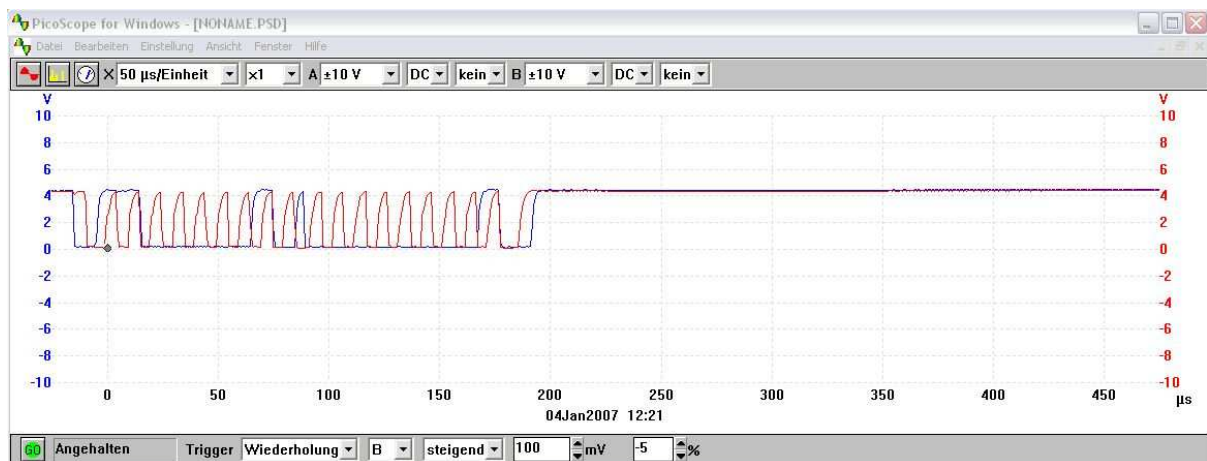


Bild 6.8: Sensordaten beim Übertragen über die I²c- Schnittstelle

Bei dieser Übertragung wird bei einer fallenden Flanke des SCL (serial clock line), in rot dargestellt, ein SDA (serial data line) Pegel, in blau dargestellt, ausgewertet. Dabei bedeutet ein “Highpegel“ eine 1 und ein “Lowpegel“ eine 0. Eine Übertragung über die I²c- Schnittstelle setzt sich immer aus einer Startsequenz gefolgt von der Adresse mit dem Read / Write Bit, welche mit einem Acknowledge des Empfängers. Danach werden entweder die Nutzdaten gesendet oder vom Slave gelesen. Am Ende wird dann vom Master noch eine Stoppssequenz gesendet. Aus diesem Diagramm ergibt sich dann folgende Übertragung.

0 11000001 1 00000001 01 = 0 C1 1 01 01 = Startsequenz(0) + Adresse/ Read(11000001) + Ack(1) + Nutzdaten(00000001) + Stoppssequenz(01)

7 Resümee

In diesem Kapitel möchte ich einen Rückblick auf diese Bachelorarbeit und einen Ausblick auf zukünftige mögliche Arbeiten geben.

7.1 Erweiterungsmöglichkeiten

Eine erste Erweiterungsmöglichkeit wäre es die autonomen mobilen Systeme auf Ultraschallsensoren umzurüsten oder zu erweitern. Dadurch ist es dem System möglich Hindernisse genauer wahrzunehmen, wodurch eine präzisere Steuerung möglich wäre. Mit den Infrarotsensoren ist es im Augenblick nicht möglich Hindernisse welche sich in einem bestimmten Winkel vom Roboter liegen wahrzunehmen.

Eine andere Möglichkeit wäre es die Suchfunktion so zu verändern, dass an die autonomen Systeme ein Signalgeber angebracht wird, welcher in die verschiedenen Richtungen der Roboter abstrahlt. Dadurch könnte das vorher langandauernde Suchverfahren so optimiert werden, dass ein schnelleres Suchen und Finden gewährleistet ist. Durch diese Erweiterung könnte beim Empfangen eines dieser Signale festgestellt werden in welcher Richtung sich das andere autonome System befindet, womit sich der Suchbereich erheblich einschränken würde. Wenn dann dem anderen autonomen System noch gesagt wird, welches Signal von ihm empfangen wurde, kann dieses ebenfalls direkt seine Suche darauf einstellen. Dies geschieht indem es seine Suche auf diesen Bereich in den dieses Signal abgesendet wurde einschränkt.

Eine weitere Möglichkeit wäre, dass die beiden Roboter ihre eigenen relativen Positionen im Raum kennen würden. Diese ließe sich zum Beispiel durch ein externes Verfahren wie zum Beispiel eine Triangulation oder anderen Verfahren zur Positionsbestimmung herausfinden. Wenn nun die eigene relative Position im Raum bekannt ist ließe, sich durch einen Vergleich dieser beiden Positionen ein optimaler Weg zum Treffpunkt berechnen.

Aufbauend auf dieser Arbeit könnte eine Kooperation entwickelt werden, welche autonome mobile Systeme verleitet gemeinsam beispielsweise einen Flur zu reinigen. Der eine fegt vorweg oder saugt, der zweite wischt und der dritte trocknet.

Das Einsatzgebiet dieser Arbeit findet sich in Bereichen in denen eine Automatisierung eingeführt oder diese erweitert werden soll. Ein Beispiel in dem so eine Automatisierung bereits stattgefunden hat ist das Container Terminal Altenwerder. Hier werden die einzelnen Container schon voll automatisch von sogenannten "Automated Guided Vehicles" zwischen dem Schiff und dem Stellplatz hin und her transportiert. Eine Aufrüstung dieser "Vehicle" wäre denkbar, so das sie gemeinsam einen überlangen Container transportieren können.

Ein weitere Einsatzgebiet wäre die Automobilbranche. In der die Fahrzeuge durch zunehmende Überwachung beim Erkennen anderer Fahrzeuge entsprechend zum Beispiel durch ein Ausweichmanöver reagieren können.

7.2 Zusammenfassung dieser Arbeit

Autonome Systeme sind darauf ausgelegt ohne Hilfe von außen, Aufgaben in einem bestimmten Rahmen zu erledigen. Autonome mobile Systeme nehmen dem Menschen Arbeiten ab, welche über seine natürlichen Grenzen hinausgehen. Mit zunehmenden Anforderungen stoßen diese Systeme ebenfalls an ihre technisch begrenzten Möglichkeiten. Intelligente Systeme sollten deshalb dahingehend erweitert werden, dass sie komplexere Aufgaben gemeinsam erledigen können. Das Ziel meiner Arbeit bestand aus einer kameragestützten Funkkommunikation, welche so eine Kooperation im Weiteren ermöglicht. Dabei ist es zum einen notwendig, dass die autonomen mobilen Systeme sich völlig frei im Raum bewegen können ohne dabei überall gegen zu fahren. Zum anderen sollen sie mittels einer Kamera das gegenüberliegende autonome System finden. Unterstützt wird dieser Suchvorgang mit einem Funkmodul, welches eine Kommunikation zwischen den beiden Systemen erlaubt. Am Ende soll sich dann folgende Situation ergeben: Beide Systeme sollen den jeweiligen "Gegenüber" mittels der Kamera gefunden haben und sich jeweils vor einander befinden. Dies führt dazu, dass sich beide Systeme am Ende der Aktion gegenüber stehen.

Zum Einsatz kamen bei dieser Arbeit Hardwarekomponenten wie das Kameramodul "CMUcam2", das Funkmodul "RN- Funk", ein Controller Board "RN- Control" bestückt mit einem Atmega32 Prozessor von Atmel, ein Controller Board mit dem AT90can128 Prozessor, ebenfalls von Atmel sowie einige Infrarotabstandssensoren "GP2D12" von der Firma Sharp.

Die einzelnen Module werden im Gesamtsystem für unterschiedliche Aufgaben und Tätigkeiten eingesetzt.

Der Atmega32 Controller übernimmt die Steuerung der beiden Motoren, die Überwachung des Suchablaufs und die Auswertung der Sensordaten, sowie die Ausführung der daraus folgenden Reaktion. Die Hauptaufgabe des AT90can128 Board ist die Informationssammlung. Um seine Umgebung beobachten und Hindernisse erkennen zu können, verfügt das Board über vier Infrarotsensoren. Die Kamera wird in dieser Arbeit eingesetzt, um ihre Umgebung nach einem bestimmten Farbmuster abzusuchen.

Die I²c-Schnittstelle dient als Kommunikationsmedium zwischen den beiden Boards. Betrieben wird die Schnittstelle im Multimasterbetrieb. Bei dieser Arbeit werden zwei serielle RS232- Schnittstellen zur Kommunikation des AT90can128 Board mit der Kamera und dem Funkmodul eingesetzt. Die Vorgehensweise in den Übertragungsfunktionen der RS232-Schnittstelle sind ebenfalls, wie die der I²C- Schnittstelle, sehr verbreitet. Die Konfiguration der Schnittstelle erfolgt über Register.

Sobald das gegenüberliegende autonome System gefunden wurde wird dieses an das Funkmodul mitgeteilt. Das Versenden kann nicht beeinflusst werden. Zur Überwachung und Überprüfung der empfangenen Funkdaten beginnen diese immer mit einem "a", welches den Anfang der Datenübertragung der eigentlichen Fundnachricht ("f","c")signalisiert. Die Daten enden immer auf einem "e", wenn alle Daten empfangen wurden. Eine "Bestätigungsnachricht" wird zurückgeschickt und die Fahrt unterbrochen.

Die Steuerung des Roboters ist auf ein Minimum der eigentlich möglichen Funktionen beschränkt. Funktionen wie zum Beispiel langgezogene Kurven blieben in dieser Arbeit unberücksichtigt. So ist es nur möglich rückwärts und vorwärts zu fahren, sowie rechts und links zu drehen. Die dabei gefahrene Geschwindigkeit wurde voreingestellt, damit sich einheitliche Ergebnisse erzielen lassen und diese nicht durch unterschiedliche Geschwindigkeiten verfälscht werden.

Diese Arbeit zeigt das mit einfachen Hardware und Softwarekomponenten möglich ist, eine Kommunikationsstruktur zwischen zwei autonomen mobilen Systemen aufzubauen. Daraus ergibt sich eine Möglichkeit deren Einsatzgebiete zu erweitern.

Das Einsatzgebiet dieser Arbeit findet sich in Bereichen in denen eine Automatisierung eingeführt oder diese erweitert werden soll. Kritisch bewertet ist die Steuerung des Roboters in dieser Arbeit jedoch auf ein Minimum der eigentlich möglichen Funktionen beschränkt. So ergaben sich einschränkende Rahmenbedingungen wie auch eine eingeschränkte Mobilität der autonomen mobilen Systeme. Annahmen wie das ein Rückwärtsfahren nicht notwendig ist, sorgen dafür das sich der Aufwand der Betrachtung der Sensordaten auf ein Minimum reduzierte. Die gemeinsame aktive Durchführung gestellter Aufgaben wurde ebenfalls nicht berücksichtigt. Die vorliegende Arbeit bezieht sich nur darauf eine Kommunikationsmöglichkeit zweier Systeme herzustellen.

Diese Arbeit bietet somit nur eine Grundlage weiterer notwendiger Entwicklungen wie z.B. die Erweiterung der autonomen mobilen Systeme mit zusätzlichen Sensoren zur genaueren Wahrnehmung der Umwelt. Weitere Möglichkeiten ergeben sich durch die Verbesserung der Suchfunktionen z.B. durch zusätzliche Angaben der relative Position im Raum. Aufbauend auf dieser Arbeit wäre es möglich eine Kooperationen zu entwickeln, bei welcher die beiden Systeme gemeinsam Aufgaben erledigen.

Literaturverzeichnis

Name vorname(ini)(autor):titel. Literaturstelle : seitenzahlen, erscheinungsjahr

1. **Atmel: Datasheet des AT90can128 doc4250.** Seite 193, Erschienen 09.05
2. **Atmel: Datasheet des AT90can128 doc4250.** Seite 195, Erschienen 09.05
3. **Atmel: Datasheet des AT90can128 doc4250.** Seite 196, Erschienen 09.05
4. **Atmel: Datasheet des AT90can128 doc4250.** Seite 197, Erschienen 09.05
5. **Atmel: Datasheet des Atmega32 doc2503.** Seite 169, Erschienen 04.06
6. **Atmel: Datasheet des Atmega32 doc2503.** Seite 172, Erschienen 04.06
7. **Atmel: Datasheet des Atmega32 doc2503.** Seite 177, Erschienen 04.06
8. **Atmel: Datasheet des Atmega32 doc2503.** Seite 179, Erschienen 04.06
9. **Robotics Institute der Carnegie Mellon University: CMUcam2 Vision Sensor User Guide.** Seite 28
10. **Sharp: GP2D12.** Seite 2
11. **Sharp: GP2D12.** Seite 3
12. **Unbekannt: Wikipedia.** <http://de.wikipedia.org/wiki/Sensor>
13. **Unbekannt: Wikipedia.** http://de.wikipedia.org/wiki/Autonome_mobile_Roboter

Anhang

Der Anhang, welcher den kompletten Programmcode, diese Arbeit als PDF Format und einige Bilder enthält befindet sich auf der beiliegenden CD.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Ort, Datum

Unterschrift