BILDUNGS ZENTRUM TECHNIK UND GESTALTUNG OLDENBURG	**Projektkurs - Dokumentation**	Schüler: Theis Ammermann, Jan-Tarek Butt, Garrit Garre, Tobias Beyer, Sören Janke	Klassse: BFI 2 Datum: 11.03.2014

Inhaltsverzeichnis:

Einleitung:

Das Ziel des Projekts war es, den Mikrocontroller „BFGTmega32“ des BZTG's über WLAN/UMTS anzusteuern. Um das sichtbar zu machen, wurde der Mikrocontroller mit Rädern ausgestattet, sodass man ihn über das Internet als eine Art ferngesteuerten Roboter ansteuern kann. Der Roboter soll vorwärts, rückwärts, nach links und nach rechts fahren können. Der Mikrocontroller wird mit einem Raspberry PI ausgestattet. Dieser bildet die Schnittstelle zur Ansteuerung über das Internet oder über das lokale Netzwerk.

Die beteiligten Gruppenmitglieder haben bereits vielfältige Erfahrungen in Bereichen, die für die Umsetzung des Projekts relevanten sind. Theis Ammermann besitzt gute Kenntnisse im Bereich Design und Dokumentation. Jan-Tarek Butt hat sehr gute Kenntnisse in den Programmiersprachen C, Python und Bash, was für die Programmierung des Roboters notwendig ist. Zusätzlich besitzt Jan-Tarek Butt auch ein umfangreiches Wissen über das Betriebssystem Linux, welches auf dem Raspbarry PI zum Einsatz kommt. Garrit Garre besitzt zusätzlich solide Kenntnisse in der C Programmierung für den Mikrocontroller. Tobias Beyer und Sören Janke haben viel Erfahrung in Protokollierung und in Projektstrukturierung, was Zeitmanagement mit einschließt. Somit konnten wir eine relativ optimierte Aufgabenverteilung erzielen.

Das Projekt wird zum Teil außerhalb des Unterrichts bearbeitet. In dem Fach Projektkurs, welches zweimal wöchentlich belegt wird, werden organisatorische Besprechungen und Planungen durchgeführt. Außerdem wird an der Dokumentation und an der Programmierung gearbeitet. Außerhalb des Unterrichts werden zum Beispiel Komponenten zum Mikrocontroller hinzugefügt.

Als Software verwenden wir „AVR STUDIO 6.1“ zur „C“ Programmierung des Mikrocontrollers. Des Weiteren wurden auf dem Raspberry PI die Programmiersprache „Python“ und die Script-Sprache „Bash“ verwendet. Zum Festhalten der Protokolle und der Dokumentation wurde Libre Office verwendet. Für die Erstellung der Projektstrukturpläne und des Flussdiagramms (siehe: Anhang 1) haben wir DIA verwendet.

Die Aufgabenverteilung sah folgendermaßen aus: Jan-Tarek Butt und Garrit Garre befassten sich hauptsächlich mit der Programmierung des Mikrocontrollers. Sören Janke und Tobias Beyer befassten sich größtenteils mit der Gestaltung der Projektstrukturpläne. Theis Ammermann beschäftigte sich grundsätzlich mit der Fertigstellung der Dokumentation. Für die Fertigstellung der Protokolle sorgten Jan-Tarek Butt, Garrit Garre und Theis Ammermann.

Um das Projekt durchführen zu können wird ein Mikrocontroller benötigt, der zu einem Fahrzeug umfunktioniert wird. Außerdem wird ein Raspberry PI vorausgesetzt sowie ein Rechner. Optional wird ein vServer im Internet benötigt um den µC über das UMTS Netz zu steuern.

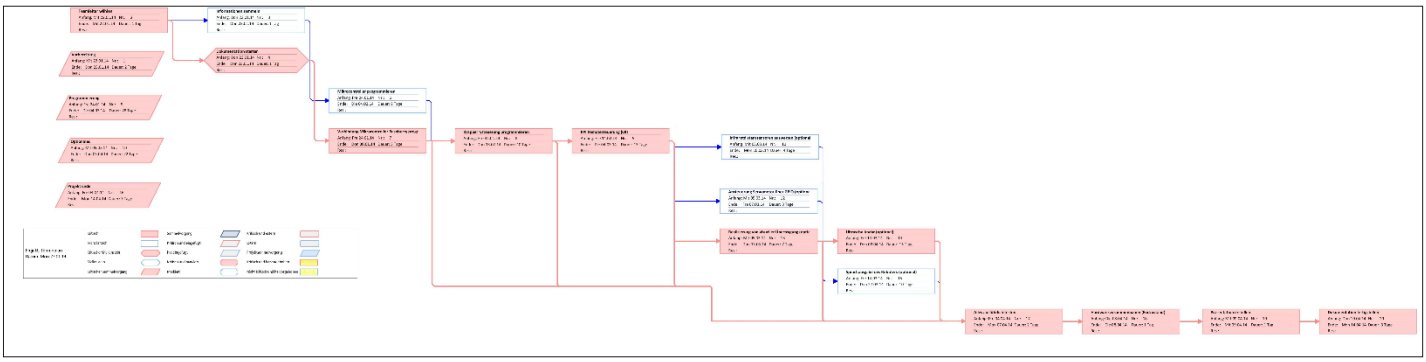
Der Mikrocontroller wird über das TCP Protokoll mit Hilfe von einer Netzwerkverbindungen gesteuert. Des Weiteren gibt es eine Infrarot-Videokamera, einen Ultraschall-Radar und Infrarot-Distanzsensoren.

Durchführung des Projektes:

Zu Beginn des Projektes wurde eine ToDo-Liste(siehe: Anhang 2) erstellt. Im Anschluss wurde eine darauf basierende Vorgangsliste (siehe: Anhang 3) geschrieben, die in einzelnen Etappen abgearbeitet wurde. Außerdem wurde die Vorgangsliste auf einen Projektstrukturplan (siehe: Anhang 4) übertragen, in welchem die Vorgänge in Kategorien eingeteilt wurden und letztendlich auch auf einen Terminplan übertragen wurden.

Der Terminplan wurde mit MS-Project angefertigt, wodurch auch automatisch ein Netzplan erzeugt wurde. Dieser wurde danach als Bild exportiert und soweit es ging komprimiert.

Da das Team schon für vorherige Projekte Terminpläne angelegt hatte, war es bereits mit der Verwendung des Programms und dem Ablauf der Erzeugung vertraut.



Für dieses Projekt gibt es grundlegende Voraussetzungen, wie insbesondere die C-Programmierung und der allgemeine Umgang mit dem Mikrocontroller. Es gibt innerhalb der Gruppe bereits Erfahrung mit der Programmierung eines zum Auto umfunktionierten Mikrocontrollers, da dies einmal ein Arbeitsauftrag im Unterricht „Technische Prozesse“ war.

Das Ziel des Projektes ist es, den Mikrocontroller „ATmega32“ in Form eines Roboters mit einigen Sensoren sowie mit der Hilfe eines Raspberry PI über WLAN anzusteuern.

Der Gruppe wurde von der Schule ein bereits zum Fahrzeug umfunktionierter Mikrocontroller zur Verfügung gestellt. Dieser wurde nach und nach mit neuen Bauteilen versehen. Als Erstes kam der Raspberry PI, der den Kontakt vom Mikrocontroller zum Internet überhaupt erst möglich macht. An einem USB-Port wurde eine WLAN-Karte angebracht. Außerdem wurde eine Kamera zur Bildübertragung per Flachbandkabel angeschlossen.

Als dies abgeschlossen war, wurden Infrarot-Sensoren links, rechts und an der Hinterseite des Fahrzeugs angebracht. Außerdem befindet sich auf dem Fahrzeug ein Schrittmotor woran ein Ultraschallsensor angebracht ist.

Infrotsensoren

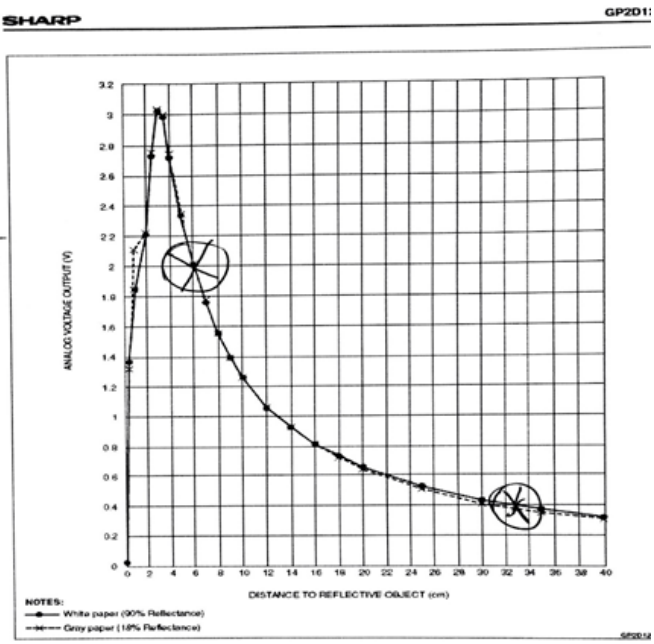
Im Folgenden kommen einige Berechnungen zur Funktions-Annäherung und Filterung der Infrarot-Distanzsensoren:

Linearisierung der Kennlinie:
Nebenstehend ist die Ausgangskennlinie U(d) des GP2D120 dargestellt.

Die Kennlinie des Distanzsensors von Sharp ist vom Typ $U(d) = \frac{1}{a \cdot d + b}$

$f(x) = \frac{1}{a \cdot x + b}$

- Bestimmen Sie die Parameter a und b
- Legen Sie für die AD-Wandlung einen geeigneten Wandlungsbereich fest
- Realisieren Sie die Distanzberechnung und überprüfen Sie ihr Ergebnis
- Nutzen Sie die Funktion d(U) für die Steuerung der Lokomotivengeschwindigkeit



Kennlinie der Sharp Infrarot Sensoren

Mathematischer Aufgabenbereich 17.02.2014

Linearisierung der Kennlinie des Distanzsensor GP2D120(4-40cm) für „Lokführer durch Handauflegen“

Mathematischer Aufsatz laut Datenblatt:

$$U(d) = \frac{1}{a*d+b}$$ mit P1(33cm/0,4V)

P2(6 cm/2V)

$$1: \quad 2V = \frac{1}{6a+b} \quad > \quad 12a + 2b = 1$$

$$2: 0{,}4V = \frac{1}{33a+b} \quad > \quad 13{,}2a + 0{,}4b = 1$$

$$a = \frac{2}{27}; b = \frac{1}{18}$$

Mathematischer Augabenbereich 19.02.2014

$1: 12a + 2b = 1$

$2: 13{,}2a + 0{,}4b = 1$

$1: 2 > 6a + b = 0{,}5 \; / \; -6a$

$\underline{b = 0{,}5 - 6a}$

$b = 0{,}5 - 6a \; in \; 2 \; einsetzen:$

$2: 13{,}2a + \; 0{,}4(0{,}5 - 6a) = 1$

$13{,}2 + 0{,}2 - 2{,}4a = 1 \; / -2a$

$10{,}8a = 0{,}8 \; / \; durch \; 10{,}8$

$a = \frac{0{,}8}{10{,}8} \approx 0{,}074$

$a = \frac{2}{27}$

$b = 0{,}5 - 6 * a$

$= 0{,}5 - 6 * \frac{2}{27}$

$= 0{,}05 = \frac{1}{18}$

Mathematischer Augabenbereich 19.02.2014

Spannungsteiler:

$$Uein = 12V$$

$$Uaus = 0V$$

Versuch 1:

$$R1 = 100k\Omega$$

$$R2 = \frac{Uaus*R1}{Uein-IB*R1-Uaus}$$

$$\frac{12}{7} - 100 = 98{,}28\Omega$$

$$\frac{U2}{Ug} = \frac{R2}{R1+R2} / durch\ Ug$$

$$U2 = Ug * \frac{R2}{R1+R2} / durch\ (R1 + R2)$$

$$U2 * (R1 + R2) = Ug * R2\ / durch\ R2$$

$$U2 * \frac{R1+R2}{R2} = Ug\ / durch\ U2$$

$$\frac{R1+R2}{R2} = \frac{Ug}{U2} \quad \frac{U2}{Ug} = \frac{R2}{R1+R2} / * (R1 + R2)$$

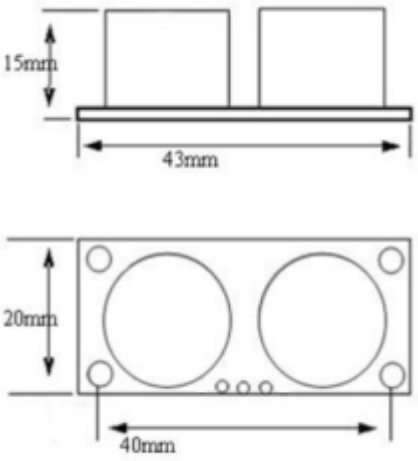
$$\frac{U2}{Ug} * R1 + R2 = R2$$

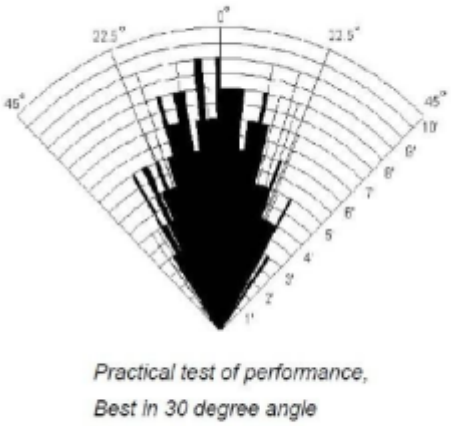
Finden einer Formel, die den Abstand in Abhängigkeit von der Spannung beschreibt.

$$U(d) = \frac{1}{a*d+b} > d(U) = \frac{1-U*b}{U*a} > d(U) = \frac{\left(\frac{1}{U}\right)-b}{a} > d(U) = \frac{1}{Ua} - \frac{b}{a}$$

Alternative Werte: $a = \frac{7}{96}$ $b = \frac{1}{16}$

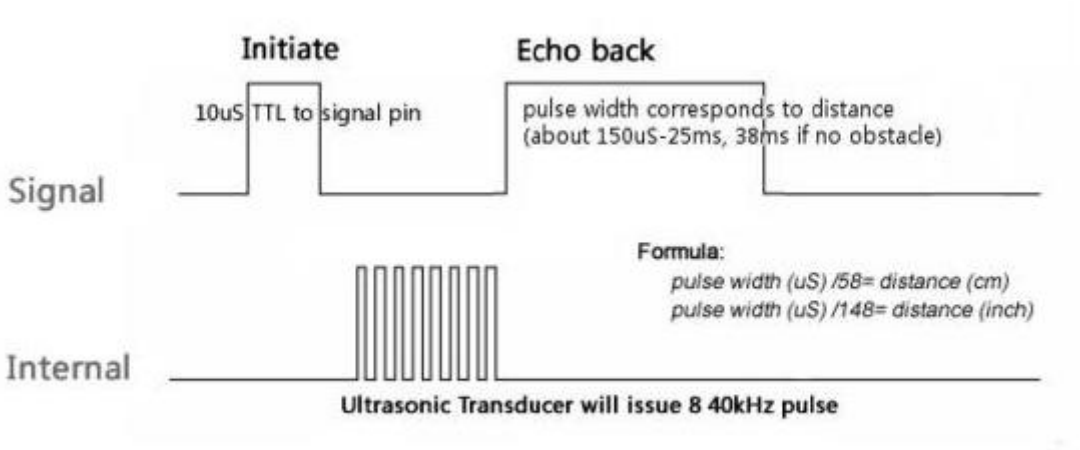
Ultraschall Sensor





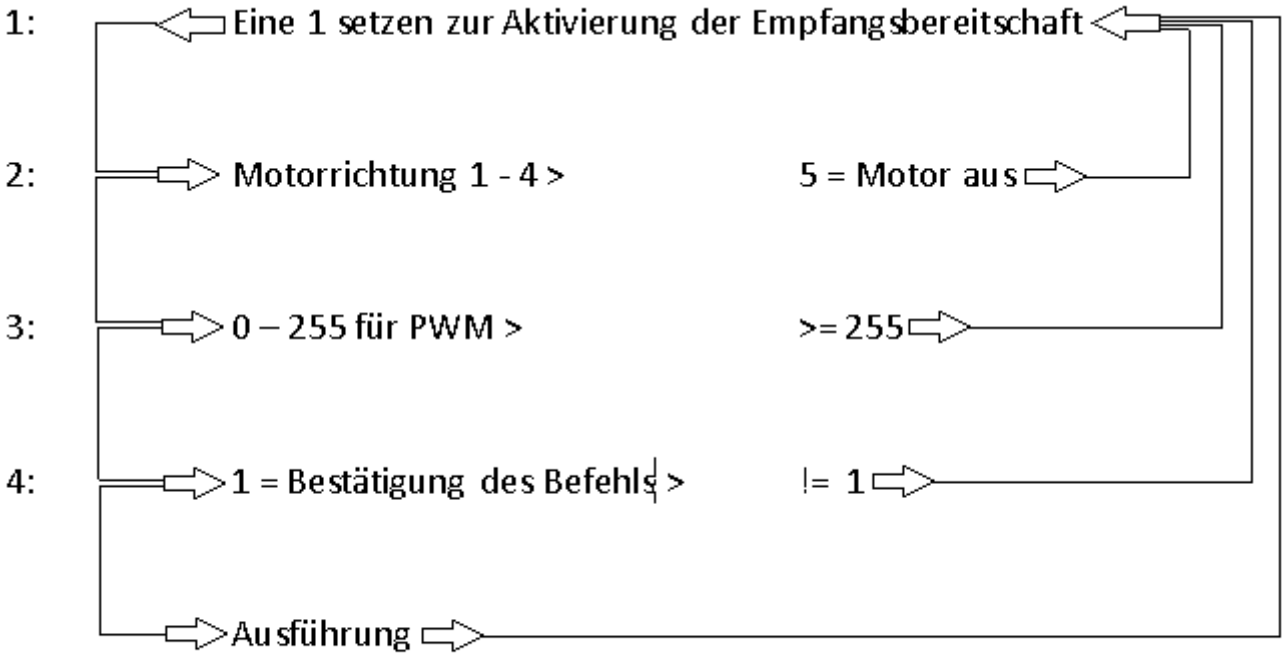
Practical test of performance,
Best in 30 degree angle

Der Ultraschallsensor hat einen ungefähren Öffnungswinkel von 30°. Die Bauform ist dem oberen Bild zu entnehmen.



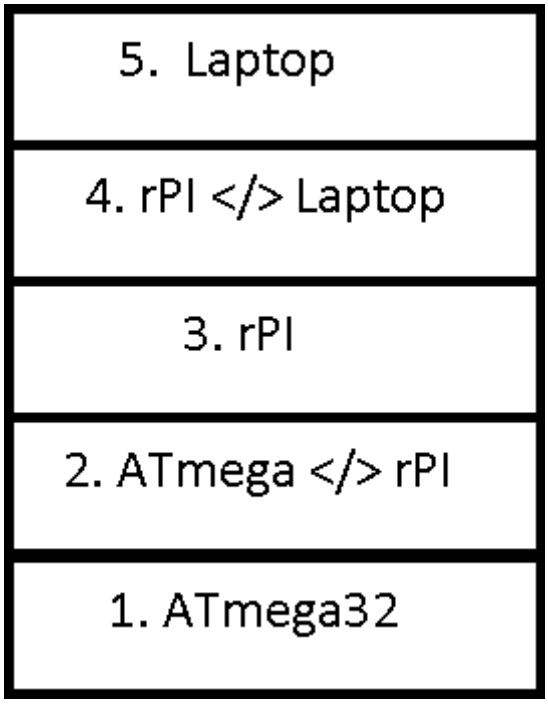
Die Ansteuerung des Ultraschall Sensors ist relativ komplex. Man muss ein circa 10 µs Sekunden langes Signal auf den Trigger-Pin legen und bekommt ein Echo Signal auf dem Echo-Pin zurück. Die Länge des Echosignals wird mit einem Timer gemessen, da die Länge mit der gemessenen Distanz des Sensors variiert.

Entwicklung des Protokolls zwischen Raspberry PI und ATmega32:



Um eine Datenübertragung zu starten wird eine 50-49 (ASCII-Zeichen == 1) gesendet. Anschließend wartet der ATmega32 auf zwei Zeichen zwischen 1-5 für den Motorzustand. Daraufhin sendet man zweimal einen Wert zwischen 0-255 für das PWM, was die Geschwindigkeit der Motoren regelt. Sobald man die PWM Werte gesetzt hat, sendet der Mikrocontroller die Werte zurück, wenn die Werte äquivalent zu den eingegebenen Werten sind, wird eine 1 gesendet, was vergleichbar mit einem ACK-Byte ist. Daraufhin führt der ATmega32 seinen Befehl aus.

Monolithischer Aufbau der Software Ebenen:



Kommunikation zwischen Raspberry PI und Laptop

rPI ist zu verstehen als Raspberry PI

Kommunikation zwischen ATmega32 und Raspberry PI

Port Belegung des ATmega32's:

PORT zuweisungen 17.02.2014

PORTA = PA1, PA2, PA3

PORTB = PB0, PB2, PB4, PB6

PORTC = PC0, PC1, PC2, PC3, PC4, PC5, PC6, PC7

PORTD = PD0, PD1, PD2, PD4, PD4, PD5, PD6

Motor = PC2, PC3, PC4, PC5	Schrittmotor Treiber = PB0, PB2, PB4, PB6
Serial = PD0, PD1	Serial Übertragung = PC6
ext. Interrupt = PD2	Mµ aktiv = PC1
PWM = PD4, PD5	Lichtschranke = PD6
IR Sensor = PA1, PA2, PA3	Schrittmotor = PB0, PB2, PB4, PB6
Ultraschall = PC0, PC7	

Reflexion:

Das Projekt lief sehr gut und die Arbeit in der Gruppe ging zügig voran, da schon Erfahrungen bei der Programmierung in der Gruppe vorhanden waren. Aber auch bei der Programmierung gab es Probleme, da keiner der Gruppenmitglieder ausreichende Kenntnisse mit der Ultraschallsensor Programmierung hatte. Nach längerer Suche im Internet und häufigen Austestens wurde das Problem jedoch gelöst. Im Integrieren des Ultraschallsensors gab es die meisten Probleme auf Grund der unerfahrenen Bearbeitung, hierin lag das größte, aber nicht das einzige Problem. Es gab noch viele kleinere Probleme wie, zum Beispiel die Ansteuerung der Infrarot-Sensoren, welche jedoch schnell gelöst werden konnten.

Obwohl das Projekt sehr viele Funktionen hat, ist es noch erweiterbar und offen für viele weitere Funktionen.

Zukünftige Projekte?

Ein weiteres Projekt könnte eine Oberflächen-Bearbeitung darstellen, bei welcher ein Kettenantrieb und eine Abdeckung der Hardware angebracht würden.

Erweiterungsmöglichkeiten?

Das Projekt könnte durch eine Erweiterung um eine C# programmierte Oberfläche einfacher zu steuern sein und auch bezüglich des Designs optimiert werden. Unter anderem könnte man eine Personen- und Gestenerkennung Implementieren.

Andere Lösungsansätze?

Es bestehen keine anderen Lösungsansätze.

4.Literaturverzeichnis:

Quellenangabe:

http://raspberrypiguide.de/howtos/raspberry-pi-gpio-how-to/

http://www.roboternetz.de/community/threads/62655-Raspberry-Pi-mit-Vb-net

http://www.mydealz.de/25645/raspberry-pi-modell-b-fur-30e-mini-rechner-zum-basteln/

https://www.google.com/search?q=Gabellichtschranke+schaltplan&safe=off&source=lnms&tbm=isch&sa=X&ei=ZGYQU-yoNIbasga17oD4CA&ved=0CAkQ_AUoAQ&biw=1364&bih=607#imgdii=_

http://www.rasppishop.de/raspberry-pi-welt/erweiterungen/1/raspberry-pi-model-b-512mb-ram-rev.-2.0

https://github.com/2tata/BFGT-Roboter/blob/master/BFGTmega32-Board/Roboter.c

http://www.thingiverse.com/thing:36305/#files

http://markslaboratory.com/2013/04/mirroring-an-stl-for-makerware/

http://kampis-elektroecke.de/?page_id=3066

http://pymotw.com/2/socket/tcp.html

http://ipv6friday.org/blog/2011/11/ipv6addresses/

Anhang:

Protokolle:

Lfd. Nr. : 1	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	09:15 Oldenburg (im BZTG) Tarek, Sören, Tobias, Garrit	
Ziel(e) der Sitzung?	Findung vom Projekt	
1. Verlauf	Besprechung der Möglichkeiten, Ideen sammeln,	
2. Ergebnisse	Folgende Projekt Möglichkeiten: µC Ethernet Steuerung, Freifunk 2.0, µC über C# steuern, 3D Drucker/ Scanner	
3. Weiterarbeit	Information sammeln, Ordnerstrukturen erstellen	

Oldenburg, den 13.01.2014	Garrit Garre	

Lfd. Nr. : 2	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	11:15 Oldenburg (im BZTG) Tarek, Tobias, Garrit	
Ziel(e) der Sitzung?	Erstellung der ToDo-Liste	
1. Verlauf	Besprechung über die Features des µC und wie wir sie umsetzen werden.	
2. Ergebnisse	Es wurde eine ToDo-Liste erstellt und festgelegt, was der µC können soll.	
3. Weiterarbeit	-Vorgangsliste -OIL anlegen	

Oldenburg, den 15.01.2014	Tobias Beyer	

Lfd. Nr. : 3	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	09:20 Oldenburg (im BZTG) Tarek, Sören, Tobias	
Ziel(e) der Sitzung?	Vorgangsliste und Programmierung	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Fertigstellung der Vorgangsliste, Prototyp des Programmes, Erstellung der OIL.	
3. Weiterarbeit	Terminplanung, Projektstrukturplan erstellen, OIL aktualisieren.	

Oldenburg, den 20.01.2014	Sören Janke	

Lfd. Nr. : 4	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	11:12 Oldenburg (im BZTG) Tarek, Sören, Tobias, Garrit	
Ziel(e) der Sitzung?	Projektstrukturplan und Programmierung	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Fertigstellung des Projektstrukturplans, Terminplan wurde erstellt.	
3. Weiterarbeit	Terminplan aktualisieren, OIL aktualisieren, mit Programmierarbeit fortfahren.	

Oldenburg, den 22.01.2014	Tobias Beyer	

Lfd. Nr. : 5	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	09:23 Oldenburg (im BZTG) Tarek, Tobias	
Ziel(e) der Sitzung?	-Terminplan überarbeiten, -Servoansteuerung implementieren	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	- Terminplan wurde überarbeitet und als JPG gespeichert	
3. Weiterarbeit	Fahrzeugbau	

Oldenburg, den 27.01.2014	Tobias Beyer	

Lfd. Nr. : 6	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	09:12 Oldenburg (im BZTG) Tarek, Tobias, Garrit, Theis	
Ziel(e) der Sitzung?	Den Roboter energieeffizienter programmieren, an der Dokumentation arbeiten	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Fahrzeug wurde erweitert, Start der Dokumentation	
3. Weiterarbeit	Dokumentation ausarbeiten, neue Fahrzeug teile integrieren	

Oldenburg, den 03.02.2014	Tobias Beyer	

Lfd. Nr. : 7	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	10:58 Oldenburg (im BZTG) Tarek, Tobias, Garrit, Theis	
Ziel(e) der Sitzung?	Den Roboter energieeffizienter programmieren	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Der Roboter wurde energieeffizienter programmiert.	
3. Weiterarbeit	Raspberry integrieren	

Oldenburg, den 05.02.2014	Tobias Beyer	

Lfd. Nr. : 8	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	09:10 Oldenburg (im BZTG) Tarek, Tobias, Garrit, Theis, Sören	
Ziel(e) der Sitzung?	Serielles Protokoll implementieren	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Raspbarry intigriert, serielles Protokoll wurde implementiert	
3. Weiterarbeit	Serielles Protokoll implementieren	

Oldenburg, den 10.02.2014	Tobias Beyer	

Lfd. Nr. : 9	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	11:09 Oldenburg (im BZTG) Tarek, Tobias, Theis	
Ziel(e) der Sitzung?	Den Schrittmotor einbauen	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Serielles Protokoll implementiert	
3. Weiterarbeit	Schrittmotor intrigieren	

Oldenburg, den 12.02.2014	Tobias Beyer	

Lfd. Nr. : 10	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	08:56 Oldenburg (im BZTG) Tarek, Tobias, Theis	
Ziel(e) der Sitzung?	An der Dokumentation weiterarbeiten, die Benennung der Protokoll-Dateien verbessern	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Schrittmotor weiterbau, OIL	
3. Weiterarbeit	Raspberry erweitern	

Oldenburg, den 17.02.2014	Tobias Beyer	

Lfd. Nr. : 11	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	08:56 Oldenburg (im BZTG) Tarek, Tobias, Theis	
Ziel(e) der Sitzung?	An der Dokumentation weiterarbeiten	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Infrarot-Distanz-Sensor anbauen	
3. Weiterarbeit	OIL aktualisieren, Programmierarbeit fortsetzen.	

Oldenburg, den 19.02.2014	Tobias Beyer	

Lfd. Nr. : 12	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	08:56 Oldenburg (im BZTG) Tarek, Tobias, Theis	
Ziel(e) der Sitzung?	Infrarot-Distanz-Sensor	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Programmierung Infrarot-Distanz-Sensor	
3. Weiterarbeit	Zusammensetzung der Dokumentation	

Oldenburg, den 24.02.2014	Tobias Beyer	

Lfd. Nr. : 13	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	08:56 Oldenburg (im BZTG) Tarek, Tobias, Theis	
Ziel(e) der Sitzung?	An der Dokumentation weiterarbeiten	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Dokumentation erweitert	
3. Weiterarbeit	Dokumentation erweitern	

Oldenburg, den 26.02.2014	Tobias Beyer	

Lfd. Nr. : 14	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	08:56 Oldenburg (im BZTG) Tarek, Tobias, Theis	
Ziel(e) der Sitzung?	An der Dokumentation weiterarbeiten	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Dokumentation erweitert	
3. Weiterarbeit	Dokumentation erweitert, OIL , Grundlegendes	

Oldenburg, den 03.03.2014	Tobias Beyer	

Lfd. Nr. : 15	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	08:56 Oldenburg (im BZTG) Tarek, Tobias, Theis	
Ziel(e) der Sitzung?	An der Dokumentation weiterarbeiten, OIL, Grundlegendes	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Es wurde an der Dokumentation geschrieben, Software-Test	
3. Weiterarbeit	Dokumentation abschließen, allgemeine Tests und Überprüfung der Dateien	

Oldenburg, den 05.03.2014	Tobias Beyer	

Lfd. Nr. : 16	Protokoll	Bemerkungen:

Arbeitsgruppe: Gruppenthema:	Tarek, Sören, Tobias, Theis, Garrit µC über Internet	
Zeit / Ort: Anwesende:	08:56 Oldenburg (im BZTG) Tarek, Tobias, Theis	
Ziel(e) der Sitzung?	Die Dokumentation fertigstellen, allgemeine Tests	
1. Verlauf	Sammlung von Vorgängen, Ideen zur Programmierung	
2. Ergebnisse	Dokumentation fertigstellen, Präsentation erstellen, Abschluss-Tests	
3. Weiterarbeit	Abgabe	

Oldenburg, den 10.03.2014	Tobias Beyer	

Code

```
#!/bin/bash
#----------------------------
# Steuerungsprogramm des Roboters
# Autor: Jan-Tarek Butt
# Datum: 12.03.2014
#----------------------------
clear

RPI_WLAN_IPV6=fe80::9644:52ff:fe04:2db6
RPI_ETH_IPV6=fe80::ba27:ebff:fecd:5984
PORT=4444
LP_PORT=4445
W_INTERFACE=wlan0
E_INTERFACE=eth0
STATUS=0

echo "Ping test..."
ping6 -c 1 $RPI_WLAN_IPV6%$W_INTERFACE
WLANW=$?
ping6 -c 1 $RPI_WLAN_IPV6%$E_INTERFACE
WLANE=$?
ping6 -c 1 $RPI_ETH_IPV6%$E_INTERFACE
ETHE=$?
ping6 -c 1 $RPI_ETH_IPV6%$W_INTERFACE
ETHW=$?

if [ $WLANW -ne 0 -a $WLANE -ne 0 -a $ETHE -ne 0 -a $ETHW -ne 0 ]; then
        echo "Roboter nicht errechtbar!"
      exit
fi;

if [ $WLANW -eq 0 ]; then
      INTERFACE="$RPI_WLAN_IPV6%$W_INTERFACE"
      echo "Wlan verbindung"
```

```
fi;

if [ $WLANE -eq 0 ]; then
        INTERFACE="$RPI_WLAN_IPV6%$E_INTERFACE"
        echo "Wlan verbindung"
fi;

if [ $ETHE -eq 0 ]; then
        INTERFACE="$RPI_ETH_IPV6%$E_INTERFACE"
      echo "Lan verbindung"
fi;

if [ $ETHW -eq 0 ]; then
        INTERFACE="$RPI_ETH_IPV6%$W_INTERFACE"
        echo "Lan verbindung"
fi;

echo "Befel eingeben:"
while true
do
      read -n 1 -s COMMAND
      echo $COMMAND

      if [ "$COMMAND" == "1"  ]; then
            nc -6lp 5001 | mplayer -fps 15 -cache 512 - &
            sleep 1
            echo "1" | nc -6 $INTERFACE $PORT
      fi;

      if [ "$COMMAND" == "w" ]; then
            echo $COMMAND | nc -6 $INTERFACE $PORT
      fi;

      if [ "$COMMAND" == "s" ]; then
            echo $COMMAND | nc -6 $INTERFACE $PORT
```

```
	fi;
	if [ "$COMMAND" == "a" ]; then
		echo $COMMAND | nc -6 $INTERFACE $PORT
	fi;
	if [ "$COMMAND" == "d" ]; then
		echo $COMMAND | nc -6 $INTERFACE $PORT
	fi;
	if [ "$COMMAND" == "q" ]; then
		echo $COMMAND | nc -6 $INTERFACE $PORT
	fi;
done
```

```
#!/bin/bash
#-----------------------
# Raspbarry PI Server zur Steuerung des Roboters
# Autor: Jan-Tarek Butt
# Datum: 12.03.2014
#-----------------------
clear

PORT=4444
BAUTRATE=57600
SCHNITSTELLE="/dev/ttyUSB0"
LP_ADRESSE="fe80::223:aeff:fe42:21b1%wlan0"
LP_PORT=4445
echo "Server start..."

while true
do
        stty -F $SCHNITSTELLE raw ispeed $BAUTRATE ospeed $BAUTRATE cs8 -ignpar -cstopb
-echo
        COMMAND=$( nc -6lp $PORT )
        echo $COMMAND

        read -i -s Line < /dev/ttyUSB0
        echo $Line

        if [ "$COMMAND" == "1"  ]; then
                raspivid -w 640 -h 480 -t 999999 -fps 10 -b 4000000 -o - | nc -6
fe80::223:14ff:fee1:aa80%wlan0 5001 &
        fi;

        if [ "$COMMAND" == "s" ]; then
                echo -n "1" >$SCHNITSTELLE
                echo -n "2" >$SCHNITSTELLE
                echo -n "3" >$SCHNITSTELLE
                echo -n "255" >$SCHNITSTELLE
                echo -n "255" >$SCHNITSTELLE
```

```
                echo -n "1" >$SCHNITSTELLE
        fi;
        if [ "$COMMAND" == "w" ]; then
                echo -n "1" >$SCHNITSTELLE
                echo -n "1" >$SCHNITSTELLE
                echo -n "4" >$SCHNITSTELLE
                echo -n "255" >$SCHNITSTELLE
                echo -n "255" >$SCHNITSTELLE
                echo -n "1" >$SCHNITSTELLE
        fi;
        if [ "$COMMAND" == "d" ]; then
                echo -n "1" >$SCHNITSTELLE
                echo -n "2" >$SCHNITSTELLE
                echo -n "3" >$SCHNITSTELLE
                echo -n "255" >$SCHNITSTELLE
                echo -n "000" >$SCHNITSTELLE
                echo -n "1" >$SCHNITSTELLE
        fi;
        if [ "$COMMAND" == "a" ]; then
                echo -n "1" >$SCHNITSTELLE
                echo -n "2" >$SCHNITSTELLE
                echo -n "3" >$SCHNITSTELLE
                echo -n "000" >$SCHNITSTELLE
                echo -n "255" >$SCHNITSTELLE
                echo -n "1" >$SCHNITSTELLE
        fi;
        if [ "$COMMAND" == "q"  ]; then
                echo -n "1" >$SCHNITSTELLE
                echo -n "5" >$SCHNITSTELLE
                echo -n "5" >$SCHNITSTELLE
        fi;
done
```

```
#--------------------
# Kamera Servo steuerung
# Autor: Jan-Tarek Butt
# Datum: 12.03.2014
#--------------------
import RPi.GPIO as GPIO
import time
import os

# Pin 26 als Ausgang deklarieren
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(26, GPIO.OUT)

while True:
        # PWM mit 50Hz an Pin 26 starten
        Servo = GPIO.PWM(26, 50)

        # Richtungseingabe
        Eingabe = raw_input("Bitte treffen Sie Ihre Wahl: ")

        # Richtung "Rechts"
        if(Eingabe == "r"):

                # Schrittweite eingeben
                Schritte = raw_input("Schrittweite: ")
                print Schritte, "Schritte nach Rechts"

                # PWM mit 10% Dutycycle (2ms) generieren
                Servo.start(10)
                for Counter in range(int(Schritte)):
                        time.sleep(0.01)

                # PWM stoppen
                Servo.stop()
```

```
# Mittelstellung einnehmen
elif(Eingabe == "m"):
        Servo.start(7)
        print "Drehung in die Mitte"
        time.sleep(1)
        Servo.stop()

# Richtung "Links"
elif(Eingabe == "l"):

        # Schrittweite eingeben
        Schritte = raw_input("Schrittweite: ")
        print Schritte, "Schritte nach Links"

        # PWM mit 5% Dutycycle (1ms) generieren
        Servo.start(5)
        for Counter in range(int(Schritte)):
                time.sleep(0.01)

        # PWM stoppen
        Servo.stop()

# Programm beenden
elif(Eingabe == "q"):
        print "Programm wird beendet......"
        os._exit(1)
        Servo.stop()
        GPIO.cleanup()

# Ungueltige Eingabe
else:
        print "Ungueltige Eingabe!"
```

```
/*
BFGTmega32.h
header-Datei für die Benutzung des µC-Boards "BFGTmega32"
des BZTG Oldenburg für das LCDisplay, die EEPROM-Nutzung,
die Nutzung der seriellen Schnittstelle, ...
am
Bildungszentrum für Technik und Gestaltung der Stadt Oldenburg
Standard: LC-Display an PORTB
*/

// Definitionen für Bitmanipulationen
#define setbit(PORT, bit)           (PORT|= (1<<bit))  // Bit setzen
#define clearbit(PORT, bit)         (PORT&= ~(1<<bit)) // Bit löschen
#define togglebit(PORT, bit)        (PORT^= (1<<bit))  // Bit invertieren

// Definitionen für LCDisplays, die kompatibel sind zu HD44780
// Pinning adapted to set interrupt pins free
// LCD control commands
#define LCD_CLEAR       0x01  /*Clear display:        0b 0000 0001      */
#define LCD_HOME        0x02  /*Cursor home:          0b 0000 0010      */
#define LCD_ON          0x0C  /*Cursor invisible:     0b 0000 1100      */
#define LCD_OFF         0x08  /*Display off:          0b 0000 1000      */
#define POS_01          0x80  /*Zeile 1 - Spalte 0    0b 1000 0000      */
#define POS_02          0xC0  /*Zeile 2 - Spalte 0    0b 1100 0000      */

// Festlegung der PORT-Pins
#define LCDPORT               PORTB
#define LCDDDR                DDRB
#define LCD_PIN_RS            2
#define LCD_PIN_E             3
#define LCD_PIN_D4            4
#define LCD_PIN_D5            5
#define LCD_PIN_D6            6
#define LCD_PIN_D7            7
#define COMMAND               0
```

```
#define DATA                1

void toggle_enable_pin(void)  // Unterstützungsfunktion zum Invertieren von Bits
{
      setbit(LCDPORT, LCD_PIN_E);
      clearbit(LCDPORT, LCD_PIN_E);
}

#ifdef _UTIL_DELAY_H_   // wenn delay.h eingebunden ist dann ...
void lcd_send(unsigned char type, unsigned char c)   // wird von lcd_write() aufgerufen
{
      unsigned char sic_c;               // backup for c
      // send high nibble
      sic_c = c;                         // save original c
      sic_c &= ~0x0f;                    // set bit 0-3 == 0
      if (type==DATA) sic_c |= (1<<LCD_PIN_RS);// data: RS = 1
      LCDPORT = sic_c; toggle_enable_pin();    // send high nibble
      // send low nibble
      sic_c = c;                         // save original c
      sic_c = sic_c<<4;                  // exchange nibbles
      sic_c &= ~0x0f;                    // set bit 0-3 == 0
      if (type==DATA) sic_c |= (1<<LCD_PIN_RS);// data: RS = 1
      LCDPORT = sic_c; toggle_enable_pin();    // send low nibble
      _delay_ms(5);                      // Wait for LCD controller
}

void lcd_init()                    // Initializing des LCDisplays - vgl. Datenblatt
{
      /* Set Port to Output */
      LCDDDR      = 0xFF;LCDPORT = 0x00;
      _delay_ms(50); // Wait for LCD

      /* 4-bit Modus config */
      setbit(LCDPORT, LCD_PIN_D5);  // Funktion Set : 4-Bit-Modus
      _delay_ms(5);
```

```
	/* 2 Lines, 4-Bit Mode */
	lcd_send(COMMAND, 0x28);	// Funktion Set : 2zeilige Display, 5x7 dots
	lcd_send(COMMAND, LCD_OFF);	lcd_send(COMMAND, LCD_CLEAR);
	lcd_send(COMMAND, 0x06);	// Entry Mode Set : Inkrement, Cursor schieben
	lcd_send(COMMAND, LCD_ON);
}

void lcd_write(char *data)    // Schreibt eine Zeichenkette (String) auf das Display
{
    while(*data){lcd_send(DATA, *data); data++;}
}

void lcd_set_cursor(uint8_t zeile, uint8_t zeichen)
// Cursor auf Zeile und Zeichen setzen
{
	uint8_t i;
	switch (zeile)
	{
		case 1: i=0x80+0x00+zeichen; break;// 1. Zeile
		case 2: i=0x80+0x40+zeichen; break;// 2. Zeile
		default: return;                   // invalid line
	}
	lcd_send(COMMAND, i);
}
#endif

void eepromwritebyte(uint16_t adresse, uint8_t data) // Byte in EEPROM schreiben
{
	while(EECR & (1<<EEWE));
	EEAR = adresse;
	EEDR = data;
	EECR |= (1<<EEMWE);
	EECR |= (1<<EEWE);
}
```

```
uint8_t eepromreadbyte(uint16_t adresse) // Byte aus EEPROM lesen
{
      while(EECR & (1<<EEWE));
      EEAR = adresse;
      EECR |= (1<<EERE);
      return EEDR;
}


#ifdef BAUDRATE              // Wenn BAUDRATE definirt ist dann ...
void uart_init(void)         // Initialisierung der seriellen Schnittstelle
{
      UBRRL = (unsigned char)(F_CPU/(BAUDRATE*16L)-1);
// Formel zur Berechnung der Baudrate variabel zur Tacktferqenz
      UCSRB |= (1<<TXEN) | (1<<RXEN) | (1<<RXCIE);
// Sendemodul aktivieren, Empfangsmodul aktivieren, // Char Interuppt Aktivieren
      UCSRC |= (1<<UCSZ1) | (1<<UCSZ0) | (1<<URSEL);
// Länge eines Zeichens auf 8 bit setzen
// URSEL benötigt um auf UCSRC zuzugreifen
}

void uart_send_char(unsigned char c)     // Sendet einen einzelnen char
{
      while (!(UCSRA & (1<<UDRE))); // Warten bis der UART zum Senden bereit
      UDR = c;                                  // „c“ in Ausgangsregister schreiben
                                                // Übertragung wird automatisch gestartet
}

void uart_send_string(char *s)     // Senset einen String auf die serielle Schnittstelle
{
      while(*s)                           //solange s != \0
      {
            uart_send_char(*s);           //einzelnes Zeichen senden
            s++;                          //weiter zum nächsten Zeichen
```

```
	}
}

int uart_get_int(void)                    // Empfängt ein integer
{
	while(bit_is_clear(UCSRA,RXC));     //warten auf Receive Complete
	return UDR;
}

void uart_send_int(unsigned int zahl, int sges)
// Sendet einen integer der Zeichenlänge sges
{
	//Ausgabe der Integerzahl zahl formatiert mit sges Stellen
	char buffer[17];
	uint8_t l=0,n;
	char *z=buffer;
	utoa(zahl,buffer,10);
	while(*z!=0){l++; z++;}                    //Bufferlänge l
	for(n=l;n<sges;n++) uart_send_char(' ');
	uart_send_string(buffer);
}

void uart_send_float(float zahl, int sges, int snach)
// Sendet einen float mit sges Zeichen und snach Nachkommastellen
{
	//Ausgabe einer Fließkommazahl mit sges Gesamtstellen.
	//Hiervon sind snach Nachkommastellen.
	//Die Nachkollastellen werden gerundet.
	char buffer[16];
	dtostrf(zahl,sges,snach,buffer);
	uart_send_string(buffer);
}

int ausgabe(void)             // Beispiel einer Ausgabe und Tastaturabfrage
{
```

```
	int z;
	uart_send_string("\n\n\r");   // 2x Zeilenumbruch, 1x Rücklauf nach links
	uart_send_string("1)  Digitale Ausgaben an den Ports B und C\n\r");
	uart_send_string("2)  Taster als digitale Eingänge an Port A\n\r");
	uart_send_string("\nGeben Sie Ihre Wahl ein: ");
	z=uart_get_int();
	z=z-48;
	return z;
}
#endif
```

```
/*
 * ATmega32 Steuerungs Programm
 *
 * Created: 15.01.2014 14:23:42
 *  Author: Jan-Tarek Butt

 ------------------------------------------------

 DDRA =||PA1|PA2|PA3|||||
 DDRB =|PB0||PB2||PB4||PB6||
 DDRC =|PC0|PC1|PC2|PC3|PC4|PC5|PC6|PC7|
 DDRD =|PD0|PD1|PD2||PD4|PD5|PD6||

 DC_Motor               =PC2,PC3,PC4,PC5
 Serial                 =PD0,PD1
 externe Interrups      =PD2
 PWM                    =PD4,PD5
 IR_Sensor              =PA1,PA2,PA3
 Ultraschall            =PC0,PC7
 Polulu                 =PB0,PB2,PB4,PB6
 Serial Übertragung     =PC6
 uC_aktiv               =PC1
 Lichtschranke          =PD6
 Schrittmotor           =PB0,PB2,PB4,PB6

  ------------------------------------------------

 */
#define F_CPU 16000000                  // Taktfrequenz fest legen
#define BAUDRATE 57200
// Baudrate fuer die serielle Schnittstelle fest legen

#include <avr/io.h>
#include <avr/sleep.h>              // lib for Sleep mode
#include <avr/interrupt.h>          // lib for Interrupts
```

```
#include <util/delay.h>             // lib for delays
#include "BFGTmega32.h"             // lib for Serial communication

int I_Wert;                         // ISR variable fuer serielle Kommunikation starten
int rPI;                            // ISR variable pruefen ob raspberry PI erreichbar ist
int unsigned zaehler= 0;
// ISR Timer0 variable fuer overflow benoetigt fuer Ultraschall Radar

ISR(INT0_vect) // Interrupt service routine loest aus wenn Raspberry errechtbar ist.
{
      cli();                        // Deaktivieren aller Interrupts
      if (PIND & (1<<PD2))          // Wenn PD2 == High dann ...
      {
            rPI = 1;                // rPI == Online
      }else
      {
            rPI = 0;                // rPI == Offline
      }
      sei();                        // Aktivieren aller Interrupts
}

ISR(USART_RXC_vect)
// Interrupt Service Routine loest aus wenn Daten empfangen werden
{
      cli();                              // Deaktivieren aller Interrupts
      while(bit_is_clear(UCSRA,RXC));     // warten auf Receive Complete
      I_Wert = UDR;
      sei();                              // Aktivieren aller Interrupts
}

ISR (TIMER0_OVF_vect)
// Interrupt Service Routine loest aus wenn timer0 Register ueberlaeuft
{
      cli();            // Deaktivieren aller Interrupts
      zaehler++;
```

```
	sei();			// Aktivieren aller Interrupts
}

void Motor_richtung(int Motor_R, int Motor_L) // Motor richtungs Steuerung
{
	if (Motor_R == 1)			// Rechter Motor
	{
		PORTC &= ~(1<<PC3);
		PORTC |= (1<<PC2);
	}
	if (Motor_R == 2)
	{
		PORTC &= ~(1<<PC2);
		PORTC |= (1<<PC3);
	}
	if (Motor_L == 3)			// Linker Motor
	{
		PORTC &= ~(1<<PC5);
		PORTC |= (1<<PC4);
	}
	if (Motor_L == 4)
	{
		PORTC &= ~(1<<PC4);
		PORTC |= (1<<PC5);
	}
	if (Motor_R == 5 || Motor_L == 5)  // Motoren Ausschalten
	{
		PORTC &= ~((1<<PC2)|(1<<PC3)|(1<<PC4)|(1<<PC5));
	}
}

int SensorFront(void) // Messung Abstand Front
{
	ADMUX |= (1 << MUX1);			// PA2 fuer AD Wandlung EIN
	ADCSRA |= (1 << ADSC);			// AD-Wandlung starten
```

```
	_delay_us(500);
	int IR[4] = {0,ADCW,0,0};		// AD-Wandlung 10-Bit
	ADCSRA |= (1 << ADSC);			// AD-Wandlung starten
	_delay_us(500);
	IR[2] = ADCW;				// AD-Wandlung 10-Bit
	ADCSRA |= (1 << ADSC);			// AD-Wandlung starten
	_delay_us(500);
	IR[3] = ADCW;				// AD-Wandlung 10-Bit
	IR[0] = ((IR[1]+IR[2]+IR[3])/3);	// Mittlung der AD-Wandlung
	ADMUX &= ~(1 << MUX1);			// PC2 fuer AD Wandlung AUS
	double d = ((1 - ((IR[0] * 0.0048828125) * (1.0 / 18.0))) / ((IR[0] * 0.0048828125)
* (2.0 / 27.0)))*2;	// Distanz Umrechnung in cm
	if (d < 5)				// Wenn d kleiner als 5 dann ...
	d = 400;				// Minimum 5cm, Meldung 400
	else if (d > 50)			// Wenn d größer als 50 dann ...
	d = 300;				// Maximum 50cm, Meldung 300
	return d;				// Rückgabe d
}

int SensorRechts(void) // Messung Abstand Rechts
{
	ADMUX |= ((1 << MUX0)|(1 << MUX1));	// PA3 für AD Wandlung EIN
	ADCSRA |= (1 << ADSC);			// AD-Wandlung starten
	_delay_us(500);
	int IR[4] = {0,ADCW,0,0};		// AD-Wandlung 10-Bit
	ADCSRA |= (1 << ADSC);			// AD-Wandlung starten
	_delay_us(500);
	IR[2] = ADCW;				// AD-Wandlung 10-Bit
	ADCSRA |= (1 << ADSC);			// AD-Wandlung starten
	_delay_us(500);
	IR[3] = ADCW;				// AD-Wandlung 10-Bit
	IR[0] = ((IR[1]+IR[2]+IR[3])/3);	// Mittlung der AD-Wandlung
	ADMUX &= ~((1 << MUX0)|(1 << MUX1));	// PC3 für AD Wandlung AUS
	double d = ((1 - ((IR[0] * 0.0048828125) * (1.0 / 18.0))) / ((IR[0] * 0.0048828125)
* (2.0 / 14.0)))*2;	// Distanz Umrechnung in cm
	if (d < 5)				// Wenn d kleiner als 5 dann ...
```

```
	d = 400;                                    // Minimum 5cm, Meldung 400
	else if (d > 50)                            // Wenn d größer als 50 dann ...
	d = 300;                                    // Maximum 50cm, Meldung 300
	return d;                                   // Rückgabe d
}

int SensorLinks(void) // Messung Abstand Links
{
	ADMUX |= (1 << MUX0);                       // PA1 für AD Wandlung EIN
	ADCSRA |= (1 << ADSC);                      // AD-Wandlung starten
	_delay_us(500);
	int IR[4] = {0,ADCW,0,0};                   // AD-Wandlung 10-Bit
	ADCSRA |= (1 << ADSC);                      // AD-Wandlung starten
	_delay_us(500);
	IR[2] = ADCW;                               // AD-Wandlung 10-Bit
	ADCSRA |= (1 << ADSC);                      // AD-Wandlung starten
	_delay_us(500);
	IR[3] = ADCW;                               // AD-Wandlung 10-Bit
	IR[0] = ((IR[1]+IR[2]+IR[3])/3);    // Mittlung der AD-Wandlung
	ADMUX &= ~(1 << MUX0);                      // PC4 für AD Wandlung AUS
	double d = ((1 - ((IR[0] * 0.0048828125) * (1.0 / 18.0))) / ((IR[0] * 0.0048828125)
* (2.0 / 12.0)))*2;     // Distanz Umrechnung in cm
	if (d < 5)                                  // Wenn d kleiner als 5 dann ...
	d = 400;                                    // Minimum 5cm, Meldung 400
	else if (d > 50)                            // Wenn d größer als 50 dann ...
	d = 300;                                    // Maximum 50cm, Meldung 300
	return d;                                   // Rückgabe d
}

int Usensor(void) // Messung Ultraschall Sensor
{
	PORTC |= (1<<PC0);
	_delay_us(10);
	PORTC &= ~(1<<PC0);                 // Ultraschall Trigger ausloesen
	while((PINC&(1<<PC7))==0);          // Wartet auf Steigende Flanke
	TCCR0 |= (1<<CS01);                 // Timer0 mit Prescale von 8 starten
```

```
	while(PINC&(1<<PC7));				// Warten solange Echo High
	TCCR0 = 0;					// Timer0 Stoppen
	double t = (((zaehler<<8)+TCNT0)/2)*0.5; // zeit in us umrechnen
	unsigned int s = (0.0343*t);		// Distanz in cm berechnen
	zaehler=0;					// Timer 0 Overflow auf Null setzen
	uart_send_string(" UL:");
	uart_send_int(s,1);
	uart_send_string(" ");			// Serielle ausgaben von Ultraschall werten
	_delay_ms(1);
	return 0;
}

int SchrittMotor(void) // Schrittmotor Steuerung fuer Ultraschall Radar
{
	PORTB &= ~(1<<PB0);			// Schrittmotor Endstufe ENABLE
	PORTB |= (1<<PB4);
	_delay_ms(5);
	PORTB &= ~(1<<PB4);
	_delay_ms(5);				// Ein step vorwaerst
	PORTB |= (1<<PB0);			// Schrittmotor Endstufe DIESABLE
	int Smotor = 0;
	if (PIND & (1<<PD6))			// Wenn Lichtschranke ausloest dann ...
	{
		Smotor = 1;
	}
	Usensor();				// Ultraschall Messung Ausloesen
	return Smotor;
}

int Sensor_Ausgabe(void) //Ausgabe der Sensoren
{
	int Sensoren[4]={SensorFront(),SensorRechts(),SensorLinks(),SchrittMotor()};
// Array, Sensoren auslesen
	uart_send_string("SF:");
	uart_send_int(Sensoren[0],1); // Infrarot Front Sensor ausgeben
```

```
	uart_send_string(" SR:");
	uart_send_int(Sensoren[1],1); // Infrarot Rechter Sensor ausgeben
	uart_send_string(" SL:");
	uart_send_int(Sensoren[2],1); // Infrarot Linker Sensor ausgeben
	uart_send_string(" SM:");
	uart_send_int(Sensoren[3],1);
// Schrittmotor Referenz Signal der Lichtschranke ausgeben
	uart_send_string("\n\r");          // Zeilenumbruch
	_delay_ms(1);
	return 0;
}

int main(void)
{
	uart_init();                              // Serial Inizialisirung
	DDRC = 0b01111111;
	DDRB = 0b11111111;
	DDRA = 0b11110001;
	DDRD &= ~((1 << PD2)|(1<<PD6));           // INT0 und Lichtschranke input...
	DDRD |= (1<<PD4)|(1<<PD5);                // PWM Ausgaenge festlegen
	ADCSRA = 0b10000111;                      // Prescaler 128
	ADMUX |= (1<<REFS0);                      // 5V Referenz Spannung
	TCCR1A |= (1<<WGM10)|(1<<COM1A1)|(1<<COM1B1);
	TCCR1B |= (1<<WGM12)|(1<<CS12);           // Prescaler 256
	GICR |= (1<<INT0);                        // IRS INT0 externer Interrupt aktivieren
	MCUCR |= (1<<ISC00);                      // ISR INT0 Reaktion auf jede aenderung
	PORTB |= ((1<<PB0)|(1<<PB6));             // Schrittmotor Endstufe aus schalten
	TIMSK |= (1<<TOIE0);                      // Timer0 Overflow Interrupt erlauben

    while(1)
    {
        sei();                                // Aktivieren aller Interrupts
        uart_init();                          // Serial Inizialisirung
        if (rPI == 1)                         // Wenn INT0 == High dann ...
        {
```

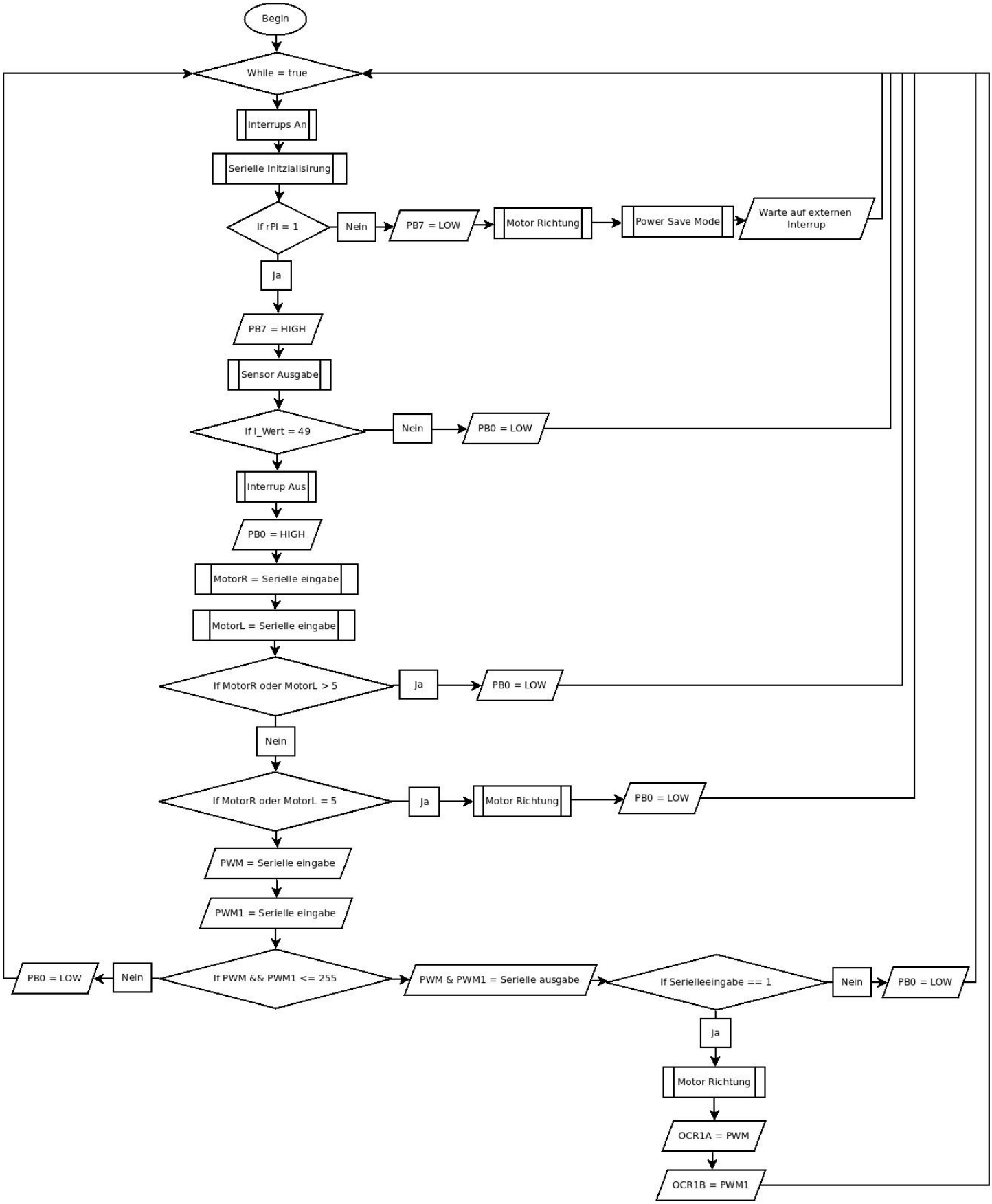
```
                PORTC |= (1<<PC1);
                Sensor_Ausgabe();            // Pereperie auswerten
                if (I_Wert == 49)            // Wenn Eingabe == 49 dann ...
                {
                      cli();                 // Deaktivieren aller Interrupts
                      I_Wert = 0;
                      PORTC |= (1<<PC6);
// anzeigen das Daten jetzt uebertragen werden koennen
                      int Motor_R = (uart_get_int()-48);
// Richtungs Vorgabe der Motoren 1-2 Vor-Rueckwaerts 5 = Motoren Aus
                      int Motor_L = (uart_get_int()-48);
// Richtungs Vorgabe der Motoren 3-4 Vor-Rueckwaerts 5 = Motoren Aus
                      if(Motor_R > 5 || Motor_L > 5)
// Wenn Eingabe größer als 5 dann ...
                      {
                            PORTC &= ~(1<<PC6);
// anzeigen das Daten nicht mehr uebertragen werden koennen
                            return main();
// Zurueck an denn Anfang der main Funktion
                      }
                      if(Motor_R == 5 || Motor_L == 5)
// Wenn Eingabe == 5 dann ...
                      {
                            Motor_richtung(Motor_R,Motor_L);
// uebergabe an Motor_richtung
                            PORTC &= ~(1<<PC6);
// anzeigen das Daten nicht mehr uebertragen werden koennen
                            return main();
// Zurueck an denn Anfang der main Funktion
                      }
                      int PWM[4] = {0,(uart_get_int()-48),(uart_get_int()-48),
(uart_get_int()-48)};   // Annahmen der Motor Geschwindigkeit 0-255
                      PWM[0] = (PWM[1]*100+PWM[2]*10+PWM[3]);
                      int PWM1[4] = {0,(uart_get_int()-48),(uart_get_int()-48),
(uart_get_int()-48)};   // Annahmen der Motor Geschwindigkeit 0-255
                      PWM1[0] = (PWM1[1]*100+PWM1[2]*10+PWM1[3]);
```

```
                        if (PWM[0] <= 255 && PWM1[0] <= 255)
// Wenn PWM kleiner als 255 dann ...
                        {
                              uart_send_string("MR: ");
                              uart_send_int(Motor_R,1);
                              uart_send_string(" ");
                              uart_send_int(PWM[0],1);
                              uart_send_string("\n\r");          // Sende Zeilenumbruch
                              uart_send_string("ML: ");
                              uart_send_int(Motor_L,1);
                              uart_send_string(" ");
                              uart_send_int(PWM1[0],1);
                              uart_send_string("\n\r");          // Sende Zeilenumbruch
                              if ((uart_get_int()-48) == 1)
// Bestaetigung der uebertragenen werte wenn Eingabe == 1 dann ...
                              {
                                    Motor_richtung(Motor_R,Motor_L);
                                    OCR1B = PWM[0];              // PWM setzen
                                    OCR1A = PWM1[0];             // PWM setzen
                              }
                        }
                        PORTC &= ~(1<<PC6);
// anzeigen das Daten nicht mehr uebertragen werden koennen
                  }else
                  {
                        PORTC &= ~(1<<PC6);
// anzeigen das Daten nicht mehr uebertragen werden koennen
                  }
            }
            else
            {
                  PORTC &= ~(1<<PC1);
                  Motor_richtung(5,5);                // uebergabe an Motor_richtung
                  set_sleep_mode(SLEEP_MODE_IDLE);    // sleep mode Aktivieren
                  sleep_mode();                       // uC geht Ideln
```

```
            }
        }
    }
```

Zusätzliches Material:

Flussdiagramm (Anhang 1)



Todo-liste: (Anhang 2)

Stepp 1 Mikrocontroler

Was soll der Mikrocontroler können?

- I/O Steuerung
- ADW Steuerung
- PWM Steuerung
-I2C Steuerung

Stepp 2 Microcontroler > Rassberry Pi

Wie soll der Microcontroler gesteuert werden?

- GPI/O Pins
- Serielle Steuerung
- I2C Steuerung
- JTAG Steuerung

Stepp 3 Rasberry Pi steuerung

Was soll der Rasberry können?

- Remoute Schnittstelle spielen
- Server spielen
- Protokoll Sockel bilden
- Viertual Serial Com emulirung
- SFTP Server
- Netcat
- SSH
- TCP-/IP-Stack
- ?

Stepp 4 Rasberry Pi remoute Steuerung

Wie soll der Rasberry gesteuert werden Windows/Linux?

- C# Programm (Windows)
- Webinterface (OS unabhängig)
- C Programm (Linux)
- Java Programm (Windows und Linux)
- (Video Stream Implementierung ins Programm)

Vorgangsliste: (Anhang 3)

Nr.	Vorgang	Dauer (Tage)	Vorgänger
1	Teamleiter wählen✔	1	-
2	Informationen sammeln✔	1	1
3	Dokumentation starten✔	M	1
4	Mikrocontroller programmieren✔	8	3
5	Verbindung mit Mikrocontroller und Raspberry Pi programmieren✔	5	3
6	Raspberrysteuerung programmieren✔	10	5
7	RPI Remotesteuerung (C#) (optional)	13	6
8	Infrarotdistanzsensoren auswerten (optional)✔	4	7
9	Ansteuerung Servomotor über GPIO (optional)✔	3	7
10	Realisierung von visueller Übertragung (optional)✔	7	7
11	Ultraschallradar (optional)✔	15	10
12	Sprachausgabe des Roboters (optional)	10	10
13	Alles ausführlich testen✔	2	4-12
14	Hardware zusammenbauen (Endzustand)✔	1	13
15	Präsentation erstellen✔	1	14
16	Dokumentation fertigstellen✔	3	15

Projektstrukturplan: (Anhang 4)