Bildungszentrum für Technik und

Gestaltung Oldenburg (BZTG)

Straßburger Str. 2

26123 Oldenburg

Tel.: 0441-98377-0

Facharbeit

Netzwerk gesteuerter Roboter

Verfasser: Jan-Tarek Butt

Betreuungslehrer: StR Rainer Lüllmann

Inhaltsverzeichnis

Einleitung

Das Ziel der Facharbeit war es, den Mikrocontroller „BFGTmega32“ des BZTG's über WLAN/UMTS anzusteuern. Um das sichtbar zu machen, wurde der Mikrocontroller mit Rädern und Sensoren 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. Der Raspberry PI ist mit einen Debian Wheezy Linux ausgestattet. Als Software wurde mit „AVR STUDIO 6.1“ zur „C“ Programmierung des ATmega32 Mikrocontrollers gearbeitet. Des Weiteren wurden auf dem Raspberry PI die Programmiersprache „Python“ und die Script-Sprache „Bash“ verwendet. Das Steuerungsprogramm wurde ebenfalls in Bash geschrieben. Um eine Übersicht über die C Programmierung des AT-Mega32 Mikrocontroller Boards zu behalten wurde ein Flussdiagramm (siehe: Anhang) mit DIA erstellt. Um eine Steuerung über das UMTS Netz zu realisieren wird ein vServer im Internet benötigt. Dieser vServer muss eine VPN (Virtual Private Network) Verbindung zum Raspberry PI und zum Steuerungsrechner aufbauen. Wenn der Roboter nur im lokalen WLAN gesteuert werden soll genügt ein AD-Hoc Netzwerk.

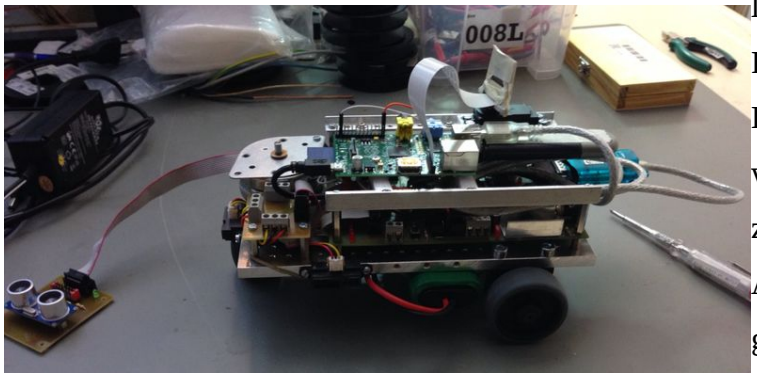


Abbildung 1 Vollständiger Roboter

Durchführung des Projektes

Zu Beginn des Projektes wurde eine ToDo-Liste (siehe: Anhang) erstellt. Im Anschluss wurde eine darauf basierende Vorgangsliste (siehe: Anhang) geschrieben, die in einzelnen Etappen abgearbeitet wurde. Außerdem wurde die Vorgangsliste auf einen Projektstrukturplan (siehe: Anhang) übertragen, in welchem die Vorgänge in Kategorien eingeteilt wurden. Für dieses Projekt gibt es grundlegende Voraussetzungen, wie insbesondere die C-Programmierung und der allgemeine Umgang mit Mikrocontrollern. 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/UMTS anzusteuern. Von der Schule wurde ein Mikrocontroller Board und verschiedene Sensoren zur Verfügung gestellt. Der Mikrocontroller wurde nach und nach mit neuen Bauteilen versehen.

Beschreibung der Konstruktion

Angefangen wurde mit dem Raspberry PI, der den Kontakt vom Mikrocontroller zum Internet überhaupt erst möglich macht. An den ersten USB-Port wurde eine WLAN-Karte angebracht, der zweite USB-Port wurde für den USB-Seriell Adapter benötigt. Außerdem wurde eine Infrarot-Kamera zur Bildübertragung per Flachbandkabel angeschlossen; diese bietet eine visuelle Übertragungsmöglichkeit. Durch den nicht vorhandenen Infrarotfilter kann man die Kamera sowohl in einer hellen als auch in einer dunklen Umgebung nutzen. Die IR-Kamera wurde auf einem digitalen Servomotor befestigt der über die GPIO Pins des Raspberry gesteuert wird. Als dies abgeschlossen war, wurden Infrarot-Sensoren links, rechts und an der Hinterseite des Fahrzeugs angebracht. Auf dem Roboter befindet sich auch noch ein Schrittmotor der über einen Polulu Schrittmotortreiber auf der Unterseite des Roboters vom Mikrocontroller gesteuert wird. Seitwärts am Schrittmotor ist eine Lichtschranke angebracht, diese liefert ein Referenzsignal was zur Kalibrierung des Ultraschallradars notwendig ist. Auf dem Schrittmotor ist ein HC-SR04 Ultraschallsensor angebracht. Dieser erkennt in einem Umkreis von ca. 5 Metern Hindernisse. Der Mikrocontroller wird mit einem seriellen Interface über den Raspberry PI gesteuert. Auf

den Raspberry PI wird mit einem nativen TCP/IP Socket gearbeitet. Über das TCP Protokoll erhält der Roboter seine Befehle und sendet seinen Status und sämtliche Sensorwerte an den Steuerungsrechner. Die Verbindung zum Raspberry PI wird über IPv6 realisiert. Um manuell auf dem Raspberry PI zu arbeiten kann man sich über SSH (Secure Shell) anmelden.

Fotos, Zeichnungen, Baupläne, etc.

Mikrocontroller

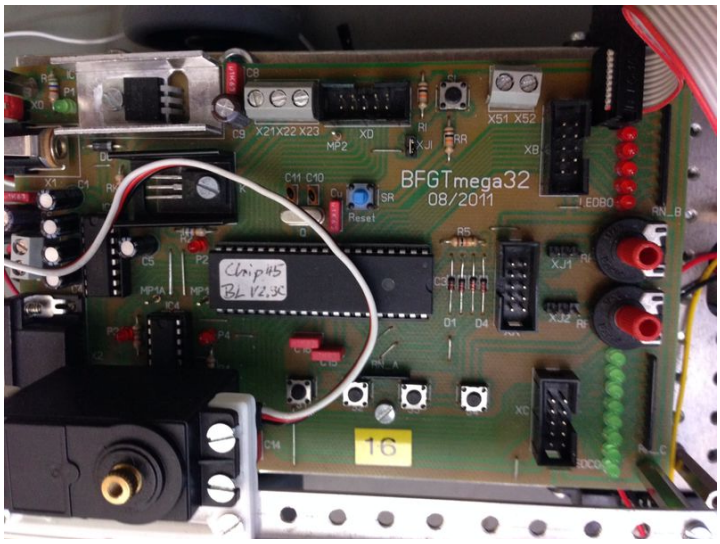


Abbildung 2: Mikrocontroller "BFGTmega32"

Servomotor

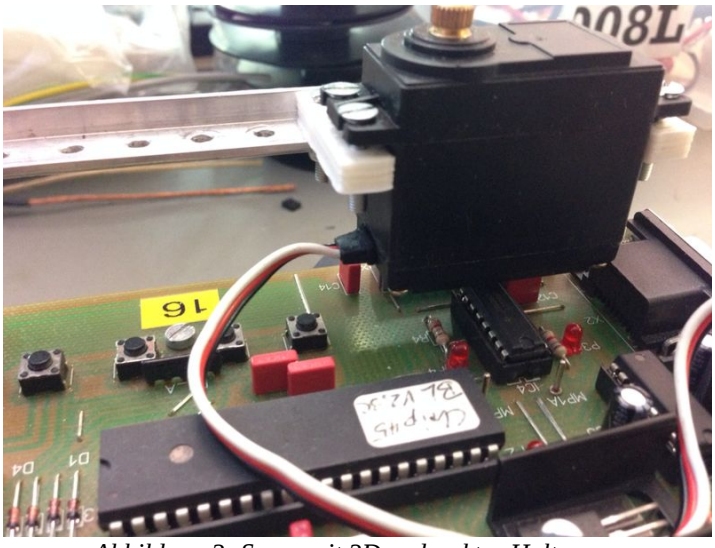


Abbildung 3: Servo mit 3D gedruckter Halterung

Spannungsversorgung für Roboter Komponenten

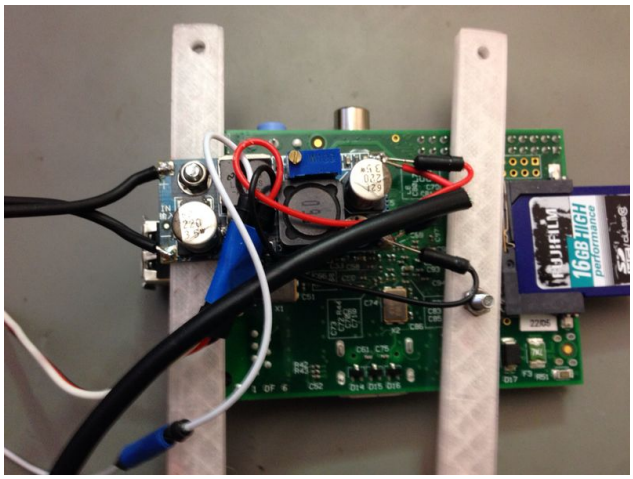


Abbildung 4: 5V Schaltregler für Raspberry PI und Servo

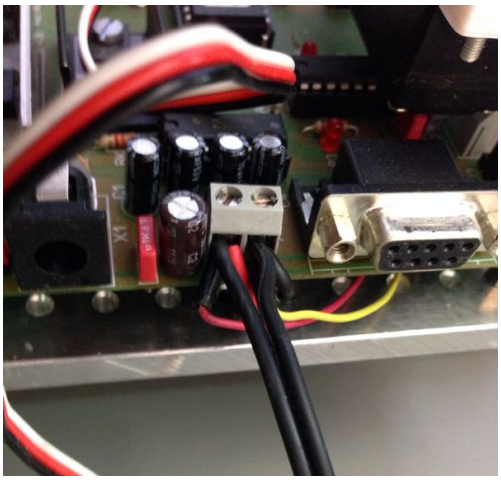


Abbildung 5: Spannungsversorgung des Roboters (7-12V)

GPIO Pins des Raspberry PI

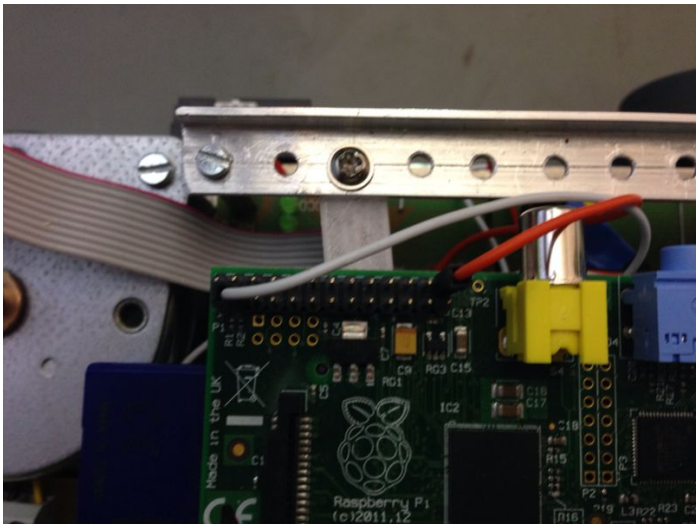


Abbildung 6: (Orangenes Kabel ist für ein PWM Signal des Servomotors)(Weißes Kabel ist für einen Interrupt Pin am ATmega32)

Schrittmotor Treiber

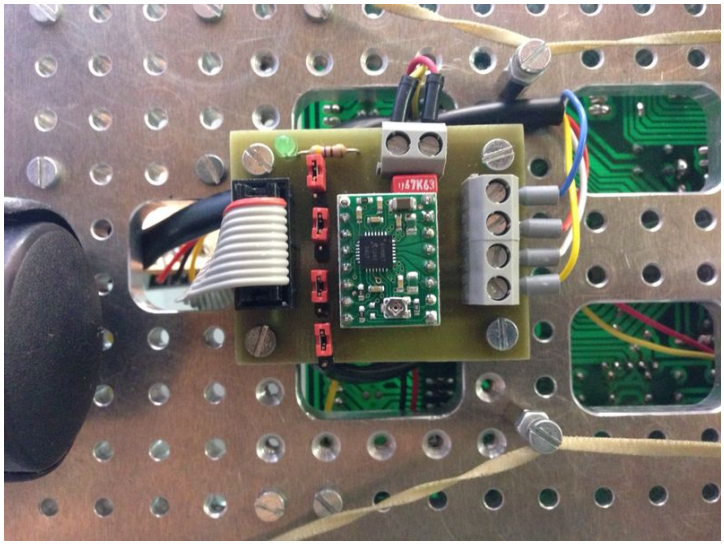


Abbildung 7: Polulu Schrittmotortreiber für das Ultraschall Radar

Testlauf der Roboter Komponenten

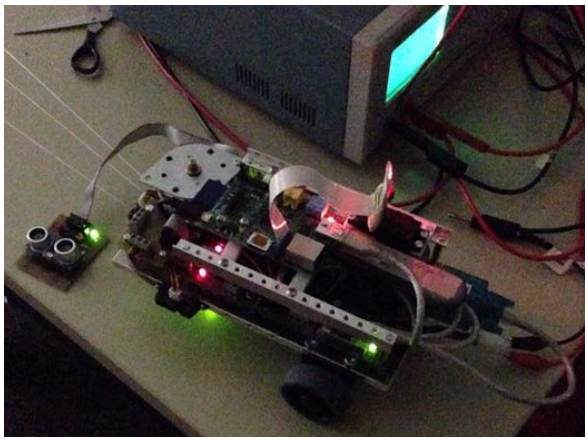


Abbildung 8: Prüfung ob alle Komponenten Ordnungsgemäß angeschlossen sind

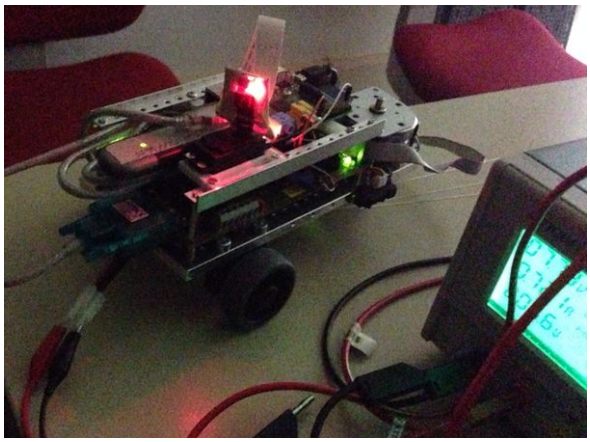


Abbildung 9: Prüfung der Kamerastream Übertragung

Infrarotsensoren

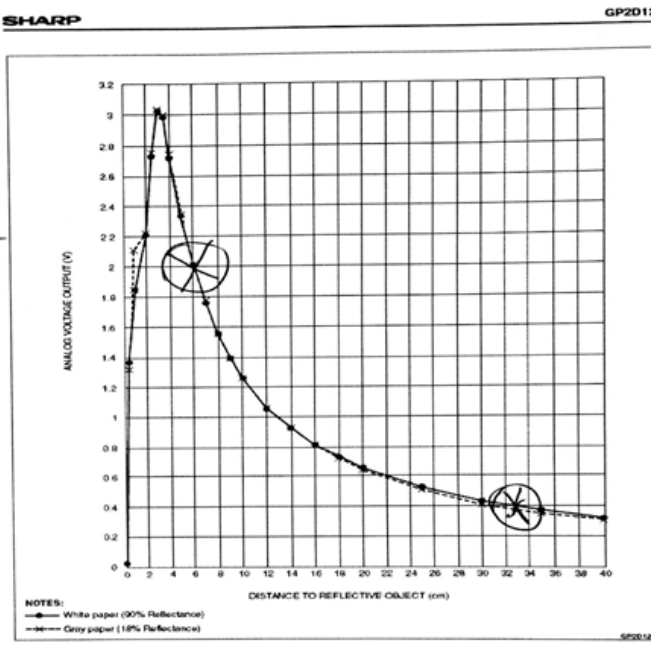
Im Folgenden kommen einige Berechnungen zur Funktions-Annäherung und Filterung der Infrarot-Distanzsensoren:

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:

$$R1 = 100k\Omega$$

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

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

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

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

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

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

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

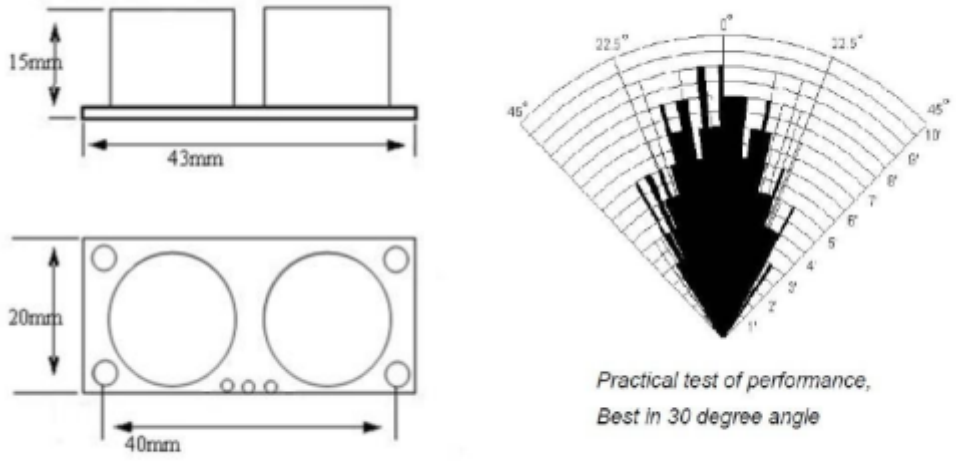
$$\frac{U2}{Ug} * R1 + R2 = R2$$

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

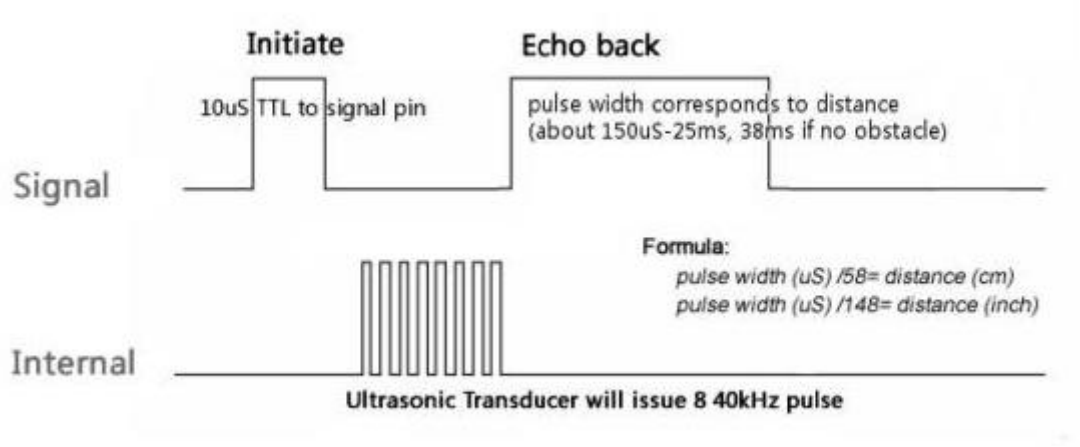
$$U(d) = \frac{1}{a*d+b} > d(U) = \frac{1-U*b}{U*a} > d(U) = \frac{\left(\frac{1}{U}\right)-b}{a} > d(U) = \frac{1}{Ua} - \frac{b}{a}$$

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

Ultraschall Sensor



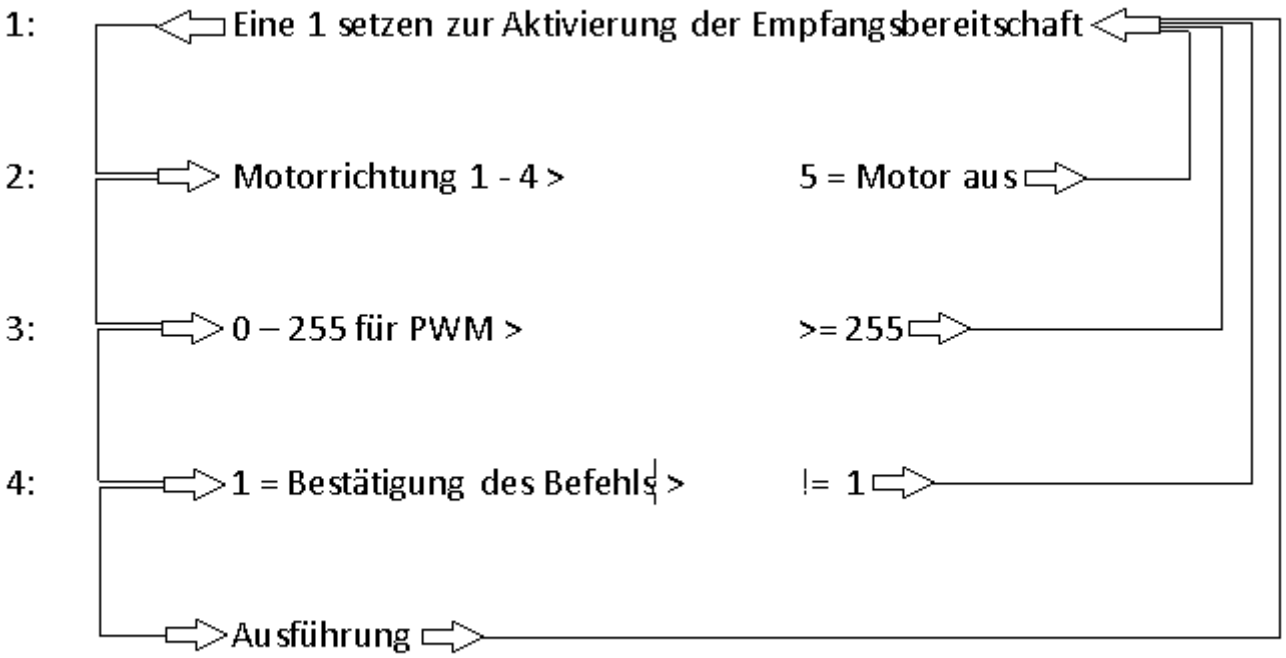
Der Ultraschallsensor hat einen ungefähren Öffnungswinkel von 30°. Die Bauform ist dem oberen Bild zu entnehmen.



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 Echosignal auf dem Echo-Pin zurück. Die Länge des Echosignals wird mit einem Timer gemessen, da die Länge mit der gemessenen Distanz des Sensors variiert.

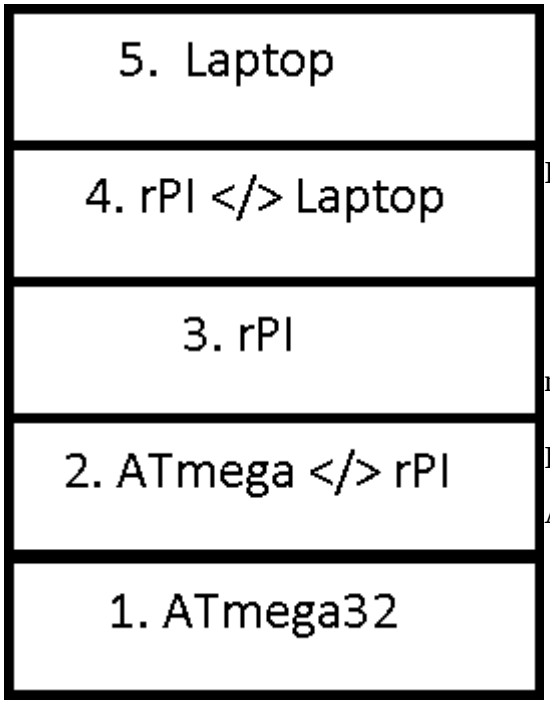
Protokoll Entwicklung

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 Motorenzustand. Daraufhin sendet man zwei mal 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	

Diskussion

Die Bauzeit des Roboters betrug ca. 150 Stunden, die Programmierung ausgeschlossen.
Der Aufbauprozess war relativ flüssig.
Während des Aufbauprozesses traten folgende Probleme auf:

- Hohe thermische Probleme und hoher Stromverbrauch:
  Aufgrund der Menge an Hardwarekomponenten entstanden beim Zusammenbau Platzprobleme, was zur Folge hatte das Abwärme schlecht entweichen konnte. Dazu kam noch das die Schaltregler und Linearregler mit zunehmenden Hardwarekomponenten höhere Lasten zu tragen hatten. Die Folge war das durch den Hitzestau teilweise im Roboter Temperaturen von über 100°C herrschten was zur Zerstörung einiger Bauelemente führte u.a. beim LM-7805. Das Problem konnte größtenteils durch eine Energiemanagement bewusste Programmierung behoben werden. Zusätzlich wurden einige Elemente durch energieeffizientere Komponenten ausgetauscht.

- Hohe Kamerastream Verzögerung:
  Bei geringer Netzwerkbandbreite trat eine sehr hohe Verzögerung des Kamerastreams auf: teilweise bis zu einer Minute. Die Folge war das ein Steuern des Roboters nahe zu unmöglich war. Durch das Deaktivieren des Streamcaches und verringern der FPS konnte das Auftreten der Verzögerung stark verringert werden.

- Erkennen des Abstürzen des Betriebssystem des Raspberry PI bei kurzzeitigem Spannungsabfall:
  Wenn die Akkuleistung zur Neige geht führt dies dazu dass das Betriebssystem des Raspberry PI abstürzt. Das Problem dabei war dass das Mikrocontroller Board seinen letzten Zustand behalten hat und z.B. immer weiter geradeaus gefahren ist. Durch einen externen Interrupt Pin am Mikrocontroller Board kann durch Setzen eines GPIO Pins am Raspberry PI verifiziert werden ob das Betriebssystem noch läuft.

Reflexion

Das Projekt lief sehr gut und die Entwicklung des Roboters ging zügig voran, da schon Erfahrungen bei der Programmierung vorhanden waren. Aber auch bei der Programmierung gab es Probleme: Unter anderem bei der Ultraschallsensor Programmierung entstanden Probleme aufgrund einer relativ komplexen Ansteuerung. Nach längerer Suche im Internet und häufigem Austesten wurde das Problem jedoch gelöst. Im Integrieren des Ultraschallsensors gab es die meisten Probleme auf Grund der unerfahrenen Bearbeitung. 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 Oberflächenbearbeitung darstellen, bei welcher ein Kettenantrieb und eine Abdeckung der Hardware angebracht würden.

Erweiterungsmöglichkeiten

Das Projekt könnte durch eine Erweiterung um eine GUI programmierte Oberfläche einfacher zu steuern sein und auch bezüglich des Designs optimiert werden da zurzeit nur eine reine Konsolensteuerung möglich ist. Unter anderem könnte man eine Personen- und Gestenerkennung implementieren. Der Raspberry PI hätte dafür nicht genügend Leistung, jedoch könnte man den Kamerastream verwenden und diesen an einem leistungsstärkeren Rechner auswerten lassen, um anschließend die Ergebnisse wieder zurück an den Raspberry PI zu senden.

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

(ToDo-Liste)

Stepp 1 Mikrocontroller

Was soll der Mikrocontroller können?

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

Stepp 2 Mikrocontroller > Raspberry PI

Wie soll der Mikrocontroller gesteuert werden?

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

Stepp 3 Raspberry PI Steuerung

Was soll der Raspberry PI können?

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

Stepp 4 Raspberry PI remote Steuerung

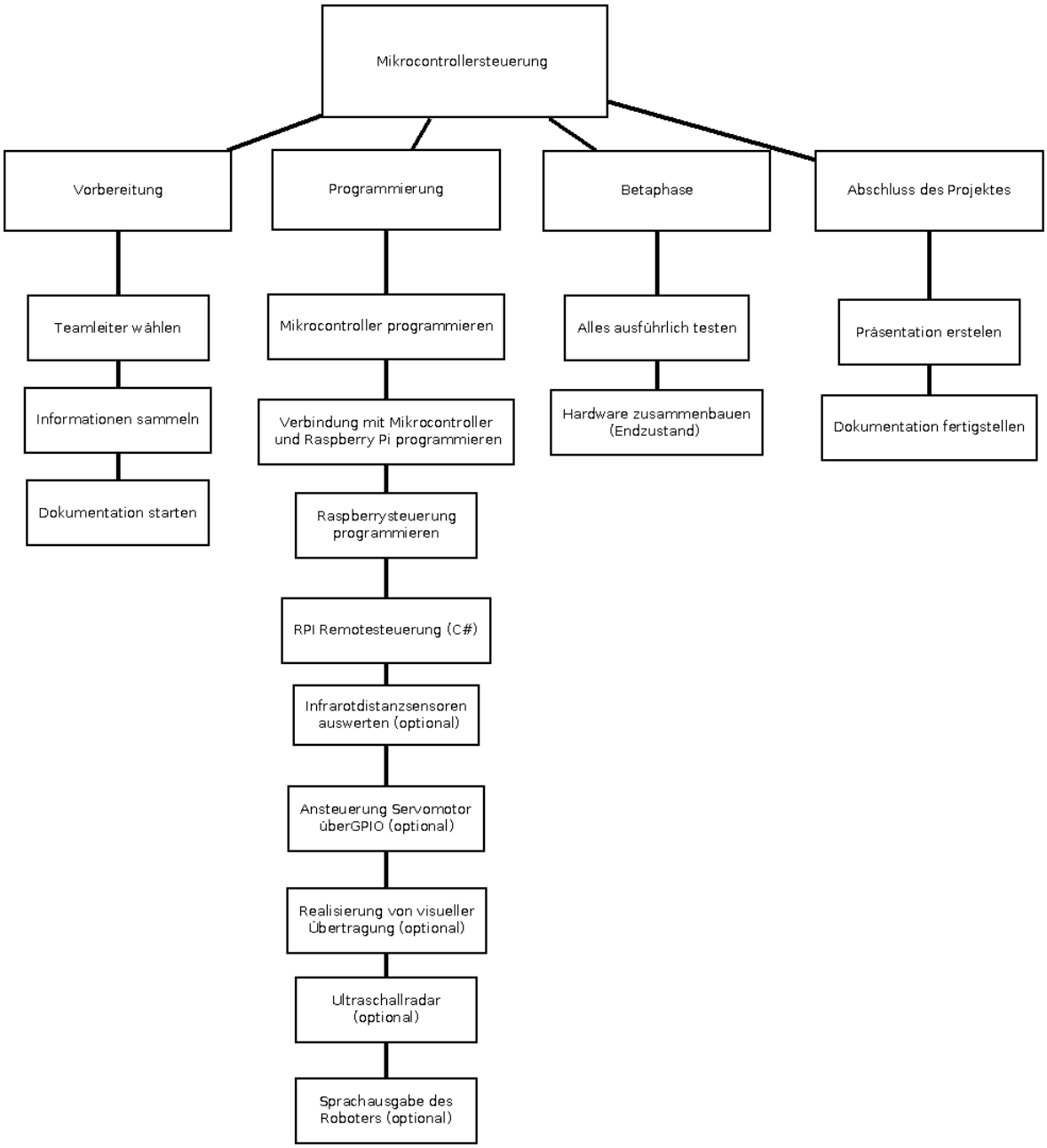
Wie soll der Raspberry gesteuert werden Windows/Linux?

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

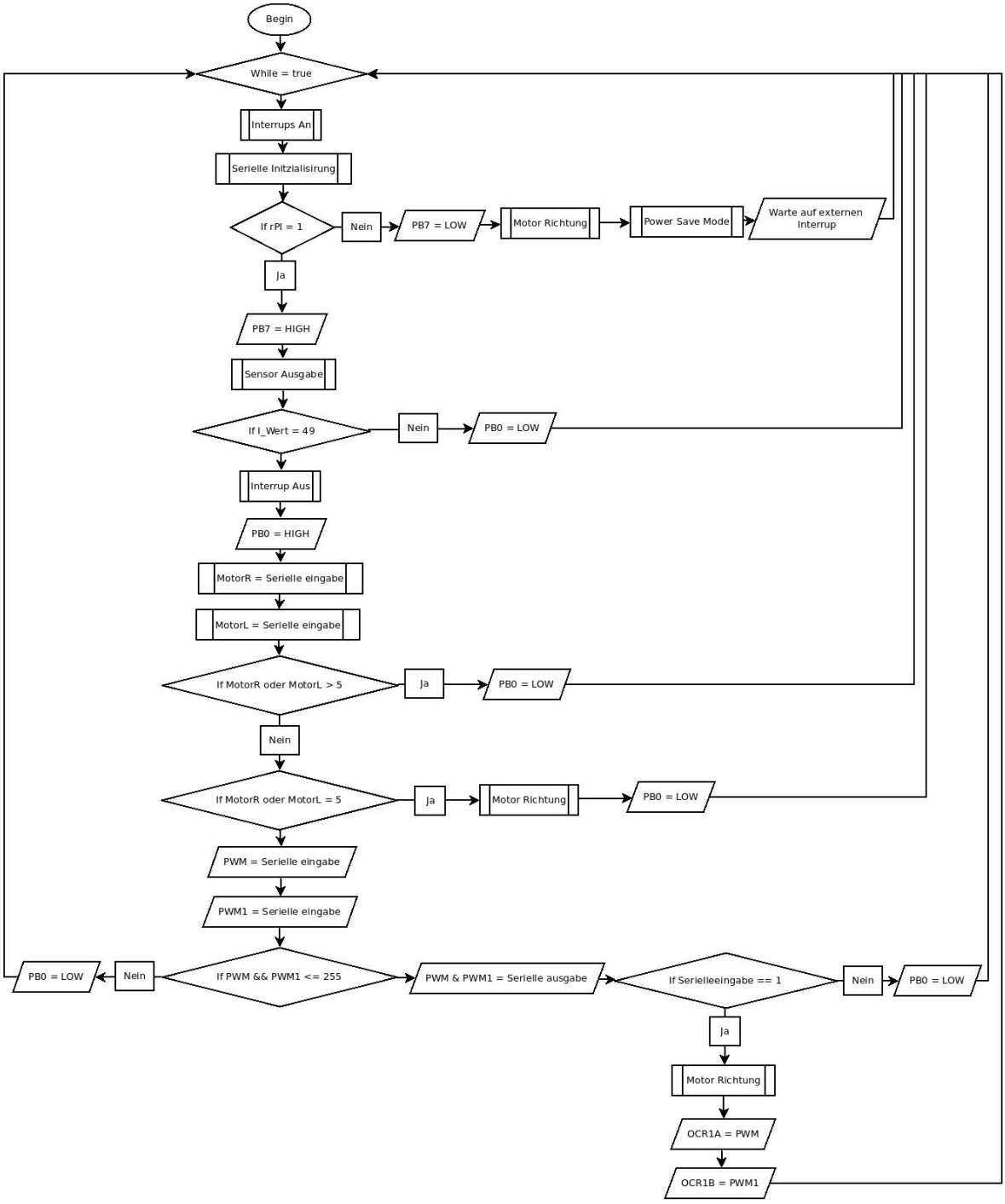
(Vorgangs liste)

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

(Projektstrukturplan)



(Fluss Diagramm)



Quellcode

(ATmega32)

```
/*
 * uC_Serial.c
 *
 * 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                          // Taktfrequentz fest legen
#define BAUDRATE 57200                          // Baudrate fuer die serielleschnittstelle
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 komunikation
starten
int rPI;                                // ISR variable pruefen ob raspberry PI errechbar 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();                                  // Deaktiviren aller interrups
        if (PIND & (1<<PD2))                    // Wenn PD2 == High dann ...
        {
                rPI = 1;                        // rPI == Online
        }else
        {
                rPI = 0;                        // rPI == Offline
        }
        sei();                                  // Aktiviren aller interrups
}


ISR(USART_RXC_vect)     // Interrupt service routine loest aus wenn daten empfangen
werden
```

```
{
        cli();                                          // Deaktiviren aller interrups
        while(bit_is_clear(UCSRA,RXC));   // warten auf Receive Complete
        I_Wert = UDR;
        sei();                                          // Aktiviren aller interrups
}

ISR (TIMER0_OVF_vect) // Interrupt service routine loest aus wenn timer0 Register ueberlaeuft
{
        cli();          // Deaktiviren aller interrups
        zaehler++;
        sei();          // Aktiviren aller interrups
}

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
Steigene 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 Referrers signal der Lichtschranke ausgeben
	uart_send_string("\n\r");		// Zeielnumbruch
	_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 Referrens spannung
	TCCR1A |= (1<<WGM10)|(1<<COM1A1)|(1<<COM1B1);
	TCCR1B |= (1<<WGM12)|(1<<CS12);				// Prescaler 256
	GICR |= (1<<INT0);					// IRS INT0 externer Interrupt aktiviren
	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();				// Aktiviren aller interrups
		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();
        // Deaktiviren aller interrups

                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 geschwindigkein 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 geschwindigkein 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
zeielnumbruch
                        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
zeielnumbruch
                        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 Aktiviren
                        sleep_mode();                                   // uC geht Ideln
                }
    }
}
```

(Raspberry PI)

```
#!/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
```

Rasperry PI Kamerasteuerung

```
#--------------------
# 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!"
```

(Roboter Steuerungs Skript)

```
#!/bin/bash
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
```