



# „TeachRobot08“

## EntwurfsDokumentation

EntwurfsDokumentation.doc

### Inhaltsverzeichnis

<b>1 GROBDESIGN</b> .....	<b>4</b>
<b>1.1 grafische Benutzeroberfläche</b> .....	<b>4</b>
<b>1.2 Steuerung des Roboters</b> .....	<b>4</b>
<b>1.3 Schnittstelle zwischen Steuerung und der Benutzeroberfläche</b> .....	<b>4</b>
<b>1.4 Inpout32.dll</b> .....	<b>4</b>
<b>2 DETAILDESIGN</b> .....	<b>5</b>
<b>2.1 Klassen</b> .....	<b>5</b>
<b>2.2 Klasse Programm</b> .....	<b>6</b>
<b>2.3 Klasse Gui</b> .....	<b>6</b>
2.3.1 Gui.Designer.cs .....	6
2.3.2 Gui.cs .....	6
<b>2.4 Klasse RoboterSteuerung</b> .....	<b>7</b>
<b>2.5 Klasse HardwareSchnittstelle</b> .....	<b>7</b>
<b>2.6 Klasse Positionen</b> .....	<b>8</b>
<b>2.7 Klasse Initialisieren</b> .....	<b>8</b>
<b>2.8 Klasse Motor</b> .....	<b>9</b>
<b>2.9 Klasse Aufzeichnen</b> .....	<b>9</b>

### Tabellenverzeichnis

<b>TABELLE 1: MOTORENKENNZEICHNUNG</b> .....	<b>6</b>
--	----------

### Abbildungsverzeichnis

<b>ABBILDUNG 1: KLASSEN TEACHROBOT08</b> .....	<b>4</b>
<b>ABBILDUNG 2: KLASSENDIAGRAMM TEACHROBOT08</b> .....	<b>5</b>
<b>ABBILDUNG 3: GRAFISCHES USER INTERFACE</b> .....	<b>6</b>
<b>ABBILDUNG 4: HARDWARE-SCHEMA</b> .....	<b>8</b>

## *Änderungsgeschichte*

<u>Datum</u>	<u>Version</u>	<u>Autor</u>	<u>Beschreibung</u>
2008-03-24	1.0	Gion Bärtsch, Christian Landolt, Gerhart Clemend	Dokument erstellt

# 1 Grobdesign

Programmiersprache: C#  
Entwicklungsumgebung: Visual Studio 2005 Professional  
Projektname: TeachRobot08  
Folgende Klassen werden implementiert:

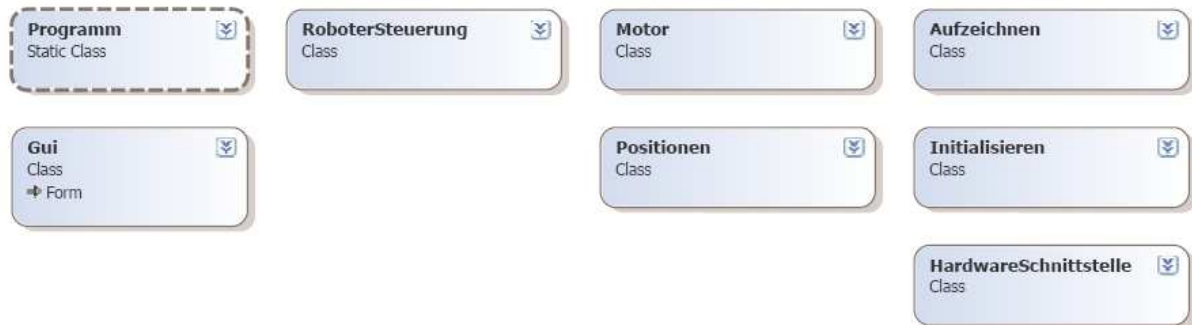


Abbildung 1: Klassen TeachRobot08

## Klassen für die grafische Benutzeroberfläche:

Gui und Programm

## Klassen für die Steuerung des Roboters:

Motor, HardwareSchnittstelle, Initialisieren, Positionen, Aufzeichnen

## Schnittstelle zwischen Benutzeroberfläche und Steuerung:

RoboterSteuerung

## Schnittstelle zwischen Steuerung und Hardware

Inpout32.dll

### 1.1 grafische Benutzeroberfläche

Die grafische Benutzeroberfläche besteht im Wesentlichen aus der Klasse Gui. Diese Klasse nimmt die Benutzereingaben entgegen, verhindert Fehlmanipulationen, leitet die Eingaben weiter und meldet dem Benutzer Ereignisse zurück. Die grafische Benutzeroberfläche soll selbsterklärend sein und den Benutzer soweit informieren, dass er weiss, welche Schritte ausgeführt werden und wann er die nächste Eingabe tätigen kann.

### 1.2 Steuerung des Roboters

Dieser Teil dient dazu, die Befehle für den Roboter aufzuarbeiten. Hier werden die richtigen Motoren ausgewählt und über die Klasse HardwareSchnittstelle werden die Pins der Parallelschnittstelle angesprochen oder abgefragt.

Für das speichern oder lesen der Textdatei, in der Bewegungsabläufe gespeichert werden, ist die Klasse Aufzeichnen verantwortlich.

Mit der Klasse Initialisieren wird der Roboter in eine Grundstellung gefahren.

### 1.3 Schnittstelle zwischen Steuerung und der Benutzeroberfläche

Die Klasse RoboterSteuerung dient als alleinige Schnittstelle zwischen Benutzereingabe und dem Roboter. Hier werden die Benutzereingaben den richtigen Objektmethoden zugeordnet. Variablen die gemeinsam genutzt werden, sind hier abgelegt. Events werden in dieser Klasse weitergereicht ans Gui.

### 1.4 Inpout32.dll

Bibliotheksdatei für Windows NT Betriebssysteme. Sie enthält einen Treiber für den Parallelport und stellt zwei Funktionen zur Verfügung: Inp32 und Out32.

## 2 Detaildesign

### 2.1 Klassen



Abbildung 2: Klassendiagramm TeachRobot08

### 2.2 Klasse Programm

Die Klasse Programm dient als Einstiegspunkt ins Programm. Das Gui wird von hier aus gestartet.

## 2.3 Klasse Gui

Das Gui wird in 2 partielle Klassen unterteilt, Gui.Designer.cs und Gui.cs.

### 2.3.1 Gui.Designer.cs

Dieser Teil enthält das Design der Steuerelemente und erzeugt diese.

Bezeichnungen der Variablen:

Checkboxen = ...Chbx  
 RadioButton = ...Rbtn  
 Buttons = ...Btn  
 Textbox = TextBox  
 Label = Label

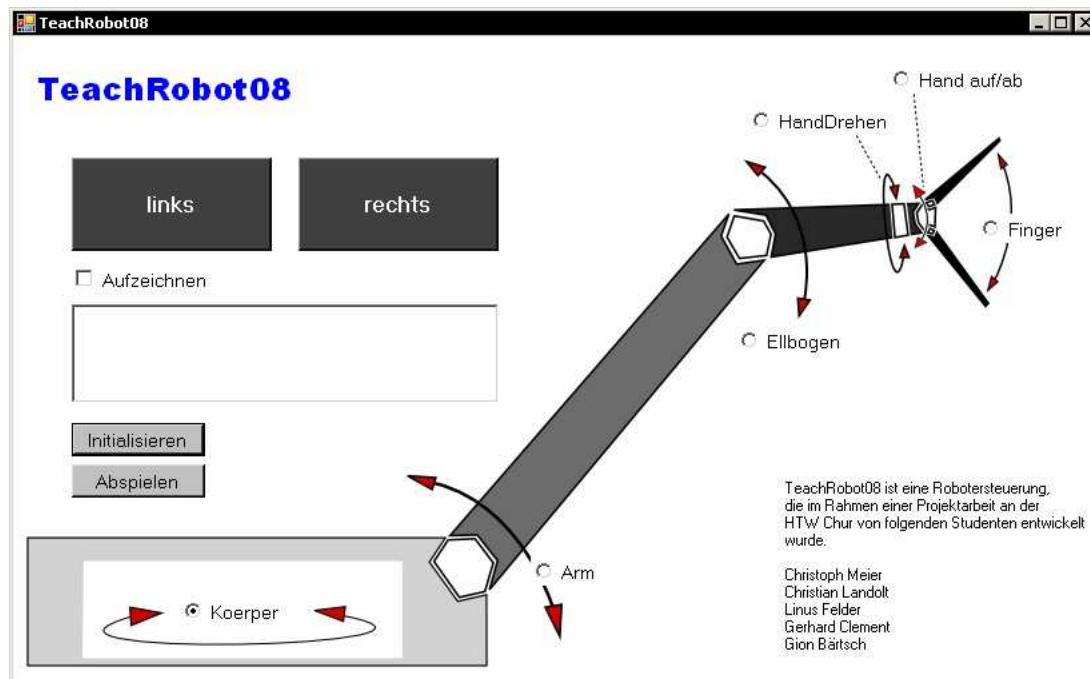


Abbildung 3: Grafisches User Interface

### 2.3.2 Gui.cs

Darin enthalten sind die Events und Methoden die durch die Steuerelemente ausgelöst werden und Events die von der Steuerung des Roboters ausgelöst werden.

Die Auswahl der Motoren erfolgt über Radio Buttons. Gekennzeichnet sind die Radio Buttons mit Textbeschreibungen, an die RoboterSteuerung werden jedoch Integerwerte wie folgt übergeben:

Bezeichnung	Wert der übergeben wird	Bewegungsrichtungen	Richtungswert
Finger	1	auf / zu	0 / 1
Hand drehen	2	links / rechts	0 / 1
Hand auf/ab	3	auf / ab	1 / 0
Ellbogen	4	auf / ab	0 / 1
Arm	5	auf / ab	1 / 0
Koerper	6	links / rechts	1 / 0

Tabelle 1: Motorenkennzeichnung

Alle Methoden die aufgerufen werden und mehr als eine Anweisung beinhalten, werden mit einem neuen Thread gestartet, der danach wieder beendet wird. Die separaten Threads sind notwendig, weil sonst keine Rückmeldungen während der Verarbeitung auf die Textbox gemacht werden können. Weitere Threads, wie parallele Abläufe auf dem Roboter dürfen nicht gestartet werden. Dies wird mit der Variable ThreadAktiv überwacht. Somit werden folgende Fehlmanipulationen verhindert:

- Steuern während dem Initialisieren
- Steuern während dem Abspielen
- Initialisieren während dem Abspielen
- Abspielen während dem Initialisieren

Während dem Aufzeichnen darf nicht initialisiert oder abgespielt werden. Dies wird verhindert, indem beim Drücken der Buttons - Initialisieren oder Abspielen - überprüft wird, ob die Checkbox aufzeichnen aktiviert ist. Ist dies der Fall, wird der Benutzer mit einer Messagebox benachrichtigt.

### **Events von der Steuerung des Roboters**

ThreadEndeEvent: Der letzte Thread der gestartet wurde, wird beendet und es wird eine Textmeldung ausgegeben.

TextEvent: Eine Textmeldung wird ausgegeben.

## **2.4 Klasse RoboterSteuerung**

Alle Events die vom Gui ausgelöst werden, rufen Methoden dieser Klasse auf.

Alle Events die von der Steuerung ausgelöst werden, werden hier an die Klasse Gui weitergereicht.

Die Variable Text enthält die Textmeldungen die auf dem Gui ausgegeben werden. Diese Variable wird an die Methoden der Steuerung by Reference weitergegeben. So wird ein Zeiger auf Text weitergegeben und der Wert kann aus den Klassen der Steuerung verändert werden. Wird ein ThreadEndeEvent oder TextEvent ausgelöst erscheint eine Textmeldung in der Textbox.

## **2.5 Klasse HardwareSchnittstelle**

Die Klasse HardwareSchnittstelle, ist die Schnittstelle zwischen der Software und der Parallel-Schnittstelle am PC.

Diese Klasse beinhaltet zwei Funktionen.

### **Output**

Die erste Funktion gibt Signale auf die 8 Daten Pins der Parallel-Schnittstelle und steuert so die Motoren an.

### **Input**

Die zweite Funktion macht es uns möglich, das Pulssignal der Motoren einzulesen. Die Motoren haben Inkrementgeber, bei jeder Umdrehung kommen vier Pulse zurück.

Der Roboter hat 6 Motoren. Um alle 6 Motoren für Links und Rechts anzusteuern brauchen wir 12 verschiedene Signale. Da es an der Parallel-Schnittstelle nur 8 Datenausgänge gibt, wurde eine Multiplexer Schaltung gebaut. Die Multiplexer Schaltung ermöglicht es über 4 Adress-Leitungen, 16 verschiedene Signale auszugeben. Wovon 12 für die Robotersteuerung verwendet werden.

Das gleiche Prinzip wurde für das einlesen verwendet. Es gibt 6 Inkrementgeber und die zurückkommenden Pulssignale werden über den gleichen Pin eingelesen. Die Adressierung dieses Multiplexers geschieht mit den gleichen Signalen, wie für das Ansteuern der Motoren. Da nur 6 Signale zurückkommen, wird von 1-6 adressiert.

Verwendete Pins an der Parallel-Schnittstelle:

Ausgeben der Adressleitungen zum Ansteuern der Motoren: D0-D3 Pin2 – Pin4  
Einlesen der Pulssignale vom Inkrementgeber: Error Pin15

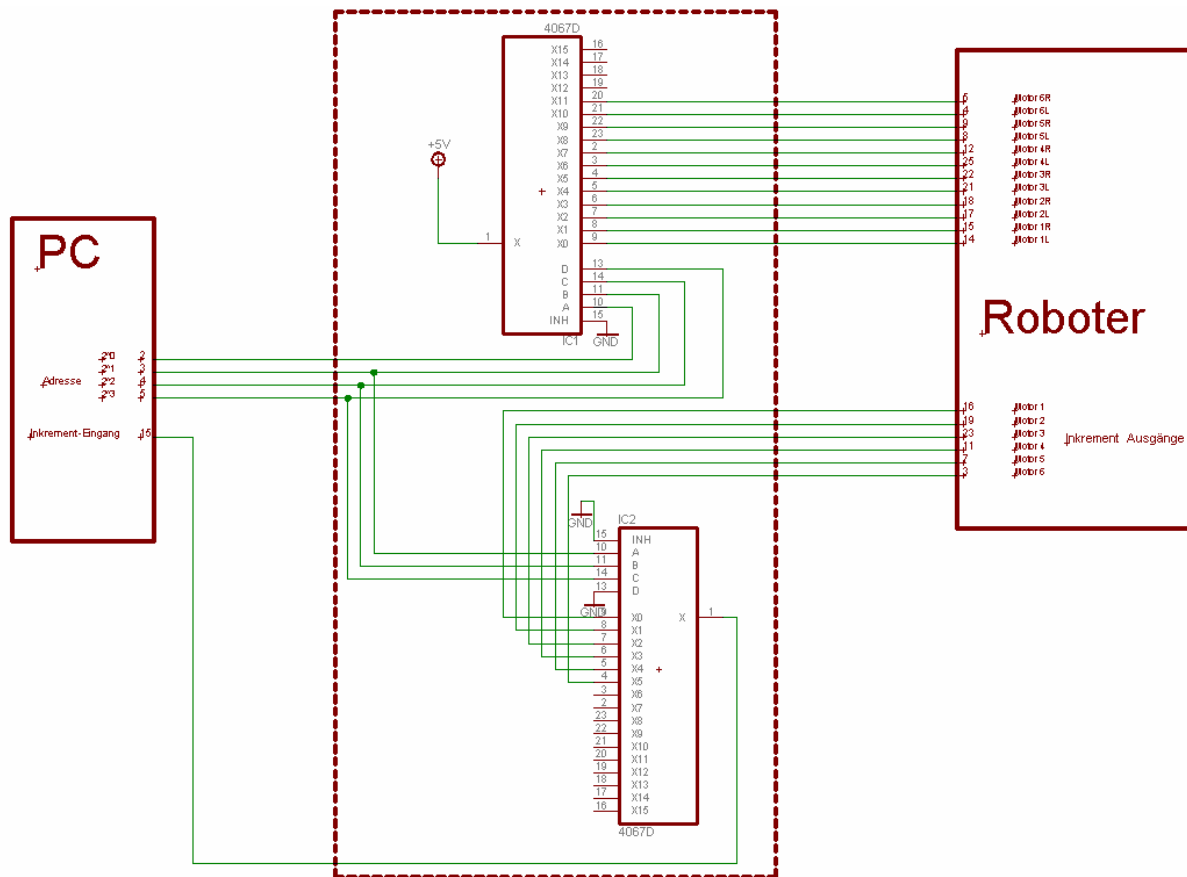


Abbildung 4: Hardware-Schema

## 2.6 Klasse Positionen

Damit die Positionen jederzeit identifiziert werden können, braucht es Positionsvariablen. Jeder Motor hat eine eigene Variabel in der seine jeweilige Absolutposition gespeichert ist.

Die Klasse beinhaltet zwei Funktionen. Eine mit der die Positionsvariablen überschrieben werden können und eine mit der die aktuelle Position der Motoren ausgelesen werden kann.

## 2.7 Klasse Initialisieren

Wenn die Software gestartet wird, muss der Roboter in eine Grundposition fahren. Das geschieht mit der Klasse Initialisieren und ihren Funktionen.

Beim Initialisieren wird jeder Motor der Reihe nach in seine Grundposition gefahren. Die Grundposition der Motoren ist ca. in der Mitte ihres Bewegungsspielraums.

Beim Initialisieren wird zuerst der erste Motor bis an einen Endschalter gefahren. Sobald er am Endschalter angekommen ist, wechselt er die Richtung und fährt zur Mitte seines Bewegungsspielraums zurück. Derselbe Ablauf wird für den zweiten, dritten... bis sechsten Motor durchlaufen.

Wenn die Motoren in der Grundposition sind, werden die Positionsvariablen der Motoren auf ihre Grundposition gesetzt.

## **2.8 Klasse Motor**

Mit der Klasse Motor werden die Motoren angesteuert und die Pulse der Inkrementgeber zurückgelesen.

Für das Fahren der Motoren wird unterschieden, ob mittels Tastendruck auf dem Gui der Roboter bewegt wird, oder ob ein aufgezeichnetes File abgespielt wird. Für beide Varianten gibt es eine separate Methode.

Sobald die Motoren angesteuert sind, müssen die zurückkommenden Pulse von den Inkrementgeber erfasst und gezählt werden.

Für das gibt es auch wieder zwei Methoden. Eine für das zählen wenn eine Aufnahme abgespielt wird und eine wenn über die Gui Eingabe gefahren wird.

Wenn ein Textfile abgespielt wird, ist diese Funktion solange aktiv, bis die Zielposition erreicht ist.

Beim Fahren über das Gui wird die Funktion unterbrochen wenn der Taster losgelassen wird.

Am Schluss der Funktionen werden die Positionsvariablen auf ihren aktuellen Stand gebracht.

## **2.9 Klasse Aufzeichnen**

Diese Klasse ist für die Aufzeichnung der Bewegungsabläufe zuständig. Die Datei enthält zwei Hauptmethoden. Zum einen werden die Daten gespeichert, zum anderen werden die gespeicherten Daten ausgelesen und zurückgeliefert. Die Daten werden in einer Textdatei mit dem Namen „Aufzeichnung.txt“ im gleichen Verzeichnis wie das ausgeführte Programm abgelegt. Den entsprechenden Pfad ermittelt eine Methode in der Klasse „Aufzeichnen“.

Die Daten werden als Integer-Array an die Methode „Speichern“ übermittelt. In diesem Array sind die Anzahl Schritte zum jeweiligen Motor gespeichert. Wird das Textfile ausgelesen, wird ein entsprechendes Array zurückgeliefert.