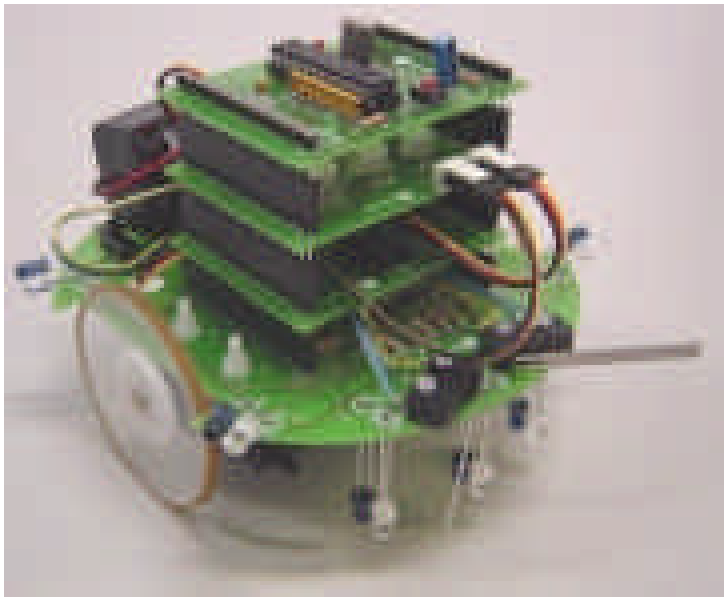
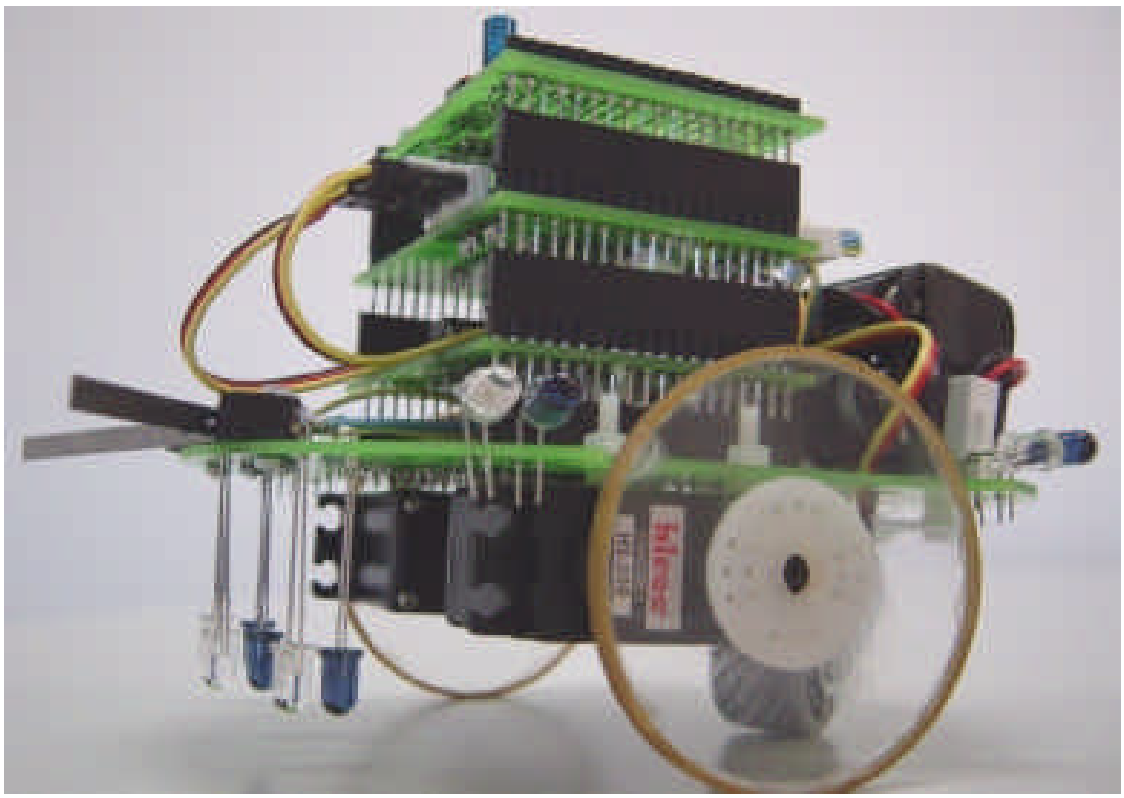


# Circular Robot II



April 2003

# Id-Mind – Circular Robot II Technisches Handbuch

## Vorwort

Gratulation! Sie haben soeben ein Set mit IdMind Qualität erworben!

Bei IdMind sehen wir die Informationstechnik als einen effektiven Lernweg. Diese Fähigkeiten werden durch praktische Tätigkeiten verbessert. Wir glauben dass „ **learning by doing"** ein effektiver Weg ist um Wissen zu vermitteln.

Das breit gefächerte Spektrum der „Robotik" gibt uns ein ideales Mittel für das Verständnis der verschiedenen Aspekte von Wissenschaft und Technologie, speziell Elektrotechnik, Mechanik und Computer Technik.

Das „Circular Robot" Set basiert auf einer runden Scheibe mit den Dimensionen einer CD. Sein Hauptschwerpunkt liegt auf der Modularität: der Roboter ist kompatibel mit verschiedenen Sensortypen. Die Ausführung von Programmen wird möglich durch die Programmierung eines Mikrocontroller´s. Der Roboter wird über den PC programmiert, durch ein grafisches Programmiersoftware, wo der Programmierer nur einige Felder für den gewählten Roboter einsetzen und miteinander verbinden muss. Die Möglichkeiten des Roboters wird nur durch die Phantasie des Programmierers begrenzt.

Dieses Handbuch übermittelt die grundlegenden Konstruktionsabschnitte des Roboters. Für weitere Fragen stehen wir ihnen im Internet unter

e-mail: info@idmind.pt
http://www.idmind.pt

Der Roboter wird in Deutschland durch die Firma

Krause Robotik
Nordstr. 29
06800 Jeßnitz
mail: info@krause-robotik.de
www.krause-robotik.de

vertrieben.

Inhaltsverzeichnis


1. Einleitung
   1.1 Funktionsstruktur des Roboters
2. Roboter Konstruktion Einige Fakten zum Merken
   2.1 Plattform Konstruktion
   2.2 Die Motoren heraussuchen
      2.2.1 Kraft und Geschwindigkeitsabgleich
      2.2.2 DC Motoren Geschwindigkeitskontrolle
      2.2.3 Richtungssteuerung
3. beiliegende Komponenten
   3.1 Allgemeine Beschreibung
   3.2 Funktionsweise
4. Teilezusammenbau
   4.1 Grundlagen des Lötens
   4.2 Einzelteile
      4.2.1 Widerstände
      4.2.2 Widerstandsnetzwerk
      4.2.3 Kondensatoren
      4.2.4 Dioden und LED´s
      4.2.5 Integrierte Schaltkreise
   4.3 Konstruktion der Roboterboard´s
      4.3.1 Konstruktion des Controllerboards ID-PIC002
      4.3.2 Konstruktion des Digitalboards ID-DS002
      4.3.3 Konstruktion der Analogschnittstelle ID-AS002
      4.3.4 Konstruktion des Sensorboards ID-BASEIR002
      4.3.5 Verbindung der verschiedenen Boards
      4.3.6 Herstellung der Programmierschnittstelle ID-PROG002
5. Mikrokontroller
   5.1 Ein- und Ausgangskonfiguration
   5.2 Analoge Eingänge
      5.2.1 ADCON0
      5.2.2 ADCON1
   5.3 PWM (Impulsweitenmodulations-) Ausgänge
   5.4 PWM Periodendauer
      5.4.1 PWM Tastverhältnis
      5.4.2 Maximale Bitanzahl
      5.4.3 Schritte für die Initialisierung der PWM
   5.5 Verwendung von Tabellen
   5.6 Interruptionen (Unterbrechungen)
      5.6.1 INTERCON Datensatz
      5.6.2 PIE1 Datensatz
      5.6.3 PIR1 Datensatz
   5.7 Struktur eines Programms für die PI-Regelung (PIC)
6. Programmierung
   6.1 Programmierung der Steuerung des Infrarot-Board 's
   6.2 Programm zum Steuern der zwei Servomotoren
7. Experimentierbeispiel
   7.1 Fehlermeldungen

Anhang A Schaltpläne
   A1) Steuereinheiten (ID-PIC002, ID-DS002e, ID-AS002)
   A2) Infrarot-Board

Anhang B Bauteile und Boards
   B1) Steuerungsboard ID-PIC002
   B2) Digitales Schnittstellen-Board ID-DS002
   B3) Analoges Schnittstellen-Board ID-AS002
   B4) Programmierboard ID-PROG002
   B5) Infrarot-Sensoren-Board ID-BASEIR002

Anhang C – Änderungen am Roboter, um ihm das Fußballspielen zu ermöglichen

# 1. Einleitung

Roboter sind in unseren Gedanken gut durchgedacht. Vor kurzem dachte jeder, dass Roboter ein großes komplexes System sind, sehr teuer, eine Mischung aus elektrischen Drähten und komplexen Schaltkreisen.

Gerade in den letzten Jahren: die erreichten Errungenschaften im Bereich der Mikrocontroller, besonders die Reduzierung der Größe und Kosten - im Gegenzug: die Steigerung des Einsatzpotentials, der Bedienung, auf einem einfachen Weg, einen Satz von Robotersystemen zu entwickeln, die im Stande sind, einige Aufgaben autonom zu übernehmen.

Es muss auch bemerkt werden, dass die Bauweise das Roboters nicht unbedingt seiner Programmierung entspricht. Das Konzept des Roboters basiert auf Grundkenntnissen der Mechanik, Elektrotechnik und Informatik. Dieser komplexe Prozess bietet uns eine extreme Herausforderung.

In diesem Handbuch sind alle wichtigen Aspekte für eine kleine Roboterkonstruktion, wobei es durch ein elektronisches Set unterstützt wird, welches durch den Entwickler dieses Projektes entworfen wurde. Der Benutzer wird erkennen, dass während des Konstruktionsprozesses, die Vorstellung der verschiedenen Schritte zur Roboterentwicklung, angefangen bei der Komponentenauswahl bis hin zur Mikrocontroller-Programmierung ineinander greifen.

## 1.1 Funktionsstruktur des Roboters

Der Zweck der Funktionsstruktur des Roboters ist die Antwort auf die folgenden 3 Hauptprobleme:

*Wo bin ich? Was soll ich machen? Wie kann ich das tun?*

Wenn sich ein Roboter in einer spezifischen Umgebung befindet, benutzt er seine Sensoren, um seine Position zu bestimmen und seine Ziele zu finden. Er benutzt seinen Antrieb, um sich fortzubewegen, oder um Objekte zu manipulieren. Am Ende benutzt er seinen Mikrocontroller, um die Zusammenarbeit zwischen Sensoren und Aktoren zu gewährleisten.

Bild 1.1 zeigt die Arbeit des Mikrocontrollers.

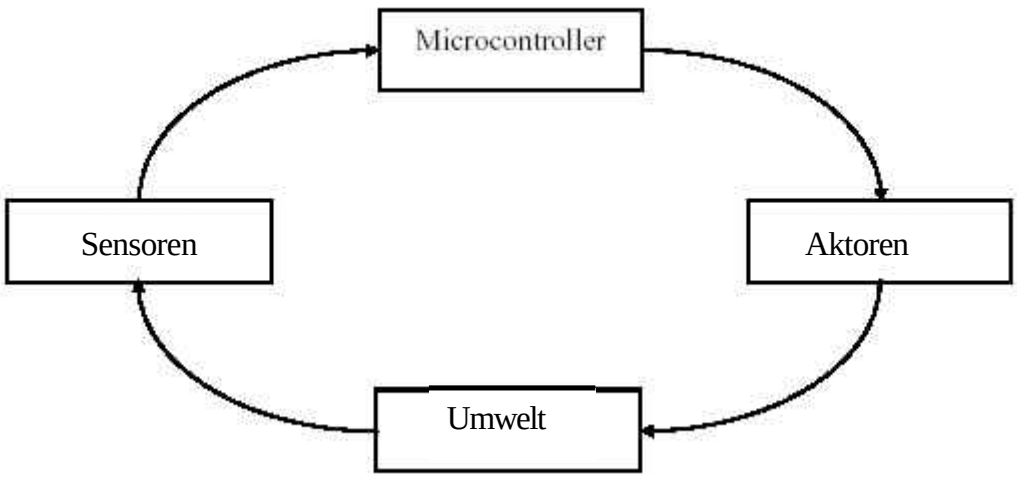



Bild 1.1 – Roboter – Funktionszyklus

# 2. Roboter Konstruktion - Einige Fakten zum Merken

Da das Handbuch alle Details des Entwurfs von elektronischen Komponenten und die Roboterprogrammierung bündelt, wollen wir in diesem Abschnitt einige Fakten, die man bei der Konstruktion eines Roboters beachten sollte, genauer betrachten.

## 2.1 Plattform Konstruktion

Die Plattform kann verschiede Gestalten annehmen, aber in seinem Entwurf sollte einer der folgenden Punkte erfüllt sein:

- Einfachheit – Minimieren Sie die Anzahl der beweglichen Teile und die Komplexität des Roboters
- Robustheit – Stabilität gegen Erschütterungen
- Modularität – der Roboter sollte aus einem Satz von verbundenen Modulen bestehen, so dass, wenn ein Modul entfernt werden muß, nicht der ganze Roboter auseinandergenommen wird.

Verschiedene Materialien können dazu verwendet werden. Am meisten wird PVC verwendet, aber auch rostfreier Stahl, Aluminium, Sperrholz und Acryl.

Bild 2.1 zeigt eine typische Struktur eines Roboters. Sie beinhaltet zwei unabhängige Motoren (gleiche Zugleistung) und ein Stützrad um die Balance zu halten.

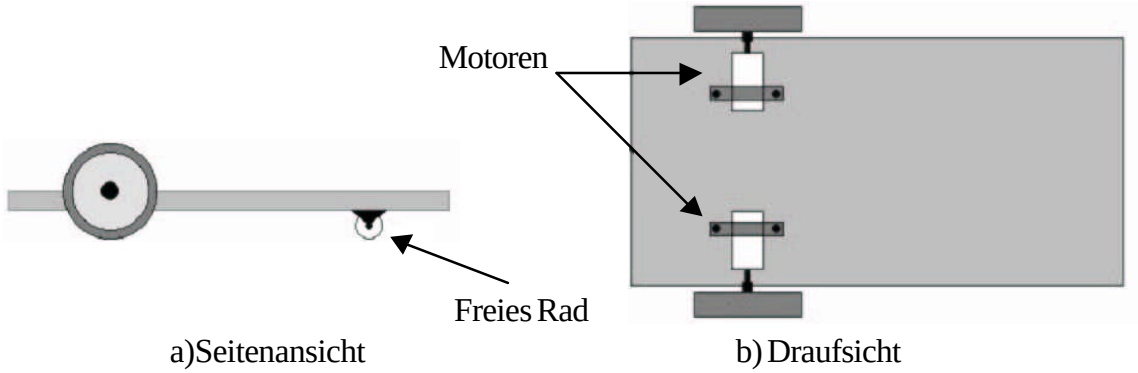



Bild 2.1 Roboter mit zwei Motoren

## 2.2 Die Motoren heraussuchen

Um einen angemessenen Motor für eine bestimmte Roboteraufgabe auszuwählen sollten einige Fakten beachtet werden.

- Größe und Form
- Gewicht
- Verfügbare Leistung
- Höchstgeschwindigkeit
- Maximaler Strom/Leistung

Bei der großen Auswahl von Motoren wird sich einer auf dem Markt finden – Gleich-/Wechselstrom, Schrittmotoren – die Gleichstrommotoren sind auch als DC-Motoren (Direct-Current) bekannt und sind am meisten verbreitet, wegen ihrem besseren Größen/Leistungsverhältnisses und ihrer Mannigfaltigkeit.

### 2.2.1 Kraft und Geschwindigkeitsabgleich

Normalerweise ist ein Motor nicht dafür gebaut ein Rad direkt zu befestigen, manchmal wegen zu hohen Geschwindigkeiten, oder andermal wegen zu wenig Kraft die der Motor bringt. Um seinen gewünschten Motor zu erhalten, muß man die vorhandenen Motoren seinen Wünschen anpassen. Dies geschieht meistens mit Getrieben, um die Drehzahl zu verringern und die Kraft zu erhöhen.

### 2.2.2 DC Motoren Geschwindigkeitskontrolle

Die Motorgeschwindigkeit kann durch die Eingangsspannung (Strom) geregelt werden. Um zu verhindern, dass die Bauteile überhitzen, verwendet man statt einem kontinuierlichen Signal eine PWM (Pulswei ten Modulation). Das sind die Längen der Impulse, die der Controller an den Motor sendet, welche für die Geschwindigkeit verantwortlich sind. Diese verhält sich proportional zu der Fläche unterhalb des positiven Bereiches einer Periode (Bild 2.2).

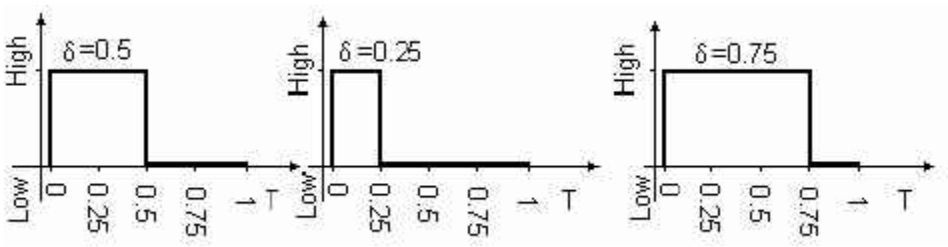



Bild 2.2 – PWM Signal

Da die meisten Mikrocontroller diesen Signalstandard besitzen, ist es sehr einfach diesen auch zu nutzen. Andererseits ist der thermische Verlust in den Bauteilen auch reduziert, da nicht immer ein Strom fließt.

### 2.2.3 Richtungssteuerung

Die Umkehrung der Motordrehrichtung kann durch die Umpolung der Motoren geschehen. Die klassische Lösung dafür ist die Benutzung einer H-Brücke wie sie in Bild 2.3 gezeigt wird. Das PWM Signal könnte am „Enable“ Anschluß angelegt werden.

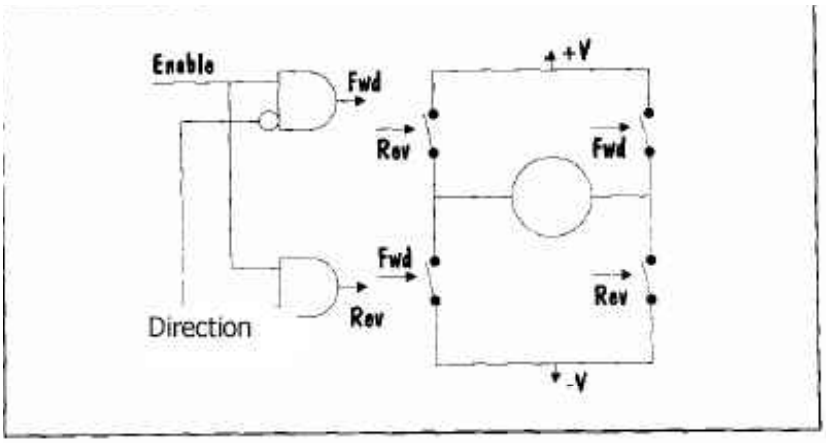



Bild 2.3 - H-Brücke

# 3. beiliegende Komponenten

Der „Circular Robot“ Satz enthält folgende Bauteile:

- Controllerboard mit PIC (ID-PIC002)
- Digitalinterface (ID-DS002)
- Analoginterface (ID-AS002)
- Programmierboard (ID-PROG002)
- Board als Basis mit Schnittstelle für IR-Sensoren (ID-BASEIR002)
- 2 Servomotoren HITEC HS300
- RS232 Anschlußkabel (9-Pol-Belegung)
- Andere kleine Komponenten

## 3.1 Allgemeine Beschreibung

Dieser Satz beinhaltet alle notwendigen Bauteile, um einen kleinen Roboter zu bauen.
Es wurde so konzipiert, dass er leicht zusammenzubauen und leicht zu programmieren ist. So kann jeder, der etwas Interesse an Robotern und Automatisierung hat, diesen Roboter zusammenbauen.

Den Roboter zu konstruieren wird bei jedem das Verständnis für elektronische Bauteile verbessern.

1. Identifikation und Bau von elektrischen Schaltkreisen
2. Verständnis der verschiedenen Teile
    - Sensoren
    - Aktoren
    - Mikrocontroller
3. Entwicklung eines eigenen Programms um den Roboter bestimmte Aufgaben lösen zu lassen.

3.1.1.1 Sensoren

Dies beinhaltet 7 Paare von IR-Sendern bzw. Empfängern, die es möglich machen z.B. eine Linie auf dem Boden zu erkennen oder sogar ein Hindernis. Das beinhaltet auch zwei Taster, welche Kollisionen registrieren können.

3.1.1.2 Aktoren

Ein Paar modifizierte Servomotoren.

3.1.1.3 Mikrocontroller

Der Mikrocontroller ist das „Gehirn“ des Roboters. Dort werden die Informationen, die von den Sensoren kommen, verarbeitet und in Bewegungsbefehle umgewandelt und an die Motoren weitergeleitet.

## 3.2 Funktionsweise

Das ID-PIC002 Board ist eine gedruckte Leiterplatte, wo der PIC16F876 angebracht wird. Dieses Board ermöglicht den Anschluß von Sensoren und Aktoren. Dafür werden zweiseitige Anschlußschienen verwendet um alle Ein/Ausgangspins zu nutzen.

In dem Set sind auch zwei Board´s die den Zugriff auf die Ein/Ausgänge des Mikrocontrollers ermöglichen. Einer von ihnen ist das Digaleninterface (ID-DS002) und das andere das Analoginterface (ID-AS002). Bei dem

letzteren ist es möglich, nur einen analoge Eingang zu nutzen, um eine hohe Zahl von gebündelten Eingängen auszulesen.

In dem Set ist auch ein Infrarotmodul (ID -BASEIR002), um es dem Roboter möglich zu machen, einer Linie (Schwarz auf weißem Hintergrund) zu folgen, oder um Hindernisse zu erkennen. Dieses Modul besitzt 7 Paar von Infrarotsensoren und ist auch die Basis des Roboters.

Da der Mikrocontroller nicht alle Sensoren gleichzeitig abfragen kann, wird immer nur einer abgefragt. Diese Methode spart außerdem Energie.

Das Grundfunktionsprinzip ist sehr einfach. Als Sender wird einfach nur ein 100 Ohm Widerstand mit einer Infrarotdiode genutzt. Auf der Empfängerseite ist es ausreichend zwischen dem Emitter des Fototransistors und Masse einen Widerstand zu legen während der Kollektor des Transistors an + 5 V gelegt wird. Am Emitter kann das Signal abgegriffen werden.

Der Anschluß der analogen Sensoren ist sehr einfach. Es reicht den Sensorausgang mit dem verfügbaren Eingangssignal des Roboters zu verbinden, konkret am Analoginterface ID-AS002. Aus diesem Board gibt es 4 analoge Eingänge mit den Kanälen Eins bis Vier am Port A. Diese sind verfügbar an den Stellen PA1, PA2, PA3 und PA5. Der Anschluß PA4 ist ein digitaler Ein/Ausgang.

Das Digitalinterface ID-DS002 bietet 11 Ein/Ausgänge, welche für die Erweiterung von Sensoren und Aktoren genutzt werden können. In diesem Satz sind 2 Taster vorhanden, welche in diesen Anschlüssen genutzt werden können. Als Aktoren haben wir zwei umgebaute Servomotoren verwendet, welche mit dem PWM-Signal des Mikrocontrollers angetrieben werden. Die Verwendung der Servomotoren ist ein Vorteil, da an ihrer Hauptantriebswelle schon der Abtrieb befestigt wurde und der Stromkreis an die H -Brücke angeschlossen wurde.
Bild 3.1 zeigt den weiteren Funktionsablauf.

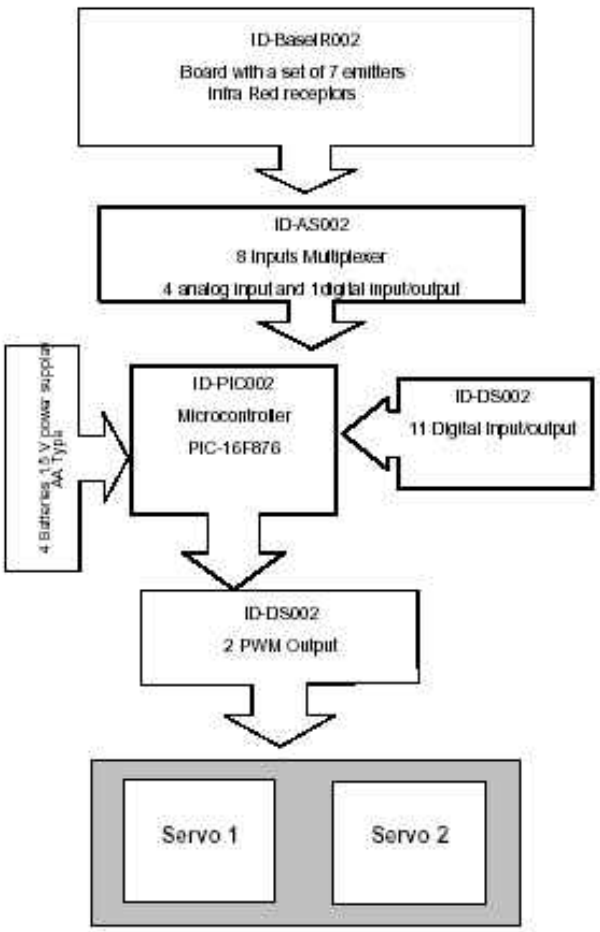


Bild 3.1 Hardware Architektur

# 4. Zusammenbau der Teile

## 4.1 Grundlagen des Lötens

Der erfolgreiche Zusammenbau des Satzes hängt hauptsächlich vom Verlöten der Komponenten auf den aufgedruckten Leiterbahnen ab. (Bild 4.1) Die Erfüllung einiger Hauptregeln in diesem Prozeß werden wie folgt notwendig:

- Nach Möglichkeit sollte ein Lötkolben mit feiner Lötspitze und nicht mehr als 25 Watt Leistung genutzt werden. Diese Lötkolbenart findet man in Elektronik-Spezialgeschäften.
- Während der Benutzung ist die Lötspitze sauber zu halten, am Besten mit einem nassen Schwamm.
- Es sollte nie eine rauhe oder verschmutzte Lötspitze benutzt werden. Ein Austausch ist zu bevorzugen.
- Es sollte 0,7 mm starkes Lötzinn verwendet werden.
- Vor dem Löten sind die Pin´s der Komponenten mit feinem Sandpapier zu reinigen.
- Halten Sie den Lötkolben, so dass die Spitze rückseitig der Komponente befindet und füge das Lötzinn hinzu. (Bild 4.1) Dieser Vorgang sollte nicht länger als 5 Sekunden andauern. (Das Nichtbefolgen dieser Regel könnte zu Beschädigung des Bauteils führen)
- Nach Abschluß des Lötvorgangs sollte die Lötoberfläche ein metallisches Glänzen besitzen. Dies ist der perfekte Moment, die überstehenden Enden der Komponenten zu entfernen.
- Wenn Sie Lötzinn entfernen müssen, nutzen Sie einen Lötzinnentferner.
- Während des Verlötens einer empfindlichen Komponente sollte man sie mit einer Zange festhalten, um die Wärme abzuführen. Anderenfalls könnte das Bauteil Schaden nehmen.
- Es sollte nie zu viel Lötzinn verarbeitet werden, um einen Kurzschluß zwischen den Leiterbahnen zu verhindern. (Bild 4.2)

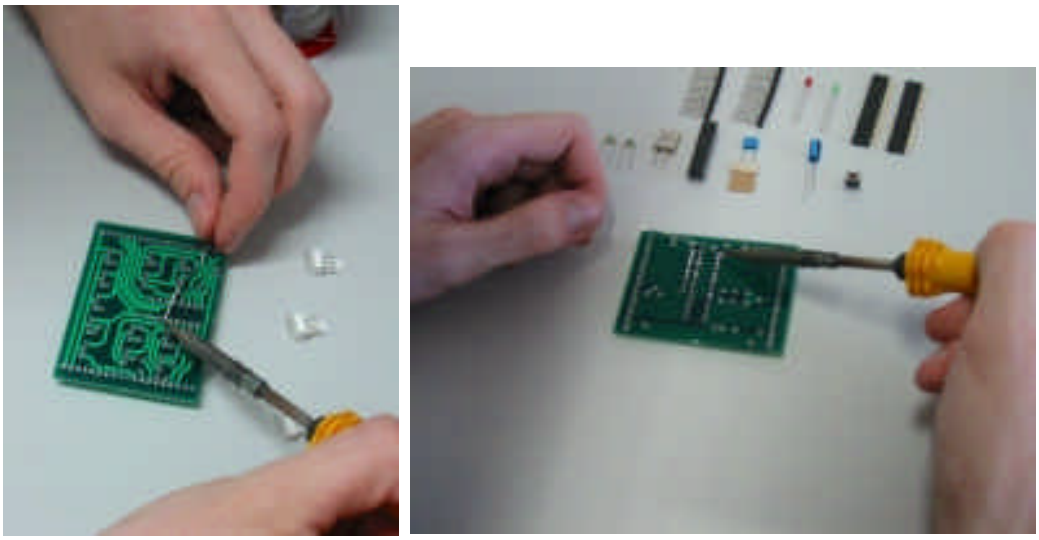
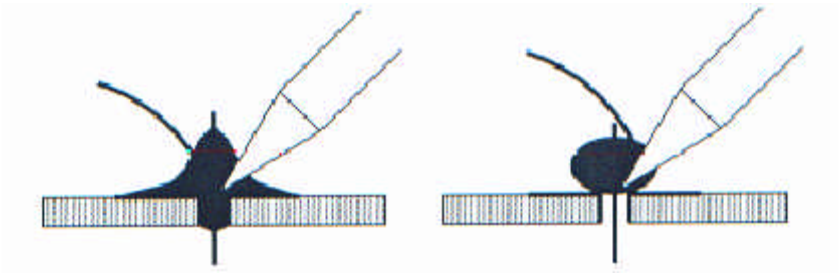
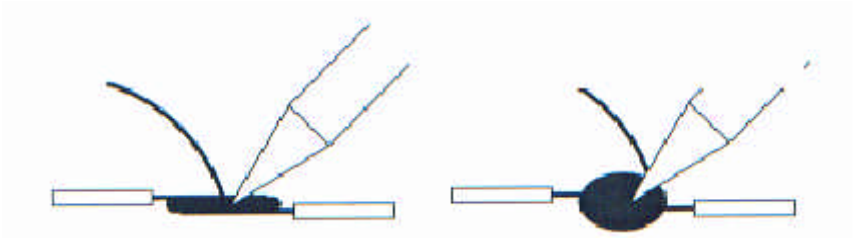


Bild 4.1 – Löten der Komponenten



Richtig Falsch



Richtig Falsch

Bild 4.2 - Lötmethoden

## 4.2 Einzelteile

Ein elektrischer Schaltkreis setzt sich aus mehreren Einzelteilen verschiedener Form, Größe und Codes, welche eine Unterscheidung zulassen, zusammen. In einer Schaltung sollte der Nutzer wissen, wie er die verschiednen Komponenten identifizieren kann.

Es sollte Ihnen bewußt sein, dass einige Komponenten polarisiert sind und nur in einer Richtung funktionieren. Ein mangelhafter Einbau der Komponente kann zu einer fehlerhaften Funktion oder zur Zerstörung des Schaltkreises führen.

Das Ziel dieses Kapitels ist es, dem Benutzer die verschiedenen Komponenten eines einfachen Schaltkreises näher zu bringen.

### 4.2.1 Widerstände

Normalerweise ist ein Widerstand ein kleiner Zylinder mit einigen farbigen Streifen. Die in diesem Set beigefügten Widerstände besitzen 0,25 W, welche sich für niedrige Spannung eignen, aber ausreichend für eine gute Performance des Roboters sind. Der Farbcode ist standardisiert, bestehend aus 4 farbigen Streifen um den Zylinder. Die ersten zwei stellen die Basis dar und der dritte den Exponent.

Ein Widerstand kann dadurch erkannt werden, daß eine zweistellige Zahl aus den ersten beiden farbigen Streifen gebildet wird. Der erste Streifen steht für den Zehnerbereich (**1**0,**2**0,**3**0…) und die zweite für den Einerbereich (0**1**,0**2**,0**3**...). Daraus entstehen Zahlen wie **12** oder **25**. Der dritte farbige Streifen kennzeichnet den Exponenten zur Basis 10, welcher mit der Zahl, die sich aus den ersten beiden Streifen ergeben hat, multipliziert wird.

Der vierte Streifen repräsentiert die Widerstandstoleranz. Wenn dieser Streifen silberfarben ist, so hat der Widerstand einen Toleranz von 10%. Wenn er golden ist, so besitzt er eine Toleranz von 5%. Die Toleranz gibt die mögliche Abweichung vom Nennwert des Widerstandes an, abhängig vom Hersteller. Zum Beispiel: Für einen Widerstand von 100Ω und einer Toleranz von 5%, kann der reale Wert zwischen 95Ω und 105Ω liegen.

| Farbe | Streifenwert | Multiplikationsfaktor |
|---|---|---|
| Schwarz | 0 | 1 |
| Braun | 1 | 10 |
| Rot | 2 | 100 |
| Orange | 3 | 1000 |
| Gelb | 4 | 10000 |
| Grün | 5 | 100000 |
| Blau | 6 | 1000000 |
| Violet | 7 | |
| Grau | 8 | |
| Weiß | 9 | |

Tabelle 4.1 – Widerstände Farbcode

Beispiel : Hat ein Widerstand folgende Farbsequenz: Orange, Weiß und Rot

**Orange = 3**
**Weiß = 9** à **39**
**Rot = 2** à **$10^2$**
à **39 x $10^2$ = 3900** W

so hat der Widerstand eine Größe von 3900 Ω.

#### 4.2.2 Widerstandsnetzwerk

Eine weitere Widerstandsart kann in einem elektrischen Schaltkreis gefunden werden. Sie besteht nicht aus einzelnen Widerständen, daß heißt sie sind in einer Gruppe angeordnet, welche von 3 bis 9 Widerständen variieren kann (Bild 4.2 zeigt ein Netzwerk aus 7 Widerständen).

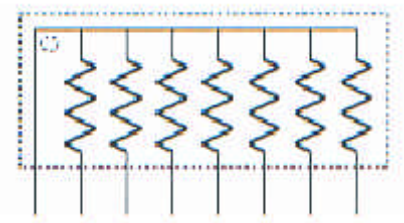


Bild 4.2 – Widerstandsnetzwerk

In diesem Bausatz besitzen die genutzten Netzwerke 8 Widerstände mit ein gemeinsamen Anschluß (gekennzeichnet mit einem Punkt). Zu beachten ist, daß dieses Bauteil polarisiert ist, und daher geprüft werden muß, ob es richtig eingesetzt wurde, bevor man dieses anlötet.

#### 4.2.3 Kondensatoren

Es gibt verschiedene Kondensatoren, die zum Kauf angeboten werden: monolithische, elektrolytische und Tantalkondensatoren.

4.2.3.1 Monolithische Kondensatoren

Sie bestehen aus Keramik oder Polyester, sind für gewöhnlich klein und besitzen eine niedrige Kapazität (pF oder nF). Diese Kondensatoren sind nie polarisiert.

4.2.3.2 Elektrolytische Kondensatoren

Diese Art von Kondensatoren sind meist von zylindrischer Gestalt, sind mit einer Kunststoffkapsel umhüllt und bieten hohe Kapazitätswerte ( > 1 µF). Dieser Kondensator ist polarisiert, und er ist größer, je größer der Wert der Kapazität ist.

4.2.3.2 Tantalkondensatoren

Dies sind sehr kompakte Kondensatoren und haben eine Tropfenform. Die Kapazitätswerte besitzen etwa die selbe Höhe wie die der elektrolytischen Kondensatoren. Der Vorteil ihrer Nutzung ist deren Zuverlässigkeit, aber dementsprechend sind sie auch teurer. Die Tantalkondensatoren sind immer polarisiert.

### 4.2.4 Dioden und LED's

Die Dioden und LED's (Licht emittierende Diode) haben zwei Anschlußbeine: die Anode und die Kathode. Die Anode muß, in Bezug auf die Kathode, mit einer positiven Klemmspannung verbunden werden, um einen Stromfluß zu gewährleisten. Wenn die Polung nicht beachtet wird, erlaubt diese Bauteilkomponente keinen Durchfluß. Bild 4.3 zeigt, wie man die Polung richtig erkennt.

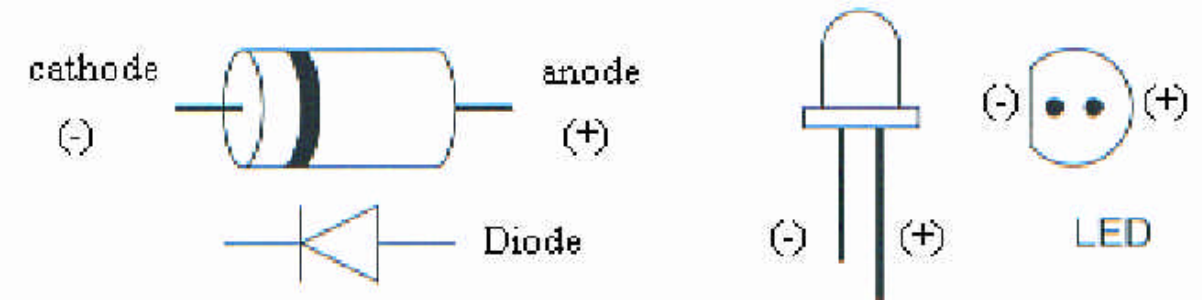



Bild 4.3 – Dioden- und LED - Polarisierung

Im Normalfall sind Dioden kleine Zylinder mit einem Streifen darum, welcher es möglich macht, die Kathode zu identifizieren. LED's sind Dioden, welche leuchten, wenn Strom durch ihre Anschlüsse fließt. Ihre Polarität wird anhand eines längeren Beines (Anode (+) ) gekennzeichnet, beziehungsweise, dort , von der Seite betrachtet, wo der Umkreis der LED flach ist, befindet sich die Kathode.

### 4.2.5 Integrierte Schaltkreis

Integrierte Schaltkreise (IC's) sind Teile, welche intern aus komplexen Stromkreisen zusammengesetzt sind. Sie sind in verschiedenen Formen und Größen erhältlich. Meistens, im Aufbau von handgefertigten Schaltkreisen, werden die DIPs (Dual-in-line-Gehäuse) genutzt.
Die IC's müssen entsprechend ihrer Numerierung, welche nicht auf ihrer Oberfläche erkennbar ist, angebracht werden. Trotzdem existieren 2 Markierungen, welche den Kontakt 1 anzeigen. Eine dieser Markierung ist ein kleiner Halbmond, welcher auf einer der beiden kontaktlosen Seiten des Chips angebracht ist, und zeigt, daß sich der Kontakt 1 zu seiner rechten befindet. Die zweite Markierung ist ein kleiner Kreis, welcher sich über der Kontaktstelle 1 plaziert wurde (Bild 4.4). Ein fehlerhafter Einbau dieses Teiles führt Gewöhnlicherweise zu dessen Zerstörung.

Bild 4.4 – Integrierter Schaltkreis der Form DIP

## 4.3 Konstruktion der Roboterboard´s

In diesem Kapitel wird der Aufbau der Leiterplattensätze erklärt, welche den Roboter bilden. Bild 4.5 zeigt einen Satz gedruckter Leiterplatten, die in diesem Elektronikbausatz enthalten sind.

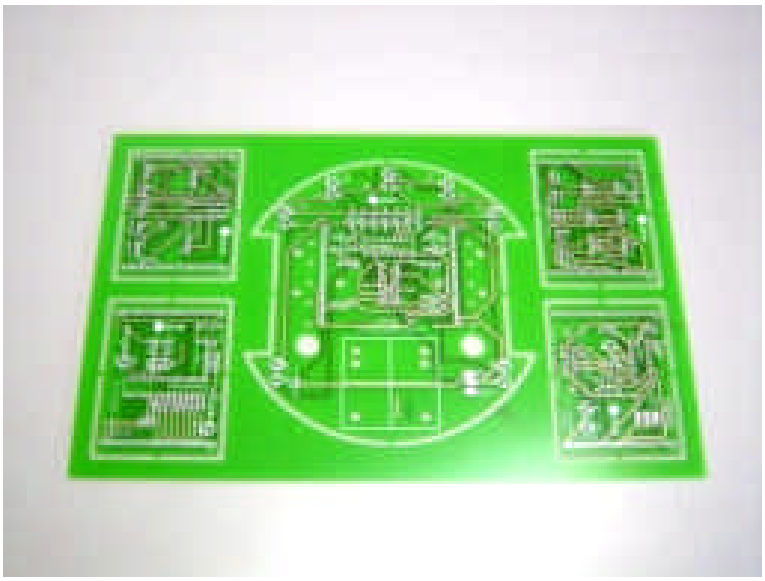


Bild 4.5 – Leiterplattenset, welches in der Roboterkonstruktion genutzt wird

### 4.3.1 Konstruktion des Controllerboards ID-PIC002

Begonnen wird getrennt mit dem ID-PIC002 – Board (Bild 4.6) Trenne dazu die kleinen Beine durch, welche es mit dem Rest der Leiterplatten verbindet. Wenn Nötig sind diese B eine mit Sandpapier einzuebnen. Suche dann die, für dieses Board benötigten Komponenten zusammen.

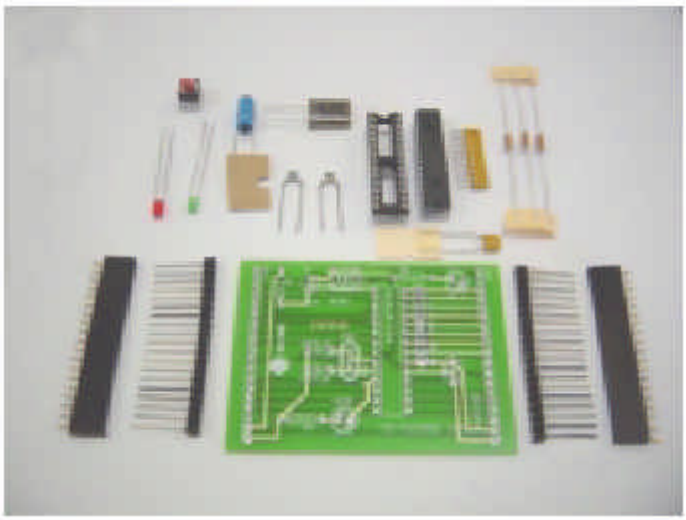


Bild 4.6 – Bauteile, die für das ID-PIC002-Board benötigt werden

Zu Beginn der Konstruktion werden die 3,3 kΩ Widerstände auf die gekennzeichneten Stellen gelötet. (Kapitel 4.2.1) Danach werden die Widerstandsnetzwerke aus 9 Widerständen aufgebracht, unter Beachtung dass der Punkt auf diesen zu der Markierung auf dem Board paßt.

Um die Orientierung dieses Bauteils zu erleichtern, ist ein weißer Umriß auf das Board aufgemalt worden. Dieser Umriß muß mit der Form des Widerstandes übereinstimmen.

Nun muß der Mikrokontrollersockel mit 28 Beinen aufgelötet werden. Es ist darauf zu achten, daß auch hier die Formen übereinstimmen, besonders auf die Übereinstimmung der Halbmonde. (Kapitel 4.2.5)

Löte nun die LED's unter Beachtung der Polarität. (Kapitel 4.2.4, grüne LED auf „LED 1" und rote LED auf „LED 2") Jetzt sind die zwei 15pF und der 100nF Kondensator aufzulöten. Nun wird der ELKO aufgelötet unter Beachtung der Polarität. (Kapitel 4.2.3) Löte nun den Druckknopf auf „S 1" und den Quarz auf „CRZ". Verlöte nun die zwei 19-beinigen Buchsenleisten auf und stecke die zwei dazugehörigen Stiftleisten auf. Am Ende wird der Mikrokontroller in den entsprechenden Sockel gesteckt.

Bild 4.7 zeigt das Kontrollerboard nach dem Zusammenbauen. Beachte, daß dieses Board zwei Buchsen- und zwei Stiftleisten besitzt. Die Buchsenleisten sind aufgelötet und die Stiftleisten anschließend nur drauf gesteckt.

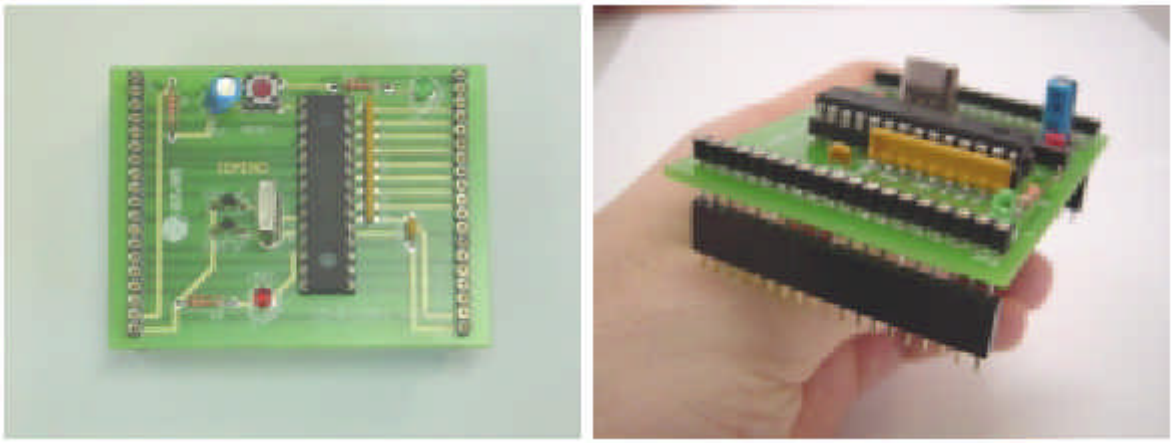


Bild 4.7 Mikrokontrollerboard

### 4.3.2. Konstruktion des Digitalboards ID-DS002

Trenne das ID-DS002 Board heraus. (Bild 4.8) Trenne dazu die kleinen Beine durch, welche es mit dem Rest der Leiterplatten verbindet. Wenn Nötig sind diese Beine mit Sandpapier einzuebnen. Suche dann die für dieses Board benötigten Komponenten zusammen.

Bild 4.8 – Bauteile, die für das ID-DSC002-Board benötigt werden

Begonnen wird bei der Konstruktion mit dem anlöten der verschiedenen 3-beinigen Anschlüssen (im Winkel von 90° ),zu beachten ist hierbei, daß die zwei PWM Anschlußstellen in einem Winkel von 180° zu den ersten

beiden angebracht sind (Bild 4.9) Löte nun zwei 19–beinige Buchsenleisten auf das Board und stecke die zwei dazugehörigen Stiftleisten hinein.

Bild 4.9 zeigt das zusammengebaute Digital - Board. Alle Kontaktleisten im Anschluß B und einige der Bits von Anschluß C sind ansprechbar. Jeder Sensor ist mit einem der 3–beinigen Anschlüsse verbunden. Im Bild wird an jeder Anschlußstelle gezeigt, welcher Anschluß zu dem Masse-Signal „ - “ paßt. Der mittlere Kontaktstift stellt die Spannung von 5 V bereit und die anderen Kontaktstifte entsprechen dem Signalausgang.

Das selbe Bild zeigt die Anschlußstellen C1 und C2 (PWM2, PWM1) befinden sich im Winkel von 180° mit den restlichen Anschlußstellen. Wie das Kontroller - Board besitzt dieses auch zwei aufgelötete 19-beinige Buchsenleisten und die Stiftleisten werden wieder aufgesteckt.

**Vorderseite**

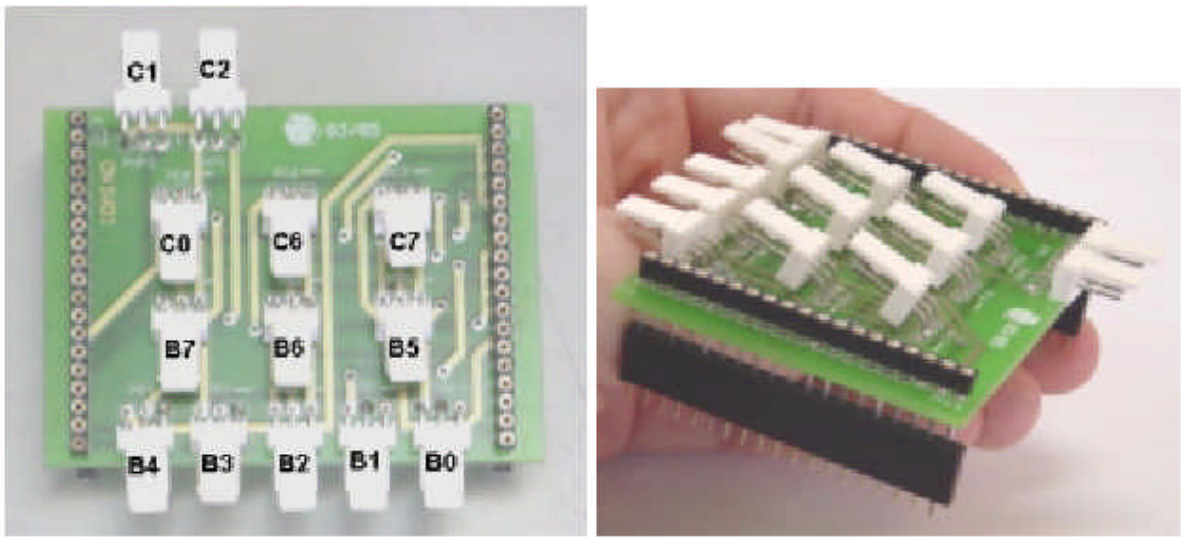



Bild 4.9 – Digitales Schnittstellen – Board

#### 4.3.3 Konstruktion der Analogschnittstelle ID-AS002

Trenne auch das ID -AS002 - Board, ebenso wie die vorhergehenden, von den restlichen Leiterplatten, und suche die zugehörigen Bauteile heraus.

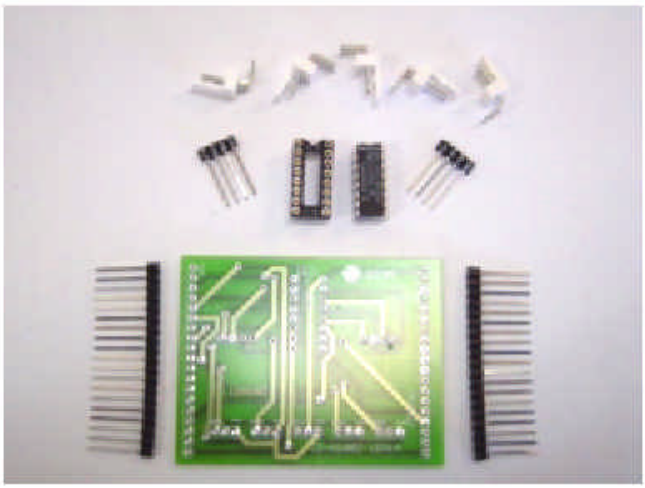


Bild 4.10 – Bauteile, die für das ID-AS002-Board benötigt werden

Beginne den Zusammenbau des analogen Board’s mit dem Auflöten der 3-beinigen Anschlußstellen (um 90° angewinkelt), auf die Stellen die mit PA1, PA2, PA3, PA4 und PA5 gekennzeichnet sind. Die Richtung wird in Bild 4.11 dargestellt. Löte die zwei 4–beinige Buchsenleisten auf die Stellen, welche mit JP9 und JP10

gekennzeichnet sind. Löte den 16-beinigen Sockel entsprechend den Board – Linien, und danach die 19-beinigen Buchsenleisten auf das Board. Stecke den integrierten Schaltkreis 74HC2051 auf den Sockel (Halbmond auf Halbmond).

Bild 4.11 zeigt ein zusammengebautes Analogschnittstellenboard. Alle Anschlusstellen des Bereiches A sind ansprechbar. Die Anschlüsse 1 bis 5 von Bereich A sind direkt ansprechbar und jeder von diesen verfügt über 8 Eingänge.

**Vorderseite**

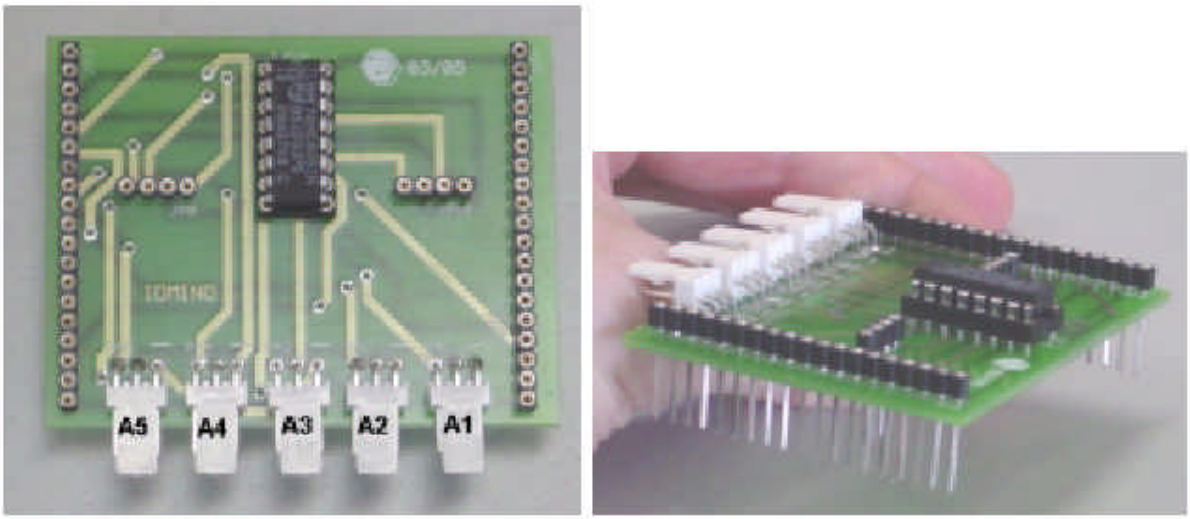



Bild 4.11 – Analogschnitstellen-Board

### 4.3.4 Aufbau des Sensorboards ID-BASEIR002

Trenne auch dieses Board von dem Leiterplattenset, und entferne eventuelle Grate, falls nötig mit feinem Sandpapier. Suche auch hier wieder die benötigten Bauteile zusammen.

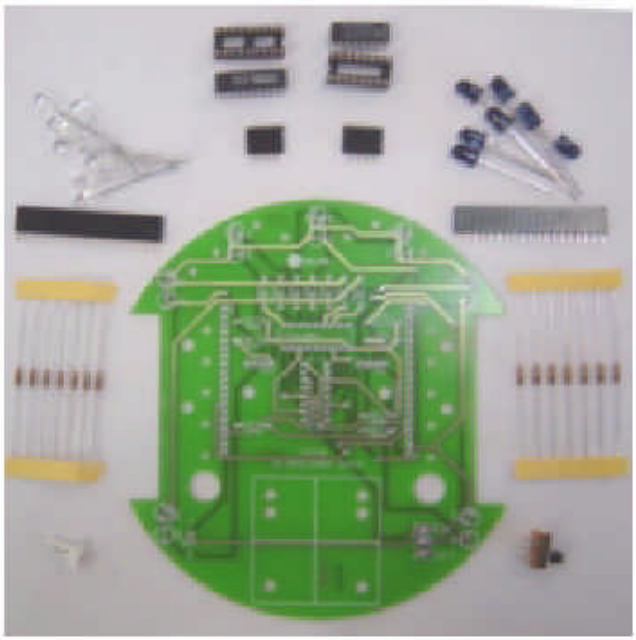


Bild 4.12 – Komponenten die zum Bau des ID-BASEIR002 –Boards benötigt werden

Als erstes werden nun die 7 100 • Widerstände (RIR0 bis RIR6) und die 7 10 k• Widerstände (RTR0 bis RTR6) aufgelötet. Anschließend werden die zwei 16- und 18-beinigen Sockel in Übereinstimmung mit den Linien des Boards aufgelötet. Die 2 und 4-beinigen Stiftleisten werden an auf die Stellen gelötet, die mit JP11 und JP10 gekennzeichnet sind. Nun werden die 19-beinigen Stiftleisten auf das Board aufgelötet. Anschließend muss noch der ON-OFF Schalter auf das Board gebracht werden.

Suche die Infrarot -Sender und Empfänger heraus, und bestimmen deren Polarisierung (siehe Kapitel 4.2.4). Ermitteln sie, wo sie befestigt werden müssen. Die Emitter (Licht -Sender) müssen an die Stellen gelötet werden, welche mit IR0 bis IR6 gekennzeichnet sind. Die Receiver (Empfänger) werden auf die Stellen aufgebracht, die mit TR0 bis TR6 gekennzeichnet sind. Beim Befestigen dieser Sensoren ist auf deren Verwendung zu achten. Zum Beispiel, wenn die Sensoren zur Hindernis -Erkennung genutzt werden sollen, müssen die Sensoren oberhalb des Boards angebracht werden. Wenn sie eine auf dem Boden befindliche Linie erkennen sollen, müssten sie unterhalb des Boards angebracht werden. Bild 4.14 zeigt ein Einbaubeispiel mit drei Emitter/Receiver Paaren im Vordergrund (IR2, IR3, IR4 und TR2, TR3, TR4), welche unter dem Board platziert wurden, um eine Linienverfolgung durchführen zu können. Die restlichen Sensoren wurden für die Hindernis-Erkennung oberhalb angebracht.
Beachte (Bild 4.1 4), dass die auf dem Kopf stehenden Sensoren so gelötet werden müssen, dass eine bestimmter Abstand vom Board zu den Sensorenden eingehalten wird. Dieser Abstand sollte 2,5 bis 3 cm betragen. Plaziere nun die IC's auf den Sockeln, und achte dabei auf die korrekte Ausrichtung (Halbmond auf Halbmond).

Bild 4.13 – Aufbau der Batteriehalterung

Jetzt wird die weiße Anschlussbuchse auf JP8 gelötet. Die Batteriehalterung wird, wie in Abbildung 4.13 gezeigt, eingebaut. Dazu werden die kleinen metallischen Ans chlüsse mit den Drähten der Halterung verlötet, und anschließend in den dazugehörigen weißen Halter gesteckt (der mit den 2 Löchern). Dieser Halter wird mit den Stromanschlüssen "+" und "-" des ID-BASEIR002-Boards verbunden. Achte beim Verbinden darauf, da ss der rote Draht zur positiven Anschlussbuchse gehört.

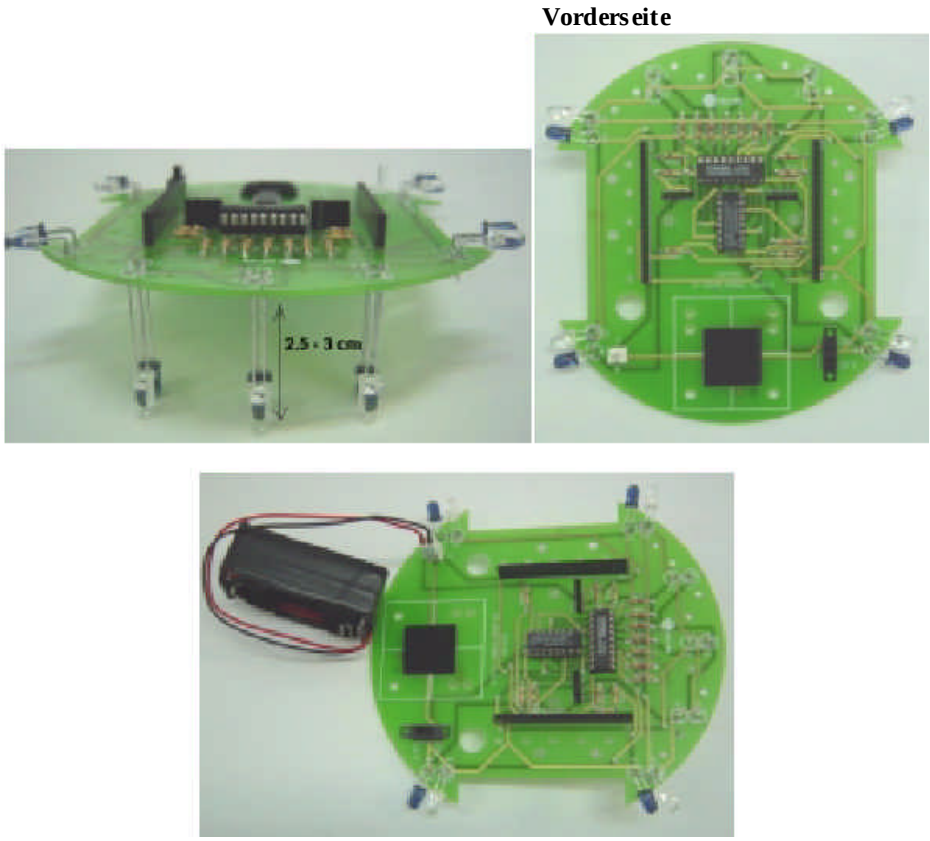



Bild 4.14 - Vorderseite der ID -BASEIR002-Boards

Wenn Sie auf der Vorderseite des Roboters die mitgelieferten Mikroschalter verwenden wollen, müssen sie mit dem Löten auf einer Seite der drei Drahtkabel mit dem Mikroschalter beginnen und auf die anderen Seite kommen dann kleine Metallanschlüsse, welche auch in dem Bastelsatz enthalten sind.
Stecke dann die Anschlüsse in den weißen Halter ( der mit den drei Löchern). Achte darauf, dass der gemeinsame Anschluss, als "COM" im Mikroschalter dargestellt, in den linken Anschluss der weißen Halterung gehört, wenn die zwei kleinen Plastikstifte des Halters auf der Oberseite sind (Bild 4.15).

Bild 4.15 - Mikroschalterzusammenbau

Stecke die 2 Mikroschalter in das ID-BASEIR002-Board und die 4 Schrauben, die sich auf der Board-Vorderseite in der Nähe von IR4 und IR2 befinden(Bild 4.16).

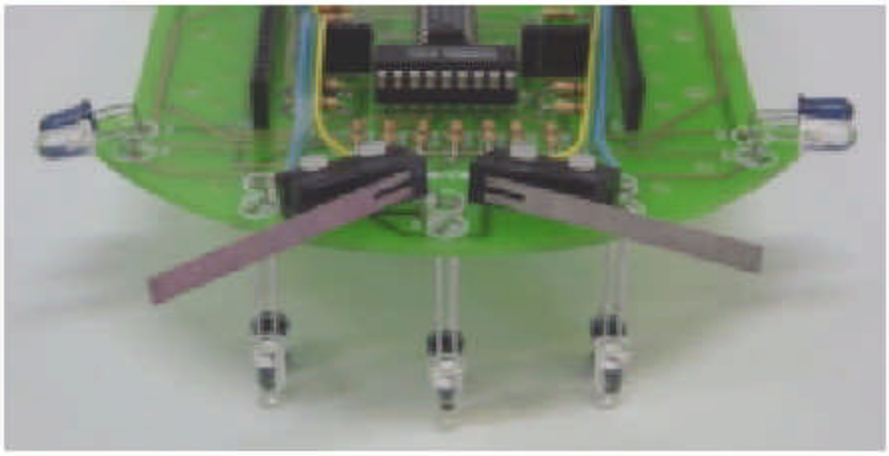


Bild 4.16 - Mikroschalterplatzierung

### 4.3.5 Zusammenschalten der verschiedenen Boards

Dieses Kapitel zeigt, wie die verschiedenen Boards und Komponenten des Roboters verbunden werden. Plaziere die Motoren und der Freilaufrolle wie in Abbildung 4.17 gezeigt.

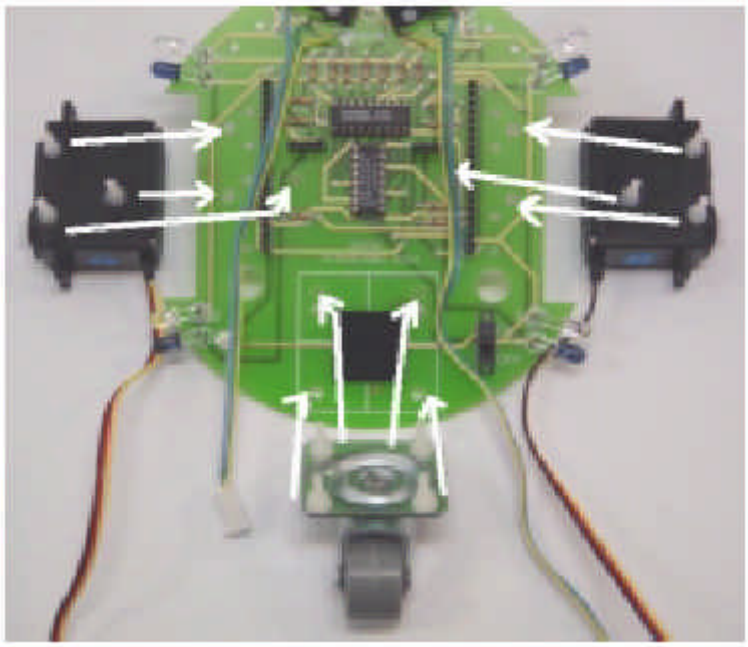


Bild 4.17 - Motoren und Freilaufrad

Die zwei Motoren müssen über unter dem ID-BASEIR002-Board platziert werden. Für jeden Motor existieren dazu fünf Löcher, aber nur die hinteren drei müssen benutzt werden (Bild 4.17). Befestige nun auch die Freilaufrolle wie in Abbildung 4.17 gezeigt, und nutze dazu die 4 restlichen Löcher. Führe die Drähte der Servomotoren durch die Löcher, die sich im mittleren Teil des Boards befinden (Bild 4.18).

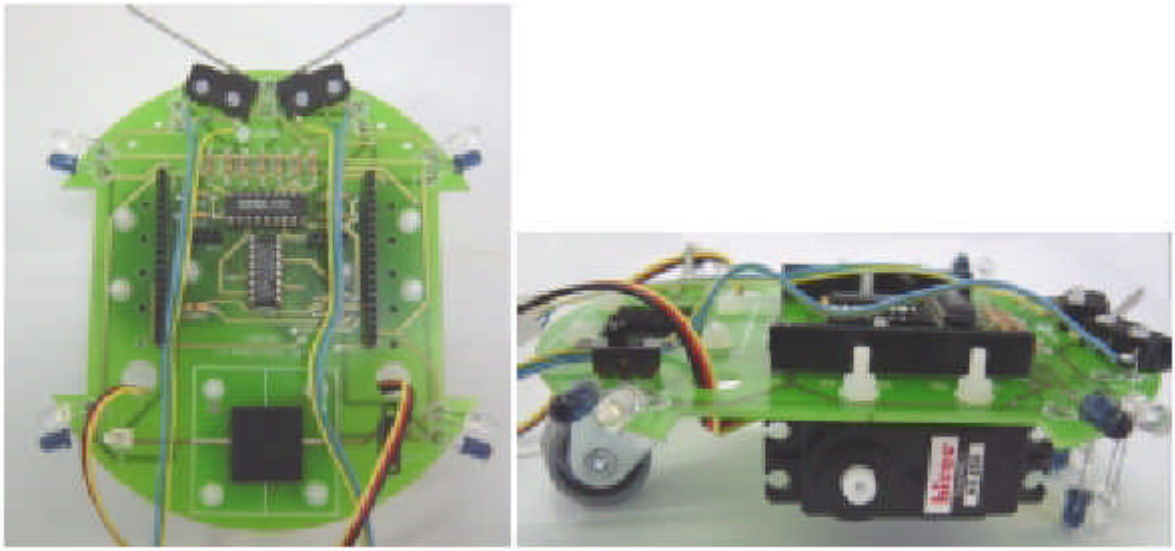


Bild 4.18 - Einbau der Motoren und des Laufrades

Setze die Räder auf die Motoren, und ziehe deren Schrauben aus der Mitte fest (Bild 4.19).

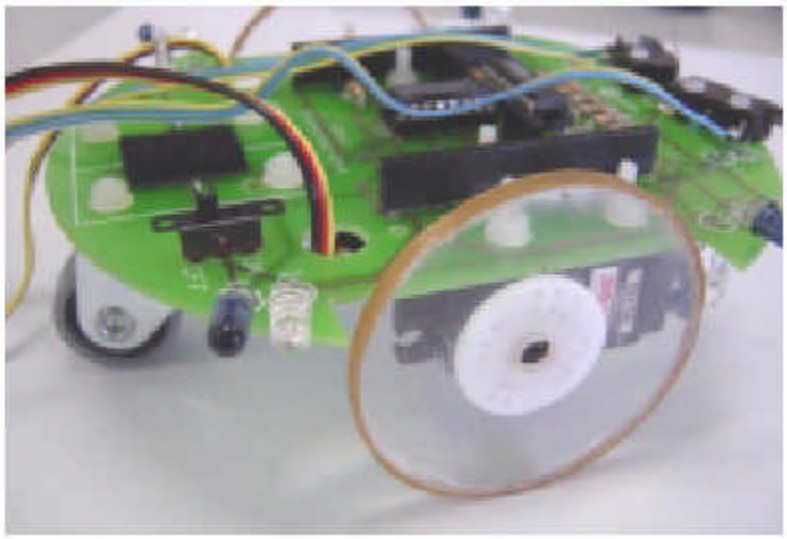


Bild 4.19 - Zusammenbau von Motoren und Rädern

Stecke nun das ID-AS002-Board in das ID-BASEIR002-Board. Überprüfe, ob alle Beine der Stiftleisten in die dazugehörigen Buchsenleisten hineingesteckt werden können (19- und 3-beinige Leisten). Drücke beide Boards solange aufeinander, bis sie vollständig ineinander stecken (Bild 4.20).

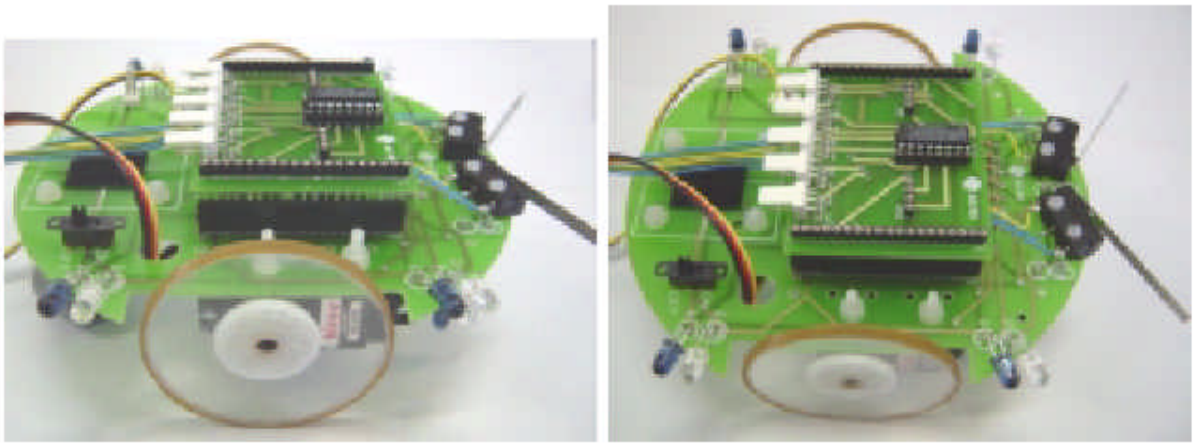


Bild 4.20 - Zusammenfügen von ID-BASEIR002- und ID-AS002-Board

Verbinde die Mikroschalter Anschlussstecker: den linken Mikroschalter mit dem Anschluss PB4, und den rechten Mikroschalter mit dem Anschluss PB3 des ID-DS002-Boards. Verbinde nun die Motorenanschlussstecker mit dem PWM -Ausgang: der rechte Motor mit dem PWM1-Ausgang und der linke Motor mit dem PWM2-Ausgang. Achten Sie auf das schwarze Kabel (Masse), welches auf der linken Seite eingesteckt werden muss, wenn der Roboter dem Nutzer seine Rückseite zeigt (Bild 4.2.1).

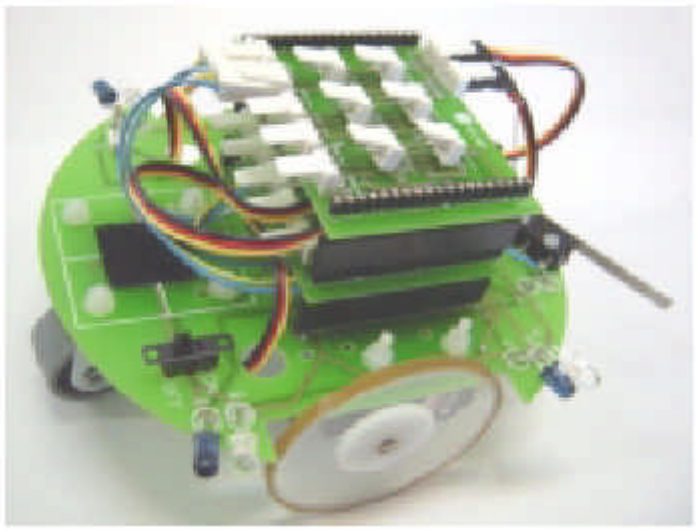


Bild 4.21 - ID-DS002-Board Verkleidung

Zum Schluss wird noch das ID-PIC002-Board über das ID-DS002-Board gesetzt. Beide sollten wieder so aufeinandergedrückt werden, dass sie vollständig miteinander verbunden sind. Danach sollte der Roboter in etwa so aussehen, wie in Abbildung 4.22 gezeigt wird.

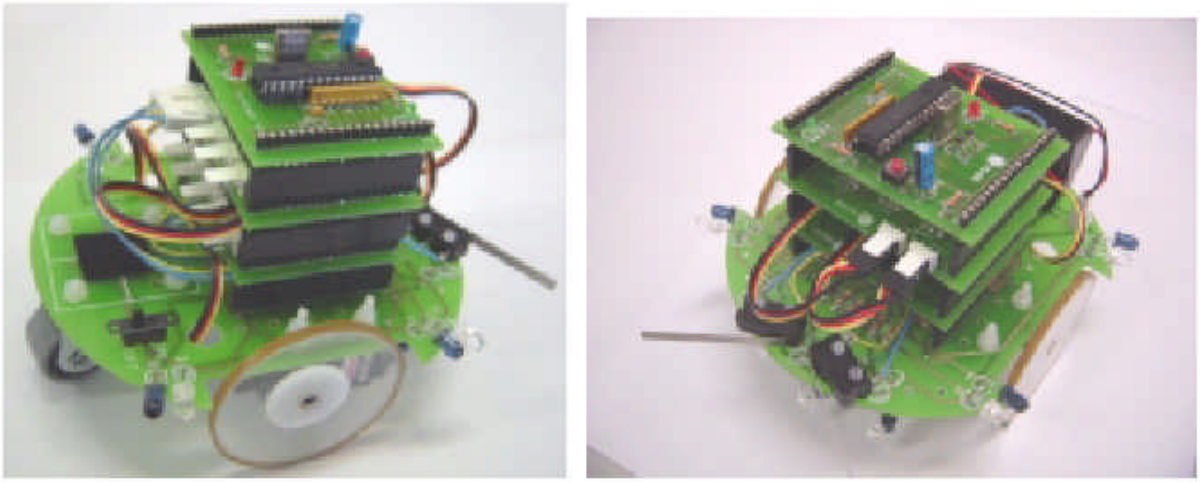


Bild 4.22 - Roboter Aufbau

Wie in Kapitel 3.2 beschrieben, besetzen die Infrarotsensoren nur einen von fünf analogen Eingängen des Mikrocontrollers. Vier analoge Eingänge sind noch verfügbar, und der Nutzer kann sie auch verwenden, um andere analoge Sensoren hinzuzufügen.
Die Verbindung neuer Sensoren, mit analoger Berechnung, ist sehr einfach. Es reicht, den Ausgangskontakt des neuen Sensors mit dem Eingang eines verfügbaren analogen Signals vom Roboter zu verbinden, um genauer zu sein, mit dem analogen Schnittstellenboard ID-AS002.
Das digitale Schnittstellenboard ID-DS002 verfügt über 11 digitale Ein- und Ausgänge, welche auch genutzt werden können um die Zahl der Sensoren und Aktoren zu steigern. Die zwei Mikroschalter besetzen zwei dieser Eingänge.

### 4.3.6 Zusammenbau des Programmierboards ID-PROG002

Trenne auch die ID-PROG002 - Leiterplatte von den Restlichen (Bild 4.23), und entferne eventuelle Grate z.B. mit feinem Sandpapier. Suche anschließend die zugehörigen Bauteilkomponenten zusammen.

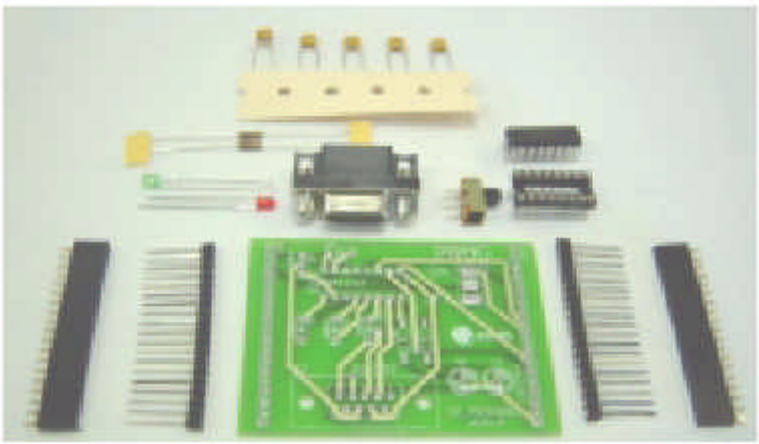


Bild 4.23 -Bauteile, welche für das ID-PROG002-Board benötigt werden

Dieses Board ermöglicht die Programmierung des Roboters über einen seriellen Anschluss eines Computers.
Löte als erstes die 1 kΩ Widerstände auf. Anschließend ist der 16-beinige Sockel an der Reihe, unter Beachtung der Board - Linien. Die beiden LEDs müssen auf TX (Grün) und RX (Rot) gelötet werden, wobei auch hier wieder auf deren Polung zu achten ist (Kapitel 4.2.4).
Nun werden die fünf 100 nF Kondensatoren (C5 bis C9) aufgelötet. Danach geht es mit den zwei 19-beinigen Buchsenleisten weiter. Jetzt ist noch der ON-OFF Taster aufzulöten, und der serielle Anschlussstecker an die mit RS-232 gekennzeichnete Stelle. Zum Schluss werden die zwei 19-beinigen Stiftleisten in die Buchsenleisten gesteckt, und der 232 IC in den dazugehörigen Sockel (Halbmond auf Halbmond).
Nach diesen Vorgängen sollte das Board wie in Abbildung 4.24 aussehen.

**Vorderseite**

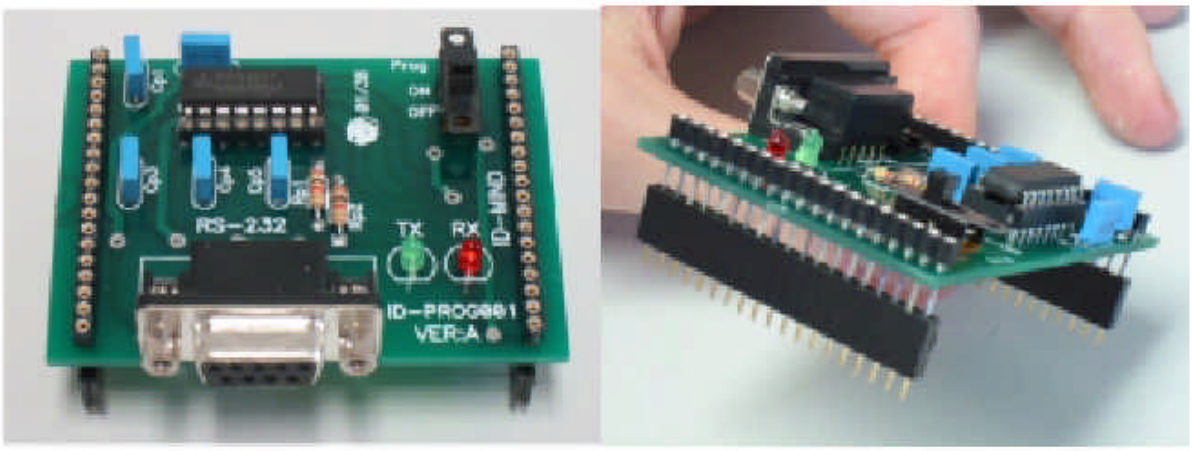



Bild 4.24 -Seitenansicht des ID -PROG002 -Boards

# 5 Mikrocontroller

Die verwendete Mikrocontroller gehören zu der PIC 16F876 Familie von Mikrochip. Diese *Chips* besitzen große Vorteile gegenüber der Großzahl an Mikrocontrollern, da sie verschiedene Funktionsblocks beinhalten, die es ermöglichen eine große Zahl an Aufgaben ohne die Verwendung zusätzlicher Elektronik zu lösen.
Mit diesem Chip ist die Möglichkeit von A/D –Wandlung, Zeitsteuerung, serieller und paralleler Datenübertragung, PWM-Ausgänge, Interruptoren, usw. zu finden ( Bild 7.1).
Er enthält einen EEPROM Datenspeicher von 8 K Wörtern mit je 14 Bits, Datenspeicher für RAM Daten von 368 Wörtern mit je 8 Bits. Der Mikrocontroller kann durch den Nutzer programmiert werden, da eine einfache Programmiersprache verwendet wird, die aus nur 35 Basisinstruktionen aufgebaut ist, welche der Mplab - Hersteller für den Mikrochip bereitstellt. (http: \\www.microchip.com) Auf der Seite:
http://www.microchip.com/10/tools/picmicro/devenv/mplabi/software/v540/index.htm
Die Programmierung des Chips geschieht durch die Verwendung des Programmierboards ID-PROG002, und durch das **IdMind' Loader1** Programm, welches die erstellte Datei (*.hex) über **Mplab** lädt.

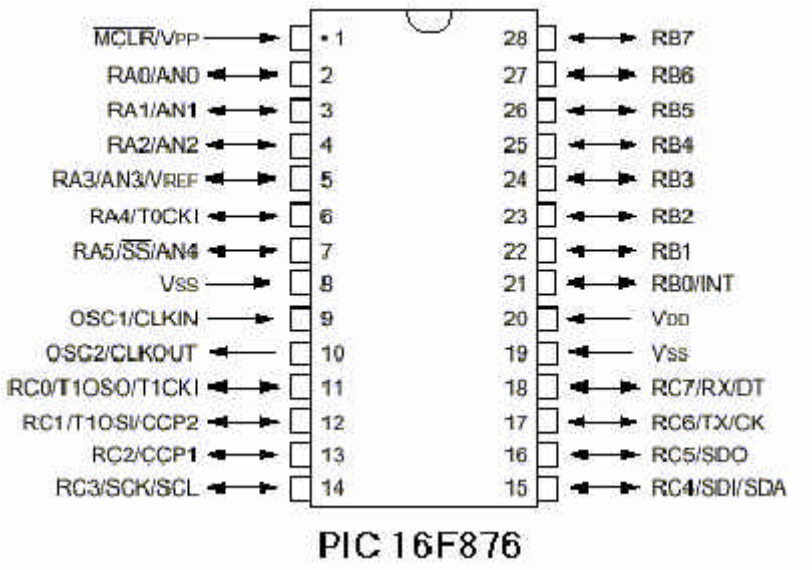



Bild 5.1 - PIC 16F876 Mikrocontroller

Die Ein - und Ausgänge werden wie folgt organisiert:

**Anschluß A** à 6 digitale Eingänge/Ausgänge oder 5 analoge Eingänge
**Anschluß B** à 8 digitale Eingänge/Ausgänge
**Anschluß C** à 8 digitale Eingänge/Ausgänge , welche 2 PWM Ausgänge haben können.

Dieser Mikrocontroller besitzt interne Datensätze, welche ihm erlauben, Situationen von einigen durchgeführten Operationen zu testen. Zum Beispiel ist es möglich:

zu testen, ob eine Abweichung vom Ergebnis positiv oder negativ ist,
zu testen, wieviel Zeit verstrichen ist, seit dem der Timer angeschaltet wurde,
zu testen, ob ein analoges Auslesen schon durchgeführt wurde,
oder den Ausgangsanschluß in einen Eingangsanschluß zu ändern.

Diese Datensätze werden in vier Datenbanken (Bank 0,1,2 und 3) gespeichert. Die Tabelle in Bild 5.2 zeigt die Verteilung der Datensätze auf die verschiedenen Datenbanken. Später werden wir auf die wichtigsten Datensätze für die Initialisierung dieser Mikrocontroller eingehen.

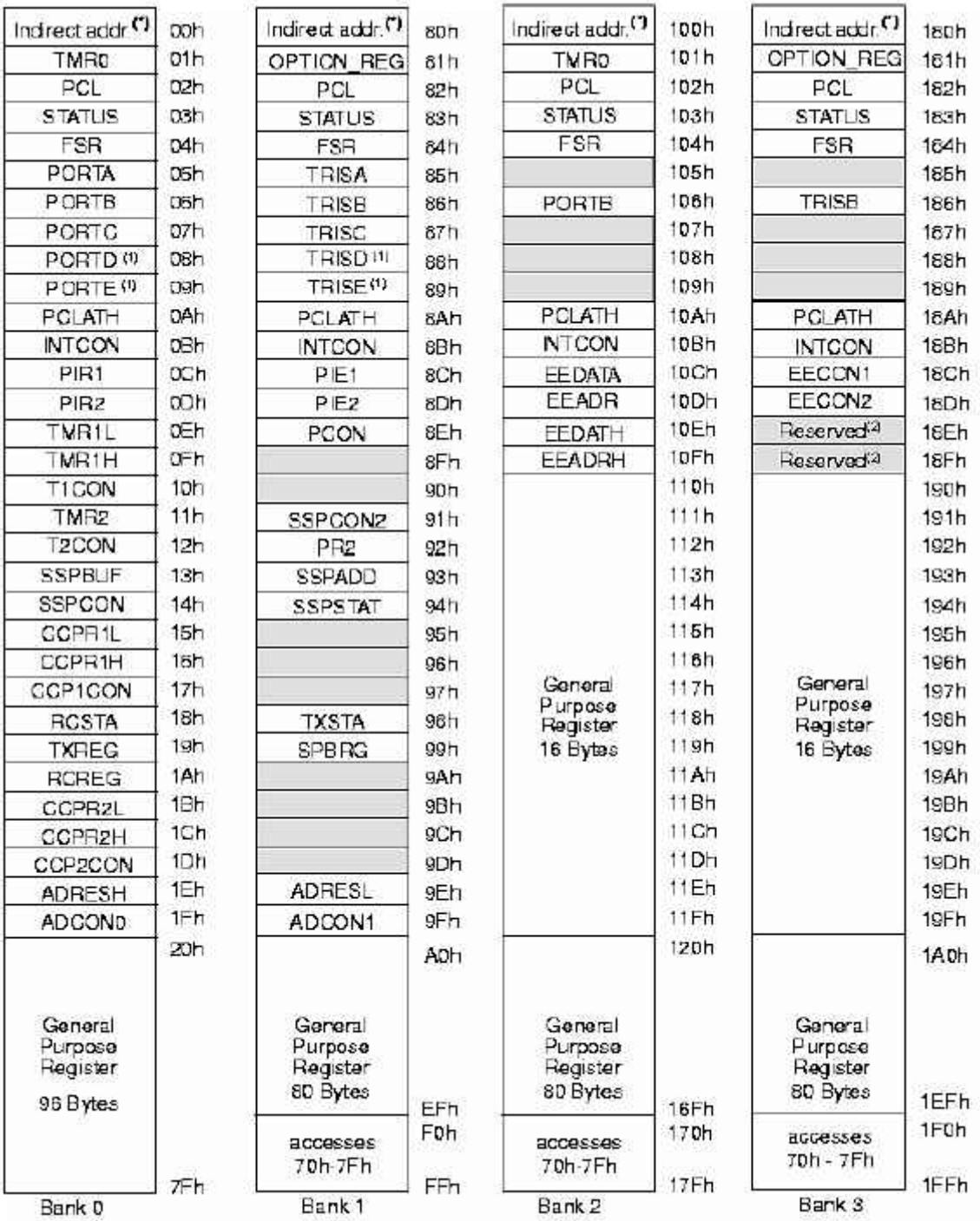
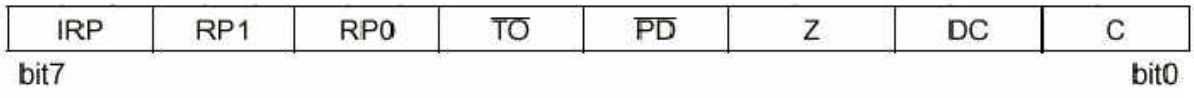



Bild 5.2 – Datensatzverteilung

Einer der wichtigsten Datensätze ist der **STATUS** – Datensatz. Dies ist ein Datensatz, der es ermöglicht, festzulegen, in welcher Datenbank wir uns befinden, und zu testen, ob das Ergebnis einer arithmetischen oder logischen Operation gleich 0 ist, oder negativ.

**STATUS** – Datensatz:




**RP1:RP0:** Auswahl der Datenbank-Bits

11 = *Bank* 3

10 = *Bank* 2

00 = *Bank* 1

01 = *Bank* 0

**Z:** *Bit Null*

Wenn **Z** = 1, dann war die letzte arithmetische oder logische Operation gleich Null.

Wenn **Z** = 0, dann war die letzte arithmetische oder logische Operation ungleich Null.

**C:** *Carry* (Verwendung von mehr als 8 Bits)

**C** = 1, im Fall das *Carry* eintritt

**C** = 0, in allen anderen Fällen

Für ein besseres Verständnis der Programmierung, und für die Verwendung von Datensatz in Einführungsbeispielen, werden in der nächsten Tabelle die Anweisungen an den Mikrocontroller gezeigt. In der rechten Spalte werden die Änderungen des STATUS – Datensatzes angezeigt.

| **Operation** | **Beschreibung** | **STATUS** |
|---|---|---|
| **Operationen mit Datensatzwörtern ( d = 0 à Ergebnis in w, d = 1 à Ergebnis in f)** | | |
| **ADDWF f,d** | W+f | C, Z |
| **ANDWF f,d** | W AND f | Z |
| **CLRF f** | f = 0 | Z |
| **CLRW** | W = 0 | Z |
| **COMF f,d** | Zahlenkomplement von f | Z |
| **DECF f,d** | f – 1 | Z |
| **DECFSZ f,d** | f – 1, überspringe nächste Operation, wenn f-1=0 | |
| **INCF f,d** | f + 1 | Z |
| **INCFSZ f,d** | f + 1, überspringe nächste Operation, wenn f + 1=0 | |
| **IORWF f,d** | w OR f | Z |
| **MOVF f,d** | w = f oder f = f | Z |
| **MOVWF f** | f = w | |

| | | |
|---|---|---|
| **NOP** | macht nichts | |
| **RLF f,d** | speichert 1 Bit links von Carry | C |
| **RRF f,d** | speichert 1 Bit rechts von Carry | C |
| **SUBWF f,d** | f - w | C, Z |
| **SWAPF f,d** | verbinde die ersten 4 mit den letzten 4 Bits | |
| **XORWF f,d** | f XOR w | Z |

| **Operationen mit Datensatzbits ( b = 0 bis 7, Anzahl der Bits die geändert oder getestet werde)** | | |
|---|---|---|
| **BCF f,b** | Bit b vom Datensatz f ist gleich 0 | |
| **BSF f,b** | Bit b vom Datensatz f ist gleich 1 | |
| **BTFSC f,b** | teste Bit b von f, wenn gleich 0, dann überspringe nächste Operation | |
| **BTFSS f,b** | teste Bit b von f, wenn gleich 1, dann überspringe nächste Operation | |

| **Operationen mit Buchstaben ( k als Zahl von 0 bis 255 in mathematischen Operationen, und von 0 bis 2047 in Unterprogrammen** | | |
|---|---|---|
| ADDLW k | w = w + k | C, Z |
| ANDLW k | w = w AND k | Z |
| CALL k | Das Unterprogramm k wird aufgerufen | |
| CLRWDT | Überwachungstimer auf 0 | |
| GOTO k | Gehe zum Unterprogramm k | |
| IORLW | w = w OR k | Z |
| MOVLW k | w = k | |
| RETFIE | Rückgabe einer Interruption | |
| RETLW k | Rückgabe eines Buchstaben aus k | |
| RETURN | Rückgabe aus einem Unterprogramms | |
| SLEEP | Geht in Standby-Modus | |
| SUBLW k | w = k – w | Z, C |
| XORLW k | w = w XOR k | Z |

Wie gezeigt, können arithmetische und logische Funktionen Änderungen im **STATUS** – Datensatz hervorrufen.

Wenn man die Datensätze und die Befehle kennt, ist es möglich einige Programmblöcke zu erzeugen. Zum Beispiel die Konfiguration der verschiedenen Anschlüsse (A, B und C) als Eingänge, Ausgänge oder gemischt.

## 5.1 Ein- und Ausgangskonfiguration

Um jeden Pin des Mikrocontrollers als Eingang oder Ausgang zu gestallten, ist es notwendig, in den dafür geeigneten internen Datensätzen die Art festzulegen, wie diese Pins genutzt werden sollen. Diese Konfiguration von jedem **Port** (Anschluß), wird durch die Konfiguration der passenden **Bits** aus der Datenbank 1 gemacht.

Der Prozeß funktioniert wie folgt:

1. Gehe zu Datenbank 1
2. Konfiguriere die **Ports** durch schreiben in die **Bits** (**BitA**, **BitB** und **BitC**)
3. Um sie als Ausgänge festzulegen, muß der **Bit** den Wert 0 erhalten
4. Um sie als Eingänge festzulegen, muß der **Bit** den wert 1 erhalten
5. Gehe zurück zu Datenbank 0

Konfigurationsbeispiel:

```
IN_OUT:                     ; Konfiguration der Ein - und Ausgänge
        bsf STATUS,RP0      ; gehe zu Datenbank 1

        bcf STATUS,RP1

        movlw .255          ; konfiguriere PortA als Eingänge

        movwf TRISA

        movlw .0

        movwf TRISC         ; konfiguriere PortC als Ausgänge

        bsf TRISC,7         ; mache aus Bit 7 von Anschluß C einen Ausgang

        movlw .255

        movwf TRISB         ; PortB wird als Eingänge initialisiert

        bcf TRISB,7         ; konfiguriere Bit 7 von Anschluß B als Ausgang

        bcf TRISB,5         ; konfiguriere Bit 5 von Anschluss B als Ausgang

        bcf TRISB,3         ; konfiguriere Bit 3 von Anschluss B als Ausgang

        bcf TRISB,1         ; konfiguriere Bit 1 von Anschluss B als Ausgang

        bcf STATUS,RP0      ; Datenbank 0
```

return

Eine andere Funktion des PI-Controllers (**PIC**), welche genutzt wird, ist die Erfassung und die Umwandlung eines analogen Signals in ein digitales Signal (Datenwort aus 8 Bits). Das nächste Kapitel erklärt, wie der AD-Wandler (Analog-Digital) des Mikrocontroller intern funktioniert.

## 5.2 Analoge Eingänge

Das Umwandlungsmodul A/D des PIC 16F876 besteht aus 5 Eingängen, von denen alle gebündelt sind, wie in Bild 5.3 gezeigt wird.

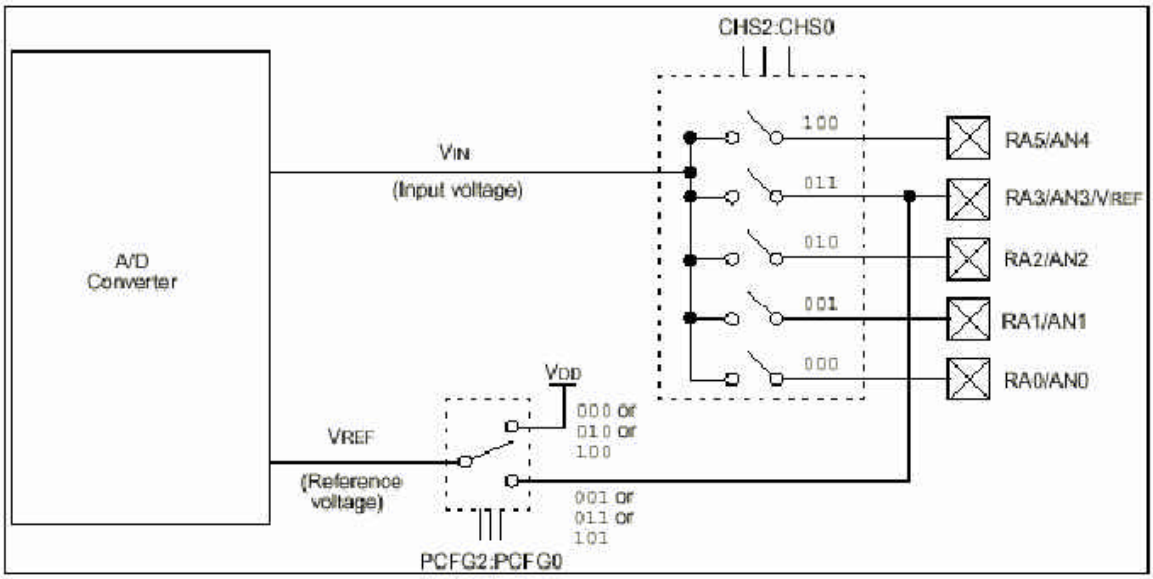



Bild 5.3 – Inneres Diagramm des PIC AD – Wandlers

Dieses Modul erlaubt die Umwandlung von einem analogen Signal in ein entsprechendes digitales Signal von 8 Bits. Dazu gibt es die Möglichkeit eine externe Bezugsspannung zu nutzen, deren maximaler Umwandlungswert (255) dieser Bezugsspan nung entspricht.

Falls irgendeine Nennspannung verwendet wird, wird die Leistung der Spannung als Sollwert genutzt.

Um dieses Modul programmieren zu können, müssen drei neue Datensätze erstellt werden, welche als nächstes eingeführt werden.

**ADRES** - Ausgänge der Analog-Digital Umwandlung

**ADCON0** - Kontrolle über den 0-ten Datensatz

( Dieser erlaubt, die Umwandlungszeit und den Kanal, der konvertiert werden soll, zu wählen. Außerdem ermöglicht dieser, die Durchführung der Umwandlung, und die Überprüfung, ob die Umwandlung schon erledigt wurde.)

**ADCON1** - Kontrolle über den 1-ten Datensatz

( Ermöglicht die Festlegung der Eingänge des Anschlusses A als Analog oder Digital)

#### 5.2.1 ADCON0

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| ADCS1 | ADCS0 | CHS2 | CHS1 | CHS0 | GO/DONE | -- | ADON |

**ADCS1:ADCS0**: durch Frequenzwandlung ausgewählte Bits

00 = **Fosc**/2

01 = **Fosc**/8

10 = **Fosc**/32

11 = **FRC** ( Frequenz des internen RC -Schwingers)

**Fosc** ist die Quarzfrequenz, welche im **PIC** – Aufbau verwendet wird. Hier beträgt sie 4 MHz. Die Festlegung dieser Frequenz ist wichtig, weil dadurch sichergestellt werden kann, dass die Umwandlung richtig durchgeführt wurde. Tabelle 5.1 zeigt die zeitlichen Relationen, um eine ordentliche Umwandlung sicherzustellen.

| **Analog -Digitaler Taktgeber (TAD)** | | **eingesetzte Frequenz** | | | |
|---|---|---|---|---|---|
| **Operation** | **ADCS1:ADCS0** | **20 MHz** | **5 MHz** | **1,25 MHz** | **333,33 kHz** |
| 2Tosc | 00 | 100 ns(2) | 400 ns(2) | 1,6 • s | 6 • s(3) |
| 8Tosc | 01 | 400 ns(2) | 1,6 • s | 6,4 • s | 24 • s(3) |
| 32Tosc | 10 | 1,6 • s | 6,4 • s | 25,6 • s(3) | 96 • s(3) |
| RC(5) | 11 | 2˙6 • s(1,4) | 2˙6 • s(1,4) | 2˙6 • s(1,4) | 2˙6 • s (1) |

Tabelle 5.1 – Wahl der Umwandlungswartezeit

Sie müssen einen analogen Kontakt auswählen, dessen Signal Sie in ein digitales Signal umwandeln wollen. Denken Sie auch daran, dass, obwohl die 5 analoge Eingänge gebündelt wurden, es möglich ist, nur ein einziges Signal zu konvertieren.

**CHS2:CHS0**: Auswahl der Bits des Kanals, die umgewandelt werden sollen
000 = channel 0, (RA0/AN0)
001 = channel 1, (RA1/AN1)
010 = channel 2, (RA2/AN2)
011 = channel 3, (RA3/AN3)
100 = channel 4, (RA5/AN5)

Das nächste Bit sagt dem Prozessor, dass dieser die Konvertierung durchführen soll. Indem dasselbe Bit getestet wird, kann man überprüfen, ob die Umwandlung schon durchgeführt wurde. Dies kann man sich als Ergebnis ausgeben lassen.
**GO/DONE**: Bit in der Umwandlungsphase

**If ADON=1**
and
If **GO/DONE**=1 - A/D - Umwandlung wird durchgeführt
If **GO/DONE**=0 - A/D - Umwandlung ist angehalten (Dieses Bit wird automatisch 0, wenn die Konvertierung fertig ist)

Das nächste Bit erlaubt diesem Modul das An - und Ausschalten.

**ADON:** aktiviert das A/D – Umwandlungsmodul
**ADON**=1 - der A/D - Wandler ist funktionsbereit
**ADON**=0 - der A/D - Wandler wurde ausgeschaltet und auf Energiesparmodus gesetzt.

Der nächste Datensatz, welcher vorhin genannt wurde, zeigt dem Prozessor, welche Beine des Mikrocontrollers für beide Signale (analog und digital) genutzt werden könne. Er ermöglicht außerdem festzulegen, ob eine Nennspannung existieren soll oder nicht.

#### 5.2.2 ADCON1

| **Bit 7** | | | | | **Bit 2** | **Bit 1** | **Bit 0** |
|---|---|---|---|---|---|---|---|
| -- | -- | -- | -- | -- | PCFG2 | PCFG1 | PCFG0 |

| PCFG2:PCFG0 | RA0 | RA1 | RA2 | RA3 | RA5 | Vref |
|---|---|---|---|---|---|---|
| 000 | A | A | A | A | A | VDD |
| 001 | A | A | A | A | Vref | RA3 |
| 010 | A | A | A | A | A | VDD |
| 011 | A | A | A | A | Vref | RA3 |
| 100 | A | A | D | D | A | VDD |
| 101 | A | A | D | D | Vref | RA3 |
| 11x | D | D | D | D | D | - |

Beispiel einer Analog-Digital Umwandlung im Kanal 0:

Initialization:

```
bsf STATUS,RP0      ; wählt die Bank 0 aus dem STATUS Datensatz
bcf STATUS,RP1      ; wählt die Bank 1 aus dem STATUS Datensatz
clrf ADCON1         ; Festlegen der Analog-Digitalen Eingänge
bcf STATUS,RP0      ; geht zurück zur Datenbank 0
movlw b'01000001'   ; Fosc/8, A/D ist aktiviert, Kanal 0 ist ausgewählt
movwf ADCON0
bsf ADCON,2         ; beginnt die Umwandlung
```

Converte:

```
btfsc ADCON0,2      ; überprüft, ob die Umwandlung vorüber ist
goto Converte       ; falls nicht, überprüfe solange, bis die Umwandlung vorüber ist
movf ADRESH,0       ; der Wert befindet sich im ADRESH Datensatz
```

### 5.3 PWM (Impulsweitenmodulations-) Ausgänge

Das PWM Signal ist eine rechteckige Welle mit konstanter Periodendauer, kann aber ein veränderbares Tastverhältnis aufweisen. Das Tastverhältnis besteht aus dem Teil einer Periode, in dem der Wert der Welle maximal ist. Bild 5.4 zeigt eine Darstellung dieser PWM Welle.

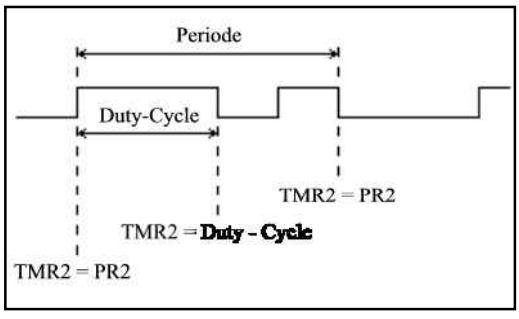



Bild 5.4 PWM Welle

Der gewählte PIC besitzt bereits zwei Module mit PWM Ausgängen. Diese Module sind sehr wichtig, da sie für die Aktivierung der Robotermotoren genutzt werden, und mit ihnen die Geschwindigkeit der einzelnen Räder

verändert werden kann. Diese zwei PWM – Bauteile können eine Auflösung von bis zu 10 Bits haben. Um die beiden Teile programmieren zu können, muss eine **PWM Periodendauer** gewählt werden, welche danach nicht mehr geändert werden sollte. Anschließend wird die Größe des Duty-Cycles gewählt, dessen Wert nach Wunsch änderbar ist.

### 5.4 PWM Periodendauer

Zu Beginn müssen Sie einen Wert für die Periodendauer der **PWM** Welle wählen. Anschließend , muss folgende Gleichung gelöst werden, um **PR2** zu erhalten (dies ist ein Datensatz mit 8 Bits).

**Period of PWM** = [(**PR2** ) + 1)] x 4 x **Tosc** x (**TMR2 prescale value** )

Der **TMR2** ( Anstieg des vorherigen Wertes) kann drei mögliche Werte aufweisen: 1, 4 oder 16. **Tosc** ist das reziprok von der **Fosc** Frequenz.

#### 5.4.1 PWM Duty – Cycle

Wenn der Nutzer während der Programmierung den Wert des Taktverhältnisses (Arbeitszyklus) ändern möchte, muss folgende Gleichung verwendet werden:

**Duty –Cycle do PWM = (CCPR1L:CCP1CON<5:4>)** x **Tosc** x **(TMR2 prescale value)**

### 5.4.2 Maximal Bitanzahl

Je größer die PWM Frequenz ist, desto näher ist die Frequenz der Frequenz des Schwingquarzes. Dieser kann aber nur eine begrenzte Anzahl der Umwandlungsbits nutzen, statt der vollen 10 Bits. Um zu bestimmen, wie viele Bits während der Umwandlung genutzt werden können, reicht es aus, folgende Operation durchzuführen:

$$\text{Maximale Bitanzahl} \quad \frac{\log\left(\frac{F_{osc}}{F_{PWM}}\right)}{\log(2)} \quad \text{Bits}$$

Um eine höhere Auflösung zu erhalten, muss man die Frequenz verringern. Um eine höhere Frequenz zu bekommen, muss die Auflösung heruntergesetzt werden.

Die nächste Tabelle zeigt ein paar Frequenzen, für welche die Werte von **PR2**, die **TMR2 prescale value**– Werte und deren Auflösung bekannt sind.

| PWM Frequenz | 1,22 kHz | 4,88 kHz | 19,53 kHz | 78,12 kHz | 156,3 kHz | 208,3 kHz |
|---|---|---|---|---|---|---|
| Teilerverhältnis (1, 4, 16) | 16 | 4 | 1 | 1 | 1 | 1 |
| Wert von PR2 | 0xFF | 0xFF | 0xFF | 0x3F | 0x1F | 0x17 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Maximale Auflösung (Bits) | 10 | 10 | 10 | 8 | 7 | 5,5 |

Tabelle 5.2 – PWM Perioden und dazugehörige Konfigurationen

In diesem Teil wurden einige neue Datensätze eingeführt, welche noch erklärt werden müssen. Vorhin wurden drei Temporärregister vorgestellt. Hier wird der Temporärregister 2 (**Timer2**) genutzt. Dadurch kann der Wert einer Periode einer Welle abgeschätzt werden, um diesen dann anschließend mit einem Wert zu vergleichen, der durch **PR2** definiert ist.

**PR2**: Festlegen einer PWM Periode

**T2CON**: Festlegen des **TMR2 prescale value**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| -- | -- | -- | -- | -- | TMR2ON | T2CKPS1 | T2CKPS0 |

**TMR2ON** : aktiviert das Temporärregister *Timer 2*

**TMR2ON** = 1 – *Timer 2* aktiv

**TMR2ON** = 0 – *Timer 2* nicht aktiv

**T2CKPS1: T2CKPS0** : Auswahl *der Bits* des TMR2 prescale value

**T2CKPS1: T2CKPS0** = 00 – *prescale value* ist gleich 1

**T2CKPS1: T2CKPS0** = 01 – *prescale value* ist gleich 4

**T2CKPS1: T2CKPS0** = 1x – *prescale value* ist gleich 16

**CCPR1L und CCP1CON<5:4>:** Festlegung desTaktverhältnisses

**CCPR1L** Datensatz korrespondiert mit den wichtigsten 8 *Bits*, und der **CCP1CON<5:4>** Datensatz mit den 2 weniger wichtigen *Bits*.

### 5.4.3 Schritte für die Initialisierung der PWM

1. Initialisieren Sie, dass die **PWM** Periode in den **PR2** Datensatz geschrieben wird
2. Initialisieren Sie, dass der **PWM** *Duty - Cycle* in den **CCPR1L** und **CCP1CON<5:4>** Datensatz geschrieben wird

3. Transformieren Sie die **CCP1** und **CCP2** *Bits* in Ausgänge und schreiben Sie eine Null in die *Bits* 1 und 2 von **TrisC**

4. Setzen Sie den **TMR2 prescale value** fest und aktivieren Sie den *Timer2*, welcher seine Werte in **T2CON** schreiben soll

5. Bereiten Sie die **CCP1CON** - und **CCP2CON** – Module für die **PWM** Operationen vor, indem der binäre Wert b'00001111' in beide Datensätze geschrieben wird

## Beispiel einer PWM Programmierung:

```
bsf STATUS,RPO          ; gehe in die Bank 0, um PR2 zu festzulegen
movlw .255              ; Festlegen des PR2 Datensatz ( entspricht der Periodendauer )
movwf PR2
bcf STATUS,RPO          ; gehe zurück zu Bank 0
movlw .12               ; setze die 8 Bits von den CCP1CON und CCP2CON Datensätzen fest
movwf CCP1CON           ; mit dem Wert 12 (Festlegung des Duty-Cycle)
movwf CCP2CON
movlw .5                ; aktiviere Timer2 und nutze dazu den vorherigen Anstieg von 1
movwf T2CON
movf Andar_Esquerda,0
movwf CCPR2L
movf Andar_Direita,0
movwf CCPR1L
```

### 5.5 Verwendung von Tabellen

In verschiedenen Situationen, in denen der Programmschwerpunkt verändert werden soll ist die Nutzung von Tabellen mit vordefinierten Werten sehr wichtig. Im Fall unseres Roboters werden Tabellen benötigt, um dem Roboter verschiedene Verhaltensweisen, welche er in verschiedenen Situationen aufweisen sollte, vorzuschreiben.

Zum Beispiel, wenn der Roboter weiß, dass er genau über einer auf den Boden gemalten Linie fährt, könnte er schneller werden. Ein anderes Beispiel sollen folgende drei Situationen zeigen: Der Roboter hat auf seiner Seite eine Linie, oder die Linie befindet sich genau in der Mitte, oder die Linie befindet sich genau auf der anderen Seite des Roboters. Wenn dies der Fall wäre, ist es möglich, diese Situationen in einer Tabelle zu definieren, und in zwei anderen Tabellen werden die PWM Werte eingetragen, welche zu jedem der beiden Motoren geschickt werden müssen.
Beispiel:

Pista:

```
        addwf PCL, 1        ; Tabelle, welche die drei möglichen Robotersituationen darstellt
        retlw b“11100000“
        retlw b“00111000“
        retlw b“00000111“

MotorDireito:

        addwf PCL, 1        ; Tabelle, welche die PWM Werte für den Motors darstellt
        retlw .10           ; von der rechten Seite zu einer der drei Situationen
        retlw .200
        retlw .255

MotorEsquerdo:

        addwf PCL, 1        ; Tabelle, welche die PWM Werte für den Motors darstellt
        retlw .255          ; von der linken Seite zu einer der drei Situationen
        retlw .200
        retlw .10
```

Um jede dieser Tabellen aufzurufen reicht es, einen Aufruf durchzuführen, indem man einen Wert in den (w) Datensatz einführt. Wenn w=0 mit dem ersten Element der Tabelle übereinstimmt, dann stimmt w=1 mit dem zweiten Element der Tabelle überein, usw. .

```
Movlw .2                ; w = 2
call MotorDireito       ; Ruft das MotorDireito – Programm ab und gibt einen Wert zurück

----------              ; w = 255
```

## 5.6 Interruptionen (Unterbrechungen)

In verschieden Fällen ist es unvermeidbar, dass einige Ereignisse auftreten, welche sofort beim Erscheinen bearbeitet werden müssen. Für solch einen Fall kann der Nutzer **Interruptserviceroutinen** erstellen, welche in erster Stelle außerhalb des normalen Programmablaufs abgearbeitet werden. Diese Unterbrechungen können von Signalen in einigen Mikrocontroller-Kontakten erzeugt werden, oder sie können durch bedingte Veränderungen einiger Datensätze entstehen. Das folgende Programm zeigt, wie die Aktivierung einer Unterbrechung jedes Mal, wenn **Timer2** gleich dem Wert des **PR2** Datensatzes ist, erfolgt.

```
; Initialisierung der Interruption - Datensätze

        movlw   b’11000000’
        movwf   INTCON
        bsf     STATUS,RP0
        bcf     STATUS,RP1
        movlw   b’00000010’
        movwf   PIE1
        bcf     STATUS,RP0
        bcf     STATUS,RP1
```

### 5.6.1 INTCON Datensatz

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| GIE | PEIE | TOIE | INTE | RBIE | TOIF | INTF | RBIF |

Wie in dem vorherigen Programm gezeigt, haben nur die zwei bedeutenderen Bits den Wert 1 erhalten. Wenn das **GIE** Bit den Wert 1 annimmt, aktiviert es die Interruptionen, und das **PEIE** Bit aktiviert die Interruptionen

der Art des Timer2. Die anderen nicht erwähnten Bits ermöglichen andere Arten von Interruptionen, Signale anderer Timer und externe Signale.

### 5.6.2 PIE1 Datensatz

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE |

In diesem Fall ist **TMR2IE** aktiv, und erzeugt eine Interruption jedes Mal, wenn *Timer2* gleich dem **PR2** Datensatz ist.

```
; Interruptionenbearbeitung

interruptions:

        movwf dabliu            ; zuvor müssen alle Informationen gespeichert werden, als erstes die Daten ...
        swapf STATUS,w          ; ... aus w, und anschließend die Informationen des Datensatzes STATUS
        movwf statos            ; Der dabliu und der statos Datensatz müssen zu Beginn des ...
                                ; ... Programms definiert werden
        bcf STATUS,RP0
        btfsc PRI1, TMR2IF      ; Überprüfen, ob die Interruption, welche sich auf  TMR2 bezieht, die ...
                                ; ... timer2 Interruption war
        goto testa_tmr2         ; gehe zu testa_tmr2
        bcf PIR1,0              ; wenn es sich um eine andere Interruption handelt, gehe zurück zum ...
                                ; ... Hauptprogramm
        goto fim_int

testa_tmr2:                     ; Unterprogramm zur Bearbeitung der Interruption durch Timer2
        bcf PIR1,0              ; blockiert die Interruption
        goto PWM
PWM:
        movwf Anda_esquerda,0   ; führt die gewünschte Bearbeitung durch
        movwf CCPR2L            ; tauscht zum Beispiel die PWM1 und die PWM2
        movwf Anda_direita,0
        movwf CCPR1L
        goto fim_int

fim_int:
```

```
        bcf PIR1, TMR2IF        ; löscht den Inhalt des Bits, da bereits abgearbeitet
        bsf STATUS,RP0
        bsf PIE1, TMR2IE        ; aktiviert die Interruptionen nochmals
        bcf STATUS,RP0
        movlw b'11000000'       ; aktiviert die Interruptionen nochmals
        movwf INTCON

        swapf statos, w         ; geht zurück zu den alten Werten von STATUS und w
        movwf STATUS
        swapf dabliu, w
retfie                          ; geht zum Startprogramm zurück
```

**5.6.3 PIR1 Datensatz**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE |

Dieser Datensatz erlaubt zu testen, welcher der Interruptionen durch **PIE1** aktiviert wurden. In diesem Fall wird der Test mit **TMR2IF** durchgeführt, da dieser mit der aktiven Interruption von *Timer2* übereinstimmt. Eine Interruption ist immer nachweisbar, und nach der Bearbeitung muss dieses Bit den Wert 0 übernehmen.

## 5.7 Struktur eines Programms für die PI-Regelung (PIC)

Die Programmierung eines Mikrocontrollers muss einer festgelegten Reihenfolge folgen. Diese Folge erlaubt nicht nur eine ordentliche Bearbeitung, sondern auch eine leichte Lesbarkeit. In diesem Abschnitt wird ein Beispiel für eine Programmorganisation gezeigt.

```
; Beginn des Programms

; Konfiguration des Mikrocontrollers und Auflisten der Bibliotheken, welche genutzt werden sollen

List P=16F876

Include <P16F876.inc>

; Aufbewahrung des Datenspeichers in Hilfsvariablen

sensor1 EQU 0x20        ; Der Datenspeicher mit der Position 0x20 ist reserviert als sensor1, ...
sensor2 EQU 0x21        ; ... die Position 0x21 ist für sensor2 reserviert, usw.
sensor3 EQU 0x22        ; Der erste Wert muss immer mit einer Adresse beginnen, die gleich …
```

```
                    ; ... oder höherstehender als 0x20 ist
sensor4 EQU 0x23
sensor5 EQU 0x24
sensor6 EQU 0x25
sensor7 EQU 0x26
sensor8 EQU 0x27
contador EQU 0x28
AUX EQU 0x29
pista EQU 0x2A      ; 0x2A legt den Wert mit sechs Dezimalstellen nach der 0x29 fest

; Programmstart
org 0x00            ; definieren der Programmposition 0
goto start          ; Sprung zu einer beliebigen Position des Programmanfangs
org 0x04            ; reservierter Bereich für Interruptionen
call interrupções   ; ruft ein Unterprogramm zur Abarbeitung von Interruptionen auf
return              ; geht zurück zum Hauptprogramm
org 0x10            ; Bereich von Tabellen
niveis:
    addwf PCL, 1
    retlw .50
    retlw .50
    retlw .50
    retlw .50
    retlw .50
    retlw .50
    retlw .50
    retlw .50

;Hauptprogrammablauf

    org 0x30        ; legt die vom Nutzer gewählte Anfangsposition fest
Start:
```

```
        call In_Out     ; ruft das Programm auf, welches die Anschlüsse festlegt
Ciclo:                  ; Zyklus der sich immer selbst wiederholt
        ----
        ----
        Any call
        ----
        goto Ciclo      ; wiederholt den Ablauf
```

**; Unterprogramme, welche während des Hauptablaufes aufgerufen werden**

```
qualquer:
        ------
        ------
        ------
return                  ; kehrt zu der nächsten Zeile zurück, von wo aus er aufgerufen wurde
```

**; Programm zur Abarbeitung der Interruption**

```
interruptions:
        ------
        ------
        ------
retfie                  ; kehrt zu der Zeile zurück, welche durchgeführt wurde, als die Interruption auftrat
```

**; Programm zur Input/Output Konfiguration**

```
In_Out:
        –
        –
        –
        –
return

end                     ; es gibt keinen weiteren Programmcode
```

# 6 Programmierung

In diesem Kapitel schenken wir dem Steuerungssystem der zwei Servomotoren des Infrarot-Boards die Aufmerksamkeit.

## 6.1 Programmierung der Steuerung des Infrarot-Board 's

Durch die Eigenschaften des Analog/Digital-Wandlers vom genutzten Mikrocontroller ist eine kurze Zeit zum Laden der Umwandlungskapazität wichtig, bevor mit der Umwandlung begonnen wird.

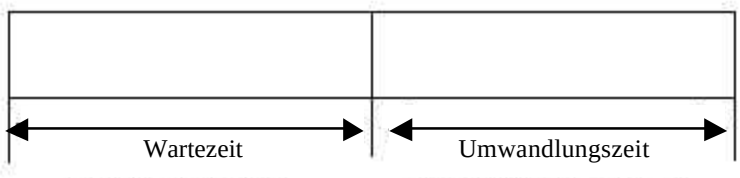



Bild 6.1 – Wartezeit und Umwandlungszeit

In unserem Programmierbeispiel zum Beispiel, wird eine Wartezeit von 3 ms für alle *LED's* genutzt. Die Umwandlungszeit wird durch den Mikrocontroller reguliert. Die Zeit bis zum Leuchten der LED's hängt von der Summe der beiden Zeiten ab (Wartezeit + Umwandlungszeit). Bild 6.2 zeigt die gewünschte zeitliche Struktur der LED's.

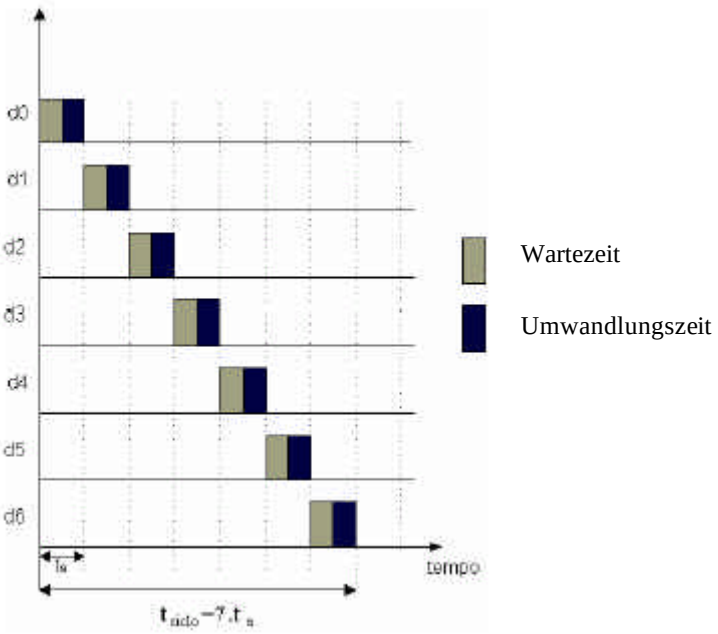



Bild 6.2 – zeitliche Strukturierung

Um **S0, S1** und **S2** zu beeinflussen, wurde die Kontakte 3,4 und 5 von **Anschluss C** genutzt. Um die **Steuerung** von **Z** anzuschalten wurde der Kontakt 0 von **Anschluss C** genutzt.

Die analog erhaltenen Werte werden zum ID-AS002 – Board geschickt, welches die Bündelung der 7 Eingänge zum Infrarot-Board übernimmt. Dabei werden die Bits 3, 4 und 5 des Anschlusses C verwendet. Diese Bündelung bewirkt, dass nur ein analoger Kanal genutzt werden muss, und zwar der Kontakt 0 an Anschluss A.

| | Bit 5 | Bit 4 | Bit 3 |
|---|---|---|---|
| d0 | 0 | 0 | 0 |
| d1 | 0 | 0 | 1 |
| d2 | 0 | 1 | 0 |
| d3 | 0 | 1 | 1 |
| d4 | 1 | 0 | 0 |
| d5 | 1 | 0 | 1 |
| d6 | 1 | 1 | 0 |

Tabelle 6.1 – vorgeschriebene Infrarot – Sende- und Empfänger – Konfiguration

Programm:

```
LIST P=16F876

include <P16F876.inc>

Contadorleds EQU 0x24
Contador EQU 0x2C
Pista EQU 0x29
…

org 0x00

goto start

org 0x24
call interrupt
```

```
        return

org 0x10                ; Anfang des Tabellenbereiches


limiar:                 ; Tabelle zum Vergleich der TresHalter mit den
analogen Werten

        addwf PCL,1
        retlw.70
        retlw.78
        retlw.55
        retlw.53
        retlw.30
        retlw.52
        retlw.48

org 0x30                ; Anfang des Programmzyklus 's


start:

        call In_Out     ; Initialisierung der Anschlüsse

        movlw b'00110001'       ; Start von Timer 1 mit den Wartezeiten, usw.
        movwf T1CON

        call segue_linha:

goto start


segue_linha:

        bsf PORTC,0             ; Bit Pc7 wird zu 1, Anschalten der
Infrarotstahlen
        bcf PORTC,3             ; Bits Pc4, Pc5 und Pc6 bekommen die 0
zugeordnet
        bcf PORTC,4
        bcf PORTC,5
        clrf Contadorleds       ; zähle die Zahl der bisher ausgelesenen LED's
        clrf Pista      ; beinhaltet den binären Wert aller LED's
```

```
inicioconversao:

        movf TMR1L,0            ; Anfang der Wartezeit
        btfss STATUS,2
        goto $-2

        movlw 0xFF              ; Wartezeit
        subwf TMR1L,0
        btfss STATUS,2
        goto $-3

        bsf ADCON0,2            ; um Umwandeln zu können
        btfsc ADCON0,2          ; Umwandlungszeit
        goto $-1


debug:

        movf Contadorleds,0     ; Zahl der gelesenen LED's
        call limiar             ; Vergleich mit einem Treshalter (in der Tabelle
definiert)
        subwf ADRESH,0
        btfsc STATUS,0
        goto preto              ; Falls der Wert unter dem TresHalter oder Schwarz
liegt
        rlf Pista               ; ansonsten, oder Weiß
        bsf Pista,0
        goto Contadores


preto:

        rlf Pista
        bcf Pista,0
        goto Contadores


Contadores:

        incf Contadorleds,1     ; liest die nächsten IV Paare
        btfsc Contadorledes,3   ; testet, ob sie alle gelesen sind
        return
```

```
        movlw b'00001000'       ; erhöht Pc4, Pc5, Pc6
        addwf PORTC,1

        goto inicioconversao
In_Out:

        bsf STATUS, RP0
        bcf STATUS, RP1
        bsf TRISA,0             ; Am Anschluss A wird der Kontakt 1 als
Eingang festgelegt

        clfr PORTC          ; Am Anschluss C werden alle Kontakte als Ausgänge
festgelegt
        bcf STATUS,RP0

        return

end
```

### 6.2 Programm zum Steuern der zwei Servomotoren

Die verwendeten Motoren sind veränderte Servomotoren. Das bedeutet, dass die Servomotoren, welche normalerweise für Winkelsteuerung genutzt werden, hier so verändert wurden, dass die Geschwindigkeit gesteuert werden kann. Sie werden über ein **PWM** Signal gesteuert, wie Bild 6.3 zeigt.

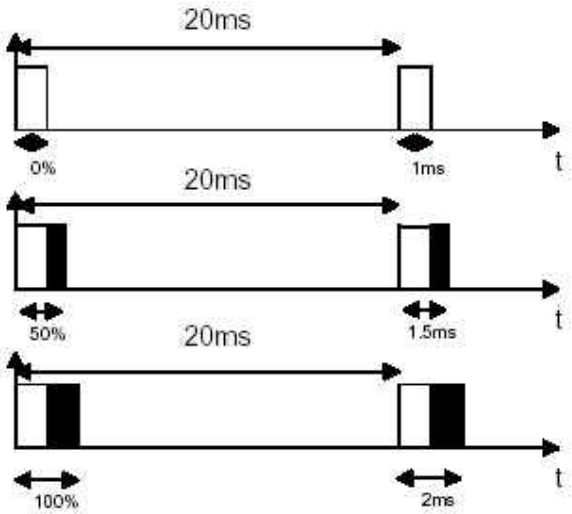



Bild 6.3 – Nutzung der PWM - Welle in Servomotoren

Wie leicht überprüft werden kann, wird die Periodendauer dieser Welle 20 ms groß sein. Diese Periodendauer muss völlig beendet werden, da davon das ordentliche Arbeiten dieser Motorelektronik abhängt. Die **PWM** – Welle lässt sich in drei verschiedene Teile zerlegen: Im ersten Bereich hat die Welle für eine Millisekunde einen Wert von 5 V; der zweite Bereich, Duty-Cycle genannt, da die Geschwindigkeit nur von diesem Bereich abhängig ist, stimmt mit einem Duty-Cycle eine Millisekunde lang 100% überein; im letzten Bereich wird das Signal für 18 ms auf Null gesetzt. Noch mal zurück zum Duty-Cycle, wenn dieser bei 0% ist, wird das zugehörige Rad mit maximaler Geschwindigkeit zurückrollen. Wenn der Duty-Cycle bei 50 % liegt, dann wird das Rad sich nicht drehen, und bei 100 % wird das Rad mit maximaler Geschwindigkeit vorwärts rollen.

Das Programm, welches ermöglicht, dass das Signal zu allen beiden Motoren geschickt wird, teilt dieses Signal in zwanzig Signale mit jeweils einer Länge von 1 ms, und hat auch den Wert 1 in der ersten Millisekunde. Das zweite wird mit einem festgelegten Duty-Cycle geändert und wechselt auf Null für die restlichen 18 ms. Um das Signal in 1 ms –Intervalle zu teilen, wurde eine PWM von 1 ms Länge programmiert. Die Zählenden sind immer bei 1 ms, dort wird eine Interruption erzeugt, welche, falls im Interruption – Programm formuliert, den nächsten Duty-Cycle auswählt (100% oder 0 %). Das nächste Programm zeigt, wie es ermöglicht wird, diese Art von Signal zu erzeugen.

```
LIST P=16F876

include <P16F876.inc>

dabliu EQU 0x20                 ; Hilfsvariable, welche W beinhaltet

AUXSTATUS EQU 0x21              ; Hilfsvariable, welche STATUS
beinhaltet

ContadorPWM EQU 0x22            ; PWM - Zähler für 20 Zyklen

Anda_direita EQU 0x23           ; Variable, welche den nächsten PWM – Wert
des rechten Motors beinhaltet

Anda_esquerda EQU 0x24          ; Variable, welche den nächsten PWM – Wert
des linken Motors beinhaltet
```

```
org 0
goto 0x46
org 0x04
        call interrupt              ; Aufruf der Interruptionen
        return
org 0x46                    ; Anfang der Programmgestaltung
        call In_Out         ; Initialisierung der Anschlüsse
        movlw b'11000000'
        movwf INTCON        ; Aktiviert die Interruptionen
        bcf STATUS,6
        bsf STATUS,5
        movlw b'00000010'   ; Konfiguration der Interruptionen
        movwf PIE1
        movlw .249          ; Konfiguration der PWM – Periodendauer
        movwf PR2
        bcf STATUS,5
        movlw b'00000101'   ; Prescale Wert
        movwf T2CON
        movlw b'00001111'   ; PWM – Schwingmodus - Gestaltung von
Kontakt 3 am Anschluss C
        movwf CCP1CON
```

```
        movlw b'00001111'       ; PWM – Schwingmodus - Gestaltung von
Kontakt 2 am Anschluss C

        movwf CCP2CON

        movlw b'00110001'

        movwf T1CON

        movlw .19               ; Festlegung der Anzahl von 1 ms – Zyklen, 19 + 1

        movwf ContadorPWM


start:                          ; Werte, welche zu den PWM's gesendet werden
könnten

        movlw .200

        movwf Anda_esquerda

        movlw .50

        movwf Anda_direita

        goto start


interrupt:

        movwf dabliu            ; alle relevanten Informationen müssen vorher
gespeichert werden

        swapf STATUS,w ; Als Erstes wird die Information aus w, und danach
aus dem Datensatz von ..

        movwf AUXSTATUS     ; ... STATUS abgespeichert. Der dabliu –
        Datensatz muss am Anfang des ...

                            ; ... Programms definiert werden.

        btfsc   PIR1,TMR2IF     ; es wird überprüft, ob die Interruption von
Timer 2 ausgelöst wurde
```

```
        goto    testa_tmr2

        bcf PIR1, 0

        goto fim_int


test_tmr2:

        bcf PIR,1               ; deaktiviert die Interruptionen

        movlw .19               ; überprüft, in welchem Zyklus er sich gerade befindet

        subwf ContadorPWM,0

        btfsc STATUS,2

        goto PWM_100            ; falls er sich im ersten Zyklus befindet, bleibt
die PWM bei 100%

        movlw .18

        subwf ContadorPWM,0     ; falls er sich im zweiten Zyklus befindet, geht er
zum Programmieren des ...

                                ; ... PWM über

        btfsc STATUS,2          ; andernfalls ist die PWM bei 0%

        goto PWM_PWM

        movf ContadorPWM,0

        btfsc STATUS,2
        goto Novacontagem

        clrf CCPR1L

        clrf CCPR2L

        decf ContadorPWM
```

```
        goto fim_int

PWM_100:            ; im ersten Zyklus müssen beide PWMs auf 100 %
gesetzt werden

        movlw .255

        movwf CCPR1L

        movwf CCPR2L

        decf ContadorPWM

        goto fim_int

PWM_PWM:            ; im zweiten Zyklus wird der, in die Variablen
                      geschriebene PWM - Wert ...
                    ; ... genutzt

        movf Anda esquerda, 0   ; linkerer Motorenwert

        movwf CCPR2L

        movf Anda_direita,0     ; und der rechte Motorenwert

        movwf CCPR1L

        decf ContadorPWM

        goto fim_int

Novacontagem:       ; wenn die 20 Zyklen vorbei sein sollten, gehe
zurück zum Ersten

        movlw .19

        movwf ContadorPWM

        goto fim_int
```

```
fim_int:                    ; aktiviert nochmals die Interruptionen
        movlw b'11000000'
        movwf INTCON
        swapf AUXSTATUS,w   ; geht zu den alten Werten von STATUS und w zurück
        movwf STATUS
        swapfdabliu,w
retfie
In_Out:
        bsf STATUS, RP0
        bcf STATUS, RP1
        clrf TRISC          ; Anschluss C Konfiguration als Ausgänge
        bcf STATUS, RP0
        return
end
```

## 7. Experimentierbeispiel

In diesem Kapitel soll an einem kleinen Testbeispiel das Können des Roboters verdeutlicht werden. Die nötigen Schritte zum Programmieren des Roboters sollen im Folgenden dargelegt werden. Dazu wird das ID-PROG002 – Programmier-Board verwendet.

Dem Bastelsatz ist eine CD beigefügt, welche ein Testprogramm beinhaltet, not builded (.asm), welche der Nutzer zur Überprüfung einer ordentlichen Funktionsweise des Roboters nutzen kann. Dieses Beispiel wird auf den Roboteraufbau angewandt, wie er in den vorherigen Kapiteln beschrieben wurde. Im Endeffekt sollte er so sein, wie der, welcher in Bild 4.22 gezeigt wurde. Nach seiner Programmierung wird der Roboter einer auf den Boden (dunkel oder hell) aufgemalten Linie (hell oder dunkel) folgen können und einer

Kollision mit einem anderen Objekt, unter Benutzung der *Mikro-Schalter*, aus dem Weg gehen können.

Kopieren Sie nun den Inhalt der CD in ein Verzeichnis des Computers. Führen Sie dann das Programm „Loaderstp.exe“ aus, um die Applikation zu installieren, welche dann das Downloaden des Programms auf den Roboter erlaubt.

Installieren Sie, auf einen PC, das Programm Mplab von ‚Microchip’. Nutzen Sie dazu folgenden Link, um diese Software downzuloaden:

http://www.microchip.com/10/tools/picmicro/devenv/mplabi/software/v540/index.htm

Nach der Installation des Programms kann das Programm Mplab ausgeführt werden. Öffnen Sie das Projekt „Circ3.pjt“ (Project – open Project), Bild 7.1. Erstellung des Projektes (Project – Build All).

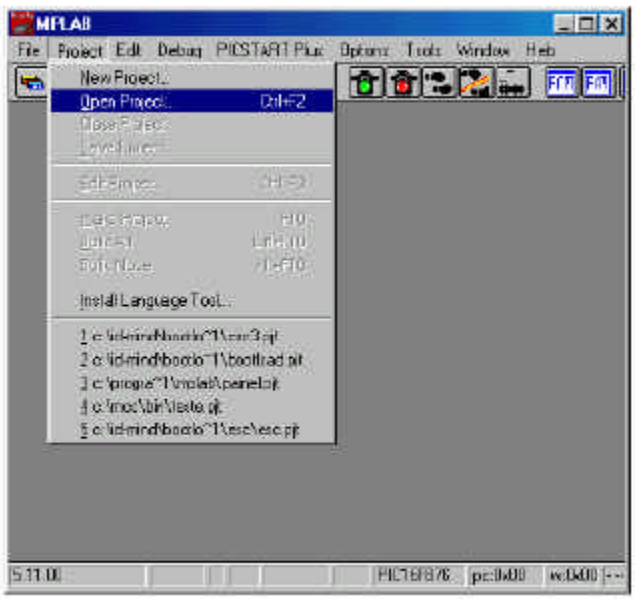



Bild 7.1 – Mplab vom Mikrochip

Nach der Erstellung ist nun im Verzeichnis die Datei „CIRC3.HEX“ erzeugt worden.

Setzen Sie das ID-PROG002 Programmier-Board auf den Roboter – auf die Oberseite des ID-PIC002 – Boards. Verbinden Sie den männlichen Anschlussteil des seriellen Kabels mit dem Programmier-Board und den weiblichen Anschluss mit einem seriellen Eingang des PCs (Bild 7.2).

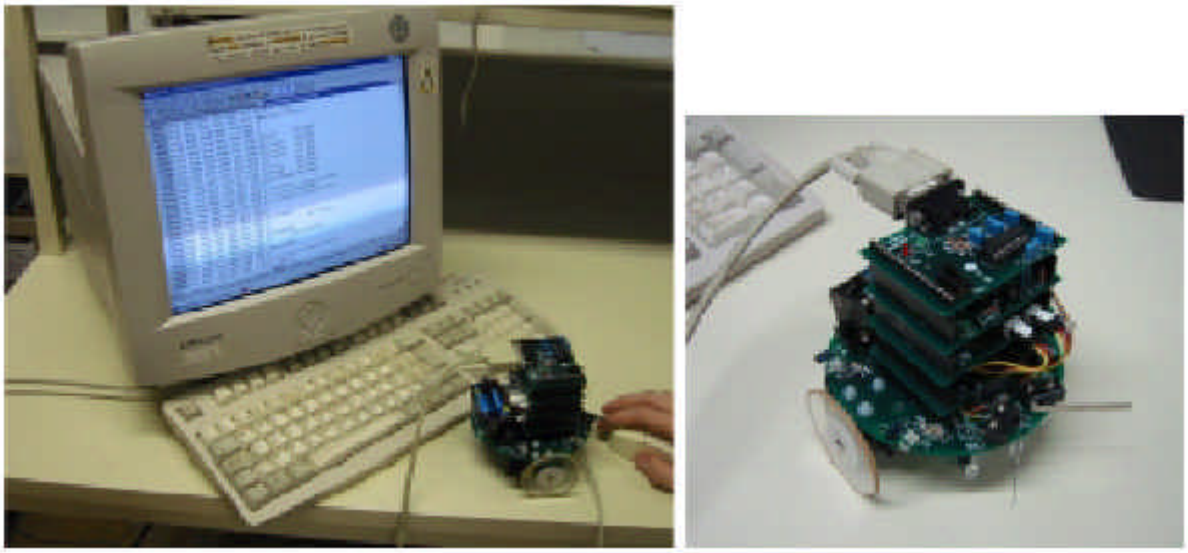


Bild 7.2 – Aufbau zum Programmieren

Nutzen Sie nun das Programm **Loader**, von IdMind geliefert, um das erstellte Programm auf den Mikrocontroller des Roboters zu laden (.hex). Dafür muss das das Programm „Loader.exe“ ausgeführt werden, welches aus der Installation der Datei „Loaderstp.exe“ resultierte. Legen Sie im Programm fest, welche seriellen Anschlüsse von Ihnen verwendet werden („com1“ oder „com2“).

Öffne die Datei „CIRC3.HEX“, Bild 7.3 .

Bild 7.3 – Programmier-Board Schnittstelle

Nach dem der Roboter ausgeschaltet wurde ( Unterbrecher des ID-BASEIR002 – Boards auf „OFF“ gestellt), schalten Sie den Unterbrecher des ID-PROG002 –

Boards auf „ON“. Schalten Sie nun auch den Unterbrecher des ID-BASEIR002 – Boards auch auf „ON“.

Drücken Sie auf „Send“ im Loader Programm und das Programm wird zum Mikrocontroller übertragen werden (Bild 7.4).

Bild 7.4 – Programmübertragung

Wenn die Nachricht „Success“ erscheint, kann der Roboter ausgekuppelt werden (Interruptor vom ID-BASEIR002 – Board auf „OFF“ schalten), und der Interruptor vom Programmier-Board kann ebenfalls abgetrennt werden. Entfernen Sie das serielle Kabel vom Programmier-Board.

Um den Roboter in Bewegung zu setzen, reicht es jetzt aus, den Interruptor vom ID-BASEIR002 – Board zu betätigen. Der Roboter kann die Programme ausführen, egal, ob das Programmier-Board aufgesteckt ist oder nicht. Um sie auszuführen, während das Programmier-Board aufgesteckt ist, muss dessen Interruptor auf „OFF“ gestellt werden. Denken Sie daran, dass die Ausführung der Programme mit diesem Board zu einem kleinen Anstieg des Energieverbrauchs führen wird.

### 7.1 Fehlermeldungen

Während das Programm an den Roboter übertragen wird, kann es passieren, dass einige Fehler wegen falscher Prozeduren auftreten. Daher sind als nächstes einige gewöhnliche Fehler aufgeführt, und die dazu gehörige Fehlerursache.

**Fehler:** „ERROR: Serial Port Selected not Enable!!!“ ( - Fehler: der gewählte serielle Anschluss ist nicht bereit!!!)

**Ursache**: An den gewählte seriellen Anschluss ist bereits ein Peripheriegerät angeschlossen (z.B. die Maus).

**Lösung**: Wählen Sie den anderen „com“ im Programm

**Fehler**: „ERROR: The Pic does not respond “ ( - Fehler: Der PIC antwortet nicht)

**Ursache**: Der Roboter ist nicht auf „ON“ gestellt

Der Steuerungs-Board - Interrupter ist nicht auf „ON“ geschaltet

Der serielle Anschluss, welcher angeschlossen ist, ist nicht der, der definiert wurde

**Lösung**: Wenn der Roboter ausgeschaltet sein sollte ( ID-BASEIR002 – Board auf „OFF“), schalten Sie den Interruptor vom ID-PROG002 auf „ON“, und anschließend den Interruptor des ID-BASEIR002 – Boards auch auf „ON“.
Kontrollieren Sie den seriellen Anschluss oder wähle das andere „com“ im Programm.

**Fehler**: „ERROR: The Pic can’t be programmed“ ( - Fehler: Der PIC kann nicht programmiert werden)

**Ursache**: Die Computerdaten sind nicht korrekt

Der PIC kann die Computerdaten nicht durchlaufen lassen

**Lösung**: Erstellen Sie die Daten nochmals

Vergrößern Sie das Programm „Delay“

**Fehler**: „ERROR: Stop by USER!!!“ ( - Fehler: Angehalten durch Nutzer!!!)

**Ursache**: Der Nutzer hat den Download-Vorgang durch das Drücken der Stop-Taste angehalten

**Lösung**: Schicken Sie das Programm „Send“ erneut

## Anhang A – Schaltpläne

### A1) Steuereinheiten (ID-PIC002, ID-DS002 und ID-AS002)

Bild A.1 zeigt das angewandte Schaltbild des Steuerungssatzes

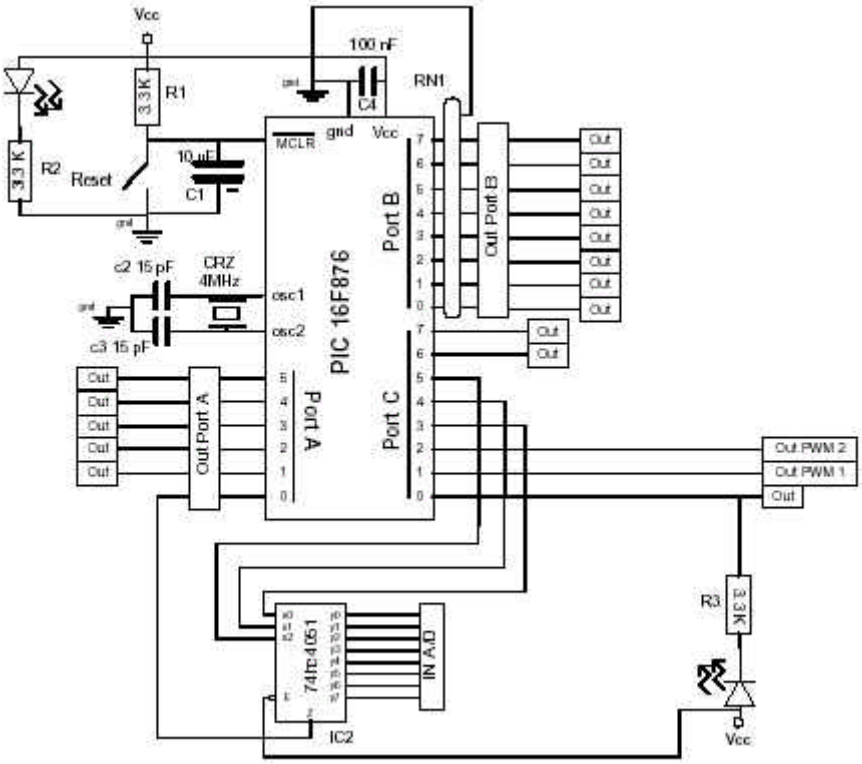



Bild A.1 – Steuer-Board – Schaltbild

### A2) Infrarot – Board

Das Schaltbild des Infrarot-Boards (ID_BASEIR002) wird in Bild A.2 dargestellt.

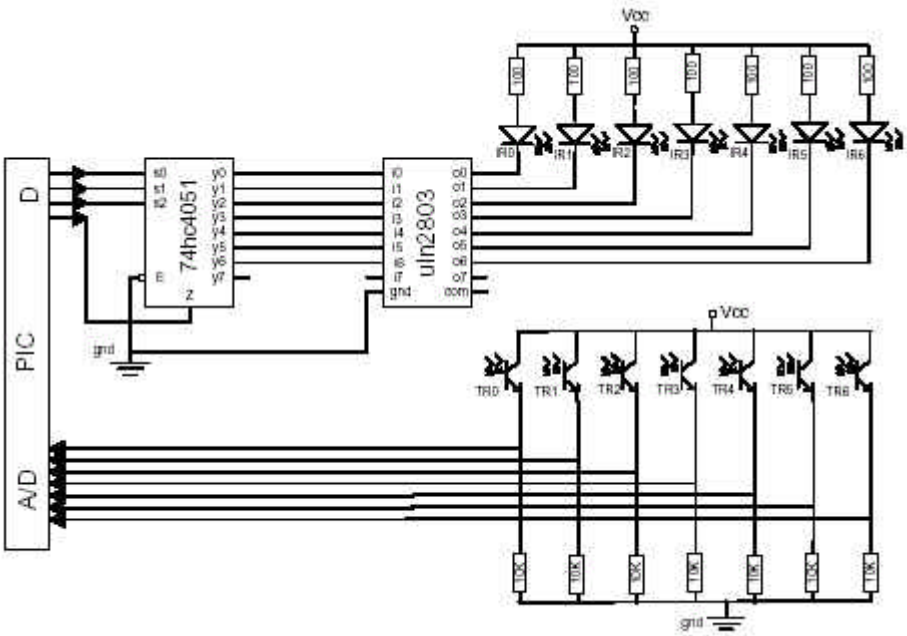



Bild A.2 – Infrarot – Board – Schaltbild

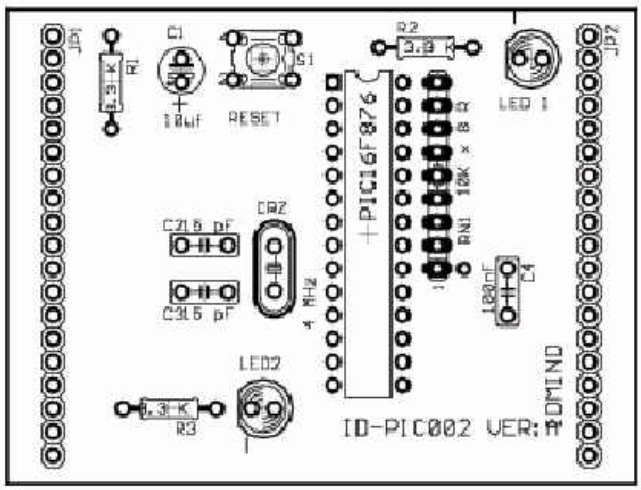
In diesem Schaltkreis werden vier digitale Ausgänge des Anschlusses C genutzt. Die Bits 3, 4 und 5 von diesem Anschluss sind verbunden mit „S0“, „S1“ und „S2“, und sie definieren, welche der LED’s als Emitter fungieren. Der vierte Ausgang, Bit 0 von Anschluss A, ist mit dem Ausgang „z“ verbunden. Dieser erlaubt zu entscheiden, ob die Emitter verbunden sein sollen, oder nicht.

Die sieben analogen Ausgänge, welche vom Infrarot – Empfänger kommen, sind mit den analogen Eingängen des Mikrocontrollers verbunden. Diese Verbindung wird über die Schnittstelle zum analogen Board gemacht, welches eine Bündelung dieser sieben Eingänge ermöglicht, und jedes einzelne von ihnen lesen kann, obwohl nur ein analoger Eingang des Mikrocontrollers genutzt wird.

## Anhang B – Bauteile und Boards

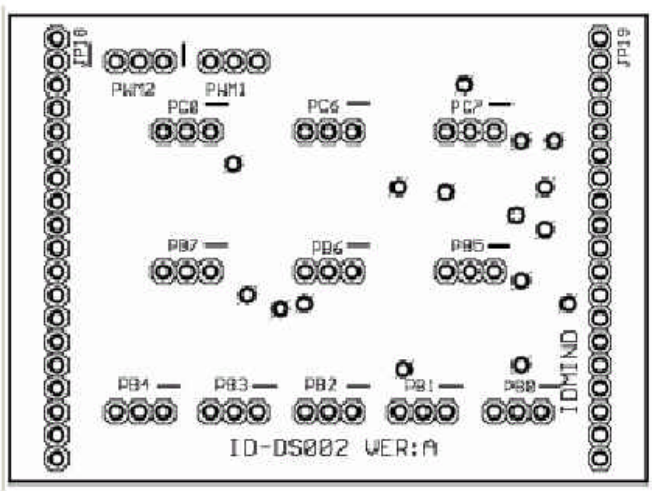
### B1) Steuerungsboard – ID -PIC002

| Benennung | Anzahl |
|---|---|
| PIC 16F876 | 1 |
| Sockel 0,3” 28 way | 1 |
| 100 nF Kondensator | 1 |
| 3,3 k• Widerstand | 3 |
| 8 * 1 k• Widerstand (Widerstandspacket) | 1 |
| Grüne LED 3 mm | 1 |
| Rote LED 3 mm | 1 |
| 15 pF Keramik - Kondensator | 2 |
| 10 • F/50V Elektrolyt-Kondensator | 1 |
| Schalter (Reset) | 1 |
| Cr. 4 MHz | 1 |
| 19-beinige Stiftleiste | 2 |
| 19-beinige Buchsenleiste | 2 |



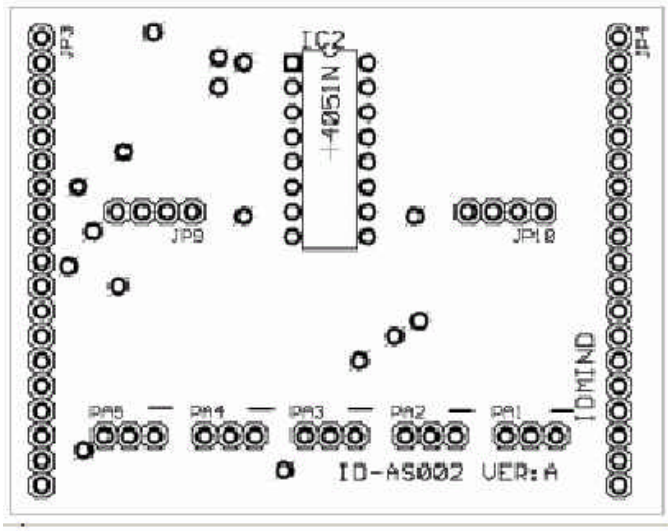

**B2) Digitales Schnittstellen-Board ID-DS002**

| Benennung | Anzahl |
|---|---|
| 19-beinige Stiftleiste | 2 |
| 19-beinige Buchsenleiste | 2 |
| 3-beiniger Verbindungsstift, um 90° gewinkelt | 13 |



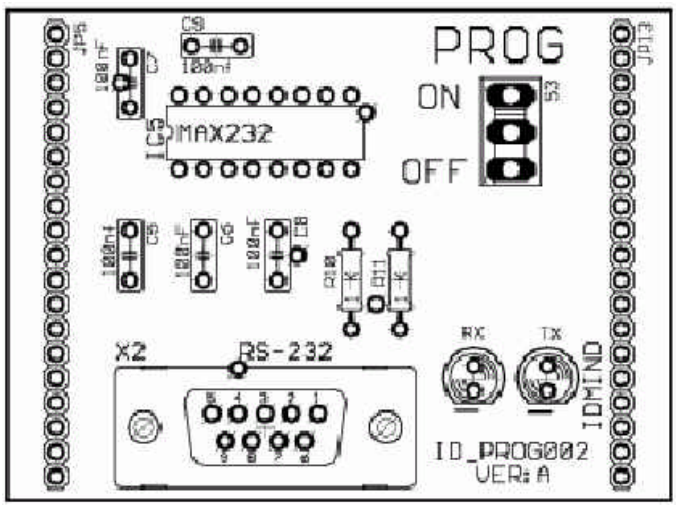
**B3) Analoges Schnittstellen-Board ID-AS002**

| Benennung | Anzahl |
|---|---|
| 4-beinige Buchsenleisten | 2 |
| 19-beinige Buchsenleisten | 2 |
| 3-beiniger Verbindungsstift, um 90° gewinkelt | 5 |
| 16-beiniger Sockel | 1 |
| 74HC4051N | 1 |



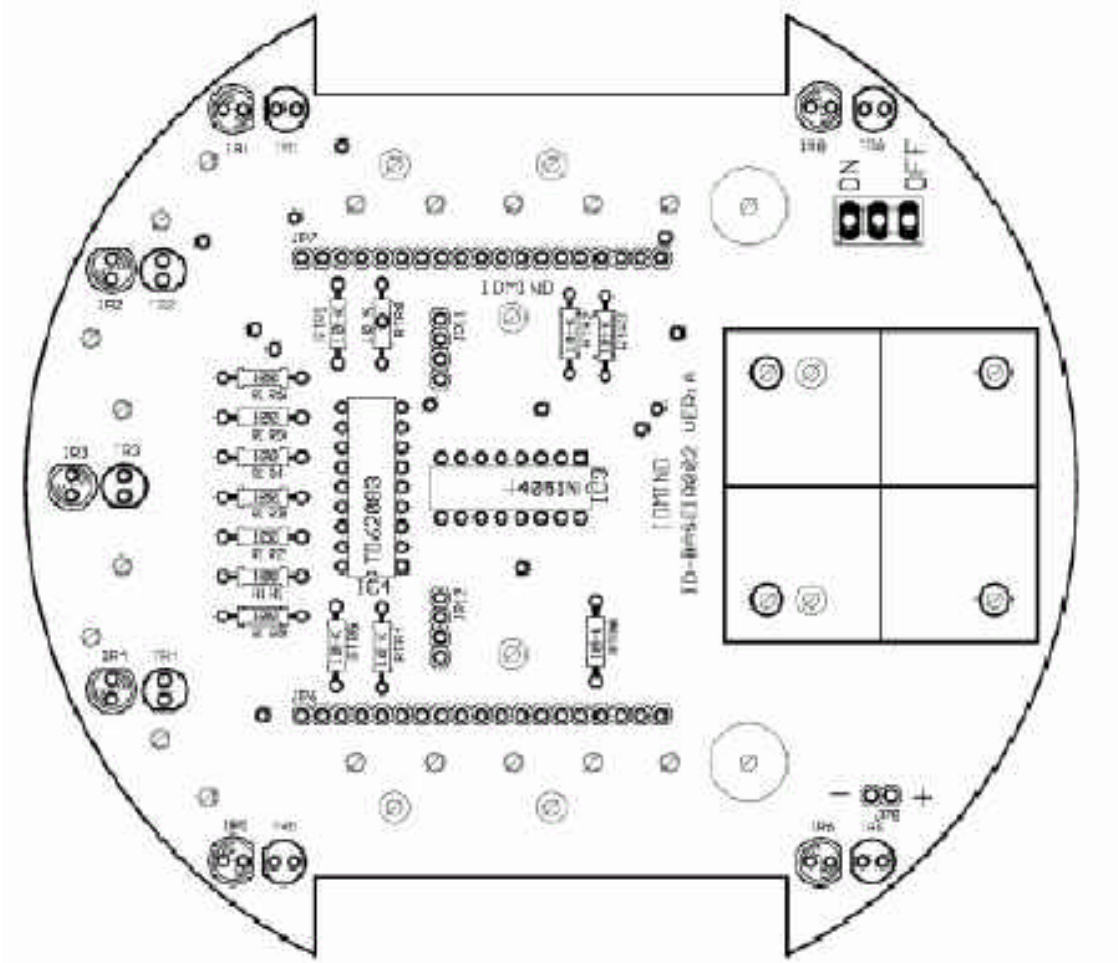

**B4) Programmierboard ID-PROG002**

| Benennung | Anzahl |
|---|---|
| 19-beinige Stiftleiste | 2 |
| 19-beinige Buchsenleiste | 2 |
| 1 k• Widerstand | 2 |
| LED | 2 |
| 100 nF Kondensator | 5 |
| Interruptor | 1 |
| RS-232 zu IDC | 1 |
| 16-beiniger Sockel | 1 |
| Max 232 | 1 |



## B5) Infrarot-Sensoren – Board ID-BASEIR002

| Benennung | Anzahl |
|---|---|
| Infrarot-Emitter | 7 |
| Infrarot-Receiver | 7 |
| 100 • Widerstand | 7 |
| 10 k• Widerstand | 7 |
| u1n2803A Entkoppler | 1 |
| 74HC4051N | 1 |
| 18-beiniger Sockel | 1 |
| 16-beiniger Sockel | 1 |
| 4-beinige Stiftleisten | 2 |
| 19-beinige Buchsenleisten | 2 |
| Verrieglung 16A mit 2 Öffnungen | 1 |
| Batterienhalter für 4*AA | 1 |
| Interruptor | 1 |
| 2-beinige Verbindungsstecker | 1 |
| Abschalter | 2 |


ON
OFF

## Anhang C – Änderungen am Roboter, um ihm das Fußballspielen zu ermöglichen

e-mail: info@idmind.pt
http://www.idmind.pt

# Soccer ROBOT

## Assembly Instructions

V 2.0

April 2003

Inhaltsverzeichnis

# 1 Zusammenbau eines Fußballroboters


## 4.3. Einleitung

## 4.4. Grundaufbaustruktur

## 4.5. Montage spezifischer Komponenten

#### 4.5.4 LDR

#### 4.5.5 Halb-Mond

#### 4.5.6 Tast-Sensoren

## 4.6. Anwendungsbeispiele

# 1 Zusammenbau eines Fußballroboters

## 4.3. Einleitung

Dieses Handbuch erläutert den Montagevorgang eines fußballspielenden Roboters, basierend auf dem *Circular Robot* von IdMind..

Unter Verwendung des *Circular - Robot* - Baukastens, entsteht die Konstruktion des Fußballroboters in zwei Phasen: a) der Grundaufbau; b) spezifische Gesichtspunkte beim Zusammenbau dieses Robotertyps.

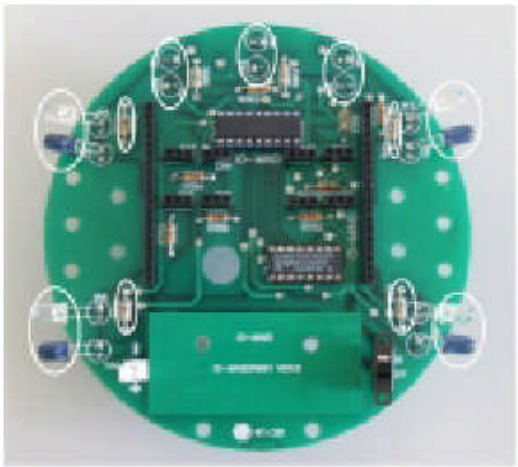
Die nächsten Kapitel zeigen den Aufbau nach diesen zwei Phasen.

Die obere Abbildung zeigt das fertige Aussehen des Fußballroboters, und einen infrarot (IR)-Ball, welcher in den Wettkämpfen genutzt wird.

## 4.4. Grundaufbaustruktur

Wenn sie die Bauanweisungen des *Circular - Robot* - Bausatzes befolgen, beachten sie bitte, dass die in der Abbildung gekennzeichneten Bauteile **nicht** eingebaut werden dürfen:

| Stelle | Bauteil |
|---|---|
| RIR 0, 1, 5, 6 | 100Ω Widerstände |
| IR 0, 1, 2, 3, 4, 5, 6 | Infrarot - Transmitter (Sender) |
| TR 0, 1, 5, 6 | Infrarot - Receiver (Empfänger) |
| | Mikroschalter für die Hindernis-Erkennung |



Die Infrarot-Empfänger **TR2**, **TR3** und **TR4** müssen mit einer Länge von **1 cm** eingebaut werden, was als ungünstig in den Hinweisen zum *Circular Robot* beschrieben steht.

Nachdem der Einbau der Infrarot-Empfänger fertig ist, müssen nun die spezifischen Elemente des Fußballroboters eingebaut werden. Diese Bauteile müssen nach den Regeln des Roboterfußballwettkampfes (*RoboCup Junior*) ausgewählt werden.

## 4.5. Montage spezifischer Komponenten

### 4.5.5 LDR

Eine Möglichkeit, den Roboter innerhalb des Fußballfeldes zu orientieren, ist es, die Markierungen auf der Feldoberfläche zu nutzen. Zu diesem Zweck müssen die LDRs eingebaut werden, welche dazu dienen, das von der Oberfläche reflektierte Licht zu messen. Dadurch ist es möglich, verschiedene Farben (oder Graustufen) zu erkennen.

Da infrarote Lichtaussendung im Wettkampf nicht erlaubt ist, wird die Oberfläche mit Licht bestrahlt, welches von einer grünen LED ausgesendet wird. Diese LED wird parallel zu der Empfänger LED eingebaut. Um eine größere Emissionskraft zu erhalten, muss der Emitterstrom erhöht werden. Dazu ist es notwendig, die Werte der Widerstände, welche in Reihe mit den Lichtemittern angeordnet sind (RIR 0, 1,5, 6), zu verringern, dass heißt von 100 Ω auf 33 Ω.

Also sind die Komponenten, welche in diesem Abschnitt eingebaut werden müssen:

| Stelle | einzubauendes Bauteil |
|---|---|
| RIR 0, 1, 5, 6 | 33 Ω Widerstände |
| IR 0, 1, 2, 5, 6 | Grüne LEDs mit hoher Intensität |
| TR 0, 1, 5, 6 | LDR |

Zu achten ist dabei auf die Polung der grünen LEDs. Die LDRs sind nicht polarisiert. Sie müssen genauso hoch eingebaut werden wie die LEDs, welche das Licht aussenden. Gezeigt wird dies auch auf dem rechten Bild (etwa 3 cm vom Board bis zur Oberseite der LEDs). Ein Hilfsmittel, z.B ein lichtundurchlässiges Rohr, muss so vor die LEDs platziert werden, dass beide Strahlen gerichteter werden, und um den Empfänger, dass er kein seitliches Licht abbekommt.

Die untere Abbildung zeigt zwei von vier Widerständen, die entfernt werden müssen. Sie befinden sich auf der Rückseite des Roboters (weiße Pfeile).

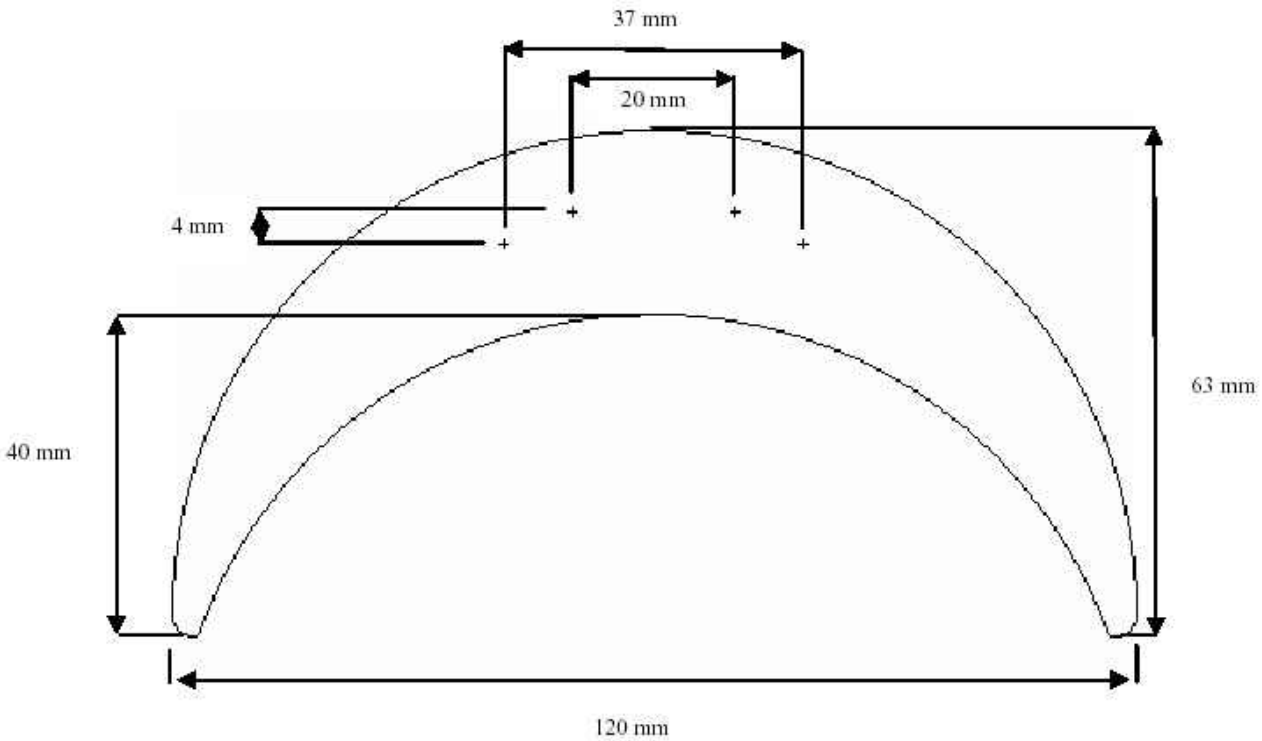
Jedes Sender/Empfänger Paar muss mit einer lichtundurchlässigen Verdunklung abgedeckt werden, so dass keine äußeren Lichtstrahlen Interferenzen verursachen können., also dass sie von äußeren Einflüssen abgeschirmt werden. Für diese Abdeckung empfehlen wir, ein schwarzes Bristolbrett zu verwenden (3,5 cm x 3,5 cm), mit einem Rohrstück und einer Abdeckung aus Isolierband. Das Rohr muss die Seiten 1 cm und 0,5 cm besitzen.

Die Verdunklung muss auf die Servomotoren geklebt werden, dazu wird hitzefester Leim benutzt, wie in der unteren Abbildung gezeigt wird (Das Bristolbrett wird auf einen der Servomotoren geleimt):

Wenn die Verdunklung und der blickdichte Zylinder montiert sind, muss der letztere mit dem Isolierband auf den TR2, TR3 und TR4 Sensoren befestigt werden. Nachdem das Band platziert wurde, müssen die Sensoren parallel zum Boden ausgerichtet werden. Sie müssen dahin zeigen, wie auf dem nebenstehenden Bild gezeigt wird:

**4.5.6 Halb-Mond**

Eine der wichtigsten Anforderungen in einem Fußballspiel ist es, den Ball unter Kontrolle zu halten. Idealerweise sollte der Roboter den Ball immer an seiner Vorderseite halten können. Wir empfehlen dazu, eine halbmondförmige Vorrichtung an der Vorderseite des Roboters anzubringen, welche parallel zum Boden verläuft. Dies wird eine sichere Ballführung erlauben, und schafft außerdem eine gute Kontrolle über seine Richtung.
Die nächste Abbildung zeigt ein Beispiel für solch eine halbmondförmige Vorrichtung. Die letztliche Form für diese Vorrichtung, kann natürlich auch ganz anders aussehen, das liegt bei ihnen.




Die Löcher welche genutzt werden, um die Vorrichtung (+) zu befestigen, müssen mit den Markierungen des Basisroboters übereinstimmen. Die Apparatur muss mit Schrauben an dem
Grundroboter befestigt werden, wie die untere Abbildung zeigt.
Durch die halbmondförmige Apparatur werden die Kontaktsensoren (wie sie beim *Circular Robot* verwendet wurden)
an einer anderen Stelle eingebaut werden.

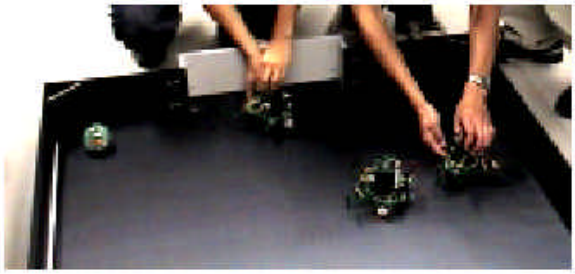
**1.3.3 Tast-Sensoren**

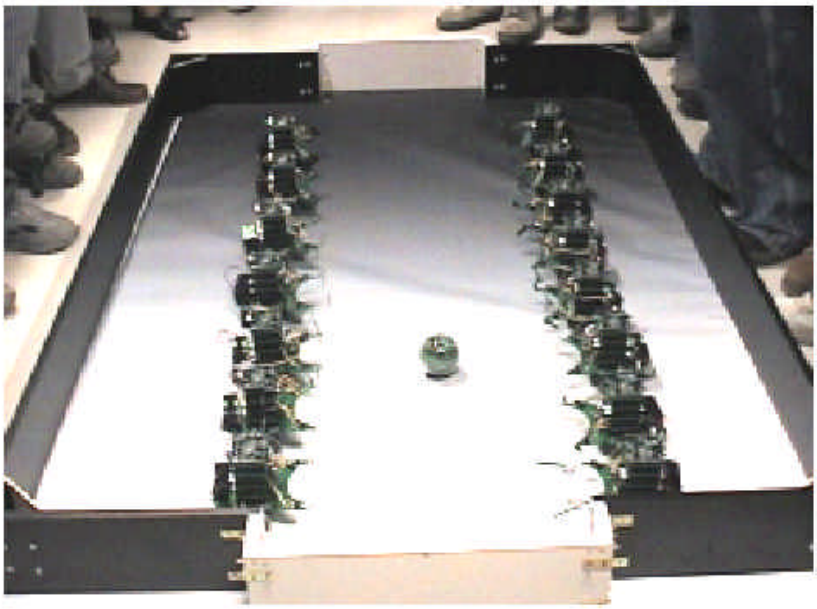
Da die Tastsensoren durch die Halbmondapparatur nicht mehr an der beim *Circular Robot* vorgesehenen Stelle eingebaut werden brauchen/können, können sie jetzt an einer anderen Stelle eingesetzt werden, wo sie zum Beispiel zur Kollisionserkennung genutzt werden können.

Wir empfehlen, diese Sensoren dort zu platzieren, wo sie unerwünschte Kollisionen erkennen können. Diese können dem Provider dann Informationen bereitstellen, um während eines Matches Konfliktsituationen während des Spiels lösen zu können. Eine gute Stelle für deren Platzierung ist die Halbmondvorrichtung selbst.

## 1.4 Anwendungsbeispiele

Die unten abgebildeten Beispiele zeigen mehrere Wettkämpfe und Events wo IdMind - Fußballroboter verwendet wurden:





Viel Spaß!!!

Kontakt zu IdMind:

**IdMind - Engenharia de Sistemas, Lda**
Centro de Incubação e Desenvolvimento - CID
Pólo Tecnológico de Lisboa, Lote1
1600-546 Lisboa
PORTUGAL

Tel: +351 21 7101100
Fax: +351 21 7101103
**http://www.idmind.pt**
**e-mail: info@idmind.pt**