Ausgabe 1.0

RC-SOCCERBOT

Übungen / Lehrerbegleitbuch

Mechanik

Elektronik

Informatik

GRAUPNER GmbH & Co. KG

Best.-Nr. R1001.110

RC-SOCCERBOT Übungen / Lehrerbegleitbuch Ausgabe 1.0
Mechanik - Elektronik - Informatik:

Vollständige Begleitunterlage um den Roboter RC-SOCCERBOT erfolgreich im Unterricht einzusetzen;
Alle Programme in der Praxis getestet.

Buch. 2007

Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden.
Herausgeber und Autor können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler ist der Herausgeber dankbar.

PN.LJ-01

Best.-Nr. R1001.110

Herausgeber: GRAUPNER GmbH & Co. KG
Autor: Hannes Runknagel
Herstellung: GRAUPNER GmbH & Co. KG
eMail: info@graupner.de
Internet: www.graupner-robotics.de
Printed in Germany

Vorwort

Dieses Buch ist die vollständige Begleitunterlage um den Roboter RC-SOCCERBOT erfolgreich im Unterricht einzusetzen und sich mit ihm vertraut zu machen. Beginnend mit der Installation der Software, einfachen Übungen zum Einstieg, wird ebenso der Aufbau des Roboters anhand von 3D-CAD Zeichnungen genau gezeigt. Den Übungseinheiten am komplett aufgebauten Roboter folgen abschliessend Kapitel mit den verschiedensten Erweiterungen. Sämtliche, Kapitel begleitende Roboterprogramme die in diesem Buch verwendet werden stehen zum kostenlosen Download unter ***www.graupner-robotics.de*** bereit. Zudem kann die komplette Montage des Roboters, unterstützt durch 3D-CAD Animationen, heruntergeladen werden.

In den Bereichen Mechanik, Elektronik und natürlich Informatik kann dieses Robotermodell zu den verschiedensten Unterrichtseinheiten eingesetzt werden. Ob mechanische Konstruktionsaufgaben durchgeführt werden sollen, das Thema Sensorik/Aktuatorik den Schülern an praktischen Beispielen näher gebracht werden soll, oder die Programmiersprache C++ auf dem Lehrplan steht, für verschiedenste Anforderungen und Aufgabenstellungen ist dieser Roboter die ideale Grundlage. Die von uns verarbeiteten Komponenten sind Industriestandard und werden vom Schüler im späteren Berufsleben in der selben Ausführung wieder gefunden.

Unterstützen Sie Ihren Unterricht mit spannenden, praktischen Übungen am Roboter.

Inhaltsverzeichnis

1. Installation der Software **6**
1.1 Voraussetzungen 6
1.2 Inhalt der CD 8
1.3 Installation der Software 10

2. Einführung in die Programmierung **14**
2.1 Inbetriebnahme des Controllerboards 14
2.2 Programme in den Editor laden 16
2.3 Programme übersetzen 18
2.4 Programm auf den Controller übertragen 20
2.5 Programme selbst erstellen 22

3. Flussdiagramme **24**
3.1 Grundelemente 24
3.2 Umsetzung in ein C++ Programm 30

4. Grundübungen **32**
4.1 Aufbau der Übungen 32
4.2 LED anschalten 34
4.3 LED Blinker 36
4.4 LED Lauflicht 38
4.5 Button abfragen 40
4.6 Motor ansteuern 42
4.7 Digital-Eingänge abfragen 44
4.8 Funktion ohne Rückgabewert 46
4.9 Funktion mit Rückgabewert 50

5. Aufbau des RC-SOCCERBOT **54**
5.1 Räder montieren 54
5.2 Motoren vorbereiten 56
5.3 Antriebseinheit montieren 58
5.4 Kicker montieren 60
5.5 Platinenhalter und Antriebe montieren 62
5.6 Kicker auf Antriebsplattform montieren 64
5.7 Controllerboard anbauen 66
5.8 Aktuatoren anschliessen 68
5.9 Kabelsatz montieren 70
5.10 Haube montieren 72

6. Übungen am Standard Roboter **76**
6.1 Motor anschalten 78
6.2 Motor an- und abschalten 80
6.3 Vorwärts fahren 82
6.4 Vorwärts mit Start 84
6.5 Startknopf mit LED als Funktion 86
6.6 Stopknopf mit LED als Funktion 90
6.7 Drehen mit Start und Stop 94
6.8 Motordrehzahl regeln 100

7. Übungen mit Erweiterungen **104**
7.1 Braitenberg Vehikel 106
7.2 Linie verfolgen 110
7.3 Bumper abfragen 114
7.4 Fahren in Richtung Lichtquelle 118
7.5 LC-Display ansteuern 122
7.6 Externes Lauflicht 126
8. Anhang A Beschaltung **130**
8.1 SoccerBoard Beschaltung 130
9. Anhang B Befehlsübersicht **132**
9.1 Globale Funktionen 132
9.2 Klasse SoccerBoard 134
9.3 Klasse LCD 136
9.4 Übersicht der Wertebereiche 138
10. Anhang C Standard-Programm **140**
10.1 Standard-Programm 140

1. Installation der Software

Vorab:
Dieses Buch beschreibt den Installationsvorgang für die Softwareversion 1.2.2, die Sie auf der Internetseite ***www.graupner-robotics.de*** unter ***Downloads --> Software*** kostenlos herunterladen können.
Bei diesem Download handelt es sich um eine CD-Image Datei, die zu einer Software CD gebrannt werden kann.
Neben der Software stehen sämtliche, Kapitel begleitende Roboterprogramme die in diesem Buch verwendet werden zum Download bereit. Diese können dann z.B. direkt auf den Roboter geladen werden.
Zudem kann die komplette Montage des Roboters, unterstützt durch 3D-CAD Animationen, heruntergeladen werden.

Um die Erstinbetriebnahme vorzunehmen ist lediglich ein passender Akku sowie ein Download Kabel/Adapter notwendig um mit dem Roboter zu kommunizieren.

Akku 2 x 6-NiMH1100, 7,2 V / 1,1 Ah	Best.-Nr. R7100
USB Download Adapter mit Kabel	Best.-Nr. R7310

Beachten Sie auch sämtliche Hinweise bezüglich Sicherheit und Handhabung aus der Bedienungsanleitung des Roboters und des USB Download Adapters!
Programme nur bei ausgeschaltetem Hauptschalter auf den Roboter downloaden, da die Motorausgänge hierbei angesteuert werden könnten.
Die Abbildungen zeigen den zusammen- und ausgebauten Roboter.

1.1 Voraussetzungen

- Die Software kann unter Windows XP oder Windows 2000 installiert werden.
- Eine freie USB Schnittstelle an Ihrem PC wird für den USB Download Adapter benötigt.

Für Ihre Notizen

1.2 Inhalt der CD

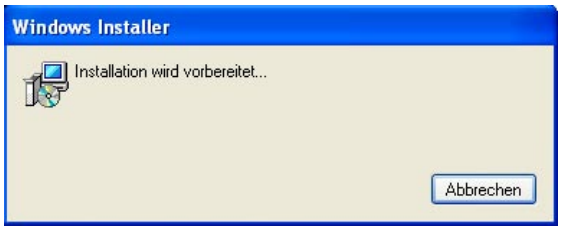
Auf der CD sind folgende Einträge vorhanden:

- Datei ***README.txt***
- Datei ***LIESMICH.txt***
- Datei ***qfixSoftware-1.2.2.msi***
- Verzeichnis ***doc***
- Verzeichnis ***examples***

Für Ihre Notizen

1.3 Installation der Software

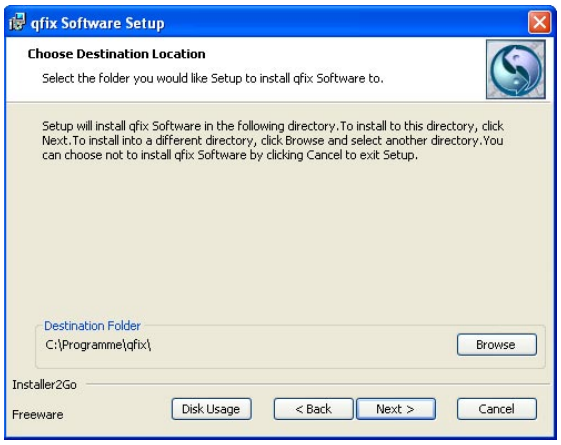
Bitte starten Sie die Installation durch doppelklicken auf ***qfixSoftware-1.2.2.msi***. Der Installationsprozess startet mit der folgenden Abbildung.



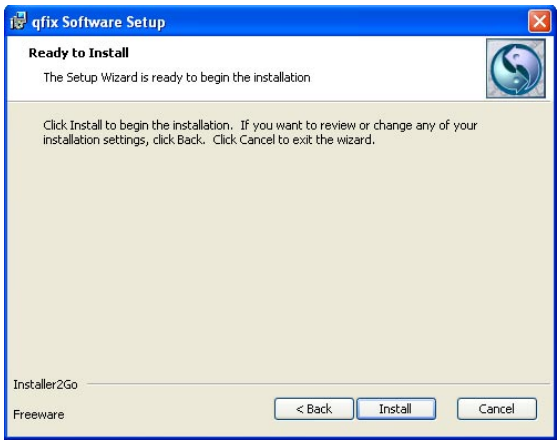
Im folgenden Willkommenfenster kann die Installation abgebrochen werden. Betätigen Sie bitte den Button „Next“ um die Installation fortzuführen.

Nun erscheint ein Fenster zur Auswahl des Zielverzeichnisses. Bitte lassen Sie den vorgeschlagenen Pfad unverändert (C:\Programme\qfix\). Betätigen Sie erneut den Button „Next“ um zum nächsten Fenster zu gelangen.

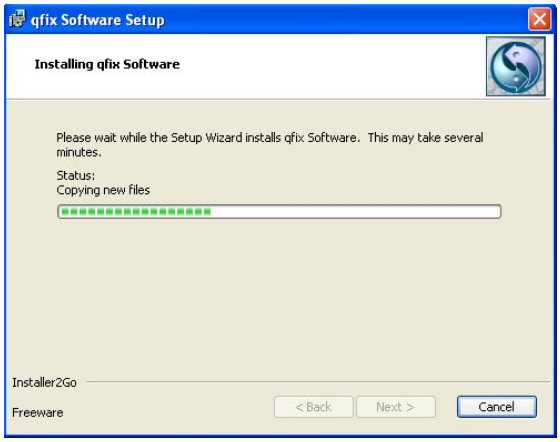
Achtung: Bei falsch gewähltem Pfad kompilieren die mitgelieferten Programme nicht richtig!
(Die Software kann allerdings jederzeit nochmals de-installiert und wieder korrekt installiert werden.)



Die folgende Abbildung zeigt das letzte Fenster, ehe die eigentliche Installation beginnt. An dieser Stelle können Sie den Installationsvorgang nochmals abbrechen. Durch Betätigen des Button „Install“ starten Sie bitte den Installationsvorgang.



Nun öffnet sich ein Fenster, das durch einen Fortschrittsbalken den Status der Installation anzeigt. Dieser Vorgang kann je nach Rechner ein paar Minuten dauern.



Das folgende Fenster zeigt den erfolgreichen Abschluss der Installation an. Durch Drücken des Button „Next“ wird dieses Fenster quittiert.

Das letzte Fenster zeigt noch Informationen über das verwendete Tool zur Installation an. Dieses Fenster kann mit dem Button „Finish“ bestätigt werden.

Damit ist die benötigte Software installiert!

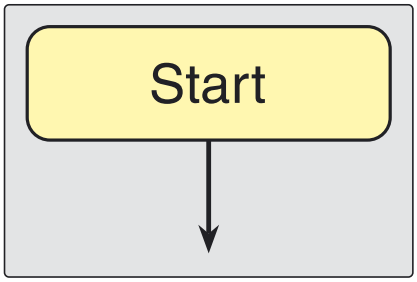
Über ***Start --> Programme --> qfix --> Programmers Notepad*** kann der Editor zum Erstellen und Ändern von Roboter-Programmen verwendet werden.

3.1 Grundelemente

In den Übungsaufgaben der folgenden Kapitel liegt der Fokus auf der Erstellung von Algorithmen und der Umsetzung dieser Algorithmen in Programmcode. Aus diesem Grund werden in diesem Kapitel Flussdiagramme eingeführt und dann benutzt, um Programmabläufe zu beschreiben.

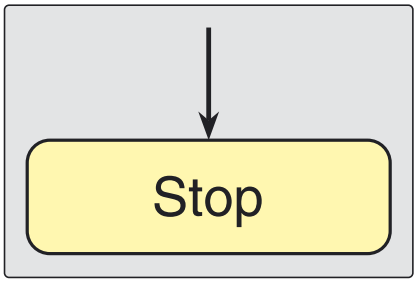
Flussdiagramme sind eine normierte Methode zur grafischen Darstellung von Programmen. Ein komplettes Flussdiagramm ist immer aus einer Anzahl von Grundelementen zusammengesetzt. Die wichtigsten Grundelelemente werden im Folgenden gezeigt.

Programm-Start:



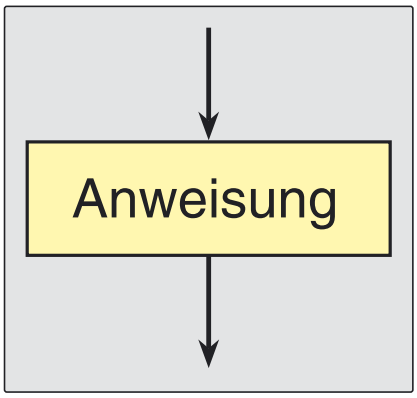
Das Element Start ist am Anfang von jedem Flussdiagramm zwingend notwendig.

Programm-Ende:

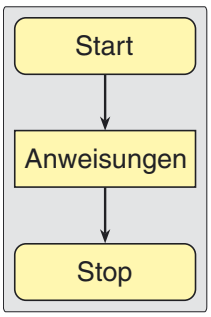
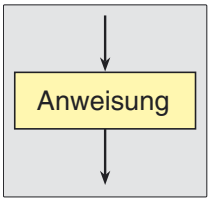
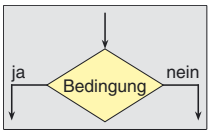
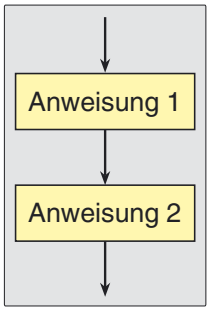
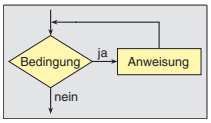


Mit Stop kann ein Flussdiagramm enden. Dieses Elemet ist jedoch nicht zwingend erforderlich.

Anweisung:



Hinter dem Element Anweisung kann sich eine Zuweisung oder z.B. ein Methodenaufruf verbergen.

Name	Symbol	C++ Syntax	Beispiel
Standard-Programm		***<Includes>*** ***<Instanzen>;*** ***int main()*** ***{*** ***<Anweisungen>;*** ***}***	***#include "qfixSoccerBoard.h"*** ***SoccerBoard robot;*** ***int main()*** ***{*** ***robot.motor(1, 200);*** ***}***
Anweisung		Zuweisung: ***<Variable> = <Wert>;*** Methodenaufruf: ***<Objekt>.<Methode>(<Parameter>);***	 ***x = 100;*** ***robot.motor(0, 223);***
Verzweigung		 ***if (<Bedingung>) {*** ***<Anweisungen>;*** ***}*** ***else {*** ***<Anweisungen>;*** ***}***	***x = robot.analog(0);*** ***if (x > 100) {*** ***robot.motor(0, 35);*** ***}*** ***else {*** ***robot.motor(0, 85);*** ***}***
Hinterein-ander Ausführung		***<Anweisung>;*** ***<Anweisung>;*** ***<Anweisung>;*** ***<Anweisung>;***	***x = 222;*** ***robot.motor(0, x);*** ***y = -165;*** ***robot.motor(1, y);***
While Schleife		 ***while (<Bedingung>) {*** ***<Anweisungen>;*** ***}***	***x = robot.analog(0);*** ***while (x > 100) {*** ***robot.motor(0, 35);*** ***}***

Die Aufgaben in diesem Kapitel dienen dazu, den Controller und seine Programmierung kennenzulernen. Hierfür ist es nicht unbedingt nötig, den kompletten RC-SOCCERBOT Roboter aufzubauen. Es genügt vielmehr, lediglich das Controllerboard wie im Kapitel 2 beschrieben, in Betrieb zu nehmen. Natürlich kann auch der komplette Roboter wie in Kapitel 5 beschrieben montiert werden. Programme nur bei ausgeschaltetem Hauptschalter downloaden, da die Motorausgänge hierbei angesteuert werden könnten.

4.1 Aufbau der Übungen

Voraussetzung für die folgenden Übungsaufgaben ist, dass die Software installiert und dem Übenden bekannt ist, wie man eigene Programme auf das Controllerboard lädt. Weiterhin sollten Grundkenntnisse in C/C++ vorhanden sein.

Die Übungen selbst entsprechen jeweils folgender Struktur:

- Aufgabe
- Lösungsansatz
- Flussdiagramm
- Programm
- Erweiterungen

Die „Aufgabe“ beinhaltet die Übungsaufgabe an sich. Hier werden oft Abkürzungen der Art „LED 0“, anstatt „LED“ mit dem „Index 0“, verwendet.

Der „Lösungsansatz“ gibt erste Hinweise zur Lösung, beispielsweise welche Befehle verwendet werden können oder wie der logische Ablauf aussehen muss.

Unter „Programm“ wird das tatsächliche Lösungsprogramm in C/C++ aufgelistet. Hierfür wird immer von folgendem Standard-Programm[1] ausgegangen.

```
#include “qfixSoccerBoard.h“

SoccerBoard robot;

int main()
{
  /* Hier stehen die Anweisungen */
}
```

Die Befehle, die zur Steuerung des Contollerboard bzw. des Roboters zur Verfügung stehen, sind im „Anhang B“ aufgelistet.

Im Abschnitt „Erweiterungen“ werden Vorschläge zur Abänderung der Übungsaufgabe oder zu weitergehenden Übungen gegeben.

[1] Siehe auch „Anhang C“

4.4 LED Lauflicht

Aufgabe

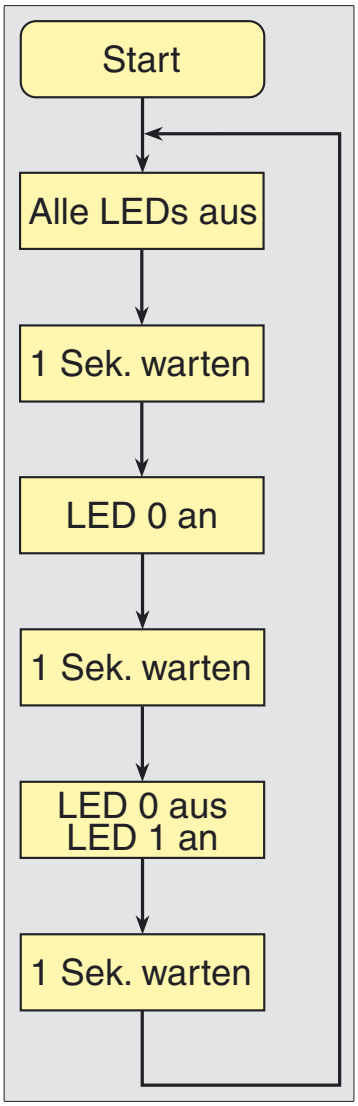
Die LED 0-1 soll jeweils ca. eine Sekunde lang als Lauflicht blinken. Dabei soll der Ablauf bei LED 0 beginnen und bei LED 1 enden. Erneut soll eine Endlosschleife zum Einsatz kommen.

Lösungsansatz

Da das Lauflicht erneut unendlich lange wiederholt werden soll, muss eine Endlosschleife programmiert werden. Für das Warten kann entweder die bekannte Funktionen „sleep“ oder aber „msleep“ verwendet werden.

Beim Ablauf muss man sich klar werden, dass die LEDs eingeschalten und abgeschalten werden müssen.

Flussdiagramm



Programm

```
#include "qfixSoccerBoard.h"

SoccerBoard robot;

int main()
{
  while (true) {
    robot.ledOff(0);
    robot.ledOff(1);
    sleep(1);
    robot.ledOn(0);
    sleep(1);
    robot.ledOff(0);
    robot.ledOn(1);
    sleep(1);
  }
}
```

Erweiterungen

Anstatt einer Sekunde können andere Wartezeiten vorgegeben werden.

Nachdem die LED 1 eingeschalten ist, soll der Ablauf von oben nach unten erfolgen, was soviel heisst wie nochmals LED 1 einschalten, danach LED 1 ausschalten und LED 0 einschalten.

Der Ablauf soll dahingehend geändert werden, dass die LEDs zur Anzeige einer Zählung im Dualsystem verwendet werden sollen.

Zeit 00: LED 0 aus - LED 1 aus
Zeit 01: LED 0 ein - LED 1 aus
Zeit 02: LED 0 aus - LED 1 ein
Zeit 03: LED 0 ein - LED 1 ein
Zeit 00: LED 0 aus - LED 1 aus

Vom Dezimalwert 3 soll nun „heruntergezählt“ werden.

Zeit 00: LED 0 ein - LED 1 ein
Zeit 01: LED 0 aus - LED 1 ein
Zeit 02: LED 0 ein - LED 1 aus
Zeit 03: LED 0 aus - LED 1 aus
Zeit 00: LED 0 ein - LED 1 ein

4.8 Funktion ohne Rückgabewert

Aufgabe

Es soll eine Funktion „showButton“ geschrieben werden, die als Parameter einen ganzzahligen Wert „index“ entgegennimmt, selbst jedoch keinen Rückgabewert liefert. Die Funktion soll den Taster mit dem Index „index“ abfragen und je nach aktuellem Zustand die LED mit dem Index „index“ an- oder ausschalten.

Die Funktion „showButton“ soll dann im Hauptprogramm benutzt werden, um andauernd den Zustand aller zwei Taster über die LEDs auszugeben.

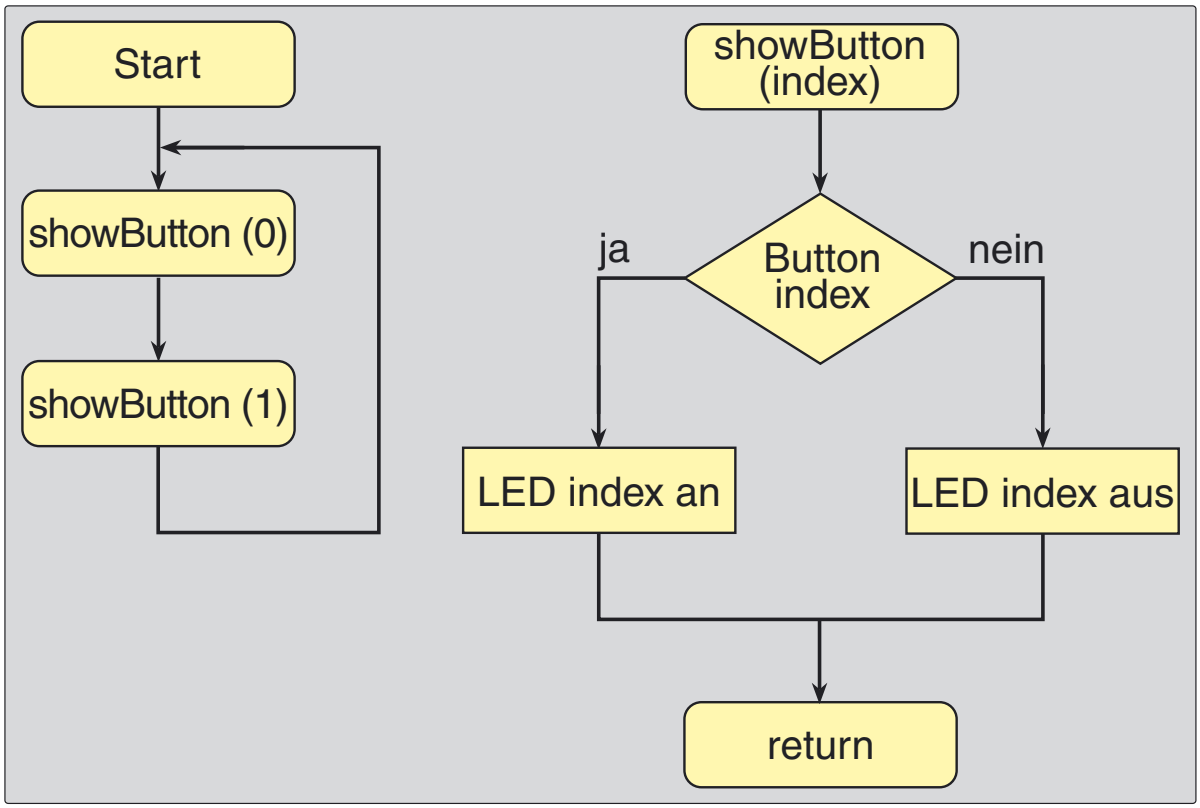
Lösungsansatz

Die Signatur der gewünschten Funktion lautet wie folgt: ***void showButton(int index)***. Da die Funktion auf das Objekt „robot“ zugreifen muss, muss sie nach der Deklaration ***SoccerBoard robot;***, aber vor dem Hauptprogramm definiert werden.

Als Funktionsrumpf enthält die Funktion genau die Fallunterscheidung aus Kapitel 4.5 Button abfragen, außer dass der konstante Wert 0 durch den Parameter „index“ ersetzt wird.

Im Hauptprogramm wird dann eine Endlosschleife programmiert, die lediglich die Funktionsaufrufe ***showButton(0);*** bis ***showButton(1);*** enthält.

Flussdiagramm



Programm

```
#include "qfixSoccerBoard.h"

SoccerBoard robot;


void showButton(int index)
{
  if (robot.button(index)) robot.ledOn(index);
  else                     robot.ledOff(index);
}


int main()
{
  while (true) {
    showButton(0);
    showButton(1);
  }
}
```

Fortgeschrittene benutzen evtl. eine „for-Schleife", um die zwei Aufrufe der Funktion zu umgehen. Dies führt zu folgendem Programm.

```
#include "qfixSoccerBoard.h"

SoccerBoard robot;


void showButton(int index)
{
  if (robot.button(index)) robot.ledOn(index);
  else                     robot.ledOff(index);
}


int main()
{
  while (true) {
    for (int i=0; i<=1; i++) {
      showButton(i);
    }
  }
}
```

Erweiterungen

Um sichere Programme zu schreiben, sollte eine Funktion stets die an sie übergebenen Parameter prüfen. So kann die Funktion „showButton“ um eine Sicherheitsabfrage erweitert werden, die den Parameter „index“ prüft und alle Werte außerhalb des erlaubten Bereichs [0-1] ignoriert.

Dies kann folgendermaßen implementiert werden.

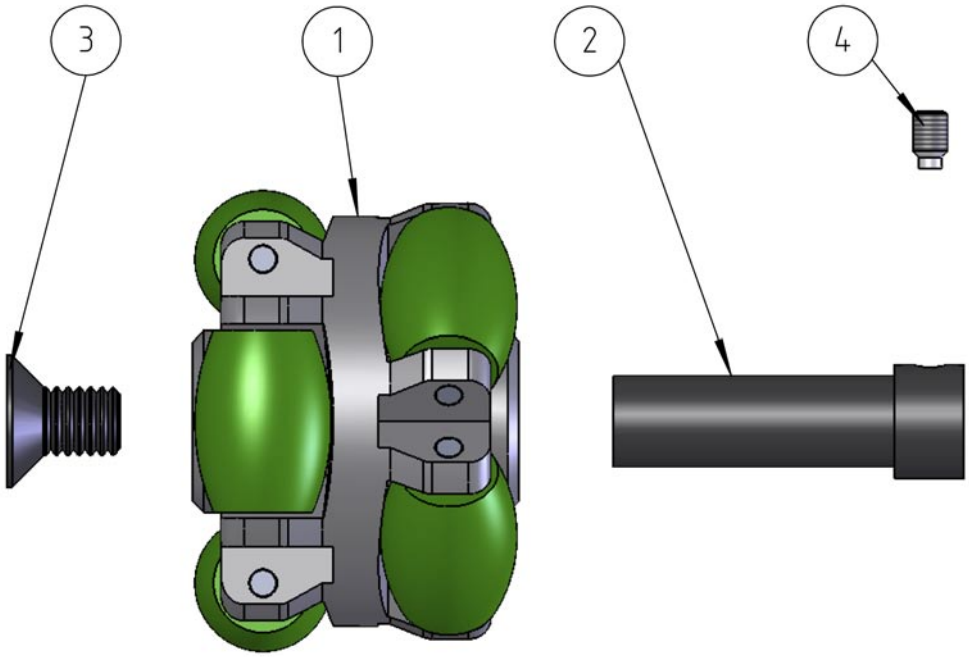
```
#include “qfixSoccerBoard.h“

SoccerBoard robot;


void showButton(int index)
{
  if (index<0) return;
  if (index>1) return;

  if (robot.button(index)) robot.ledOn(index);
  else                     robot.ledOff(index);
}
```

5.1 Räder montieren

Benötigte Komponenten

POS-NR.	BENENNUNG	MENGE
1	Omnirad	1
2	Radbuchse	1
3	Senkkopf-Schraube M6 x 10	1
4	Gewindestift	1

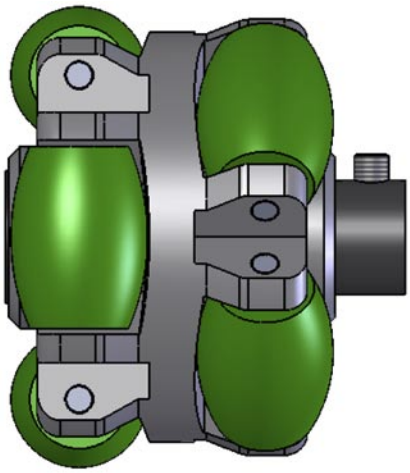
Vorgehensweise

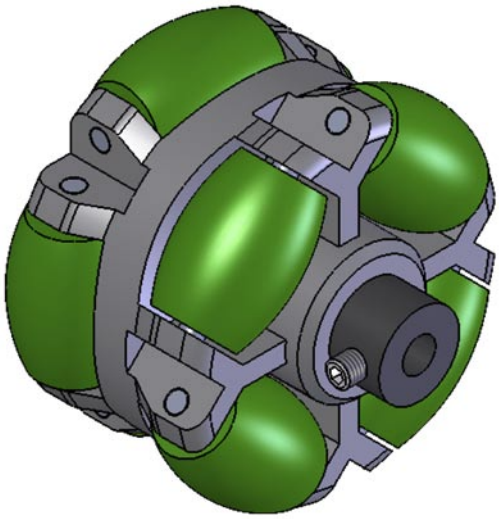


Ablauf

- Gewindestift (4) in die Radbuchse (2) einschrauben.
- Radbuchse (2) so weit wie möglich in das Omnirad (1) einschieben.
- Eingeschobene Radbuchse (2) mit Senkkopf-Schraube (3) von vorne her fest verschrauben.
- Da insgesamt drei dieser Räder benötigt werden, muss dieser Montageschritt noch zweimal wiederholt werden.

Fertig montiert



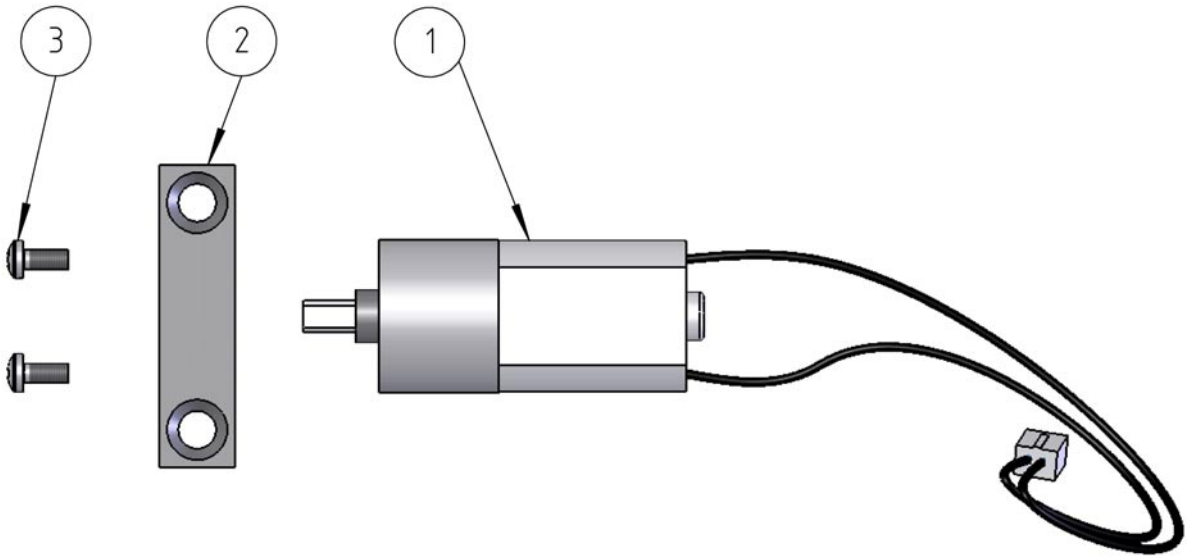


5.2 Motoren vorbereiten

Benötigte Komponenten

POS-NR.	BENENNUNG	MENGE
1	Motor	1
2	Motorblock	1
3	Schraube M2,5x6	2

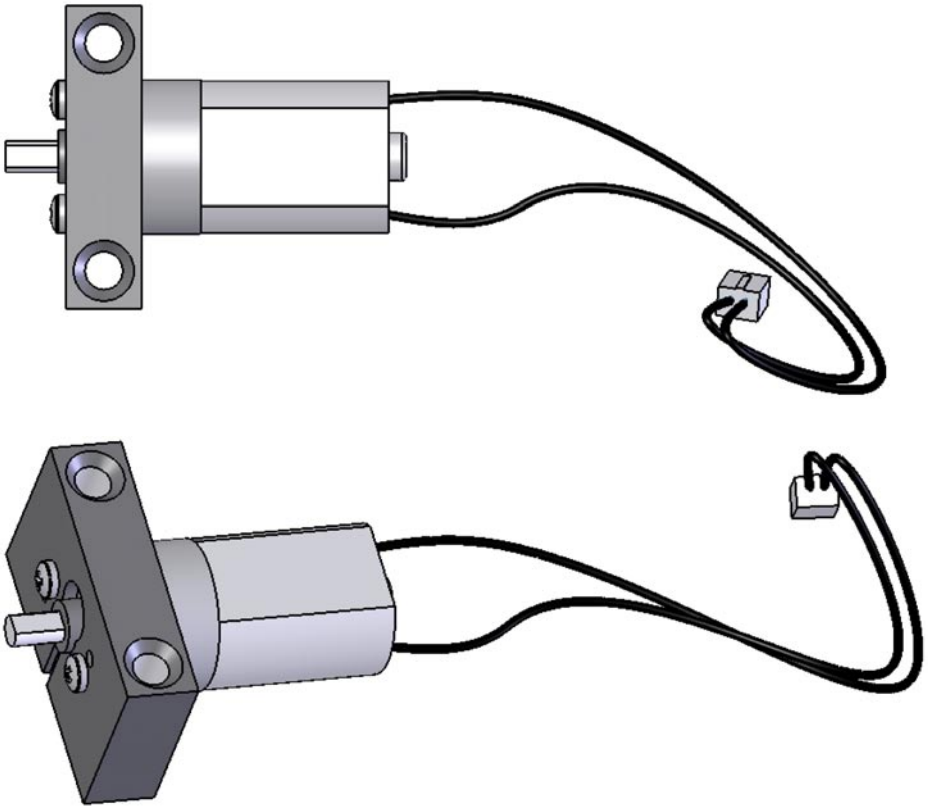
Vorgehensweise



Ablauf

- Motor (1) in die passende, runde Öffnung des Motorblocks (2) einschieben.
- Eingeschobener Motor (1) mit dem Motorblock (2) durch 2 x Schraube (3) von vorne her verschrauben.
- Da insgesamt drei dieser Motoren benötigt werden, muss dieser Montageschritt noch zweimal wiederholt werden.

Fertig montiert

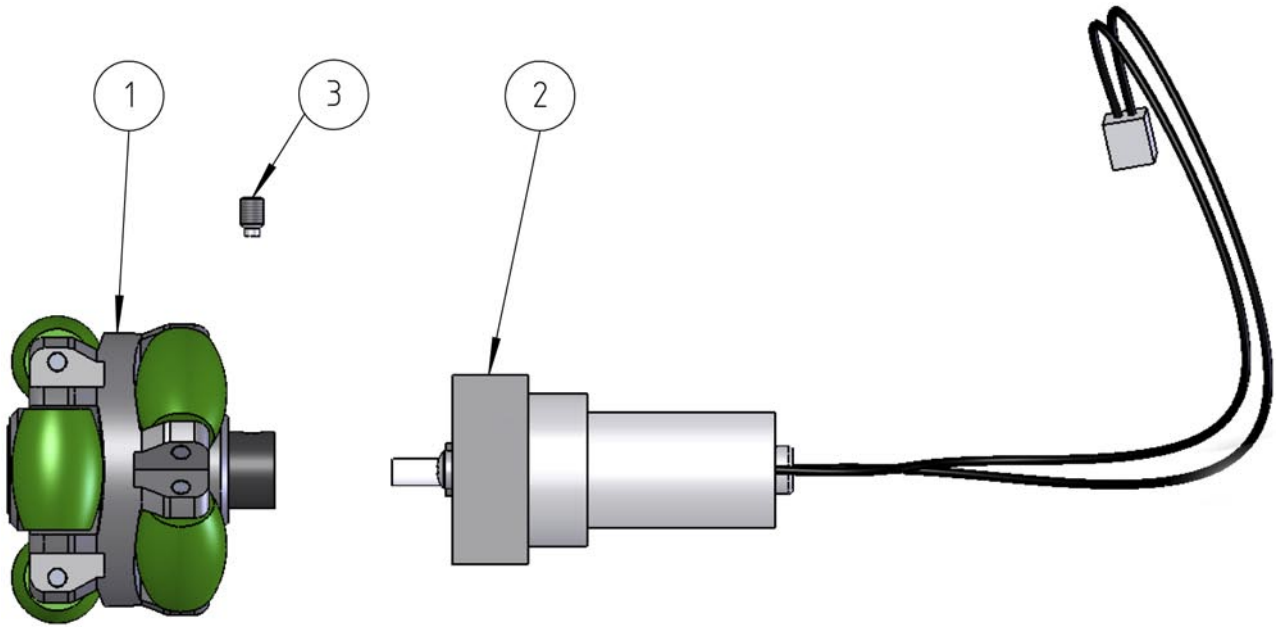


5.3 Antriebseinheit montieren

Benötigte Komponenten

POS-NR.	BENENNUNG	MENGE
1	Omni-Rad montiert	1
2	Motorblock montiert	1
3	Gewindestift	1

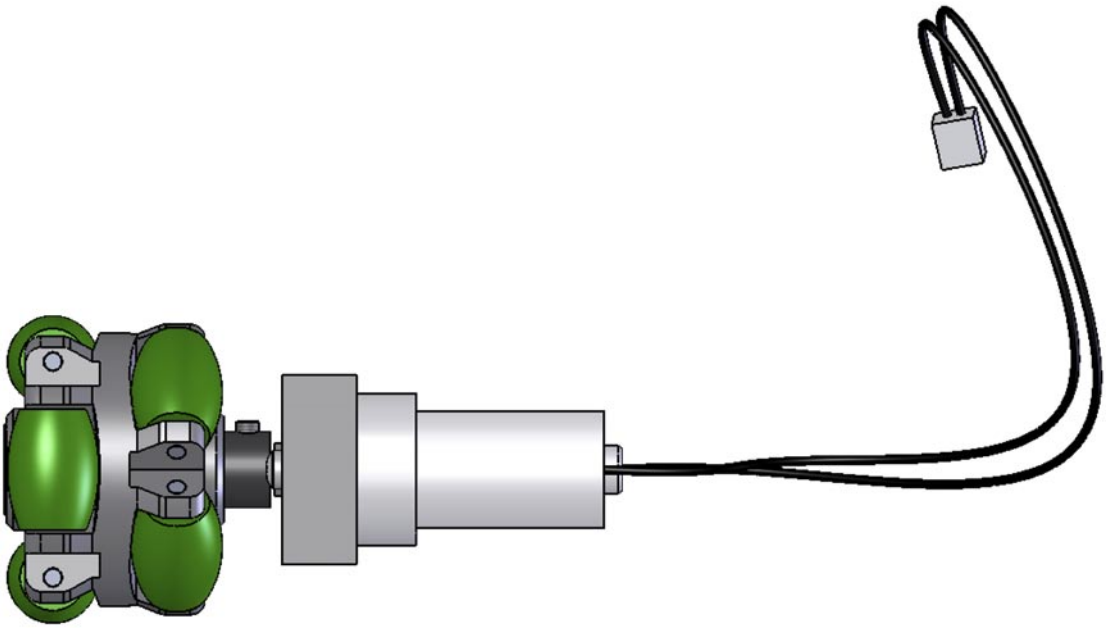
Vorgehensweise

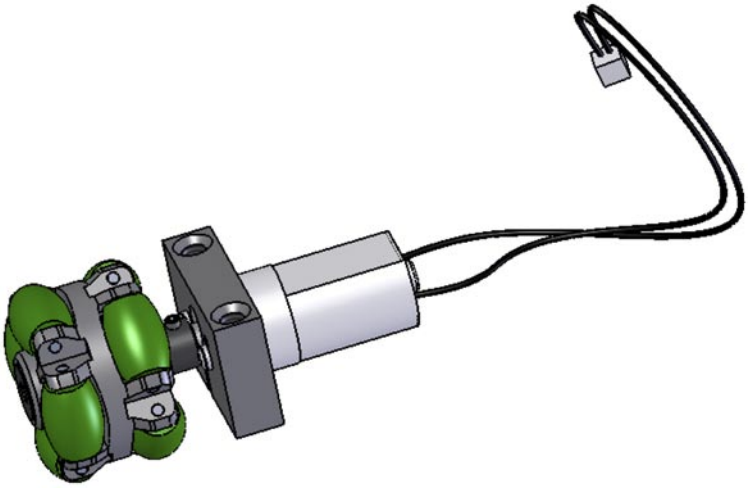


Ablauf

- Montierten Gewindestift (3) aus Radbuchse des Omi-Rad (1) entfernen.
- Omni-Rad (1) auf Motorwelle des Motorblocks (2) aufschieben und so ausrichten, dass der Gewindestift (3) auf die abgeflachte Seite der Motorwelle zeigt.
- Gewindestift wieder in die Radbuchse des Omni-Rad (1) eindrehen und damit die Motorwelle fixieren.
- Da insgesamt drei dieser Räder benötigt werden, muss dieser Montageschritt noch zweimal wiederholt werden.

Fertig montiert



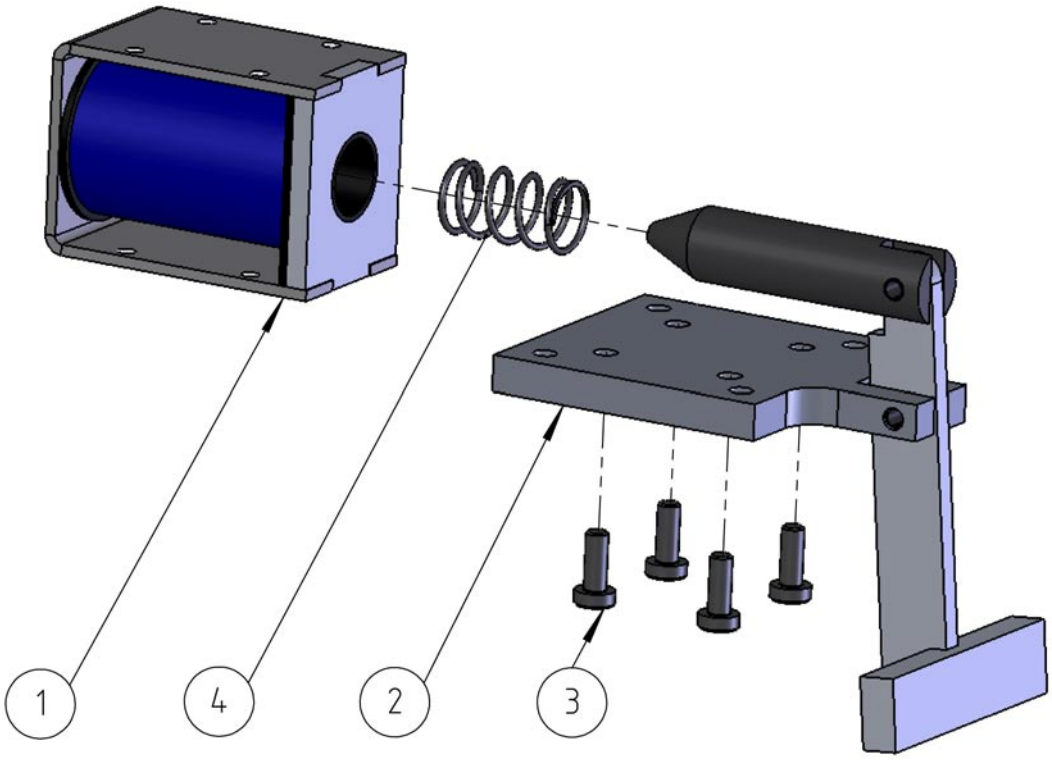


5.4 Kicker montieren

Benötigte Komponenten

POS-NR.	BENENNUNG	MENGE
1	Magnet	1
2	Schussmechanik	1
3	Schraube M3 x 8	4
4	Feder	1

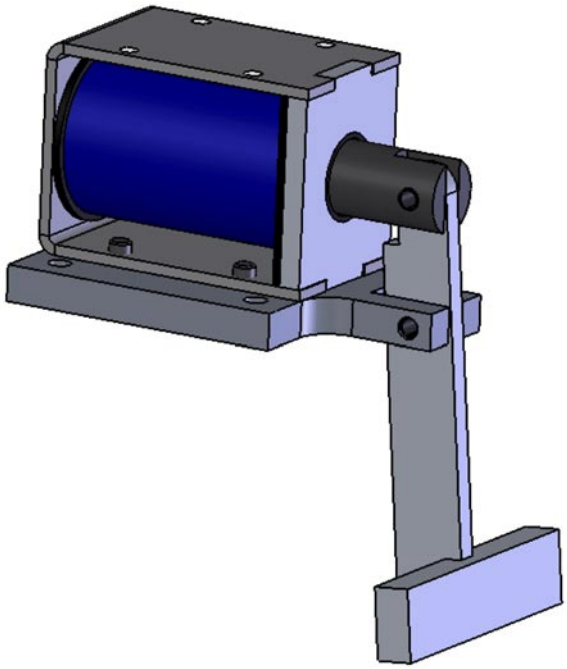
Vorgehensweise



Ablauf

- Feder (4) auf den runden beweglichen Teil der Schussmechanik (2) aufsetzen.
- Den runden beweglichen Teil der Schussmechanik (2) in den Magnet (1) einführen.
- Mittels der 4 x Schrauben (3) den Magnet (1) mit der Schussmechanik (2) von unten her verschrauben.

Fertig montiert

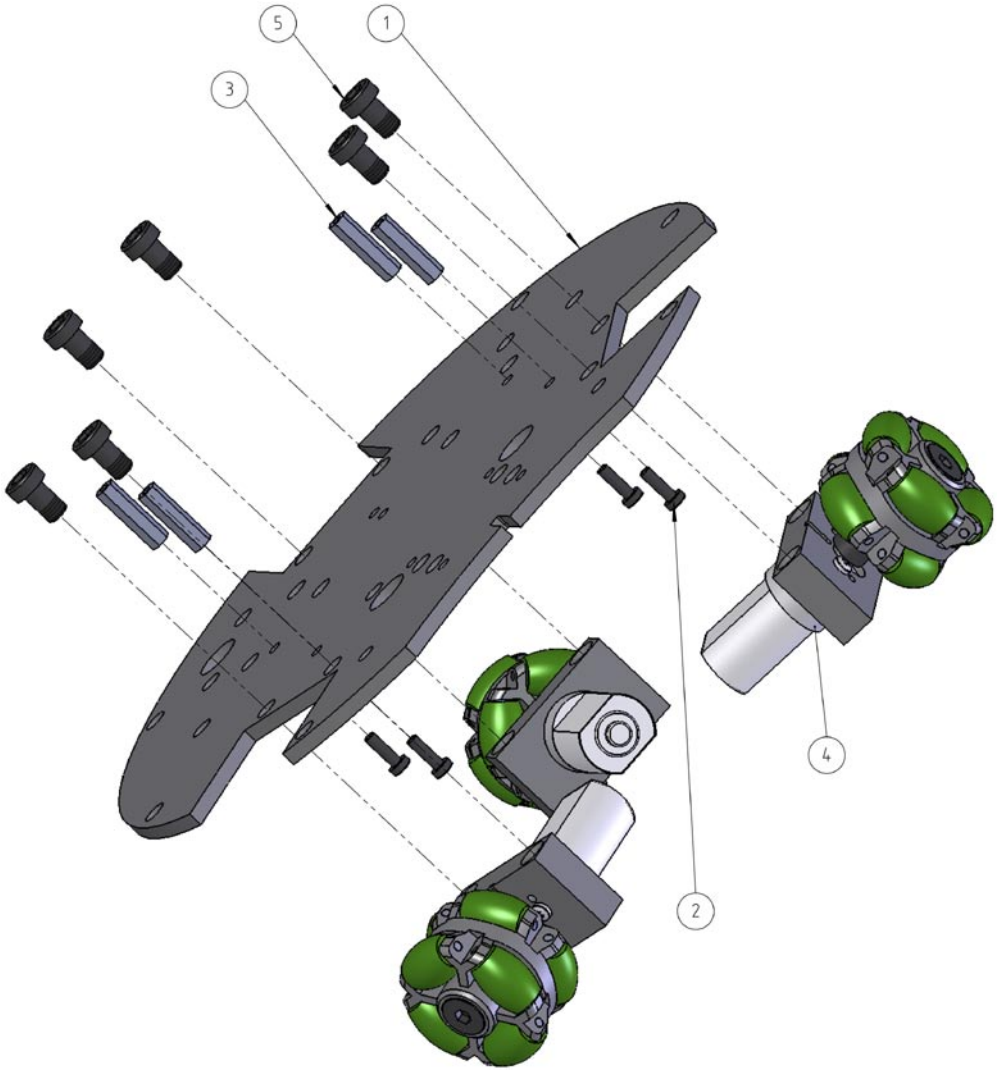


5.5 Platinenhalter und Antriebe montieren

Benötigte Komponenten

POS-NR.	BENENNUNG	MENGE
1	Grundplatte	1
2	Schraube M3 x 10	4
3	Abstandshalter M3x20	4
4	Antriebseinheit	3
5	Schraube M6 x 10	6

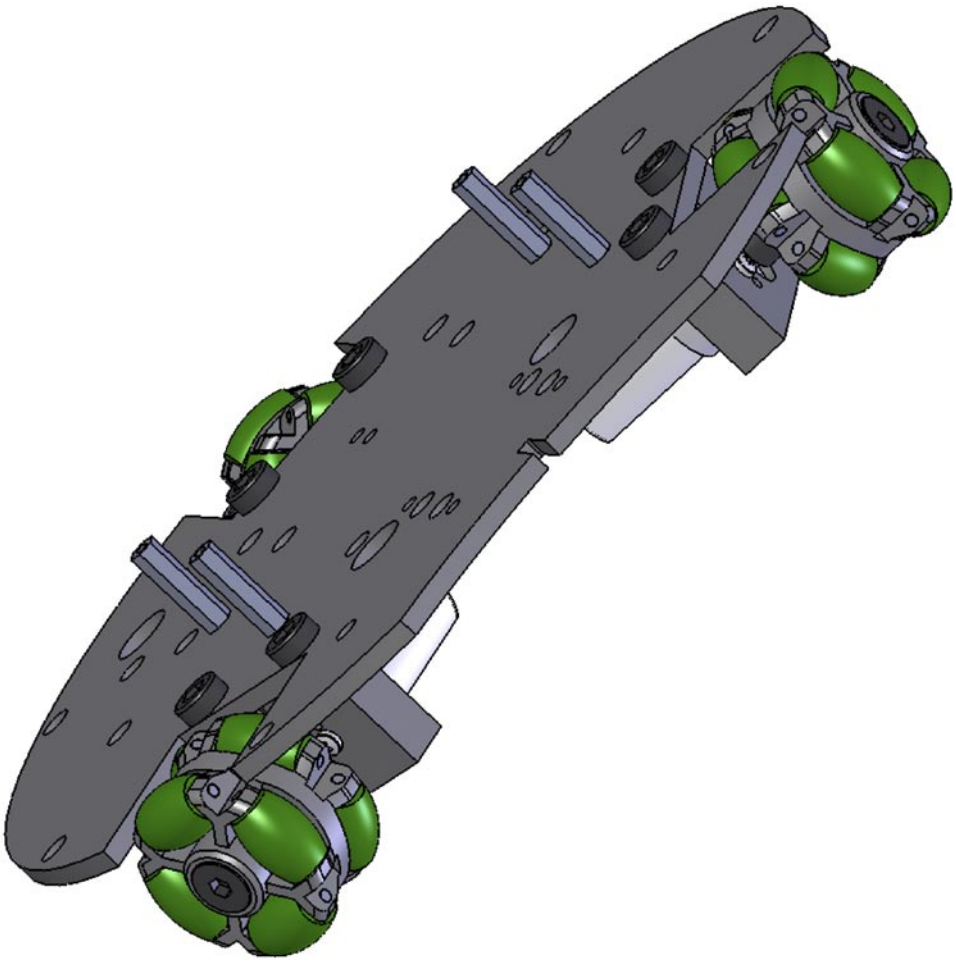
Vorgehensweise



Ablauf

- 4 x Abstandshalter (3) auf der Grundplatte (1) mittels der 4 x Schrauben (2) von unten her fixieren.
- 3 x Antriebseinheit (4) mittels der 6 x Schrauben (5) von oben her mit der Grundplatte verschrauben.

Fertig montiert

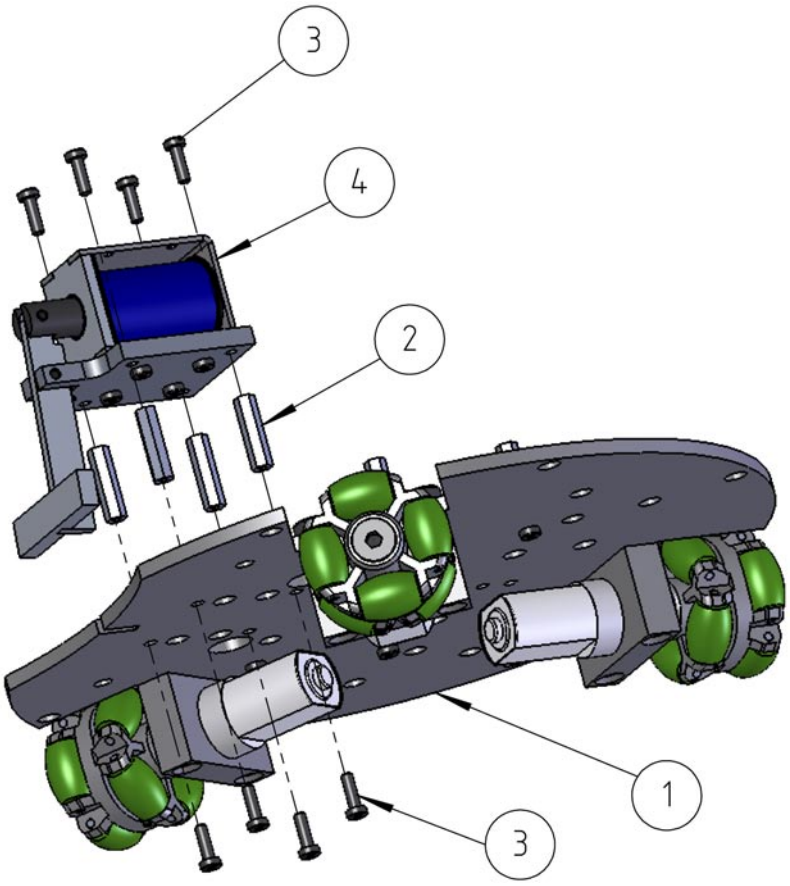


5.6 Kicker auf Antriebsplattform montieren

Benötigte Komponenten

POS-NR.	BENENNUNG	MENGE
1	Antriebsplattform	1
2	Abstandshalter M3x20	4
3	Schraube M3 x 10	8
4	Kicker	1

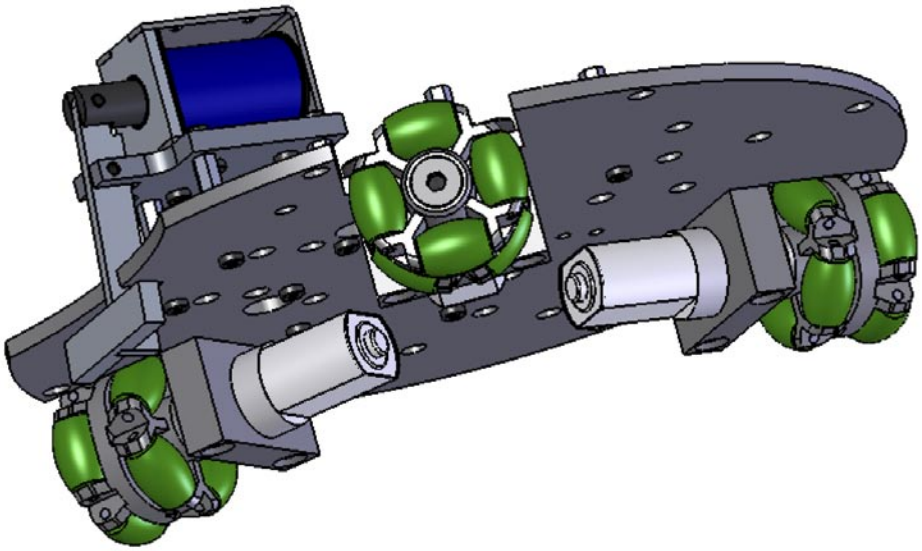
Vorgehensweise

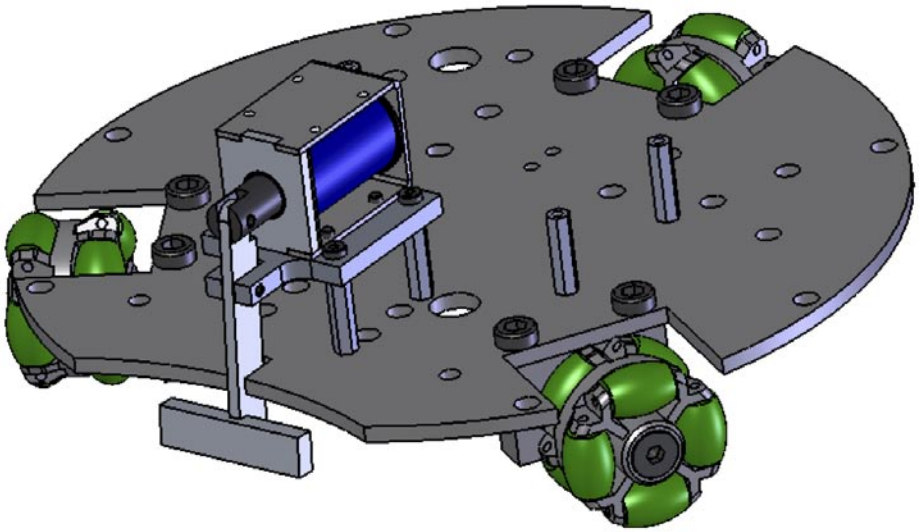


Ablauf

- 4 x Abstandshalter (2) auf der Antriebsplattform (1) mittels der 4 x Schrauben (3) von unten her fixieren.
- Den Kicker (4) mittels der 4 x Schrauben (3) von oben her mit den 4 x Abstandshaltern (2) verschrauben.

Fertig montiert



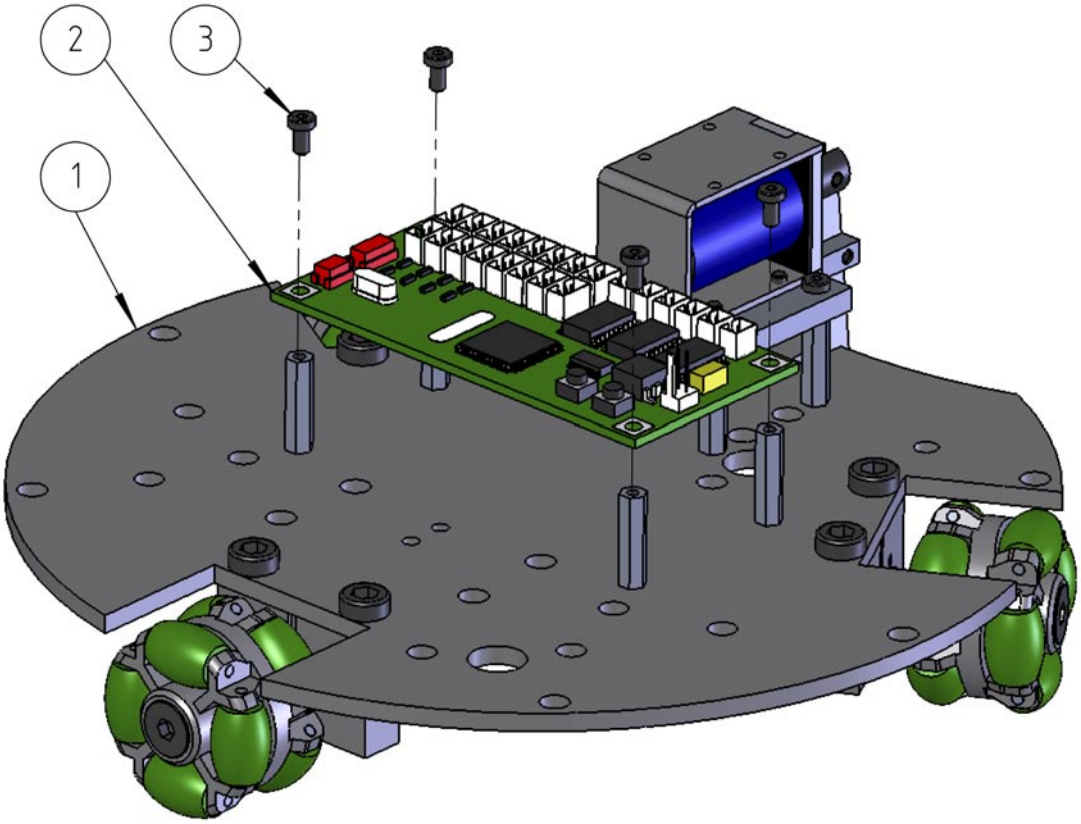


5.7 Controllerboard anbauen

Benötigte Komponenten

POS-NR.	BENENNUNG	MENGE
1	Antriebsplattform	1
2	Controllerboard	1
3	Schraube M3 x 10	4

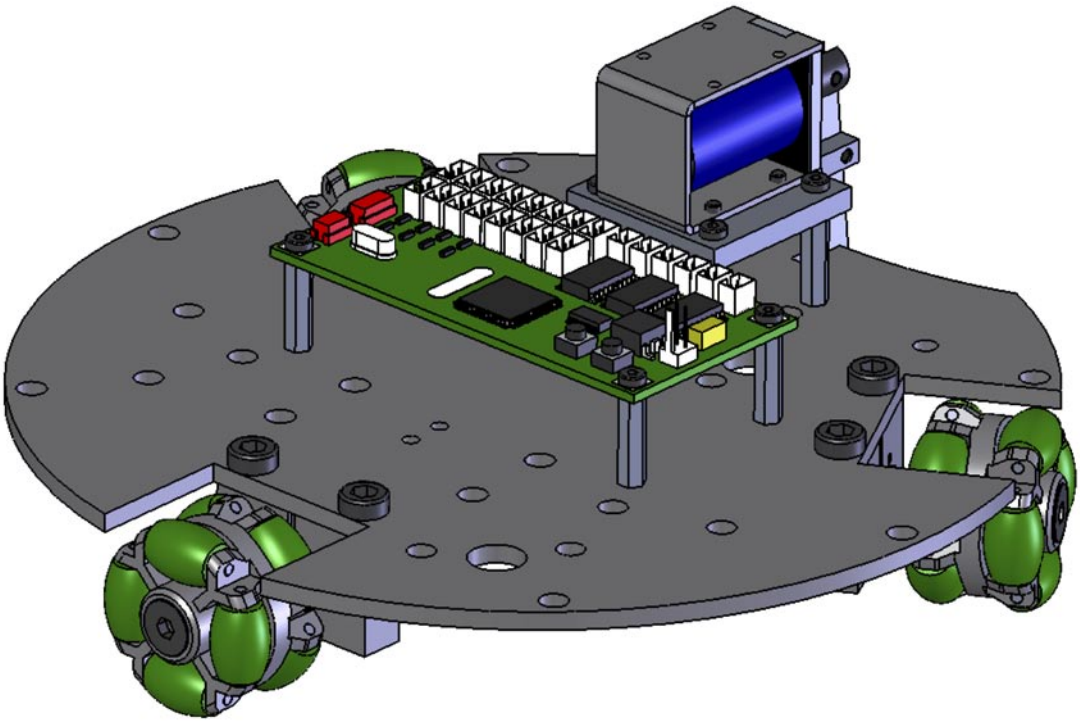
Vorgehensweise



Ablauf

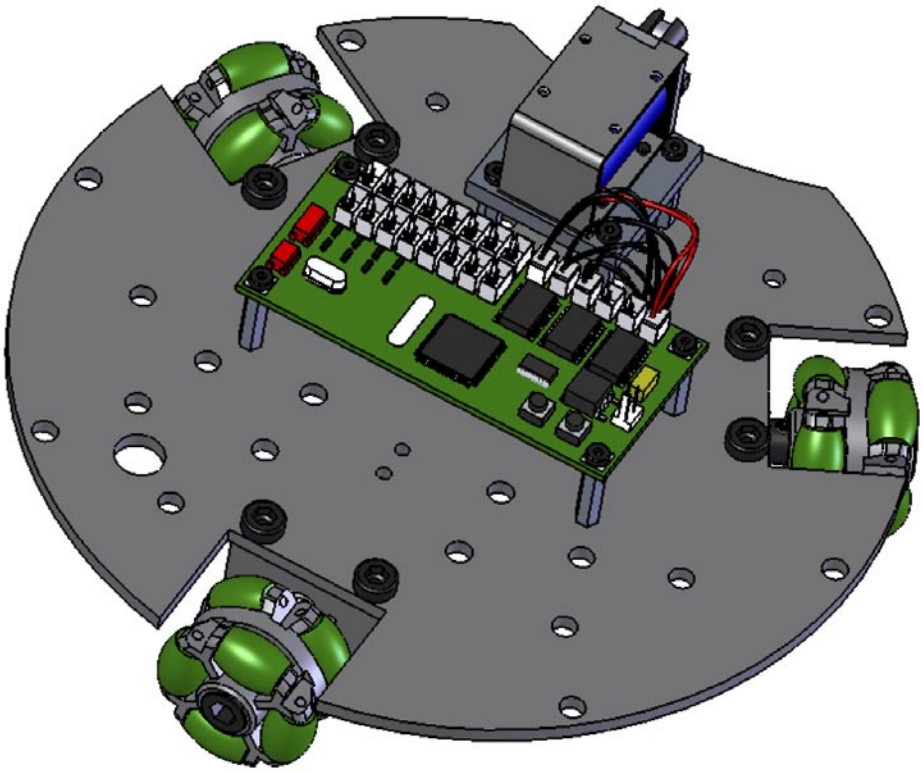
- Das Controllerboard (2) mittels der 4 x Schrauben (3) auf die Abstandshalter der Antriebsplattform (1) anbringen.

Fertig montiert



5.8 Aktuatoren anschliessen

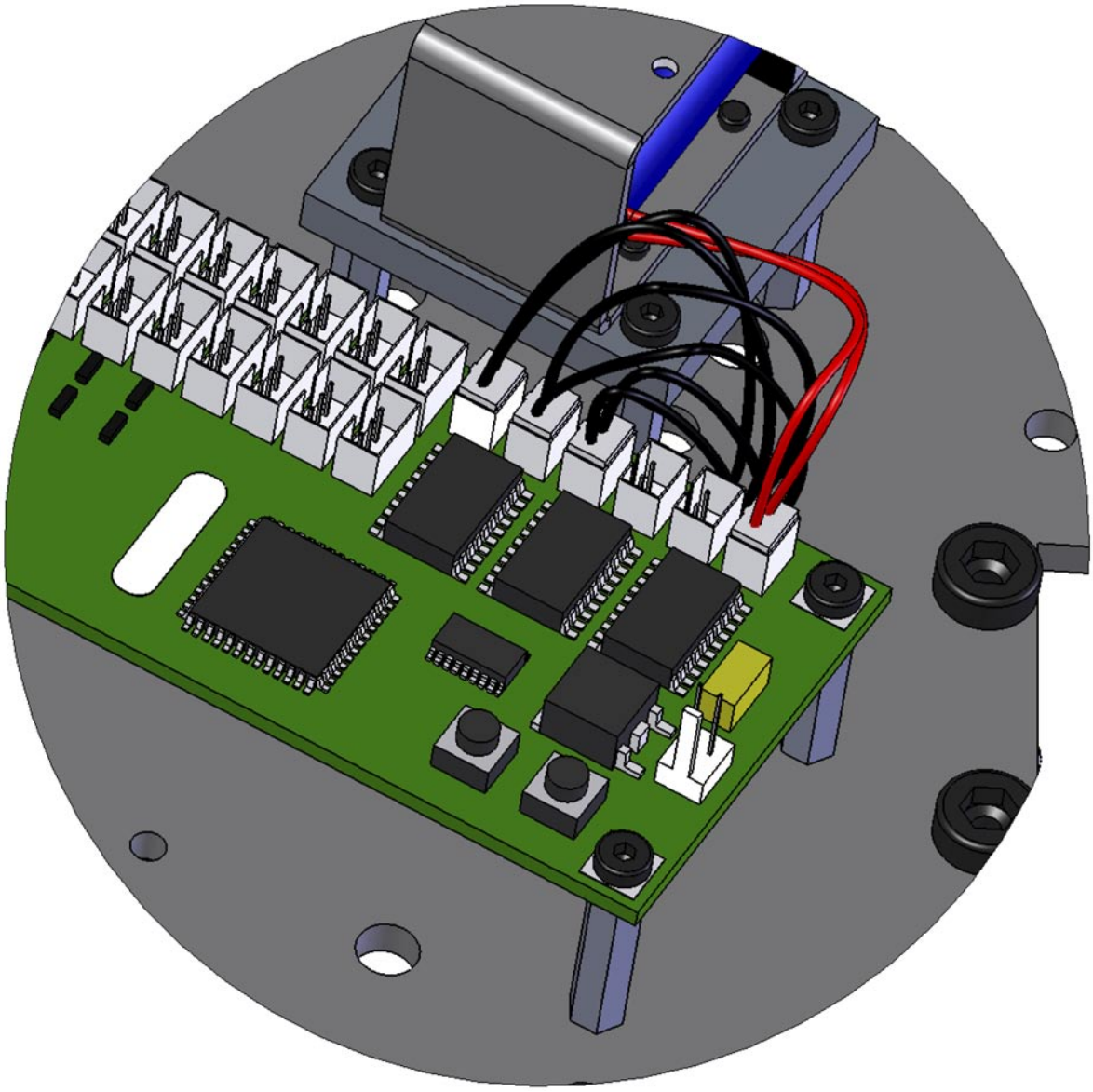
Vorgehensweise



Ablauf

- Die Motorkabel (schwarz markiert) durch die grösseren Bohrungen der Antriebsplattform nach oben in Richtung Controllerboard führen.
- Weissen Stecker von Motor „vorne links“ in Buchse „Mo0“ des Controllerboards einstecken, dabei die Codierung von Stecker und Buchse beachten.
- Weissen Stecker von Motor „vorne rechts“ in Buchse „Mo1“ des Controllerboards einstecken, dabei die Codierung von Stecker und Buchse beachten.
- Weissen Stecker von Motor „hinten“ in Buchse „Mo2“ des Controllerboards einstecken, dabei die Codierung von Stecker und Buchse beachten.
- Das Kickerkabel (rot markiert) in Buchse „Mo5“ des Controllerboards einstecken, dabei die Codierung von Stecker und Buchse beachten.

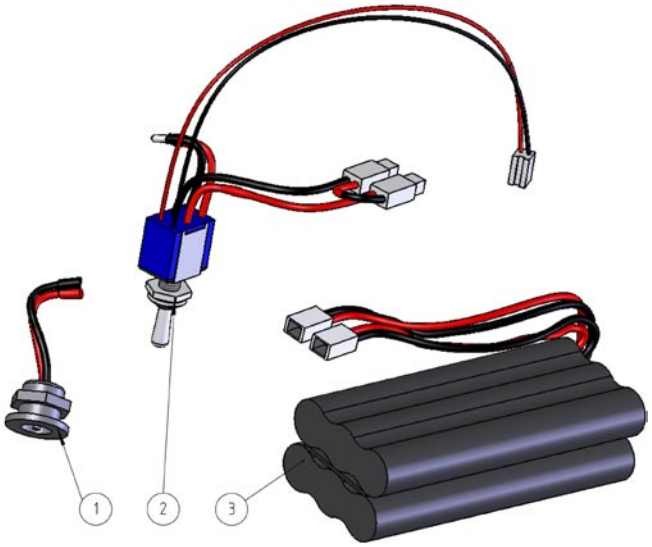
Fertig montiert

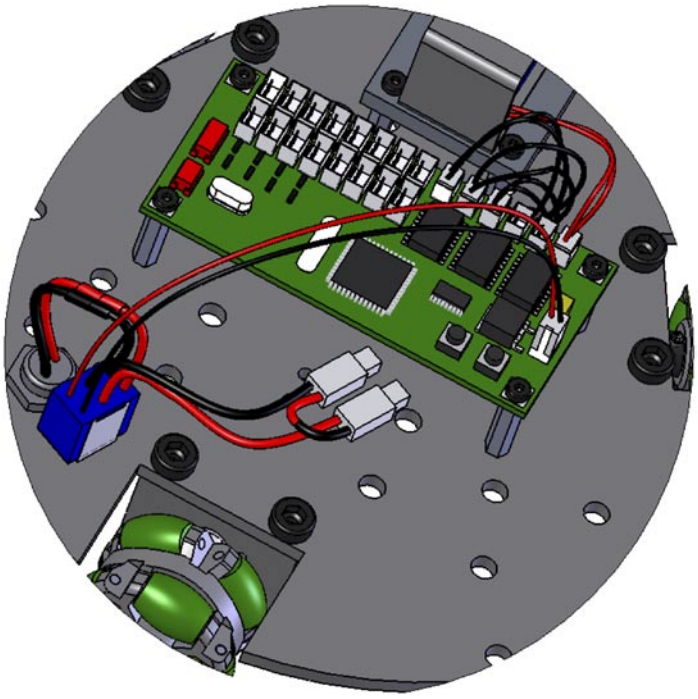


5.9 Kabelsatz montieren

Benötigte Komponenten - (Akku (3) ist nicht im Lieferumfang enthalten)

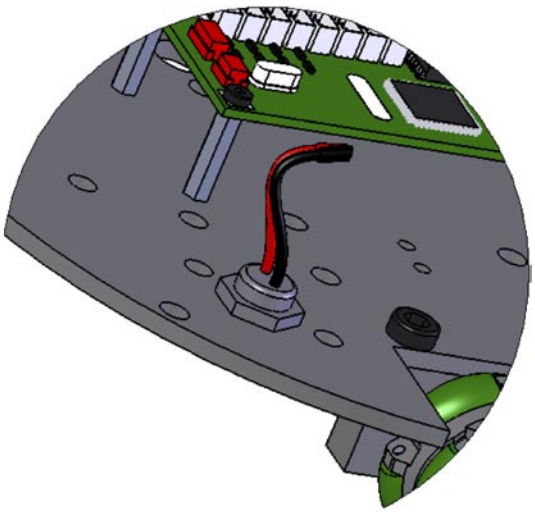
POS-NR.	BENENNUNG	MENGE
1	Ladebuchse	1
2	Hauptschalter	1
3	Akku NiMH 2 x 7,2V	1



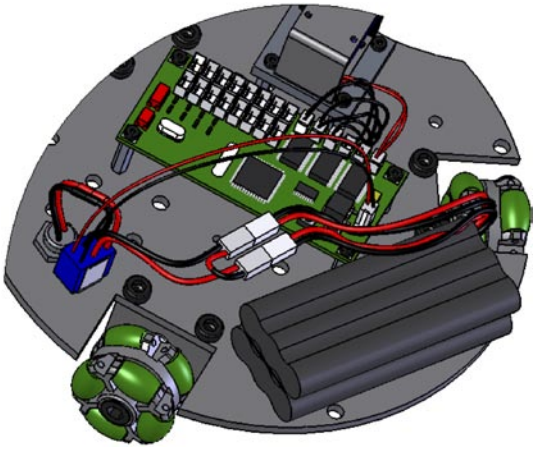


Ablauf

- Ladebuchse (1) mit den Kabeln von unten her durch die entsprechende, grössere Bohrung der Antriebsplattform führen und von oben verschrauben.



- Hauptschalter (2) von oben her durch die Bohrung neben der Ladebuchse (1) stecken und von unten verschrauben.



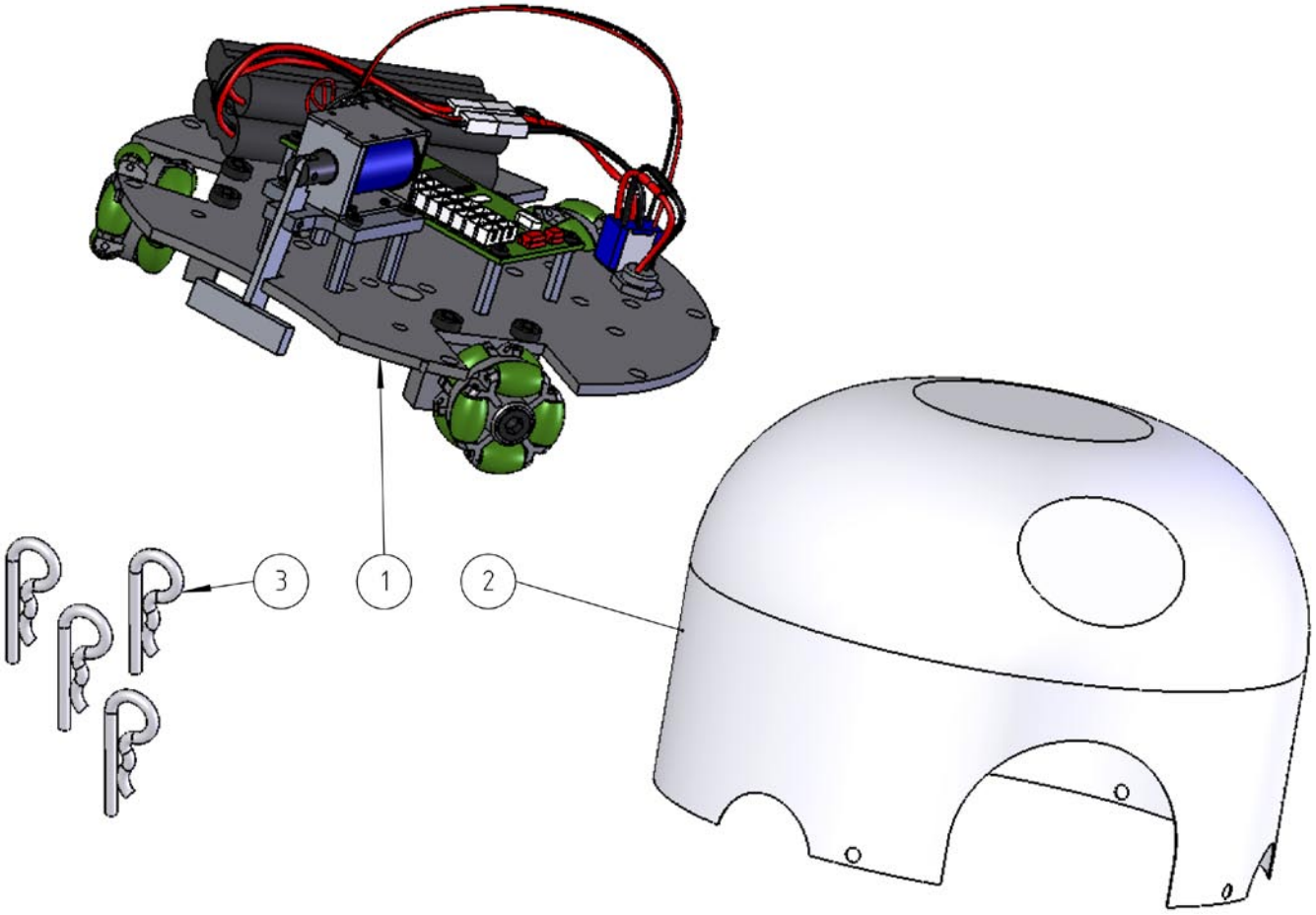
- Das rote und schwarze Kabel des Hauptschalter (2) mit denen der Ladebuchse (1) verbinden und dabei auf die richtige Farbcodierung achten.
- Das kleinere weisse Kabel des Hauptschalter (2) wird mit dem Controllerboard verbunden, dabei die Codierung von Stecker und Buchse beachten.
- Zuletzt kann ein Akku zur Stromversorgung angeschlossen werden. Dieser ist nicht im Lieferumfang enthalten, wir empfehlen jedoch die Best.-Nr. R7100 - Akku 2 x 6-NiMH1100, 7,2 V / 1,1 Ah.

5.10 Haube montieren

Benötigte Komponenten

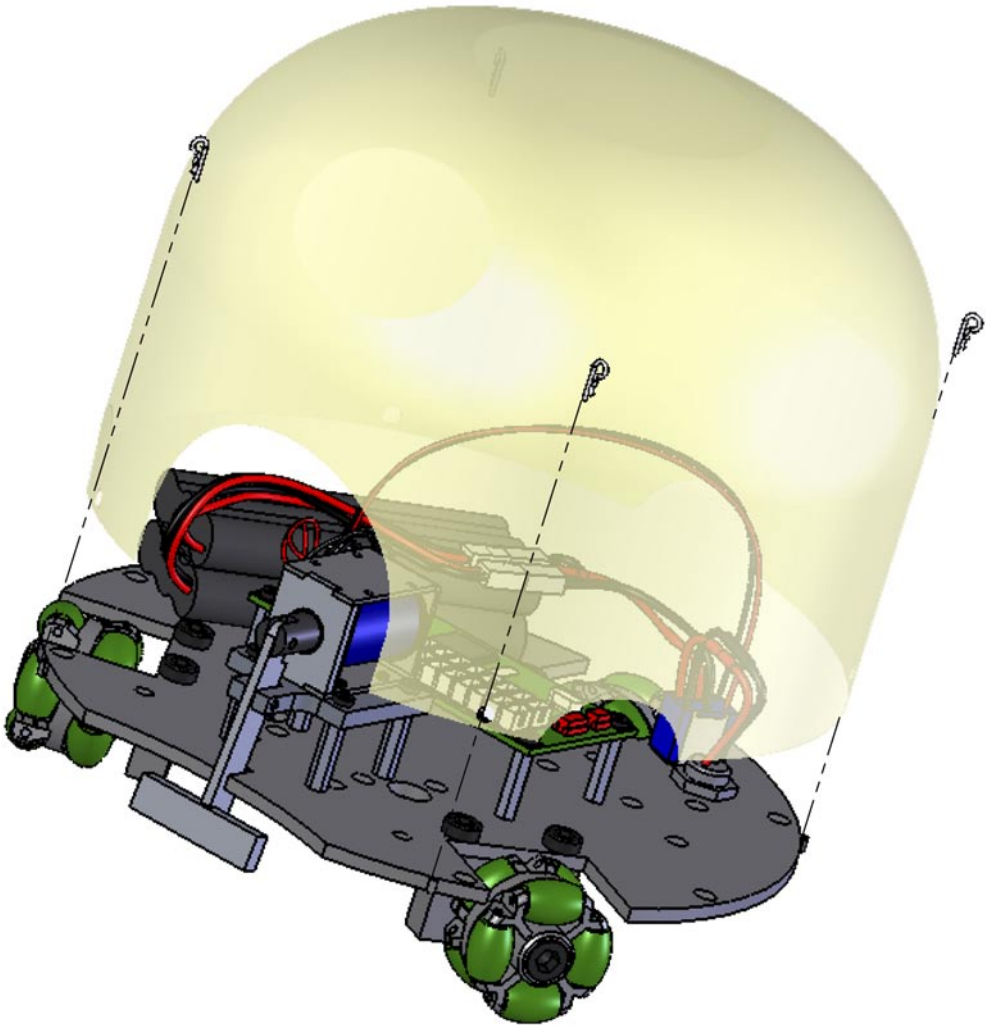
POS-NR.	BENENNUNG	MENGE
1	Antriebsplattform	1
2	Haube	1
3	Halteclip	4

Vorgehensweise

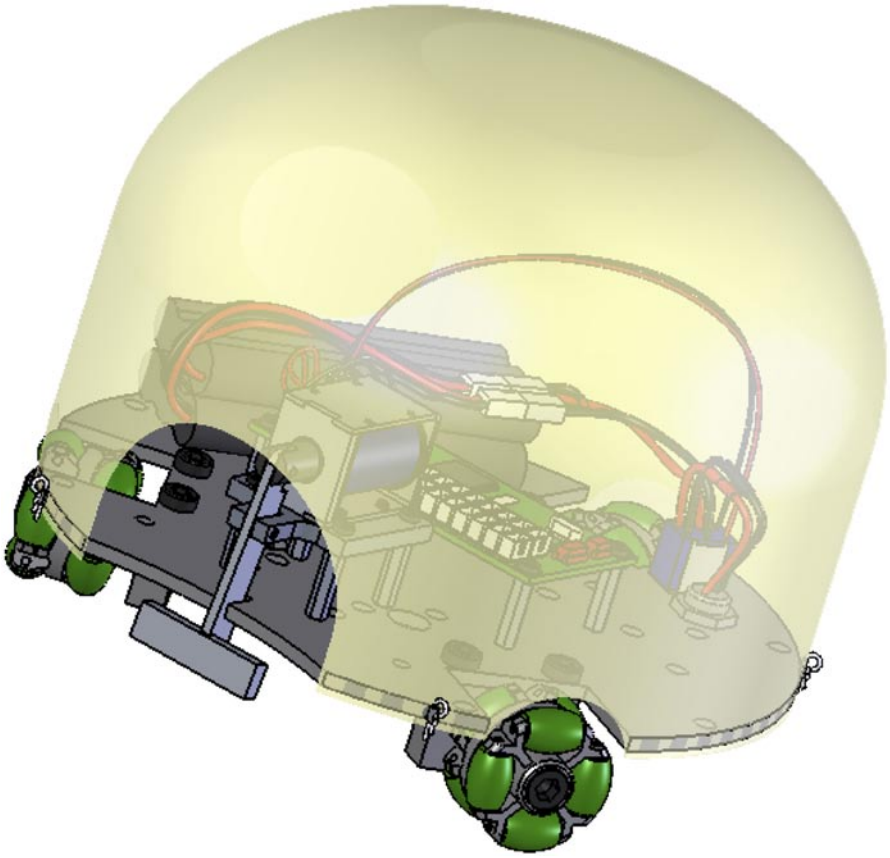


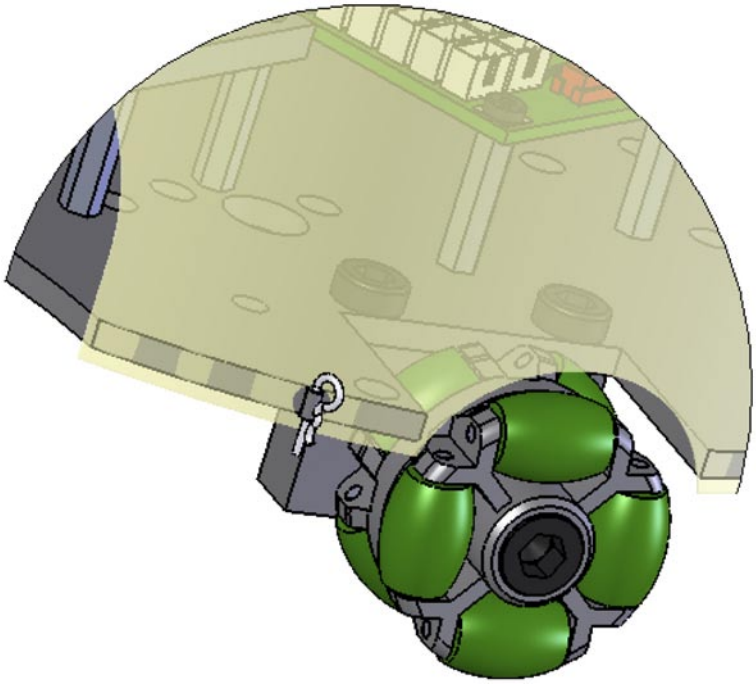
Ablauf

- Zuerst muss die Haube von den fertigungsbedingten Rändern befreit werden. Dies kann z.B. mit einer Schere erfolgen. Achten Sie darauf, den Markierungen genau entlang zu schneiden. Als erstes schneidet man am unteren Rand entlang im Kreis um die Haube herum. Danach werden die Ausschnitte der Räder und des Kickers gemacht. Die vier Löcher für die Halteclips am besten mit einem Aktenlocher an den vorgezeichneten Markierungen auslochen. Zuletzt werden die Schneidkanten mit einem feinen Schmirgelpapier bearbeitet.
- Nun kann die Haube (2) auf die Antriebsplattform (1) aufgesetzt und mittels der 4 x Halteclips (3) fixiert werden.



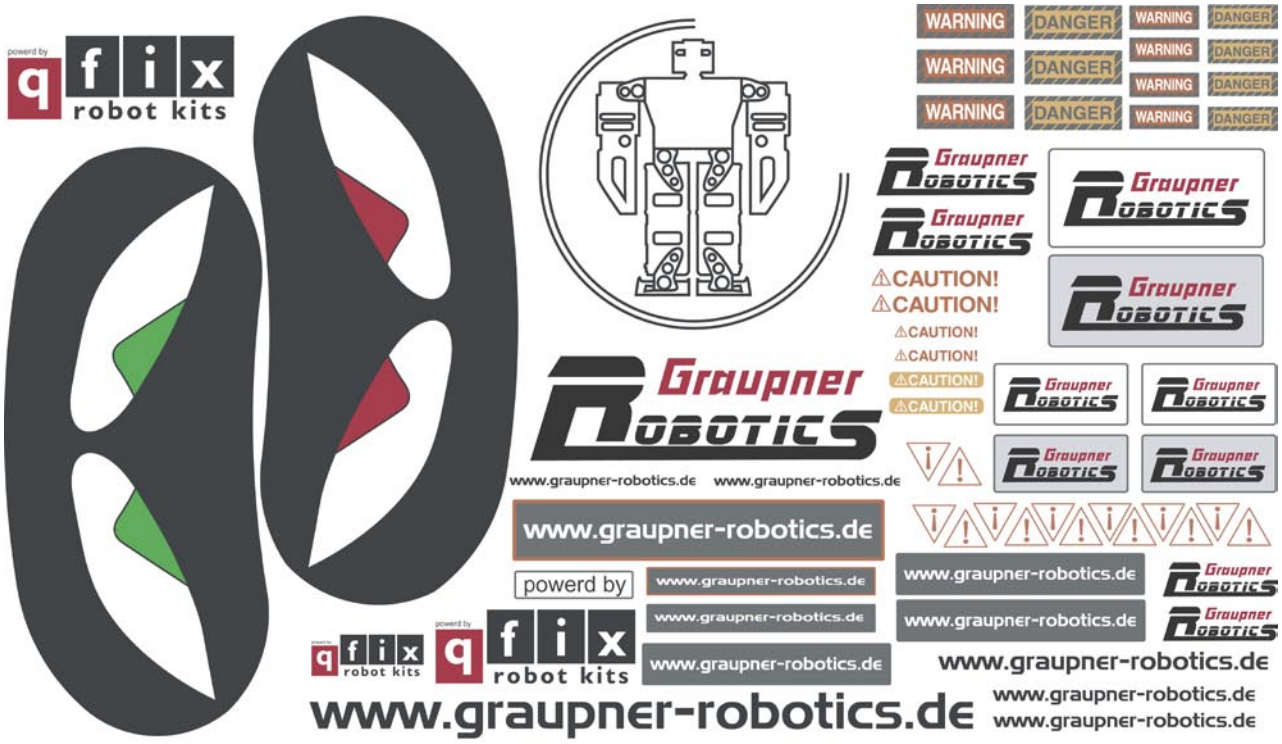
Fertig montiert





Bemerkungen

Die Haube kann nun nach Belieben beklebt bzw. bemalt werden. Beispiele für solche Dekorbögen sind im Downloadbereich der Homepage ***www.graupner-robotics.de*** zu finden.



6. Übungen am Standard Roboter

Die Aufgaben in diesem Kapitel basieren auf dem komplett aufgebauten RC-SOCCERBOT, wie in Kapitel 5 erläutert.

Es werden keine Erweiterungen benötigt.
Programme nur bei ausgeschaltetem Hauptschalter downloaden, da die Motorausgänge hierbei angesteuert werden könnten.

6.8 Motordrehzahl regeln

Aufgabe

Die Motorendrehzahlen des Roboters sollen durch die beiden Taster verändert werden. Taster 0 soll ein immer schneller werdendes Vorwärtsfahren realisieren, Taster 1 soll dieses wiederum verringern bis Rückwärts gefahren wird. Zu Beginn sollen alle Motoren aus sein. Zusätzlich soll ein Vorwärtsfahren mit der LED 0 und ein Rückwärtsfahren mit der LED 1 angezeigt werden.

Lösungsansatz

Das Programm arbeitet in einer Endlosschleife.

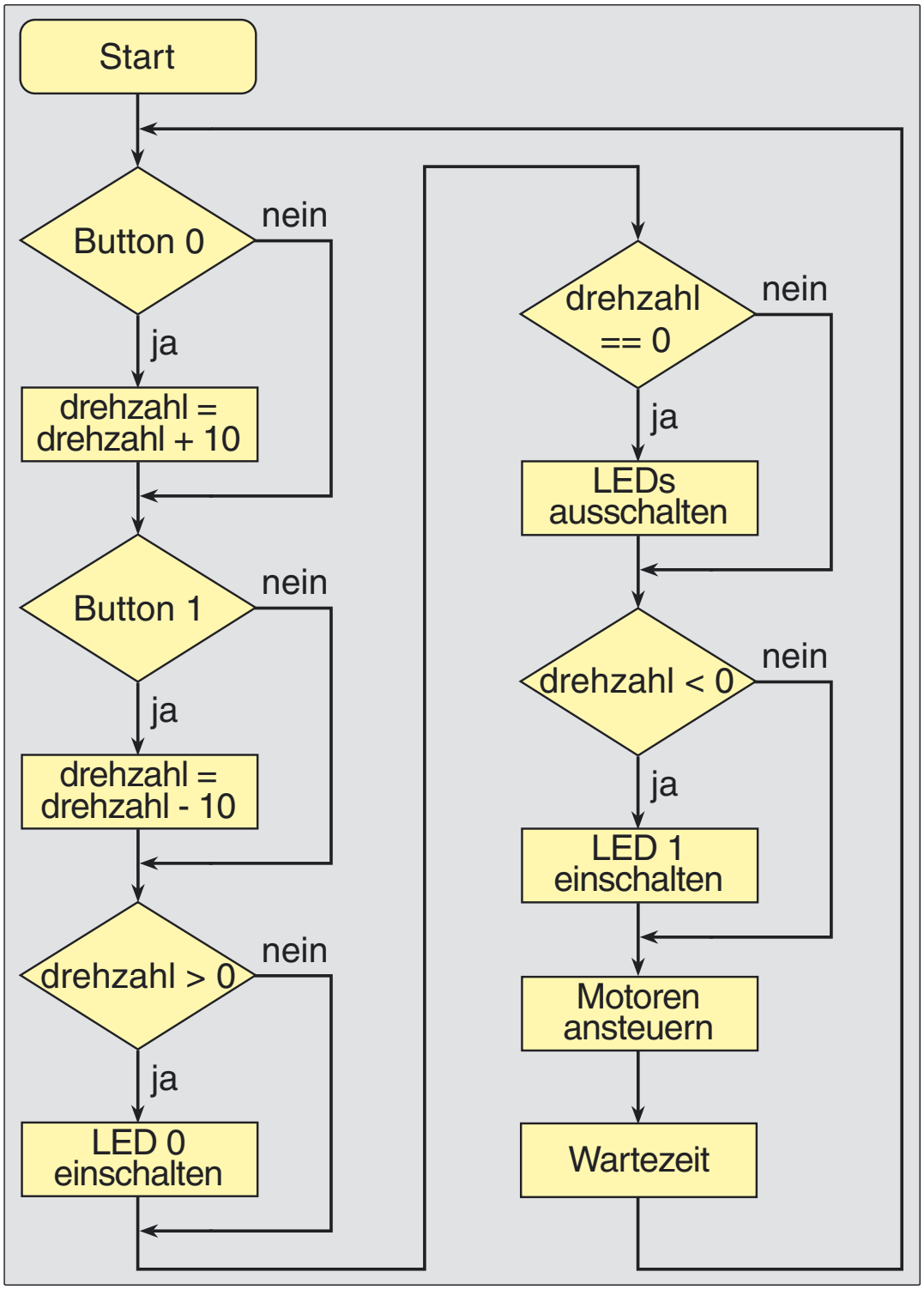
Für das An- und Ausschalten der LEDs können wieder die Befehle „ledOn“ und „ledOff“ benutzt werden.

Es wird eine Variable für die Motordrehzahl benötigt.

Die Motoren werden mit dem bekannten Befehl „motor“ angesteuert.

Die Auswertung der Taster erfolgt mittels einer „if“ Abfrage. Ebenso erfolgt die Auswertung der Variablen ob Vorwärts, Rückwärts oder gar nicht gefahren wird.

Flussdiagramm



Programm

```
#include "qfixSoccerBoard.h"

SoccerBoard robot;


int main()
{
  int drehzahl = 0;                          //Variable deklarieren

  while(true) {                              //Endlosschleife
    if (robot.button(0))                     //Drehzahlwert erhöhen
      {
      drehzahl = drehzahl + 10;
      }
    if (robot.button(1))                     //Drehzahlwert verringern
      {
      drehzahl = drehzahl - 10;
      }
    if (drehzahl > 0)                        //LEDs ansteuern
      {
      robot.ledOn(0);
      robot.ledOff(1);
      }
    if (drehzahl == 0)                       //LEDs ansteuern
      {
      robot.ledsOff();
      }
    if (drehzahl < 0)                        //LEDs ansteuern
      {
      robot.ledOff(0);
      robot.ledOn(1);
      }

      robot.motor(0, -drehzahl);             //Motoren ansteuern
      robot.motor(1, drehzahl);
      robot.msleep(150);                     //Wartezeit
  }
}
```

Erweiterungen

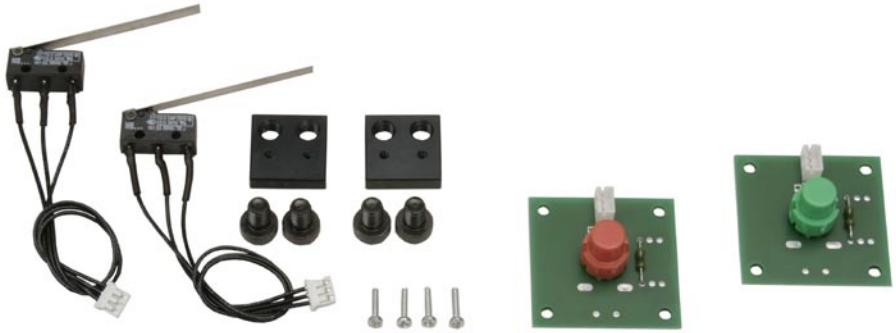
Im jetzigen Programm ist eine ständige Erhöhung bzw. Verringerung der Drehzahl auch über +/- 255 möglich. Wird nun ein Taster sehr lange betätigt, wird ebenfalls der Drehzahlwert weit grösser als +/- 255. Dies soll durch eine entsprechende Programmierung verhindert werden, d.h. die Variable „drehzahl“ soll nur im Bereich +/- 255 verändert werden können.

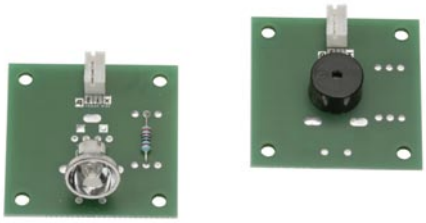
```
    if (robot.button(0))                //Drehzahlwert erhöhen
      {
      drehzahl = drehzahl + 10;

      if (drehzahl > 255)               //Drehzahlwert prüfen und korrigieren
        {
        drehzahl = 255;
        }
      }
```

Ab einem bestimmten Wert der Drehzahl soll der Kicker kurz aktiviert werden. Der Reiz dieser Übung besteht darin, den Kicker nur eine kurze Zeit anzusteuern und nicht dauerhaft zu aktivieren wenn der Schwellenwert überschritten bleibt. Der Roboter soll während dieser Zeit zudem trotzdem noch bedienbar bleiben.

```
    if (drehzahl > 78)                  //Drehzahlschwelle erreicht
      {
      if (variable)                     //Prüfen ob aktiviert werden soll
        {
        robot.motor(5, 255);            //Kicker ansteuern
        variable = false;               //Variable zurücksetzen
        }
      }
    else
      {
      variable = true;                  //Variable setzen
      }
```

Die beiden Taster sollen zusätzlich noch den dritten Motor entsprechend beeinflussen um unterschiedliche Fahrmuster zu erhalten. Auch kann dies wiederum erst ab einer bestimmten Drehzahl erfolgen.





7. Übungen mit Erweiterungen

Vorab:
Bei den Übungen in diesen Kapiteln handelt es sich um Aufgaben, für die zusätzliche Hardware benötigt wird. Welche Bauteile jeweils benötigt werden, wird am Anfang des entsprechenden Kapitels beschrieben. Um diese Komponenten an den Roboter anzubauen wurde versucht so wenig wie möglich zusätzliche Bauteile zu verwenden. Natürlich können eigene Konstruktionen verwendet werden.

Die folgende Tabelle beschreibt die zusätzlich verwendeten Komponenten sowie ihr Vorkommen in den Kapiteln. Unser Sortiment beinhaltet noch weitere Komponenten um eigene Aufgaben realisieren zu können.

Kapitel	Best.-Nr.	Bezeichnung
1.	R7100	Akku 2 x 6-NiMH1100, 7,2 V / 1,1 Ah
1.	R7310	USB Download Adapter mit Kabel
4.7	R3101	Taster Grün
7.1	R3010	2 x Infrarot Sensor Set (4 cm - 30 cm)
7.2	R3020	Linien Sensor Set
7.3	R3040	Bumper Set
7.4	R3090	2 x Licht Sensor
7.4	R7200	Sensorkabel Set 3 Stk.
7.5	R5100	LC-Display Board
7.5	R3070	Potentiometer
7.5	R3080	Temperatur Sensor
7.5	R3090	Licht Sensor
7.5	R7200	Sensorkabel Set 3 Stk.
7.6	R5000	8 x Power LED
7.6	R7201	Sensorkabel Set 10 Stk.
7.6	R6200	2 x Abstandhalter 10 Stk. M3
7.6	R6102	2 x Schrauben Set 10 Stk. M3x6

Programme nur bei ausgeschaltetem Hauptschalter downloaden, da die Motorausgänge hierbei angesteuert werden könnten.

Am Ende eines jeden Kapitels ist ein Montagebeispiel mit verschiedenen Abbildungen zu finden. Diese sind als Vorschlag zu verstehen und können natürlich durch eigene Ausführungen ersetzt oder erweitert werden.

7.2 Linie verfolgen

Aufgabe

Der Roboter soll mit Hilfe des Liniensensors einer schwarzen Linie folgen. Die Linie wird mit schwarzem Klebeband auf einer weißen Fläche vorgegeben.

Zusätzliche Bauteile

- Linien Sensor Set Best.-Nr. R3020

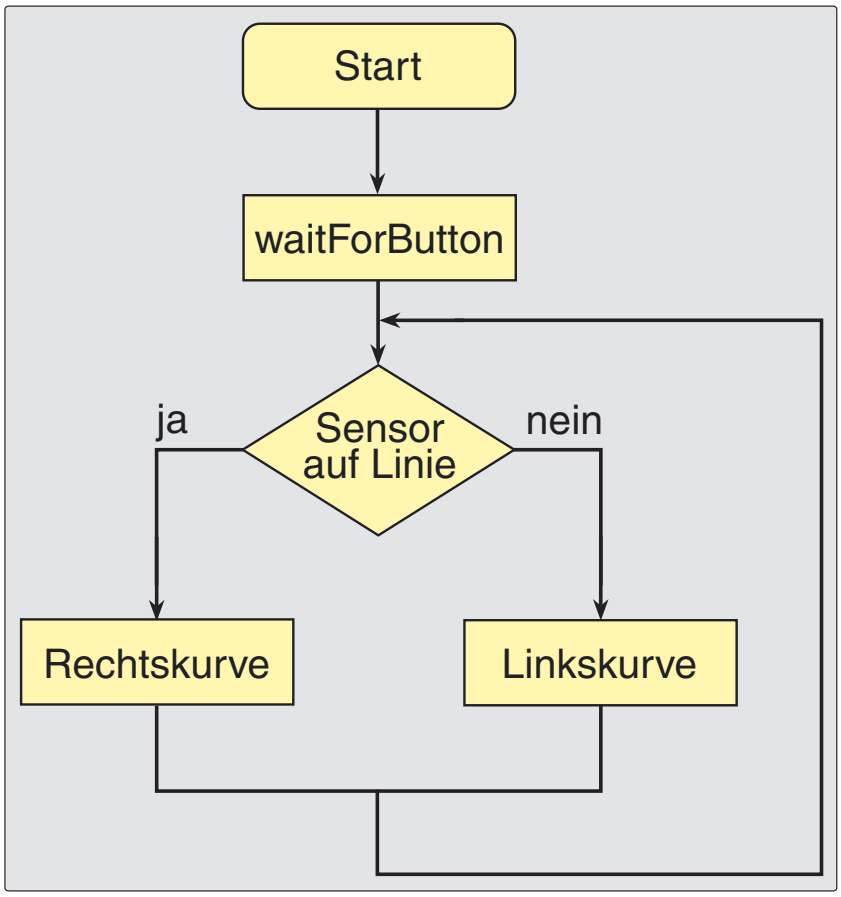
Lösungsansatz

Für diese Aufgabe soll der Liniensensor an den Roboter montiert werden, mit dessen Hilfe dieser die Helligkeit des Bodens abtasten kann.

Die einfachste Art, einer Linie entlang zu fahren ist es, wenn der Roboter auf der Linie leicht nach rechts fährt und bei Verlassen der Linie wieder leicht nach links. Auf diese Weise folgt er schlängelnd der Linienkante.

Das Abfragen des Sensors erfolgt über den Analog-Eingang z.B. mittels ***analog(3)***, falls der Liniensensor an „An3“ angeschlossen wurde. Die Entscheidungsschwelle (engl. „threashold“), also der Wert, bei dem sich der Untergrund von weiß nach schwarz ändert, muss experimentell ermittelt werden und ist von der Bodenfarbe und dem Lichverhältnis abhängig.

Flussdiagramm



Programm

```
#include "qfixSoccerBoard.h"

const int THREASHOLD = 100;

SoccerBoard robot;


void waitForStart()
{
  robot.ledOn(0);
  robot.waitForButton(0);
  robot.ledOff(0);
}


int main()
{
  waitForStart();                          //auf Startknopf warten

  while(true) {                            //Endlosschleife
    if (robot.analog(3) > THREASHOLD) {    //auf Linie?
      robot.motor(0, -155);                //Rechtskurve
      robot.motor(1, 55);
      robot.motor(2, -45);
    }
    else {
      robot.motor(0, -55);                 //Linkskurve
      robot.motor(1, 155);
      robot.motor(2, 45);
    }
  }
}
```

Bemerkungen

Die Schwelle „THREASHOLD“ mit dem Wert 100 muss experimentell ermittelt und entsprechend verändert werden.

Erweiterungen

Eine Kalibrierung des Schwellwertes bietet sich an. Direkt nach Anlegen der Betriebsspannung wird der Sensor auf den weißen Untergrund gestellt und durch Drücken des Tasters 0 bestätigt, dass ein Helligkeitswert für „weiß“ gemessen werden kann. Nun wir der Sensor auf die schwarze Linie gestellt und nach Drücken von Taster 1 der Wert für „schwarz“ eingelesen. Die Entscheidungsschwelle ergibt sich dann durch den Durchschnitt: schwelle = (schwarz+weiß)/2.

Als Erweiterung der Aufgabe können zusätzliche Sensoren eingesetzt werden. Beispielsweise kann mit zwei Sensoren eine „schönere“ Fahrweise erzeugt werden.

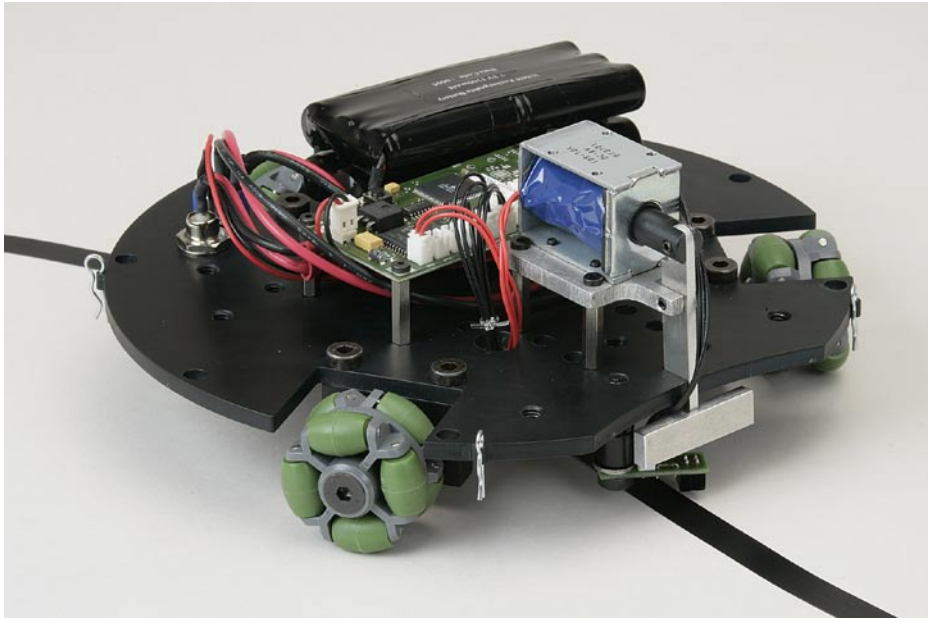
Wird der Liniensensor z.B. über einen Ausleger weiter vorne am Roboter montiert, ergeben sich komplett neue Werte für die Motoransteuerung sowie Fahrmuster.

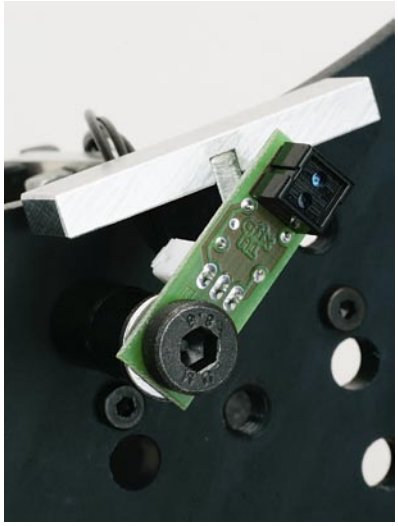
Neue Geschwindigkeiten für die Motoren sowie Differenzen zwischen den Motoren können ausprobiert werden um neue Fahrmuster zu entwickeln.

Abstandssensoren können zusätzlich eingesetzt werden, um beispielsweise auf der Linie stehende Hindernisse zu detektieren und vorher anzuhalten.

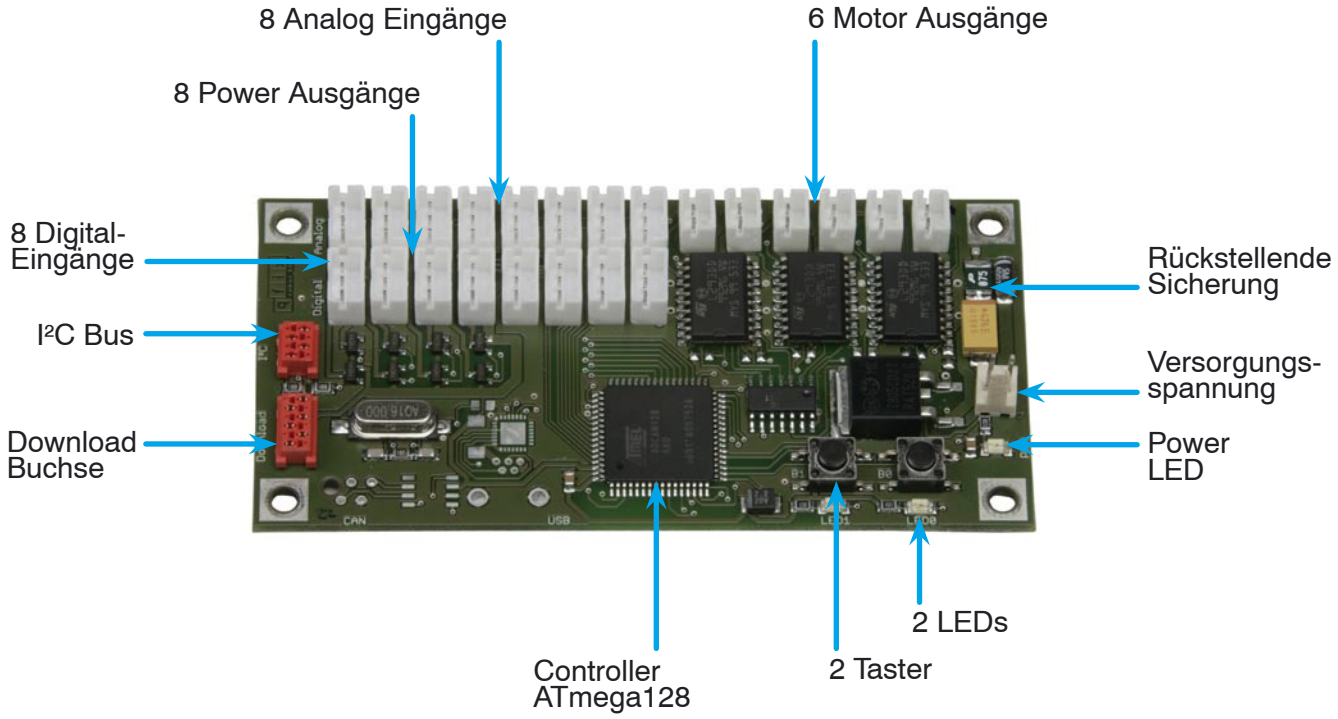
- Infrarot Sensor Set (4cm - 30cm) Best.-Nr. R3010
- Infrarot Sensor Set (10cm - 80cm) Best.-Nr. R3011

Montagebeispiel





8.1 SoccerBoard Beschaltung



Controller

- Atmel ATmega128 Controller.
- 16 MHz Takt für 16 Mio. Instruktionen pro Sekunde.
- 128 KB Flash Speicher für Programme.
- 4 KB RAM Speicher für Daten.
- 4 KB EPROM Speicher für nichtflüchtige Daten.
- I²C Bus für Erweiterungen (z.B. LC-Display, ...).

I/O Ports

- 2 x Taster (on board).
- 2 x LED (on board).
- 8 x Analog-Eingänge für analoge Sensoren (z.B. Distanzsensoren, Liniensensoren).
- 8 x Digital-Eingänge für digitale Sensoren (z.B. Fühler, Bumper, Not-Aus).
- 6 x Motor-Ausgänge für z.B. Gleichstrommotoren bis 600mA.
- Eingänge sind zusätzlich per Software als Ausgang schaltbar (TTL-Pegel).

Sonstiges

- Spannungsversorgung: 7 - 12V mit Betriebsanzeige über Power LED
- Selbstrückstellende Sicherung

Taster

Taster	Controller - Pin
T0	PG4
T1	PG3

8. Anhang A Beschaltung
8.1 SoccerBoard Beschaltung

Sensor-Eingänge
Die Schalt-Ausgänge schalten 5V bei maximal 100 mA.

+5V (schaltbar)
GND (Masse)
Signal

Buchse	5V - Ausgang	Signal
An0	PC0	PF0
An1	PC1	PF1
An2	PC2	PF2
An3	PC3	PF3
An4		PF4
An5		PF5
An6		PF6
An7		PF7
Di0	PC4	PA0
Di1	PC5	PA1
Di2	PC6	PE2
Di3	PC7	PE3
Di4		PE4
Di5		PE5
Di6		PE6
Di7		PE7

LEDs

LED	Controller - Pin
L0	PB0
L1	PB2

I²C Bus

1 SDA (Datenleitung)
2 Vcc (Board-Spannung)
3 GND (Masse)
4 SCL (Clock-Leitung)

Bus - Pin	Controller - Pin
1 SDA	PC1
4 SCL	PC0

Download Buchse
Belegung entspricht Atmel IDC-6 Standard

1 MISO
2 Vcc (5V)
3 SCK
4 MOSI
5 Reset
6 GND (Masse)

Buchse - Pin	Controller - Pin
1 MISO	PB6
3 SCK	PB7
4 MOSI	PB5
5 Reset	RESET

9.1 Globale Funktionen

Globale Funktionen		
RWert	**Funktionsname**	**Erklärung**
int	***abs(int a)***	Liefert den Absolutwert von ***a*** zurück.
void	***sleep(int s)***	Wartet ***s*** Sekunden. (Die globale Wartezeit ist etwas ungenau.)
void	***msleep(int ms)***	Wartet ***ms*** Millisekunden.

Die globalen Funktionen können direkt (also ohne ein Objekt) aufgerufen werden.

Beispiel: ***sleep(5);*** wartet 5 Sekunden.

9.2 Klasse SoccerBoard

Die folgende Tabelle listet die Funktionen (oder „Methoden“) der Klasse „SoccerBoard“ auf. Um diese Methoden zu benutzen, muss ein Objekt dieser Klasse existieren, z.B. „robot“. Siehe hierzu das Standard-Programm in Anhang C. Die Methoden werden mit Angabe des Objektes aufgerufen.

Beispiel: ***robot.ledOn(1);*** schaltet die LED 1 an.

Klasse: SoccerBoard		Datei: ***qfixSoccerBoard.h***
RWert	**Funktionsname**	**Erklärung**
	SoccerBoard()	Diese Methode wird durch das Anlegen des Objektes automatisch aufgerufen („Konstruktor“) und initialisiert das Objekt folgendermaßen: Die Motoren werden auf 0 gesetzt, alle LEDs sind aus, alle Power-Ausgänge sind angeschaltet.
void	***ledOn(int i)***	Schaltet die LED mit Index ***i*** an. ***i*** muss im Bereich 0 bis 1 liegen.
void	***ledOff(int i)***	Schaltet die LED mit Index ***i*** aus. ***i*** muss im Bereich 0 bis 1 liegen.
void	***ledsOff()***	Schaltet alle LEDs aus.
void	***led(int i, bool state)***	Schaltet die LED mit dem Index ***i*** an, falls ***state=true***, bzw. aus, falls ***state=false***. ***i*** muss im Bereich 0 bis 1 liegen.
bool	***button(int i)***	Liefert ***true*** zurück, falls der Button mit dem Index ***i*** gedrückt ist, ansonsten ***false***. i muss im Bereich 0 bis 1 liegen.
void	***waitForButton(int i)***	Wartet, bis der Button mit dem Index ***i*** gedrückt und wieder losgelassen wurde. ***i*** muss im Bereich 0 bis 1 liegen.
void	***motor(int i, int speed)***	Setzt den Motor mit dem Index ***i*** auf den Wert ***speed***, der im Bereich -255 bis +255 liegen muss. ***i*** muss im Bereich 0 bis 5 liegen.
void	***motorsOff()***	Schaltet alle Motoren aus.
int	***analog(int i)***	Liefert den Wert des Analog-Eingangs mit dem Index ***i*** zurück. Der Rückgabewert liegt im Bereich 0-255. ***i*** muss im Bereich 0 bis 7 liegen.
bool	***digital(int i)***	Liefert ***true*** zurück, falls der Digital-Eingang mit dem Index ***i*** High ist, ansonsten ***false***. ***i*** muss im Bereich 0 bis 7 liegen.
void	***powerOn(int i)***	Schaltet den Power-Ausgang mit Index ***i*** an. ***i*** muss im Bereich 0 bis 7 liegen.
void	***powerOff(int i)***	Schaltet den Power-Ausgang mit Index ***i*** aus. ***i*** muss im Bereich 0 bis 7 liegen.
void	***power(int i, bool state)***	Schaltet den Power-Ausgang mit dem Index ***i*** an, falls ***state=true***, bzw. aus, falls ***state=false***. ***i*** muss im Bereich 0 bis 7 liegen.
void	***sleep(int s)***	Wartet ***s*** Sekunden. (Genauer als die globale Wartezeit.)
void	***msleep(int ms)***	Wartet ***ms*** Millisekunden. (Genauer als die globale Wartezeit.)

9.3 Klasse LCD

Die Klasse LCD unterstützt den Betrieb des qfix LCD Displays. Das Display muss über den I²C-Bus an das Controllerboard angeschlossen sein.

Klasse: LCD		Datei: ***qfixLCD.h***
RWert	**Funktionsname**	**Erklärung**
	LCD()	Der Konstruktor initialisiert die Verbindung zum LC-Display, löscht das Display und setzt den Cursor auf (0, 0).
void	***clear()***	Löscht das Display.
void	***print(int i)***	Druckt die Zahl ***i*** an der aktuellen Cursor-Position aus.
void	***print(char* s)***	Druckt den String ***s*** an der aktuellen Cursor-Position aus.
void	***print(int row, int col, int i)***	Druckt die Zahl ***i*** in Zeile ***row*** in Spalte ***col*** aus.
void	***print(int row, int col, char* s)***	Druckt den String ***s*** in Zeile ***row*** in Spalte ***col*** aus.
void	***lightOn()***	Schaltet die Hintergrundbeleuchtung an.
void	***lightOff()***	Schaltet die Hintergrundbeleuchtung aus.

9.4 Übersicht der Wertebereiche

In den obigen Tabellen beschreibt die Spalte **RWert** für jede Methode den Typ des Rückgabewerts.

Folgende Typen werden hierbei verwendet:

Wertebereich	Erklärung
void	Kein Wert (wird z.B. bei Funktionen oder Methoden ohne Rückgabewert benutzt).
bool	Boolscher Wert (***true*** oder ***false***).
int	Ganzzahliger Wert (-32767 bis 32768).
char	Zeichen (z.B. ***´A´***).
char*	Zeiger auf ein Zeichen. Steht dies in einer Funktion, kann als Parameter ein String übergeben werden (z.B. ***"Hallo"***).

10.1 Standard-Programm

```
//     ****************************************
//     **              RC-SOCCERBOT            **
//     **    Uebungen / Lehrerbegleitbuch      **
//     **                                      **
//     **      Autor: Hannes Runknagel         **
//     **  Copyright: 2007 by GRAUPNER         **
//     **  www.graupner-robotics.de            **
//     ****************************************

#include “qfixSoccerBoard.h“

SoccerBoard robot;

int main()
{
  /* Hier stehen die Anweisungen */
}
```

Das Standard Programm kann als Ausgangspunkt für alle Übungen benutzt werden. Es lädt zuerst die benötigte Bibliothek ***qfixSoccerBoard.h*** und legt dann ein Objekt ***robot*** an, das für die weiteren Befehle benutzt werden kann.

Durch den Ausdruck ***robot.<Befehl>;*** können so sämtliche Befehle der Klasse ***SoccerBoard*** (siehe Anhang B) ausgeführt werden.

Henriettenstr. 94-96, D-73230 Kirchheim/Teck Germany

eMail: info@graupner.de
Internet: www.graupner-robotics.de