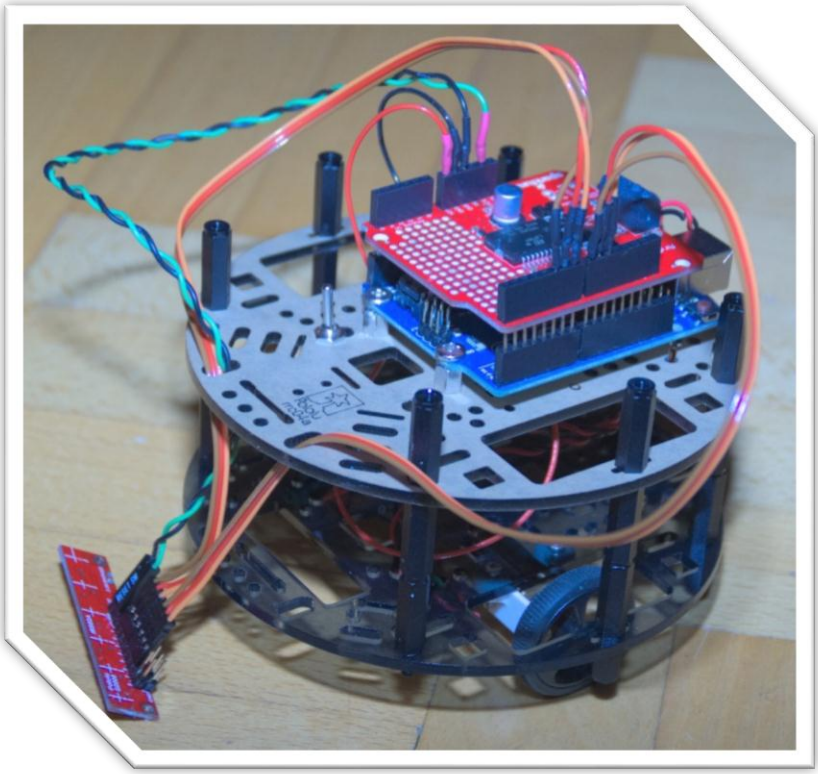
Projektarbeit BIM 2012
Arduino LINE-TRACKER Roboter

Datum: 23.09.2012



Thema: Projektvorstellung

Version: 1.0.0

© 2012 - by M.Koch

Inhaltsverzeichnis

Inhalt

Aufgabe:

Nach mehreren Überlegungen und vielen Ideen sind wir nun zu dem Schluss gekommen, dass wir uns mit dem Thema Robotik mit einem Microcontroller („Arduino“) beschäftigen werden.

Als Beispiel für den praktischen Einsatz eines solchen Systems haben wir einen Line-Tracking entwickelt, welcher mit Hilfe von Infrarot Sensoren seine Position über einer Linie (Schwarze Linie auf weisem Grund) bestimmt und versucht immer über dieser in Mittelposition zu bleiben.

Ziele:

- Kennenlernen verschiedener Datentypen und deren Wertebereiche
- Ansteuerung und auslesen digitaler und analoger Ports eines MCs
- Kennenlernen des Arduino-Systems und dessen Möglichkeiten (Input, Output, …)
- Auswerten und weiterverarbeiten von empfangenen Daten
- Programmiersprache C++ kennenlernen
- Einfache Programmstruktur und Verzweigungen kennenlernen

Was ist ein Line-Tracking Roboter?

Die Aufgabe eines solchen Roboters besteht darin, dass er einer vorgegeben Linie, meist schwarze Linie auf weisem Untergrund, abzufahren und so gut wie möglich in der Mitte zu bleiben.

Warum sollte man einen Line-Tracking Roboter bauen?

Sie erlernen, wie Sensordaten über digitale Ports eingelesen und die daraus resultierenden Werte wieder mit Hilfe von Digital oder Analog Ports ausgegeben werden können. Der Roboter soll dann in der Lage sein, die Sensorwerte mit Grenzwerten zu vergleichen und je nach Abweichung die notwendige Aktion ausführen.

Der Microcontroller

Die Hardware basiert auf einem Atmel AVR-Mikrocontroller aus der megaAVR-Serie. Dieser wird über USB oder eine externe Spannungsquelle mit 5 Volt versorgt und verfügen über einen 16 MHz-Quarzoszillator.

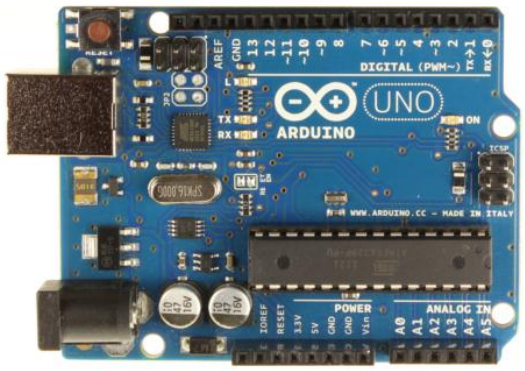


Abbildung 1: Arduino Uno

Das Arduino-Board stellt die Digital-Pins des Mikrocontrollers zur Nutzung für elektronische Schaltungen zur Verfügung. Die aktuellen Boards bieten 14 digitale I/O-Pins, von denen sechs PWM-Signale ausgeben können und sechs als analoge Eingänge dienen können. Für die Erweiterung werden sogenannte „Shields“ angeboten, die auf das Arduino-Board aufsteckt werden können. Es können aber auch Steckplatinen für den Aufbau von Schaltungen verwendet werden.

Benötigte Hardware

Pos.	MENGE	BESCHREIBUNG	STÜCKPREIS	BESTELLNUMMER	BETRAG
1	1	Arduino UNO rev.3	€ 25,80	A000066	€ 25,80
2	2	5" Robot Chassis RRC04A transparent	€ 6,86	1506	€ 13,72
3	2	GM18 30:1 Mini Getriebemotor	€ 15,00	GM18	€ 30,00
4	1	Getriebemotor Halterung (Set 2 Stk.)	€ 4,95	1089	€ 4,95
5	1	Kunststoffräder 32mm (Set 2Stk.)	€ 6,15	1087	€ 6,15
6	1	QTR-8RC IR Abstandssensor	€ 12,47	961	€ 12,47
7	1	Ardumoto - MotorDriver-Shield	€ 21,57	DEV-09815	€ 21,57
8	1	Ball Caster mit 12,7mm Metallkugel	€ 3,49	953	€ 3,49
					€ -
				Zwischensumme	*€ 118,15*
				Zusatzmaterial ca.	*€ 15,00*
				Summe	***€ 133,15***

Alle oben genannten Bauteile können online bei der deutschen Firma „Watterott.com“ bestellt werden. Link: http://www.watterott.com/

Zusatzmaterial / Montagematerial

Zusätzlich werden noch einige Kabel, so wie Abstandshalter (Distanzbolzen oder Gewindestange M3 + Muttern) und Montageschrauben benötigt.
Passende Quellen: Conrad-Electronic, Pollin-Electronic

Benötigte Software

Zur Programmierung des Arduino-Systems wird die speziell dafür entwickelte Open Project IDE „**Arduino 1.0**“ benötigt welche kostenfrei unter folgender Adresse heruntergeladen werden kann: http://arduino.cc/en/Main/Software

Um die Programmierung und den Empfang der Sensorleiste „**QTR-8RC**“ zu erleichtern gibt es bereits eine fertige **Library** welche in ein neues Programm inkludiert werden kann zum freien Download unter: https://github.com/pololu/qtr-sensors-arduino/zipball/master
Die Library ist auch vom Programmierer offen zugänglich und kann je nach Bedarf angepasst werden, dies wird jedoch in unserem Fall nicht nötig.

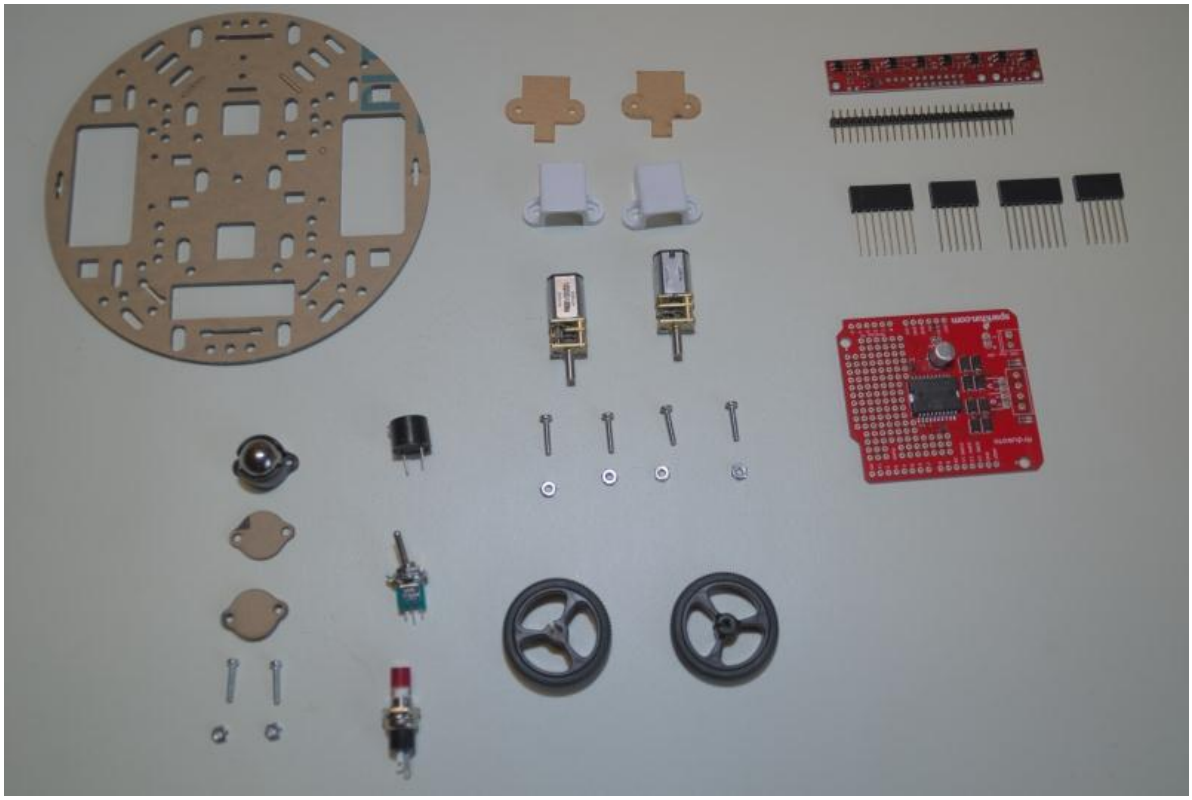
Für eine genaue Planung der Schaltungen kann die ebenfalls frei verfügbare Software „Eagle“ (http://www.cadsoftusa.com/downloads/?language=en) oder die Software „Fritzing“ (http://fritzing.org/download/) verwendet werden.

Benötigtes Werkzeug

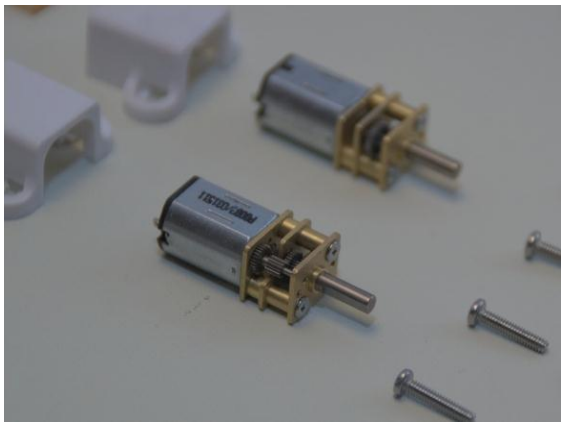
- Lötstation (regelbar)
- Lötzinn
- Platinenhalter-Löthilfe
- Seitenschneider
- Kombizange
- Gabelschlüssel (4mm)
- Abisolierzange
- Heißluftfön (alternativ ein Feuerzeug)
- Schraubenzieher (Kreuz-Schlitz)
- Akkuladegerät (optional, nur bei Akkubetrieb nötig)

Der Bauplan in Bildern

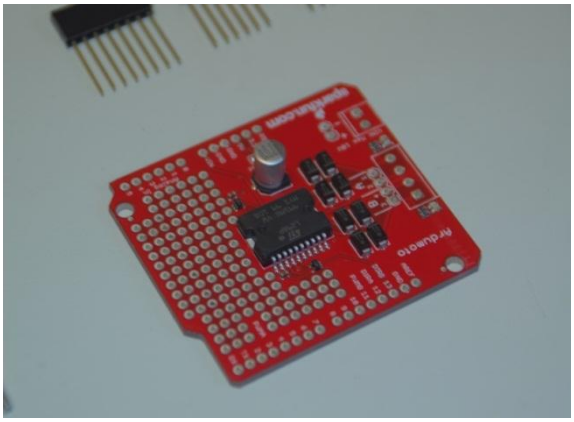
Da Bilder meist aussagekräftiger sind als viele Wort haben wir uns entschieden, den Zusammenbau des Eigenbau-Roboters in einer Bilderserie zu zeigen.



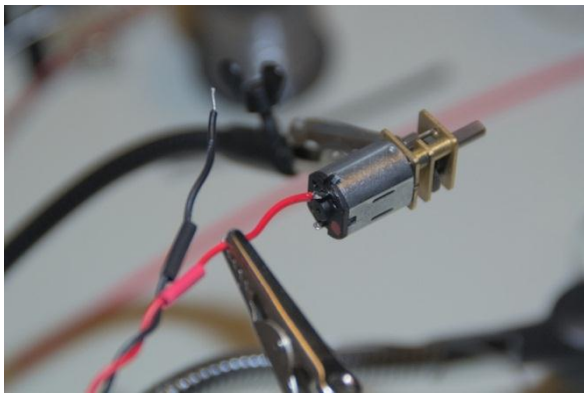
Übersicht der minimal benötigten Bauteile.



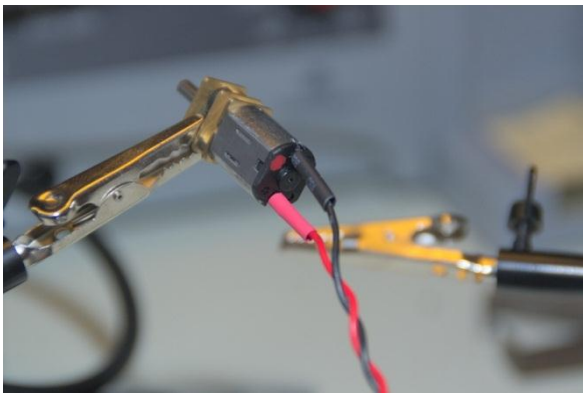
DC Vollmetall Getriebemotor 30:1



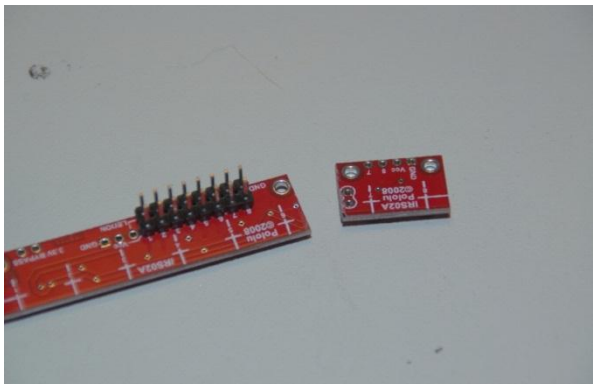
Motortreibermodul für Arduino



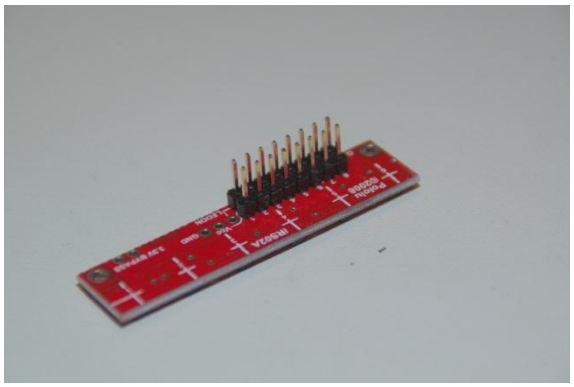
Nun werden die beiden Kabel an den Motor gelötet. Zusätzlich werden über die Lötstellen sogenannte Schrumpfschläuche gesteckt.



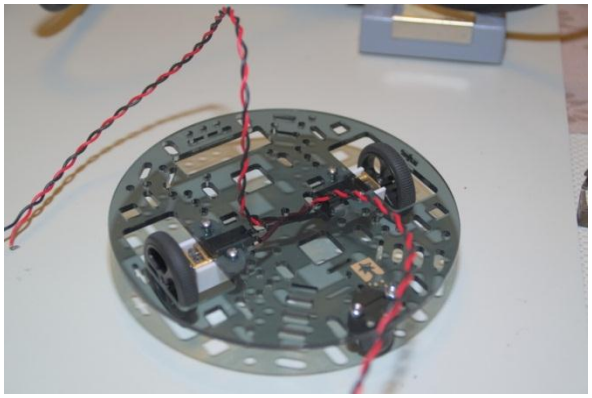
Diese werden dann mit einer Heißluftpistole erwärmt und ziehen sich zusammen. Jetzt können die beiden Kabel verdrillt werden.



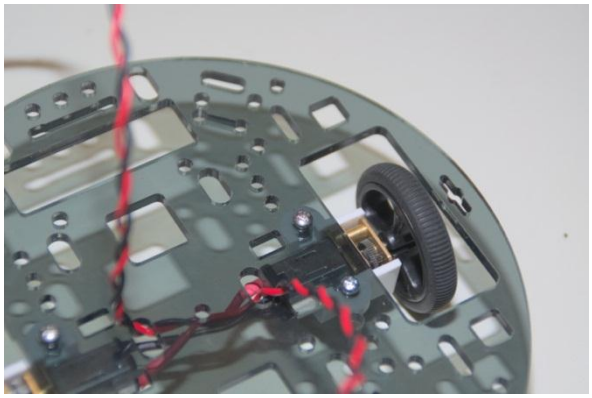
Da wir den IR Sensor nicht mit allen 8 LEDs verwenden kann dieser an einer Sollbruchstellen zwischen LED 6 und LED 7 sehr vorsichtig (mit einer feinen Platinensäge) geteilt werden.



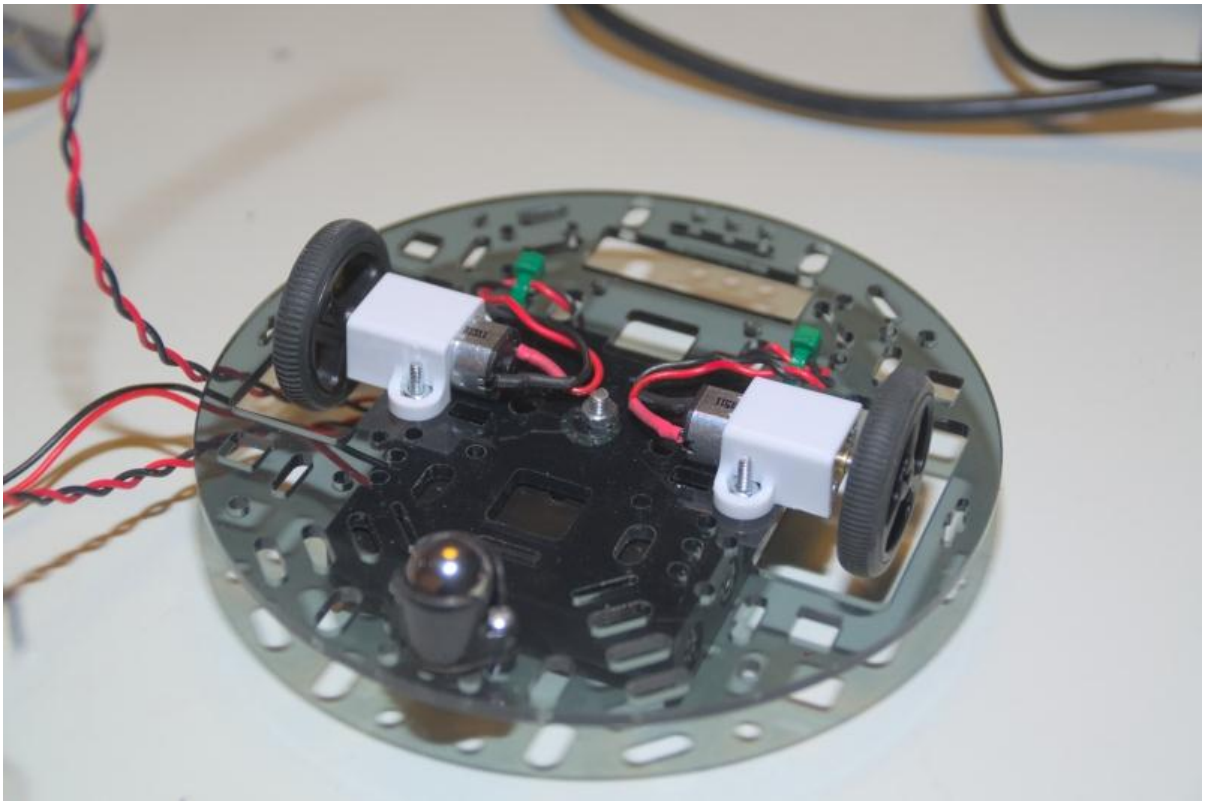
Der bereits fertig mit Stiftleisten bestückte Sensor mit nur mehr 6 LEDs



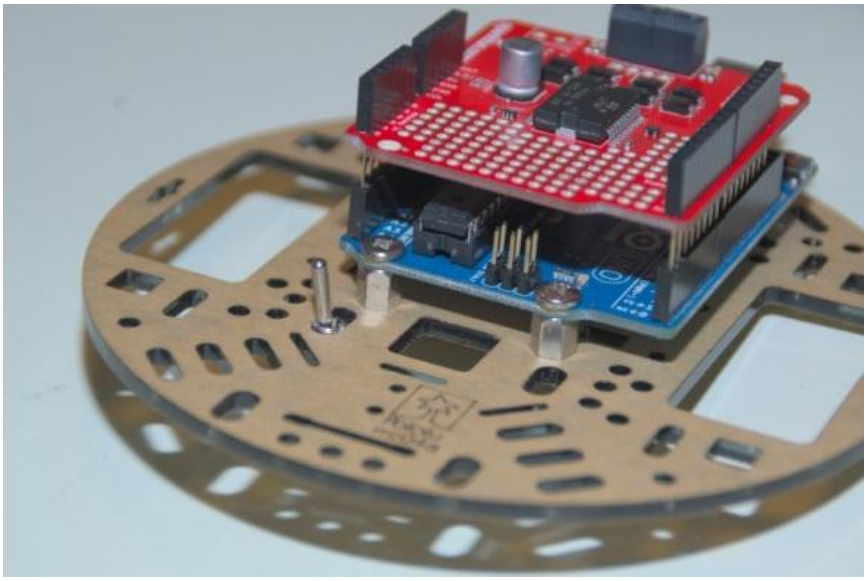
Hier sieht man die Grundplatte mit fertig montierten Motoren und am Heck montiertem „Ball Caster“



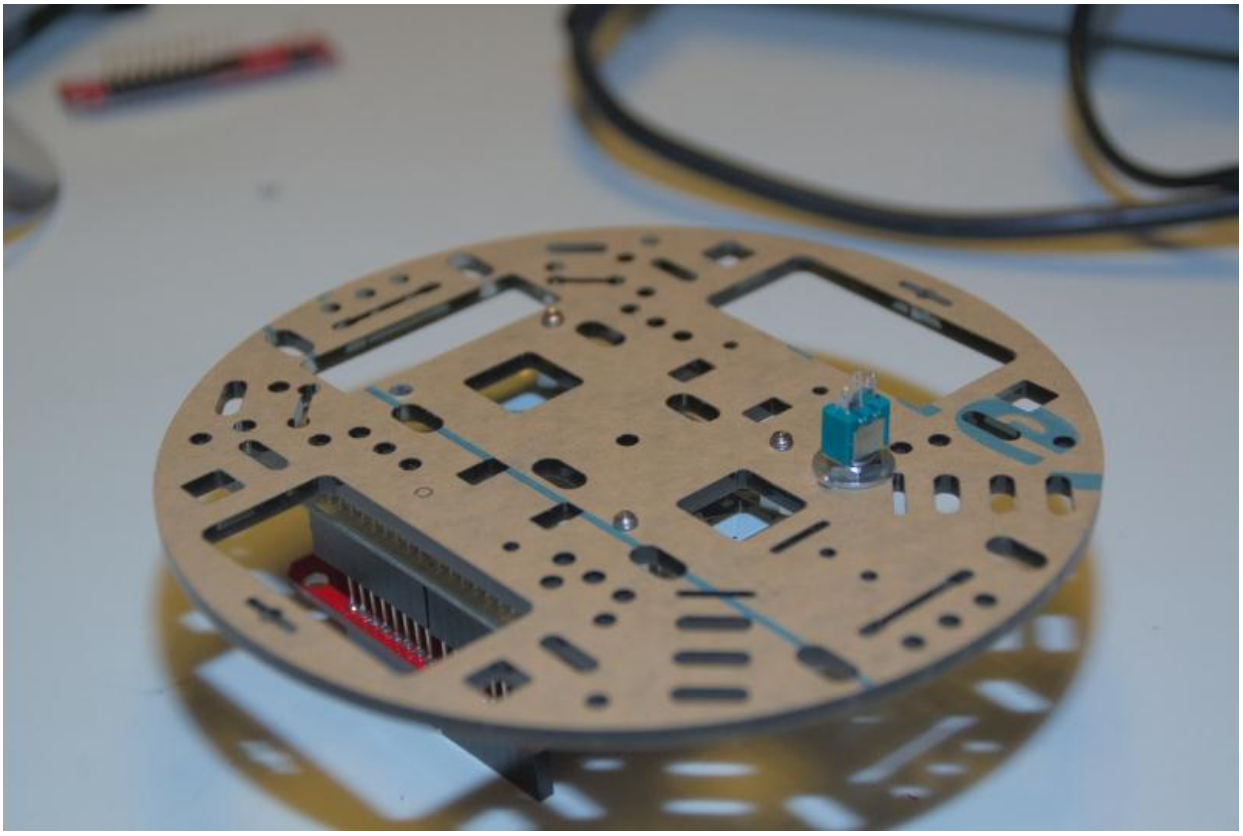
Der montierte Motor im Detail.



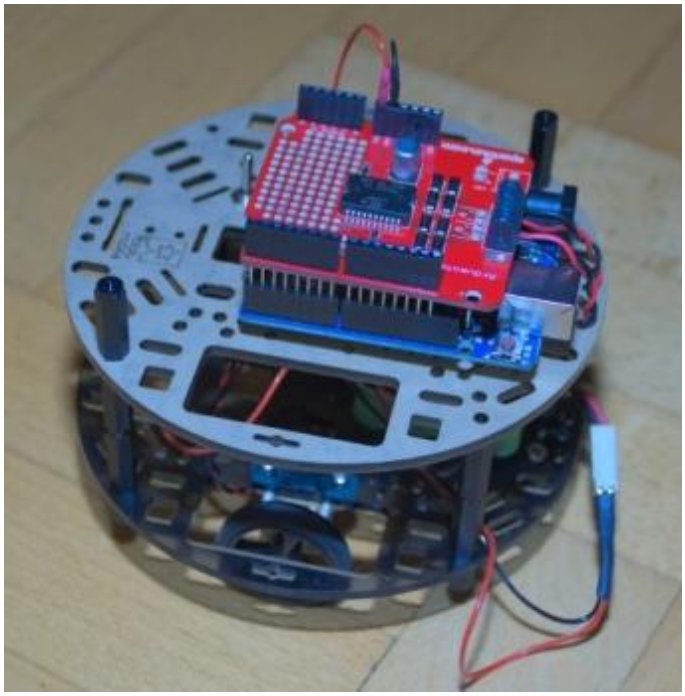
Hier nochmals aus Sicht von unten. Der große schwarze Bauteil der in der Mitte der Grundplatte zu sehen ist, ist ein Batteriefach für 4 Alkaline Batterien 1,5 V (ges. 6V). Dies wurde nur Temporär eingebaut da wir zu diesem Zeitpunkt kein Akkupack zur Hand hatten. Das Fach kann einfach durch entfernen der mittleren Schraube abgenommen werden.



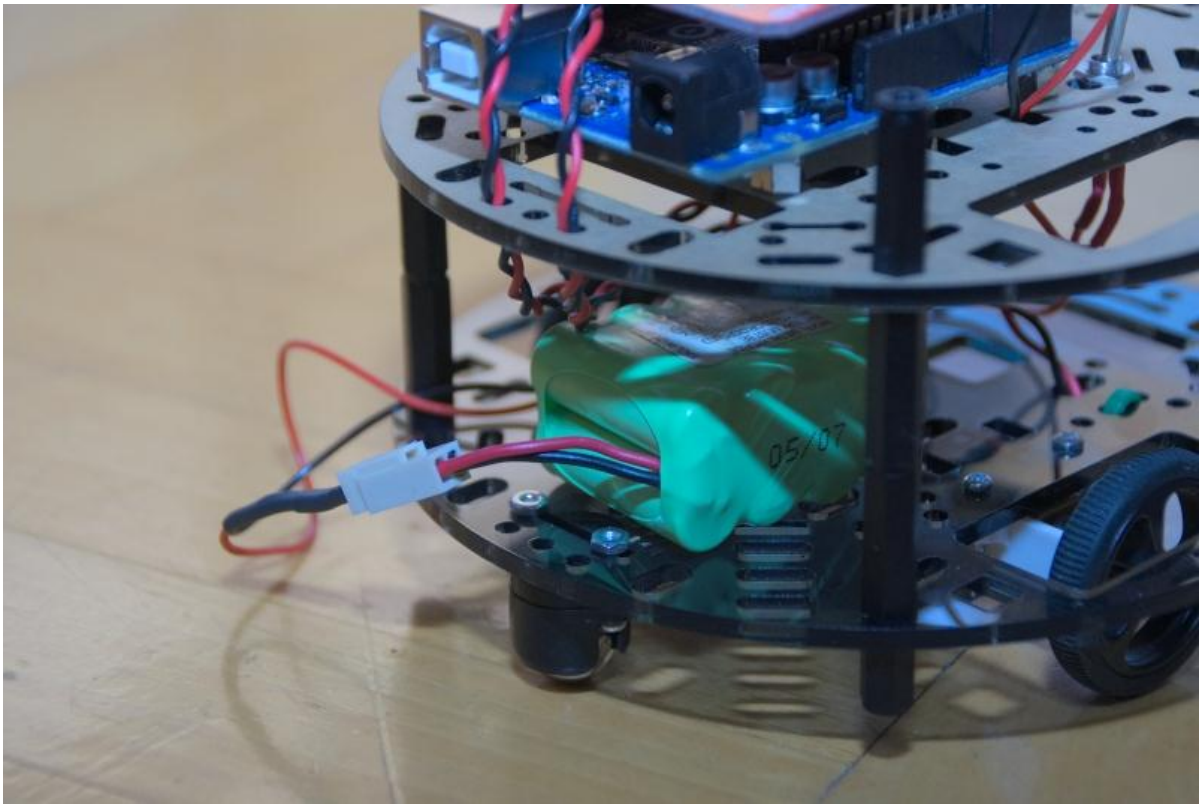
Nun wenden wir uns dem Herzstück des Roboters zu. Das Motortreibermodul wird ganz einfach, nach dem es mit den notwendigen Stiftleisten bestückt wurde auf den Arduino gesteckt werden. Die Distanzbolzen (aus Metall) wurden vorsichtig in die Grundplatte eingeschraubt. Links ist auch noch der Stromschalter zu sehen der auch durch die Platte geschraubt wurde.



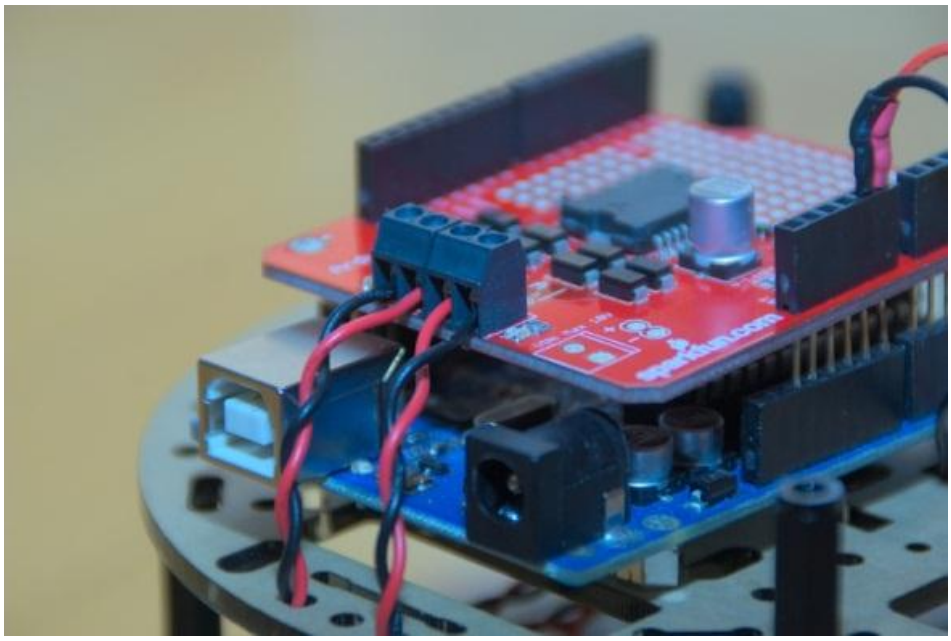
Die ober Grundplatte mit montiertem Arduino und Schalter von unten. Der Schalter wird zwischen Akku und Arduino „eingehängt“. Die später angelöteten Kabel werden ebenfalls mit den Schrumpfschläuchen isoliert.



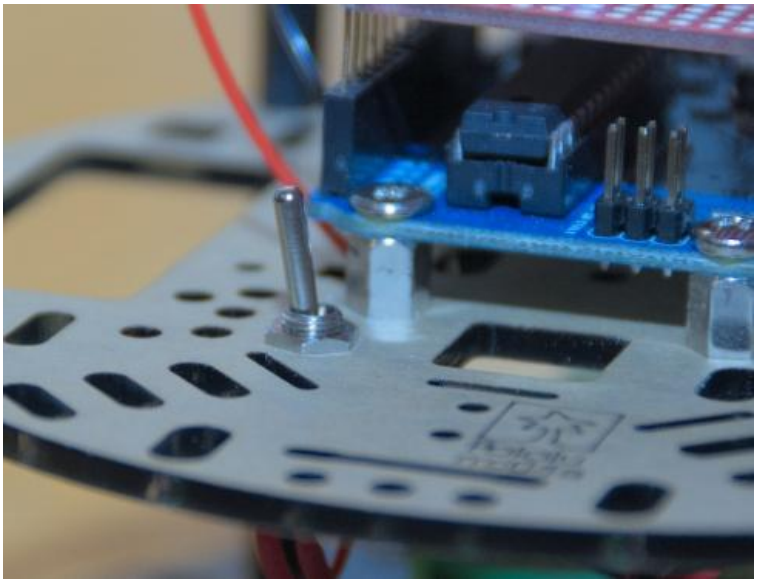
Nach dem der Schalter angeschlossen wurde können wir die beiden Ebenen mit Hilfe der Distanzbolzen (Plastik oder Carbon) testweise zusammenbauen.



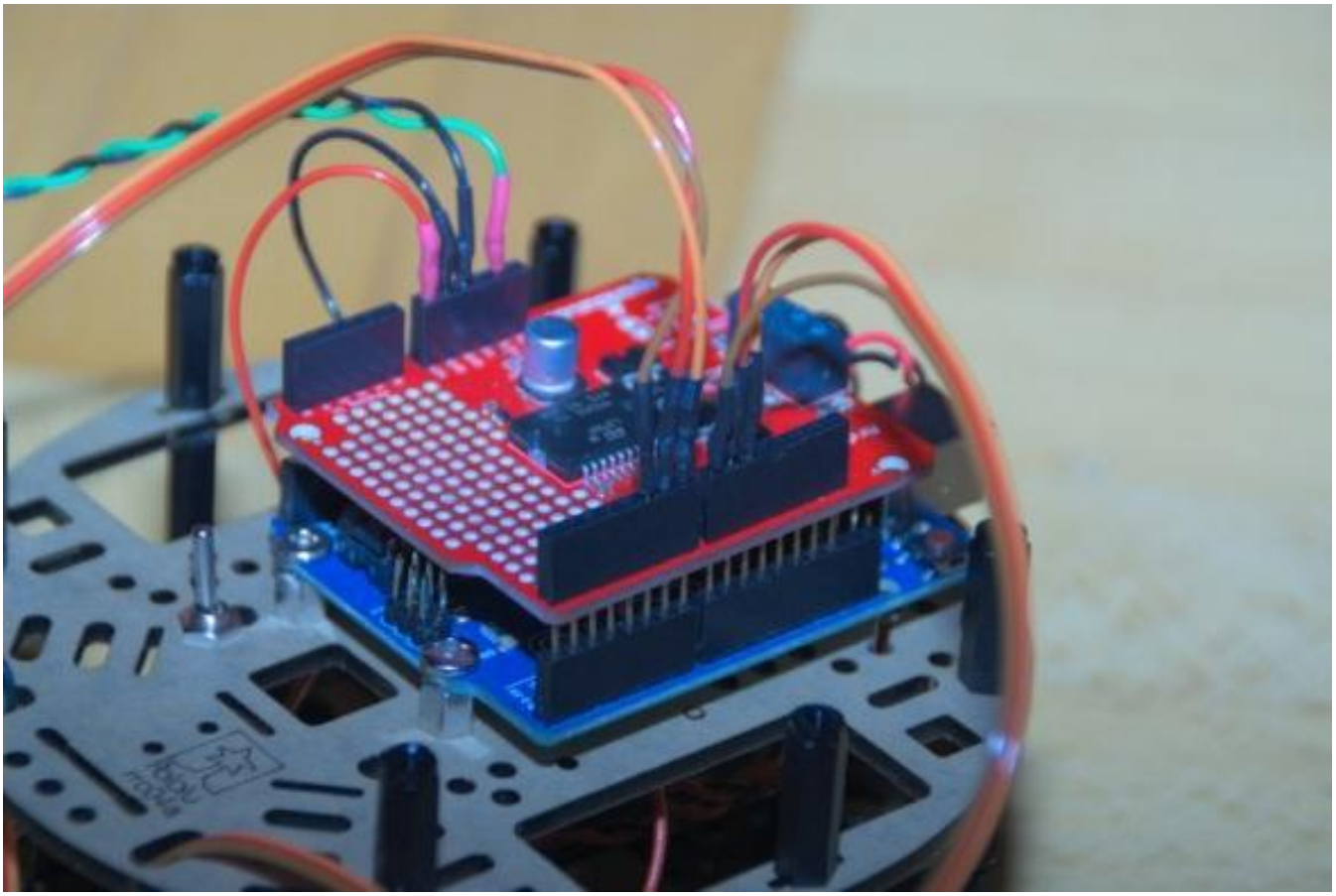
Hier zu sehen sind die links und rechts angebrachten Dintanzbolzen, sowie das bereits vorhin angesprochene und nun ausgetauschte Akkupack mit Systemstecker. Dieses sollte möglichst weit hinten und mittig montiert (Kabelbinder) werden, um ein kippen nach vorne zu vermeiden.



Da wir eine externe Spannungsquelle verwenden, jedoch ein Hohlstecker (rechts vorne) etwas groß und unpraktisch wirkt, verwenden wir den externen Zugang über die Stiftleisten. Das Motortreibermodul verwendet in diesem Fall dieselbe Quelle und benötigt keine weiteren Kabel zum Akku.

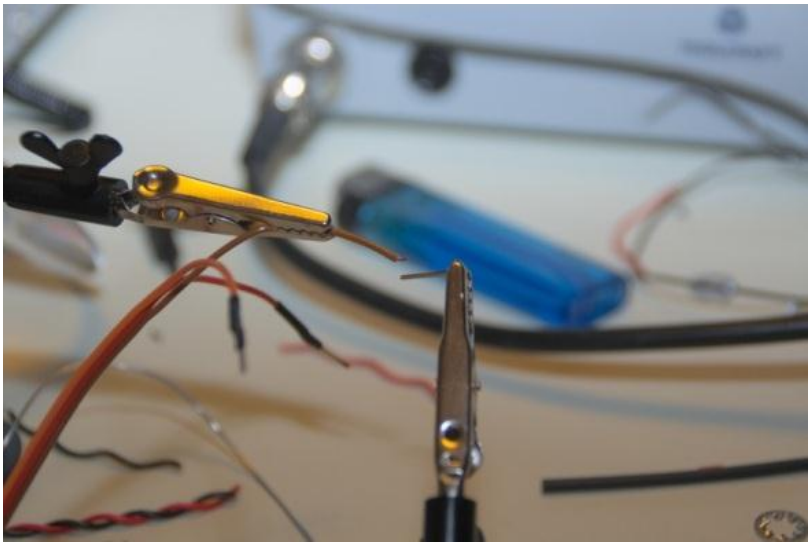


Hier noch eine Detailansicht des zuvor genannten Schalters zwischen Akku und Arduino. Der Kippschalter wurde noch zusätzlich mit einer Mutter gegen Verdrehen gesichert.

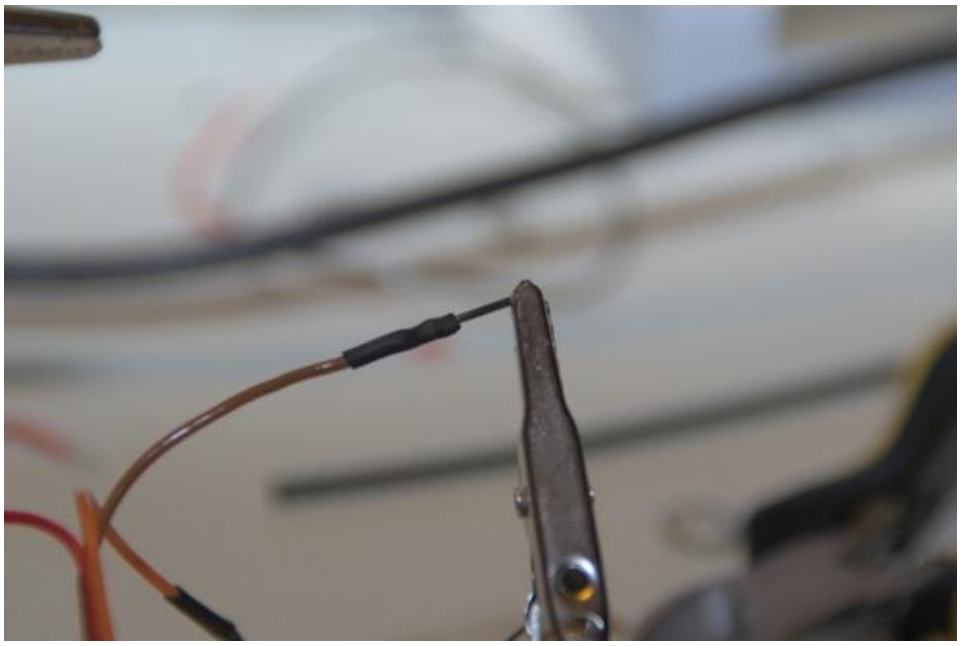


Um nun die Sensorleiste mit dem Arduino, wie oben gezeigt, verbinden zu können müssen an den Kabeln noch Pins angebracht und isoliert werden.

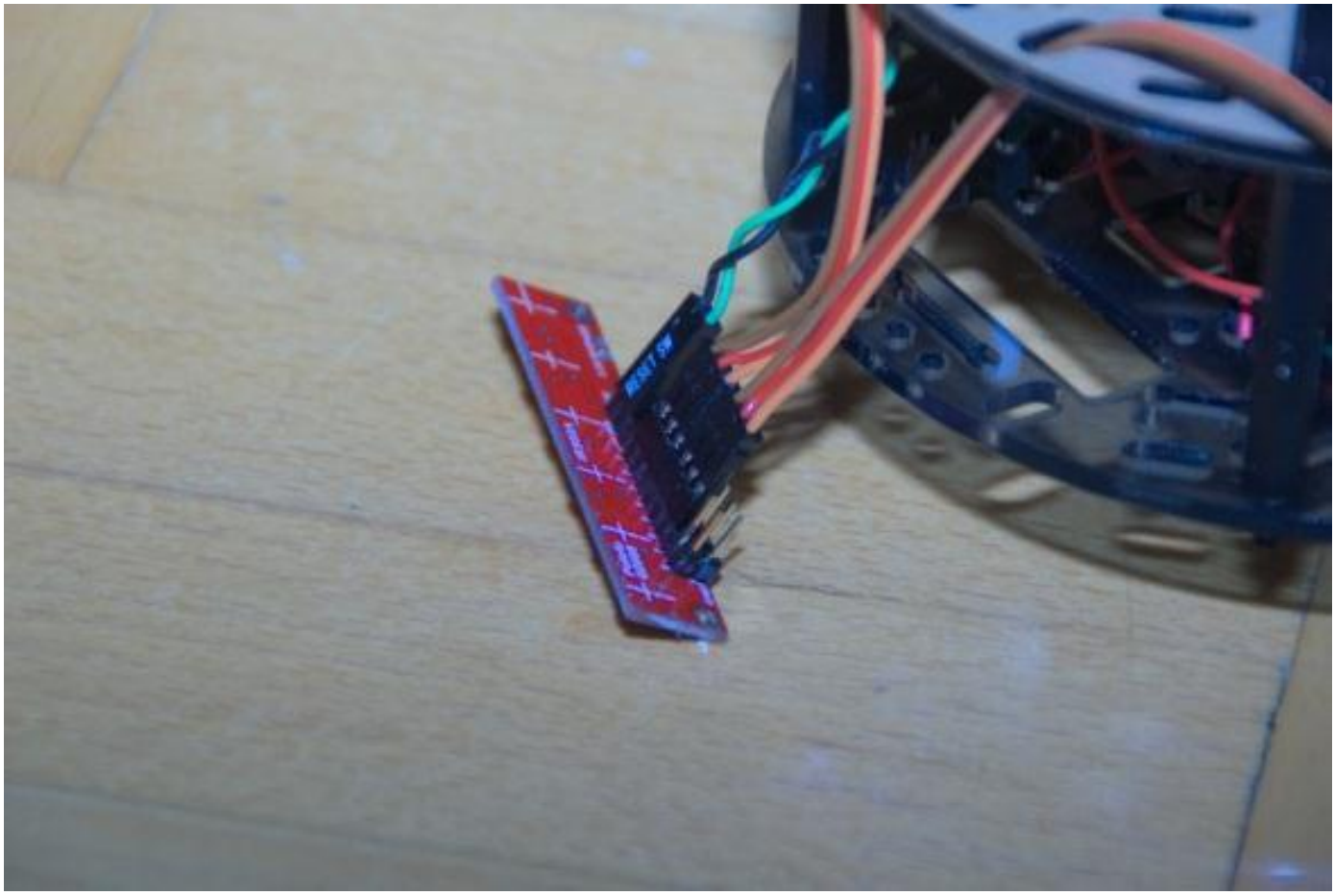
Für die Pins werden die einzelnen Verbinder einer Stiftleiste heraus gezogen.



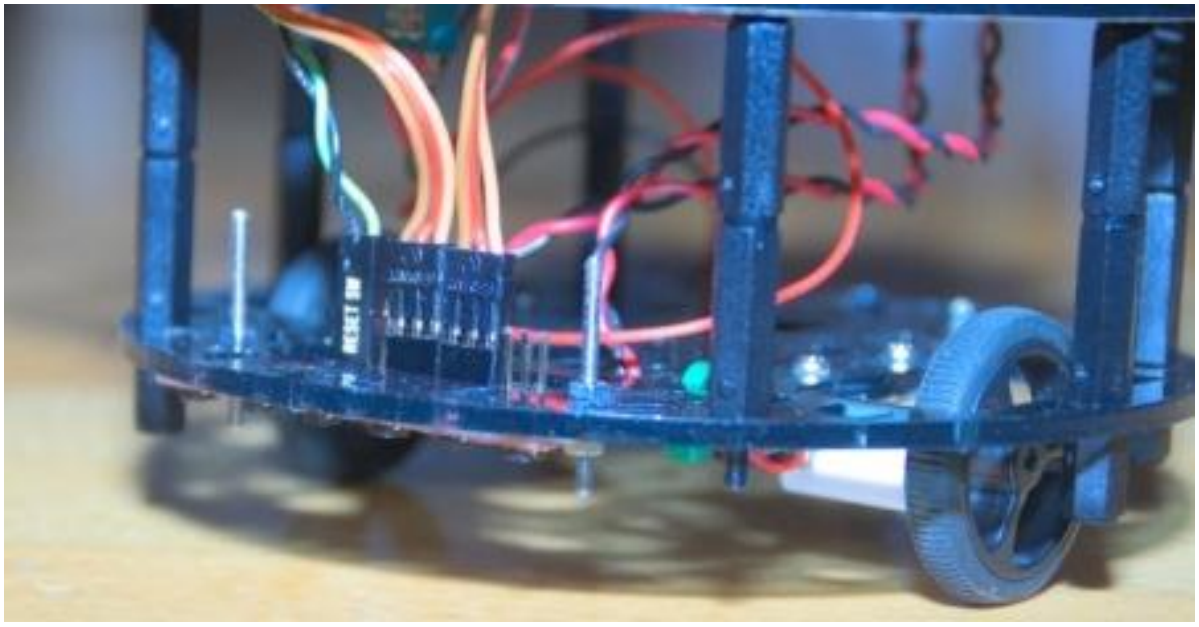
Das Kabel wird nur max. 1mm abisoliert und mit Hilfe einer Klammer an den Pin angelegt.



Um beim späteren Betrieb Kurzschlüsse oder dergleichen zu vermeiden, wird auch hier wieder die Lötstelle mit einem Schrumpfschlauch isoliert. Natürlich nur im hinteren Drittel.



Für den Anschluss der Sensorleiste werden fertige Systemkabel (Servokabel) verwendet.



Nun kann die Sensorleiste verkabelt und eingebaut werden.

Technische – Schaltungsdetails

Pinbelegung für *Sensorleiste:*

Bezeichnung	Pin auf Arduino
Schwarz	GND
Grün	+5V
Braun(*Rechte Seite*)	5
Rot(*Rechte Seite*)	6
Gelb(*Rechte Seite*)	7
Braun(*Linke Seite*)	8
Rot(*Linke Seite*)	9
Gelb(*Linke Seite*)	10
Status LED	2
ProgramCable	4

Pinbelegung für *Stromversorung*

Bezeichnung	Pin auf Arduino
Schwarz	GND
Rot	VIN 9V

Motor

Spannung: DC 2 V - 6 V
Strom min. (Leerlauf): ca. 32 mA
Strom (RUNNING): 80 mA – 100 mA
Strom max. (STALL): 500 mA
Übersetzung: 30:1
Getriebe: Metall

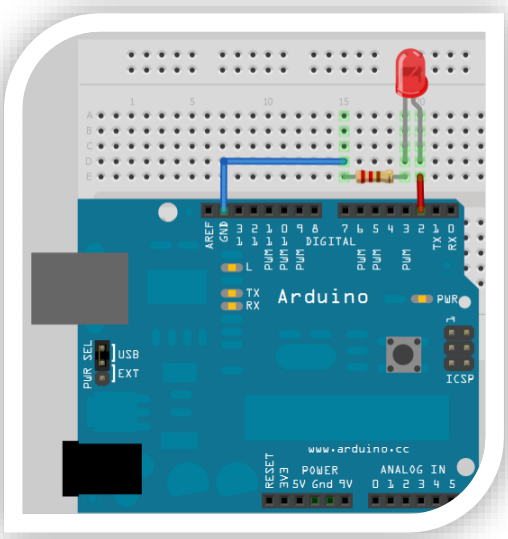
Akku

Typ: Lithium Ionen
Strom: 500 mA
Spannung: 9 Volt

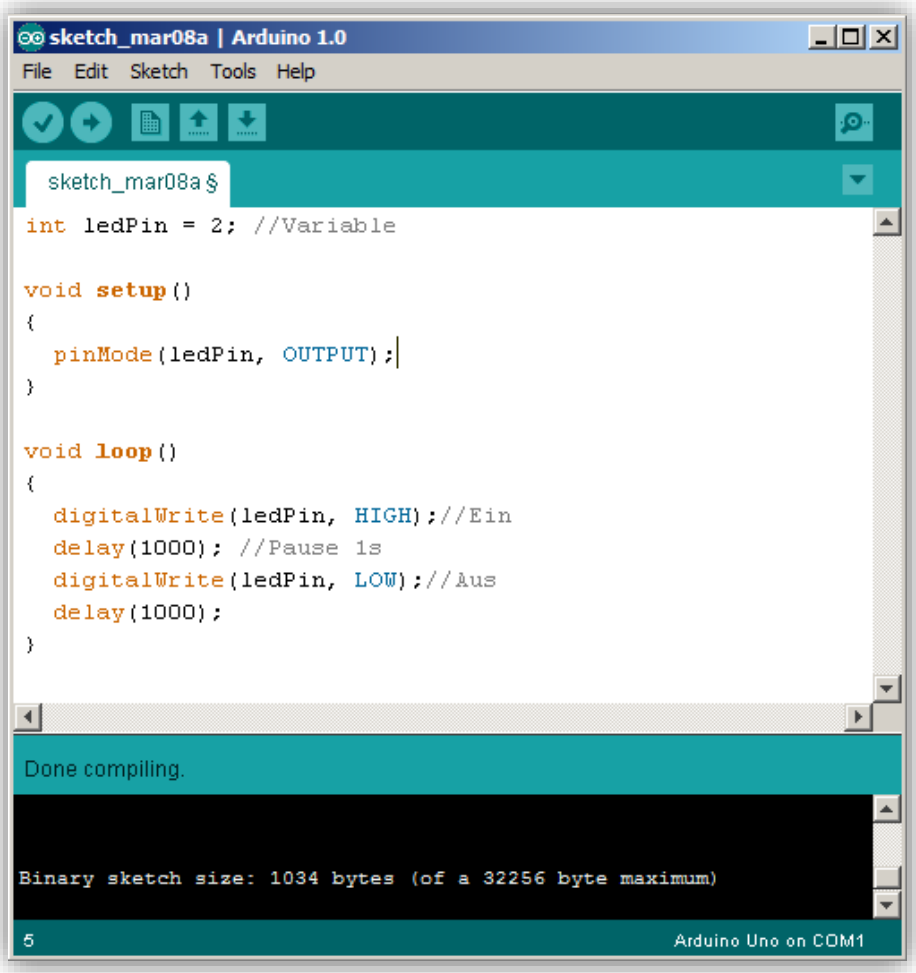
Grundlagen zur Programmierung

Die gesamte Referenz zu den Arduino spezifischen Befehlen finden Sie in Englischer Sprache unter folgender Internetadresse: http://arduino.cc/it/Reference/HomePage

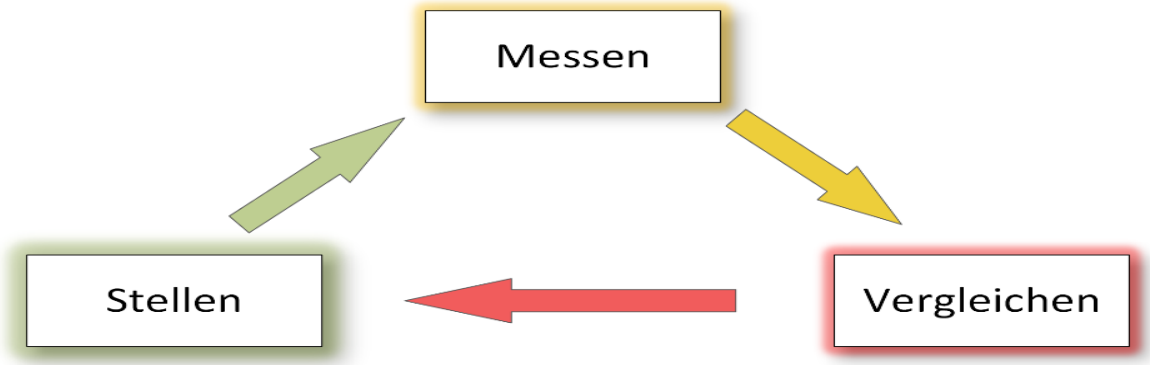
Es folgt ein kleines Beispiel, bei dem gezeigt wird, wie mit einem digitalen Port eine LED ein und wieder ausgeschaltet werden kann. Die passiert alle 1000 ms.

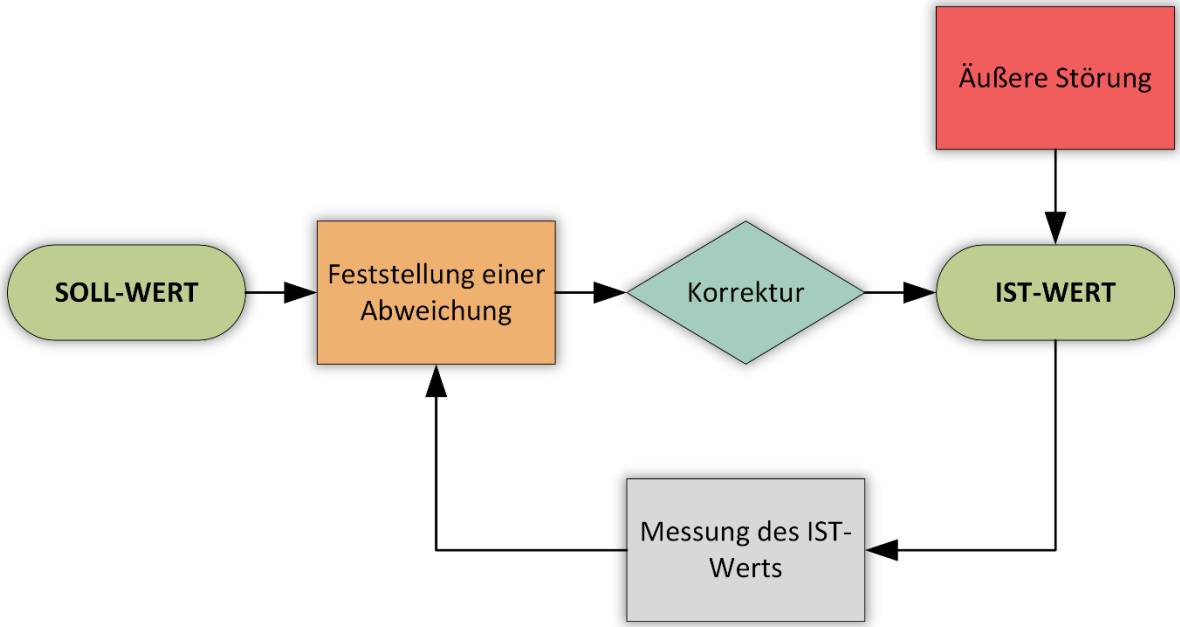


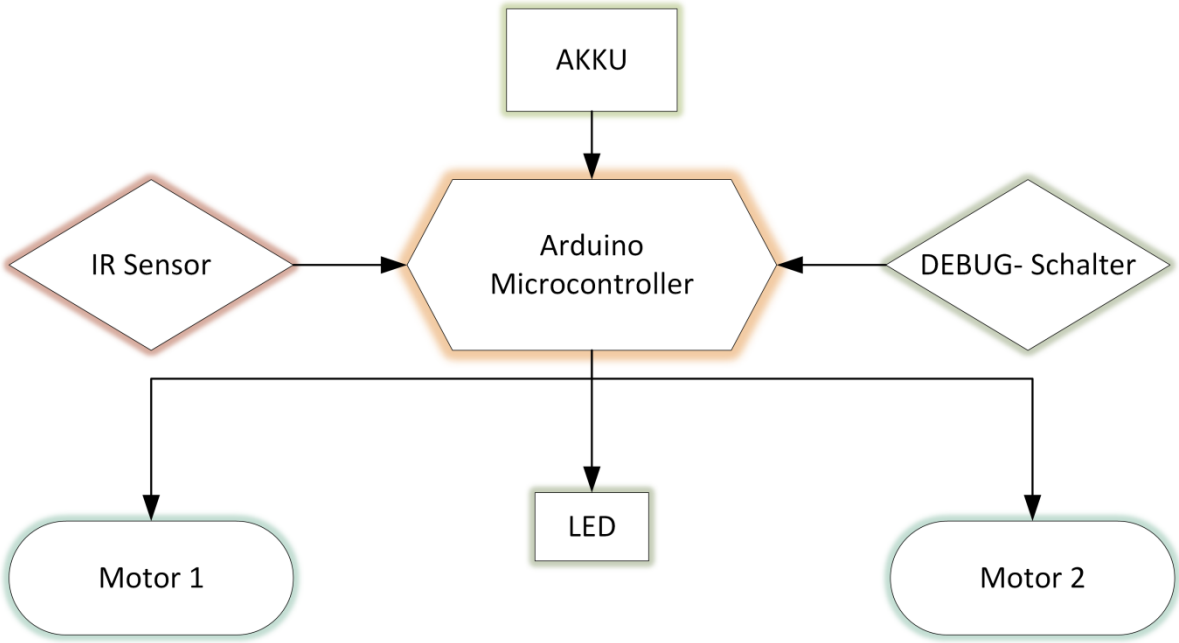

```
int ledPin = 2; //Variable

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  digitalWrite(ledPin, HIGH);//Ein
  delay(1000); //Pause 1s
  digitalWrite(ledPin, LOW);//Aus
  delay(1000);
}
```



Programmübersicht – Diagramm







Die Config.h – Default

```
//PID Control
#define KP .18
#define KD 5

//Engine System
#define MOTOR1_NORMAL_SPEED 50
#define MOTOR2_NORMAL_SPEED 50
#define MOTOR1_MAXIMAL_SPEED 70
#define MOTOR2_MAXIMAL_SPEED 70

//Engine Navigation
#define FWD 0 //Vorwärts
#define REV 1 //Rückwärts

//Engine Connection and PWM
#define PwmPinMotorA 3
#define PwmPinMotorB 11
#define DirectionPinMotorA 12
#define DirectionPinMotorB 13

//IR Sensor
#define NUM_SENSORS 4 // Anzahl an verwendeten Sensoren
#define TIMEOUT 1500
#define EMITTER_PIN 2

//Debug Mode
#define DEBUG 0 //  Debug Modus 0 = Aus, 1 = Ein
#define DEBUG_OUPUT_SPEED 700 // Debug Ausgabe Geschwindigkeit
#define PROGRAMCABLE 4
#define STATUS_LED 2
```

Bezugsquellen

Hardware

Arduino: *http://www.physicalcomputing.at/*
Robotik: *http://www.watterott.com/*
Kleinteile/Zubehör: *http://www.conrad.at*

Software

Arduino IDE: *http://arduino.cc/en/Main/Software*
Arduino Tutorials: *http://arduino.cc/en/Tutorial/HomePage*
Arduino Referenz: *http://arduino.cc/en/Reference/HomePage*

Fritzing: *http://fritzing.org/*
Eagle (Schaltplan): *http://www.cadsoftusa.com/downloads/?lang=de*

SourceCode und Librarys

SourceCode(Ardubot): *https://github.com/neo3000/Ardubot-2.0/zipball/master*

QTR Sensor Library: *https://github.com/pololu/qtr-sensors-arduino/zipball/master*