AREXX Engineering

AREXX ARDUINO ROBOT

AAR

BAUANLEITUNG: AAR-04

© AREXX - DIE NIEDERLANDE V052012

Inhaltsverzeichnis

AREXX und AAR sind registrierte Warenzeichen von AREXX Engineering - HOLLAND.

© Deutsche Übersetzung/German translation (March 2012): AREXX Engineering (NL).
Diese Beschreibung ist urheberrechtlich geschützt. Der Inhalt darf auch nicht teilweise kopiert oder über-nommen werden ohne schriftlicher Zustimmung des europäischen Importeurs:
AREXX Engineering - Zwolle (NL).

Hersteller und Vertreiber sind nicht haftbar oder verantwortlich für die Folgen unsachgemäßer Behandlung, Einbaufehler und oder Bedienung dieses Produkts bei Mißachtung der Bauanleitung. Der Inhalt dieser Gebrauchsanleitung kann ohne vorheriger Ankündigung unsererseits geändert werden.

Fabrikant:
AREXX Engineering
JAMA Oriental

Europäischer Importeur:
AREXX Engineering
ZWOLLE Die Niederlande

Technische Unterstützung beim Bauen des Roboters:

WWW.AREXX.COM
WWW.ROBOTERNETZ.DE

© AREXX Holland und JAMA Taiwan
© Deutsche Übersetzung: AREXX - Die Niederlande

1. PRODUKTBESCHREIBUNG AAR

1.1 Die ARDUINO Roboterfamilie

Arduino ist eine „open source "-Plattform[1] zur Entwicklung von Elektronikprototypen, die uns einen Mikrocontroller einschließlich aller peripheren Schnittstellen und benötigter Software zur Verfügung stellt.

Das Arduino-Konzept wurde entwickelt um auf möglichst einfacher Weise die Handhabung mit der modernen Elektronik zu erlernen, die man in der Roboterwelt, Softwaresteuerung und Sensoren verwendet.

Als Nachfolger des ASURO-Roboters, der in der Programmiersprache C programmiert wird, ist jetzt der AREXX Arduino Roboter entstanden, der seinem Vorgänger ASURO ziemlich stark ähnelt, sich aber aufgrund der „open source"-Programmiersprache Arduino wesentlich einfacher programmieren lässt.

[1] Open Source und quelloffen ist eine Palette von Lizenzen für Software, deren Quell text öffentlich zugänglich ist und durch die Lizenz Weiterentwicklungen fördert.

1.2. Spezifikationen:

Motoren	2 Gleichstrommotoren (3 Volt)
Prozessortyp	ATmega328P
Programmiersprache	ARDUINO
Spannung	4 St. AAA Akku oder Batterie 4,8 - 6 Volt Max.
Stromverbrauch	Min. 10 mA Max. 600 mA
Kommunikation	USB-Stecker
Erweiterungen	die ASURO-Erweiterungen sind einsetzbar
Höhe	40 mm
Breite	120 mm
Tiefe	180 mm

1.3 Warnungen

1. Sie müssen dieses Handbuch gelesen haben ehe Sie auf einem der Anschlüssen eine Stromquelle anschließen! Fehlerhafte Anschlüsse können die Hardware schädigen.
2. Überprüfen Sie bitte sorgfältig die Anschlussbelegung! Arbeiten Sie bei der Verdrahtung des Systems sehr sorgfältig. Falsche Anschlüsse können Komponenten beschädigen. Beachten Sie die korrekte Polarität der Stromversorgungsanschlüsse. Eine Umpolung der Stromversorgungsanschlüsse kann die Hardware beschädigen.
3. Verwenden Sie keine Stromversorgungssysteme mit Spannungen, die über den spezifizierten Werten liegen! Benutzen Sie stabilisierte und gefilterte Stromversorgungssysteme um Spannungsspitzen zu vermeiden.
4. Die Leiterplatte bietet keinerlei Schutz gegen Wasser- oder Feuchtigkeitseinwirkung. Zur Aufbewahrung und Benutzung des Systems sind nur trockene Räume geeignet.
5. Vermeiden Sie Kurzschlüsse mit jeglichen Metallgegenständen und vermeiden Sie jegliche Überbelastung der Leiterplatte oder Anschlüssen durch Ziehen, Drücken oder Gewichten.
6. Vermeiden Sie elektrostatische Entladungen[2] (siehe dazu die Vorsichtsmaßnahmen, Warnungen und Dokumentation in Wikipedias Eintrag „Elektrostatische Entladung").

1.4 Algemein

* Mit dem Öffnen der Plastikbeutel mit Komponenten und Teilen erlischt das Rückgaberecht.
* Lese vor dem Bauen zuerst die Gebrauchsanleitung aufmerksam durch.
* Sei vorsichtig beim Hantieren der Werkzeuge.
* Baue nicht im Beisein kleiner Kinder. Die Kinder können sich verletzen an den Werkzeugen oder kleine Komponenten und Teile in den Mund stecken.
* Achte auf die Polung der Batterien.
* Sorge dafür, daß die Batterien und die Batteriehalter trocken bleiben. Falls der Roboter naß wird, entferne dann die Batterien und trockne alle Teile, so gut es geht.
* Entferne die Batterien, wenn der Roboter mehr als eine Woche ruht.

[2] *Engl. electrostatic discharge, kurz ESD*

2. ARDUINO ALLGEMEINE INFO

2.1. Wer oder was ist ARDUINO?

Arduino ist ein open source-Einplatinen Mikrocontroller, der insbesondere Künstlern, Designern, Bastlern und anderen Interessierten den Zugang zur Programmierung und zu Mikrocontrollern und die Projektarbeit an interaktiven Objekten erleichtern soll.

Die Arduino-Plattform basiert auf einer ATmega168 oder ATmega-328 Mikrocontroller von Atmel. Das System stellt dem Anwender sowohl digitale Ein- und Ausgänge als analoge Eingänge zur Verfügung. Damit kann das Arduino-System aus der Umgebung Signale empfangen und anschließend darauf reagieren.

Es werden verschiedene Arduino-Platinen auf dem markt angeboten, wie zum Beispiel Arduino Uno, Arduino LilyPad und Arduino Mega 2560. Da jede Arduino-Platine individuell seine spezifischen Eigenschaften aufweist kann man für wohl jedes Projekt die ideale Arduino-Baugruppe auswählen.

Eingangssignale können zum Beispiel durch Schaltern, Lichtsensoren, Bewegungssensoren, Abstandssensoren und Temperatursensoren geliefert werden. Auch kann man Kommandos aus dem Internetbereich als Eingangssignale zuliefern. Ausgangssignale wiederum können Motoren, Lämpchen, Pumpen und Bildschirme ansteuern. Zur Programmierung verfügt das System über einem Compiler für eine standardisierte Programmiersprache und einem Bootloader. Die Programmiersprache basiert auf die Wiring-Programmiersprache, die mit C++ übereinstimmt.

Arduino wurde zunächst in 2005 als Projekt gestartet in Ivrea, Italien. Erklärtes Ziel war ursprünglich die Idee Studenten in Projektarbeiten zu unterstützen, wobei die Prototyp-Erstellung deutlich preiswerter sein sollte als bei vergleichbaren herkömmlichen Methoden.

Die Entwicklergruppe um Massimo Banzi und David Cuartielles benannten das Projekt nach einer historischen Gestalt mit dem namen ‘Arduin von Ivrea’. Das Wort ‘Arduino’ bedeutet ‘Kräftiger Freund’.

2.2 Mikrocontroller!

2.2.1 Anwendungen

Ein Mikrocontroller (manchmal in der verkürzten Schreibweise auch µC, uC oder MCU genannt) ist ein kleiner Computer in einer integrierten Einzelschaltung, der den Prozessorkern, Speicher und und einen Satz programmierbarer Ein-/Ausgangsanschlüsse enthält.

Programmspeicher und ein kleiner Datenspeicher, RAM (beziehungsweise Random Access Memory) gehören oft ebenfalls zur Ausstattung des Chips. Mikrocontroller werden in automatisch gesteuerten Anlagen und Systemen eingesetzt, wie zum Beispiel in der Motorensteuerung, Implantaten, Fernsteuerungen, Bürosystemen, Spielzeug und Hochleistungswerkzeugen.

Die Gewichtseinsparung und Kosteneinsparung durch Integration des Mikroprozessors, Speicher und Ein-/Ausgangsanschlüsse auf einem Einzelchip führt dazu dass die Digitalsteuerung für immer mehr Anwendungsbereiche noch wirtschaftlicher wird.

Ein typischer Haushalt in einer fortschrittlichen Wohngegend verfügt über vier allgemeinen Mikroprozessoren und drei Dutzend Mikrocontroller Schaltungen. Ein Durchschnittstyp eines Mittelklassen-PKW verwendet 30 oder mehr Mikrocontroller. Auch findet man diese Bauteile in vielen elektrischen Maschinen wie Waschmaschinen, Mikrowellenherden und Telefonen.

2.3. Leistungsverbrauch und Geschwindigkeit

Manche Mikrocontrollers arbeiten bei einer niedrigen Taktfrequenz von 4 kHz und weisen einen geringen Leistungsverbrauch im Milliwatt- oder Mikrowatt-Bereich auf. Sie können üblicherweise sofort auf einem Knopfdruck oder Interrupt-Prozess aktiviert werden. Der Leistungsverbrauch liegt beim Warten (bei abgeschaltetem CPU-Clockgenerator sowie fast der kompletten Zusatzbeschaltung) im Nanowattbereich, was eine Langlebigkeit der Batterien sicherstellt. Andere Mikrocontroller werden eher im Hochleistungsbereich verwendet, wo man sie zum Beispiel als Digitale Signalprozessoren (DSP) mit höheren Taktraten und höherem Leistungsverbrauch einsetzt.

Das Arduino-System arbeitet mit einem leistungsfähigen Atmel ATmega328P Single-Chip, der mit einem 8-bit Mikrocontroller (getaktet mit 16MHz) und 32K Bytes In-System programmierbarem Flash-Speicher ausgestattet ist. Die Stromversorgung ist recht flexibel im Bereich von DC7-12V, gewählt worden, wodurch man auf stabilen und ordentlich abgesicherten Arbeitsbedingungen für den Chip und abgetrennten Leistungsleitungen bis 2A für die Motorenversorgung aufbauen kann.

2.4 Mikrocontroller Programme

Die Software für Mikrocontroller muss im auf dem Chip verfügbaren Speicherplatz passen, denn ein externer Zusatzspeicher wäre zu kostspielig. Die Compiler und Assembler sind optimiert für die Umsetzung der Hochsprache und Assemblercodes in kompakte Maschinenbefehle, die in den Speicher des Mikrocontrollers abgelegt werden.

Je nach Chiptyp kann das Programm in einem permanenten ROM[3] -Speicher, der nur während der Fertigung des Chips beschrieben werden kann, oder in einem immer wieder beschreibbaren Flash – oder mehrfach beschreibbarem ROM-Speicher gespeichert werden. Ursprünglich hat man Mikrocontroller nur in Assemblersprache programmiert, aber zur Zeit sind auch verschiedene höheren Programmiersprachen für die Programmierung der Mikrocontroller verfügbar. Diese Sprachen sind entweder Spezialsprachen oder Varianten der allgemeinen Hochsprachen wie zum Beispiel C. Verkäufer der Mikrocontroller bieten oft gratis Werkzeuge an um die Implementierung der Hardware zu vereinfachen. Das Arduino-System stellt uns etwa 32 kBytes Flash-Speicher für die sogenannten Sketch-Programme zur Verfügung, die in C programmiert werden können.

2.5 Schnittstellenarchitektur

Mikrocontroller verfügen in der Regel über verschiedenen bis Dutzenden von frei programmierbaren Ein-/Ausgangspins (GPIO). Die GPIO-Anschlüsse sind mittels Softwarekommandos programmierbar als Eingangs- beziehungsweise Ausgangspins. Bei Programmierung als Eingangspins werden sie oft zum Ablesen der Sensoren oder externen Signalen verwendet. Programmiert man sie als Ausgangspins kann man sie zum Beispiel zur LED- oder Motoransteuerung verwenden.
Eine Vielzahl an eingebetteten Systemen benötigt die Ablesemöglichkeit der Analogsignalen von den Sensoren. Zu diesem Zweck werden Analog/Digitalwandler (A/D-Wandler oder in Englisch ADC[4]) eingesetzt.

Da die Prozessoren speziell zur Verarbeitung von Digitaldaten, das heißt Nullen und Einsen, entwickelt worden sind, können sie mit den Analogsignalen der Sensoren an sich nichts anfangen. Deshalb werden die A/D-Wandler eingesetzt um die eintreffenden Daten in eine für den Prozessor lesbaren Form zu verwandeln.

[3] ROM = Read Only Memory (nur lesbarer Speicher)
[4] Analog-to-digital converter (ADC)

Ein weniger üblichen Form der Datenverwandlung in Mikrocontroller ist die Digital zu Analog-Umwandlung (im D/A-Wandler oder DAC[5]) womit der Prozessor Analogsignale oder Spannungspegel erzeugen kann.
Zusätzlich zu den Konvertern verfügen einige eingebettete Systeme auch noch über Zeituhr-Schaltungen (Timers). Eins der gängigen Zeituhrtypen ist die Programmierbare Intervall-Zeituhr (PIT[6]) , der von einem beliebigen Anfangswert herabzählt nach Null. Nachdem es die Null erreicht hat sendet die Uhr einen Interrupt an den Prozessor als Signal, dass die eingestellte Zeit abgelaufen ist. Das ist eine nützliche Funktion für zum Beispiel Thermostaten, die regelmäßig die Umgebungstemperatur registrieren und prüfen sollen, um – falls notwendig – die Klimaanlage beziehungsweise Heizung ein zu schalten.

Das universelle asynchrone UART[7]-Transceivermodul erlaubt uns ohne allzu viel Prozessoraufwand Kommunikationsdaten über einer seriellen Leitung mit anderen Systemen aus zu tauschen (das heißt zu versenden und zu empfangen).

Spezielle integrierte Hardware im Chip ermöglicht oft auch die Kommunikation mit anderen Chips in Digitalformaten wie I2C und Serial Peripheral Interface (SPI).

Das Arduinosystem stellt uns 14 digitale I/O-Anschlüsse und 7 analoge I/O-Anschlüsse zur Verfügung (I/O- das heisst: Eingang/Ausgangs-).

[5] *Digital-to-analog converter (DAC)*
[6] *Programmable Interval Timer (PIT)*
[7] *Universal Asynchronous Receiver/Transmitter (UART)*

3. AREXX ARDUINO ROBOT

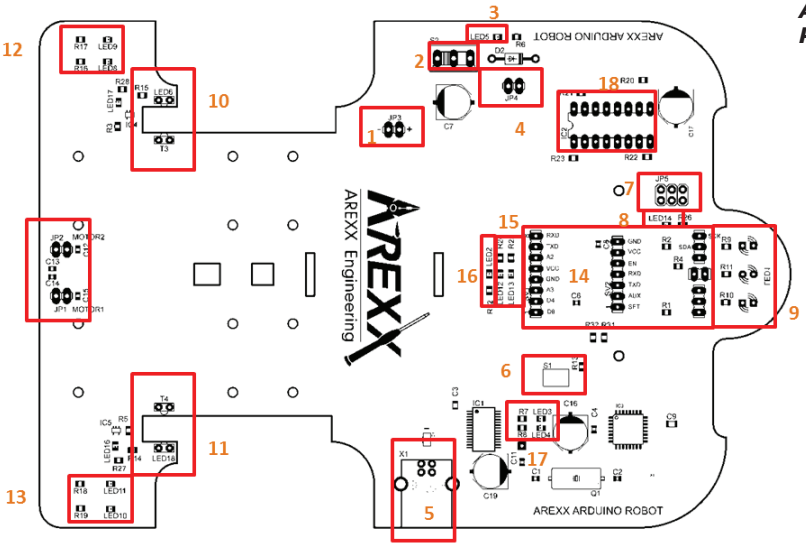


Abb. 1: AAR Platine

3.1 Blockschaltbild des ARDUINO ROBOTERS

1. Anschluss für den Batteriehalter. (Beachten Sie bitte die Polarität)
2. Ein/Aus-Schalter für den Roboter.
3. Status-LED: Anzeige, dass der Roboter von der Stromversorgung gespeist wird.
4. Falls Sie aufladbare Akkuzellen verwenden können Sie dieses Pin-Paar überbrücken, so dass der Roboter die korrekte Arbeitsspannung erhält (Achtung: die Polarität-Überprüfung mittels Diode wird dabei abgeschaltet).
5. USB-Anschluss zum Programmieren des Roboters mittels Arduino-Software.
6. Reset-Taste: für den manuellen Reset des Roboters.
7. ISP- Steckverbinder: mit diesem Steckverbinder kann man ggf. einen anderen Bootloader einlesen.
8. LED 14: dieser LED ist frei programmierbar und blinkt falls der Bootloader neu gestartet wird.
9. Linien-Folger: Dieses Modul ist frei programmierbar, so dass der Roboter eine Linie verfolgen kann.
10. Radsensor links: dieser liefert eine Impulsfolge beim Drehen des linken Rads.
11. Radsensor rechts: dieser liefert eine Impulsfolge beim Drehen des rechten Rads.
12. Status LEDs für den Motor auf der linken Seite: Diese LEDs zeigen an ob dieser Motor vorwärts oder rückwärts antreibt.
13. Status LEDs für den Motor auf der rechten Seite: Diese LEDs zeigen an ob dieser Motor vorwärts oder rückwärts antreibt.
14. Steckverbinder für die Erweiterungsplatine: an diesem Stechverbinder kann man zum Beispiel ein APC220 Funkmodul oder Snake Vision am Arduino-System angeschlossen werden.
15. Status LEDs für die RS232 Data-Kommunikation.
16. Status LED 2: Frei programmierbarer LED.
17. Status LEDs für die USB Data-Kommunikation.
18. Motorcontroller

3.2 Hintergrundinformation zum AAR

Der AAR ist ein Arduino-Roboter, der speziell entworfen wurde zum Erlernen der Arduino Software. An der Vorderseite befindet sich die USB-Schnittstelle mit dem FT232 IC, welches das USB-Signal in ein RS232 UART-Signal verwandelt, das vom ATMEGA328P Prozessor (rechts vorne) verarbeitet werden kann.

Auf der gegenüberliegenden Seite wurde der EIN/AUS-Schalter mit dem JP3-Steckerplatz für den Stromversorgungsanschluss und den Motortreiber IC2 montiert. An der Rückseite der Leiterplatte befinden sich die beiden Motoren und Radsensoren.

Die Radsensoren enthalten eine Lichtschleuse. Die Zahnräder sind mit vier Löchern ausgestattet, die in einem 90°-Muster angeordnet wurden. Sobald das Licht des LEDs durch die Löcher den Sensor erreicht, meldet der Radsensor dem Prozessor für das betreffende Rad einen Impuls. Dabei wird LED16 beziehungsweise LED17 zur optischen Anzeige aktiviert. Auf dieser Weise kann man ziemlich genau die Drehgeschwindigkeit der einzelnen Hinterräder messen.

An der Vorderseite befinden sich die Steckplätze für die Erweiterungsplatinen und an der Unterseite der Leiterplatte die Sensoren für den Linienfolger. Der Linienfolger enthält eine LED, der an zentraler Stelle den Boden ausleuchtet. Daneben wurden zwei Infrarotsensoren angeordnet, die das reflektierte Licht vom Boden registrieren. Außerdem enthält die Leiterplatte noch die benötigten Bauteile (LEDs, Widerstände und Kondensatoren) um den Linienfolge zu einem funktionsfähigen Modul zu gestalten.

Der Roboter wurde ausgestattet mit einer Arduino-Leiterplatte, die mit der Arduino Duemilanove Platine verglichen werden kann. Als Systemherz des Arduino-Roboters gilt der ATMEGA328P. Dieser Mikrocontroller verfügt über 14 digitale Ein- und Ausgängen, wobei 6 als PulsBreitenModulierten (PBM-) Ausgänge verwendbar sind. Außerdem verfügt der Roboter über 6 Analogeingängen, einem 16MHz Kristalloszillator und einem USB-Anschluss zur Programmierung und Steuerung. Außerdem gibt es noch einen ISP-Steckverbinder, mit dem die mehr erfahrenen Hobbyisten selbst einen Bootloader programmieren können.

Der Roboter arbeitet mit einer Bordspannung von 5V und ist eventuell auch zufrieden mit der USB-Spannungsquelle, was beim testen und programmieren einige Handgriffe einspart. Ausgesprochen nützlich sind in diesem Roboterkonzept die Steckplätze, auf denen Sie die eigenen oder die AREXX-Erweiterungsplatinen aus der ASURO-Reihe platzieren können

3.3 HINTERGRUNDINFO ZUR ARDUINO SOFTWARE

Die Arduino Software gehört zur Open Source-Kategorie und ist deshalb für jeden verfügbar. Deshalb sind auch die Quellcodes der Programmierumgebung frei verfügbar.

Die Arduino Programmierumgebung verfügt über einem Texteditor, einem Meldefenster und einer Textkonsole. Die Programmierumgebung kann direkt mit dem AAR kommunizieren und auf einfachster Weise Programme in den Prozessor zu übertragen.
In Arduino geschriebenen Programme werden „Sketches" (in Deutsch: „Skizzen") genannt. Der Quellcode wird mit dem Texteditor geschrieben. Die Sketch-Datei wird mit der Extension „.ino" auf der Festplatte des PCs gespeichert.

Im Meldefenster erscheint die Mitteilung dass die Datei gespeichert wird und eventuell eine Meldung überetwaigen Fehler im Quellcode. Rechts unten im Bildschirmfenster wird das aktuelle Arduino-Board und die seriellen Schnittstelle dargestellt .

Arduino verfügt über Bibliotheken („Libraries") mit extra Funktionen. Eine Bibliothek („Library") ist ein Paket mit verschiedenen zusammengesetzten Funktionen, die nun nicht mehr wiederholt neu geschrieben werden müssen. Diese Funktionen können in Arduino einfach aufgerufen werden.

Man kann ein Arduino-Programm in drei Teilen gliedern: Struktur, variable beziehungsweise konstante Definitionen und Funktionen. Eine Arduino-Struktur besteht aus einem Setup und einer Schleife (die „Loop"-Funktion). Der Setup ist zuständig für die Initialisierung der Variablen, Pin-Einstellungen („Pin-Modes") und Bibliotheken („Libraries").

Die Schleife („Loop") wird ständig wiederholt, sodass das Programm immer wieder reagieren kann. „Variable"-Definitionen werden zur Datenspeicherung benötigt, während Konstanten zum Beispiel verwendet werden um einen Pin als Ein- beziehungsweise Ausgang zu definieren und dafür zu sorgen, dass einem Pin eine Spannung zugeordnet wird.

4. Erste Schritte auf dem Weg zur Installation

4.1 Download und Installation der Arduino Software

Installieren Sie bitte zuerst die Erstversion der Arduino Software von der CD, da wir dann sicher sind, dass diese Version richtig funktioniert. Später können Sie auch die Arduino Webseite besuchen und von dort die aktuelleren Versionen abrufen.

WICHTIG:

Es ist möglich dass manche ARDUINO-Softwareversionen mit diversen Versionen der Applikationssoftware Schwierigkeiten bereiten. Manchmal muss man auch nach der Aktualisierung der ARDUINO Software Ihre Applikationsprogramme bei Bedarf (insbesondere bei Fehlfunktionen) nacharbeiten!

4.2 Die Programmiersprache Arduino

Der Syntax der Programmiersprache Arduino wird in der offiziellen Arduino-Webseite dokumentiert. Vertiefen Sie bitte Ihre Sprachkenntnisse dieser Programmiersprache soweit Sie diese benötigen.

4.3 Installation eines USB-Treibers

Wenn Sie die Arduino-Platine erstmalig mit dem PC verbinden sollte Windows den Installationsprozess für den Treiber starten. Auf Windows Vista und aktuelleren Windowsinstallationen wird der Treiber normalerweise automatisch aus dem Netz geladen und installiert.

Selektieren Sie dann bitte die serielle Schnittstelle für die Arduino-Platine im Menü Tools > Serial Port. In der Regel bietet das System COM3 oder höher an. COM1 und COM2 sind normalerweise für in der Hardware definierten serielle Ports reserviert. Um das herauszufinden können Sie das Arduino System abkoppeln und das besagte Menü nochmals eröffnen. Der nun neu hinzugefügte Eintrag ist dann vermutlich der korrekte Port. Schließen Sie die Platine wieder an und selektieren Sie dann diesen seriellen Port

4.4 AAR Hardware

4.4.1 Das Anschließen von Akkus

Der Roboter wurde dimensioniert für eine Stromversorgung aus einem Batteriebehälter für vier 1,5V Batterien. Falls Sie stattdessen aufladbare Akkus verwenden, sollte die Drahtbrücke JP4 für die Verwendung von Akkus durchverbunden werden.
(siehe Abb. 1 Nummer 4).

ACHTUNG!
Sobald die Drahtbrücke JP4 platziert wird, geht der Schutz gegen fehlerhaften Polarität der Stromversorgung verloren. Fehlerhafte Anschlüsse kann dann den Roboter unwiderruflich beschädigen.

Schließen Sie den Batteriebehälter an sowie in der Abbildung dokumentiert (siehe Abb. 2)

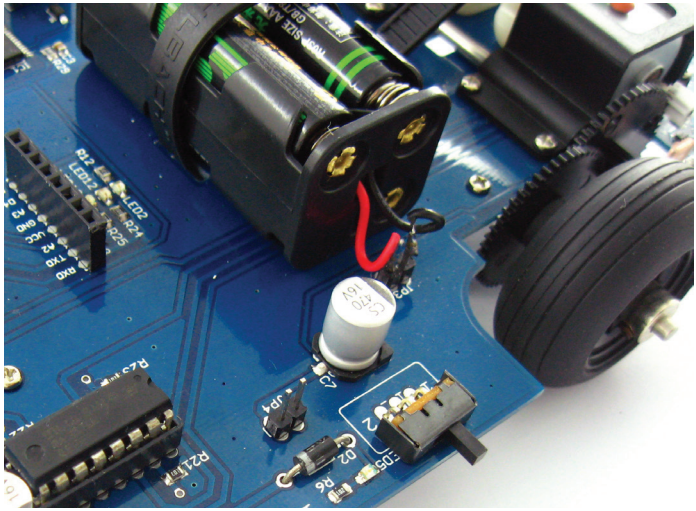


Abb. 2: Anschluss des Batteriebehälters

Schalten Sie nun den Roboter ein mit dem Schalter. Der neben dem Schalter platzierten LED5 wird jetzt aufleuchten.

4.5 ARDUINO Software

4.5.1 Programmierung des Roboters mit der Arduino Software.

Schließen Sie den Roboter mit einem USB-Kabel auf dem PC an. Falls der Roboter auf dem USB-Port angeschlossen ist, benötigt Arduino nicht unbedingt eine Batteriespannung. Der USB-Anschluss des PCs übernimmt dann die Funktion der Stromversorgung.

ACHTUNG:
Der Roboter ist immer eingeschaltet sobald dieser auf dem PC angeschlossen ist, der Schalter und LED5 sind nur bei Batteriebetrieb aktiv.

Öffnen Sie jetzt die Arduino-Software (siehe Abb. 3a).

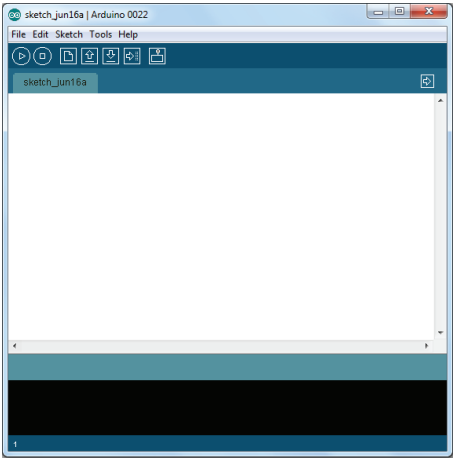


Abb. 3a Arduino Software

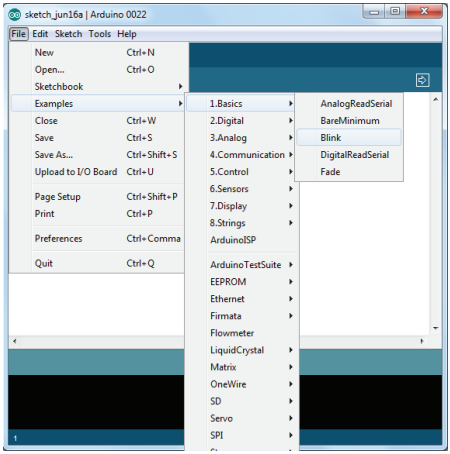
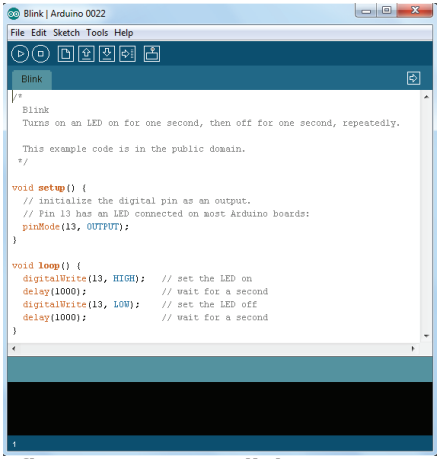


Abb. 3b Öffnen des Blink-Programms

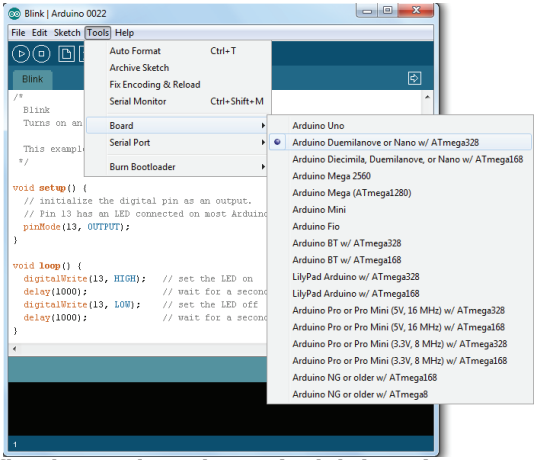
4.5.2 Selektieren eines Arduino Programms

Als einfaches Beispiel laden wir zuerst das Programm „blink" in den Roboter, das dafür sorgt, dass LED1 anfängt zu blinken.

Klicken Sie dazu in der Arduino Software auf File>Examples>1. Basics>Blink (siehe Abb. 3b), sodass nachfolgende Informationen auf dem Bildschirm erscheinen (Abb. 4a).



Afb. 4a Programm Blink

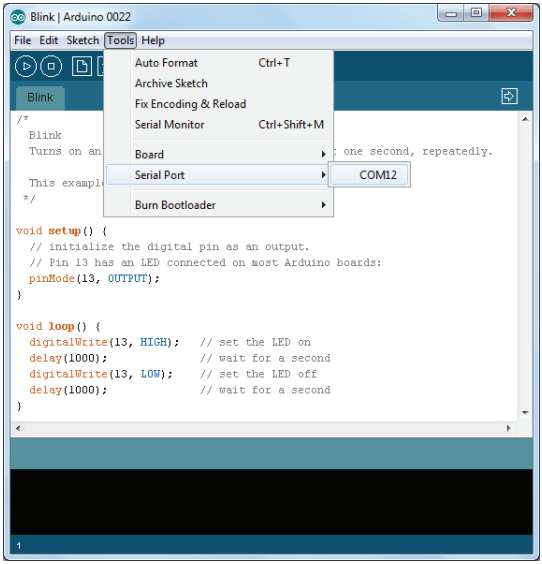


Afb. 4b Board Festlegen (Selektieren)

Wir müssen jetzt den korrekten Arduino-Platinentyp festlegen.
Klicken Sie dazu auf Tools>Board> Arduino Duemilanove or[11] Nano w/Atmega328 (siehe Abb. 4b)

4.5.3 Festlegung der COM-Schnittstelle

Der nächste Schritt legt die korrekten COM-Schnittstelle in der Arduino-Software fest. Die richtige COM-Schnittstelle (COM-Port) für den Anschluss des Roboters ist COM 12.
Zur Festlegung der COM-Schnittstelle folgen Sie die Menuwahl: Tools>Serial Port>COM 12.
(Siehe Abb. 5)



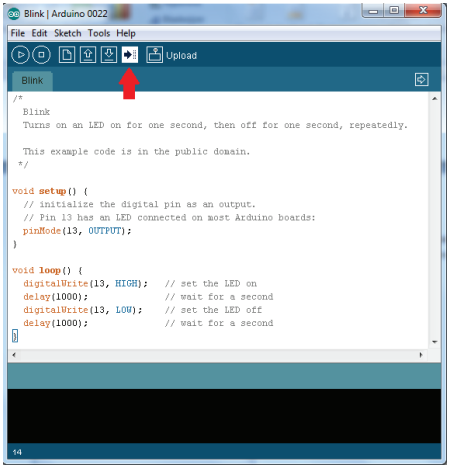
Afb. 5
Festlegung des Com-Ports

[11]Englischer Ausdruck für: „oder"

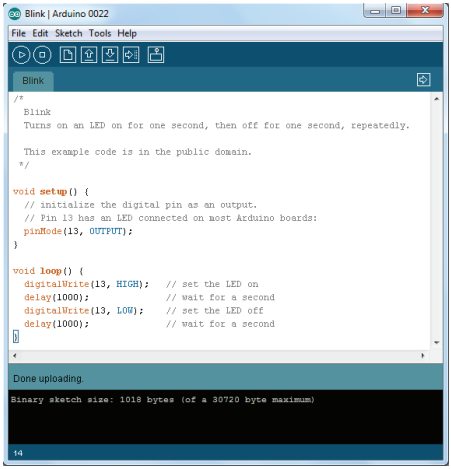
4.5.4 Übertragung eines Programms zum Arduino Roboter

Klicken Sie anschließend auf die mit einem roten Pfeil markierte Taste (oder im Menu auf „File>Uploading to I/O board“) um dieses Programm an den angeschlossenen Arduino Roboter zu übertragen (siehe Abb. 6).

Im Statusbalken meldet die Software jetzt dass das System das Programm compiliert und anschließend einen Upload startet.



Afb. 6 Übertragung eines Programms zum Arduino Roboter (Upload-Vorgang).



Afb. 7 „Done uploading“ - Meldung: „Upload-Vorgang beendet“

Jetzt können Sie den Roboter vom PC abkoppeln indem Sie das USB-Kabel entfernen, die Batteriespannung anschließen und den Roboter starten.

Für weitere Informationen und Downloads laden wir Sie ein die Foren zu besuchen auf den Webseiten:

www.arexx.com --> Forum
www.roboternetz.de --> Forum

5. Hintergrundinformation zur H-Brücke

Die H-Brücke ist eine Elektronikschaltung womit man mittels vier Schalter zum Beispiel einen Gleichstrommotor umpolen kann. Ein solcher Schalter wird in der Robotertechnologie oft angewandt um Motoren in zwei Richtungen an zu steuern.
Meistens sind für solche Schaltungen integrierte Module verfügbar, aber zur Verdeutlichung der Funktionsweise und Dimensionierung der Stromversorgung kann es gelegentlich interessant sein auch eine ältere Schaltung zu studieren.

5.1 Eine H-Brücke für 3 Volt Stromquellen

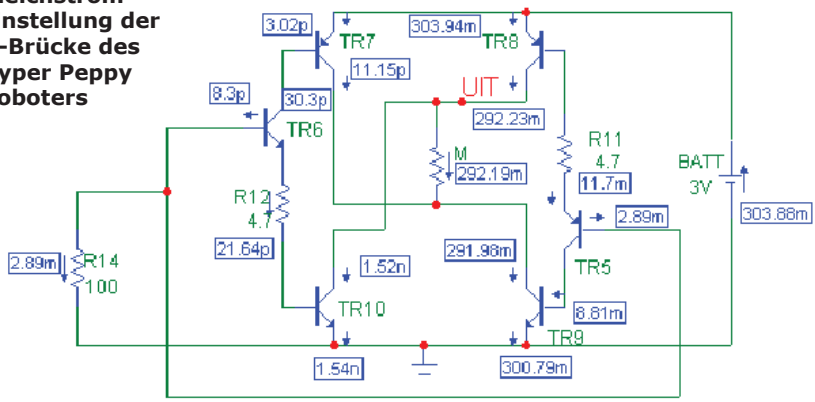
In der Endstufe des Hyper-Peppy roboters wurden die vier Schalter der H-Brücke als zwei PNP-Transistoren TR7 und TR8, beziehungsweise NPN-Transistoren TR9 und TR10 gestaltet. In dieser Schaltung dürfen immer nur zwei dieser Transistoren den Strom durchlassen und zwar so, dass der Strom:

über TR7 und TR10 oder aber
über TR8 und TR9

durch den Motor M geführt wird.

Mit Hilfe des (kostenlosen) Microcap-Simulators können wir recht einfach die Gleichstromeinstellung dieser Schaltung kalkulieren lassen und im Schaltbild ablesen.

Afb. 8: Gleichstrom-einstellung der H-Brücke des Hyper Peppy Roboters



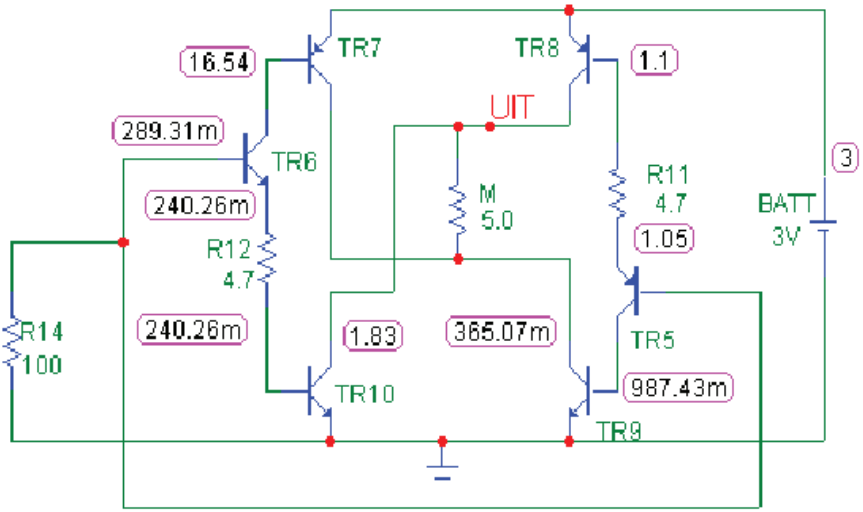
In der Endstufe ist M der Motor und wird die Steuerungsstufe des Vorverstärkers mit einem Widerstand R14 simuliert. Dieser Widerstand R14 erdet die Basisanschlüsse der Transistoren TR6 und TR5. Es leitet deshalb auch nur die Endstufe auf der rechten Seite einen nennenswerten Strom.

Die Transistoren TR8, TR5 und TR9 führen Strom und die drei übrigen sind gesperrt. Falls wir R14 an einer positiven Spannung anschließen, sperren wir den Rechten Zweig der Endstufe und wird der Motorstrom umgepolt.

Der Microcap Simulator erlaubt uns die Stromstärke in jedem Bauteil zu berechnen und im Schaltbild an Bildschirm abzulesen. Der Gesamtstromverbrauch der Endstufe beträgt etwa 300mA bei 3Volt Batteriespannung.

Das erstaunliche dieser Schaltung ist dass die Endstufe bei 3V mit Silizium Transistoren arbeitet. Normalerweise beträgt die Kniespannung eines Silizium Transistors 0,7V. Der Motor wurde jedoch zwischen den Kollektoren geschaltet, die im Idealfall nur 0,3V Spannungsabfall verzeichnen. Für den Motor bleibt in der Praxis immer noch ein großzügiger Restwert von 1,5V übrig. Die vom Microcapsimulator berechnete Spannungswerte sind in Abbildung 9 dokumentiert.

Abb. 9: H-Brücke mit dem L293D-Chip

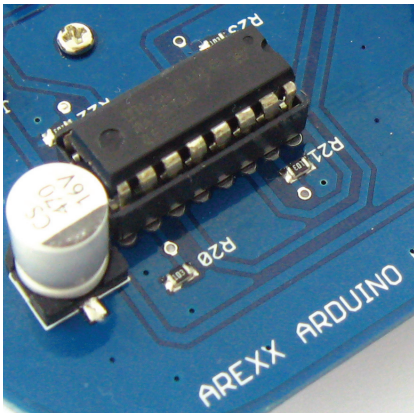


Die 3V-Stromversorgung ist eine ideale Ausgangsbasis für ein Roboter, der von nur 2 Batterien gespeist wird. Die PNP-Transistoren sind jedoch schlecht geeignet für eine Implementierung in einen integrierten Schaltkreis wie der L293D. Ein IC ist dagegen in anderer Hinsicht vorteilhaft, wie zum Beispiel im Bereich der Zuverlässigkeit, Absicherung gegen Schaltfehler und geringeres Platzbedarfes sowie Gewichts. Aus diesen Gründen haben wir den AAR-Roboter mit einem L293D-chip mit doppelter H-Brückenschaltung ausgestattet, der uns erlaubt gleichzeitig zwei Motoren anzusteuern.

5.2 Eine H-Brücke für 4,5 Volt

Der Schaltkreis L293D (siehe Abb. 10) kann pro Ausgangsstufe einen Ausgangsstrom von 600 mA (maximal: 1,2A Spitzenwert) ansteuern. Die Stromversorgung der Endstufe (VCC2) darf zwischen 4,5V und 36V variieren, sodass wir den L293D-Chip als Ideallösung zur Ansteuerung von Gleichstrommotoren betrachten dürfen.

Die minimale Stromversorgungsspannung (VCC2) beträgt jedoch 4,5V, sodass wir minimal 4 Batteriezellen oder Akkuzellen einsetzen müssen. Diese Investition erhöht selbstverständlich wiederum das Gewicht des Roboters. Das ist der Preis den wir für den Einsatz des modernen ICs bezahlen müssen.



Afb. 10
H-Brücke mit dem L293D-Chip

6. Odometrie-Systeme

In diesem Kapitel werden wie einige interessante Konzepte für die Anwendung des AAR-Roboters beschreiben. Es sind Ideen für Studien und Kunstprojekte. Die Entwicklung solcher Arduino-Software mag uns beim Programmieren des Mikrocontrollers beflügeln und beseelen.

6.1 Linienfolger, Farbsucher und Farbflüchter

Mit lichtempfindlichen Sensoren kann man einen Roboter zum Linienfolger, Farbsucher und Farbflüchter programmieren. Im ersten Fall soll der Roboter artig und getreu durchgezogenen Linien in einem 8-Muster folgen, in dem die Endloslinie ihn wie in einer Tretmühle herumirren lässt.

Im zweiten und dritten Fall wird der Roboter grundsätzlich rotes Licht aus dem Weg gehen und dabei vielleicht gleichzeitig vom grünen Licht angezogen werden. Solche Verhaltensmuster gehören bereits zu den praktischen Strategien der einfachen Lebensformen.

6.2 Angsthasen und Musikliebhaber

Interessant ist auch das Verhaltensmuster, das auf die umgebenden Geräuschkulisse reagiert. Ein schreckhafter Roboter mit eingebautem Mikrofon könnte eine Musik mit vielen schweren Bässen aus dem Weg gehen, aber gleichzeitig eine Vorliebe für hohen Flötentöne zeigen. Die Vorliebe für hohen Flötentöne könnte gar die Angst vor schweren Bässen übertönen. So kann man den Roboter zwingen trotz schwerer Rockmusik die Quelle der zauberhaften Blockflöten aus zu filtern und zu suchen.

Verhaltensmuster, die auf Bässen und Hochfrequente Musiktönen, Licht und Farben reagieren, benötigen nur einige wenigen Sensoren, wie zum Beispiel ein Mikrofon, zwei Tonfiltern, und wenige lichtempfindlichen Sensoren, die mit Farbfiltern ausgestattet werden.

6.3 Komplexe Linienfolger

Roboter, die Linien folgen oder Linienmuster aus dem Weg gehen, verwenden in der Regel eine Lichtquelle und zwei oder mehr Lichtsensoren, womit das Suchsystem eine Linie identifizieren und folgen kann. Zunächst einmal kann man den Roboter mit einer speziellen Suchroutine ausstatten, in dem der Detektor im Suchmodus nach einer ganz speziellen Taktik zu suchen beginnt, zum Beispiel in einem Spiralmuster in immer größeren Kreisen herumfährt bis der Sensor ein auffälliges Linienmuster entdeckt und die gefundene Linie zu folgen beginnt.

Das Schreiben einer Software, die eine solche Suchprozedur für willkürliche Linienmuster statistisch gesehen befriedigend löst, gehört bereits zu den ausgesprochen schwierigen Programmieraufgaben.

6.3.1 Komplexes Verhaltensmuster (als Programmieraufgabe)

Das Projekt kann jedoch noch erweitert werden indem man die Suchprozedur in einem willkürlich buntgefärbtem Linienmuster stattfinden lässt, in dem der Roboter durch Lärmmuster verscheucht wird und daraufhin sofort die nächstbeste rote Linie sucht, die ihn in die sichere Unterkunft einer dunklen „Garage" führt.

Sobald die Geräuschkulisse sich für eine gewisse Zeit beruhigt hat, kann nun der Roboter sich vorsichtig aus seiner „Garage" heraus begeben und die Suche nach einer grünen Linie, welche das Fahrzeug nach eine zweite „Garage" mit intensiver grünen Beleuchtung führt, in dem sich der Roboter auch unter lauter Bassmusik sicher „daheim" fühlt.
Sobald man jedoch der Musik hohe Blockflötentöne beimischt, wird der Roboter unruhig. Er verlässt seine grüne Behausung um die rote Linien zu seinem dunklen Verlies zu suchen.

Man kann sich denken, dass dieses Verhaltensmuster mit Linienfolger, Farbabhängigkeiten, unterschiedlichen Geräuschquellen und einem komplexen Rollenmuster eine systematische Organisation der Softwarestruktur erfordert. Nur eine sauberes, modular strukturiertes Konzept führt in solchen Fällen zum Ziel, indem der Roboter sich unter allen Bedingungen stabil und zuverlässig am zuvor spezifizierte Verhalten orientiert.

Aus der Komplexität der benötigten Software mag der Programmierer in Staunen versetzt werden und eine Bewunderung für die lebende Organismen entwickeln, die ein solches Verhaltensmuster kombinieren mit der regelmäßigen Suche nach Nahrung und einer erfolgreichen Fortpflanzungsstrategie. Es ist wahrlich die großartige Leistung der Natur solche Verhaltensmuster immer wieder zu perfektionieren und am Leben zu erhalten.

7. Programmierung des Bootloaders

Achtung!
Die Verfahren in diesem Kapitel setzen Programmiererfahrung voraus !

Man kann den Arduino Bootloader zum Beispiel mittels STK500 hochladen. Zur Übertragung der in Arduino geschriebenen Programmen in den Atmega Mikrocontroller muss der Atmega-Prozessor von einer speziellen Arduino-Bootloader versehen werden. Der Bootloader sorgt dafür, dass die übertragene Codezeichen an der korrekten Stelle des Atmega-Speichers geschrieben werden.

Zum Anlegen des Bootloaders benötigen wir folgende Komponenten:

* ein AVR Programmer Board (zum Beispiel das STK500 Board)
* eine 12 Volt Stromversorgung
* AAR Roboter mit einer auf der Leiterplatte verfügbaren ISP-Steckverbinder (Abb. 11)
* PC mit einem physikalischen COM-Port (vorzugsweise kein USB-RS232 Konverter, zur Vermeidung der Risiken durch Timing-Fehlern)

Installieren (oder aktualisieren) Sie die aktuellen Version der Arduino-Software, die sich auf der Webseite www.arduino.cc befindet. Die abgerufene Datei wird normalerweise vom Typ .ZIP oder .RAR sein. Entpacken Sie diese Dateien und platzieren sie diese auf Ihrer Festplatte.

Verwenden Sie nun zum Beispiel WINAVR um den Arduino Bootloader in den Roboter zu übertragen.

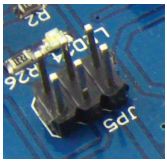


Abb. 11: ISP-Steckverbinder

Achtung!
Die ARDUINO Software gehört komplett zur Kategorie Freeware und es kommt schon mal vor, dass unterschiedliche Versionen der Arduino Software und Arduino Bootloader-Programme sich nicht richtig vertragen!

Besuchen Sie bei Problemen in diesem Bereich die diversen Arduino-Websites beziehungsweise Foren!

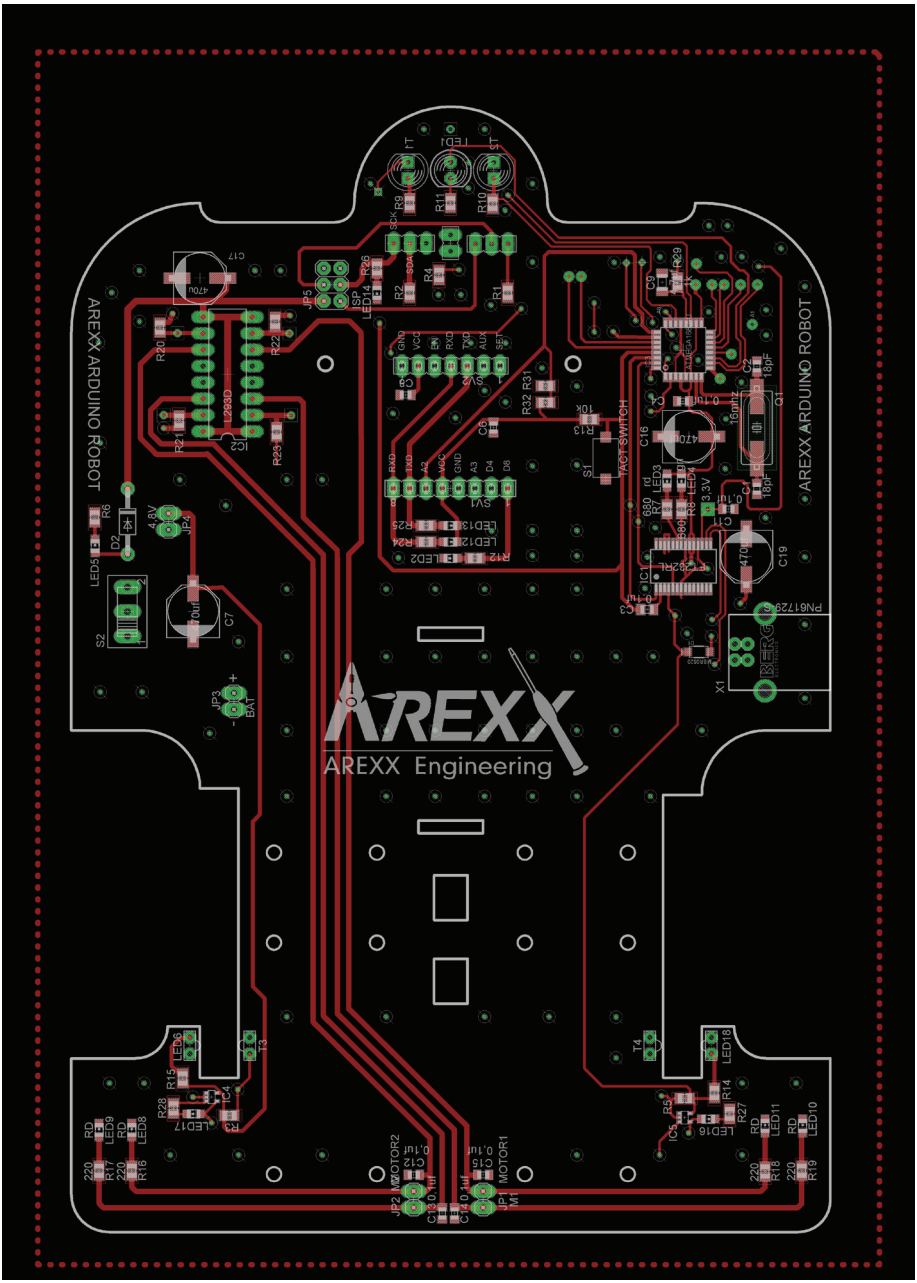
APPENDIX

A. Teileliste

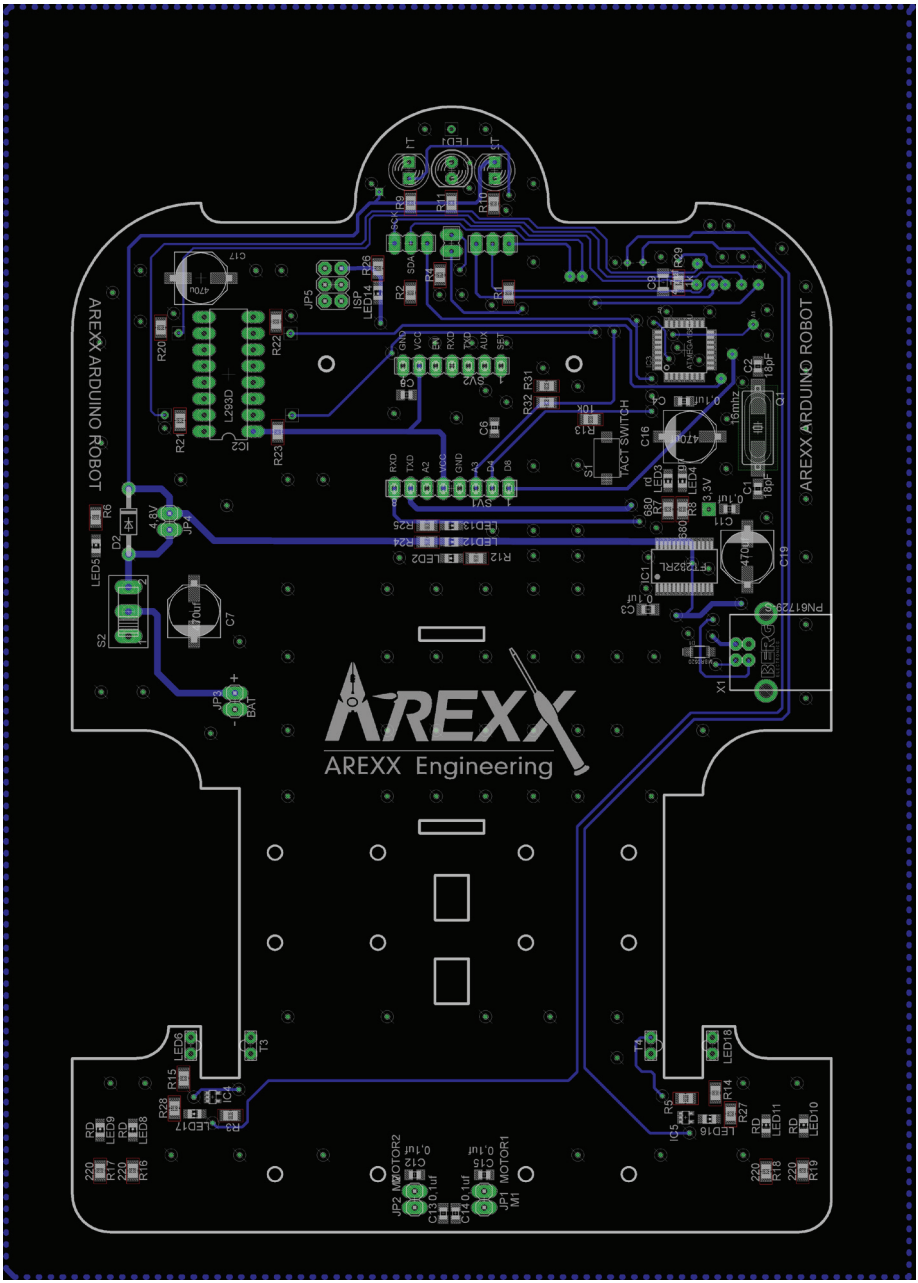
Teil	Wert	Ausführung
C1	18pF	0805
C2	18pF	0805
C3	0.1uF	C0805K
C4	0,1uF	0805
C6	0,1uF	0805
C7	470uF	CPOL-USF
C8	0,1uF	0805
C9	4,7uF	1206
C11	0,1uF	0805
C12	0,1uF	0805
C13	0,1uF	0805
C14	0,1uF	0805
C15	0,1uF	0805
C16	470uF	CPOL-USF
C17	470uF	CPOL-USF
C19	470uF	CPOL-USF
D1	MBR0520	SOD-123
D2	1N4001	DO41-10
IC1	FT232RL	SSOP28
IC2	L293D	DIL16
IC3	ATMEGA168-AU	ATMEGA168-AU
IC4	74AHC1G14DCK	74AHC1G14DCK
IC5	74AHC1G14DCK	74AHC1G14DCK
JP1	M1	1X02
JP2	M2	1X02
JP3	BAT	1X02
JP4	4,8V	1X02
JP5	ISP	2X03
SV2	fem header	FE07-1
T1	SFH300	LED5MM
T2	SFH300	LED5MM
T3	LPT80A	LPT80A
T4	LPT80A	LPT80A
U$1	3,3V	PIN-T
U$2	FE03-1	FE03-1
U$3	FE03-1	FE03-1
U$4	FE02-1	FE02-1
X1	PN61729-S	PN61729-S
LED1	Rd	LED5MM
LED2	Bl	LEDCHIP-LED0805
LED3	Rd	LEDCHIP-LED0805
LED4	Gn	LEDCHIP-LED0805
LED5	Bl	LEDCHIP-LED0805
LED6	Rd	LEDIRL80A

Teil	Wert	Ausführung
LED8	Rd	LEDCHIP-LED0805
LED9	Rd	LEDCHIP-LED0805
LED10	Rd	LEDCHIP-LED0805
LED11	Rd	LEDCHIP-LED0805
LED12	Gn	LEDCHIP-LED0805
LED13	Rd	LEDCHIP-LED0805
LED14	Bl	LEDCHIP-LED0805
LED16	Rd	LEDCHIP-LED0805
LED17	Rd	LEDCHIP-LED0805
LED18	Rd	LEDIRL80A
Q1	16mhz	CRYSTALHC49UP
R1	20k	R-US_R0805
R2	20k	R-US_R0805
R3	1k5	R-US_R0805
R4	220	R-US_R0805
R5	1k5	R-US_R0805
R6	1k	R-US_R0805
R7	680	R-US_R0805
R8	680	R-US_R0805
R9	20k	R-US_R0805
R10	20k	R-US_R0805
R11	220	R-US_R0805
R12	220	R-US_R0805
R13	10k	R-US_R0805
R14	220	R-US_R0805
R15	220	R-US_R0805
R16	220	R-US_R0805
R17	220	R-US_R0805
R18	220	R-US_R0805
R19	220	R-US_R0805
R20	10k	R-US_R0805
R21	10k	R-US_R0805
R22	10k	R-US_R0805
R23	10k	R-US_R0805
R24	220	R-US_R0805
R25	220	R-US_R0805
R26	220	R-US_R0805
R27	220	R-US_R0805
R28	220	R-US_R0805
R29/C3	0.1uF	R-US_R0805
R31	10k	R-US_R0805
R32	12k	R-US_R0805
S1	TACT SWITCH	TACT_SWITCH
S2	255SB	255SB
SV1	fem header	FE08-1

B. Hauptplatine Obenseite



C. Hauptplatine Unterseite



D. Schaltbild AAR