

# Unterrichtspraxis mit qfix Robotern



Ein Lehr-, Lern- und Ideenbuch für Schüler ab der Klasse 7.

Vom Einstieg bis zur Umsetzung  
in die C++ Programmierung.  
Kopiervorlagen, Arbeitsblätter

© Copyright 2011 Frank Engeln

**Autor:**

Frank Engeln  
Bursibantstraße. 3  
48429 Rheine  
Web: [www.engeln.info](http://www.engeln.info)

Alle Rechte vorbehalten

**Herausgeber:**

qfix robotics GmbH  
Erich-Rittinghaus-Straße 2/2  
89250 Senden  
E-Mail: [info@qfix.de](mailto:info@qfix.de)  
Web: [www.qfix.de](http://www.qfix.de)

Alle Rechte vorbehalten

Der Nachdruck, auch auszugsweise, ist nur nach vorheriger Genehmigung durch den Herausgeber gestattet.

Unterrichten mit Robotern und der Programmiersprache C++.

Die Gesellschaft für Informatik e.V. (GI) fordert in einem Memorandum, der "digitalen Spaltung in Deutschland entgegen zu treten und die Grundlagen für das Verständnis moderner digitaler Hilfsmittel bereits in der Schulausbildung zu legen". Konkret plädiert die GI für die Einführung eines Pflichtfachs "Informatik" an jeder Schule.

Übergeordnetes Ziel des Informatikunterrichts ist es, Schülerinnen und Schüler bestmöglich auf ein Leben in einer Gesellschaft vorzubereiten, die immer mehr durch den Einsatz von Informations- und Kommunikationstechnologien sowohl im privaten als auch im beruflichen Bereich geprägt ist. Da in allen gesellschaftlichen Bereichen die Nutzung und der Umgang mit Informatiksystemen unumgänglich geworden sind, ist eine solide informatische Grundkenntnis die Voraussetzung für die Teilnahme am beruflichen und gesellschaftlichen Leben.

In den Schulen werden häufig Roboter der Firmen Lego® oder Fischertechnik® eingesetzt. Es handelt sich hierbei um gute Produkte, die aber ihre Grenzen in der Anwendung aufzeigen, was beispielsweise für die Mechanik/Statik gilt. Was ist aber mit den älteren Schülern, mit der Oberstufe? Wurde z.B. in der Sek. I. mit Lego Robotern und der Programmiersprache NXC gearbeitet, fällt ein Wechsel in eine höhere Programmiersprache nicht schwer. Aber auch als Einstieg in die Welt der Robotik können die in diesem Buch vorgestellten Controllerboards samt Zubehör sehr gut genutzt werden.

Dieses Buch ist eine Einführung in die Programmiersprache C++ und setzt keine Erfahrung im Programmieren mit voraus. Wer allerdings schon mit RobotC (Lego Mindstorms NXT) oder der Programmiersprache C (Asuro) gearbeitet hat, ist klar im Vorteil.

Das Handbuch ist eine aus der Praxis entstandene Konzeption zur Einführung in die C++ Programmierung mit qfix Robotern ab Klasse 7.

**In diesem Buch wird das qfix SoccerBoard verwendet. Das MiniBoard wird identisch programmiert, verfügt aber über weniger Ein- und Ausgänge. Im Anhang finden Sie eine Befehlsübersicht.**

Ich habe absichtlich keinen fertigen Roboterbausatz als Grundlage genommen. So kann man mit einigen Teilen beginnen und gemäß des eigenen Interesses und Lernfortschritts sein Set ergänzen.

## Inhalt

Was ist C++ .....	6
Was sind qfix Roboter?.....	7
Der Roboterbausatz „MiniBot“ .....	7
Der Roboterbausatz „Crash Bobby“ .....	7
Umfangreiches Zubehör .....	8
SoftwareInstallation .....	9
Programm schreiben .....	11
Kommentare.....	14
Erste Töne.....	15
if – else .....	17
<b>Eine einfache Ampelanlage .....</b>	<b>19</b>
Motoren.....	22
Programmanweisungen.....	23
Durchmesser, Radius und Umfang .....	25
Motoren starten mit einem Button.....	28
Unterprogramm oder Subroutine .....	29
Schleifen .....	31
Der Inkrementierungs - und Dekrementierungsoperator.....	33
Variable.....	35
Konstante.....	40
Zufallszahlen .....	43
Sensoren .....	45
Display.....	47
Infrarot Distanzsensoren .....	48
Sensoren – Lichtsensor .....	50
Lichtmessung .....	53
Sensoren – Tastsensor .....	54
Potentiometer .....	55
Aufgabenblatt .....	57
Programmbeispiele.....	58
C++- Befehle im Überblick .....	61
Das überwachte Wohnhaus .....	67
Siegeszug der künstlichen Intelligenz .....	69
Roboter – Feind oder Freund? .....	70
Künstliche Intelligenz im Dienste des Menschen.....	73

Lösungen.....	75
Internetrallye: Kleine Geschichte(n) der Roboter .....	77
Lösung Kleine Geschichte(n) der Roboter:.....	79
Finde mit Hilfe des Internet folgende Fragen heraus .....	80
Lösung: Finde mit Hilfe des Internet folgende Fragen heraus.....	81
Anhang.....	83
Anhang B.....	85

## WAS IST C++

**C++** ist eine von der ISO (Internationale Organisation für Normung) standardisierte höhere Programmiersprache. Sie wurde ab 1979 von Bjarne Stroustrup bei AT&T als Erweiterung der Programmiersprache C entwickelt.

### **Anwendungsgebiete von C++:**

- C++ ist eine allgemein verwendbare Programmiersprache
- Hauptsächliches Anwendungsgebiet ist die Systemprogrammierung im weitesten Sinne.
- Ein weiterer Einsatz von C++ liegt auch in der Spieleprogrammierung

Besonders zeitkritische Spiele wie 3D-Shooter (zum Beispiel Counter Strike) profitieren von der hohen Geschwindigkeit von C++. Vorteil gegenüber C ist die Objektorientierung mit den Klassen. Dies verlangt neben dem einfachen Erlernen neuer Sprachelemente auch eine neue "objektorientierte" Denkweise. Herkömmliche Softwareentwicklung bestand oftmals darin, zur Lösung eines vorgegebenen Problems Algorithmen zu entwerfen und diese in Prozeduren zu gießen, die in einer Programmiersprache - wie zum Beispiel C - formuliert sind. Die objektorientierte Programmierung verwendet neue, andere Begriffe. Die einzelnen Bausteine, aus denen ein objektorientiertes Programm während seiner Abarbeitung besteht, werden als Objekte bezeichnet.

Da dieses ein Lehrerhandbuch ist nehme ich einmal das Beispiel „Schule“. Die Klassen einer Schule können durch ihre Bezeichnung, die Nummer des Klassenraums und die Klassengröße (Anzahl der Schülerinnen und Schüler) beschrieben werden, also durch drei Zeichenketten (9a, 121,24).

Die Schüler und Schülerinnen selbst werden durch ihren Namen und Vornamen identifiziert, zwei Zeichenkette (Engeln, Frank). Ebenso sind die Lehrerinnen und Lehrer über ihren Namen oder Namenskürzel, sowie ihren Fächern in der Klasse definiert, also drei und mehr Zeichenketten (ENG, Inf, Mu).

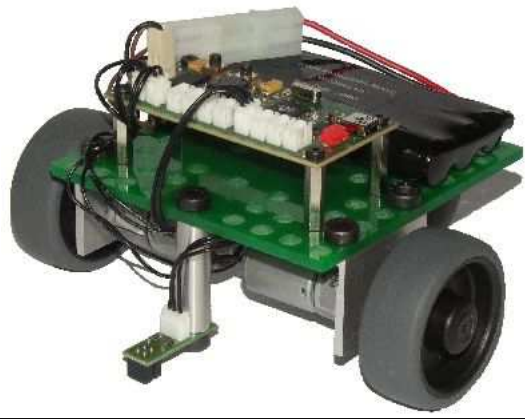
Alle diese Informationen werden zum Beispiel zu einem Stundenplan verarbeitet.

Grob gesehen gehören alle Schüler und alle Lehrer zur Oberklasse MENSCHEN. Der Schulpsychologe gehört ebenso zu den Menschen und dieser Schule. Er wird zur „Instanz“ dieser Klasse. „Tochterklassen“ sind spezielle Eigenschaftender übergeordneten Klasse. Z.B. Geburtsdaten, Anschrift, Wohnort etc.

## WAS SIND QFIX ROBOTER?

Die qfix robotics GmbH ist eine Firma im bayrischen Senden (nahe Ulm), die Mechatronikprodukte entwickelt und herstellt. Eine Produktfamilie sind die qfix Roboterbausätze, bei denen es sowohl komplette Roboterbausätze gibt als auch Zubehör wie Motoren, Sensoren, Mechanikbauteile, Kabel, Akkus, usw. Die qfix Bausätze bieten einen schnellen und einfachen Einstieg in die Robotik, setzen aber nach oben keine Grenzen. Mit den beiden Einstiegs-Sets "MiniBot" und "Crash-Bobby" lassen sich komplette Roboter inklusive Motoren, Sensoren und Controller-Board bauen. Sie hauchen ihm Leben ein, indem Sie Programme in C++ schreiben und auf den Controller laden.

### Der Roboterbausatz „MiniBot“



Der qfix Minibot Bausatz beinhaltet eine Roboterplattform mit Differentialantrieb, ein USB-Controllerboard sowie einen Bodensensor z.B. zum Entlangfahren einer Linie oder Erkennen einer Absturzgefahr von Tischkanten. Der MiniBot ist mit vielen qfix-Bauteilen kombinierbar und kann mit bis zu 8 Sensoren oder LCD-Display erweitert werden. Programmiert wird vom PC aus in C++ oder mit der grafischen Programmierumgebung "qfix Grape".

Lieferumfang:

- stabile Basisplatte aus Kunststoff mit vielen Anbaumöglichkeiten
- 2 Antriebsmotoren mit Metallgetriebe und 50mm-Räder
- 1 Stützrad
- Controllerplatine "MiniBoard" mit ATmega32-Controller, I2C-Bus und USB Interface
- Infrarot-Bodensensor
- Maße: ca. 150 x 120 x 70 mm

### Der Roboterbausatz „Crash Bobby“

Dieser Bausatz unterscheidet sich vom MiniBot durch stabilere Mechanikkomponenten und weitere Sensoren, mit denen Abstände zur Wand oder zu Hindernissen gemessen werden kann.

Lieferumfang:

- qfix Controller-Platine "MiniBoard" mit USB-Anschluss
- Stabile Basisplatte aus blau eloxiertem Aluminium
- 2 Getriebemotoren mit Rädern d=50mm

- 3 Infrarot Distanzsensoren (Sharp GP2D120)
- Montagematerial für Motoren, Sensoren, Platine, etc.
- Erforderliches Werkzeug
- CD mit Anleitung und Software (GNU C/C++ Compiler, Download-Tool)
- USB-Kabel
- Anbaumöglichkeiten
- Maße: 150 x 130 x 80 mm



### Umfangreiches Zubehör

Als Ergänzung zu den qfix Bausätzen und Plattformen gibt es umfangreiches Zubehör, z.B. Sensoren, Antriebe, Kabel, Aktuatoren, Controllerboards und einzelne Bauteile.



Mehr Infos unter: [www.roboter-in-der-schule.de](http://www.roboter-in-der-schule.de) oder [www.qfix.de](http://www.qfix.de).

## SOFTWAREINSTALLATION

Diese Anleitung bezieht sich auf die qfix Software CD Version 1.4.0. Bitte prüfen Sie, ob es sich bei Ihrer um die neueste Version handelt oder ob auf der Internetseite [www.qfix.de](http://www.qfix.de) unter "downloads" eine neuere Version zum Herunterladen bereitsteht. Die Versionsnummer Ihrer CD entnehmen Sie bitte der Datei README.txt bzw. LIESMICH.txt im Hauptverzeichnis der CD.

### **Voraussetzungen**

Die Software kann unter mindestens Windows XP installiert werden. Für das Laden von Programmen auf den qfix Controller wird ein USB Port benötigt. Zur Installation der qfix Software legen Sie die CD ein und starten das Setup-Programm "qfixSoftware-X.X.X.msi" (wobei X.X.X für die aktuelle Version steht).



Die folgenden Fenster immer mit Next > bestätigen.



**Den Anweisungen folgen und bestätigen.**

Durch diesen Installationsablauf wurden folgende Pakete installiert:

- GNU C++ Compiler
- Editor "Programmiers-Notepad"
- qfix C++ Klassenbibliothek
- qfix Tool "portSwitch"
- Dokumentation in Deutsch und Englisch
- Beispielprogramme

### Installation des Download-Treibers

Das Übertragen ("Downloaden") von Programmen auf das qfix Controllerboard kann über den parallelen Port (Drucker-Anschluss, LPT) oder einen USB Port erfolgen. Vor der Auswahl des entsprechenden Ports muss einmalig der entsprechende Treiber installiert werden. Sie müssen nur denjenigen Treiber installieren, dessen Port Sie benutzen möchten.

Bitte beachten Sie, dass Sie hierfür über Administratorrechte verfügen müssen.

### Installation des USB Treibers

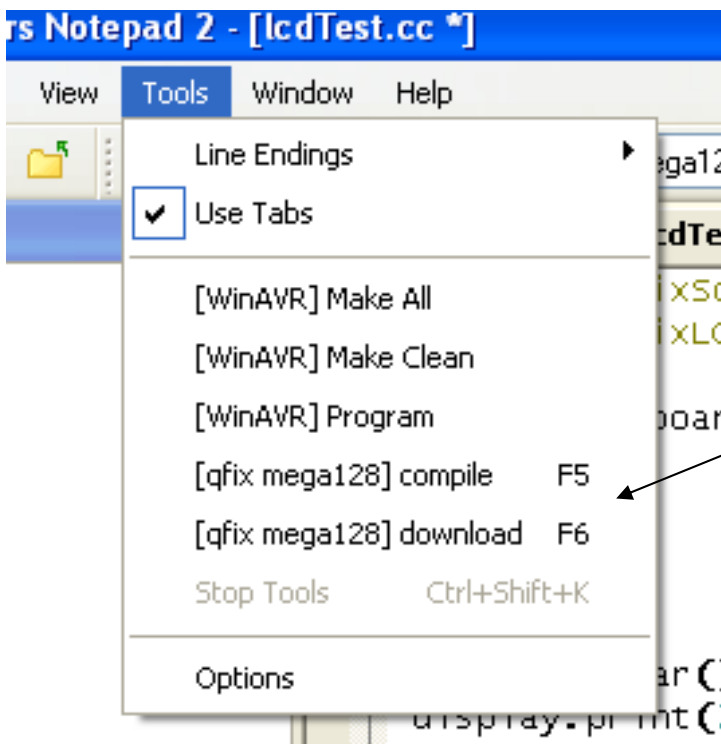
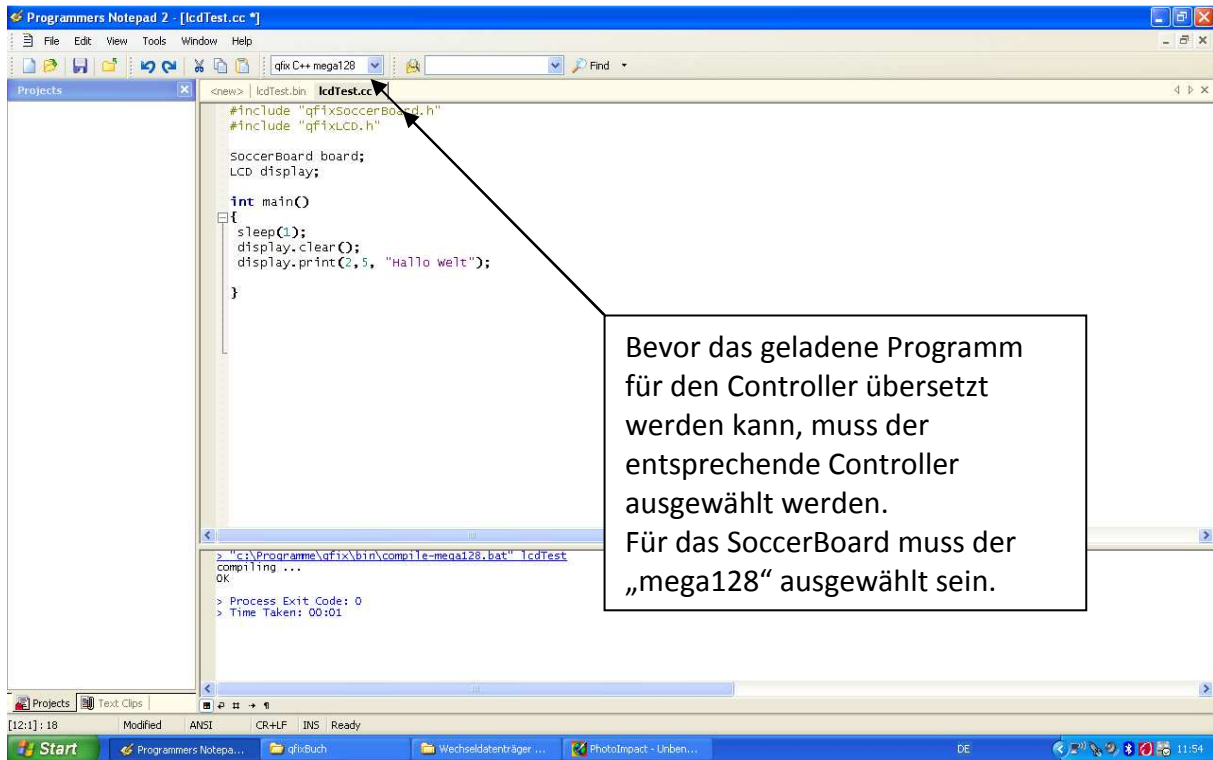
Vor der Installation des USB Treibers vergewissern Sie sich bitte, dass KEIN qfix Controllerboard mit dem PC verbunden ist!

Rufen Sie nun über das qfix Menü den Eintrag "Install USB driver" auf (bei Standardinstallation: Start → Programme → qfix software → Install USB driver).



## PROGRAMM SCHREIBEN

Die Programme werden im "Programmers Notepad" geschrieben. Hier werden sie auch übersetzt (kompiliert) und übertragen (download).



Die qfix Bausätze verwenden momentan drei verschiedene Mikrocontroller der Firma Atmel: mega32, mega128 und can128. Die folgende Tabelle zeigt, welcher Roboterbausatz welches Controllerboard benutzt und welcher Controller ausgewählt werden muss:

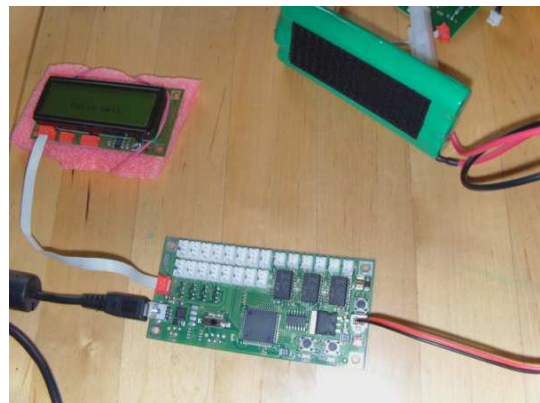
Roboterbausatz	Controllerboard	Controller
MiniBot	MiniBoard	mega32
Crash-Bobby	MiniBoard	mega32
OmniBot	SoccerBoard	Mega128
SoccerBot	SoccerBoard	mega128

Um mit der Arbeit beginnen zu können, benötigt man nur das Board und etwas Zubehör.

### HALLO WELT

Wie in allen Programmierbüchern wollte ich es mir nicht nehmen lassen ein Programm vorab zu zeigen. Der Klassiker – das „Hallo Welt2“ Programm.

Das Display ist zwar nicht in einem Baukasten enthalten, dennoch ein wichtiges Zubehör für die Roboter-Programmierung. Daten lassen sich so einfach ausgeben und ablesen. Zu dem Display komme ich später noch einmal. Mir ging es nur um die „Ehre“.



Programm:

```
#include "qfixSoccerBoard.h"
#include "qfixLCD.h"

SoccerBoard board;
LCD display;

int main()
{
    display.clear();
    display.print(2,5, "Hallo Welt!");
}
```

## STANDARD-PROGRAMM

Das Standardprogramm für alle Übungen. Es lädt die zuerst benötigte Bibliothek *qfixSoccerBoard.h* und legt dann das Objekt *robot* an, das für weitere Befehle genutzt werden kann. Durch den Ausdruck *robot.\**; können so sämtliche Befehle der Klasse *SoccerBoard* ausgeführt werden.

```
#include "qfixSoccerBoard.h"

SoccerBoard robot; // SoccerBoard ist die Klasse
                   // der Name z.B. „robot“ kann frei
                   // gewählt werden

int main()
{
    /* Hier kommt das Programm rein
}

```

Achtung:

- Alle Programme müssen die Endung \*.cc haben.
- In Programmname keine Umlaute (ä,ö,ü,ß) und keine Leerzeichen verwenden.

### Die häufigsten Programmierfehler

- **Semikolon am Ende eines Befehls vergessen.**
- **Anfang und Ende von Kommentaren müssen korrekt bezeichnet sein.**  
 Kommentaranfang: /\*  
 Kommentarende: \*/
- **Klammern beim Aufruf von Funktionen ohne Parameter.**  
 Falls zum Beispiel eine parameterlose Funktion getValue existiert, so ist der Funktionsaufruf `x = getValue;` /\*falsch.\*/  
 Der korrekte Aufruf wäre `x = getValue();`
- **Die logischen Operatoren && und || dürfen nicht mit den Bitoperatoren & und | verwechselt werden!**
- **Der Zuweisungsoperator = wird oft mit dem Vergleichsoperator == verwechselt.**  
 Zum Beispiel: `if (number = 9) { /* falsch */  
 number = 0; }`

Diese Sequenz ist syntaktisch korrekt, compiliert fehlerfrei, wird jedoch bei der Abarbeitung nicht das gewünschte tun. Anstatt der Variablen number den Wert 0 zuzuweisen, falls diese gleich 9 ist, wird number immer auf 9 gesetzt!

Abhilfe kann man schaffen, indem man sich angewöhnt, *immer zuerst den Wert, gegen den verglichen wird, hinzuschreiben.*

Also: `if (9 == number) { number = 0; }`

Wird hier statt == aus Versehen = geschrieben, so wird schon der Compiler eine Fehlermeldung ausgeben.

## KOMMENTARE

Kommentare werden benutzt, um Programme lesbarer zu machen. Sie beinhalten also verständliche deutsche (oft auch englische) Anmerkungen zum Programm. Der Compiler ignoriert den Text innerhalb eines Kommentars komplett.

In C++ gibt es zwei Arten von Kommentaren:

- Die erste Art gab es bereits in der Programmiersprache C. Hier beginnt ein Kommentar mit `/*` und endet mit `*/`. Sämtliche Zeichen zwischen diesen beiden Markierungen sind Kommentar. So kann man sogar über mehrere Zeilen kommentieren. Das ist wichtig für die Dokumentation im Programm oder für den Urheber des Programms, sowie Hinweise.

Beispiel: `/* Das ist ein Kommentar */`

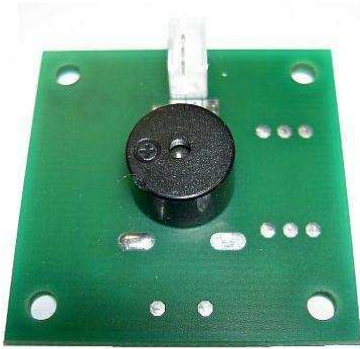
- Die zweite Art wird benutzt, wenn das Ende einer Programmzeile vollends als Kommentar gelten soll. In der Zeile steht dann meist ein Befehl, dann folgt die Kommentar-Markierung `//` und danach kommt der eigentliche Kommentar.

Beispiel: `board.sleep(3); // 3 Sekunden warten`

- Aufgabe:**
1. Erstellt ein Standard-Programm und schreibt Kommentare in der Form wie es unten zu sehen ist.
  2. Versuche einmal ein Bild aus Kommentare zu zeichnen.
  3. Macht einen Urheber oder Programmierhinweis

Beispielbild aus Kommentaren:

```
/*
    " " "
    ( o o )
---ooO--- ( _ ) ---Ooo---
*/
```



## ERSTE TÖNE

Der **Morsecode** oder **Morsekode** ist ein Verfahren zur Übermittlung von Buchstaben und Zeichen. Dabei wird ein konstantes Signal ein- oder ausgeschaltet.

Der Code verwendet drei Symbole, die *Punkt* (·), *Strich* (—) und *Pause* ( ) genannt werden, gesprochen als *Dit*, *Dah* und „Schweigen“. Genauer gilt Folgendes:

- Ein *Dah* ist üblicherweise dreimal so lang wie ein *Dit*.
- Die Pause zwischen zwei gesendeten Symbolen ist ein *Dit* lang.
- Zwischen Buchstaben in einem Wort wird eine Pause von *Dah* eingeschoben.
- Die Pause zwischen Wörtern beträgt sieben *Dits*.

— — — — · · · · · / — · — — — — · ·  
M O R S E (space) C O D E

Lateinische Buchstaben Buchstabe im Code

A · —	J · — — —	S · · ·
B — · · ·	K — · —	T —
C — · — ·	L · — · ·	U · · —
D — · ·	M — —	V · · · —
E ·	N — ·	W · — —
F · · — ·	O — — —	X — · · —
G — — ·	P · — — ·	Y — · — —
H · · · ·	Q — — · —	Z — — · ·
I · ·	R · — ·	

An das SoccerBoard wird der Beeper (siehe Bild oben) an den Analog Eingang 0 angeschlossen. Sollte kein Beeper vorhanden sein kann auch eine Power-LED benutzt werden. Beeper oder Power-LED müssen an die ersten vier Analog- oder Digital-"Eingänge" angeschlossen werden. Diese können nämlich als Power-Ausgang verwendet werden. Um die Power-Ein- und Ausgänge zu ansteuern benutzt man die Anweisung: `*.powerOn();` und `*.powerOff();`. In die Klammer kommt der Ein/Ausgang, z.B. 0 für den ersten, 1 für den zweiten, usw. Wie lange soll nun der Ton klingen? Die Dauer wird mit `*.msleep();` oder `*.sleep();` angegeben. Die Werte in Millisekunden oder Sekunden. Ersteres ist genauer.

Das Programm:

```
#include "qfixSoccerBoard.h"

SoccerBoard board;

int main()
{
    /* wenn LED Button 0 gedrückt wird startet dasProgramm */
    board.waitForButton(0);

    while (true)           //Endlosschleife
    {
        board.powerOn(0);
        board.msleep(100); //kurzer Ton
        board.powerOff(0);
        board.sleep(1);    //Pause zwischen den Tönen

        board.powerOn(0);
        board.sleep(1);    //langer Ton
        board.powerOff(0);

        board.sleep(1);    //Pause zwischen den Tönen
    }
}
```

- Aufgabe:**
1. Schreibe das o.a. Programm ab. Speichere dieses mit dem Namen „morsen.cc“ ab.
  2. Was bedeutet:  $\cdot - \cdot \cdot \cdot - - - \cdot - - - / - \cdot - \cdot \cdot - - - ?$

3. Programmiere deinen Namen im Morsecode. Schreibe dazu erst deinen Namen in Punkte, Strichen und Pausen auf. Schreibe nun das Programm und speichere es unter „morseNamen.cc“ ab. Spiele es nun deinem Tischnachbarn vor und lasse ihn den Morsecode übersetzen.

*Variante: Nimm eine LED und lies den Morsecode.*

### Erweiterung des Programms.

Viel einfacher ist es, eine Morsetaste zu benutzen und selbst zu morsen. Für dieses Beispiel habe ich das Programm erweitert. Es soll immer dann ein Ton erklingen, wenn die Taste (der Knopf) gedrückt wird.

"Wenn" bzw. "wenn nicht" heißt im Englischen "if / else"...



## IF – ELSE.

Um den Kontrollfluss eines Programmes zu kontrollieren braucht man Bedingungen, die entweder wahr (true) oder falsch (false) sein können. Je nachdem, ob eine Bedingung *true* oder *false* ist wird eine andere Aktion (oder mehrere) ausgeführt.

```

if (Bedingung)
{
    wenn Bedingung wahr ist,
    dann wird das hier gemacht.
    wenn sie nicht wahr ist,
    wird das hier nicht ausgeführt.
}
hier geht es weiter, egal ob die Bedingung wahr oder falsch war.
    
```

**Unser kleiner Versuch wird nun um einen Knopf (engl. button) erweitert.** Übrigens ist die Farbe des Knopfes egal.

```

#include "qfixSoccerBoard.h"
SoccerBoard board;

int main()
{
    board.ledOn(0);           // LED0 an
    board.waitForButton(0);  // Button0 -> Start

    while (true) //Endlosschleife
    {
        /* wenn der Knopf (Button) am Digital Eingang 0 gedrückt
           wird, dann wird der Power-Ausgang an Analog 0 freigegeben */
        if (board.digital(0))
            board.powerOn(0);

        /* ansonsten bleibt der Ausgang Analog 0 aus (off) */
        else
            board.powerOff(0);
    }
}
    
```

**Aufgabe:** Schreibe das Programm ab und probiere es mit der LED aus.



## Morsen im Eigenbau

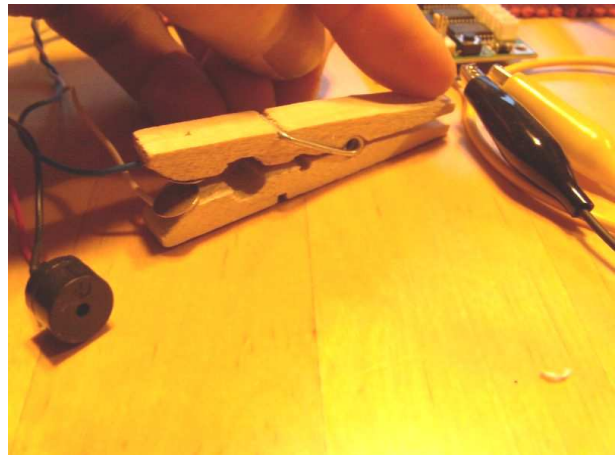
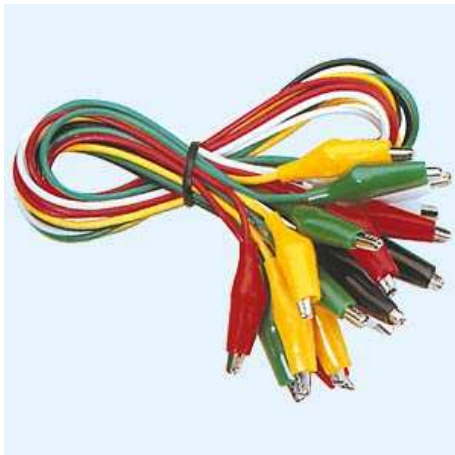
Material zum Verarbeiten gibt es genügend, man muss es sich nur suchen. Um eine Morsetaste und ein Morsegerät zu bauen braucht es nicht viel.

Es sind nur wenige Bastelarbeiten notwendig, die auch noch Spaß machen.

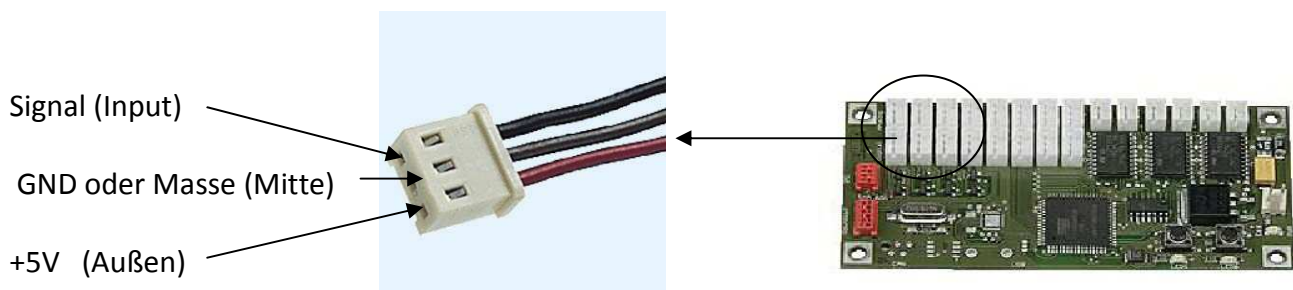
Aus einem alten Computer habe ich alle Kabel herausgezogen. Diese eignen sich sehr gut für den Roboterbau.

Ebenfalls habe ich den Beeper aus dem Computer gezogen. Es ist der Beeper, der bei älteren PC's die Fehlermeldungen - oder auch keine - beim booten angibt.

Zum Experimentieren schneidet man einfach ein Original qfix Kabel in der Mitte durch und isoliert die Enden ab. Mit Krokodilklemmen (z.B. von [www.reichelt.de](http://www.reichelt.de)) kann man sehr gute Kabel selbst bauen.



Anschlussbelegung der 3-poligen Buchsen:

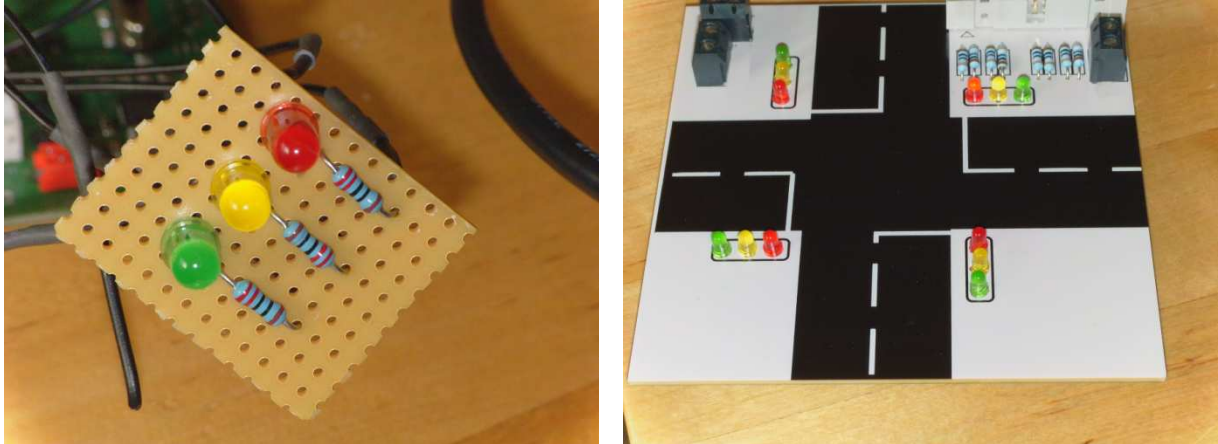


Die Stecker haben eine sogenannte „Nase“. Die Nase auf dem Foto links ist auf der Rückseite und somit jetzt nicht zu sehen.

### EINE EINFACHE AMPELANLAGE

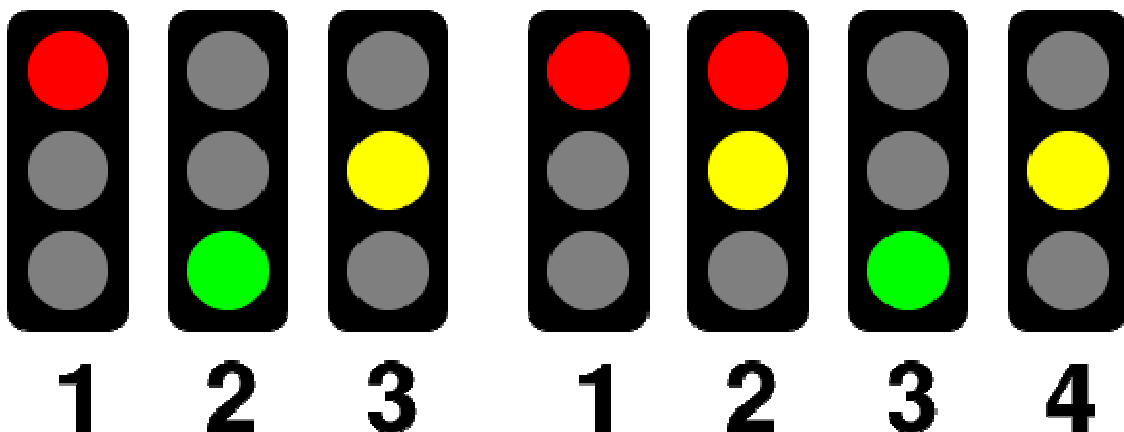
Für die Ampelschaltung habe ich 3 LEDs genommen und Sie mit 220 Ohm Widerständen vor zu hoher Spannung „geschützt“ und an + und Masse angeschlossen. Die Dioden werden an die Power Ausgänge angeschlossen. Der äußere Stift ist +5V und in der Mitte des 3er-Schuhs auf dem Board ist Masse.

Hier ein Foto meines einfachen Modells und eine Profivariante, die auch leicht zu erstellen ist:



Eine normale europäische Lichtzeichenanlage steuert den Verkehr dabei mit Hilfe der drei Signalfarben Grün, Gelb und Rot. Zur Regelung des Verkehrs werden diese Farben einzeln oder in Kombination angezeigt. Die Reihenfolge (auch *Signalfolge* oder *Farbbildfolge* genannt) solch einer Lichtzeichenanlage ist dabei immer:

- Grün: Der Verkehr ist freigegeben
- Gelb: Auf nächstes Signal warten
- Rot: Keine Einfahrerlaubnis



Aus: <http://de.wikipedia.org>

**Aufgabe:** 1. Erstelle zwei Ampelschaltungen wie auf dem Bild (1,2,3, und 1,2,3,4).  
Speichere diese Programme mit den Namen „ampel\_nl.cc“ und „ampel\_Brd.cc“ ab. (siehe Programm Seite 15)

Varianten der Standard-Lichtzeichenanlage in verschiedenen Ländern:

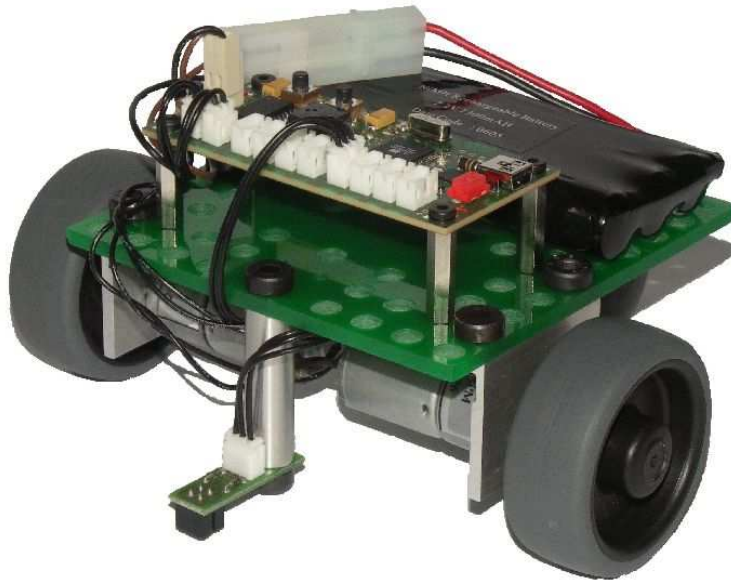
In einzelnen Ländern sind noch zusätzliche Farbkombinationen zugleich oder hintereinander möglich:

Land	Ampelphase	Bemerkung
Deutschland, Großbritannien, Österreich, Ungarn, Schweiz, Polen, Norwegen, Russland,	Rot-Gelb (gleichzeitig): Zwischen rot und grün  (Siehe Bild rechts)	<i>Achtung, gleich wird die Erlaubnis zur Fahrt gegeben</i>
Schweden	Grün-Gelb:	<i>Achtung, es wird gleich rot</i>
Belgien, Dänemark, Frankreich, Griechenland, Irland, Italien, Luxemburg, Niederlanden, Australien, Taiwan, Brasilien und den Vereinigten Staaten.	Grün folgt direkt auf Rot	----
Vereinigte Staaten, Südafrika, Taiwan	Rotes Blinklicht	<i>Stopp! Anhalten, dann langsam weiterfahren, wenn Kreuzung frei.</i>
Österreich, Litauen, der Türkei, Mexiko und Israel	Grünes Blinklicht am Ende der Grünphase	<i>Achtung, es wird gleich gelb gezeigt</i>
China	Grünes Blinklicht am Ende der Grünphase	<i>Achtung, es wird gleich rot gezeigt</i>

**Aufgabe:** 2. Erstelle die verschiedenen Ampelschaltungen der unterschiedlichen Länder. Speichere diese mit dem Namen z.B. „Ampel\_KFZ-Zeichen.cc“ ab.

3. Erstelle ein Lauflicht mit mehreren LEDs.

Für die nächsten Aufgaben brauchst du einen einfachen Roboter mit 2 Motoren an Ausgang Mo0 und Mo1.



## MOTOREN

Für die Ansteuerung der Motoren gibt es bestimmte Funktionen. Das qfix SoccerBoard verfügt über acht Motorausgänge mit den Bezeichnungen Mo0 bis Mo7.

Beispiel:

Der Motor mit dem Index 0 soll angeschaltet werden.

```
#include "qfixSoccerBoard.h"
SoccerBoard board;

int main()
{
    board.motor(0,255);
}
```

Das Programm besteht im Wesentlichen aus dem Standard-Programm. Als einzige Anweisung wird der Befehl „motor“ benutzt:

motor(Ausgang, Leistung)

Die Leistung kann negativ oder positiv sein. Werte von -255 bis 0 und bis 255 sind möglich.

*Zur Übertragung von Fahrzeugprogrammen habe ich den Roboter immer auf einen selbstgebaute Ständer gestellt.*

## PROGRAMMANWEISUNGEN

Dieses Programm hat 3 Anweisungen.

```
#include "qfixSoccerBoard.h"
SoccerBoard board;

int main()
{
    board.motor(0,255);
    board.sleep(1);
    board.motorsOff();
}
```

`board.motor(0,255);`

Diese Anweisung sagt dem Roboter den Motor, der an Motorausgang 0 angeschlossen ist mit einer Vorwärtsdrehung zu starten. Die Geschwindigkeit des Motors ist die volle Leistung: der Wert 255. Negative Werte bedeuten eine Umkehr der Drehrichtung.

`board.sleep(1);`

Diese Anweisung sagt dem Roboter, dass er für 1 Sekunde warten soll.

`board.motorsOff();`

Schaltet die Motoren aus.

### **Aufgaben:**

1. Tippe das unten stehende Programm ab.
2. Schreibe neben jede Zeile, was das Programm macht
3. Beseitige alle Fehler im Programm
4. Kompiliere und übertrage das Programm
5. Speichere das Programm unter deinem Namen ab.
6. Drucke das Programm aus.

```
// Der Roboter soll.....  
// Name und Datum  
  
int main()          // _____  
{  
    board.motor(0,122); // _____  
    board.sleep(3);    // _____  
    board.motor(0,-75); // _____  
    board.motor(1,-75); // _____  
    board.motorsOff(); // _____  
}
```

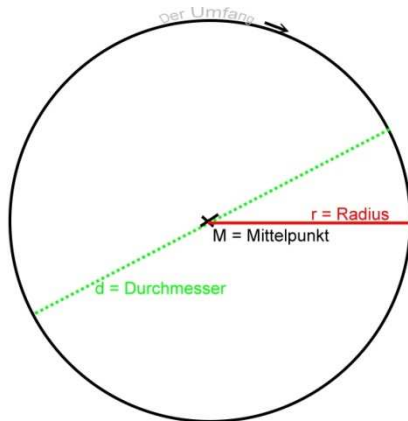
**Aufgaben:**

1. Lasse den Roboter für 3 Sekunden mit einer Leistung von 80 vorwärts fahren. Die Motoren sind an Mo0 und Mo1 angeschlossen. Ändere die Werte und probiere den Roboter fahren zu lassen.
2. Der Roboter soll sich um 90<sup>0</sup> nach rechts drehen. Speichere die Ergebnisse unter "drehung90.cc" ab.

## DURCHMESSER, RADIUS UND UMFANG

Radius, Durchmesser und Umfang. Diese drei Faktoren sind sehr wichtig in der Robotik. Soll ein Roboter besonders schnell fahren oder soll er eine Strecke mit Hindernissen (Unebenheiten auf dem Boden) meistern? Ist es ein Fahrzeug für drinnen oder draußen (Off-Road)?

Als **Radius r** (von lat. *radius*; „Strahl“) (deutsch: **Halbmesser**) bezeichnet man in der Geometrie den Abstand zwischen dem Mittelpunkt *M* eines Kreises und der Kreislinie.



Der Radius *r* entspricht dem halben Durchmesser *d*. Zum Kreisumfang *U* verhält sich der

Radius wie folgt:  $r = \frac{U}{2\pi}$

Bei dem qfix Rad ist das:



Der **Durchmesser d** (griech. *Diameter*) ist die Entfernung zwischen den Schnittpunkten eines Kreises mit einer Geraden, die dessen Mittelpunkt schneidet.

Der **Umfang U** ist die Strecke, die das Rad bei einer Umdrehung zurücklegt. Den Umfang errechnet man mit der Formel:

$d \cdot \pi = U$  // Der Durchmesser mal Pi = der Umfang

Pi ( $\pi$ ) ist eine mathematische Konstante, eine Kreiszahl mit dem unendlichen Wert:  
 3,14159 26535 89793 23846 26433 83279 50288 41971 69399 37510 58209 74944 59230  
 78164 06286 20899 86280 34825 34211 70679 ...

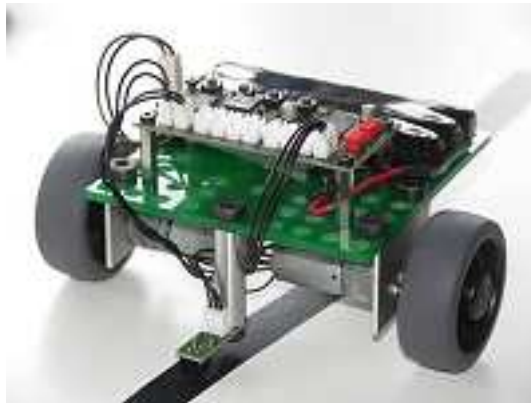
Der Umfang des Rades bzw. Reifen bedeutet abgewickelt auf die Straße den zurückgelegten Weg.

Beispiel:

Ein Roboterrad mit einem Durchmesser von 3,18 cm hat einen Umfang von 9,99 cm. Nach einer Umdrehung hat das Rad also 9,99 cm zurückgelegt. Jetzt ein paar Aufgaben zum Üben. Die richtige Radgrößenwahl kann manchmal einen Wettbewerb entscheiden!

**Aufgaben:**

1. Du hast den Roboter im Bild rechts. Der Durchmesser des Rades beträgt 5,5 cm.



Wie weit ist der Roboter gefahren, wenn sich das Rad 8 mal gedreht hat?

\_\_\_\_\_ cm

2. Du bereitest dich auf einen Wettkampf vor. Die Aufgabe lautet: Der Roboter soll eine Strecke von 10 m so schnell es geht fahren. Du hast 4 Räder mit folgenden Durchmessern zur Auswahl:

2,86 cm , 3,18 cm , 5,5 cm und 7,94 cm.

Wie viele Umdrehungen muss dein gewähltes Rad machen?

Erstelle eine Tabelle in der du die Umdrehungen vergleichen kannst.

Durchmesser	Formel	Umdrehungen
2,86 cm	_____ . $\pi$ =	_____
3,18 cm		
5,5 cm		
7,94 cm		

Wenn der Roboter eine bestimmte Streckenlänge fahren soll, er dies aber nur ungenau macht, kann es verschiedene Gründe dafür geben. Manchmal hängt es vom Antrieb deines Roboters, dem Ladezustand der Batterien oder des Akku und von der Art der Oberfläche ab, auf welcher der Roboter läuft.

Hier hilft es den Durchschnitt zu errechnen.

**Aufgabe:**

1. Nimm deinen Roboter und lasse ihn 3 mal für jeweils drei Sekunden vorwärts fahren.

Markiere mit Kreide einen Punkt auf den Reifen und zähle die exakten Umdrehungen. Was stellst du fest?

Berechne nun den Durchschnitt:

$$\frac{\text{Ergebnis 1} + \text{Ergebnis 2} + \text{Ergebnis 3}}{3}$$



d= 5,5 cm

Der Roboter legt in 3 Sekunden im Durchschnitt eine Strecke von \_\_\_\_\_ cm zurück.

## MOTOREN STARTEN MIT EINEM BUTTON

Wenn die Motoren nicht sofort starten sollen, sondern wir einen Button drücken wollen um eine Aktion zu beginnen, benötigt man eine Abfrage.

Auf dem Soccerboard sind zwei Buttons: Button 0 und Button 1.

Wir fragen nun andauernd den Button 0 ab. Wenn dieser gedrückt ist, wird Motor 0 gestartet, ansonsten wieder gestoppt.

```
#include "qfixSoccerBoard.h"

SoccerBoard board;

int main()
{
    while(true)
    {
        if (board.button(0))
            board.motor(0,255);
        else
            board.motorsOff();
    }
}
```

In der ersten Zeile wird die benötigte Bibliothek eingelesen. Dann wird ein Objekt *board* der Klasse *SoccerBoard* deklariert. In der Hauptfunktion wird dann der Button mit dem Index 0 abgefragt. Ist dieser gedrückt, wird Motor 0 auf den Wert 255 gesetzt, ist er nicht gedrückt werden sämtliche Motoren gestoppt.



**Aufgabe:**

Wenn ein externer Button zur Verfügung steht, schließe ihn an die digitalen Eingänge des Boards an. Ein Digitaleingang liefert den Wert *false* (aus) oder *true* (an oder gedrückt) zurück und kann dadurch direkt per *if* abgefragt werden.

Programmiere den Roboter mit einem Button.



**Erweiterung:**

wenn man einen alten Lüfter aus einem PC ausbaut, so kann dieser mit etwas Geschick als Motor verwendet werden. Baut euch eine kleine Klimaanlage. Wenn der Button gedrückt wird, soll der Ventilator laufen. Im gleichen Prinzip kann eine einfache Alarmanlage mit Kontakten (Kupferdraht) gebaut werden (siehe Morsetaste).

## UNTERPROGRAMM ODER SUBROUTINE

Manchmal benötigt man exakt dieselbe Abfolge von Programmschritten an mehreren Stellen eines Programms. Man kann sie nun mehrmals schreiben oder einfach als Unterprogramm von der gewünschten Stelle aus aufrufen.

Ein **Unterprogramm** oder eine **Subroutine** ist ein Teil eines Programmes, der aus gegebenenfalls mehreren anderen Programmteilen heraus gerufen werden kann und nach Abschluss der Abarbeitung jeweils in das aufrufende Programm wieder zurückkehrt.

Damit ein Programm nicht nach dem Übertragen auf das Board sofort startet, kann ein Unterprogramm „waitForStart“ geschrieben werden. Die Funktion soll im Hauptprogramm benutzt werden, um dem Benutzer anzuzeigen, welche Taster zum Starten des Roboters drücken soll.

### **Aufgabe:**

Ergänze das folgende Programm, so dass der Roboter nach Drücken des Tasters 1 für 5 Sekunden vorwärts fahren.

```
#include "qfixSoccerBoard.h"

SoccerBoard board;

void waitForStart()
{
    board.waitForButton(1);
}

int main()
{
    waitForStart();

    // Ergänze den fehlenden Code

}
```

**Aufgabe:**

Programmiere einen Roboter mit einem Start- und Stoppknopf:

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

void waitForStop()
{
  board.ledOn(0);
  board.waitForButton(0);
  board.ledOff(0);
}
```

**Erweiterung:**

Nimm zwei externe Buttons um den Roboter zu starten und zu stoppen.

## SCHLEIFEN

Schleifen dienen zur Wiederholung von Anweisungen. Die Wiederholung wird durch eine Bedingung kontrolliert, die vor oder nach jeder Wiederholung überprüft wird. Innerhalb einer Schleife sollte es eine Veränderung derart geben, dass die Bedingung irgendwann einmal unwahr wird, so dass die Schleife abbricht.

### while - Schleife:

Syntax: `while(Bedingung) Anweisung bzw. {Anweisungen}`

Die Anweisung wird ausgeführt, solange die Bedingung wahr ist. Die Bedingung wird jeweils zuerst geprüft und danach, falls die Bedingung wahr ist, die Anweisung ausgeführt. Wenn die Bedingung von vornherein unwahr ist, wird die Anweisung nicht ausgeführt.

### for - Schleife:

Syntax: `for (Initialisierung; Bedingung; Veränderung) Anweisung bzw. {Anweisungen}`

Beispiel: `for (int i=0; i<n; i++)`

Die erste Anweisung in der Klammer initialisiert die Laufvariable `i` (`int` wird nur gebraucht, wenn `i` noch keinen Typ zugewiesen bekommen hat). Die Schleife wird solange ausgeführt, wie die zweite Anweisung wahr ist. Falls die Bedingung wahr ist, werden die Anweisung und die Veränderung der Laufvariablen (`i++`) ausgeführt. Anstatt `i++` kann man auch `i=i+1` schreiben.

### do while - Schleife:

Oft kommt man aber in die Situation, bei der sich die Voraussetzungen für die Schleifen-Bedingung erst im Schleifenrumpf ergeben (z.B. durch eine Eingabe). Dafür gibt es die "annehmende" Schleife. Die im Rumpf der `do` Schleife enthaltenen Anweisungen werden solange ausgeführt, bis die Auswertung des Schleifenkopfs `false` ergibt.

Beispiel:

```
int main()
{
    int i = 10;
    do
    {
        cout << i << endl;
        i--;
    }
    while(i>0);
}
```

Im Gegensatz zur [while](#) Schleife wird der Rumpf mindestens einmal durchlaufen. Vor dem ersten Durchlauf wird also die Bedingung der Schleife noch nicht berücksichtigt.

## IF UND CASE

Wiederholung: **if-Anweisung**

Die „if“-Anweisung ähnelt der „while“-Schleife, deshalb hier nochmals die genaue Definition.

Syntax: `if (Bedingung) Anweisung bzw. {Anweisungen}`

oder `if (Bedingung) Anweisung bzw. {Anweisungen}`  
`else Anweisung bzw. {Anweisungen}`

Eine *if*-Anweisung erlaubt die Auswahl zwischen 2 Möglichkeiten. Durch Verschachtelung kann eine Auswahl zwischen mehreren Möglichkeiten getroffen werden. Dies kann bei größerer Verschachtelungstiefe allerdings sehr unübersichtlich werden.

Wenn die Bedingung zwischen den Klammern ( ) zutrifft, wird der Teil zwischen den Klammern { } durchgeführt. Andernfalls wird der Teil zwischen den Klammern { } nach dem Wort „else“ durchgeführt.

Beachte bitte, dass, die „if“-Anweisung aus zwei Teilen besteht. Die eine Hälfte wird sofort nach der „if“-Bedingung ausgeführt, wenn die Bedingung zutreffend ist, und die andere Hälfte wird nach dem „else“ ausgeführt, wenn die Bedingung falsch ist. Das Schlüsselwort „else“ und die Anweisungen nach ihm müssen aber nicht unbedingt sein. Sie können weggelassen werden, wenn es nichts gibt zu tun, falls die Bedingung falsch ist.

Will man nacheinander prüfen, ob eine Variable beispielsweise die Werte 1, 2, 3, 4 oder 5 enthält, so ist das mit der if-Anweisung etwas mühsam (versuche es selbst!)

Für diesen Zweck gibt es eine passendere Anweisung: switch/case. Sie funktioniert folgendermaßen:

```
int a = 3;

switch(a) { // Variable a soll geprüft werden
  case 1: cout << "a hat den Wert 1" << endl;
          break;
  case 2: cout << "a hat den Wert 2" << endl;
          break;
  case 3: cout << "a hat den Wert 3" << endl;
          break;
  case 4: cout << "a hat den Wert 4" << endl;
          break;
  case 5: cout << "a hat den Wert 5" << endl;
          break;
  default: cout << "a hat einen anderen Wert 1" << endl;
}

```

Achtung: Das "break" nach jeder case-Anweisung nicht vergessen!

## BEDINGUNGEN (FÜR IF UND WHILE)

Sowohl bei der Fallunterscheidung mit `if` als auch bei einer `while`-Schleife wird eine Bedingung benötigt. Es soll ja in beiden Fällen geprüft werden, ob *etwas* gilt oder ob es nicht bzw. nicht mehr gilt. Dieses *etwas* ist die sogenannte Bedingung.

In einer Bedingung vergleicht man meist Werte oder Variablen (siehe unten) miteinander. Dies kann auf unterschiedliche Weise geschehen, wobei immer ein Wert links von einem Vergleichsoperator mit einem Wert rechts von dem Operator verglichen wird. Als Operatoren stehen zur Verfügung:

<code>==</code>	ist genau gleich
<code>&lt;</code>	kleiner als
<code>&lt;=</code>	kleiner oder gleich
<code>&gt;</code>	größer als
<code>&gt;=</code>	größer oder gleich
<code>!=</code>	ungleich

### **Beispiel:**

Wenn der Roboter (bzw. das Rad) 5 weniger als Umdrehungen gemacht hat, dann fahre weiter vorwärts, ansonsten fahre rückwärts.

Mehrere Bedingungen lassen sich miteinander kombinieren, um auszudrücken, dass mehrere Bedingungen gleichzeitig erfüllt sein müssen oder entweder die eine oder die andere Bedingung erfüllt sein muss. Das Symbol `&&` bedeutet hier "und", während das Symbol `||` "oder" bedeutet.

(Den `|`-Strich erreichst du mit der Tastenkombination „Alt Gr“ und „><“).

Hier sind noch weitere Beispiele für Bedingungen, die in `if` oder `while` benutzt werden können:

```
true           immer wahr
false          nie wahr
ttt != 3       wahr, wenn ttt ungleich 3 ist
(ttt >= 5) && (ttt <= 10)
                wahr, wenn ttt zwischen 5 (inclusive) und 10 (inclusive) liegt
(aaa == 10) || (bbb == 10)
                wahr, wenn aaa oder bbb (oder beide) genau 10 sind
```

## DER INKREMENTIERUNGS- UND DEKREMENTIERUNGSOPERATOR

Unter Inkrementierung wird die Addition von 1 verstanden und unter Dekrementierung die Subtraktion von 1.

Inkrementierungsoperator: ++

Dekrementierungsoperator: --

Beide Operatoren können vor oder hinter einer Variablen stehen.

`i++` und `++i` sind Kurzformen von `i=i+1`

`i--` und `--i` sind Kurzformen von `i=i-1`

### **Achtung:**

Die beiden Operatoren können auch innerhalb eines größeren Ausdrucks verwendet werden. Dann wird auf einmal die Reihenfolge (vor oder hinter der Variable) wichtig!

Beispiel:

Die Variable `a` habe den Wert 5.

Dann hat der Ausdruck `(++a)+10` den Wert 16, weil `a` *zuerst* inkrementiert wird (also zu 6) und dann der Wert 10 addiert wird.

Der Ausdruck `(a++)+10` hat jedoch den Wert 15, weil der Wert von `a` zu 10 addiert wird (was den Wert des Gesamtausdrucks ergibt) und *danach* noch `a` um 1 erhöht wird. `a` hat also nachher ebenfalls den Wert 6.

Eine weitere Möglichkeit, `i=i+1` zu schreiben ist übrigens `i+=1`.

## VARIABLEN

Eine Variable ist mit einem kleinen Behälter vergleichbar, in dem man Dinge – vor allem Zahlen – speichern kann, die man später wieder an unterschiedlichen Stellen im Programm verwenden oder ändern möchte.

Welche "Dinge" man in der Variablen speichern können soll, muss man vor der ersten Benutzung bei der sogenannten *Deklaration* festlegen.

Beispiel:

Um eine neue Variable des Datentyps *int* zu deklarieren benutzt man folgende Anweisung:

```
int name;
```

Diese Variable ist jetzt mit "name" abrufbar.

Der Variablentyp *int* entspricht einer Ganzzahl, also einer positiven oder negativen Zahl ohne Dezimalstellen (Ganzzahlen werden auch als "ganze Zahlen" bezeichnet).

*int* (engl. integer) = einfache fortlaufende positive oder negative Ganzzahl.

In der Fachsprache wird zwischen *Deklaration* und *Definition* unterschieden. Eine Deklaration legt den Namen und den Typ einer Variablen fest, während eine Definition darüber hinaus Speicherplatz für die Variable reserviert. (Jede Definition ist also eine Deklaration, aber nicht umgekehrt).

**Beispiele:**

```
int a = 2;
int b = 3;
int ergebnis;
ergebnis = a + b;
```

Wir deklarieren zuerst die Variablen *a* und *b* vom Typ *int* und weisen ihnen die Werte 2 bzw. 3 zu. Dieser Vorgang der Wertzuweisung heißt *Initialisierung*. Eine Variable muss nicht sofort mit einem Wert initialisiert werden. Es ist auch möglich, sie zunächst nur zu definieren und ihr später einen Wert zuzuweisen, so wie es im Beispiel mit der Variablen *ergebnis* geschieht.

Im Beispiel wird der Variablen *ergebnis* der Wert *a + b* zugewiesen. Das Gleichzeichen = ist ein Zuweisungsoperator. Er weist der Variablen auf der linken Seite den Wert auf der rechten Seite zu.

Variablen können nicht nur den Typ *int* erhalten, sondern auch z.B. reelle Zahlen (Gleitkommatypen) oder alphanumerische Zeichen (A, B, C, ...) sein.

### C++ hat acht Basis-Datentypen:

- boolescher Typ (`bool`)

nimmt lediglich die booleschen Werte `true` (*wahr*) und `false` (*falsch*) an; bei Umwandlung in eine Zahl werden die beiden Wahrheitswerte als 1 bzw. 0 dargestellt

- Zeichen (`char`)

speichert typischerweise einzelne Zeichen wie z.B. 'A' oder '#', wobei es neben dem unspezifischen `char` auch die vorzeichenlose Variante `unsigned char` und die vorzeichenbehaftete `signed char` gibt; streng genommen handelt es sich bei diesem Typ um „sehr kleine Zahlen“ mit dem Wertebereich `-128...+127` bzw. `0...255`.

1. drei ganzzahlige Typen (`int`, `short` und `long`)

speichern je nach Typ ganze Zahlen in verschiedenen Wertebereichen; ohne nähere Angabe oder mit `signed` sind die Werte vorzeichenbehaftet, mit `unsigned` hingegen vorzeichenlos.

- drei Gleitkommatypen (`float`, `double` und `long double`)

dienen zur Speicherung von Realzahlen mit verschiedener Genauigkeit (z.B. `double` = doppelte Genauigkeit). Der Typ `float` reicht für die allermeisten Fälle aus.

### Beispiel:

```
int main()
{
    int fahrzeit = 2; // Variable fahrzeit mit Startwert 2

    while(true)      // weißt du weißt noch?  Endlosschleife!
    {
        board.motor(0,255); // Motor 0 anschalten
        board.sleep(fahrzeit); // warten (und solange fahren)
        board.motorsOff(); // Motoren aus
    }
}
```

**Aufgabe:** 1. Schreibe das Programm ab. Ändere die Werte. Speichere das veränderte Programm unter dem Namen „fahren.cc“ ab und probier es aus.

## VARIABLEN II

Noch einige Informationen zu den Variablen.

Neben addieren (+=) zu einer Variablen können wir einer Variable auch mit einer Zahl multiplizieren (\*=), subtrahieren (-=) und durch das Verwenden von /= teilen.

Beim Teilen zweier int-Zahlen wird das Ergebnis immer nach unten gerundet. Zur

Erinnerung: `int = ganze Zahlen`

Es können auch mehrere Variablen des gleichen Typs in einer Zeile definiert werden:

```
int ersterWert;
int zweiterWert, dritterWert;

int main()
{
    ersterWert = 10;
    zweiterWert = 20 * 5;
    dritterWert = zweiterWert;
    dritterWert /= ersterWert;
    dritterWert -= 1;
    ersterWert = 10 * (dritterWert + 3);
}
```

Beispiel:

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.

```
#include "qfixSoccerBoard.h"
SoccerBoard board;
```

```
int main()
{
    int drehzahl = 0; //Variable deklarieren
    while(true) //Endlosschleife
    {
        if (board.button(0)) //Drehzahlwert erhöhen
        {
            drehzahl = drehzahl + 10;
        }
        if (board.button(1)) //Drehzahlwert verringern
        {
            drehzahl = drehzahl - 10;
        }
        if (drehzahl > 0) //LEDs ansteuern
        {
            board.ledOn(0);
        }
    }
}
```

```

    board.ledOff(1);
}
if (drehzahl == 0) //LEDs ansteuern
{
    board.ledsOff();
}
if (drehzahl < 0) //LEDs ansteuern
{
    board.ledOff(0);
    board.ledOn(1);
}
board.motor(0, -drehzahl); //Motoren ansteuern
board.motor(1, drehzahl);
board.msleep(150); //Wartezeit
}
}

```

Tippe das Beispiel ab. Was passiert? Ändere die Werte.

#### Erweiterung 1

Im jetzigen Programm ist eine ständige Erhöhung bzw. Verringerung der Drehzahl auch über +/- 255 möglich. Dieses kann durch eine entsprechende Programmierung verhindert werden, d.h. die Variable „drehzahl“ soll nur im Bereich +/- 255 verändert werden können.

```

...
    if (board.button(0)) //Drehzahlwert erhöhen
    {
        drehzahl = drehzahl + 10;
        if (drehzahl > 255) //Korrektur der Drehzahl
        {
            drehzahl = 255;
        }
    }
    ...

```

Ergänze das Beispiel!

### Erweiterung 2

Ab einem bestimmten Drehzahlwert soll eine LED kurz aufleuchten. Der Reiz dieser Übung besteht darin, den LED 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. Die LED ist an den Analog-Eingang 2 anzuschließen.

```
board.powerOff(2); // LED vorerst ausschalten

...

if (drehzahl > 60) //Drehzahlschwelle erreicht
{
    if (variable) //Prüfen ob aktiviert werden soll
    {
        board.powerOn(2); //LED anschalten
        variable = false; //Variable zurücksetzen
    }
}
else
{
    variable = true; //Variable setzen
}
```

Bei **Variablen** und **Konstanten** handelt es sich um Bereiche im Speicher, in denen Werte abgelegt werden, damit man diese später im Programm wieder verwenden oder (bei Variablen) auch verändern kann. Jede Variable und Konstante hat einen Namen (Bezeichner, *identifizier*) und einen Typ (*type*).

## KONSTANTEN

Wenn der Roboter sich um  $90^0$  drehen soll, er das aber nicht exakt so macht, kann es verschiedene Gründe dafür geben. Manchmal hängt es vom Antrieb deines Roboters, dem Ladezustand der Batterien oder des Akku und von der Art der Oberfläche ab, auf welcher der Roboter läuft. Anstatt, dieses im Programm zu ändern, ist es einfacher, einen Namen für diese Zahl zu verwenden. Konstante Werte (= immer gleichbleibende Werte) können in C++ festgelegt werden.

Eine Konstante wird behandelt wie eine Variable, der man jedoch nur einmal einen Wert zuweisen kann. Dies geschieht direkt bei der Deklaration. Um bekannt zu geben, dass es sich um eine Konstante handelt, muss vor die Variablendefinition das Schlüsselwort `const` gestellt werden:

```
const float Pi = 3.14159;
```

Der Wert der Konstanten `Pi` kann im weiteren Programmverlauf nicht mehr verändert werden. Die Anweisung `Pi=5;` wäre also ein Fehler.

Eine Konstante im falschen Kontext zu verwenden, wird dadurch verhindert. Allerdings ist es immer noch möglich mit unbedacht platzierten, konstanten Zuweisungen schwer überblickbare Folgefehler zu erzeugen.

Programm-**Auszug** - Bsp.:

```
#include "qfixSoccerBoard.h"

SoccerBoard board;

const int HALTEPUNKT = 5000; // Konstante

int main()
{
    int zaehler = 0; //Variable deklarieren
    while(true)      //Endlosschleife
    {
        if (zaehler < HALTEPUNKT)
            board.motor(0,255);
        else
            board.motor(0,-255);

        zaehler = zaehler +1;
        if (zaehler == 2*HALTEPUNKT)
            zaehler = 0;
    }
}
```

Der Ausdruck `const int HALTEPUNKT` definiert die Konstante. Diese kann während des gesamten Programmablaufs verwendet werden.

Gründe die für die Verwendung von Konstanten sprechen:

1. Dein Programm wird übersichtlicher und lesbarer
2. Du kannst den Konstanten logische Namen geben
3. Bei größeren Programmen sind die Werte leichter zu finden, da die Konstanten zu Beginn des Programms festgelegt werden können, als irgendwo zwischen den vielen anderen Anweisungen.
4. Wenn der Wert einer Konstanten geändert werden soll, reicht es aus, dies 1x am Anfang des Programms zu tun.

**Aufgabe:** 1. Schreibe ein Programm mit dem der Roboter ein Quadrat fährt. Speichere dieses mit dem Namen „quadrat.cc“ ab.

Wenn der Roboter kein schönes Quadrat gefahren ist, verändere die Werte in den ersten beiden Zeilen.

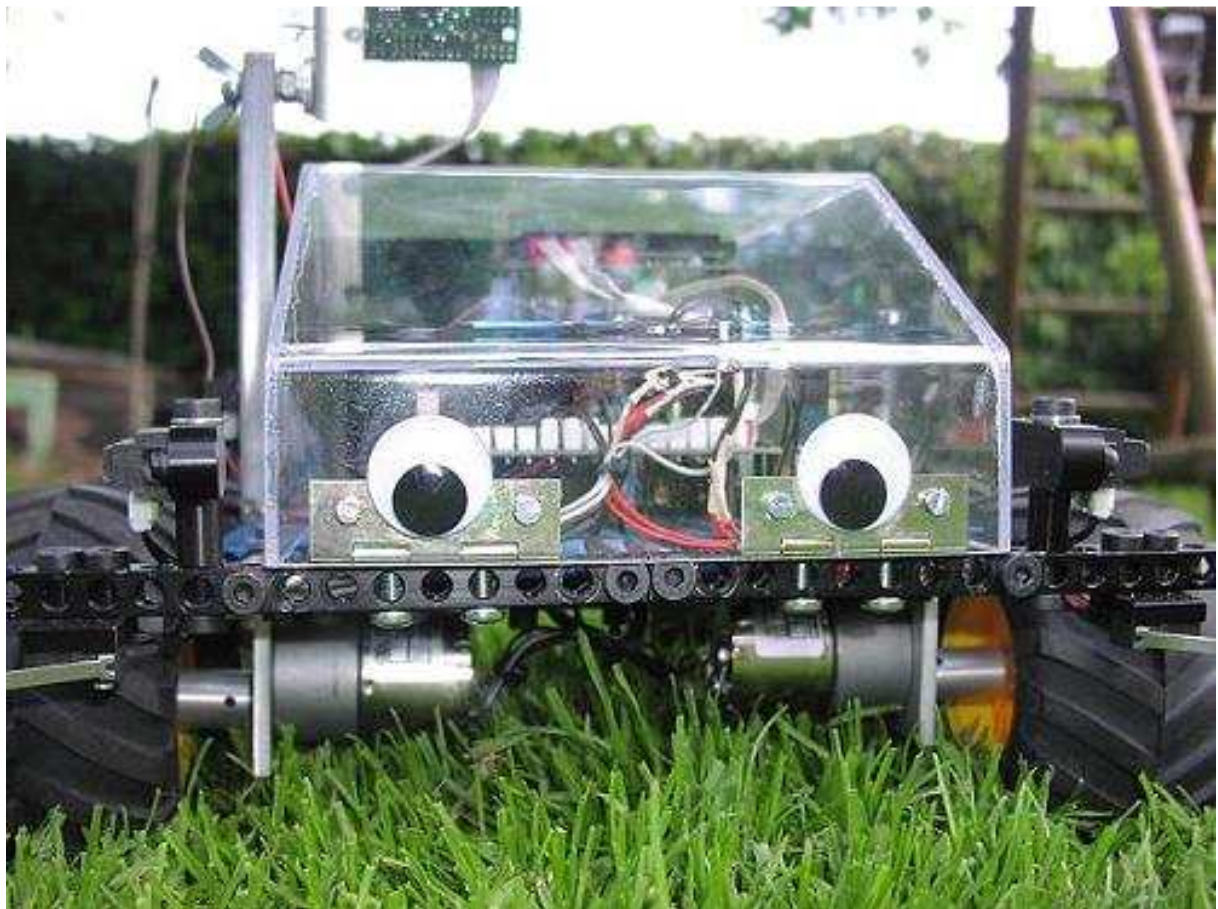
## MOTORENTEST

Wer eine Strecke genau fahren möchte, benötigt eine exakte Kontrolle der Motoren.

**Aufgabe:** 1. Schreibe ein kleines Programm, wobei der Roboter mit einer Leistung von 100 2 Sekunden vorwärts fahren, bremsen, 5 Sekunden warten und dann 2 Sekunden rückwärtsfahren soll. Nun soll der Roboter stoppen aber nicht bremsen. Mache eine Markierung, von wo du gestartet bist. Was stellst du fest? Beschreibe dein Ergebnis und erkläre den Unterschied von `motorsOff()` und `motor(0,0)`.

Für *Leistung* ist eine Zahl zwischen -255 und 255 anzugeben. Negative Werte bedeuten dabei eine Umkehr der Drehrichtung.

*Vorsicht:* Bei Werten knapp unterhalb von ca. 20 ist die Reaktion der Motoren nicht genau klar. Manche Motoren bewegen sich bereits, während andere nahezu regungslos bleiben. Das liegt an der geringen Ausgangsleistung.



## ZUFALLSZAHLEN

Was macht der Roboter eigentlich, wenn wir ihm nicht genau sagen, was er machen soll? Manchmal gibt es keinen Idealweg und der Roboter soll Dinge machen, die wir nicht genau vorhersehen können. Dafür benutzt man Zufallszahlen wie bei einem Würfelspiel. Im Gegensatz zu einem Würfel sind Zufallszahlen im Computer durch ein Programm berechnet und sehen nur wie rein zufällig aus. Das eingebaute Zufallszahlenprogramm muss erst initialisiert werden und das tut man am besten mit der aktuellen Zeit, dann bekommt man wirklich zufällige Zahlen.

Initialisierung des Zufallszahlengenerators: `srand(time(NULL));`  
 Zufallszahl abfragen: `rand();`

Der Befehl `rand()` liefert allerdings immer eine Zufallszahl zwischen 0 und 1, also z.B. 0.3648738524. Um daraus beispielsweise eine Zahl zwischen 1 und 6 zu generieren benutzt man am besten folgende Befehle:

```
int min=1;           // kleinste Zufallszahl
int max=6;           // größte Zufallszahl
int zufall;          // speichert die "gewürfelte" Zahl

srand(time(NULL));   // Zufallszahlen initialisieren
zufall=min + rand()%(max-min+1); // und würfeln ...
```

Der Operator `%` realisiert die sogenannte Modulo-Rechnung. Sie liefert den Rest einer ganzzahligen Division. Wenn Sie sich daran erinnern, wie Sie in den ersten Schulklassen dividiert haben, dann fallen Ihnen vielleicht noch Sätze ein wie: »25 geteilt durch 7 sind 3, Rest 4«. Diese Restberechnung gibt es auch unter C++. Man bezeichnet sie als die Modulo-Rechnung. Als Operatorzeichen wird das Prozentzeichen verwendet.

Beispielprogramm:

```
#include <stdlib.h>
int min=100,max=1000,zufall;

void init() {
    srand(time(NULL));
}

void do() {
    zufall=min + rand()%(max-min+1);
}

int main(...) {
    init();
    ...
    do();
    do();
    return 0;
}
```

**Aufgabe:**

1. Schreibe ein eigenes Programm und lasse den Roboter „Freestyle-tanzen“.
2. Dein Roboter soll für mindestens 15, höchstens aber für 30 Sekunden „tanzen“. Er soll sich dabei zufällig nach links und rechts drehen.

## SENSOREN

Diese Sensorplatine wird in den *Liniensensor-Sets* verwendet. Sie verwendet den Infrarot-Reflexkoppler CNY-70, der Infrarotlicht ausstrahlt und das wieder eintreffende Infrarotlicht misst. Er liefert den Messwert als analoges Signal. Die Platine kann direkt an einen Analogeingang eines qfix Controllerboards angeschlossen werden.



### Aufgabe

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

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(1)`, falls der Liniensensor an „An1“ angeschlossen wurde. Die Entscheidungsschwelle (engl. „threshold“), also der Wert, bei dem sich der Untergrund von weiß nach schwarz ändert, muss experimentell ermittelt werden und ist von der Bodenfarbe und dem Lichtverhältnis abhängig.

```
#include "qfixSoccerBoard.h"
const int THREASHOLD = 100;

SoccerBoard board;

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

Als Erweiterung der Aufgabe können zusätzliche Sensoren eingesetzt werden. Beispielsweise kann mit drei Sensoren eine "schönere" Fahrweise erzeugt werden.

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

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

## DISPLAY

Das LCD Display wird über den I2C Bus mit einem qfix Controllerboard verbunden. Es kann 4 Zeilen a 20 Zeichen ausgeben.



### **Eigenschaften**

- Controller: Atmel ATmega8 8MHz
- Auflösung 4 Zeilen mit jeweils 20 Zeichen
- Hintergrundbeleuchtung softwaremäßig anschaltbar
- Kontrastregler
- Selbstrückstellende Sicherung
- Abmessungen: 78x38 mm

Programm:

```
#include "qfixSoccerBoard.h"
#include „qfixLCD.h“

SoccerBoard board;
LCD display; // Für das LCD gibt es die Klasse LCD

int main()
{
    display.clear();
    display.print(2,5, „Hallo Welt!“);
}
```

## INFRAROT DISTANZSENSOREN

Mit einem Distanzsensor ist es möglich, Abstände genau per Infrarot messen. Solche Sensoren zählen zu den beliebtesten Sensoren für Robotik-Anwendungen, weil damit das Erkennen von Hindernissen oder die Orientierung im Raum relativ einfach wird.



Drei Ausführungen sind von qfix lieferbar:

1. Der Sharp GP2D120 Infrarot-Sensor ermöglicht eine Abstandsmessung von 4 cm bis 30 cm, die am Ausgang analog anliegt.
2. Der Sharp GP2D12 Infrarot-Sensor ermöglicht eine Abstandsmessung von 10 cm bis 80 cm, die am Ausgang analog anliegt.
3. Der Sharp GP2Y0A02YK Infrarot-Sensor ermöglicht eine Abstandsmessung von 20 cm bis 150 cm, die am Ausgang analog anliegt.

Die Besonderheit besteht in der einfachen Ansteuerung. Diese besteht nur aus Stromversorgung (5V) und einem Ausgangs-Signal. Es muss kein aufwändiges Taktsignal oder ähnliches generiert werden. Die Sensoren sind farbusabhängig, so dass ein weißes Hindernis in einem bestimmten Abstand das gleiche Signal liefert, wie ein schwarzes Hindernis im gleichen Abstand. Ein nahes Hindernis liefert die größte Ausgangsspannung, mit der Entfernung nimmt die Spannung dann ab.

### **Wie funktioniert das?**

Oft werden selbst in kleinen Roboter-Projekten mehrere dieser Sensoren genutzt, da sie nur einen äußerst engen Erfassungsbereich haben. Sie eignen sich daher sehr gut zur Vermessung einer Umgebung, insbesondere dann, wenn sie drehbar auf einem Servo montiert werden. Als Kollisionsschutz sind sie wegen des engen Winkels nur bedingt geeignet, zu diesem Zweck eignen sich zum Beispiel Ultraschallsensoren besser. Das Funktionsprinzip der Sensoren ist einfach. Der Sensor besteht aus Sender und Empfänger. Der Sender sendet einen Infrarot-Strahl aus, der vom Hindernis reflektiert wird. Je nach Entfernung trifft der reflektierte Strahl an einer unterschiedlichen Stelle auf den Empfänger (ein sog. Position Sensitive Device, PSD). Der Empfänger setzt den Auftreffpunkt in einen analogen Spannungswert um. (Quelle: *rn-wissen.de*)

Beispielprogramm:

```
#include "qfixSoccerBoard.h"

SoccerBoard board;

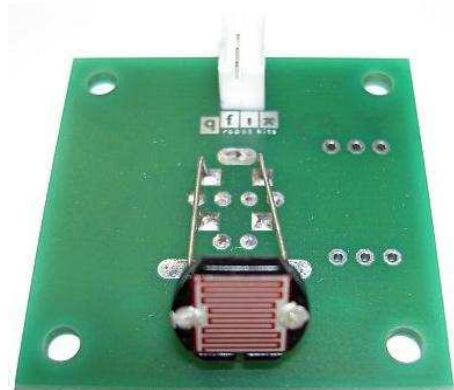
int main()
{
    while(true) {
        board.motor(0, -255 + 10 * board.analog(1));
        board.motor(1, 255 - 10 * board.analog(0));
    }
}
```

Das Programm steuert die beiden Motoren 0 und 1 in Abhängigkeit von den Entfernungswerten der Sensoren 0 und 1. Welches Verhalten entsteht hierbei?

**Aufgabe:** Schreibe nun ein Programm für eine Alarmanlage. Sobald ein Hindernis in 25 cm Nähe des Sensors kommt, soll ein lautes Signal ertönen.

## SENSOREN – LICHTSENSOR

Dieser Sensor kann einfallendes Licht messen und z.B. in einer Lichtschranke verwendet werden. Er wird an einen analogen Eingang des Controllerboards angeschlossen. Je stärker die Lichteinstrahlung, umso höher die Ausgangsspannung.



Um den Wert auszulesen benutzen wir das Display.

```
#include "qfixSoccerBoard.h"
#include "qfixLCD.h"

int value0 = 0;

SoccerBoard board;
LCD display;

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

int main()
{
    display.clear();//Display löschen

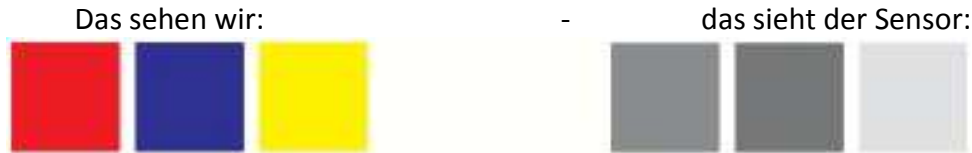
    display.print(0,0, "Anz. Analog-Eing.");

    waitForStart();

    while(true) { //Endlosschleife
        value0 = board.analog(0);          // Wert 1 lesen
        value0 = board.analog(0);          // Wert 1 2. Mal lesen
        display.print(1, 0, " Temp. Sensor:"); //Titel 2.Zeile
        display.print(1, 16, value0);      //Wert 1
        sleep(2);                           //warten
    }
}
```

Die Empfindlichkeit des Sensors ist bei frontalem Lichteinfall und bei Licht mit verhältnismäßig langer Wellenlänge (rotes und infrarotes Licht) am größten. Je grösser der Winkel des Lichteinfalls, desto schlechter wird das Licht vom Sensor detektiert.

Der Lichtsensor erkennt keine Farben, kann diese aber „erahnen“.



Wir benötigen auch die Testauflage mit der schwarzen Linie. Dieser sollte möglichst nah über den Boden angebracht werden. Das Grundprinzip der „Linie folgen“ ist, dass der Roboter versucht, auf dem Rand der schwarzen Linie zu bleiben und sich von der hellen Umgebung abwendet.

Der Roboter fährt im Uhrzeigersinn.

```
#include "qfixSoccerBoard.h"

const int THREASHOLD = 100;

SoccerBoard board;

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

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

    while(true) {
        //Endlosschleife
        if (board.analog(3) > THREASHOLD) {    //auf Linie?

            board.motor(0, -155);                //Rechtskurve
            board.motor(1, 55);
            board.motor(2, -45);
        }
    }
}
```

```
else {  
    board.motor(0, -55);           //Linkskurve  
    board.motor(1, 155);  
    board.motor(2, 45);  
}  
}  
}
```

- Aufgabe:**
1. Schreibe ein Programm wobei der Roboter einer schwarzen Linie folgt, aber gegen den Uhrzeigersinn fährt.
  2. Baue einen Roboter, der sich nur innerhalb eines mit dunklen Rand begrenzten Feldes bewegt.
  3. Schreibe ein Programm, welches verhindert dass dein Roboter über die Tischkante hinausfährt und damit vom Tisch fällt.

LICHTMESSUNG

**Aufgabe:** Schreibe das Programm von Seite 43 ab, führe das Programm aus und ermittle die Sensorwerte für die Farbe rot, blau und gelb etc. Trage die Ergebnisse in die Tabelle ein und vergleicht eure Werte. Die Werte werden auf dem Display ausgegeben.

Farbe	rot	blau	gelb	weiß	schwarz
Wert					



## SENSOREN – TASTSENSOR



Ein Tastsensor wird digital betrieben.  
Der Tastsensor besitzt nur die beiden Zustände:

- 0 = nicht gedrückt (false)
- 1 = gedrückt (true)

Er kann mit Hilfe einer If-Anweisung leicht abgefragt werden:

```
if (board.digital(0) == false) ...
```

Beispielprogramm:

```
#include "qfixSoccerBoard.h"
SoccerBoard board;

int main()
{
  while(true)
  {
    //Endlosschleife
    board.motor(0, -100); //Vorwärts
    board.motor(1, 100);

    if (board.digital(0) == false) //Kollision ?
    {
      board.motor(0, 90); //Zurück
      board.motor(1, -90);
      board.msleep(1000); //warten
    }
  }
}
```

**Aufgabe:** Schließe zwei Tastsensoren an. Der Roboter soll **so lange** geradeaus fahren, **bis** er gegen ein Hindernis stößt. Stößt er links an, soll er rechts herum fahren und umgekehrt. Versuche das Programm zu erarbeiten, indem du den Text unten durcharbeitest.

- Schreibe ein neues Programm mit dem Namen „tastsensor.cc“.
- Finde heraus, ob der Tastsensor true oder false liefert, wenn er gedrückt wird (benutze hierfür eine LED).
- Der Roboter soll nun vorwärtsfahren, **so lange bis** der Taster gedrückt wird.
- Nun soll der Roboter auf die beiden Taster unterschiedlich reagieren.

## POTENTIOMETER



Ein Potentiometer ist ein einstellbarer Widerstand. Wird es an das Controllerboard angeschlossen, so kann man beliebige Analogwerte eingeben.

Beispielprogramm:

```
#include "qfixSoccerBoard.h"
#include "qfixLCD.h"

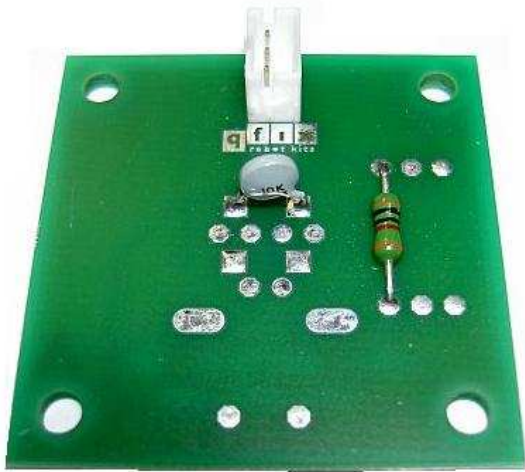
int value0 = 0;

SoccerBoard board;
LCD display;

int main()
{
    display.clear();           //Display löschen

    display.print(0,0, "Anz. Analog-Eing."); // 1.Zeile

    while(true) {             //Endlosschleife
        value0 = board.analog(0); // Wert lesen
        value0 = board.analog(0);
        display.print(1, 0, "Potentiometer:");
        display.print(1, 16, value0);
        sleep(2);
    }
}
```



## TEMPERATURSENSOR

Der Temperatursensor wird an einen Analogeingang des Controllerboards angeschlossen.

Beispielprogramm:

```
#include "qfixSoccerBoard.h"
#include "qfixLCD.h"

int value0 = 0;

SoccerBoard board;
LCD display;

int main()
{
    display.clear();                // Display löschen

    display.print(0,0, "Anz. Analog-Eing."); // Titel 1.Zeile

    while(true) {                  // Endlosschleife
        value1 = board.analog(1);   // Wert 2 lesen
        value1 = board.analog(1);
        display.print(1, 0, "Temp. Sensor:"); // Titel 1.Zeile
        display.print(1, 16, value1);

        sleep(2);
    }
}
```

### AUFGABENBLATT

1. Schreibe ein Programm das deinen Roboter 2 Sekunden vorwärts fahren lässt, dann 2 Sekunden stehen bleibt und wieder zurückfährt.
2. Der Roboter soll vorwärtsfahren und dabei „beschleunigen“ (immer schneller werden). Nach Erreichen der Spitzengeschwindigkeit soll der Roboter allmählich wieder langsamer werden.
3. Finde heraus, mit welchen Parametern der Roboter um  $90^{\circ}$ ,  $45^{\circ}$  oder  $360^{\circ}$  gedreht werden kann.
4. Lasse deinen Roboter ein geschlossenes Sechseck durchfahren. Benutze hierzu geeignete Schleifen.
5. Lasse deinen Roboter eine immer größer werdende Spirale fahren.
6. Der Roboter soll 2 Sekunden exakt geradeaus fahren und dann zufallsgesteuert entweder um  $90^{\circ}$  nach links oder rechts fahren.
7. Der Roboter soll geradeaus fahren, bis er auf einen Gegenstand trifft und dann stoppen.
8. Der Roboter soll selbstständig durch einen Raum navigieren und dabei Gegenständen ausweichen.
9. Der Roboter soll durch einen Raum fahren, ohne irgendwo gegen zu fahren. Bei einem lauten Geräusch soll er stoppen und sich um 180 Grad drehen und die Prozedur soll von vorne beginnen (Schleife).
10. Der Roboter soll einer schwarzen Linie folgen.
11. Schreibe ein Programm, dass dein Roboter nicht vom Tisch fallen kann.
12. Der Roboter soll einen Gegenstand kreisförmig Umfahren.
13. Der Roboter soll bis zu einer schwarzen Linie fahren, 1 Sekunde rückwärts fahren, sich dann um  $180^{\circ}$  drehen, bis zum Hindernis fahren und stehen bleiben. Variiere das Programm. Suche die passenden Sensoren und arbeite mit mehreren Tasks oder Unterprogrammen.

PROGRAMMBEISPIELE

**Drehung**

```
#include "qfixSoccerBoard.h"

SoccerBoard board;

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

void stopperKnopf() //Datentyp erstellen
{
    board.ledOn(1);
    if (board.button(1)) {
        board.motorsOff();
        board.ledOff(1);
        waitForStart();
    }
}

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

    while(true) { //Endlosschleife
        board.motor(0, 70); //Linksdrehung
        board.motor(1, 70);
        board.motor(2, 70);
        sleep(3); //warten
        board.motor(0, -70); //Rechtsdrehung
        board.motor(1, -70);
        board.motor(2, -70);
        sleep(3); //warten
        stopperKnopf(); //Stopknopf prüfen
    }
}
```

## Start und Stop

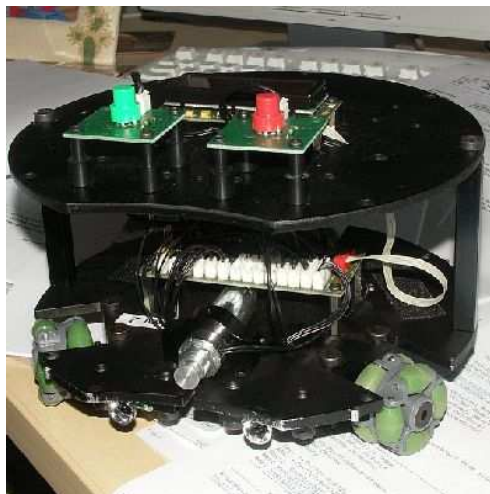
```
#include "qfixSoccerBoard.h"

SoccerBoard board;

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

void waitForStop()
{
    board.ledOn(1);
    board.waitForButton(1);
    board.ledOff(1);
}

int main()
{
    while(true) { //Endlosschleife des Programms
        waitForStart();//auf Startknopf warten, wenn gedrückt dann:
        board.motor(0, -255);//Motoren an
        board.motor(1, 255);
        waitForStop(); //auf Stopknopf warten, wenn gedrückt dann:
        board.motorsOff(); //Motoren aus
    }
}
```



## LED Lauflicht

```
#include "qfixSoccerBoard.h"

int delay = 1000;

SoccerBoard board;

int main()
{
  for (int i=0; i<=7; i++) { //Power-Ausgänge löschen
    board.powerOff(i);
  }

  while(true) { //Endlosschleife

    for (int x=0; x<=7; x++) { //Lauflicht
      if (board.button(0)) { //Wartezeit bei Taster 0
        delay = 50; //50ms
      }

      if (board.button(1)) { //Wartezeit bei Taster 1
        delay = 1000; //1000ms
      }

      board.powerOn(x); //Power-Ausgang ansteuern

      board.msleep(delay); //Wartezeit

      for (int j=0; j<=7; j++) { //Power-Ausgänge löschen
        board.powerOff(j);
      }

      board.msleep(delay); //Wartezeit
    }
  }
}
```

## C++- BEFEHLE IM ÜBERBLICK

Hier siehst du eine Liste aller Statements, Bedingungen, Befehle und Ausdrücke.

Statements sind Schlüsselworte, die ein Programm anweisen etwas auszuführen.

Statement	Beschreibung
<b>while (Bedingung) Anweisung</b>	Führe <i>Anweisung</i> nie oder solange aus, wie die <i>Bedingung</i> zutrifft
<b>do Anweisung while (Bedingung)</b>	Führe <i>Anweisung</i> einmal oder solange aus, wie die <i>Bedingung</i> zutrifft
<b>until (Bedingung) Anweisung</b>	Führe <i>Anweisung</i> nie oder solange aus, bis die <i>Bedingung</i> zutrifft
<b>if (Bedingung) Anweisung 1</b>	Führe <i>Anweisung 1</i> aus, wenn die <i>Bedingung</i> zutrifft
<b>if (Bedingung) Anweisung 1 else Anweisung 2</b>	Führe <i>Anweisung 1</i> aus, wenn die <i>Bedingung</i> zutrifft, andernfalls führe <i>Anweisung 2</i> aus

Bezeichner werden für Variable, Aufgabe, Funktion und Subroutinenamen benutzt. Der erste Buchstabe eines Bezeichners muss ein Groß- oder Kleinbuchstabe oder der Unterstrich sein ('\_'). Restliche Zeichen können Buchstaben, Zahlen und ein Unterstrich sein. Einige mögliche Bezeichner sind für den Gebrauch in der NXC Sprache selbst reserviert. Diese reservierten Worte sind Anrufschlüsselwörter und können möglicherweise nicht als Bezeichner verwendet werden. Eine komplette Liste von Schlüsselwörtern:

asm auto bool break case catch char class const const\_cast continue default delete do double dynamic\_cast else enum explicit extern false float for friend goto if inline int long mutable namespace new operator private protected public register reinterpret\_cast return short signed sizeof static static\_cast struct switch template this throw true try typedef typeid typename union unsigned using virtual void volatile wchar\_t while

### Variablen

Alle Variablen C++ sind von den folgenden Arten:

Type Name	Information
<b>bool</b>	8 bit unsigned
<b>byte, unsigned char</b>	8 bit unsigned
<b>unsigned int</b>	16 bit unsigned
<b>unsigned long</b>	32 bit unsigned
<b>long</b>	32 bit signed

<b>short, int</b>	16 bit signed
<b>char</b>	8 bit signed
<b>mutex</b>	Spezielle Art benutzt für exklusiven Codezugang
<b>string</b>	Array von byte
<b>struct</b>	Benutzer definiertes structure types
<b>Arrays</b>	Arrays irgendeiner Art

„signed“ bedeutet, dass der Datentyp *char* wie der numerische Bereich -128..127 behandelt wird, das ASCII-Zeichen 255 wird also (numerisch) als -1 interpretiert; demgegenüber bedeutet „unsigned“, dass „char“ wie der Bereich 0..255 verwendet wird, das ASCII-Zeichen 255 wird also auch numerisch als 255 interpretiert.

### Anweisungen

Es gibt neun verschiedene Zuweisungsoperatoren. Der grundlegendste Operator, '=', weist einfach den Wert des Ausdrucks der Variablen zu. Die anderen Operatoren ändern den Variablen-Wert auf irgendeine andere Art wie die Tabelle zeigt.

Operator	Tätigkeit
=	Weist der Variablen einen Ausdruck zu
+=	Addiere den Ausdruck zur Variablen hinzu
-=	Subtrahiere den Ausdruck zur Variablen
*=	Multipliziere die Variable durch den Ausdruck
/=	Dividiert die Variable durch den Ausdruck
%=	Weist einer Variablen den Modulo-Wert zu
&=	Bitweises UND Ausdruck in die Variable
=	Bitweises ODER Ausdruck in die Variable
^=	Nur Bitweises ODER in die Variable
=	Setzt die Variable zu einem absoluten Wert
+-=	Setzt die Variable zum Zeichen (- 1, +1.0) des Ausdrucks

## Bedingungen

Bedingung	Bedeutung
true	Immer wahr
false	Immer falsch
Ausdruck 1 == Ausdruck 2	Prüfe, ob die <i>Ausdrücke</i> gleich sind
Ausdruck 1 != Ausdruck 2	Prüfe, ob die <i>Ausdrücke</i> nicht gleich sind
Ausdruck 1 < Ausdruck 2	Prüfe ob <i>Ausdruck 1</i> kleiner als <i>Ausdruck 2</i> ist
Ausdruck 1 <= Ausdruck 2	Prüfe ob <i>Ausdruck 1</i> kleiner oder gleich <i>Ausdruck 2</i> ist
Ausdruck 1 > Ausdruck 2	Prüfe ob <i>Ausdruck 1</i> größer als <i>Ausdruck 2</i> ist
Ausdruck 1 >= Ausdruck 2	Prüfe ob <i>Ausdruck 1</i> größer oder gleich <i>Ausdruck 2</i> ist
Bedingung 1 && Bedingung 2	Logisches UND der beiden <i>Bedingungen</i> Nur dann wahr, wenn beide <i>Bedingungen</i> wahr sind
Bedingung 1    Bedingung 2	Logisches ODER der beiden <i>Bedingungen</i> Wahr, wenn eine der beiden <i>Bedingungen</i> wahr ist

## Mathematische Operatoren

Operator	Beschreibung	Bezug	Einschränkung	Beispiel
<b>abs()</b>	Absoluter Wert	n/a		abs(x)
<b>sign()</b>	Vorzeichen	n/a		sign(x)
<b>++</b>	Addition	left	Nur Variablen	x++ oder ++x
<b>--</b>	Subtraktion	left	Nur Variablen	x-- oder --x
<b>-</b>	Vorzeichenwechsel	rechts	Nur Konstanten	-x
<b>~</b>	Bitweise Verneinung	rechts		~123
<b>!</b>	Logische Negation			!x
<b>*</b>	Multiplikation	links		x * y
<b>/</b>	Division	links		x / y
<b>%</b>	Modulo (ganzzahliger Rest)	links		123 % 4
<b>+</b>	Addition	links		x + y
<b>-</b>	Subtraktion	links		x - y
<b>&lt;&lt;</b>	Left shift	links		x << 4
<b>&gt;&gt;</b>	Right shift	links		x >> 4
<b>&lt;, &gt;, &lt;=, &gt;=</b>	Relationale Operatoren	links		x<y

---

==, !=	gleich, nicht gleich	links	$x == 1$
&	Bitweises UND	links	$x \& y$
^	Bitweises XOR	links	$x \wedge y$
	Bitweise ODER	links	$x   y$
&&	Logisches UND	links	$x \&\& y$
	logisches ODER	links	$x    y$
?:	bedingter Wert	n/a	$x == 1 ? y : z$

---

# IDEENSAMMLUNG

## EIN AUTOMATISCHER FEUERMELDER

Früher wohnten Wächter auf hohen Türmen in den Städten, die ein entstehendes Feuer durch ein Hornsignal zu melden hatten.

Feuermelder in unserer Zeit sehen anders aus.

Öffentliche Feuermelder sind der Öffentlichkeit zugänglich und werden von Hand ausgelöst.

Beim Betätigen des Feuermelders wird sofort Alarm gegeben; zugleich läuft eine Kontaktscheibe ab und gibt in der Meldezentrale den Standort an, von dem die Meldung kommt. Dennoch kommt es vor, dass die Feuerwehr zu spät an die Brandstelle gelangt.

Deshalb hat man automatische Feuermelder gebaut, die bei jeder gefährlichen Temperaturerhöhung in einem Raum, auf einem Schiff, in einem Bergwerk etc. selbständig Alarm geben, ja sogar von sich aus die Brandbekämpfung einleiten.



### **Aufgaben:**

1. Konstruiere einen Feuermelder mit dem qfix Controllerboard.
2. Schreibe ein Programm, wobei bei einer Temperaturerhöhung ein Signalton und eine Lampe leuchten sollen (Alarm).

**Tipp:** Als Feuerstelle kannst du ein Teelicht nehmen.

### **Für Fortgeschrittene:**

3. Entwerfe eine Löschvorrichtung und versuche die Kerze zu löschen (Ventilator, oder Hydraulikpumpe – Wasser)

## WIR BAUEN EIN FERNTHERMOMETER

Es gibt in der Technik Räume, deren Temperatur von der Ferne her überwacht werden muss. Entweder ist es in ihnen so warm, dass man nicht in unmittelbarer Nähe mit einem Thermometer messen kann, oder die Anwesenheit von Menschen ist aus anderen Gründen unmöglich. In solchen Fällen braucht man ein Fernthermometer.

Versuchsdurchführung:

Umwickelt das Thermometer des NXT mit Kupferdraht, so dass ihr so eine Art „Fühler“ habt. Stelle nun eine Kerze im bestimmten Abstand in die Nähe des Drahtes.

### **Aufgaben:**

Wie ist die Wärmeentwicklung im Abstand zum Draht? Lege eine Tabelle an und übertrage die Werte. Fasse Deine Beobachtungen zusammen.

## KÜHLSCHRANKALARM UND ALARMANLAGE

(wenn Licht angeht, dann Sound) oder eine Figur wackelt.

Inspiziert von [www.amazon.de](http://www.amazon.de): *Kühlschrankschwein oder Diätschwein*

Dieses niedliche Kühlschrankschwein kann Ihnen helfen die Kühlschrantür, doch besser wieder zu schließen und nicht mehr zu naschen! Das Diät-Schweinchen reagiert auf den Lichtsensor im Kühlschrank, wenn die Tür sich öffnet und das Licht angeht, dann grunzt es ganz niedlich.....



**Mit dem gleichen Prinzip kann man eine Alarmanlage mit dem Distanzsensoren bauen.**

## DAS ÜBERWACHTE WOHNHAUS

Um mit qfix Boards und Sensoren ein ganzes Haus zu regeln, benötigt man ein schönes Spielhaus. Das Playmobil-Haus eignet sich darum, da es sehr groß und von einer Seite offen ist. Dieses Haus kann nun mit verschiedenen Sensoren und Motoren „verbessert“ werden.

- die Temperatursteuerung soll im Gebäude optimiert wird. Beispielsweise werden Wohnräume hauptsächlich zwischen 7 und 10, bzw. 17 und 23 Uhr geheizt
- Die lichtsensorgesteuerten Rollläden bewirken eine optimale Sonneneinstrahlung
- Integriert in diese intelligente Steuerung sind selbstverständlich Wassersensoren, Regenmelder, Windmesser, Bewegungsmelder sowie andere Sensoren für eine Alarmanlage
- Die Beleuchtung kann an – oder abgeschaltet werden
- Türklingel
- Das Sonnensegel soll ausfahren bei Sonne
- Die Fenster schließen bei Regen
- Automatischer Türöffner
- Zeitlichtsteuerung
- Erweiterungen selbst bauen

Für den Wintergarten kann eine Lüftung mit Temperatur eingebaut werden. Es gibt viele Möglichkeiten!



Foto: **PLAYMOBIL**® Neues Wohnhaus Traumhaus **Haus** 4279 und **PLAYMOBIL**® 4281 - **PLAYMOBIL**® - Wintergarten mit Sonnenterrasse

Und vieles mehr.

Projektbeschreibung im Anhang S. 86





## AUTOSELBSTFAHRER

### **Roboterauto "Leonie" rollt durch Braunschweig**

Zum ersten Mal hat ein vollautomatisch gesteuertes Fahrzeug eine Strecke im realen Straßenverkehr zurückgelegt. Ohne Fahrer und wie von Geisterhand gelenkt, fuhr das

Forschungsauto "Leonie" sicher auf einer zweispurigen Straße.

Auf dem dichtbefahrenen Stadtring von Braunschweig sollte "Leonie", ein umgebauter VW Passat, seine Qualitäten beweisen. "Das ist etwas ganz anderes als auf den abgesperrten Versuchsgeländen, wo solche Tests bislang stattfanden", erklärte Markus Maurer, der Leiter des Instituts für Regelungstechnik an der Technischen Universität Braunschweig. Die Hochschule und ihr Niedersächsisches Forschungszentrum Fahrzeugtechnik haben das Projekt "Stadtpilot" entwickelt, zu dem auch "Leonie" gehört.

Äußerlich fällt das Versuchsauto vor allem durch einen auf dem Dach montierten Laserscanner auf. Er sieht der Kamera des umstrittenen Internet-Portals Google Street View nicht unähnlich und übermittelt Daten über Abstand und Bewegungen anderer Fahrzeuge sowie möglicher Hindernisse in Echtzeit an die Bordelektronik. Dadurch kann Leonie ihr Tempo blitzschnell an die jeweilige Verkehrssituation anpassen.

(Quelle: [www.stern.de](http://www.stern.de))

Ein Modellauto umbauen und mit Sensoren ausstatten.

## SIEGESZUG DER KÜNSTLICHEN INTELLIGENZ

Seine technologische Vision beschreibt Moravec in diesem Buch wie folgt:

„Ich sehe diese Maschinen als unsere Nachkommen. Im Augenblick glaubt man das kaum, weil sie eben nur so intelligent sind wie Insekten. Aber mit der Zeit werden wir das große Potential erkennen, das in ihnen steckt. Und wir werden unsere neuen Roboterkinder gern haben, denn sie werden angenehmer sein als Menschen. Man muss ja nicht all die negativen menschlichen Eigenschaften, die es seit der Steinzeit gibt, in diese Maschinen einbauen. Damals waren diese Eigenschaften für den Menschen wichtig. Aggressionen etwa brauchte er, um zu überleben. Heute, in unseren großen zivilisierten Gesellschaften machen diese Instinkte keinen Sinn mehr. Diese Dinge kann man einfach weglassen - genauso wie den Wesenszug der Menschen, dass sie ihr Leben auf Kosten anderer sichern wollen. Ein Roboter hat das alles nicht. Er ist ein reines Geschöpf unserer Kultur und sein Erfolg hängt davon ab, wie diese Kultur sich weiterentwickelt. Er wird sich also sehr viel besser eingliedern als viele Menschen das tun. Wir werden sie also mögen und wir werden uns mit ihnen identifizieren. Wir werden sie als Kinder annehmen - als Kinder, die nicht durch unsere Gene geprägt sind, sondern die wir mit unseren Händen und mit unserem Geist gebaut haben.<sup>[1]</sup>“

<sup>1</sup> aus: „Computer übernehmen die Macht: Vom Siegeszug der künstlichen Intelligenz“, Hamburg 1999, Seite 136

### **Aufgaben:**

1. Diskutiert das Zukunftsbild Moravecs und sammelt Pro- und Contra Argumente zur Robotik.
2. Sucht im Internet nach Beispielen zu „Insekten Robotern“, Humanoiden etc.
3. Wer ist Hans Moravec?

## Roboter – Feind oder Freund?



### Text 1

#### Das Jahr 2050: Endstation Pflegeroboter-Heim

Deutschland im Jahr 2050: 40 Prozent der Einwohner sind 60 Jahre und älter - 17 Pflegebedürftige werden von einem Pfleger betreut. Wohl dem, der sich dann einen CARE-O-BOT oder einen RI-MAN leisten kann.

„Mädchen, die heute auf die Welt kommen, haben eine 50-prozentige Chance, einmal 100 zu werden“, so Frank Schirmacher, Autor von „Das Methusalem-Komplott“. Deutschland vergreist: Niemals zuvor gab es hierzulande mehr ältere als jüngere Menschen. Die Kombination aus steigender Lebenserwartung und geringer Geburtenrate ist bereits heute eine enorme Herausforderung für unsere Sozialsysteme. Und wenn in den kommenden Jahren weiterhin nur 1,36 Kinder je Frau geboren werden, so wird sich die Situation in naher Zukunft weiter verschärfen. Für 2050 hat das Statistische Bundesamt folgende Prognose berechnet: Deutschland wird 68,5 Millionen Einwohner haben, heute sind es 82,4 Millionen. Fast 40 Prozent werden über 60 sein, ihre Zahl steigt um 6,6 Millionen. Nur jeder siebte Deutsche wird noch unter 20 sein, heute ist es jeder fünfte. Auf 100 Deutsche zwischen 20 und 60 kommen 85 Senioren. Das Durchschnittsalter liegt bei über 50 Jahren....

Forschung und Wirtschaft haben diese Entwicklung längst erkannt: Pflege und Rehabilitation bilden in der Robotik seit vielen Jahren einen sich rasch entwickelnden Forschungsbereich. Weltweit arbeiten Institute und Firmen unter Hochdruck an der Entwicklung von künstlichen Humanoiden. In den USA sind bereits Krankenhausroboter auf dem Markt, die Röntgenbilder von einer Abteilung in die andere bringen. An der Carnegie Mellon University in Pittsburgh, Pennsylvania, wird derzeit an einem "Nursebot" gearbeitet, einem persönlichen Roboterassistenten für gebrechliche und kranke Menschen. *Von Oliver Creutz*

Text2:

Roboter kann fehlerlos Violine spielen

Da staunt das Publikum: Toyota hat einen Roboter entwickelt, der auf einer Violine die fehlerfreie Version des Stücks „Pomp and Circumstance“ von Edgar Elgar spielen kann. Doch damit nicht genug - der Roboter kann auch bei der Hausarbeit oder in der Pflege von Menschen eingesetzt werden.

Bei den neu vorgestellten Humanoiden handelt es sich um einen Violine spielenden Roboter, der sogar wie ein echter Mensch ein Vibrato hinbekommt. Der 1,50 Meter große Roboter läuft wie seine menschlichen Ebenbilder auf zwei Beinen, wiegt 56 Kilogramm und ist mit insgesamt 17 Gelenken in seinen Händen und Armen versehen, um auch sehr feinfühlig Bewegungen ähnlich denen eines Menschen auszuführen. Der Roboter sei so konzipiert, dass er Mitarbeitern im Pflegedienst zur Hand gehen und im Haushalt helfen kann, hieß es.

Toyotas andere neue Roboterschöpfung, der einsitzige „Mobi-ro“, eine Abkürzung von Mobilitätsroboter, kann Menschen angetrieben von einer Batterie mit einer Geschwindigkeit von 20 Kilometern in der Stunde über kurze Distanzen hinweg transportieren. Dabei ist er in der Lage, selbstständig Hindernissen auszuweichen und kann auch per Fernbedienung gesteuert werden.

„Wir nutzen Industrieroboter seit den 80er Jahren, sie wurden mit der Zeit perfektioniert und können an mehreren Automodellen mitarbeiten oder unterschiedliche Aufgaben erledigen“, sagte Toyota-Chef Katsuaki Watanabe. „Jetzt wollen wir die Entwicklung von gesellschaftlich nützlichen Robotern beschleunigen und dabei auf unsere Kenntnisse im Bereich der Automobile zurückgreifen“, sagte Watanabe.

Vor drei Jahren hatte Toyota seinen ersten menschenähnlichen Roboter vorgestellt, der Trompete spielen konnte. Die Konkurrenten Honda und Sony sind dem Autobauer aber um Längen voraus. Japan gilt als großer potenzieller Markt für Roboter, da der Anteil der alten Menschen an der Bevölkerung rapide zunimmt. Mehr als 30.000 Japaner sind über 100 Jahre alt.

*Von Oliver Creutz*

(DPA/AFP/OC) nach: <http://www.welt.de> (Welt online)

Aufgaben:

B1. Beschreibe das Bild und werte es aus.

B2. Sucht euch einen Text aus und fasst das Ergebnis des Textes mit eigenen Worten zusammen.

B3. Diskutiert, ob man Menschen durch Roboter ersetzen kann oder soll. Welche Gefahren oder Chancen entstehen durch den Einsatz von Robotern?

## Schlagzeilen

Roboter bedient und hilft im Altenheim

# Menschenähnlich

Holländer gründen Heim für Roboter

Roboter werden aus Erfahrung klug

Roboteranzug macht gelähmte Menschen mobil

Menschelnde Roboter werden besser behandelt

### Aufgaben:

- i. Bildet Kleingruppen. Besprecht die Zeitungsschlagzeilen in der Kleingruppe und versucht sie einer Branche, einem Unternehmen zuzuordnen.
- ii. Sucht euch eine Schlagzeile aus. Schreibt einen Text zu der Schlagzeile.  
Hilfe: Wie ist das Verhältnis Roboter – Mensch? Ist der Mensch überflüssig?
- iii. Präsentiert eure Ergebnisse.

## Künstliche Intelligenz im Dienste des Menschen

### Beruf Wachmann

- Um zehn Uhr abends dreht sich der Schlüssel am Haupttor der Universität ein letztes Mal im Schloss, dann nimmt der Pförtner schnaufend seine Jacke und macht sich auf den Heimweg. Nur die schnauzbärtigen Herren vom Wachschutz flanieren noch durch die notdürftig beleuchteten Fakultätsflure und Dekanate. Suchend wandert die Stabtaschenlampe über Schreibtische und Regale. Hier summt noch eine übereifrige Kaffeemaschine, dort schlummert ein dezent alkoholisierter Landstreicher über mitgebrachten Plastiktüten. Ansonsten keine weiteren Vorkommnisse, die ganze Nacht lang nicht. Und genau genommen die ganze nächste Woche, den nächsten Monat, das nächste Jahr auch nicht. Wenig aufregend ist das alles. Und so sitzen die ordnungsgemäß bemützten Wachmänner im Dienstzimmer und trinken Kaffee. Es knattert das Funkgerät, im Radio knarzt Rod Stewart. In zwei Stunden werden sie wieder zum Rundgang durch die Universität aufbrechen. Durch die Biologielabore, wo Mäuse hinter Labortüren piepsen. Durch das Theologiedekanat, wo immer alles so schön aufgeräumt ist. In zwei Stunden. Doch jetzt sitzen sie da, starren auf die Schreibtischlampe und träumen vom großen Auftritt. Was könnte nicht alles passieren: ein Flächenbrand mit Funkenflug, eine Geiselnahme im Audimax... Plötzlich schreckt der Wachmann auf. Ein Griff zur Taschenlampe, ein prüfender Blick zur Uhr. Der Kollege ist eingeknickt, sein Kinn sinkt langsam auf die Brust. Vier Uhr morgens, bald Zeit für den Rundgang. Gemächlich schenkt er den Kaffee ein. Noch zwei Stunden bis Dienstende. *Von Philipp Köster, aus: <http://www.spiegel.de> (spiegel online)*
- Wachleute arbeiten meistens mit Detekteien zusammen, doch sie können auch externe Aufträge übernehmen. Sie arbeiten dann direkt für die Unternehmen oder Betriebe. Eine offizielle Ausbildung gibt es nicht. Viele der angestellten Wachleute machen nach abgeschlossener Ausbildung und Arbeitserfahrung im Betrieb Karriere. Sie können an Weiterbildungskurse und Wochenendseminare teilnehmen. Außerdem können Wachleute weitere Qualifikationen anstreben und Prüfungen ablegen, die ihnen vom Vorteil sind und ihrer Karriere auf die Sprünge helfen könnten. Es gibt unterschiedliche Ausbildungsgänge, die den Lehrling auf den Beruf zum Wachmann vorbereiten. Einige der angebotenen Ausbildungen sind in Blockform organisiert und dauert zwischen einem und drei Jahren. Früher wurde dieser Beruf ohne eine qualifizierte Ausbildung ausgeübt, wobei es auch heute noch manchmal so ist. Außerdem wurde dieser Beruf auch als Zweitberuf ausgeübt. Am Ende jedes Lehrganges musste man eine schriftliche und eine mündliche Abschlussprüfung absolvieren. Viele der Wachschutzleute sind ehemalige Polizisten, Sicherheitsleute oder Werkschutzkräfte, die auf Grund gesundheitlicher Probleme ihren Beruf nicht mehr ausüben können. Durch bestimmte Fortbildungskurse können sie die Qualifikation zum Wachmann erreichen.
- BERLIN - Wachmann Mosro ist wahrscheinlich der ungewöhnlichste Mitarbeiter, den ein Sicherheitsdienst engagieren kann: Er misst gerade mal 118 Zentimeter, wiegt 25 Kilogramm und besteht vorwiegend aus einer metallfarbenen Plastikhaut, Elektrochips und Kabeln. Sein Körper erinnert an eine schmal geschnittene, rechteckige Tonne, statt

eines Kopfes trägt er eine knallrote Signalleuchte auf dem Rumpf. Wachmann Mosro ist ein Roboter. Der Kasten auf Rädern funktioniert vollelektronisch, arbeitet maximal 14 bis 16 Stunden am Stück, bevor er sich für einige Stunden zurückzieht und neue Lebenskraft aus dem 220-Volt- Stromnetz saugt. Sein Revier: Lagerhallen, Rechenzentren, Kunstaustellungen – überall dort, wo mobiles Wachpersonal gebraucht wird.

Bevor Mosro seinen Dienst aufnehmen kann, wird seine Route vom Wachschatz programmiert. Erkennt er mit seinem eingebauten Infrarotsensor einen Eindringling, gibt der Roboter per Funk Alarm – und wartet auf Verstärkung aus der Leitzentrale. Eingesetzt wurde Mosro (das steht für mobiler Sicherheitsroboter) bereits auf Großveranstaltungen, zuletzt bei der Fußball-Weltmeisterschaft im vergangenen Jahr. Im Olympiastadion streifte er nachts zwischen den VIP-Shuttles in der Garage umher, um die Fahrzeuge vor Manipulationen zu schützen.

Entworfen und programmiert werden die Sicherheitsroboter von der Firma Robowatch Technologies GmbH in Berlin-Pankow. Dann setzen Fachleute aus Jena und Korea die Elektroteile zusammen. Jens Hanke gründete Robowatch im Oktober 2000. Heute arbeiten 45 Mitarbeiter in Pankow – Entwickler, Ingenieure, Informatiker, Programmierer und Designer. Geschäftsführer Hanke rechnet für das kommende Jahr erstmals mit einem Umsatz im zweistelligen Millionenbereich. Mehr will er nicht verraten. *(Auszug aus dem gedruckten Tagesspiegel vom 24.11.2007)*

Lest alle Texte durch. Unterstreicht alle unbekanntenen Wörter und versucht diese Hilfe des Internets zu klären.

- Aufgaben:
- Was ist die Aufgabe eines Wachmanns und wie wird man Wachmann?
  - Warum werden heute immer mehr Überwachungsroboter eingesetzt?
  - Nenne Vor- und Nachteile von Wachrobotern
  - Sucht im Internet nach einem Beispiel für einen Wachroboter.

## LÖSUNGEN

### Aufgaben

1.

Pro: Verbesserung von Produktion und Produktionsabläufen, Steigerung der Lebensqualität, schnellere und längere Arbeitsweise

Contra: Arbeitsplätze werden verringert, menschliche Arbeitskraft wird überflüssig, könnten ohne menschliche Kontrolle Entscheidungen treffen...

2.

Humanoide: <http://www.dribblers.de> , [www.graupner-robotics.de](http://www.graupner-robotics.de)

Insekten: <http://www.sri.com/robotics/climbing.html> , <http://www.lynxmotion.com> ,  
<http://www.hexapodrobot.com>

3. **Hans Peter Moravec** (\* 30. November 1948 in Kautzen, Österreich) ist Professor für Robotik an der Carnegie Mellon University in Pittsburgh, Pennsylvania, USA. Der kanadische Staatsbürger ist seit den 1970er Jahren Vorreiter in der Roboterentwicklung, mit sehr vielen Veröffentlichungen zur künstlichen Intelligenz, zum Transhumanismus und zur Futurologie. Moravec promovierte 1980 an der Stanford University zum Doctor of Philosophy durch die Entwicklung eines ferngesteuerten, mit einer Fernsehkamera ausgerüsteten Roboters der durch einen Computer gesteuert wurde.

2003 war er Mitbegründer der *SEEGRID Corporation*, einem Roboter- Unternehmen, das sich die Entwicklung von völlig autonomen Robotern zum Ziel gesetzt hat. (Quelle: Wikipedia)

B1 Ein Roboter der Geige spielt. Musik ist Gefühl, Emotionen. Roboter drücken Gefühle aus, werden dem Menschen ähnlich gemacht....

B3 Chance: Roboter im Gesundheitswesen, „Roboterpatient“

(<http://www.ehrensfn.de/linktipps/roboter-patient>) , Wachdienst, Haushalt, Raumfahrt...

Gefahr: keine menschliche Zuwendung (Altenpflege), Kampfroboter ...

i. Verschieden Thematiken werden angedeutet: Roboter übernehmen Aufgaben von Menschen im Altenheim, helfen Kranken, werden wie Menschen behandelt (Heim für Roboter). Es geht um Ethik. Roboter „menschenähnlich“.

### a) Was ist die Aufgabe eines Wachmanns und wie wird man Wachmann?

Wachleute im Objektschutz üben die so genannte "Revierkontrolle" aus. Das heißt, dass die Sicherheitsmitarbeiter an den Objekten der verschiedenen Kunden des Wachschutzunternehmens regelmäßig vor Ort sind. Die Aufgabe der Mitarbeiter ist es, die verschiedenen gewerblichen, öffentlichen und privaten Objekte regelmäßig zu kontrollieren. Das bedeutet nicht einfach nur, dass man dort vorbeifährt. Vielmehr machen die Sicherheitsmitarbeiter Rundgänge bei den zu überwachenden Objekten und kontrollieren die zum Beispiel die Schließung der Türen, die Fenster und ob sonst alles in Ordnung ist. Beispielsweise kann es auch zu den Aufgaben gehören, die Tätigkeit der Angehörigen von Fremdfirmen und des Reinigungspersonals, insbesondere wenn diese in

sicherheitsrelevanten Abteilungen tätig sind, zu überwachen. Im Alarmfall ist es wichtig, schnell und zuverlässig zu reagieren und zu entscheiden, was zu tun ist.

Um diese Tätigkeit ausüben zu können, ist üblicherweise eine Aus- oder Weiterbildung im Wach- und Sicherheitsgewerbe erforderlich.

**b) Warum werden heute immer mehr Überwachungsroboter eingesetzt?**

Die Roboter arbeiten länger, ohne Pause und werden nicht krank. Er braucht keinen Lohn – nur Einmalkosten für die Anschaffung – evtl. Wartungskosten. Er braucht keinen Urlaub und ist nicht manipulierbar. Er zeigt keine Angst.

**c) Nenne Vor- und Nachteile von Wachrobotern**

**Nachteil:** Arbeitsplätze gehen verloren, fehlende Menschlichkeit, Beweglicher bei Verfolgung

**Vorteil:** Überwachungsroboter sind sinnvoll, wenn die Überwachung eines Objekts durch Menschen gefährlich, unzumutbar oder aus Kostengründen nicht möglich ist.

Überwachungsroboter sind schlichte Konstruktionen, die sich in einer einfach strukturierten Welt zurechtfinden. Schon eine Büroeingang kann sich als zu schwierige Hürde erweisen.

**d) Beispiel: MOSRO Der mobile Sicherheitsroboter**

Der mobile und unabhängige Überwachungsroboter MOSRO eignet sich als sinnvolle Ergänzung zu qualifizierten Überwachungsdiensten und stationären Alarmanlagen. Er garantiert eine Dauerüberwachung ohne Ermüdungserscheinungen oder Konzentrationsschwächen.

Mosro wird nach Kundenwunsch ausgestattet (bis zu 240 Sensoren), individuell programmiert und exakt auf seinen Einsatzort und Aufgabenbereich eingestellt. Über neuronale Netze navigiert er sich mit ca. 4 km/h selbstständig durch Hallen, Fabriken, Geschäftsräume oder Einkaufszentren. Ohne Pause – bis zu 18 Stunden lang.

Der Wachroboter kann sehen, hören, riechen, fühlen und sogar sprechen.

In seinem 25 kg leichten, wendigen Korpus bündelt er die modernsten Technologien der gesamten Branche. Ausgestattet mit Radarsensoren, Bewegungsscannern, Ultraschallsensoren und Kamera ist er in der Lage, selbst bei völliger Dunkelheit und durch Wände hindurch Bewegungen zu identifizieren.

Dabei ist MOSRO permanent über Funk, ISDN oder GSM mit der Einsatzzentrale verbunden.

Er erkennt potenzielle Gefahren wie Rauch, Gas oder auch ungewöhnliche Temperaturschwankungen und alarmiert unmittelbar die Kontrollstation. Einmal entdeckte Personen verliert der Roboter nicht mehr aus den „Augen“. Er fordert sie dazu auf, sich über ihren Fingerabdruck zu identifizieren – mit deutlichen Worten in bis zu 24 Sprachen. Wird die Identifikation verweigert oder schlägt sie fehl, löst er Alarm aus.



### Internetrallye: Kleine Geschichte(n) der Roboter

Notiere jetzt zuerst die Uhrzeit, zu der Du mit der Bearbeitung anfängst.

Versuche, die folgenden Fragen zu beantworten. Jede Aufgabe ergibt bei korrekter Beantwortung 5 Punkte.

Für jede der Aufgaben hast Du bis zu **5 Minuten** Zeit. Für besonders schnelle Bearbeitung gibt es Zusatzpunkte, für besonders langsame Bearbeitung werden Punkte abgezogen.

Wichtig: Du musst die angegebenen Links nutzen!

#### Was ist ein Roboter?

Link: <http://www.chemie.de/articles/d/45201/>

---

---

---

---

#### Was steht hinter dem Begriff "Roboter – Robotik"?

<http://www.graupner-robotics.de/robot/einfuehrung.html>

---

---

---

---

#### Wann erschien der erste kommerziell genutzte Roboter?

Link:

[http://www.zdnet.de/it\\_business\\_technik\\_invasion\\_der\\_arbeitsmaschinen\\_roboter\\_als\\_flexible\\_helfer\\_story-11000009-39121507-4.htm](http://www.zdnet.de/it_business_technik_invasion_der_arbeitsmaschinen_roboter_als_flexible_helfer_story-11000009-39121507-4.htm)

---

---

---

#### Suche und schreibe 6 Arten von Roboter auf!

Link: <http://de.wikipedia.org/wiki/Roboter>

---

---

---

**Einer der berühmtesten Roboterhunde heißt und wurde entwickelt für:**

*Link: <http://de.wikipedia.org/wiki/Aibo>*

---

---

---

**Was versteht man unter Androiden & Cyborgs?**

*Link: <http://www.kantel.de/robot/folio030.html>*

---

---

---

---

**Von wem wurde der Begriff „Künstliche Intelligenz“ wann geprägt?**

*<http://www.tagesschau.de/inland/meldung228316.html>*

---

---

**Was ist eine RoboBoa™?**

*Links: <http://www.robosapien.de/RoboBoa.html> oder*

*<http://www.wowwee.com/en/products/toys/robots/robotics/robocreatures:roboboa>*

---

---

---

**Wie groß und was wiegt der Honda Asimo?**

*<http://www.honda-robots.com/german/html/asimo/frameset2.html>*

---

---

## Lösung Kleine **Geschichte(n) der Roboter:**

### **Was ist ein Roboter?**

**Nach der Definition des *Robot Institute of America* von 1979 ist ein Roboter ein unprogrammierbares, multifunktionales Manipulationswerkzeug, das dazu dient, Materialien, Teile, Werkzeuge oder spezialisierte Geräte anhand verschiedener vorprogrammierter Bewegungsabläufe zu bewegen, um eine Reihe an Aufgaben zu erledigen.**

### **Was steht hinter dem Begriff "Roboter – Robotik"?**

Die Bezeichnung Roboter leitet sich aus dem Ursprung „Fronddienst leisten“ sowie „ausüben schwerer Tätigkeiten“ ab, welche in abgewandelten Formen bis in das 14. Jahrhundert zurückreichen.

### **Wann erschien der erste kommerziell genutzte Roboter?**

Die ersten kommerziell genutzten Roboter erschienen in den frühen 60er Jahren auf der Bildfläche, als die ganze Welt von der Atomzeitalter-Wissenschaft des Kalten Krieges besessen war. Das von Engelberger gegründete Unternehmen Unimation entwickelte Roboterarme für Fabriken, während Barrett Electronics mit einem fahrerlosen Wagen für Gemüselagerhäuser herauskam, der sich an im Boden verlegten Drähten orientierte, von denen Signale ausgingen.

### **Suche und schreibe 6 Arten von Roboter auf!**

- Autonome mobile Roboter, Humanoide Roboter, Industrieroboter, Personal Robots, Portalroboter, Serviceroboter, Spielzeugroboter, Erkundungsroboter, Laufroboter, BEAM, Transportroboter

### **Einer der berühmtesten Roboterhunde heißt und wurde entwickelt für:**

AIBO - ein Haustier-Ersatz für Allergiker

### **Was versteht man unter Androiden & Cyborgs?**

In der weitesten Definition sind Androiden künstliche Wesen, die von Menschen nicht zu unterscheiden sind. Cyborg (Cybernetic Organism = kybernetischer Organismus), eine Mischform aus menschlichen und künstlichen Organismen.

### **Von wem wurde der Begriff „Künstliche Intelligenz“ wann geprägt?**

Geprägt wurde der Begriff der Künstlichen Intelligenz (artificial intelligence) 1955 von dem amerikanischen Informatiker John McCarthy.

### **Was ist eine Roboboa™?**

Der erste **Schlangen-Roboter** fürs Kinderzimmer  
(Film unter: <http://de.youtube.com/watch?v=zhl-2dSr25g&feature=related> )

### **Wie groß und was wiegt der Honda Asimo?**

Die Größe und das Gewicht wurden bei ASIMO weiter verringert, um ihn benutzerfreundlicher zu machen. So wurde der Roboter von 160 cm auf 120 cm

geschrumpft. Dabei verringert sich das Gewicht auf 54 kg, wenn man das Verhältnis Volumen/Gewicht beibehält.

### Finde mit Hilfe des Internet folgende Fragen heraus

- a) Notiere deine Ergebnisse auf ein Blatt oder in dein Heft. Schreibe die Fragen mit ab.
- b) Woher hast du die Information? Schreibe die Quelle (Link) mit auf.

- 1) W. Grey Walter entwickelte 1948/1949 die zwei "Schildkröten" Elsie und Elmer. Was konnten diese Roboter?**
- 2) Wie heißt der Roboter in Fritz Langs Film "Metropolis"?**
- 3) Wie lautet das "Nullte" Robotergesetz von Isaac Asimov?**
- 4) Wann landete das erste Roboterfahrzeug "Sojourner" erfolgreich auf dem Mars?**
- 5) In welcher Disziplin treten Forscherteams und ihre Roboter beim internationalen Wettbewerb "RoboCup" an?**
- 6) Das Wort Roboter wird 1921 erstmals in einem Theaterstück erwähnt. Wer schrieb dieses Stück?**
- 7) Was konnte Shakey Ende der 1960er Jahre als erster Roboter tun?**
- 8) Warum reiten seit 2005 Roboter Kamele?**
- 9) Was ist „ReWalk“? Für welche Menschen?**
- 10) Nenne 5 Roboterarten, die dem Menschen das „Leben leichter“ machen.**

Lösung: Finde mit Hilfe des Internet folgende Fragen heraus

**W. Grey Walter entwickelte 1948/1949 die zwei "Schildkröten" Elsie und Elmer. Was konnten diese Roboter?**

**Lösung:** W. Grey Walters Schildkröten waren **lichtempfindlich**. Die Namen Elmer und Elsie der dreirädrige Roboterfahrzeuge mit Schutzpanzer waren abgeleitet aus

**ELektroMEchanische Roboter, LichtSensitiv.**

**Wie heißt der Roboter in Fritz Langs Film "Metropolis"?**

1927 kommt es zur Erstvorführung von Fritz Langs "Metropolis". Darin entwickelt der wahnsinnige Wissenschaftler Rotwang den Roboter namens "**Falsche Maria**", den er seiner verstorbenen Frau nachempfandet.

**Wie lautet das "Nullte" Robotergesetz von Isaac Asimov?**

In "Robots and Empire" erweitert Science-Fiction-Autor Isaac Asimov seine drei Gesetze der Robotik um ein weiteres, Nulltes Gesetz:

*"Ein Roboter darf der Menschheit keinen Schaden zufügen oder durch Untätigkeit gestatten, dass die Menschheit zu Schaden kommt."*

Die ursprünglich drei Gesetze der Robotik von Isaac Asimov lauten:

1. Ein Roboter darf keinen Menschen verletzen oder durch Untätigkeit zu Schaden kommen lassen.
2. Ein Roboter muss den Befehlen eines Menschen gehorchen, es sei denn, diese Befehle stünden im Widerspruch zum Ersten Gesetz.
3. Ein Roboter muss seine eigene Existenz schützen, solange dieser Schutz nicht dem Ersten oder Zweiten Gesetz widerspricht.

**Wann landete das erste Roboterfahrzeug "Sojourner" erfolgreich auf dem Mars?**

Der amerikanischen Sonde Pathfinder mit dem Marsmobil "Sojourner" gelingt am 4. Juli **1997** eine spektakuläre Landung:

"Um Treibstoff und damit Geld zu sparen, fällt das Landefahrer nur leicht gebremst auf die Marsoberfläche. Riesige Airbags dämpfen den Aufprall. Die Fotos der Pathfinder-Sonde und die Ausflüge des Sojourner sind (fast) live im Internet zu beobachten. Mehr als 500 Millionen Surfer klicken sich im ersten Monat auf die Webseite der Mission."

**In welcher Disziplin treten Forscherteams und ihre Roboter beim internationalen Wettbewerb "RoboCup" an?**

Fußball

**Das Wort Roboter wird 1921 erstmals in einem Theaterstück erwähnt. Wer schrieb dieses Stück?**

Karel Capek (1890 - 1938)

**Was konnte Shakey Ende der 1960er Jahre als erster Roboter tun?**

Shakey war der erste selbstständig navigierende Roboter.

**Warum reiten seit 2005 Roboter Kamele?**

Kamelrennen haben in arabischen Ländern eine lange Tradition. Genauso lange ist die Tradition, Kinder als Jockeys bei den Rennen einzusetzen. Nach heftiger Kritik hatte der Präsident der Vereinigten Arabischen Emirate, Anfang Juli 2005 das Mindestalter für Kamel-

Jockeys auf 18 Jahre festgesetzt. So entstand die Idee, Jockey-Roboter zu entwickeln, die leichter sind als erwachsene Menschen.

ANHANG

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

Die Klasse LCD unterstützt den Betrieb des qfix LCD Displays. Das Display muss über den I<sup>2</sup>C-Bus an das Controllerboard angeschlossen sein.

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

Klasse: SoccerBoard		Datei: <i>qfixSoccerBoard.h</i>
RWert	Funktionsname	Erklärung
<i>SoccerBoard()</i>		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.
<i>void</i>	<i>ledOn(int i)</i>	Schaltet die LED mit Index <i>i</i> an. <i>i</i> muss im Bereich 0 bis 1 liegen.
<i>void</i>	<i>ledOff(int i)</i>	Schaltet die LED mit Index <i>i</i> aus. <i>i</i> muss im Bereich 0 bis 1 liegen.
<i>void</i>	<i>ledsOff()</i>	Schaltet alle LEDs aus.
<i>void</i>	<i>led(int i, bool state)</i>	Schaltet die LED mit dem Index <i>i</i> an, falls <b>state=true</b> , bzw. aus, falls <b>state=false</b> . <i>i</i> muss im Bereich 0 bis 1 liegen.
<i>bool</i>	<i>button(int i)</i>	Liefert <b>true</b> zurück, falls der Button mit dem Index <i>i</i> gedrückt ist, ansonsten <b>false</b> . <i>i</i> muss im Bereich 0 bis 1 liegen.
<i>void</i>	<i>waitForButton(int i)</i>	Wartet, bis der Button mit dem Index <i>i</i> gedrückt und wieder losgelassen wurde. <i>i</i> muss im Bereich 0 bis 1 liegen.
<i>void</i>	<i>motor(int i, int speed)</i>	Setzt den Motor mit dem Index <i>i</i> auf den Wert <b>speed</b> , der im Bereich -255 bis +255 liegen muss. <i>i</i> muss im Bereich 0 bis 5 liegen.
<i>void</i>	<i>motorsOff()</i>	Schaltet alle Motoren aus.
<i>int</i>	<i>analog(int i)</i>	Liefert den Wert des Analog-Eingangs mit dem Index <i>i</i> zurück. Der Rückgabewert liegt im Bereich 0-255. <i>i</i> muss im Bereich 0 bis 7 liegen.
<i>bool</i>	<i>digital(int i)</i>	Liefert <b>true</b> zurück, falls der Digital-Eingang mit dem Index <i>i</i> High ist, ansonsten <b>false</b> . <i>i</i> muss im Bereich 0 bis 7 liegen.
<i>void</i>	<i>powerOn(int i)</i>	Schaltet den Power-Ausgang mit Index <i>i</i> an. <i>i</i> muss im Bereich 0 bis 7 liegen.
<i>void</i>	<i>powerOff(int i)</i>	Schaltet den Power-Ausgang mit Index <i>i</i> aus. <i>i</i> muss im Bereich 0 bis 7 liegen.
<i>void</i>	<i>power(int i, bool state)</i>	Schaltet den Power-Ausgang mit dem Index <i>i</i> an, falls <b>state=true</b> , bzw. aus, falls <b>state=false</b> . <i>i</i> muss im Bereich 0 bis 7 liegen.
<i>void</i>	<i>sleep(int s)</i>	Wartet <b>s</b> Sekunden. (Genauer als die globale Wartezeit.)
<i>void</i>	<i>msleep(int ms)</i>	Wartet <b>ms</b> Millisekunden. (Genauer als die globale Wartezeit)

## ANHANG B

### Standard-Programme

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

Als Befehle stehen sämtliche Methoden der Klasse `BobbyBoard` zur Verfügung. Beispiel: Anschalten eines Motors mit `bobby.motor(0,255);`

### MiniBoard

Das entsprechende Standard-Programm für das MiniBoard sieht folgendermaßen aus:

```
#include "qfixMiniBoard.h"
MiniBoard robot;
int main()
{
  /* Hier stehen die Anweisungen */
}
```

Als Befehle stehen die Methoden der Klasse `MiniBoard` zur Verfügung.

### SoccerBoard

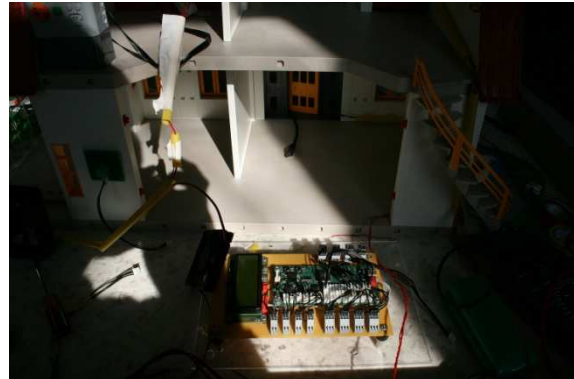
Das entsprechende Standard-Programm für das SoccerBoard sieht folgendermaßen aus:

```
#include "qfixSoccerBoard.h"
SoccerBoard robot;
int main()
{
  /* Hier stehen die Anweisungen */
}
```

Als Befehle stehen die Methoden der Klasse `SoccerBoard` zur Verfügung. Beispiel: Anschalten eines Motors mit `robot.motor(0,255);`

PROJEKTBESCHREIBUNG: DAS ÜBERWACHTE/GESTEUERTE WOHNHAUS S. 67

Von der Rückseite her wird das Haus verkabelt. Die Kabelkanäle – in diesem Haus Strohhalme – werden mit Teppichklebeband oder Klett befestigt. Das Soccerboard wird auf eine Platine befestigt und die Anschlüsse mit Steckverbindungen vorbereitet. Das hat den Vorteil, dass die Kabel direkt genutzt werden können, ohne viel Kabel mit Schuhen löten zu müssen.



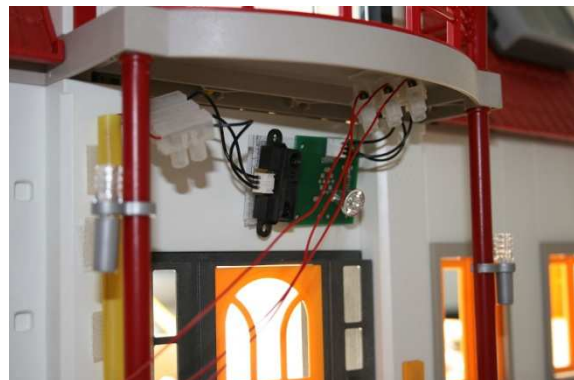
Auf diesem Bild sieht man die Installation einer Alarmanlage mit einer Fotozelle. Wenn der Lichtkontakt unterbrochen wird, soll eine Sirene (Beeper) erklingen und die Alarmlampe leuchten. Der Alarm soll durch einen Drucktaster gestoppt werden können.



Auf diesem Bild sieht man die Alarm-LED am Giebel des Hauses.



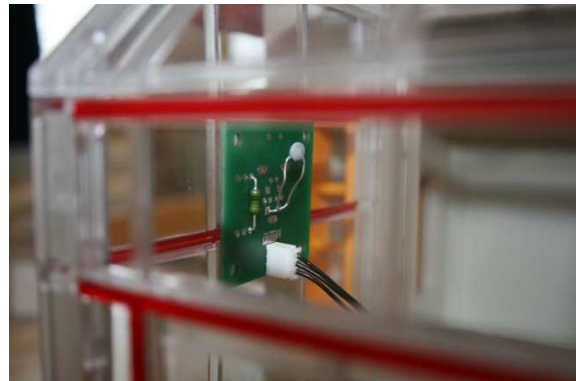
An der Haustür wird ein Bewegungsmelder installiert (Infrarotsensor). Bei Annäherung soll die LED-Lampe für 5 Sekunden leuchten, danach sich wieder abschalten. Dieses Prinzip soll auch für mehrere Bereiche im und am Haus realisiert werden.



Der Ausschalter zur Deaktivierung der Alarmanlage. Dieser ist an der Fassade angebracht.



In dem Haus und im Wintergarten befinden sich die Temperatursensoren. Steigt die Temperatur über einen bestimmten Wert, so soll ein Ventilator (PC) Kühlung bringen, bzw. die Luft umwälzen.



Die Fenster sind mit Kontakten ausgestattet. Dieses kann genutzt werden:

- als Alarmanlage
- zur Heizungssteuerung.

Ist eine bestimmte Temperatur im Raum erreicht, kann die Heizung bei geöffnetem Fenster sich ausschalten. Im Modell schwierig, da das Haus auf der Rückseite offen ist.



Solarkollektoren liefern Strom. In diesem Fall wurde auf das LEGO 9688 Energie Addon zurückgegriffen. In diesem Bild wird ein Windrad angetrieben.

