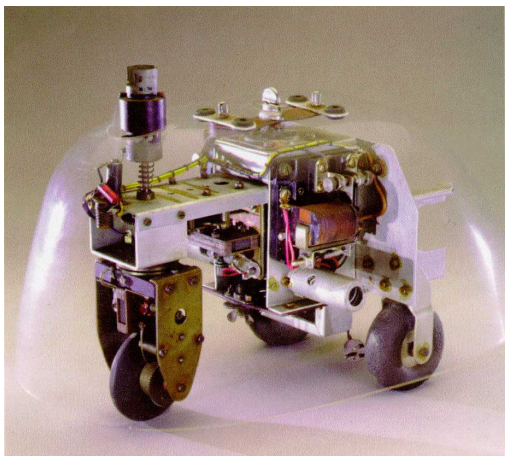
Inhalt

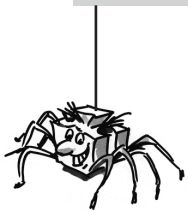
Wozu brauchen wir Roboter?

■ Der Begriff des Roboters wird erstmals 1923 im Roman „Golem" von Carel Capek verwendet. Diese künstlich erschaffene Figur sollte mit ihren Fähigkeiten menschliche Arbeit ersetzen.
In den 30er und 40er Jahren des 20. Jahrhunderts wird aus dem Roboter eher eine Art Automat. Diverse Versuche, ihn mit menschlichen Eigenschaften zu versehen, z. B. einem Kopf mit blinkenden Lampen als Augen, rufen bei uns heute höchstens noch ein müdes Lächeln hervor. Von Mobilität oder gar Intelligenz ist bei diesen Maschinen wenig zu bemerken. Da das Prinzip der Steuerung großen Einfluss auf die Robotertechnik hat, wurde mit dem Aufkommen elektronischer Schaltungen der Aufbau von Robotern realistischer. Die Frage nach der „Intelligenz" des Roboters ist auch heute noch Forschungs- und Untersuchungsgegenstand vieler Firmen, Institute und Universitäten.

■ Erste Lösungsansätze versprach man sich von der so genannten Kybernetik. Die Bezeichnung „Kybernetik" ist von dem griechischen Wort Kybernetes abgeleitet. Der Kybernetes war der Navigator auf den griechischen Ruderschiffen. Er mußte den Schiffsort bestimmen und den notwendigen Kurs bis zum Ziel errechnen.
Damit ist klar, die Kybernetik sollte den Roboter „intelligent" machen. Wie kann man sich ein solches intelligentes Verhalten überhaupt vorstellen?
Wir wollen versuchen, uns dies mit Hilfe eines Gedankenexperimentes zu verdeutlichen. Jeder wird schon einmal das Verhalten einer Motte im Lichtkreis einer Lampe beobachtet haben. Die Motte erkennt die Lichtquelle, fliegt darauf zu, um dann kurz vor dem Aufprall auf die Lampe auszuweichen. Es ist klar, dass die Motte für dieses Verhalten die Lichtquelle erkennen, einen Weg dahin ermitteln und dann darauf zufliegen muss. Diese Fähigkeiten basieren auf instinktiven, intelligenten Verhaltensmustern des Insekts.
Versuchen wir nun diese Fähigkeiten in ein technisches System zu übertragen. Wir müssen die Lichtquelle erkennen (optische Sensoren), eine Bewegung ausführen (Motoren steuern) und wir müssen einen sinnvollen Zusammenhang zwischen Erkennung und Bewegung aufstellen (das Programm).

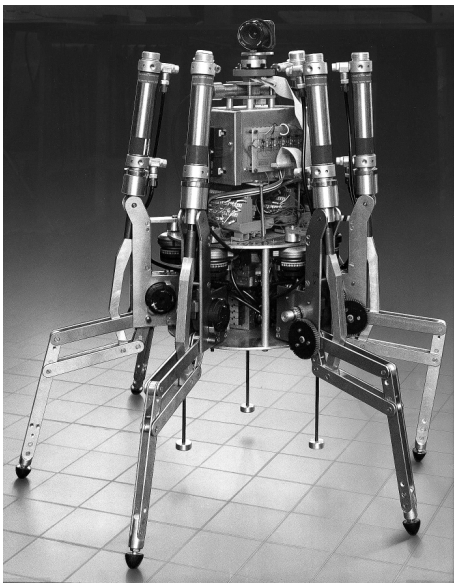


■ Das oben beschriebene Gedankenexperiment hat der Engländer Walter Grey in den 50er Jahren in die Tat umgesetzt.
Mit Hilfe einfacher Sensoren, Motoren und elektronischer Schaltungen wurden verschiedene „kybernetische" Tiere geschaffen, denen dann ganz spezifisches Verhalten, wie z. B. das einer Motte, eigen waren. Die Abbildung zeigt einen Nachbau der „kybernetischen" Schildkröte die im Smithsonian Museum, Washington ausgestellt ist.
Basierend auf diesen Überlegungen werden auch wir für unsere Roboter entsprechende „Verhaltensmuster" erstellen und versuchen diese in Form von Programmen dem Roboter verständlich zu machen.



■ Doch wozu brauchen wir nun mobile Roboter? Versuchen wir einmal das Verhalten unserer „Gedankenmotte“ auf technische Geräte anzuwenden. Ein einfaches Beispiel dafür ist die Lichtsuche. Wir modifizieren die Lichtquelle, indem wir einen hellen Streifen, die Leitlinie, auf dem Boden anbringen und die Sensoren nicht mehr nach vorn, sondern nach unten ausrichten. Mit Hilfe derartiger Leitlinien kann sich ein mobiler Roboter beispielsweise in einer Lagerhalle orientieren. Zusätzliche Informationen, z. B. in Form eines Strichcodes an bestimmten Stellen der Linie, veranlassen den Roboter an diesen Positionen zu weiteren Aktionen, wie z. B. das Aufnehmen oder Absetzen einer Palette.
Solche Robotersysteme existieren bereits tatsächlich. In großen Krankenhäusern fallen zum Teil recht lange Transportwege für Verbrauchsmaterialen, wie z. B. Bettwäsche, an. Der Transport dieser Materialen durch das Pflegepersonal ist aufwändig und zum Teil mit schwerer körperlicher Arbeit verbunden. Zudem schränken solche Tätigkeiten die für die Pflege der Patienten bereit stehende Zeit ein.

■ Seit einigen Jahren beschäftigen sich Wissenschaftler mit einer weiteren, in der Natur sehr verbreiteten Bewegungsform, dem Gehen bzw. Laufen. Es werden Roboter entwickelt, die in der Lage sind, sich auf Beinen fort zu bewegen. Ein Beispiel für einen sechsbeinigen Laufroboter ist der an der Königlichen Militärakademie in Brüssel entwickelte elektropneumatische Laufroboter „Achille“. Ausgestattet mit einer Kamera oben und an den sechs Beinen, soll dieser Roboter mechanisch auf erhöhte oder vertiefte Hindernisse (Gegenstände oder Löcher) reagieren.
Solche Laufmaschinen könnten überall dort eingesetzt werden, wo Rad- und Kettenfahrzeuge kaum mehr eine Chance hätten, so z. B. in äußerst unebenem oder nachgiebigem Gelände, beim Klettern über Hindernisse, Treppen steigen, Überwinden von Gräben oder beim Einsatz an schwer zugänglichen und gefährlichen Stellen in Kernkraftwerken, Bergwerkstollen oder bei Rettungsaktionen.



Wir erkennen also, dass mobile Roboter durchaus einen wichtigen Platz in der modernen Gesellschaft einnehmen können.

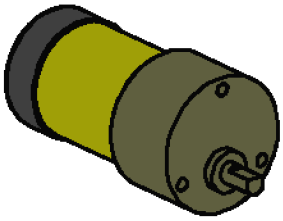
Roboter aus fischertechnik

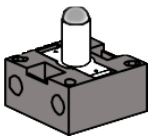
■ Wie können wir nun aus unserem fischertechnik-Baukasten Roboter bauen? Für einen Roboter brauchen wir außer den Sensoren (z. B Taster,) und Aktoren (z. B. Motoren) viele mechanische Teile um daraus ein Modell zu konstruieren. Der fischertechnik-Baukasten ROBO Mobile Set ist hierzu die ideale Grundlage. Folgende Sensoren und Aktoren sind in diesem Baukasten enthalten:

Aktoren

Powermotor:

Zwei dieser kräftigen Gleichstrommotoren (9VDC/2,4W) mit eingebautem Getriebe und einer Untersetzung von 50:1 treiben die mobilen Robotermodelle an (d. h. der Motor dreht sich 50 mal und die Welle, die aus dem Motor ragt, in derselben Zeit nur einmal).



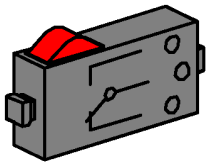


Linsenlampe:

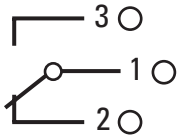
Zur Ausgabe einfacher Lichtsignale dient diese Glühlampe (9VDC/150mA).
Im Glaskolben der Lampe ist eine Linse integriert, die das austretende Licht bündelt. Richtet man den Lichtstrahl auf einen Helligkeitssensor (Fototransistor, siehe unten) kann man so eine Lichtschranke bauen, die Hell und Dunkel unterscheidet. Die Lampe kann auch zum Anzeigen bestimmter Zustände oder als blinkende Lampe zur Ausgabe von Warnmeldungen verwendet werden. Im Baukasten wird die Lampe zusammen mit 2 Fototransistoren als Spezialsensor zur Linienerkennung verwendet.

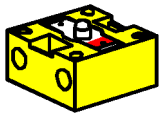
Sensoren

■ Bei Sensoren unterscheidet man zwischen digitalen und analogen Sensoren.
Der Taster ist ein Beispiel für einen digitalen Sensor. Digitale Größen können nur 2 verschiedene Zustände annehmen. Diese Zustände werden mit 0 bzw. 1 gekennzeichnet. Für den Taster bedeutet "0" es fließt kein Strom zwischen den Anschlüssen, „1" bedeutet es fließt Strom.



Der fischertechnik **Taster** ist als Wechselschalter ausgelegt. Deshalb besitzt er 3 Anschlüsse. Beim Drücken des roten Knopfes wird mechanisch ein Schalter betätigt, der die Anschlüsse 1 und 3 miteinander verbindet. Gleichzeitig wird der Kontakt zwischen den Anschlüssen 1 und 2 unterbrochen, welche im Ruhezustand miteinander verbunden waren. Auf diese Weise können beide möglichen Ausgangslagen abgefragt werden:
Im Ruhezustand geschlossen (Anschlüsse 1 und 2 belegt)
Im Ruhezustand geöffnet (Anschlüsse 1 und 3 belegt).





Der **Fototransistor** kann sowohl als digitaler als auch als analoger Sensor verwendet werden. Im ersten Fall dient er zur Erkennung deutlicher Hell-Dunkel-Übergänge, z. B. einer markierten Linie. Es können jedoch auch Lichtmengen in ihrer Stärke unterschieden werden, dann arbeitet der Fototransistor als analoger Sensor. Analoge Werte können sich beliebig zwischen ihren Extremwerten ändern. Damit diese Größen vom Computer verarbeitet werden können, müssen sie in entsprechende Zahlenwerte umgewandelt werden.

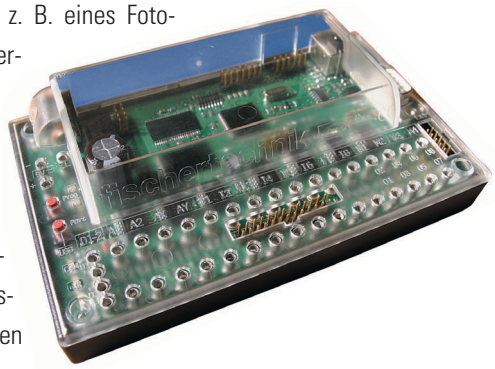
Beim Fototransistor handelt es sich übrigens um ein so genanntes Halbleiterbauelement, dessen elektrische Eigenschaften lichtstärkeabhängig sind. Jeder kennt Solarzellen, mit denen aus Sonnenlicht

Strom gewonnen wird. Den Fototransistor können wir als Kombination von Minisolarzelle und Transistor verstehen. Auf den Fototransistor auftreffende Lichtimpulse (Photonen) erzeugen einen sehr kleinen Strom, der dann vom Transistor verstärkt wird.

Hinweis:
Beim Anschluss des Fototransistors auf richtige Polung achten: Rote Markierung = Plus.
Zulässige Spannung: 30V max.

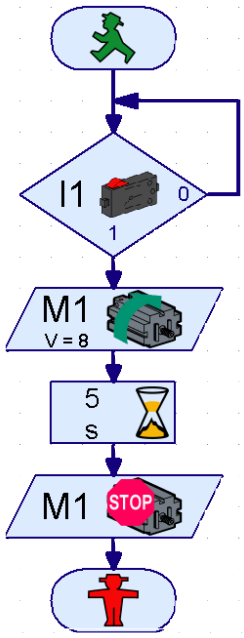
ROBO Interface

■ An das ROBO Interface können wir verschiedene Sensoren und Aktoren anschließen und auswerten. Neben 8 digitalen Eingängen stellt das ROBO Interface mehrere Analogeingänge bereit. So z. B. wird ein an den Eingängen AX und AY angelegter Widerstandswert zwischen 0 und 5,5 kΩ in einen Zahlenwert zwischen 0 und 1024 umgewandelt. Die Messwerte eines Helligkeitssensors, z. B. eines Fototransistors, werden damit erfasst und stehen für eine weitere Bearbeitung zur Verfügung. An den analogen Eingängen A1 und A2 können Spannungen zwischen 0 und 10VDC gemessen werden.



Die wichtigste Funktion des Interfaces besteht in der logischen Verknüpfung der Eingangsgrößen. Dazu benötigt das Interface ein Programm. Das Programm entscheidet, in welcher Weise aus Eingangsdaten, den Sensorsignalen, passende Ausgangsdaten, Motorsteuersignale etc., entstehen. Mit dem ROBO Interface verfügen wir über genug Rechenpower um auch anspruchsvolle Programme zu entwerfen.

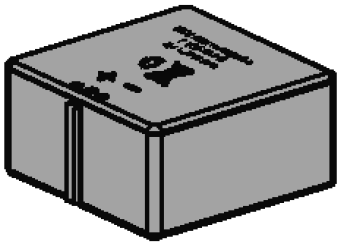
Software ROBO Pro



■ Damit wir möglichst effektiv die für das Interface notwendigen Programme erstellen können, gibt es eine grafische Programmieroberfläche. Hinter dem Begriff „Programmieroberfläche" verbirgt sich eine Software, die es uns ermöglicht, auf sehr komfortable Weise unsere Programme zu erstellen. Dies geschieht mit Hilfe grafischer Symbole. Der Computer des ROBO Interface kann eigentlich nur Befehle aus seinem so genannten Maschinenbefehlssatz ausführen. Das sind im Wesentlichen einfache Kontrollstrukturen, deren Anwendungen für Einsteiger außerordentlich schwierig sind. Die PC-Software ROBO Pro stellt deswegen Grafikelemente bereit, die anschließend in eine für das Interface ausführbare Sprache übersetzt werden.

Stromversorgung

■ Das Einzige, was wir zum Baukasten ROBO Mobile Set noch zusätzlich benötigen, ist das Accu Set Art.-Nr. 34969. Es enthält den Akku-Pack als mobile Stromversorgung für unsere Robotermodelle und ein spezielles Ladegerät für den Akku-Pack.
Am Besten laden wir den Akkupack gleich mit dem Ladegerät auf, damit er voll ist, wenn wir mit den Experimenten beginnen wollen.



Vorgehensweise beim Experimentieren

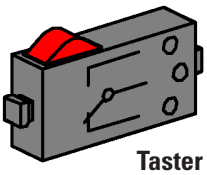
■ Bei unserem Eindringen in die faszinierende Welt der mobilen Roboter werden wir schrittweise vorgehen. Wir beginnen mit einem einfachen Testaufbau zur Prüfung der Grundfunktionen von Interface und Sensorik. Danach bauen wir einfache Modelle, denen bestimmte Aufgaben zugeordnet sind und versuchen uns dann mit immer komplizierteren Systemen.
Wem das Erstellen eigener Programme irgendwann zu kompliziert wird und zu lange dauert, der kann einfach die mitgelieferten Beispielprogramme auf das Interface laden und die Roboter damit betreiben. Damit man bei auftretenden Fehlern nicht verzweifelt, gibt es am Ende ein Kapitel zur Fehlersuche.

Ein sehr wichtiger Punkt ist die Sorgfalt beim Aufbau und der Inbetriebnahme unserer Roboter. Beim Anschluss der elektrischen Komponenten halten wir uns eng an die Vorgaben und überprüfen lieber doppelt oder dreifach, ob alles stimmt. Bei den mechanischen Konstruktionen, auch bei eigenen Kreationen, achten wir sehr auf Leichtgängigkeit und geringes Spiel in den Getrieben und Befestigungen. Es ist unserer Kreativität vorbehalten, eigene Programme zu schreiben und damit neues "Verhalten" zu definieren. Begrenzt wird das nur durch die Speichermenge und Rechenleistung der Hardware. Die folgenden Beispiele geben dazu einige Anregungen.

Erste Schritte

■ Nach den theoretischen Überlegungen wollen wir nun beginnen eigene Experimente durchzuführen. Sicherlich möchte der eine oder andere sofort starten, vielleicht sogar mit dem großen Laufroboter. Das ist natürlich möglich und bei sorgfältiger Beachtung der Bauanleitung gelingt der Aufbau des Modells auch auf Anhieb.
Doch was tun, wenn es nicht funktioniert? In diesen Fällen muß systematisch nach der Fehlerursache gesucht werden. Doch bevor wir uns damit beschäftigen, prüfen wir erst einmal das Zusammenspiel von Computer und Interface.

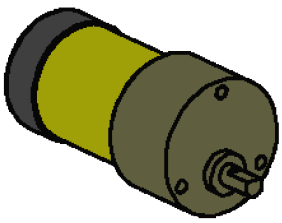
Im Handbuch der Software ROBO Pro wird in Kapitel 1 und 2 beschrieben, wie die Steuersoftware auf dem PC installiert und das Interface angeschlossen wird. Mit Hilfe des Interface-Tests testen wir die unterschiedlichen Sensoren bzw. Aktoren.



Taster

Taster

Wir können jetzt z. B. einen Taster an den Digitaleingang I1 anschließen und beobachten wie sich der Zustand des Eingangs beim Betätigen des Tasters ändert.



Powermotor

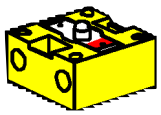
Powermotor

Die Ausgänge prüfen wir, indem wir einen Motor mit einem Motorausgang, z. B. M1, verbinden. Mit der linken Maustaste können wir den Motor in Drehung versetzen und mit dem Schieber die Geschwindigkeit verändern.

Fototransistor

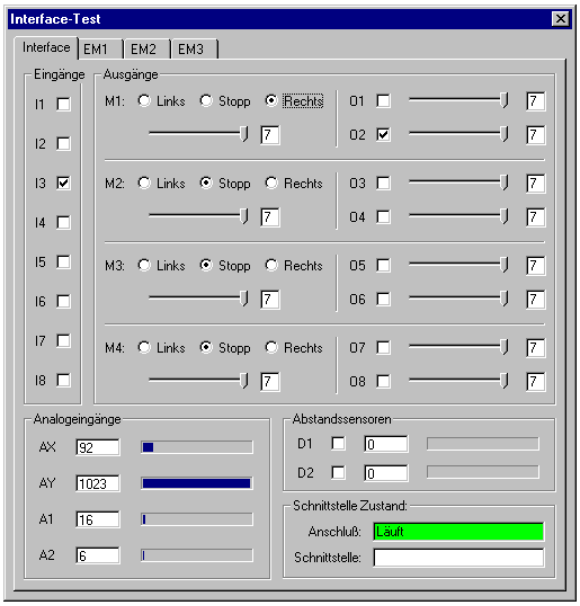
Wollen wir auch den Analogeingang AX testen, können wir dazu einen Fototransistor als Analogsensor verwenden.

Während beim Motor bzw. Taster die Polarität der Anschlüsse keine Rolle spielt (der Motor dreht sich im ungünstigsten Fall verkehrt herum), ist der richtige Anschluss des Fototransistors für die korrekte Funktion zwingend notwendig.
Den Kontakt des Transistors mit der roten Markierung verbinden wir mit einem roten Anschlußstecker, den anderen Kontakt mit einem grünen Stecker. Der zweite grüne Stecker kommt in die näher am Rand des Interfaces liegende Buchse des Eingangs AX, der zweite rote Stecker passt in die weiter innen liegende Buchse von AX. (Achtung: Schließt man den Fototransistor an einen Digitaleingang I1-I8 an, gehört der rote Stecker in die Buchse, die näher am Gehäuserand liegt).

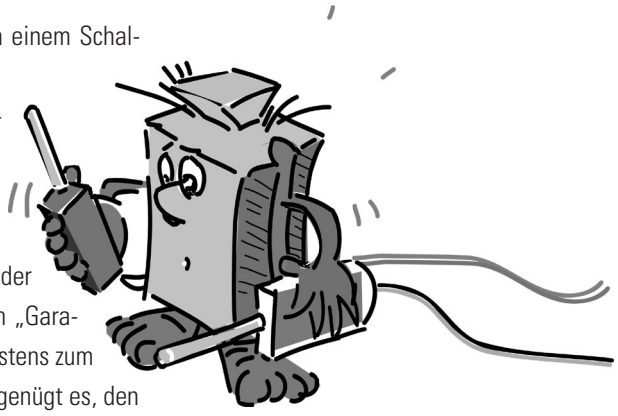


Fototransistor

Nun können wir mit Hilfe einer Taschenlampe die Beleuchtungsstärke des Fototransistors variieren und damit den Ausschlag des blauen Balkens von AX verändern. Sollte der Zeiger sich nicht von seinem Maximalausschlag wegbewegen, schauen wir nochmals nach den Anschlüssen des Fototransistors. Ist hingegen auch bei ausgeschalteter Taschenlampe der Zeiger auf Null, so kann es sein, dass die Beleuchtung im Raum, die Umgebungshelligkeit, zu groß ist. Der Ausschlag ändert sich dann, wenn wir den Fototransistor abdecken.

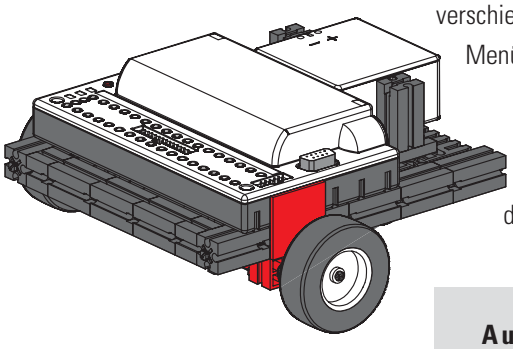


Um nochmals kurz auf die Farbzuordnung der Stecker zu kommen: Wir achten genau darauf beim Zusammenbau immer einen roten Stecker an die rote Leitung anzuschließen und einen grünen Stecker mit der grünen Leitung zu verbinden. Wenn es in einem Schaltungsaufbau auf die richtige Polarität ankommt, nehmen wir immer eine rote Leitung für den Pluspol und eine grüne Leitung für den Minuspol. Das mag ein wenig übergenau erscheinen, aber für eine systematische Fehlersuche ist eine eindeutige Farbzuordnung eine erhebliche Erleichterung.



Mit einem einfachen Programm wollen wir unsere ersten Schritte auf dem Gebiet der Robotik abschließen. Das im Kapitel 3 des ROBO Pro Handbuchs erklärte Programm „Garagentorsteuerung“ hat zwar nichts mit mobilen Robotern zu tun, es eignet sich aber bestens zum Kennenlernen der Software ROBO Pro. Um das Programm nachvollziehen zu können genügt es, den Motor und drei Taster aus dem Baukasten ROBO Mobile Set an das Interface anzuschließen. Alles weitere wird ausführlich im Softwarehandbuch beschrieben.

Der erste einfache Roboter



■ Nach Interfacetest und Garagentorsteuerung wollen wir nun endlich den ersten Roboter in Betrieb nehmen. Wir bauen das Modell „Einfacher Roboter" mit den beiden Antriebsmotoren entsprechend der Bauanleitung auf. Das geht recht einfach und schnell, weil dieses Modell bewusst nur das Nötigste enthält, was ein Roboter zum Fahren braucht. Die Motoren schließen wir an die Ausgänge M1 und M2 an.

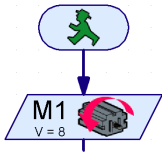
Wir öffnen die Software ROBO Pro und legen ein neues Programm an (DATEI – NEU). ROBO Pro bietet verschiedene Schwierigkeitsstufen an, in denen wir arbeiten können. Sie lassen sich in ROBO Pro im Menüpunkt LEVEL einstellen. Momentan genügt uns Level 1.

Es erscheint ein leeres Arbeitsblatt und am linken Bildschirmrand das Elementfenster, aus dem wir die verschiedenen Programmelemente mit der linken Maustaste auswählen und auf der Arbeitsfläche platzieren können. Mit der rechten Maustaste ändern wir die Eigenschaften.

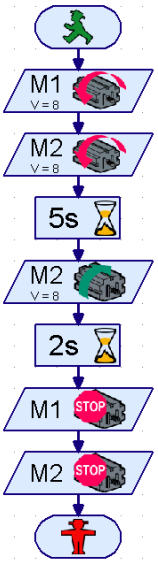
Aufgabe 1 (Level 1):
Unser „Einfacher Roboter" soll 5 Sekunden lang geradeaus fahren, anschließend 2 Sekunden lang im Kreis drehen und danach stehen bleiben.

Tipps:

Den ersten Roboter programmieren wir Schritt für Schritt gemeinsam:



- Wir beginnen mit dem grünen Ampelmännchen. Es symbolisiert den Programmstart.
- Dann holen wir uns zunächst das Motorsymbol aus dem Elementfenster und setzen es unter das Startelement, so dass die Verbindungslinie automatisch gezeichnet wird. Im Eigenschaftsfenster stellen wir den Motorausgang „M1" sowie die Drehrichtung „Links" ein und bestätigen mit OK.



- Unter dieses Symbol setzen wir auf die gleiche Weise noch ein Motorsymbol und schalten damit den Motor 2 ein.
- Um eine bestimmte Zeit zu warten, verwenden wir das Element Wartezeit, platzieren es unter das zweite Motorsymbol und stellen die Zeit auf 5 Sekunden ein.
- Danach lassen wir den Motor M2 in die andere Richtung drehen (nach rechts) warten danach 2 Sekunden und schalten schließlich beide Motoren aus. Unser Programm endet mit dem Endesymbol, dem roten Ampelmännchen. Die Abbildung zeigt den fertigen Programmablauf.

Wer sich nicht sicher ist, ob alles richtig ist, vergleicht sein Programm mit dem mitgelieferten Beispielprogramm. Dazu wird das eigene Programm vorher gespeichert und die Datei Einfacher Roboter 1.rpp aus dem Beispielverzeichnis von ROBO Pro geladen (Standardeinstellung C:\Programme\ROBO Pro\Beispielprogramme\ROBO Mobile Set).

Ist alles in Ordnung, wird das Programm per Download in das Interface geladen. Nach Drücken des Buttons Download erscheint ein Dialogfenster. Dort stellen wir ein, dass das Programm in den FLASH-Speicher 1 geladen und sofort nach dem Download gestartet werden soll.

Gleich nach dem Download fährt unser Modell los, dreht sich danach kurz und bleibt stehen. Wollen wir das Programm erneut starten, drücken wir am Interface kurz die Prog-Taste. Die LED Prog1 blinkt dann wieder, solange das Programm läuft. Danach leuchtet sie dauernd. Das Programm im FLASH-Speicher

des Interfaces bleibt übrigens auch dann erhalten, wenn die Stromversorgung am Interface unterbrochen wird. Wir probieren das aus, indem wir einen Stecker am Akkupack abziehen. Wir stecken den Stecker wieder ein, wählen das gespeicherte Programm aus, indem wir die Prog-Taste so lange drücken, bis die LED Prog1 leuchtet. Wir drücken die Taste erneut und starten so das Programm.

Ziemlich wenig, was der Roboter bis jetzt macht, oder? Also wollen wir die Aufgabe etwas erweitern.

Aufgabe 2 (Level 1):

Damit unser Roboter nicht schon nach 7 Sekunden stehen bleibt, wollen wir ihm jetzt das Tanzen beibringen.

- **Lass ihn unterschiedlich lange geradeaus, linksrum, rechtsrum, rückwärts fahren, und das in unterschiedlichen Geschwindigkeiten.**
- **Der Ablauf soll sich so lange wiederholen, bis das Programm mit dem Prog-Taster am Interface beendet wird.**

Tipps:

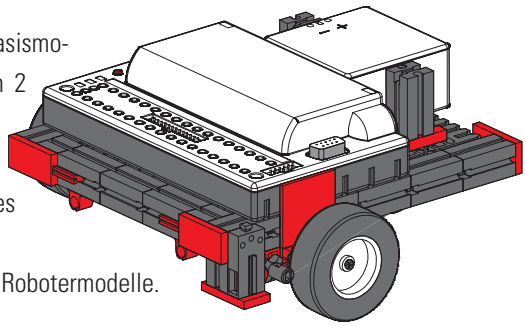
- Pole einfach immer wieder die Motoren so um, dass der Roboter in die gewünschten Richtungen fährt.
- Die Geschwindigkeit der Motoren kannst du im Eigenschaftsfenster von jedem Motorsymbol zwischen 1 und 8 einstellen. Drehen M1 und M2 mit unterschiedlichen Geschwindigkeiten in die gleiche Richtung, fährt der Roboter ein Kurve.
- Damit das Programm ständig wiederholt wird, ziehst du eine Verbindungslinie vom Ausgang des letzten Programmelements zu der Linie, die in das erste Element hineinführt.
- Ein fertiges Beispiel findest du unter Einfacher Roboter 2.rpp.

Herzlichen Glückwunsch, du hast deinen ersten eigenen Roboter gebaut und selbst programmiert. Er ist zwar noch nicht sonderlich intelligent, denn er erkennt keine Hindernisse und fällt vom Tisch, wenn du nicht aufpasst. Aber das wird sich im Laufe der weiteren Experimente noch ändern.

Intelligente Fahrroboter

■ Damit Roboter ihre Umgebung erkennen können, benötigen sie Sensoren. Mit den folgenden Modellvorschlägen werden einige Varianten mobiler Roboter vorgestellt, an denen der Einsatz unterschiedlicher Sensoren erprobt wird. Dabei kommt es darauf an, sowohl interne Zustände des Roboters, z.B. Wegstreckenmessung durch Impulsräder als auch Signale von außen, z. B. bei der Licht- oder Spurensuche, zu verknüpfen. Zu jedem Modell werden dazu bestimmte Aufgaben gestellt. Sie sollen als Anregung dienen und dich mit der Materie vertraut machen. Die Programme zu den einzelnen Aufgaben befinden sich im ROBO Pro Verzeichnis unter \Beispielprogramme\ROBO Mobile Set\. Lass dir zu den Modellen aber auch ruhig eigene Aufgaben einfallen. Wenn du die folgenden Beispiele durchgegangen bist, hast du sicher noch viele weitere Ideen.

Basismodell



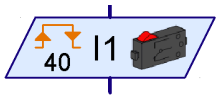
■ Gegenüber unserem ersten „Einfachen Roboter" ist das Basismodell stabiler und robuster aufgebaut. Es enthält außerdem 2 Sensoren zur Wegmessung, die jeweils aus einem Taster und einem Impulsrad bestehen. Das Impulsrad ist mit der Motorachse verbunden und betätigt bei jeder Umdrehung des Motors vier Mal einen Taster.
Dieses Modell dient als Grundlage für die weiteren mobilen Robotermodelle.

Baue das Basismodell entsprechend der Bauanleitung auf. Gehe beim Aufbau sehr sorgfältig vor. Wenn mechanisch alles fertig ist, prüfst du die Leichtgängigkeit der Motoren indem du jeden Motor kurz ohne das Interface direkt mit dem Akku verbindest.

Aufgabe 1 (Level 1):

- **Programmiere das Interface so, dass das Modell 40 Impulse geradeaus fährt. Verwende zur Messung der Impulse den Zähltaster am Eingang I1.**
- **Messe die Strecke, die das Modell zurücklegt und berechne, welche Strecke pro Impuls zurückgelegt wird.**
- **Wiederhole den Versuch 3 Mal und halte in der Tabelle fest, wie stark die Werte schwanken.**

Tipps:



- Schalte zunächst beide Motoren ein (Drehrichtung links).
- Um die Impulse an I1 zu zählen verwendest du das Programmelement **Impulszähler**.
- Zähle beide Impulsflanken (0-1 beim Drücken, 1-0 beim Loslassen des Tasters). Dies kannst du im Eigenschaftsfenster unter **Impulstyp** einstellen. Du erhöhst damit die Genauigkeit der Wegmessung.
- Danach schaltest du die Motoren wieder aus und beendest das Programm.
- Das fertige Programm findest du unter Basismodell1.rpp.

Ergebnis:

	Anzahl Impulse	Zurückgelegte Strecke	Strecke/Impuls
Versuch 1	**40**		
Versuch 2	**40**		
Versuch 3	**40**		

Ganz grob kann man festhalten, dass das Modell pro Impuls etwa eine Strecke von einem Zentimeter zurücklegt.

Inzwischen weißt du auch, welche Drehrichtung du für die einzelnen Motoren einstellen musst, damit das Modell in eine bestimmte Richtung fährt. Halte diese Erkenntnisse in der folgenden Tabelle fest, damit du nicht bei jeder Fahrtrichtungsänderung darüber nachdenken musst. Wenn du die Modelle exakt so verkabelst, wie es in der Bauanleitung dargestellt ist, bedeutet linke Drehrichtung bei jedem Motor, dass sich das Rad vorwärts dreht. So sind in allen Beispielprogrammen die Motoren programmiert.

Ergänze die Tabelle:

Fahrtrichtung Modell	Drehrichtung M1	Drehrichtung M2
Vorwärts	**Links**	**Links**
Rückwärts		
Links		
Rechts		
Stopp		

Um nicht bei jedem Richtungswechsel zwei Motorsymbole auf den Bildschirm setzen zu müssen, kannst du für jede Fahrtrichtung ein Unterprogramm erstellen, das diese Aufgabe übernimmt. Dies vereinfacht die Programmierung enorm. Wie du Unterprogramme erstellst, ist im Handbuch der Software ROBO Pro in Kapitel 4 beschrieben. Sobald du dieses Kapitel durchgelesen hast, kannst du dich an die nächste Aufgabe wagen. In ROBO Pro schaltest du jetzt auf **Level 2** um.

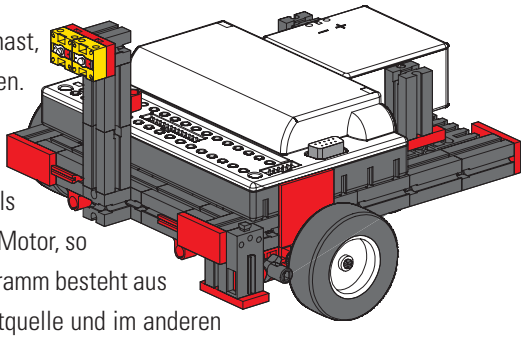
Aufgabe 2 (Level 2):

- **Erstelle für jede Fahrtrichtung ein Unterprogramm.**
- **Programmiere das Modell so, dass es ein Quadrat mit der Kantenlänge von einem Meter abfährt.**
- **Wie groß ist die Wiederholgenauigkeit?**

Tipps:

- Erstelle zunächst ein Unterprogramm **Vorwärts**. Die anderen Unterprogramme kannst du durch Kopieren dieses Unterprogramms erzeugen. Du musst darin dann nur noch die Drehrichtungen der Motoren anpassen.
- Verwende zum Drehen nach links und rechts eine geringere Geschwindigkeit. Das erhöht die Genauigkeit.
- Zum Zählen der Impulse verwendest du wieder das Element **Impulszähler** und den Taster am Eingang I1.
- Lade das Programm zum Ausprobieren erst in den RAM, so lange, bis du herausgefunden hast, wie viele Impulse du benötigst um eine 90°-Drehung auszuführen. Erstens geht das Laden in den RAM schneller als das Laden in den FLASH-Speicher und zweitens hat der Flash-Speicher nur eine „begrenzte" Lebensdauer von ca. 100.000 Downloads.
- Das fertige Programm heißt Basismodell2.rpp.

Der Lichtsucher



■ Nachdem du das Basismodell nun ausgiebig untersucht hast, soll der Roboter nun lernen auf Umweltsignale zu reagieren. Ähnlich wie die Motte aus unserem Gedankenexperiment im ersten Kapitel soll er eine Lichtquelle erkennen und ihr folgen. Der Baukasten enthält 2 Fototransistoren, die wir als Lichtdetektor einsetzen. Jeder Sensor wirkt dabei auf einen Motor, so dass eine Verfolgung der Lichtquelle möglich wird. Das Programm besteht aus zwei Teilen. Der eine Teil enthält die Suche nach einer Lichtquelle und im anderen Teil wird die Verfolgung bzw. das Ansteuern der Lichtquelle realisiert. Dazu werden wieder Unterprogramme verwendet. Nach dem Einschalten wird das Unterprogramm Lichtsuche aktiviert. Dieses Unterprogramm wird erst verlassen, nachdem eine Lichtquelle gefunden wurde. Das Hauptprogramm versucht den Roboter auf die Lichtquelle zu zusteuern. Immer wenn die Richtung des Roboters stark von der Ideallinie abweicht, wird einer der Sensoren nicht mehr von der Lichtquelle bestrahlt. Daraufhin korrigiert der Roboter seine Fahrtrichtung, so dass beide Sensoren die Lichtquelle wieder erkennen können.

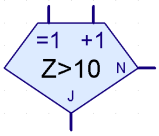
Baue zunächst das Modell Lichtsucher wie in der Bauanleitung beschrieben auf.

Aufgabe 1 (Level 2):

- **Programmiere zuerst die Funktion „Lichtsuche". Der Roboter soll sich dabei langsam um mindestens 360°drehen. Wird während der Suche ein Licht gefunden, stoppt der Roboter. Ansonsten dreht er sich noch einmal um 360° in die andere Richtung. Findet er dann immer noch keine Lichtquelle, soll er 5 Sekunden warten und dann erneut mit der Suche beginnen.**
- **Ist die Lichtsuche erfolgreich, soll das Modell die Lichtquelle ansteuern. Bewegt sich die Lichtquelle nach links oder rechts, soll der Roboter den Bewegungen des Lichts folgen. Verliert er den Kontakt, soll das Programm erneut mit der Lichtsuche beginnen. Probiere aus, ob du den Roboter mit einer Taschenlampe anlocken und durch einen Hindernisparcours führen kannst.**

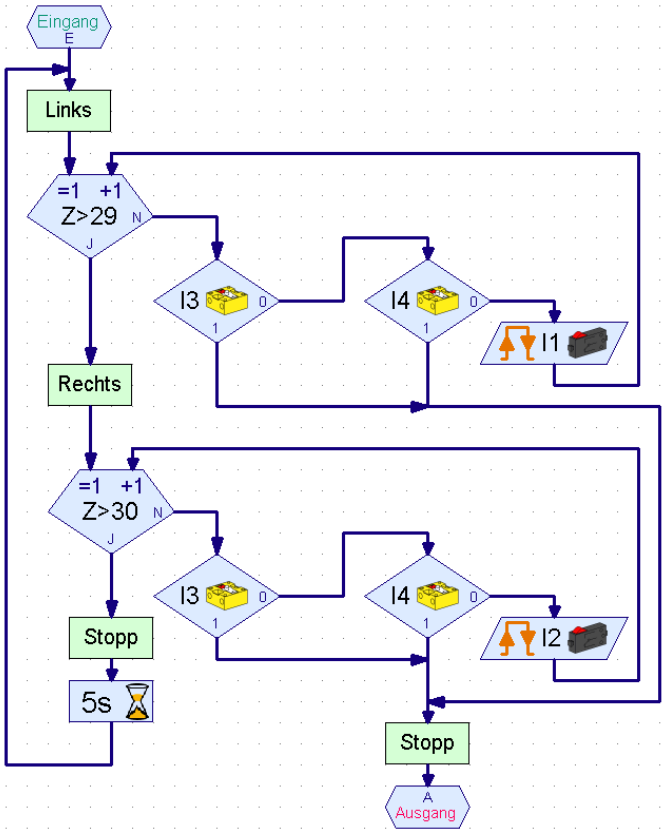
Tipps:

- Verwende für die verschiedenen Fahrtrichtungen die Unterprogramme, die du bereits für das Basismodell programmiert hast. Sobald das Programm Basismodell2.rpp geöffnet ist, findest du im **Elementgruppenfenster** von ROBO Pro unter **Geladene Programme** das Programm Basismodell2 und darunter die Unterprogramme die im Programm Basismodell2 enthalten sind. Diese Unterprogramme kannst du dann einfach in dein neues Programm einfügen.
- Für das Unterprogramm „Lichtsuche" verwendest du das Element **Zählschleife**. (Beschreibung des Elements siehe ROBO Pro Handbuch).



Zählschleife

- In der Schleife zwischen dem Anschluss „N" und Anschluss „+1" fragst du die Fototransistoren ab und zählst einen Impuls am Impulstaster I1. Die Schleife wird so oft durchlaufen, bis der Roboter Licht gefunden hat oder sich um 360° gedreht hat. Wie oft er die Schleife für eine volle Umdrehung durchlaufen muss, probierst du einfach aus und setzt dem entsprechend den Wert „Z" im Element Zählschleife.
- Eine zweite Schleife programmierst du anschließend genau gleich für die Suche mit umgekehrter Drehrichtung.
- Findet der Roboter Licht, stoppt er und verlässt das Unterprogramm.
- Hier das komplette Unterprogramm Lichtsuche:

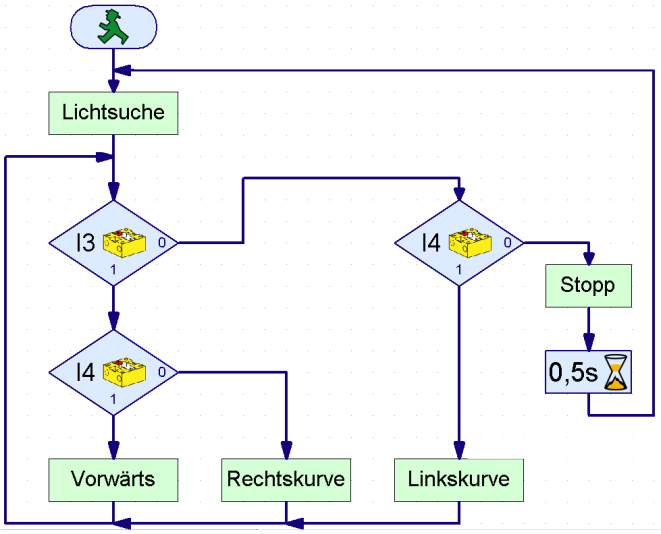


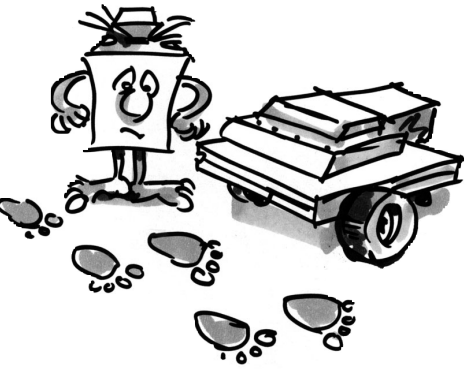
- Im Hauptprogramm fragst du wieder die Fototransistoren ab und steuerst die Motoren abhängig davon, welcher Fototransistor Licht erkennt:

Licht bei I3 und I4	**Vorwärts**
Licht nur bei I3	**Rechtskurve**
Licht nur bei I4	**Linkskurve**
Kein Licht erkannt	**Stopp, zurück zum Unterprogramm Lichtsuche**

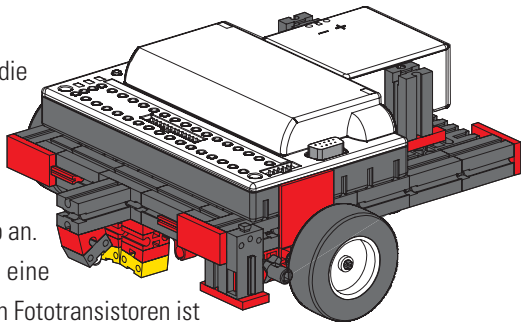
- Die Rechts- und Linkskurve erzeugst du durch unterschiedliche Geschwindigkeiten von M1 und M2 bei gleicher Drehrichtung. Dadurch ergibt sich ein sehr harmonischer Fahrstil.

- Das Hauptprogramm sieht dann wie folgt aus:
- Das fertige Programm findest du unter Lichtsucher.rpp.
- Benutze als Lichtquelle eine Taschenlampe. Versuche dabei den Lichtstrahl nicht zu klein zu fokussieren, damit beide Fotosensoren von der Lichtquelle bestrahlt werden. Beachte, dass in sehr hellen Räumen deine Taschenlampe von anderen Lichtquellen, z. B. Sonnenlicht von einem großen Fenster, überstrahlt werden. Der Roboter fährt dann unter Umständen an deiner Lampe vorbei auf das hellere Licht zu.





Der Spurensucher



■ Suche und Verfolgung sind wesentliche Eigenschaften, die intelligente Wesen besitzen. Mit dem Lichtsucher hast du einen Roboter gebaut und programmiert, der auf direkte Signale von seinem Ziel reagiert hat.
Mit dem Spurensucher wenden wir ein anderes Suchprinzip an. Anstelle der zielgenauen Fahrt zur Lichtquelle wird eine schwarze Linie markiert, der ein Roboter folgen soll. Mit den Fototransistoren ist diese Aufgabe relativ leicht zu lösen. Das reflektierte Licht der Markierung wird gemessen und danach die Motoren korrigiert. Damit das auch exakt funktioniert, wird die Linie mit der Lampe beleuchtet. Achte darauf, dass nicht durch eine un-günstige Anordnung die Fotosensoren vom Streulicht der Lampe geblendet werden. Besonders günstig wirkt sich in diesem Zusammenhang die Lichtbündelung der optischen Linse der Glühlampe aus.

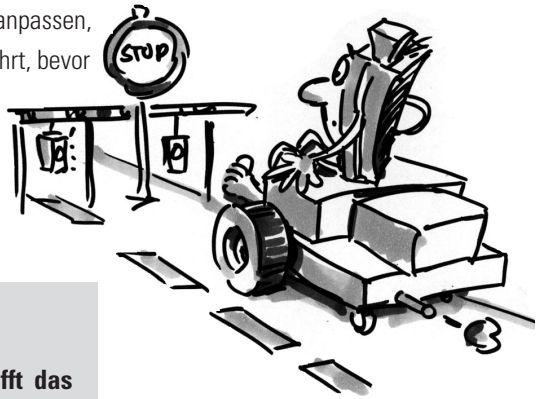
Baue nun das Modell Spurensucher gemäß der Bauanleitung auf.

Aufgabe 1:

- **Schreibe als Erstes ein Unterprogramm, mit dem die Spur gesucht wird. Der Roboter soll sich dazu einmal im Kreis drehen.**
- **Findet er keine Spur, soll er ein Stück geradeaus fahren und dann erneut suchen. Zur Spurerkennung werden die Fototransistoren abgefragt.**
- **Hat der Roboter die Spur entdeckt, soll er ihr folgen.**
- **Ist die Spur zu Ende oder verliert der Roboter sie, z. B. wegen einer starken Richtungsänderung der Spur, soll die Suche erneut beginnen.**

Tipps:

- Nach dem Einschalten der Lampe muss eine kurze Zeit gewartet werden (ca. eine Sekunde) bevor die Fototransistoren abgefragt werden. Sonst erkennt der Fototransistor „dunkel", das heißt eine Spur, wo keine ist, weil die Abfrage erfolgt, bevor die Lampe richtig hell ist.
- Als Spur verwendest du ca. 20mm breites schwarzes Isolierband, oder du malst mit Filzstift eine schwarze Spur in dieser Breite auf ein weißes Blatt Papier. Die Kurven dürfen nicht zu eng sein, sonst verliert der Roboter zu oft die Spur.
- Überprüfe zuerst mit dem Interfacetest, ob deine Spur von den Fototransistoren richtig erkannt wird. Vergiss dabei nicht die Lampe einzuschalten.
- Justiere die Lampe so, dass auf hellem Untergrund beide Fototransistoren den Wert 1 liefern, auch wenn die Motoren M1 und M2 eingeschaltet werden. Ist dein Akku etwas schwach, wird beim Einschalten der Motoren die Lampe etwas dunkler. Ist sie nicht richtig justiert, kann es sein, dass ein Fototransistor „dunkel" anzeigt obwohl er gar keine Spur gefunden hat.
- Die Spursuche funktioniert ähnlich wie die Lichtsuche. Du musst lediglich die Suche so anpassen, dass das Modell bei erfolgloser Suche nach einer vollen Umdrehung ein Stück geradeaus fährt, bevor es weiter sucht.
- Beachte, dass das Modell bei der Spurverfolgung geradeaus fahren soll, wenn beide Fototransistoren den Wert „dunkel" (=0) liefern.
- Das fertige Programm findest du unter Spurensucher.rpp.

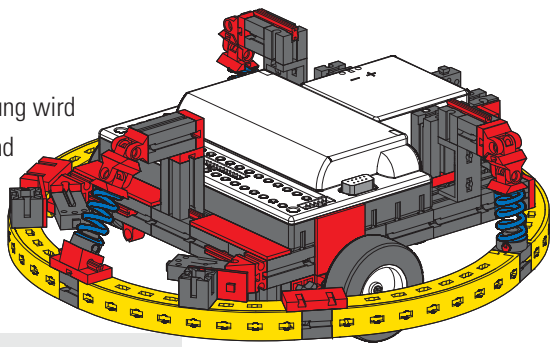


Aufgabe 2:

- **Erzeuge eine Spur mit verschieden engen Kurven. Welchen Radius schafft das Modell gerade noch?**
- **Experimentiere beim Korrigieren der Spur mit verschiedenen Geschwindigkeiten von M1 und M2. Welche Kombination liefert das beste Ergebnis?**
- **Erzeuge als Spur einen Rundkurs. Versuche die Geschwindigkeiten so zu optimieren, dass der Roboter eine möglichst schnelle Rundenzeit schafft. Diese Aufgabe eignet sich bestens für einen Wettbewerb mit mehreren Robotern.**

Roboter mit Hinderniserkennung

■ Alle bisher gebauten Roboter können eine bestimmte Wegstrecke zurücklegen sowie einer Lichtquelle oder einer Spur folgen. Doch was passiert, wenn sich ihnen ein Hindernis in den Weg stellt? Nun, entweder das Hindernis wird beiseite geschoben oder der Roboter rennt sinnlos dagegen an bis der Akku leer ist. Viel intelligenter wäre es natürlich, der Roboter würde das Hindernis erkennen und entsprechend ausweichen. Dazu erhält der Roboter eine bewegliche Rundum-Stoßstange mit drei Tastern. Mit dieser Stoßstange kann er unterscheiden, ob sich ein Hindernis links, rechts oder hinter ihm befindet.Wie er darauf reagieren soll, ist dann nur noch eine Frage der Programmierung.



Baue zunächst das Modell „Roboter mit Hinderniserkennung" auf. Für die Wegmessung wird nur einen Taster (I1) benötigt. Deshalb wird vom Basismodell der Taster I2 entfernt und für die Hinderniserkennung verwendet.

Aufgabe 1 (Level 2):

- **Der Roboter soll zunächst geradeaus fahren. Stößt er links an ein Hindernis (I4), soll er ein Stück zurück und dann nach rechts ausweichen.**
- **Stößt er rechts an ein Hindernis (I3), soll er ein Stück zurück und dann nach links ausweichen.**

Tipps:

- Die Hinderniserkennung beim Rückwärtsfahren wird momentan noch nicht betrachtet.
- Hauptprogramm werden die Taster abgefragt. Je nachdem welcher Taster betätigt wird, weicht das Modell nach links oder rechts aus. Dies geschieht jeweils in einem Unterprogramm.
- Die Impulszahl bei der Rechtsdrehung sollte sich von der Impulszahl bei der Linksdrehung unterscheiden (z. B. 3 Impulse nach rechts, 5 Impulse nach links). Sonst kann es sein, dass das Modell in eine Ecke fährt und nicht mehr heraus findet, weil es sich immer gleich weit nach links und rechts dreht.
- Das fertige Programm heißt Hindernis1.rpp.

Zwei Dinge kann der Hinderniserkenner noch nicht: Beim Rückwärtsfahren erkennt er noch keine Hindernisse. Er merkt auch noch nicht, wenn sich ein Hindernis direkt vor ihm befindet. Beides könnte er aber erkennen. Wird während der Rückwärtsfahrt I5 gedrückt, ist ein Hindernis hinter dem Modell. Werden beim Vorwärtsfahren I3 und I4 gleichzeitig betätigt, befindet sich ein Hindernis direkt vor dem Modell. In diesem Fall könnte sich der Roboter gleich um 90° drehen. Insgesamt sind jetzt also folgende Möglichkeiten vorhanden, auf die der Roboter reagieren soll:

Hindernis	Taster	Reaktion
rechts	Nur I3	Nach links ausweichen (ca. 30° Drehung)
links	Nur I4	Nach rechts ausweichen (ca. 45°)
vorne	I3 und I4	Nach links ausweichen (ca. 90°)
hinten	I5	Wird nur bei Rückfahrtsfahren abgefragt. Anhalten, danach wie geplant weiter ausweichen

Um diese Aufgabe elegant zu lösen, können dir einige neue Programmelemente wie z. B. Operatoren (z. B. UND, ODER) aus ROBO Pro Level 3 sehr gut helfen. In Level 3 gibt es auch die Möglichkeit über orange Pfeile Daten zwischen verschiedenen Elementen auszutauschen. Schalte deshalb in der Software auf diesen Level um. Danach nimmst du am Besten das ROBO Pro Handbuch und liest das Kapitel 5 aufmerksam durch. Danach bist du fit für die nächste Aufgabe.

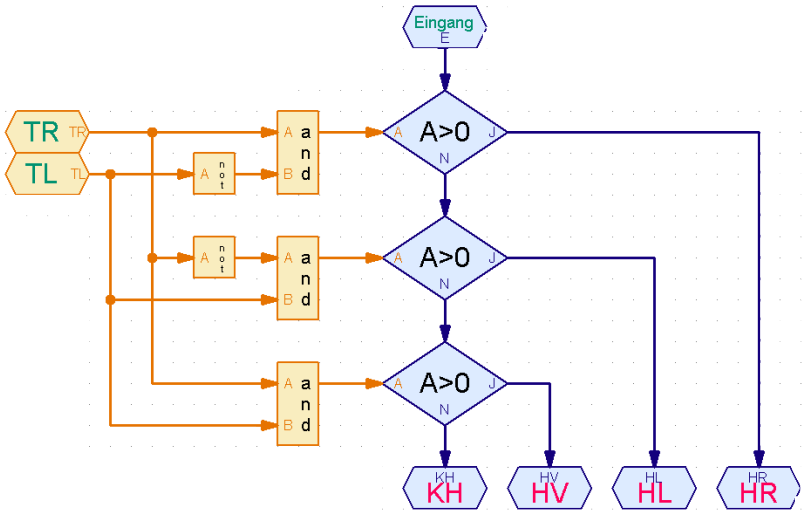
Aufgabe 2 (Level 3):

- **Gestalte dein Hindernisprogramm so um, dass das Modell wie in der oben stehenden Tabelle reagiert.**
- **Verwende dazu die Möglichkeiten aus ROBO Pro Level 3.**

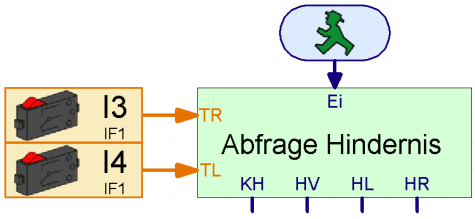
Tipps:

- Mit Hilfe von Operatoren werden in einem Unterprogramm „Abfrage Hindernis“ die verschiedenen möglichen Tasterkombinationen abgefragt. Für jede Möglichkeit besitzt das Unterprogramm einen eigenen Ausgang.

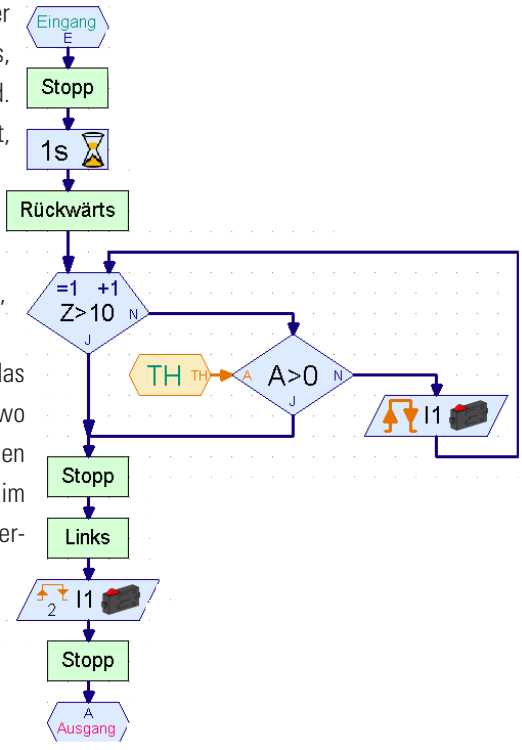
Dateneingang TR = Taster rechts
Dateneingang TL = Taster links
Ausgang KH = kein Hindernis
Ausgang HV = Hindernis vorne
Ausgang HL = Hindernis links
Ausgang HR = Hindernis rechts



- Damit sofort erkennbar ist, welche Taster abgefragt werden, platzierst du die orangen Tasterelemente im Hauptprogramm und verbindest sie über Dateneingänge mit dem Unterprogramm.



- In den verschiedenen Ausweich-Unterprogrammen wird während der Rückwärtsfahrt I5 abgefragt. Das Modell fährt dann so lange rückwärts, bis entweder die eingestellte Impulszahl erreicht oder I5 gedrückt wird. I5 wird wieder im Hauptprogramm platziert, damit sofort ersichtlich ist, in welchen Unterprogrammen er abgefragt wird.
- Das komplette Programm findest du unter Hindernis2.rpp.



Ein Vorteil der in dieser Aufgabe angewandten Programmiertechnik ist, dass du direkt im Hauptprogramm siehst, welcher Taster in welchem Unterprogramm abgefragt wird. Willst du den Eingang ändern, musst du das nur an einer Stelle tun und nicht in sämtlichen Unterprogrammen suchen, wo sich der Taster überall versteckt haben könnte. Außerdem kann man mit den Operatoren sehr anschaulich logische Verknüpfungen erstellen. Das geht im Prinzip zwar auch mit Verzweigungselementen, wird aber schnell unübersichtlich, wenn mehrere Fälle abgefragt werden.

Lichtsucher mit Hinderniserkennung

■ Es ist noch lange nicht Schluss mit den Möglichkeiten, die das ROBO Mobile Set bietet. Deshalb sollen jetzt die beiden Funktionen Lichtsuche und Hinderniserkennung kombiniert werden. Aus wissenschaftlicher Sicht ist der Roboter dann mit zwei Verhaltensweisen ausgestattet. Da jedoch nicht beide Verhaltensmuster gleichzeitig aktiv sein können, erhalten sie unterschiedliche Prioritäten. Der Roboter ist normalerweise auf Lichtsuche. Erkennt er ein Hindernis, also eine Gefahr für ihn, wird das Verhalten Hindernisvermeidung aktiv. Ist alles im grünen Bereich, kann der Roboter weiter nach der Lichtquelle forschen.

Wenn professionelle Softwareentwickler sich an so eine anspruchsvolle Aufgabe machen, programmieren sie nicht einfach wild drauf los, sondern sie verwenden eine bestimmte Strategie für die Entwicklung des Programms. Eine dieser Methoden nennt man „Top-Down-Entwurf". Bei dieser Vorgehensweise wird das Gesamtsystem von oben herab definiert, ohne dass man sich am Anfang mit allen Details befasst. Dieser Methode bedienen wir uns bei diesem Problem ebenfalls.

Aufgabe 1 (Level 3):

Bringe dem Roboter folgende Verhaltensweisen bei:

- **Suche nach einer Lichtquelle.**
- **Sobald du sie gefunden hast, folge ihr.**
- **Taucht unterwegs ein Hindernis auf, weiche ihm aus.**
- **Suche danach erneut nach einer Lichtquelle**

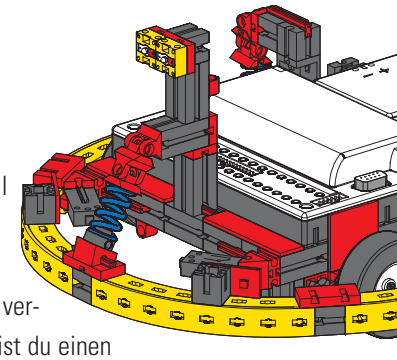
Verwende zur Lösung die Programmelemente aus ROBO Pro Level 3.

Löse die Aufgabe „von oben herab" nach dem Top-Down-Verfahren.

Tipps:

Du gliederst die Aufgabe zunächst in drei Teile:

- Abfragen ob der Roboter eine Lichtquelle sieht (Unterprogramm „Licht")
- Abfragen ob er an ein Hindernis stößt (Unterprogramm „Hindernis")
- Abhängig von diesen Ergebnissen teilst du dem Roboter mit, was er tun soll (Unterprogramm „Fahren")



Für die Unterprogramme „Licht" und „Hindernis" überlegst du nun, welche verschiedenen Situationen der Roboter wahrnehmen kann. Jeder Situation weist du einen Zahlenwert zu, den du mit Hilfe eines Befehlselements in einer Variablen speicherst. Aus jeder Situation resultiert dann eine Reaktion, die im Unterprogramm „Fahren" ausgeführt wird.

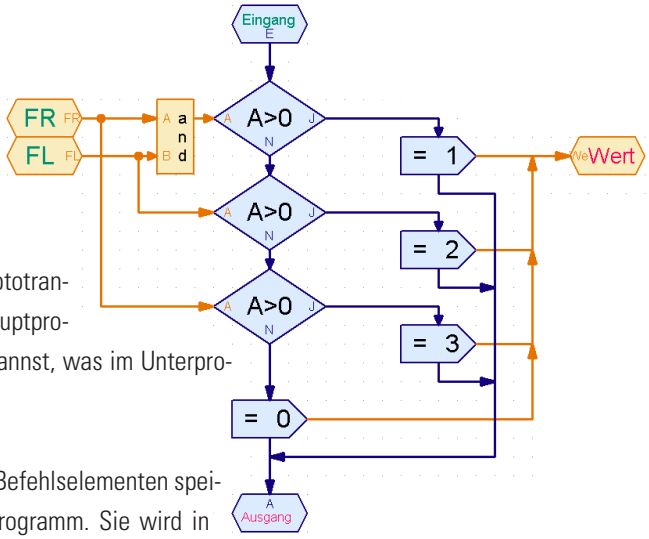
Unterprogramm Licht:

Nr.	Situation	Zustand der Sensoren	Reaktion
0	Keine Lichtquelle vorhanden	I6=0; I7=0	Licht suchen
1	Lichtquelle genau vor Roboter	I6=1; I7=1	Geradeaus fahren
2	Lichtquelle links vom Roboter	I7=1	Linkskurve fahren
3	Lichtquelle rechts vom Roboter	I6=1	Rechtskurve fahren

Unterprogramm Hindernis:			
Nr.	**Situation**	**Zustand der Sensoren**	**Reaktion**
4	**Hindernis direkt vor Roboter**	**I3=1; I4=1**	**90° ausweichen**
5	**Hindernis rechts vom Roboter**	**I3=1**	**Nach links ausweichen**
6	**Hindernis links vom Roboter**	**I4=1**	**Nach rechts ausweichen**

Jetzt musst du nur noch diese Erkenntnisse in ROBO Pro mit Programmelementen darstellen.

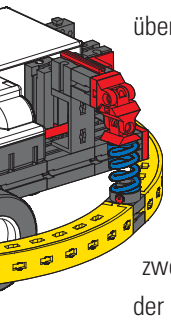
Unterprogramm Licht:



FR=Fototransistor rechts
FL=Fototransistor links

Die Elemente für die Abfrage der Fototransistoren platzierst du wieder im Hauptprogramm, damit du sofort erkennen kannst, was im Unterprogramm abgefragt wird.

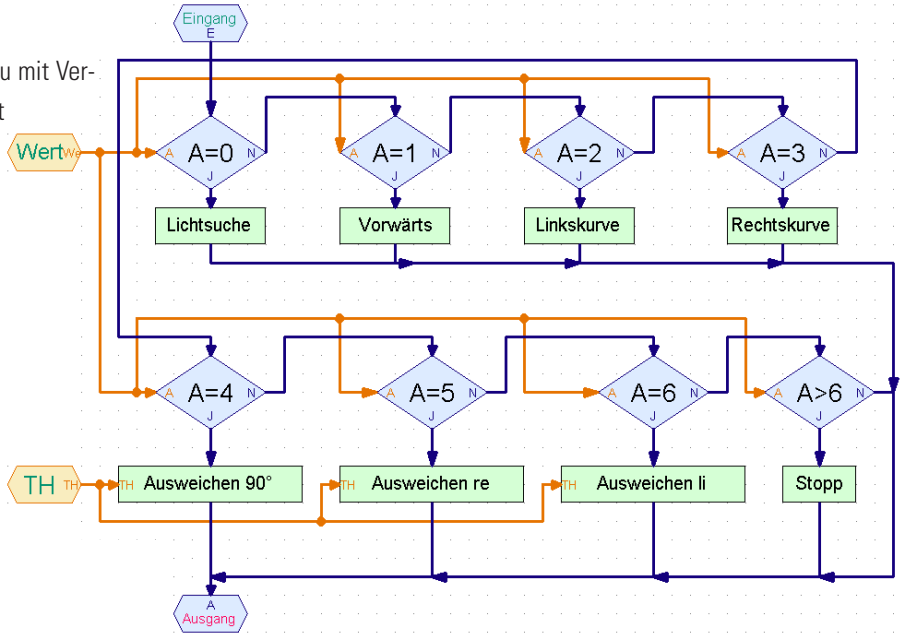
Die Variable, die den Wert aus den Befehlselementen speichert, kommt ebenfalls ins Hauptprogramm. Sie wird in mehreren Unterprogrammen verwendet. Du verbindest sie über einen Datenausgang mit dem Unterprogramm.



Das Unterprogramm „Hindernis“ erstellst du nach dem selben Prinzip wie das Unterprogramm „Licht“.

Im Unterprogramm „Fahren“ fragst du mit Verzweigungselementen den aktuellen Wert der Variablen ab und programmierst die entsprechende Reaktion:

TH=Taster hinten



Als letztes Detail musst du jetzt noch die Unterprogramme erstellen, die in diesem Unterprogramm verwendet werden.
Aber Moment mal! Die gibt's doch fast alle schon. Das Unterprogramm Lichtsuche z. B. kannst du aus dem Programm für das Modell Lichtsucher kopieren. Wenn du nicht mehr weißt wie das geht, lies im ROBO Pro Handbuch Im Kapitel 4 nach.

Aber Vorsicht:
Bei dem Modell Lichtsucher waren die Fototransistoren am Eingang I3 und I4 angeschlossen. Jetzt liegen sie aber auf I6 und I7. Außerdem wurde dort für das Zählen der Impulse beim Drehen nach links der Taster I1 und beim Drehen nach rechts I2 abgefragt. Jetzt gibt es nur noch I1 zum Zählen der Impulse, was übrigens genauso gut funktioniert. Du musst das Unterprogramm Lichtsuche nach dem Kopieren also anpassen. Da die Abfrage der Taster im Unterprogramm versteckt ist, ist das leicht zu übersehen. Das passiert nicht mehr, wenn du die Eingänge im Hauptprogramm platzierst und über Dateneingänge mit dem Unterprogramm verbindest. Aber beim Lichtsucher kanntest du diese Möglichkeit ja noch nicht.

Die Unterprogramme zum Ausweichen sind auch schon vorhanden, nämlich beim Modell Hinderniserkennung. Hier ist sogar schon der Taster I5, der zusätzlich beim Rückwärtsfahren abgefragt wird, nach außen geführt.

Das fertige Programm kannst du dir anschauen unter Hindernis-Licht.rpp.
Das Hauptprogramm sieht auf den ersten Blick doch sehr übersichtlich und einfach aus. Und doch steckt jede Menge Gehirnschmalz in den Unterprogrammen. Aber mit Hilfe des schrittweisen Vorgehens mit der Top-Down-Methode konntest du auch so ein komplexes Problem lösen.

Übrigens, wenn du einen Freund hast, der ebenfalls einen Baukasten ROBO Mobile Set besitzt, könnt ihr das Experimentieren mit diesem Roboter noch weiter treiben. An jeden der beiden Roboter wird einfach eine Lichtquelle montiert. Dann suchen sich die Roboter gegenseitig.

Roboter mit Kantenerkennung

■ Nachdem du im letzten Beispiel gesehen hast, wie man bei der Programmierung eines komplexeren Problems vorgeht, kannst du dich nun einem weiteren sehr wichtigen Verhalten eines mobilen Roboters zuwenden. Er soll nämlich lernen, nicht vom Tisch zu fallen. Fährt der Roboter gegen ein Hindernis, macht ihm das in den meisten Fällen nichts aus. Fällt er aber von einem Tisch fast einen Meter in die Tiefe, könnte er schon den einen oder anderen Schaden davon tragen, obwohl die fischertechnik Bausteine ja sehr stabil sind. Aus diesem Grund bekommt der Roboter Sensoren, mit denen er Kanten erkennen kann. Diese Kantendetektoren bestehen jeweils aus einem Taster, der von einem drehbar gelagerten Rad betätigt wird. Dieses Rad kann sich auch auf und ab bewegen. Sobald sich das Rad über die Tischkante hinaus bewegt, fällt es nach unten, der Taster wird nicht mehr betätigt, das Programm erkennt, dass sich das Modell an einem Abgrund befindet und reagiert entsprechend darauf. Der Roboter hat insgesamt 4 Kantendetektoren, so dass er sowohl beim Vorwärts- als auch beim Rückwärtsfahren auf beiden Seiten nach Abgründen tasten kann. Dafür hat dieses Modell keine Impulstaster für die Wegmessung. Der zurückgelegte Weg wird über die Einschaltdauer der Motoren gesteuert.
Baue zunächst das Modell wie in der Bauanleitung beschrieben auf.

Kontrolliere genau, ob die Kantendetektoren richtig auslösen:

- wenn das Modell an die Tischkante kommt und der Taster wieder exakt gedrückt wird,
- wenn das Rad wieder auf dem Tisch steht.

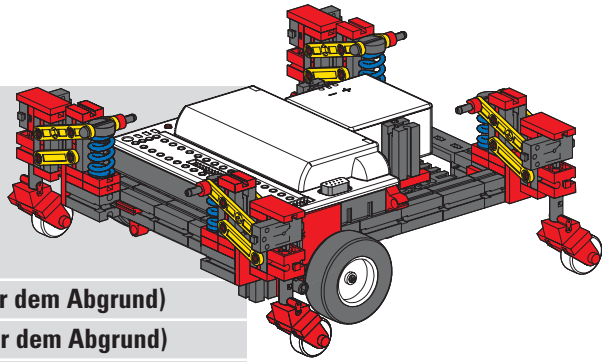
Eventuell musst du den einen oder anderen Taster etwas nach oben oder unten justieren.

Aufgabe 1 (Level 3):

- **Überlege zunächst einmal, wie der Roboter reagieren soll, wenn er an einen Abgrund kommt.**
- **Bei genauerem Nachdenken wird dir auffallen, dass es ziemlich viele Kombinationsmöglichkeiten gibt, welche Sensoren sich über dem Abgrund befinden können. Es kann einer der 4 Detektoren ausgelöst werden, gleichzeitig 2 oder 3 verschiedene oder auch alle 4 Sensoren.**
- **Wie soll der Roboter jedes Mal darauf reagieren?**

Tipps:

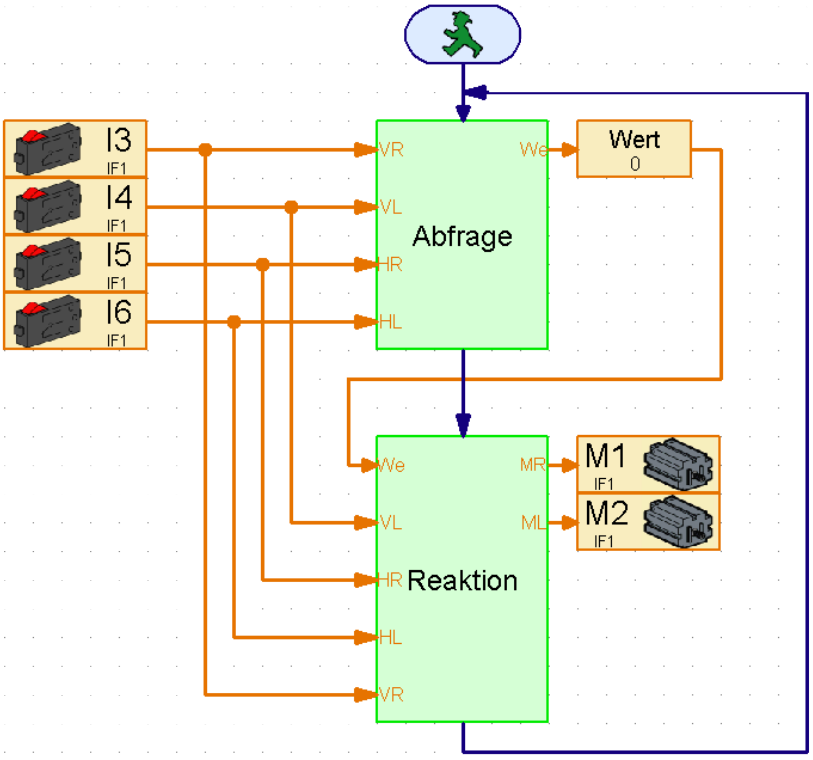
Die Lösung findest du in der folgenden Tabelle. Die Sensoren, die sich über dem Abgrund befinden (Taster=0) sind markiert. Jede Kombination erhält eine Nummer. Im Programm, das später erstellt wird, wird jeder Möglichkeit die entsprechende Zahl zugewiesen. Anhand der Zahl reagiert der Roboter auf die vorliegende Situation. Doch dazu später mehr. Überlege zunächst nur, wie der Roboter steht, damit die jeweilige Kombination auftritt, und ob er richtig reagiert.



Nr.	Vorne rechts (I3)	Vorne links (I4)	Hinten rechts (I5)	Hinten links (I6)	Reaktion
0					Vorwärts (Kein Sensor über dem Abgrund)
1	●	●	●	●	Stopp (alle 4 Sensoren über dem Abgrund)
2	●	●	●		Ein wenig nach rechts drehen
3	●	●		●	Ein wenig nach links drehen
4	●		●	●	Ein wenig nach links drehen
5		●	●	●	Ein wenig nach rechts drehen
6	●	●			Erst zurück, dann nach rechts drehen
7	●		●		Ein wenig nach links drehen
8	●			●	Ein wenig nach links drehen
9		●	●		Ein wenig nach rechts drehen
10		●		●	Ein wenig nach rechts drehen
11			●	●	Ein wenig nach vorne fahren
12	●				Erst zurück, dann nach links drehen
13		●			Erst zurück, dann nach rechts drehen
14			●		Ein wenig nach vorne fahren
15				●	Ein wenig nach vorne fahren

Ganz schön heftig, was? Aber keine Angst, für dieses Modell gibt es ein fertiges Programm, das alle Vorzüge von ROBO Pro nutzt. Es heißt Kanten.rpp.

Die wichtigsten Elemente befinden sich im Hauptprogramm, so dass du den Gesamtablauf verstehen kannst. Die komplexe Abfrage der Taster und die Ansteuerung der Motoren sind in Unterprogrammen versteckt. Hier zunächst das Hauptprogramm:



Der Ablauf beginnt mit der Abfrage der 4 Taster. Ganz links siehst du, welche Taster abgefragt werden. Sie sind über Dateneingänge mit dem Unterprogramm verbunden. Das Unterprogramm „Abfrage" ermittelt, welche Taster gedrückt sind und vergibt den in der Tabelle beschriebenen Wert. Dieser Wert wird der gleichnamigen Variablen zugewiesen, die du wieder im Hauptprogramm erkennen kannst. Der Variablenwert wird an das Unterprogramm „Reaktion" weitergeleitet, das dann abhängig von diesem Wert die beiden Motoren ansteuert. Im Unterprogramm „Reaktion" werden ebenfalls noch die Taster eingelesen, da die Kantensensoren auch abgefragt werden, während das Modell ausweicht.

Du könntest jetzt ohne weiteres die Tasterbelegung am Interface ändern, ebenso die Motorausgänge, ohne dass du alle Unterprogramme durchforsten müsstest, wo sich noch irgendwo ein Eingangselement oder ein Motorsymbol versteckt hat. Jeder Eingang und jeder Ausgang kommt nur einmal vor.

Diese Programmiertechnik kannst du vor allem dann anwenden, wenn ein Unterprogramm in vielen verschiedenen Modellen angewendet werden soll und du vorher noch nicht genau weißt, welche Ein- und Ausgänge am Interface dafür verwendet werden sollen.

Wenn du jetzt neugierig geworden bist, schau einfach mal in die Unterprogramme hinein und versuche sie zu verstehen. Das Prinzip der Programmierung ist ähnlich wie beim Modell „Lichtsucher mit Hinderniserkennung".

Aufgabe 2 (Level 3):

Lade das Programm auf das Interface und lasse das Modell auf einem Tisch fahren.

- **Reagiert das Modell immer richtig?**
- **Sollte es sich bei bestimmten Tastenkombinationen anders verhalten?**
- **Optimiere bei Bedarf das Programm.**

Der Laufroboter

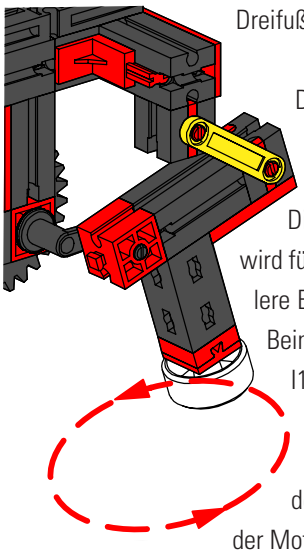
■ Nachdem wir uns ausführlich mit den fahrbaren Robotern beschäftigt haben, wenden wir uns nun einer anderen Fortbewegungsart zu, die wir für mobile Roboter nutzen können, dem Laufen.
Die Gangart der Insekten eignet sich hervorragend als Vorbild für den Antrieb von „maschinellen Sechsbeinern". Beim sogenannten Dreifußgang heben immer drei der sechs Beine gleichzeitig vom Boden ab, das vordere und hintere Bein der einen Seite zusammen mit dem mittleren Bein der anderen Seite:

Dreifußgang

Die Beine, die auf dem Boden stehen (schwarz dargestellt), bilden ein stabiles Dreibein, so dass das Modell immer sicher steht und beim Laufen nicht umkippt.

Die Beine des fischertechnik-Laufroboters sind so konstruiert, dass sie ein so genanntes Viergelenkgetriebe ergeben. Die Bauform des hier verwendeten Viergelenks nennt man „Kurbelschwinge". Angetrieben von einer Kurbel führen die beweglich gelagerten Glieder des Getriebes schwingende Bewegungen aus. Die Abstände zwischen den einzelnen Gelenken und die Lage des Fußpunktes (das ist das untere Ende des Beins), sind so gewählt, dass der Fußpunkt eine elliptische Bewegung beschreibt, wenn sich die Antriebskurbel dreht. Dadurch entsteht eine Bewegung, die einem Schritt beim Laufen ähnelt.

Die 6 Kurbeln, die die Beine antreiben, müssen genau so justiert werden, wie in der Bauanleitung gezeigt. Die drei Beine, die gleichzeitig auf dem Boden aufsetzen, haben die gleiche Kurbelstellung. Die Kurbeln der 3 Beine, die zu diesem Zeitpunkt in der Luft stehen, sind dazu um 180° verdreht. Die richtige Stellung der Kurbeln zueinander gewährleistet, dass das Modell in der richtigen Schrittfolge, dem Dreifußgang, laufen kann.



Die Nabenmuttern, mit denen man die Zahnräder auf den Achsen fixiert, müssen gut festgedreht werden, damit sich die Kurbeln während des Laufens nicht verstellen.

Die rechte und linke Seite des Modells werden von je einem Motor angetrieben (das wird für das Kurvenlaufen benötigt). Deshalb muss dafür gesorgt werden, dass sich das mittlere Bein der einen Seite immer in der gleichen Stellung befindet wie die beiden äußeren Beine der anderen Seite. Diese Synchronisation erfolgt softwaregesteuert über die Taster I1 und I2.

Baue zunächst das Modell wie in der Bauanleitung beschrieben auf. Kontrolliere mit dem Interfacetest, ob alle Taster und Motoren richtig angeschlossen sind. Drehrichtung der Motoren: linke Drehrichtung=vorwärts.

Aufgabe 1 (Level 1):
Bringe dem Roboter das Laufen bei.

- **Programmiere das Modell so, dass es im Dreifußgang geradeaus läuft.**
- **Benutze die Taster I1 und I2 zur Synchronisation der linken und rechten Beine.**
- **Beachte dabei, dass immer die beiden äußeren Beine der einen Seite und das mittlere Bein der anderen Seite die gleiche Stellung haben.**

Tipps:

- Bringe zunächst die Beine der linken und rechten Seite in ihre Ausgangsposition. Schalte dazu beide Motoren ein (Drehrichtung links).
- Der Ablauf soll erst weitergehen, wenn beide Taster I1 und I2 nicht gedrückt sind (diese Abfrage wird notwendig, sobald das Modell den zweiten Schritt machen soll).
- Lasse die Motoren so lange laufen, bis der jeweilige Taster (I1 für M1, I2 für M2) wieder gedrückt ist. Wichtig dabei ist, dass das Modell den nächsten Schritt erst beginnt, wenn beide Taster gedrückt sind. Dann stehen nämlich die Beine in der richtigen Stellung zueinander. Voraussetzung dafür ist aber auch, dass die Kurbeln, die die Beine antreiben, richtig justiert sind, so wie es in der Bauanleitung beschrieben ist.
- Jetzt kann der Ablauf wieder von vorne beginnen und der Roboter macht den zweiten Schritt. Nun läuft das Modell so lange geradeaus, bis du das Programm stoppst.
- Das fertige Programm findest du unter Laufroboter1.rpp.

Ähnlich wie bei dem fahrbaren Basismodell kannst du jetzt durch Verändern der Motordrehrichtungen das Modell nach links, rechts oder zurück laufen lassen. Für das Zählen der Schritte kannst du I1 oder I2 verwenden.

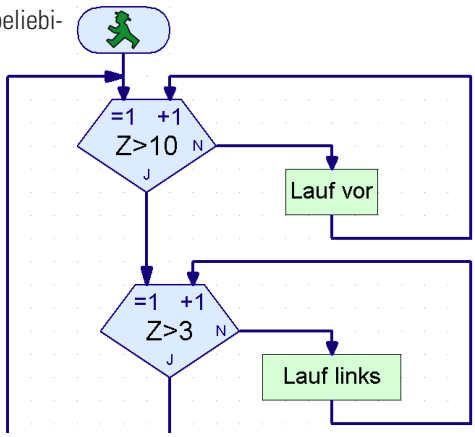
Aufgabe 2 (Level 2):

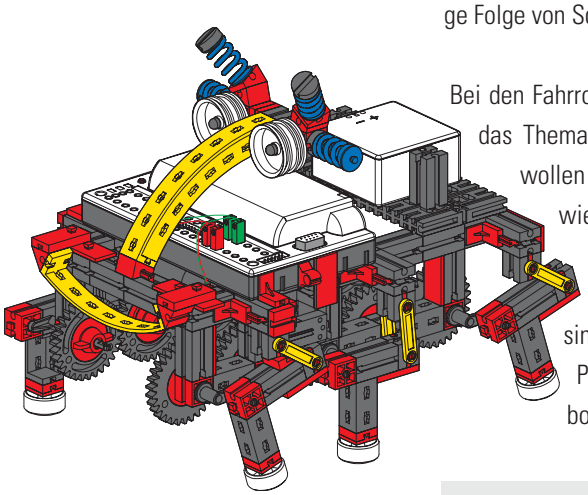
- **Programmiere dein Modell so, dass es 10 Schritte nach vorne, 3 Schritte nach links, 3 Schritte nach rechts und wieder 10 Schritte zurück läuft.**
- **Lege für jede Richtung ein eigenes Unterprogramm an.**
- **Verwende zum Zählen der Schritte das Element Zählschleife.**

Tipps:

- Kopiere einfach das Programm Laufroboter1.rpp in ein Unterprogramm.
- Kopiere dieses Unterprogramm so oft, wie es für die verschiedenen Laufrichtungen notwendig ist. Ändere in jedem Unterprogramm die Motordrehrichtungen so, dass sich das Modell in die gewünschte Richtung bewegt.
- Verwende das Element Zählschleife um die Anzahl der Schritte für jede Drehrichtung zu zählen. Das Modell macht bei jedem Durchlauf eines Unterprogramms einen Schritt. Durchläuft das Programm die Schleife mit dem Unterprogramm 10 mal, macht das Modell 10 Schritte.

Auf diese Weise kannst du deinem Laufroboter eine beliebige Folge von Schritten beibringen (Laufroboter2.rpp).





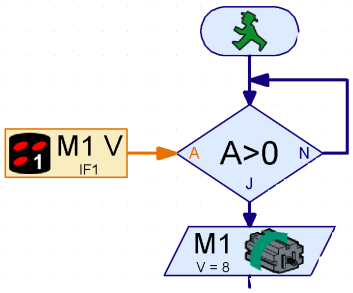
Bei den Fahrrobotern haben wir bereits ausführlich das Thema Hinderniserkennung behandelt. Wir wollen es an dieser Stelle nicht noch einmal wiederholen. Aber versuche doch einmal dieses Verhalten auf den Laufroboter zu übertragen. Die Sensoren dafür sind im Baukasten enthalten. Bei der Programmierung kannst du den Fahrroboter als Vorbild nehmen. Viel Erfolg!

Erweiterungs-möglichkeiten

■ Das ROBO Interface bietet mehr Funktionalität als bisher bei den mobilen Robotern gezeigt. Dafür werden aber zusätzliche Komponenten benötigt, die nicht im Lieferumfang des Baukastens enthalten sind. Da sie aber für die Roboter äußerst interessant sind, wollen wir einige davon an dieser Stelle kurz vorstellen.

Infrarot Handsender

■ Im ROBO Interface ist eine Infrarot Empfängerdiode für den Handsender aus dem IR Control Set Art.-Nr. 30344 enthalten. In der Software ROBO Pro kannst du damit die Tasten des Handsenders wie digitale Eingänge abfragen und damit z. B. Motoren ein- und ausschalten.



Als Programmbeispiel haben wir eine Fernsteuerung für den Laufroboter programmiert. Mit den 4 ovalen Pfeiltasten an der Fernbedienung kannst du das Modell vorwärts, rückwärts nach links und rechts steuern. Vorher musst du nur das Programm Laufroboter-IR.rpp auf das Interface laden.

Ein weiteres geniales Programm in Verbindung mit der Fernbedienung ist das Programm Mobile-Teach-IR.rpp. Mit diesem Teach-In-Programm kannst du einen der Fahrroboter, z. B. den einfachen Roboter oder das Basismodell, fernsteuern. Das Modell merkt sich dabei die gefahrene Strecke und kann sie danach beliebig oft wiederholen. Die gespeicherte Strecke wird allerdings gelöscht, wenn das Programm gestoppt wird.
Möglich wird so ein Programm durch das Programmelement „Liste" in ROBO Pro. In diesem Element kann man viele Werte speichern und wieder auslesen (siehe auch ROBO Pro Handbuch). Das Programm selbst ist zwar ziemlich komplex, die Anwendung ist aber ganz einfach:

1. Programm Mobile-Teach-In.rpp in den Flashspeicher des ROBO Interface laden und starten.
2. An der Fernbedienung Taste **M1** ▸/▸▸ drücken. Der „Lernvorgang" wird gestartet.
3. Mit den ovalen Pfeiltasten das Modell in die gewünschte Richtung steuern.
4. Taste **M2** ▸/▸▸ drücken. Die abgefahrene Strecke wird gespeichert.
5. Taste **M3** ▸/▸▸ drücken. Die gespeicherte Strecke wird abgefahren.

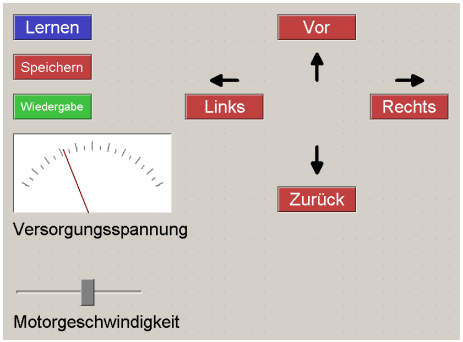
Mit so einer Anwendung wird das Programmieren von Robotern zum Kinderspiel! Du musst beachten, dass die gespeicherte Strecke gelöscht wird, sobald du das Programm mit dem Prog.-Taster am Interface stoppst.

ROBO RF Data Link

■ Die Funkschnittstelle ROBO RF Data Link Art.-Nr. 93295 ersetzt das Schnittstellenkabel zwischen PC und Interface durch eine Funk-Datenübertragung. Das ist eine feine Sache. Erstens musst du nicht jedes Mal, wenn du ein Programm auf das Interface lädst, das Kabel ein- und wieder ausstecken. Zweitens kannst du Programme kabellos im Onlinemodus betreiben und so Fehler viel leichter finden als im Downloadbetrieb. Und schließlich lassen sich die mobilen Roboter im Onlinemodus über ein Bedienfeld in ROBO Pro ähnlich wie mit der IR-Fernbedienung online über den Bildschirm steuern. Im Gegensatz zur Fernbedienung kann man sich am Bildschirm zusätzlich auch noch Daten, die das Interface liefert, anschauen, z. B. Werte von Variablen oder Analogeingängen, Versorgungsspannung, die der Akku liefert, Geschwindigkeit der Motoren.

Als Beispiel haben wir das Teach-In-Programm umgebaut und steuern den Fahrroboter über ein Bedienfeld. Das Programm heißt Mobile-Teach-RF.rpp. Du kannst es natürlich auch mit dem Schnittstellenkabel ausprobieren. Das ist aber ziemlich unkomfortabel. Der Aktionsradius des Modells ist beschränkt, das Kabel verwickelt sich und der Roboter dreht sich nicht mehr richtig. Sicher wirst du dich danach sofort auf den Weg machen und dir den RF Data Link besorgen.

Lade das Programm Mobile-Teach-RF.rpp.
Schalte in der Funktionsleiste des Hauptprogramms um auf Bedienfeld. Danach startest du das Programm im Online-Modus. Nun kannst du das Modell über die Knöpfe im Bedienfeld steuern und programmieren.



1. Taste „Lernen“ drücken. Der „Lernvorgang“ wird gestartet.
2. Mit den Pfeiltasten das Modell in die gewünschte Richtung steuern.
3. Taste „Speichern“ drücken. Die abgefahrene Strecke wird gespeichert.
4. Taste Wiedergabe drücken. Die gespeicherte Strecke wird abgefahren.

Auch hier geht der gespeicherte Weg verloren, wenn das Programm beendet wird.

Mehr über das Erstellen von Bedienfeldern erfährst du im ROBO Pro Handbuch.

ROBO I/O-Extension

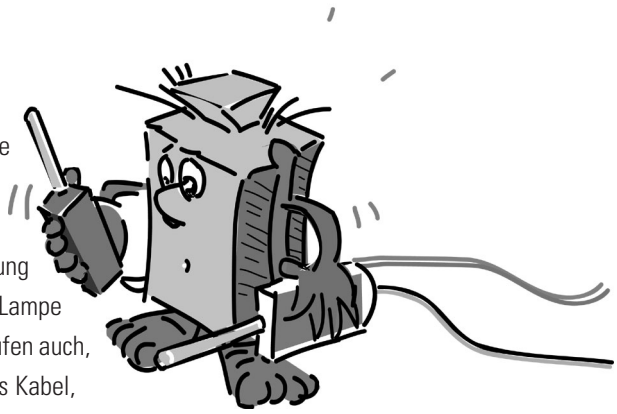
■ Falls du ein Modell mit so vielen Sensoren und Motoren baust, dass die Ein- und Ausgänge des ROBO Interface nicht ausreichen, kannst du ein ROBO I/O-Extension Art.-Nr. 93294 an das Interface anschließen. Damit stehen dir weitere 8 Digitaleingänge, 4 Motorausgänge und ein analoger Widerstandseingang zur Verfügung. An dieses I/O-Extension kannst du ein zweites und an das zweite ein drittes Modul anschließen, die dann alle von einem ROBO Interface angesteuert werden. Insgesamt stehen dir dann 16 Motorausgänge, 32 Digitaleingänge, 5 analoge Widerstandseingänge, 2 analoge Spannungseingänge sowie 2 Eingänge für Abstandssensoren zur Verfügung.

Hast du dann immer noch nicht genug, kannst du an den PC auch mehrere Interfaces im Online-Modus steuern, z. B. eines an einer seriellen COM-Schnittstelle, eines am USB-Port oder 2 Interfaces am USB-Port und jedes mit bis zu 3 ROBO I/O-Extensions! Da kann es einem ganz schwindelig werden. Wie das Ganze funktioniert ist ebenfalls im ROBO Pro Handbuch in Kapitel 6 erläutert.

Fehlersuche

■ Experimentieren macht Spaß. Doch nur solange alles funktioniert. Meistens ist das auch der Fall. Doch leider nicht immer.
Erst wenn ein Modell nicht arbeitet, stellt sich heraus, ob man den Mechanismus genau verstanden hat und den Fehler sofort findet.
Bei mechanischen Fehlern kann man immerhin noch etwas sehen (falsch zusammengebaut) oder fühlen (Schwergängigkeit). Kommen elektrische Probleme hinzu, wird es schwieriger.
Die Profis nutzen zur Fehlersuche eine Reihe sehr unterschiedlicher Meßinstrumente, wie z. B. Spannungsmesser oder Oszillograf. Solche Geräte hat nicht jeder greifbar. Wir wollen deswegen versuchen, mit einfachen Mitteln einen Fehler einzukreisen und zu beheben.

Kabelmontage



Bevor wir mit unseren Experimenten beginnen, müssen wir einige Komponenten aus dem Fischertechnik-Baukasten erst fertigstellen. Es werden z. B. die mitgelieferten Stecker an die einzelnen Kabelabschnitte angeklemmt.
Zuerst wird das Kabel zugeschnitten. Wir messen dazu die vorgegebenen Längen ab und schneiden die Abschnitte zu. Jedes Kabel wird nach Fertigstellung durchgemessen. Dazu brauchen wir den Akku und die Lampe. Leuchtet die Lampe nachdem sie mit dem Akku verbunden wurde, ist das Kabel in Ordnung. Wir prüfen auch, ob die Farbzuordnung stimmt, roter Stecker rotes Kabel, grüner Stecker grünes Kabel,

Interfacetest

Arbeitet das Programm (auch das mitgelieferte) nicht mit unserem Modell zusammen, starten wir den Interfacetest. Dieses Hilfsprogramm gestattet uns, die Ein- und Ausgänge separat zu testen. Funktionieren die Sensoren? Drehen sich die Motoren in die richtige Richtung? Bei allen unseren mobilen Roboter sind die Motoren so angeschlossen, dass sich bei Drehrichtung=links das Rad oder das Bein vorwärts bewegt. Ist hier ebenfalls alles in Ordnung, suchen wir die mechanische Ursache.

Wackelkontakte

Ein übler Fehler sind Wackelkontakte. Zum einen können die Anschlussstecker lose in den Buchsen sitzen. Ist dies der Fall, werden mit einem kleinen Schraubendreher die Kontaktfedern der Stecker etwas aufgeweitet. Vorsicht, zu starkes Aufweiten führt zum Bruch der Kontakte oder zu Schwergängigkeit beim Einstecken.
Eine andere Ursache für Wackelkontakte sind gelockerte Klemmverbindungen an den Anschraubstellen der Stecker. Bitte vorsichtig festschrauben! Bei der Gelegenheit wird gleich geprüft, ob keine der dünnen Kupferdrähtchen abgebrochen sind.

Kurzschlüsse

Es könnte auch einmal vorkommen, dass man durch falsches Verlegen der Kabel einen Kurzschluss produziert. Dann funktioniert auch nichts mehr wie es soll. Im Akkupack ist eine Sicherung eingebaut, die bei zu hohem Strom oder zu hoher Temperatur den Strom abschaltet. Die Ausgänge des Interface werden bei Überhitzung ebenfalls stillgelegt.
Zu einem Kurzschluss kann es auch kommen, wenn man an den elektrischen Steckern das Schräubchen, mit dem das Kabel festgeklemmt wird, nicht richtig festzieht. Dann ragt es eventuell über den Rand des Steckers hinaus. Steckt man dann zwei Stecker so in zwei direkt nebeneinander liegende Buchsen am

Interface, dass sich die Schräubchen berühren, kommt es zu einem Kurzschluss. Deshalb sollten die Schräubchen immer gut festgezogen werden und die Stecker so eingesteckt werden, dass sich die Schräubchen nicht berühren können.

Stromversorgung

Gibt es unerklärliche Aussetzer während des Betriebes, kann ein beinahe leerer Akku die Ursache sein. Die Spannung sinkt beim Zuschalten einer Last (Motor ein) kurz ab und damit wird ein Reset für den Prozessor auf dem Interface ausgelöst. Das ROBO Interface zeigt durch das Leuchten der roten LED an, wenn die Spannung der Stromversorgung zu niedrig ist. Dann muss der Akku aufgeladen werden.

Programmierfehler

Treten Fehler bei selbst geschriebenen Programmen auf, die man sich nicht erklären kann, sollte sicherheitshalber ein möglichst ähnliches der mitgelieferten Programme eingespielt werden, damit elektrische oder mechanische Defekte ausgeschlossen werden können. Im Online-Modus lässt sich der Programmfluss am Bildschirm verfolgen. Geht das Programm an einer bestimmten Stelle nicht weiter, kann man hier nach der Ursache suchen, z. B. falscher Eingang oder Motor ausgewählt, bei einer Verzweigung falschen Wert abgefragt oder J/N-Anschlüsse vertauscht.

Führt dies alles nicht zum Erfolg, bleibt noch der Kontakt zum fischertechnik-Service (email: info@fischertechnik.de).

Oder du besuchst uns im Internet, unter www.fischertechnik.de. Da gibt es ein Forum, Chat, Marktplatz, Galerie und du kannst kostenlos Mitglied im Fischertechnik Fanclub werden.

Wir wünschen dir mit dem ROBO-Mobile-Set noch viele Stunden Spaß mit reichlich Aha-Effekten.