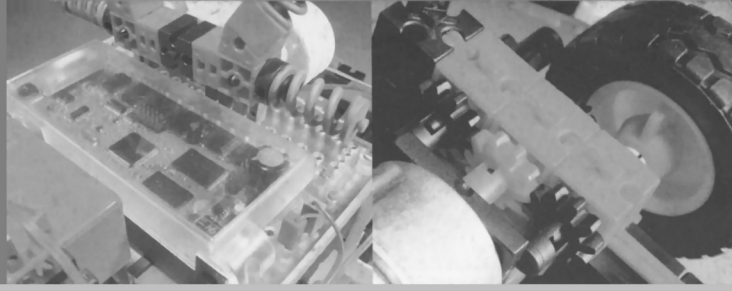


fischertechnik

COMPUTING

Begleitheft
Activity booklet
Manuel d´accompagnement
Begeleidend boekje
Cuarderno adjunto
Folheto

ROBO MOBILE SET

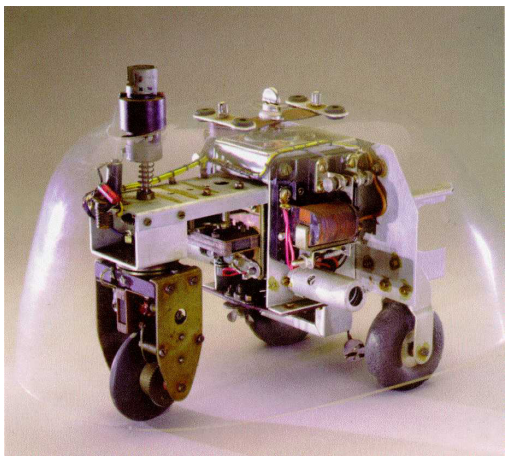
8 MODELS

Inhalt

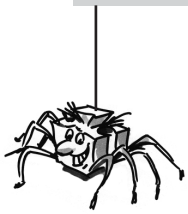
Wozu brauchen wir Roboter?

■ 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).

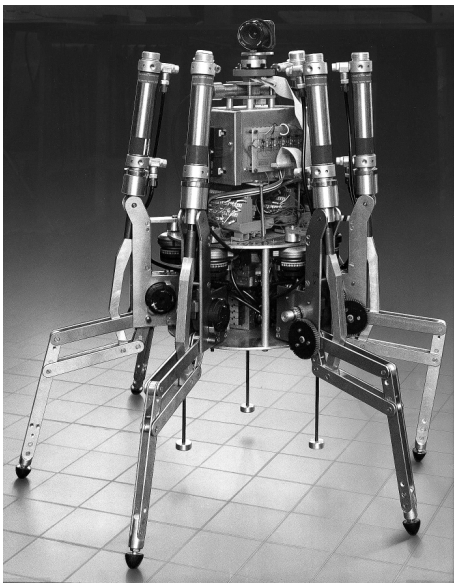


■ Das oben beschriebene Gedankenexperiment hat der Engländer Walter Grey in den 50er Jahren in die Tat umgesetzt.
Mit Hilfe einfacher Sensoren, Motoren und elektronischer Schaltungen wurden verschiedene „kybernetische" Tiere geschaffen, denen dann ganz spezifisches Verhalten, wie z. B. das einer Motte, 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.

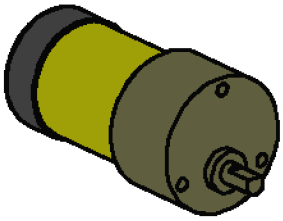
Roboter aus fischertechnik

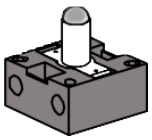
■ Wie können wir nun aus unserem fischertechnik-Baukasten Roboter bauen? Für einen Roboter brauchen wir außer den Sensoren (z. B Taster,) und Aktoren (z. B. Motoren) viele mechanische Teile um daraus ein Modell zu konstruieren. Der fischertechnik-Baukasten ROBO Mobile Set ist hierzu die ideale Grundlage.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



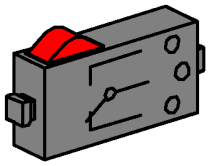


Linsenlampe:

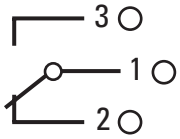
Zur Ausgabe einfacher Lichtsignale dient diese Glühlampe (9VDC/150mA).
Im Glaskolben der Lampe ist eine Linse integriert, die das austretende Licht bündelt. Richtet man den Lichtstrahl auf einen Helligkeitssensor (Fototransistor, siehe unten) kann man so eine Lichtschranke bauen, die Hell und Dunkel 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.

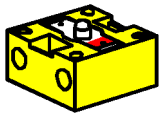
Sensoren

■ 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.

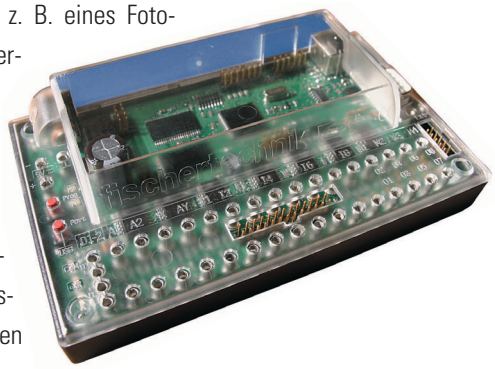
Beim Fototransistor handelt es sich übrigens um ein so genanntes Halbleiterbauelement, dessen elektrische Eigenschaften lichtstärkeabhängig sind. Jeder kennt Solarzellen, mit denen aus Sonnenlicht

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.

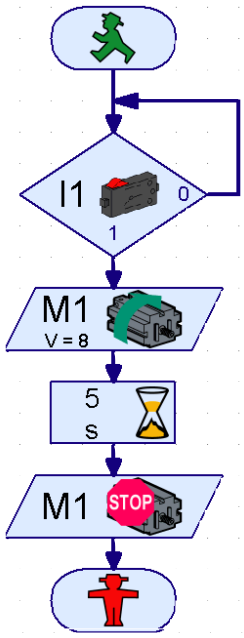
ROBO Interface

■ 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.

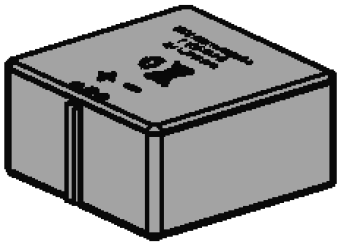
Software ROBO Pro



■ 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

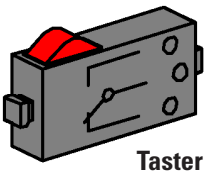
■ Bei unserem Eindringen in die faszinierende Welt der mobilen Roboter werden wir schrittweise vorgehen. Wir beginnen mit einem einfachen Testaufbau zur Prüfung der Grundfunktionen von Interface und Sensorik. Danach bauen wir einfache Modelle, denen bestimmte Aufgaben zugeordnet sind und versuchen uns 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.

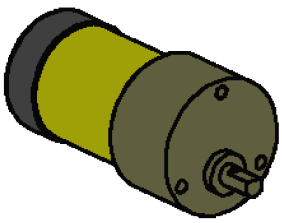
Im Handbuch der Software ROBO Pro wird in Kapitel 1 und 2 beschrieben, wie die Steuersoftware auf dem PC installiert und das Interface angeschlossen wird. Mit Hilfe des Interface-Tests testen wir die unterschiedlichen Sensoren bzw. Aktoren.



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

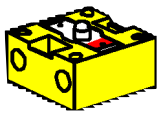
Powermotor

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

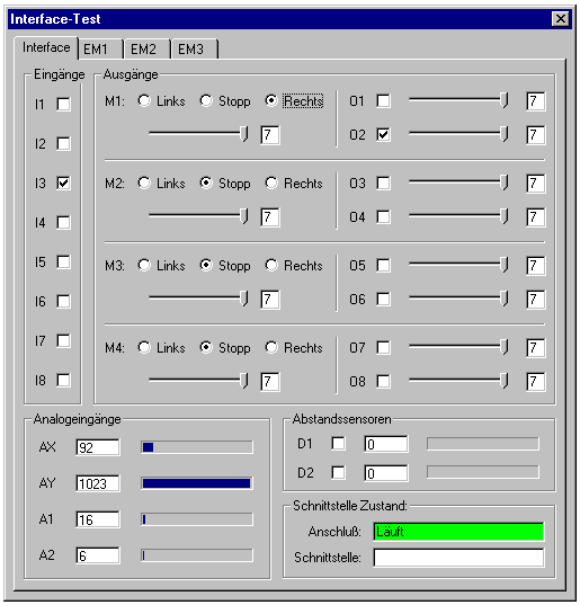
Wollen wir auch den Analogeingang AX testen, können wir dazu einen Fototransistor als Analogsensor verwenden.

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).

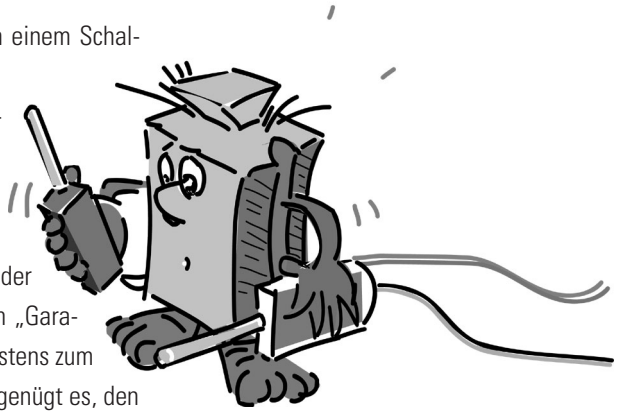


Fototransistor

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.

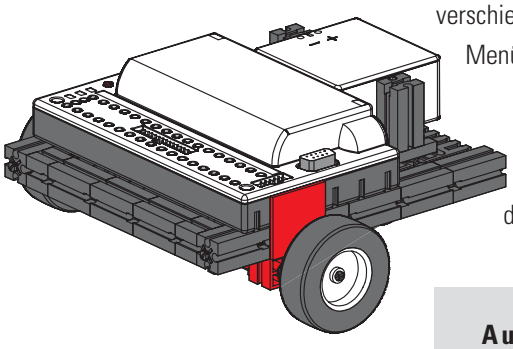


Um nochmals kurz auf die Farbzuordnung der Stecker zu kommen: Wir achten genau darauf beim Zusammenbau immer einen roten Stecker an die rote Leitung anzuschließen und einen grünen Stecker mit der grünen Leitung zu verbinden. Wenn es in einem Schaltungsaufbau auf die richtige Polarität ankommt, nehmen 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.

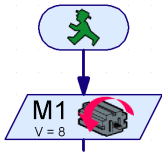
Wir öffnen die Software ROBO Pro und legen ein neues Programm an (DATEI – NEU). ROBO Pro bietet verschiedene Schwierigkeitsstufen an, in denen wir arbeiten können. Sie lassen sich in ROBO Pro im Menüpunkt LEVEL einstellen. Momentan genügt uns Level 1.

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.

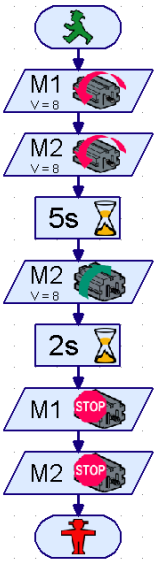
Aufgabe 1 (Level 1):
Unser „Einfacher Roboter" soll 5 Sekunden lang geradeaus fahren, anschließend 2 Sekunden lang im Kreis drehen und danach stehen bleiben.

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:

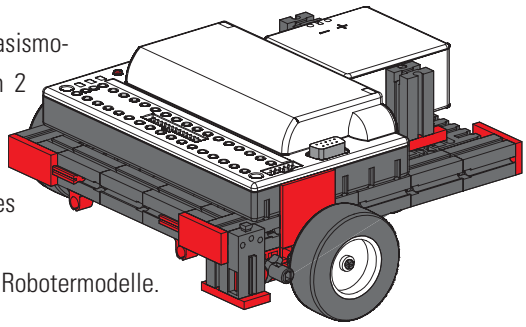
- Pole einfach immer wieder die Motoren so um, dass der Roboter in die gewünschten Richtungen fährt.
- Die Geschwindigkeit der Motoren kannst du im Eigenschaftsfenster von jedem Motorsymbol zwischen 1 und 8 einstellen. Drehen M1 und M2 mit unterschiedlichen Geschwindigkeiten in die gleiche Richtung, 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



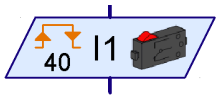
■ Gegenüber unserem ersten „Einfachen Roboter" ist das Basismodell stabiler und robuster aufgebaut. Es enthält außerdem 2 Sensoren zur Wegmessung, die jeweils aus einem Taster und einem Impulsrad bestehen. Das Impulsrad ist mit der Motorachse verbunden und betätigt bei jeder Umdrehung des Motors vier 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.

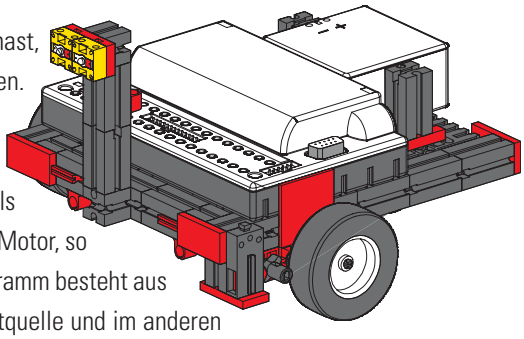
Aufgabe 2 (Level 2):

- **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.

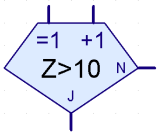
Baue zunächst das Modell Lichtsucher wie in der Bauanleitung beschrieben auf.

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.**

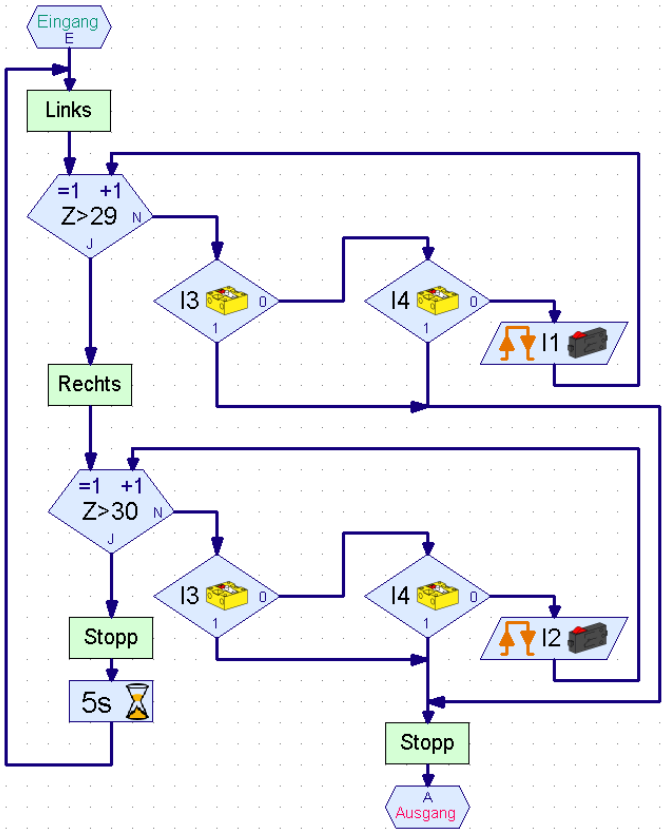
Tipps:

- 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:

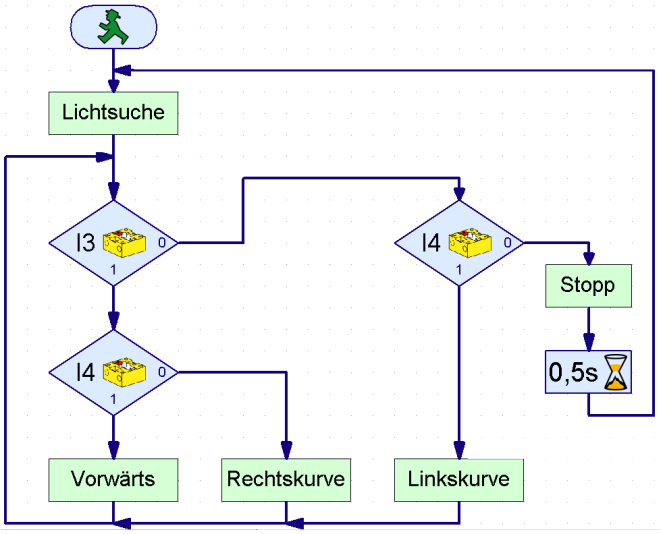


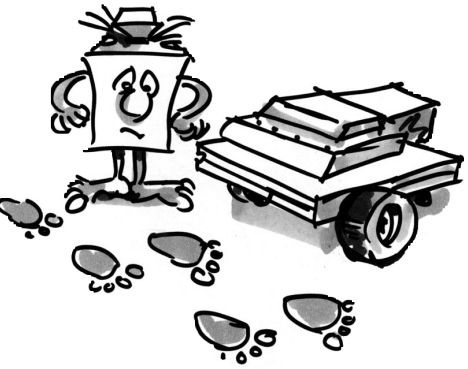
- Im Hauptprogramm fragst du wieder die Fototransistoren ab und steuerst die Motoren abhängig davon, welcher Fototransistor Licht erkennt:

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

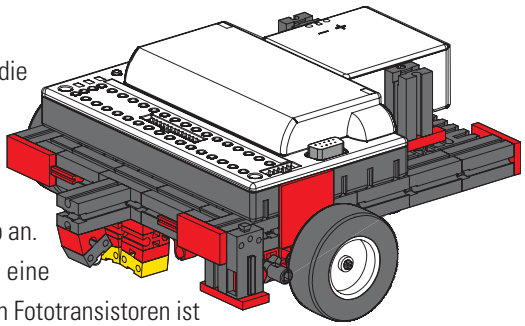
- Die Rechts- und Linkskurve erzeugst du durch unterschiedliche Geschwindigkeiten von M1 und M2 bei gleicher Drehrichtung. Dadurch ergibt sich ein sehr harmonischer Fahrstil.

- 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.

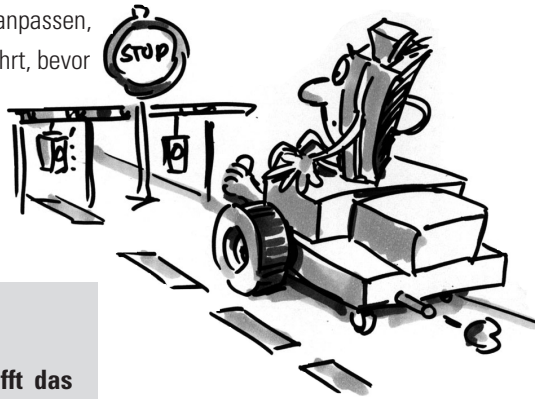
Baue nun das Modell Spurensucher gemäß der Bauanleitung auf.

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.

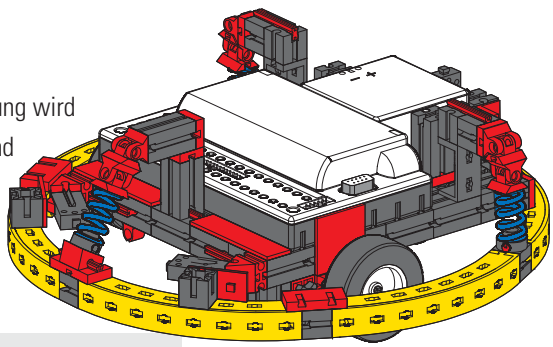


Aufgabe 2:

- **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.

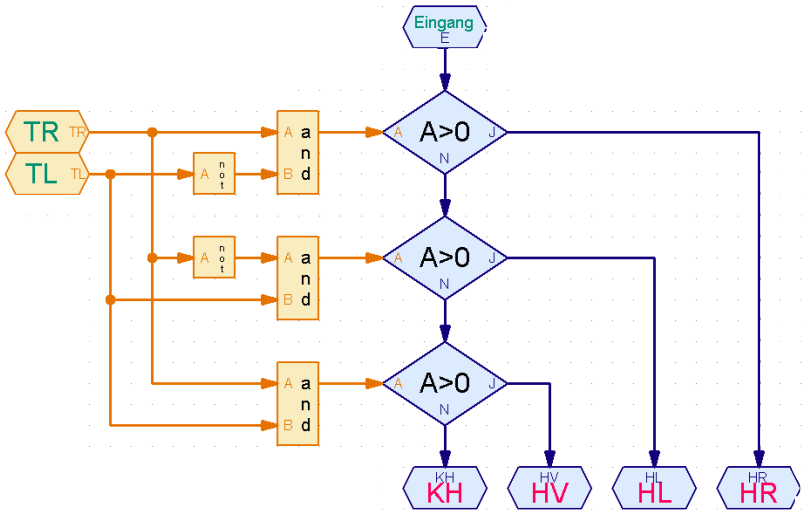
Aufgabe 2 (Level 3):

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

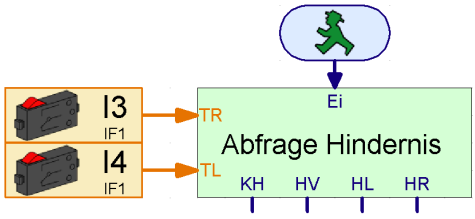
Tipps:

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

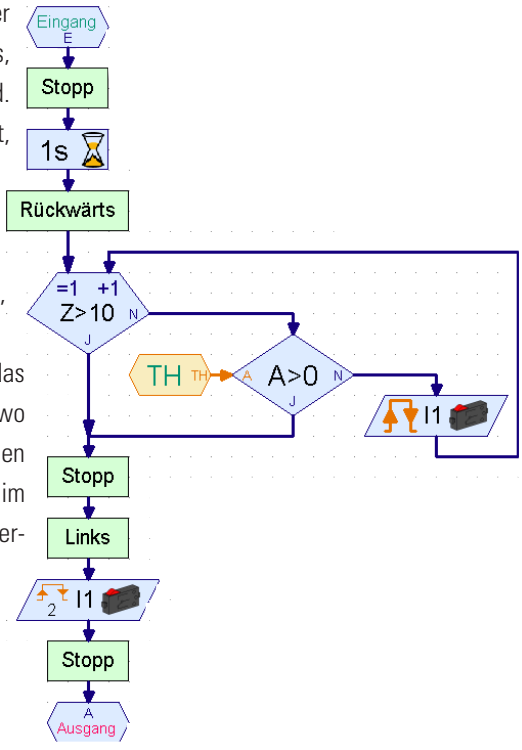
Dateneingang TR = Taster rechts
Dateneingang TL = Taster links
Ausgang KH = kein Hindernis
Ausgang HV = Hindernis vorne
Ausgang HL = Hindernis links
Ausgang HR = Hindernis rechts



- 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**

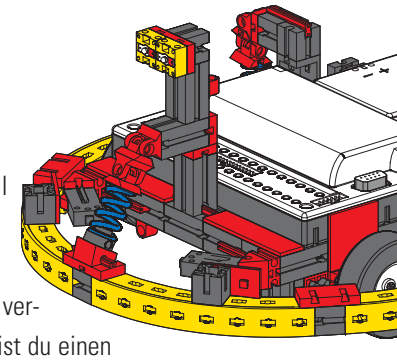
Verwende zur Lösung die Programmelemente aus ROBO Pro Level 3.

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.

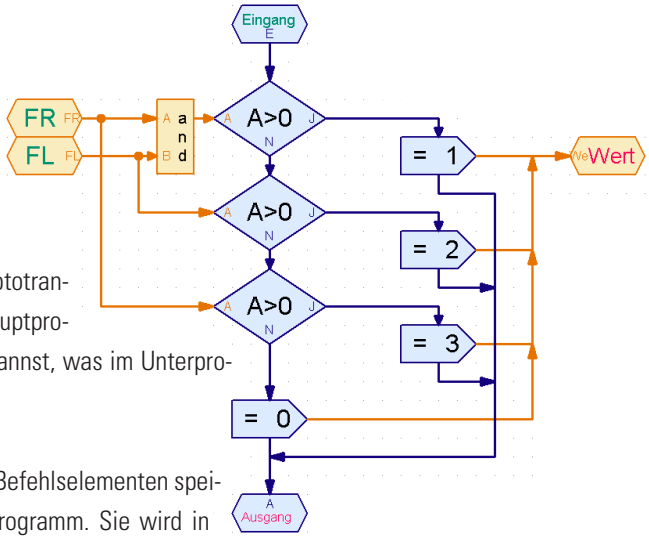
Unterprogramm Licht:

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.

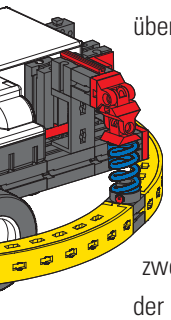
Unterprogramm Licht:



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.

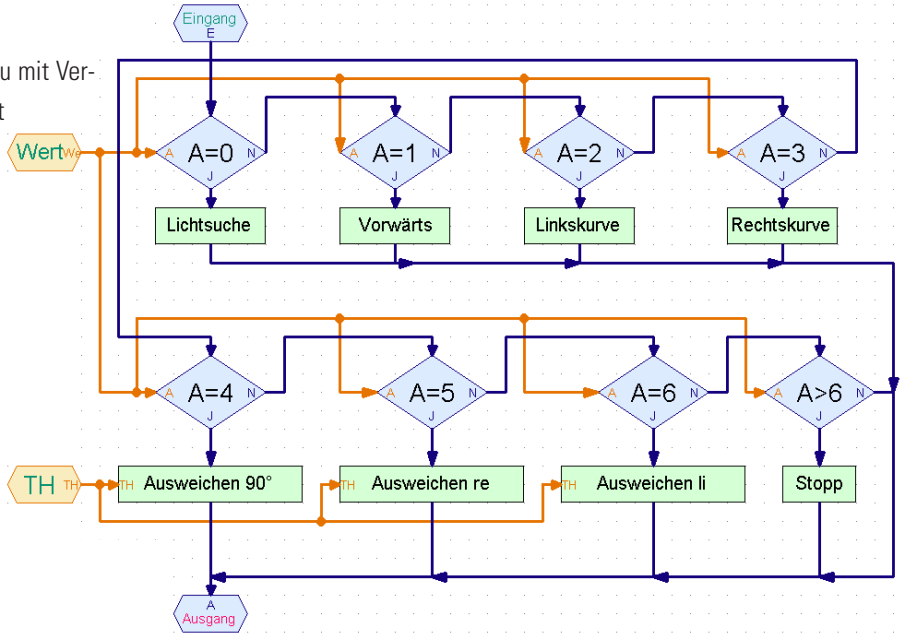
Die Variable, die den Wert aus den Befehlselementen speichert, kommt ebenfalls ins Hauptprogramm. Sie wird in mehreren Unterprogrammen verwendet. Du verbindest sie über einen Datenausgang mit dem Unterprogramm.



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.

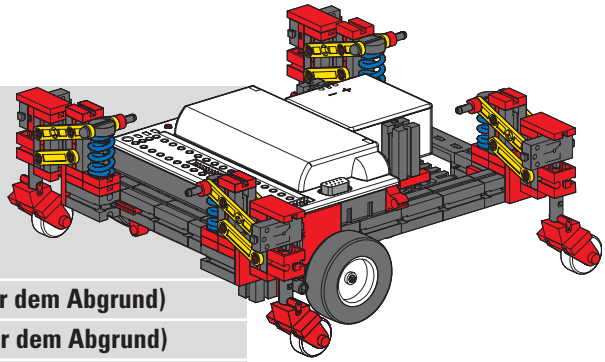
Eventuell musst du den einen oder anderen Taster etwas nach oben oder unten justieren.

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:

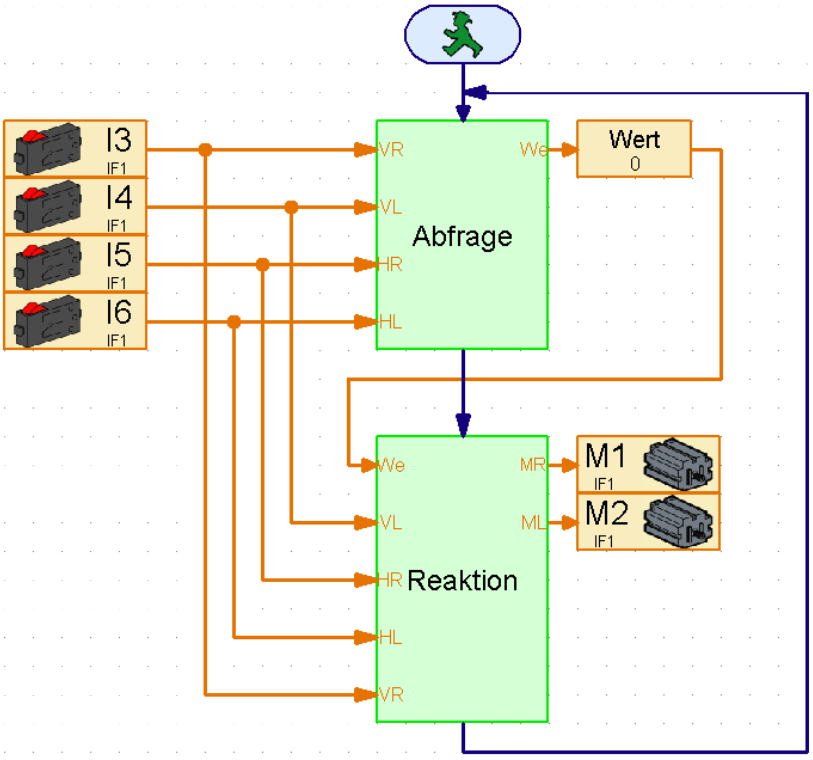
Die Lösung findest du in der folgenden Tabelle. Die Sensoren, die sich über dem Abgrund befinden (Taster=0) sind markiert. Jede Kombination erhält eine Nummer. Im Programm, das später erstellt wird, wird jeder Möglichkeit die entsprechende Zahl zugewiesen. Anhand der Zahl reagiert der Roboter auf die 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

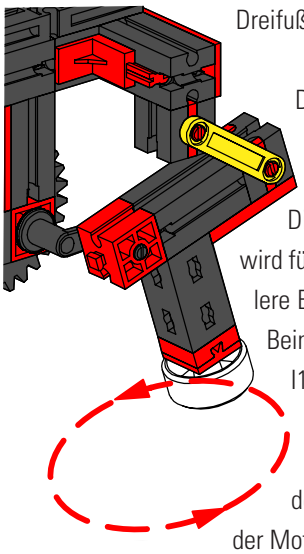
■ Nachdem wir uns ausführlich mit den fahrbaren Robotern beschäftigt haben, wenden wir uns nun einer anderen Fortbewegungsart zu, die wir für mobile Roboter nutzen können, dem Laufen.
Die Gangart der Insekten eignet sich hervorragend als Vorbild für den Antrieb von „maschinellen Sechsbeinern". Beim sogenannten 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.

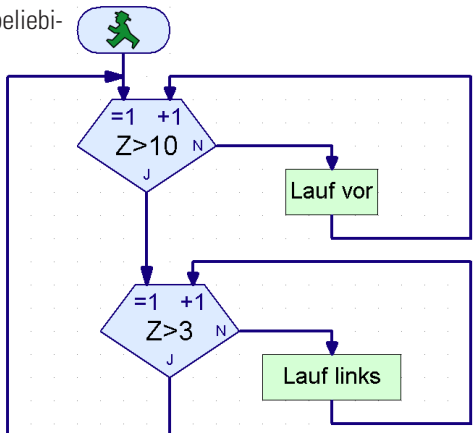
Aufgabe 2 (Level 2):

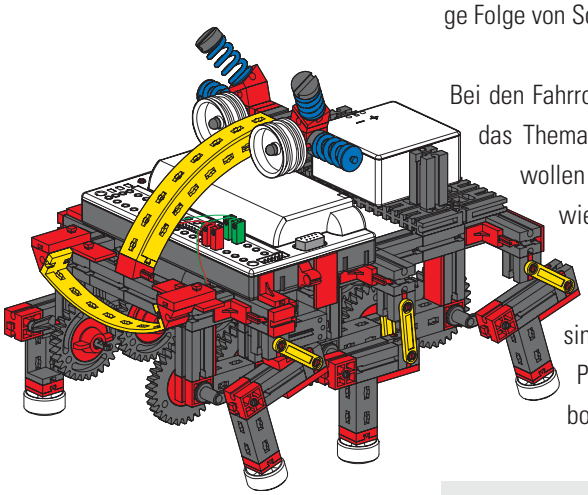
- **Programmiere dein Modell so, dass es 10 Schritte nach vorne, 3 Schritte nach links, 3 Schritte nach rechts und wieder 10 Schritte zurück läuft.**
- **Lege für jede Richtung ein eigenes Unterprogramm an.**
- **Verwende zum Zählen der Schritte das Element Zählschleife.**

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).





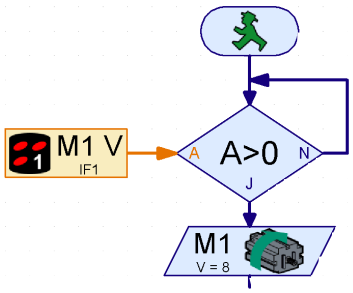
Bei den Fahrrobotern haben wir bereits ausführlich das Thema Hinderniserkennung behandelt. Wir wollen es an dieser Stelle nicht noch einmal wiederholen. Aber versuche doch einmal dieses Verhalten auf den Laufroboter zu übertragen. Die Sensoren dafür sind im Baukasten enthalten. Bei der Programmierung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

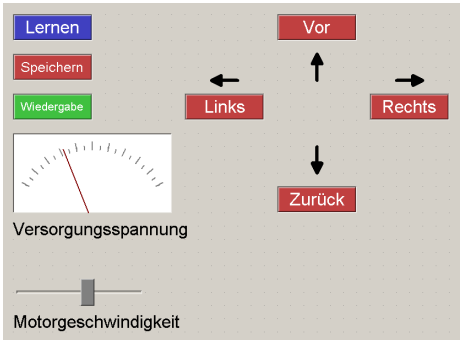
Mit so einer Anwendung wird das Programmieren von Robotern zum Kinderspiel! Du musst beachten, dass die gespeicherte Strecke gelöscht wird, sobald du das Programm mit dem Prog.-Taster am Interface stoppst.

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

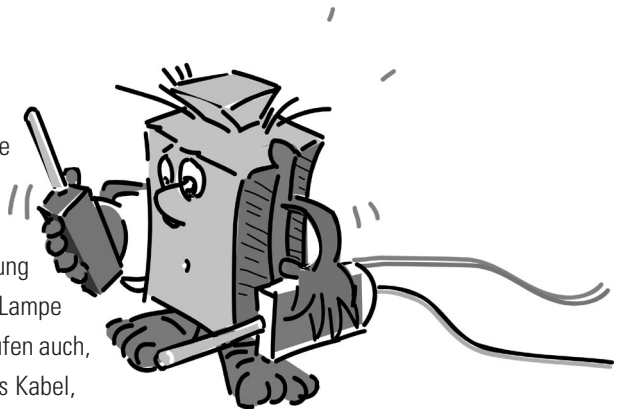
■ Falls du ein Modell mit so vielen Sensoren und Motoren baust, dass die Ein- und Ausgänge des ROBO Interface nicht ausreichen, kannst du ein ROBO I/O-Extension Art.-Nr. 93294 an das Interface anschließen. Damit stehen dir weitere 8 Digitaleingänge, 4 Motorausgänge und ein analoger Widerstandseingang 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.

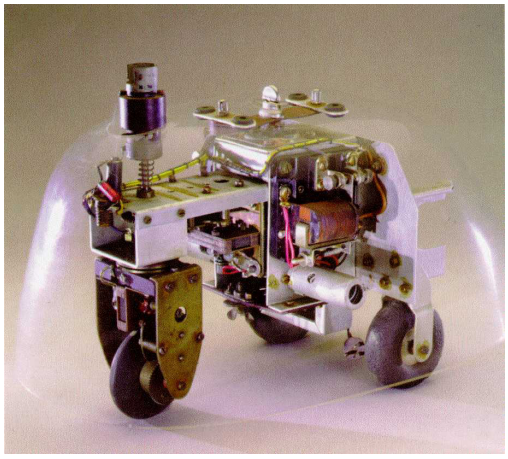
Wir wünschen dir mit dem ROBO-Mobile-Set noch viele Stunden Spaß mit reichlich Aha-Effekten.

Contents

Why Do We Need Robots?

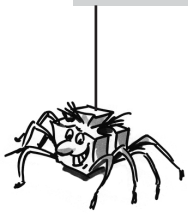
■ Carel Capek coined the term robot in his 1923 novel "Golem". This artificially created figure was designed to take over human labor with his abilities.
In the 30's and 40's of the last century the robot became more of an automaton. Today we can look back and smile at the different attempts to divest it with human characteristics such as a head with blinking lights as eyes, etc. These machines showed little sign of intelligence or even mobility. Since the principle of control has great influence on the robotics, the design of robots became more realistic with the advent of electronic circuits. Even today, the question of the "intelligence" of the robot is subject of much research and fact-finding in many companies, institutes and universities.

■ So called cybernetics offered the first promising approach to the problem. The term "cybernetics" is derived from the Greek word kybernetes. The kybernetes was the navigator on Greek ships. He had to determine the ship's position and chart the course to the destination.
Clearly cybernetics was supposed to make the robot "intelligent". But what would such intelligent behavior look like?
We shall try to illustrate this using a thought experiment. Everybody has probably observed a moth's behavior in the light of a lamp. The moth detects the source of the light, flies toward it, and then avoids hitting the lamp at the last moment. It is clear that in order to exhibit this behavior, the moth has to detect the source of light, plot a course there and then fly toward it. This ability is based on instinctive intelligent behavior patterns of the insect.
Now lets try to apply these abilities to a technical system. We have to detect the source of the light (optical sensors), execute a movement (operate motors) and we have to establish a meaningful connection between the detection and the movement (the program).



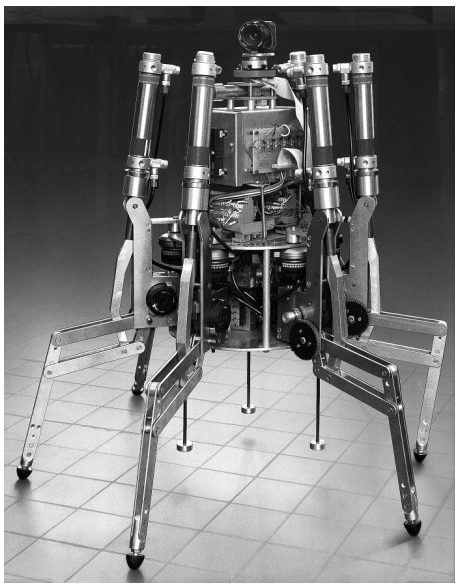
■ It was the Englishman Walter Grey who put the thought experiment described above into practice in the 1950's.
With the help of simple sensors, motors and electronic circuits he created a variety of "cybernetic" animals that displayed the very specific behavior of, lets say, a moth. The photograph shows a replica of the "cybernetic" turtle, exhibited at the Smithsonian Museum in Washington.

Based on these ideas we will create similar "patterns of behavior" for our robots and will try to communicate them to the robot in the form of programs.



■ But why do we need mobile robots? Lets try to apply the behavior of our "imaginary moth" to technical devices. A simple example for this is light-seeking behavior. We modify the light source by attaching a bright strip, a guideline, to the floor and align the sensors to face not forward but downward. With the help of such guidelines a mobile robot can find its way in a warehouse for example. Additional information at specific points along the line, in the form of a bar code for example, direct the robot to perform further actions at these locations such as picking up or dropping a pallet.
In fact such robot systems are already in existence today. In large hospitals it is often necessary to cover long distances for the transportation of consumable supplies such as bed linens. It is time-consuming and expensive to have the nursing staff transport these materials and often requires hard physical labor. Additionally performing such tasks reduces the time available to take care of patients.

■ For the last couple of years scientists have begun to deal with another from of movement that is very common in nature, walking or running. Robots have been developed that have the ability to move about on legs. The electropneumatic walking robot "Achille" that was developed by the royal military academy in Brussels is an example of a six-legged walking robot. Equipped with one camera above and one on each of the six legs, this robot is designed to be able to react mechanically to raised or sunk obstacles (objects or holes).
Such walking machines can be deployed everywhere where wheeled and track vehicles don't have much of a chance, such as rough or soft terrain, for climbing over obstacles, ascending stairs, surmounting ditches or for operation in inaccessible or dangerous areas in nuclear power plants, mining tunnels or during rescue operations.



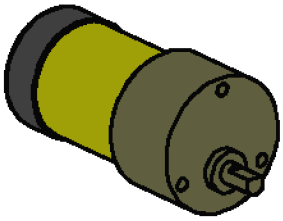
It is easy to recognize that mobile robots can play an important role in modern society.

Robots Built with fischertechnik

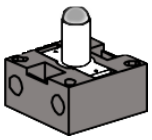
■ So how can we build a robot with our fischertechnik construction kit? To build a robot we need sensors (e.g. pushbutton sensors,) and actuators (e.g. motors) but also many mechanical parts to construct a model. The fischertechnik ROBO Mobile Set offers an ideal foundation for this. The following sensors and actuators are included in this construction set:

Actuators

Power Motor:



Two of these powerful DC motors (9VDC/2,4W) with built-in gearbox and a reduction of 50:1 drive the mobile robots (this means that while the motor performs 50 revolutions the shaft extending from the motor turns only once during the same time.

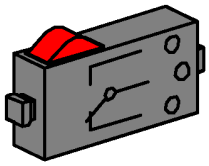


Lens Tip Lamp:

This incandescent bulb (9VDC/150mA) enables the output of simple light signals.
A lens is integrated into the glass bulb of the lamp focusing the emitted light. By directing a beam of light toward a light sensor (phototransistor, see below) you can build a light barrier able to differentiate between light and dark. The lamp can also be used to display certain states or to generate warning messages in the form of a blinking lamp. In this construction kit the lamp is used together with 2 phototransistors as special sensor for line recognition.

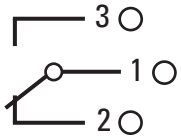
Sensors

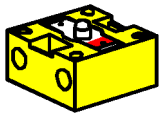
■ The pushbutton sensor is an example of a digital sensor. Digital values can only assume 2 different states. These states are identified with 0 or 1 respectively. For the pushbutton sensor "0" signifies that no current flows between the contacts, "1" signifies that current is flowing.



The fischertechnik **pushbutton** sensor is designed as a three-way switch. For this reason there are three terminals. When the red button is pressed the switch connecting the terminals 1 and 3 is activated mechanically. At the same time the connection between the terminals 1 and 2, which was connected during the quiescent state, is interrupted. This way both possible starting positions can be selected:

Closed in quiescent state (terminal 1 and 2 connected)
Open in quiescent state (terminal 1 and 3 connected).





The **phototransistor** may be used both as digital as well analog sensor. In the first case it serves to recognize clear transitions between light and dark, a marked line for example. But it is also possible to distinguish the amount of light according to its brightness. In this case the phototransistor works as analog sensor. Analog values can vary freely between their extreme values. These values have to be converted into their respective numerical values in order to be processed by the computer.

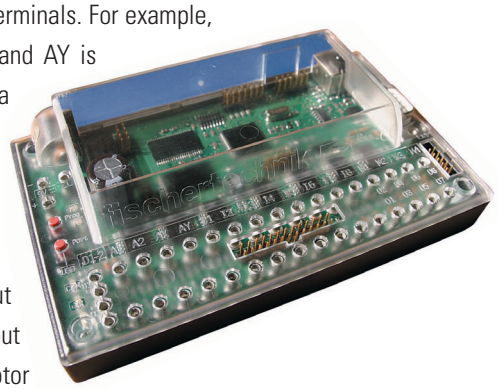
Incidentally phototransistors belong to the so-called semiconductor devices, its electrical characteristics are dependent on the intensity of the light. Everybody knows about solar cells that use sunlight to

generate electricity. A phototransistor can be understood as a combination of a mini solar cell and a transistor. Light impulses (photons) received by the phototransistor generate a very low current that is then amplified by the transistor.

Note:
Please make sure the polarity is correct before connecting the phototransistor: Red marking = plus pole.
Maximum voltage: 30V max

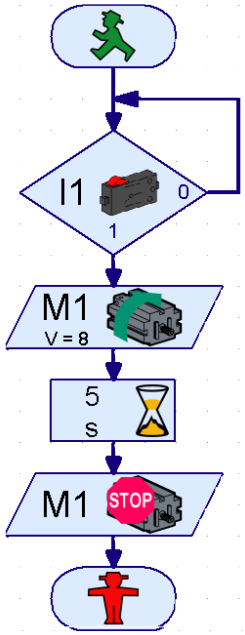
ROBO Interface

■ We can connect different sensors and actuators to the ROBO Interface and interpret them. In addition to 8 digital input terminals the ROBO Interface also offers several analog input terminals. For example, a resistance value between 0 and 5,5 kΩ applied to the input terminals AX and AY is converted into a numerical value between 0 and 1024. The measured values of a brightness sensor, such as a phototransistor, can so be acquired and are available for further processing. Voltages between 0 and 10VDC can be measured at the analog input terminals A1 and A2.



The most important role of the Interface lies in the logical connection of the input values. The Interface needs a program to do this. The program decides in how input data and sensor signals are processed to generate appropriate output data, motor control signals, etc. With the ROBO Interface we have enough computing power at our disposal to design even the most sophisticated programs.

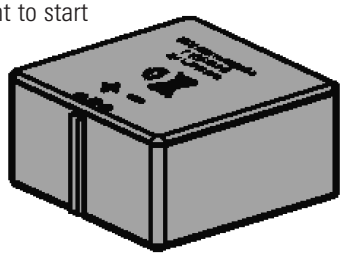
Software ROBO Pro



■ A graphical programming interface provides us with the most effective way to create the necessary programs for the ROBO Interface. The term "programming interface" stands for a software that enables us to create our programs in a very comfortable way, using graphical symbols. Actually, the computer of the ROBO Interface can only execute commands contained in its so-called machine instruction set. These are essentially simple control structures that are extremely difficult to use for beginners. That's why the PC software ROBO Pro uses graphical elements that are later translated into a language that can be executed by the Interface.

Power Supply

■ The only thing you need in addition to the ROBO Mobile Set is the Accu Set. It contains the battery pack serving as mobile power supply for our robot models and a special charger for the battery pack.
It would be best to start charging the battery pack right away using the charger. This way it will be fully charged when we want to start experimenting.



Approach to Experimentation

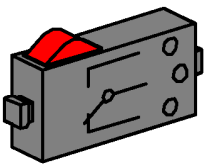
■ We will go step by step in our exploration of the fascinating world of mobile robots. We will start with a simple test setup to check the basic functions of Interface and sensors. Then we will build simple models, which will be assigned specific functions and later attempt more and more complicated systems.
Should you feel at some point that creating your own programs is too complicated or takes too much time you can load the supplied sample programs into the Interface and use them to operate the robots.
At the end of this resource guide is a chapter on troubleshooting so you don't despair should errors occur.

It is very important to take proper care during the construction and initial operation of our robots. When connecting electrical components we'll stick closely to the specifications, double and triple checking to make sure everything is ok. When it comes to the mechanical construction, also for your own creations, we will pay close attention to the smoothness of operation and low play in the gears and fastenings. It is up to you and your creativity to write your own programs defining new "behavior". You are only limited by the amount of memory and computing power of your hardware. The following examples can give you some ideas.

First Steps

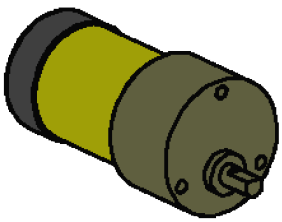
■ Now that we have covered the theoretical considerations we want to start conducting our own experiments. Some of you might want to start right away, maybe even with the big walking robot. This is, of course, possible and if you follow the construction manual closely you will succeed in building the model on the first try.
But what do you do if it isn't working? In this case the cause of the fault must be tracked down systematically. But before dealing with this let's check the interaction between computer and Interface.

Chapter 1 and 2 of the ROBO Pro software manual describes how to install the control software on your PC and how to connect the Interface. With the help of Interface tests we will test the different sensors and actuators.



Pushbutton Sensor

Pushbutton Sensor
We can for example connect a pushbutton sensor to the digital input terminal I1 and observe how the state of the input changes if the pushbutton is pressed.

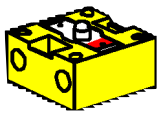


Power Motor

Power Motor
We will test the output terminals by connecting a motor to a motor output terminal, e.g. M1. Using the left mouse button we can start the rotation of the motor and with the slider we can change the speed.

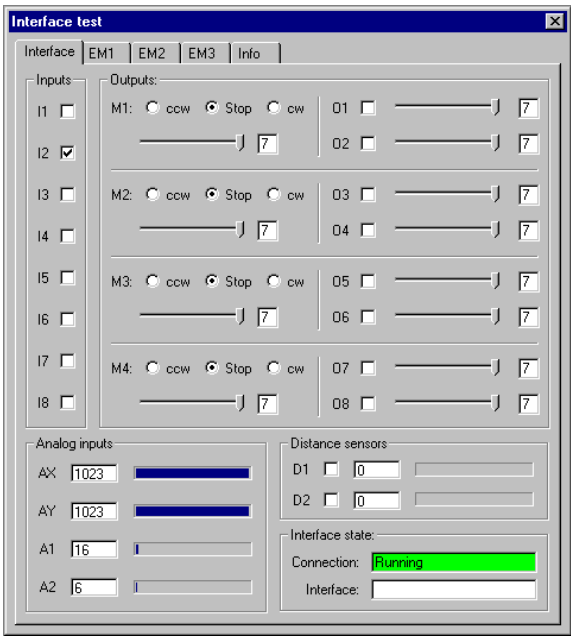
Phototransistor
If we also want to test the analog input terminal AX a phototransistor can be used as analog sensor.

While polarity plays no role in connecting a motor or pushbutton sensor (in the worst case the motor will rotate in the wrong direction) it is vital for the function of the phototransistor to connect it correctly. The contact of the transistor with the red marking should be connected to the red connector and the other contact with the green connector. The second green connector belongs into the socket of the input terminal AX that is located closer to the edge of the Interface. The second red connector fits into the socket of AX that is located further on the inside. (Attention: When connecting the phototransistor to a digital input terminal I1-I8 the red connector needs to go into the socket located closer to the edge of the case.

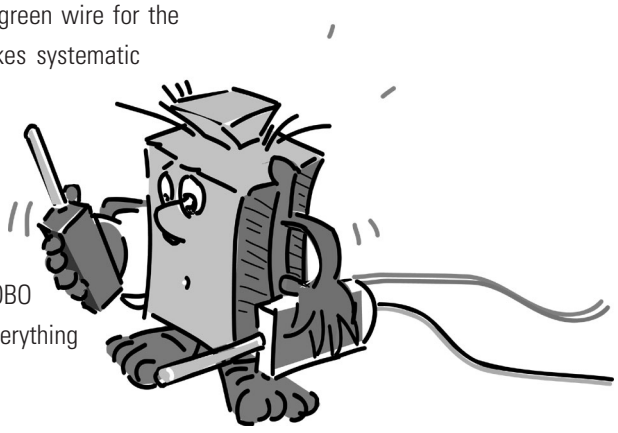


Phototransistor

Now we can vary the intensity of the light of the phototransistor using a flashlight. This will change the reading of the blue bar of AX. If the indicator does not move from its maximum position we should take another look at the connections of the phototransistor. If however the indicator remains at zero even with the flashlight turned off it is possible that the lighting in the room, the ambient light, is too bright. The position of the bar will change when we cover the phototransistor.

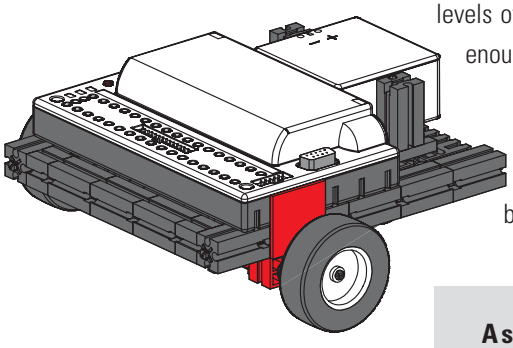


To come back to the color-coding of the connectors again: During assembly we will always be sure to connect the red connector to the red wire and the green connector to the green wire. If the right polarity is important for the circuitry layout we will always choose a red wire for the positive pole and a green wire for the negative pole. While this might seem overly meticulous a clear color-coding makes systematic troubleshooting that much easier.



A simple program will round out our first steps in the area of robotics. The program "Garage Door Control System", explained in chapter 3 of the ROBO Pro manual might have nothing to do with mobile robots but it is an excellent way to get to know the ROBO Pro software. Only the motor and three pushbutton sensors from the ROBO Mobile Set need to be connected to the Interface in order to recreate that program. Everything else is described in detail in the software manual.

The First Simple Robot



■ After Interface test and Garage Door Control System we finally want to put our first robot into operation. We will assemble the model "Simple Robot" with the two drive motors according to the construction manual. This will be pretty quick and easy since this model deliberately only holds what is absolutely necessary for a robot to drive. The motors we'll connect to the output terminals M1 and M2.

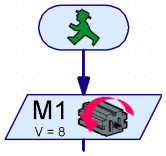
We will open the ROBO Pro software and set up a new program (FILE – NEW). ROBO Pro offers different levels of difficulty to work in. They can be set in the ROBO Pro menu item LEVEL. For now Level 1 is enough for us.

An empty work sheet will appear and on the left side the element window. There you can select the different program elements and place them on the work area using the left mouse button. The right mouse button will let you change the properties.

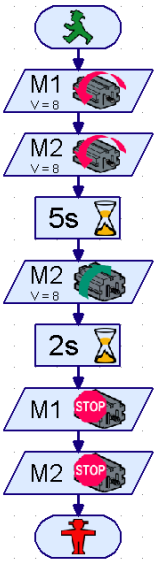
Assignment 1 (Level 1):
Our "simple robot" is should drive straight ahead for 5 seconds, then spin in circles for 2 seconds and after that come to a stop.

Tipps:

Together we will program the first robot step by step:



- We will start with the little green GO man. It represents the start of the program.
- Then we get the motor symbol from the element window and put it below the Start element causing an automatic connection line to be drawn. In the property window we'll set the motor output terminal to "M1" and the rotational direction to "ccw" and confirm with OK.



- By following the same steps to place another motor symbol below the first one we switch on motor 2.
- To wait a certain amount of time we use the Time Delay element placing it below the second motor symbol and set the time to 5 seconds.
- After that we set motor M2 to rotate in the other direction (cw) then wait 2 seconds and finally switch both motors off. Our program concludes with the End symbol, the little red STOP man. The illustration shows the finished program flow chart.

If you are not sure you did everything correctly, you can compare your program with the supplied sample program. To do this you first save your own program and then load the file Simple Robot 1.rpp from the samples directory of ROBO Pro (default setting C:\Program Files\ROBO Pro\Sample Programs\ROBO Mobile Set).

If everything is ok the program can be downloaded into the Interface. After clicking the Download button a pop-up window will appear. There we select the for program to be loaded into FLASH memory 1 and that it should be started as soon as it is downloaded.

Immediately after the download the model will start driving straight ahead, turn for a short time and then stop. If you would like to start the program again press the Prog button on the Interface for a short time. The LED Prog1 will start blinking again for as long as the program is running. After that it will remain on. By the way, the program will remain in the FLASH memory of the Interface even if the power

supply is interrupted. To test this we disconnect the battery pack. Then we reconnect the battery pack and select the saved program by pressing the Prog button until the LED Prog1 lights up. To start the program we simply press the button again.

Our robot can do much so far, right? Why don't we extend the assignment a bit?

Assignment 2 (Level 1):

In order to keep the robot from stopping after just 7 seconds, we will now teach it to dance.

- **Lets have it go straight, turn right, turn left, go backwards for different lengths of time and at various speeds.**
- **This should to continue until the program is stopped, by pressing the Prog button on the Interface.**

Tips:

- Simply keep reversing the polarity of the motors to make the robot go in the desired directions.
- The speed of the motors can be set between 1 and 8 in the property window of each motor symbol. If M1 and M2 rotate in the same direction at different speeds the robot will make a turn.
- Draw a connecting line from the exit of the last program element to the line leading into the first element to make the program repeat continually.
- You can find a finished example under Simple Robot 2.rpp.

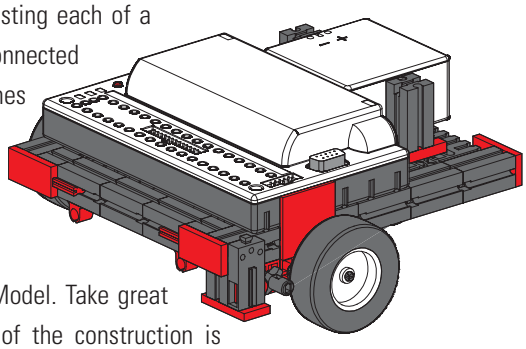
Congratulations, you have now built your first robot and programmed it yourself. It might not be especially intelligent since it is unable to recognize obstacles and would fall of the table if you don't watch out. But this will change during the course of our further experiments.

Intelligent Wheeled Robots

■ Robots require sensors in order to be aware of their environment. The following suggested models introduce a few different mobile robots enabling us to try out the operation of the different sensors. It is thereby imperative that internal states of the robot e.g. measurement of distance traveled using pulse wheels, as well as external signals e.g. light-seeking or tracking are linked. Different assignments have been created for each model. They are designed to give you ideas and to make you familiar with the subject matter. The programs for each assignment can be found in the ROBO Pro directory under \Sample Programs\ROBO Mobile Set\. But feel free to come up with your own assignments for each model. Once you have completed the following examples you're sure to come up with many more ideas.

Basic Model

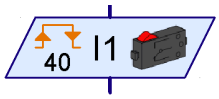
■ Compared to out first "Simple Robot" the basic model is more stabile and robust. In addition it contains 2 sensors to measure the distance traveled, consisting each of a pushbutton sensor and a pulse wheel. The pulse wheel is connected to the motor shaft and activates a pushbutton sensor four times with each rotation of the motor.
This model serves as a foundation for the other mobile robot models.



Follow the construction manual to put together the Basic Model. Take great care during the construction. When the mechanical part of the construction is complete test the smoothness of operation of each motor by connecting it directly to the battery pack without using the Interface.

Assignment 1 (Level 1):

- **Program the Interface so the model drives straight ahead for 40 pulses.**
- **Use the counting sensor at input terminal I1 to measure the pulses.**
- **Measure the distance covered by the model and calculate the distance traveled per pulse.**
- **Repeat this test 3 times and record the variations of the values in a table.**



Tips:

- First switch on both motors (rotational direction left).
- Use the program element **Pulse Counter** to count the pulses at I1.
- Count both pulse edges (0-1 when pressing, 1-0 when letting go of the pushbutton). You can set this under **pulse type** in the property window. This will make the measurement of the distance traveled more exact.
- Then switch off the motors and end the program.
- You will find the finished program under Basic Model1.rpp.

Result:

	Amount of Pulses	Covered Distance	Distance/Pulse
Test 1	**40**		
Test 2	**40**		
Test 3	**40**		

You can say that the model travels a distance of roughly one centimeter (or 0.394 in.) per pulse.

By now you will also know what rotational direction you have to set for each motor in order for the model to drive in a certain direction. Record what you have learned in the table below so you don't have to think about it each time you want to change the driving direction. If you connect the wires exactly as described in the construction manual, a rotational direction to the left causes the wheel to go forward for any motor. That's how the motors are programmed in all sample programs.

Complete the table:

Direction of Model	Rotational Direction M1	Rotational Direction M2
Straight	**ccw**	**ccw**
Backwards		
Left		
Right		
Stop		

Normally you would have to place two motor symbols on the screen for each change of direction. This can be avoided by creating a subprogram for each direction. This will simplify the programming enormously. Chapter 4 of the ROBO Pro software manual describes in detail how to create a subprogram. As soon as you have read this chapter you are ready to tackle the next assignment. You can now switch to **level 2** in ROBO Pro.

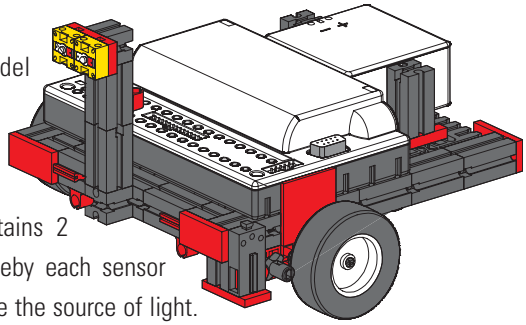
Assignment 2 (Level 2):

- **Create a subprogram for each direction.**
- **Program the model to drive along the path of a square with an edge length of one meter (or 3.28 ft.).**
- **How high is the repeat accuracy?**

Tips:

- First create a subprogram **"Straight"**. Then you can create the other subprograms by copying the first one. The only thing left to do is to adjust the rotational direction of the motors.
- Using a lower speed when turning left and right will increase the accuracy.
- Once again, use the **Pulse Counter** element and the pushbutton sensor at input terminal I1 to count the pulses.
- First load the program into the RAM until you have found out how many pulses are necessary to perform a 90° turn. For one thing loading into the RAM is much faster than loading into the FLASH memory and secondly the Flash memory has a "limited" life span of approx. 100.000 downloads.
- The finished program is called Basic Model 2.rpp.

The Lightseeker



■ Since you have now worked with the Basic Model extensively it is time for your robot to learn to react to signals from its environment. Similar to the moth from our thought experiment of the first chapter it will detect a source of light and follow it. The construction kit contains 2 phototransistors that we will use as light detector. Thereby each sensor affects one motor making it possible for the robot to pursue the source of light.
The program consists of two parts. One part deals with searching for a source of light, the other part implements the pursuit or driving toward the light source. Again we will use subprograms to accomplish this. After switching it on the light-seeking subprogram will be activated. This subprogram will continue until a source of light has been detected. The main program tries to steer the robot toward the source of light. Whenever the direction of the robot greatly deviates from the ideal line, one of the sensors will no longer be illuminated from the light source. This makes the robot change its direction until both sensors can again detect the source of light.

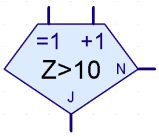
First assemble the light-seeking model according to the construction manual.

Assignment 1 (Level 2):

- **First program the "light-seeking" function. The robot should turn slowly at least 360°. Should it find a light, the robot will stop. Otherwise it will turn another 360° in the other direction. If it is still unable to detect a light source it will wait for 5 seconds and then begin again with its search.**
- **If it successfully detects a source of light the model should drive toward it. If the light source moves to the left or the right, the robot should follow the movements of the light If the robot looses contact the program should return to the light-seeking routine. See if you can attract the robot with a flashlight and guide it through an obstacle course.**

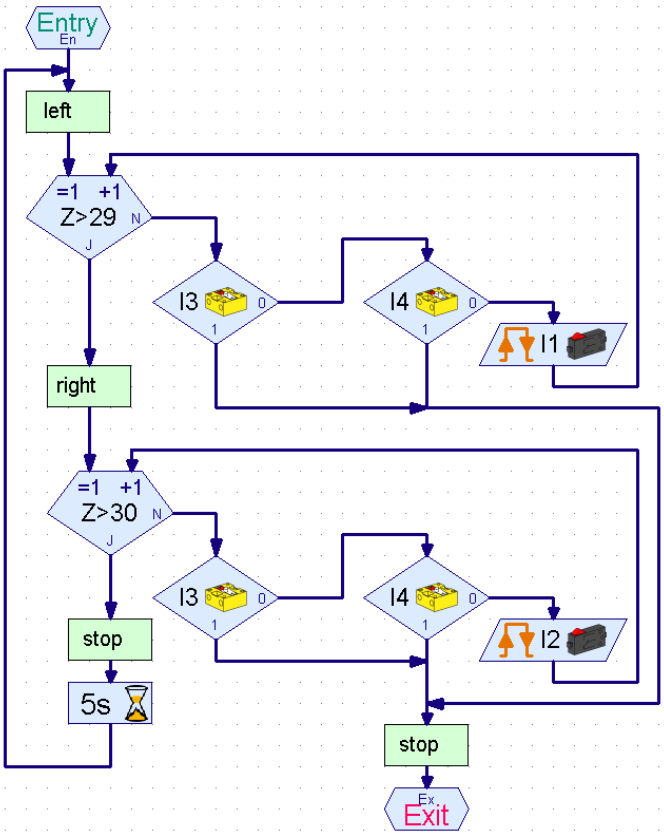
Tips:

- Use the subprograms you already programmed for the Basic Model for the different directions. As soon as the program Basic model 2.rpp has loaded you can find the program Basic model 2 and with it its subprograms in the **element group window** of ROBO Pro under **loaded programs.** You can simply insert these subprograms into your new program.
- For the "light-seeking" subprogram use the **Count Loop** element. (For a description of this element please see the ROBO Pro manual).



Count Loop

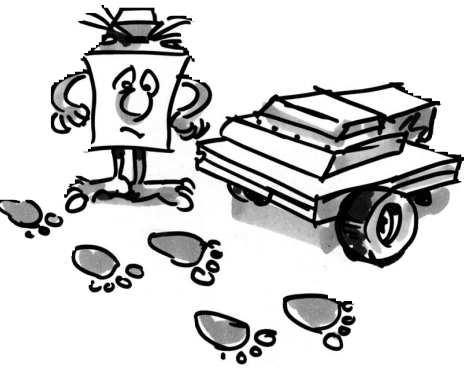
- In the loop between connection "N" and connection "+1" you query the phototransistors and count one pulse at the pulse sensor I1. The loop iterates until the robot has detected light or has rotated 360°. You simply have to try out how many loops it takes until the robot has completed a full rotation. Then enter that value "Z" in the count loop element.
- A second loop is then programmed in exactly the same way for the search in the opposite direction.
- If the robot detects light, it stops, and exits the subprogram.
- Here the complete light-seeking subprogram:



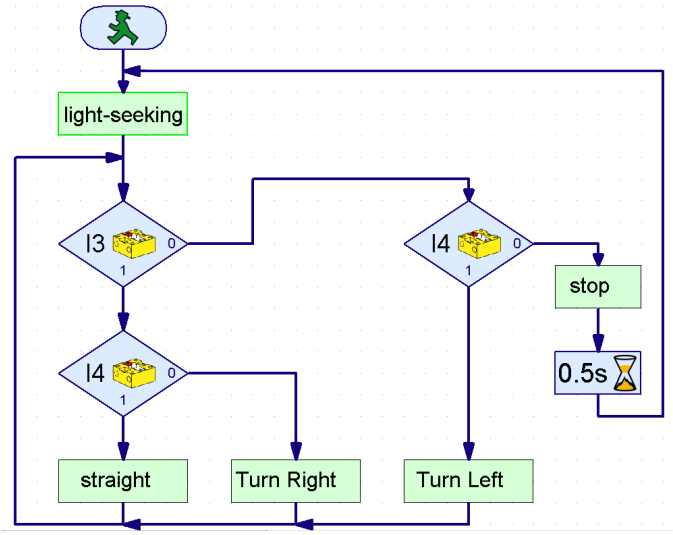
- In the main program you again query the phototransistors and control the motors depending on which phototransistor has detected the light:

Light at I3 and I4	**Straight Ahead**
Light only at I3	**Turn Right**
Light only at I4	**Turn Left**
No light detected	**Stop and return to the light-seeking subprogram**

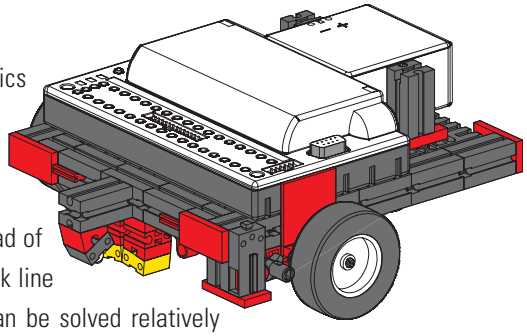
- You can achieve the right and left turns by setting different speeds for M1 and M2 with the same rotational direction. This results in a very smooth driving style.



- The main program finally looks like this:
- You will find the finished program under Lightseeker.rpp.
- Use a flashlight as light source. Take care that the beam of light is not focused too narrowly so that both photosensors are illuminated by the light source. Please note that in a very bright space another source of light such as sunlight through a big window might outshine the flashlight. This might cause the robot to drive right past your lamp toward the brighter light.



The Tracker



■ Search and pursuit are fundamental characteristics inherent to intelligent beings. With the Lightseeker you built and programmed a robot that is able to react to direct signals from its target.
With the Tracker we apply another search principle. Instead of a targeted approach to the source of light we mark a black line that the robot is supposed to follow. This assignment can be solved relatively easily by using the phototransistors. They measure the light reflected by the marking and the motors are corrected accordingly. To make sure this functions accurately the line is illuminated with the lamp. Take care to avoid an unfavorable configuration that causes stray light from the lamp to throw off the photo sensors. The light focusing properties of the incandescent bulb's optical lens is especially helpful in this regard.

Now follow the construction manual to put together the Tracker Model.

Assignment 1

- **First write the subprogram used to find the track. In order to do this the robot should turn around once.**
- **If it is unable to find a track it should drive straight for a short while and then start searching again. The phototransistors are queried for track recognition.**
- **If the robot has detected a track it should follow it.**
- **If the track has run out or the robot lost it, due to a sharp directional change for example, it should start a new search.**

Tips:

- After the lamp has been turned on you have to wait a short while (about one second) before you can query the phototransistors. Otherwise the phototransistor will detect "dark", meaning a track, where there is none, because the reading is done before the lamp is fully lit.
- As track you can use black duct tape that is about 20mm (or 0.787 in.) wide or simply draw a black track with this width on white paper. The turns should not be too tight otherwise the robot will lose sight of the track too frequently. First use the Interface test to make sure that the phototransistors are able to detect your track accurately. Don't forget to switch on the lamp when you do this.
- Adjust the lamp so both phototransistors output the value 1 on a light background, even with motors M1 and M2 turned on. If the battery charge is a little low, the lamp will be somewhat darker when the motors are turned on. If the lamp hasn't been adjusted properly it is possible that the phototransistor will read "dark" even though it hasn't found a track.
- Tracking works in a similar way as light-seeking. You just have to adjust the search so the model drives straight ahead for a bit, after the full rotation, before searching again.
- Please note that the model is supposed to drive straight ahead whenever both phototransistors output the value "dark" (=0).
- You will find the finished program under Tracker.rpp.

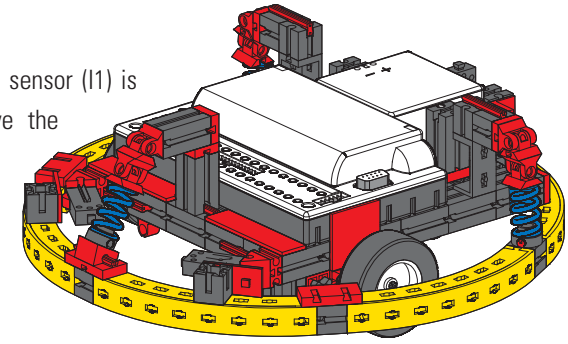
Assignment 2

- **Create a track with curves of varying degrees of tightness. Which is the smallest radius the model can handle?**
- **When correcting the track, experiment with different speeds of M1 and M2. Which combination offers the best result?**
- **Create a round track. Try to optimize the speeds in such a way that the robot achieves the best possible lap time. This assignment lends itself perfectly to a competition between several robots.**

Robot with Obstacle Detection

■ All of the robots we have built so far were able to cover a certain distance as well as follow a light source or a track. But what happens if there is an obstacle in its way? Well, either the obstacle is pushed aside or the robot keeps running against it senselessly until the battery is empty. It would, of course, be much more intelligent if the robot would be able to recognize the obstacle and avoid it accordingly. To accomplish this, the robot is equipped with a flexible circular bumper with three pushbutton sensors. With this bumper it can differentiate if an obstacle is to the left, to the right or behind it.
How it reacts to the obstacle remains only a question of the programming.

First assemble the model "Robot with Obstacle Detection". Only one pushbutton sensor (I1) is necessary to measure the distance traveled. For this reason we can remove the pushbutton sensor I2 from the Basic Model and use it for the obstacle detection.



Assignment 1 (Level 2):

- **First the robot should drive straight ahead. If it encounters an obstacle (I4) to the left, it should back up a bit and then move to the right.**
- **If it encounters an obstacle (I3) to the right, it should back up a bit and then move to the left.**

Tips:

- Obstacle recognition when going backwards will be set aside until later.
- The main program queries the pushbutton sensors. Depending on which pushbutton sensor is activated the model evades to the left or to the right. In each instance this is done using a subprogram.
- The pulse count when turning right should be different from the pulse count when turning left (e.g. 3 pulses to the right, 5 pulses to the left). Otherwise it can happen that the model drives into a corner and gets struck because it turns to the left and right in equal amounts.
- The finished program is called Obstacle 1.rpp.

There are two things the obstacle recognition model does not yet know how to do: It cannot recognize obstacles when going backwards. Equally it does not yet recognize when there is an obstacle directly in front of it. But it could be able to recognize both. If I5 is pressed while going backwards an obstacle is behind the model. Are I3 and I4 activated at the same time while going forward an obstacle is located directly in front of the model. In this case the robot could turn 90° immediately. In all we now have the following possibilities the robot should be reacting to:

Obstacle	Pushbutton Sensor	Reaction
right	**only I3**	**move to the left (turn approx. 30°)**
left	**only I4**	**move to the right (turn approx. 45°)**
in front	**I3 and I4**	**move to the left (turn approx. 90°)**
behind	**I5**	**Only queried when going backwards. Stop, then continue evading as planned.**

Some new program elements such as Operators (e.g. AND, OR) from ROBO Pro Level 3 can help you to solve this problem in an elegant way. Level 3 offers you the possibility to exchange data between different elements by using orange arrows. Switch to that level in the software to take full advantage of these options. It would now be a good idea to take out the ROBO Pro manual and read chapter 5 carefully. After that you'll be ready for the next assignment.

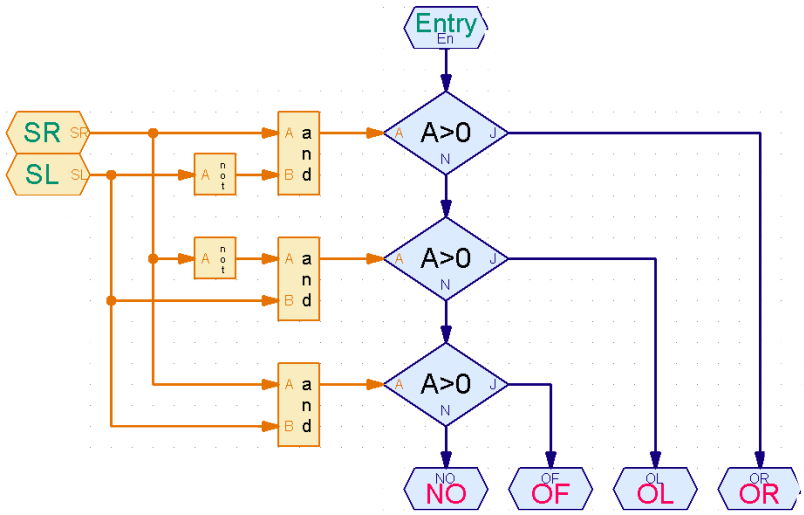
Assignment 2 (Level 3):

- **Modify your obstacle recognition program in such a way that the model will react as described in the table above.**
- **Take advantage of the possibilities offered by ROBO Pro level 3.**

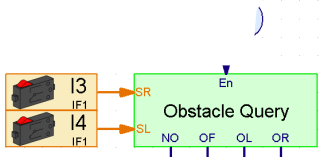
Tips:

- In the "Obstacle Query" subprogram the different sensor combination possibilities are queried using operators. The subprogram has a separate exit for each of the possibilities.

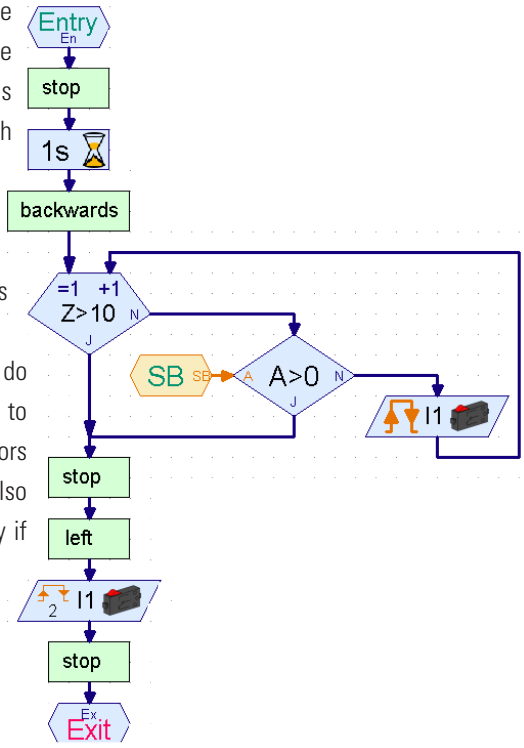
Data Input SR = sensor right
Data Input SL= sensor left
Exit NO = no obstacle
Exit OF = obstacle in front
Exit OL = obstacle left
Exit OR = obstacle right



- Place the orange Sensor elements into the main program and connect them to the subprogram using data inputs, so you can see right away which sensor is being queried.



- In each of the different evasion subprograms, I5 is queried when the robot is going backwards. The model will go backwards until either the set pulse count has been reached or until I5 is pressed. Again, I5 is placed into the main program, so you can see right away in which subprograms the sensor is queried.
- You will find the complete program under Obstacle 2.rpp.



The advantage of the programming technique used in this assignment is that you can see directly in the main program which sensor is being queried in the subprogram. If you would like to change the input you can do so at one single point without having to search through all subprograms to find out where the sensor might be hiding. Additionally, using operators enables you to create very clear logic operations. In principal this is also possible using branch elements. But this becomes confusing very quickly if several cases are queried.

Lightseeker with Obstacle Detection

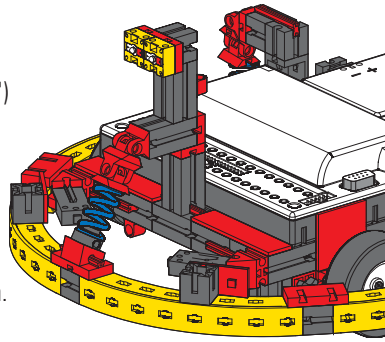
■ We are nowhere near the end of the possibilities offered by the ROBO Mobile Set.
For this reason we will now combine the two functions of light-seeking and obstacle detection. From a scientific point of view the robot will then be equipped with two behaviors. Since both behavior patterns can't be active at the same time, they will receive different priorities. As a rule the robot performs its light-seeking behavior. Should it detect an obstacle, a hazard to the robot, the obstacle recognition behavior becomes active. If everything is clear the robot can continue its light-seeking.

Professional software developers, when faced with such a demanding task won't simply plough ahead with programming. No, they use a specific strategy to develop the program. One of these methods is called "top-down design". With this approach the system is defined as a whole from the top down without dealing with the details at the beginning. It is this method we will use to work out this problem.

Assignment 1 (Level 3):
Teach the robot the following behavior patterns:
- **Search for a source of light.**
- **As soon as you have found it, follow it.**
- **Should an obstacle appear on the way, evade it.**
- **Then begin searching for a source of light again.**

Use the program elements of ROBO Pro level 3 to find the solution.
Apply the "top-down" approach when working on the assignment.

Tips:
First divide the assignment into three parts:
- Query if the robot sees a source of light (subprogram "Light")
- Query if it encounters an obstacle (subprogram "Obstacle")
- Tell the robot what to do depending on these results (subprogram "Driving")



Now consider the different situations the robot is able to perceive to create the subprograms "Light" and "Obstacle". Assign a numerical value to each situation. This value is saved as variable using a Command element. Each situation will result in a reaction that is executed in the "driving" subprogram.

Subprogram light:

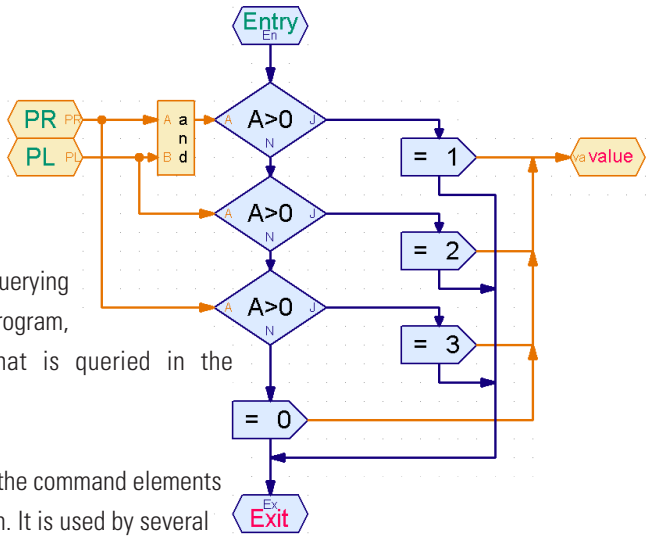
No	Situation	State of the Sensors	Reaction
0	No light source present	I6=0; I7=0	seek light
1	Light source directly in front of robot	I6=1; I7=1	drive straight ahead
2	Light source to the left of robot	I7=1	execute a left turn
3	Light source to the right of robot	I6=1	execute a right turn

Subprogram obstacle:

No	Situation	State of the Sensors	Reaction
4	Obstacle directly in front of robot	I3=1; I4=1	evade 90°
5	Obstacle to the right of robot	I3=1	evade to the left
6	Obstacle to the left of robot	I4=1	evade to the right

Now you simply have to reproduce these conclusions in ROBO Pro using program elements.

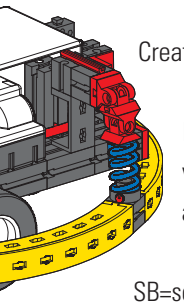
Subprogram light:



PR=phototransistor right
PL=phototransistor left

As before, place the elements for querying the phototransistors into the main program, so you can see right away what is queried in the subprogram.

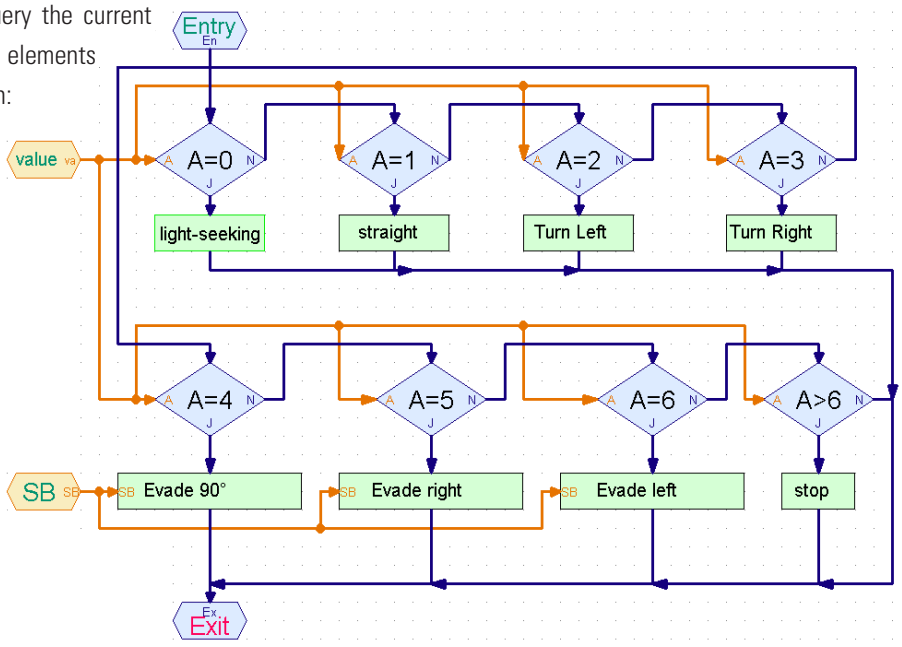
The variable storing the value from the command elements is also placed into the main program. It is used by several subprograms. You connect it to the subprogram using a data output.



Create the subprogram "Obstacle" according to the same principles as the subprogram "Light".

In the subprogram "Driving" you query the current value of the variables using branch elements and program the appropriate reaction:

SB=sensor behind



As a final detail you now have to create the subprograms used in this subprogram.
But wait a minute! Most of them already exist. The light-seeking subprogram for example can be copied from the program for the Lightseeker Model. If you don't remember how to do this, please read chapter 4 of the ROBO Pro manual.

But watch out:
For the Lightseeker Model the phototransistors were connected to input terminal I3 and I4. But now they are on I6 and I7. Additionally, we queried pushbutton sensor I1 to count the pulses when turning left, and I2 when turning right. Now there is only I1 to count the pulses, which works just as well by the way. This means that you will have to adapt the light-seeking subprogram after you have copied it. Since the sensor query is hidden in the subprogram it is easy to miss. This won't happen if you place the inputs into the main program and connect them to the subprogram using data inputs. But you weren't aware of this yet when we worked on the Lightseeker.

The subprograms for evasion also already exist, we wrote them for the Obstacle Recognition Model. Here the pushbutton sensor I5, queried additionally when going backwards, has already been placed into the main program

You can take a look at the finished program under Obstacle-Light.rpp.
At first glance the main program looks very clear and simple. But there is a lot of mental elbow grease behind the subprograms. Still, by using the step-by-step approach of the top-down method you too would be able to tackle such a complex program.

By the way, if you have a friend who owns a ROBO Mobile Set as well, you can go even further with these experiments. Simply mount a source of light on each of the robots. And both robots will be seeking each other.

Robot with Edge Detection

■ We have just seen, in the previous example, how to approach the programming of a more complex program. Now you can turn to another very important behavior of a mobile robot. It is supposed to learn not to fall off the table. In most cases driving against an obstacle won't hurt the robot. But if it falls off a table that is almost three feet high it might be damaged in one way or the other, even if the fischertechnik building blocks are very robust. For this reason the robot will be equipped with sensors that enable it to recognize edges. These edge detectors each consist of a pushbutton sensor that is activated by a rotating wheel. This wheel is also able to move up and down. As soon as the wheel moves over the edge of the table it drops, the pushbutton sensor is no longer activated, the program realizes that the model has reached a precipice and reacts accordingly. The robot has 4 edge detectors in total, enabling it to feel for a precipice on each side while going forward or backwards. As a result this model does not have a pulse sensor to measure the distance traveled. The covered distance will be controlled using the on-time of the motors.

First assemble the model according to the construction manual.

Check carefully if the edge detectors respond correctly:

- when the model reaches the edge of the table and the pushbutton is pressed precisely again
- when the wheel is on the table again.

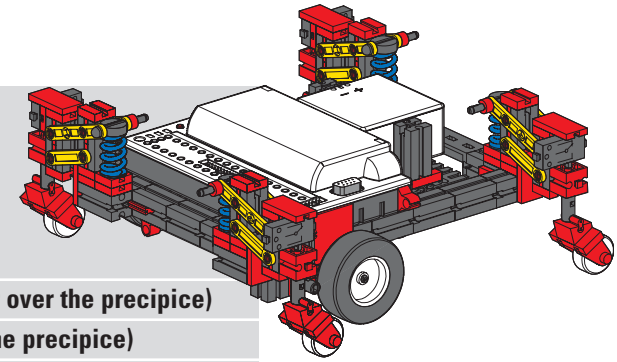
If necessary, one or the other pushbutton sensors might have to be adjusted up or down a bit.

Assignment 1 (Level 3):

- **First consider how the robot should react when reaching a precipice.**
- **Upon closer examination you will realize that there are many possible combinations of sensors located over the precipice.**
- **One of the 4 detectors could be activated, or 2 or 3 different ones at the same time, or even all four sensors.**
- **How should the robot react to each of these possibilities?**

Tips:

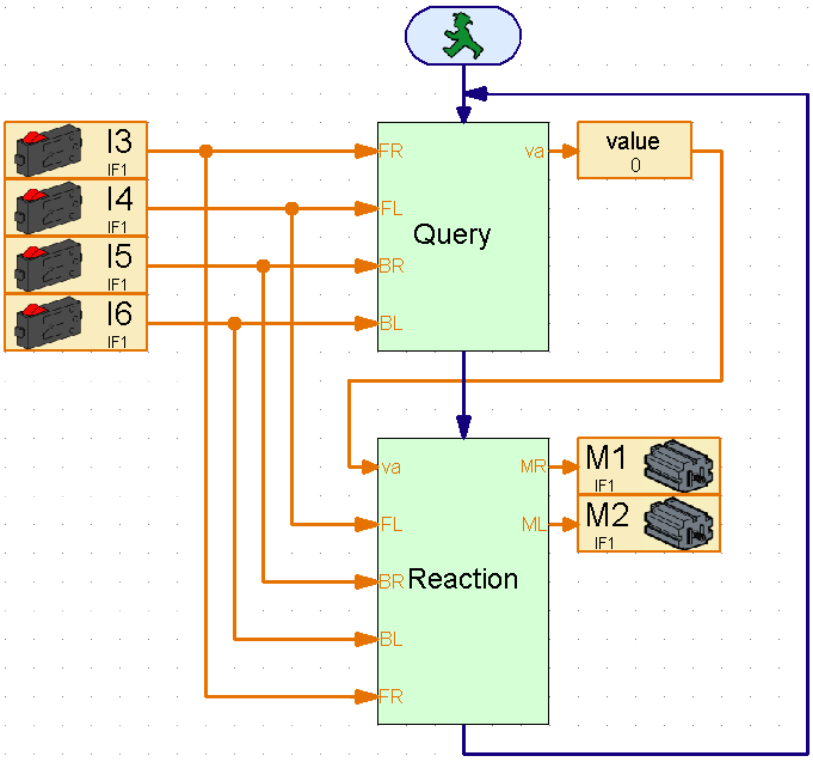
You will find the solution in the following table. Those sensors located over a precipice (pushbutton sensor=0) are marked with an ●. Each combination receives a number. In the program that will be created later each option is assigned the corresponding number. The robot will react to the current situation on the basis of that number. But more about this later. First, only think about how the robot must stand for a certain combination to occur and if it reacts correctly.



No.	Front Right (I3)	Front Left (I4)	Back Right (I5)	Back Left (I6)	Reaction
0					Straight ahead (no sensor over the precipice)
1	●	●	●	●	Stop (all 4 sensors over the precipice)
2	●	●	●		Turn a bit to the right
3	●	●		●	Turn a bit to the left
4	●		●	●	Turn a bit to the left
5		●	●	●	Turn a bit to the right
6	●	●			First back up, then turn to the right
7	●		●		Turn a bit to the left
8	●			●	Turn a bit to the left
9		●	●		Turn a bit to the right
10		●		●	Turn a bit to the right
11			●	●	Drive forward a bit
12	●				First back up, then turn to left right
13		●			First back up, then turn to the right
14			●		Drive forward a bit
15				●	Drive forward a bit

Pretty intense, right? But don't worry there is a finished program for this model using all the advantages ROBO Pro has to offer. It is called Edges.rpp.

The most important elements are located in the main program making it easy for you to understand the sequence as a whole. The complex query of the sensors and the control of the motors are hidden in subprograms. First the main program:



The sequence starts with the query of the 4 pushbutton sensors. All the way to the left you can see which pushbutton sensor is being queried. They are connected to the subprogram via data inputs. The "Query" subprogram determines which pushbutton is pressed and assigns the value detailed in the table above. This value is assigned to a variable with the same name that you'll be able to recognize in the main program. The value of the variable is then sent to the "Reaction" subprogram that controls both motors depending on this value. The pushbutton sensors are read again in the "Reaction" subprogram since the edge sensors are still queried while the model is avoiding the precipice.

If need be, you are now able to change the assignment of the pushbutton sensors as well as the motor outputs on the Interface, without having to dig through the subprograms to see where an input element or a motor symbol might be hiding. Each input and each output appears only once.

This programming technique is especially useful when you would like to use a subprogram for many different models and don't know yet which inputs and outputs to you'll use on the Interface in each case.

If your curiosity has been piqued simply take a look at the subprograms and try to understand them. The programming principle is similar to the "Lightseeker with Obstacle Detection" Model.

Assignment 2 (Level 3):

Load the program into the Interface and let the model drive around a table.

- **Does the model always react in the right way?**
- **Should it behave differently with certain sensor combinations?**
- **Tweak the program as required.**

The Walking Robot

■ After concentrating on wheeled robots we now turn to another method of locomotion that we can use for mobile robots, walking.
The gait of insects lends itself perfectly as model for the movement of "mechanical six-legged walkers". During the so-called tripod walk three of the six legs always lift off the floor simultaneously. The front and back legs on one side lift together with the middle leg of the other side.

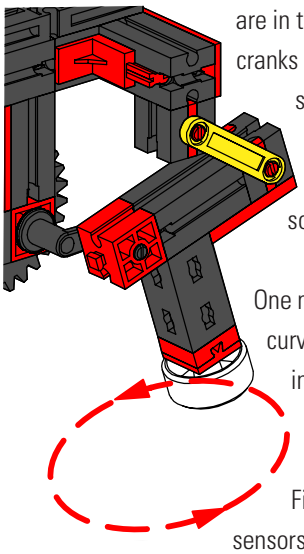
Tripod Walk

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.

The legs that remain on the floor (shown in black), form a stable tripod, so the model always has a solid stance and does not tip over when walking.

The legs of the fischertechnik Walking Robot are constructed as so-called four-joint gears. The design of the four-joints used here is called a "crank-rocker mechanism". Driven by a crank, the floating members of the gears make oscillating movements. The distance between the individual joints and the position of the nadir (this is the bottom of the leg), are arranged in such a way that the foot performs an elliptical movement when the drive crank is turning. This results in a movement resembling a walking step.

The 6 cranks driving the legs have to be adjusted exactly as shown in the construction manual. The three legs that touch the floor at the same time have the same crank position. The cranks of the 3 legs that are in the air at that time are rotated 180° against the other three. The correct position of the cranks in relation to each other ensures that the model is able to walk with the correct sequence of steps, the tripod walk.



The hub nuts used to secure the gears on the shafts have to be screwed in tightly, so the cranks won't become misaligned during walking motion.

One motor each drives the right and the left side of the model (this is necessary for walking curves). For this reason you have to make sure that the middle leg on one side is always in the same position as the two outer legs on the other side. The software controls this synchronization via the pushbutton sensors I1 and I2.

First assemble the model according to the construction manual. Double-check all sensors and motors using the Interface test, to make sure they are connected accurately. Rotational direction of the motors: rotational direction left = straight ahead.

Assignment 1 (Level 1):
Teach the robot how to walk.

- **Program the model to walk straight ahead using the tripod walk.**
- **Use the pushbutton sensors I1 and I2 to synchronize the left and right legs.**
- **When doing so make sure that the two outer legs on one side and the middle leg on the other side are in the same position.**

Tips:

- First bring the legs on the left and the right side into the starting position. Switching on both motors will allow you do this (rotational direction left).
- The sequence should continue only when both pushbutton sensors I1 and I2 are not depressed (This query is necessary as soon as the model is supposed to take its second step).
- Let the motors run until the appropriate pushbutton sensor (I1 for M1, I2 for M2) is depressed again. It is very important here that the model doesn't begin the next step until both pushbutton sensors are depressed. Because only then the legs are in the right position to each other. Provided, of course, that the cranks driving the legs are adjusted correctly as shown in the construction manual.
- Now the sequence can start over and the robot will take its second step. The model will now walk straight ahead until you stop the program.
- You will find the finished program under Walking Robot 1.rpp.

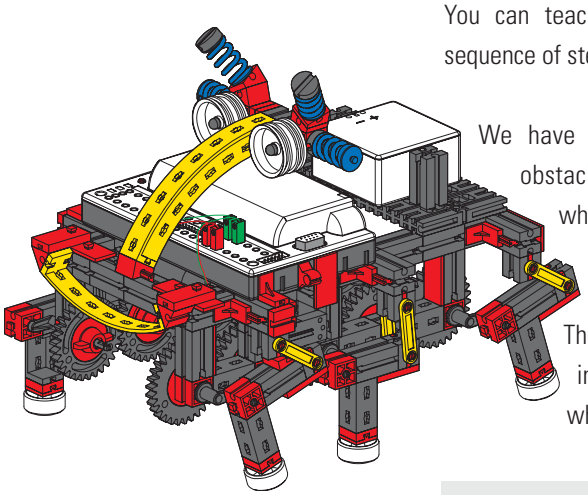
Similar to what we did with the wheeled basic model you can now make the model walk to the left, to the right or backwards by changing the rotational direction of the motors. You can use I1 or I2 to count the steps.

Assignment 2 (Level 2):

- **Program your model to take 10 steps forward, 3 steps to the left, 3 steps to the right and 10 steps back again.**
- **Create an individual subroutine for each direction.**
- **Use the Count Loop element to count the steps.**

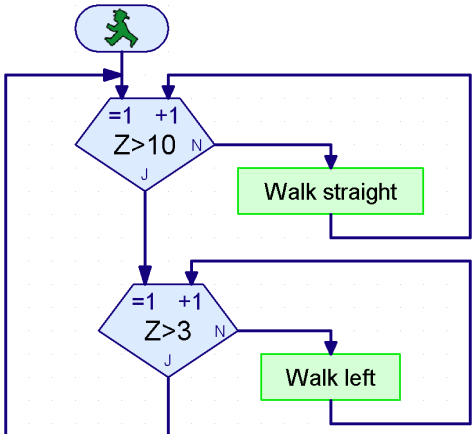
Tips:

- Simply copy the program Walking Robot 1.rpp into a subprogram.
- Copy this subprogram as often as necessary for each of the different walking directions. Change the rotational directions of the motors in each subprogram to make the model move into the desired direction.
- Use the Count Loop element to count the amount of steps for each rotational direction. With each cycle of the subprogram the model takes one step. If the program cycles through the loop with the subprogram 10 times, the model takes 10 steps.



You can teach your walking robot any desired sequence of steps in this way (Walking Robot 2.rpp).

We have already discussed the subject of obstacle recognition in detail for the wheeled robots. So we won't repeat it here. But why don't you try to apply this behavior to the walking robot. The necessary sensors are all included in the construction kit. You can use the wheeled robot as example during the programming. Best of Luck!

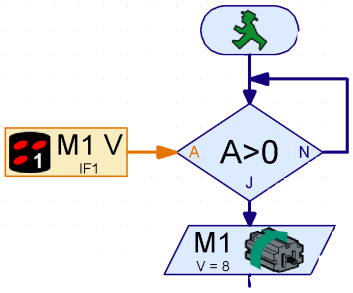


Expansion Possibilities

■ The ROBO Interface offers many more functions than discussed previously with the mobile robots. But you will need additional components that are not included in the construction kit to take full advantage of them. But since they offer very interesting options for the robots we now want to introduce a few of them here.

Handheld Infrared Transmitter

■ The ROBO Interface has an infrared receiver diode for the handheld transmitter included in the IR Control Set Art No. 30344. This enables you to query the buttons on the transmitter in the ROBO Pro software as digital inputs and so for example turn motors on and off.



We have programmed a remote control for the walking robot as a sample program. With the 4 oval arrow buttons on the remote control you can make the model go forward, backwards, to the left and to the right. You only have to load the program Walking Robot IR.rpp into the Interface to get started.

Another ingenious program in connection with the remote control is the program Mobile-Teach-IR.rpp. Using this teach-in program you can control a wheeled robot, e.g. the Simple Robot or the Basic Model. The model will remember the path it traveled and will then be able to repeat it as often as you want it to. But once the program is stopped this saved path will be erased.
It is the "List" program element in ROBO Pro that makes such a program possible. Many values can be stored in this element and then called up again (please also see the ROBO Pro manual). The program itself might be pretty complex but using it is very simple:

1. Load the program Mobile-Teach-IR.rpp into the Flash memory of the ROBO Interface and start it.
2. Press the **M1** ▸/▸▸ button on the remote control. This starts the "learning process".
3. Use the oval arrow buttons to steer the model into the desired direction.
4. Press the **M2** ▸/▸▸ button. This will save the path the robot traveled.
5. Press the **M3** ▸/▸▸ button. This will make the robot travel along the saved path.

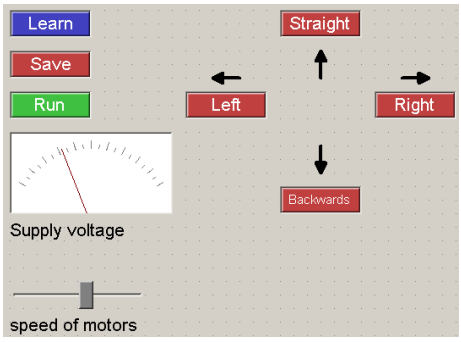
With an application like this, the programming of robots is a breeze! You should note that the saved path will be erased as soon as you stop the program with the Prog button on the Interface.

ROBO RF Data Link

■ The radio interface ROBO RF Data Link Art No. 93295 replaces the interface cable between PC and the Interface with radio data transmission. This is a fine thing indeed. First of all you won't have to keep connecting and disconnecting the cable each time you load a program into the Interface. Secondly, you will be able to run programs wirelessly in online mode. This will make it much easier to find errors than with the regular download operation. And finally you will be able to control the mobile robots in online mode on your screen using a panel in ROBO Pro, similarly to using the IR remote control. But differently from the remote control, the screen additionally displays the data provided by the interface such as values of variables or analog inputs, supply voltage from the battery pack and speed of the motors.

As an example we have modified the teach-in program and are now able to control the wheeled robot using a software panel. The program is called Mobile-Teach-RF.rpp. Of course you can also try it out using the interface cable. But that might be pretty uncomfortable. The operating range of the model is limited, the cable gets tangled and the robot no longer turns correctly. After trying this, you will probably run out right away and get yourself the RF Data Link.

Load the program Mobile-Teach-RF.rpp.
Switch to Panel in the Function Bar of the main program . Then start the program in online mode. Now you'll be able to control and program the model using the buttons in the panel.



1. Press the "Learn" button. This starts the "learning process".
2. Use arrow buttons to steer the model into the desired direction.
3. Press the "Save" button. This will save the path the robot traveled.
4. Press the "Run" button. This will make the robot travel along the saved path

Here as well, the saved path will be lost once the program is terminated.

Please refer to the ROBO Pro manual to learn more about creating panels.

ROBO I/O-Extension

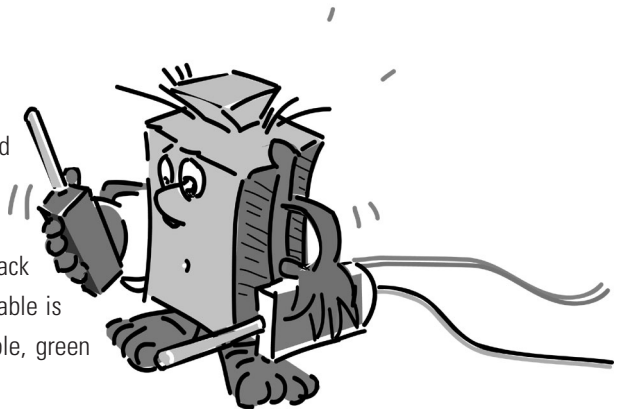
■ Should you build a model with so many sensors and motors that the input and output terminals of the ROBO Interface are not enough, you can connect a ROBO I/O-Extension Art No. 93294 to the Interface. This will provide you with an additional 8 digital inputs, 4 motor outputs and one analog resistance input. A second and a third module can be connected to this I/O-Extension, all controlled using a ROBO Interface. This will provide you with a total of 16 motor outputs, 32 digital inputs, 5 analog resistance inputs, 2 analog voltage inputs as well as 2 inputs for distance sensors.

If this is still not enough for you, you can even control several Interfaces from your PC in online mode. For example, one connected to a serial COM interface, one to the USB port, or 2 Interfaces connected to the USB port and each with up to 3 ROBO I/O-Extensions! Dizzying, isn't it? Chapter 6 of the ROBO Pro manual also explains how the whole thing works.

Trouble Shooting

■ Experimenting is fun. That is, as long as everything is working. Most of the time this will be the case, but unfortunately not always.
It is only when a model is not working right, that you will find out if you truly understand the mechanism and are able to find the fault right away.
With mechanical faults at least there is something to see (assembled incorrectly) or feel (stiff to move). But if electrical problems also arise it becomes much more difficult.
The pros use a series of different measuring instruments to troubleshoot, such as voltmeter or oscillograph. But not everyone has such devices at hand. For this reason we want to try to zero in on a fault with simple means and fix it.

Assembly of Cables



Before we begin with our experiments, we first have to get some of the components from the Fischertechnik construction kit ready. The supplied connectors, for example, are clamped to the individual cable segments.
First we cut the cables to size. We measure the specified lengths and cut the segments accordingly. Each cable is tested after assembly using the battery pack and the lamp. If the lamp lights up after it is connected to the battery, the cable is ok. We will also check if the color-coding is correct, red connector red cable, green connector green cable.

Interface Test

If a program (even a supplied one) does not work in connection with our model we start the Interface test. This utility program enables us to test each input and output separately. Are the sensors working? Are the motors rotating in the right direction? For all our mobile robots the motors are connected in such a way that the wheel or the leg will move forward if the rotational direction=ccw. If everything is ok here as well we'll start looking for a mechanical cause.

Loose Connections

Loose connections are a nasty fault. On one hand it is possible that the way the connectors fit into the sockets is too loose. In this case you can adjust the contact springs a bit using a small screwdriver. But be careful, if you bend them too much the contacts might break or the fit might become too tight.
Another cause of loose connections are the clamp locations where the connector is secured to the cable with a screw. Please tighten the screws carefully! This is also an excellent opportunity to check if any of the thin copper wires might have broken off.

Short-Circuits

On occasion you might also create a short-circuit by connecting cables incorrectly. In that case, nothing will work as it should. The battery pack has a built-in fuse that will interrupt the current when the temperature or the current is too high. In case of overheating the outputs terminals of the Interface will also be shut down.
There can also be a short circuit if you fail to tighten the little screw properly that secures the electrical connector to the cable. The screw might then stick out over the edge of the connector. If you plug two connectors into two adjacent sockets on the Interface and their screws come into contact a short-circuit

will result. For this reason the little screws should always be tightened properly. Always make sure that the screws don't come into contact with each other when plugging in the connectors.

Power Supply

If there are inexplicable interruptions during operation an almost empty battery pack might be the cause. The voltage falls for a shot time when connecting a load (motor on). This causes a reset of the processor on the Interface. When the red LED on the ROBO Interface lights up, the voltage of the power supply is too low. This means the battery pack needs to be charged.

Programming Error

If errors occur in a program you have written yourself and you cannot find a way to explain them, it might be a good idea to be on the safe side and load one of the supplied programs that comes closest to yours. This way you will be able to rule out electrical or mechanical defects. During online mode you can follow the program flow on your screen. If the program gets stuck at a certain point, then this is the place to look for the cause. You might have selected an incorrect input or motor for example or maybe an incorrect value is queried at a branch or a Y/N connection has been switched.

If none of this is successful, you can always contact fischertechnik service (e-mail: info@fischertechnik.de).

Or visit our website at www.fischertechnik.de. There you will find a Forum, Chat, Market place, Gallery and you can join the Fischertechnik Fan-Club for free.

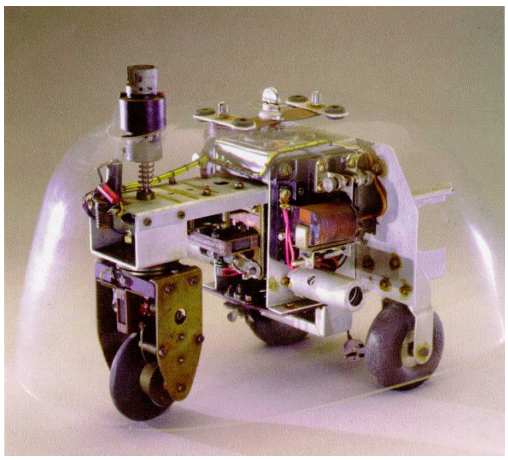
We hope you'll have many hours of fun with the ROBO Mobile Set with plenty of surprises and sudden insights.

Table des matières

Pourquoi avons-nous besoin de robots?

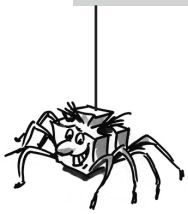
■ Le terme « robot » a été utilisé pour la première fois en 1923 dans le roman « Golem » de Carel Capek. Ce personnage virtuel devait, grâce à ses capacités, effectuer le travail des hommes.
Dans les années 30 et 40 du 20 ème siècle, le robot devient plutôt un type d'automate. Les diverses tentatives pour le doter de caractéristiques humaines, comme p.e. une tête avec des lampes clignotantes en guise d'yeux, ne suscitent plus chez nous aujourd'hui qu'un éventuel sourire. Ces machines ne possèdent que très peu de mobilité et encore moins d'intelligence. Le principe de la commande ayant une grande influence sur la robotique, la conception des robots est devenue plus réaliste avec l'apparition des circuits électroniques. La question de « l'intelligence » du robot fait encore aujourd'hui l'objet de recherches et d'études pour de nombreuses sociétés et universités, ainsi que de nombreux instituts.

■ On attendait les premières ébauches de solution grâce à la cybernétique. Le terme « cybernétique » est dérivé du mot grec Kybernetes. Le Kybernetes était le navigateur sur les bateaux à rames grecs. Il devait déterminer la position du bateau et calculer le cap jusqu'à destination.
Il est donc évident que la cybernétique devait rendre le robot « intelligent ». Comment peut-on vraiment se représenter un tel comportement intelligent ?
Nous voulons essayer d'y réfléchir à l'aide d'une expérience mentale. Chacun a déjà une fois observé le comportement d'une mite dans le cercle de lumière d'une lampe. La mite reconnaît la source lumineuse, vole dans sa direction puis fait une embardée tout juste avant de s'écraser contre la lampe. Il est évident que pour avoir ce comportement la mite doit reconnaître la source lumineuse, découvrir un chemin pour s'y rendre et ensuite voler dans sa direction. Ces capacités sont basées sur les modèles de comportement instinctifs et intelligents de l'insecte.
Essayez à présent de transposer ces capacités dans un système technique. Vous devez reconnaître la source lumineuse (capteurs optiques), effectuer un déplacement (actionner les moteurs) et établir un rapport sensé entre la reconnaissance et le déplacement (le programme).



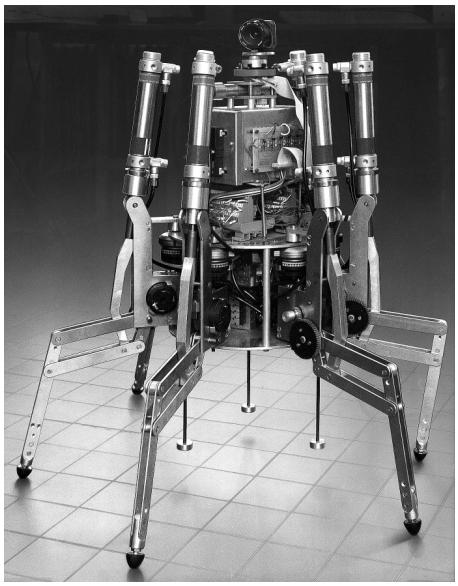
■ Dans les années 50, l'Anglais Walter Grey a mis à exécution l'expérience mentale susmentionnée.
Divers animaux « cybernétiques » possédant un comportement particulier, comme par exemple celui de la mite, furent créés à l'aide de capteurs, de moteurs et de circuits électroniques simples. L'illustration montre une reproduction de la tortue "cybernétique" exposée au musée Smithsonian de Washington.

■ En nous basant sur ces réflexions, nous allons également créer pour nos robots des « modèles de comportement » appropriés et nous essaierons de les faire comprendre à nos robots sous la forme de programmes.



Mais, pourquoi avons-nous désormais besoin de robots mobiles ? Essayons une fois d'appliquer le comportement de notre « mite virtuelle » aux instruments techniques. La recherche de la lumière en constitue un exemple simple. Modifiez la source lumineuse en plaçant une bande claire, la ligne directrice, sur le sol et en orientant les capteurs non plus vers l'avant mais vers l'arrière. Un robot mobile peut s'orienter, à titre d'exemple dans un entrepôt, à l'aide de ce type de lignes directrices. Des informations supplémentaires, p.e. sous forme d'un code-barres à certains endroits de la ligne, incitent les robots à accomplir d'autres actions à ces endroits, comme p.e. la prise ou la pose d'une palette.
De tels systèmes de robots existent réellement. Dans les grands hôpitaux, l'acheminement des consommables, comme p.e. les draps et les taies d'oreiller, nécessite parfois de longs trajets. Le transport de ces consommables par le personnel soignant est onéreux et partiellement associé à un travail corporel lourd. Ces activités restreignent qui plus est le temps consacré aux soins des patients.

■ Les scientifiques s'intéressent depuis quelques années à une autre forme de déplacement très répandue dans la nature, à savoir la marche et la course. Des robots capables de se déplacer sur des jambes ont été développés. Le robot électropneumatique « Achille » développé à l'Académie royale militaire de Bruxelles est un exemple de robots marchant à six jambes. Equipé d'une caméra sur sa partie supérieure et sur ses six jambes, ce robot doit mécaniquement réagir aux obstacles en hauteur ou en profondeur (objets ou trous).
De telles machines marchantes ont pu être utilisées partout là où les véhicules à roues ou à chenilles n'avaient guère aucune chance, p.e. sur des terrains extrêmement accidentés ou meubles, pour escalader des obstacles, monter des escaliers, franchir des fossés ou accéder à des endroits difficiles ou dangereux dans des centrales nucléaires ou lors d'opérations de sauvetage.

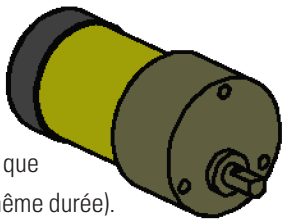


Nous admettons donc que les robots mobiles peuvent tout à fait occuper une place importante dans la société moderne.

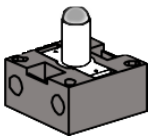
Robots fischertechnik

■ Comment pouvez-vous maintenant construire des robots à partir de notre boîte de construction Fischertechnik? Outre les capteurs (p. ex. les palpeurs) et les acteurs (p. ex. les moteurs), vous avez besoin de nombreuses pièces mécaniques pour construire une maquette. La boîte de construction Fischertechnik ROBO Mobile Set en constitue la base idéale. Les capteurs et acteurs suivants sont inclus dans cette boîte de construction:

Acteurs



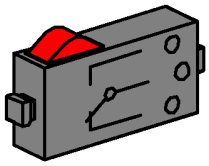
Powermotor:
Deux de ces puissants moteurs à courant continu (9 VCC/2, 4 W) avec transmission intégrée et une démultiplication de 50:1 propulsent les maquettes de robots mobiles (ce qui signifie que le moteur tourne 50 fois et que l'arbre qui se dresse au-dessus du moteur ne tourne qu'une seule fois sur la même durée).



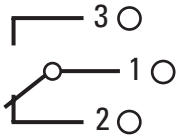
Ampoule lentille:
Cette lampe à incandescence (9 VCC/150 mA) sert à émettre des signaux lumineux simples.
Une lentille qui focalise la lumière émergente est intégrée à l'ampoule. En dirigeant le rayon lumineux sur un capteur de luminosité (transistor photo, voir ci-après), on peut créer une barrière lumineuse qui fait la distinction entre le clair et l'obscur. La lampe peut également être utilisée pour signaler certaines situations ou en tant que lampe clignotante pour émettre des signaux d'avertissement. Dans la boîte de construction, la lampe est utilisée avec 2 transistors photo en tant que capteur spécial pour la reconnaissance de la ligne.

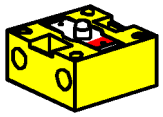
Capteurs

■ Le palpeur est un exemple de capteur numérique. Les grandeurs numériques ne peuvent accepter que 2 situations différentes. Ces situations sont indiquées par les valeurs 0 et 1. Pour le palpeur, « 0 » signifie qu'aucun courant ne circule entre les connexions et « 1 » signifie que le courant circule.



Pour les capteurs, on fait la différence entre les capteurs numériques et les capteurs analogiques.
Le **palpeur** Fischertechnik est conçu comme un commutateur inverseur. C'est la raison pour laquelle il possède 3 connexions. En appuyant sur le bouton rouge, un commutateur qui relie les connexions 1 et 3 l'une à l'autre est mécaniquement activé. Le contact entre les connexions 1 et 2 qui étaient reliées l'une à l'autre à l'état de repos est simultanément interrompu. Les deux positions de sortie possibles peuvent ainsi être consultées:
Fermée à l'état de repos (connexions 1 et 2 occupées)
Ouverte à l'état de repos (connexions 1 et 3 occupées).





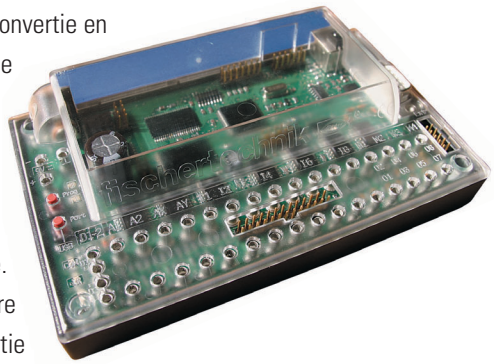
Le **transistor** photo peut être utilisé aussi bien comme capteur numérique que comme capteur analogique. Dans le premier cas, il sert à reconnaître les passages distincts du clair à l'obscur, p. ex. d'une ligne signalisée. Cependant, l'intensité des quantités de lumière peut également être différenciée, le transistor photo fonctionne alors comme un capteur analogique. Les valeurs analogiques peuvent se modifier de n'importe quelle manière entre des valeurs extrêmes. Afin que ces grandeurs puissent être traitées par l'ordinateur, elles doivent être converties dans leurs valeurs numériques correspondantes.

Pour le transistor photo, il s'agit en fait d'un composant dit à semi-conducteurs dont les propriétés électriques dépendent de la lumière. Tout le monde connaît les cellules solaires qui permettent d'obtenir

du courant à partir de la lumière du soleil. Nous pouvons considérer le transistor photo comme l'association d'une cellule solaire miniature et d'un transistor. Les impulsions lumineuses (de photons) qui apparaissent sur le phototransistor produisent un courant très faible qui est ensuite amplifié par le transistor.

Remarque :
Respectez la polarité correcte lors du raccordement du phototransistor : Marquage rouge = Plus.
Tension admissible : 30 V maxi.

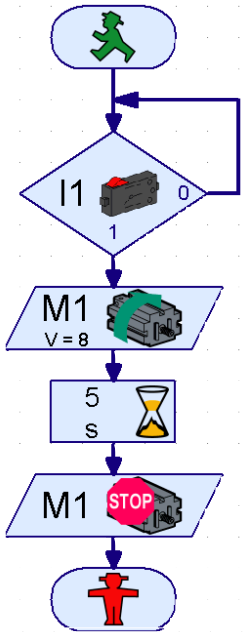
ROBO Interface



■ Vous pouvez raccorder et évaluer divers capteurs et acteurs sur la ROBO Interface. En complément des 8 entrées numériques, la ROBO Interface dispose de plusieurs entrées analogiques. Ainsi, p.e., une valeur de résistance appliquée sur les entrées AX et AY et comprise entre 0 et 5,5 kΩ est convertie en une valeur numérique comprise entre 0 et 1024. Les valeurs mesurées par un capteur de luminosité, p. ex. un transistor photo, sont de ce fait enregistrées et sont disponibles pour un traitement ultérieur. Des tensions comprises entre 0 et 10 VCC peuvent être mesurées aux entrées analogiques A1 et A2.

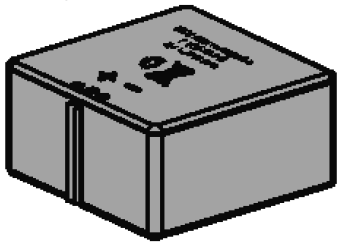
La fonction la plus importante de l'Interface consiste à associer les grandeurs d'entrée. L'Interface a, à cet effet, besoin d'un programme. Le programme décide de quelle manière produire, à partir des données d'entrée, les signaux des capteurs, les données de sortie appropriées, les signaux de commande du moteur, etc. Avec la ROBO Interface, vous disposez d'une puissance de calcul suffisante pour concevoir également des programmes ambitieux.

Logiciel ROBO Pro



■ Il existe une interface de programmation graphique qui vous permet de développer le plus efficacement possible les programmes nécessaires pour l'Interface. Sous le terme « d'interface de programmation » se cache un logiciel qui vous permet de développer très confortablement nos programmes. Ceci s'effectue à l'aide de symboles graphiques. L'ordinateur de la ROBO Interface ne peut en fait exécuter que les instructions émanant de son jeu dit d'instruction des machines. Il s'agit, pour l'essentiel, de structures simples de contrôle d'utilisation extrêmement difficile pour des débutants. Le logiciel ROBO Pro prévoit en conséquence des éléments graphiques qui peuvent ensuite être traduits dans une langue exécutable par l'Interface.

Alimentation électrique



■ Le seul élément supplémentaire dont vous ayez besoin pour la boîte de construction ROBO Mobile Set est l'Accu Set, Art. n° 34969. Il contient l'accupack en tant qu'alimentation électrique mobile pour nos maquettes de robots et un chargeur spécial pour l'accupack.
Il est conseillé de charger immédiatement l'accupack avec le chargeur afin qu'il soit plein lorsque vous voudrez commencer vos expériences.

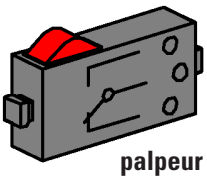
Processus d'expérimentation

■ Nous avancerons progressivement pour faire notre entrée dans le monde fascinant des robots mobiles. Commencez par un assemblage-test simple afin de vérifier les fonctions de base de l'Interface et de l'analyse sensorielle. Vous construirez ensuite des maquettes simples auxquels des tâches définies sont attribuées et vous vous essayerez ensuite avec des systèmes de plus en plus compliqués. Si, à un moment quelconque, le développement des programmes propres devient trop compliqué et dure trop longtemps, il est possible de télécharger tout simplement les exemples de programmes livrés sur l'Interface et de faire fonctionner les robots avec eux.
Un chapitre consacré à la recherche des pannes est inclus à la fin pour que vous ne vous désespériez pas en cas de pannes.

Le soin apporté à la construction et à la mise en service de nos robots est un point très important. Suivez exactement les instructions pour le raccordement des composants électriques et vérifiez de préférence deux ou trois fois si tout est correct. Pour les constructions mécaniques, même les créations personnelles, veillez à obtenir une souplesse et un jeu minimal des organes de commande et des fixations. Votre créativité se réserve le droit d'écrire des programmes personnels et de ce fait de définir un nouveau « comportement ». Ceci n'est limité que par la capacité de mémoire et de calcul du matériel. Les exemples qui suivent proposent quelques suggestions à cet effet.

Premières étapes

■ Après les réflexions théoriques, vous voulez maintenant commencer à effectuer quelques expériences. Certains souhaiteraient sans doute commencer immédiatement, et pourquoi pas avec le grand robot marchant. Ceci est bien entendu possible et, en suivant soigneusement les instructions de montage, la construction de la maquette réussit aussi du premier coup.
Cependant que faire lorsque cela ne fonctionne pas ? Dans ce cas, l'origine de la panne doit systématiquement être recherchée. Mais, avant de vous en préoccuper, vérifiez tout d'abord la liaison entre l'ordinateur et l'Interface.

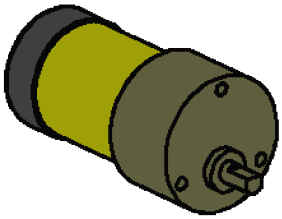


palpeur

Les chapitres 1 et 2 du manuel du logiciel ROBO Pro expliquent comment le logiciel de commande doit être installé sur l'ordinateur et comment l'Interface doit être raccordée. A l'aide du test de l'Interface, testez d'abord les différents capteurs et acteurs.

palpeur

A présent, vous pouvez, p.e. raccorder un palpeur à l'entrée numérique I1 et observer comment l'état de l'entrée se modifie lors de l'activation du palpeur.



Powermotor

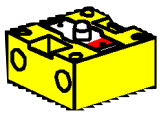
Powermotor

Vérifiez les sorties en reliant un moteur à une sortie Moteur, p. ex. M1. Avec la touche de gauche de la souris, vous pouvez faire tourner le moteur et modifier la vitesse au moyen du régulateur à coulisse.

Fototransistor

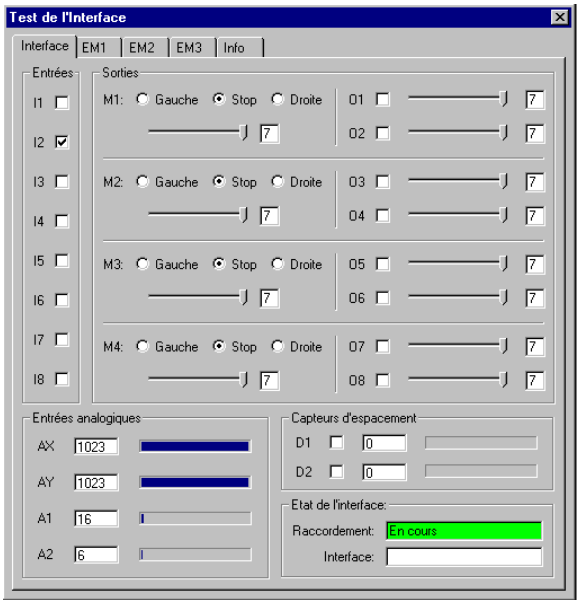
Si vous voulez également tester la sortie analogique AX, vous pouvez utiliser un transistor photo comme capteur analogique.

Alors que la polarité des raccordements ne joue aucun rôle pour le moteur et le palpeur (le moteur tourne au pire dans le mauvais sens), le raccordement exact du transistor photo est absolument nécessaire pour un fonctionnement correct.
Reliez le contact du transistor avec le repère rouge au moyen du pin de raccordement rouge, l'autre contact avec le pin vert. Le second pin vert est inséré dans la prise femelle de l'entrée AX la plus proche du bord de l'Interface ; le second pin rouge, dans la fiche femelle la plus éloignée de AX (Attention : si vous raccordez le transistor photo à une entrée numérique I1-I8, le pin rouge doit être inséré dans la prise femelle la plus proche du bord du boîtier).

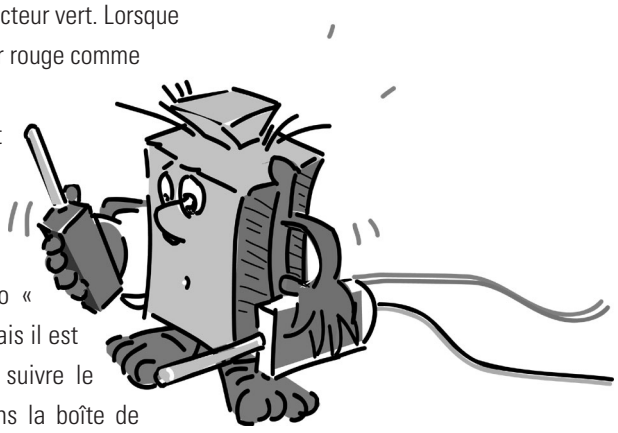


Fototransistor

Vous pouvez maintenant modifier l'éclairement du transistor photo à l'aide d'une lampe de poche et modifier ainsi la déviation de la barre bleue de AX. Si l'aiguille ne bouge pas de sa déviation maximale, vérifiez à nouveau les raccordements du transistor photo. Si, par contre, l'aiguille est aussi sur zéro avec la lampe de poche éteinte, il possible que l'éclairage de la pièce, à savoir la luminosité ambiante, soit trop intense. La déviation se modifie alors lorsque que vous couvrez le transistor photo.



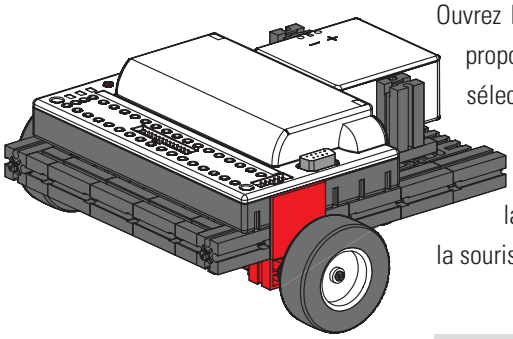
Pour revenir à nouveau brièvement sur la classification des couleurs des pins : lors de l'assemblage, veillez à toujours raccorder un pin rouge au conducteur rouge et un pin vert au conducteur vert. Lorsque la polarité exacte doit être respectée pour le montage, prenez toujours un conducteur rouge comme pôle positif et un conducteur vert comme pôle négatif. Ceci peut paraître quelque peu méticuleux, mais une classification explicite des couleurs facilite considérablement une recherche systématique des pannes.



Nous terminerons ces premiers pas dans le domaine de la robotique avec un programme simple. Le programme expliqué au chapitre 3 du manuel ROBO Pro « Commande d'une porte de garage » n'a certes rien à voir avec les robots mobiles, mais il est parfaitement approprié à l'apprentissage du logiciel ROBO Pro. Afin de pouvoir suivre le programme, il vous suffit de raccorder le moteur et les trois palpeurs inclus dans la boîte de construction ROBO Mobile Set à l'Interface. De plus amples détails sont inclus dans le manuel du logiciel.

Le premier robot simple

■ Après le test de l'Interface et la commande de la porte de garages, vous pourrez enfin mettre les premiers robots en service. Construisez la maquette "Robot simple" avec deux moteurs de commandes, conformément aux instructions de montage. Ceci est très simple et rapide car cette maquette ne contient délibérément que ce qui est nécessaire au pilotage d'un robot. Raccordez les moteurs aux sorties M1 et M2.



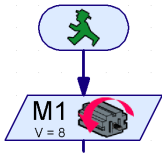
Ouvrez le logiciel ROBO Pro et développez un nouveau programme (FICHIER - NOUVEAU). ROBO Pro propose divers niveaux de difficulté avec lesquels vous pouvez travailler. Ils peuvent être sélectionnés dans le menu de ROBO Pro sous NIVEAU. Le niveau 1 suffit pour le moment.

Un tableau vide s'affiche ainsi que, sur le bord gauche de l'écran, la fenêtre d'élément via laquelle vous pourrez sélectionner les divers éléments du programme avec la touche gauche de la souris et les placer sur la surface de travail. Modifiez les propriétés avec la touche droite de la souris.

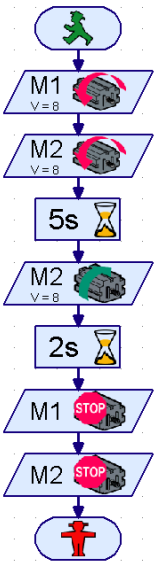
Exercice 1 (Niveau 1) :
Votre "robot simple" doit aller tout droit pendant 5 secondes, puis tourner en rond pendant 2 secondes et ensuite s'immobiliser.

Astuces :

Nous programmerons les premiers robots ensemble, pas à pas:



- Commencez avec l'élément vert représentant un bonhomme en marche. Il symbolise le démarrage du programme.
- Sélectionnez ensuite le symbole du moteur dans la fenêtre d'éléments et positionnez-le sous l'élément de démarrage afin que la ligne de liaison soit automatiquement marquée. Dans la fenêtre d'éléments, réglez la sortie Moteur « M1 », ainsi que le sens de rotation « à gauche » et validez avec OK.



- Placez, de la même manière, sous ce symbole un autre élément Moteur et mettez ainsi en marche le moteur 2.
- Pour attendre pendant un certain temps, utilisez l'élément Attente, placez-le sous le second élément Moteur et réglez l'attente sur 5 secondes.
- Faites ensuite tourner le moteur M2 dans l'autre direction (vers la droite), attendez ensuite 2 secondes et coupez pour finir les deux moteurs. Notre programme se termine avec l'élément de fin, l'élément rouge représentant un bonhomme à l'arrêt. L'illustration montre le déroulement du programme terminé.

Si vous n'êtes pas certain que tout est correct, comparez votre programme avec l'exemple de programme fourni. Pour cela, le programme personnel doit préalablement être sauvegardé et le fichier Robot simple 1.rpp téléchargé à partir du répertoire d'exemples de ROBO Pro (paramètre standard C:\Programme\ROBO Pro\Programmes Exemples\ROBO Mobile Set).
Si tout est correct, le programme est chargé dans l'Interface par téléchargement. Après avoir appuyé sur le bouton Téléchargement, une fenêtre de dialogue s'ouvre. Là, indiquez que le programme doit être chargé dans la mémoire FLASH 1 et doit démarrer immédiatement après le téléchargement.

Notre maquette démarre tout de suite après le téléchargement, tourne ensuite brièvement et s'immobilise. Si vous voulez redémarrer le programme, appuyez brièvement sur la touche Prog de l'Interface. La DEL Prog1 clignote alors de nouveau tant que le programme fonctionne. Elle est ensuite allumée en continu. Le programme dans la mémoire FLASH de l'Interface reste d'ailleurs sauvegardé même en cas de coupure de l'alimentation électrique de l'Interface. Vérifiez ce détail en retirant une fiche de l'accupack. Rebranchez ensuite la fiche, sélectionnez le programme sauvegardé en appuyant sur la touche Prog jusqu'à ce que la DEL Prog 1 s'allume. Appuyez à nouveau sur la touche et lancez ainsi le programme.
Le robot ne fait pas encore grand chose, n'est-ce pas ? Elargissons donc quelque peu l'exercice.

Exercice 2 (Niveau 1) :
Afin que votre robot ne s'immobilise pas déjà après 7 secondes, vous voulez maintenant lui apprendre à danser.

- **Faites-le avancer tout droit, à gauche, à droite, en arrière pendant une durée variable, et ce, à différentes vitesses.**
- **Le processus doit être répété jusqu'à la fin du programme avec la touche Prog sur l'Interface.**

Astuces :

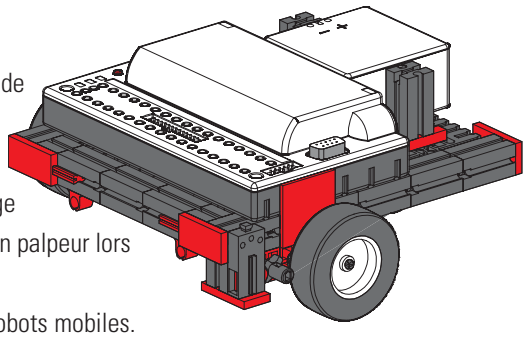
- Inversez à nouveau la polarité des moteurs de manière à ce que le robot se déplace dans les directions souhaitées.
- Vous pouvez régler la vitesse des moteurs entre 1 et 8, dans la fenêtre de propriétés de chaque élément Moteur. Si M1 et M2 tournent à des vitesses différentes dans la même direction, le robot se déplace en formant une courbe.
- Afin de répéter continuellement le programme, tirez une ligne de liaison depuis la sortie du dernier élément de programme jusqu'à la ligne qui rentre dans le premier élément.
- Vous trouverez un exemple terminé sous Robot simple 2.rpp.

Félicitations ! Vous avez construit et programmez vous-même votre premier robot personnel. Il n'est certes pas encore particulièrement intelligent, puisqu'il ne reconnaît aucun obstacle et tombe de la table lorsque vous ne faites pas attention, mais ceci va changer au cours des autres expériences.

Robots roulants intelligents

■ Pour pouvoir reconnaître leur environnement, les robots ont besoin de capteurs. Quelques variantes de robots mobiles sur lesquels l'utilisation de divers capteurs a été testée sont présentées à l'aide des propositions de maquettes suivantes. Il importe à cet effet d'associer aussi bien les états internes du robot, p. ex. la mesure de la distance parcourue par les engrenages à impulsions, que les signaux provenant de l'extérieur, p. ex. la recherche de la lumière ou de traces. Des tâches précises sont à cet effet assignées à chaque maquette. Elles doivent servir de suggestions et vous permettre de vous familiariser avec le matériel. Les programmes relatifs aux diverses tâches se trouvent dans le répertoire ROBO Pro sous \Programmes Exemples\ROBO Mobile Set\. Mais trouvez aussi quelques tâches personnelles à assigner aux maquettes. Lorsque vous aurez terminé avec les exemples suivants, vous aurez sans doute encore plein d'autres idées.

Maquette de base

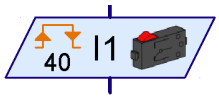


■ Par rapport à notre premier « robot simple », le maquette de base est plus stable et plus robuste. Il contient en outre 2 capteurs de mesure de distance se composant chacun d'un interrupteur et d'un engrenage d'impulsions. L'engrenage d'impulsions est relié à l'arbre du moteur et actionne 4 fois un palpeur lors de chaque rotation du moteur.
Cette maquette sert de base pour les autres maquettes de robots mobiles.

Construisez la maquette de base selon les instructions de montage. Procèdez très soigneusement pour le montage. Lorsque toute la partie mécanique est terminée, vérifiez la souplesse des moteurs en reliant directement pendant une courte durée chaque moteur à l'accumulateur sans passer par l'Interface.

Exercice 1 (Niveau 1) :

- **Programmez l'Interface de manière à ce que la maquette se déplace tout droit pendant 40 impulsions. Pour mesurer les impulsions, utilisez le interrupteur sur l'entrée I1.**
- **Mesurez la distance parcourue par la maquette et calculez la distance parcourue par impulsion.**
- **Répétez l'essai 3 fois et inscrivez dans un tableau les variations des valeurs.**



Astuces :

- Démarrez tout d'abord les deux moteurs (sens de rotation à gauche).
- Pour compter les impulsions au niveau de I1, utilisez l'élément de programme **Compteur d'impulsions**.
- Comptez les deux types d'impulsion (0-1 en appuyant sur l'interrupteur, 1-0 en le relâchant). Vous pouvez paramétrer ceci dans la fenêtre de propriétés sous Type d'impulsion. Augmentez ainsi la précision de la mesure de la distance.
- Coupez ensuite les moteurs et terminez le programme.
- Vous trouverez le programme terminé sous Maquette de base1.rpp.

Résultat :

	Nombre d'impulsions	Distance parcourue	Distance/Impulsion
Essai 1	**40**		
Essai 2	**40**		
Essai 3	**40**		

Vous pouvez observer en gros que la maquette parcourt environ une distance d'un centimètre par impulsion.

Vous savez aussi quel sens de rotation, vous devez paramétrer pour les différentes maquettes afin que la maquette avance dans une certaine direction. Inscrivez ces informations dans le tableau suivant afin de ne pas avoir à y réfléchir à chaque modification du sens de rotation. Si vous câblez les maquettes exactement comme indiqué dans les instructions de montage, le sens de rotation à gauche signifie pour chaque moteur que la roue tourne vers l'avant. C'est ainsi que les moteurs sont programmés dans tous les programmes d'exemple.

Complètez le tableau :

Sens de marche de la maquette	Sens de rotation M1	Sens de rotation M2
Vers l'avant	**A gauche**	**A gauche**
Vers l'arrière		
A gauche		
A droite		
Arrêt		

Afin de ne pas devoir placer deux éléments Moteur sur l'écran lors de chaque changement de direction, vous pouvez créer pour chaque sens de marche un sous-programme qui assume cette tâche. Ceci simplifie considérablement la programmation. Le chapitre 4 du manuel du logiciel ROBO Pro explique comment créer des sous-programmes. Dès que vous aurez lu ce chapitre, vous pourrez vous risquer à faire l'exercice suivant. Passez maintenant au **niveau 2** dans ROBO Pro.

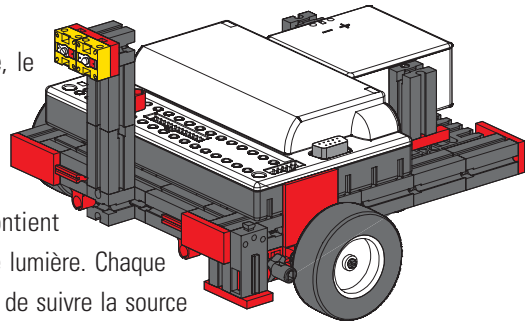
Exercice 2 (Niveau 2):

- **Créez un sous-programme pour chaque sens de marche.**
- **Programmez la maquette de manière à ce qu'elle parcourre un carré d'un mètre de côté.**
- **Quelle est la fidélité de reproduction ?**

Astuces :

- Créez tout d'abord un sous-programme **Vers l'avant**. Vous pouvez créer les autres sous-programmes en copiant celui-ci. Il ne vous restera alors qu'à ajuster les sens de rotation des moteurs.
- Utilisez, pour tourner à gauche ou à droite, une vitesse réduite. Ceci augmente la précision.
- Pour compter les impulsions, utilisez à nouveau l'élément **Compteur d'impulsions** et l'interrupteur sur l'entrée I1.
- Chargez tout d'abord le programme dans la mémoire vive pour faire un essai jusqu'à ce que vous sachiez de combien d'impulsions vous avez besoin pour effectuer une rotation de 90°. Premièrement, le chargement dans la mémoire vive est plus rapide que le chargement dans la mémoire FLASH et deuxièmement la mémoire FLASH n'a qu'une durée de vie « limitée » d'environ 100.000 téléchargements.
- Le programme terminé s'appelle Maquette de base2.rpp.

Le viseur de lumière



■ Après votre étude approfondie de la maquette de base, le robot doit apprendre à réagir aux signaux de l'environnement. Tout comme la mite dans notre expérience mentale du premier chapitre, il doit reconnaître une source lumineuse et la suivre. La boîte de construction contient 2 transistors photo que nous utilisons comme détecteur de lumière. Chaque capteur agit à cet effet sur un moteur afin de lui permettre de suivre la source lumineuse. Le programme est composé de deux parties. La première contient la recherche d'une source lumineuse et la seconde indique comment suivre et viser cette source lumineuse. Pour cela, des sous-programmes sont à nouveau utilisés. Le sous-programme Recherche de la lumière s'active après le démarrage. Ce sous-programme n'est quitté que lorsqu'une source lumineuse a été trouvée. Le programme principal essaie alors de diriger le robot vers la source lumineuse. Lorsque la direction du robot dévie beaucoup de la ligne idéale, l'un des capteurs n'est plus illuminé par la source lumineuse. Le robot corrige alors son sens de marche afin que les deux capteurs puissent à nouveau reconnaître la source lumineuse.

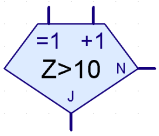
Construisez tout d'abord la maquette Viseur de lumière selon les indications des instructions de montage.

Exercice 1 (Niveau 2) :

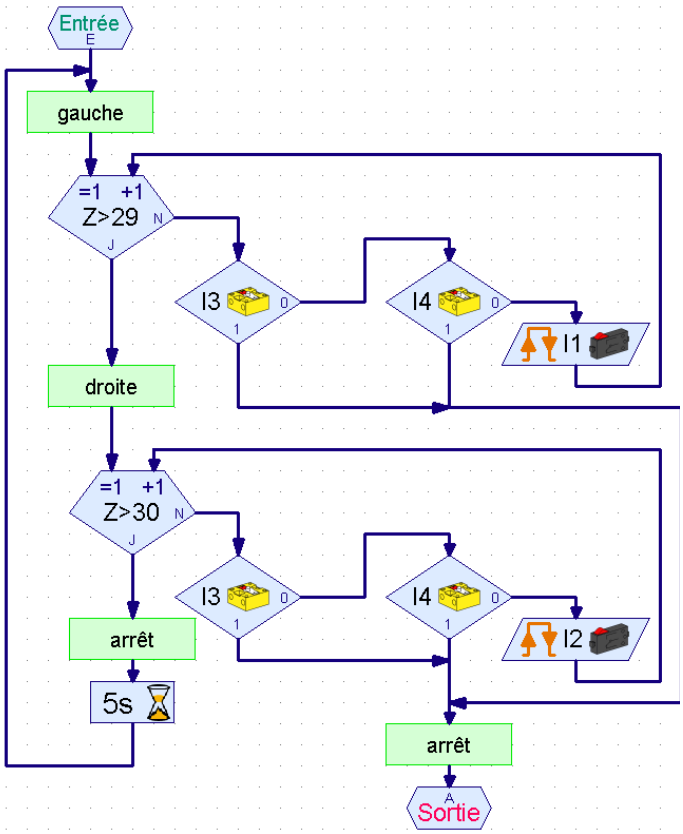
- **Programmez tout d'abord la fonction « Recherche de la lumière ». Le robot doit pour ce faire effectuer une lente rotation de 360° minimum. Si une lumière est trouvée pendant la recherche, le robot s'arrête. Sinon, il effectue une nouvelle rotation de 360° dans l'autre direction. S'il ne trouve toujours pas de source lumineuse, il doit attendre pendant 5 secondes puis commencer une nouvelle recherche.**
- **Si la recherche de la lumière est couronnée de succès, le robot doit viser la source lumineuse. Si la source lumineuse se déplace vers la gauche ou vers la droite, le robot doit suivre les mouvements de la lumière. S'il perd le contact, le programme doit à nouveau être démarré avec la recherche de la lumière. Essayez d'attirer le robot avec une lampe de poche et d'effectuer un parcours comportant un obstacle.**

Astuces :

- Utilisez, pour les différents sens de marche, les sous-programmes que vous avez déjà créés pour la maquette de base. Dès que le programme Maquette de base2.rpp s'ouvrira, vous trouverez le programme Maquette de base2 dans la **fenêtre des groupes d'éléments** de ROBO Pro sous le **programme chargé** et en dessous les sous-programmes contenus dans le programme Maquette de base2. Vous pouvez simplement ajouter ces sous-programmes dans votre nouveau progamme.
- Pour le sous-programme « Recherche de la lumière », utilisez l'élément **Boucle de comptage**. (Reportez-vous au manuel ROBO Pro pour la description de l'élément).
- Dans la boucle entre la sortie « N » et la sortie « +1 », envoyez une requête aux transistors photo et comptez une impulsion sur le l'interrupteur I1. La boucle est parcourue jusqu'à ce que le robot ait trouvé une lumière ou ait effectué une rotation de 360°. Déterminez simplement combien de fois il doit parcourir la boucle pour effectuer une rotation complète et paramètrez la valeur « Z » correspondante dans l'élément Boucle de comptage.
- Programmez ensuite une seconde boucle exactement comme pour la recherche mais avec un sens de rotation inversé.
- Si le robot trouve une lumière, il s'arrête et quitte le sous-programme.
- Voici le sous-programme Recherche de lumière complet :



Boucle de comptage

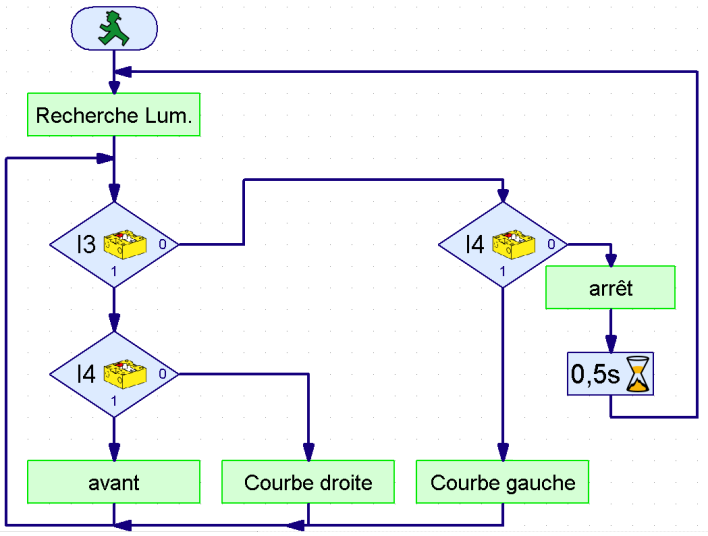


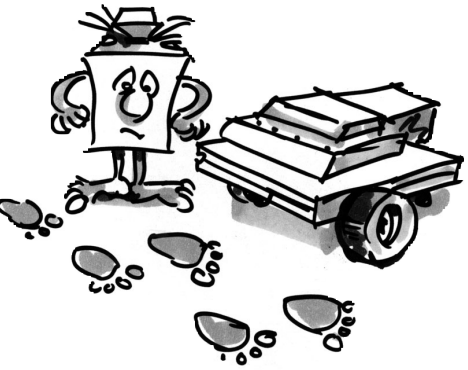
- Interrogez à nouveau les transistors photo dans le programme principal et actionnez les moteurs en fonction du transistor photo qui reconnaît la lumière :

Lumière à I3 et I4	**Vers l'avant**
Lumière uniquement à I3	**Courbe vers la droite**
Lumière uniquement à I4	**Courbe vers la gauche**
Aucune lumière reconnue	**Arrêt, retour au sous-programme Recherche de la lumière**

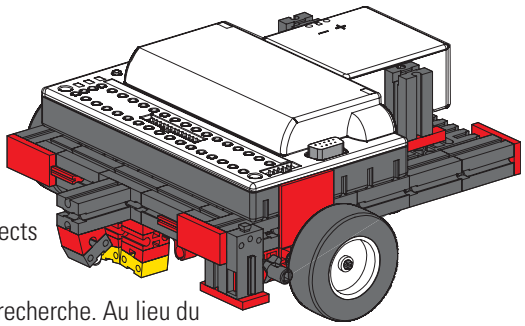
- créez la courbe vers la droite et vers la gauche avec des vitesses différentes de M1 et de M2 avec un sens de rotation identique. Il en résulte un style de marche très harmonieux.

- Voici à quoi ressemble maintenant le programme principal :
- Vous trouverez le programme terminé sous Recherche de la lumière.rpp.
- Utilisez une lampe de poche comme source de lumière. Essayez de ne pas focaliser le rayon lumineux trop étroitement afin que les deux photocapteurs soient illuminés par la source lumineuse. Veillez à ce que, dans les pièces très claires, votre lampe de poche ne perde pas de focalisation en raison des autres sources lumineuses, p.e. la lumière solaire entrant par une grande fenêtre. Dans ces conditions, le robot passe devant votre lampe et se dirige vers la lumière plus claire.





Le viseur de trace



■ Rechercher et suivre sont des propriétés essentielles des créatures intelligentes. Avec le viseur de lumière, vous avez créé et programmé un robot qui réagit aux signaux directs de son objectif.

Avec le viseur de trace, nous utilisons un autre principe de recherche. Au lieu du parcours exact jusqu'à la source lumineuse, nous traçons une ligne noire qu'un robot doit suivre. Cet exercice peut être facilement résolu avec les transistors photo. La lumière réfléchie du tracé est mesurée et les moteurs sont ensuite corrigés. Afin que ceci fonctionne également correctement, la ligne est éclairée avec la lampe. Veillez à ce que les photocapteurs ne soient pas éblouis par la lumière diffusée par la lampe en raison d'une disposition défavorable. Dans ce contexte, la focalisation de la lumière de la lentille optique de la lampe à incandescence s'avère particulièrement avantageuse.

Construisez maintenant la maquette Viseur de trace selon les instructions de montage.

Exercice 1 :

- **Créez tout d'abord un sous-programme avec lequel la trace sera recherchée. Le robot doit pour cela tourner en rond une fois.**
- **S'il ne trouve pas de trace, il doit avancer tout droit et effectuer une nouvelle recherche. Des requêtes sont envoyées aux transistors photo pour la reconnaissance de la trace.**
- **Lorsque le robot a trouvé une trace, il doit la suivre.**
- **Si la trace est finie ou si le robot la perd, p.e. en raison d'un grand changement de direction, la recherche doit être recommencée.**

Astuces :

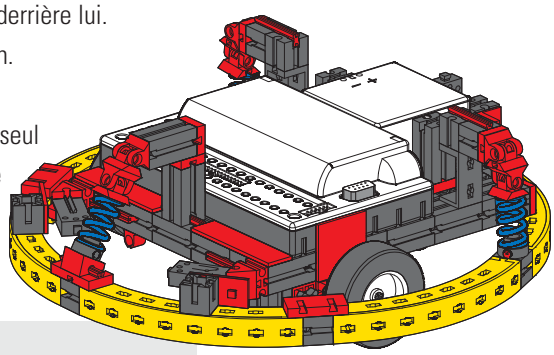
- Il faut attendre un court instant (env. une seconde) après l'allumage de la lampe avant d'envoyer une requête aux transistors photo. Dans le cas contraire, le transistor photo reconnaît « l'obscurité », à savoir une trace là où il n'y en a pas, parce que l'interrogation a lieu avant que la lampe soit réellement claire.
- Comme trace, utilisez un ruban isolant d'env. 20 mm de large ou dessinez avec un feutre une trace noire de cette largeur sur une feuille de papier blanc. Les courbes ne doivent pas être trop étroites, sinon le robot perd trop souvent la trace.
- Vérifiez d'abord avec le test de l'Interface si votre trace est correctement reconnue par les transistors photo. N'oubliez pas d'allumer la lampe.
- Règlez la lampe de manière à ce que les deux transistors photo produisent la valeur de 1 sur un fond clair, même lorsque les moteurs M1 et M2 sont en marche. Si votre accumulateur est un peu faible, la lampe devient un peu plus foncée lors du démarrage des moteurs. Si elle n'est pas correctement réglée, il est possible qu'un transistor photo signale de l' « obsurité » bien qu'il n'ait trouvé aucune trace.
- La recherche d'un trace fonctionne de la même façon que la recherche de la lumière. Vous devez seulement ajuster la recherche afin qu'en cas d'échec de la recherche la maquette avance un peu vers l'avant après avoir accompli un tour complet avant de continuer la recherche.
- N'oubliez pas que la maquette doit avancer tout droit pour suivre la trace lorsque les deux transistors photo produisent la valeur « Obscur » (=0).
- Vous trouverez le programme fini sous Viseur de trace.rpp.

Exercice 2 :

- **Créez une trace avec différentes courbes étroites. Dans quel rayon la maquette arrive-t-elle encore à aller droit ?**
- **Faites l'expérience en corrigeant la trace avec différentes vitesses de M1 et M2. Quelle combinaison produit le meilleur résultat ?**
- **Créez une trace avec un tracé rond. Essayez d'optimiser les vitesses afin que le robot parcoure le cercle aussi rapidement que possible. Cet exercice convient en particulier pour un concours avec plusieurs robots.**

Robot avec reconnaissance d'obstacle

■ Tous les robots construits jusqu'à présent peuvent parcourir une certaine distance et suivre une source lumineuse ou une trace. Cependant, que se passe-t-il lorsqu'un obstacle se trouve sur le chemin ? Et bien, soit l'obstacle est poussé de côté, soit le robot bute vainement contre lui jusqu'à ce que l'accumilateur soit vide. Il serait bien entendu plus intelligent que le robot reconnaisse et évite l'obstacle. Pour ce faire, le robot est muni d'une tige de butée circulaire mobile avec trois palpeurs. Avec cette tige de butée, il peut détecter si un obstacle se trouve à gauche, à droite ou derrière lui.
La manière dont il doit réagir alors n'est ensuite qu'une question de programmation.



Construisez tout d'abord la maquette « Robot avec reconnaissance d'obstacle ». Un seul interrupteur (I1) est nécessaire pour mesurer la distance. L'interrupteur I2 est enlevé de la maquette de base et utilisé pour la reconnaissance d'obstacle.

Exercice 1 (Niveau 2) :

- **Le robot doit tout d'abord avancer tout droit. S'il bute à gauche contre un obstacle (E4), il doit revenir un peu en arrière puis faire une embardée vers la droite.**
- **S'il bute à droite contre un obstacle (E3), il doit revenir un peu en arrière puis faire une embardée vers la gauche.**

Astuces :

- La reconnaissance d'obstacle en marche arrière n'est pas prise en compte pour le moment.
- Le programme principal envoie des requêtes aux interrupteurs. En fonction de l'interrupteur activé, la maquette fait une embardée vers la gauche ou vers la droite. Ceci se produit à chaque fois dans un sous-programme.
- Le nombre d'impulsions lors d'une rotation vers la droite doit être différent de celui de la rotation vers la gauche (p. ex. 3 impulsions vers la droite, 5 impulsions vers la gauche). Dans le cas contraire, la maquette risque d'aller dans un coin et de ne plus en sortir parce qu'elle tourne toujours autant vers la gauche que vers la droite.
- Le programme terminé s'appelle Obstacle1.rpp.

Il y a deux choses que le détecteur d'obstacles ne peut pas encore faire : En marche arrière, il ne reconnaît encore aucun obstacle. Lorsque les obstacles se trouvent juste devant lui, il ne les remarque pas non plus. Il pourrait cependant reconnaître ces deux choses. Si I5 est activé pendant la marche arrière, un obstacle se trouve derrière la maquette. Si I3 et 34 sont simultanément activés pendant la marche avant, un obstacle se trouve juste devant la maquette. Dans ce cas, le robot pourrait tout de suite tourner sur 90°. Les possibilités auxquelles le robot doit maintenant réagir sont les suivantes :

Obstacle	Palpeur	Réaction
rà droite	Uniquement I3	Faire une embardée vers la gauche (rotation d'env. 30°)
à gauche	Uniquement I4	Faire une embardée vers la droite (env. 45°)
devant	I3 et I4	Faire une embardée vers la gauche (env. 90°)
derrière	I5	Uniquement interrogé en marche arrière. S'arrêter puis faire une embardée comme prévu

Quelques nouveaux éléments du programme, comme p. ex. les opérateurs (p. ex. ET, OU, = du ROBO Pro Niveau 3) peuvent vous être très utiles pour résoudre cet exercice avec facilité. Le niveau 3 permet également d'échanger des données entre divers éléments via les flèches orange. Passez, en conséquence, à ce niveau dans le logiciel. Prenez ensuite le manuel ROBO Pro et lisez attentivement le chapitre 5. Vous êtes alors prêt pour l'exercice suivant.

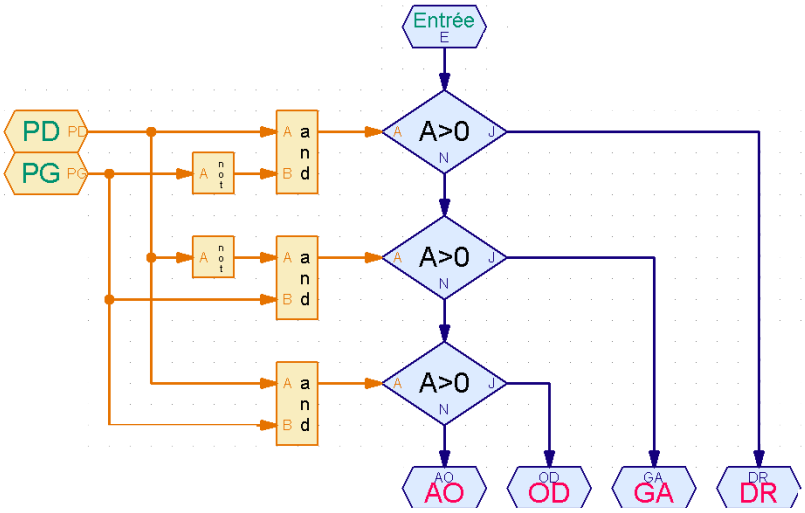
Exercice 2 (Niveau 3) :

- **Modifiez le programme Obstacle afin que la maquette réagisse selon les indications du tableau ci-dessus.**
- **Utilisez pour ce faire les possibilités du Niveau 3 de ROBO Pro.**

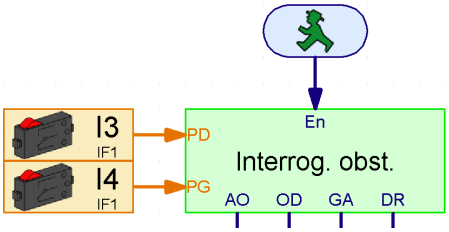
Astuces :

- Des requêtes sont envoyées aux différentes combinaisons d'interrupteurs dans le sous-programme « Interrogation obstacle » à l'aide des opérateurs. Le sous-programme dispose d'une sortie pour chaque possibilité.

Saisie des données PD = Interrupteur à droite
Saisie des données PG = Interrupteur à gauche
Sortie AO = aucun obstacle
Sortie OD = obstacle devant
Sortie GA = obstacle à gauche
Sortie DR = obstacle à droite

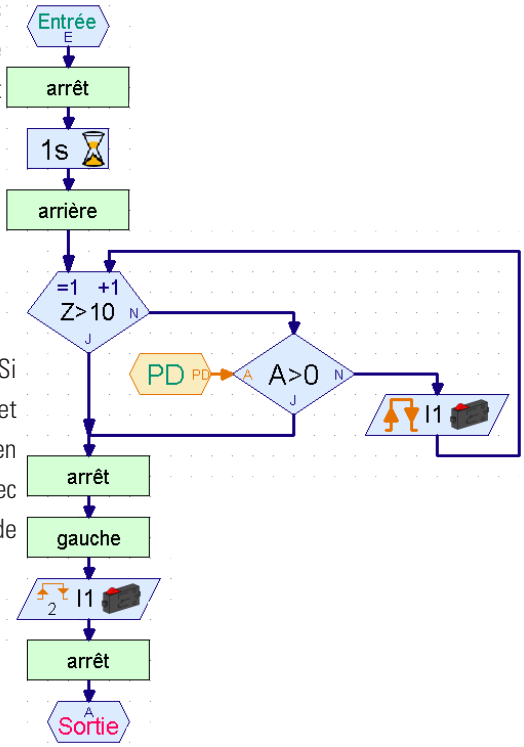


- Pour reconnaître immédiatement le palpeur à interroger, placez les éléments orange de l'interrupteur dans le programme principal et associez-les en saisissant les données dans le sous-programme.



- Une requête est envoyée à L5 pendant la marche arrière dans les différents sous-programmes d'évitement. La maquette recule jusqu'à ce que le nombre d'impulsions défini soit atteint ou que I5 soit pressé. I5 est remis dans le programme principal afin savoir immédiatement dans à quel sous-programme une requête doit être envoyée.
- Vous trouverez le programme complet sous Obstacle2.rpp.

L'avantage de la technique de programmation utilisée dans cet exercice est que vous voyez directement dans le programme principal à quel interrupteur une requête doit être envoyée et dans quel sous-programme. Si vous voulez modifier la saisie, vous ne devez le faire qu'à un seul endroit et non chercher dans tous les sous-programmes où l'interrupteur pourrait bien s'être caché. On peut en outre créer des associations logiques claires avec les opérateurs. Certes, ceci est aussi possible avec les éléments de branchement, mais devient rapidement confus lorsqu'une requête est envoyée pour plusieurs cas.



Viseur de lumière avec reconnaissance d'obstacle

■ Les possibilités offertes par le ROBO Mobile Set sont loin d'être épuisées.
C'est pourquoi les deux fonctions Recherche de la lumière et Reconnaissance d'obstacle doivent à présent être combinées. D'un point de vue scientifique, le robot dispose alors de deux modes de comportement. Cependant, dans la mesure où les deux types de comportement peuvent être simultanément actifs, des priorités différentes leur sont attribuées. Le robot est normalement réglé sur la recherche de la lumière. S'il reconnaît un obstacle, donc un danger pour lui, le comportement Evitement d'un obstacle s'active. Lorsque tout est dans la zone verte, le robot peut reprendre la recherche de la source lumineuse.

Lorsque les concepteurs professionnels de logiciels se mettent à une tâche aussi ambitieuse, ils ne programment pas à la légère, mais ils emploient une certaine stratégie pour développer le programme. L'une de ces méthodes s'appelle le « projet Top-Down ». Pour ce type de procédure, tout le système est défini du haut vers le bas, sans se préoccuper de tous les détails, du moins au début. Nous emploierons également cette méthode pour résoudre ce problème.

Exercice 1 (Niveau 3) :

Apprenez les comportements suivants au robot :

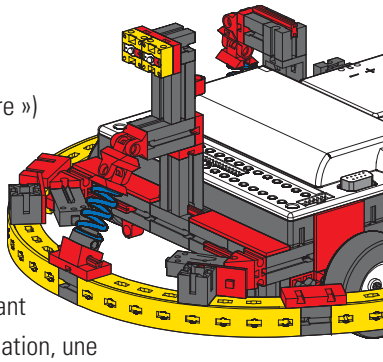
- **Recherchez une source lumineuse.**
- **Dès que vous l'avez trouvée, suivez-là.**
- **Si un obstacle se trouve sur son chemin, évitez-le.**
- **Recherchez ensuite à nouveau une source lumineuse**

Pour le résoudre, utilisez les éléments du programme de ROBO Pro Niveau 3.
Résolvez l'exercice « du haut vers le bas » selon la méthode Top-Down.

Astuces :

Divisez l'exercice en trois parties :

- Demandez si le robot voit une source lumineuse (sous-programme « Lumière »)
- Demandez s'il bute contre un obstacle (sous-programme « Obstacle »)
- En fonction de ces résultats, indiquez au robot ce qu'il doit faire (sous-programme « Avancer »)



Pour les sous-programmes « Lumière » et « Obstacle », réfléchissez maintenant aux différentes situations que le robot peut percevoir. Attribuez, à chaque situation, une valeur numérique que vous sauvegarderez dans une variable à l'aide d'un élément de commande. Chaque situation engendre alors une réaction qui est exécutée dans le sous-programme « Avancer ».

Sous-programme Lumière :

N°	Situation	Etat des capteurs	Réaction
0	Aucune source lumineuse présente	I6=0; I7=0	Rechercher la lumière
1	Source lumineuse juste devant le robot	I6=1; I7=1	Avancer tout droit
2	Source lumineuse à gauche du robot	I7=1	Tourner sur la gauche
3	Source lumineuse à droite du robot	I6=1	Tourner sur la droite

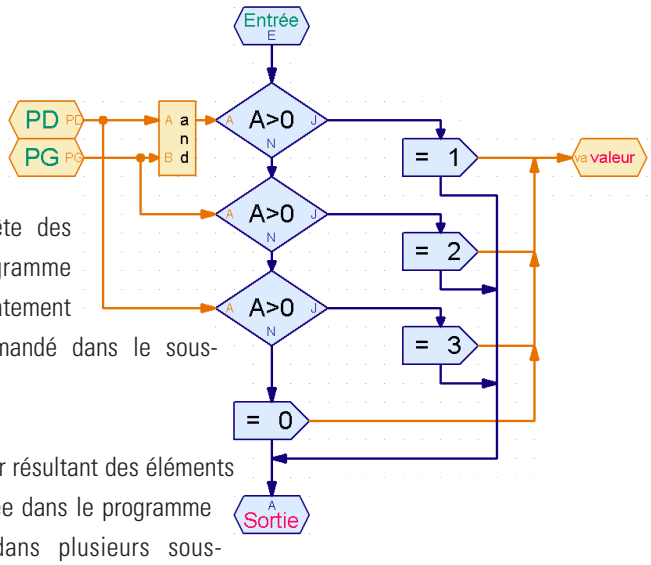
Sous-programme Obstacle :

N°	Situation	Etat des capteurs	Réaction
4	Obstacle directement devant le robot	I3=1; I4=1	Faire une embardée de 90°
5	Obstacle à droite du robot	I3=1	Faire une embardée sur la gauche
6	Obstacle à gauche du robot	I4=1	Faire une embardée sur la droite

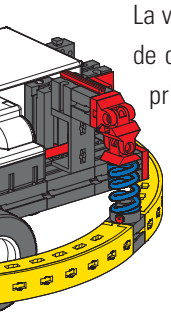
Il ne vous reste maintenant plus qu'à représenter ces connaissances dans ROBO Pro avec les éléments du programme.

Sous-programme Lumière :

PD= transistor photo de droite
PG= transistor photo de gauche



Remettez les éléments de requête des transistors photo dans le programme principal afin de pouvoir immédiatement reconnaître ce qui doit être demandé dans le sous-programme.

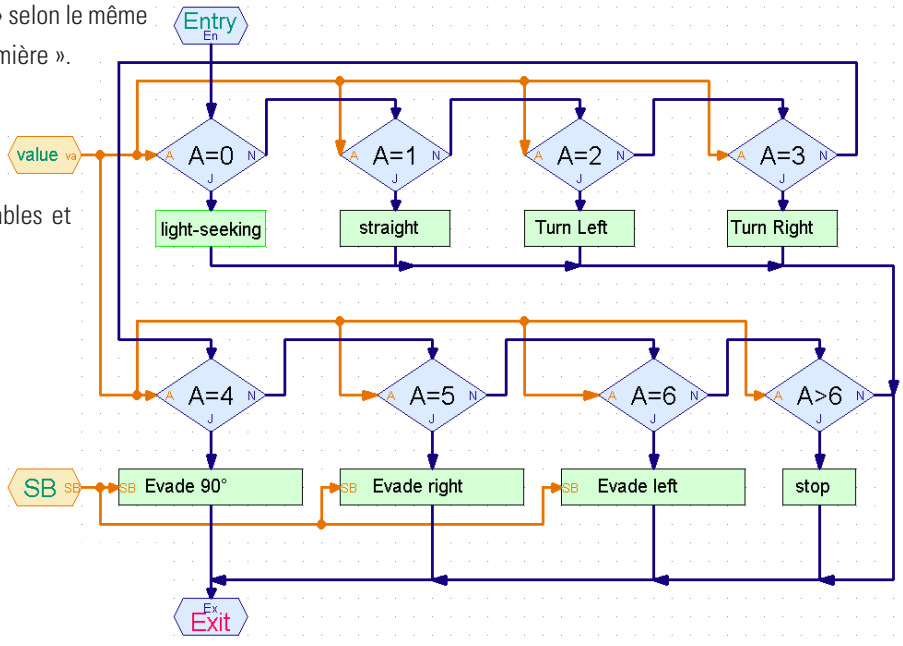


La variable qui sauvegarde la valeur résultant des éléments de commande est également placée dans le programme principale. Elle est utilisée dans plusieurs sous-programmes. Associez-la au sous-programme via une sortie de données.

Créez le sous-programme « Obstacle » selon le même principe que le sous-programme « Lumière ».

Dans le sous-programme « Avancer », demandez, au moyen des éléments de branchement, la valeur actuelle des variables et programmez la réaction appropriée :

PD= Interrupteur derrière



Pour finir, vous devez encore créer les sous-programmes qui sont utilisés dans ce sous-programme. Mais, une minute ! Il existe déjà presque tous. Vous pouvez p.e. copier le sous-programme Recherche de la lumière dans le programme de la maquette Viseur de lumière. Si vous ne savez plus comment faire, relisez le chapitre 4 du manuel ROBO Pro.

Mais attention :
Pour la maquette Viseur de lumière, les transistors photo I3 et I4 étaient raccordés. Mais maintenant ils se trouvent sur I6 et I7. En outre, pour le comptage des impulsions, l'interrupteur I1 a été interrogé lors d'une rotation vers la gauche et l'interrutpeur I12 pour une rotation vers la droite. Maintenant, il ne reste plus que I1 pour compter les impulsions, ce qui en fait fonctionne tout aussi bien. Vous devez donc ajuster le sous-programme Recherche de la lumière après l'avoir copié. La requête envoyée aux interrupteurs étant cachée dans le sous-programme, il est facile de pas la voir. Ceci ne se produit plus si vous placez les entrées dan le programme principal et que vous les associez avec le sous-programme en saisissant les données. Mais, pour le viseur de lumière, vousd ne connaissiez pas encore cette possibilité.

Les sous-programmes d'évitement existent déjà également, notamment pour la maquette Reconnaissance d'obstacle. L'interrupteur I5 qui est aussi interrogé pour la marche arrière est sorti.

Vous pouvez regarder le programme fini sous Obstacle-Lumière.rpp.
Le programme principal a l'air très clair et très simple au premier regard. Et pourtant, il y a de quoi s'arracher les cheveaux dans les sous-programmes. Mais, en procédant progressivement avec la méthode Top-Down, vous auriez pu aussi résoudre un problème complexe.

Au fait… Si vous avez un ami qui possède aussi une boîte de construction ROBO Mobile Set, vous pouvez pousser l'expérimentation encore plus loin. Une source lumineuse est simplement installée sur chacun des deux robots. Les robots se cherchent alors mutuellement.

Robot avec reconnaissance d'arête

■ Après avoir vu dans le dernier exemple comment procéder lors de la programmation d'un problème complexe, vous pouvez maintenant vous consacrer à un autre comportement très important d'un robot mobile. Il doit notamment apprendre à ne pas tomber d'une table. Buter contre un obstacle ne fait, dans la plupart des cas, aucun effet sur le robot. Cependant, tomber d'une table d'une hauteur de presque un mètre pourrait l'endommager d'une manière ou d'une autre, bien que les modules Fischertechnik soient très solides. C'est pour cette raison que le robot est muni de capteurs lui permettant de reconnaître les arêtes. Ces détecteurs d'arêtes sont composés d'un interrupteur qui est actionné par un engrenage mobile à paliers. Cet engrenage peut aussi se déplacer vers le haut et vers le bas. Dès que l'engrenage dépasse du bord de la table, il s'abaisse, l'interrupteur n'est plus actionné, le programme reconnaît que la maquette se trouve dans une abîme et réagit en conséquence. Le robot possède au total 4 détecteurs d'arêtes afin de pouvoir palper les abîmes sur les deux côtés en marche avant, comme en marche arrière. En revanche, cette maquette n'a pas de capteur d'impulsions pour mesurer la distance. La distance parcourue est commandée au moyen de la durée de fonctionnement des moteurs.
Construisez tout d'abord la maquette selon les instructions de montage.

Vérifiez avec précision que les détecteurs d'arêtes se déclenchent correctement :

- lorsque la maquette s'approche du bord de la table et que l'interrupteur est à nouveau pressé,
- lorsque la roue se trouve à nouveau sur la table.

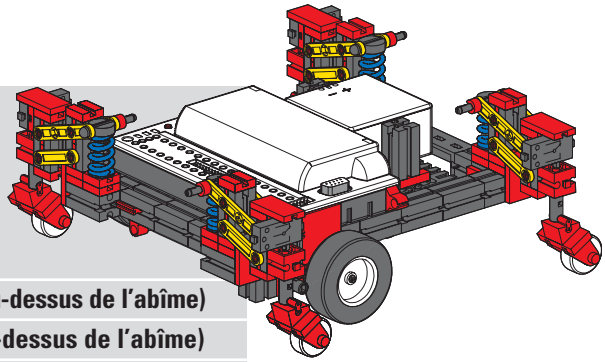
Vous devez éventuellement ajuster l'un ou l'autre des interrupteurs vers le haut ou vers le bas.

Exercice 1 (Niveau 3) :

- **Réfléchissez tout d'abord à la façon dont le robot doit réagir lorsqu'il arrive dans une abîme.**
- **En réfléchissant davantage, vous vous rendrez compte qu'il existe de nombreuses possibilités de combinaisons dans lesquelles les capteurs pourraient se trouver dans une abîme. Il est possible qu'un des 4 détecteurs se déclenche ainsi que 2 ou 3, voire les 4 interrupteurs simultanément.**
- **Comment le robot doit-il à chaque fois réagir ?**

Astuces :

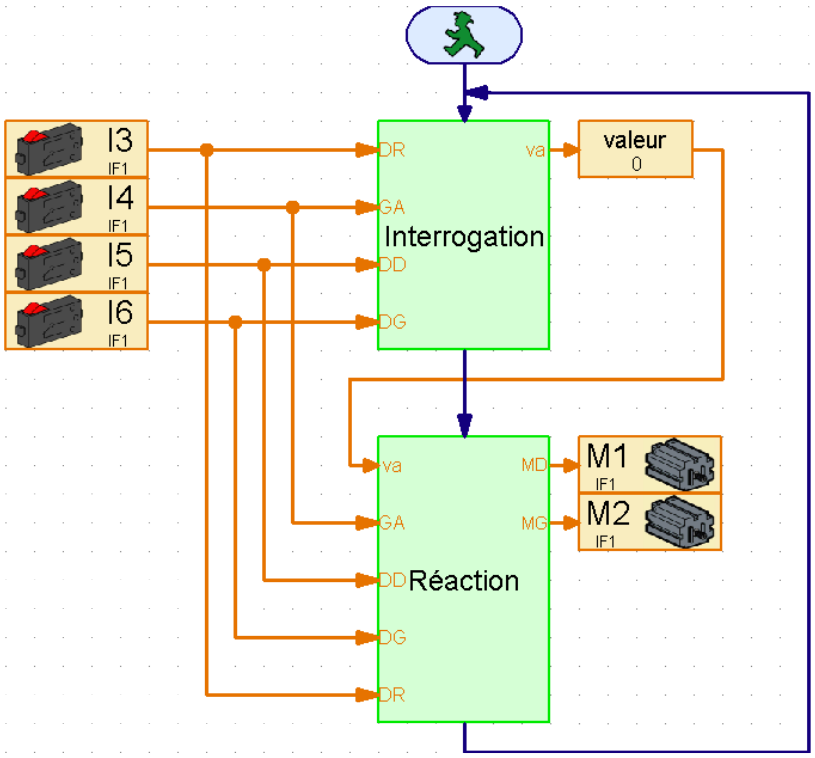
Vous trouverez la solution dans le tableau suivant. Les capteurs qui se trouvent au-dessus de l'abîme (palpeur=0) sont cochés. Un nombre est attribué à chaque combinaison. Le chiffre correspondant est attribué à chaque possibilité dans le programme qui sera créé ultérieurement. Le robot réagit à la situation présente à l'aide du nombre. Mais nous y reviendrons plus en détail. Réfléchissez tout d'abord uniquement à la façon dont se tient le robot afin que la combinaison correspondante se produise et qu'il réagisse correctement.



N°	Devant à droite (I3)	Devant àgauche (I4)	Derrière à droite (I5)	Derrière àgauche (I6)	Réactions
0					Avancer (aucun capteur au-dessus de l'abîme)
1	●	●	●	●	Arrêter (les 4 capteurs au-dessus de l'abîme)
2	●	●	●		Tourner légèrement vers la droite
3	●	●		●	Tourner légèrement vers la gauche
4	●		●	●	Tourner légèrement vers la gauche
5		●	●	●	Tourner légèrement vers la droite
6	●	●			Reculer puis tourner vers la droite
7	●		●		Tourner légèrement vers la gauche
8	●			●	Tourner légèrement vers la gauche
9		●	●		Tourner légèrement vers la droite
10		●		●	Tourner légèrement vers la droite
11			●	●	Avancer légèrement
12	●				Reculer puis tourner vers la gauche
13		●			Reculer puis tourner vers la droite
14			●		Avancer légèrement
15				●	Avancer légèrement

Assez violent, n'est-ce pas ? Mais n'ayez crainte, il existe pour cette maquette un programme complet qui utilise tous les avantages de ROBO Pro. Il s'appelle Arêtes.rpp.

Les éléments les plus importants se trouvent dans le programme principal afin que vous puissiez comprendre l'ensemble du processus. Les requêtes complexes envoyées aux interrupteurs et la commande des moteurs sont cachées dans les sous-programmes. Voici tout d'abord le programme principal :



Le déroulement commence par l'interrogation des 4 interrupteurs. Voyez, à l'extrémité gauche, les interrupteurs auxquels une requête est envoyée. Ils sont associés au sous-programme via les entrées des données. Le sous-programme « Interrogation » détermine les interrupterus qui sont pressés et donne la valeur indiquée dans le tableau. Des variables de même nom que vous pouvez reconnaître dans le programme principal sont attribuées à cette valeur. La valeur des variables est transmise au sous-programme « Réaction » qui commande ensuite les deux moteurs en fonction de cette valeur. Les interrupterus sont également insérés dans le sous-programme « Réaction », car les capteurs d'arêtes sont aussi interrogés pendant que la maquette fait une embardée.

Vous pourriez maintenant modifier sans problèmes l'affectation des palpeurs sur l'Interface, ainsi que les sorties Moteurs sans devoir faire une recherche dans tous les sous-programmes dans lesquels un élément d'entrée ou un symbole de moteur s'est caché. On ne rencontre qu'une seule fois chaque entrée et chaque sortie.

Vous pouvez avant tout utiliser cette technique de programmation lorsqu'un sous-programme doit être utilisé pour de nombreuses maquettes différentes et lorsque vous ne connaissez pas exactement les entrées et les sorties qu'il convient d'utiliser pour cela dans l'Interface.

Si votre curiosité est maintenant excitée, parcourez simplement les sous-programmes et essayez de les comprendre. Le principe de programmation est similaire à celui de la maquette « Viseur de lumière avec reconnaissance d'obstacle ».

Exercice 2 (Niveau 3) :

Chargez le programme sur l'Interface et faites avancer le robot sur une table.

- **La maquette réagit-elle toujours correctement ?**
- **Devrait-elle se comporter différemment avec certaines combinaisons de touches ?**
- **Optimisez, si nécessaire, le programme.**

Le robot marchant

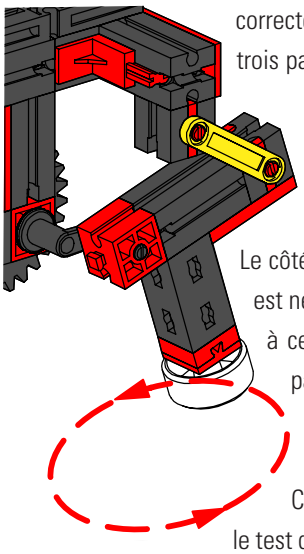
■ Après nous être occupés dans les détails des robots roulants, passons à une autre forme de déplacement que nous pouvons utiliser pour les robots mobiles, à savoir la marche.
L'allure des insectes convient à la perfection comme maquette de « créature mécanique à six pattes ». Pour marcher sur trois pattes, trois des six pattes se soulèvent simultanément du sol, la patte avant et la patte arrière d'un même côté en même temps que la patte du milieu de l'autre côté :

Marche sur trois pattes

Les pattes qui restent au sol (représentées en noir) forment un trépied stable afin que la maquette soit toujours debout en toute sécurité et ne se reverse pas en marchant.

Les pattes du robot marchant Fischertechnik sont conçues de manière à constituer un quadrilatère articulé. La forme de construction du quadrilatère articulé utilisé ici s'appelle un « mécanisme à manivelle et bielle oscillante ». Propulsés par une manivelle, les éléments mobiles de l'organe de commande effectuent des mouvements oscillants. Les écartements entre les diverses articulations et la position de la base (à savoir l'extrémité inférieure de la patte) sont choisis de manière à ce que la base décrive un mouvement elliptique lorsque la manivelle de commande tourne. Il en résulte un mouvement qui ressemble à un pas effectué en marchant.

Les 6 manivelles qui commandent les pattes doivent être exactement ajustées selon les indications des instructions de montage. Les trois pattes qui reposent en même temps sur le sol ont la même position de manivelle. Les manivelles des 3 pattes qui sont alors soulevées tournent sur 1800. La position correcte des manivelles l'une par rapport à l'autre garantit que la maquette peut marcher sur trois pattes à la cadence correcte.



Les contre-écrous avec lesquels les roues dentées sont fixées sur les axes doivent être correctement serrés afin que les manivelles ne se déplacent pas pendant la marche.

Le côté droit et le côté gauche de la maquette sont chacun commandés par un moteur (ceci est nécessaire pour le fonctionnement des manivelles). C'est pourquoi il convient de veiller à ce que la patte centrale d'un côté soit toujours dans la même position que les deux pattes extérieures de l'autre côté. Cette synchronisation est commandée par ordinateur via les palpeurs I1 et I2.

Construisez tout d'abord la maquette selon les instructions de montage. Vérifiez, avec le test de l'Interface, que tous les interrupteurs et moteurs sont correctement raccordés. Sens de rotation des moteurs : sens de rotation à gauche = vers l'avant.

Exercice 1 (Niveau 1) :
Apprenez au robot à marcher.

- **Programmez la maquette afin qu'elle marche tout droit sur trois pattes.**
- **Utilisez les interrupteurs I1 et I2 pour synchroniser les pattes gauches et droites.**
- **Veillez pour cela à ce que les deux pattes extérieures d'un côté et que la patte centrale de l'autre côté soient toujours dans la même position.**

Astuces :

- Mettez tout d'abord les pattes du côté gauche et du côté droit dans leur position initiale. Mettez les deux moteurs en marche (sens de rotation vers la gauche).
- Le processus ne doit se poursuivre que lorsque les deux interrupteurs I1 et I2 ne sont pas pressés (cette interrogation est nécessaire dès que la maquette veut effectuer le second pas).
- Laissez les moteurs tourner jusqu'à ce que l'interrupteur approprié (I1 pour M1 et I2 pour M2) soit à nouveau pressé. Il est important à ce effet que la maquette ne commence le pas suivant que lorsque les deux interrupteurs sont pressés. Les pattes sont alors dans la position correcte l'une par rapport à l'autre. La condition préalable est que les manivelles qui commandent les pattes soient correctement ajustées, comme indiqué dans les instructions de montage.
- Le processus peut maintenant recommencer du début et le robot fait le deuxième pas. La maquette marche alors tout droit jusqu'à ce que le programme s'arrête.
- Vous trouverez le programme complet sous Robot marchant1.rpp.

Tout comme pour la maquette roulante de base, vous pouvez faire marcher la maquette vers la gauche, vers la droite ou en arrière en modifiant les sens de rotation des moteurs. Pour compter les pas, vous pouvez utiliser I1 ou I2.

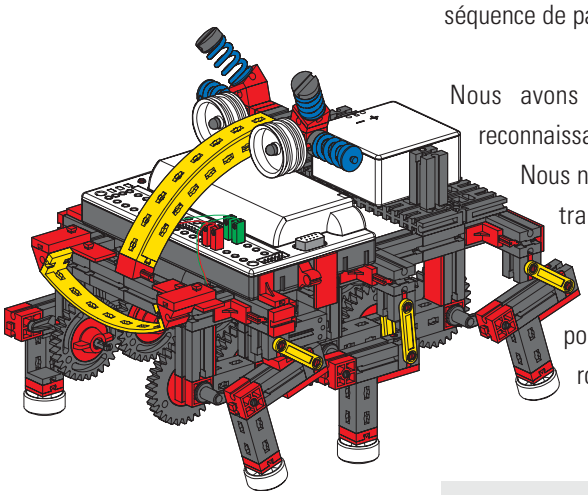
Exercice 2 (Niveau 2) :

- **Programmez votre maquette afin qu'il fasse 10 pas vers l'avant, 3 pas vers la gauche, 3 pas vers la droite et à nouveau 10 pas en arrière.**
- **Créez un sous-programme pour chaque direction.**
- **Utilisez l'élément Boucle de comptage pour compter les pas.**

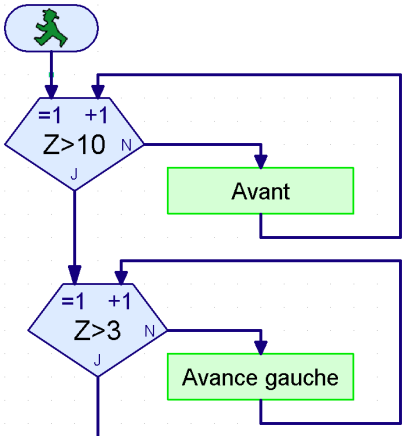
Astuces :

- Copiez simplement le programme Robot marchant1.rpp dans un sous-programme.
- Copiez ce sous-programme autant de fois que nécessaire pour les différents sens de marche. Modifiez les sens de rotation des moteurs dans chaque sous-programme afin que la maquette se déplace dans la direction souhaitée.
- Utilisez l'élément Boucle de comptage pour compter le nombre de pas pour chaque sens de rotation. La maquette fait un pas à chaque déroulement d'un sous-programme. Si le programme parcourt la boucle 10 fois avec le sous-programme, la maquette fait 10 pas.

Vous pouvez ainsi apprendre à votre robot marchant à effectuer une séquence de pas quelconque (Robot marchant2.rpp).



Nous avons déjà traité en détail le sujet de la reconnaissance d'obstacle pour les robots roulants. Nous ne voulons pas y revenir ici. Mais, essaiez de transposer ce comportement au robot marchant. Les capteurs nécessaires sont inclus dans la boîte de construction. vous pouvez, pour la programmation, prendre le robot roulant comme modèle. Bonne chance !

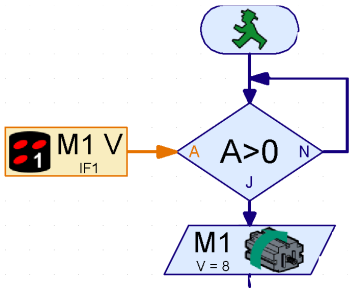


Possibilités d'extension

■ La ROBO Interface offre davantage de fonctionnalités que celle dont les robots mobiles disposaient jusqu'à présent. Des composants supplémentaires qui ne sont pas inclus dans la livraison de la boîte de construction sont disponibles. Cependant, comme ils sont extrêmement intéressants pour les robots, nous voulons vous les présenter brièvement ici.

Emetteur manuel à infrarouge

■ La ROBO Interface contient une diode de réception à infrarouge pour l'émetteur manuel du IR Control Set Art. n° 30344. Dans le logiciel ROBO Pro, vous pouvez ainsi interroger les touches de l'émetteur manuel comme des entrées numériques et ainsi démarrer et arrêter les moteurs.



Nous avons programmé, en guise d'exemple de programmation, une commande de fenêtre pour le robot marchant. Vous pouvez commander la maquette pour qu'elle se déplace vers l'avant, vers l'arrière, vers la gauche et vers la droite à l'aide des 4 flèches ovales. Vous devez juste charger préalablement le programme IR-Robot marchant.rpp sur l'Interface.

Le programme Mobile-Teach-IR.rpp est un autre programme génial en liaison avec la commande à distance. Avec ce programme Teach-In, vous pouvez télécommander l'un des robots roulants, p.e. le robot simple ou la maquette de base. La maquette retient alors la distance parcourue et peut la reproduire à volonté. La distance sauvegardée est toutefois effacée lorsque le programme s'arrête.
Un tel programme est possible grâce à l'élément de programme « Liste » dans ROBO Pro. On peut sauvegarder dans cet éléments de nombreuses valeurs et les appeler (reportez-vous également au manuel ROBO Pro). Le programme lui-même est certes assez complexe, mais son application est très simple :

1. Chargez et démarrez le programme Mobile-Teach-In.rpp dans la mémoire Flash de la ROBO Interface.
2. Appuyez sur la touche de commande à distance **M1** ▸ / ▸▸. Le « processus d'apprentissage » démarre.
3. Avec les flèches ovales, orientez la maquette dans la direction souhaitée.
4. Appuyez sur la touche **M2** ▸ / ▸▸. La distance parcourue est sauvegardée.
5. Appuyez sur la touche **M3** ▸ / ▸▸. La distance sauvegardée est parcourue.

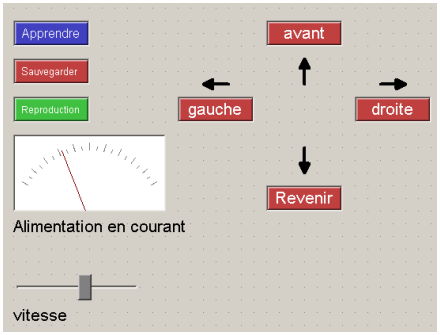
Grâce à cette application, programmer des robots devient un jeu d'enfant. Vous ne devez pas oublier que la distance sauvegardée est effacée dès que le programme est arrêté avec le palpeur Prog de l'Interface.

ROBO RF Data Link

■ L'interface radio ROBO RF Data Link Art. N° 93295 remplace le câble d'Interface entre l'ordinateur et l'Interface par une transmission hertzienne des données. Ceci est très astucieux. Premièrement, vous ne devez pas brancher et débrancher le câble à chaque fois que vous chargez un programme sur l'Interface. Deuxièmement, vous pouvez exécuter le programme sans câble en mode Online et trouver ainsi plus facilement les erreurs qu'en mode de téléchargement. Et, enfin, les robots mobiles peuvent être commandés en mode Online via le panneau de commande de ROBO Pro, de la même façon qu'avec la commande à distance IR Online via l'écran. Contrairement à la commande à distance, on peut également visualiser sur l'écran les données que l'Interface fournit, p.e. les valeurs des variables ou des entrées analogiques, la tension d'alimentation fournie par l'accumulateur, la vitesse des moteurs.

Nous avons, comme exemple, transformé le programme Teach-In et commandé le robot roulant via un panneau de commande. Le programme s'appelle Mobile-Teach-RF.rpp. Vous pouvez, bien entendu, l'essayer avec le câble d'Interface. Mais ceci est assez inconfortable. Le rayon d'action de la maquette est limité, le câble s'emmêle et le robot ne tourne plus correctement. Il est certain que vous sortirez ensuite pour acheter la RF Data Link.

Chargez le programme Mobile-Teach-RF.rpp.
Passez dans la barre de fonctions du programme principal sur le panneau de commande. Démarrez ensuite le programme en mode Online. Vous pouvez maintenant commander et programmer la maquette via les boutons du panneau de commande.



1. Appuyez sur la touche « Apprendre ». Le « processus d'apprentissage » démarre.
2. Orientez la maquette dans la direction souhaitée avec les flèches.
3. Appuyez sur la touche « Sauvegarder ». La distance parcourue est sauvegardée.
4. Appuyez sur la touche « Reproduction ». La distance sauvegardée est parcourue.

Le trajet sauvegardé est également effacé lorsque le programme s'achève.

Vous en apprendrez davantage sur la création des panneaux de commande dans le manuel ROBO Pro.

ROBO I/O-Extension

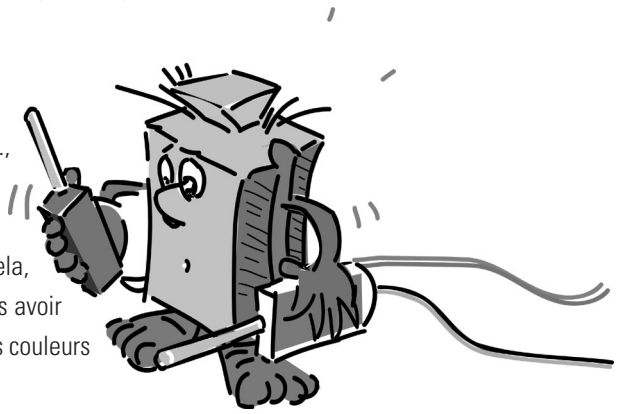
■ Si vous construisez une maquette avec tellement de capteurs et de moteurs de manière que les sorties et les entrées de la ROBO Interface ne suffisent plus, vous pouvez connecter une ROBO I/O-Extension Art. N° 93294 à l'Interface. Vous disposez ainsi de 8 entrées numériques, 4 sorties Moteurs et d'une entrée de résistance analogique supplémentaires. Vous pouvez raccorder un second module sur cette extension I/O et, sur ce second module, vous pouvez encore raccorder un troisième module qui sont tous commandés par une ROBO Interface. Vous disposerez alors d'un total de 16 sorties Moteurs, 32 entrées numériques, 5 entrées de résistance analogiques, 2 entrées de tension analogiques et 2 entrées pour les capteurs de distance.

Si vous n'en avez toujours pas assez, vous pouvez aussi commander sur l'ordinateur plusieurs Interfaces en mode Online, p.e. une interface sérielle COM, une interface USB ou 2 interfaces sur le bus USB et chacun avec jusqu'à 3 ROBO I/O-Extensions ! Cela peut facilement faire tourner la tête. Le chapitre 6 du manuel ROBO Pro explique également comment tout cela fonctionne.

Recherche des pannes

■ Faire des expériences est amusant. Et il est encore plus amusant que tout fonctionne, ce qui est la plupart du temps le cas, mais malheureusement pas toujours.
Ce n'est que lorsqu'une maquette ne fonctionne pas que l'on découvre si l'on a correctement compris le mécanisme et que l'on peut tout de suite trouver la panne.
Pour les pannes mécaniques, on arrive encore à voir (montage incorrect) ou à sentir (dureté) quelque chose. Mais quand des problèmes électriques s'y ajoutent, cela devient plus difficile.
Pour rechercher les pannes, les professionnels utilisent toute une série d'instruments de mesure divers, comme p.e. un tensiomètre ou un oscillographe. Tout le monde ne dispose pas de tels appareils. Nous voulons en conséquence essayer de détecter et de réparer une panne par des moyens simples.

Câblage



Avant de commencer nos expériences, vous devez tout d'abord achever quelques composants de la boîte de construction Fischertechnik. Il faut, p.e., raccorder les connecteurs livrés aux diverses sections de câble.
Le câble doit tout d'abord être coupé. Mesurez les longueurs indiquées et découpez-les sections. Chaque câble doit être mesuré après la finition. Pour cela, vous avez besoin de l'accumulateur et de la lampe. Si la lampe s'allume après avoir été reliée à l'accumulateur, le câble convient. Vérifiez aussi si l'affectation des couleurs est correcte, pin rouge avec câble rouge, pin vert avec câble vert.

Test de l'Interface

Si le programme (même celui fourni) ne fonctionne pas avec votre maquette, démarrez le test de l'Interface. Ce programme d'aide vous permet de tester séparément les entées et les sorties. Les capteurs fonctionnent-ils ? Les moteurs tournent-ils dans le sens correct ? Les moteurs de tous vos robots mobiles sont raccordés de manière à ce que, dans le sens de rotation vers la gauche, la roue ou la patte se déplace vers l'avant. Si là aussi tout est correct, recherchez la cause mécanique.

Mauvais contacts

Les mauvais contacts sont des pannes fâcheuses. Les pins de raccordement peuvent, d'une part, être mobiles dans les douilles. Si tel est le cas, les ressorts de contact des connecteurs doivent être un peu élargis à l'aide d'un petit tournevis. Attention, un trop grand élargissement peut provoquer la rupture des contacts ou rendre difficile l'introduction dans la prise.
Les jonctions par serrage relâchées au point de fixation des connecteurs peuvent être une autre cause de mauvais contacts. Resserrez-les avec précaution ! Il faut vérifier en même temps si aucun mince fil de cuivre n'est cassé.

Courts-circuits

Il peut aussi arriver que l'on provoque un court-circuit si un câble est mal posé. Rien ne fonctionne alors plus comme cela devrait le faire. L'accupack contient un fusible qui coupe le courant en cas de surintensité de courant ou de température trop élevée. Les sorties de l'Interface sont également fermées en cas de surchauffe.
Un court-circuit peut aussi se produire si la petite vis de la prise électrique avec laquelle le câble est fixé n'est pas serrée suffisamment sur la prise électrique. Elle dépasse alors du bord de la prise. Si l'on branche deux connecteurs dans deux douilles situées côte à côte sur l'Interface et que les petits vis sont

en contact, un court-circuit se produit. C'est pourquoi les petites vis doivent toujours être bien serrées et les connecteurs branchés de manière à ce que les petites vis ne soient pas en contact.

Alimentation électrique

En cas de raté inexpliqué pendant le fonctionnement, la cause peut pratiquement toujours être attribuée à un accumulateur vide. La tension chute brièvement lors de la mise sous tension d'une charge (moteur allumé), ce qui déclenche une remise à zéro du processeur de l'Interface. La ROBO Interface indique, par l'allumage de la DEL rouge, que la tension de l'alimentation électrique est trop basse. L'accumulateur doit alors être rechargé.

Erreur de programmation

Si des pannes que l'ont ne peut pas expliquer surviennent pour les programmes décrits, il convient, par mesure de précaution, d'exécuter un programme aussi identique que possible aux programmes fournis afin de pouvoir exclure des défauts électriques ou mécaniques. En mode Online, il est possible de suivre le déroulement du programme sur l'écran. Si le programme s'arrête à un certain endroit, la cause peut alors être recherchée ici, p.e. sélection incorrecte de l'entrée ou du moteur, valeur incorrecte interrogée lors d'un branchement ou inversion des raccordements O/N.

Si tout ceci ne sert à rien, il ne vous reste qu'à prendre contact avec le service après-vente Fischertechnik (email : info@fischertechnik.de).

Vous pouvez aussi visiter notre sité Internet, sous www.fischertechnik.de. Vous y trouverez un forum, un chat, un marché, une galerie et vous pourrez gratuitement devenir membre du fan club Fischertechnik.

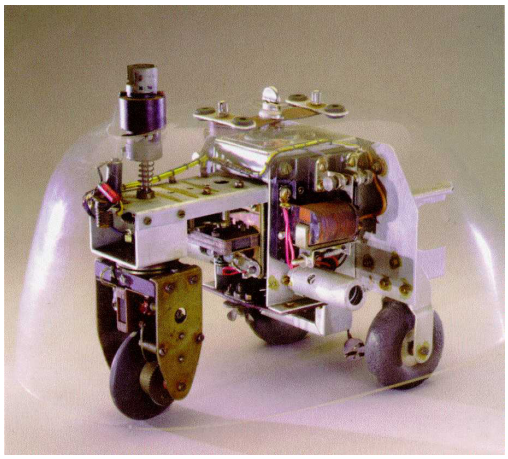
Nous vous souhaitons de vous amuser longtemps avec le ROBO Mobile Set avec de nombreux effet « ha ! ha ».

Inhoudsopgave

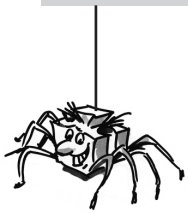
Waarvoor hebben we robots nodig?

■ Het begrip 'robot' werd voor het eerst in 1923 gebruikt in de roman "Golem" van Carel Capek. Deze kunstmatig gecreëerde figuur moest de juiste vaardigheden hebben om werkzaamheden over te nemen van de mens.
In de jaren 30 en 40 van de 20e eeuw werd de robot veel meer als een automaat gezien. Diverse pogingen om hem van menselijke eigenschappen te voorzien, zoals bijv. een hoofd met knipperlichten als ogen, werken tegenwoordig alleen nog maar op de lachspieren. Van mobiliteit of zelfs intelligentie is bij deze machines weinig te merken. Omdat het besturingsprincipe van grote invloed is op de robottechniek werd de bouw van robots een stuk realistischer toen de eerste elektronische schakelingen beschikbaar werden. Het streven robots van "intelligentie" te voorzien is vandaag de dag nog altijd een onderwerp van onderzoek bij tal van bedrijven, instituten en universiteiten.

■ De eerste oplossingen dacht men te vinden met de opkomst van de zogenaamde cybernetica. De term "cybernetica" is afgeleid van het Griekse woord 'kybernetes'. De kybernetes was de navigator op een Griekse roeiboot. Hij moest de locatie van het schip bepalen en de noodzakelijke koers tot het doel berekenen.
Cybernetica moest de robot "intelligent" maken. Hoe moet je je een dergelijk intelligent gedrag eigenlijk voorstellen?
We zullen proberen om dit met een vergelijking te verduidelijken. Iedereen heeft wel eens gezien hoe een motvlinder zich in het schijnsel van een lamp gedraagt. Hij herkent de lichtbron en vliegt er naartoe om dan vlak voordat hij tegen de lamp zou vliegen uit te wijken. Het is duidelijk dat de mot voor dit gedrag de lichtbron moet herkennen, een weg daar naartoe moet bepalen en er naartoe moet vliegen. Deze vaardigheden zijn gebaseerd op instinctieve, intelligente gedragspatronen van het insect.
Laten we nu eens proberen deze vaardigheden over te brengen op een technisch systeem. We moeten de lichtbron herkennen (optische sensoren), een beweging uitvoeren (motoren aansturen) en we moeten een zinvolle samenhang realiseren tussen herkenning en beweging (het programma).

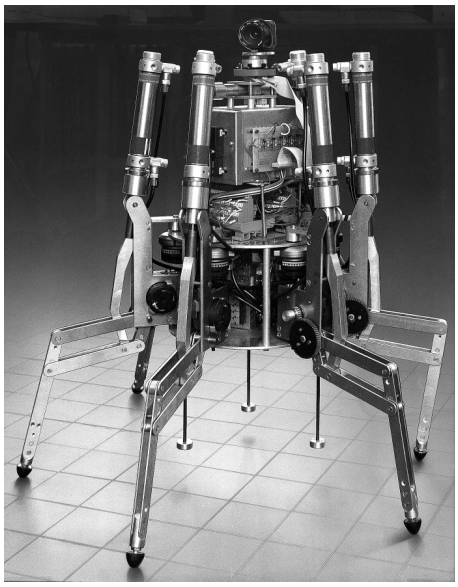


■ Het hierboven beschreven principe heeft de Engelsman Walter Grey in de jaren 50 van de twintigste eeuw in praktijk gebracht. Met behulp van eenvoudige sensoren, motoren en elektronische schakelingen creëerde hij diverse "cybernetische" dieren, die elk voor zich een zeer specifiek gedrag, zoals bijv. dat van een mot, vertoonden. Op de foto is een kopie te zien van de "cybernetische" schildpad die te zien is in het Smithsonian Museum in Washington.
Op basis van het bovenstaande gaan wij voor onze robots ook de gewenste gedragspatronen opstellen en proberen we deze in de vorm van programma's aan de robots duidelijk te maken.



■ Maar waarvoor hebben we eigenlijk mobiele robots nodig? Laten we het gedrag van onze denkbeeldige mot eens op technische apparaten toepassen. Een eenvoudig voorbeeld daarvan is licht zoeken. We modificeren de lichtbron door een lichte streep, de richtlijn, op de vloer aan te brengen en de sensoren naar beneden te laten wijzen. Met behulp van dergelijke richtlijnen kan een mobiele robot zich bijvoorbeeld in een magazijn oriënteren. Via aanvullende informatie, bijv. in de vorm van een barcode op bepaalde posities op de lijn, krijgt de robot opdrachten om andere acties uit te voeren, zoals bijv. een pallet opnemen of afzetten.
Dergelijke robotsystemen bestaan al in de praktijk. In grote ziekenhuizen doen zich vaak lange transportwegen voor verbruiksmaterialen zoals beddengoed voor. Het transport van dit materiaal door het verplegend personeel is erg kostbaar en gaat deels gepaard met zware lichamelijke arbeid. Bovendien gaan dergelijke activiteiten ten koste van de tijd die beschikbaar is voor het verzorgen van de patiënten.

■ Sinds een aantal jaren houden wetenschappers zich bezig met een andere, in de natuur wijd verbreide bewegingsvorm, namelijk het lopen. Er worden robots ontwikkeld, die in staat zijn zich op poten voort te bewegen. Een voorbeeld van een lopende robot op zes poten is de elektropneumatische "Achille" die ontwikkeld is aan de Koninklijke Militaire Academie in Brussel. Deze robot is uitgerust met zes poten en een camera aan de bovenkant en moet mechanisch reageren op verhoogde of verlaagde hindernissen (objecten of gaten).
Dergelijke lopende machines kunnen overal worden ingezet waar wiel- of kettingvoertuigen vrijwel onbruikbaar zijn, dus bijv. op zeer ongelijk of drassig terrein, om over hindernissen te klimmen, trappen te beklimmen, greppels te passeren of op moeilijk toegankelijke of gevaarlijke locaties zoals kerncentrales, in de mijnbouw of bij reddingsacties.



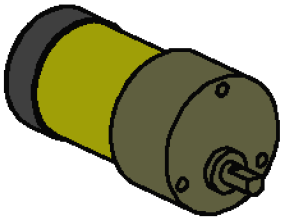
Hieruit valt af te leiden dat mobiele robots een zeer belangrijke plaats kunnen innemen in de moderne maatschappij.

Robots van fischertechnik

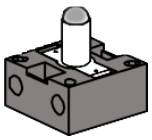
■ Hoe kun je een robot maken van je fischertechnik-dozen? Voor een robot hebben we naast de sensoren (bijv. contactschakelaars) en actuatoren (bijv. motoren) veel mechanische onderdelen nodig om een model te kunnen bouwen. Hiervoor is de fischertechnik-doos ROBO Mobile Set de ideale basis. Deze doos bevat de volgende sensoren en actuatoren:

Actuatoren

Power Motor:



Twee van deze krachtige gelijkstroommotoren (9VDC/2,4W) met een ingebouwd drijfwerk en een tandwielreductie van 50:1 drijven de mobiele robotmodellen aan (d.w.z. dat de motor 50 keer ronddraait terwijl de as die uit de motor steekt in diezelfde tijd slechts eenmaal ronddraait).

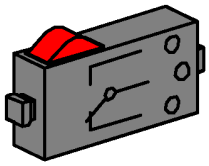


Lenslamp:

Dit gloeilampje (9VDC/150mA) dient voor het afgeven van eenvoudige lichtsignalen.
In de glazen ballon van de lamp is een lens geïntegreerd, die het licht bundelt dat wordt uitgestraald. Door de lichtstraal op een helderheidssensor (zoals een fototransistor, zie hieronder) te richten, kun je een sensor bouwen die licht en donker onderscheidt. De lamp kan ook worden gebruikt voor het aangeven van bepaalde toestanden of als knipperlicht voor waarschuwingsmeldingen. In de bouwdoos wordt de lamp samen met 2 foto transistors als speciale sensor gebruikt voor lijnherkenning.

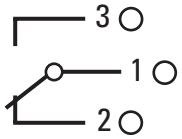
Sensoren

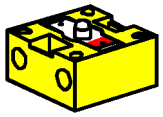
■ Bij sensoren maakt men onderscheid tussen digitale en analoge sensoren.
De contactschakelaar is een voorbeeld van een digitale sensor. Digitale waarden kunnen slechts 2 verschillende toestanden aannemen. Deze toestanden worden met 0 resp. 1 aangegeven. Voor de contactschakelaar betekent "0" dat er geen stroom tussen de aansluitingen loopt en betekent "1" dat er wel stroom is.



De **contactschakelaar** van fischertechnik, die in het programma ROBO PRO 'toets' wordt genoemd, is uitgevoerd als een wisselschakelaar. Daarom heeft hij 3 aansluitingen. Als de rode knop wordt ingedrukt, wordt er mechanisch een schakelaar geactiveerd, die aansluiting 1 en 3 met elkaar verbindt. Tegelijkertijd wordt het contact onderbroken tussen aansluiting 1 en 2 die in rusttoestand met elkaar verbonden waren. Op deze manier kunnen beide mogelijke uitgangsposities worden afgevraagd:

in rusttoestand gesloten (aansluiting 1 en 2 bezet)
in rusttoestand open (aansluiting 1 en 3 bezet)





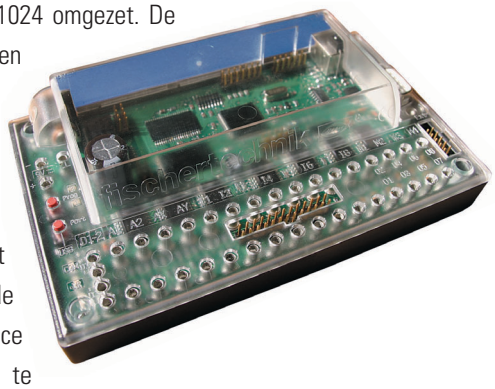
De **fototransistor** kan als digitale en als analoge sensor worden gebruikt. In het eerste geval dient hij voor het herkennen van duidelijke overgangen tussen licht en donker, zoals bijv. een gemarkeerde lijn. Ook is het mogelijk verschillende lichtsterktes te onderscheiden; de fototransistor werkt dan als een analoge sensor. Analoge waarden kunnen alle mogelijke waarden tussen hun laagste en hoogste waarde aannemen. Om deze waarden door de computer te kunnen laten verwerken, moeten ze worden omgezet in desbetreffende getalswaarden.

Bij de fototransistor gaat het overigens om een zogeheten halfgeleidercomponent, met elektrische eigenschappen die afhankelijk zijn van de lichtsterkte. Iedereen kent wel zonnecellen, waarmee stroom

opgewekt wordt uit zonlicht. De fototransistor kunnen we zien als een combinatie van een minizonnecel en een transistor. Lichtimpulsen (fotonen) die op de fototransistor vallen genereren een zeer geringe stroom, die vervolgens door de transistor wordt versterkt.

Let op:
Let bij het aansluiten van de fototransistor op de juiste aansluiting van de polen: rode markering = plus. Toelaatbare spanning: 30V max.

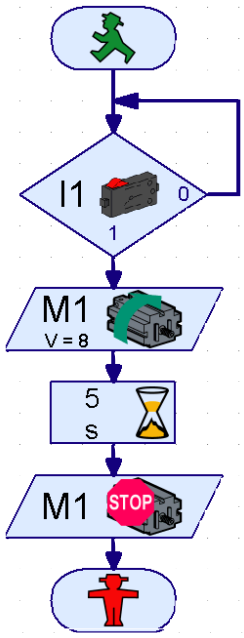
ROBO Interface



■ Verschillende sensoren en actuatoren kunnen worden aangesloten op de ROBO Interface en ermee worden geanalyseerd. Naast 8 digitale ingangen biedt de ROBO Interface de beschikking over een aantal analoge ingangen. Zo wordt bijvoorbeeld een op de ingangen AX en AY uitgeoefende weerstandswaarde tussen 0 en 5,5 kΩ in een getalswaarde van tussen 0 en 1024 omgezet. De meetwaarden van een helderheidssensor, zoals bijv. een fototransistor, worden daarmee geregistreerd en zijn beschikbaar voor verder gebruik. Op de analoge ingangen A1 en A2 kunnen spanningen tussen 0 en 10VDC worden gemeten.

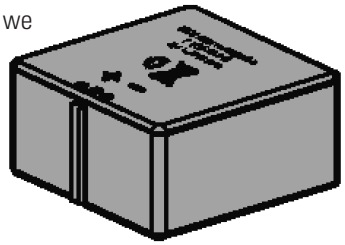
De belangrijkste functie van de interface is de logische koppeling van de ingangswaarden. Daarvoor heeft de interface een programma nodig. Dit programma beslist hoe uit de ingangsgegevens (de sensorsignalen) de bijpassende uitgangsgegevens zoals motorstuursignalen etc. ontstaan. Met de ROBO Interface beschikken we over genoeg rekenvermogen om ook moeilijke programma's te ontwerpen.

Software ROBO Pro



■ Om zo effectief mogelijk de voor de interface benodigde programma's te kunnen maken, is er een grafisch programmeeroppervlak gemaakt. Het begrip "programmeeroppervlak" staat hier voor een softwareoplossing die het ons mogelijk maakt op zeer gemakkelijke wijze onze programma's te maken. Dit gebeurt met behulp van grafische symbolen. De computer van de ROBO Interface kan eigenlijk alleen maar commando's uit zijn zogenaamde 'machinecommandopakket' uitvoeren. In principe zijn dit eenvoudige besturingsstructuren waarvan de toepassing buitengewoon moeilijk is voor beginners. De pc-software ROBO Pro stelt daarvoor de grafische elementen ter beschikking, die vervolgens worden omgezet in een taal die de interface kan uitvoeren.

Voeding



■ Het enige dat we nog extra nodig hebben in aanvulling op de doos ROBO Mobile Set, is de Accu Set, art.nr. 34969. Deze bevat de accu die we gebruiken als de mobiele voeding voor onze robots en een speciale oplader voor de accu.
Het is het beste om de accu meteen met de oplader op te laden, zodat hij vol is als we met de experimenten willen beginnen.

Procedures voor het experimenteren

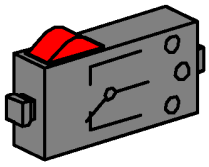
■ We zullen stap voor stap de fascinerende wereld van mobiele robots binnengaan. We beginnen met een eenvoudige testopzet voor het testen van de basisfuncties van de interface en de sensoren. Vervolgens bouwen we eenvoudige modellen, die bepaalde opdrachten krijgen. Daarna proberen we het met steeds complexere systemen.
Als het maken van eigen programma's op een bepaald moment te gecompliceerd wordt en te lang duurt, kun je ook de meegeleverde voorbeeldprogramma's op de interface laden en de robots daarmee aansturen. Aan het einde van dit boekje vind je een hoofdstuk over het oplossen van fouten om je te helpen bij eventuele storingen.

Een zeer belangrijk punt is de zorgvuldigheid bij het opbouwen en in gebruik nemen van de robots. Bij het aansluiten van de elektrische componenten moet je de instructies nauw opvolgen en liefst twee of zelfs drie keer controleren of alles klopt. Bij de mechanische constructies, ook bij eigen creaties, zijn onbelemmerde beweging en weinig speling op de tandwielen en bevestigingen van belang. Het wordt aan je creativiteit overgelaten om eigen programma's te schrijven en daarmee een nieuwe "gedraging" te definiëren. Dit wordt slechts beperkt door de geheugen- en rekencapaciteit van de hardware. De volgende voorbeelden dienen daarbij als interessante vingeroefeningen.

De eerste stappen

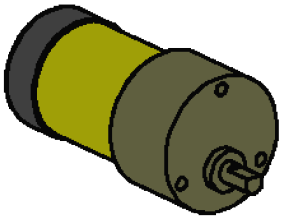
■ Na de theoretische bespiegelingen gaan we nu onze eigen experimenten uitvoeren. Waarschijnlijk sta je te popelen om direct te starten, en dan nog het liefst meteen met de grote lopende robot. Dat is natuurlijk mogelijk en als de bouwinstructies zorgvuldig worden opgevolgd zal het ook wel lukken om het model op te bouwen.
Maar wat te doen als het niet werkt? In dergelijke gevallen moet je systematisch de oorzaak van de fout zoeken. Maar voordat we ons daarmee gaan bezighouden, controleren we eerst nog een keer de samenwerking tussen de computer en de interface.

In hoofdstuk 1 en 2 van het handboek van de software ROBO Pro wordt het installeren van de besturingssoftware op de pc en het aansluiten van de interface beschreven. Met behulp van de Interfacetest kun je de verschillende sensoren en/of actuatoren testen



Contactschakelaar

Contactschakelaar
Je kunt nu bijv. een contactschakelaar aansluiten op de digitale ingang I1 en kijken hoe de toestand van de ingang verandert als de contactschakelaar geactiveerd wordt.

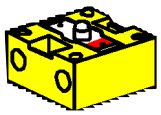


Powermotor

Power Motor
Je kunt de uitgangen controleren door een motor te verbinden met een motoruitgang, bijv. M1. Met de linker muisknop kun je de motor de andere kant op laten draaien en met de schuifregelaar kun je de snelheid aanpassen.

Fototransistor

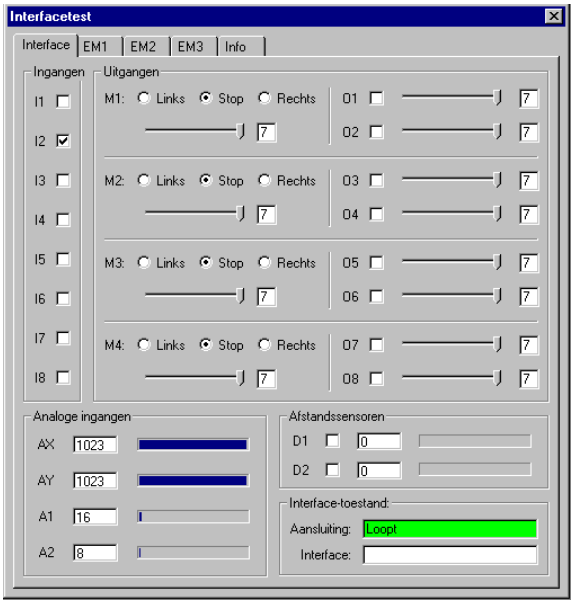
Als je ook de analoge ingang AX wil testen, kun je een fototransistor als analoge sensor toepassen. Waar bij de motor resp. de contactschakelaar de polariteit van de aansluitingen geen rol speelt (de motor draait in het ongunstigste geval verkeerd om), is een correcte aansluiting van de polen bij de fototransistor vereist om hem correct te laten werken.



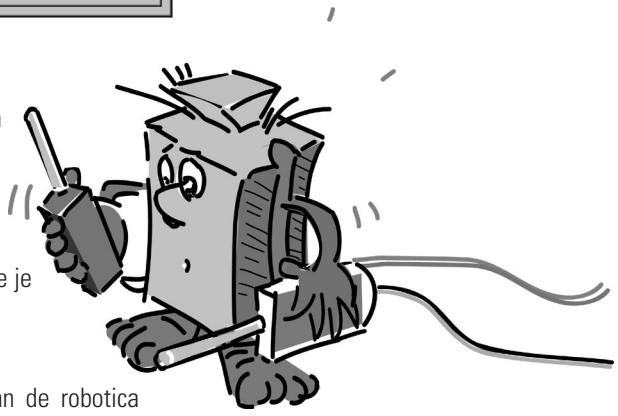
Fototransistor

Verbind het met rood gemarkeerde contact van de transistor met een rode aansluitstekker en verbind het andere contact met een groene stekker. De tweede groene stekker komt in de bus van ingang AX die zich dichter bij de rand van de interface bevindt; de tweede rode stekker past in de verder naar binnen gelegen bus van AX. (Let op: als de fototransistor wordt aangesloten op een digitale ingang I1-I8, moet de rode stekker in de bus het dichtst bij de rand van de behuizing worden gestoken).

Nu kun je met behulp van een zaklamp de belichtingssterkte van de fototransistor variëren en daarmee de uitslag van de blauwe balk van AX veranderen. Als de wijzer niet tot voorbij het maximum uitslaat, dan moet je de aansluitingen van de fototransistor nog eens goed bekijken. Als de wijzer daartegen ook bij een uitgeschakelde zaklamp op nul staat, dan kan het zijn dat de belichting in het vertrek, het omgevingslicht, te sterk is. De uitslag verandert dan als je de fototransistor afdekt.

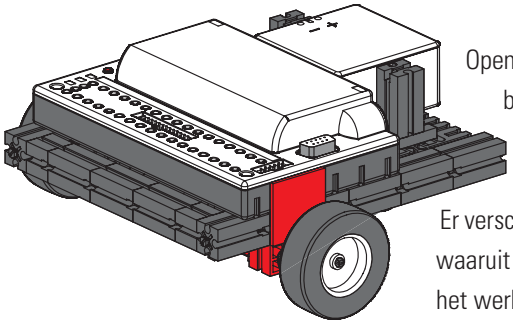


Om nog even kort terug te komen op de kleurcodering van de stekker: Let er goed op dat je tijdens het monteren altijd een rode stekker op de rode geleider aansluit en een groene stekker op de groene geleider. Als de juiste polariteit van belang is voor de opbouw van een circuit, neem je altijd een rode geleider voor de pluspool en een groene geleider voor de minpool. Dit lijkt misschien een beetje hypernauwkeurig, maar het systematisch fouten kunnen zoeken wordt aanzienlijk vergemakkelijkt als je je strikt aan de juiste kleurcoderingen houdt.



Met een eenvoudig programma gaan we onze eerste stappen op het gebied van de robotica afsluiten. In hoofdstuk 3 van het ROBO Pro handboek wordt het programma "Garagedeurbesturing" uitgelegd. Dit heeft weliswaar niets met mobiele robots te maken, maar het is prima geschikt om vertrouwd te raken met de ROBO Pro software. Om dit programma te kunnen uitvoeren is het voldoende de motor en drie contactschakelaars uit de bouwdoos ROBO Mobile Set op de interface aan te sluiten. De rest wordt uitvoerig beschreven in het softwarehandboek.

De eerste eenvoudige robot



■ Na de Interfacetest en de Garagedeurbesturing gaan we nu eindelijk de eerste robot in gebruik nemen. We bouwen het model "Eenvoudige robot" met de twee aandrijfmotoren op zoals beschreven in de bouwinstructies. Dat gaat erg eenvoudig en snel, omdat dit model met opzet alleen voorzien is van de basiselementen die een robot nodig heeft om te kunnen rijden. De motoren sluiten we aan op uitgang M1 en M2.

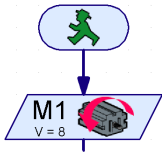
Open de software ROBO Pro en maak een nieuw programma aan (BESTAND – NIEUW]). ROBO Pro biedt verschillende moeilijkheidsniveaus om in te werken. Deze kunnen in ROBO Pro worden ingesteld in het menuonderdeel NIVEAU. Voorlopig is Niveau 1 goed.

Er verschijnt een leeg werkblad en aan de linkerrand van het beeldscherm verschijnt het elementvenster waaruit je de verschillende programma-elementen kunt selecteren met de linker muisknop om deze op het werkblad te plaatsen. Je kunt de eigenschappen wijzigen met de rechter muisknop.

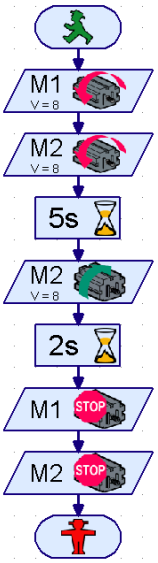
Opdracht 1 (Niveau 1):
De "Eenvoudige robot" moet 5 seconden lang recht vooruit rijden, vervolgens 2 seconden lang een rondje draaien en daarna blijven stilstaan.

Tips:

De eerste robot programmeren we stap voor stap samen:



- We beginnen met het groene mannetje. Dit mannetje symboliseert de start van het programma.
- Vervolgens halen we het motorsymbool uit het elementvenster. Plaats dit symbool onder het startelement, zodat de verbindingslijn automatisch getekend wordt. In het venster met eigenschappen stellen we de motoruitgang "M1" en de draairichting "links" in en we bevestigen dit met OK.



- Onder dit symbool plaatsen we op dezelfde manier nog een motorsymbool. Daarmee schakelen we motor 2 in.
- Om gedurende een bepaalde tijd te wachten, maken we gebruik van het element Wachttijd, dat we onder het tweede motorsymbool plaatsen. We stellen de tijd hierbij in op 5 seconden.
- Vervolgens laten we de motor M2 in de andere richting draaien (rechtsom), waarna we 2 seconden wachten om tenslotte beide motoren uit te schakelen. Ons programma eindigt met het eindsymbool, het rode mannetje. Op de afbeelding is de complete programmastructuur te zien.

Als je niet zeker weet of alles klopt, kun je het programma vergelijken met het meegeleverde voorbeeldprogramma. Sla het eigen programma dan eerst op en laad het bestand Eenvoudige robot 1.rpp uit de voorbeeldmap van ROBO Pro (Standaardinstelling C:\Programmas Files\ROBO Pro\ Voorbeeldprogrammas\ROBO Mobile Set).

Als alles in orde is, dan kan het programma via de downloadfunctie in de interface worden geladen. Na het indrukken van de knop Download verschijnt er een dialoogvenster. Daar stellen we in dat het programma in FLASH-geheugen 1 moet worden geladen, waarna het direct na het downloaden moet worden gestart.

Direct na het downloaden rijdt ons model weg, hij draait even rond en blijft staan. Om het programma opnieuw te starten, kun je op de interface de Prog-knop even indrukken. De LED Prog1 knippert dan

weer, zolang het programma loopt. Daarna gaat hij constant branden. Het programma in het FLASH-geheugen blijft trouwens ook bewaard als de stroomvoorziening naar de interface onderbroken wordt. Probeer dit uit door de stekker van de accu uit te trekken. Steek de stekker weer in en selecteer het opgeslagen programma door zolang op de Prog-knop te drukken tot de LED Prog1 gaat branden. Druk nogmaals op de knop om het programma te starten.

Onze robot kan nog niet echt veel, maar daar gaan we wat aan doen.

Opdracht 2 (Niveau 1):

Om te zorgen dat onze robot niet al na 7 seconden blijft stilstaan, gaan we hem nu dansen leren.

- **Laat hem over verschillende afstanden recht vooruit, linksom, rechtsom en achteruit rijden en op verschillende snelheden.**
- **Deze procedure moet door blijven gaan totdat het programma beëindigd wordt met de Prog-knop op de interface.**

Tips:

- Pool eenvoudigweg de motoren steeds weer om om de robot in de gewenste richtingen te laten rijden.
- De snelheid van de motoren kun je in het venster met eigenschappen voor elk motorsymbool instellen tussen 1 en 8. Als M1 en M2 met verschillende snelheid in dezelfde richting draaien, dan maakt de robot een bocht.
- Om te zorgen dat het programma voortdurend herhaald wordt, moet je een verbindingslijn trekken van de uitgang van het laatste programma-element naar de lijn die in het eerste element naar binnen gaat.
- Een kant-en-klaar voorbeeld vind je onder Eenvoudige robot 2.rpp.

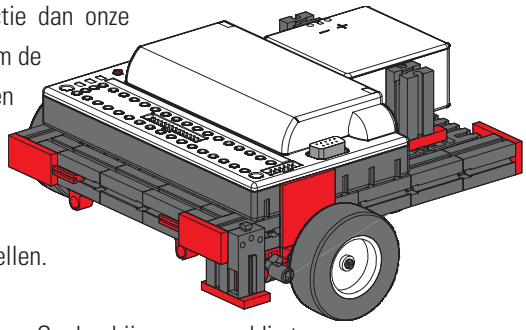
Gefeliciteerd, je hebt nu je eerste eigen robot gebouwd en hem zelf geprogrammeerd. Hij is echter nog niet bijzonder intelligent, want hij herkent nog geen hindernissen en valt nog van de tafel als je niet oppast. Maar in de loop van de verdere experimenten verandert dat nog.

Intelligente rijdende robot

■ Om robots hun omgeving te laten herkennen, hebben ze sensoren nodig. De volgende modelsuggesties zijn voorbeelden van varianten van mobiele robots waarmee het gebruik van verschillende sensoren kan worden uitgeprobeerd. Daarbij gaat het erom zowel interne toestanden van de robot, zoals de trajectmeting door impulswielen, als externe signalen, zoals bij het spoor- of licht zoeken, aan elkaar te koppelen. Bij elk model worden daarvoor bepaalde opdrachten gegeven. Dit zijn interessante oefeningen waardoor je vertrouwd raakt met de materie. De programma's voor de verschillende opdrachten bevinden zich in de directory van ROBO Pro onder \Voorbeeldprogrammas\ROBO Mobile Set\. Maar je mag natuurlijk ook je eigen opdrachten bedenken voor de modellen. Als je de volgende voorbeelden uitgeprobeerd hebt, heb je vast en zeker nog een heleboel andere ideeën.

Basismodel

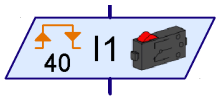
■ Het basismodel is stabieler en robuuster van constructie dan onze eerste "Eenvoudige robot". Hij heeft bovendien 2 sensoren om de weg te bepalen. Deze bestaan uit een contactschakelaar en een impulswiel. Het impulswiel is met de motoras verbonden en activeert bij elke draaicyclus van de motor vier maal een contactschakelaar.
Dit model dient als basis voor de verdere mobiele robotmodellen.



Bouw het basismodel zoals in de bouwinstructies is beschreven. Ga daarbij zeer zorgvuldig te werk. Als mechanisch alles klaar is, controleer je of de motoren soepel draaien door elke motor even zonder de interface rechtstreeks op de accu aan te sluiten.

Opdracht 1 (Niveau 1):

- **Programmeer de interface zo dat het model 40 pulsen recht vooruit rijdt. Gebruik voor het meten van de impulsen de telcontactschakelaar op ingang I1.**
- **Meet de afstand die het model aflegt en bereken welke afstand er per impuls wordt afgelegd.**
- **Herhaal deze proef 3 maal en leg in de tabel vast, hoe sterk de waardes van elkaar afwijken.**



Tips:

- Schakel eerst beide motoren in (draairichting linksom).
- Om de impulsen op I1 te tellen maak je gebruik van het programma-element **Impulsteller**.
- Tel beide impulsflanken (0-1 bij het indrukken, 1-0 bij het loslaten van de contactschakelaar). Dit kun je in het venster met eigenschappen instellen onder **Impulstype**. Daarmee verhoog je de nauwkeurigheid van de wegbepaling.
- Schakel de motoren daarna weer uit en beëindig het programma.
- Je vindt het kant-en-klare programma onder Basismodel 1.rpp.

Resultaat:

	Aantal pulsen	Afgelegde afstand	Afstand/impuls
Poging 1	**40**		
Poging 2	**40**		
Poging 3	**40**		

Grofweg kun je stellen dat het model per impuls ongeveer een afstand van een centimeter aflegt.

Intussen weet je ook welke draairichting je voor de afzonderlijke motoren moet instellen om het model in een bepaalde richting te laten rijden. Leg deze resultaten vast in de volgende tabel, zodat je er niet meer bij elke richtingsverandering over hoeft na te denken. Als je de bekabeling op het model exact uitvoert zoals beschreven in de bouwinstructie, betekent de draairichting linksom bij elke motor dat het wiel vooruit draait. Zo zijn de motoren in alle voorbeeldprogramma's geprogrammeerd.

Vul de tabel in:

Rijrichting model	Draairichting M1	Draairichting M2
Vooruit	**Linksom**	**Linksom**
Achteruit		
Links		
Rechts		
Stop		

Om niet bij elke richtingsverandering twee motorsymbolen op het beeldscherm te hoeven plaatsen, kun je voor elke rijrichting een subprogramma maken dat deze taak op zich neemt. Dit vereenvoudigt het programmeren enorm. Het aanmaken van subprogramma's wordt beschreven in hoofdstuk 4 van het handboek bij de software ROBO Pro. Zodra je dit hoofdstuk hebt gelezen, kun je je aan de volgende opdracht wagen. Schakel nu in ROBO Pro over op **Niveau 2**.

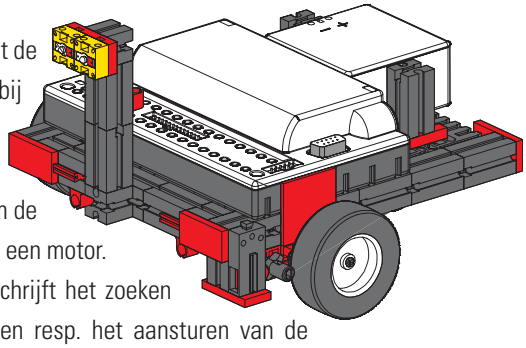
Opdracht 2 (Niveau 2):

- **Maak een subprogramma aan voor elke rijrichting.**
- **Programmeer het model zo, dat het een vierkant met een meter lange zijden rijdt.**
- **Hoe groot is de herhalingsnauwkeurigheid?**

Tips:

- Maak eerst een subprogramma **Vooruit**. De andere subprogramma's kun je dan genereren door dit subprogramma te kopiëren. Daarin moet je dan nog wel de draairichtingen van de motoren aanpassen.
- Gebruik een lagere snelheid om naar links en rechts te draaien. Dat verhoogt de nauwkeurigheid.
- Om de impulsen te tellen, maak je weer gebruik van het element **Impulsteller** en de contactschakelaar bij ingang I1.
- Laad het programma eerst in het RAM om het uit te proberen en laat het daarin staan tot je hebt uitgezocht hoeveel impulsen je nodig hebt om een draaiing van 90° uit te voeren. Ten eerste gaat het laden in het RAM sneller dan het laden in het FLASH-geheugen en ten tweede heeft het FLASH-geheugen slechts een "beperkte" levensduur van ca. 100.000 downloads.
- Het kant-en-klare programma heet Basismodel 2.rpp.

De Lichtzoeker



■ Je hebt het basismodel nu uitgebreid onderzocht. Nu moet de robot leren om op omgevingssignalen te reageren. Evenals bij de mot in onze vergelijking in het eerste hoofdstuk moet hij een lichtbron herkennen en volgen. De bouwdoos bevat 2 fototransistors, die we als lichtdetector gaan toepassen. Om de lichtbron te kunnen volgen wordt elke sensor gekoppeld aan een motor. Het programma bestaat uit twee delen. Het ene deel beschrijft het zoeken naar een lichtbron en in het andere deel wordt het volgen resp. het aansturen van de lichtbron gerealiseerd. Daarvoor worden weer subprogramma's gebruikt. Na het inschakelen wordt het subprogramma 'Licht zoeken' geactiveerd. Dit subprogramma wordt pas afgesloten nadat er een lichtbron is gevonden. Het hoofdprogramma probeert de robot op basis van de lichtbron te sturen. Telkens als de richting van de robot sterk van de ideale lijn afwijkt, wordt een van de twee sensoren niet meer beschenen door de lichtbron. Als gevolg daarvan corrigeert de robot zijn rijrichting, zodat beide sensoren de lichtbron weer kunnen herkennen.

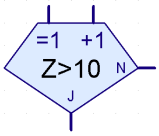
Bouw eerst het model Lichtzoeker zoals in de bouwinstructies is beschreven.

Opdracht 1 (Niveau 2):

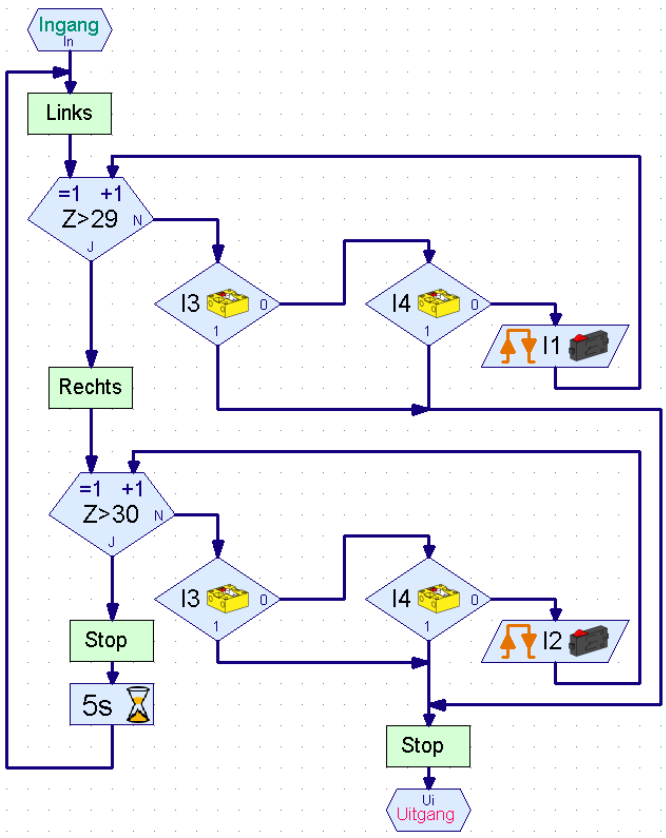
- **Programmeer eerst de functie "Licht zoeken". De robot moet zich daarbij langzaam ten minste 360° ronddraaien om te zoeken. Als de robot tijdens het zoeken een lichtbron vindt, stopt de robot. Vindt hij niets, dan draait hij nog eens 360° in de andere richting. Als hij dan nog altijd geen lichtbron vindt, moet hij 5 seconden wachten en daarna opnieuw gaan zoeken.**
- **Als het licht gevonden is, moet het model door de lichtbron worden aangestuurd. Als de lichtbron naar links of rechts beweegt, moet de robot de bewegingen van het licht volgen. Als hij het contact verliest, moet het programma opnieuw beginnen met het licht zoeken. Probeer de robot met een zaklamp te lokken en door een hindernisparcours te leiden.**

Tips:

- Pas voor de verschillende rijrichtingen de subprogramma's toe die je al voor het basismodel hebt geprogrammeerd. Zodra het programma Basismodel 2.rpp geopend is, vind je **in het venster met elementgroepen** van ROBO Pro onder **Geladen programma** het programma Basismodel 2 en daaronder de subprogramma's die zich bevinden in het programma Basismodel 2. Deze subprogramma's kun je dan eenvoudig invoegen in je nieuwe programma.
- Voor het subprogramma "Licht zoeken" gebruik je het element **Getallenlus**. (Zie voor de beschrijving van dit element het handboek van ROBO Pro).
- In de lus tussen aansluiting "N" en aansluiting "+1" vraag je de fototransistors af en tel je een impuls op de impulscontactschakelaar I1. De lus wordt zo vaak doorlopen totdat de robot licht heeft gevonden of zich 360° heeft gedraaid. Je moet gewoon even uitproberen hoe vaak hij de lus voor een volledige draai moet doorlopen. Stel de waarde "Z" in het element Getallenlus op basis daarvan in.
- Vervolgens programmeer je precies op dezelfde manier een tweede lus voor het zoeken met omgekeerde draairichting.
- Als de robot licht vindt, stopt hij en verlaat hij het subprogramma.
- Hieronder volgt het complete subprogramma voor het licht zoeken:



Getallenlus

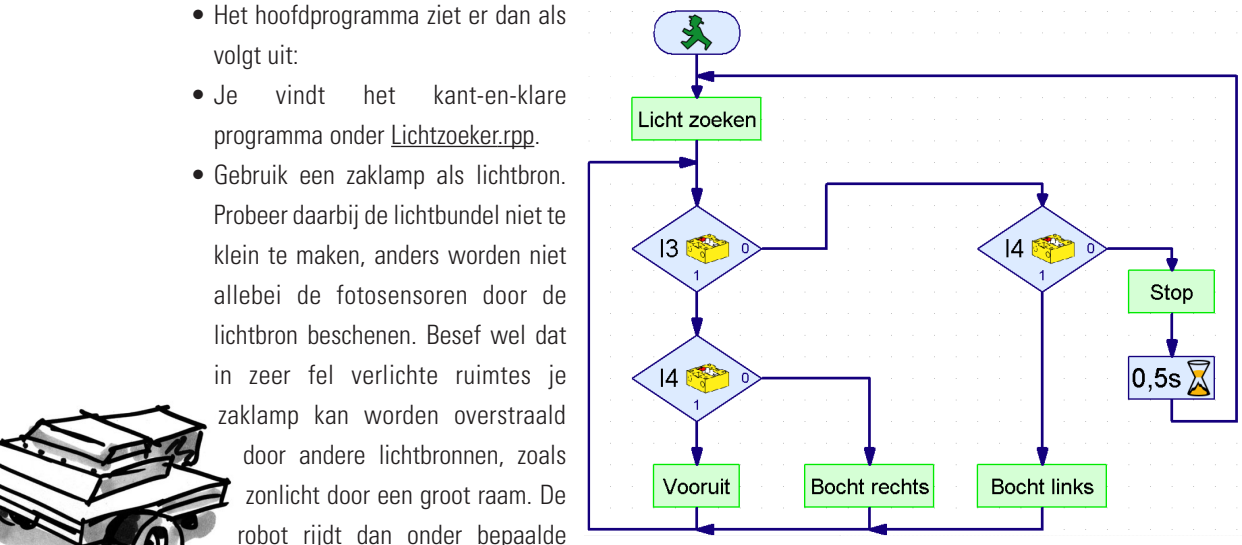


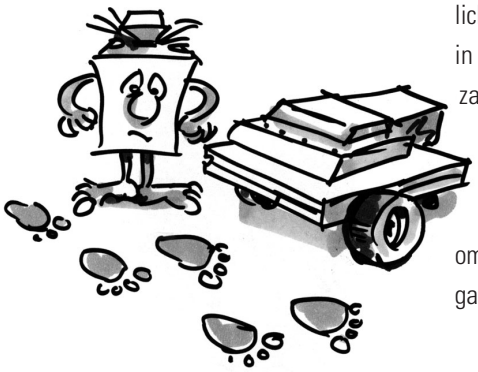
- In het hoofdprogramma vraag je weer de fototransistors af en stuur je de motoren aan op basis van welke fototransistor licht ziet:

Licht bij I3 en I4	**Vooruit**
Licht alleen bij I3	**Bocht naar rechts**
Licht alleen bij I4	**Bocht naar links**
Geen licht gevonden	**Stop en terug naar het subprogramma Licht zoeken**

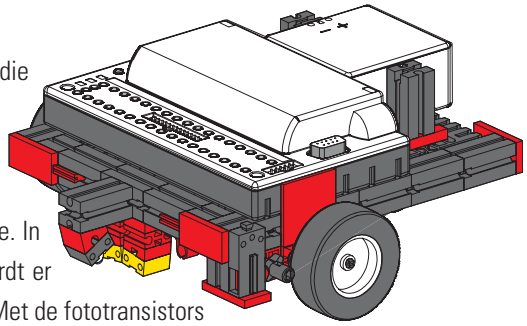
- De bocht naar rechts of naar links genereer je door M1 en M2 op verschillende snelheden te laten lopen bij een gelijke draairichting. Dit leidt tot een zeer harmonische rijstijl.

- Het hoofdprogramma ziet er dan als volgt uit:
- Je vindt het kant-en-klare programma onder Lichtzoeker.rpp.
- Gebruik een zaklamp als lichtbron. Probeer daarbij de lichtbundel niet te klein te maken, anders worden niet allebei de fotosensoren door de lichtbron beschenen. Besef wel dat in zeer fel verlichte ruimtes je zaklamp kan worden overstraald door andere lichtbronnen, zoals zonlicht door een groot raam. De robot rijdt dan onder bepaalde omstandigheden de lamp voorbij en gaat naar het fellere licht toe.





De Spoorzoeker



■ Zoeken en volgen zijn wezenlijke eigenschappen, die intelligente wezens bezitten. Met de Lichtzoeker heb je een robot gebouwd en geprogrammeerd die op directe signalen van zijn doel reageert.
Met de Spoorzoeker passen we een ander zoekprincipe toe. In plaats van doelgericht naar de lichtbron toe te rijden, wordt er nu een zwarte lijn aangebracht die de robot moet volgen. Met de fototransistors is deze opdracht betrekkelijk gemakkelijk op te lossen. Het door de zwarte lijn gereflecteerde licht wordt gemeten en de motoren worden aan de hand daarvan gecorrigeerd. Om te zorgen dat dit nauwkeurig werkt, wordt de lijn door de lamp verlicht. Let op dat de fotosensoren niet als gevolg van een ongunstige configuratie worden verblind door het verstrooide licht van de lamp. In deze samenhang heeft bundeling van het licht van de optische lens van de gloeilamp een zeer goed effect.

Bouw nu het model Spoorzoeker zoals in de bouwinstructies is beschreven.

Opdracht 1:

- **Schrijf eerst een subprogramma om het spoor te zoeken. De robot moet daarbij eenmaal in de rondte draaien.**
- **Vindt hij daarbij geen spoor, dan moet hij een stukje recht vooruit rijden en daar opnieuw gaan zoeken. Om het spoor te herkennen worden de fototransistors afgevraagd.**
- **Als de robot het spoor ontdekt heeft, moet hij dit volgen.**
- **Als het spoor ten einde is of als de robot het spoor kwijtraakt, bijv. omdat de richting van het spoor te sterk verandert, moet het zoeken opnieuw beginnen.**

Tips:

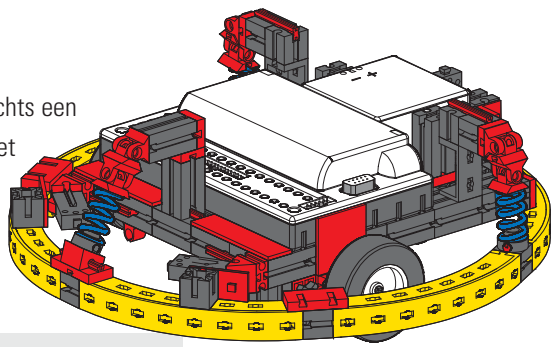
- Na het inschakelen van de lamp moet er even gewacht worden (ca. een seconde) voordat de fototransistors worden afgevraagd. Anders herkent de fototransistor "donker", dat wil zeggen een spoor, waar er geen is, omdat het afvragen al begint voordat de lamp echt fel brandt.
- Als spoor maak je gebruik van ca. 20 mm brede zwarte isoleertape of je tekent met viltstift een zwart spoor van deze breedte op een wit vel papier. De bochten mogen niet te krap zijn, anders verliest de robot het spoor te vaak.
- Controleer eerst met de interfacetest of de fototransistors het spoor ook goed herkennen. Vergeet daarbij niet de lamp in te schakelen.
- Regel de lamp zo af dat op een lichte ondergrond beide fototransistors de waarde 1 opleveren, ook als de motoren M1 en M2 worden ingeschakeld. Als de accu wat zwakker is, gaat de lamp iets minder fel branden als de motoren worden ingeschakeld. Als de lamp niet goed is afgeregeld, kan het zijn dat een fototransistor "donker" aangeeft hoewel hij helemaal geen spoor heeft gevonden.
- Het spoorzoeken functioneert bijna hetzelfde als het licht zoeken. Je moet alleen het zoeken zo aanpassen, dat het model bij een niet-geslaagde zoekpoging na een volle draai in de rondte een stuk recht vooruit rijdt, voordat hij verder zoekt.
- Let erop dat het model bij het volgen van het spoor recht vooruit moet rijden, als beide fototransistors de waarde "donker" (=0) opleveren.
- Je vindt het kant-en-klare programma onder Spoorzoeker.rpp.

Opdracht 2:

- **Maak een spoor met verschillende krappe bochten. Welke radius kan het model nog net halen?**
- **Experimenteer bij het corrigeren van het spoor met verschillende snelheden voor M1 en M2. Welke combinatie levert het beste resultaat op?**
- **Zet nu een rond traject uit als spoor. Probeer de snelheden zo te optimaliseren dat de robot een zo kort mogelijke rondetijd haalt. Deze opdracht is prima geschikt voor een wedstrijd tussen meer robots.**

Robot met hindernisherkenning

■ Alle tot dusverre gebouwde robots kunnen een bepaald traject afleggen en een lichtbron of een spoor volgen. Maar wat gebeurt er als ze een hindernis tegenkomen? Waarschijnlijk zullen ze de hindernis opzij duwen of blijft de robot er tegenaan stoten totdat de accu leeg is. Het zou natuurlijk veel intelligenter zijn als de robot de hindernis herkende en op de juiste wijze zou uitwijken. Daarvoor wordt de robot voorzien van een bewegende stootstang rondom met drie contactschakelaars. Met deze stootstang kan hij bepalen of er zich links of rechts van hem of achter hem een hindernis bevindt.
Hoe hij daarop moet reageren, is dan alleen nog een kwestie van programmeren.



Bouw nu eerst het model "Robot met hindernisherkenning". Voor wegbepaling is slechts een contactschakelaar (I1) nodig. Daarom verwijderen we contactschakelaar I2 van het basismodel en gaan we die gebruiken voor de hindernisherkenning.

Opdracht 1 (Niveau 2):

- **De robot moet eerst recht vooruit rijden. Als hij links tegen een hindernis (E4) stoot, moet hij een stukje terug en daarna naar rechts uitwijken.**
- **Als hij rechts tegen een hindernis (E3) stoot, moet hij een stukje terug en daarna naar links uitwijken.**

Tips:

- De hindernisherkenning bij het achteruit rijden wordt momenteel nog niet toegepast.
- In het hoofdprogramma worden de contactschakelaars afgevraagd. Afhankelijk van de contactschakelaar die geactiveerd wordt, wijkt het model naar links of naar rechts uit. Dit gebeurt telkens in een subprogramma.
- Het impulsgetal voor een draai naar rechts moet afwijken van het impulsgetal voor een draai naar links (bijv. 3 impulsen naar rechts, 5 impulsen naar links). Anders kan het zijn dat het model in een hoek terechtkomt en daar niet meer uitkomt, omdat hij altijd evenveel naar links als naar rechts draait.
- Het kant-en-klare programma heet Hindernis1.rpp.

Twee dingen kan de hindernisherkenner nog niet. Tijdens het achteruit rijden herkent hij nog geen hindernissen. Hij merkt ook nog niet of er zich een hindernis recht voor hem bevindt. Hij zou ze echter wel allebei kunnen herkennen. Als tijdens het achteruit rijden I5 wordt ingedrukt, is er een hindernis achter het model. Als tijdens het vooruit rijden I3 en I4 tegelijk worden ingedrukt, bevindt er zich een hindernis recht voor het model. In dit geval zou de robot zich direct 90° kunnen draaien. In totaal hebben we nu dus de volgende mogelijkheden waarop de robot moet reageren:

Hindernis	Contact schakelaarr	Reactie
Rechts	Alleen I3	Naar links uitwijken (ca. 30° draaien)
Links	Alleen I4	Naar rechts uitwijken (ca. 45° draaien)
Voor	I3 en I4	Naar links uitwijken (ca. 90° draaien)
Achter	I5	Wordt alleen afgevraagd tijdens het achteruit rijden. Stoppen en daarna zoals gepland verder uitwijken

Om deze opdracht elegant op te lossen, kunnen enkele nieuwe programma-elementen zoals bijv. bedieners (ook wel operatoren genoemd, bijv. EN, OF) uit niveau 3 van ROBO Pro zeer goed van pas komen. In niveau 3 bestaat tevens de mogelijkheid via oranje pijlen gegevens uit te wisselen tussen verschillende elementen. Schakel daarom in de software over op dit niveau. Daarna kun je het beste eerst even hoofdstuk 5 van het ROBO Pro handboek grondig doorlezen. Je bent dan klaar voor de volgende opdracht.

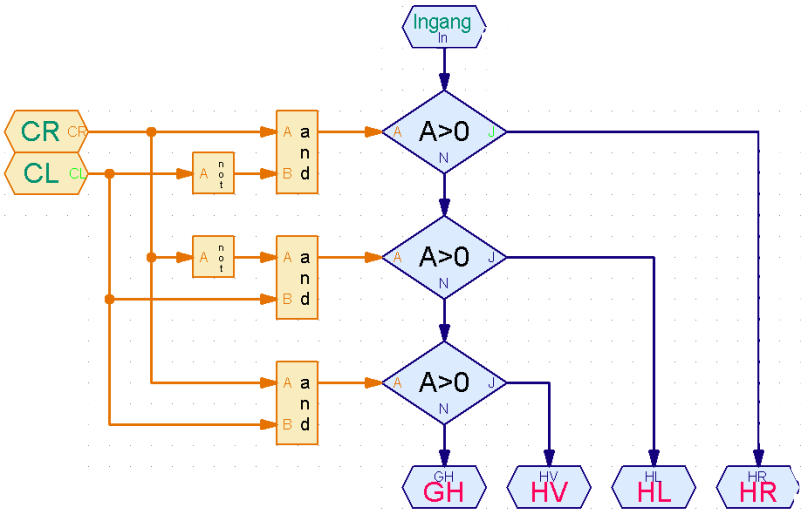
Opdracht 2 (Niveau 3):

- **Stel het hindernisprogramma zo op dat het model reageert zoals in bovenstaande tabel is aangegeven.**
- **Maak daarbij gebruik van de mogelijkheden uit ROBO Pro Niveau 3.**

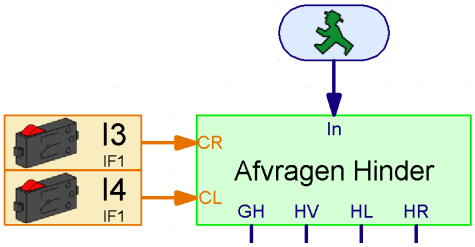
Tips:

- Met behulp van bedieners (of operatoren) worden via een subprogramma "Afvragen Hindernis" de verschillende mogelijke combinaties van contactschakelaars afgevraagd. Voor elke mogelijkheid heeft het subprogramma een eigen uitgang.

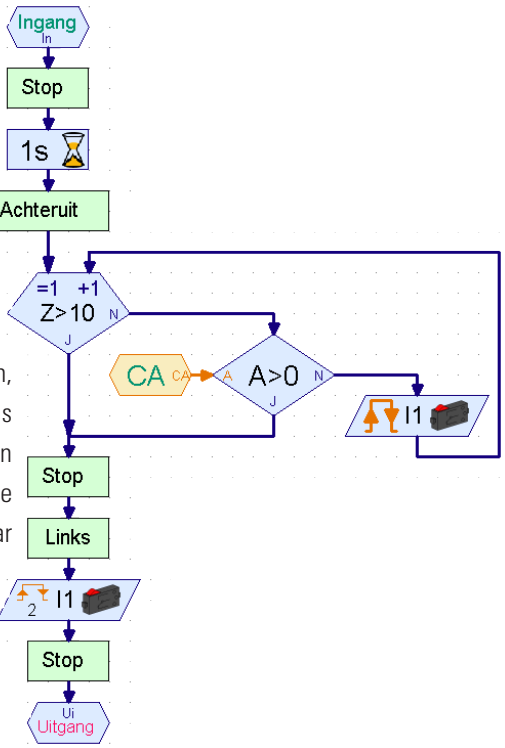
Gegevensinvoer CR = contactschakelaar rechts
Gegevensinvoer CL = contactschakelaar links
Uitgang GH = geen hindernis
Uitgang HV = hindernis voor
Uitgang HL = hindernis links
Uitgang HR = hindernis rechts



- Om te zorgen dat het onmiddellijk duidelijk is welke contactschakelaar wordt afgevraagd, plaats je de oranje contactschakelaarelementen in het hoofdprogramma en verbind je ze via gegevensinvoeren met het subprogramma.

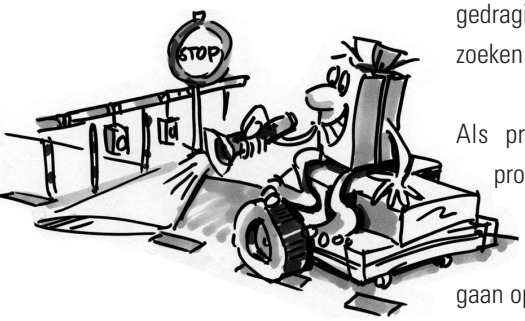


- In de verschillende subprogramma's voor het uitwijken wordt tijdens het achteruit rijden I5 afgevraagd. Het model rijdt dan zolang achteruit totdat het ingestelde impulsgetal wordt bereikt of I5 wordt ingedrukt. I5 wordt weer in het hoofdprogramma gezet, zodat onmiddellijk zichtbaar is in welk subprogramma hij wordt afgevraagd.
- Je vindt het complete programma onder Hindernis2.rpp.



Een voordeel van de in deze opdracht toegepaste programmeertechniek is dat je direct in het hoofdprogramma kunt zien welke contactschakelaar in welk subprogramma wordt afgevraagd. Als je de ingang wilt veranderen, hoef je dat slechts op één plek te doen en hoef je niet in alle subprogramma's te zoeken waar de contactschakelaar overal verstopt kan zitten. Bovendien kun je met de operatoren op zeer aanschouwelijke wijze logische koppelingen maken. In principe kan dat ook met Vertakkingselementen, maar dit wordt al snel zeer onoverzichtelijk als er meer situaties moeten worden afgevraagd.

Lichtzoeker met hindernisherkenning



■ We hebben nog lang niet alle mogelijkheden uitgeprobeerd die de ROBO Mobile Set biedt. Daarom gaan we nu de twee functies licht zoeken en hindernis herkennen met elkaar combineren. Uit wetenschappelijk opzicht is de robot dan van twee gedragingen voorzien. Omdat echter niet allebei de gedragspatronen tegelijk actief kunnen zijn, krijgen ze verschillende prioriteiten. De robot is normaal gesproken op zoek naar licht. Herkent hij een hindernis, dus een gevaar voor zichzelf, dan wordt de gedraging "hindernis vermijden" actief. Als alles in het groene bereik (veilig) blijft, kan de robot verder zoeken naar de lichtbron.

Als professionele softwareontwikkelaars een dermate gecompliceerde opdracht aanpakken, programmeren ze er niet zomaar op los, maar maken ze gebruik van een bepaalde strategie voor het ontwikkelen van het programma. Een van deze methodes noemt men "Top-Down ontwerpen". Bij deze procedure wordt het totale systeem van bovenaf gedefinieerd, zonder al direct in te gaan op alle details. Van deze methode maken wij voor dit probleem ook gebruik.

Opdracht 1 (Niveau 3):

Leer de robot de volgende gedragingen aan:

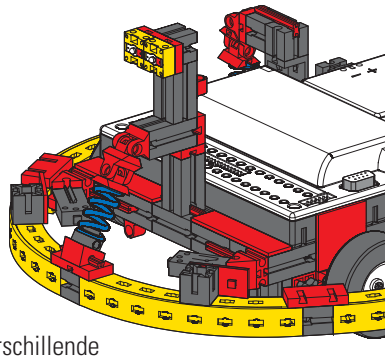
- **Zoek een lichtbron.**
- **Volg deze zodra je hem gevonden hebt.**
- **Als er onderweg een hindernis opduikt, moet je ervoor uitwijken.**
- **Zoek daarna opnieuw een lichtbron.**

Maak voor de oplossing gebruik van de programma-elementen uit ROBO Pro Niveau 3.
Los de opdracht "van bovenaf" op volgens de Top-Down-aanpak.

Tips:

Verdeel de opdracht eerst in drie onderdelen:

- Afvragen of de robot een lichtbron ziet (subprogramma "Licht")
- Afvragen of hij tegen een hindernis botst (subprogramma "Hindernis")
- Afhankelijk van deze resultaten laat je de robot weten, wat hij moet doen (subprogramma "Rijden")



Voor de subprogramma's "Licht" en "Hindernis" bedenk je nu welke verschillende situaties de robot kan waarnemen. Je wijst een getalswaarde toe aan elke situatie en deze sla je met behulp van een Bevelsegment op in een variabele. Uit elke situatie volgt dan een reactie die wordt uitgevoerd in het subprogramma "Rijden".

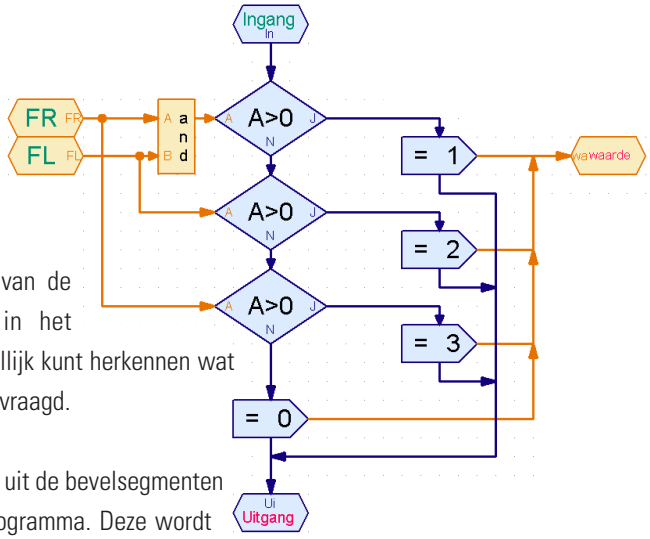
Subprogramma Licht:

Nr.	Situatie	Toestand van de sensoren	Reactie
0	Geen lichtbron beschikbaar	I6=0; I7=0	Licht zoeken
1	Lichtbron recht voor robot	I6=1; I7=1	Recht vooruit rijden
2	Lichtbron links van de robot	I7=1	Bocht naar links maken
3	Lichtbron rechts van de robot	I6=1	Bocht naar rechts maken

Subprogramma Hindernis:			
Nr.	**Situatie**	**Toestand van de sensoren**	**Reactie**
4	**Hindernis direct voor de robot**	**I3=1; I4=1**	**90° uitwijken**
5	**Hindernis rechts van de robot**	**I3=1**	**Naar links uitwijken**
6	**Hindernis links van de robot**	**I4=1**	**Naar rechts uitwijken**

Nu moet je deze inzichten nog met programma-elementen reproduceren in ROBO Pro.

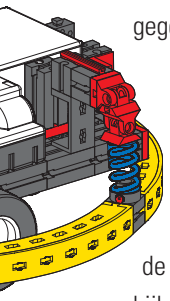
Subprogramma Licht:



FR=Fototransistor rechts
FL=Fototransistor links

De elementen voor het afvragen van de fototransistors plaats je weer in het hoofdprogramma, zodat je onmiddellijk kunt herkennen wat er in het subprogramma wordt afgevraagd.

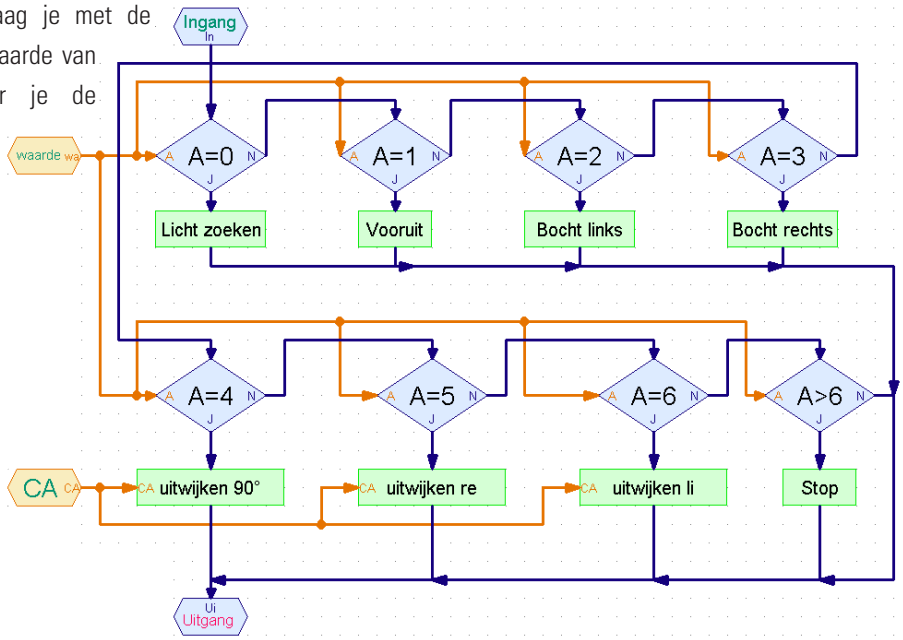
Ook de variabele waarin de waarde uit de bevelsegmenten is opgeslagen komt in het hoofdprogramma. Deze wordt in meer subprogramma's toegepast. Je verbindt deze via een gegevensuitgang met het subprogramma.



Het subprogramma "Hindernis" maak je volgens hetzelfde principe als het subprogramma "Licht".

In het subprogramma "Rijden" vraag je met de Vertakkingselementen de actuele waarde van de variabelen af en programmeer je de bijbehorende reactie:

CA=contactschakelaar achter



Als laatste detail moet je nu nog de subprogramma's maken die in dit subprogramma worden toegepast. Maar wacht eens even! Die hebben we bijna allemaal al. Zo kun je het subprogramma "Licht zoeken" bijvoorbeeld al kopiëren uit het programma voor het model Lichtzoeker. Als je niet meer weet hoe dit moet, moet je hoofdstuk 4 van het ROBO Pro handboek even doorlezen.

Let op:
Bij het model Lichtzoeker waren de fototransistors aangesloten op ingang I3 en I4. Nu zijn ze echter aangesloten op I6 en I7. Bovendien werd daar voor het tellen van de impulsen bij het draaien naar links contactschakelaar I1 afgevraagd en bij het draaien naar rechts I2. Nu is alleen nog maar I1 beschikbaar voor het tellen van de impulsen, maar dat werkt even goed. Je moet het subprogramma "Licht zoeken" na het kopiëren dus aanpassen. Omdat het afvragen van de contactschakelaar in het subprogramma verstopt zit, kun je er gemakkelijk overheen kijken. Dit gebeurt niet meer als je de ingangen in het hoofdprogramma plaatst en deze via gegevensinvoeren met het subprogramma verbindt. Bij de Lichtzoeker kende je deze mogelijkheid echter nog niet.

Ook de subprogramma's voor het uitwijken zijn al beschikbaar, namelijk bij het model voor hindernisherkenning. Hier is de contactschakelaar I5, die extra wordt afgevraagd bij het achteruit rijden, al naar buiten gebracht.

Je vindt het kant-en-klare programma onder Hindernis-Licht.rpp.
Het hoofdprogramma ziet er op het eerste gezicht zeer overzichtelijk en eenvoudig uit. Toch zitten er heel wat hoofdbrekens in de subprogramma's. Maar met behulp van de stapsgewijze procedure volgens de Top-Down-methode kun je dus ook een dergelijk complex probleem oplossen.

Mocht je trouwens iemand kennen die ook een bouwdoos voor de ROBO Mobile Set heeft, dan kunnen jullie nog verder experimenteren met deze robots. Monteer dan een lichtbron aan beide robots. Ze gaan elkaar dan zoeken.

Robot met randherkenning

■ Nu je in het vorige voorbeeld hebt gezien hoe je te werk moet gaan om een complexer probleem te programmeren, kun je de mobiele robot een zeer belangrijke gedraging bijbrengen. Hij moet namelijk leren niet van de tafel te vallen. Als de robot tegen een hindernis rijdt, dan maakt dat voor hem in de meeste gevallen niets uit. Als hij echter van een tafel valt en één meter lager op de grond valt, dan kan hij behoorlijk beschadigd raken, ondanks het feit dat de componenten van fischertechnik zeer robuust zijn. Daarom krijgt de robot sensoren zodat hij randen kan herkennen. Deze randdetectoren bestaan telkens uit een contactschakelaar, die geactiveerd wordt door een draaibaar gelagerd wiel. Dit wiel kan ook omhoog en omlaag bewegen. Zodra het wiel voorbij de rand van de tafel komt, valt het naar beneden. De contactschakelaar wordt nu niet meer geactiveerd en het programma weet dat het model zich bij een afgrond bevindt en reageert daar op gepaste wijze op. De robot heeft in totaal 4 randdetectoren, zodat hij zowel tijdens het vooruit- als het achteruitrijden aan beide kanten naar afgronden kan zoeken. Als gevolg daarvan heeft dit model geen impulscontactschakelaar voor trajectmeting. Het afgelegde traject wordt aangestuurd door de inschakelduur van de motoren.
Bouw eerst het model zoals in de bouwinstructies is beschreven.

Controleer zorgvuldig of de randdetectoren goed reageren:

- als het model bij de tafelrand komt en de contactschakelaar weer precies wordt ingedrukt;
- op het moment dat het wiel weer op de tafel staat.

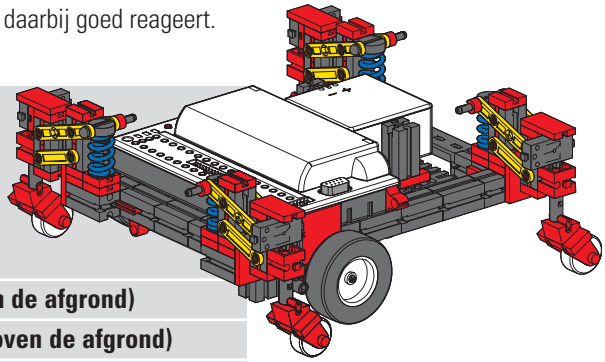
Misschien moet je de ene of andere contactschakelaar nog iets omlaag of omhoog verstellen.

Opdracht 1 (Niveau 3):

- **Bedenk eerst hoe de robot moet reageren als hij bij een afgrond komt.**
- **Als je hier iets dieper over nadenkt, zal het je opvallen dat er erg veel combinaties mogelijk zijn van sensoren die zich boven de afgrond kunnen bevinden. Er kan slechts een van de 4 detectoren worden geactiveerd, 2 of 3 verschillende tegelijk of alle 4 de sensoren.**
- **Hoe moet de robot op elke verschillende situatie reageren?**

Tips:

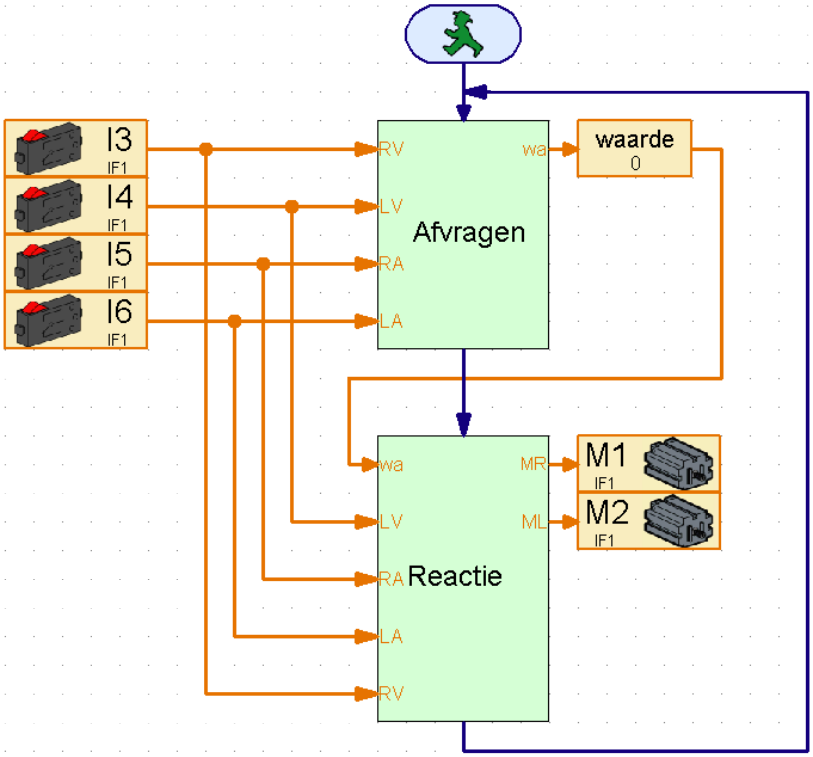
De oplossing vind je in de onderstaande tabel. De sensoren die zich boven de afgrond bevinden (contactschakelaar=0) zijn aangekruist. Elke combinatie krijgt een nummer. In het programma dat je gaat maken, krijgt elke mogelijkheid het bijbehorende getal. Aan de hand van dit getal reageert de robot op de actuele situatie. Maar hierover later meer. Je hoeft je in eerste instantie alleen maar te bedenken hoe de robot moet staan om de desbetreffende combinatie te laten optreden en of hij daarbij goed reageert.



Nr.	Rechts-voor (I3)	Links-voor (I4)	Rechts achter (I5)	Links-achter (I6)	Reactie
0					Vooruit (geen sensor boven de afgrond)
1	●	●	●	●	Stop (alle 4 de sensoren boven de afgrond)
2	●	●	●		Een stukje naar rechts draaien
3	●	●		●	Een stukje naar links draaien
4	●		●	●	Een stukje naar links draaien
5		●	●	●	Een stukje naar rechts draaien
6	●	●			Eerst terug en daarna naar rechts draaien
7	●		●		Een stukje naar links draaien
8	●			●	Een stukje naar links draaien
9		●	●		Een stukje naar rechts draaien
10		●		●	Een stukje naar rechts draaien
11			●	●	Een stukje vooruit rijden
12	●				Eerst terug en daarna naar links draaien
13		●			Eerst terug en daarna naar rechts draaien
14			●		Een stukje vooruit rijden
15				●	Een stukje vooruit rijden

Dat is best wel pittig, hè? Maar wees niet bang, voor dit model is er een kant-en-klaar programma, dat gebruikmaakt van alle voorkeuren van ROBO Pro. Dit heet Randen.rpp.

De belangrijkste elementen bevinden zich in het hoofdprogramma, zodat je de totale procedure kunt begrijpen. Het complexe afvragen van de contactschakelaars en de aansturing van de motoren is in subprogramma's verborgen. Hier eerst het hoofdprogramma:



Het programma begint met het afvragen van de 4 contactschakelaars. Helemaal links zie je welke contactschakelaars worden afgevraagd. Deze zijn via gegevensinvoeren met het subprogramma verbonden. Het subprogramma "Afvragen" bepaalt welke contactschakelaars zijn ingedrukt en levert dan de in de tabel beschreven waarde op. Deze waarde wordt toegewezen aan de gelijknamige variabele, die je weer kunt herkennen in het hoofdprogramma. De waarde van de variabelen wordt doorgegeven aan het subprogramma "Reactie", dat vervolgens onafhankelijk van deze waarde de beide motoren aanstuurt. In het subprogramma "Reactie" worden ook de contactschakelaars nog ingelezen, omdat de randsensoren ook afgevraagd worden, terwijl het model uitwijkt.

Je zou de contactschakelaartoewijzing op de interface evenals de motoruitgangen nu kunnen veranderen zonder dat je alle subprogramma's hoeft door te nemen om te controleren waar er nog een invoerelement of een motorsymbool verstopt zit. Elke ingang en elke uitgang komt slechts eenmaal voor.

Deze programmeertechniek kun je vooral toepassen in situaties waar een subprogramma voor veel verschillende modellen moet worden toegepast en je vooraf nog niet exact weet, welke in- en uitgangen op de interface daarvoor moeten worden gebruikt.

Als je nu nieuwsgierig bent geworden, moet je maar eens in de subprogramma's kijken om te zien of je ze begrijpt. Het programmeerprincipe is hetzelfde als bij het model "Lichtzoeker met hindernisherkenning".

Opdracht 2 (Niveau 3):

Laad het programma in de interface en laat het model op een tafel rijden.

- **Reageert het model altijd correct?**
- **Moet het zich bij bepaalde combinaties van contactschakelaars anders gaan gedragen?**
- **Optimaliseer het programma eventueel als dat nodig is.**

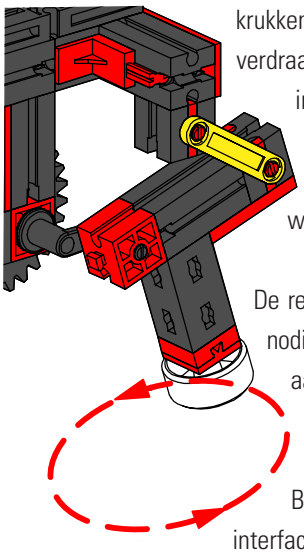
De lopende robot

■ Nadat we ons uitvoerig hebben bezig gehouden met de rijdende robot, stappen we nu over op een andere manier van voortbewegen die we kunnen gebruiken voor mobiele robots: lopen.
De manier van lopen van insecten is prima geschikt als voorbeeld voor de aandrijving van "machinale zespotigen". Bij wat wij zullen aanduiden als het "lopen op drie voeten" worden steeds drie van de zes poten tegelijk opgetild van de grond: de voorste en achterste poot aan de ene kant, samen met de middelste poot aan de andere kant:

Lopen op drie voeten
De poten die op de grond blijven staan (met zwart weergegeven) vormen een stabiele driepoot, zodat het model altijd stevig staat en niet omvalt tijdens het lopen.

De poten van de lopende robot van fischertechnik zijn zo geconstrueerd, dat zij een zogenaamde vierstangsaandrijving vormen. De constructie van de hier toegepaste aandrijving is een soort van schaarblokbeweging. Aangedreven door een kruk voeren de bewegend gelagerde elementen van het drijfwerk schommelbewegingen uit. De afstanden tussen de afzonderlijke scharnierpunten en de positie van het voetpunt (het onderste uiteinde van de poot) zijn zo geselecteerd dat het voetpunt een elliptische beweging beschrijft, als de aandrijfkruk draait. Daardoor ontstaat een beweging, die lijkt op een het uitvoeren van een stap tijdens het lopen.

De 6 krukken die de poten aandrijven, moeten precies worden afgesteld zoals aangegeven in de bouwinstructies. De drie poten die tegelijkertijd op de vloer staan, hebben dezelfde krukstand. De krukken van de 3 poten die op dat moment in de lucht steken, zijn ten opzichte daarvan 180° verdraaid. De juiste stand van de krukken ten opzichte van elkaar garandeert dat het model in de juiste volgorde van stappen op drie voeten kan lopen.



De naafmoeren, waarmee de tandwielen op de assen worden vastgezet, moeten goed worden vastgedraaid, zodat de afstelling van de krukken tijdens het lopen niet verandert.

De rechter- en linkerzijde van het model worden elk door een eigen motor aangedreven (is nodig om bochten te kunnen nemen). Daarom moet je ervoor zorgen dat de middelste poot aan de ene zijde altijd in dezelfde stand staat als de twee buitenste poten aan de andere zijde. Deze synchronisatie gebeurt door de software via contactschakelaar I1 en I2.

Bouw eerst het model zoals in de bouwinstructies is beschreven. Controleer met de interfacetest of alle contactschakelaars en motoren correct zijn aangesloten. Draairichting van de motoren: draairichting linksom = vooruit.

Opdracht 1 (Niveau 1):
Leer de robot lopen.

- **Programmeer het model zo dat het op drie voeten recht vooruit loopt.**
- **Maak gebruik van de contactschakelaars I1 en I2 om de linker- en rechterpoot met elkaar te synchroniseren.**
- **Let er daarbij op dat de twee buitenste poten aan de ene zijde en de middelste poot aan de andere zijde altijd in dezelfde stand moeten staan.**

Tips:

- Breng eerst de poten aan de linker- en rechterzijde in hun uitgangspositie. Schakel daarvoor beide motoren in (draairichting linksom).
- De procedure mag pas verdergaan als contactschakelaar I1 en I2 allebei niet zijn ingedrukt (dit moet worden afgevraagd zodra het model de tweede stap moet nemen).
- Laat de motoren lopen totdat de desbetreffende contactschakelaar (I1 voor M1, I2 voor M2) weer is ingedrukt. Daarbij is het belangrijk dat het model pas begint met de volgende stap als beide contactschakelaars zijn ingedrukt. Dan staan de poten namelijk in de juiste stand ten opzichte van elkaar. Een voorwaarde daarvoor is echter ook dat de krukken die de poten aandrijven correct zijn afgesteld, zoals in de bouwinstructies is beschreven.
- Nu kan de cyclus weer vooraan beginnen en voert de robot zijn tweede stap uit. Het model blijft nu vooruit lopen totdat jij het programma stopt.
- Je vindt het kant-en-klare programma onder Lopende robot1.rpp.

Op dezelfde manier als bij het rijdende basismodel kun je het model nu naar links, rechts of achteruit laten lopen door de motordraairichtingen te veranderen. Voor het tellen van de stappen kun je I1 of I2 gebruiken.

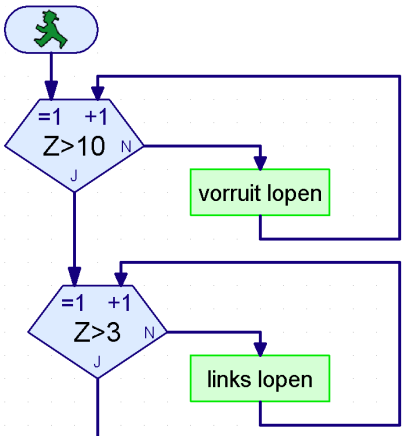
Opdracht 2 (Niveau 2):

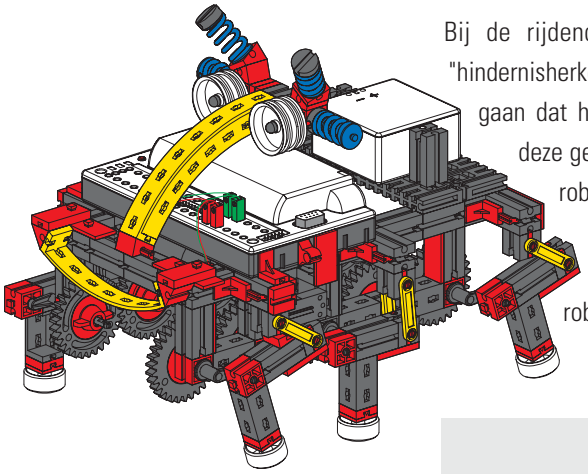
- **Programmeer je model zo, dat het 10 stappen vooruit, 3 stappen naar links, 3 stappen naar rechts en weer 10 stappen achteruit loopt.**
- **Maak voor elke richting een afzonderlijk subprogramma aan.**
- **Maak voor het tellen van de stappen gebruik van het element Getallenlus.**

Tips:

- Kopieer gewoon het programma Laufroboter1.rpp naar een subprogramma.
- Kopieer dit subprogramma zo vaak als nodig is voor de verschillende looprichtingen. Verander de draairichting van de motoren in elk subprogramma zodat het model zich in de gewenste richting verplaatst.
- Gebruik het element Getallenlus om het aantal stappen voor elke draairichting te tellen. Het model voert een stap uit telkens als er een subprogramma wordt doorlopen. Als het programma de lus met het subprogramma 10 maal doorloopt, dan voert het model 10 stappen uit.

Op deze manier kun je je lopende robot een willekeurige reeks stappen bijbrengen (Lopende robot 2.rpp).





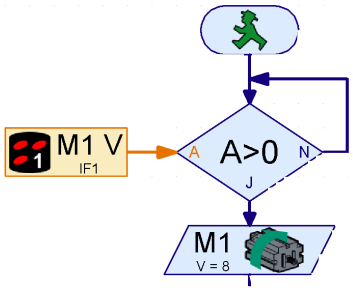
Bij de rijdende robots hebben we het thema "hindernisherkenning" al uitvoerig behandeld. We gaan dat hier niet nog eens herhalen. Probeer deze gedraging echter eens door de lopende robot te laten uitvoeren. De sensoren daarvoor vind je in de bouwdoos. Bij het programmeren kun je de rijdende robot als voorbeeld nemen. Veel succes!

Uitbreidings-mogelijkheden

■ De ROBO Interface biedt nog veel meer functionaliteit dan we tot nu toe hebben laten zien met de mobiele robots. Daarvoor heb je echter extra componenten nodig die niet zijn meegeleverd bij deze bouwdoos. Omdat deze echter zeer interessant zijn voor het maken van robots, willen we een aantal van die componenten hier kort presenteren.

Infrarood handzender

■ De ROBO Interface bevat een infrarood ontvangerdiode voor de handzender uit de IR Control Set art.-nr. 30344. In de ROBO Pro sofware kun je daarmee de toetsen van de handzender als digitale ingangen afvragen en zo bijv. de motoren in- en uitschakelen.



Als programmavoorbeeld hebben wij een afstandsbesturing voor de looprobot geprogrammeerd. Met de 4 ovale pijltjestoetsen op de afstandsbediening kun je het model vooruit, achteruit, naar links en naar rechts sturen. Vooraf moet je echter wel het programma Lopende robot-IR.rpp op de interface laden.

Een ander geniaal programma met betrekking tot de afstandsbediening is het programma Mobile-Teach-IR.rpp. Met dit Teach-In-programma kun je een van de rijdende robots, bijv. de Eenvoudige robot of het Basismodel, op afstand besturen. Het model registreert daarbij het afgelegde traject en kan dit daarna willekeurig vaak herhalen. Het opgeslagen traject wordt echter gewist als het programma gestopt wordt.
Hiermee wordt een programma mogelijk gemaakt via het programma-element "Lijst" in ROBO Pro. In dit element kun je een groot aantal waarden opslaan en weer uitlezen (zie ook het handboek van ROBO Pro). Het programma op zich is behoorlijk complex, maar de toepassing ervan is zeer eenvoudig:

1. Laad het programma Mobile-Teach-In.rpp in het FLASH-geheugen van de ROBO Interface en start het.
2. Druk op de afstandsbediening op de toets **M1** ▸/▸▸. De "Leerprocedure" wordt gestart.
3. Stuur het model met de ovale pijltjestoetsen in de gewenste richting.
4. Druk op de toets **M2** ▸/▸▸ Het afgelegde traject wordt opgeslagen.
5. Druk op de toets **M3** ▸/▸▸ Het opgeslagen traject wordt afgelegd.

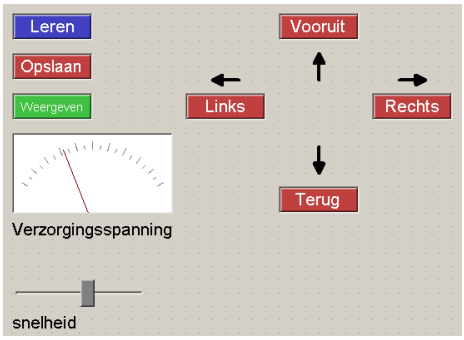
Met dergelijke toepassingen wordt het programmeren van robots kinderspel! Realiseer je wel dat het opgeslagen traject gewist wordt, zodra je het programma met de toets Prog. op de interface stopt.

ROBO RF Data Link

■ De ROBO RF Data Link art.-nr. 93295 vervangt de interfacekabel tussen de pc en de interface door draadloze datatransmissie. Dat is heel prettig. Ten eerste hoef je dan niet telkens als je een programma op de interface laadt de kabel aan te sluiten en weer los te koppelen. Ten tweede kun je programma's draadloos in de online-modus laten werken. Je kunt de fouten dan veel gemakkelijker vinden dan in de download-modus. En tenslotte kun je de mobiele robots in de online-modus via een bedieningsveld in ROBO Pro op dezelfde manier als met de IR-afstandsbediening on line via het beeldscherm besturen. In tegenstelling tot de afstandsbediening kun je op het beeldscherm bovendien ook nog gegevens bekijken die de interface verstrekt, zoals waarden van variabelen of analoge ingangen, de voedingsspanning die de accu levert en de snelheid van de motoren.

Als voorbeeld hebben we het Teach-In-programma gemodificeerd en sturen we de rijdende robot via een bedieningsveld aan. Dit programma heet Mobile-Teach-RF.rpp. Je kunt het natuurlijk ook met de interfacekabel proberen. Dat is echter vrij lastig. De actieradius van het model is dan beperkt, de kabel draait zich op en de robot kan niet goed meer draaien. De kans is zeer groot dat je daarna direct op pad gaat om de RF Data Link aan te schaffen.

Laad het programma Mobile-Teach-RF.rpp.
Schakel in de Functiebalk van het hoofdprogramma over naar Bedieningsveld. Daarna start je het programma in de Online-modus. Nu kun je het model met de knoppen in het bedieningsveld besturen en programmeren.



1. Druk op de knop "Leren". De "Leerprocedure" wordt gestart.
2. Stuur het model met de pijltjesknoppen in de gewenste richting.
3. Druk op de knop "Opslaan". Het afgelegde traject wordt opgeslagen.
4. Druk op de knop "Weergeven". Het opgeslagen traject wordt afgelegd.

Ook hier raakt het opgeslagen traject verloren, als het programma wordt beëindigd.

Meer details over het maken van bedieningsvelden vind je in het handboek bij ROBO Pro.

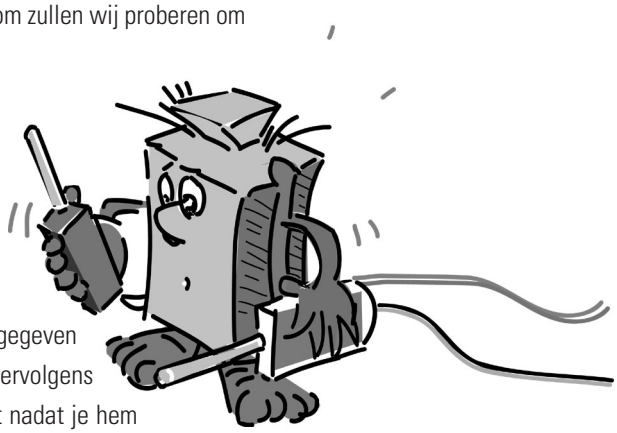
ROBO I/O-Extension

■ Als je een model met zo veel sensoren en motoren bouwt dat de in- en uitgangen van de ROBO Interface niet voldoende zijn, kun je een ROBO I/O-Extension art.-nr. 93294 op de interface aansluiten. Daarmee heb je de beschikking over nog eens 8 digitale ingangen, 4 motoruitgangen en een analoge weerstandsingang. Op deze I/O-Extension kun je een tweede module aansluiten en daarna weer een derde module, die vervolgens allemaal worden aangestuurd door een ROBO Interface. In totaal heb je dan de beschikking over 16 motoruitgangen, 32 digitale ingangen, 5 analoge weerstandsingangen, 2 analoge spanningsingangen en 2 ingangen voor afstandssensoren.

Als je dan nog niet genoeg hebt, kun je via de pc ook meer interfaces in de Online-modus besturen, bijv. een op een seriële COM-interface, een op de USB-poort of 2 interfaces op de USB-poort en dan ook nog eens elk met max. 3 ROBO I/O-Extensions! Kun jij het nog helemaal volgen? Om dit allemaal goed op een rijtje te krijgen, kun je hoofdstuk 6 van het ROBO Pro handboek doornemen.

Problemen verhelpen

■ Experimenteren kan leuk zijn. Zolang alles tenminste naar wens werkt. Meestal is dat ook het geval. Maar helaas niet altijd.
Pas als een model niet werkt, weet je of je het mechanische principe goed hebt begrepen, waarna je de fout meestal snel kunt vinden.
Bij mechanische fouten kun je in ieder geval nog iets zien (foute montage) of voelen (het draait moeilijk). Als er elektrische problemen bijkomen dan wordt het al lastiger.
Professionals gebruiken allerlei meetinstrumenten om fouten op te zoeken, zoals een spanningsmeter of een oscillograaf. Dergelijke apparatuur hebben we niet allemaal in huis. Daarom zullen wij proberen om een fout met eenvoudige middelen te lokaliseren en te verhelpen.



Kabelmontage

Voordat we met onze experimenten beginnen, moeten we eerst een paar componenten uit de fischertechnik-bouwdoos voorbereiden. Zo moet je bijvoorbeeld de meegeleverde stekkers aan de verschillende draden bevestigen.
Daarvoor moet je de kabels eerst op maat maken. Meet daarvoor de aangegeven lengte af en maak de kabel op maat en strip het uiteinde. Elke kabel wordt vervolgens doorgemeten. Daarvoor heb je de accu en de lamp nodig. Als de lamp brandt nadat je hem aan de accu hebt aangesloten, dan is de kabel in orde. Controleer ook of de kleuren kloppen: rode stekker aan rode kabel en groene stekker aan groene kabel.

Interfacetest

Als het programma (dit geldt ook voor het meegeleverde programma) niet met ons model samenwerkt, dan starten we de interfacetest. Dit hulpprogramma stelt je in staat de in- en uitgangen apart te testen. Werken de sensoren? Draaien de motoren in de juiste richting? Bij al onze mobiele robots zijn de motoren zo aangesloten, dat bij draairichting=links het wiel of de poot vooruit beweegt. Als hier alles in orde is, dan zoeken we naar de mechanische oorzaak.

Losse contacten

Een veel voorkomende fout zijn losse contacten. Zo kunnen de aansluitstekkers los in de aansluitbussen zitten. Is dit geval dan moet je de contactveren van de stekker met een kleine schroevendraaier iets uit elkaar drukken. Wees daarbij wel voorzichtig want als je de contactveren te veel verbuigt kunnen de contacten breken of kun je de stekker moeilijk insteken.
Een andere oorzaak voor losse contacten is dat de schroefjes van de klemverbindingen in de stekker zijn losgeraakt. Draai ze voorzichtig vast! Controleer dan tegelijkertijd of geen van de dunne koperdraadjes zijn afgebroken.

Kortsluiting

Het zou kunnen gebeuren dat er kortsluiting optreedt als een kabel verkeerd wordt aangesloten. Dan werkt ook niets meer zoals het hoort. In de accu is een zekering ingebouwd, die de stroom uitschakelt als de stroom of de temperatuur te hoog wordt. Ook de uitgangen van de interface worden bij oververhitting uitgeschakeld.
Een kortsluiting kan ook optreden als het schroefje van de elektrische stekkers waarmee de kabel wordt vastgeklemd niet goed wordt vastgedraaid. Het kan dan buiten de stekker uitsteken. Als er dan twee

stekkers in twee bussen naast elkaar op de interface worden gestoken waarbij de schroefjes elkaar raken, dan ontstaat er kortsluiting. Daarom moeten de schroefjes altijd goed worden vastgedraaid en moet je de stekkers zo insteken dat de schroefjes elkaar niet kunnen raken.

Voeding

Als de apparatuur plotseling om onverklaarbare reden uitvalt, kan het zijn dat de accu bijna leeg is. De spanning daalt kort even als er een last extra wordt ingeschakeld (motor aan) waarmee een reset voor de processor wordt gegenereerd op de interface. Op de ROBO Interface gaat een rode LED branden als de spanning van de voeding te laag is. De accu moet dan worden opgeladen.

Programmeerfouten

Als er fouten optreden bij programma's die je zelf hebt geschreven en je het probleem zelf niet kunt vinden, moet je voor de zekerheid een meegeleverd programma draaien dat het doel van je eigen programma zo dicht mogelijk benadert, om elektrische of mechanische defecten te kunnen uitsluiten. In de Online-modus kun je het verloop van het programma op het beeldscherm volgen. Als het programma vanaf een bepaalde positie niet meer verdergaat, kun je daar de oorzaak zoeken, bijv. verkeerde ingang of motor geselecteerd, bij een vertakking een verkeerde waarde afgevraagd of J/N-aansluitingen verwisseld.

Als dit alles niet tot het gewenste resultaat leidt, kun je altijd nog contact opnemen met de serviceafdeling van fischertechnik (e-mail: info@fischertechnik.de).

Of bezoek ons op internet op www.fischertechnik.de. Daar vind je een forum, chatmogelijkheden, een ruilmarkt, een galerie en kun je gratis lid worden van de Fischertechnik Fanclub.

We wensen je nog heel veel uren plezier en verbazing met de ROBO-Mobile-Set.

Contenidos

Para qué necesitamos los robots?

■ El concepto de robot se utilizó por primera vez en 1923 en la novela de Karel Capek El Golem. Se trataba de una figura de fabricación artificial concebida para reemplazar a los trabajadores humanos gracias a sus funcionalidades.

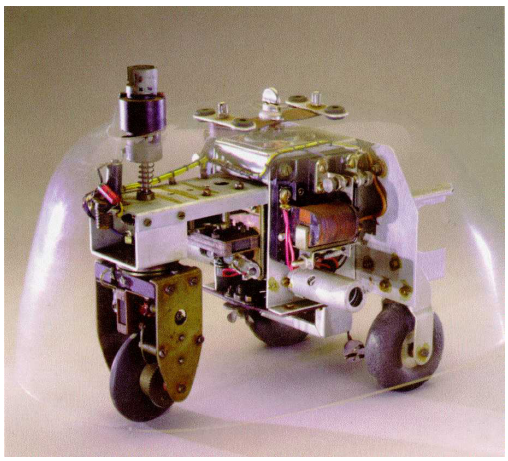
En los años treinta y cuarenta del pasado siglo surgió del robot una especie de autómata. Los diversos intentos que se hicieron para dotar a los robots de características humanas (por ejemplo, ponerles cabeza con luces intermitentes a modo de ojos) se nos antojan hoy en día una simple curiosidad sin mayor utilidad ni importancia. Poco hay que decir acerca de la movilidad y sobre todo de la inteligencia de estas máquinas. Puesto que el principio del control ha tenido una gran influencia sobre la robótica, el advenimiento de los circuitos electrónicos trajo consigo una mayor dosis de realismo en la construcción de los robots. La cuestión de la "inteligencia" de los robots continúa siendo hoy en día objeto de investigación y análisis en innumerables empresas, institutos y universidades.

■ Las primeras soluciones parecieron estar a punto de llegar de manos de la llamada cibernética. El término "cibernética" procede del griego "kybernetes", una palabra que designaba al piloto de las naves de remos de la Antigua Grecia. El piloto debía determinar la posición de la nave y calcular el rumbo necesario para alcanzar su destino.

De ello se desprende que el objetivo de la cibernética es precisamente dotar al robot de "inteligencia". Pero qué entendemos, para empezar, por un comportamiento inteligente?

Vamos a tratar de aclarar este punto con ayuda de un experimento mental. Todos hemos observado alguna vez el comportamiento de una polilla atraída por el círculo de luz de una bombilla. La polilla reconoce la fuente de luz y vuela decidida hacia ella, para poco antes del choque apartarse de la lámpara. Es evidente por este comportamiento que la polilla detecta la fuente lumínica, localiza un camino que la lleve a ella y emprende el vuelo hacia allí. Estas capacidades se basan en el modelo de conducta inteligente que los insectos poseen como instinto.

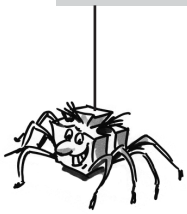
Lo que nosotros intentamos es trasladar estas capacidades a un sistema técnico. Debemos no sólo detectar la fuente de luz (sensores ópticos) y llevar a cabo un movimiento (controlar un motor), sino también establecer una conexión práctica y útil entre la detección y el movimiento (programa).



■ El experimento mental que acabamos de realizar fue llevado a la práctica en los años cincuenta por Walter Grey.

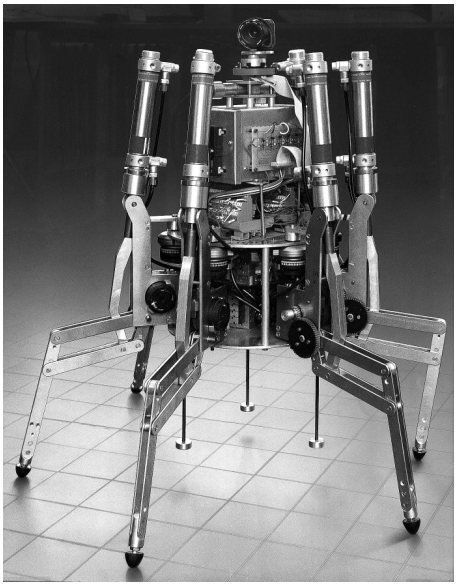
Con ayuda de sensores, motores y circuitos electrónicos sencillos consiguió crear una serie de animales cibernéticos que exhibían comportamientos muy específicos, como por ejemplo el de la polilla. La fotografía muestra una reproducción de la tortuga cibernética que se muestra en el Museo Smithsoniano de Washington.

También nosotros nos basaremos en la misma reflexión e instilaremos en nuestros robots los modelos de conducta correspondientes, transformándolos en programas para que los robots puedan comprenderlos.



■ Pero para qué necesitamos robots móviles? Vamos a tratar de aplicar a un ingenio técnico el comportamiento de nuestra polilla imaginaria. Para ello comencemos por un ejemplo sencillo: la detección lumínica. Modificamos la fuente de luz trazando una línea luminosa (la pauta o línea directriz) en el suelo y variamos la dirección de los sensores de manera que apunten hacia abajo y no hacia adelante. Con ayuda de la línea directriz, un robot móvil puede orientarse en un almacén, por ejemplo. Otra información adicional, como puede ser en forma de un código de barras que aparezca en determinados puntos de la línea, hace que el robot lleve a cabo otras acciones determinadas al alcanzar esos puntos concretos, como por ejemplo levantar o depositar un palé.
Hoy en día ya existen sistemas robóticos de estas características. Los grandes hospitales se hallan recorridos en parte por grandes vías de transporte de material de desecho, como es la ropa de cama que va a la lavandería. El transporte de estos materiales por parte de personal sanitario resulta caro e implica grandes esfuerzos físicos; además, este tipo de tareas reducen el tiempo que el personal sanitario puede dedicar a los pacientes.

■ Desde hace varios años los científicos trabajan en una forma de movimiento más compleja y muy extendida en la naturaleza: la marcha o la carrera. Se están desarrollando robots dotados de la capacidad de dar pasos sobre patas o piernas. Un ejemplo de robot andador de seis patas es Aquiles, un ingenio electroneumático desarrollado en la Real Academia Militar de Bruselas. Equipado con una cámara en la parte superior y en cada una de las seis patas, este robot está diseñado para reaccionar mecánicamente ante discontinuidades de la superficie sobre la que se desplaza (obstáculos y hoyos).
Estas máquinas andadoras serían de especial utilidad en aquellos lugares que resultan inaccesibles para los vehículos de ruedas o cadenas, como por ejemplo cuando hay que transitar por terrenos extremadamente blandos o accidentados, superar obstáculos, subir escaleras, salvar zanjas o llegar a los puntos de peor acceso y mayor peligrosidad de centrales nucleares, galerías de minas u operaciones de rescate.

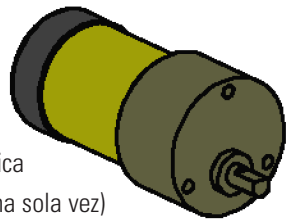


No cabe duda de los robots móviles pueden llegar a desempeñar un papel de importancia en la sociedad moderna.

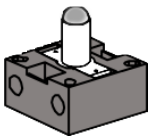
Los robots fischertechnik

■ Cómo podemos construir robots con los kits de construcción modular fischertechnik? Para construir un robot necesitamos, además de los sensores (por ejemplo pulsadores) y actuadores (por ejemplo motores) un buen número de componentes mecánicos que nos permitan montar un modelo. El ROBO Mobile Set que fischertechnik ofrece en su gama de kits de construcción modular constituye la base ideal para hacerlo. Este kit contiene los siguientes sensores y actuadores:

Actuadores



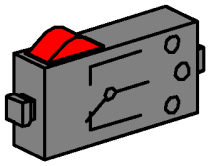
Motor eléctrico:
Dos de estos potentes motores de corriente continua (9VDC/2,4W) con transmisión incorporada y una relación de desmultiplicación de 50:1 propulsan los modelos de robot móvil (la relación de desmultiplicación significa que, por cada 50 revoluciones del motor, los árboles que de él salen giran una sola vez)



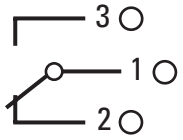
Lámpara lenticular:
Esta lámpara incandescente (9VDC/150mA) sirve para emitir señales luminosas sencillas.
El globo de la bombilla tiene una lente integrada que concentra la luz emitida. Si se dirige el rayo de luz hacia un sensor de luminosidad (un fototransistor, como veremos más adelante) puede construirse una barrera de luz que distinga la luz de la oscuridad. La bombilla también se puede utilizar para indicar determinados estados o como lámpara intermitente para emitir advertencias de peligro. En el kit modular la lámpara se utiliza en conjunción con 2 fototransistores como sensor especial de detección de líneas.

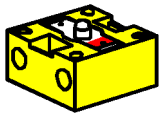
Sensores

■ Dentro de los sensores existen dos tipos: los analógicos y los digitales.
El pulsador es un ejemplo de sensor digital. Los sistemas digitales pueden adoptar solamente uno de dos estados posibles, que se nombran con los valores 0 y 1. En el caso del pulsador, un valor de 0 indica que el flujo de corriente está interrumpido, mientras que un valor de 1 indica que pasa corriente.



El **pulsador** fischertechnik se ha concebido como conmutador, por lo que está equipado con 3 contactos. Al oprimir el botón rojo, el pulsador conecta físicamente los contactos 1 y 3. Al mismo tiempo se interrumpe el contacto entre el 1 y el 2, que en estado de reposo estaban tocándose. De este modo puede averiguarse ante cuál de las dos posibilidades de salida estamos:
Cerrado en estado de reposo (contactos 1 y 2 juntos)
Abierto en estado de reposo (contactos 1 y 3 juntos).





El **fototransistor** puede utilizarse indistintamente como sensor analógico o digital. En el primer caso su función es detectar transiciones claras de luz a oscuridad, como puede ser, por ejemplo, una línea que se haya trazado. También puede utilizarse, sin embargo, para discriminar grados de luminosidad, puesto que el fototransistor también puede funcionar en modo analógico. Los valores analógicos pueden modificarse a voluntad entre los dos valores extremos. Para que estas magnitudes puedan procesarse informáticamente, tienen que transformarse en sus correspondientes valores numéricos.

El fototransistor constituye, por otra parte, un componente semiconductor cuyas propiedades eléctricas dependen de la intensidad de la luz. De todos son conocidas las células solares, que transforman la luz

del sol en electricidad. Pues bien: el fototransistor puede entenderse como la combinación de una célula solar en miniatura y un transistor. Los impulsos lumínicos que impactan en el fototransistor (fotones) generan una corriente eléctrica muy pequeña que el transistor se encarga de reforzar.

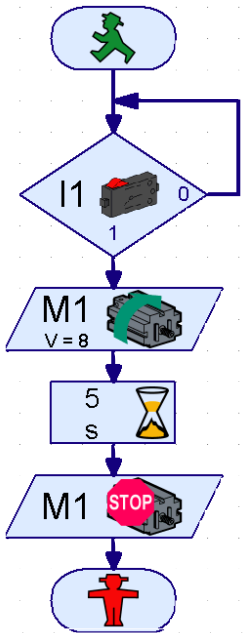
Atención:
Al conectar el fototransistor es importante localizar perfectamente el polo correcto: rojo = positivo.
Tensión soportada: 30V máx.

ROBO Interface

■ La ROBO Interface permite la conexión y la utilización de distintos sensores y actuadores. Además de con sus 8 entradas digitales, la ROBO Interface cuenta con varias entradas analógicas. De este modo, una resistencia medida en las entradas AX y AY que oscile entre 0 y 5,5kΩ se transforma en un valor numérico de entre 0 y 1024. Así se obtienen los valores de medición de un sensor de luminosidad, como puede ser un fototransistor, dispuestos para ser procesados de la manera que queramos. En las entradas analógicas A1 y A2 pueden medirse tensiones de entre 0 y 10VDC.

La función más importante de la interfase es el procesamiento lógico de las magnitudes de entrada. Para poder desempeñarla, la interfase necesita una aplicación informática. El programa decide de qué manera se derivarán de los datos de entrada las señales de sensores, los correspondientes datos de salida, las señales de motor etc. Con la ROBO Interface disponemos de suficiente capacidad de computación para diseñar incluso programas complicados.

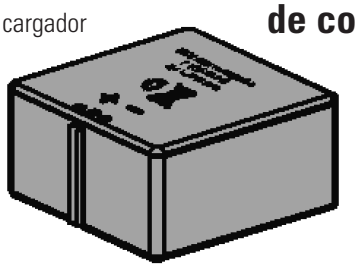
Software ROBO Pro



■ Para que podamos elaborar los programas que necesita la interfase de la manera más efectiva posible, tenemos a nuestra disposición una plataforma gráfica de programación. Por plataforma de programación se entiende un software que nos permite crear nuestro programa con gran comodidad. Se trata de una plataforma gráfica en la que podemos programar con ayuda de símbolos visuales. En realidad, el computador interno de la ROBO Interface solamente es capaz de ejecutar las órdenes que forman parte de su juego de instrucciones. Este juego de instrucciones es, en esencia, un conjunto de sencillas estructuras de control cuyas aplicaciones resultan extraordinariamente complicadas para los que se inician en el tema. Por esta razón, la aplicación informática ROBO Pro ofrece un conjunto de elementos gráficos que son traducidos inmediatamente a un lenguaje que la interfase pueda procesar.

Alimentación de corriente

■ Lo único que nos falta para completar el ROBO Mobile Set es el Accu Set, Art. Nº 34969. Contiene un acumulador que funciona como alimentación portátil para los modelos de robot que utilizaremos y un cargador especial para utilizar con el acumulador.



Lo mejor es que conectemos el acumulador al cargador ya mismo, para que esté cargado cuando más tarde queramos empezar el experimento.

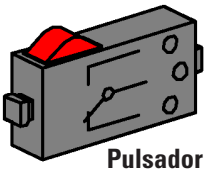
Pasos de la experimentación

■ Para entrar en el fascinante mundo de los robots móviles vamos a proceder paso a paso. Empezaremos con un sencillo montaje de prueba para probar las funciones básicas de la interfase y el conjunto de sensores. A continuación construiremos unos modelos sencillos que desempeñan tareas concretas y ordenadas en función de su dificultad, y pasaremos después a sistemas de creciente complicación.
Si en cualquier momento comprobamos que crear nuestros propios programas resulta demasiado complicado y lleva demasiado tiempo, siempre podemos limitarnos a cargar los programas de ejemplo que se adjuntan y manejar los robots con ellos.
Para evitar desesperarnos con los fallos que se puedan presentar en el proceso, al final de este manual hay un capítulo dedicado a la búsqueda de errores.

Es muy importante proceder con gran cuidado y meticulosidad durante la construcción y puesta en servicio de los robots. Al conectar los componentes eléctricos debemos seguir las instrucciones al pie de la letra y comprobar que todo está correcto no sólo una, sino preferiblemente dos o incluso tres veces. En las construcciones mecánicas, y también con nuestras propias creaciones, siempre pondremos especial cuidado en que los elementos trabajen con suavidad y nos aseguraremos de que se acoplen y funcionen con la holgura justa. Queda en manos de nuestra creatividad el escribir nuestros propios programas y con ello definir nuevos comportamientos para los robots. El único límite serán las capacidades de memoria y procesamiento del hardware. Los siguientes ejemplos ofrecen algunas sugerencias.

Primeros pasos

■ Tras las anteriores consideraciones teóricas, queremos empezar ya a realizar nuestros propios experimentos. Seguramente habrá quien desee empezar a hacer cosas enseguida, quizás incluso con el robot andador grande. Es una opción perfectamente posible: de hecho, si se observan con esmero las instrucciones de montaje, el modelo quedará perfectamente construido incluso tratándose del primer intento.
Pero qué podemos hacer si no funciona? En ese caso habrá que emprender una búsqueda sistemática de la causa del fallo. Eso sí: antes de ponernos a trabajar, debemos probar la interacción entre el ordenador y la interfase.

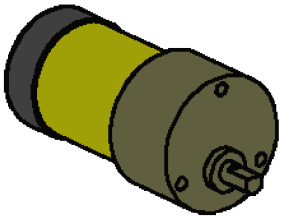


Pulsador

En los capítulos 1 y 2 del manual del software ROBO Pro se explica cómo instalar el software de control en el PC y cómo conectar la interfase. Con ayuda de la prueba de interfase comprobamos los distintos sensores y actuadores.

Pulsador

Podemos, por ejemplo, conectar un pulsador a la entrada digital I1 y observar cómo varía el estado de la entrada al accionar el pulsador.



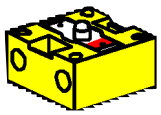
Motor eléctrico

Motor eléctrico

Comprobamos las salidas conectando un motor a una de las de motor, por ejemplo M1. Con el botón izquierdo del ratón podemos establecer el sentido de giro del motor; el regulador deslizante, por su parte, nos permite determinar la velocidad de giro.

Fototransistor

Si también queremos probar la entrada analógica AX, podemos utilizar un fototransistor como sensor analógico.

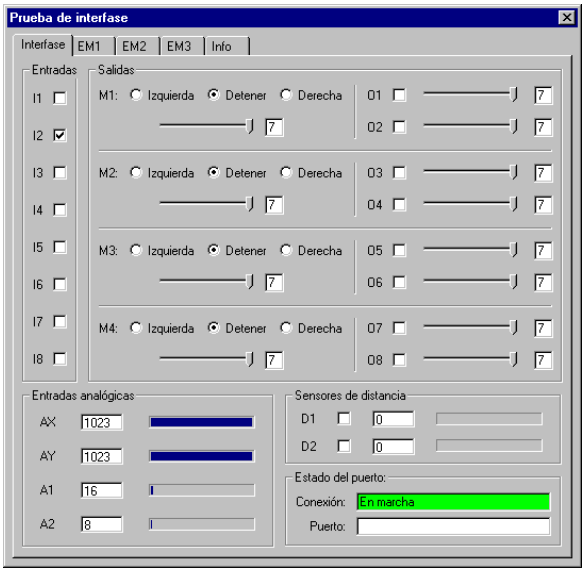


Fototransistor

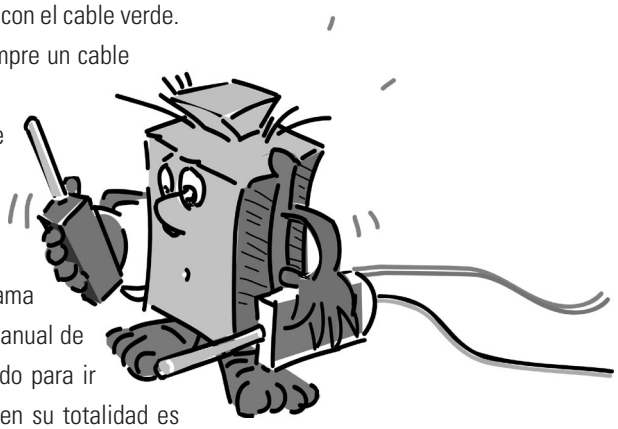
Mientras que en el caso del motor y del pulsador la polaridad de los contactos no desempeña ningún papel (en el peor de los casos el motor giraría en sentido inverso), para que el fototransistor funcione correctamente es imprescindible que esté conectado como es debido.

El contacto del transistor de la marca roja va con un conector también rojo; el otro, con un conector verde. Los dos conectores verdes van a la hembrilla de la entrada AX, situada cerca del borde de la interfase; el segundo conector rojo va en la hembrilla de AX que queda más al centro. (Atención: si se conecta el fototransistor a una entrada digital I1-I8, el conector rojo va en la hembrilla que queda más cerca del borde de la carcasa).

Ahora podemos modificar la intensidad lumínica detectada por el fototransistor con ayuda de una linterna y con ello hacer que se mueva la rayita azul de AX. Si el indicador no se moviese hasta el máximo, deberemos revisar otra vez los contactos del fototransistor. Si por el contrario el indicador sigue estando a cero a pesar de la linterna, el problema puede ser que la luminosidad de la habitación (la claridad ambiental) sea demasiado alta. El indicador empezará a oscilar si tapamos el fototransistor.



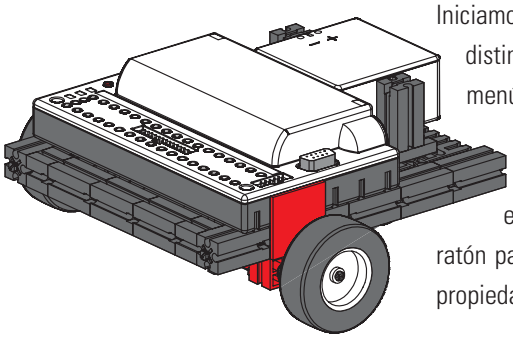
Recordemos brevemente el código cromático de los conectores: ponemos especialísimo cuidado durante el montaje en conectar siempre un conector rojo con el cable rojo y un conector verde con el cable verde. Si al montar un circuito hemos proporcionado la polaridad correcta, tomaremos siempre un cable rojo para el polo positivo y un cable verde para el negativo. Tanta insistencia puede parecer un poco exagerada, pero a la hora de realizar una búsqueda sistemática de errores contar con una clasificación cromática meridianamente clara constituye una ventaja incuestionable.



Queremos dar nuestros primeros pasos en el mundo de la robótica con un programa sencillo. El programa de control de puerta de garaje explicado en el capítulo 3 del manual de Robo PRO no tiene nada que ver con los robots móviles, pero resulta muy apropiado para ir familiarizándonos con el software ROBO Pro. Para poder comprender el programa en su totalidad es suficiente conectar a la interfase el motor y tres pulsadores del kit de construcción ROBO Mobile Set. Todo lo demás viene detalladamente explicado en el manual del software.

El primer robot sencillo

■ Una vez llevada a cabo la prueba de interfase y probado el programa de control de puerta de garaje, procedemos por fin a poner en marcha nuestro primer robot. Montamos el modelo "Robot sencillo" con los dos motores conforme a las instrucciones de construcción. El proceso resulta extremadamente simple y rápido, ya que con toda la intención este modelo se ha concebido únicamente con los elementos imprescindibles para el movimiento de un robot. Conectamos los motores a las salidas M1 y M2.



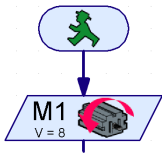
Iniciamos el software ROBO Pro y creamos un programa nuevo en (ARCHIVO - NUEVO). ROBO Pro ofrece distintos niveles de dificultad en los que podemos operar. El nivel se configura en el elemento de menú NIVEL. Por el momento será suficiente con escoger un nivel 1.

Aparecerá una superficie de trabajo vacía y, en el margen derecho, la ventana de elementos en la que iremos seleccionando los distintos elementos de programa con el botón derecho del ratón para colocarlos sobre la superficie de trabajo. Con el botón derecho del ratón modificamos las propiedades.

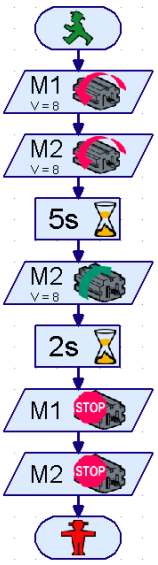
Tarea 1 (nivel 1):
Nuestro "Robot sencillo" debe avanzar en línea recta durante 5 segundos, a continuación girar sobre sí mismo durante 2 segundos y finalmente detenerse.

Consejos:

El primer robot lo programaremos juntos paso a paso:



- Empezamos con el hombrecillo verde que recuerda al de un semáforo. Simboliza el inicio del programa.
- A continuación tomamos el símbolo de motor de la ventana de elementos y lo situamos debajo del elemento de inicio; se dibujará automáticamente una línea de unión entre ambos. En la ventana de propiedades marcamos la salida de motor como "M1" y el sentido de giro como "izquierda", y confirmamos la selección con OK.



- Debajo de este símbolo colocamos de la misma manera otro símbolo de motor, de modo que quede conectado el motor 2.
- Para indicar un margen de espera determinado, utilizamos el elemento tiempo de espera, que colocamos debajo del segundo símbolo de motor y que configuramos con un intervalo de 5 segundos.
- A continuación hacemos que el motor M2 gire en el sentido inverso (hacia la derecha), que espere después 2 segundos y para terminar apagamos ambos motores. Nuestro programa concluye con el símbolo de fin, el hombrecillo rojo. La ilustración muestra el programa completo.

Si no estamos seguros de que todo esté correcto, podemos comparar nuestro programa con el de ejemplo que ya viene incluido. Para ello tenemos que guardar antes de nada el programa que hemos creado y el cargar el archivo Robot sencillo 1.rpp que se halla en la carpeta de ejemplos de ROBO Pro (la ubicación por defecto es C:\Archivos de programa\ROBO Pro\Programmas de ejemplo\ROBO Mobile Set). Si todo está en orden, el programa se traslada a la interfase mediante la funcionalidad de descarga. Tras oprimir el botón Descarga se abre una ventana de diálogo. En ella indicamos que el programa debe almacenarse en la memoria FLASH 1 y que debe iniciarse al terminar la descarga.

Inmediatamente después de la descarga, nuestro modelo empezará a avanzar, girará durante un momento y acabará por detenerse. Si queremos iniciar el programa de nuevo, oprimimos brevemente el botón Prog de la interfase. El LED Prog1 volverá a parpadear durante toda la duración del programa y al final se iluminará permanentemente. El programa continuará almacenado en la memoria FLASH de la interfase incluso después de interrumpida la alimentación de corriente. Vamos a comprobarlo desconectando el acumulador. Tras sacar el conector, volvemos a introducirlo y seleccionamos el programa almacenado apretando el botón Prog hasta que el LED Prog1 se ilumine. Oprimimos de nuevo el botón e iniciamos así el programa.

Hasta ahora el robot no ha hecho demasiado, verdad? Es cierto: por eso queremos ampliar un poco la tarea.

Tarea 2 (nivel 1):

Para que nuestro robot no se detenga al pasar los 7 segundos de la primera tarea, queremos ahora enseñarse un baile.

- **Haz que se mueva hacia delante, a la izquierda, a la derecha y hacia atrás durante intervalos diferentes y a distintas velocidades,**
- **repitiéndose la secuencia hasta que se finalice el programa apretando el botón Prog de la interfase.**

Consejos:

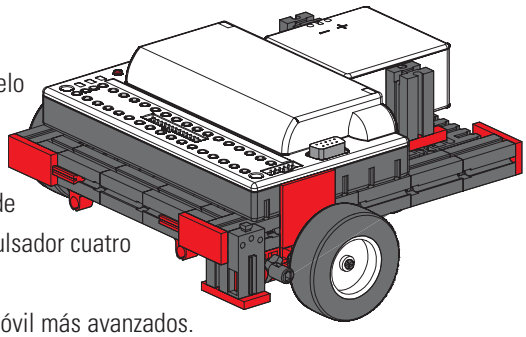
- Para que el robot se mueva en cada momento hacia el lado que desees, simplemente invierte la polaridad cuantas veces sea necesario.
- La velocidad de los motores puedes modificarla en la ventana de propiedades de cada símbolo de motor, ajustándola de 1 a 8. Si los motores M1 y M2 giran a diferentes velocidades pero en el mismo sentido, el robot describirá una curva.
- Para que el programa se repita continuamente, dibuja una línea que conecte la salida del último elemento de programa con la línea que conduce al primer elemento.
- Encontrarás un ejemplo ya terminado de este programa en el archivo Robot sencillo 2.rpp.

Felicidades: acabas de construir y programar tu primer robot! El robot no es demasiado inteligente, porque todavía no detecta los obstáculos y se cae de la mesa si no lo vigilas. Pero eso ya lo cambiaremos en el transcurso de posteriores experimentos.

Robots inteligentes sobre ruedas

■ Para que los robots sean capaces de reconocer su entorno, precisamos utilizar sensores. Las siguientes sugerencias de modelos presentan varias variaciones de robots móviles en los que probaremos las funcionalidades de diferentes sensores. Todo depende de cómo combinemos tanto las condiciones internas del robot (por ejemplo, medición del trayecto mediante ruedas de impulsos) como las señales del exterior (por ejemplo, detección de luz o seguimiento de rastros). Para cada modelo se establecerán, por tanto, tareas específicas. Todas ellas deben tomarse como sugerencias que te servirán para ir entrando en materia. Los programas que ejecutan cada tarea específica se hallan en la carpeta de ROBO Pro, en \Programas de ejemplo\ROBO Mobile Set\. Con el tiempo se te ocurrirán nuevas tareas para los modelos; cuando acabes de repasar los siguientes ejemplos, sin duda se te habrán ocurrido ya muchas ideas.

Modelo básico

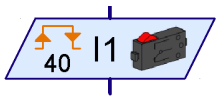


■ A diferencia de nuestro robot sencillo inicial, el modelo básico posee una construcción más estable y robusta. Contiene además 2 sensores de trayecto, consistentes cada uno en un pulsador y una rueda de impulsos. La rueda de impulsos está conectada con el eje del motor y acciona un pulsador cuatro veces por cada revolución del motor.
Este modelo sirve como base para otros modelos de robot móvil más avanzados.

Construye el modelo básico conforme a las instrucciones, poniendo especial esmero en cada uno de los pasos del montaje. Cuando el mecanismo esté completo, comprueba que los motores funcionan sin trabas conectando cada uno de ellos directamente a los acumuladores (sin interfase) durante unos instantes.

Tarea 1 (nivel 1):

- **Programa la interfase para que el modelo avance en línea recta 40 impulsos. Para medir los impulsos, utiliza el pulsador contador de la entrada I1.**
- **Mide el trayecto que recorre el modelo y calcula cuál es la distancia por impulso.**
- **Repite el proceso 3 veces y apunta en la tabla cuánto oscilan los valores.**



Consejos:

- Empieza por encender ambos motores (sentido de giro a la izquierda).
- Para contar los impulsos de I1, utiliza el elemento de programa **contador de impulsos**.
- Cuenta los dos flancos posibles (0-1 al apretar y 1-0 al soltar el pulsador). Puedes seleccionar uno u otro en la ventana de propiedades, en la sección de tipo de impulso. Al hacerlo aumentas la exactitud de la medición del trayecto.
- Después vuelve a apagar los motores y finaliza el programa.
- Encontrarás el programa completo en el archivo Modelo básico 1.rpp.

Resultado:

	Número de impulsos	Distancia recorrida	Distancia/Impulso
Intento 1	**40**		
Intento 2	**40**		
Intento 3	**40**		

Grosso modo puede calcularse que el modelo recorre una distancia de un centímetro por cada impulso.

Al mismo tiempo esto te permite saber qué sentido de giro debes establecer en cada uno de los motores para que el modelo avance hacia uno u otro lado. Apunta estos hallazgos en la tabla siguiente para no tener que deducirlo todo de nuevo cada vez que quieras modificar el sentido de la marcha. Si cableas el modelo exactamente como se ilustra en las instrucciones de de montaje, el giro a la derecha de cada uno de los motores hará que la rueda se mueva hacia adelante. Así están programados todos los motores en los programas de ejemplo.

Completa la tabla:

Sentido de la marcha del modelo	Sentido de giro M1	Sentido de giro M2
Avance	**Izquierda**	**Izquierda**
Retroceso		
Izquierda		
Derecha		
Detención		

Para no tener que colocar dos símbolos de motor en la pantalla cada vez que queremos hacer un cambio en el movimiento, podemos crear un subprograma que se ocupe de esta tarea para cada sentido de la marcha. Esto simplifica enormemente la programación. Las instrucciones para crear un subprograma se hallan en el capítulo 4 del manual del software ROBO Pro. Cuando hayas leído este capítulo completo, estarás en condiciones de atreverte con la siguiente tarea. Pasa la aplicación ROBO Pro al **nivel 2**.

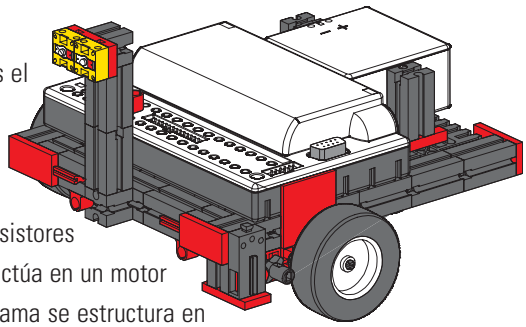
Tarea 2 (nivel 2):

- **Genera un subprograma para cada sentido de la marcha.**
- **Programa el modelo para que recorra un cuadrado de un metro de lado.**
- **Cuál es la exactitud de repetición?**

Consejos:

- Crea primero un subprograma de **avance**. El resto de los subprogramas puedes fabricarlos a partir de una copia del subprograma de avance, simplemente cambiando el sentido de giro de los motores.
- Tanto para el giro hacia la izquierda como hacia la derecha, utiliza una velocidad reducida. Esto aumenta la precisión.
- Para contar los impulsos utiliza bien el elemento **contador de impulsos** o bien el pulsador de la entrada I1.
- Primero carga el programa en la RAM para ir probándolo hasta que hayas averiguado cuántos impulsos necesitas para que el robot ejecute un giro de 90º. En primer lugar, es más rápido cargar el programa en la memoria RAM que en la FLASH; en segundo lugar, la memoria FLASH tiene una vida limitada de aproximadamente 100.000 descargas.
- El programa ya completo se llama Modelo básico 2.rpp.

Robot con detección lumínica



■ Una vez investigado en profundidad el modelo básico, es el momento de hacer que el robot aprenda a reaccionar ante señales ambientales. Igual que la polilla del experimento mental propuesto en el capítulo primero, el robot deberá detectar y seguir una fuente de luz. El kit contiene 2 fototransistores que vamos a utilizar como detectores de luz. Cada sensor actúa en un motor para posibilitar el seguimiento de la fuente de luz. El programa se estructura en dos partes. La primera contiene la búsqueda de la fuente lumínica; en la segunda se efectúa el seguimiento o persecución de la fuente. Dentro del programa se emplean varios subprogramas. Inmediatamente tras la puesta en marcha se activa el subprograma de detección lumínica, que no se abandona hasta que se haya localizado la fuente de luz. El programa principal intenta poner al robot rumbo a la fuente de luz. Cada vez que la dirección del robot se desvía perceptiblemente de la línea ideal, unos de los sensores deja de estar bajo la fuente de luz; si eso ocurre, el robot corrige la dirección de modo que ambos sensores puedan detectar de nuevo la fuente lumínica.

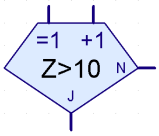
Construye ahora el modelo de detección lumínica tal y como se describe en las instrucciones de montaje.

Tarea 1 (nivel 2):

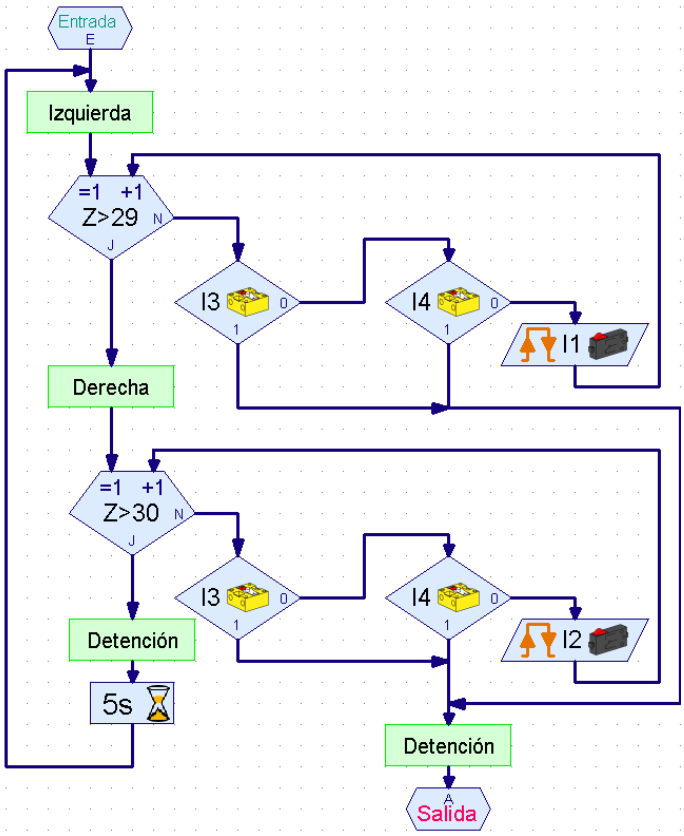
- **Empieza programando la función de detección lumínica. En ella, el robot tiene que definir un giro lento de al menos 360º. Si durante la búsqueda encuentra una luz, el robot se detiene. Si no es así, vuelve a girar otros 360º en sentido inverso. Si sigue sin encontrar ninguna luz, el robot espera 5 segundos y vuelve a emprender la búsqueda.**
- **Si la búsqueda da frutos, el modelo tendrá que poner rumbo hacia la fuente de luz. Si la fuente se mueve a izquierda o derecha, el robot debe seguir el movimiento de la luz. Si pierde el contacto, el programa debe volver al principio y emprender una nueva secuencia de detección lumínica. Comprueba si puedes atraer al robot con una linterna y hacer que atraviese un recorrido de obstáculos.**

Consejos:

- Para controlar los distintos sentidos de la marcha, utiliza los subprogramas que ya habías programado para el modelo básico. Abriendo el programa Modelo básico 2.rpp encontrarás en la **ventana de grupos de elementos** de ROBO Pro, en **Programas cargados**, el programa Modelo básico 2 y dentro de él los subprogramas que contiene. No tienes más que insertar estos subprogramas en el nuevo programa.
- el subprograma de detección lumínica, utiliza el elemento **bucle contador**. (La descripción de este elemento se encuentra en el manual de ROBO Pro).
- En el bucle entre los contactos N y +1 consulta a los fototransistores y cuenta un impulso en el pulsador I1. El bucle debe repetirse tantas veces como sea necesario hasta que el robot detecte la luz o haya rotado 360º. Para saber cuántas veces se tiene que repetir un bucle para cada rotación completa, simplemente haz la prueba; después ajusta el valor Z del elemento bucle contador en función de lo que hayas averiguado.
- Inmediatamente después, repite el mismo método para programar un segundo bucle, que servirá para realizar la misma búsqueda pero en sentido de giro inverso.
- Si el robot encuentra luz, se detiene y abandona el subprograma.
- Este es el subprograma completo de detección lumínica:



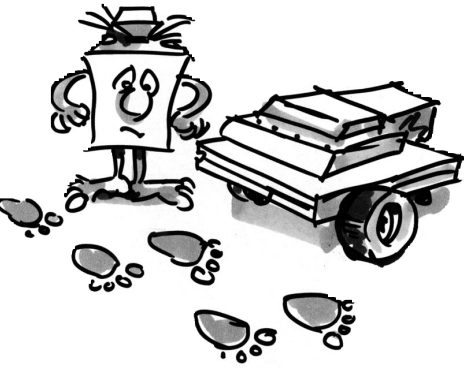
bucle contador



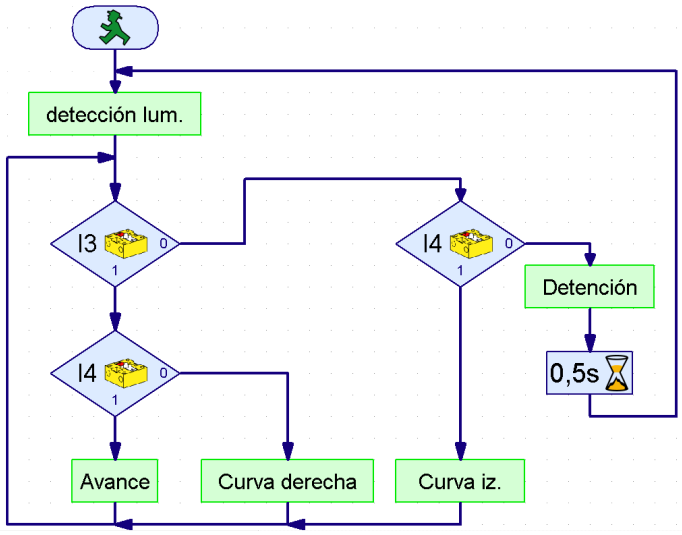
- En el programa principal, vuelve a introducir la consulta a los fototransistores y controla los motores en función de cuál de ellos haya reconocido la luz:

Luz en I3 e I4	**Avance**
Luz sólo en I3	**Curva a la derecha**
Luz sólo en I4	**Curva a la izquierda**
No se detecta luz	**Parar, volver al subprograma de detección lumínica**

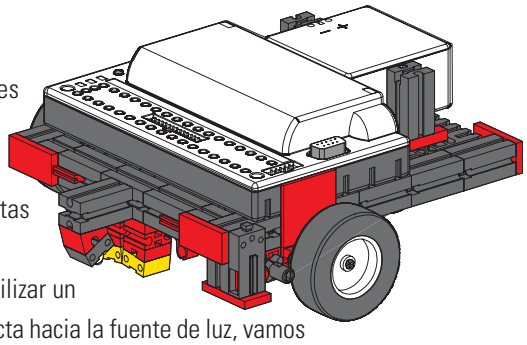
- La curva a la derecha o a la izquierda se consigue aplicando diferentes velocidades a M1 y M2 pero manteniendo el mismo sentido de giro. De este modo se obtiene un estilo de movimiento muy armónico.



- El programa principal tendrá la siguiente apariencia:
- Encontrarás el programa completo en Detección lumínica.rpp.
- Como fuente de luz utiliza una linterna. Procura que el rayo de luz no quede enfocado en un punto demasiado pequeño, para que puedan recibir luminosidad los dos fotosensores. Ten en cuenta que en habitaciones muy iluminadas la luz de la linterna quedará anulada por la de otras fuentes (por ejemplo, la luz del sol que entre por una ventana amplia). En esas circunstancias, el robot no hará caso de la linterna y se encaminará a la luz más intensa.



Robot con seguimiento de rastros



■ La localización y el seguimiento son propiedades esenciales de las que gozan los seres inteligentes. Con la función de detección lumínica hemos construido y programado un robot que reacciona a señales directas emitidas por su objetivo.
Con la funcionalidad de seguimiento de rastros vamos a utilizar un principio de búsqueda diferente. En lugar de la marcha directa hacia la fuente de luz, vamos a marcar una línea negra que el robot deberá seguir. Esta tarea resulta relativamente fácil de resolver gracias a los fototransistores. Se mide la luz reflejada de la línea trazada y los motores realizan la corrección oportuna. Para que el sistema funcione con precisión, la línea es iluminada por la lámpara. Debemos tener cuidado de que los fotosensores no resulten deslumbrados por luces parásitas de la bombilla a causa de una colocación inconveniente. Un excelente sistema para evitarlo pasa por el enfoque del haz de luz que realiza la lente óptica de la bombilla.

Construye el modelo de seguimiento de rastros conforme a las instrucciones de montaje.

Tarea 1:

- **Para empezar, escribe un subprograma que busque el rastro. El robot tendrá que girar una vez sobre sí mismo.**
- **Si no encuentra ningún rastro, queremos que avance un poco en línea recta y vuelva a realizar la búsqueda. El reconocimiento del rastro se basa en la consulta a los fototransistores.**
- **Si el robot descubre un rastro, debe seguirlo.**
- **Si llega al fin del rastro o lo pierde (si se presenta, por ejemplo, un cambio brusco de dirección), él robot deberá emprender de nuevo la búsqueda.**

Consejos:

- Una vez encendida la lámpara hay que esperar un momento (aproximadamente un segundo) antes de poder consultar a los fototransistores. De otro modo el fototransistor detectará oscuridad y con ella la presencia de un rastro que no existe, ya que se le realiza la consulta antes de que la bombilla haya tenido tiempo de encenderse por completo.
- Usaremos como rastro un trozo de cinta aislante negra de aproximadamente 20 mm de ancho; también podemos dibujar con un rotulador un rastro negro del mismo ancho sobre una hoja de papel de color blanco. Las curvas no deben ser demasiado cerradas, para evitar que el robot pierda constantemente el rastro.
- Lleva a cabo antes de nada la prueba de interfase para comprobar si los fototransistores reconocen correctamente el rastro. No olvides encender la bombilla.
- Ajusta la lámpara de forma que los dos fototransistores devuelvan el valor 1 al encontrarse sobre una superficie luminosa, incluso estando en marcha los motores M1 y M2. Si el acumulador no se halla totalmente cargado, al encender los motores la luz de la bombilla perderá algo de intensidad. Si no está bien ajustada, puede ocurrir que uno de los fototransistores comunique que detecta oscuridad aunque no haya encontrado ningún rastro.
- El seguimiento de rastros funciona de manera semejante a la detección lumínica. Lo único que debes hacer es adaptar la búsqueda de luz de modo que, si el modelo no detecta nada tras dar una vuelta sobre sí mismo, avance un poco hacia delante antes de volver a rotar.
- Ten en cuenta que en la funcionalidad de seguimiento de rastros el modelo tiene que avanzar en línea recta cuando ambos fototransistores devuelven el valor "oscuridad" (= 0).
- Encontrarás el programa completo en el archivo Seguimiento de rastros.rpp.

Tarea 2:

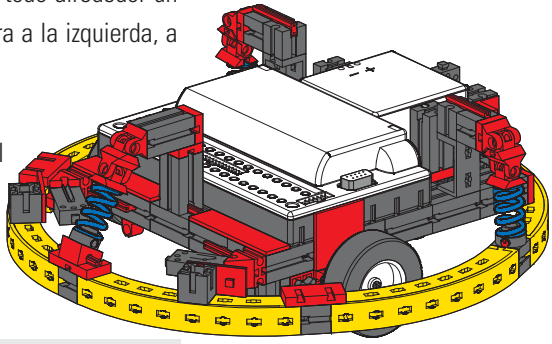
- **Fabrica un rastro con varias curvas cerradas. Qué radio describe el modelo?**
- **Experimenta la introducción de correcciones con diferentes velocidades de M1 y M2. Qué combinación da mejor resultado?**
- **Crea un rastro en forma de circunferencia. Trata de optimizar las velocidades de modo que el robot complete el círculo lo más rápidamente posible. Estas tareas son perfectas para organizar competiciones con varios robots.**

Robot con detección de obstáculos

■ Todos los robots que hemos construido hasta ahora son capaces de recorrer un trayecto determinado o de seguir una fuente de luz o un rastro. Pero qué pasa si se encuentran con un obstáculo en su camino? Una de dos: o bien apartamos el obstáculo a un lado o el robot se queda empujando absurdamente contra él hasta que se le agote el acumulador. Mucho más inteligente sería, sin duda, que el robot detectase el obstáculo y lo evitase en consecuencia. Para ello se le coloca al robot todo alrededor un parachoques flexible con tres pulsadores, que le permiten determinar si se encuentra a la izquierda, a la derecha o detrás del obstáculo.

La reacción que entonces adopte es cuestión ya de la programación que le demos.

Empieza montando el modelo "Robot con detección de obstáculos". Para medir el trayecto necesitamos sólo un pulsador (I1), por eso podemos extraer el pulsador I2 del modelo básico y utilizarlo para la función de detección de obstáculos.



Tarea 1 (nivel 2):

- **Primeramente el robot debe avanzar hacia adelante. Si se encuentra con un obstáculo a la izquierda (E4), debe retroceder un poco y evitarlo por la derecha.**
- **Si se encuentra con un obstáculo a la derecha (E3), debe retroceder un poco y evitarlo por la izquierda.**

Consejos:

- La detección de obstáculos durante la marcha atrás no será tratada por el momento.
- En el programa principal se consulta a los pulsadores. En función de cuál de los pulsadores sea activado, el modelo evitará el obstáculo por la izquierda o por la derecha. Cada caso está regulado por su correspondiente subprograma.
- Al contar el número de impulsos registrados en el giro debe distinguirse la izquierda de la derecha (por ejemplo, 3 impulsos a la derecha y 5 a la izquierda), porque si no, puede ocurrir que el modelo vaya a parar a un rincón y no encuentre nunca más la forma de salir de él.
- El programa ya terminado se llama Obstáculos 1.rpp.

Hay dos cosas que el robot detector de obstáculos todavía no puede hacer: detectar obstáculos mientras se mueve hacia atrás y darse cuenta de que tiene un obstáculo justo delante de él. No las hace todavía, pero puede hacerlas. Si durante la marcha atrás se presiona el I5, quiere decir que hay un obstáculo detrás del modelo. Si durante la marcha adelante se activan a la vez el I3 y el I4, quiere decir que hay un obstáculo justo delante del modelo. En este caso el robot podría girar 90º. En conjunto, estas son todas las reacciones con las que puede responder el robot:

Obstáculo	Pulsador	Reacción
derecha	sólo I3	Evitar por la izquierda (giro de aprox. 30º)
izquierda	sólo I4	Evitar por la derecha (aprox. 45º)
delante	I3 e I4	Evitar por la izquierda (aprox. 90º)
detrás	I5	Sólo durante la marcha atrás. Parar, después continuar con maniobra de evasión según lo planeado

Para completar esta tarea con elegancia, te vendrán muy bien una serie de nuevos elementos de programa (como por ejemplo los operadores: Y, O etc.) asociados al nivel 3 de ROBO Pro. En este nivel también se nos da la posibilidad de utilizar flechas naranjas para intercambiar datos entre distintos elementos. Por esta razón, debes empezar cambiando el nivel en el programa. A continuación lo más recomendable es que leas con atención todo el capítulo 5 del manual de ROBO Pro. Una vez lo hayas leído, estarás preparado para emprender la siguiente tarea.

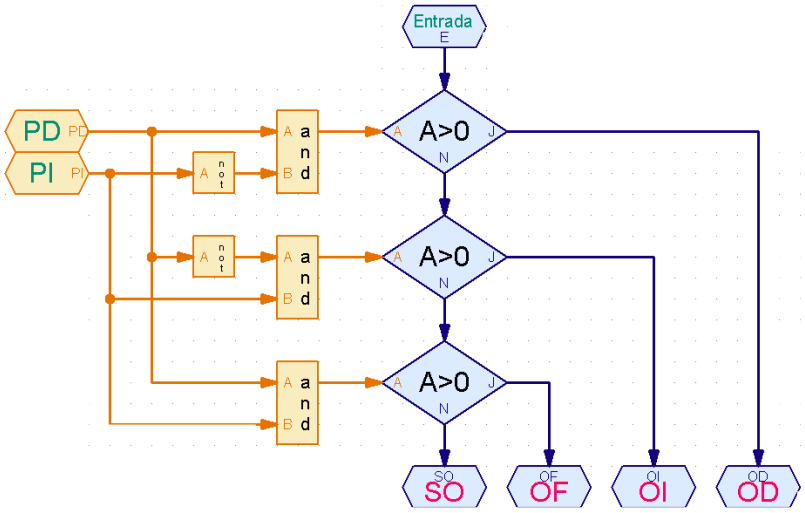
Tarea 2 (nivel 3)

- **Transforma el programa de detección de obstáculos para que el modelo reaccione como se muestra en la tabla anterior.**
- **Utiliza para ello las posibilidades que ofrece el nivel 3 de ROBO Pro.**

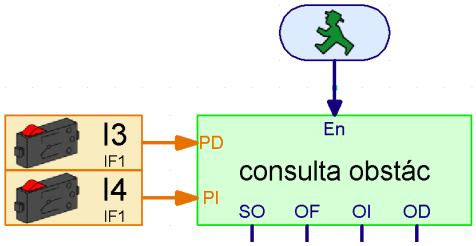
Consejos:

- Con ayuda de los operadores, se consultan en un subprograma de consulta de obstáculos las diferentes combinaciones de activación de pulsadores posibles. Para cada posibilidad el subprograma posee una salida específica.

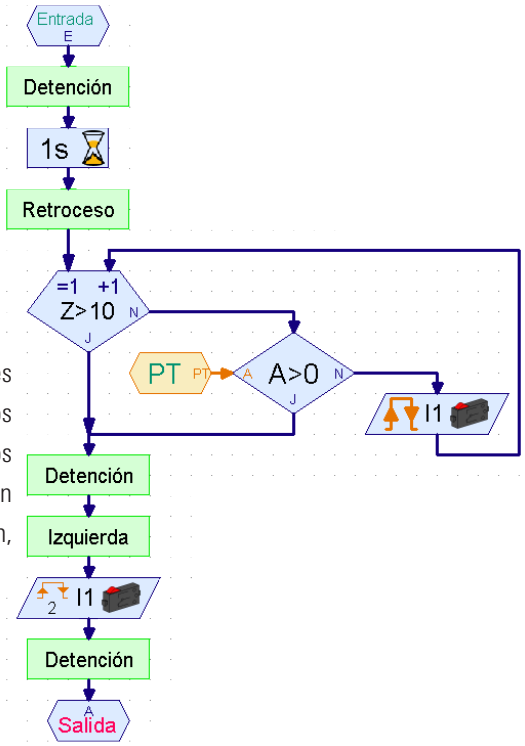
Entrada de datos PD = pulsador derecha
Entrada de datos PI = pulsador izquierda
Salida SO = sin obstáculo
Salida OF = obstáculo al frente
Salida OI = obstáculo a la izquierda
Salida OD = obstáculo a la derecha



- Para poder saber rápidamente qué pulsadores se consultan, coloca los elementos de pulsador de color naranja en el programa principal y conéctalos con el subprograma mediante entradas de datos.



- En los diferentes subprogramas de superación de obstáculos se consulta I5 durante la marcha atrás. El modelo continúa retrocediendo hasta que alcance el número de impulsos fijado o se active I5. I5 se coloca otra vez en el programa principal para que se perciba rápidamente en qué subprogramas se consulta.
- Encontrarás el programa completo en el archivo Obstáculos 2.rpp.

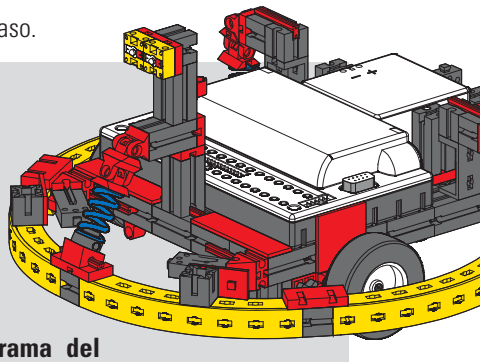


Una ventaja de la técnica de programación utilizada en esta tarea es que te permite ver directamente en el programa principal qué pulsadores se consultan en cada subprograma. Si quieres modificar la entrada, lo haces solamente en un punto y no tienes que dedicarte a buscar en todos los subprogramas dónde puede estar oculto el pulsador que buscas. Además, los operadores permiten crear cadenas lógicas que se comprenden de un vistazo. En principio puede decirse lo mismo de los elementos de bifurcación, pero cuando llegamos a casos de consultas múltiples enseguida se complican las cosas y resulta difícil interpretar lo que se ve.

Robot con detección lumínica y de obstáculos

■ Todavía no hemos explorado ni mucho menos todas las posibilidades que ofrece el ROBO Mobile Set. Vamos a combinar ahora dos funcionalidades que ya hemos visto: la detección lumínica y la detección de obstáculos. Desde el punto de vista científico, el robot está equipado con dos modos de comportamiento. Puesto que los dos modelos de conducta no pueden estar activos simultáneamente, han de recibir diferentes grados de prioridad. El robot se encuentra normalmente en el modo de detección lumínica. Si detecta un obstáculo, es decir, un peligro para él, pasa automáticamente al modo que le permite evitarlo. Si todo está en orden, el robot puede entonces continuar su búsqueda de la fuente de luz.

Cuando los programadores profesionales se enfrentan a una tarea tan complicada como esta, no se lanzan a programar como si tal cosa, sino que proceden a hacer uso de una u otra estrategia de programación. Uno de estos métodos es el llamado enfoque "top-down" (o "de arriba a abajo"). En esta manera de proceder se define la totalidad del sistema desde los niveles superiores hacia los inferiores, sin preocuparse al principio de concretar todos y cada uno de los detalles. Este es el método que precisamente vamos a utilizar nosotros en este caso.



Tarea 1 (nivel 3)

Enseña al robot a actuar de la siguiente manera:

- **Busca una fuente de luz.**
- **En cuanto la localices, síguela.**
- **Si te topas con un obstáculo en el camino, evítalo.**
- **A continuación vuelve a buscar una fuente de luz.**

Completa la tarea utilizando los elementos de programa del nivel 3 de ROBO Pro. Resuélvela empleando un enfoque "de arriba a abajo".

Consejos:

Para empezar, estructura la tarea en tres partes:

- Averiguar si el robot ve alguna fuente de luz (subprograma de detección lumínica)
- Preguntar si está chocando con algún obstáculo (subprograma de detección de obstáculos)
- En función de los resultados de esas dos preguntas, decirle al robot lo que tiene que hacer (subprograma de movimiento).

Para generar los subprogramas de detección lumínica y de obstáculos, debemos empezar por preguntarnos cuáles son las diferentes situaciones en las que se puede encontrar el robot. Asígnale a cada una de ellas un valor numérico y haz que se guarde en una variable con ayuda de un elemento de comando. Cada situación generará entonces una reacción, que se ejecutará en el subprograma de movimiento.

Subprograma de detección lumínica:

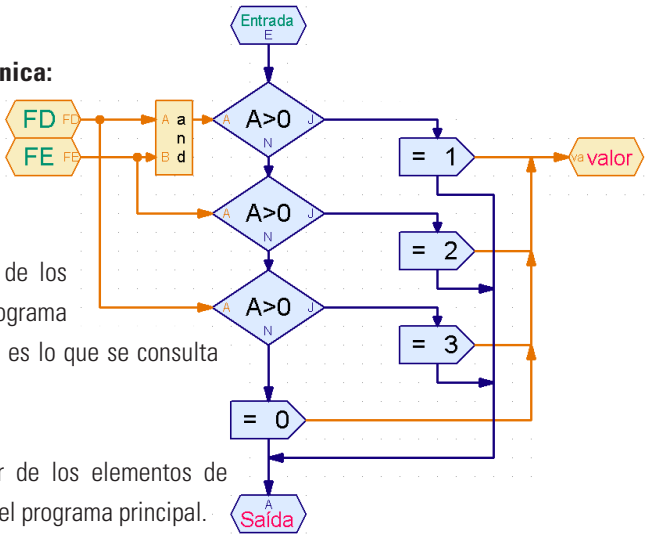
Nº	Situación	Estado de los sensores	Reacción
0	No hay fuente de luz disponible	I6=0; I7=0	Buscar luz
1	Fuente de luz justo ante el robot	I6=1; I7=1	Avanzar en línea recta
2	Fuente de luz a la izquierda del robot	I7=1	Describir curva a la izquierda
3	Fuente de luz a la derecha del robot	I6=1	Describir curva a la derecha

Subprograma de detección de obstáculos:			
Nº	**Situación**	**Estado de los sensores**	**Reacción**
4	**Obstáculo justo ante el robot**	**I3=1; I4=1**	**Evitar virando 90°**
5	**Obstáculo a la derecha del robot**	**I3=1**	**Evitar por la izquierda**
6	**Obstáculo a la izquierda del robot**	**I4=1**	**Evitar por la derecha**

Ahora debes plasmar esta casuística en ROBO Pro en forma de elementos de programa.

Subprograma de detección lumínica:

FD = fototransistor derecho
FI = fototransistor izquierdo



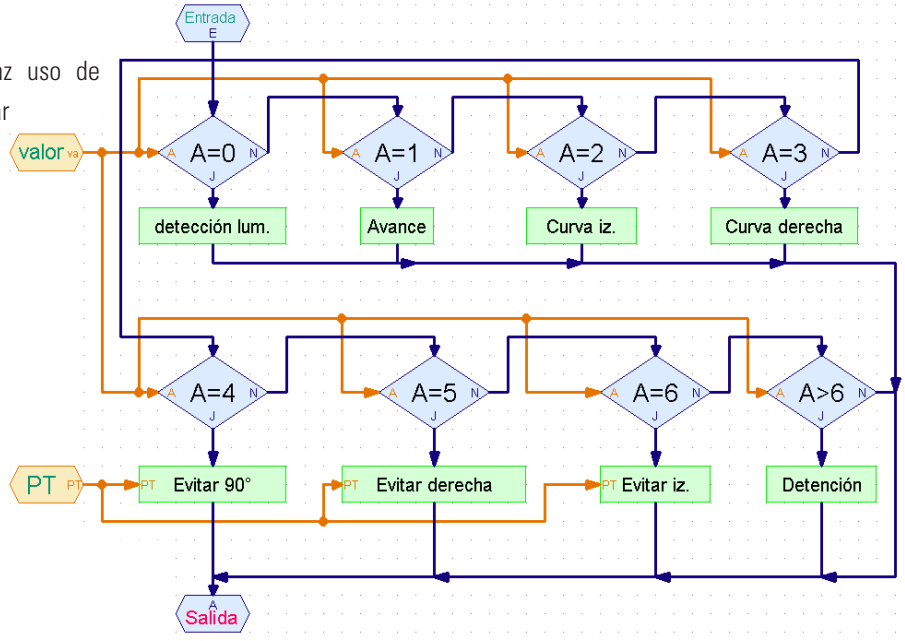
Coloca los elementos de consulta de los fototransistores también en el programa principal, para poder reconocer qué es lo que se consulta en cada momento.

La variable que almacena el valor de los elementos de comando también debe incluirse en el programa principal. Será utilizada en más de un subprograma. Debes conectarla al subprograma mediante una salida de datos.

Elabora el subprograma de detección de obstáculos conforme el mismo principio que el de detección lumínica.

En el subprograma de movimiento haz uso de elementos de bifurcación para consultar cuál es el valor actual de la variable y programa la reacción correspondiente:

PT = pulsador trasero



Como detalle final resta ahora por crear los subprogramas que se usarán en este subprograma.
Pero... un momento! Esa tarea está prácticamente ya terminada: El subprograma de detección lumínica, por ejemplo, podemos copiarlo del programa del modelo dotado de la funcionalidad de detección lumínica. Si ya no recuerdas cómo se hacía, léelo en el capítulo 4 del manual de ROBO Pro.

Pero ten cuidado:
En aquel modelo de detección lumínica, los fototransistores estaban conectados en las entradas I3 e I4, mientras que ahora tienen que ir en I6 e I7. Además, en aquel caso la cuenta de impulsos estaba asociada al pulsador I1 para el giro a la derecha y al pulsador I2 para el giro a la derecha, mientras que ahora sólo tenemos a I1 para esa función, que por otra parte funciona igual de bien. En resumen: aunque puedes copiar el subprograma de detección lumínica, también tendrás que adaptarlo. Como la consulta del pulsador está oculta en el subprograma, es fácil pasarla por alto. Esto se evita colocando las entradas en el programa principal y conectándolas con el subprograma mediante entradas de datos. Pero en el caso de la detección lumínica esto deja de ser una posibilidad.

Los subprogramas que indican al robot cómo evitar los obstáculos ya están hechos, concretamente en el modelo dotado con la funcionalidad de detección de obstáculos. En este caso incluso el pulsador I5, un extra que se consulta en la marcha atrás, se deriva hacia fuera.

Encontrarás el programa completo en el archivo Obstáculos y luz.rpp.
El programa principal parece ya a simple vista extremadamente claro y sencillo. Y sin embargo detrás de los subprogramas se esconde mucho discurrir. Con ayuda del avance paso a paso asociado al enfoque de arriba a abajo, sin embargo, puedes un problema de semejante complejidad.

Una idea: si algún amigo tuyo tiene también un kit de construcción modular ROBO Mobile Set, podéis llevar el experimento con estos robots todavía más lejos. Simplemente tenéis que acoplar una fuente de luz a cada robot para que así los robots se busquen mutuamente.

Robot con detección de bordes

■ Una vez que hemos visto en el anterior ejemplo cómo se procede en la programación de un problema complejo, podemos pasar ya a ocuparnos de otro comportamiento de los robots móviles, un comportamiento de enorme importancia: el robot tiene que aprender a no caerse de la mesa. Si el robot choca contra un obstáculo, lo más probable es que no le ocurra nada malo. Pero si se cae de una mesa, desde una altura de casi un metro desde el suelo, es fácil que quede dañado en algún punto, por muy robustos que sean (como de hecho son) los componentes de fischertechnik. Por esta razón el robot posee unos sensores encargados de detectar la presencia de bordes o discontinuidades en la superficie por la que avanza. Cada detector de bordes consiste en un pulsador que es activado por una rueda giratoria. La rueda no pone resistencia y puede calcarse hacia dentro o salirse hacia fuera. En el momento en que la rueda vaya más allá del borde de la mesa, queda en el aire y sale hacia fuera al no haber ya nada (ningún suelo) que la mantenga calcada hacia dentro; entonces el pulsador deja de estar activado, el programa reconoce que el modelo ha perdido suelo y reacciona en consecuencia. El robot tiene en total 4 detectores de bordes que le permiten reconocer la discontinuidad de la superficie de apoyo tanto en la marcha hacia delante como en el retroceso, en cualquiera de los flancos. Este modelo, en cambio, no tiene pulsador de impulsos para la medición del trayecto. El camino que se recorre es controlado por la duración del encendido de los motores.

Construye, para empezar, el modelo tal y como se describe en las instrucciones de montaje.
Controla especialmente que los detectores de bordes se activan correctamente:

- cuando el modelo llega al borde de una mesa y el pulsador vuelve a presionarse con todo cuidado,
- cuando la rueda vuelve a tomar contacto con la mesa.

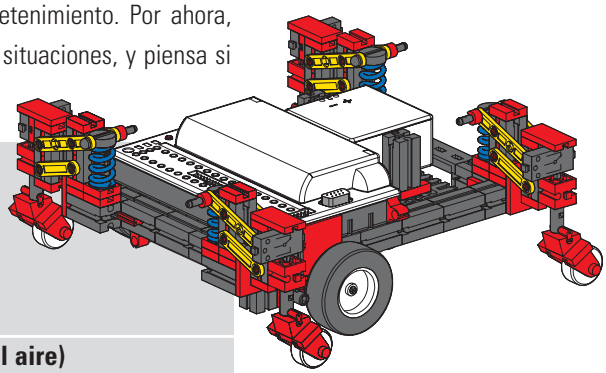
Es posible que tengas que ajustar alguno de los pulsadores un poquito hacia arriba o hacia abajo.

Tarea 1 (nivel 3):

- **Piensa durante unos instantes cómo debe reaccionar el robot cuando llegue a una discontinuidad de la superficie de apoyo.**
- **Si lo piensas bien, llegarás a la conclusión de que existe un buen número de posibilidades de combinación en las que se pueden encontrar los sensores que detectan la pérdida de suelo. Puede que se active uno de los 4 detectores, 2 ó 3 al mismo tiempo o incluso los 4 a la vez.**
- **Cómo debería reaccionar el robot en cada caso?**

Consejos:

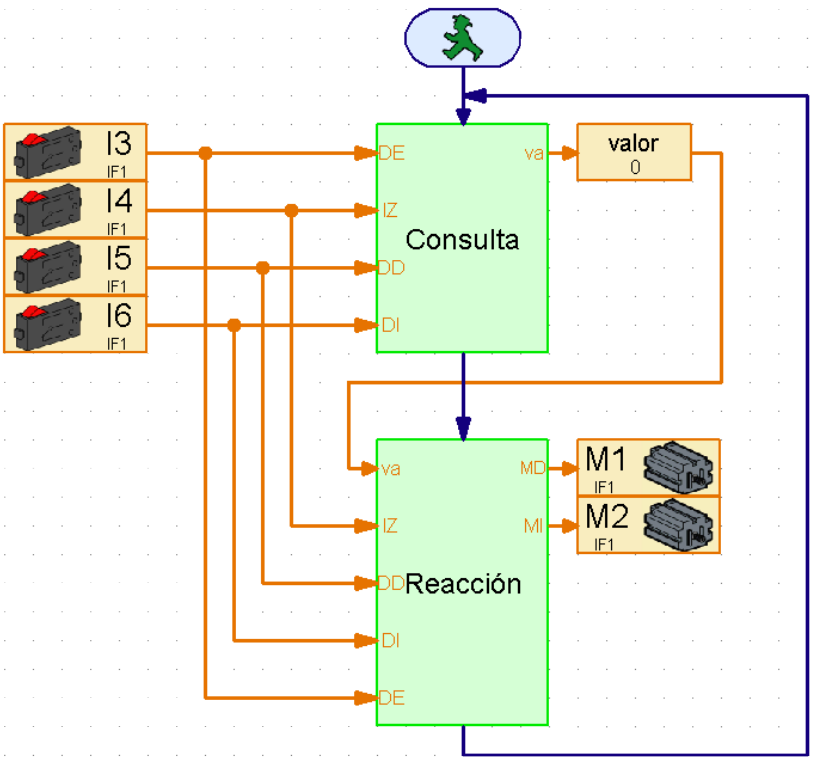
Encontrarás la solución en la tabla que sigue. Los sensores que se encuentran sobre la discontinuidad (pulsador = 0) están marcados con una equis. A cada combinación se le asigna un número. En el programa que crearemos más adelante asignaremos a cada posibilidad el valor numérico correspondiente. Este valor numérico le permite al robot reaccionar a la situación que se registre en cada momento. Más adelante volveremos a tratar esta cuestión con más detenimiento. Por ahora, discurre cómo tiene que encontrarse el robot para que se dé cada una de las situaciones, y piensa si reacciona correctamente.



Nº	delante a la derecha (I3)	delante a la izquierd (I4)	detrás a ala derecha (I5)	detrás a la izquierd (I6)	Reacción
0					Avanzar (no hay sensor en el aire)
1	●	●	●	●	Parar (los 4 sensores están en el aire)
2	●	●	●		Girar un poco a la derecha
3	●	●		●	Girar un poco a la izquierda
4	●		●	●	Girar un poco a la izquierda
5		●	●	●	Girar un poco a la derecha
6	●	●			Primero retroceso, después girar a la derecha
7	●		●		Girar un poco a la izquierda
8	●			●	Girar un poco a la izquierda
9		●	●		Girar un poco a la derecha
10		●		●	Girar un poco a la derecha
11			●	●	Avanzar un poco hacia adelante
12	●				Primero retroceso, después girar a la izquierda
13		●			Primero retroceso, después girar a la derecha
14			●		Avanzar un poco hacia adelante
15				●	Avanzar un poco hacia adelante

Parece bastante complicado, verdad? Pero no hay por qué preocuparse: para este modelo contamos con un programa ya completo que hace uso de todas las ventajas ofrecidas por ROBO Pro. Se llama Bordes.rpp.

Los elementos más importantes se encuentran en el programa principal, para que puedas comprender el conjunto del proceso. Las complejas consultas de los pulsadores están ocultas dentro de los subprogramas. Empecemos con el programa principal:



El proceso comienza con la consulta a los 4 pulsadores. A la izquierda ves qué pulsador se está consultando. Están conectados al subprograma mediante entradas de datos. El subprograma de consultas averigua qué pulsadores están oprimidos y devuelve el valor descrito en la tabla. Este valor se le asigna a la variable del mismo nombre, que puedes observar también en el programa principal. El valor de la variable se transmite al subprograma de reacción, que a continuación controla los dos motores en función de este valor. En el subprograma de reacción se leen igualmente los pulsadores, ya que los sensores de bordes también se consultan mientras el modelo realiza la maniobra evasiva.

Ahora puedes modificar sin más la posición de los pulsadores en la interfase e incluso las salidas de motores, sin tener que repasar punto por punto todos y cada uno de los subprogramas que esconden por alguna parte algún elemento de entrada o un símbolo de motor. Las entradas y las salidas aparecen una sola vez.

Esta técnica de programación es especialmente apropiada cuando queremos usar un subprograma en muchos modelos diferentes y no sabemos exactamente con antelación qué entradas y salidas de la interfase querremos utilizar en cada caso.

Si con la explicación te ha picado la curiosidad, no tienes más que echarle un vistazo al subprograma y tratar de comprenderlo. El principio de programación es semejante al que seguimos en el modelo con detección lumínica y de obstáculos.

Tarea 2 (nivel 3):

Carga el programa en la interfase y haz que el modelo se mueva por encima de una mesa.

- **Reacciona siempre como es debido?**
- **Debería actuar de un modo diferente en determinadas combinaciones de activación de pulsadores?**
- **Si es necesario, optimiza el programa.**

El robot andador

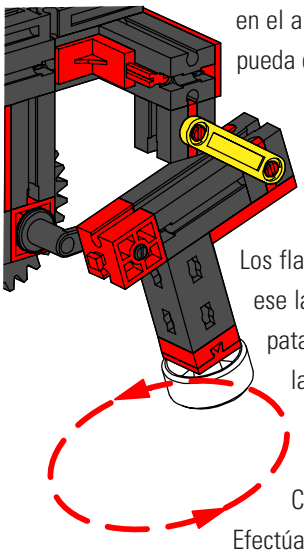
■ Una vez que hemos hablado con todo detalle los robots que se mueven sobre ruedas, pasamos a otro modo de locomoción que también podemos utilizar en los robots móviles: la carrera o marcha.
El modo de andar de los insectos se antoja extraordinariamente apropiado como modelo para la locomoción de lo que podríamos llamar "robots insectoides". En lo que se denominado "paso a tres", siempre se encuentran separadas del suelo simultáneamente tres de las seis patas: la delantera y la trasera de uno de los lados y la intermedia del lado contrario.

Paso a tres

Las patas que quedan sobre el suelo (marcadas en negro) actúan como un trípode estable, de forma que el modelo siempre está seguro y no vuelca al andar.

Las patas del robot andador de fischertechnik están concebidas como mecanismos de cuatro articulaciones. El sistema de articulación utilizado en estas patas se conoce con el nombre de "mecanismo biela manivela". Impulsadas por una manivela, las piezas móviles del mecanismo efectúan un movimiento alternativo. La distancia entre cada articulación y el punto podal (el punto de apoyo en el suelo) se calcula de modo que el punto podal describa un movimiento elíptico al girar la manivela impulsora. De este modo se produce un movimiento que recuerda al de los pasos dados durante la carrera.

Las 6 manivelas que impulsan las patas deben ajustarse siguiendo al pie de la letra las instrucciones de montaje. La posición de las manivelas de las tres patas que permanecen en el suelo simultáneamente sigue siendo la misma, mientras que las manivelas de las tres patas que en ese momento se levantan en el aire giran 180º. La correcta posición de las manivelas entre sí garantiza que el modelo pueda correr con la secuencia de pasos adecuada, el paso a tres.



Las tuercas con las que se fijan las ruedas dentadas al eje deben estar bien apretadas para que las manivelas no se desajusten durante la carrera.

Los flancos izquierdo y derecho del modelo son impulsados por el motor que corresponde a ese lado (algo necesario para la carrera en curva). Por ello es importante garantizar que la pata intermedia de cualquiera de los lados se encuentra siempre en la misma posición que las patas delantera y trasera del lado contrario. De esta sincronización se ocupa el software gracias a los pulsadores I1 e I2.

Construye ahora el modelo tal y como se describe en las instrucciones de montaje. Efectúa una prueba de interfase para comprobar si todos los pulsadores y motores están bien conectados. Sentido de giro de los motores: izquierda = avance.

Tarea 1 (nivel 1):
Enseña al robot a correr.

- **Programa el modelo para que avance en línea recta con paso a tres.**
- **Utiliza los pulsadores I1 e I2 para sincronizar las patas de la derecha y de la izquierda.**
- **Recuerda que las dos patas exteriores de un lado y la pata intermedia del otro deben estar siempre en la misma posición.**

Consejos:

- Empieza por colocar las patas de ambos flancos en su posición inicial. Enciende después los dos motores (con sentido de giro a la izquierda).
- La carrera no debe continuar hasta que los dos pulsadores I1 e I2 estén oprimidos (necesitamos efectuar estas consultas en el instante en que el modelo haya de dar el segundo paso).
- Los motores deben seguir en marcha hasta que el pulsador correspondiente (I1 para M1, I2 para M2) vuelva a presionarse. Es importante que el modelo no dé el siguiente paso hasta que se opriman los dos pulsadores, porque así es como cada pata se coloca en posición correcta con respecto a las demás. Eso sí: para que funcione es imprescindible que las manivelas impulsoras de las patas estén bien ajustadas, conforme a lo se indica en las instrucciones de construcción.
- En ese momento puede reemprenderse el proceso y el robot dará el segundo paso. El modelo seguirá andando hacia adelante hasta que nosotros detengamos el programa.
- Encontrarás el programa completo en el archivo Robot andador 1.rpp.

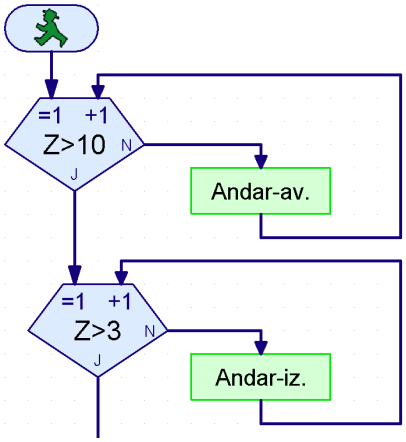
Igual que con el modelo básico que se desplazaba sobre ruedas, puedes hacer que el modelo ande hacia la izquierda, hacia la derecha o hacia atrás variando el sentido de giro de los motores. Para contar los pasos puedes usar I1 o I2.

Tarea 2 (nivel 2):

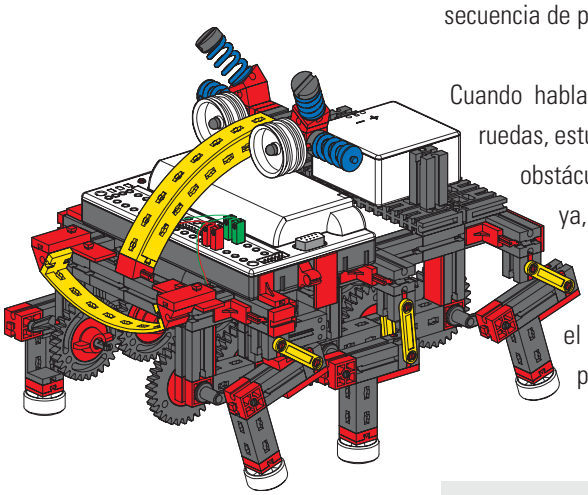
- **Programa el modelo para que dé 10 pasos hacia adelante, 3 hacia la izquierda, 3 hacia la derecha y otros 10 hacia atrás.**
- **Aplica un subprograma para cada dirección.**
- **Para contar los pasos utiliza el elemento bucle contador.**

Consejos:

- Copiar sencillamente el programa "Robot andador 1.rpp" en un subprograma.
- Después copia este subprograma tantas veces como sea necesario para cubrir las distintas direcciones de carrera. Ve modificando en cada subprograma el sentido de giro de los motores para que el modelo se mueva en la dirección deseada.
- Utiliza el elemento bucle contador para contar el número de pasos por cada sentido de giro. El modelo da un paso por cada ejecución del subprograma; dicho de otra forma: si el programa hace 10 repeticiones del ciclo con el subprograma, el modelo da 10 pasos.



De esta manera puedes enseñar al robot a efectuar la secuencia de pasos que quieras (Robot andador 2.rpp).



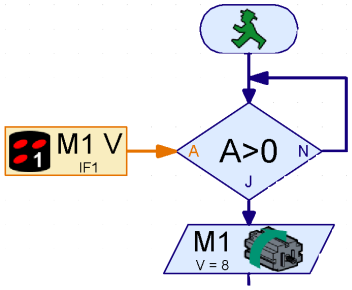
Cuando hablamos de los robots que se desplazan sobre ruedas, estudiamos con detalle el tema de la detección de obstáculos. No vamos a repetir todo que se ha dicho ya, pero sí queremos proponerte que reproduzcas el mismo comportamiento en el robot andador. Encontrarás todos los sensores necesarios en el kit de construcción modular; para la parte de programación, guíate por lo que hicimos con el robot sobre ruedas. Mucha suerte!

Posibilidades de extensión

■ Las funcionalidades que ofrece la ROBO Interface no se limitan a las que se han mostrado hasta ahora en referencia a los robots móviles. Para explorarlas, son necesarios determinados componentes extra que no forman parte del kit de construcción modular. Como quiera que entrañan un enorme interés para la actividad con robots, no queremos dejar de presentarte algunos de ellos.

Emisor de infrarrojos manual

■ La ROBO Interface contiene un diodo que recibe los infrarrojos enviados por el emisor manual IR Control Set Art. Nº 30344. Esto te permite consultar los pulsadores del emisor manual desde ROBO Pro como si fuesen entradas digitales para así poder, por ejemplo, encender y apagar los motores.



A modo de ejemplo hemos programado un mando a distancia para el robot andador. Con las 4 flechas ovaladas del mando a distancia podrás controlar el modelo para que avance y retroceda a izquierda y derecha. Antes, eso sí, deberás cargar el programa Robot andador IR.rpp en la interfase.

Otro programa genial también relacionado con el control remoto es el Mobile-Teach-IR.rpp. Este programa Teach-In te permite teledirigir uno de los robots sobre ruedas (por ejemplo el robot sencillo o el modelo básico). El modelo recuerda los trayectos recorridos y puede así repetirlos cuantas veces quieras. Los trayectos memorizados, sin embargo, se pierden al detener el programa.
Crear este tipo de programa es posible gracias al elemento Listado de ROBO Pro. En este elemento pueden almacenarse y recuperarse un gran número de valores (ver el manual de ROBO Pro). El programa en sí es bastante complejo, pero su aplicación no puede ser más sencilla:

1. Carga el programa Mobile-Teach-In.rpp en la memoria FLASH de la ROBO Interface e inícialo.
2. Oprime el botón **M1** ▸/▸▸ del mando a distancia. Empezará entonces el "proceso de aprendizaje".
3. Dirige el modelo en la dirección que desees con las flechas ovaladas.
4. Oprime el botón **M2** ▸/▸▸. El trayecto recorrido quedará memorizado.
5. Oprime el botón **M3** ▸/▸▸. El robot recorrerá el trayecto memorizado.

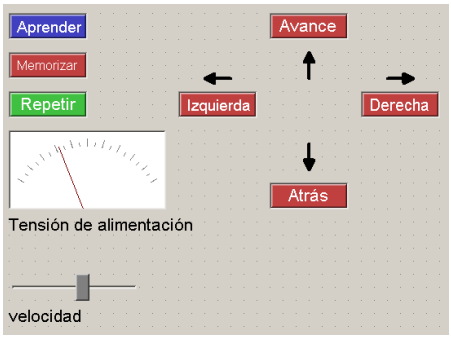
Con esta aplicación, programar robots se convierte en un juego de niños! Debes tener en cuenta, eso sí, que los trayectos memorizados se pierden al detener el programa con el botón Prog de la interfase.

ROBO RF Data Link

■ El enlace de datos por radiofrecuencia ROBO RF Data Link Art. Nº 93295 sustituye al cable de conexión del PC con la interfase estableciendo una conexión por radiofrecuencia para la transmisión de datos, lo que constituye una gran ventaja. Lo primero que debes hacer cada vez que cargues un programa en la interfase es conectar primero y desconectar después el cable en cuestión. Lo segundo, ejecutar el programa en modo online para detectar los posibles errores con mayor facilidad que si lo hicieses en modo de descarga. Finalmente controlamos desde la pantalla los robots móviles en modo online con un panel de control de ROBO Pro igual que hacemos con el mando a distancia de infrarrojos. A diferencia de lo que ocurre con el mando a distancia, desde la pantalla pueden verse también los datos que devuelve la interfase como por ejemplo valores de variables o entradas analógicas, tensiones de alimentación arrojadas por el acumulador o velocidades de los motores.

Como ejemplo hemos modificado el programa Teach-In para manejar el robot sobre ruedas desde un panel de control. El programa se llama Mobile-Teach-RF.rpp. Huelga decir que también puedes probar con el cable, pero descubrirás que es bastante incómodo. El radio de acción del modelo quedará restringido, el cable acabará hecho un lío y el robot no será capaz de girar bien. Sin duda acabarás deseando hacerte cuanto antes con el RF Data Link.

Carga el programa Mobile-Teach-RF.rpp.
Cambia a panel de control en la barra de función del programa principal. A continuación inicia el programa en modo online. Ya puedes controlar y programar el modelo desde los botones del panel de control.



1. Oprime el botón "Aprender". Empezará entonces el "proceso de aprendizaje".
2. Usa las flechas para dirigir el modelo en la dirección que desees.
3. Oprime el botón "Memorizar". El trayecto recorrido quedará memorizado.
4. Oprime el botón "Repetir". El robot recorrerá el trayecto memorizado.

Aquí también se pierde el camino recorrido al cerrar el programa.

En el manual ROBO Pro encontrarás más información sobre los paneles de control.

ROBO I/O-Extension

■ Si por cualquier razón construyeses un modelo con tantos sensores y motores que las entradas y salidas de la ROBO Interface no fuesen suficientes, puedes conectar a la interfase una extensión ROBO I/O-Extension Art. Nº 93294. De este modo obtendrás otras 8 entradas digitales, 4 salidas de motor y una entrada analógica de resistencia. A esta extensión de entradas y salidas puedes conectar todavía un segundo módulo, y a ese segundo módulo todavía un tercero, que podrás después gestionar en su totalidad con una ROBO Interface. En total tendrás a tu disposición 16 salidas de motor, 32 entradas digitales, 5 entradas analógicas de resistencia, 2 entradas analógicas de tensión y 2 entradas para sensores de distancia.

Si todo esto sigue quedándosete corto, todavía tienes la posibilidad de gestionar desde el PC varias interfases en modo online; por ejemplo, una en un puerto COM en serie y otra en un puerto USB, ó 2 interfases en el puerto USB, con un máximo de hasta 3 módulos ROBO I/O-Extension cada una. Da vértigo sólo de pensarlo! El funcionamiento del conjunto se trata también en el capítulo 6 del manual de ROBO Pro.

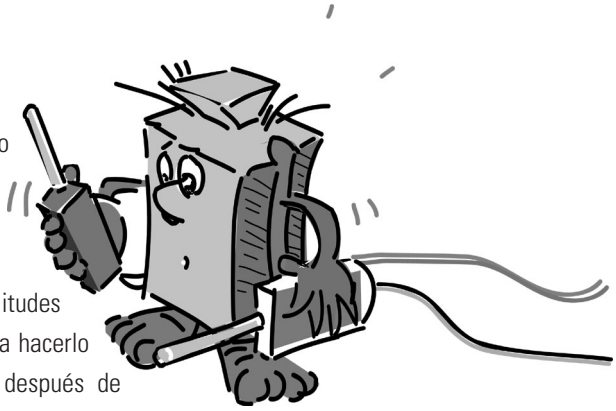
Búsqueda de errores

■ Experimentar es divertido... mientras todo va saliendo como debe. Casi siempre sale todo bien. Casi, pero no siempre.

Hasta que un modelo no nos da problemas, no comprobamos realmente si hemos comprendido el mecanismo en su totalidad ni detectamos los posibles fallos que podamos haber cometido.

En el caso de los fallos mecánicos, al menos hay algo que se puede ver (componentes mal montados) o percibir (falta de suavidad en el funcionamiento). Pero si hablamos de problemas eléctricos, las cosas se complican.

Para localizar errores, los profesionales utilizan una serie de instrumentos de medición muy variados, desde voltímetros a oscilógrafos. Estos aparatos no están al alcance de cualquiera, por eso vamos a tratar de localizar y corregir errores con medios más sencillos.



Cableado

Antes de empezar con nuestros experimentos, tenemos que poner a punto algunos componentes del kit de construcción modular fischertechnik. Por ejemplo, hay que fijar los conectores que se proporcionan a cada segmento de cable.

Lo primero que debemos hacer es cortar los cables. Para ello medimos las longitudes fijadas y cortamos los segmentos indicados. Después hay que probarlos; para hacerlo necesitaremos el acumulador y la lámpara. Si la bombilla se enciende después de conectarla al acumulador, el cable está en perfecto estado. También tenemos que verificar que los códigos cromáticos están correctos: conectores rojos con cables rojos, conectores verdes con cables verdes.

Prueba de interfase

Si los programas (incluso los que se proporcionan ya hechos) no funcionan con nuestro modelo, procedemos a realizar una prueba de interfase. Esta utilidad nos permite probar por separado las entradas y las salidas. Funcionan los sensores? Giran los motores en el sentido correcto? En todos nuestros robots móviles los motores se conectan para que al girar a la izquierda la rueda o la pata se mueva hacia adelante. Si aquí está todo correcto, buscamos una causa mecánica.

Contactos flojos

Un fallo peor es la presencia de contactos flojos. Por un lado, es fácil que los conectores queden sueltos en las hembrillas. Si este es el caso, podemos abrir un poco las pestañas del conector con un destornillador pequeñito. Hay que tener mucho cuidado, porque si nos excedemos podemos llegar a romper los contactos o, cuando menos, hacer que después sea difícil introducirlos en las hembrillas.

Otra posible causa de que los contactos queden flojos es que los elementos de sujeción de los conectores y los cables se hayan aflojado, por eso hay que cerciorarse de que estén bien apretados. Conviene aprovechar la ocasión para comprobar también que no se haya roto ninguno de los hilos de cobre.

Cortocircuitos

Puede darse también el caso de que un cableado mal puesto produzca un cortocircuito que impida desde ese momento que el sistema funcione como debe. El acumulador trae incorporado un sistema de seguridad que interrumpe el flujo eléctrico en caso de que se produzca una subida de tensión o un

recalentamiento. Así mismo, las salidas de la interfase se cierran también en caso de sobrecalentamiento.
También se puede producir un cortocircuito si el tornillo con el que se sujeta el cable no queda bien asegurado y sobresale por encima del borde del conector. Si se introducen dos conectores en dos hembrillas directamente contiguas de la interfase y los tornillos se tocan entre sí, se produce el cortocircuito. Por eso hay que poner especial cuidado en que los tornillos queden siempre perfectamente ajustados y en introducir los conectores de forma que los tornillos no puedan contactar entre sí.

Alimentación de corriente

Si durante el funcionamiento de los aparatos se producen interrupciones inexplicables, puede deberse a que el acumulador esté casi agotado. Justo en el momento de tener que mover una carga (al encender un motor) cae la tensión, lo que dispara la función reset del procesador de la interfase. Si se enciende el LED rojo de la ROBO Interface es señal de que la tensión de alimentación es demasiado baja. Es el momento de recargar el acumulador.

Errores de programación

Si en los programas escritos por ti mismo surgen errores que no consigues explicarte, es conveniente, para estar seguro, que ejecutes uno de los programas que se proporcionan ya hechos, el más parecido posible al que te dé problemas. Esto te permite descartar que los fallos se estén produciendo por defectos mecánicos o eléctricos. En el modo online es posible ir siguiendo los pasos del programa en la pantalla. Si el programa llega a un punto del que no consigue pasar, podemos tratar de buscar la causa en ese punto; por ejemplo, puede ser que haya una entrada o un motor mal seleccionado, que en una bifurcación se consulte el valor que no es o que estén invertidos los contactos S/N.

Si todo esto sigue sin dar resultado, existe la opción de contactar con el servicio técnico de fischertechnik (correo electrónico info@fischertechnik.de).

O de visitar la página de fischertechnik en internet: www.fischertechnik.de, donde encontrarás un foro, un chat, un tablón de anuncios de compra y venta y una galería de imágenes, y donde tendrás la oportunidad de afiliarte sin coste alguno al Fischertechnik Fanclub.

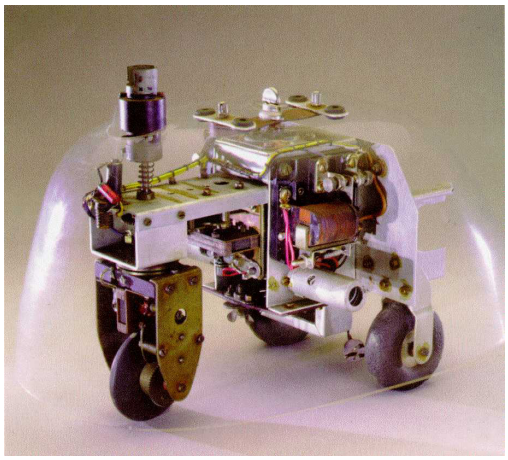
Te deseamos muchas horas de diversión con el ROBO Mobile Set. Estamos seguros de que serán muchas las veces que exclames "eureka!".

Conteúdo

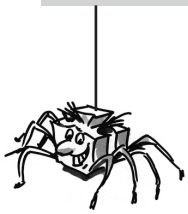
Para que precisamos de robôs?

■ O termo Robô foi utilizado pela primeira vez em 1923 no romance "Golem" de Carel Capek. Esta figura criada no meio artístico devia, com as suas características, substituir o trabalho humano.
Nos anos 30 e 40 do século XX, a partir do robô surgiu uma arte robótica. Diversas tentativas de lhe fornecer características humanas, por exemplo uma cabeça com olhos formados por lâmpadas a piscar, nos provocam ainda atualmente um sorriso cansado. Mobilidade ou até mesmo inteligência ainda se nota muito pouco nestas máquinas. Como o princípio do controle tem grande influência na técnica robótica, a chegada de conexões eletrónicas tornaram a construção de robôs mais realista. A questão da "inteligência" de robôs é ainda hoje objeto de pesquisa e investigação de muitas firmas, institutos e universidades.

■ As primeiras aproximações ao problema foram esperadas da denominada cibernética. A designação "cibernética" deriva da palavra grega Kybernetes. Kybernetes era o navegador de barcos a remos gregos. Ele tinha de determinar o local do barco e alcançar a rota necessária até ao destino.
Com isso fica claro que a cibernética deveria tornar o robô "inteligente". Como se pode representar tal comportamento inteligente?
Nós queremos tentar esclarecer este fato com a ajuda de uma experiência de pensamento. Todos já observamos o comportamento de uma mariposa no círculo de luz de uma lâmpada. A mariposa detecta a fonte de luz, voa para lá, e a seguir se afasta pouco antes de bater na lâmpada. Através deste comportamento se torna evidente que a mariposa detecta a fonte de luz, pesquisa um caminho até ela e, depois, tem de voar para lá. Estas capacidades se baseiam em modelos de comportamento instintivos, inteligentes do inseto.
Tentamos transferir agora estas capacidades para um sistema técnico. Temos de reconhecer a fonte de luz (sensores ópticos), executar um movimento (comandar motores) e temos de estabelecer uma relação conveniente entre reconhecimento e movimento (o programa).



■ A experiência de pensamento descrita anteriormente foi realizada pelo inglês Walter Grey nos anos 50.
Com a ajuda de sensores simples, motores e conexões eletrônicas foram conseguidos diferentes animais "cibernéticos" que adotaram comportamentos muito específicos, como por exemplo o de uma mariposa. A figura mostra uma réplica da tartaruga "cibernética" que está exposta no Museu Smithsonian em Washington

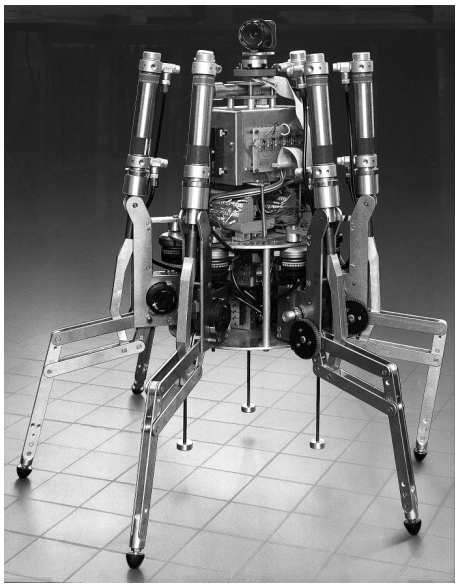


■ Mas para que precisamos de robôs móveis? Uma vez tentamos aplicar o comportamento da nossa "mariposa de pensamento" a aparelhos técnicos. Um exemplo simples é a busca de luz. Modificamos a fonte de luz, colocando uma tira clara, a linha guia, no chão e os sensores deixam de estar orientados para a frente e sim para baixo. Com a ajuda das linhas guia desse tipo, um robô móvel pode se orientar, por exemplo, num hall de armazém. Informações adicionais, por exemplo na forma de código de barras em determinados locais, levam o robô nestas posições a outras ações, como por exemplo a recolha ou depósito de uma palete.

Tais sistemas robóticos já existem na realidade. Em grandes hospitais existem longas vias de transporte para materiais de consumo, como por exemplo roupas de cama. O transporte destes materiais pelo pessoal da manutenção é dispendioso e está ligado, em parte, a trabalho físico intenso. Além disso, tais atividades se limitam ao tempo existente para a manutenção dos pacientes.

■ Há alguns anos, os cientistas se ocuparam com uma outra forma de movimento muito generalizada na natureza, o andar e o correr. Foram desenvolvidos robôs que conseguiam se movimentar com pernas. Um exemplo de um robô andante de seis pernas é o robô andante eletropneumático "Achille" desenvolvido na academia real militar em Bruxelas. Equipado com uma câmara no topo e nas seis pernas, este robô deve reagir mecanicamente a obstáculos elevados ou profundos (objetos ou buracos).

Tais máquinas andantes podem ser aplicadas onde os veículos de rodas e de lagarta não têm qualquer chance, como por exemplo em terrenos exteriores irregulares ou flexíveis, no caso ultrapassar obstáculos, subir escadas, passar valas ou na aplicação em pontos perigosos e de difícil acesso em centrais nucleares, galerias de minas ou em ações de salvamento.



Reconhecemos também que o robô móvel pode ocupar um lugar importante na sociedade moderna.

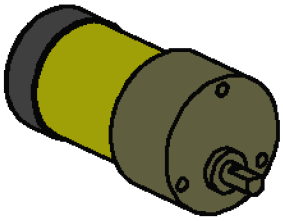
Robôs da fischertechnik

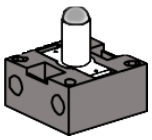
■ Como podemos construir robôs a partir dos nossos módulos da fischertechnik? Para um robô precisamos, além dos sensores (por exemplo botões) e atuadores (por exemplo motores), de muitas peças para construir um modelo. O módulo da fischertechnik ROBO Mobile Set é a base ideal. Os seguintes sensores e atuadores estão incluídos neste módulo:

Atuadores

Motor power:

Dois destes motores em série de corrente contínua (9VDC/2,4W) com caixa de velocidades incorporada e uma redução de 50:1 acionam os modelos de robôs móveis (i.e. o motor roda 50 vezes à volta do veio, que se salienta do motor, no mesmo tempo apenas uma vez).





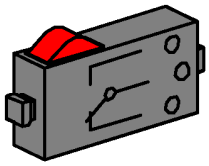
Lâmpada de formato lenticular:

Para a edição de sinais de luz mais simples é apropriada a lâmpada incandescente (9VDC/150mA). Na ampola de vidro da lâmpada está integrada uma lente que enfaixa a luz que sai. Se o feixe de luz for direcionado para um sensor de luminosidade (fototransístor, ver em baixo) pode se montar uma barreira fotoeléctrica que distingue a luminosidade da escuridão. A lâmpada pode também ser utilizada para indicar determinados objetos ou como lâmpada a piscar para emissão de mensagens de aviso. No módulo é utilizada a lâmpada juntamente com 2 fototransístores como sensor especial para reconhecimento da linha.

Sensores

■ No caso de sensores se distingue entre sensores digitais e analógicos.

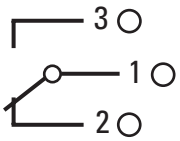
O **botão** é um exemplo de um sensor digital. Dimensões digitais podem aceitar 2 estados diferentes. Estes estados são designados por 0 ou 1. Para o botão, o "0" significa que não passa qualquer corrente entre as ligações, "1" significa que há corrente.

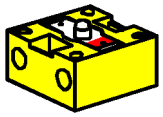


O botão da fischertechnik está concebido como seletor. Por isso, possui 3 ligações. Ao premir o botão vermelho é acionado mecanicamente um interruptor que liga as conexões 1 e 3 entre si. Simultaneamente, o contato é interrompido entre as ligações 1 e 2 que estavam ligadas no estado de repouso. Desta forma, podem ser consultadas ambas as posições iniciais possíveis:

No estado de repouso fechado (ligações 1 e 2 ocupadas)

No estado de repouso aberto (ligações 1 e 3 ocupadas)





O **fototransístor** pode ser utilizado como sensor digital assim como sensor analógico. No primeiro caso, ele serve para a detecção da passagem nítida claro-escuro, p. ex. de uma linha selecionada. As quantidades de luz se podem distinguir pela sua intensidade, nesse caso o fototransístor funciona como sensor analógico. Os valores analógicos podem ser alterados como quiser entre os seus valores limite. Para que estas grandezas possam ser processadas por computador, têm de ser convertidas em valores numéricos correspondentes.

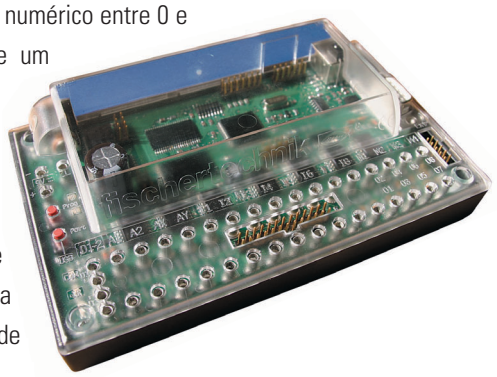
No caso do fototransístor, se trata normalmente de um chamado dispositivo semicondutor, cujas propriedades elétricas estão dependentes da luminância. Todos conhecem as células solares com as

quais se pode produzir corrente a partir da luz solar. Podemos entender o fototransístor como uma combinação de mini células solares e transístor. Os impulsos de luz que incidem no fototransístor (fotões) criam uma corrente muito pequena que é fortalecida pelo transístor.

Nota:
Na ligação do fototransístor prestar atenção à polaridade correta: Marcação vermelha = positivo.
Tensão permitida: 30V máx.

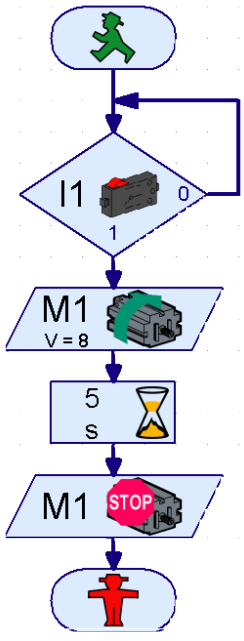
ROBO Interface

■ No ROBO Interface podemos ligar e avaliar diferentes sensores e atuadores. Além das 8 entradas digitais, o ROBO Interface dispõe de outras entradas analógicas. Assim, por exemplo, um valor de resistência aplicado nas entradas AX e AY entre 0 e 5,5 kΩ é convertido num valor numérico entre 0 e 1024. Os valores de medição de um sensor de luminosidade, por exemplo de um fototransístor, são recolhidos e estão disponíveis para um outro processamento. Nas entradas analógicas A1 e A2 podem ser medidas tensões entre 0 e 10VDC.



A função mais importante da interface consiste na combinação lógica de grandezas de entrada. Para isso, a interface necessita de um programa. O programa decide que tipo de dados de entrada são considerados os sinais do sensor, os dados de saída adequados, sinais de controle do motor etc.. Com o ROBO Interface dispomos de suficiente potência de processamento de dados para projetar programas exigentes.

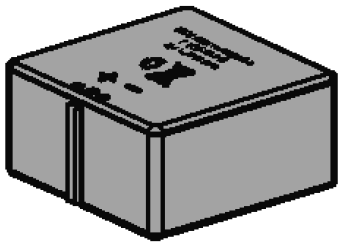
Software ROBO Pro



■ Com isso é efetivamente possível conseguir criar os programas necessários para a interface, pois existe uma superfície gráfica de programação. Por trás do termo "superfície gráfica de programação" se esconde um software que nos possibilita criar os nossos programas de uma forma muito confortável. Isto acontece com a ajuda de símbolos gráficos. O computador do ROBO Interface pode executar apenas comandos a partir do seu conjunto de comandos conhecido. Essencialmente, são estruturas de controle simples, cujas aplicações são extraordinariamente difíceis para os principiantes. O software do PC do ROBO Pro coloca, por isso, à disposição elementos gráficos que são traduzidos, posteriormente num idioma executável para a interface.

Alimentação de corrente

■ A única coisa de que ainda necessitamos para o módulo ROBO Mobile Set, é o Accu Set art. n.º 34969. Contém o conjunto de baterias como alimentação de corrente móvel para os nossos modelos de robôs e um carregador especial para o conjunto de baterias.



O melhor será carregar a bateria logo com o carregador, para que esteja carregada quando desejarmos começar as experiências.

Procedimento em experiências

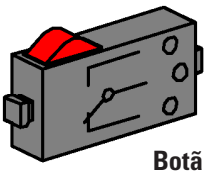
■ Na nossa entrada no fascinante mundo dos robôs móveis iremos avançar passo a passo. Começamos com uma montagem de teste simples para a verificação das funções básicas da interface e sensor. Em seguida, montamos modelos simples cujas tarefas estabelecidas estão atribuídas e procuramos sempre sistemas mais complexos.
Quando a criação de programas próprios se torna demasiado complexa e demora demasiado tempo, então se pode carregar os programas de exemplo fornecidos na interface e acionar com eles o robô.
Para não desesperar quando ocorrerem erros, existe no final um capítulo sobre busca de falhas.

Um ponto muito importante é o cuidado na montagem e na colocação em funcionamento dos nossos robôs. Na ligação dos componentes elétricos seguimos escrupulosamente as instruções e verificamos duas ou três vezes se tudo estava em condições. No caso de construções mecânicas, assim como no caso de criações próprias, prestamos muita atenção à mobilidade e à folga reduzida nas caixas de velocidades e fixações. A nossa criatividade está reservada para escrever programas próprios e definir um novo "comportamento". Só está limitada à quantidade de memória e capacidade de processamento de dados do hardware. Os seguintes exemplos fornecem alguns estímulos.

Primeiros passos

■ Após as reflexões teóricas pretendemos agora começar a efetuar algumas experiências. Certamente que vai querer experimentar imediatamente, talvez até com um robô andante grande. Isso é possível e, prestando uma cuidadosa atenção às instruções de montagem, a montagem do modelo se consegue logo na primeira tentativa.
Mas o que fazer quando não funciona? Nestes casos, tem de se tentar encontrar sistematicamente causas para os erros. Antes de nos ocuparmos disso, verificamos a combinação do computador e interface.

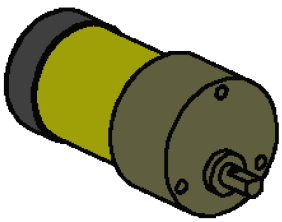
No manual de instruções do ROBO Pro é descrito no capítulo 1 e 2, como o software de controle é instalado no PC e como a interface é conectada. Com a ajuda do teste de interface testamos os diferentes sensores e atuadores.



Botã

Botã

Podemos agora, por exemplo, ligar um botão na entrada digital I1 e observar a alteração do estado da entrada ao acionar o botão.



Motor power

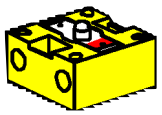
Motor power

Verificamos as saídas ao mesmo tempo que ligamos um motor com uma saída de motor, por exemplo M1. Com o botão esquerdo do mouse, podemos colocar o motor em rotação e alterar a velocidade com a válvula corrediça.

Fototransístor

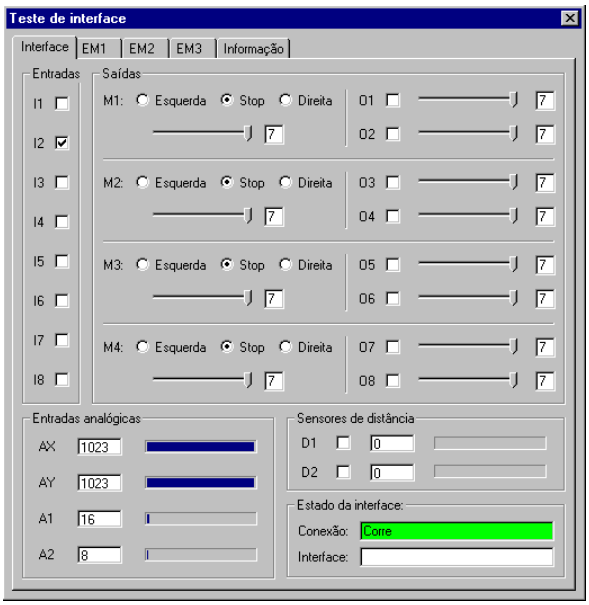
Se pretendermos testar também a entrada analógica AX, podemos utilizar um fototransístor como sensor analógico.

Enquanto que no caso do motor ou do botão a polaridade não é relevante (o motor funciona na pior das hipóteses em rotação contrária), a ligação correta do fototransístor é extremamente importante para o funcionamento correto.
Ligamos o contato do transístor com a marcação vermelha com um conector adaptador, o outro contato com um conector verde. O segundo conector verde encaixa no plugue da entrada AX que se encontra perto da margem da interface, o segundo conector vermelho se adapta na plugue de AX que se encontra bem no interior. (atenção: Se se fechar o fototransístor na entrada digital I1-I8, o conector vermelho encaixa na plugue que se encontra perto da margem da caixa).

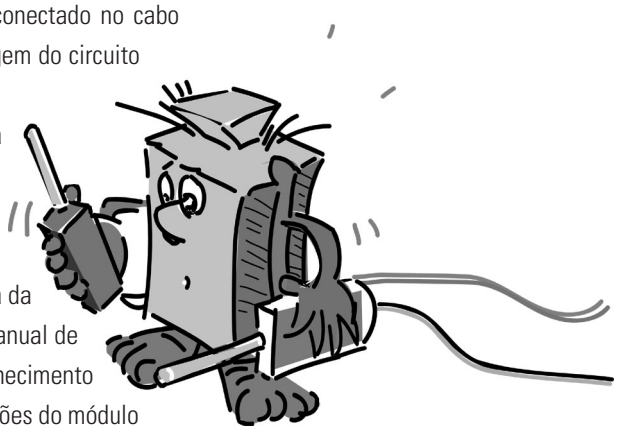


Fototransistor

Agora podemos variar a intensidade da luminosidade do transístor com a ajuda de uma lanterna de bolso e alterar a amplitude da viga azul de AX. Se o indicador não se mover em caso de uma amplitude máxima, volte a observar as ligações do fototransístor. Se, pelo contrário, o indicador estiver em zero com a lâmpada de bolso desligada, então pode ser que a iluminação na sala, a luminosidade do ambiente, seja demasiado elevada. A amplitude se altera quando destapamos o fototransístor.



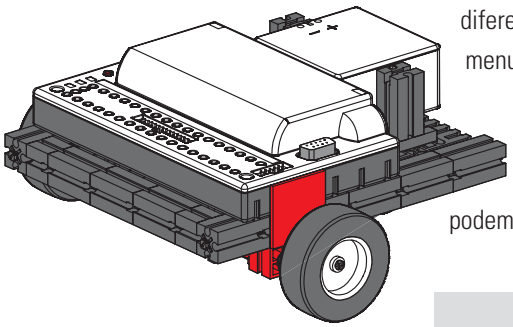
Voltando novamente ao assunto da distribuição de cores dos conectores: Prestamos sempre atenção para que, na montagem, um conector vermelho seja conectado no cabo vermelho e para que um conector verde seja conectado no cabo verde. Se a montagem do circuito depender da polaridade correta, colocamos sempre um cabo vermelho para o pólo positivo e um verde para o negativo. Pode parecer um pouco meticuloso, mas para uma busca de falhas sistemática, uma atribuição clara de cores torna tudo consideravelmente mais fácil.



Com um programa simples pretendemos concluir os nossos primeiros passos na área da robótica. O programa "controle do portão da garagem" explicado no capítulo 3 do manual de instruções ROBO Pro não tem a ver com robôs móveis, sendo mais adequado ao conhecimento do software ROBO Pro. Para poder executar o programa basta ligar o motor e três botões do módulo ROBO Mobile Set à interface. Tudo o resto é descrito pormenorizadamente no manual de instruções do software.

O primeiro robô simples

■ Após o teste de interface e o controle do portão da garagem pretendemos, por fim, colocar o primeiro robô em funcionamento. Montamos o modelo "robô simples" com ambos os motores de acionamento de acordo com as instruções de montagem. Isso é relativamente fácil e rápido, porque este modelo contém apenas o necessário que um robô precisa para andar. Ligamos os motores nas saídas M1 e M2.



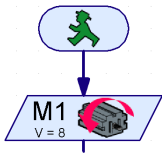
Nós acedemos ao ROBO Pro e colocamos um novo programa (ARQUIVO – NOVO). ROBO Pro oferece diferentes níveis de dificuldade em que podemos trabalhar. No ROBO Pro pode configurar o item do menu LEVEL. Para já, basta-nos o Level 1.

Surge uma folha de trabalho vazia e na margem esquerda da tela a janela elementar, na qual podemos selecionar diferentes elementos de programa com o botão esquerdo do mouse e podemos colocar na superfície de trabalho. Com o botão direito do mouse alteramos as propriedades.

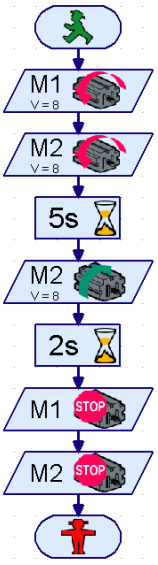
Tarefa 1 (Level 1):
O nosso "robô simples" deve andar em frente por 5 segundos, a seguir em circuito por 2 segundos e, depois, permanecer parado.

Dicas:

Programamos o primeiro robô passo a passo em conjunto:



- Começamos com o semáforo pedestre em movimento. Simboliza o início do programa.
- Em primeiro lugar, buscamos o símbolo do motor da janela do elemento e colocámo-lo por baixo do elemento inicial de forma que a linha de conexão é indicada automaticamente. Na janela Propriedades configuramos a saída do motor "M1" assim como o sentido de rotação "à esquerda" e confirmamos com OK.



- Por baixo deste símbolo colocamos da mesma forma um símbolo do motor e ligamos o motor 2.
- Para aguardar um determinado tempo, utilizamos o elemento Tempo de espera, colocámo-lo por baixo do segundo símbolo do motor e configuramos para 5 segundos.
- A seguir, permitimos ao motor M2 rodar na outra direção (para a direita), aguardamos 2 segundos e desligamos ambos os motores. O nosso programa termina com o símbolo de fim, o semáforo pedestre parado. A figura indica o processo do programa pronto.

Quem não tiver a certeza se está tudo pronto, pode comparar o seu programa com o programa de exemplo fornecido. Para isso, o próprio programa é salvado posteriormente e o arquivo Robô simples 1.rpp é carregado a partir do diretório de exemplo do ROBO Pro (configuração padrão C:\Programas\ROBO Pro\Programas de exemplo\ROBO Mobile Set).

Se tudo estiver em condições, o programa é carregado por download na interface. Após pressionar o botão Download surge uma janela de diálogo. Aí configuramos que o programa deve ser carregado na memória FLASH 1 e iniciado imediatamente após o download.

Logo após o download o nosso modelo se põe em marcha, roda e, a seguir, permanece parado. Se desejarmos reiniciar o programa, premimos a tecla Prog na interface. O LED Prog1 volta a piscar, enquanto o programa corre. A seguir, se acende permanentemente. O programa na memória FLASH da interface permanece parado quando a alimentação de corrente é interrompida na interface.

Experimentamos ao mesmo tempo que retiramos o conector na bateria. Voltamos a encaixar o conector, selecionamos o programa salvado, ao mesmo tempo que premimos a tecla Prog até acender o LED Prog1. Voltamos a premir a tecla e iniciamos o programa.

É muito pouco aquilo que o robô faz até agora, não? Então vamos alargar um pouco a tarefa.

Tarefa 2 (Level 1):

Para que o nosso robô não pare logo após 7 segundos, pretendemos apresentar-lhe agora a dança.

- **Faça-o andar para em frente, para a esquerda, para a direita, para trás, e a diferentes velocidades.**
- **O decurso se deve repetir até o programa ser concluído com o botão Prog na interface.**

Dicas:

- Inverta simplesmente a polaridade dos motores de forma que o robô se desloque nas direções desejadas.
- Pode configurar as velocidades dos motores na janela Propriedades com cada símbolo do motor de 1 a 8. Rodando M1 e M2 com diferentes velocidades na mesma direção, o robô faz uma curva.
- Para que o programa se repita constantemente, arraste a linha de conexão da saída do último elemento do programa para a linha que acompanha o primeiro elemento.
- Encontra um exemplo pronto em Robô simples 2.rpp.

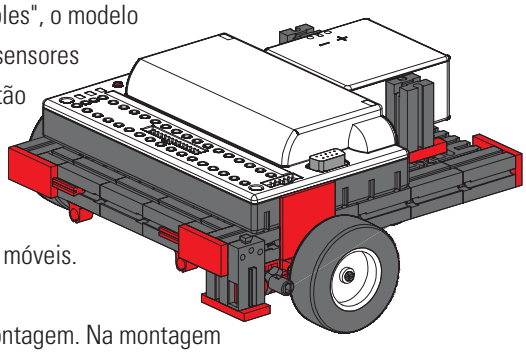
Parabéns, conseguiu construir e programar sozinho o seu primeiro robô. Não é especialmente inteligente, pois não detecta obstáculos e cai da mesa se não prestar atenção. Mas isso ainda se poderá alterar no decorrer de outras experiências.

Robôs rolantes inteligentes

■ Para que um robô possa detectar o seu ambiente, ele necessita de sensores. Com as seguintes propostas de modelos são apresentadas algumas variantes de robôs móveis, nos quais é experimentada a aplicação de diferentes sensores. Nesse caso, depende de associar não só os estados internos do robô, por exemplo medição da distância através de rodas movidas por impulsos, mas também sinais do exterior, por exemplo na busca de luz ou de trilha. A este modelo são propostas determinadas tarefas. Elas devem servir de estímulo e familiarizá-lo com a matéria. Os programas sobre as tarefas individuais se encontram no diretório ROBO Pro em \Programas de exemplo\ROBO Mobile Set\. Procure encontrar outras funções para os modelos. Ao dar uma olhada nos seguintes exemplos, certamente que vão surgir mais idéias.

Modelo básico

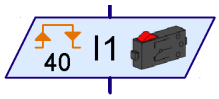
■ Em comparação com os nossos primeiros robôs "robô simples", o modelo básico é mais estável e mais robusto. Contém, além disso, 2 sensores para a medição da distância que são constituídos por um botão e por uma roda movida por impulsos. A roda movida por impulsos está conectada com o eixo do motor e aciona quatro vezes uma tecla em cada rotação do motor.
Este modelo serve como base para outros modelos de robôs móveis.



Construa o modelo básico de acordo com as instruções de montagem. Na montagem proceda com muito cuidado. Quando tudo estiver pronto do ponto de vista mecânico, verifique a mobilidade dos motores, conectando cada motor sem a interface diretamente à bateria.

Tarefa 1 (Level 1):

- **Programar a interface de forma que o modelo se desloque 40 impulsos para a frente. Para a medição dos impulsos utilize o botão de contagem na entrada I1.**
- **Meça o percurso que o modelo percorre e calcule que distância é percorrida por impulso.**
- **Repita a tentativa 3 vezes e tome nota na tabela a forma como os valores oscilam.**



Dicas:

- A seguir, ligue ambos os motores (sentido de rotação à esquerda).
- Para contar os impulsos em I1, utilize o elemento do programa **Contador de pulsos**.
- Conte ambos os flancos de impulsos (0-1 ao premir, 1-0 ao largar o botão). Pode configurar isto na janela Propriedades em **Tipo de impulso**. Dessa forma, aumenta a exatidão da medição da distância.
- A seguir, volte a desligar os motores e termine o programa.
- Pode consultar o programa já concluído em Modelo básico1.rpp.

Resultado:

	Quantidade de impulsos	Percurso percorrido	Percurso/impulso
Tentativa 1	**40**		
Tentativa 2	**40**		
Tentativa 3	**40**		

De forma aproximada se pode anotar que o modelo percorre cerca de uma distância de um centímetro por impulso.

Entretanto, sabe também que sentido de rotação tem de configurar para os motores individuais, para que o modelo se desloque numa determinada direção. Anote estes conhecimentos na seguinte tabela para que não tenha de pensar neles em cada alteração do sentido do deslocamento. Se cablou corretamente os modelos, como está descrito nas instruções de montagem, o sentido de rotação à esquerda significa em cada motor que a roda gira para a frente. É desta forma que são programados os motores em todos os programas de exemplo.

Complete a tabela:

Sentido marcha modelo	Sentido rotação M1	Sentido rotação M2
Para a frente	**Esquerda**	**Esquerda**
Para trás		
Esquerda		
Direita		
Pare		

Para não precisar colocar dois símbolos do motor na tela em cada mudança de direção, pode criar para cada direção um subprograma que assume esta tarefa. Isto facilita muito a programação. A forma como criar subprogramas está descrita no manual de instruções do software ROBO Pro no capítulo 4. Logo que tenha lido este capítulo, pode se aventurar na próxima tarefa. No ROBO Pro passa para o **Level 2**.

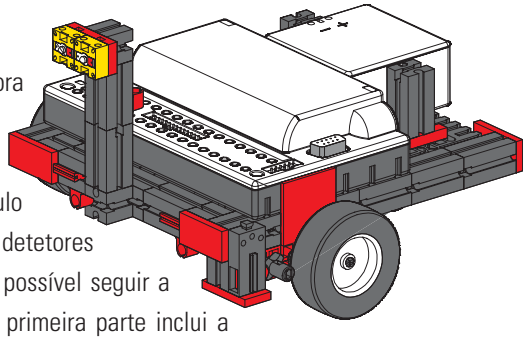
Tarefa 2 (Level 2):

- **Crie para cada sentido de marcha um subprograma.**
- **Programe o modelo de forma que percorra um quadrado com bordas de um metro.**
- **Qual é a precisão da repetição?**

Dicas:

- Crie, em primeiro lugar, um subprograma **Para a frente**. Pode criar os outros subprogramas através da cópia deste subprograma. Tem de adaptar apenas os sentidos de rotação dos motores.
- Para virar para a esquerda e para a direita, utilize uma velocidade reduzida. Isso aumenta a precisão.
- Para a contagem de impulsos utilize novamente o elemento Contador de pulsos e o botão na entrada I1.
- Carregue o programa para experimentar primeiro na RAM, até descobrir quantos impulsos necessita para efetuar uma rotação de 90°. Em primeiro lugar, o carregamento na RAM é mais rápido do que o carregamento na memória FLASH e, em segundo lugar, a memória Flash tem apenas uma vida útil "limitada" de aprox. 100.000 downloads.
- O programa já concluído se chama Modelo básico2.rpp.

O robô que segue um foco de luz



■ Depois de ter analisado o modelo básico, o robô deve agora aprender a reagir a sinais do ambiente. Semelhante à mariposa da nossa experiência de pensamento do primeiro capítulo, ele deve detectar a fonte de luz e segui-la. O módulo contém 2 fototransístores que estabelecemos como sendo detetores de luz. Cada sensor reage a um motor de forma que seja possível seguir a fonte de luz. O programa é composto por duas partes. A primeira parte inclui a procura de uma fonte de luz e na outra parte é realizado o seguimento ou a ativação da fonte de luz. Para isso são utilizados outros subprogramas. Após a conexão é ativado o subprograma Busca de luz. Este subprograma só é concluído depois de ser encontrada uma fonte de luz. O programa principal tenta dirigir o robô na direção da fonte de luz. Sempre que a direção do robô se desvia muito da linha ideal, um dos sensores não é iluminado pela fonte de luz. Por conseguinte, o robô corrige o seu sentido de marcha de forma que ambos os sensores possam detectar a fonte de luz.

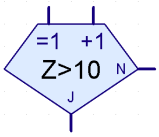
Primeiro, monte o modelo que segue um foco de luz conforme descrito nas instruções de montagem.

Tarefa 1 (Level 2):

- **Programe, em primeiro lugar, a função "Busca de luz". O robô deve virar lentamente e pelo menos a 360°. Se, durante a busca, for encontrada uma luz, o robô pára. Caso contrário, ele gira 360° na outra direção. Se ele não encontrar qualquer fonte de luz, ele deve aguardar 5 segundos antes de voltar a iniciar a busca.**
- **Se a busca de luz for bem sucedida, o modelo deve ativar a fonte de luz. Se a fonte e luz se movimentar para a esquerda ou para a direita, o robô deve seguir os movimentos da luz. Se ele perder o contato, o programa deve iniciar novamente a busca de luz. Veja se consegue atrair o robô com uma lanterna de bolso e conduzi-lo por um percurso com obstáculos.**

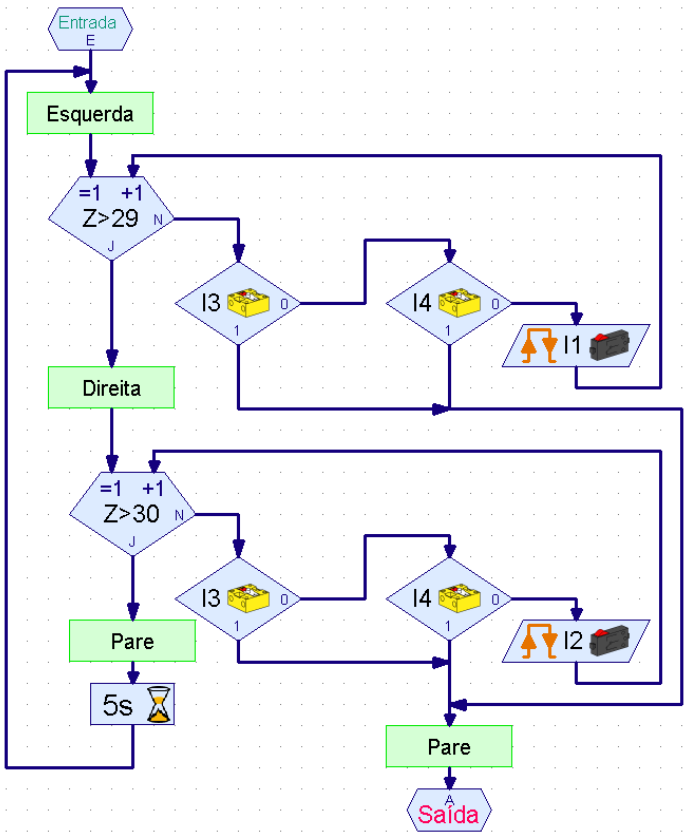
Dicas:

- Para os diferentes sentidos de marcha utilize os subprogramas que já programou para o modelo básico. Assim que o programa Modelo básico2.rpp esteja aberto, encontra **na janela grupos de elementos** do ROBO Pro em **Programas carregados** o programa Modelo básico2 e por baixo os subprogramas que estão incluídos no programa Modelo básico2. Pode introduzir facilmente estes subprogramas no novo programa.
- Para o subprograma "Busca de luz" utilize o elemento **Ciclo de contagem**. (sobre a descrição do elemento ver manual de instruções ROBO Pro).



Ciclo de contagem

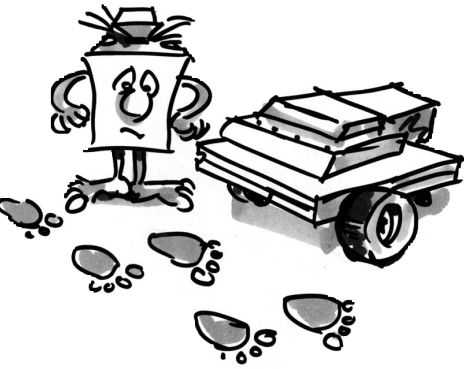
- No ciclo entre a ligação "N" e a ligação "+1" consulte os fototransístores e conte um impulso no botão de pulso I1. O ciclo é percorrido até o robô encontrar luz ou ter realizado uma rotação de 360°. Experimente com que freqüência ele tem de percorrer o ciclo até fazer uma rotação completa e coloque, em função disso, o valor "Z" no elemento Ciclo de contagem.
- Programe um segundo ciclo semelhante para a busca na direção inversa.
- Se o robô encontrar luz, ele pára e abandona o subprograma.
- Aqui está representado o subprograma completo:



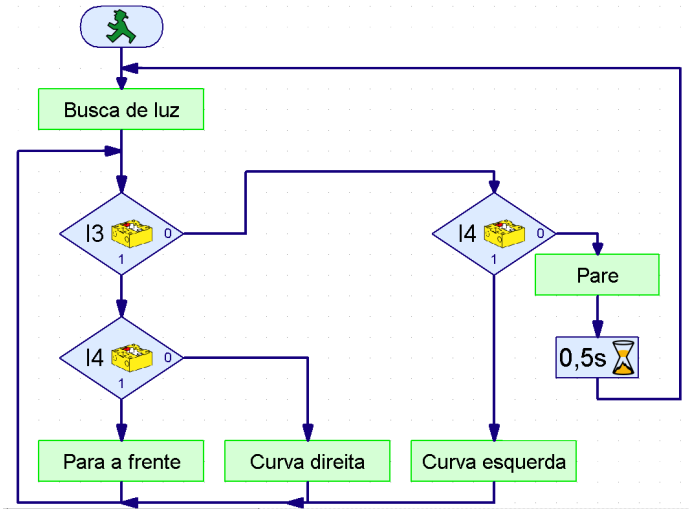
- No programa principal volte a consultar os fototransístores e comande os motores em função de que fototransístor detecta luz:

Luz em I3 e I4	**Para a frente**
Luz apenas em I3	**Curva à direita**
Luz apenas em I4	**Curva à esquerda**
Luz não detectada	**Pára, volta para o subprograma Busca de luz**

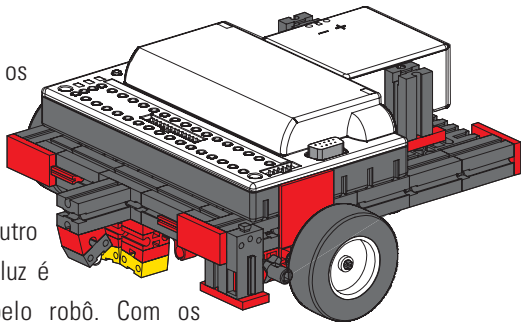
- Crie a curva à direita e à esquerda por meio de diferentes velocidades do M1 e M2 no mesmo sentido de rotação. Com isso se cria um estilo de marcha muito harmonioso.



- O programa principal tem o seguinte aspecto:
- Pode consultar o programa já concluído em Busca de luz.rpp.
- Utilize uma lanterna de bolsa como fonte de luz. Tente focalizar o feixe de luz com intensidade para que ambos os fotosensores sejam iluminados pela fonte de luz. Preste atenção para o fato de em lugares muito iluminados a sua lanterna de bolso poder ser ofuscada por outras fontes de luz, como por exemplo luz solar de uma grande janela. O robô passa ao lado da sua lâmpada em direção à luz mais forte.



O robô que segue uma trilha



■ A busca e seguimento são propriedades essenciais que os seres inteligentes possuem. Com o robô que segue um foco de luz construiu e programou um robô que reagiu a sinais diretos do seu objetivo.

Com o robô que segue uma trilha empregamos um outro princípio de busca. Em vez da marcha para a fonte de luz é marcada uma linha preta que deve ser seguida pelo robô. Com os fototransístores esta tarefa é relativamente fácil de efetuar. A luz da marcação refletida é medida e os motores são corrigidos. Para que funcione corretamente, a linha é iluminada com a lâmpada. Preste atenção para o fato de os fotosensores não serem ofuscados pela luz difusa da lâmpada através de uma disposição desfavorável. Neste contexto a luz da lente óptica da lâmpada incandescente atua de forma especialmente vantajosa.

Construa agora o modelo que segue uma trilha de acordo com as instruções de montagem.

Tarefa 1:

- **Em primeiro lugar, escreva um subprograma com o qual a trilha é procurada. O robô deve rodar uma vez em circuito.**
- **Se não encontrar qualquer trilha, deve se deslocar um pouco para a frente e voltar a procurar em seguida. Para a detecção de trilhas são consultados os fototransístores.**
- **Quando o robô descobrir a trilha, deve segui-la.**
- **Se a trilha terminar ou se o robô a perder, devido, por exemplo, a uma alteração brusca da direção da trilha, a procura deve ser reiniciada.**

Dicas:

- Após ligar a lâmpada tem de se aguardar um instante (aprox. um segundo) antes dos fototransístores serem consultados. Caso contrário, o fototransístor detecta "escuro", ou seja uma trilha, onde não há nenhuma, porque a consulta foi efetuada antes da lâmpada estar corretamente acesa.
- Você pode utilizar aprox. 20mm de fita isolante preta larga como trilha ou pode pintar uma trilha preta dessa largura com caneta de feltro num papel branco. As curvas não devem ser muito acentuadas, caso contrário o robô perde a trilha demasiadas vezes.
- Primeiro, com o teste de interface verifique se a sua trilha foi detectada corretamente pelos fototransístores. Não se esqueça de desligar a lâmpada.
- Ajuste a lâmpada de modo que os dois fototransístores forneçam o valor 1 relativamente ao fundo claro, mesmo quando os motores M1 e M2 são ligados. Se a sua bateria estiver um pouco fraca, ao ligar os motores a lâmpada fica um pouco escura. Caso esta não esteja corretamente ajustada, um fototransístor pode indicar "escuro" apesar de não ter encontrado qualquer trilha.
- A busca de trilha funciona do mesmo modo que a busca de luz. Você apenas tem de adaptar a busca de modo que, em caso de busca malograda, o modelo se desloque um pouco em linha depois de uma rotação completa, antes de continuar a busca.
- Preste atenção para que o modelo tenha de se deslocar em linha reta, em caso de seguimento de trilha, quando os dois fototransístores fornecerem o valor "escuro" (=0).
- Pode consultar o programa já concluído em Busca de trilha.rpp.

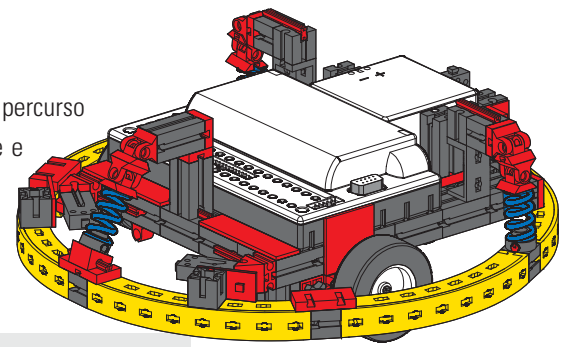
Tarefa 2:

- **Crie uma trilha com várias curvas acentuadas. Qual é o raio que o modelo ainda consegue efetuar?**
- **Ao corrigir a trilha experimente as diferentes velocidades dos M1 e M2. Que combinação proporciona o melhor resultado?**
- **Crie uma trilha com um percurso circular. Procure otimizar as velocidades de modo que o robô consiga efetuar o percurso no menor tempo possível. Esta tarefa é mais adequada para uma competição com mais robôs.**

Robô com detecção de obstáculos

■ Todos os robôs construídos até agora conseguem percorrer uma determinada distância assim como seguir uma fonte de luz ou uma trilha. Mas o que sucede quando se coloca um obstáculo no caminho? Ou o obstáculo é empurrado para o lado ou o robô se desloca contra ele até que a bateria fique vazia. Naturalmente seria muito mais inteligente se o robô detectasse o obstáculo e se desviasse respectivamente. Para isso, o robô contém um pára-choques móvel em volta com três botões. Com este pára-choques ele pode distinguir se existe um obstáculo à esquerda, direita ou atrás.
O modo como ele deve reagir é apenas uma questão de programação.

Primeiro construa o modelo "robô com detecção de obstáculos". Para a medição do percurso é necessário apenas um botão (I1). Por isso, o botão I2 é retirado do modelo base e utilizado para a detecção de obstáculos.



Tarefa 1 (Level 2):

- **Primeiro, o robô deve se deslocar em linha reta. Caso ele bata num obstáculo pela esquerda (I4), ele deve recuar um pouco e, a seguir, desviar para a direita.**
- **Caso ele bata num obstáculo pela direita (I3), ele deve recuar um pouco e, a seguir, desviar para a esquerda.**

Dicas:

- A detecção de obstáculos ao fazer marcha a ré ainda não é tida em conta.
- Os botões são consultados no programa principal. Dependendo de qual botão é acionado, o modelo se desvia para a esquerda ou para a direita. Isto se efetua num subprograma.
- O número de pulsos na rotação à direita se devia diferenciar do número de pulsos na rotação à esquerda (p. ex. 3 pulsos para a direita, 5 pulsos para a esquerda). Caso contrário, o modelo pode ir para um canto e não conseguir descobrir onde está, porque gira sempre a mesma distância para a direita e para a esquerda.
- O programa já concluído se chama Obstáculo1.rpp.

O detetor de obstáculos ainda não consegue fazer duas coisas: Ao fazer marcha a ré não detecta quaisquer obstáculos. Ele também ainda não reconhece quando se encontra um obstáculo diretamente à sua frente. Mas, ele poderia detectar ambos. Se durante a marcha a ré I5 for premido, se encontra um obstáculo atrás do modelo. Se na marcha à frente forem premidos simultaneamente I3 e I4, então se encontra um obstáculo diretamente à frente do modelo. Neste caso, o robô poderia girar em 90°. No total, estão disponíveis as seguintes possibilidades, às quais o robô deve reagir:

Obstáculo	Botão	Reação
à direita	apenas I3	Desviar para a esquerda (rotação de aprox. 30°)
à esquerda	apenas I4	Desviar para a direita (aprox. 45°)
à frente	I3 e I4	Desviar para a esquerda (aprox. 90°)
atrás	I5	É apenas consultado na marcha à ré. Parar e, a seguir, continuar a desviar conforme planejado

Para solucionar esta tarefa facilmente, alguns novos elementos de programa, como p. ex. operadores (p. ex. E, OU), do ROBO Pro Level 3 podem ser uma boa ajuda. No Level 3 também é possível trocar dados entre diferentes elementos através das setas cor de laranja. Para isso, mude para este level no software. A seguir, você deve pegar no manual de instruções ROBO Pro e ler o capítulo 5 cuidadosamente. A seguir, você está preparado para a próxima tarefa.

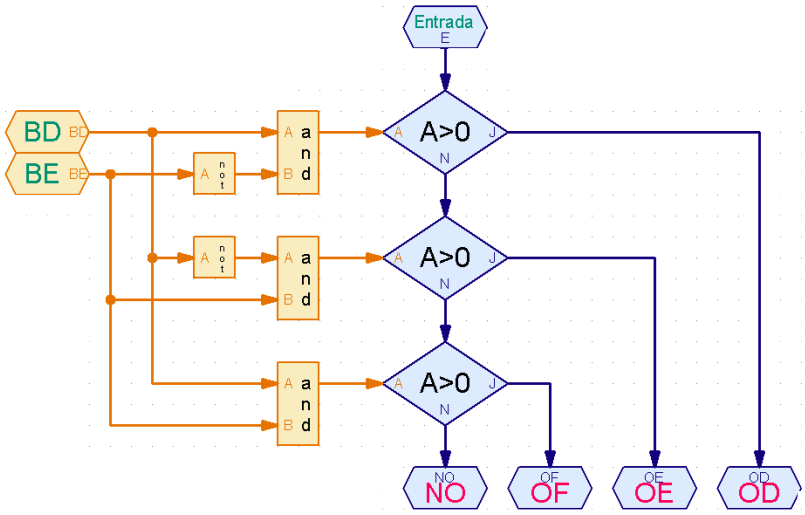
Tarefa 2 (Level 3):

- **Remodele o seu programa de obstáculos de modo que o modelo reaja de acordo com a tabela acima.**
- **Para isso, utilize as possibilidades do ROBO Pro level 3.**

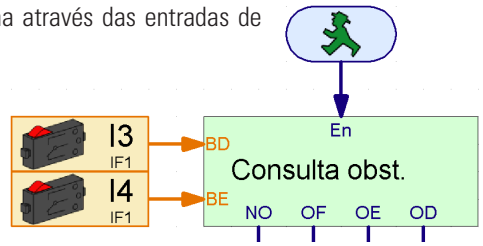
Dicas:

- Com a ajuda dos operadores, as diferentes combinações possíveis de teclas são consultadas num subprograma "Consulta de obstáculos". O subprograma possui uma saída própria para cada possibilidade.

Entrada de dados BD = Botão direito
Entrada de dados BE = Botão esquerdo
Saída NO = Nenhum obstáculo
Saída OF = Obstáculo à frente
Saída OE = Obstáculo à esquerda
Saída OD = Obstáculo à direita

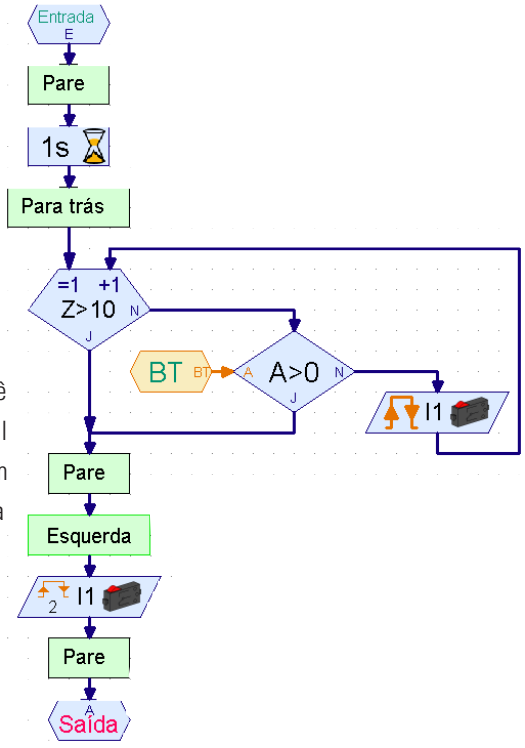


- Para que se detecte de imediato quais botões são consultados, você deve colocar os elementos cor de laranja dos botões no programa principal e os ligar ao subprograma através das entradas de dados.
- Nos vários subprogramas de desvio, I5 é consultado durante a marcha a ré. O modelo se desloca de marcha a ré até que o número de pulsos ajustado seja alcançado ou que I5 seja pressionado. I5 é novamente colocado no programa principal, para que seja visível de imediato em que subprogamas ele é consultado.



- Pode consultar o programa completo em Obstáculo2.rpp

Uma vantagem da técnica de programação utilizada nesta tarefa é que você pode ver diretamente no programa principal que botão é consultado em qual subprograma. Caso pretenda alterar a entrada, você tem de fazer isso num local e não procurar em todos os subprogramas o local onde o botão poderia estar escondido. Além disso, se pode estabelecer uma combinação lógica muito evidente com os operadores. Em principio, isto também pode ser efetuado com elementos de ramificação, mas rapidamente se torna de difícil compreensão quando são consultados vários casos.



Robô que segue um foco de luz com detecção de obstáculos

■ Ainda falta muito para acabarem as possibilidades que o ROBO Mobile Set oferece.
Por isso, agora devem ser combinadas as duas funções: busca de luz e detecção de obstáculos. Do ponto de vista científico o robô está equipado com dois tipos de comportamento. Mas, visto que, os dois modelos de comportamento não podem estar ativos simultaneamente, eles possuem diferentes prioridades. Normalmente o robô responde à busca de luz. Ele detecta um obstáculo, ou seja um perigo para ele, e o comportamento para evitar o obstáculo fica ativo. Caso tudo se encontre na área verde, o robô pode continuar a procurar a fonte de luz.

Se engenheiros profissionais de software se ocuparem com uma tarefa tão exigente, eles não começam a programar à sorte, mas sim utilizam uma determinada estratégia para o desenvolvimento do programa. Um destes métodos é chamado "Rascunho Top-Down". Neste procedimento, todo o sistema é definido a partir do topo, sem se preocupar com todos os detalhes logo no início. Nós também utilizamos este método neste problema.

Tarefa 1 (Level 3):

Programe o robô com o seguinte tipo de comportamento:

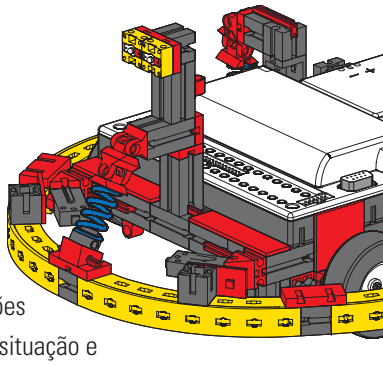
- **Busca de uma fonte de luz.**
- **Logo que a tenha encontrado, siga-a.**
- **Pelo caminho surge um obstáculo, evite-o.**
- **A seguir, volta a buscar uma fonte de luz**

Para a solução, utilize os elementos do programa do ROBO Pro Level 3.
Solucione esta tarefa "a partir de cima" de acordo com o procedimento Top-Down.

Dicas:

Primeiro você divide a tarefa em três partes:

- Consulta se o robô detecta uma fonte de luz (subprograma "Luz")
- Consulta se ele bateu num obstáculo (subprograma "Obstáculo")
- Dependendo destes resultados, você comunica ao robô o que ele deve fazer (subprograma "Deslocamento")



Agora pondere para os subprogramas "Luz" e "Obstáculo", quais situações diferentes o robô pode percepcionar. Você designa um valor numérico a cada situação e o salva na variante com a ajuda de um elemento de comando. De cada situação resulta uma reação, que é executada no subprograma "Deslocamento".

Subprograma Luz:

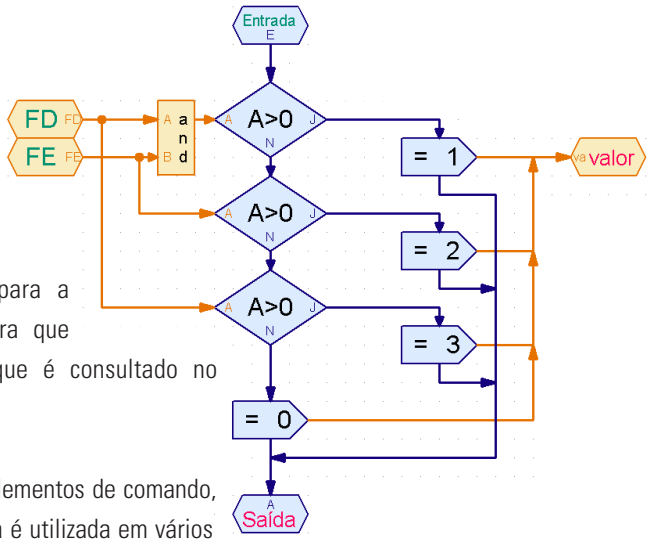
N.º	Situação	Estado dos sensores	Reação
0	Nenhuma fonte de luz disponível	I6=0; I7=0	Buscar luz
1	Fonte de luz à frente do robô	I6=1; I7=1	Deslocar em linha reta
2	Fonte de luz à esquerda do robô	I7=1	Efetuar curva à esquerda
3	Fonte de luz à direita do robô	I6=1	Efetuar curva à direita

Subprograma Obstáculo:

N.º	Situação	Estado dos sensores	Reação
4	Obstáculo diretamente à frente do robô	I3=1; I4=1	Desviar 90°
5	Obstáculo à direita do robô	I3=1	Desviar para a esquerda
6	Obstáculo à esquerda do robô	I4=1	Desviar para a direita

Agora você ainda tem de apresentar estas detecções no ROBO Pro com elementos do programa.

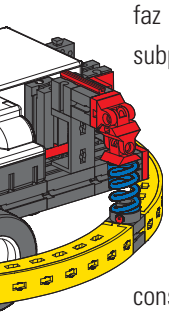
Subprograma Luz:



FD=Fototransístor direito
FE=Fototransístor esquerdo

Volte a colocar os elementos para a consulta dos fototransístores, para que possa detectar de imediato o que é consultado no subprograma.

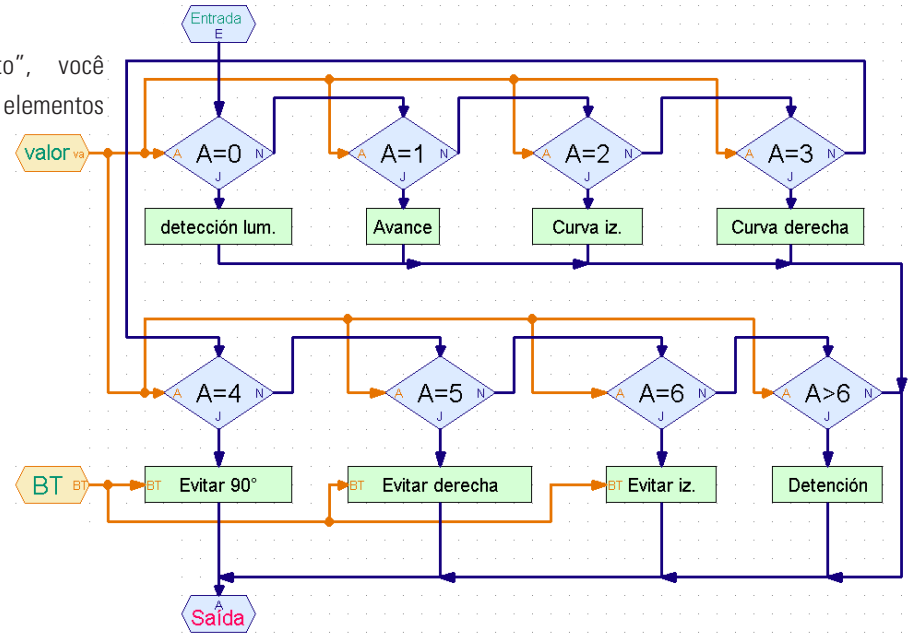
A variante, que salva o valor dos elementos de comando, faz parte do programa principal. Ela é utilizada em vários subprogramas. Ela é conectada ao subprograma através de uma saída de dados.



Você gera o subprograma "Obstáculos" de acordo com o mesmo princípio com o subprograma "Luz".

No subprograma "Deslocamento", você consulta o valor atual da variante com os elementos de ramificação e programa a respetiva reação:

BT=Botão traseiro



Por último, tem de gerar os subprogramas, que são utilizados neste subprograma.
Mas, espere um pouco! Já estão quase todos gerados. O subprograma Busca de luz, p. ex., pode ser copiado do programa para o modelo que segue um foco de luz. Se não souber como isso se efetua, leia o capítulo 4 do manual de instruções ROBO Pro.

Mas cuidado:
No modelo que segue um foco de luz, os fototransístores estavam conectados na entrada I3 e I4. Mas agora estão na I6 e I7. Além disso, o botão I1 é consultado para contar os pulsos quando gira para a esquerda e o botão I2 quando gira para a direita. Agora existe apenas o I1 para contar os pulsos, o que normalmente também funciona bem. Você tem de adaptar o subprograma busca de luz depois de o copiar. Visto que a consulta dos botões está escondida no subprograma, esta passa despercebida com facilidade. Isto não acontece, se você colocar as entradas no programa principal e as conectar com o subprograma através das entradas de dados. Mas esta possibilidade não é possível com o robô que segue um foco de luz.

Os subprogramas para desvio também já estão disponíveis, nomeadamente no modelo com detecção de obstáculos. Neste caso o botão I5, que é consultado adicionalmente ao fazer marcha a ré, está saliente.

Você pode ver o programa concluído em Luz do obstáculo.rpp.
À primeira vista, o programa principal parece muito simples e fácil de compreender. E, no entanto, há muito por detrás dos subprogramas. Mas com a ajuda do procedimento passo-a-passo com o método Top-Down, você pode solucionar um problema com esta complexidade.

Além disso, se um dos seus amigos possuir um módulo ROBO Mobile Set, vocês podem continuar as experiências com este robô. Em cada um dos robôs é montada apenas uma fonte de luz. A seguir, os robôs se buscam mutuamente.

Robô com detecção de bordas

■ Depois de você ter visto no último exemplo como se procede na programação de um problema mais complexo, agora você pode se dedicar a outro comportamento muito importante de um robô móvel. Ele deve aprender a não cair de uma mesa. Se o robô bater num obstáculo, na maioria dos casos, isso não tem qualquer importância. Mas se ele cair de uma mesa com um metro de altura, ele pode sofrer alguns danos, apesar dos módulos da fischertechnik serem muito estáveis. Por esta razão, o robô recebe sensores com os quais ele pode detectar as bordas. Este detetores de bordas são compostos por um botão, que é ativado por uma roda rotativa apoiada. Esta roda pode se movimentar para os dois lados. Logo que a roda ultrapasse a borda da mesa, esta cai, o botão não é mais ativado, o programa detecta que o modelo se encontra num precipício e reage de forma correspondente. O robô tem um total de 4 detetores de bordas de modo que este pode procurar precipícios na marcha à frente e na marcha à ré de ambos os lados. Para isso, este modelo não tem qualquer botão de pulso para a medição do percurso. O caminho percorrido é controlado pela duração da conexão dos motores.

Primeiro, monte o modelo conforme descrito nas instruções de montagem.

Verifique com exatidão se os detetores de bordas são corretamente ativados:

- quando o modelo chega à borda da mesa e o botão for novamente premido exatamente
- quando a roda volta para cima da mesa.

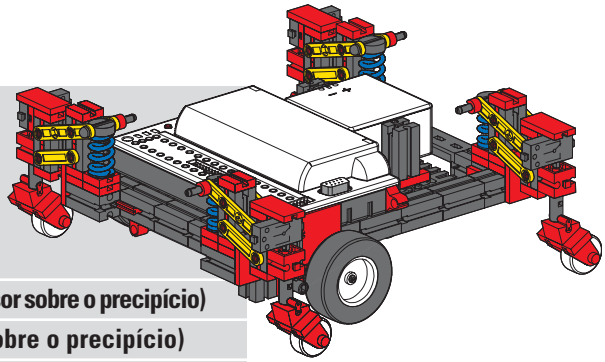
Eventualmente, você tem de ajustar um ou outro botão um pouco para cima ou para baixo.

Tarefa 1 (Level 3):

- **Primeiro, pondere sobre o modo como o robô deve reagir, quando ele chega a um precipício.**
- **Se pensar cuidadosamente, lhe irá ocorrer que existem muitas possibilidades de combinação relativamente a quais sensores se podem encontrar no precipício. A ativação pode ocorrer em 1, em 2 ou 3 sensores diferentes simultaneamente ou até em 4 sensores.**
- **Como é que o robô deve reagir em cada uma das vezes?**

Dicas:

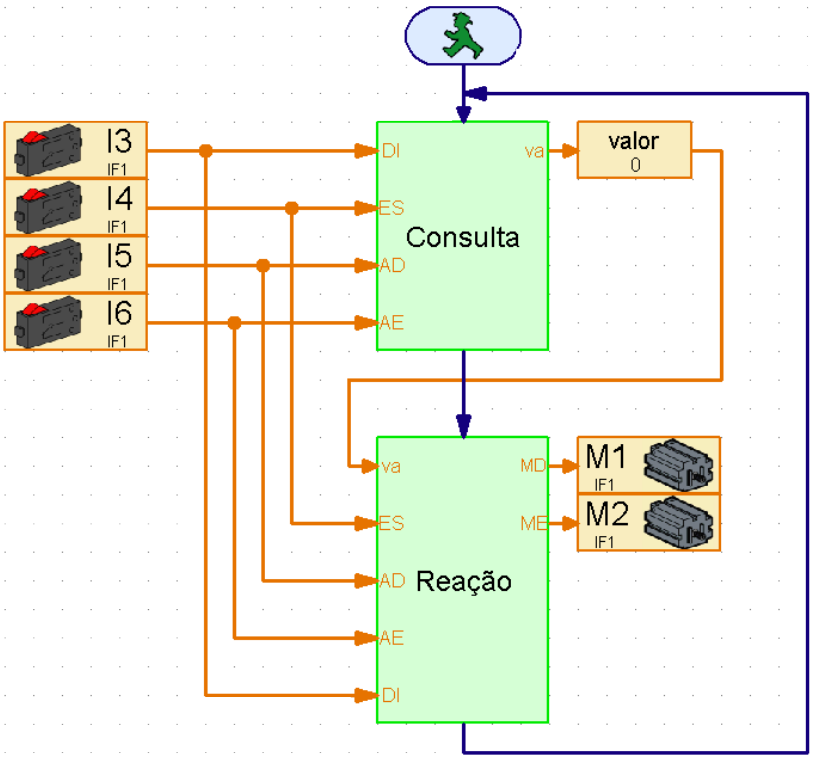
Pode encontrar a solução na tabela seguinte. Os sensores, que se encontrarem sobre o precipício (botão=0) estão marcados com uma cruz. Cada combinação contém um número. No programa, que é gerado mais tarde, é atribuído o número respetivo a cada possibilidade. De acordo com o número, o robô reage à situação atual. Mas isso será mais tarde. Primeiro, reflita apenas sobre o lugar onde o robô deve estar para que a combinação se efetue e verifique se ele reage.



N.º	Frente direita (I3)	Frente esquerda (I4)	Atrás direita (I5)	Atrás esquerda (I6)	Reação
0					Para a frente (Nenhum sensor sobre o precipício)
1	●	●	●	●	Parada (os 4 sensores sobre o precipício)
2	●	●	●		Girar um pouco para a direita
3	●	●		●	Girar um pouco para a esquerda
4	●		●	●	Girar um pouco para a esquerda
5		●	●	●	Girar um pouco para a direita
6	●	●			Primeiro para trás, depois gira para a direita
7	●		●		Girar um pouco para a esquerda
8	●			●	Girar um pouco para a esquerda
9		●	●		Girar um pouco para a direita
10		●		●	Girar um pouco para a direita
11			●	●	Se desloca um pouco para a frente
12	●				Primeiro para trás, depois gira para a esquerda
13		●			Primeiro para trás, depois gira para a direita
14			●		Se desloca um pouco para a frente
15				●	Se desloca um pouco para a frente

Bastante difícil, não? Mas, não se preocupe, existe um programa já concluído para este modelo, que utiliza todas as vantagens do ROBO Pro. É o Cantos.rpp.

Os elementos mais importantes se encontram no programa principal de modo que você possa compreender todo o decurso. A consulta complexa do botão e a ativação dos motores estão escondidas nos subprogramas. Primeiro o programa principal:



O decurso se inicia com a consulta dos 4 botões. À esquerda você pode ver qual botão é consultado. Eles estão conectados ao subprograma através das entradas de dados. O subprograma "Consulta" determina qual botão é pressionado e emite o valor descrito na tabela. Este valor é atribuído às variantes com o mesmo nome, que você pode reconhecer no programa principal. O valor da variante é transmitido ao subprograma "Reação", que, a seguir, ativa um dos dois motores dependendo deste valor. No subprograma "Reação" também são lidos os botões, visto que os sensores de bordas são consultados, enquanto o modelo se desvia.

Agora você pode alterar a ocupação dos botões na interface, assim como as saídas do motor, sem ter de examinar exaustivamente em todos os subprogramas o local onde poderia estar escondido um elemento de entrada ou um símbolo do motor. Cada entrada e cada saída só surge uma vez.

Você pode utilizar esta técnica de programação principalmente quando um subprograma tiver de ser utilizado em muitos modelos e você antes não souber exatamente que entradas e saídas da interface devem ser utilizadas.

Se você tiver ficado curioso, pode dar uma olhada nos subprogramas e tente compreender. O princípio da programação é semelhante ao do modelo "robô que segue um foco de luz com detecção de obstáculos"

Tarefa 2 (Level 3):

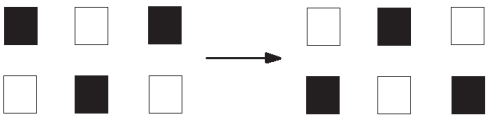
Carregue o programa na interface e coloque o modelo a se deslocar numa mesa.

- **O modelo reage sempre de modo correto?**
- **Ele deveria responder de outro modo a determinadas combinações de botões?**
- **Se necessário, otimize o programa.**

O robô andante

■ Depois de nos termos ocupado detalhadamente com os robôs rolantes, debrucemo-nos agora sobre um outro tipo de movimentação, que podemos utilizar para robôs móveis, o andar.
A forma de caminhar dos insetos se apropria de modo excelente como modelo para o acionamento de "objetos com seis pernas mecânicas". Na chamada "marcha com três pés" se elevam simultaneamente do chão sempre três das seis pernas, a perna dianteira e traseira de um dos lados, juntamente com a perna central do outro lado:

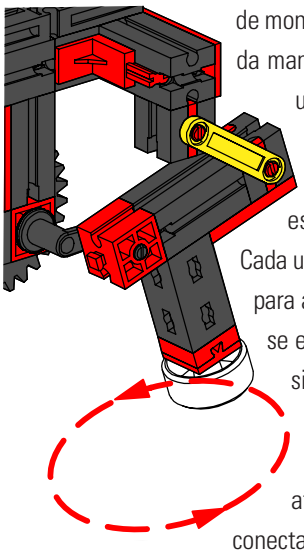
Marcha com três pés



As pernas que permanecem no chão (apresentadas a preto), formam um objeto de três pernas estável, de modo a que o modelo se mantenha sempre estável e não vire ao andar.

As pernas do robô andante da fischertechnik são construídas de tal modo, que resultam numa articulação de quatro membros. A forma de construção dos quatro membros aqui utilizada é denominada de "Mecanismo corrediça-manivela oscilante". Acionados por uma manivela, os membros munidos de movimento da articulação efetuam movimentos oscilantes. As distâncias entre os membros individuais e o local da extremidade do pé (isso é a extremidade inferior do pé) estão escolhidas de tal forma, que a extremidade do pé descreve um movimento elíptico, quando a manivela de acionamento roda. Através disso é originado um movimento, que se assemelha a um passo ao caminhar.

As 6 manivelas, que acionam as pernas têm de ser ajustadas exatamente como indicado nas instruções de montagem. As três pernas que assentam simultaneamente no chão, têm a mesma posição da manivela. As manivelas das 3 pernas, que neste momento se mantêm elevadas, sofrem uma rotação de 180°. A posição correta das manivelas entre si garante que o modelo se possa deslocar numa seqüência de passos correta, a "marcha com três pés".



As porcas de cubo, com as quais as rodas dentadas são fixadas nos eixos, têm de estar bem apertadas, de modo a que as manivelas não saiam do lugar durante o andar. Cada um dos lados direito e esquerdo do modelo são acionados por um motor (isso é necessário para andar em curvas). Por isso deve-se ter em atenção, que a perna central de um dos lados se encontre sempre na mesma posição que ambas as pernas exteriores do outro lado. Esta sincronização é efetuada através de controle por software, através dos botões I1 e I2.

Primeiro, monte o modelo conforme descrito nas instruções de montagem. Controle, através do teste de interface, se todos os botões e motores se encontram corretamente conectados. Sentido de rotação dos motores: Sentido de rotação para esquerda=para a frente.

Tarefa 1 (Level 1):
Programe o andar do robô:

- **Programe o modelo de forma que este se desloque na marcha com três pés, para a frente.**
- **Utilize os botões I1 e I2 para a sincronização das pernas esquerdas e direitas.**
- **Tenha em atenção que ambas as pernas exteriores de um dos lados e a perna central do lado oposto têm de ter sempre a mesma posição.**

Dicas:

- Coloque primeiro as pernas dos lados esquerdo e direito na sua posição de saída. Ligue ambos os motores (sentido de rotação à esquerda).
- O processo só deve continuar quando ambos os botões I1 e I2 não estão premidos (esta consulta é necessária, desde que o modelo tenha de fazer o segundo passo).
- Deixe os motores trabalhar, até o respectivo botão (I1 para M1, I2 para M2) estar novamente premido. Aí é importante que o modelo só inicie o próximo passo quando ambos os botões estão premidos. Aí as pernas estão numa posição correta entre si. A condição para tal é também que as manivelas que acionam as pernas estejam ajustadas corretamente como indicado nas instruções de montagem.
- Agora o processo pode começar pela frente e o robô dá o segundo passo. Agora o modelo anda para a frente, até que pare o programa.
- Pode consultar o programa já concluído em Robô andante1.rpp.

Pode agora, de forma semelhante ao modelo básico rolante, deixar o modelo andar para a esquerda, direita ou para trás, alterando os sentidos de rotação do motor. Para contar os passos, pode utilizar I1 ou I2.

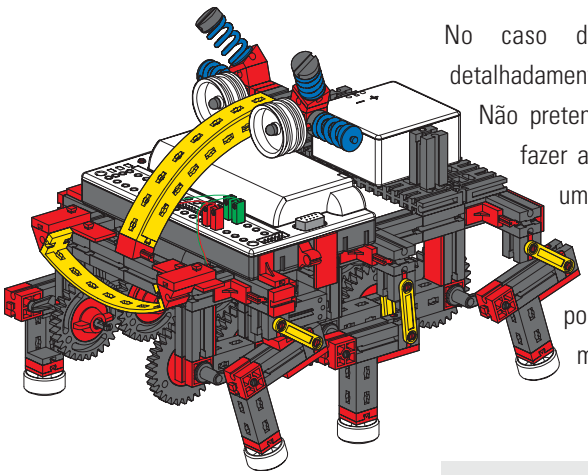
Tarefa 2 (Level 2):

- **Programe o seu modelo de forma que este ande 10 passos para a frente, 3 passos para a esquerda, 3 passos para a direita e novamente 10 passos para trás.**
- **Aplique para cada sentido um subprograma próprio.**
- **Para a contagem dos passos, utilize o elemento Ciclo de contagem.**

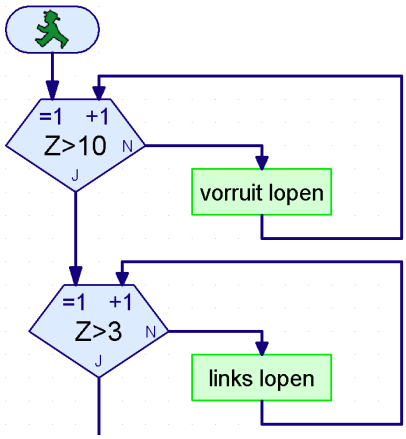
Dicas:

- Copie simplesmente o programa Robô andante1.rpp para um subprograma.
- Copie este subprograma as vezes necessárias para os diferentes sentidos de marcha. Altere em cada subprograma os sentidos de rotação de forma que o modelo se movimente no sentido pretendido.
- Utilize o elemento Ciclo de contagem para contar o número de passos para cada sentido de rotação. O modelo dá um passo em cada execução de um subprograma. Se o programa executa o ciclo com o subprograma 10 vezes, o modelo dá 10 passos.

Desta forma, pode programar o seu robô andante com uma seqüência de passos desejada (robô andante 2.rpp).



No caso dos robôs rolantes, abordamos detalhadamente o tema detecção de obstáculos. Não pretendemos nos repetir aqui. Mas tente fazer a transferência deste comportamento uma vez para os robôs andantes. Os sensores para isso estão incluídos no módulo. Durante a programação pode tomar os robôs rolantes como modelo. Bom trabalho!

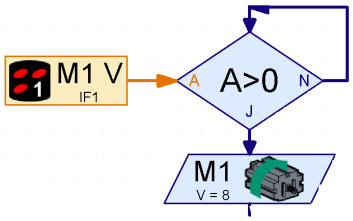


Possibilidades de expansão

■ O ROBO Interface oferece mais funcionalidade do que a apresentada até ao momento pelos robôs móveis. Para isso são necessários, no entanto, componentes adicionais, que não se encontram incluídos no volume de fornecimento do módulo. Uma vez que estes são também extremamente interessantes para os robôs, gostaríamos de apresentar aqui alguns deles.

Emissor manual de infravermelhos

■ No ROBO Interface está incluído um diodo receptor de infra-vermelhos do IR Control Set, n.º de art. 30344, para o emissor manual. No software ROBO Pro pode consultar, desse modo, as teclas do emissor manual, como entradas digitais e, assim, p. ex. ligar e desligar motores.



Como exemplo do programa programamos um comando à distância para robô andante. Com as 4 teclas de setas ovais no controle remoto pode comandar o modelo para a frente. Para trás, para a esquerda e para a direita. Só necessita de carregar previamente o programa Robô andante-IR.rpp na interface.

Um outro programa genial em conjunto com o controle remoto é o programa Mobile-Teach-IR.rpp. Através deste programa Teach-In pode comandar à distância um dos robôs rolantes, p. ex. o robô simples ou o modelo básico. O modelo memoriza o percurso percorrido e é capaz de o repetir mais tarde, sempre que desejado. O percurso memorizado é contudo apagado, quando o programa para.
Um programa destes se torna possível através do elemento do programa "Liste" no ROBO Pro. Neste elemento, é possível memorizar e voltar a ler vários valores (ver também o manual de instruções do ROBO Pro). O próprio programa é extremamente complexo, mas a utilização é bastante fácil:

1. Carregar e iniciar o programa Mobile-Teach-In.rpp na memória Flash da interface ROBO.
2. Premir a tecla **M1** ▸/▸▸ no controle remoto. O "processo de programação" é iniciado.
3. Comandar o modelo na direção pretendida, através das teclas de setas ovais.
4. Premir a tecla **M2** ▸/▸▸ O percurso percorrido é memorizado.
5. Premir a tecla **M3** ▸/▸▸. O percurso memorizado é percorrido.

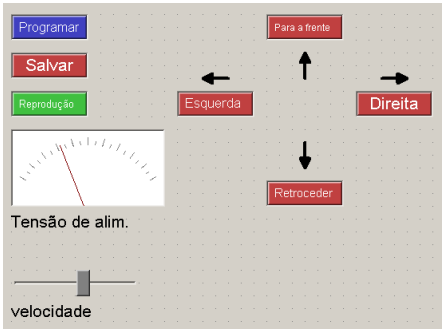
Com uma aplicação deste gênero, a programação de robôs se transforma numa brincadeira de crianças! Tem de ter em atenção, que o percurso memorizado é eliminado, assim que pára o programa através do botão Prog. na interface.

ROBO RF Data Link

■ A interface aérea ROBO RF Data Link, n.º de art. 93295 substitui o cabo da interface entre PC e interface através de uma transmissão de dados sem fios. Isso é ótimo. Primeiro não é necessário conectar e voltar a desconectar o cabo toda vez que carrega um programa na interface. E em segundo lugar, pode operar programas sem cabos, no modo online, e, deste modo, detectar erros muito mais facilmente do que no modo download. E, finalmente, é possível comandar os robôs móveis no modo online, através de um painel de comando no ROBO Pro, de modo semelhante ao controle remoto IR online, através da tela. Ao contrário do controle remoto, é possível ainda visualizar adicionalmente na tela dados, que a interface fornece, p. ex. valores de variáveis ou entradas analógicas, tensão de alimentação, que a bateria fornece, velocidade dos motores.

Como exemplo, reconstruímos o programa Teach-In e comandamos o robô rolante através de um painel de comando. O programa se chama Mobile-Teach-RF.rpp. Também o pode experimentar através do cabo de interface. Isso é, no entanto, extremamente desconfortável. O raio de ação do modelo fica limitado, o cabo se enrola e o robô deixa de conseguir virar corretamente. Com certeza que vai querer adquirir imediatamente o RF Data Link.

Carregue o programa Mobile-Teach-RF.rpp.
Mude na barra de funções do programa principal para painel de comando. A seguir inicie o programa no modo online. Agora pode comandar e programar o modelo através dos botões no painel de comando.



1. Premir a tecla "Programar" O "processo de programação" é iniciado.
2. Comandar o modelo na direção pretendida, através das teclas de setas.
3. Premir a tecla "Salvar" O percurso percorrido é memorizado.
4. Premir a tecla "Reprodução" O percurso memorizado é percorrido.

O percurso memorizado também se perde aqui, quando o programa termina.

Obtém mais informações sobre a criação de painéis de comando no manual de instruções do ROBO Pro.

ROBO I/O-Extension

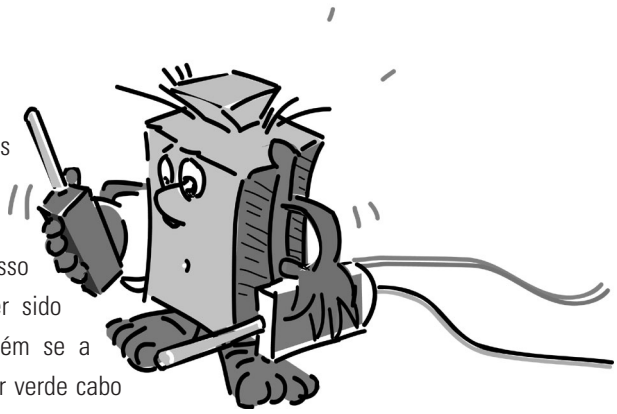
■ Caso construa um modelo com tantos sensores e motores, que as entradas e saídas da interface ROBO não sejam suficientes, pode conectar uma ROBO I/O-Extension, n.º de art. 93294 à interface. Desse modo, ficam disponíveis mais 8 entradas digitais, 4 saídas de motor e uma entrada de resistência analógica. Nesta extensão I/O pode conectar um segundo módulo, e conectar no segundo, um terceiro módulo, que serão depois todos comandados por uma ROBO Interface. Ficam então disponíveis, na totalidade, 16 saídas de motor, 32 entradas digitais, 5 entradas de resistência analógicas, 2 entradas de tensão analógicas, bem como 2 entradas para sensores de distância.

Se ainda não for suficiente, pode ainda comandar várias interfaces no PC, no Modo online, p. ex. uma numa interface serial COM, uma na porta USB ou 2 interfaces na porta USB, cada uma com até 3 ROBO I/O-Extensions! Isso pode se tornar muito confuso. O funcionamento de tudo isto se encontra descrito no capítulo 6 do manual de instruções ROBO Pro.

Busca de falhas

■ Experimentar é divertido. Mas somente desde que tudo funcione. Na maioria das vezes isso acontece. Mas, infelizmente, nem sempre.
Quando um modelo não funciona, se questiona em primeiro lugar, se o mecanismo foi entendido correta e completamente e se a falha é imediatamente encontrada.
Em caso de falhas mecânicas, é possível ainda ver (montagem incorreta) ou sentir (falta de mobilidade) mais alguma coisa. Se surgirem ainda problemas elétricos, é mais complicado.
Os profissionais utilizam uma série de diferentes instrumentos de medição para a busca de falhas, como p. ex. medidor de tensão ou oscilógrafo. Nem todos têm acesso a este tipo de aparelhos. Pretendemos por isso tentar isolar e eliminar uma falha com meios mais simples.

Montagem dos cabos



Antes de começarmos com as nossas experiências, temos de terminar ainda alguns componentes dos módulos da Fischertechnik. P. ex. os conectores fornecidos devem ser conectados ás respectivas seções de cabos.
Primeiro o cabo é cortado. Medimos para isso os comprimentos indicados e cortamos as seções. Cada cabo é medido após estar completo. Para isso necessitamos da bateria e da lâmpada. Se a lâmpada acender, após ter sido conectada com a bateria, o cabo está em condições. Verificamos também se a atribuição de cores está correta, conector vermelho cabo vermelho, conector verde cabo verde.

Teste de interface

Se o programa (também o fornecido) não trabalhar juntamente com o nosso modelo, iniciamos o teste de interface. Este programa de ajuda permite-nos testar as entradas e saídas separadamente. Os sensores funcionam? Os motores rodam na direção correta? Os motores estão, em todos os nossos robôs móveis, ligados de tal forma que, em caso de sentido de rotação=esquerda, a roda ou a perna se movimenta para a frente. Se aqui se encontrar tudo em condições, procuramos a causa mecânica.

Maus contatos

Maus contatos são erros freqüentes. Por um lado, os conectores adaptadores podem estar soltos no plugue. Se for esse o caso, as molas do contato do conector são alargadas com uma pequena chave de parafusos. Cuidado, alargar demasiado provoca a quebra dos contatos ou conduz a uma falta de mobilidade ao encaixar.
Outra causa para os maus contatos são os acoplamentos de aperto soltos, nos locais de aparafusamento do conector. Por favor aparafusar cuidadosamente! Na ocasião é igualmente verificado, se nenhum dos fiozinhos de cobre finos estão partidos.

Curto-circuitos

Também poderia acontecer que seja provocado um curto-circuito devido à colocação errada dos cabos. Nesse caso, tudo deixa de funcionar como devia. Na bateria está integrado um fusível, que, em caso de corrente ou temperatura demasiado elevada, desliga a corrente. As saídas da interface são também desligadas, em caso de sobreaquecimento.
Pode também ser provocado curto-circuito, se não se apertar corretamente os pequenos parafusos nas fichas elétricas, com os quais o cabo é apertado. Estes ficam eventualmente a sobressair do bordo da

ficha. Se forem introduzidas duas fichas em dois plugues que se encontram diretamente ao lado um do outro na interface, de tal modo que dois dos pequenos parafusos se toquem, acontece um curto-circuito. Por isso, os pequenos parafusos devem ser sempre bem apertados e a ficha deve ser inserida de tal forma, que os parafusos não possam entrar em contato uns com os outros.

Alimentação de corrente

Se acontecerem falhas inexplicáveis durante o funcionamento, a causa pode ser a bateria vazia. A tensão desce brevemente ao conectar uma carga (motor ligado) e é ativado um reinicializar para o processador na interface. Ao acender o LED vermelho, a interface ROBO avisa quando a tensão da alimentação de corrente é demasiado reduzida. A bateria necessita então de ser carregada.

Erro de programação

Se, nos programas por si elaborados, surgirem erros que não são possíveis de explicar, deve, por razões de segurança, ser executado um programa o mais parecido possível com os fornecidos, de modo a que seja possível excluir avarias elétricas ou mecânicas. No Modo online é possível seguir o fluxo do programa na tela. Se o programa chegar a um determinado ponto e não continuar, é possível procurar aqui a causa, p. ex. entrada ou motor incorretos selecionados, valor incorreto consultado numa ramificação ou ligações J/N trocadas.

Se nada disto tiver sucesso, resta contatar o serviço de assistência fischertechnik (e-mail: info@fischertechnik.de).

Ou então visite o nosso site na Internet, em www.fischertechnik.de. Lá tem acesso a um fórum, chat, mercado, galeria e pode se tornar membro do Fischertechnik Fanclub gratuitamente.

Desejamos a você ainda muitas horas de divertimento com o ROBO-Mobile-Set com vários efeitos Aha.

fischertechnik COMPUTING